



华章 IT

**热销图书《Spark 原理、机制及应用》的续作，本书理论结合实践，以通熟易懂的** **方式深入解析大数据治理与安全在学术界和工业界的应用场景与重要作用，是不可多** **得的技术实践类前沿佳作。**



**Big Data Governance and Security**

**From Theory to Implementation**

**大数据治理与安全**

**从理论到开源实践**

**刘驰胡柏青谢一** **◎等编著**

机 械 工 业 出 版 社

China Machine Press



**技术丛书**



技术丛书

**Big Data Governance and Security**

From Theory to Implementation

**大数据治理与安全**

**从理论到开源实践**

刘 驰 胡 柏 青 谢 一 施盟捷陈喆毓林秋霞

◎编著





图书在版编目(CIP) 数 据

大数据治理与安全：从理论到开源实践/刘驰等编著.一北京：机械工业出版社，2017.8 (大数据技术丛书)

ISBN 978-7-111-57997-7

I. 大 … IⅡ. 刘 … ⅢI. 数据处理 IV.TP274

中国版本图书馆 CIP 数据核字(2017)第222527号

本书主要从理论和实践两个部分对大数据治理与安全技术展开详尽描述。其中理论篇主要从大数据治理的概念、 作用、重要性，以及大数据治理的原则、范围及评估内容做出了详细介绍；之后从大数据安全、隐私和审计三个方面， 探讨了大数据安全所面临的挑战，以及解决这些问题的技术与方案、作用与意义。开源实践篇分别从Apache 的四个 开源组件 Falcon 、Atlas 、Ranger 和Sentry 以及 Kerberos 软件框架与工具介绍其在大数据治理与安全方面的功 能与实践应用方案。

本书适用于大数据应用技术爱好者以及具有一定开发经验的读者，也可以作为大数据相关课程的教学参考书，供 云计算、大数据相关专业方向的本科生、研究生阅读，亦可作为相关从业人员与一线软件开发人员的参考资料。

大数据治理与安全：从理论到开源实践

出版发行：机械工业出版社(北京市西城区百万庄大街22号 邮政编码：100037)

责任编辑：陈佳媛 责任校对：李秋荣

印 刷 ：北京诚信伟业印刷有限公司 版 次：2017年9月第1版第1次印刷

开 本： 186mm×240mm 1/16 印 张：24.75

书 号： ISBN 978-7-111-57997-7 定 价：79.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换 客服热线：(010)88379426 88361066

购书热线：(010)68326294 88379649 68995259

投稿热线：(010)88379604

读者信箱：hzit@hzbook.com

版权所有·侵权必究

封底无防伪标均为盗版

本书法律顾问：北京大成律师事务所 韩光/邹晓东

 **前** **言**

在大数据时代，随着信息量与日俱增，数据价值也得到越来越多人的认可。但大数据 在迅猛发展的同时也带来不少问题，如怎样管理数据、实现数据价值最大化等，这些问题 始终未得到完美的解答。在不同时间段，针对不同业务需求，数据的价值也不尽相同。为 了最大化大数据的价值，互联网数据共享不可避免。然而，由于各个企业和部门之间相互 独立，数据所在的系统甚至数据存储结构存在较大差异，数据之间难以进行信息共享，从 而造成信息孤岛这一普遍现象。同时，互联网庞大的使用群体，也使得互联网数据在实现 共享时，难以保障数据的安全性以及数据隐私。

为了解决这些问题，大数据治理与安全成为当下学术界与工业界最热门的研究领域之 一。大数据治理主要在于建立一个统一标准化平台，从不同数据源中获取数据，在对数据 进行生命周期管理的同时允许各方对数据进行相应操作(例如数据审计、数据筛选以及数据 迁移等),从而实现数据价值最大化。而在数据业务流程中，这个统一标准化平台能够针对 不同用户，根据不同的时间点以及IP 地址，对不同的元数据进行权限设置，以保证数据使 用的安全性。

本书总体分为两部分。第一篇：理论篇，包括第1章和第2章。第1章从大数据治理 的概念以及作用两方面，阐述大数据治理的重要性，并对大数据治理的原则、范围及评估 内容做了详尽介绍。第2章从大数据安全、隐私和审计三个方面出发，探讨了大数据安全 所面临的挑战与问题，以及解决这些问题的技术与方案。

第二篇：开源实现篇，包括第3～7章。作者对开源社区中的大数据治理与安全相关 的开源项目做了充分的介绍和实践，将内容根据不同组件分类，汇总成为该篇的主要内容。 该篇全面介绍了 Apache Falcon 、Apache Atlas 、Apache Ranger 、Apache Sentry 与 Kerberos 等大数据治理与安全开源组件的技术概况、配置与使用、场景设计与实现以及具体应用举 例等多方面的内容。

第3章深入介绍建立在 Hadoop 环境下的数据过程及数据集管理系统 Apache Falcon 的

IV

技术概况与架构特点。在此基础上，对集群上进行数据保留、生命周期管理、数据血统及 追踪等功能进行介绍。并且设计与实现了日常生产环境中可能用到的数据处理场景，可作 为相关从业者的参考。最后作者举例说明了Falcon 在数据流程管理领域的使用前景。

第4章全面介绍元数据管理框架 Apache Atlas 的技术概况、配置使用与具体使用场 景等核心内容。本章首先介绍 Apache Atlas在元数据管理方面的突出优势，进而对Hive、 Sqoop 、Storm 及 Falcon 等多种元数据导入方式进行了介绍，并对元数据的管理做了十分深 入的阐述。在此基础上，对Atlas 的实时数据、非实时数据等元数据管理场景进行了设计与 实现，可以作为类似场景下构建与使用的参考。

第5章讲述安全认证框架 Apache Ranger 的技术概况、发展近况、插件集成和功能验 证等内容。本章首先介绍 Apache Ranger 在 Hadoop 生态系统中实施安全认证的优势和特 点，并对 Hadoop 生态组件如 HDFS 、Hive 、HBase 等如何进行安全数据访问控制做出详 细阐述。最后给出了 Ranger 四种不同策略的实际场景，对其安全功能进行了验证。

第6章对Cloudera 公司发布的高度模块化的权限管理组件Apache Sentry 做了深入的 介绍，弥补了Hadoop 文件系统 HDFS 缺乏对数据和元数据细粒度权限访问支持的问题。从 Sentry 的特点、优势、发展近况三个方面，对其架构中的 Binding 、Policy Engine 和 Policy Provider三大核心组件进行了详细的阐述。并介绍了Sentry 的搭建与部署步骤，以及其与 Impala 的集成步骤和在各类场景下 Sentry 的设计与使用方法。

第7章除了对网络认证协议Kerberos 的特点与组成、架构与应用等做了介绍以外，还 对大数据应用下的诸多组件与Kerberos 的集成做了详细的实践介绍，包括HDFS 、Yarn、 Zookeeper 、Hive 、HBase 、Sqoop 、Hue 、Spark 、Solr 、Kafka 、Storm 与 Impala, 几乎涵盖 了大部分学术界与工业界所涉及的各类组件，能够为高校科研人员与企业开发人员提供有 效的参考与帮助。

作者认为大数据治理与安全理论部分已经有一些书籍进行了较好的阐述，而实践应用 部分却十分匮乏。因此本书着重在实践部分使用大量篇幅进行详细的讲解描述。若读者想 要查阅大数据治理与安全的相关理论内容，作者推荐桑尼尔·索雷斯的《大数据治理》和 张邵华的《大数据治理与服务》两本书作为进一步的参考。

本书的作者除了封面和内封提到的六位之外，还有王文杰、段雄、吴琪、方久鑫、 童楚云、陈超源、徐杰、陈喆、吴岳秋、吴成、张晶。

大数据发展迅速，而大数据治理与安全作为其分支，发展更是日新月异。由于作者水 平有限，书中难免有不足与谬误之处，若读者发现问题并不吝告知，不胜感激。

本书讲述的相关组件，请读者到 www.bitlinc.cn 进行下载。

刘驰 lincbit@gmail.com

**entents 目** **录**

前 言

**第一篇** **理论篇**

**第** **1** **章** **大** **数** **据** **治** **理** **技** **术** [2](#bookmark1)

1.1 概 述 [2](#bookmark2)

1.1.1 大数据治理的基本概念 [2](#bookmark3)

1.1.2 大数据治理的意义和重要作用 5

1.2 框 架 [7](#bookmark4)

1.2.1 大数据治理框架概述 [7](#bookmark5)

1.2.2 大数据治理的原则 [9](#bookmark6)

1.2.3 大数据治理的范围 [11](#bookmark7)

1.2.4 大数据治理的实施与评估 [14](#bookmark8)

**第2章** **大数据安全、隐私保护和**

**审** **计** **技** **术** [19](#bookmark9)

2.1 大数据安全 [19](#bookmark10)

2.1.1 大数据安全的意义和重要作用 [19](#bookmark11)

2.1.2 大数据安全面临的问题与挑战 [21](#bookmark12)

2.1.3 大数据安全防护技术 [23](#bookmark13)

2.2 大数据隐私保护 [26](#bookmark14)

2.2.1 大数据隐私保护的意义和重要

作 用 [26](#bookmark15)

2.2.2 大数据隐私保护面临的问题与

挑战 [28](#bookmark16)

2.2.3 大数据隐私保护技术 [31](#bookmark17)

2.3 大数据治理审计 [34](#bookmark18)

2.3.1 大数据治理审计概述 [34](#bookmark19)

2.3.2 大数据治理审计内容 [37](#bookmark20)

2.3.3 大数据治理审计方法和技术 [39](#bookmark21)

2.3.4 大数据治理审计流程 [43](#bookmark22)

**第二篇** **开源实现篇**

**第3章** **大数据治理之** **Apache**

Falcon [48](#bookmark23)

3.1 Apache Falcon 概述 [48](#bookmark24)

3.1.1 Apache Falcon 技术概况 [49](#bookmark25)

3.1.2 Apache Falcon 发展近况 [50](#bookmark26)

3.1.3 Apache Falcon 技术优势 [50](#bookmark27)

3.1.4 Apache Falcon 架构 [51](#bookmark28)

3.2 Apache Falcon 的使用 [53](#bookmark29)

3.2.1 Oozie 的安装与配置 [56](#bookmark30)

3.2.2 Falcon 的安装与配置 [61](#bookmark31)

VI

3.2.3 实体XML 的创建与声明 [63](#bookmark32)

3.3 Apache Falcon 场景设计与实现 74

3.3.1 数据管道 [74](#bookmark33)

3.3.2 结构化数据导入分布式文件

系统 [82](#bookmark34)

3.3.3 结构化数据库与数据仓库的

交互 [89](#bookmark35)

3.3.4 跨集群数据传输 [104](#bookmark36)

3.3.5 数据镜像 [109](#bookmark37)

3.3.6 数据仓库中的数据操作 [113](#bookmark38)

3.4 Apache Falcon 优化与性能分析 [118](#bookmark39)

3.4.1 Apache Falcon 控制流 [118](#bookmark40)

3.4.2 分布式部署 [119](#bookmark41)

3.4.3 安全模式 [120](#bookmark42)

3.4.4 Apache Falcon 优化 [122](#bookmark43)

3.5 Apache Falcon 应 用 举 例 [123](#bookmark44)

3.5.1 InMobi 基于 Falcon 的数据

治 理 [123](#bookmark45)

3.5.2 Expedia 基于 Falcon 的数据

治理 [125](#bookmark46)

3.6 本 章 小 结 [126](#bookmark47)

**第** **4** **章** **大** **数** **据** **治** **理** **之** **Apache**

**Atlas** [127](#bookmark48)

4.1 Apache Atlas 概述 [127](#bookmark49)

4.1.1 Apache Atlas 技术概况 [127](#bookmark50)

4.1.2 Apache Atlas 发展近况 [130](#bookmark51)

4.1.3 Apache Atlas 技术优势 [133](#bookmark52)

4.1.4 Apache Atlas 架 构 [136](#bookmark53)

4.2 Apache Atlas 的配置与使用 [143](#bookmark54)

4.2.1 安装配置 Apache Atlas [143](#bookmark55)

4.2.2 添加或修改 Atlas Web UI 的

登录账户 [158](#bookmark56)

4.2.3 配置 Hive 通过 Hive HOOK

导入数据 [159](#bookmark57)

4.2.4 配置 Sqoop 通过 Sqoop HOOK

导入数据 [163](#bookmark58)

4.2.5 配置Storm 通过 Storm HOOK

导入数据 [167](#bookmark59)

4.2.6 配置 Falcon 通过 Falcon HOOK

导入数据 [173](#bookmark60)

4.3 Apache Atlas 的场景设计 [176](#bookmark61)

4.3.1 Atlas 总场景介绍 [176](#bookmark62)

4.3.2 Atlas 非实时数据场景 [178](#bookmark63)

4.3.3 Atlas 实时数据场景 [183](#bookmark64)

4.3.4 Hive 数据表操作 [183](#bookmark65)

4.4 Apache Atlas 优化与性能分析 190

4.5 本章小结 [193](#bookmark66)

**第** **5** **章** **大数据安全之** **Apache**

**Ranger** [194](#bookmark67)

5.1 Apache Ranger 概述 [194](#bookmark68)

5.1.1 Ranger 技术概况 [194](#bookmark69)

5.1.2 Ranger 发展史及近况 [196](#bookmark70)

5.1.3 Ranger 的特点和作用 [197](#bookmark71)

5.1.4 Ranger 架构 [199](#bookmark72)

5.1.5 Ranger 应用场景 [200](#bookmark73)

5.2 Apache Ranger 的安全认证配置 201

5.2.1 Ranger 安装与部署 [201](#bookmark74)

5.2.2 安全及访问权限控制机制……206

5.2.3 Ranger 集成 HDFS 的安全认证

机制与配置 [208](#bookmark75)

5.2.4 Ranger 集成 YARN 的安全认证

机制与配置 [213](#bookmark76)

5.2.5 Ranger 集成 Hive 的安全认证

机制与配置 [217](#bookmark77)

VI

5.2.6 Ranger 集成 HBase 的安全认证

机制与配置 [221](#bookmark78)

5.2.7 Ranger 集成 Kafka 的安全认证

机制与配置 [228](#bookmark79)

5.2.8 Ranger 集成 Atlas 的安全认证

机制与配置 [235](#bookmark80)

5.2.9 Ranger 集成 Storm 的安全认证

机制与配置 [238](#bookmark81)

5.2.10 Ranger 集成 Solr 的安全认证

机制与配置 [246](#bookmark82)

5.3 Apache Ranger 的功能配置 [254](#bookmark83)

5.3.1 Tag 同步验证 [254](#bookmark84)

5.3.2 各类 Policy 验证 [255](#bookmark85)

5.4 Apache Ranger 优化与性能分析 262

5.5 本章小结 [263](#bookmark86)

**第6章** **大数据安全之** **Apache**

**Sentry** [265](#bookmark87)

6.1 Apache Sentry 概述 [265](#bookmark88)

6.1.1 Apache Sentry 技术概况 [265](#bookmark89)

6.1.2 Apache Sentry 发展近况 [267](#bookmark90)

6.1.3 Apache Sentry 技术优势 [269](#bookmark91)

6.1.4 Apache Sentry 架构 [272](#bookmark92)

6.2 Apache Sentry 的安装与配置 274

6.2.1 先决条件 [274](#bookmark93)

6.2.2 Impala 的安装与调试 [274](#bookmark94)

6.2.3 Apache Sentry 的安装和配置 …282

6.2.4 Apache Sentry 与 Impala 的

集成 [286](#bookmark95)

6.3 Apache Sentry 场景设计之 Sentry

对 Impala 的 控 制 [288](#bookmark96)

6.3.1 场景数据准备 [288](#bookmark97)

6.3.2 基于文件存储元数据的场景

验 证 [288](#bookmark98)

6.3.3 基于数据库存储元数据的场景

验证 [292](#bookmark99)

6.4 Apache Sentry 场景设计之 Sentry

对 Hive 的控制 [296](#bookmark100)

6.4.1 Hive 与 Sentry 的集成配置 296

6.4.2 准备实验数据 [298](#bookmark101)

6.4.3 基于文件存储方式的数据表

操作 [298](#bookmark102)

6.4.4 基于数据库存储方式的数据表

操作 [301](#bookmark103)

6.5 本 章 小 结 [305](#bookmark104)

**第** **7** **章** **大数据安全之** **Kerberos**

**认** **证** [306](#bookmark105)

7.1 Kerberos 概述 [306](#bookmark106)

7.1.1 Kerberos 技术概况 [306](#bookmark107)

7.1.2 Kerberos 发展史及近况 [307](#bookmark108)

7.1.3 Kerberos 架 构 [308](#bookmark109)

7.1.4 Kerberos 的认证流程 [309](#bookmark110)

7.1.5 Kerberos 的风险与缺陷 [311](#bookmark111)

7.1.6 Kerberos 应用举例 [312](#bookmark112)

7.2 Kerberos 使用操作说明 [314](#bookmark113)

7.2.1 名词解释 [314](#bookmark114)

7.2.2 KDC 常用操作 [315](#bookmark115)

7.2.3 Client 常用操作 [316](#bookmark116)

7.3 Kerberos 集成环境配置 [317](#bookmark117)

7.3.1 Kerberos 服务配置 [318](#bookmark118)

7.3.2 HDFS 集成 Kerberos 的安装与

调 试 [322](#bookmark119)

7.3.3 YARN集成 Kerberos 的安装与

调试 [330](#bookmark120)

7.3.4 Hive 集成 Kerberos 的配置与

VHI

调 试 [336](#bookmark121)

7.3.5 Zookeeper 集成 Kerberos 的配置

与调试 [341](#bookmark122)

7.3.6 HBase 集成 Kerberos 的配置与

调 试 [343](#bookmark123)

7.3.7 Sqoop 集成 Kerberos 的配置与

调试 [348](#bookmark124)

7.3.8 Hue 集成 Kerberos 的安装与

调 试 [351](#bookmark125)

7.3.9 Spark 集成 Kerberos 的安装与

调试 [361](#bookmark126)

7.3.10 Solr 集成 Kerberos 的安装与

调 试 [366](#bookmark127)

7.3.11 Kafka 集成 Kerberos 的配置

与调试 [371](#bookmark128)

7.3.12 Storm 集成 Kerberos 的安装

与 调 试 [377](#bookmark129)

7.3.13 Impala 集成 Kerberos 的安装

与调试 [382](#bookmark130)

7.4 Kerberos 配置优化及常见问题 386

7.4.1 Kerberos 的认证方式 [386](#bookmark131)

7.4.2 时间同步 [386](#bookmark132)

7.4.3 ticket 周期 [387](#bookmark133)

7.4.4 KVNO 导致的认证失败 [387](#bookmark134)

7.5 本 章 小 结 [388](#bookmark135)

第一篇 

**理** **论** **篇**

理论篇主要介绍大数据治理技术的相关背景知 识，包括大数据安全、隐私和审计技术。其中，第 1章总体介绍大数据治理的基本概念以及框架，使 读者了解大数据治理的现状以及为此制定的相关原 则、范围以及评估细则。第2章分别就大数据安 全、隐私以及审计技术做了更加细致的介绍，让读 者对大数据治理面临的挑战和解决方案有一个更加 全面的了解。本篇的目的是使读者了解大数据治理 的背景，包括急需解决的问题和常用的技术方案， 使得读者对大数据治理有一个理论上的认知，为后 续开源实现篇的学习打好基础。



■■

■

■ 暴

■■■ ■■

■

■ ■

■■■

■ ■

■■ ■

■1

■■■■国

■| ■■ ■!

■!

■

■ ■

■

■■

■

■

■

■

■

第1章

大数据治理技术

**1.1** **概述**

**1.1.1** **大数据治理的基本概念**

现如今，我们已被数据包围，数据正在逐渐将我们淹没。来自于社交媒体、网络日 志、GPS 信号、RFID 标签、网络音频、数字图片等方面的数据扑面而来。大数据被炒得火 热，大数据时代已然来临。而大数据本身是一个比较抽象的概念，如果我们仅仅从字面来 理解，它表示数据规模的庞大。但是仅仅数量上的庞大这一简单的理解显得有些狭隘，难 以区分这一概念和以往的“海量数据”“超大规模数据”等概念的区别。而现如今，当谈到 大数据定义时都运用比较有代表性的3V定义，即认为大数据需满足以下3个特点：规模性 (Volume) 、多样性 (Variety) 和高速性 (Velocity) 。 而 IDC 认为还应该添加数据具有的价值 性 (Value),IBM 认为大数据必然具有真实性 (Veracity) 。 当然每个人对大数据有不同的理 解，当我们面对实际问题时，没必要拘泥于这些现有的定义，只要符合业务规则即可。

伴随着网络和信息技术的不断发展与普及，人类产生的数据量正在呈指数级增长，在 历史上从未有哪个时代产生如此海量的数据。数据的产生已经完全不受时间、地点的限制， 大约每两年就会翻一倍，换句话说，每两年产生的数据量相当于之前产生的全部数据量。 并且根据现有的数据量监测，这个速度还会在很长一段时间内保持下去。信息数据的单位 由 TB→PB→EB→ZB 的级别暴增，而这样的数据很明显已经远远超出了我们人力所能处 理的范围，因此大数据应运而生。它的重要性也因此而得之。

伴随着数据行业的昌盛发展，很自然就产生了一个对应的问题：这些数据作为原材料 应该怎么管理?虽然数据管理并不新鲜，很早以前我们也一直在做，但随着数据爆炸性地 呈指数级增长，我们如今所讲的数据和以往已经大大不同。而这也不仅仅体现在数据的大

第1章 大数据治理技术

小上，同时也体现在数据的内容、来源、结构上。举个简单的例子，现如今 Facebook 的日 均新增数据量可达600TB 左右，未来必然会更高。那么处理如此大量的数据，我们不禁要 问：以往的算法还可能吗?应用还能正常运行吗?答案是否定的。随着数据的变化，我们 的算法也要升级，同样，我们以往的数据管理方式与思路也无法完全适应，也需要创新。 因此大数据治理的概念应运而生。

既然已提出大数据治理的概念，那么它应该和大数据管理有明显的区别。COBIT5 日对

两者进行了精准的区分定义。

**1.管理定义**

管理 (Management) 是指按照治理机构设定的方向展开计划、建设、运营和监控活动， 以实现企业目标。

基于此定义，管理包含计划、建设、运营和监控4个关键活动，并且活动必须符合治 理机构所设定的方向和目标。

**2.治理定义**

治理 (Governance) 是指评估利益相关者的需求、条件和选择以达成平衡一致的企业目 标，通过优先排序和决策机制来设定方向，然后根据方向和目标来监督绩效与规范。

基于此定义，治理包括评估、指导和监督3个关键活动，并且输出结果与设定方向必 须和预期的目标一致。

从上述定义可做如下总结。

1)关键活动不同：管理包含计划、建设、运营和监控4个关键活动，治理包含评估、 治理和监督3个关键活动。

2)过程不同：根据COBIT5 的定义，管理包括4个域， APO (调整、计划和组织)、 BAI (建立、获取和实施)、DSS (交付、服务和支持)、MEA (监视、评价和评估),每个域 又包含若干个流程。而治理包含如下过程，框架的设置与维护、确保资源化、风险化、收 益交付、利益相关透明。

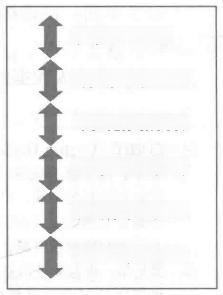
3)分工不同：治理相当于决策者，制定决策；管理相当于执行者，负责制定和实施决

策的过程。

目前最权威的大数据治理的定义由桑尼尔·索雷斯提出，主要包含如下6个部分：

θ COBIT(Control Objectives for Information and related Technology) 是目前国际上通用的信息系统审计 标准，由信息系统审计与控制协会在1996年公布。这是一个在国际上公认的、权威的安全与信息技术 管理和控制的标准，目前已经更新至5.0版。它在商业风险、控制需要和技术问题之间架起了一座桥梁， 以满足管理的多方面需要。该标准体系已在世界100多个国家的重要组织与企业中运用，指导这些组织 有效地利用信息资源，有效地管理与信息相关的风险。

e 桑尼尔·索雷斯是信息资产公司LLC的创始人和执行合伙人 (LLC专注于帮助组织构建信息治理计划), 他曾任IBM的信息治理总监，其合作客户遍布六大洲和众多行业，他是较早提出大数据安全与治理理 念的先驱之一。

4 第一篇 理 论 篇

1)大数据治理应该被纳入现有的信息治理框架内。

2)大数据治理的工作就是制定策略。

3)大数据必须被优化。

4)大数据的隐私保护很重要。

5)大数据必须被货币化，即创造商业价值。

6)大数据治理必须协调好多个职能部门的目标和利益。

根据上述相关定义可知，为了形成有效的治理体系，治理和管理必须相互作用，相互 配合，才能取得最优效果。很多技术上的相关领域涉及治理框架、数据优化、隐私保护等。

大数据的大规模性、高速性和多样性等特征，使得它不同于小量数据。将小量数据的 隐私保护方法用在大数据上会有很大的局限性：大数据的多样性带来的多源数据融合使得 传统的匿名化和模糊化技术几乎无法生效；大数据的大规模性与高速性带来的实时性分析 使得传统的加密和密码学技术遇到了极大的瓶颈。此外，大规模的数据采集技术、新型存 储技术以及高级分析技术使得大数据的隐私保护面临更大的挑战。因此数据的隐私保护与 安全也是大数据治理的重要关注点之一。

而在数据治理的框架下，元数据的管理也显得尤为重要。元数据按照数据类别信息进 行区分可分为技术元数据与业务元数据。

技术元数据是存储关于数据仓库系统技术细节的数据，是开发和管理数据仓库的使用 的数据，它主要包括以下信息：数据仓库结构的描述，包括仓库模式、视图、维、层次结 构和导出数据的定义，以及数据集的位置和内容；业务系统、数据仓库和数据集的体系结 构和模式。

业务元数据从业务角度描述了数据仓库中的数据，它提供了介于使用者和实际系统之 间的语义层，使得不懂计算机技术的业务人员也能够“读懂”数据仓库中的数据。业务元 数据主要包括以下信息：使用者的业务术语所表达的数据模型、对象名和属性名；访问数 据的原则和数据的来源；系统所提供的分析方法以及公式和报表的信息。还包括企业概念

模型，这是业务元数据所应提供的重要信息，它表示企业数

**数据参考框架**

**解决数据模糊性**

**可视化数据流动**

**影响和血统分析**

推进标准化建设

规范化数据审计

大数据安全与治理体系

据模型的高层信息、整个企业的业务概念和相互关系。

而对于元数据的管理又可分为以下两部分。

1)数据质量的管理：就像超市对物品进行清理一样，我

们的数据也需要定期清理。

2)信息生命周期的管理：对大数据进行存档，并在没必

要继续保存某些数据时将它删除。

大数据安全与治理体系下需要解决的问题如图1-1所示。

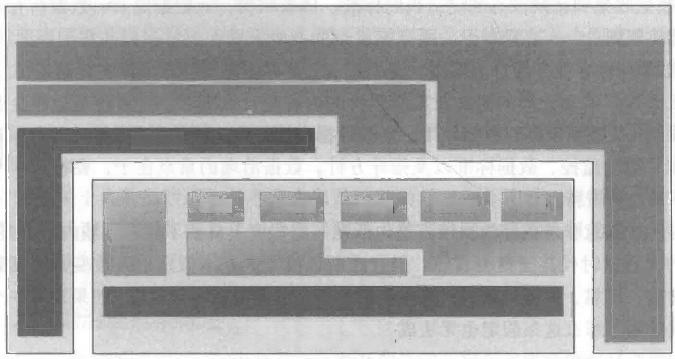
本书中，通过将Apache 的 Ranger 、Atlas 、Falcon 以

及 Hadoop 生态下的其他组件进行整合，形成完整的大数据

安全与治理体系，以此来完成安全与隐私保护、元数据管理、图1-1

第1章 大数据治理技术 5

数据生命周期管理等问题。本书中的大数据治理框架如图1-2所示。读者初看时可能难以 有清晰直观的认识，当读完本书再回头观看此图时定会有不一样的理解。

**大数据治理**

**Ranger**

Atlas

**Falcon**

Hive **HBase Storm**

**Kafka** Solr

Zookeeper

Sqoop

YARN

**HDFS**

原始Hadoop生态系统

图1-2 大数据治理框架

大数据的快速发展，使它成为IT 领域的又一大新兴产业。据估算，国外大数据行业约 有1000亿美元的市场，而且每年以10%的速度增长，增速是软件行业的2倍。而我国的 大数据行业因起步稍晚，增速更为迅猛。而目前中国政府和企业对数据治理的重视程度也 不断提升，在通信行业、银行行业、能源行业、互联网行业都已经开展了大数据治理的相 关工作。在这个过程中，学术界和工业界做了很多探索，建立了较为科学、完整的数据治 理理论体系和框架。本文从理论到实践引导读者加深理解，上文所提及的治理框架、数据 安全、隐私保护、数据质量管理、数据生命周期管理都将在实践篇给出具体的实现。

**1.1.2** **大数据治理的意义和重要作用**

如今，我们的生活已经被数据所淹没，但是目前主流的软件往往无法在合理的时间内 完成对数据的撷取、管理、处理并整理成为帮助企业经营决策的重要资讯这些工作，而随 着数据量的逐步扩增，这一现象会更加明显。所以企业经常要面对超出其基础设施和流程 处理能力的大量数据，而从数据中挖掘出对制定有效决策有实际价值的情报更是难上加难。 如今，由于种类、数量日益成倍增加的数据从社交媒体及各种在线渠道汹涌而来，导致处 理上述数据的迫切性也日益加强，企业面临着更多的技术难题和挑战。

大数据不断从各种渠道、以多种格式涌入，其中蕴含着大量商业价值，但仅利用传统 的数据处理方法和技术无法处理它们。故而早在2009年年初，《大数据资产：智慧企业如 何在数据治理中胜出》的作者 Tony Fisher 就指出，如果基础数据不可靠，多数企业或大数 据计划会失败，或者效果会低于预期。导致上述结果的关键原因是数据进入生命周期的不 一致，数据不准确，数据不可靠。这些原因可能是多样性的：

6 第 一 篇 理 论 篇

1)大数据计划中的数据识别不完整。目前还不清楚如何获取数据，如何使用数据，哪 些业务目标要满足，哪些人有权拥有数据。

2)数据收集和转换没有制定适当的标准、体系结构、元数据定义、数据所有权、策略 和数据转换规则。

3)数据传输在业务用户上下文、安全性、数据和业务流程方面没有正确定义。

那么大数据治理计划的意义及其所包含的内容是什么呢?数据治理是指在企业数据生 命整个周期(从数据采集到数据使用，直至数据存档)中，制定由业务推动的数据政策、数 据所有权、数据监控、数据标准以及指导方针。数据治理的重点在于，要将数据明确作为 企业的一种资产看待。

更好的数据意味着更好的决策，这句话在一定程度上反映了数据领域内的主要关注点， 在当今的大数据时代甚至更为真切。但它之所以成立的基本假定也未改变，那就是“基本 数据是准确、可靠、值得信赖的，来龙去脉清楚，并且具有一致性”。如果没有一个可靠的 数据治理计划，那么这条假定也无法成立。

我们都听过诸如此类的说辞： “IT 技术融入业务对我们的企业至关重要”“IT 技术促成 各种业务功能的实现”。但对企业上下进行实际的评估，能实现上述说辞的情况却是屈指可 数。对大多数企业而言， IT 技术与各种业务目标之间仍存在差距，首席信息官及各高级主 管仍在努力设法使IT 技术能配合各种业务目标，从而促进企业战略目标的实现。在对成功 企业进行分析后，可以得出一个很明确的结论，那就是“有效的数据治理计划”是成功企 业的法宝。

任何大数据计划都应该考虑数据的以下特性：数量大、种类多、产生频率高、质量可 靠性低、模糊性高。那么数据处理团队想要完全识别、定义并分析这些数据，就要征询企 业各方利益相关者的意见。这样做才能让企业拥有者、数据拥有者以及数据治理部门在数 据治理初期就避免一些错误，确保框架的正确搭建及实施，从而达到数据集规划与业务流 程紧密联系且合理有效的目的。

现在，伴随着大数据运用时代的到来，所谓“数据驱动”已然成为未来全世界的发展 趋势。现在大数据已经应用于全球的生产、分配及消费活动等，并且对于国家经济的运营 体制、社会民生和国家的治理生产、制造能力等都会产生非常重要的影响。在未来，国家 之间的竞争可能会从资本和土地等资源的争夺转移到大数据的争夺。所以，现在大数据已 经成为每个国家的战略资源的基础设施，同时，大数据治理也成为多个国家提升现代治理 能力的一个重要标杆。

随着互联网、云计算等网络相关的新技术的不断完善和知识普及，我们的社会已经进 入大数据时代，大量数据的产生和流转都将成为再平常不过的事。到2016年年底，全球近 50%的人口在使用互联网，人人都拥有一 台或多台网络终端设备，随时随地都可以上网， 所以全球的数据量也在飞速增长。2020年，预计全球的数据使用量将会达到40ZB, 每个行 业都将产生并使用大数据，大数据也将成为发展的新趋势。而大数据治理将为社会经济能

第1章 大数据治理技术 7

力发展提供新的动力。

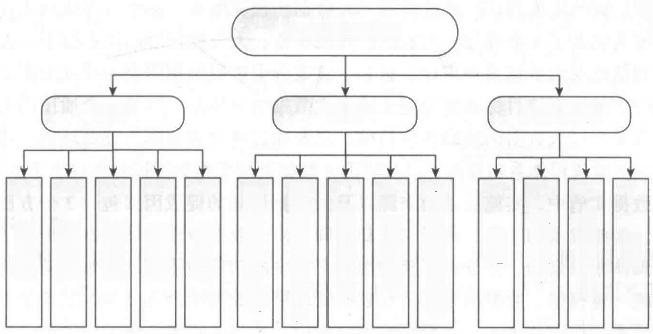
在这个大数据时代，世界上各个国家都将大数据看作国家的核心资产。因此，对大数 据的开发、利用和保护的概念就越来越强，可能还会产生对于大数据的争夺。大数据概念 的出现就使得国家的强弱对比不仅体现在经济发展层面，还体现在一个国家大数据治理实 力如何。所以对于大数据安全与治理的挑战也才刚刚开始。

**1.2** **框架**

在讲述了关于大数据治理的基本概念以及治理的意义和作用后，我们对大数据治理已 经有了简单的认识。接下来将会从3个维度阐述大数据治理的框架，目的是让读者更加深 刻地认识、理解大数据治理。

**1.2.1** **大数据治理框架概述**

大数据治理框架从全局视角描述了大数据治理的主要内容，下面我们从大数据治理原 则、治理范围、治理的实施与评估3个维度给出大数据治理的全貌，展现大数据治理的重 要性以及如何进行大数据治理，如图1-3所示。



审 计

成熟度评估

体系框架

实施

大数据服务创新

数据质量

大数据架构

大数据安全与隐私

大數据生命周期

安 全 性

开 放 性

统 一 性

价 值 化

有 效 性

大数据治理框架

实施与评估

范围

原则

图1-3 大数据治理框架

其中大数据治理的原则给出了大数据治理过程中所遵循的、首要的、基本的指导性法 则，即有效性原则、价值化原则、统一性原则、开放性原则、安全性原则，这5个部分分 别从各个层面、各个角度解释了大数据治理所应遵循的原则的重要性与必要性。其中，有 效性原则体现了大数据治理过程中数据的标准、质量、价值、管控的有效性、高效性；价 值化原则体现了大数据治理过程中以数据资产为价值核心，最大化大数据平台的数据价值； 统一性原则能够形成一套规范的、有条理的、可遵循的准则，能够节约很大的成本、时间， 对大数据的治理具有重要意义和作用；开放性原则是为了提高数据治理的透明度，不让海

8 第一篇 理 论 篇

量数据信息在封闭的环境中沉睡，同时共享信息，安全合理地共享数据，使数据之间形成 关联，形成一个良好的数据标准；安全性原则体现了安全的重要性、必要性，保障大数据 平台的数据安全和数据治理过程中数据的安全可控。

大数据治理的范围描述了大数据治理的关键域，即大数据治理决策层应该在哪些关键 领域内做出决策。该维度共包含5个关键领域：大数据生命周期、大数据架构、大数据安 全与隐私、数据质量以及大数据服务创新。这5个关键领域就是大数据治理的主要决策领 域，规定了大数据治理主要应用的地方以及方向。其中，大数据生命周期是指数据产生、 获取到销毁的全过程，在大数据治理中生命周期的管理更注重在成本可控的情况下有效地 管理并使用大数据，从而创造出更大的价值。大数据生命周期管理包含了数据捕获、数据 维护、数据合成、数据利用、数据发布、数据归档和数据清除。大数据架构是指大数据在 IT 环境下进行存储、使用以及管理的逻辑或物理架构，主要包含了大数据来源、大数据存 储、大数据分析以及大数据应用和服务4个部分。大数据安全与隐私提供了大数据隐私管 理的几个步骤，来对大数据云计算时代的数据进行隐私安全保障。数据质量领域总结了大 数据产生质量问题的原因，以及应该从哪几个方面入手去有效提升大数据质量。大数据服 务创新领域提出应该从基于数据本身进行创新、基于业务需求进行创新、基于数据分析的 创新3个方面进行探讨，来体现对大数据服务的创新。

大数据治理的实施与评估维度描述了大数据治理实施和评估中需要重点关注的关键内 容，该维度共包含了4个部分：大数据治理的实施、大数据治理的体系框架、大数据治理 的成熟度评估以及大数据治理审计。它为企业实施大数据治理提供指导性方案。其中，大 数据治理的实施的直接目标就是为企业建立大数据治理体系，形成一个通用的大数据治理 架构。而为了实现大数据治理的实施目标，需要通过建立大数据治理的环境、建立完善的 大数据治理实施流程体系和规范，以及明确大数据治理实施的阶段目标这3个方面来完成。 同时在大数据实施中，实施的动力来源以及大数据治理的促成因素包含3个方面：治理实 施的环境、实施技术和工具、流程与活动管理。而大数据治理的体系框架提出了一个通用 的数据治理体系及架构，并分析了架构内各个模块的功能与作用，从数据持久化层、数据 集成层、统一建模层、数据质量层、元数据管理层和数据治理人员组织层5个方面对大数 据治理的体系结构进行阐述。大数据治理过程中，通过成熟度评估可以了解当前大数据治 理实施的状态和实施方向，成熟度可以帮助了解治理的重要性。根据能力成熟度分类的方 法，将成熟度分为5个等级，等级由低到高分别为：初始级、受管级、定义级、定量管理 级、优化级。大数据治理的审计不仅可以提高大数据治理的实施水平，还能从更全面的角 度为大数据治理提供实施意见，而且大数据审计还可以满足企业监管的需要，改善大数据 在治理过程中的安全和隐私。

相关组织及企业可根据上述3个维度的指导原则，从大数据治理原则、治理范围、 治理的实施与评估3个维度了解大数据的治理工作，按照治理原则中所遵循的指导性法 则、治理范围中的治理关键域以及实施与评估维度中的关键内容，持续稳步地推进大数

第1章 大数据治理技术 9

据治理工作。

**1.2.2** **大数据治理的原则**

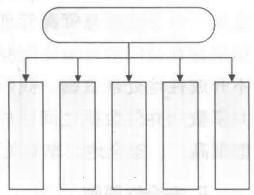
大数据治理原则是指大数据治理所遵循的、首要的、基本的指导性法则。大数据治理 原则对大数据治理实践起指导作用，只有将原则融入实践过程中，才能实现大数据治理的

战略和目标。提高大数据运用能力，可以有效增强政府服务和

监管的有效性。为了高效采集、有效整合、充分运用庞大的数 据，提出以下5项大数据治理的基本原则，如图1-4所示。

**1.有效性原则**

有效性原则体现了大数据治理过程中数据的标准、质量、 价值、管控的有效性、高效性。在大数据治理的过程中，首先 需要的是对数据处理的信息准确度高、理解上不存在歧义，遵 循有效性原则，选择有用数据，淘汰无用数据，识别出有代表



安全性原则

开放性原则

统 一 性原则

价值化原则

有效性原则

原则

图1-4 大数据治理原则

性的本质数据，去除细枝末节或无意义的非本质数据。这种有效性原则在大数据的收集、 挖掘、算法和实施中具有重要作用。运用有效性原则就能够获取可靠数据，减少数据集规 模，提高数据抽象程度，提升数据挖掘的效率，使之在实际工作中可以根据需要选用具体 的分析数据和合适的处理方法，以达到操作上的简单、简洁、简约和高效。具体来说，当 一位认知主体面对收集到的大量数据和一些非结构化的数据对象，如文档、图片、饰品等 物件时，不仅需要掌握大数据管理、大数据集成的技术和方法，遵循“有效性原则”和“数 据集成原则”,学会数据的归档、分析、建模和元数据管理，还需要在大量数据激增的过程 中，学会规约、选择、评估和发现某些潜在的本质性变化，包括对新课题、新项目的兴趣 开发。

**2.价值化原则**

价值化原则指大数据治理过程中以数据资产为价值核心，最大化大数据平台的数据价 值。数据本身不产生价值，但是从庞杂的数据背后挖掘、分析用户的行为习惯和喜好，找 出更符合用户“口味”的产品和服务，并结合用户需求有针对性地调整和优化自身，这具 有很大的价值。大数据在各个行业应用都是通过大数据技术来获知事情发展的真相，最终 利用这个“真相”来更加合理地配置资源。而要实现大数据的核心价值，需要3个重要的 步骤，第1步是通过“众包”的形式收集数据，第2步是通过大数据的技术途径进行全面 的数据挖掘，最后利用分析结果进行资源优化配置。

**3.统一性原则**

统一性原则是在数据标准管理组织架构的推动和指导下，遵循协商一致制定的数据标 准规范，借助标准化管控流程得以实施数据统一性的原则。如今的大数据和云计算已经成 为社会发展动力中新一轮的创新平台，基于大数据系统做一个数据产品，需要数据采集、

10 **第一篇** **理** **论** **篇**

收集、存储和计算等多个步骤，整个流程很长。经过统一规范后，通过标准配置，能够大 大缩短数据采集的整个流程。大数据治理遵循统一性原则，能够节约很大的成本及时间， 同时形成一个规范，这对于大数据的治理具有重要意义与作用。

**4.开放性原则**

在大数据和云环境下，要以开放的理念确立起信息公开的政策思想，运用开放、透明、 发展、共享的信息资源管理理念对数据进行处理，提高数据治理的透明度，不让海量的数 据信息在封闭的环境中沉睡。我们不能以信息安全为理由使很多数据处于沉睡的状态，而 不开放性地处理数据。我们需要对信息数据进行自由共享，向公众开放数据，安全合理地 共享数据并使数据之间形成关联，形成一个良好的数据标准和强有力的数据保护框架，使 数据高效、安全地共享和关联，在保护公民个人自由的同时促进经济的增长和创新。

**5.安全性原则**

大数据治理的安全性原则体现了安全的重要性、必要性，保障大数据平台数据安全和 数据治理过程中数据的安全可控。大数据的安全性直接关系到大数据业务能否全面推广， 数据治理过程中在利用大数据优势的基础上，要明确其安全性，从技术层面到管理层面采 用多种策略，提升大数据本身及其平台的安全性。在大数据时代，业务数据和安全需求相 结合，才能够有效提高企业的安全防护水平。大数据的汇集不可避免地加大了用户隐私数 据信息泄漏的风险。由于数据中包含大量的用户信息，使得对大数据的开发利用很容易侵 犯公民的隐私，恶意利用公民隐私的技术门槛大大降低。在大数据应用环境下，数据呈现 动态特征，面对数据库中属性和表现形式不断随机变化，基于静态数据集的传统数据隐私 保护技术面临挑战。各领域对于用户隐私保护有多方面要求和特点，数据之间存在复杂的 关联和敏感性，而大部分现有隐私保护模型和算法都是仅针对传统的关系型数据，而不能 直接将其移植到大数据应用中。

传统数据安全往往是围绕数据生命周期部署的，即数据的产生、存储、使用和销毁。 随着大数据应用的增多，数据的拥有者和管理者相分离，原来的数据生命周期逐渐转变成 数据的产生、传输、存储和使用。由于大数据的规模没有上限，且许多数据的生命周期极 为短暂，因此，传统安全产品要想继续发挥作用，需要随时关注大数据存储和处理的动态 化、并行化特征，动态跟踪数据边界，管理对数据的操作行为。

大数据安全不同于关系型数据安全，大数据无论是在数据体量、结构类型、处理速度、 价值密度方面，还是在数据存储、查询模式、分析应用上都与关系型数据有着显著差异。

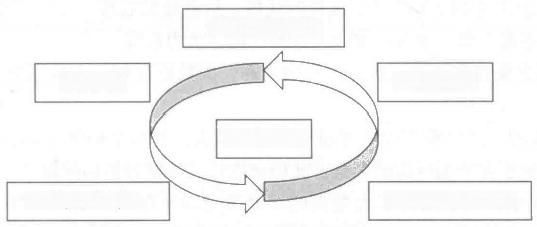
为解决大数据自身的安全问题，需要重新设计和构建大数据安全架构和开放数据服务， 从网络安全、数据安全、灾难备份、安全风险管理、安全运营管理、安全事件管理、安全 治理等各个角度考虑，部署整体的安全解决方案，以保障大数据计算过程、数据形态、应 用价值的安全。

第1章 大数据治理技术  11

**1.2.3** **大数据治理的范围**

大数据蕴含价值的逐步释放，使其成为IT 信息产业中最具潜力的蓝海。大数据正以一 种革命风暴的姿态闯入人们的视野，其技术和市场在快速发展，从而使数据治理的范围变 成不可忽略的因素。

大数据治理范围着重描述了大数据治理的关键领域。大数据治理的关键领域包括：大 数据生命周期，大数据架构(大数据存储、元数据、数据仓库、业务应用),大数据安全与 隐私，数据质量，大数据服务创新，如图1-5所示。



大数据生命周期

大数据架构

大数据治理

大数据安全与隐私

大数据服务创新

数据质量

图1-5 大数据治理关键领域

**1.大数据生命周期**

大数据生命周期是指数据产生、获取到销毁的全过程，具体可分为数据捕获、数据维 护、数据合成、数据利用、数据发布、数据归档、数据清除等。

传统数据的生命周期管理的重点在于节省成本和保存管理。而在大数据时代，数据的 生命周期管理的重点则发生了翻天覆地的变化，更注重在成本可控的情况下，有效地管理 并使用大数据，从而创造出更大的价值。

大数据生命周期管理面临着巨大的挑战，其中包括3个主要类别：无穷无尽的数据总 量，新数据的短期有效性，以及数据的一致性。

大数据生命周期管理主要包括以下部分：

1)数据捕获，即创建尚不存在或者虽然存在但并没有被采集的数据。主要包括3个方 面的数据来源，数据采集、数据输入、数据接收。

2)数据维护，即数据内容的维护(无错漏、无冗余、无有害数据)、数据更新、数据逻 辑一致性等方面的维护。

3)数据合成，即利用其他已经存在的数据作为输入，经过逻辑转换生成新的数据。例 如我们已知计算公式：净销售额=销售总额-税收，如果知道销售总额和税收，就可以计 算出净销售额。

4)数据利用，即在企业中如何使用数据，把数据本身当作企业的一个产品或者服务进 行运行和管理。

5)数据发布，即在数据使用过程中，可能由于业务的需要将数据从企业内部发送到企

12 第一篇 理 论 篇

业外部。

6)数据归档，即将不再经常使用的数据移到一个单独的存储设备上进行长期保存的过

程，对涉及的数据进行离线存储，以备非常规查询等。

7)数据清除，即在企业中清除数据的每一份拷贝。

**2.大数据架构**

大数据架构是指大数据在 IT 环境中如何进行存储、使用及管理的逻辑或者物理架构。 它由大数据架构师或者设计师在实现一个大数据解决方案的物理实施之前创建，从逻辑上 定义了大数据关于其存储方案、核心组件的使用、信息流的管理、安全措施等的解决方案。 建立大数据架构通常需要以业务需求和大数据性能需求为前提。

大数据架构主要包含4个层次：大数据来源，大数据存储，大数据分析，大数据应用 和服务。

1)大数据来源：此层负责收集可用于分析的数据，包括结构化、半结构化和非结构化 的数据，提供解决业务问题所需的洞察。此层是进行大数据分析的前提。

2)大数据存储：主要定义了大数据的存储设施以及存储方案，以进一步进行数据分 析处理。通常这一层提供多个数据存储选项，比如分布式文件存储、云、结构化数据源、 NoSQL 等。此层是大数据架构的基础。

3)大数据分析：提供大数据分析的工具以及分析需求，从数据中提取业务洞察，是大 数据架构的核心。分析的要素主要包含元数据、数据仓库。

4)大数据应用和服务：提供大数据可视化、交易、共享等，由组织内的各个用户和组 织外部的实体(比如客户、供应商、合作伙伴和提供商)使用，是大数据价值的最终体现。

**3.大数据安全与隐私**

大数据作为社会的又一个基础性资源，将给社会进步、经济发展带来强大的驱动力。 大数据代表了先进技术的发展方向，已经成为不可阻挡的趋势。在大数据时代，数据的收 集与保护成为竞争的着力点。从个人隐私安全层面看，大数据将大众带入开放、透明的 “裸奔”时代，若对数据安全保护不利，将引发不可估量的问题。解决传统网络安全的基本 思想是划分边界，在每个边界设立网关设备和网络流量设备，用守住边界的办法来解决安 全问题。但随着移动互联网、云服务的出现，网络边界实际上已经消亡了。因此，在开放 大数据共享的同时，也带来了对数据安全的隐忧。大数据安全是“互联网+”时代的核心 挑战，安全问题具有线上和线下融合在一起的特征。

可以尝试以下方法进行大数据的隐私管理°:

1)定义和发现敏感的大数据，并在元数据库中将敏感大数据进行标记和分类。

2)在收集、存储和使用个人数据时，需要严格执行所在地关于隐私方面的法律法规， 并制定合理的数据保留、处理政策，遵循公司法律顾问和首席隐私官的建议。

日 引用自 <http://www.thebigdata.cn/JieJueFangAn/13655.html>。

第1章 大数据治理技术 13

3)在存储和使用过程中，对敏感大数据进行加密和反识别处理。

4)加强对系统特权用户的管理，防止特权用户访问敏感大数据。

5)在数据的使用过程中，需要对大数据用户进行认证、授权、访问和审计等管理，尤 其是要监控用户对机密数据的访问和使用。

6)审计大数据认证、授权和访问的合规性。

大数据也和其他领域的新技术一样，给我们带来了安全与隐私问题。另外，它们也不 断地对我们管理计算机的方法提出挑战。正如印刷机的发明引发了社会自我管理的变革一 样，大数据也是如此。它迫使我们借助新方法来应对长期存在的安全与隐私挑战，并且通 过借鉴基本原理对新的隐患进行应对。我们在不断推进科学技术进步的同时，也应确保我 们自身的安全。

**4.数据质量**

当前大数据在多个领域广泛存在，大数据的质量对其有效应用起着至关重要的作用， 而且在大数据使用过程中，如果存在数据质量问题，将会带来严重的后果，因而需要对大 数据进行质量管理。大数据产生数据质量问题的具体原因如下：

1)由于规模大，其在收集、存储、传输和计算过程中可能产生更多的错误，如果对其 采用人工错误检测与修复，将导致成本极其巨大而难以有效实施。

2)由于高速性，数据在使用过程中难以保证其一致性。

3)大数据的多样性使其具有更大的可能产生不一致和冲突。

如果没有良好的数据质量，大数据将会对决策产生误导，甚至产生有害的结果。高质 量的数据是进行数据分析和数据使用以及保证数据质量的前提。大数据质量控制在实施大 数据质量和减轻大数据治理并发症过程中发挥着重要作用，它能够把社会媒体或其他非传 统的数据源进行标准化，并且可以有效防止数据散落。

建立可持续改进的数据管控平台，有效提升大数据质量管理，可以从以下几个方面 入手：

1)数据质量评估，提供全方位数据质量评估能力，如数据的正确性、完全性、 一致 性、合规性等，对数据进行全面体检。

2)数据质量检核和执行，提供配置化的度量规则和检核方法生成能力，提供检核脚本 的定时调度执行。

3)数据质量监控，系统提供报警机制，对检核规则或方法进行阈值设置，对超出阈值 的规则进行不同级别的告警和通知。

4)流程化问题处理机制，对数据问题进行流程处理支持，规范问题处理机制和步骤， 强化问题认证，提升数据质量。

5)根据血统关系锁定在仓库中使用频率较高的对象，进行高级安全管理，避免误操作。

数据质量管理是一个综合的治理过程，不能只通过简单的技术手段解决，需要从企业 的高度加以重视，才能在大数据世界里博采众长，抢占先机。

14 第一篇 理 论 篇

**5.大数据服务创新**

在信息经济发展迅猛的今天，随着数据扮演生产要素的角色，云计算发挥公共计算基 础设施的作用，数据的开放、共享与流动成为可能，大数据的服务创新将激发新的生产力。 在大数据时代，各个企业的核心竞争力不仅仅是数据量的竞争，多类数据之间融合、分析、 挖掘与利用才是各企业间竞争的主要内容，加强数据服务创新将成为竞争的关键因素。下 面，将主要从基于数据本身进行创新、基于业务需求进行创新、基于数据分析的创新3个 方面探讨大数据服务创新。

1)基于数据本身进行创新：直接分析、统计、挖掘、可视化拥有的数据，从而发现一 些规律，对业务进行创新。

2)基于业务需求进行创新：通过对数据的价值链、业务关联接口、业务要素等方面的 创新，可以深入洞察业务需求，发现特色数据，进而提供更加个性化的服务。

3)基于数据分析的创新：针对数据定义，通过数据中间处理以及数据处理自动化、智 能化的创新，进一步更清晰地呈现数据，并对数据进行更明确的分析和更深层的解读。

**1.2.4** **大数据治理的实施与评估**

大数据治理的实施与评估描述了大数据治理的实施和评估过程中需要重点关注的内容， 包含大数据治理的实施环境、实施步骤，以及实施结果的评估，为企业实施大数据治理提 供指导性方案。

**1.大数据治理的实施**

大数据治理的实施的最直接目标就是为企业建立大数据治理的体系，凭借IT 方面治理 的实施方法论，并结合大数据治理的特征，形成一个通用的大数据治理框架，并着重指出 在每个阶段需要关注的关键要素以及在各个阶段的产出物。为实现大数据治理的目标，主 要进行以下三方面的实施。首先，需要建立大数据治理的软硬件环境，综合考虑数据量大 小、用户及时性需求等来建立大数据治理的环境，这是大数据治理实施的基础。其次，需 要建立完善的大数据治理实施流程体系和规范，完善的流程是保障大数据治理顺利实施的 重要措施。最后，明确制定大数据治理实施的阶段目标，明确目标将会促使大数据治理实 施能够高质量地完成。实施大数据治理的长期目标是通过大数据治理，为企业的利益相关 者带来价值，这种价值主要体现在三个方面，分别是业务创新、价值获取、风险控制。

在大数据治理的实施过程中，首先必须明确大数据治理的未来目标以及促成因素，从 而让企业的决策者对大数据治理的实施制定总体规划。

大数据实施的动力主要来源于大数据治理的业务需求，这些需求包括内部需求和外部 需求。这些需求从高到低分别为：企业高层管理根据企业的价值方向确定大数据治理的发 展策略以及重大决策；业务管理员根据提升管理水平、降低大数据的运营成本等目标，制 定企业的具体运作和管理任务；业务操作员根据提升业务处理水平来实施具体业务，而不 负责监督其他管理任务；基础设施层主要负责为大数据治理的实施提供统一的基础设施

第1章 大数据治理技术 15

管理。

大数据治理的促成因素是指对大数据治理的成功实施具有关键性作用的因素，主要包 括三方面：治理实施的环境、实施技术和工具、流程与活动管理。治理实施的环境主要包 括内部环境和外部环境，内部环境主要包括企业内部文化，外部环境主要包括大数据实施 环境、企业现在所具备的技能和知识等。实施技术和工具主要是指为大数据治理实施提供 有力的支撑和保障，在使用相应的技术和工具时主要包含以下内容：技术与工具的安全性 保障，具备大数据的访问和控制技术；利用技术和工具对数据的生命周期等进行配置管理； 审计和报告工具来完成对业务流程的监控，提前发现可疑活动，减轻系统管理的负担，提 高问题处理效率。流程与活动管理主要包括详细定义流程的作用和流程的目的，优化用户 和大数据之间的沟通效率。

**2.大数据治理的体系框架**

近些年国内的研究更关注于某一行业或者领域的数据治理应用情况，缺乏通用的大数 据治理框架体系。本书针对这一问题，提出一个通用的大数据治理体系架构，并分析了架 构内各个模块的功能与作用。数据治理体系框架包括数据持久化层、数据集成层、统一建 模层、数据质量层、元数据管理层和数据治理人员组织层。

1)持久化。持久化是数据治理的基础问题。在传统的数据管理层场景， 一般用关系型 数据库作为数据持久化的载体。对于这一问题已有许多研究，本书不赘述。然而伴随大数 据时代的到来， 一方面，传统的关系型数据库很难应对数据量过大的问题，因此在数据治 理体系内引入NoSQL 数据库是大数据问题驱动的必然选择；另一方面，非结构化数据往往 以大文件的形式存在，这些大文件通常依赖于分布式的文件系统，如HDFS 、TFS 等，相比 传统的数据治理方法，新一代的数据治理体系应当对这些新生的大数据技术给予支持。

2)数据集成。企业内部不同系统之间往往存在许多共有的复用数据。在传统的数据管 理体系下，并未对这些数据给出明确定义。在数据治理体系下，这些数据以“主数据”的 形式表达出来。主数据是指具有高业务价值的、可以在企业内跨越各个业务部门被重复使 用的数据，是单一、准确、权威的数据来源。主数据的实施，更有利于系统的集成和数据 的协调管理。

作为不同系统产生的大文件数据，上层的系统应用需要统一的 SQL 接口来部署和管 理，这就用到了数据仓库技术。例如 Hadoop 体系中的 Hive, 可以很好地将不同的大文件 抽象出统一的SQL 接口供上层使用，通过将SQL 语句转化为大数据常用的 MapReduce 程 序来实现数据查找等功能。这一过程对上层完全透明，大大简化了开发难度。

3)统一建模。统一建模是主数据和数据仓库建立的重要标准。从本质上看，数据治理 体系下的数据建模与传统的关系型数据库的建模并无本质不同，同样存在着三级数据模型， 即概念数据模型、逻辑数据模型和物理数据模型。

4)数据质量。数据质量是数据治理的重要内涵，我们可以把数据质量理解为“数据满 足要求的程度”。其中，数据质量又有许多评估维度，例如数据的来源是否可靠?数据是否

16 ◆ 第一篇 理 论 篇

完整?数据是否可访问?数据是否安全?等等。这些都要根据客观的实际需要来制定。

5)元数据管理。元数据 (Metadata) 通常被用来表达实体数据的描述信息，即可称为 “数据的数据”。抽象出这种用来表述数据特征的数据，是为了加强数据的统一管理，实现 数据资源的科学整合，有利于数据的长期保存。例如，统一建模的描述信息、数据质量的 定义等信息可统统交由元数据库来管理。

6)数据治理人员组织。数据治理的真正实施，说到底还是要依赖企业的人员组织部 门。具体来说，是由战略决策者、业务管理者、业务操作员具体推进数据治理的实施。战 略决策者制定企业大数据发展的重要战略和决策，其主要人员往往是企业的决策和高层管 理人员，如企业技术总监、首席数据官和首席架构师等。战略决策层实施大数据治理的动 力在于利用大数据辅助企业高层管理者制定重大决策，支持企业风险管控、价值实现和服 务创新，从而建立并保持企业的竞争优势。业务管理者一般是IT 项目经理、IT 部门主管或 者 IT 部门经理，负责企业的具体运作和管理任务。业务管理者在数据治理实施过程中负责 提升企业 IT管理水平，降低数据的运营成本，提高数据的客户服务水平，控制数据管理的 风险等。业务操作员一般不具有监督和管理的职责，通常负责执行具体的工作。在业务操 作层，大数据治理实施的动力就是规范和优化大数据应用的活动和流程，提升大数据的业 务处理水平，具体包括大数据应用的效果和质量，大数据应用的可持续性、时效性、有效 性和可靠性等。通过有效的人员组织，将会大大提高数据治理落地速度和实施效果。

**3.大数据治理的成熟度评估**

大数据治理过程中，通过成熟度评估可以了解当前大数据治理实施的状态和实施方向， 认识数据治理的重要性，为实现数据价值最大化提供依据，帮助企业管理者更智慧地经营 和决策，以达到确保数据的及时性、高品质、可分享性和一致性的目的。达成这些目标对 实现灵活的商业运营和成果丰富的数据分析至关重要，进而才能据此做出针对性精准的商 业决策。因此，大数据治理的成熟度评估是大数据治理成功实施的至关重要的一步。

根据能力成熟度模型提供的分类方法，可以将成熟度分为5个等级： I 级为初始级，此 时流程通常是临时的，整体环境不够稳定；2级为受管级，实施成功是可重复发生的，但可 能无法针对组织中所有项目重复流程，存在基本的项目管理和流程规则，但仍有超出预期 成本和时间的风险；3级为定义级，建立了标准流程集，通过组织的标准流程集定制标准、 流程描述和项目流程，以适应特定项目或组织单位；4级为定量管理级，对流程进行定量度 量和控制，所选的子流程大大提高了整体流程绩效；5级为优化级，在该级明确了组织的定 量流程改进目标，并不断优化，以适应变化的业务目标日。

IBM 数据治理成熟度模型共使用了11个类别来度量数据治理能力，11个类别又分为4 个相互关联的组。

1)数据风险管理及合规性：确定数据治理与风险管理关联度，用来量化、跟踪、避免

e <https://www.ibm.com/developerworks/cn/data/library/bd-1503bigdatagovernance4/index.html>

第1章 大数据治理技术  17

或转移风险等。

2)价值创造：确定数据资产是否能帮助企业创造更大价值。

3)组织结构和意识：主要用来评估企业针对数据治理是否拥有合适的数据治理委员会、 数据治理工作组和全职的数据治理人员，是否建立了数据治理规章以及高级主管对数据是 否重视等。

4)管理工作：是指质量控制规程，用来管理数据以实现资产增值和风险控制等。

5)策略：为企业如何管理数据在高级别上指明方向。

6)数据质量管理：主要指用来提高数据质量，保证数据准确性、 一致性和完整性的各

种方法。

7)信息生命周期管理：主要指对结构化、半结构化以及非结构信息化全生命周期管理 相关的策略、流程和分类等。

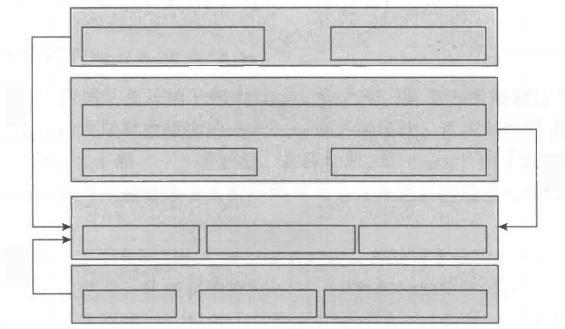
8)信息安全与隐私：主要指保护数据资产、降低风险的各种策略、实践和控制方法。 9)数据架构：是指系统的体系结构设计，支持向适当用户提供和分配数据。

10)分类与元数据：是指用于业务元数据和技术元数据以及元模型、存储库创建通用

语义定义的方法和工具。

11)审计信息记录与报告：是指与数据审计、内部控制、合规和监控超级用户等有关 的管理流程。

可以通过回答问题来评估企业当前数据治理的成熟度。例如是否已经确定了大数据治 理计划的关键业务相关人员、是否能对大数据治理提供的财务收益进行量化等问题。上述 数据治理成熟度模型如图1-6所示。



成果

数据风险管理及合规性

支持条件

组织结构与意识

管理工作 策略

核心规程

数据质量管理 信息生命周期管理 信息安全与隐私

支持规程

数据架构 分类与元数据 审计信息记录与报告

价值创造

要求

增强

支持

图1-6 IBM成熟度模型

**4.大数据治理的审计**

审计是成功实施大数据治理的基础，以第三方的客观立场对大数据治理过程进行综合

检查、监督和评价，并给出详细的、有价值的审计意见，促进大数据治理的规范性，保证

18 第一篇 理 论 篇

大数据的一致性、可靠性、有效性和安全性，进一步提升大数据的利用价值，有助于对大 数据治理实施提供指导性意见，以及为企业发展的战略决策提供可靠依据。大数据治理审 计不仅可以提高大数据治理的实施水平，从更全面的视角为大数据治理提供实施意见，而 且大数据治理审计还可以满足企业监管的需要，可以改善大数据在治理过程中的安全和隐 私。大数据审计的对象称为审计客体，即在数据治理工程中被作用的对象，这类对象不仅 包括大数据治理的整个生命周期，还应该涵盖大数据在治理过程中的中间产物，以及进行 大数据治理的实施环境。大数据审计的内容主要包含数据一致性的审计、数据风险的审计、 数据安全与隐私的审计、数据处理过程的审计、数据质量的审计、数据生命周期的审计等。 进行大数据治理的审计主要是让企业了解大数据治理活动的总体情况，对企业数据的总体 价值利用情况进行把握，提前准备应对数据治理过程中的相关风险，并提出评价意见和改 进意见，可以提供给组织用于改善经营管理，促进实现大数据治理的目标。

总之，大数据治理的审计工作在最大化实现数据价值过程中必不可少，它能够全面地、 在更高层次来评价企业的大数据治理情况，客观地显示大数据治理的生命周期管理水平， 从而提高企业预防大数据风险的能力，满足企业发展的需要。



■ ■

■ ■

区 ■ ■ ■ 部面■|1■1

■ ■ ■

■ 图 ■|! 厘 ■■| 露 ■

■■

■■|  ■1重!

■ ■

■ 置 ■|

■ ■ ■

■

■ 四

■

第2章

**大数据安全、隐私保护和审计技术**

**2.1** **大数据安全**

大数据(Big Data) 指的是所涉及的资料量规模巨大到无法通过目前主流软件工具，在 合理时间内达到撷取、管理、处理并整理成为帮助企业经营决策目的的信息。大数据是当 下最火热的IT行业的词汇，随之数据仓库、数据安全、数据分析、数据挖掘等围绕大数据 的商业价值的利用将逐渐成为行业人士争相追捧的利润焦点。本节介绍大数据安全的意义 和重要作用、大数据面临的问题与挑战，以及大数据安全防护的主要技术。

**2.1.1** **大数据安全的意义和重要作用**

为什么要研究大数据?在开始了解大数据安全之前，需要先搞清楚这个问题。当今， 社会信息化和网络化的发展导致数据的爆炸式增长，据统计，平均每秒有200万用户在使 用谷歌搜索，各行业也在不断产生大量数据。在科学界，《Nature》和《Science》都推出了 大数据专栏对其展开探讨，这意味着大数据将成为云计算之后的信息技术领域的另一个信 息产业增长点。

现阶段，国家十分支持大数据的发展，国务院以及各级地方政府从2012年开始颁布了 大量政策来扶持大数据产业。从现有的政策来看，大数据的发展已经被列为国家发展战略 了，大数据的重要性不言而喻。大数据已经得到政府高层、互联网企业以及其他各个行业 企业的认可，对大数据的开发和应用的力度也相应加大。近年来，我国高度重视大数据发 展，仅2015年最高层面就发出了多次重视大数据的声音。2015年5月，李克强总理提出， 大数据产业是中国推动“互联网+”战略的重要支撑。2015年6月，习近平主席考察贵阳， 调研贵阳大数据交易所时说发展大数据确实有道理。在政策层面，2015年9月，国务院通

20 第一篇 理 论 篇

过《关于促进大数据发展的行动纲要》,这是支持大数据发展的第一部正式国家层面文件， 对大数据的规范化发展起到了至关重要的作用。

大数据对企业的影响也是巨大的，正是大数据对企业所产生的立竿见影的效果，现在 已经得到更多公司的重视。首先，大数据能够彻底改变企业内部运作模式，以往的管理是 “领导怎么说?”,现在变成“大数据的分析结果是什么?”。这是对传统领导力的挑战，也 推动了对企业管理岗位人才的重新定义。企业管理人才不仅要懂企业的业务流程，还要成 为数据专家，跨专业的要求改变了过去领导力主要体现在经验和过往业绩上，如今新的要 求是熟练掌握大数据分析工具，善于运用大数据分析结果，并结合企业的销售和运营管理 实践。当然大数据对企业的作用中一个不可回避的关键因素是数据的质量，有句话叫“垃 圾进，垃圾出”,是说如果采集的是大量垃圾数据，则会导致产生的分析结果也是毫无意义 的垃圾。

大数据也在影响着我们每个人的生活，使得一些服务更加贴近大家的生活。打开浏览 器上网，广告弹窗推荐的商品可能正好就是你最近想买的东西。翻阅自己的微博，查看定 位信息就能够准确回忆起一年前的今天你在哪里，做了什么。在搜索引擎中输入几个关于 自己的关键词，也许可以重温你在10年前写下的网络日志。进入淘宝网，它就能根据你的 历史浏览记录为你贴心地推荐你想要的商品。大数据现在已经进入人们的生活。

正是因为大数据对国家、企业、个人具有重要的作用，并具有很高的研究价值，所以 大数据安全现在成为学术与工业界的研究热点，是人们公认的大数据相关问题中关键的问 题之一。没有安全，发展就是空谈，数据安全是发展大数据的前提，必须将它摆在更加重 要的位置。我们在使用和发展大数据的同时，也容易出现大数据引发的个人隐私安全、企 业信息安全乃至国家安全问题。

1)与大数据安全及个人关系最密切的就是个人隐私安全，在大数据时代，想屏蔽外部 数据商挖掘个人信息是不可能的。目前，各社交网站均不同程度地开放其用户所产生的实 时数据，这些数据被一些数据提供商收集，还出现了一些监测数据的市场分析机构。通过 人们在社交网站中写入的信息、智能手机显示的位置信息等多种数据组合，已经可以以非 常高的精度锁定个人，挖掘出个人信息体系，因此，用户隐私问题堪忧。据统计，通过分 析用户4个曾经到过的位置，就可以识别出95%的用户。“你没有隐私，忘记这事吧。”有 数据统计，中国78.2%的网民个人身份信息被泄露过，包括姓名、学历、家庭住址、身份 证号及工作单位等。其中，82.3%的网民亲身感受到了个人信息泄露给日常生活造成的不良 影响。

2)企业迈进大数据时代，信息安全面临多重挑战。企业在获得“大数据时代”信息价 值增益的同时，也在不断地累积风险，大数据安全方面的挑战日益增大。首先是黑客窃密 与病毒木马对企业信息系统的入侵，大数据在云系统中进行上传、下载、交换的同时，极 易成为黑客与病毒的攻击对象。而“大数据”一旦被入侵并产生泄密，就会对企业的品牌、 信誉、研发、销售等多方面带来严重冲击，并带来难以估量的损失。通常，那些对大数据

第2章 大数据安全、隐私保护和审计技术 21

分析有较高要求的企业，会面临更多的挑战，例如电子商务、金融、天气预报的分析预测、 复杂网络计算和广域网感知等。任何一个会误导目标信息提取和检索的攻击都是有效攻击， 因为这些攻击会对安全厂商的大数据安全分析产生误导，导致其分析偏离正确的检测方向。 这些攻击需要我们集合大量数据，进行关联分析才能够知道其攻击意图。大数据安全是与 大数据业务相对应的，传统时代的安全防护思路此时难以奏效，并且成本过高。无论是从 防范黑客对数据的恶意攻击，还是从对内部数据的安全管控角度，为了保障企业信息安全， 迫切需要一种更为有效的方法对企业大数据安全进行有效管理。

3)大数据时代，国家安全将受到信息战与网络恐怖主义的威胁，大数据安全的重要性 在国家层面也需要得到重视。如今的信息时代，安全环境发生了质的变化。不管是战争时 期还是和平年代， 一国的各种信息设施和重要机构等都可能成为打击目标，而且保护它们 免受攻击已超出了军事职权和能力的范围。决策的不可靠性、信息自身的不安全性、网络 的脆弱性、攻击者数量的激增、军事战略作用的下降和地理作用的消失等，都使国家安全 受到了严峻的挑战。此外，大数据也使网络恐怖主义者有了可乘之机。庞大海量的大数据 涉及面广泛，将有可能使网络恐怖主义的势力侵入人们生活的方方面面。大数据对国家安 全的影响涉及了国家安全内容的诸多方面，我们平时关注比较多的有科技安全、信息安全， 其实大数据安全对国民安全、政治安全、意识形态安全、社会公共安全等的影响也很大。

大数据的发展给我们带来了机遇，但是也带来了挑战。大数据已经影响到个人、企业、 国家，对整个社会都有很重要的影响，在享受大数据的便利的同时我们必须重视大数据 安全。

**2.1.2** **大数据安全面临的问题与挑战**

“世界的本质是数据，大数据开启了一次重大的时代转型，也是一场生活、工作与思维 的大变革。”随着世界各国在陆、海、空、天、电、网多维度战略的部署，信息技术爆炸式 发展。基于大数据发展对国家、社会的组织结构和治理模式，对商业、企业的决策方式和 业务策略，对个人的生活、思维方式等各方面产生的深刻影响，各界逐渐开始关注“信息” 本身而不只是“技术”了。在大数据时代，人类信息管理准则也将面临重新定位，而在信 息安全问题日益突出的当下，大数据在给信息安全带来新挑战的同时，也为信息安全领域 的发展带来新机遇。

**1.大数据成为网络攻击的显著目标**

在网络空间中，大数据成为更容易被“发现”的大目标，承载着越来越多的关注度。 大数据自身规模大且集中的特点使得其在网络空间中无疑是一个更易被“发现”“命中”的 大目标，低成本、高收益的攻击效果对黑客而言是充满诱惑力的。 一方面，大数据不仅意 味着海量的数据，也意味着更复杂、更敏感的数据，这些数据成为更具吸引力的目标，会 吸引更多的潜在攻击者。另一方面，数据的大量聚集，使得黑客通过一次成功的攻击就能 够获得更多的数据，无形中降低了黑客的攻击成本，增加了“收益率”。

**22** **第一篇** **理** **论** **篇**

**2.大数据加大隐私泄露风险**

网络空间中的数据来源涵盖非常广阔的范围，例如传感器、社交网络、记录存档、电 子邮件等，大量数据的聚集不可避免地加大了用户隐私泄露的风险。 一方面，大量数据聚 集，包括大量的企业运营数据、客户信息、个人的隐私和各种行为的细节记录，这些数据 的集中存储增加了数据泄露风险，而这些数据不被滥用，成为人身安全的一部分。另一方 面， 一些敏感数据的所有权和使用权并没有明确的界定，很多基于大数据的分析都未考虑 其中涉及的个体的隐私问题。

从个人隐私的角度而言，用户在互联网中产生的数据具有累积性和关联性，单点信息 可能不会暴露隐私，但如果采用大数据关联性抽取和集成有关某用户的多点信息并进行汇 聚分析，其隐私泄露的风险将大大增加，关联性利用类似于现实生活中通过“人肉搜索” 将某人或事物暴露。

从企业、政府等大的角度而言，大数据安全标准体系尚不完善，隐私保护技术和相关 法律法规尚不健全，加之大数据所有权和使用权出现分离，使得数据公开和隐私保护很难 做到友好协调。在数据的合法使用者利用大数据技术收集、分析和挖掘有价值信息的同时， 攻击者也同样可以利用大数据技术最大限度地获取他们想要的信息，这无疑增加了企业和 政府敏感信息泄露的风险。

从大数据基础技术的角度而言，无论是被公认为大数据标准开源软件的 Hadoop, 还是 大数据依托的数据库基础 NoSQL, 其本身均存在数据安全隐患。Hadoop 作为一个分布式系 统架构对数据的汇聚增加数据泄露风险的同时，作为一个云平台也存在着云计算面临的访 问控制问题，其派生的新数据也面临加密问题。NoSQL 技术将不同系统、不同应用和不同 活动的数据进行关联，加大了隐私泄露风险。又由于数据的多元非结构化，使得企业很难 对其中的敏感信息进行定位和保护。

**3.大数据对现有的存储和安防措施提出挑战**

大数据存储带来新的安全问题。大数据集中的后果是复杂多样的数据存储在一起，例 如开发数据、客户资料和经营数据存储在一起，可能会出现违规地将某些生产数据放在经 营数据存储位置的情况，造成企业安全管理不合规。大数据的大小影响到安全控制措施能 否正确运行。对于海量数据，常规的安全扫描手段需要耗费过多的时间，已经无法满足安 全需求。安全防护手段的更新升级速度无法跟上数据量非线性增长的步伐，大数据安全防 护存在漏洞。

**4.大数据技术被应用到攻击手段中**

在企业用数据挖掘和数据分析等大数据技术获取商业价值的同时，黑客也在利用这些 大数据技术向企业发起攻击。黑客最大限度地收集更多有用信息，比如社交网络、邮件、 微博、电子商务、电话和家庭住址等信息，为发起攻击做准备，大数据分析让黑客的攻击 更精准。此外，大数据为黑客发起攻击提供了更多机会。黑客利用大数据发起“僵尸网络

第2章 大数据安全、隐私保护和审计技术 23

攻击”,可能会同时控制上百万台“傀儡机”并发起攻击，这个数量级是传统单点攻击不具 备的。

**5.大数据成为高级可持续攻击的载体**

黑客利用大数据将攻击很好地隐藏起来，用传统的防护策略难以检测出来。传统的检 测是在单个时间点进行的基于威胁特征的实时匹配检测，而高级可持续攻击(APT) 是一个 实施过程，并不具有能够被实时检测出来的明显特征，无法被实时检测。同时， APT 攻击 代码隐藏在大量数据中，很难被发现。此外，大数据的价值低密度性，让安全分析工具很 难聚焦在价值点上，黑客可以将攻击隐藏在大数据中，给安全服务提供商的分析造成很大 困难。黑客设置的任何一个会误导安全厂商目标信息提取和检索的攻击，都会导致安全监 测偏离应有的方向。

**6.大数据技术为信息安全提供新支撑**

大数据在带来新安全风险的同时也为信息安全的发展提供了新机遇。大数据正在为安 全分析提供新的可能性，对于海量数据的分析有助于信息安全服务提供商更好地刻画网络 异常行为，从而找出数据中的风险点。对实时安全和商务数据结合在一起的数据进行预防 性的分析，以便识别钓鱼攻击，防止诈骗和阻止黑客入侵。网络攻击行为总会留下蛛丝马 迹，这些痕迹都以数据的形式隐藏在大数据中，利用大数据技术整合计算和处理资源有助 于更有针对性地应对信息安全威胁，使得网络攻击行为无所遁形，有助于找到发起攻击的 源头。

**7.大数据对信息安全的合规性要求**

大数据时代出现数据拥有权和使用权的分离，数据经常脱离数据拥有者的控制范围而 活跃着，这就对数据需求合规性和用户授权合规性提出新的要求，包括数据形态和转移方 式的合规性。数据需求方为精准开展一个业务，要求数据拥有者提供原始敏感数据或未脱 敏的统计类数据，显然这违背了信息安全的本意。就算数据需求遵循最小级原则，对数据 的提供未超出合理范围，用户授权仍是数据服务的前提，包括转移数据使用的目的、范围、 方式以及授权信息的保存等各个环节。

在对信息安全提出合规性要求的同时，引入第三方的标准符合性审查服务也很必要。 如通过针对数据提供者和接受者双方的审查，包括文档资料安全规范的审查，技术辅助现 场审查，在供方和需方之间做扫描和数据检测，提供第三方公平的数据安全审查服务。

**2.1.3** **大数据安全防护技术**

**1.数据发布匿名保护技术**

数据发布匿名保护技术是对大数据中的结构化数据实现隐私保护的关键技术手段。匿 名化的处理过程可以用图2-1简单表示。具体来说，数据库对所有人都是公开的，任何人 都可以自由访问，但是却不能将数据库中的任一记录对应到具体某一个体上。为了对数据

24 第一篇 理 论 篇

匿 名 化 預 处 理

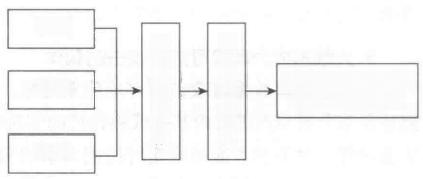
收集原始数据

表中的数据进行隐私保护，自由访问型隐私保护通常采取的办法是对原始数据实施“数据

匿名化”操作。所谓“数据匿名化”就是数

张三

据发布者在数据发布前需要对真实数据表实

攻一 李四

对应的敏感信息，从而实现对个体隐私信息 王五

的隐藏。

图2-1 匿名化处理过程

**2.社交网络匿名保护技术**

因为用户的个性化信息与用户隐私密切相关，所以互联网服务提供商一般会对用户数 据进行“数据匿名化”之后再提供共享或对外发布。表面上看，活跃于社交网络上的信息 并不泄露个人隐私。但事实上，几乎任何类型的数据都如同用户的指纹一样，能通过辨识 找到其拥有者。在当今社会， 一旦用户的通话记录、电子邮件、银行账户、信用卡信息、 医疗信息等大规模数据被无节制地搜集、分析与交易利用，那么用户都将“被透明”,不仅 个人隐私荡然无存，还将引发一系列社会问题。因此深入理解社交网络的匿名化和去匿名 化这一对相互依存的博弈过程，才能更好地在社交网络活动中保护好用户的隐私，这个问 题已成为当前大众关注的焦点。社交网络中典型的匿名保护技术如下：

1)用户标识匿名与属性匿名保护，在数据发布时隐藏了用户的标识与属性信息。属性 数据在社交网络上变化最频繁，内容最丰富，它生动地描述了用户的个性化特征，能够帮 助系统建立完整的用户轮廓，提高推荐系统的准确性。然而，用户往往不希望将所有属性 信息对外公开。例如：用户观看私密视频的记录被曝光，会对用户的网络形象造成最直接 的破坏，甚至影响用户的正常生活。属性隐私保护要求对社交网络的属性信息进行匿名化 处理，阻止攻击者对用户的属性隐私进行窥探。

2)用户间关系匿名保护，在数据发布时隐藏了用户间的关系。社交关系数据本身蕴 含着巨大的价值。互联网服务提供商可基于用户现有的社交结构分析用户的交友倾向、向 用户推荐朋友等，有助于保持社交群体的活跃度和黏性。但是与此同时，分析者也可以挖 掘出用户不愿公开的社交关系、交友群体特征，从而导致用户的社交关系隐私暴露。为此， 社交关系隐私保护要求节点对应的社交关系保持匿名，使攻击者无法确认特定用户拥有哪 些社交关系。

**3.数据水印技术**

数据水印技术是指将标识信息以难以察觉的方式嵌入数据载体内部且不影响其使用的 方法，多见于多媒体数据的版权保护，也有针对数据库和文本文件的水印方案。当然，实 现数据水印技术的前提是，数据中存在冗余信息或可容忍一定的精度误差。数据水印技术 按照不同的划分方法有不同的分类，在大数据领域，比较常用的是按照特性划分为鲁棒数 字水印和易损数字水印两类，鲁棒数字水印可用于大数据起源证明，易损数字水印可用于

第2章 大数据安全、隐私保护和审计技术 25

证明数据的真实性。

鲁棒数字水印主要用在数字作品中标识著作权信息，利用这种水印技术可以在多媒体 内容的数据中嵌入创建者、所有者的标识信息，或者嵌入购买者的标识(即序列号)。在发 生版权纠纷时，创建者或所有者的信息用于标识数据的版权所有者，而序列号用于追踪违 反协议而为盗版提供多媒体数据的用户。用于版权保护的数字水印要求有很强的鲁棒性和 安全性，除了要求在一般图像处理(如滤波、加噪声、替换、压缩等)中生存外，还需能抵 抗一些恶意攻击。

易损水印与鲁棒水印的要求相反，它主要用于完整性保护。这种水印同样是在内容数 据中嵌入不可见的信息，当内容发生改变时，这些水印信息会发生相应的改变，从而可以 鉴定原始数据是否被篡改。易损水印应对一般图像处理(如滤波、加噪声、替换、压缩等) 有较强的免疫能力(鲁棒性),同时又要有较强的敏感性，既允许一定程度的失真，又要能 将失真情况探测出来。它必须对信号的改动很敏感，人们才能根据易损水印的状态判断出 数据是否被篡改过。

**4.数据溯源技术**

数据溯源技术的目标是帮助人们确定数据仓库中各项数据的来源，也可用于文件的溯 源与恢复。数据溯源技术的意义在于根据追踪路径重现数据的历史、状态和演变过程，实 现数据历史档案的追溯。目前数据溯源的基本方法包括标注法和反向查询法θ。

标注法是一种简单且有效的数据溯源方法，使用非常广泛。它通过记录相关的信息来 追溯数据的历史状态，即用标注的方式来记录原始数据的一些重要信息，如背景、作者、 时间、出处等，并将标注和数据一起传播，通过查看目标数据的标注来获得数据的溯源。

反向查询法，有的文献也称为逆置函数法。由于标注法并不适合细粒度数据，特别是 大数据集中的数据溯源，于是，有人提出了反向查询法，此方法是通过逆向查询或构造逆 向函数对查询求逆，或者说根据转换过程反向推导，由结果追溯到原数据。这种方法是在 需要时才计算，所以又叫 lazzy 方法。

标注法和反向查询法各有优缺点，在实际使用时需要根据具体的情况进行选择。标注 法的优点是实现简单，容易管理；缺点是只适合小型系统，对于大型系统而言很难为细粒 度的数据提供详细的数据溯源信息，因为很细可能导致元数据比原始数据还多，需要额外 的存储空间，会对存储造成很大的压力，而且效率低。反向查询法的优点是追踪比较简单， 只需存储少量的元数据就可实现对数据的溯源，不需要存储中间处理信息、全过程的注释 信息；缺点是用户需要提供逆置函数(并不是所有的函数都具有可逆性)和相对应的验证函 数，构造逆置函数具有一定局限性，实现起来相对比较复杂。

**5.访问控制技术**

访问控制 (Access Control) 指系统限制用户身份及其所属的预先定义的策略组使用某

日 明华，张勇，符小辉.数据溯源技术综述[J].小型微型计算机系统，2012,33(9):47-53.

26 第 一 篇 理 论 篇

些数据资源的手段。这在数据库领域已经是很成熟的技术，在大数据安全领域也有很多访 问控制技术得到了广泛的应用，主要包括基于角色的访问控制、基于属性的访问控制和风 险自适应的访问控制技术。

基于角色的访问控制技术是应用最广泛的技术，该方法给不同角色赋予不同的访问控 制权限。其基本思想是，不是将系统操作的各种权限直接授予具体的用户，而是在用户集 合与权限集合之间建立一个角色集合，每一种角色对应一组相应的权限。 一旦用户被分配 了适当的角色，该用户就拥有此角色的所有操作权限。这样做的好处是，不必在每次创建 用户时都进行分配权限的操作，只要分配用户相应的角色即可，而且角色的权限变更比用 户的权限变更要少得多，这样将简化用户的权限管理，减少系统的开销。

基于属性的访问控制技术是通过综合考虑各类属性(如用户属性、资源属性、环境属性 等)来设定用户的访问权限。基于属性的访问控制技术实现了细粒度的权限控制，所有实体 的描述都采用同一种方式——属性来进行描述，但不同实体的属性权限可能不同，这使得 访问控制判定功能在判定时能够进行统一处理。在基于属性的访问控制中，访问判定是基 于请求者和资源具有的属性，请求者和资源在基于属性的访问控制技术中通过特性来标识， 而不是通过ID 来标识，这使得传统的基于身份的访问控制具有足够的灵活性和可扩展性， 同时使得安全的匿名访问成为可能，这在大型分布式环境下是十分重要的。

风险自适应的访问控制是针对大数据场景推荐的一种访问控制方法。风险自适应的访 问控制针对的是在大数据场景中，安全管理员可能缺乏足够的专业知识，无法准确地为用 户指定其可以访问的数据的情况。在大数据场景中，数据种类和来源复杂，用户角色也十 分复杂，往往无法准确地为用户预先指定其可以访问的数据，最好是在某个访问行为发生 时针对具体上下文进行判断，自适应的访问控制正是这样一种上下文敏感的动态系统安全 访问技术。

**2.2** **大数据隐私保护**

大数据是当前学术界以及产业界的研究热点，它正在潜移默化地影响着人们日常的工 作、学习以及娱乐方式。但是目前大数据在收集、存储和处理的过程中面临着诸多挑战， 大数据时代隐私保护问题日渐突出。收集和分析用户浏览数据，可以提高广告效应；收集 用户位置数据，可为附近商户引流。人们能想到的用户行为和隐私数据，事实上都有可能 在利益驱使下被多方知名或不知名团体收集。即使下载一款简单应用程序，也有可能在授 权时被具有不同目的的机构获取隐私数据，用于商业或其他目的。大数据所导致的隐私泄 露给用户带来了严重困扰，也给各商家带来了信任危机。

**2.2.1** **大数据隐私保护的意义和重要作用**

随着互联网技术的迅速发展，整个社会被迫进入“大数据”时代。不管人们愿意与否，

第2章 大数据安全、隐私保护和审计技术 27

用户的个人数据总是无意中被动地被某些个人和公司收集并利用。大数据爆炸式的发展席 卷了全球IT 、零售、交通等行业，给这些行业带来了巨大的变革，同时也改变了人们的生 活。大数据时代的一个特点就是可以将用户保留在互联网中的敏感数据转化为有价值的资 源，通过对这些数据的分析，无论是商家、保险公司以及其他以服务为导向的公司，都可 以提供更贴心、更多的个人服务，他们甚至比我们都“了解”自己。个人数据共享化以及 透明化已成为不可阻挡的趋势。过去，具有公共权力的政府机构合法地掌握着大量公民的 个人隐私数据，但现在很多公司和个人拥有了大量来自互联网的个人隐私数据，在某些方 面甚至可能比政府机构掌握的数据还要细致和全面。对企业而言，用户数据是宝贵的资源， 因为可以通过数据挖掘和机器学习从中获取丰富的有用价值。同时，用户的隐私数据也是 危险的“潘多拉的盒子”,一旦数据发生泄露，用户的个人隐私将被侵犯。近年来不断发生 的隐私泄露事件提醒着我们，公民个人隐私保护已面临严重的挑战。大量数据的分析和使 用，使得人们的生活更方便，同时，越来越多的人担心隐私泄露的问题。

大数据的快速发展面临着与个人隐私保护需求发生冲突的矛盾。根据 Wikibon 的报告， 在2017年，美国大数据产业的市场规模将达到500亿美元，如图2-2所示。这充分证明了 大数据将为美国的很多行业带来巨大的价值。

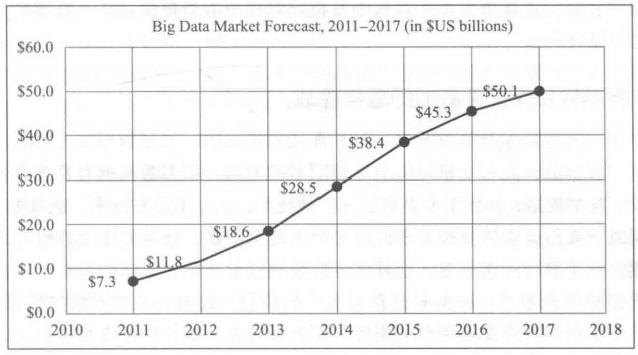


图2-2 美国大数据市场的预测(来源： Wikibon 2014)

大数据所带来的巨大经济价值吸引着各行各业的人，但是大数据是把双刃剑，频繁出 现的隐私泄露事件不仅给相关从业者敲响了警钟，也时刻提醒着用户在享受大数据应用带 来的各种便利的同时，需要更加注重保护个人隐私。 一些机构通过大数据收集个人信息并 使用，已经造成了一系列的问题。

近年来，大数据安全事件呈现高发之势。中国互联网协会发布的《2015中国网民权益 保护调查报告》显示，63.4%的个人网上活动信息被泄露过，78.2%的个人身份信息被泄露 过，因个人信息泄露、垃圾信息、诈骗信息等现象导致的总体经济损失达到了805亿元。

28 第一篇 理 论 篇

2015年8月，线上票务营销平台大麦网被发现存在安全漏洞，600余万用户账户和密 码遭到泄露，并且这些隐私数据已被黑产行业进行售卖与传播。

2016年10月，我国警方在广东破获一起高科技经济犯罪案件，犯罪嫌疑人攻破了多个 商业银行网站，并且窃取了储户的身份证号、银行卡号、支付密码等数据，同时组织一批 人在网上大肆盗刷别人的信用卡，涉案金额近15亿元，涉及银行49家。

以上案例说明： 一些企业和个人利用职务和技术之便，蓄意搜集公民个人隐私数据， 然后出售给有需要的客户，达到自己赚钱的目的，或由自己直接实施欺诈行为以获利。而 购买他人数据隐私信息者，则利用购得的信息发送广告，进行产品推广，或者进行诈骗等 犯罪活动。

大数据环境下，类似的大数据隐私泄露安全事件不胜枚举。这些安全事件不仅对用户 造成了严重的安全隐患，同时也给相关行业带来了巨大的经济损失。保护大数据隐私在提 倡“开放、共享”的大数据时代显得格格不入，但是对于大数据隐私的保护已经迫在眉睫。 大数据隐私保护不仅对于用户有重大意义，同时相关企业也可以从用户群中获取更多的信 任感，从而获取更多用户的数据，并且发掘出用户的潜在需求，不断改善产品，获取用户 好感。大数据隐私带来的保护措施，保证了数据获取与利用的可靠性，企业可以更容易地 从用户中获取数据，也能更好地从数据中发掘出对用户有价值的信息，反馈给用户，实现 大数据的良性循环利用。

**2.2.2** **大数据隐私保护面临的问题与挑战**

大数据研究需要数据开放与共享。但是在现实情况中，大量数据处于闲置、无人问津 的状态。因为数据的开放和共享可能会导致隐私的泄露，很多数据拥有者或者管理者不敢 或不愿开放、共享数据，导致工业界有数据、缺技术，而学术界有技术、缺数据。

与此同时，人们面临的威胁并不仅限于个人隐私泄露，还有基于大数据对人们状态和 行为的预测。 一个典型的例子是，国外某零售商通过对历史记录进行分析，比家长更早知 道其女儿已经怀孕的事实，并向其邮寄相关广告信息。此外，社交网络分析研究也表明， 可以通过其中的群组特性发现用户的属性。例如通过分析用户的社交平台信息，可以发现 用户的生活方式、消费习惯以及业务爱好等。

同时，搭建大数据环境，无论是硬件还是软件或者其他细节，都需要投入大量的资金， 因此大部分公司不会去花费精力搭建自己的大数据基础设施，所以微软的Azure 和阿里的 阿里云等成了一部分公司搭建云计算平台的首选。在这种情况下，云计算的安全性也需要 考虑。θ

在这样的背景下，隐私保护被提出，也是大数据应用成功的关键因素之一。可以通过 隐私保护技术，寻求一些使用数据的方法，既不妨碍第三方从带有敏感信息的数据集中获

θ <htp://www.business.com/technology/privacy-and-security-issues-in-the-age-of-big-data/>

第2章 大数据安全、隐私保护和审计技术 29

取信息，同时又可避免隐私资料的泄露。

大数据时代的隐私是由于数据融合、数据分析、数据过度收集等造成的，这与传统的 隐私泄露问题有本质的不同。大数据隐私管理要服务于数据治理的需要，其本质是要保证 数据的正确使用和交易。目前大数据隐私保护面临以下几个问题：

1)大数据依托的NoSQL (非关系型数据库)缺乏数据安全机制。从基础存储技术角度 来看，大数据依托的基础技术是NoSQL 。当前广泛应用的SQL (关系型数据库)技术，经 过长期改进和完善，在维护数据安全方面已经形成严格的权限访问控制和隐私管理工具。 而在NoSQL 技术中，并没有这样严格的要求。大数据的数据来源和承载方式多种多样，数 据处于分散的状态，使企业很难定位和保护所有这些私密数据。NoSQL 允许不断对数据记 录添加属性，其前瞻安全性变得非常重要，同时也对数据库管理员提出了新的要求。

2)社会工程学攻击带来的安全问题。社会工程学的特点是：不需要专业技术，成本低， 效率高。该攻击与其他攻击的最大不同是其攻击手段不是利用高超的攻击技术，而是利用 受害者的心理弱点进行攻击。因为不管大数据多么庞大总也少不了人的管理，如果人的信 息安全意识淡薄，那么即使在技术上防护手段已做到无懈可击，也没办法有效保障数据安 全。由于大数据的海量性、复杂性，以及攻击目标不明确，因此攻击者为了提高效率，经 常采用社会工程学攻击。此类攻击的案例很多，如黑客先攻击某论坛的网站，使用户无法 正常登录。然后再假冒管理员，以维护网站名义向用户发送提醒信息，索要用户的账号和 密码， 一般安全意识不强的用户此时会将密码和账号发送给黑客。除此以外，还有采用冒 充中奖、假冒社交好友、信用卡挂失等欺诈手段获得合法用户信息。

3)软件后门。在软件定义世界的时代，软件是IT 系统的核心，也就是大数据的核心， 所有的后门可能都是开放在软件上面的。据了解， IBM 、EMC 等各大巨头生产制造的存储、 服务器、运算设备等硬件产品，几乎都是全球代工的，在信息安全的监听(硬件)方面是很 难做手脚的。换句话说，软件才是信息安全的软肋所在。软件供应方在软件上设计了特殊 的路径处理，测试人员只按照协议上的功能进行测试，根本就无法察觉软件预留的监听后 门。换言之，如果没有自主可控的信息安全检测方案，各种安全机制和加密措施就都形同 虚设。此类安全事件繁多，比如，2015年，由于部分iOS 开发者使用了非官方渠道的带后 门的开发软件Xcode, 导致微信、网易云音乐等iOS 版本的应用软件存在后门程序，该程 序可以上传用户的系统信息到黑客指定的服务器，对大量应用软件造成了巨大的安全隐患。 所以，近期代码审计在安全领域是非常重要的。对于现代信息安全而言，最危险的行为是 将自主控制的权力交给“他人”。这就好比将自家的钥匙全部交到了外人手里，安全问题又 从何谈起呢?

4)大数据存储问题。大数据会使数据量呈非线性增长，而复杂多样的数据集中存储在 一起，多种应用的并发运行以及频繁无序的使用状况，有可能会导致数据类别存放错位的 情况，造成数据存储管理混乱或信息安全管理不合规范。现有的存储和安全控制措施无法 满足大数据安全需求，安全防护手段如果不能与大数据存储和应用安全需求同步升级，就

30 第 一篇 理 论篇

会出现大数据存储安全防护的漏洞。

5)文件的安全面临极大挑战。文件是整个数据和系统运行的核心。大多数的用户文 件都是在第三方的运行平台中存储和处理的，这些文件往往包含了很多部门和个人的敏感 信息，其安全性和隐私性自然成为重要的问题。尽管文件的保护提供了对文件的访问控制 和授权，例如Linux 自带的文件访问控制机制，通过文件访问控制列表来限制程序对文件 的操作。然而大部分文件保护机制都存在一定程度上的安全问题，它们通常使用操作系统 的功能来实现完整性验证机制，因此只依赖于操作系统本身的安全性。但是作为网络攻击， 操作系统才是最大的一个攻击点。

6)大数据安全传输的问题。大数据安全传输的问题涉及通信网络的安全、用户兴趣模 型的使用安全和私有数据的访问控制安全，既包括传统搜索过程中可能出现的网络安全威 胁，比如相关信息在网络传输时被窃听，以及恶意木马、钓鱼网站等，也包括服务器端利 用通信网络获取用户隐私的危险。

7)大数据支撑平台云计算安全。云计算的核心安全问题是用户不再对数据和环境拥有 完全且直接的控制权，云计算的出现彻底打破了地域的概念，数据不再存放于某个确定的 物理节点，而是由服务商动态提供存储空间。这些空间有可能是现实的，也可能是虚拟的， 甚至可能分布在不同国家及地区。用户对存放在云中的数据不能像从前那样具有完全的管 理权，相比传统的数据存储和处理方式，云计算时代的数据存储和处理，对于用户而言变 得非常不可控。云计算环境中用户数据安全与隐私保护难以实现。

8)大数据分析预测带来的用户隐私挑战。从核心价值角度来看，大数据关键在于数据 分析和利用，但数据分析技术的发展，对用户隐私带来极大的威胁。在大数据时代，想屏 蔽外部数据商挖掘个人信息几乎是不可能的。目前，各社交网站均不同程度地开放其用户 所产生的实时数据，这些数据被一些数据提供商收集，甚至是一些监测数据的市场分析机 构。例如国外的 Yelp 网站提供用户脱敏后的数据，供学者分析θ。通过人们在社交网站中 写入的信息、智能手机显示的位置信息等多种数据组合，已经可以以非常高的精度锁定个 人，挖掘出个人信息体系，使得用户隐私安全问题失去保障。

9)大数据共享所带来的安全性问题。我们不知道该如何分享私人数据，才能既保证 数据隐私不被泄露，又保证数据的正常使用。真实数据大部分不是静态的，而是越变越大， 并且随着时间的变化而变化。许多在线服务要求人们共享私人信息，但是，在记录级的访 问控制之外，人们根本不知道共享数据意味着什么,不知道共享后的数据会怎样被连接起 来，更不知道如何让用户对共享后的数据仍能进行细粒度控制。

10)大数据访问控制的安全性问题。访问控制是实现数据受控共享的有效手段，由于 大数据可能被用于多种不同场景，其访问控制需求十分突出，难以预设角色，实现角色划 分。由于大数据应用范围广泛，它通常要被来自不同组织或部门、不同身份与目的的用户

e <https://www.yelp.com/dataset>\_challenge

第2章 大数据安全、隐私保护和审计技术 31

所访问，实施访问控制是基本需求。然而，在大数据的场景下，有大量的用户需要实施权 限管理，且用户具体的权限要求未知。面对未知的大量数据和用户，预先设置角色十分困 难。同时，难以预知每个角色的实际权限。面对大数据，安全管理员可能无法准确地为用 户指定其可以访问的数据范围，而且这样做效率不高。不同类型的大数据存在多样化的访 问控制需求。例如，在Web2.0 个人用户数据中，存在基于历史记录的访问控制；在地理地 图数据中，存在基于尺度以及数据精度的访问控制需求；在流数据处理中，存在数据时间 区间的访问控制需求等。如何统一描述与表达访问控制需求是一个挑战。

**2.2.3** **大数据隐私保护技术**

目前用户数据的收集、存储、管理与使用等均缺乏规范，更缺乏监管，主要依靠企业 的自律。用户无法确定自己的隐私在何时何地被何人使用。在实际的商业化场景中，用户 应有权决定自己的信息被如何利用。实现用户可控的隐私保护包括：数据在采集时的隐私 保护，如数据精度控制处理；数据在共享、发布时的隐私保护，如数据的匿名处理、人工 干扰等；数据在分析时的隐私保护；数据生命周期的隐私保护；隐私数据可信销毁。面对 频发的隐私泄露事件，隐私保护问题需要得到有效的解决。解决的途径包括研发技术方法 和制定法律法规。

在技术方面，隐私保护的研究方向包括威胁发现技术、大数据认证技术、数据真实性 分析技术、数据失真处理技术、数据加密技术和限制发布技术。

(1)威胁发现技术

利用该技术，企业可以超越以往的保护 (Protection) 一 检测 (Detection)— 响应 (Reaction)—恢复(Recovery)(PDRR) 模式，更主动地发现潜在的安全威胁。相比于传统技 术，基于大数据的威胁发现技术具有分析的内容范围更大的优点。

企业信息资产包括数据资产、软件资产、实物资产、人员资产、服务资产和其他为业 务提供支持的无形资产。由于传统威胁检测技术并不能覆盖这6类信息资产，因此所能发 现的威胁有限。而通过在威胁检测方面引入大数据分析技术，能全面发现针对这些信息资 产的攻击，而且分析内容的时间跨度更长。现有威胁分析技术具有内存关联性，即实时收 集数据，采用分析技术发现攻击。分析窗口通常受限于内存大小，无法应对持续性和潜伏 性攻击。而引入大数据分析技术后，威胁分析窗口可以横跨若干年的数据，因此威胁发现 能力更强，可以有效应对APT 类攻击。

相比传统技术，基于大数据的威胁发现技术还有以下优点：

1)攻击威胁的预测性。传统安全防护技术大多是在攻击发生后对攻击行为进行分析和 归类，并做出响应。而基于大数据的威胁分析，可进行超前的预判，对未发生的攻击行为 进行预防。

2)对未知威胁的检测。传统的威胁分析常由经验丰富的专业人员根据企业需求和实际 情况展开，威胁分析结果在很大程度上依赖于个人经验，分析所发现的威胁是已知的。而

32 第 一 篇 理 论 篇

大数据分析的特点是侧重于普通的关联分析，而不侧重因果分析，因此通过采用恰当的分 析模型，可发现未知威胁。

(2)大数据认证技术

大数据认证技术指的是收集用户行为和设备行为数据，并对这些数据进行分析，获得 用户行为和设备行为的特征，进而通过鉴别操作者行为及其设备行为来确定其身份。这与 传统认证技术利用用户所知秘密、所持有凭证或具有的生物特征来确认其身份有很大不同。 该技术具有如下优点：

1)攻击者很难模拟用户行为特征来通过认证，因此更加安全。利用大数据技术所能收 集的用户行为和设备行为数据是多样的，可以包括用户使用系统的时间、经常采用的设备、 设备所处物理位置，甚至是用户的操作习惯等数据。通过这些数据的分析，能够为用户勾 画一个行为特征的轮廓。而攻击者很难在方方面面都模仿用户行为，因此其与真正用户的 行为特征轮廓必然存在一个较大偏差，无法通过认证。

2)减小了用户负担。用户行为和设备行为特征数据的采集、存储和分析都由认证系统 完成。相比于传统认证技术，这极大地减轻了用户负担。例如，用户无须记忆复杂的口令， 或随身携带硬件 USBKey。可以更好地支持各系统认证机制的统一。基于大数据的认证技 术可以让用户在整个网络空间采用相同的行为特征进行身份认证，可避免传统的不同系统 采用不同认证方式，且用户所知秘密或所持凭证各不相同而带来的种种不便。

(3)数据真实性分析技术

目前，基于大数据的数据真实性分析被广泛认为是最为有效的方法。许多企业已经开 始了这方面的研究工作，如 Yahoo和Thinkmail 等利用大数据分析技术来过滤垃圾邮件； Yelp等社交点评网络用大数据分析来识别虚假评论；新浪微博等社交媒体利用大数据分析 来鉴别各类垃圾信息等。

基于大数据的数据真实性分析技术能够提高垃圾信息的鉴别能力： 一方面，引入大数 据分析可以获得更高的识别准确率。例如，对于点评网站的虚假评论，可以通过收集评论 者的大量位置信息、评论内容、评论时间等进行分析，鉴别其评论的可靠性。如果某评论 者对某品牌多个同类产品都发表了恶意评论，则其评论的真实性就值得怀疑。另一方面， 在进行大数据分析时，通过机器学习技术，可以发现更多具有新特征的垃圾信息。然而该 技术仍然面临一些困难，主要是虚假信息的定义、分析模型的构建等。

(4)数据失真处理技术

通过添加噪声等方法，使敏感数据失真但同时保持某些数据或数据属性不变，即仍然 保持某些统计方面的性质。

(5)数据加密技术

采用加密技术在数据挖掘过程中隐藏敏感数据，即使得两个或多个站点通过某种协议 完成计算后，每一方都只知道自己的输入数据和所有数据计算后的最终结果。此外，还包 括分布式匿名化，即保证站点数据隐私、收集足够的信息实现利用率尽量大的数据匿名。

第2章 大数据安全、隐私保护和审计技术 33

(6)限制发布技术

有选择地发布原始数据，不发布或者发布精度较低的敏感数据，实现隐私保护。当前 这类技术的研究集中于“数据匿名化”,保证对敏感数据及隐私的披露风险在可容忍范围内， 包括 k-anonymity 、L-diversity 、T-closeness。

最早被广泛认同的隐私保护模型是k-anonymity(k- 匿名),它由Samarati 和 Sweeney 在2002年提出，作者正是美国马萨诸塞州医疗数据隐私泄露事件的攻击者。为应对去匿名 化攻击，k-anonymity 要求发布的数据中每一条记录都要与其他至少k-1 条记录不可区分 (称为一个等价类)。当攻击者获得k-匿名处理后的数据时，将至少得到k 个不同人的记录， 进而无法做出准确的判断。参数k 表示隐私保护的强度， k 值越大，隐私保护的强度越强， 但丢失的信息也会越多，数据的可用性越低。

然而，美国康奈尔大学的Machanavajihala 等人在2006年发现了k-匿名的缺陷，即没 有对敏感属性做任何约束，攻击者可以利用背景知识攻击、再识别攻击和一致性攻击等方 法来确认敏感数据与个人的关系，从而导致隐私泄露。例如，攻击者获得k-anonymity 的数 据，如果被攻击者所在的等价类中都是艾滋病病人，那么攻击者很容易做出被攻击者肯定 患有艾滋病的判断(上述就是一致性攻击的原理)。为了防止一致性攻击，新的隐私保护模 型L-diversity 改进了 k-anonymity, 保证任意一个等价类中的敏感属性都至少有一个不同的 值。T-closeness 在L-diversity 的基础上，要求所有等价类中敏感属性的分布尽量接近该属 性的全局分布。(a,k)-匿名原则则在k-匿名的基础上，进一步保证每一个等价类中与任意 一个敏感属性值相关记录的百分比不高于a。

然而，上述隐私保护模型依然有缺陷，需要被不断改进，但同时又有新的攻击方法出 现，使得基于k- 匿名的传统隐私保护模型陷入这样一个无休止的循环中。从根本上来说， 传统隐私保护模型的缺陷在于对攻击者的背景知识和攻击模型都给出了过多的假设。但这 些假设在现实中往往并不完全成立，因此攻击者总是能够找到各种各样的攻击方法来进行 攻击。直到差分隐私的出现，这一问题才得到较好的解决。

在法律法规方面，欧美早在20世纪70年代就有专门的隐私保护法。香港地区在回归 之前就颁布实施了个人数据条例。该条例于1995年颁布，1996年12月20日生效。条例 的执行由个人数据隐私专员监督。该条例管理个人、企业、公共机构和政府部门对于在世 人士的相关数据的使用(如果这些数据可以有效识别该在世人士)。香港的个人数据条例主 要强调了数据保护的六大原则：个人数据收集的目的和方式、个人数据的准确性和数据保 留的时间、个人数据的使用、个人数据的安全性、信息基本有效可用、个人数据的访问。

我国内地虽然没有专门的隐私保护法，但在多个法律法规的条文中涉及了隐私保护，对保 护个人隐私做了间接的、原则性的规定。例如，《中华人民共和国宪法》第三十八条、第

三十九条、第四十条明确了对公民的人格尊严、住宅、通信自由和通信秘密的保护，这是 我国法律对隐私权进行保护的最根本依据。第三十八条规定：“中华人民共和国公民的人格 尊严不受侵犯。禁止用任何方法对公民进行侮辱、诽谤和诬告陷害。”这些法律规定对于保

34 第一篇 理 论 篇

护公民的隐私权具有重要意义。

**2.3** **大数据治理审计**

**2.3.1** **大数据治理审计概述**

**1.审计的含义**

审计是由国家授权或接受委托的专职机构和人员，依照国家法规、审计准则和会计理 论，运用专门的方法，对被审计单位的财政、财务收支、经营管理活动及其相关资料的真 实性、正确性、合规性、合法性、效益性进行审查和监督，评价经济责任，鉴证经济业务， 用以维护财经法纪、改善经营管理、提高经济效益的一项独立性的经济监督活动。

审计是对资料做出证据搜集及分析，以评估企业财务状况，然后就资料及一般公认准 则之间的相关程度做出结论及报告。进行审计的人员必须有独立性及具相关专业知识。常 见的财务审计有以下3种。

1)运作审计(作业审计):检讨组织的运作程序及方法，以评估其效率及效益。

2)履行审计(遵行审计):评估组织是否遵守由更高权力机构所制订的程序、守则或 规条。

3)财务报表审计：评估企业或团体的财务报表是否根据公认会计准则编制， 一般由独 立会计师进行。

在香港地区，财务报表审计亦称为核数，而会计师事务所则俗称为会计师楼。在台湾 地区，财务报表审计亦称为查核、查账。

审计工作一直得到各国政府和社会的重视。传统手工审计是通过对纸质账簿的检查来 实现这一职责的。20世纪80年代，以查账为主要手段的审计职业遇到了信息技术的挑战。 传统审计面临着“打不开账，进不了门，审不了数”的困境。随着被审计单位信息化趋向 普及，审计对象的信息化使得审计信息化成为必然。审计信息化对审计人员和审计工作的 开展提出了更高的要求。

**2.大数据时代下的审计发展趋势**

随着信息技术的发展，大数据时代为审计提供了机遇和挑战。

当今，大数据伴随着云计算、移动互联网的发展，正在对全球经济社会产生巨大的影 响。大数据给现代审计提供了新的技术和方法，要求人们把握大数据的特点，变革现代审 计的思维与技术和方法，推动大数据时代审计的发展。

大数据的精髓在于促使人们在采集、处理和使用数据时思维的转变，这些转变将改变 人们理解和研究社会经济现象的技术和方法。

1)不再依赖抽样分析方法。在大数据时代，可以收集和处理事物所有的数据。自古以 来，当面临大的样本量时，人们都依赖于抽样分析。但是，抽样分析是在信息缺乏和取得

第2章 大数据安全、隐私保护和审计技术  35

信息受限制的条件下采用的一种方法，这其实是一种人为的限制。如今，科学技术条件已 经有了很大的改善，计算机能够处理的数据量已经大大增加，所以现在人们可以收集和处 理所有的数据。

2)在大数据时代，人们不再热衷于寻找事物的因果关系，而是充分利用事物的相关关 系。寻找因果关系是人类长期发展过程中形成的习惯。相关关系也许不能准确地告知某件 事情为何会发生，但是它会提醒人们这件事情正在发生，在许多情况下，这种提醒的帮助 作用已经足够大了。

3)不再热衷于追求数据的精确度，而是追求利用数据的效率。当测量事物的能力受限 制时，人们关注的是获取最精确的结果。但是，在大数据时代，当拥有海量数据时，大数 据纷繁多样，优劣掺杂，绝对的精准不再是人们追求的主要目标，更重要的是追求数据的 及时性和使用效率。

一个拥有巨大潜力的领域是审计转型，下一代的审计师需要有IT 相关的知识以及传统 的财务审计能力。从传统的审计方法到以无缝的方式充分整合大数据分析，这将是一个巨 大的飞跃。但目前大数据治理审计仍处于起步阶段。

面对大数据所带来的新思维、新技术和方法的变革，会计、审计人员需要应时而变， 以适应思维模式及数据处理模式的变化。大数据对会计、审计发展的影响主要表现在以下 几个方面：

(1)从事后的财务报告向实时财务报告发展

传统会计工作中，会计人员只是在企业生产经营业务发生后才编制财务报告，而且财 务报告编制过程漫长，年度财务报告一般用三四个月时间才能完成编制，这严重影响了会 计信息的及时性和利用效率。随着信息技术迅速发展，越来越多的人意识到实时财务报告 的重要性，而大数据技术使实时财务报告成为可能。实时财务报告是信息技术与大数据技 术较好地交叉融合的产物，是信息化条件下会计技术和方法发展的必然产物。尤其对业务 数据和风险控制“实时性”要求较高的特定行业，如银行、证券、保险等行业，实施实时 财务报告迫在眉睫。

(2)从抽样审计模式向总体审计模式发展

抽样审计模式由于抽取样本的有限性而忽视了大量的业务活动，无法完全发现和揭示 被审计单位的重大舞弊行为，隐藏着严重的审计风险。在大数据时代，数据的跨行业、跨 企业搜集和分析，可以不用随机抽样方法，而采用搜集和分析被审计单位所有数据的总体 审计模式。大数据环境下的总体审计模式是分析与审计对象相关的所有数据，使得审计人 员可以建立总体审计的思维模式。

(3)从单一审计报告向综合审计成果应用发展

目前，审计人员的审计成果主要是提供给被审计单位的审计报告，其格式固定，内 容单一，包含的信息较少。随着大数据技术在审计中的广泛应用，审计人员的审计成果除 了审计报告外，还包括在审计过程中采集、挖掘、分析和处理的大量资料和数据，可以提

36 第一篇 理 论 篇

供给被审计单位用于改进经营管理，促进审计成果的综合应用，提高综合审计成果的应用 效果。

审计人员对大数据技术的应用，促进了审计成果的进一步综合应用。首先，审计人员 通过对审计中获取的大量数据进行汇总、归纳，从中找出内在规律、共性问题和发展趋势， 为被审计单位投资者和其他利益相关者提供数据证明、关联分析和决策建议。其次，审计 人员通过应用大数据技术，从不同的角度、不同的层面整合提炼，以满足不同层次的需求。 再次，审计人员将审计成果进行智能化留存，通过大数据技术，将问题规则化并固化到系 统中，以便于计算或判断问题发展趋势。最后，审计人员将审计成果与被审计单位进行关 联，可以减少实地审计的时间和工作量，提高审计工作的效率。

(4)从精确的数字审计向高效的数据审计发展

直到今天，审计人员的数字审计技术依然建立在精准的基础上。这种思维方式适用于 掌握“小数据量”的情况，因为需要分析的数据很少，所以审计人员必须尽可能精准地量 化被审计单位的业务。相比依赖于小数据和精确性的时代，大数据因为更强调数据的完整 性和混杂性，帮助审计人员进一步接近事情的真相，“局部”和“精确”将不再是审计人员 追求的目标，审计人员追求的是事物的“全貌”和“高效”。

在大数据环境下，传统的很多审计技术和方法显得效率低下和无法实施，大数据时 代的超大数据体量以及占相当比例的半结构化和非结构化数据的存在，已经超越了传统数 据库的管理能力，必须使用新的大数据存储、处理和检索方法。围绕大数据， 一批新兴的 数据挖掘、数据存储、数据处理与分析技术涌现出来。在实施审计时，审计人员应使用分 布式拓扑结构、云数据库、联网审计、数据挖掘等新型的技术手段和工具，以提高审计的 效率。

**3.什么是大数据治理**

在各行各业中，随处可见因数量、速度、种类和准确性结合带来的大数据问题，为了 更好地利用大数据，大数据治理逐渐提上日程。在传统系统中，数据需要先存储到关系型 数据库/数据仓库后再进行各种查询和分析，这些数据称为静态数据。而在大数据时代，除 了静态数据以外，还有很多数据对实时性要求非常高，需要在采集数据时就进行相应的处 理，处理结果存入关系型数据库/数据仓库、MPP 数据库、Hadoop 平台、各种 NoSQL 数 据库等，这些数据称为动态数据。比如高铁机车的关键零部件上装有成百上千个传感器， 每时每刻都在生成设备状态信息，企业需要实时收集这些数据并进行分析，当发现设备可 能出现问题时及时告警。再比如在电信行业，基于用户通信行为的精准营销、位置营销等， 都会实时地采集用户数据，并根据业务模型进行相应的营销活动。

数据治理是指在企业数据整个生命周期(从数据采集到数据使用直至数据存档)制定由 业务推动的数据政策、数据所有权、数据监控、数据标准以及指导方针。数据治理的重点 在于，要将数据明确地作为企业的一种资产看待。

大数据治理的核心是为业务提供持续的、可度量的价值。大数据治理人员需要定期与

第2章 大数据安全、隐私保护和审计技术 37

企业高层管理人员进行沟通，保证大数据治理计划可以持续获得支持和帮助。相信随着时 间的推移，大数据将成为主流，企业可以从海量的数据中获得更多的价值，而大数据治理 的范围和严格程度也将逐步上升。

**4.大数据治理审计的含义与目的**

大数据治理审计是指独立于审计对象的审计人员，以第三方的客观立场对大数据治理 过程进行综合检查与评价，向审计对象的最高领导层提出问题与建议的一连串活动。

大数据治理审计的目的是了解组织大数据治理活动的总体状况，对组织是否实现大数 据治理目标进行审查和评价，充分识别与评估相关治理风险，提出评价意见及改进建议， 促进组织实现大数据治理目标。

**5.大数据治理审计的特点**

大数据治理审计除了具有传统审计的权威性、客观性、公正性特点之外，还具有一些 独有的特点，主要包括：

1)与传统审计的目的不同。传统审计的目的是“对被审计单位会计报表的合法性、公 允性及会计处理方法的一贯性发表审计意见”。上面提到过，大数据治理审计的目的是对组 织是否实现大数据治理目标进行审查和评价，充分识别与评估相关风险，提出评价意见及 改进建议。

2)大数据治理审计是事前、事中和事后审计的结合体。传统审计中的财务报表审计往 往是年度审计，属于事后审计，而大数据治理审计是事前、事中和事后审计兼而有之。由 审计人员所进行的大数据治理规划审计属于事前审计，大数据治理实施过程中的审计属于 事中审计，而对其在一定期间的运作情况所进行的审计属于事后审计。

3)大数据治理审计促使传统审计模式发生改变。抽样审计是传统的审计模式，即在不 可能收集和分析被审计单位全部数据的情况下，主要依赖于抽样技术针对抽取的样本进行 审计，并由此推断审计对象的整体情况。大数据时代能够收集和分析组织的所有相关数据， 审计模式发生了改变，已从抽样审计向总体审计模式发展，即对大数据总体进行多角度的 深层次分析，以发现其中隐藏的更具价值的信息及判断总体的特征，克服了抽样审计模式 的不足。

4)运用了大数据分析技术。运用大数据分析技术是大数据治理审计特征之一，即依托 大数据分析平台，开展组织内部业务数据与财务数据的治理审计工作。

5)更重视大数据信息的安全性。在信息化高度发展的时代，大数据的安全性关系着组 织的命运、社会的稳定及国家的安全。组织应采取各种措施(如管理措施、技术措施及物理 措施)保护大数据的安全。

**2.3.2** **大数据治理审计内容**

大数据治理审计是从审计的视角对大数据治理进行监督和评价。简单来说，大数据治

38 第一篇 理 论 篇

理是为了保证数据质量，而大数据治理审计是为了保证大数据治理的质量。 大数据治理审计的内容如下。

**1.大数据治理战略目标审计**

不同行业、不同公司的大数据治理战略目标不同。对大数据治理战略目标进行审计， 一方面可以确保大数据治理的目标符合行业或者企业需求，另一方面可以向管理层提供大 数据治理战略规划，使得大数据治理过程得到控制和监督。例如，大数据治理战略目标的 审计内容为：

1)本次大数据治理的目标是什么?

2)本次大数据治理的目标是否合理?

3)本次大数据治理的目标是否符合该企业的状况?

4)是否制定了大数据治理的计划?

5)该计划是否合理?

**2.大数据治理内容审计**

大数据治理内容审计非常重要，大数据治理的目标是什么决定着大数据治理的内容。 对大数据治理内容进行审计，可以更好地建立健全大数据治理审计系统。大数据治理审计 内容如下：

1)大数据治理审计内容是什么?

2)大数据治理审计内容是否有助于实现本组织的大数据治理目标?

**3.大数据治理架构审计**

每一个大数据治理系统都有特有的框架结构，通过对大数据治理系统框架的审计，可 以在一定程度上使大数据治理系统的架构更加合理。大数据治理架构审计内容如下：

1)该企业建立的大数据治理框架结构是什么?

2)该大数据治理框架结构是否与该企业的大数据治理战略目标一致?

3)该大数据治理框架结构是否满足该企业大数据治理内容需求?

**4.大数据安全审计**

数据安全可靠是大数据的根本。如果数据都不可靠，再多的数据量也没有什么意义。 大数据安全审计通过对大数据安全相关的评价，确保大数据安全。

**5.大数据生命周期管理审计**

数据生命周期管理 (Data Life cycle Management,DLM) 是一种基于策略的方法，用 于管理信息系统的数据在整个生命周期内的流动：从创建和初始存储，到它过时被删除。 数据生命周期管理产品涉及过程自动化，通常根据指定的策略将数据组织成各个不同的层， 并基于那些关键条件自动地将数据从一个层移动到另一个层。

对大数据生命周期管理进行审计，可以将整个大数据生命周期管理的自动化过程变得

第2章 大数据安全、隐私保护和审计技术  39

更加透明，更加可靠。对于企业来说，对大数据生命周期管理进行审计，可以使整个大数 据生命周期管理机制得到控制。

**2.3.3** **大数据治理审计方法和技术**

**1.大数据治理审计标准规范**

与会计审计遵循《审计准则》一样，大数据审计需要有一套共同遵循的审计规范。物 联网、云计算快速发展带来大数据审计的需要，各国政府、协会或民间组织也积极关注并 推行大数据审计的规范。 一般说来，大数据治理审计主要存在于信息审计或云计算的审计 规范之中。当前国外主要信息审计的相关标准如下：

信息系统审计与控制基金会在1996年制定的 IT 治理模型 (CO BIT),是国际公认的、 权威的安全与信息技术管理和控制的标准，也是国际上通用的信息系统审计的标准之一。 它的宗旨是跨越业务和 IT 控制之间的鸿沟，建立一个面向业务目标的 IT 控制框架。特别 是在最新的CO BIT5.0 版本中，被称为“一个治理和管理企业IT 的业务框架”,它是IT 技 术人员、用户、企业管理人员和IT 审计师之间的桥梁日。

美国国家标准与技术学院(NIST) 不仅发布了被广泛引用的《云计算定义》,还发布了 《联邦信息系统和机构的信息安全持续监测》(ISC M) 报告，提出：通过持续监测，保持其 对信息安全、漏洞和威胁的警觉。

美国云安全联盟 CSA 在2009年12 月发布了《云安全指南》。它涵盖了“云计算重点 13个区域的安全指导”,从云用户角度阐述了可能存在的商业隐患、安全威胁，以及推荐采 取的安全措施。

ISACA 是国际信息系统审计协会在2010年推出的云计算管理审计、保证程序 (Cloud Computing Management Audit/Assurance Program), 规定了审计过程中使用的工具、模板以 及流程。同时，ISACA 还在程序中规定了审计过程中应该关注的审查点以及遵循的标准， 从而保证审计师能够完整、真实地记录有关数据。它主要关注云计算治理的影响、服务供 应商以及客户之间的合同履约、云计算控制的具体问题等。如数据审计的审计目标是：为 云计算服务提供商的客户提供对服务提供商内部控制的有效性和安全性评估；识别客户组 织其他与服务提供商的接口是否存在内部控制缺陷；评估客户的质量和能力情况与服务提 供商的内部控制项相关的证明。

其他的信息审计标准还有欧洲网络与信息安全局的《云计算风险评估方法论》、ISO 27001 等。

在我国，由于物联网与云计算等信息化发展相对落后，至今尚未有大数据治理审计的 标准，可以参考的主要有2008年五部委共同颁布的《企业内部控制规范》和2009年银监 会颁布的《商业企业信息科技风险管理指引》。

日 许金叶，许琳.大数据审计：物联网建设的制度保障[J].会计之友，2013(33):118-121.

