对云环境中改进Mongodb自动分片技术的研究

摘要

随着网际网路2.0技术的快速发展，大规模分布式服务的需求，给存储在云计算带来的巨大的挑战，动摇了传统的的关系型数据库。NoSQL数据库从RDBMS的桎梏中解放出来成为人们关注的焦点。在这篇文章中，首先本着自动的原则和实施机制，提出了MongoDB数据库分片技术，而后为了解决分布不均的问题提出一个在数据操作的频率的基础上的改进的数据自动分片算法. 改进的均衡策略能有效平衡之间的数据碎片，提高集群的并发读写性能。

关键词 NoSQL，MongoDB，自动分片，平衡策略

一、绪论

近年来，随着互联网数据量的快速增长和互联网的Web 2.0技术的发展，如何有效地存储，处理和提取的大量数据，成为一个迫切需要解决的问题，云计算在这种情境下应运而生。云计算通过网络（通常是因特网）[1]以计算而不是一个产品的形式作为服务的交付，从而共享资源，软件和信息提供给计算机和其他作为计量的服务设备。许多大学，供应商和政府组织各地投资于发展云计算的项目，例如：亚马逊推出简单存储服务（S3）和弹性计算云（EC2），谷歌推动的BigTable和MapReduce，GFS，已经全部成功地应用在生产环境中。分布式文件

系统可以组织大量的数据在云端，并且也能够有效地读取云数据，但我们为更好的结构化数据的管理，仍然需要专门的数据管理工具。

云数据管理是一个随云计算的发展产生的新的数据管理概念，它能够在云中的海量数据集有效地管理大型数据集，这使得快速找到特定的数据，以下常见的云数据管理的特点：

（1）高并发读写性能，

（2）有效地存储和访问大量的数据

（3）高可扩展性和高可用性要求

面对这些需求，传统的关系数据管理系统（RDBMS）遇到了难以逾越的障碍。因此，NoSQL数据库系统使用量大幅上涨。主要的互联网公司，如谷歌，亚马逊，Twitter和Facebook上有显着不同的挑战，在处理数据的RDBMS解决方案无法应对。这些公司意识到，性能和实时性比一致性更重要，而传统的关系数据库花费高额的处理时间来实现一致性。因此，NoSQL数据库通常高度优化检索和追加操作，而且往往以小功能超出记录存储。减少运行时的灵活性相比， RDBMS系统的显着收益补偿可扩展性和性能。通常情况下，NoSQL数据库根据他们的数据存储和下降的方式分类根据类别，如键 - 值存储（如迪纳摩[2]），BigTableimplementations[3] anddocumentstore数据库（例如MongoDB[4]）。然而，由于未成熟的技术云数据MANGEMENT，在实际生产环境中还是有很多需要解决的问题。

本文讨论MongoDB数据库的设计原则和，实现机制专注于自动分片的原则。自动分片的目标就是分裂整个机器的数据和实现重新平衡，使得能够存储更多的数据和处理而无需大或强大的机器。但平衡算法没有那么智能，数据是不是均匀均衡地分布在服务器之间。为了解决这个问题，提出了一个改善FODO（即频率的数据操作）算法。FODO算法是在数据操作的频率的基础上的，考虑服务器负载实现的。基于FODO算法的数据均衡策略能够有效地平衡服务器之间的数据，提高集群的并发读写性能。

二、 Mongodb中的自动分片机制

A 对Mongodb的简单介绍：

MongoDB的（从“堆积如山”一词而来）是一个开放源码，以C + +编程语言编写的面向文档的NoSQL数据库系统。管理集合BSON文档。 MongoDB的开发始于2007年10月通过10gen的介绍， MongoDB的特点[5]：面向文档存储，JSON风格的文档与动态架构提供简单

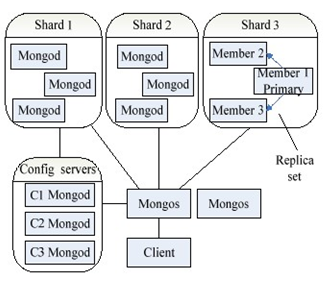
和完全索引支持，复制和高可用性，自动分片，水平缩放而不影响功能，Map / Reduce的，灵活的聚合和数据加工; GridFS，无需复杂的堆栈存储任意大小的文件。

