Hadoop YARN大数据计算框架及其资源调度机制研究

董春涛 李文婷 沈晴霓 吴中海
北京大学 北京 100086

摘 要 Hadoop 2.0提出一种新的资源管理系统YARN，它可以支持多种大数据计算框架(如MapReduce、Storm、Spark等)，目前已经成为腾讯、阿里等互联网公司部署大数据平台采用的主流计算框架。为此，文章在分析Hadoop YARN计算框架的基本结构和工作流程的基础上，重点研究Hadoop YARN中的资源调度机制，包括其资源调度器的模型与机制，及其目前已经实现的容量调度器(Capacity Scheduler)和公平调度器(Fair Scheduler)的设计思想、工作原理和脆弱性，并探讨其下一代欧米伽调度器(Omega Scheduler)的主要设计思想。

关键词 大数据；YARN计算框架；资源管理器；容量调度器；公平调度器；欧米伽调度器；拒绝服务攻击

引言

大数据技术目前已成为学术界和产业界的研究热点。Google公司提出的GFS(Google File System)、MapReduce、BigTable[1-3]等技术成为了大数据技术发展的重要基础，而Apache软件基金会基于这些技术推出的开源项目Hadoop[4]成为大数据技术发展和应用的标志性成果，许多互联网公司(如Yahoo、IBM、百度、Facebook等)的大数据平台都是以Hadoop为主，它们或自建Hadoop集群、或使用Amazon Elastic MapReduce服务。

在Hadoop1.0版本中，MapReduce(也被称为MRv1)分布式处理框架是Hadoop中的唯一计算框架，它不仅能够用于离线处理大规模非结构化数据，而且能将很多繁琐的细节隐藏，比如，自动并行化、负载均衡和灾备管理等，极大地简化了开发工作，同时，与传统的大多数分布式处理框架相比，MapReduce的伸缩性优势明显，因此，MRv1最初推出的几年，有众多的成功应用案例，并获得业界的广泛支持和肯定。但随着分布式系统集群的规模和其工作负荷的增长，特别是支持其他实时计算框架的需求越来越多，包括内存计算框架(Spark)、流式计算框架(Storm)、迭代式计算框架(iMapReduce)等新型计算框架的出现，MRv1计算框架的局限性日益突出，主要包括扩展性差、资源利用率低、存在单点故障、计算框架单一等问题[5]。为此，Hadoop 2.0提出一种新的资源管理系统YARN[6-7](也被称为MRv2)，一个多种计算框架通用的资源调度体系，为不同的并行化计算提供资源分配服务。这样，YARN支持的计算框架只要实现YARN定义的接口，便可以运行在YARN之上，从而很好地打造一个以YARN为核心的生态系统。由于YARN具有灵活且支持多计算框架的架构设计、主结点功能的分离、资源调度机制的改进、资源的隔离和Hadoop原生支持等诸多特性，它目前已经成了新一代资源管理的典型代表，许多互联网公司，如阿里的云梯集群[8]、腾讯的Gaia平台[9]等就是基于YARN建立的大数据平台。

Hadoop 2.0 YRAN架构的核心是资源管理器(Resource Manager，RM)，而资源管理器的核心是

基金项目：国家自然科学基金重点项目(61232005)；国家863计划(SS2015AA011710)；深圳科技攻关项目(JSGG20140516162852628)

资源调度器(Resource Scheduler，RS)。本文以分析Hadoop YARN的基本架构和工作流程为基础，重点研究Hadoop YARN中的资源调度器使用的模型和机制，及其自带的容量调度器(Capacity Scheduler)[10]和公平调度器(Fair Scheduler)[11]，最后将探讨下一代调度器--欧米伽调度器(Omega Scheduler)[12]的主要设计思想。

第1节分析YARN的基本架构和工作流程，第2节分析YARN资源调度器使用的模型，第3节分析YARN资源调度器的机制，第4节对YARN现有调度器和下一代调度器进行分析。

