摘 要

随着因特网应用和计算机技术的飞速发展，数据库逐渐成为信息系统的核心部分并广泛应用于企业、金融机构、政府及国防等各个领域。其中，分布式关系型数据库以其低成本、高可靠性等特点成为当前数据库理论与应用领域的研究热点。

通过对分布式理论、关系型数据库理论及相关技术的学习和研究，本文基于Java语言实现了一个分布式的关系型数据库JSQL。JSQL主要包含五大模块，网络模块接受前端应用程序的连接请求，对客服端进行认证和授权，并对连接进行管理，可实现基于Mysql的通信协议，这样便于Mysql的用户方便的迁移到本系统上来；Sql的解析和执行模块接受客户端的SQL请求，然后解析和执行数据库存储的调用，返回执行结果；审计模块是存储和分析所有对数据库的更改情况，以可视化的方式向用户展示；数据库引擎模块利用了orientdb开源的数据库引擎，实现可靠的分布式存储；分布式模块利用hazlcast实现了数据库集群。基于以上功能模块，ＪＳＱＬ具有高可用性，可扩展性，负载均衡等特性，同时从数据库底层考虑了数据库安全审计需求，加入了数据审计图形化界面显示审计结果。

论文对系统进行了功能和性能测试。功能测试结果表明，系统在功能上符合分布式数据库的基本要求，审计系统的功能也达到本论文的要求。论文通过对性能测试结果进行分析，认为系统的性能基本达到本论文的要求。但是本系统对复制SQL语句的支持还不是很完善，最后提出了改进的方案。

关键词：分布式数据库，mysql，安全审计，OLTP，NOSQL

ABSTRACT

Database, operating system and compiler and called the three systems, can be said that the cornerstone of the entire computer software. Which is closer to the application layer database, is a lot of business support. This field after decades of development, There are new developments. From the beginning of the hierarchy database and relational database, to the recent hot Nosql database, and then to the recent Google Spanner and F1 as the representative of the NewSql database.

In the Internet age, the storage and access of massive data becomes the bottleneck of system design and use. For mass data processing, King, divided into two types: online transaction processing (OLTP) and online analytical processing (OLAP).

Relational database is based on the relational model of the database, which by means of aggregation algebra and other mathematical concepts and methods to deal with the database data. Which is the most popular mysql, mysql is an open source relational database, the advantage lies in the open source code, any business and individuals can according to their own needs to modify the source code mysql.

NoSQL database, called Not Only SQL, meaning that when the relational database is used when the relational database, not applicable There is no need to use relational database is not necessary, you can consider the use of more appropriate data storage.

Oracle, mysql and other traditional relational database is very mature and has been large-scale commercial, why use NoSQL database? mainly With the development of the Internet, the amount of data is growing, the performance requirements are getting higher and higher, the traditional database of congenital defects, namely stand-alone (single Library) performance bottlenecks, and difficult to expand. This is a stand-alone single library bottleneck, but difficult to expand, naturally unable to meet the growing mass of data storage And its performance requirements, so there will be a variety of different NoSQL products.

Although in the cloud computing era, the traditional database there are congenital defects, but NoSQL database can not be replaced, NoSQL can only For the traditional data supplement can not be replaced, so to avoid the shortcomings of traditional databases is the current era of large data must be resolved.

In order to solve these problems, such as mysql and other relational database, this article describes how to design and implement a compatible mysql protocol distributed database. He can automatically find the distributed database cluster nodes, Automatically allocate data to support massive data storage. Taking into account the increasingly important security of the database, from time to time the occurrence of database administrators or other attackers malicious changes in the database data, So the database developed by the database from the bottom of the database to join the audit function.

The main work of this paper is as follows:

Based on the realization of java language compatible mysql communication protocol database;

The realization of the database cluster, to achieve the database of high availability, scalability, load balancing and other characteristics;

From the bottom of the database to consider the database security, joined the data audit and other features.

Keywords: mysql, java, mysql, java, mysql

目 录

第一章 绪论

1.1 研究背景和研究意义

互联网诞生以来在全球迅速蔓延。 2017年我国工业和信息化部最新发布的通信业经济运行情况显示， 2月末，我国移动电话用户总数达到13.3亿户，移动互联网用户总数达到11.2亿户，使用手机上网的用户数接近10.6亿户。互联网用户数量还有很大的增长空间，特别是亚洲人口众多的发展中国家。同时，智能手机的革命发展通过移动互联网大大提升用户体验，使移动互联网迅速发展。伴随着互联网的发展出现了各种基于互联网的应用服务，从传统媒体门户网站到BBS以及近年来社会媒体的兴起，电子商务的巨大发展，还有各种各样的移动互联网应用程序。

随着互联网和互联网应用的发展，数据存储的需求不断增长。 IDC报告显示，预计到2020年全球数据总量将超过40ZB，这一数据量是2011年的22倍。在过去几年，全球的数据量以每年百分之58的速度增长，在未来这个速度会更快。如果按照现在存储容量每年百分之40的增长速度计算，到2018年需要存储的数据量甚至会大于存储设备的总容量。

未来是“大数据”时代，这么大的存储需求，给数据存储技术的发展带来了很大的压力。有很多应用基于互联网提供的各种服务正在进入井喷时代的发展，而这些应用，背景的很大一部分面临同样的问题：怎么样以尽可能廉价的方式实现大规模数据存储和查询。如搜索服务，需要存储页面而分析，微博等社交网络需求的实时用户来表达查询的观点，在线旅游需要玩家的信息和操作来存储和查询，银行需要用户帐户信息和用户用于实时存储和查询。这种应用所面临的挑战总结为大数据可用性和可靠性要求高，部分应用需要实时查询响应和高并发性和数据一致性要求。在这样的环境下，分布式数据库近年来也取得了飞速发展。分布式数据库是一种数据库技术和网络技术结合的产品，相对于单节点数据库，分布式数据库容量和可用性具有很大的优势，因此能够更好的应对大规模和超大规模的数据存储需求的应用。分布式数据库系统通常使用大量廉价，独立的计算机系统构建，经济和性能上，可以满足互联网应用程序的对数据大小，可用性和性能的需求。

1.2 国内外研究历史与现状

数据库技术的发展始于20世纪60年代。在没有数据库的情况下，使用计算机存储数据的用户（主要是财务科研单位）以操作系统中的文件的形式存储数据。随着业务的发展，应用程序变得越来越复杂，人们开始需要开发管理数据在通用软件，这促成了数据库的诞生。 20世纪60年代以来，人们探索数据模型实现数据库，后来开发出三大数据库模型：层次数据模型，网格数据模型，关系数据模型，后来也出现在对象数据模型中。其中，可以使用关系数据模型严格的数学理论来描述数据库的组织和操作，具有简单灵活，数据库独立性高的特点特征。从20世纪70年代到90年代，关系数据库理论成熟并得到广泛应用，这个里程碑是1974年，IBM的圣荷西，加利福尼亚研究实验室的D.D. Chamberlin和Ray Boyce开发了SEQUEL结结构化查询语言，后来在1980年更名为SQL。 SQL是数据库中的标准数据查询语言，在1986年，由美国国家标准学会规范SQL，就这样成为了数据库系统的标准语言。 SQL包含三个部分：数据定义语言，数据操作语言，数据控制语言。 SQL是一种全面的通用关系数据语言，可以当它是一种高级的非程序语言，它允许用户在高级数据结构中工作。使用SQL 用户无需知道数据的具体存储方式。在SQL中一个简单的语句实现效果，使用其他编程语言需要很大一部分程序才能实现。另外，SQL通过使用这种语言允许用户掌握这种语言就可以使用这种语言来操作任何一个标准数据库产品。到20世纪90年代，关系数据库标准几乎适用于任何数据存储需求的应用程序。