40 **第** **一** **篇** **理** **论** **篇**

**2.大数据治理审计方法**

审计方法是指为完成审计任务、实现审计目标，所采用的各种技术手段的总称。同时， 审计方法也可以说是沟通审计主体和审计客体的桥梁，是审计程序的支柱，是审计过程趋 于合理、有效的灵魂。注重审计方法的目的在于：有利于提高审计工作效率和工作质量， 抓住问题的实质，以便更好地完成审计任务。

(1)传统审计方法

传统的审计方法又可以细分为：审查书面资料的方法、审查财产物资的方法、审计的 分析法、审计抽样方法。

审查书面资料的方法，按照资料形成的顺序，可分为顺查法、逆查法；按照审计的范 围，可分为详查法、抽查法；按照资料内容的不同，可分为审阅法、核对法、复算法、查 询法、分析法、推理法等。

审计人员在实施审计过程中，经常需要证实被审查事物的性质、形态、数量、价值等 是否真实、正确、合理， 一般采用一些特殊的方法，如盘点法、调节法、鉴定法、观察法 等。人们把这些方法称为证实客观事物的方法。

审计的分析法，也称为分析性复核法，是指审计人员在审计过程中，对审计事项的相 关指标进行对比、分析和评价，以便发现其中有无问题或异常情况，为进一步审计提供线 索的一种审计方法。常用的分析法有：比较分析法、比率分析法、平衡分析法、趋势分析 法、账户分析法、账龄分析法等。

统计抽样审计的基本程序一般分为3个阶段，即样本设计阶段、样本选取阶段和抽样 结果的评价阶段。

(2)IT 内部审计方法

审计信息化(IT 审计)也可称为信息系统审计或计算机辅助审计。IT 内部审计作为IT 治理的分支，本质是为了促进IT 治理目标的实现，实现 IT 资源的价值增值。下面主要是 选用商业银行IT 治理的案例来丰富大数据时代IT 内部审计概念。

现阶段商业银行IT 治理雏形可从以下5个核心要素进行考察：

1 ) 在IT 战略部署上，以商业目标、IT 风险及IT 投资成本为重心；

2)IT 价值交付是指在考虑时间价值的基础上，确保 IT投资与价值回报效率；

3)在风险管理方面，信息系统风险的控制在COSO 协议与 Basel 协议双重保障下相对 其他行业已发展成熟，对操作风险的控制更为专业；

4)信息系统绩效评价可借助IT 内部审计与 IT 平衡积分卡来完成；

5)IT 资源包括相关的人力资源及软硬件资源。

(3)大数据审计方法

大数据下的审计主要是指审计人员利用大数据资源，通过大数据方法，找到大数据与 被审计单位的联系，验证其经济活动的合法合规性。其具有以下特点：

1)所有数据都将成为被分析的对象。

第2章 大数据安全、隐私保护和审计技术 41

即使面对大量的样本，也不再使用抽样的方法，降低了审计风险，提高了审计结果的 准确性。例如，大数据环境下，中石油所有的收购信息都将以数据的形式保存在数据库中， 通过带有限定条件的数据库语言可以将所有收购加油站中的手续不全的商家查找出来，以 免其中有些成为漏网之鱼。

2)充分利用外部数据。

大数据为被审计单位获取和利用外部数据创造了条件，可以解决传统审计难以获取、 利用外部数据的固有弊端。利用外部数据可以从更多的视角发现可能出现的问题，提高审 计效率和准确性。大数据下，通过数据挖掘算法可找出作为企业外部数据的政府土地数据， 发现其中的违规行为并及时制止。

3)不需要函证。

大数据下环境，审计人员通过权限可以获取被审计单位的往来单位和往来银行的相关 数据。直接通过原始数据便可完成审计工作，不需要通过函证来证实被审计单位的相关经 济活动是否真实完整。这就减少了被审计单位与第三方单位舞弊的可能性，也节约了时间 和人力、物力。

4)不受时间地点限制。

大数据下的审计工作主要在互联网上进行，审计人员通过权限获取相应的数据，并对 其进行分析，得出结论。由于大部分证据都存储在网络数据库中，审计人员不需要在被审 计单位工作，也不用固定工作时间，只需一台计算机和网络环境便可进行工作，增加了审 计的灵活性，提高了效率。

**3.大数据治理审计技术**

大数据环境下，企业能够提供更多、更全面的数据，企业可以充分利用采集来的各方 面数据建立集中统一的被审计单位数据中心。在此基础上，借助不同于传统 SQL 关系数据 库的新的大数据分析技术，构建审计大数据分析平台和使用更智能的大数据分析技术，通 过分析“从数据入口到数据库平台”的更大范围的数据来源，对被审计单位的电子数据进 行系统、全面以及跨部门的综合分析，从而解决目前数据分析局限于查找单个问题的缺陷 获得更充分的审计证据，更大地发挥审计的威力。

传统的数据分析技术，如关联规则挖掘、分类、数据聚类、遗传算法、机器学习、自 然语言处理、神经网络、预测模型等，也可用于目前的大数据治理审计。但大数据环境下， 开展大数据治理审计需要更多的智能技术。目前，为了满足大数据环境下数据分析的需要， 一些专门用于处理大数据的关键技术也被研究出来，如 BigTable、云计算、分布式系统、 Hadoop 、HBase 、Map/Reduce、可视化技术等。因此，可借助以上技术进行审计大数据的 分析与结果展示。

当用户将数据存储在云服务器中时，就丧失了对数据的控制权。如果云服务提供商不 可信，其可能对数据进行篡改、丢弃，却对用户声称数据是完好的。为了防止这种危害，

42 第一篇 理 论 篇

云存储中的审计技术被提出。云存储审计指的是数据拥有者或者第三方机构对云中的数据 完整性进行审计。通过对数据进行审计，确保数据不会被云服务提供商篡改、丢弃，并且 在审计的过程中用户的隐私不会被泄露。

当前已有云存储中的审计模型有以下几种：

(1)数据持有 (Provable Data Possession,PDP) 模型

数据持有模型可以对服务器上的数据进行完整性验证。该模型先从服务器上随机采样 相应的数据块，并生成持有数据的概率证据。客户端维持一定数量的元数据，并利用元数 据来对证据进行验证。在该模型中，挑战应答协议传输的数据量非常少，因此所耗费的网 络带宽较小。

(2)可恢复证明 (Proof Of Retrievability,POR) 模型

可恢复证明模型主要利用纠错码技术和消息认证机制来保证远程数据文件的完整性和 可恢复性。在该模型中，原始文件首先被纠错码编码并产生对应标签，编码后的文件及标 签被存储在服务器上。当用户选择服务器上的某个文件块时，可以采用纠错码解码算法来 恢复原始文件。POR 模型面临的挑战在于需要构建一个高效和安全的系统来应对用户的请 求。有人改进了POR 模型，他们的模型构建基于BLS 短签名 (BLS short signature), 即基 于双线性对构造的数字签名方案，该模型拥有很短的查询和响应时间。

上述方案都只能适用于静态数据的审计，无法支持对动态数据的审计。有人改进了 PDP 模型，该模型基于对称密钥加密算法，并且支持数据的动态删除和修改。之后，Erway 等改进了PDP 模型，提出了DPDP 模型。该模型扩展了传统的PDP 模型以支持存储数据 的更新操作，该操作的时间复杂度为O(1)～O(log(n)) 。后来，又有人改进了前人的 POR 模 型，通过引入散列树来对文件块标签进行认证。同时，他们的方法也支持对数据的动态操 作，但是此方案无法对用户的隐私进行有效的保护。

第三方审计 (Third Party Auditor,TPA) 应该满足如下要求： 一是第三方审计能够高效 地完成对数据的审计，并且不给用户带来多余的负担；二是第三方审计不能为用户隐私带 来脆弱性。他们提出的方法基于公钥加密和同态认证，能够在保护用户隐私的情况下完成 公开审计。人们提出了一种用于对云中共享数据进行审计的隐私保护策略。他们在对数据 的审计过程中利用环形签名来对数据完整性进行验证。此策略能够很好地对用户的隐私进 行保护。其不足之处在于通信开销比较大。后来，人们还提出了一种名为Knox 的云中数据 的隐私保护策略。该策略利用群组签名来构造同态认证，使得第三方审计机构不需要从云 中获取整个数据即能完成对数据完整性的审计。

随着大数据时代的发展，可以预见到，未来存储在云中的数据会越来越多，这也为大 数据审计技术带来了巨大的挑战。在未来的研究中，以下几个方向也许值得研究者们关注： 一个是云中数据量越来越大、数据种类越来越丰富，如何提供更加高效、安全的审计服务 值得关注；另一个是随着人们在线上的交互越来越频繁，云中数据动态操作可能更加频繁， 如何应对如此频繁的数据动态操作也值得研究者们关注。

第2章 大数据安全、隐私保护和审计技术  43

**2.3.4** **大数据治理审计流程**

大数据治理审计流程和一般的审计流程差不多，大数据治理审计流程一般包括制定大 数据治理审计目标、确定大数据治理审计风险领域、制定大数据治理审计计划、搭建大数 据治理审计环境、出具审计结果和管理建议。

**1.制定大数据治理审计目标**

大数据治理审计目标是指人们在特定的社会生产环境中，期望通过审计实践活动达到 的大数据治理最终结果，或者说是指大数据治理审计活动的目的与要求。 一般来说，各类 大数据治理审计目标都必须满足其服务领域的特殊需要，无论是在公司还是学校，它们都 具有各自相对独立的审计目标。大数据治理审计目标的确定，除受审计对象的制约以外， 还取决于审计社会属性、审计基本职能和审计授权者或委托者对审计工作的要求。同时， 审计目标规定了审计的基本任务，决定了审计的基本过程和应办理的审计手续。

**2.确定大数据治理审计风险领域**

大数据治理审计风险是指大数据治理审计过程中可能会产生的风险。大数据治理审计 风险包括以下几个方面：

(1)数据采集风险

采集数据是审计分析的第一步，也是关系审计质量的关键一步。特别是大数据环境下 审计人员需要采集被审计单位的海量业务数据，在数据采集过程中审计人员主要面临两个 风险： 一是保证所采集数据的真实性、完整性，满足审计分析的需要；二是保证数据采集 过程中被审计单位的系统安全性。

(2)大数据存储和管理的风险

海量的大数据从被审计单位采集回来，在存储和管理方面审计机关和人员面临两方面 的风险： 一是数据存储风险，海量的大数据如何进行存储，保证数据的完整性，同时可以 供审计人员进行审计分析操作；二是数据管理的风险，被审计单位提供的数据包含大量的 个人基本信息、敏感信息，审计人员将面对如何对这些数据进行管理，从技术上和制度上 保证这些数据没有泄露到社会上的风险。

**3.制定大数据治理审计计划**

所谓大数据治理审计计划，是指为了完成各项大数据治理审计业务，达到预期的大数 据治理审计目标，在具体执行大数据治理审计程序之前编制的工作计划。大数据治理审计 计划通常可分为总体审计计划和具体审计计划两部分。

(1)总体审计计划

总体审计计划是对审计的预期范围和实施方式所做的规划，是负责大数据治理审计人 员从接受审计委托到出具审计整个过程基本工作内容的综合计划。

(2)具体审计计划

具体审计计划是依据总体审计计划制定的，对实施总体审计计划所需要的审计程序的

**44** 第 一 篇 理 论 篇

性质、时间和范围所做的详细规划与说明。 一般通过编制审计程序表的方式来体现。 一般来说，制定大数据治理审计计划有以下几个作用：

1)为大数据治理审计人员和审计工作明确方向。

现代社会的迅速发展，使审计面临和从事的工作越来越复杂。要切实解决审计面临的 问题和所从事的工作，就必须协调各个方面，调动各种资源，使所有审计人员齐心协力完 成工作。 一份良好的审计计划为审计人员制定了统一目标，使所有审计人员凝聚所有资源 朝着一个方向，共同努力来完成同一个任务，从而减少内耗，缩短时间，降低审计成本， 促进审计任务顺利实现。

2)减少未来不确定因素的负面影响。

社会在不断发展，审计也在不停地发展。无论是审计组织的外部环境因素还是审计组 织内部因素，在未来的发展中都具有一定的不确定性和变化性。审计计划是面向未来的， 能够通过周密细致的研究，系统运用各种科学方法手段来预测审计未来的发展变化，尽可 能将审计未来的变化和不确定因素转化为确定因素。通过审计计划，将各种不利因素转化 为有利因素，减少未来不确定因素的负面影响，促进审计工作的顺利进行，确保审计目标 的实现。

3)为大数据治理审计考核工作提供前提条件。

任何一项工作之后都要进行考核，为激励、组织和领导等工作提供前提条件。科学系 统的考核工作需要一个科学合理的基础。审计计划能够为审计考核工作提供一个合理前提， 也只有审计计划才能作为审计考核的基础，才能促使审计激励工作取得最大的效果。

4)为大数据治理审计控制工作提供标准。

任何一项工作在进行过程中都有可能因种种客观或主观原因而出现偏差，影响工作任 务的完成。因此，要随时对审计过程进行检查，加强审计项目过程的控制，促使审计目标 的顺利实现。要进行审计控制就需要一个控制标准，否则管理人员就无法实施控制。审计 计划是审计控制的基础，它为审计项目控制提供了控制标准。

5)提高审计效率和社会效益。

大数据治理审计计划能够通过各种科学技术方法来制定和选择科学详细的项目方案， 能够用科学决策代替经验判断，能够统筹安排审计资源，能够有针对性地根据经济社会发 展来科学安排审计项目等。这些都能够有力促进审计效率的提高，充分发挥“经济卫士” 和“经济谋士”的功能，从而促进社会效益的提高，促进经济社会的和谐发展。

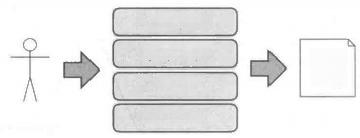
**4.搭建大数据治理审计环境**

搭建大数据治理审计环境是指根据大数据治理审计的目标和计划，设计出可以帮助进 行大数据治理审计的环境平台。 一般来说，进行大数据治理审计都要依靠审计程序，企业 产生的数据不可能都由人工去处理，所以进行大数据治理审计的开源项目应运而生。本书 中将要介绍的 Apache Ranger 就是一种可以用于大数据治理审计的软件。

通过搭建大数据治理审计环境，大数据治理审计部分工作可以交由审计程序自动进

第2章 大数据安全、隐私保护和审计技术 45

行。如图2-3所示，审计程序可以跟踪每个相关的用户和系统事件并创建审计日志。例如 Apache Ranger 是用于 Hadoop 的集中式安全管理

整合

解决方案，使管理员能够为HDFS 和其他 Hadoop

数据清洗 审计

可视化

平台组件创建和实施安全策略，并且为Hadoop 的 分析 日志

各个零部件提供细粒度的安全权限机制。它可以

对 Hadoop 生态系统上的组件如 Hive 、HBase 等进

图2-3 跟踪每个相关的用户和系统事件 行细粒度的数据访问控制，并解决授权和审计。在

并创建审计日志

Hadoop 生态系统上的组件的操作都会记录在日志

当中，这些日志中的数据可以用于定期安全审计。

具体来说， Apache Ranger可以提供这些审计能力。

1)用户行为日志： Apache Ranger维护与所有相关的用户和系统的事件和信息日志 文件。

2)安全审计日志： Apache Ranger 维护一个专门的安全审计日志，捕捉相关的安全调 查和审计行动，包括所有身份验证尝试、权限变更等。

**5.出具审计结果和管理建议**

接下来需要根据大数据治理审计计划的内容，执行大数据治理审计的计划内容。大数 据治理审计计划执行完了之后，最后一个流程是根据大数据治理审计的结果，出具审计的 结果和管理建议。审计结果一般以审计报告的形式呈现。审计报告是审计工作情况的全面 总结汇报，说明审计工作的结果。审计目标的实现结果是通过审计报告来反映的，审计报 告反映委托方的最终要求，也反映审计方完成任务的工作质量，同时也是对被审事项的评 价和结论的集中体现。在审计报告的最后一项， 一般还会给出一些大数据治理的管理建议。

第二篇 

**开源实现篇**

在了解大数据治理技术的相关基础概念和理论后，本篇将对大数据 安全与治理相关的5个开源组件 (Apache Falcon 、Apache Atlas 、Apache Ranger、Apache Sentry 和 Kerberos)进行介绍。

第3章 Apache Falcon部分为读者介绍基于Hadoop 环境的数据过程 及数据集管理系统 Apache Falcon, 同时给出具体的安装配置说明以及具 体的场景案例，并且提出相关的性能优化方案。

第4章 Apache Atlas 部分为读者介绍元数据管理框架 Apache Atlas, 通过技术概况、配置使用以及具体案例等方面的介绍，让读者能够深入地 了解使用Apache Atlas 进行元数据管理在大数据治理中起到的作用。

第5章 Apache Ranger 部分为读者介绍安全认证框架 Apache Ranger 在 Hadoop 环境下进行的安全数据访问控制，并通过具体的安装配置说明 和使用场景的案例，让读者了解 Apache Ranger的工作原理及运行机制。

第6章Apache Sentry 部分为读者介绍高度模块化的权限管理组件 Apache Sentry, 通过对Sentry 的组件架构进行深入的介绍，让读者了解 Sentry 的工作原理及其如何解决 Hadoop 文件系统缺乏对数据和 BI 应用细 粒度权限访问支持的问题。

第7章Kerberos 部分为读者介绍网络认证协议Kerberos, 通过对 Kerberos 工作原理的分析，并列举了多个 Hadoop 相关组件集成 Kerberos 的配置说明，让读者了解 Kerberos 在 Hadoop 安全模式中所扮演的角色以 及与各组件集成的方法。

本篇的目的是通过具体的实例，以Falcon 、Atlas 、Ranger 、Sentry 和Kerberos 这5个与大数据安全治理相关的开源组件为例，让读者在前一 篇概念理论的基础上，更加深入地了解大数据安全治理领域，同时为相关 从业者提供实践指导。



■■■

■■■

■■

■■

■ ■■ ■ ■■■

:

■■■

■ 画 ■

■

画|

■

■ ■■

国

■ ■

第 3 章

**大数据治理之** **Apache Falcon**

在前两章中，概述了大数据的安全和治理技术，包括大数据生命周期管理、安全保护 以及数据审计等。而之后几章将会为读者详细介绍大数据治理和安全的技术实现。在本章 中，将会说明大数据治理中最重要的部分，即大数据统一的生命周期管理以及血统追踪的 一个实现工具 Apache Falcon。自从Apache Falcon成为 Apache 基金会的孵化项目以来， 一 直备受关注，近年更是上升为 Apache 的顶级项目，逐渐成为大数据治理的主流工具。本章 将从Apache Falcon的概述讲起，逐步介绍 Falcon 的发展近况和技术优势，并通过对其架 构的分析，使读者对Falcon 有深刻了解，从而理解 Apache Falcon的使用和场景应用。

**3.1** **Apache Falcon 概述**

在数据处理过程中，由于种种原因，如数据源类型不一致、各个部分数据设计不统一， 或者不同数据源存储地址不同，造成各种问题，例如数据孤岛、数据不一致等。为了解决 这一系列问题，大数据治理也成为近些年的热门议题。

大数据治理包括数据生命周期管理、数据处理以及数据审计与追踪。其中数据生命周 期管理贯穿了对数据的采集、处理、备份保留以及清洗删除等过程。Hadoop 生态环境也 有相关组件来处理与之对应的问题，像Sqoop 实现对数据采集抽取，MapReduce 负责数据 处理备份等。但是它们就像机器的零件，虽然能处理数据治理的问题，然而却缺少有效的 手段去整合调度这些零件使其成为一台秩序运作的机器。为了解决这一问题，人们开发了 Oozie 这一Hadoop 调度工具。但是 Oozie 在设置工作流调度时，更多的是面向 Hadoop 底 层的设置，这使得用户在使用Oozie 时不但需要了解 Hadoop 的一些基础框架，同时生成的 工作流十分烦琐。为了简化 Hadoop 各个组件的调度工作，人们在Oozie 的基础上，开发了

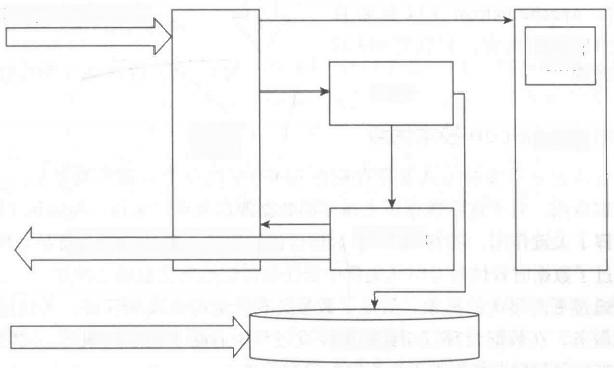
第3章 大数据治理之Apache Falcon  49

更加人性化的组件，这便是Apache Falcon。

**3.1.1 Apache Falcon 技术概况**

Apache Falcon作为 Hadoop 集群数据处理和数据生命周期管理系统框架，通过数据生 命周期管理及处理方案，解决 Hadoop 的数据复制、业务连续以及血统追踪等难题。Falcon 主要是对数据生命周期集中管理，促进了数据快速复制来实现业务连续性和灾难恢复，并 通过实体沿袭追踪和审计日志收集为审计和合规性提供基础，方便用户设定数据管理以及 处理方案，并将其提交到 Hadoop 集群调度执行。

Falcon 是一个建立在Hadoop 上的数据集及其处理流程的管理平台，如图3-1所示。 Falcon 本质上通过标准工作流引擎将用户的数据集及其流程配置转换成一系列重复的活动， 而本身不做任何烦琐的工作。所有功能以及工作流状态管理需求都委托给工作流调度器进 行调度。由于其本身并没有对工作流做额外的工作，Falcon 唯一要做的就是保持数据流程 实体之间的依赖和联系。这让开发人员在使用 Falcon 建立工作流时完全感觉不到 Oozie 调 度器以及其他基础组件的存在，使他们可以将工作重心放在数据及其处理本身，而不需要 进行任何多余操作。



Entity Client/Rest API,

Config Store

Oozie

Apache

Falcon

Process statusfnotification JMs

Entity status

Messaging

Hcatalog

Hadoop

图3-1 Falcon架构e

虽然 Falcon 将工作流交由调度器(默认调度器为Oozie, 由 于Oozie 自身的局限性， Falcon也在进行自带调度器的开发)负责调度，但是Falcon也与调度器之间保持通信(例 如JMS 消息),从而对执行路径下的每一个工作流都会产生消息追踪，确保掌握当前工作流 任务的进度以及具体情况。

θ <http://falcon.apache.org/FalconDocumentation.html>

**50** 第二篇 开源实现篇

**3.1.2** **Apache** **Falcon** **发展近况**

2013年，Srikanth 首次在印度的班加罗尔HUG 上展示和使用了Falcon 。随后不久，当 年3月，他向Apache 基金会提出将Falcon 作为 Apache 孵化项目的提案申请。2013年4月， Apache 基金会接受并通过Srikanth 的申请， Falcon 正式作为Apache 的一个孵化项目。在 2013年Hadoop 峰 会 上 ，Srikanth 和 Venkatesh 提交了一份有关展示Falcon 的讨论，并被 列入企业化数据架构方针的内容。在此之后， InMobi 公司同 Hortonworks 公司一起参与了 Apache 基金会的 Falcon 孵化工作。

经过几年的发展，2015年1月，Apache 基金会正式宣布 Apache Falcon 项目通过了孵 化阶段，上升为Apache 的一个顶级项目。在升级成为顶级项目之前，Apache Falcon已经 在多个行业中获得了广泛的应用，包括广告、医疗、移动应用等。目前InMobi 也是该平台

的最大用户之一

2016年8月8日，Apache Falcon公布

—3月 Falcon被提议作为Apache基金会的孵化项目 —4月 Falcon正式成为Apache孵化项目

—1月 Falcon升级成为Apache基金会顶级项目 —8月Falcon正式版本0.10发布

2013

2015

2016

了0.10版本，如图3-2所示。截至2016年

12月12日， Apache Falcon有17位 PMC

成员，24个社区提交者。在公布版本后

的 3 个 月 内 ，Apache Falcon社区有来自

世界各地的110位订阅者，并收到454封

E-mail邮件咨询。 图3-2 Apache Falcon 发展时间图

**3.1.3** **Apache** **Falcon** **技术优势**

Falcon 允许企业以多种方式处理存储在HDFS 中的单个大规模数据集，包括批处理、 交互和流数据应用。对于这些数据越来越多的数据源以及用户来说，Apache Falcon 的数据 治理功能发挥了关键作用。随着 Hadoop 数据价值的增长，数据清洗、商业化智能工具的准 备，以及在过了数据时效性时及时从集群中删除数据的重要性也随之增加。

Falcon 通过更高层次的抽象，简化了数据处理管道的开发和管理，通过提供开箱即用 的数据管理服务，在数据处理应用程序的开发过程中省略了复杂的编码，同时也简化了数 据移动、灾难恢复和数据复制等工作流的配置和编排。

Falcon 框架也可以作用于其他 Hadoop 组件，例如Pig 、HDFS 以 及Oozie 等。Falcon 通过提供一个定义-部署-管理数据管道的框架来实现这种简化的管理。作为开源的数据

生命周期管理项目，Apache Falcon 能够提供以下服务：

1)建立各种数据之间的关系和处理 Hadoop 环境下的元素。

2)数据集管理服务，例如数据保留、跨集群复制以及数据归档等。

3)方便进行新工作流/管道上传，支持后期数据处理和 Retry 政策。

4)集成了元数据库以及数据仓库，例如 Hive/HCatalog 集成。

5)为终端用户提供基于可用性数据集组(大部分同逻辑组的相关数据集一起使用)。

第3章 大数据治理之 Apache Falcon 51

6)支持在局部或全局聚合的本地处理来进行案例使用。

7)获取数据集和处理程序的血统。

总的来说，Apache Falcon实现的企业级数据治理需求主要体现在3个方面，如表3-1 所示。

**表3-1** **Apache Falcon 需求及特征表**

|  |  |
| --- | --- |
| **需** **求** | **特** **征** |
| 数据生命周期的统一管理 | 统一定义和管理数据采集、处理以及输出管道  确保灾难准备和业务连续性  用于数据复制和保留的开箱即用策略  端到端监控数据管道 |
| 合规性和审计 | 可视化数据管道血统  跟踪数据管道审计日志  使用事务元数据标记数据 |
| 数据库复制和归档 | 跨本地和基于云存储目标的数据复制  通过文档和示例的数据血统支持  HDFS中异构分层存储  集群中数据热/冷存储层的定义 |

**3.1.4** **Apache** **Falcon** **架构**

Apache Falcon当前架构如图3-1所示。用户通过Falcon 客户端或者 Rest API将实体 声明文件提交至Apache Falcon服务器，Falcon 根据声明信息生成工作流实体，并将其存 放在 Hadoop 环境的配置存储中。在执行工作流时， Apache Falcon 主要通过Oozie 进行任

务调度，并将实体执行状态存储至Hcatalog 中。在调度执行任务过程中，Oozie 会返回执 行过程中的状态信息以及执行命令消息，并发送至JMS 消息公告，将结果返回至Apache

Falcon。

**1.调度器**

目前Apache Falcon默认使用Oozie 作为调度引擎。虽然Oozie 效果很好，但是在有些 场景有其限制性。由于Falcon 依赖Oozie 进行调度和工作流执行，限制了数据集自然回归。 为了获得更好的调度能力，Apache Falcon项目开始进行自带原生调度器的研发工作。

如今Apache Falcon原生调度器基本可以取代Oozie 的功能，甚至做得更好。目前 Falcon 原生调度器可以完成以下功能：

1)提交并调度Falcon 定期运行处理过程(不需要数据依赖)。程序可以是PIG 脚本、 Oozie 工作流或Hive。

2)监控/查询/修改预定的处理过程——所有使用的实体 API 和实例API 都保持原有 状态。Falcon 以生命数据集的方式提供数据管理函数，它允许用户在HDFS 包含文件中提 交数据集位置作为基于时间的分区目录。

但是 Falcon 原生调度器仅支持对工作流以先进先出(FIFO) 的顺序执行，尚不支持后

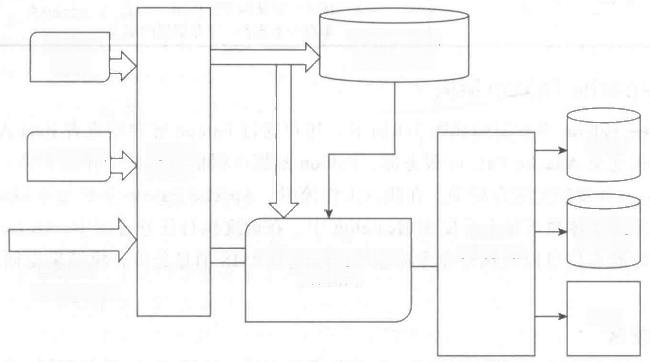
52 第二篇 开源实现篇

进先出(LIFO) 以及只取最新 (LAST\_ONLY) 顺序。

**2.工作流**

虽然工作流调度主要依靠调度器执行(例如默认Oozie 调 度 ) , 但 是 Apache Falcon 依 然 通过订阅每个工作流可能产生的消息，保留工作流的执行路径。当 Apache Falcon 在 Oozie 中 生成工作流时，此后它会使用JMS 消息传递等附加步骤对工作流进行检测。Apache Falcon 系统本身订阅了这些控制消息，并且在必要时可以执行重试以及处理最新输入数据等操作。

如图3-3所示，用户通过Cluster XML集群声明文件和Feed XML数据集声明文件向 Apache Falcon提交并声明了集群的配置信息以及数据集相关信息， Falcon 根据这两个文件 生成了 Cluster集群实体以及Feed 数据集实体，之后将这些实体信息存储至 Falcon 配置存 储中，并生成相关关系图。当需要执行保留或者备份等与数据集相关的操作时， Falcon 会 从配置存储中读取执行实体信息，并根据关系图生成相应工作流交由Oozie 调度器调度。 Oozie 会将调度结果输出至HDFS或者Hive的 Catalog Service中，并对每个 Action产生 JMS 消息公告。



Falcon config

store/Graph

Apache

Falcon

Retention/Replication

workflow

JMS

Instance

Management

Oozie Scheduler

Catalog Service

Cluster XML

Feed XML

HDFS

图3-3 数据集工作流图θ

如图3-4所示，用户通过提交Cluster XML集群声明文件、Feed XML数据集声明文 件以及Process XML数据处理声明文件至Apache Falcon 服务器，生成工作流调度相关实 体 。Apache 将这些实体信息存储至配置存储中并生成相关关系图。在执行数据处理工作流 时，Apache Falcon 根据集群实体、数据集实体以及过程实体生成相应的调度任务，并交由 Oozie 调度器调度。除了将输出数据放在HDFS 或 者 Hive Catalog Service 中 之 外 ，Oozie 也 会对每个存在的数据集生成工作流执行的JMS 消息。

综 上 所 述 ，Apache Falcon所 扮 演 的 角 色 就 像 是Oozie 的 一 个更高级的抽象层。它以调

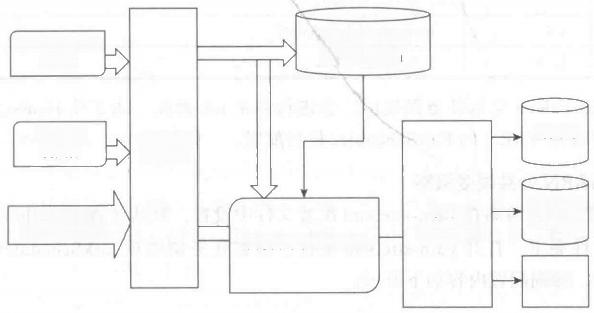
e <http://falcon.apache.org/FalconDocumentation.html>

第3章 大数据治理之 Apache Falcon 53

度器为核心，驱动调度Hive 、Sqoop 、Map Reduce 等 一 系列 Hadoop 组件任务，其本身并

没有直接负责数据治理中的实际数据处理。这 一特征也将在后面Apache Falcon 的安装部署

以及工作流编排调度中体现。



Falcon config

store/Graph

Process

XML

Instance

Management

JMS

Cluster/Feed XML

Oozie

Scheduler

Catalog Service

Apache Falcon

Process workflow

HDFS

图3-4 数据处理调度流图θ

**3.2** **Apache Falcon 的使用**

通 过 对 Apache Falcon架 构 的 介 绍 ， 我 们 知 道 Falcon 只是作为 一 个最高级的抽象层， 在编排工作流调度时，要根据实际任务引擎生成工作流作业任务，然后依赖于其核心调度 器执行任务调度工作。

本 节 将 学 习Apache Falcon和调度流作业组件的安装配置，以及工作流任务的编排创 建 。 虽 然 现 在Falcon 在进行自带原生调度器的开发工作，但目前稳定性尚不如Oozie 调 度，因此本节调度器的安装配置将着重介绍Oozie, 当然也会涉及自带调度器的启动以及与 Oozie 调度器的切换。而对于 Apache Falcon的安装配置，除了其本身的安装，也会涉及工 作流引擎如 Hive 和 Sqoop 的配置问题。在本节最后，将会介绍Falcon 工作流实体 cluster 、 feed 和 process 的编排创建。

在 介 绍 Oozie 安装配置之前，先说明推荐的安装环境配置，如表3 - 2所示。操作系统 选 择 了CentOS 6.5, 大数据环境使用 Hadoop 2.7.2 以 及Hive 1.4, 上述两个版本比较稳定， 得到广泛应用。

**表3-2** **Oozie** **环境说明表**

|  |  |  |
| --- | --- | --- |
|  | 版 本 号 | **说** **明** |
| 操作系统 | CentOS 6.5 | **轻量级操作系统，可以用其他对应版本的Linux系统代替** |

θ <http://falcon.apache.org/FalconDocumentation.html>

54 第二篇 开源实现篇

(续)

|  |  |  |
| --- | --- | --- |
|  | **版** **本** **号** | **说** **明** |
| Hadoop | 2.7.2 | 由于Apache Falcon仅支持Hadoop 2以上版本，而Hadoop 2.7.2稳定性最 好，得到广泛应用 |
| Hive | 1.4 | Hive 1.2以上版本即可 |
| MySQL | 任意 |  |

在 Apache Falcon 运行任务调度时，会进行多个job 调度，为了使 Hadoop更好地执行 任务调度，需要对YARN的 FairScheduler 进行配置。

**1.设置YARN** **公共调度策略**

YARN 相关调度策略在yarn-site.xml 配置文件中设置，默认放置在 Hadoop 安装目录的 etc/hadoop 文件夹下。打开 yarn-site.xml 文件，设置任务调度项 taskScheduler 为公平调度 FairScheduler, 详细配置内容如下所示：

<configuration>

#省略其他配置项

<!---开启YARN公平调度，如果未设置则关闭--->

<property>

<name>mapred.jobtracker.taskScheduler</name>

<value>org.apache.hadoop.mapred.FairScheduler</value>

</property>

<!---如果开启公平调度，在此配置自定义调度策略XML文件，该文件主要用于描述各个队列的属性，比如资

源量、权重等--->

<property>

<name>mapred.fairscheduler.allocation.file</name>

<value><!--- 此处为配置公平调度策略文件的具体文件路径---></value> </property>

</configuration>

配置项添加完毕后，重启 Hadoop 使 YARN 配置生效，或者启动超级用户权限更新 Hadoop 配置，命令如下：

s hdfs dfsadmin -refreshSuperUserGroupsConfiguration

在上述操作完成后，读者也可以根据自己的需求修改公平调度的具体配置，例如任务 执行队列、内存分配等设置。可以设置的配置项如下。

1)yarn.scheduler.fair.user-as-default-queue: 当应用程序未指定队列名时，是否指定用 户名作为应用程序所在的队列名。如果设置为false 或者未设置，所有未知队列的应用程序 将被提交到 default 队列中，默认值为 true。

2)yarn.scheduler.fair.preemption: 是否启用抢占机制，默认值是false。

3)yarn.scheduler.fair.sizebasedweight: 在一个队列内部分配资源时，默认情况下，采 用公平轮询的方法将资源分配给各个应用程序，而该参数则提供了另外一种资源分配方

第3章 大数据治理之 Apache Falcon 55

式——按照应用程序资源需求数目分配资源，即需求资源数量越多，分配的资源越多。默 认情况下，该参数值为 false。

4)yarn.scheduler.assignmultiple: 是否启动批量分配功能。当一个节点出现大量资源时， 可以一次分配完成，也可以多次分配完成。默认情况下，该参数值为 false。

5)yarn.scheduler.fairmax.assign: 如果开启批量分配功能，可指定一次分配的 container 数目。默认情况下，该参数值为-1,表示不限制。

6)yarn.scheduler.fair.locality.threshold.node: 当应用程序请求某个节点上的资源时，它 可以接受的可跳过的最大资源调度机会。当按照分配策略可将一个节点上的资源分配给某 个应用程序时，如果该节点不是应用程序期望的节点，可选择跳过该分配机会，暂时将资 源分配给其他应用程序，直到出现满足该应用程序需求的节点资源。通常而言， 一次心跳 代表一次调度机会，而该参数则表示跳过调度机会占节点总数的比例。默认情况下，该值 为-1.0,表示不跳过任何调度机会。

7)yarn.scheduler.fair.locality.threshold.rack: 当应用程序请求某个机架上的资源时，它 可以接受的可跳过的最大资源调度机会。

8)yarn.scheduler.increment-allocation-mb: 内存规整化单位，默认是1024,这意味 着，如果一个 container 请求资源是1.5GB, 则将被调度器规整化为ceiling(1.5GB/1GB)\*

1GB=2GB。

9)yarn.scheduler.increment-allocation-vcores: 虚拟 CPU 规整化单位，默认值是1,含

义与内存规整化单位类似。

**2.添加自定义调度策略**

FairScheduler 允许用户将队列信息专门放到 一个配置文件(默认文件名是fair- scheduler.xml) 中，对于每个队列，用户可以配置以下几个选项。

1)minResources: 最少资源保证量，设置格式为 “Xmb,Y vcores”。当一个队列的最 少资源保证量未满足时，它将优先于其他同级队列获得资源。对于不同的调度策略，最少 资源保证量的含义不同。对于fair 策略，只考虑内存资源，即如果一个队列使用的内存资 源超过了它的最少资源量，则认为它已得到了满足；对于drf 策略，则考虑主资源使用的资 源量，即如果一个队列的主资源量超过它的最少资源量，则认为该队列资源已满足。

2)maxResources: 最多可以使用的资源量，FairScheduler 会保证每个队列使用的资源 量不会超过该队列的最多可使用资源量。

3)maxRunningApps: 最多同时运行的应用程序数目。通过限制该数目，可防止超量 Map Task同时运行时产生的中间输出结果撑爆磁盘。

4)minSharePreemptionTimeout: 最小共享量抢占时间。如果一个资源池在该时间内使 用的资源量一直低于最小资源量，则开始抢占资源。

5)schedulingMode/schedulingPolicy: 队列采用的调度模式，可以是 fifo 、fair 或者 drf。 6)aclSubmitApps: 可向队列中提交应用程序的Linux 用户或用户组列表，默认情况下

56 第二篇 开源实现篇

为“\*”,表示任何用户均可以向该队列提交应用程序。需要注意的是，该属性具有继承性， 即子队列的列表会继承父队列的列表。配置该属性时，用户之间或用户组之间用“,”分隔， 用户和用户组之间用空格分隔，比如 “userl,user2 groupl,group2”。

7)aclAdministerApps: 该队列的管理员列表。 一个队列的管理员可管理该队列中的资 源和应用程序，比如可杀死任意应用程序。管理员也可为单个用户添加 maxRunningJobs 属 性限制其最多同时运行的应用程序数目。此外，管理员也可通过以下参数设置以上属性的 默认值。

口 userMaxJobsDefault: 用户的 maxRunningJobs 属性的默认值。

口 defaultMinSharePreemptionTimeout: 队列配置 minSharePreemptionTimeout 属性的

默认值。

口 defaultPoolSchedulingMode: 队列的 schedulingMode 属性的默认值。

口 fairSharePreemptionTimeout: 公平共享量抢占时间。如果一个资源池在该时间内使 用资源量一直低于公平共享量的一半，则开始抢占资源。

下面为 fair-scheduler.xml 的一个实际例子，YARN 调度队列有 hive 、hadoop, 用户管 理员为 hadoop, 读者可以根据自己的实际情况进行配置。

<?xml version="1.0"?>

"hive">

<minMaps>90</minMaps>

<minReduces>20</minReduces>

<maxRunningJobs>20</maxRunningJobs>

imeout>30</minSharePreemptionTimeout>

</pool>

<pool name="hadoop">

<minMaps>9</minMaps>

<minReduces>2</minReduces>

<maxRunningJobs>20</maxRunningJobs>

<weight>1.0</weight>

<minSharePreemptionTimeout>30</minSharePreemptionTimeout>

</pool>

<user name="hadoop">

<maxRunningJobs>6</maxRunningJobs>

</user>

<poolMaxJobsDefault>10</poolMaxJobsDefault>

<userMaxJobsDefault>8</userMaxJobsDefault>

<defaultMinSharePreemptionTimeout>600</defaultMinSharePreemptionTimeout> <fairSharePreemptionTimeout>600</fairSharePreemptionTimeout>

**3.2.1** **Oozie** **的安装与配置**

Apache Falcon通过解析用户提交的实体配置XML 文件，生成 Oozie 调度流任务，并 启动调度流引擎执行工作流。Oozie 调度器在整个工作流调度中起着举足轻重的作用，因此

第3章 大数据治理之 Apache Falcon 57

在安装 Apache Falcon 之前，先介绍Oozie 的安装与配置。

**1.Oozie 编译**

由于Oozie 调度器调度任务时需要其他 Hadoop组件的jar 包，所以最好自己针对不同 版本的 Hadoop组件设置编译参数生成 bin文件。Oozie 源码可以从以下北京理工大学镜像 网站 (mirror.bit.edu.cn) 上获取资源：

$>wget <http://mirror.bit.edu.cn/apache/oozie/4.2.0/oozie-4.2.0.tar.gz>

通过上述命令下载得到 tar 压缩包，解压缩得到源码文件 oozie-4.2.0。

$>tar zxvf oozie-4.2.0.tar.gz

进入源码文件路径，通过Maven编译Oozie(Maven 安装配置的具体步骤参见7.3.2节), 得到安装需要的二进制文件 (bin) 以及相关jar包。编译脚本为解压缩文件 oozie-4.2.0/bin/ mkdistro.sh, 调用该脚本，读者可以根据实际情况配置 Hadoop、Hive 以及Sqoop 版本参数。

/ -DskipTests -Phadoop-2 -Dhadoop.auth.version=2.6.0 -Ddistcp.

version=2.6.0 -Dhive.version=1.2.1 -Dsqoop.version=1.4.6

编译完成后将在当前路径的 distro/target/ 目录下生成 Oozie 二进制文件压缩包 oozie- 4.2.0-distro.tar.gz, 解压缩得到 Oozie 二进制文件oozie-4.2.0。

$>tar -zxf distro/target/oozie-4.2.0-distro.tar.gz

这个 oozie-4.2.0文件就是 Oozie 程序文件，读者可以将其放在需要设置的$OOZIE HOME路径下，在此先将其移动至/var/local/hadoop目录下。

$>mv oozie-4.2.0 /var/local/hadoop/

以上操作完成后，读者可以在/var/local/hadoop 目录下看到 Oozie 工程文件。之后还需 要配置 Oozie 环境变量以及相关配置项。

**2.Oozie 环境变量**

为了方便之后管理以及调用，需要在安装Oozie 的节点机器上设置Oozie 环境变量， 在/etc/profile 文件中添加如下内容：

IE\_HOME=/var/local/hadoop/oozie-4.2.0

export PATH=SPATH:SHADOOP\_HOME/bin:SOOZIE\_HOME/bin

刷新环境变量：

$source /etc/profile

**3.Oozie** **配置**

启动Oozie 时需要设置 Hadoop 镜像用户，该信息配置文件为core-site.xml, 默认路径 为 $HADOOP\_HOME/etc/hadoop 文件夹。修改 Hadoop 的core-site.xml 文件，添加如下配置：

**58**  **第二篇** **开源实现篇**

<property>

<name>hadoop.proxyuser.hdfs.hosts</name>

<value>\*</value>

</property>

oop.proxyuser.hdfs.groups</name>

<value>\*</value>

</property>

#其中hdfs 是用户之后运行Oozie 的用户名

**不重启** **Hadoop 集群，而使配置生效。**

$hdfs dfsadmin -refreshSuperUserGroupsConfiguration

s yarn rmadmin -refreshSuperUserGroupsConfiguration

**修改** **Oozie 配置文件，添加如下内容：**

<property>

<name>oozie.service.JPAService.create.db.schema</name>

<value>true</value>

</property>

zie.service.JPAService.jdbc.driver</name>

<value>com.mysql.jdbc.Driver</value>

</property>

<property>

<name>oozie.service.JPAService.jdbc.url</name>

<value>jdbc:mysql://linc-1:3306/oozie?createDatabaseIfNotExist=true</value>

</property>

<property>

<name>oozie.service.JPAService.jdbc.username</name>

<value>oozie</value>

</property>

<property>

<name>oozie.service.JPAService.jdbc.password</name>

<value>oozie</value>

</property>

<property>

<name>oozie.service.HadoopAccessorService.hadoop.configurations</name> <value>\*=/var/local/hadoop/hadoop-2.6.0/etc/hadoop</value>

</property>

<property>

<name>oozie.service.ProxyUserService.proxyuser.hdfs.hosts</name>

<value>\*</value>

</property>

<property>

<name>oozie.service.ProxyUserService.proxyuser.hdfs.groups</name> <value>\*</value>

</property>

其中 oozie.service.HadoopAccessorService.hadoop.configurations 项为 Hadoop 的环境配 置文件所在目录，即为$HADOOP\_HOME下的 etc/hadoop目录。在本例中 linc-1 是运行Oozie

第3章 大数据治理之Apache Falcon 59

的节点机器， hadoop是运行Oozie的用户名，需要与前面Hadoop的配置文件 core-site.xml 中的用户名配置内容一致。

**4.MySQL 配置**

Oozie 需要使用元数据库存储 Oozie 信息以及任务流调度状态信息，本书中选择 MySQL 作为其元数据。为了方便管理，需要在 MySQL 中添加 oozie 用户，并添加之后Oozie运行 所需的数据表。

使用管理员用户进入MySQL 控制台，指令如下。其中-u 后的值为登录用户名， -p 后 的值为登录密码：

s mysql -uroot -padmin

创建名称为 oozie 的数据库：

$mysql>create database oozie;

设置oozie 数据库的访问权限，创建用户名为oozie 、密码为oozie 的用户：

$mysql>grant all privileges on oozie.\*to 'oozie'@'localhost'identified by 'oozie';

设置oozie 数据库的访问权限：

$mysql>grant all privileges on oozie.\*to 'oozie'@'s'identified by 'oozie';

设置oozie 用户的认证权限：

$mysql>update mysql.user set host='g'where user='root'and host='localhost';

s mysql>insert into mysql.user(host,user,password)values('linc-1','oozie',

PASSWORD('oozie'));

更新配置并退出 MySQL控制台：

$mysql>FLUSH PRIVILEGES;

$mysql>quit

重启MySQL 使配置生效：

$>sudo service mysqld restart

**5.部署Oozie**

在编译安装完 Oozie 4.2.0并配置好 MySQL元数据库后，如果要运行Oozie, 还需要导 入Oozie 运行的jar 包。在此读者需要在$OOZIE\_HOME目录下创建 libext 文件夹来存放 扩展jar包。

#/var/local/hadoop/oozie-4.2.0 为之前存放Oozie 的SOOZIE\_HOME

$>cd /var/local/hadoop/oozie-4.2.0

# 创 建libext 文件夹

$>mkdir libext

#将编译得到相应组件版本的jar 包放入libext 中

$>tar zxvf oozie-sharelib-4.2.0.tar.gz

**60** 第二篇 开源实现篇



凄

为防止 Hcatalog版本与编译的 Oozie不同，可以手动将相应 Hcatalog 以及Hive 版本的 jar 包导入libext 中，其他 Hadoop组件导入过程与此类似。具体操作如下：

# 将Hadoop 的jar 包导入SOOZIE\_HOME/libext

$>cp SHADOOP\_HOME/share/hadoop/\*/\*.jar libext/

$>cp SHADOOP\_HOME/share/hadoop/\*/lib/\*.jar libext/

$>cp SHive\_HOME/lib/\*.jarlibext/

$>cp share/lib/hcatalog/\*.jar libext/

之后，读者需要将 Hadoop运行的相关jar 包导入libext 中。需要注意的是， Oozie Web UI主要搭建在Tomcat上，因此在导入Hadoop运行jar 包之前，首先要去掉Hadoop 与 Tomcat 相冲突的jar 包。

# 把Hadoop 与Tomcat 冲突的jar 包去摔

$>cd libext

$>mv servlet-api-2.5.jar servlet-api-2.5.jar.bak

$>mv jsp-api-2.1.jar jsp-api-2.1.jar.bak

S>mv jasper-compiler-5.5.23.jar jasper-compiler-5.5.23.jar.bak

$>mv jasper-runtime-5.5.23.jar jasper-runtime-5.5.23.jar.bak

# 将Hadoop 的jar 包导入Oozie 的libext

$>cp SHADOOP\_HOME/share/hadoop/\*/\*.jar libext/

$>cp SHADOOP\_HOME/share/hadoop/\*/lib/\*.jar libext/

$>cp $Hive\_HoME/lib/\*.jar libext/

$>cp share/lib/hcatalog/\*.jar libext/

Oozie Web UI页面js需要导入ext-2.2.zip包，而与MySQL连接也需要导入 MySQL 的JDBCjar 包，二者都需要放在libext 文件夹中， MySQL JDBC jar 包需要与当前 安装的 MySQL 兼容。在导入时需要注意版本问题。

**6.启动并测试** **Oozie**

顺利完成上述步骤后，读者就可以开启Oozie 了。Oozie 在执行工作流调度时，需要获 得相应工作流引擎在 HDFS 的运行jar包。Oozie在运行工作流时，会先从工作流配置文件 workflow.xml的指定目录中寻找相应jar 包，如果未找到相应jar 包，则从 HDFS上一个统 一的 Oozie jar包库中导入，默认路径为user/oozie/sharelib。具体部署步骤如下：

#打包Oozie war包

$>cd /var/local/hadoop/oozie-4.2,0

$>bin/oozie-setup.sh prepare-war

#初始化数据库

S>bin/ooziedb.sh create -sqlfile oozie.sql -run

#修改服务器端conf/server.xml 文件，注释掉下面的记录

$>vim oozie-server/conf/server.xml

<!--<Listener className="org.apache.catalina.mbeans.ServerLifecycleListener"/>-->

# 将Oozie share库中的jar 上传至HDFS

$>bin/oozie-setup.sh sharelib create -fs hdfs://linc-1:9000

在部署结束后，可以在控制台界面输入以下操作指令启动并运行 Oozie:

第3章 大数据治理之 Apache Falcon 61

注 意

#启动Oozie服务

$>cd /var/local/hadoop/oozie-4.2.0

$>bin/oozied.sh start

#检验服务是否正常启动，如果显示System model:Normal则启动成功，反之失败

$>oozie admin -oozie http://linc-1:11000/oozie -status

读者可以在浏览器输入http://linc-1:11000/oozie/进入Oozie Web端控制台查看 Oozie 运行状态，其中 linc-1 为安装并运行 Oozie 的节点机器的 IP地址。

Oozie 运行时需要确保 Hadoop historyserver 启动。通过jps 指令可以显示已启动的 Java 服务，如果列表中没有historyserver服务，则输入下列指令启动 historyserver:

SHADOOP\_HOME/sbin/mr-jobhistory-daemon.sh start historyserver

**3.2.2** **Falcon** **的安装与配置**

Apache Falcon 作为 Oozie 之上的更高级抽象层，并不与具体的 Hadoop 集群及其相关 组件直接接触，所以安装配置并不复杂。只要主调度器 Oozie 配置正常， Falcon 安装十分 简便，本节将介绍 Apache Falcon 嵌入模式的安装配置。

**1.Falcon 编译**

由于Apache Falcon目前并没有编译完整的二进制文件(bin), 所以需要自己下载有关 源码资源并通过Maven编译。由于许多jar 包资源需要从国外网站上下载，所以整体的编 译过程比较慢，需要耐心等待。

首先执行下面指令下载 Apache Falcon 的源码，资源网站依然是北理镜像网。

$>wget <http://mirror.bit.edu.cn/apache/falcon/0.9/apache-falcon-0.9-sources.tar.gz> 之后通过 Maven编译源代码，获得 Apache Falcon的二进制文件。

#解压缩得到代码文件falcon-sources-0.9 并进入

$>tar -zxvf apache-falcon-0.9-sources.tar.gz

$>cd falcon-sources-0.9/

# 设 置Maven 参数并打包编译Falcon 源码

$>export MAVEN\_OPTS="-Xmx1024m -XX:MaxPermSize=256m -noverify"&&mvn clean install -Dhadoop.version=2.6.0 -Doozie.version=4.2.0 -DskipTests

打包编译Falcon源码，如果在编译过程中出现 npm error报错，可能是当前机器没有安 装npm, 输入下列指令安装 npm 并使用国内镜像：

$>sudo yum -y install npm

$>npm --registry <https://registry.npm.taobao.org> info underscore $>mvn clean assembly:assembly -DskipTests -DskipCheck=true

编译完成后在target 文件夹下存在 apache-falcon-0.9-bin.tar.gz 和apache-falcon-0.9-bin. zip压缩包。

62 **第二篇** **开源实现篇**

#解压缩Falcon 二进制压缩包

$>tar -zxvf target/apache-falcon-0.9-bin.tar.gz

#将解压缩得到的falcon-0.9 文件移到/var/local/hadoop 目录下

S>mv falcon-0.9 /var/local/hadoop/

**2.配置** **Falcon** **环境变量**

为了方便之后管理以及调用，需要在安装 Falcon的节点机器上设置Falcon 环境变量， 通过下面命令打开环境配置文件：

$>sudo vim /etc/profile

#在文件末尾，添加如下内容

export FALCON\_HOME=/var/local/hadoop/falcon-0.9

export PATH=SPATH:SHADOOP\_HOME/bin:SFALCON\_HOME/bin

配置完成后刷新环境变量：

$>source /etc/profile

**3.同步** **Oozie** **配置项**

为了使 Apache Falcon正常通过Oozie 调度器生成工作流调度，要在Oozie 的oozie- site.xml配置文件中添加与Falcon 相关的配置内容，相关配置项信息在{falcon-server-dir}/ oozie/libext/oozie-site.xml文件中，{falcon-server-dir}为之前放置 Falcon 工程的bin 文件夹， 在本例中为/var/local/hadoop/falcon-0.9。需要将{falcon-server-dir}/oozie/conf/oozie-site. xml 中的配置信息添加至$OOZIE\_HOME/conf/oozie-site.xml。

同时，读者也要将相应jar 包 导 入 Oozie libext文 件 夹 下 ，Falcon jar 包 目 录 为 $FALCON\_

HOME/oozie/libext。

# 将Falcon 在Oozie 目录下的扩展jar 包拷贝至SOOZIE\_HOME/libext 文件夹下

$>cd /var/local/hadoop/oozie-4.2.0

$>cp /var/local/hadoop/falcon-0.9/oozie/libext/\*.jar libext/

为了使上述配置生效，需要重启Oozie:

#重新部署并启动Oozie

$>cd SOOzIE\_HOME

$>bin/oozie-stop.sh

$>bin/oozie-setup.sh prepare-war

$>bin/oozie-start.sh

**4.Falcon** **Client** **配置**

对于Falcon Client端，需要指定对于Falcon Server的相关信息， Client 端的配置文件 为 $FALCON\_HOME/conf/client.properties。

$>cd /var/local/hadoop/falcon-0.9

$>vim conf/client.properties

# 修 改client.properties 文件中falcon.url 的值，大括号()以及其中内容为相关配置信息，需要根据

实际情况替换

第3章大数据治理之Apache Falcon 63

falcon.url=https://(linc-1}:{port}/

falcon.url指定了Falcon Server的IP 地址，在本例中为linc-1 的 IP 地 址 ；port 为 Falcon 启动时配置的端口号，默认为15443。

**5.修改** **Falcon** **配置文件**

Falcon通过JVM 服务器进行消息通信，需要根据安装 Falcon 机器的IP 配置消息服务 器地址。

#编译启动配置项文件

$>cd /var/local/hadoop/falcon-0.9

$>vim conf/startup.properties

# 将 \* .broker.url 的值改动如下

\*.broker.url=tcp/linc-1:61616

|  |  |
| --- | --- |
| 清 | \*.broker.url 为 Falcon 自带 activemq 消息发送地址，即 Falcon 运行所在的节点机器， 在本例中为linc-1 的 IP 地址。 |
| 启动 Falcon服务器后，可以使用基于Web的控制台查看 Falcon 实体的状态。用户可 以在相应的端口打开浏览器以使用WebUI, 连接URL 为前面配置的 https:linc-1:15443。  Falcon控制台将RESTAPI 调用作为用户 “falcon-dashboard” 。 如果此用户不存在于你 的 Falcon 和Oozie 服务器上，请创建该用户。  # 显 示falcon-dashboard:falcon-dashboard users则创建成功  $>useradd -U -m falcon-dashboard -G users  $>groups falcon-dashboard  **6.启动** **Falcon**  完成上述步骤后，就可以启动 Falcon Server了 。Falcon Server 的启动脚本为$FALCON\_ HOME/bin/falcon-start, 关闭脚本为$FALCON\_HOME/bin/falcon-stop。  #启动Falcon'Server  cd /var/local/hadoop/falcon-0.9  bin/falcon-start  #显示Java 进程，如果列表中有Falcon Server则启动成功  jps | |
|  | 用户可以在Client 端通过浏览器输入 https://inc-1:15443 进入 Falcon Web 端控制台。 注意，Falcon Server使 用https 协议，如果输入地址为 http 则显示出错。 |

**3.2.3** **实** **体XML 的创建与声明**

从3.1节可以知道， Apache Falcon对调度工作流的声明定义主要通过实体 (entity) XML文件。实体主要分成三类： 一类是说明集群配置的 cluster 实体，还有一类是说明数据

64 **第二篇** **开源实现篇**

集 (feed) 配置的feed 实体，最后一类是说明调度进程的 process 实体。对于不同数据集类 型的工作流实体，具体的配置项也不尽相同，其中数据集主要分为HDFS 文件数据集以及 Hive 数据库数据集。

**1.cluster** **实体声明项**

该声明项说明集群的相关配置项，主要包括对HDFS 、YARN 以及Oozie 的相应配置 说明。

1)cluster XSD规范可用： 一个集群包含不同的接口，这些接口有readonly 、write、 workflow 和 messaging 。cluster实体会在之后被 feed 实体以及 process 实体所依赖。在将 cluster.xml 提交至Falcon后，在 Falcon中有唯一 cluster name 与其对应。

cluster.xml文件的标签定义如下：

<cluster colo="gs"description=""name="corp"xmlns="uri:falcon:cluster:0.1" xmlns:xsi="<http://www.w3.org/2001/XMLSchema-instance>">

素 在上面的 cluster.xml标签中， colo指明此集群属于哪个 colo,name 是指集群的名称， 并且该名称必须是唯一的。

2)接口(interface): 一个 cluster需要定义以下接口。

指定 Hadoop 只读协议的 HFTP 协议终端，示例如下：

<interface type="readonly"endpoint="hftp://localhost:50010"version="0.20.2"/>

指定写入HDFS中的端口，它的终端是fs.defaultFS 的 IP地址。Falcon 用该接口将系 统数据写入HDFS,feed 用相同的 write 接口将数据写入HFDS, 示例如下：

<interface type="write"endpoint="hdfs://localhost:8020"version="0.20.2"/>

执行JobTracker程序需要指定执行接口。其endpoint 是mapreduce.jobtracker.address 的IP 地址。在 Hadoop 2.x中，此接口值为YARN 指定的执行接口，示例如下：

<interface type="execute"endpoint="localhost:8021"version="0.20.2"/>

workflow接口指定工作流引擎的接口，例子中的endpoint是 OOZIE\_URL的 value。 Falcon 用该接口来安排在工作流引擎上所引用的集群进程，示例如下：

<interface type="workflow"endpoint="http://localhost:11000/oozie/"version="4.0"/>

register接口指定元数据目录界面，例如： Hive Metastore(或 Hcatalog) 。Falcon用该 接口注册一个给定数据库和表的分区。此外，用此信息来调度工作流引擎中基于分区的数 据的可用性事件，示例如下：

<interface type="register"endpoint="thrift://localhost:9083"version="0.11.0"/>

虽然Hive元数据支持RPC和HTTP, 但是feed 是RPC在 thrift上的实现。对于Hive HA 模式，确保 URI用逗号分隔，仅仅需要在开头添加 “thrift://” 协议。下面是一个 Hive 模式

**第3章** **大数据治理之Apache Falcon** 65

的例子：

<interface type="register"endpoint="<thrift://c6402.ambari.apache.org:9083>,c6403. ambari.apache.org:9083"version="0.11.0"/>

messaging接口指定发送 Falcon可用消息的接口，该接口的endpoint联合了TCP 和 URL 地址。

<interface type="messaging"endpoint="tcp:/Xlocalhost:61616?daemon=true"version= "5.4.6"/>

3)路径位置 (location):cluster 还需要定义一个位置列表，如下：

<location name="staging"path="/projects/falcon/staging"/>

<location name="working"path="/projects/falcon/working"/>

<!--optional-->

location 有 name和 path两个值， name 是location的类型。被允许的name 值是 staging、 temp和 working。path是每个location 的 HDFS路径。Falcon 在 HDFS 中使用location 进 行实体间接处理，因此， Falcon 在这些 location 中具有读/写执行权限。这些location必 须在将一个集群实体提交到feed 之前提交。staging 应该具有777权限，是一个强制性的 location。父目录必须有执行权限去让多个用户能够在该location 执行写操作。working 必须 具有775权限，是一个可选的location。

如果working没有被指定，那么 Falcon会以755权限在 staging location 中创建一个子 目录。working的父目录必须有执行权限才能让多个用户在该 location 进行读操作。

4)权限 (ACL):cluster 的 ACL(Access Control List) 用于实施权限和提供一种为不 同的文件所有者和文件所属用户组设置不同权限的方式。

<ACL owner="test-user"group="test-group"permission="\*"/>

ACL 表示这个集群的访问控制列表， owner 是实体所有者， group 是有权限进行read 操作，permission 表示许可权限。

5)消费属性(custom property):cluster 有一个属性列表，即一个传播工作流引擎的键 值对。

<property name="brokerImplClass"value="org.apache.activemq.ActiveMQConnectionFactory"/> 理想的JMS实现类名称的消息引擎 (brokerlmpIClass) 应该在这里进行定义。

**2.feed** **实体声明项**

feed XML文件说明 feed与 cluster之间的依赖关系，介绍XML配置项中 feed frequency、 cluster 、retention policy 以及data set location 的作用。

1)feed XSD规范可用： 一个 feed定义了数据集的各种属性，如 feed location 、frequency、 late-arrival handing 和 retention policy 。feed 同时还可以在集群上提前指定，当一个feed 在 它所保留区提前指定，那么复制进程会在给定的集群上触发。

**66** **第二篇** **开源实现篇**

<feed description="clicks log"name="clicks"xmlns="uri:falcon:feed:0.1" xmlns:xsi="<http://www.w3.org/2001/XMLSchema-instance>">

feed 应该有且只有一个特定的 name, 该 name 为之后输入/输出process 所引用。

2)存储 (storage):Falcon 引进一个新的抽象来封装给定的feed, 同时也可以表示为一 个文件系统上的路径、文件系统存储，或 Hive 、Catalog Storage 中的目录。

<xs:choice minOccurs="1"max0ccurs="1">

<xs:element type="locations"name="locations"/>

<xs:element type="catalog-table"name="table"/>

</xs:choice>

feed 应该包含两个 storage选项之一，即 File System的 location或者 catalog 中的 table。

3)文件系统存储 (File System Storage):feed 通过集群的 name 来进行引用，在提交 一个feed之前，所有的引用集群都应该被提交给 Falcon。type 指定引用的集群是否应该 被视为一个feed 源或者目标。 一个 feed可以有多个源和目标集群。如果集群的type 没有 被指定，那么不考虑集群的响应。 一个有效的 feed指定该feed 在集群上有效的持续时间。 retention指定这个集群上feed retention 的时间长度，以及retention结束之后在 feed上的活 动。retention limit由表达频率(次数)指定。例如：如果 feed 应该保留至少6小时，那么 retention limit="hours(6)" 。PartitionExp 包含分区标签。分区标签的数目必须与 feed 模式中 指定的分区数目相等。分区标签可以是通配符(\*)、静态的字符串或者表达式。至少一个 字符串必须是表达式。SLA 在这个集群上为 feed 指定sla。这是一个可选的参数，从 global sla标签上看，sla 可以相同也可以不同。这个标签提供用户在不同的集群上灵活地拥有不同 的sla, 示例如下：

<clusters>

<cluster name="test-cluster">

<validity start="2012-07-20T03:00Z"end="2099-07-16T00:00Z"/>

<retention limit="days(10)"action="delete"/>

<sla slaLow="hours(3)"slaHigh="hours(4)"/>

<locations>

<location type="data"path="/hdfsDataLocation/S{YEAR}/S{MONTH}/S{DAY}"/> <location type="stats"path="/projects/falcon/clicksStats"/>

<location type="meta"path="/projects/falcon/clicksMetaData"/>

</locations>

</cluster>

</clusters>

location标签指定location的类型(如data 、meta 、stats) 及其相应的路径。feed 应 该至少定义location的数据类型，从而指定HDFS 的 feed 周期性发生的路径模式。例如， type="data"path ="/projects/TrafficHourly/${YEAR}-${MONTH}-${DAY}/trafic", 路径中的 日期间隔样式应该至少有一个频移。其他的location-type应该是 stats 和 meta 地址，如果一 个进程参考一个 feed,那么 meta和stats 地址在进程中是有效的。

<location type="data"path="/projects/falcon/clicks"/>

**第3章** **大数据治理之Apache Falcon** 67

注意

<location type="stats"path="/projects/falcon/clicksStats"/>

<location type="meta"path="/projects/falcon/clicksMetaData"/>

4)目录存储(table):table标签在注册表中指定table URL。

catalog:sdatabase-name:Stable-name#partition-key=partition-value);partition-key= partition-value);\*

这里作为一个 URI 建模(类似一个 ISBN URI)。它对 Hive 和 Hcatalog 没有任何参考。 它十分普遍，以至于可以绑定到目录注册表信息中。配置文件 startup 中目录实现配置项， 为目录 URI 的实现提供了相关信息。

顶层分区必须是一个过时的模式，日期模式的间隔至少有一个频移。

<xs:complexType name="catalog-table">

<xs:annotation>

<xs:documentation>

catalog specifies the uri of a Hive table along with the partition spec. uri="catalog:$database:Stable#(partition-key=partition-value);+"

Example:catalog:logs-db:clicks#ds=S(YEAR}-S{MONTH}-${DAY} </xs:documentation>

pe="xs:string"name="uri"use="required"/>

</xs:complexType>

5)分区 (partitions): 一个 feed 可以定义多个分区，如果一个引用的集群定义分区， 那么在 feed 中分区的数量应该大于或者等于集群分区数量。

<pan name="country"/>

<partition name="cluster"/>

</partitions>

这只适用于文件系统存储，而不适用于被定义和维护Hive(Hcatalog) 注册表的表

存储分区。

6)分组 (groups): 一个 feed 指定一个逗号分隔组列表，group 是 feed 的一个逻辑分组， 如果属于一个组的所有feed 有效，则这个组也是有效的。属于相同 group 的所有 feed 的频 率都应该是一样的。

<groups>online,bi</groups>

7)可用性标志 (availabilityFlag): 一个 availabilityFlag 指定在 feed 数据目录创建的文 件名称，该 feed 则称为有效的。如\_SUCCESS, 如果这个元素被忽略，那么 Falcon 将考虑 feed 的数据目录的存在性。

<availabilityFlag>\_SuccEss</availabilityFlag>

8)频率 (frequency):feed 有其数据产生的特定频率。例如，它规定产生的频率为每小

**68**  第二篇 开源实现篇

时、每周、每5分钟、每天等有效的频率类型，因为feed 是几分钟、几小时、几天、几个 月。该值可以是负数、零或者正数。

<frequency>minutes(20)</frequency>

9)SLA:feed 可以设置SLA, 每 个SLA 有两种属性-slaLow 和 slaHigh 。slaLow 和 slaHigh 都是使用频率表达式进行说明的。slaLow 的作用是当 feed 实例处于失去有效性的 危险时提供警告。slaHigh 的作用是当feed 丢失 SLA时做出报告。SLA 是相对于 feed 实例 的时间。

<sla slaLow="hours(40)"slaHigh="hours(44)"/>

10)数据导入 (import):feed 有一个和它相关的导入策略。 source name 指定数据被导 入HDFS 中参照的数据源实体。tableName 指定表或者 topic 从数据源导入。 extract type 指 定 pull 机制(全部或增量提取)。全部的提取方法是从数据源中提取所有的数据。增量提取 方法的功能实现正在进行中。mergepolicy 决定数据如何被布置在HDFS 中 。snapshot 布局 是指在HDFS 中 用feed 的 location 规范创建 snapshot 数据。field 用于指定投影队列。feed 从数据库后台使用Sqoop 实现任务。任何提前的 Sqoop 选项都能通过argument 来指定。

<import>

<source name="test-hsql-db"tableName="customer">

<extract type="full"><mergepolicy>snapshot</mergepolicy></extract>

<fields><includes><field>id</field><field>name</field></includes></fields>

</source>

<arguments>

<argument name="--split-by"value="id"/>

<argument name="--num-mappers"value="2"/>

</arguments>

</import>

11)延迟到达策略 (late-arrival): 当 feed 希望能够延迟运行时，可以通过late-arrival 来指定 cut-off 值的大小达到延迟效果。例如，feed 文件希望能够延迟6小时运行，则 late- arrival 应该设置为： late-arrival cut-off-"hours(6)"。如以下代码所示：

<veme->arrival cut-off="hours(6)"/>

 这只适用于文件存储系统，目前还不适用于表存储。

12)电子邮件通知 (email notification): 用 type 属性指定的notification 元素允许用户 在一个预定的 feed实例完成时接收 email notification。电子邮件的多个收件人可以提供为 to 属性的逗号分隔的地址。发送电子邮件通知确保SMTP 参数在 Falcon startup.properties 上被定义。用户可以参考 Falcon email notification 获得更详细内容。

<notification type="email"to="bobexyz.com"/>

第3章大数据治理之Apache Falcon 69

13)ACL:feed 有ACL(Access Control List), 用于实施许可的要求和为特定用户或命 名群组提供设定不同权限的方法。

<ACL owner="test-user"group="test-group"permission="\*"/>

ACL表示这个集群的访问控制列表；owner 是这个实体的所有者； group 是指由权限进 行读操作；permission 表示权限。

14)消费属性(custom property): 一个传播到工作流引擎的键值对。queueName 和 jobPriority 具有特殊属性，分别用于指定Hadoop作业队列和优先级。同一value 被 Falcon 的 launcher工作使用。timeout 、parallel 和order 都是特殊属性，指定了等待 feed 实例时， 如果超时保留feed实例的时间。parallel 决定并行复制实例，能够以任意给定的时间和顺 序运行，如FIFO 、LIFO 和LAST\_ONLY。maxMaps 代表复制时使用的map 的最大数量。 mapBandwidth代表当复制时每个映射私用的带宽。overWrite 表示在复制过程中覆盖目标。 ignoreErrors 代表在复制过程中忽略不导致工作失败的错误。skipChecksum代表在复制过 程中绕过校验。removeDeletedFiles 表示在复制中删除目标存在的文件而不是源中的文件。 preserveBlockSize 表示在复制过程中保留块的大小。preserveReplicationNumber表示在复制 过程中保留复制数。preservePermission 表示在复制过程中保留权限。

<proy name="tmpFeedPath"value="tmpFeedPathValue"/>

<property name="field2"value="value2"/>

<property name="queueName"value="hadoopQueue"/>

<property name="jobPriority"value="VERY\_HIGH"/>

<property name="timeout"value="hours(1)"/>

<property name="parallel"value="3"/>

<property name="maxMaps"value="8"/>

<property name="mapBandwidth"value="1"/>

<property name="overwrite"value="true"/>

<property name="ignoreErrors"value="false"/>

<property name="skipChecksum"value="false"/>

<property name="removeDeletedFiles"value="true"/>

<property name="preserveBlockSize"value="true"/>

<property name="preserveReplicationNumber"value="true"/>

<property name="preservePermission"value="true"/>

<property name="order"value="LIFO"/>

</properties>

15)生命周期 (lifecycle):lifecycle 标签是feed 生命周期的各个阶段的新方法。在下 面例子中，定义了一个retention-stage 使用lifecycle 标签。你可以在群居级别、群集级别或 者两者中定义 lifecycle, 也可以在global level 、cluster level或者两者中定义lifecycle。如 果 cluster level规范丢失，cluster level 配置的优先级和 Falcon 回到 global 定义中。

<lifecycle>

<retention-stage>

<frequency>hours(10)</frequency>

**70** **第二篇** **开源实现篇**

<queue>reports</queue>

<priority>NORMAL</priority>

<properties>

<property name="retention.policy.agebaseddelete.limit" value="hours(9)"> </property>

</properties>

</retention-stage>

</lifecycle>

**3.process** **实体声明项**

该项说明 process与feed 之间的依赖关系，介绍xml 配置项中clusters 、frequency、 inputs 、outputs 以及 retry 的作用。

process 定义工作流的配置： 一个工作流是一个有向无环图 (DAG), 用于定义工作流引 擎的作业。 一个 process definition定义了运行工作流所需的配置。例如： process 定义工作 流需要运行的频率，工作流应该运行的集群，工作流的输入和输出，以及工作流失败的处 理方式，后期输入的处理方式等。

1)名称(name): 每个process 都以一个唯一的 name 进行定义。语法如下。

<process name="[process name]">

</process>

2)标签 (tags): 用于对process 分类(用逗号分隔)的可选列表。语法如下。

<tags>consumer=consumer@xyz.com,owner=producer@xyz.com,department=forecasting</tags>

3)管道 (pipelines): 指定 process 所属的数据处理流水线，是可选项列表，用逗号分 隔。只有字母、数字、下划线被允许用于 pipeline 字符串。语法如下所示。

<pipelines>test\_Pipeline,dataReplication,clickstream\_pipeline</pipelines>

4)集群 (cluster): 工作流在运行的集群。 一个进程应该包含一个或多个集群。cluster 为集群定义工作流执行、名称节点、工作追踪、消息传递等提供了终端。cluster 标签内部 说明了作业在指定集群上的运行时间。语法定义如下：

<process name="[process name]">

<clusters>

<cluster name="test-clusterl">

<validity start="2012-12-21T08:15Z"end="2100-01-01T00:00Z"/> </cluster>

<cluster name="test-cluster2">

<validity start="2012-12-21T08:15Z"end="2100-01-01T00:00Z"/> </cluster>

</clusters>

</process>

5)并发 (parallel):parallel 定义了工作流的多个实例可以并发运行的情况。它应该是

一个正整数。其语法如下。

第3章 大数据治理之Apache Falcon 71

<process name="[process name]"><parallel>[parallel]</parallel> </process>

6)序列 (order):order 定义了执行已准备的实例顺序，可以是FIFO (先进先出)、 LIFO(后进先出)或 ONLYLAST。语法定义如下。

<process name="[process name]"><order>[order]</order></process>

7)暂停 (timeout):timeout 指定了一个实例在被工作流引擎杀死之前等待 dataset的最 长时间。timeout 为指定的频率。语法定义如下。

<process name="[process name]">

<timeout>[timeunit]([frequency])</timeout>

</process>

8)频率(frequency):frequency 定义了工作流要执行的频率。frequency 的次数应该是

大于0的正整数。语法定义如下。

<process name="[process name]">

<frequency>[timeunit]([frequency])</order>

</process>

9)SLA: 一个进程可以有两个可选属性定义的SLA——shouldStartIn 和 shouldEndln。 所有的属性都是使用频率的表达式编写的。shouldStartln 为process 的开始时间， shouldEndln 为process 的结束时间。

<sla shouldstartIn="hours(2)"shouldEndIn="hours(4)"/>

10)有效性 (validity):validity 定义了工作流应该运行多长时间。它由三部分组成—— start time 、end time 和timezone。

start time 和 end time是以yyyy-mm-dd'T'HH:mm'Z '形式的UTC时间戳定义的， timezone用来从start time 开始计算下一个实例。工作流将开始于start time, 在 end time 指 定的集群之前结束。语法定义如下。

<process name="[process name]">

<validity start=[start time]end=[end time]timezone=[timezone]/>

</process>

<process name="sample-process">

<frequency>hours(1)</frequency>

<validity start="2012-03-11T08:40Z"end="2012-03-12T08:00"timezone="PST8PDT"/> </process>

样例中的工作流将在2012年3月11日08:40开始运行，下面的实例分别在09:40、 010:40等运行， 一直到第二天的7:40结束。因为DST 转换，所以在2012年3月11日只 有23个工作流实例。

11)输入 (input):input 定义了工作流的输入数据。工作流作业将在规定时间之后并且 所有的输入都有效时才开始执行。至少要有0个input, 每个 input 都记录在feed 中。输入数 据的路径和频率是在feed中获得的。每个input在表达式中也应该定义开始和结束实例，能

72 第二篇 开源实现篇

够可选地指定工作流所需的输入的特定分区。分区中的组件都应该在 feed 的分区子集中定义。 对于每个 input,Falcon 将创建一个包含 input name的属性，其中包含使用逗号分隔的

输入列表。此属性可以用于像Pig 脚本这样的工作流的操作。

语法定义如下：

<process name="[process name]">

<input name=[input name]feed=[feed name]start=[start el]end=[end el]

partin=[partition]/>

</inputs>

</process>

工作流的 input 是一小时的 feed, 并需要每天开始一小时的数据(从工作流开始工作开 始)。如果工作流运行于2012-03-01T06:40Z, 那么 input 为projects/bootcamp/feed1/2012- 03-01-00/\*/US和 projects/bootcamp/feed1/2012-03-01-01/\*/US 。此输入的属性为input1= /projects/bootcamp/feed1/2012-03-01-00/\*/US 和/projects/bootcamp/feed1/2012-03-01-01/\*/ US。此外。feed 和 Hive 的表存储可以作为一个流程的输入。

${wf:conf('falcon\_input\_database'))-database name associated with the feed for

a given input

${wf:conf('falcon\_input\_table')}-table name associated with the feed for a

given input

s{wf:conf('falcon\_input\_catalog\_url')}-Hive metastore URI for this input feed

s(wf:conf('falcon\_input\_partition\_filter\_pig')}-value of ${coord:dataInPartition Filter('sinput','pig'

$(wf:conf('falcon\_input\_partition\_filter\_hive')}-value of ${coord:dataInPartitio nFilter('sinput','hive

$(wf:conf('falcon\_input\_partition\_filter\_java')}-value of ${coord:dataInPartitio

nFilter('Sinput', 'java')}

|  |
| --- |
| 素 input 是输入配置的进程的名称，也就是 input.getName()。  <input name="input"feed="clicks-raw-table"start="yesterday(0,0)"end="yesterday (20,0)"/>  12)可选输入 (optional input): 用户可以提及一个或多个输入作为optional input。在 这种情况下，作业不会等待那些被提及的optional input。如果这些可选操作确实存在，则 它们会被作业考虑是否使用，否则作业将继续执行，而无须等待optional inputs 的输入。如 果一些 optional feed的实例存在于给定的数据窗口，那么这些 optional input 会被考虑和传 递到进程中去。当检查一个 feed 实例的存在时，如果在 feed 定义中被指定，那么在目录中 寻找 availabilityFlag。如果没有 availabilityFlag 被指定，那么已存在的实例目录被看作有效 的数据路径。 |
| 以上仅支持文件存储系统，而不包含表存储。 |

第3章 大数据治理之Apache Falcon 73

13)输出(output):output 定义由工作流生成的输出数据。 一个进程可以定义0个或多 个output。每个 output 被映射到一个feed, 并 从 feed 定义中获得路径。输出实例应该在表 达式中指定。

对于每个 input,Falcon 创建一个具有输出名称的属性，该属性包含输出数据的路径。 这可以用于路径中存储的工作流程。语法定义如下：

<process name="[process name]">

<outputs>

<output name=[input name]feed=[feed name]instance=[instance el]/>

</outputs>

</process>

工作流的输出就是今日的 feed 实例。如果工作流在2012-03-01T:40Z 开始运行，那么 工作流生成的 output为/projects/bootcamp/feed2/2012-03-01。这个可用的工作流的output 属性为 output1=/projects/bootcamp/feed2/2012-03-01。

此外，feed 和Hive 表存储可以作为一个过程的输出，从输出参数传递参数给用户的工 作流或 Pig脚本。

${wf:conf('falcon\_output\_database'))-database name associated with the feed for a given output

${wf:conf('falcon\_output\_table'))-table name associated with the feed for a

given output

S{wf:conf('falcon\_output\_catalog\_url'))-Hive metastore URI for the given output feed

${wf:conf('falcon\_output\_dataout\_partitions'))-value ons('Soutput')}

of ${coord:dataOutPartiti

output 是输出过程中配置的，也就是 output.getname()。

<output name="output"feed="clicks-summary-table"instance="today(0,0)"/>

14)消费属性 (custom property):custom property 定义了传递给工作进程的键值对。 这些键值对是可选的，并可以使用参数化定义对工作流传值。语法定义如下。

<process name="[process name]">

<properties><property name=[key]value=[value]/></properties>

</process>

以下是一些现在使用的由Falcon 发布的任务的特殊属性，同样的属性也可以用来传播 到 Pig或者 M/R上的工作流。

<property name="queueName"value="hadoopQueue"/>

<property name="jobPriority"value="VERY\_HIGH"/>

<property name="userJMSNotificationEnabled"value="false"/>

74 第二篇开源实现篇

示例中的属性值 userJMSNotificationEnabled 用于关闭 process的 JMS 消息通知，如 果未设置则默认开启。

**3.3** **Apache Falcon 场景设计与实现**

前面已经介绍了Apache Falcon的架构、发展史以及Falcon 的安装配置和使用。读者 可以发现， Falcon 作为一个简化数据管道处理和管理Hadoop 集群工作系统架构，用户能够 利用它轻松地定义各种数据和处理数据元素之间的关系，简化数据的配置，这对于大数据 的治理有重大意义。本节将会利用各个场景介绍 Apache Falcon 如何运行以及各个组件的意 义，同时进一步理解 Apache Falcon 对大数据治理的意义。

**3.3.1** **数据管道**

数据管道场景是通过Falcon 调 用Oozie MapReduce工作流来实现的。Falcon 先利用 Hadoop 工作流调度器对Oozie 进行调度， Oozie 再调用Oozie 工作流进行操作，然后使用 MapReduce 对数据进行处理，进行数据的复制保留等操作，最后将数据保存在 HDFS集群中。

**1.前期准备**

该场景需要有一个数据源，读者可以自主创建一个数据源。在该场景中，数据源是利 用bash写的一个 bash 数据脚本文件。

在linc-1 节点上，使用hdfs 用户，在任意目录下使用命令vim general.sh 创建一个数据 脚本文件。该脚本的内容如下：

rm -rf generated-data

EAR='date +8Y`

'e 8m'

0UR=`date +%H`

first second third fourth fifth)

for MINUTE in`seq -w 0059`

do

mkdir -p generated-data/00/SMINUTE/

word=${input[$RANDOM 85]}

cnt='expr SRANDOM 810

echo -e "SwordSDELIMScnt">generated-data/00/SMINUTE/data

done

hadoop fs -rmr /data/in/2013/11/15/

hadoop fs -mkdir -p /data/in/2013/11/15/

hadoop fs -put generated-data/00 /data/in/2013/11/15/

rm -rf generated-data

该脚本首先删除 generated-data 文件，以免之前创建过；然后随机生成从2013年11 月15日0点0分到59分的数据集，保存在generate-data/00 中；最后，循环结束后，将

第3章 大数据治理之 Apache Falcon 7 5

generated-data/00 中的数据存储在HDFS 的 /data/in 目录下。将该脚本文件保存退出后，进 入该脚本目录使用命令“/general.sh” 执行脚本文件。在脚本文件执行后，读者可以登录 集群在 HDFS 的 Web UI(http://linc-1:50070), 进入/data/in/2013/11/15/00目录即可看到创 建的 name 为00-59的数据。



表 HDFS 的 Web 默认端口号为50070,读者如果修改了端口号，需要以自己修改的端 口号为准。

**2.提交** **Oozie 工作流**

工作流 (workflow) 是工作流程的计算模型，是对工作流程及各操作步骤之间的业务规 则的抽象、概括描述。本书中的工作流定义了在HDFS 上使用以及工作流路径的工作流引 擎。Libraries (库)需要指定使用工作流中的自由属性元素，并使用逗号分隔 HDFS 路径。 HDFS 的工作流定义包含应该运行的实际作业，并应该确认引擎的工作流程规范。工作流 所需要的Libraries 应该在工作流路径中的lib 文件夹中。在集群和集群属性中定义的属性 (nameNode 和jobTracker) 也将用于工作流中。主要应用的工作流有Oozie 工作流、Pig 工 作 流 、Hive 工作流、Spark 工作流，这里采用的是Oozie 工作流。

Oozie 工作流是控制依赖有向非循环图 (DAG) 中安排的 Oozie 操作的集合。控制依 赖 (control dependency) 可确保之后的操作在前面的操作已成功完成后才会启动，其大多数 是基于XML 定义的。而 Oozie 是一个以工作流的形式对 Hadoop Map/Reduce 任务、Pig 任 务、Hadoop 文件系统、Java 和 Oozie的子工作流进行调度的系统。控制流节点定义了工作 流的起始和结束(例如 start 、end 、fail 节点)以及提供了工作流执行路经的控制机制(如 decision 、fork、join 节点)。

Oozie 工作流可以被参数化(例如使用“${inoutDir}” 的参数),当提交一个具有参数 的工作流时，必须提供它的参数。如果同时提供多个不同的输出路经，那么这个工作流可 以同时执行。

在linc-1 节点上，使用hdfs 用户，使用命令 “vim workflow.xml” 创建一个 Oozie 工 作流文件。Oozie 的语法如下：

<workflow-app xmlns="uri:oozie:workflow:0.2"name="map-reduce-wf"> <start to="mr-node"/>

<end name="end"/>

</workflow-app>

其中<start to="mr-node">是开始节点，是工作流工程的切入点，它表明第一个工作 流节点工作流job必须转变。当工作流启动时，它会自动转换指定的开始节点。 一个工 作流必须有一个开始节点。在该配置中会首先从 “mr-node” 节点开始执行。同样 “end name=“end”” 是一个end 节 点 ，end 节点是一个工作流的最后，表明该工作流的工作已经 成功完成。当一个工作流job 运行到 end 节点时该job 会显示 SUCCEEDED 状态。如果一

**76** 第二篇 开源实现篇

个或多个action 因为工作流job 开始执行而开始，但是已经运行到了end 节点，那么这些 action 将会被杀掉，而在这个场景中工作流的job 仍然会被视为成功运行。

在 start 和 end 节点之间添加如下配置：

<kill name="fail">

<message>Map/Reduce failed,error message[S{wf:errorMessage(wf:lastErrorNode())}]</

message>

</kill>

kill节点允许工作流job杀死它本身。当一个工作流job运行到了kill节点而结束 时，状态会显示error(KILLED) 。 如果一个或者多个action 因为工作流 job 开始执行而开 始，但是已经运行到了kill节点，那么这些action将会被杀掉。 一个工作流可以有零个或 多个kill 节点。在上述添加的配置中，kill 节点中的name 属性是 kill action节点的名称。

message 元素的内容是将会被记录的工作流job 被杀的原因。

在该工作流文件的 start 节点和 end 节点之间添加如下内容：

<action name="mr-node">

<map-reduce>

</map-reduce>

r=l"/>

</action>

其中<action name="mr-node">…</action> 定义了一个名为 “mr-node” 的 action 节 点 ，action节点是一个工作流促动器的执行计算或处理任务的机制。<map-reduce></map- reduce> 表示工作流中开始一个 Hadoop 的 Map/Reduce 的job 。使用 map-reduce 模式在开 始map reduce 作业之前可以配置为执行文件系统清理和目录创建。

在 map-reduce 中添加如下内容：

<job-tracker>${jobTracker}

</job-tracker><name-node>S(nameNode }</name-node>

<prepare>

<delete path="${outpath}"/>

</prepare>

<configuration>

</configuration>

上述代码在 map-reduce 中定义了job-tracker 和 name-node, 在 configuration 中设置了

map reduces

在 configuration 中添加如下内容：

<property>

<name>mapred.job.queue.name</name>

<value>${queueName}</value>

</property>

第3章 大数据治理之 Apache Falcon 77

其中<name>mapred.job.queue.name</name>设置了map reduce 的 job队列的名称。

<property>

<name>mapred.mapper.class</name>

<value>org.apache.hadoop.mapred.lib.IdentityMapper</value>

</property>

<name>mapred.mapper.class</name>的作用是设置 map所调用的类。

<property>

<name>mapred.map.tasks</name>

<value>1</value>

</property>

<name>mapred.map.tasks</name>的作用是设置 map任务的个数。

<property>

<name>mapred.input.dir</name>

<value>${inpaths}</value>

</property>

<name>mapred.input.dir</name>的作用是设置数据源地址。

<property>

<name>mapred.output.dir</name>

<value>${outpath}</value>

</property>

<name>mapred.output.dir</name>的作用是设置数据目标地址。

保存 workflow.xml 文件并退出后，将该工作流文件上传到 HDFS的/app/oozie-mr 目录 下，使用如下命令：

s hdfs dfs -put workflow.xml /app/oozie-mr

读者需要注意的是，上传该文件前，需要在HDFS 上新建/app/oozie-mr目录，否则

程序将会报找不到相应目录的错误。

**3.创建并提交** **cluster 实体**

在linc-1 节点上，使用hdfs 用户，在任意目录下使用命令 “vim local-cluster.xml”创 建一个cluster文件。cluster 实体会在之后被feed 实体以及 process 实体所依赖。将一个 cluster.xml实体提交给 Falcon后，在Falcon 中有唯一的 cluster name 与其对应。

在 local-cluster.xm1文件中添加如下内容：

<?xml version="1.0"?>

<cluster colo="local"description="local cluster"name="local"xmlns="uri:falcon:

cluster:0.1">

</cluster>

上述内容为 cluster.xml文件的标签定义，其中colo 指明属于哪一个集群，name 是指集

**78** **第二篇** **开源实现篇**

群的名称，并且该名称必须是唯一的。

**一个** **cluster包含有不同的接口，而这些接口有** **readonly、write 、execute、workflow 和**

messaging 属性。在 local-cluster.xml 文件中添加多个接口定义如下：

<interfaces> <interface

<interface

<interface

<interface

<interface

"5.4.3"/> </interfaces>

type="readonly"endpoint="hdfs://linc-1:9000"version="2.6.0"/>

type="write"endpoint="hdfs://linc-1:9000"version="2.6.0"/> type="execute"endpoint="linc-1:8021"version="2.6.0"/>

type="workflow"endpoint="http://linc-1:11000/oozie/"version="4.2.0"/> type="messaging"endpoint="tcp://linc-1:61616?daemon=true"version=

其中 readonly接口为 Hadoop HFTP协议指定了终端为hdfs://inc-1:9000, 这会在后 文中的feed 实体复制的内容中用到。write 接口指定写入HDFS中的端口，它的终端是 fs.defaultFS的 IP 地址。Falcon 用该接口写系统数据到HDFS中 ，feed引用该集群的相同 write接口写数据到HDFS中。execute 接口指定JobTracker 程序所需要的执行接口，它的 终端是 mapreduce.jobtracker.address 的 IP地址，在 Hadoop 2.x版本中，此接口值为YARN 指定的执行接口。workflow接口指定工作流引擎的接口，例子中的终端是OOZIE\_URL的 value,Falcon 用该接口来安排在工作流引擎上所引用的集群进程。messaging 接口指定发送 Falcon 可用消息的接口，该接口的终端联合了TCP 和URL 地址。

在 cluster中还需要定义一个位置列表。location有两种属性值name 和 path。name 是 location的类型，被允许的name值有 staging、temp 和 working。path 是每个 location 的 HDFS 路径。Falcon 在HDFS中使用location 进行实体间接处理，因此Falcon 在这些location 中 具有read 、write 和execute 权限。在 local-cluster.xml文件中添加多个location 定义如下：

<locations> <location

<location

<location

name="staging"path="/projects/falcon/staging"/><!--mandatory--> name="temp"path="/projects/falcon/tmp"/><!--optional-->

name="working"path="/projects/falcon/working"/><!--optional-->

</locations>

读者需要注意的是： staging 应该具有777权限，是一个强制性的location, 父级目 录必须有执行权限让多个用户能够在该location 具有执行写操作的权限。working必须具 有775权限，是一个可选的location, 如 果working 没有被指定，那么Falcon会以775权 限在 staging location 中创建一个子目录。working 的父级目录必须有能够让多个用户在该 location 中进行读操作的权限。

在新建的 local-cluster.xml文件中添加上述内容并保存退出后，将该.xml文件提交到 falcon-server 中，使用的命令如下：

S FALCON\_HOME/bin/falcon entity -submit -type cluster -file ./local-cluster.xml

注意，命令中的FALCON\_HOME是 Falcon 的安装目录，读者可根据实际情况进行修

第3章大数据治理之 Apache Falcon 79

改。在向 falcon-server 提交 cluster 实体文件后，若在控制台提示 falcon/default/Submit successful(cluster)local-cluster, 则表示提交cluster 实体成功。登录Falcon的 Web UI (http://linc-1:15443) 后，进入cluster 目录即可看到提交成功的 local, 单击local 后，会看 到该 cluster 文件的内容。

**4.创建并提交** **in-feed 实体**

成功提交了cluster 到 falcon-server后，需要提交一个feed 实体到 falcon-server。一 个feed 定义了各种不同的属性，例如 location 、late-arrival和 retention 策略。feed 同时还 可以安排在一个集群中， 一旦feed 的 retention安排在一个给定的集群上，复制进程就会被 触发。

在linc-1 节点上，使用hdfs用户，在任意目录下使用命令 “vim in-feed.xml” 创建输 入数据集 in-feed.xml 文件。在该文件中添加如下内容：

<?xml version="1.0"encoding="UTF-8"?>

<feed description="input"name="in"xmlns="uri:falcon:feed:0.1"></feed>

上述内容为feed.xml 文件的标签定义， feed 应该有且只有一个特定的name, 该 name 的值会在之后的 process 实体中被引用。

再在 in-feed.xml 文件中添加如下内容：

<clusters>

<cluster name="local">

<validity start="2013-01-01T00:00z"end="2030-01-01T00:00z"/>

<retention limit="hours(2)"action="delete"/>

</cluster></clusters>

在添加的 cluster中 ，cluster name 是前面提交到 falcon-server上的 cluster 的名称。feed 通过集群的name 来进行引用，在提交一个feed 之前，所有的引用集群都应该被提交给 Falcon。validity指定了集群上有效的持续时间。retention 指定了这个集群上feed retention 的时间长度以及retention 结束后在 feed上的活动。retention limit 由表达频率(次数)指定， 例如，如果feed应该保留至少2小时，那么 retention limit="hours(2)"。

然后在 in-feed.xml文件中添加以下location:

e="data"path="/data/in/S(YEAR)/$(MONTH}/S{DAY}/S(HOUR}/S(MINUTE)"/>

</locations>

location 标签指定location 的类型，在这里为 “data” 类型相应的路径。feed 应该至 少定义一种location的数据类型，从而指定HDFS的 feed 周期性发生的路径模式。例如， type="data"path="/data/in/S{YEAR}/S{MONTH}/${DAY}/S{HOUR}/S{MINUTE}", 路径中 的日期间隔样式应该至少有一个频移。这里的 path 指定了前文在 HDFS上创建的数据源。

feed 还有ACL(Access Control List), 用于实施许可的要求和为特定用户或命名群组提 供设定不同权限的方法。在XML文件中添加 ACL:

80 第二篇 开源实现篇

<ACL owner="testuser-ut-user"group="group"permission="0x644"/>

其 中ACL表示这个集群的访问控制列表， owner 是这个实体的所有者，permission 表 示实体所拥有的权限。

在新建的 in-feed.xml文件中添加上述内容并保存退出后，将该文件提交到 falcon-server 中。提交该实体使用的命令如下：

§FALCON\_HOME/bin/falcon entity -submit -type feed -file ./in-feed.xml

需要注意的是： FALCON\_HOME是 Falcon的安装目录，读者可根据实际情况进行 修改。

在 向falcon-server 提交feed 实体文件后，若在控制台提示 falcon/default/Submit successful (feed)in-cluster,则表示提交该feed实体成功。登录 Falcon的 Web UI(http://linc- 1:15443)后，进入feed目录即可看到提交成功的in, 单击in 后，会看到该 feed文件的内容。

**5.创建并提交** **out-feed 实体**

前面提到的in-feed 实体是作为输入数据集，需要再创建一个out-feed.xml 作为输 出数据集。文件的创建和内容与 in-feed.xml 文件相似，只需要修改部分参数值。将in- feed.xml 文件中的 description="input"name="in" 改为 description="output"name="out", 将<group>input<group>换 为<group>output<group>, 将 frequency改 为 minutes(5), 将 path 换为“/data/out/${YEAR}/S{MONTH}/${DAY}/S{HOUR}”。

在新建的out-feed.xml文件中添加上述内容并保存退出后，将该文件提交到 falcon- server 中，使用的命令如下：

S FALCON\_HOME/bin/falcon entity -submit -type feed -file ./out-feed.xml

在 向falcon-server 提交 feed 实体文件后，若在控制台提示falcon/default/Submit successful(feed)out-cluster, 则表示提交feed 实体成功。登录Falcon的 Web UI(http:// linc-1:15443)后，进入feed目录即可看到提交成功的out, 单击out后，会看到该feed文 件的内容。

**6.创建并提交** **process 实体**

在向 falcon-server成功提交了cluster 实体和 feed 实体后，需要继续提交 process 实体。

在linc-1 节点上，使用hdfs 用户，在任意目录下使用命令 “vim filter-process.xml” 创 建一个 process 文件。然后在该文件中添加如下内容：

<?xml version="1.0"encoding="UTF-8"?>

<process name="filter-process"xmlns="uri:falcon:process:0.1"></properties>

其中 name 是该process 实体的名称，每一个process 都以一个唯一的 name 进行定义。 将该process 的 name 定义为 filter-process。

在该 process 文件的<process></process> 之间添加如下内容：

第3章 大数据治理之Apache Falcon 81

<clusters>

<cluster name="local">

<validity start="2013-11-15T00:05Z"end="2013-11-15T01:05Z"/>

</cluster></clusters>

这里定义了工作流应该运行的集群。 一个进程应该包含一个或多个集群。cluster 为 集群定义工作流执行、名称节点、工作追踪、消息传递等提供了终端。在上述内容中， validity 定义了工作流应该运行多长时间，它由start time 和 end time 两部分组成。start time 和 end time 是以 yyyy-mm-dd'T'HH:mm'Z '形式的UTC 时间戳定义的。工作流将开始于 start time, 在 end time 指定的集群之前结束。

然后继续在 process 文件中添加如下内容：

<parallel>1</parallel>

<order>FIFO</order>

<frequency>minutes(5)</frequency>

<timezone>UTC</timezone>

其中 parallel 定义了工作流的多个实例可以并发运行的情况。它的值应该是一个正整 数。在这里设置为正数1。frequency 定义了工作流需要执行的频率， frequency 的值应该是 大于0的正整数，在这里设置为5(分钟)。

在 process 文件中添加如下内容：

<inputs> <input name="inpaths"feed="in"start="now(0,-5)"end="now(0,-1)"/></inputs>

<outputs> <output name="outpath"feed="out"instance="now(0,0)"/> </outputs> 上述 input 定义了工作流的输入数据。而 output 定义了由工作流生成的输出数据。 在 process 文件中添加如下内容：

<workflow engine="oozie"path="/app/oozie-mr"/>

这里是指定运行的工作流引擎以及目录位置信息。

然后在process 文件中添加如下内容：

<properties>

<property name="queueName"value="default"/>

<property name="time"value="${instanceTime()}"/>

</properties>

其中property 属性定义了传递给工作进程的键值对。

在新建的 process-feed.xml 文件中添加上述内容并保存退出后，将该文件提交到 falcon- server 中，使用的命令如下：

S FALCON\_HOME/bin/falcon entity -submitAndSchedule -type process -file ./filter- process.xml

在向 falcon-server 提交process 实体文件后，若在控制台提示 falcon/default/Submit successful (process)filter-process,则表示 process 实体提交成功。登录 Falcon 的 Web UI

82 **第二篇** **开源实现篇**

(http://linc-1:15443) 后，进入process 目录即可看到提交成功的 filter-process, 单击 filter- process 后，会看到该process 文件的内容。

**7.结果显示**

当 process 上传成功后，读者可以在in-feed 实体指定的/data/in/2013/11/15/00目录下 看到生成的目标源数据。

同时在out-feed 实体指定的/data/out/2013/11/15/00目录下生成 process 运行后的结果 文件，里面存储每次运行时前5分钟的数据集合。读者可以在一定时间后在 HDFS 上查看 该目录下的文件。

**3.3.2** **结构化数据导入分布式文件系统**

Falcon 提供了周期性地将外部数据作为源数据导入 Hadoop 以及将经过 Hadoop 运算的 数据导出到外部的数据库的高度抽象。在至今所有的Falcon 版本中，仅支持经过JDBC 进 行连接的关系型数据库，将来的版本可能增加支持其他类型的数据库。

**1.先决条件**

确保 Hadoop 、Falcon以及 Oozie 安装成功，并已经进行 NTP时间校正，因为时间不同 步可能会影响 Falcon 的作业执行。

**2.数据准备**

在数据准备阶段，读者需要自行建立MySQL 数据，作为导入分布式文件系统的数 据源。

$mysql -uroot -pMyNewPass4!

MySQL>create database market;

MySQL>use market;

#创建数据源customers 表

MySQL>CREATE TABLE customers(cust\_idint NOT NULL AUTO\_INCREMENT,

cust\_name char(50) NOT NULL ,cust\_address char(50),

cust\_city char(50),cust\_country char(50),cust\_email char(255),

PRIMARY KEY(cust\_id));

#数据源插入数据

MySQL>INSERT INTO customers values(10001,'Coyote Inc.','200 Maple Lane','Detroit', 'USA','ylee@coyote.com');

MySQL>INSERT INTO customers values(10002,'Mouse House','333 Fromage Lane', 'Columbus','USA','ylee@coyote.com');

MySQL>INSERT INTO customers values(10003,'Wascals','1 Sunny Place','Muncie', 'USA','rabbit@wascally.com');

读者可以对操作结果进行验证，在 MySQL中执行以下语句：

MySQL>Select \*from customers;

如图3-5所示，可以观察到数据插入成功。

第3章 大数据治理之Apache Falcon 83

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| cust\_id cust name | | cust address | cust\_city l | cust\_country | cust\_email |
| 16001  10002  10003 | Coyote Inc. Mouse House Wascals | 200 Maple Lane  333 Fromage Lane 1 Sunny Place | Detroit  Columbus  Muncie | USA  USA  USA | yleeocoyote.com  yleeacoyote.com  rabbitowascally.com |

图3-5 MySQL表中源数据展示

**3.将** **MySQL** **中的数据导入HDFS**

(1)创建并提交 database 实体

进入Falcon 安装目录并且新建 mysql\_database.xml 文件：

s cd /var/local/hadoop/falcon-0.9

$vim examples/entity/filesystem/mysql\_database.xml

文件内容如下所示：

<?xml version="1.0"encoding="UTF-8"?>

<datasource colo="west-coast"description="MySQL database on west coast"type="mysql" name="mysql-db"xmlns="uri:falcon:datasource:0.1">

<tags>owner=foobar@ambari.apache.org,consumer=phoe@ambari.apache,org</tags> <interfaces>

i!y  - 1/market">

<credential type="password-text">

<userName>root</userName>

<passwordText>admin</passwordText>

</credential>

</interface>

上述 interface 定义了需要连接的MySQL 的用户名和密码，这个接口用于从 MySQL 中

**读取数据。**

ceen - 1/market">

<credential type="password-text">

<userName>root</userName>

<passwordText>admin</passwordText>

</credential>

</interface>

上述 interface 定义了需要连接的MySQL 的用户名和密码，这个接口用于在 MySQL 中 写入数据。

<!--\*\*\*default credential \*\*\*-->

<credential type="password-text">

<userName>root</userName>

<passwordText>admin</passwordText>

</credential>

</interfaces>

<driver>

**84** **第二篇** **开源实现篇**

注意

<clazz>com.mysql.jdbc.Driver</clazz>

<jar>/user/hdfs/share/lib/lib\_20160907192204/sqoop/mysql-connector-java-5.1.39-bin</jar> </driver>

上述 driver 属性定义了进行MySQL 连接时需要用到的jar 驱动包地址。

</datasource>

数据源实体抽象了与外部关系型数据库的连接过程和权限认证的细节，它定义了带有 特有权限的数据库读和写的接口。当读和写接口中的用户凭证不能通过验证时，将会使用 默认的登录凭证来进行连接数据库操作。通常情况下，数据源实体由管理员进行创建。具 体的参数指定如表3-3所示。

**表3-3** **database 参数详解**

|  |  |
| --- | --- |
| **参数名** | **定** **义** |
| type | interface参数中的type主要指定接口的类型。可以指定的值有：read(读接口)、write(写接口) credential参数中的type主要指定用户密码的类型。可以指定的值有：password-text(文本  类型)、password-file(文件类型)、password-alias |
| userName | 参数中指定了连接数据库的用户名 |
| passwordTex | 参数中指定了连接数据库的密码 |
| endpoint | 指定需要操作的数据库名 |
| driver | 指定连接的驱动 |
| clazz | 指定在连接时使用的驱动名 |
| jar | 指定连接数据库使用的驱动 |

根据实际情况， endpoint="jdbc:mysql://linc-1/market ":market 为在本节开头数据准 备中创建的数据库名。

需要使用以下命令对数据源实体进行提交：

s falcon entity -submit -type datasource -file examples/entity/filesystem/mysql.

database.xml

操作说明：

提示 falcon/default/Submit successful(datasource)mysql-db 则表示提交成功。Falcon Web UI目前无法显示已提交的 datasource实体，因此要查看提交的 datasource 实体，可使用如 下命令：

$Falcon entity -type datasource -list

(2)创建并提交 cluster 实体

进入Falcon 安装目录，并且新建 mysql\_cluster.xml 文件：

$cd /var/local/hadoop/falcon-0.9

$vim examples/entity/filesystem/mysql\_cluster.xml

第3章 大数据治理之 Apache Falcon **85**

**文件内容如下所示：**

<?xml version="1.0"?>

<cluster colo="ua2"description="MySQL Cluster"name="mysql-cluster"xmlns="uri:falcon: cluster:0.1"

xmlns:xsi="<http://www.w3.org/2001/XMLSchema-instance>">

faac - 1:9000"version="2.6.0"/>

<interface type="write"endpoint="hdfs://linc-1:9000"version="2.6.0"/>

<interface type="execute"endpoint="linc-1:8032"version="2.6.0"/>

<interface type="workflow"endpoint="http://linc-1:11000/oozie/"version="4.2.0"/> <interface type="messaging"endpoint="tcp://linc-1:61616?daemon=true"version=

"5.4.3"/>

</interfaces>

上述 interface属性主要指定了 Hive 进行任务调度时涉及的服务端口，例如 Hadoop 集 群的读写接口等。

s> name="staging"path="/projects/falcon/staging"/>

<location name="temp"path="/tmp"/>

<location name="working"path="/projects/falcon/working"/>

</locations>

</cluster>

上述 location 属性主要指定了任务在执行过程中使用到的文件目录。注意，只有 staging 是必须配置的属性，而 temp 、working可以不进行配置。

cluster 实体中定义了 Falcon执行任务过程中将会用到的各种接口。cluster 中具体的参

数如表3-4所示。

**表3-4** **cluster 参数详解**

|  |  |
| --- | --- |
| **参数名** | **定** **义** |
| name | 指定cluster实体的名字 |
| readonly | 指定读取HDFS的地址以及端口 |
| write | 指定写入HDFS的地址以及端口 |
| execute | 指定任务资源管理的Resource Manager的地址以及端口 |
| workflow | 指定Falcon的任务解析执行工具的地址以及端口(这里使用Oozie进行工作流解析调度) |
| messaging | 指定Falcon获取工作流执行状态的端口(此端口根据Oozie的配置文件指定) |
| staging | 在HDFS上指定任务工作台路径 |
| temp | 在HDFS上指定任务临时文件路径 |
| working | 在HDFS上指定任务工作依赖路径，里面会自动导人依赖的lib包 |

对 cluster 实体进行提交：

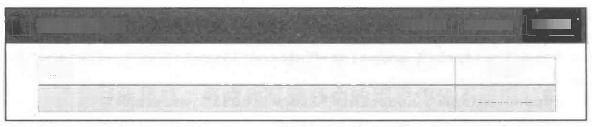
s bin/falcon entity -submit -type cluster -file examples/entity/filesystem/mysql\_

cluster.xml

86 第二篇 开源实现篇

提示 falcon/default/Submit successful (cluster)mysql-cluster 则表示提交成功。

在 Falcon 的 Web UI(https://<192.168.253.131>:15443)中进入Clusters 目录，即可看到 提交成功的 mysql-cluster, 如图3-6所示。

Apache

Falcon

Data cols Piocesses Clisters

**Name**

mysgl-duster

Status

SUBMITTED

图3-6 Falcon Web UI表中 Clusters 的显示

单击mysql-cluster, 可以看到提交的 cluster 实体的详细内容，如图3-7所示。

|  |  |  |  |
| --- | --- | --- | --- |
| mysql-cluster **undefined** | | | |
|  | Definition | | |
|  |  | | |
| <?xml verslon="1.0"encoding="UTF-8"standalone="yes"?>  <cluster name="mysql-cluster"descriplion="Mysql Cluster'colo=\*ua2"xmlns='urlfalcon.cluster.0.1">  <interfaces>  <nterface type="readonly"endpoint="hdts:// 900verslon="2.6.0>  <interface type="write"endpoint="hdfs:/ :9000"version="2.6.0">  <Interface type="execute"endpoint="s:8032 version=\*2.6.0\*/> | | |
| <interface <Interface | type="workflow"endpoint="http:// type="messaging"endpoint="tcp:// | 11000/oozie/"verslon="4.2.0">  61616?daemon=true"version="5.4.3"> |
| </interfaces>  <locations>  location name="staging"path="/projects/falcon/staging">  <location name="temp"path=\*/tmp">  <ocation name="working"path="/projects/falcon/working">  /locations>  <ACL owner="hdfs"group="hadoop">  </dlustep | | |

图3-7 Falcon Web UI 表中cluster 内容的显示

(3)提交并调度 feed 实体

在 Falcon 安装目录下新建 mysql\_in\_feed.xml 文件：

$cd /var/local/hadoop/falcon-0.9

$vim examples/entity/filesystem/mysql\_in\_feed.xml

cluster 文件内容如下所示：

<?xml version="1.0"encoding="UTF-8"?>

<feed description="raw mysql feed"name="mysql-feed"xmlns="uri:falcon:feed:0.1"> <tags>externalSystem=USWestEmailServers,classification=secure</tags>

u-iyv>amlicnuutt-e<"/)"/>

<clusters>

<cluster name="mysql-cluster"type="source">

**第3章** 大数据治理之Apache Falcon 87

<validity start="2016-09-17T09:00Z"end="2016-09-17T10:00z"/>

<retention limit="minutes(10)"action="delete"/>

<import>



<mergepolicy>snapshot</mergepolicy>

</extract>

<fields>

<includes>

<field>cust\_id</field>

<field>cust\_name</field>

<field>cust\_address</field>

<field>cust\_city</field>

<field>cust\_country</field>

<field>cust\_email</field>

</includes>

上述属性定义了需要进行导入的字段值，在默认情况下，是导出所有字段。

</fields>

</source>

</import>

</cluster>

</clusters>

<locations>

<location type="data"path="/user/hdfs/customers/S{YEAR}/S{MONTH}/${DAY}/${HOUR} /S(MINUTE}"/>

上述属性定义了将结构化数据库中的数据导入HDFS 中文件的位置，即目的位置。

<location type="stats"path="/none"/>

<location type="meta"path="/none"/>

</locations>

<ACL owner="hdfs"group="hadoop"permission="0755"/>

<schema location="/none"provider="none"/>

</feed>

feed 实体中定义了在 Falcon执行工作流的过程中将用到的数据集的各种属性，例如数 据集的位置、数据集使用的频率和数据集的有效期限等。feed 中具体的参数如表3-5所示。

**表3-5** **feed** **参数详解**

|  |  |
| --- | --- |
| **参** **数** **名** | **定** **义** |
| name | 指定feed实体的名字 |
| frequency | 指定工作流调度的频率 |
| validity | 指定feed的有效时间，其中指定了start、end时间(必须保证feed的end结束时间为将 来的某个时间，确定不能是过去的某个时间，否则程序将不能进行调度) |
| cluster name | 指定feed中依赖的cluser的名字 |
| source name | 指定数据源的database实体的名字 |
| source tableName | 指定数据源的表名(在database中指定的是数据库名，注意不要混淆) |

88 **第二篇** **开源实现篇**

(续)

|  |  |
| --- | --- |
| **参** **数** **名** | **定** **义** |
| fields | 指定关系型数据库中需要导出的字段名，默认为空，表示导出所有字段 |
| location data | 指定将源数据导出的目的HDFS的地址 |

对 feed 实体进行提交并调度：

$falcon entity -submitAndschedule -type feed -file examples/entity/filesystem/ mysql\_in\_feed.xml

若提示 schedule/default/mysql-feed(feed)scheduled successfully

submit/falcon/default/Submit successful (feed)mysql-feed,则表示提交并调度成功。 上面这个命令等同于以下两个命令的作用：

$falcon entity -type feed

$falcon entity -type feed

-file mysql in feed.xml -submit

-name mysql-feed -schedule

在 Falcon 的 Web UI(https://linc-1:15443) 中进入 Data Sets目录，即可看到提交成功

的 mysql-feed。

**4.结果分析**

在提交并调度feed 后，在 Oozie 的 Web UI(htp://linc-1:11000/oozie/) 中可以观察到 生成的Job 工作流的状态。

根据图3-8,在没有到任务指定的开始时间之前，工作流的状态为PREP 状态，工作流 调度的频率为5分钟，还可以看到工作流开始执行的时间。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | dob ld | Name | Satus | Us | GTOUP | Frequengy | unt | Started |
| 1 | 0000152-160912152654783-00ze-hdts-C FALCON\_FEE… RUNNING hds 360 MINUTE Sat,17 Sep 201608:53:00 G | | | | | | | |
| 2 | 0000153-160912152654783-oace hdts-C FALCON\_FEE PREP hafs 5 MINUTE Sat.17 Sep 201609:00.00 G | | | | | | | |

图3-8 Oozie Web UI 中工作流的信息

在集群的 Oozie Web UI 中查看 Job 执行状态。

从图3-9可以看出，Job 的执行状态为 SUCCEEDED, 说明工作流调度成功。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| dab ld | Name | Status | User |  |
| Created |
| 0000195-160912152654783-00zie-hdis-W FALCON\_FEED\_IMPORT\_mysqHeed SUCCEEDED hdfs Sat,17 Sep 20161004:34 GMT | | | | |
| 0000194-160912152654783-oozie hdfs-W FALCON\_FEED IMPORT\_mysql-feed SUCCEEDE hdfs Sat,17 Sep 2016100354 GMT | | | | |
| 0000193-160912152654783-oozie-hdfs-W FALCON\_FEED\_IMPORT\_mysql-feed SUCCEEDED hdfs Sat,17 Sep 201610:03:13 GMT | | | | |
| 0000192-160912152654783-o0zie-hdfs-W FALCON\_FEED\_IMPORT\_mysql-feod SUCCEEDED hdfs Sat,17 Sep 20161002.32 GMT | | | | |
| 0000191-160912152654783-oozie-hdfs-W FALCON\_FEED\_IMPORT\_mysqlfeed SUCCEEDED hdfs Sat,17 Sep 20161001:52 GMT | | | | |
| 0000190-160912152654783-oozie-hdfs-W FALCON\_FEED\_IMPORT\_mysql-feed SUCCEEDED hdfs Sat,17 Sep 20161001.12 GMT | | | | |
| 0000189-160912152654783-00zie-hdfs-W FALCON\_FEED\_IMPORT\_mysqHeed SUCCEEDED hdfs Sat,17 Sep 20161000.29 GMT | | | | |
| 0000188-160912152654783-oozie-hdfs-W FALCON\_FEED\_IMPORT\_mysql-feed SUCCEEDED hdfs Sat,17 Sep 201609:59:34 GMT | | | | |
| 0000187-160912152654783-00zie-hdfs-W FALCON FEED\_IMPORT\_mysql-eed SUCCEEDED hdfs Sat,17 Sep 20160958:52 GMT | | | | |
| 0000186-160912152654783-o0zie-hdfs-W FALCON\_FEED\_IMPORT\_mysqHeed SUCCEEDED hdfs Sat,17 Sep 201609:58:07 GMT | | | | |
| 0000185-160912152654783-00zie-hdfs-W FALCON FEED IMPORT mysq-eed SUCCEEDED hdfs Sat,17 Sep 20160957.23 GMT | | | | |

图3-9 工作流调度成功

第3章 大数据治理之 Apache Falcon 89

Oozie 中详细记录了工作流的其他信息，如Job ID,每个工作流在 Oozie 进行调度时都 会分配唯一的ID, 用于与其他的Job 区分；还记录了工作流的提交用户，从图3-9可以看 出，图中的用户为 hdfs, 还可以看到工作流的创建时间。

在 HDFS 中观察导入的数据。

在集群的 HDFS的 Web UI(http://linc-1:50070/) 端上，可以进入/user/hdfs/customers/ 2016/09/17/09目录下，查看按照Job 的频率由 MySQL 导 入 HDFS 中的文件记录。

**3.3.3** **结构化数据库与数据仓库的交互**

本节中，将通过Falcon 实现将数据仓库中的数据自动导入结构化数据库，这里将使用 Hive数据仓库和 MySQL数据库实现这一过程。由于 Falcon暂不支持结构化数据库与数据

仓库的直接交互，因此首先将 Hive 数据仓库中的数据导入HDFS(HiveTable 文 件 ) 中 ， 然

后 将 HDFS 中的数据导入 MySQL 数据库中来实现结构化数据库与数据仓库的交互。

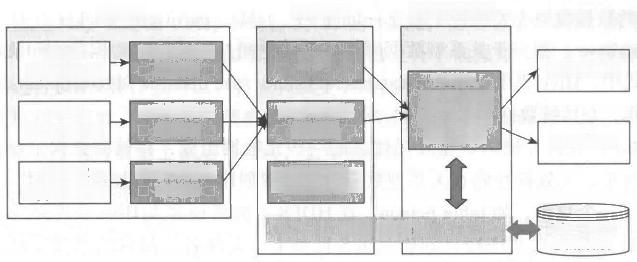
**1.Apache** **Hive** **介绍**

当用户在使用Hadoop 的 API来实现例如WordCount 等简单任务时，需要用户具有丰 富的Java 开发经验，这就将Hadoop潜在地放在了非程序员用户无法触及的位置。事实上， 许多底层任务都是由简单的Job 重复执行完成的，例如数据清洗、数据合并等。幸运的是， Hive 实现了对此类任务的高级抽象，不仅提供了 一 个 SQL 的用户编程模型，而且消除了大 量的代码，这就是引入Hive 的原因，使用户花费相当少的精力就可以完成大量的工作。

Hive作为 Hadoop的数据仓库，是一个面向主题的、集成的、不可更新的、不随时间 变化的数据集合，它主要用于支持企业或组织的决策分析处理。

(1)Hive 架构

Hive 的架构模块划分如图3- 10所示。



Services

Hive Thrift

Client

Hive JDBC

Driver

JobClient

Hive ODBC

Driver

Metastore

图3-10 Hive 架构模块

1)Clients: 远程访问Hive 服务的应用客户端，其中显示了用户可以通过JOBC、 ODBC 、Thrift 服务器来进行服务访问的几个接口。Hive 提 供 了 纯Java 的 JDBC 驱动，连

Thrift

Application

JDBC

Application

ODBC

Application

CLI

Hive Thrift Server

Driver (Compiler/ Optimizer/ Executor)

Metastore Database

Components

Hive Web ul

Namenode

Clients

90 第二篇 开源实现篇

接字符串类似于jdbc:hive://host:port/dbname,ODBC 与 JDBC 原理类似。

2)Services: 需要独立部署的服务，有3种与Hive 进行交互的方式，CLI (命令行 界面)、Hive Thrift Server 、Hive WebUI(Hive的图形用户界面),其中 Hive Thrift Server 是 Hive的一个可选组件，其允许通过指定端口访问Hive, 它作为一个软件框架，其主要致力 于跨语言服务开发， Thrift 允许客户端使用包括Java 、C++ 、Ruby 以及其他语言来通过编 程方式远程访问Hive, 可以使用户不再依赖Hive 环境来访问Hive 数据仓库。目前绝大多 数的数据分析公司都采用了Cloudera 公司的开源项目Hue 作为Hive WebUI。

3)Components:Hive 的组件，其中包含编译器、执行引擎、元数据服务，通过编译 器组件可以对输入进行编译解析，将Hive 中的执行语句解析成XML 文件，之后通过执行 引擎驱动执行内置的、原生的Mapper 和 Reducer 模块来执行Job 任务，通过与Namenode 进行交互来初始化MapReduce 任务。所有的 Hive 客户端都需要一个 Metastore service(元 数据服务),Hive 使用这个服务来存储表的结构信息以及其他元数据信息，例如：表的格式 定义、列的类型、数据源定义、数据划分情况等。 一个 Hive Service 可以包含多个互不相 关的 Metastore 。通常情况下会使用一个关系型数据库中的表来存储这些信息。在默认情况 下 ，Hive 会使用默认的 Derby SQL 服务器，但是这个服务器具有一定的局限性，它只能提 供单进程的存储服务，例如，当使用这个数据库作为元数据服务时，用户不可以执行2个 并发的Hive CLI实例。对于集群来说，可以使用MySQL 或者类似的数据库来解决此类问 题，这里采用的是 MySQL 来作为我们的元数据存储工具。

Hive 建立在 Hadoop 的其他组件之上，主要包含两部分：第1部分是依赖于HDFS 进 行数据存储，第2部分是依赖于MapReduce 完成数据操作。Hive 最适合数据仓库应用程序， 使用该应用程序进行相关的静态数据分析，不需要快速响应给出结果，而且数据本身不会 频繁变化，其可以用来维护海量数据，而且对数据进行挖掘，最后形成数据报告。

(2)Hive 数据模型

Hive 的数据模型主要包含4部分： database 、table 、partition 、bucket。

1)database: 相当于关系型数据库中的命名空间，作用是将数据库应用隔离到不同的 数据库模式中，Hive 提供了create database dbname 、use dbname 以及 drop database dbname 这样的操作，与传统数据库作用类似，这里不详述。

2)table: 表由存储的数据及描述表的一些元数据组成，存储的数据主要存储在分布 式文件系统中，元数据存储在关系型数据库中，刚创建表还没有加载数据时，在HDFS 上 只是创建了一个目录，如 table person,在 HDFS 上的路径为$(Hive 仓库路径}/person, 加 载完数据后会将数据文件拷贝到该HDFS目录下，文件名与加载的数据文件名相同，如 ${Hive 仓库路径}/person/imployer.txt 。Hive 中的 table 分为两种：内部表和外部表。下面 我们对比内部表与外部表的区别。

内部表：

口与传统数据库中的Table 在概念上类似；

第3章大数据治理之Apache Falcon 91

口每一个Table 在 Hive 中都会在 HDFS 中有一个相应的目录存储数据；

口所有 Table 中的数据(不包括External Table) 都保存在特定对应目录中；

口当执行删除表操作时，元数据以及存储在分布式文件系统中的数据都会被删除。 外部表 (External Table):

口在创建表结构的同时，将表的数据存储位置指向已经在 HDFS 中存在的数据；

口它和内部表在元数据的组织上是相同的，而实际数据的存储有较大的差异，并不是 存储在 Hive 指定的仓库路径中；

口外部表只有一个过程，加载数据和创建表同时完成，并不会移动到数据仓库中，只 是与外部数据建立一个链接。当删除一个外部表时，仅删除该链接，外部表中的数 据并不会被删除。

