Hadoop 下基于统计最优的资源调度算法

邓传华, 范通让, 高 峰
(石家庄铁道大学 信息科学与技术学院, 石家庄 050043)

摘 要: 云计算集群中的资源存在异构和节点稳定性问题。异构资源的计算能力不同会导致较突出的作业任务同步问题,而某个节点的不稳定状态会使运行于该节点的任务大量备份或重新计算。针对上述两问题将严重影响集群作业的执行进度,在 Hadoop 平台下利用统计方法,提出一种资源调度算法,对计算资源较少的节点和不稳定状态的节点进行标志并降权,让集群尽可能调度资源较好的稳定节点。实验结果表明,该算法能够在一定程度上减少作业的周转时间,提高集群的效率和吞吐量。

关键词: 云计算; 资源调度; Hadoop; MapReduce
中图分类号: TP393 **文献标志码**: A **文章编号**: 1001-3695(2013)02-0417-03
doi:10.3969/j.issn.1001-3695.2013.02.027

Resource scheduler algorithm based on statistical optimization under Hadoop

DENG Chuan-hua, FAN Tong-rang, GAO Feng
(*School of Information Science & Technology, Shijiazhuang Tiedao University, Shijiazhuang* 050043, *China*)

Abstract: There are node stability problems in a heterogeneous cloud clusters. The heterogeneous resources' differences in computing ability will lead to prominent sync issues of job' task, and a unsteady node will make the task which is running in the node been backup or been recount. The two problems above will seriously affect the execution of the cluster's jobs progress. According to the situation, now using the Hadoop workbench with statistical method, this paper put forward a kind of resource scheduling algorithm which marked and dropped the less computing resources node and instability node rightly , letting cluster scheduling resources better stability node as much as possible. The experimental results show that, the algorithm can reduce the jobs cycle time in a certain degree, and improve the efficiency of the cluster and throughput.

Key words: cloud computing; resource scheduling; Hadoop; MapReduce

作为并行计算、分布式计算、网格计算的一种商业实现,云计算通过网络将多个计算实体整合成一个相对强大的集群,具有超大规模、高可靠性和高可扩展性等特点。其核心思想是通过资源的统一调度和分配构成一个资源池,按需分配给用户使用。在众多的云计算解决方案中,Google 提出的基于分布式文件系统 GFS[1] 建立廉价集群实现 MapReduce[2] 编程思想的方案因其应用简单、高效的特点得到了广泛的支持,解决了许多大规模数据的计算问题。而 Hadoop 作为 Google 相关思想的一个开源实现,因其应用简单、高效、开源的特征,亦受到了商业公司和学术界的极大关注。Yahoo! 建立了目前最大的 Hadoop 集群为其新闻推荐系统提供支撑和用来处理日志等工作。Facebook 用其来处理 terabytes 级别的用户数据和其个性化广告推荐系统,最近也把它应用到了 Facebook messages 上,用来处理聊天、电子邮件和 SMS 等[3]。UC Berkeley AMP Lab 的研究人员通过在 Yahoo! 和 Facebook 上的数据分析提出了 Fair[4] 和 Delay[5] 等算法;国内的中国科学院计算所和清华大学等亦开展了相关工作的研究,提出了负载预测机制模型 MR-Predict[6] 等研究成果。

随着 Hadoop 的越来越流行和集群规模越大,资源调度也变得越来越重要。在一个 MapReduce 集群中,作业分成 Map 和 Reduce 阶段,每个阶段又被分割成若干个任务再分配到不同的节点上执行。伴随着集群规模的增大,集群中的各个节点可能不同构,集群规模越大,这种异构现象也越明显,一个作业同一阶段的不同任务在不同计算资源和存储资源的节点上执行时,其执行效率会有很大的差别。同时,由于集群规模增大,出现不稳定节点的情况也大大增加,因此选择一个合适的节点分配任务非常重要。

1 Hadoop 相关技术及存在的问题

Hadoop 是一个分布式集群架构,由 Apache 基金会开发,其核心是 Google 的 MapReduce 编程思想、HDFS 分布式文件集群和 Hbase 分布式数据库,分别对应 Google 论文发表的三大核心组件中的 GFS、MapReduce 和 BigTable[7]。Hadoop 旨在方便地以廉价的 Linux 集群组成数据中心运行各种应用。

