

MapReduce 中数据倾斜解决方法的研究

王 刚,李盛恩

(山东建筑大学 计算机科学与技术学院,山东 济南 250101)

摘 要:随着移动互联网和物联网的飞速发展,数据规模呈爆炸性增长态势,人们已经进入大数据时代。MapReduce 是一种分布式计算框架,具备海量数据处理的能力,已成为大数据领域研究的热点。但是 MapReduce 的性能严重依赖于数据的分布,当数据存在倾斜时,MapReduce 默认的 Hash 划分无法保证 Reduce 阶段节点负载均衡,负载重的节点会影响作业的最终完成时间。为解决这一问题,利用了抽样的方法。在用户作业执行前运行一个 MapReduce 作业进行并行抽样,抽样获得 key 的频次分布后结合数据本地性实现负载均衡的数据分配策略。搭建了实验平台,在实验平台上测试 WordCount 实例。实验结果表明,采用抽样方法实现的数据划分策略性能要优于 MapReduce 默认的哈希划分方法,结合了数据本地性的抽样划分方法的效果要优于没有考虑数据本地性的抽样划分方法。

关键词:大数据;MapReduce;负载均衡;抽样

中图分类号:TP301

文献标识码:A

文章编号:1673-629X(2016)09-0201-04

doi:10.3969/j.issn.1673-629X.2016.09.045

Research on Handling Data Skew in MapReduce

WANG Gang, LI Sheng-en

(School of Computer Science and Technology, Shandong Jianzhu University,
Jinan 250101, China)

Abstract: With the rapid development of mobile Internet and the Internet of Things, the data size explosively grows, and people have been in the era of big data. As a distributed computing framework, MapReduce has the ability of processing massive data and becomes a focus in big data. But the performance of MapReduce depends on the distribution of data. The Hash partition function defaulted by MapReduce can't guarantee load balancing when data is skewed. The time of job is affected by the node which has more data to process. In order to solve the problem, sampling is used. It does a MapReduce job to sample before dealing with user's job in this paper. After learning the distribution of key, load balance of data partition is achieved using data locality. The example of WordCount is tested in experimental platform. Results show that data partition using sample is better than Hash partition, and taking data locality is much better than that using sample but no data locality.

Key words: big data; MapReduce; load balancing; sampling

0 引 言

伴随着互联网、物联网和移动互联网的快速发展,每天会产生海量数据,数据处于爆炸式的增长状态,这预示着大数据时代的到来。MapReduce^[1]是 Google 提出的一种分布式计算框架。由于其具有高可扩展性、高效性和容错性等特点,在大规模数据处理中得到了广泛应用。用户使用 MapReduce 处理海量数据时,只需根据业务逻辑编写 Map 和 Reduce 函数即可,并行化、容错、数据分发和负载均衡等复杂的技术由 MapReduce 运行时库自动完成。Hadoop 是 MapReduce 的

开源实现,在云计算和大数据处理等领域应用广泛,成为了研究的热点。

性能优化的重点就是负载均衡。在 MapReduce 分布式计算框架下,用户提交的作业被划分成若干块,每个块被分配给一个 Mapper 执行,Map 阶段产生的中间结果经过划分函数交给 Reducer 执行并产生最终的结果。整个作业的完成时间由 Reducer 运行最慢的决定。当节点的负载出现不平衡时,负载重的节点会制约作业的完成时间。因此,每个节点能否在一致的时间内完成是影响分布式计算性能的关键因素。

收稿日期:2015-10-22

修回日期:2016-02-24

网络出版时间:2016-08-01

基金项目:国家自然科学基金资助项目(61170052)

作者简介:王 刚(1990-),男,硕士研究生,CCF 会员,研究方向为大数据、数据库;李盛恩,教授,研究方向为数据库、数据挖掘。

网络出版地址: <http://www.cnki.net/kcms/detail/61.1450.TP.20160801.0842.012.html>

文中首先通过抽样获取 key 的频率分布信息,然后根据数据本地性特征,利用贪心策略实现 Reduce 节点的负载均衡。

1 相关工作

1.1 MapReduce

Google 过去在处理海量数据时,采用了高配置服务集群的方法,但是随着数据规模越来越大,传统方法在性能方面表现不足。Google 设计了新的分布式计算模型 MapReduce, MapReduce 可以部署在廉价的商用机器上,提高了处理海量数据的性能,降低了硬件成本。MapReduce 最大的优势就是简单易用。