3)Partition: 在 Hive Select 查询中一般会扫描整个表内容，会消耗很多时间做没必要 的工作。有时候只需要扫描表中所关心的一部分数据，因此建表时引入了Partition 的概念。 Partition 可以使用户对特定的数据进行操作，提高数据的查询速度。 一个表可以拥有一个或 者多个分区，每个分区以文件夹的形式单独存在表文件夹的目录下。分区以字段的形式在 表结构中存在，通过describe table 命令可以查看到字段存在，但是该字段不存放实际的数 据内容，仅仅是分区的表示。分区分为两种， 一种是单分区，也就是说在表文件夹目录下 只有一级文件夹目录； 一种是多分区，表文件夹下出现多文件夹嵌套模式。

4)Bucket: 对于每一个表或者分区， Hive 可以进一步将数据组织成桶，也就是说桶是 更为细粒度的数据范围划分。Hive 针对某一列进行桶的组织划分。Hive 采用对列值哈希， 然后除以桶的个数求余的方式决定该条记录存放在哪个桶中。

把数据再次细分组织成桶的必要性如下：

口获得更高的查询处理效率。桶为表加上了额外的结构， Hive 在处理有些查询时能利 用这个结构。具体而言，连接两个在(包含连接列的)相同列上划分了桶的表，可 以使用Map 端连接 (Map-side join) 高效地实现，比如 JOIN 操作。对于JOIN 操作 两个表有一个相同的列，如果对这两个表都进行了桶操作。那么对保存相同列值的 桶进行 JOIN 操作就可以，将大大减少 JOIN 的数据量。

口使取样更高效。在处理大规模数据集时，在开发和修改查询的阶段，如果能在数据 集的一小部分数据上试运行查询，会带来很多方便。

**2.Hive** **的安装与使用**

由上小节我们已经了解了Hive 的结构与工作原理，现在来看看Hive 的安装与使用。

在前面已经对Hive 的原理、配置和使用有详细的说明，下面我们来讲讲Ranger-Hive

的工作原理及配置使用。

(1)Hive 下载与解压

在linc-1 节点上，使用hdfs 用户，从<http://hive.apache.org/downloads.html> 下载 Hive版 本号2.1.1版本(Ranger 只支持Hive 版本2.1.0及以上版本),然后解压到/var/local/hadoop

92 第二篇 开源实现篇

目录下。

$cd ~

$wget <http://mirror.bit.edu.cn/apache/hive/hive-2.1.0/apache-hive-2.1.0-bin.tar.gz> $tar zxvf apache-hive-2.1.0-bin.tar.gz

$mv apache-hive-2.1.0-bin /var/local/hadoop/hive-2.1.0

( 2 ) 修 改 Hive 配置文件 hive-env.sh

在 linc-1 节点上，使用hdfs 用户，在 hive-env.sh 中添加 Hadoop 路径以及 Hive 配置文 件的路径。

$cd /var/local/hadoop/hive-2.1.0/conf

$cp hive-env.sh.template hive-env.sh

s vim hive-env.sh

#添加以下内容

export HADOOP\_HOME=/var/local/hadoop/hadoop-2.7.2

export Hive\_CoNF\_DIR=/var/local/hadoop/hive-2.1.0/conf

( 3 ) 修 改 Hive 配置文件 hive-site.xml

在修改配置文件前，由于Hive 的文件存储在 HDFS 分布式系统上，所以不需要在本地 新建用来存放数据仓库信息的文件夹，只需要配置hive-site.xml, 然后初始化Hive 时，会 自动在分布式系统中新建文件夹。

$cd /var/local/hadoop/hive-2.1.0/conf

$cp hive-default.xml.template hive-site.xml

$vim hive-site.xml

#删除之前<configuration> 标签内全部内容，并添加以下内容：

<property>

<name>hive.metastore.warehouse.dir</name>

<value>/var/local/hadoop/hive-2.1.0/warehouse</value>

</property>

上述这个 property 属性指定了数据仓库 Hive 在 HDFS 上的存储目录，读者可根据自己 的需求自定义存储路径。

<property>

<name>hive.exec.scratchdir</name>

<value>/var/local/hadoop/hive-2.1.0/tmp</value>

</property>

上述这个 property 属性指定了Hive 在执行任务过程中产生的临时文件的存储地址，用 于存储不同Map/Reduce 阶段的执行计划和这些阶段的中间输出结果。

<property>

<name>hive.querylog.location</name>

<value>/var/local/hadoop/hive-2.1.0/logs</value>

</property>

这个 property 属性指定了Hive 实时查询日志所在的目录，如果该值为空，将不创建实 时的查询日志。

第3章 大数据治理之Apache Falcon  93

<property>

<name>hive.server2.thrift.port</name>

<value>10000</value>

</property>

这个property 属性指定了HiveTep的监听端口。默认这个端口的值为10000。

<property>

<name>hive.server2.thrift.bind.host</name>

<value>linc-1</value>

</property>

这个property 属性指定了上述 Tcp端口绑定的主机，默认为LocalHost 。如果需要在远 程访问该主机，需要将值改为IP 地址。

<property>

<name>hive.server2.enable.doAs</name>

<value>true</value>

</property>

默认情况下， HiveServer2 以提交查询的用户执行查询 (true), 如果值设置为 false, 查

询将以运行 HiveServer2 进程的用户运行。

<property>

<name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:mysql://linc-1:3306/hive?createDatabaseIfNotExist=true</value> </property>

这个 property 属性指定了Hive 将元数据存储在结构化数据库MySQL 中，用于在 MySQL 数据库中建立 Hive 的元数据库。

<property>

<name>javax.jdo.option.ConnectionDriverName</name>

<value>com.mysql.jdbc.Driver</value>

</property

这个property 属性指定了 Hive 连接结构化数据库 MySQL 使用的驱动包，注意这里需 要填写类的全名。

<property>

<name>javax.jdo.option.ConnectionUserName</name>

<value>root</value>

</property>

<property>

<name>javax.jdo.option.ConnectionPassword</name>

<!--这里是数据库密码-->

<value>MyNewPass4!</value>

</property>

上述这两个 property 属性分别指定了连接结构化数据库 MySQL时的用户名以及用户 密码。

94 第二篇 开源实现篇

( 4 ) 添 加 jar 包

**添加** **MySQL 数据库连接jar包到** **Hive 路径下，根据前几小节中所用的** **MySQL连接** ja**r 包版本为5.1.38,并用Hive 中** **jline-2.12.jar 替换** **Hadoop 中的jar, 具体操作如下：**

$cp /usr/share/java/mysql-connector-java-5.1.38-bin.jar /var/local/hadoop/hive-2.1.0/

lib

$rm /var/local/hadoop/hadoop-2.7.2/share/hadoop/yarn/lib/jline-0.9.94.jar

$cp /var/local/hadoop/hive-2.1.0/lib/jline-2.12.jar /var/local/hadoop/hadoop-2.7.2/

share/hadoop/yarn/lib/

( 5 ) 配 置Hive 的环境变量

在 linc-1 上，使用hdfs 用户修改/etc/profile 文件，添加 Hive 环境变量，并刷新环境变量。

$sudo vim /etc/profile

#这里要求输入密码

#在文件最后一行添加以下内容

export Hive\_HOME=/var/local/hadoop/hive-2.1.0

export PATH=SPATH:SHive\_HOME/bin

刷/e/ile

(6)初始化数据库

将 Hive 元数据库进行初始化，如果不先初始化，启动 Hive 命令行时，将会产生 Hive metastore database is not initialized 的错误。解决方法如下：

#初始化Hive元数据库

s schematool -dbType mysql -initSchema

#启动Hive, 注意启动前一定要先启动start-dfs.sh

$hive

清 在启动 Hive 过程中，若出现 Hive metastore database is not initialized 的错误，说明

没有初始化 Hive 元数据库。

( 7 ) 测 试 Hive 启动后能够在 MySQL 中生成相应的元数据库

首先查看未创建 Hive 表 时 ，MySQL 中元数据库的记录为空。

#进入数据库

$mysql -uroot -pMyNewPass4!

mysql >show databases;

mysql >use hive;

mysql>show tables;

mysql>select \*from TBLS;

若查询TBLS 表没有任何反馈，则可能是 Hive 安装配置有问题。若查询到TBLS 表的 内容是空，或者有其他内容，则说明 Hive 安装配置正确。

( 8 ) 测 试 Hive 创建表格

在 Hive 命令行中创建一个表，其中包括姓名、性别和年龄3个字段。

第3章 大数据治理之Apache Falcon 95

hive>create table student(name string,sex string,age int);

$hive

hive>show tables;

hive>desc student;

创建表后，按照步骤(7)中的命令，查看元数据库，发现有一条student 的元数据

记录。

**3.将数据仓库表中数据导出到分布式文件系统**

( 1 ) 先 决 条 件

确 保 Hadoop 、Falcon 、Sqoop 以 及Oozie 安装成功，已经进行NTP 时间校正。因为时 间如果不同步可能会影响 Falcon 的 job 作业实施。

( 2 ) 数 据 准 备

在数据准备阶段，读者需要自行建立 Hive 数据源，作为导入分布式文件系统的数据源。

#进入Hive 数据仓库的命令行模式

s hive

#创建数据库 test

s create database test;

#进入数据库 test

$use test;

#创建源数据person\_in 表，并且使用ds 字段进行分区

$create table person\_in(idint,stockname string,value int,cash int)partitioned by(ds string);

# 给person\_in 表添加'2016-09-19-14-00'字段

s alter table person\_in add partition(ds='2016-09-19-14-00');

s \_261='2016-09- 19- 14-05');

#给相应的字段中添加值

S

s

S

s

insert into 14-00');

insert into

14-00');

insert

into

14-05');

into

insert

14-05');

person\_in

person\_in

person\_in

person\_in

PARTITION(ds)values(10001,'ZGYH',120000,80000,'2016-09-19- PARTITION(ds)values(10002,'ZSYH',1204400,32000,'2016-09-19- PARTITION(ds)values(10003,'JSYH',21100,53200,'2016-09-19-

PARTITION(ds)values(10004,'NYYH',324100,7500,'2016-09-19-

读者可以对插入数据进行验证，在Hive 中执行语句：

s select \*from person\_in;

可以观察到数据插入成功，如图3- 11所示。

以上都属于环境准备阶段，在接下来的叙述里

进行结构化数据库与数据仓库交互的第1部分。 ( 3 ) 将 Hive 表中的数据导入HDFS

1)创建并提交 Cluster 实体。

Aves setect·tfon person in;

0K

|  |  |  |  |
| --- | --- | --- | --- |
| 18801  10682  10883  10084 | ZGYH  Z5YH  15YH  NYYH | 12800880800  120446032860  21108 53280  3241007508 | 2016-09-19-14-00  2016-09-19-14-88 2016.00-19.14-84  2816-89-19 -14-05 |

Jime taken:2.156 seconds,Fetched:4 rew(s)

图3-11 Hive 表中源数据展示

**96** **第二篇** **开源实现篇**

**在** **Falcon 安装目录下新建** **hive\_mysql\_cluster.xml 文件：**

s vim examples/entity/filesystem/hive\_mysql\_cluster.xml

**文件内容如下所示：**

<?xml version="1.0"?>

<cluster colo="local"description="Standalone cluster"name="hive-mysql-cluster"

xmlns="uri:falcon:cluster:0.1">

<interfaces>

<interface type="readonly"endpoint="hdfs://linc-1:9000"version="2.6.0"/> <interface type="write"endpoint="hdfs://linc-1:9000"version="2.6.0"/>

<interface type="execute"endpoint="linc-1:8032"version="2.6.0"/>

<interface type="workflow"endpoint="http://linc-1:11000/oozie/"version="4.2.0"/>

<inte="messaging"endpoint="tcp://linc-1:61616?daemon=true" version=

<interface type="registry"endpoint="thrift://linc-1:9083" version="1.2.1"/>

</interfaces>

上述 interface 属性主要指定了Hive 进行任务调度时涉及的服务的端口，例如 Hadoop 集群的读写接口等。

<locations>

<location name="staging"path="/projects/falcon/hcat-staging"/><!--mandatory-->

<location name="temp"path="/projects/falcon/hcat-tmp"/><!--optional-->

<location name="working"path="/projects/falcon/hcat-working"/><!--optional--> </locations>

上述 location 属性主要指定了任务在执行过程中使用到的文件目录。注意，只有 staging 是必须要配置的属性，另外两个属性 temp、working可以不进行配置。

<properties>

</properties>

</cluster>

cluster实体中定义了Falcon执行任务过程中将会用到的各种接口。对 cluster 实体进行 提交：

s bin/falcon entity -submit -type cluster -file examples/entity/filesystem/hive\_

hivetable\_cluster.xml

操作说明：

若提示 falcon/default/Submit successful (cluster)hive-hivetable-cluster,则表示提交成功。 2)创建并提交 feed 实体。

在 Falcon安装目录下新建 hive-hivetable-out.xml文件：

S vim examples/entity/filesystem/hive-hivetable-out.xml

文件内容如下所示：

<?xml version="1.0"encoding="UTF-8"?>

**第3章** 大数据治理之Apache Falcon 97

<feed description="output"name="hive-hivetable-out"xmlns="uri:falcon:feed:0.1">

<frequency>minutes(5)</frequency>

z-Cm-oefzfn"einutes(5)"/>

me="hive-mysql-cluster">

<validity start="2016-09-17T12:00Z"end="2016-10-01T00:00Z"/>

<retention limit="minutes(5)"action="delete"/>

</cluster>

</clusters>

e="data"path="/var/local/hadoop/hivetable/S{YEAR}/S(MONTH)/S{DAY}"/>

<location type="stats"path="/none"/>

<location type="meta"path="/none"/>

="hdfs"group="hadoop"permission="0x644"/>

<schema location="/schema/log/log.format.csv"provider="csv"/>

</feed>

feed实体中定义了在Falcon执行工作流的过程中将使用到的数据集的各种属性，例如 数据集的位置、数据集使用的频率和数据集的有效期限等。下面的 feed 指定了 Hive 中的数 据源将导入到的目的地址。

s bin/falcon entity -submit -type feed -file examples/entity/filesystem/hive-

hivetable-out.xml

操作说明：

若提示 falcon/default/Submit successful (feed)hive-hivetable-out, 则表示提交成功。 3)创建并提交 feed 实体(指定Hive数据表)。

在 Falcon安装目录下新建hive-hivetable-in.xml 文件：

s vim examples/entity/filesystem/hive-hivetable-in.xml

**文件内容如下所示：**

<feed description="input"name="hive-hivetable-out"xmlns="uri:falcon:feed:0.1"> <frequency>minutes(5)</frequency>

<?xml version="1.0"encoding="UTF-8"?>

<timezone>UTC</timezone>

<late-arrival cut-off="minutes(5)"/>

<clusters>

<cluster name="hive-mysql-cluster">

itoyn sta:0-01T00:00Z"/>

</cluster>

satalog:test:person#ds=${YEAR}-S{MONTH}- §{DAY}-S{HOUR)-S{MINUTE}"/>

<ACL owner="hdfs"group="hadoop"permission="0x644"/>

<schema location="/schema/out/out.format.csv"provider="csv"/>

</feed>

98 第二篇 开源实现篇

feed 实体中定义了在 Falcon 执行工作流的过程中将使用到的数据集的各种属性，例如 数据集的位置、数据集使用的频率和数据集的有效期限等。这个 feed 实体指向了 Hive数据 源，作为需要进行导出的数据。

土

慧

<table uri="catalog:test:person#ds=${YEAR}-${MONTH}-S{DAY}-${HOUR}- ${MINUTE}"/>: 这里的test 是指数据库名， person 是指需要导出的数据库中的表 名，ds 是person 表中的分区字段。对于后面的分区的值，用户可根据自己的需要进 行调整。

对feed实体进行提交：

$bin/falcon entity -submit -type feed -file examples/entity/filesystem/hive-

hivetable-in.xml

若提示 falcon/default/Submit successful (feed)hive-hivetable-in, 则表示提交成功。 4)提交 HQL 脚本。

在 %FALCON\_HOME%目录下新建 hive-hivetable.xml 文件：

$vim examples/entity/filesystem/hive-hivetable.hql

文件内容如下所示：

s insert overwrite directory 'Sloutpart}'row format delimited fields terminated by ','select id,stockname,value,cash from $(falcon\_inparts\_database}.${falcon\_ inparts\_table};

将 HQL 脚本提交到 HDFS:

$hdfs dfs -put examples/entity/filesystem/hive-hivetable.hql /app/hive/

HQL脚本中主要指定了 Falcon 工作流中的数据处理逻辑。在 process 实体中可以指定使 用Hive引擎来执行HQL文件，其中的${outpart} 、${falcon\_inparts\_database}和${falcon\_ inparts\_table}为参数形式，将从 Oozie中获取变量的值在这里进行使用。这个HQL语句的 意义在于将表中的数据中的特定字段抽取出来保存在HDFS中。这里的 HDFS的目录可以 随意指定，但必须与process中的目录保持一致。

5)提交 process实体。

在 Falcon安装目录下新建 hive\_hivetable\_process.xml 文件：

s vim examples/entity/filesystem/hive\_hivetable\_process.xml

文件内容如下所示：

<?xml version="1.0"encoding="UTF-8"?>

<process name="hive-hivetable-process"xmlns="uri:falcon:process:0.1"> <clusters>

<cluster name="hive-mysql-cluster">

<validity start="2016-09-19T14:00Z"end="2016-09-19T14:10Z"/>

第3章 大数据治理之Apache Falcon  99

</cluster>

</clusters>

<parallel>1</parallel>

<order>FIFO</order>

<frequency>minutes(5)</frequency>

<timezone>UTC</timezone>

上述属性包含了process在执行任务时的执行顺序、执行频率等属性，以及使用的时区 信息。

<inputs>

<input name="inparts"feed="hive-hivetable-in"start="now(0,0)"end="now(0,0)"/>

</inputs>

<outputs>

<output name="outpart"feed="hive-hivetable-out"instance="now(0,0)"/> </outputs>

上述 input 、output 属性通过赋值数据集的名字来指定对应输入、输出数据集的位置。

<workflow engine="hive"path="/app/hive/hive-hivetable.hql"/></process>

上述属性指定了 Hive的 HQL的脚本位置，定义了Job的执行业务规则。

这个 process 定义了使用 Hive引擎的详细配置信息，工作流的执行逻辑将由路径中的 HQL进行详细指定。其他还指定了这个process依赖于hive-mysql-cluster 这个 cluster 实 体，并且将hive-hivetable-in这个数据集作为这个 process的输入， hive-hivetable-out作为 process 的输出，以及 process的执行的频率。

process 的时间必须在所依赖的 feed 数据集的有效时间之内，因为process 实体会对 feed 实体存在依赖。

对 process 实体进行提交并调度：

s falcon entity -submitAndSchedule -type process -file examples/entity/filesystem/ hivetable\_hive\_process.xml

若提示 schedule/default/hive-hivetable-process(process)scheduled successfully,submit/ falcon/default/Submit successful(process)hive-hivetable-process, 则表示提交成功。

从集群在 Falcon的 Web UI(https://linc-1:15443) 进入 Processes 目录，即可看到提交 成功的 hive-hivetable-process。可以看出里面的内容与我们所提交的hive-hivetable-process 的内容相符。

(4)结果分析

可以在 Oozie的 Web UI(http://linc-1:11000/oozie/) 查看 Job 执行的状态。

如图3-12所示，可以在 Status 中观察到 Job执行成功，也可以观察到这个Job 的名称、 已经执行的状态、Job 创建时间、Job 开始时间，并可知运行这个Job 的用户为hdfs 。Oozie 为用户提供了简明的界面，方便用户观察任务的执行。

100 **第二篇** **开源实现篇**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Stalus | User | Created | Started |
| FALCON\_PROCESS\_DEFAULT\_hive-hivetable-procass SUCCEEDED hdfs Mon,19 Sep 201609.06:12 GMon,19 Sep 201609:06:12 G | | | | |

图3-12 Oozie Web UI中工作流执行成功的记录

如图3-13所示，在集群的HDFS Web UI(http://linc-1:50070/) 端进入/var/local/ hadoop/hivetable/, 可以查看由MySQL导入HDFS中的文件记录。文件记录的属性包括了 文件的权限、文件所属的用户和组、文件的大小、文件的副本数等信息，方便用户对产生 的结果进行观察管理。

|  |  |
| --- | --- |
| /var/local/hadoop/hivetable/2016/09/19 | Go! |
| **Permission** **Omer** **Group** **Size** **Replication** **Block** **Size** **Nam**e | |
| rwxr-xr-x hdfs supergroup 95 B 3 128 MB 000000-0 | |

图3-13 导入的文件记录

**4.将分布式文件系统中的数据导入结构化数据库中**

(1)数据准备

在数据准备阶段，读者需要自行建立 MySQL 数据库，作为导出数据的目的地。

#登录MySQL的命令行模式

mysql -uroot -pMyNewPass4!

#创建数据库

create database test;

#进入test 数据库

use test;

#创建test 数据库，并且将id 作为主键

CREATE TABLE person\_out(idint,stockname varchar(20),value int,cash int,primary key(id));

(2)将 HDFS的数据导入MySQL数据库中

1)创建并提交(源)cluster 实体。

在 %FALCON\_HOME%目录下新建hivetable\_mysql\_cluster.xm1 文件：

$vim examples/entity/filesystem/hivetable\_mysql\_cluster.xml

文件内容如下所示：

rsioal"description="Standalone cluster"name="hivetable-mysql-

cluster"xmlns ="uri:falcon:cluster:0.1">

<interfaces>

<interface type="readonly"endpoint="hdfs://linc-1:9000"version="2.6.0"/>

<interface type="write"endpoint="hdfs://linc-1:9000"version="2.6.0"/>

<interface type="execute"endpoint="linc-1:8032"version="2.6.0"/>

<interface type="workflow"endpoint="http://linc-1:11000/oozie/"version="4.2.0"/> <interface type="messaging"endpoint="tcp://linc-1:61616?daemon=true" version=

"5.4.3"/>

**第3章大数据治理之** **Apache Falcon 101**

<interface type="registry"endpoint="thrift://linc-1:9083" version="1.2.1"/> </interfaces>

<locations>

<location name="staging"path="/projects/falcon/hcat-staging"/><!--mandatory-->

<location name="temp"path="/projects/falcon/hcat-tmp"/><!--optional-->

<location name="working"path="/projects/falcon/hcat-working"/><!--optional-->

</locations>

<properties></properties>

</cluster>

cluster实体中定义了 Falcon执行任务过程中将会用到的各种接口。然后对 cluster 实体 进行提交：

s falcon entity -submit -type cluster -file examples/entity/filesystem/hivetable

mysql\_cluster.xml

若提示falcon/default/Submit successful (cluster)hivetable-mysql-cluster, 则表示提交成功。 2)创建并提交(目的)cluster 实体。

在 Falcon安装目录下新建 hivetable\_mysql\_cluster2.xml 文件：

s vim examples/entity/filesystem/hivetable\_mysql\_cluster2.xml

文件内容如下所示：

<?xml version="1.0"?>

<cluster colo="local"description="Standalone cluster"name="hivetable-mysql- cluster2"xmlns="uri:falcon:cluster:0.1">

<interfaces>

<interface type="readonly"endpoint="hdfs://linc-1:9000"version="2.6.0"/>

<interface type="write"endpoint="hdfs://linc-1:9000"version="2.6.0"/>

<interface type="execute"endpoint="linc-1:8032"version="2.6.0"/>

<interface type="workflow"endpoint="http://linc-1:11000/oozie/"version="4.2.0"/>

<intfc"pe="messaging" endpoint="tcp://linc-1:61616?daemon=true" version=

<interface type="registry"endpoint="thrift://linc-1:9083" version="1.2.1"/>

</interfaces>

<locations>

<location name="staging"path="/projects/falcon/hcat-staging"/><!--mandatory--> <location name="temp"path="/projects/falcon/hcat-tmp"/><!--optional-->

<location name="working"path="/projects/falcon/hcat-working"/><!--optional-->

</locations>

<properties></properties>

</cluster>

cluster 实体中定义了 Falcon执行任务过程中将会用到的各种接口。

对cluster 实体进行提交：

s falcon entity -submit -type cluster -file examples/entity/filesystem/hivetable mysql\_cluster2.xml

若提示 falcon/default/Submit successful (cluster)hivetable-mysql-cluster2, 表示提交

102 **第二篇** **开源实现篇**

成功。

这个场景指定两个 cluster, 以满足在 Feed 实体中处理数据的需要。

3)创建并提交 database 实体。

在 Falcon 安装目录下新建 mysql\_database.xml 文件：

s vim examples/entity/filesystem/mysql\_database.xml

文件内容如下所示：

s=.0""-dcion="MySQL database on **west** coast"type=

"mysql"name="mysql-db"xmlns="uri:falcon:datasource:0.1">

<interfaces>

e\* - 1/market">

<credential type="password-text">

<userName>root</userName>

<passwordText>admin</passwordText>

</credential>

</interfaces>

上述 interface 定义了需要连接的 MySQL 的用户名和密码，这个接口用于从MySQL 中 读取数据。

i!"itrd - 1/market">

<credential type="password-text">

<userName>root</userName>

<passwordText>admin</passwordText>

上述 interface 定义了需要连接的 MySQL 的用户名和密码，这个接口用于在 MySQL 中 写入数据。

</credential>

</interface>

taelf=iradl-t>

<userName>root</userName>

<passwordText>admin</passwordText>

</credential>

</interfaces>

<driver>

<clazz>com.mysql.jdbc.Driver</clazz>

<jar>/user/hdfs/share/lib/lib\_20160907192204/sqoop/mysql-connector-java-5.1.39-bin </jar>

</driver>

上述 driver 属性定义了进行 MySQL 连接时需要使用到的 jar 驱动包地址。

</datasource>

第3章大数据治理之Apache Falcon 103

关于datasource 的具体说明，请参照第3章中的3.3.2节关于 DataSource 的叙述。数据 源实体进行提交：

s falcon entity -submit -type datasource -file examples/entity/hivetable\_mysql

database.xml

若提示 falcon/default/Submit successful (datasource)hivetable-mysql-db,则表示提交成功。 可用如下命令进行查看：

$falcon entity -type datasource -list

4)创建并提交 feed 实体。

在 Falcon 安装目录下新建 hivetable\_mysql\_feed 文件：

$vim examples/entity/filesystem/hivetable\_mysql\_feed

文件内容如下所示：



feed:0.1">

<tags>externalSystem=USWestEmailServers,classification=secure</tags>

u-iyv>adlat(-uinuctye>s(5)"/>

<clusters>

<cluster name="hive-mysql-cluster" type="source">

nsi:09-20T14:00Z"/>

</cluster>

<cluster name="hivetable-mysql-cluster2">

nsi:09-20T14:00Z"/>

m/>ysql-db"tableName="person\_out">

<fields>

<includes>

<field>id</field>

<field>stockname</field>

<field>value</field>

<field>cash</field>

</includes>

</fields>

</target>

</export>

上述 export 字段定义了需要将数据导出的目的位置、在导出结果中相应的对应字段以 及导出数据在 MySQL中的插入方式。fields 可以不进行指定，默认为全部字段。

</cluster>

</clusters>

**104** **第二篇** **开源实现篇**

<locations>

<location type="data"path="/var/local/hadoop/hivetable/S{YEAR}/S{MONTH}/${DAY)"/>

上述 location 字段定义了源数据所在的 HDFS 中的地址。

<location type="stats"path="/none"/>

<location type="meta"path="/none"/>

</locations>

<ACL owner="hdfs"group="hadoop"permission="0755"/>

<schema location="/none"provider="none"/>

</feed>

feed 实体中指定了两个 cluster实体， 一个 cluster指定了数据源， 一个 cluster指定了数 据目的，在location中指定的是HDFS上的源数据，将源数据中的数据导入MySQL中指定 的相应字段中。对 feed 实体进行提交：

$falcon entity -submitAndSchedule -type feed -file examples/entity/filesystem/

mysql\_hive\_feed.xml

若提示 schedule/default/hivetable-mysql-feed(feed)scheduled successfully submit/falcon/ default/Submit successful (feed)hivetable-mysql-feed, 则表示提交成功。

(3)结果分析

1)在集群在 Oozie 中 的 Web UI(http://linc-1:11000/oozie/) 查 看 Job 执行状态。根据图 3-14可以看到任务执行成功。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Status | R. | User | Created | Started |
| FALCON\_FEE SUCCEE… 0 hdfs Mon,19 Sep 201615:53:13 G… Mon,19 Sep 201615:53:13 G… | | | | | |
| FALCON\_FEE SUCCEE.. 0 hdfs Mon,19 Sep 201615:53:13 G… Mon,19 Sep 201615:53:13 G… | | | | | |
| FALCON\_FEE… SUCCEE. 0 hdfs Mon,19 Sep 201615:53:07 G Mon,19 Sep 201615:53:07 G… | | | | | |

图3-14 任务执行成功

2)进入MySQL查看数据：

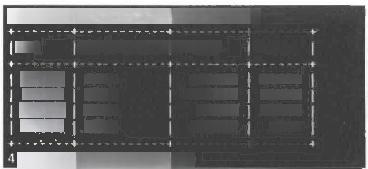
$mysql -uroot -padmin

$use test;

s select \*from person\_out;

从图3-15可以看出，Hive数据仓库中的数 据已经成功导入MySQL 数据中。

**3.3.4** **跨集群数据传输**



mysql>selectfrom person\_out;

id stockname I value cash

10801

16002

10803

10004

rows in set(6.02 sec)

120000

1204400

21100

324100

ZGYH

ZSYH

JSYH

NYYH

80000

32006

53200

7500

图3-15 数据仓库中的数据导入MySQL 中

本场景涉及跨集群数据传输，所以需要有两个 Hadoop 集群，请保证两者各项组件使用 同一版本，并且都装有 Hadoop、Falcon、Oozie以及其他共用基础组件。

**1.前期工作**

确保 Hadoop 、Falcon 以 及 Oozie 安装成功，已经进行NTP 时间校正。因为时间如果不

第3章 大数据治理之 Apache Falcon 105

注 意

同步可能会影响 Hadoop 的 Job 作业调度。

1)修改集群2的Falcon 配置文件，修改其运行的IP 以及端口。

s cd /var/local/hadoop/falcon-0.9

s vim conf/client.properties

2)修改 client.properties 文件中 falcon.url 的值为集群1的 Falcon IP 与端口。

falcon.url=https://linc-1:15443

**2.确保** **DistCp 的可用性**

因为 Falcon进行跨集群传输的时候，需要依赖DistCp 服务，所以读者先要确保其能够 正常的使用。在linc-1 上，使用hdfs 用户在任意目录下创建 distcptest 目录，并且上传任意 一文件，并更改其目录权限为777。操作命令如下：

$hadoop fs -mkdir /distcptest

s hadoop fs -put 你要上传的任意文件地址/distcptest

$hadoop fs -chmod 777 /distcptest

$hadoop distcp hdfs:// <192.168.91.100>:9000/distcptest hdfs:// <192.168.91.200>:9000/

上述命令的两个IP 分别为集群1主节点IP 和集群2主节点IP, 读者需根据自己实 际IP 进行更改。后面内容出现具体IP, 也同样处理。

9

执行过程中会进行一次 MapReduce 作业，成功后情形如图3-16所示。



16000i 18:58:44 INF⁰tools.DistCp:DistCp joh-id:job 14727239425430014 16.09 Q118:58:44 INFO mspreduce.Job:Running Jcb:Job 1472723242543.0014

d1690:l53 INF⁰mapredice.Job:Job 36b 14727232425130014 runniag in uber m

16/090118:58:53 INF⁰mapreduce.Job:map Oh reduce Oh

1609/0118:59:19 INFO mapreduce.Job:map 50%reduce 0%

16.090118:59:20 INF⁰mapreduce.Job:map 100%reduce Ok

16.090118:59:22 INF⁰mapreduce,job:Job job\_ 1472723242543.0014 completed successt

图3-16 DistCp 执行过程

出现如上成功提示，说明执行成功，证明 DistCp 可用。

**3.提交source-cluster.xml**

cluster 中相关参数的定义如表3-6所示。

**表3-6** **cluster 部分参数定义**

|  |  |
| --- | --- |
| **参** **数** **名** | **定** **义** |
| type | interface参数中的type主要指定接口的类型。可以指定的值有：read(读接口)、write(写接口)、 excute(执行)、workflow(工作流)、messaging(消息)  credential参数中的type主要指定用户密码的类型。可以指定的值有：password-text(文本类 型)、password-file(文件类型)、password-alias |
| endpoint | 指定集群内相关功能的地址及端口 |
| name | location中的name用来指定工作相关路径位置 |

106 第二篇 开源实现篇

 以下配置文件中的100、200分别代表集群1的主节点IP 及集群2的主节点IP。

按照配置文件，由上而下地分别指定了目标数据集群内的如下参数：

1)读取 hdfs 的地址以及端口。

2)写入hdfs的地址以及端口。

3)执行监控的Resource Manager 管理地址以及端口。

4)工作流处理工具Oozie的地址。

5)指定消息处理工具。

6)在hdfs上指定任务工作台路径。

7)在hdfs上指定任务临时文件路径。

8)在 hdfs上指定任务工作依赖路径，里面会自动导入依赖的 lib包。

在集群1的 linc-1上，使用hdfs用户，在/var/local/hadoop/falcon-0.9/目录下执行以下 命令：

S vim examples/entity/filesystem/source-cluster.xml

文件内容修改为如下内容：

<?xml version="1.0"?>

<cluster colo="sourcehdfs"description="Standalone cluster"name="sourcehdfs"xmlns= "uri:falcon:cluster:0.1">

<interfaces>

<interface type="readonly"endpoint="hdfs:// <192.168.91.100>:9000"version="2.6.0"/> <interface type="write"endpoint="hdfs:// <192.168.91.100>:9000"version="2.6.0"/>

<interface type="execute"endpoint="hdfs:// <192.168.91.100>:8032"version="2.6.0"/>

<intt/y>pe="workflow"endpoint="http:// <192.168.91.100>:11000/oozie/"version=

<interface type="messaging"endpoint="tcp:// <192.168.91.100>:61616?daemon=true"

version="5.4.3"/>

</interfaces>

<locations>

<location name="staging"path="/apps/falcon/staging"/><!--mandatory--> <location name="temp"path="/apps/falcon/tmp"/><!--optional-->

<location name="working"path="/apps/falcon/working"/><!--optional-->

</locations>

</cluster>

之后将上述修改的 source-cluster 文件上传至 Falcon中。

$bin/falcon entity -submit -type cluster -file examples/entity/filesystem/source- cluster.xml

上传cluster后可在 Web UI内的clusters 分目录下，上传配置好的cluster 文件，内容与 source-cluster.xml 文件内容类似，并且状态为 submitted。

点开 Web UI的 Clusters 条目下刚上传好的文件后，可见文档内容与上文我们所写入文 件内容一致，表示上传成功。

第3章大数据治理之Apache Falcon 107

**4.提交target-cluster.xml**

cluster 中相关参数的功能如表3-6所示。

在集群1的 linc-1上，使用hdfs 用户，在任意目录下执行以下命令：

S vim examples/entity/filesystem/target -cluster.xml

**文件内容修改为如下内容：**

trethdfs"colo="targethdfs"description="Standalone cluster"

xmlns="uri:falcon:cluster:0.1">

<interfaces>

<interface type="readonly"endpoint="hdfs:// <192.168.91.200>:9000"version="2.6.0"/> <interface type="write"endpoint="hdfs:// <192.168.91.200>:9000"version="2.6.0"/>

<interface type="execute"endpoint=" <192.168.91.200>:8032"version="2.6.0"/>

<intfc"pe="workflow"endpoint="http:// <192.168.91.200>:11000/oozie/"version=

<interface type="messaging"endpoint="tcp://linc-1:61616?daemon=true"version=

"5.4.3"/>

</interfaces>

<locations>

<location name="staging"path="/apps/falcon/staging"/><!--mandatory-->

<location name="temp"path="/apps/falcon/tmp"/><!--optional-->

<location name="working"path="/apps/falcon/working"/><!--optional--> </locations>

</cluster>

**之后上传** **target-cluster 至** **Falcon 中。**

$bin/falcon entity -submit -type cluster -file examples/entity/filesystem/target-cluster.xml

上传 cluster后可在 Web UI内的clusters 分目录下上传配置好的 cluster文件，内容与 target-cluster.xml文件内内容类似，并且状态为 submitted。

点开 Web UI的 Clusters 条目下刚上传好的文件后，可见文档内容与上文我们所写入文 件内容一致，表示上传成功。

**5.新建source-target-feed.xml 及提交**

feed 中相关参数的定义如表3-7所示。

**表3-7** **feed 部分参数定义**

|  |  |
| --- | --- |
| **参** **数** **名** | **定** **义** |
| frequency | 代表文件检查时间，每相隔frequency时间会重新检测HDFS内的文件，如果这个时间段内有文 件变化，如新的文件加入，那么下一个frequency内会将新的文件也传输过去 |
| clusters | clusters内分别写入上面依赖的两个cluster的相关配置，cluster内需包含如下属性，cluster name,类型(类型来区分source以及target),validity内指定任务开始时间以及结束时间，  retention指定上述设定的结束时间后数据的保存时间，以及对应处理的策略(这里保存2小时，对 应策略为删除) |

第二篇 开源实现篇

(续)

|  |  |
| --- | --- |
| **参** **数** **名** | **定** **义** |
| **locations** | **类型为target的cluster内locations指定的为目标集群内hdfs上的目标存放位置。cluster外部的** **locations指定的源数据集群内hdfs上的源数据存放位置** |

其中重要参数如下：

1)frequency代表文件检查时间，每相隔 frequency 时间会重新检测 hdfs 内的文件，如 果这个时间段内有文件变化，如有新的文件加入，那么下一个 frequency 内会将新的文件也 传输过去。

2)clusters 内分别写入上面依赖的两个 cluster 的相关配置， cluster 内需包含如下属性， cluster name,类型(类型来区分 source 以及target),validity 内指定任务开始时间以及结束 时间，retention 指定上述设定的结束时间后数据的保存时间，以及对应处理的策略(这里保 存2小时，对应策略为删除)。

3)类型为target的 cluster内 locations 指定的为目标集群内hdfs上的目标存放位置。 cluster 外部的 locations 指定的为源数据集群内hdfs上的源数据存放位置。

在集群1的linc-1 上，使用hdfs 用户，在任意目录下新建任意文件上传到 hdfs 上作为 源数据，然后执行如下命令：

S hadoop fs -put -p ./任意文件/fangsource

s vim /examples/entity/filesystem/source-target-feed.xml

文件内容修改为如下内容：

v8"ed"name="repl-feed"xmlns="uri:falcon:

feed:0.1">

<groups>input</groups>

<timezone>UTC</timezone> <clusters>

<frequency>minutes(10)</frequency>

<late-arrival cut-off="hours(1)"/>

<cluster name="sourcehdfs"type="source">

<validity start="2013-01-01T00:00Z"end="2030-01-01T00:00Z"/>

<retention limit="hours(2)"action="delete"/>

</cluster>

<cluster

<validity <retention

name="targethdfs"type="target">

start="2013-11-15T00:00Z"end="2030-01-01T00:00Z"/>

limit="hours(2)"action="delete"/>

<locations>

<location type="data"path="/fangtarget"/>

</locations> </cluster>

<location type="data"path="/fangsource"/></locations>

<ACL owner="hdfs"group="group"permission="0x644"/>

<schema location="/schema/log/log.format.csv"-provider="csv"/>

</feed>

**然后将实体上传至** **Falcon** **中：**

第3章 大数据治理之Apache Falcon 109

s falcon entity -submitAndSchedule -type feed -file examples/entity/filesystem/ source-target-feed.xml

上传 feed 后可在 Web UI内的 Data sets分目录下上传配置好的 feed 文件，内容与 source-target-feed.xml文件内容类似，并且状态为RUNNING, 表示任务开始运行。

**6.结果验证**

等待一定时间后，在集群1的linc-1上，可以在Falcon的 Web UI中看到状态由 RUNNING 变成 SUCCESS。之后在集群2的linc-1上，使用hdfs用户，在任意目录下执行 下面命令：

$hadoop fs -ls /fangtarget

如果在集群2中创建了repl-in 文件夹，并且其中有文件内容被成功传入，并且与在集 群1中上传的文件相符，那就证明传输成功。

也可在集群2的Oozie Web端查看具体的调度任务，并且查看其状态，可以发现 SUCCESS 状态，这代表任务已经执行成功。

还可以在HDFS Web 端进行文件的浏览，对任务执行的结果进行验证，可以发现在设 定的目录文件夹下有数据传入，并且附带一个\_SUCCESS 的文件，这代表数据已经传输 成功。

**3.3.5** **数据镜像**

Data-Mirroring 是Falcon另一个重要的功能。顾名思义，数据镜像可用于数据的备份 以及恢复，使得关于数据的处理和恢复更为方便。而且这是一个与前文上传实体等方式不 同的另一操作体系，同时数据镜像也支持跨集群的传输。

要完成此项场景需要确保 Falcon以及Oozie 安装成功，已经进行过NTP时间校正。如 果时间不同步可能会影响 Hadoop 的 Job作业调度。

**1.配置必要文件夹**

配置文件中的/apps/falcon/staging 、/apps/falcon/working 、tmp文件夹如果不存在，请 在集群1的 linc-1上按照如下方式创建：

s hdfs dfs -mkdir /apps

$hdfs dfs -mkdir /apps/falcon

s hdfs dfs -mkdir /apps/falcon/staging

s hdfs dfs -mkdir /tmp

s hdfs dfs -mkdir /apps/falcon/working

s hdfs dfs -chmod -R 777 /apps/falcon/staging

$hdfs dfs -chmod -R 755 /apps/falcon/working

**2.提交cluster 实体**

下面详解其中主要参数。

110 第二篇 开源实现篇

按照配置文件，由上而下地分别指定了目标数据集群内的以下参数：

1)读取 hdfs 的地址以及端口。

2)写入 hdfs 的地址以及端口。

3)执行监控的Resource Manager 管理地址以及端口。

4)工作流处理工具 Oozie 的地址。

5)指定消息处理工具。

6)在hdfs 上指定任务工作台路径。

7)在 hdfs上指定任务临时文件路径。

8 ) 在 hdfs 上指定任务工作依赖路径，里面会自动导入依赖的lib 包。

在linc-1 上，使用hdfs 用户，在/var/local/hadoop/falcon-0.9/目录下创建一个文件夹， 并且在其中编写Cluster 文件。

$mkdir test

s vim test/primarycluster.xml

文件内容修改为如下内容：

ersehdfs"description="Standalone cluster"name="primaryCluster"

xmlns="uri:falcon:cluster:0.1">

<interfaces>

<interface type="readonly"endpoint="hdfs:// <192.168.91.100>:9000"version="2.6.0"/> <interface type="write"endpoint="hdfs:// <192.168.91.100>:9000"version="2.6.0"/>

<interface type="execute"endpoint="hdfs:// <192.168.91.100>:8032"version="2.6.0"/> <interface type="workflow"endpoint="http:// <192.168.91.100>:11000/oozie/"version=

"4.2.0"/>

<intae3"/>type="messaging"endpoint="tcp://linc-1:61616?daemon=true" version=

</interfaces>

<locations>

<location name="staging"path="/apps/falcon/staging"/><!--mandatory-->

 k"ipna/ --optional-->

</locations>

</cluster>

根据实际情况，修改配置文件中的IP 地址<192.168.91.100>替换成实际运行 namenode 的主节点IP 地址。

s bin/falcon entity -submit -type cluster -file test/primarycluster.xml

上传 cluster 后可在 Web UI内 clusters 分目录下看到很多cluster 实体列表，并且会

发现刚刚上传好的 cluster实体，其内容与primaryCluster.xml文件内容一致，并且状态为 submitted。

第3章 大数据治理之Apache Falcon 111

在Web UI分目录下点开实体后可见文档内容与我们所写入文件内容一致，说明上传 成功。

**3.上传** **workflows 文件**

workflows 为工作流文件，其中各项涉及 Hadoop 的job-tracker 、name-node以及 Oozie、 Falcon 运行等相关依赖参数，其中填写方法都是运用环境变量的方式，所以不需要进行过 多修改，按照如下方式填写即可。

在linc-1 上，使用hdfs 用户，在/var/local/hadoop/falcon-0.9/test 目录下创建 workflow 实体文件，并对其中内容进行编写。将其内容相关 Hadoop 、Oozie 参数符合系统环境参 数变量，具体文件内容可以参考Falcon 安装路径下 data-mirroring hdfs-replication\hdfs- replication-workflow.xml文件进行修改。

因为Oozie 工作流 workflow 文档需要上传至hdfs 上，任务才能正常的执行，并且hdfs 目录应该和上述文件中配置相一致，所以接下来执行如下命令：

S hadoop fs -mkdir -p /apps/data-mirroring/workflows/

S hadoop fs -put hdfs-replication-workflow.xml /apps/data-mirroring/workflows/

**4.提交** **Recipe properties**

Recipe properties 为数据镜像的配置文件，其中重要的参数如表3-8所示。

**表3-8** **Recipe 部分参数定义**

|  |  |
| --- | --- |
| **参** **数** **名** | **定** **义** |
| falcon.recipe.workflow.path | 指定工作流文件存放在hdfs上的位置 |
| falcon.recipe.workflow.lib.path | 指定Oozie所依赖的sharelib位置。 |
| falcon.recipe.cluster.name | 指定工作流所依赖的cluster name |
| falcon.recipe.cluster.hdfs.writeEndPoint | 指定读写的hdfs地址及端口 |
| falcon.recipe.cluster.validity.start | 指定任务开始时间 |
| Falcon.recipe.cluster.validity.end | 指定任务结束时间 |
| alcon.recipe.acl.owner=hdfs  falcon.recipe.acl.group=hadoop  falcon.recipe.acl.permission=0x755 | 运行账户以及权限设置 |
| drSourceDir=/fangsource  drSourceClusterFS=hdfs://<192.168.91.100>:9000 drTargetDir=/fangtargetmirorring  drTargetClusterFS=hdfs://<192.168.91.200>:9000 | 指定源数据文件夹、位置、目标数据文件夹位置 |

其中值得注意的是，Recipe 的开始时间即 falcon.recipe.cluster.validity.start 参数，不可 以偏离当前时间过远，可以选择设定开始时间为当前实际时间的前一天以内即可。

在linc-1 上，使用hdfs 用户，在/var/local/hadoop/falcon-0.9/test 目录下执行如下命令 创建一个 hdfs目录，此目录要和上述文件描述的相一致，并且将配置文件上传到其中。

**112** **第二篇** **开源实现篇**

$hadoop fs -mkdir -p /fangsource

$hadoop fs -put ./primaryCluster.xml /fangsource

**修改** **recipe 文件，具体文件内容可以参考** **Falcon安装目录下子目录下文件** **data-mirrori**ng\ **hdfs-replication\hdfs-replication.properties。**

s vim hdfs-replication.properties

**文件中主要地方修改为如下内容：**

#####Recipe properties

falcon.recipe.name=sales-monthly

falcon.recipe.workflow.lib.path=/user/hdfs/share/lib/lib\_20160824222955 #####Cluster properties

falcon.recipe.cluster.name=primaryCluster

#Change the cluster hdfs write end point here.This is mandatory. falcon.recipe.cluster.hdfs.writeEndPoint=hdfs:// <192.168.91.100>:9000

rve-e-T00:00Z

rve-0h9e-T00:00Z

#####ACL properties -Uncomment and change ACL if authorization is enabled

falcon.recipe.acl.owner=hdfs

falcon.recipe.acl.group=hadoop

falcon.recipe.acl.permission=0x755

#falcon.recipe.nn.principal=nn/\_HOSTQEXAMPLE.COM

#####Custom Job properties

#Specify multiple comma separated source directories

drSourceDir=/fangsource

drSourceClusterFS=hdfs:// <192.168.91.100>:9000

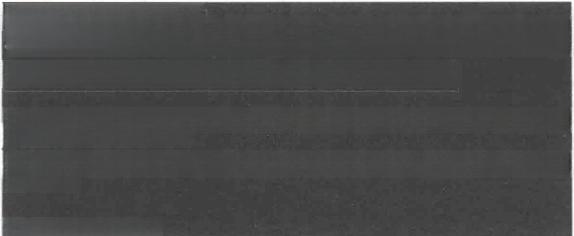
drTargetDir=/fangtargetmirorring

drTargetClusterFS=hdfs:// <192.168.91.200>:9000

**接下来将其上传到** **Falcon 中：**

s falcon recipe -name hdfs-replication -operation HDFS\_REPLICATION -properties hdfs- replication.properties

以上命令可以上传到 recipe 并形成 process 实体，具体过程如图3-17所示。



**Completed recipe PTocessing**

**<2xml version-1.0 encoding**-LTF-8 standnlone-yes ?<result)Xstatus>SUCCEEDEI /status)(message)schedule/default/sales-monthly(PROCESS)scheduled successfully

submit/falcon/default/Submit successful(PROCESS)sales-monthly

K/messageXrequest1d schedule/defauit/5560421708qtp-1871010343-132-02a41199-94 06-4e2c-9f3b-54845ae115d0

submft/falcon/default/5560421706tp-1871010343-132-02a41199-9406-4c2c-9F³b-5ds 45ae115d0

K/requestId>K/result>

图3-17 recipe 形成process 实体

第3章大数据治理之Apache Falcon 113

上述命令执行后会形成一个 process, 在 Falcon 下的 Process 分目录下可以看见形成 process 的 entity 实体，并且任务的状态为 RUNNING, 表示任务已经正在运行。

**5.结果验证**

等待一定时间后在集群2 linc-1 上，使用hdfs 用户，在任意目录下执行如下命令：

$hadoop fs -ls /fangtargetmirorring

显示如图3-18所示，证明运行成功。



[hdfstzte-1 S hadosp ts -Is fangtarsetairorrIng

Found 1 item3s hdrs supergroup 9472016-09-1911:22Kangtargetmlrorring

primaryCluster.xnl

图3-18 执行成功信息

可以在集群2的hdfs Web UI端进行数据验证。fangtargetmirorring 文件夹创建成功并 且内部拥有数据，表示传输成功。

值得注意的是，当前Data-Mirroring 使用的是 Recipe 策略，该策略在 Falcon 0.9 版本 以前主要在 Falcon Client端进行设置并提交，但是在 Falcon 0.10版本之后，Recipe 框架被 移到Falcon Server 端，并更名为 server side extensions。

至此，数据镜像场景已完成，希望各位读者能有所收获，并且可以在此基础上进行开 发，尝试更多配置项的具体使用，并且运用到实际场景之中。

**3.3.6** **数据仓库中的数据操作**

数据仓库中的数据操作场景可实现在Hive 中删除想要删除的数据。该过程的实现原理 是Falcon 首先调度Oozie, 然后后者再调用数据仓库的Hive 引擎，最后Hive 引擎再执行 Hive 脚本，在 Hive 脚本中设置特定的条件，最后达到删除 Hive 表中特定数据的目的。

**1.先决条件**

读者首先需要确保 Hadoop 、Falcon以 及Oozie 安装成功，同时已经进行NTP 时间校 正，如果时间不同步可能会影响 Job 的作业实施。

**2.开启** **Hcatalog 服务**

在前面 Hive 安装配置中，已经讲解了 Hcatalog 的作用，在这里就不重复说明了。

读者需要注意的是，在使用Falcon 调度Hive 例子时，可能会出现因为Oozie 有关 Hcatalog的 libpath 库中jar 包缺失问题。出现这种情况时，读者可以根据oozie logs 中错 误日志提示的class, 将相应的jar 包上传至Oozie 在HDFS 的 share/lib/libXXX/hcatalog、 share/lib/libXXX/hive 和 share/lib/libXXX/hive2 中，并重启Oozie 。其 中XXX 是根据读者 上传文件的时间随机生成的时间后缀。

在 linc-1 节点上，使用hdfs 用户，在任意目录下使用下列命令开启 Hcatalog 服务：

**114 第二篇** **开源实现篇**

9 Hive\_HOME/hcatalog/sbin/hcat\_server.sh start

其中 “Hive\_HOME” 是指 Hive组件的安装路径。Hcatalog Server的默认端口号为9083。

开启Hcatalog 服务后，读者可以在控制台输入jps, 若看到一个 Runjar 进程和对应的 进程号，则表示 Hcatalog 服务开启成功。

**3.创建** **Hive 数据**

在linc-1 节点上，使用hdfs 用户，在Falcon 的安装目录下使用以下命令创建一个 generate.sh 脚本文件。

s vim examples/entity/data/generate.sh

该脚本文件的作用是创建一个数据源，前面在数据管道场景中创建过，有详细的说明， 在这里不重复说明。若读者在之前的场景中已经进行了这一步，则不需要重复进行该操作。

在linc-1 节点上，使用hdfs 用户，在Falcon 的安装目录下使用以下命令创建一个 hcat- generate.sh 脚本文件。

s vim examples/entity/data/hcat-generate.sh

脚本文件的内容如下：

#!/bin/bash

PRG="§(0)"

hcat -e "CREATE TABLE if not exists in\_table(word STRING,cnt INT)

PARTITIONED BY(ds STRING);"

for MINUTE in'seq -w 0059`

do

hcat -e "ALTER TABLE in\_table ADD PARTITION(ds='2013-11-15-00-

SMINUTE')LOCATION'/data/in/2013/11/15/00/SMINUTE';"

done

该脚本文件的作用是将 Hive 中 ds='2013-11-15-00-$MINUTE'分区存储位置指向/data/ in/2013/11/15/00/$MINUTE。

将两个脚本文件内容修改后，保存退出，然后在Hive 的安装目录下执行该脚本文件，

使用命令如下：

$example/data/generate.sh linc-1

$examples/data/hcat-generate.sh linc-1

在创建Hive 数据后，进入Hive 中，可以看到 in\_table 表中按照时间顺序生成的数据，每分钟生成 一个随机数据，在数据的第1列显示的是随机生成的 first 、second 、third 、fourth 、fifth 中的一个，第2列 是随机生成的0～10数字，第4列是指从2013-11- 15-00-00这个时间戳开始，每一分钟执行一次。最后 Hive 表中的数据如图3-19所示。



hives select\*from in\_tabie;

OK

h**i**dth

2013-11-15-00-01

2013-11-15-90-02 2013

2013-11-15-00-04 2013-11-15-00-05 2013-11-15-00-06 2013·11-15-00-07 2013-11-15-00-08 2013-11-15-00-09

2013-11-15-80-10

NULL

2013-11-15-08-11

NULL

NULL NULL

NULL NULL NULL NULL NULL NULL

·

fourth

first

fourth

fifth

fifth

d

rdh

2013-11-15-08-08

5

图3-19 in\_table 部分数据

第3章大数据治理之Apache Falcon 115

**4.提交** **HQL** **脚本**

Falcon增加了Hive 引擎作为Hive 集成的一部分，它使用户能够将Hive 脚本嵌入一个 进程中去。该Hive 脚本作为工作流时属于Hive 工作流。在本章的3.3.3节中有关于Hive 的详细介绍，请有兴趣的读者查看3.3.3节关于 Apache Hive 的介绍，在这里不赘述。

在linc-1 节点上，使用hdfs 用户，在任意目录下(最好是在 Falcon安装目录下，便于 查找管理)使用下列命令创建一个Hive 脚本。该脚本主要指定了Falcon 工作流中的数据处 理逻辑，在此处的作用是从 Falcon 指定的数据库表中删除数据。

s vim examples/entity/filesystem/hcat.hql

将 HQL脚本文件内容修改为如下内容：

ALTER TABLE ${falcon\_inparts\_database).$(falcon\_inparts\_table}DROP PARTITIONS{falcon\_inparts\_filter};

保存退出后，将HQL 脚本上传到HDFS的/app/hive/目录下，使用如下命令进行该 操作：

$hdfs dfs -put examples/entity/filesystem/hcat.hql /app/hive/

在后文中的process实体中可以指定使用Hive 工作引擎来执行 HQL脚本文件。其中 S{falcon\_inparts\_database} 、${falcon\_inparts\_table} 、${falcon\_inparts\_filter} 为参数形式， 将从Oozie 中获取变量的值。

**5.提交** **cluster 实体**

cluster 实体的作用是说明集群的相关配置项，主要有对HDFS、YARN 以及Oozie 相应 配置的说明。在本章前面已经详细介绍了cluster 的作用以及配置项的具体意义，这里不赘述。

在linc-1 节点上，使用hdfs用户，使用下列命令新建一个 hivetable-cluster.xml 文件。

$vim /var/local/Hadoop/falcon-0.9/examples/entity/filesystem/hivetable-cluster.xml

/var/local/Hadoop/falco n-0.9/ 为 Falcon的安装目录，读者可以根据实际情况进行修 改。理论上hivetable-cluster.xml创建目录可以为任意目录，但是为了方便管理，将 其建在 Falcon的安装目录下的 filesystem 文件夹下。

**该文件的内容如下：**

<?xml version="1.0"?>

<cluster colo="ua2"description="Hive Cluster"name="hivetable-cluster"

xmlns="uri:falcon:cluster:0.1"

xmlns:xsi="<http://www.w3.org/2001/XMLSchema-instance>">

<interfaces> <interface

<interface

<interface

<interface

type="readonly"endpoint="hdfs://linc-1:9000"version="2.6.0"/>

type="write"endpoint="hdfs://linc-1:9000"version="2.6.0"/>

type="execute"endpoint="linc-1:8032"version="2.6.0"/>

type="workflow"endpoint="http://linc-1:11000/oozie/"

**116** **第二篇** **开源实现篇**

 - 1:61616?daemon=true"

5.4.3"pe="registry"endpoint="thrift://linc- 1:9083"version="1.2.1"/>

</interfaces>

<locations>

<location name="staging"path="/projects/falcon/hivetable-staging"/>

<location name="temp"path="/projects/falcon/hivetable-tmp"/>

<location name="working"path="/projects/falcon/hivetable-working"/>

</locations>

<properties>

</properties>

</cluster>

该 cluster的名称为hivetable-cluster, 并定义了Falcon 执行任务过程中将会用到的各个 接口。

在 cluster 中 location 需要在 hdfs上有相应的目录，在提交该 cluster 前需要在 hdfs上 创建相应的目录文件，否则在上传 cluster 文件时会报找不到路径的错误。

在将 cluster 文件保存退出后，在 Falcon 目录下使用以下命令提交 cluster 实体：

$bin/falcon entity -submit -type cluster -file examples/entity/filesystem/hivetable\_ cluster.xml

提交后，在控制台若提示 falcon/default/Submit successful(cluster)hivetable-cluster 则表 示该 cluster文件提交成功。在集群的 Falcon的 Web UI(https://linc-1:15443) 进入clusters 目录即可看到提交成功的 hivetable-cluster。

素 linc-1是主节点的IP 地址，读者根据实际情况输入该IP。

单击hivetable-cluster, 可以看到 cluster 的详细内容。

**6.提交** **feed** **实体**

feed实体文件说明 feed与cluster之间的依赖关系，介绍xml配置项中 feed frequency、 cluster 、retention policy,以及data set location的作用。该 feed实体指向 hivetable 元数据。 在linc-1节点上，使用hdfs用户，在Falcon安装目录下使用如下命令创建一个 feed 实体。

$vim examples/entity/filesystem/hcat-in-feed.xml

将 feed文件的内容修改为如下：

 vgn=a"mUeT=F"--in"xmlns="uri:falcon:feed:0.1">

<frequency>minutes(1)</frequency>

<timezone>UTC</timezone>

<late-arrival cut-off="hours(1)"/>

<clusters>

<cluster name="hivetable-cluster ">

<validity start="2013-01-01T00:00Z"end="2030-01-01T00:00z"/>

**第3章** **大数据治理之Apache Falcon** 117

<retention

</cluster> </clusters> <!--<table

limit="hours(2)"action="delete"/>

uri="catalog:default:in table#ds=2013-11-15-00-01"/>-->

<table uri="catalog:default:in\_ <ACL owner="testuser-ut-user"group="group"permission="0x644"/>

<schema location="/schema/log/log.format.csv"provider="csv"/></feed>

feed 实体定义了在Falcon 执行工作流的过程中将使用到的数据集的各种属性，例如数 据集的位置、数据及使用的频率，以及数据集的有效期限等。

将上述 feed 文件保存退出后，在 Falcon安装目录下使用如下命令，将 feed 实体上传到

HDFS。

$falcon entity -submit -type feed -file examples/entity/filesystem/hivetable-

infeed.xml

上传 feed实体后，若在控制台提示 falcon/default/Submit successful (feed)hcat-in 则表 示提交成功。从集群在 Falcon的 Web UI(https://linc-1:15443) 进入feed 目录即可看到提 交成功。单击hcat-in, 可以看到 feed 实体的详细内容。

**7.提交** **process** **实体**

process 实体定义了运行工作流所需要的配置。在linc-1 节点上，使用hdfs 用户，在 Falcon安装目录下使用如下命令创建一个process 实体。

$vim examples/entity/filesystem/hive-process.xml

**将该** **process 实体内容修改为如下：**

<?xml version="1.0"encoding="UTF-8"?>

<process name="hive-process"xmlns="uri:falcon:process:0.1">

<clusters>

<cluster name="hivetable-cluster ">

<validity start="2013-11-15T00:00z"end="2013-11-15T01:00z"/></cluster> </clusters>

<parallel>1</parallel>

<order>FIFO</order>

<frequency>minutes(5)</frequency>

<timezone>UTC</timezone>

<inputs>

<!--In the pig script,the input paths will be available in a variable 'inparts'--> <input name="inparts"feed="hcat-in"start="now(0,0)" end="now(0,0)"feed="hcat-

in”/>

</inputs>

<workflow engine="hive"path="/app/hive/hcat.hql"/>

</process>

该process 实体指定工作的集群以及有效的工作时间和频率，在工作流区域调用的脚本 文件。保存退出后，在Falcon 安装目录下使用如下命令将 process 实体上传到 HDFS。

**118** **第二篇** 开源实现篇

s bin/falcon entity -submitAndSchedule -type process -file examples/entity/filesystem/ hive-process.xml

上传 process 实体后，若在控制台提示 schedule/default/hive-process(process)scheduled successfully submit/falcon/default/Submit successful (process)hive-process 则表示提交成功。 读者可以登录集群在 Falcon 的 Web UI(https://linc-1:15443) 进入 processes 目录，即可看 到提交成功的 hive-process 。单击hive-process, 可以看到提交的 process 详细内容。

**8.测试结果**

将process 成功上传到HDFS 后，可以在集群的Oozie 的 Web UI(http://linc-1:11000/ oozie) 上查看Job 正在执行。

当整个Job 完成后，在HDFS 的 Web UI(htp://inc-1:50070) 中的/data/in/2013/11/15/00 中查看数据，可以发现，所有Name 为5的倍数的数据均已经被删除，这是由process 中 <frequency>minutes(5)</frequency> 决定的，每5分钟执行一次。删除后的数据如图3-20

所示。

|  |
| --- |
| **Browse** **Directory** |
| /data/in/2013/11/15/00 |
| **Peaission** OwDer GrotP Size Replication Block Size Nane |
| drvxr-xr-x hdfs supergroup 0B 0 0 B 01 |
| yxr-X- hdfs super group 0 B 0 0 B 02 |
| dryxr-xT-X hdfs super group 0B 0 0 B 03 |
| dryxr-xr-X hdfs supergroup 0 B 0 0 B 04 |
| davxr-xr-X hdfs supergroup 0 B 0 0 B 06 |
| \*yx-xr-X hdfs supergroup 0 B 0 0 B 07 |
| ayxr-xr-X hdfs super group 0 B 0 0 B 08 |

图3-20 数据定期删除后的显示数据

**3.4** **Apache** **Falcon** **优化与性能分析**

经过前面几章的介绍，大家已经对Apache Falcon 以及其作业编排调度有了一定了解。 由于Apache Falcon位于Oozie 之上更高级的抽象层，只负责对 Hadoop 工作流的调度与管 理，对于具体的例如Map Reduce 过程，还是要依赖于具体的 Hadoop 组件，因此对于工作 流执行效率，更多地还是要依靠 Hive 、Map Reduce 本身。

**3.4.1 Apache Falcon 控制流**

如表3-9所示，目前 Apache Falcon 0.10 版本支持的 Hadoop工作流组件有Hive 、Sqoop、

第3章 大数据治理之Apache Falcon 119

Map Reduce 以及 Distcp。由 于Apache Falcon作为数据治理平台的一个工具，主要负责 对数据生命周期管理以及对数据处理管道的管理调度，因此它所支持的组件工具都是与 Hadoop 数据相关的组件。

**表3-9** **Apache Falcon v0.10 支持组件表**

|  |  |  |
| --- | --- | --- |
| **组** **件** **名** | **版** **本** **号** | **说** **明** |
| Oozie | 4.2.0+ | Falcon核心调度器，负责Hadoop任务执行调度 |
| Hadoop | 2.X | 主要有HDFS分布式文件存储系统及Map Reduce数据计算框架 |
| Pig | 0.14+ | 结构化数据处理脚本语言 |
| Hive | 1.1X+ | 分布式数据库存储，使用HQL脚本语言 |
| Sqoop | 1.4.6+ | 数据库连接器，负责不同存储类型的数据的导入导出 |
| Distcp | 2.X | 跨集群数据传输工具 |
| Atlas | 0.6+ | 数据审计及追踪工具 |

其 中Apache Falcon目前放弃了对Hadoop 1.0版本的支持，这意味着之前能够在 Hadoop 1.0版本下运行的组件无法兼容Falcon0.10, 其中影响最大的就是Oozie。 因为 Oozie 的工作流文件workflow.xml 在 Hadoop 1.X版本中的设置参数与配置项与 Hadoop 2 不尽相同(这也是 Oozie 被诟病最多的地方之一，许多协议与设计前后版本更替之间存在冲 突),所以 Falcon 当前仅支持Oozie 4.X版本及以上。

Apache Falcon也支持对数据库中存储数据进行数据治理，其中对Hive 的整合主要依 靠 Hcatalog工具得以实现。在之前，Hive 脚本语句主要利用类 SQL 查询中对数据库中的表 进行数据操作，无法获得数据库中数据在文件系统中的存储位置以及数据结构等具体信息。 Hcatalog 屏蔽了底层数据存储的位置格式等信息，为上层计算处理流程提供统一共享的元 数据 metadata, 实现了对于表以及底层数据的统一管理。这使得我们可以通过Hcatalog 定 义Hive 数据库表数据文件所在的 HDFS存储路径，也可以根据 HDFS 文件所在路径生成 Hcatalog执行脚本实现对Hive 数据库中的表数据进行处理。在0.11版本后，Hive 集成了 Hcatalog 。Apache Falcon 通过 Hcactlog 来执行 HQL 脚本以及Pig 脚本语句， Apache Falcon 通过周期性地调度Sqoop 任务，实现采集数据库数据到 HDFS 文件系统，以及将 HDFS 文 件导出到数据库中存储。但是截至Apache Falcon 0.10版本，它仅支持进行JDBC 连接的关 系型数据库 (MySQL 以及Oracle)。

对于Falcon 控制流来说，由于底层数据处理分别交由各个Hadoop 模块组件完成， Apache Falcon 只是通过调度器协调各个组件之间的工作，这使得整个工作流效率很容易产 生一个瓶颈，且难以逾越。

**3.4.2** **分布式部署**

大数据治理往往需要采集不同数据源的数据，并在不同集群任务之间进行合作调度。

120 第二篇 开源实现篇

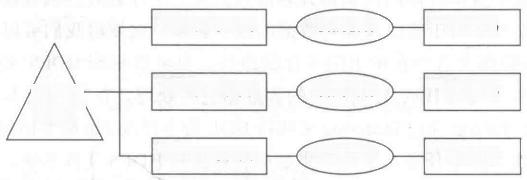
这意味着一个大数据治理平台，必须能够统筹兼顾位于不同网段地址、不同物理地址的不 同集群。对此， Apache Falcon 在单机模式基础上，扩展了分布式部署模式。

Apache Falcon分布式模式有两个组件， Prism 和 Server 。Prism作为两个 Server 的连接 点，所有命令对Prism 使用，而Server 只能调用read以及实例instance 相关的API 命令。 Prism 和 Falcon 也有各自不同的启动及运行配置项，配置文件存储地址也各不相同。

Prism 主要保证不同集群的配置项产生的工作流实体及运行实例之间的同步，实现对 任务调度的协调，操作请求可以来自Prism 本身，也可以通过在Client 端或者 Web 端请求 API后面添加-colo 参数执行特定集群，具体操作包括提交Submit 、调度 schedule、暂停 Suspend 以及继续 Resume 等。总而言之，Prism 主要负责保证 Hadoop 任务工作流正常运 行，或者工作流调度指令，并发送至 Server端。

Apache Falcon Server 端主要部署在工作流调度所在集群上，负责接受Prism 提交或运 行Hadoop任务实体的请求，其中cluster 实体由Prism发送至每一个 Server, 而对于 feed 以及process实体，则根据XML 文件中cluster 配置标签属性中相应内容，发送至指定集群 的 Server 端。

Prism整体全局视图如图3-21所示。Falcon Prism连接着多个 Falcon Colo集群，协 调并运行上面的调度任务。Prism 与 Falcon Server之间的通信(用户提交 submit 或者更新 update 实体功能)连接主要基于HTTPS协议实现，身份安全验证基于客户端证书的验证。 因此 Prism 服务器提供有效的客户端证书供 Falcon Server接 收 操 作 。Apache Falcon 可以实 现基于安全传输层协议 (TLS) 进行消息传输，保证 Prism与Server 之间数据的保密性和 数据完整性。如果需要启动TLS 协议，则 Falcon Server和 Prism 服务器中的启动属性文件 startup.property 都需要配置以下密钥信息：\*keystore.file\*keystore.password。



|  |  |  |  |
| --- | --- | --- | --- |
|  | Falcon Colo 1 | Schedule | Hadoop |
| Falcon  Prism | Falcon Colo 2 | Schedule | Hadoop |
|  | Falcon Colo 3 | Schedule | Hadoop |

图3-21 Apache Falcon Prism 全局视图e

通过分布式部署的 Falcon, 可以实现跨集群工作流协调调度，保证不同 Hadoop 集群在 统一标准的数据治理平台下的连贯性与一致性。

**3.4.3** **安全模式**

当出现一些特殊情况，Apache Falcon管理员希望可以防止 Falcon 用户调度工作流引擎

θ <http://falcon.apache.org/FalconDocumentation.html>

第3章 大数据治理之Apache Falcon 121

中的实体时，启动安全模式 (Safemode) 则显得很有必要。这些特殊情况可能有：

1)即将或正在升级 Hadoop集群。当集群管理员正在升级 Hadoop 或者其他例如 Hive、 Pig 等组件时，这些组件所涉及的工作流任务无法正常运行。

2)正在更新 Falcon 中已提交存储的工作流实体。对于之前提交的工作流实体，无论是 cluster 、feed, 还是 process, 在更新时可能都会涉及集群属性配置、数据源文件路径或者 整个工作流业务逻辑的变更。Apache Falcon 服务器中存储的源工作流实体文件前后不一致， 也会使得调度任务毫无意义。

在安全模式下，Falcon 用户只能进行有限操作，具体的操作有以下9项：

1)可以对其有认证授权的实体执行读取操作。如果该用户没有获得要操作实体的读取 权限，则无法进行该操作。

2)拥有超级用户权限的用户可以执行对集群的实体进行更新操作。

3)如果用户想要暂停特定的工作流操作，可以对该实体工作流中的单个实例执行挂起 或终止操作。

4)对可调度的实体，可以执行挂起操作。因为在进行版本更新时(无论是集群更新还 是工作流实体更新),可能会出现实体的作业更新前后版本不一致的情况，而这显然是不被 允许的。因此用户可能希望在滚动升级期间挂起实体，以处理与更新版本不兼容的作业。

5)其他所有操作用户都将不被允许操作。包括：

口所有实体的 submit 和 submitAndSchedule 操作。

口实体不能被执行如下操作update、schedule、touch、delete、submit、submitAnd-

Schedule 以及 resume 等。

口实例操作例如 rerun、resume。

Falcon 进程用户可以使用以下命令在启动 Falcon 时开启安全模式。

s falcon-server>:/bin/falcon-start -setsafemode <true/false>

拥有超级用户权限或者管理员权限的用户通过Client 端或者 RestAPI设置Falcon Server 开启或退出安全模式。如果用户为Falcon 进程启动用户，或者属于 startup.property 启动配 置文件中<falcon.security.authorization.superusergroup>属性指定组，则该用户被视为超级 用户。如果用户被列在 startup.property 启动配置文件中<falcon.security.authorization.admin. users> 属性中，或者属于startup.property 启动配置文件中<falcon.security.authorization.

admin.groups> 中列出的组，则该用户被视为超级管理员用户。

Client 端启动及退出命令如下：

s falcon-server>/bin/falcon admin [-setsafemode <true/false>]

RestAPI 连接两个 URL 如下所示。

**口** 开启安全模式 API:GET http://falcon-server:15000/api/admin/setSafeMode/true **口** 退出安全模式 API:GET http://falcon-server:15000/api/admin/setSafeMode/false

122 第二篇 开源实现篇

需要注意的是，普通的用户也可以通过Admin Version API 来查看 Falcon Server 是否处 于安全模式。 一旦 Server 端被置为安全模式，则该状态在重启之前都会一直保存。如果退 出安全模式，则该状态将会被清除。

通过安全模式，可避免特殊情况下用户不当操作对整个工作流造成的不良影响，保证 了整个调度流程的一致性。

**3.4.4 Apache Falcon 优化**

Apache Falcon将底层任务交由 Hadoop 各个功能组件完成，然后通过Oozie 调度器实 现对各个Hadoop 任务的调度工作，所以当各个任务执行效率达到瓶颈时， Apache Falcon 也无法提高其底层运行效率。目前 Apache Falcon 可以考虑的优化方案有两方面， 一个是调 度器调度优化，还有一个是作业流编排优化。

**1.调度器优化**

目前 Oozie 调度器可以覆盖完成 Apache Falcon所能调度的工作流引擎组件，但是随着 Falcon 的发展， Oozie 的弊端也逐渐显现。首先是 Oozie 工作流调度复杂，对于初学者上手 学习比较困难，而且缺少指导文档，甚至Oozie 版本更替造成了API 前后不一，而许多文 档也没有及时更新，这对于初学者来说简直就是灾难。然而对Apache Falcon来说， Oozie 最为严重严重的一点是Oozie 的更新与 Apache Falcon未保持一致，且Oozie 目前是 Falcon 核心调度器，无论安装还是作业工作流部署步骤都十分烦琐，这意味着 Apache Falcon 之 后发展将一直受限于Oozie 的开发。作为底层工作流引擎，例如 Hive 、Map Reduce 等， Apache Falcon 如果重新开发或者补充，则代价过大。而Oozie 作为中间层调度器，重新开 发一个新的调度器成本代价却比较低廉，而且可以保证新的调度器与 Apache Falcon可以更 好地结合。Falcon 自带原生调度器 (Native Scheduler) 就是 Apache Falcon 项目目前开发的 重心。目前原生调度器可以实现的功能有以下两点：

1)提交和编排运行 Falcon 进程 (Process) 计划(无数据依赖)。该 Falcon 进程可以是 Pig 脚本、Oozie 工作流以及Hive, 也就是目前使用Oozie 调度器所适用的工作流引擎都可 以使用原生调度器替换。

2)监视、查询、修改调度中的 process 进程。所有适用的实体 API 和实例 API都可以 按照现有的方式工作。此外 Falcon 也通过声明性编程提供了feed 数据集管理功能。它允许 用户在包含文件的 HDFS 上使用基于时间的分区目录来表示 feed 数据集所在路径。

当前原生调度器可以代替 Oozie 调度器实现大部分功能，但是依然有需要优化的地方。 在未来几个版本中，原生调度器可能会实现以下6个功能：

口基于日历的周期性日程安排。

口基于数据可用性的调度。

口基于外部触发/通知的调度。

口支持定期/不定期采集的数据集。

第3章 大数据治理之 Apache Falcon 123

口支持可选/强制数据集。可以指定要使用数据实例的数值选项包括最小(minimum)、

最大(maximum) 、精确选择(exactly-N), 以供用户使用。

口在重新运行时可以重新调整工作流实体之间的依赖关系。

**2.工作流编排优化**

除了对调度器进行优化外，合理地安排数据治理的任务作业，不但可以增加工作流引 擎执行效率，也可以保证工作流的流畅性。在作业编排时需要注意的优化点有以下几点：

1)基于时间为数据集文件命名。由于大数据治理经常涉及有关数据生命周期管理部 分，许多数据处理操作都与时间相关，当使用时间来确定数据集所在路径时，无论是输入 数据集还是输出数据集，不但可以方便检索，也能够保证前后工作流顺畅运行。

2)选择最优JMS 消息服务器。Falcon 内部自动生成系统通知，并用于监控 Falcon 编 排的工作流作业。默认情况下， Apache Falcon启动Falcon机器上端口号为61616的 ActiveMQ 嵌入式JMS 服务器作为守护程序。用户可以根据自己的实际情况选择已有的JMS 消息服务 器来代替。代替步骤如下所示：

在 startup.properties 配置文件中如下修改代理链接地址。

\*.broker.url=tcp://jms-server-host:61616

或者在启动Falcon 进程时添加参数项： -Dfalcon.embeddedmq=false。

3)确保 Falcon 实体标题名称与JMS 消息 Topic一致。为了保证之后数据血统追踪以 及用户的消息消费能够与工作流实体相对应，用户可以直接从JMS 代理获取实体执行状 态。在一段时间(默认3天)后，JMS 消息会被Falcon JMS管理服务自动清除。可以在 Falcon的 startup.properties 文件中配置JMS 消息的 TTL (生存时间)。

**3.5 Apache Falcon 应用举例**

**3.5.1** **InMobi** **基于** **Falcon** **的数据治理**

2015年1月， Apache 基金会宣布 Apache Falcon 项目通过了孵化阶段，目前已经成为 基金会的顶级项目，这个项目是由InMobi 公司开源的。InMobi 于2007年在印度的班加罗 尔建立，现在是印度最大的移动广告公司，全球第二大移动广告公司，全球最大的独立移 动广告网络。InMobi 公司对Falcon 的应用是最早的，Falcon 最先由InMobi 公司应用。在 升级成为顶级项目之前，Apache Falcon已经在多个行业中获得了广泛的应用，包括广告、 医疗、移动应用等，而InMobi 是该平台的最大用户之一。

这一切是怎么开始的呢?在2011年年末， InMobi 团队通过构建数据和数据处理管道 来分析移动广告数据。InMobi 的广告业务每天有超过100亿的事件到达。对这些事件进行 分析、报告和推断，使得InMobi的广告在合适的时间推送给合适的人是InMobi 的开发人 员所要做的事情。面对如此大量的事件，InMobi 仅仅广告业务的处理就拥有200多个复杂

124 第二篇 开源实现篇

的大数据管道和不同的数据源。随着移动端广告惊人的增长，他们意识到他们必须迅速地 改变现有的管道，并铺设新的管道。随着业务的复杂度的增加，他们想到要简化管理和操 作这些管道，只需要将数据处理管道中的共同的需求抽象出来并构成一个平台即可。所以 InMobi 的研发人员开发了一个数据生命周期管理框架，也就是现在成为Apache 顶级项目 的 Falcon

在InMobi 的业务中，对客户的行为进行分析是必不可少的一部分，比如对于用户对广 告的点击、广告的转化率等。当时InMobi 遇到了一个难题，他们想要对全球的数据进行分 析，他们希望不仅是存储一两个来源的部分交易，而是希望能够获取和保存尽可能多的不 同来源的交易，以便有选择、有针对性地投放广告。InMobi 一个大洲原先有几百个数据源 来源，对这些数据源的数据进行管理已经很麻烦了，如果推广到全球的数据，他们已有的 框架已经不能够再支撑他们的业务了。而且在扩大他们的业务之后，客户每小时产生的点 击流数据达到数百MB, 每月增加的数据达数十亿行。InMobi 公司分析每个广告的布置并 确定点击率和转换率。在展示文件和点击文件存储在多个不同的数据源的情况下，机构没 有方法来将产生的不同的数据快速地上传到 Hadoop, 所以他们必须对不同的数据源制定不 同的上传策略。但是对于海量的数据以及众多的数据来源，现有的解决方案已经显得捉襟 见肘了，这个时候他们自己研发的 Falcon 进入了他们的视野。

数据治理框架 Falcon 帮助InMobi 公司快速处理海量数据，跟上市场瞬息万变的节奏， 其通过制定多个工作流组合分析来定位其广告的点击量，并改善其用户体验，尤其是针对 高价值客户进行这些工作。在使用Falcon 进行数据生命周期的管理，并使用Falcon 定义不 同的工作流之后，数据的上传以及数据流的处理变得简单了，工程师们只需要专注于业务 的分析，而不用花费大量的时间去构建不同的工作流。正是因为Falcon 对公司广告业务的 提 升 ，InMobi 公司仅仅一季度之后其广告的营收就增加了10%。

现在随着开源社区对Falcon 的完善， InMobi 对Apache Falcon的应用已经深入到其 业务的各个方面。今天，400多个管道和2000多feed 构 成 了InMobi 移动数据分析的主 要部分。InMobi 在部署如此多的管道时也遇到了麻烦，即如何保证管道在部署到生产环 境之前的正确性。对于管道的测试是解决这个问题的最佳方案，但是现在对于管道的测试 是烦琐的。为了测试这些管道，需要先在QA 环境中部署Apache Hadoop 、Apache Oozie (scheduler) 和Apache Falcon 。InMobi 在测试这些管道时遇到了许多的问题。为了解决这 些问题，InMobi 自己对开源的Apache Falcon进行了个性化的改进，该公司现在所使用的 Falcon 是由该公司的工程师对Apache Falcon 进行改进之后的版本。

Apache Falcon起源于InMobi 公司，并由其开源，后来经过Apache 社区人们的努力， 使得这项技术进入大众视野，造福全球。Apache Falcon 通过建立一系列的管道来使得数 据的处理变得简单，对于一个工作流只需要对实体、feed 等进行声明，大大简化了开发流 程。这正是Apache Falcon平台最大的特点，它本身只维护依赖关系，而并不做任何繁重的 工作。虽然 Apache Falcon的主要作用不是以最终的形式直观地展现，但是它对于整个使用

第3章 大数据治理之Apache Falcon 125

Hadoop 平台进行数据分析的业务带来了巨大的提升。

**3.5.2 Expedia 基于** **Falcon 的数据治理**

Expedia 作为全球最大的在线旅游公司，于2007年通过与艺龙的合作正式进入中国市 场，现有业务部门遍及美国、加拿大、法国、英国、比利时、德国、意大利以及西班牙。 TripAdvisor 是Expedia 旗下品牌，目前是全球最大的旅游社区，在酒店和景点点评服务上 拥有绝对领导性地位。Expedia 网站提供机票预定、租车公司，以及全球超过3000个地点 的旅馆及超值优惠的房价，旅馆的详细信息也可于线上浏览。旅客只要输入心中理想价位 与地区等，在最短的时间内，即可得到最及时且准确的报价。

说到 Expedia 和 Apache Falcon之间的关系，Falcon 成为顶级项目之前， Expedia 就已 经开始大规模的使用Falcon 进行数据管理了。当时 Expedia 的发展非常快，其数据被存储 在多个系统上， Expedia 的业务又非常多，而且业务之间交叉较多，通常一个业务所需要用 到的业务源达到了5个之多，对于数据的管理就显得尤为重要，此时 Falcon 进入了他们的 视线。Apache Falcon 是一个面向 Hadoop 的、新的数据处理和管理平台，设计用于数据移 动、数据管道协调、生命周期管理和数据发现。它使终端用户可以快速地将他们的数据及 其相关的处理和管理任务“上载(onboard)” 到 Hadoop 集群。Expedia 通过调研发现，在 Apache Falcon中，“基础设施端点 (infrastructure endpoint)”、数据集(也称 feed)、处理规 则均是声明式的。这种声明式配置显式定义了实体之间的依赖关系。这也是该平台的一个 特点，它本身只维护依赖关系，而并不做任何繁重的工作。所有的功能和工作流状态管理 需求都委托给工作流调度程序来完成。这正是 Expedia 工程师所要寻找的。

Expedia 引进了 Falcon 平台，通过Falcon, 他们将 Hadoop 环境中的各种数据和处理数 据的规则进行了整合，使数据和数据的使用规则之间建立了联系。在Expedia 内部，工程 师们将HDFS 上的数据进行了整合， Hadoop 、Hive 、HBase 上的数据都被关联起来。“这是 一个令人难以置信的平台，它完美地解决了我们公司由于业务的发展而产生的问题。”时任 Expedia首席技术官的Pierre Samec当时这样表示。最为重要的是，通过Falcon,Expedia 公司构建了数据湖，在数据湖中，不同种类的数据汇聚到一起，Expedia 通过数据湖对公司 的业务提供了实时的分析，最为典型的就是对用户关注最热的地区进行实时分析，并为该 地区的旅行项目提供广告支持，以吸引更多的游客，并有针对性地对客户投放广告。通过 这项技术，该公司的广告营业收入在应用当月就得到了相当大的提升。不仅仅是实时的分 析，对于其他的BI 分析， Falcon 也提供了巨大的帮助。通过Falcon,Expedia 技术团队也 构建了一个复杂的管道，这些管道如果仅仅使用Apache Oozie将会造成巨大的工作量，甚 至可能难以完成。

在数据治理的安全方面，Expedia 公司有很高的要求，因为 Expedia 直接面向用户提供 服务，所以对于数据的安全管理是Expedia 选择 Falcon 的原因之一。Falcon 在安全方面也 为 Expedia 提供了巨大的帮助，Falcon 在传输层面提供了安全保证，以确保数据的保密性

126 第二篇 开源实现篇

和完整性，对于Expedia 这种大型的互联网公司来说，安全的重要性是不言而喻的。Falcon 提供了简单的安全认证和Kerberos 这两种安全认证方式。在简单的认证方式下， Falcon 的 使用安全不能得到保证，但是Expedia 使用的是 Kerberos 集成的 Apache Falcon 服务。通过 ACL(访问控制表),Expedia 为不同的用户组提供了不同的权限，完美地解决了数据访问权 限管理的问题。Expedia 的用户较多，提供的服务也很多，所以数据的安全性需要得到保障， 正是Falcon 的安全机制，使得 Expedia 的数据管理得到了质的提升。

现阶段， Falcon 越来越成熟，可以被应用到更多方面，这正是使用Falcon 平台的公司 不断推进的结果。Expedia 的数据存储在 HDFS 相关的平台上，所以在对数据进行管理时， Falcon 对 Hive 等平台的支持尤为重要， Expedia 通过自己的力量也促进了Falcon 对 Hive的 集成。Expedia 公司不仅将Falcon 应用到了其公司的业务中，还推动着 Falcon 的发展。正 是这些使用Falcon 的公司活跃在开源社区，对Falcon 的发展贡献着自己的一份力量。

**3.6** **本章小结**

通过这一章节，我们接触了大数据生命周期以及数据管道管理工具Apache Falcon, 了 解了Falcon 的原理架构以及工作流编排过程。对于从不同数据源采集数据至“数据湖 (data lake)”, 到对“数据湖”的数据管道管理，乃至对数据处理操作作业，Apache Falcon 都是 目前最为成熟的模块。而如果单独使用Oozie 来调度任务的话，工作量和工作复杂度都十 分庞大。

目前 Apache Falcon 已经在一些领域有了成功案例，主要是在医疗以及通信等需要大规 模数据治理的行业。然而随着大数据时代的发展，信息孤岛以及信息安全危机等问题日益 严重，大数据治理的重要性也愈加显现出来，而在那个时候，Apache Falcon也一定会随着 开源社区的开发，得到广泛关注与应用。

■

置

■■|B| ■1■ ■ ■|

■

**大数据治理之** **Apache**

|  |  |  |  |
| --- | --- | --- | --- |
| ■  ■1 | ■1  ■ ■ ■ | ■1  ■1  ■ | ■ ■  ■  ■  H  ■ ■ ■ |

■ 国

■

■

■ ■

■

■

第4章

**Atlas**



本书的第3章介绍了大数据治理组件之一——Falcon, 它能够对大数据平台执行包括 数据采集、数据处理、数据备份和数据清洗在内的数据生命周期管理，也能够对大数据平 台的各种组件进行更好的调度。但大数据治理的内容远远不止数据生命周期的管理和集中 调度，它还包括元数据管理、数据生命周期的审计和可视化显示、数据血统的搜索以及数 据安全与隐私等内容，因此Apache Atlas作为大数据治理的另一个重要组件，与 Apache Falcon 一起在大数据治理中发挥着十分重要的作用。本章将会对 Apache Atlas 的基本概况、 发展历程、优势与架构、配置与使用以及场景设计与配置进行详细介绍。

**4.1** **Apache Atlas 概述**

**4.1.1 Apache Atlas 技术概况**

**1.元数据管理概述**

元数据是描述数据的数据(data about data), 是指从信息资源中抽取出来用于描述其特 征与内容的数据。从一般意义上来说，元数据是指数据的类型、名称和值等；在关系型数 据库中，常常指数据表的属性、取值范围、数据来源，以及数据之间的关系等。

元数据的管理有着十分重要的作用，它能够为数据用户提供完整的数据定义信息，减 少数据冗余，有利于识别与查找数据。同时，能够追踪数据在数据库中发生的任何变化， 帮助用户理解数据在整个生命周期中的来龙去脉，实现简单高效地管理大数据系统中的海 量数据，并且通过对数据资源的有效跟踪、发现、查找来挖掘大数据系统中数据的价值。

在大数据治理活动中，元数据与元数据管理有以下要点。

128 第二篇 开源实现篇

(1)数据管理

数据管理要求能够追踪数据的整个生命周期，包括数据的来源、数据修改与删除，并 能够支持快速检索。

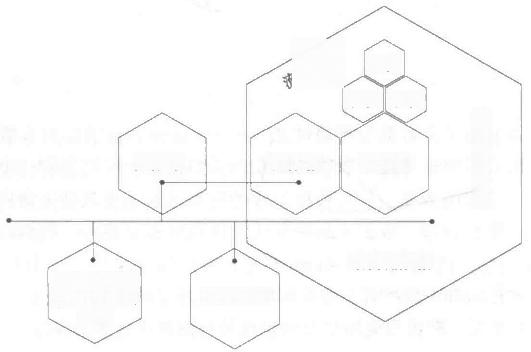
(2)元数据建模

元数据建模通过结合标签与数据属性的方式来更好地理解数据及其生命周期，从而实

现对数据的快速建模。

(3)易于交互的解决方案

通过建立统一 的、贯穿 Hadoop 生态系统的元数据库，定义统一 的元数据标准，为系统 中不同组件的元数据信息进行交互提供基础。图4- 1描述了围绕数据湖 (Data Lake) 的结构 化与非结构化数据、实时数据与非实时数据的元数据与相关应用的拓扑关系。



 项目

项目 项目

大数据 数据湖

结构化数据

传统关系型

MPP应用

数据库管理

系统

非结构化数据

大数据

图4-1 元数据管理拓扑图e

**2.元数据管理工具** **Apache** **Atlas**

Apache Atlas 是一个可伸缩和可扩展的元数据管理工具与大数据治理服务，其设计的 目的是为了与其他大数据系统组件交换元数据，改变以往标准各异、各自为战的元数据管 理方式，构建统一的元数据库与元数据定义标准，并且与Hadoop生态系统中各类组件相集 成，建立统一、高效且可扩展的元数据管理平台。

对于需要元数据驱动的企业级Hadoop 系统来说， Apache Atlas 提供了可扩展的管理 方式，并且能够十分方便地支持对新的商业流程和数据资产进行建模。其内置的类型系统 (Type System) 允许 Atlas 与 Hadoop 大数据生态系统之内或之外的各种大数据组件进行元 数据交换，这使得建立与平台无关的大数据管理系统成为可能。同时，面对不同系统之间 的差异以及需求的一致性问题，Atlas 都提供了十分有效的解决方案。

O <https://zh.hortonworks.com/apache/atlas/>

第4章 大数据治理之 Apache Atlas 129

Atlas 能够在满足企业对 Hadoop 生态系统的预设要求的条件下，高效地与企业平台的 所有生态系统组件进行集成。同时，Atlas 可以运用预先设定的模型在 Hadoop 中实现数据 的可视化，提供易于操作的数据审计功能，并通过数据血统查询来丰富企业的各类商业元 数据。它也能够让任何元数据消费者与其相互协作而不需要在两者之间构建分离的接口。 另外，Atlas 中的元数据的准确性和安全性由 Apache Ranger 来保证， Ranger 能够在运行时 阻止那些不具备权限的数据访问请求。

**3.大数据治理的核心治理服务**

Apache Atlas 使得企业能够有效且高效地解决各类需求的原因在于，它提供了大数据治 理中可扩展的核心治理服务，这些服务具体如下。

1)元数据交换：允许从当前的组件导入已存在的元数据或模型到Atlas 中，也允许导 出元数据到下游系统中。

2)数据血统：Atlas 在平台层次上，针对 Hadoop 组件抓取数据血统信息，并根据数据 血统间的关系构建数据的生命周期。

3)数据生命周期可视化：通过Web 服务将数据生命周期以可视化的方式展现给客户。

4)快速数据建模： Atlas 内置的类型系统允许通过继承已有类型的方式来自定义元数 据结构，以满足新的商业场景的需求。

5)丰富的 API: 提供了目前比较流行且灵活的方式，能够对 Atlas 服务、HDP(Horton- work Data Platform) 组件、UI 及外部组件进行访问。

**4.Apache** **Atlas** **的主要特性**

(1)数据分类

1)Atlas 提供了导入或定义数据注释的功能，这些数据注释可以根据具体的商业业务 分类来定义。通过这些分类后的数据注释，可以实现数据分类的功能。

2)Atlas 提供了定义、添加注释以及自动获取数据集与基础元素之间关系的功能，这 些基础元素包括数据源、数据目标及其衍生的过程。

3)向第三方系统导出元数据。

(2)集中审计

1)对于每一个访问数据的应用以及交互过程，Atlas 会抓取其安全访问信息。

2)对于每一个执行的操作活动及其具体步骤， Atlas 能够将这些操作信息抓取下来。 (3)搜索与数据血统

1 ) 在Atlas 中，用户可以预先定义访问路径，并通过这些路径来浏览数据分类与数据 审计的信息。

2)用户利用Atlas 全文搜索这一特性，可以快速与准确地定位相关数据及审计事件。

3)可视化的数据血统允许用户深入挖掘数据具体的来源、操作方式以及安全策略等整 个数据生命周期中的各类信息。

130 第二篇 开源实现篇

(4)安全与策略引擎

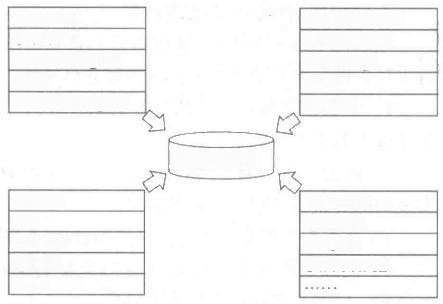
1)基于数据分类的计划、属性和角色，Atlas 使得数据管理策略间的关系更加合理化。 2)通过数据分类，Atlas 也支持自定义策略以防止数据不适当地衍生。

3)通过数据表项中的值或者属性，Atlas 支持对数据表中的列或者行添加标签。

**4.1.2 Apache Atlas 发展近况**

为了实现对元数据进行管理，目前行 业中已经有许多数据管理工具，如图4-2 所示。围绕元数据管理的各项具体内容， 已经有若干建模工具、数据集成工具与 BI (Business Intelligence) 工具。

在以往的元数据管理系统中，各类建 模工具与数据集成工具并存，各自独立， 且遵循的标准也不尽相同。这就给元数据 收集、交换与管理带来了诸多问题。因此， 定义统一 的元数据标准，建立集中化的元 数据库有着十分重要的意义，而 Apache Atlas 最初就是为了这 一 目的而设计的。



建模工具

ERwin

PowerDesigner

Rose

Oracle Designer

元数据管理

BI工具

BO

Cognos

MSTR

…

数据集成工具 PowerCenter DataStage

SAS

Sagent

元数据源

RDBMS

EXCEL

CWM XML

图4-2 各类元数据管理工具

**1.Apache** **Atlas** **发展概述**

Apache Atlas 由 Hortonworks 公 司 的 Seetharam Venkatesh 发起，其作为首席架构师， 负 责Atlas 开发的一切事宜，而Andrew Ahn则作为 Apache Atlas 的项目经理，负责管理与 宣传 Atlas。

参与开发Atlas 的企业与组织众多，包括如图4-3所示的 Hortonworks 、SAS 等公司。



Ssas

**Schlumberger**

**O TARGET**

**aetna"**

MERCK

**意急急**

**Hortonworks**

**Global Financial**

**Company**

图4-3 参与Atlas 开发的企业和组织θ

Apache Atlas 最初的0.5版本于2015年7月发布，之后于2015年年末发布了0.6版本。 截止到2016年5月， Atlas 已升级为Apache 的顶级开源项目，并在2016年4月25日发布 了 Atlas 0.7版本，在GitHub 上有1047个开源贡献，有23位主要贡献者。Atlas 作为 HDP 中大数据治理的核心组件，随着 HDP 的开发而发展起来。HDP 的发展历程如图4-4所示。

θ <htps://www.apache.org/foundation/thanks.html>

第4章 大数据治理之 Apache Atlas 1 31

由于Apache Atlas 出色的运算性能和丰富的特性，它已经成为当前十分有前景的大数 据治理的元数据管理框架。它提供了多种方式与各种大数据组件相互协作，包括在0 . 5版本 中提供了两种方式与Hive交互，在0.6版本中提供了与Kafka 、Sqoop 和Storm 的协作功 能，并且在未来会提供与更多第三方组件进行协作的途径来提高 Atlas的大数据治理能力。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2014年12月 | 2015年5月 | 2015年7月 | 2016年1月 | 2016年夏季 |
| DGI group | Apache | HDP 2.3 | HDP 2.4 | HDP 2.5 |
| Kickoff | Atlas  Incubation | Foundation GA Release | Kafka/Storm Falcon/Sqoop | Business Catalog |

Tag Based Secerity

图4 - 4 HDP 的发展历程e

**2.Apache** **Atlas** **各版本功能特性简介**

目前最稳定的 Atlas 版本致力于提供可扩展的元数据服务，该服务能够结合具体的商业 术语，对任意商业过程进行建模，同时致力于提高从其他系统或组件导入和导出元数据的 效率，并扩展其使用范围。

对于Apache Atlas 0.7 版本，其功能特性如表4-1所示。

**表4-1** **Apache** **Atlas** **0.7** **功能特性**

|  |  |  |
| --- | --- | --- |
| **Apache** **Atlas版本** | **功** **能** **特** **性** | |
| **Apache** **Atlas** **0.7** | 企业级部署  支持高可用  支持组件血统  支持新组件Falcor  支持Kerberos  支持基于标签的动态安全策略 提供用户接口  支持商业业务分类 | 性能进一步增强  支持新组件Kafka  支持新组件Storm  支持新组件Scoop  提供安全保护  支持与Ranger整合  完善GUI  符合Governance-ready生态标准 |

对于Apache Atlas 0.6版本，其功能特性如表4-2所示。

**表4-2** **Apache** **Atlas** **0.6** **功能特性**

|  |  |
| --- | --- |
| **Apache** **Atlas版本** | **功** **能** **特** **性** |
| Apache Atlas 0.6 | **内置数据类型支持HDFS**  **支持元数据标签管理**  **扩展了对Hive的支持** |

对 于 Apache Atlas 0.5 版本，其功能特性如表4- 3所示。

θ <https://zh.hortonworks.com>

132  第二篇 开源实现篇

**表4-3** **Apache** **Atlas** **0.5** **功能特性e**

|  |  |
| --- | --- |
| **Apache** **Atlas版本** | **功** **能** **特** **性** |
| **Apache** **Atlas** **0.5** | **支持可扩展的元数据服务**  **支持业务单元级别的建模**  **扩展了对HDFS、Hive数据库、表、列的可视化**  **支持以更为灵活的方式访问Atlas服务**  **整合Hive管理已存在的元数据**  **支持对元数据的导入与导出**  **支持直接获取SQL运行时数据**  **支持查看Hive表的血统以及特定领域的搜索**  **支持关键字、随机字符等多种搜索方式** |

**3.Apache Atlas 的设计与开发原则**

Atlas 是围绕如下两点原则来进行开发的。

(1)元数据在 Hadoop 生态系统中至关重要

Atlas 实现了对于Hadoop 生态系统的可视化，通过使用内置的连接器与 Hadoop 组件 进行连接，提供了十分专业且易于操作的跟踪功能，这一功能能够跟踪丰富且多样的商业 分类元数据。通过允许元数据消费者共享元数据库，Atlas 使得交换元数据变得十分便捷， 同时也使得元数据生产者之间的交互变得十分容易。

(2)保证开发过程的开源状态

来 自Aetna 、Merck 、SAS 等公司或组织的工程师一同协作，确保 Apache Atlas 是为 解决大数据治理中商业领域的实际问题而构建的。这一方式同时也成为开源社区的典型范 例，帮助软件产品能够快速成型，帮助企业将时间转化为价值，并服务于将数据放在首位 的企业。

**4.Apache** **Atlas** **应用广泛**

Hortonworks 作为设计与开发 Apache Atlas 的公司，自2011年获得雅虎投资创立以来便 受到了极大的关注，并且一直专注于Apache Hadoop生态系统的开发和支持。Hortonworks 与Cloudera 和 MapR 两家公司不同，它完全开放其所有产品，并将核心技术放到开源社 区之中。其中最为重要的核心产品就是 Hortonworks Data Platform(HDP) 和 Hortonworks Data Flow(HDF)。

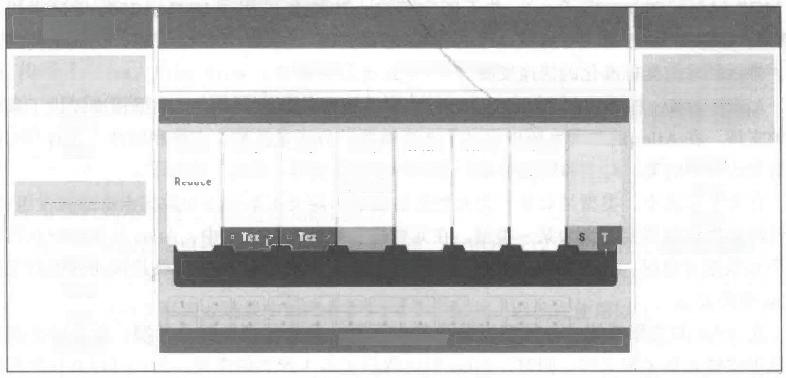
元数据管理框架 Apache Atlas 作为大数据治理的核心框架，已经被集成到HDP的大数 据治理模块中。HDP 的架构图如图4-5所示，中间底部是 HDFS 分布式文件系统，通过分 布式调度工具YARN与中间的若干数据访问(Data Access) 组件进行协作，右侧的 Ranger 与Atlas 配合，为大数据平台提供安全的访问机制，左侧则是大数据治理的若干组件，其中 核心的数据生命周期与元数据管理组件由Atlas 来承担。

随着 Hortonworks的 发 展 ，HDP 的客户不断增加，其作为解决海量数据的通用处理平

θ <https://zh.hortonworks.com/apache/atlas/#section>\_5

第4章大数据治理之Apache Atlas 133

台，已被广泛应用到医疗、教育、交通等多个行业。Apache Atlas 除了被集成到 HDP 中得 以广泛应用外，国内许多著名公司如中兴通讯公司等，也将其集成到自己的产品中。可以 预见，随着HDP的广泛应用，Apache Atlas 会不断地获得更多的市场占有率，这一优势将 会更加有力地推动大数据治理的发展。



GOVERNANCE

INTEGRATION

Data Lifecyele &

Governance

Falcon

Atlas

Ranget

Data workflow

Atlas

Sqoop

HDFS Eneryption

Flume

**YARIH:Date Operating System**

Kafka

NFS

WebHDFs

HDFSHadoop Distributed File System

**DATA MANAGEMENT**

NoSql

HBase

Accumulo

Phoenix

Slider

Administration Authentication Authorization

Auditing Data Protection

Others

HAWQ Partners

In-Mem

Spark

Batch

ap

Script

Pig

Search

Solr

Stream

Storm

Ambari Uset Views

Sql

Hive

DATA ACCESS

Zeppelin

SECURITY

T⁰0LS

Knox

图4-5 HDP 架构图e

**4.1.3** **Apache** **Atlas** **技术优势**

在最近这一次大数据浪潮来临之前，人们就已经认识到，高效且安全地管理数据系统 的关键在于对元数据进行科学有效的管理，因此在元数据管理的理论研究与实际应用中， 众多相关组织提出了各自的元数据标准。国内外企业与开源社区也有自己相应的元数据管 理工具，但这些元数据工具所遵循的标准各异，这使得进行元数据高效且可靠的交换变得 难以实现。并且由于元数据管理工具的设计理念不同，其针对元数据的管理方式与提供的 功能也有极大的不同，这为大数据技术及其应用的发展带来了诸多障碍。

为了解决大数据治理中最为核心的元数据管理问题，Apache Atlas 从理念的提出到具体 设计与开发，都致力于定义统一的元数据标准，建立高效的元数据交换体系，提供友好的 商业业务定义接口，获取主流大数据组件元数据信息，提供可视化的血统查询显示与数据 审计功能。这些特点都成为Apache Atlas 的优势，能够为企业大数据治理的实际应用提供 十分有力的支持。

**1.定义统一的元数据标准**

元数据的标准大致可以分为两类： 一类是指元数据建模，即对将来的元数据的建模规 范进行定义，使得元数据建模的标准在制定之后，所产生的元数据都以统一的方式建模和组

O <https://zh.hortonworks.com/products/data-center/hdp/>

134 第二篇 开源实现篇

织，从而保证了元数据管理的一致性。另一类是指元数据的交互，是对已有的元数据组织 方式以及相互交互的格式加以规范定义，从而实现不同组件、不同系统之间的元数据交互。 为了推动元数据标准化的发展，对象管理组织 (Object Management Group,OMG)

和元数据联合会 (Meta Data Coalition) 都致力于统一元数据标准。OMG 在1995年采用 了MOF(Meta Object Facility),并不断完善之；1997年采用了UML;2000 年又采用了 CWM 。UML 、MOF和 CWM 这3个标准形成了OMG 建模和元数据管理、交换结构的基 础，推动了元数据标准化的快速发展。

Apache Atlas 核心中的 Type System(类型系统)为定义统一的元数据标准提供了最重 要的支持。在 Atlas 的类型系统中定义了3个概念，分别是类型、实体和属性。若将其与面 向对象语言中的类、对象和属性类比，这3个概念就变得十分易于理解了。

在类型系统中，类型是对某一类元数据的描述，定义了某一类元数据由哪些属性组成， 属性的属性值也需要定义为某一类型。在元数据管理的实际应用中， Atlas 从数据源获取某 一个元数据对象时，会根据其隶属的类型建立相应的实体，这个实体就是该元数据对象在 Atlas 中的表示。

在Atlas 的类型系统中，元型可分为基本元型、集合元型、复合元型，所有的类型都 是基于这些元型来定义的。同时， Atlas 中也提供了若干预置的类型，用户可以直接使用这 些类型，或者通过继承的方式来复用这些类型。正是由于所有类型的背后都是统一的元型， 并且所有类型都是继承自某些预置的类型，这实际上就给元数据对象的建模定义了标准。 这样统一的规范和标准使得高效且可靠的元数据交换成为可能。

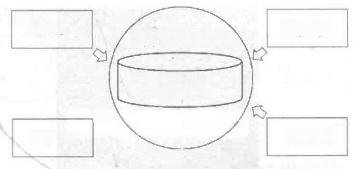
**2.高效的元数据获取与交换体系**

为了建立可扩展、松耦合的元数据管理体系， Apache Atlas 支持多种元数据获取方式， 并且针对大数据生态系统中的不同组件，其元数据的获取方式是相互独立的，这就满足了 大数据系统高内聚和低耦合的要求。另外，Apache Atlas 的元数据库是唯一的，统一的元数 据库保证了元数据的一致性，减少了元数据交换过程中不必要的转换，使不同组件之间的 元数据交换高效而稳定。

Apache Atlas 获取元数据包括 Batch 批处理和 HOOK 两种方式。对于通过 Batch 批处 理的方式获取元数据，目前Atlas 0.7只支持Hive。在该方式中， Atlas 允许用户执行某一脚 本获取相应组件的元数据信息，将该组件的元数据信息更新到元数据库中。即，当用户不 执行获取元数据的脚本时，相应组件的数据变更不会导致 Atlas 元数据库中的信息变更；当 用户执行获取元数据的脚本时，相应组件中若存在数据的变更， Atlas 就会将其所有新增的 元数据信息存入元数据库中。

对于通过HOOK 的方式获取元数据，目前Atlas 0.7支 持Hive 、Falcon 、Sqoop 和 Storm 这4个组件，针对每一种组件，都有相应的HOOK, 用户可以根据自身需要针对不 同组件对 HOOK 进行配置。当配置完成后，相应组件的 HOOK 会监听该组件的各种操 作， 一旦该组件的状态发生变化，HOOK 会自动创建相应的元数据对象，并发送给 Atlas

第4章 大数据治理之 Apache Atlas 135



消息通知系统

元数据库

*口’*

Falcon

图4-6 Atlas获取元数据架构图

Sqoop

Storm

Hive

处理。

各类组件的HOOK与 Atlas是通过消息通知系统来交互的。图4-6展示了Atlas 获

取元数据的架构图， Atlas 0.7支 持Hive 、Sqoop、

Falcon 和 Storm 作为数据源，并使用Kafka 作为消

息通知系统 (Notification) 。 即不同组件只需要与

Kafka 进行交互，再由Kafka 将元数据对象封装成

消息传递给 Atlas 。当向 Atlas 元数据管理系统中添

加新的大数据组件时，只需要将遵循 Kafka 规范的

HOOK 添加到系统之中，即可让Atlas 对这一新的

组件进行管理，从而满足了元数据管理系统的高扩展性要求。

**3.允许针对不同商业对象进行元数据建模**

以往的元数据管理组件考虑了用户的诸多需求，为用户设计了诸多的元数据类型，但这 种设计思想往往也限制了元数据管理组件的应用。因为不管元数据管理组件的设计者如何 高明，也难以概括实际商业场景中涉及的所有元数据对象，因此在使用以往的元数据管理组 件时，用户常常会遇到实际商业场景中的元数据对象与组件提供的建模模型不匹配的情况， 只能选择近似的类型对实际场景中的元数据对象进行建模，这使得元数据的管理极为不便。

但 Apache Atlas 有所不同，它提供了若干的预置类型，这些类型的背后也定义了统一 且易于复用的元数据对象的元型，并且允许用户通过继承的方式来创建符合实际需求的元 数据类型，这就极大地满足了用户对于不同商业对象进行建模的需求，解决了其他元数据 管理组件难以匹配所有商业场景中元数据对象的难题。

**4.可视化的数据血统与生命周期**

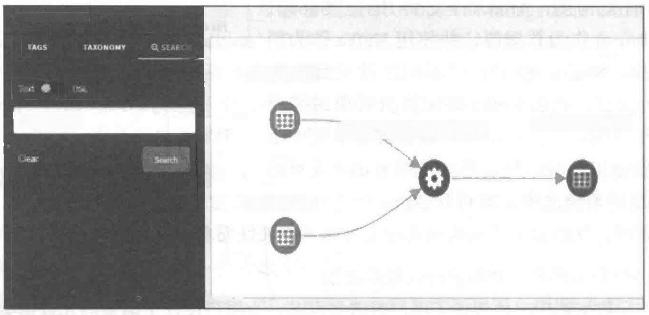
Apache Atlas 能够通过批处理或者HOOK 的方式从元数据源获取元数据信息，前者需 要用户手动运行脚本来执行，后者则会自动监听相应组件的各类操作。无论采取怎样的方 式，从各类组件获取的元数据对象是十分丰富与多样的，包括采集数据的数据源和采集方 式，被采集数据的结构，数据的状态变化及其相应操作，以及数据最后被删除等各种元数 据对象信息。这些信息都会被包装成相应的元数据类型，并生成对应的元数据实体，通过 消息通知系统发送给 Atlas 并存储到元数据库中。

但Atlas 并不是简单地将这一系列的元数据信息直接存入元数据库中，而是将它们之间 的关系也存入元数据库中。同时，为了更好地表示元数据之间的关系，Atlas 在其Web UI 中提供了对于数据血统的可视化显示，能够为用户提供直观且明晰的数据生命周期图像， 使得用户从一幅数据血统图中就能够了解数据从进入大数据系统开始，到中间经历各种变 化，到最后从大数据系统中消亡的整个生命周期。

Atlas 提供Web 页面的方式来查看与管理血统，如图4-7所示。在该 Web页面的左侧 提供了标签、数据分类与搜索的功能，对于搜索又提供了Text文本与DSL 两种搜索方式。

136 第二篇 开源实现篇

右侧显示搜索到的元数据信息及其血统 (LINEAGE), 并以图的形式形象生动地展示出数据 的生命周期。

Apache Atlas

UNEAGE

**employee\_2**

create table employ …

employee\_union

employee\_union

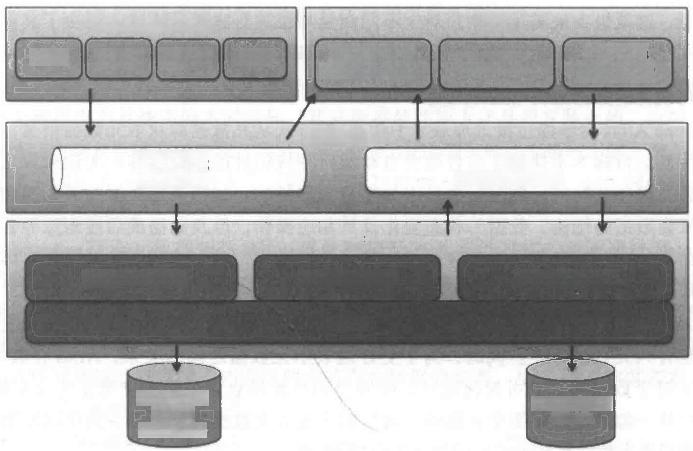
employee\_1

图4- 7 Atlas 数据生命周期示意图

**4.1.4** **Apache** **Atlas** **架构**

**1.架构简介**

Apache Atlas 的各组成部分的架构图如图4-8所示。



Apps

Admin U]

Integration

API

<HTTP/REST>

Core

**Type System**

**Titan**

**Metadata**

**Store**

**<HBase>**

Ranger Tag

Based Policies

Metadata

Sqoop

Sources

Falcon

Messaging

<Kafka>

Business Taxonomy

**Index Store <Solr>**

**Graph Engine**

**Ingest/Export**

Storm

Hive

图 4 - 8 Apache Atlas架构图2

θ <http://atlas.apache.org/Architecture.html>

第4章大数据治理之Apache Atlas 1 37

(1)元数据源 (Metadata Sources)

Atlas 支持与多种数据源相互整合，在未来会有更多的数据源被整合到Atlas 之中。目 前 ，Atlas 0.7 版本导入与管理的数据源有 Hive 、Sqoop 、Falcon 、Storm。

这意味着：在Atlas 中定义了原生的元数据模型来表示这些组件的各种对象； Atlas 中提供了相应的模块从这些组件中导入元数据对象，包括实时导入 (HOOK) 和批处理 (Batch) 导入两种方式。

(2)应用简介 (Apps)

在Atlas 的元数据库中存储着各种组件的元数据，这些元数据将被各式各样的应用所使 用，以满足各种现实业务与大数据治理的需要。

1)Atlas 管理界面：作为其中的一个应用是基于Web UI方式的，它允许管理员与数 据科学家发现元数据信息和添加元数据注解。在诸多主要的功能中， Atlas 提供了搜索接 口与类 SQL 语言，这些特性在 Atlas 的架构中扮演着十分重要的角色，它们能够被用于查 询Atlas 中的元数据类型和对象。另外，该管理界面使用Atlas 的 REST API来构建它的 功能。

2)基于各种策略的标签验证：对于整合了诸多Hadoop 组件的 Hadoop 生态系统， Apache Ranger 是一个高级安全解决方案。通过与Atlas 整 合 ，Ranger 允许管理员自定义 元数据的安全驱动策略来对大数据进行高效的治理。当元数据库中的元数据发生改变时， Atlas 会以发送事件的方式通知 Ranger。

3)商业业务分类：从各类元数据源中导入Atlas 的元数据以最原始的形式存储在 元数据库中，这些元数据还保留了许多技术特征。为了加强挖掘与治理大数据的能力， Atlas 提供了一个商业业务分类接口，允许用户对其商业领域内的各种术语建立一个具有 层次结构的术语集合，并将它们整合成能够被Atlas 管理的元数据实体。商业业务分类 这一应用，目前是作为Atlas 管理界面的一部分而存在的，它通过REST API来与Atlas 集成。

(3)集成交互模块 (Integration)

Atlas 提供了两种方式供用户管理元数据。

1)API:Atlas 的所有功能都可以通过RESTAPI 的方式暴露给用户，以便用户可以对 Atlas 中的类型和实体进行创建、更新和删除等操作。同时， REST API也是 Atlas 中查询类 型和实体的主要机制。

2)消息 (Messaging) 系统：除了RESTAPI, 用户可以选择基于Kafka 的消息接口来 与Atlas 集成。这种方式有利于与Atlas 进行元数据对象的交换，也有利于其他应用对 Atlas 中的元数据事件进行获取和消费。当用户需要以一种松耦合的方式来集成 Atlas 时，消息系 统接口变得尤为重要，因为它能提供更好的可扩展性和稳定性。在Atlas 中，使用Kafka 作 为消息通知的服务器，从而使得上游不同组件的钩子 (HOOK) 能够与元数据事件的下游消 费者进行交互。这些事件被 Atlas 的钩子所创建，并冠以不同的 Kafka 主题。

138 第二篇 开源实现篇

(4)核心 (Core) 模块

在 Atlas 的架构中，其核心组成部分为其核心功能提供了最为重要的支持。

1)类型系统 (Type System):Apache Atlas 允许用户根据自身需求来对元数据对象进 行建模。这样的模型由被称为“类型” (Type) 的概念组成，类型的实例被称为“实体” (Entity), 实体能够呈现出元数据管理系统中实际元数据对象的具体内容。同时， Atlas 中的 这一建模特点允许系统管理员定义具有技术性质的元数据和具有商业性质的元数据，这也 使得在 Atlas 的两个特性之间定义丰富的关系成为可能。

2)导入/导出 (Ingest/Export):Atlas 中的导入模块允许将元数据添加到 Atlas 中，而 导出模块将元数据的状态暴露出来，当状态发生改变时，便会生成相应的事件。下游的消 费者组件会获取并消费这一事件，从而实时地对元数据的改变做出响应。

3)图引擎 (Graph Engine): 在 Atlas 内部， Atlas 使用图模型(一种数据结构)来表示 元数据对象，这一表示方法的优势在于可以获得更好的灵活性，同时有利于在不同元数据 对象之间建立丰富的关系。图引擎负责对类型系统中的类型和实体进行转换，并与底层图 模型进行交互。除了管理图对象，图引擎也负责为元数据对象创建合适的索引，使得搜索 元数据变得更为高效。

4)Titan: 目 前 ，Atlas 使 用Titan 图数据库来存储元数据对象。Titan 图数据库使 用两个数据库来存储数据，分别是元数据库和索引数据库。默认情况下，元数据库使用 HBase, 索引数据库使用Solr 。同 时 ，Atlas 也允许更改相应配置文件，将 BerkeleyDB 和 ElasticSearch 作为其元数据库和索引数据库。元数据库的作用是存储元数据，而索引数据 库的作用是存储元数据各项属性的索引，从而提高搜索的效率。

**2.类型系统**

类型系统作为Atlas 中的一个重要组件，允许用户定义与管理各种类型与实体。对于 Atlas 自身系统之外的所有元数据对象(例如 Hive表格),也是通过类型来进行建模的，然 后以实体的方式来呈现。为了在 Atlas 中存储一种新的元数据类型，有必要清楚地了解类型 系统这一概念。

(1)类型

在Atlas 中，类型这一概念表征了一类特别的元数据对象是以怎样的方式来存储和访问 的。具体来说， 一个类型表示一个或一系列属性，这些属性定义了元数据对象的各种特性。

对于具有编程背景的用户来说，在Atlas 中的类型与面向对象编程中的“类”(Class) 有 着十分相似的地方，与关系型数据库中的表模式 (Table schema) 也有相似之处。

Atlas 提供了诸如 Hive Table等许多原生的类型，它通过如下属性定义Hive Table:

|  |
| --- |
| Name:hive\_table  MetaType:Class  SuperTypes:DataSet |

第4章 大数据治理之 Apache Atlas 139

|  |
| --- |
| Attributes:  name:String(name of the table)  db:Database object of type hive\_db  owner:String  createTime:Date  lastAccessTime:Date  comment:String  retention:int  sd:Storage Description object of type hive\_storagedesc  partitionKeys:Array of objects of type hive\_column  aliases:Array of strings  columns:Array of objects of type hive\_column  parameters:Map of String keys to String values  viewOriginalText:String  viewExpandedText:String  tableType:String  temporary:Boolean |

从如上所示的类型定义中，可以知道，在Atlas中，类型是通过属性name 进行唯一 标识的；每一种类型都包含一种元型 (Metatype), 用来表示其在Atlas 中的建模类型。在 Atlas 中主要有4类元型。

口基本元型，例如 Int 、String 、Boolean 等。

口枚举元型。

口集合元型，例如Array、Map。

口复合元型，例如 Class、Struct、Trait 等。

类型可以通过继承的方式从父类型 (Supertype) 处获得在父类型中定义的所有属性。这 一机制使得用户可以将相关联的类型的共有属性抽取出来定义为新的类型。这与面向对象 语言中的定义父类十分类似。同时，在 Atlas 中也支持多继承。在本例中，类型 hive\_table 继承自 DataSet 的类型，该类型是被预先定义的。

元型为Class 、Struct 或 Trait 的类型可以拥有一系列的属性，每一个属性都有属性名， 可以通过“类型名.属性名”的方式来调用这些属性。需要注意的是，属性本身也是通过 Atlas元型来定义的。在本例中， hive\_table.name 是一个String 类 型 ，hive\_table.aliases 是 一个 String 数组， hive\_table.db 表示类型 hive\_db 的一个实例。

Atlas 允许用户通过“类型名.属性名”的方式来调用属性，这使得用户可以在两种类 型之间定义任意的关系并建立丰富的模型。需要注意的是，类型中的属性也可以是某个类 型的集合。在本例中， hive\_table.columns 表示的是类型hive\_table 中由类型 hive\_column 组成的列表。

(2)实体

Atlas 中的实体是指一个具体的值或者一个Class 类的实例，因此，它表示了现实世 界中的某一个具体的元数据对象。类似地，在面向对象语言中， 一个实例是指一个类的

140 第二篇 开源实现篇

对象。

下面以 Hive Table 为例来具体说明实体的概念。下面所示的 Hive Table 是类型 hive\_ table 的实体，由于它是hive\_table 这 一Class 类的实例，因此它拥有hive\_table 的每一个属 性，并且每一个属性都有具体的值。

id:"9ba387dd-fa76-429c-b791-ffc338d3c91f"

typeName:"hive\_table"

values:

name:"customers"

db:"b42c6cfc-cle7-42fd-a9e6-890e0adf33bc"

owner:"admin"

createTime:"2016-06-20T⁰6:13:28.000Z"

lastAccessTime;"2016-06-20T06:13:28.000Z"

comment:null

retention:0

sd:"ff58025f-6854-4195-9f75-3a3058dd8dcf"

partitionKeys:null

aliases:null

columns:["65e2204f-6a23-4130-934a-9679af6a211f","d726de70-faca-46fb-9c99- cf04f6b579a6",...]

parameters:{"transient\_lastDdlTime":"1466403208"}

vieworiginalText:null

viewExpandedText:null

tableType:"MANAGED\_TABLE"

temporary:false

从上面这一例子中可以看到：

1)实体作为每 一个Class 类型的实例，通过GUID 来唯一地标识。当对象被建立时， GUID由 Atlas 服务器产生，并且与实体的生命周期保持一致。无论何时，只要通过GUID, 用户都可以访问任何特定的实体。在本例中，名为“customers” 的 hive\_table 通过唯一的 GUID“9ba387dd-fa76-429c-b791-ffc338d3c91f” 来进行标识。

2)每一个实体都属于某一类型，并且需要在实体的定义中指明具体类型的名称。在本 例中，名为 “customers” 的 Hive Table 是一个 hive\_table 的实体。

3)实体的值是指由属性与属性值组成的 map, 这些属性在实体所属的类型中定义。 4)实体中每一个属性值的类型是由其属性的元型来决定的。

口在本例中，基本元型的例子如‘name’=‘customers',‘temporary’=‘false’。

口在本例中，集合元型的例子如 parameters={“transient\_lastDdITime”:“1466403208”}。

口复合元型：对于Class 类型的实体，其属性值通常是另一个与其相关联的实体。在

本例中，名为 customers 的 hive\_table 实体存在于默认数据库中。hive\_table 实体与 数据库的关系将通过名为db 的属性来确定，因此，属性 db的值是能够唯一标识默 认hive\_db 的 GUID 值 。

第4章 大数据治理之 Apache Atlas 141

从Atlas 中的实体的概念可以知道Class 元型和Struct 元型的区别。尽管它们都是以其 他类型作为其属性的方式来组成的，但是，Class类型的实体拥有ID 属性(其属性值通常是 GUID 值),并且能够通过该 GUID值指向其他实体(本例中，通过GUID 从 hive\_table 的实 体指向了hivedb 的实体),而Struct 类型的实例并不具有ID 属 性 ，Struct 类型的值仅仅是 一系列属性的集合，这些属性被嵌入其实体自身中。

(3)属性

通过上文的介绍可知，属性是被定义在诸如Class 和 Struct 等元型之中的，并且属性可 能会被简单地认为仅仅只有名字和元型的值。但是，在Atlas 中的属性还具有其他更多的特 性，其作用是定义更多的概念来与类型系统关联。

一个属性通常有如下特性：

|  |
| --- |
| name:string  dataTypeName:string  isComposite:boolean  isIndexable:boolean  isUnique:boolean  multiplicity:enum  reverseAttributeName: string |

这些特性的含义如下。

1)name: 表示属性的名字。

2)dataTypeName: 表示属性的元型名称(包括原生元型、集合元型和复合元型)。

3)isComposite:

●如果某一属性被定义为 composite,表示该属性不能独立于包含它的实体而存在。 在 hive\_table 的例子中，属性 columns 很好地说明了该特性的含义。因为属性 columns 作 为hive\_table 的一部分，且不能独立于hive\_table 而存在，所以属性 columns 被定义为 composite 的属性。

●对于某一 composite 属性，它必须随着包含它的实体的创建而创建。例如一个

hive\_column 必须随 hive\_table 的创建而创建。

4)isIndexable: 表示该属性能否支持索引。当支持索引时，能够通过属性值来进行查 找，从而提高查找效率。

5)isUnique:

●该特性也与索引有关，表示属性是否唯一。属性被定义为唯一时，表示该属性在 Titan 图数据库中会建立唯一的索引。

●当某属性的这一特性被设置为true时，该属性将为作为实体的主键(Primary Key), 从而与其他实体相区别。因此，在建模时要确保现实世界中的某属性是唯 一的，才能将该属性的这一特性设置为true。

在上面的例子中，对于hive\_table 的 name 属性，它在 Atlas 中不是唯一的属性，因为

142 第二篇 开源实现篇

同名的数据表可以存储在不同的数据库中，甚至在多个集群中，数据库名和表名都可以相 同。只有对于单集群的情况，数据库名才应该被设定为唯一的。

6)multiplicity: 表示该属性是必需的还是可选的，或者是两者皆可的。在创建某一实 体时，如果该实体的属性的multiplicity 不符合该特性在类型中的声明，那么将导致该实体 创建失败。因此，这一特性被用于定义一些针对元数据的约束条件。

在上面的例子中，可以将hive\_table 的其他属性扩展开，比如 hive\_table 的属性 db表 示其属于某一个数据库，展开 db 后得到的特性如下所示：

|  |
| --- |
| "dataTypeName":"hive\_db"  "isComposite":false  "isIndexable":true  "isUnique":false  "multiplicity":"required"  "name":"db"  "reverseAttributeName":null |

从上面的特性中看到， multiplicity 的值为 required, 表示对于一个hive\_table 的实体， 它必须属于某一个数据库。

columns 的特性如下所示：

|  |
| --- |
| "dataTypeName":"array<hive\_column>"  "isComposite":true  "isIndexable":true  "isUnique":false  "multiplicity":"optional"  "name":"columns"  "reverseAttributeName":null |

从上面的特性中可知，isComposite 的值为true, 表示该 columns 实体必须依赖于 table 实体，不能独立于table 实体而存在。

从上面的描述和例子中读者将能够了解到，属性的定义可以影响具体的建模行为，包 括索引、各类约束等。

(4)系统预置类型及其意义

Atlas 自带了一些预置的系统类型，下面将会介绍一些预置的系统类型，以及它们的 意义。

1)Referenceable: 该类型包含了能够通过唯一的属性来搜索的实体，这些唯一的属性 被称为 qualifiedName.

