DOI：10.3969/j.issn.1001-8972.2011.20.044
基金项目：无线创新校园网络体系结构及关键技术研究项目资助。

MongoDB 数据库的应用研究和方案优化

王光磊　　北京邮电大学网络教育学院,北京 100876

MongoDB database of applied research and program optimization

Wang Guanglei
Institute of Educational Technology Beijing University of Posts and Telecommunications, BUPT Beijing, China

摘　要

MongoDB 作为一款性能优良，功能丰富，支持海量数据存储的产品受到很多商家的青睐。但由于 MongoDB 系统中采用的 Auto-Sharding 的算法存在着数据在各个节点上的分配不均匀的现象，使 CPU 占有率过高，直接影响了系统的性能。通过对 MongoDB 的研究分析，提出了用一致性哈希算法进行优化的方案，设计了一个针对海量数据存储的分布式文件系统，以有效解决数据分配不均匀的问题。

中图分类号:TP333　文献标识码：A

关键词

海量数据;MongoDB;哈希算法

Abstract

MongoDB as a good performance, feature - rich, supporting massive data storage products are favored by many businesses. However, because of Auto-Sharding algorithmthere which is used by MongoDB, the data on each node is uneven distributed, the CPU share is too high, there is a direct impact on system performance. Based on the research and analysis of MongoDB, we use the improved hash algorithm to optimize the program, design a distributed file system for mass data storage, and effectively address the problem of uneven distribution of data.

Key words

Massive data；MongoDB；Hashing Algorithm

概述

随着互联网技术的飞速发展，数据量呈现爆炸性增长趋势，对互联网中应用的数据库提出了高并发读写、高效率存储以及高可扩展性和高可用性的需求，传统数据库越来越显现出力不从心。MongoDB 以其独特的优势为 WEB 应用提供了可扩展的高性能数据存储解决方案。MongoDB 的主要特点是开源设计、高性能、易部署、易使用，存储数据方便，它的面向集合存储、模式自由、支持动态查询、支持完全索引等优势正在被越来越多的公司和个人所接受。

MongoDB 是一个介于关系数据库和非关系数据库之间的产品，是非关系数据库当中功能最丰富，最像关系数据库的。它支持的数据结构非常松散，是类似 json 的 bjson 格式，因此可以存储比较复杂的数据类型。

MongoDB 有一个强大、灵活和可扩展数据存储区。它结合了能力 扩展与关系数据库的许多最有用的功能，如次索引范围查询和排序。MongoDB 也具备十分强大的分片功能，如内装的支持处理样式聚集和空间的索引。

但 MongoDB 在机制上仍属于 NoSQL，由于 NoSQL 的使用经验缺乏、CPU 占有率过大等缺点也制约了 MongoDB 的发展。

本文首先分析了 Auto-Sharding 在 MongoDB 的机制，并对 MongoDB 的并发吞吐量和 CPU 占有率进行了测试分析，提出了 Mongodb 分片机制的优化方案，作为一种抑制 MongoDB 在大数据量情况下 CPU 占有率过高的解决方法。

1 Auto-Sharding 在 MongoDB 的应用

Sharding 是 MongoDB 扩展负载的方法。Sharding 可以在不影响程序运行的情况下添加更多硬件设备来满足数据的需求。

MongoDB sharding 的基本概念是将 collection 划分为较小的 chunks，这些 chunks 分布在 sharding 上。我们不需要知道数据存储在那个 sharding 上，只需要运行 mongos，由 mongos 知道所有数据所在，所以应用程序可以正常连接到该数据库并发出请求。当有请求时，只需要通过 mongos 就知道 chunks 的存储位置，mongos 收集数据并将其发送给应用程序。（如图 2），将 sharding 与应用程序分离，在这种机制下，用户可以不必更改程序就可以扩展系统。

