

架构师

ARCHITECT

| 特刊 |

Hadoop十年回顾

SPECIAL ISSUE



InfoQ^{new}

目录



05 | Hadoop 十年解读与发展预测

25 | Hadoop YARN 在 Hulu 的成功实践

33 | 专访王峰：Hadoop 生态下一代计算引擎

streaming 和 batch 的统一

39 | 大数据开放平台搭建，难点何在



序言

Kyligence 联合创始人兼 CTO **李扬**

自从 2006 年 Hadoop 的缔造者们提交第一行代码已经过去了整整十年。有人说近来 Hadoop 核心社区活跃度下降，又面临如 Spark 和 Mesos 的强大挑战。Hadoop 的现状如何，未来又会怎样？

说到 Hadoop，有狭义和广义两种理解。狭义指 Apache Hadoop 开源项目本身，由 HDFS、YARN、MapReduce 三部分组成的核心。而广义来说，Hadoop 生态系统不仅仅是那只黄色的小象，而是以它为中心的整个动物园！经过了十年的风雨锤炼，Hadoop 核心已经非常完善，如果它的邮件列表活跃度有所下降，我毫不惊讶。有良好的技术服务，运行稳定，自然不需要来问问题。这反而说明了 Hadoop 核心社区的成熟和稳定，参考 Maven 就明白了，不十分活跃，但非常成熟，有广大的用户群体。

Hadoop 生态系统经过多年的发展，俨然已经成为大数据平台的事实标准，被世界范围内几乎所有的高科技公司一致采用（Google 大概是唯一的例外）。其底层由 HDFS 和 YARN 组成集群操作系统，之上有如 MapReduce 和 Spark 的运算框架，有如 HBase 和 Phoenix 的数据管理，有如 Zookeeper 和 Oozie 的协作模块，有如 Hive 和 Kylin 的数据分析，有如 Storm 和 Spark Streaming 的流式处理，有如 Mahout 和 Spark ML 的机器学习，有如 Ranger 和 Eagle 的安全监控等等。其覆盖大数据从采集到存储，从运算到分析，从安全到监控，无所不包，无所不有。如此强大的生态系统，已经完成了对大数据技术的事实垄断。试想如果有人要摒弃 Hadoop 从零做起，如何能抛开与这么多相关技术的合作？

即便强大如 Spark 有一天完全取代了 MapReduce，那也只是默默地替换了整个 Hadoop 拼图中的一块，也还是免不了被潜移默化慢慢融入 Hadoop 生态圈，成为其中一员。

刚刚结束的 Hadoop Summit 2016 充分展现了这一点。大会的主办方、赞助商、演讲嘉宾、与会听众，囊括了几乎世界上所有的大数据技术厂商，共襄盛举。在大会上，技术厂商和科研机构的分享也让我们感受到 Hadoop 今后的发展方向。

- 数据正在改变商业世界。大数据不再是象牙塔和实验室里的玩具，它已经能切实地创造商业价值，深切地改变商业世界。零售商通过大数据技术做精准市场预测，洞察物流效率，每年可以节省 7000 万美元系统开支，营收增长 8%，利润增长 3%。保险公司通过实时分析司机的驾驶模式，动态计算行驶风险并奖励安全驾驶，带来每年 26 亿美金的保险金增长，减少 4% 的理赔损失。类似的变革将在所有的行业中慢慢发生。
- HDFS 和 YARN 作为大数据的操作系统已经非常成熟，将来是中间件和上层应用百花齐放的年代。数据流处理方面竞争激烈，Storm、Spark Streaming、Flink、Nifi 等互有侧重但又各有缺陷。数据分析方面 Hive 2.0 想要王者回归，Kylin 从预计算角度另辟蹊径。安全领域比如 Ranger 和 Atlas，也是大公司的重点。机器学习持续火热，技术逐渐普及化。
- 系统层面。YARN.NEXT 试图重新定义 Hadoop 应用，根据组装描述文件自动适配资源，部署应用到整个集群，而不是仅管理组成应用的每个部件。Tiered HDFS 根据数据的特性（比如活跃度）透明地在多种性价比不同的存储介质之间移动数据，从而提高数据存取的效率。极小化软件对运行环境和类库的依赖，为在物联网小微设备上的运行做好准备。
- Technoethics（技术伦理学）也是不可避免的话题。大数据和人工智能技术一旦被滥用，将对整个人类社会造成及其可怕的后果，这不是科幻小说里的妄想，而很可能正在我们身边发生。应当立即行动，为技术伦理制定规范。

Hadoop 已经走过了第一个黄金十年，看起来正走向第二个黄金的十年。

Hadoop 十年解读与发展预测

作者 陈飏

【编者按】Hadoop 于 2006 年 1 月 28 日诞生，至今已有 10 年，它改变了企业对数据的存储、处理和分析的过程，加速了大数据的发展，形成了自己的极其火爆的技术生态圈，并受到非常广泛的应用。在 2016 年 Hadoop 十岁生日之际，InfoQ 策划了一个 Hadoop 热点系列文章，为大家梳理 Hadoop 这十年的变化，技术圈的生态状况，回顾以前，激励当下。本文是 Cloudera 资深工程师讲解 Hadoop，让您一篇文章就能了解 Hadoop 的过去和未来。

“昔我十年前，与君始相识。”

——白居易，《酬元九对新栽竹有怀见寄》

一瞬间 Hadoop 也到了要初中择校的年龄了。

十年前还没有 Hadoop，几年前国内 IT 圈里还不知道什么是 Hadoop，而现在几乎所有大型企业的 IT 系统中有已经有了 Hadoop 的集群在运行了各式各样的任务。

2006 年项目成立的一开始，“Hadoop”这个单词只代表了两个组件——HDFS 和 MapReduce。到现在的 10 个年头，这个单词代表的是“核心”（即 Core Hadoop 项目）以及与之相关的一个不断成长的生态系统。这个和 Linux 非常类似，都是由一个核心和一个生态系统组成。

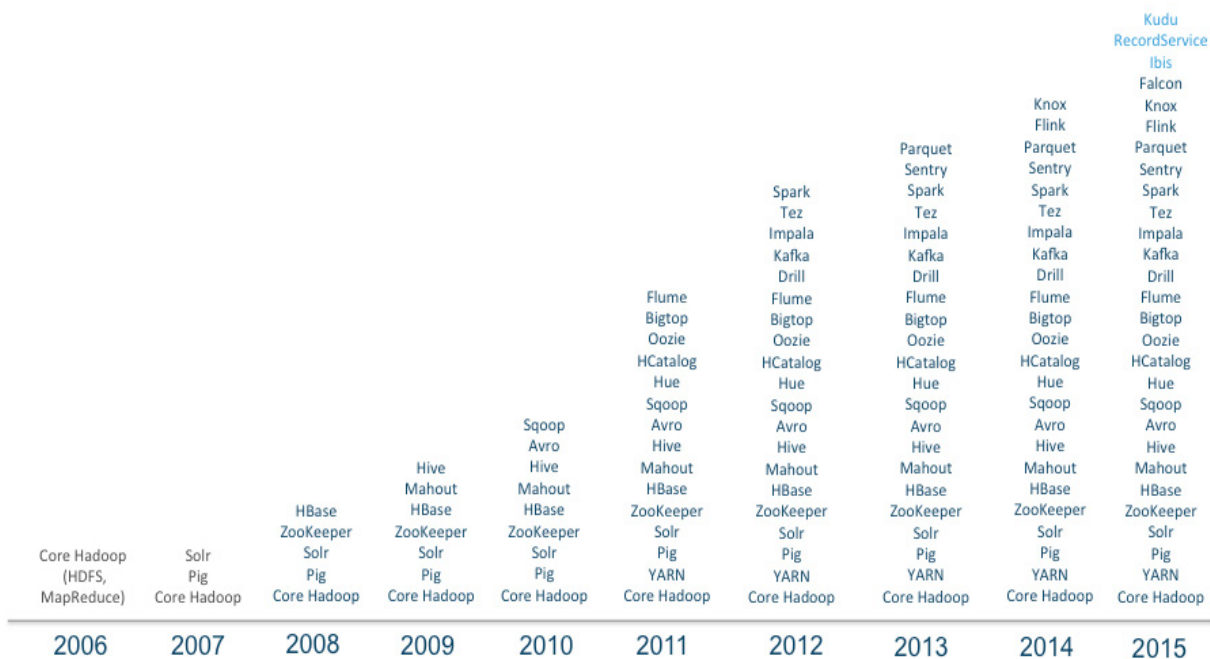
现在 Hadoop 俨然已经成为企业数据平台的“新常态”。我们很荣幸能够见证 Hadoop 十年从无到有，再到称王。在我们感动于技术的日新月异时，希望能通过本文能为 Hadoop 的昨天、今天和明天做出一点自己的解读，算是为 Hadoop 庆祝 10 岁生日献上礼物。

Hadoop 编年史

- 2002 年 10 月，Doug Cutting 和 Mike Cafarella 创建了开源网页爬虫项目 Nutch。
- 2003 年 10 月，Google 发表 Google File System 论文。
- 2004 年 7 月，Doug Cutting 和 Mike Cafarella 在 Nutch 中实现了类似 GFS 的功能，即后来 HDFS 的前身。
- 2004 年 10 月，Google 发表了 MapReduce 论文。
- 2005 年 2 月，Mike Cafarella 在 Nutch 中实现了 MapReduce 的最初版本。
- 2005 年 12 月，开源搜索项目 Nutch 移植到新框架，使用 MapReduce 和 NDfs (Nutch Distributed File System) 来运行，在 20 个节点稳定运行。
- 2006 年 1 月，Doug Cutting 加入雅虎，Yahoo! 提供一个专门的团队和资源将 Hadoop 发展成一个可在网络上运行的系统。
- 2006 年 2 月，Apache Hadoop 项目正式启动以支持 MapReduce 和 HDFS 的独立发展。
- 2006 年 2 月，Yahoo! 的网格计算团队采用 Hadoop。
- 2006 年 3 月，Yahoo! 建设了第一个 Hadoop 集群用于开发。
- 2006 年 4 月，第一个 Apache Hadoop 发布。
- 2006 年 4 月，在 188 个节点上（每个节点 10GB）运行排序测试集需要 47.9 个小时。
- 2006 年 5 月，Yahoo! 建立了一个 300 个节点的 Hadoop 研究集群。
- 2006 年 5 月，在 500 个节点上运行排序测试集需要 42 个小时（硬件配置比 4 月的更好）。
- 2006 年 11 月，研究集群增加到 600 个节点。
- 2006 年 11 月，Google 发表了 Bigtable 论文，这最终激发了 HBase 的创建。
- 2006 年 12 月，排序测试集在 20 个节点上运行 1.8 个小时，100 个节点上运行 3.3 小时，500 个节点上运行 5.2 小时，900 个节点上运行 7.8 个小时。
- 2007 年 1 月，研究集群增加到 900 个节点。
- 2007 年 4 月，研究集群增加到两个 1000 个节点的集群。
- 2007 年 10 月，第一个 Hadoop 用户组会议召开，社区贡献开始急剧上升。
- 2007 年，百度开始使用 Hadoop 做离线处理。
- 2007 年，中国移动开始在“大云”研究中使用 Hadoop 技术。
- 2008 年，淘宝开始投入研究基于 Hadoop 的系统——云梯，并将其用于处理电子商务相关数据。
- 2008 年 1 月，Hadoop 成为 Apache 顶级项目。
- 2008 年 2 月，Yahoo! 运行了世界上最大的 Hadoop 应用，宣布其搜索引擎产品部署在一个拥有 1 万个内核的 Hadoop 集群上。

- 2008 年 4 月，在 900 个节点上运行 1TB 排序测试集仅需 209 秒，成为世界最快。
- 2008 年 6 月，Hadoop 的第一个 SQL 框架——Hive 成为了 Hadoop 的子项目。
- 2008 年 7 月，Hadoop 打破 1TB 数据排序基准测试记录。Yahoo! 的一个 Hadoop 集群用 209 秒完成 1TB 数据的排序，比上一年的纪录保持者保持的 297 秒快了将近 90 秒。
- 2008 年 8 月，第一个 Hadoop 商业化公司 Cloudera 成立。
- 2008 年 10 月，研究集群每天装载 10TB 的数据。
- 2008 年 11 月，Apache Pig 的最初版本发布。
- 2009 年 3 月，17 个集群总共 24000 台机器。
- 2009 年 3 月，Cloudera 推出世界上首个 Hadoop 发行版——CDH (Cloudera's Distribution including Apache Hadoop) 平台，完全由开放源码软件组成。
- 2009 年 4 月，赢得每分钟排序，59 秒内排序 500GB（在 1400 个节点上）和 173 分钟内排序 100TB 数据（在 3400 个节点上）。
- 2009 年 5 月，Yahoo 的团队使用 Hadoop 对 1 TB 的数据进行排序只花了 62 秒时间。
- 2009 年 6 月，Cloudera 的工程师 Tom White 编写的《Hadoop 权威指南》初版出版，后被誉为 Hadoop 圣经。
- 2009 年 7 月，Hadoop Core 项目更名为 Hadoop Common;
- 2009 年 7 月，MapReduce 和 Hadoop Distributed File System (HDFS) 成为 Hadoop 项目的独立子项目。
- 2009 年 7 月，Avro 和 Chukwa 成为 Hadoop 新的子项目。
- 2009 年 8 月，Hadoop 创始人 Doug Cutting 加入 Cloudera 担任首席架构师。
- 2009 年 10 月，首届 Hadoop World 大会在纽约召开。
- 2010 年 5 月，Avro 脱离 Hadoop 项目，成为 Apache 顶级项目。
- 2010 年 5 月，HBase 脱离 Hadoop 项目，成为 Apache 顶级项目。
- 2010 年 5 月，IBM 提供了基于 Hadoop 的大数据分析软件——InfoSphere BigInsights，包括基础版和企业版。
- 2010 年 9 月，Hive(Facebook) 脱离 Hadoop，成为 Apache 顶级项目。
- 2010 年 9 月，Pig 脱离 Hadoop，成为 Apache 顶级项目。
- 2010 年~2011 年，扩大的 Hadoop 社区忙于建立大量的新组件 (Crunch, Sqoop, Flume, Oozie 等) 来扩展 Hadoop 的使用场景和可用性。
- 2011 年 1 月，ZooKeeper 脱离 Hadoop，成为 Apache 顶级项目。
- 2011 年 3 月，Apache Hadoop 获得 Media Guardian Innovation Awards。
- 2011 年 3 月，Platform Computing 宣布在它的 Symphony 软件中支持 Hadoop MapReduce API。

- 2011 年 5 月, Mapr Technologies 公司推出分布式文件系统和 MapReduce 引擎——MapR Distribution for Apache Hadoop。
- 2011 年 5 月, HCatalog 1.0 发布。该项目由 Hortonworks 在 2010 年 3 月份提出, HCatalog 主要用于解决数据存储、元数据的问题, 主要解决 HDFS 的瓶颈, 它提供了一个地方来存储数据的状态信息, 这使得 数据清理和归档工具可以很容易的进行处理。
- 2011 年 4 月, SGI (Silicon Graphics International) 基 于 SGI Rackable 和 CloudRack 服务器产品线提供 Hadoop 优化的解决方案。
- 2011 年 5 月, EMC 为客户推出一种新的基于开源 Hadoop 解决方案的数据中心设备——GreenPlum HD, 以助其满足客户日益增长的数据分析需求并加快利用开源数据分析软件。Greenplum 是 EMC 在 2010 年 7 月收购的一家开源数据仓库公司。
- 2011 年 5 月, 在收购了 Engenio 之后, NetApp 推出与 Hadoop 应用结合的产品 E5400 存储系统。
- 2011 年 6 月, Calxeda 公司发起了“开拓者行动”, 一个由 10 家软件公司组成的团队将为基于 Calxeda 即将推出的 ARM 系统上芯片设计的服务器提供支持。并为 Hadoop 提供低功耗服务器技术。
- 2011 年 6 月, 数据集成供应商 Informatica 发布了其旗舰产品, 产品设计初衷是处理当今事务和社会媒体所产生的海量数据, 同时支持 Hadoop。
- 2011 年 7 月, Yahoo! 和硅谷风险投资公司 Benchmark Capital 创建了 Hortonworks 公司, 旨在让 Hadoop 更加可靠, 并让企业用户更容易安装、管理和使用 Hadoop。
- 2011 年 8 月, Cloudera 公布了一项有益于合作伙伴生态系统的计划——创建一个生态系统, 以便硬件供应商、软件供应商以及系统集成商可以一起探索如何使用 Hadoop 更好的洞察数据。
- 2011 年 8 月, Dell 与 Cloudera 联合推出 Hadoop 解决方案——Cloudera Enterprise。Cloudera Enterprise 基于 Dell PowerEdge C2100 机架服务器以及 Dell PowerConnect 6248 以太网交换机。
- 2012 年 3 月, 企业必须的重要功能 HDFS NameNode HA 被加入 Hadoop 主版本。
- 2012 年 8 月, 另外一个重要的企业适用功能 YARN 成为 Hadoop 子项目。
- 2012 年 10 月, 第一个 Hadoop 原生 MPP 查询引擎 Impala 加入到了 Hadoop 生态圈。
- 2014 年 2 月, Spark 逐渐代替 MapReduce 成为 Hadoop 的缺省执行引擎, 并成为 Apache 基金会顶级项目。
- 2015 年 2 月, Hortonworks 和 Pivotal 抱团提出“Open Data Platform”的倡议, 受到传统企业如 Microsoft、IBM 等企业支持, 但其它两大 Hadoop 厂商 Cloudera 和 MapR 拒绝参与。



