面向请求预处理的实时按需数据广播调度方法

胡文斌 邱振宇 聂 聪 王 欢 严丽平 杜 博

(武汉大学计算机学院 武汉 430072)

摘 要 随着无线技术以及移动计算的飞速发展,实时按需数据广播被广泛应用于一些高度动态的广播环境.用户请求数量的剧增导致上行信道传输的数据量以及服务器每秒响应次数的激增,这使得上行信道带宽成为广播系统的瓶颈,也使得数据服务器面临巨大的并发压力.目前的研究成果少有在考虑上行带宽和服务器压力的同时能保证良好的广播效率.基于此,该文研究了实时按需数据广播请求预处理问题,并且提出了一种请求预处理方法(Request Pre-Process Method,RPPM)以减轻上行信道及服务器并发压力,保证系统广播效率.RPPM 包含如下两个算法:(1)针对最大化请求合并问题(Maximum Request Merge problem,MRM),该文提出一种最佳请求合并算法(Optimal Request Merge algorithm,ORM)以合并用户请求,减少请求数量,降低上行信道及服务器压力;(2)针对合并请求的最佳合并请求优先级问题(Optimal Merged Request Priority problem,OMRP),该文提出一种合并请求优先级和剪枝算法(Merged Request Priority and Prune algorithm,MRPP)以综合衡量合并请求优先级,剪枝多余合并请求.基于 RPPM,该文提出了一种基于请求预处理的实时按需数据广播全局调度方案以进一步提高系统广播效率.该文以请求失效率(Loss Rate,*LR*)和平均访问时间(Average Access Time,*AAT*)作为评价指标,将 RPPM 同其他最新的调度算法进行了对比实验.同时该文定义请求合并率(Request Merge Rate,*RMR*)以量化分析 RPPM 对于上行信道及服务器压力减少的贡献值.大量实验结果表明,RPPM 在取得与最新调度算法接近的 *LR* 和 *AAT* 的同时,能够提高约 50%的 *RMR*,即减少约 50%的用户请求数量,缓解上行信道以及服务器压力.

关键词 按需数据广播;请求预处理;请求合并;请求优先级;调度

中图法分类号 TP18 **DOI 号** 10.11897/SP.J.1016.2018.02060

A Request Pre-Process Method for Real-Time and On-Demand Data Broadcast Scheduling Title

HU Wen-Bin QIU Zhen-Yu NIE Cong WANG Huan YAN Li-Ping DU Bo

(*School of Computer Science, Wuhan University, Wuhan* 430072)

Abstract With the rapid development of wireless technology and mobile computing, there is an increasing need for scalable wireless data dissemination techniques that can concurrently deliver data to a large number of users. On-demand data broadcast has the advantage of channel sharing technology to allow an arbitrary number of clients to access data simultaneously and has become the preferred method to dispense data in certain highly dynamic environments, where a large number of users are interested in hot information such as news reports, stock quotes, flight schedules, or traffic reports. However, with the rapid increase in client requests, the data that need to be transmitted by an uplink channel of broadcast system and requests that need to be responded to via the server increase rapidly, which leads to the uplink bandwidth becoming a bottleneck and the server facing huge parallel processing pressure. On one hand, restricted by

收稿日期:2017-02-26;在线出版日期:2017-12-20.本课题得到国家自然科学基金(61572369,61711530238)、湖北省自然科学基金(2015CFB423)、武汉市重大科技计划项目(2015010101010023)资助.**胡文斌**,男,1976 年生,博士,教授,博士生导师,主要研究领域为复杂网络、人工智能、调度优化.E-mail:hwb@whu.edu.cn.**邱振宇**,男,1992 年生,博士研究生,主要研究方向为复杂网络、数据广播.**聂 聪**,男,1994 年生,硕士研究生,主要研究方向为智能交通、数据挖掘.**王 欢**,男,1989 年生,博士研究生,主要研究方向为社会网络分析、数据挖掘.**严丽萍**,女,1980 年生,博士研究生,主要研究方向为社会网络分析、数据挖掘.**杜 博**,男,1983 年生,博士,教授,博士生导师,主要研究领域为复杂网络、社会网络分析.

mobile terminal, the bandwidth between a server and a client is asymmetric, the uplink bandwidth is much smaller than the downlink. With the increase of user request, the uplink bandwidth becomes a bottleneck in on-demand data broadcast. On the other hand, a large number of client access the server simultaneously also put a great challenge of parallel processing ability to the server. So far, no scheduling algorithms have been presented that can balance the pressure of the uplink channel and server and the broadcast efficiency. Thus, we investigate the problem of request pre-process and develop a request pre-process method (RPPM) for on-demand data broadcast to reduce the pressure on the uplink channel and server. RPPM integrated request pre-process and data broadcast scheduling to improve the overall performance of the on-demand data broadcast system. Our method includes two algorithms: an optimal request merge algorithm (ORM) for the maximum request merge problem (MRM) to reduce the number of requests, ORM reduce the number of requests by merging requests, and achieve a tradeoff between invalid data rate and reduce of request rate; a merged request priority and prune algorithm (MRPP) for the optimal merged request priority problem (OMRP) to evaluate the priority of merged requests and prune them, MRPP synthetically takes into account the number of requests include in the merged request, the longest waiting time and the number of lost request of the merged request not be responded during the current time. MRPP can accurately measure the priority of the data to ensure that the most urgent data first broadcast. Based on RPPM, we develop a scheduling scheme for an overall on-demand data broadcast schedule to further improve the broadcast efficiency. We compare the proposed method with other state-of-the-art scheduling methods with the general performance metrics: the request loss rate (LR) and the average access time (AAT). We present the request merge rate (RMR) for quantitative analysis of the reduction in uplink channel and server pressures. The results reveal that RPPM can significantly reduce the number of requests with a high RMR and can obtain similar AAT and LR to the other state-of-the-art algorithms.

Keywords　on-demand broadcast; requests pre-process; request merge; request priority; scheduling scheme

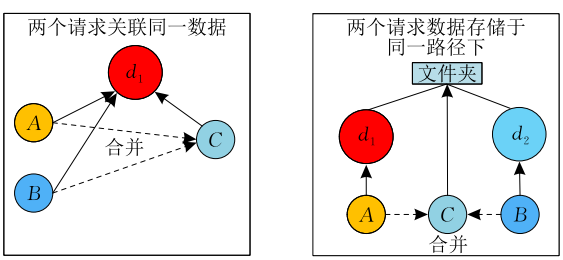
1　引　言

随着 4G 技术成熟、5G 技术的迅速发展，无线网络与移动计算不断地融入到我们日常生活中. 大量的移动数据服务迫切需要一种可扩展的无线传播方法，以实现大规模数据的分发. 无线数据广播正是这样一种方法，其利用信道共享技术，可以实现任意数量用户同时访问数据[1]，非常适用于高负载系统(如股票信息发布系统、实时交通信息发布系统)的信息传输[2-3]. 根据在广播过程中是否考虑实时用户请求，数据广播可分为两类[4]：静态数据广播[5](push-based)和实时按需数据广播[6](pull-based). 静态数据广播依赖于数据访问模式的先验知识，周期性地以静态的方式广播预定的数据项[7]，广播过程中不考虑实时用户需求. 相反，实时按需数据广播通过上行信道接收实时用户请求，依据用户请求动态组织数据广播. 实时按需数据广播兼具点对点通信的灵活性，以及数据广播的高效性，可扩展到大规模实时数据库，可适应动态数据访问模式以满足实时需求，越来越受到学者的关注[8-9].

在按需数据广播系统中，用户通过上行信道上传请求，服务器响应请求，动态组织被请求数据广播. 现有文献关于按需数据广播的研究主要集中在数据调度算法[10-11]，然而，随着用户请求量的激增，因大规模数据请求导致的问题不容忽视. 一方面受限于移动终端，按需数据广播系统中移动终端和服务器之间的上下信道带宽存在不对称性，即上行信道带宽远小于下行信道带宽. 随着用户请求数的不断增加，上行信道带宽成为按需数据广播系统的瓶颈. 另一方面，大量用户同时请求数据，服务器必须在极短的时间内响应，这也是对服务器并发处理请求能力的巨大挑战. 例如，在股票更新期间，大量用户请求同时涌向股票服务器以获取最新的股票信

息，将消耗大量上行带宽，导致极高的服务器并发量，严重时甚至会导致服务器崩溃.这些问题激励我们研究按需数据广播系统中用户请求预处理问题.本文讨论的请求预处理问题主要包含两个方面：请求合并和合并请求优先级量化，研究成果可应用于许多系统.例如，在2016年里约奥运会期间，几乎所有用户都对最新的比赛结果、奖牌榜等感兴趣，一个带有请求预处理技术的实时按需数据广播系统可以在保证上行信道带宽和服务器正常运行的情况下第一时间满足用户更新最新信息的需求.

在真实的按需数据广播系统中，用户倾向于获取热点信息，即存在大量用户同时请求同一数据项的情况，这些请求可能仅仅是用户信息不同，并不影响广播调度.然而在传统的按需调度过程中，上行信道需要上传每一个请求，服务器需要对每一个请求作出响应，这是对系统资源的极大浪费.如图1(a)所示，两个请求A、B请求数据项d_1，请求A、B特定信息(截止期、请求时间)不同.若能找到合理的方法将请求A、B合并为请求C，使得合并请求C能够同时满足请求A、B，并准确地量化合并请求C的优先级，请求数将由原来的2减小到1且不会导致任何请求可满足性受到影响.此外，大多数服务器提供给移动终端访问的目标数据项以分层目录树方式组织[12]，这为同一目录下的请求合并提供了可能性.如图1(b)所示，请求A指向数据项d_1，请求B指向数据项d_2，数据项d_1、d_2存储于同一文件夹下.可将请求A、B合并为指向文件夹的请求C，即请求C请求该文件夹下所有的数据项.基于以上分析，本文致力于提出一种有效的基于请求预处理的实时按需数据广播调度方法，对用户请求进行合并、优先级量化以及合并请求剪枝操作，从而达到解决上行带宽瓶颈，缓解服务器并发压力，保证广播效率的目的.