1.1 Hadoop 中作业执行的流程

和其他的 MapReduce 实现一样,Hadoop 集群也是用从节点(slave)来执行其 Map 和 Reduce 任务的。一个作业提交给 Hadoop,MapReduce 作业的执行需要涉及四个独立的实体,即客户端(client)、JobTracker、TaskTracker 和 HDFS。其执行步骤如下:a)提交作业,Client 将程序代码、接口和输入输出路径等配置完毕后将作业提交给 JobTracker,并将相关资源复制到 HDFS 上;b)初始化作业,检查作业相关路径,让 JobTracker 从 HDFS 和 JobClient 中获取相关信息,计算作业的输入划分,创

收稿日期: 2012-06-18; **修回日期**: 2012-07-28

作者简介: 邓传华(1987-),男(瑶族),湖南汝城人,硕士研究生,主要研究方向为云计算、电子政务(icefox@outlook.com);范通让(1965-),女,教授,博士,主要研究方向为网络技术与网络安全、信息处理、网络与教育;高峰(1982-),男,博士,主要研究方向为计算机网络.

建并初始化 Map 和 Reduce 任务;c)分配任务,当 JobTracker 收到 TaskTracker 的心跳信息并发现 TaskTracker 在请求一个任务时,JobTracker 会将任务分配给它;d)执行任务,TaskTracker 申请到新的任务后将任务本地化,并启动 JVM 运行任务;e)更新任务执行进度和状态,TaskTracker 每隔 5 s 在发送给 JobTracker 的心跳中封装报告自己的任务执行状态;f)JobTracker 汇总 Tasktracker 任务的执行进度信息,当收到最后一个任务已完成的通知后,设置任务状态为“成功”,并报告给 JobClient。整个过程如图 1 所示[8,9]。

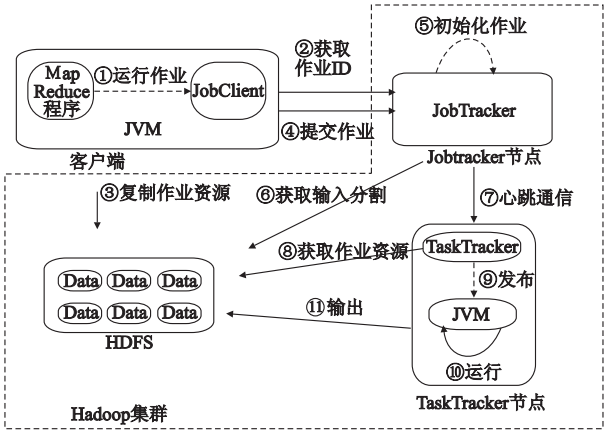

图1 Hadoop作业执行过程

1.2 异构集群存在的问题

随着集群新老硬件的更替和集群规模的增大,集群将不可避免地遇到内部异构的问题。比如各个节点的 CPU、内存等不尽相同,或者由于增删硬件导致的网络拓扑异构等。然而,Hadoop通过配置每个节点的任务槽个数来管理资源,并不对任务槽的具体资源情况和网络位置进行区分[10]。当 JobTracker 需要分配任务时,它会将任务分配给第一个请求任务心跳到达的任务槽。这种分配方式很简单,但是会带来一系列的问题。

首先是执行任务的节点和任务数据的存储节点的网络跳数太多而导致的效率以及大量的网络流量和传输延时问题。由于 HDFS 的分布式存储,当任务运算时可能需将大量的数据从不同的存储节点通过网络传输到运算节点上,这就会造成运算节点等待数据和网络拥堵,给性能造成消极影响。对于该问题,Hadoop 通过在分配任务时考虑选取将一个距离任务执行节点较近的输入划分文件这一策略对于缓和这个问题取得了一定的效果;同时,学者们提出了很多相关策略解决这一问题。Zaharia 等人[5]提出了 Delay 来解耦严格的作业顺序,在一定程度上缓解了位置问题;He 等人[11]对通过本地任务优先贪婪策略对这一问题提出了解决方案,Zhang 等人[12]通过 next-k-node 策略提出了见解。

