# Hadoop云平台的一种新的任务调度和监控机制

许 丞[1] 刘 洪[1] 谭 良[1,2]

(四川师范大学计算机科学学院 成都 610068)[1] (中国科学院计算技术研究所 北京 100190)[2]

**摘 要** 云平台任务监控与资源调度机制是云平台的核心功能之一。Hadoop云平台中任务监控和资源管理的任务是由JobTracker负责处理,并通过slave节点向其发送心跳消息来实现。这种方式导致JobTracker的负载过重,降低了Hadoop云平台的工作效率,限制了Hadoop云平台的规模。提出了一种新的任务监控方案,该方案将JobTracker的任务监控和资源管理功能分离,任务监控功能仍由JobTracker节点完成,资源管理功能由新增的资源管理节点完成,JobTracker通过增量更新的算法将任务调度所需的对象信息动态同步到资源管理节点上,资源管理节点根据心跳消息进行任务分配,并将分配结果返回给JobTracker节点。实验结果表明,本方案不仅通过监控节点实现了任务的监控,增加了监控的灵活性和鲁棒性,而且降低了Jobtracker节点的负担,可有效提高Hadoop云平台的工作效率和规模。

**关键词** 云计算,Hadoop,任务监控,任务调度,资源管理,增量更新算法

**中图法分类号** TP309 **文献标识码** A

## New Mechanism of Monitoring on Hadoop Cloud Platform

XU Cheng[1] LIU Hong[1] TAN Liang[1,2]

(College of Computer, Sichuan Normal University, Chengdu 610068, China)[1]

(Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China)[2]

**Abstract** Job monitoring and resource managing mechanism in cloud computing platform are the core functions. The tasks of job monitoring and resource managing in Hadoop are assigned to JobTracker which is implemented on receiving heartbeat message sending by slave node, so JobTracker is the bottleneck of Hadoop. A new mechanism of job monitoring and resource managing scheme was presented. In this scheme, job monitoring and resource managing are separated from original JobTracker, but job monitoring function is still assigned to JobTracker node, and the resource managing function is accomplish by newly added resource monitoring node. JobTracker sends the necessary information of Objects to resource managing node by using a new method of delta updates algorithm. Resource managing node schedules tasks with heartbeat message, and returns the result to JobTracker node. Experimental results show that the scheme implements monitoring node monitor job completely, makes JobTracker node more flexible and robust, reduces the loading of JobTracker node, improves the efficiency of Hadoop platform.

**Keywords** Cloud computing, Hadoop, Task scheduling, Resource managing, Delta updating algorithm

## 1 引言

随着信息量的急速增加,信息处理速度的需求也与日俱增。信息处理能力的增长方式也发生了巨大的变化。“云计算”是一种新兴的商业计算模型,通过这种方式,用户可以在不需要了解“云”内部构造的情况下,向终端用户提供海量的计算能力。在众多提供云平台的软件中,Hadoop[1]是一款重量级的基础平台。它是Google提出的MapReduce编程模型的开源实现,其思想来源于函数式编程语言,非常适用于大规模数据(大于1TB)的处理。在处理TB和PB级数据方面,MapReduce已经成为使用最为广泛的并行编程模型之一[2]。许多著名的IT企业都将Hadoop作为自己的业务平台,包括Yahoo!、facebook、IBM和Twitter等[3]。在中科院计算所主办的第四届“Hadoop in china 2010中国云计算大会”[4]上,中科院计算所常务副所长孙凝晖指出“Hadoop功能所限制的架构比商业软件好,受到大家的追捧,国内很多大的互联网去哪个公司,他们的IT系统还有上面的应用基本上是以Hadoop为基础架构的”。

JobTracker是Hadoop的核心部件,主要完成云平台任务监控与资源调度功能。当Hadoop接收到来自客户端的任务时,JobTracker通过映射的方式,把需要处理的任务分布到各节点上并行运算,再通过化简的方式,对各节点返回的结果

到稿日期:2012-04-05 返修日期:2012-08-01 本文受国家自然科学基金(60970113),国家自然科学基金青年基金(60903073),四川省教育厅青年基金项目(08zb02),四川师范大学校级项目(11KYL03)资助。

许 丞(1988—),男,硕士生,主要研究领域为云计算、信息安全,E-mail:lonemoonstar@126.com;刘 洪(1977—),男,硕士,讲师,主要研究领域为Web开发;谭 良(1972—),男,博士,教授,CCF高级会员,主要研究领域为可信计算、网络安全。

进行处理。为保证任务的可靠性，Hadoop 平台引入了任务监控机制。具体办法是：每隔一段时间，slave 节点就会向 Jobtracker 节点发送心跳消息，Jobtracker 接受心跳消息并对整个框架负责任务的调度和监控，以及重新执行已经失败的任务。显然，JobTracker 兼具资源管理和任务监控这两个核心任务，而在一个 Hadoop 集群中，只存在一个 JobTracker 节点，因此 JobTracker 的负载会常常过重[5]，使得 JobTracker 工作效率下降，影响了集群的整体的效率，限制了 Hadoop 集群的规模。

