基于 MapReduce 数据密集型负载调度策略研究

秦 军[1],童 毅[2],戴新华[2],林巧民[1]
(1. 南京邮电大学 教育科学与技术学院,江苏 南京 210003;
2. 南京邮电大学 计算机学院,江苏 南京 210003)

摘 要:针对云计算环境中大规模数据集的处理,MapReduce 集群已成为一个强大的处理平台。文中提出了一种基于虚拟化平台动态资源重配置的资源评价和动态资源重新配置调度算法。该算法动态地评估作业在截止时间内完成所需要的 Map 和 Reduce 计算资源数量,并在不违反用户设定的时间目标的情况下,通过动态地增加或减少独立虚拟机的方式来调整 CPU 资源,以实现提高数据本地性,同时提高系统在运行作业时的资源利用率。仿真实验结果表明,该算法可以使集群上的 MapReduce 作业的吞吐率有明显的提高。

关键词:云计算;数据本地性;MapReduce;Hadoop
中图分类号:TP301.6 **文献标识码**:A **文章编号**:1673-629X(2015)04-0048-05
doi:10.3969/j.issn.1673-629X.2015.04.012

Research on Scheduling Strategy of Data Intensive Workloads Based on MapReduce

QIN Jun[1],TONG Yi[2],DAI Xin-hua[2],LIN Qiao-min[1]
(1. College of Education Science & Technology,Nanjing University of Posts and Telecommunications,Nanjing 210003,China;
2. College of Computer,Nanjing University of Posts and Telecommunications,Nanjing 210003,China)

Abstract:Aiming at processing of large-scale data set in cloud computing environment,MapReduce has become a powerful processing platform. In this paper,propose a resource evaluation and dynamic resource reconfiguration and scheduling algorithm based on virtualization platform dynamic resource reconfiguration. It can dynamically evaluate the required number of Map/Reduce slots for every job to meet completion time guarantee and adjust the CPU resources while not violating completion time goals of the users by dynamically increasing or decreasing individual VMs to maximize data locality and also to maximize the use of resources within the system among the active jobs. Simulation results show that the algorithm can improve the throughput of MapReduce jobs on the cluster significantly.

Key words:cloud computing;data locality;MapReduce;Hadoop

0 引 言

现如今,云计算正在一些分布式平台上提供更多面向数据密集型计算的服务,这些平台包括 Dryad、Hadoop[1-2]等。在这些云的分布式平台上,物理机器是被虚拟化的,种类繁多的虚拟机形成虚拟机群,MapReduce 作业的完成时间目标,对于云服务提供商能否获得高额利润和提高资源利用率有重要影响;根据任务是否被分配到含有输入数据的节点(本地任务或非本地任务),任务的运行时间可能会相差很大,以往的研究表明,数据的本地性对 Hadoop 作业的吞吐量影响相当大,即使其他因素也会对 MapReduce 作业的性能产生影响[3]。因此,为了提高数据本地性,当一个任务被调度时,存储相应数据的计算节点应该有空闲的计算资源分配给该任务;否则任务就会被分配到远端节点,这样就需要额外从其他节点传输该任务的输入数据。所以,需要解决的问题就是如何有效地调度作业,来同时满足数据本地性和截止期限要求。

文中就上述问题提出一种新的调度算法,动态地评估满足作业期限要求所需的资源和最大限度地提高数据本地性。算法首先提出评估每个作业在截止时间

收稿日期:2014-06-15 **修回日期**:2014-09-23 **网络出版时间**:2015-02-23
基金项目:江苏省自然科学基金项目(BK20130882)
作者简介:秦 军(1955-),女,教授,研究方向为计算机网络技术、多媒体技术、数据库技术。
网络出版地址:http://www.cnki.net/kcms/detail/61.1450.TP.20150223.1252.052.html

内完成所需资源的模型,作业的特征通过运行该作业的一些初始任务集取得,根据这些任务特性,再结合作业的截止时间,评估和适当分配在截止时间前完成作业所需的 MapReduce 资源,随着作业的运行和截止时间的接近,能够重新计算任务的所需资源数以满足截止时间;另一方面通过动态重配置资源来实现在虚拟资源中数据本地性的最大化。

1　相关概念

MapReduce 通过将海量作业用 Map 函数分割为众多子任务,而各个子任务交由集群中不同的计算节点并行处理,处理后的中间结果通过 Reduce 函数进行简单合并就可以得到最终结果。