其次是同一作业任务的进度不同步问题。进度不同步有很多的原因,如网络延迟、没有足够的任务槽等,这里主要考虑异构集群的计算资源问题。当不同类型的作业任务在同构的任务槽中执行时,执行时间相差不会太大;而在异构的任务槽中,由于任务执行能获取的资源量不同,执行进度就可能相差非常大,这将会导致大量的备份任务,将不同的任务备份执行时会导致大量的计算资源消耗和大量的网络流程传输,严重影响集群性能。学者们也提出了部分方案进行解决。Deadline Scheduler 根据作业的运行进度和剩余时间动态调整作业获得的资源量,以便作业尽可能地在截止时间内完成这一策略来针对该问题[13];而 LATE[10]通过预备份某个作业里面进度最慢的任务这一机制来从另一个侧面解决这个问题[6]。

上述两个问题在使用前面提到的算法过程中,由于没有保存相关调度历史数据,下一个作业的执行过程中,此调整调度过程重复执行,带来了较大的系统开销,影响系统效率。

1.3 节点故障问题

节点故障是任何一个集群都无法避免的问题。由于集群规模增大,因此某一个时刻有不稳定的节点或故障节点的情况大大增加,Yahoo! 的研究人员发现,在 1 000 个长期工作的节点的集群中,每天大概有 2% 到 3% 个节点发生故障,新节点的故障率更是一般节点故障率的 3 倍[14]。Hadoop 中 JobTracker 是唯一的,因此只能通过创建备用节点,在失败后采用领导选举算法等方法来确定新的 JobTracker 节点,这里不作详细描述。TaskTracker 错误比较常见,通常的做法是 JobTracker 要求其他的 TaskTracker 节点重新执行原本故障节点的任务。研究发现,重计算中超过一半发生在 5% 的机器上[15],但是上述简单的重新执行算法并没有考虑这一点,在下一个任务的分配过程中,由于故障节点心跳可能依然正常发送给 JobTracker,而 JobTracker 并不会规避这些存在问题的节点,这将会导致集群重计算概率大大提高;JobTracker 必须进行重复调度,导致作业任务执行效率低下,同时重计算会对任务同步产生影响,这些操作都会增大网络数据传输,影响集群性能。

2 RSSO 算法设计

为了尽量减少异构集群中性能较低的节点和易错节点带来的负面影响,本文提出了一种基于统计分析的 Hadoop 任务槽分配机制算法 RSSO(resource scheduler algorithm based on statistical optimization)。通过对相关参数的配置,可以灵活地根据实际集群调整参数来达到较好的效果,从而减少整个作业中的备份任务和重新计算任务次数,达到提高集群效率的目的。

2.1 统计分析和更新槽位统计信息算法

对于一个集群,可能有 m 个 Map 槽和 n 个 Reduce 槽,分别标记为(M_1,M_2,…,M_m)和(R_1,R_2,…,R_n)。而每个作业可能需求 i 个 Map 槽和 j 个 Reduce 槽。

2.1.1 统计初始化和监控

在集群初始化时,对所有的 Map、Reduce 槽的权重标志为 100。当有槽位因各种原因从集群中删除的同时,将其权重信息从集群中删除;而有新的槽位加入集群时,标记为 100。将权重标志为 100 主要是给一个初始权重,防止因为作业对不同资源需求而导致的过早将某个槽位标记为次级槽位。对删除或者加入集群的槽位进行监控是为了反映集群的动态性。

2.1.2 权重更新

当一个作业执行完毕的同时更新统计信息,具体如下:

a)获取所有任务槽的执行情况。在 TaskTracker 向 JobTracker 心跳通信的同时会将作业执行情况返回,如果任务成功执行则获取该任务执行时间;如果任务重计算或者被备份,则标记为 -1。

b)根据 a)获取的情况按以下情况更新权重:如果返回为 -1,或者其完成时间在所有的槽位中是最慢的 20%(这里的总数包括了没有成功执行的槽位),当其权重小于或等于 100 时则对其权重减 1(权重不能为负),当其权重大于 100 时,对其大于 100 的部分减半处理(下取整);如果完成时间是中间的 20% ~70%,权重不变;如果是前 30%,权重加 1。上述权重

变化根据 Map 和 Reduce 的不同保存到集群里面。该操作的 Map 阶段伪代码如下所示,其 Reduce 阶段操作类似,这里不再赘述。

算法 1　权重更新算法

```
Sort Map[1…i] by ExcuteTime
x←1
for 1 to i
  if x < i/5
    if w[x] < =100
      w[x]←w[x] - 1
    else then
      w[x]←100 + Int( (w[x] - 100)/2 )
    end if
else x > 7i/10 then
    w[x]←w[x] + 1
end if
x←x + 1
end for
```

