

云计算下 MapReduce 多组容错机制架构的分析与研究

张治斌, 李燕歌

(河南理工大学 计算机科学与技术学院, 河南 焦作 454000)

摘 要: 提出了 MapReduce 多组容错机制, 在传统的 Hadoop MapReduce 架构上进行改进, 即在同机柜中的 TaskTracker 节点之间增加了多组关系, 这样可以缩短发现失效节点的时间, 同时减轻 JobTracker 节点的负荷, 减低了带宽使用率, 减少网络拥塞. 通过实验证明, MapReduce 多组容错机制提高了 MapReduce 的工作效率.

关键词: 云计算; MapReduce 并行编程模型; 多组容错机制

中图分类号: TP302.8

文献标识码: A

文章编号: 1000-7180(2014)01-0052-04

Analysis and Study on under Cloud Computing Multiple Sets of Fault Tolerance Strategy Architecture of MapReduce

ZHANG Zhi-bin, LI Yan-ge

(College of Computer Science and Technology, Henan Polytechnic University, Jiaozuo 454000, China)

Abstract: This paper presents a Multiple Sets of Fault Tolerance Strategy of MapReduce, improve Hadoop MapReduce framework, between TaskTracker node in the same cabinet group relationship added multiple sets relations, it can short the time of founding the failed node, at the same time, reduce the load of JobTracker node, bandwidth utilization and network congestion. The experimental results show that Multiple Sets of Fault Tolerance Strategy of MapReduce improved the efficiency of MapReduce.

Key words: cloud computing; MapReduce parallel programming model; multiple sets of fault tolerance strategy

1 引言

随着信息化程度的日益剧增, 信息量成指数级增长, 各大商家需要从 PB 级甚至是 EB 级的海量数据中挖掘出有效信息, 伴随着这些问题云计算产生了, 但是云计算只是一种思维模式, 要想真正地解决这些问题就必须处理好支持云计算后台的云计算数据中心^[1]. Google 公司提出 MapReduce 编程模型简单易用、并行化程度高、系统易扩张等特点吸引了很多使用者, 但是在 MapReduce 并行编程模型^[2]集群系统中使用普通 PC 代替价格昂贵的服务器, 这样从很大程度上减低了运行成本, 然而人为操作错误或硬件故障等造成的机器失效现象时常发生, 并且随着集群规模的扩大, 这些问题会日益增加, 所以 MapReduce 的容错技术成为研究的热点^[3]. 传统

容错问题的解决首先是 JobTracker 节点需要 TaskTracker 节点周期性发送的心跳消息获得节点的工作状态, 其次对出现故障的节点任务进行再执行或者备份启动, 这一切的前提都是发现失效节点, 如何快速准确地发现失效节点是容错技术中需要解决的问题.

在文献[4]中提出的主动容错技术中虽然很好地预知了节点失效但是复杂的预知策略加重了 MapReduce 的运行负担; 在文献[5]中提出的自适应超时时间算法能缩短发现节点失效的时间, 但是由于估算作业运算时间不准确造成超期时间设置的不精确. 本文提出多组容错机制相比于其他设计在性能方面的优势有: 第一, 利用多组关系更快地发现失效节点, 因为 JobTracker 节点和 TaskTracker 节点之间心跳消息的传输时间肯定比同机柜中 Task-

收稿日期: 2013-02-18; 修回日期: 2013-04-27

基金项目: 河南省软件科学研究(102400450064)

Tracker 节点之间传输时间长,无论是长作业还是短作业都能提前发现失效节点;第二,减少网络拥塞,因为 JobTracker 和 TaskTracker 节点的通信必须通过交换机,是高带宽,而同机柜中的各个 TaskTracker 节点在局域网中进行通信是低带宽,在多组容错机制中通过低带宽来监测节点失效节约了带宽资源,从根本上减少网络拥塞;第三,节点失效的监测工作部分交给 JobTracker 来完成,减少了 JobTracker 节点的负载量,降低了因为控制节点任务繁重带来的性能瓶颈的可能性。

