云环境下 Hadoop 平台的作业调度算法①

周文琼[1], 王乐球[2], 叶 玫[1]

[1](广东科学技术职业学院 计算机工程技术学院, 珠海 519080)
[2](中山大学 资讯管理学院, 广州 510000)

摘 要: 云计算的研究和应用将有一片广阔的前景, 详细研究和分析了 Hadoop 平台架构和核心原理, 研究了 Hadoop 现有的典型作业调度算法, 并针对算法存在需要预先配置的问题, 提出了基于朴素贝叶斯分类的作业调度算法, 通过仿真实验, 可以看出改进的算法具备了良好的学习能力, 性能良好, 可以减轻管理员的负担, 提高管理效率, 减少人工错误的可能性.

关键词: 云; Hadoop; 算法; 朴素贝叶斯

Hadoop Platform Job Scheduling Algorithm Under Cloud Computing

ZHOU Wen-Qiong[1], WANG Le-Qiu[2], YE Mei[1]

[1](Computer Engineering Technical College, Guangdong Institude of Science and Technology, Zhuhai 519080, China)
[2](School of Information Management, Sun Yat-Sen University, Guangzhou 510000, China)

Abstract: Faced with the broad prospect in research & application of cloud computing, the paper has studied and analyzed architecture and core principle of Hadoop platform, and its existing typical job scheduling algorithm. Furthermore, to resolve the pre-configuration issue of the algorithm, improved algorithm is proposed based on Naive-Bayes classification. According to the simulation test result, the improved algorithm is proven to be with satisfied performance, easy to learn. It may reduce administrator's workload, improve management efficiency, and reduce human error.

Key words: cloud; Hadoop; algorithm; naive Bayes

云计算是 2007 年 google 率先提出的概念, 它是一种全新的商业模式, 其实现形式主要是利用多台廉价机器相互连接和协同运作, 使其综合计算能力与价格昂贵的超级服务器匹配, 从而避免昂贵的硬件投资.

云计算的商业前景和应用需求已经毋庸置疑, 目前所有 IT 行业巨头都将云计算作为未来发展的主要战略之一. 2007 年 08 月 IBM 推出“蓝云(Blue Cloud)”计划; 2007 年 10 月 Google 宣布了云计划, 并与 IBM 合作, 将全球多所大学纳入“云计算”计划; 2007 年亚马逊开放了名为“弹性计算机云”(Elastic Compute Cloud, EC2)的服务, 以便中小企业可以按需购买亚马逊数据中心的处理能力; 2007 年 11 月雅虎建立一个小规模的云, 开放给卡内基梅隆大学的研究人员; 微软公司正在开发互联网操作系统“Midori”, Midori 的目标是大规模应用云计算技术. 云计算应用及研究的技术方向日益丰富, 主要包括: 云计算存储结构研究、虚拟化技术、云数据管理研究、云编程模式、云网络的研究和云安全的研究. 目前云计算的数据存储技术主要有两大体系: Google 的非开源 GFS(Google File System, 谷歌文件系统)体系和 Hadoop[1-3]团队的开源 HDFS(Hadoop Distributed FileSystem, Hadoop 分布式文件系统).

Hadoop 拥有良好的扩展性、容错性和可用性, 基于该平台, 不仅能较容易地搭建云环境下的数据存贮与分析平台, 而且推动了云环境下数据处理平台技术的发展, Hadoop 已经得到工业界的广泛应用和学术界

① 基金项目:国家自然科学基金(61003253);广东省科技专项资金(1312212200105)

收稿时间:2013-10-09;收到修改稿时间:2014-11-08

的关注. Hadoop 调度器的主要功能是分配计算资源和控制作业执行的顺序, 对平台的计算资源分配和作业执行起着决定性作用, 但现有的Hadoop平台作业调度算法存在不足之处, 所以, 研究现有 Hadoop 平台作业调度算法, 改进算法不足之处, 对提高系统资源的利用和整体平台的性能具有十分重要的意义.