2.1.3　重置操作

重置操作触发情况为满足下列条件之一,即时间间隔 T 到来或者有某一个权重大于等于 200。重置操作对权重小于 100 的槽位重置为 100;对于权重高于 100 的槽位,将其高于 100 的部分减半取整。这样做是为了给权重较低但可能已经恢复了计算能力的机器一个公平的调度机会,同时又能够平衡负载,防止任务集中在某几个节点上。时间 T 一般根据实际情况设置为 24 ~ 48 h 比较合理。综上所述,对 Map 槽重置算法如下所示,Reduce 槽的重置算法类似,不再赘述。

算法 2　权重重置算法

```
x = 1
for 1 to m
  if w[x] < = 100
    W[x] = 100
  else then
    W[x] ← Int( (w[x] - 100)/2 ) + 100
  end if
  x ← x + 1
end for
```

2.2　分配任务槽方法

当一个作业请求任务槽时,若当前集群没有空闲任务槽,当某个任务释放任务槽则直接分配给它;如果当前有多个空闲的任务槽,则按以下方法进行选择:

假设一般集群的任务槽选择策略(如优先级、到达时间、位置等)对选择任务槽的影响权重为 B,本文提到的权重对策略选择的影响为 W,则此集群的影响权重总和为($\varepsilon B + \zeta W$)。其中 B 和 W 根据调度算法的反馈取值均映射为 0 ~ 1 之间,ε 和 ζ 为可变的调整因子,其和为 1。集群管理员可根据集群情况和需求作出调整,以适应不同情况的需求。一般情况下,ε 为 0.30 ~ 0.45 之间较为合适。在同构的集群或者异地集群中,ε 可设置为 0.70 以上,在节点差异非常大的情况下,ε 可以取值 0.3 以下。集群将综合权重选择最佳的任务槽分配给任务。

3　验证与结果分析

为了体现异构性,在搭建环境时,本文采取了如下方案,具体配置如表 1 所示。

表 1　配置列表

用途	配置	数量
主节点	8 GBram,80 GB 硬盘,4 核 Intel Xeon	1
节点类型 1	4 GBram 80 GB 硬盘,2 核 Intel Xeon	15
节点类型 2	2 GBram 80 GB 硬盘,1 核 Intel Xeon	25
节点类型 3	1 GBram 80 GB 硬盘,1 核 Intel Xeon	5

同时,为了模拟不稳定的环境,将四个典型作业 word count、table join、group by、grep 分别运行 150 次,并随机对其中三个节点稳定性进行了干扰,分别统计取平均值获取了的结果如图 2 所示。

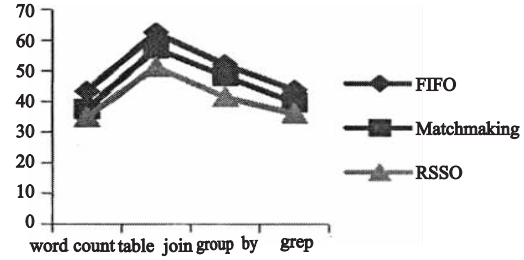

图2　word count、table join、group by、grep平均执行时间

由于该算法的统计信息都是在 Hadoop 集群默认的心跳通信中获取数据,并仅在某个作业完成时进行统计和更新权重,因此对网络负载影响极微,仅对 JobTracker 的运算量略有提升,约占 JobTracker 运算量的 0.03% ~ 0.35%。通过图 2 的对比发现,RSSO 在四个典型作业中速度均比单纯的 FIFO 要高出 15% 左右,比对位置进行优化了的 Matchmaking 高 3% ~ 8%。从实验结果上来分析,在异构集群中,RSSO 由于综合考虑了资源位置分布和节点计算能力,提升了系统的效率和吞吐量;同时,由于在 FIFO 的基础上尽量规避了不稳定的节点,对集群的性能和稳定性亦有较好的提升。

4　结束语

本文利用统计分析方法,对 Hadoop 集群的异构问题和不稳定问题提出了一个新的算法,选取较优节点和规避重计算较多的不稳定节点,实验证明,能够在一定程度上提高集群的性能。下一步将继续对 Hadoop 异构环境下的资源调度算法进行改进。

参考文献:

[1] GHEMAWAT S, GOBIOFF H, LEUNG S T. The google file system [C]//Proc of the 19th ACM Symposium on Operating Systems Principles. New York: ACM Press,2003: 29-43.