MongoDB 中分片机制提供如下功能：

1）、当数据量和负载个数发生变化时，分片机制可以自动平衡负载和数据分布的变化

2）、分片机制可以灵活的添加新的硬件设备

3）、扩展性能强，最大可以扩展到一千节点

4）、可以做到无单点故障

5）、出现故障时，利用 sharding 可以实现自动故障转移

Auto-Sharing 的架构图如图 1 所示

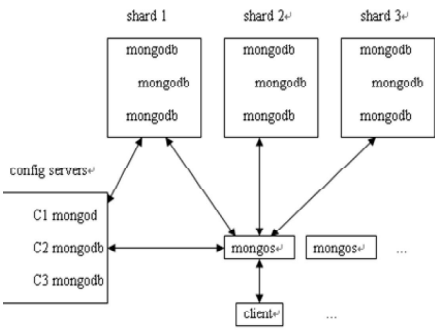


图　1　Auto-Sharing 的架构图

Auto-sharding，是通过 mongos 的自动分片功能建立一个水平扩展的数据库集群系统，将数据存储在 sharding 的各个节点上。

Auto-sharding 功能的最大特点是能够自动的分片。通过这一功能，用户只需要指定数据的某个字段值作为分片的 key，数据会自动平均分配到后端结点。

当增加结点时，sharding 能够自动对数据进行分片，对整个系统进行负载平衡。

2 MongoDB 的性能测试

2.1 MongoDB 并发性的测试

为了说明 MongoDB 的处理大数据量的并发性能，我们对在相同的配置环境下，MongoDB 和 MySQL 的并发性能测试结果进行比较分析。图 2 中横坐标表示并发数，纵坐标代表吞吐量，不同颜色的折线分别表示在查询、插入和变更状态下 MongoDB 和 MySQL 不同的吞吐量的变化趋势。从图中可以看出：

1）、mongdb 随着并发数的增加，无论是 insert 吞吐量、update 还是 select，其吞吐量在不同程度的下降，但较之 Mysql 的性能会好得多。

2）、在 insert 及 update 下，Mongodb 吞吐量大约是 Mysql 的 2~3 倍，select 下，MongoDB 大多也优于 Mysql。因此,Mongodb 的总并发性能比 Mysq 要好。这也是很多商家选择 MongoDB 原因之一。

2.2 MongoDB 的 CPU 占有率的测试

CPU 占有率是数据库选择另一个重要指标之一。本文通过对 HandlerSocket 和 MongoDB 的 CPU 占有率性能的测试，进行比较分析，HandlerSocket 与 MongoDB 一样，也是一个 NoSQL 产品。在一百万数据量情况下，HandlerSocket 和 MongoDB 的 CPU 占有率性能如图 4。

由图 3 可知，在大数据量测试情况下，Mongodb 的 CPU 占有率几乎都接近 100%，而 HandlerSocket 的 CPU 占有率只保持在 40% 左右。

分析占用空间过大的主要原因如下：

空间的预分配问题：为避免形成过多的硬盘碎片，mongodb 每次空间不足时都会申请生成一大块的硬盘空间，而且申请的量从 64M、128M、256M 呈指数递增，直到 2G 为单个文件的最大体积。随着数据量的增加，在其数据目录里可以看到这些整块生成容量

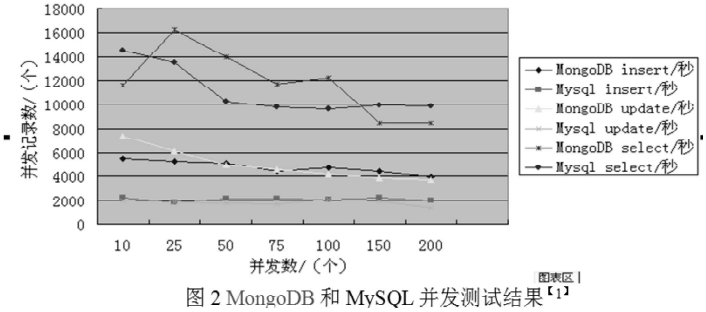


图 2 MongoDB 和 MySQL 并发测试结果[1]