分布式数据库系统是数据库系统技术与网络技术的结合。分布式数据库系统（DDBS）的研究始于20世纪70年代。但是，分布式数据库的理论与应用成为一个热门话题，是20世纪90年代的事情。 90年代，互联网网络出现爆炸式增长，同时各种应用程序对存储的需求与日俱增。在这样的环境下，分布式数据库系统的研究成为了热门话题，人们探索分布式数据库系统理论和关键技术，并快速应用理论实践，开发了各种商业应用价值的数据库产品。 2002年，Eric Brewer提出了引导分布式数据库研究的重要理论，并且后来证明是正确的，这个理论就是CAP定理，CAP定理的核心观点是：在分布式计算系统中，不可能同时满足以下三点：一致性，可用性，分区容忍性。 CAP理论认为分布式数据库产品的设计必须介于三者之间，其中至少有两个可以满足，不可能同时满足三个。根据CAP理论的指导，有学者认为，基于传统的关系数据库构建分布式数据库，很难满足当前大型数据存储需求的各类互联网应用，如搜索引擎，社交网络等。由于传统的关系数据库非常重视数据一致性，在传统的关系数据库中，交易的四点必须保证要求：ACID（Atomicity，一致性，隔离，持久性）。有部分人认为，根据CAP理论，在

满足强大的一致性之后，也希望获得互联网应用的高可用性性是非常困难的。因此，在二十世纪九十年代末和二十一世纪初，互联网应用大幅增长出现了一些人认为是革命性的分布式数据库概念：NoSQL（not only SQL）。 NoSQL和传统的关系数据库非常的不同，对于一个概念，NoSQL的显着特点是：非关系型，分布式，不提供ACID数据库设计模式。 NoSQL不支持复杂的关系操作，不提供对交易一致性的支持。由于摆脱了必须满足支持一致性的功能要求，根据NoSQL概念设计产品生成的高可扩展性高并发高可用性性支持得到了很大的改善，各种类似的开源或商业的NoSQL产品出现了，并且被大量和被各种互联网应用程序所使用。这些NoSQL产品被应用到新的产品，如博客，论坛，微博等其他社交服务。这些应用程序具有高度可扩展性，高可用性，并发性。高可用性这些功能很重要，因为这些应用压力主要来自大规模数据存储，大量并发访问及时响应， NoSQL拒绝一致性支持是合理的，因为像社交网络应用程序一样用户丢失数据库的后果不会太严重，毕竟，这不会导致用户遭受巨大的损失，如经济利益。作为分布式数据库，NoSQL在过去两年爆发式增长，均受益于互联网高速发展，也是各种应用到“大数据”的发展趋势，CAP理论再一次证明了它的正确性。

毕竟，NoSQL放弃了一致性的支持，这些产品可以应用到请求的一致性不高在应用上，随着互联网的快速发展和“大数据”时代的到来，那些需要关系操作，需要高级一致性并且面临大规模数据存储查询要求的应用，比如电子商务，这样的应用程序不能使用NoSQL产品。所以，目前的分布式数据库开发方向，根据CAP理论，一个是尽可能的减少一致性，提高可用性，另一个方向是需要尝试确保操作之间的关系，一致性的支持，同时利用分布式技术的优势，提供尽量高性能的服务。

1.3 论文的主要工作

通过对分布式理论、关系型数据库理论及相关技术的学习和研究，论文基于Java语言实现了一个分布式的关系型数据库JSQL ，jsql是一个兼容mysqk通信协议的分布式数据库。论文的主要工作包括：

1.利用orientdb开源项目设计和实现了分布式数据库的本地存储。

2.实现了mysql通信协议，使得可以通过mysql客服端来连接jsql分布式数据库，方便mysql用户迁移到本数据库系统。

3.实现了对sql语句的解析和执行。

4.实现了数据库的分布式架构，使得数据可以存储在多台计算机上面。

5.实现了系统的审计系统，使得系统的安全性得到增强。

1.4 本论文的结构安排

第一章作为本论文的绪论，首先介绍了论文的选题背景和本论文进行研究的意义。接着阐述了数据库的发展历史和现状。最后一节里介绍了论文的主要工作。

第二章，主要介绍了本论文相关的理论基础和相关的技术，首先给数据库分类，给出数据库相关的概念，然后对分布式数据库相关的技术进行了介绍，主要包括负载均衡技术，数据分片技术以及数据库高可用技术。本章最后给出了mysql的体系结构，作为本系统的参考架构。

第三章，作为论文的需求分析，本章给出了分布式数据库jsql的实现目标，同时也包括对通信协议的实现和编程模式的分析。

第四章，是系统的设计，包括总体设计和模块划分，本章对系统的总体架构和各个模块的详细架构给出了阐述。系统主要包括数据库模块，集群架构，已经数据审计功能模块。

第五章，是关于系统的具体实现，本章分别从数据库模块，集群架构模块，数据库审计模块详细说明了每个模块的实现。接着就每个关键模块的实现进行了讲解。最后对系统主要功能的流程进行讲解。

第六章里，论文对jsq分布式数据库进行测试，包括功能测试和性能测试。

最后部分，是关于致谢和参考资料。

1.5 本章小结

本章首先阐述了论文的背景，并表明本论文的研究对于该领域的发展具有非常重要的意义。然后介绍本课题的国内外研究历史和发展现状。最后介绍了本文的主要工作和章节部分安排。

第二章 理论基础和相关的技术

本章介绍分布式数据库方面的相关理论和技术，还有mysql协议的研究和学习。

2.1 数据库的分类和架构

2.1.1 键值存储数据库

键值数据库就类似传统语言中使用的哈希表。可以通过key来添加、查询或者删除数据库，因为使用key主键访问，所以会获得很高的性能及扩展性。

键值数据库主要使用一个哈希表，这个表中有一个特定的键和一个指针指向特定的数据。Key/value模型对于IT系统来说的优势在于简单、易部署、高并发。

典型产品：Memcached、Redis、MemcacheDB。

2.1.2 列存储数据库

列存储数据库将数据存储在列族中，一个列族存储经常被一起查询的相关数据，比如人类，我们经常会查询某个人的姓名和年龄，而不是薪资。这种情况下姓名和年龄会被放到一个列族中，薪资会被放到另一个列族中。

这种数据库通常用来应对分布式存储海量数据。

典型产品：Cassandra、HBase。

2.1.3 面向文档数据库

文档型数据库的灵感是来自于Lotus Notes办公软件，而且它同第一种键值数据库类似。该类型的数据模型是版本化的文档，半结构化的文档以特定的格式存储，比如JSON。文档型数据库可以看作是键值数据库的升级版，允许之间嵌套键值。而且文档型数据库比键值数据库的查询效率更高。

面向文档数据库会将数据以文档形式存储。每个文档都是自包含的数据单元，是一系列数据项的集合。每个数据项都有一个名词与对应值，值既可以是简单的数据类型，如字符串、数字和日期等；也可以是复杂的类型，如有序列表和关联对象。数据存储的最小单位是文档，同一个表中存储的文档属性可以是不同的，数据可以使用X ML、JSON或JSONB等多种形式存储。

典型产品：MongoDB、CouchDB。

2.1.4 图形数据库

图形数据库允许我们将数据以图的方式存储。实体会被作为顶点，而实体之间的关系则会被作为边。比如我们有三个实体，Steve Jobs、Apple和Next，则会有两个“F ounded by”的边将Apple和Next连接到Steve Jobs。

典型产品：Neo4J、InforGri。

2.1.5 关系型数据库

虽然网状数据库和层次数据库已经很好的解决了数据的集中和共享问题，但是在数据库独立性和抽象级别上仍有很大欠缺。用户在对这两种数据库进行存取时，仍然需要明确数据的存储结构，指出存取路径。而关系型数据库就可以较好的解决这些问题。

关系型数据库模型是把复杂的数据结构归结为简单的二元关系。在关系型数据库中，对数据的操作几乎全部建立在一个或多个关系表格上，通过对这些关联的表格分类、合并、连接或选取等运算来实现数据库的管理。

关系型数据库诞生40多年了，从理论产生发展到现实产品，例如：Oracle和MySQL，O racle在数据库领域上升到霸主地位，形成每年高达数百亿美元的庞大产业市场。