为降低 JobTracker 节点负担，减少对 Hadoop 集群规模的限制，提高 Hadoop 集群的工作效率，在 Hadoop 云平台中需要改进任务监控机与资源管理机制，在保证任务可靠性的前提下，降低 JobTracker 的工作负担。基于此，本文提出了一种新的任务监控与资源管理机制，该机制将 Hadoop 的两个核心功能：任务监控与资源管理分开，任务监控由去掉资源管理功能的 JobTracker 节点承担，本文称它为 Monitor Node 节点。新增一个资源管理节点承担资源管理任务，slave 节点定期向 Monitor Node 节点和资源管理节点（ResManage Node）发送心跳消息，Monitor Node 在处理完心跳消息后，通过一种增量更新的算法将资源管理所需的信息动态同步到资源管理节点上，资源管理节点根据收集到的心跳消息进行任务分配，并将分配结果返回给 Monitor Node 节点。

## 2 相关工作

Hadoop 的任务调度与资源管理问题一直是 Hadoop 云计算框架领域的研究热点之一。由于 Hadoop 中的各个节点的能力与任务需求的差异，如何为任务分配最合适的资源并使得资源管理最优化，一直是该领域的重点研究对象。

2009 年，北加州大学对 Hadoop 任务调度模型中默认的 FIFO 调度方法进行了分析[6]，指出了其中的不足，提出一种基于作业最后完成期限的调度方式。这种方式的调度不依赖于集群中运行的作业的数量，而是将用户对任务的截止完成时间作为输入的一部分，并以此决定执行基于作业成本模型的调度。同年，HP 实验室提出了一种在 Hadoop 云计算框架下收集集群中共享资源和计算的系统[7]，该系统采用用户为作业分配服务等级、系统动态调整资源收集模式和动态估算与侦测集群系统瓶颈的策略来提高 Hadoop 中作业的调度效率。实验表明，采用该种方式作业的完成时间与不同种类作业的优化效率有了较大的提高。同年，伯克利大学经研究发现[8]，由于数据本地化和 map/reduce 任务间的依赖性问题，致使传统的调度算法在 MapReduce 作业的调度上表现欠佳。为解决该问题，研究人员提出了延迟调度和拷贝计算的方式，使 FAIR 调度器达到了独立、低延迟响应和和高吞吐的目标。2010 年，巴塞罗那超算中心与 IBM 沃森研究中心提出了一种新的 Hadoop 云计算框架下的任务调度机制[9]，在该机制中提出了一种基于 MapReduce 任务性能驱动模型的新型调度器。新型的任务调度器通过对获取提交的任务和尚未完成的任务以及已经完成的任务所花的时间进行监控，将这些信息用于预测作业的完成时间，对 Hadoop 运行中的应用进行实时调度。后来该文的作者又对 Hadoop 资源感知与调度调度机制中的 slot 资源表示机制进行了改进[10]，并提出了一种新型以工作需求为驱动的资源表示模型与调度机制。实验表明，在这种机制下，Hadoop 的资源的最大化利用率得到了显著提高，更加符合现实任务对 Hadoop 资源的需求。

虽然这些改进对提高 Hadoop 云计算框架的资源管理与任务调度都取得了一定的成果，但是 JobTracker 始终还是同时肩负资源管理与任务调度的职责，自身负载过大，特别是随着 Hadoop 集群规范的增大，JobTracker 成了整个 Hadoop 云计算框架的性能瓶颈。2010 年，为了优化 JobTracker 的性能，IBM Watson 和 Almaden 研究中心研究了 Hadoop 中的 HFS 调度策略[11]，并在此基础上提出了一种更为灵活的调度器。这种调度器既可以作为 HFS 的替代品，也可以与 HFS 调度器协同进行工作。与 HFS 相比，这种调度器不仅具有 HFS 调度器为作业保证最小资源和公平的特点，还可以选取并优化多种测量参数，但优化效果不够理想。2011，在 Yahoo 开发的下一代 Hadoop 计划[12]中，提出对 JobTracker 的任务监控和资源管理功能进行改进，为每个应用创建 ApplicationMaster，ApplicationMaster 用于任务的监控和调度，而 ResourceManager 用来管理资源的分配。

