

发布/订阅系统中的历史数据分布式存储算法

黄 强

(南京电子技术研究所, 江苏 南京 210039)

摘 要: 在分析当前发布/订阅系统在历史数据存储方面相关工作的基础上, 提出了一种利用系统中的存储代理对历史数据进行分布式存储的算法. 算法将存储代理组成覆盖网, 利用一致性哈希定位主题元数据位置, 主题数据分段存储在存储设备上, 并采用文件索引、并行读取等方法来提高数据存取性能. 在对比实验中, 验证了算法在负载均衡和扩展性方面的优势.

关键词: 发布/订阅系统; 历史数据; 分布式存储

中图分类号: TN402

文献标识码: A

文章编号: 1000-7180(2014)09-0138-05

The Historic Data Distributed Storage Algorithm in Publish/Subscribe System

HUANG Qiang

(Nanjing Research Institute of Electronics Technology, Nanjing 210039, China)

Abstract: Based on the summarization of current research work on historic data storage in P/S systems, the paper presents a distributed algorithm to store historic data in P/S systems by proxies. The proxies are organized to form an overlay network. Consistent hashing is used to locate metadata agent, and topic data are stored by segments. Topic file indexing, parallel reading and other methods are applied in order to improve the performance. In comparative experiments, the proposed algorithm outperforms in load balancing and scalability.

Key words: publish/subscribe system; historic data; distributed storage

1 引言

发布/订阅 (Publish/Subscribe, 简称 pub/sub 或 P/S) 系统是一种使分布式系统中的各个参与者能以发布/订阅的方式进行交互的中间件系统^[1].

传统的 P/S 系统主要侧重于消息匹配、路由算法、可扩展性等问题, 涉及到历史数据存储方面的问题都被简化处理, 如仅缓存订阅者离线期间的数据^[2]、利用外置的数据库存储数据^[3]、只缓存系统最近一段时间的数据^[4]. 其中仅文献[3]支持历史数据的永久可访问, 但当系统数据存取量、主题数以及代理数量增加时, 使用该策略的系统会出现明显的负载不均衡, 导致部分节点过热、系统性能严重下降等问题, 可扩展性遭到严重的制约. 因此, 本文提出一种算法利用发布/订阅系统中的代理对历史数据进

行分布式存储, 支持大规模主题数据存取, 最后用实验仿真证明算法良好的负载均衡和可扩展性.

2 历史数据分布式存储算法

2.1 系统模型

在 P/S 系统中, 由一组订阅者和发布者组成. 一个物理节点上可以同时存在多个订阅者或者发布者. 每个物理节点上配有一个存储代理, 用于负责订阅者或者发布者与存储系统之间的交互. 本文使用一致性哈希算法 Chord^[5]将存储代理组成环状覆盖网络, 根据存储代理名的 SHA-1 哈希值确定每个存储代理在环中的位置, 订阅者或者发布者根据主题哈希值定位负责该主题历史数据存储的存储代理. 一致性哈希方法较好地解决了如何在动态的网络拓扑中分布存储和路由的问题, 每个节点仅需维护少

量相邻节点的信息,并且在节点加入/退出系统时,仅有相关的少量节点参与到拓扑的维护中。

存储代理内部的系统架构如图1所示,架构底层的网络管理模块负责管理存储代理的网络资源以及维护与所有存储代理的通信连接。存储代理的存储管理具有两个功能模块:元数据管理以及数据存储。元数据管理模块负责存储主题历史数据的元数据(各个时间段的主题数据存储所在的存储代理位置等信息)、指定当前主题数据的存储节点以及响应客户端的查询请求;数据存储模块负责存储实际的历史数据、生成历史数据的时间索引以及负责与客户端的数据传输。