- 2015 年 10 月，Cloudera 公布继 HBase 以后的第一个 Hadoop 原生存储替代方案——Kudu。
- 2015 年 12 月，Cloudera 发起的 Impala 和 Kudu 项目加入 Apache 孵化器。

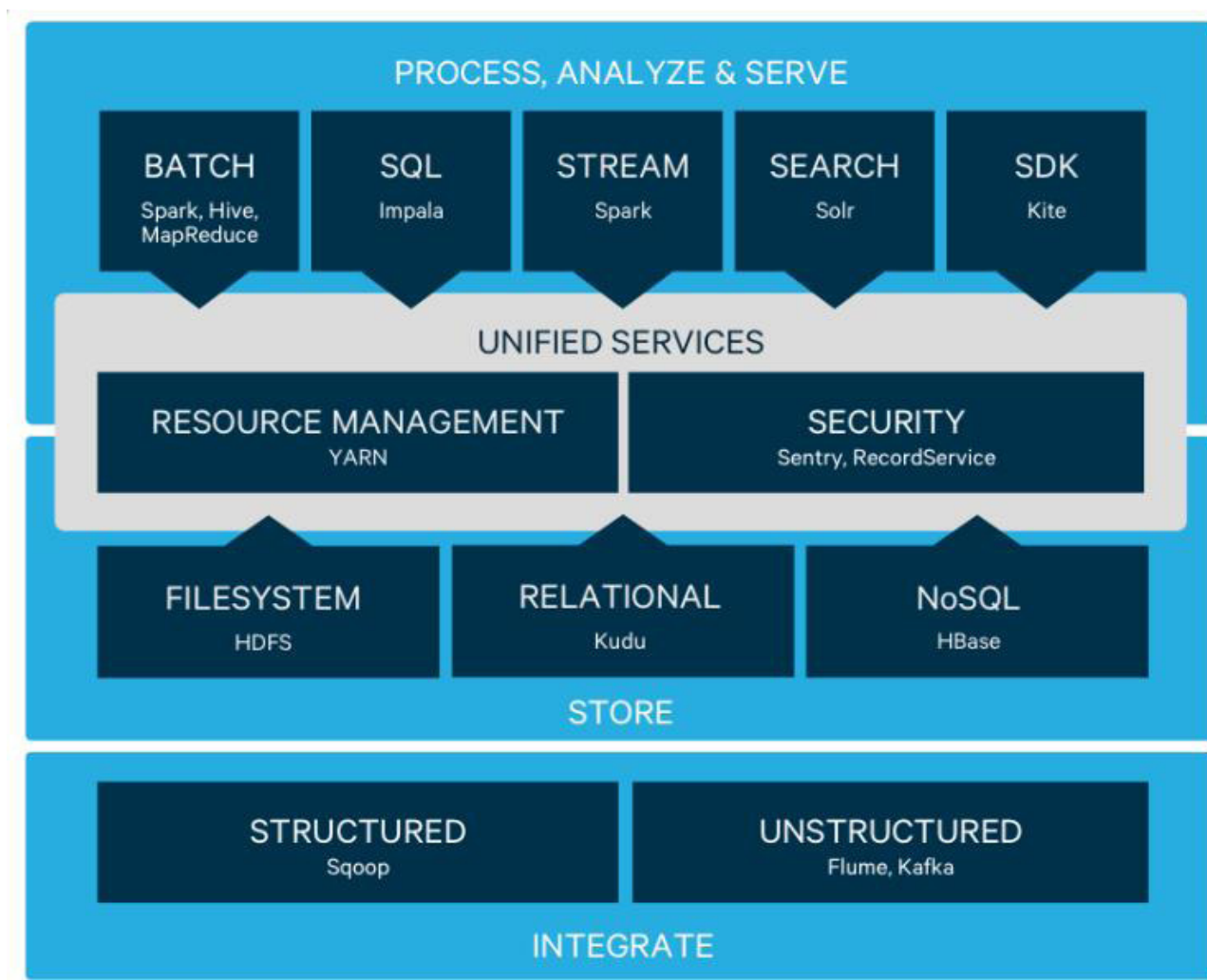
技术篇

现在 Hadoop 在一月发布了 2.7.2 的稳定版，已经从传统的 Hadoop 三驾马车 HDFS，MapReduce 和 HBase 社区发展为 60 多个相关组件组成的庞大生态，其中包含在各大发行版中的组件就有 25 个以上，包括数据存储、执行引擎、编程和数据访问框架等。

Hadoop 在 2.0 将资源管理从 MapReduce 中独立出来变成通用框架后，就从 1.0 的三层结构演变为了现在的四层架构：

- 底层——存储层，文件系统 HDFS
- 中间层——资源及数据管理层，YARN 以及 Sentry 等
- 上层——MapReduce、Impala、Spark 等计算引擎
- 顶层——基于 MapReduce、Spark 等计算引擎的高级封装及工具，如 Hive、Pig、Mahout 等

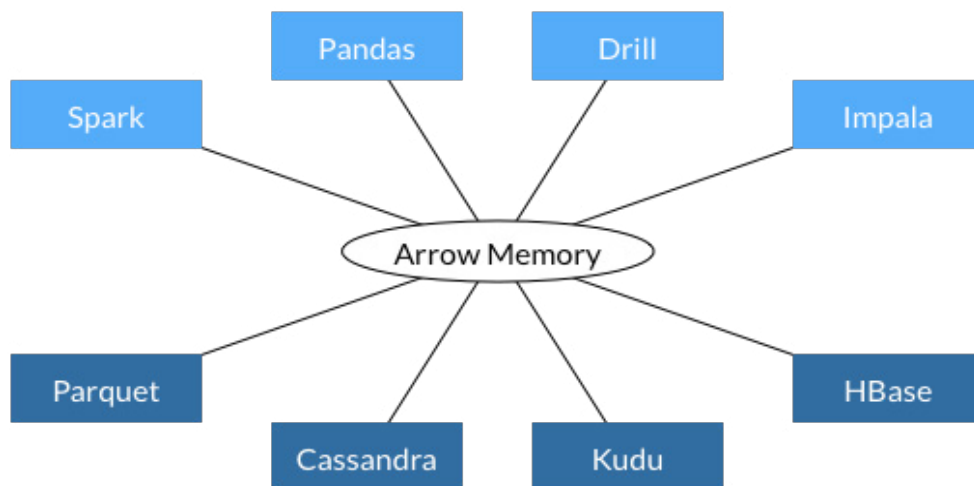
我们欣慰地看到开源文化为 Hadoop 社区和生态带来的蓬蓬发展，但又确实存在一些碎片化和重复化现象。复杂的生态和过多的组件几乎让很多企业仍然等待一个像以前 IBM 一样的巨头厂商能提供标准化的解决方案。不过随着围绕 Hadoop 和 Spark 的生态圈日益稳固，核心会变得稳定得多。



存储层

HDFS 已经成为了大数据磁盘存储的事实标准，用于海量日志类大文件的在线存储。经过这些年的发展，HDFS 的架构和功能基本固化，像 HA、异构存储、本地数据短路访问等重要特性已经实现，在路线图中除了 Erasure Code 已经没什么让人兴奋的 feature。随着 HDFS 越来越稳定，社区的活跃度也越来越低，同时 HDFS 的使用场景也变得成熟和固定，而上层会有越来越多的文件格式封装：列式存储的文件格式，如 Parquet，很好的解决了现有 BI 类数据分析场景；以后还会出现新的存储格式来适应更多的应用场景，如数组存储来服务机器学习类应用等。未来 HDFS 会继续扩展对于新兴存储介质和服务架构的支持。随着数据量的增大，跨机房部署相信也终会在基线版本中实现。基于 HDFS 的存储机制，

将 HBase 作为存储层似乎有点牵强，其底层使用 HDFS 作为文件存储。不过在逻辑角度，还是倾向与将 HBase 定位为存储层或数据访问层，因为其提供了另外一种场景的数据存储和访问方案。2015 年 HBase 发布了 1.0 版本，这也代表着 HBase 走向了稳定。最新



HBase 新增特性包括：更加清晰的接口定义，多 Region 副本以支持高可用读，Family 粒度的 Flush 以及 RPC 读写队列分离等。未来 HBase 不会再添加大的新功能，而将会更多的在稳定性和性能方面进化，尤其是大内存支持、内存 GC 效率等。

Kudu 是 Cloudera 在 2015 年 10 月才对外公布的新的分布式存储架构，与 HDFS 完全独立。其实现参考了 2012 年 Google 发表的 Spanner 论文。鉴于 Spanner 在 Google 内部的巨大成功，Kudu 被誉为下一代分析平台的重要组成，用于处理快速数据的查询和分析，填补 HDFS 和 HBase 之间的空白。其出现将进一步把 Hadoop 市场向传统数据仓库市场靠拢。

另一方面，分布式内存文件系统也在兴起，旨在消除不同任务或不同计算框架间的数据共享时的转化代价，并提供内存缓存以提高热数据处理性能。这一市场以前使用第三方 Redis 或 Memcached，到后来能为分析提供更多原生支持的 Tachyon 或 Ignite，而现在又迎来了新贵 Arrow。

Apache Arrow 项目为列式内存存储的处理和交互提供了规范。目前来自 Apache Hadoop 社区的开发者们致力于将它制定为大数据系统项目的事实性标准。

Arrow 项目受到了 Cloudera、Databricks 等多个大数据巨头公司支持，很多 committer 同时也是其他明星大数据项目（如 HBase、Spark、Kudu 等）的核心开发人员。再考虑到 Tachyon 等似乎还没有找到太多实际接地气的应用场景，Arrow 的高调出场可能会成为未来新的内存分析文件接口标准。

管控层

管控又分为数据管控和资源管控。

随着 Hadoop 集群规模的增大以及对外服务的扩展，如何有效可靠的共享利用资源是管控层需要解决的问题。脱胎于 MapReduce1.0 的 YARN 成为了 Hadoop 2.0 通用资源管理平台。由于占据了 Hadoop 的地利，业界对其在资源管理领域未来的前景非常看好。传统其他资源管理框架如 Mesos，还有现在兴起的 Docker 等都会对 YARN 未来的发展产生影响。如何提高 YARN 性能、如何与容器技术深度融合，如何更好的适应短任务的调度，如何更完整的多租户支持、如何细粒度的资源管控等都是企业实际生产中迫在眉睫的需求，需要 YARN 解决。要让 Hadoop 走得更远，未来 YARN 需要做的工作还很多。

另一方面大数据的安全和隐私越来越多的受到关注。Hadoop 依靠且仅依靠 Kerberos 来实现安全机制，但每一个组件都将进行自己的验证和授权策略。开源社区似乎从来不真正关心安全问题，如果不使用来自 Hortonworks 的 Ranger 或来自 Cloudera 的 Sentry 这样的组件，那么大数据平台基本上谈不上安全可靠。

Cloudera 刚推出的 RecordService 组件使得 Sentry 在安全竞赛中拔得先机。RecordService 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（有点像 Spring，代替了原来 Kite SDK 的作用），让上层的应用和下层存储解耦合的同时、提供了跨组件的可复用数据模型。

计算引擎层

Hadoop 生态和其他生态最大的不同之一就是“单一平台多种应用”的理念了。传的数据库底层只有一个引擎，只处理关系型应用，所以是“单一平台单一应用”；而 NoSQL 市场有上百个 NoSQL 软件，每一个都针对不同的应用场景且完全独立，因此是“多平台多应用”的模式。而 Hadoop 在底层共用一份 HDFS 存储，上层有很多个组件分别服务多种应用场景，如：

- 确定性数据分析：主要是简单的数据统计任务，例如 OLAP，关注快速响应，实现组件有 Impala 等；
- 探索性数据分析：主要是信息关联性发现任务，例如搜索，关注非结构化全量信息收集，实现组件有 Search 等；
- 预测性数据分析：主要是机器学习类任务，例如逻辑回归等，关注计算模型的先进性和计算能力，实现组件有 Spark、MapReduce 等；
- 数据处理及转化：主要是 ETL 类任务，例如数据管道等，关注 IO 吞吐率和可靠性，实现组件有 MapReduce 等

其中，最耀眼的就是 Spark 了。IBM 宣布培养 100 万名 Spark 开发人员，Cloudera 在 One Platform 倡议中宣布支持 Spark 为 Hadoop 的缺省通用任务执行引擎，加上

Hortonworks 全力支持 Spark，我们相信 Spark 将会是未来大数据分析的核心。

虽然 Spark 很快，但现在在生产环境中仍然不尽人意，无论扩展性、稳定性、管理性等方面都需要进一步增强。同时，Spark 在流处理领域能力有限，如果来实现亚秒级或大容量的数据获取或处理需要其他流处理产品。Cloudera 宣布旨在让 Spark 流数据处理技术适用于 80% 的使用场合，就考虑到了这一缺陷。我们确实看到实时分析（而非简单数据过滤或分发）场景中，很多以前使用 S4 或 Storm 等流式处理引擎的实现已经逐渐 Kafka+Spark Streaming 代替。

Spark 的流行将逐渐让 MapReduce、Tez 走进博物馆。

服务层

服务层是包装底层引擎的编程 API 细节，对业务人员提供更高抽象的访问模型，如 Pig、Hive 等。而其中最炙手可热的就是 OLAP 的 SQL 市场了。现在，Spark 有 70% 的访问量来自于 SparkSQL！SQL on Hadoop 到底哪家强？Hive、Facebook 的 Pheonix、Presto、SparkSQL、Cloudera 推的 Impala、MapR 推的 Drill、IBM 的 BigSQL、还是 Pivotal 开源的 HAWQ？

这也许是碎片化最严重的地方了，从技术上讲几乎每个组件都有特定的应用场景，从生态上讲各个厂家都有自己的宠爱，因此 Hadoop 上 SQL 引擎已经不仅仅是技术上的博弈（也因此考虑到本篇中立性，此处不做评论）。可以遇见的是，未来所有的 SQL 工具都将被整合，有些产品已经在竞争中逐渐落伍，我们期待市场的选择。

周边的工具更是百花齐放，最重要的莫过于可视化、任务管理和数据管理了。

有很多开源工具都支持基于 Hadoop 的查询程序编写以及即时的图形化表示，如 HUE、Zeppelin 等。用户可以编写一些 SQL 或 Spark 代码以及描述代码的一些标记，并指定可视化的模版，执行后保存起来，就可供其他人复用，这种模式也被叫做“敏捷 BI”。这个领域的商业产品更是竞争激烈，如 Tableau、Qlik 等。

调度类工具的鼻祖 Oozie 能实现几个 MapReduce 任务串连运行的场景，后来的 Nifi 及 Kettle 等其他工具则提供了更加强化的调度实现，值得一试。