2)Asset: 该类型包含name 、description 和owner 等属性，其中 name 是一个必备的属 性 (multiplicity=required),而其他属性是可选的。Referenceable和Asset类型的意义在于， 使得用户在定义与查询实体时，强化实体的一致性。

3)Infrastructure: 该类型继承自Referenceable 和 Asset, 它的主要作用是被作为基础

第4章 大数据治理之Apache Atlas 143

设施元数据对象的共同父类型，这些元数据对象包括 cluster 、host 等。

4)Data Set: 该类型继承自Referenceable 和 Asset。从概念上说，它被用于表示一个能 够存储数据的类型。在 Atlas 中 ，hive\_table 和 Sqoop RDBMS table 等类型都是继承自 Data Set的。继承自Data Set 的类型都有一个 Schema, 从某种意义上说，这些类型可以通过拥 有一个属性来定义 Data Set的若干属性。例如， hive\_table中的 columns 属性就是这样。另 外，继承自Data Set的类型的实体可以参与数据的变换，并且这种变换能够被 Atlas 的血统 图获取。

5)Process: 该类型继承自 Referenceable 和 Asset, 从概念上说，它被用于表示任何一 个数据变换操作。例如，对于一个将某一 Hive Table 转换为另一个 Hive Table 的 ETL 过程 来说，该过程可以用一个继承自Process 的类来表示。Process 类型拥有两个具体的属性， 分别是 inputs 和 outputs,inputs 和 outputs 都是由 Data Set的实体所组成的数组。因此， Process 类型的实例能够通过输入和输出来获取 Data Set的血统。

**4.2 Apache Atlas 的配置与使用**

在4.1节介绍了Apache Atlas的技术概况、发展情况、优势以及架构，读者应该对 Atlas已经有了初步的了解。本节将会从Atlas 的源码编译、部署、运行以及HOOK 的安装 与使用等角度来介绍 Atlas, 帮助读者更好地了解与使用Atlas。

Atlas 的运行依赖于很多组件，如本书安装的Atlas 依赖于HDFS 、YARN 、Zookeeper、 HBase 、Solr和 Kafka, 要保证Atlas 正常运行，就要保证其基础组件能够正常提供服务。 Atlas 使用 Titan 作为图数据库，而Titan 的数据引擎可以使用 BerkeleyDB 或 HBase, 它的 搜索引擎可以使用ElasticSearch或者 Solr, 本书安装的 Atlas 使用HBase 和 Solr 作为 Titan 的数据引擎和搜索引擎。Kafka 是一种高吞吐量的分布式发布订阅消息系统，它能够帮助 Atlas完成与一些组件进行元数据交换的功能。HBase 、Solr 、Kafka 的正常运行则需要以 HDFS 、YARN 、Zookeeper 为基础。

**4.2.1** **安装配置Apache Atlas**

**1.先决条件**

(1)Java

Atlas使用Java 语言编写，运行在Java 虚拟机 (JVM) 中，无论是编译还是安装都需 要在系统中安装 Java7 或者更高的版本。考虑到后期的开发需求，建议读者直接从 Oracle 官网上(<http://www.oracle.com/technetwork/java/javase/downloads/index.html>) 下载合适的 JDK版本安装，并将JAVA\_HOME环境变量的值设置为 Java 的安装目录，在PATH环境变 量中添加 Java 安装目录下的 bin 目录。安装完毕后执行如下命令检查JDK 是否已经被正确 安装，命令返回结果会因安装的 JDK 版本而异。

**144**  第二篇 开源实现篇

s javac -version

javac 1.7.0\_79

(2)Git

Atlas 的源码托管在GitHub 上，若通过GitHub 下载 Atlas 源代码需要安装 Git,Atlas 的源代码地址为<https://github.com/apache/incubator-atlas>, 若直接下载源代码，可忽略 Git 的安装。Apache Atlas的官方也提供了源码，其版本比 GitHub 落后，但也尽可能地避免了 使用最新代码可能出现的未知错误，能够保证其已有功能的稳定运行。若从官网直接下载 源码可以忽略Git 的安装。

以 CentOS 为例，可以通过以下命令安装 Git。

在 linc-1 上，使用root 用户在任意目录下执行如下命令：

$sudo yum git

Atlas 使用 Maven 编译，因此需要安装 Maven。若读者未安装 Maven, 请参考7.3.2节 完成 Maven 的安装。

此外Atlas 涉及的组件有Hadoop 、Zookeeper 、HBase 、Kafka 、Solr, 因此安装Atlas 之前需确保这些组件可以使用，并且需要确保这些组件没有使用Kerberos 进行鉴权， Hadoop 的介绍及安装在此不赘述。

法

意

目前 Atlas 与 Kerberos 的整合源码实现不是很完善。本书中第5章会介绍 Ranger 集 成Solr, 此时 Solr 必须配置 Kerberos,Atlas 向配置了Kerberos 的 Solr 服务认证身 份的方法在第5章具体介绍。

(3)Zookeeper 与 HBase

ZooKeeper 是一个分布式的、开放源码的应用程序协调服务，它为分布式应用提供一致 性服务，包括配置维护、域名服务、分布式同步、组服务等功能。HBase 、Kafka 与 Solr 都 依赖于Zookeeper 服务。

HBase(Hadoop Database) 是一个高可靠性、高性能、面向列、可伸缩的分布式存储 系统，利用HBase 技术可在廉价PC Server上搭建起大规模结构化存储集群。由于Atlas 的 Titan数据库采用 HBase 作为数据引擎，因此需要按照本书相关部分安装好HBase。

请参考5.2.5节来安装 Zookeeper 与HBase, 但不要安装该节中介绍的Ranger-HBase 插件。

(4)Kafka

Kafka 是一种高吞吐量的分布式发布-订阅消息系统，它提供了一种独特的消息系统功 能，它可以处理消费者规模的网站中的所有动作流数据。Hive 、Sqoop 等组件通过 Kafka 将 消息发送给 Apache Atlas, 完成元数据的导入， Kafka 的安装方法请参考5.2.7节，但不要 安装该节中介绍的Ranger-Kafka 插件。

(5)Solr

Solr 是一个高性能的搜索服务器，采用Java 5开发。它是基于Lucene 的全文搜索服务

第4章大数据治理之Apache Atlas 145

器，并对其进行了扩展，提供了比Lucene 更为丰富的查询语言，同时实现了可配置、可扩 展，并对查询性能进行了优化。它还提供了一个完善的功能管理界面，是一款非常优秀的 全文搜索引擎。Apache Atlas中的 Titan 采用 Solr 作为搜索引擎，完成元数据的搜索。读者 请根据5.2.10节完成 Solr的安装，但不要安装该节中介绍的Ranger-Solr 插件。

**2.编译** **Atlas**

(1)下载 Atlas 源码

Apache Atlas官网提供了稳定的版本，可以在 Apache Atlas 的官网找到项目源码，其官 网为： <http://atlas.incubator.apache.org/>, 将源码下载至linc-1 机器的/home/hdfs 目录下。

此 外 ，Atlas 是 GitHub 上的开源项目，由于Apache Atlas 源码在不断更新，其最新的 版本特性在此不列举，有需要的读者可以自行了解新版本添加的功能以及解决的 bug, 可 以 参 考 ：https:/github.com/apache/incubator-atlas/blob/master/release-log.txt 。 若选择下载 GitHub上最新的代码，可访问GitHub 的 Apache Atlas代码库：<https://github.com/apache/> incubator-atlas, 选择相应的源码版本分支，可以直接下载源代码的压缩包并解压到/home/ hdfs 目录下，或者使用Git 来克隆源代码。

在 linc-1 上，使用hdfs 用户，在/home/hdfs 目录下执行以下命令：

s git clone <https://github.com/apache/incubator-atlas.git>

完成之后可以在/home/hdfs 发现 Atlas 的源码文件。

(2)设置Maven 的 http 代理

Atlas 在编译的过程中会下载依赖的jar包，而依赖的下载源很多在国外。因为国内网 络环境的原因，如果没有http 代理，编译的过程会因为无法下载到依赖而失败。Maven 的 http 代理设置在此不详细阐述。

(3)手动编译 Atlas 源码与构建发行版

1)编译Atlas 源码。由于Maven 工具默认分配的内存比较小，因此我们在编译时需要 将其内存上限调整为一个合适值。在linc-1 上，使用hdfs 用户，在 Apache Atlas 的源码根 目录下执行以下命令：

$export MAVEN\_OPTS="-Xmx1536m-XX:MaxPermSize=512m"&&mvn -DskipTests clean install 或者使用以下命令，该命令不会跳过测试代码的编译：

S export MAVEN\_OPTS="-Xmx1536m -XX:MaxPermSize=512m"66 mvn clean install

上面的编译操作耗时特别长，而由于网络原因可能会导致编译失败，需要重复多次。 2)构建发行包。Atlas 在构件发行版时， maven 命令有一些必选以及可选参数。

口 dist: 必选参数，使用dist 参数编译将会激活Atlas 编译配置文件中的 maven- assembly-plugin 插件，它定义了发行包的文件结构，同时也指定了将源码中的哪些 资源文件拷贝到最终的发行包中，这些资源文件也包括源码中内置的 HBase 和 solr 的资源文件。若安装 Atlas 时配置为使用内置的 HBase 和 Solr 服务，将会用到发行

**146** 第二篇 开源实现篇

包中的 HBase 和 Solr 的资源文件。

使用以下命令构建发行包，得到的Atlas 发行包默认配置为使用外部的 HBase 和 Solr 服务。但用户可以在安装Atlas 时更改配置文件 atlas-application.propertities 中的相关参数， 从而使其使用内置的 HBase 和 Solr, 只是这样过于麻烦。如果需要的话，读者可以参考 Atlas 源码中 distro->src->main->assemblies 文件夹下 maven-assembly-plugin 的配置文件来 了解相关的内容，这些文件中描述了使用内置或外部的HBase 和 Solr 时对应的配置参数。

$mvn clean package -Pdist

**口** **external-hbase-solr:** 可选参数，如果准备在安装 Atlas 时让其使用外部的 HBas e 和 Solr, 那么在构建发行包的时候可以使用参数 external-hbase-solr, 上面介绍的构 建发行包命令默认使用了external-hbase-solr。

s mvn clean package -Pdist,external-hbase-solr

**口** **embedded-hbase-solr:** 可选参数，使用该参数后，得到的Atlas 发行包默认使用 内置的 HBase 和 Solr 服务。若使用内置的 HBase 和 Solr, 启动或关闭 Atlas 的服务 时，也会随之启动或关闭一个 HBase 服务实例和一个Solr 的服务实例。

$mvn clean package -Pdist,embedded-hbase-solr

**口** **berkeley-elasticsearch:** 可选参数，Altas 支持在构建发行包时选择使用 BerKeleyDB 数据库和 Elastic Search 搜索引擎，而非 HBase 和 Solr, 这种情况下，编译时需要 显式使用参数 berkeley-elasticsearch, 这样得到的 Atlas 发行包中的 Titan 的存储引 擎配置为 BerkeleyDB, 而搜索引擎配置为 Elastic。

$mvn clean package -Pdist,berkeley-elasticsearch

要注意的是，如果使用BerKeleyDB 和 Elastic Search,需要一些额外的操作。由于未 取得许可权，在构建发行包时，Atlas 并没有绑定Java 版本的BerKeleyDB 。安装 Atlas 时， 读者需要从<http://download.oracle.com/otn/berkeley-db/je-5.0.73.zip> 下载 BerkeleyDB 的 jar 包，并将其拷贝到目录%ATLAS\_HOME%/libext下才能正常使用。

构建完成后的Atlas 安装包文件地址为<Atlas 的源码根目录>/distro/target/apache- atlas-${project.version}-bin.tar.gz。

(4)使用脚本自行编译

除了自行编译外，读者可以使用本书提供的名为atlas-compile.sh 的自动编译脚本完成编 译。此脚本中的Atlas 下载源是 Apache官网，且版本为0.7,构建发行包的命令是 mvn clean package -Pdist, 读者使用时可根据需求自行更改。该脚本可以存储在任意目录下并执行， 本书推荐存储在/home/hdfs 目录下，脚本文件的所属用户为hdfs, 并使用hdfs 用户执行。

该脚本可实现功能的如下：

自由设定编译后文件存放的目录；当编译、打包过程中发生错误时，则会重新执行该

第4章大数据治理之 Apache Atlas 147

步骤，直到编译打包成功为止；将编译、打包过程中成功或失败的相关信息记录到日志中， 可以根据日志查看编译和打包情况。

**3.安装** **Atlas**

Apache Atlas 的官方暂时未提供编译好的安装包，因此需要自行编译。为了方便读者的 使用，本书提供了编译完成的Atlas 安装包，该安装包使用如下命令编译与构建，读者可以 在本书的官网下载此安装包。

s export MAVEN\_OPTS="-Xmx1536m-XX:MaxPermSize=512m"&&mvn -DskipTests clean install s mvn clean package -Pdist

笔者编译时跳过了测试代码的编译，该安装包默认使用外部的HBase 和 Solr 服务。若 要了解更详细的有关安装包的信息，请参考上面有关编译 Atlas 部分。

在Atlas 的解压包中的 conf 目录下有一些文件，其中 atlas-application.properties 文件是 Atlas 的主要的属性配置文件，atlas.conf是自己创建的文件，该文件的用途下面会介绍，如 图4-9所示。

|  |
| --- |
| -rw-r--r--1 hdfs hadoop 8254 Nov 1717:16 atlas-application.properties  -rw-r--r--1 hdfs hadoop 227 Nov 1621:49 atlas.conf  -rw-r--r--1 hdfs hadoop 3545 Nov 1711:12 atlas-env.sh  -rw-r--r--1 hdfs hadoop 3462 Nov 1616:34 atlas-log4j.xml  drwxr-xr-x 2 hdfs hadoop 4096 Nov 1610:21 hbase  -rw-r--r--1 hdfs hadoop574 Nov 1615:26 policy-store.txt  drwxr-xr-x 3 hdfs hadoop 4096 Nov 1810:46 solr  -rw-r--r--1 hdfs hadoop187 Nov 1615:21 users-credentials.properties |

图4-9 Atlas 解压包的 conf目录下的文件

( 1 ) 修 改 atlas-application.properties

按照本书的集群环境搭建 Atlas 需要修改 atlas-application.properties 中以下配置项，对 应的属性值需要根据自己的集群环境做适当的修改。除了以下需要修改的配置项外，其余 配置项均采用默认值。

口 表 示 Atlas 的 Titan 的存储引擎使用 HBase:

atlas.graph.storage.backend=hbase

口 在 HBase 中的表名为 apache\_atlas\_titan:

atlas.graph.storage.hbase.table=apache\_atlas\_titan

口 HBase 安装的集群主机名称(根据自己HBase 集群的主机名来填写):

atlas.graph.storage.hostname=linc-1,linc-2,linc-3

口 表 示 Titan 的索引引擎使用 solr cloud 模式：

atlas.graph.index.search.backend=solr5

atlas.graph.index.search.solr.mode=cloud

在下面这项配置中，/solr 表示 Solr 在 Zookeeper 中存储文件的根路径，该值在配置

148 第二篇 开源实现篇

Solr时会被指定，需要与Solr 的配置一致，该值在%SOLR\_HOME %/bin/solr.in.sh 中，相 应的属性名为ZK\_HOST 。2181表示 Zookeeper的服务端口号，2181为Zookeeper 的默 认端口，Zookeeper端口号配置请查找 Zookeeper的配置文件zoo.cfg, 相应的属性名为 clientPort,“:” 之前的是Solr 集群的节点名称。这些节点需要安装 Zookeeper 服务。

atlas.graph.index.search.solr.zookeeper-url=linc-1:2181/solr,linc-2:2181/solr, 3:2181/solr

linc-

下面表示是否使用内置的 Kafka服务：

atlas.notification.embedded=false

${sys:atlas.home}系统会自动识别，不需要更改：

atlas.kafka.data=${sys:atlas.home}/data/kafka

下面表示 Kafka集群安装在哪些节点上，即这些节点需要安装 Zookeeper服务。

atlas.kafka.zookeeper.connect=linc-1:2181,linc-2:2181,linc-3:2181

9092表示Kafka 的 broker server服务端口，该值请查找Kafka的配置文件server. properties, 相应的属性名为 port,9092 为默认值，本书采用的集群也采用默认值。

atlas.kafka.bootstrap.servers=linc-1:9092,linc-2:9092,linc-3:9092

atlas.kafka.zookeeper.session.timeout.ms=400

atlas.kafka.zookeeper.connection.timeout.ms=200

atlas.kafka.zookeeper.sync.time.ms=20

atlas.kafka.auto.commit.interval.ms=1000

**lk**aafkaf**=**stlaalest

atlas.kafka.auto.commit.enable=false

atlas.notification.create.topics=true

atlas.notification.replicas=1

下面表示在 Kafka中用到的 topics 名称，这些 topics 需要在 Kafka中预创建，创建方 法稍后会介绍。

atlas.notification.topics=ATLAS\_HOOK,ATLAS\_ENTITIES

atlas.audit.hbase.tablename=apache\_atlas\_entity\_audit

虽然针对本集群所需要更改的Atlas配置已经在上文中介绍过，但是，在atlas-application. propertities 中还有许多其他的属性，这些属性虽然没有在本集群中进行配置，但它们能够 实现更为丰富的功能，在介绍完 Atlas的安装后会对其进行介绍，仅供读者学习参考。

(2)修改atlas-env.sh

该文件中设置了Atlas 在启动时添加的环境变量，需要在该文件中修改以下参数：

#Atlas的根路径

export ATLAS\_HOME\_DIR=/var/local/hadoop/atlas-0.7

#HBase 的conf 路径，方便Atlas 能够读取HBase 的配置

export HBASE\_CONF\_DIR=/var/local/hadoop/hbase-1.1.5/conf

第4章大数据治理之Apache Atlas 149

说 明

export MANAGE\_LOCAL\_HBASE=false

xepxoANSO\_CRR\_=S="-server -Djava.security.auth.login.config=/var/local/

hadoop/atlas-0.7/conf/atlas.conf"

ATLAS\_SERVER\_OPTS在0.7版本之前的名称为MATADATA\_OPTS,0.7 版本及

回

0.7版本之后更名为 ATLAS\_SERVER\_OPTS。 读者可以在该参数后追加 Atlas JVM

的启动参数，在使用Kerberos 时可以追加与Java 中的JAAS(Java Authentication Authorization Service)相关的配置文件，如上面配置内容中提供的-Djava.security. auth.login.config。

(3)新建 Solr Collection

默认情况下，Atlas 使用Titan 作为图数据库，并且现在Atlas 只支持Titan 。此外，若 未安装 Cloud 模式启动的 Solr(Atlas 目前只支持 Cloud 模式的 Solr), 请先参考5.2.10节。

Atlas 需要在Solr 中用到3个Collection:vertex\_index 、fulltext\_index 、edge\_index。

在 Atlas 源码的 org.apache.atlas.repository.graph.GraphBackedSearchIndexer.java 中，可 以找到 Titan 的搜索引擎 (Solr) 初始化的相关内容，该类中的部分内容如下所示。在搜索 引擎初始化的过程中，如果在Solr 中未查找到 vertex\_index 、fulltext\_index 和 edge\_index, Atlas 会自动创建它们。为了保证能正确运行， 一般我们会手动创建这3个Collection。

org.apache.atlas.repository.graph.GraphBackedSearchIndexer java 的部分内容如下，代码 中加下划线部分调用相应的方法，分别创建了vertex\_index 、edge\_index 、fulltex\_index。

private void initialize(AtlasGraph graph)throws RepositoryException,IndexException

{

AtlasGraphManagement management =graph.getManagementSystem(); try {

management.createVertexIndex (Constants.VERTEX\_INDEX,Constants. BACKING\_INDEX,Collections.<AtlasPropertyKey>emptylist());

management.createEdgeIndex (Constants.EDGE\_INDEX,Constants. BACKING\_INDEX);

//Index for full text search

createFullTextIndex(management);

//Indexes for graph backed type system store

createTypeStoreIndexes(management);

commit(management);

}catch(Throwable t){

rollback(management);

throw new RepositoryException(t);

)

private void createFullTextIndex(AtlasGraphManagement management){

AtlasPropertyKey fullText =

150 **第二篇** **开源实现篇**

说明

management.makePropertyKey(Constants.ENTITY\_TEXT\_PROPERTY\_KEY, String.class,AtlasCardinality.SINGLE);

management.createFullTextIndex (Constants.FULLTEXT\_INDEX,fullText, Constants.BACKING\_INDEX);

}

|  |  |
| --- | --- |
|  | 在上面代码中有几个变量的值没有指明，这些值是，Constants.VERTEX\_INDEX= "vertex\_index" 、Constants.EDGE\_INDEX="edge\_index" 、Constants.FULLTEXT\_ INDEX="fulltex\_index"。 |
| 按照如下方式操作，能够手动创建Atlas所用的3个 Collection。  在linc-1 上，使用hdfs 用户，在%SOLR\_HOME%目录下执行如下命令：  $bin/solr create -c vertex\_index -d <SOLR\_CONF>-shards <numShards>-replicationFactor <replicationFactor>  s bin/solr create -c edge\_index -d <SOLR\_CONF>-shards <numShards>-replicationFactor <replicationFactor>  $bin/solr create -c fulltext\_index -d <SOLR\_CONF>-shards <numShards>-replicationFactor <replicationFactor> | |

如果没有指定numShards和 replicationFactor, 则它们默认的值为1,如果将Atlas 和Solr 安装在一个节点上，使用默认值即可满足需求。否则numShards 的值根据 Solr 集群所在主机的数量和它配置中maxShardsPerNode 的值来填写，shards 的数量 不能超过 Solr Cloud 集群节点的数量。

回

在本书中， Atlas 和 Solr安装在同一节点上，创建的具体命令如下(以创建 vertex\_ index 为例):

./bin/solr create -c vertex\_index -d server/solr/configsets/data\_driven\_schema\_ configs/conf -shards 1 -replicationFactor 1

同理，可完成 edge\_index 和 fulltext\_index 的创建。

(4)新建 Kafka Topic

1)使用脚本创建 topic。

Atlas 使用Kafka获取其他组件运行时产生的元数据信息。在使用Atlas 之前，需要根 据 Kafka的配置完成创建一组 topic 的准备工作。Atlas 提供了一个脚本来完成这件事情： bin/atlas\_kafka\_setup.py, 该脚本能够在 Atlas Server 端执行。

使用 hdfs用户，在%ATLAS\_HOME%/bin下使用以下命令即可完成这些 topic 的创建： 9./atlas\_kafka\_setup.Py

该脚本的内容如下：该脚本调用了org.apache.atlas.hook.AtlasTopicCreator 的 main 方法来创建 Kafka topic, 调用时传递需要创建的topics 名称列表以及其他的必要参数， 在 %ATLAS\_HOME%/conf/atlas-application.propertities 文件中配置了这些 Kafka topic 的名

第4章 大数据治理之Apache Atlas 151

称，配置项为 atlas.notification.topics=ATLASHOOK,ATLAS\_ENTITIES,其中 ATLAS\_HOOK 与 ATLAS\_ENTITIES 为 Kafka topic 的名称。

import sys

import atlas\_client\_cmdline as cmdline

import atlas\_config as mc

def main():

conf\_dir =cmdline.setup\_conf\_dir()

jvm\_opts\_list =cmdline.setup\_jvm\_opts\_list(conf\_dir,'atlas\_kafka\_setup.log') atlas\_classpath =cmdline.get\_atlas\_classpath(conf\_dir)

topics\_array =mc.get\_topics\_to\_create(conf\_dir)

process =mc.java("**org.apache.atlas.hook.AtlasTopicCreator",** topics\_array,

atlas\_classpath,jvm\_opts\_list)

return process.wait()

if name ==' main\_':

try: returncode =main()

except Exception as e:

print "Exception in setting up Kafka topics for Atlas:8s"g str(e)

returncode =-1

sys.exit(returncode)

在某些情况下，在 Atlas server启动之前， 一些 HOOK 可能已经开始工作了。这种情况 下，可以通过脚本%ATLAS\_HOME%/bin/atlas\_kafka\_setup\_hook.py 让这些topics 运行在 这些 HOOK 安装的主机上。

使用hdfs 用户，在%ATLAS\_HOME%/bin下使用以下命令即可完成这些 topic 创建：

S./atlas\_kafka\_setup\_hook.py

该脚本内容与 atlas\_kafka\_setup.py 基本相同，唯一不同的是其中的加粗部分， org. apache.atlas.hook.AtlasTopicCreator 在运行过程中会把相应的运行日志输出到 atlas\_kafka\_ setup\_hook.log 中而非atlas\_kafka\_setup.log 中。

import sys

import atlas\_client\_cmdline as cmdline

import atlas\_config as mc

def main():

mnf\_\_s\_lis.setup(etup\_jvm\_opts\_list(conf\_dir,'atlas\_kafka\_setup.

**hook.log')**

//此处省略部分代码

2)手动创建 topic。

除了使用Atlas 提供的创建脚本之外，也可以使用Kafka 命令手动创建这些 topic, 过 程如下：

在 linc-1 节点上，在%KAFKA\_HOME%下使用hdfs 用户执行以下命令：

S./bin/kafka-topics.sh --create --zookeeper linc-1:2181,linc-2:2181,linc-3:2181

**152** **第二篇** **开源实现篇**

--topic ATLAS\_HOOK --replication-factor 1 --partitions 1

$./bin/kafka-topics.sh --create --zookeeper linc-1:2181,linc-2:2181,linc-3:2181 --topic ATLAS\_ENTITIES --replication-factor 1 --partitions 1

**回端**

linc-1 、linc-2 、linc-3 是 Kafka 所在的集群，这些集群需要安装 Zookeeper,2181 表 示 Zookeeper 的服务端口号，本书的集群使用默认值2181。Zookeeper 端口号配置

请查阅Zookeeper 的配置文件zoo . cfg,相应的属性名为client Port 。

(5)启动 Atlas

在Atlas 的安装目录下执行以下命令：

$./bin/atlas\_start.Py

Atlas的日志文件在$ATLAS\_HOME/logs中，其中 application.log 记录了Atlas 的全部 日志信息，可以在该文件中观察到 Atlas 的启动运行情况。为了方便动态地观察，可以通过 命令$tail-f application.log来实时地将日志文件信息打印在屏幕上。

(6)查看 Atlas 的 Web UI

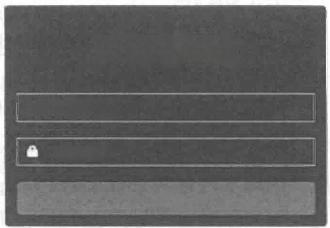
在与Atlas 安装的节点互连的任意机器上，使用浏览器，在浏览器的地址中键入

“<atlas-host-ip>:21000” 便可以访问Atlas 的 Web UI, 如图4-10所示。

启动后第一次登录 Web UI可能会访问不到，需 要等待一段时间， Atlas 需要完成一些初始化的工作。

初始化的用户名和密码都是 admin, 单击Login 即可登录。若返回的是一串字符串，将浏览器地址栏 中21000之后的内容删除，按回车键，即可正常访问 Atlas的 Web UI。

**4.Atlas** **配置参数简介**



**Apache Atlas**

Usemame

Password

Login

图4-10 Atlas 登录界面

Atlas可支持的功能有很多，本书在介绍Atlas 安装时仅仅用到了部分的功能，在 此，详细地介绍一下Atlas 可配置的功能。以下参数配置在%ATLAS\_HOME%/conf/atlas- application.properties 中。

(1)SSL

单向 SSL (服务器端认证)和双向 SSL (服务器/客户端认证)在Atlas 中都已经得到支 持。使用以下属性配置SSL:

口 atlas.enableTLS(false|true) [默认为 false] 启用/关闭 SSL 监听。 口 atlas.enableTLS 服务器使用的 Keystore 文件的路径。

口 truststore.file truststore文件的路径， truststore 文件包含了其他可信实体的证书， 例如：如果启用双向SSL, 该文件会存放客户端程序的证书。

口 client.auth.enabled(false|true) [默认为 false] 启用/关闭客户端的认证。如果启用

第4章大数据治理之Apache Atlas 153

注 意

了客户端认证，在 transport session的密钥创建过程中，客户端需要向服务器认证 身份。

口 cert.stores.credential.provider.path 凭据提供程序的存储文件的路径，密钥库、信 任库和服务器证书的密码都保存在该路径的文件中。

为了防止用户使用空白的密码，Atlas 使用凭据提供程序，保证存储的是正确的 密码格式，利用在 “bin” 目录下的名称为cputil 脚本可以创建符合要求的密码。执行 cputil 后，需要输入信任凭据提供程序的文件路径，路径可以是本地文件路径，格式为： jceks://file/local/file/path/file.jceks, 也可以是 hdfs 上的文件路径，格式为： jceks://hdfs@ namenodehost:port/path/in/hdfs/to/file.jceks, 信任凭据程序的文件格式通常是 jceks。输入完 信任凭据提供程序路径后，下一步就是输入密钥库、信任库和服务器证书的密码。

口 atlas.ssl.exclude.cipher.suites 指明要排除的加密方式列表。

NULL. 、.\*RC4.\* 、.\*MD5.\* 、.\*DES.\* 、.\*DSS.\*都是默认被排除的不安全的加密方式。 如果还有其他加密方式需要排除，首先添加默认排除的加密方式，如：

atlas.ssl.exclude.cipher.suites=.\*NULL.\*,.\*RC4.\*,.\*MD5.\*,.\*DES.\*,.\*DSS.\*

然后再追加其他需要被排除的加密方式列表，并使用半角“,”分隔每个加密方式。 该属性中被添加的加密方式可以使用全名称也可使用匹配表达式。如果配置了atlas.ssl. exclude.cipher.suites, 该配置项会覆盖默认配置，默认配置将不起作用。

(2)Service Authentication

Atlas 在启动时会关联一个安全可靠的身份。默认情况下，在一个不安全的集群环境 中，该身份和在系统中启动 Atlas 服务的用户名是一样的，但是在集成了Kerberos 的安全 的集群环境中，需要配置一个keytab 和 principal, 从而让Atlas 服务能够向KDC 鉴定身 份，这使得 Atlas 服务能够持续地与其他的安全集群中的服务交互，如 HDFS。

口 atlas.authentication.method (simple|kerberos)[默认为 simple] 表明所使用的身份 鉴定的方法。 “simple” 是默认的身份鉴定机制，它会使用系统中安全的用户身份。 “kerberos” 表明服务需要使用配置文件中提供的 keytab 和principal 来向 KDC 认证 身份，此时需要提供以下两个属性： atlas.authentication.keytab keytab, 即文件的路 径； atlas.authentication.principal, 用于向 KDC认证身份，通常情况下，其格式为 “<user>/<host>@<realm>”, 可以使用\_HOST 来替代<host>,\_HOST 能够在运行 时被替换为系统中的 local hostname, 即主机名。

若配置了HBase 作为存储引擎，并且HBase 配置了Kerberos, 则 Titan 需要有足够 的用户权限来创建和访问HBase 中的表，此时需要在HBase 中赋予Atlas 用户足够 的权限。本书中 HBase 未配置 Kerberos, 所以不需要赋予权限的步骤。

(3)Authentication

Atlas 支持以下身份认证的方法：File 、Kerberos 和LDAP。

**154** **第二篇** **开源实现篇**

在 atlas-application.propertities 中设置以下的属性为 true 或 false 来启用相应的身份认 证方法：

atlas.authentication.method.kerberos=truelfalse

atlas.authentication.method.ldap=truelfalse

atlas.authentication.method.file=truelfalse

如果两个或两个以上的身份认证方法被设置为了true, 会按照配置的先后顺序来使用， 如果前一个身份认证方法失败了则按照顺序采用下一个被启用了的身份认证方法。

1)File 方法。

如果使用File 身份认证方法，需要在用户凭据存储文件中提供以特定格式存储的用户 登录所需信息，atlas-application.properties 中的 atlas.authentication.method.file.filename 配 置项需要配置为该凭据存储文件的路径，如下：

ti}on ename=${sys:atlas.home}/conf/users-credentials.

properties

用户凭据文件中的用户信息存储格式如下：

username=group::sha256-password

用户组可以为ADMIN 、DATA\_STEWARD,或者 DATA\_SCIENTIST。

用户的密码需要使用sha256 编码格式，可以使用系统自带工具得到该编码格式的密 码，如：

$echo -n "Password"l sha256sum

e7cf3ef4f17c3999a94f2c6f612e8a888e5b1026878e4e19398b23bd38ec221a

2)Kerberos 方法。

启用Atlas 的 kerberos模式的身份认证，需要在 atlas-application.properties 中将 atlas. authentication.method.kerberos 设置为 true。

atlas.authentication.method.kerberos =true

同时，需要设置以下的属性：

atlas.authentication.method.kerberos.principal=<principal>/<fqdn>EEXAMPLE.COM

atlas.authentication.method.kerberos.keytab =<key tab filepath>/<filename>.keytab atlas.authentication.method.kerberos.name.rules=RULE:[2:$1@S0](atlas@EXAMPLE.

CoM)s/.\*/atlas/

3)LDAP 方法。

启用Atlas 的 LDAP模式的身份认证，需要在 atlas-application.properties 中设置 atlas. authentication.method.ldap 为true, 同时设置atlas.authentication.method.ldap.type 为ldap 或 者 ad: 如果要连接到 Active Directory 则设置为 ad, 否则为 ldap。

atlas.authentication.method.ldap=true

**第4章** **大数据治理之Apache Atla**s 155

atlas.authentication.method.ldap.type=1daplad

**此外需要根据选择进行不同的设置。**

**若要连接到** **Active Directory, 则设置以下配置项：**

atlas,authentication.method.ldap.ad.domain=example.com

atlas.authentication,<method.ldap.ad.url=1dap://><AD server ip>:389 atlas.authentication.method.ldap.ad.base.dn=DC=example,DC=com

atlas.authentication.method.ldap.ad.bind.dn=CN=Administrator,CN=Users,DC=example, DC=com

atlas.authentication.method.ldap.ad.bind.password=<password>

atlas.authentication.method.ldap.ad.referral=ignore

atlas.authentication.method.ldap.ad.user.searchfilter=(sAMAccountName={0}) atlas.authentication.method.ldap.ad.default.role=ROLE\_USER

**若连接到** **LDAP Directory, 则需要设置以下配置项：**

atlas.authentication.method.1dap.url=ldap://<Ldap server ip>:389

atlas.authentication.method.ldap.userDNpattern=uid={0],ou=users,dc=example,dc=com atlas.authentication.method.ldap.groupSearchBase=dc=example,dc=com

atlas.authentication.method.ldap.groupSearchFilter=(member=cn={0),ou=users,dc=ex ample,dc=com

atlas.authentication.method.ldap.groupRoleAttribute=cn

atlas.authentication.method.ldap.base.dn=dc=example,dc=com

atlas.authentication.method.ldap.bind.dn=cn=Manager,dc=example,dc=com atlas.authentication.method.ldap.bind.password=<password>

atlas.authentication.method.ldap.referral=ignore

atlas,authentication.method.ldap.user.searchfilter=(uid={0})

atlas.authentication.method.1dap.default.role=ROLE\_USER

**(4)JAAS configuration**

**在安全的集群中，许多Atlas需要使用的组件，(如Kafka) 需要Atlas 使用JAAS 向它**

**们认证身份，从而能够正常地访问服务。可以使用以下属性配置JAAS:**

份en鉴t-ModuleName =com.sun.security.auth.module.Krb5LoginModule i - CsFlag=real

en来t- n eyTab=true

-，iestoreKey=true

s.t行-i.ceName=<name>

#keytab文件路径

atlas.jaas.client-id.option.keyTab=<atlas

.cri-itioipal=<atlas

**例如，下面的配置便符合上述格式：**

keytab filepath>

principal>

**156** **第二篇** **开源实现篇**

atlas.jaas.KafkaClient.loginModuleName =com.sun.security.auth.module.Krb5LoginModule atlas.jaas.KafkaClient.loginModuleControlFlag =required

atlas.jaas.KafkaClient.option.useKeyTab =true

atlas.jaas.KafkaClient.option.storeKey =true

atlas.jaas.KafkaClient.option.serviceName =kafka

atlas.jaas.KafkaClient.option.keyTab=/etc/security/keytabs/kafka\_client.keytab atlas.jaas.KafkaClient.option.principal =kafka-client-18EXAMPLE.COM

以上配置项能够完成JAAS 的配置，但是Java 本身也有自己的JAAS 配置方法，上面 的配置项等同于在JAAS 文件中添加了如下内容：

KafkaClient {

com.sun.security.auth.module.Krb5LoginModule required

useKeyTab=true

storeKey=true

serviceName=kafka

keyTab="/etc/security/keytabs/kafka\_client.keytab"

principal="kafka-client-1QEXAMPLE.COM";

);

(5)SPNEGO-based HTTP Authentication

开启Atlas 的 SPNEGO 支持后，可以将Atlas 的 HTTP 访问保护起来。现在有两种身份 认证机制： simple 和kerberos。

simple 机制下，通过提供的用户名鉴定身份。

kerberos 机制下，客户端使用一个经过KDC 认证的身份向Atlas 服务器认证自己的身 份。客户端访问服务器前先认证身份，通常情况下，通过 “kinit” 命令完成。认证后，被 认证的身份会通过SPNEGO 协议机制与服务关联，从而使客户端能够访问服务器。

配置支持 SPNEGO 的属性如下：

口 atlas.http.authentication.enabled(true|false) [默认为 false] 该属性表明是否启用 HTTP 访问的身份认证。

口 atlas.http.authentication.type (simple|kerberos)[默认为 simple] 配置身份认证的类型。

口 atlas.http.authentication.kerberos.principal Kerberos principal 的名称必须以 “HTTP/” 开头，如 “HTTP/localhost@LOCALHOST” 。 该属性没有默认值。

口 atlas.http.authentication.kerberos.keytab 包 含 Kerberos principal 的 keytab 文件的 路径。

口 atlas.rest.address Atlas的网页端应用的访问地址，格式为<http/https>://<atlas- fqdn>:<atlas port>。

(6)Client security configuration

如果需要Atlas 客户端与配置了SSL 或者 Kerberos 身份认证的Atlas 服务交互，用户 需要给客户端提供配置了相关配置项的配置文件，以允许客户端与服务器交互。在 atlas- application.properties 中添加如下属性，并拷贝该文件到客户端的 classpath 或者被客户端系 统的 “atlas.conf” 环境变量所标识的目录下，以便客户端能够找到该文件。

第4章 大数据治理之 Apache Atlas 157

口 atlas.enableTLS 启用/关闭 SSL 客户端的基础通信设施。

口 keystore.file 客户端使用的keystore 文件路径。该文件仅仅在Atlas 启用了双向 SSL 才需要，并且该文件中包含客户端的凭据。

口 truststore.file 信任库文件的路径。该文件中包含了可信实体的凭证，如服务器的

凭据或者共享权威认证证书。此文件在启用单向或双向 SSL 时都需要提供。

口 cert.stores.credential.provider.path 凭据提供程序存储文件的路径、密钥库、信任

和客户端证书的密码保存在这个安全的文件中。

如果身份认证被启用，需要配置用来向服务认证身份的以下属性：

口 atlas.http.authentication.type (simple|kerberos)[默认为 simple] 身份认证机制类型。

(7)Authorization

Atlas 的授权方式有 simple 和 ranger, 设置以下属性来启用相应的授权方式：

#<Qualified Authorizer Class Name>表示带有包名前缀的授权实现类

atlas.authorizer.impl=simple I ranger l <Qualified Authorizer Class Name>

1)simple 授权方式。

simple 授权方式中，授权策略存储文件在本地配置，该授权策略存储文件路径通过 atlas-application.properties 中的 atlas.auth.policy.file 配置项来配置。

atlas.auth.policy.file={{conf\_dir}}/policy-store.txt

策略存储文件的内容格式如下：

Policy\_Name;;User\_Name:Operations\_Allowed;;Group\_Name:Operations\_Allowed;;

Resource\_Type:Resource\_Name

如 admin 的策略如下：

adminPolicy;;admin:rwud;;ROLE\_ADMIN:rwud;;type:\*,entity:\*,operation:\*,taxonomy:\*, term:\*

谵 User\_Name、Group\_Name 和 Operations\_Allowed 的值可以是一个列表，列表中的

明

值使用英文逗号“,”隔开。

Resource\_Type 的值如下： Operation 、Type 、Entity 、Taxonomy 、Term 、Unknown。

Operations\_Allowed 的值中，r=read 、w=write 、u=update 、d=delete。

2)ranger 授权方式。

使用ranger 授权方式需要启用Atlas-Ranger 插件，请参照5.2.8节启用Atlas-Ranger 插件。

Atlas 支持与很多组件进行元数据交换，它的实现方式是使用一种被称为HOOK 的技 术。目前Atlas 支持的 HOOK 包括如下几项： Hive HOOK 、Sqoop HOOK 、Falcon HOOK、 Storm HOOK,这 些 HOOK 能够做到在相应组件中有数据被操作时，可以获取这些操作

158 第二篇 开源实现篇

的元数据和被操作的数据的元数据，并存储到Atlas 中，以便Atlas 能够监控这些数据的 血统。

对于初次使用Atlas 的用户，建议阅读本节的部署、运行和HOOK 的使用部分。Atlas 是 Apache 的顶级孵化项目，它的发展前景非常好。但是正如所有新生技术那样，它同样面 临着不成熟的问题，在编写本书时，Atlas 的部分功能依旧有些问题或者没有实现。如果后 期 Atlas 有重大的更新，而读者对于新的功能有迫切的需求，从而需要重新编译，可以阅读 源码编译的章节。

**4.2.2** **添加或修改** **Atlas Web UI的登录账户**

如4.2.1 节所述，如果使用 File 身份认证方法，需要在用户凭据存储文件中提供特定格 式存储的用户登录信息，并且需要使用atlas-application.properties 中的 atlas.authentication. method.file.filename 来指定该凭据存储文件路径。

#§{sys:atlas.home} 能够自动识别，不需要做任何更改

atlas.authentication.method.file.filename=${sys:atlas.home}/conf/users-credentials, properties

用户凭据文件中的用户信息存储格式如下：

username=group::sha256-password

用户组可以为ADMIN 、DATA\_STEWARD或者 DATA\_SCIENTIST。

用户的密码需要使用sha256 编码格式，得到该编码格式的密码可以使用Linux 自带工 具，命令如下：

$echo -n "Password"l sha256sum

e7cf3ef4f17c3999a94f2c6f612e8a888e5b1026878e4e19398b23bd38ec221a

这里，添加用户名为 linc, 密码为123456,用户组为ADMIN 的用户信息。过程如下： 1)计算出密码的 sha256 的哈希密文：

$echo -n "123456"l sha256sum

8d969eef6ecad3c29a3a629280e686cf0c3f5d5a86aff3ca12020c923adc6c92 -

2)将用户信息添到 users-credentials.properties 的末尾，在末尾添加以下内容。值得注 意的是，添加的哈希密文必须是小写字母。

zte=ADMIN::8d969eef6ecad3c29a3a629280e686cf0c3f5d5a86aff3cal2020c923adc6c92

3)重启 Atlas 的服务，即可在 Web UI中使用新账户登录。

初次使用新用户登录时会有权限的问题，虽然给新用户添加了ADMIN 的用户组，但 是用户权限需要额外的操作来指定。

日 本书Atlas 的安装配置参考：<http://atlas.apache.org/>。

第4章 大数据治理之Apache Atlas 159

如果采用simple 授权方式，授权策略存储文件在本地配置，配置atlas-application. properties 中的 atlas.auth.policy.file 属性值为策略存储文件的路径。

atlas.auth.policy.file={{conf\_dir}}/policy-store.txt

若要了解该文件具体的内容，请参照4.2.1节中的认证部分。

在该文件中的 admin:rwud后添加新用户的操作权限linc:rwud, 与 admin:rwud 之间使 用英文逗号“,”隔开。

4)重启Atlas 服务即可使用新用户。

如果采用ranger授权方式，请参照5.2.9节配置新用户的权限。

**4.2.3** **配** **置** **Hive 通过** **Hive HOOK 导入数据**

配置 Hive HOOK 之前请先参考3.3.3节。完成Hive的安装。

**1.Hive Model**

Atlas源码中的 org.apache.atlas.hive.model.HiveDataModelGenerator 定义了默认支持的 Hive 模型，其中有如下几种类型。

口 hive\_db(类)-继承自Referenceable-属性： name,clusterName,description,locationUri,

parameters,ownerName,ownerType

口 hive\_storagedesc(类) - 继承自Referenceable- 属性： cols,location,inputFormat, outputFormat,compressed,numBuckets,serdeInfo,bucketCols,sortCols,parameters,

storedAsSubDirectories

口 hive\_column(类)-继承自Referenceable]- 属性：name,type,comment,table

口 hive\_table(类)-继承自DataSet- 属性： name,db,owner,createTime,lastAccessTime, comment,retention,sd,partitionKeys,columns,aliases,parameters,viewOriginalText,

viewExpandedText,tableType,temporary

O hive\_process(类) - 继承自Process- 属性： name,startTime,endTime,userName,

operationType,queryText,queryPlan,queryld

O hive\_principal\_type(枚举类型)-属性： USER,ROLE,GROUP

口 hive\_order(Struct 类型)-属性： col,order

口 hive\_serde(Struct 类型)-属性：name,serializationLib,parameters

Atlas中的 Entitiy在创建时使用唯一的名称进行标识，这些 Entity 提供了命名空 间，也可以用于查询和血统。dbName 、tableName 和columnName应该使用小写字母。 clusterName 描述如下：

口 hive\_db- 属性名称-<dbName>@<clusterName>

口 hive\_table- 属性名称-<dbName>,<tableName>@<clusterName>

O hive\_column-属性名称-<dbName>.<tableName>.<columnName>@<clusterName>

**160** **第二篇** **开源实现篇**

口 hive\_process -属性名称-<queryString>-trimmed query string in lower case

**2.导入Hive 元数据**

在 Atlas 源码中的 org.apache.atlas.hive.bridge.HiveMetaStoreBridge 实现了将 Hive的元 数据导入Atlas中，导入时使用 org.apache.atlas.hive.model.HiveDataModelGenerator 中定义 的 Hive 数据模型构造元数据。

import-hive.sh 脚本调用org.apache.atlas.hive.bridge.HiveMetaStoreBridge, 从而实现 Hive数据导入的功能。 org.apache.atlas.hive.bridge.HiveMetaStoreBridge 需要 Hadoop 和Hive 的 classpath 下的jar 包的支持才能生效，因此需要添加相应的环境变量。

访问 Hadoop下的jar 包， 一种方法是设置HADOOP\_CLASSPATH 环境变量，另一种 方法是设置HADOOP\_HOME 指 向Hadoop安装的根目录。这里采用第2种方法，读者可 自行选择。

对于Hive 下的jar 包，第1种办法是设置Hive\_HOME 指向Hive 安装目录的根目录， 第2种办法是设置Hive\_CONF\_DIR 指向Hive 的配置文件根目录(%Hive\_HOME%/conf), 即在/etc/profile 中追加 export Hive\_CONF\_DIR=/var/local/hadoop/hive-1.2.1/conf, 读者请 自行选择任一办法。

在 linc-1 上，使用hdfs 用户，在%ATLAS\_HOME%/bin目录下执行 import-hive.sh 脚 本，即可完成Hive 的数据导入。

$./bin/import-hive.sh

该脚本的内容如下：

#检查与设置环境变量，过程略

…

while [[§{1}=~^\-D ]];do

JAVA\_PROPERTIES="S{JAVA\_PROPERTIES}§{1}"

shift

done

echo Using Hive configuration directory ["$Hive\_Cp"]

echo "Log file for import is SLOGFILE"

"S{JAVA\_BIN}"S{JAVA\_PROPERTIES}-cp"S{CP}"

org.apache.atlas.hive.bridge.HiveMeta-StoreBridge

RETVAL=$?

[SRETVAL -eq 0]&&echo Hive Data Model imported successfully!!! [SRETVAL -ne 0 ]66 echo Failed to import Hive Data Model!!!

该脚本主要执行了org.apache.atlas.hive.bridge.HiveMetaStoreBridge 中的main 方法， 该 Java的代码实现将 Hive中的元数据信息加载进Atlas 中。

该脚本执行的日志存储在%ATLAS\_HOME%/logs/import-hive.log中。在执行的过程中 需要输入用户名和密码，Atlas 默认的用户名和密码都是admin。在按照前一节配置了其他 用户名和密码后，也可以使用配置的新的用户名和密码。数据导入成功后，控制台会有如

第4章 大数据治理之Apache Atlas 161

5

|  |
| --- |
| result for hive |
| Name **Type** **Name** |
| default hive\_db |
| localtable hive\_table |
| student hive\_table |
| 3eb6be6e-4680-4a4f-91da-02d1674b1580 hive\_storagedesc |
| 6578146b-4a03-412c-bac3-c9988ac6464b hive\_storagedesc |

下输出：

Hive Data Model imported successfully!!!

进入Atlas 的 Web UI, 单击 search->Text 标签，输入Hive 执行脚本前Hive 中的任意 一个表名，即可查找到导入Atlas 中的该表 的元数据。或者搜索Hive, 结果如图4-11

所示。

**3.Hive HOOK**

import-hive.sh 脚本能够完成将Hive中的 元数据信息导入Atlas中，但这个过程需要手 动完成，并且不能实时地监控 Hive中的数据

图4-11 Atlas中搜索 Hive 的结果

变动。本节会配置 Hive HOOK, 完成对Hive 的监听，自动导入Atlas 规定格式的元数据到 Atlas 中。

Hive 通过 Hive HOOK 监听 Hive的执行命令， Hive HOOK 可以用来添加、更新、删除 Atlas 中的实体。Hive HOOK 提交请求到执行线程池以防止在 Hive 命令的执行过程中被阻 塞。这些线程以实体作为信息提交到通知服务器， Atlas 服务读取这些信息并在自己的服务 中注册这些实体。

Hive HOOK的配置过程如下：

1 ) 在hive-site.xml 中的<configuration></configuration> 之间添加如下内容，hive-site. xml 文件在%Hive\_HOME%/conf 下。

<property>

<name>hive.exec.post.hooks</name>

<value>org.apache.atlas.hive.hook.HiveHook</value>

</property>

<property>

<name>atlas.cluster.name</name>

<value>atlas</value>

</property>

说明： <value>atlas</value> 表示集群的名称，名称可以自己设定，没有明确要求。

2 ) 打 开 %Hive\_HOME%/conf/hive-env.sh 文件，并在该文件的最后追加环境变量

Hive\_AUX\_JARS\_PATH。

s vim hive-env.sh

export Hive\_AUX\_JARS\_PATH=/var/local/hadoop/atlas-0.7/hook/hive

其中/var/local/hadoop/atlas-0.7 为 Atlas 的安装根目录，读者可根据自己集群更改。

3 ) 将 %ATLAS\_HOME%/conf/atlas-application.properties 拷贝到 Hive 的 conf 目

录下。

**162** **第二篇** **开源实现篇**

$cp /var/local/hadoop/atlas-0.7/atlas-application.properties /var/local/hadoop/

hive-1.2.1/conf

4)Atlas 的 atlas-application.properties 文件中提供了一些属性，控制Hive HOOK的线 程池和通知系统的性能。本集群通常采用默认值，读者可根据自己的需要，对这些参数进 行调整。详细介绍请参考4.4节。

**4.测试** **Hive HOOK**

Hive在创建表的时候，会存储到HDFS 上，其存储的目录使用hive-site.xml 文件 中 hive.metastore.warehouse.dir 配置项指定，在本集群中其存储目录为/var/local/hadoop/ hive-1.2.1/warehouse, 可以在 Hadoop Web UI上查看。需要特别指出的是，启动Hive 的 用户需要对 HDFS上的该目录有写入权限，因此需要赋予权限。此外通过配置 hdfs-site.xml 中的dfs.permissions.enabled 为 false, 也可以解决Hive 在创建表的过程中没有写入权限的 问题。

在配置完 Hive HOOK 之后，通过以下命令来测试 Hive HOOK 是否生效：

S hive

s\_\_l select id,city from localtable;

hive >exit;

由于已经配置了Hive HOOK, 因此这 里创建表格，会自动将最新创建的表格的 元数据导入Atlas 中，而不用我们手动执行 import\_hive.sh 以批处理方式导入元数据。

在Atlas 的 Web UI上，单击 search >Text 标签，搜索 ctas\_localtable, 可以搜索到之 前创建的表的元数据信息，如图4-12所示。

**5.限制**

|  |  |
| --- | --- |
| Name **Type** **Name** | |
| create table ctas\_localtable as select id,city from localtable hive\_process | |
| c303479b-94e3-4147-bbda-700c7ef63561 hive\_storagedesc | |
| ctas\_localtable | hive\_table |
| hive\_column id | |
| city hive\_column | |

图4-12 Atlas 中搜索 ctas\_localtable 的结果

由于在Hive中数据库名称、表名称、列名称对大小写字母不敏感，因此通过Atlas API 查询entity 的名称的时候应该使用小写字母。

到目前为止，以下的 Hive操作可以被 Hive HOOK 捕捉到：

口 create database

口 create table/view,create table as select

口 load,import,export

口 DMLs(insert)

口 alter database

口 alter view

口 alter table (skewed table information,stored as,protection is not supported)

第4章 大数据治理之 Apache Atlas 163

**4.2.4** **配** **置** **Sqoop** **通** **过** **Sqoop** **HOOK** **导入数据**

Sqoop 是 Hadoop 环境下连接关系型数据库和 Hadoop 存储系统的桥梁，支持多种关系 数据源和 Hive 、HDFS 、HBase 的相互导入。为了更好地管理这些数据，Atlas 支持通过 Sqoop HOOK 导入被操作的数据的元数据信息和这些操作的元数据信息，从而知道数据的 血统，以方便用户便利地最大化地挖掘出数据的价值。

在写此书的时候， Apache Sqoop 的官方提供的Sqoop1 的最新版本为Sqoop 1.4.6, 该 版本不提供Sqoop HOOK的功能。这里将介绍如何让官方下载的Sqoop 1.4.6 支持 Sqoop

HOOK。

**1.下载** **Apache Sqoop**

本书以下载 Sqoop 1.4.6 为例，可以通过下面的命令下载 Sqoop 1.4.6。建议读者浏览 Apache Sqoop的官方网站，如果在阅读本书时官方提供了Sqoop 1更新的版本，如1.4.7, 建议下载新版。1.4.6版本不支持 Sqoop HOOK,需要按照下面的内容进行配置，以让其支 持Sqoop HOOK。Sqoop 2 是一个全新架构的版本，该版本采用主从结构设计，版本标记为 Sqoop 1.99.x。在撰写本书时， Sqoop2 的最新版本为1.99.7,该版本不支持 Sqoop HOOK。

在linc-1 上，使用hdfs 用户，在/home/hdfs 目录下执行以下命令完成 Sqoop 的下载与 解压：

s wget <http://mirror.bit.edu.cn/apache/sqoop/1.4.6/sqoop-1.4.6.bin>\_\_hadoop-2.0.4- alpha.tar.gz

$tar zxvf sqoop-1.4.6.bin hadoop-2.0.4-alpha.tar.gz

$mv sqoop-1.4.6.bin\_hadoop-2.0.4-alpha /var/local/hadoop/sqoop-1.4.6

本书将 Sqoop 安装在了/var/local/hadoop 目录下。

**2.替换jar 包**

将本书提供的Sqoop 1.4.7的 jar 包导入Sqoop 的安装目录下，删除原有的1.4.6.jar 包， 并将导入的 Sqoop1.4.7.jar 重命名为 Sqoop-1.4.6.jar。

**3.配置环境变量**

本书以 CentOS 6.5 为例添加环境变量。

S sudo vim /etc/profile

**在该文件中的后面追加以下内容：**

#/var/local/hadoop/sqoop-1.4.6 为sqoop 安装根目录，读者请根据集群环境自行更改

export SQ0OP\_HOME=/var/local/hadoop/sqoop-1.4.6

export PATH=SPATH:SSQOOP\_HOME/bin

**4.复制jar 包到** **Sqoop的** **lib 中**

本节以MySQL 与 Hive 互导为例子，因此需要有 MySQL 的连接包，如果使用其他的 数据库，需要将对应数据库的连接包拷贝到%SQOOP\_HOME%/lib 下。

164 第二篇 开源实现篇

拷 贝mysql-connector-java-5.1.38-bin.,jar 到 Sqoop 的 lib下，鉴于本书中使用的MySQL 版 本 ，MySQL的连接包版本最好为5.1.38以上。本书中的MySQL 连接包在/home/hdfs/ mysql-connector-java-5.1.38/mysql-connector-java-5.1.38-bin.jar, 因此使用以下命令拷贝连 接包：

$cp /home/hdfs/mysql-connector-java-5.1.38/mysql-connector-java-5.1.38-bin.jar /var/

local/hadoop/sqoop-1.4.6/lib

如果在拷贝连接包时出现权限错误，需要检查%SQOOP\_HOME% 的文件夹及其子文 件夹的权限。如果是权限不够的问题，增加%SQOOP\_HOME%的用户权限。

$sudo chmod -R 775 /var/local/hadoop/sqoop-1.4.6

**5.测试** **Sqoop**

在任意目录下，执行以下命令来测试 Sqoop 是否能够访问MySQL。

$sqoop list-databases -connect jdbc:mysql://linc-1:3306 -username root -password

MyNewPass4!

在本书使用的集群环境中，MySQL 安装在linc-1,MySQL 的用户名为 root,密码为 “MyNewPass4!” 。 使用该命令时，读者可根据自己的集群环境做适当更改。

**6.配置** **Sqoop HOOK**

在任意目录下，使用 hdfs 用户执行以下操作：

s vim /var/local/hadoop/sqoop-1.4.6/conf/sqoop-site.xml

在 <configuration></configuration> 标签之间，添加如下内容：

<property>

<name>sqoop.job.data.publish.class</name>

<value>org.apache.atlas.sqoop.hook.SqoopHook</value>

</property>

org.apache.atlas.sqoop.hook.SqoopHook 是 Atlas 中提供的类，该类定义了Sqoop 相关 的元数据如何导入Atlas, 该类所在的包名为 sqoop-bridge-0.7-incubatingjar, 该包所在位置 为 %ATLAS\_HOME%/hook/sqoop/, 稍后会将Atlas 中与 Sqoop 相关的包拷贝到%SQOOP\_ HOME%/lib下。

目前，Sqoop HOOK 只支持hiveimport, 即数据导入Hive 中，导入时使用在org.apache. atlas.sqoop.model.SqoopDataModelGenerator 中定义的数据模型。此外 Sqoop HOOK 目前只 支持一个publishing class,即 org.apache.atlas.sqoop.hook.SqoopHook。

此处介绍 Sqoop HOOK 的代码是如何实现的，有兴趣的读者可以了解下，否则可以跳 过以下代码部分。

在 Sqoop 中 ，PublishJobData 类负责将Sqoop 操作时产生的元数据信息导入Atlas 中 PublishJobData 代码如下：

**第4章** 大数据治理之Apache Atlas **165**

public final class PublishJobData {

public static void publishJobData(Configuration conf,Sqoopoptions options,

String operation, String tableName,long

startTime){

long endTime =new Date().getTime();

//取得sqoop-site.xml 配置文件中添加的sqoop.job.data.publish.class 配置项的值

String publishClassName =conf.get (ConfigurationConstants.DATA\_PUBLISH\_CLASS); if(!StringUtils.isEmpty(publishClassName)){

try {

Class publishClass = Class.forName(publishClassName);

//得到org.apache.atlas.sqoop.hook.SqoopHook 的类型信息

//利用类型信息初始化org.apache.atlas.sqoop.hook.SqoopHook

Object obj=publishClass.newInstance();

//SqoopHook 继承自SqoopJobDataPublisher, 因此该判断可以通过

if(obj instanceof SqoopJobDataPublisher){

SqoopJobDataPublisher publisher =(SqoopJobDataPublisher)obj; //构造需要导入Atlas 的元数据信息

SqoopJobDataPublisher.Data data =

new SqoopJobDataPublisher.Data(operation,options,tableName, startTime,endTime);

//通过org.apache.atlas.sqoop.hook.SqoopHook publisher.publish(data);

中的方法将元数据导入Atlas

}else{// 输出错误到日志

)catch(Exception ex){

由于在 Sqoop 中使用了Atlas 的类，因此，需要导入部分Atlas 的配置文件和jar 包。

复 制 %ATLAS\_HOME%/conf/atlas-application.properties 和 %ATLAS\_HOME%/hook/ sqoop/\*.jar 到Sqoop。

$cp /var/local/hadoop/atlas-0.7/conf/atlas-application.properties /var/local/hadoop/

sqoop-1.4.6/conf/

§cp /var/local/hadoop/atlas-0.7/hook/sqoop/\*/var/local/hadoop/sqoop-1.4.6/lib/

Atlas安 装 在 /var/local/hadoop/atlas-0.7,Sqoop 安 装 在 /var/local/hadoop/sqoop-1.4.6, 上述命令可参照集群自行更改。

到此，Sqoop HOOK 配置完成，下面会以 MySQL导入Hive 为例，来验证 Sqoop HOOK。

7.Sqoop 数据模型

Atlas 源码中的org.apache.atlas.Sqoop.model.SqoopDataModelGenerator 定义了默认支持 的 Sqoop 数据模型，其中有如下几种类型。

口 sqoop\_operation\_type (枚举类型)-值： [IMPORT,EXPORT,EVAL]

口 sqoop\_dbstore\_usage (枚举类型)-值：[TABLE,QUERY,PROCEDURE,OTHER]

**166** 第二篇 开源实现篇

口 sqoop\_process (类)-继承自[Process]- 属性： [name,operation,dbStore,hiveTable,

commandlineOpts,startTime,endTime,userName]

口 sqoop\_dbdatastore (类) - 继承自[DataSet]- 属性： [name,dbStoreType,storeUse, storeUri,source,description,ownerName]

每一个数据实体被创建后都使用唯一的名称属性来标识。它们提供命名空间，也可以 用于查询和血统。

8.用 Sqoop 从 MySQL 导入数据到 Hive 中

在linc-1 上，使用hdfs 用户，在任意的目录下执行以下命令：

$sqoop import -connect jdbc:mysql://linc-1:3306/hive -username root -password

MyNewPass4!-table TBLS -hive-import -hive-table sqoophook

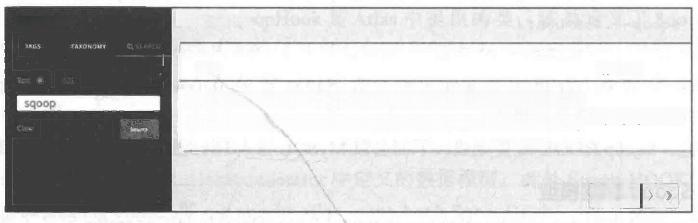
该命令将MySQL 中 Hive 数据库下的 TBLS 表导入Hive 中，该语句执行后会在Hive 中产生一个名为sqoophook 的表，该表的内容与TBLS 的内容一样。TBLS 表是 Hive 以 MySQL作为元数据存储数据库而产生的表。此处以该表作为测试表，读者也可以使用其他 的表进行测试。

在本书中使用的集群环境中， MySQL 安装在 linc-1,MySQL 的用户名为 root, 密码为 “MyNewPass4!”, 读者可根据自己的集群环境做适当更改。

执行该语句的时候如果出现缺失commons.lang3.jar 的错误，需要导入 commons.lang3. jar到 %SQOOP\_HOME%/lib 下。导入完成后，重新执行时如果提示 “HDFS 上 TBLS. java已经存在”,应先删除HDFS 上 TBLSjava, 相应的命令为： hdfs dfs -rmr <file-path>。 <file-path> 的具体值在提示错误时会提示。

**9.在** **Atlas UI上查看**

在Atlas 中搜索 sqoop\_process 或 sqoop 可以搜索到如图4-13所示的操作的元数据 信息。



1result for

**Type Name**

sqoopimport-comect jitbcmysqt//inc-1:3306/hve?zeroDateTimeehavior-convertToNull- table TBLS-hive-Import-hve-database detault-hive-table sqoophook-hive-Clister primary

Showing1tolofl entries

<<■

sqoop\_process

Apsche Allas

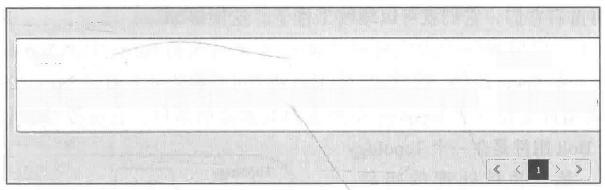
sqoop

Name

图4-13 Atlas 中搜索 sqoop 的结果

在Atlas 中搜索 soop\_dbdatastore 可以搜索到从MySQL 导入数据到 Hive 中操作的数据 源，如图4-14所示。

第 4 章 大 数 据 治 理 之 Apache Atlas 167



1 result for **sqoop\_dbdatastore**

**Name**

mysql-url jdbc:mysqt://linc-1:3306/hive?zeroDateTimeBehavior=convertToNull-table TBLS

Showing 1 to 1 of1 entrles

**TypeName**

sqoop\_dbdatastore

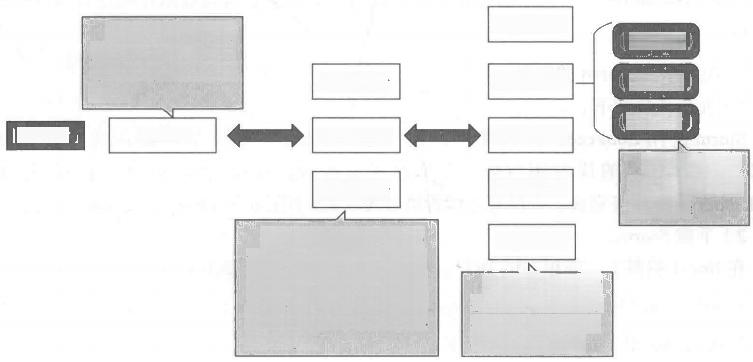
图4 - 14 Atlas 中搜索 sqoop\_dbdatastore 的结果

**4.2.5** **配置** **Storm** **通过** **Storm** **HOOK** **导入数据**

**1.Apache** **Storm** **简介**

Strom是一个分布式的、高容错的实时计算系统。Storm的集群表面上与Hadoop 的 集群非常像，但是Hadoop上面运行的是Map Reduce的Job, 而 Storm上面运行的是 Topology。Storm集群主要由一个主节点 (master node) 和一群工作节点 (worker nodes) 组 成，通过Zookeeper集群进行协调。

主节点通常运行一个后台程序Nimbus, 用于响应分布在集群中的节点、分配任务和 监测集群故障。工作节点同样会运行一个后台程序 Supervisor, 用于监听分配给它那台机 器的工作，收听工作指派并要求运行工作进程。每 一 个工作进程执行 一 个 Topology 的 一 个 子 集 ， 因 此 每 一 个 工 作 节 点 都 是 Topology 中 的 一 个 子 集 的 表 现 。 其 结 构 如 图 4 - 1 5 所 示 。



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nimbus: 主 控 节 点 ， 用于提交任务，分配 集群任务，集群监 控等  Master Nimbus | Zookeeper  Zookeeper  Zookeeper  Zookeeper: 集群中协 调，共有数据的存放(如 心跳信息，集群的状态和 配置信息)、Nimbus将分 配给Supervisor的任务写入 Zookeeper | |  |  | | --- | --- | | Supervisor  Supervisor  Supervisor  Supervisor  Supervisor | Worker  Worker  Worker  Worker: 运行 具体处理组件逻 辑的进程 |   Supervisor: 负责  接收Nimbus分配的  任务，管理属于自  己 的Worker 进 程 |

图4-15 Storm 结 构e

Nimbus和 Supervisor之间的所有协调工作是通过一个Zookeeper集群来完成的，并 且Nimbus 进 程 和 Supervisor 进 程 都 是 快 速 失 败 (fail-fast) 和 无 状 态 的 。 所 有 的 状 态 要 么 在 Zookeeper 里 ， 要 么 在 本 地 磁 盘 上 。 因 此 ， 读 者 可 以 用kill 命 令 来 杀 死Nimbus 和 Supervisor

⊙<https://zh.hortonworks.com/apache/storm/#section>\_2



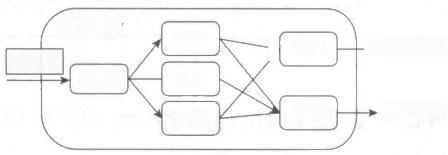
**168** 第二篇 开源实现篇

进程，然后再重启它们，它们就可以继续工作了。这使得 Storm 具有不可思议的稳定性。

在 Storm中，应用程序实现实时处理的逻辑被封装进Storm中的Topology 中， 一个 Topology是由一组Spout组件(数据源)和Bolt组件(数据操作)通过Stream Groupings进行 连接的。Spout组件是在一个Topology中产生源数据流的组件，它从源处读取数据并放入

Topology 中 ；Bolt 组件是在一个 Topology 中接收数据，然后执行处理的组件。 Topology 结构如图4-16所示。

Stream 是 Storm中的 一 个核心概 念， Storm 将输入的数据看成流，它是以 Tuple 为单位组成的 一 条有向无界的数据 流，如图4-17所示。



Topology

→实时结果

数据流

Spout

实时结果

Bolt

Bolt

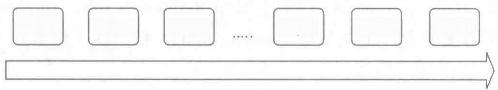
Bolt

Bolt

Bolt

元

图4-16 Storm 的 Topology 结构e



Tuple Tuple Tuple Tuple Tuple Tuple

图4-17 Storm 的数据流图

Storm 和 Atlas 整合的目的是使 Atlas 可以捕捉到 Storm 的操作的拓扑元数据。

下面在介绍完 Apache Storm的安装后，会就两个主要的内容介绍 Storm HOOK, 一 个

是能够表示出Storm 中相关信息的数据模型，另一个是能够将元数据存储进Atlas 的 Storm

HOOK。

**2.Apache** **Storm** **的安装**

1)先决条件如下。

Storm 使用Zookeeper 协调集群，由于Zookeeper并不用于消息传递，因此Storm 给Zookeeper带来的压力相当小。如果读者已经按照4.2.1节完成Atlas 的安装，那么 Zookeeper 已经安装完成，若没有，读者请参考5.2.5节完成Zookeeper 的安装配置。

2 ) 下 载 Storm。

在 linc-1机器上，使用hdfs用户，在/home/hdfs/目录下执行以下命令下载 Storm:

s wget <http://mirror.bit.edu.cn/apache/storm/apache-storm-1.0.0/apache-storm-1.0.0.tar.gz>

解压 Storm至/var/local/hadoop下：

$tar zxvf /home/hdfs/apache-storm-1.0.0.tar.gz

$mv /home/hdfs/apache-storm-1.0.0 /var/local/hadoop/

3 ) 修 改 storm.yaml 配置文件。

在 linc-1 上 ， 使 用 hdfs 用户，在任意目录下执行以下命令：

θ <https://zh.hortonworks.com/apache/storm/#section>\_2

第4章 大数据治理之Apache Atlas 169

s vim /var/local/hadoop/apache-storm-1.0.0/conf/storm.yaml

将文件的内容修改如下(忽略#所在的说明行):

#Storm 所依赖的Zookeeper 集群的地址列表，本集群依次为： linc-1,linc-2,linc-3

storm.zookeeper.servers:

-"linc-1"

-"linc-2"

-"linc-3"

#工作节点需要知道哪台机器是主节点的候补者。为了从这些节点下载 Topology 的 jar 和confs, 本书的集群主节点为linc-1

nimbus.seeds:["linc-1"]

#Storm 所依赖的Zookeeper集群的端口号默认为2181,如有更改请查看 Zookeeper 配置

storm.zookeeper.port:2181

#Nimbus 的 thrift 监听端口

nimbus.thrift.port:6627

#Storm UI的服务端口号

ui.port:8080

#Storm logviewer 端口号

logviewer.port:8000

#Storm DRCP的相关端口号

drpc.port:3772

drpc.invocations.port:3773

drpc.http.port:3774

#拓扑提交时使用的通知插件的类，即实现 HOOK 的类

storm.topology.submission.notifier.plugin.class:"org.apache.atlas.storm.hook. StormAtlasHook"

#Atlas 集群名称，这个名称没有特别的要求

atlas.cluster.name:"atlas\_cluster"

4)修改 storm\_env.ini 文件。

在 linc-1 机器上，使用hdfs 用户，在/var/local/hadoop/apache-storm-1.0.0/conf 目录下 执行以下命令：

$vim storm\_env.ini

在该文件中添加如下内容：

STORM\_JAR\_JVM\_OPTS:"-Datlas.conf=/var/local/hadoop/atlas-0.7/conf/"

170 第二篇 开源实现篇

该参数是让 Storm 能够找到 Atlas的配置文件，从而读取 Atlas 的相关配置信息。 5)拷贝 Atlas HOOK依赖包到$storm/extlib目录下。

在 linc-1上，使用 hdfs用户，在任意目录下执行以下命令：

9 cp /var/local/hadoop/atlas-0.7/hook/storm/\*.\*/var/local/hadoop/apache-storm-1.0.0/ extlib

6)拷贝storm 目录到其他节点。

在 linc-1上，使用 hdfs用户，在任意目录下执行以下命令：

$scp -r /var/local/hadoop/apache-storm-1.0.0/hdfselinc-2:/var/local/hadoop

§scp -r /var/local/hadoop/apache-storm-1.0.0/hdfselinc-3:/var/local/hadoop

说明：本书使用的集群列表为：linc-1 、linc-2 、linc-3, 因此只将 Storm目录拷贝到这 些集群上，读者可根据自己的集群配置，拷贝Storm 到对应的机器上。

7)为各节点添加环境变量。

在 Storm集群的各个节点中，使用 hdfs 用户，在任意目录下执行以下命令： $sudo vim /etc/profile

在该文件的末尾添加以下内容：

export STORM\_HOME=/var/local/hadoop/apache-storm-1.0.0

export PATH=SPATH;SSTORM/bin

添加完后刷新环境变量：

s source /etc/profile

8)启动 Storm Nimbus

在linc-1 上，使用hdfs 用户，在任意目录下执行以下命令：

$storm nimbus 6

9)启动 Storm Supervisor。

在 Storm集群的所有节点上，使用 hdfs 用户，在任意目录下执行以下命令：

s storm supervisor &

10)启动 Storm ui。

在 linc-1上，使用 hdfs 用户，在任意目录下执行以下命令：

$storm ui 6

11)测试 Storm传入元数据。

如果 Storm 能够正常启动，说明 Storm 已经安装成功，但是需要一个步骤来测试 Storm HOOK 是否配置成功。

为了便捷地测试Storm 传入元数据，本书提供了可执行jar 包，将本书提供的 storm- starter-1.0.0jar 拷贝至%STORM\_HOME%/examples/storm-starter/下，完成后执行以下命令：

第4章 大数据治理之 Apache Atlas **171**

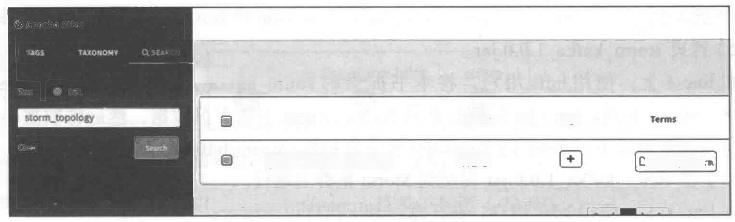
$/var/local/hadoop/apache-storm-1.0.0/bin/storm jar /var/local/hadoop/apache-storm-1.0.0/ examples/storm-starter/storm-starter-1.0.0.jar storm.starter.StatefulTopology

production-topology remote

完成后可以在Atlas Web 端使用 DSL 模式搜索 storm\_topology, 可以看到如图4-18所 示的内容。

**3.数据模型**

一个数据模型代表Atlas 的一个数据类型，它包括在拓扑图中各种节点的描述信息，例 如 Spout 和 Bolt 以及相应的生产者和消费者的类型描述。

1 result for storm\_topology

Tags

Owner

Name

Assign Ten

hdfs

production-topology

owng1totdi min

图4-18 Atlas中搜索storm\_topology 的结果

在 Atlas 中 与Storm 相关的数据类型有以下几种，这些数据类型在org.apache.atlas. storm.model.StormDataModel 中定义。

口 storm\_topology: 代表粗粒度拓扑，storm\_topology 从 Atlas 的 Process 类型派生出，

因此可以用于告知Atlas 数据的数据血统。

口 kafka\_topic,jms\_topic,hbase\_table,hdfs\_data\_set: 这些数据集都从 Atlas 的 Dataset

类型派生，因此它们都能够作为血统图的终点。

口 storm\_spout: 有输出的数据生产者，如Kafka 、JMS 等。

口 storm\_bolt: 有输入和输出的数据消费者，如 Hive 、HBase 、HDFS 等。

如果 Storm HOOK 用到了一些Atlas Server 没有的数据类型，它会自动注册依赖的数据 模型，如 Hive 数据模型。

**4.Storm HOOK**

(1)限制

只有新的拓扑被提交时才会在 Atlas 中注册，而拓扑的生命周期的改变不会通知Atlas。 当一个新提交的拓扑需要被捕捉的时候，需要Atlas Server 保持在线状态。

现在的 HOOK 不支持捕捉用户自定义的 Spouts 和Bolts 的数据血统。

(2)Atlas 监控 Kafka->Storm->HDFS 验证

1)创建 Kafka“test”Topic。

在linc-1 上，使用hdfs 用户，在/var/local/hadoop/kafka\_2.10-<0.10.0.0>/bin 目录下执行

172 第二篇 开源实现篇

Text

DSL

以下命令：

$./kafka-topics.sh --create --replication-factor 3 --partitions 1 --topic test

--zookeeper linc-1:2181

查看 topic 是否创建成功：

$./kafka-topics.sh --zookeeper linc-1:2181 --list

执行之后可以看到如下内容：

[hdfselinc-1 bin]$./kafka-topic.sh -zookeeper linc-1:2181 -list

Atlas\_ENTITIES

Atlas\_HOOK

2)拷贝 storm\_kafka\_ 1.0.0.jar。

在linc-1 上，使用hdfs 用户，将本书提供的storm\_kafka-1.0.0jar 拷贝到/home/hdfs 目录下。 storm\_kafka-1.0.0jar 能够实现从Kafka 的 test 主题下读数据，然后将数据存储到 hdfs 的/storm 目录下，storm 的 Topology 名字为kafka-storm-hdfs。

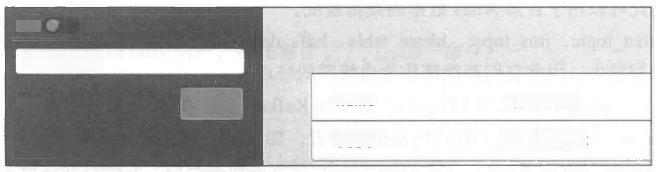
( 3 ) 将 storm\_kafka\_ 1.0.0.jar 提交到 Storm 集群上运行。

在linc-1 上，使用hdfs 用户，在任意目录下执行以下命令：

$/var/local/hadoop/apache-storm-1.0.0/bin/storm jar /home/hdfs/storm\_kafka-

1.0.0.jar xyz.xujie.zte.MyTopology

完成后在 Atlas 的 Web UI 上 的Text 模式下搜索 kafka\_topic 、hdfs\_path, 或者 storm\_ topology, 可以得到如图4-19、图4-20和图4-21 所示的结果。



Text DSL

1 result for kafka\_topic

kafka\_topic

**Name**

**test**

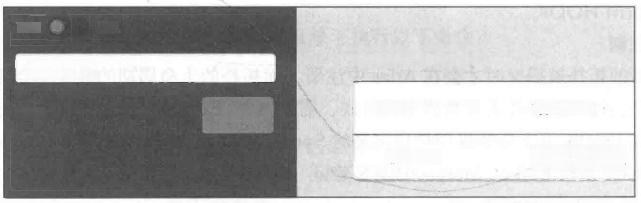
**Type Name**

kafka\_topic

Search

Clear

图4-19 Atlas中搜索 kafka\_topic 的结果



1 result for **hdfs\_path**

Name

storm

**Type Name**

hdfs\_path

hdis\_path

Seircl

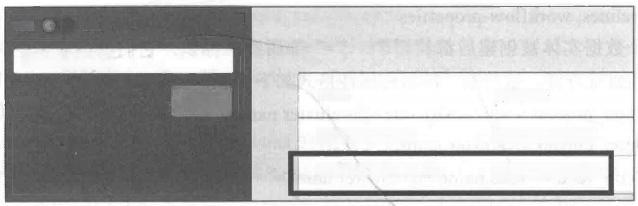
Cleaf

图4-20 Atlas 中搜索hdfs\_path 的结果

第4章大数据治理之Apache Atlas 1 73

Text

DSL



-2 result for **storm\_topology**

storm\_topology

**Name**

Clear

haha-topology

kafka-storm-hdfs

**TypeName**

storm\_topology

storm\_topology

Searh

图4-21 Atlas 中搜索 storm\_topology 的结果

打开 kafka\_topic 下的 test 和 hdfs\_path 下 的 storm, 可以发现血统图，如图4-22所示。

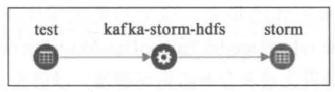


图4-22 Atlas 中的kafka-storm-hdfs 的血统

同时在 hdfs Web端可以看到新建的 storm 目录，如图4-23所示。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | |  |
|  | | | | | | | Col |
| Pemisrian 0mm Grenp Sire Replicatiam Bled Sine ame | | | | | | | |
| drwxr-xr-X drwxr-xr-X | hdfs  hdfs | supergroup  supertroup | 0 B  0B | 0  0 | 0B  0B | hbase  solr | |
| drwxr-xr-x hafs  drwxr-xr-x hdfs | | supergroup  supergroup | CF  0B | 0  0 | UF  0 B | storm  test | |

图4-23 hdfs 文件系统 Web 端的目录结构

**4.2.6** **配置** **Falcon** **通过** **Falcon** **HOOK** **导入数据**

配置 Falcon HOOK之前，请先参考3.2节，完成 Falcon 的安装。

**1.Falcon Model**

Falcon 数据模型在Atlas 的 org.apache.atlas.falcon.model.FalconDataModelGenerator 中 定义，其主要类型有以下几种。

口 falcon\_cluster(ClassType)- 继承自 Infrastructure- 属性： timestamp,colo,owner,tags

口 falcon\_feed(ClassType)- 继 承 自 DataSet- 属 性 ：timestamp,stored-in,owner,

groups,tags

O falcon\_feed\_creation(ClassType)-继承自 Process- 属性： timestamp,stored-in,owner

**口** falcon\_feed\_replication(ClassType)- 继承自 Process- 属性： timestamp,owner

口 falcon\_process(ClassType)- 继承自 Process- 属 性 ：timestamp,runs-on,owner,tags,

174 第二篇 开源实现篇

pipelines,workflow-properties

每一个数据实体被创建后都使用唯一的名称属性来标识。它们提供命名空间，也可以 用于查询和血统查询。这些唯一的名称属性格式如下：

口 falcon\_process-<process name>@<cluster name>

口 falcon\_cluster-<cluster name>

口 falcon\_feed -<feed name>@<cluster name>

口 falcon\_feed\_creation -<feed name>

口 falcon\_feed\_replication-<feed name>

**2.Falcon HOOK**

Falcon 能够监听 Falcon 实体的提交，这个功能能够用于在Atlas 中添加相关实体， Atlas 使用在 org.apache.atlas.falcon.model.FalconDataModelGenerator 定义的数据模型来导 入 Falcon 的数据。HOOK 的提交请求会发送到线程池，以防止出现 Falcon 的命令执行。线 程将实体作为一个消息发送到通知服务器， Atlas服务器从通知服务器上读取这些信息，并 将这些实体注册在自己的服务中。

下面介绍如何配置 Falcon HOOK。

1)修改 Falcon 的 startup.properties 文件。

在linc-1 上，使用 hdfs 用户，在任意目录下执行以下命令：

$vim /var/local/hadoop/falcon-0.9/conf/startup.properties

**修改\*.application.services 的内容如下：**

\*.application.services=org.apache.falcon.security.AuthenticationInitializationService, org.apache.falcon.workflow.WorkflowJobEndNotificationService,

org.apache.falcon.service.ProcessSubscriberService,

org.apache.falcon.service.FeedSLAMonitoringService,

org.apache.falcon.service.LifecyclePolicyMap,

org.apache.falcon.entity.store.ConfigurationStore,

org.apache.falcon.rerun.service.RetryService,

org.apache.falcon.rerun.service.LateRunService,

org.apache.falcon.metadata.MetadataMappingService,

org.apache.falcon.service.LogCleanupService,

org.apache.falcon.service.GroupsService,

org.apache.falcon.service.ProxyUserService,

org.apache.atlas.falcon.service.AtlasService

2)修改 Falcon 的 falcon-env.sh 文件。

在 linc-1 上，使用 hdfs 用户，在任意目录下执行以下命令：

$vim /var/local/hadoop/falcon-0.9/conf/falcon-env.sh

修改的内容如下，该配置用于使Falcon Sever的 JVM 能够找到Atlas 的配置信息，以 供 Falcon HOOK 使用。

第4章 大数据治理之Apache Atlas 17 5

export FALCON\_SERVER\_OPTS="SFALCON\_SERVER\_OPTS -Datlas.conf=/var/local/hadoop/

atlas-0.7/conf

本书的 Atlas安装在/var/local/hadoop/atlas-0.7/conf/ 目录下。

3)添加相应支持包的软链接。

Falcon Server除了需要读取部分Atlas 的配置文件外，还需要使用Atlas 中的一些jar 包，这部分jar 包 在 %ATLAS\_HOME%/hook/falcon/下。在%FALCON\_HOME%/server/ webapp/falcon/WEB-INF/lib/下创建%ATLAS\_HOME%/hook/falcon/\*的软链接，操作命令 如下：

S 1n -s /var/local/hadoop/atlas-0.7/hook/falcon/\*/var/local/hadoop/falcon-0.9/ server/webapp/falcon /WEB-INF/lib/

命令执行后可以在%FALCON\_HOME%/server/webapp/falcon/WEB-INF/lib/下看见相 应软链接。

4)参数调整。

Atlas 的 atlas-application.properties 文件中提供了一些属性，控制Falcon HOOK 的线程 池和通知系统的性能。本集群通常采用默认值，读者可根据自己的需要，对这些参数进行 调整。详细介绍请参考4.4节。

**3.测试** **Falcon Hook**

Falcon HOOK能够监听Falcon提交 falcon\_cluster、falcon\_feed 、falcon\_feed\_creation、 falcon\_feed\_replication。本书将会以提交 falcon\_cluster 为例，测试 Falcon HOOK 能够正常 使用，提交的 falcon\_cluster不用于任何 Falcon任务，仅仅用于测试 Falcon HOOK的状态。

创建需要提交的 cluster 文件。

在linc-1上，使用 hdfs 用户，在/home/hdfs下执行以下命令：

s vim local-cluster.xml

在该文件中添加如下内容：

trluster"description="testcluster"name="testcluster"xmlns=

"uri:falcon:cluster:0.1">

<intce type="readonly"endpoint="hdfs://linc- 1:9000"version="2.6.0"/>

<interface type="write"endpoint="hdfs://linc-1:9000"version="2.6.0"/>

<interface type="execute"endpoint="linc-1:8021"version="2.6.0"/>

<interface type="workflow"endpoint="http://linc-1:11000/oozie/"version=

<int0"/> type="messaging"endpoint="tcp://linc- 1:61616?daemon=true"

version="5.4.3"/>

</interfaces>

<locations>

<location name="staging"path="/projects/falcon/staging"/><!—必须-->

**176 第二篇** **开源实现篇**

<location name="temp"path="/projects/falcon/tmp"/><!-可选-->

<location name="working"path="/projects/falcon/working"/><!一可选--> </locations>

<properties>

</properties>

</cluster>

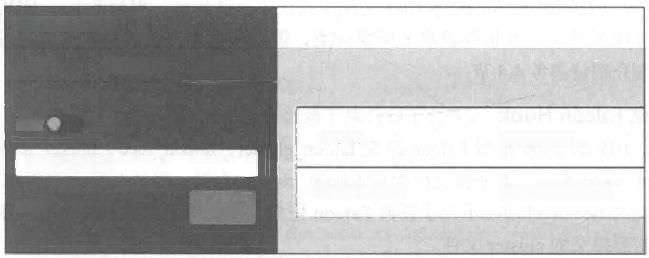
在上一章中，/projects/falcon/staging 、/projects/falcon/tmp 、/projects/falcon/working 已 经创建。如果在运行中出现权限错误，可修改 staging 权限为777,working 权限为775。如 果不存在，则在 hdfs 上创建相应的文件夹并修改其访问权限。

在 %Falcon\_HOME% 目录下，执行以下命令提交 cluster 实体：

$bin/falcon entity -submit -type cluster -file /home/hdfs/local-cluster.xml

若提示 falcon/default/Submit successful(cluster)local-cluster 则表示提交成功。

提交完成之后可以在Atlas 面板 Text 模式下搜索 falcon\_cluster, 可以搜索到提交的 cluster, 如图4-24所示。



Apache Atlas

3 result for **falcon\_cluster**

Text DsL

**Type Name**

falcon\_clusten

falcon\_cluster

Clear

TAXONOMY Q.5EARCH

testcluster

**Name**

Seach

TAGS

图4-24 在 Atlas 下搜索 falcon\_cluster

**4.3** **Apache Atlas 的场景设计**

**4.3.1** **Atlas** **总场景介绍**

看完前面的章节，读者应该对大数据家庭的组件有了一定的了解，特别是在配置完 Atlas 后，可能会想这个组件的用处是什么?什么是元数据处理?什么是元数据?

前面的章节已经对元数据做了简单的介绍，在这一节由于元数据起到了重要的作用， 因此做一个更加详细的介绍。元数据 (Metadata) 又称中介数据、中继数据，是描述数据的 数据 (data about data),主要是描述数据属性(property) 的信息，用来支持如指示存储位 置、历史数据、资源查找、文件记录等功能。元数据算是一种电子式目录，为了达到编制 目录的目的，必须描述并收藏数据的内容或特色，进而达到协助数据检索的目的。

元数据是描述信息资源或数据等对象的数据，其使用目的在于：识别资源；评价资源；

第4章大数据治理之Apache Atlas 177

追踪资源在使用过程中的变化；简单高效地管理大量网络化数据；实现信息资源的有效发现、 查找、 一体化组织和对使用资源的有效管理。元数据的基本特点如下：

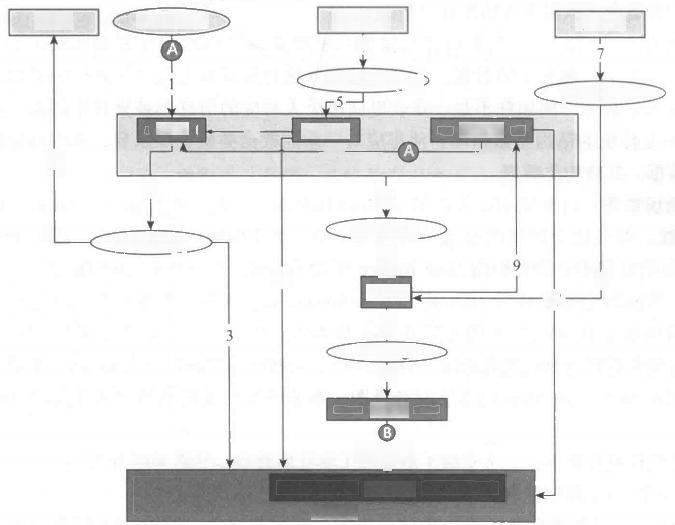
1)元数据一经建立，便可共享。元数据的结构和完整性依赖于信息资源的价值和使用 环境，元数据的开发与利用环境往往是一个变化的分布式环境，任何一种格式都不可能完 全满足不同团体的不同需要。

2)元数据首先是一种编码体系。元数据是用来描述数字化信息资源，特别是网络信息 资源的编码体系，这导致元数据和传统数据编码体系有根本区别；元数据的最为重要的特 征和功能是为数字化信息资源建立一种机器可理解框架。

举例来说，元数据体系构建了某机构电子政务的逻辑框架和基本模型，从而决定了电 子政务的功能特征、运行模式和系统运行的总体性能。则该电子政务系统的运作都基于元 数据来实现，其主要作用有：描述功能、整合功能、控制功能和代理功能。

由于元数据也是数据，因此可以用类似数据的方法在数据库中进行存储和获取。如果 提供数据元的组织同时提供描述数据元的元数据，将会使数据元的使用变得准确而高效。 用户在使用数据时可以首先查看其元数据，以便能够获取自己所需的信息。

现在读者已经了解元数据了，但是对元数据在组件中的流通并没有一个成型的概念。如 图4-25所示是 Atlas 与其他组件配合使用的总场景图，它从横向的数据来源以及纵向的数据 生命周期展示了元数据在大数据组件中的具体流通。分场景的部分会在后面进行详细说明。



MySQL

4

Mysol Table

Opermte Dat

6 · Sqoop

10-

HOOK

l!

MetaData

HiveTable

16

Kafka:

13

\*

Message

14

Atlas

15Entity



HBase

HDFS

Local File

**Storm**

Hive

Source

Hive

20

2

17

l

8

-

图4-25 Atlas 总场景图

178 第二篇 开源实现篇

图4-25中最上面一层是数据的来源，数据的来源是多种多样的。Sqoop 、Storm 、Kafka、 Atlas 等都是用来处理元数据的一些组件。长椭圆代表的则是中间数据，是数据在组件间 流通的不同形式。最大的矩形是前面提到过的Atlas 的精髓HOOK, 在这里包含Hive 、 Sqoop 、Storm 这几个组件，而实际情况则不是包含关系。图中并没有刻意凸显 Atlas 的 作用，但是将它作为管理元数据的工具，Atlas是大数据管理体系中作为中流砥柱的一环。

**4.3. 2** **Atlas** **非实时数据场景**

在实际的生活工作中，需要处理的数据是多种多样的，使用的存储手段也是五花八门 的。我们不可能穷尽所有的应用情况，只能从中抽象出几类主要的场景。

首先是非实时的数据场景，什么是非实时数据?例如，许多在线游戏的服务器在内测 或者公测时会搜集很多在线玩家的操作日志，这些日志会被提供给工程师们进行数据分析， 来排除玩家在游戏中遇到的一些 bug。这些 bug 的排除能很好地改善游戏的体验，并且极有 可能避免损失。当然这些日志可能会保存在本地，也可能保存在 MySQL 之类的数据库中。

我们先来分析保存在本地的数据。这些数据首先会被导入Hive 中 ，Hive 是基于 Hadoop 的一个数据仓库工具，可以将结构化的数据文件映射为一张数据库表，并提供简单 的 SQL查询功能，可以将SQL 语句转换为 MapReduce 任务进行运行。其优点是学习成本 低，可以通过类SQL 语句快速实现简单的 MapReduce 统计，不必开发专门的 MapReduce 应用，十分适合于数据仓库的统计分析。

可能有人会问，为什么不用将数据导入更加常见的Oracle 或者 SQLServer 这样的数 据库呢?事实上，现实中的数据，并没有被刻意地处理成结构化的数据，更多的情况是数 据呈杂乱无章状态，例如并不是一张表里面每个人对应的信息都被整齐地填满。因此使用 Hive 而不是传统的结构型数据库，使得应对复杂的数据类型更加从容，如结构化数据、半 结构化数据、非结构化数据等。

回到场景图，由于此时已经配置了Hive HOOK, 因此，将数据导入Hive 的操作会被 Atlas 捕捉，并且这个捕捉的过程是自动完成的，不需要输入任何命令。可能读者在前面 Atlas 安装的过程中会看到利用Hive Bridge 导入元数据，这与Hive HOOK导入元数据会产 生冲突，当同时配置完 Hive HOOK 和Hive Bridge 后会发现，数据都会自动完成导入，但 不能分辨到底是HOOK 导入的还是Bridge 导入的，因此提醒读者注意区分。Hive 生成的 数据表最终会存到分布式文件系统 (HDFS) 中，当然这个过程同样会被 Atlas 捕获。

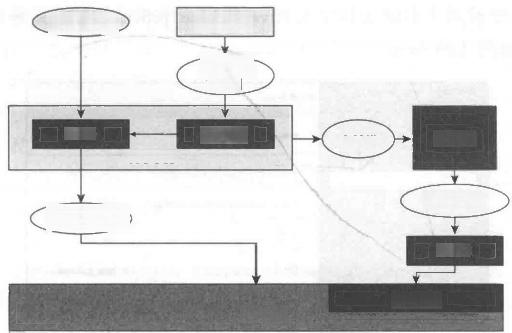
如图4-26所示是 Atlas 的非实时场景图，本节将按照该场景图完成非实时数据操作的 监控。

具体操作过程如下：

1)先在 Hive 里创建表。

在Atlas 及其依赖服务正常启动的前提下，在主节点上，使用hdfs 用户在任意目录下 进行建表操作。

第 4 章 大 数 据 治 理 之Apache Atlas 1 79



MySQL

MysoL

Tabel

Kaika

HOOK

Message

Hive Table)

Atlas

HBase

HDFS

Meta

Data

Local File

Sqoop

Hive

图4-26 Atlas 非实时场景图

S hive

hive>create table czl (id int, name string,age int,tel string) ROW FORMAT hive>DELIMITED FIELDS TERMINATED BY '\t'STORED AS TEXTFILE;

创建一个有4个字段的表，存储用户的基本信息，ROW FORMAT DELIMITED FIELDS TERMINATED BY 可以设定建表的格式，而不管有多少个字段；\t 表示外部数据用这个方 式来间隔彼此。

2)在本地创建测试文件。

在主节点上，使用 hdfs 用户，在任意目录下执行以下命令：

s cd $touch $vim

czl.txt

czl.txt

由于Hive表中规定了表数据以\t 分隔，所以数据列之间用tab 分隔。 3)将文件导入Hive 表。

在主节点上，使用 hdfs 用户，在任意目录下执行以下命令：

$cd

$hive

hive>load data local inpath 'czl.txt'into table cz1;

在 Atlas 的 Web UI中可以观察到 Hive导入的表的元数据信息和导入表的操作过程，如 图4-27所示。

4)将文件从 Hive导入HDFS。

在主节点上，使用hdfs 用户，在任意目录下执行以下命令：

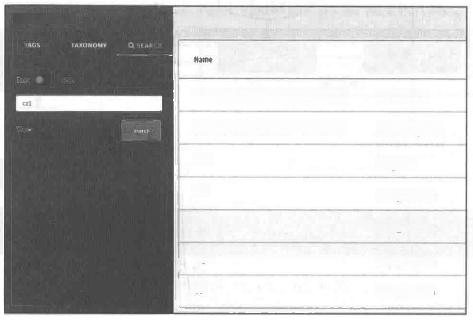
$hive

hive>insert overwrite directory'/home/hdfs/cz'select \*from cz1;

将 数 据 从 cz1 表 导 入 hdfs 的 /home/hdfs/cz 目 录 。 通 过 命 令 hadoop fs-ls /home/hdfs/cz 查

180 第二篇 开源实现篇

看。在Atlas 中可以看到从本地文件导入Hive 的 hive\_process 和 Hive 导出到 hdfs 的 hive\_ process 与血统，如图4-28所示。



Apsiche Atlas

Type Mame

load data local inpath'tz ttinto tableal

4db555a5-7b0-4220-b703-744d86167990

cz1

tel

name

age

id

hive\_process

hive storagedes

hve\_table

hive\_column

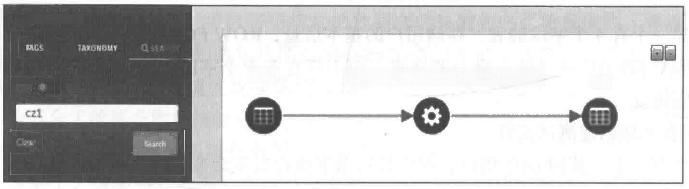
hive\_column

hive\_column

hive\_column

7esultorcz

图4-27 Atlas 对 Hive操作的捕获图

QApache Atlas

LINEAGE

insert overwrite di...

czl

CZ

bst

let

图4-28 Atlas捕获操作后生成的血统图

场景中的另一条线路是从关系型数据库例如 MySQL中将数据导入Hive, 因为关系型 数据库在生产及生活中占有很大的比例，例如公司员工的信息、银行的账户信息等，各种 关系型数据库也是层出不穷。场景中以 MySQL 为典型，便于读者理解。当然结构化的数据 存在了关系型数据库中，想将其导入非关系型数据库，可以借助一个叫作 Sqoop 的工具。

Sqoop是一款开源的工具，主要用于在Hadoop(Hive) 与传统的数据库 (MySQL、 PostgreSQL 等)间进行数据的传递，可以将一个关系型数据库(如 MySQL 、Oracle 、Postgres 等)中的数据导入Hadoop 的 HDFS中，也可以将HDFS 的数据导入关系型数据库中。

接下来的过程就如同之前从本地导入数据到 Hive一样，只不过除了Hive 的处理过程 会被捕获外， Sqoop 的操作也会被 Atlas 捕获。被捕获的元数据会被传到Kafka 工具。

Kafka是一种高吞吐量的分布式发布订阅消息系统，它可以处理消费者规模的网站中的 所有动作流数据。这种动作(网页浏览、搜索和其他用户操作)是在现代网络上的许多社会 功能的一个关键因素。这些数据通常由于吞吐量的要求而通过处理日志和日志聚合来解决。 对于像 Hadoop一样的日志数据和离线分析系统，但又有实时处理的限制，通过Kafka 可以 解决问题。Kafka 的目的是通过Hadoop 的并行加载机制来统一线上和离线的消息处理，也

第4章 大数据治理之Apache Atlas 181

是为了通过集群机来提供实时的消费。

Kafka 将搜集的元数据打包成一个Message, 并将其传送给 Atlas,Atlas 会产生相应的 实体 (Entity), 这个实体最后会被存储在HBase 中。

具体操作如下：

1)创建 SQL 表。

在主节点上，使用hdfs 用户，在任意目录下创建数据库，并且插入数据。这个过程需 要有一些数据库操作基础，数据库的密码读者可自行设置。

$mysql -uroot -pMyNewPass4!

mysql>CREATE TABLE person(id int,stockname varchar(20),value int,cash mysql>

int,primary key(id));

#以下建表过程不需要完全按照此处的步骤，读者可以新建自己想要的表

mysql>insert into person values(10001,'ZGYH',120000,80000);

mysql>insert into person values(10002,'ZSYH',1204400,32000);

mysql>insert into person values(10003,'JSYH',21100,53200);

mysql>insert into person values(10004,'NYYH',324100,7500);

mysql>insert into person values(10005,'GSYH',376100,59700);

mysql>insert into person values(10006,'BJYH',4346100,545430);

mysql>select \*from person;

2 ) 用Sqoop 从 MySQL 导入数据到 Hive 中。

在主节点上，使用 hdfs 用户，在任意目录下执行以下命令：

$sqoop import -connect jdbc:mysql://XXX:3306/test -username root -password admin -table person -hive-import -hive-table animal

从Atlas Web UI可以查到 Sqoop 导入数据的过程和 Hive 导入数据的过程及血统图，如 图4-29、图4-30和图4-31所示。

3)将文件从 Hive 导入 HDFS。

在主节点上，使用 hdfs 用户，任意目录下执行以下命令：

$hive

hive>insert overwrite directory'/home/hdfs/person'select \*from person;

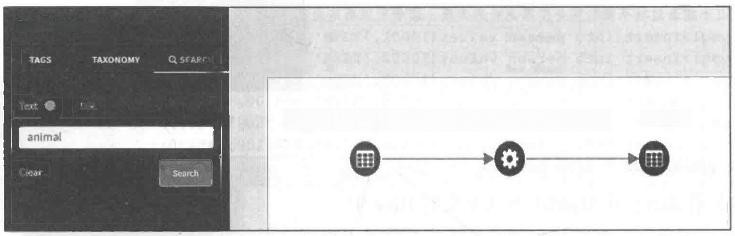
|  |  |  |
| --- | --- | --- |
| Apache Atlas  AH3 AXQNQNT | ao reult for person | |
| Name Type Name | |
| ext |
| load data inpath 'hdfs://zte-1;9000/user/hdfs/person'into table'person" hive\_process | |
| Serson  Car | mysql-url  person | jdbc:mysql:/zte-1:3306/test?zeroDateTimeBehavior=convertToNull-table  sqoop\_dbdatastore |
| 322d1108-53ec-4623-9800-9623755e6{3b hive\_storagedesc | |
| sqoop import-connect jdbc:mysqt//zte-1:3306/test?  zeroDateTmeBehavior=convertToNul-table person -hive-Import-hive-database sqoop\_process default-hive-table person -hive-dluster zteCluster | |
| person hive\_table | |

图4-29 Atlas 对 Sqoop 操作的捕获图

182 **第二篇** **开源实现篇**

|  |  |
| --- | --- |
| Apache Atlas  AG5  nima | 2result for animal |
| Name **Type** **Name** |
| animal hive\_table |
| sqoop import-comnect jctbcmyst/zte-1:3306/test?zeroDateTimeBehaviorFconvertioNull-  sqoop\_process  able person-hive-import -hive-database default -hive-table animal-hive-dluster primary |
| Showing 1 to 2of2enties |

图4-30 Atlas 对 Hive 操作的捕获图



Apache Atlas

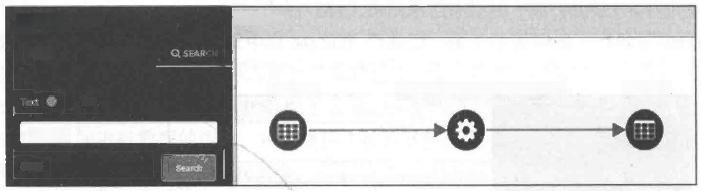
LUNEAGE

mysql-url jdbc:my...sqoop import --conn **animal**

图4-31 Atlas 捕获操作后生成的血统图

将数据从person 表导入hdfs 的/home/hdfs/person目录，可通过命令hadoop fs -ls / home/hdfs/person 查看导入 hdfs 的数据。

在Atlas中可以看到从本地文件导入Hive 的 hive\_process 和从 Hive导出到 hdfs的 hive\_process与血统，如图4-32所示。



Apache Atlas

TAGS TAXONOMY

person insert overwrite di.. person

DSt

person

Ccar

LINEAGE

图4-32 Atlas 捕获操作后生成的血统图

Atlas 对 Hive并不是完全控制的，只有以下操作能被它的 HOOK 捕获： 口 create database

create table/view,create table as select

口 load,import,export

DMLs(insert)

第4章大数据治理之Apache Atlas 183

口 alter database

口 alter view

口 alter table (skewed table information,stored as,protection is not supported) 同样 Atlas 对Sqoop 也仅仅支持 Hive Import 操作。

**4.3.3** **Atlas** **实时数据场景**

在了解完非实时数据的场景流程后，我们应该可以联想到与之对应的实时数据的场景。 海量数据处理使用的大多是鼎鼎大名的 Hadoop 或者 Hive, 作为一个批处理系统， Hadoop 以其吞吐量大、自动容错等优点，在海量数据处理上得到了广泛的使用。但是，Hadoop 不 擅长实时计算，因为它天然就是为批处理而生的，这也是业界一致的共识。由此而产生了 Storm 等一些系统来帮助 Hadoop 进行实时的计算处理。

Storm 被广泛应用于实时分析、在线机器学习、持续计算、分布式远程调用等领域。来 看一些实际的应用。

1)一淘——实时分析系统 pora: 实时分析用户的属性，并反馈给搜索引擎。最初，用 户属性分析是通过每天在云梯上定时运行的MR job来完成的。为了满足实时性要求， 一淘 希望能够实时分析用户的行为日志，将最新的用户属性反馈给搜索引擎，以便为用户展现 最接近其当前需求的结果。

2)携程——网站性能监控：实时分析系统监控携程网的网站性能。利用HTML5 提供 的 performance 标准获得可用的指标，并记录日志； Storm 集群实时分析日志和入库；使用 DRPC聚合成报表，通过历史数据对比等判断规则，触发预警事件。

试想下，如果一个游戏新版本上线，它用一个实时分析系统收集游戏中的数据，运营 者或者开发者可以在上线后几秒钟得到持续不断更新的游戏监控报告和分析结果，然后马 上针对游戏的参数和平衡性进行调整。这样就能够大大缩短游戏的迭代周期，加强游戏的 生命力。

回到实时数据的场景图，我们可以看到，从数据源获得的操作数据会被前面介绍的 擅长处理消费者规模的网站中的所有动作流数据的Kafka 搜集，之后打包成一个消息传 给Atlas, 这个过程与之前的场景大同小异。另一条路线是经过Kafka 之后，操作信息会 被 Storm 搜集传输给HDFS, 而这个过程会被 Atlas 的 Storm HOOK 捕获，之后还是经过 Kafka传递给 Atlas 。整个过程并不复杂，但是却可以处理实际场景中大量的问题。

如图4-33所示是 Atlas的实时场景图。

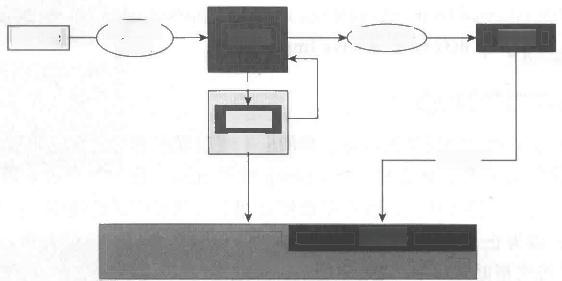
实时数据场景的具体操作在4.2.5节中已经介绍过，其中的第4部分 Storm HOOK 中的 Atlas 监控 Kafka->Storm->HDFS 验证就是该场景图的具体实现。

**4.3.4** **Hive** **数据表操作**

前面的场景中提到了Hive 的使用，这一节的场景只是延展Hive 的使用，来丰富Atlas

184 第二篇 开源实现篇

的功能。如图4-34所示为表拆分和合并场景图，本节将按照此场景图完成表拆分与表合并 的监控。



Operate

Data

Storm

Hook

Entity-

Hbase

HDFS

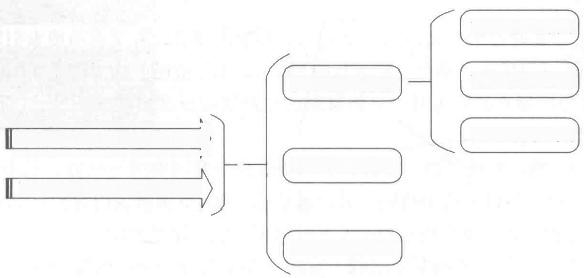
Message

Kafka

Source

Atlas

图4-33 Atlas 实时场景图



New HiveTable

New HiveTable

New HiveTable

Local file>Hive HiveTahle

New HiveTable

Local file->Hive:HiveTable

New HiveTable

New HiveTable

图4-34 表拆分和合并场景图

具体的操作过程如下。

**1.线路** **—** **:Hive 表合并**

1)在本地创建数据文件。

在Atlas 不相关服务开启前，在主节点上，使用hdfs 用户，在任意目录下执行以下命令：

$cd ~

s vim employee-1

输入如下内容：

1,employee1,23,man

2,employee2,22,woman

3,employee3,20,man

4,employee4,46,woman

vim employee-2

**输入如下内容：**

第4章大数据治理之Apache Atlas 185

5,employee5,21,man

6,employee6,22,woman

7,employee7,22,man

8,employee8,39,man

**2)在Hive 里创建表。**

**在主节点上，使用hdfs 用户，在任意目录下执行以下命令：**

s hive

hive>create table employee\_ 1(id int,name string,age int,sex

delimited fields terminated by ',';

hive>load data local inpath "/home/hdfs/employee-1"into table hive>select \*from employee\_ 1;

结果如图4-35所示。

string)row format

employee\_ 1;



hive-select \*from employee\_ 1;

0K

employeel 23 man

employee2 22 woman

amployee3 20 man

employee4 46 woman

Time taken:1.08 seconds,Fetched:4 row(s)

hive>

**图4-35** **Hive 中查询employee\_1 数据**

**创建表2:**

hive>create table employee\_2(id int,name string,age int,sex string)row format

delimited fields terminated by ',';

s load data local inpath "/home/hdfs/employee-2"into table employee\_2; s select \*from employee\_2;

查询表2,如图4-36所示。

hives select \*from employee\_2:

|  |  |  |
| --- | --- | --- |
| CK |  |  |
|  | employees  enployee6 | |  |  | | --- | --- | | 21 | man | | 22 | woman | |
| 7 | employee? | 22 man |
| 8 | employee8 | 39 man |
| Time  hive=[ | taken:0.865 | seconds,Fetched:4 row(s) |

图4-36 Hive中查询 employee\_2 数据

hive>exit;

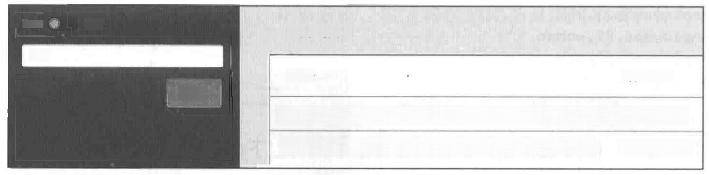
将本地数据导入Hive 中完成 Hive 表的创建。

创建完成之后在 Atlas 中搜索 load 可以查看数据的 load 过程，如图4-37所示。 3)Hive 表的合并。

在主节点上，使用hdfs 用户，在任意目录下执行以下命令：

$hive

186 **第二篇** **开源实现篇**

Text

load

Clear

DsL

Search

5 result for load

**Type Name**

Mame

load datalocallnpath "/home/hafs/employee-1"into table employee

1 hive\_process

hive\_process

load data localinpath"home/hafsyemployee-2"intotable employee\_2

图4-37 Atlas 捕获 Hive 操作

使用union all 关键字进行合并。

hive>create table employee\_union as select \*from employee\_ 1 union all select \*

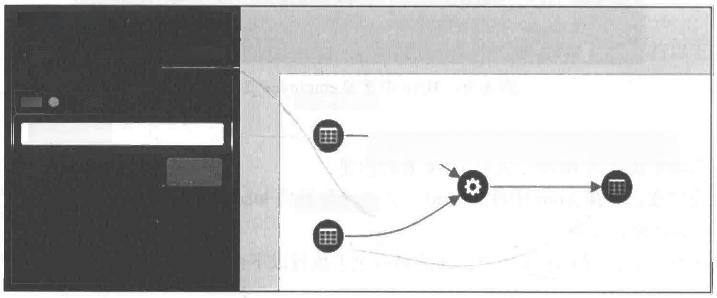
from employee\_2;

在 Atlas 的 Web UI中搜索 employee\_union, 便可看到 hive\_process, 如图4-38所示。

|  |  |  |  |
| --- | --- | --- | --- |
| TAXONOMV  Text Ds | Q SEARCH | 7result for employee\_unlon | |
| Name | Type Name |
| create ableemployee union as select"fromemployee I union al select'from employee2 | hive\_process |
| employee union | |
| 6Mf3c72-6abe-4d6b-9106-b915d03ada95 | hive storagedesc |
| employee\_union | hive\_table |
| sex | hive\_column |
| name | hive\_column |
| age | hive\_column |
| id | hive\_column |
|  |  |

图4-38 Atlas 捕获 Hive 操作

如图4-39所示为 employee\_union 的血统图。

Apache Atlas

TAGS TAXONOMY 5EARCI LINEAGE

DSL employee\_2

employee\_union

create table employ.. employee\_union

Search

Clear

employee\_ 1

图4-39 Atlas 捕获操作后生成的血统图

**第4章大数据治理之Apache Atlas** 187

hive>select \*from employee\_union;

如图4-40所示为查询结果。



select"fron

enployeel enployee2

employee3

enployeed employee5

enployee⁶

employee7

enplovens

taken:6.111

21 man

22 woman

22 man

39 man

seconds,Fetched:8 row(s)

employee union;

woman

*man*

woman

hives 0K

Time hivee

23 man

**图4-40** **Hive中查询employee\_union 表数据**

使用Join 关键字进行合并。

hive>Create table employee\_join as select el.id,el.name,el.age,el.sex from

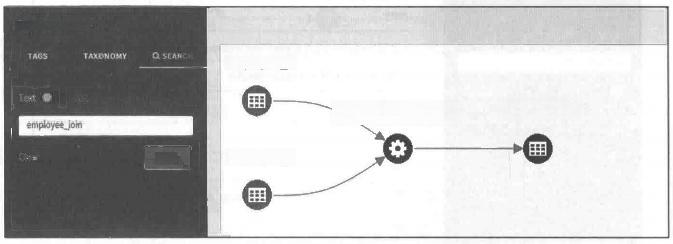
employee\_ 1 el join employee\_2 e2 on(el.age=e2.age);

在 Atlas 的 Web UI中搜索 employee\_join, 可以看到如图4-41所示的捕获图和图4-42

所示的血统图。

|  |  |  |
| --- | --- | --- |
| Apache Atlas  A55 TAXQNONY SEARCH  51 | 7 result for **employee\_Join** | |
|  | **Name** **Type** **Maime** |
| ceate table employee join as selectel.id,eL name,el.age,el.sex from employee1el  hive\_process  join employee\_2e2on (el.age=e2age) |
|  | | 24aBd24b-bf08-435a-a776-e0a543d4b07e hive\_storagedesc |
| 周 |  |
| employee\_join hive\_table |
| hive\_column sex |
| id hive\_column |
| age hive\_column |
| hive\_column  name |

图4-41 Atlas 捕获Hive 操作

UNEAGE

Apache Atlas

**employee\_2**

ceate able employ …

employee\_join

employee\_1

图4-42 Atlas 捕获操作后生成的血统图

**188** **第二篇** **开源实现篇**

**使用** **LEFT OUTER JOIN 关键字进行合并。**

hive>Create table employee\_join2 as select el.id,el.name,el.age,e2.sex from

employee\_ 1 el left outer join employee\_2 e2 on(el.age=e2.age);

在 Atlas 的 Web UI 中搜索 employee\_join2, 可以看到如图4-43所示的捕获图。

|  |  |  |
| --- | --- | --- |
| Apache Atlas  Q 5ARCH  iet DSL |  |  |
|  | Name Type Name |
| 1450b3f0-0fdc-4e30-a71f-c34cfad11eb2 hive\_storagedesc |
| ceatetable employee join2as selecteLid,eLname,el.age,e2.ser from employee1 hve\_process  el let outerjpin employee 2e2on(elage-e2age |
| employee\_jòn2 |
| employee\_join2 hive\_table |
| R 50rch |
| sex hive\_column |
| name hive\_column |
| age hive\_column |
| id hive\_column |

图4-43 Atlas 捕获 Hive 操作

进入employee\_join2, 可以看到它的血统图， 如图4-44所示。

**2.Hive** **表的拆分**

在主节点上，使用hdfs 用户，在任意目录下

执行以下命令：

$hive

1)拆分出表1:

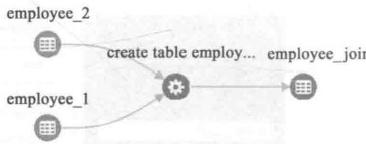
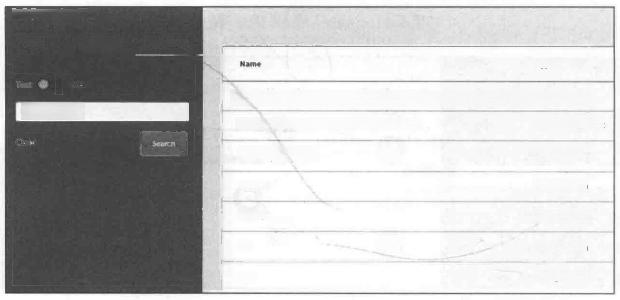


图4-44 Atlas 捕获操作后生成的血统图

hive>create table employee\_part1 as select \*from employee\_union where age<23; 在 Atlas 中搜索 employee\_part1, 可以看到搜索结果，如图4-45所示。



Atlas

7reultfor employee\_parti

TAxONOMV

TypeName

create tableemployee partl as select"fom employee union where age23 hive protess

employee\_art

758fa006(7b1-47bf-91dc-0a3e15d4202e

employee\_part1

sex

name

id

age

hve\_storagedes

hive\_table

hive\_column

hive\_column

hive column

hive\_column

Apache

a sEARC

TAGs

图4-45 Atlas 捕获 Hive 操作

第4章大数据治理之 Apache Atlas 189

进 入 employee\_part1 可以看到employee\_partl 的血统图，如图4-46所示。



**employee\_2**

create table employee …

employee\_union

**employee\_1**

create table employ..employee\_part1

图4-46 Atlas 捕获操作后生成的血统图

2)拆分出表2:

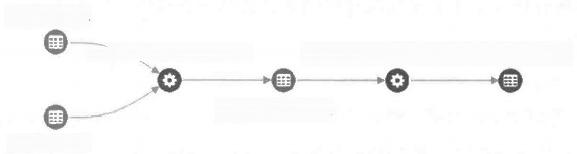
hive>create table employee\_part2 as select \*from employee\_union where age>=23;

在 Atlas 中搜索 employee\_part2, 可以看到搜索结果，如图4-47所示。

|  |  |  |  |
| --- | --- | --- | --- |
| Apache Atlas  TAGS TARON0MY Q SEARCH  Text 1E | 7 result for employee\_part2 | | |
|  | **Name** Type Name | |
| createtable employee\_part2 as select'from employee\_union where sex-=23 | hive\_process |
| employee part | | 9e4b1469-cd19-470f-ad8b-0861f59efcif hive\_storagedesc | |
| 0 |  | employee\_part2 | hive\_table |
| sex hive\_column | |
| age hive\_column | |
| name hive\_column | |
| hive\_column id | |

图4-47 Atlas 捕获 Hive 操作

进入 employee\_part2, 可以看到它的血统图，如图4-48所示。

employee\_2

create table employee … create table employ..employee\_part2



employee\_1

employee\_union

图4-48 Atlas 捕获操作后生成的血统图

3)拆分出表3:

hive>create table employee\_part3 as select \*from employee\_union where sex='man';

在 Atlas 中搜索 employee\_part3, 可以看到搜索结果，如图4-49所示。 进入 employee\_part3, 可以看到它的血统图，如图4-50所示。

190 **第二篇** **开源实现篇**

|  |  |  |  |
| --- | --- | --- | --- |
| Apache Atlas | |  | **Name** **Type** **Name** |
| TAG5  式 | TAXONOMY Q SEARCH  DSL | create table employee\_part3 as select\*from employee\_union where sex='man'hive\_process |
| efc3feof-037b-4d0e-9461-d7ff060595bf hive\_storagedesc |
| HP | | employee\_part3 hive\_table |
| Clea h | | age hive colume |
| **id** hive\_column |
| **name** hive\_column |
| sex hive\_column |

图4-49 Atlas 捕获 Hive操作



employee\_2

create table employ. …

employee\_1

employee union employee\_part3

create table employ.