## 3 Hadoop 云平台的一种新的任务调度与监控机制

### 3.1 Hadoop 云平台现有任务监控和调度模型介绍

目前，Hadoop 云平台的体系结构模型如图 1 所示。

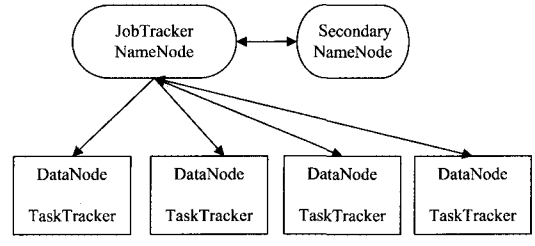


图 1 Hadoop 云平台的体系结构模型

一个 Hadoop 云平台中有唯一的一个 JobTracker 节点，负责集群的资源管理和任务的监控与调度。有多个 slave（TaskTracker）节点，负责执行具体的 MapReduce 任务，当有用户提交新的作业时，用户会先和 NameNode 节点联系，NameNode 会将用户需要处理的数据分配在云平台中的存储位置并告知用户，接着用户会把需要处理的数据上传到云平台中，保存在 Slave 节点上。Slave 节点在启动后，会向 JobTracker 发送心跳消息，JobTracker 接受到心跳消息后根据待处理数据的分配情况，向具体的 Slave 节点分配 Map/Reduce Task。Slave 启动后，会定期向 JobTracker 发送心跳消息（默认 5s），JobTracker 根据 Slave 发来的心跳消息和集群中资源的当前情况，向 Slave 节点分发任务。

Hadoop 云平台中现有任务监控和调度模型如图 1 所示。JobTracker 接到 TaskTracker 的 heartbeat 调用后，首先会检查上一个心跳响应是否完成、本条心跳消息是否来自于允许的 TaskTracker，如果一切正常，就会调用 processHeartbeat 方法处理这条心跳消息。processHeartbeat 首先会更新 TaskTracker 的状态和集群中 slot 数量等信息，然后根据心跳消息确定 TaskTracker 是丢失或者是新加入到集群中，然后分别调用不同的函数进行处理。接下来，将调用 updateTaskStatuses 方法更新 TaskTracker 上执行的任务的状态，并根据任务报告的状态：如失败、完成、普通等不同状态进行处理，对于异常的状态报告：如失败，将调用 failedTask 方法进行处

理，以保证任务的正确执行。如果任务报告状态及 TaskTracker 状态一切正常，并且请求新的 Task，JobTracker 将调用 TaskScheduler 的 assignTasks 方法分配新的 Task，并将分配的结果通过心跳消息返回给 TaskTracker。TaskTracker 从心跳消息中解析出自己需要执行的任务，完成一次心跳消息的交互。

通过 JobTracker 中任务监控和资源管理的过程的分析，发现 Hadoop 的资源管理功能就是根据一定的调度策略，将新任务分配到合适的 TaskTracker 上执行，以 Hadoop 平台默认的调度器为例，JobTracker 在执行资源管理的时候，通过调用实现 TaskScheduler 接口的 JobQueueTaskScheduler 类型实例的 assignTasks 方法进行任务调度。由图分析可知，如果存在一种方法可将调度所需的信息同步到其他节点上，让这个节点对 Hadoop 的资源进行管理，就可以降低 JobTracker 的负担，使 JobTracker 任务的执行效率更高。

### 3.2 Hadoop 云平台的一种新的任务调度和监控机制

从上面的分析可以看出，只要将调度所需的信息提取出来，就可以将资源管理功能从 JobTracker 中分离，有效降低 JobTracker 的负担。因此，本文提出 Hadoop 云平台中一种新的任务监控与资源管理功能的机制，即将 JobTracker 的功能分担到两个节点上完成。新机制中的 MonitorNode 节点实现任务监控部分的功能，而资源管理功能由 ResManageNode 节点实现。TaskTracker 的心跳消息首先发送 MonitorNode 节点，MonitorNode 节点对心跳消息进行处理，更新 TaskTracker 的状态和 TaskTracker 上 Tasks 的状态。状态更新完成后，通过增量更新的方式，传给 ResManageNode 节点，ResManageNode 节点根据增量更新后的 TaskTracker 和 Tasks 的状态信息，正确调度新的任务到 TaskTracker 节点上。由于 Task 失败和 TaskTracker 失效相对 Task 正确执行来说，出现的可能性小很多，因此增量更新的方式只会增加 MonitorNode 节点少量的负担，远小于调度 Task 所需的计算量。所以，MonitorNode 节点和 ResManageNode 节点的负担都将在一个相对较低的水平。