1 YARN基本架构

YARN基本设计思想是将原MapReduce架构中JobTracker的两个主要功能，即资源管理和作业调度/监控分成两个独立组件，全局的ResourceManager和与每个应用相关的ApplicationMaster。下面我们从基本组成结构和工作流程两个方面分析YARN计算框架。

1.1 YARN基本组成结构

YARN的基本组成结构如图1所示。YARN总体上仍然是一个Master/Slave结构，在整个YARN资源管理框架中，资源管理器(ResourceManager)为Master，节点管理器(NodeManager)为Slave，ResourceManager负责对各个NodeManager上的资源进行统一管理和调度。用户提交应用程序时，需要提供一个跟踪和管理这个程序的应用程序主控节点(ApplicationMaster)，由它向ResourceManager申请资源，并要求NodeManager按ApplicationMaster申请到的Container资源信息来启动任务。

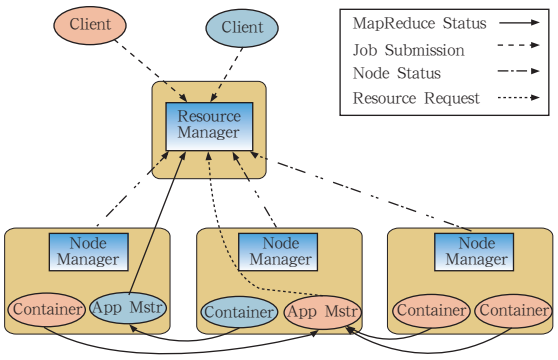

图1 YARN的基本组成结构

通过上述描述可知，YARN主要由Resource Manager、NodeManager、ApplicationMaster和Container等四个组件构成。它们的基本组成和功能描述如表1所示。

表1 YARN的组件及其功能描述

组件名称	功能描述
Resource Manager (RM)	全局的资源管理器，负责整个系统的资源管理和分配。主要包括一个资源调度器，负责分配资源给各个正在运行的应用程序；一个应用程序管理器，它负责整个系统中应用程序的启动和关闭、访问权限、资源使用期限等
Application Master (AM)	用户提交的每个应用程序均包含一个应用程序主控节点AM，负责跟踪和管理应用程序，主要负责：①与RM的资源调度器协商以获取资源；②将得到的资源分配给内部任务；③与NM通信以启动/停止任务；④监控所有任务运行状态，并在任务运行失败时重新运行任务
Node Manager (NM)	负责每个节点上资源和任务的管理，主要负责：①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；②接收并处理来自AM的Container启动/停止等请求
Container	容器是动态资源(内存、CPU、磁盘、网络等)的分配单位，负责封装某个节点上的资源，当AM向RM申请资源时，RM为AM返回的资源以Container表示

ResourceManager的资源调度器是一个“纯调度器”，它不再从事任何与具体应用程序相关的工作，比如不负责监控或者跟踪应用的执行状态，也不负责重新启动运行失败的任务，这些均交由ApplicationMaster完成。

1.2 YARN的工作流程

当用户向YARN提交应用程序时，YARN分两个阶段运行该应用程序：第一个阶段启动ApplicationMaster；第二个阶段由ApplicationMaster为应用程序申请资源，并监控整个运行过程，直到完成。YARN的工作流程如图2所示，主要分为以下几个步骤。

1) 用户向YARN提交应用程序，其中包括用户程序、启动ApplicationMaster命令等。

2) ResourceManager为该应用程序分配第一个Container，并与对应的NodeManager通信，要求它启动应用程序的ApplicationMaster。