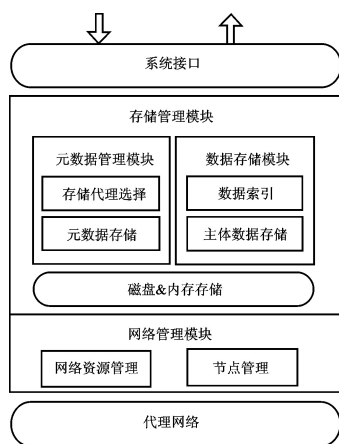


图1 存储代理结构

如图2所示,客户端的存取操作分成两个步骤:通过主题名的SHA-1哈希在环上定位该主题的元数据管理信息位置;然后,客户端根据相应响应信息连接指定的存储代理获取(存储)该主题的数据。

2.2 本地文件索引

本地文件存储格式如图3所示,用户的每条主题数据在存储代理上都会标有一个时间戳,同一主题的数据顺序存储在同一文件内。由于用户根据时间区间请求数据,在对文件进程查找时,需要从文件头开始顺序扫描,在文件较大的情况下会产生很大的无用磁盘操作。本文在文件头部加入索引部分,对文件每隔一段长度生成一个“关键帧”,将此“关键帧”的时间戳以及偏移量存在索引内。为了保证文件在不停追加的时候可以同时生成关键帧,需要将头部索引长度固定,本文中的索引长度设为1024,在每隔1MB生成一个索引的条件下可以满足1GB文件的存储。存储代理上的读取操作根据请求的时间标志查找文件头的索引,将文件指针定位到索引中时间段最接近请求时间的偏移量部分,然后通过小范围顺序扫描定位到客户端请求的数据。

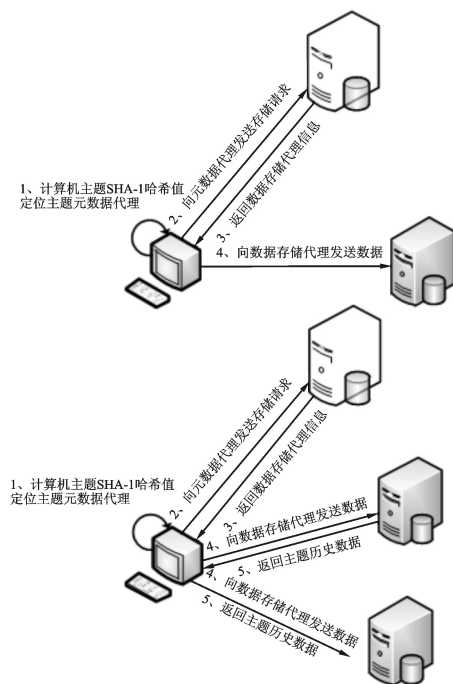


图2 客户端存取过程

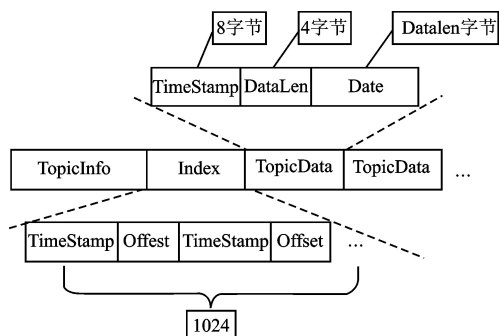


图3 历史数据文件格式

2.3 数据段长度划分

在主题数据的存储中,如果只选定一个代理作为特定主题数据的存储节点,当该主题一直有数据产生时,随着时间的推移,最终的数据容量会超过代理的存储上限,所以需要选择多个存储代理对主题数据分段存储。在数据分段存储中的一个关键问题是数据段长的设置,在本文的研究环境中,还需要考虑主题数据的时间跨度问题。如果采用默认大小分段,当主题数据产生速度过高,则会频繁变更代理存储节点;当主题数据产生速度过低,时间范围很广的数据都会存储在一个节点,进而会有大量的读取请求定位到这一存储节点。因此,本文引入函数 $W = \text{Interval} \cdot \text{Len}$,其中Interval代表数据段的时间跨度,Len代表数据段的长度,W为一固定值,当 $\text{Interval} \cdot \text{Len}$ 的值大于W时,选择其他代理存储数据。从函数特性可以看出,当数据产生速率较高时,