2.1.6 Mysql数据库体系结构

因为 MySQL 采用的是客户机/服务器体系结构，所以在使用 MySQL 进行存取数据操作时，必须至少使用两个或者是两类程序：一个是位于存放数据的主机上的程序——数据库服务器。数据库服务器在网络上监听来自客户机的请求，然后根据客户机的这些请求访问数据库数据，访问之后再向客户机提供它们想得到的信息; 连接到数据库服务器的程序——客户机，这些程序是作为用户和服务器之间交互信息的工具，并且告诉服务器需要查询信息的内容。

MySQL 的架构可以描述为层次性子系统组合。MySQL 的源代码不是按照单组件或者模块的方式编写的，但是各个层次的源代码还是能够被分离出来，大部分的子系统依赖于一些通用的底层库。MySQL 包含以下子系统：网络连接和网络通信协议子系统；线程、进程和内存分配子系统；查询解析和查询优化子系统；存储引擎接口子系统；各类存储引擎子系统；安全管理子系统；日志子系统；mysql核心库文件等。

当一个客户端通过网络连接 MySQL 数据库服务时，网络连接子系统执行一系列的与网络协议有关的底层任务。然后网络连接子系统将控制权交给线程子系

统，线程子系统提供一个线程来处理这个连接，这个连接称之为连接线程。随后连接线程得到控制权，它首先调用安全管理子系统来验证用户访问的合法性。连接线程将获得的数据传给控制系统，其中一些请求在内核代码中被称作命令。这些命令中的一部分可以由这个控制系统直接完成，对于不可以直接由系统分发来完成查询的，分发系统将调用解析子系统对 SQL 语句进行解析。同时，如果在配置 MySQL 系统时采用了日志功能，那么分发系统还会调用日志系统去记录此次的信息。随后解析子系统将解析结果传给调用优化子系统以优化 SQL 语句。接着进行表操作，并将一系列请求发往存储引擎接口子系统。存储引擎接口子系统将上述调用自动转化为某个具体的存储子系统方法。上述过程完成后，相应的模块将 SQL 执行结果发往客户端，最后再由服务器将控制权交给连接线程，连接线程完成某些清理工作，并在此等待客户端的连接或者其他查询，直到客户端输入 Quit 命令为止，到此本次通话才会结束。

2.2 关系数据模型

分布式关系数据库提供基于关系数据模型的数据存储。关系数据模型是三大数据模型里研究最为成熟，应用最为广泛的数据模型，其他两个分别是层次数据模型和网状数据模型。数据模型是现实世界数据特征的抽象，用于描述一组数据的概念和定义。数据库的类型是根据数据模型来划分的，每一个数据库产品都是针对某种数据模型而设计的。基于关系数据模型的数据库，其数据结构是关系结构，数据操作是建立在严格的数学理论之上的，这种数学理论是关系代数，经过多年的发展，SQL 已经成为了表述关系操作的标准语言。

2.3 分布式关系数据库

2.3.1 分布式存储

分布式存储，相对于集中式存储，通常而言，是将数据存储在一个由网络连通的多个结点构成的集群之上，向上层提供存储接口的一种存储系统。随着互联网的普及以及各类在线应用的大量出现，许多互联网公司都面对海量数据存储的压力。而分布式存储技术能提供对大规模数据的存储和访问，并且支持高可扩展性，高并发，高性能。另外，很重要的一点，分布式技术能以非常廉价的成本提供这样的服务，很多公司在起步阶段，着眼于业务的发展，而向数据存储厂商购买数据存储服务，这些大型的高端服务器及其配套的软件和后续维护花费巨大，在公司发展壮大之后多会转而自行部署分布式存储系统来解决数据存储的问题。

2.3.2 分布式数据库的介绍

分布式数据库是用计算机网络将物理上分散的多个数据库单元连接起来组成的一个逻辑上统一的数据库。每个被连接起来的数据库单元称为站点或节点。分布式数据库有一个统一的数据库管理系统来进行管理，称为分布式数据库管理系统。

分布式数据库的基本特点包括：物理分布性、逻辑整体性和站点自治性。从这三个基本特点还可以导出的其它特点有：数据分布透明性、按既定协议达成共识的机制、适当的数据冗余度和事务管理的分布性。

分布式数据库按照各站点中数据库管理系统的数据模型的异同分为异构型分布式数据库和同构型分布式数据库，按照控制系统的类型分为全局控制集中性、全局控制分散型和全局控制可变型。

优点：

1.随时能针对各区域的用户做调整

2.数据共用和分布式控制

3.增加处理绩效，可作平行处理

4.系统管理费用较低

5.质量维持容易

缺点：

1.重复存储数据很花时间

2.数据处理与管理上具复杂度

3.数据的保密性与安全性受到威胁

2.4 分布式数据库相关技术

2.4.1 负载均衡技术

面对大量用户访问、高并发请求，海量数据，可以使用高性能的服务器、大型数据库，存储设备，高性能Web服务器，采用高效率的编程语言，当单机容量达到极限时，我们需要考虑业务拆分和分布式部署，来解决大型网站访问量大，并发量高，海量数据的问题。

从单机网站到分布式网站，很重要的区别是业务拆分和分布式部署，将应用拆分后，部署到不同的机器上，实现大规模分布式系统。分布式和业务拆分解决了，从集中到分布的问题，但是每个部署的独立业务还存在单点的问题和访问统一入口问题，为解决单点故障，我们可以采取冗余的方式。将相同的应用部署到

多台机器上。解决访问统一入口问题，我们可以在集群前面增加负载均衡设备，实现流量分发。

负载均衡，意思是将负载进行平衡、分摊到多个操作单元上进行执行。是解决高性能，单点故障,高可用，扩展性的终极解决方案。

系统的扩展可分为纵向垂直扩展和横向水平扩展。纵向扩展，是从单机的角度通过增加硬件处理能力，比如CPU处理能力，内存容量，磁盘等方面，实现服务器处理能力的提升，不能满足大型分布式系统大流量，高并发，海量数据的问题。因此需要采用横向扩展的方式，通过添加机器来满足大型网站服务的处理能力。比如：一台机器不能满足，则增加两台或者多台机器，共同承担访问压力。

负载均衡的作用：

1.解决并发压力，提高应用处理性能增加吞吐量，加强网络处理能力

2.提供故障转移，实现高可用

3.通过添加或减少服务器数量，提供网站伸缩性1扩展性

4.安全防护，负载均衡设备上做一些过滤，黑白名单等处理

根据实现技术不同，可分为DNS负载均衡，HTTP负载均衡，IP负载均衡，链路层负载均衡等。

2.4.1.1 DNS负载均衡

DNS负载均衡是最早的负载均衡技术，利用域名解析实现负载均衡，在DNS服务器，配置多个A记录，这些A记录对应的服务器构成集群。大型网站总是部分使用DNS解析，作为第一级负载均衡。

优点：

1.使用简单：负载均衡工作，交给DNS服务器处理，省掉了负载均衡服务器维护的麻烦

2.提高性能：可以支持基于地址的域名解析，解析成距离用户最近的服务器地址，可以加快访问速度，改善性能；

缺点

1.可用性差：DNS解析是多级解析，新增修改DNS后，解析时间较长；解析过程中，用户访问网站将失败；

2.扩展性低：DNS负载均衡的控制权在域名商那里，无法对其做更多的改善和扩展；

3.维护性差：也不能反映服务器的当前运行状态；支持的算法少；不能区分服务器的差异,不能根据系统与服务的状态来判断负载.