#### 3.2.1 MonitorNode 主要功能

如图 2 右上方所示，MonitorNode 节点主要实现原 JobTracker 节点中的任务监控功能，接受来自 TaskTracker 的心跳消息，并更新 TaskTracker 和该 TaskTracker 上 Tasks 的状态。Tasks 的状态更新将引起部分对象的改变，为保证 ResManageNode 节点正确执行 Task 调度的任务，MonitorNode 节点将对象更新后的最终值（简单数据类型变量）或对象的最终状态（对象类型的变量）所需的最少操作构造成一个 Object[] 类型的数组传给 ResManageNode 节点。MonitorNode 节点在创建 JobInProgress 类（用于监控和管理作业的执行）和 TaskInProgress 对象（用于监控和管理的任务执行）的时候，构造出一棵双向映射的树，数的每个节点由 4 部分组成，第一部分是父对象的引用，第二部分是对象本身的引用，第三部分是对象是子对象的引用数组，第四部分是对象相对于父对象节点的偏移地址，构造完成后，ResManageNode 解析这个数组，执行最必要的操作步骤，同步更新 TaskTracker 的状态和 Tasks 的状态。以 TaskInProgress. updateStatus (TaskStatus status) 方法为例，MonitorNode 节点在调用该方法更新 Task 状态的同时，在 Object[] 类型的数组中构造取得相关对象和变量的最少步骤，对于简单数据类型的变量，只需要将数据更新后的值和所属对象的引用编号放入 Object[] 类型的数组中并指明长度，即可完成更新。对于对象类型的变量，将对象的引用编号和执行操作的编号放入 Object[] 类型的数组中，传给 ResManageNode 节点。

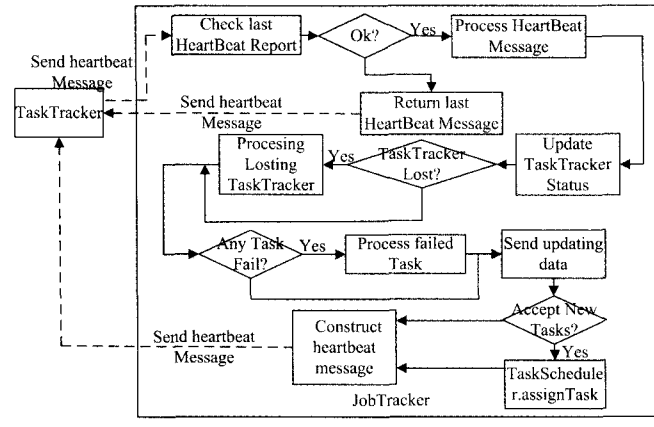


图 2 Hadoop 现有任务监控与资源管理机制

#### 3.2.2 ResManageNode 主要功能

ResManageNode 节点主要完成原 JobTracker 节点的资源管理功能，它在初始化时也需要构造一棵与 MonitorNode 节点编号相同的双向映射的树，用于正确解析 MonitorNode 节点发来的消息，更新相关对象的状态，并进行任务调度。由于调度本身也会引起部分对象状态的变化，同时调度的结果 TaskInprogress 本身也是一个对象类型的变量，因此调度的结果 TaskInProgress 对象的引用编号和相关对象的编号采用和 MonitorNode 节点相同的方法，传给 MonitorNode 节点。

#### 3.2.3 增量更新动态同步机制

由于在网络中传输数据，只能传输简单类型的数据，因此不能直接采用对象序列化的技术，因为可以序列化的对象必须实现 Writeble 接口，而包含资源管理的信息的对象很多是 Java 已封装好的对象，并没有实现 Writeble 接口。其次，直接序列化只能采用浅表拷贝的方式，而监控任务和作业执行的对象（TaskInprogress 和 JobInProgress 对象）自身又包含对其它对象的引用，直接采用对戏拷贝的方式会使工作量过大。因此，本文提出一种增量更新同步算法，即将对象本身的引用和对象所包含的子对象引用构造成一棵树，树中每个节点的结构图 3 所示，每个节点包含对象本身的引用、对象的所有子对象的引用数组、父节点的引用和对象在父对象子对象引用数组中的相对位置，当对象的状态发生变化时，将构造一个 String[] 类型的数组，用于对 ResManage 节点上相同的对象进行更新。例如当 JobInProgress 的第一个 TaskInProgress 对象的部分简单类型的属性发生变化时，将 TaskInProgress 映射成编号 11，将 TaskInProgress 对象所关联的简单类型的属性（如 numMaps）构造成一个静态内部类，并将这些属性的最新值赋给这个静态类中的成员，然后发送给 ResManage 节点，ResManage 节点根据接收到的信息，直接对 ResManage 节点上的第一个 TaskInprogress 对象的相关属性进行赋值。当 completedTask 方法被调用时，会执行 activeTasks. remove(taskId) 操作，此时将 taskId 映射成的编码长度、编码、操作码发送给 ResManage 节点。由于 ResManage节点与 MonitorNode 节点具有相同的对象编码，因此根