虽然很快会产生较多的数据,但由于时间跨度很短,两者最终的乘积不会过大,这样在数据产生速率过高时存储代理更改次数不会太频繁;反之,在数据产生速率较低时,也不会因为产生的数据很少而不更改存储代理。

2.4 存储节点选择

在 P/S 系统中,由于客户端能力有限且数量众多,并且网络资源非常宝贵,因此要求存储算法尽量减少网络带宽的占用,同时又尽量避免热点问题. 存储代理选择伪代码参见图 4. 系统中所有的存储代理根据自身能力具有等比例的健康值 h_i 和恢复能力 h_i/ϵ , i 为存储代理编号, ϵ 为系统可配值,默认取值为 10. 存储客户端所在存储代理的健康值以及恢复能力是默认值的 n/θ 倍, n 为代理总数, θ 为系统可配值,默认取值为 5. 各个存储代理的健康值被顺序映射到同一区间,元数据管理模块在选择存储代理时生成区间内的一个随机数,随机数所在的健康值区间的存储代理将会被选作为存储代理. 被选择到的存储代理完成存储数据 i 后,其健康值会急剧下降到 h_i/ϵ ,下一轮被选中的概率将会变低. 那些未被选择到的存储代理,其健康值会恢复 h_i/ϵ ,直到到达初始的健康值.

2.5 数据并行获取

本文中主题的历史数据采用分段存储的形式,元数据管理返回的客户端数据获取请求的信息可能包含多个存储代理. 这时客户端可以采取并行的方式从多个存储代理上同时获取历史数据. 如果客户端应用需要顺序的历史数据,可以将时间段靠后的数据缓存在本地,等前面的数据读取完毕后再交付给应用程序,这样可以充分发挥出历史数据分段存储的优势.

3 仿真实验结果

3.1 参数取值对系统性能的影响

本小节采用实验的方式分析存储代理选择算法中的参数取值对系统性能的影响. 实验采用 1 000 个存储代理和 1 个主题,存储代理对于主题存储行为符合正态分布,生成 10 000 个主题存储请求,在不同存储代理恢复轮数 ϵ 和客户端存储参数 θ 的实验结果中,记录主题存储请求本地化的次数和每个存储代理上主题存储的次数,计算本地存储概率和代理间存储次数的方差. 实验结果如图 5 所示. 当 ϵ 取值较小时,本地存储概率很高,但存储代理间存储方差较大,反映出存储代理负载的不均衡. 当 ϵ 和 θ 取较大值时,本地化概率和随机选择的概率基本一致,大部分数据存储需要通过网络