1.1　MapReduce

MapReduce 编程模型由 2 个数据处理功能组成[4]:Map 和 Reduce。在 Map 阶段,输入数据被拆分成固定大小的块并交由 Map 函数进行处理,输出键值对的组合集作为中间结果,Map 任务在每个记录上面实现用户定义动作再缓存随后的中间结果,这些中间数据是基于键值的哈希分区的并被写入运行 Map 任务所在节点的本地磁盘。每个 Reduce 任务首先把 Map 阶段的输出数据复制到 Reduce 节点,其次把复制过来的数据根据键值进行排序,最后进行 Reduce 操作。

作业是任务的集合,Hadoop 中的作业调度是由 Job Master(Job Tracker)通过管理集群中的工作节点(Task Tracker)实现,每个工作节点都有一定数量的 Map 和 Reduce 计算资源运行任务。不同的调度策略采用不同方法提高 MapReduce 作业的性能[5],Map 和 Reduce 资源数目一般都是静态设置的,工作节点会定期向控制节点发心跳信息报告空闲资源的数目和当前运行任务的进度,基于可用资源的情况和调度策略,控制节点可以分配任务给那些有空闲资源的节点,同时 HDFS(Hadoop Distributed File System)也通过复制和存储 Hadoop 作业的输入输出以保证 MapReduce 计算的可靠性和容错性。

FIFO 算法是 Hadoop 的默认调度策略。在 FIFO 调度中,该算法的 JobTracker 从工作队列中拉取作业,最早的作业最先运行。这种调度方法不会考虑作业的优先级,但很容易实现,而且效率较高;公平调度[6]的核心概念是,随着时间推移平均分配资源,这样每个作业都能平均地共享到资源。当只有一个作业在运行时,它将使用整个资源。当有其他作业同时运行时,系统会将任务空闲时间片分配新的作业,以使得每一个作业都大概获取到等量的 CPU 时间;容量调度的原理与公平调度有些相似,首先容量调度是用于大型集群,它们有多个独立用户和目标应用程序,容量调度能提供更大的控制和性能保证,提供用户之间最小容量保证并在用户之间共享多余的容量。

1.2　资源估算模型

对于需要服务质量保证的用户,如果处理数据规模为 Ω 的 MapReduce 作业 J,需要在时间 D 内完成,而这个作业是部署在拥有 N_m 个 Map 资源和 N_r 个 Reduce 资源的节点集群上,同时还有 K 个其他作业在运行,这里就必须考虑在时间 D 内完成需要分配多少计算资源给作业 J。

MapReduce 作业资源估算模型在作业的执行过程中动态地估算作业的完成时间,并以此满足基于 Hadoop 的数据运算中最后截止期限的要求。这是因为 MapReduce 作业是大量的小型任务的集合,即可以根据一些已经完成的任务的情况来评估剩下的任务,MapReduce 任务在执行特性上的差异是因为处理不同的数据集。因此,这个估算方式并不总是能够提供准确预测,但与动态调度相结合就可以公平地在截止期限内完成多个任务。文中提出的调度算法采用完成时间作为评估作业所需资源的依据,资源分配的最小单位是槽位(slot),对应于 TaskTracker 创建的工作进程。

MapReduce 作业评估模型中所用符号说明如下:

D:作业的完成截止时间;

J:作业集合;

n_m:作业分配到的 Map 资源数量;

n_r:作业分配到的 Reduce 资源数量;

u_m:作业的 Map 任务数量;

u_r:作业的 Reduce 任务数量;

t_m:完成一个 Map 任务所需时间;

d_m:Map 任务完成所需的剩余时间;

e_m:Map 任务已经运行的时间;

t_r:完成一个 Reduce 任务所需时间;

t_s:在排序阶段复制操作所需时间;

C_j:作业已经完成的 Map 任务集合;

U_j:作业还没开始的任务集合;

R_j:作业正在进行的任务集合;

S_{rq}:释放队列里以降序排列的所有保存运行任务 t_i 所需数据的节点集合;

S_{aq}:分配队列里以升序排列的所有保存运行任务 t_i 所需数据的节点集合。

每个 TaskTracker 都有一定数量的槽位分别分配给 Map 任务和 Reduce 任务,一般来说 MapReduce 应用的完成时间主要是由 Map 任务的完成时间决定,也有一些是由 Reduce 任务决定,在两种情况下,作业都是先由 Map 阶段开始,然后收集已运行任务性能数据,再开始 Reduce 过程。调度器不能在没有分析已完成