图4-50 Atlas 捕获操作后生成的血统图

**4.4** **Apache** **Atlas** **优化与性能分析**

Apache Atlas 是 Apache 的孵化项目，虽然已经将其用于生产环境，不过还是希望广大 读者能在使用中继续探索。

在前面的部分了解了如何配置Atlas 以及Atlas 在场景中的具体使用后，我们结合官方 文档和对 Atlas 的理解，来对 Atlas 的性能进行分析和优化。

**1.JVM** **性能调优**

如果需要运行成百上千的大数据对象的话，需要配置调优。在文件 conf/atals-env.sh 中 执行以下命令：

export ATLAS\_SERVER\_OPTS="-server

1)SoftRefLRUPolicyMSPerMB 官方解释是 softly reachable objects will remain alive for some amount of time after the last time they were referenced.The default value is one second of lifetime per free megabyte in the heap。但笔者觉得没必要等1秒。

-XX:SoftRefLRUPolicyMSPerMB=0

2)CMSClassUnloadingEnabled 参数表示在使用CMS 垃圾回收机制时是否启用类卸载 功能。默认这个设置为不启用，若读者想启用这个功能，需要在Java 参数中明确设置。

如果读者启用了CMSClassUnloadingEnabled, 垃圾回收会清理持久代对象，移除不再

第4章大数据治理之Apache Atlas 19 1

使用的 classes 。这个参数只有在 UseConcMarkSweepGC 也启用的情况下才有用。

-XX:+CMSClassUnloadingEnabled

-XX:+UseConcMarkSweepGC

3)降低标记停顿：为了减少第2次暂停的时间，应开启并行remark:-XX:+CMSParallel- RemarkEnabled。如果 remark 还是过长的话，可以开启-XX:+CMSScavengeBeforeRemark 选 项，强制 remark 之前开始一次 minor gc, 减少 remark 的暂停时间，但是在 remark 之后也 将立即开始又一次minor gc。

-XX:+CMSParallelRemarkEnabled

4)查看每次 minor GC后新的存活周期的阈值：PrintTenuringDistribution 的输出显示 在survivor 空间里面有效的对象的岁数情况。阅读-XX:+PrintTenuringDistribution 输出的方 式是观察在每一个岁数上对象的存活的数量及其增减情况，以及HotSpot VM计算的任期阈 值是不是等于或者近似于设定的最大任期阈值。

-XX:+PrintTenuringDistribution 在 Minor GC的时候产生任期分布信息。它可以同 其他选项一 同使用，比如-XX:+PrintGCDateStamps 、-XX:+PrintGCTimeStamps, 以 及-XX:+PringGCDetails 。若调整 survivor 空间大小以获得有效的对象岁数分布，应该使 用-XX:+PrintTenuringDistribution。在生产环境中，它同样非常有用，可以用来判断 stop- the-world 的垃圾回收是否发生。

-XX:+PrintTenuringDistribution

5)若使用了标志 -XX:+HeapDumpOnOutOfMemoryError,JVM 会 在 遇 到OutOf- MemoryError 时拍摄一个“堆转储快照”,并将其保存在一个文件中。

-XX:+HeapDumpOnOutOfMemoryError

6)设置 Dump的路径。

-XX:HeapDumpPath=dumps/atlas\_server.hprof -Xloggc:logs/gc-worker.log -verbose:gc

7)启用GC 日志文件自动转储。

-XX:+UseGCLogFileRotation

8)设置GC 日志文件的循环次数。

-XX:NumberOfGCLogFiles=10

9)控制GC 日志文件的大小。

-XX:GCLogFileSize=1m

10)只要设置-XX:+PrintGCDetails, 就会自动设置-verbose:gc 和-XX:+PrintGC。

-XX:+PrintGCDetails

**192** 第二篇 开源实现篇

11)打印 GC 前后的详细堆栈信息。

-XX:+PrintHeapAtGC

12)打开-XX:+PrintGCTimeStamps 开关，可以了解这些垃圾收集发生的时间，自 JVM启动以后以秒计量。

-XX:+PrintGCTimeStamps"

#Atlas 服务器的Java 堆大小默认是1024M

#export ATLAS\_SERVER\_HEAP=

#indicative values for large number of metadata entities (equal or more than 10,000s)for JDK 7

#export ATLAS\_SERVER\_HEAP="-Xms15360m -Xmx15360m -XX:MaxNewSize=3072m -XX:PermSize=100M -XX:MaxPermSize=512m"

#indicative values for large number of metadata entities(equal or more than 10,000s)for JDK 8

#export ATLAS\_SERVER\_HEAP="-Xms15360m -Xmx15360m -XX:MaxNewSize=5120m -XX:MetaspaceSize=100M -XX:MaxMetaspaceSize=512m"

**2.Atlas** **中** **HOOK** **功能的调优**

以 Hive HOOK为例子，Hive HOOK将元数据传递给 Atlas 时，是使用独立线程来实 现的，%ATLASS\_HOME%/conf/atlas-application.properties 中有 一 些参数能够控制这些 HOOK 的线程池和通知系统的配置。

口 atlas.HOOK.hive.synchronous: 布尔类型，值为 true 表示 Atlas 使用同步方式运行 Hive HOOK,否则为异步模式。默认值是 false 。推荐设为 false, 能够防止出现 HOOK 提交时阻塞Hive 命令正常执行。

口 atlas.HOOK.hive.numRetries: 通知失败的情况下重试次数，默认值为3。 口 atlas.HOOK.hive.minThreads: 线程的最小数量，默认为5。

□ atlas.HOOK.hive.maxThreads: 线程的最大数量，默认为5。

口 atlas.HOOK.hive.keepAliveTime: 保持存活时间(毫秒级),默认值是10。 口 atlas.HOOK.hive.queueSize: 线程池的队列长度，默认值是10000。

正如配置项的说明一样，可以使用这些参数调节线程池的大小以及队列长度等，正常 情况下使用默认的配置即可满足需求，但是如果集群使用非常频繁，产生的元数据量很庞 大，默认的配置可能会无法满足需求。在这种情况下，可以更改这些参数进行调优。

以上配置为 Hive HOOK 的调优参数，此外 Falcon HOOK 也有对应的调优参数，其参 数内容基本相同。Falcon HOOK 调优参数如下：

口 atlas.HOOK.falcon.synchronous:boolean 值，为true 表示 Atlas 同步运行 Falcon

HOOK。默认为 false, 即异步模式，异步模式能够防止HOOK 提交时阻塞 Falcon 命令正常执行。

口 atlas.HOOK.falcon.numRetries: 通知系统失败时重复的次数。默认值为3。 O atlas.HOOK.falcon.minThreads:HOOK 的最小的线程数。默认值为5。

第4章大数据治理之Apache Atlas 193

口 atlas.HOOK.falcon.maxThreads:HOOK 的最大的线程数。默认值为5。 **O** atlas.HOOK.falcon.keepAliveTime: 存活时间，单位为毫秒。默认为10

口 atlas.HOOK.falcon.queueSize:HOOK 线程池使用的队列长度。默认为10000。

Sqoop HOOK 与 Storm HOOK 的调优参数在官网上没有相应的资料，暂时可能不支持 参数调优。

**4.5** **本章小结**

本章向读者介绍了大数据治理的元数据管理框架Apache Atlas,从大数据治理的理论内 涵开始，介绍了Atlas 的各项基本特性，包括大数据系统中的元数据管理、数据生命周期可 视化显示、数据血统的搜索、基于标签的商业数据分类等。在这些特性的基础之上，讲述 了Atlas 的发展历程与优势，并且为了帮助读者加深对Atlas 的理解，也对Atlas 的架构和 重要组成部分着重进行了介绍。

除了对Atlas 的理论介绍外，为了便于读者学习和研究如何使用大数据治理框架 Atlas, 本章也详细介绍了Atlas 配置与使用的实践内容。基于第3章的一些基本配置，读者参考这 一章的内容，即可轻松地在大数据平台上完成对 Atlas 的安装与配置。同时，按照本章的步 骤，也能够实现 Atlas 的各项重要功能。

为了方便读者最大限度地发挥Atlas 的能力，更好地实践大数据治理的内涵，本章的最 后在结合大数据行业的实践内容基础上，对一些常用场景进行了设计与配置，并对Atlas 的 性能优化进行了介绍。读者依照这些内容，可以直接将一些设计场景应用于生产实际，并 且可根据自身的实际情况对Atlas 的性能进行优化，提高大数据平台生成环境的运行效率。



夏■■■

■■

■

■ ■ ■ ■

■ ■

■ ■| ■

国■面|■■ ■■

■ ■

画i ■

■|■l ■ ■

■

国 ■1

第 5 章

**大数据安全之** **Apache** **Ranger**

随着 Hadoop生态系统越来越成熟，其现在已经能够支持完整的数据湖 (Data Lake, 是 指来自多个不同来源的结构化或者非结构化数据，有选择地被收集在多个数据库或者存储 系统上)架构。企业在 Hadoop系统上可以在多客户环境中运行多个工作负载。由于需要支 持多用户访问企业数据湖，而数据对于企业来说是重要的资产，为了保护这些各种不同类 型的用户的数据，专门应对大数据的分布式用户安全架构应运而生。

本章主要向读者介绍Ranger的基本结构、发展历程、特点，以及与Hadoop 生态系 统组件(如 HDFS 、Hive 、HBase 、Kafka等)的安全认证机制和配置，同时详细地介绍 Ranger的4种策略配置以及优化和性能分析。目的在于让读者对 Apache Ranger安全架构 的背景及主流组件的安全配置和使用有一个全面的了解。

**5.1** **Apache Ranger 概述**

Hadoop 生态系统中的组件就像一个零件包中的零件，每个零件都需要单独保护。直到 Apache Ranger诞生，才使得Hadoop的各个组件的安全性有了保障。Apache Ranger 是用 于Hadoop的集中式安全管理解决方案，使管理员能够为HDFS 和其他 Hadoop 平台组件创 建和实施安全策略，并且为Hadoop的各个零部件提供细粒度的安全权限机制。本节主要就 Ranger 的技术概况、发展历程及其特点和应用场景做出详细叙述。

**5.1.1** **Ranger** **技术概况**

Apache Ranger 是一个集中式安全管理框架，提供集中管理的安全策略和监控用户访 问，而且它可以对Hadoop生态系统上的组件如 Hive 、HBase 等进行细粒度的数据访问控

第5章 大数据安全之Apache Ranger **195**

制，并解决授权和审计问题。通过操作Ranger Web UI控制台，管理员可以轻松地通过配 置策略来控制用户访问权限。Ranger 还提供全面的日志审计，使用户不仅能够清晰地看到 请求信息和数据的权限状态，而且还能够通过基于Web UI管理控制台来配置访问权限。正 如你所看到任何一个企业级安全项目解决方案一样，Ranger 通过LDAP/AD(Lightweight Directory Access Protocol,轻量目录访问协议)支持同步用户和组的信息。

为了实现用户和组的信息同步， Ranger 集成了一个独立的守护进程模块，负责与 LDAP/AD同步，并将策略分发到集群中的节点。轻量级代理程序嵌入在需要数据保护的单 个 Hadoop组件(如 Hive 、HBase 等)中，并使用这些组件中内置的安全挂钩。此代理还作 为NameNode 的一部分运行，为HDFS 提供访问控制，并收集存储在审核日志中的请求详 细信息。策略实施在本地层次节点上执行，这意味着 Ranger 在运行时间上没有显著的性能 影响。

Ranger 通过在 Hive 1.3中使用标准的 SQL 授权，进一步与 Hadoop 进行深度集成，并 且允许在表上使用 SQL grant /revoke 功能。此外，Ranger 提供了一种模式，可以通过使用 Hadoop 文件级权限，验证 HDFS中的访问。Ranger 与基于 Web的管理控制台一起，支持 用于策略管理的 REST API。

Apache Ranger 所做的是为 Hadoop 的每个部分和公共认证存储库提供插件，并允许用 户在集中位置定义策略，实现权限控制。长期来说，Ranger 已经涵盖Hadoop 安全性方面 的以下目标：

1)集中安全管理中心，管理在核心用户界面或者使用REST APIs 过程中所有与安全相 关的任务。

2)使用Hadoop 组件/工具进行特定操作的细粒度授权 (Fine grained authorization), 通过中央管理工具进行管理。

3)通过 Hadoop 组件来标准化授权方法。

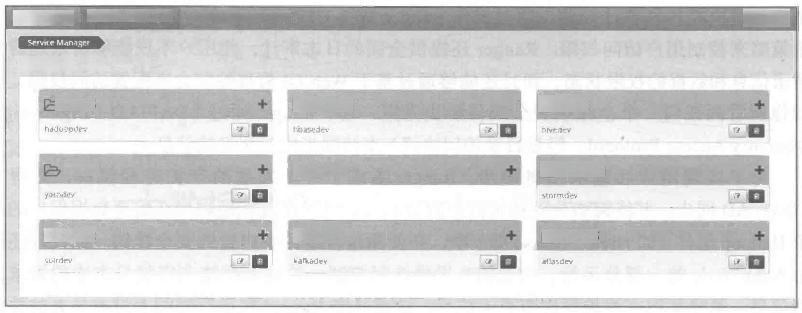
4)增强对不同授权方法的支持，如基于角色 (role) 的访问控制，或者基于属性 (attribute) 的访问控制。

5)集中 Hadoop 的所有组成部分的用户访问的审计 (auditing of user access) 和安全相 关管理操作。

而且Ranger 还可以与现有的IdM 和 SSO 解决方案(包括 Siteminder 、Tivoli Access Manager 、Oracle Access Management Suite 和基于 SAML 的解决方案)进行互操作。通过 内置Knox 集 成 ，Ranger 可用于保护任何提供对存储在 Hadoop 集群中的数据的周边访问的 REST端点。Ranger 甚至使用HiveServer 2来保护 ODBC/JDBC 连接，只要将 Thrift 网关 配置为对这些调用使用HTTP。

如图5-1所示为Ranger 0.6.2版本的 Web UI。截至写书为止，Ranger 支持9个 Hadoop 子组件，分别为HDFS(Hadoop Distributed File System) 、HBase 、Hive 、YARN 、Knox、 Storm 、Solr 、Kafka 、Atlas 。有关这些子组件将在后面几节详述。

196 **第二篇** **开源实现篇**



TNcess Manayer Audit Setings

Service Manager

3 HDFS **HBASE** + SHIVE

SSTORM

≥KAFKA

图5-1 Ranger 支持的组件

Ranger

-dmn

≥KNOX

**≥SOLR**

ATLAS

YARN

**5.1.2** **Ranger** **发展史及近况**

Ranger前身是由Hortonworks 开发的商用的 Hadoop安全管理系统——XA Secure 系统。 2014年5月，Hortonworks 开源了XA Secure系统，并将XA Secure系统的源代码捐赠给 Apache 软件基金会，其成为 Apache Ranger项目的基础。 Ranger (以前称作Apache Argus) 是Apache 孵化项目，成为Apache 家族的一个成员。

XA Secure 系统支持集中式的安全管理。策略更新机制和详细的日志审计功能，但只支 持对HDFS 、Hive 和HBase 进行细粒度权限管理控制。2014年11月， Ranger 第一个发行 版本 Ranger 0.4.0正式发布，其在XA Secure系统基础上添加了对Knox 、Storm 两个组件 的支持管理和审计，同时在 Apache Hive 和 Apache HBase上支持授权和撤销授权的功能。 Ranger 0.4.0将日志审计存储在HDFS 上，并且支持Oracle 数据库作为策略和审计的存储。

而到了2015年6月，发行的Ranger 0.5.0版本不仅支持High Availablity(Hadoop 的高 可用性模式),同时还支持通过KMS 对数据进行透明加密。在Ranger 0.5.0 中还支持更多的 插件，分别为Yarn 、Kafka 和 Solr 。该版本在上一个版本中添加了一个重要功能，就是通 过使用Solr 支持实时的审计和查询。Ranger 0.5.0 中优化和汇总了审计数据，而且支持将策 略数据存储到 PostGreSQL 和 MS-SQL 数据库中。

2016年8月发布的 Ranger 0.6.0版本，移除了基于数据库的审计支持，改用 Solr 作为 索引审计数据和用HDFS 来存储审计数据。这样做的目的在于使Ranger 插件在扩展插件时 也可以扩展审计日志和索引。在 Ranger 0.6.0版本中，改进了Ranger 管理中的 Web UI: 可 以提供资源访问、标记、策略类型等增强搜索，还可以将报表下载为Excel 或 csv 文件。而 且在 Ranger Web UI 中还提供了删除用户和组功能以及支持基于组的搜索。在 Ranger 0.6.0 版本中， Apache Ranger添加了对Atlas 组件的管理和审计，集成了Atlas 以支持基于分类 (标记)以及其他动态策略(基于位置、禁止、数据生命周期)。这是第一次将用于安全的 Apache Ranger 和用于数据治理的 Apache Atlas 集成，以授权客户定义和实现基于动态分类

第5章 大数据安全之Apache Ranger **197**

的安全策略。

Atlas-Ranger 集成代表了大数据治理和安全的范式转变。通过将 Atlas 与 Ranger 集 成，除了提供基于角色的安全性之外，企业现在还可以实施动态的、基于分类的安全规则。 Ranger 的集中式平台使数据管理员可以根据 Atlas 元数据标签或属性来定义安全策略，并 将此策略实时应用于数据资产的整个层次结构，包括数据库、表和列。

尽管Ranger 今天具有重要的功能，但是仍然存在一些关于如何适应更大的 Hadoop 安 全生态系统的问题。例如， 一些Ranger 目标与Apache Sentry(详细介绍参见第6章)的目 标重叠，而且目前对于项目如何同步其工作似乎很少有共识。此外，由于大多数Hadoop 子 组件被开发为单独的项目(通常在 Apache 基金会内),具有不同组的提交者和不同的PMC, 所以不清楚所有Hadoop 项目是否会主动选择使用Ranger。 由于Ranger 目前还是 Apache 基金会的一个孵化项目，所以还有很多地方可以改进。如果读者感兴趣，想对Ranger 进行 开源贡献，只需要去 Apache Ranger 官网上注册一个账号，然后提交所要贡献的代码即可。

**②** **素**

目前，所有的 Apache 项目都需要经过孵化器孵化，满足 Apache 的一系列质量要求 之后才可毕业。从孵化器里毕业的项目，要么独立成为顶级项目，要么成为其他顶 级项目的子项目。

**5.1.3** **Ranger** **的特点和作用**

Ranger 之所以能成为Hadoop 家族中的一员，主要因为其有着与其他大数据安全管理 平台不同的特点。主要特点如下。

**1.细粒度授权**

不同于其他Hadoop 的安全管理机制， Ranger 支持细粒度的管理，对Hadoop 组件进 行安全保护。Ranger 对不同组件有不同的权限控制，例如对HDFS 文件读写，对Hive 和 HBase表的增删改查，以及Kafka 消息发布和消费等不同组件的权限控制。从表5-1可以看 出，Ranger 对不同的组件，针对不同对象，拥有不同的授权选项，具体的各个授权对象(或 者说策略适用的资源规范，例如HDFS 文件/目录、Hive 数据库/表/列、HBase 表/列- 家 族、列等资源)和授权选项(应允许访问用户/组，访问类型和自定义条件)将在接下来的 各节中介绍。Ranger 不仅能对组件有不同的权限控制，而且还能够根据用户所在的地理位、 IP 地址和时间给出不同的权限，这使得 Ranger 用户可以有效地对不同地理位置和不同客户 授予不同的权限。

**表5-1** **Ranger** **对不同的组件权限选项**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **授权组件** | **授权对象** | **授权选项** | | | |
| HDFS | 分布式系统的资源路径 | 读、 | 写、 | 执行 |  |
| HBase | 表、列-家族、列 | 读、 | 写、 | 执行、 | admin |

**198** **第二篇** **开源实现篇**

(续)

|  |  |  |
| --- | --- | --- |
| **授权组件** | **授权对象** | **授权选项** |
| Hive | 数据库、表、家族 | select、update、create、drop、alter、index、lock |
| YARN | Queue | admin-queue、submit-app |
| Storm | Storm Topology | submit Topology、file Upload、file download、kill Topology |
| Kafka | Collection | query、update、others、Solr Admin |
| Solr | Topic | publish、consume、configure、describe、create、delete、Kafka admin |
| Atlas | term、type、operation、taxonomy | read、create、update、delete、all |

**2.集中化审计日志管理和策略管理**

除了细粒度的权限控制，Ranger 还提供了集中化的日志审计功能。所有的不同用户和 组对安装 Ranger 的插件的 Hadoop 组件进行操作，都会生成一次该操作的日志，并存储在 数据库中。Ranger 还会更新策略，每次更新策略之前的策略信息都能查看到，原因在于每 次更新策略会创建一个策略，该策略会替换原来的策略，而原来的策略将失效，但在策略 信息中保存了所有从策略创建开始的信息，修改后的信息将保存在数据库中。

**3.易于操作控制权限**

相对于其他安全组件，Ranger 对各个组件进行权限控制，只需要用户登录 Ranger Web UI, 即可对相应组件的服务设置相应的策略。Ranger 强大之处在于，用户可以随时根据 需求更改组件的权限，只需要在Web UI中修改对应组件的策略即可。例如用户在需要对 HDFS上某一文件权限中读写权限做出修改，只需要修改该文件的策略中的权限选项，保 存策略之后，该文件的读写权限将改变。Ranger 不仅支持在策略中设置条件允许 (Allow Conditions) 的权限控制，而且还允许用户在策略中设置条件拒绝 (Deny Conditions)、从允 许条件中排除 (Excludes from Allow Conditions) 和从拒绝条件中排除 (Excludes from Deny Conditions) 的权限控制。只不过前提是，要将条件拒绝的权限控制在安装配置Ranger 时配 置相关参数。有关条件拒绝、从允许条件中排除和从拒绝条件中排除的权限控制，以及如 何开启这些权限控制，将会在后几节进行详述。

**4.统一的操作平台**

统一的操作平台主要体现在两个方面：第一，Ranger 把所有的组件的服务和策略的创 建和更新都在 Ranger Web UI上完成，而且组件可直接配置，使用非常简单方便；第二， Ranger 安装各个组件的插件时，只需要修改相应的配置文件，就能使Ranger 的权限控制功 能生效。当然，读者也可以通过RESTAPI 来配置相应组件的策略信息。

**5.全面支持Hadoop 的组件**

目前， Ranger 已经可以支持Hadoop 大部分核心组件，诸如 Hive 、HBase 、YARN、 Storm等 Hadoop 生态系统中的组件。当前Ranger支持 HDFS、Hive、HBase、Hive、Kafka 和Atlas, 在无Kerberos 条件下，也能对相应组件进行权限控制；而 Storm 和 Solr 则需要在

第 5 章 大 数 据 安 全 之Apache Ranger 199

Hadoop Components Enterpnise

User

Ranger Admin

Kerberos条件下，才能进行权限控制。当然，其他组件也支持在有 Kerberos的条件下，对 组件进行权限控制。未来，Ranger 将会支持更多组件，继续维护 Hadoop 生态系统的安全。

**5.1.4** **Ranger 架构**

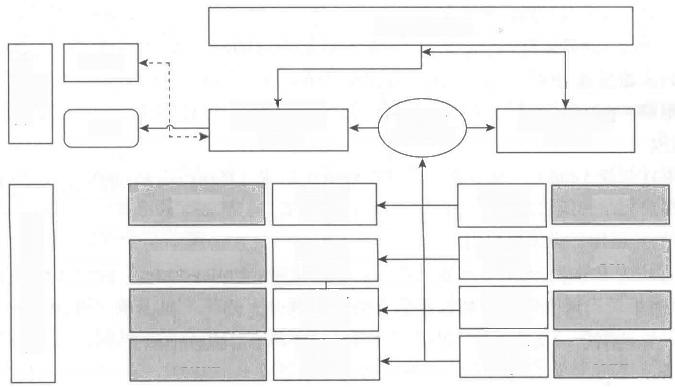
图5-2是 Ranger 安全认证机制的整体框架，主要包括 Ranger Admin 、Ranger Plugin 和 Ranger Usersync 三部分

**1.Ranger Admin**

Ranger Admin是安全管理的核心接口，也是Ranger 框架的管理中心。用户可以在它 提供的 Web UI上管理系统用户权限，创建和更新权限认证策略，然后将这些策略存储在数 据库中。每个组件的插件会定期监测这些策略。它还提供一个审计服务，可以收集存储在 HDFS 或者关系数据库中的数据并进行审计。

**2.Ranger Plugin**

Ranger Plugin 是权限安全管理的核心，它是一个轻量级的可以嵌入各个集群组件中的 Java 程序。例如，Ranger 对于其高度支持的Hive, 提供一个可以嵌入Hive Server 2服务中 的插件，这个插件能从Ranger Admin中提取到关于Hive 的所有权限认证策略，并将这些 策略存储在本地文件中。当用户请求通过Hive 组件时，这些插件会拦截请求，并安装认证 策略进行评估，确认其是否符合设置的安全策略。同时，这个插件还能从用户请求中收集 数据，并创建一个单独的线程，将数据传输到审计服务器中。需要注意的是，Ranger 对其 组 件 的 版 本 要 求 比 较 高 ， 如 Hadoop 必须是2 . 7 . 0以上版本，才能支持Ranger。



HDFS

Ranger Audi Server

|  |  |
| --- | --- |
| HBase  Hive Sever 2  HDFS  Storm | Ranger Plugin  Ranger Plugin  Ranger  Plugin  Ranger  Plugin |

|  |  |
| --- | --- |
| Ranger  Plugin  Ranger  Plugin  Ranger  Plugin  Ranger  Plugin | Atlas  Solr  Kafka  YARN |

Ranger Policy Server

Ranger Usersync

Solr

图5 - 2 Ranger 架构日

θ <https://zh.hortonworks.com/apache/ranger/>

**200** 第二篇 开源实现篇

**3.Ranger Usersync**

Ranger Usersync 是一个非常重要的工具，它用于将用户/组从UNIX 系统或 LDAP同步 到 Ranger Admin。此独立进程也可以用作 Ranger Admin的身份验证服务器，以使用Linux 用 户/密码登录到 Ranger Admin。用户或组信息存储在 Ranger Admin中，用于策略定义。而且 用户可以手动增加、删除、修改用户或用户组信息，以便对这些用户或用户组设置权限。

通过操作Ranger 控制台，管理员可以轻松地通过配置策略来控制用户访问HDFS 文件 夹、HDFS 文件、数据库、表、字段权限。这些策略可以为不同的用户和组设置，同时权限 可与 Hadoop 无缝对接。

**5.1.5** **Ranger** **应用场景**

Apache Ranger 提供一个集中式安全管理框架，并解决授权和审计问题。它可以对 Hadoop生态的组件如HDFS 、YARN 、Hive 、HBase 等进行细粒度的数据访问控制。通过 操作 Ranger 控制台，管理员可以轻松地通过配置策略来控制用户访问权限。Ranger 支持的 服务有HDFS 、HBase 、Hive 、YARN 、Strom 、Kafka 、Knox 、Solr。所以当用户需要对这 些框架进行权限管理的时候，或者整个大数据生态系统包含这些框架的时候，就可以使用 Ranger 来对这些框架进行权限控制。

同时，对于目前世界上大数据系统中心而言，都是以集群形式存在的，而Ranger 能对 各个节点上的不同用户进行权限控制，这对于整个系统而言非常方便，而且效果非常好。

通过Ranger 控制台Web界面，可以配置策略管理权限，其具体职能如下。

1)服务管理 (service manager): 目前支持的服务有HDFS 、HBase 、Hive 、YARN、 Strom 、Kafka 、Knox 、Solr。

2 ) 服 务 (service): 通过某 一个服务管理如HDFS, 创建 一个服务service, 如 hadoopdev, 通过这个服务， Ranger 控制台和具体环境的 HDFS 相关联。

3)策略 (policy): 有了某个服务后，如 hadoopdev, 可以用这个服务创建相关的策略 来管理权限。

4)审计管理 (audit): 对所做的操作进行审计，通过界面可以看到操作日志信息。

5)用户和用户组管理 (user and group): 可以增加、删除、修改用户或用户组信息，以 便对这些用户或用户组设置权限。

如上所述，Ranger 主要具有如下特性：基于策略 (Policy-based) 的访问权限模型；通 用的策略同步与决策逻辑，方便控制插件的扩展接入；内置常见系统(如HDFS 、YARN、 HBase) 的控制插件，且可扩展；内置基于LDAP 、文件的用户同步机制，且可扩展；统一 的管理界面，包括策略管理、审计查看、插件管理等。

所以，有审计以及权限认证需要、有策略支持的操作系统集群，非常适合使用 Ranger 来对系统进行管理。包括各种需要监控数据流，需要对数据进行加密，对不同用户有不同 权限设置的集群，甚至屏蔽特殊IP 、对不同节点有不同的限制的需求，都可以使用 Ranger

第5章 大数据安全之Apache Ranger 201

简单地实现。更重要的是，对于整个权限认证系统而言，Ranger 具有非常强的扩展性，不 但可以满足目前大部分生成环境中的权限认证需求，而且对特定需求也可以基于Ranger 进 行升级，满足这些特定需求。

而且对于特定软件而言，比如使用Ambari 搭建系统的话，那么 Ranger就几乎是一个 必须要使用的权限认证框架了。

Apache Ambari是一种基于Web 的工具，支持Apache Hadoop 集群的供应、管理和监 控。Ambari 目前已支持大多数Hadoop 组件，包括 HDFS、MapReduce、Hive、Pig、HBase、 Zookeper、Sqoop 和 Hcatalog 等。

Ambari与 Hadoop 等开源软件一样，也是 Apache Software Foundation 中的一个项目， 并且是顶级项目。目前最新的发布版本是2.0.1,未来不久将发布2.1版本。Ambari 的作用 就是创建、管理、监视 Hadoop 的集群，但是这里的 Hadoop 是广义的，指的是 Hadoop 整 个生态圈(例如 Hive 、HBase 、Sqoop 、Zookeeper 等),而并非特指 Hadoop。用一句话来说， Ambari 就是为了让 Hadoop 以及相关的大数据软件更容易使用的一个工具。

Ambari 目前主要取得了以下成绩：

1)通过一步一步地安装向导简化了集群供应。

2)预先配置好关键的运维指标 (metrics), 可以直接查看Hadoop Core(HDFS 和 MapReduce) 及相关项目(如 HBase 、Hive 和 HCatalog) 是否健康。

3)支持作业与任务执行的可视化与分析，能够更好地查看依赖和性能。

4)通过一个完整的RESTfulAPI 把监控信息暴露出来，集成了现有的运维工具。

5)用户界面非常直观，用户可以轻松有效地查看信息并控制集群。

Ambari使用Ganglia 收集度量指标，用Nagios 支持系统报警，当需要引起管理员的关注 (比如，出现节点停机或磁盘剩余空间不足等问题)时，系统将向其发送邮件。此外，Ambari 能够安装安全的(基于Kerberos)Hadoop 集群，以此实现了对Hadoop 安全的支持，提供 了基于角色的用户认证、授权和审计功能，并为用户管理集成了LDAP 和 Active Directory。

**5.2 Apache Ranger 的安全认证配置**

**5.2.1 Ranger 安装与部署**

Ranger 只能运行在 GNU/Linux平台上， Windows 与 MacOS 平台则不能作为 Ranger 的基 础环境。所以需要部署Ranger 作为大数据安全的集中式安全管理框架时，只能选择 GNU/ Linux操作系统，本章后续所有内容也都基于Linux 发行版 CentOS 6.5, 其他Linux发行版 的操作与之类似，仅仅是一些基础命令与之有差别。

Ranger使用Java 语言编写，运行在Java 虚拟机 (JVM) 上，因此需要在系统中安装 Java 6或者更高版本。考虑到后期的开发需求，建议读者直接从Oracle 官网上 (http://www. oracle.com/technetwork/java/javase/downloads/index.html) 下载合适的JDK 版本安装，并将

202 第二篇 开源实现篇

JAVA\_HOME环境变量的值设置为Java 的安装目录，PATH环境变量添加 Java安装目录下 的 bin 目录。安装完毕之后执行如下命令，检查JDK 是否已经被正确安装，命令返回结果 会因安装 JDK 版本而异。

$javac -version

javac 1.7.079

**1.安装** **MySQL**

安装 Ranger 前需要在本地安装好一个数据库，用于之后各类数据的存储，如果没 有数据库的支持， Ranger 将无法正确安装。Ranger 支 持MySQL 、Oracle 、MS SQL和 PostgreSQL RDBMS这4种数据库，建议选择MySQL。 本节选用了MySQL 作为Ranger 的数据仓库，后续内容也都基于MySQL, 如读者希望选用其他数据库，请参阅相应数据库 供应商网站的官方安装。本节提供安装MySQL 的快速指导，完整的细节请参阅 http://dev. mysql.com/doc/mysql-yum-repo-quick-guide/en。

首先需要确定Linux 的操作系统与版本号，对于CentOS 以及其他 Red Hat系统，可以 采用以下方式：

s cat /etc/issue

确认操作系统之后，访问<http://dev.mysql.com/downloads/> 并下载符合的 MySQL。本节 使用的是MySQL 5.7,可以在 <http://dev.mysql.com/downloads/repo/yum/> 中进行下载。本节 的操作系统版本为CentOS release 6.5, 下载符合标准的安装包到本地，或者执行以下命令， 直接下载安装包即可。

s wget <http://dev.mysql.com/get/mysq157-community-release-el6-8.noarch.rpm>

接下来，安装MySQL 。MySQL在新版本中加入了默认密码，安装时会将其存储在/var/ log/mysqld.log中，并标识在包含temporary password 的一行中，所以引号内的 “Spasswords” 为用 grep命令查询出来的临时密码。

s rpm -Uvh mysq157-community-release-el6-8.noarch.rpm

$service mysqld start

s grep 'temporary password'/var/log/mysqld.log

s mysql -uroot -p “Spasswords”

运行以上命令之后，即进入MySQL 控制台，可以对MySQL 进行各种操作。为了满足 Ranger 的需求，需要修改MySQL 密码并设置数据库的sql\_mode 为空。同时，在进行数据 库连接时，需要注意其中密码是前面的临时密码，修改密码为 “MyNewPass4!” 。 修改访问 权限，重启MySQL 。其 中“MyNewPass4!” 为新的数据库密码，读者可以根据自己的习惯 设置。而后修改MySQL的访问权限，设置成所有用户可以访问数据库。命令如下：

mysql>ALTER USER'root'@'localhost'IDENTIFIED BY'MyNewPass4!'; mysql>use mysql;

mysql>update user set host='&'where user='root'and host='localhost';

第5章 大数据安全之Apache Ranger 203

mysql>delete from user where host !='g';

mysql>set GLOBAL sql\_mode='';

mysql>flush privileges;

mysql>exit;

使用exit 命令退出 MySQL 命令台之后，需要对 MySQL 进行重启，并设置开机启动， 使以上设置生效，并保证 Ranger 启动前 MySQL已经启动。

s service mysqld restart

$chkconfig mysqld on

由于 Ranger 需要 MySQL支持，但是 Ranger 没有自带的数据库连接工具，所以需要下 载数据库连接的jar 包，使两者贯通。

$wget <http://mirror.bit.edu.cn/mysql/Downloads/Connector-J/mysql-connector-java-5.1.38>. tar.gz

$tar zxvf mysql-connector-java-5.1.38.tar.gz

s cp mysql-connector-java-5.1.38/mysql-connector-java-5.1.38-bin.jar /usr/share/java/

**2.下载并编译** **Apache Ranger**

Ranger 在本书编写之时还不提供可用的预先编译完成的二进制压缩包，甚至没有打包 好的源码包，所以必须从GitHub 上下载源码进行编译。在编译前需要检查一下版本，确定 将要编译的版本，因为各个版本的功能差别很大，建议使用最新的 release 版本。本书编译 的版本为0.6,编译时会自动选择最新的 release 版本。在本书编写的时候，最新的 release 版本为0.6.3,所以本书之后所有的安装以及功能使用都基于0.6.3版本。

由于网络原因，中国大部分地区克隆 GitHub 上的源码的速度会比较慢，而且 Ranger 代码量也比较大，所以需要等待一段时间。

$mkdir ~/dev

$cd ~/dev

s git clone <https://github.com/apache/incubator-ranger.git>

$cd incubator-ranger

s git checkout ranger-0.6

Ranger 只能使用 Maven 编译，所以在编译前确定 Maven 已经安装成功。Maven 的安 装细节不在此赘述。需要注意的是， Ranger 编译对内存要求较高，编译平台的内存一般要 大于2GB, 如果出现 “Java heap space” 错误提示，则需要加大内存。

$cd ~/dev/incubator-ranger

s export MAVEN\_OPTS="-Xmx512M"

S mvn -DskipTests=true clean compile package install assembly:assembly

编译过程缓慢，可能会出现依赖包下载错误，只需要多次执行最后一行代码即可。编 译完成后在 target 目录下会有ranger-admin 的压缩包以及各类插件的压缩包。

**3.安装** **Apache Solr**

安装 Ranger 之前需要先安装 Apache Solr 的 Standalone 模式，用以存储部分日志信息，

**204** 第二篇 开源实现篇

以及完成审计功能。作为此功能的 Solr,推荐使用Ranger自带的安装程序，因为实现此功 能的 Solr最好能够单独安装与运行，如果 Solr需要实现其他的功能，推荐安装与此分离的 Solr 集群，以保证功能不会有冲突。

s cd ~/dev/incubator-ranger/security-admin/contrib/solr\_for\_audit\_setup

s vim install.properties

#将文件相应的配置修改为如下内容

JAVA\_HOME=/var/local/jdk1.7.0\_79

SOLR\_USER=root

SOLR\_INSTALL=true

SOLR\_DOWNLOAD\_URL=<http://archive.apache.org/dist/lucene/solr/5.2.1/solr-5.2.1.tgz>

SOLR\_INSTALL\_FOLDER=/opt/solr

在进行上述配置时，需要注意，将JAVA\_HOME修改为本机的JDK地址； SOLR\_USER 一般指定为 root, 当然也可以修改，但最好具有超级管理员权限； SOLR\_INSTALL 必须设 置为 true, 以指示需要安装； SOLR\_DOWNLOAD\_URL需要配置下载地址，上述配置是官 网下载地址，可以设置为其他镜像地址； SOLR\_INSTALL\_FOLDER 表示安装路径，可根 据实际情况设置。

以上只是安装的必需配置项，配置文件中还有更多的配置项，如果需要的话，可以更 深入地了解。

使用Apache Ranger自带的安装程序进行 Standalone 安装，只需要在配置完成之后， 执行以下命令即可：

$./setup.sh

安装完成之后，可以查看文件/opt/solr/ranger\_audit\_server/install\_notes.txt, 文件中会 显示启动、关闭 Solr的命令。

#启动Solr 的命令

$/opt/solr/ranger\_audit\_server/scripts/start\_solr.sh

#注意启动Solr 时，报“solr.log 日志不能打开”错误时，应当给机器增加内存

#关闭Solr 的命令：

§/opt/solr/ranger\_audit\_server/scripts/stop\_solr.sh

**4.安装与启动** **Apache Ranger**

在前期准备完成之后，就可以安装了。安装分为两个部分， 一部分是 Ranger Admin服 务，提供Web UI和整个系统的基础框架，另一部分是 Ranger Usersync 服务，负责管理各 个系统用户的权限。首先安装 Ranger Admin。

$cd /usr/local

s tar zxvf ~/dev/incubator-ranger/target/ranger-0.6.2-SNAPSHOT-admin.tar.gz #设置ranger-admin 的连接

$1n -s ranger-0.6.2-SNAPSHOT-admin/ranger-admin

设置完成之后，需要对Ranger目录下 install.properties 文件的安装配置进行设置。首 先需要对数据库进行设置， Ranger 数据库主要设置项如下：

**第** **5** **章** 大数据安全之Apache Ranger 205

#数据库连接包

SQL\_CONNECTOR\_JAR=/usr/share/java/mysql-connector-java-5.1.38-bin.jar

#连接数据库配置

db\_root\_user=root

bb\_\_ryNewPass4!

#数据存储地址配置

db\_name=ranger

db\_user=rangeradmin

db\_password=MyNewPass4!

除了数据库之外，还需要对上述文件中的其他配置项进行修改，主要配置内容为系统 用户配置以及与 Ranger 关 联 的 Solr 配置。

#Solr配置

audit\_solr\_urls=http://localhost:6083/solr/ranger\_audits

#系统用户配置

unix\_user=root

unix\_group=root

上述配置中，需要注意，SQL\_CONNECTOR\_JAR 配置的地址是之前下载的数据连接包 存储的地址，这个地址可以按照读者的情况更改，如果读者更改了存放地址的话，需要修改 为更改之后的位置。数据库配置也是如此，如果之前配置有所更改，在上述配置项中同样需 要注意，尤其是密码，本书默认密码为 “MyNewPass4!”, 可根据自己的设置进行修改。而 数据存储地址配置，也可以按照自己的意愿进行修改，此部分主要用于创建Ranger 的数据 库存储地址以及连接数据库的用户。Solr 配置与之前 Standalone 安装配置相关，若不做修 改的话，按照如上配置即可。系统配置一般采用root 用户，因为需要有超级管理员权限。

 Ranger 服务的默认Web UI地址为http://localhost:6080, 不建议读者进行修改。如

果需要修改，请修改install.properties 中的policym gr external url配置项。

配置完成之后，可以进行下一步的安装。

$cd /usr/local/ranger-admin

9./setup.sh

#添加Ranger 用户

s adduser ranger

#启动Ranger Admin服务

$ranger-admin start

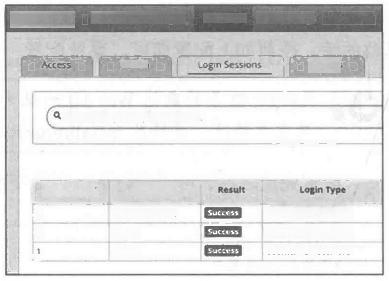
#停止Ranger Admin服务

$ranger-admin stop

在启动之后，可以验证Ranger Admin服务，在浏览器中输入http://ocalhost:;6080, 如果 出现 Ranger的登录界面，就表明安装成功。WebUI 默认的登录用户名/密码为 admin/admin。

Ranger Admin 服务启动之后，需要安装 Ranger Usersync 服务，以实现基本的功能。首 先需要配置一个链接，方便访问。

**206** **第二篇** **开源实现篇**



**Ranger**

Admin Phugn

Search for your login sesslons.

5esslonid- Legin d

3 rangerusersyne Username/Psssword

2 admin Username/Password

admin Username/Password

D Audit 6Setting

AcessManager

$cd /usr/local

$tar xzvf ~/dev/incubator-ranger/target/ranger-0.6.2-SNAPSHOT-usersync.tar.gz #设置Ranger Usersync的链接

s 1n -s ranger-0.6.2-SNAPSHOT-usersync/ranger-usersync

链接设置完成之后，可以修改 Ranger Usersync 服务安装配置项，首先需要在某个目录 下创建一个 Ranger Usersync服务日志目录，例如创建一个/var/log/ranger-usersync 目录， 然后对Ranger 地址中 ranger-usersync 目录下的 install.properties 配置文件进行修改。

#修改配置文件，将instal1.properties 文件中相应的地方修改如下：

aLnIgrY的\_RU\_IL =http://localhost:6080

SYNC\_SOURCE =unix

#系统用户配置

unix\_user=root

unix\_group=root

#日志存储地址

logdir =/var/log/ranger/usersync

上述配置中，除日志存储地址读者可以按照自己的习惯指定之外，其他配置建议按照这 里进行配置。此外，如果读者之前曾修改过Web UI 地址，同样需要修改配置项 POLICY

MGR\_URL。

配置确认无误即可进行安装。

$./setup.sh

#启动ranger-usersync 服务

s ranger-usersync start

#停止ranger-usersync 服务

s ranger-usersync stop

ranger-usersync 服务启动后，可以登录

Ranger Web UI查看 Audit 以及 Settings 中的内

容。Audit 的 Login Sessions中 出 现Login ld 为

rangerusersync 的记录，并在 Settings 中确定产生

了用户 rangerusersyne, 即确认 Ranger Usersync

服务安装完成，可以进行之后的安装。如图5-3

所示为 RangerAudit 中的登录记录。 图5-3 Ranger Audit中的登录记录

**5.2.2** **安全及访问权限控制机制**

从权限模型的角度剖析 Ranger 如何实现 centralized 访问控制，访问权限无非是定义了 “用户-资源-权限”这三者间的关系， Ranger 基于策略来抽象这种关系，进而延伸出自己 的权限模型。从“用户-资源-权限”的角度来详解上述表达的话，即有下面的定义。

1)用户：由User 或 Group 来 表 达 ，User 代表访问资源的用户，Group 代表用户所属 的用户组。

第5章 大数据安全之 Apache Ranger 207

2)资源：由(Service,Resource) 二元组来表达， 一条Policy 唯一对应一个 Service, 但可以对应多个 Resource。

3)权限：由(AllowACL,DenyACL) 二元组来表达，两者都包含两组 Accessltem。而 Accessltem 则描述一组用户与一组访问之间的关系——在AllowACL 中表示允许执行，而 在 DenyACL中表示拒绝执行。

表5-2列出了几种常见系统的模型实体枚举值。

**表5-2** **Ranger** **模型实体枚举值**

|  |  |  |
| --- | --- | --- |
| 服务类型 | **资源类型** | 权限类型 |
| HDFS | Path | Read,Write,Execute |
| YARN | Queue | Submit,Admin |
| HBase | Table,Column Family,Column | Read,Write,Create,Admin |
| Hive | Database,Table,Column | Select,Update,Create,Drop,Alter,Index,Lock |

Apache Ranger为了实现企业级Hadoop 生态系统的综合安全权限管理，提供了一个全 面的 Hadoop 集群的安全策略。它的定义提供了一个集中的平台，管理和授权所有安全策略 一致的 Hadoop生态系统的组件，所以Apache Ranger 设计了一个分散的结构。

Ranger 的分散结构使得它能在不需要使用某些组件的时候，可以不安装这个组件的插 件，只要安装需要进行权限控制的组件即可。这种方式大大减轻了系统的复杂度，尤其是 当需求只针对某一两个组件的时候， Ranger 能很轻量级地实现权限控制。同时Ranger 还提 供一些多功能插件，让用户在实现特定需求的时候使用。

Ranger 的安全认证机制还有很多细节设计，例如在5.3.2节中所述的 Deny Condition 以 及 Exclude策略，此策略启用之后， Ranger Admin 服务将产生4种配置项： Allow Condition、 Exclude from Allow Conditions 、Deny Condition 、Exclude from Deny Condition。这4种配 置主要被分为两个部分，Allow 与 Deny, 而 Condition 部分是整体控制，系统会识别其中配 置的策略，对请求进行认证，但是如果需要更为细化地对配置进行精简，也可以在 Exclude 中配置，将把整体配置中的某些权限排除在外，实现更细粒度的权限控制。

例如，可以在Allow Condition中对用户组 public 设置所有权限，但是在这个用户组 中 ，hdfs 用户不应该拥有read 权限，那么就在 Exclude from Allow Conditions 中将 hdfs 的 read 权限排除，使hdfs 用户不能拥有 public 用户组所拥有的 Allow Condition 配置；此外， 也可以在 Deny Condition中进行配置，拒绝 hdfs 发出的read 请求。这两者的最终效果是一 样的，但请求所经过的认证流程是不一样的。

Ranger 还提供各类认证策略，认证策略包括Geo-based 认证策略，基于Hive 的 Row- Level Filtering 和 Column-Masking 验证，以及 Tag同步认证等，这些策略能按照用户的需 求对系统权限控制进行调配，具有非常好的适应性。但有些认证策略与使用的框架有很大 的关系，所以需要用户根据具体需求使用。

Geo-based 认证策略是基于IP 地址的一个认证策略，可以针对集群中各个扮演不同角

**208** 第二篇 开源实现篇

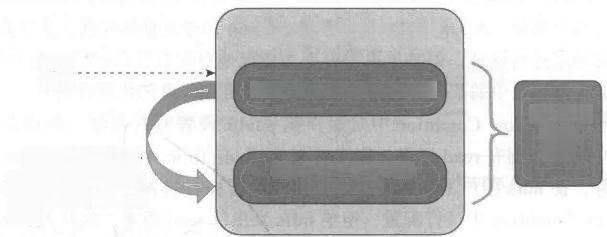
色的节点设置不同的权限，比如Master 可以设置全部权限，而存储节点只能拥有数据仓库 的读写权限等。这个策略具有非常好的适应性，而且支持所有组件，是常用的认证策略之 一。而基于Hive的 Row-Level Filtering和 Column-Masking验证，能对 Hive内的数据进行 细粒度的过滤，也能对数据进行各类加密。

**④** 素 Ranger 安装之后，对组件的控制权限还未生效，直到成功安装相应组件的 Ranger插 件之后，才能对组件进行细粒度权限控制。这时需要注意的是，当Ranger 对某个组 件权限控制生效时，若使用该组件时产生权限问题，那么说明当前操作的用户没有 权限对该组件进行操作，这就需要在 Ranger Web UI中对该用户进行权限控制授权。

**5.2.3 Ranger 集成** **HDFS 的安全认证机制与配置**

在上一节中介绍了如何安装部署Ranger, 并介绍了Ranger如何实现对插件进行安全权 限访问控制。接下来的几个小节分别介绍 Ranger 插件的配置以及如何对插件进行安全认证。

HDFS是所有Hadoop部署的核心部分，为了确保数据在Hadoop平台中得到保护， 安全性需要嵌入HDFS 层中。HDFS使用Kerberos身份验证以及使用POSIX(Portable Operating System Interface,便携式操作系统接口，是UNIX 系统的一个设计标准)样式 权限/HDFS ACL, 或使用Apache Ranger(Ranger ACL)的授权进行保护。图5-4展示了 Apache Ranger策略如何工作， Apache Ranger为HDFS提供联合授权模型，包括 HDFS ACL和 Ranger ACL(Access Control List, 访问控制列表)。首先HDFS的 Ranger插件检 查 Ranger策略，如果存在策略，则授予用户访问权限。如果 Ranger中不存在策略，那么 Ranger将默认使用HDFS(POSIX 或 HDFS ACL) 中的本地权限模型。此联合模型适用于 Ranger 的 HDFS和 YARN 服务。



Ranger NameNode Plugin

Ranger Policies for HDFS

POSIX Pemissions/ HDFS ACL

用户

请求

没有设置Ranger Policy, 那么权限就 调用HDFS本地权限 控制

请求授权 允许/

拒绝

图5-4 联合授权模型e

对于其他服务(如 Hive或 HBase),Ranger 作为唯一授权服务，这意味着只有Ranger

策略生效，才能控制相应的权限。

e <https://hortonworks.com/blog/best-practices-in-hdfs-authorization-with-apache-ranger/>

第5章 大数据安全之Apache Ranger **209**

**1.先决条件**

Ranger使用Java 语言编写，运行在Java 虚拟机 (JVM) 上，而且Hadoop 也需要在 Java环境中运行，因此需要在系统中安装Java7(Java 8可能不兼容)。建议读者直接从 Oracle 官网上 (<http://www.oracle.com/technetwork/java/javase/downloads/index.html>) 下载 合适的JDK 版本安装，并将JAVA\_HOME环境变量的值设置为Java 的安装目录，在PATH 环境变量添加 Java安装目录下的 bin目录。安装完毕之后执行如下命令，检查JDK 是否已 经被正确安装，命令返回结果会因安装的JDK 版本而异。

$javac -version

§javac 1.7.0\_79

由于 Ranger 0.6.2 版本支持的Hadoop 版本为2.7.2或者更高版本，需要安装Hadoop 2.7.2。Hadoop的安装路径为/var/local/hadoop/hadoop-2.7.2。可自行安装 Hadoop分布式开 发环境， 一个主节点linc-1,2 个从节点linc-2 、linc-3。如没有特殊说明，默认将 Hadoop 组件安装在 hdfs用户中，插件安装在 root 用户下的/usr/local 文件夹下。

注意修改 hdfs 配置文件目录下的 hdfs-site.xml 文件，开启dfs 安全配置，同时设置新 创建文件的umask 码为077,添加如下配置项(如已添加则不需要重复添加):

<property>

<name>dfs.permissions</name>

<value>true</value>

</property>

<property>

<name>fs.permissions.umask-mode</name>

<value>077</value>

</property>

注 在hdfs-site.xml中，添加配置项的目的在于限制用户在hdfs的Web UI中查看 hdfs

意

文件系统，文件只有创建它的用户才有权限查看，或者被Ranger授权后才能访问。

**2.Ranger-HDFS** **插件安装**

在linc-1 节点上，在root 用户下，解压在5.2.1节所编译生成的 Ranger 的 HDFS插件， 其目录在~/dev/incubator-ranger/target/ranger-0.6.2-SNAPSHOT-hdfs-plugin.tar.gz, 将插件 解压到/usr/local目录下。然后建立软链接(又称符号链接),其中第3行命令将解压后的文 件夹软链接到 ranger-hdfs-plugin。这里In 表示建立链接，加上“-s” 表示软链接，去掉“-s” 就表示硬链接，硬链接只能链接文件而不能链接文件夹。软链接文件类似于 Windows 的快 捷方式，它实际上是一个特殊的文件。在符号链接中，文件实际上是一个文本文件，其中 包含另一文件的位置信息。

s cd /usr/local

s tar zxvf ~/dev/incubator-ranger/target/ranger-0.6.2-SNAPSHOT-hdfs-plugin.tar.gz

s ln -s ranger-0.6.2-SNAPSHOT-hdfs-plugin/ ranger-hdfs-plugin

210

$cd $vim

**第二篇** **开源实现篇**

ranger-hdfs-plugin

install.properties

注 删除软链接时，只需要执行rm/usr/local/ranger-hdfs-plugi n, 不能在文件夹后面多加

意

一个“/”,否则将会删除软链接对应文件夹下的所有文件。

进入 ranger-hdfs-plugin 文件夹，修改 install.properties 文件，修改的内容如下，其余配 置项不变。

POLICY\_MGR\_URL=http://linc-1:6080

SQI\_CONNECTOR\_JAR=/usr/share/java/mysql-connector-java-5.1.38-bin.jar

REPOSITORY\_NAME=hadoopdev

XAAUDIT.SOLR.ENABLE=true

XAAUDIT.SOLR.URL=http://linc-1:6083/solr/ranger\_audits

CUSTOM\_USER=hdfs

CUSTOM\_GROUP=hadoop

参数说明： POLICY\_MGR\_URL 指定当前HDFS 服务权限控制的 URL, 其中可以将 linc-1 换为读者安装 Ranger admin 所指定的机器名(也可以为IP 地址),一般所有的插件都 为同一URL;SQL\_CONNECTO R\_JAR 是指向数据库连接Jar 包的位置，这里是用MySQL 作为数据库； XAAUDIT.SOLR.ENABLE 和 XAAUDIT.SOLR.URL 分别表示将 Solr 作为审 计日志和 Solr 的 URL 地址； CUSTOM\_USER 和CUSTOM\_GROUP分别表示安装当前插件 所指的用户和组的信息。

设置好Ranger-HDFS插件配置文件后，应该将 Hadoop 配置文件夹和 lib 文件夹作为软 链接，其目的在于在启动插件安装程序(./enable-hdfs-plugin.sh) 时，能找到 Hadoop 所在 目录，并在 conf 目录下添加插件所需要的 xml 配置文件。注意：如果没有设置软链接， 在启动./enable-hdfs-plugin.sh 时会产生错误，而且/enable-hdfs-plugin.sh 必须在 root 用户下 启动。

§1n -s /var/local/hadoop/hadoop-2.7.2/etc/hadoop/ /usr/local/hadoop/conf

§cp/usr/local/ranger-hdfs-plugin/lib/ranger-hdfs-plugin-impl/\*.jar /var/local/

hadoop/hadoop-2.7.2/share/hadoop /hdfs/lib/

$1n -s /var/local/hadoop/hadoop-2.7.2/share/hadoop/hdfs/lib//usr/local/hadoop/lib $./enable-hdfs-plugin.sh

在运行./enable-hdfs-plugin.sh 之后，会在 Hadoop 配置文件 hdfs-site.xml 中自动添加如

下参数：

<property>

<name>dfs.namenode.inode.attributes.provider.class</name>

<value>org.apache.ranger.authorization.hadoop.RangerHDFSAuthorizer</value> </property>

这个属性设置由Ranger 对 HDFS 进行授权管理。而且还在 Hadoop 的配置文件夹中生

成了几个XML 文件，其中有Ranger-hdfs-audit.xml 和 Ranger-hdfs-security.xml,Ranger-

第5章 大数据安全之 Apache Ranger 21 1

hdfs-audit.xml是对hdfs权限控制审计日志配置信息文件，而Ranger-hdfs-security.xm1 是一 些Ranger服务、策略的权限控制信息。下面给出部分Ranger-hdfs-security.xml 文件的信息 以及说明。

<property>

<name>ranger.plugin.hdfs.service.name</name>

<value>hadoopdev</value>

<description>Ranger-hdfs服务名称</description>

</property>

<property>

<name>ranger.plugin.hdfs.policy.source.impl</name>

<value>org.apache.ranger.admin.client.RangerAdminRESTClient</value>

<description> 从资源中获取策略的类名称</description>

</property>

<property>

<name>xasecure.add-hadoop-authorization</name>

<value>true</value>

<description>

如果Ranger 授权失败，启用/禁用默认Hadoop授权(根据资源上的rwxrwxrwx权限)

</description>

</property>

**3.Ranger 对** **HDFS 权限控制的测试**

首先在linc-1节点上，启动Hadoop 、Solr、Ranger Admin、Ranger UserSync 服务，具 体命令如下：

$ranger-admin start

$ranger-usersync start

$/opt/solr/ranger\_audit\_server/scripts/start\_solr.sh

8 hdfs namenode -format(只需要安装hdfs 插件后第一次时格式化namenode, 文件系统可以不用格式化) s start-dfs.sh

注意，前3条命令在 root用户下启动，后两条命令在 hdfs 用户下启动。如果 Solr 启动 时报错，可以尝试给 linc-1节点增大内存，或者关闭 Solr后，重新启动 Solr。

在 linc-1 主节点中，使用浏览器打开http://linc-1:6080 网 址 (Ranger Web UI), 用户名、 密码分别为 admin 、admin。在 Service Manager中新建一个hdfs服务，对其中的配置项做 如下修改：

Service Name =hadoopdev

Username=hdfs

#这里密码为用户hdfs 的密码

Password=123456

Namenode URL=hdfs://linc-1:9000

修改完后，保存，再次进入修改界面，单击Test Connection按钮，刚刚开始单击可能 不会成功，多尝试几次一般会出现Connected Successful, 而且可以在 Audit→Plugins 中看 到 Range-HDFS 插件安装成功的记录，如图5-5所示。

**212** **第二篇** **开源实现篇**

|  |  |  |  |
| --- | --- | --- | --- |
| **Service** **Name** | Plugin id | Plugin IP | Http Response Code Status |
| **hadoopdev** | hdfsozte-1-hadoopde | <192.168.118.161> | **200** Policles synced to plugin |

图5-5 Ranger HDFS 插件安装成功记录

打开在Service Manage下 的 hadoopdev 服务，单击Action 栏下的新建policy, 设置 policy 的参数如下：

Policy Name:testpath

Resource Path:test,\*(\*代表所有路径)

t blic

Permission:Read 、Write 、Execute

参数说明：指定 Resource Path一定要包含一个\*路径和一个有控制权限的路径。该路 径所选的用户和组允许有 Read 、Write 、Execute 权限中的一个或者多个，在这里是指定了 3个权限：读、写和可执行。

在 linc-1 节点上，新建一个测试用户test, 并赋予其 sudo 权利，使用 test 用户新建路径 testpath,test 用户将被允许新建该路径，因为该用户对该路径拥有Read、Write、Execute 权限， 如图5-6所示。

[teste /1$hdfs dfs -mkdir /test

16/12/8923:27:11 WARN util NativeCodeLoader:Unable to load nativ

hadoop Ubrary for your platform...using builtin-java classes w

here applicable

hadoop library for your platform...using builtin-java classes w

here applicable

[test@ /1s hafs dfs -is /

16/12/6923:27:16 WARN util NativeCodeloader:Unable to load nativ

Found 1 items

drwx-

test supergroup θ2016-12-0923:27 /test

图5-6 测试用户 test

在Ranger的 Web UI中，查看日志信息Audit→Access, 将会看到该路径已被允许创 建，如图5-7所示。从图中可以看到 Access Enforcer(权限执行者)为 ranger-acl。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Policy ID | Event Time ? | User | Service | Resource | Access Type | Result | Access Enforcer | Client IP |
| Name /Type | Name/Type |
| 16 | 12/10/201607:21:38 AM | test | hadoopdev | /test | WRITE | Allowec | ranger-ad | <192.168.118.161> |
| hdfs | path |

图 5 - 7 Web UI审计日志记录

而后通过Ranger Web UI改变Resource Path的权限，只给 test2 路径可读权限，修改 testpath 如下：

Policy Name:testpath

Resource Path:test2,\*(\*代表所有路径)

Select Group:public

Select User:test

Permission:Read

第5章 大数据安全之 Apache Ranger 213

Waster Node

Slave Node

如图5-8所示，可以看出test 用户没有创建test2 路径的权限，被拒绝了。只有 hdfs 用 户 (Hadoop 所有者)有创建 test2 路径权限。

[test@ /1s hdfs dfs -mkdir /test2

-62d/:l5ib4yWfoutltafoLinogadbe:iUltia-a c**l**t

here applicable

mkdir:Permission denied:user=test,access=WRITE,inode="/test2":

hdfs:supergroup:drwxr-xr-x

图5-8 拒绝创建路径日志信息

在 Ranger 的 Web UI中，查看审计日志信息 Audit>Acess, 将会看到该路径被拒绝修 改或者创建，如图5-9所示。从图中可以看到 Access Enforcer(权限执行者)为 hadoop-acl。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Policy ID | Event Time" | User | Service  Name/Type | Resource  Name/Type | Access Type | Result Access Enforcer | | Client IP |
|  | 12/10/201607:54:22 AM | test | hadoopdev | /test2 | WRJTE | Denies | hadoop-acl | <192.168.118.161> |
| hdfs | path |

图5-9 WebUI 审计日志

**5.2.4** **Ranger集** **成YARN** **的安全认证机制与配置**

**1.Apache** **Hadoop** **YARN** **介绍**

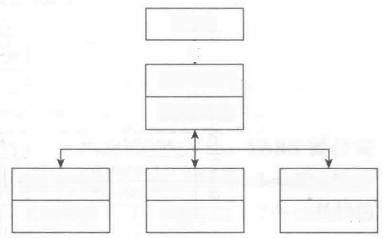
在介绍 Apache Hadoop YARN之前，首先来看看 Hadoop MapReduce1.0(MRv1) 的架构 和其所存在的缺陷。图5-10为 Hadoop MRv1的架构图，当一个客户端向一个 Hadoop 集群 发出一个请求时，此请求由JobTracker 管理。

JobTracker 与 NameNode 联合将工作分发到离 它所处理的数据尽可能近的位置。NameNode

是文件系统的主系统，提供元数据服务来执行

数据分发和复制。JobTracker 将 Map 和 Reduce

任务安排到一个或多个 TaskTracker 上的可用插 槽中。TaskTracker 与 DataNode (分布式文件系 统) 一起对来自 DataNode 的数据执行Map 和 Reduce 任务。当Map 和 Reduce 任务完成时，



Client

JobTracker

NameNode

TaskTracker

DataNode

TaskTracker

DataNode

TaskTracker

DataNode

1

图5-10 MRv1 架构图

TaskTracker 会告知 JobTracker, 后者确定所有任务何时完成并最终告知客户作业已完成。 MRv1 架构的局限性主要有如下几点：

1)JobTracker 单点瓶颈。JobTracker 不仅需要负责作业分发、管理和调度，同时还必 须和集群中所有的节点保持通信，了解机器的运行状态。如果集群的数量和提交的Job 数 量不断增加，那么JobTracker 的任务量将随之快速上涨，从而造成JobTracker 内存和网络 带宽的快速消耗。

2)TaskTracker 由于作业分配信息过于简单，有可能将多个资源消耗多或者运行时间长

214 第二篇 开源实现篇

Master Node

Slave Node

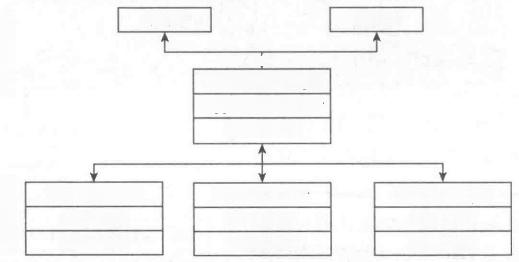
的 Task分配到同一个Node 节点上，这样会造成作业的单点失败或者等待时间过长。

3)作业延迟过高。在运行 MapReduce 之前，需要TaskTracker汇报自己的资源情况和 运行情况，JobTracker 根据获取的信息分配任务， TaskTracker 获取任务之后再运行。这样 就造成作业启动时间过长， 一个小的作业都要启动很长时间。

针对MRv1 架构的局限性， Hadoop 新的调度系统YARN 在一定程度上解决了以上的局 限。Apache Hadoop YARN(Yet Another Resource Negotiator,另一种资源协调者)是一种新 的 Hadoop 资源管理器，它是一个通用资源管理系统。YARN的基本思想是将资源管理和作 业调度/监视的功能分为两个独立的守护进程、 一个全局 ResourceManager(RM) 和若干个 应用程序ApplicationMaster(AM) 。 这里的应用程序是指传统的 MapReduce 作业或作业的 DAG(有向无环图)。

**2.YARN** **的主要架构**

图5-11是YARN的分层结构图，首先需要来自包含一个或者多个应用程序的客户的请 求，然后 ResourceManager 协商一个容器的必要资源，启动一个 ApplicationMaster 来表示 已提交的应用程序。通过使用一个资源请求协议， ApplicationMaster 协商每个节点上供应 用程序使用的资源容器。执行应用程序时， ApplicationMaster 监视容器直到完成。当应用 程序完成时， ApplicationMaster 从 ResourceManager 注销其容器，执行周期就完成了。下 面给出了各个 YARN的组件的功能。



Client

立

ResourceManager

ApplicationMaster

NameNode

NodeManager

Container

DataNode

NodeManager

Container

DataNode

NodeManager Container

DataNode

Client

图5-11 YARN 架构图

1)ResourceManager(RM): 一个全局的资源管理器，负责整个系统的资源管理和分配。 它主要由两个组件构成：调度器 (Scheduler) 和应用程序管理器(Applications Manager, ASM) 。调度器根据容量、队列等限制条件(如每个队列分配一定的资源，最多执行一定数 量的作业等),将系统中的资源分配给各个正在运行的应用程序。

2)ApplicationMaster(AM): 用户提交的每个应用程序均包含一个 AM, 主要功能包

括，与RM 调度器协商以获取资源(用Container 表示);将得到的任务进一步分配给内部的 任务(资源的二次分配);与 NodeManager 通信以启动或停止任务；监控所有任务运行状态， 并在任务运行失败时重新为任务申请资源以重启任务。应用程序管理器负责管理整个系统

第5章大数据安全之Apache Ranger 215

中所有应用程序，包括应用程序提交、与调度器协商资源以启动 ApplicationMaster、监控 ApplicationMaster 运行状态并在失败时重新启动它等。

3)NodeManager(NM):NM 是每个节点上的资源和任务管理器， 一方面，它会定时 向RM 汇报本节点上的资源使用情况和各个Container的运行状态；另一方面，它接收并处 理来自AM的 Container启动或停止等各种请求。

4)Container:Container 是 YARN中的资源抽象，它封装了某个节点上的多维度资 源，如内存、CPU、磁盘、网络等，当AM向 RM申请资源时， RM为AM返回的资源便用 Container 表示。YARN会为每个任务分配一个 Container, 且该任务只能使用该 Container 中 描述的资源。

**3.Ranger-YARN** **插件安装**

Hadoop-YARN的配置和安装已在前面章节讲述过，直接进入Ranger-YARN 插件的安 装部分。首先是解压插件，链接软链接，具体操作如下：

s cd $tar

s

s cd

/usr/local

zxvf ~/dev/incubator-ranger/target/ranger-0.6.2-SNAPSHOT-yarn-plugin.tar.gz

1n -s ranger-0.6.2-SNAPSHOT-yarn-plugin ranger-yarn-plugin

ranger-yarn-plugin

进入解压后的文件，修改 YARN插件的配置文件 install.properties 的参数REPOSITOR Y\_NAME为 yarndev, 即 YARN服务名称为 yarndev。其他参数配置和 Ranger-HDFS 插件 的配置一致。Hadoop 配置文件conf 和库文件 lib 软链接，已在 hdfs 插件的时候添加，所以 只需要拷贝 Ranger-YARN 的 jar 包到 lib 文件，操作命令如下：

$cp /usr/local/ranger-yarn-plugin/lib/\*.jar /var/local/hadoop/hadoop-2.7.2/share/ hadoop/yarn/lib/

s cp /usr/local/ranger-yarn-plugin/lib/ranger-yarn-plugin-impl/\*.jar /var/local/

hadoop/hadoop-2.7.2/share /hadoop/yarn/lib/

在root 用户下生效YARN插件，安装命令为./enable-yarn-plugin.sh。YARN 插件安 装后会在路径/usr/local/hadoop/conf (该路径为软链接路径，真实路径为/var/local/hadoop/ hadoop-2.7.2/etc/hadoop/) 生成 ranger-yarn-audit.xml 和ranger-yarn-security.xml 文件，具体的 文件说明已在hdfs插件部分说明。修改ranger-yarn-security 文件，在标签<configuration> 内添加配置项如下：

<property>

<name>ranger.add-yarn-authorization</name>

<value>false</value>

</property>

添加的目的在于使得YARN 只使用Ranger ACL控制权限，而忽略 YARNACL控制。

**4.Ranger-YARN 权限控制测试**

Ranger对YARN 支持对队列的提交作业和对队列的管理，通过设置 policy 对 YARN 的

216 第二篇 开源实现篇

队列进行权限控制。测试时，使用Hadoop 自带的MapReduce 例子 pi 来测试 YARN 的作业 提交和处理。

首先在linc-1 节点中启动 Ranger-admin 、Ranger-usersync 和 Solr 服务，具体命令见上 一节。由于要在linc-1 的 test 用户进行 MapReduce 作业，所以需要启动 YARN服务(具体 配置安装和启动命令见第3章)。

在 inc-1 主节点中，使用浏览器打开http://linc-1:6080 网址 (Ranger Web UI), 用户名 和密码分别为admin 、admin 。在 Service Manager 中新建一个 YARN服务，修改其中的配 置项为：

Service Name =yarndev

Username=hdfs

Password=123456

YARN REST URL=http://linc-1:8088

保存服务后，可以在 Audit→Plugins 下看到 yarn-plugin 插件安装成功的日志信息。

打开在 Service Manage 下的 yarndev 服务，单击 Action栏，修改当前 policy, 可以看 到默认policy 是对所有队列(\*表示所有队列),但只有hdfs 用户有submit-app 和admin- queue的权限，其他用户没有该权限。然后，在linc-1 上，使用hdfs 用户运行一个简单 MapReduce 作业，命令如下：

#启动Hadoop和Yarn

s start-dfs.sh;start-yarn.sh

s yarn jar /var/local/hadoop/hadoop-2.7.2/share/hadoop/mapreduce/hadoop-mapreduce- examples-2.7.2.jar pi 210

该命令是为一个求解π值的MapReduce 实例，执行部分日志信息如图5-12所示。可 以看出，此次作业的用户允许提交和执行，最后执行计算出的π值大约为3.80。注意在执 行过程需要test 对 hdfs 文件系统中/tmp 给予权限，所以最好在 hdfs 插件中新建 policy, 给 test用户对所有 hdfs 路径可以有Read 、Write 和Execute 权限。



16/12/1216:31:51 INF0 mepreduce.JobSubmitter:Subnltting tokens fo

r job:10b\_1481511269424\_8603

1l2:2 148151I9424\_00il.YarnClientImpl:Submitted application a

16/12/1216:31:52 INF⁰mapreduce.Job:The url to track the job:htt p://zte-1:8888/proxy/application\_1481511269424\_6883/

16/12/1216:31:52 INF0 mapreduce.Job:Running job:iob\_ 148151126942

46803

图5 - 12 YARN 作业日志

接着，从 Ranger的 WebUI中，查看日志信息Audit→Access, 将会有此次提交作业 成功的日志信息，如图5-13所示。从图中可以看到， Access Enforcer (权限执行者)为 ranger-acl。

然后在linc-1 上，使用test 用户执行计算π的命令，执行过程如图5-14所示。发现提 交作业失败，提示 AccessControlException, 说明 Ranger 对 YARN的权限控制起了作用。

第 5 章 大数据安全之Apache Ranger 217

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| User | Service  Name/Type | Resource  Name /Type | Access Type | Result | Access Enforcer |
| hdfs | yamdey  yarn | root.default queye | SUBMIT\_APP | Allowec | ranger-acl |

图5-13 日志审计



16/12/1217:42:57 INF0 mapreduce.JobSubmitter:Cleaning up the stag

ing area /tmp/hadoop-yarn/staging/test/.staging/job\_ 1481511269424\_6

004

java.io.I0Exception:org.apache.hadoop.yarn.exceptions.YarnExceptio

n:Falled to submit appiication\_ 1481511269424\_0864 to YARN:org.aP

ache.hadoop.security.AccessControlException:User test cannot submi

t applications to queue root.default

图5-14 YARN 提交任务失败日志信息

在 Ranger 的 Web UI中，查看 Audit→Access 。 如图5- 15所示，可以看到 YARN 提 交

任务失败。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Policy ID | Event Time" | User | Service  Name/Type | Resource  Name /Type | Access Type | Result | Access Enforcer |
|  | 12/13/201601:42:57 AM | test | yarndey  yarn | root.default queue | SUBMIT\_APP | Denied | ranger-ad |
|  | 12/13/201601:42:57 AM | test | yamde  yarn | root default queue | ADMINISTER\_QUEUE | Denied | ranger-acl |

图5-15 YARN日志审信息

可以看到 Ranger 策 略 的 权 限 设 置 生 效 ，Ranger 可 以 对 YARN 的权限进行控制。如 图5- 16所示，在YARN 资源管理UI(http://linc-1:8088/cluster/) 中同样展示了作业和队列 的信息的执行情况。其中第1个由test 用户执行的作业执行状态为FAILED, 第 2 个 由hdfs 用户执行的作业执行成功。

|  |  |  |  |
| --- | --- | --- | --- |
| anplication 14815112694240004 | test QuasifonteCarlo MAPREDUCE | defaul t Han Dec 12  17:42:57 +0800  2016 | Mon Dec 12 FAILED FATLED  17:42:57  +08002016 |
| apolication 1481511269424.0003 | hdfs QuasiMonteCarlo MAPREDUCE defaul t Mon Dec  12  16:31:51 +0800  2016 | | Mon Dec 12 FINISHED SUCCEEDED  16:32:19  +08002016 |

图5-16 YARN 作业状态信息

**5.2.5** **Ranger 集成** **Hive 的安全认证机制与配置**

**1.HiveServer2 简介与使用**

Hive 及其配置安装已在前面章节中介绍过，读者需要注意的是，Ranger 只支持 Hive 2.1.0

及以上版本，如果使用低于2.1.0版本的Hive, 将 Ranger-Hive 插件安装上后 HiveServer2 服

218 第二篇 开源实现篇

注  息

务将无法使用。

HiveServer2(HS2) 是一个服务器接口，使远程客户端能够执行Hive 的查询并检索结果。 HS2的实现基于 Thrift RPC,是 HiveServer 的改进版本，支持多客户端并发和身份验证。它旨 在为开放的API客户端(如JDBC和ODBC) 提供更好的支持。HS2 是作为组合服务运行的单 个进程，其包括基于Thrift的 Hive服务 (TCP或 HTTP) 和用于Web UI的 Jetty Web 服务器。

在 linc-1上，使用hdfs用 户 启 动HiveServer2服务，并用beeline 命令连接Hive。 HiveServer2 提供了一个新的命令行工具beeline, 它是基于SQLLine CLI的 JDBC 客户端。 首先，在后台启动 Metastore 服务和 HiveServer2服务，使用 beeline 命令连接数据库。

#后台启动Metastore 服务和HiveServer2 服务

$nohup hive --service metastorek

S nohup hive --service hiveserver26

#启动HiveServer2

s beeline -u jdbc:hive2://linc-1:10000/default -n hdfs -p 123456

#成功进入beeline 连接JDBC数据库

$0:jdbc:hive2://linc-1:10000/default>show $0:jdbc:hive2://linc-1:10000/default>show

$0:jdbc:hive2://linc-1:10000/default>!q

databases;

tables;

**注意，如果执行** **beeline 命令连接** **Hive数据库不能正常启动，报如下错误**：

Failed to open new session:java.lang.RuntimeException:org.apache.hadoop.ipc. RemoteException (org.apache.hadoop.security.authorize.AuthorizationException): User:hdfs is not allowed to impersonate anonymous(state=,code=0)

分析错误发现为Hadoop访问权限问题，需要在 Hadoop的配置文件core-site.xml 中加 入如下配置 (hadoop.proxyuser.hdfs.hosts 中的hdfs 表示 hdfs用户可以代理节点上的所有用 户，当然也包括 anonymous 用户):



<property>

</property>

<property>

<name>hadoop.proxyuser.hdfs.groups</name> <value>\*</value>

</property>

<name>hadoop.proxyuser.hdfs.hosts</name> <value>\*</value>

个

在启动HiveServer2服务过程中，查看nohup.out文件发现报 Unable to open a test connection to given database 错误，说明 MySQL数据库未启动或者启动异常，可以 用 service mysqld restart 命令重启 MySQL。

**2.Ranger-Hive** **插件安装**

1)解压Ranger-Hive 插件。

根据前几小节的Ranger插件安装配置，首先在linc-1节点上，使用hdfs用户解压插

第5章 大数据安全之Apache Ranger **219**

件，之后修改配置文件install.properties 。修改配置时只需要将REPOSITORY\_NAME 设为 hivedev, 其他配置项与 Ranger-HDFS 插件配置相一致。具体命令如下：

s cd /usr/local

s tar zxvf ~/dev/incubator-ranger/target/ranger-0.6.2-SNAPSHOT-hive-plugin.tar.gz

s ln -s ranger-0.6.2-SNAPSHOT-hive-plugin ranger-hive-plugin

s cd ranger-hive-plugin

s vim install.properties

2)建立 Hive的配置文件和 lib 的软链接。

建立Hive 的 conf 文件夹和lib 文件夹的软链接，并将相应Ranger-Hive 的 jar拷贝到 Hive 的 lib 文件夹下。命令如下：

$mkdir -p /usr/local/hive

$ln -s /var/local/hadoop/hive-2.1.0/conf//usr/local/hive/conf

$1n -s /var/local/hadoop/hive-2.1.0/lib//usr/local/hive/lib

$cp /usr/local/ranger-hive-plugin/lib/ranger-hive-plugin-impl/\*.jar /var/local/

hadoop/hive-2.1.0/lib/

3)使 Ranger-Hive 插件生效，并启动HiveServer2 和Metastore 元数据服务。

在linc-1 节点上，使用root 用户，使Ranger-Hive 插件生效，若不用root 用户，将不 能使插件生效。命令如下：

s cd /usr/local/ranger-hive-plugin/

§./enable-hive-plugin.sh

启动 Hive相关服务，使用hdfs用户，使用nohup命令在后台开启Metastore 和 HiveServer2 服务。命令如下：

$nohup hive --service metastorek

s nohup hive --service hiveserver26

**3.Ranger-Hive 权限控制测试**

1)在 Ranger Web UI 中创建 Hive 服务。

在创建Hive 服务前，需要在linc 集群上启动 Hadoop 和 Hive。在 linc-client 中，使用 浏览器打开http://linc-1:6080 网址，用户名和密码分别为admin 、admin 。登录 Ranger Web UI后，在 Service Manager 中新建一个 Hive 服务，修改其中的配置项如下：

Service Name =hivedev

Username=hdfs

Password=123456

jdbc.driverClassName=org.apache.hive.jdbc.HiveDriver

jdbc.url=jdbc:hive2://linc-1:10000

在节点linc-1 上，使用hdfs 用户在后台启动Metastore 和 HiveServer2 服务，接着重 新打开Hive 服务进入Hive 服务修改界面，单击Test Connection按钮，出现Connected Successful 提示，并且可以在Audit→Plugins 中看到hive-plugin 插件安装成功。

220 第二篇 开源实现篇

3素 如果没有在后台开启 Metastore 和 HiveServer2服务，单击 TestConnection 将发现报

Unable to connect to Hive Thrift Server instance 错误，因为 Thrift 服务没有打开。

2)测试Ranger-Hive 的 Allowed 权限。

在linc-client 中，测试刚刚新建 service 后产生policy 权限，即对所有用户授予所有 权限。

#beeline 中新建一个table, 名为test, 有一个字段id

$beeline -u jdbc:hive2://linc-1:10000/default -n 0:jdbc:hive2://linc-1:10000/default>create table

0:jdbc:hive2://linc-1:10000/default>!q

hdfs -p test(id

123456

int);

执行上述语句，会发现在Ranger Web UI下 Audit→Access,hdfs 如图5-17所示。

用户允许创建 test 表，

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| User | Service | **Resgurce** | **Access** **Type** | Result | **Access** **Enforcer** |
| Name/Type | **Name/Type** |
| hdfs | hivedev | default/test/id | SELECT | Allowed | ranger-acl |
| hive | ②column |

图5-17 Allowed 权限部分日志信息

在 Hive 中向test 表插入数据，并用beeline 查询，也能在 Ranger Web UI中查看到相应 的审计日志的信息。

$hive

hive>insert into test values(23);

hive >exit;

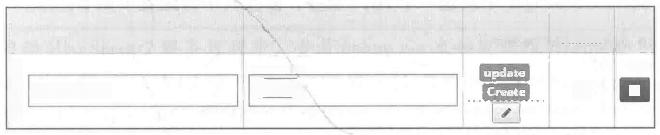
beeline -u jdbc:hive2://linc-1:10000/default -n hdfs -p 123456

0:jdbc:hive2://linc-1:10000/default>select \*from test;

0:jdbc:hive2://linc-1:10000/default>!q

3)测试 Ranger-Hive 的 Denied 权限。

修改之前的policy, 修改项如图5-18所示。其中只给予hdfs 对test 表的 update 和 create 权限，没有给予 select 权限。



Delegate Permissions

Admin

回 %

Select User

x hars

Select Group

Select Group



图5-18 Ranger-Hive策略配置信息

接着在 beeline 下查询表test 的内容，查询结果如图5-19所示。可以明显地从命令行中 看到， Hive 权限控制异常，hdfs 用户没有权限查询 default 数据库 test 表中的数据。

第 5 章 大 数 据 安 全 之 Apache Ranger 22 1



0:jdbc:hive2:// :16608/default>select t from test;

Error:Error while compiling statement:FAILED:HiveAccessControlEx

ception Permission denied:user [hdfs]does not have ISELECT]privi

lege on[default/test/id](state-42806,code-40800)

图5-19 Hive 权限控制异常

而且在审计日志中可以查看 Ranger 对 Hive 的这次操作的具体权限控制信息，其信息包 括刚刚在命令行中所出现的信息，唯独多了个Client IP, 即操作的用户IP, 如图5 - 20所示。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Policy ID | Event Time | User | Service | Resource | Access Type | Result | Access Enforcer | Client IP |
| Name/Type | Name/Type |
|  | 12/22/201606:1009 AM | hdfs | hivedev  hive | default/test/id  @column | SELECT | Deniee | ranger-ad | <192.168.118.161> |

图5-20 Ranger-Hive 权限日志审计信息

**5.2.6 Ranger 集成** **HBase 的安全认证机制与配置**

**1.HBase** **简介**

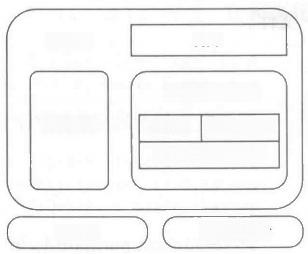
HBase是一个开放源代码、非关系型、分布式数据库，以Google的 BigTable 为模型，用 Java 编写。它作为 Apache Software Foundation 的 Apache Hadoop 项目的 一 部分开发，并在 HDFS(Hadoop 分布式文件系统)之上运行，为Hadoop 提供类似 BigTable 的功能。也就是 说，它提供了 一种容错方式来存储大量稀疏数据(在大量空或不重要数据集中捕获的小量信 息，例如在 一 组20亿条记录中查找50个最大项，或者找到非零项代表小于0 . 1%的大集合)。

HBase 在原始 BigTable 论文中概述了基于每列的压缩，内存操作和Bloom 过滤器。 HBase中的表可以作为在 Hadoop 中运行的MapReduce 作业的输入和输出，并且可以通 过JavaAPI 访问，也可以通过REST 、Avro或 Thrift 网关API 访问。HBase 是一个面向 列的键值数据存储，并且由于其能与Hadoop 和HDFS 系统很好地结合而被广泛使用。 HBase 在 HDFS上运行，非常适合于对具有高吞吐量和低输入/输出延迟的大型数据集进 行更快的读取和写入操作。HBase 在0.92版之后引入了协处理器 (coprocessors), 实现

一些激动人心的新特性：能够轻易建立二次索引、复 杂过滤器以及访问控制等 (coprocessors 的详细介绍 见<https://blogs.apache.org/HBase/entry/coprocessor>\_

introduction)。

如图5-21所示，HBase 在完全分布式环境下，由 Master进程负责管理Region Server集群的负载均衡以 及资源分配， Zookeeper 负责集群元数据的维护，并且 监控集群的状态以防止单点故障，每个Region Server 会 负 责 具 体 数 据 块 的 读 写 ，HBase 所 有 的 数 据 存 储 在 HDSF 系统上。



API

Region Server

HFile MemStore

Write-AheadLog

Zookeepei

Master

HBase

HDFS

图5-21 HBase 结构图

**222** 第二篇 开源实现篇

**2.Zookeeper 简介与配置**

由于需要安装配置分布式的 HBase, 其依赖于 Zookeeper 集群，所有的节点和客户端都 必须能够正常访问 Zookeeper。

Zookeeper 是一个分布式的应用程序协调服务，它可以为用户提供配置维护、域名服 务、分布式同步、组服务等功能。Zookeeper 的目标就是封装好复杂、易出错的关键服务， 将简单易用的接口和性能高效、功能稳定的系统提供给用户，因此它的文件系统使用了目 录树结构。众所周知，分布式环境下的程序为了达到协调一致的目的，通常具备简单性、 有序性等特点。下面将介绍如何配置和使用 Zookeeper。

(1)下载解压Zookeeper

首先，安装Zookeeper 需要Java 的支持，并且要求1.6以上的版本。此外对于集群的 安装，Zookeeper 至少需要3个节点，系统要求大多数机器处于可用的状态，如果想要容忍 m 台机器的故障，那么整个集群至少需要2m+1台机器。

从Apache 官方网站下载Zookeeper 的最新稳定版本，网站如下：

<http://www-eu.apache.org/dist/zookeeper/>

作为国内用户来说，选择最近的源文件服务器所在地，能够节省不少时间，比如以下站点：

<http://mirror.bit.edu.cn/apache/zookeeper/>

这里在linc-1 节点上，使用hdfs 用户，采用国内的服务器进行下载。

s cd ~

9 wget <http://mirror.bit.edu.cn/apache/zookeeper/zookeeper-3.4.8/zookeeper-3.4.8.tar.gz>

安装文件到本地目录中，执行如下命令进行解压缩，而后将压缩文件移至插件安装目 录/var/local/hadoop 中。读者可以根据自己实际情况，对安装目录进行更改。

$tar zxvf zookeeper-3.4.8.tar.gz

$mv zookeeper-3.4.8//var/local/hadoop/

(2)在集群下配置zoo.cfg 文件

为了运行 Zookeeper, 需要在 zookeeper-3.4.8 目录的 conf 文件下，修改其默认的配置 文件。

s cd /var/local/hadoop/zookeeper-3.4.8/conf

$cp zoo\_sample.cfg zoo.cfg

需要在文件的最后添加如下内容：

server.1=linc-1:2888:3888

server.2=linc-2:2888:3888

server.3=linc-3:2888:3888

serverid=host;port:port 标识了不同的 Zookeeper 服务器的配置，在这个配置中，第1 个端口是跟随者连接到领导者 (leader) 的端口，第2个是用来选举 leader 的端口。

接着修改 Zookeeper 的日志存储路径，修改的内容如下：

第5章 大数据安全之 Apache Ranger 22 3

dataDir=/var/local/hadoop/zookeeper-3.4.8/dataDir

dataLogDir=/var/local/hadoop/zookeeper-3.4.8/logs

口 dataDir: 存储内存中数据库的快照，如果不设置参数，快照将被存储到默认位置。

口 dataLogDir: 存储事务的日志即顺序日志，如果不设置参数，更新事务的日志将被

存储到 dataDir 所指定的目录中。

在 Zookeeper 中创建相应的文件夹：

$mkdir $mkdir

-p

-p

/var/local/hadoop/zookeeper-3.4.8/dataDir

/var/local/hadoop/zookeeper-3.4.8/logs

文件 zoo.cfg 中其余默认配置项说明如下。

口 tickTime: 基本事件单元，以毫秒为单位。其他需要用到时间的地方都会以多少 tick 来表示。

口 clientport: 监听客户端连接的端口，接受客户端的访问请求。

口 initLimit: 设定了允许所有跟随者与领导者进行连接并同步的时间，如果在设定的 时间段内，半数以上的跟随者未能完成同步，领导者便会宣布放弃领导地位，进行 一次领导者选举。

口 synLimit: 设定了允许一个跟随者与一个领导者进行同步的时间，如果在设定的时 间段内，若跟随者未完成同步，它将会被集群丢弃。所有关联到这个跟随者的客户 端将连接到另外一个跟随者。

(3)将配置好的 Zookeeper 发送到其他节点

将配置好的 Zookeeper 同步到其他节点。

$scp -r /var/local/hadoop/zookeeper-3.4.8 linc-2:/var/local/hadoop

$scp -r /var/local/hadoop/zookeeper-3.4.8 linc-3:/var/local/hadoop

(4)配置 myid

在集群 dataDir 指定的目录里，手动新建一个 myid 文件，文件内容为一个数字，用来 唯一标识这个服务。 一定要保证这个id 在集群中的唯一性。Zookeeper 会根据这个 id 来取 出 serverx 上的配置。比如当前id 为1,则对应着zoo.cfg 里的 server.1 的配置。比如在这 个集群中linc-1 的 myid 为 1 ,linc-2 的 myid 为 2 , linc-3 的 myid 为3。

(5)配置Zookeeper 环境变量

为了以后操作方便，需要对各个节点(linc-1、linc-2、linc-3)中 的Zookeeper环境变 量进行配置，方法如下。在/etc/profile/ 文件中加入如下内容：

export ZO0KEEPER\_HOME=/var/local/hadoop/zookeeper-3.4.8

export PATH=SPATH:SZOOKEEPER\_HOME/bin

运行如下命令使其环境变量生效：

$source /etc/profile

224 第二篇 开源实现篇

(6)在集群中运行 Zookeeper

在3个节点中，使用hdfs 用户分别执行如下命令启动服务：

zkServer.sh start

集群如果成功启动，在启动目录下的 zookeeper.out文件中将看到如图5-22所示信息。



**ZooKeeper JMX enabled by default**

**Using config:/var/local/zookeepe**r-3.4.8/bin/../conf/zoo.cfg

**Starting zookeeper …STARTED**

图5-22 集群启动成功

**3.HBase** **安装与使用**

(1)下载 HBase 源文件

首先， HBase 需要 Zookeeper 的支持，必须保证 Zookeeper 可以成功开启。本书所使用 的 HBase 版本为1.1.7版本，读者可以根据需求使用HBase 的更高版本。可以从 Apache 官 方网站选择下载源文件的服务器，网站如下：

<http://www.apache.org/dyn/closer.cgi/hbase>

作为国内用户来说，选择最近的源文件服务器所在地，能够节省不少时间，比如以下站点： <http://mirror.bit.edu.cn/apache/hbase>

这里采用国内的服务器进行下载，在 linc-1 节点，使用 hdfs 用户执行以下命令：

s cd ~

$wget <http://mirror.bit.edu.cn/apache/hbase/1.1.7/hbase-1.1.7-bin.tar.gz> (2)解压源文件并修改 HBase-env.sh 文件

在linc-1 节点上，使用hdfs 用户解压源文件，将其解压到/var/local/hadoop 路径下。

s tar zxvf hbase-1.1.7-bin.tar.gz

$mv hbase-1.1.7 /var/local/hadoop/

在启动 HBase之前需要设置Java 环境变量、Hadoop 所在路径等参数、修改 HBase-env.sh 文件，将部分参数做如下修改：

#设置Zookeeper 随着HBase 启动

#如果设置成false, 启动HBase 前需要先启动Zookeeper

export HBASE\_MANAGES\_ZK=true

#设置HBase 日志路径

export HBASE\_LOG\_DIR=/var/local/hadoop/hbase-1.1.7/1ogs

#设置Java 环境变量

export JAVA\_HOME=/var/local/jdk1.7.0\_79

#设置Hadoop 安装路径

export HADOOP\_HOME=/var/local/hadoop/hadoop-2.7.2

#设置HBase路径

export HBASE\_HOME=/var/local/hadoop/hbase-1.1.7

**(3)修改** **regionservers 文件：**

$cd /var/local/hadoop/hbase-1.1.7/conf

第5章 大数据安全之Apache Ranger 225

s vim regionservers

-

linc-3

(4)修改 HBase-site.xml 文件

在文件/var/local/hadoop/hbase-1.1.7/conf/hbase-site.xml 中添加HBase 在分布式系统中 的 root 路径，并选择 HBase的集群为分布式的，而且指定 Zookeeper 节点信息，这里为多 个节点，用逗号隔开。具体信息如下：

<property>

<name>HBase.rootdir</name>

<value>hdfs://linc-1:9000/hbase</value>

</property>

<property>

<name>HBase.cluster.distributed</name>

<value>true</value>

</property>

<property>

<name>HBase.zookeeper.quorum</name>

<value>linc-1,linc-2,linc-3</value>

</property>

<property>

<name>HBase.zookeeper.property.dataDir</name>

<value>/var/local/hadoop/zookeeper-3.4.8/dataDir</value>

</property>

同时在该xml 文件中，设置HMaster 和 HRegion服务 Web UI的地址名。注意 HRegion 的地址名需要在linc-2 和 linc-3机器上改为相应的机器名。

<!--HMaster....-->

<property>

<name>HBase.master.info.bindAddress</name>

<value>linc-1</value>

<description>HBase Master Web UI<0.0.0.0></description>

</property>

<!--HRegionServer....-->

<property>

<name>HBase.regionserver.info.bindAddress</name>

<!--下面这项需要在linc-2 机器和linc-3 机器上改为linc-2 和linc-3 -->

<value>linc-1</value>

<description>The address for the HBase RegionServer web UI</description> </property>

其中有些使用默认值的配置项没有添加，所有配置项的信息请参考Apache 网站http:// hbase.apache.org/book.html#config.files。

(5)将HBase 发送到其他节点上

s scp -r /var/local/hadoop/HBase-1.1.7/hdfselinc-2:/var/local/hadoop

**226** **第二篇** **开源实现篇**

s scp -r /var/local/hadoop/HBase-1.1.7/hdfselinc-3:/var/local/hadoop

#在linc-2 机器上，将/var/local/hadoop/HBase-1.1.7/conf/HBase-site.xm1 中倒数第4行的

linc-1改为linc-[2](#bookmark1)

#在linc-3 机器上，将/var/local/hadoop/HBase-1.1.7/conf/HBase-site.xml 中倒数第4行的

linc-1改为linc-[3](#bookmark136)

( 6 ) 启 动 HBase

在linc-1节点上，使用hdfs用户启动 HBase, 并打开 HBase Shell 命令行。

$start-HBase.sh

$HBase shell

#创建一个test表，并查看

hbase(main):001:0>create 'test','cfl'

hbase(main):002:0>scan 'test'

在创建表的过程中，如果报Master is initialing的错误，查看 linc-2机器下 HBase/logs 目录的日志 HBase-hdfs-regionserver-linc-2.log, 发现 “Time difference of 97385ms >max allowed of 30000ms” 时间不同步，导致 regionserver开启后马上关闭了。这时只需要同步 linc-1 、linc-2 、linc-3 的时间即可。同步时间命令如下(也可使用其他同步集群时间的方法):

#需要在root 用户下，将所有节点同步

s ntpdate us.pool.ntp.org

表5-3列出了 HBase Shell 的一些常用操作命令。

**表5-3** **HBase** **Shell** **常用操作命令**

|  |  |  |
| --- | --- | --- |
| **命令名称** | **名** **称** | **命令表达式** |
| create | 创建表 | create'表名称','列名称1',列名称2','列名称N' |
| put | 添加记录 | put'表名称','行名称','列名称：','值' |
| get | 查看记录 | get'表名称'行名称 |
| count | 查看表中的记录总数 | count'表名称' |
| delete | 删除记录 | delete'表名','行名称','列名称' |
| disable和drop | 删除一张表 | 先屏蔽该表，才能对该表进行删除，即先disable'表名称', 然后drop'表名称' |
| scan | 查看所有记录 | scan"表名称" |
| 查看某个表某个列中所有数据 | scan”表名称”,[列名称：] |

**4.Ranger-HBase 安装配置**

Ranger-HBase 解压和建立软链接部分和前几节类似，这里不重复叙述。修改解压后的 HBase 插件的配置文件，只需要将 REPOSITORY\_NAME 修改为HBasedev 即可，其他选项 和 HDFS 插件一样。

接下来将 Ranger-HBase 插件发送到各个节点：

$scp -r /usr/local/ranger-HBase-plugin linc-2:/usr/local/ranger-HBase-plugin

$scp -r /usr/local/ranger-HBase-plugin linc-3:/usr/local/ranger-HBase-plugin

第5章 大数据安全之 Apache Ranger 227

读者可能注意到，之前的 Ranger 组件只需要将插件安装到主节点上，而不需要安装到 其他节点上。Ranger-HBase 插件必须作为coprocessors 部署在 Master 和 Regionservers 上 ， 所以必须在所有节点上都安装 Ranger-HBase 插件。

在 linc-1 、linc-2 、linc-3 上，使用hdfs 用户分别执行如下命令：

S mkdir -p /usr/local/HBase

#建立软链接

s ln -s /var/local/hadoop/HBase-1.1.7/conf $ln -s /var/local/hadoop/HBase-1.1.7/lib $cd /usr/local/ranger-HBase-plugin/

$./enable-HBase-plugin,sh

Yusr/local/HBase/conf

/usr/local/HBase/lib

注意：如果没有设置软链接，在启动./enable-hdfs-plugin.sh 时会产生错误，而且./enable- hdfs-plugin.sh 必须在 root 用户下启动。

**5.Ranger-HBase 权限控制测试**

在linc-1 中，使用浏览器打开http://linc-1:6080 网址，用户名和密码分别为admin、 admin 。在 Service Manager 中新建一个 HBase 服务，修改其中的配置项如下：

Service Name =HBasedev

Username=hdfs

rr.property.clientPort=2181

HBase.zookeeper.quorum=linc-1,linc-2,linc-3

zookeeper.znode.parent=/HBase

保存后，可以在 Audit→Plugins 中看到 HBase 插件安装成功。若未显示插件安装成 功，有可能是未启动 Hadoop 文件系统和 HBase, 只需要启动即可。

在 WebUI 中的 policy 中单击 Add New Policy按钮，新建 policy 。建立的 policy 值如下：

Policy Name: testpolicy

HBase Table: test

HBase Column-family: \*

HBase Column: \*

Allow Conditions:

Select Group: public

Select User: test

Permissions: Read Write Create

从设置的policy 可以看出，设置test 用户对表 test 具有读、写和创建的权限。下面在 linc-1 节点上使用test 用户，由于在安装 HBase 时已经创建了 test 表，这里就直接添加记录。

$hbase shell

hbase(main):001:0>put 'test','rowl','cfl:a','valuel'

运行后，发现添加记录成功并可在Ranger Web UI下 看 到 ，Ranger 对该次操作的审计 日志信息如图5-23所示。

228 **第二篇** **开源实现篇**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| User | Service | Resource | Access Type | **Result** | Access Enforcer | Client IP |
| Name/Type | Name/Type |
| test | hbasedev | test/cf1/a | put | Allowed | ranger-acl | <192.168.237.101> |
| hbase | column |

图5-23 添加记录成功

接着，修改之前创建的 testpolicy 策略，只给 test 用户只读权限。继续添加第2条记录 到 test 表中，命令如下：

hbase(main):001:0>put 'test','row2','cf1:a','value2'

ERROR:Failed 1 action:org.apache.hadoop.HBase.security.AccessDeniedException: Insufficient permissions for user 'test',action:put,tableName:test,

family:cf1,column:a

发现添加记录失败，权限不允许。同时也能在 Ranger Web UI 中查看到该条拒绝信息，

如图5-24所示。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| User | Service | Resource | Access Type | Result | Access Enforcer |
| Name/Type | Name/Type |
| test | hbasedev | test/cf1/a | put | Deniec | ranger-acl |
| hbase | column |

图5-24 添加记录失败

**5.2.7** **Ranger 集** **成** **Kafka 的** **安** **全** **认** **证** **机** **制** **与** **配** **置**

**1.Kafka 简介**

Kafka 是一个高吞吐量的分布式发布订阅消息系统，以为处理实时数据提供一个统一、 高通量、低等待的平台为设计目标。它最初由LinkedIn 公司开发，之后成为Apache 项目 的一部分。主要用于处理活跃的流式数据，实现了生产者和消费者的无缝连接，它以可水 平扩展和高吞吐率的特点而被广泛使用。

**2.Kafka 设计原理**

在介绍 Kafka 之前，先来学习 一 下 Kafka 中使用到的专业术语，借助这些术语可以更

详细地理解这个组件。表5-4中详细解释了Kafka 中的专业术语。

**表5-4** **kafka 专业术语介绍**

|  |  |
| --- | --- |
| **专业术语** | **解** **释** |
| Broker | Kafka集群包含一个或多个服务器，这种服务器被称为Broker |
| Topic | 每条发布到Kafka集群的消息都有一个类别，这个类别被称为Topic(物理上不同Topic的 消息分开存储，逻辑上一个Topic的消息虽然保存于一个或多个Broker上，但用户只需指定 消息的Topic即可生产或消费数据而不必关心数据存于何处) |

第5章 大数据安全之 Apache Ranger **229**

(续)

|  |  |
| --- | --- |
| **专业术语** | **解** **释** |
| Producer | 负责发布消息到Kafka Broker |
| Consumer | 消息消费者，向Kafka Broker读取消息的客户端 |
| Consumer Group | 每个Consumer属于一个特定的Consumer Group(可为每个Consumer指定Group Name, 若不指定Group Name则属于默认的Group) |

Kafka 与其他发布订阅消息系统相比，有以下几个特点。

1)高吞吐量、低延迟：同时为发布和订阅提供高吞吐量，Kafka 每秒可以处理55万条 消息，它的延迟最低只有几毫秒。

2)可扩展性：所有的Producer 、Broker 和Consumer 都可以有多个，均为分布式的。 Kafka集群支持热扩展，当需要增加 Broker节点时，新增的 Broker会 向Zookeeper 注册，而 Producer 及 Consumer会根据注册在 Zookeeper上的 Watcher 感知这些变化，并及时做出调整。

3)持久性、可靠性：消息被持久化到本地磁盘，充分利用磁盘的顺序读写性能，并且 支持数据备份，防止数据丢失。

4)容错性：允许集群中节点失败(若副本数量为n, 则允许n-1 个节点失败),消息被 处理的状态由Consumer 端维护，而不是由Server 端维护，失败时能自动平衡。

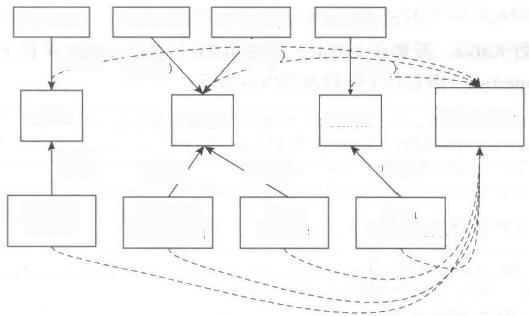
5)高并发：支持数千个客户端同时读写。

图5-25介绍了Kafka 整体架构。它是显式分布式架构，其中 Producer 和Consumer 都 可以有多个，并且以推-拉 (push-pull) 的模式进行， Producer 将消息推到 Kafka Broker 的 Topic 中 ，Kafka Broker用来存储消息， Consumer 可以订阅一个或者多个话题来拉取话题 中的消息。其中 Kafka Brokers 用 Zookeeper 来获取消息状态，并且通过Zookeeper 来存储 集群的元数据信息。对于每一个主题，Kafka 集群保留一个用于缩放、并行化和容错性的分 区。每个分区是一个有序的、不可变的消息序列，并不断追加并提交到日志文件。

(Producer)

(Broker)

(Consumer)



Service

(push).

(push)

|  |  |
| --- | --- |
| Kafka  (pull)/  Real-time  Monitoring | *(pull)*  Other Service |

Front End

(push

Front End

push

(pull)

Data

Hadoop Cluster

Zookeeper

warehouse

Front End

(pull)

Kafka

Kafka

图5-25 Kafka架构θ

日 来自 Apache Kafka 官网。

230 **第二篇** **开源实现篇**

**3.Kafka 安装配置**

(1)先决条件

首先， Kafka 需 要Zookeeper 的支持，必须保证Zookeeper 可以成功开启。为了保证 Kafka 功能的稳定， Kafka 需在其用户下进行下载安装。如果节点中没有Kafka 用户，首先 必须新建Kafka 用户。

# 新 建Kafka 用户并且设定Kafka 用户组为Hadoop

$useradd kafka -g hadoop

# 设 定Kafka 用户的密码

$passwd kafka

编辑文件/etc/sudoers,使 Kafka 用户拥有超级用户权限，然后以下步骤均由Kafka 用 户进行操作。

#在root $kafka

ALL=(ALL)

ALL=(ALL)

ALL 下一行添加内容

ALL

(2)下载 Kafka

从 Apache 官方网站下载 Kafka 的最新稳定版本，网站如下：

<http://kafka.apache.org/downloads>

作为国内用户来说，选择最近的源文件服务器所在地，能够节省不少时间，比如以下站点： <http://mirror.bit.edu.cn/apache/kafka>

这里我们采用国内的服务器进行下载

$wget <http://mirror.bit.edu.cn/apache/kafka/0.10.0.0/kafka>\_2.10-<0.10.0.0>.tgz

(3)安装配置 Kafka

对 Kafka 安装文件进行解压缩并且将压缩文件移动到用户安装目录下。

$tar xzvf kafka\_2.10-<0.10.0.0>.tgz

S mv kafka\_2.10-<0.10.0.0>//var/local/hadoop/

为了运行Kafka, 需要在 kafka\_2.10-<0.10.0.0> 目录的 config 文件下，修改其默认配置文 件 server.properties,需要对文件修改的内容如下：

broker.id=1 (注：在linc-2,linc-3 上broker.id 分 别 为 2 , 3 )

listeners=PLAINTEXT://linc-1:9092 (注：在linc-2,linc-3 上分别为PLAINTEXT://linc-

2:9092 PLAINTEXT://linc-3:9092)

zookeeper.connect=linc-1:2181,linc-2:2181,linc-3:2181

需要对文件新增的内容如下：

advertised.listeners=PLAINTEXT://linc-1:9092

( 注 ：linc-2,linc-3

上分别为 PLAINTEXT://

linc-2:9092 PLAINTEXT://linc-3:9092)

delete.topic.enable=true

上述的配置参数详细的叙述如表5-5所示。读者如果遇到不清楚的配置可以查阅该表。

第5章 大数据安全之 Apache Ranger 231

**表5-5** **配置参数详解**

**解** **释**

每一个Broker在集群中的唯一表示，要求是正数。当该服务器的IP地址发 生改变时，如果broker.id没有变化，则不会影响Consumers的消息情况

指定了Kafka需要连接的Zookeeper的客户端的参数，可以是一个也可以是 多个，多个之间用“;”进行分隔，为了提高可靠性，建议都添上。

(效果同之前的版本的hostname及port)为Kafka的Broker的监听端口， 是Kafka的服务端口

用来指定Kafka中的Topic是否可以被彻底删除，如果不指定这个参数， Kafka默认为false,当尝试删除Kafka中的Topic时，会将Topic标记为删

除，但是并不会彻底删除Topic的数据

表示Kafka的监听地址，这个属性可以使远端直接访问IP即可连接Kafka

Broker处理消息的最大线程数， 一般情况下不需要修改

Broker处理磁盘IO的线程数，数值应该大于硬盘数

Topic每个分区的最大文件大小。当值为-1时，表示大小没有限制

指Zookeeper的连接超时时间

advertised.listeners

num.network.threads

num.io.threads

log.retention.bytes

zookeeper.connection.timeout.ms

broker.id

zookeeper.connect

delete.topic.enable

**参** **数** **名**

Listeners

将配置好的 Zookeeper 同步到其他节点：

S scp -r /var/local/hadoop/kafka\_2.10-<0.10.0.0>

$scp -r /var/local/hadoop/kafka\_2.10-<0.10.0.0>

$scp -r /var/local/hadoop/kafka\_2.10-<0.10.0.0>

linc-2:/var/local/hadoop

linc-3:/var/local/hadoop

linc-client:/var/local/hadoop

关于设置Kafka 的环境变量前面章节中有过详细介绍，这里不赘述。

(4)启动 Kafka 服务

注意在启动 Kafka之前，需要成功启动Zookeeper。在3个节点上分别执行以下命令：

$nohup ./bin/kafka-server-start.sh config/server.properties 6

如果在 nohup 文件中没有出现出错信息，则表示 Kafka 启动成功。

在成功配置并且开启Kafka 服务后，需要对Kafka 服务进行验证，观察它是否可以正 常使用。

在linc-1 上新建 topic, 名称为 topic, 并且设置其副本为1。

$kafka-topics.sh --zookeeper linc-1:2181 --partitions 1 --replication-factor 1 --create --topic topic

在客户端利用生产者发布一些消息到特定的 topic 上。

s kafka-console-producer.sh --broker-list linc-1:9092,linc-2:9092,linc-3:9092

--topic topic

如图5-26所示，在消息生产者发布消息时，发布“This is a message from producer” 消息。

232  第二篇 开源实现篇



@ kafka\_2.10-0.10.0.815 te-2:9892,zte-3:9892 --topic topic This is a message from producer

bin/kafka-console-producer,sh

zte-1:9892,z

--broker-list

图5-26 发布消息

在客户端利用消费者来消费刚才发布的消息。

$kafka-console-consumer.sh --zookeeper linc-1:2181,linc-2:2181,linc-3:2181 --topic

topic --from-beginning

如图5-27所示，消费者得到的是刚才生产者发布的消息，说明Kafka 安装成功。



[@ kafka\_2.18-<0.10.0.015> bin/kafka-console-consumer.sh --zookeeper zte-1:2181,zte

2:2181,zte-3:2181-topic topic -from-beginning

This is a wessage from producer

~CProcessed a total of i messages

图5-27 消费消息

**4.Ranger-Kafka** **插件安装**

Ranger-Kafka 的解压与建立软连接，与前面章节中其他插件操作步骤类似，这里不赘述。 修改配置 Kafka 插件的选项：

$cd ranger-kafka-plugin

s vim install.properties

修改以下配置项：

COMPONENT\_INSTALL\_DIR\_NAME=/var/local/hadoop/kafka\_2.10-<0.10.0.0>

CUSTOM\_USER=kafka

CUSTOM\_GROUP=hadoop

REPOSITORY\_NAME=kafkadev

COMPONENT\_INSTALL\_DIR\_NAME 指定 Kafka 的安装目录。其他配置选项与前面 几节中其他插件的配置意义一致。

修改 Kafka 的安装目录中 config 下的 server.properties 文件，添加如下属性：

authorizer.class.name=org.apache.ranger.authorization.kafka.authorizer.RangerKafkaAuthorizer

前面几节中的其他插件会在执行安装后，在相应的配置目录中生成这个权限属性，在 Kakfa 配置过程中需要读者自行添加，这个属性设置了由Ranger 对 Kafka 进行授权。

将Ranger-Kafka插件所需的jar 包复制到 Kafka的 libs目录中，并且改变 libs权限为 kafka。

$cp /usr/local/ranger-0.6.2-SNAPSHOT-kafka-plugin/lib/ranger-kafka-plugin-impl/\*.jar

/var/local/hadoop/kafka\_2.10-<0.10.0.0>/libs

S sudo chown -R kafka:hadoop /var/local/hadoop/kafka\_2.10-<0.10.0.0>/libs

在/etc/profile 中添加 Kafka 的配置文件的目录，并对环境变量进行刷新。

s s

export CLASSPATH=/var/local/hadoop/kafka\_2.10-<0.10.0.0>/config

source /etc/profile

第5章 大数据安全之Apache Ranger 233

如果不添加此属性，在Kafka 插件安装完成后会找不到相应的插件配置文件。然后按 照前面几节所述步骤安装插件并启动Kafka。

**5.Ranger-Kafka** **插件功能验证**

在 linc-1中，使用浏览器打开http://linc-1:6080 网址，用户名、密码分别为 admin 、admin。 在 Service Manager 中新建一个 Kafka 服务，修改其中的配置项为：

Service Name =kafkadev

Username=kafka

Password=123456

Zookeeper Connect String =linc-1:2181,linc-2:2181,linc-3:2181

配置项中的内容与 Kafka 配置项中的内容保持一致。

保存配置信息后，单击下面的TestConnection 后显示“Connected Successfully”,说明 Ranger 已经成功连接 Kafka。如果没有成功连接的话，请根据上述配置信息核对配置文件 信息，检查是否有误。

保存后，可以在 Audit→Plugins 中看到 Kafka 插件安装记录，如图5-28所示。其中记 录了在配置文件中配置的Service Name 、Plugin Id 、IP以及 Http 服务的状态，请确认上述 信息正确无误。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Service Name | Plugin Id | Plugin IP | Http Response Code | Status |
| kafkadev | kafka@zte-1-kafkadev | <192.168.237.101> | 00 | Policies synced to plugin |

图5-28 插件安装记录

在主页的 Policy 中，进入 Kafka 的 Policy中进行以下修改：

Policy Name:all-topic

Topic:\*(\*代表所有Topic)

Select Group:public

Select User:kafka

Policy Conditions:<192.168.237.101>,<192.168.237.102>,<192.168.237.103>

Permission:Publish,Consumer,Configure,Describe,Create,Delete,Kafka Admin

在配置过程中，必须保证所有的 Broker 具 有Kafka Admin的权限，对没有安装 Kerberos的非安全模式下的Kafka,Ranger 并没有检测 Kafka 用户的权限，所以这里设置 Public, 意味着允许所有用户通过，这也是后面在Ranger 上产生记录时，User 下的记录为 ANONYMOUS 的原因。

然后我们需要在第1个节点上，使用Kafka用户创建测试 Ranger 所控制的Kafka Topic。

s kafka-topics.sh --zookeeper linc-1:2181 --partitions 1 --replication-factor 1 --create --topic testtopic

在 Kafka Policy 下新建 Policy, 参数如下：

Policy Name:testpolicy

**234** **第二篇** **开源实现篇**

注意

Topic:testtopic

Select Group:public

Select User:kafka

Policy Conditions: <192.168.237.105>

Permission:Publish,Consumer,Configure,Describe,Create,Delete,Kafka Admin

在 Policy Conditions 中需要指定的是客户端的IP 地址。注意在所有的 Broker 上不

适合对 Publish 或 者Consume 进行权限控制。

在客户端验证 Producer:

$bin/kafka-console-producer.sh --broker-list linc-1:9092,linc-2:9092,linc-3:9092

--topic testtopic

如图5-29所示，在消息生产者发布消息时，发布“This is a message from kafka producer” 消息。可根据消费者中是否可以收到生产者发送的消息来决定是否安装成功。

I\*ae et kafka 2.10-<0.10.0.01>$ bin/katka-console-producer.sh --broker-list

zte-1:9892,zte-2:9692,zte-3:9892 topic testtopic This is a message from kafka producer

^C

图5-29 发布消息

可以在Ranger 的 Web UI中查看 Audit→Access, 可以观察到 testtopic 中的生产消息

的记录，如图5-30所示。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Policy ID | Event Time" | User | Service | Resource | Access Type | Result | Access Enforcer |
| Name /Type | Name /Type |
| 5 | 11/18/201611:11:53 AM | ANONYMOUS | kafkadev  kafka | testtopic  topic | Publish | Allowed | ranger-ad |
| 5 | 11718/201611:11:52 AM | ANONYMOUS | kafkadev  kafka | testtopic  topic | describe | Allowed | ranger-ad |

图5-30 生产消息记录

在客户端验证 Consumer:

s bin/kafka-console-consumer.sh --zookeeper linc-1:2181 --topic testtopic

-from-beginning

如图5-31所示，消费者得到的是刚才生产者发布的消息，说明 Kafka 中的 testtopic 可 以进行正常通信。



[@ N=kafka 2.10-<0.10.0.01>$bin/kafka-console-producer.sh --broker-list

zte-1:9092,zte-2:9092,zte-3:9092--topic testtopic

This is a message from kafka producer

C

图5-31 消费消息

可以在 Ranger 的 Web UI中查看 Audit→Access, 可以观察到 testtopic 中的消费者消

第5章 大数据安全之Apache Ranger  235

息记录，如图5-32所示。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| User | Service  Name /Type | Resource  Name /Type | Access Type | Result | **Access** **Enforcer** |
| ANONYMOUS | kafkadev | testtopic | consume | **Allowed** | ranger-acl |
| kafka | topic |

图5-32 testtopic 消费者记录

接下来通过 Ranger改变 Kafka topic 的权限，来验证 Ranger 可以对 Kafka 进行消息控 制。修改 testpolicy 为：

Policy Name:testpolicy

Topic:testtopic

Select Group:public

Select User:kafka

Policy Conditions: <192.168.237.105>

Permission:Publish,Describe,

**在客户端验证** **Consumer:**

s bin/kafka-console-consumer.sh --zookeeper linc-1:2181 --topic testtopic -from-

beginning

如图5-33所示，可以看出消费消息时的提示信息，从TopicAuthorizationException 提 示可以看出错误是由于没有权限所导致的。



qpic from-bepi.10-<0.10.0.015> bHn/kaf Ccnsole Consumsr.Sh Zcokeeper

12616-11-1863:31:84,7151 ERROR[ConsumerFetcher Thread-console-consumer-17044

k8a7r1C,Eormeretchpearrihtd) [testtopic,0]to broker 1:kafka.common.TapicAuthorlzatianException(ka

1201611-1803:31:04,9181 ERROR IConsumerfetcherthread-console consumer-17044 1479457860621-f 8073ce1-8-11,Error for partition [testtopic,0l to broker 1:kafka.common.TopicAuthorizationException(ka

ltka.consumer ConsumerFetcher Thrend)

ZteL:ZIl tcple tett

.1479457860621-1

图5-33 没有权限消费记录

如图5-34所示，在Ranger的 Web UI中查看 Audit→Access, 可以看到消费消息拒绝 的记录，由此可以判定我们的权限设置成功，Ranger 可以对 Kafka的权限进行控制。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| User | Service | Resource | Access Type | Result | Access Enforcer | Client IP |
| Name /Type | Name/Type |
| ANONYMOUS | kafkadev  kafka | testtopic  topic | consume | Denied | ranger-ad | <192.168.237.105> |

图5-34 testtopic 消费消息记录

**5.2.8** **Ranger 集** **成** **Atlas 的安全认证机制与配置**

**1.Atlas Ranger 集成**

Atlas 提供的数据治理能力和作为一个通用的元数据存储器的目的是为了交换内外的

236 第二篇 开源实现篇

Hadoop 堆栈的元数据。Ranger 提供了一个集中的用户界面，可以用来定义、执行和管理安

全策略一致的Hadoop 的堆栈中的所有组件。Atlas-Ranger 在 Ranger的安全执行策略中统 一了Atlas 的数据分类和元数据存储功能。

读者可以用Atlas 和Ranger 实现基于安全策略的动态分类，除此之外也有基于角色的安 全策略。Ranger 的集中平台使数据管理员定义基于Atlas 元数据标签或属性的安全策略。可实 时应用此策略到整个层次结构的组成部分中，包括数据库、表和列，从而防止违反安全规定。

**2.Atlas** **访问策略**

(1)基于分类的访问控制

一个数据资产，如表或列可以被与合规性或业务分类相关的元数据标签标记。然后该 标签将用于分配给用户或组权限。这体现了基于角色授权的演变，这需要离散的静态用户 /组和资源，如表或文件之间的一对一映射。例如， 一个数据管理员可以创建一个分类标签 “PI” (个人身份信息)和分配一定的Hive 的表或列的标签 “PI” 。 这样，数据管理员会指明 存储在列或表的任何数据都被视为 “PI” 。 数据管理员现在有能力在Ranger 中为这个分类建 立一个安全策略，并允许某些群体或用户访问与此分类相关的数据，同时拒绝访问其他组或 用户。用户访问任何数据将会被归类为 “PI”,Atlas 将通过已定义了 Ranger政策自动执行。

(2)基于数据过期的访问策略

对于某些业务用例，数据可能是有害的，并业务使用的日期是有期限的。这种使用情 况下，这个业务用例可以被 Atlas 和Ranger 获取。Apache Atlas 可以指定到期日期到数据标 签。Ranger 继承到期日期，并自动拒绝访问过期的标签数据。

(3)特定位置的访问策略

类似于基于时间的访问策略，管理员现在可以制定基于地理的授权。例如， 一个处于 美国的用户当她在国内时可能被授予访问数据权限，但当她在欧洲时可能就无访问权限。 虽然相同的用户可能会试图访问相同的数据，但是不同的地理范围将实现并触发一组不同 的隐私规则，对行为进行评估。

(4)禁止数据集组合

Atlas-Ranger 集成后，可以定义一个安全策略，限制了两个数据集相结合。例如，考虑 一个场景，其中一列由客户的账户号码组成，另一列包含客户名称。这些列单独可能是合 乎规范的，但如果合并作为查询的一部分，将会违规。管理员现在可以将元数据标签应用 到两个数据集上，以防止它们被合并。

**3.跨组件的血统**

Apache Atlas 现在提供了可视化的交叉分量系的能力，俗称血统，提供跨多个分析引 擎，如Apache Storm 、Kafka 、Falcon 和Hive 的数据流动视图。

这一功能为数据管家和审计者提供了便利。例如，数据开始通过Kafka bolt 或 Storm 的拓扑结构，同时经过Hive 分析和汇总成数据组，然后结合RDBMS 通过 Sqoop 的相关数

第5章 大数据安全之Apache Ranger 237

据，可以由Atlas 在其生命周期的各个阶段进行管理。数据管家、操作和规范性如今能将 数据可视化为血统，然后向下钻取操作、安全和起源的相关细节。这是在跟踪平台的水平， 任何使用这些引擎的程序会被自动跟踪。这允许扩展的可视性超出一个单一的应用程序视 图。Ranger-Atlas 结合的具体过程如下。

**4.Ranger-Atlas** **插件安装**

在安装之前，读者必须要有一些 Linux 的基础知识，不然接下来的一些操作会比较难 以理解。

(1)解压 ranger-Atlas 插件

在集群的主节点上，使用root 用户，在任意目录下操作如下命令：

s cd /usr/local

s tar zxvf ~/dev/incubator-ranger/target/ranger-0.6.2-SNAPSHOT-Atlas-plugin.tar.gz

$ln -s ranger-0.6.2-SNAPSHOT-Atlas-plugin ranger-Atlas-plugin

$cd ranger-Atlas-plugin

ln 的链接分软链接和硬链接两种：

1)软链接就是： “In-s 源文件目标文件”,只会在选定的位置上生成一个文件的镜像， 不会占用磁盘空间，类似于Windows 的快捷方式。

2)硬链接是 “In 源文件目标文件”,没有参数-s, 会在选定的位置上生成一个与源文 件大小相同的文件。

无论是软链接还是硬链接，文件都保持同步变化。

(2)修改 Ranger-Atlas 插件的配置文件

在集群的主节点上，使用root 用户，在任意目录下操作如下命令：

s cd /usr/local/ranger-Atlas-plugin

修改配置文件，将 install.properties 文件中相应的地方修改如下：

POLICY\_MGR\_URL=http://Yours IP:6080

SQL\_CoNNECTOR\_JAR=/usr/share/java/mysql-connector-java-5.1.38-bin.jar

REPOSITORY\_NAME=Atlasdev

XAAUDIT.SOLR.ENABLE=true

XAAUDIT.SOLR.URL=http://Yours IP:6083/solr/ranger\_audits

组件对应的用户，这里设置为空。 一般 Hadoop 的内置用户是 hdfs 或者hadoop。

CUSTOM\_USER=hdfs

CUSTOM\_GROUP=hadoop

(3)软连接Atlas 下的 conf 和 lib

在集群的主节点上，使用root 用户，在任意目录下操作如下命令：

$mkdir /usr/local/Atlas

$ln -s /var/local/hadoop/Atlas-0.7/conf//usr/local/Atlas/conf

$1n-s/var/local/hadoop/Atlas-0.7/server/webapp/Atlas/WEB-INF

**238** **第二篇** **开源实现篇**

/lib /usr/local/Atlas/libext

$cd /usr/local/ranger-Atlas-plugin/

$./enable-Atlas-plugin.sh

#建立软连接后，启用Atlas 插件

#文件夹Atlas 需要新建

#若遇到缺少jar 包，将guava-17.0.jar 导入/usr/local/ranger-plugin/install/lib 即可

( 4 ) 启 动 Atlas

如果前面启动过Atlas, 需要重新启动。然后在集群的主节点上，使用root 用户，在任

意目录下操作如下命令：

$cd /var/local/hadoop/Atlas-0.7 $./bin/Atlas\_start.py

#正常情况下，若显示Apache Atlas Server

started!!!, 则表示Atlas 服务器已经启动。

( 5 ) 在 Ranger Web UI 中添加 Atlas 服务

打开浏览器，这里使用的是Chrome (没有测试过其他浏览器，用户可以自行测试)。在 浏览器中地址栏中输入：

http://Yours IP:6080

#6080为Ranger 网络服务的端口号，正如Atlas 网络端口号是21000一样

登录用户名、密码，分别为admin 、admin

#在Service Manage中新建一个Atlas 服务，修改其中配置项，标记有\*号的部分为必填的选项(Storm 配

置这一步会和其他插件不同)

Service Name =Atlasdev

Username=hdfs

Password=123456

Atlas.rest.address=http:// 读者的IP:21000

保存后，可以在Audit→Plugins 中看到 Atlas 插件安装成功。

可能遇到的问题是连接Atlas 插件会显示失败，并在ranger\_admin.log 中 报 错 ：Atlas failed to find service class org.apache.ranger.services.Atlas.RangerServiceAtlas.Resource lookup will not be available.Please make sure plugin jar is in the correct place.

