文章编号:1000－2367(2016)03－0154－06 DOI:10.16366/j.cnki.1000－2367.2016.03.027

基于 Dijkstra 算法分布式 JobTracker 节点模型通信方式的优化

黄伟建,杨海龙

(河北工程大学 信息与电气工程学院,河北 邯郸 056038)

摘 要:针对 MapReduce 框架下 JobTracker 单点失效而引发的系统运行崩溃问题,通过建立分布式 JobTracker 节点模型来改善 JobTracker 的可靠性,并基于 Dijkstra 算法的单源最短路径思想对新建立模型中主从节点间的通信方式进行优化与改进,使任务节点优先与其最近的不同机架和不同数据中心的 3 个控制节点进行通信.实验结果表明,分布式 JobTracker 节点模型能在一定程度上缓解由单 JobTracker 节点失效引起的作业失败问题,并且改进后的通信方式能够缓解单 JobTracker 节点存在的瓶颈,均衡 JobTracker 节点的负载.

关键词:MapReduce;通信方式;Dijkstra;JobTracker;TaskTracker

中图分类号:TP301 **文献标志码**:A

Hadoop 是 Apache 基金会基于 Google 的两篇关于云计算的论文而开发的一个开源云计算平台,主要由 HDFS(Hadoop Distributed File System)和 MapReduce 两部分构成,其中 HDFS 是基于谷歌 GFS(Google File System)开源的分布式文件系统,主要用于存储和管理执行作业所需的大规模数据和 Jar 文件,MapReduce 并行处理模型则是一种分布式处理系统,可以通过编写 Map(归约)与 Reduce(化简)两个函数进行数据的分析和处理.Hadoop 平台采用的是主从式的 Master/Slave 的框架,从 HDFS 角度来看,它由唯一的 NameNode 主控节点来管理各个 DataNode 节点上的数据与存储,从 MapReduce 角度来看,它由唯一的 JobTracker 控制节点来调度和分派待处理的任务给有空闲任务槽的 TaskTracker 节点,由于系统设计之初仅定义了一个用来存储元数据的 NameNode 节点和一个管理与监控作业运行的 JobTracker 节点,因而单点失效问题成为了 Hadoop 的瓶颈,即当主节点失效的时候会引起整个系统的失效[1].因此很多研究学者围绕 Hadoop 下的单点可靠性存在的不足进行了大量的研究与实验,如基于建立 NameNode 节点的热备份来提高 NameNode 的可靠性[2],基于分布式的 HA(High availability)机制优化因单节点失效而引起的 HDFS 集群重启时间过长,配置参数无法动态更改的问题[3],通过分布式 Pi 推出 NameNode 发生单节点宕机的时机[4],以及为了减轻单节点负载而将 JobTracker 拆分为两个独立进程资源管理和作业控制(包括作业监控、容错等)的下一代 MapReduce 框架 Yarn(MRV2)[5].通过对前人的研究和分析可以看出,基于新增热备节点对 HDFS 的单点问题研究比较多而且相对完善,但是对 MapReduce 的单点问题的改进并不是很多,受 HA 机制的启发,相比于前人基于热备和负载最优的研究,下面通过建分布式的多对多 JobTracker 模型来研究 MapReduce 中 JobTracker 的单点可靠性,并基于 Dijkstra 算法从通信距离角度进一步对新模型主从节点间的通信方式进行优化.

收稿日期:2015-03-31;**修回日期**:2016-03-10.

基金项目:河北省自然科学基金(F2015402077),河北省重点基础研究项目(14964206D).

第 1 作者简介:黄伟建(1964－),男,山西交口人,河北工程大学教授,研究方向为云计算、并行计算、信息管理系统.

通信作者:杨海龙(1988－),男,河北唐山人,河北工程大学硕士研究生,研究方向为云计算与网络信息安全,E-mail:yhl198851@foxmail.com.

1　MapReduce 下节点间的通信方式

MapReduce 采用的是和 HDFS 一样的主从式结构，包括唯一的 JobTracker 主控制节点和多个 TaskTracker 从任务节点，Hadoop 采用了 RPC(Remote Procedure Call Protocol)以使集群中各个组件的通信更加简单便捷. RPC 是一种远程通信协议，在无须了解底层网络技术的情况下，可以通过网络从远程计算机上请求服务. MapReduce 框架下的心跳反馈机制就是建立在 RPC 基础上的通信模型，因此通过心跳机制实现了 MapReduce 中的主从节点间的通信[6]. TaskTracker 从节点是 Hadoop 集群中运行于各个节点上的服务，它扮演着“通信枢纽的角色”，用于沟通 JobTracker 和任务进程. Taskracker 通过心跳机制从 JobTracker 端获取各种命令，启动新任务，提交任务，杀死任务，杀死作业和重新初始化等. MapReduce 下节点间通信的实现依赖 3 种协议，如图 1 所示.