Reduce 任务的情况下对剩余其他 Reduce 任务进行假设,因此,由于缺少前次执行的信息,调度器需要把 Reduce 阶段和 Map 阶段进行对比评估,从而可以在 Reduce 阶段开始前预测整个作业的完成时间。当有完成时间目标的作业被提交时,没有数据可以用来评估所需资源数量,下面是计算作业的单个任务完成时间的方法。

作业的 Map 任务的平均完成时间:

$$m_m = \frac{1}{C_j}\sum_{m \in C_j} t_m \tag{1}$$

假设集群的所有节点都是相似的,再假设 Map 任务和 Reduce 任务的完成时间也相同,因此

$$t_m = t_r \tag{2}$$

因为作业包括 u_m 个 Map 任务和 u_r 个 Reduce 任务,所以 Map 阶段的时间可以表示为 $\frac{u_m \times t_m}{n_m}$。

同理,Reduce 阶段的时间为 $\frac{u_r \times t_r}{n_r}$。

拥有 u_m 个 Map 任务和 u_r 个 Reduce 任务的作业,因为每个 Map 都有可能输出 Reduce 所需的中间结果,所以还要有 $(u_m \times u_r)$ 次复制操作,所需时间为 $(u_m \times u_r) \times t_s$。这样,整个作业的完成时间为:

$$\frac{u_m \times t_m}{n_m} + \frac{u_r \times t_r}{n_r} + (u_m \times u_r) \times t_s \leqslant D \tag{3}$$

也即:

$$\frac{u_m \times t_m}{n_m} + \frac{u_r \times t_r}{n_r} \leqslant D - (u_m \times u_r) \times t_s \tag{4}$$

再令 $A = u_m \times t_m$;$B = u_r \times t_r$;$C = D - (u_m \times u_r) \times t_s$;则

$$n_m = \frac{\sqrt{A}(\sqrt{A} + \sqrt{B})}{C}, n_r = \frac{\sqrt{B}(\sqrt{A} + \sqrt{B})}{C} \tag{5}$$

式中,n_m 和 n_r 是作业在截止时间内完成所需最少的 Map 和 Reduce 资源数量。

2 算法实现过程

在虚拟化的物理集群上,需要一种新的调度算法,该算法可以有效调度虚拟化空闲资源,实现任务本地性的最大化,提高云平台的运算效率。

2.1 资源重配置

部署在物理集群上的传统数据密集型计算不能动态地改变集群资源数量[7],每个物理集群都有固定数量的计算资源,然而,如果在物理集群上建立虚拟集群的话,就可以通过虚拟化实现虚拟机器的重配置[8]。运行在可以提供高可扩展环境的虚拟集群上的数据密集型云平台,例如 Hadoop,能根据用户的需求提供不同的计算选择,云提供商会把不同用户所需的虚拟集群整合到一个数据中心,以实现资源利用的最大化[9]。

当前用于数据密集型计算的虚拟集群已经可以实现在经过配置的集群中灵活地选择计算节点的类型和数量,用户可以选择最合适的集群配置,以满足他们的计算需求。尽管集群中的虚拟机经过这样的静态配置可以实现比物理机器更好的灵活性,但仍不能满足虚拟集群整个生命周期内动态变化的计算需求。为了适应集群中虚拟机不断变化的需求,每个虚拟机可以动态地重新配置,只要物理资源可用,每个虚拟机就可以在运行时使用更多的 CPU 和内存[10]。然而,在当前可用的云服务中,并没有实现这种对虚拟机的动态重配置。配置管理器(CM)发送分配和释放请求给每个物理机器上运行的虚拟机管理程序,每个虚拟集群都有一个 CM 并且由集群的主节点运行 CM,物理机器上的虚拟机如果需要资源重配置,CM 就会发送分配和释放请求给运行在物理系统上的机器管理器(MM),从 CM 收到请求后,MM 会把所请求的虚拟 CPU 分配给虚拟机。在基于资源重配置的 Map 任务分配算法 1 中,为了最大限度实现数据本地性,一个任务在目标节点有可用的资源前不会被执行,在资源和虚拟机中释放与分配可用资源的方式是不一样的,如果虚拟机有空闲资源,会把这个资源加入系统的释放队列(RQ),如果虚拟机有待运行的本地任务,该任务会被附加到系统的分配队列(AQ),只有 RQ 和 AQ 同时非空,虚拟资源的重配置才会发生,释放一个虚拟机的资源,再分配给系统内的另一虚拟机。