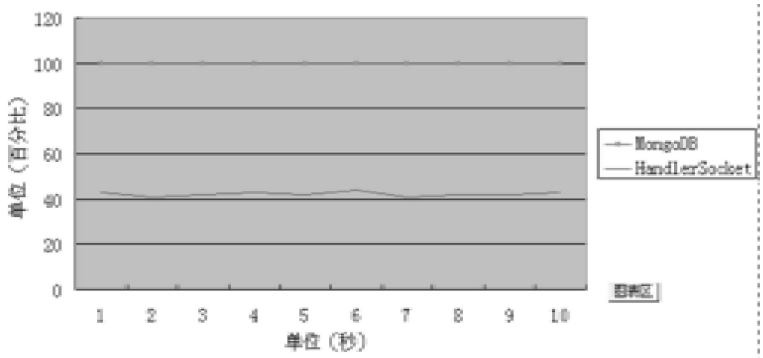


图 3 MongoDB CPU 占有率的线性图

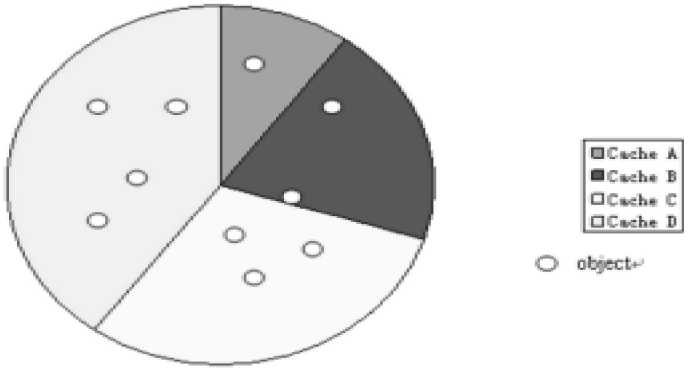


图 4 哈希算法示意图

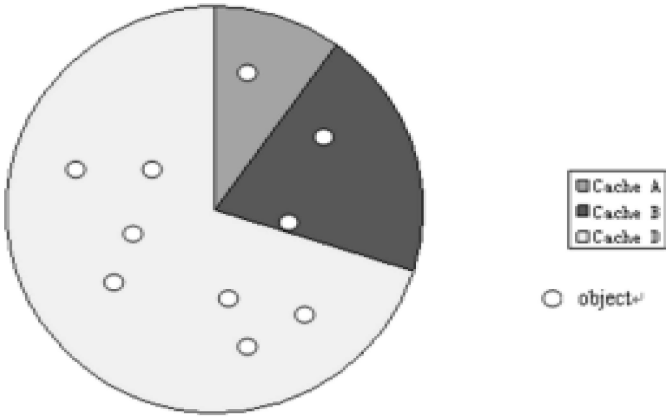
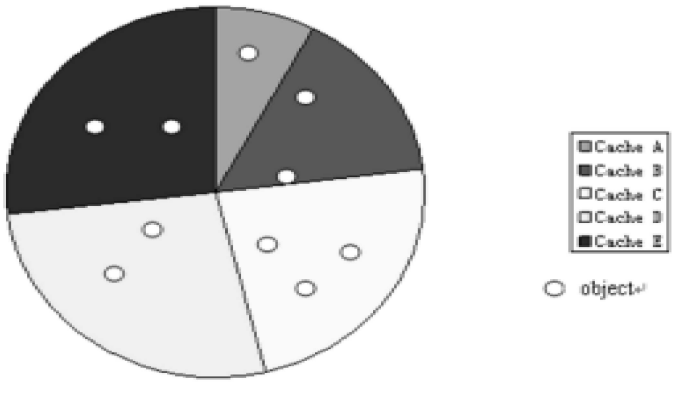


图 5 删除 Cache C 的示意图



图六 删除 cacheE 的示意图

不断递增的文件。而空间的预分配是由 MongoDB 的 Auto-Sharding 机制决定的。

MongoDB 的 Sharding 在存储数据时将数据进行了分块存储（Chunk），每一块的大小默认为 200M。而 Sharding 中每一次数据迁移的最小单元就是数据块。