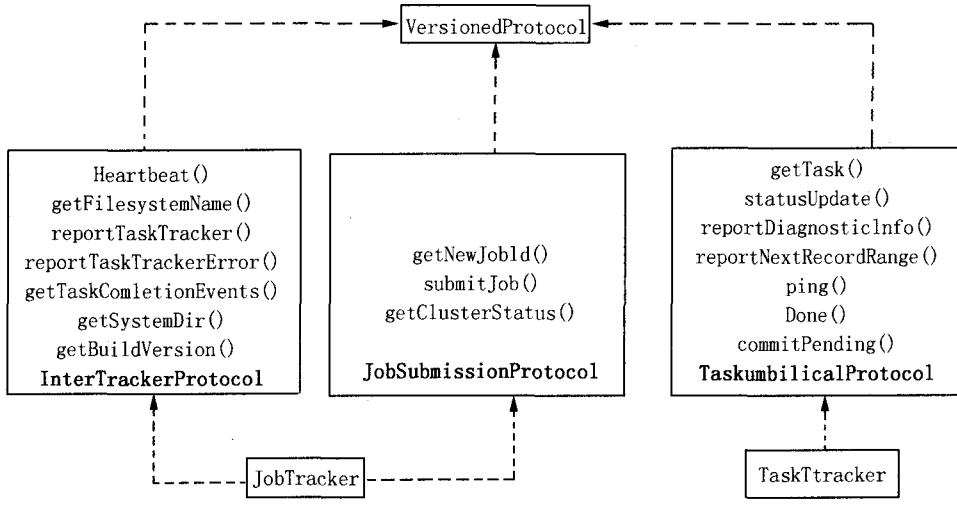

图1 MapReduce通信接口模型

这 3 个协议是 Hadoop 的待实现接口，且都继承自统一公共接口 VersionedProtocol. 其中 JobTracker 类实现了两个通信接口：保证主调节点 JobTracker 与任务节点 TaskTracker 之间能够正常进行通信的 InterTrackerProtocol 和确保客户端与主调节点之间通信的 JobSubmissionProtocol；TaskTracker 类则实现了任务与 TaskTracker 节点之间进行通信的 TaskUmbilicalProtocol[7]. 由图 1 通信模型的接口方法中可以看出，TaskTracker 节点每隔一段时间会调用方法 Heartbeat()控制发送节点向心跳信息提交当前任务节点所在机器的状态信息，如内存大小、外存储器的大小、中央处理器的使用率和可分配给任务的资源的多少等；主控节点接收到心跳信息后做出反馈 HeartbeatResponse，反馈信息的属性包括应答 ID、每次心跳的间隔时间，以及主节点需要从节点所做的操作，如能否开启新任务. 从节点通过调用 getSystemDir()方法与 getFileSystemName 方法能够获取 Job 所需 Jar 文件路径和文件系统路径. 通过图 1 可以看出，如果控制节点失效，则作业的运行会被中断，这对将要运行完成的作业无疑是一种资源和时间的浪费.

2　分布式 JobTracker 节点模型的建立

JobTracker 是 MapReduce 模型的重要组成部分，基本功能包括由任务调度器 TaskScheduler 实现的可用空闲资源的管理和由控制节点及其相关模块实现的对作业的控制与作业完成进度的监控. 由于 JobTracker 同时进行可用资源监控和任务调度，负载过重，因此影响着 Hadoop MapReduce 框架的可扩展性和资源利用率. 而且当 JobTracker 节点失效时，正在执行作业的节点执行状态的元数据会丢失，如一个作业已经执行了一多半，则是一种计算资源和时间的浪费. 而从新运行 JobTracker 和初始化配置运行环境无形增加了时间成本. 因此针对这种情形，雅虎公司的新 Hadoop MapReduce 模型 Yarn 把监控与调度功能拆分，

并由专门的组件实现监控功能[8],JobTracker 则仅负责客户提交的待执行作业的调度问题. 不同于雅虎公司对单独功能组件的实现,下面通过建立分布式 JobTracker 模型,希望能够在不分离监控和调度模块的情况下改进 JobTracker 节点的瓶颈和单点失效问题. 分布式 JobTracker 模型如图 2 所示.

