Hadoop 迭代优化技术的研究

王晓军,邹亮亮
(南京邮电大学 信息网络技术研究所,江苏 南京 210003)

摘 要:Hadoop 是处理海量数据的分布式计算框架,已经得到了广泛的应用。但是 Hadoop 处理图结构数据存在一些不足。图结构数据的强耦合特性,无法通过一次 MapReduce 计算得出结果,而是需要迭代计算,甚至一次迭代需要多次 MapReduce 完成。而重新启动 MapReduce 作业,开销较大,以及迭代过程中可能存在静态数据的不必要传输。文中在 Hadoop 的基础之上,提出 map 端存储的策略,即将静态数据存储在 map 端,在 map 端完成静态与动态数据相关的计算,减少了整个迭代计算的总运行时间。通过搭建修改过的 Hadoop 平台,与改进前迭代方案进行比较,实验结果表明 map 端存储策略运行时间得到了一定程度的减少。

关键词:Hadoop;迭代;map 端存储
中图分类号:TP31 **文献标识码**:A **文章编号**:1673-629X(2014)09-0098-05
doi:10.3969/j.issn.1673-629X.2014.09.022

Research on Optimizing Iterative Technology of Hadoop

WANG Xiao-jun, ZOU Liang-liang
(Institute of Information Network Technology, Nanjing University of Posts and Telecommunications, Nanjing 210003, China)

Abstract: Hadoop is a distributed computing framework which has been widely used for dealing with huge data. But Hadoop has some disadvantages to process graph data. Because of strong coupling, graph structure data need multiple iterations which may contains several MapReduce computations instead of one MapReduce computation. It costs too much to restart MapReduce job and exists unnecessary transmission for static data in iteration. Propose map side storage strategy based on Hadoop, the static data is stored in map side and finish some related computations with state data. This strategy could reduce whole running time. Experimental results have shown that map side storage strategy spends less time compared with previous strategy through Hadoop platform.

Key words: Hadoop; iteration; map side storage

0 引 言

随着互联网的发展,互联网上的数据也不断增加,如何处理这些庞大的数据或者是从中挖掘出有价值的数据,是当今工业界与学术界研究的重点。而庞大的数据,使用传统的关系数据库存在一定的瓶颈[1],同时数据的集中处理需要考虑机器的性能,以及硬件成本问题。目前有很多优秀的云计算解决方案,如 Dryad 平台和 Apache 在 MapReduce[2] 基础上提出的 Hadoop[3-4] 是解决大数据的有效的平台。Hadoop 的基本思想是将计算分布在大量的不同商用计算节点上,这不仅提供了高效的计算效率和性能,还节约了处理大数据的硬件成本。

在大量数据中,有数据以图结构形式存在,如社交网络数据、网页数据(网页之间的链接关系)。而 Hadoop 也是处理大规模图的一种方式。但图结构数据的强耦合性特点[5-6],无法在一次计算中得出结果,而是需要迭代计算。Hadoop 中的 MapReduce 执行过程分为两个阶段,Map 阶段和 Reduce 阶段,并且每个阶段都是以键值对的形式作为输入与输出。Map 阶段负责将源数据进行整理计算,之后将 Map 计算的键值对结果分布到 Reduce 阶段的节点上进行计算,得出相应的结果。但是迭代计算时,需要重复启动 Job 而造成开销大的问题。在文献[7]中提出了 HaLoop,它是在 Hadoop 的基础上针对迭代的需求进行的改进。通

收稿日期:2013-11-13 **修回日期**:2014-02-16 **网络出版时间**:2014-07-17
基金项目:国家自然科学基金资助项目(61003237)
作者简介:王晓军(1968-),女,研究生导师,副研究员,研究方向为分布计算技术与应用、云计算、数据库技术等;邹亮亮(1988-),男,硕士研究生,研究方向为分布计算技术与应用。
网络出版地址:http://www.cnki.net/kcms/detail/61.1450.TP.20140717.1228.016.html

过在 Master 中增加的 Loop 控制器来控制重复地启动 Map-Reduce,并采用缓存技术来缓存不需要重复传输的数据。增加的 Loop 控制器会减少每次迭代重新启动 Job 的开销,而缓存技术无疑是减少了网络的通信开销。在文献[8-9]中都提出了持久化的作业策略,即在迭代过程中某一 Reduce 作业完毕后不再等待其他 Reduce 作业计算完毕,而直接传输数据到 Map 端,这就会充分利用到计算机的处理性能。同时在解决大规模图的问题时,除了 Hadoop 还有其他的方案,如 Google 提出的 Pregel[10],它是通过消息传递机制和 BSP 并行模型[11]来完成的,但所有的处理数据放置在内存中。