MongoDB中文档是核心的概念，这是MongoDB的数据的基本单位，大致上，相当于在RDBMS中的一列。同样的，一个集合可以认为是相当于无架构的一个表​​。一个单一的MongoDB的实例可以承载多个独立的数据库，每一个都可以有自己的集合和权限。

B 自动分片机制

分片是指分割数据的过程中和存储的数据的不同部分在不同的机器上。由于分裂数据跨机器，能够存储更多的数据和处理更多的负载，而无需大型或强大的机器[6]。 MongoDB支持自动分片，从而消除了一些手工分片，集群拆分数据和自动平衡。 MongoDB的分片提供：（1）在负载和数据更改自动平衡分配（2）方便地添加新的机器无需停机（3）无单点故障（4）自动故障转移。

MongoDB的分片背后的基本概念就是要打破涨成较小的块的集合。这些块可以是分布的碎片，使每个分片负责总的数据集的子集。要集合，MongoDB的分区指定碎片关键的图案命名一个或多个fileds的定义后，我们的关键数据分发。因为碎片键块可以被描述为一个三元收集，minkey的和maxkey。块增长的最大尺寸通常是200MB，一旦一大块已经达到近似的大小，块分裂成两个新的块。自动分片的体系结构如图1所示：



MongoDB的分片集群由两个或两个以上的分片，一个或多个配置服务器，以及任意数目的路由组成。每一个组成部分说明如下。

shards：一个shard为一组mongod，通常一组为两台，主从或互为主从，这一组mongod中的数据时相同的。数据分割按有序分割方式，每个分片上的数据为某一范围的数据块，故可支持指定分片的范围查询，这同google的BigTable 类似。数据块有指定的最大容量，一旦某个数据块的容量增长到最大容量时，这个数据块会切分成为两块；当分片的数据过多时，数据块将被迁移到系统的其他分片中。另外，新的分片加入时，数据块也会迁移。

config server：存储集群的信息，包括分片和块数据信息。主要存储块数据信息，每个config server上都有一份所有块数据信息的拷贝，以保证每台config server上的数据的一致性。

mongos：可以有多个，相当于一个控制中心，负责路由和协调操作，使得集群像一个整体的系统。mongos可以运行在任何一台服务器上，有些选择放在shards服务器上，也有放在client 服务器上的。mongos启动时需要从config servers上获取基本信息，然后接受client端的请求，路由到shards服务器上，然后整理返回的结果发回给client服务器。

C 自动分片的负载均衡

MongoDB使用平衡器来保持所有的块均匀地分布在集群服务器。传送单位是一大块，平衡器等待不均匀块计数的阈值发生。在该字段中，具有8组块之间的差异，至少装的碎片表明是一个很好的启发。一旦达到阈值时，平衡器将重新分配块，直到差异块beween的任意两个碎片下降到2个组块。

为了减少传输的数据量，每个碎片包含多个范围。当一个新的碎片加入集群或一些碎片含有太多的数据到达阈值时，均衡器将压缩块的顶部数据量最多的碎片，并将这些块的最低集合众多的碎片，使数据均匀地分布在集群移动最低限度下。

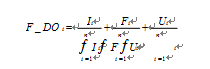
平衡器的目的不仅是为了均匀地保持数据分布而且也可以最大限度地减少传输的数据量。平衡器的算法是极端的智能。它的动作基于组块的整体尺寸的碎片[7]。迁移块还仅仅是位于顶部的每个碎片，但数据的操作还没有考虑到。碎片之间的数据传输可能不经常被使用，这将使得系统的负荷，不能达到有效的平衡。在MongoDB中提高自动分片的均衡策略是十分必要的。

三、基于数据操作频率的算法

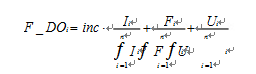
A FODO算法的基本思想

为了解决自动分配数据的分布不均的问题，提出了一种基于数据操作的频率的改进算法（FODO算法）

在MongoDB中，转让的单位是一大块。有n个大块碎片中，表示为第i块Ci和其频率运行数据值F\_DOi。在MongoDB数据库中，主营业务数据的插入，查找，更新和删除。由于集群中的数据很少会频繁删除，则该操作影响系统的性能主要集中前三操作。我们使用二，网络连接和Ui的数字代表这三种操作一大块。因此，在归一化的值F\_DOi形式为：