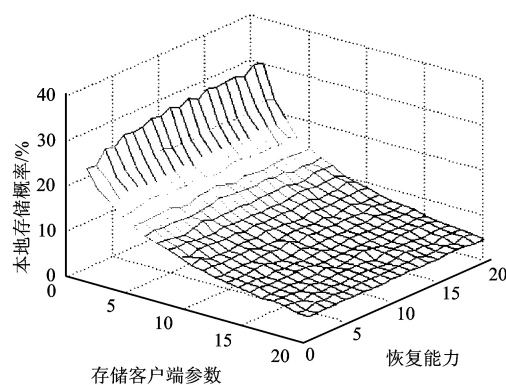
```

FindStoreProxy(topic)
1.  hsum = 0
2.  for i = 1 to n
3.  //如果代理是请求节点,健康值、恢复力增大
4.    if ( proxy[i]. ClientStoreBegin(topic) )
5.      hmax[i] * = n/theta
6.      r[i] * = n/theta
7.  //如果代理不再作为请求节点,对应参数减小
8.  if ( proxy[i]. ClientStoreEnd(topic) )
9.    hmax[i] /= n/theta
10.   r[i] /= n/theta
11. //代理没有该主题的存储将恢复健康值
12.   if ( ! proxy[i]. ProxyStore(topic) )
13.     if ( h[i] < hmax[i] )
14.       h[i] += hmax[i]
15.       if ( h[i] > hmax[i] )
16.         h[i] = hmax[i]
17.   hsum += h[i]
18.   randH = random( 0, hsum )
19. //根据生成的随机值确定代理 id
20. id = GetIndexByH( randH, h, n )
21. //存储代理由于承担存储任务受伤
22.   h[id] = r[id]
23.   return id
end

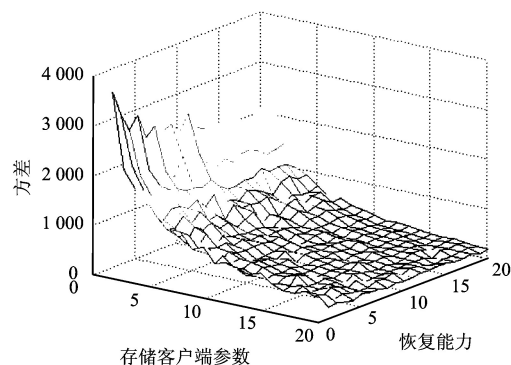
```

图 4 存储代理选择算法

转发到存储代理,占用大量系统的网络资源. 选取 2.4 节中的默认值($\epsilon=10$, $\theta=5$)时,可以增加本地存储概率,节省网络资源使用,存储代理负载处于较为均衡水平,防止出现热点问题.



(a)本地存储概率统计图



(b)代理间存储方差统计图

图 5 ϵ 和 θ 取不同值的命中概率和方差

3.2 算法性能比较

本文与 PADRES 的静态划分策略进行对比. PADRES 的静态划分策略将主题随机划分到存储节点,客户端读写请求直接定位到对应节点.假设主题访问的热度服从 Zipf-like^[6]分布,参数 $\alpha=1.0$.

本小节在不同存储代理节点数、主题数以及静态划分存储节点数条件下进行四组对比实验,对比参数包括存储代理数据传输量、平均队列长度和平均响应时间.数据传输量表示存储节点传输的数据总量,包括读取与存储;平均队列长度表示存储代理任务队列的平均长度,可反映否出现热点;平均响应时间表示请求从发出到存储代理执行并返回数据的平均时间.

实验中设置 10 个存储代理,主题数为 100. 实

验结果如图 6 所示,其中图 6 上行中指定前三个代理为 PADRES 静态存储代理,图 6 下行中所有代理为 PADRES 静态存储代理.当静态存储代理数为 3 时,系统中的任务传输量大部分集中在存储代理节点,存储代理节点的平均队列长度明显高于普通代理,所有代理请求的平均响应时间超过 170 ms. 本文算法中节点在数据传输量、平均队列长度、带宽利用率方面较均衡,平均请求时间都在 50 ms 以内.当系统中所有代理都为静态存储节点时,虽然各个代理负责均匀数量的主题,但由于主题数据的访问特性并不均匀,导致部分代理节点数据访问负载过重,如图 6 下行中,代理节点 1 的负载明显较高.