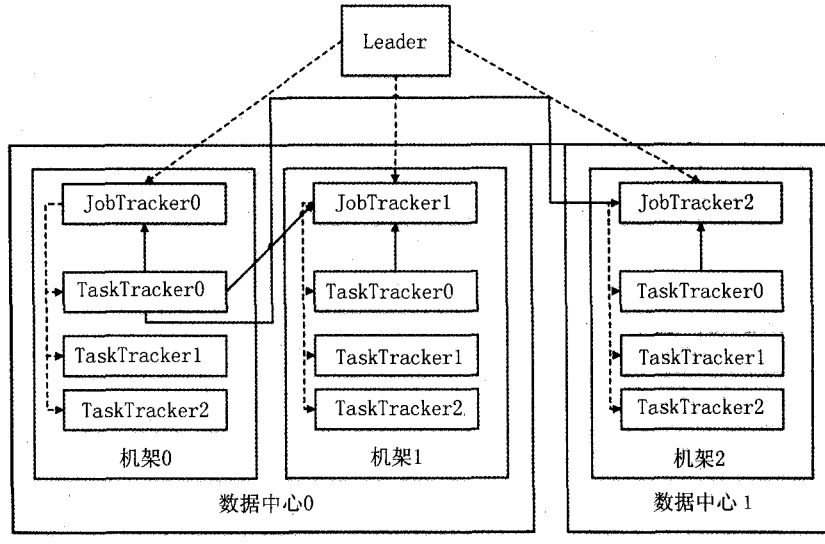

图2 分布式JobTracker节点模型

首先在每个本地机架上运行一个 JobTracker 的实例,通过本地的 JobTracker 对机架上的资源进行管理和任务调度,TaskTracker 节点对任务的需求首先依靠发送心跳信息传达给同一机架上的 JobTracker,从而减少了网络传输数据量,节省网络带宽,而且能较快收到反馈信息. 然后由一个高性能专用节点作为 Leader 节点来维护不同机架上的 JobTracker 节点,当某机架上的 JobTracker 节点失效后由 Leader 节点重新调度与失效节点相邻最近或负载最低的 JobTracker 来执行任务的分派功能. 由于分布式 JobTracker 模型控制节点的增多,在一定程度上增加了本地机架上控制节点的存储空间,相应的影响了本地任务槽的可用数量,然而却可以均衡单一 JobTracker 节点的负载,而且当 JobTracker 节点失效时无须再重启集群并放弃已经正在运行的任务,从而能够节省因节点失效而导致的作业完成时间的延长和计算资源的浪费. 若其中一台 JobTracker 节点宕机后,Leader 节点也失效,那么则根据下面 Dijkstra 算法最近一次更新后的结果优先调度离自己最近的 JobTracker 节点.

3 基于 Dijkstra 算法对模型通信方式的优化

3.1 算法思想

平时经常会遇到在很多限制下针对某问题去寻求最优解的情形. 有很多诸如"线性规划","动态规划"这些经典方法来寻求整体最优解,而在多对多节点模型构成的图形结构中,Dijkstra 算法是关于求"最短路径"中最优解算法中的一种典型方法. Dijkstra 算法是图中求两点之间最短距离的一种算法,引进了一个中间数组 D,数组中每个元素 $D[i]$代表目前能够找到的从起始节点 v 到任意终点 v_i 的距离,是从始发点到终点距离的相对最短,相对的含义是说在数组 D 中的元素是在逐步靠近最小值而中间过程中得到的不一定就是最短距离. MapReduce 模型本身的通信模型是主从式的树形机构,而建立了分布式 JobTracker 节点模型后,使得各从节点不再只向单一的控制节点请求任务,而是可以与多个 JobTracker 节点进行通信,因此模型结构由原来的一对多模型变成了多对多的图形结构. JobTracker 节点与 TaskTracker 节点之间采用了 pull 通信模型,即控制节点不是主动和任务请求节点进行通信,而是通过 TaskTracker 周期性地向 Jobracler 汇报信息并领取任务从而形成心跳通信[9]. 下面对 Dijkstra 算法做出了适用性的改进,目的是选出离 TaskTracker 最近的 3 个 JobTracker 节点进行通信,其中本地机架的 JobTracker 作为 Active Node 优先响应任务请求,建立通信的另外两个 JobTracker 作为 Standby Node 保存 TaskTracker 的元数据. 算法思想如图 3

所示.