将DNS作为第一级负载均衡，A记录对应着内部负载均衡的IP地址，通过内部负载均衡将请求分发到真实的Web服务器上。一般用于互联网公司，复杂的业务系统不合适使用。

2.4.1.2 IP负载均衡

IP负载均衡是在网络层通过修改请求目标地址进行负载均衡。用户请求数据包，到达负载均衡服务器后，负载均衡服务器在操作系统内核进程获取网络数据包，根据负载均衡算法得到一台真实服务器地址，然后将请求目的地址修改为，获得的真实ip地址，不需要经过用户进程处理。真实服务器处理完成后，响应数据包回到负载均衡服务器，负载均衡服务器，再将数据包源地址修改为自身的ip地址，发送给用户浏览器。 IP负载均衡，真实物理服务器返回给负载均衡服务器，存在两种方式：负载均衡服务器在修改目的ip地址的同时修改源地址。将数据包源地址设为自身盘，即源地址转换1snat。将负载均衡服务器同时作为真实物理服务器集群的网关服务器。

优点：在内核进程完成数据分发，比在应用层分发性能更好。缺点：所有请求响应都需要经过负载均衡服务器，集群最大吞吐量受限于负载均衡服务器网卡带宽。

2.4.1.3 链路层负载均衡

链路层负载均衡是在通信协议的数据链路层修改mac地址，进行负载均衡。数据分发时，不修改ip地址，指修改目标mac地址，配置真实物理服务器集群所有机器虚拟ip和负载均衡服务器ip地址一致，达到不修改数据包的源地址和目标地址，进行数据分发的目的。实际处理服务器ip和数据请求目的ip一致，不需要经过负载均衡服务器进行地址转换，可将响应数据包直接返回给用户浏览器，避免负载均衡服务器网卡带宽成为瓶颈。也称为直接路由模式1DR模式.

优点：

1.性能好

2.配置复杂

由于多个服务器群内硬件设备、各自的规模、提供的服务等的差异，可以考虑给每个服务器群采用最合适的负载均衡方式，然后又在这多个服务器群间再一次负载均衡或群集起来以一个整体向外界提供服务即把这多个服务器群当做一个新的服务器群，从而达到最佳的性能。将这种方式称之为混合型负载均衡。此种方式有时也用于单台均衡设备的性能不能满足大量连接请求的情况下。是目前大型互联网公司，普遍使用的方式。

2.4.2 数据分片技术

在分布式存储系统中，数据需要分散存储在多台设备上，数据分片就是用来确定数据在多台存储设备上分布的技术。数据分片要达到三个目的：

1.分布均匀，即每台设备上的数据量要尽可能相近；

2.负载均衡，即每台设备上的请求量要尽可能相近；

3.扩缩容时产生的数据迁移尽可能少。

数据分片一般都是使用Key或Key的哈希值来计算Key的分布，常见的几种数据分片的方法如下：

1.划分号段。这种一般适用于Key为整型的情况，每台设备上存放相同大小的号段区间，如把Key为[1, 10000]的数据放在第一台设备上，把Key为[10001, 20000]的数据放在第二台设备上，依次类推。这种方法实现很简单，扩容也比较方便，成倍增加设备即可，如原来有N台设备，再新增N台设备来扩容，把每台老设备上一半的数据迁移到新设备上，原来号段为[1, 10000]的设备，扩容后只保留号段[1, 5000]的数据，把号段为[5001, 10000]的数据迁移到一台新增的设备上。此方法的缺点是数据可能分布不均匀，如小号段数据量可能比大号段的数据量要大，同样的各个号段的热度也可能不一样，导致各个设备的负载不均衡；并且扩容也不够灵活，只能成倍地增加设备。

2.取模。这种方法先计算Key的哈希值，再对设备数量取模1整型的Key也可直接用Key 取模1，假设有N台设备，编号为0 N-1，通过Hash1Key1N就可以确定数据所在的设备编号。这种方法实现也非常简单，数据分布和负载也会比较均匀，可以新增任何数量的设备来扩容。主要的问题是扩容的时候，会产生大量的数据迁移，比如从N台设备扩容到N+1台，绝大部分的数据都要在设备间进行迁移。检索表。在检索表中存储Key和设备的映射关系，通过查找检索表就可以确定数据分布，这里的检索表也可以比较灵活，可以对每个Key都存储映射关系，也可结合号段划分等方法来减小检索表的容量。这样可以做到数据均匀分布、负载均衡和扩缩容数据迁移量少。缺点是需要存储检索表的空间可能比较大，并且为了保证扩缩容引起的数据迁移量比较少，确定映射关系的算法也比较复杂。

3.一致性哈希。一致性哈希算法在1997年由麻省理工学院提出的一种分布式哈希实现算法，设计目标是为了解决因特网中的热点问题，一致性哈希的算法简单而巧妙，很容易做到数据均分布，其单调性也保证了扩缩容的数据迁移是比较少的。通过上面的对比，在这个系统选择一致性哈希的方法来进行数据分片。

2.4.3 数据库高可用技术

高可用性指的是通过尽量缩短因日常维护操作和突发的系统崩溃所导致的停机时间，以提高系统和应用的可用性。HA系统是目前企业防止核心计算机系统因故障停机的最有效手段。

随着IT信息系统的不断发展，数据在企业的应用越来越广，如何提高IT系统的高可用性成为建设稳健的计算机系统的首要任务之一。构成计算机网络系统的三大要素是：网络系统，服务器系统，存储系统。网络系统包括防火墙，路由器等网络设备，服务器系统主要指用户使用的各种服务器系统，存储系统，则是用户最主要的数据存储放的地点。因此IT系统的高可用建设应包括网络设备高可用性，服务器设备高可用性，及存储设备的高可用性三个方面:

1.网络高可用,由于网络存储的快速发展，网络冗余技术被不断提升，提高IT系统的高可用性的关键应用就是网络高可用性，网络高可用性与网络高可靠性是有区别的，网络高可用性是通过匹配冗余的网络设备实现网络设备的冗余，达到高可用的目的。比如冗余的交换机，冗余的路由器等

2.服务器高可用, 服务器高可用主要使用的是服务器集群软件或高可用软件来实现。

3.存储高可用, 使用软件或硬件技术实现存储的高度可用性。其主要技术指标是存储切换功能，数据复制功能，数据快照功能等。当一台存储出现故障时，另一台备用的存储可以快速切换，达一存储不停机的目的。

2.5 网络相关技术

2.5.1 TCP/IP协议

TCPIP协议全称Transmission Control ProtocolInternet Protocol，中译名为传输控制协议因特网互联协议，又名网络通讯协议，是Internet最基本的协议、 Internet国际互联网络的基础。

TCPIP是一个协议集合，寻址和路由选择以及传输控制是它的核心功能。HTTP协议的基础是TCPIP协议，HTTP实现了服务器与客户端间的请求与响应， TCPIP则实现了两端底层的数据传输‘36。37】。

网络层：网络层的协议包括IP协议、IGMP1因特网组管理协议1、ARP1地址解析协议1、ICMP1因特网控制报文协议1、RARP1反向地址解析协议1。其中IP协议向其它高层协议提供了基本的数据传输的功能，是无连接的、不可靠数据报协议。传输层：TCP和UDP协议都使用IP网络层协议，是传输层的两个著名的协

议。为客户端和服务器上的应用程序提供端到端的通信。虽然TCP使用了不可靠的IP网络层协议，但它却可以为主机间提供可靠的、面向连接的传输层数据通信服务。与TCP不同的是，UDP为主机间提供无连接的、不可靠的传输层数据通信。由于UDP协议开销很小，因此在特定领域也有着广泛的应用应用层：用于处理应用程序的细节，这一层有许多协议，如Http、FTP、TELNET 世 i手。

连接建立、数据传送和连接终止是TCP连接的三个状态。TCP使用三次握手1three．way handshake1协议来建立连接，终止一个连接要经过4次握手。在TCP建立连接的过程中，将会初始化很多参数，例如对报文序号的初始化来保证连接的强壮性和按序号传输。

111建立连接：在有些情况下会出现一对终端同时初始化一个他们之间的链接的情况，但通常是由服务器端被动打开一个套接字1socket1监听来自客户端的连接。客户端发送一个SYN报文段指明需要连接的服务器的端口，以及初始顺序号来建立一个主动打开，作为三路握手的一部分。服务器发回包含服务器的初始顺序号的SYN报文段1SY’N为11作为应答。同时，将确认号设置为客户的ISN加1以对客户的SYN报文段进行确认1ACK也为11。服务器端接收到来自客户端的SYN后一般会验证客户端的合法性，为每个合法的客户端发送一个SYN／ACK。最后，客户端接收到服务器的响应后再发送一个ACK。这样三路握手就完成了进入链接建立的状态。