3) ApplicationMaster向ResourceManager注册后，为各个任务申请资源，并监控它们的运行状态，直到运行结束。

4) ApplicationMaster采用轮询的方式通过RPC协议向ResourceManager申请和领取资源。

5) ApplicationMaster申请到资源后，便与对应的NodeManager通信，要求它启动任务。

6) NodeManager为任务设置好运行环境(环境变量、JAR包、二进制程序等)后，将任务启动命令写到脚本中，并通过运行该脚本启动任务。

7) 各个任务通过某个RPC协议向ApplicationMaster汇报自己的状态和进度，可以在任务失败时重新启动任务。

8) 应用程序运行完成后，ApplicationMaster向ResourceManager注销并关闭自己。

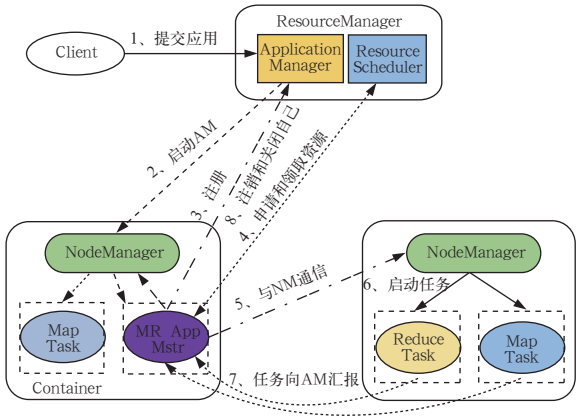

图2 YARN的工作流程

在整个工作流程中，YARN的资源调度器主要关注ApplicationMaster如何向ResourceManager申请和领取资源，这是YARN工作的核心。下面将重点分析YARN的资源调度器相关的模型、机制，以及其实现的两种调度器和下一代调度器。

2 YARN资源调度器的模型

在MRv1和YARN中，资源调度器的实现原理基本一致，不同的是YARN采用了事件驱动的编程模型和独特的资源表示模型。YARN的资源调度器更为复杂，这就要求用户在研究YARN现有调度器的工作原理和编写新的调度器之前，必须先理解其资源调度器采用的模型。下面分别介绍YARN调度器采用的两种模型。

2.1 事件驱动模型

YARN的资源调度器采用事件驱动的编程模型。它实际上成为一个事件处理器，需要处理来自外部的6种类型的事件：NODE_REMOVED、NODE_ADDED、APPLICATION_ADDED、APPLICATION_REMOVED、CONTAINER_EXPIRED和NODE_UPDATE，如表2所示。其中，NODE_UPDATE是6个事件中最重要的，如果此时有Container得到释放，则它会触发资源调度器最核心的资源分配机制，触发资源分配。

表2 资源调度器需要处理的事件

事件	事件含义	资源调度器进行的处理
NODE_REMOVED	集群中移除一个计算节点	从分配资源总量中移除相应的资源量
NODE_ADDED	集群中增加一个计算节点	将新增资源量添加到可分配资源总量中
APPLICATION_ADDED	RM收到一个新的AM	将该应用程序添加到相应的队列中
APPLICATION_REMOVED	一个应用程序运行结束	将该应用程序从相应的队列中清除
CONTAINER_EXPIRED	分配给AM的Container失效	对该Container进行回收再分配
NODE_UPDATE	RM收到NM的心跳信息	如果有Container得到释放，则会触发资源分配

2.2 资源表示模型

YARN同MRv1一样，采用动态资源分配机制。不同的是，YARN中资源的表示方式是容器(Container)，而不是槽位(Slot)。Container是YARN中资源的抽象，封装了某个节点上一定量的资源。与资源量固定的Slot相比，Container资源量动态可变，更加灵活。Container主要包含优先级、期望资源所在节点、资源量、Container数目和是否松弛本地性5类信息。

当前YARN支持内存和CPU两种类型资源的管理和分配。NodeManager启动时会向ResourceManager注册，注册信息中包含该节点可分配的CPU和内存总量。YARN在将来还会支持磁盘容量、网络和磁盘I/O等资源。

YARN为了更加友好地为应用程序分配资源，定义了一些调度语义。当前YARN支持的调度语义包括：请求某个节点上的特定资源量、请求某个特定机架上的特

定资源量、将某些节点加入或移出黑名单和请求将资源归还给集群。随着YARN的发展，YARN将支持更多的调度语义。

3 YARN资源调度器的机制

资源调度机制是YARN资源调度器的核心。YARN资源调度器采用的调度机制主要包括：双层资源调度机制、层级队列管理机制、资源保证和资源抢占机制。其中双层资源调度机制是其核心，是YARN进行资源分配的总体架构；层级队列管理机制是YARN对上层资源分配队列的管理方式；资源保证和资源抢占机制是YARN保证任务资源需求的机制。下面分别介绍YARN资源调度器采用的三种机制。