Hadoop 开源实现了 Google 的 MapReduce 模型,并且提供了分布式文件系统 HDFS。用户提交作业后,数据被等大小切分给 Map 节点处理,Map 节点执行 Map 函数产生中间结果并根据划分函数保存在本地磁盘。Reduce 节点读取中间结果执行 Reduce 函数产生最终的结果。在这个过程中,用户不必关注分布式处理的细节,作业调度、数据划分以及容错处理这些细节由 MapReduce 自动完成。除了 Google 的内部实现外,MapReduce 还有一个应用广泛的开源实现 Hadoop。

1.2 已有工作

MapReduce 默认的划分方法是把哈希值相同的 key 分配给同一个 Reducer 节点,在数据倾斜的情况下,容易造成 Reducer 端负载不均,影响任务的完成时间。目前研究人员针对 Reducer 端负载不平衡做了大量的研究工作。文献[2]提出的 SkewTune 系统对 Hadoop 进行功能增强,当有空闲节点时,系统会将当前负载最重的任务分配给空闲节点,从而缩短整个作业的执行时间。文献[3]提出了 LEEN 算法。该算法设计了最优的划分函数,通过把一个 key 分配到最合适的分组来实现负载均衡。这些方法都是在 MapReduce 运行过程中进行调整,操作复杂,通用性低。

除了修改 MapReduce 框架来消除负载不均外,目前还有一种常用的方法就是抽样。文献[4]通过抽样把 key 划分成大、中、小三种负载,划分函数根据负载大小的不同会有不同的处理方式。文献[5]先执行一个 MapReduce 作业,抽样统计 key 的分布情况,从而给出自定义的划分函数。文献[6]基于 range partition 提出了改进的方法。文献[7]在简单采样的基础上提出性能更优的动态划分方法。

基于以上研究工作,文中尝试利用数据本地性和抽样来完善 Reduce 负载均衡机制。首先通过抽样获取 key 的分布,其次理论分析 Hash 算法的不足,结合数据本地性提出贪心策略的 Reduce 端负载均衡,最后通过大规模的数据验证算法的有效性。

2 基于抽样的数据划分

2.1 数据倾斜

MapReduce 的性能很大程度上依赖于数据的分布^[8],如果数据分布不均匀性能就无法保证。但是科学数据往往都是存在倾斜的,MapReduce 在处理倾斜数据时,Map 阶段的中间结果利用哈希函数分配给 Reduce 阶段的节点,MapReduce 哈希划分: $\text{partition-Num} = \text{key.hashCode}() \% \text{REDUCER_NUM}$ 。这种方法可以保证每个 Reducer 处理的划分中包含的分组数目相同,但无法保证分组内部记录总数相同,特别是在数据倾斜的情况下^[9]。使用 Hash 算法时,多个 key 的 hashcode 与 Reduce 节点数量求余数之后可能具有相同的值,从而使数据划分集中于某一个 Reduce 节点,造成数据分布不平衡。Hash 算法没有考虑 key 的频次,可能存在一些频次大的 key 被划分到同一 Reduce 节点,造成数据不平衡^[10]。

如图 1 所示,有 3 个数据节点,Map 端输入数据有 9 个 key 值,每个 key 值的数据量不相等,但是每个数据节点的总量是相等的。图中 Node₁ 的 key 值 K₃ 的数据量为 12,表示为 K₃:12。计算可得 Node₁ 的数据总量为 70。则由 Hadoop 默认的哈希划分函数分区之后,Reduce 端输入的数据量不相等,出现了数据倾斜,三个 Reducer 的数据量分别为 34,56,120,Reducer₃ 的数据量比其他两个节点的数据量多,数据倾斜会影响任务的最终完成时间。

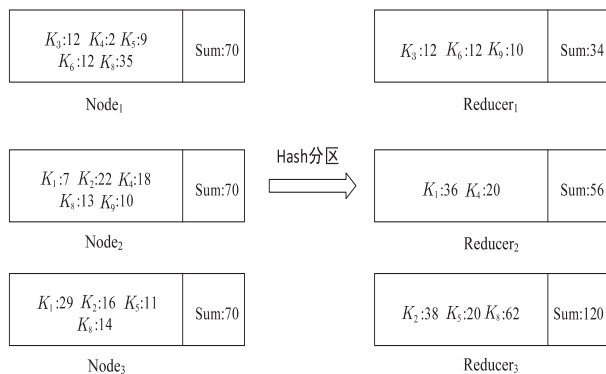


图 1 哈希划分不平衡示例

2.2 抽 样

从总体单位中抽取部分单位作为样本的方法就是抽样^[11]。其基本要求是要保证所抽取的样品单位对全部样品具有充分的代表性。文中算法针对大规模的数据进行处理,如果对所有的数据进行统计,成本太高,因此采用抽样方法,获取 key 的频率分布^[12]。

在抽样前,需要总体单位有序,根据样本容量确定抽样间隔。假设总体单位容量为 M ,样本容量为 N ,则抽样间隔为 $K = M/N$ 。从总体中随机确定一个单位作为第一个样本,然后每隔 K 个距离确定一个样本单