但在弹出的框中没有报任何错误。

解决方法如下：

1 ) 在 /usr/localranger-0.6.2-SNAPSHOT-admin/ews/webapp/WEB-INF/classes/ranger-plugins/

路径下创建文件夹Atlas。

2 ) 进 入 ranger-Atlas-plugin-0.6.2-SNAPSHOT.jar 所在的目录，执行如下命令：

$cp ranger-Atlas-plugin-0.6.2-SNAPSHOT.jar /usr/local/ranger-0.6.2-SNAPSHOT-

admin/ews/webapp/WEB-INF/classes/ranger-plugins/Atlas

3 ) 将jar 包放入指定文件夹，则可解决插件连接不上的错误。

**5.2.9 Ranger集成** **Storm 的安全认证机制与配置**

前面几节中详细介绍了Ranger 插件的配置以及如何对插件进行设置，本节主要介绍

第5章 大数据安全之Apache Ranger **23 9**

Ranger 对Storm 的服务。Ranger 通过 Web控制台界面，可以配置管理组件的权限。Ranger 首先通过Storm 的服务管理创建一个 Service 服务，然后通过这个 Service 服务，Ranger 控 制台就可以和Storm 组件相关联；有了Storm 服务后， Ranger 就能够用它创建相关的策略 来管理Storm 的权限了。同时 Ranger 可以通过审计管理对Storm 所做的操作进行审计，通 过界面可以看到操作的日志信息，还可以对 Storm 的用户和用户组进行增加、删除、修改， 以便于对这些用户或用户组设置权限。Apache Storm的安装配置以及介绍在第4章中已经 详细介绍过，在这里不重复介绍了。本节主要介绍Ranger-Storm 插件相关的工作原理。

**1.配置项文件修改**

读者在下载安装 Storm 后，在 Storm 安装包中含有一个在 conf 目录下的storm.yaml 文 件，该文件是用于配置Storm 集群的各种属性的。 storm.yaml 会覆盖 defaults.yam1中的各 个配置项的默认值。在linc-1 节点上，使用hdfs 用户编辑 storm.yaml 文件，将该文件修改 为如下内容：

storm.zookeeper.servers:

-"linc-1"

-" linc-1"

-"linc-1"

nimbus.seeds:["linc-1"]

storm.zookeeper.port:2181

nimbus.thrift.port:6627

ui.port:8080

logviewer.port:8000

drpc,port:3772

drpc.invocations.port:3773

drpc,http.port:3774

Storm.thrift.transport:"org.apache.Storm.security.auth.kerberos.KerberosSasl- TransportPlugin"

java.security.auth.login.config:"/var/local/hadoop/apache-Storm-1.0.0/conf/Storm- jaas.conf"

java.security.krb5.conf:"/etc/krb5.conf"

Storm.principal.tolocal:"org.apache.Storm.security.auth.KerberosPrincipalToLocal"

集群号linc-1 、linc-2 、linc-3 均为本书中定义的集群号，读者可根据实际情况进行 修改。zookeeper 、ui 、logviewer 等端口号均为默认端口号，读者可根据实际情况进 行修改。

在 storm.yaml 中定义了Storm 需要访问的端口号。具体的参数解释如表5-6所示。

**表5-6** **storm.yaml 参数详解**

|  |  |
| --- | --- |
| **参** **数** **名** | **解** **释** |
| storm.zookeeper.servers | Storm关联Zookeeper集群的地址列表 |
| nimbus.seeds | 工作节点(worker node)需要知道哪台机器是主节点的候补者，以便从这些节 点下载topology的jar和confs |

**240** 第二篇 开源实现篇

(续)

|  |  |
| --- | --- |
| **参** **数** **名** | **解** **释** |
| storm.zookeeper.port | Storm所依赖的Zookeeper集群的端口号 |
| nimbus.thrift.port | nimbus的Thrift监听端口 |
| ui.port | Storm ui的服务端口号 |
| logviewer.port | Storm logviewer端口号 |
| drpc.port | Storm DRCP的相关端口号 |
| Storm.thritf.transport | 在Storm中使用Kerberos身份验证 |
| java.security.auth.login.config | 指定Storm-jaas.conf文件的地址，为了在Storm中启动Kerberos授权 |
| java.security.krb5.conf | Kerberos中krb5.conf文件的地址，krb5.conf中包含Kerberos配置信息。例如， KDC的位置，Kerberos的admin的realms等。需要所有的Kerberos机器上的配  置文件都同步 |
| Storm.principal.tolocal | nimbus需要转化本地用户名，以便其他服务可以使用这个名字 |

Storm 的安装包中还含有 一 个在conf 目录下的 storm-jaas.conf 文件。该配置项文件包 含两个内容： StormServer 和 StormClient 。 其 中 StormServer 由 nimbus 和 DRPC 节点所使 用，而 supervisor 节点则不需要使用。StormClient 由 Storm 客户端使用，以便与 nimbus、 UI 、logviewer 和 supervisor 进行会话。该文件的作用是与 Kerberos 进行授权连接。

在 linc-1 节点上，使用hdfs 用户，在/var/local/Hadoop/apache-Storm-1.0.0/conf 目录下

修改 storm-jaas.conf 文件：

StormServer {

com.sun.security.auth.module.Krb5LoginModule required

useKeyTab=true

keyTab="/var/local/hadoop/apache-Storm-1.0.0/conf/keytabs/Storm.keytab"

storeKey=true

useTicketCache=true

principal="Storm/linc-1@LINC.COM";

};

StormClient {

com.sun.security.auth.module.Krb5LoginModule required

useKeyTab=true

keyTab="/var/local/hadoop/apache-Storm-1.0.0/conf/keytabs/Storm.keytab"

storeKey=true

useTicketCache=true

serviceName="Storm"

principal="Storm/matse@LINC.COM";

};

**2.安** **装** **Kerberos 客户端及创建** **Storm 的** **principal**

在 Ranger 集 成 Storm 安全认证机制中，读者需要对 Storm 进行 Kerberos 授权，若读者 只是单独运用Storm, 则不需要安装 Kerberos, 后文配置文件中也不需要进行 Kerberos 的 相关配置。Kerberos 是一种网络认证协议，其设计目标是通过密钥系统为客户机/服务器应 用程序提供强大的认证服务。对 Kerberos 的介绍以及安装可以参考本书第7章。

第 5 章 大 数 据 安 全 之 Apache Ranger 241

注 意

在Kerberos搭建完毕后，需要在Kerberos客户端中使用 addprinc命令添加一个 Storm 用户，再使用 xst 命令生成对应的keytab。

在linc-krb节点上，使用root 用户，在/root/keytabs 目录下执行以下命令：

S kadmin.local

s addprinc -randkey

s addprinc -randkey

s addprinc -randkey

$xst -k Storm.keytab s xst -k Storm.keytab s xst -k Storm.keytab $exit

storm/linc-1@LINC.CoM

storm/linc-2@LINC.COM

storm/linc-3QLINC.COM

storm/linc-1@LINC.COM

storm/linc-2ELINC.COM

storm/linc-3ELINC.COM

读者进行以上操作后，会在kerberos目录下生成一个storm.keytab 文件，该文件就是 Storm的一个安全凭证。读者需要将该文件复制并传输到 linc-1节点上。可以使用 scp 命令 完成此项工作。

scp storm.keytab rootelinc-1:/var/local/hadoop/apache-storm-1.0.0/conf/keytabs

 读者需要在linc-1节点上使用hdfs用户，将storm.keytab文件需要的权限修改为400。

**3.复制** **Storm 目录**

完成上述内容后，需要将linc-1 节点上配置好的 Storm目录全部复制到其他所有节 点上，目录所存放的文件位置在所有节点上必须是一致的。读者可以使用 scp命令来完成 此项工作，如下命令为一条示例，用于将linc-1 节点上的apache-Storm-1.0.0目录复制到 linc-1 节点的/var/local/hadoop目录下，读者可以根据实际情况来进行修改。如果节点数目 比较多的话，建议读者编写一个 Shell 脚本完成此项任务。

$scp -r apache-Storm-1.0.0/hdfs@slavel:/var/local/hadoop/

在将 Storm目录复制完成后，需要在其他节点上将上述 Storm-jaas.conf 文件中的Storm/ linc-1@LINC.COM 中的 linc-1名称改为其他节点对应的名称，例如， linc-2。读者可根据实 际情况进行修改。

**4.运行和关闭** **Storm** **集群**

读者在使用Storm 之前，为了方便Storm 启动/关闭，建议在所有节点上将 Storm的 bin目录加入系统的PATH环境变量当中，这样设置以后，在执行bin 目录下的程序时，可 以直接使用程序名，例如 “storm ui”, 而无须指定程序的路径，如“./bin/Stormui” 。 具体 设置步骤如下。

在 所 有 节 点 上 使 用root 用 户 ， 在 任 意 目 录 下 使 用 下 列 命 令 编 辑 文 件 /etc/profile:

vim /etc/profile

\_下H语O句ME=/path/to/storm

**242** **第二篇** **开源实现篇**

$export PATH =SSTORM\_HOME/bin

#保存退出

其中/path/to/storm 需要替换成用户完整的 Storm 安装路径，在这里应该写为/var/local/ hadoop, 读者可以根据实际情况进行修改。

执行命令 “source /etc/profile” 将上述添加内容更新，完成后在任意目录下执行命令 “storm -help”, 若没有提示“命令找不到”,则说明设置成功。

②素 Storm 的启动分为3步： nimbus、supervisor、ui。读者在启动 Storm 之前需要启动

Zookeeper.

1)启动nimbus: 在 linc-1 节点上执行命令 “storm nimbus&”, 在启动 nimbus时，需要 等待几秒钟，控制台将会输出系列路径地址，等到输出结束后，读者可继续进行后续操作。

再执行命令“jps”会显示后台运行的进程，读者可以

[ecotn⁸zte-1~]51PB

看到 nimbus 进程及进程号，说明 nimbus 启动成功，如 2066 nimbug

图5-35所示。 Jps QucrumPeerMain

2)启动 supervisor: 在所有节点上执行命令 “storm [SEerm?zte-1 -1s

图5-35 nimbus 进程号

supervisor &” 启动 supervisor, 也需要等待几秒后，控制

台会有系列路径地址输出，等到输出结束后，读者可继续进行后续操作。

执行命令 “jps” 可以看到 supervisor 进程及其进程号，说明 supervisor 启动成功。

3)启动 Storm ui: 在 linc-1 节点上执行命令 “stormui&”, 在启动ui 时，需要等待几 秒钟后，控制台将会有输出。执行命令 “jps” 可以看到 “core” 及对应的端口号。

在Storm ui启动后，读者可以打开浏览器，访问http://linc-1:8080, 其中 linc-1需要替换 成自己启动nimbus 节点(主节点)的真实IP 地址，可以看到Web监控界面，如图5-36所示。 该页面显示了集群所有节点信息， Nimbu s 、Topology 、Supervisors, 以及Nimbus 配置信息。

|  |  |  |
| --- | --- | --- |
| **Storm** **UI** | | |
| **Cluster** **Summary** | | |
| Verslon **Supervisors** Used slots Free slots Total slot Fxecutors Tasks | | |
| 1.0.0 0 4 4 0 0 | | |
| **Nimbus**  Host | **Summary**  \***Port** | st Verslon UpTime |
| zte-1 6627 Leader 1.0.0 30m 43s | | |
| Showng 1 to 1 of 1 ennes | | |
| **Topology** **Summary** | | |
| Name Owner StatusUptime Num workers Num executors Num tasks Replication count Asslgned Mem (MB) Scheduler Info | | |
| No data avallable in table | | |

图5-36 Storm UI界面

由于 nimbus 进程和 supervisor 进程都是快速失败 (fail-fast) 和无状态的，因此，读者 可以用 kill-9 来杀死 nimbus 和 supervisor 进程。执行命令“kill-9111222333” 则可以关

第5章 大数据安全之 Apache Ranger 243

闭 Storm 所有进程。其中111、222、333分别表示 nimbus 、supervisor 、ui 的进程号。

**5.Ranger-Storm** **插件的安装**

(1)解压插件

在linc-1 节点上，使用root 用户，进入Ranger的安装目录下，找到 Ranger-Storm-plugin 的压缩包，例如“～/dev/incubator-Ranger/target/Ranger-0.6.2-SNAPSHOT-Storm-plugin.tar. gz” 。 将该压缩包进行解压，读者可用使用 “tar” 命令完成该解压过程。然后进入解压后 的 Storm 插件文件夹。在这里建议读者可以在/usr/local 目录下使用命令 “In” 建立一个软 链接，软链接的名称更加简单，且能明白这个链接的作用，同时软链接也可以放到读者想 放的文件夹中，更加方便管理。

$1n -s Ranger-0.62-SNAPSHOT-Storm-plugin Ranger-Storm-plugin

(2)修改插件配置项文件

进入解压缩得到Storm 插件目录后(软链接),读者会发现一个 install.properties 文件， 使用 “vim” 命令编辑该文件，将文件中的属性按照如表5-7所示进行修改。

**表5-7** **install.properties 修改内容**

|  |  |
| --- | --- |
| PROPERTY | VALUE |
| POLICY\_MGR\_URL | http://linc-1:6080 |
| SQL\_CONNECTOR\_JAR | /usr/share/java/mysql-connector-java.jar |
| REPOSITORY\_NAME | stormdev |
| XAAUDIT.SOLR.ENABLE | true |
| XAAUDIT.SOLR.URL | http://localhost:6083/solr/Ranger\_audits |
| CUSTOM\_USER | hdfs |
| CUSTOM\_GROUP | Hadoop |

各属性的含义如表5-8所示。

**表5-8** **install.properties** **参数含义**

|  |  |
| --- | --- |
| **属性名** | **含** **义** |
| POLICY\_MGR\_URL | 指定当前Apache Storm服务权限控制的URL,其中linc-1可以换为读者安装 Ranger Admin所指定的机器名(也可以为IP地址) |
| SQL\_CONNECTOR\_JAR | 指向数据库连接jar包的位置，这里是用MySQL作为数据库，读者根据实际的 安装地址进行修改 |
| REPOSITORY\_NAME | 策略的名称 |
| XAAUDIT.SOLR.ENABLE | 将Solr作为审计日志 |
| XAAUDIT.SOLR.URL | Solr的URL地址 |
| CUSTOM\_USER | 安装当前插件所指的用户信息 |
| CUSTOM\_GROUP | 安装当前插件所指的用户组的信息 |

244 第二篇 开源实现篇

(3)进行软链接

Ranger鉴权的本质是通过安装组件时生成的配置文件以及组件自带的jar 包，通过 HOOK 方式调用组件服务达到权限管理。

在linc-1节点上，使用root用户进入插件安装目录，例如上文中建立的软链接。在/ Ranger-Storm-plugin目录下执行命令“./enable-Storm-plugin.sh”,该命令的目的是安装 Storm插件。但是读者会发现控制台会提示在/usr/local/Storm/conf 目录下找不到安装文件 和/ustr/local/Storm/extlibdaemon目录下找不到相关jar包，这是因为在进行插件安装时， 系统会自动到特定目录 conf 和lib 下寻找 Storm 相关文件和 jar包，因此用如下命令进行 软链接，将Storm 安装目录下的conf 和lib 软链接到系统要求的目录下，使得插件正确 安装。

$ln -s ./Storm/conf /usr/local/Storm/conf

$ln -s ./Storm/lib /usr/local/Storm/extlib-daemon

需要注意的是，命令中“./Storm” 是用户安装 Storm 时的目录。

(4)启动插件

进行上述的软链接后，再次回到 Ranger-Storm 插件解压后的目录下进行插件的安装， 在linc-1节点上，使用root 用户执行以下安装命令：

$./enable-str-plugin.sh

读者在进行 Ranger-Storm插件安装后，需要重启Zookeeper 和 Storm。若 Ranger 之前 已经启动，也需要重新启动。但是读者在重启Storm 时会启动失败，这是因为启动 Ranger- Storm的脚本后将./Storm/conf权限自动修改为了644,修改了 conf下的 kerberos 文件夹的 权限，导致配置了Kerberos的 Storm无法正常启动。因此需要将 kerberos 文件夹权限修改 回775。在修改了kerberos 文件夹权限后，便可以正常启动 Storm了。

(5)在 Ranger Web UI 中添加 Storm服务

在正确安装了Ranger-Storm插件并开启Ranger、Zookeeper、Storm 、Solr后，读者可 以登录 Ranger Admin,在浏览器中输入http://inc-1:6080, 输入账号、密码： admin 、admin。 其中linc-1 是主节点的IP 地址。在Service Manage 中新建一个 Storm 服务，如图5-37 所示。

在Service Manage 中的必须填写的配置项参数如表5-9所示。

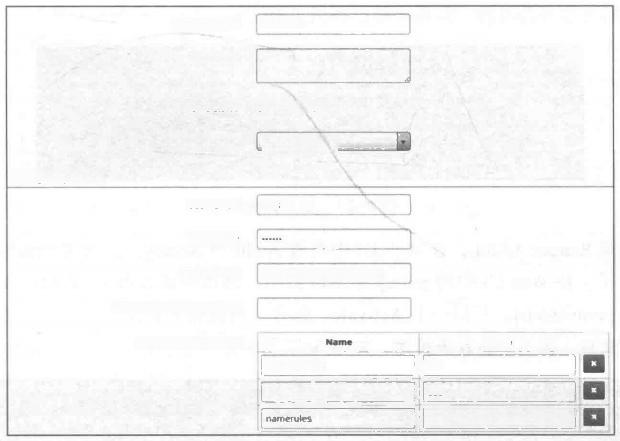
其中Service Name是指服务名称，在前面Storm 配置文件install.properties 中配置 过。UserName 是用户名，在install.properties 也配置过。Nimbus URL 是 nimbus 的 IP 地址。

在 Add New Configuration 下面需要添加3个新的属性。

1)lookupkeytab, 对应的值为非空任意值。

2)lookupprincipal, 对应值为非空任意值。

第5章大数据安全之Apache Ranger 245

Service Name\* stormdev

Description

Active Status

Select Tag Service

**Config Properties:**

usename

\*

Password

Nimbus URL\*

Common Name for Certificate

Add New Configurations

● Enabled ● Disabled

selet Tag Servke

hdfs

hitp://<192.168.237.101>:8080

lookupkeytab

lookupprincipal

**yalut**

abc

abc

RULEL251@4OMIIM@\*ZDHCOM

图5-37 新建Storm 服务

3)namerules, 值为： RULE:[2:$1@$0]([jt]

t@.\*LINC.COM)s/.\*/$MAPRED\_USER/RULE:

**表5-9** **配置项参数**

[2:$1@S0]([nd]n@.\*LINC.COM)s/.\*/$HDFS\_USER/

|  |  |
| --- | --- |
| Service Name | stormdev |
| Username | hdfs |
| Password | 123456 |
| Nimbus URL | http://linc-1:8080 |

DEFAULT。

再单击连接测试，会显示连接成功。

中看到

在保存后，可以在Audit→Plugins

Storm 插件安装成功，如图5-38所示。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Export Date(中国标准时间)” | Seryice Name | Plugin ld | Plugin IP | Http Response Code | Status |
| 11/08/201604:12:55 PM | stormdev | stormozte1.zdh.com-stormdev | <10.4.20.231> | 200 | Policies synced to plugi |
| 11/08/201603:51:47 PM | stormdev | storm@zte1.zdh,com-stormdev | <10.4.20.231> | **200** | Policies synced to plugin |
| 11/08/201602:50:12 PM | stormdev | stormozte1 .zdh.com-stormdev | <10.4.20.231> | 200 | Policies synced to plugin |
| 11/08/201602:18:13 PM | stormdev | stormoztel.zdh.com-stormdev | 10,4.20.231 | 200 | Policies synced ta plugin |

图5-38 Storm 插件安装成功

单击 stormdev, 添加用户组，再在 Select Group 中添加 public 用户组。 (6)运行Example 程序

在 Storm 中有自带的例子，只需要读者提交即可。在linc-1上，使用hdfs用户，在 Storm 安装目录的 bin 目录下执行如下命令，提交名为 StatefulTopoArgs1 的文件。

s./Storm jar /var/local/hadoop/apache-Storm-1.0.0/examples/Storm-starter/storm- starter-topologies-1.0.0.jar/Storm.starter.StatefulTopology StatefulTopoArgs1

246 第二篇 开源实现篇

在提交后需要等待几秒，控制台输出如图5-39所示。



P829 [x1n]INF0 0.a.5.M.n.Loyin -suecesefuliy 1ogged in.

sg lmtn]Rro o.a.s.m,b.Leyth -auecesstuliy ioined ia.

9149 [msin]WARN 0.a.s.u.BimbusClient -Uaing deprecated config nimbus.hase tar backward comPaciktlity.Please update your store.yanl so it only hes config nir

kus.seeds

5159 [maip]INFO 0.a:3,.n.Login -successtully 1agged in.

3181 [main]INF⁰0.a.3.m.n.Login -suecessfully 1ogged in.

3209 [msin]WAR 0.a.3.u.RAmbusC11ent -Using deprecated config ntabus.host for

baekvard cempatibtlitY.Please update your storn.yenl so it only hag conf1g nih

bus.seedg

图5-39 控制台输出

再次登录 Ranger Admin, 在 Web UI 中查看 Audit→Access, 会发现 StatefulTopoArgs1 已经被提交了。在Web UI中的 policy 中单击按钮，修改原始 policy (也可以新建 policy), 修改其中的 permissions, 只留一个 Activate, 保存，再次提交文件。

文件提交时，提示一个权限错误，如图5-40所示。

**exception in Chread Wmsin"Java.lang.Rntimelxcepeion:AuchorizationExcepeion (mg**

**at org apache,storm.StormSubmitter.submitTonologyAs(Stormsubmitter.java:**

图5-40 报错提示

然后在 Web UI中查看 Audit→Access, 会发现刚才提交的文件被拒绝了，如图5-41 所示。因此说明，Ranger 集成 Storm 的安全机制与配置完成。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Policy ID | Event Time | User | Service | Resource | Access Type | Result | Access Enforcer |
| Name /Type | Name /Type |
|  | 11/16/201602:26:04 AM | storm | stormdev  storm |  | fileUpload | Denies | ranger-acl |
|  | 11/16/201602:24:17 AM | storm | stormdev  storm | StatefulTopoArgs3 topology | submitTopology | **Allowec** | ranger-ad |

图5-41 文件提交被拒绝

**5.2.10** **Ranger集成** **Solr的安全认证机制与配置**

Solr 是 Apache Lucene 项目下的一个开源的企业搜索平台。其主要功能包括：全文搜 索，命中标识，分面搜索，实时索引，动态聚类，数据库集成，NoSQL, 以及富文档(如 Word 、PDF) 处理。由于提供了分布式搜索和索引复制，Solr 的可扩展性和容错能力都很 强。强大的开源社区更是让Solr 在诞生后10年期间快速演进。也可以称其为一个独立的企 业级搜索应用服务器，非常接近于我们平常所知的搜索引擎系统，通过各种API 可以让你 的应用使用搜索服务，而不需要将搜索逻辑耦合在应用中。而且Solr 可以根据配置文件定 义数据解析的方式，提供XML/HTTP 和 JSON/Python/Ruby API接口。用户可以通过http 请求，向搜索引擎服务器提交一定格式的XML 文件，生成索引；也可以通过 Http Get 操作 提出查找请求，并得到XML 格式的返回结果。

第5章大数据安全之Apache Ranger 247

Solr 本身就可部署成分布式形态，这和Hadoop 非常契合，因而在大数据领域它也是非 常流行的数据搜索框架，为大数据搜索提供了强有力的支持。但也请注意，本节的 Solr 与 上文 Ranger 安装部分中的Solr 有所不同，上文中的Solr 的作用是 Ranger-audit 审计，是 Ranger 运行的基础服务，运行于6083端口；而本节所说的Solr 是真正做搜索之用的 Solr, 运行于8983端口。Atlas 就会依托于此端口的Solr 进行搜索的相关操作。

当然，本节的重点是通过Ranger 对 Solr 的权限进行管理，给不同用户授予不同的 Solr 权限。下面就从安装到鉴权，再到授权， 一步步带领读者来进行实践。

|  |  |  |
| --- | --- | --- |
| 由于 Solr 运行所需要的内存空间较大，至少保证主节点拥有2G 以上内存。 | | |
| **1.初期准备**  首先我们进行一些初期的准备工作。在此部分，我们添加 Solr 运行时所属用户，此处 用户名也定义为 solr。然后将 solr 用户升级为 sudo 用户。  新建 solr 用户，在linc-1 上，使用root 用户执行如下命令：  $useradd solr -g hadoop  $echo "123456"I passwd --stdin solr  **成功后显示如下：**  Changing password for user solr.  passwd:all authentication tokens updated successfully.  将 solr 用户升级为 sudo 用户，在linc-1 上，使用root 用户执行如下命令：  s vim /etc/sudoers  找到以下原有的文件内容：  root ALL=(ALL) ALL  在之后加上：  solr ALL=(ALL) ALL | | |
| Q | 注 意 | 保存文件时使用“:wq!” 可强制写入(因为此文件不可随意更改)。 |

**2.Solr** **的安装与配置**

接下来我们就需要安装作为搜索之用的Solr, 而非之前 Ranger运行部分的 Solr 。并且 要保证此 Solr 能够正常运行。

首先到 Apache 官网下载5.5.1版本的 Solr, 在 linc-1 上，使用 solr 用户，在/home/solr 目录下执行如下命令：

s wget <http://archive.apache.org/dist/lucene/solr/5.5.1/solr-5.5.1.tgz>

s tar -zxvf solr-5.5.1.tgz

**248** **第二篇** **开源实现篇**

S mv solr-5.5.1 /var/local/hadoop

在linc-1 上，使用solr 用户，在/var/local/hadoop/solr-5.5.1/bin 目录下，执行如下命令 对 Solr 启动脚本进行修改：

s vim solr.in.sh

修改以下配置项：

SOLR\_HEAP="1024m"

**在该文件的末尾加入如下内容：**

ZK\_HOST=linc-1,linc-2,linc-3/solr

-SDDOp-yf2ctory

上述修改为Solr 的一些运行项修改，设置堆的内存大小， Solr 的运行依托于Zookeeper 和hdfs 。第1行选项为指定 Zookeeper 内 Solr 相关配置存放位置。Solr 相关的配置项都会 放入此位置，这个文件夹在之后我们会手动创建。第2行为其运行的一些操作选项，包含 文件工厂模式、锁类型、Solr 在 hdfs 内的存放位置，以及指定 hdfs 的配置文件。

在linc-1 上，使用solr 用户，在目录/var/local/hadoop/solr5.5.1/server/solr/configsets/ data\_driven\_schema\_configs/conf 下，操作如下命令对 Solr 的配置文件进行修改：

s vim solrconfig.xml

**找到原来的<directoryFactory>标签，改成如下内容：**

<directoryFactory name="DirectoryFactory"class="solr.HDFSDirectoryFactory">

<str name="solr.hdfs.confdir">/var/local/hadoop/hadoop-2.7.2/etc/hadoop</str> <str name="solr.hdfs.home">hdfs://linc-1:9000/solr</str>

<bool name="solr.hdfs.blockcache.enabled">true</bool>

<int name="solr.hdfs.blockcache.slab.count">1</int>

<bool name="solr.hdfs.blockcache.direct.memory.allocation">true</bool>

<int name="solr.hdfs.blockcache.blocksperbank">16384</int>

<bool name="solr.hdfs.blockcache.read.enabled">true</bool>

<bool name="solr.hdfs.blockcache.write.enabled">true</bool>

<bool name="solr.hdfs.nrtcachingdirectory.enable">true</bool>

<int name="solr.hdfs.nrtcachingdirectory.maxmergesizemb">16</int>

<int name="solr.hdfs.nrtcachingdirectory.maxcachedmb">192</int> </directoryFactory>

找到locktype 标签，并将里面的内容替换为 hdfs。

对Solr config配置文件进行修改，主要修改两部分， 一是将 DirectoryFactory 修改为 hdfs 的相关配置，还有就是修改 locktype 类型为 hdfs。

下面的操作即是上文所提及的在 Zookeeper 上创建 Solr 配置存放位置的文件夹。

$cd /var/local/hadoop/solr-5.5.1

第5章 大数据安全之 Apache Ranger 2 49

s server/scripts/cloud-scripts/zkcli.sh -zkhost linc-1,linc-2,linc-3 -cmd makepath

/solr

在 Zookeeper 上创建 Solr 的 chroot 目录，该目录只需要创建一次。

至此 Solr 的基础配置已经完成。下面我们开启 Solr。

在linc-1上，使用solr用户，在/var/local/hadoop/solr-5.5.1 目录下操作如下命令来启动

Solr:

$./bin/solr start -c

以 cloud 模式启动 Solr, 关闭的命令如下：

$./bin/solr/stop -c -p 8983

既然是搜索，那么肯定就需要有数据存在。下面我们就为 Solr 创建一个 collection, 此 collection 所用的配置即是我们上文修改过的 data\_driven\_schema\_config 文件夹下的相关配置。

在linc-1上，使用solr用户，在/var/local/hadoop/solr-5.5.1 目录下操作如下命令：

9 hadoop dfs -chmod -R 775 /

$hadoop dfs -chown -R hdfs:hadoop /

$./bin/solr create -c gettingstarted -d server/solr/configsets/data\_driven\_ schema\_configs/conf -shards 1 -replicationFactor 1

若节点一直无法启动，可以尝试删除节点./bin/solr delete -c indexname, 再次执行上面 的创建命令。

**3.Solr测试**

因为 Solr是可以支持Json数据查询的，下面我们上传一个测试例，并且对 collection 进行测试，以确保Solr可以正常工作。在linc-1上，使用solr用户，在/var/local/hadoop/ solr-5.5.1 目录下操作如下命令：

s bin/post -c gettingstarted example/films/films.json

http://linc- 1:8983/solr/gettingstarted/select?q=film

#此时可以看到搜索出来的XML结果信息

#或者在linc-client 上使用浏览器访问http://linc-1:8983/solr/gettingstarted/select?q=film 并查看结果

**4.Solr** **的鉴权**

为完成此部分工作，请先确保JCE、Kerberos 正常安装， Kerberos安装及详细介绍可 参考第7章，在这里不介绍。接下来为了保证正常运行不出错，我们再次进行NTP 时间校 正，以保证各个节点间的时间一致且准确。

在所有节点上，使用任意用户，在任意目录下操作如下命令：

$yum -y install ntpdate

$ntpdate us.pool.ntp.org

250 **第二篇** **开源实现篇**

如果显示连接失败，则尝试使用以下命令：

s ntpdate time.nist.gov

或

S ntpdate cn.pool.ntp.org

如果上述方法均无法成功同步时间，可以采用其他的时间同步方式，只要满足各节点 之间时间同步即可。

时间同步后，创建 Solr的 principal, 在 linc-krb上使用root 用户，在/rootkeytabs 目录 下操作如下命令：

$kadmin.local

$addprinc -randkey solr/linc-1@LINC.COM

$addprinc -randkey HTTP/linc-1@LINC.COM

$xst -k solr.keytab solr/linc-1@LINC.COM

$xst -k HTTP.keytab HTTP/linc-1@LINC.COM

$exit

$scp solr.keytab linc-1:/opt/solr/conf/keytabs

$scp HTTP.keytab linc-1:/opt/solr/conf/keytabs

操作成功后可以在该目录下看见 solr:keytab 和zkcli.keytab文件。

修改 keytab权限，在 linc-1上使用root 用户，在/opt/solr/conf/keytabs 目录下操作如下 命令：

$chown solr:hadoop solr.keytab

$chmod 444 solr.keytab

S chown solr:hadoop HTTP.keytab

$chmod 444 HTTP.keytab

修改 solrin.sh 文件，在 linc-1上，使solr用户，在/var/local/hadoop/solr-5.5.1/bin 目录 下操作如下命令：

$vim solr.in.sh

在该文件的末尾加入如下内容：

SOLR\_JAAS\_FILE=/opt/solr/conf/solr\_jaas.conf

SOLR\_HOST=`hostname -f`

ZK\_HOST="linc-1:2181,linc-2:2181,linc-3:2181/solr"

RERO\_KS\_ER/conf/keytabs/solr.keytab

SOLR\_KERB\_PRINCIPAL=HTTP/linc-1ES{KERBEROS\_REALM}

SOLR\_KERB\_KEYTAB=/opt/solr/conf/keytabs/HTTP.keytab

SOLR\_AUTHENTICATION\_CLIENT\_CONFIGURER="org.apache.solr.client.solrj.impl.Krb5Http

ClientConfigurer"

SOLR\_AUTHENTICATION\_OPTS="-Dsolr.kerberos.jaas.appname=SolrClient

-DauthenticationPlugin=org.apache.solr.security.KerberosPlugin

-Djava.security.auth.login.config=SSOLR\_JAAS\_FILE

-Dsolr.kerberos.principal=S(SOLR\_KERB\_PRINCIPAL}

**第5章** 大数据安全之Apache Ranger 251

-Dsolr.kerberos.keytab=S{SOLR\_KERB\_KEYTAB}

-Dsolr.kerberos.cookie.domain=S(SOLR\_HOST}

-Dhost=S{SOLR\_HOST}

-Dsolr.kerberos.name.rules=DEFAULT"

-S=HDFSDirectoryFactory -Dsolr.lock.type=hdfs

--doop-2.7.2/etc/hadoop"

**5.创建solr\_jaas.conf 文件**

在linc-1 上，使用root 用户，在/opt/solr/conf/目录下操作如下命令：

$vim solr\_jaas.conf

在该文件中添加如下内容：

SolrClient {

com.sun.security.auth.module.Krb5LoginModule

required

yaTbat/solr/conf/keytabs/solr.keytab"

storeKey=true

useTicketCache=true

debug=true

principal="solr/linc-1@LINC.COM";

};

6.上传 security.json 文件

在linc-1 上，使用solr 用户，在/var/local/hadoop/solr-5.5.1 目录下操作如下命令：

s server/scripts/cloud-scripts/zkcli.sh -zkhost linc-1:2181,linc-2:2181,linc-3:2181 -cmd put /solr/security.json '{"authentication":{"class":"org.apache.solr. security.KerberosPlugin"}}'

**接下来重启** **Solr:**

S solr ./bin/solr stop -c -p 8983

$./bin/solr start -c

再次打开 Web UI界面，会发现401 错误：

HTTP ERROR 401

Problem accessing /solr/.Reason: Authentication required Powered by Jetty://

解决步骤如下：

1)在linc-1上，使用root 用户。

2)打开 Firefox 浏览器，在地址栏中输入 about:config, 进入设置页面。 3)在顶部的search 栏中输人 network.negotiate-auth.trusted-uris。

4)双击行，在对话框中输入http://linc-1,http://inc-2,http://linc-3。

5)单击OK 按钮。

**252** **第二篇** **开源实现篇**

**6)进入** **Terminal, 执行以下命令：**

s kinit -kt /opt/solr/conf/keytabs/HTTP.keytab HTTP/linc-18LINC.COM

再次打开 http://linc-1:8983, 可以正常进入了。

7.Solr的授权与 Ranger-Solr 插件安装

在linc-1 上，使用root 用户，在/usr/local 目录下操作如下命令：

$tar zxf~/dev/incubator-ranger/target/ranger-0.6.2-SNAPSHOT-solr-plugin.tar.gz

$In -s ranger-0.6.2-SNAPSHOT-solr-plugin/ranger-solr-plugin

$cd ranger-solr-plugin

$vim install.properties

**修改以下配置项：**

COMPONENT\_INSTALL\_DIR\_NAME=/var/local/hadoop/solr-5.5.1/server

POLICY\_MGR\_URL=http://linc-1:6080

SQL\_CONNECTOR\_JAR=/usr/share/java/mysql-connector-java-5.1.38-bin.jar

REPOSITORY\_NAME=solrdev

XAAUDIT.SOLR.ENABLE=true

XAAUDIT.SOLR.URL=http://linc-1:6083/solr/ranger\_audits

CUSToM\_USER=solr

CUSTOM\_GROUP=hadoop

接下来执行连接脚本：

$./enable-solr-plugin.sh

上传 Solr 所需配置项：

s cd /opt/solr

$server/scripts/cloud-scripts/zkcli.sh -zkhost linc-1:2181,linc-2:2181,linc-3:2181

cmd put /solr/security.json'{"authentication":{"class":"org.apache.solr.

security.KerberosPlugin"},"authorization":{"class":"org.apache.ranger.

authorization.solr.authorizer.RangerSolrAuthorizer"})'

在 Ranger Web UI界面中设置 Solr-plugin。

在 Access Manager 中添加 Solr 服务，如图5-42所示。

添加Solr 服务后，单击solrdev 进入 Solr plugin界面，单击Add New Policy 按钮可添 加管理权限，添加示例如图5-43所示。

**8.结果验证**

做完上述全部操作后，重启 Solr。

$/var/local/hadoop/solr-5.5.1/bin/solr $/var/local/hadoop/solr-5.5.1/bin/solr

stop

start

-c.

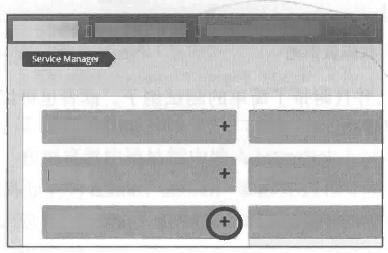
-c

然后在Ranger→Audit→Plugin 中可见如图5-44所示信息。

进入 Solr 安装目录：

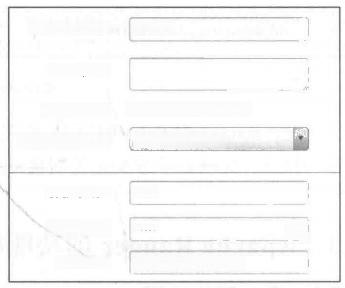
$cd /var/local/hadoop/solr-5.5.1

第5章 大数据安全之Apache Ranger 253



|  |  |
| --- | --- |
| **Ranger** TAccess Mansge  Service Manager  HDFS  YARN  5OLR | Audit Setting  ≥HBASE  ≥KNOX  KAFKA |

图5-42 添加Solr 服务



Service Name \* solrdev

Description

● Enabled ● Disabled

Select Tag Service

root

\* …

http://zte-1:8983/solr/

Active Status

Select Tag Service

Password

Solr URL\*

Username\*

回

图5-43 Policy 添加示例

**Ranger** Access Mansger Audit 0 Settins ad

Q.Search for your plugins..

Last updated Time: **bzotY ?szP** c

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Export** **Date(中国标准时间)"** | **Service** **Name** | **Plugin** **lo** | **Plugin** **IP** | **Http** **Response** **Code** | | **Status** |
|  | 10/28/201607:16:17 PM | solrdev | solrozte-1-solrdev | <192.168.91.161> | **200** |  | Policies synced to plug |
| 10/28/201603:20:28 PM | solrdev | solr@zte-1-solrdev | <192.168.91.161> |  | | Policies synced to plug |

**图5-44** **插件显示信息**

**给solr 用户授权：**

s kinit -kt conf/keytabs/solr.keytab solr/linc-1eLINC.COM

s curl --negotiate -u:http://linc-1:8983/solr/gettingstarted/select?q=film

**solr 用户有权限可以正常查询，查询结果如图5-45所示。**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Policy** **ID** | **Event** **Time** | **User** | **Service** | **Resource** | **Access** **Type** | **Result** | **Access** **Enforcer** |
| **Name/Type** | **Name/Type** |
|  |  |  |  |  |  |  |  |
| 3 | 10/29/201603:24:16 AM | solr | solrdev | gettingstarted | query | **Allowed** | ranger-ag |
| solr | collection |
|  |  |  |  |  |  |

图5-45 查询结果

切换为 test 用户，执行以下命令：

$su test

$kinit -kt /opt/solr/conf/keytabs/HTTP.keytab HTTP/linc-1@LINC.COM

$curl --negotiate -u:http://linc-1:8983/solr/gettingstarted/select?q=film

因没有权限，所以不能正常查询而被拒绝，如图5-46所示。

254 **第二篇** **开源实现篇**

|  |  |  |  |
| --- | --- | --- | --- |
|  | 10/29/201603:24:07 AM HTTP | solrdev  solr | query Denied ranger-acl  gettingstarted  collection |

图5-46 没有权限被拒绝

至此，Ranger 对 Solr 插件的集成完成了，并且附带了简单的测试例子。读者也可以如 此举一反三，实现更多的Solr 控制相关的权限管理。

**5.3** **Apache Ranger 的功能配置**

**5.3.1** **Tag** **同步验证**

Ranger 的 Tag 同步功能是 Atlas 与Ranger 集成的一个非常重要的功能，Ranger 如果能 将Atlas 中设置的 Tag 读取出来，并存储在 Ranger 服务器中，那么 Ranger就可以对Tag 设 置的内容进行权限控制，这样就能实现更细粒度的权限控制，对系统有更完整、更方便的 功能配置。但是因为 Tag是基于Atlas的，对于Atlas不支持的组件 Ranger也会受到一部分 影响，具体将在之后的功能配置中说明。

实现 Ranger同步认证需要配置Ranger Tagsync 服务，第一步是安装 Tagsync 插件。 Tagsync 插件能让 Ranger读取Atlas 中设置的 Tag, 甚至会自动为一些Tag设置权限。而对 于另一些Tag, 系统能读取 Atlas 设置的 Tag 含义，并保存在 Ranger 服务器中。

Tagsync 插件安装步骤如下。

**1.配置** **Ranger Tagsync 服务**

首先应该安装 Tagsync 插件，而如同之前安装其他插件那样，首先需要解压编译好的 包，并设置链接。

$cd /usr/local

s tar zxvf ~/dev/incubator-ranger/target/ranger-0.6.2-SNAPSHOT-tagsync.tar.gz

91n -s ranger-0.6.2-SNAPSHOT-tagsync/ranger-tagsync

设置链接之后，需要修改安装配置，为之后的安装做好准备。

$cd /usr/local/ranger-tagsync

s vim install.properties

nrest.us修改如下：

ranger.tagsync.source.atlasrest.password=admin

hadoop\_conf=/var/local//hadoop/hadoop-2.7.2/etc/hadoop

上述配置中， username 和 password 指的是 Atlas Web UI的登录账户名和密码， 一般默 认是admin/admin 。如读者修改了默认用户名与密码，则需要对上述两个配置进行修改。而 Hadoop\_conf 应该与实际情况中 Hadoop 的存放位置相匹配。

第5章 大数据安全之Apache Ranger 255

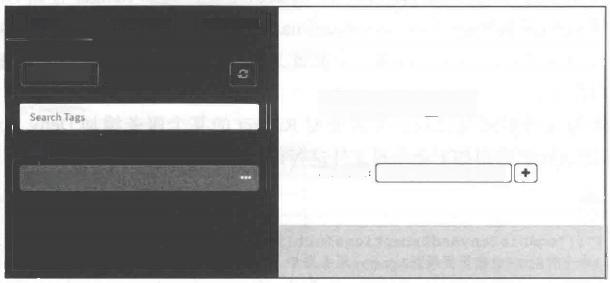
配置完成之后，就可以安装插件并启动服务了。服务启动之后，如果在 Ranger 的 Web UI Setting 中查询到新创建的用户 ranger-tagsync, 则说明插件安装成功。

$./setup.sh

$service ranger-tagsync start

**2.添加** **Atlas Tag**

在浏览器中地址栏中输入http://link-1:21000, 登 录 Atlas Web UI,默认的登录用户名和 密码分别为admin 、admin, 单击左边框中的菜单栏中的TAGS, 然后再单击Create Tag 按 钮，添加两个Tag, 分别为PⅡ 和 EXPIRES\_ON。接下来对EXPIRES\_ON 进行额外设置，单 击EXPIRES\_ON, 在 Attributes 文件框中添加 expiry\_date='2016/10/10', 最终效果如图5-47 所示。

TAGS TAXONOMY

+CreateTag

PI

EXPIRES ON

Q SEARCIH

**EXPIRES\_ON**

Attributes: expiry\_date=2016/10/10'

图5-47 添加 Atlas Tag

**3.验证** **Tag 同步**

在浏览器中的地址栏中输入http://link-1:6080, 即 Ranger 的 Web UI 地址。默认的登录 用户名和密码分别为admin 、admin。登录成功之后，单击Access Manager, 找到 Tag Based Manager 列表，通过单击列表中TAG旁边的加号增加一个 Service, 并命名为 tagdev。

配置好后，单击保存按钮，返回 Service Manager界面。然后单击 tagdev, 进入 tagdev Policies 页面，可以看到列表上添加了一条 Tag Policy记录，则表示 Atlas 中的 Tag已经同 步到Ranger 服务器上了。

**5.3.2** **各类** **Policy 验证**

Ranger提供各种 Policy 验证，这些 Policy 验证大部分已经包含在系统中了，不需要 安装额外的插件。这些验证机制能细粒度地监控 Ranger 中的各种数据流动、用户提交的数 据，并具有数据库的增删改查等功能，提供了各种系统权限监控方式，以便于大数据的生 态系统保持良好的用户权限分离。以下简单列举了几种的 Policy 验证配置方式。

**256** 第二篇 开源实现篇

**1.配置** **Deny Condition 以及** **Exclude 选项**

Ranger 提供丰富的认证方式，但是在简单安装完成并使用的时候，Ranger 会默认关闭 这些功能，因为开发者设计的是能让初步使用Ranger 的用户操作最为简单的界面，但当需 要深入使用时，就要进一步的配置。

Deny Condition以及 Exclude 选项需要对 Ranger Web UI的内部参数进行配置，这个参 数一般存储在系统的/etc/ranger 中，登录之后，也可以访问形如http://ink-1:6080/service/

public/v2/api/servicedef/name/hdfs 的地址，查看目前的配置。也可以通过以下 curl 命令获 取Ranger 各个服务的定义配置项，并存储在本地的 hdfs.json 文件中。

s curl -u admin:admin link-1:6080/service/public/v2/api/servicedef/name/hdfs >hdfs. json

此命令可以通过name/${servicename}$ 来取出配置，例如 Ranger 存储 Hive的配置就 是 link-1:6080/service/public/v2/api/servicedef/name/hive, 而后面的文件名hdfs.json 可以自 主设置。运行命令之后，本地会出现一个配置文件，可以使用Linux 的各种查看文件的命 令查看配置内容。

了解了配置文件的意义之后，如需要对Ranger 的某个服务增加 Deny Condition 以及 Exclude 选项时，可以使用如下命令对文件进行修改：

$vim hdfs.json

#修改以下配置项：

"options":{"enableDenyAndExceptionsInPolicies":"true"}

#通过cur1 命令将HDFS的配置更新到Ranger 服务器中

s curl -iv -u admin:admin -X PUT -H "Accept:application/json"-H "Content-Type: application/json"-d Ehdfs.json linc-1:6080/service/public/v2/api/servicedef/ name/hdfs



Ranger服务的配置可以通过访问本地的/etc/ranger文件进行读取，也可以直接对文 件中的内容进行修改。但经尝试，此方法十分不稳定，不推荐使用。

上述配置中的 enableDenyAndExceptionsInPolicies 指的就是是否开启 Deny Condition 以及 Exclude 选项， 一般默认为false, 所以在没有做任何操作的时候， options 内没有任何 配置，也就没有启用 Deny Condition 以及 Exclude 选项。

启用之后，访问Ranger Web UI,登录成功后访问Resource Based Policie, 单击 HDFS 下的 hadoopdev, 并从列表中单击一个 policy, 如图5-48所示。

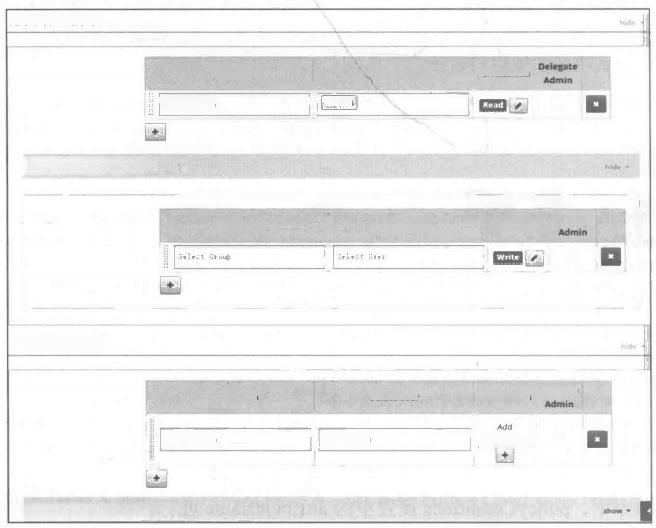
出现 Deny Condition以及 Exclude from Allow/Deny Condition,则说明禁止及排除选项 配置成功。读者可以尝试在上述配置项中加入权限管理，进行测试。

**2.配置** **Geo-based Policy 验证**

Geo-based Policy 能对IP 地址进行筛选，能对各个地址中的用户进行细分，实现细粒 度的权限分离，能控制各个不同的节点权限。

第5章 大数据安全之 Apache Ranger 257

Geo-based Policy 需要创建一个 IP 文件，由于此IP 文件也属于配置中的一项，所以推 荐存放在 Ranger 的配置文件存储地址/etc/ranger 中。读者也可以自行设置，但在之后的配 置中需要更改地址。

**Allow Conditions:**

Select 6roup

Seleet Group

Select User

%hdfs

Permissions

@

Exclude from Alow Conditions:

Select Group

select User

Delegate

Permisslons

口

**Deny Conditions:**

**select Groug**

Select User

Permissions

pelegate

Seleet Group Salect taer ■

Pemhsvom

Exclude from Deny Conditlons:

图5-48 Deny Condition 以及 Exclude 选项设置

$cd /etc/ranger

$mkdir geo

$cd geo

$vim geo.txt

#创建geo文件

IP\_FROM,IP\_TO,COUNTRY\_CODE,COUNTRY\_NAME,REGION,CITY

<10.0.0.255>,<10.0.3.0>,"Us","United States","California","Santa Clara" <20.0.100.80>,<20.0.100.89>,"Us","United States","Colorado","Broomfield"

<20.0.100.110>,<20.0.100.119>,"US","United States","Texas","Irving"

<192.168.237.100>,<192.168.237.110>,"CN","China","Beijing","Local"

geotxt 文件中的IP\_FROM 表示地址区间的起始地址，IP\_TO 表示地址区间的终止地

址 ，COUNTRY\_CODE 表示国家的简写， COUNTRY\_NAME、REGION、CITY 分别表示

国家名称、省份(州)、城市。这些名称都可以进行修改，修改之后，后面的设置需要与此 处的设置同步。

④ 素 Ranger 还提供数字地址的读取，如果需要的话，可以进行设置。

第二篇 开源实现篇

同时请读者将上述代码中最后一行改为自己IP 的区间，由于本书的测试服务器linc-1 的IP 地址为<192.168.237.101>,位于<192.168.237.100>～<192.168.237.110>区间内，所以本书 将测试的IP 地址区间设置为上述区间。

IP 地址映射文件配置完成之后，需要对Ranger系统进行配置，激活 Geo-based Policy 验证功能，如同配置 Deny Condition以及 Exclude选项一样，需要将配置文件从系统中读 取出来，并对其中的配置进行更改。

#通过cur1 命令获取Ranger 对于hdfs 的定义配置项，并存储在本地的hdfs.json 文件中

s curl -u admin:admin zte-1:6080/service/public/v2/api/servicedef/name/hdfs >hdfs.

json

$vim hdfs.json

#修改policyConditions

"policyConditions":[{"itemId":1,"name":"location-outside","label":"Accessedfromoutsi deoflocation?","description":"Accessedfromoutsideoflocation?","evaluator":"org.

apache.ranger.plugin.conditionevaluator.RangerContextAttributeValueNotInCondit

ion","evaluatorOptions":["attributeName":"LOCATION\_COUNTRY\_CODE"}] #修改contextEnrichers

"contextEnrichers":[{"itemId":1,"enricher":"org.apache.ranger.plugin.contextenricher. RangerFileBasedGeolocationProvider","enricherOptions":{"FilePath":"/etc/ranger/ geo/geo.txt","IPInDotFormat":"true"}}]

#通过cur1 命令将HDFS的配置更新到Ranger 服务器中

s curl -iv -u admin:admin -X PUT -H "Accept:application/json"-H "Content-Type: application/json"-d @hdfs.json zte-1:6080/service/public/v2/api/servicedef/ name/hdfs

上述命令中，policyConditions设置中的 attributeName 可以设置为文件中的其他名称， 但是需要在名称前加上LOCATION\_, 例如需要以COUNTRY\_NAME作为配置名，那么需 要将 attributeName 设置为LOCATION\_COUNTRY\_NAME。设置完成之后，在 Ranger Web UI上所有关于Geo 的配置都将基于attributeName。而 contextEnrichers设置中的 FilePath 需要设置为geo 文件的地址，若IPInDotFormat 使用的是IP 地址，则设置为 true; 若使用 的是数字地址，则设置为 false。

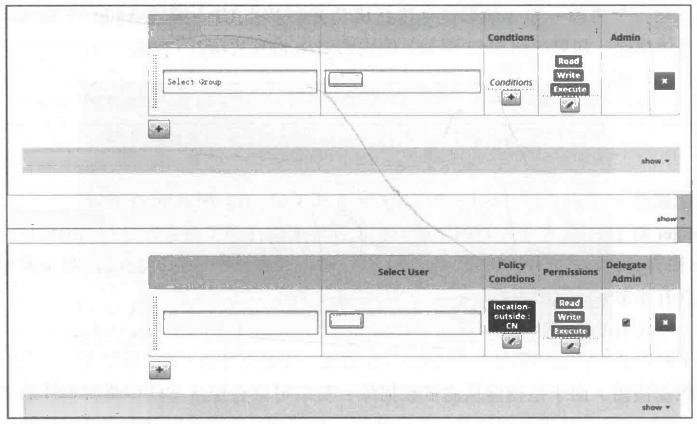
设置完成之后，需要对功能进行认证，本书的认证方式是通过Ranger Web UI对 HDFS 设置禁止本地访问。首先访问Ranger Web UI,登录成功之后，进入 Resource Based Policies列表，单击HDFS下的 hadoopdev 进入HDFS Service 页面，并从页面中的列表里 选择一个 policy。本测试设置的是hdfs 上/hdfs2的访问权限，在 Deny Conditions 设置项中 配置禁止本地访问，认证 Geo-based Policy 是否生效，配置内容如图5-49所示。

配置成功之后，即可测试刚刚设置的 hdfs权限是否生效。在 Master上使用test 用户运 行如下 shell 命令：

s hdfs dfs -put /etc/passwd /test

运行失败后，在Web UI上通过Audit→Access 看到操作信息具体情况，如图5-50所 示。测试结果表明上述命令已被 Ranger 的权限控制机制拒绝。

第5章 大数据安全之Apache Ranger 259



Select Group

Ada

g test

Exclude from Allow Conditlons:

**Deny Conditions:**

**select Group**

Saleet.Qrog

Exclude from Deny Conditlons:

Select user

Permissions

Delegate

Policy

■

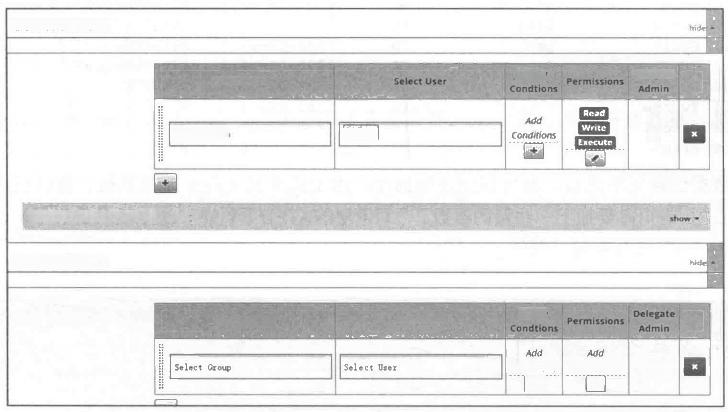
test

图5-49 通过 Ranger Web UI 对HDFS设置禁止本地访问

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 14 | 12/12/201603:19:48 PM | test | hadoopdev | /hdfs2 | EXECUTE | Denied | ranger-ad |
| hdfs | path |

图5-50 本地访问被拒绝

然后删除 Deny Conditions内的权限设置，或者将Policy Conditions设置为US, 测 试 是否还会被 Ranger 拒绝。配置具体信息如图5-51 所示。



Allow Conditions:

Select Group

Smleet Oog test 回

Exclude from Allow Conditions

**Deny Conditions:**

Policy

Select Group

Permksions

+

Select User

Delegate

Conditions



Palicy

■

图5-51 通过Ranger Web UI对 HDFS 设置允许本地访问

260 第二篇 开源实现篇

测试命令与之前一样，如果命令执行成功并在Web UI上通过Audit→Access 看到操

作信息具体情况，如果如图5-52所示，则说明Geo 设置成功。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 14 | 12/12/201603:27:19 PM | test | hadoopdev | /hdfs2/passwd | WRITE | Allowed | ranger-acl |
| hdfs | path |

图5-52 允许本地访问

3.配置基于 Hive 的 Row-Level Filtering 和 Column-Masking 验证

Ranger 对于Hive 有非常好的支持，它比其他的组件多了两个非常有用的验证机制，对 于Hive 而言，这两个验证机制能对Hive 中存储的数据进行细粒度的处理和保密处理，对 敏感数据有非常好的权限控制。

(1)创建 Hive 测试数据库

为了配置并验证这两种验证机制，需要创建一个新的Hive 测试数据库，以免影响其他 部分的数据使用。由于是创建从测试数据库，所以可以直接从文件中将数据读取出来，存 储到 Hive中 ，Hive 具有读取这类文件的功能。创建测试数据并存储在文件中的操作如下：

$vim customer.txt

|  |  |  |
| --- | --- | --- |
| #创建customer 文件，并将下列数据复制到该文件中 |  |  |
| 1 Mackenzy Smith US | 1993-12-18 | 123-456-7890 |
| 2 Sherlyn Miller US | 1975-03-22 | 234-567-8901 |
| 3 Khiana Wilson US | 1989-08-14 | 345-678-9012 |
| 4 Jack Thompson US | 1962-10-28 | 456-789-0123 |
| 5 Audrey Taylor UK | 1985-01-11 | 12-3456-7890 |
| 6 Ruford Walker UK | 1976-05-19 | 23-4567-8901 |
| 7 Marta Lloyd UK | 1981-07-23 | 34-5678-9012 |
| 8 Derick Schneider DE | 1982-04-17 | 12-345-67890 |
| 9 Anna Richter DE | 1995-09-07 | 23-456-78901 |
| 10 Raina Graf DE | 1999-02-06 | 34-567-89012 |
| 11 Felix Lee CA | 1982-04-17 | 321-654-0987 |
| 12 Adam Brown CA | 1995-09-07 | 432-765-1098 |
| 13 Lucas Jones CA | 1999-02-06 | 543-876-2109 |
| 14 Yvonne Dupont FR | 1982-04-17 | 01-23-45-67-89 |
| 15 Pascal Fournier FR  16 Ariel Simon FR | 1995-09-07  1999-02-06 | 23-45-67-89-01  34-56-78-90-12 |

数据创建完毕之后，就可以创建数据库，由于需要从文件中读取数据，所以创建数据 库的 DDL 语句需要加上一些新的参数，“TERMINATED BY \t'” 这一段参数表示上述的数 据是用“\t” 来分隔每一列的。

s hive

#进入Hive

hive>create database cust;

hive>use cust;

hive>create table customer(id int,name\_first string,name\_last string,addr\_country

string,date\_of\_birth date,phone\_num String)ROW FORMAT DELIMITED FIELDS

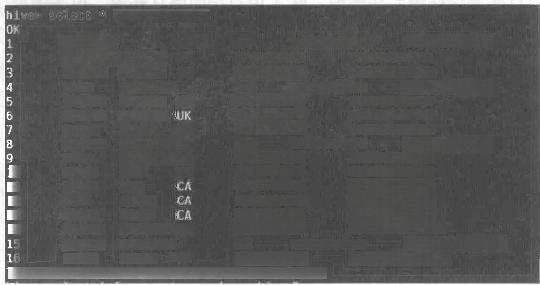
TERMINATED BY '\t'STORED AS TEXTFILE;

hive>load data local inpath 'customer.txt'into table customer;

hive>select \*from customer;

第 5 章 大 数 据 安 全 之 Apa che Ranger 261

上述命令将文件中的数据读取到Hive 中，建表语句需要加上配置项 STORED AS TEXTFILE, 之后的选择语句输出结果如图5-53所示。



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| from customer;  Mac kenzy Smith  Sherlyn Miller US   |  |  | | --- | --- | | Khiana  Jack  Audrey Ruford Narta Derick Anna  Raina  felix  ABT  Lucas Yvonne | Wilson us Thompson  Taylor UK Walker  Lloyd UK Schneider Richter DE Graf DE Lee  Brown  Jones  DupontF FR | | US 1993-12-18 123.456-7890 1975-03-22 234-567-8901  1989-88-14 345-678-9012  Us 1962-10-28 456-789-6123 1985-81-11 12-3456-7890  1976-65-19 23-4567-8981  1981-67-23 34-5678-9812  DE 1982-04-17 12-345-67898   |  |  | | --- | --- | | 1995-89-07 | 23-456-78981 | | 1999-02-06 | 34-567-89812 | | 1982-04-17 | 321-654-0987 | | 1995-69-07 | 432-765-1898 | | 1999-02-06 | 543-876-2189 | | 1982-94-17 | 81-23-45-67-89 | |

Pascal Fourmier FR 1995-09-67 23-45-67-89-01

Ariel Simon FR 1999-02-66 34-56-78-90-12

taken:0.375 seconds,Fetched:16 row(s)

18

11

12

13

14

Time

图5- 53 Hive 输出结果

(2)配置 Row-Level Filtering

Row-Level Filtering 可以基于每一行的数据，让用户只能访问到特定的数据，例如设置 行专门存储用户标识，那么经过设置之后，特定的用户只能访问特定的数据，以防用户访 问未授权的数据。

设置时需要登录 Ranger Web UI, 单击 Access Manager→Resource Based Manager,选 择 Hive 下的 Service Policies, 单击Row level filtering进入页面，单击Add New Policy 创

建新的 Policy, 并进行如下配置：

Hive Database =cust

Hive Table =customer

Select User =hdfs

Row Level Filter =addr\_country='us'

Ranger 还提供更细粒度的设置，可以直接填写SQL 语句，用于筛选数据，甚至可 以通过SQL 语句访问数据表来对数据进行筛选。

配置完成之后，可以对数据进行测试，上述配置的意思是hdfs 用户只能在 customer 表 中访问addr\_country 值为 “US” 的数据，所以在使用select 语句的时候，所有访问到的数

据 的 addr\_country 值 都 为 “US” 。 测 试 结 果 如 图 5 - 5 4 所 示 ， 说 明 配 置 成 功 。

|  |  |  |  |
| --- | --- | --- | --- |
| Macket | Smith | 1993-12-18 | 123-456-7896 |
| Sherlw  h | Wiiler  Wiison | 1975-83-22  1980-88-14 | 234-567-8981 |
|  |  | 1962-16-28 | 456-789-0123 |

**图5-** **54** **测试结果正确**

262 第二篇 开源实现篇

(3)配置 Column-Masking

Column-Masking能基于每一列的数据，对列里面特定字段进行加密，经过设置Masking 之后，特定的用户只能访问加密后的数据，防止用户访问隐私数据，以此来保证数据的安全性。

设置需要登录 Ranger Web UI,单击 Access Manager→Resource Based Manager, 选择 Hive 下的Service Policies,单击Masking 进入页面，单击Add New Policy 创建新的 Policy, 并进行如下配置：

Hive Database =cust

Hive Table =customer

Hive Column =phone\_num

**S**llt MU s Types =Partial mask:show last 4

上述配置的是hdfs 用户访问phone\_num 的时候只能显示后4位尾号。然后再次点击 Add New Policy 创建新的 Policy, 并进行如下配置：

Hive Database =cust

Hive Table =customer

Hive Column =name\_first

Select User =hdfs

Select Mask Mask Types =Custom initcap(reverse({col}))

上述配置的是当 hdfs 用户访问name\_first 时，数据变成逆序。所以在配置完成之后使 用 Select 语句展示出来的结果中，name\_first 中所有数据都变成逆序了，而在 phone\_num 这一行中，前几位数字都不会显示，具体测试结果如图5-55所示。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | rTS | | | |
| Y空m@Kc m  Nylrehs  Ansihk  KCaj | custoner:nam |  | customer.date of | customer.plone |  |
| Smith  Willer  Wi1son  Thospso | 污g  器 | 1993-12 -18  1975-03-22  1989-88-14 | 属-7比  XNX-xxx-8J01  XXX-xXx-4DIF  X口家家- |  |
| selected(0.766 secands | | | | | |

图5-55 配置 Masking结果

**5.4** **Apache** **Ranger** **优化与性能分析**

Ranger 系统配置主要依赖于Web UI的各类配置，以及各个附加功能的使用，而 Web UI中的配置并不仅仅是按照5.2节中配置好即可，还需要对系统有更深入的了解，这样才 能使用更少、更好的配置，来达到需求。系统配置不仅能实现各个功能，还能对系统功能 进行进一步的增强和优化。

首先，5.2节中的系统配置只要验证插件可以使用即可，但是对配置细节并没有做很好 的优化，例如， HDFS 插件配置中只是将HDFS 中的某个特定目录下的某个权限赋予某个 用户，使得这个用户不能使用未被赋予的权限。具体而言，如果只给用户赋予 Write 权限的

第5章大数据安全之 Apache Ranger 263

话，他是无法使用Read 权限的，那么那些没有被特定赋予权限的用户，则可以使用所有权 限的。同样，某个用户如果只是被配置了特定的一个路径的权限，那么这个用户可以随意 访问其他目录下的文件。

在如此配置的情况下，如果使用浏览器访问hdfs, 发现是可以访问的， Ranger Access 结果如图5-56所示。图中浏览器的用户名为dr.who, 并且IP 地址那一栏为空，这就表示对 于这个路径而言，除 test 之外的所有用户都是可以访问的，而且不能追踪是哪个IP 访问了 这个地址，这是一个极大的安全漏洞。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| dr.who | hadoopdev  hdfs | /output path | READ\_EXECUTE | Allowed | ranger-ad |  |  |
| dr.who | hadoopdev  hdfs | /  path | READ\_EXECUTE | Alowed | ranger-acl |  |  |

图5-56 Ranger 中的 Access 列表结果显示

这种在配置没有完全包括所有情况的时候，系统对那些疏漏的用户是不存在权限控制 的，也就是说，新建一个用户，如果没有马上对用户进行赋权，那么这个用户就会是一个 高权限用户，这与权限控制的初衷不符。同时，这种配置方式需要对每一个用户和每一个 路径配置权限，这样做实在过于烦琐。

对于5.2节中大部分的配置而言，赋予权限的过程其实是在对特定用户禁止特殊的权 限。所有类似的配置是非常不适合生产环节的，因为这其实并没有限制新产生的用户的权 限。所以，在生产环境中，对于系统而言，不能采用这种既会产生权限漏洞，又需要大量 的配置的方法来保障系统的权限控制，这是不合适的。

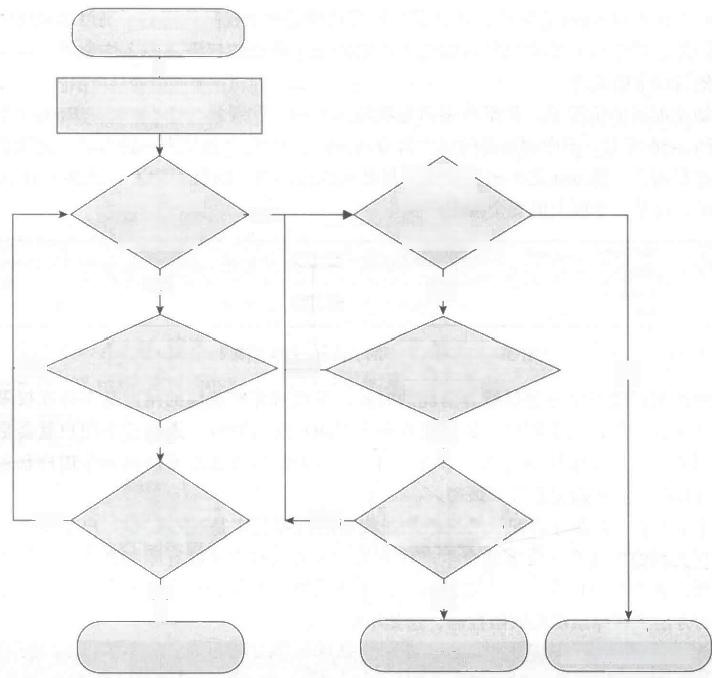
此外，比较常见的配置优化还有，在启动配置 Deny Condition以及 Exclude 选项之后， 系统对于权限的判断，如图5-57所示。从这个流程中可以发现，系统先判断是否有 Deny 选项，再判断Allow, 所以如果需要对系统进行更细致的权限控制的话，最好能对需要拒绝 的用户采用 Deny Condition。而对需要同意的用户使用Allow Condition。在配置不是特别 复杂的情况下，能不使用 Exclude选项就不使用，这在理论上会有一定的性能损耗。当然， 在所需的配置非常复杂的情况下， Exclude 能让配置更清晰，这时就可以使用 Exclude。

同时，在Ranger 中，还有更多的 Policy 验证策略，包括 Geo-based Policy 验证、基于 Hive 的 Row-Level Filtering 和 Column-Masking 验证等，这些验证策略提高了系统的功能， 也能对系统进行进一步的深度权限认证。所以在需要的时候，可以使用这些验证策略，以 使系统具有符合生产环境的更细粒度的验证。

**5.5** **本章小结**

本章主要介绍了Ranger 概述、各个插件及其功能的配置以及系统的配置优化，研究了 Ranger目前的近况以及未来的发展，主要关注Ranger 的基本功能的应用以及当前版本的附 加功能的使用，还对系统的配置优化做出了一些可行的推荐。

**264** **第二篇** **开源实现篇**

Resource access

Request

立

Find policies for the

accessed resource

Has more

Policies with

Deny

Conditions?

Yes

No—

Has more

Policies with

Allow

Conditions?

Yes

-No<

Request matches

a deny condition in

the policy?

Yes

Request matches

<No- an allow condition in

the policy?

Yes

上

No

Yes-

Request matches

a deny exclude?

Yes

\_

Request matches

an allow exclude?

No



DENY

No

ALLOW

Denyl

PASS\_THROUGH

图5-57 Ranger 权限判断流程e

5.1节主要关注于Ranger 的技术概况、发展近况、特点和作用以及应用场景。对 Ranger 项目的创立目的、目前的发展状况以及未来的研究方向进行了叙述，同时对Ranger 的各个方面的特点以及适用的场景进行了分析，并做出了可行的建议。

5.2节主要研究Ranger 的安全认证配置，对Ranger 自身的安装部署以及系统中的各类 安全访问控制等进行了研究，展示Ranger 的整体框架和功能特点，同时对Ranger 支持的 各个组件的配置进行了说明，提供了搭建基于Ranger 的大数据权限管理框架的途径。

5.3节主要对Ranger 目前支持的部分附加功能进行了叙述，主要介绍了与Atlas 集成的 Tag同步功能、Ranger 支持的各类 Policy 功能，以及简单支持的静态加密功能，这些可以 对系统功能进行进一步的拓展。

5.4节主要针对系统的配置优化进行了研究，主要对系统内部的认证机制、系统的各类 配置项等进行了相关说明，并给出了配置系统的建议。

日 来 自<https://cwiki.apache.org/confluence/display/RANGER/Deny-conditions>+and+excludes+in+Ranger+policies。



■■

■ ■

置

意量

|  |  |
| --- | --- |
| 四1■|■1■1  ■|■ ■  ■  ■ | ■■■■ ■■■  ■  ■■重  ■■  ■■■ ■■ ■ ■ |

■

■

■|

■

■

第6章

**大数据安全之** **Apache** **Sentry**

随着 Hadoop 生态系统的广泛部署，会有TB 、PB 甚 至EB 量级的数据存储在 Hadoop 集群中，如此庞大的数量，其安全就显得格外重要。虽然 Hadoop 在文件系统级别有很强的 安全性，但它缺乏更细粒度的数据读取权限控制。这个问题强迫用户做出抉择：保护整个 文件数据或完全不保护。通常采用的后一种方案极大地抑制了Hadoop 中数据的读取。而 Apache Sentry 提供了对存储在 Hadoop 集群上的数据和元数据执行基于细粒度角色授权的 功能。

本章主要向读者介绍Sentry 的技术概况、安装与配置、与现有 Hadoop 系统组件的集 成与使用，以及与 Sentry 相关的场景设计。目的在于让读者对Apache Sentry 有一个完整的 了解，具备将 ApacheSentry 用于生产的能力。

**6.1 ApacheSentry 概述**

Apache Sentry 是一个用于 Hadoop 生态系统的细粒度授权系统，于2012年由Cloudera 公司以 Cloudera Access的名称进行开发，2013年成为 Apache 基金的孵化项目，到2016年 便成为 Apache 基金的顶级项目。

在过去的4年时间里， Apache Sentry 发展迅速，在 Hadoop 数据安全方面的影响令人 瞩目。本节主要介绍 Apache Sentry的发展历程和特点。

**6.1.1** **Apache** **Sentry技术概况**

Apache Sentry 和 Apache Ranger类似，对Hadoop 生态系统上的组件，如 Hive 、Impala 等进行细粒度的数据访问控制，对 Hadoop 集群上经过身份验证的用户和应用程序的数据提

266 第二篇 开源实现篇

或规则，在 Hadoop 中对于文件具有读、写两 级特权，如表6-1所示。在 Sentry 中，不同对 象可以拥有的特权级别略有不同。

7)角色 (Role):Sentry 独有的特征，为 一组特权的集合，或者理解为组合多个访问规 则的模板。

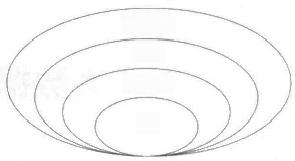
供控制和实施精确的权限控制的功能。在 Hadoop 生态系统现有的组映射环境下，只需通过 对Sentry 独有的角色 (Role) 进行操作，即可轻松实现权限的管理。

和 Hadoop 原生文件权限控制系统相比， Sentry 的功能特性有一些独特之处，下面对一

些关键概念进行分析。

1)对象(Object):Hadoop 原生权限控制在文件层次， 而通过 Sentry 授权规则保护的对象则具有更丰富的层次， 例如服务器、数据库、表等，其层次关系如图6-1所示。

2 ) 验 证 (Authentication): 如 果 Hadoop 的所有配 置都是默认的话， Hadoop 不会验证用户。如果用户发起 一个请求，声称自己是超级用户， Hadoop 都会说“Ok,I believe that”,允许用户做任何超级用户可以做的事。而



服务器

URI

数据库

表

图6-1 对象层次关系

Sentry 依赖底层认证系统，如 Kerberos 或 LDAP, 通过验证凭据可靠地识别用户，这极大 地增强系统安全性。

3)授权(Authorization): 限制用户对给定资源的访问，在Hadoop 中授权的实体为用 户，而 Sentry是基于角色进行访问控制的，授权的实体是角色，这是一种强大的机制，适 用于管理大量用户和数据对象的授权。

4 ) 用 户(User): 在 Hadoop 和Sentry 中均是由系统识别个人的，区别在于上面介绍的 验证方式不同。

5 ) 组 (Group): 由认证系统维护的一组用户，Sentry 使用在 Hadoop 中配置的组映射 机制，以确保 Sentry 看到与 Hadoop 生态系统的其他组件相同的组映射。

6)特权 (Privilege): 允许访问对象的指令 **表6-1** **特权与对象对应关系表**

|  |  |
| --- | --- |
| **特** **权** | **对** **象** |
| INSERT | 数据库，表 |
| SELECT | 数据库，表，列 |
| ALL | 服务器，数据库，表，URI |

8)授权模型 (Authorization models):定义要接受授权规则的对象和允许的操作粒度， Sentry 比 Hadoop 能够实现更细粒度的授权模型，例如，在 SQL 模型中，对象可以是数据 库或表，操作是选择、插入、创建等。对于搜索模型，对象可以是索引、集合和文档，操 作可以为查询、更新等。

通俗地说，借助 Sentry 提供的功能，用户可以在 Hadoop 平台使用更加简洁的方法更 有效地完成繁杂而又琐碎的文件保护工作。假设在一个销售部门中，经理Jack 对业绩表有 读写特权，要求部员Tom 、Selina 有表的读特权，在原有 Hadoop 环境中，将部员和经理设 置在同一组中，将Jack 设置为表的拥有者，即可达到目的。此时，经理的秘书Bob 也需要

第6章 大数据安全之ApacheSentry **267**

有业绩表的读写特权，但是不能拥有经理用户对其他文件的其他特权。在使用了Sentry 的 Hadoop环境中，只需要设置一个对业绩表有读写特权的角色1和只有读特权的角色2,将 经理Jack 和秘书Bob 与角色1连接，将部员与角色2连接，即可达到权限设置的目的。可 以看到，这种权限管理方式在简化权限管理的同时体现出极大的灵活性与高可扩展性，能 够为企业和政府用户实现更结构化、更易扩展的权限管理和文件保护的目标。

为了实现这些功能，与 Ranger 类 似 ，Sentry 也使用了插件-服务器的结构进行权限管 理。Sentry 通过挂钩 (HOOK) 将插件嵌入需要数据保护的单个 Hadoop组件中，拦截操作 请求并进行权限验证。为了实现 Sentry 可扩展的设计目标，它使用结合层 (Binding) 来标 准化授权请求，使得授权验证可以面向多种数据引擎。

经过4年的发展，当前Sentry 支 持Apache Hive 、Hive Metastore/HCatalog 、Solr、 Cloudera Impala 以及 HDFS 中 Hive 的表格数据。而作为一个高度模块化的应用，未来， Sentry 协议会被扩展到更多其他组件。

**6.1.2 ApacheSentry 发展近况**

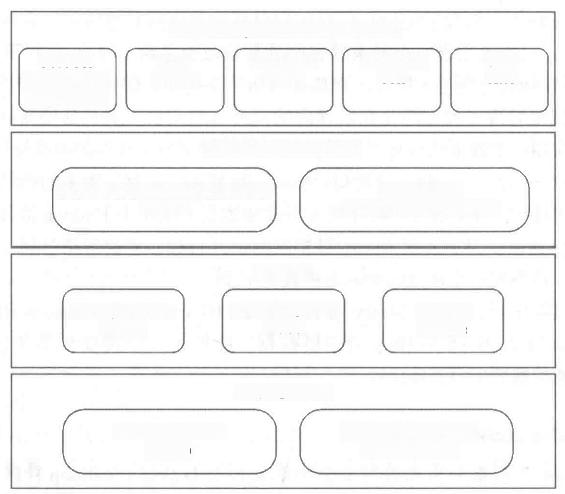
随着 Hadoop 在数据分析和处理平台中的地位日益凸显， Hadoop 社区意识到他们需 要更加健壮的安全控制来解决层出不穷的威胁。与此同时，越来越多的业务场景要求大数 据访问控制的粒度也不再局限在文件级别，而是更加细致地约束文件内部的数据。在这 样的背景下，2008年Cloudera 成为第一个 Hadoop 商业化公司，并在2009年推出第一个 Hadoop发行版，利用HiveServer2 中现有的授权管理模型，扩展并细化了很多细节，完成 了一个相对具有使用价值的授权管理工具Sentry 。HiveServer2 与 Sentry 的授权管理模型对 比如表6-2所示。

**表6-2** **HiveServer2 与Sentry 的授权管理模型对比**

|  |  |  |
| --- | --- | --- |
|  | **HiveServer2** | Sentry |
| 用户/组/角色 | 权限可以被授予给用户，也可以被授予给角色  用户被指定属于一个或多个角色  存在两个默认角色：Public和Admin;可以对  Public角色授权，使得所有用户均具有该权限； 在配置文件中指明属于Admin角色的权限 | 权限只能被授予给角色  角色被指定属于一个或多个组  可以设定一个Sentry管理员所属的组； 所有属于管理员组的用户都具有管理员的  权限 |
| 授权对象 | 数据库/表/视图 | 服务器/数据库/表/视图/列/URL |
| 授权级别 | SELECT,INSERT,UPDATE,DELETE,  ALL | SELECT,NSERT,ALL |

Cloudera 刚推出的RecordService 组件使得 Sentry 在安全竞赛中争得先机。它不仅提供 了跨所有组件一致的安全颗粒度，而且提供了基于Record 的底层抽象，为上层的应用和下 层的存储在解耦合的同时，提供了跨组件的可复用数据模型，使得 Sentry 在 Hadoop 的四 层架构中的资源与数据管理层充当了一个举足轻重的角色，如图6-2所示。

268 **第二篇** **开源实现篇**

PROCESS,ANALYZE&SERVE

BATCH

Spark,Hive, MapReduce

SQL

Impala

STREAM

Spark

SEARCH

Solr

SDK

Kite

UNIFIED SERVICES

RESOURCE MANAGEMENT YARN

SECURITY

Sentry,RecordService

STORE

FILESYSTEM HDFS

RELATIONAL

Kudu

NoSQL

HBase

INTEGRATE

STRUCTURED

Sqoop

UNSTRUCTURED

Flume,Kafka

图6-2 Hadoop 2.0 四层架构θ

说到Sentry, 就不得不提到它一直以来的竞争对手Ranger 。它们各自支持 Hadoop 堆 栈中相应厂商支持的那一部分，但其实这两款产品在很多功能上十分类似，比如都支持细 粒度的访问控制、安全的授权机制，支持多项Hadoop 组件。对比之下，Ranger 更为全面， 所以在业界可能更胜一筹。然而，Cloudera 之前已经宣布了安全方面的重大计划，作为“一 个平台”战略的受益者之一的 Sentry, 其发展前景也是不可估量的。

2013年5月，Cloudera 将 Sentry 贡献给了 Apache 开源社区。Sentry 的出现推动了大 数据技术在更多行业、组织和终端用户的使用。

2013年8月，Sentry 成 为Apache 的孵化项目。在两年半的时间里，开发社区增长迅 猛，很多组织为其贡献代码，涌现了50多个贡献者，其中31个成为 committer。

2013年9月， Sentry 社区发布 Apache Sentry 1.2.0 版本，其中包括修复17个缺陷，完 成5个改进，并确定了15个任务和3个子任务。

2014年5月，Apache Sentry社区宣布发布 Apache Sentry 1.3.0 版本，修复的缺陷和完 成的改进共53个，确定任务3个、子任务12个，更新功能1个。

2014年8月，Apache Sentry社区发布 Apache Sentry 1.4.0 版本。相比前面发布的版本， 此版本中添加的最显著的功能是将安全策略持久存储到数据库中。该功能允许动态更改 Sentry 中的策略，通过Hive 授权 HOOK的相应增强， Sentry 1.4.0孵化与 Hive 的 GRANT 和REVOKE DDL处理集成，可以将策略更改动态捕获到 Sentry 策略中。Sentry 还提供了

O <http://www.raincent.com/content-85-6023-4.html>

第6章 大数据安全之ApacheSentry 269

HOOK Hive Metastore 服务，以防止未经安全策略授权的更改。

2015年7月，Apache Sentry 社区宣布发布 Apache Sentry 1.5.1。重要的工作已纳入这 个版本，其中包括8个主要功能、18个改进和131个缺陷修复。

以下是此版本中的重要功能。

1)列级访问控制：此功能提供对组件(例如 Hive) 列级的访问控制支持。

2)对 DBModel 的更细粒度的权限：此功能添加更多权限类型，如 CREATE 、DROP、 INDEX 、LOCK等。

3)Sentry 服务的高可用性：提供在同一集群中使用备用服务器运行更多冗余 Sentry 实 例的选项。这允许在发生中断的情况下快速故障转移到新的 Sentry 实例。

4)Solr Sentry插件与 DB 存储集成： Solr 可以将安全策略持久化到数据库存储中，就 像 Hive 插件那样。

2015年9月， Sentry 社区宣布发布Apache Sentry 1.6.0。其中修复和完成了115个缺陷 或改进，团队添加了4个新功能。

以下是此版本中的新增功能。

1)Sqoop2 与 Sentry 集成：这是1.6.0中最显著的功能，此新功能允许 Sqoop 授权策略 由 Sentry 管理。

2)为 Solr Sentry 处理程序添加审计日志支持：可以为Solr 活动实现数据管理。

3)支持 CredentialProvider: 管理员和用户可以在外部加密文件中指定密码。

4 ) 以 Sentry 特定格式转储整个内容工具：通过这个转储工具，可以将数据加载到 Sentry 服务的另一个实例中，这对于备份(例如从一个后端存储迁移到另一个后端存储)和 调试底层数据库的内容非常有帮助。

2016年4月， Sentry 成为 Apache 的顶级项目。对于大数据平台上的细粒度授权访问和 其本身来说都是一个重要的里程碑，它推动大数据技术为更多行业、组织和终端用户所使 用，同时为管理员提供灵活、多租户的管理以及统一的平台。

2016年6月，Apache Sentry 1.7.0 正式发布。其中修复缺陷81个，改进29个，新增 功能3个。

Apache Sentry 作为通往 Hadoop 安全的关键一步， 一直对 Hadoop 生态圈的健康发展做 出贡献。但如何促进 Sentry 周边项目的开展，并且能够对 Hadoop 生态圈提供源源不断的 活力，仍需要每一位开发人员和 Sentry 的共同努力。

**6.1.3** **Apache** **Sentry** **技术优势**

由于Hadoop 缺少书面的安全、合规、加密、政策支持和风险管理等协议，其安全性 问题导致使用者往往无法存储非常敏感的数据。这些年来，国外各大厂商纷纷力求通过各 自的核心技术或解决方案做出瞩目的开发成果来弥补 Hadoop 的安全漏洞，例如本书第5章 提到的Hortonworks 公司开发的 Ranger 。而同样作为Apache 项目之一的 Cloudera Sentry,

**270** 第二篇 开源实现篇

因其出色的规范性、安全的授权机制、细粒度的访问控制、基于角色的管理、多租户管理、 统一平台等特点，在市场上脱颖而出。

**1.出色的规范性**

Sentry 在开发过程中考虑到很多国际法案，包括 SOX 、PCI 、HIPAA 、EAL3等，为其 规范性提供了依据。

**2.安全的授权机制**

Cloudera 刚推出的 RecordService 组件在一定程度上使得 Sentry 在安全方面有了保障。 RecordService 不仅提供了跨所有组件一致性的安全颗粒度，而且提供了基于Record 的底层 抽象，为上层的应用和下层存储在解耦合的同时，提供了跨组件的可复用数据模型。

在分布式系统中，Sentry 通常与Kerberos 认证结合使用， Kerberos 认证定义允许哪 些主机连接到服务器上，Sentry 对已通过验证的用户提供数据访问特权。使用Sentry 和 Kerberos 的组合可防止恶意用户通过在不受信任的计算机上创建命名账户来获取服务器上 的敏感数据。管理员也可以通过Sentry 和带选择语句的视图或UDF, 根据需要在文件内屏 蔽数据。

**3.细粒度的访问控制**

由于HDFS 访问控制级别通常基于文件层次，因此对于某个文件的访问，用户往往是 要么可以访问全部内容，要么就任何信息都得不到。此外， HDFS 权限模式不允许多个组对 同一个数据集有不同级别的访问权限。

Apache Sentry 是 Hadoop 生态中负责跨组件统一访问控制的安全解决方案。Record- Service 和 Sentry 等组件结合，提供了跨整个平台的细粒度的统一访问控制策略，消除了 Hive 、HBase 等组件分散而差异的访问粒度控制。服务的对象层次结构涵盖服务器、URL、 数据库、视图、表和列，对其提供不同特权等级的访问控制，包括查询、插入、创建或修 改等。此外，在对象实际存在之前为表或视图指定特权，可以避免敏感信息泄露。例如： 当对一个表使用LOAD DATA 语句时，如果没有足够的权限来执行操作，则错误消息不会 公开对象是否存在。

**4.基于角色的管理**

在 Sentry 中，只能通过给角色授予其相应的权限集来进行权限管理。也就是说，读者 可以规定具有该角色的用户可以访问哪些对象，以及他们可以对这些对象执行哪些操作。还 可以轻易地将访问同一数据集的不同特权级别授予多个组。因此， Sentry 存在一条清晰的映 射关系，权限→角色→用户组→用户。从权限到角色，从角色再到用户组都是通过 grant/ revoke 的 SQL语句来实现的。而从用户组到用户则通过Hadoop自身的用户/组映射自动授予。

和其他 RBAC 系统一样，Sentry 提供了以下功能：

口有层次结构的对象，自动地从上层对象继承权限。

口包含了一组多个对象/权限对的规则。

第6章大数据安全之 ApacheSentry 271

口用户组可以被授予一个或多个角色。

口用户可以被指定到一个或多个用户组中。

**5.多租户管理**

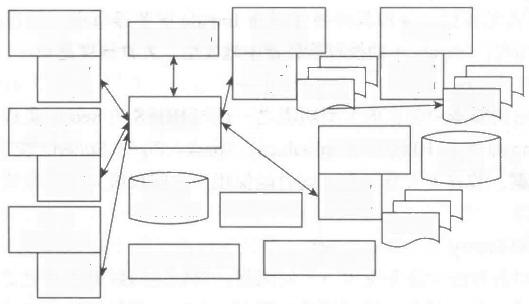
Sentry 可以创建及管理租户基本信息，并为租户分配计算资源和存储资源，为租户分配 相应的权限模型。在 Hive/Impala的情况下，Sentry可以在数据库/schema 级别进行权限管理。

允许多个用户同时共用一个应用程序或运算环境，并且仍可确保各用户间数据的隔 离性。

监控租户资源的使用情况，包括当前CPU 、内存资源的使用，以及历史使用情况、存 储资源的占用及空闲情况、租户下运行作业情况等。

**6.统一平台**

Sentry 为确保数据安全，提供了一个统一平台，使用现有的 Hadoop Kerberos 实现安全 认证。同时， Apache Sentry使用多个 Hadoop组件。在核心有 Sentry 服务器，它存储授权 元数据，并为工具提供安全检索和修改此元数据的API 。如图6-3所示，Sentry 已经可以支 持 Apache Kafka 、Apache Solr 和 Apache Sqoop,通过 Hive 或 Impala 访问数据时可以使用 同样的 Sentry 协议e。



|  |  |  |  |
| --- | --- | --- | --- |
| |  |  | | --- | --- | | Admin App  Sentry | Impala Catalog  Sentry Plugin |   Policy  Metadata | Impala  Sentry  Plugin  Cache  NameNode  HDFS  Sentry  Plugin  Cache |

Metastore

Authentication Group Mapping

Hive

Server2

Apache Solr

Sentry Plugin

Sentry Plugin

Sentry Plugin

Audit Trail

Cache

Hive

图6-3 Sentry 与 Hadoop 生态系统集成②

(1)Hive 和 Sentry

在实际中，授权决策是由在 Hive 或 Impala 的数据处理应用中运行的策略引擎进行的。 Hive加载 Sentry 插件，包括用于处理Sentry 服务的服务客户端和用于验证授权请求的策略 引擎。例如，用户提交以下 Hive 查询：

select \*from production.sales

Hive 会标识用户对sales 表的 SELECT 访问。此时Hive 将会要求 Sentry 插件验证用户的访

e 引用自 <https://ewiki.apache.org/confluence/display/SENTRY/Sentry>+Tutorial。

e <htps://www.cloudera.com/documentation/enterprise/latesttopics/sg>\_sentry\_overview.html

272 第二篇 开源实现篇

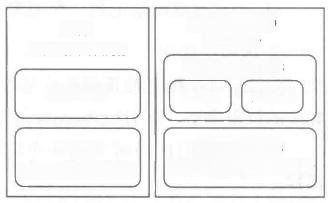
问请求。插件将检索用户与表的相关权限，策略引擎将确定请求是否有效。

接下来，看看 Sentry 是如何保证Hive 安全的。 图6-4显示了不同的身份验证和授权部分如何组合在 一起。

其要点如下：

1)Sentry 要求将 HiveServer2 配置为使用强认 证 。HiveServer2支持 Kerberos以及LDAP身份验证 机制。

2 ) 在 Sentry授权级别，支持两种用户组映射 形式。



|  |  |
| --- | --- |
| HS2  Authentication  Kerberos  LDAP  (OpenLDAP/AD) | Sentry Group  Provider  HadoopGroups  Shell LDAP  LocalGroups  (testing purposes only) |

图6-4 身份验证与授权部分的组合e

3)HadoopGroup 映射使用底层的 Hadoop 组。

4)Hadoop 组又支持基于 Shell 的映射以及LDAP 组映射。

5)LocalGroups 中的用户和组可以使用[users] 部分在策略文件中本地定义(仅用于测 试目的)。

(2)Impala 和 Sentry

Impala 中的授权处理类似于Hive 中的处理，主要区别是缓存特权。Impala的目录服 务器管理缓存模式元数据，并将其传播到所有Impala 服务器节点。此目录服务器还缓存 Sentry元数据。因此，Impala 中的授权验证在本地发生，并且速度更快。

(3)HDFS 和 Sentry

CDH5.3 之后的版本中包含的大型功能之一是与HDFS 的 Sentry集成，使客户能够轻 松地在 Hive 、Impala 和与 HDFS(MapReduce 、Spark 、Pig 和 Sqoop) 交互的其他 Hadoop 组件之间共享数据，节省了大量工作。同时确保用户访问权限只需要设置一次，并且它们 被统一实施。

(4)Search 和 Sentry

Sentry可以对各种搜索任务应用一系列限制，例如访问数据或创建集合。无论用户尝 试完成操作的方式如何，都会一律应用这些限制。例如，限制对集合中的数据的访问会限 制来自命令行、浏览器或通过管理控制台的访问查询。

**6.1.4** **Apache** **Sentry** **架构**

如图6-5所示为 Sentry 和 Hadoop 生态系统集成的示例图。

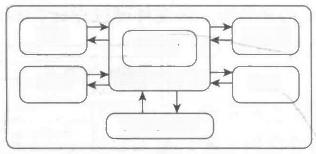
可以看到， Sentry 主要由如图6-6所示的3个组件组成，存在于集成配置中。

Sentry 服务器 (Sentry server):Sentry 服务器管理授权元数据，它支持安全检索和操作

元数据的接口。

e http:/blog.cloudera.com/blog/2013/12/how-to-get-started-with-sentry-in-hive/

第6章 大数据安全之 Apache Sentry 273

Sentry

Slor

Hive

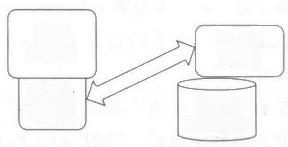
Impala

Hue

Sentry Plugin

NameNode

图6 - 5 Apache Sentry 与 Hadoop 生态系统集成e

Data

Engine

Sentry server

Sentry

Policy Metadata

Plugin

图6-6 Apache Sentry组件e

数据引擎 (Data Engine): 这是需要授权访问数据或元数据的数据处理程序，例如 Hive和Impala。数据引擎加载 Sentry 插件，并且拦截所有访问资源的客户端请求，然后由 Sentry 插件进行验证。

Sentry插件 (Sentry Plugin):Sentry 插件在数据引擎中运行，它提供了操作存储在 Sentry 服务器中的授权元数据的接口，并且包括使用从服务器检索的授权元数据评估访问

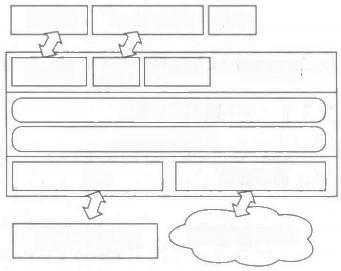
请求的授权策略引擎。

实际上， Sentry 服务器的主要目的只是管理元数 据，真正的授权决策由Sentry 插件中的策略引擎做出。

如图6-7所示，Sentry 架构中包含3个重要的 层次国。

(1)连接层 (Bindings)

如前所述， Sentry 的策略引擎作为 Sentry 插 件 的 一部分，由Hive等数据引擎调用，而连接层模块则 是数据引擎和Sentry 授权之间的桥梁，负责以请求 者原生格式获取授权请求，并将其转换为可由 Sentry 策略引擎处理的授权请求。



lmpala HiveServer2 Solr

Impala Hive Search Bindings

Policy Engine

Policy Provider

Database

Sentry DB Store

Local FS/HDFS

File

图6-7 Apache Sentry 架 构 图

在与数据引擎集成时， Sentry 将 自 己 的HOOK 函数插入各 SQL 引擎的编译、执行不同 阶 段 ，HOOK 函数收集 SQL 语句执行的对象、操作等信息，同时连接层将这些信息转换为 Sentry 授权请求，并将其传递给策略引擎层。

(2)策略引擎层 (Policy Engine)

这是Sentry 授权的核心，策略引擎层从连接层获取请求的特权，并从策略提供者层获 取所需的特权。它比较请求的权限和所需的权限，并决定是否允许操作。

( 3 ) 策 略 提 供 者 层 (Policy Provider)

这是使授权元数据可用于策略引擎层的抽象。这允许元数据的使用与元数据的存储方 式无关。目前 Sentry 支持基于文件的存储和基于关系型数据库的存储。

e <https://www.cloudera.com/documentation/enterprise/latest/topics/sg>\_sentry\_overview.html e <https://www.cloudera.com/documentation/enterprise/latest/topics/sg>\_sentry\_overview.html ③ 引自<https://blogs.apache.org/sentry/>。

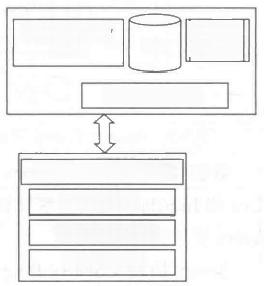
<http://www.open-open.com/news/view/22a9e75>

274 第二篇 开源实现篇

基于文件的方案是将元数据存储在ini 格式的文件中。该文件可以存储在本地文件系统

或者 HDFS 中，文件内包含了组与角色、角色与特权间的 两组映射。

但是文件难以使用编程方式修改，修改过程会存在资 源竞争，不利于维护。同时 Hive 和 Impala需要提供工业 标准的 SQL接口来管理授权策略，要求能够使用编程方 式进行管理。如图6-8所示，在基于关系型数据库存储方 式 中 ，Sentry Policy Store 和 Sentry Service 将角色与特权、 组与角色间的映射持久化到 RDBMS 中e, 例 如 MySQL 、 Postgres 等，并提供创建、查询、更新和删除的编程接 口。这使得 Sentry的客户端可以并行和安全地获取和修改 权限。



Sentry Policy

Store

Sentry Service

Sentry Service Clinet Sentry Core

Sentry Binding

HiveServer2

DB Config

图6-8 Apache Sentry 元数据基于 RDBMS的存储

**6.2 Apache Sentry 的安装与配置**

**6.2.1** **先决条件**

Apache Sentry 最 初 由Cloudera 进行开发，其对应的 Hadoop 生态系统均为CDH 版本， 与本书其他章使用的 Apache Hadoop 生态环境略有不同，建议单独安装环境。

首先，需要有一个由 Cloudera Manager 5.1.x(或更高版本)管理的 Hadoop 环境，在本 书实践过程中，使用Hadoop 2.6.0-cdh5.4.2 版本。由于 Sentry 的架构需求，还需要在环境 里安装 Hive 和 Impala,Hive 版本为 Hive-1.1.0-cdh5.4.2。

需要注意的是，在本节所述安装配置过程中，安装 Hadoop目录为/var/local/hadoop/ hadoop-2.6.0-cdh5.4.2,Hive 安装目录为/var/local/hadoop/hive-1.1.0-cdh5.4.2, 读者可根据 实际集群情况进行更改。

j 注土 本书在先决条件的安装过程中，创建了hadoop 组和 linc 用户，并且linc 用户具有

意

root 权限。

**6.2.2** **Impala** **的安装与调试**

**1.Impala 概况及环境配置**

Impala 是海量并发的查询执行引擎，运行在现存 Hadoop 集群的上百台机器上。与传统 数据库不同，它与底层存储引擎解耦。

日 引自<https://blogs.apache.org/sentry/>。

② <https://yq.aliyun.com/articles/25491>

第6章 大数据安全之 ApacheSentry **275**

Impal 内部的核心组件为： Impala Dacmon 、Impala Statestore 以及 Impala Catalog。其 原理如下。

(1)Impala Daemon

Impala的核心组件是运行在各个节点上的impalad这个守护进程(Impala Daemon), 它 负责读写数据文件，接收从 impala-shell 、Hue 、JDBC 、ODBC 等接口发送的查询语句，并 行化查询语句和分发工作任务到Impala 集群的各个节点上，同时负责将本地计算好的查询 结果发送给协调器节点 (coordinator node)。

可以向运行在任意节点的Impala Daemon提交查询，这个节点将会作为这个查询的协 调器节点 (coordinator node), 其他节点将会传输部分结果集给这个协调器节点，由这个协 调器节点构建最终的结果集。在做实验或者测试时，为了方便，往往连接到同一个 Impala Daemon 来执行查询，但是在生产环境下运行产品级的应用时，应该循环(按顺序)地在不 同节点上提交查询，这样才能使得集群的负载达到均衡。

Impala Daemon不间断地跟 Statestore 进行通信交流，从而让 Statestore 确认健康的、 能接收新的工作任务节点，同时接收 Catalog Daemon 传来的广播消息来更新元数据信息。

(2)Impala Statestore

Impala Statestore 检查集群各个节点上Impala Daemon的健康状态，同时不间断地将结 果反馈给各个 Impala Daemon。这个服务的物理进程名称是 statestored, 在整个集群中仅需 要一个这样的进程即可。如果某个Impala 节点由于硬件错误、软件错误或者其他原因而离 线 ，Statestore 就会通知其他的节点，避免其他节点再向这个离线的节点发送请求。

由于Statestore 只在集群节点有问题的时候起通知作用，所以它对Impala 集群并没有 关键影响。如果 Statestore 没有运行或者运行失败，其他节点和分布式任务会照常运行，只 是当节点掉线的时候集群会变得没那么健壮。当 Statestore 恢复正常运行时，它就又开始与 其他节点通信并进行监控。

(3)Impala Catalog

Imppalla Catalog服务将SQL 语句产出的元数据变化广播给集群的各个节点，Catalog 服务的物理进程名称是 catalogd, 在整个集群中仅需要一个这样的进程。当集群中的任意节 点 create 、alter 、drop 任意对象，或者执行INSERT 、LOAD DATA 的时候会触发广播消息。 由于它的请求会跟 Statestore Daemon交互，所以最好让 statestored 和catalogd 这两个进程 在同一节点上。

本书环境里linc-1(Namenode) 节点安装 Statestore 和 Catalog 服务，而linc-2 、linc-3 (Datanode) 安装Impala 服务。

**2.Impala 下载及安装**

读者需要从<https://archive.cloudera.com/cdh5/redhat/5/x86>\_64/cdh/5.4.2/RPMS/x86\_64/和 <https://archive.cloudera.com/cdh5/redhat/5/x86>\_64/cdh/5.4.2/RPMS/noarch/下 载 RPM软件包， 可以在图形化浏览器界面进行下载，也可以在/var/local/ 目录下执行如下命令进行下载。

**276** **第二篇** **开源实现篇**

**1)Impala 依赖软件包：**

s wget <https://archive.cloudera.com/cdh5/redhat/5/x86>\_64/cdh/5.4.2/RPMS/noarch/ bigtop-utils-0.7.0+cdh5.4.2+0-1.cdh5.4.2.p0.4.e15.noarch.rpm

**2)Impala 基础服务软件包：**

s wget <https://archive.cloudera.com/cdh5/redhat/5/x86>\_64/cdh/5.4.2/RPMS/x86\_64/ impala-2.2.0+cdh5.4.2+0-1.cdh5.4.2.p0.4.el5.x86\_64.rpm

**3)Impala Statestore 软件包：**

s wget <https://archive.cloudera.com/cdh5/redhat/5/x86>\_64/cdh/5.4.2/RPMS/x86\_64/ impala-state-store-2.2.0+cdh5.4.2+0-1.cdh5.4.2.p0.4.el5.x86\_64.rpm

**4)Impala Catalog 软件包：**

s wget <https://archive.cloudera.com/cdh5/redhat/5/x86>\_64/cdh/5.4.2/RPMS/x86\_64/ impala-catalog-2.2.0+cdh5.4.2+0-1.cdh5.4.2.p0.4.e15.x86\_64.rpm

**5)Impala** **server** **软件包：**

s wget <https://archive.cloudera.com/cdh5/redhat/5/x86>\_64/cdh/5.4.2/RPMS/x86\_64/ impala-server-2.2.0+cdh5.4.2+0-1.cdh5.4.2.p0.4.el5.x86\_64.rpm

**6)Impala** **user** **define** **function** **软件包：**

s wget <https://archive.cloudera.com/cdh5/redhat/5/x86>\_64/cdh/5.4.2/RPMS/x86\_64/ impala-udf-devel-2.2.0+cdh5.4.2+0-1.cdh5.4.2.p0.4.e15.x86\_64.rpm

未指明用户时均为在 root 用户下执行命令。

下载完成后，使用rpm 命令进行手动安装。例如 Impala 依赖软件包的安装命令如下： $rpm-ivh bigtop-utils-0.7.0+cdh5.4.2+0-1.cdh5.4.2.p0.4.el5.noarch.rpm

参数-i 表示将要执行的是安装 (install), 而不是-U升级 (Update) 行为。 -v 表示想看 到更多细节，通常与-h 结合使用，系统将打印“#”模拟进度条，显示安装进度。

在上文约定的在linc-1 节点安装 Statestore 和 Catalog 软件包，在 linc-2 、linc-3 节点安 装 Impala server软件包，同时每个节点均安装依赖软件包和基础服务软件包，均使用-ivh 参数。特别地，在安装基础服务软件包时，在包名后添加“--force --nodeps” 参数，使 rpm 包安装时不检查依赖环境，并且强制安装。

**3.Impala 配置**

执行完Impala 安装命令后，系统将会自动创建impala 用户。读者需要将impala 用户 加入 hadoop 用户组中，在所有节点上执行如下命令：

$usermod -a -G Hadoop impala

参数-a 和 -G 合用，将 hadoop 组设置为impala 用户的次要群组，不与主要群组冲突。 修改 impala 用户的密码并设置 root 权限，具体操作步骤如下。

第6章 大数据安全之ApacheSentry **277**

1)输入如下命令：

$passwd impala

然后输入两次密码，为123456。

2)输入命令：

$visudo

在原有内容

root ALL=(ALL)ALL

**下一行添加代码：**

impala ALL=(ALL)ALL

保存文件并退出。

在接下来的文件配置时，需要在每个节点进行相同的操作。建议读者在一个节点上进 行配置操作，然后使用scp 命令，将配置文件或文件夹分发到其他节点的相同位置，以减 少重复操作。

法 意

使 用scp 命令时需要提前开启ssh 服 务 ，Centos 6.5 版本无须单独安装，输入命令 service sshd start 即可启动。

为Impala 依赖设置Java 路径，在/etc/default/bigtop-utils 文件中加入如下语句，保存 并退出，其中%JDK\_HOME%需要替换为读者自己完整的Java JDK安装位置。

s export JAVA\_HOME=8JDK\_HOMES

读者需要将Impala 不同服务安装位置配置在/etc/default/impala 文件中，修改如下部 分，保存并退出。将其中state-store-ip 替换为安装 Statestore 服务节点的 IP 地 址 ，catalog- ip 替换为安装 Catalog 服务节点的 IP 地址。

IMPALA\_CATALOG\_SERVICE\_HOST="catalog-ip"

IMPALA\_STATE\_STORE\_HOST="state-store-ip"

为使Impala 与现有环境统一，用户需要将当前环境中的 Hadoop 、Hive 、hdfs 配置文件 中的参数为Impala 进行配置，具体步骤如下：

1)使用如下命令将原有设置文件core-site.xml 、hdfs-site.xml 、hive-ste.xml 复制到/ etc/impala/conf文件夹下。

S cp /var/local/hadoop/hadoop-2.6.0-cdh5.4.2/etc/hadoop/core-site.xml /var/local/ hadoop/hadoop-2.6.0-cdh5.4.2/etc/hadoop/hdfs-site.xml /var/local/hadoop/hive- 1.1.0-cdh5.4.2/conf/hive-site.xml /etc/impala/conf

2)为了提高性能，需要设置Impala 可以直接从文件系统读取本地数据，这需要启用 短路读取(short-circuit read)。为了对其进行必要的配置，需要在/etc/impala/conf/hdfs-site.

**278** **第二篇** **开源实现篇**

**xml 文件中添加如下内容：**

<property>

<name>dfs.client.read.shortcircuit</name>

<value>true</value>

</property>

<name>dfs.domain.socket.path</name>

<property>

<value>/var/run/hadoop-hdfs/dn</value>

</property>

<property>

<name>dfs.datanode.hdfs-blocks-metadata.enabled</name>

<value>true</value>

</property>

<property>

<name>dfs.client.use.legacy.blockreader.local</name>

<value>false</value>

</property>

<property>

<name>dfs.datanode.data.dir.perm</name>

<value>750</value>

</property>

<property>

<name>dfs.block.local-path-access,user</name>

<value>impala</value>

</property>

<property>

<name>dfs,client.file-block-storage-locations.timeout</name>

<value>30000</value>

</property>

慧 在对配置文件的修改过程中，注意使用半角“/”字符。

读者需要对Impala 中文件软链接进行重新设置，具体步骤如下(选择任一节点均可):

1)执行如下命令，删除/usr/lib/impala/lib目录下原有部分软链接。其中filename.jar 依次替换为avro\*jar,hadoop-\*jar,hive-\*jar,hbase-\*jar,parquet-hadoop-bundle.jar,

sentry-\*.jar,zookeeperjar,libhadoop.so,libhadoop.so.1.0.0,libhdfs.so,libhdfs.so.0.0.0。

s sudo rm -rf /usr/lib/impala/lib/filename.jar

2)需要重新设置软链接或者直接复制文件到目录下，无论哪一种方法，均需要使用 如下命令先下载设置链接的文件，其中filename依次替换为 sentry-1.4.0-cdh5.4.2.tar.gz、 hbase-1.0.0-cdh5.4.2.tar.gz。

s wget <http://archive.cloudera.com/cdh5/cdh/5/filename>

3)下载完毕后使用如下命令解压缩下载的文件，其中filename依次替换为 sentry- 1.4.0-cdh5.4.2.tar.gz 、hbase-1.0.0-cdh5.4.2.tar.gz。

第6章 大数据安全之ApacheSentry 279

s tar -jxvf filename

**解压完成后，使用如下命令将hbase-1.0.0-cdh5.4.2/lib文件夹下一些jar包复制到/var/**

local/目录下。

s cp hbase-1.0.0-cdh5.4.2/lib/hbase-annotations-1.0.0-cdh5.4.2.jar hbase-1.0.0-

cdh5.4.2/lib/hbase-client-1.0.0-cdh5.4.2.jar hbase-1.0.0-cdh5.4.2/lib/hbase-

common-1.0.0-cdh5.4.2.jar hbase-1.0.0-cdh5.4.2/lib/hbase-protocol-1.0.0-

cdh5.4.2.jar /var/local

**同样，将** **apache-sentry-1.4.0-cdh5.4.2-bin/lib文件夹下一些jar包复制到/var/local/目录下。**

$cp apache-sentry-1.4.0-cdh5.4.2-bin/lib/sentry-binding-hive-1.4.0-cdh5.4.2.jar apache-sentry-1.4.0-cdh5.4.2-bin/lib/sentry-core-common-1.4.0-cdh5.4.2.jar

apache-sentry-1.4.0-cdh5.4.2-bin/lib/sentry-core-model-db-1.4.0-cdh5.4.2.jar apache-sentry-1.4.0-cdh5.4.2-bin/lib/sentry-policy-common-1.4.0-cdh5.4.2.jar apache-sentry-1.4.0-cdh5.4.2-bin/lib/sentry-policy-db-1.4.0-cdh5.4.2.jar

apache-sentry-1.4.0-cdh5.4.2-bin/lib/sentry-provider-cache-1.4.0-cdh5.4.2.jar

apache-sentry-1.4.0-cdh5.4.2-bin/lib/sentry-provider-common-1.4.0-cdh5.4.2.jar apache-sentry-1.4.0-cdh5.4.2-bin/lib/sentry-provider-db-1.4.0-cdh5.4.2.jar

apache-sentry-1.4.0-cdh5.4.2-bin/lib/sentry-provider-file-1.4.0-cdh5.4.2.jar /var/local

4)文件下载完毕后，使用如下命令重建软链接，读者也可以采用直接复制相应jar 包 到/usr/lib/impala/lib文件夹下的方法。

|  |  |
| --- | --- |
|  | **本步骤语句中出现的%HADOOP\_HOME%、%Hive\_HOME%,** **均需要分别替换为**  读者实际的 Hadoop 和Hive 的安装路径。 |
| $sudo 1n -s /var/local/hadoop/hadoop-2.6.0-cdh5.4.2/share/hadoop/common/lib/ filename-x.y.z-cdh5.4.2.jar /usr/lib/impala/lib/filename.jar  这条语句中，读者需要将filename-x.y.z-cdh5.4.2.jar依次替换为hadoop-annotations- 2.6.0-cdh5.4.2.jar,hadoop-auth-2.6.0-cdh5.4.2.jar 、hadoop-common-2.6.0-cdh5.4.2.jar、同时 将 filename.jar 依次替换为 hadoop-annotations.jar、hadoop-auth.jar、hadoop-common.jar。 | |
| 清 值得注意的是，创建的软链接名称filename.jar 是对应源文件名filename -x.y.z- cdh5.4.2jar去掉了-x.y.z-cdh5.4.2这部分版本信息，例如hadoop-auth-2.6.0-cdh5.4.2,jar 和对应的 hadoop-auth,jar。在接下来的软链接创建中，将继续使用这一规则，不再 做特殊说明。若读者直接进行复制操作，也需要保持这样的命名规则。 | |

s sudo ln -s /var/local/hadoop/hive-1.1.0-cdh5.4.2/lib/filename-x.y.z-cdh5.4.2.jar /usr/lib/impala/lib/filename.jar

其中 filename-x.y.z-cdh5.4.2.jar 依次替换为avro-1.7.6-cdh5.4.2.jar 、hive-ant-1.1.0- cdh5.4.2.jar,hive-beeline-1.1.0-cdh5.4.2.jar,hive-common-1.1.0-cdh5.4.2.jar 、hive-exec-

1.1.0-cdh5.4.2.jar 、hive-hbase-handler-1.1.0-cdh5.4.2.jar 、hive-metastore-1.1.0-cdh5.4.2.jar、

**280** **第二篇** 开源实现篇

hive-serde-1.1.0-cdh5.4.2.jar 、hive-service-1.1.0-cdh5.4.2.jar 、hive-shims-common-1.1.0- cdh5.4.2.jar、hive-shims-1.1.0-cdh5.4.2.jar、hive-shims-scheduler-1.1.0-cdh5.4.2.jar、parquet- hadoop-bundle-1.5.0-cdh5.4.2.jar 、zookeeper-3.4.5-cdh5.4.2.jar, 同时将 filename.jar 替换为 对应名称。

依次输入如下语句，建立相应软链接。

s sudo 1n -s /var/local/hadoop/hadoop-2.6.0-cdh5.4.2/share/hadoop/tools/lib/ hadoop-aws-2.6.0-cdh5.4.2.jar /usr/lib/impala/lib/hadoop-aws.jar

$sudo ln -s /var/local/hadoop/hadoop-2.6.0-cdh5.4.2/share/hadoop/hdfs/hadoop- hdfs-2.6.0-cdh5.4.2.jar /usr/lib/impala/lib/hadoop-hdfs.jar

在下一条语句中，将filename-x.y.z-cdh5.4.2.jar依次替换为 hadoop-mapreduce-client- common-2.6.0-cdh5.4.2.jar 、hadoop-mapreduce-client-core-2.6.0-cdh5.4.2.jar 、hadoop-

mapreduce-client-jobclient-2.6.0-cdh5.4.2.jar 、hadoop-mapreduce-client-shuffle-2.6.0- edh5.4.2.jar, 并将 filename.jar 替换为对应名称。

$sudo ln -s /var/local/hadoop/hadoop-2.6.0-cdh5.4.2/share/hadoop/mapreduce/ filename-x.y.z-cdh5.4.2.jar /usr/lib/impala/lib/filename-module.jar

在下一条语句中，将filename-x.y.z-cdh5.4.2.jar 依次替换为hadoop-yarn-api-2.6.0- cdh5.4.2.jar 、hadoop-yarn-client-2.6.0-cdh5.4.2.jar 、hadoop-yarn-common-2.6.0-cdh5.4.2.jar、 hadoop-yarn-server-applicationhistoryservice-2.6.0-cdh5.4.2.jar、hadoop-yarn-server-common- 2.6.0-cdh5.4.2.jar 、hadoop-yarn-server-nodemanager-2.6.0-cdh5.4.2.jar 、hadoop-yarn-server- resourcemanager-2.6.0-cdh5.4.2.jar 、hadoop-yarn-server-web-proxy-2.6.0-cdh5.4.2.jar,并将 filename.jar替换为对应名称。

$sudo ln -s /var/local/hadoop/hadoop-2.6.0-cdh5.4.2/share/hadoop/yarn/filename-x. y.z-cdh5.4.2.jar /usr/lib/impala/lib/filename.jar

在下一条语句中，将filename 依次替换为 libhadoop.so 、libhadoop.So.1.0.0 、libhdfs.so、 libhdfs.so.0.0.0。

S sudo ln -s /var/local/hadoop/hadoop-2.6.0-cdh5.4.2/lib/native/filename /usr/lib/ impala/lib/filename

使用如下语句，对/var/local/目录下在第2步中解压出来的jar文件建立软链接，其中 filename-x.y.z-cdh5.4.2.jar 依次替换为在第2步中解压出来的jar 文件名，并将 filename.jar 替换为对应名称。

$sudo ln -s /var/local/filename-x.y.z-cdh5.4.2.jar /usr/lib/impala/lib/filename.jar 5)在其余节点上执行1)～4)步骤，或者使用如下语句将配置好的/usr/lib/impala/lib

文件内容分发到其他节点，其中 linc-IP需要依次替换为读者各节点实际 IP 地址。

$scp /usr/lib/impala/lib/\*linc-IP:/usr/lib/impala/lib

6)为了与前文hdfs-site.xml 文件中dfs.domain.socket.path 对应，使用如下命令，在每

第6章大数据安全之ApacheSentry 281

个节点上/var/run目录下创建名为hadoop-hdfs的文件夹，并将其拥有者设置为hadoop组， impala 用户，文件夹权限属性修改为775。

s mkdir /var/run/hadoop-hdfs

s sudo chown -R impala:hadoop /var/run/hadoop-hdfs

$sudo chmod 775 -R /var/run/hadoop-hdfs

**4.Impala** **启动**

1)在 Namenode 节点使用如下命令启动 Hadoop 和 YARN:

$./var/local/hadoop/hadoop-2.6.0-cdh5.4.2/sbin/start-dfs.sh

$./var/local/hadoop/hadoop-2.6.0-cdh5.4.2/sbin/start-yarn.sh

2)使用如下命令使 Namenode 节点在后台开启 Hive:

$nohup /var/local/hadoop/hive-1.1.0-cdh5.4.2/bin/hive -service metastore&

S nohup /var/local/hadoop/hive-1.1.0-cdh5.4.2/bin/hive -service hiveserver26

3)在安装了 Impala Statestore 的节点上使用如下命令开启服务：

s sudo service impala-state-store start

在安装了Impala Catalog 的节点上使用如下命令开启服务：

s sudo service impala-catalog start

在安装了Impala server的节点上使用如下命令开启服务：

$sudo service impala-server start

读者可以通过执行如下语句，分别查看Impala Server 、Impala State store 和Impala Catelog 是否启动成功。

s sudo service impala-server status

s sudo service impala-state-store status

$sudo service impala-catalog status

同时，在浏览器地址中输入 “http:// 安装 state-store 节点的 IP:25010”和 “http:// 安装 impala-server 节点的 IP:25000”,如果能看到如图6-9所示界面，也说明Impala 启动成功。

Impala的启动日志位于/var/log/impala 中，如果启动失败，可于日志中查看出错报告。

**5.Impala Shell 安装**

Impala Shell 是一个客户端工具，该客户端工具提供一个交互接口，供使用者发起数据 查询或管理任务，比如连接到 impalad。这些查询请求会传给 ODBC 这个标准查询接口。

读者可以选择在新的节点安装 Impala Shell, 或者在已安装 Impala服务的节点上安装， 这样可以省去上面安装 Impala 依赖和基础服务软件包的步骤。

请使用在与上面相同的方法，下载并安装impala-shell-2.2.0+cdh5.4.2+0-1.cdh5.4.2.p0.4.el5. x86\_64.rpm软件包。安装完成后，输入impala-shell 即可启动 Impala Shell, 然后就可以进 行交互了。

282 **第二篇** **开源实现篇**

|  |  |
| --- | --- |
| stalestored /Aogs /memz metrics rpcz fsubscribers Ahreadz Aopics Narz  Version |  |
| statestored version 2.2.0-cdh5.4.2 RELEASE(build b7f0e80e29971632ae1c422243d56c9ef651 |  |
| Built on Tue,19 Nay 201517:83:14 PST |  |
| 1 |  |
| Hardware Info |  |
| Cpu Info: |  |
| Nodel:Intel(R)Core(TH)2 Quad CPU 09580 2.83GHz |  |
| Cores:4 |  |
| L1 Cache:32.00 KB |  |
| L2 Cache:3.00 MB |  |
| L3 Cache:0 |  |
| Hardvare Supports: |  |
| ssse3 |  |
| sse4\_1 |  |
| Physical Nemory:7.50 GB |  |
| Disk Info: |  |
| Num disks 2: |  |
| sda(rotational=true) |  |
| dm-(rotational=true) |  |
| OS Info |  |
| os version:Linux version 2.6.32-431.e16.x86\_64(mockbuildgc6b8.bsys.dev.centos.org) |  |
| 4■ |  |
| Process Info |  |
| Process ID:24927 |  |

图6-9 Impala 启动成功界面

**6.2.3** **Apache** **Sentry的安装和配置**

目前 Sentry 有多种安装方式，例如使用如下 yum命令自动安装，但可能会遇到 “No package sentry available.” 的问题。这里使用手动安装的方法。

s yum install sentry\*-y

**1.下载** **Sentry**

在/var/local 目录下，使用如下命令下载安装包，其中x.y.z 更换为用户实际 Hadoop 的 CDH版本号，本书使用的版本为5.4.2。

s wget <http://archive.cloudera.com/cdh5/cdh/5/sentry-1.4.0-cdhx.y.z.tar.gz>

如果读者想使用高版本 Sentry, 也可以选择国内镜像网站(如北京理工大学镜像网站 mirror.bit.edu.cn/apache/sentry/) 下载高版本进行编译，编译方法不在本书范围之内，请读 者查阅相关资料。

将源码包解压缩后，文件夹内会有README或 BUILD 文件对编译方法进行简介， 可按提示进行尝试。

第6章 大数据安全之ApacheSentry 283

**2.安装Sentry** **和配置**

使用如下语句进行解压缩和安装，将其中x.y.z 更换为读者实际下载安装包的版本号。

s tar -zxvf sentry-1.4.0-cdhx.y.z.tar.gz

$mv sentry-1.4.0-cdhx.y.z sentry-1.4.0

在修改配置文件时，可使用如下语句获取配置文件模板进行修改，也可以直接创建配 置文件。

$cp /var/local/sentry-1.4.0/conf/sentry-site.xml.service.template /var/local/

sentry-1.4.0/conf/sentry-site.xml

$vim /var/local/sentry-1.4.0/conf/sentry-site.xml

1)对 Sentry server一些基本参数进行配置，具体参数及其值如下所示：

<property>

<!一设置Sentry 的管理员用户组，该用户组将拥有在Sentry 创建角色、更改权限的权限--> <name>sentry.service.admin.group</name>

<value>hadoop,root</value>

</property>

<!一设置允许连接Sentry 服务的用户名称，读者根据实际情况修改-->

<property>

<name>sentry.service.allow.connect</name>

<value>impala,hive,hdfs,root</value>

</property>

<property>

<name>sentry.verify.schema.version</name>

<value>true</value>

</property>

<!一设置Sentry Server节点的IP 地址-->

<property>

<name>sentry.service.server.rpc-address</name>

<value> <10.40.20.222></value>

</property>

<property>

<name>sentry.service.server.rpc-port</name>

<value>8038</value>

</property>

**2)如果读者需要使用Kerberos 认证，则还需要配置以下参数：**

<!—如果不使用Kerberos 认证，将此参数值修改为none 即可-->

<name>sentry.service.security.mode</name>

<property>

<value>kerberos</value>

</property>

<!一设置实际使用的Kerberos Principal-->

<property>

<name>sentry.service.server.principal</name>

<value></value>

</property>

第二篇 开源实现篇

<!一设置，keytab 文件的路径-->

<property>

<name>sentry.service.server.keytab</name>

<value></value>

</property>

3)Sentry 1.4.0 及更高版本支持基于文件和基于数据库的存储方式，如果使用基于 SimpleDbProviderBackend的方式，则需要设置如下jdbc 相关的参数。

<!—设置数据库连接地址，读者根据数据库实际使用情况进行修改，最终能够访问Sentry 数据库即可，这里使用 的是MySQL数据库，不同数据库对应不同数据库驱动，如com.mysq1.jdbc.Driver 、org.postgresql.