2 多组容错机制架构的分析

MapReduce 架构^[6]采用主从架构,由一个提供 Master 服务 JobTracker 节点和多个提供 Slave 服务 TaskTracker 节点组成。JobTracker 节点相当于 MapReduce 的神经中枢,负责对 TaskTracker 节点分配任务,同时还通过心跳消息监测 TaskTracker 节点的工作状态。TaskTracker 节点负责执行 map 任务和 reduce 任务。

2.1 需求分析

MapReduce 并行模型的多组容错机制在设计时,除了要继承集群的主从架构和原有系统的功能外,还要兼顾 TaskTracker 节点的扩展性、引入多组机制带来的系统管理复杂等问题,本文采用演进的方式,主要考虑以下需求:

(1)继承原有系统各个 TaskTracker 节点计算的独立性以及高度的并行处理能力;

(2)改变传统容错方案中再执行和备份的方法,从如何更快更准确的感知失效节点入手;

(3)解决传统容错技术中的 JobTracker 节点任务量繁重的问题;

(4)解决在同机柜中两个以上 TaskTracker 节点竞争失效节点未完成任务问题。

2.2 实现方案

(1)多组监测

与传统的 MapReduce 容错机制最大的不同点在于各个计算节点并不是孤立存在的,每个机柜中的 TaskTracker 节点存在多组关系,它们可以通过局域网发送询问消息来监测彼此工作状态。多组监测行为是在 TaskTracker 节点空闲时主动执行,例如当机柜 1 中的 TaskTracker a 节点完成了 JobTracker 分配的任务后,它会通过局域网向同机柜其他的 TaskTracker 节点发送消息询问工作状态,一旦发现某个节点失效,监测工作完成。

只有 TaskTracker 节点空闲时才主动执行多组监测的任务,不会影响工作节点计算的主要任务,既保证了数据计算的独立性,又把节点空闲的时间利用起来去分担 JobTracker 节点的任务,并且多组监测的过程是在局域网中进行的,不经过交换机,节约了带宽,降低了网络拥塞的可能。

(2)请求申请

请求申请在多组容错中起到承上启下的作用。TaskTracker 节点周期性向 JobTracker 节点发送心跳消息告诉自己的工作状态,空闲 TaskTracker 节点会把监测到本机柜中失效的 TaskTracker 节点信息也包含在给 JobTracker 节点发送的心跳消息中,JobTracker 节点收到消息后查找任务分配表,找出失效节点的任务是什么和任务执行到哪一步。因为对于失效节点正在执行 Map 任务和 Reduce 任务的处理方法是不同的,如果执行的是 map 工作,因为 map 的输出是在本地存储的,一旦失效则结果就无法得到,那么就同意请求者的申请,准备进行任务迁移;如果执行的是 reduce 工作,因为 reduce 的结果是全局量,节点在完成后失效并不影响最终结果,所以就发送消息给申请者已知晓该节点失效的消息,不需要进行任务迁移。

(3)任务迁移

任务迁移技术是多组容错机制中的关键技术,根据前两步的判断,对失效节点进行任务迁移,任务迁移技术可保证对用户透明。另外,JobTracker 向从失效节点获取中间结果的节点发送通知,远程获取结果的地址已经改变,并把新地址发送给它们。

以上三个步骤在 MapReduce 多组容错技术结构关系如图 1 所示,JobTracker 为客户端和工作节点的桥梁,在整个 MapReduce 工作中起到神经中枢的作用,它主要工作任务是为客户端完成请求和告知计算结果的位置信息,为分配任务并对其工作状态进行实时监测。TaskTracker 节点主要是完成 Map 工作和 Reduce 工作,是计算的核心,在空闲之余主动触发多组监测和响应任务节点批准的任务迁移。

3 MapReduce 多组关系

3.1 多组关系建立

在现实的网络环境中,若干个 TaskTracker 放在一个机柜里组成一个局域网,它们彼此之间通信不需要通过交换机,JobTracker 节点与各个机柜之间的通信都必须通过交换机。MapReduce 多组架构