121数据传输：为保证在TCP的数据传送状态下的可靠性和强壮性，主要采取了一下机制，它们包括：对收到的TCP报文采用序号进行排序来检测重复数据；采用校验和对报文段的错误进行检测；利用计时器对丢包或延时进行检测和纠正。在的连接已经建立的状态下，两个主机的TCP层互相交换初始序号1ISN1。初始序号除了能够对字节流中的数据进行标识以外，还可以用来对应用层的数据字节进行记数。通常情况下

，序号和确认号存在于每个TCP报文段中。报文的发送者将自己的字节编号称为序号，将接收者的字节编号称为确认号。为了保证报文的可靠性，接收者在接收到一定数量的连续字节后才发送确认，这种被称为选择确认的机制1SACK1是TCP的一种扩展。当数据以乱序到达接收者端时，如果没有选择确认的机制，报文中的字节就不能按正确的顺序传递给应用层。 131链接终止：TCP用三个分节建立一个连接，终止一个连接则需要四个分节，在标准的拆接过程中，链接的每一端都要提供一个FIN和ACK对。

2.5.2 java流IO技术

在Java的I／O类库中，“流”这一抽象的概念，通常用来表示接收数据的接收端对象或是产生数据的数据源对象。“流”有效地封装了I／O中的实现细节，是 9 NIO高性能框架的研究与应用 Java平台I／O的基础。

就流的运动方向而言，流可以分为输入流1InputStream1和输出流 1OutputStream1，输入流代表从外设流入计算机的数据序列，而输出流则代表从计算机流向外设的数据序列。在Java平台中，所有继承于InputStream的与输入有关的类和所有继承于OutputStream与输出相关联的类，分别称为Java的I／O类库中的输入和输出，是组成JavaI／O类库的两个主要组成部分。从不同数据源产生输入的类用InputStream来表示。这些数据源包括： 111字节数组； 121String对象； 131文件； 141“管道”，工作模式类似于实际的管道，从一端输入，从另一端输出； 151序列； 161其他数据源，例如网络连接等。 abstract int read11是InputStream中的一个最主要的抽象方法方法该方法会返回从输入源中读到的一个字节，当返回结果为一1时，表示已经读到了输入源的末尾。设计具体的输入流类的时候，设计者需覆盖这个方法来提供详细的功能实现。 OutputStream决定数据输出的各种不同的目的地，这些目的地包括字节数组、文件或管道等。与InputStream类似，OutputStream也提供了一个抽象方法abstract int write1int b1交由设计者去覆盖实现，这个方法从要输出的数据中的一个字节写到指定的目的地。

Javal.1版本对基本的I／O流类库进行了极大程度的修改。一开始看到 Reader1阅读器1和Writer1书写器1时，可能会误认为它们是用来替代InputStrearn 和OutputStream，但实际上不是这样的。InputStream和OutputStream仍然非常有价值，特别是在面向字节形式的I／O中，Reader和Writer则为Unicode和面向字符的I／O功能提供兼容。当需要把面

向字节形式的I／O中的类和向字符形式的I／O 中的类结合起来使用时，“适配器”1adapter1类可以起到中间桥梁的作用。在“适配器”类中InputStreamReader能够将InputStream转换成Reader，而 OutputStreamWriter能够将OutputStream转换成Writer。Reader和Writer继承层次结构的设计是为了满足国际化的需求。

Unicode是国际组织制定的可以容纳世界上所有文字和符号的字符编码方案，由于在I／O流的继承层次结构中只能够支持8位的字节流，当遇到16位的Unicode字符时处理起来很不方便。所以Javal.1 版本添加了Reader和Writer的继承层次结构为了来支持Unicode编码。

流I／O技术数据的输入输出时面向字节的，输入流每次一个一个字节地从数据源读取数据，输出流也是每次向输出流一个一个字节的写入数据。相对于面向

块的数据读取而言，通过这种方式进行的数据读写速度很慢。同时，流I／O采用的是阻塞的调用方式。在真正的写入或读取操作完成以前，工作线程都会被阻塞。这意味着在复杂的网络环境中，由于网络连接繁忙而导致数据不能被流立即读取或写入时，Java就会挂起这个工作线程，一直使其陷入等待的状态，直到流再次可用为止。

2.5.3 Socket编程技术

Socket套接字是网络上服务器端与客户机端之间进行双向通信的一方，可以发送或接受连接请求，Socket将通信双方一端写入的信息发送至另外一端的Socket中，利用Socket套接字可以方便地进行数据的传输。类似于文件传输的输入输出流原理，Socket套接字通信可以通过读写数据实现对网络资源的使用。Java中提供了两种套接字方式，分别为流式Socket和数据报式Socket，二者的比较如表2．1：表2．1流式与数据报式Socket类型比较流式Socket通信服务的特点有：面向连接、可靠性高、有序、无差错、无重复、可移植性好，支持TCP协议，可实现不同类型计算机之间的通信。

在客户机／服务器模式中，可以把Socket实例看作一个特殊的实例对象，用于描述IP地址和端12，是一个通信链的句柄。客户机端流套接字为Socket对象，服务器端流套接字为ServerSocket对象，这两个对象中都封装了两个方法，分别是输入流getlnputStream0和输出流getOutputStream0，是通信过程中实现Socket 通信连接的关键方法。Socket实现网络通信的流程如图2．2所示：NIO高性能框架的研究与应用客户端调用9Dck烈O创建一个会话调用oonnectO与咖端连接调用recvO／send0进行会话 aiseS：x：ketO关闭套接字服务器调用9xket仓]建监听调用bjndO为监听 9:，cket选择通信对象调用list 调用ao：eptO接收连接同时生成会话

三次握手，建立连接请求连接数据应答调用recv11／sendO进行会话 aosegx：ketO关闭套接字图2．2 Socket通1言机制根据图2．3所示原理，PC机客户监控端与服务器端进行通信传输时，首先服务器端创建Socket，调用bind11为监听Socket选择通信对象，同时调用listen11 和accept11等待客户端请求连接，客户端创建Socket并调用connect11尝试与服务器端连接，然后开始调用read11和write11与服务器端数据进行数据传输，连接建立完成。

服务器端程序编写：Java中使用专门建立Sock et服务器的类ServerSocket来创建服务器对象，采用端口号作为传递参数，且端口号是为了唯一标识每台PC机的唯一服务的41。在服务器端建立Socke t通信连接的步骤如下： 111首先在服务器端创建Socket并绑定到端口上； ServerSocket server=new ServerSocket1int port1；

121不停监听客户机的连接请求，一旦监听到客户机连接请求后，服务器调用accept11函数并接受连接请求，完成S ocket通信连接的建立； Socket s=SS．accept11； 131服务器端调用Socket类的输入输出函数以获取输入输出流，并利用输入输出流进行数据的传输。 OutputStream OS=S．getOutputStream11； ／／创建输出流 InputStream is=s．getlnputStreamO； ／／仓JJ建输入流 OS．write1”Hello，welcome you!”．getBytes111； byte[】aa=new byte[100]； ／／建立字节数组 12 工程硕士学位论文 int len=is．read1a1； ／／读取数据到数组中并返回实际读取的字节数 System．out．print1new String1a，0，1en11； 141关闭通信套接字。客户机端和服务器端的类似： 111首先在客户机端创建Socket流套接字，并连接到服务器端； Socket s=new Socket1InetAddress．getByName1null1，port1； 121客户机端调用Socket对象的输入输出函数以获取输入输出流，并利用输入输出流进行数据的传输。 OutputStream OS=s．getOutputStreamO； InputStream is 2s．getInputStreamO； byte[】a=new byte[100]；int len=is．read1a1； ／／从服务器端读取数据 System．out．print1new String1a，0，len11； OS．write1”Hello，this is mengmeng”．getBytes01； 131关闭通信套接字。

2.5.4 NIO技术

在JDKI．4提出的新特性中，NIO技术是最大的亮点。与传统的流I／O技术相比，NIO主要带来了两点改进：1．弥补了原来的I／O的不足，提供了面向块的、非阻塞的I／O处理能力。2．使得Java应用程序能够更加充分地发挥操作系统的性能，进行高性能的I／O操作。通道、缓冲区和选择器是NIO中三个最为关键的内容