毫无疑问，相对与传统的数据库生态，Hadoop 的数据治理相对简单。Atlas 是 Hortonworks 新的数据治理工具，虽然还谈不上完全成熟，不过正取得进展。Cloudera 的 Navigator 是 Cloudera 商业版本的核心，汇聚了生命周期管理、数据溯源、安全、审计、SQL 迁移工具等一系列功能。Cloudera 收购 Explain.io 以后将其产品整合为 Navigator Optimizator 组件，能帮助用户把传统的 SQL 应用迁移到 Hadoop 平台并提供优化建议，可以节省数人月的工作量。

算法及机器学习

实现基于机器学习的自动的智能化数据价值挖掘是大数据和 Hadoop 最诱人的愿景了，也是很多企业对大数据平台的最终期望。随着可获得的数据越来越多，未来大数据平台的价值更多的取决于其计算人工智能的程度。

现在机器学习正慢慢跨出象牙塔，从一个少部分学术界人士研究的科技课题变成很多企业正在验证使用的数据分析工具，而且已经越来越多的进入我们的日常生活。

机器学习的开源项目除了之前的 Mahout、MLlib、Oryx 等，今年发生了很多令人瞩目的大事，迎来了数个明星巨头的重磅加入：





- 2015 年 1 月，Facebook 开源前沿深度学习工具 “Torch”。
- 2015 年 4 月，亚马逊启动其机器学习平台 Amazon Machine Learning，这是一项全面的托管服务，让开发者能够轻松使用历史数据开发并部署预测模型。
- 2015 年 11 月，谷歌开源其机器学习平台 TensorFlow。
- 同一月，IBM 开源 SystemML 并成为 Apache 官方孵化项目。
- 同时，微软亚洲研究院将分布式机器学习工具 DMTK 通过 Github 开源。DMTK 由一个服务于分布式机器学习的框架和一组分布式机器学习算法组成，可将机器学习算法应用到大数据中。
- 2015 年 12 月，Facebook 开源针对神经网络研究的服务器 “Big Sur”，配有高性能图形处理单元（GPUs），转为深度学习方向设计的芯片。

产业篇

现在使用 Hadoop 的企业以及靠 Hadoop 赚钱的企业已经成千上万。几乎大的企业或多或少的已经使用或者计划尝试使用 Hadoop 技术。就对 Hadoop 定位和使用不同，可以将 Hadoop 业界公司划分为四类：

- 第一梯队：这类公司已经将 Hadoop 当作大数据战略武器。
- 第二梯队：这类公司将 Hadoop 产品化。
- 第三梯队：这类公司创造对 Hadoop 整体生态系统产生附加价值的产品。
- 第四梯队：这类公司消费 Hadoop，并给规模比第一类和第二类小的公司提供基于 Hadoop 的服务。

时至今日，Hadoop 虽然在技术上已经得到验证、认可甚至已经到了成熟期。但与之对应的以 Hadoop 为代表的大数据基础平台产业界仍然还在迷茫和探索。虽然大数据的市场很大，但单纯 Hadoop 产品和服务市场，和传统关系型事务数据库市场相比还不到 1%。

Class 1	Class 2	Class 3	Class 4
			

虽然很多高调的创业公司上线也拿到引人注目的风险投资，但只是到达大数据部署和早期成熟阶段。

其中最能代表 Hadoop 发展轨迹的莫过于商业公司推出的 Hadoop 发行版了。自从 2008 年 Cloudera 成为第一个 Hadoop 商业化公司，并在 2009 年推出第一个 Hadoop 发行版后，很多大公司也加入了做 Hadoop 产品化的行列。“发行版”这个词是开源文化特有的符号，看起来任何一个公司只要将开源代码打个包，再多多少少加个佐料就能有一个“发行版”，然而背后是对海量生态系统组件的价值筛选、兼容和集成保证以及支撑服务。

2012 年以前的发行版基本为对 Hadoop 打补丁为主，出现了好几个私有化 Hadoop 版本，所折射的是 Hadoop 产品在质量上的缺陷。同期 HDFS、HBase 等社区的超高活跃度印证了这个事实。

而之后的公司更多是工具、集成、管理，所提供的不是“更好的 Hadoop”而是如何更好的用好“现有”的 Hadoop。

2014 年以后，随着 Spark 和其他 OLAP 产品的兴起，折射出来是 Hadoop 善长的离线场景等已经能够很好的解决，希望通过扩大生态来适应新的硬件和拓展新的市场。

对于开源产品，一直有拥抱开源和提供私有化这两种流派，商业模式要么是提供技

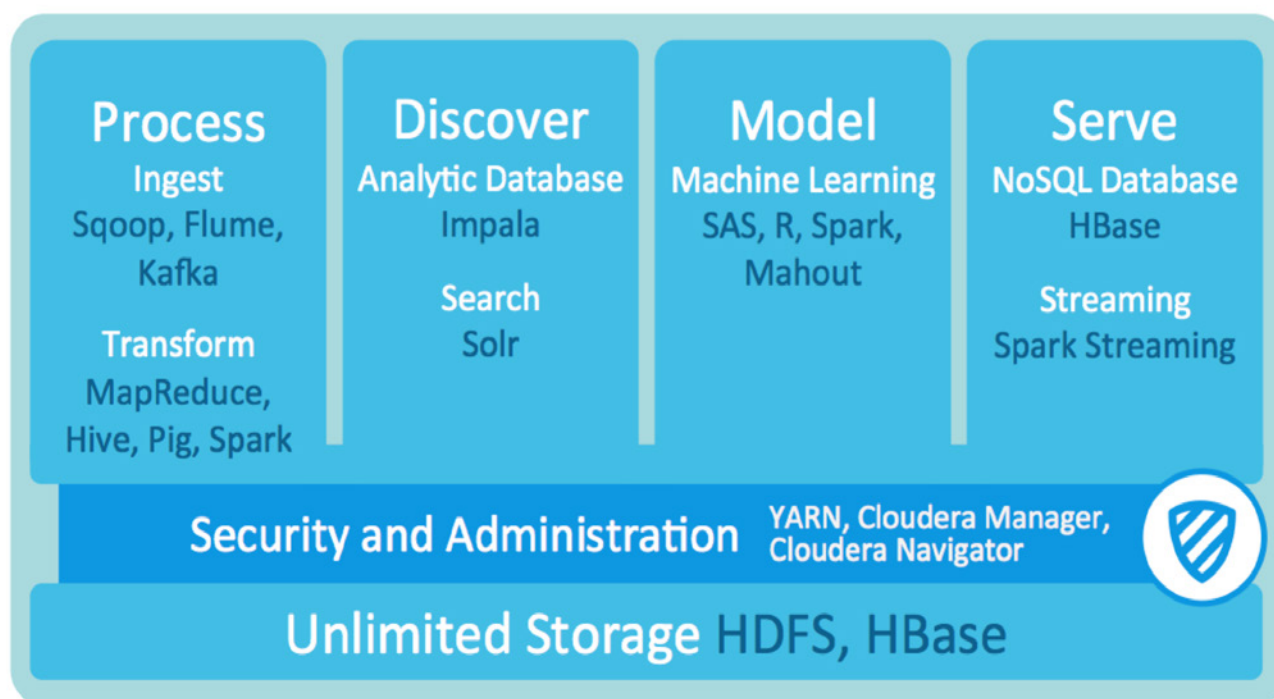
术支持服务，要么是提供私有化的增强版本。对于 Hadoop 的产品化也不例外。但就目前的情况看，曾经私有化 Hadoop 版本的代表 Pivotal 和 Intel 都已经放弃，IBM 几乎事实上放弃了自有 Hadoop，再考虑到之前 Taobao 放弃私有 Hadoop 路线，似乎证明了在像 Hadoop 这样生态庞大、发展迅速的产品，与局部私有增强带来的好处相比，长期独立站在世界的对立面并不断地与整体社区版本做代码合并似乎是越来越不可承受之痛。

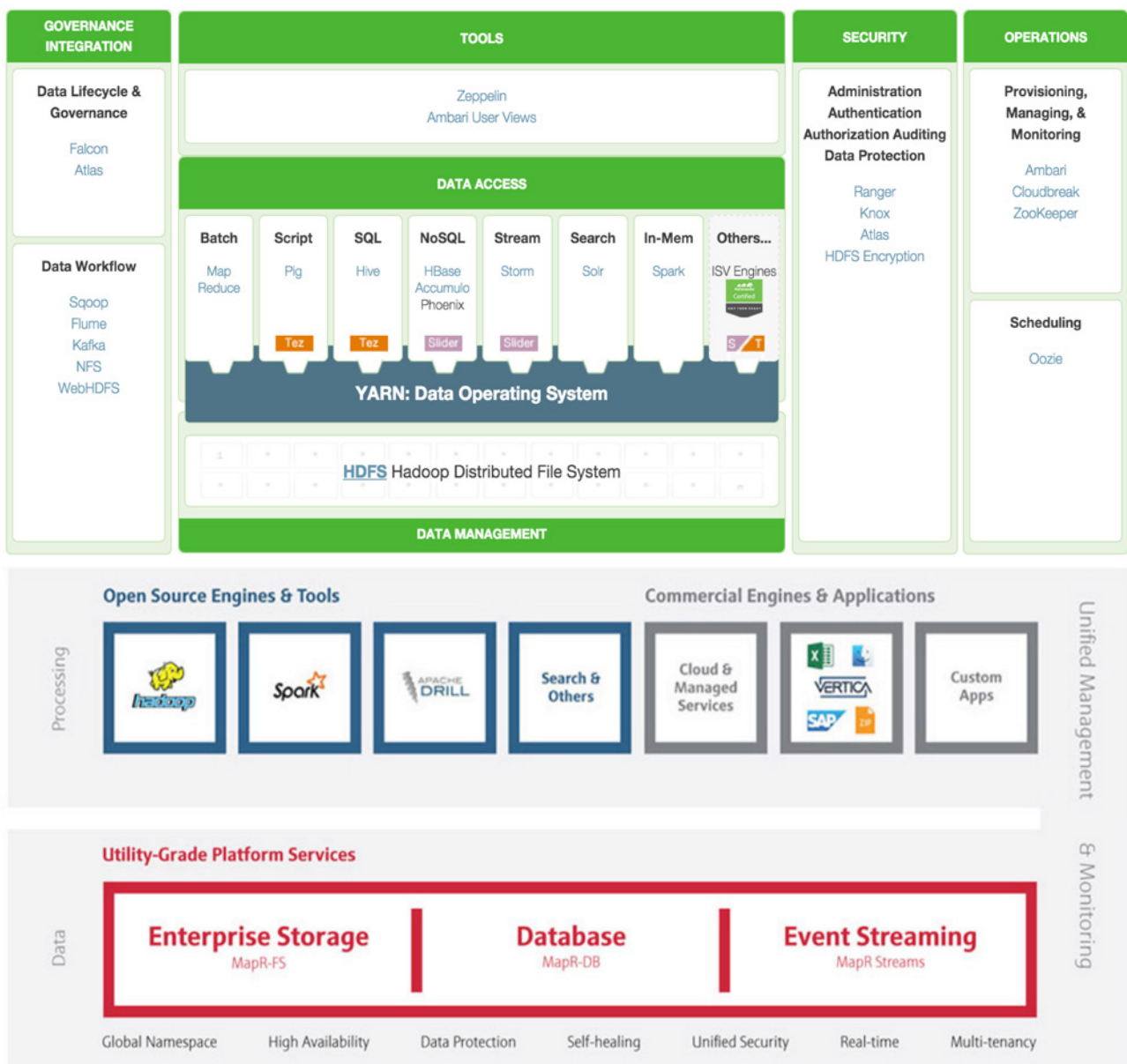
如今，主要的 Hadoop 产品化厂商只剩下了三家厂商，并且使用了三种截然不同的商业模式。过去几年，虽然尚无任何数据现实三家厂商实现财务盈利，但在资本市场都名声赫赫，且不断收购扩张。从另外一个角度说明，Hadoop 市场仍然再初级发展阶段。

Cloudera 提出了 Hybrid Open Source 的架构：核心组件名称叫 CDH (Cloudera's Distribution including Apache Hadoop)，开源免费并与 Apache 社区同步，用户无限制使用，保证 Hadoop 基本功能持续可用，不会被厂家绑定；数据治理和系统管理组件闭源且需要商业许可，支持客户可以更好更方便的使用 Hadoop 技术，如部署安全策略等。Cloudera 也在商业组件部分提供在企业生产环境中运行 Hadoop 所必需的运维功能，而这些功能并不被开源社区所覆盖，如无宕机滚动升级、异步灾备等。

Hortonworks 采用了 100% 完全开源策略，产品名称为 HDP (Hortonworks Data Platform)。所有软件产品开源，用户免费使用，Hortonworks 提供商业的技术支持服务。与 CDH 相比，管理软件使用开源 Ambari，数据治理使用 Atlas，安全组件使用 Ranger 而非 Sentry，SQL 继续紧抱 Hive 大腿。

MapR 采用了传统软件厂商的模式，使用私有化的实现。用户购买软件许可后才能使用。其 OLAP 产品主推 Drill，又不排斥 Impala。





不过，三家厂商的处境有所不同。Hortonworks 虽然业绩不断进步，但直到现在仍未能实现盈利。上市后的市值未能走高，可见市场对于 Hadoop 纯服务公司的未来价值增值期望不高。而另厢 Cloudera 估值近 50 亿美金，最后一轮收到的来自 Intel 的 7.8 亿美元已经超过 Hortonworks 最近 6.8 亿的估值，被誉为 2016 年最有希望上市的高科技软件公司。

现在，Cloudera 和 Hortonworks 的定位已经不是 Hadoop 发行版软件开发商了，而是现代化的数据管理和分析平台建设厂家，足见其“狼子野心”。

另一方面，传统企业数据管理软件巨头仍然对既有格局信心满满，对于 Hadoop 产品还是观望态度，通常 OEM 发行版厂商而非自己研发 Hadoop 产品，如 Oracle、Dell，Teradata 公司的大数据一体机都是采用 OEM 化 Cloudera 的企业版本产品。

现在主流的公有云如 AWS、Azure 等都已经是在原有提供虚拟机的 IaaS 服务之外，提供基于 Hadoop 的 PaaS 云计算服务。未来这块市场的发展将超过私有 Hadoop 部署。

作为大数据基础设施平台的 Hadoop 虽然是技术上是核心，但商业上还只是整个大数据生态系统中非常小的部分。

应用篇

Hadoop 平台释放了前所未有的计算能力，同时大大降低了计算成本。底层核心基础设施生产力的发展，必然带来的是大数据应用层的迅速建立。

对于 Hadoop 上的应用大致可以分为以下两类。

IT 优化

将已经实现的应用和业务搬迁到 Hadoop 平台，以获得更多的数据、更好的性能或更低的成本。通过提高产出比、降低生产和维护成本等方式为企业带来好处。

这几年 Hadoop 在数个此类应用场景中已经被证明是非常适合的解决方案，主要包括一下方案。

- 历史日志数据在线查询：传统的解决方案将数据存放在昂贵的关系型数据库中，不仅成本高、效率低，而且无法满足在线服务时高并发的访问量。以 HBase 为底层存储和查询引擎的架构非常适合有固定场景（非 ad hoc）的查询需求，如航班查询、个人交易记录查询等等。现在已经成为在线查询应用的标准方案，中国移动在企业技术指导意见中明确指明使用 HBase 技术来实现所有分公司的清账单查询业务。
- ETL 任务：不少厂商已经提供了非常优秀的 ETL 产品和解决方案，并在市场中得到了广泛的应用。然而在大数据的场景中，传统 ETL 遇到了性能和 QoS 保证上的严重挑战。多数 ETL 任务是轻计算重 IO 类型的，而传统的 IT 硬件方案，如承载数据库的小型计算机，都是为计算类任务设计的，即使使用了最新的网络技术，IO 也顶多到达几十 GB。采用分布式架构的 Hadoop 提供了完美的解决方案，不仅使用 share-nothing 的 scale-out 架构提供了能线性扩展的无限 IO，保证了 ETL 任务的效率，同时框架已经提供负载均衡、自动 Fail0ver 等特性保证了任务执行的可靠性和可用性。
- 数据仓库 offload：传统数据仓库中有很多离线的批量数据处理业务，如日报表、月报表等，占用了大量的硬件资源。而这些任务通常又是 Hadoop 所善长的。