3.1 双层资源调度机制

YARN采用双层资源调度机制，如图3所示。第一层是资源管理系统将资源分配给应用程序；第二层是应用程序将收到的资源分配给内部任务。资源调度器主要关注第一层的调度问题，第二层的调度策略则由用户应用程序的ApplicationMaster决定。

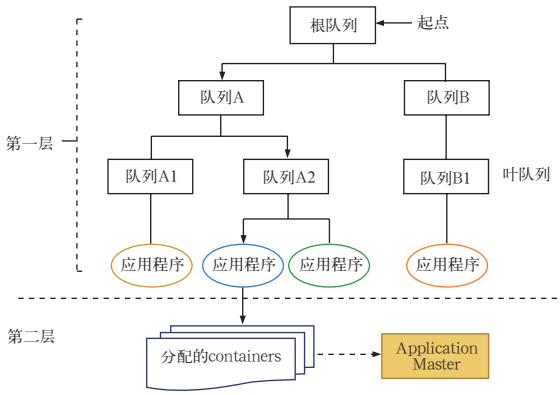

图3 YARN双层资源调度机制

YARN的资源分配过程是异步的，采用了基于拉模式(pull-based)的通信模型。资源调度器将资源分配给一个应用程序后，不会立刻推送给对应的ApplicationMaster，而是暂时放到缓冲区，等待ApplicationMaster通过周期性的心跳来取。YARN的资源分配过程如图4所示。

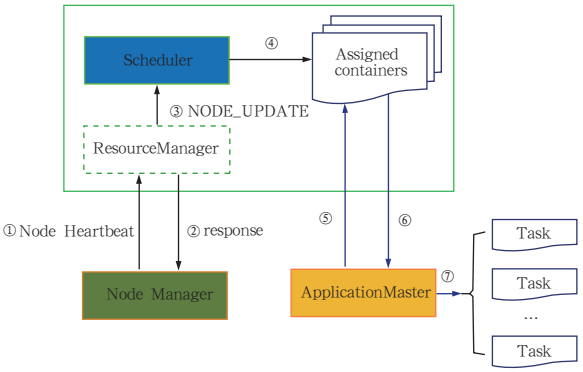

图4 YARN的资源分配过程

NodeManager需要通过周期性的心跳向ResourceManager汇报节点信息。ResourceManager收到信息后，返回心跳应答，包括需释放的Container列表等信息。同时，会触发一个NODE_UPDATE事件，如果此时有Container得到释放，则该事件会触发资源分配。资源调度器收到事件后，按照一定的策略将该节点上的资源分配给应用程序，并将分配结果放到一个内存数据结构中。

应用程序的ApplicationMaster需要向ResourceManager发送周期性心跳，以领取最新分配的Container。ResourceManager收到信息后，将分配的Container以心跳应答的形式返回给ApplicationMaster。ApplicationMaster收到新分配的Container列表后，将这些Container分配给内部任务。

3.2 层级队列管理机制

用户和资源管理机制是任何资源调度器的基础。在YARN中，用户以层级队列的形式组织。该队列组织方式具有队列可嵌套、最少容量和最大容量等特点。每个队列被划分了一定比例的资源。每个用户可属于一个或多个队列，且只能向这些队列提交应用程序。为防止队列名称冲突和便于识别队列，YARN采用了自顶向下的路径命名队列。

通常而言，不同的调度器对资源管理的方式是不同的，第4节将具体分析容量调度器和公平调度器的层级队列管理机制的具体工作原理。

3.3 资源保证与抢占机制

YARN作为一个资源分配系统，必须保证任务的资源需求。如果发生资源暂时无法满足任务需求和资源被其他任务占用等情况，YARN必须有相应的机制来解决这些问题。YARN当前主要采用资源保证和资源抢占两种机制来保证任务的资源需求。