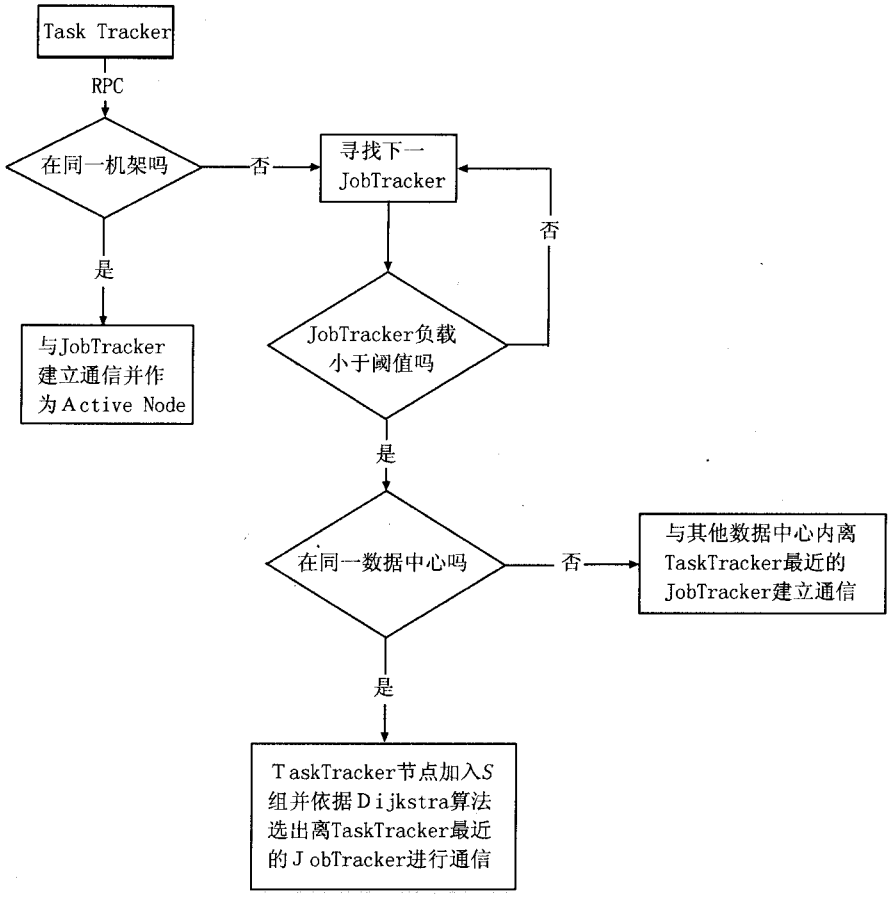

图3 基于Dijkstra算法的通信模型

基于 JobTracker 负载并按路径长度递增次序生成算法.

把 TaskTracker 节点集合 V 分成两组:

(1) S:已发出任务请求并已经与 3 个 JobTracker 节点建立了通信链接的 TaskTracker 集合(初始时只含有机架 0 中的 TaskTracker0).

(2) $V-S=T$:尚未发出任务请求的 TaskTracker 顶点集合.

把 T 中节点依次添加到 S 内,确保下面(1)、(2)成立.

(1)每个 JobTracker 顶点对应一个负载值.

(2)在不超过 JobTracker 节点负载阀值(初始定义为 0.75)的前提下,除节点所在机架外,再从本地数据中心其他机架和其他数据中心各选一个 JobTracker 与其进行通信,TaskTracker 所在机架的 JobTracker 对 TaskTracker 任务的请求优先响应,Standby Node 用于在 TaskTracker 所在机架的控制节点宕机时作为热备继续完成响应任务请求的工作[10]. 已经建立链接且负载低于阀值的 JobTracker 节点到 S 中各任务节点的距离都不大于从 JobTracker 到 T 中任何任务节点的路径距离.

S 中顶点:从 TaskTracker 到 JobTracker 的长度,由于一个机架定义一个分派任务的主节点,因此设同机架内的距离长度为 0,且长度随着机架的距离而逐步加 1. 则任务节点到其他 JobTracker 的距离就是 TaskTracker 到不同机架的距离.

T 中顶点:从 TaskTracker 到非本地机架 JobTracker 的距离只包括从 S 中顶点机架作为中间顶点的最短路径长度.