据这个编码可以快速获得在 ResManage 节点上相同的 taskId 对象的引用，ResManage 节点只需要根据对发送的数据进行解析，就可以完成对象状态的更新，不需要再重复执行 Monitor Node 其他的计算过程，有效减少了接收方的计算量和发送方序列化对象的工作量。在 Monitor Node 和 ResManage 节点同步的过程中，作业监控对象（例如 JobInProgress）及 Monitor Node（原 JobTracker）本身在初始化的时候即可在树中建立，对于特定的任务对象（例如 TaskInProgress），可能存在多个任务对象管理同一个 Task 的情形，对象生成后可以再添加到树中，以减少建树操作对系统资源的开销。每次心跳消息返回后，Monitor Node 都会更新 TaskTracker 和 Task 的状态，对于普通的状态更新，只需要传送封装好的简单变量即可。特定状态（如任务失败）则需要在 Monitor Node 和 ResManage 节点两端实时同步相关对象和变量的信息，包括简单变量的值更新、对象的操作和在树中的映射关系的改变。

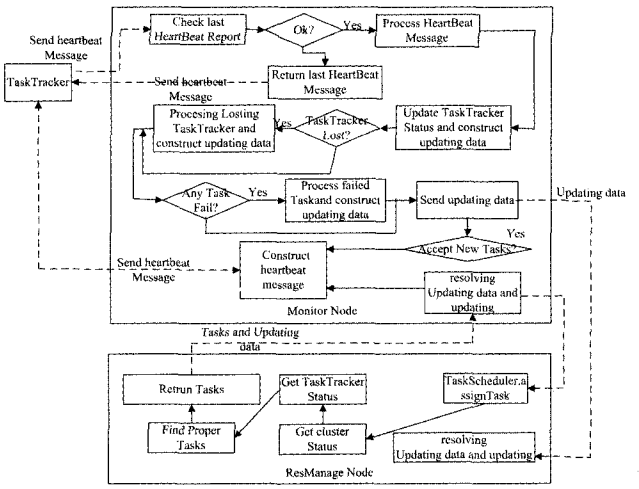


图 3　Hadoop 新任务监控与资源管理机制

图 4 为增量更新动态同步机制节点结构；图 5 为增量更新动态同步机制示例；图 6 为对象编码构造算法。

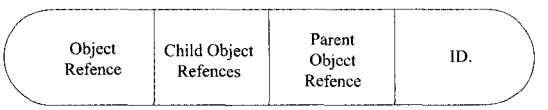


图 4　增量更新动态同步机制节点结构

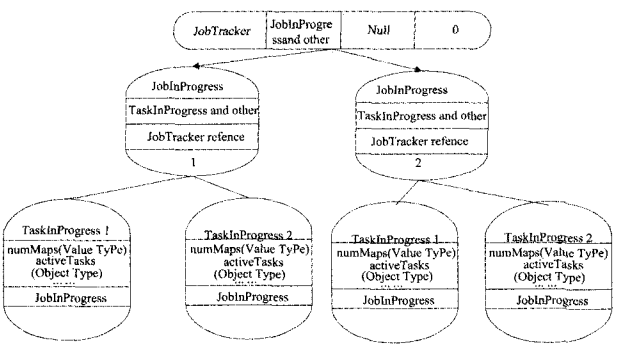


图 5　增量更新动态同步机制示例

**算法 1**　动态增量更新算法 Delta updating algorithm between Monitor Node and ResManage Node(Object instance or var type data)

输入：Object instance or var type data

输出：String type array

1. Do While Object's name == "JobTracker"
2. Counter=Counter+1
3. Contact strOfObject[1] with the Object. ID
4. Get parent node of this node
5. EndWhile
6. StrOfObject[2]=Counter

图 6　对象编码构造算法

## 4　实验结果与分析

实验在单机上通过 cloudsim 进行模拟验证，cloudsim 平台是由澳大利亚墨尔本大学开发的云计算仿真软件，该软件基于爱丁堡大学的 SimJava 离散事件模拟包开发，扩展了 SimJava 包的功能并且解决了在单机上由于模拟大规模集群引起的模拟效率下降问题，提供了一个高效、灵活的云计算仿真框架。实验的硬件平台为：AMD AthlonX4 645@3.1GHZ 处理器，4GB DDR3 内存，软件平台为 WindowsXP 32bit、JDK1.6 和 cloudsim 2.0。

为了评估新的资源管理与任务监控方案对降低负载的作用，试验中采用了 3 组不同的数据进行对比试验。由于本次实验的重点在于评估采用新方案后 JobTracker 节点负载高低，因此假定实验中 Slave 节点的细节（如传输数据的延迟、网络延迟等）不是本实验考虑的重点。实验中假定每个 Slave 节点的配置一样，每次运行的 Job 由 Map Task 构成。每个节点配有 1 个 Map Slot。对于 Master 节点而言，采用 Hadoop 默认 FIFO 调度策略。实验中假定没有 Slave 节点失效的问题，同时不考虑 Hadoop 的推测执行功能。实验数据分为 3 组，分别代表不同特征的集群和作业。通过扩展 CloudSim 平台的 DataCenter、DataCenterBroker 和 Host 类，模拟 Slave 节点和 Master 节点的行为。每隔 5 个时钟周期，DataCenter 集中调用 Host 类的 updateVmsProcessing（double currentTime）类，把每个 Host 上任务的执行信息和状态信息通过模拟封装的心跳消息发送给 DataCenterBroker。DataCenterBroker 调用 heatbeat 方法处理心跳消息并分配任务给每个 Host，同时记录 DataCenterBroker 节点每个时刻执行的任务指令长度，放入 HashMap〈Double，Long〉workLoadStaticMap 中，当模拟完成后，输出结果并进行分析。Task 失败的概率假定是 1%