(a) 合并同一数据的请求 (b) 合并相同路径的请求

图1 请求合并示例图

目前针对实时按需数据广播调度的研究主要包括如下几个方面：(1)索引技术[13-15].其目的为缩短调谐时间以节省终端能量消耗.Sun等人[13]针对XML文档提出一种two-tier索引结构，第一层索引给用户提供全局XML镜像，第二层索引提供具体索引细节，能够以最小的索引代价提前通知移动终端数据广播时间，节省移动终端能耗；(2)考虑用户访问时间(Average Access Time，AAT).相关文献研究广播调度以最大限度的降低AAT[16-21].Aksoy等人[16]综合考虑数据项请求数和最长等待时间，提出R×W算法，以数据项请求数与最长等待时间的乘积衡量数据项广播优先级，但其不考虑周期广播.Aksoy等人[16]考虑紧急度和请求满足率，提出SIN-α，对高速率请求环境有很好的适应性.Lu等人[20]研究最大产出请求选择问题(Maximum Throughput Request Selection problem，MTRS)和最小延迟请求排序问题(Minimum Latency Request Order problem，MLRO)，针对用户请求，数据项提出两层调度算法(Scheduling scheme with Least Loss Heuristic，SLLH)，能够获得最佳的广播效率；(3)考虑请求截止期.在请求带有截止期的广播环境，请求失效率(Loss Rate，LR)代替等待时间成为最重要的考虑因素.相关文献对考虑截止期的实时按需数据广播调度进行了研究[22-28].Hu等人[22]提出L×R×W算法，将即将失效数L、数据项请求数R、数据项最长等待时间W同时考虑，其权重指标能够全面地衡量数据项的优先级.Lei等人[27]同时考虑失效率和访问时间，其定义$Cost_{d_i}$来表示当前时间节点不广播d_i所造成的开销，提出最大价值优先算法(Maximum Value Gained First，MVGF)，每次选择$Cost$值最小的数据项广播.其$Cost_{d_i}$由不广播数据项d_i所导致的失效请求数以及用户增长的平均访问时间共同决定.此外，相关文献针对多信道、多数据项问题进行了研究[29-31]，主要优化多信道广播环境下可能出现的数据重叠[29]、数据冲突[30]等问题，优化调度，提高信道利用率.上述算法在降低移动终端能耗、减小等待时间、降低失效率等方面具有良好的效果，但没有关注到大规模数据广播调度应用过程中的上行带宽瓶颈和服务器并发压力等问题，目前仍缺少一种有效结合数据广播调度以及请求预处理的调度算法来处理这一难题.

为了打破按需数据广播上行带宽瓶颈、缓解服务器并发压力、保证高效的广播效率，本文对按需数据广播调度过程的请求预处理进行了研究，提出一种基于请求预处理的实时按需数据广播调度方法(Request Pre-Process Method，RPPM).本文的主要工作包括：

计算机学报

(1) 为了打破上行带宽瓶颈，缓解服务器并发压力，本文首先研究了最大化请求合并问题(Maximum Request Merge problem，MRM)并且提出了最佳请求合并算法(Optimal Request Merge algorithm，ORM)，其综合考虑无效数据率和请求减少率，能够高效地确定不同请求环境下二者的最佳权重，实时动态合并用户请求，有效减少用户请求数，达到解决上述问题的目的；

(2) 为了进一步提高信道利用率，本文研究了最佳合并请求优先级问题(Optimal Merged Request Priority problem，OMRP)，提出合并请求优先级和剪枝算法(Merged Request Priority and Prune algorithm，MRPP)量化分析合并请求优先级，剪枝部分低优先级合并请求；

(3) 结合 ORM 和 MRPP 提出一种基于请求预处理的实时按需数据广播调度方法 RPPM，其结合请求预处理和数据广播调度策略，能够高效地合并用户请求，准确地量化合并请求优先，保证高效的系统广播效率.

本文第 2 节介绍基于请求预处理的实时按需数据广播系统结构以及问题描述；第 3 节详细介绍 RPPM，其中 3.1 节详细介绍了针对 MRM 问题的 ORM 算法，3.2 节详细介绍了针对 OMRP 问题的 MRPP 算法；第 4 节详细介绍基于请求预处理的实时按需数据广播全局调度；第 5 节通过仿真实验验证了 RPPM 的有效性，并分析实验结果；第 6 节是结论及展望.

2　按需数据广播

本节为全文的知识准备，包括按需数据广播调度系统结构、基于分层目录树的用户请求、按需数据广播调度问题描述和评价指标. 2.1 节描述了基于请求预处理的实时按需数据广播系统结构；2.2 节介绍了基于分层目录树的用户请求；2.3 节为问题描述；最后，2.4 节介绍评价指标.

2.1　系统结构

本节介绍基于请求预处理的按需数据广播系统结构. 假设按需数据广播系统包含一个网络服务器，多个蜂窝网络 *cell*. 一个 *cell* 内包含一个中继节点(Relay Node，RN)以及多个移动客户端. RN 是用户同服务器移动通信的交换中心，并且具有计算能力. 每个广播周期，移动客户端将带有截止期的请求上传至 RN，随后侦听下行信道以获取请求数据. 若请求在截止期内得到满足，则请求成功，否则请求失败.

为了不失一般性，本文讨论 *cell* 内部的数据广播调度. 如图 2 所示，RN 接收用户请求 Req 保存于请求缓冲池 Q，待第 $k-1$ 个广播周期结束，RN 调用 RPPM 将 Q 中请求进行预处理，并将处理后的合并请求通过上行信道发送至网络服务器以获取对应数据(忽略网络服务器查找数据的时间)，RN 通过下行信道获取数据后立即转发，进行第 k 个周期数据广播(忽略 RN 转发数据时延). 具体地，用户请求 Req_i 为三元组，$Req_i=\{\ominus_i, t_i, p_i\}$，其中 $\ominus_i$ 为 Req_i 的请求截止期，t_i 为 Req_i 的请求发送时间，p_i 为 Req_i 的请求数据所在服务器的绝对查询路径，唯一标识数据项. RN 接收的请求保存于 $Q=\{Req_1, \cdots, Req_i, \cdots, Req_n\}$，RN 周期性的调用 ORM 算法将 Q 内请求进行合并处理生成合并请求队列 $MRQ=\{mReq_1, \cdots, mReq_i, \cdots, mReq_m\}$. 随后，RN 调用 MRPP 算法量化 MRQ 内各合并请求 $mReq_i$ 的优先级，并剪枝 MRQ 内的部分低优先级合并请求. 一旦 MRQ 的剪枝操作完成，RN 立即将其上传至服务器以获取请求数据. 服务器依据 MRQ 内合并请求优先级按序组织数据发送至 RN，RN 接收到数据立即转发至客户端以进行第 k 次广播.

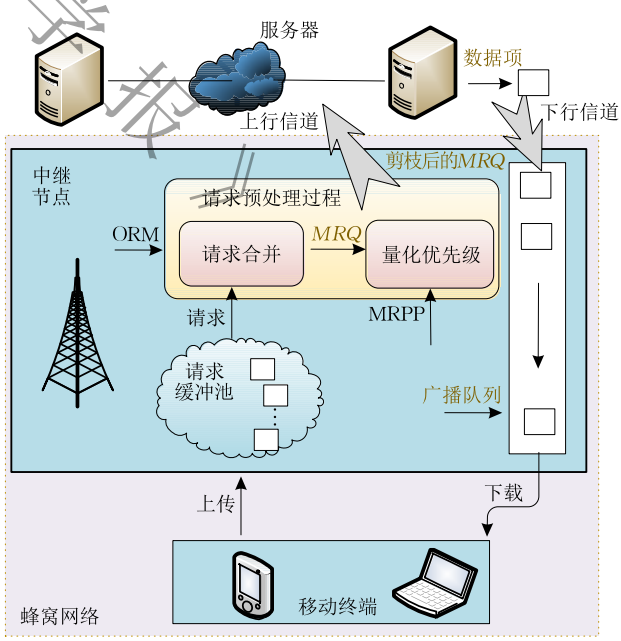


图 2　基于请求预处理的实时按需数据广播系统结构

2.2　基于分层目录树的请求

一般地，服务器提供给移动终端访问的目标数据项大多以分层目录树的方式组织，用户访问数据时通过查询路径(例如：$/D_1/W_2/d_3$)对数据项进行

定位.最常见的资源定位方式为统一资源定位符(Uniform Resource Locator,URL),其主要由协议、服务器域名、路径以及文件名四部分组成.为了不失一般性,本文假设用户查询路径包含 URL 中后两部分,即文件路径以及文件名.文件路径以及文件名指明了资源所在服务器的存储位置,其取值空间构成服务器提供给移动终端访问的树形目录结构.基于分层目录树的请求示意图如图 3 所示,其中方框表示文件夹,圆框表示可访问数据文件,用户请求通过查询路径 p 将请求定位于对应数据项,例如 $p_1=:/D_1/W_2/d_3$,故 Req_1 被映射至图 3 中数据项 d_3 所对应的位置.

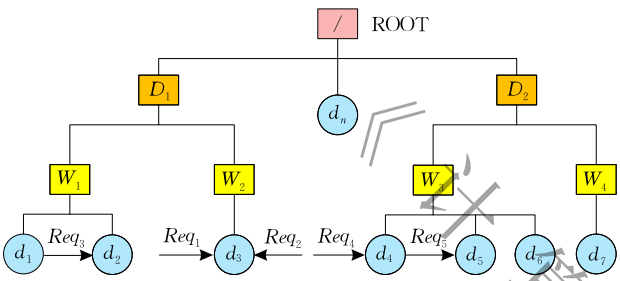


图 3 基于分层目录树的请求映射图

依据请求数据项所在分层目录树位置的不同,可将用户请求分为三类,不同用户请求类别对合并有重大影响,故本文将用户请求类别定义如下.

定义 1(第一类请求). 针对于同一数据项的不同请求为第一类请求,由符号 Q_1 表示.对于这一类请求,各请求的不同点为请求时间、请求截止期.

定义 2(第二类请求). 针对于同一文件目录下不同数据项的请求为第二类请求,由符号 Q_2 表示,对于这一类请求,其请求时间、请求截止期以及请求查询路径 p 均可能不同,但 p 中仅文件名(即数据项名称)不同.

定义 3(第三类请求). 针对于不同文件目录下不同数据项的请求为第三类请求,由符号 Q_3 表示,对于这一类请求,其请求时间、请求截止期以及请求查询路径 p 均可能不同,且 p 中文件路径以及文件名均不同.

2.3 问题定义

前文已提到由于大规模用户请求导致的上行带宽瓶颈以及服务器并发压力问题不可忽视,2.2 节将基于分层目录树的用户请求进行了分类定义,本文基于上述分类定义提出合并用户请求的思想以控制用户请求数量,缓解上述问题.为了保证合并请求的有效性,最大限度减少用户请求,本文对问题 MRM 进行了研究,MRM 的目标为合理、高效的合并请求.由上述关于用户请求分类定义可知,第一类请求针对于同一数据项,而这些请求的差异性对广播并无影响,因为广播被请求数据项可同时满足多个用户,因此可考虑将第一类请求进行合并.第二类请求所请求的数据项位于同一文件目录下,若能合理地将第二类请求合并为对该文件目录的请求(包含文件夹中所有数据),将大大减少用户请求数.下面我们以图 3 为例,进一步阐明 MRM 问题的核心思想.

例 1. 图 3 中包含五个用户请求 $Req=\{Req_1,Req_2,\cdots,Req_5\}$,其中 $p_1=:/D_1/W_2/d_3$,$p_2=:/D_1/W_2/d_3$,$p_3=:/D_1/W_1/d_2$,$p_4=:/D_2/W_3/d_4$ 以及 $p_5=:/D_2/W_3/d_5$.首先,将这五个请求依据 p_i 映射至对应请求数据项如图 4 所示.由定义可知,Req_1,Req_2 属于 Q_1,Req_4,Req_5 属于 Q_2,Req_1,Req_3 属于 Q_3.然后,我们将属于 Q_1,Q_2 的请求进行合并.具体地,将 Req_1,Req_2 合并生成合并请求 $mReq_1$,将 Req_4,Req_5 合并生成合并请求 $mReq_2$.合并请求后的结果如图 4 所示,通过合并,请求数得到了有效的控制,由原来的 5 减少为 3.然而,合并请求也暴露了如下问题:

(1)如何准确地确定合并请求特征参数以全面地替代被合并请求(包括请求等待时间、截止期,其影响调度优先级的确定),保证被合并请求的可满足性;

(2)合并 Q_2 中的请求将产生无效数据,导致广播信道的带宽浪费.无效数据的定义由定义 4 给出.