本文主要介绍开源云计算平台 Hadoop 的整体框架, 对 Hadoop 中的两个重要部分 MapReduce 和 HDFS 做了详细分析; 在此基础之上研究了 Hadoop 任务调度算法, 并提出了一个新的调度算法.

1 Hadoop平台研究

Hadoop 是 Apache 的一个开源分布式计算开源框架, 采用 java 语言开发, 使用户可以在不了解分布式底层细节的情况下开发分布式程序, 能充分利用集群的威力高速运算和存储. Hadoop 的核心设计是 MapReduce、HDFS 和 HBase, 目前已在在很多大型网站上都已经得到了应用. Hadoop 的整体技术框架如图 1 所示.

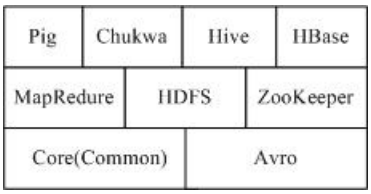

图 1　Hadoop 技术框架图

Core(Common)是 Hadoop 项目的核心, 其他子项目都是在 Core(Common)的基础上发展的; Avro 是 Hadoop 的 RPC(Remote Procedure Call, 远程方法调用); MapRedure 是 Google 的 MapRedure 的开源实现, 是一种大型数据的分布式计算模型; HDFS 是 Google 的 GFS 的开源实现, 是一种分布式文件存贮系统; Pig 是 SawZall 的开源实现, 是一种在 MapReduce 上构建的高级数据流语言; Chukwa 是一种管理大型分布式系统的数据采集系统; Hive 是提供数据摘要和查询功能的数据仓库; HBase 是 Google 的 BigTable 的开源实现, 是一种支持结构化数据存贮的分布式数据库.

1.1 HDFS 分布式文件系统

HDFS 是适合运行在通用硬件上的分布式文件系统, 具有高度容错性, 能提供高吞吐量的数据访问, 适合部署大规模数据集上的应用.

HDFS 采用主从(Master/Slave)架构, 一个 HD FS 集群有一个 Master 和多个 Slave, 前者是名字节点 NameNode, 后者是数据节点 DataNode. Namenode 是中心服务器, 主要管理文件系统的命名空间 namespace 和客户端对文件的访问; Datanode 管理所在节点的存储; 在HDFS内部, 将大文件划分为n个数据块(n>=1), 数据块存储在 Datanode 节点, Namenode 执行文件系统的命名空间操作, 包括打开、关闭、重命名等, 同时决定了数据块存贮的 Datanode 节点位置. 图 2 是 HDFS 的结构示意图.

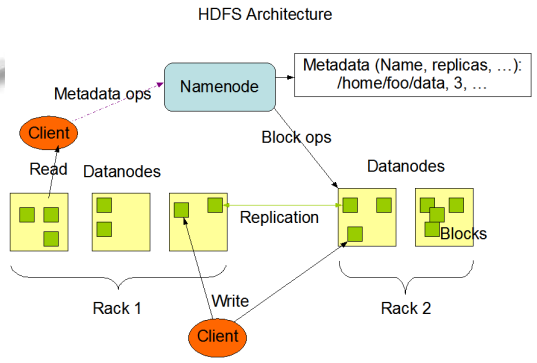

图 2　HDFS 结构示意图

如图所示, Namenode 存储数据块的元数据内容, 描述数据块在 DataNode 中的位置, 具体存储工作由 Datanode 完成. HDFS 的具体操作有: ①客户端向 Namenode 发送文件请求; ②Namenode 查找元数据, 并将该数据所在位置通知客户端; ③客户端到 Datanode 本地查找该文件. 在 HDFS 体系下, 单一节点 Namenode 简化了系统的架构, 为了提高 Namenode 的性能, 全部文件的元数据都在内存中维护, 所以由于内存大小的限制将导致一个 HDFS 集群的提供服务的文件数量的上限.