缓冲区是一个数据容器，缓冲区对象由一个用来存储数据的数组和一系列控制数据读写的属性组成。在NIO技术中，所有数据的传输都是通过缓冲区来完成的，这体现了与流I／O技术的一点主要的不同。在流I／O技术中，数据时直接读写至1]Stream对象中的，而在NIO技术中，所有数据都是从缓冲区来读出和写入。所有的缓冲区都具有四个属性来提供关于其所包含的数据元素的信息，它们分别是： 111容量1Capacity1：缓冲区能够容纳的数据元素的最大数量。这一容量在缓冲区创建时被设定，并且永远不能被改变。 121上界1Limit1：缓冲区的第一个不能被读或写的元素。或者说，缓冲区中现存元素的计数。 131位置1Position1：下一个要被读或写的元素的索引。位置会自动由相应的get11和put11方法更新。141标记1Mark1：一个备忘位置。调用mark11来设定mark=postion。调用reset11设定position=mark。标记在设定前是未定义的1undefined1。这四个属性之间总是遵

循以下关系：0¡=mark¡=position¡=limit¡= capacity。缓冲区结构图如图3．1所示：17 通道1Channel1是java．nio的第二个主要创新。它们既不是一个扩展也不是一项增强，而是全新、极好的Java I／O示例，提供与I／O服务的直接连接。Channel 用于在字节缓冲区和位于通道另一侧的实体1通常是一个文件或套接字1之间有效地传输数据。通道与流相比不同之处在于通道是双向的，由于它是双向的，所以通道比流能够更好地反映底层操作系统的真实情况。

选择器维护着注册过的通道的集合，并且这些注册关系中的任意一个都是封装在SelectionKey对象中的。选择键封装了特定的通道与特定的选择器的注册关系。选择键对象被SelectableChannel．register11返回并提供一个表示这种注册关系的标记。选择键包含了两个比特集1以整数的形式进行编码1，指示了该注册关系所关心的通道操作，以及通道已经准备好的操作。每一个Selector对象维护三个键的集合： 111已注册的键的集合1Registered key set1：与选择器关联的已经注册的键的集合。 121已选择的键的集合1Selected key set1：已注册的键的集合的子集。这个集合的每个成员都是相关的通道被选择器1在前一个选择操作中1判断为已经准备好的，并且包含于键的interest集合中的操作。 131已取消的键的集合1Cancelled key set1：已注册的键的集合的子集，这个集合包含了cancel11方法被调用过的键1这个键已经被无效化1，但它们还没有被注销。选择器提供选择执行已经就绪的任务的能力，这使得多元I／O成为可能。就绪选择和多元执行使得单线程能够有效率地同时管理多个I／O通道1channels1。 C／C++代码的工具箱中，许多年前就已经有select11和poll11这两个POSIX1可移植性操作系统接口1系统调用可供使用了。许过操作系统也提供相似的功能，但对Java程序员来说，就绪选择功能直到JDK 1．4才成为可行的方案。对于主要的 18 工程硕士学位论文工作经验都是基于Java环境的开发的程序员来说，之前可能还没有碰到过这种 I／O模型。

数据打包和传输的方式是流I／O技术与NIO技术的一个重要的区别。流I／O技术是面向字节的，而NIO技术通过块的方式来进行数据读写。在流I／O技术中，数据的读写以一次一个字节地进行，这种处理方式I／O字节流必须顺序读取，处理速度比较慢，经常用于间歇性输入。而在向块的I／O系统中，每一次操作都消费一个数据块，比一个一个字节的读写数据要快得多。 NIO对非阻塞I／O操作的支持是其另外一个重要的特征。在NIO包中的多路复用对象1Selector1提供选择执行已经就绪的任务的能力，这使得非阻塞I／O成为可能。就绪选择使得单线程能够有效率地同时管理多个I／O通道1channels1。 C／C++代码的工具箱中，许多年前就已经有select11和poll11这两个POSIX1可移植性操作系统接口1系统调用可供

使用了。许过操作系统也提供相似的功能，但对Java程序员来说，就绪选择功能直到JDK 1．4才成为可行的方案。

2.5.5 线程池技术

线程池（英语：thread pool）：一种线程使用模式。线程过多会带来调度开销，进而影响缓存局部性和整体性能。而线程池维护着多个线程，等待着监督管理者分配可并发执行的任务。这避免了在处理短时间任务时创建与销毁线程的代价。线程池不仅能够保证内核的充分利用，还能防止过分调度。可用线程数量应该取决于可用的并发处理器、处理器内核、内存、网络sockets等的数量。例如，线程数一般取cpu数量+2比较合适，线程数过多会导致额外的线程切换开销。任务调度以执行线程的常见方法是使用同步队列，称作任务队列。池中的线程等待队列中的任务，并把执行完的任务放入完成队列中。线程池模式一般分为两种：HS/HA半同步/半异步模式、L/F领导者与跟随者模式。

半同步/半异步模式又称为生产者消费者模式，是比较常见的实现方式，比较简单。分为同步层、队列层、异步层三层。同步层的主线程处理工作任务并存入工作队列，工作线程从工作队列取出任务进行处理，如果工作队列为空，则取不到任务的工作线程进入挂起状态。由于线程间有数据通信，因此不适于大数据量交换的场合。

领导者跟随者模式，在线程池中的线程可处在3种状态之一：领导者leader、追随者follower或工作者processor。任何时刻线程池只有一个领导者线程。事件到达时，领导者线程负责消息分离，并从处于追随者线程中选出一个来当继任领导者，然后将自身设置为工作者状态去处置该事件。处理完毕后工作者线程将自身的状态置为追随者。这一模式实现复杂，但避免了线程间交换任务数据，提高了CPU cache相似性。在ACE(Adaptive Communication Environment)中，提供了领导者跟随者模式实现。

2.5.6 Reactor模式

Reactor模型是一个事件触发模型，当有I／O读写操作准备就绪时，触发操作，然后从线程池中选择线程进行处理。这种机制能够极大地减少线程浪费在等待数据读写操作准备就绪上花费的时间，提高Java网络应用的并发性能。

分而治之是Reactor模型的模型思想，在Reactor模型中，客户端的请求事件分为I／O事件和非I／O事件。这两种事件的区别在于I／O事件只有当I／O准备就绪后才能进行，而非I／0事件是可以立即执行的，因此Reactor模型将这两种事件

分别进行处理。数据的读写属于I / O事件，编码、计算和解码属于非I / O事件。NIO 技术使得Java具备了非阻塞的读写能力，为这种Reactor模型的实现提供了很好的支持。

reactor设计模式，是一种基于事件驱动的设计模式。Reactor框架是ACE各个框架中最基础的一个框架，其他框架都或多或少地用到了Reactor框架。

在事件驱动的应用中，将一个或多个客户的服务请求分离（demultiplex）和调度（dispatch）给应用程序。在事件驱动的应用中，同步地、有序地处理同时接收的多个服务请求。

reactor模式与外观模式有点像。不过，观察者模式与单个事件源关联，而反应器模式则与多个事件源关联。当一个主体发生改变时，所有依属体都得到通知。

模型优点:

1.响应快，不必为单个同步时间所阻塞，虽然Reactor本身依然是同步的；

2.编程相对简单，可以最大程度的避免复杂的多线程及同步问题，并且避免了多线程/进程的切换开销；

3.可扩展性，可以方便的通过增加Reactor实例个数来充分利用CPU资源；

4.可复用性，reactor框架本身与具体事件处理逻辑无关，具有很高的复用性；

模型缺点:

1.相比传统的简单模型，Reactor增加了一定的复杂性，因而有一定的门槛，并且不易于调试。

2.Reactor模式需要底层的Synchronous Event Demultiplexer支持，比如Java中的Selector支持，操作系统的select系统调用支持，如果要自己实现Synchronous Event Demultiplexer可能不会有那么高效。