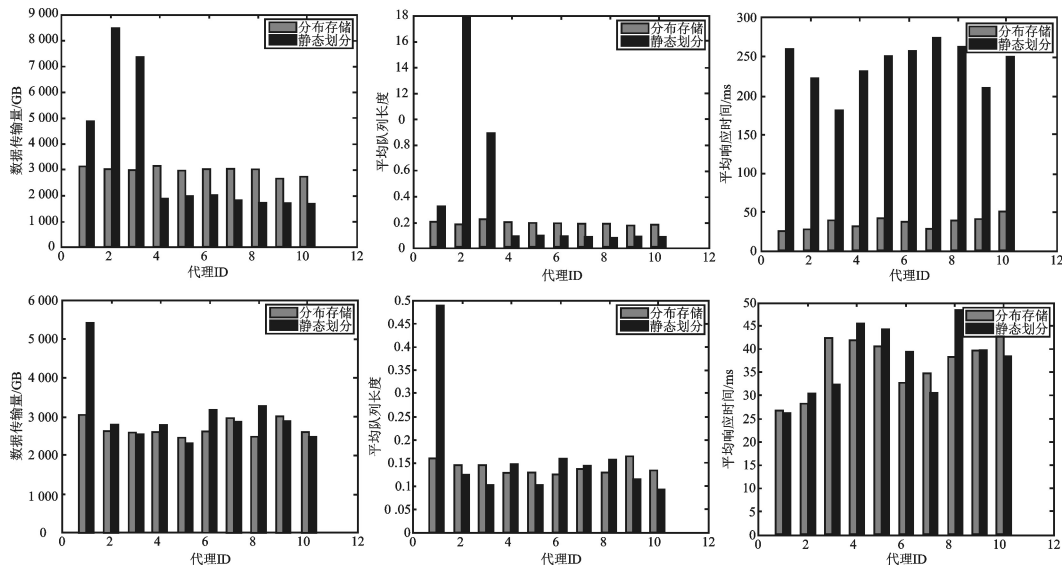


图 6 代理数为 10 的对比结果

当代理数增到 100、PADRES 静态存储代理数为 10、主题数量增至 1 000 时,实验结果如图 7 上行所示.由于 PADRES 静态存储代理的数据传输量剧增,系统基本处于崩溃状态,部分存储代理的平均队列长度超过 1 000,响应时间增至数十秒,而采用本文算法的系统一直保持良好的工作状态,系统中各个节点负载较均匀,响应时间在 100 ms 以内.当代理数增加到 1 000, PADRES 静态存储数为 10,主题数量增至 10 000 时,为展示方便,图 7 下行只展示前 100 个代理的统计指标.普通代理以及本文算法存储代理的带宽利用率一直保持在较低水平,而静态存储代理负载量相当大,每个存储代理的数据传输量比普通代理要高出数十倍,同时静态存储代理的利用率相当高且波动很大.本文算法在响应时间上明显优于静态存储策略.

4 结束语

本文针对发布/订阅系统中历史数据存储问题提出了一种新的分布式存储历史数据的算法.仿真实验表明,在节点数过多时,本文算法相对传统的静态划分算法具有明显的优势.本文算法并未考虑代理失效、代理频繁加入退出等情况,所以未来还需要在可靠性、负载迁移等方面进行深入的研究.

参考文献:

- [1] P T Eugster, P A Felber, R Guerraoui, et al. The many faces of publish/subscribe [J]. ACM Computing Surveys, 2003(35):114-131.
- [2] Caporuscio M, Carzaniga A, Wolf A L. Design and evaluation of a support service for mobile, wireless publish/subscribe applications[J]. IEEE Transactions