分布式计算框架中，资源调度器主要有两种资源保证机制：增量资源分配和一次性资源分配。1)当应用程序申请的资源暂时无法保证时，增量资源分配：优先为它预留一个节点上的资源，直到累计释放的资源满足需求；2)一次性资源分配：暂时放弃当前资源直到一个节点剩余资源一次性满足需求。两种机制均有缺点，增量资源分配进行资源预留会导致资源浪费，降低资源利用率；而一次性资源分配会产生“饿死”现象，YARN采用了增量资源分配。

资源调度器为每个队列设置一个最大和最小资源量。最大资源量是队列资源量的上限，最小资源量是在资源紧缺时每个队列需要保证的资源量，是资源抢占发生的原因。通常为提高资源利用率，调度器会将负载较轻的队列的资源暂时分配给负载较重的队列。当负载较轻的队列收到新提交的应用程序时，调度器会将本属于该队列的资源分配给它，但需要等其他队列释放资源。为防止等待时间过长，调度器等待一段时间后若发现资源并未得到释放，则进行资源抢占。

4 YARN的资源调度器与下一代调度器

Google把资源调度器分为三代[12]，第一代是独立的集群调度，第二代是双层调度(Mesos，YARN)，第三代是共享状态调度(Omega)。YARN是双层调度的代表，实现了FIFO、Capacity Scheduler和Fair Scheduler三种调度器。FIFO作为简单的批处理调度器，不能满足多样化需求和充分利用资源，因此，YARN用Capacity Scheduler取代FIFO调度器作为默认调度器。Capacity Scheduler和Fair Scheduler属于多用户调度器，采用树形多级队列形式组织资源，更适合应用需求。下面分别介绍这两种多用户资源调度器及其存在的脆弱性和第三代资源调度器Omega。

4.1 容量调度器

容量调度器(Capacity Scheduler)是Yahoo开发的多用户调度器，以队列为单位划分资源，每个队列可设定资源最低保证和使用上限。当队列的资源有剩余时，可暂时将剩余资源共享。Capacity Scheduler主要有以下特点：容量保证、灵活性、多重租赁、安全保证和动态更新配置文件等。下面具体分析Capacity Scheduler的工作原理。

应用程序提交后，ResourceManager会向Capacity Scheduler发送一个事件，Capacity Scheduler收到后，将为应用程序创建一个对象跟踪和维护其运行时的信息，同时，将它提交到对应的叶子队列中，队列会对应用程序进行合法性检查。通过检查，应用程序才算提交成功。提交成功后，它的ApplicationMaster会为它申请资源，Capacity Scheduler收到资源申请后，暂时将这些请求存放到一个数据结构中，以等待为其分配合适的资源。

NodeManager发送的心跳信息有两类需要Capacity Scheduler处理：一类是最新启动的Container；另一类是运行完成的Container，Capacity Scheduler将回收它使用的资源进行再分配。Capacity Scheduler采用三级资源分配策略(即将双层调度机制中的第一层分为选择队列与选择应用程序两级)，当一个节点上有空闲资源时，它会依次选择队列、应用程序和Container(请求)使用该资源，接下来介绍三级资源分配策略。

第一级：选择队列。Capacity Scheduler采用基于优先级的深度优先遍历算法选择队列：从根队列开始，按照资源使用率由小到大遍历子队列。如果子队列是叶子队列，则按照第二、三级的方法在队列中选择一个Container，否则以该队列为根队列，重复以上过程，直到找到合适的Container并退出。

第二级：选择应用程序。选中一个叶子队列后，

Capacity Scheduler按照Application ID对叶子队列中的应用程序进行排序，依次遍历排序后的应用程序，找到合适的Container。

第三级：选择Container。选中一个应用程序后，先满足优先级高的Container。同一优先级，先满足本地化的Container，依次选择节点本地化、机架本地化和非本地化的Container。

4.2 公平调度器

公平调度器(Fair Scheduler)是Facebook开发的多用户调度器，与Capacity Scheduler有很多相同之处。Fair Scheduler与Capacity Scheduler的不同主要体现在资源公平共享、负载均衡、调度策略配置灵活和提高小应用的响应时间等方面。