1　迭代处理方案

1.1　Hadoop 处理

使用 Hadoop 处理迭代问题,需要多次启动 MapReduce 框架。主程序对源数据进行数据分片,之后进行 Map 阶段操作,并将其结果通过 Partitioner 分区函数指定到相关的 Reduce 的计算节点。Reduce 通过 RPC 从 Map 端获取数据,并进行聚合计算,得到目标结果后将结果写到分布式文件系统[12](HDFS)中。整个过程是在一次作业(Job)中完成的。但对图结构数据进行计算处理时,如 PageRank[13-14]、社交网络中关系查询、HITS 等,需要多次启动作业才能完成,这需要上一次作业计算的结果作为下一次作业的输入源,然后重新分片、计算,直至计算结果符合条件为止。然而存在的问题是每次启动一次 Job 需要大量的准备工作,开销太大,一旦计算需求迭代次数较多时,就会导致未能充分利用计算机的性能。另外一个不足是在迭代的整个过程中可能存在不变化的数据(静态数据),但是传统的 Hadoop 计算过程中仍然将此类型数据在网络中传输,尤其是在异构环境下,造成了大量的网络开销。

图 1 是利用 Hadoop 处理迭代应用 PageRank 的例子。

URL	PageRank
www.aa.com	1
www.bb.com	1
www.cc.com	1
www.dd.com	1
www.ee.com	1

(a)各网页初始值

URL_src	URL_dest
www.aa.com	www.bb.com
www.aa.com	www.cc.com
www.cc.com	www.dd.com
www.bb.com	www.ee.com
www.bb.com	www.cc.com
www.dd.com	www.aa.com
www.ee.com	www.aa.com

(b)各网页链接关系

图 1　PageRank 求值示意图

PageRank 的求解公式:

$$\mathrm{PageRank}(p_i)=\frac{1-d}{N}+d\sum_{p_j\in M(p_i)}\frac{\mathrm{PageRank}(p_j)}{L(p_j)}$$

其中,$p_1,p_2,\cdots,p_n$ 是需要计算的页面;$M(p_i)$ 是 p_i 链入页面的集合;$L(p_j)$ 是 p_j 链出页面的数量;d 为 0.85;N 是所有页面的数量。

采用 MapReduce 处理上述公式的操作步骤如下:

C1:对 Ta 表中 URL 属性与 Tb 表中 URL_src 属性进行等值连接[15],生成 TL,其中 TL=(url,pagerank,url_dest)。

$$\mathrm{Ta}\underset{\mathrm{URL}=\mathrm{URL\_src}}{\bowtie}\mathrm{Tb}\longrightarrow\mathrm{TL}$$

C2:是根据 TL 表中相同 url 对应的 URL_dest,生成 T_url_dest_pr 表。

$$\mathrm{TL.\ pagerank}/L(\mathrm{url})\longrightarrow\mathrm{T\_url\_dest\_pr}$$

其中,L(url)为 url 链出页面的数量;T_url_dest_pr=(url_dest,pagerank)保存了 URL_dest 局部的 PageRank 值。

C3:根据 C2 生成的表 T_url_dest_pr 中的值按照相同的 URL_dest 值进行聚合操作,完成一次迭代计算,获得 T_new_pr=(url,pagerank)。

$$(1-d)/N+d*\mathrm{sum}(\mathrm{T\_url\_dest\_pr.\ pagerank})\rightarrow\mathrm{T\_new\_pr}$$

Tb 为网页之间的链接关系(静态数据)。Hadoop 在处理时,该数据在每次迭代中都会重复的传输。C1 与 C2 由一次 MapReduce 计算完成,C3 则由另外一次 MapReduce 完成计算,所以计算 PageRank 的一次迭代需要两次 MapReduce 计算。

1.2　HaLoop 处理

HaLoop 是在 Hadoop 的基础上完成的,专门针对需要迭代计算的数据。其改进分为两部分:

(1)添加 Loop 控制器。在传统的 Hadoop 中启动一次 MapReduce 计算就对应一个 Job,迭代就要重复地启动 Job。HaLoop 采用如下方法:每次迭代计算时,Loop 控制器根据自定义 Fixpoint 评估函数来判断计算结果是否满足条件,如果条件未满足,继续在原有 Job 上执行下一次迭代,而非重新启动一次 Job,直到结果满足条件为止,减少了启动 Job 的开销。