MongoDB 后台有一个叫做 Balancer 的后台程序，会检查各个 Sharding 结点的分布情况，当发现不同结点间 Chunk 数相差达到一定大小时，就开始进行数据迁移，以平衡数据在各个 Sharding 结点的分布。Balancer 迁移数据的过程是，先把要迁移的数据从磁盘读到内存，再进行迁移。

但当数据量比较大时，通常需要迁移的数据可能并没有被加载到内存里，这时 0 会先把内存中的热数据踢除，以加载从磁盘读取的需要迁移的数据，然后在迁移完数据后再删除掉原来结点上的数据。在这过程中，由于热数据被踢回磁盘中，会严重影响系统的性能。

3 MongoDB 的改进

针对 mongodb 的缺点，本文采用哈希算法对数据进行分配，从而避免 mongodb 占用空间过大的问题。

哈希算法一般分两步进行。首先求出 cache 的哈希值。接着计算 object 的哈希值，如果 object 的哈希值与某个 cache 的哈希值匹配，则将其映射到该 cache 上。哈希算法的示意图如图 5 所示

由图 4 可知：

Cache A 上的存储对象有 object 1；

Cache B 上的存储对象有 object 2、object 3；

Cache C 上的存储对象有 object 4、object5、object6；

Cache D 上的存储对象有 object 7、object8、object9、object10；

3.1 Cache 的排序规则和优化方案

当 Cache C 删除时，原本存储在 Cache C 上的 object 会转移到 Cache D 上，造成 Cache D 的负载过高。同样当有 Cache E 添加时，由于新增的 Cache E 的哈希值不确定，新增的 Cache E 可能落到负载少的 Cache A 和 Cache D 之间，使得 Cache A 的负载为零，造成资源的浪费。

根据以上问题，我们提出一个优化方案。根据位于在 Cache 上的对象个数和其对象访问量的大小，对 Cache 进行排序。当有 Cache 删除时，将位于被删除的 Cache 上的 object 转移到最小的 Cache 上。当有 Cache 加入时，将 Cache 添加到最大的 Cache 上，从而达到负载的平衡。假设每个 object 的访问量相同，根据图 4 可知，Cache 的排序从大到小为：Cache D、Cache C、Cache B、Cache A。

假设每个 object 的访问频率相同，根据图 4 可知，Cache 的排序从大到小为：Cache D、Cache C、Cache B、Cache A。当 Cache C 删除时，根据 Cache 的排序大小，将原本位于在 Cache C 上的 object 转移到 Cache A 上，避免负载压力过大，如图 5 所示。

当有 Cache 添加时，由于 Cache D 上的负载最大，应该将新 Cache 放置在 Cache C 和 Cache D 之间，这样可以减小 Cache D 的负载压力，如图 6 所示。

3.2 存储对象排序的优化方案

当添加或删除 cache 时，由于 cache 数量的变化，其余 cache 的哈希值会发生变化，同时存储对象的哈希值也会发生变化，对象 a

下转第 96 页 ▶▶

以 1.00 元累计；2.3 公里以上，35 公里以下时，在 2.3 公里计费的基础上以每公里 2.6 元累计；35 公里以上时，在前面计费基础上以每公里 3.5 元累计。

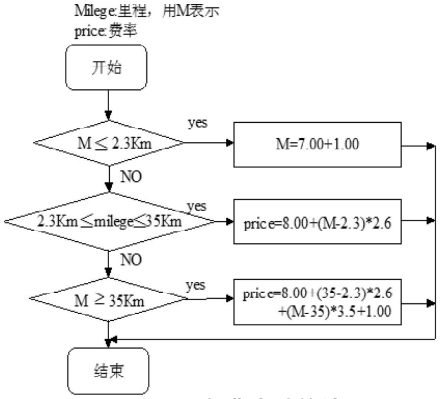


图 3 里程识别与费率计算流程图

2.2 费率计算程序实现

```
void price()
 {
  if(RunMilege<=2300)
     //TotalPrice=7.00+1.00;
     TotalPrice=8.00;
  if((RunMilege>2300)&&(RunMilege<=35000))
     //TotalPrice=700+(RunMilege-2300)*0.26+100;
  TotalPrice=800+(RunMilege-2300)*0.26
     if(RunMilege>35000)
  //TotalPrice=700+(35000-2300)*0.26+(RunM ilege-35000)*0.35+1.00;
  TotalPrice=1112+(RunMilege-35000)*0.35;
        Price=(ulong)TotalPrice;
  }
```