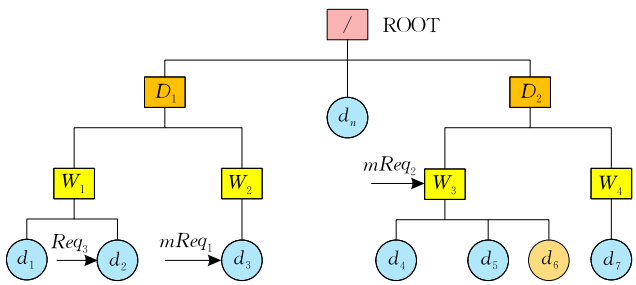


图 4 基于分层目录树的请求合并结果示例图

定义 4(无效数据). 没有被用户请求,但因合并第二类请求后导致被动广播的数据项为无效数据.例如图 4 中数据项 d_6 并没有被请求,但由于合并 Req_4,Req_5 后生成的 $mReq_2$ 指向路径 $:/D_2/W_3$ 下所有数据项,d_6 仍需要被广播,故 d_6 为无效数据.

通过以上分析可知,合并请求有利有弊.一方面合并请求可减少请求数目,有利于减轻上行带宽和

服务器压力；另一方面，合并请求的特征参数难以确定，并且合并请求可能导致无效数据的产生. MRM 问题的目的是权衡请求合并的利弊，找到一种合理的合并请求方法. 为了方便 MRM 问题的正式定义，我们首先介绍本文中符号定义如下：

(1) $Ftree$ 表示服务器提供给移动终端访问的目标数据项分层目录树；

(2) $Bnode_i$ 为 $Ftree$ 中第 i 个分支节点，即分层目录树中第 i 个文件夹，$Bnode_i=\{p_i,D_i,N_i\}$，其中 p_i 为分支节点 $Bnode_i$ 的绝对文件路径，$D_i=\{d_{i,1},\cdots,d_{i,j},\cdots,d_{i,h}\}$ 为 $Bnode_i$ 路径下包含的数据项集合，N_i 为 $Bnode_i$ 路径下包含的子分支节点集合. 如图 4 中 D_1 为分支节点，用 $Bnode_{D_1}$ 表示，其中 $p_{D_1}=:/D_1$，$D_{D_1}=\varnothing$，$N_{D_1}=\{Bnode_{W_1},Bnode_{W_2}\}$；

(3) U_i 为分支节点 $Bnode_i$ 下包含的有效数据项(即被请求的数据项)集合，V_i 为分支节点 $Bnode_i$ 下包含的无效数据项集合，R_i 为 $Bnode_i$ 包含的请求数. 如图 4 中 $U_{W_3}=\{d_4,d_5\}$ 为 $Bnode_{W_3}$ 下有效数据项集合，$V_{W_3}=\{d_6\}$ 为 $Bnode_{W_3}$ 下无效数据项集合，$R_{W_3}=2$；

(4) S_{d_i} 为数据项 d_i 的大小，S_{U_i} 为 $Bnode_i$ 下 U_i 包含的有效数据项大小总和，即 $S_{U_i}=\sum_{d_i\in U_i}S_{d_i}$，$S_{V_i}$ 为 $Bnode_i$ 下 V_i 包含的无效数据项大小总和，即 $S_{V_i}=\sum_{d_i\in V_i}S_{d_i}$；

(5) $\delta_{d_{i,j}}$ 为 $Bnode_i$ 内数据项 $d_{i,j}$ 包含的请求队列，$|\delta_{d_{i,j}}|$ 为请求队列的长度，即 $d_{i,j}$ 包含的请求个数；

(6) IVR_i 为 $Bnode_i$ 下无效数据大小占总数据量大小的比率，即无效数据率. IVR_i 影响请求合并的决策，若 IVR_i 过大，合并 $Bnode_i$ 中包含的 Q_2 请求将导致过量的带宽浪费，则不能合并 Q_2 的请求. 其中 IVR_i 的求解如式(1)：

$$IVR_i=\frac{S_{V_i}}{S_{U_i}+S_{V_i}}=\frac{\sum_{d_i\in V_i}S_{d_i}}{\sum_{d_i\in U_i}S_{d_i}+\sum_{d_i\in V_i}S_{d_i}} \tag{1}$$

(7) RRR_i 为合并 $Bnode_i$ 内请求前后，$Bnode_i$ 内请求数据减少的比率，即请求减少率. RN_i 为合并前 $Bnode_i$ 包含的请求个数，则 $RN_i=\sum_{d_{i,j}\in U_i}|\delta_{d_{i,j}}|$；

(8) MRM_i 为合并后 $Bnode_i$ 包含的请求个数，由定义可知，$Bnode_i$ 包含的请求均可归为 Q_2，若能合并 Q_2，则 $MRM_i=1$，若不可合并，MRM_i 为 U_i 包含的有效数据项个数 $|U_i|$. 即 $RRR_i=\frac{RN_i}{MRM_i}$；

(9) $mReq_i$ 为请求合并生成的合并请求，$mReq_i=\{mr_i,m\partial_i,mt_i,mp_i,ms_i\}$，其中 mr_i 为 $mReq_i$ 包含的请求数，$m\partial_i$ 为 $mReq_i$ 截止期，mt_i 为 $mReq_i$ 请求时间，mp_i 为 $mReq_i$ 指向请求数据的绝对路径，ms_i 为 $mReq_i$ 请求数据项大小；

(10) $weight_i$ 为 $mReq_i$ 的优先级，$weight_i$ 值越大，$mReq_i$ 优先级越高，越先得到响应；

(11) Bw 为下行广播信道带宽，T 为广播周期长度. MRM 问题的正式定义如下：

MRM. 给定一组数据项集合 $D=\{d_1,\cdots,d_i,\cdots,d_n\}$ 以分层目录树的形式组织在 $Ftree$ 中，请求缓冲池中包含一组用户请求 $Q=\{Req_1,\cdots,Req_i,\cdots,Req_h\}$ 与 D 关联. MRM 问题为找到一个合适的方法，有效地合并 Q 中第一类 Q_1 请求和第二类 Q_2 请求，生成合并请求队列 $MRQ=\{mReq_1,\cdots,mReq_i,\cdots,mReq_m\}$；同时确定生成的 $mReq_i$ 特征参数的取值，包括 $m\partial_i$，mt_i，使得 $mReq_i$ 能同时满足被合并的请求.

解决 MRM 问题生成的 MRQ 包含 Q 内所有的用户请求，若 MRQ 所请求的数据量大于一个广播周期内所能广播的数据量，则服务器无法在一个周期内响应 MRQ 内所有的请求，而周期调度算法将会在下一个广播周期重新生成新的 MRQ 以覆盖前者，故为了进一步缩短 MRQ 长度，减少上行信道以及服务器需要处理的请求数，需要对 MRQ 做进一步剪枝处理. 同时，为了保证合并请求后不影响广播效率，需要对合并请求优先级进行准确的量化分析. 基于此，我们研究了最佳合并请求优先级问题 OMRP 定义如下：

OMRP. 给定一组合并请求队列 $MRQ=\{mReq_1,\cdots,mReq_i,\cdots,mReq_m\}$，OMRP 问题为准确量化合并请求 $mReq_i$ 的优先级，并将 MRQ 进行剪枝操作，删除 MRQ 部分低优先级的合并请求，使得 $SumS_{MRQ}\le Bw\times T$，其中 $SumS_{MRQ}$ 为 MRQ 所请求的数据项大小之和，$SumS_{MRQ}$ 求解如式(2)：

$$SumS_{MRQ}=\sum_{mReq_i\in MRQ}ms_i \tag{2}$$

2.4　评价指标

实时按需数据广播系统中，请求数据时效性明显，如实时交通信息数据过期便失效，故请求截止期是实时按需数据广播系统中重要的限制因素. 调度算法首要的目的便是满足用户在请求失效(即截止期)前获得数据，降低请求失效率(Loss Rate，LR)[19]. 此外，在算法实时满足用户请求的情况下，

若用户等待时间过长，同样将影响用户体验，故用户平均访问时间(Average Access Time，AAT)[13]同样应该考虑.本文的一个目标为合理地调度数据广播以获得最低的 LR 和 AAT.

用户请求失效数总和与所有请求总和的比值被称为请求失效率.M 个周期总请求失效数为 $L_M=\sum_{k=1}^{M}\sum_{i=1}^{N}Led_{d_i,k}$，$Led_{d_i,k}$为数据项 d_i 在第 k 个周期的请求失效数.M 个周期总请求成功数为 $SC_M=\sum_{k=1}^{M}\sum_{i=1}^{N}SC_{d_i,k}$，$SC_{d_i,k}$为数据项 d_i 在第 k 个周期的成功请求数.则请求失效率可表示如式(3)所示.

$$LR=\frac{L_{k<M}}{L_{k<M}+SC_{k<M}}=\frac{\sum_{k=1}^{M}\sum_{i=1}^{N}Led_{d_i,k}}{\sum_{k=1}^{M}\sum_{i=1}^{N}Led_{d_i,k}+\sum_{k=1}^{M}\sum_{i=1}^{N}SC_{d_i,k}} \tag{3}$$

访问时间是指用户从发送请求到请求被满足或者请求失效的等待时间，$AT_{Req_{i,k}}$ 为 $mReq_i$ 在第 k 个周期结束时的访问时间，则 M 个周期的平均访问时间如式(4)所示，其中 $|Req_k|$ 为第 k 个周期中的请求数.

$$AAT=\frac{\sum_{k=1}^{M}\sum_{i=1}^{|Req_k|}AT_{i,k}}{\sum_{k=1}^{M}|Req_k|} \tag{4}$$

如前文所提到的，本文的另一个目标为减少上行信道发送和服务器响应的请求数以减轻上行带宽以及服务器并发压力.为了量化分析算法效果，本文定义请求合并率(Request Merge Rate，RMR)作为另一个重要的评价指标.RMR 是上行带宽和服务器并发压力降低的度量，因为上行带宽和服务器并发压力来源为过量的用户请求，而请求合并率刻画请求减少率.例如 $RMR=0.5$，则表示合并后请求数减少了 50%，则上行信道需要发送的数据量减少了 50%，同理，服务器并发每秒的并发数减少了 50%.

定义 5(请求合并率). 请求合并率 RMR 是指合并请求后请求数目减少量占原请求数目的比率，M 个广播周期 RMR 的求解如式(5)所示.其中 $|Req_k|$ 为第 k 周期请求数，$|MRQ_k|$ 为第 k 周期请求合并后的合并请求数.

$$RMR=\frac{\sum_{k=1}^{M}|Req_k|-\sum_{k=1}^{M}|MRQ_k|}{\sum_{k=1}^{M}|Req_k|} \tag{5}$$

3 请求预处理方法

按需数据广播系统中服务器时接收用户请求，随着数据广播系统中移动用户和实时应用数量的增长，上行信道带宽瓶颈问题和服务器并发处理请求压力问题变得更加棘手，尤其 3G、4G 网络环境下，其用户数据请求呈现请求多次、单次请求数据量小的特点.基于此，本文对用户请求预处理进行了研究，提出最大化请求合并问题 MRM 和最佳合并请求优先级问题 OMRP，并提出一种基于请求预处理的实时按需数据广播调度方法 RPPM，针对问题 MRM 和 OMRP，RPPM 包含最佳请求合并算法 ORM 和合并请求优先级和剪枝算法 MRPP，RPPM 框架如图 5 所示.