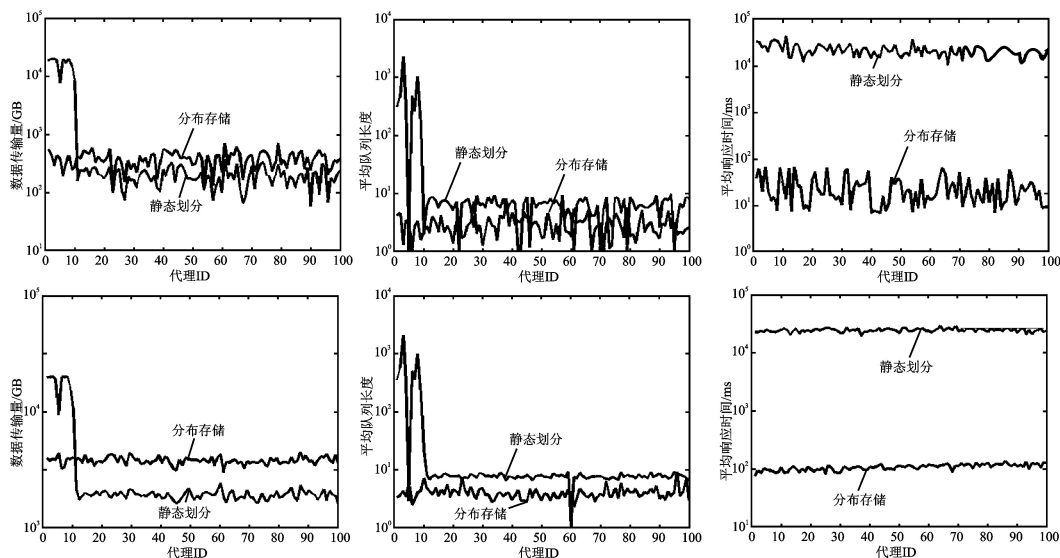


图 7 代理数为 100 和 1 000 情况下的对比结果

- on Software Engineering, 2003, 29(12): 1059-1071.
- [3] Li G, Cheung A, Hou S, et al. Historic data access in publish/subscribe[C]// Proceedings of ACM International Conference on Distributed Event-based Systems. [s. l.]:IEEE, 2007: 80-84.
- [4] Sourlas V, Paschos G S, Flegkas P, et al. Caching in content-based publish/subscribe systems [C]// Proceedings of GLOBECOM. Honolulu, HI:IEEE, 2009: 1-6.
- [5] Stoica I, Morris R, Karger D, et al. Chord: A scal-

able peer-to-peer lookup service for internet applications[J]. ACM SIGCOMM Computer Communication Review, 2001, 31(4): 149-160.

[6] Breslau L, Cao P, Fan L, et al. Web caching and Zipf-like distributions: Evidence and implications[J]. INFOCOM, 1999(1):126-134.

作者简介:

黄 强 男, (1977-), 博士, 高级工程师. 研究方向为雷达数据处理、多传感器数据融合、目标跟踪.

(上接第 137 页)

- [2] 齐爱琴, 申永军. 基于天线传感器网络的双因子认证协议的设计与研究[J]. 微电子学与计算机, 2012, 29(1): 121-123.
- [3] 张文丽. 基于 Hash 函数的 RFID 安全认证协议的设计[J]. 电脑与信息技术, 2012, 20(2): 56-59.
- [4] Sarma S, Weis S, Engels D. RFID systems and security and privacy implications [C] // Proc of the 4th Int Workshop on Cryptographic Hardware and Embedded systems. Berlin: Springer, 2002: 454-469.
- [5] Rivest L, Weis A, Sarma E, et al. Security aspects of low-cost RFID systems [C] // Proc of the 1st Security in Computing. Berlin: Springer, 2003: 202-212.
- [6] OhKubo M, Suzuki K, Kinoshita S. Hash-chain based forward secure privacy protection scheme for low-cost RFID[C] // Proceedings of the 2004 Symposium on

Cryptography and Information Security. Sedai: IEEE, 2004: 719-724.

- [7] 王娟. 基于散列函数的 RFID 认证协议研究[D]. 南京: 南京邮电大学, 2011.
- [8] 伍新华, 唐翠婷. 一种基于 Hash 的 RFID 双向认证协议[J]. 武汉理工大学学报: 交通科学与工程版, 2011, 35(3): 571-574.
- [9] 李慧贤. 轻量级 RFID 双向认证协议设计与分析[J]. 西安电子科技大学学报: 自然科学版, 2012, 39(1): 173-178.

作者简介:

龚 晶 女, (1968-), 副教授. 研究方向为信息安全与电子技术