经常被问到的一个问题就是，Hadoop 是否可以代替数据仓库，或者说企业是否可以使用免费的 Hadoop 来避免采购昂贵的数据仓库产品。数据库界的泰斗 Mike Stonebroker

在一次技术交流中说：数据仓库和 Hadoop 所针对的场景重合型非常高，未来这两个市场一定会合并。我们相信在数据仓库市场 Hadoop 会迟早替代到现在的产品，只不过，那时候的 Hadoop 已经又不是现在的样子了。就现在来讲，Hadoop 还只是数据仓库产品的一个补充，和数据仓库一起构建混搭架构为上层应用联合提供服务。

业务优化

在 Hadoop 上实现原来尚未实现的算法、应用，从原有的生产线中孵化出新的产品和业务，创造新的价值。通过新业务为企业带来新的市场和客户，从而增加企业收入。

Hadoop 提供了强大的计算能力，专业大数据应用已经在几乎任何垂直领域都很出色，从银行业（反欺诈、征信等）、医疗保健（特别是在基因组学和药物研究），到零售业、服务业（个性化服务、智能服务，如 UBer 的自动派车功能等）。

在企业内部，各种工具已经出现，以帮助企业用户操作核心功能。例如，大数据通过大量的内部和外部的数据，实时更新数据，可以帮助销售和市场营销弄清楚哪些客户最有可能购买。客户服务应用可以帮助个性化服务；HR 应用程序可帮助找出如何吸引和留住最优秀的员工等。

不过，互联网以外的传统行业内部，现在大数据的应用和业务普遍尚处在探索阶段，虽然不少企业已经从数据和深度挖掘数据价值中得到的甜头，但更多的企业在实现数据分析时缺少业务的指导和支撑，可量化可规模化的大数据业务闭环尚未建立，更多是站在改善用户体验等角度改善现有运营效率。

虽然行业性的大数据新兴业务解决方案尚未出现，但很多新兴的公司信心满满的进入这个市场，并收到资本市场的热捧，或提供辅助工具，或提供 Big Data-as-a-Service 服务，或提供基于大数据的商业设计咨询，目的是适应大数据以及分析专家和需要他们所服务客户的需求，包括大数据准备评估、路线图、预测操作界面、算法和一些针对特定市场和企业消费分析解决方案等等。如 Palantir、营销的大数据分析工具 Qubit、针对 CRM 领域的人工智能 Neokami，等等。

为什么 Hadoop 如此成功

这个问题似乎是个马后炮，但当我们今天惊叹于 Hadoop 在短短 10 年时间取得如此统治性地位的时候，确实会自然而然地思考为什么这一切会发生。基于与同期其他项目的比较，我们认为有很多因素的综合作用造就了这一奇迹。

- 技术架构：Hadoop 推崇的本地化计算理念，其现在可扩展性、可靠性上的优势，

以及有弹性的多层级架构等都是领先其他产品而获得成功的内在因素。没有其他任何一个这样复杂的系统能快速的满足不断变化的用户需求。

- 硬件发展：摩尔定律为代表的 scale up 架构遇到了技术瓶颈，不断增加的计算需求迫使软件技术不得不转到分布式方向寻找解决方案。同时，PC 服务器技术的发展使得像 Hadoop 这样使用廉价节点组群的技术变为可行，同时还具有很诱人的性价比优势。
- 工程验证：Google 发表 GFS 和 MapReduce 论文时已经在内部有了可观的部署和实际的应用，而 Hadoop 在推向业界之前已经在 Yahoo 等互联网公司验证了工程上的可靠性和可用性，极大的增加了业界信心，从而迅速被接纳流行。而大量的部署实例又促进了 Hadoop 的发展成熟。
- 社区推动：Hadoop 生态一直坚持开源开放，友好的 Apache 许可基本消除了厂商和用户的进入门槛，从而构建了有史以来最大最多样化最活跃的开发社区，持续地推动着技术发展，让 Hadoop 超越了很多以前和同期的项目。
- 关注底层：Hadoop 的根基是打造一个分布式计算框架，让应用程序开发人员更容易的工作。业界持续推动的重点一直在不断夯实底层，并在诸如资源管理和安全领域等领域不断开花结果，为企业生产环境部署不断扫清障碍。

下一代分析平台

过去的十年中 Apache Hadoop 社区以疯狂的速度发展，现在俨然已经是事实上的大数据平台标准。我们经历了 Hadoop 实现这一愿景的巨大进步，见证了 Hadoop 如何从一个存储和批处理架构慢慢转变为一个现代化的、模块化的数据平台。三年前，Hadoop 通过 Impala 等分析型 SQL 引擎实现了互动的数据发现。两年前，Cloudera 迎来了 Apache Spark，并将其视为 Hadoop 生态系统的下一代数据处理层，能同时处理各种批次和流工作负载，为开发人员提供更好的易用性和更高的性能。

但仍有更多的工作要做！

大数据应用未来的价值在于预测，而预测的核心是分析。下一代的分析平台会是什么样呢？它必定会面临、同时也必须要解决以下的问题：

更多更快的数据。未来的大数据来源更多的是来自物联网（IoT，Internet of Things），将有超过 160 亿的设备联网并不断产生数据。数据量更大，而且对数据处理的实时性要求的更高。

更新的硬件特性及架构。Hadoop、Spark 等技术兴起的重要推动原因都是硬件的发展。现在摩尔定律已经退出历史舞台，未来硬件架构可能呈现多样化发展，可靠性越来越高，

存储和计算成本继续降低，内存的容量和速度越来越快，持久化或非挥发性存储的发展会对现有的存储设计带来新的技术和架构。

更高级的分析。技术的发展背后总是业务需求的驱动。但现在的大数据项目多是初级阶段的 IT 系统，目的是解决目前 IT 有限的能力限制和成本压力，并非针对业务创造新的价值，甚至没有对业务有直接互动和反馈。未来的需求是要使用实时数据建立更好的模型，使用机器学习等高级数据分析技术，能够改善用户体验、优化业务运营，实现大数据业务的闭环。

更安全。随着企业希望能把手里的数据资源开放变现，但频发的安全事故又让企业驻足不前，很少有人敢冒风险进行开放尝试。需要通过安全机制实时地保护用户和企业的资产；通过行为分析和稽查保证流程的正确性和结果的可信性。

因此，未来的几年，我们会继续见证“后 Hadoop 时代”的下一代企业大数据平台。

1. 内存计算时代的来临。随着高级分析和实时应用的增长，对处理能力提出了更高的要求，数据处理重点从 IO 重新回到 CPU。以内存计算为核心的 Spark 将代替以 IO 吞吐为核心的 MapReduce 成为分布式大数据处理的缺省通用引擎。做为既支持批处理有支持准实时流处理的通用引擎，Spark 将能满足 80% 以上的应用场景。Cloudera 公司近日公布了 One Platform 的倡议，推动 Spark 成为 Hadoop 的默认数据处理引擎。为了最终取代 MapReduce，Cloudera 集中力量推动解决 Spark 现在企业大规模应用时在四个关键领域仍然存在的短板：管理，安全，规模和流。

然而，Spark 毕竟核心还是批处理，擅长迭代式的计算，但并不能满足所有的应用场景。其他为特殊应用场景设计的工具会对其补充，包括：

- OLAP。OLAP，尤其是聚合类的在线统计分析应用，对于数据的存储、组织和处理都和单纯离线批处理应用有很大不同。以 Impala 为代表的 SQL-on-Hadoop 引擎借鉴了传统数据处理和 MPP 等技术，底层使用 HDFS 存储，是传统 BI 系统很好的替代方案候选。
- 知识发现。与传统应用解决已知问题不同，大数据的价值在于发现并解决未知问题。因此，要最大限度地发挥分析人员的智能，将数据检索变为数据探索。Apache Solr 项目是一个功能丰富的可扩展的搜索解决方案，内包括了 Apache Lucene 和 Apache Tika。Cloudera 的 Search 将 Solr 集成到了 Hadoop，并使用高度自动化的流水线为 Hadoop 上的数据创建索引，在提高部署效率的同时，提供了更加直观方便的大数据平台搜索引擎。

2. 统一数据访问管理。现在的数据访问由于数据存储的格式不同、位置不同，用户需要使用不同的接口、模型甚至语言。同时，不同的数据存储粒度都带来了在安全控制、管理治理上的诸多挑战。未来的趋势是将底层部署运维细节和上层业务开发进行隔离，

因此，平台需要系统如下的功能保证：

- 安全。能够大数据平台上实现和传统数据管理系统中相同口径的数据管理安全策略，包括跨组件和工具的一体化的用户权利管理、细粒度访问控制、加解密和审计。
- 统一数据模型。通过抽象定义的数据描述，不仅可以统一管理数据模型、复用数据解析代码，还可以对于上层处理屏蔽底层存储的细节，从而实现开发 / 处理与运维 / 部署的解耦。

Cloudera 最近发布的 RecordService 正是为此而生。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。RecordService 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。DFS 执行的新的核心服务。同时 RecordService 屏蔽了底层存储细节，向上暴露基于记录的面向对象的数据描述，为编程人员提供了更好的封装和抽象。

3. 简化实时应用。现在用户不仅关心如何实时的收集数据，而且关心同时尽快的实现数据可见和分析结果上线。无论是以前的 delta 架构还是现在 lambda 架构等，都希望能够有一种解决快速数据的方案，使用 HDFS 和 HBase 的混合体，在快速更新数据的同时进行快速分析，然而结果复杂的架构令人望而却步，无论开发还是运维都不胜其繁。Cloudera 最新公开的 Kudu 虽然还没有进入产品发布，但却是现在解决这个问题可能的最佳方案：采用了使用单一平台简化了快速数据的“存取用”实现，是未来日志类数据分析的新的解决方案。

最近新面世的这些项目将彻底改变 Hadoop 的存储架构，进一步巩固其安全基础，推动 Hadoop 不断发展和扩大，成为新一代的现代分析的领先平台。

下一个十年

Hadoop 的未来是什么样的？10 年以后大数据是不是已经进博物馆了？会不会有一个新公司成为数据管理界的新的巨头，犹如今日的 Oracle？会不会有高富帅的企业已经有百万、千万甚至更多机器组成的数据中心？

有许多的可能，但我们相信 Hadoop 所“发明”的分布式计算框架仍然会是大数据的核心标志。

10 年前谁也没有料想到 Hadoop 能取得今天这样的成就，而如今一切均在眼前。Hadoop 之父 Doug Cutting 则认为 Hadoop 正处于蓬勃的发展期，而且这样的蓬勃发展至少还可以持续几十年。

10 年以后的 Hadoop 应该只是一个生态和标准的“代名词”了，下层的存储层不只

是 HDFS、HBase 和 Kudu 等现有的存储架构，上层的处理组件更会像 app store 里的应用一样多，任何第三方都可以根据 Hadoop 的数据访问和计算通信协议开发出自己的组件，用户市场中根据自己数据的使用特性和计算需求选择相应的组件自动部署。

当然，有一些明显的趋势必然影响着 Hadoop 的前进。

- 云计算

现在 50% 的大数据任务已经运行在云端，在 3 年以后这个比例可能会上升到 80%。Hadoop 在公有云的发展要求更加有保障的本地化支持。

- 硬件

快速硬件的进步会迫使社区重新审视 Hadoop 的根基。回顾历史，任何一次硬件的革新都会翻开软件业的新篇章。现在 CPU 发展摩尔定律已经退出历史舞台，但新型的硬件，如 3D point 等即将登场企业数据中心。现在虽然尚未有与之相应的软件产品，但必然会出现，而 Hadoop 社区也绝不会袖手旁观。

- 物联网

物联网的发展会带来海量的、分布的和分散的数据源。Intel CEO 预测 2020 年将有 500 亿设备联网，会带来 50 万亿 GB 的数据；世界经济论坛预测 2022 年将有 1 万亿传感器入网；按照梅特卡夫定律，5 年后全球 IoT 自动服务网的总体价值将是现在的 517 倍。Hadoop 将适应这种发展。

以后的十年会发生什么？以下是笔者的一些猜想。

- SQL 和 NoSQL 市场会合并，NewSQL 和 Hadoop 技术相互借鉴而最终走向统一，Hadoop 市场和数据仓库市场会合并，然而产品碎片化会继续存在。
- Hadoop 与其他资源管理技术和云平台集成，融合 docker 和 unikernal 等技术统一资源调度管理，提供完整多租户和 QoS 能力，企业数据分析中心合并为单一架构。
- 企业大数据产品场景化。以后直接提供产品和技术的公司趋于成熟并且转向服务。越来越多的新公司提供的是行业化、场景化的解决方案，如个人网络征信套件以及服务。
- 大数据平台的场景“分裂”。与现在谈及大数据言必称 Hadoop 以及某某框架不同，未来的数据平台将根据不同量级的数据（从几十 TB 到 ZB）、不同的应用场景（各种专属应用集群）出现细分的阶梯型的解决方案和产品，甚至出现定制化一体化产品。
- 无论 10 年或 20 年后的 Hadoop 看起来像什么样，无可质疑的是由于数据量、数据种类和数据速度的提升会带来更强大的使用用例。如何把原始数据转化为可执行的洞察力将是最清晰最有力的推动力量。正如 Cloudera 的首席科学家、

Hadoop 的创始人 Doug Cutting 所说：“我们在本世纪取得的大部分进展将来自于对所产生的数据的理解的增加。”

后记

笔者水平有限，加之时间紧迫，肤浅粗糙之处，还请各位读者原谅和指教。文中有些内容引自网络，某些出处未能找到，还请原作者原谅。

Hadoop 的组件生态组件太多，参加 Cloudera 的全套 Hadoop 课程就需要花费 1 个月以上的时间，让人“累觉不爱”。本文中只是蜻蜓点水，很多东西尚未详述，请参见相关产品手册。欢迎访问网站，观看 Doug Cutting 关于 Hadoop 十年的视屏。

大数据的明天是美好的，未来 Hadoop 一定是企业软件的必备技能，希望我们能一起见证。

陈飏，Cloudera 售前技术经理、行业领域顾问、资深方案架构师，原 Intel Hadoop 发行版核心开发人员。2006 年加入 Intel 编译器部门从事服务器中间件软件开发，擅长服务器软件调试与优化，曾带领团队开发出世界上性能领先的 XSLT 语言处理器。2010 年后开始 Hadoop 产品开发及方案顾问，先后负责 Hadoop 产品化、HBase 性能调优，以及行业解决方案顾问，已在交通、通信等行业成功实施并支持多个上百节点 Hadoop 集群。

**北大美女博士：如何将大数据建模
在商业领域玩转得风声水起**



Hadoop YARN 在 Hulu 的成功实践

作者 董西成

Hadoop YARN 生态系统介绍

为了能够对集群中的资源进行统一管理和调度，Hadoop 2.0 引入了数据操作系统 YARN。YARN 的引入，大大提高了集群的资源利用率，并降低了集群管理成本。首先，YARN 允许多个应用程序运行在一个集群中，并将资源按需分配给它们，这大大提高了资源利用率，其次，YARN 允许各类短作业和长服务混合部署在一个集群中，并提供了容错、资源隔离及负载均衡等方面的支持，这大大简化了作业和服务的部署和管理成本。

YARN 总体上采用 master/slave 架构，如图 1 所示，其中，master 被称为 ResourceManager，slave 被称为 NodeManager，ResourceManager 负责对各个 NodeManager 上的资源进行统一管理和调度。当用户提交一个应用程序时，需要提供一个用以跟踪和管理这个程序的 ApplicationMaster，它负责向 ResourceManager 申请资源，并要求 NodeManager 启动可以占用一定资源的 Container。由于不同的 ApplicationMaster 被分布到不同的节点上，并通过一定的隔离机制进行了资源隔离，因此它们之间不会相互影响。

YARN 中的资源管理和调度功能由资源调度器负责，它是 Hadoop YARN 中最核心的组件之一，是 ResourceManager 中的一个插拔式服务组件。YARN 通过层级化队列的方式组织和划分资源，并提供了多种多租户资源调度器，这种调度器允许管理员按照应用需求对用户或者应用程序分组，并为不同的分组分配不同的资源量，同时通过添加各种约束防止单个用户或者应用程序独占资源，进而能够满足各种 QoS 需求，典型代表是 Yahoo! 的 Capacity Scheduler 和 Facebook 的 Fair Scheduler。