位,达到样本容量即停止。抽取的样本容量越多,抽样的准确度越高。

2.3 负载均衡方法

文中提出基于采样的方法,对 Mapper 的输出结果进行采样,增加一个 mapreduce job 来获得 key 的分布信息。这个过程主要包括两个步骤:

步骤 1:运行一个 mapreduce job 获得中间数据集样本,然后统计 key 的分布,根据样本分布产生划分。划分结果用一个映射数据结构表示:(k,p) 键为 k 被划分到了 p 。

步骤 2:运行真正的数据处理任务。划分函数根据步骤 1 获得的(k,p)产生划分策略而不再利用散列划分。

然而,在 MapReduce 现有的调度策略中并未充分考虑数据本地性^[13],在任务的调度过程中只是简单地从队列中取出第一个待分配的任务给当前可用节点而忽略了中间数据的分布特点,因此可能导致大量的中间文件必须跨网络传输到该节点,如图 2 所示。

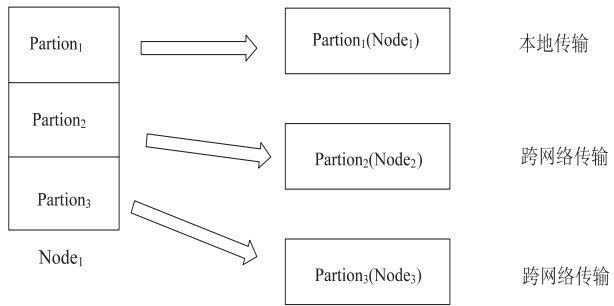


图 2 数据跨网络传输

在图 2 中,Partition₂ 和 Partition₃ 都要跨网络传输,增加了时间开销。为了减少网络传输带来的开销,减少作业运行时间,文中提出了数据本地性感知的抽样划分算法。

定义 1:设 M 表示输入数据的总量,可以用输入数据的行数近似表示, N 表示参与计算的节点数,则划分算法应该使节点的负载接近 M/N 。设 P 表示键值 key 被划分的分区, V 表示节点已分配的数据总量, TK 表示键为 key 的总记录数,元组 ($key, sum, node$) 表示节点 node 上键 key 的数量为 sum。

结合抽样技术和数据本地性算法的具体执行过程如下:

Step1:在每个节点上进行抽样,抽样的结果形式为 ($key, sum, node$)。

Step2:统计所有节点上键为 key 的总数,用 TK 表示。

Step3:将中间结果按数量从大到小排序。

Step4:遍历 ($key, sum, node$),如果 ($sum+V$) 小于 M/N ,则将 key 划分到节点 node。

Step5:遍历处理步骤 4 中没有涉及的 key,把键 key 分配到 V 最小的节点中。

Step6:在抽样过程中,没有抽取的 key 认为是小概率数据,不影响 Reduce 端的负载均衡。没有抽取的 key 使用 Hadoop 默认的哈希划分。

图 1 的例子利用文中的负载均衡算法进行数据划分之后的结果如图 3 所示。可以看出,文中所提的利用数据本地性的抽样方法获得了较好的 Reduce 端负载均衡,优化了默认的哈希划分和简单的抽样划分。

<div><div>K₃: 12 K₈: 62</div><div>Sum: 74</div></div>	Node ₁
<div><div>K₂: 38 K₄: 20 K₉: 10</div><div>Sum: 68</div></div>	Node ₂
<div><div>K₁: 36 K₅: 20 K₆: 12</div><div>Sum: 68</div></div>	Node ₃

图 3 基于本地性抽样划分
算法 1:基于数据本地性的抽样划分。

Input: pairs of ($key, sum, node$), M, N

Output: partition result P

1. $T \leftarrow$ total rows of each key value in all node, put all key into K , initialize the map P and list V
2. while K is not null
3. if $P[key]$ is not partition and $V[node] + TK[key] \leq M/N$
4. $P.add(key, node)$
5. $V[node] += TK[key]$
6. remove the key from K
7. end if
8. end while
9. while K is not null
10. node \leftarrow search minimum $V[node]$ in V
11. if $P[key]$ is not partition
12. $P.add(key, node)$
13. $V[node] += TK[key]$
14. remove the key form K
15. end if
16. end while
17. return P

3 实验结果与分析

文中搭建 Hadoop 集群来验证算法的有效性。实验集群由 6 台计算机组成,每台计算机内存 2 G,磁盘空间 500 G,奔腾处理器。Hadoop 版本 1.0.0,操作系统为 CentOS6.6, JDK1.6。实验所采用的测试方法为利用 Hadoop 进行 WordCount 计算,并分别与默认 Hadoop 和文献[6]中的方法进行比较。