(2)数据本地性。HaLoop 区分静态数据与动态数据,并采用了缓存技术,将在迭代过程中所需要的静态数据直接存储到本地节点中,以便在下次迭代计算时可以直接从本地节点中获取静态数据,从而使得静态数据不必在网络中重复地传输,这样就会减少网络开销,提高计算机的计算效率。

文中提出了一种 Map 端存储策略,通过扩展 Mapper 接口以及静态数据存取控制条件的接口,可以有效

地减少在一次迭代中 MapReduce 计算的次数或步骤,以及避免静态数据在网络上传输,以达到缩短计算时间的目的。

2 改进方案

2.1 Map 端存储策略

在迭代处理中,若一次迭代存在一次或一次以上的 MapReduce 计算时,减少每次迭代中 MapReduce 的计算步骤或次数是解决问题的可选方案之一。文中研究提出了 Map 端存储策略,其改进从以下五个方面进行说明。

(1)扩展 Mapper 接口。在 Map 阶段操作时需区分数据类型,区分数据类型是为了方便在 Map 阶段操作时完成两类相关联的数据计算。文中研究通过扩展 Mapper 接口,以满足两类相关联的数据计算需求,并充分地利用 Map 阶段的计算任务。而 MapReduce 中的 Map 阶段完成分布式文件系统加载所有的数据并进行 Map 操作的过程,其结果通过分区函数将数据分配到各 Reduce 计算节点上,由 Reduce 端完成两类数据的具体计算。

(2)Map 端存储静态数据。在 HaLoop 的解决方案中,提出了 3 种缓存:Map 端输入缓存、Reduce 端输入缓存以及 Reduce 端输出缓存。虽然采用 HaLoop 中的 Map 端输入缓存可以解决数据本地性问题,但是此策略仍然无法在 Map 端完成相关联的两类数据的计算。而 Map 端存储策略不仅扩展了 Mapper 接口,而且还将静态数据存放在本地文件系统中,迭代更新的数据仅为动态数据,从而减少了静态数据在混洗阶段的开销。因此,Map 端存储静态数据与扩展 Mapper 接口是相辅相成的。

(3)划分分片策略。一个输入分片为一个 Map 操作的输入块,每次 Map 操作处理一个输入分片,并将每个分片划分成键值对记录。而 Map 端存储策略可以为不同应用的数据选择不同的划分策略,如 PageRank 可以使用指定行数策略,可将源数据的前 100 000 行作为一个分片,剩下的数据再根据 10 000 行进行划分。不同的划分策略有利于扩展 Mapper 接口中的 Map 操作。

(4)迭代终止判定。当应用迭代一定次数时,会满足开发用户的需要。迭代终止条件可以有两种类型:一是程序设置固定的迭代次数,到达阈值就会终止迭代;二是数据结果是否达到收敛,即两次间隔迭代差值是否达到预期设置的值。Map 端存储策略支持上述两种方式。对于第二种控制条件类型,借鉴 HaLoop 中的 Reduce 输出缓存方法,在每次迭代结束时,根据 Reduce 的缓存数据结果来判定是否达到收敛,而不需要像传统 Hadoop 一样,再额外启动一次 MapReduce 来判断终止条件。

(5)动态数据与计算所需的静态数据须在相同的 Map 节点。在迭代应用中,通过扩展的 Mapper 接口,可以允许静态数据与动态数据的计算在 Map 阶段完成操作,但是前提是动态数据与计算时所需要的静态数据必须存储到相同的 Map 端。这可以通过两种扩展方式实现:

· Reduce 结果排序。对 Reduce 的输出结果进行排序,使得排序后的结果可以采用合理的分片方式,保证后续迭代的 Map 操作获取的动态数据与其所需要的静态数据存储在相同的 Map 节点。但此方法不足之处是,仅适用于每次迭代产生的动态数据的记录数稳定的情况,如计算 PageRank 就可以采用这种方式。

· 确保 Reduce 的结果文件被特定的 Map 端获取。这种方式首先将不同 Reduce 任务输出的动态数据写到不同的文件中,然后通过 Hadoop 的 Map 任务的调度,使得启动的 Map 端节点可以获取指定的动态数据文件,以保证 Map 计算时所需要的动态数据与静态数据指派到相同的计算节点上。但这种方案要求 Map 节点数与 Reduce 节点数必须相同。