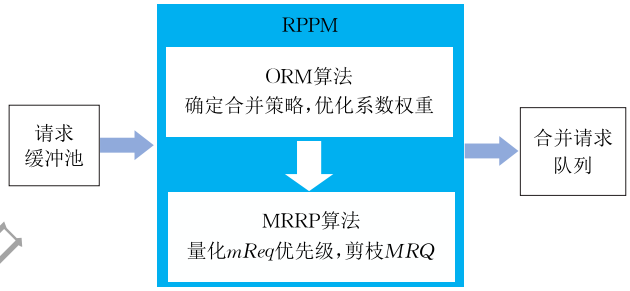


图 5 RMMP 方法框架图

(1) ORM 能以最佳的策略将用户请求进行合并，减少上行信道及服务器所需处理请求数，生成请求合并队列 MRQ.其中 ORM 大致可以分为如下几个步骤：① 确定基于阈值的合并策略；② 基于量子粒子群算法(Quantum-behaved Particle Swarm algorithm，QPSO)[32]快速确定最佳权重系数；③ 合并请求生成 MRQ；

(2) MRPP 借助 ORM 生成的合并请求队列 MRQ，量化 MRQ 中合并请求的优先级，将各个请求按优先级由大到小排序，依据系统周期广播数据量剪枝 MRQ，删除 MRQ 中部分低优先级合并请求.MRPP 能够准确量化合并请求优先级，保证优先级最高的请求最先得到响应，并进一步减少合并请求的数量.

3.1 算法 ORM

回顾前文，MRM 包含两个子问题：如何确定合并请求特征参数？如何权衡请求减少率 RRR 和无效数据率 IVR？本节首先对 MRM 问题进行分析，随后给出最佳请求合并 ORM 算法的具体实现.针对子问题一，本文通过一个用例来分析论证 ORM

《计算机学报》

算法思想.

例 2.　假设存在如下三个请求：$Req_1=\{\partial_1=1,t_1=1,p_1=:/d_1\}$，$Req_2=\{\partial_2=1,t_2=2,p_2=:/d_1\}$，$Req_3=\{\partial_3=4,t_3=3,p_3=:/d_1\}$需要合并生成 $mReq_i=\{mr_i,m\partial_i,mt_i,mp_i,ms_i\}$，其中 mr_i，mp_i，ms_i由实际请求情况直接确定，$m\partial_i$，mt_i需通过算法确定. 常见的综合数据特征的方法包括平均值、中位数、众数等，但均不适用于 $m\partial_i$的确定. 以平均数为例，若 $m\partial_i=avg(\partial_1,\partial_2,\partial_3)=2$，合并请求将导致 Req_1和 Req_2失效，因为 $\partial_1=\partial_2=1<m\partial_i=2$，故为了保证 $m\partial_i$能满足所有被合并请求的截止期，应取 $m\partial_i=\min(\partial_1,\partial_2,\partial_3)$. 针对 mt_i的确定，因 mt_i不同于 $m\partial_i$，其没有确定的判优指标，mt_i刻画的是被合并请求整体的等待时间. 例如：$mReq_a$包含请求 Req_1，Req_2和 Req_3，对应的 $t_1=1$，$t_2=5$，$t_3=6$. $mReq_b$包含请求 Req_4，Req_5和 Req_6，对应的 $t_4=2$，$t_5=2$，$t_6=2$. 若 $mt_i=\min(t_1,t_2,\cdots,t_n)$，则 $mt_a=1<mt_b=2$. 则单从请求时间考虑，$mReq_a$优先级高于 $mReq_b$. 若 $mt_i=avg(t_1,t_2,\cdots,t_n)$，则 $mt_a=4>mt_b=2$. 则单从请求时间考虑，$mReq_a$优先级低于 $mReq_b$. 显然，取 $mt_i=avg(t_1,t_2,\cdots,t_n)$更为合理，因为 $mReq_b$整体紧急程度要高于 $mReq_a$. 考虑到整体的平均访问时间 AAT 并兼顾公平性，取 $mt_i=avg(t_1,t_2,t_3)$更为合理. 综上分析，子问题一可通过式(1)得到解决.

$$m\partial_i=\min(\partial_1,\partial_2,\cdots,\partial_n),\quad mt_i=avg(t_1,t_2,\cdots,t_n)\tag{6}$$

针对子问题二，由例 1 的分析可知，合并属于 Q_1的请求并不会导致无效数据的产生，故问题存在于第二类请求 Q_2的合并. 由定义可知，属于 Q_2请求必将关联于同一文件夹，即位于 $Ftree$ 中某一分支节点 $Bnode_i$下. 子问题二的关键即根据 IVR_i和 RRR_i来判定是否合并 $Bnode_i$内的第二类请求. 其中 IVR_i为 $Bnode_i$包含的无效数据率，若 IVR_i越大，合并请求导致的无效数据越多，广播效率越低. RRR_i为请求的减少率，RRR_i越大，合并请求所减少的请求越多，上行带宽和服务器负载减轻越明显. 根据用户请求的随机性，RRR_i的增大不可避免地将引起 IVR_i的增大，RRR_i和 IVR_i相互作用和矛盾，故子问题二为典型的多目标二分决策问题. 常见的多目标决策方法有分层序列法、目标规划法、线性加权和法等，其中线性加权和法更简单高效，本文基于线性加权和法定义合并代价 $Merge\text{-}C$，提出基于阈值分析的合并策略. 其中合并代价定义如下.

定义 6(合并代价).　因合并 $Bnode_i$内 Q_2请求所产生的代价即为合并代价 $Merge\text{-}C$，其权衡 IVR 和 RRR. 对于 $Bnode_i$，基于线性加权和法的合并代价 可由式(7)表示.

$$Merge\text{-}C_i=w_1IVR_i+w_2\frac{1}{RRR_i}\tag{7}$$

$Merge\text{-}C_i$刻画合并 $Bnode_i$内 Q_2请求的代价，为了保证合并的有效性以及广播效率，本文提出基于合并阈值 $Merge\text{-}T$ 分析的合并策略，针对 $Bnode_i$内 Q_2请求合并决策如式(8).

$$Merge=\begin{cases}\text{true}, & Merge\text{-}C_i\le Merge\text{-}T\\ \text{false}, & Merge\text{-}C_i>Merge\text{-}T\end{cases}\tag{8}$$

由式(7)、(8)可知，$Merge\text{-}T$ 决定了合并粒度，$Merge\text{-}T$ 越大合并粒度越大. 在真实的数据广播系统中，$Merge\text{-}T$ 应根据实际的系统需求，综合考虑广播效率和系统负载压力确定. 本文在 5.2 节中详细分析了 $Merge\text{-}T$ 取值对调度影响并给出了建议取值. 式(7)中将 IVR_i和 RRR_i基于线性加权和确定 $Merge\text{-}C$，其中权重系数 w_1，w_2决定了 IVR_i和 RRR_i对 $Merge\text{-}C$ 的影响大小，是寻求合并最优解的关键. 结合式(6)～(8)，本文提出 ORM 算法，其主要包括两部分：(1) 基于量子粒子群优化算法 QPSO 的最佳权重搜索策略(Optimal Weight Search，OWS)以快速确定式(7)中最佳权重系数；(2) 借助于确定最佳权重系数的式(7)以及合并决策式(8)，合并请求. 具体如下：

OWS 策略.　量子信息的基本存储单元是量子比特，$|0\rangle$和$|1\rangle$表示一个量子比特的两种极化状态，量子比特状态可表示为 $P_{i,c}|0\rangle+P_{i,s}|1\rangle$，$P_{i,c}$和 $P_{i,s}$为$|0\rangle$和$|1\rangle$的概率幅. 式(7)中权重系数组成权重数组 $W=\{w_1,w_2\}$，令 $fitness(W)=\dfrac{RMR}{LR\times AAT}$为权重数组 W 对应的适应度值，适应度值越大，则表示权重数组 W 越优秀. OWS 策略具体可分为四步：

(1) 生成携带权重数组的初始量子粒子群. 假设初始化 m 个量子粒子，粒子数目越多，初始权重数组差异性越大. 其中第 i 个量子态粒子编码方式如式(9)所示. 量子粒子携带权重数组，因权重数组中包含两个参数，故每个粒子编码方式包含两组正余弦值. 其中 $\theta_{i,j}=2\pi\times rnd$，$rnd$ 为(0,1)内的随机数，$i=1,2,\cdots,m$. m 为量子粒子群中粒子数目.

$$P_i=\left[\begin{vmatrix}\cos\theta_{i,1}\\ \sin\theta_{i,1}\end{vmatrix}\ \begin{vmatrix}\cos\theta_{i,2}\\ \sin\theta_{i,2}\end{vmatrix}\right]\tag{9}$$

将上面的量子粒子的位置按照正余弦拆分，即可

得到每个量子态粒子所占据的两个位置，其分别对应于概率幅 $P_{i,s}$ 和 $P_{i,c}$，可用式(10)和式(11)表示.

$$P_{i,s}=(\sin\theta_{i,1},\sin\theta_{i,2}) \tag{10}$$

$$P_{i,c}=(\cos\theta_{i,1},\cos\theta_{i,2}) \tag{11}$$

每个量子态粒子的概率幅 $P_{i,s}$ 和 $P_{i,c}$ 可通过式(12)和(13)转化为权重数组 $W_{i,s}$ 和 $W_{i,c}$，W 可取值为 $W_{i,s}$ 或 $W_{i,c}$.

$$W_{i,s}=\left\{\frac{\sin\theta_{i,1}}{\sin\theta_{i,1}+\sin\theta_{i,2}},\ \frac{\sin\theta_{i,2}}{\sin\theta_{i,1}+\sin\theta_{i,2}}\right\} \tag{12}$$

$$W_{i,c}=\left\{\frac{\cos\theta_{i,1}}{\cos\theta_{i,1}+\cos\theta_{i,2}},\ \frac{\cos\theta_{i,2}}{\cos\theta_{i,1}+\cos\theta_{i,2}}\right\} \tag{13}$$

(2)更新权重数组. 权重数组的更新是随着概率幅 $P_{i,s}$ 和 $P_{i,c}$ 的更新而实现的. 每次迭代，都将 $P_{i,s}$ 和 $P_{i,c}$ 通过式(14)得到 $\overline{P_{i,s}}$ 和 $\overline{P_{i,c}}$，然后令 $P_{i,s}=\overline{P_{i,s}}$，$P_{i,c}=\overline{P_{i,c}}$ 实现更新，继续下次迭代.

$$\overline{P_{is}}=(\sin(\theta_{i,1}(t)+\Delta\theta_{i,1}(t+1)),\ \sin(\theta_{i,2}(t)+\Delta\theta_{i,2}(t+1))),$$

$$\overline{P_{i,c}}=(\cos(\theta_{i,1}(t)+\Delta\theta_{i,1}(t+1)),\ \cos(\theta_{i,2}(t)+\Delta\theta_{i,2}(t+1))),$$

其中 $\Delta\theta_{i,j}(t+1)=w\Delta\theta_{i,j}(t)+c_1r_1(\Delta\theta_l)+c_2r_2(\Delta\theta_g)$，

$$\Delta\theta_l=\begin{cases}2\pi+\theta_{i,l,j}+\theta_{i,j}(\theta_{i,l,j}-\theta_{i,j}<-\pi)\\ \theta_{i,l,j}-\theta_{i,j}(-\pi\leq\theta_{i,l,j}-\theta_{i,j}\leq\pi)\\ \theta_{i,l,j}-\theta_{i,j}-2\pi(\theta_{i,l,j}-\theta_{i,j}>\pi)\end{cases},$$

$$\Delta\theta_g=\begin{cases}2\pi+\theta_{g,j}+\theta_{i,j}(\theta_{g,j}-\theta_{i,j}<-\pi)\\ \theta_{g,j}-\theta_{i,j}(-\pi\leq\theta_{g,j}-\theta_{i,j}\leq\pi)\\ \theta_{g,j}-\theta_{i,j}-2\pi(\theta_{g,j}-\theta_{i,j}>\pi)\end{cases} \tag{14}$$

由式(14)可知，每次迭代，量子粒子概率幅的更新均基于上一次迭代结果进行，通过 $\Delta\theta_{i,j}(t+1)$ 通过上一步 $\Delta\theta_{i,j}(t)$ 确定，从而确保了已搜索得到的最优解不会被抛弃.