实验结果如图 4 所示。从图 4 中可以看到,当数据分布均匀时,即倾斜度为 0 时,Hash 划分的性能是最好的。随着数据的倾斜度上升,抽样的方法运行时间上升缓慢,而 Hash 上升很快。原因在于当数据分布

均匀时,Hash 划分可以保证负载均匀,而抽样的方法增加了抽样过程的代价,导致运行时间增加,但是代价很小。当倾斜严重时,抽样的划分使负载均衡,性能受倾斜影响不大。

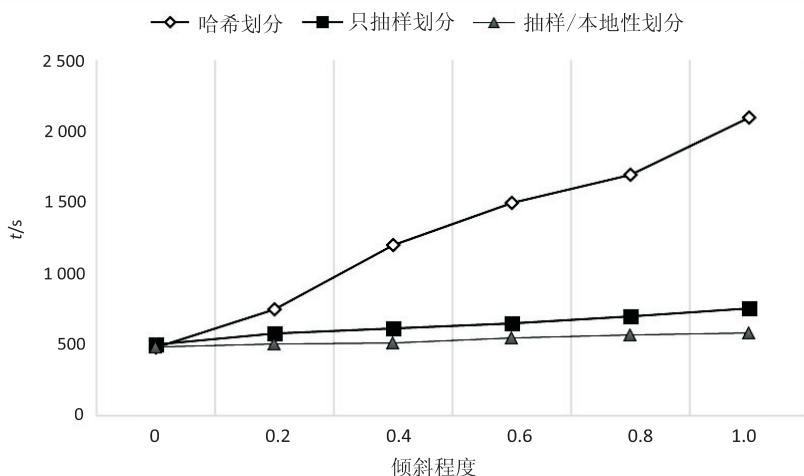


图 4 不同倾斜程度下的运行时间比较

从图 4 中还可以看出,基于数据本地性的抽样划分性能比仅简单抽样划分的性能要好。

4 结束语

文中研究了数据倾斜下的负载均衡优化问题,分析了 MapReduce 中导致节点负载不均的原因,提出了基于数据本地性的抽样划分方法。实验结果表明,与传统的 Hash 划分和只简单抽样的划分相比,文中提出的方法具有更高的效率。

参考文献:

- [1] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters[J]. Communications of the ACM, 2008, 51(1): 107-113.
- [2] Kwon Y C, Balazinska M, Howe B, et al. Skewtune: mitigating skew in MapReduce applications[C]//Proceedings of the 2012 ACM SIGMOD international conference on management of data. [s.l.]: ACM, 2012: 25-36.
- [3] Ibrahim S, Jin H, Lu L, et al. Handling partitioning skew in MapReduce using LEEN[J]. Peer-to-Peer Networking and Applications, 2013, 6(4): 409-424.
- [4] Ramakrishnan S R, Swart G, Urmanov A. Balancing reducer skew in MapReduce workloads using progressive sampling

[C]//Proceedings of the third ACM symposium on cloud computing. [s.l.]: ACM, 2012.

- [5] Xu Y, Zou P, Qu W, et al. Sampling-based partitioning in MapReduce for skewed data[C]//ChinaGrid annual conference. [s.l.]: IEEE, 2012: 1-8.
- [6] 韩 蕾, 孙徐湛, 吴志川, 等. MapReduce 上基于抽样的数据划分最优化研究[J]. 计算机研究与发展, 2013, 50(S): 77-84.
- [7] 周家帅, 王 琦, 高 军. 一种基于动态划分的 MapReduce 负载均衡方法[J]. 计算机研究与发展, 2013, 50(S): 369-377.
- [8] 宛 婉, 周国祥. Hadoop 平台的海量数据并行随机抽样[J]. 计算机工程与应用, 2014, 50(20): 115-118.
- [9] 万 聪, 王翠荣, 王 聪, 等. MapReduce 模型中 reduce 阶段负载均衡分区算法研究[J]. 小型微型计算机系统, 2015, 36(2): 240-243.
- [10] 傅 杰, 都志辉. 一种周期性 MapReduce 作业的负载均衡策略[J]. 计算机科学, 2013, 40(3): 38-40.
- [11] 李 乔, 郑 啸. 云计算研究现状综述[J]. 计算机科学, 2011, 38(4): 32-37.
- [12] 刘寒梅, 韩宏莹. 基于反馈调度的 MapReduce 负载均衡分区算法研究[J]. 信息通信, 2015(10): 41-42.
- [13] 李航晨, 秦小麟, 沈 尧. 数据本地性感知的 MapReduce 负载均衡策略[J]. 计算机科学, 2015, 42(10): 50-56.