3.Reactor模式在IO读写数据时还是在同一个线程中实现的，即使使用多个Reactor机制的情况下，那些共享一个Reactor的Channel如果出现一个长时间的数据读写，会影响这个Reactor中其他Channel的相应时间，比如在大文件传输时，IO操作就会影响其他Client的相应时间，因而对这种操作，使用传统的Thread-Per-Connection或许是一个更好的选择，或则此时使用Proactor模式。

2.6 Mysql通信协议的研究

在MySQL数据库通信过程开始时，服务器会使用TCP监听一个本地socket 端口或本地socket链接。当一个客户端的连接请求到达，就会执行握手和权限验证。如果验证成功，会话开始。客户端发送消息，服务器会以一个适合该发送命令的数据类型的数据集或一条消息进行回复。当客户端发送完成后，会发送一个特殊的命令，告诉服务器已发送，然后会话结束。通信的基本单位是应用程序包。多个指令可以合成一个包；答复可以包含多个包。

2.6.1 交互过程

MySQL客户端与服务器的交互主要分为两个阶段：握手认证阶段和命令执行阶段。

握手认证阶段为客户端与服务器建立连接后进行，开始连接的时候，客服端发送握手初始化报文，然后客服端返回给服务器认证报文，最后服务器返回认证结果。

客户端认证成功后，会进入命令执行阶段，交互过程如下：客户端发送给服务器命令报文；服务器返回客户端命令执行结果。

2.6.2 协议基本类型

mysql协议中主要有以下四种基本类型：整型值，MySQL报文中整型值分别有l、2、3、4、8字节长度，使用小字节序传输；以null结束的字符串；长度编码的字符串字符串长度不固定，无null结束符；长度编码的二进制数据，数据长度不固定，长度值由数据前的1-9个字节决定，其中长度值所占的字节数不定，字节数由第1个字节决定。

2.6.3 mysql报文结构

报文分为报文头和报文数据两部分，其中报文头占用固定的4个字节，报文数据用于存放请求的内容及响应的数据，长度由报文头中的长度值决定。

2.6.4 报文类型

主要有下面几种报文类型：

1.登陆认证交互报文,建立TCP连接后，MySQL服务器向客户端

2.客服端命令请求报文,，对服务器的权能进行设置并等待客户端的验证包

3.服务器响应报文

2.6.5 Netty简介

Netty是NIO客户端服务器框架，可以快速轻松地开发诸如协议服务器和客户端之类的网络应用程序。它大大简化了网络编程流程，如TCP和UDP套接字服务器。

“快速和容易”并不意味着由此产生的应用程序将遭受可维护性或性能问题的困扰。 Netty已经通过执行许多协议（如FTP，SMTP， HTTP以及各种基于二进制和基于文本的传统协议）获得的经验进行了精心设计。因此，Netty成功地找到了一种方法来实现轻松的开发，性能，稳定性和灵活性，而无需妥协。

2.7 本章小结

本章主要介绍了本论文相关的关键的理论和技术，包括分布式数据库方面的理论，mysql协议的研究和学习，还有java相关的理论和技术。

第三章 系统分析

本章主要写了又那些数据库，每种数据库的特点。

3.1 需求分析

3.1.1 功能需求

数据库，简单来说就是电脑文件中存储的信息，用户可以对文件中的数据运行新增、截取、更新、删除等操作。数据库指的是以一定方式储存在一起、能为多个用户共享、具有尽可能小的冗余度、与应用程序彼此独立的数据集合。

数据库管理系统（英语：Database Management System，简称DBMS）是为管理数据库而设计的电脑软件系统，一般具有存储、截取、安全保障、备份等基础功能。数据库管理系统可以依据它所支持的数据库模型来作分类，最重要的是关系型数据库。

结构化查询语言（英语：Structured Query Language，缩写：SQL），是一种特殊目的之编程语言，用于数据库中的标准数据查询语言。不过各种通行的数据库系统在其实践过程中都对SQ L规范作了某些编改和扩充。所以，实际上不同数据库系统之间的SQL不能完全相互通用。本论文实现的Jsql是采用的mysql的SQL方言。

所以简单的说，要实现jsql数据库，对于用户来说，就是要实现mysql支持的sql语句。

mysql支持的sql语句有下面几种类型：

1.数据定义语句

2.数据操作语句

3.交易和锁定声明

4.复制语句

5.准备的SQL语句

6.复合语句

7.数据库管理语句

8.工具语句

当然，mysql本身是一个单机版数据库，所以有的sql语句类型不适合jsql这样的分布式数据库。比如，因为单机mysql无法满足应用的要求，所以有了复制的功能。但是，jsql本身就是一个分布式数据库，所以也就不需要再实现这样的复制功能了。

3.1.2 性能需求

Jsql系统是一个基于java语言实现的分布式数据库。它采用了和mysql一样的sql语法作为查询语言。

一方面，当查询只涉及到单结点上的数据库时，JSQL 的表现应当与传统数据库相当，即，当 JSQL 执行子计划中有对单表的增加，删除，修改，和查找操作时，这些操作的性能表现应当与 Mysql 相似。

另一方面，当查询涉及到跨结点的关系时。由于结点之间消息传递，就时延这一性能而言，在执行相同操作时，必然是分布式关系数据库不如传统数据库。但是，当面对的数据量足够大时，网络开销在总时间中的占比可能会较小，使得分布式数据库能有接近传统数据库的表现。

3.2 集群实现分析

集群（cluster）技术是一种较新的技术，通过集群技术，可以在付出较低成本的情况下获得在性能、可靠性、灵活性方面的相对较高的收益，其任务调度则是集群系统中的核心技术。集群是一组相互独立的、通过高速网络互联的计算机，它们构成了一个组，并以单一系统的模式加以管理。一个客户与集群相互作用时，集群像是一个独立的服务器。集群配置是用于提高可用性和可缩放性。

要实现数据库集群是一个非常困难的工作，一般人很难没有差错的完成这个功能。所以市面上出现了各种各样的功能库以减少开发者的工作。为了减少工作量，增加稳定性。论文决定采用hazelcast这个分布式集群库。

Hazelcast 没有任何中心节点（文中的节点可以理解为运行在任意服务器的独立jvm，下同），或者说Hazelcast 不需要特别指定一个中心节点。在运行的过程中，它自己选定集群中的某个节点作为中心点来管理所有的节点。

Hazelcast 的数据是分布式存储的。他会将数据尽量存储在需要使用该项数据的节点上，以实现数据去中心化的目的。在传统的数据存储模型中（MySql、MongDB、Redis 等等）数据都是独立于应用单独存放，当需要提升数据库的性能时，需要不断加固单个数据库应用的性能。即使是现在大量的数据库支持集群模式或读写分离，但是基本思路都是某几个库支持写入数据，其他的库不断的拷贝

更新数据副本。这样做的坏处一是会产生大量脏读的问题，二是消耗大量的资源来传递数据——从数据源频繁读写数据会耗费额外资源，当数据量增长或创建的主从服务越来越多时，这个消耗呈指数级增长。

使用 Hazelcast 可以有效的解决数据中心化问题。他将数据分散的存储在每个节点中，节点越多越分散。每个节点都有各自的应用服务，而Hazelcast集群会根据每个应用的数据使用情况分散存储这些数据，在应用过程中数据会尽量“靠近”应用存放。这些在集群中的数据共享整个集群的存储空间和计算资源。

集群中的节点是无中心化的，每个节点都有可能随时退出或随时进入。因此，在集群中存储的数据都会有一个备份（可以配置备份的个数，也可以关闭数据备份）。这样的方式有点类似于 hadoop，某项数据存放在一个节点时，在其他节点必定有至少一个备份存在。当某个节点退出时，节点上存放的数据会由备份数据替代，而集群会重新创建新的备份数据。

所有的 Hazelcast 功能只需引用一个jar包，除此之外，他不依赖任何第三方包。因此可以非常便捷高效的将其嵌入到各种应用服务器中，而不必担心带来额外的问题（jar包冲突、类型冲突等等）。他仅仅提供一系列分布式功能，而不需要绑定任何框架来使用，因此适用于任何场景。