以上两步是为了提供符合 QPSO 的操作设定，而量子信息的引入主要是在下面的第(3)步中. 其能够有效地保证搜索的全局收敛性；

(3)权重值变异. 原始的 QPSO 算法易陷入局部最优，主要原因在于搜索过程中权重数组的多样性丢失. OWS 借助量子非门实现变异操作来避免多样性丢失. 设权重数据变异概率为 P_m，在(0,1)之间随机生成随机数 rnd_i，如果 $rnd_i<P_m$，则将该量子粒子上的量子比特通过式(15)进行变异操作.

$$\begin{bmatrix}0&1\\0&1\end{bmatrix}\begin{bmatrix}\cos\theta_{i,j}\\ \sin\theta_{i,j}\end{bmatrix}=\begin{bmatrix}\cos\left(\frac{\pi}{2}-\theta_{i,j}\right)\\ \sin\left(\frac{\pi}{2}-\theta_{i,j}\right)\end{bmatrix},\ j=1,2 \tag{15}$$

(4)设 $P_{i,l}$ 对应的 $W_{i,l}$ 为粒子 i 当前搜索到的适应度值最高的权重数组，P_g 对应的 W_g 为整个粒子群当前搜索到的适应度最大的权重数组. 迭代 $g_{\max}$ 的步骤如过程 1 所示.

过程 1.

1. FOR $g=1$ to $g_{\max}$ DO:
2. FOR $i=1$ to m:
3. 依据 $P_{i,s}$ 和 $P_{i,c}$ 通过式(12)、(13)得到 $W_{i,s}$ 和 $W_{i,c}$;
4. IF $fitness(W_{i,c})>fitness(W_{i,l})$ THEN $P_{i,l}=P_{i,c}$;
5. IF $fitness(W_{i,s})>fitness(W_{i,l})$ THEN $P_{i,l}=P_{i,s}$;
6. IF $fitness(W_{i,l})>fitness(W_g)$ THEN $P_g=P_{i,l}$;
7. END FOR
8. END FOR

请求合并. 一旦式(7)中最佳权重系数 w_1,w_2 确定，便可依据式(6)、式(7)和式(8)进行请求合并. 具体的请求合并分为如下三步：

(1)请求映射. 依据 p_i 将 Req_i 映射至 $Ftree$，即将 Req_i 添加至对应请求数据项 $d_{i,j}$ 的请求队列 $\delta_{d_{i,j}}$ 尾部；

(2)计算合并代价 $Merge\text{-}C$. 以节点编号递减的顺序计算 $Ftree$ 中各分支节点的 $Merge\text{-}C_i$ 值，针对 $Bnode_i$ 其 $Merge\text{-}C_i$ 求解方法如下：①对于 $Bnode_i$ 内每一个数据项 $d_{i,j}$，如果 $\delta_{d_{i,j}}=\varnothing$，则将 $d_{i,j}$ 加入到集合 V_i，否则将 $d_{i,j}$ 加入 U_i，$R_i=R_i+|\delta_{d_{i,j}}|$；②对于 $Bnode_i$ 中的每一个分支节点 $Bnode_j$，$U_i=U_i\bigcup U_j$，$V_i=V_i\bigcup V_j$，依据式(1)和(2)计算 IVR_i 和 RRR_i；③根据式(7)计算 $Merge\text{-}C_i$；

(3)生成合并请求队列 MRQ. 递归遍历 $Ftree$ 中每一个分支节点 $Bnode_i$，依据式(8)考察请求合并. 其递归函数 $Merge(Bnode)$ 如式(16).

$$Merge(Bnode)\begin{cases}return, & Bnode=\text{null}\\ merge\ Q_2\ return, & Merge\text{-}C_i\leq Merge\text{-}T\\ merge\ Q_1,Merge(child), & Merge\text{-}C_i>Merge\text{-}T\end{cases} \tag{16}$$

在递归遍历过程中，若 $Bnode_i=\text{null}$，则函数直接返回. 若 $Merge\text{-}C_i\leq Merge\text{-}T$，则合并 $Bnode_i$ 中 Q_2 请求. 若 $Merge\text{-}C_i>Merge\text{-}T$，则不合并 $Bnode_i$ 中 Q_2，但无条件合并 Q_1. 合并请求 $mReq_j$ 生成过程如下：①依据式(6)计算 $m\ni_j$，mt_j；②如果 $mReq_j$ 由 Q_1 生成，Q_1 指向数据项 d_i，则 $mp_j=p_i$，$ms_j=S_{d_{i,j}}$，$mr_j=|\delta_{d_{i,j}}|$. 若果 $mReq_j$ 由 Q_2 生成，Q_2 关联 $Bnode_i$，则 $mp_j=p_i$，$ms_j=S_{U_i}+S_{V_i}$，$mr_j=R_i$.

综上，ORM 算法的具体实现如下.

《计算机学报》

算法 1.　ORM 算法.

输入：Q，$Ftree$

输出：MRQ

1. 初始化合并请求队列 $MRQ=\varnothing$；
2. 确定合并阈值 $Merge\text{-}T$；
3. 依据 OWS 策略确定最佳权重系数 w_1，w_2；
4. FOR $Req_i \in Q$ DO：
5. 　依据 p_i 将 Req_i 映射至 $Ftree$；
6. END FOR
7. FOR $Bnode_i$ in $Ftree$ DO：
8. 　以节点编号 i 递减的顺序依据式(7)计算 $Merge\text{-}C_i$；
9. END FOR
10. FOR $Bnode_i \in Ftree$ DO：
11. 　依据式(16)递归遍历 $Ftree$，生成合并 MRQ；
12. END FOR

3.2　算法 MRPP

ORM 算法生成的合并请求队列 MRQ 包含的请求数据可能在一个广播周期内无法全部得到响应，并且由于合并，MRQ 内合并请求优先级无法确定，这导致 OMRP 问题的产生. 本节首先分析影响合并请求优先级的策略因素，随后详细介绍针对 OMRP 问题的请求优先级和剪枝算法 MRPP.

现有的调度算法大多都是针对数据项优先级的量化分析，根据一定的策略因素衡量数据项优先级，一般的策略因素包括数据项大小[11]、等待时间[16]、请求数[22]、截止期[23]等. 根据合并请求的定义可知，其同数据项拥有相同的数据特性，即 $mReq_i$ 包含请求数据项大小、合并请求数、截止期和等待时间等基本信息，为量化合并请求优先级提供了策略因素. 各策略因素对 $mReq_i$ 优先级的影响分析如下：

(1) 数据项大小. 合并请求所对应的数据项越大，响应该请求所耗费的时间越长. 换而言之，数据项越大所消耗的带宽越多，其他条件相同的情况下，为了降低 AAT 和 LR，数据项大小同合并请求优先级成反比；

(2) 等待时间. 为了防止热度较低的数据项请求出现饿死现象，降低 AAT，合并请求的等待时长必须被考虑，即等待时间越长，优先级越高；

(3) 合并请求数. 合并请求所包含的请求数目是衡量一个合并请求优先级的重要因素，合并请求数越大，说明响应该请求所能同时满足的用户数越大，AAT 和 LR 越低，故合并请求数越大，优先级越高；

(4) 截止期. 截止期是影响合并请求优先级另一重要因素，截止期刻画合并请求的紧急度，必须优先响应截止期小的请求以降低 LR. 故截止期与优先级成反比.

基于以上分析，我们提出了 MRPP 算法，其综合考虑以上四个因素，量化合并请求优先级. 其中等待时间影响平均访问时间 AAT，合并请求数决定热度，可直接作为量化指标. 数据项大小影响广播时间，广播时间同截止期结合，影响请求失效率 LR，MRPP 算法通过将其转化为失效数的方式更直观的转化为量化指标. 具体地，合并请求 $mReq_i$ 的优先级 $weight_i$ 可由式(17)求解，其中 mr_i 为 $mReq_i$ 包含的请求数，W_i 为 $mReq_i$ 的等待时间，设当前系统时间为 t，则 $W_i=t-mt_i$，SL_i 为假设当前时间节点 t 响应 $mReq_i$ 将导致的其他请求失效的失效数，可由式(18)求解.

$$weight_i=\frac{mr_i\times W_i}{SL_i} \tag{17}$$

$$SL_i=\sum_{m\partial_j<t+\frac{Bw}{ms_j},j\neq i} mr_j \tag{18}$$

其中 mr_j 为合并请求 $mReq_j$ 所包含的请求数. SL_i 的取值可能为 0，此时就用 $mr_i\times W_i$ 作为 $mReq_i$ 的优先级，因此将式(17)修改成式(19).

$$weight_i=\begin{cases}\dfrac{mr_i\times W_i}{SL_i}, & SL_i\neq 0\\ mr_i\times W_i, & SL_i=0\end{cases} \tag{19}$$

MRPP 基于式(19)，先量化 MRQ 中合并请求优先级，随后依据优先级剪枝 MRQ. 设 ρ 为加入当前广播周期考虑范围内的合并请求个数，σ 为加入当前广播周期数据项大小总和. 具体的 MRPP 实现如算法 2 所示.

算法 2.　MRPP 算法.

输入：MRQ

输出：MRQ

1. 初始化 $\rho=0$，$\sigma=0$；
2. FOR $mReq_i \in Q$ DO：
3. 　依据式(17)、(19)计算 $mReq_i$ 的 $weight_i$；
4. 　$\rho=\rho+1$；
5. END FOR
6. 将 MRQ 按照 $weight_i$ 递减的顺序排序；
7. FOR $i=1$ to ρ DO：
8. 　IF $\sigma<Bw\times T$ THEN：
9. 　　$\sigma=\sigma+ms_i$，其中 ms_i 为 $mReq_i$ 包含的数据量；
10. 　ELSE 将 $mReq_i$ 从 MRQ 中删除；
11. 　END IF
12. END FOR

4　全局调度流程

基于 RPPM，我们提出了基于请求预处理的实

时按需数据广播全局调度流程.具体地,其主要流程包含四个部分:

(1)接收请求.在第 $k-1$ 个广播周期内,移动客户端生成请求 Req_i,并传送至所属蜂窝网络 $cell$ 内的中继节点 RN. RN 接收请求,先检查 Req_i 所请求数据项 d_i 是否存在于广播队列,如果存在,则直接将 Req_i 加入相应数据项的请求列表 δ_{d_i} 中;如果不在,则将请求添加至请求缓冲池 Q 内;

(2)请求预处理.第 $k-1$ 个广播周期结束,RN 调用 RPPM 将 Q 中请求进行预处理.具体地,利用 ORM 算法将用户请求进行合并,生成请求合并队列 MRQ,然后利用 MRPP 算法量化 MRQ 内各合并请求 $mReq_i$ 的优先级,并剪枝 MRQ 内部分低优先级合并请求;

(3)获取数据.RN 通过上行信道上传 MRQ,服务器接收到 MRQ 后,依据 MRQ 从因特网、数据库中查找满足请求的数据项.本文不考虑服务器获取数据的时间;

(4)调度广播.服务器将获取的数据项集合按照 MRQ 所请求的顺序排序,组织生成广播队列进行第 k 个周期广播.