3 结论

出租车计费系统融合了信息识别技术和计算机测控技术，为多功能、智能化系统，其功能可较好地满足实际应用。由于采用了按里程智能识别的实时计费系统，在软件设计时，有多种方案可供选择。经过设计、仿真，实现了主控部分计费功能，对于外围功能扩展部分，如车票资料打印、IC 卡付费和电脑串行通信等功能，有待于进一步开发和完善。系统经过验证表明：其性能稳定可靠、性价比高、安装简便、有良好的应用效果和广阔的应用前景。

参考文献
[1]张鑫,华臻.陈书谦.单片机原理及应用[M].电子工业出版社.2005
[2]丁元杰,吴大伟.单片微机习题集与实验指导书[M].机械工业出版社.2004
[3]李汉玲,熊再荣.基于SOPC技术的出租车计费系统的设计与实现[J],工业控制计算机.2009，22(9):26～27
[4]吴冬梅,吴延海,邓玉玖. 基于CPLD/FPGA的出租车计费器[J].电子技术应用.2004第11期:71～73
[5] 王飞,张友鹏,贺宁. 基于FPGA与MSC-51的出租车计费系统设计[J].电子元器件应用.2007,9(9):42～44

作者简介
王晓利，(1975－)，女（汉），河南舞阳人，硕士研究生，讲师，主要研究方向为电子通信、信息处理。

上接第 86 页

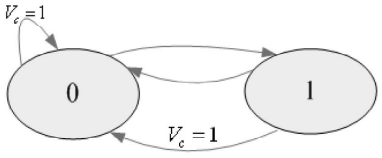


图 6 数控振荡器状态转移图

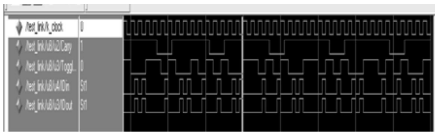


图 7 数控振荡器仿真波形

N 分频控制器将数控振荡器的输出信号频率进行 N 分频，分频因子的获得是由高频信号对输入信号进行采样，可以根据输入信号的改变而变化，通过采样比可以改变分频因子的值，实现对锁定范围的调整。

3、系统仿真与实现

本文设计的数字锁相环，使用 Verilog 硬件语言进行模块描述，采用 XILINX 公司的 ISE 软件进行综合，目标芯片为 XILINX 公司的 SPARTAN2 系列的 XC2S100，使用了 120 个 LUT 和 32 个触发器单位。

采用 Modelsim 6.5 进行时序仿真验证，仿真过程中输入时钟为 100MHz，M=16，k=4。仿真结果如图 8 所示。

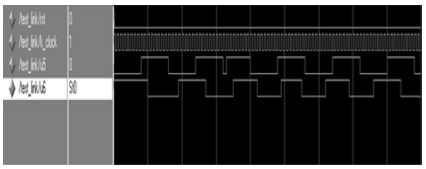


图 8 系统仿真图

由图 8 我们不难看出，当输入信号频率出现抖动时，经过数字锁相环后，经过 40 个时钟周期，相位重新锁定（输入信号与反馈信号相位相差为 90°）。

4 结论

本文详细介绍了数字锁相环的工作原理和基于 FPGA 的设计方法，利用 Verilog 语言进行了建模与描述，并利用 XILINX 公司的 ISE 软件进行了综合，并下载到了目标芯片，完成了一种基于 FPGA 的数字锁相环的设计与实现。仿真结果表明，该设计具有实现方便、稳定性高、易于移植重构和锁定速度快等优点，为工程应用提供了极大的便利和性能保证。