图1 MapReduce 多组容错技术结构关系图

如图2所示。

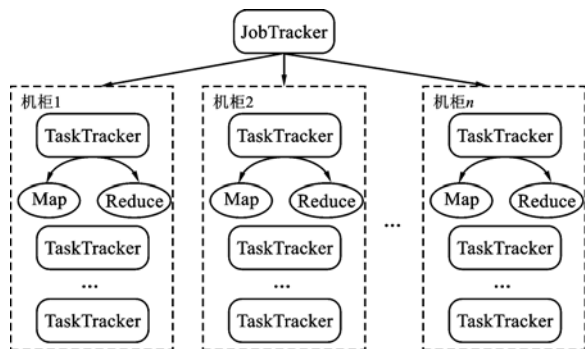


图2 MapReduce 多组架构图

在多组容错机制中最重要的是在同机柜的所有节点之间建立多组关系,已经完成任务的空闲节点担当机柜的管理者角色,利用多组关系向本机柜中的其他节点发送询问命令询问工作状态,多组关系只能在本机柜中有效,是不能跨越机柜的。在每个节点中都有存放记录本机柜其他节点工作状态的多组报告表,这个表中的每条记录都是节点在空闲时经过询问和应答消息方式一条一条添加的,而且这些记录是有期限的,每次 TaskTracker 节点向 JobTracker 节点申请过之后都会清空,这样以免重复的申请加重节点负荷。MapReduce 多组关系如图3所示。

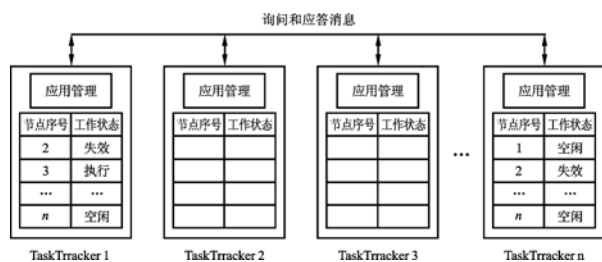


图3 MapReduce 多组关系图

3.2 多组关系带来的资源竞争

由于多组关系的存在,一个失效节点可能被同机柜中多个空闲节点监测到,也有可能多个节点失效被同一个空闲节点监测到,因此会导致了两种竞争资源的现象:多个空闲节点竞争获得一个失效节点的任务和多个失效节点的任务竞争一个空闲

节点。

第一种现象,解决方案是择优选择,首先从 JobTracker 节点那里得知失效节点没有完成的任务执行需要相关的数据块存放地址,根据数据本地性原则,如果该地址正好是申请任务迁移的空闲节点其中一个,那么 JobTracker 节点就同意那个节点进行任务迁移,拒绝其他节点的请求。如果没有一个节点符合,那么就对申请的所有空闲节点进行计算效率比较,节点的计算效率主要是由磁盘读取速率、cpu 执行效率以及网络带宽三个因素制约的,因此定义下面公式作为节点计算能力的值:

$$p_{\text{node}} = \alpha \times \frac{V_{\text{disk}}}{\max V_{\text{disk}}} + \beta \times \frac{V_{\text{cpu}} \times n}{\max V_{\text{cpu}}} + \lambda \times \frac{V_{\text{bandwidth}}}{\max V_{\text{bandwidth}}}$$

式中, V_{disk} , V_{cpu} , $V_{\text{bandwidth}}$ 分别指的是节点的磁盘读取速率,cpu 执行效率,网络带宽速率, $\max V_{\text{disk}}$, $\max V_{\text{cpu}}$, $\max V_{\text{bandwidth}}$ 分别指的是本机柜中最快的磁盘读取速率,最快的 cpu 执行效率,最快的网络带宽速率, α, β, λ 是三个不同的参数指标, $\alpha > 0, \beta > 0, \lambda > 0$, 且 $\alpha + \beta + \lambda = 1$, $V_{\text{disk}}, V_{\text{cpu}}, V_{\text{bandwidth}}$ 是动态变化的。

第二种现象解决办法是 JobTracker 节点从其他机柜中找出空闲节点,首先,优先选择数据本地性,任务需要的数据块存储在本节点上;其次,选择数据本机柜性,任务需要的数据块存储在同机柜的其他节点上;再次,以上两者都不满足的话,就根据现象一中节点计算效率比较择优选择。