如前文提到的,中继节点 RN 保存 Q 和 MRQ,服务器保存广播队列 D.全局调度流程如过程 1 所示.其中步骤 1~8 为第一部分,步骤 9~20 为第二部分,步骤 21~22 为第三部分,步骤 23~26 为第四部分.

过程 2. 基于 RPPM 全局调度流程.

1. 对于当前广播周期 $k-1$ 的时间节点 t DO:
2. IF 用户在时间区间 $(t-1,t]$ 提交了新的请求 Req_i THEN
3. IF Req_i 所请求数据项 d_i 存在于 D 内,且未广播
4. THEN 将 Req_i 添加至 d_i 的请求队列 δ_{d_i} 中
5. ELSE
6. 将 Req_i 添加至请求缓冲池 Q;
7. END IF
8. END IF
9. IF 广播周期 $k-1$ 结束 THEN
10. FOR $Req_i \in Q$ DO:
11. IF $\partial_i < t$ THEN
12. 将 Req_i 从 Q 内删除
13. END IF
14. END FOR
15. 获取 $Ftree$;
16. FOR $Req_i \in Q$ DO:
17. 运用 OMP 算法生成合并请求队列 MRQ;
18. 运用 MRPP 算法量化 MRQ 内每一个 $mReq_i$ 优先级;
19. END FOR
20. 将 MRQ 按优先级排序,并剪枝;
21. 将 MRQ 传送至服务器;
22. 服务器依据 MRQ 或取数据生成广播队列 D;
23. 组织 D 进行广播;
24. IF d_i 在时间节点 t 广播结束 THEN
25. 将 d_i 从广播队列 D 内删除,并将 d_i 的所有请求从 Q 中删除;
26. END IF
27. END IF
28. 时间节点 $t=t+1$;

5 实验分析

本节通过 RPPM 同两个最新算法 SLLH[20]、MVGF[27]和一个基准算法 R×W[16]在不同广播环境下的对比实验来验证 RPPM 方法的有效性.其中 SLLH,MVGF 作为最新调度算法,能够取得最优的系统服务质量.而 R×W 作为基准算法,能够保证 RPPM 的最低有效性.5.1 节介绍了实验环境,包括实验数据集、基本的参数设定.5.2 节验证了合并阈值对 RPPM 的影响,并基于基准对比算法 R×W 和请求合并率,给出了在不同截止期分布下合并阈值的建议取值区间.基于 5.2 节得出的合并阈值建议取值,5.3 节~5.5 节通过 RPPM 同上述对比算法在真实数据集上的仿真实验,分别验证了 RPPM 在请求截止期、请求速率、请求模式 Zipf 分布偏度值不同的情况下的有效性.

5.1 实验环境

为了验证 RPPM 的有效性,实验所用数据集必须符合实时按需数据广播系统高负载的特征.基于此,本文选取 1998 年世界杯网站①第 38 天的真实访问记录作为实验数据集,并实现 recreate 工具②,依据访问记录逆向生成了服务器提供给移动终端访问的目标数据项分层目录树.该天访问记录共包含 4923 个不同的 Web 页面和图像对象,共有超过 700 000 个请求,平均请求速率为 83 条/s,符合高并发高负载的特征要求.数据集中访问记录服从典型的网络数据访问 Zipf 分布[33],数据项 d_i 在 Zipf 分

① World Cup 98 Web Site Access Logs. http://ita.ee.lbl.gov/html/contrib/WorldCup.html[EB/OL], 1998

② Tools recreate. ftp://ita.ee.lbl.gov/software/WorldCup_tools.tar.gz, 1998

布下的访问概率 $P_i=\frac{(1/i)^{\theta}}{\sum_{i=1}^{4923}(1/i)^{\theta}}$，$\theta=0.8$. 本文设定平均截止期大小为 60 s，可服从 Web 数据请求最典型的三种分布：均匀分布、指数分布、固定分布[24]. 设定数据项大小服从均值为 10 kb 的均匀分布[25]. 具体参数的默认设置如表 1 所示.

表 1　实验参数设置

参数名称	分布	默认值	参数可调
广播周期	—	5 s	Y
带宽	—	10 M/s	Y
运行时间	—	6000 s	N
截止期	均匀分布	60 s	Y
数据项大小	均匀分布	100 kb	Y
请求模式	Zipf 分布	—	Y
请求速率	—	100	Y
数据集大小	—	4923	N
Zipf 偏度值	—	0.8	Y

5.2　合并阈值

Merge-T 的取值直接决定了合并粒度的大小，合并粒度越大，请求合并率越高，但同时无效数据率也将随着升高. 合理地合并粒度对有效降低服务器响应请求开销，提高信道利用率有重要作用. 本节通过仿真实验来分析在截止期服从不同分布情况下，*Merge-T* 的取值对调度的影响，并给出了在不同截止期分布下 *Merge-T* 的建议取值. 仿真实验中 *Merge-T* 的取值范围为(0,1]，截止期服从均匀分布、指数分布、固定分布. 均匀分布截止期取值范围为[2,120]，指数分布率参数 $\lambda=1/60$，固定分布截止期取值为 60 s. 其余参数设置参照表 1 的默认取值.

Merge-T 取值变化对服务质量影响如图 6 所示. 图中散点为算法实际运行结果，图中实线为实际运行结果拟合曲线，以便于分析实验结果走向趋势. 从图中 6(a)、(b)、(c)可以看出，在截止期服从三种不同分布情况下，*LR*，*AAT* 随着*Merge-T* 取值变化的趋势相似，即随着 *Merge-T* 取值从 0 增大到 0.5，*LR*、*AAT* 逐渐增大，当 *Merge-T* 值达到后 0.5 左右，*LR*、*AAT* 不再增大，逐渐趋于稳定. 这是由于 *Merge-T* 的取值决定合并粒度大小，当合并粒度增大时，将导致系统广播的无效数据增加，从而导致 *LR*、*AAT* 的增大. 但随着 *Merge-T* 的继续增大，始终有部分属于 Q_3 类的请求无法合并，无效数据不再随着 *Merge-T* 的增大而增加，从而 *LR*、*AAT* 不再变化. 在 *Merge-T* 处于(0,0.5]内，请求合并率 *RMR* 随着 *Merge-T* 值的增大快速增大，当 *Merge-T* 处于(0.5, 1]内，*RMR* 增长速率逐渐降低，最终收敛于 80%左右. 原因显而易见，随着 *Merge-T* 的增大，合并粒度逐渐增大，故 *RMR* 不断增大，但无论 *Merge-T* 取值为多少，始终有部分属于 Q_3 类的请求无法合并最，故 *RMR* 收敛于 80%左右.

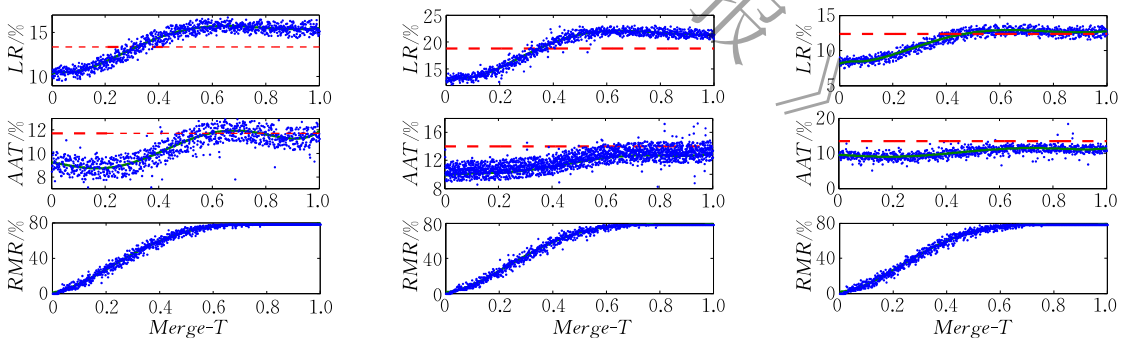


图 6　合并阈值 *Merge-T* 在不同截止期分布下对调度的影响

基于以上分析可知，*Merge-T* 取值的增大将导致无效数据率的增加，从而引起 *LR*、*AAT* 的增大，系统服务质量随着降低；同时，*Merge-T* 取值的增大也能更大程度的增加 *RMR*，减少合并请求数，降低系统的请求负载. 由此可见，应根据实际系统需求，权衡系统服务质量以及系统请求负载，确定 *Merge-T* 的取值. 本文选取 R×W 算法作为基准算法，图 6 中红色虚线为 R×W 在不同情况下的 *LR*、*AAT*. *Merge-T* 取值对应的 *LR*、*AAT* 应不大于基准算法，基于以上实验结果，本文给出 *Merge-T* 不同截止期分布下的建议取值区间如表 2 所示.

表 2　不同截止期分布下合并阈值 *Merge-T* 的建议取值区间

截止期分布	*Merge-T* 建议取值区间
均匀分布	[0.20,0.30]
指数分布	[0.25,0.35]
固定分布	[0.30,0.40]

《计算机学报》

5.3 截止期变化对调度的影响

在具有时效性约束的广播系统中，用户请求截止期是影响调度及服务质量的重要因素之一. 本组实验对比 RPPM 同 SLLH、MVGF 以及 R×W 算法在截止期分别服从均匀分布、指数分布、固定分布下 *LR* 的表现. 为了模拟截止期时间约束严格以及宽松的环境，本文取截止期的变化范围为[10，35，60，…，210]. 同时，本文还通过实验研究了请求截止期对 RPPM 请求合并率 *RMR* 的影响. 截止期服从均匀分布时、指数分布、固定分布情况下，*Merge-T* 取值分别为 0.25，0.30，0.35. 其余参数设置参照表 1 的默认取值.