Driver、org.apache.derby.jdbc.EmbeddedDriver -->

<property>

j/r06/sentry?createDatabaseIfNotExist=true </value>

</property>

<!—设置数据库驱动名称，根据读者实际使用数据库进行修改-->

<property>

<name>sentry.store.jdbc.driver</name>

<value>com.mysql.jdbc.Driver</value>

</property>

<!—设置连接数据库时使用的用户名和密码-->

<property>

<name>sentry.store.jdbc.user</name>

<value>root</value>

</property>

<property>

<name>sentry.store.jdbc.password</name>

<value>root</value>

</property>

4)Sentry store 的组映射 sentry.store.group.mapping 有两种配置方式： org.apache. sentry.provider.common.HadoopGroupMappingService 或者org.apache.sentry.provider.file. LocalGroupMapping, 当使用后者的时候，还需要配置 sentry.store.group.mapping.resource 参数，即设置 Policy file 的路径。

<property>

<name>sentry.store.group.mapping</name>

<value>org.apache.sentry.provider.common.HadoopGroupMappingService</value> </property>

<property>

<name>sentry.store.group.mapping.resource</name>

<value></value>

<description>Policy file for group mapping.Policy file path for local group mapping,when sentry.store.group.mapping is set to LocalGroupMapping Service class.</description>

</property>

5)可以通过设置，启动 Sentry 的 Web 服务器来查看一些报告，具体的设置项如下所示：

<property>

**第6章** 大数据安全之ApacheSentry 285

<name>sentry.service.web.enable </name>

<value>true </value>

</property>

<!—Sentry Web 服务器使用的端口号-->

<property>

<name>sentry.service.web.port</name>

<value>51000</value>

</property>

<!一设置用来报告数据的工具： jmx 或者 console -->

<property>

<name>sentry.service.reporter</name>

<value>jmx</value>

</property>

Cloudera官方强烈要求对 Sentry 的Web 服务器启用 Kerberos 认证以进行安全保护，读 者可根据自己需要决定，具体参数设置如下：

<!一设置 Kerberos 认证保护类型，如果不使用Kerberos 则设置为none -

<property>

<name>sentry.service.web.authentication.type</name>

<value>KERBEROS</value>

</property>

<!一设置实际使用的Kerberos Principal-->

<property>

<name>sentry.service.web.authentication.kerberos.principal</name>

<value></value>

</property>

<!一设置.keytab 文件的路径-->

<property>

<name>sentry.service.web.authentication.kerberos.keytab</name>

<value></value>

</property>

<!—设置允许访问Sentry Web 服务器的用户列表-->

<property>

<name>sentry.service.web.authentication.allow.connect.users</name> <value>hdfs,impala</value>

</property>

6)更改完设置后，保存文件并退出。

正如6.1.4节所述，使用基于数据库的存储方式有诸多便利，这里以 MySQL 数据库为 例，介绍数据库端的配置。如果在上面配置文件设置第3步时，数据库用户为root权限则 可跳过此步骤。

1)进入MySQL后，使用如下语句创建数据库并对用户的授权，其中User、Password 与第3步配置中相同。

mysql>create mysql>CREATE

mysql>GRANT mysql>flush

database sentry;

|  |  |  |  |
| --- | --- | --- | --- |
| USER all | User ON | IDENTIFIED BY'Password;  sentry.\*TO User@'%'IDENTIFIED | BY'Password'; |

privileges;

**286** **第二篇** **开源实现篇**

2)使用如下命令将jdbc 驱动包复制到 Sentry 安装目录下lib 文件夹中，其中 path/to/ mysql-connector-java-x.y.z-bin.jar 替换为读者实际 jar 包路径，其中 x.y.z 为实际版本号，本 书使用的是5.1.18。

$cp /path/to/mysql-connector-java-x.y.z-bin.jar /var/local/sentry-1.4.0/lib

港 可以使用如下命令，在北京理工大学开源镜像网站下载并解压缩所需jar 包，其中 x.y.z 为版本号。

$wget <http://mirror.bit.edu.cn/mysql/Downloads/Connector-J/mysql-connector-java-x>. y.z.tar.gz

s tar -zxvf mysql-connector-java-x.y.z.tar.gz

数据库配置完成后，需要在系统中添加 Sentry 的环境变量。

1)使用如下命令打开系统环境变量文件：

vim /etc/profile

2 ) 在export Path 的上一行添加如下内容，并在其末尾添加“:${SENTRY\_HOME}/bin”。

$export SENTRY\_HOME=/var/local/sentry-1.4.0

到这一步 Sentry 的安装就结束了。在启动 Sentry 服务之前，输入如下语句进行 Sentry 数据库的初始化，将其中的Type 替换为实际使用的数据库，如 MySQL 、Derby 、Postgres。 如果出现后两行提示则表明初始化成功，可以开启 Sentry 服务了。

s sentry --command schema-tool -conffile S(SENTRY\_HOME}/conf/sentry-site.xml -

dbType Type -initSchema

<中间省略部分输出>

Initialization script completed

Sentry schemaTool completed

现在使用如下语句开始 Sentry 服务：

$sentry --command service -conffile ${SENTRY\_HOME}/conf/sentry-site.xml

直接使用如上命令可以启动 Sentry, 但是会发现无法输入命令，命令行被 Sentry 的

素

3

输出“占领”了。读者可以使用如下命令在后台运行 Sentry, 腾出命令行空间继续 做其他操作。

$nohup sentry --command service -conffile S(SENTRY\_HOME}/conf/sentry-site.xml &

**6.2.4** **Apache** **Sentry与** **Impala** **的集成**

Impala 作为 Sentry 的客户端，向其提出授权请求，由Policy Engine判定此操作是否可 以执行。正如6.1.4中提到的， Sentry 的 Bindings 结构使得 Policy Engine 并不需要知晓是 由哪一种组件发来的授权请求，Impala 与 Sentry 集成时只需要通过对Impala 端进行配置，

第6章 大数据安全之ApacheSentry 287

让其能够与 Sentry 组件连通即可，而 Sentry 端则不需要做任何改动。

在root 用户下，执行以下语句，将sentry-site.xml 复制到/etc/impala/conf目录下并打 开进行编辑。

S cp S(SENTRY\_HOME}/conf/sentry-site.xml /etc/impala/conf/

$vim /etc/impala/conf/sentry-site.xml

删除标签<configuration>内原有内容，并添加如下属性。修改完成后，保存并退出。

<!一设置Kerberos 权限认证，这里没用使用Kerberos 认证 - - >

<property>

<name>sentry.service.security.mode</name>

<value>none</value>

</property>

<property>

<name>sentry.service.server.principal</name>

<value></value>

</property>

<property>

<name>sentry.service.server.keytab</name>

<value></value>

</property>

<!一设置Sentry 服务的IP 地址和端口，读者根据实际情况进行修改-->

<property>

<name>sentry.service.client.server.rpc-address</name>

<value> <10.40.20.222></value>

</property>

<property>

<name>sentry.service.client.server.rpc-port</name>

<value>21000</value>

</property>

使用如下命令打开Impala 配置文件并进行编辑。

s vim /etc/default/impala

在 “IMPALA\_CATALOG\_ARGS=” 和 “IMPALA\_SERVER\_ARGS=” 下分别添加属性值 “-sentry\_config=/etc/impala/conf/sentry-site.xml”。需要注意的是，在 “IMPALA\_SERVER\_ ARGS=” 设定服务器名称“-server\_name=ServerName”, 将其中的ServerName更换为 读者设置的服务器名称。如果读者打算采用基于文件存储元数据方式，那么还需要在 “IMPALA\_SERVER\_ARGS=” 代码下添加“-authorization\_policy\_file=/path/to/auth-policy. ini”, 将其中“/path/to/auth-policy.ini” 修改为读者实际元数据存储文件在 hdfs上路径。

修改完成后保存退出。

在相应节点上将Impala Catalog 和Impala server进行重启，即完成Impala 与 Sentry的

集成目标。

s service impala-catalog restart

$service impala-server restart

**288** 第二篇 开源实现篇

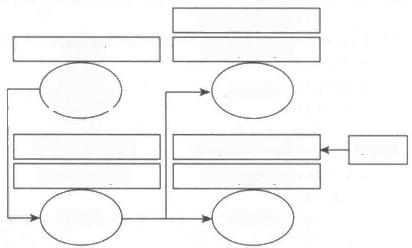
句分发给健康节点，并将节点返回的结果集 合返回给客户端。而Impala Catalog负责将 元数据更新消息在集群中广播。

Impala 与Sentry的集成如图6-10所示。 Impala Catalog 服务不仅广播元数据更新的 消息，而且将 Sentry元数据缓存到本地，并 将其分发到各个Impala Server节点，使得 Impala的验证实际上在本地进行，这样速度

**6.3 Apache Sentry 场景设计之** **Sentry 对** **Impala 的控制**

正如在配置Impala时介绍的，Impala 核心组件为Impala Daemon(Impala Server)、 Impala Statestore 、Impala Catalog。

在Impala使用场景中， Impala Shell 、JDBC客户端、ODBC客户端、Hue 均可作为 客户端，连接到集群中任一Impala Server上，它们向其提交操作语句，该节点作为协调器 (coordinate node), 向 Impala Statestore 节点确认健康节点的信息，将客户端提交的操作语



Impala Catalog Sentry Plugin

Cache

Impala Catalog

Sentry Plugin

Cache

Sentry

Policy

Metadata,

Impala Catalog

Sentry Plugin

Cache

Client

更快。 图6 - 10 Impala 与 Sentry 集成

**6.3.1** **场景数据准备**

使用如下命令创建一个 CSV 文件并进行编辑，作为场景验证时的外部数据使用。 s vim tabl.csv

在 tab1.csv 中输入如下内容后，保存并退出。

2,false,1243.5,2012-10-2513:40:00

1,true,123.123,2012-10-2408:55:00

3,false,24453.325,2008-08-2209:33:21.123

4,false,243423.325,2007-05-1222:32:21.33454

5,true,243.325,1953-04-2209:11:33

由于被Impala 加载的数据文件必须在HDFS文件系统之上，这里需要使用如下命令将 tabl.csv 文件上传至/user/hdfs/datafile/tab1路径下，将其中/path/to/tab1.csv 替换为读者实 际文件路径。

s hdfs dfs -mkdir -p /user/hdfs/datafile/tabl

$hdfs dfs -put /path/to/tabl.csv /user/hdfs/datafile/tabl

Impala 查询或读取外部数据文件时，并不能指定一个文件，而是在指定文件夹内查询 或读取所有文件，所以这里为 tab1.csv单独创建一个文件夹tab1。

**6.3.2** **基于文件存储元数据的场景验证**

本节将采用基于文件的方式进行场景验证 ， 分别测试服务器 、 数据库 、 表等不同层级

第6章 大数据安全之ApacheSentry 289

以及外部数据文件的权限管理。

**1.权限设定**

创建如下auth-policy.ini文件，将文件上传至HDFS相应位置，即在$Hive\_HOME/ conf/hive-site.xml文件中 “hive.metastore.warehouse.dir” 项所设定的位置，并在Impala配 置文件中添加相应配置，具体配置项请参看6.2.3节。

[groups]

#将组与角色连接

root =entire server

#root 组用户在服务器拥有所有权限，也就是管理员用户

hadoop =one\_database,staging\_dir

#hadoop 组用户拥有在test\_db 数据库所有权限以及读取HDFS上数据文件权限

test=read\_all\_tables,one\_database

#test 用户组拥有读取服务器上所有表的权限和test\_db 数据库所有权限

[roles]

#角色创建

read\_all\_tables

=server=server1->db=\*->table=\*->action=SELECT

l在l\_所t有serserverl->db=\*->table=\*

#在所有表拥有所有权限

all databases =server=serverl->db=\*

#在所有数据库拥有所有权限

one\_database =server=serverl->db=test\_db

t在aisnt\_gd\_bdi据库拥有所有rver=serverl->uri=hdfs://linc- 1:9000/user/hdfs/metatable/tabl-

>action=\*

#读取外部数据文件夹权限，其中“hdfs://linc-1:9000” 部分应与hadoop 配置文件core-site.xml 中

eni.d\_seFs”rverl

#在整个服务器拥有所有权限

在某一层次拥有所有权限只意味着对所有其子层次上的表具有增删改等权限，但在 所有数据库拥有所有权限并不能进行创建数据库的操作。

首先在root 用户下，输入如下命令进入Impala Shell 客户端，通常启动成功后会显示 Impala 一些基本信息，如下所示：

$impala-shell

Starting Impala Shell without Kerberos authentication

Connected to linc-2:21000

Server version:impalad version 2.2.0-cdh5.4.2 RELEASE(build b7f0e80e29971632ae 1c422243d56c9ef65b8c5b)

Welcome to the Impala shell.Press TAB twice to see a list of available commands.

Copyright(c)2012 Cloudera,Inc.All rights reserved.

(Shell build version:Impala Shell v2.2.0-cdh5.4.2(b7f0e80)built on Tue May 19 16:45:28 PDT 2015)

290 **第二篇** **开源实现篇**

[linc-2:21000]>

有时启动并不会自动连接某个 Impala Server节点，需要使用如下命令手动进行连接， 将其中 linc-2 替换为实际要连接的Impala Server节点名。

[Not connected] >connect linc-2;

2.服务器层次权限验证

首先输入如下语句创建数据库 test\_db, 如果返回操作时间则操作成功。

[linc-2:21000]>create database test\_db;

Query:create database test\_db

Fetched 0 row(s)in 0.13s

**继续创建一个表default.test\_table**。

[linc-2:21000]>create table default.test\_table(x int);

Query:create table default.test\_table(x int)

Fetched 0 row(s) in 0.12s

其 中“default.test\_table(x int)” 表明 test\_table 表存在 default 数据库中，也可以先使 用 “use default;” 命令，再使用“create table test\_db(x int);” 达到相同的效果。

此时输入 “exit;” 退出当前Impala Shell,并使用hadoop 组任一用户，这里使用的是 hdfs 用户，重新启动Impala Shell。首先看看当前服务器内都有哪些数据库，并尝试创建数 据库 test2。

[linc-2:21000]>show databases;

Query:show databases

+---------+

I name

+ +

default l

test\_db l

+--------- -十

Fetched 2 row(s)in 0.12s

[linc-2:21000]>create database test2;

Query:create database test2

ERROR:AuthorizationException:User 'hdfs'does not have privileges to execute

'CREATE'on: test2

可以看到，hadoop 用户组只拥有在 test\_db 数据库的所有权限，所以不能像 root 用户 那样在服务器层次进行增加数据库的操作。

**3.数据库各个层次权限验证**

接下来，测试数据库各个层次的权限管理，这里分别尝试在test\_db 和 default 数据库进 行表相关操作。

[linc-2:21000]>create table default.test(x int);

第6章大数据安全之ApacheSentry 291

Query:create table default.test(x int)

ERROR:AuthorizationException:User 'hdfs'does not have privileges to execute 'CREATE' on:default.test

[linc-2:21000] >show tables in default;

Query:show tables in default

ERROR:AuthorizationException:User 'hdfs'does not have privileges to access: default.\*

[linc-2:21000]>create table test\_db.test(x int);

Query:create table test\_db.test(x int)

Fetched 0 row(s)in 0.31s

[linc-2:21000]>show tables in test\_db;

Query:show tables in test\_db

+ +

name

+

l test

I test\_table l

+ +

Fetched 2 row(s)in 0.01s

[linc-2:21000]>drop table test\_db.test;

Query:drop table test\_db.test

可以看到，使用hdfs 用户在 default 数据库中创建表和查询表时，均由于没有权限而被 拒绝，而在 test\_db数据库中成功创建 test 表，并进行显示和删除。

**4.外部数据文件权限验证**

为了验证外部数据文件的权限，此时切换到 test用户组用户，这里使用的是 test 用户， 重新进入Impala Shell,对外部数据文件使用的权限进行验证。

[linc-2:21000]>create external table test\_db.tabl

>(

>id int,

>col\_ 1 boolean,

>col\_2 double,

>col\_3 timestamp

row format delimited fields terminated by ','

>location '/user/hdfs/metatable/tabl';

< - 省略部分提示语句 - >

ERROR:AuthorizationException:User 'test'does not have privileges to access: hdfs://linc-1:9000/user/hdfs/metatable/tabl

由于 test 用户组未拥有外部数据文件的权限，无法读取文件内数据而导致操作失败。 此时切换到 hadoop组任意用户，重新进入Impala Shell,提交相同命令进行测试，并查看 表中数据，可以看到已经成功创建表tab1, 并将数据文件内数据导入表中。

[linc-2:21000]>create external table test\_db.tabl

< - 省略部分语句与基础提示信息 - >

WARNINGS:Impala does not have READ\_WRITE access to path 'hdfs://linc-2:9000/

**292** **第二篇** **开源实现篇**

user/hdfs/metatable'

Fetched 0 row(s)in 0.10s

[linc-2:21000]>select \*from test\_db.tabl;

Query:select \*from test\_db.tab1

- -- - l\_1 l col\_2 l \_3

+ ·

I 1 I true 123.123 2012-10-24 08:55:00

I 2 false 1243.5  2012-10-2513:40:00

|3 l false 24453.325 2008-08-2209:33:21.123000000

4 false 243423.325 2007-05-1222:32:21.334540000

5 true 243.325 1953-04-2209:11:33

Fetched 5 row(s)in 3.67s

**5.权限变更测试**

基于文件储存元数据有很多不便之处，例如无法使用编程接口，可能存在由于多用 户同时修改而造成资源竞争等不易维护的问题，而且在权限变更时，由于需要由Impala Catalog 将变更分发到各Impala Server, 会造成一定延迟。

测试变更权限，首先对本地auth\_policy.ini 文件进行修改，将 “test=read\_all\_tables,

one\_database”修改为“test=read\_all\_tables”,保存退出后，先删除 HDFS 上保存的 auth\_ policy.ini,再将修改后的文件上传至HDFS。

使用test 用户启动Impala Shell, 并使用如下语句尝试在 test\_db 数据库创建一张表，

可以看到权限文件的修改并没有对 test 用户的权限起到作用。

[linc-2:21000]>create table test\_db.test3(x boolean);

Query:create table test\_db.test3(x boolean)

Fetched 0 row(s)in 0.12s

为了让修改后的文件立即生效，需要退出Impala Shell并重启 Impala Server,或者稍等 几分钟， Impala Catalog会自动广播元数据的改变， Impala Server将对本地缓存进行同步， 此时若尝试删除刚创建的表，test 用户组便会因为权限不足被拒绝删除操作如下所示：

[linc-2:21000]>drop table test\_db.test3;

Query:drop table test\_db.test3

ERROR:AuthorizationException:User 'test'does not have privileges to execute

'DROP'on:test\_db.test3

**6.3.3** **基于数据库存储元数据的场景验证**

相比基于文件存储元数据，基于数据库存储有不存在资源竞争、便于维护、提供编程 接口等诸多优点，本节主要对这些优点进行场景验证。

开始本节实验前，不要忘记将元数据储存方式修改为基于数据库的。

第6章 大数据安全之ApacheSentry 293

**1.权限设定**

需要使用管理员角色对权限进行控制，即使用配置文件sentry-site.xml 中 sentry. service.admin.group 设定用户组启动Impala Shell 才能进行角色创建等管理。使用如下命令 创建与文件管理时相同的角色与组对应关系。

创建角色 entire\_server 、staging\_dir 、one\_database 、read\_all\_tables:

[linc-1:21000]>create role entire\_server;

Query:create role entire\_server

Fetched 0 row(s)in 0.15

#因篇幅限制，以下将省略此基础输出信息

[linc-1:21000]>create [linc-1:21000]>create [linc-1:21000]>create

role staging\_dir;

role one\_database;

role read\_all\_tables;

分别赋予 entire\_server在服务器上的所有权限， one\_database 角色在数据库 test\_db上 的所有权限，staging\_dir 对外部数据文件操作权限， read\_all\_tables角色在所有数据库上的 读取的权限。

[linc-1:21000]>grant all on server to role entire\_server;

[linc-1:21000]>grant all on database test\_db to role one\_database;

[linc-1:21000]>grant all on URI "hdfs://linc-2:9000/user/hdfs/metatable/tabl"

to role staging\_dir;

[linc-1:21000]>grant select on database default to role read\_all\_tables; [linc-1:21000]>grant select on database test\_db to role read\_all\_tables;

将 entire\_server与 root组连接、角色one\_database 与组 hadoop 和组 test 连接、角色 staging\_dir 与组 hadoop连接、角色read\_all\_tables 与组 test 连接，即root组成员拥有在服务 器上的所有权限，hadoop组成员拥有test\_db数据库所有权限和读取外部数据文件的权限， test 组成员拥有 test\_db数据库所有权限和读取所有表的权限。

[linc-1:21000]>grant role entire\_server to group root;

[linc-1:21000]>grant role one\_database to group hadoop;

[linc-1:21000]>grant role one\_database to group test;

[linc-1:21000]>grant role staging\_dir to group hadoop;

[linc-1:21000]>grant role read\_all\_tables to group test;

此时各个组已经与相应角色连接，拥有相应权限，可分别在hadoop组、test组、root 组输入如下语句， root 组成员可以查看所有角色信息，而其他组则会返回如下所示拒绝访问 信息。

[linc-1:21000] >show roles;

Query:show roles

ERROR:

AuthorizationException:User 'test'does not have privileges to access the requested policy metadata or Sentry Service is unavailable.

294 **第二篇** **开源实现篇**

如果用户确实有权限，就需要检查 Sentry 服务有没有启动或者发生意外停止。

**2.Impala Shell 客户端测试和验证**

对于服务器和数据库各个层次测试与验证以及对外部数据文件的测试验证，操作过程 与步骤均与6.3.2节相同，请读者参照实现。

**3.编程接口进行测试验证**

为了直接使用编程接口而不安装或者进行其他配置环境，这里直接采用编写Shell 脚本

的方法进行验证。

1)使用如下语句创建 Shell 脚本并添加执行权限。

$touch test.sh

$chmod +x test.sh

2)对 Shell 脚本内容进行编辑，使用如下命令打开test.sh。

$vim test.sh

3)让这个脚本自动执行一条 select语句。

#!/bin/bash

#Program:

# 执行一条select 语句

#使用hdfs 用户打开Impala Shel1

impala-shell -u hdfs <<EOF

select \*from test\_db.tabl;

EOF

**4)编辑好后保存退出，并使用如下语句执行这个脚本。**

s test.sh

Starting Impala Shell without Kerberos authentication

<省略部分输出信息>

Query:select \*from test\_db.tabl

[linc-1:21000]>

-  - - l-\_- 1 l col\_2 l col\_3

+----+--

1 true 123.123 2012-10-2408:55:00

12 l false 1 1243.5 2012-10-2513:40:00

13 l false 24453.325 2008-08-2209:33:21.123000000

14 l false l 243423.325 2007-05-1222:32:21.334540000

15 |true 243.325 1953-04-2209:11:33

+----+-------+----

WARNINGS: Unknown disk id. This will negatively affect performance.Check your

hdfs settings to enable block location metadata.

Fetched 5 row(s)in 0.29s

Goodbye hdfs

5)test.sh 脚本成功地执行了一条 select 语句，接下来让这个脚本完成更复杂一点的权

第6章大数据安全之 ApacheSentry 295

**限控制的任务，比如执行创建角色的任务。文件修改如下：**

#!/bin/bash

#Program:

#创建一个新角色，拥有在default数据库所有权限

#使用root用户打开Impala Shell

impala-shell -u root <<EOF

create role def\_database;

grant all on database default to role def\_database;

EOF

6)修改完成后保存并退出，执行脚本并观察输出。

$test.sh

Starting Impala Shell without Kerberos authentication

<省略部分输出信息>

Query:create role def\_database

[linc-1:21000]>

Fetched 0 row(s)in 0.28s

Query:grant all on database default to role def\_database

[linc-1:21000]>

Fetched 0 row(s)in 0.12s

Goodbye root

7)通过脚本成功创建了一个 def\_database 角色，最后通过交互操作将这个角色与组连 接起来。

#!/bin/bash

#Program:

#创建一个新角色，拥有在default数据库所有权限

read -p "Please input your first name:"NAME # 提示使用者输入

#使用root用户打开Impala Shell

impala-shell -u root <<EOF

#将角色与组连接

grant role def\_database to group SNAME;

EOF

8)修改完成后保存并退出，执行脚本，在提示“Please input your first name:”后输入

读者想连接的组名，并观察输出。

$test.sh

Please input your first name:test

Starting Impala Shell without Kerberos authentication

<省略部分输出信息>

Query:grant role def\_database to group test

[linc-1:21000]>

Fetched 0 row(s)in 0.08s

Goodbye root

可以观察到，通过这个脚本，实现了对数据库的操作，对角色的创建，并通过交互输 入完成了角色和组的连接工作，这些在基于文件存储元数据模式下都是难以实现的。

**296** 第二篇 开源实现篇

**6.4 Apache Sentry 场景设计之** **Sentry 对** **Hive 的控制**

**6.4.1** **Hive与** **Sentry 的集成配置**

在前面几节的学习中可以了解到，Sentry 是通过使用策略提供程序来定义对Hive 的访 问控制的。策略提供程序不仅可以支持基于文件的数据，也可以支持数据库，给用户带来 多样化的选择。在这一节我们首先给大家介绍基于文件的集成配置，关于基于数据库的集 成配置请读者参考6.4.4节。

|  |  |
| --- | --- |
| 素 | 在进行本节的配置之前，建议读者使用的CDH 为4.3.0或更高版本，Hive为1.1.0 或以上版本。 |
| **1.创建软链接**  拷贝Sentry 相关jar 包到 Hive 的 lib 目录内，使 Hive 添加对 Sentry 的依赖。  $cp /usr/lib/sentry/lib/sentry\*.jarSHive\_HOME/lib/  $cp /usr/lib/sentry/lib/shiro-core-\*.jarSHive\_HOME/lib | |
| 素 这里的 Hive-HOME 是环境变量，指向 Hive 根目录。 | |

**2.修改** **hive-site.xml 配置文件**

在linc-2节点上，进入Hive安装目录下的 conf目录，对hive-site.xml 配置文件进行修改。 修改内容如下：

1)Hive Metastore 集成 Sentry。

<property>

<name>hive.metastore.pre.event.listeners</name>

<value>org.apache.sentry.binding.metastore.MetastoreAuthzBinding</value> </property>

<property>

<name>hive.metastore.event.listeners</name>

<value>org.apache.sentry.binding.metastore.SentryMetastorePostEventListener</ value>

</property>

2)禁止 HiveServer2 impersonation。将 hive.server2.enable.impersonation 设置为 false, 表明是运行 HiveServer的 user 执行语句。

<property>

<name>hive.server2.enable.impersonation</name>

<value>false</value>

</property>

3)HiveServer2 集成 Sentry。

<property>

**第6章** **大数据安全之** **ApacheSentry 297**

<name>hive.server2.session.hook</name>

<value>org.apache.sentry.binding.hive.HiveAuthzBindingSessionHook</value> </property>

<property>

<name>hive.security.authorization.task.factory</name>

<value>org.apache.sentry.binding.hive.SentryHiveAuthorizationTaskFactoryImpl</value> </property>

4)Hive 与 Sentry集成。这里需要指明 sentry-site.xml 路径，默认文件在 conf目录下。 该文件需要用户自行创建，在后面会介绍。

<property>

<name>hive.sentry.conf.url</name>

<value>file:///var/local/hadoop/hive-1.1.0-cdh5.4.2/conf/sentry-site.xml</value> </property>

**3.创建sentry-site.xml 配置文件**

在 linc-2节点上，进入Hive 安装目录下的conf 目录，读者需要创建一个 sentry-site. xml 配置文件。

部分配置内容如下：

<prope>sentry.service.client.server.rpc-port</name>

<value>8038</value>

</property>

<prope>sentry.service.client.server.rpc-address</name>

<value>linc-2</value>

</property>

<property>

<name>sentry.service.client.server.rpc-connection-timeout</name> <value>200000</value>

</property>

<propeme>sentry.metastore.service.users</name>

<value>hive</value>

</property>

<property>

<name>sentry.hive.provider.resource</name>

<value>/var/local/hadoop/hive-1.1.0-cdh5.4.2/warehouse/sentry-provider. ini</value>

</property>

属性sentry.hive.provider.resource 是用来定义全局策略文件的路径。

**4.修改** **Hive 仓库目录权限**

Hive 仓库目录和所有子目录的权限必须为770。并且所有文件和目录(/user/hive/ warehouse或读者在 Hive配置中指定为hive.metastore.warehouse.dir 的路径)都应由 hive 用

**298** **第二篇** **开源实现篇**

户和组拥有。

修改/user/hive/warehouse 权限：

$hdfs dfs -chmod -R 770 /user/hive/warehouse

s hdfs dfs -chown -R hive:hive /user/hive/warehouse

**5.启动** **Hive**

使用hive 用户，输入启动命令如下：

s nohup hive --service metastore &

s nohup hive --service hiveserver26

**6.4.2** **准备实验数据**

**1.创建animals.csv 文件**

准备测试数据，创建 animals.csv文件，方便后续直接导入数据库。在根目录下，输入 如下命令：

$cat /tmp/animals.csv

1.2,dog,bark,bone

2.3,cat,meow,fish

3.4,rabbit,drum,carrot

4.5,monkey,chatter,banana

5.6,tiger,roar,all

**2.创建数据库**

这里引用上面提到的csv 文件，创建一个名为zoo 的数据库，并给它建立一个由ip、 specie 、sound 、food 四个字段组成的数据表zoo.animals。创建名为 garden 的数据库，直接 从 zoo.animals数据表中选择3个字段建立新的表 garden.animals,并从该表中选择数据项 为 specie='cat', 建立表garden.animal\_catonly。

在 Hive 中运行下面 SQL语句：

create database zoo;

create table zoo.animals (

ip STRING,specie STRING,sound STRING,food STRING

)ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';

load data local inpath '/tmp/animals.csv'overwrite into table zoo.animals;

create database garden;

create view garden.animals\_catonly as select \*from garden.animals where specie = 'cat';

create view garden.animals as select specie,sound,food from zoo.animals;

**6.4.3** **基于文件存储方式的数据表操作**

Sentry 目前可以支持基于文件的策略提供程序。在这种方式下，用户使用 ini 格式的文 件保存元数据信息。

第6章 大数据安全之 ApacheSentry 299

注意

**1.创建角色和组**

创建 sentry-provider.ini 文件并将其上传到hdfs 的/user/hive/sentry/ 目录。其中， groups 选项匹配组和角色， roles 选项匹配角色和权限。

创建内容如下：

rtire\_server

test =select\_garden

[roles]

rcet\_\_n =serv-t->\*a-=SELECT

在上面创建了两个角色：

口 口

entire\_server, 具有管理员权限，可以读写所有数据库，并授权给 root 用户和组。 select\_garden, 只能读 garden 数据库，这里把它授权给 test 用户和组。

root 和 test 用户是笔者在节点上创建的用户。如果系统上没有 test 用户和组，需要 读者手动创建。

**2.entire\_server** **角色对数据表操作**

(1)beeline 访问 HiveServer2

beeline 是Hive 0.11引入的新的交互式CLI, 它基于SQLLine, 可以作为Hive JDBC Client 端访问HiveServer2, 启动一个beeline 就是维护了一个 session。

beeline 使用JDBC 连接到远程HiveServer2 实例。因此，连接参数是在基于JDBC 的 客户端中常见的JDBC URL:>beeline-u <url>-n <username>-p <password>。这里要测试 的是 entire\_server 角色，所以username 是该角色下的root 用户，密码是123456。

使用 root 用户，访问 beeline:

S beeline -u"jdbc:hive2://linc-2:10000"-n root -p 123456

HiveServer2 启动时默认的地址是localhost:10000,linc-2 是实验主机名称。

(2)访问权限

当beeline 成功连接 HiveServer2 后，等待输入命令的界面如下：

0:jdbc:hive2://linc-2:10000>

查看所有数据库，得到结果如下：

0:jdbc:hive2://linc-2:10000>show databases;

+- · -+--+

I database\_name

|zoo

300 第二篇 开源实现篇

garden

default

test\_db

+-- ·

4 rows selected(0.346 seconds)

选择 zoo 数据库，得到结果如下：

0:jdbc:hive2://linc-2:10000>use zoo;

No rows affected(0.11 seconds)

选择 garden数据库，得到结果如下：

0:jdbc:hive2://linc-2:10000>use garden;

No rows affected(0.108 seconds)

可见，root 用户可以访问所有数据库。

(3)其他权限

在zoo 数据库下，创建一个由id 、name 、location 三个字段组成的表 plants。

输入命令后结果如下：

0:jdbc:hive2://linc-2:10000>create string);

No rows affected(0.204 seconds)

插入一条数据如下：

0:jdbc:hive2://linc-2:10000>insert No rows affected(0.062 seconds)

**查找** **plants 表数据如下：**

0:jdbc:hive2://linc-2:10000>select

table plants(id int,name string, location

into table plants values(1,'lemontree','hill');

\*from plants;

l plants.id l plants.name

+---- --+----

-----+--+

l plants.location

-+--

|1

+-- ·

l lemontree

+-

l hill

---+

删除 plants 表后结果如下：

0:jdbc:hive2://linc-2:10000>drop table plants;

No rows affected(0.633 seconds)

**3.select\_garden** **角色对数据表操作**

(1)beeline 访问 HiveServer 2

这次要验证select\_garden角色对数据表进行的操作，所以设置连接HiveServer2的用 户是拥有该角色的 test 用户。

使用 test 用户，访问 beeline:

$beeline -u"jdbc:hive2://linc-2:10000"-n test -p 123456

**第6章** 大数据安全之ApacheSentry 301

(2)访问权限

之前设置 select\_garden角色只能访问garden 数据库，输入以下命令验证：

0:jdbc:hive2://linc-2:10000>use garden;

No rows affected(0.117 seconds)

0:jdbc:hive2://linc-2:10000>use zoo;

Error:Error while compiling statement:FAILED:SemanticException No valid privileges

User test does not have privileges for SWITCHDATABASE(state=42000,code=40000)

由上面的反馈信息可以看出， test 用户只可以访问garden 数据库，这是因为 select\_ garden 角色在策略文件中定义不能访问除了 garden 之外的数据库。

在 garden 数据库下，操作查询所有数据表：

0:jdbc:hive2://linc-2:10000>show tables;

+- -+

tab\_name

+---

l animals

1 animals\_catonly

+----------------+--+

2 rows selected(0.104 seconds)

(3)其他权限

查询 garden.animals 数据表：

0:jdbc:hive2://linc-2:10000>select \*from garden.animals;

|  |  |  |
| --- | --- | --- |
| +----- | --+-- | 十 |
| I specie | I sound | |food |
| 十—-  l dog  cat  I rabbit  l monkey  I tiger | l bark  meow  ldrum  l chatter  roar | bone  |t  banana  all |
| 十—-- |  |  |

5 rows selected(0.258 seconds)

用test 用户执行删除 animals 表命令，结果显示无法执行：

0:jdbc:hive2://linc-2:10000>drop table animals;

Error: Error while processing statement:FAILED:Execution Error,return code 1from org.apache.hadoop.hive.ql.exec.DDLTask.MetaException(message:hive. metastore.execute.setugi can't be false in nonsecure mode)(state=08S01,code=1)

由于select\_garden角色只能对garden数据库中的表执行 select 命令，所以 test 用户试 图进行 insert 、drop 操作都会被禁止。

**6.4.4** **基于数据库存储方式的数据表操作**

与上节不同，如果使用基于数据库存储的方式配置Sentry store, 可以更方便地以编码

302 第二篇 开源实现篇

式方式管理授权策略。接下来，将具体介绍如何修改配置以及一些操作是如何实现的。

**1.修改** **sentry-site.xml 配置文件**

修改项如下：

<property>

<name>sentry.hive.provider</name>

<value>org.apache.sentry.provider.file.HadoopGroupResourceAuthorizationProvi

der</value>

</property>

<property>

<name>sentry.hive.provider.backend</name>

<value>org.apache.sentry.provider.db.SimpleDBProviderBackend</value> </property>

**删除项如下：**

<property>

<name>sentry.hive.provider.resource</name>

<value>/var/local/hadoop/hive-1.1.0-cdh5.4.2/warehouse/sentry-provider.ini</value> </property>

**2.创建角色和组**

不同于编辑文件管理授权，可以直接在 beeline 下执行创建用户和组，并授予它们具体 的权限。

例如，在beeline 中执行下面的SQL 语句创建角色entire\_server 、test\_server, 创建组

root 、test。

create role entire\_server;

R\_rse;entire\_server;

create role test\_server;

GRANT ALL ON DATABASE garden TO ROLE test\_server;

use zoo;

GRANT SELECT(specie)on TABLE zoo.animals TO ROLE test\_server;

GRANT ROLE test\_server TO GROUP test;

上面创建了两个角色：

口 entire\_server, 具有管理员权限，可以读写所有数据库，并授权给 root 组。

口 test\_server, 只能读写garden 数据库，只能读取zoo 库中 animals 表中的 specie 字 段，并授权给test 组。

**3.entire\_server 角色对数据表操作**

下面展示root 组在 entire\_server 角色下所能执行的一部分操作。

(1)beeline 访问 HiveServer2

使用 root 用户访问beeline:

第6章 大数据安全之ApacheSentry 303

s beeline -u "jdbc:hive2://linc-2:10000"-n root -p 123456

(2)查看当前系统用户

由于beeline 的命令行不会显示用户身份，当需要查询目前与beeline 连接的是哪个系 统用户时，可以输入以下命令：

0:jdbc:hive2://linc-2:10000>set

+--

set

+- --+

I system:user.name=root

-+--+

1 row selected(0.042 seconds)

system:user.name;



set system:user.name命令只能在基于数据库方式下使用，基于文件存储不支持这样 的操作命令，包括之后提到的 show roles 命令。

(3)查看所有角色

root 用户属于entire\_server 角色下，具有管理员权限，可以查看所有角色。

0:jdbc:hive2://linc-2:10000>show roles;

+-- · --+--+

role

+-, -+--+

l entire\_server

l test\_server

- +

**(4)访问权限**

**root 用户可以访问任意数据库。**

0:jdbc:hive2://linc-2:10000>use garden;

No rows affected(0.117 seconds)

0:jdbc:hive2://linc-2:10000>use zoo;

No rows affected(0.105 seconds)

**(5)其他权限**

**可以查看** **zoo.animals 数据表。**

0:jdbc:hive2://linc-2:10000>select \*from zoo.animals;

+-- 十 · +-- +-

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ip I | specie | l sound food | |  |
| 十 = -  1.2  2.3  3.4  4.5  5.6 | dog  cat  rabbit  monkey  tiger | bark  meow  drum  chatter  roar | bone  I fish  carrot  I banana  all |  |

**304** **第二篇** **开源实现篇**

chatter roar

5 rows selected(0.257 seconds)

**4.test\_server 角色对数据表操作**

使用test 用户访问 beeline。

$beeline -u "jdbc:hive2://linc-2:10000"-n test -p 123456

(1)查看所有角色

test 用户不是管理员，是不能查看所有角色的。

0:jdbc:hive2://linc-2:10000>show roles;

ERROR:Error processing Sentry command:Access denied to test.Please grant admin privilege to test.

Error:Error while processing statement:FAILED: Execution Error, return

SentryAccessDeniedException:Access denied to test(state=08S01,code=1)

code 1 from org.apache.hadoop.hive.ql.exec.SentrygrantRevokeTask.

(2)访问权限

test 用户可以列出所有数据库。

0:jdbc:hive2://linc-2:10000>show databases;

+-------------+--+

1 database\_name

+---------

l zoo

I garden

l default

l test\_db

4 rows selected(0.346 seconds)

test 用户可以查询garden数据库。

0:jdbc:hive2://linc-2:10000>use

garden;

No rows affected(0.052 seconds)

(3)其他权限

查询 garden数据库中的 animals表格。

0:jdbc:hive2://linc-2:10000/>select \*from

l specie

l food

l bone

l fish

I carrot

banana l all

l sound

l bark

meow

|drum

+--

|  |  |
| --- | --- |
| l  l  I  l | dog  cat  rabbit  monkey tiger |

十-

5 rows selected(0.199 seconds)

garden.animals;

-+

+

+

**第6章** 大数据安全之Apache Sentry **305**

**test 用户只能查看** **zoo 数据库中** **animals 表中的** **specie 字段，其他的字段都无法查看。**

0:jdbc:hive2://linc-2:10000>use zoo;

No rows affected(0.112 seconds)

0:jdbc:hive2://linc-2:10000/>select.\*from animals;

Error:Error while compiling statement:FAILED:SemanticException No valid privileges

User test does not have privileges for QUERY(state=42000,code=40000) 0:jdbc:hive2://linc-2:10000/>select specie from animals;

+---- · -+--+

I specie

+

| dog

cat

rabbit

1.monkey

l tiger

**6.5** **本章小结**

本章首先介绍了 Apache Sentry 的技术概况，它作为 Hadoop 组件中一个策略引擎插件， 被数据处理工具用来验证访问权限。为了让读者对 Apache Sentry 有个直观的了解，本章以 时间顺序列出其发展近况：从最开始 Cloudera 将 Sentry 贡献给了 Apache 开源社区，在过 去的4年时间里， Apache Sentry 共发布过6个版本， Sentry 社区也一直在不断发展壮大， 为 Hadoop 生态圈带去源源不断的活力。

而目前，Sentry 作为 Apache 的顶级项目，其特点优势也不容小觑。它具有规范性、安 全的授权机制、细粒度访问控制、基于角色的管理、多租户管理、统一平台，这推动了大 数据技术在更多行业、组织和终端用户的使用。

之后，为读者介绍了Apache Sentry 的架构，并逐层分析了 Sentry 的架构组件： Sentry 服务器、数据引擎、Sentry 插件，还对各组件的工作原理和彼此之间如何交互进行了简要 分析。为了让读者理解 Apache Sentry 的核心思想，并学会基于 Hadoop 搭建一个细粒度授 权系统，又介绍了 Apache Sentry 安装配置，在与现有 Hadoop 系统组件的集成与使用方面， 介绍了Impala 的部署安装与调试，以及它与 Apache Sentry 的集成操作。

最后做了两个场景设计：关于Sentry 对 Impala 的控制和 Sentry 对 Hive 的控制。读者 在对这些设计场景全面地学习后，可以更加深入透彻地了解 ApacheSentry 的特点优势。

通过以上对Apache Sentry 的基本介绍，希望读者能了解并学会 Sentry 的操作和功能， 逐步掌握并灵活运用，在对其进行安装、配置、集成后能有自己的体会和收获。



■国

■■

■■

■ ■

■

■

■1■|■ ■■

■

■

■1■ ■

■

国面海面

■■ 题

■

培邮加置

 ml ■1

■

■

■

第7章

**大数据安全之** **Kerberos** **认证**

在之前的章节中，主要介绍了通过 Apache Falcon 与 Apache Atlas 对 Hadoop 集群上的 数据进行治理，并且使用Apache Ranger 和Apache Sentry 对该方案进行安全认证以及权限 管理。本章将介绍Kerberos 网络授权协议，使用 Kerberos 对本书所介绍的方案进行网络传 输方面的保障，同时也作为第5章中 Apache Ranger 进行细粒度安全认证的配置条件之一。

**7.1** **Kerberos 概述**

**7.1.1** **Kerberos** **技术概况**

Kerberos 协议是一种计算机网络授权协议，用于在非安全的网络环境下对个人通信进 行加密认证。Kerberos 协议认证过程的实现不依赖于主机操作系统的认证，无须基于主机 地址的信任，不要求网络上所有主机的物理安全，并且是以假定网络上传送的数据包可以 被任意地读取、修改和插入数据为前提设计的。

Kerberos 设计的目标如下：

1)用户的密码不能在网络上传输。

2)用户的密码绝不能以任何形式存储在客户机上，使用后必须立刻销毁。

3)用户的密码不应该以未加密的形式被存储。

4)用户在一次工作会话期间只会被要求输入一次密码，因此用户能够在这次会话期间 透明地访问所有服务而无须重新输入密码。

5)认证信息的管理集中在认证服务器上，应用服务器上不能保留用户的身份验证信 息，以此来实现以下的功能。

口管理员可以通过修改认证服务器的设置来对指定用户的账户进行修改，而不用修改

第7章 大数据安全之 Kerberos 认证 307

应用服务器。

口当用户修改密码时，将在所有服务器上生效。

口没有需要保护的冗余信息。

口不仅用户需要向应用服务器证明他们的身份，应用服务器在返回时也需要向客户端 证明它们的真实性，实现双向验证。

口在完成身份的验证后，客户端与服务器必须能够建立一个加密的连接，因此 Kerberos 需要为生成和交换加密密钥提供支持。

本章中所使用的Kerberos 是麻省理工学院为 Kerberos 协议所开发的一套 C/S 架构的软 件，客户端与服务端之间可以进行相互认证，保护双方免受窃听与重复攻击。

Kerberos 通过定义用户和服务端所使用的认证身份 (Principal) 来进行访问控制，用户 通过Kerberos 客户端使用自己的 Principal 向认证服务器进行身份的认证，认证成功后服务 器会将表示用户身份的票据 (Ticket) 返回用户，在之后的通信过程中Kerberos 客户端使用 已认证的票据进行安全的通信。

Kerberos的应用覆盖 Windows 、Linux 以及Mac OS系统，多用于大型系统、Web 应 用、企业网等需要高安全性的系统软件中。微软、苹果、Red Hat 等公司的产品中均使用 Kerberos, 它甚至在X-Box、有线电视产品中也扮演了十分重要的角色。可以说，Kerberos 是计算机网络发展中应用范围最广的认证方式之一。

本章将主要介绍Kerberos 的使用和原理，并讲解将Kerberos 应用于分布式系统 Hadoop以及与其相关的10个组件所需要的集成配置操作。由于不同的 Hadoop 组件集成 Kerberos 的步骤与组件的正常安装步骤不同，因此本章中会基于前文中的正常安装步骤， 对于需要修改、添加的特殊步骤进行说明。

**7.1.2** **Kerberos 发展史及近况**

最初的 Kerberos协议是由美国麻省理工学院为了保护雅典娜工程 (Project Athena) 提 供的网络服务器而研发的。目前该协议存在一些版本，其中版本1～3只在麻省理工学院内 部发行使用。

1980年年末， Kerberos v4由 Steve Miller 和 Clifford Neuman发布，这个版本主要针对雅 典娜工程 (Project Athena) 。Kerberos v5 由 John Kohl和 Clifford Neuman 设计，在1993年作 为 RFC1510 颁布(在2005年被RFC4120 取代),目的在于克服v4 的局限性和安全问题。

麻省理工学院在版权许可的情况下，开发了一个Kerberos 的免费实现工具，这种情况 类似于BSD 。在2007年，麻省理工学院组成了一个Kerberos 协会，以此推动 Kerberos 的 持续发展。

因为使用了 DES 加密算法(56位密钥),当时的美国出口管制当局把 Kerberos 归类为 军需品，并禁止其出口。因而瑞典皇家理工大学研发了一款 Kerberos v4的实现工具 KTH- KRB, 它使得这套系统在美国更改密码出口管理条例(2000年)前，就可以在美国境外使

**308** 第二篇 开源实现篇

用。瑞典的实现工具基于一个叫作eBones 的版本，而eBones 基于麻省理工学院对外发行 的基于Kerberos v4的补丁9的Bones (跳过加密公式和对它们的函数调用)。Kerberos y5 的实现工具 Heimdal, 基本上也是由发布KTH-KRB 的同一组人发布的。

在那之后，微软在 Windows 2000 以及后续的操作系统中都以 Kerberos 协议为默认的 认证方法，虽然没有使用麻省理工学院开发的Kerberos 软件，但也成为了Kerberos 广泛使 用的标杆。继而 Red Hat Enterprise Linux4和后续的操作系统也使用Kerberos 的客户和服 务器版本，苹果计算机公司的Mac OS X也使用 Kerberos 的客户和服务器版本。

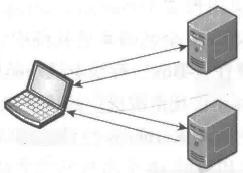
近年来，随着大数据领域的不断发展，分布式系统的安全成为大数据安全中最为重要 的部分之一，Apache 旗下的 Hadoop 、Hive 、HBase 等多个组件也逐渐对Kerberos 提供了 支持， Kerberos 已经成为应用最为广泛的安全认证方式之一。

**7.1.3** **Kerberos 架构**

本节将介绍Kerberos 服务中涉及的各个组件，以此描述Kerberos 服务的架构。图7-1

描绘了Kerberos 的总体架构，它由Kerberos 服务器 KDC、客 户 机 (Client) 和应用服务 器 (Application Server) 组成，其中KDC 包 含了Authentication Service 和 Ticket Granting Service 两种服务。

1)KDC:Key Distribution Center, 即 密钥分发中心。就是通常所说的认证服务器， 它是参与用户和服务认证的基本对象。由于



Database

Authentication Service Ticket Granting Service

KDC

Client

Application Server

图7-1 Kerberos 架构

它有密钥分配的功能并且可以作为服务接入，因此被称为密钥分发中心，简称KDC 。KDC 通常是一台单独的物理服务器，它可以从逻辑上分为3个部分：数据库(Database) 、 认证 服务(Authentication Service) 和票据授权服务 (Ticket Granting Service)。

2)Database: 数据库，用于存放用户和服务的记录， Kerberos 的数据库中使用 Principal 来命名和引用一条记录。Kerberos 数据库中的记录包含以下内容：

●记录所关联的 Principal。

●加密密钥和相关的KVNO。

● 与Principal关联的票据的最长有效期。

● 与Principal关联的票据的最长更新周期。

●描述票据的参数或标志。

●密码过期时间。

● Principal 的过期时间。

3)Application Server: 应用服务器，在本章中指所有提供 Hadoop 以及相关服务的主 机，应用服务器上需要安装Kerberos 客户端，在相关服务中开启对 Kerberos 协议的支持，

第7章大数据安全之 Kerberos 认证 309

并且对服务所使用的 Principal 进行配置。

4)Client: 客户机，用户使用客户机来获得应用服务器提供的各项服务，客户机上需要 安装 Kerberos 客户端。在使用时，用户需要先向KDC 进行身份的认证，才能从应用服务 器获得相应的服务。

5)AS:Authentication Service, 认证服务。认证服务是 KDC 中用于回复客户端最初的 认证请求的部分，如果用户没有认证过，必须输入密码。在回应认证请求时， AS 会授予一 个特殊的被称为票据授权的票据 (Ticket Granting Ticket),简称 TGT。如果用户确实是他们 所声称的身份，他们就可以使用TGT 在无须再次输入密码的情况下获得其他服务的票据。

6)TGS:Ticket Granting Service,票据授权服务。票据授权服务是KDC 中负责根据 用户提交的有效TGT 分配服务票据 (Service Ticket) 的组件，同时 TGS 保证向应用服务器 请求资源的身份的真实性。TGS 可以被视为一个应用服务器，提供服务票据的功能。

7)ST:Service Ticket,服务票据，由KDC 的 TGS 发放，任何一个应用 (Application) 都需要一张有效的服务票据才能访问。如果能正确接收 ST, 说明Client 和 Server 之间的信 任关系已经被建立。ST 通常为一张数字加密的证书。

8)TGT:Ticket Granting Ticket,票据授权票据，由KDC 的 AS 发放。获得这样一 张票据后，再申请其他应用的服务票据 (ST) 时，就不需要向KDC提交身份认证信息 (Credential) 。TGT 具有一定的有效期，到期后需要更新来续约。

9)Ticket: 票据。票据是客户端提交给应用服务器用于证明其身份真实性的。票据由 认证服务器颁发，并使用所需要的服务端密钥加密。由于服务端密钥只存在于认证服务器 与应用服务器之间，获取到该票据的客户端也无法知道服务端密钥，因而也无法对票据进 行修改。 一张票据中包含了以下的信息：

●请求用户的 Principal(一般来说是用户名)。

●用户所请求的服务 Principal。

●可以使用该票据的客户端IP 地址(可选)。

●票据的生效日期与时间(使用时间戳格式)。

●票据的有效时间。

●会话密钥 (Session Key)。

每张票据都会有过期时间(通常为10小时),这是必要的。虽然认证服务器的管理员可 以控制不再发布新的票据，但无法阻止用户使用已经发布的票据。因为认证服务器无法对 已经发布的票据进行控制，所以过期时间的设置可以防止票据被滥用。

10)KVNO:Key Version Number, 密钥版本号。当用户更改密码或管理员更新应用服 务器的密钥时，这种修改将会使版本号增加。

**7.1.4** **Kerberos** **的认证流程**

本节将会对认证过程中被传递的数据包进行说明，以此来描述Kerberos 认证的简要流

310 第二篇 开源实现篇

程。需要强调的是，应用服务器在整个过程中不会直接与KDC 进行通信，服务票据、TGS 等数据全部由客户端发送至它们所需要的应用服务器。

Kerberos 认证流程如图7-2所示。客户端传输的数据包顺序是从上至下，其中数据包 内容如下。

1)AS\_REQ:AS\_REQ 是 最 初 的 用 户 认 证 请 求 ， 由kinit 命令生成。这个信息指向 KDC 中 的AS 服务。

2)AS\_REP:AS\_REP 是 AS 服务对先前请求的答复，它主要包含TGT ( 使 用TGS 密

钥加密)和会话密钥 SK\_TGS (使用请求认证的用户密钥加密)。

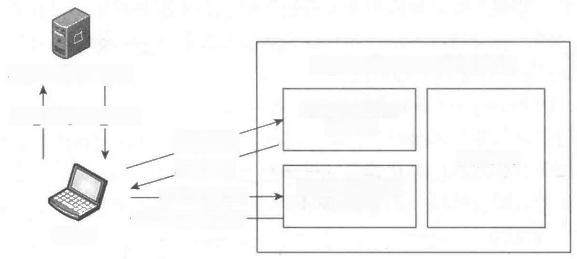
3)TGS\_REQ:TGS\_REQ 是客户端向TGS 服务请求服务票据 (ST) 的请求包。这个

数据包包含了之前的 TGT 数据以及一个由会话密钥加密过的客户端身份凭证。

4)TGS\_REP:TGS\_REP 是 TGS 服务对之前请求的答复。其中包含了客户端所请求的 服务票据(使用服务密钥加密)以及 一个由TGS 生成的会话密钥 SK\_Service (使用之前 AS 服务生成的会话密钥加密)。

5)AP\_REQ:AP\_REQ 是客户端发送给应用服务器用于使用服务的请求。其中包含了 之前从TGS 服务请求到的服务票据以及由客户端生成的身份凭证，但这次使用由TGS 生 成 的会话密钥 (SK\_Service) 加密。

6)AP\_REP:AP\_REP 由应用服务器发送给客户端，证明自己确实是用户所请求的服 务端。这个数据包是可选的，只有在使用了相互认证机制时客户端才会请求这个数据包。



Application Server

AP\_REQAP\_REP

AS\_REQ AS\_REP

—TGS\_REQ ·

←—TGS\_REP ·

Client

Key Distribution Center (KDC)

Authentication

Service

Database

Ticket Granting

Service

图7-2 Kerberos 认证流程

认证的具体流程如下：

1 ) 客 户 端 向KDC 中 的AS 服务发送AS\_REQ 请 求 身 份 验 证 ，AS 服务返回AS\_REP, 其中包括为用户和TGS 生成的 一 个会话密钥SK\_TGS, 并发送使用用户密钥加密的TGT、

SK\_TGS。

这个请求验证的过程实际上是使用kinit 命令来完成的， kinit 将用户名传给AS 服务， AS 服务查找用户名的密码，将TGT 和 SK\_TGS 使用用户密码加密后发送给 kinit,kinit 要 求用户输入密码，解密后得到TGT 和 SK 。其 中 ，TGT 使 用 TGS 的密钥加密。

第7章 大数据安全之 Kerberos 认证 311

2)客户端向KDC 中的TGS 服务发送 TGS\_REQ, 请求访问某个应用服务器的服务票 据 (ST), 发送 TGT 和身份凭证(Authenticator)。

其中，身份凭证用于验证发送该请求的用户就是TGT 中所声明的，身份凭证是使 用TGS 和用户之间的会话密钥SK 加密的，防止TGT 被盗。TGS 先使用自己的密钥解开 TGT, 获得它与用户之间的会话密钥，然后使用SK 解密身份凭证，验证用户和有效期。

3)TGS 判断无误后，为用户和应用服务器之间生成一个新的会话密钥： SK\_Service, 然后发送 TGS\_REP给用户，其中包括 SK\_Service 和服务票据 (ST)。

其中，服务票据是使用应用服务器的密钥加密的，SK\_Service 使用 TGS 和用户之间的 会话密钥 (SK\_TGS) 加密的。

4)用户使用与TGS 之间的会话密钥SK\_TGS 解开包得到与应用服务器之间的会话秘 钥 SK\_Service, 然后使用SK\_Service 生成一个身份凭证 (Authenticator), 向应用服务器发 送AP\_REQ, 其中包括服务票据 (ST) 和身份凭证。

其中，此处的身份凭证是使用用户和应用服务器之间的会话密钥 (SK\_Service) 加 密的，应用服务器收到后先使用服务器的密钥解密服务票据 (ST), 或者会话密钥 (SK Service), 然后使用会话密钥 (SK\_Service) 解密身份凭证 (Authenticator) 来验证发送请求 的用户就是票据中所声明的用户。

5)应用服务器向用户发送一个数据包AP\_REP, 以证明自己的身份，这个包使用会话 密钥 (SK\_Service) 加密。客户端会等待应用服务器发送确认信息，如果不是正确的应用服 务器，就无法解开 ST, 也就无法获得会话密钥，从而避免用户使用错误的服务器。

此后用户与应用服务器之间使用SK\_Service 进行通信，且在 TGT有效期内，用户将跳 过第1步的身份验证，直接从第2步使用TGT 向TGS 证明自己的身份。

**7.1.5** **Kerberos** **的风险与缺陷**

Kerberos 虽然是一套性能较高的安全加密系统，但若使用不当或疏于管理，同样会遇 到风险。本节简要介绍在 Kerberos 使用过程中可能遇到的风险及其缺陷。

**1.单点失败**

通过上文对 Kerberos 认证流程的介绍可以发现， Kerberos 服务几乎全部依赖于KDC 上 的服务， 一旦KDC 所在的主机发生故障，将导致所有配置了Kerberos 的服务无法使用。

目前对于Kerberos 单点失败的问题，主要通过以下两种方法来弥补。 (1)复合Kerberos 服务器

复合Kerberos 服务器，顾名思义，即使用多台服务器以备用服务器或分布式服务器的 形式代替原本单一KDC 的方法，这种方法能够在一定程度上减少KDC 故障崩溃后对整个 系统的影响。

(2)后备认证机制

后备认证机制指的是当一种首选的认证方法失败后，启用预先准备好的次要认证方法

312 第二篇 开源实现篇

的机制。通常这些机制需要相对应的服务软件提供支持， 一些成熟的服务软件能够支持多 种认证方式，并提供后备认证机制，以此来减少故障后的影响。

**2.时钟同步**

Kerberos 认证机制中，认证服务器要求所有参与的主机时间同步。同时，票据存在有 效期，如果客户端与服务端的时钟不同步，则会导致认证失败。MIT 给出的默认配置中， 要求时钟偏差不能超过5分钟。在实际使用过程中，通常会使用网络时间协议守护进程 (Network Time Protocol daemons) 来保证时间的同步。然而大多数情况下，网络时间协议的 安全性并不高， 一旦主机的时间发生错误，就很有可能影响整个系统的运作。

**3.安全依赖**

虽然 Kerberos 能够让主机间的通信变得更为安全，但那仅仅是对于使用Kerberos 加密 协议进行通信的软件。如果搭载了 Kerberos 的主机本身安全性薄弱，导致发生被攻击者入 侵或主机账号被盗取等情况， Kerberos 为系统软件提供的保护极有可能因为本地 Keytab 文 件被盗取等原因而失效。

**4.集中式管理**

KDC管理着所有用户与服务的 Principal 以及密码，在大型分布式系统中，可能存在多 个作用域、多个认证服务器的情况，域之间会话密钥的数量惊人，这对于密钥的管理、分 配和存储，以及主机的负载都将是严峻的挑战。

**5.krbtgt** **账户**

krbtgt是 KDC 的服务账户，在每一个Realm 中都存在一个 krbtgt 账户。krbtgt 账户用 来创建 TGT的加密密钥。在Kerberos 的认证机制中，使用krbtgt 生成的密钥来加密TGT, 因此理论上只有两方能够解密TGT, 而只要能够正确地解密TGT,Kerberos 就会认为其中 的信息是可信的。

在这个前提下， Kerberos 中的 krbtgt 账户是唯一一个密码不会自动更新的账户，只有 在主机进行灾害恢复，或作用域进行功能升级导致账户变动时才会更新。因此在长期使用 的情况下，如果 krbtgt 的账户密码被盗取，则可能会产生严重的安全问题。

**7.1.6** **Kerberos 应用举例**

本节主要介绍 Kerberos 在与 Hadoop 及其相关组件集成之后，通过LDAP 、PAM 、SSSD 等组件进行集群用户的管理案例。

**1.应用需求**

之所以要在 Hadoop 集成 Kerberos 的同时，使用LDAP 、PAM 、SSSD来进行用户管理， 是为了应对Hadoop 在配置了Kerberos 的情况下，会使用用户Principal 中的前缀名作为启 动Hadoop 的用户名，这样的机制虽然能够迎合Kerberos 的安全需求，但 Hadoop 在通过

第7章 大数据安全之 Kerberos 认证 313

YARN执行任务时，在Slave 节点上需要同名的系统用户作为Container 的启动用户。

在通常的使用中，可能会在各个集群节点上为Hadoop 服务单独创建一个名为hadoop 的系统用户作为服务的启动用户和作业的提交用户。Hadoop 也会沿用Linux 用户作为作业 的提交用户，因此在通常情况下，使用hadoop 用户提交作业时，由于集群节点上都存在 hadoop用户，因此可以正常启动Container 。但是在配置了Kerberos 的情况下，Hadoop 会 沿用提交用户的 Principal 前缀作为作业的提交用户，由于Principal 账户往往不会存在于所 有集群节点上，因此在启动Container 的过程中会遇到找不到相应用户的情况。也就是说， 如果使用了类似 user@LINC.COM 的 Principal 进行作业的提交，那么 Hadoop 会使用 user 作为作业的提交用户，从而在相应的集群节点上使用user进行作业的启动。

为了能够适应企业中对于Linux 系统用户以及Kerberos 账户的管理需求，这里希望每 个用户能够使用自己的 Principal进行作业的提交，同时为了不影响 Hadoop 作业的正常运 行，需要能够在各个集群节点上方便地创建与 Principal 相对应的系统账户的方法。

**2.组件介绍**

1)LDAP, 全称为Lightweight Directory Access Protocol, 轻量目录访问协议。可以理 解为一种以树结构存储数据的数据库，其中每一条记录都是树中的一个节点，每个节点都 有唯一的路径来标识。这种结构使得 LDAP 在数据的读取和查找方面性能优异，非常适合 用于作为用户账号信息的管理。

2)PAM, 全称为 Pluggable Authentication Modules,可插拔认证模块。是集成于Linux 系统中的一种认证机制，它通过提供一些动态链接库和一套统一的API, 将系统提供的服 务与服务的认证方式分开，使得系统管理员可以灵活地根据需要给不同的服务配置不同的 认证方式而无须更改服务程序，同时也可以方便地添加新的认证方法。

3)SSSD,System Security Services Daemon, 系 统 安 全 服 务 守 护 进 程 。SSSD 是 Red Hat公司旗下Linux 系统中存在的守护进程，它可以用来访问多种认证服务器，例如 Kerberos 、LDAP 等，并提供授权。SSSD 是介于本地用户和数据存储之间的进程，本地客 户端先连接 SSSD, 再 由SSSD 连接外部认证服务器。

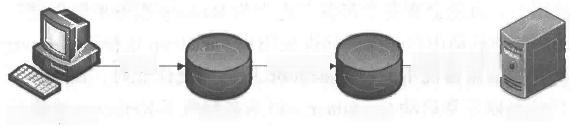
**3.工作原理**

通过配置LDAP 、SSSD 、PAM组件，可以实现将LDAP 中存储的用户账号自动地映射 到系统本地，也就是说用户可以在任意一台配置了PAM-SSSD-LDAP 组件的节点上，使用 LDAP 中存储的账号作为系统用户登录，即使原本系统中不存在该用户。同时也可以在任 意一台配置了这些组件的节点上查询 LDAP 服务器上的用户、用户组信息。

如图7-3所示，这套方案的工作原理是通过配置PAM组件，让系统将LDAP 作为系统 用户的认证方式之一，并且将这部分用户的认证过程交给LDAP, 在认证成功后，如果系 统本地没有该用户则会根据LDAP 中的数据自动创建。再通过SSSD 组件的配置将认证的 信息交付给所需要的LDAP 服务器完成认证。SSSD 在连接LDAP 服务器后会周期性地缓

314 第二篇 开源实现篇

存 LDAP 中存储的用户信息至本地，以保证节点上的数据与LDAP 中的同步。

认证

认证源→

认证/缓存

Local System

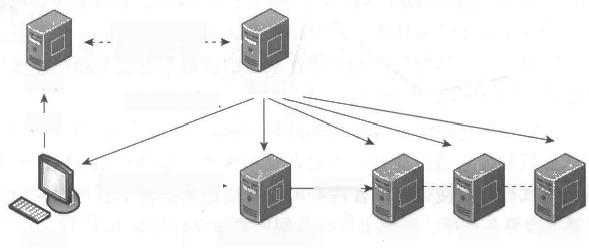
SSSD

PAM

LDAP

图7-3 PAM-SSSD-LDAP 认证流程

这套方案的架构如图7-4所示。在此基础上，需要保证用户在LDAP 服务器中的账号与 用户的 Principal 前缀相同。这个过程可以在新建账户时手动完成，或通过程序自动实现，此 处不赘述。在实现了这套方案后，假设用户user通过Kerberos认证了user@LINC.COM 的 Principal, 并 向 Hadoop 提 交 了 作 业 ，Hadoop 会 将 Principal 中 的 user 前缀提取出来作为作 业的提交者，并且在集群节点上以user 用户的名义开始作业。由于保证了用户的 Principal 前 缀 在LDAP 中存在同名的用户账号，因此在所有集群节点上，都能够找到由PAM 、SSSD 组件所缓存的存在于LDAP 服 务 器 中 的user 用户，因而作业能够正常完成。



Principal前缀

对应用户名

LDAP

同步用户信息

认证

—提交任务 → -启动一

Client Hadoop Master Slave Slave Slave

KDC

图7-4 PAM-SSSD-LDAP 方案架构

**7.2** **Kerberos** **使用操作说明**

本节将会对Kerberos 的使用操作进行说明，并对本章中用到的专有名词进行解释。

7.2.1 名词解释

1)KDC:Key Distribution Center,密钥分配中心。本章中专指Kerberos 认证服务的 密钥分配中心，负责Kerberos 密钥的分发和存储， KDC中自带一个数据库用于Realm 和 Principal 等数据的存储。

2)Principal:Principal 是 Kerberos 中参与认证的基本实体。通常来说 Principal 的含义

有两种， 一 种用于表示 Kerberos 数据库中的用户， 一 种用于表示 一 台特定的主机，可以认 为 Principal是代表了用户或主机的身份信息。

第7章 大数据安全之Kerberos 认证 315

3)Principal 的命名格式：通常 Principal 的命名格式为 Name/Instance@Realm 。其中 Name 代表了用户名或主机名，Instance 是对Name 指代用户或主机的进一步描述，可以是 用户所在的主机或类型等信息。Instance 可以省略，在未省略的情况下需要与Name 使用 “/”号隔开。

在本章中，为了对各个服务进行合理的权限规划，在每一台服务节点上为每一个单独 服务创建单独的系统用户，每个服务由各自对应的系统用户进行开启和维护，例如 hdfs、 yarn 、hive 等。因此所使用的Principal 主要采用了以下两种格式：用户使用Username@ Realm, 服务节点的系统用户使用 Username/Hostname@Realm。

4)Principal 的认证： Kerberos 中支持的用户认证方式有两种， 一是由用户输入密码 进行认证，二是将Principal 导出至Keytab 文件后使用 Keytab 文件免密码认证。对于一个 Principal 来说只能选择其中一种方式进行认证。由于 Hadoop 及其各组件中使用的机制原 因，对于服务端的 Kerberos 认证只能使用Keytab 文件进行认证。

5)Realm: 作用域，指认证系统所管理的范围，用于指定认证服务所影响的用户、主机 以及服务的边界。用户与服务并不一定要属于同一个Realm, 如果双方所属不同的 Realm, 并且这两个Realm 直接存在信赖关系，就能够实现交叉认证。 一般来说，当一个用户或服务 属于一个 Realm时，它与认证服务器直接存在一个共享的Realm 密钥。Realm 的名称也是 KDC中存储 Principal 的域数据库名。Realm 在使用时对字母的大小写敏感，因此一般使用 大写字母来区分，同时建议使用网络环境的 DNS 作为Realm。在输入Principal 时 ，Realm 可以省略，省略后默认使用本地的Realm 。 在未省略时使用“@”号与 Name 或 Instance 分隔(本章中为了降低管理配置的复杂度，只使用一个 Realm, 并命名为LINC.COM)。

6)Keytab: 保存了一个或多个 Principal 及其信息的密钥文件，由KDC 管理员导出生 成，用户可以直接使用该文件进行免密码认证，因此需要注意密钥文件的文件权限。

需要注意的是， KDC 与 Keytab存在一种过期机制，当KDC 导出一个 Principal 时，将 会生成一个该 Principal 的 KVNO, 这个版本号会被保存在 KDC 以及导出的Keytab 文件 中，当用户使用 Keytab 认证时，如果Keytab 中的KVNO 标签与KDC 中 该 Principal 最后 生成的KVNO 不一致，将会认证失败。简单来说，如果使用KDC 将一个 Principal 导出多 次，那么最终只有最后导出的 Keytab 中的信息可以用于认证。

因此，如果需要将某个 Principal 信息导出到多个不同的 Keytab 文件来使用，则需要 使用ktutil 工具将 Keytab 文件中的信息进行复制、合并。本章中由于用户规划以及Hadoop 中机制问题，会出现两个不同服务需要使用同一个 Principal 的情况，请注意按照配置步骤 进行操作。

**7.2.2** **KDC** **常用操作**

1)kadmin.local: 打开KDC 控制台，需要root 权限。

2)[addprinc <principal>]: 添加 Principal, 在 KDC 控制台使用，执行后需要两次输入

316 第二篇 开源实现篇

该Principal 的密码。

3)[delprinc <principal>]: 删除Principal, 在 KDC 控制台使用，执行后需要确认是否 删除。

4)[xst-k <keytab fle path><principal>]: 导出指定Principal 到指定Keytab 文件，在 KDC控制台使用。如果指定的Keytab 文件已存在，则会将指定Principal 信息合并至 Keytab 文件中。

5)[modprinc -<parameter><parameter value><principal>]: 针对指定的Principal 进行 某项参数的修改，在KDC 控制台使用。需要注意的是，使用modprinc 命令修改的参数， 优先级会比 kdc.conf 中的高。

6)[getprinc <principal>]: 查看指定 Principal 的信息，包括每种密钥的加密类型、 KVNO标签等，在KDC 控制台使用。

7)[listprincs]: 列出所有 Principal, 在 KDC 控制台使用。

**7.2.3** **Client** **常用操作**

1)klist: 列出当前系统用户的 Kerberos 认证情况。

2)klist -kt<keytab file path>: 列出指定 keytab 文件中包含的 Principal 信息，需要该 文件的读权限。

3)kinit<principal>: 使用输入密码的方式认证指定的Principal。

4)kinit -kt <keytab file path><principal>: 使用指定 keytab 中的指定 Principal 进行 认证，需要该文件的读权限。

5)kdestory: 注销当前已经认证的Principal。

6)ktutil: 进入 keytab 工具控制台。

7)[list][1]: 列出当前 ktutil 中的密钥表。

8)[clear][clear\_list]: 清除当前密钥表。

9)[rkt<keytab file path>]: 读取一个 keytab 中的所有 Principal 信息到密钥表，需要有 该文件的读权限，在ktutil 中使用。

10)[wkt <keytab file path>]:将密钥表中的所有 Principal 信息写入指定文件中。如果 文件已存在，则需要该文件的写权限，信息会附加至文件末，在ktutil 中使用。

11)[addent -password -p <principal>-k <KVNO>-e <enctype>]: 手 动 添 加 一 条 Principal 信息到密钥表，执行后需要输入指定Principal 的密码。由于手动添加的信息 ktutil 不会进行验证，因此不推荐使用。

12)[delent <entity number>]: 从密钥表中删除指定行号的Principal 信息，行号可使用 list或1命令查看，在ktutil 中使用。

13)[lr][list\_request]: 列出所有 ktutil 中可用的命令，在ktutil 中使用。 14)[quit][q][exit]: 退出 ktutil 控制台。

第7章 大数据安全之Kerberos 认证 317

**7.3** **Kerberos 集成环境配置**

本节将会对如何进行Hadoop多项组件的Kerberos集成配置进行说明，包括的组件有 HDFS、YARN 、Hive、Zookeeper、HBase 、Sqoop 、Hue 、Spark、Solr、Kafka 、Storm 、Impala。

本节中所包含的安装配置都基于以下的前提。

**1.集群环境**

系统版本： CentOS 6.5

组件版本：如表7-1所示。

**表7-1** **集群环境所使用的组件版本**

|  |  |  |  |
| --- | --- | --- | --- |
| **组** **件** **名** | **版** **本** **号** | **组** **件** **名** | **版** **本** **号** |
| Hadoop | 2.5.0 | Hue | 3.9.0 |
| Kerberos | 5 | Storm | 1.0.0 |
| Zookeeper | 3.4.8 | Spark | 1.6.2 |
| Hive | 1.2.1 | Solr | 5.3.0 |
| HBase | 1.0.3 | Kafka | 2.10 |
| Sqoop | 1.4.6 | Impala | 2.2.0 |

**2.系统用户名和用户组**

本节中每一个 Hadoop 组件均分配一个拥有 root 权限的系统用户，作为组件的执行者， 同时定义testuser 用户作为测试用户，如表7-2所示。

**表7-2** **集群上所使用的系统用户和用户组以及对应的组件**

|  |  |  |  |
| --- | --- | --- | --- |
| **用户：用户组** | **对** **应** **组** **件** | **用户：用户组** | **对** **应** **组** **件** |
| hdfs:hadoop | Hadoop HDFS | hbase:hadoop | HBase |
| yarn:hadoop | Hadoop YARN | sqoop:hadoop | Sqoop |
| mapred:hadoop | Hadoop YARN | solr:hadoop | Solr |
| hive:hadoop | Hive | kafka:hadoop | Kafka |
| zookeeper:hadoop | Zookeeper | storm:hadoop | Storm |
| hue:hadoop | Hue | impala:hadoop | Impala |

**3.主机名**

由于 Hadoop 会将所有的主机名转化为小写字母执行，而Kerberos 对大小写字母敏感， 因此集群中的所有主机名必须为小写字母才能正常运行。本章中所有的主机名均以linc 为 前缀。

**4.集群说明**

本节中所使用的Hadoop 集群如表7-3所示。使用3台主机作为集群节点，1台主机作 为Kerberos 服务端，1台主机作为测试用客户机。

318  第二篇 开源实现篇

**表7-3** **集群中各节点角色及服务说明**

|  |  |  |  |
| --- | --- | --- | --- |
| **Host** | **操** **作** **系** **统** | **角** **色** | **服** **务** |
| linc-1 | CentOS 6.5 server | 集群Master节点 集群Slave节点 | NameNode、DataNode、ResourceManager、Zookeeper、 Jobhistory、Hive、HBase、Hue、Sqoop、Storm、Solr、 Spark、Kafka、Impala |
| linc-2 | CentOS 6.5 server | 集群Slave节点 | DataNode、NodeManger、Zookeeper、HBase、Storm、 Spark、Kafka、Impala |
| linc-3 | CentOS 6.5 server | 集群Slave节点 | DataNode、NodeManger、Zookeeper、HBase、Storm、 Spark、Kafka、Impala |
| linc-krb | CentOS 6.5 server | Kerberos服务器 | KDC |
| linc-client | CentOS 6.5 desktop | 用户机 | Hadoop客户端、Hive客户端、HBase客户端、Storm客 户端、Spark客户端、Impala客户端 |

**5.路径说明**

本节中所有的Hadoop组件目录均放置于/var/local/hadoop/路径下。

**6.Realm** **说明**

本节中所使用的 Kerberos Realm 均使用 LINC.COM表示。

**7.3.1** **Kerberos** **服务配置**

Kerberos 能够运行在 UNIX Like(如Linux 、OS X等)平台上。由于本章内容主要介绍 将 Kerberos服务应用于Hadoop集群中的各个组件中，因此本章中的内容及操作全部基于 CentOS 6.5 平台。

**1.安装** **KDC**

建议Kerberos的服务端KDC选择一台单独的主机进行安装，这里暂且将其命名为 linc-krb。在 CentOS平台上，可以使用 yum命令直接对 Kerberos 服务端进行安装。

$sudo yum -y install krb5 krb5-server

清 yum 是 CentOS 系统预安装的一款 Shell 前端软件包管理器，用于从指定的服务器自 动下载软件的 RPM 包并且安装。本章中将会频繁使用该命令进行软件的安装。读

者也可以选用其他软件如apt-get来进行软件的安装。

**2.安装** **Kerberos 客户端**

Kerberos 的客户端需要安装在所有需要使用Kerberos协议进行加密传输的主机上，包 括用户所使用的主机。可以使用以下 yum命令直接对 Kerberos客户端进行安装。

S sudo yum -y install krb5-devel

第7章 大数据安全之Kerberos 认证 319

**3.配置Kerberos 服务端**

Kerberos 服务端的配置主要通过修改kdc.conf 文件进行，修改 linc-krb 上的 kdc.conf 文件。

$sudo vim /var/kerberos/krb5kdc/kdc.conf

将内容修改为以下内容，其中LINC.COM 是为集群 Kerberos协议所起的域名，本章后 续内容中也将使用LINC.COM在代码中指代该域名。由于kdc.conf 中可以使用的配置项较 多，此处只列出必须配置的以及较为重要的配置项。

[kdedefaults] 部分主要控制 KDC的整体行为：

[kdcdefaults]

kdc\_ports =88

#Kerberos 服务端监听的端口号，若填写多个则使用逗号隔开，默认值为88和750

kdc\_tcp\_ports =88

#Kerberos 服务端监听TCP连接的端口号，不建议修改，标准端口号为88。如果不进行配置则Kerberos 不

会对TCP连接进行监听

**[realms] 部分包含了Kerberos 中所有的域以及域参数：**

[realms]

LINC.COM ={

master\_key\_type =aes256-cts

#主密钥的密钥类型

acl\_file =/var/kerberos/krb5kdc/kadm5.acl

#指定访问控制列表的位置，此处仅作为重要配置项列出，在本章中实际没有用到

dict\_file =/usr/share/dict/words

#指定一个包含了不允许被设置为密码的字符串的文件路径

max\_renewable\_life =7d max life =1d

#指定域中的票据更新的最长周期

#指定域中的票据最长的有效期

admin\_keytab =/var/kerberos/krb5kdc/kadm5.keytab

#指定Kerberos 管理员用于认证KDC 数据库的keytab 文件路径

supported\_enctypes =aes256-cts:normal aes128-cts:normal des3-hmac-shal:normal arcfour-hmac:normal des-hmac-shal:normal des-cbc-md5:normal des-cbc-crc:normal

#指定域中的Principal 使用的密钥/盐值的组合形式，以 “key:salt” 的形式组合，使用空格分隔

4.创建 Kerberos 数据库

为了存储之后需要用到的Principal,需要在 linc-krb上以域名来创建 KDC 中的数据库。

s kdb5\_util create -r LINC.COM -s

#之后输入该数据库的密码，后文中将这个密码称为“域名数据库密码”

数据库的创建操作需要root 权限。若出现 Loading random data 并卡住的情况，可以 尝试另开一个终端执行 cat /dev/sda→/dev/urandom。

5.启动 Kerberos 服务

在 linc-krb上使用root 用户执行以下命令开启 Kerberos 服务：

**320** **第二篇** **开源实现篇**

$chkconfig --level 35 krb5kdc on

$chkconfig --level 35 kadmin on

S service krb5kdc start

$service kadmin start

**6.创建** **Kerberos 管理员**

为了管理KDC 中存储的Principal, 需要在KDC 上为Kerberos 建立管理员账号。在 linc-krb 上使用root 账号执行以下命令：

$kadmin.local -q "addprinc root/admin"

#输入管理员账号的密码，后文中将这个密码称为 “KDC管理员密码”

此处需要手动输入两次密码，密码不能为空，且需妥善保存。

**7.测试** **Kerberos 服务**

为了确认Kerberos 服务端的配置成功，可以尝试在KDC(linc-krb) 上以 root 用户执行

以下命令进行测试：

$kadmin.local

kadmin.local>addprinc userl

#提示输入密码，需要输入两次

kadmin.local>list\_principals

#如果能够正确列出所创建的Principal“user1” kadmin.local>exit

则说明服务配置成功，可以正常运行

**8.配置** **Kerberos 客户端**

Kerberos 客户端通过修改配置文件来进行配置，客户端的配置文件通常存放于/etc/ krb5.conf, 需要在所有安装了Kerberos 客户端的主机上使用root 用户对其进行修改配置。

$vim /etc/krb5.conf

将内容参考以下列出的配置项进行相应的修改，其中LINC.COM 为域名，linc-krb为

Kerberos 服务端的hostname。

[logging]部分包括关于日志的配置：

[lt =FILE:/var/log/krb5libs.log

kdc =FILE:/var/log/krb5kdc.log

admin\_server =FILE:/var/log/kadmind.log

[libdefaults] 部分包含了Kerberos V5库中的各项参数：

[libdefaults]

ul\_ lm\_real

dns\_lookup\_kdc =false

ticket\_lifetime =24h

第7章大数据安全之 Kerberos认证 32 1

renew\_lifetime =7d forwardable = true default\_tgs\_enctypes default\_tkt\_enctypes permitted\_enctypes

clockskew =120

udp\_preference\_limit

=aes256-cts-hmac-sha1-96

=aes256-cts-hmac-sha1-96

=aes256-cts-hmac-shal-96

=1

**[realms]部分包含了各个域的设置参数：**

[realms]

LI-krb

admin\_server =linc-krb

}

|  |  |
| --- | --- |
|  | 此处填写的linc-krb 是主机的hostname, 还需要在系统/etc/hosts 中对其他主机的 hostname 作出相应的定义才能够生效。如果没有配置 hosts文件，则此处可以填写 相对应的IP 地址。 |
| [domain\_realm] 部分包含了一组主机域名和Kerberos 域名的关系，用于让程序分辨某 个主机属于哪个域。使用全称域名。  [domain\_realm]  .example.com =LINC.COM  example.com =LINC.COM  **9.测试** **Kerberos 认证**  要测试 Kerberos的认证，需要在KDC 中创建新的Principal, 并将其导出为 keytab 文 件分发至客户端，并在客户端使用该 keytab 文件进行Kerberos 认证。  在 linc-krb 中，使用root 用户执行以下命令，创建Principal 并导出为 keytab:  $kadmin.local  kadmin.local >addprinc testuserl@LINC.COM  #输入该Principal 的密码，需要输入两次  kadmin.local >xst -k testuserl.keytab testuser1@LINC.COM  kadmin.local >exit  导出后的keytab 文件将会生成在用户当前目录，将其分发至客户端后，在客户端执行 以下命令进行 Kerberos 认证：  $kinit -kt testuserl.keytab  s klist  #如果认证成功则会列出当前已认证的Principal | |
|  | 此处应填写keytab 文件路径；由于客户端的keytab 是能够无须密码对 Principal 进 行认证的密钥文件，因此需要注意keytab文件的权限管理，在认证时要确保当前用 户拥有该文件的读权限。 |

322 第二篇 开源实现篇

**7.3.2** **HDFS 集** **成** **Kerberos 的安装与调试**

本节将基于5.2.3节中 HDFS的配置结果，介绍 Hadoop HDFS 服务与 Kerberos 的集成 配置，并实现以下功能：

1)没有经过Kerberos 认证的用户无法使用 HDFS 服务以及 Web UI。 2)经过 Kerberos 认证的用户可以正常使用HDFS 服务以及 Web UI。

本节中所使用的 Hadoop 版本为 Apache Hadoop 2.5.0,其余相关软件均选用了与该版 本相配的版本，在实际操作时请根据使用的 Hadoop 版本进行调整。

②素

由 于HDFS 与 YARN需要单独与Kerberos 进行集成配置，同时YARN组件与 Kerberos 的集成需要重新编译 Hadoop, 因此本节中会先给出编译 Hadoop 所需要 的环境准备以及编译过程，在编译完成后再根据5.2.3中的正常配置步骤对Hadoop HDFS 进行配置，最后再按本节中列出的步骤进行 Kerberos 相关配置。



**1.编译环境的安装配置**

(1)安装 Maven

Maven 是一个项目构建和管理的工具，提供了能够帮助管理构建、项目结构的功能， 可以方便地编译代码、进行依赖管理、管理二进制库等。

Hadoop项目使用了Maven 进行管理，因此编译 Hadoop需要使用Maven 来进行。编译 Apache Hadoop 2.5.0 所需要的Maven 版本是3.3.3,请自行下载后解压至合适的目录。

修改/etc/profile,为 Maven 添加路径：

MAVEN\_HOME=// 此处填写apache-maven-3.3.3 目录的完整路径

export MAVEN\_HOME

#在PATH 后加上

:S{MAVEN\_HOME}/bin

修改完成后可以执行以下命令确认是否安装完成：

$source /etc/profile

$mvn -version

#如果能够正常显示Maven 的软件版本则说明安装完成

(2)安装编译所需的依赖包

本节中使用CentOS 中自带的 yum命令进行安装，若需使用其他平台可自行安装相应 的依赖包。

$sudo yum -y install cmake gcc-c++glibc-header openssl-devel zlib-devel

(3)安装 Protobuf

Protobuf, 即 Protocol Buffer 。Protocol Buffer 是 Google 开源的一种轻便高效的结构化 数据存储格式，可以用于结构化数据序列化和反序列化，很适合用于数据存储或 RPC 数据 交换格式。简单理解就是某个进程把一些结构化数据通过网络通信的形式传递给另外一个

第7章 大数据安全之 Kerberos 认证 ◆ 323

进程(典型应用就是RPC), 而Hadoop 使用了Protocol Buffer作为RPC 的默认实现，因此

在编译 Hadoop 时也需要用到 Protocol Buffer。

编译 Apache Hadoop 2.5.0所需的 Protobuf 版本为2.5.0,可以根据以下命令下载并安装

Protobuf。

s wget <https://protobuf.googlecode.com/files/protobuf-2.5.0.tar.gz>

s tar -zxf protobuf-2.5.0.tar.gz

$mkdir protobuf

$cd protobuf-2.5.0

s./configure --prefix=//此处输入之前通过mkdir 创建的protobuf 目录的完整路径

等待运行完成后继续操作。

S make 66 make install

安装过程中可以根据 Log信息判断安装是否成功。

安装完成后修改/etc/profile,添加 Protobuf的路径：

#在PATH变量后加入上文protobuf 目录下bin 目录的完整路径

:/xxx/protobuf/bin

修改完成后可执行以下命令检测安装是否成功：

s source /etc/profile

$protoc

若出现提示 “Missing input file.” 则说明安装成功。

**2.编译** **Hadoop**

(1)下载 Hadoop 源码

本节中给出 Apache Hadoop 2.5.0的源码下载地址，请根据需要自行调整。

s wget <https://archive.apache.org/dist/hadoop/core/hadoop-2.5.0/hadoop-2.5.0-src.tar.gz> s tar -zxf hadoop-2.5.0-src.tar.gz

(2)编译

使用以下命令编译 Hadoop, 注意下文中的 executor.conf.dir 路径必须是一个所有父目 录所有者都为root 用户的目录，且权限小于755,具体说明请查看7.3.3节中的相关说明， 本章中使用/etc目录。

s cd ~/hadoop-2.5.0-src

S mvn package -Pdist,native -DskipTests -Dtar -Dcontainer-executor.conf.dir=/etc

等待编译完成，根据Log 判断编译是否成功。

(3)获取编译后的 Hadoop

编译完成后的 Hadoop 文件存放于以下地址：

hadoop-2.5.0-src/hadoop-dist/target/hadoop-2.5.0.tar.gz

**324** 第二篇 开源实现篇

**3.配** **置** **Hadoop**

请参照5.2.3节中给出的 Hadoop 集群安装配置说明进行配置。

**4.创建** **HDFS 的** **Principal**

在 KDC(linc-krb) 中 创 建HDFS 服务所使用的Principal 并 导 出 至keytab 文件。由于

一些组件中对于Kerberos 的实现问题，某些服务要求使用以HTTP 为开头的 Principal, 因

此需要创建 HDFS 以 及 HTTP 两种 Principal 。 使 用 root 用户执行以下命令：

$kadmin.local

kadmin.local>addprinc -randkey hdfs/linc-1@LINC.COM

kadmin.local>addprinc -randkey hdfs/linc-2QLINC.COM

kadmin.local>addprinc -randkey hdfs/linc-3@LINC.COM

kadmin.local>addprinc -randkey HTTP/linc-1@LINC.COM

kadmin.local>addprinc -randkey HTTP/linc-2@LINC.COM

kadmin.local>addprinc -randkey HTTP/linc-3@LINC.COM

kadmin.local>xst -k hadoop.keytab hdfs/linc-1@LINC.COM

kadmin.local>xst -k hadoop.keytab hdfs/linc-2ELINC.COM

kadmin.local>xst -k hadoop.keytab hdfs/linc-3@LINC.COM

kadmin.local>xst -k HTTP.keytab HTTP/linc-1QLINC.COM

kadmin.local>xst -k HTTP,keytab HTTP/linc-2@LINC.COM

kadmin.local>xst -k HTTP.keytab HTTP/linc-3@LINC.COM

kadmin.local>exit

操作成功后，可以在当前目录下看到 hadoop.keytab 与 HTTP.keytab 两个文件。

**5.合并** **keytab**

为了对keytab 中 Principal 进行更好的管理，建议将某个服务所使用的Principal 通过 Kerberos 合并工具合并到一个keytab 中 ，HDFS 需要使用的Principal是 hdfs与 HTTP 两个， 因此使用 ktutil 进行合并。

$ktutil

ktutil>rkt hadoop.keytab

ktutil>rkt HTTP.keytab

ktutil>wkt hadoop.keytab

ktutil>exit

合并后可以使用以下命令确认 hadoop.keytab 中包含了hdfs 与 HTTP 两个 Principal:

s klist -kt hadoop.keytab

**6.分** **发** **keytab 文** **件**

将 keytab 分发至所有 Hadoop 的 服 务 端(linc-1 、linc-2 、linc-3), 建议使用scp 命令来

远程拷贝 keytab 文件。

建议对 keytab 文件进行适当的权限设置，以避免出现安全问题。

7.修改 Hadoop 配置文件

要 让 Hadoop HDFS 使用 Kerberos 协议，需要在配置文件中开启相应的配置项。修改

第7章 大数据安全之Kerberos 认证 325

core-site.xml 与 hdfs-site.xml 文件，确保以下配置项的值。

**core-site.xml:**

<property>

<name>hadoop.security.authentication</name>

<value>kerberos</value>

</property>

<property>

<name>hadoop.security.authorization</name>

<value>true</value>

</property>

core-site.xml 部分主要为了开启 Hadoop 核心模块中的安全认证机制，并指定Kerberos 认证。配置项说明如表7-4所示。

**表7-4** **core-site.xml 配置项说明**

core-site.xml

|  |  |  |
| --- | --- | --- |
| **配置项名称** | **建议配置值** | **含** **义** |
| hadoop.security.authentication | Kerberos | Hadoop安全认证模式，可用的值为simple(无安全认证模式)和kerberos |
| hadoop.security.authorization | true | 是否开启Hadoop安全认证 |

**hdfs-site.xml:**

<property>

<name>dfs.namenode.keytab.file</name>

<value>hadoop.keytab</value>

</property>

<property>

<name>dfs.namenode.kerberos.principal</name>

<value>hdfs/\_HOST@LINC.COM</value>

</property>

<property>

<name>dfs.namenode.kerberos.https.principal</name>

<value>HTTP/\_HOSTeLINC.COM</value>

</property>

以上部分配置了Hadoop HDFS 中 NameNode 使用的 Principal 和 Keytab 文件路径。配

置项说明如表7-5所示。

**表7-5** **NameNode 部分配置项说明**

hdfs-site.xml

|  |  |  |  |
| --- | --- | --- | --- |
| **配置项名称** | **建议配置值** | **含** **义** | **约束关系及格式要求** |
| dfs.namenode. keytab.file | /var/local/hadoop/ hadoop-2.5.0/etc/keytabs/ hadoop.keytab | Name Node上开启服 务所使用的Principal所 在的keytab文件路径 | 需要保证在NameNode上该keytab文 件中存在dfs.namenode.kerberos.principal 和dfs.namenode.kerberos.https.principal 中填写的Principa |
| dfs.namenode. kerberos.principal | hdfs/\_HOST@LINC. COM | Name Node上开启服 务所使用的Principa | HOST变量会自动替换为启动服务时 主机的hostname |

**326**  **第二篇** **开源实现篇**

(续)

|  |  |  |  |
| --- | --- | --- | --- |
| **配置项名称** | **建议配置值** | **含** **义** | **约束关系及格式要求** |
| **dfs.namenode.** **kerberos.https.** **principal** | **HTTP/\_HOST@LINC.** **COM** | **NameNode上https所** **使用的Principa** | **\_HOST变量会自动替换为启动服务时** **主机的hostname** |

**hdfs-site.xml:**

<property>

<name>dfs.datanode.keytab.file</name>

<value>hadoop.keytab</value>

</property>

<property>

<name>dfs.datanode.kerberos.principal</name>

<value>hdfs/\_HosTeLINC.coM</value>

</property>

<property>

<name>dfs.datanode.kerberos.https.principal</name>

<value>HTTP/\_HOST@LINC.COM</value>

</property>

以上部分配置了 Hadoop HDFS中 DataNode 使用的 Principal 和 keytab 文件路径。配置

项说明如表7-6所示。

**表7-6** **DataNode 部分配置项说明**

hdfs-site.xml

|  |  |  |  |
| --- | --- | --- | --- |
| **配置项名称** | **建议配置值** | **含** **义** | **约束关系及格式要求** |
| dfs.datanode. keytab.file | /var/local/hadoop/  hadoop-2.5.0/etc/keytabs/ hadoop.keytab | Data Node上开启服 务所使用的Principal所 在的keytab文件路径 | 需要保证在DataNode上该keytab文件  中存在dfs.datanode.kerberos.principal和 dfs.datanode.kerberos.https.principal中填 写的Principa |
| dfs.datanode. kerberos.principal | hdfs/\_HOST@LINC.  COM | DataNode上开启服 各所使用的Principa | HOST变量会自动替换为启动服务时 主机的hostname |
| dfs.datanode. kerberos.https. principal | HTTP/\_HOST@LINC COM | DataNode上https所 使用的Principa | HOST变量会自动替换为启动服务时 主机的hostname |

**hdfs-site.xml:**

<property>

<name>dfs.webhdfs.enabled</name>

<value>true</value>

</property>

<property>

<name>dfs.web.authentication.kerberos.principal</name>

<value>HTTP/\_HOST@LINC.COM</value>

</property>

<property>

<name>dfs.web.authentication.kerberos.keytab</name>

第7章 大数据安全之 Kerberos 认证 327

<value>HTTP.keytab</value>

</property>

以上部分配置了Hadoop HDFS中 Web UI使用的 Principal 和 keytab 文件路径。配置项 说明如表7-7所示。

**表7-7** **Web UI 部分配置项说明**

hdfs-site.xml

|  |  |  |  |
| --- | --- | --- | --- |
| **配置项名称** | **建议配置值** | **含** **义** | **约束关系及格式要求** |
| dfs.webhdfs.enabled | true | 是否开启webhdfs |  |
| dfs.web.authentication kerberos.principal | HTTP/\_HOST@LINC. COM | webhdfs服务使用 的Principa | 建议使用HTTP开头的principal, HOST变量会自动替换为启动服务  时主机的hostname |
| dfs.web.authentication kerberos.keytab | /yar/local/hadoop/  hadoop-2.5.0/etc/keytabs/ hadoop.keytab | webhdfs服务使用 的Principal所在的 keytab文件路径 | 需要保证在该keytab文件中包  含dfs.web.authentication.kerberos principal中填写的Principal |

在大部分 Hadoop 组件的配置文件中，对于Principal 的配置项都可以使用\_HOST 来 指代主机的 hostname。

**8.配置** **JSVC**

JSVC 是 Java 应用的辅助程序，其中包含了一定数量的库和应用程序，能够使一些 Java 程序使用 root 权限进行一些需要权限的操作，并进行用户之间的切换。

由于HDFS 中 的 DataNode 在配置了Kerberos 后需要root 权限进行操作，因此需要安 装JSVC 来辅助运行。

(1)下载并解压 JSVC

s wget <http://apache.fayea.com//commons/daemon/binaries/commons-daemon-1.0.15-bin>. tar.gz

s wget <http://apache.fayea.com//commons/daemon/source/commons-daemon-1.0.15-src>.

tar.gz

$tar zxvf commons-daemon-1.0.15-src.tar.gz

$tar zxvf commons-daemon-1.0.15-bin.tar.gz

**(2)编译生成** **JSVC**

在 commons-daemon-1.0.15-src/src/native/unix/目录下，执行以下命令进行编译：

$./configure

S make

(3)分发JSVC

需要将编译后的 JSVC 文件拷贝至所有安装了Hadoop 主机的相应目录下：

$cp jsvc hadoop-2.5.0/libexec

#此处的hadoop-2.5.0 需根据实际情况替换为Hadoop目录的完整路径

328 第二篇 开源实现篇

(4)分发 JSVC 依赖包

JSVC 的启动需要依赖于commons-daemon 包，因此需要相对应版本的jar 包。将上文 中下载的 commons-daemon-1.0.15-bin.tar.gz 文件解压后，将目录中的 commons-daemon-1. 0.15.jar拷贝至所有安装了Hadoop 主机的相应目录下，同时删除相应目录下的旧版本jar 包。

$cp commons-daemon-1.0.15.jar hadoop-2.5.0/share/hadoop/hdfs/lib/

$rm hadoop-2.5.0/share/hadoop/hdfs/lib/commons-daemon-\*.jar

#此处的hadoop-2.5.0 需根据实际情况替换为Hadoop目录的完整路径

**9.修改** **Hadoop 配置文件**

修改 hadoop-2.5.0/etc/hadoop/hadoop-env.sh 文件，确保以下配置项。配置项说明如表7-8 所示。

export HADOOP\_SECURE\_DN\_USER=hdfs

export JSVC\_HOME=hadoop-2.5.0/libexec

**表7-8** **hadoop-env.sh 部分配置项说明**

**hadoop-env.sh**

|  |  |  |
| --- | --- | --- |
| **配置项名称** | **建议配置值** | **含** **义** |
| HADOOP\_SECURE\_DN\_USER | hdfs | 启动DataNode安全模式的用户(由于启动datanode  安全模式必须使用root用户，启动后会将启动用户变更 为这个配置项所填写的用户) |
| JSVC\_HOME | /var/local/hadoop/ hadoop-2.5.0/libexec | JSVC所在目录 |

|  |  |
| --- | --- |
|  | HADOOP\_SECURE\_DN\_USER 配置项指定了安全模式下开启Hadoop DataNode 的 用户，根据本章中的用户规划，此处使用hdfs 用户，可以根据实际情况修改。 |
| **10.启动** **HDFS**  在配置了 Kerberos 或其他安全认证机制的情况下，启动HDFS 的步骤如下，请确保运 行的用户拥有 root 权限。  $start-dfs.sh  #启动NameNode, 与正常情况下一样  $sudo start-secure-dns.sh  #启动所有slaves上的DataNode | |

 此处默认配置了/etc/profile 中的 Hadoop 相关路径，如果没有配置请替换为完整路径。 运行后可以使用jps 命令来查看当前主机上正在运行的服务，确保NameNode 和

DataNode 正常启动

第 7 章 大数据安全之Kerberos认 证 329

配置了安全认证机制后的 DataNode, 使 用jps 命令查看时只显示端口号，进程名被 隐藏。

**11.测试** **HDFS**

在客户机上，使用测试用户，先在没有经过Kerberos 认证的情况尝试操作 HDFS。

s kdestroy

$hdfs dfs -ls /

操作失败，如图7-5所示。没有经过Kerberos 认证的用户无法对 HDFS 进行操作。



**testuserelinc-elient -lS kdestroy**

tt0ue/r1ei-:cit18 W-1sARdfiePc. fienEs xcption encountered while connecting to the server

:javax.security,sasl.SaslException:Gss initiate failed ICaused by GSSException:No val id credentials provided (Mechanism level:Failed to find any Kerberos tgt)1

ls:Failed on local exception:java.io.IOException:javax.security.sasl.SaslException:G ss initiate failed [Caused by GSsException:No valid credentials provided(Mechanism lev el:Failed to find any Kerberos tgt)l;Host Details :local host is:"line-client/192.16 8.140.142";destination host is:"linc-1":9000;

Itestuserelinc-client ~1S

图7-5 没有经过 Kerberos 认证的失败信息

**进行** **Kerberos认证，再进行** **HDFS 的操作。**

$kinit -kt testuser.keytab testuser@LINC.COM

$klist

#确认获取到了ticket

$hdfs dfs -ls /

操作成功，如图7-6所示。可以正常操作HDFS。



/ete/profile

-1s /

02017-03-1816:34 /testdirectory

92017-03-1816:33 /testfile.txt

|  |  |
| --- | --- |
| [testuserQlinc-client [testuserQlinc-client Found 2 items  drwxr-xr-x -hdfs | ~1S source ~1S hdfs dfs  supergrour |

-IW-I--T-- 2 hdfs supergroup

图7-6 经过Kerberos 认证后正常操作的情况

**12.测试** **HDFS Web UI**

Hadoop Web UI在开启了Kerberos 验证后也同样需要Kerberos 的票据才能够访问。由 于目前主流浏览器不会自动将 Kerberos 的票据提交至Web 页面，因此需要针对不同的浏览

器做出不同的调整。本节中使用客户机上安装的 Firefox 浏览器。

1)在客户机上打开Firefox 浏览器，在地址栏中输入about:config, 进入设置页面。 2)在顶部的 search 栏中输人 network.negotiate-auth.trusted-uris 搜索该配置项。

3)双击行，在对话框中输入需要浏览器提交票据的url, 可以使用逗号分隔多个 url。

配置完成后，先在没有认证的情况下，访问NameNode 的 Web UI, 默认端口为50070,

会显示 “HTTP ERROR 401 Authentication required”。

330 **第二篇** **开源实现篇**

进入Terminal, 进 行 Kerberos 认证：

s kinit -kt testuser.keytab testuser@LINC.COM

s klist

#确认获取到了票据

再访问NameNode Web UI, 即可正常访问。

注 当访问成功后，如果使用 kdestory 注销认证，会因为浏览器缓存的原因，在一段时间

意

内可以继续访问需要Kerberos 认证的网页。可以通过清空浏览器缓存来解决这个问题。

**7.3.3 YARN 集成** **Kerberos 的安装与调试**

本节将基于7.3.2节的配置结果，继续对Hadoop YARN 进行 Kerberos 的集成配置。

**1.创建** **YARN 服务使用的** **Principal**

根据 Kerberos的使用规范，需要为YARN服务创建Principal, 在 YARN 服务的运行过 程中，需要yarn 和mapred 两个 Principal, 在KDC(linc-krb) 上使用root用户创建Principal。

|  |  |
| --- | --- |
| $kadmin.local |  |
| kadmin.local>addprinc | -randkey yarn/linc-1@LINC.COM |
| kadmin.local>addprinc | -randkey yarn/linc-2@LINC.COM |
| kadmin.local>addprinc | -randkey yarn/linc-3@LINC.COM |
| kadmin.local>addprinc | -randkey mapred/linc-18LINC.COM |
| kadmin.local>addprinc | -randkey mapred/linc-2@LINC.COM |
| kadmin.local>addprinc | -randkey mapred/linc-3QLINC.COM |
| kadmin.local>xst -k | yarn.keytab yarn/linc-1@LINC.COM |
| kadmin.local>xst -k | yarn.keytab yarn/linc-2@LINC.COM |
| kadmin.local>xst -k | yarn.keytab yarn/linc-3@LINC.COM |
| kadmin.local>xst -k | mapred.keytab mapred/linc-18LINC.COM |
| kadmin.local>xst -k | mapred.keytab mapred/linc-2@LINC.COM |
| kadmin.local>xst -k  kadmin,local>exit | mapred.keytab mapred/linc-3QLINC.COM |

操作成功后，可在该目录下看到 yarn.keytab 与 mapred.keytab 两个文件。

**2.合并** **keytab**

由于一些服务的配置文件 (XML) 中会默认一个服务中使用的多个 Principal 来源于一 个统一的配置项指向的keytab文件，因此建议将一个服务中使用的多个 Principal 进行合 并。在KDC(linc-krb)上使用root用户进行合并：

$ktutil

ktutil>rkt mapred.keytab

ktutil>wkt yarn.keytab

ktutil>exit

3.分 发 Keytab 文 件

将 keytab 分发至所有 Hadoop 的服务端 (linc-1 、linc-2 、linc-3), 建议使用 scp 命令来

第7章 大数据安全之Kerberos 认证

331

远程拷贝 keytab 文件。

**4.修改** **YARN 配置文件**

确保各个Hadoop 服务端节点配置文件的内容如下，建议修改后统一分发。

**core-site.xml:**

<property>

<name>hadoop.http.authentication.signature.secret.file</name>

<value>/var/local/hadoop/hadoop-http-auth-signature-secret</value> </property>

配置项说明如表7-9所示。

**表7-9** **core-site.xml 部分配置项说明**

core-site.xml

|  |  |  |  |
| --- | --- | --- | --- |
| **配置项名称** | **建议配置值** | **含** **义** | **约束关系及格式要求** |
| **hadoop.http.authen-** **tication.signature.** **secret.file** | **/var/local/hadoop/** **hadoop-http-auth-** **signature-secret** | **hadoop-http-auth-**  **signature-secret文件所** **在目录** | **默认值为${user.home}/hadoop-http-auth-** **signature-secret,文件需要自己创建并且赋** **予启动服务的用户读权限** |

hadoop.http.authentication.signature.secret.file 配置项是 Hadoop 安全模式下使用的 一个配置项，该配置项指向了一个文件， Hadoop 允许文件中所指明的用户访问 Hadoop Web UI。该机制属于Hadoop自身的一套安全机制，与本章配置Kerberos 的安全机制不相符合，因此这里将路径指向一个空文件(文件仍然需要存在)。

**yarn-site.xml:**

<property>

<name>yarn.nodemanager.container-executor.class</name>

<value>org.apache.hadoop.yarn.server.nodemanager.LinuxContainerExecutor</ value>

</property>

<property>

<name>yarn.nodemanager.linux-container-executor.group</name>

<value>hadoop</value>

</property>

<property>

<name>yarn.resourcemanager.webapp.delegation-token-auth-filter.enabled</name>

<value>false</value>

</property>

<property>

<name>hadoop.http.authentication.type</name>

<value>kerberos</value>

</property>

<prrmty

<value>org.apache.hadoop.security.AuthenticationFilterInitializer</value> </property>

**332** **第二篇** 开源实现篇

yai )n' 配置项的值需要与 .cfg 文件中

计

意

的 yarn.nodemanager.linux-container-executor.group 值一致，同时需要确保 yarn. nodemanager.local-dirs 和 yarn.nodemanager.log -dirs 所指向的目录所属用户组与该值 一致，且这两个目录的权限至少需要 drwxr-xr-x。

以上配置项指定了YARN 所使用的安全认证机制以及具体的安全认证类等配置信息。 配置项说明如表7-10所示。

**表7-10** **yarn-site.xml 部分配置项说明**

yarn-site.xml

|  |  |  |
| --- | --- | --- |
| **配置项名称** | **建议配置值** | **含** **义** |
| yarn.nodemanager.container- executor.class | org.apache.hadoop.yarn.server. nodemanager.LinuxContainerExecutor | container的启动者 |
| yarn,nodemanager.linux-container executor.group | hadoop | nodemanager上container所属用 户组 |
| yarn.resourcemanager.webapp. delegation-token-auth-filter.enabled | false | 是否用RM认证过滤来覆盖默认的 Kerberos认证过滤器，使得允许使用 授权令牌认证(如果令牌丢失则会自 动退回Kerberos认证) |
| hadoop.http.authentication.type | kerberos | Hadoop中http服务使用的安全认 证模式 |
| hadoop.http.filter.initializers | org.apache.hadoop.security.Authen- ticationFilterInitializer | Hadoop中http使用的过滤器设置 |

**yarn-site.xml**

<property>

<name>yarn.resourcemanager.keytab</name>

<value>yarn.keytab</value>

</property>

<property>

<name>yarn.resourcemanager.principal</name>

<value>yarn/\_HOST@LINC.COM</value>

</property>

<property>

<name>yarn.nodemanager.keytab</name>

<value>yarn.keytab</value>

</property>

<property>

<name>yarn.nodemanager.principal</name>

<value>yarn/\_HOST@LINC.COM</value>

</property>

<property>

<name>hadoop.http.authentication.kerberos.principal</name>

<value>HTTP/\_HOST@LINC.COM</value>

</property>

第7章 大数据安全之Kerberos 认证 333

<property>

<name>hadoop.http.authentication.kerberos.keytab</name>

<value>HTTP.keytab</value>

</property>

以上配置项指定了YARN所使用的Principal 以及 Principal 所在的 keytab 文件路径。 配置项说明如表7-11所示。

**表7-11** **yarn-site.xml 部分配置项说明**

**yarn-site.xm**

|  |  |  |  |
| --- | --- | --- | --- |
| **配置项名称** | **建议配置值** | 含 义 | **约束关系及格式要求** |
| yarn.resource- manager.keytab | /var/local/hadoop/  hadoop-2.5.0/etc/keytabs/ yarn.keytab | resourcemanager服务所使 用的Principal所在的keytab 文件路径 | 需要保证在该keytab文件中包  含yarn.resourcemanager.principal 中填写的Principal |
| yarn.resource- manager.principal | yarn/\_HOST@LINC.  COM | resourcemanager服务所 使用的Principa | HOST变量会自动替换为启动 服务时主机的hostname |
| yarn.nodeman- ager.keytab | /var/local/hadoop/  hadoop-2.5.0/etc/keytabs/ yarn.keytab | nodemanager服务所使用 的Principal所在的keytab 文件路径 | 需要保证在该keytab文件中包  含yarn.nodemanager.principal中填 写的principa |
| yarn.nodeman- ager.principal | yarn/\_HOST@LINC.  COM | nodemanager服务所使用 的Principa | HOST变量会自动替换为启动 服务时主机的hostname |
| hadoop.http.  authentication. kerberos.principal | HTTP/\_HOST@LINC.  COM | http通信所使用的Prin- cipal | HOST变量会自动替换为启动 服务时主机的hostname |
| hadoop.http  authentication. kerberos.keytab | /var/local/hadoop/  hadoop-2.5.0/etc/keytabs/ HTTP.keytab | http通信所使用的Prin-  cipal所在的keytab文件 路径 | 需要保证在该keytab文件中包  含hadoop.http.authentication.  kerberos.principal中填写的Principal |

**5.创建container-executor 配置文件**

container-executor 配置文件是 Hadoop安全模式下的机制，文件主要用于配置启动 Container的用户，并做出限制。

此处也是上一节中重新编译 Hadoop的原因，由于Hadoop 要求 container-executor 配 置文件的所有父目录均要属于root 用户，而官网下载的编译后的版本默认将路径设置为了 hadoop目录/etc/hadoop/container-executor.cfg。

考虑到 Hadoop目录的权限设置问题，所以在上一节中，在重新编译 Hadoop 时重定义 了该文件的路径。

因此这里需要在编译时指定的位置创建 container-executor 配置文件。

s sudo vim /etc/container-executor.cfg

添加以下内容，请严格按照格式输入：

#运行container 的用户

yarn.nodemanager.linux-container-executor.group=hadoop

#这个是不允许运行应用的用户列表，默认是全部可以运行

**334** **第二篇** **开源实现篇**

banned.users=banneduser

#这个是允许提交job的最小的userid 的值。CentOS中一般用户的id 在500以上

min.user.id=500

**6.分发container-executor 配置文件并设置权限**

将 container-executor配置文件分发至所有 Hadoop 服务端节点，并放置到编译时指定 的路径，建议使用 scp命令进行传输。

分发完成后，需要设置每个Hadoop服务端节点上的 container-executor 和 container- executor.cfg 两个文件的权限。

s sudo chown root:hadoop hadoop目录/bin/container-executor

S sudo chmod 4750 hadoop目录/bin/container-executor

$sudo chown root:root /etc/container-executor.cfg

9 sudo chmod 600 /etc/container-executor.cfg

**操作成功后，可以看到两个文件的权限如下：**

-rwsr-x---root hadoop container-executor

-rw-------root root container-executor.cfg

**7.创建** **Hadoop-secret 文件**

**在上面core-site.xml中定义过的** **hadoop-http-auth-signature-secret 文件路径下创建该文**

件，不需要输入内容，修改权限。

$vim /var/local/hadoop/hadoop-http-auth-signature-secret

#保存退出

$sudo chmod 440 /var/local/hadoop/hadoop-http-auth-signature-secret

**8.测试准备**

在测试前，需要确保 Hadoop 没有在安全模式下，并且确保HDFS 上目录的写权限，由 于现在已经配置了Kerberos, 因此在操作HDFS 时需要使用HDFS 上的管理级用户 hdfs 来 进行这些设置。然而在设置了Kerberos认证下的 HDFS 不再将当前的系统用户作为HDFS 用户，而是将当前 Principal中的前缀作为HDFS中的操作用户，例如使用hdfs/linc-1@ LINC.COM这个Principal时，HDFS的操作用户将会是 hdfs, 而不是 hdfs/inc-1。

因此，这里需要使用拥有hdfs前缀的 Principal进行认证，例如hdfs/linc-1@LINC.COM。 在 linc-1 上，使用 hdfs 用户进行以下操作：

$kinit -kt hadoop.keytab hdfs/linc-1eLINC.COM

s hdfs dfs -chmod -R 777 /

$hdfs dfsadmin -safemode leave

并确保以下目录的权限：

$sudo chmod-R 775 /var/local/hadoop/hadoop-2.5.0/tmp

完成后在 linc-2 和 linc-3 上使用 root 用户进行以下操作。

第7章 大数据安全之Kerberos 认证 335

注意

1)确保/var/local/hadoop/hadoop-2.5.0/logs/userlogs 的拥有者为 yarn, 如果为 hdfs, 则 删除即可，如果不存在该目录则跳过。

2)确保/var/local/hadoop/hadoop-2.5.0/tmp/nm-local-dir的拥有者为 yarn, 如果为 hdfs, 则删除即可，如果不存在该目录则跳过。

3)确保/var/local/hadoop/hadoop-2.5.0/tmp 目录拥有用户组的写权限。

s sudo chmod -R 775 /var/local/hadoop/hadoop-2.5.0/tmp

由于上文中可能曾经使用hdfs用户来开启YARN 服务，因此这里需要确保这些目录 的所有者。



**9.启动** **YARN 服务**

在 linc-1 上使用 yarn 用户进行以下操作：

s cd /var/local/hadoop/hadoop-2.5.0/sbin

$./start-yarn.sh

操作成功后，稍等一会后输入以下命令：

$jps

可以看到 ResourceManager 和 NodeManager 已启动。

切换到 linc-2 、linc-3 的 yarn 用户，输入以下命令：

$jps

可以看到 NodeManager 已启动。

**10.启动JobHistory 服务**

在 linc-1 上使用 yarn 用户进行以下操作：

s cd /var/local/hadoop/hadoop-2.5.0/sbin

§./mr-jobhistory-daemon.sh start historyserver

操作成功后，稍等一会后输入以下命令：

$jps

可以看到 JobHistoryServer 已启动。

**11.创建** **Principal 相关用户**

上文中曾经提到， Hadoop 在 Kerberos 验证机制下，运行时 Hadoop 内部用户会使用 Principal 的前缀，这导致了在使用YARN的过程中，任务的提交者也将会是 Principal 的前 缀，而安全机制对Container 的启动者存在限制，因此需要保证在所有的NodeManager 节 点上都存在与 Principal 前缀相同的系统用户。

这里需要在客户机上使用 testuser@LINC.COM 进行测试，因此需要在所有 NodeManager

336 第二篇 开源实现篇

节点上创建testuser 用户。

**12.测试提交** **Mapreduce 任务**

在客户机上使用 testuser 用户，执行以下命令：

s kdestroy

s yarn jar /home/testuser/hadoop-2.5.0/share/hadoop/mapreduce/hadoop-mapreduce- examples-2.5.0.ja r pi 10100

操作失败，提示 “No valid credentials provided” 以 及 “Failed to find any Kerberos tgt”, 即没有通过 Kerberos 认证的情况下无法使用。

9 kinit -kt /home/testuser/keytabs/testuser.keytab testuser@LINC.COM

9 yarn jar /home/testuser/hadoop-2.5.0/share/hadoop/mapreduce/hadoop-mapreduce- examples-2.5.0.ja r pi 10100

成功执行 mapreduce 任务，作业正常运行，如图7-7所示。



Bb: map 20\*reduce s

mapreduce.1ob: map 20\*reduce 7%

mapreduce.Job: map 30%reduce 7%

mapreduce.Job: map 5%reduce 7%

mapreduce.Job:map 60%reduce 7%

mapreduce.Job: map 100%reduce 7%

mapreduce,Job:1ob job 14646608492418001 cbmpleted suecessfully

RECE

16/06/0110:08:24

16/06/0110:08:38 16/06/0110:11:27

16/06/0110:11:30 16/06/0110:11:32

16/06/0110:11:35 16/06/0110:11:58 16/06/0110:11:51

mapreduce.Job:map 100%reduce 108%

INFO INFO INF0 INFO INFO INFO INFO

INF0

图 7 - 7 经 过 Kerberos 认 证 后 的 正 常 操 作 情 况

**13.Web** **UI测试**

测试 Web UI 时对浏览器的配置操作与7.3.2节中的步骤12一样，在没有进行 Kerberos 认证的情况下，访问 YARN的 Web UI:ResourceManager http(linc-1:8088) 、NodeManager http(linc-2:8042 、linc-3:8042) 和 JobHistoryServer http(linc-1:19888), 将会访问失败， 出现 “HTTP ERROR401” 的页面，显示 “Authentication required.”。

在经过了 Kerberos 认证后，即可成功访问，显示Cluster 管理页面。

**7.3.4** **Hive** **集** **成** **Kerberos** **的配置与调试**

通过阅读前面几章可以知道， Apache Hive 需要以 MySQL 作为元数据库，因此本节将 基于5.2.1节中MySQL 的配置以及5.2.6节中Hive 的配置结果，介绍 Apache Hive 服务与 Kerberos 的集成配置。本节中所使用的 Apache Hive 版本为 Apache Hive 1.2.1, 在实际操作 时请根据使用的 Hive 版本进行调整。

土计 意

o

由于 Apache Hive server2的配置部分与基础配置有所不同，因此本节将首先介绍 HiveServer2 的相关配置，之后再随本节中列出的步骤进行Kerberos 的相关配置。

**1.修改** **Hive 配置文件**

要让 Apache Hive 开启 server2 服务，需要在配置文件中开启相应的配置项。修改 hive-

第7章 大数据安全之 Kerberos 认证 337

**site.xml 文件，确保配置项的值如下：**

<property>

<name>hive.server2.thrift.port</name>

<value>10000</value>

</property>

<property>

<name>hive.server2.thrift.bind.host</name>

<value>linc-1</value>

</property>

<property>

<name>hive.server2.enable.doAs</name>

<value>true</value>

</property>

**2.创建** **Hive 服务使用的** **Principal**

根 据 Kerberos 的 使 用 规 范 ， 需 要 为Hive 服 务 创 建 相 应 的Principal, 在 Hive 的 metastore 和 server2 服务运行过程中，只需要一个Hive 的 Principal, 同时本书中 Hive 为 单节点模式，因此只需要为linc-1 节点创建一个 Hive Principal。在 KDC(linc-krb) 中使用 root 用户执行以下命令，来创建Hive 服务需要使用的Principal 与 keytab 文件：

$kadmin.local

kadmin.local>addprinc -randkey hive/linc-1@LINC.COM

kadmin.local>xst -k hive.keytab hive/linc-1@LINC.COM

kadmin.local>exit

操作成功后，可以在当前目录下看到 hive.keytab 文件。

**3.分发keytab 文件**

同上文中的相应操作，将keytab 分发至Hive 的服务端 (linc-1), 建议使用 scp 命令来

远程拷贝 keytab 文件。

慧 建 议 对 keytab 文件进行适当的权限设置，以避免出现安全问题。

**4.修改** **Hive** **配置文件**

在 Hive 服务端节点，确保 hive-site.xml 配置文件内容如下：

<property>

<name>hive.server2.authentication</name>// 这个配置项指定了server2使用的身份认证方式 <value>kerberos</value>

</property>

<property>

<name>hive.server2.authentication.kerberos.principal</name>

<value>hive/\_HOST@LINC.COM</value>

</property>

<property>

<name>hive.server2.authentication.kerberos.keytab</name>

**338** **第二篇** **开源实现篇**

<value><!--- 这里填写server2 使用的keytab

</property>

<property>

<name>hive.metastore.sasl.enabled</name>

<value>true</value>

</property>

<property>

文件的完整路径---></value>

// 这个配置项指定了是否开启metastore 的sasl

<name>hive.metastore.kerberos.keytab.file</name>

<value><!--- 这里填写metastore 使用的keytab 文件的完整路径---></value>

</property>

<property>

<name>hive.metastore.kerberos.principal</name>

<value>hive/\_HOST@LINC.COM</value>

</property>

其中各配置项的参数说明及建议配置值如表7-12所示。

**表7-12** **hive-site** **配置文件参数说明**

hive-site.xml

|  |  |  |  |
| --- | --- | --- | --- |
| **配置项名称** | **建议配置值** | 含 义 | **约束关系及格式要求** |
| hive.server2.authen- tication | kerberos | Hive server2使用的 安全认证机制 |  |
| hive.server2.authen- tication.kerberos.prin- cipal | hive/\_HOST@LINC.COM | Hive server2使用的 Principal身份 | 需要与keytab中的Principa 一致，格式为{服务名}/{主  机 名 } @ { 域 名 |
| hive.server2.authen- tication.kerberos.keytab | /var/local/hadoop/hive-1.2.1/ conf/keytabs/hive.keytab | Hive server2使用的 keytab | 指向hive.keytab文件 |
| hive.metastore.sasl enabled | true | 是 否 开 启 H i v e 的 metastore的sasl认证 | true开启，false关闭 |
| hive.metastore.ker- beros.keytab.file | /var/local/hadoop/hive-1.2.1/ conf/keytabs/hive.keytab | Hive metastore使用 的Principal身份 | 指向hive.keytab文件 |
| hive.metastore.ker- beros.principal | hive/\_HOST@LINC.COM | Hive metastore使用 的keytab | 需要与keytab中的Principal 一致，格式为{服务名}/{主  机 名 } @ { 域 名 |

**5.修改相应Hadoop 目录下** **core-site.xml 配置文件**

由于Apache Hive 对 Kerberos集成的实现同时需要Hadoop 服务的支持，因此需要对 Hadoop的core-site.xml 配置文件进行相应的修改。添加的配置项内容如下：

<property>

<name>hadoop.proxyuser.hive.hosts</name>

<value>\*</value>

</property>

<property>

<name>hadoop.proxyuser.hive.groups</name>

<value>\*</value>

</property>

**第7章** **大数据安全之** **Kerberos 认证** 339

<property>

<name>hadoop.proxyuser.hdfs.hosts</name>

<value>\*</value>

</property>

<property>

<name>hadoop.proxyuser.hdfs.groups</name>

<value>\*</value>

</property>

<property>

<name>hadoop.proxyuser.HTTP.hosts</name>

<value>\*</value>

</property>

<property>

<name>hadoop.proxyuser.HTTP.groups</name>

<value>\*</value>

</property>

以上配置项的作用是：将hdfs 、HTTP 以及Hive 的 proxy 功能授权用户设置为所有人， 读者可根据需要更改<value> 内的用户名。

**6.Hive server2 与** **metastore 服务启动**

metastore 服务组件，即元数据服务组件，该组件用来存储Hive 的元数据，而Hive 的 元数据存储在关系数据库里，本书中Hive 所使用的关系数据库为MySQL 。Hive server2 服 务是Hive server的改进版本， server2 相比 server 而言更加稳定，支持的功能更多，本书中 所使用的 Hive既包含server 也包含 server2 服务，但这里以 server2 服务为例进行测试。

在linc-1 节点需要对配置Kerberos 的 Hive 相应服务重新启动，通过hive 用户关闭原 先服务后通过以下命令重启服务：

S hive --service metastore

s hive --service hiveserver2

以上命令为 Hive server2 和 Hive metastore的启动命令，但是该命令会在前端启动相 应服务，导致终端无法进行其他操作。因此建议使用后台启动方式对服务进行启动， 后台启动方式命令如下：

$nohup hive --service metastore 6

$nohup hive --service hiveserver26

以 metastore 服务为例，若使用前端启动方式，则操作成功后，会在终端界面输出如下 信息：

Starting Hive Metastore Server

此时该信息会停留，除非使用组合键Ctrl+C 强制结束 metastore 服务，不然 metastore 服务虽然会正常运行，并且该终端将无法进行其他操作。

若以后台启动方式启动，则操作成功后会有相应提示 “metastore 服务启动”,之后可以

340 第二篇 开源实现篇

正常进行其他操作。

可以通过执行以下命令查看服务进程运行情况：

$jps

此时，若看到有相应的 Runjar 进程在运行，说明 metastore 服务已启动。

也可以通过ps 命令来查看该Runjar 进程的启动命令信息，如图7-8所示为metastore 服务进程启动信息。

$ps -ef l grep (该Runjar 进程号)

hive 26202585819:06 PtS/0 00:00:11 /vaz/local/}dk¹ 7 0 79/bin/java

cal/hadaop/hadoop-2 6 0-cdh5 4 2 -Dhadoop id str-hive -Dhadoop,rcot 1ogger-INFO,

-Dhadoop.security.logger=INFO,NuliAppender org.apache,hadoop,util.Rundar /var/

nive 27422585 019:08 pta/0 00:00:00 grep 2620

图7-8 metastore 服务进程启动信息

**7.测试** **Hive metastore 服务**

在客户机上，使用测试用户在未获取 Kerberos 授权情况下执行如下操作步骤，来尝试 使 用 metastore 服务：

$hive

此时会有如图7-9所示报错信息出现，证明无法正常使用 metastore 服务。



Caused by:Java.1o.IOException:Javax.security.sasi.SasiException:GsS initiate

d[io valid credentials provided Mechanism level:

图7-9 metastore 报错信息

之后通过测试用户获取 Kerberos 授权，并再次尝试使用metastore 服务，执行如下操作 步骤：

$kinit -kt /home/testuser/keytabs/testuser.keytab testuser@LINC.COM

$hive

hive>show tables;

此时测试用户可以成功使用metastore 服务，进入Hive Shell 界面，可查看到 Hive 中存 在的所有表名。

**8.测试** **Hive server2 服务**

Hive server2提供了一个新的命令行工具beeline, 它是基于SQLLine CLI的 JDBC 客 户端。Beeline 工作模式有两种，即本地嵌入模式和远程模式。嵌入模式情况下，它返回一 个嵌入式的 Hive (类似于Hive CLI)。而远程模式则是通过Thrift 协议与某个单独的 Hive server2 进程进行连接通信，本节使用beeline 的远程模式进行测试。

在客户机上，使用测试用户在未获取 Kerberos 授权情况下执行如下操作步骤，来尝试 使用 server2 服务：

**第7章** **大数据安全之Kerberos 认证** 341

$beeline

beeline>!connect

{testuser密码]

jdbc:hive2://linc-1:10000/default;principal=testusereLINC.COM

**此时会有如图7-10所示报错信息出现，证明无法正常使用** **server2** **服务。**

Error:Couid not onen client transport with dDBC Uri:JabcHhive2:// 图7-10 server2 报错信息

之后通过测试用户获取Kerberos 授权，并再次尝试使用 server2 服务，执行如下操作步骤：

s kinit -kt /home/testuser/keytabs/testuser.keytab testuser@LINC.COM

$beeline

beeline>!connect

{testuser密码]

jdbc:hive2://linc-1:10000/default;principal=testuser

@LINC.COM

此时，测试用户可以成功使用server2 服务，进入beeline shell界面，并通过JDBC 连 接到了Hive 中。

**7.3.5 Zookeeper 集** **成** **Kerberos 的** **配** **置** **与** **调** **试**

本节将基于5.2.5节中 Zookeeper 的配置结果，介绍 Apache Zookeeper 服务与 Kerberos 的集成配置。本节中所使用的 Apache Zookeeper 版本为 Apache Zookeeper 3.4.8, 在实际操 作时请根据所使用的 Zookeeper 版本进行调整。

**1.创建** **Zookeeper 服务使用的** **Principal**

根据 Kerberos 的使用规范，需要为Zookeeper 服务创建相应的 Principal, 在 Zookeeper 服务的运行过程中，需要Zookeeper Principal。在 KDC(linc-krb) 上使用 root 用户执行以下

**命令来创建** **Zookeeper Principal 与** **keytab 文件：**

$kadmin.local

kadmin.local>addprinc -randkey zookeeper/linc-1@LINC.COM

kadmin.local>addprinc -randkey zookeeper/linc-2@LINC.COM

kadmin.local>addprinc -randkey zookeeper/linc-3@LINC.COM

kadmin.local>xst -k zookeeper.keytab zookeeper/linc-18LINC.COM

kadmin.local>xst -k zookeeper.keytab zookeeper/linc-2@LINC.COM

kadmin.local>xst -k zookeeper.keytab zookeeper/linc-3QLINC.COM

kadmin.local>exit

操作成功后，可以在当前目录下看到 zookeeper.keytab 文件。

**2.分** **发** **keytab 文件**

同上一节中的相应操作，将keytab 分发至 Zookeeper 的服务端 (linc-1 、linc-2 、linc-3), 建议使用 scp 命令来远程拷贝 keytab 文件。

 建议对 keytab 文件进行适当的权限设置，以避免出现安全问题。

**342** 4 **第二篇** **开源实现篇**

3.修改 Zookeeper 的 zoo.cfg 配置文件

在各个 Zookeeper 服务端节点，确保zoo.cfg 配置文件的内容如下：

authProvider.1=org.apache.zookeeper.server.auth.SASLAuthenticationProvider jaasLoginRenew=3600000

kerberos.removeHostFromPrincipal=true

kerberos.removeRealmFromPrincipal=true

其中各配置项的参数说明如表7-13所示。

**表7-13** **zoo.cfg 配置文件参数说明**

zoo.cfg

|  |  |  |
| --- | --- | --- |
| **配置项名称** | **建议配置值** | 含 义 |
| authProvider.1 | org.apache.zookeeper.server.auth.SASLAuthen ticationProvider | 权限控制容器 |
| jaasLoginRenew | 3600000 | jaas重连接时间 |
| kerberos.removeHostFromPrincipal | true | 忽视Kerberos的主机名 |
| kerberos.removeRealmFromPrincipal | true | 忽视Kerberos的域名 |

**4.创建java.env文件**

由于其他一些服务的正常使用需要调用Zookeeper 服务的相应功能，而当Zookeeper 与 Kerberos 集成后，该调用过程同样需要增加一步认证过程，而该过程的实现是将 Zookeeper 服务作为server 端，其他需要调用Zookeeper 的服务作为client 端，因此需要引入一个 JVM 机制，具体实现步骤如下。

在所有 Zookeeper 服务端的 conf 目录下创建一个java.env 文件，内容如下：

export JVMFLAGS=“-Djava.security.auth.login.config=<!--这里填写jaas.conf 文件的完整路径-->” 该文件作用是通过配置JVMFLAGS 指示jaas 文件路径。

**5.创建jaas.conf 文件**

创建完成java.env 文件之后，需要再在所有Zookeeper 服务端的 conf 目录下创建jaas. conf 文件，内容如下

Server {

com.sun.security.auth.module.Krb5LoginModule required

useKeyTab=true

keyTab="<!--- 这里填写zookeeper.keytab 文件的完整路径--->"

storeKey=true

useTicketCache=true

principal="zookeeper/linc-1@LINC.COM";

);

其中各配置项的参数说明如表7-14所示。

第7章 大数据安全之 Kerberos认证 343

**表7-14** **jaas.conf** **配置文件参数说明**

jaas.conf

|  |  |  |
| --- | --- | --- |
| **配置项名称** | **建议配置值** | **含** **义** |
| com.sun.security.auth module.Krb5LoginModule | Required | Kerberos验证模块开启 |
| useKey Tab | true | 是否使用keytab |
| key Tab | "/var/local/hadoop/zookeeper-3.4.8/ conf/keytabs/zookeeper.keytab" | keytab位置，需要对应zookeeper.keytab 文件的位置 |
| storeKey | true | 是否存储key |
| useTicketCache | true | 是否使用ticket缓存机制 |
| principal | zookeeper/linc-1@LINC.COM | server的身份，需要与keytab中的Principa 一致，格式为{服务名}/{主机名}@{域名} |

|  |
| --- |
| 3素 principal="zookeeperlinc-1@LINC.COM" 该配置项在linc-2与linc-3节点相应值分别为 principal="zookeeper/linc-2@LINC.COM"与 principal="zookeeper/linc-3@LINC.COM"。 |
| **6.启动** **Zookeeper 服务**  Zookeeper 与 Kerberos 集成完后，需要关闭原先的服务，再重新启动操作。在所有 Zookeeper 服务端以 zookeeper 用户执行以下命令进行启动：  $zkServer.sh start  此时发现， Zookeeper 与 Kerberos 完成集成后，启动时在终端打印出的启动信息与未集 成 Kerberos 情况下一致。但当查看 Zookeeper 的日志文件 zookeeper.out 时，里面的启动信 息日志与未集成 Kerberos 情况下的有所区别，会有相应的Kerberos 输出，当启动成功时会 有 “successfully logged in”信息。该信息说明Zookeeper 在集成 Kerberos 后成功启动，可 正常使用。  **7.3.6** **HBase 集** **成** **Kerberos 的配置与调试**  本节将基于7.3.5节中 Zookeeper 集成 Kerberos的配置以及5.2.5节中 HBase 的配置结 果，介绍 Apache HBase服务与 Kerberos 的集成配置。本节中所使用的 Apache HBase 版本 为 Apache HBase 1.0.3, 在实际操作时请根据使用的 HBase 版本进行具体调整。 |
| 清 H Base 集成 Kerberos 需要依赖于已集成 Kerberos的 Zookeeper 服务，因此请先完成 Zookeeper 部分的集成与配置再继续进行以下集成。 |

**1.创建** **HBase** **服务使用的** **Principal**

根据Kerberos 的使用规范，需要为HBase 服务创建相应的Principal, 在 HBase 服务的 运行过程中，需要HBase Principal。在 KDC(linc-krb) 上使用root 用户执行以下命令，来

**344** **第二篇** **开源实现篇**

**创建HBase Principal 与** **keytab 文件：**

|  |  |  |
| --- | --- | --- |
| $kadmin.local |  |  |
| kadmin.local>addprinc | -randkey | hbase/linc-10LINC.COM |
| kadmin.local>addprinc | -randkey | hbase/linc-2QLINC.COM |
| kadmin.local>addprinc | -randkey | hbase/linc-3@LINC.COM |
| kadmin.local>xst | -k hbase.keytab | hbase/linc-1QLINC.COM |
| kadmin.local>xst | -k hbase.keytab | hbase/linc-2@LINC.COM |
| kadmin.local>xst | -k hbase,keytab | hbase/linc-3@LINC.COM |
| kadmin.local>exit |  |  |

操作成功后，可以在当前目录下看到 hbase.keytab 文件。

**2.分发** **keytab 文件**

与上文中的相应操作，将keytab 分发至HBase 的服务端 (linc-1 、linc-2 、linc-3), 建

议使用scp 命令来远程拷贝 keytab 文件。

 建议对keytab 文件进行适当的权限设置，以避免出现安全问题。

**3.修改** **HBase 配置文件**

在所有 HBase 服务端节点，确保 hbase-site.xml 配置文件的内容如下：

<property>

<name>hbase.security.authentication</name>// 这个配置项指定了HBase 身份认证方式

<value>kerberos</value>

</property>

<property>

<name>hbase.security.authorization</name>// 这个配置项指定了开启HBase 身份认证方式

<value>true</value>

</property>

<property>

<name>hbase.coprocessor.region.classes</name>

//这个配置项指定了HBase 身份认证实现通过的类名

<value>org.apache.hadoop.hbase.security.token.TokenProvider</value>

</property>

以上配置项主要定义了HBase 的身份认证方式及认证需要的类，其中各配置项的参数 说明如表7-15所示。

**表7-15** **hbase-site.xml 配置文件部分配置项说明**

hbase-site.xml

|  |  |  |
| --- | --- | --- |
| **配置项名称** | **建议配置值** | **含** **义** |
| hbase.security.authentication | kerberos | HBase集群安全认证机制 |
| hbase.security.authorization | true | 是否开启HBase集群安全授权机制 |
| hbase.coprocessor.region.classes | org.apache.hadoop.hbase.security.token TokenProvider | HBase的coprocessor使用的容器 |

**第7章** **大数据安全之Kerberos 认证** 345

<property>

<name>hbase.thrift.keytab.file</name>

<value><!--- 这里填写HBase thrift服务使用的keytab 文件的完整路径---></value> </property>

<property>

<name>hbase.thrift.kerberos.principal</name>

<value>hbase/\_HOST@LINC.COM</value>

</property>

<property>

<name>hbase.rest.keytab.file</name>

<value><!--- 这里填写HBase rest服务使用的keytab 文件的完整路径---></value> </property>

<property>

<name>hbase.rest.kerberos.principal</name>

<value>hbase/\_HOST@LINC.COM</value>

</property>

以 上 配 置 项 主 要 定 义 了HBase 中 thrift 与 rest 服 务 需 要 的 keytab 文 件 及 Principal, 其

中各配置项的参数说明如表7- 16所示。

**表7-16** **hbase-site.xml 配置文件部分配置项说明**

hbase-site.xml

|  |  |  |  |
| --- | --- | --- | --- |
| **配置项名称** | **建议配置值** | **含** **义** | **约束关系及格式要求** |
| hbase.thrift.keytab file | /var/local/hadoop/ hbase-1.0.3/conf/key- tabs/hbase.keytab | HBase的thrift服务 使用的keytab文件 | 指向hbase.keytab文件 |
| hbase.thrift.kerberos principal | hbase/HOST@  LINC.COM | HBase的thrift服务 使用的Principal身份 | \_HOST会自动解析为该机主机名，需 要与keytab中的Principal一致，格式为 {服务名}/\_HOST@{域名} |
| hbase.rest.keytab file | /var/local/hadoop/ hbase-1.0.3/conf/key- tabs/hbase.keytab | HBase的rest服务 使用的keytab文件 | 指向hbase.keytab文件 |
| hbase.rest.kerberos principal | hbase/\_HOST@  LINC.COM | HBase的rest服务  使用的Principal身份 | \_HOST会自动解析为该机主机名，需 要与keytab中的Principal一致，格式为 {服务名}/\_HOST@{域名 |

<property>

<name>hbase.regionserver.kerberos.principal</name>

<value>hbase/\_HOST@LINC.COM</value>

</property>

<property>

<name>hbase.regionserver.keytab.file</name>

<value><!--- 这里填写HBase各regionserver 节点使用的keytab 文件的完整路径---></value> </property>

<property>

<name>hbase.master.kerberos.principal</name>.