YARN 作为一个通用数据操作系统，既可以运行像 MapReduce、Spark 这样的短作业，也可以部署像 Web Server、MySQL Server 这种长服务，真正实现一个集群多用途，这样的集群，我们通常称为轻量级弹性计算平台，说它轻量级，是因为 YARN 采用了 cgroups

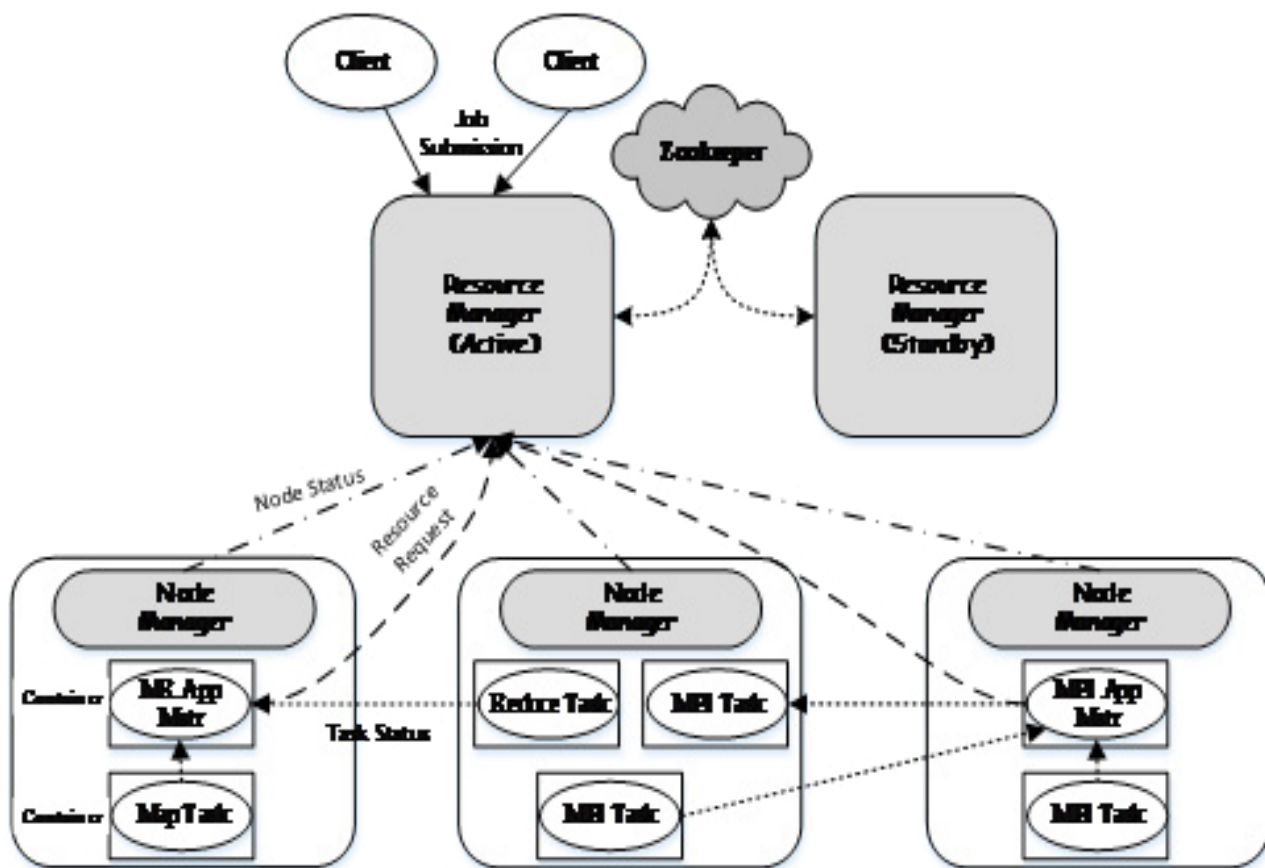


图 1 Apache YARN 的基本架构

轻量级隔离方案，说它弹性，是因为 YARN 能根据各种计算框架或者应用的负载或者需求调整它们各自占用的资源，实现集群资源共享，资源弹性收缩。

Hadoop YARN 在异构集群中的应用

从 2.6.0 版本开始，YARN 引入了一种新的调度策略：基于标签的调度机制。该机制的主要引入动机是更好地让 YARN 运行在异构集群中，进而更好地管理和调度混合类型的应用程序。

什么是基于标签的调度

故名思议，基于标签的调度是一种调度策略，就像基于优先级的调度一样，是调度器中众多调度策略中的一种，可以跟其他调度策略混合使用。该策略的基本思想是：用户可为每个 NodeManager 打上标签，比如 highmem, highdisk 等，以作为 NodeManager 的基本属性；同时，用户可以为调度器中的队列设置若干标签，以限制该队列只能占用包含对应标签的节点资源，这样，提交到某个队列中的作业，只能运行在特定一些节点上。通过打标签，用户可将 Hadoop 分成若干个子集群，进而使得用户可将应用程序运行到符

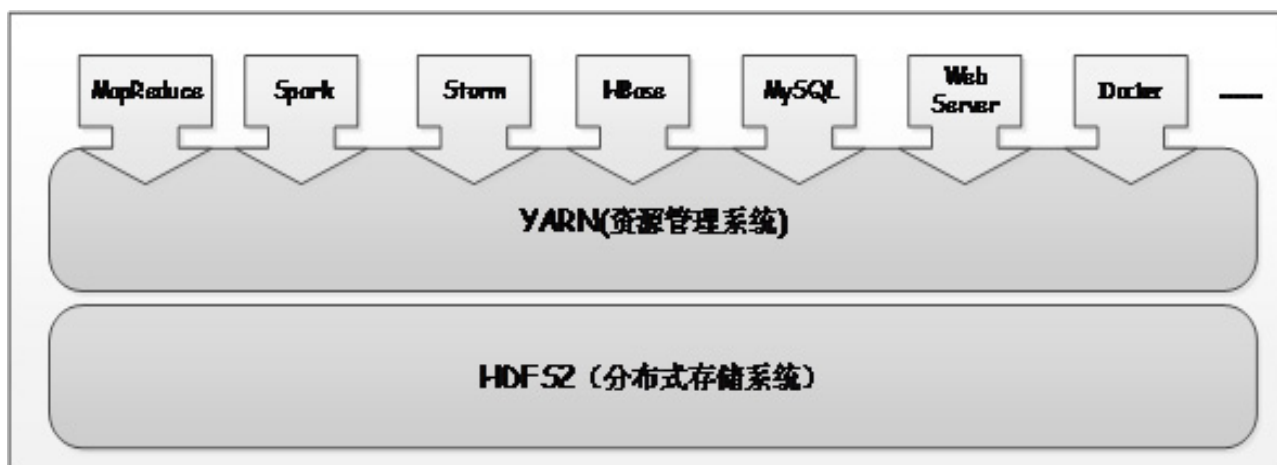


图 2 以 YARN 为核心的生态系统

合某种特征的节点上，比如可将内存密集型的应用程序（比如 Spark）运行到大内存节点上。

Hulu 应用案例

基于标签的调度策略在 Hulu 内部有广泛的应用。之所以启用该机制，主要出于以下三点考虑：

（1）集群是异构的。在 Hadoop 集群演化过程中，后来新增机器的配置通常比旧机器好，这使得集群最终变为一个异构的集群。Hadoop 设计之初众多设计机制假定集群是同构的，即使发展到现在，Hadoop 对异构集群的支持仍然很不完善，比如 MapReduce 推测执行机制尚未考虑异构集群情形。

（2）应用是多样化的。Hulu 在 YARN 集群之上同时部署了 MapReduce、Spark、Spark Streaming、Docker Service 等多种类型的应用程序。当在异构集群混合运行多类应用程序时，经常发生由于机器配置不一导致并行化任务完成时间相差较大的情况，这非常不利于分布式程序的高效执行。此外，由于 YARN 无法进行完全的资源隔离，多个应用程序混合运行在一个节点上容易相互干扰，对于低延迟类型的应用通常是难以容忍的。

（3）个性化机器需求。由于对特殊环境的依赖，有些应用程序只能运行在大集群中的特定节点上。典型的代表是 spark 和 docker，spark MLLib 可能用到一些 native 库，为了防止污染系统，这些库通常只会安装在若干节点上；docker container 的运行依赖于 docker engine，为了简化运维成本，我们只会让 docker 运行在若干指定的节点上。

为了解决以上问题，Hulu 在 Capacity Scheduler 基础上启用了基于标签的调度策略。

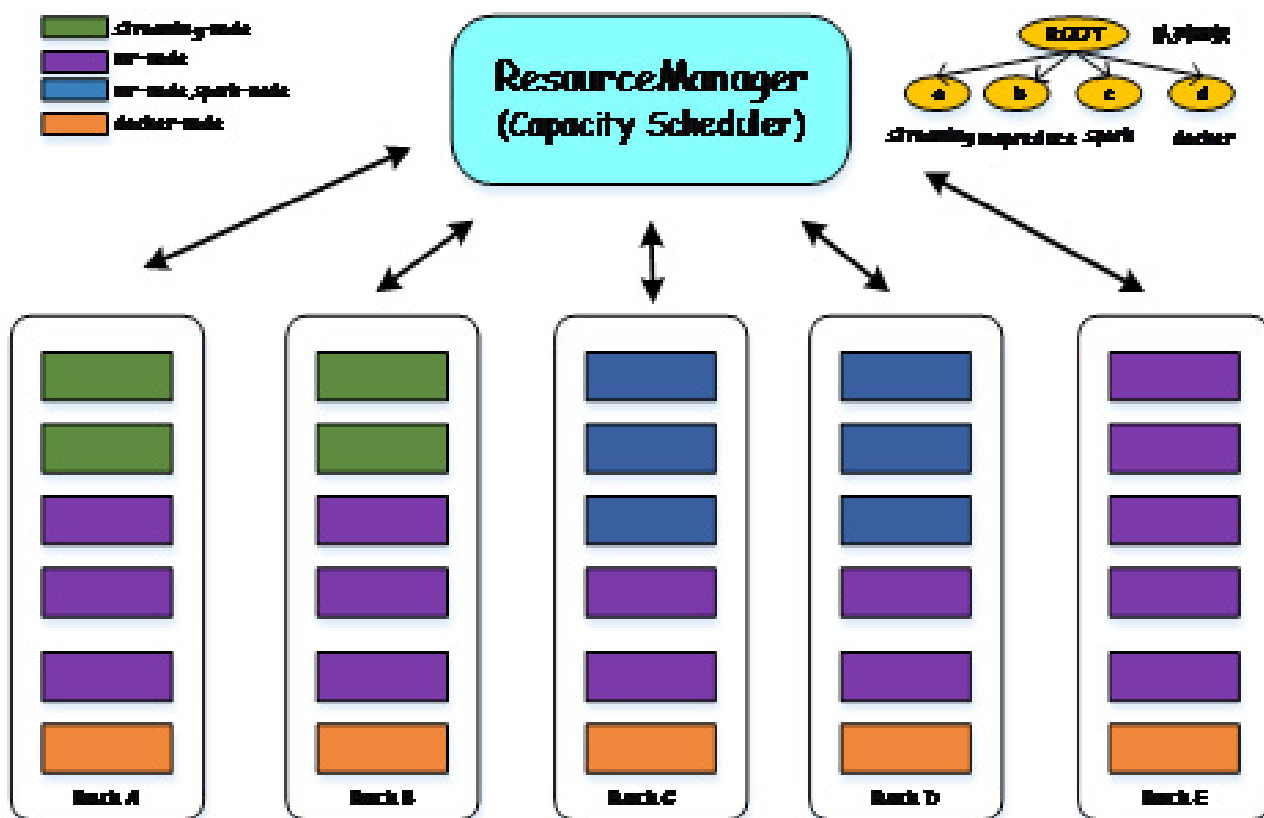


图 3 YARN 部署示例

如图 3 所示，我们根据机器配置和应用程序需求，为集群中的节点打上了多种标签，包括：

- `spark-node`：用于运行 spark 作业的机器，这些机器通常配置较高，尤其是内存较大；
- `mr-node`：运行 MapReduce 作业的机器，这些机器配置是多样的；
- `docker-node`：运行 docker 应用程序的机器，这些机器上装有 docker engine；
- `streaming-node`：运行 spark streaming 流式应用的机器。

需要注意的是，YARN 允许一个节点同时存在多个标签，进而实现一台机器混合运行多类应用程序（在 hulu 内部，我们允许一些节点是共享的，同时可以运行多种应用程序）。表面上看来，通过引入标签将集群分成了多个物理集群，但实际上，这些物理集群跟传统意义上完全隔离的集群是不同的，这些集群既相互独立又相互关联，用户可非常容易地通过修改标签动态调整某个节点的用途。

Hadoop YARN 应用案例及经验总结

Hadoop YARN 应用案例

Hadoop YARN 作为一个数据操作系统，提供了丰富的 API 供用户开发应用程序。Hulu 在 YARN 应用程序设计方面进行了大量探索和实践，开发了多个可直接运行在 YARN

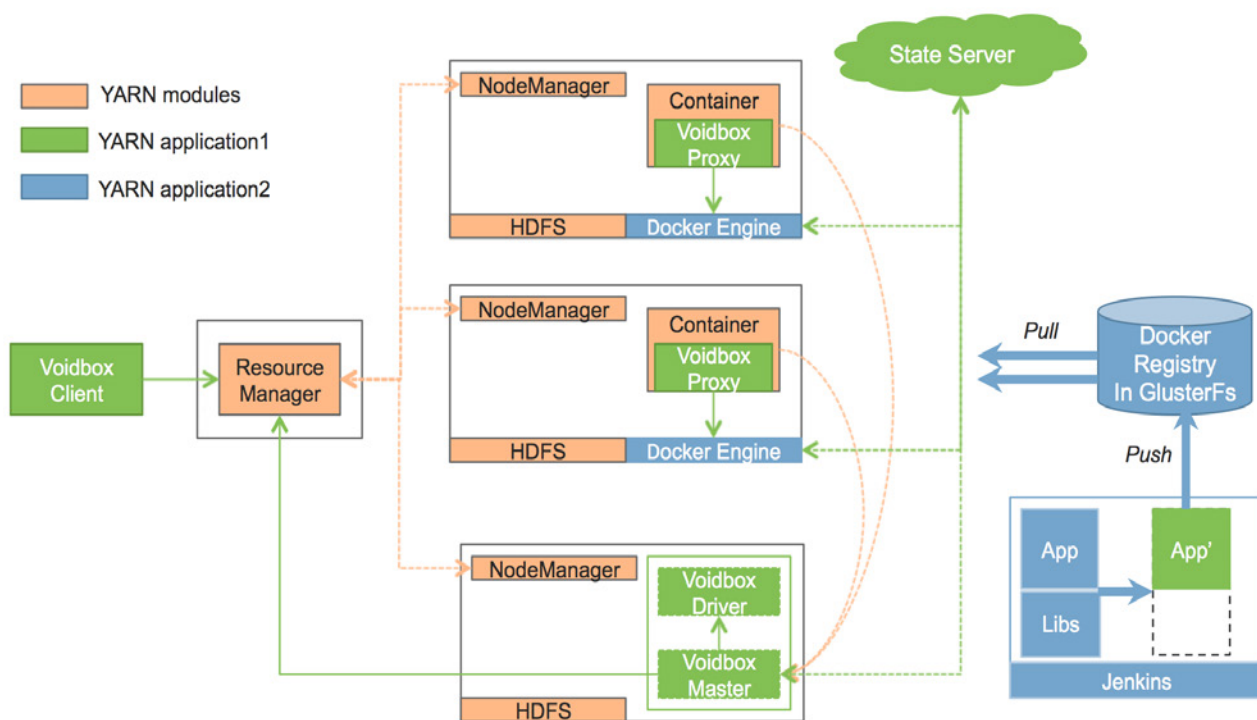


图 4 Voidbox 系统架构

上的分布式计算框架和计算引擎，典型的代表是 voidbox 和 nesto。

(1) 基于 Docker 的容器计算框架 voidbox

Docker 是近两年非常流行的容器虚拟化技术，可以自动化打包部署绝大部分应用，它使得任何程序能够运行在资源隔离的容器环境，从而提供了一套更加优雅的项目构建、发布、运行的解决方案。