图 7(a)、(b)、(c)分别为截止期服从均匀分布、指数分布和固定分布时截止期大小变化对调度的影响. 如图所示，三种分布情况下，所有算法 *LR* 均随着请求截止期的增大而降低，这是因为请求截止期越大，用户请求数据的时效性限制越低，用户请求被满足的可能性越大. 在截止期服从指数分布情况下，各算法的 *LR* 最高，而在截止期服从固定分布时，*LR* 最低. 在截止期服从均匀分布情况下，RPPM 同 MVGF 的 *LR* 十分接近. 随着截止期增大至 85，RPPM 的 *LR* 甚至略低于 MVGF 一个百分点. 同表现最优的 SLLH 算法相比，也仅高 3%左右. 而当截止期处于[10～35]，即用户请求时效性限制较高的情况下，RPPM 同 SLLH 方法的 *LR* 表现差别相对明显，约相差 5%. 这是因为 RPPM 合并用户请求，不可避免地将导致少量无效数据的广播，故不能充分提高信道利用率. 在截止期服从指数分布情况下，RPPM 同 SLLH、MVGF 在 *LR* 上的差距大于均匀分布，这是因为指数分布截止期时效性要求高，但仍然低于基准算法 R×W，这得益于 RPPM 中 *Merge-T* 的最佳取值. 截止期服从固定分布时，随着截止期的不断增大，RPPM 同 SLLH 在 *LR* 上的差距逐渐减小，最后均趋近与 0，而基准算法 R×W 始终高于其余三个对比算法.

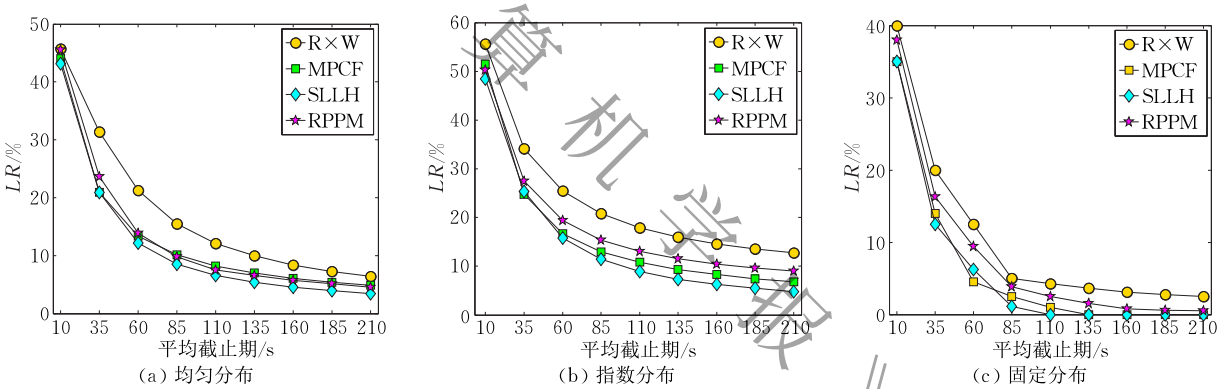


图 7 截止期分布对 *LR* 的影响

因 RPPM 的 *RMR* 在截止期服从不同分布情况下差别不大，故本节仅介绍截止期服从均匀分布的情况. 表 3 为截止期服从均匀分布情况下，截止期大小变化对 RPPM 请求合并率 *MRM* 的影响. 因其余三个对比算法不考虑请求合并，故服务器接收速率为请求速率，*MRM* 为 0，不在表中显示. 从表中可以看出，随着截止期的不断增大，*MRM* 逐渐增大，最后稳定在 50%～57%之间. 这是由于 RPPM 在判断请求是否合并的过程中将即将失效数作为一个判断指标. 若合并造成的即将失效数过高，则不进行请求合并. 在截止期较小时，合并请求产生的即将失效数较大，故 *MRM* 低，随着截止期的增大，合并请求导致的即将失效数减小，*MRM* 随之增大，最后趋于稳定. 同其余三种调度算法相比，RPPM 合并请求后将请求速率维持在 43～65 之间，而其余三种算法由于其没有考虑请求合并，故其请求速率始终为 100. 因此，相对于其他对比算法，RPPM 考虑请求合并可减少 35%～57%的用户请求，大大减少服务器响应用户请求的开销.

表 3 均匀分布截止期变化对 *RMR* 的影响

截止期/s	请求速率/(次/s)	服务器接收速率/(次/s)	*RMR*/%
10	100	65.1	**34.9**
35	100	60.3	**39.7**
60	100	57.4	**42.6**
85	100	55.3	**44.7**
110	100	49.6	**50.4**
135	100	45.7	**54.3**
160	100	43.9	**56.1**
185	100	43.5	**56.5**
210	100	43.3	**56.7**

综上，本组实验验证了 RPPM 在截止期服从均

匀、指数、固定三种情况下的有效性. 在以上三种分布中，RPPM 的 *LR* 始终低于基准算法 R×W，但略高于最优的 SLLH 算法，并且在截止期服从均匀分布情况下略低于 MVGF 算法. RPPM 在请求失效率有所保证的前提下，其 *MRM* 可以达到 35%～57%. 换而言之，同其他优秀算法相比，RPPM 以请求失效率略微高 2%～4%左右的代价来减少服务 35%～57%的响应请求开销.

5.4　请求速率对调度的影响

请求速率决定系统负载，本组实验对比 RPPM 同 SLLH、MVGF 以及 R×W 算法在不同请求速率下的表现. 请求速率的变化范围为[20,80,…,500]. 截止期服从均匀分布，*Merge-T* 为 0.25. 其余参数设置参照表 1 的默认取值.

请求速率对调度的影响如图 8 所示，所有算法的 *LR* 均随着请求速率的增大而增大，当请求速率达到 200 以后，逐渐达到最大值. 请求速率越高，单位时间内用户产生的请求越多，系统负载增大，而系统广播带宽一定，故导致 *LR* 增大. 在请求速率小于 60 时，RPPM 的 *LR* 甚至高于基准算法 R×W，随着请求速率的增大，才逐渐与 MVGF 相近. 当请求速率大于 120 时，RPPM 的 *LR* 低于 MVGF，接近于 SLLH. 这是因为请求速率低，将导致 RPPM 请求合并效率低.

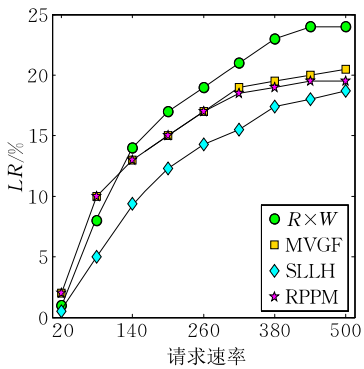


图 8　请求速率对 *LR* 的影响

请求速率对 RPPM 的 *RMR* 影响如表 4 所示，同 5.3 节，对算法 *RMR* 为 0，没有在表中显示. RPPM 的 *RMR* 随着请求速率的增大先逐渐增大，最后稳定在 50%左右. 当请求速率为 20 时，由于请求速率过小，达不到 RPPM 合并请求的门限，故 *RMR* 为 0. 当请求速率达到 200 时，RPPM 的 *RMR* 达到峰值，并不在随着请求速率的增大而有明显的波动. 虽然 *RMR* 不在发生较大变化，相对于不考虑请求合并的情况，RPPM 每秒减少的服务器接收请求数却不断增加.

表 4　请求速率变化对 *RMR* 的影响

请求速率/s	服务器接收速率/(次/s)	*RMR*/%
20	20	**0**
80	60.6	**24.3**
140	97.58	**30.3**
200	97.6	**51.2**
260	129.5	**50.2**
320	156.5	**51.1**
380	177.8	**53.2**
440	204.6	**53.5**
500	231.5	**53.7**

5.5　Zipf 分布偏度值 θ 对调度的影响

假设用户请求模式服从 Zipf 分布，即数据项 d_i 的访问概率 $P_i=\dfrac{(1/i)^\theta}{\sum_{i=1}^{N}(1/i)^\theta}$，$\theta\in[0,1]$. 其中偏度因子 θ 的取值决定了用户请求的集中度，当 $\theta=0$ 时，Zipf 分布等同于均匀分布，用户请求均匀分布在 N 个数据项内. 随着 θ 值的增大，用户请求偏度越集中. 本节通过仿真实验研究了参数值 θ 的变化对 RPPM 的影响，其中 θ 的取值为[0.1,0.2,…,1]. 截止期服从均匀分布，*Merge-T* 为 0.25. 其余参数设置参照表 1 的默认取值.

Zipf 偏度因子 θ 对调度的影响如图 9 所示，所有算法的 *LR* 均随着 θ 的增大而减小，且随着 θ 的增大，*LR* 下降速率不断增大. 这是因为 θ 值的越大，用户请求分布越集中在少部分热点数据中，广播相同的数据量可同时满足更多的用户请求. RPPM 在 θ 大于 0.9 时 *LR* 低于 MVGF 算法，接近于 SLLH，说明 RPPM 更加适应于用户请求偏度更高的请求环境. 这是因为用户请求越集中，RPPM 合并请求所产生的无效数据越少，合并效率越高. θ 对 RPPM 的 *MRM* 影响如表 5 所示. 从表中数据可以看出，θ 取值的变化对 RPPM 的 *RMR* 影响较大. 当 θ 取值较小的情况下，用户请求相对分散，从而导致式(7)中 *IVR* 以及 1/*RRR* 同时增大. 这是因为当请求分散至各个服务器文件系统的各个文件夹中，使得单个文件内无效数据同有效数据的比值增大，同时，对同一个数据项的请求减少，使得 1/*RRR* 值增大. 因此，当 θ 取值为 0.1～0.5 时，RPPM 的 *RMR* 仅为 25%～30%左右. 随着 θ 的不断增大，用户请求集中于服务器的部分数据中，使得合并请求所产生的代价降低，故 *RMR* 随之增大. 同不考虑请求合并的其他三个算法对比而言，即使在最坏的情况下，运用 RPPM 也能降低服务器 25.3%的请求速率，并且随着 θ 的增大，RPPM 最高可降低服务器 56.2%的请求速率.

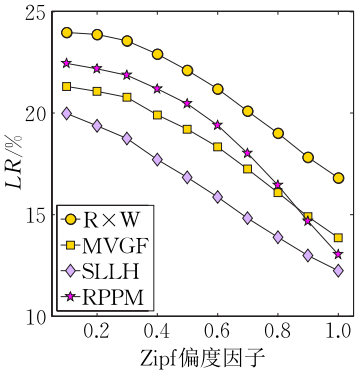


图 9 Zipf 偏度因子 θ 对 LR 的影响

表 5 Zipf 偏度因子 θ 对 RMR 的影响

θ	请求速率/(次/s)	服务器接收速率(次/s)	RMR/%
0.1	100	74.8	**25.2**
0.2	100	73.6	**26.4**
0.3	100	71.4	**28.6**
0.4	100	69.9	**30.1**
0.5	100	65.5	**34.5**
0.6	100	58.9	**41.1**
0.7	100	51.7	**48.3**
0.8	100	49.6	**50.4**
0.9	100	45.0	**55.0**
1.0	100	43.2	**56.2**

6 结论与展望

为了减少用户请求，缓解上行带宽及服务器并发压力，本文提出了一种按需数据广播请求预处理方法 RPPM：利用最佳请求合并算法 ORM 合并请求，减少请求数目，确定合并请求特征参数；然后，基于 ORM 生成的合并请求队列 MRQ，利用合并请求优先级和剪枝算法 MRPP 对其进行优先级量化以及剪枝操作. 基于 RPPM 提出了一种基于请求预处理的实时按需数据广播全局调度方案. 为了检验 RPPM 方法的有效性，本文基于 1998 年世界杯网站真实访问数据集进行了大量对比实验，并得出如下结论：

(1) 合并阈值 $Merge\text{-}T$ 对 RPPM 的表现有重要影响，合理的 $Merge\text{-}T$ 可使得 RPPM 在保证系统服务质量(即较低的请求失效率和平均访问时间)的同时，明显地提高请求合并率，缓解上行带宽及服务器并发压力；

(2) 请求截止期、请求速率、请求 Zipf 分布偏度值均对 RPPM 的请求失效率和请求合并率有影响，相对于其他三种对比算法，RPPM 的请求失效率在不同实验设定下均有较好的表现，基本接近于表现最优的 SLLH 算法，与此同时 RPPM 考虑请求预处理，通过请求合并，剪枝等操作可有效地减少约 50% 的用户请求，即缓解上行带宽及服务器并发压力约 50%；

(3) 不可否认，RPPM 相对于 MVGF 以及 SLLH 所适用的场景较为狭窄. 仅在高请求速率、高 Zipf 偏度因子的情况下才能发挥最佳的表现. 这表明 RPPM 受广播环境的影响较大. 相对于其他对比算法，RPPM 需要兼顾请求失效率和请求合并率，而由上文分析中可知，在合并请求第二类请求时将不可避免的导致无效数据的产生，直接导致请求失效率的升高. 而在请求速率低，Zipf 偏度因子小(请求分散)的情况下，合并第二类请求所产生的无效数据较多，故 RPPM 在此种情况下表现较差.