1.2 MapReduce

MapReduce[4-6]是 Hadoop 是实现分布式计算模型的简易编程模型, 用于大规模数据集的并行运算, 该模型解决了下列复杂问题: 数据分布式存储、作业调度、容错、机器间通信等问题, 基于该平台, IT 工程师能较轻松编写出结构简单、应用于集群处理大规模数据的并行分布式程序.

Hadoop MapReduce 基于“分而治之”思想, 将计算任务抽象成 map 和 reduce 两个计算过程, 即“分散

运算—归并结果”的过程. 首先, MapReduce 程序将输入数据分割成不相关的键/值对(key1/value1)集合, 由多个 map 任务并行地处理这些键/值对; 其次, MapReduce 对 map 的输出(key2/value2 键值对集合)按照 key2 进行排序; 再次, 将同一个 key2 的所有 value2 组合在一起作为 reduce 任务的输入; 最后, reduce 任务计算出最终结果, 输出 key3/value3 键值对.

MapReduce 运行框架由五部分组成: JobClient、Jobtracker、Tasktracker、ChildJVM、HDFS. MapReduce 作业运行机制: ①JobClient 将任务的所有资源复制到 HDFS, 并将任务提交给 JobTracker; ②Jobtracker 调度所有 TaskTracker, 默认配置下按照先进先出 FIFO 策略进行调度, 类似 HDFS 的 dataname 节点; ③Tasktracker 负责执行所有 Task, 类似 HDFS 的 datanode 节点; ④ChildJVM 是执行 MapReduce 程序的子进程, 被本地 TaskTracker 控制, 每个 Child 执行一个 Map 任务或 Reduce 任务; ⑤HDFS 保存所有程序资源, 文件的输入输出都在 HDFS 中保存. Jobtracker 不会主动分配任务给 Tasktracker, 而是被动的接受来自 Tasktracker 的任务请求, 然后进行调度分配; Tasktracker 每三秒钟向 Jobtracker 发送一个心跳包, 用来汇报 Tasktracker 的情况和任务运行状态, 如 Tasktracker 节点有空余的计算资源, 会在心跳包中发送请求任务信息, Jobtracker 根据调度器的策略进行任务分配.

通常情况下, MapReduce 框架和分布式文件系统是运行在一组相同的节点上的, 即计算节点和存储节点物理上在相同节点. 在该配置下, 框架在那些已经存好数据的节点上高效地调度任务, 使整个集群的网络带宽被高效地利用.

2 改进的任务调度算法

2.1 现有 Hadoop 任务调度算法分析

现阶段 Hadoop 典型的任务调度算法[7]主要有三种: FIFO(First In First Out, 先进先出)调度算法、Capacity 调度算法、Fair 调度算法. FIFO 是 Hadoop 的默认任务调度算法, Capacity 是由 Yahoo 开发的基于容量的作业调度算法, Fair 是由 Facebook 开发的公平调度算法.

(1)FIFO 调度算法

FIFO 是 Hadoop 默认的调度算法, 用户作业都被提交到一个队列中, 由 JobTracker 为每个节点分配任务, 选择将被执行的业的策略是按照作业的优先级(通常是提交时间的先后顺序), 且当任务在某节点中被执行时, 是不可被剥夺的, 算法的具体实现是 JobQueueTaskScheduler.

FIFO 算法的优点是实现简单、运行稳定; 因为 FIFO 算法主要针对单用户单类型作业设计, 所以处理多用户多类型作业时性能低下, 容易导致队列尾端作业饿死、机群整体计算资源浪费.

(2) Fair 调度算法

Fair 调度算法是由 Facebook 提出的作业调度算法, 其目标是使 Hadoop 应用满足多类型作业的并行执行需求, 其设计思想是尽可能保证所有的作业都能够获得等量的资源份额. 具体地说, 当第一个作业执行时, 该作业独占整个集群并使用所有的计算资源; 当其他作业提交时, 就有 TaskTracker 被释放并分配给新作业, 这样, 可保证全部提交的作业都能获得相同的计算资源.