第一组实验假定每个 Slave 节点的处理能力为 1000MIPS，集群中有 30 个节点。作业的每个 Task 需要的基础指令长度为 5000MIPS，并且加入一个随机扰动因子增加或减少每个 Task 3000MIPS 的指令长度，作业共包含 200 个 Task。实验数据如图 7 所示。

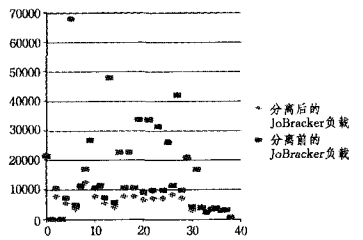


图 7　第一组实验数据

第二组实验假定每个 Slave 节点的处理能力为 1000MIPS，集群中有 80 个节点。作业的每个 Task 需要的基础指令长度为 8000MIPS，并且加入一个随机扰动因子最大增加或减少每个 Task 2000MIPS 的指令长度，作业共包含

500 个 Task。实验数据如图 8 所示。

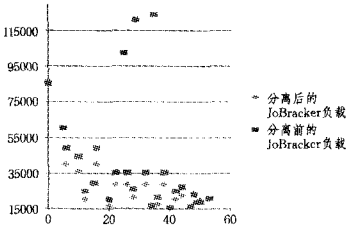


图 8　第二组实验数据

第三组实验假定每个 Slave 节点的处理能力为 1000MIPS，集群中有 100 个节点。作业的每个 Task 需要的基础指令长度为 20000MIPS，并且加入一个随机扰动因子最大增加或减少每个 Task 4000MIPS 的指令长度，作业共包含 120 个 Task。实验数据如图 9 所示。

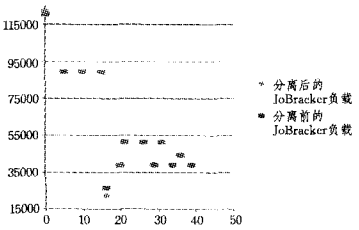


图 9　第三组实验数据

对比每组实验数据可以发现，在不进行资源调度的时候，新方案和现有方案中 Monito Node 的负载变化明显，新方案的负载还会略高，这是由于新方案中进行了信息同步操作。但是当 Master（新方案中的 Monitor Node，现有方案中的 JobTracker）节点对收集的心跳消息进行任务调度时，两种方案有着显著的差异。当调度算法在 Master 节点上执行时，负载较高，特别是当大量 TaskTracker 同时向 JobTracker 请求新任务时，原方案因为要执行调度算法，负载明显高于新方案中 Master 节点的负载。在新的方案下，Master 节点的负载主要是由资源同步操作导致的，由于新方案中的同步信息采用 String 类型字符串的编码方式，且编码的操作和树的深度呈正比关系，因此编码操作的花销低于调度算法的资源花销。通过对象序列化写入和套接字传输，把需要同步的信息和 Slave 节点的请求传送到 ResManage 节点进行处理，ResManage 节点根据 Master 节点发送来的指令同步信息，完成调度后将调度的结果返回给 Master 节点并同步 Master 节点上新 Task 的状态。

## 5　评价与比较分析

目前，与本方案总体思路上最为接近的工作是 hadoop 0.23中的 YARN 方案，该方案最基本的设计思想是将 JobTracker 的两个主要功能，即资源管理和作业调度/监控分成两个独立的进程[12]，如图 10 所示。

在该解决方案中包含两个组件：全局的 Resource Manager（RM）和与每个应用相关的 ApplicationMaster（AM）。这里的“Application”既可以代表一个传统的 MapReduce 作业，也可以是 DAG 作业。RM 的功能和当前的 Hadoop 架构中的 JobTracker 组件的功能相似，扮演资源管理与分配的功能，但是任务监控与调度的功能将交给 AM 组件执行。同时，当前 Hadoop 平台中的 TaskTracker 概念在 YARN 架构中不复存在，取而代之的是 NodeManager（NM）组件。NM 是在集群中的每台机器上运行的一个代理，负责载入 Application 所需的 container，container 的概念与当前 Hadoop 平台中的 slot 概念相似，代表集群中的资源，但是它不区分 Map Slot 和 Reduce Slot 的概念，同时 container 代表的资源数据具有更强的扩展性，既可以指可用内存的大小，还可以将网络、计算能力等信息加入其中，监控资源（内存、CPU、磁盘、网络等）的使用情况并将之汇报给调度器负责管理的 AM 组件，ApplicationsMaster（ASM）主要负责接收作业，并管理 AM 的生命周期。