根据上述改进的五个方面,提出的基于 Map 端存储策略的 Hadoop 来处理需迭代的应用,其执行过程包含两大步骤。

(1)预处理。将原始数据进行预处理,可使用一次 MapReduce 任务完成计算。预处理的职责是依据具体的应用计算需求,将数据划分为两种类型:静态与动态数据,并设置相应标记。不同的应用采用的预处理方式有所不同。预处理结果作为正式迭代计算的数据源。

(2)正式迭代计算。正式迭代分两个阶段,Map 阶段与 Reduce 阶段。在 Map 操作之前,以适当的分片策略进行分片,Map 阶段操作时根据标记将静态数据存储到 Map 操作的节点上,并利用两类数据完成 Map 操作,然后将处理结果通过分区函数指定到相应的 Reduce 计算节点上。而在 Reduce 阶段,首先获取 Map 阶段输出的中间结果,然后进行聚合操作。若需要继续迭代,则下次迭代的 Map 获取上次 Reduce 的计算结果,从而再次进入 Map 阶段。由于静态数据已存放在 Map 端,因此在下次迭代时,Map 端只需获取上次迭代时 Reduce 端产生的动态数据,并与已经存储的静态数据完成相关计算。

Map 端存储策略与 HaLoop 的静态数据处理策略不同之处在于:Map 端存储策略的 Map 阶段操作需要同时读取动态数据与静态数据的键值对记录,并通过扩展 Mapper 接口完成相应的计算;而 HaLoop 在 Map

阶段无法完成此步骤,它需要将 Map 计算出的动态数据传输到 Reduce 端,在 Reduce 阶段完成两类数据的计算操作。

2.2　两种策略求解 PageRank

本节以 PageRank 为例说明 Map 存储策略与 HaLoop 策略的差异。图 2 给出了采用 HaLoop 实现 PageRank 的处理流程(图中静态数据用下划线表示),该流程中每次迭代包含有两次 Map-Reduce 运算。流程中的第一次 Map-Reduce 用于完成页面 PageRank 值与页面链接数据的连接操作,即 C1 与 C2 步骤。同时在 Reduce1 中对相同 key 值的链接数据(静态)进行聚合操作,然后将聚合的静态数据存储在相应的本地文件系统中。第二次 Map-Reduce 是完成 PageRank 值的计算,即 C3 步骤。这次计算中的 Map2 阶段首先对 Map1-Reduce1 的结果数据进行加载计算,然后在 Reduce2 阶段完成 PageRank 值的聚合计算,最后将计算结果写到 HDFS 中。

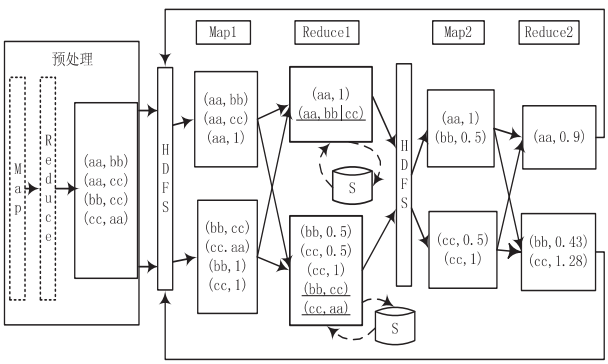

图 2　HaLoop 迭代处理 PageRank 示意图

图 3 描述了采用 Map 存储策略实现的 PageRank 计算流程。图中,Map 计算之前需执行预处理,预处理对链路数据以相同的键值(key 值)进行聚合操作,形成(key,$value_1$|$value_2$…)结构的静态数据,并将静态数据写到 HDFS 中。正式迭代时,采用指定行的分片策略进行分片,并将分片作为 Map 操作的输入数据。第一次迭代时,Map 操作完成每个页面的 PageRank 值的初始化,并将初始化的数据与分片中的记录进行连接操作,形成页面的局部 PageRank 值,并将静态数据存储在本地文件系统中,即 Map 阶段将通过扩展的 Mapper 完成 C1 与 C2 操作;然后 Reduce 阶段完成页面局部 PageRank 的聚合操作,即 C3 步骤。若迭代条件未满足,则对每个 Reduce 的结果进行排序并写到 HDFS 中,并继续迭代。由于 PageRank 的动态数据记录数在每次的迭代中是不变的,并且有序,所以采用指定行的分片策略能够保证 Map 操作从 HDFS 中获取与本地静态数据相同键值的动态数据。