为了整合 YARN 和 Docker 各自的独特优势，Hulu 北京大数据团队开发了 Voidbox。Voidbox 是一个分布式的计算框架，利用 YARN 作为资源管理模块，用 Docker 作为执行任务的引擎，从而让 YARN 既可以调度传统的 MapReduce 和 Spark 等类型的应用程序，也可以调度封装在 Docker 镜像中的应用程序。

Voidbox 支持基于 Docker Container 的 DAG（有向无环图）任务和长服务（比如 web service），提供命令行方式与 IDE 方式等多种应用程序提交方式，满足了生产环境和开发环境的需求。此外，Voidbox 可以配合 Jenkins，GitLab，私有的 Docker 仓库完成一整套开发、测试、自动发布的流程。

在 Voidbox 中，YARN 负责集群的资源调度，Docker 作为一个执行引擎，从 Docker Registry 中拉取镜像执行。Voidbox 负责为基于容器的 DAG 任务申请资源，运行 Docker 任务。如图 4 所示，每个黑线框代表一台机器，上面运行着几个模块，具体如下：

- Voidbox 组件：
 - VoidboxClient：客户端程序。用户可通过该组件管理 Voidbox 应用程序（Voidbox 应用程序包含一个或多个 Docker 作业，一个作业包含一个或多个

Docker 任务)，比如提交和杀死 Voidbox 应用程序等。

- o VoidboxMaster: 实际上是一个 YARN 的 Application Master, 负责向 YARN 申请资源, 并将得到的资源进一步分配给内部的 Docker 任务。
- o VoidboxDriver: 负责单个 Voidbox 应用程序的任务调度。Voidbox 支持基于 Docker Container 的 DAG 任务调度并且在任务之间可以插入其他用户代码, Voidbox Driver 负责处理 DAG 任务之间的依赖顺序调度以及运行用户代码。
- o VoidboxProxy: 是 YARN 与 Docker 引擎之间的桥梁, 负责中转 YARN 发向 Docker 引擎的命令, 比如启动或杀死 Docker 容器等。
- o StateServer: 维护各个 Docker 引擎的健康状况信息, 向 Voidbox Master 提供可运行 Docker Container 的机器列表, 使得 Voidbox Master 可以更有效地申请资源。
- Docker 组件:
 - o DockerRegistry: 存储 Docker 镜像, 作为内部 Docker 镜像的版本管理工具。
 - o DockerEngine: Docker Container 执行的引擎, 从 Docker Registry 获取相应的 Docker 镜像, 执行 Docker 相关命令。
 - o Jenkins: 配合 GitLab 进行应用程序的版本管理, 当应用版本更新时, Jenkins 负责编译打包, 生成 Docker 镜像, 上传至 Docker Registry, 从而完成应用程序自动发布的流程。

类似于 spark on yarn, Voidbox 也提供两种应用程序运行模式, 分别是 yarn-cluster 模式和 yarn-client 模式。yarn-cluster 模式中应用程序的控制组件和资源管理组件都运行在集群中, Voidbox 应用程序提交成功后, 客户端可以随时退出而不影响集群中应用程序的运行。yarn-cluster 模式适合生产环境提交应用程序; yarn-client 模式中应用程序的控制组件运行在客户端, 其他组件运行在集群中, 客户端可以看到关于应用程序运行状态的更多信息, 客户端退出后, 在集群中运行的应用程序也随即退出, yarn-client 模式可以方便用户进行调试。

(2) 并行计算引擎 nesto

nesto 是 hulu 内部一个类似于 presto/impala 的 MPP 计算引擎, 它是专门为处理复杂的嵌套式数据而设计的, 支持复杂的数据处理逻辑(SQL 难以表达), 其采用了列式存储、code generation 等优化技术以加速数据处理效率。Nesto 架构类似于 presto/impala, 它是无中心化的, 多个 nesto server 通过 zookeeper 进行服务发现。

为了简化 nesto 部署和管理成本, hulu 直接将 nesto 部署到 YARN 上。这样, nesto 安装部署过程将变得非常简单: Nesto 安装程序(包括配置文件和 jar 包)被打成一个独立的压缩包存放到 HDFS, 用户可通过运行一个提交命令, 并指定启动的 nesto server

数目、每个 server 需要的资源等信息，即可快速部署一套 nesto 集群。

Nesto on yarn 程序由一个 ApplicationMaster 和多个 Executor 构成，其中 ApplicationMaster 负责像 YARN 申请资源，并启动 Executor，而 Executor 的作用是启动 nesto server，关键设计点在 ApplicationMaster，它的功能包括：

与 ResourceManager 通信，申请资源，这些资源需保证来自不同的结点，以达到每个节点只启动一个 Executor 的目的；

与 NodeManager 通信，启动 Executor，并监控这些 Executor 健康状况，一旦发现某个 Executor 出现故障，则重新在其他节点上启动一个新的 Executor；

提供一个嵌入式 web server，以便展示各个 nesto server 中任务运行状况。

Hadoop YARN 开发经验总结

（1）巧用资源申请 API

Hadoop YARN 提供了较为丰富的资源表达语义，用户可以申请特定节点 / 机架上的资源，也可以通过黑名单的方式不再接受某个节点上的资源。

（2）注意 memory overhead

一个 container 的内存是由 java heap, jvm overhead 和 non-java memory 三部分构成的，如果用户为应用程序设置的内存大小为 X GB (-xmxXg)，则 ApplicationMaster 为其申请的 container 内存大小应为 X+D，其中 D 为 jvm overhead，否则可能会因总内存超出限制被 YARN 杀死。

（3）log rotation

对于长服务而言，服务日志会越积攒越多，因而 log rotation 显得尤为重要。由于启动之前，应用程序是无法知道日志具体存放位置（比如哪个节点的哪个目录下）的，为了方便用户操作日志目录，YARN 提供了宏 <LOG_DIR>，当该宏出现在启动命令中时，YARN 会自动将其替换为具体的日志目录，比如：

```
echo $log4jcontent > $PWD/log4j.properties && java -Dlog4j.configuration=log4j.properties ...
```

```
com.example.NestoServer 1>><LOG_DIR>/server.log 2>><LOG_DIR>/server.log
```

其中变量 log4jcontent 内容如下：

（4）调试技巧

```
log4j.rootLogger=INFO,console  
  
log4j.appender.console=org.apache.log4j.RollingFileAppender  
log4j.appender.console.layout=org.apache.log4j.PatternLayout  
log4j.appender.console.target=System.err  
log4j.appender.console.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %5p %c:%L - %m%n  
log4j.appender.console.File=<LOG_DIR>/server.log  
log4j.appender.console.MaxFileSize=200MB  
log4j.appender.console.MaxBackupIndex=10
```



```
vargs.add("cat $PWD/launch_container.sh > /tmp/launch_container.sh && ");
vargs.add("ls -la $PWD/ > /tmp/container_pwd.sh && ");
vargs.add(ApplicationConstants.Environment.JAVA_HOME.$$() + "/bin/java");
// Set am memory size
vargs.add("-Xms" + containerMemory + "m");
vargs.add("-Xmx" + containerMemory + "m");
vargs.add("-Djava.io.tmpdir=$PWD/tmp");
vargs.add("-Dlog4j.configuration=" + NestoYarnConstants.NESTO_YARN_APPCONTAINER_LOG4J);
```

NodeManager 启动 Container 之前，会将该 Container 相关的环境变量、启动命令等信息写入一个 shell 脚本，并通过启动该脚本的方式启动 Container。有些情况下，Container 启动失败可能是由于启动命令写错的缘故（比如某些特殊字符被转义了），为此，可通过查看最后执行脚本内容判断启动命令是否存在问题，具体方法是，在 container 执行命令之前添加打印脚本内容的命令。

（5）共享集群带来的性能问题

当在 YARN 集群中同时运行多种应用程序时，可能造成节点负载不一，进而导致某些节点上的任务运行速度慢于其他节点，这对于 OLAP 需求的应用是不能接受的。为了解决该问题，通常有两种解决方式：1）通过打标签的方式将这类应用运行到一些独享的节点上 2）在应用程序内部实现类似于 MapReduce 和 Spark 的推测执行机制，为慢任务额外启动一个或多个同样的任务，以空间换时间的方式，避免慢任务拖慢整个应用程序的运行效率。

Hadoop YARN 发展趋势

对于 YARN，会朝着通用资源管理和调度方向发展，而不仅仅限于大数据处理领域，包括对 MapReduce、Spark 短作业的支持，以及对 Web Service 等长服务的支持。

董西成，就职于 Hulu，专注于分布式计算和资源管理系统等相关技术。《Hadoop 技术内幕：深入解析 MapReduce 架构设计与实现原理》和《Hadoop 技术内幕：深入解析 YARN 架构设计与实现原理》作者，dongxicheng.org 博主。

专访王峰：Hadoop 生态下一代计算引擎 streaming 和 batch 的统一

作者 姚梦龙

InfoQ：您是 2009 年开始关注 Hadoop 生态技术发展，并逐步将其引入阿里电商搜索技术体系。那时的 Hadoop 生态圈是怎样的？可否介绍下 Hadoop 在阿里的历史？

王峰：对于 Hadoop，我个人很早就了解了。Hadoop 06 年出来，我们 07 在雅虎中国见到用 Hadoop 做 search，搜索引擎是大数据的第一个应用场景。当时和雅虎美国合作看过 Hadoop 应用，那时还是 0.1x 版本，07 年也见到了 HBase 被首次尝试用于 yahoo vertical search 中。在 08 年阿里第一个 Hadoop 项目——云梯，就在我们搜索技术中心诞生的。09、10 年我重新到淘宝搜索后台开始建立 Hadoop，算是正式将 Hadoop 用于生产系统，以前是直接做离线数据分析、BI、统计，不支持在线业务。10 年我们将整个阿里巴巴的搜索、后台数据、商品更新这些数据用 Hadoop 从 MySQL 同步过来，做处理，建索引，更新到线上的搜索引擎，供买家搜索商品，做跟交易相关的线上系统的连接。那时也开始做 HBase，刚起步，做阿里集团以及全网商品的存储。10 年时集群才 100、200 台，而现在个性化搜索、推荐这种真正引导电商成交的关键链路都是在 Hadoop 上做，现在应该有几万台规模做实时交易的数据处理，这已经不是大数据场景的数据挖掘、数据训练，这是 online 或者说 nearline。我们是服务于在线的数据分析，比方说一个卖家更新一个商品，我们可以秒级同步到线上让买家看到。这听起来简单，但像淘宝这样的电商，这是要经过好几百个 feature 的计算，很多数据处理的环节，这个过程是很长的。而我们甚至在双 11 的场景，也能做到秒级。

InfoQ：对于 HBase 而言，Java GC 和读写性能是它的两大问题，这方面你们做了哪些优化吗？

王峰：HBase 我们是长期投入，现在我们有一些成员专注跟 HBase。GC 方面也是比较大的投入，在高并发读写、大数据量情况下，比如双 11，就有 2 千万 QPS，那 GC 就成了大问题。但又不好调，Java 的 GC 又不能代码级改动。现在主流是用 CMS GC，相对来说还是最稳定可调的，我们也针对我们应用场景做了定制调法，因为 GC 调法没有规范，

只能说适合你的系统。比如你要考虑访问 IO 特性，来的 request 大小，新生代旧生代规律，像我们就是除了 GC log，还配合 visualvm 等工具直接去看 GC 情况，尝试各种参数组合去调，然后找到相对最优的，其实是磨出来的，经过实践找到的。

而关于读写性能，HBase 用 HDFS 做文件系统，而 HDFS 是高吞吐而不是低延迟，所以本身不是随机访问能力很强的。而且我们情况更为困难，为了提高效率，我们 HDFS，YARN 和 HBase 是混布的，HBase 和 HDFS 做存储，YARN 做计算调度，资源是共享的。对此我们的解决方案是利用 HDFS 的新特性：支持内存、SSD、HDD 混合存储。不过混合存储不很稳定，不很成熟，我们做了自己的优化，改进了 one SSD 策略。就是 HDFS 有三个冗余备份，我们用一块冗余备份做 SSD，同时我们在混合架构下也做了自己策略的优化，就是有些机器有 SSD，有些没有，因为集群的机器不可能都是一样的。我们做了特制的优化，做到访问热点数据或关键表数据都是 SSD，随机读在 SSD，顺序读在普通的机械硬盘做。这样随机读、顺序读、HDFS 和 HBase 在 IO 的隔离，就天然地实现了。这样后，在高并发时，SSD 的高 QPS 的特性可保证 HBase 的 latency，而且成本低很多，因为只是把重要的数据的一份放在 HDFS 上。

InfoQ：有人说，目前 YARN 和 HDFS 是同级别的模块，而以后 HDFS 会成为 YARN 的一个组件，也由 YARN 统一调度。

王峰：我觉得理论上是可以的，但这里有一个相互依赖的问题，你要用 YARN，就要有一些 Storage 的支持，现在是 HDFS 作支持，这是循环依赖。比如我提交了我的代码，部署一个 application 到 YARN，比如 HDFS 作为第一个 application，那它存在哪里，要把包传上来，所以至少要有个存储。或者说你为它单独定制一个存储，但我个人认为这个意义不大，比如很多人谈 HBase on YARN，Kafka on YARN，我觉得运维或临时搭集群才有这个需求，生产系统没有。因为 HDFS，HBase，Kafka 都用到本地磁盘，不可能临时搭一个就换了，因为 on YARN 的东西就是可以随时漂移嘛，但我搭了 HDFS，肯定是每台机器一个嘛，不能是有些机器有，有些机器没有，而且过两天还要换？！数据是不适合来回移动的，否则成本会很高。你有这个需求那就类似搭虚拟机一样，我先起来一个 instance，过两天不用了就关掉，过两天再起起来，这个适合临时集群，对稳定的生产集群意义不大。

InfoQ：YARN 增加了对 Docker 的支持，您觉得 Docker 对 YARN 意义大吗？

王峰：Docker on YARN 也是一种尝试嘛，这并不冲突，YARN 作为调度管理。Docker 可作为容器部分和执行器，这个趋势更像一个轻量级虚拟化的资源管理方式，我个人觉得这是有用武之地的，特别是做一些轻量级的 application 的资源复用。做一些简单的 web 服务啊，python 程序，如果都做一个虚拟机，隔离太重了，毕竟有开销。而这么做就好很多。YARN 专注于调度，将隔离交给 Docker，是不错的解决方案，阿里内部就有很

多这样的思路，已经实现了。

InfoQ: Yarn 会朝着通用资源管理和调度方向发展吧？包括对 MapReduce、Spark 短作业的支持，以及对 Web Service 等长服务的支持

王峰: 恩。我觉得这是 Hadoop 社区最大的成长空间，一开始 1.0 是 HDFS +MapReduce，2.0 后是 HDFS +YARN。HDFS 作为基于大文件的分布式文件系统基本比较通用了，没有必要找替代物了，但 YARN 还有类似的系统，如 Mesos。不过它与 YARN 理念不完全一样，也有不少场景在用 Mesos。

刚刚说，作为 Hadoop 生态系统最大的生长空间其实在于 YARN，因为 HDFS 比较稳定，文件系统的新功能推出也相对慢一些，重点还是在性能改进。反而是 YARN 可以让更多的生态进来，比如 Spark，Flink 这些东西都可以 on YARN，因为你在上面就可以和大家共享计算资源，所以 YARN 更像一个大的 Hadoop 生态的容器，希望把更多的新的技术包进来。这个做的好的话，我觉得它就类似 linux 成为大数据的 os，不管什么好的东西出来，你都可以运行在这个平台上。到那时 Hadoop 就只是一个代名词，那就意味着一种大数据的开源生态啦。现在 Hadoop 已经是一个生态，但现在它和 Spark 是非常微妙的，Spark 也是一个生态，不过 YARN 的生态和 Spark 的生态不是一个概念。YARN 说 Spark 可以跑在我上面，Spark 说我可以不跑在你上面，这是一个矩阵交叉的感觉。但任何计算模型都是有生命周期的。所以我觉得 Hadoop 做的聪明的一点就是，把 YARN 这种 os 的概念放进来了，我不管你多好，我都可以把你放在我上面。你基于我来做，我不断沉淀我的 web os 系统，以后的模型都可以跑在我上面。所以它放掉了 MapReduce，MapReduce 其实是配角了，我觉得它慢慢就会荒废掉。但如果以后的计算模型可以和 YARN 结合好，那 Hadoop 生态社区就非常丰富，百花齐放。不同的新的好的东西都 share 资源，跑在一起，我个人觉得这是比较健康的发展方向

InfoQ: Spark vs Hadoop？它们是竞争关系？

王峰: 我觉得现在来看，竞争关系也有一部分。就跟我们做产品一样，这就像微信和 ios 的关系，Spark 像微信，YARN 像 ios，当你的计算模型强到一定程度时，就希望把底下的东西盖住。就比如微信强到一定程度时，你就不需要用 ios 的东西，用我微信的东西就足够了，微信里什么东西都很全面。Spark 就这样，它很强势，它有自己的管理系统，其实你不用考虑在 YARN 上起别的东西，我这已经是个生态了，要跑批处理、跑流式、跑机器学习，它很全面，它都有。这就相当于说我不是直接说和 Hadoop 竞争，不是对等关系，但从用户角度来说，你的需求在 Spark 这层已经解决了，你可以忘记 YARN 那个东西，是这么一个竞争关系。但也可以是一个竞合关系，就是 Spark 可以把他的资源管理和 YARN 合作，你不能假设你解决了一切的问题，这是用户的选择，你不可能想到一切，而如果 YARN 上其他方案能解决，那你 run on YARN，这就是合作关系。

InfoQ: 可否介绍下 Hadoop 生态目前最前沿的技术和概念?