Fair Scheduler也需要进行程序的合法性检查、处理心跳信息和资源分配等工作，同样采用三级分配策略。不同的是，Fair Scheduler提供了更多样化的调度策略，调度策略在队列间和队列内部可单独设置。当前有三种策略可选，分别是先来先服务(FIFO)、公平调度(Fair)[12]和主资源公平调度(Dominant Resource Fairness，DRF)[13]。1)先来先服务：按优先级高低调度，优先级相同，则按提交时间先后顺序调度；提交时间相同，则按名称大小调度。2)公平调度：按内存资源使用比率大小调度。3)主资源公平调度：按主资源公平调度算法进行调度，所需份额最大的资源称为主资源，把最大最小公平算法应用于主资源上，将多维资源调度问题转化为单资源调度问题。

DRF算法伪代码：

R = <r1, … , rm> //m种资源对应容量

C = <c1, … , cm> //已经使用资源量，初始为0

si (i = 1…n) //用户i的主资源所需份额，初始为0

Ui = <ui,1, … , ui,m> (i = 1…n) //分配给用户i的资源量，初始为0

挑选出所需主资源si最小的用户i；

Di ← 用户i下一个任务所需的资源量

if C + Di <= R then

C = C + Di //更新C

Ui = Ui + Di //更新U

$S_i = \max_{j=1}^{m}\{u_{i,j}/r_j\}$

else

return //资源已经用完

end if

随着Hadoop版本的演化，Fair Scheduler和Capacity Scheduler的功能越来越完善，两者同质化也越来越严重。两者在应用场景、支持的特性、内部实现等方面非常接近，而由于Fair Scheduler支持多种调度策略，可认为Fair Scheduler具备了Capacity Scheduler的所有功能。

4.3 资源调度器的脆弱性

资源调度器作为YARN的核心组件，其安全性与可用性对整个系统起到至关重要的作用，当前的资源调度器存在许多脆弱性问题需要解决。

目前，已经发现有多租户的YARN系统存在DoS攻击的安全隐患[14]。这主要是由于YARN当前所实现的资源调度器最多只对内存和CPU两种资源进行限制，却没有对其他的计算资源(网络带宽、磁盘读写等)进行限制，恶意用户可以通过使用受限制的合法资源在节点上启动恶意计算任务，大量地占用没有进行限制的资源，从而影响节点上其他计算任务的正常运行。如果恶意用户可以用有限的受限制的资源启动尽可能多的恶意计算任务，并将它们分配到尽可能多的节点上，那么就会影响整个系统的计算能力，甚至导致整个系统不可用。要解决这一脆弱性，资源调度器必须对其他的计算资源进行合理的限制。

除此之外，YARN的资源调度器还存在资源本地化考虑不够全面、向ApplicationMaster告知Container的实际物理地址等许多脆弱性问题需要研究和解决。

4.4 第三代资源调度器

YARN作为第二代调度器的代表在日益完善但并非完美。调度机制还在快速发展中，为克服双层调度机制

的缺点，Google开发了下一代资源管理系统Omega。

Omega是一种基于共享状态的调度器，没有集中式调度模块，应用程序的调度器进行自我管理和控制。Omega将双层调度器中的集中式调度模块简化成持久化的共享数据(整个集群的实时资源使用信息)和针对这些数据的验证代码。

由于没有集中式调度模块，Omega不能像YARN一样在一个统一模块中对整个集群中的资源分组、限制每类应用程序的使用量和限制每个用户的使用量等，这些功能由各个应用程序的调度器实现。Omega将优先级这一限制放到共享数据的验证代码中，当有多个应用程序同时申请一份资源时，优先级最高的那个应用程序将获得该资源，其他资源限制全部下放到各个子调度器。

5 结束语

大数据时代，资源管理方式正在发生变革。在大数据的主流应用平台Hadoop系统中，掌控好资源调度就抓住了整个Hadoop系统的核心。YARN作为一个独立的资源管理系统是新一代Hadoop中最核心的组件，代表了大数据平台的发展趋势。资源系统是Hadoop YARN系统的研究重点，而资源调度机制是资源调度系统核心。分析和研究YARN的资源调度机制，不仅有助于部署平台，更有助于明确资源调度系统存在的脆弱性与不足，不断完善资源调度系统，为构建满足不同实际应用需求的大数据平台奠定基础。