4 实验与结果分析

本文提出的 MapReduce 多组容错技术缩短失效节点的发现时间,减轻 JobTracker 节点的任务量,减低了带宽使用率,从宏观上来看,最终都体现在提高了 MapReduce 的工作效率。

搭建实验平台,我们采用 Hadoop 开源平台,安装的是 Hadoop 0.20.1。实验配置:整个实验平台由 9 台 PC 机组成,NameNode 1 台,DataNode 8 台,具体如表 1 所示。

表1 实验平台搭建表

主机名称	IP	功能
Host1	218.198.241.181	JobTracker 节点
Host2	218.198.241.182	TaskTracker 节点
...	...	...
Host9	218.198.241.189	TaskTracker 节点

其中 Host2-5 放在机柜 1 中, Host6-9 放在机柜 2 中, 本实验采用 Ubuntu10.10 操作系统, 学校校园网, 使用 MyEclipse7.5 编程平台的实验环境。

使用 Hadoop GridMix^[7] 自带的代表性作业 javaSort 作为实验测试基准, javaSort 作业包括 10 个 Map 任务, 15 个 Reduce 任务个数, 作业大小为 45M。我们通过几组实验数据来分析:

(1) 在没有节点失效情况下, MapReduce 多组容错机制和传统容错运行原理相同, 在整个作业运行过程中没有节点失效的情况下, 运行 8 次分别得到的作业完成时间, 具体实验结果如图 4 所示, 得出所有节点都正常工作时作业运行完成的平均时间为 43 s。

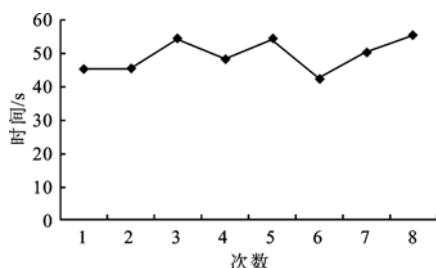


图4 没有节点失效情况下作业运行时间

(2) 机柜 1 中一个节点失效情况下通过 8 次实验得多组容错和传统容错中作业运行时间对比。多组容错和传统容错技术中作业平均运行时间分别是 71 s 和 77 s, 多组容错技术比传统技术提前 6 s 完成任务, 在大规模的作业中工作效率高的优势将更加明显, 具体实验结果如图 5 所示。

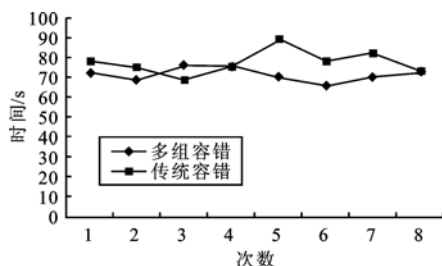


图5 一个节点失效两种容错机制运行时间对比图

但是从图 5 中可以看出, 在第 3 次实验时传统容错机制要比多组容错机制花费的时间要短, 这是因为可能存在数据本地性的原因, 在同机柜中不存在数据本地性的空闲节点, 可是在别的机柜中存在, 在这种情况下传统的容错机制可能就更占优势, 但是在刚开始作业运行开始时任务调度中就考虑到数据本地性, 所以这样情况发生的概率太低对多组容错机制的整体优势影响不大。

(3) 机柜 1 中两个节点失效情况下通过 8 次实

验得出多组容错和传统容错中作业运行时间对比如图 6 所示。在这种情况下两种技术作业完成平均的运行时间约是 119 s 和 149 s, 多组容错技术比传统技术提前足足 30 s 完成任务, 具体实验结果如图 6 所示。