在 Fair 公平调度算法的具体实现中, 有两个关键问题: 如何计算每个作业的公平份额和当有 TaskTracker 空闲时应选择哪个作业执行. 为了选择合适的作业执行, Fair 算法会跟踪每个作业的赤字, 即一个作业在理想情况下应该获得的计算资源量与它实际获得的计算资源量之间的差; 赤字是衡量作业调度公平度的指标, 如果某作业的赤字越大, 则该作业之前受到的不公平越多. 公平调度算法会周期性更新作业的赤字值, 一旦有空闲的 TaskTracker 出现, 它会被首先分配给尚未获得最低资源保障的作业池的从属作业, 然后分配给当前最高赤字的作业使用.

Fair 算法侧重于多用户之间对机群计算资源的公平共享, 提交顺序较晚的用户的作业不易饿死; 但 Fair 算法只强调了队列中作业的整体运行, 忽视了作业的优先级.

(3) Capacity 调度算法

Capacity 调度算法是由雅虎提出的作业调度算法, 它提供了类似公平调度算法的功能, 但是在设计与实现上两者在很多方面存在着差别. 该算法基于资源的调度, 支持多用户多队列, 每个队列可配置一定比例的计算的资源, 作业队列支持作业优先级, 对同一用户提交的作业所占资源量设置上限, 以防止资源都为某个用户服务.

当有 Tasktracker 请求任务时, Jobtracker 从粗粒度到细粒度依次选择一个队列, 队列选择策略是: 计算

每个队列中正在运行的任务数与其能够运行的任务数的最大容量的比值(numRunningTasks/taskCapacity), 选择比值最小的队列; 在该队列中选择一个作业, 作业选择策略是: 选中队列中, 作业按优先级排序, 依次匹配队列中的作业, 选择满足下列两个条件的作业: 作业的所属用户占用资源未达到系统配置的资源使用上限、请求任务的 Tasktracker 内存足够该作业的任务运行; 然后尽量选择该作业的本地化计算 Map 任务即可.

Capacity 调度算法的核心思想是使计算资源在队列之间, 用户之间合理分配, 但因为该算法对数据本地化计算放到最后考虑, 所以会影响作业执行效率.

2.2 朴素贝叶斯分类器算法

贝叶斯分类算法是一类利用概率统计知识进行分类的算法, 该算法能运用到大型数据库中, 且方法简单、分类准确率高、速度快. 朴素贝叶斯分类[8]的定义如下:

(1) 设 X={a1,a2,⋯,an}为一个待分类项, 每个 a 为 x 的一个特征属性;

(2) 有类别集合 C={c1,c2,⋯,cm}; (3)将一个待分类样本 X 分配给类 C i(1≤i≤m), 当且仅当:

P(Ci | X) > P(C j|X) (1≤i, j≤m, j ≠i);

(4) 根据贝叶斯定理 $P(C_i \mid X)=\frac{P(C_i)P(X \mid C_i)}{P(X)}$,

因为 P(X)对于所有类别均为常数, 所以有:

$$P(C_i \mid X)=\frac{P(C_i)P(X \mid C_i)}{P(X)} \propto P(C_i)P(X \mid C_i)$$

其中, $P(C_i)=\frac{S_i}{S}$, (Si 是 Ci 在训练样本中的实例数; S 是训练样本的总实例数); 朴素贝叶斯分类模型的公式为:

$$C_{nb}(X)=\arg\max_{C_i \in C} P(C_i)\prod_{k=1}^{n} P(a_k \mid C_i)$$

其中, 概率 P(a1 | Ci)、P(a2 | Ci),⋯, P(an | Ci)由训练样本估值.