[2] DEAN J, GHEMAWAT S. MapReduce: simplified data processing on large clusters[C]//Proc of the 6th Conference on Symposium on Opearting Systems Design & Implementation. Berkeley, CA: USENIX Association, 2004:137-150.

[3] BORTHAKUR D, GRAY J, SARMA J S, *et al*. Apache hadoop goes realtime at facebook[C]//Proc of International Conference on Management of Data. New York:ACM Press, 2011:1071-1080.

[4] ZAHARIA M, BORTHAKUR D, SARMA J S, *et al*. Job scheduling for multi-user mapreduce clusters[R]. Berkeley: EECS Department, University of California,2009.

[5] ZAHARIA M, BORTHAKUR D, SARMA J S, *et al*. Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling[C]//Proc of the 5th European Conference on Computer Systems. New York: ACM Press,2010:265-278.

[6] TIAN Chao, ZHOU Hao-jie, HE Yong-giang, *et al*. A dynamic mapreduce scheduler for heterogeneous workloads[C]//Proc of the 8th International Conference on Grid and Cooperative Computing. Washington DC: IEEE Computer Society, 2009:218-224.

(下转第 422 页)

例 1

$$u=\frac{1}{\sqrt{2}}\delta_0+\frac{1}{\sqrt{2}}\delta_3,\ v=\frac{1}{\sqrt{2}}\delta_0-\frac{1}{\sqrt{2}}\delta_3$$

容易验证它们的系统矩阵为酉矩阵。另设

$$\phi(x)=\begin{cases}\frac{1}{3}, 0\leqslant x<3\\ 0, x<0 \text{ 或 } x\geqslant 3\end{cases}$$

则有细分方程 $\varphi(x)=\varphi(2x)+\varphi(2x-3)$。但是可以验证 $\{\varphi(x-k):k\in Z\}$ 不是正交的，所以 u 并不是函数空间中某正交小波的低通滤波器。

下面选取一段语音信号(图 3(a))，加入强度为 0.05 的噪声(图 3(b))，然后对其作三层小波分解后进行阈值消噪。图 3 给出了基于不同小波的消噪效果。图 3(c) ~ (e)和(f)分别是三层小波分解均选用 db3、db4、db5 和 db6 小波的消噪结果，(g)和(h)是三层小波分解分别选用 db3、db4、db5 和 db4、db5、db6 的消噪结果。

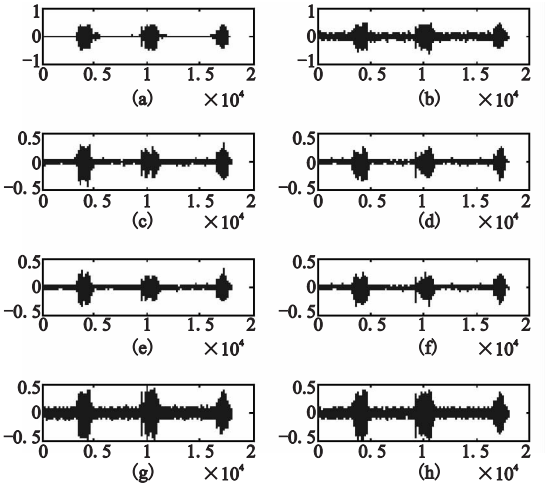

图3 基于不同小波的消噪结果

为了对比以上六种算法的去噪效果，选取信噪比(SNR)、均方误差(RMSE)和剩余噪声标准偏差(RNSD)作为去噪性能的对比指标。去噪后，SNR 越大，RMSE 和 RNSD 越小，则去噪信号就越接近原始信号，去噪效果和质量就越好。表 1 给出以上算法的指标对比，结果表明：各层使用不同小波去噪后 SNR 大，RMSE 和 RNSD 小，消噪效果优于各层均使用同一种小波。事实上，小波函数 dbn 中的 n 反映的就是小波函数的消失矩，消失矩越高，将使小波变换之后的高频小波系数越小，小波分解后的能量也就越集中。因此，通常总是选用消失矩较高的小波函数，但并不是消失矩的阶数越高越好，随着消失矩的增加，一个负面的影响是其支撑长度变大，使得对边缘信息和奇异点的定位不准确，且运算量增加。同时，不同的信号对消失矩的要求不一样，同一信号在不同分解层对消失矩的要求也不一样，目前还没有一个关于如何选取消失矩的准则。尽管如此，上述实验结果说明：在很多情况下，各层分解选择不同的小波基进行小波变换可以取得更好的处理效果，同时避免了额外的运算量。