在图 3 中,TaskTracker 与本地机架的 JpbTracker 建立通信的时候并没有判断是否低于阀值,因为本

地机架 TaskTracker 数量有限,不会对 JobTracker 造成很大的负荷,只有当与不在同一机架的控制节点提出任务请求的时候才有可能引起其他机架 JobTracker 的负荷增加. 因而需要维护一个同一数据中心不同机架的距离最短表和一个不同数据中心的距离最短表. 因为 Yarn 中 NameNode 节点主要是用于存储任务节点的元数据信息,信息量相对比较大,而 MapReduce 中的 JobTracker 节点主要是用于任务的分派与调度和对资源的监控,元数据所占内存相对较少,但是对任务的响应时间要求相对较高[11]. 所以分布式 JobTracker 下基于 Dijkstra 算法的通信模型,在不高于主调节点阀值的前提下,不同于 Yarn 中 NameNode 侧重于选择负载最低的节点优先通信,而是优先与距离最近的 JobTracker 进行通信,侧重于节省跨机架和跨数据中心的通信时间.

4 实验验证与结果

实验条件采用 5 台计算机进行环境配置,其中一台配置为 CPU-I5-2. 7 GHz,内存 8 GB,SSD256 GB,其余 4 台配置为 CPU 四核 AMD-b40,主频 3. 0 GHz,内存 4 GB,机械硬盘 320 GB,实验平台采用 Ubuntu14. 04,实验系统采用开源的 Hadoop-1. 0. 0[12]. 实验数据通过 DataFactory5. 6 生成,大小为 1 GB,3 GB,6 GB,9 GB,12 GB. 作业用例采用 Hadoop 自带的经典 WordCount,根据进程 ID 通过杀死 JobTracker 进程来模拟 JobTracker 节点失效,假定这会影响 MapReduce 作业,但不会对 HDFS 文件系统造成影响[13]. 在单 JobTracker 模型中,客户端进程和从节点任务执行进程都要通过与仅有的 JobTracker 通信才能完成作业的调度和任务的分派,当单节点模型中的 JobTracker 停止运行,通信无法继续,作业就以失败告终. 随后执行了一些 HDFS 命令,可以发现运行异常的 MapReduce 集群不会对 HDFS 造成自己影响,所有的客户端都可访问 HDFS 并执行 HDFS 命令. 单点模型的 MapRedcue 集群的维护较为简单,只要重启 JobTracker,所有后续 MapReduce 作业都会成功运行.

通过图 4 可以看出,在分布式节点模型下若采用原有的通信方式,因为 TaskTracker 只和本地机架的 JobTracker 建立了通信,在控制节点不宕机的情况下,运行作业比较快,但是若把数据量增大到两倍到多倍,随着数据量的增大,发现基于 Dijkstra 的通信模型与原通信模型作业运行时间相差不是很大,但是当宕机时却对集群的安全性提供了更加可靠的保障. 适用于 HDFS 的基于负载优先的通信模型在集群初始启动数据量不大时低负载控制节点好确定,作业运行相对较快,但若宕机后再次找寻控制节点因为要广播发送多次比较才能确定负载最低的节点,在非本地机架的寻找过程中会相对浪费一定的时间[14].

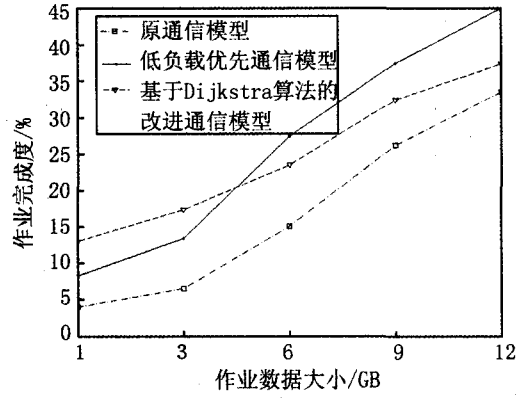

图4 原模型和低负载优先与改进的 Dijkstra通信模型

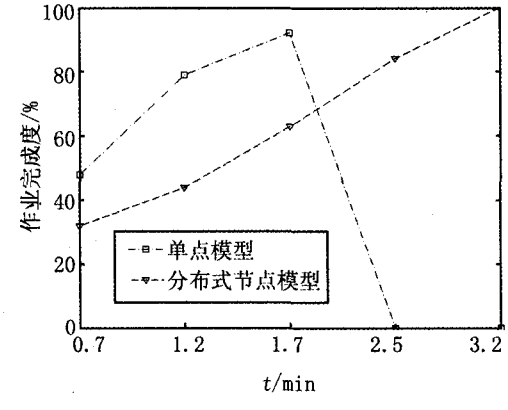

图5 单机的和分布式模型作业的运行

通过图 5 看以看出,在初始阶段,因为分布式模型下的 TaskTracker 要寻找 JobTracker 并和多个 JobTracker 进行通信,所以作业的运行时间相对单点模型比较慢,但是在单点模型中当运行作业的过程中遇到 JobTracker 进程意外结束,那么作业将被抛弃,因为没有节点再为 TaskTracker 节点分派所需任务和监视其资源的使用情况;而分布式 JobTracker 节点模型因为与 TaskTracker 通信的控制节点不止一个,所以能顺利的完成任务.