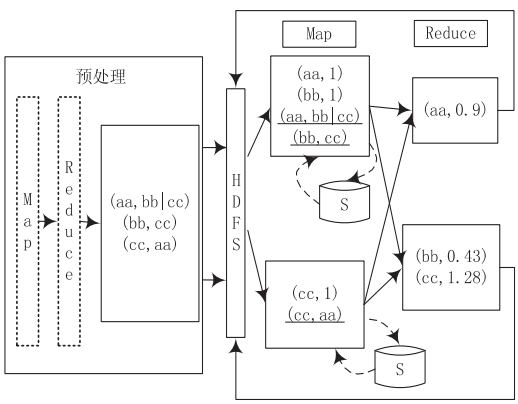

图 3　Map 端存储策略处理 PageRank 示意图

这两种策略不同之处在于预处理、Map 阶段和 Reduce 输出的处理方式:

(1)预处理方式不同。HaLoop 的预处理只是标记出区分静态数据;而 Map 端存储策略是将所有数据整理成(网页 A,网页 A 的所有出链)的结构,以便在 Map 阶段进行存储。

(2)Map 端存储策略扩展了 Mapper 接口,并采用 Map 端存储技术使得动态数据和静态数据的连接操作在 Map 阶段完成,减少了每次迭代中的一次 MapReduce 计算。

(3)对 Reduce 输出的结果进行排序,保证 Map 操作的动态数据与所需要的静态数据在相同的 Map 端上。

3　实验及结果分析

通过上述理论分析显示了 Map 端存储策略可减少迭代应用的计算时间。为比较 Map 端存储策略与 HaLoop 处理应用的实际效果,文中研究对 Hadoop 源码进行修改,并对三组数据集各进行了 3 次实验,实验结果为该 3 次实验的平均值。其三组数据集为:soc-LiveJournal,边数 10 106 166 条,顶点数 997 454 个;wiki-Talk,边数 5 021 410 条,顶点数 2 394 385 个;web-BerkStan,边数 760 059 条,顶点数 685 230 个。

这三组数据集中,第一组数据为社交网站数据,后两组数据为网页数据。整个实验平台是由 4 台 Ubutun12.04 虚拟机构成,每台虚拟机的内存为 2 G,Java 的环境为 Linux JDK1.6,Hadoop 的版本是基于 Hadoop0.21 的改进。

图 4 是计算 soc-LiveJournal 数据集 PageRank 值 8 次迭代中的每次迭代所需的时间图(第一次迭代为预处理)。Map 端存储策略比 HaLoop 在处理迭代应用问题上时间有一定的减少,这是因为 Map 端存储策略减少了一次迭代过程中的 MapReduce 计算次数,所以减少了磁盘开销、网络开销。假设一次迭代中减少的 MapReduce 的计算时间为 T,则在 N 次迭代的情况下,整个运行的时间减少了 $N*T$。图 5 是针对 3 组数据 8 次迭代总运行时间图。总体上,优化后的计算时间得

到一定程度的减少。

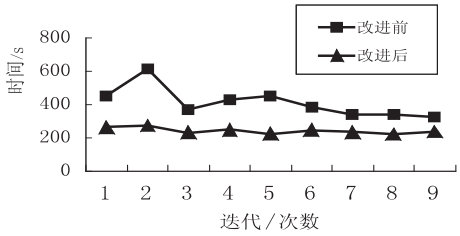

图 4 每次迭代对比图

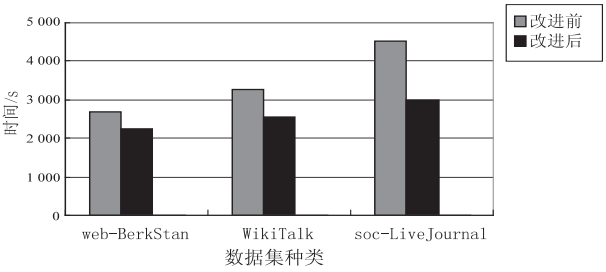

图 5 三组数据 8 次迭代时间对比图

4 结束语

文中首先总结了 Hadoop 处理迭代应用存在的问题,并叙述了目前的解决方案。然后在 HaLoop 的基础上提出了 Map 端存储策略,通过保证数据的本地性以及对 Map 端的扩展,有效地减少了每次迭代运行 MapReduce 的次数或步骤,从而减少了运行时间。最后通过实验验证了改进后的方案比之前的方法在性能上有一定的提高,达到了预期的结果。

参考文献:

[1] 覃雄派,王会举,杜小勇,等. 大数据分析—RDBMS 与 MapReduce 的竞争与共生[J]. 软件学报,2012,23(1):32-45.