参考文献

[1] Ghemawat S,Gobioff H,Leung S T.The Google file system[C]//The Nineteenth ACM Symposium on Operating Systems Principles,2003

[2] Jeffrey Dean,Sanjay Ghemawat.MapReduce: simplified data processing on large clusters[J]. Communications of the ACM-50th anniversary issue:1958-2008,2008,1(51):107-113

[3] Fay Chang,Jeffrey Dean,Sanjay Ghemawat,et al. Bigtable:A Distributed Storage System for Structured Data[EB/OL].[2015-01-07].http://labs.google.com/papers/bigtable.html

[4] Apache hadoop[EB/OL].[2015-01-07].http://hadoop.apache.org.

[5] Apachetez[EB/OL].[2015-01-07].http://incubator.apache.org/projects/tez.html.

[6] Vinod Kumar Vavilapalli,Arun C Murthy,Chris Douglas,et al.Apache Hadoop YARN:Yet Another Resource Negotiator[C]//The Fourth ACM Symposium on Cloud Computing(SoCC'13),2013

[7] 董西成.Hadoop技术内幕:深入解析YARN架构设计与实现原理[M].北京:机械工业出版社,2013:153-184

[8] 杨卓荦.基于YARN构建多功能分布式集群[J].程序员,2013,(11):105-107

[9] SACC2014:腾讯资源调度平台Gaia分享[EB/OL].[2015-01-07].http://cloud.it168.com/a2014/0918/1667/000001667383.shtml

[10] Hadoop Capacity Scheduler[EB/OL].[2015-01-07].http://hadoop.apache.org/common/docs/r2.2.0/capacity_scheduler.html

[11] Hadoop Fair Scheduler[EB/OL].[2015-01-07].http://hadoop.apache.org/common/docs/r2.2.0/fair_scheduler.html

[12] Malte Schwarzkopfy,Andy Konwinskiz,Michael Abd-El-Malekx.Omega flexible, scalable schedulers for large compute clusters[C]//ACM SIGOPS European Conference on Computer Systems(EuroSys 2013)

[13] Ali Ghodsi,Matei Zaharia,Benjamin Hindman.Dominant Resource Fairness:Fair Allocation of Multiple Resource Types[R].Technical Report No.UCB/EECS-2011-18. March 6,2011

[14] Huang J W,Nicol D M,Campbell R H.Denial-of-Service Threat to Hadoop/YARN Clusters with Multi-Tenancy[C]//2014 IEEE International Congress on Big Data

作者简历

董春涛

硕士研究生，研究方向为大数据安全与隐私。

李文婷

硕士研究生，研究方向为大数据安全与隐私。

沈晴霓

博士，教授，研究方向为操作系统与虚拟化安全、云计算和大数据安全与隐私与可信计算等。

吴中海

博士，教授，博士生导师，研究方向为情境数据融合、情境感知服务、云计算与大数据安全和高可信嵌入式软件。

Research on the Framework and Resource Scheduling Mechanisms of Hadoop YARN

Dong Chuntao
Li Wenting
Shen Qingni
Wu Zhonghai

Peking University, Beijing 100086, China

Abstract Hadoop 2.0 proposes a new resource management system, named as YARN, which can support a variety of big data computing frameworks(such as MapReduce, Storm, Spark, etc.). YARN has become the current mainstream computing framework to deploy large data platform for many internet corporations(e.g. Alibaba Group, Tencent, etc.). Therefore, this paper analyzes the basic structure and the working flow of the YARN framework, then focuses on its resource scheduling mechanisms used in the ResourceManager, including its resource management model, management mechanisms and vulnerabilities, and the implemented resource schedulers, the Capacity Scheduler and Fair Scheduler. Finally it also discusses the next generation of Omega scheduler.

Keywords Big Data; YARN Computing Framework; Resource Manager; Capacity Scheduler; Fair Scheduler; Omega Scheduler; Denial of Service Threat