参考文献
[1]Roland E Best. Phase-Locked loops Design, Simulation,and Application[M].清华大学出版社.2003.12
[2]夏宇闻.Verilog 数字系统设计教程[M].北京航空航天大学出版社.2003.7
[3]谢稷宏.全数字锁相环的设计[J].电子设计应用.2004. 2
[4]张厥盛，等.锁相技术[M].西安电子科技大学出版社.2002

作者简介
李冠贤(1981-) ,男, 广西桂林人,大学本科,研究方向: DSP、FPGA 设计与实现

上接第 94 页

因为哈希值的变化，从 cacehA 转移到 cacheB,而对象 b 可能因为同样的原因从 cacehC 转移到 cacheB,由于 a、b 之间没有查询优先的判断，对象 a、b 在 cacheB 的存储是随机的，这对系统的查询性能是不利的。

优化方案提出对存储对象被查询的概率进行排序，在 cache 发生变化时，发生冲突的对象根据被查询的概率大小，来决定存储的位置，通过这种措施，实现系统查询速度的提高。

1)在哈希表的结构中，添加键频(Count)项，即概率键 ,在建立哈希表添加元素时直接和所在链表处的其他对象所对应的键频(Count) 进行对比排序；

2) 将改进后的存储对象加入 hash 算法中

当 Cache C 删除时，原本位于在 Cache C 上的 object 会转移到 Cache A 上，坐落在 Cache A 上的 object 有四个，根据 Object 元素自身所带的频键，根据频键的大小来决定 Object 的存储位置。通过这个改进，可以实现对数据查询速度的优化。

4 总结

虽然 MongoDB 吞吐量性能很高，但是当出现海量数据的读取时，CPU 占有率过大，直接影响系统的性能，通过改进后的哈希算法，可以将存储的数据均匀的分布在节点上，实现数据均匀分片，减少系统对 CPU 的占有率，实现对系统的优化。该系统可以更好实现对海量数据的存储，对 Web 应用提供有力的支持。

系统的优化效果还需要进一步的试验验证和比对分析。基于 MongoDB 的云存储系统是一个新兴的研究领域,可以实现一个稳定、安全、高效的系统,还需要进一步研究和实践。

参考文献
[1]S.Shepler,B.Callaghan.RFC 3530:Network File System(NFS)version 4Protocol.The Internet Society, 2003
[2] Ghemaw at S, Gobioff H, Leu ng S T . T he google
f ile system/ / Proceedin gs of the 19th ACM Sym pos ium on OperatingSys t ems Principles . Sagamore, 2003: 29～43
[3] ZHU S, SET IA S, JAJOD IA S. LEAP: efficient securitym ech an ism s for large-scale d is-tributed sensor networks[C] / /Proc. of the 10th ACM Con fon Computer and Communications Security, 2003: 62-2
[4] Thomas Anderson, Michael Dahlin, Jeanna Neefe, David Patterson, Drew Roselli, and Randolph Wang. Serverless networkfil e systems. In Proceedings of the15th ACM Symposium on Operating SystemPrinciples, pages 109 – 126, Copper Mountain Resort,Colorado, December 1995
[5]唐箭.云存储系统的分析与应用研究.电脑知识与技术.2009，5(20):13～14
[6]KenDunham.TheProblemwithPZP. InformationSeeurityJournal:AGlobalPerspeetive，2006，15(2):5～8
[7]Kristina Cbodorow. MongoDB 权威指南
[8] Kelsey J, Kohno T. Herding hash functions and the nostradamus attack [C] .EUROCRYPT 2006. Berlin: Springer-Verlag, 2006, :183～200
[9] R.Merkle. One Way hash Functions and DES . Advances in Cryptology-Crypto ‘ 89. Springer-Verlag, 1990

作者简介
王光磊，26，年生，男，学历硕士，非中国计算机学会会员。