王峰: 分三个方向，一是调度，以 YARN 为主，这是一个比较核心的能力，往在线高可用方向发展。Hadoop 以前是离线，以后应该走向 online。以前离线调度作业，失败就失败了，而走向 online 调度就很有挑战。现在 YARN 也在努力，比如说没有单点，但离 Borg 还有差距。

二是存储，HBase ,HDFS 是老牌的，新的存储出现了，druid, pinot 这种新的列存储，真正的 column 加上 index 的这种列存储，尤其是做数据分析的列存储技术都是相对新的，不再是老的那些概念了。也有一些新的方向，tachyon 这种分布式内存的，支持 Spark 的分布式内存存储系统，这都是一些新的思路，而且以后会有很好的业务场景的，因为这都是需求逼出来的系统，有很大的应用场景。

最火的还是计算，MapReduce 基本就更新掉，但不会马上离开，它的稳定性、吞吐量还是最好的，因为它积累了太多年，在一定场景内可以存在。但我觉得就是落后产业了。新的像实时计算，Storm，不过 Storm 本身不是很先进的系统，可以说是上时代的产物了，Twitter 的 heron 以及阿里的 jstorm 还在发展 Storm 体系，但我觉得这也不是体量大的系统。最大的还是 Spark，Spark 是现在最辉煌的系统。我们还关注 Flink 这个系统，这是一个新的计算引擎。从理论来看，它比 Spark 更没有边界，更先进。Spark 很火，但是它毕竟是一个 (batch) 系统，做离线的，批处理的是它的先天优势，可能做一些内存迭代、机器学习或者替代 MapReduce 做算子层更丰富的批处理。但它本质是一个 batch，它最大的问题就是 Spark stream，转换成一个离散的流，可以认为是把 batch 切小，每个 batch 可以认为是无限小，比如网络上一秒的数据形成一个 RDD 的 batch，再来一个，再提一个，不停地提 job。但是这个 batch，就相当于你再切，他还是一个 batch，这是一个理论问题，不是你技术牛不牛的问题，就相当于给你一个木棍子，你切切切，你再切，切再薄的片，它还是有厚度的。但 Flink 是反的，它认为 stream 是一切，batch 是 special case，batch 只是一个有明确 start 和 end 的 stream，它是德国那个论文发的，Flink 的模型在我们看来是更先进的，它自己号称是第四代，它是说 MapReduce 第一代，Storm 第二代，Spark 第三代，它第四代。Flink 还没发展起来，没大厂验证，但它基因更好，它拿流可以模拟 batch，这是完全没有约束的，就像我可以拿原子组成物体。而这是不可逆的，就像你不能把一个东西碎成原子，你只是一直切切切，但是原子可以组成任何东西。从这里来说，Flink 是 unlimited，它没有明显的技术限制。现实的例子是，我们想把数据做到毫秒级更新超大规模场景，Spark 是做不到的，根据他现在的理论架构无论如何做不到，RDD 是如何也做不到的。一个 batch，你怎么弄也做不到一毫秒提一个 job，这是不可能的。batch 粒度有点粗，你要能做到，你要做无限优化，还不敢保证成功。而 Flink 若做无限优化，就可以 stream 转 batch，所有你有的东西我都有。这其

实是很恐怖的，特别在一个互联网领域，一个小东西如果基因足够好，有足够的成长空间，它可以长到无限大。而再大的东西遇到瓶颈的话，也会遇到危险。反正现在能看到的就是，Spark 能稳定的往前走，Flink 就看能不能冲起来。

InfoQ: 最后一个问题，现有的 Hadoop 集群，版本是如何管理的？

王峰: 我们团队是使用社区的版本，做一些自己的改进，并且会把改进提交到社区。因为我觉得很私有地维护一个版本是有问题的，因为社区发展很快，你私有一个版本，短期你觉得占便宜了，把很多新的东西弄进来，自己加了一些东西社区都没有。但一两年以后呢？这是长跑，你可以冲刺一下，但不能年年这样。你的团队会有变化，人员会有更改，会有业务上的紧张，资源进度上的变化。长期跑赢社区几乎是不可能的。我们选择的就专门几个人跟这个方向，如果我们觉得这是一个好的抽象好的 feature，我们就会提交给社区。我们也在培养自己的 committer。我们不私有化这个东西，但肯定有一些非常定制的，这是不可避免的。比如社区就是不认这个东西，但我们业务的确有这个需求。我们会有一个阿里内部的发行版，用作集群部署的版本管理。这个版本会结合 Cloudera、社区版的 patch，把有些东西拿过来，和我们自己的 patch 合进来，但是和社区是兼容的，比如社区升级，我们也会升过来，有些 patch 已经被合进社区版本了，没合进去的我们再合一次。但不会保留太多私有版本。

王峰，淘宝花名莫问，2006 年毕业后即加入阿里巴巴集团，长期从事搜索和大数据基础技术研发工作，目前负责阿里搜索事业部离线基础平台团队。自 2010 年开始将 hadoop 生态技术正式引入阿里搜索技术体系，带领团队打造出的数据基础设施，每日可实时处理阿里集团内的海量商品和用户行为数据更新，直接影响并提升搜索和推荐引导成交，承担了阿里电商数据流程的核心职责。我们团队管理着目前阿里最大的 Hadoop 集群，上面运行的搜索以及推荐业务也是阿里集团最核心的电商场景。我们希望站在 Hadoop 生态圈的肩膀上，基于自身业务场景优势，在实践中不断产生新技术并回馈给社区，与开源技术发展形成良性循环和促进，欢迎各位 Hadoop 生态圈内的有志之士加入我们（微博：淘莫问），一起打造世界一流的数据基础设施团队。



[大数据技术的回顾与展望](#)

编辑评语：Hadoop 老矣，尚能饭否？



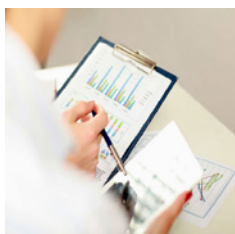
[卢亿雷：十年之痒，Hadoop 生态的最新发展是怎样的？](#)

编辑评语：老司机告诉你：Hadoop 代码越来越大，学起来成本更大，怎样才能更优雅的掌握？



[达观数据分析平台架构和 Hive 实践](#)

编辑评语： 本文将从 Hive 原理、数据分析平台架构、数据分析实战、Hive 优化等四个方面来分享 Hive 的心得和实战经验。



[如何建立完整可用的安全大数据平台](#)

编辑评语： 在现有的 Hadoop 和大数据生态圈内根据实际需求挑选并整合各部分合适的组件来构建能够支撑多种查询和分析功能的系统平台。



[运营、报表、分析三位一体化，什么样的 SQL 引擎能经得住挑战？](#)

编辑评语： Hadoop 生态圈所代表的合作开发，带来了大数据产业的繁荣。那么，这会带来哪些技术趋势呢？



[Hadoop 之父给黄色小象的十岁生日祝词](#)

编辑评语： Hadoop 十岁生日之际，开源大牛 Hadoop 之父 Doug Cutting 会对这只小象说些什么呢？

大数据开放平台搭建，难点何在

作者 孙利兵

大数据发展趋势

大概在 08 年左右，Hadoop 从 Nuch 里的一个 package 开始的独立出来，不断的被大家所关注，它本身不断的进化，包括对压缩算法的 Native 实现，Checksum 机制的优化，还有 ShortCircuit Read（支持直接文件的内置的短读），这算是读取性能的优化。在这些不断的优化下，Hadoop 逐渐变得健壮。Hadoop 的 2.3 版本，是一个架构上的根本变化，把资源管理提到了一个很高的高度，就是我们新一代的 Yarn&Mapreduce2.0。还有一个 HDFS 上的架构优化，就是 NameNode Federation。NameNode 分布式管理，可以极大的增进机群的可扩展性。

Hadoop 本身不断完善，围绕着 Hadoop 的生态系统也在不断的完善，像 Oozie，就是与 Hadoop 结合非常紧密的一个工作流引擎；Flume 是围绕 Hadoop 的日志收集类的 ETL 工具；Hive 是一个 SQL On Hadoop 的实现，也是最早的一个实现，现在已经有很多了；Pig 类似 Hive，但是是一个内置脚本、有自己独立语法，相对来说是一个优化版的 Hive 实现。但是它的语法，相对来说要比 Hive 性能高一些，在 Facebook 内部使用比较多。另外 Cloudera 推出了 SQL On Hadoop 的 Impala 的实现。另外还有 Spark 也是新兴的一个内存计算模型的技术。

大数据技术应用，困难何在

围绕着 Hadoop 的周边的生态系统在不断的完善，运维管理工具上也得到了极大的进展。因为在早期，大家部署一个 Hadoop 系统会非常麻烦；从 08 年到现在，我经历过大大小小的系统，自建的，内建的，包括公司各种平台项目里面建的，不计其数，特别



的辛苦。包括脚本，权限设置，一些目录权限的设置，安装一个系统大概要半天左右到一天左右时间。非常熟练之后也要半天左右时间。现在像 Apache Ambari, Cloudera Manager，已经把整个机群的部署变得非常简单，基本上几分钟就能把我们需要花几十分钟的机群部署起来，甚至是一个几十台机器规模的机群。

大数据技术在不断的发展，但是它还是有些天生的不足。首先是技术本身在百花齐放，生态系统里面的技术在不断的完善，包括 Hadoop, Hive, Spark 等，那么就会存在选择上的困难——如何应用好每一项技术，就是一个难题。另外，大数据技术本身内部的融合性也是不太够的。现在有一种趋势，每一个开源工具都在强调自己的性能有多好，都想围绕着自己去建立生态系统。另外就是怎么合理的使用这些技术。比如说我们以前的系统是围绕着一项技术建立的，然后又引入了一个新的技术，两个技术怎么实现融合，这就是一个很大的难题。

另外一个就是大数据技术与其他传统技术的融合性不够。传统技术，比如以前使用的数据库，一些其他的 service，Solr 检索服务什么的，也没有成熟的融合方案。在实践中会比较陌生，没有一套成熟的体系支撑。你可能需要去翻很多的文档，自己做很多开发，才能实现这些功能。那么我们现在缺少什么呢？缺少一个能融合现有大数据技术的技术，这个真的是非常非常关键的。

技术领域是怎么来面对这个问题的

大概在 13 年，Doug Cutting 做过一次技术分享，The State of the Apache，这个文档大家可以在百度文库上可以检索到。他当时提出了一个概念叫 Apache Hadoop

大数据基础技术的风向标

- ④ The Ecosystem is the System
 - ④ Hadoop has become the kernel
 - ④ of the distributed operating system for Big Data
 - ④ a de-facto industry standard
- ④ No one uses the kernel alone
- ④ A collection of projects at Apache
- ④ Avro support across components



以Hadoop为核心，融合其他技术的平台系统
Avro是实现融合的关键技术

Ecosystem。Ecosystem，E 应该是代表 Easy，CO 是 Cross，System 系统平台。这在他的论文里面摘录的几句关键的话。

文档本身的介绍非常简单，寥寥几页，但很经典。我觉得他说得非常在点子上。我的解读，这个总体来说应该是这样一个思想：以 Hadoop 为核心，融合其他技术的平台级系统，Avro 将是实现融合技术的关键技术。

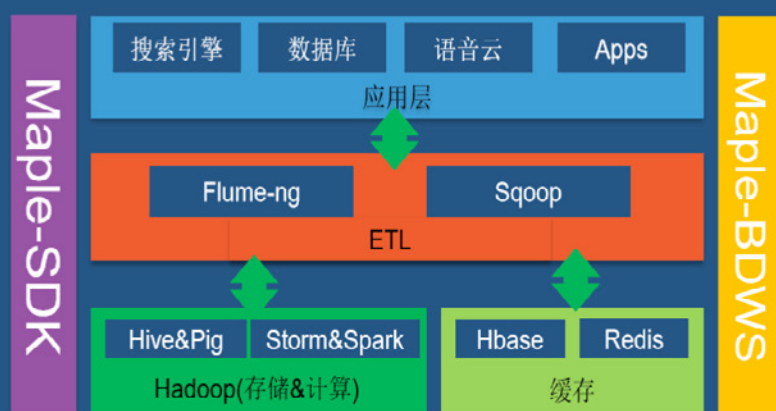
在行业内，Cloudera 应该是 Hadoop 就是围绕着大数据 Big Data 这种解决方案的一个国外很有影响力的一个公司。Cloudera 是在做 Hadoop 的应用体现，想让 Hadoop 越来越容易被大家所应用。他提供了两个解决方案。首先，建立了 Hue 这套系统，提出 Use Hadoop with Hue。另外就是 Administer Hadoop，就是运维管理 Hadoop。建立了叫 Cloudera Manager 这一整套的工具。像 Hue 和 Administer 已经是说解决了两方面的问题，Hue 用来调度管理任务。Administer 是管理和运维平台。

这是 Cloudera 给出的答案，在科大讯飞的实际开发过程中，我们是怎么应付和解决这些难题的呢？我们围绕着 Doug Cutting 提出的 ECoSystem 的思想上，我们也开始逐渐建立自己的大数据开放平台 Maple。

我们的实现以数据导向为理念。数据导向为理念是一个思想。以前我们在做一件事情的时候都会考虑，做这件事情要使用什么样的技术。因为首先大数据是围绕着数据去做的，很多时候会偏离数据，而去考虑很多技术的细节问题。但是在做实际业务开发的过程中，我们需要围绕着数据去想问题，而不是围绕着技术去想问题，这就是需要数据导向为理念。我们所有的业务开发围绕着数据，数据是什么样的，我们就怎么处理。整个系统平台是以 Hadoop 为核心，这也是符合 Doug Cutting 提出的 EcoSystem 的思想。

最后我们提出了以 EcoSystem 设计理念，以 Hadoop 为核心，融合优秀的技术，因地

讯飞大数据开放平台-架构图



制宜的使用技术。综合来看，每个技术都有它的特长，因地制宜的使用技术，才能让这个技术得到更好的发挥。我们还需提升大数据的应用体验。如果你是早期接触的话，在它上面开发任务，提交任务，整个的流程管理是相当复杂的。