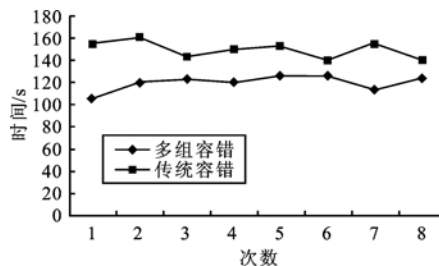


图6 两个节点失效两种容错机制运行时间对比图

从上面所有的实验可以看出多组容错技术提高了 MapReduce 的工作效率, 并且随着集群规模的增大, 失效机器的增多, 增效会更加明显。

5 结束语

本文提出了云计算下 MapReduce 多组容错机制针对传统容错技术再执行和备份方案的前提进行补充, 提出建立多组关系可以更快更准确地发现失效节点, 通过实验对 MapReduce 多组容错机制的性能进行测试, 结果表明, 能够减少发现失效节点的时间, 提高 MapReduce 的工作效率。

MapReduce 多组容错机制相对于传统的容错技术没有从集群全局考虑数据本地性造成机柜失效机器极少, 的情况下可能选择任务迁移的机器不是最理想的, 但是在任务分配初期就考虑数据本地性的影响使这种情况发生的概率极低, 对多组容错机制的整体优势影响不大。在为失效节点启动任务迁移时并没有考虑到负载均衡问题, 将来可以考虑负载均衡的影响, 进一步提升系统性能。

参考文献:

- [1] 王鄂, 李铭. 云计算下的海量数据挖掘研究[J]. 现代计算机: 专业版, 2009(11): 22-25.
- [2] Dean J, Ghemawant S. MapReduce: simplified data processing on large clusters[J]. Communications of the ACM, 2008, 55(1): 13.
- [3] Ko S Y, Ho Que I, Chko B, et al. Making cloud intermediate data fault-tolerant[C]//SOCC 2010. Princeton University, 2010.
- [4] 付金辉, 董国良. 主动容错技术在集群系统中的研究与应用[J]. 高性能计算技术, 2012(5): 64-65.

(下转第 59 页)

- ming to enhance physical layer security using relays [J]. IEEE Transactions on Signal Processing, 2011, 59(3):1317-1322.
- [2] 温祥西, 孟相如, 马志强, 等. 小时间尺度网络流量混沌性分析及趋势预测[J]. 电子学报, 2012, 40(8): 1609-1616.
- [3] 胡治国, 张大陆, 侯翠平, 等. 基于随机网络的多步网络时延预测模型[J]. 计算机科学, 2009, 36(7): 85-87.
- [4] Li S, Yin Q, Guo P. A hierarchical mixture of software reliability model for prediction[J]. Applied Mathematics and Computation, 2007, 185(2):1120-1130.
- [5] Farsi H, Gopal F. Artificial neural network simulator for supercapacitor performance prediction[J]. Computational Materials Science, 2007, 39(3): 678-683.
- [6] 单伟, 何群. 基于非线性时间序列的预测模型检验与优化的研究[J]. 电子学报, 2008, 36(12):2485-2489.
- [7] 程祥, 张忠宝, 苏森, 等. 基于粒子群优化的虚拟网络映射算法[J]. 电子学报, 2011, 39(10): 2240-2244.
- [8] 潘乔, 罗辛, 王高丽, 等. 基于 FARIMA 模型的流量抽样测量方法[J]. 计算机工程, 2010, 36(15): 7-8.
- [9] 陈卫民, 陈志刚. 基于 PSR-LSSVM 的网络流量预测[J]. 计算机科学, 2012, 39(7): 92-94.

作者简介:

李文泽 男, (1979—), 硕士, 讲师. 研究方向为网络优化.

盛光磊 男, (1981—), 硕士, 讲师. 研究方向为网络安全.

(上接第 55 页)

- [5] 朱浩. 云环境下 MapReduce 容错技术的研究[D]. 上海: 上海交通大学, 2012.
- [6] 王晋川, 何宏, 张福临. MapReduce 框架与调度容错机制研究[J]. 中国储运, 2010(12): 90-91.
- [7] Cheng D. GridMix introduction [EB/OL]. <http://dongxicheng.org/mapreduce/hadoop-gridmix-benchmark/>.

作者简介:

张治斌 男, (1953—), 教授. 研究方向为云计算、分布式计算等.

李燕歌 女, (1987—), 硕士研究生. 研究方向为云计算、MapReduce 并行编程模型.