2.3 改进的基于朴素贝叶斯分类器的作业调度算法

从现有 Hadoop 典型调度算法分析可知, 实现 Hadoop 应用的任务调度, 需要预先进行配置. 配置数据的恰当与否, 直接关系到任务的顺利完成和系统资源的使用效率, 要准确设置参数, 要求管理员对 MapReduce 作业的资源使用特性和 TaskTracker 的资源情况都有着清晰认识. 在实际工作中, 面对大量用户、作业和大集群时, 预先配置往往忽略了作业在实际运行时的资源使用情况, 管理员难以完成配置, 容易出错. 为了解决实际工作环境中预先配置问题, 需要设计一种算法, 该算法不需要预先配置, 而是通过不断监控或者学习任务执行过程中的实际资源使用情况来调整作业选择及任务分配. 我们提出了基于朴素贝叶斯分类器的任务调度算法, 通过学习的方式来避免现有算法的预先配置问题.

(1) 算法思想

利用朴素贝叶斯分类器将作业队列里的作业分为两类[9]: 好作业和坏作业(设C1=0表示好作业, C2=1表示坏作业); 当 JobTracker 收到任务请求心跳时, 首先在作业队列中先选择一个好作业, 然后在从好作业中选择任务来分配, 最后任务的运行结果又将反馈影响以后的作业分类, 从而影响作业选择及任务分配. 为了选择一个最适合在某 TaskTracker 上运行的作业, 需要考两类特征变量: 作业特征和资源特征, 前者主要描述作业对资源的使用情况, 后者主要描述一个 TaskTracker 计算节点上的计算资源的状态和质量.

(2) 基于朴素贝叶斯分类器的任务调度算法流程

该算法流程如图 3 所示, 算法执行步骤主要有五步.

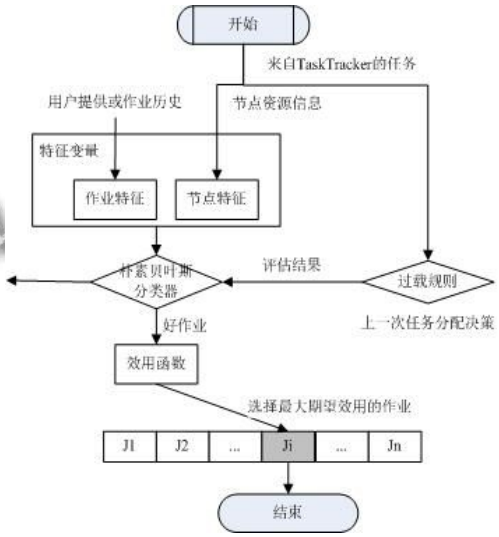

图 3　基于朴素贝叶斯分类的作业调度算法流程

①TaskTracker 向 JobTracker 发送任务请求心跳, 该请求心跳中包含 TaskTracker 上的动态节点资源信息: CPU 使用率、I/O 读写速率、网络传输速率、空闲物理内存容量、空闲虚拟内存容量和磁盘可用空间. 这些资源使用信息有两类用途, 其一是用作节点特征,

其二是发送给过载规则判断上次的分配任务是否导致该 TaskTracker 过载.

②过载规则综合 TaskTracker 节点的动态资源使用信息和过载标准判断上次分配的任务是否导致该 TaskTracker 过载, 将评估结果反馈给朴素贝叶斯分类器.

③朴素贝叶斯分类器根据过载规则的评估结果更新训练样本的概率: P(C1)、P(C2)、P(Fi| C1)、P(Fi| C2)(i=1,..,n)等值, 当新作业 j 提交时, 朴素贝叶斯分类器根据最新训练数据进行鉴别, 判断作业 j 是好作业还是坏作业, 并将作业 j 放置到相应队列.

④如果好作业队列有多个好作业, 则需要对每个好作业计算其期望效用, 最后选择期望效用值最大的作业, 将该作业的任务分配到请求任务的 TaskTracker 上执行.

⑤该任务在 TaskTracker 执行完成之后, TaskTracker 发送新的任务请求心跳.