基于资源重配置的 Map 任务分配算法流程图如图 1 所示。

这种调度策略的存在是因为 CPU 不能超越物理系统的边界直接转移,即使运行源虚拟机的系统有空闲的资源,该计算资源也不能直接提供给目标虚拟机;然而,随着多个虚拟机共享同一个物理系统,当一台虚拟机完成一个任务且没有待执行的本地任务时,目标虚拟机很快就会得到空闲资源。在这种方式下,排队延迟是影响集群性能的重要因素,因为如果对分配和释放队列请求的处理由于大量排队延迟而推迟的话,资源的利用率也可能降低。

2.2 基于完成时间的调度策略

根据上面提到的作业资源评估模型和基于资源重配置的 Map 任务分配,文中提出了资源评价和动态资源重新配置调度算法(Scheduler Algorithm based on Resource Estimation and Dynamic Resource Reconfiguration, SAREDRR)。当作业被提交的时候,没有数据用于评估所需的资源数目和预计完成时间,因此,那些有未完成任务或正在运行任务的作业总是优先于其他作业,如果有多个这样的任务,最早的作业会先执行。

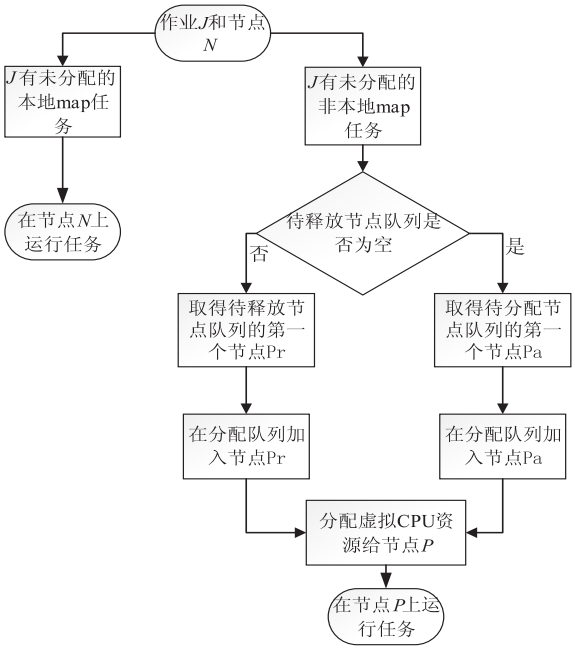


图 1　基于资源重配置的 Map 任务分配算法流程图

SAREDRR 首要的部分就是评估在截止期限内完成作业所需的最少资源数量,最初,由于没有作业的相关性能数据,作业的一些任务会单独运行以获取任务的预估完成时间,然后估计作业在截止期限内完成所需的最少资源数量。TaskTracker 节点会定期更新自己的状态给 JobTracker,或在任务完成时利用心跳机制进行通信,通常心跳间隔是 3 s。由于不同输入拆分的多个 Map 任务可并行执行,数据本地性在 Map 阶段是较为关键的,相反,在 Reduce 阶段,输入数据来自于所有的运行 Map 作业节点,数据本地性就没有那么重要。在调度中只考虑 Map 阶段的数据本地性最大化。

SAREDRR 可以根据作业的一些任务的执行情况决定是否需要继续执行完该作业,如果正运行任务所需的资源数量超过当前可用资源,下个作业会进入调度序列。虚拟机的配置可以在集群资源不变的情况下动态地增减虚拟 CPU 的数量,这样实现数据本地性的最大化。

基于完成时间的调度算法实现步骤描述如下:

步骤 1:作业 J 进入调度,根据式(5) 计算完成作业 J 所需的最少 Map 和 Reduce 资源数量 (N_m,N_r)。

步骤 2:根据收到的节点 N 心跳信息判断,若 N 无空闲计算资源,则结束调度。

步骤 3:若作业 J 有未完成 Map 任务且当前被调度 Map 任务数少于 N_m,在节点 N 上运行 Map 任务。

步骤 4:若作业 J 无未完成 Map 任务且当前被调度 Reduce 任务数少于 N_r,在节点 N 上运行 Reduce 任务。

步骤 5:统计已完成任务数量,判断作业 J 是否完成,若完成,则结束调度。

步骤 6:根据作业 J 的已完成任务情况,根据式(5) 计算完成作业 J 所需的最少 Map 和 Reduce 资源数量,重复步骤 2 ~ 5。

3　仿真实验结果与分析

通过仿真实验评估该调度算法相对于 Hadoop 公平调度算法的性能表现[11]。实验选择了 5 种不同的 MapReduce 作业,模拟一个建立在拥有 20 台物理机器的集群上的数据中心环境,每个节点配置成有 2 个 Map 接口资源和 2 个 Reduce 接口资源,Xen[12] 用于物理机的虚拟化配置,Xen 的调度器用来实现虚拟 CPU 的动态分配,在 Hadoop 上执行 MapReduce 任务。通过对 5 种作业在不同大小输入数据下的执行情况的对比,可以得到:

(1)字数统计:采用一组文本文件作为输入,再统计每个单词出现的次数。

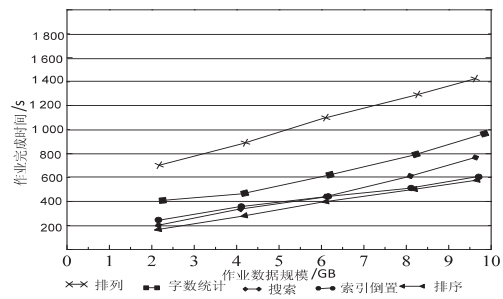
(2)排序:对一组随机生成的记录进行排序。

(3)搜索:简单地判断一个单词是否在输入数据中出现。

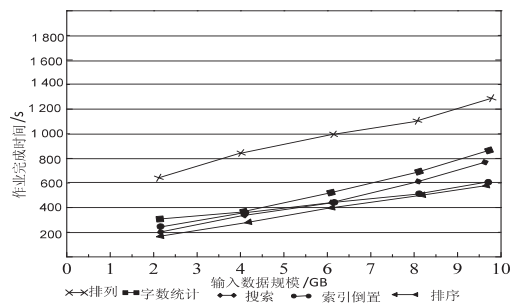
(4)排列组合:从输入字符串里整合排列组合。

(5)倒排索引:生成单词的序列和包含单词的文档列表,所有输出对形成简单的倒置索引。

上述所有的作业都是分别在 2 GB,4 GB,6 GB,8 GB,10 GB 大小的输入数据的情况下执行的,在相同规模的输入数据下,分别调用公平算法和 SAREDRR 算法,结果如图 2 所示。



(a)调用公平调度



(b)调用 SAREDRR

图 2　在不同的输入数据规模下的作业完成时间

从图 2 可见,随着输入数据的规模增大,作业的执行时间也会变长,由于不同的负载对资源的要求也不

同,完成时间随着负载的改变而改变。例如排列组合的作业会产生大量的中间数据,这需要很多的复制操作,因而相对其他作业的完成时间更长。

另一个实验,输入数据大小是随机的,在任务开始提交时,没有可用的数据来通过公式(5)确定最少所需的计算资源。表 1 给出了作业的截止期限以及在截止期内完成作业所需的最少资源数量。

表 1 各类型作业在满足截止时间的情况下所需的计算资源

作业类型	作业完成时间/s	输入数据规模/GB	满足截止时间的计算资源分配	
			Map 资源	Reduce 资源
单词搜索	640	10	25	9
单词统计	510	5	15	8
排序	520	10	21	12
排列组合	840	4	16	15
索引倒置	710	8	13	9

分别使用公平调度和 SAREDRR 算法执行这些作业,执行结果见图 3。

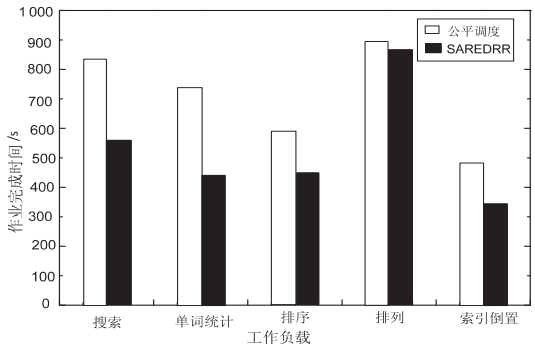


图 3 基于作业完成时间的比较

由结果可以得出这样的结论,SAREDRR 算法通过执行更多的本地任务减少作业的完成时间,从图 3 可以明显看到排列作业在两种调度下的完成时间几乎相同,因为该作业在 Reduce 阶段要做大量的复制操作,在 Reduce 阶段的输入数据是从所有的 Mapper 复制到 Reducer,这时数据本地化没那么重要,运行本地化任务也就没有意义[13]。