进一步的研究仍需要在以下几个方面继续：

(1) 合并阈值 $Merge\text{-}T$ 对 RPPM 方法表现影响较大，合理的 $Merge\text{-}T$ 将有助于降低请求失效率，提高请求合并率，未来的研究将尝试通过研究最佳 $Merge\text{-}T$ 的取值来提高 RPPM 的表现；

(2) 在算法 ORM 中通过取最小值和平均值的方式确定合并请求特征参数 $m\partial_i$ 和 mt_i 过于简单，在未来的研究中我们将尝试提出更加有效的方式确定 $m\partial_i$ 和 mt_i.

参 考 文 献

[1] Chen J, Lee V C S, Liu K, et al. Efficient processing of requests with network coding in on-demand data broadcast environments. Information Sciences, 2013, 232(5): 27-43

[2] Lu Z, Shi Y, Wu W, et al. Data retrieval scheduling for multi-item requests in multi-channel wireless broadcast environments. IEEE Transactions on Mobile Computing, 2014, 13(4): 752-765

[3] Ji H, Lee V C S, Chow C Y, et al. Coding-based cooperative caching in on-demand data broadcast environments. Information Sciences, 2017, 385(c): 138-156

[4] Xu S, Lu W, Xu L. Push- and pull-based epidemic spreading in networks: thresholds and deeper insights. ACM Transactions on Autonomous and Adaptive Systems, 2012, 7(3): 1-26

[5] Chen J, Lee V C S, Liu K. On the performance of real-time multi-item request scheduling in data broadcast environments. Journal of Systems and Software, 2010, 83(8): 1337-1345

[6] Costa R, Brasileiro F, Filho G L, et al. Using broadcast networks to create on-demand extremely large scale high-throughput computing infrastructures. Journal of Grid Computing, 2012, 10(3): 419-445

[7] Li X, Xiong J, Liu B, et al. A capacity improving and energy saving scheduling scheme in push-based converged wireless broadcasting and cellular networks//Proceedings of the IEEE International Symposium on Broadband Multimedia Systems and Broadcasting. Nara, Japan, 2016: 1-6

[8] Li G, Zhou Q, Li J. A novel scheduling algorithm for supporting periodic queries in broadcast environments. IEEE Transactions on Mobile Computing, 2015, 14(12): 2419-2432

[9] Zhou J, Fox J R, Hu R Q, et al. Scaling of on-demand broadcast scheduling in stressed networks. IEEE Transactions on Communications, 2016, 64(8): 3419-3429

[10] Wu X, Lee V C S. Wireless real-time on-demand data broadcast scheduling with dual deadlines. Journal of Parallel and Distributed Computing, 2005, 65(6): 714-728

[11] Hu W, Xia C, Du B, et al. An on-demanded data broadcasting scheduling considering the data item size. Wireless Networks, 2015, 21(1): 35-56

[12] Balaanand M, Leena C, Raj A D. File system organization paradigm using meta aware data. Immunological Reviews, 2014, 224(1): 265-283

[13] Sun W, Qin Y, Wu J, et al. Air indexing for on-demand XML data broadcast. IEEE Transactions on Parallel and Distributed Systems, 2014, 25(6): 1371-1381

[14] Hu W, Qiu Z, Wang H, et al. A real-time scheduling algorithm for on-demand wireless XML data broadcasting. Journal of Network and Computer Applications, 2016, 68: 151-163

[15] Ge Wei, Luo Sheng-Mei, Zhou Wen-Hui, et al. HiBase: A hierarchical indexing mechanism and system for efficient HBase query. Chinese Journal of Computers, 2016, 39(1): 140-153(in Chinese)
(葛微，罗圣美，周文辉等. HiBase：一种基于分层式索引的高效 HBase 查询技术与系统. 计算机学报，2016，39(1)：140-153)

[16] Aksoy D, Franklin M. R×W: A scheduling approach for large-scale on-demand data broadcast. IEEE/ACM Transactions on Networking, 1999, 7(6): 846-860

[17] Cheng Hong-Ju, Huang Xing-Bo, et al. Minimum-energy broadcast algorithm for wireless sensor networks with unreliable communication. Journal of Software, 2014, 25(5): 1101-1112(in Chinese)
(程红举，黄行波等. 不可靠通信环境下无线传感器网络最小能耗广播算法. 软件学报，2014，25(5)：1101-1112)

[18] Zhao Rui-Qin, Liu Zeng-Ji, Wen Ai-Jun. An Efficient energy-saving broadcast mechanism for wireless sensor networks. Acta Electronica Sinica, 2009, 37(11): 2457-2462(in Chinese)
(赵瑞琴，刘增基，文爱军. 一种适用于无线传感器网络的高效节能广播机制. 电子学报，2009，37(11)：2457-2462)

[19] Lv J, Lee V C S, Li M, et al. Supporting multi-level quality of services in data broadcast systems. International Journal of Sensor Networks, 2015, 18(3/4): 142-153

[20] Lu Z, Wu W, Li W W, et al. Efficient scheduling algorithms for on-demand wireless data broadcast//Proceedings of the IEEE International Conference on Computer Communications. San Francisco, USA, 2016: 1-9

[21] Feng Cheng, Li Zhi-Jun, Jiang Shou-Xu. Data aggregation scheduling in wireless mobile multichannel sensor networks. Chinese Journal of Computers, 2016, 39(5): 931-945(in Chinese)
(冯诚，李治军，姜守旭. 无线移动多信道感知网络上的数据聚集传输规划. 计算机学报，2016，39(5)：931-945

[22] Hu W, Fan C, Luo J, et al. An on-demand data broadcasting scheduling algorithm based on dynamic index strategy. Wireless Communications and Mobile Computing, 2015, 15(5): 947-965

[23] Liu C M, Su T C. Broadcasting on-demand data with time constraints using multiple channels in wireless broadcast environments. Information Sciences, 2013, 242(2): 76-91

[24] He P, Shen H. On-demand multimedia data broadcast in MIMO wireless networks. Physical Communication, 2016, 20: 1-16

[25] He P, Shen H, Tian H. On-demand data broadcast with deadlines for avoiding conflicts in wireless networks. Journal of Systems and Software, 2015, 103(C): 118-127

[26] Chen C L, Lv J S, Jiang Y. Algorithms of admission control and batch scheduling of on-demand broadcast with deadlines. Applied Mechanics and Materials, 2014, 511-512: 892-897

[27] Lei M, Vrbsky S V, Xiao Y. Scheduling on-demand data broadcast in mixed-type request environments. Computer Networks, 2010, 54(5): 811-825

[28] Ostovari P, Khreishah A, Wu J. Broadcasting with hard deadlines in wireless multichip networks using network coding. Wireless Communications and Mobile Computing, 2015, 15(5): 983-999

[29] Lim J H, Naito K, Yun J H, et al. Revisiting overlapped channels: Efficient broadcast in multi-channel wireless networks//Proceedings of the IEEE International Conference on Computer Communications. Hong Kong, China, 2015: 1984-1992

[30] Gao X, Yang Y, Chen G, et al. Global optimization for multi-channel wireless data broadcast with AH-tree indexing scheme. IEEE Transactions on Computers, 2016, 65(7): 2104-2117

[31] Nawaz Ali G G M, Lee V C S, Chan E, et al. Admission control-based multichannel data broadcasting for real-time multi-item queries. IEEE Transactions on Broadcasting, 2014, 60(4): 589-605

[32] Tang D, Cai Y, Zhao J, et al. A quantum-behaved particle swarm optimization with memetic algorithm and memory for continuous non-linear large scale problems. Information Sciences, 2014, 289(24): 162-189

[33] Yang Y, Zhu J. Write skew and Zipf distribution: Evidence and implications. ACM Transactions on Storage, 2016, 12(4): 21

《计算机学报》

HU Wen-Bin, born in 1976, Ph. D., professor, Ph. D. suppervisor. His main research interests include intelligent simulation and optimization, multi-agent systems, and swarm intelligence algorithm.

QIU Zhen-Yu, born in 1992, Ph. D. candidate. His main research interests include complex networks and data broadcasting.

NIE Cong, born in 1994, M. S. His main research interests include intelligent traffic and data mining.

WANG Huan, born in 1989, Ph. D. candidate. His main research interests include social network analysis and data mining.

YAN Li-Ping, born in 1980, Ph. D. candidate. Her main research interests include social network analysis and data mining.

DU Bo, born in 1983, Ph. D., professor, Ph. D. suppervisor. His main research interests include complex network and social network analysis.

Background

Data broadcasting has the advantage of channel sharing technology to allow an arbitrary number of clients to access data simultaneously, is a popular solution in mobile communication for propagating public information. Currently, a great number of researchers have paid much attention to the area of the data schedule in on-demand data broadcast. Various scheduling algorithms have been proposed to reduce the tuning time, access latency and request loss rate and have made great achievements. However, the excessive client requests lead to the uplink bandwidth becoming a bottleneck and the server facing huge parallel processing pressure, and no scheduling algorithms have been presented that can balance the pressure of the uplink channel and server and the broadcast efficiency. For this predicament, this paper proposed a request pre-process method (RPPM) for on-demand data broadcast. The main idea of RPPM is merge the clients' requests based on the hierarchical directory tress to reduce the number of requests. The merge of requests will lead to the generation of invalid data and the losing of request priority, thus, an optimal request merge algorithm (ORM) and a merged request priority and prune algorithm (MRPP) are include in the proposed method. We first introduce the preliminaries of on-demand data broadcast, then detail the ORM for request merge and MRPP for merged request priority evaluation and prune. To maximize the performance of the broadcast system, an overall scheduling scheme for on-demand data broadcast based on RPPM has been proposed also. Finally, the mathematics analysis and simulations carried out based on a real data set demonstrate the RPPM can minimize the number of requests, reduce the pressures of uplink channel and server, and improve the utilization of system resource.

This work is supported by the National Natural Science Foundation of China under Grant Nos. 61572369 and 61711530238. And supported by the Nature Science Foundation of Hubei Province No. 2015CFB423 and Major Science and Technology Plan Projects in Wuhan City No. 2015010101010023.

《计算机学报》