然而，插入，查找和更新操作上的集群中进行负载，并不应该被视为的产生不同的影响而一视同仁。在自动分片环境，这是没有必要的的物理连接时，从数据库中查询记录因为兑现已经保持新鲜的数据。此外，每个碎片将包括的副本集生产形势。次要节点副本集可以读取数据写入从节点，因此它可以在一定程度上减轻主节点的查询负载。反之，插入和更新操作每次当他们发生了操作直接导致数据库，这占据了整个的集群工作的更大的一部分负载。特别是，插入数据，将导致在每一个片的数据的数量是不同的。一旦的阈值是达到平衡器平衡数据自动分片机制，这将极大地消耗系统资源。所以FODO算法在插入在方程中的一部分之前添加参数INC（始终为> 1），被称为插入系数，把更多的权重赋予插入操作。这里是FODO的最后的值的定义：



每个分片的F\_DO值是所有它的副本集的F\_DO值得和：



我们需要修改记录块的数据结构信息，添加I，F，U和F\_DOi的四个变量来存储插入，查找，更新和F\_DOi值。每一个操作的数据必须记录在相应的四个变量中。

B FODO算法的负载均衡

F\_DOi值表示每个块的数据操作的频率。如果一大块F\_DOi值高，这意味着数据在经常使用这个块。每个碎片数据的阈值再分配仍然是块的数目的差异，但迁移块是根据它的选择F\_DOi值。有取舍之间的总体规模碎片和操作在此均衡策略的数据的频率。该数据平衡的过程中有三个步骤。

1. 数据迁移的门槛：计算数字在每个碎片块。如果该差值大于8，则开始，以平衡数据，直到差值小于2。从数据最多的碎片称为块将被迁移从碎片最少数据最多的碎片，叫做分片。
2. 选择迁移大块：计算差异的F\_DO值之间从碎片和碎片。如果F（F\_DO）> T（F\_DO），然后从碎片的F\_DOi值选择最大块。否则，从碎片块选择最小F\_DOi的值。
3. 迁移数据块：迁移在步骤2中所选择的组块到碎片，并重新计算每个块的的F\_DOi值在从碎片和到shard.Repeat的步骤1。

此外，我们应该确定业务的价值和权重。若业务是插入操作，所以把更多的重量，该值应该大于1。业务的具体价值依赖于服务器相关的参数和系统负载，我们可以使用在设定初始值，然后根据实际业务需求调整。

四、性能提升

MongoDB的自动测试环境的基础上分片群集，它包含10个虚拟机，其中每一个具有相同的硬件配置：8GB的内存和CPU速度是2.4G HZ。每个虚拟机都有一个Linux RedHat的操作系统，并通过100Mbps的连接局域网中。 MongoDB的版本是1.8.2，自动分片集群有三个碎片，每一个都包含一个副本集。每个副本集包含三个节点：一个主节点和两个次要节点。我们应该运行的rs.slaveOk（）为辅助节点的命令来查询数据。

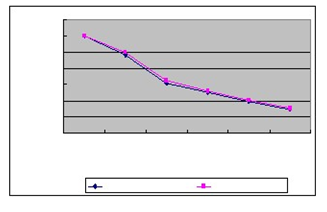
我们在MongoDB中实现FODO的算法，比较这两个算法通过测试并发读取和写集群的性能。测试中使用的数据集是一个简单的线数据，包括四个字段如int，long，字符串。此外，增量值在FODO参数算法是1.5。

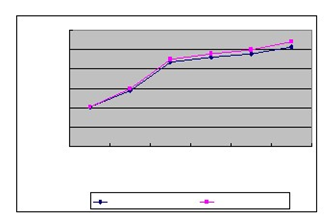
为了看到FODO算法的效果，我们只添加两个碎片，插入1000000条记录，并使用随机生成的ID做查找和更新操作。然后，我们添加第三个碎片到集群。此操作的目的是确保群集具有在测试开始的值的数据F\_DO不为零。

首先，我们测试集群的并发写入性能。 10000000条记录插入到集群中，并保持记录的总量不同的并发数字下是相同的。这两个并发写入性能图中所示的算法。

保持并发的数量和记录保持不变，测试并发读取性能的集群。该这两种算法的并发读取性能

图所示。





五、结论

本文分析 MongoDB的自动分片原理。对于数据碎片分布不均的问题，我们引入了一种改进的均衡算法算法基于数据操作频率（FODM）。基于FODO的算法，提出数据的均衡策略是并通过实验验证了其有效性。该并发自动写入和读取性能分片群集显着改善。在多个方面的的FODO算法值得探索和研究，如测定的INC价值。