对于其他的作业,SAREDRR 算法明显地减少了作业的完成时间。由于作业都是同时运行的,很可能虚拟机没有空闲资源运行本地任务,因此任务需要一定的等待时间,这个等待时间也要在评估作业的吞吐量时进行考量[14]。在该实验中,观察到这个时间可以忽略,因为 MapReduce 作业的任务可以在不到一分钟的时间内完成,这样虚拟机就可以释放资源分配给其他任务。相对比公平调度,SAREDRR 算法可以使集群上的 MapReduce 作业的吞吐率大约提高 12% 左右。

4 结束语

很多部署在云上的应用程序都有完成目标,例如截止期限等。文中提出并评估了一种资源评价和动态资源重新配置调度算法,该算法扩展了实时集群的调度方法,在运行作业时根据截止期限等限制条件来评估所需的最少资源数量。通过在虚拟集群中临时增加虚拟机内核来运行本地任务的方式,提高了数据的本地性,经过虚拟机的重配置,每个节点都能获得所需计算资源。仿真实验结果表明,该算法提高了作业的吞吐率。

参考文献:

[1] Apache Hadoop[EB/OL]. 2012-04-16. http://hadoop.apache.org.

[2] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters[C]//Proc of sixth symposium on operating system design and implementation. Berkeley: USENIX Association, 2004: 124-151.

[3] 朱 珠. 基于 Hadoop 的海量数据处理模型研究和应用[D]. 北京:北京邮电大学,2008.

[4] Lammel R. Google's MapReduce programming model-revisited[J]. Science of Computer Programming, 2008, 70(1): 1-30.

[5] Zaharia M, Borthakur D, Sarma J S. Job scheduling for multi-user MapReduce cluster[C]//Proceedings of the 5th European conference. Washington: IEEE Computer Society, 2009: 145-161.

[6] Hadoop 公平调度算法[EB/OL], 2010-02-19. http://hadoop.apache.org/docs/r0.20.2/fair_scheduler.html.

[7] Buyya R, Ranjan R, Calheiros R N. Modeling and simulation of scalable cloud computing environments and the CloudSim-Toolkit: challenge and opportunities[C]//Proc of the 7th high performance computing and simulation conference. Leipzig: IEEE, 2009.

[8] 江 雪,李小勇. 虚拟机动态迁移的研究[J]. 计算机应用, 2008, 28(9): 2375-2377.

[9] Susanta N. A survey on virtualization technologies[EB/OL]. 2011-06-20. http://www.ecsl.cs.sunysb.edu/tr/TR179.PDF.

[10] Anton B, Rajkumar B. Energy efficient resource management in virtualized cloud data center[C]//Proc of IEEE/ACM international conference on cluster, cloud and grid computing. Melbourne, Australia: IEEE Computer Society, 2010.

[11] CloudSim[EB/OL]. 2012-02-11. http://www.cloudbus.org/cloudsim/.

[12] 胡冷非. 虚拟机 Xen 网络带宽分配的研究和改进[D]. 上海:上海交通大学,2009.

[13] 秦 军,张建平,王 昊,等. 基于蚁群优化算法的 MapReduce 集群调度策略[J]. 计算机技术与发展, 2013, 23(6): 74-78.

[14] 郝树魁. Hadoop HDFS 和 MapReduce 架构浅析[J]. 邮电设计技术, 2012(7): 37-42.

基于MapReduce数据密集型负载调度策略研究

作者： 秦军， 童毅， 戴新华， 林巧民， QIN Jun， TONG Yi， DAI Xin-hua， LIN Qiao-min

作者单位： 秦军,林巧民,QIN Jun,LIN Qiao-min(南京邮电大学 教育科学与技术学院,江苏 南京,210003)， 童毅,戴新华,TONG Yi,DAI Xin-hua(南京邮电大学 计算机学院,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)： 2015(4)