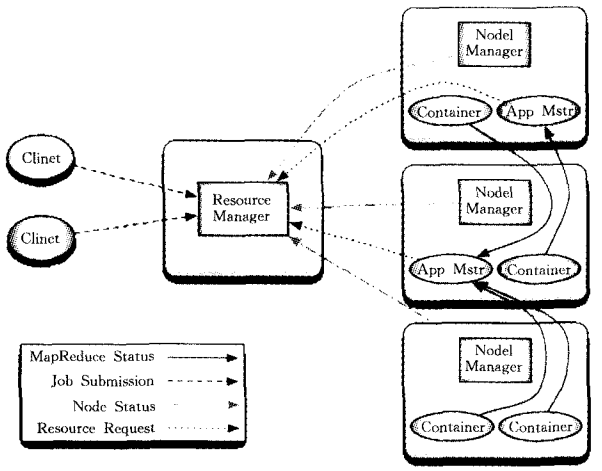


图 10　Hadoop YARN 架构图[12]

本方案与 YARN 架构最大的不同包含以下几方面：

1. JobTracker 功能的分离方式不同。在 YARN 方案中，JobTracker 的功能被分成 RM 和 AM 2 个独立的进程，使 RM 专注于资源管理，降低了当前 Hadoop 版本中 JobTracker 的复杂度，减轻了 JobTracker 的负担，使得 JobTracker 不再成为制约集群的性能瓶颈，而任务监控与调度的功能组件 AM 被分配到每个 Node 上执行，并通过 ASM 进行管理。对于每个应用，都创建一个 AM 用于监控和调度。而本方案中，JobTracker 被分成 Monitor Node 和 ResManage 节点，Monitor Node 执行原 JobTracker 中的任务监控与调度的功能，而新加入的 ResManage 节点则负责资源管理的功能，降低原 JobTracker 节点的工作负担，并且集群中所有的 Task 的监控与调度功能都交给 Monitor Node 管理，而不是为每个不同的 Job 都创建一个 Monitor Node。由于每个 AM 在 slave 节点上运行，而 slave 节点的可靠程度明显低于 Master 节点，因此本方案中的 Monitor Node 和 ResManage 节点更加稳定可靠。YARN 的分离方案变化更大，需要用户参与的操作也更多。例如 AM 的创建需要用户指定请求的内存资源，和现有方案相比，用户在使用 JobConf 进行配置时，考虑个因素更少。

2. 资源表示的方法不同。在 YARN 方案中，资源不再区分 Map 资源和 Reduce 资源，统一采用 Container 的概念进行表示，与 Slot 表示资源的方式相比，YARN 的资源表示方案更加合理，更容易实现精确的调度。而本方案依然延续当前 Hadoop 的资源表示方式。YARN 方案的管理方式显然比当前 Hadoop 的资源表示方式更加合理。

**结束语**　云平台的任务监控与资源管理机制是云平台的核心功能之一。当前 Hadoop 云平台中的任务监控与资源管理机制由于过于集中，导致 JobTracker 的负载过大，使得 JobTracker 成为 Hadoop 云平台中的性能瓶颈。本文通过一

种新式的增量更新算法，在 Monitor Node 节点和 ResManage Node 节点之间动态同步资源管理所需的数据，将原 JobTracker 节点的资源管理功能分配到 ResManage Node 节点上执行，使得 JobTracker 节点的负载有效降低。实验分析表明，在典型的 Hadoop 系统中，当 JobTracker 执行调度操作时，本文提出的方案可降低 18%～26%的负载，提高了 Hadoop 云平台的工作效率。

## 参 考 文 献


[1] Apache Hadoop [EB/ OL]. http://hadoop. apache. org/ 2012 March

[2] 李建江，崔健. MapReduce 并行编程模型研究综述[J]. 电子学报，2011(11)：2636-2641

[3] Apache Hadoop [EB/ OL] . http://zh. wikipedia. org/zh-cn/Hadoop

[4] 孙凝晖. 发展开放平台技术 [EB/ OL]. http://it. chinabyte. com/357/11520857. shtml. 2010

[5] The Next Generation of Apache Hadoop MapReduce[EB/OL]. http://developer. yahoo. com/blogs/hadoop/posts/2011/02/mapreduce-nextgen/

[6] Kc K, Anyanwu K. Scheduling hadoop jobs to meet deadlines [C]// 2nd IEEE International Conference on Cloud Computing Technology and Science (CloudCom). 2010：388-392

[7] Sandholm T, Lai K. MapReduce optimization using regulated dynamic prioritization [J]. Performance Evaluation Review, 2009, 37(1)：299-310