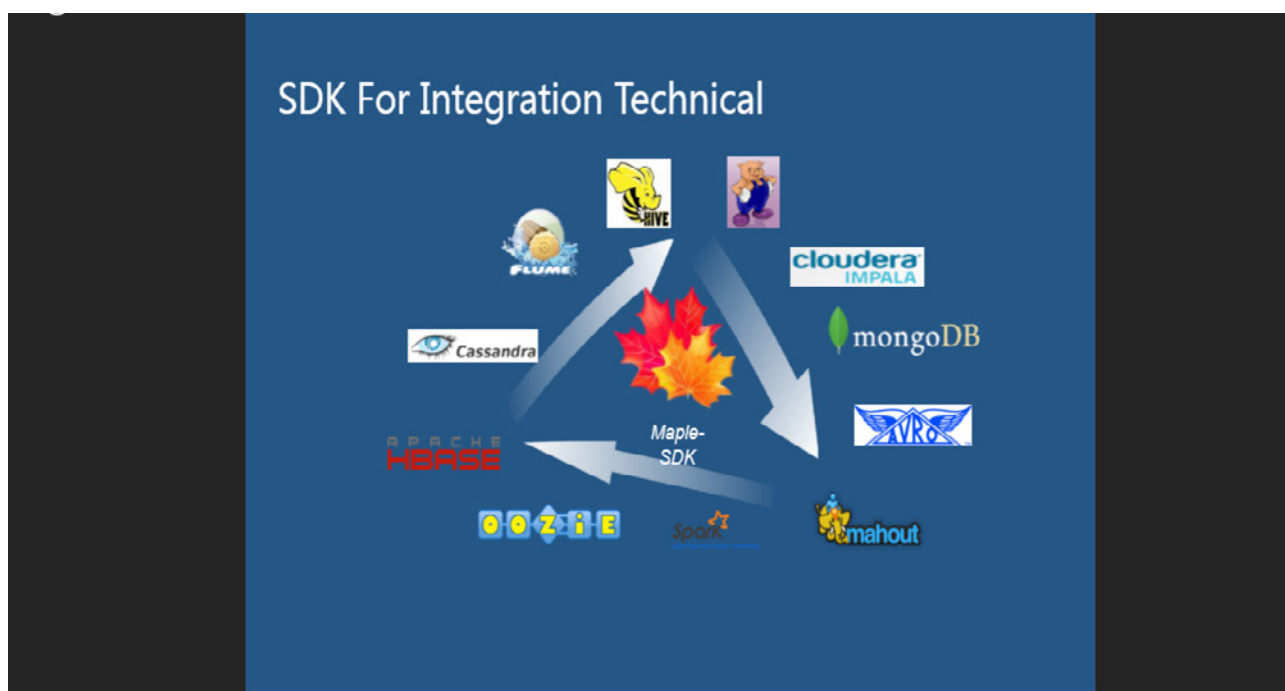
科大讯飞的大数据平台

我们大数据开放平台分成三部分。1. 基础机群，围绕着 Cloudera 发行版本 CDH 来构建的。2. 我们上层构建了自己的 Maple SDK，是面向开发者提供的开发包。3. 是 Maple BDWS。

大数据平台的整个的架构。从上层的应用层到 ETL 层，到我计算和存储层，这是整个的数据流。以这个上面的这些设计为基础开展大数据开放平台的工作。非常的值得去借鉴是我们架构上不仅定义了数据流向，也定义了开发的过程，Maple BDWS 应该是我们整个大数据开放平台的一个门户，解决代码托管，编译部署， workflow 设计，任务调度，数据和任务信息浏览。特性：支持多集群管理，支持多版本 Hadoop，支持多项目管理，在线编译部署（one button to use）。整个平台是用 Python 去做的，支持了 Python 扩展。可以在线的测试和运行 Python 的代码。

Maple BDWS 是我们整个大数据开放平台的一个门户，Maple SDK 就是我们整个大数据开放平台的灵魂。

在设计 SDK 的时候，我们的目标是为了实现 Integration Technical，就是融合技术，希望能把各种技术都提供一种标准的开发方式，开发模板。通过在实践中应用成熟以后，把开发模式，融合的编码规范分享出去。围绕着 SDK，我们融合了 Hbase，oozie，



Flume、Avro 这样的技术。SDK 里面包含一套数据建模的功能，基于 Avro 的 Mapreduce 编程库。还有一套 Flume-ng 的扩展组件。统计分析也是一个常见的业务，Maple-Report 是一个统计分析解决方案。另外还含有一个分布式索引的库，叫 Maple-Index。大数据建模系统 Data Source，适用于大数据的动态自动建模系统。

实现技术融合的关键

用大数据的眼光看数据，会跟平常我们看数据会有什么不同呢？用大数据的眼光看数据，会发现数据会分成两种基本属性。一个是 Schema，一个是 Partition。在 Partition 和 Schema 下应该支持多种文件的数据存储格式，包括文本格式，Avro 格式，列存储，数据库文件。

Avro 是融合整个技术的关键，在我们内部大量使用了 Avro 的数据存储格式。我们要围绕着 DataSource 去建立数据导向的 API，提供一个清洗过程的 API。另外提供两个 DataSource 实现 Merge 和 Join 的功能。还围绕 DataSource 实现跟外部数据这种交互。建立了 HiveQL On DataSource 这样的 API，支持 Spark 去 Load 处理，Impala On DataSource，Pig On DataSource 等。

了解 Avro 可以看官网的 Introduction。Avro 经常会被跟 Thrift 和 Protobuf 这两个序列化系统做比较。因为 Avro 本身也是一个序列化系统。那么我们要提出一个问题，在 Thrift 和 Protobuf 已经很成熟的这种基础上，为什么要选择 Avro？在 08 年，10 年左右，我关注这个项目，后来发现所有的代码的提交修改记录，全是 Doug Cutting，里面有 90% 的工作都是 Doug Cutting 本人去做的。Doug Cutting 早期是 Lucene 的项

用大数据的眼光看数据-DataSource



目的创始人，也是 Hadoop 的创始人，一手把 Hadoop 开源项目带起来，甚至都是他亲身去开发的。他花费那么多精力去搞 Avro，必有其独到之处。

Avro 开发中代码生成是可选的，这是一个跟其他系统，就是跟 Thrift 和 Protobuf 有很大区分的一个特性。另外 Avro 支持通用数据读取，不依赖于代码生成。有了这两个特性，Avro 就更能适应大数据变化的特性。Doug Cutting 当时是在 Thrift 和 Protobuf 很成熟的基础上开始着手建立 Avro 的，是非常有想法的。

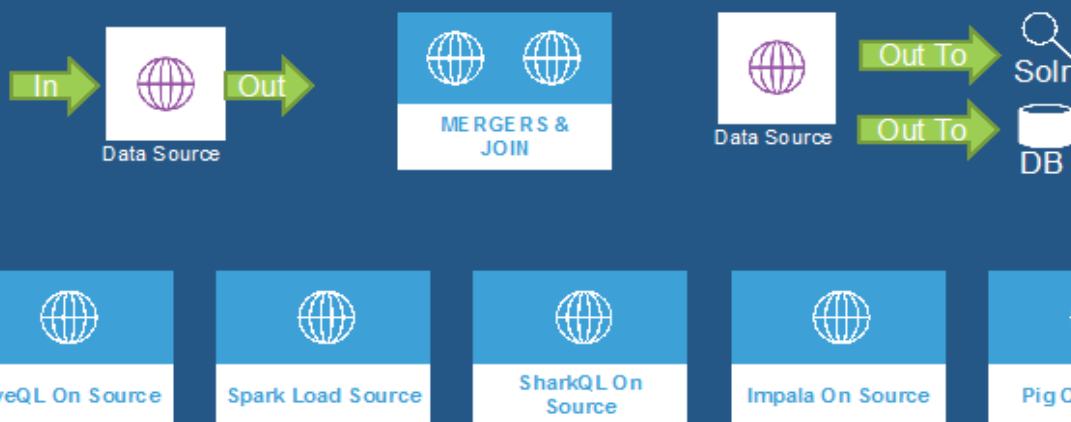
Avro 在讯飞大数据开放平台的应用

首先我们有一套 Avro-Mapreduce 编程框架，围绕 Avro 这种 Mapreduce 开发。Avro 为整个 Mapreduce 过程提供了高性能的数据序列化，是在整个 Mapreduce 生命周期里面一个非常关键的环节，也是非常影响性能的。Avro-Mapreduce 是一个简化的，面向对象的，富于设计的 Mapreduce 编程库。支持 Generic、Specific、Reflect 三种模式。

Avro 还用在整个大数据的数据存储上，这种数据存储是支持通用数据读写，支持多语言读取，内置了很多的压缩算法。因为它内置了很多压缩算法，我们可选，如果适当选择压缩算法，它与传统的文本相比，同样的内容可以节省 10 倍的空间，这个也是非常关键的。它还有更高的读取性能，因为它有内置的压缩，比较精简，它的序列化性能又很高，所以这个数据读取的性能非常高，这是我们现在目前 Avro 在数据存储上的应用。我们 Avro 还用在数据收集的环节。我们在数据收集上也支持多语言的开发，因为前端的应用有很多，包括 PHP，Java，还有 C 等各种语言。

另外，我们以 Flume-ng 融合，实现了结构化的日志收集。Avro 提供了这种对结构

围绕DataSource建立的数据导向API



化数据格式的支持，可以更高效的传输数据。

下面说一些我们融合技术的一些具体案例。我们依赖于 Flume+Avro 实现了内部的 ETL 方案，分布式结构化日志收集。目前我们部署的节点已经超过了一千个，每天数据的收集超过上千亿。我们用 Avro 封装了 FlumeEvent，实现了结构化日志收集。以前在 Log 中大部分日志都是文本，但是现在支持 Log 自定义的数据结构体。另外也支持一些通用的 Array 或者 Map 等这种数据类型。我们得益于 Avro，它传输数据更简洁，速度更快。现在我们每天的千亿数据都通过这种方式实时收集。在数据收集阶段，我们还要注意一些流处理与其他系统去做对接。Flume-ng 提供了一个支持二次开发的 SDK，方便业务类功能扩展。

围绕 Flume-ng 的优化

围绕着 Flume-ng 还做了很多的优化工作，其实我们在一开始做技术选型的时候，Flume-ng 当时还不太成熟，也遇到了很多问题。我们没有放弃，比如说我们以 AvroFile 为缓存，实现了一个新的 File ChannelPlus，极大的提升了速度和稳定性。它本身的 File ChannelPlus，出于安全和可靠性的保障，性能比较差，并且经常出问题。后来我们就重写了一个 File ChannelPlus。现在 FileChannelPlus 吞吐基本上达到每秒钟 6 万左右的 TPS。

我们还改进了 HDFS 的端的存储接口，支持了 Stable。我们数据收集上来要跟后面的数据处理流程要做对接。如果数据在接收并且在写一个文件的过程中，后面永远不会知道这个数据什么时候该处理。所以我们在这个 Sink 上实现了一个 Stable 的机制，数

据会定时的被放到一个 Stable 的目录，让这些数据变成可处理的状态。后面就会写触发条件去处理 Stable 的数据，就跟传输层能做一个很好的隔离。

另外我们还实现了分布式的节点监控和智能的配置管理服务，就是因为 Flume-ng 配置非常灵活，在上千个节点的部署上管理起来是非常麻烦的，那么我们实现了一个整个的配置管理中心服务，然后弥补了 Flume-ng 配置管理复杂上的这些问题。如果大家在实际开发过程中应用 Flume 的话，应用层应该是没有什么问题的，如果大家遇到什么样的问题，可以按照我们的思路，尝试着对它进行扩展。

日志收集系统 Loglib，用了 Flume、Avro 和 Solr 技术，实现了我们的分布式的实时日志检索的。我们每天的日志的索引量超过 15 个亿，一天的独立的索引记录数超过 15 个亿，支持几个月的记录，最近又改成了两个月。我们每天保证 15 亿的索引的稳定。另外我们做到了即用即搜。

开放平台统计分析

接下面来介绍核心的语音项目 Sunflower，语音云统计分析平台，和开放统计分析平台，是用什么技术来融合去做的呢？我们用了 Datasource + Avro-Mapreduce + Spark，来实现了语音云统计分析系统和开放统计分析系统。最一开始，我们面临的问题是日两亿次 PV，现在语音的服务量大概在两亿次左右。我们要每天在这个数据量上做大概 7 大类，50 多个小的类的统计工作，综合指标大概有上千个左右。最开始，我们尝试基于 Hive 去实现这样的统计工作，后来我们发现通过分解所有的需求，分解出来的 Sql 语句都有上千句，运行非常缓慢。我们又尝试基于 Pig，但是 Pig 的脚本也有几百行，执行速度也非常慢。我们开始对分布式技术进行一些思考，为什么 Hive 和 Pig 会这么慢？根本点在什么？因为我们有很多的指标，很多纬度。在指标和维度分解出来以后会形成很多的 Pig 和 Hive 语句。每个语句在执行的时候，都要对数据进行 Load，进行分布式处理。同一份数据被反复的 Load，非常耗费时间。对同一份数据的不同纬度和指标的统计分析，能否一次完成任务？计算结果的中间数据是否能够被重复利用？根据小时报表，其实可以重新计算出来日报表。我们围绕这些优化方向，形成了我们新的一个 Maple-Report，全新的统计分析解决方案。

我们通过报表定义与计算的分离，实现了多引擎的支持。Report Engine 目前支持 Avro-MapReduce，依赖于 Mapreduce 这样分布式计算的实现，还有 Spark 这样的更高速的实现，依赖于内存的实现。我们也真正的实现了同数据源的多维度，多指标，一次性计算完成，小时日周数据可以循环依次利用。20 分钟左右就能得到日志报表，同样周报表得到的时间也非常短，月报表，甚至半年报表也没有什么困难。Maple-Report 综合解

决方案，我们上线运行很长时间了。语音云统计分析系统，和开放统计分析系统，都是依赖于我们这套方案去做的。

整个 Maple 承载着公司级的大数据战略，像现在整个云平台、研究院、平嵌、移动互联和智能电视，都通过我们的 Maple 平台进行数据和技术的共享。另外，我们面向互联网的好多产品，包括讯飞开放平台、语音输入法、灵犀、酷音铃声，所有的数据均汇集到了 Maple 开放平台。然后很多小组都使用这个系统去分享挖掘数据。整个系统还在不断的发展中，公司整个战略是要把所有的产品线上的数据都汇集到一个平台上，将来能够都提供技术 & 数据分享，能够深入的挖掘数据的价值。

总结致敬

最后我想发表一些感慨，向那些以 Doug Cutting 为代表的，依然耕耘在技术前线，勤于 Coding 的前辈表示敬意。他们的分享和贡献精神，带给了我们实实在在的大数据技术。国内有一个很不好的一个想法，我也听过很多人讲，就是我多少多少岁以后就不做开发了。其实这个思想，我觉得大家应该适当的有些改变，应该以前辈为表率，去学习他们那种无论到什么时候，都能埋头去 Coding 这种精神！我到现在也在做 Coding 的一些事情，这是我们需要在整个技术上面需要形成这种风气。

孙利兵，科大讯飞云平台研发部资深大数据架构师，早期就曾接触 Hadoop 分布式计算技术，对 Mapreduce 分布式计算亦有很深刻的理解。精通 Avro 技术，在 2008 年曾编写了应用于实际环境的 AvroMapreduce 编程库，对 Doug Cutting 的以 Hadoop 为核心，Avro 为关键技术的 EcoSystem 构想非常向往，并推进在科大讯飞云计算组工作中进行实践，打造了 Maple（大数据开放平台）。

版权声明

InfoQ 中文站出品

架构师特刊：Hadoop 十年回顾

©2016 极客邦控股（北京）有限公司

本书版权为极客邦控股（北京）有限公司所有，未经出版者预先的书面许可，不得以任何方式复制或抄袭本书的任何部分，本书任何部分不得用于再印刷，存储于可重复使用的系统，或者以任何方式进行电子、机械、复印和录制等形式传播。

本书提到的公司产品或者使用到的商标为产品公司所有。

如果读者要了解具体的商标和注册信息，应该联系相应的公司。

出版：极客邦控股（北京）有限公司

北京市朝阳区洛娃大厦 C 座 1607

欢迎共同参与 InfoQ 中文站的内容建设工作，包括原创投稿和翻译，请联系 editors@cn.infoq.com。

网 址：www.infoq.com.cn

Geekbang.

极客邦科技

整合全球优质学习资源，帮助技术人 and 企业成长

InfoQ

技术媒体

EGO

职业社交

StuQ

在线教育

Git

企业培训



扫一扫关注InfoQ