5　总结与展望

通过实验可以发现，分布式 JobTracker 模型的建立从作业处理的角度改进了单点可靠性的问题，虽然控制节点的增多占用了一定的空间和额外的通信时间，但是却能有效保障作业的顺利完成，不会因一个 JobTracker 节点的崩溃而抛弃快要执行完成的作业. 基于 Dijkstra 算法的分布式通信模型有助于均衡任务的负载，避免单 JobTracker 节点随着请求任务的增多而造成的单点瓶颈，并能在一定程度上减少响应 TaskTracker 节点任务请求的控制节点数量，缩短节点间的通信路径从而减少通信时间.

参　考　文　献

[1] 罗军舟，金嘉晖，宋爱波，等. 云计算：体系架构与关键技术[J]. 通信学报，2011，32(7)：3-20.

[2] 邓　鹏，李枚毅，何　诚. NameNode 单点故障解决方案研究[J]. 计算机工程，2012，21(7)：78-83.

[3] Cho Joong-Yeon, Jin Hyun-Wook, Lee Min, et al. Dynamic core affinity for high-performance file upload on Hadoop Distributed File System[J]. Parellel Computing, 2014, 40: 722-737.

[4] 吴方超. 基于分布式 Pi 演算的 NameNode 单点故障研究与实现[D]. 西安：西安电子科技大学，2014.

[5] Cho J, Jin H W, Lee M, et al. On the core affinity and file upload performance of Hadoop[C]. Proceedings of the International Workshop on Data-Intensive Scalable Computing System, Denver, 2013.

[6] Polo J, Carrera D, Becerra Y. Performance-driven task co-scheduling for MapReduce environments [C]. In proceedings of the 2010 IEEE/IFIP Network Operations and Management Symposium-NOMS 2010, Osaka, 2010.

[7] 金伟健，王春枝. 基于匹配规则的 MapReduce 任务调度模型[J]. 计算机应用，2014，34(4)：1010-1013.

[8] 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)：239-243.

[9] 乔媛媛，刘　芳，凌　艳，等. 云计算环境下 MapReduce 的资源建模与性能预测[J]. 北京邮电大学学报，2014，37：115-119.

[10] 陈　波，沈　炜. 基于 HDFS 的动态副本策略设计与实现[J]. 工业控制计算机，2015，28(1)：103-105.

[11] Aysan Rasooli, Douglas G. Guidelines for Selecting Hadoop Schedulers Based on System Heterogeneity[J]. J Grid Computing, 2014, 12: 499-519.

[12] Apache. Apache hadoop[EB/OL]. [2015-12-18]. http://hadoop.apache.org/releases.html.

[13] Tom White. Hadoop：权威指南[M]. 2 版. 周敏奇，王晓玲，金澈清，等译. 北京：清华大学出版社，2011.

[14] 冯国富，王　明，李　亮，等. 共享存储 MapReduce 云计算性能测试方法[J]. 计算机工程，2013，36(6)：50-52.

Based on the Dijkstra Algorithm Optimize the Distributed JobTracker Node Model of Communication

HUANG Weijian, YANG Hailong

(School of Information & Electrical Engineering, Hebei University of Engineering, Handan 056038, China)

Abstract: For the faliure of single JobTracker which may cause the running crashes of system under MapReduce framework, establishing a model of distributed JobTracker node can improve the reliability of JobTracker, and through optimizing and improving communication based on the idea of single-source shortest path Dijkstra algorithm, task nodes can communicate with a JobTracker priority which is in the nearest rack or in the nearest data center. Experimental results have shown that the model of distributed JobTracker node can alleviate problems caused by the failutre of JobTracker to some extent. In addition, improved communication can also alleviate bottlenecks of single JobTracker and balance the load of JobTracker.

Keywords: Hadoop; mapreduce; communication mode; dijkstra; JobTracker; tasktracker