[8] Job Scheduling for Multi-User MapReduce Clusters[EB/ OL]. http://www. eecs. berkeley. edu/Pubs/TechRpts/2009/EECS-2009-55. html. 2009

[9] Polo J, Carrera D, et al. Performance-Driven Task Co-Scheduling for MapReduce Environments[C]// IEEE Network Operations and Management Symposium (NOMS) Conference. 2010：373-380

[10] Polo J, Castillo C, et al. Resource-aware Adaptive Scheduling for MapReduce Clusters[J]. Lecture Notes in Computer Science, 2011, 7049：187-207

[11] Wolf J, Rajan D, et al. FLEX：A Slot Allocation Scheduling Optimizer for MapReduce Workload[J]. Lecture Notes in Computer Science, 2010, 6452：1-20

[12] Apache Hadoop NextGen MapReduce (YARN) [EB/OL]. http://hadoop. apache. org/common/docs/r0. 23. 0/hadoop-yarn/hadoop-yarn-site/YARN. html 2011 November


(上接第 90 页)

好，这对于网络负载较轻的 WMBAN 来说是极为有利的。图 6 为 ASCEMAC 和 DQBAN-MAC 与 FL-MAC 在时延方面的仿真对比。

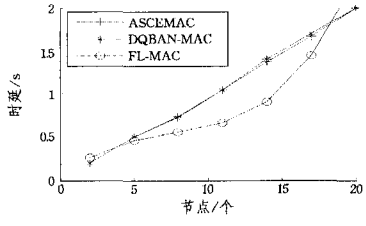


图 6 时延比较

由图 6 可知，FL-MAC 仅在网络负载较轻的情况下($5<n<16$)才具有明显的优势，当发送队列中的节点数 $m$ 与网络总节点数 $n$ 相差较小($n<4$)或较大($n>16$)时，实时性反而降低，这是因为 FL-MAC 算法引入了额外的保证时隙 GTS，增加了节点的时延。

**结束语** 本文分析了已有的 MAC 协议在节能方面的贡献，并指出了它们在无线医疗体域网应用中的局限性，以此提出了一种基于模糊控制理论的全新 MAC 协议。该协议通过模糊逻辑算法对节点所采集的生理信息进行了选择性的处理，在一定程度上减少了节点射频装置的工作次数，有效降低了节点的能量耗费和网络的时延，提高了整个网络的服务质量。理论分析和仿真实验表明，相比于传统的基于 TDMA 的模糊逻辑 MAC 协议，FL-MAC 在能耗和实时性方面都具有很大的优势，特别适用于网络负载较轻的无线医疗体域网。

## 参 考 文 献


[1] 王博，陈伟，王磊. 躯体传感器网络的高能效 MAC 协议设计[J]. 武汉理工大学学报，2009，31(20)：130-132

[2] Ren Qing-chun, Liang Qi-lian. An Energy-Efficient MAC Protocol for Wireless Sensor Networks[C]// IEEE Global Telecommunications Conference. 2005：157-161

[3] Otal B, Alonso L, Verikoukis C. Highly Reliable Energy-Saving MAC for Wireless Body Sensor Networks in Healthcare Systems[J]. Selected Areas in Communications, 2009, 27(4)：553-565

[4] 柯欣，孙利民. 无线传感器网络的 MAC 协议研究[J]. 计算机科学，2004，31(9)：29-31

[5] Gopalan S A, Kim D H, Nah J W, et al. A survey on power-efficient MAC protocols for wireless body area networks[C]// IEEE International Conference on Broadband Network and Multimedia Technology. 2010：1230-1234

[6] Li Hua-ming, Tan Jin-dong. An Ultra-low-power Medium Access Control Protocol for Body Sensor Network[C]// 27th Annual International Conference of the Engineering in Medicine and Biology Society. 2005：2451-2454

[7] Liu Bin, Yan Zhi-sheng. CA-MAC：A Hybrid Context-aware MAC Protocol for Wireless Body Area Networks[C]// 13th International Conference on e-Health Networking Applications and Services. 2011：213-216

[8] Maman M, Ouvry L. BATMAC：An Adaptive TDMA MAC for Body Area Networks Performed With a Space-Time Dependent Channel Model[C]// 5th International Symposium on Medical Information & Communication. 2011：1-5

[9] Fang Geng-fa, Dutkiewicz E. BodyMAC：Energy Efficient TDMA-based MAC Protocol for Wireless Body Area Networks[C]// 9th International Symposium on Communications and Information Technology. 2009：1455-1459

[10] Timmons N F, Scanlon W G. An Adaptive Energy Efficient MAC Protocol for the Medical Body Area Network[C]// 1st International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology. 2009：587-593

[11] Ullah N, Khan P, Kwak K S. A Very Low Power MAC (VLPM) Protocol for Wireless Body Area Networks[J]. Sensors, 2011, 11(4)：3717-3737