3 仿真实验与结果分析

仿真系统运行环境采用八个节点的集群, 其中一个节点用做主控节点(部署为 NameNode 和 JobTracker), 其余七个节点用做 DataNode 节点. 在仿真实验中, 我们分别使用四种算法运行三个应用: 单词出现频率应用 WordCount、网页爬虫抓取网页行为应用 URLGet 、对输入的 key-value 对进行大整数乘法计算应用 CPUActivity . 每种算法运行 10 次, 记录每次算法运行的时间, 每次取三个应用的合计运行时间, 仿真实验结果如图 4 所示.

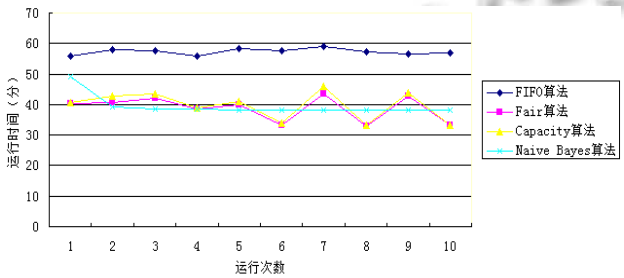

图 4 性能比较实验结果

由图4看出, FIFO 算法稳定, 但是运行时间最长、效率最低; Fair 算法和 Capacity 算法的运行曲线比较一致, 运行时间随着配置(最多可同时运行的任务数、作业池或者作业队列的配置等)不同而存在较大的波动, 在参数配置不当时运行时间明显增加; 朴素贝叶斯任务配置算法在学习训练阶段时(系统刚开始运行时)运行时间比较长, 但是随着系统的继续运行, 运行时间约稳定在 38 分钟.

从实验结果可以得知, 基于朴素贝叶斯分类器的任务调度算法的性能优于 FIFO 算法的性能, 但与公平算法和计算能力算法比较, 在预先配置适当的情况下, 改进算法性能可能略逊, 但在预先配置不当的情况下, 改进算法的性能更优秀.

4 结语

云计算的研究和应用将有一片广阔的前景[10,11], 本文详细研究和分析了 Hadoop 平台架构和核心原理, 在此基础上, 研究了 Hadoop 现有的典型作业调度算法, 并针对算法存在需要预先配置的问题, 提出了基于朴素贝叶斯分类的作业调度算法, 通过仿真实验, 可以看出改进的算法具备了良好的学习能力, 性能良好, 可以减轻管理员的负担, 提高管理效率, 减少人工错误的可能性.

参考文献

1 赵彦荣,王位平,孟丹,等. 基于 Hadoop 的高效连接查询处理算法 CHMJ.软件学报,2012,23(8):2032−2041.

2 申德荣,于戈,王习特,等.支持大数据管理的 NoSQL 系统研究综述.软件学报,2013,24(8):1786−1803.

3 廖彬,于炯,张陶,等.基于分布式文件系统 HDFS 的节能算法.计算机学报,2013,36(5):1047−1064.

4 翟岩龙,罗壮,杨凯,等.基于 Hadoop 的高性能海量数据处理平台研究.计算机科学,2013,40(3):100−103.

5 葛君伟,蒋仙,方义秋.消息代理机制下的 MapReduce 数据流优化.计算机工程与应用,2013,49(5):120−122,262.

6 陈彦舟,曹金璇.基于 Hadoop 的微博舆情监控系统.计算机系统应用,2013,22(4):18−22,9.

7 傅杰,都志辉.一种周期性 MapReduce 作业的负载均衡策略.计算机科学,2013,40(3):37−40.

8 张依杨,向阳,蒋锐权,等.朴素贝叶斯算法的 MapReduce 并行化分析与实现.计算机技术与发展,2013,23(3):23−26.

9 余正祥.基于学习方式对 Hadoop 作业调度的改进研究.计算机科学,2012,39(6):220−222.

10 周文琼,徐猛,尚敏,等.基于工作日历的工作流时间管理的研究与实现.测控技术,2013,32(7):91−94.