表 1 指标对比

指标	图像 3					
	(c)	(d)	(e)	(f)	(g)	(h)
SNR	2.988 3	4.258 0	4.389 9	4.231 0	4.452 5	4.410 2
RMSE	0.049 1	0.042 4	0.041 7	0.042 9	0.041 4	0.041 6
RNSD	6.597 2	5.703 6	5.619 0	5.728 3	5.579 2	5.602 1

4 结束语

本文首先证明了离散空间中的周期小波和非周期小波变换同样可以按照 Mallat 算法的塔式结构实现，从而将离散小波分析与连续小波分析的实现联系起来；其次，通过对比分析说明离散小波变换在滤波器的选择上更加灵活；最后，数值算例表明，在实际应用中可以考虑在多层小波分解中选用不同的小波函数，在很多情况下可以取得更好的处理效果。本文结果还有需要改进的地方，即没有一个标准来确定什么样的信号适合在不同分解层上选择不同的小波。

参考文献：

[1] DAUBECHIES I. Ten lectures on wavelets[M]. Philadelphia: Society for Industrial and Applied Mathematics,1992.
[2] 张德丰. MATLAB 小波分析[M]. 北京:机械工业出版社,2012.
[3] 王大凯,彭进业. 小波分析及其在信号处理中的应用[M]. 北京:电子工业出版社,2006.
[4] MALLAT S. 信号处理的小波导引:稀疏方法[M]. 戴道清,杨力华,译. 北京:机械工业出版社,2012.
[5] 陈仲英,巫斌. 小波分析[M]. 北京:科学出版社,2007.
[6] FRAZIER M W. An introduction to wavelets through linear algebra [M]. New York: Springer, 1999.
[7] 彭宝瑜. 二维离散点集上的小波[D]. 西安:西安建筑科技大学,2010.
[8] 蒋英春,刘有明. 离散空间 $l^2(Z)$ 中正交小波系的完备性[J]. 数学学报,2006,49(5): 1075-1085.
[9] 蒋英春,刘勇. 离散正交小波的性质兼容性[J]. 北京工业大学学报,2006,32(4): 375-379.

(上接第 419 页)

[7] CHANG F, DEAN J, GHEMAWAT S, *et al*. Bigtable: a distributed storage system for structured data[C]//Proc of the 7th USENIX Symposium on Operating Systems Design and Implementation. New York: ACM Press,2006:205-218.
[8] HadoopMapReduce [EB/OL]. (2012-04-12). http://wiki.apache.org/hadoop/HadoopMapReduce.
[9] 陆嘉恒. Hadoop 实战[M]. 北京:机械工业出版社,2012.
[10] ZAHARIA M, KONWINSKI A, JOSEPH A D, *et al*. Improving mapReduce performance in heterogeneous environments[C]//Proc of the 8th USENIX Conference on Operating Systems Design and Implementation. Berkeley, CA: USENIX Association,2008:29-42.
[11] HE Chen, LU Ying, SWANSON D. Matchmaking: a new mapreduce scheduling technique[C]//Proc of the 3rd IEEE International Conference on Cloud Computing Technology and Science. Washington DC: IEEE Computer Society,2011:40-47.
[12] ZHANG Xiao-hong, ZHONG Zhi-yong, FENG Sheng-zhong, *et al*. Improving data locality of Mapeduce by scheduling in homogeneous computing environments [C]//Proc of the 9th IEEE International Symposium on Parallel and Distributed Processing with Applications. Washington DC: IEEE Computer Society,2011:120-126.
[13] POLO J, CARRERA D, BECERRA Y, *et al*. Performance-driven task co-scheduling for mapreduce environments[C]//Proc of the 12th IEEE/IFIP Network Operations and Management Symposium. Piscataway: IEEE Press,2010:373-380.
[14] RAO B T, SRIDEVEI N V, REDDY V K, *et al*. Performance issues of heterogeneous Hadoop clusters in cloud computing[J]. Global Journal Computer Science & Technology,2011,11(8):81-87.
[15] ANANTHANARYANAN G, KANDULA S, GREENBERG A, *et al*. Reining in the outliers in MapReduce clusters using mantri[C]//Proc of the 9th USENIX Conference on Operating Systems Design and implementation. Berkeley, CA: USENIX Association,2010: 1-16.