[2] Dean J, Ghemawat S. MapReduce:simplified data processing on large clusters[J]. Communications of the ACM,2008,51(1):107-113.

[3] White T. Hadoop 权威指南[M]. 曾大聃,周傲英,译. 北京:清华大学出版社,2010.

[4] 江务学,张 璟,王志明,等. MapReduce 并行编程架构模型研究[J]. 微电子学与计算机,2010,28(6):168-170.

[5] 于 戈,谷 峪,鲍玉斌,等. 云计算环境下的大规模图数据处理技术[J]. 计算机学报,2011,34(10):1753-1767.

[6] 饶 君,张仁波,东昱晓,等. 基于 MapReduce 的大规模图挖掘并行计算模型[J]. 应用科技,2012,39(3):56-60.

[7] Bu Yingyi, Howe B, Balazinska M, et al. HaLoop:efficient iterative data processing on large clusters[J]. Proceedings of the VLDB Endowment,2010,3(1-2):285-296.

[8] Elnikety E, Elsayed T, Ramadan H E. iHadoop:asynchronous iterations for MapReduce[C]//Proc of IEEE third international conference on cloud computing technology and science. Athens:IEEE,2011:81-90.

[9] Zhang Yanfeng, Gao Qixin, Gao Lixin, et al. iMapReduce:a distributed computing framework for iterative computation[J]. Journal of Grid Computing,2012,10(1):47-68.

[10] Malewicz G, Austern M H, Bik A J C, et al. Pregel:a system for large-scale graph processing[C]//Proceedings of the 2010 ACM SIGMOD international conference on management of data. [s. l.]:ACM,2010:135-146.

[11] 任年海. 一个有效的并行模型—BSP 并行模型[J]. 计算机与现代化,2006(3):34-36.

[12] Ghemawat S, Gobioff H, Leung S T. The Google file system [J]. ACM SIGOPS Operating Systems Review,2003,37(5):29-43.

[13] Kamvar S, Haveliwala T, Golub G. Adaptive methods for the computation of PageRank[J]. Linear Algebra and Its Applications,2004,386:51-65.

[14] 李远方,邓世昆,闻玉彪,等. Hadoop-MapReduce 下的 PageRank 矩阵分块算法[J]. 计算机技术与发展,2011,21(8):6-9.

[15] 王晓军,孙 惠. 基于 MapReduce 的多路连接优化方法研究[J]. 计算机技术与发展,2013,23(6):59-62.

(上接第 97 页)

tional conference on wireless and optical communications networks. Cairo:IEEE,2009:1-6.

[11] Kim Y H, Kim S S, Lee S J, et al. Improved 4-ary query tree algorithm for anti-collision in RFID system[C]//Proc of international conference on advanced information networking and applications. Bradford:IEEE,2009:699-704.

[12] Kim Y H, Kim S S, Ahn K. A rapid tag identification method with two slots in RFID systems[C]//Proc of eighth IEEE international symposium on network computing and applications. Cambridge:IEEE,2009:292-295.

[13] Myung J, Lee W J, Srivastava J. Adaptive binary splitting for efficient RFID tag anti-collision[J]. IEEE Communications Letters,2006,10(3):144-146.

[14] Myung J, Lee W. Adaptive binary splitting:a RFID tag collision arbitration protocol for tag identification[J]. Mobile Networks and Applications,2006,5(11):711-722.

[15] Kim S S, Kim Y H, Ahn K. An enhanced slotted binary tree algorithm with intelligent separation in RFID systems[C]//Proc of IEEE symposium on computers and communications. Sousse:IEEE,2009:237-242.

[16] 张学军,王 娟,王锁萍. 基于标签识别码分组的连续识别防碰撞算法研究[J]. 电子与信息学报,2011,33(5):1159-1165.

Hadoop迭代优化技术的研究

万方数据 WANFANG DATA 文献链接

作者： 王晓军， 邹亮亮， WANG Xiao-jun， ZOU Liang-liang
作者单位： 南京邮电大学 信息网络技术研究所,江苏 南京,210003
刊名： 计算机技术与发展 ISTIC
英文刊名： Computer Technology and Development
年，卷(期)： 2014(9)

本文链接：http://d.g.wanfangdata.com.cn/Periodical_wjfz201409022.aspx