<value>hbase/\_HOSTQLINC.COM</value>

</property>

第二篇 开源实现篇

<property>

<name>hbase.master.keytab.file</name>

<value><!--- 这里填写HBase 的master 节点使用的keytab 文件的完整路径---></value> </property>

以上配置项主要定义了HBase 中 regionserver 与master 两部分服务的 keytab文件及 Principal, 其中各配置项的参数说明如表7-17所示。

**表7-17** **hbase-site.xml 配置文件部分配置项说明**

hbase-site.xml

|  |  |  |  |
| --- | --- | --- | --- |
| **配置项名称** | **建议配置值** | **含** **义** | **约束关系及格式要求** |
| hbase.regionserver kerberos.principal | hbase/\_HOST@  LINC.COM | HBase的regionserver 服务使用的keytab文件 | HOST会自动解析为该机主机名，  需要与keytab中的Principal一致，格 式为{服务名}/\_HOST@{域名} |
| hbase.regionserver keytab.file | /var/local/hadoop hbase-1.0.3/conf/key- tabs/hbase.keytab | HBase的regionserver 服务使用的Principal身份 | 指向hbase.keytab文件 |
| hbase.master.ker- beros.principal | hbase/\_HOST@  LINC.COM | HBase的master使用 的Principal身份 | \_HOST会自动解析为该机主机名，  需要与keytab中的Principal一致，格 式为{服务名}/HOST@{域名} |
| hbase.master.keytab file | /var/local/hadoop/ hbase-1.0.3/conf/key tabs/hbase.keytab | HBase的master使用 的keytab文件 | 指向hbase.keytab文件 |

<property>

<name>hbase.zookeeper.property.authProvider.1</name>

//这个配置项指定了HBase 在Kerberos 模式下获取Zookeeper 服务时使用的provider

<value>org.apache.zookeeper.server.auth.SASLAuthenticationProvider</value> </property>

<property>

<name>hbase.zookeeper.property.kerberos.removeHostFromPrincipal</name> //这个配置项指定了HBase 获取Zookeeper 服务时省略Principal 的host 名

<value>true</value>

</property>

<property>

<name>hbase.zookeeper.property.kerberos.removeRealmFromPrincipal</name>

//这个配置项指定了HBase 获取Zookeeper 服务时省略Principal 的realm 名

<value>true</value>

</property>

以上配置项主要定义了配置Kerberos 后 HBase与 Zookeeper 连接时需要的信息及格 式，其中各配置项的参数说明如表7-18所示。

**表7-18** **hbase-site.xml 配置文件部分配置项说明**

hbase-site.xml

|  |  |  |
| --- | --- | --- |
| **配置项名称** | **建议配置值** | **含** **义** |
| **hbase.zookeeper.property.auth-** **Provider.1** | **org.apache.zookeeper.server.auth.** **SASLAuthenticationProvider** | **Zookeeper使用的验证控制容器** |

第7章 大数据安全之Kerberos 认证 **347**

注意

(续)

|  |  |  |
| --- | --- | --- |
| **配置项名称** | **建议配置值** | **含** **义** |
| hbase.zookeeper.property.kerberos removeHostFromPrincipal | true | 忽视Kerberos的主机名称 |
| hbase.zookeeper.property.kerberos removeRealmFromPrincipal | true | 忽视Kerberos的域名 |

**4.创建** **zk-jaas.conf 文件**

从上文中可以知道， HBase 服务的正常启动与使用需要调用Zookeeper 服务，而在集成 Kerberos 后 ，HBase 需要以一个安全的client 身份向 Zookeeper 发送服务调用请求，因此需 要在 HBase 的 conf 目录下创建相应的 client 认证文件。zk-jaas.conf 文件内容如下：



Client {

com.sun.security.auth.module.Krb5LoginModule useKeyTab=true

useTicketCache=false

keyTab="<!--- 这里填写HBase 作为client 使用的keytab

principal="hbase/linc-1@LINC.COM";

};

required

文件的完整路径--->"

其中各配置项的参数说明如表7-19所示。

**表7-19** **zk-jaas.conf 配置文件配置项说明**

hbase-site.xml

|  |  |  |  |
| --- | --- | --- | --- |
| **配置项名称** | **建议配置值** | **含** **义** | **约束关系及格式要求** |
| com.sun.security.auth module.Krb5LoginModule | required | kerberos验证模块 开启 |  |
| useKeyTab | true | 是否使用keytab | true开启，false关闭 |
| useTicketCache | false | 是否使用ticket缓 存机制 | true开启，false关闭 |
| keyTab | "/var/local/hadoop/ hbase-1.0.3/conf/  keytabs/hbase.keytab" | 连接到Zookeepei 所使用的keytab文件 | 指向hbase.keytab文件 |
| principal | hbase/linc-1@ | 连接到Zookeeper所 | 需要与keytab中的Principal一致， |
| LINC.COM | 使用的Principal身份 | 格式为{服务名}/{主机名}@{域名) |

principal="hbase/linc-1@LINC.COM"该配置项在linc-2与linc-3节点相应值分别 为principal="hbase/linc-2@LINC.COM"与principal="hbase/linc-3@LINC.COM"。

**5.启动** **HBase** **服务**

HBase 与 Kerberos 完成集成后，需要关闭原来的服务，再进行重新启动操作。在所有 HBase 服务端，以hbase 用户执行以下命令进行启动：

s start-hbase.sh

**348** **第二篇** **开源实现篇**

执行启动命令之后，会出现相应的启动提示信息，启动完成后可通过jps命令查看到 linc-1 节点上出现了HMaster 进程，以及linc-2 、linc-3 节点上的 HRegionserver 进程，证明 HBase 服务启动成功。

**6.测试** **HBase** **服务**

在客户机上，使用测试用户在未获取 Kerberos 授权情况下执行如下操作步骤，来尝试 进入HBase Shell 模式，并使用HBase 服务：

$hbase shell

此时会出现如图7-11所示报错信息。说明因未经身份认证，无法进入HBase Shell 界面， 从而无法正常使用HBase 服务。



by:GSSException:No valid credentials provided Hechanisn level:Falled te find any Rerte at sun.security.Jgss,krb5,Ktb5TnitCredential.getInetance(Rrbsiniteredential.java:147)

at sun.security.1gss.krhs.RrhSYechFactory.getCredentialEierent (RrbSMechFactorY.Java:121) ae sun,security.3gee.krhs,Rrb⁵MechFactery.setHechani griContext (KrbsMechFactary.Java:187)

at gun.security.3999,GSSContextirpl,initSecContext.(ESSCentextIrP1.JaVa:212)

cam.sun.sgeucnu.ereir**g**eg,GkSeetRtcPsiC.Iitetct(leli)saRVrot:51ent.Java:193)

245 eEe

.Jg Aanagerimp1.get? Context(GSSManagerinp1.java 223)

图7-11 HBase 认证失败信息

之后通过测试用户获取 Kerberos 授权并再次尝试使用HBase 服务，执行如下操作：

$kinit -kt /home/testuser/keytabs/testuser.keytab testuser@LINC.COM

$hbase shell

hbase(main)>list;

此时测试用户可以成功进入HBase Shell 模式，并 成功通过相应的HBase 命令查看到了HBase 上存在的所 有表信息，结果如图7-12所示。

**7.3.7 Sqoop 集** **成** **Kerberos 的配置与调试**



e(ta):005:0>HPe

FABLE

temter

2 Fow(9)4n 0.00603ecnnst

>l\*member"]

图7-12 HBase 认证成功并正常使用

本节将基于4.2.4节中Sqoop 的配置结果，介绍 Apache Sqoop 服务与 Kerberos的集 成配置。本节中所使用的Sqoop 版本为 Apache Sqoop 1.4.6, 在实际操作时请根据使用的 Sqoop 版本进行调整。

通过阅读上文可以知道，Sqoop 服务可以实现Apache 集群上跨服务的文件复制功能， 因此为保证 Sqoop 服务能正常使用，需要首先完成其他服务对Kerberos 的集成并开

启相应服务，同时还需要保证sqoop目录下包含相应服务所需要的jar包。

**1.创建** **Sqoop 服务使用的** **Principal**

根据 Kerberos 的使用规范，需要为Sqoop 服务创建Principal, 在 Sqoop 服务运行过 程中，只需要一个 sqoop principal。同时本书中 Sqoop 为单节点模式，因此只需要为linc-1

第7章 大数据安全之Kerberos 认证 349

节点创建一个sqoop principal。在KDC(linc-krb) 中使用root用户执行以下命令，来创建 Sqoop 服务需要使用的 Principal 与keytab 文件：

$kadmin.local

kadmin.local>addprinc kadmin.local>xst -k

kadmin.local>exit

-randkey

sqoop.keytab

sqoop/linc-1@LINC.COM

sqoop/linc-1@LINC.COM

操作成功后，可以在当前目录下看到 sqoop.keytab 文件。

**2.分发keytab 文件**

同上文中的相应操作，将 keytab分发至Sqoop的服务端(linc-1), 建议使用 scp命令 来远程拷贝 keytab 文件。

建议对 keytab 文件进行适当的权限设置，以避免出现安全问题。

**3.创建** **sqoop.properties 文件**

由 于Sqoop 原目录中并不存在相应的配置文件，因此为实现 Sqoop 与 Kerberos 的集 成，需要单独在 Sqoop 的 conf 目录下创建相应sqoop.properties 文件，配置文件内容如下：

org.apache.sqoop.authentication.type=KERBEROS

org.apache.sqoop.authentication.handler=org.apache.sqoop.security.Authentication. KerberosAuthenticationHandler

org.apache.sqoop.authentication.kerberos.principal=sqoop/\_HOSTELINC.COM

org.apache.sqoop.authentication.kerberos.keytab=<!--- 这里填写Sqoop使用的keytab文件的 完整路径--->

org.apache.sqoop.authentication.kerberos.http.principal=sqoop/\_HOSTELINC.COM

org.apache.sqoop.authentication.kerberos.http.keytab=<!--- 这里填写Sqoop使用的keytab 文件的完整路径--->

org.apache.sqoop.authentication.kerberos.proxyuser=true

其中各配置项的参数说明如表7-20所示。

**表7-20** **sqoop.properties 配置文件配置项说明**

sqoop.properties

|  |  |  |  |
| --- | --- | --- | --- |
| **配置项名称** | **建议配置值** | **含** **义** | **约束关系及格式要求** |
| org.apache.sqoop. auth-entication.type | KERBEROS | Sqoop使用的安全 认证机制 |  |
| org.apache.sqoop.auth- entication.handler | org.apache.sqoop.security. Authentication.Kerberos Authen- ticationHandler | Sqoop使用的Ker- beros验证机制的容器 |  |
| org.apache.sqoop. authentication.kerberos principa | sqoop/\_HOST@LINC.COM | Sqoop服务使用的 Principal身份 | 需要与keytab中的prin-  cipal一致，格式为{服务名}/ {主机名}@{域名} |
| org.apache.sqoop. authentication.kerberos keytab | /var/local/hadoop/sqoop-1.4.6/ conf/keytabs/sqoop.keytab | Sqoop服务使用的 keytab | 指向sqoop.keytab文件 |

**350** **第二篇** **开源实现篇**

(续)

|  |  |  |  |
| --- | --- | --- | --- |
| **配置项名称** | **建议配置值** | **含** **义** | **约束关系及格式要求** |
| org.apache.sqoop.  authentication.kerberos http.principal | sqoop/HOST@LINC,COM | Sqoop的http服务  使用的Principal身份 | 需要与keytab中的principal 一致，格式为{服务名}/{主 机名}@{域名) |
| org.apache.sqoop. authentication.kerberos http.keytab | /var/local/hadoop/sqoop-1.4.6/ conf/keytabs/sqoop.keytab | Sqoop的http服务 使用的keytat | 指向sqoop.keytab文件 |
| org.apache.sqoop. authentication.kerberos proxyuser | true | 是否开启Sqoop的 Proxyuser服务 | true开启，false关闭 |

**4.测试** **Sqoop** **服务**

本节以 Sqoop对 MySQL中的表与hdfs 文件的互导入功能作为测试功能，因此要先保 证 Sqoop的 lib目录下包含相应的MySQL 连接jar包。本书中使用的jar包版本为 mysql- connector-java-5.1.18-bin.jar。同时保证 hdfs 功能已集成 Kerberos 并成功开启。

清Sqoop 服务在使用时只需要直接通过相应的sqoop 命令来实现其功能，因此此处并

不 需 要 先 开 启 S q o o p 服 务 。

测试步骤与相应命令如下：

1 ) 在linc-1 上，通过root 用户进入MySQL的 test 数据库，创建相应的测试表及数据， 命令如下：

$mysql -uroot -pMyNewPass4!

mysql>use test;

mysql>create table tl(id int primary key);

mysql>insert into tl values(1),(2);

mysql>exit;

2)在客户机上使用测试用户，在未获取 Kerberos认证情况下调用 Sqoop, 命令如下：

$sqoop import --connect jdbc:mysql://linc-1/test --username root --password

MyNewPass4!--table t1

此处会发现命令运行失败，出现如图7- 13所示的错误提示信息。

Causea by:Java.1o.IOException:Javax.security.sa91.Sas1Exception:GsS initiate tatied ICaused by GSSExceptiont Ho Valld credentials ProVAaed hechanten level:

Ratied to find any Kerterag tot)l

图7-13 Sqoop 运行失败信息

3)在客户机上再次使用测试用户，获取 Kerberos 认证后调用 Sqoop, 命令如下：

s kinit -kt /home/testuser/keytabs/testuser.keytab testuser@LINC.COM

S sqoop import --connect jdbc:mysql://linc-1/test --username root --password

MyNewPass4!--table t1

此时发现 Sqoop命令运行成功，可以在hdfs 的 /user/linc/tl 目录中看到相应文件。

第7章 大数据安全之 Kerberos 认证 351

若按照本书中的配置步骤配置成功后本次操作依然失败，则可能是 hdfs 相应目录权限 问题，此时使用hdfs 用户将该目录权限设置为777后再次执行该命令，会成功执行。

4 ) 尝 试 使 用Sqoop 从 hdfs 上将文件内容导入相应的MySQL 数据库表中。首先在 linc-1 上用 root 用户删除之前创建的 tl 表中的数据，命令如下：

$mysql -uroot -pMyNewPass4!

mysql>use test;

mysql>delete from t1 where id =1 or id =2;

mysql>exit;

5)之后在客户机上使用测试用户，在未获取 Kerberos 认证情况下调用 Sqoop, 命令如下：

s sqoop export --connect jdbc:mysql://linc-1/test --username root --password

MyNewPass4!--table t1 --export-dir /user/linc/t1

此时会发现命令运行失败，同样会出现如图7-13所示的错误信息。

6)在客户机上再次使用测试用户，获取 Kerberos 认证后调用 Sqoop, 命令如下：

s kinit -kt /home/testuser/keytabs/testuser.keytab testuser@LINC.COM

s sqoop export --connect jdbc:mysql://linc-1/test --username root --password MyNewPass4!--table t1 --export-dir /user/linc/tl

此时发现 Sqoop 命令运行成功，可以在 MySQL 的 t1 表中查看到相应数据。

**7.3.8** **Hue 集成** **Kerberos 的安装与调试**

Hue 是 一 个 开 源 的Apache Hadoop UI, 它 由Cloudera Desktop 演化而来，最后 Cloudera 公司将其贡献给 Apache 基金会的 Hadoop 社区。

通过使用Hue, 用户可以在浏览器端的 Web 控制台上与Hadoop 以及与其相关的多 个组件进行交互，例如查看、操作HDFS 上的数据，运行MapReduce 任务，执行Hive 的 SQL 语句，浏览 HBase 数据库等。

本节中将会介绍Hue 的安装，对Hue 进行 Kerberos 的集成配置，并且使用Hue 对 HDFS、 YARN、Hive 以及 HBase 进行管理。因此在进行本章的配置前，请先完成以上组件的安装配置。

为了适配本章中所使用的 Apache Hadoop, 这里将重新编译 Hue 组件。

**1.下载** **Hue** **源码**

本节中使用3.9.0版本的Hue, 使 用Hue 用户下载解压缩后放置在本章开头定义的目录下。

$cd ~

s wget <https://dl.dropboxusercontent.com/u/730827/hue/releases/3.9.0/hue-3.9.0.tgz>

$tar -zxvf hue-3.9.0.tgz

s mv hue-3.9.0 /var/local/hadoop/

**2.安装依赖包**

这里使用CentOS 自带的 yum 命令安装编译所需的依赖包。

**352** **第二篇** **开源实现篇**

$sudo yum install -y ant asciidoc cyrus-sasl-devel cyrus-sasl-gssapi cyrus-sasl- plain gcc gcc-c++krb5-devel libffi-devel libxml2-devel libxslt-devel mysql mysql-devel openldap-devel python-devel sqlite-devel openssl-devel gmp-devel

**3.编译** **Hue**

$cd /var/local/hadoop/hue-3.9.0

S make apps

**4.修改** **Hadoop 配置文件**

要想让Hue 能够对Hadoop 进行管理，需要配置Hue 的代理用户。在linc-1 上使用 hdfs 用户对 core-site.xml 进行修改。

$vim /var/local/hadoop/hadoop-2.5.0/etc/hadoop/core-site.xml

确保配置项如下：

<property>

<name>hadoop.proxyuser.hue.groups</name>

<value>\*</value><!--A group which all users of Hue belong to,or the wildcard value "\*"-->

</property>

<property>

<name>hadoop.proxyuser.hue.hosts</name>

<value>\*</value>

</property>

**5.修改** **Hue** **配置文件**

在 linc-1 上使用hue 用户对 Hue 进行配置。

$vim /var/local/hadoop/hue-3.9.0/desktop/conf/hue.ini

由于Hue 的配置项过多，此处只给出需要修改的配置项。

[desktop] 部分主要为Hue 组件的Web UI 功能进行总体配置，此处主要对启动用户、 域名、端口等进行配置。配置项说明如表7-21 所示。

hpe\_slinc- 1

http\_port=8000

server\_user=hue

server\_group=hadoop

enable\_server=yes

**表7-21** **hue.ini [desktop] 部分配置项说明**

hue.ini [desktop

|  |  |  |
| --- | --- | --- |
| **配置项名称** | **建议配置值** | **含** **义** |
| http\_host | linc-1 | Hue服务的host地址 |
| http\_port | 8000 | Hue服务端口 |
| server user | hue | 开启Hue服务的用户 |

第7章 大数据安全之 Kerberos 认证 353

(续)

|  |  |  |
| --- | --- | --- |
| **配置项名称** | **建议配置值** | **含** **义** |
| server\_group | hadoop | 开启Hue服务的用户组 |
| enable\_server | yes | 是否开启服务 |

[hadoop] 部分主要为连接Hadoop 集群进行配置，此处主要对Hadoop HDFS进行 Hue 的连接设置。配置项说明如表7-22所示。

[hadoop]

[[hdfs\_clusters]]

[ddeefflltt]f]s]=hdfs://linc- 1:9000

ys\_\_

logical\_name=linc-1

hadoop\_conf\_dir=/var/local/hadoop/hadoop-2.5.0/etc/hadoop

**表7-22** **hue.ini [hadoop][[hdfs\_clusters]] 部分配置项说明**

hue.ini [hadoop][[hdfs\_clusters]][[[default]]]

|  |  |  |
| --- | --- | --- |
| **配置项名称** | **建议配置值** | **含** **义** |
| fs\_defaultfs | hdfs://linc-1:9000 | hdfs文件系统的uri,对应Hadoop中core- site.xml中的fs.defaultFS |
| logical\_name | linc-1 | namenode的逻辑名 |
| webhdfs\_url | http://linc-1:50070/webhdfs/v1 | webhdfs的url |
| security\_enabled | true | hdfs是否配置了Kerberos |
| hadoop\_conf\_dir | /var/local/hadoop/hadoop-2.5.0/etc/hadoop | Hadoop配置文件的目录 |

[yarn\_clusters] 部分主要为连接Hadoop 集群进行配置，此处主要对 Hadoop YARN的 地址、端口、配置文件路径等进行配置。配置项说明如表7-23所示。

[[yarn\_clusters]]

[[[default]]]

resourcemanager\_host=linc-1

resourcemanager\_port=8032

muriitt\_yt\_oed=True

resourcemanager\_api\_url=http://linc-1:8088

proxy\_api\_url=http://linc-1:8088

history\_server\_api\_url=http://linc-1:19888

**表7-23** **hue.ini [hadoop][[yarn\_clusters]] 部分配置项说明**

hue.ini [hadoop][[yarn\_clusters]][[[default]]]

|  |  |  |
| --- | --- | --- |
| **配置项名称** | **建议配置值** | **含** **义** |
| resourcemanager\_host | linc-1 | resourcemanager服务所在的host |
| resourcemanager\_port | 8032 | resourcemanager服务端口 |

**354** **第二篇** **开源实现篇**

(续)

|  |  |  |
| --- | --- | --- |
| **配置项名称** | **建议配置值** | **含** **义** |
| submit\_to | true | 是否向这个cluster提交任务 |
| security\_enabled | true | yarn cluster是否配置了Kerberos |
| resourcemanager\_api\_url | http://linc-1:8088 | resourcemanager服务api的url,对应yarn-site.xml中的yarn. resourcemanager.webapp.address |
| proxy\_api ur | http://linc-1:8088 | proxyserver服务Api的ur |
| history\_server\_api\_ur | http://linc-1:19888 | historyserver服务Api的url,对应mapred-site.xml中的 mapreduce.jobhistory.webapp.address |

[kerberos] 部分主要对Hue 组件进行 Kerberos 协议的集成，指定了Hue 组件所使用的 Principal 以及 Principal 所在 keytab 文件路径。配置项说明如表7-24所示。

[kerberos]

hue\_keytab=/var/local/hadoop/hue-3.9.0/desktop/conf/hue.keytab

tp\_rinpa=l-/1k@inLiNC.COM

**表7-24** **hue.ini [kerberos] 部分配置项说明**

hue.ini [kerberos]

|  |  |  |
| --- | --- | --- |
| **配置项名称** | **建议配置值** | **含** **义** |
| hue\_keytab | /var/local/hadoop/hue-  3.9.0/desktop/conf/hue.keytab | Hue服务使用的Principal所在的keytab文件目录，需要保证  在该keytab文件中包含mapreduce.jobhistory.principal中填写的 Principal |
| hue\_principal | hue/linc-1@LINC.COM | Hue服务使用的Principal,此处不明确能否使用HOST变量 |
| kinit\_path | /usr/bin/kinit | kinit程序所在的目录，就是kinit命令对应的程序 |

[librdbms]部分主要对外部数据库管理程序的连接进行配置，其中[[databases]] [[[mysql]]] 部分主要对 Hue 组件中所需要连接的 MySQL 数据库进行配置。配置项说明如 表7-25所示。

[librdbms]

[[databases]]

[[[mysql]]]

name=hue

engine=mysql

host=localhost

port=3306

user=hue

password=123456

**表7-25** **hue.ini [librdbms][[databases]] 部分配置项说明**

hue.ini [librdbms][[databases]][[[mysql]]

|  |  |  |
| --- | --- | --- |
| **配置项名称** | **建议配置值** | **含** **义** |
| **name** | **hue** | **数据库中Hue使用的database** **name** |

第7章 大数据安全之Kerberos认证 **355**

(续)

|  |  |  |
| --- | --- | --- |
| 配置项名称 | 建议配置值 | **含** **义** |
| engine | mysql | 使用的数据库类型 |
| host | localhost | 数据库地址 |
| port | 3306 | 数据库端口 |
| user | hue | 连接MySQL使用的username(MySQL中的用户) |
| passwor | 123456 | 连接MySQL使用的password |

[beeswax] 部分主要对Hue 组件中用于与Hive 组件交互的 Beeswax 组件进行配置，其 中包括 Hive 组件的主机名、端口、配置目录等。配置项说明如表7-26所示。

hbveees\_er\_host=linc- 1

hive\_server\_port=10000

hive\_conf\_dir=/var/local/hadoop/hive-1.2.1/conf

**表7-26** **hue.ini [beeswax] 部分配置项说明**

hue.ini [beeswax]

|  |  |  |
| --- | --- | --- |
| **配置项名称** | 建议配置值 | **含** **义** |
| hive\_server\_host | linc-1 | HiveServer2所在的host,如果开启了Kerberos则此处需要使用 FQDN,对应hive-site.xml中的hive.server2.thrift.bind.host |
| hive\_server\_port | 10000 | HiverServer2的Thrift服务运行的端口，对应hive-site.xml中的 hive.server2.thrift.port |
| hive\_conf\_dir | /var/local/hadoop/hive- 1.2.1/conf | Hive的配置文件所在目录(hive-site.xml所在目录) |

[hbase] 部分主要对 Hue 组件连接HBase 组件进行配置，其中包括 HBase 集群、配置 目录、Thrift 传输方式的配置。配置项说明如表7-27所示。

[hbase]

hbase\_clusters=(Clusterllinc-1:9090)

hbase\_conf\_dir=/var/local/hadoop/hbase-1.0.3/conf

thrift\_transport=buffered

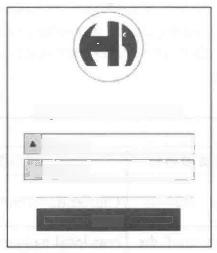
**表7-27** **hue.ini [hbase]部分配置项说明**

hue.ini [hbase]

|  |  |  |
| --- | --- | --- |
| **配置项名称** | **建议配置值** | **含** **义** |
| hbase\_clusters | Cluster|linc-1:9090 | HBase Thrift服务下的clusters列表，使用逗号隔开，格式 为(name|host:port |
| hbase\_conf\_dir | /var/local/hadoop/hbase-1.0.3/conf | HBase的配置文件目录(hbase-site.xml所在目录) |
| thrift\_transport | buffered | Thrift的传输类型，buffered是HBase的默认类型，并且 支持安全模式 |

6.为 Hue 创建 Principal

在KDC(linc-krb) 上，使用root 用户创建Principal。

**356** **第二篇** **开源实现篇**

注意

j

$kadmin.local

kadmin.local>addprinc -randkey hue/linc-1QLINC,COM

kadmin.local>xst -k hue.keytab hue/linc-1@LINC.COM

kadmin.local>exit

操作成功后，可在该目录下看到 hue.keytab。

**7.分发** **keytab**

将keytab 分发至Hue 的服务端节点(linc-1), 并给予hue 用户相应权限。

s scp hue.keytab hueelinc-1:/var/local/hadoop/hue-3.9.0/desktop/conf/ s chown hue:hadoop hue.keytab

**8.修改** **profile 文件**

在 linc-1上修改 profile 文件，为 Hue 添加相应路径。

$sudo vim /etc/profile

#在文件尾加上：

export HUE\_HOME=/var/local/hadoop/hue-3.9.0

export PATH=SHUE\_HOME/build/env/bin:SPATH

#保存退出

s source /etc/profile

**9.开启** **Hue 服务**

在 linc-1 上使用 hue 用户开启 Hue 服务。

s supervisor

supervisor属于前台服务，启动后本次Shell会话将会被占用，这里为了确保服务的 正常开启没有使用后台开启。可以使用nohup 命令后台启动服务。

**10.测试** **Hue 连接** **HDFS和** **YARN**

在 linc-client上，使用testuser用户进行以下操作：

1)在测试前请先确保linc-1 、linc-2 、linc-3上的

Hadoop 服务已正常启动。

2)使用浏览器打开网址：linc-1:8000,显示界面如

Sign in to continue to Hue

Usemame

Password

图7-14所示。

3)输入用户名： hue, 密码：hue 。登录后，单击右

siyn in

上角的 “File Browser” 按钮，显示界面如图7-15所示。

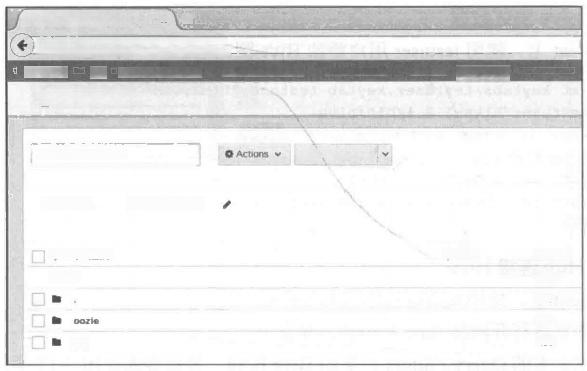
单击右上角的 “Job Browser” 按钮，显示界面如

图7-14 Hue组件的 Web 登录界面

图7-16所示。

注意：需要删除Username框中的 “hue” 才能查看所有job, 此处显示的job 为 YARN 中执行过的任务，可以执行YARN配置章节中的任务来确认此处是否能够正确显示job 状态。

**第7章** **大数据安全之Kerberos 认证** 357



Hue-Flle Browser x 中

①|10.4.21.163.8000filebrowser

HUe 费 Qery Edioa

File Browser

Search for fie name

着 Home /user/hue

\*User

hdls

hue

hue

hue

Sach s\*W

Dutk Browesrs~ Werktlows

\*Move to trash

Size

pig

Name

」

图7-15 Hue 组件的 File Browser 功能界面



)①|<10.4.21.163>:8000fobbrowserf

骨 Quey Edtors

Browser

Search lor usemame

Status User Maps

 1466331762107\_0001 QuasiMonteCarlo zte

Showng 1 to 1 ot 1 enthies

HiUE

**Logs**

Job

Name

ID

图7-16 Hue 组件的Job Browser功能界面

**11.在** **MySQL 中创建** **hue 用户**

在linc-1 上使用hue 用户进行以下操作，此处暂且将密码设置为123456。

$mysql -uroot -padmin

mysql>CREATE USER'hue'@'localhost'IDENDIFIED BY '123456';

mysql>exit

**12.开启** **HiveServer2**

在 linc-1 上，使用 hive 用户开启 HiveServer2。

s hiveserver2

HiveServer2属于前台服务，启动后本次Shell会话将会被占用，这里为了确保服务 的正常开启没有使用后台开启。可以使用 nohup 命令后台启动服务。

**358** **第二篇** **开源实现篇**

hive>exit;

**13.** **添加** **Hive** **测试数据**

**在** **linc-client 上，使用** **testuser 用户添加** **Hive 测试数据。**

s kinit -kt keytabs/testuser.keytab testuser@LINC.COM

$/home/testuser/hive-1.2.1/bin/hive

hive>create database testhue;

hive>use testhue;

hive>create table huetb(a int);

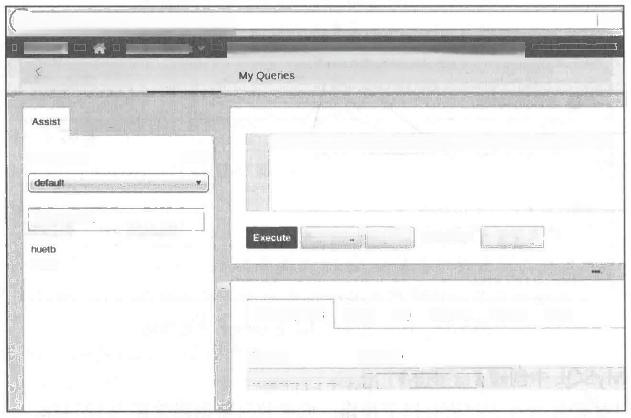
hive>insert into huetb values(1),(2);

**14.测试** **Hue 连接** **Hive**

在 linc-client上，使用testuser 用户进行测试。

1)使用浏览器打开网址：linc-1:8000, 并登录。

2)单击左上方的Query Editors 栏下的 Hive按钮，显示界面如图7-17所示。



c

① 1042116380006eeswaveguey

iUE Query Editors

Hive Editor Query Editor

Settings

1 Example:SELECT \*FROM tablename,or press CTRL +space

DATABASE C0

Table name. …

Save as … Explain or create a Now query

testtb

!

Recent quenies Query Log Columns Results *Chart*

◆Time Que

06/13/20166:36:21 PM select \*from testtb;

Seet

History

Bat Browstre ortewt

Saved Queries

Seeirty

图7-17 Hue 组件的 Hive Editor 功能界面

3)在左侧 DATABASE栏中选择 testhue, 显示如图7-18所示。

4)在右侧输入框中输入：

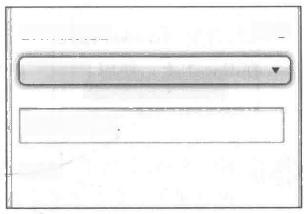
select \*from huetb;

C ②

**DATABASE**

testhue

5)单击 Execute按钮，结果如图7-19所示。

说明能够正常查询 Hive 中的数据。 Table name.

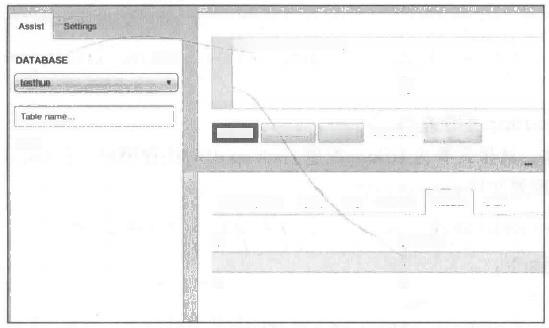
**15.** **修** **改** **HBase** **配置文件**

在 linc-1 上 ， 使 用hbase 用 户 修 改 HBase 配置文件 hbase-site.xml。

huetb

图7-18 选择 testhue 数据库界面

第7章 大数据安全之 Kerberos 认证 359



1 select from huetb;



Execuls Save as … Explaln or crente a Naw quary

Racmt queries Qmmy Log Columns Resuls Chart

|  |  |
| --- | --- |
| \*  0  1 | \*hueth.a 1  2 |

c0

huetb

1

图7-19 Hue组件 Hive Editor 功能中运行结果界面

$vim /var/local/hadoop/hbase-1.0.3/conf/hbase-site.xml

**确保配置项如下：**

<property>

<name>hbase.thrift.support.proxyuser</name>

<value>true</value>

</property>

<property>

<name>hbase.regionserver.thrift.http</name>

<value>true</value>

</property>

<property>

<name>hbase.thrift.kerberos.principal</name>

<value>HTTP/\_HoST@LINC.COM</value>

</property>

其中部分配置项的参数说明如表7-28所示。

**表7-28** **hbase-site.xml 部分配置项说明**

hbase-site.xml

|  |  |  |
| --- | --- | --- |
| **配置项名称** | **建议配置值** | **含** **义** |
| hbase.thrift.support.proxyuser | true | Thrift是否支持代理用户模式 |
| hbase.regionserver.thrift.http | true | 是否开启regionserver的Thrift的http服务 |
| hbase.thrift.kerberos.principal | HTTP/\_HOST@LINC.COM | HBase的Thrift使用的Principal |

此处经测试发现使用hbase/\_HOST 会导致 Hue 连接认证失败的问题，改为HTTP 后 即可正常运行。

**16.分发修改后的** **HBase 配置文件**

将修改后的 hbase-site.xml分发至所有 HBase 服务端节点，建议使用scp 命令进行复制。

**360** **第二篇** **开源实现篇**

$scp /var/local/hadoop/hbase-1.0.3/conf/hbase-site.xml hbase-1.0.3/conf/

$scp /var/local/hadoop/hbase-1.0.3/conf/hbase-site.xml hbase-1.0.3/conf/

linc-2:/var/local/hadoop/

linc-3:/var/local/hadoop/

**17.修改** **Hadoop 配置文件**

**与Hue一样，这里需要为HBase 配置** **Hadoop中的代理用户。在** **linc-1 上使用** **hdfs 用**

户修改 Hadoop 配置文件 core-site.xml。

$vim /var/local/hadoop/hadoop-2.5.0/etc/hadoop/core-site.xml

确保配置项如下：

<property>

<name>hadoop.proxyuser.hbase.hosts</name>

<value>\*</value>

</property>

<property>

<name>hadoop.proxyuser.hbase.groups</name>

<value>\*</value>

</property>

**18.** **分发修改后的** **Hadoop 配置文件**

将修改后的 core-site.xml 分发至所有Hadoop 服务端节点，建议使用 scp 命令进行拷贝。

$scp /var/local/hadoop/hadoop-2.5.0/etc/hadoop/core-site.xml hadoop/hadoop-2.5.0/etc/hadoop/

$scp /var/local/hadoop/hadoop-2.5.0/etc/hadoop/core-site.xml hadoop/hadoop-2.5.0/etc/hadoop/

linc-2:/var/local/

linc-3:/var/local/

**19.** **重启** **Hadoop** **服务**

在linc-1 上，使用 hdfs 用户对Hadoop 进行重启，以确保更新的配置项生效。

S cd /var/local/hadoop/hadoop-2.5.0/sbin

$./stop-dfs.sh

s sudo ./stop-secure-dfs.sh

§./start-dfs.sh

$sudo ./start-secure-dfs.sh

**20.为** **HBase 添加** **HTTP 的** **Principal**

这里需要将原先 Hadoop 使用的 Principal“HTTP/linc-1@LINC.COM” 合并入 HBase 所使用的 keytab。

需要注意的是， 一旦使用了kadmin.local 中的xst 命令来导出Principal, 这些 Principal 的状态会被刷新，导致之前导出的同名 Principal 无法继续使用。也就是说如果这里重新导 出HTTP 的 Principal, 那么 Hadoop 所使用的 hdfs.keytab 中 的HTTP Principal将会无法继 续使用，需要重新将新导出的 keytab 合并入hdfs.keytab。

因此上文配置HDFS 的过程中将HTTP 的 Principal 单独导出到了HTTP.keytab, 这里

第7章 大数据安全之 Kerberos 认证 **361**

**只需将** **HTTP.keytab 与** **hbase.keytab 合并即可。**

$ktutil

ktutil>rkt HTTP.keytab

ktutil>wkt /var/local/hadoop/hbase-1.0.3/conf/keytabs/hbase.keytab

ktutil>exit

**21.** **开** **启HBase 服务**

在linc-1上，使用hbase用户执行以下操作，开启 HBase 服务：

tart-l/hadoop/hbase-1.0.3/bin

§./hbase-daemon,sh start thrift

开启后，使用jps命令，可以看到 HMaster 和 ThriftServer。

**22.添加** **HBase** **测试数据**

在 linc-client 上，使用testuser 用户添加 HBase 测试数据。

9 kinit -kt keytabs/testuser.keytab testuser@LINC.COM

$hbase-1.0.3/bin/hbase shell

hbase>create 'member','m\_id','address','info'

hbase>put'member','scutshuxue','info:age','24'

hbase>put'member','scutshuxue','info:birthday','1987-06-17'

hbase>put'member','scutshuxue','info:company','alibaba'

hbase>quit

**23.** **测试** **Hue 连接** **HBase**

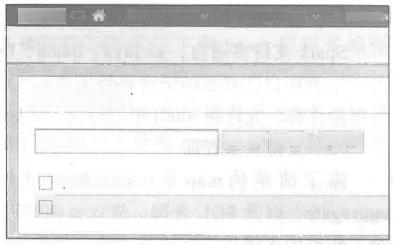
在linc-client上，使用testuser用户进行 测试。

1)使用浏览器打开网址：linc-1:8000, 并 登录。

2)单击左上角的 Data Browsers中的 HBase, 如图7-20所示。

3)单击 member 选项，如图7-21所示。

说明 Hue 连接 HBase 成功。



HiUe Query Editois

**H HBase Browser**

**主页-Cluster**

**搜索表名**

**Table Name**

member

☑启用 □禁用 愈丢弃

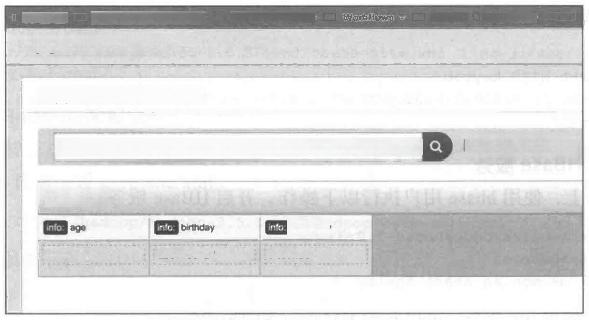
Date Broweem Worktlo

图7-20 Hue 组件 HBase Browser 功能界面

**7.3.9** **Spark 集成** **Kerberos 的安装与调试**

Spark 是一个基于内存的开源计算框架，启用了基于内存的分布数据集，除了能够提供 交互式查询外，它还可以优化迭代工作负载，支持分布式数据集上的迭代作业，可以将其 看成对 Hadoop 的补充，因此可以在 Hadoop 文件系统中并行运行。Spark 功能是通过 Scala 语言来实现的，它将 Scala 用作其应用程序框架，用 Scala 编写的程序可以像操作本地集合 对象一样轻松地操作分布式数据集。

**362** **第二篇** **开源实现篇**



Search Soeurly

**HiHBase Browser**

**主页-Cluster/member**

row\_key,row\_prefx\*+scan Jen [ool1,family:col2,fam3:,col\_preftx

scutshuxue

compamy

alibaba

ktg mcm

QueryEdton

HUE

1987-06-17

hPrem

33

图7-21 Hue组件HBase Browser 功能中查看数据界面

Spark 之所以这么受关注并被使用，主要是因为它具有与其他大数据平台不同的优势特点。

(1)轻量级快速处理

大数据处理中速度往往被置于第一位，我们经常寻找能尽快处理数据的工具。Spark 允 许传统 Hadoop 集群中的应用程序在内存中以100倍的速度运行，即使在磁盘上运行也能快 10倍。Spark 通过减少磁盘IO 来达到性能的提升，它将中间处理数据全部放到了内存中。 Spark 使用了RDD(Resilient Distributed Datasets) 数据抽象，这允许它可以在内存中存储 数据，只在需要时才持久化到磁盘。这种做法大大减少了数据处理过程中磁盘的读写操作， 大幅度降低了运行时间。

(2)易于使用

Spark 支持多语言，如 Java 、Scala 、Python 及 R(Spark 1.4版最新支持),这允许更多 的开发者在自己熟悉的语言环境下进行工作，普及了Spark 的应用范围。它自带80多个高 等级操作符，允许在Shell 中进行交互式查询，多种使用模式的特点让它的应用更灵活。

(3)支持复杂查询

除了简单的map 及 reduce 操作之外， Spark 还 支 持filter 、foreach 、reduceByKey 、 aggregate, 以及SQL 查询、流式查询等复杂查询。Spark 更为强大之处是用户可以在同一 个工作流中无缝地搭配这些功能，例如 Spark可以通过 Spark Streaming 获取流数据，然后 对数据进行实时 SQL 查询或使用MLlib库进行系统推荐，而且这些复杂业务的集成并不复 杂，因为它们都基于RDD 这一抽象数据集在不同业务过程中进行转换，转换代价小，体现 了统一引擎解决不同类型工作场景的特点。有关 Streaming技术以及MLlib 库和 RDD 将会 在后面进行详述。

(4)实时的流处理

相比MapReduce 只能处理离线数据，Spark 还能支持实时流计算。Spark Streaming 主 要用来对数据进行实时处理，当然在YARN 之后 Hadoop 也可以借助其他的工具进行流式

第7章 大数据安全之Kerberos 认证 363

计算。著名的大数据产品开发公司Cloudera 曾经对 Spark Streaming 有如下评价：

1)简单、轻量且具备功能强大的API,Spark Streaming允许用户快速开发流应用程序。

2)容错能力强，不像其他的流解决方案，比如使用Storm 需要额外的配置，而 Spark 无须额外的代码和配置，因为直接使用其上层应用框架 Spark Streaming 就可以做大量的恢 复和交付工作，让 Spark 的流计算更适应不同的需求。

3)集成性好，为流处理和批处理重用了同样的代码，甚至可以将流数据保存到历史数 据中(如 HDFS)。

(5)与已存 Hadoop 数据整合

Spark 不仅可以独立的运行(使用 standalone 模式),还可以运行在当下的 YARN 管理 集群中。它还可以读取已有的任何Hadoop 数据，这是个非常大的优势，它可以运行在任何 Hadoop 数据源土，比如 HBase 、HDFS 等。如果合适的话，这个特性让用户可以轻易迁移 已有的 Hadoop 应用。

(6)活跃且不断壮大的社区

Spark 起源于2009年，当前已有超过50个机构，730个工程师贡献过代码。与2014 年6月相比，2015年代码行数扩大了近3倍(数据源于Spark Summit 2015公布的数据), 这是个惊人的增长。

本节主要介绍Apache Spark 与 Kerberos 的集成配置，读者若有兴趣想要继续深入学习了 解 Spark 的相关原理与机制，可以阅读《 Spark: 原理、机制及应用》 一书。本节中所使用的 Apache Spark 版本为 Apache Spark 1.5.2,在实际操作时请根据使用的 Spark 版本进行调整°。

**1.下载并解压** **Spark** **tar** **包**

当创建完spark 用户并设置好ssh 免密码登录后，可以通过下载 Spark 源码并编译 Spark 二进制文件来进行安装。但由于编译过程需要根据 maven 的 pom.xml 下载依赖jar 包，整个过程十分漫长，在此推荐直接从官网上下载已编译好的相应版本的 Spark 二进制 tar 文件。在 linc-1 节点上，使用spark 用户从北理工镜像网上获取 Apache Spark 的资源并 解压，命令如下：

s wget <http://mirror.bit.edu.cn/apache/spark/spark-1.5.2/spark-1.5.2-bin-hadoop2.6.tgz>

$tar zxf ./spark-1.5.2-bin-hadoop2.6.tgz -C /var/local/hadoop/

解压 Spark 二进制tar 包之后，需要对 Spark 目录进行权限修改操作。使用root 用户进 行如下命令：

s chown -R spark:hadoop /var/local/hadoop/spark-1.5.2

$chmod -R 775 /var/local/hadoop/spark-1.5.2

**2.创建** **Spark 服务使用的** **Principal**

根据 Kerberos 的使用规范，需要为Spark 服务创建相应的 Principal, 在 Spark 服务的

θ GB/T7714刘驰.Spark 原理、机制及应用[M].北京：机械工业出版社，2016.

**364 第二篇** **开源实现篇**

运行过程中，需要 Spark Principal。在 KDC(linc-krb) 上，使用 root 用户执行以下命令，来

**创建** **spark principal 与** **keytab 文件：**

§kadmin.local

kadmin.local>addprinc kadmin.local>addprinc kadmin.local>addprinc

|  |  |
| --- | --- |
| kadmin.local>xst kadmin.local>xst kadmin.local>xst kadmin.local>exit | -k -k -k |

spark/linc-1@LINC.COM

spark/linc-2@LINC.COM

spark/linc-3@LINC.COM

|  |  |
| --- | --- |
| spark.keytab spark.keytab spark.keytab | spark/linc-1@LINC.COM spark/linc-2@LINC.COM spark/linc3@LINC.COM |

操作成功后，可以在当前目录下看到 spark.keytab 文件。

**3.分发** **keytab 文件**

同上文中的相应操作，将keytab 分 发 至Spark 的服务端 (linc-1 、linc-2 、linc-3), 建议

使用scp 命令来远程拷贝 keytab 文件。

清建议对 keytab文件进行适当的权限设置，以避免出现安全问题。

**4.** **配** **置** **Spark 文件**

本节将同时介绍 Spark 的基础安装配置以及与Kerberos 集成安装配置。在所有 Spark

服务端节点，确保以下配置文件的内容。

slaves 配置文件指定了作为 slave 的机器名，内容如下：

linc-2

linc-3

spark-env.sh 配置文件指定了Spark需要的环境变量以及Kerberos 需要的相关信息，内 容如下：

HADOOP\_CONF\_DIR=<!---这里填写Hadoop的/etc/hadoop 目录路径--->

export Hive\_CoNF\_DIR=<!---这里填写Hive 的conf 目录路径--->

export JAVA\_HOME=<!---这里填写jdk目录路径--->

export SPARK\_HISTORY\_OPTS="SSPARK\_HISTORY\_OPTS -Dspark.history.kerberos.enabled=true -Dspark.history.kerberos.principal=spark/linc-1eLINC.COM -Dspark.history. kerberos.keytab=<!--- 这里需要填写Spark 使用的keytab 文件路径--->-Dspark.history. ui.acls.enable=true"

其中各配置项的参数说明如表7-29所示。

**表7-29** **spark-env.sh** **配置文件配置项说明**

spark-env.sh

|  |  |  |
| --- | --- | --- |
| **配置项名称** | **建议配置值** | **含** **义** |
| **Dspark.history.kerberos.enabled** | **true** | **开启Spark** **Kerberos认证并配置Spark** **Kerberos认证参数** |

第7章 大数据安全之Kerberos 认证 365

(续)

|  |  |  |
| --- | --- | --- |
| **配置项名称** | **建议配置值** | **含** **义** |
| Dspark.history.kerberos. principal | spark/\_HOST@LINC.COM | Kerberos Principle认证配置，其中\_HOST 参数应该替换为该机器的hostname或者IP 地址 |
| Dspark.history.kerberos.keytab | /var/local/spark/conf/spark keytab | Kerberos秘钥文件 |
| Dspark.history.ui.acls.enable | true | Spark history UI开启ac |

-Dspark.history.kerberos.principal=spark/linc-1@LINC.COM 在 linc-2节点为-Dspark. history.kerberos.principal=spark/linc-2@LINC.COM, 在 linc-3 节点为-Dspark.history. kerberos.principal=spark/linc-3@LINC.COM。

spark-defaults.conf 配置文件指定了Spark 运行时的日志保存位置，内容如下：

spark.master=spark://linc-1:7077

spark.eventLog.dir=/applicationHistory

spark.eventLog.enabled=true

spark.yarn.historyServer.address=http://linc-1:19888

**5.启动** **Spark**

Spark 启动脚本文件名start-all.sh, 关闭脚本文件名为 stop-all.sh。当通过系统 PATH变 量运行脚本时，可能会与Hadoop 启动和关闭脚本混淆(后者脚本文件名与前者一样),因 此为了区别，在所有节点修改 Spark 脚本文件名，命令如下：

$cd /var/local/hadoop/spark-1.5.2/sbin

$mv start-all.sh start-spark-all.sh

$mv stop-all.sh stop-spark-all.sh

在完成上述配置后，在linc-1 节点通过 spark 用户启动 Spark, 命令如下： $start-spark-all.sh

启动完成后，可通过jps 命令在linc-1 节点查看到Master 进程，在linc-2 、linc-3 节点 查看到 Slave 进程，则说明启动成功。

由于Spark 关闭脚本已被修改为stop-spark-all.sh, 需 在linc-1 节点上执行该脚本来关 闭 Spark 服务：

s stop-spark-all.sh

**6.测试** **Spark** **服务**

在客户机上，使用测试用户在未获取 Kerberos 授权情况下执行如下操作步骤，来尝试 使用Spark 服务：

$spark-shell clint --master yarn

**366** **第二篇** **开源实现篇**

此时可以看到 spark-shell 进入失败，会出现如下报错信息：

Gss initiate failed [Caused by GSSException:No valid credentials provided

(Mechanism level:Failed to find any Kerberos tgt)];

之后通过测试用户获取 Kerberos授权，并再次尝试使用 Spark 服务，执行如下操作步骤：

$spark-shell clint --master yarn

此时发现无报错信息， sc 对象可正常使用。

需要注意的是， spark-shell 目前并不支持访问配置Kerberos 集群，也就是说通过 spark- shell 无法获取HDFS 上的文件，如果Spark 程序需要使用到HDFS, 可以使用 Spark on Yarn 模式运行程序。

在使用Spark on Yarn 运行 Spark程序前，请先确保 Hadoop 以及YARN服务已经开启。 本例通过spark 用户运行的 Spark程序为 Spark 自带的计算 Pi的 example, 运行结果为 Pi值。 在$SPARK\_HOME目录下，使用Spark 运行脚本运行 spark example。该脚本使用 Spark on Yarn提交 org.apache.spark.examples.SparkPi 程序并交由 YARN调度，命令如下：

$bin/run-example org.apache.spark.examples.SparkPi

运行成功后，可以看到出现返回值： Pi is roughly 3.14438。

**7.3.10** **Solr 集** **成** **Kerberos 的安装与调试**

本节将会对 Solr 进行 Kerberos的集成配置， Solr 将会使用HDFS 进行存储以及使用 Zookeeper作为配置存储服务。请在进行 Solr配置之前，先完成Kerberos、HDFS 以及 Zookeeper 的安装配置。

**1.下载并解压** **Solr**

本节中使用5.3.0版本的 Solr, 使用 Solr 用户下载解压缩后，放置在本章开头定义的目录下。

$cd ~

s wget <http://archive.apache.org/dist/lucene/solr/5.3.0/solr-5.3.0.tgz>

s tar zxvf solr-5.3.0.tgz

$mv solr-5.3.0 /var/local/hadoop/solr-5.3.0

**2.更改** **solr 目录权限**

更改 solr 目录的权限以避免使用 solr时出现权限问题。

s chown -R solr:hadoop /var/local/hadoop/solr-5.3.0

$chmod -R 775 /var/local/hadoop/solr-5.3.0

**3.创建** **Solr 服务使用的** **Principal**

根据 Kerberos的使用规范，需要为Solr 服务创建相应的Principal, 在 Solr 服务的运行 过程中，需要 Solr Principal。在 KDC(linc-krb) 上，使用 root 用户执行以下命令，来创建 Solr Principal 与 keytab 文件：

**第7章** **大数据安全之Kerberos 认证** 367

$kadmin.local

kadmin.local >addprinc kadmin.local>addprinc

kadmin.local>xst -k kadmin.local>xst -k

kadmin.local >exit

-randkey -randkey

solr.keytab zkcli.keytab

solr/linc-1QLINC.COM

zkcli/linc-1@LINC.COM

solr/linc-1@LINC.COM

zkcli/linc-1QLINC.COM

操作成功后，可以在当前目录下看到 solr.keytab 和 zkcli.keytab 文件。

**4.分发** **keytab 文件**

同上文中的相应操作，将两个 solr 的 keytab 分 发 至solr 的服务端(linc-1), 建议使用

scp 命令来远程拷贝 keytab 文件。

 建议对 keytab 文件进行适当的权限设置，以避免出现安全问题。

**5.修改** **Solr 启动配置文件**

在linc-1 上 使 用Solr 用 户 对 Solr 进 行 配 置 。Solr 的启动配置文件名为 solr.in.sh, 在

Solr 主目录的 bin 目录下。Solr 在启动的时候会使用该文件的配置信息来启动运行。

$vim /var/local/hadoop/solr-5.3.0/bin/solr.in.sh

在该文件的末尾加入如下内容：

ZK\_HOST=linc-1,linc-2,linc-3/solr

SOLR\_OPTS="SSOLR\_OPTS -Dsolr.directoryFactory=HDFSDirectoryFactory -Dsol

r.lock.type=hdfs -Dsolr.hdfs.home=hdfs://linc-1:9000/solr -Dsolr.hdfs.confdir=

/var/local/hadoop/hadoop-2.5.0/etc/hadoop"

SOLR\_AUTHENTICATION\_CLIENT\_CONFIGURER=org.apache.solr.client.solrj.imp 1.Krb5HttpClientConfigurer

SOLR\_AUTHENTICATION\_OPTS="-Djava.security.auth.login.config=/var/local/h

adoop/solr-5.3.0/conf/jaas-client.conf -Dsolr.kerberos.cookie.domain=linc-1\ -Dsolr.kerberos.principal=zkcli/linc-1QLINC.COM \

-Dsolr.kerberos.keytab=/var/local/hadoop/solr-5.3.0/conf/keytabs/zkcli.keytab"

**其中部分配置项的参数说明如表7-30所示。**

**表7-30** **solr.in.sh 文件配置项说明**

**solr.in.sh**

|  |  |  |
| --- | --- | --- |
| **配置项名称** | **建议配置值** | 含 义 |
| SOLR\_AUTHENTICATION\_  CLIENT\_CONFIGURER | org.apache.solr.client.solrj.impl.Krb5- HttpClientConfigurer | 设置Solr认证客户端的配置器 可用的值为simple(无安全认证模 式)和kerberos |
| SOLR\_AUTHENTICATION\_  OPTS | "-Djava.security.auth.login.config=/var/ local/hadoop/solr-5.3.0/conf/jaas-client.conf  Dsolr.kerberos.cookie.domain=linc-1  -Dsolr.kerberos.principal=zkcli/linc-1@ LINC.COM  -Dsolr.kerberos.keytab=/var/local/hadoop/ solr-5.3.0/conf/keytabs/zkcli.keytab" | 设置Solr认证的参数，这里可  以设置So lr使用的Kerberos的 principal和keytab位置等 |

**368** 第二篇 开源实现篇

配置文件中使用ZK\_HOST 来设置Zookeeper 的主机节点信息。如果没有填写ZK\_ HOST 信息的话，Solr 将会启动其内置的 Zookeeper 来存储 Solr 的配置信息。

**6.创建** **jaas-client.conf 文件**

为了让Solr 能够连接到使用Kerberos 安全验证的 Zookeeper, 需要创建 jaas-client.conf 文件，来使用JAAS 验证连接到 Zookeeper 。JAAS认证是以可插拔的方式执行的，因此 Java 应用程序可以从底层的认证技术中独立出来，在运行时指定期望的认证技术的配置信息。

在 Solr 的 conf 目录下创建 jaas-client.conf 文件。

s cd /var/local/hadoop/solr-5.3.0/conf

$vimjaas-client.conf

在该文件中添加如下内容：

Client {

com.sun.security.auth.module.Krb5LoginModule required

useKeyTab=true

keyTab="/var/local/hadoop/solr-5.3.0/conf/keytabs/zkcli.keytab"

storeKey=true

ache=true

principal="zkcli/linc-1@LINC.COM";

};

其中各配置项的参数说明如表7-31所示。

**表7-31** **jaas-client.conf 文件配置项说明**

jaas-client.conf

|  |  |  |  |
| --- | --- | --- | --- |
| **配置项名称** | **建议配置值** | **含** **义** | **约束关系及格式要求** |
| com.sun.security.auth. module.Krb5LoginModule | required | Kerberos验证模块开启 |  |
| useKeyTab | true | 是否使用keytat | true开启，false关闭 |
| key Tab | "/yar/local/hadoop/ solr-5.3.0/conf/keytabs/  zkcli.keytab" | keytab位置 | 需要是zkcli.keytab文件的 位置 |
| storeKey | true | 是否存储key | true开启，false关闭 |
| useTicketCache | true | 是否使用ticket缓存机制 | true开启，false关闭 |
| principa | "zkcli/linc-1@ LINC.COM" | server的身份 | 需要与keytab中的Principa  一致，格式为{服务名}/{主机 名}@{域名} |

这里使用的是principal="zkcli/linc-1@LINC.COM", 而 不 是 principal="solr/linc-1@

LINC.COM"。

第7章 大数据安全之 Kerberos认证  **369**

**7.修改** **Solr** **的配置文件**

Solr的配置文件名为 solrconfig.xml。在 linc-1上，使用solr 用户对 Solr进行配置。首 先进入 Solr的配置文件目录，然后再修改 Solr 的相关配置。

s cd /var/local/hadoop/solr-5.3.0/server/solr/configsets/data\_driven\_schema\_configs/conf $vim solrconfig.xml

找到原来的<directoryFactory>标签，改成如下内容：

<directoryFactory name="DirectoryFactory"class="solr.HDFSDirectoryFactory"> <str name="solr.hdfs.confdir">/var/local/hadoop/hadoop-2.5.0/etc/hadoop</str> <str name="solr.hdfs.home">hdfs://linc-1:9000/solr</str>

<bool name="solr.hdfs.blockcache.enabled">true</bool>

<int name="solr.hdfs.blockcache.slab.count">1</int>

<bool name="solr.hdfs.blockcache.direct.memory.allocation">true</bool>

<int name="solr.hdfs.blockcache.blocksperbank">16384</int>

<bool name="solr.hdfs.blockcache.read.enabled">true</bool>

<bool name="solr.hdfs.blockcache.write.enabled">true</bool>

<bool name="solr.hdfs.nrtcachingdirectory.enable">true</bool>

<int name="solr.hdfs.nrtcachingdirectory.maxmergesizemb">16</int> <int name="solr.hdfs.nrtcachingdirectory.maxcachedmb">192</int>

<bool name="solr.hdfs.security.kerberos.enabled">true</bool>

<str name="solr.hdfs.security.kerberos.keytabfile">/var/local/hadoop/solr-5. 3.0/conf/keytabs/solr.keytab</str>

<str name="solr.hdfs.security.kerberos.principal">solr/linc-1QLINC.COM</str>

</directoryFactory>

#保存退出

上述配置中配置了 collection的存储位置，并且设置了Kerberos 安全认证机制。 其中部分配置项的参数说明如表7-32所示。

**表7-32** **solrconfig.xml 文件配置项说明**

solrconfig.xml

|  |  |  |  |
| --- | --- | --- | --- |
| **配置项名称** | **建议配置值** | **含** **义** | **约束关系及格式要求** |
| solr.hdfs.security kerberos.enabled | true | 设置启用Kerberos 验证连接hdfs | true开启，false关闭 |
| solr.hdfs.security kerberos.keytabfile | /var/local/hadoop/solr- 5.3.0/conf/keytabs/solr. keytab | keytab的位置 | 需要是solr.keytab文件的位置 |
| solr.hdfs.security kerberos.principal | solr/linc-1@LINC.COM | 连接hdfs使用的身份 | 需要与keytab中的Principal一  致，格式为{服务名}/{主机名}@ (域名} |

这里使用的是 principal="solr/linc-1@LINC.COM", 而不是 principal="zkcli/linc-1@

LINC.COM"。

**370** **第二篇** **开源实现篇**

**8.在** **Zookeeper 上创建** **Solr 的** **chroot 目录**

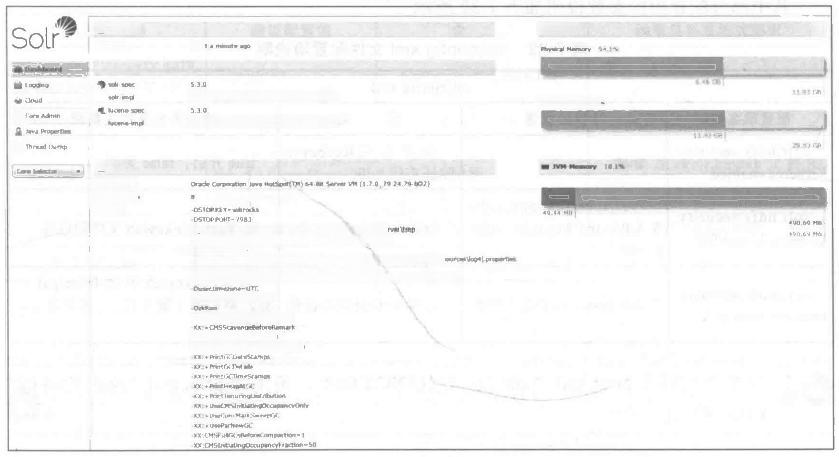
s cd /var/local/hadoop/solr-5.3.0

s server/scripts/cloud-scripts/zkcli.sh -zkhost linc-1,linc-2,linc-3 -cmd makepath

/solr

|  |  |
| --- | --- |
|  | chroot 目录是指在 Zookeeper上用于存放 Solr 的配置文件的根目录， Solr 所有的配 置文件都将存放在这个目录下，这里创建的 chroot 目录路径为/solr。 |
| **9.在** **securityjson 中添加配置信息并上传到** **Zookeeper**  s server/scripts/cloud-scripts/zkcli.sh -zkhost linc-1:2181 -cmd put /solr/secu rity.json'{"authentication":("class":"org.apache.solr.security.KerberosPlugin"}}'  10.启动 Solr 并创建 collection  使用Solr 启动命令启动 Solr, 并创建一个名叫 gettingstarted 的 collection。  $bin/solr -e cloud -noprompt | |
|  | Solr 在 SolrCloud 的模式下创建的用于搜索的数据集称作collection, 在 standalone 模式下创建的数据集称作 core。  -e cloud表示在一台机器上创建一个有1～4个node的 solrcloud 模式的集群， -noprompt 表示使用-e cloud 创建collection时的所有默认参数。如果不加-noprompt 参数，将会 提示输入新的值。默认使用4个 node, 默认创建的 collection 名称为gettingstarted。 |

启动成功后，可以使用浏览器访问 http:/linc-1:8983/solr, 可以看到如图7-22所示的界面。

I tusame

Sat

盘 wersbns

■svdam 0

but

(

5.3.01826229-notte-2015-08-1717:10:43

5.301896229-nolt-2oi5-0#4716:50=3

e H

n

a vm

|  |  |  |
| --- | --- | --- |
| a | untme  Per  P | Djvalatmpdi=C:UaeniwytPriktap'ost-5.3.0 sae-5.3.0a  Datr hane-c usrleveiDssteple s 10w#5 Bwane  ojtty pat-89n  bsogll antsurstin-fta CuhesiwraiDettpa\*-3.3 53.amv  Oottinstal d=C Uhrwre'DeAtytst-5.3.-5.3.0  Gst ssthant-CuhaieneiDedtop'sst 5.3.0wt 5JYeaputrdaufuat |

0ddntTmsst=15000

xi+OePu sRamartEsabef

wal Csh

dndis

h

图7-22 Solr的Web 登录界面

第7章 大数据安全之 Kerberos认证 371

**11.测试使用** **Solr**

首先上传 example 目录下的测试数据到 Solr 上名叫 gettingstarted 的 collection 中。

$bin/post -c gettingstarted example/films/films.json

接下来使用如下命令查询 film 关键字。

$curl http://linc-1:8983/solr/gettingstarted/select?q=film

或者在浏览器上访问http://linc-1:8983/solr/gettingstarted/select?q=film, 字，并查看返回的XML 结果，如图7-23所示。

查 询 film 关 键

|  |
| --- |
| This XIL file does not appear to have any style information associated with it.The document tree is shown below. |
| W<response)  v<lst nane="respanseHeader">  <int mane="status">0</int)  <int name="QTime\*>22</int>  w<lst name="parans\*>  <str mame="q\*>film</str>  (/t>  W<result namee"response"numFound="4"-start="0"maxSoore="0.5291085  "(doc)  <str name=\*id\*>/en//300\_2007(str>  V<arr nane="directed\_by">  (str>Zadk Srnyder/str>  (/arr)  "(arr nane-initial\_release date">  <date>2006-12-09T00:00:00Z</date?  (/arr》  T<arr nane-"genre">  <str>Epic film/str>  <str>Adventure Filn</str>  (str>Fantasy/str)  <str>Action Film(/str)  <str>Historical ficticm</str>  <str)War filnQstr)  <str>Supethero novie(/str)  <str>Historical Epic(/gtr>  (/arr》 |

图7-23 Solr 查询返回的 XML 结果

**7.3.11** **Kafka 集成** **Kerberos 的配置与调试**

通过阅读前面几章可以知道，Apache Kafka需 要 使 用Zookeeper 协调资源，因此本 节将基于5.2.7节中 Zookeeper 的配置以及Kafka 的配置结果，介绍Apache Kafka服务与 Kerberos 的集成配置。本节中所使用的Apache Kafka版本为 Apache Kafka 2.10-<0.10.0.0>, 在实际操作时请根据使用的Kafka 版本进行调整。



先对 Kafka 相关配置。

Server 部分进行 Kerberos 配置，再对 Kafka Client部分进行 Kerberos 的

**1.修改** **Kafka** **Server** **配置文件**

要让 Apache Kafka Server 开 启Kerberos 服务，需要在配置文件中开启相应的配置项。 修改 server.properties 文件，确保配置项的值如下：

**372** **第二篇** **开源实现篇**

advertised.listeners=SASL\_PLAINTEXT://linc-1:9092

(注：linc-2,linc-3 上分别为SASL\_PLAINTEXT://linc-2:9092 SASL\_PLAINTEXT://linc-3:9092)

security.inter.broker.protocol=SASL\_PLAINTEXT

sas1.mechanism.inter.broker.protocol=GSSAPI

sasl.enabled.mechanisms=GSSAPI

sasl.kerberos.service.name=kafka

listeners=SASL\_PLAINTEXT://linc-1:9092

其中部分配置项的参数说明如表7-33所示。

**表7-33** **server.properties 部分配置项说明**

server.properties

|  |  |  |
| --- | --- | --- |
| **配置项名称** | **建议配置值** | **含** **义** |
| advertised.listeners | SASL\_PLAINTEXT://linc-1:9092 | 设置Kafka Server的建议监听端口及 协议为SASL |
| security.inter.broker.protocol | SASL\_PLAINTEXT | Kafka Server之间通信的协议为SASL |
| sasl.mechanism.inter.broker.protocol | GSSAPI | SASL采用GSSAPI认证 |

**2.创建** **Kafka Server 服务使用的** **Principal**

根据 Kerberos的使用规范，需要为Kafka 服务创建Principal 。Kakfa Server 以多节点 的方式运行，需要为每个 Kafka Server创建一个 Principal。因此在 KDC(linc-krb) 中使用 root 用户执行以下命令，来创建Kafka Server需要使用的 Principal 与 keytab 文件：

$kadmin.local

kadmin.local>addprinc -randkey kafka/linc-1@LINC.COM

kadmin.local>addprinc -randkey kafka/linc-2@LINC.COM

kadmin.local>addprinc -randkey kafka/linc-3QLINC.COM

kadmin.local>xst -k kafka.keytab kafka/linc-1@LINC.COM

kadmin.local>xst -k kafka.keytab kafka/linc-2@LINC.COM

kadmin.local>xst -k kafka.keytab kafka/linc-3ELINC.COM

kadmin.local>exit

操作成功后，可以在当前目录下看到 kafka.keytab 文件。

**3.分发** **keytab** **文件到各** **Kafka** **Server端**

同上文中的相应操作，将keytab 分发至的服务端 (linc-1 、linc-2 、linc-3), 建议使用

scp 命令来远程拷贝 keytab 文件。

藩 建议对keytab 文件进行适当的权限设置，以避免出现安全问题。

**4.创建** **kafka\_server\_jaas.conf 文件**

在分发完keytab 文件后，在所有Kafka Server端 的 conf 文件夹下创建 kafka\_server\_

jaas.conf 文件，内容如下：

KafkaServer {

com.sun.security.auth.module.Krb5LoginModule required

**第7章** 大数据安全之 Kerberos 认证 **373**

注 意

useKeyTab=true

storeKey=true

keyTab="/var/local/hadoop/kafka\_2.10-<0.10.0.0>/config/keytabs/kafka.keytab" principal="kafka/linc-1@LINC.COM";

};

Client {

com.sun.security.auth.module.Krb5LoginModule required

useKeyTab=true

storeKey=true

keyTab="/var/local/hadoop/kafka\_2.10-<0.10.0.0>/config/keytabs/kafka.keytab" principal="kafka/linc-1@LINC.COM";

};

其中部分配置项的参数说明如表7-34所示。

**表7-34** **kafka\_server\_jaas.comf 部分配置项说明**

kafka\_server\_jaas.conf

|  |  |  |
| --- | --- | --- |
| **配置项名称** | **建议配置值** | **含** **义** |
| com.sun.security.auth.module.Krb5LoginModule | required | Kerberos验证模块开启 |
| useKeyTab | true | 是否使用keytat |

KafkaServer 部分用于确认 Kafka Server 之间相互通信使用的 Kerberos 的认证依据， Client 部分用于确认 Kafka Server 与 Zookeeper 之间相互通信使用的 Kerberos 的认证 依据。同时需要注意的是，两者的 keytab部分的地址为上节各节点分发的地址。此

外需要注意的是 principal="zookeeper/linc-1@LINC.COM" 该配置项在 linc-2 与 linc-3 节点相应值分别为 principal="kafka/linc-2@LINC.COM" 与 principal="kafka/linc-3@

LINC.COM"。

**5.修改** **Kafka Server 端kafka-run-class.sh 文件**

由于其他一些服务的正常使用需要调用Kafka 服务的相应功能，而当 Kafka 与 Kerberos 集成后，该调用过程同样需要增加一步认证过程，而该过程的实现是将Kafka 服务作为 server 端，其他需要调用Kafka 的服务作为 client 端，因此需要引入一个JVM 机制，具体实 现步骤如下。

在所有 Kafka Server(包括 linc-1 、linc-2 、linc-3) 的 bin 目录下找到 kafka-run-class.sh 文件，在kafka-run-class.sh文件中找到KAFKA\_JVM\_PERFORMANCE\_OPTS, 并增加两 个 JVM 参数：

-Djava.security.krb5.conf=/etc/krb5.conf

-Djava.security.auth.login.config =/var/local/hadoop/kafka\_2.10-<0.10.0.0>/config/ kafka\_server\_jaas.conf <!---这里填写jaas.conf文件的完整路径--->

**6.创建** **Kafka Client 端** **kafka\_client\_jaas.conf 文件**

Kafka Client 端包括了消息生产者以及消息消费者。为了方便，producer 和 consumer 都

**374** 第二篇 开源实现篇

安装在linc-client 端。与 Sever 端一样同样需要jaas 文件来控制客户端部分的Kerberos 配 置项。在 Kafka Client 的 config 目录下，创建 kafka\_client\_jaas.conf 文件，文件内容如下：

KafkaServer {

com.sun.security.auth.module.Krb5LoginModule required

useTicketCache=true;

};

Client {

com.sun.security.auth.module.Krb5LoginModule required

useTicketCache=true;

};

其中部分配置项的参数说明如表7-35所示。

**表7-35** **kafka\_client\_jaas.comf 部分配置项说明**

kafka\_client\_jaas.conf

|  |  |  |
| --- | --- | --- |
| **配置项名称** | **建议配置值** | 含 义 |
| com.sun.security.auth.module.Krb5LoginModule | required | Kerberos验证模块开启 |
| useTicketCache | false | 是否使用ticket缓存机制 |

KafkaServer 部分用于确认Kafka Client 与 Server 之间相互通信使用的Kerberos 的 认证依据，Client 部分用于确认Kafka Client与 Zookeeper 之间相互通信使用的 Kerberos 的认证依据。同时需要注意的是，与 Sever 采用静态 keytab 文件作为通信 令牌不同的是，客户端采用的是动态申请令牌的方式。

7.修改 Kafka Client 消费者端 console-consumer.properties 文件

在 config 目录下找到 console-consumer.properties 文件，并添加以下内容：

group.id=test-consumer-group

security.protocol=SASL\_PLAINTEXT

sasl.kerberos.service.name=kafka

8.修改 Kafka Client生产者端 console-producer.properties 文件

在 config 目录下找到 console-producer.properties 文件，并添加以下内容：

security.protocol=SASL\_PLAINTEXT

sasl.kerberos.service.name=kafka

**9.修改** **Kafka** **Client** **端** **kafka-run-class.sh** **文件**

与Server端 一 样，Kafka启动时需要指定Kerberos的参数，需要对kafka-run-class.sh 配 置文件进行修改。在bin 目录下找到kafka-run-class.sh文件，找到配置项KAFKA\_JVM PERFORMANCE\_OPTS, 并更改为：

-Djava.security.auth.login.config =/var/local/hadoop/kafka\_2.10-<0.10.0.0>/config/kafka\_

client\_jaas.conf <!---这里填写jaas.conf 文件的完整路径--->

第7章 大数据安全之 Kerberos 认证 ◆ 375

**10.启动** **Kafka Server**

在 linc-1 、linc-2 、linc-3 上，在/var/local/hadoop/kafka\_2.10-<0.10.0.0>/ 目录下，依次分 别执行以下命令：

$nohup ./bin/kafka-server-start.sh config/server.properties &

**11.在** **Kafka Server 端创建** **topic**

在 linc-1 上的/var/local/hadoop/kafka\_2.10-0.9:0.1/bin 目录下执行以下操作命令：

S./kafka-topics.sh --create --replication-factor 3 --partitions 1 --topic mytopic

--zookeeper linc-1:2181

可以看到如图7-24所示结果。



[kafka& bin]S·/kafka-topics,ah s 1 --topic mytopic --zookeeper

Created topic "mytepic".

--create EeBIICaLlON-faca

:2181

图7-24 创建topic 主题

|  |  |
| --- | --- |
|  | 根据实际情况，可以动态地调整备份数 (replication-factor) 和分区数 (partitions),  以适应不同的安全性和效率需求。 |
| **12.测试** **Kafka Client(producer)**  在linc-client 端 bin 目录下，依次输入以下命令：  s kinit -kt /home/linc/keytabs/linc.keytab linc@LINC.COM  $./kafka-console-producer.sh --topic mytopic --broker-list linc-1:9092,linc- 2:9092,linc-3:9092 --producer.config ../config/console-producer.properties | |
|  | kinit-kt/home/linc/keytabs/linc.keytab linc@LINC.COM 动态申请令牌。 |

此时可以输入消息，此时输入 “This is the message from kafka”,如图7-25所示。



bin]S ·/kafka-console-Producer.sh --topic mytopic --broker-1ist

:9092, :9092, :9092 --producer.config …/config/console-producer.pr

Thig is the message from kafka

c12016-07-0400:44:34,7331 WARN TGTseneWal thread has teen interrupted and wil

ext:.(org-apachbei.ka.ccmmon.security,kerteros,Login)

loperties

图7-25 经过 Kerberos 认证producer 发送消息

使用 kinit -kt /home/linc/keytabs/linc.keytab linc@LINC.COM 的指令毁掉动态令牌，再 来重复以上实验，可以看到如图7-26所示的报错信息，无法发送消息。

**13.测试** **Kafka Client(consumer)**

在linc-client 端 bin 目录下，依次输入以下命令：

$kinit -kt /home/linc/keytabs/linc.keytab linc@LINC.COM

**376** **第二篇** **开源实现篇**

Katka client code does not currently support cbtainind a Paseword trem the user

$./kafka-console-consumer.sh --bootstrap-server linc-1:9092,linc-2:9092,linc- 3:9092 --topic mytopic --from-beginning --consumer.config ../config/console- consumer.properties --new-consumer

:

1

:9092,

**binls kdestroy**

binls ·/kafka-console-producer.ah --tepie mytepie -broker-itst

:9092, :9092 -producer.config …/contig/console-Producer.B

eF

org.apache.kafka.common.RafkaExcept1on:Fatled to congtruct kafka producer

<init>(KafkaProduce:

clients.producer.RafkaProducer.

java:321)

at org.apache.kafka.cltents.producer.RafkaPraducer.<4nit>(KafkaProducer 3ava:181)

at kafka.producer.RewShinyPxoducer.cinit>(BaseProducer.scaie:36)

at katka,tools.CensoleProducers.main(CongoleProducer,scala:46)

at katka.tools.CongolePraducer.main(ConsoleProducer.scala)

Caused by:org.apache.katka,comran.Rafkazxceptian:Javax.security.auth.iegin.Les

Make sure -DJava.security.auth,login.contig propertY Passed to JWM and the clie

nt te cenfigured to use a tieket eache(ueing the JRhS configuratien setting 'ut

eTicketCache-true)' Make sure you are using FQDN of the Kafka broker you are ti

ying to connect to.not avatlabie to sarner authentication inforsation trem ths

图7-26 未经过 Kerberos 认证 producer 发送消息



使 用consumer 的时候需要指定 - -new-consumer

Kerberos 验证。

参 数 ， 旧 版 本 的 consumer 不 支 持

此时可以看到上面 producer 产生的消息，如图7- 27所示。

:9092, :b9katkc--tOIcneSn--fram-beqinniSCP-- une .cantig

*/cen*

Ag/console-cenguner.propertieg --new-censuner

This is the message trom katka

~C12016-07-0400:48:30,178]WARN TGT reNewal Thread has been interrupted and wil

exit.(erg.apache.kafka.cenmon.security.kerberos.Logtn)

Processed a total of i messages

bin]sⅡ

图7-27 经 过 Kerberos 认证 consumer 接收消息

同样，毁掉动态令牌后，再重复以上实验，可以发现执行错误，如图7- 28所示。

Beginning -conSumeF.ConT1G

ig/console-consuner.preperties --new-conguner

12016-07-0400:46:36,950]ERROR Unknown error when running consuner: (kafka.t

is.ConsoleConsuners)

or⁹.apeche.katka.Cotron.RafkaException:Failed to congtruet kafka consumer

at org.apache.kafka.cltenta.consumer.RatkaConsuner.<tnit>(RaFkaCcns Java:648)

at org.apache.kafka.clsents,conguner.KafkaConguter,<intt>(KafkaCensumer.

java:542)

at org.apache.katka.clients.consurer.RafkaCansuner.<init>(RatkaCe

java:524)

at kafka.consumer.NewshinyConsuner,<intt>(BaseCensuner.scala:42)

at kafka.tools.ConsoleConsumers,run(ConsoleConsuner.scala:60)

at katka.tools.ConsoleConsuner$.main(ConsoleConsumer.scala:47)

at kafka.tools,ConsoleConsumer.main(ConsoieConsuner.acala)

Caused by:org apache kafka comron KafkaException:Javax security auth 1cgin Log

tnExcepcion:Could not login:the client ig being asked tor a Paegword,but the

t**k** n-tDcJearttactriyn,scuorfgFasrtraenndtt**h**ee clie .

at is contigured to use a ticket cache(using the dnas contiguration setting'ua

eTicketCache-true)',Make using FQDN of the Rafka broker tr

ying to cennect to.not avatlable to sarner authenticatiot informatlon frem the

图7-28 未经过Kerberos 认证 consumer 接收消息

第7章 大数据安全之Kerberos 认证 377

**7.3.12** **Storm 集** **成** **Kerberos 的安装与调试**

**1.创建** **storm 用户**

首先，为了方便区分管理，可以在原来的集群节点上新建用户。这里为每个节点添加 一个用户，用户名为 storm, 密码设为123456。Shell 指令如下：

s useradd storm -g hadoop

s echo "123456"l passwd --stdin storm

返回如下信息，则用户创建成功：

Changing password for user storm.

passwd:all authentication tokens updated successfully.

之后为 storm 用户设置ssh 免密码登录，需要分别在每个节点上创建 ssh 秘钥文件，并 保存连接信息。

$ssh-keygen

#通过storm 用户名连接{hostname) 节点

$ssh-copy-id storm@(hostname}

在输入ssh-keygen 指令后或出现提示信息，按照默认配置按回车键即可。其中 hostname 为要连接的节点IP 地址或相应域名。在交互 (yes/no) 时需要输入 yes, 提示输入 密码时需要输入密码。出现类似以下信息：

Now try logging into the machine,with "ssh 'stormelinc-1'",and check in:.ssh/

authorized\_keys

则表示成功，如果显示“…can't established” 则表示发生错误。之后需要赋予 storm 用户

root 权限，命令如下：

s vim /etc/sudoers

在原有的文件内容中为 root 赋予权限的配置之后添加配置，添加之后结果如下，并在 修改之后保存退出，并用 source 命令使上述配置生效。

root

storm

ALL=(ALL)

ALL=(ALL)

ALL

ALL

**2.下载并解压缩** **Storm** **文件**

当创建 storm 用户并设置好 ssh 免密码登录后，需要下载并根据自己设置编译 Storm 二 进制文件。由于编译过程需要根据Maven 的 pom.xml 下载依赖 jar 包，整个过程十分漫长， 在此推荐直接从官网上下载已编译好的相应版本的 Storm 二进制文件。本书使用的版本为 1.0.0,登录 storm 用户并从北理工镜像网上获取Apache Storm的资源：

s wget <http://mirror.bit.edu.cn/apache/storm/apache-storm-1.0.0/apache-storm-1.0.0.tar.gz>

通过上述指令获取的 Storm 二进制文件保存在当前目录下(如果之前没更换过目录，则 默认当前目录为/home/storm)。

**378** **第二篇** **开源实现篇**

**解压缩** **apache-storm-1.0.0/apache-storm-1.0.0.tar.gz 得到** **storm-1.0.0文件，并存放在/** **var/local** **目录下。**

s tar zxf ~/spark-1.5.2-bin-hadoop2.6.tgz -C /var/local/

$sudo chown -R spark:hadoop /var/local/spark-1.5.2

s sudo chmod -R 775 /var/local/spark-1.5.2

s tar zxvf apache-storm-1.0.0.tar.gz

$mv apache-storm-1.0.0 /var/local/hadoop/

#修改storm-1.0.0 的文件拥有用户为storm, 拥有组为hadoop

S chown -R storm:hadoop /var/local/hadoop/apache-storm-1.0.0

#修改storm-1.0.0 的文件权限

§chmod -R 775 /var/local/hadoop/apache-storm-1.0.0

#创建storm 的keytabs 目录

$mkdir /var/local/hadoop/apache-storm-1.0.0/conf/keytabs

**3.配置** **Storm** **文件**

Storm配置文件主要存放在SSTORM\_HOME/conf文件下，在本例中的路径为/var/ local/apache-storm-1.0.0/conf。

首先，修改 storm.yaml 配置文件。 storm.yaml 文件主要保存Storm 集群的各种配置，包 括 Zookeeper 的 IP 以及端口， Storm 的 Nimbus 、Web UI的地址等，同时 Storm 与 Kerbores 的集成也需要在这里配置。需要编辑 storm 文件夹下的 conf 目录中的 storm.yaml 文件，添 加内容如下：

storm.zookeeper.servers:

-"linc-1"

-"linc-2"

-"linc-3"

nimbus.host:"linc-1"

nimbus.seeds: ["linc-1"]

storm.zookeeper.port:2181

nimbus.thrift.port:6627

ui.port:8080

logviewer.port:8000

在修改完 Spark 相应配置后，为了方便之后调用以及管理，需要修改系统环境变量配 置文件，修改配置文件命令如下所示：

$vim /etc/profile

T添O加R如M下\_ME=/var/local/hadoop/apache-storm- 1.0.0

export PATH=SPATH:SSTORM\_HOME/bin

#修改完成后输入下面命令刷新环境变量

s source /etc/profile

**4.配置** **Kerberos 认证的** **Storm**

对 Storm 进行配置Kerberos 认证，需要在storm.yaml 中 配 置Kerbores 需要的各类信 息。需要编辑storm 文件夹下的 conf 目录中的storm.yaml 文件，添加内容如下：

第7章 大数据安全之 Kerberos 认证 **379**

drpc.port:3772

drpc.invocations.port:3773

drpc.http.port:3774

ui.filter:"org.apache.hadoop.security.authentication.server.AuthenticationFilter" ui.filter.params:

"type":"kerberos"

"kerberos.principal":"storm/linc-1"

"kerberos.keytab":"/var/local/hadoop/apache-storm-1.0.0/conf/keytabs/storm. keytab"

"kerberos.name.rules":"RULE:[2:518$0]([jt]t@.\*EXAMPLE.COM)S/.\*/SMAPRED\_USER/ RULE:[2:$1@$0]([nd]n@.\*EXAMPLE.COM)s/.\*/SHDFS\_USER/DEFAULT"

此外在此文件中还需要加入如下配置，配置中的路径名根据读者自己的实际环境进行 修改，使得系统框架能够找到需要识别的文件，并从中获取系统需要的配置项，更改系统 的默认配置。

storm.thrift.transport:"org.apache.storm.security.auth.kerberos.KerberosSasl- TransportPlugin"

java.security.auth.login.config:"/var/local/hadoop/apache-storm-1.0.0/conf/

nimj-Xmx1024m -Djava.security.auth.login.config=/var/local/hadoop/

ui.cm-X-st-ty.auth.login.config=/var/local/hadoop/

apache-storm-1.0.0/conf/storm-jaas.conf"

supervisor.childopts:"-Xmx256m -Djava.security.auth.login.config=/var/local/

hadoop/apache-storm-1.0.0/conf/storm-jaas.conf"

storm.principal.tolocal:"org.apache.storm.security.auth.KerberosPrincipalToLocal" storm.zookeeper.superACL:"sasl:zookeeper"

nimbus.authorizer:"org.apache.storm.security.auth.authorizer.SimpleACLAuthorizer"

其中部分配置项的参数说明如表7-36所示。

**表7-36** **storm.yaml 部分配置项说明**

**storm.yam**

|  |  |  |  |
| --- | --- | --- | --- |
| **配置项名称** | **建议配置值** | **含** **义** | **约束关系及格式要求** |
| storm.zookeeper. servers | -"linc-1"  -"linc-2"  -"linc-3" | Strom节点地址 | ”节点地址" |
| nimbus.host | "linc-1" | Storm主机地址 | "主节点地址" |
| storm.zookeeper.port | 2181 | Zookeeper端口 | 端口号(数字0～65535) |
| nimbus.thrift.port | 6627 | Storm Nimbus端口 | 端口号(数字0～65535) |
| ui.port | 8080 | Storm Web UI端口 | 端口号(数字0～65535) |
| kerberos.principal | Storm/linc-1@LINC. COM | 连接hdfs使用的身份 | 需要与keytab中的Principal一致， 格式为{服务名}/{主机名}@{域名} |
| kerberos.keytab | "/var/local/hadoop/ apache-storm-1.0.0/conf/  keytabs/storm.keytab" | keytab位置 | 需要是storm.keytab文件的位置 |

380 *第二篇开源实现篇*

Storm 配置 Kerbores 需要创建 storm-jaas.conf 配置文件，这个配置文件与上述配置相 关，名称可以更改，但上面的内容同样需要修改。配置主要包括两个部分， 一个部分是配 置 Storm 的主节点部分，主要配置如下：

StormServer {

com.sun.security.auth.module.Krb5LoginModule required

useKeyTab=true

keyTab="/var/local/hadoop/apache-storm-1.0.0/conf/keytabs/storm.keytab" storeKey=true

useTicketCache=true

principal="storm/linc-1@LINC.COM";

另一个部分是Strom 中各个节点的配置，目的是让各个分节点与主节点通信时能够通 过Kerbores 的认证。主要配置如下：

Stormclient {

com.sun.security.auth.module.Krb5LoginModule required

useKeyTab=true

keyTab="/var/local/hadoop/apache-storm-1.0.0/conf/keytabs/storm.keytab"

storeKey=true

useTicketCache=true

serviceName="storm"

principal="storm@LINC.COM";

};

Client {

com.sun.security.auth.module.Krb5LoginModule required

useKeyTab=true

storeKey=true

keyTab="/var/local/hadoop/apache-storm-1.0.0/conf/keytabs/storm.keytab" principal="storm/linc-1@LINC.COM";

};

其中部分配置项的参数说明如表7-37所示。

**表7-37** **jaas.conf 部分配置项说明**

jaas.conf

|  |  |  |  |
| --- | --- | --- | --- |
| **配置项名称** | **建议配置值** | **含** **义** | **约束关系及格式要求** |
| com.sun.security.auth. module.Krb5LoginModule | required | kerberos验证模块开启 |  |
| useKey Tab | true | 是否使用keytab | true开启，false关闭 |
| keyTab | "/var/local/hadoop/solr-5.3.0/ conf/keytabs/zkcli.keytab" | keytab位置 | 需要是zkcli.keytab文件 的位置 |
| storeKey | true | 是否存储key | true开启，false关闭 |
| useTicketCache | true | 是否使用ticket缓存机制 | true开启，false关闭 |
| principal | "zkcli/linc-1@LINC.COM" | server的身份 | 需要与keytab中的Prin- cipal一致，格式为{服务 名}/{主机名}@{域名} |

第7章 大数据安全之Kerberos认证 **381**

为了配置Storm 集群，需要将配置好的Storm 复制到其他节点上，需要使用scp 命令。

本书创建了两个节点，所以需要复制的命令如下：

s scp -r apache-storm-1.0.0/storm@linc-2:/var/local/hadoop

$scp -r apache-storm-1.0.0/stormelinc-3:/var/local/hadoop

s ssh stormelinc-2 "chown -R storm:hadoop /var/local/hadoop/apache-storm-1.0.0" s ssh storm@linc-3 "chown -R storm:hadoop /var/local/hadoop/apache-storm-1.0.0"

同时因为 Storm 的特性，所以需要对各个节点上的 storm-jass.conf 配置进行修改，需要 将节点的linc-1 修改为对应节点的linc-2 或者linc-3。还需要在各个节点上配置环境变量， 并刷新环境变量以使各个节点正常启动。

此外，为了进行 Storm 权限认证，首先需要在Kerberos 服务器上创建 Storm的 Principal。 在 Kerberos 服务器上，使用root 或者 root 权限用户，执行下面 Shell 指令：

$kadmin.local

$addprinc storm/linc-1@LINC.COM

$addprinc storm/linc-2@LINC.COM

s addprinc storm/linc-3@LINC.COM

$xst -k storm.keytab storm/linc-1@LINC.COM

$xst -k storm.keytab storm/linc-2@LINC,COM

$xst -k storm.keytab storm/linc3@LINC.COM

s exit

#3个Principal 的密码均设置为123,或自行定义

上述指令中，Storm 为 Kerberos 认证用户名， linc-1 为Master 节 点 域 名 ，linc-2 与 linc-3 为之前配置的节点域名， LINC.COM 为配置Kerberos 设置的Kerberos 服务器。通过 scp 工具将生产的秘钥文件发送至 Spark 各个节点。

$scp storm.keytab stormelinc-1:/var/local/hadoop/apache-storm-1.0.0/conf/keytabs

$scp storm.keytab stormelinc-2:/var/local/hadoop/apache-storm-1.0.0/conf/keytabs

$scp storm.keytab stormelinc-3:/var/local/hadoop/apache-storm-1.0.0/conf/keytabs

5.配置 Kerberos 认证运行 Storm

首先开启 Storm 服务。首先在 Master 上启动 Nimbus。

$storm nimbus k

$jps

#输入jps 命令后在linc-1 上可以看到nimbus 进程

然后在各个节点包括 Master 上启动 superviser。

$storm supervisor &

$jps

#输入jps 命令后在linc-1、linc-2、linc-3 上可以看到superviser 进程

如果需要Client 端的话，还需要再配置一个Client 节点的 Storm 用以提交任务， Storm 需要对Client 节点进行特殊的配置。配置方式如下：

s vim conf/storm-jaas.conf

**382** **第二篇** **开源实现篇**

#修改为以下内容

StormClient {

com.sun.security.auth.module.Krb5LoginModule

required

doNotPrompt=false

useTicketCache=true

serviceName="storm";

};

为了确认Kerberos 认证，在未登录 Kerberos 时进入 Spark-shell, 在 Spark Master 节点

或者 Client 节点上运行下面脚本：

s kinit -kt /home/linc/keytabs/linc.keytab linc@LINC.COM

$/home/linc/apache-storm-1.0.0/bin/storm jar examples/storm-starter/storm-starter-1.0.0.jar production-topology remote

/home/linc/apache-storm-1.0.0/ storm.starter.StatefulTopology

脚本会运行成功。但是如果将 Kerberos 认证删除，然后再提交，会发现终端会出现报 错信息，出现报错信息表示Kerberos 配置成功。

s kdestroy

$/home/linc/apache-storm-1.0.0/bin/storm

jar /home/linc/apache-storm-1.0.0/examples/

storm-starter/storm-starter-1.0.0.jar storm.starter.StatefulTopology

production-

topology remote

**7.3.13 Impala 集成** **Kerberos 的安装与调试**

在本书的6.2.2节中已经介绍了Impala的部署安装与调试，但是对于Impala与 Kerberos的集成并没有涉及，在本节中将会对Impala集成 Kerberos进行说明。Impala包 含4个组件，分别为Impala Clients 、Hive Metastore 、Cloudera Impala 及 HBase 或 HDFS, 所以Impala集成Kerberos前必须先完成HDFS、YARN、Hive 配置的 Kerberos 认证。本 书在安装过程的过程中使用的是cdh 版本的Hadoop, 因为Impala 现在只支持 cdh 版本的 Hadoop, 具体的版本是Hadoop-2.6.0-cdh5.4.2, 读者在安装过程中请选择 Hadoop 对应的 Impala 版本进行安装。

在完成 HDFS 、YARN 、Hive 配置的 Kerberos 认证之后，需要根据6.2.2节安装好 cdh 版本的Impala, 之后再根据下面的安装流程进行集成 Kerberos 的安装与调试。本节在安装 的过程中选择了4台机器进行安装，分别为： linc-1 、linc-2 、linc-3 、linc-client 。 在 Impala 的安装过程中，本书在linc-1 上安装了state-store 和 catalog 服务，在 linc-2 、linc-3 上安装 了 impala-server, 在4台机器上都安装了Impala 基础服务。

**1.安装依赖**

先在安装 Kerberos 的机器上执行下面的两个命令：

s yum install python-devel openssl-devel python-pip cyrus-sasl cyrus-sasl-gssapi

cyrus-sasl-devel -y

$pip-python install ssl

第7章 大数据安全之 Kerberos 认证 **383**

**2.创建** **Impala 使用的** **Principal**

根据 Kerberos的使用规范，需要为Impala 服务创建Principal,Impala 以多节点的方式 运行，需要为每个 Impala 创建一个 Principal 。 因此在KDC(linc-krb) 中使用root 用户，在 目录/root/keytabs 下执行以下命令，来创建Impala 需要使用的 Principal 与 keytab 文件：

$kadmin.local

kadmin.local>addprinc kadmin.local>addprinc kadmin.local>addprinc kadmin.local>addprinc kadmin.local>xst -k kadmin.local>xst -k kadmin.local>xst -k kadmin.local>xst -k

kadmin.local>exit

-randkey impala/linc-18LINC.COM

-randkey impala/linc-2@LINC.COM

-randkey impala/linc-3QLINC.COM

-randkey impala/linc-client@LINC.COM

impala.keytab impala/linc-1@LINC.COM

impala.keytab impala/linc-2@LINC.COM

impala.keytab impala/linc-3QLINC.COM

impala.keytab impala/linc-client@LINC.COM

操作成功后，可以在当前目录下看到 imapla.keytab 文件。

**3.将生成的** **impala.keytab 分发至各个机器**

同上文中的相应操作，将impala.keytab 分发至安装了Impala 的客户端 (linc-1 、linc-2、 linc-3 、linc-client), 建议使用scp 命令来远程拷贝 impala.keytab 文件。

② 素建议对 keytab 文件进行适当的权限设置，以避免出现安全问题。

**4.修改** **impala.keytab 文件权限**

在 linc-1 、linc-2 、linc-3 、linc-client 上，使用root 用户在/etc/impala/conf/ 目录下执行

以下命令：

$sudo chown impala:hadoop impala.keytab

$sudo chmod 400 impala.keytab

**5.修改** **Impala 配置文件**

在linc-1 上，使用impala 用户打开/etc/default/impala 文件，在 IMPALA\_CATALOG\_ ARGS,IMPALA\_STATE\_STORE\_ARGS 和IMPALA\_SERVER\_ARGS 中添加下面参数：

-kerberos\_reinit\_interval=60

-principal=impala/\_HOSTELINC.COM

-keytab\_file=/etc/impala/conf/impala.keytab

其中配置项的具体含义如表7-38所示。

**表7-38** **impala 文件配置项说明**

impala

|  |  |  |
| --- | --- | --- |
| **配置项名称** | **建议配置值** | 含 义 |
| **IMPALA\_CATALOG\_SER-**  **VICE\_HOST** | **linc-1** | catalog所在机器的主机名 |

**384** **第二篇** **开源实现篇**

(续)

|  |  |  |
| --- | --- | --- |
| **配置项名称** | **建议配置值** | **含** **义** |
| IMPALA\_STATE\_STORE\_  HOST | linc-1 | state store所在机器的主机名 |
| IMPALA\_CATALOG\_ARGS | -kerberos\_reinit\_interval'=60  '-principal'=impala/\_HOST@LINC.COM  keytab\_file'=/etc/impala/conf/impala.keytab | kerberos\_reinit\_interval代表 Kerberos重新初始化的时间间隔 |
| IMPALA\_STATE\_STORE\_  ARGS | -kerberos\_reinit\_interval'=60  '-principal'=impala/\_HOST@LINC.COM  -keytab\_file'=/etc/impala/conf/impala.keytab | Principal代表Impala服务使用 的Principal身份 |
| IMPALA\_SERVER\_ARGS | -kerberos\_reinit\_interval'=60  -principal'=impala/\_HOST@LINC.COM  -keytab\_file'=/etc/impala/conf/impala.keytab | Keytab\_file代表Impala服务 使用的keytab |

同时将该 impala文件发送到 linc-2 、linc-3 、linc-client 节点上。

**6.更新** **Impala** **配置文件下的文件**

在 linc-1 上，使用root 用户将core-site.xml 和 hdfs-site.xml文件复制到/etc/impala/ conf/ 下，使用下面的命令：

$cp /var/local/hadoop/hadoop-2.6.0-cdh5.4.2/etc/hadoop/core-site.xml /etc/impala/conf $cp /var/local/hadoop/hadoop-2.6.0-cdh5.4.2/ete/hadoop/hdfs-site.xml /etc/impala/conf

再将/var/local/hive-1.1.0-cdh5.4.2/conf中 的hive-site.xml复制到/etc/impala/conf/下， 使用下面的命令：

S ep /var/local/hive-1.1.0-cdh5.4.2/conf/hive-site.xml /etc/impala/conf

还需要在/etc/impala/conf/下的 hdfs-site.xml 文件中添加以下内容：

<property>

<name>dfs.client.read.shortcircuit</name>

<value>true</value>

</property>

<property>

<name>dfs.domain.socket.path</name>

<value>/var/run/hadoop-hdfs/dn</value>

</property>

<property>

<name>dfs.datanode.hdfs-blocks-metadata.enabled</name>

<value>true</value>

</property>

<property>

<name>dfs.client.use.legacy.blockreader.local</name>

<value>false</value>

</property>

<property>

<name>dfs.datanode.data.dir.perm</name>

<value>750</value>

</property>

**第7章** **大数据安全之Kerberos 认证**

**385**

<property>

<name>dfs.block.local-path-access.user</name>

<value>impala</value>

</property>

<property>

<name>dfs.client.file-block-storage-locations.timeout</name>

<value>30000</value>

</property>

其中配置项的具体含义如表7-39所示。

**表7-39** **hdfs-site.xml 部分配置项说明**

hdfs-site.xml

|  |  |  |
| --- | --- | --- |
| **配置项名称** | 建议配置值 | **含** **义** |
| dfs.client.read.shortcircuit | true | 是否开启短路本地读特性，需要安装Hadoop 的native包libhadoop.so |
| dfs.domain.socket.path | /var/run/hadoop-hdfs/dn | DataNode和DFSClient之间沟通的Socket的本  地路径，/var/run/hadoop-hdfs目录需要root权限 |
| dfs.datanode.hdfs-blocks-metadata enabled | true | 可以在调度CPU资源时让不同的CPU读不同 的磁盘，避免查询内和查询间的IO竞争 |
| dfs.client.use.legacy.blockreader local | false | 客户端是否使用传统的块本地读写 |
| dfs.datanode.data.dir.perm | 750 | DataNode数据存放目录的读写权限 |
| dfs.block.local-path-access.user | impala | 读取dfs文件的用户，需要先将impala用户加 入hdfs用户组 |
| dfs.client.file-block-storage-locations timeout | 30000 | 客户端的文件块存储位置超时时间 |

同时将该 impala 文件发送到 linc-2 、linc-3 、linc-client 机器的相应目录上。

**7.启动** **Impala 的** **state-store 和** **catalog 服务**

在完成上面的6个步骤之后，就可以启动Impala 的 state-store 和 catalog 服务了。因为 在安装Impala 的过程中，本书将state-store 和 catalog 安装在linc-1 上，所以在linc-1 上， 使用impala 用户执行下面的命令，开启state-store 服务：

$kinit -kt /etc/impala/conf/impala.keytab impala/linc-1@LINC.COM

$sudo service impala-state-store start

**执行下面的命令开启** **catalog 服务：**

s kinit -kt /etc/impala/conf/impala.keytab impala/linc-1@LINC.COM s sudo service impala-catalog start

8.启动 impala-shell 服务

在 linc-client 上，使用impala 用户执行下面的命令，开启impala-shell 服务：

$kinit -kt /etc/impala/conf/impala.keytab impala/linc-client@LINC.COM

$impala-shell -k

386 第二篇 开源实现篇

请注意第2条命令，在启用了Kerberos 之后，运行impala-shell 时，需要添加-k 参数。 此时就可以在 impala-shell 中使用带 Kerberos 集成的 Impala 进行查询等操作了。

**7.4** **Kerberos 配置优化及常见问题**

**7.4.1** **Kerberos** **的认证方式**

根据上文中的说明，可以了解到 Kerberos 认证有两种方式：

(1)使用 Principal 与密码进行认证

通常适用于用户认证，用户通过kinit 命令向KDC 提交自己的 Principal 后，会被要求 输入密码，经KDC 验证后获得 ticket。

(2)使用事先导出的 keytab 文件进行认证

通常适用于服务端认证，keytab 文件中存储着Principal 以及一个由 Principal 密码生成 的密钥，可以在无须输入密码的情况下进行认证。因而可以使用脚本文件等方式自动地进 行 Kerberos 认证操作，被广泛使用于服务认证中。

然而，作者在实践中发现，当一个Principal 被导出至 keytab 文件中后，再使用密码进 行认证时，会出现 “kinit:Password incorrect while geting initial credentials.” 的错误提示， 而keytab 文件可以正常用于认证。造成这个问题的原因目前还不明确，可能 Kerberos 本身 不支持一个 Principal 同时存在两种认证方式，也有可能在导出Principal 的同时将KDC 中 的密码替换为了 keytab 文件中的密钥，导致原来的密码无法使用。

**7.4.2** **时间同步**

根据上文对Kerberos 的介绍，可以知道Kerberos 机制中时间同步的重要性。Linux 系 统中时间同步的方法有很多，这里列出一种使用ntp 服务自动进行时间同步的方法。

1)使用 yum 安装 ntpdate (需要root 权限)。

$yum -y install ntpdate

|  |  |
| --- | --- |
| 注  意 | -y 代表yum 安装过程中的询问步骤全部自动选 yes, 这里是为了省略安装过程中需 要输入 yes/no 的步骤。 |
| 2)手动进行时间同步(需要root权限)。  s ntpdate cn.pool.ntp.org  $clock -w | |
| 素 | ntpdate 是让主机连接ntp 服务器进行时间的同步，cn.poolntp.org 是中国的 ntp 服务 器地址。clock 命令用于修改系统内置的硬件时间， clock-w 代表将当前系统时间写 入硬件时间。 |

第7章大数据安全之Kerberos 认 证 387

3)设置后台自动执行同步。

这里使用的是CentOS 系统中的任务计划功能，将时间同步的命令设置为定时执行。

$crontab -e



crontab 是 CentOS 任务计划命令， -e 代表编辑任务计划。

执行后会进入编辑任务计划文件的步骤，在文件中需要按照crontab 的格式编写任务 计划。

0-59/10\*\*\*\*(/usr/sbin/ntpdate cn.pool.ntp.org;/sbin/clock -w)

crontab 中任务计划以行为单位，每行中包含5项表示时间的参数，最后跟上需要执行的 命令。前5项的含义分别为：分钟(0～59),小时(0～23),日期(1～31),月份(1～12), 星期(0～6)。每一项中可以指定单个数字也可以指定数字区间，可以用“/”号来连接设 置循环周期。这一行命令中，“0-59/10”代表当时间的分钟在0～59区间内时，每10分钟 触发。之后的“\*”号表示对小时、日期、月份、星期都无要求。这一行的任务计划代表每 10分钟执行一次 ntpdate 和clock 命令。

**7.4.3** **ticket 周** **期**

在使用中需要注意的是，服务端Principal 的 ticket 有效期一定要比用户的长。如果用 户 Principal 的 ticket 有效期是一周，而所连接的服务端 Principal 的有效期只有一天，那么 缓存的服务端 Principal 的 ticket 将先行过期，结果是 ticket 缓存无法正常工作。

Kerberos 中某个 Principal 的 ticket 有效期受多个参数的影响，实际的值取决于以下几 个参数中的最小值：

●Kerberos服务器上/var/kerberos/krb5kdc/kdc.conf中的 max\_life 参数。

●内置 Principal krbtgt 的 maximum ticket life(可在kadmin 中使用 getprinc 命令查看)。

●指定Principal 的 maximum ticket life(可在 kadmin 命令行下用 getprinc命令查看)。 ●Kerberos客户端上/etc/krb5.conf 中的 ticket\_lifetime 参数。

●kinit- 1参数后面指定的时间。

**7.4.4 KVNO 导致的认证失败**

在前面曾经提到过Kerberos 中的KVNO 机 制 ，KVNO 相当于KDC 数据库中存储的 Principal 密钥的版本号，在使用kadmin 命令行导出Principal 到 keytab 文件时，会使用 Principal 的密码生成密钥存储在 keytab 文件中。由于每次导出时的密钥都不同，因此KDC 中只会存储最新生成的密钥，而KVNO 就用于标识密钥的版本。KVNO 使用简单的数字来 标识，每一次的导出操作都会使得KVNO 值增加1。

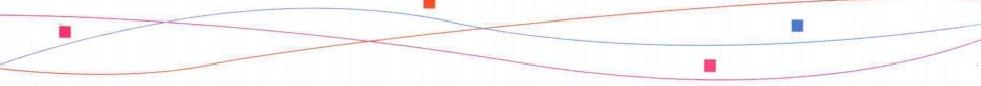
由于KDC 中只存储最新生成的密钥，因此在使用Kerberos 导出 Principal 的过程中需

388 第二篇 开源实现篇

要尤其注意KVNO 的版本。例如在上文中曾经出现了多个组件需要使用同一个Principal 的情况，如果多次导出该 Principal, 那么之前导出的密钥就会失效。建议将需要在多处使 用的Principal 导出到单独的keytab 中，然后使用ktutil 工具将密钥合并到服务所使用的 keytab 文件中。

**7.5** **本章小结**

本章主要为读者讲述了将Kerberos 集成到 Hadoop 分布式系统以及其相关组件的方 法和步骤，并且对Kerberos 本身的工作原理和使用方法进行了介绍，说明了Kerberos 在 Hadoop 系统的使用过程中是如何保障集群间通信安全的。本章中配置步骤主要针对相异于 各个组件正常配置流程的步骤进行详细说明，其中部分配置步骤需要参阅前面相关章节介 绍的正常配置流程。在介绍了Hadoop 及相关组件集成Kerberos 的配置流程后，还详细说 明了配置过程中涉及的相关配置项，便于读者查阅。



**本书特色**

技术前沿佳作

作为大数据技术的新兴前沿领域，市面上大数据治理与安全的相关书籍屈指可数。本书在结合大数据学术界最新动

态与行业生产实践的同时，对大数据治理与安全的最新理论与实践内容做了详尽的介绍。

**理论结合实践**

本书分为理论篇和开源实践篇，特别注重系统搭建与部署，书中介绍的步骤与代码从易到难、逐步深化，同时对所 涉及的知识点、难点均进行了详细的解释，从而满足不同读者的多元化需求。此外，结合实际使用场景进行了设计与实 现，为相关读者的实践操作提供有效参考。

**文字通俗易懂**

本书避免了过于理论的描述方式，简单风趣的写作风格贯穿全书，对大数据治理与安全的实践内容中的每一个组 件、每一步操作、每一行代码都进行了详细的解释，使得读者在阅读过程中能容易地按照本书指导同步上机操作，搭建

大数据治理与安全实际系统。

**作者简介**

**刘** **驰** 教授、博士生导师、北京理工大学软件学院副院长。先后入选2015年度国家人社部“高层次留学人才回国 资助计划”、中国科协青年人才托举工程、中国电子学会优秀科技工作者和第八批陕西省百人计划(短期)。分别于清 华大学和英国帝国理工学院获得学士和博士学位，后在德国电信研究院(柏林)、美国IBM T.J.Watson研究中心和 IBM 中国研究院任博士后研究员和研究主管。主要研究方向是绿色物联网大数据高效传输与处理技术。发表高水平SCl EI论文80余篇，授权国内外发明专利7项，编著中英文书籍8本。主持了国家自然科学基金、工信部2013年电子商务集 成创新试点工程等20余个省部级重点项目，现任中国自动化学会大数据专委会委员、中国工程院中国信息与电子工程科

技发展战略研究中心特聘专家(计算机应用领域)等。

投稿热线：(010)88379604

客服热线：(010)8837942688361066

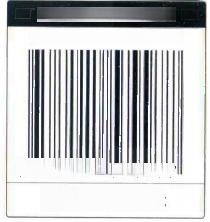
购书热线：(010)683262948837964968995259



华章网站：www.hzbook.com

网上购书：www.china-pub.com

数字阅读：www.hzmedia.com.cn



上架指导：计算机/大数据

ISBN 978-7-111-57997-7

198ll**g**11 15799771

定价：79.00元