利用hazelcast，我们开发了无中心节点的分布式集群数据库。

3.3 审计实现分析

数据库是任何商业和公共安全中最具有战略性的资产，通常都保存着重要的商业伙伴和客户信息，这些信息需要被保护起来，以防止竞争者和其他非法者获酬。互联网的急速发展使得企业数据库信息的价值及可访问性得到了提升，同时，也致使数据库信息资产面临严峻的挑战，概括起来主要表现在以下三个层面：

1．管理风险：主要表现为人员的职责、流程有待完善，内部员工的日常操作有待规范，第三方维护人员的操作监控失效等等，离职员工的后门，致使安全事件发生时，无法追溯并定位真实的操作者。其中典型的例子，曾为西藏移动进行设备安装工作的原深圳某高科技公司的工程师利用它为西藏移动做技术时使用的密码(此密码自工程师离开后一直没有更改)，轻松进入了西藏移动的服务器，再跳转到北京移动的服务器，从而通过修改系统的数据库轻松获得了14000个充值卡密码并获得380万元的利益。

2．技术风险：Oracle，SQL Server是一个庞大而复杂的系统，安全漏洞如溢出，注入层出不穷，每一次的CPUIl4J(Critical Patch Update)都疲丁奔命，而企业

和政府处于稳定性考虑，往往对补丁的跟进非常延后，更何况通过应用层的注入攻击使得数据库处于一个无辜受害的状态。

3．审计层面：现有的依赖于数据库日志文件的审计方法，存在诸多的弊端，比如：数据库审计功能的开启会影响数据库本身的性能、数据库日志文件本身存在被篡改的风险，难于体现审计信息的有效性和公正性。此外，对于海量数据的挖掘和迅速定位也是任何审计系统必须面对和解决的一个核心问题之一【1引。伴随着数据库信息价值以及可访问性提升，使得数据库面对来自内部和外部的安全风险大大增加，如违规越权操作、恶意入侵导致机密信息窃取泄漏，但事后却无法有效追溯和审计。

数据库审计的目标概括来说主要是三个方面：一是让管理者实时全面了解数据库实际发生的操作情况；二是在可疑行为发生时可以自动启动预先设置的告警流程，尽可能防范数据库风险的发生；三是一旦发生非法操作，触发事先设置好的防御策略，实行阻断，实现主动防御。因此，如何采取一种可信赖的综合途径，确保数据库活动记录的百分之百的被捕获是极为重要的，任何一种遗漏关键活动的行为，都会导致数据库安全上的错误判断，并且干扰数据库在运行时的性能。

本论文的安全审计子系统，能够根监视安全数据库系统中的相关活动，并执行相应动作。通过考．察、跟踪审计信息，审计员可以查看特定用户在过去一段时间内的数据访问行为，特定数据对象曾被访问的情况，以及曾试图对该数据库系统进行的非法操‘作等。审计员可以监督包括管理员在内的所有用户的操作。此外，还能通过对审计记录的分析，对系统的运行情况进行检查，排除可能存在的安全漏洞。

为实现这些审计目标，首先要对数据库的各种操作记录继续记录，防止记录被其他人更改，还要让安全管理员方便的查询审计记录。根据审计记录做个预警也是一种很重要的功能

3.4 编程模型分析

数据库的实现需要网络和磁盘的io，如何高效的利用操作系统和各种技术实现高性能的io模型是每个网络应该程序，包括数据库系统的重要目标。本小节介绍各种io模型。

说到IO模型，都会牵扯到同步、异步、阻塞、非阻塞这几个词。从词的表面上看，很多人都觉得很容易理解。但是细细一想，却总会发现有点摸不着头脑。自己也曾被这几个词弄的迷迷糊糊的，每次看相关资料弄明白了，然后很快又给搞混了。经历过这么几次之后，发现这东西必须得有所总结提炼才不至于再次混

为一谈。尤其是最近看到好几篇讲这个的文章，很多都有谬误，很容易把本来就搞不清楚的人弄的更加迷糊。

最适合IO模型的例子应该是咱们平常生活中的去餐馆吃饭这个场景，下文就结合这个来讲解一下经典的几个IO模型。在此之前，先需要说明以下几点： IO有内存IO、网络IO和磁盘IO三种，通常我们说的IO指的是后两者。阻塞和非阻塞，是函数/方法的实现方式，即在数据就绪之前是立刻返回还是等待，即发起IO请求是否会被阻塞。以文件IO为例,一个IO读过程是文件数据从磁盘到内核缓冲区再到用户内存的过程。同步与异步的区别主要在于数据从内核缓冲区到用户内存这个过程需不需要用户进程等待，即实际的IO读写是否阻塞请求进程。(网络IO把磁盘换做网卡即可)

3.4.1 同步阻塞

去餐馆吃饭，点一个自己最爱吃的盖浇饭，然后在原地等着一直到盖浇饭做好，自己端到餐桌就餐。这就是典型的同步阻塞。当厨师给你做饭的时候，你需要一直在那里等着。

网络编程中，读取客户端的数据需要调用recvfrom。在默认情况下，这个调用会一直阻塞直到数据接收完毕，就是一个同步阻塞的IO方式。这也是最简单的IO模型，在通常fd较少、就绪很快的情况下使用是没有问题的。

3.4.2 同步非阻塞

接着上面的例子，你每次点完饭就在那里等着，突然有一天你发现自己真傻。于是，你点完之后，就回桌子那里坐着，然后估计差不多了，就问老板饭好了没，如果好了就去端，没好的话就等一会再去问，依次循环直到饭做好。这就是同步非阻塞。

这种方式在编程中对socket设置ONONBLOCK即可。但此方式仅仅针对网络IO有效，对磁盘IO并没有作用。因为本地文件IO就没有被认为是阻塞，我们所说的网络IO的阻塞是因为网路IO有无限阻塞的可能，而本地文件除非是被锁住，否则是不可能无限阻塞的，因此只有锁这种情况下，ONONBLOCK才会有作用。而且，磁盘IO时要么数据在内核缓冲区中直接可以返回，要么需要调用物理设备去读取，这时候进程的其他工作都需要等待。因此，后续的IO复用和信号驱动IO对文件IO也是没有意义的。

此外，需要说明的一点是nginx和node中对于本地文件的IO 是用线程的方式模拟非阻塞的效果的，而对于静态文件的io ，使用zero copy(例如sendfile)的效率是非常高的。

3.4.3 IO复用

接着上面的列子，你点一份饭然后循环的去问好没好显然有点得不偿失，还不如就等在那里直到准备好，但是当你点了好几样饭菜的时候，你每次都去问一下所有饭菜的状态(未做好/已做好)肯定比你每次阻塞在那里等着好多了。当然，你问的时候是需要阻塞的，一直到有准备好的饭菜或者你等的不耐烦(超时)。这就引出了IO复用，也叫多路IO就绪通知。这是一种进程预先告知内核的能力，让内核发现进程指定的一个或多个IO条件就绪了，就通知进程。使得一个进程能在一连串的事件上等待。 IO复用的实现方式目前主要有select、poll和epoll。

select和poll的原理基本相同：注册待侦听的fd(这里的fd创建时最好使用非阻塞)；每次调用都去检查这些fd的状态，当有一个或者多个fd就绪的时候返回；返回结果中包括已就绪和未就绪的fd。相比select，poll解决了单个进程能够打开的文件描述符数量有限制这个问题：select受限于FDSIZE的限制，如果修改则需要修改这个宏重新编译内核；而po ll通过一个pollfd数组向内核传递需要关注的事件，避开了文件描述符数量限制。此外，select和poll共同具有的一个很大的缺点就是包含大量fd的数组被整体复制于用户态和内核态地址空间之间，开销会随着fd数量增多而线性增大。

select和poll就类似于上面说的就餐方式。但当你每次都去询问时，老板会把所有你点的饭菜都轮询一遍再告诉你情况，当大量饭菜很长时间都不能准备好的情况下是很低效的。于是，老板有些不耐烦了，就让厨师每做好一个菜就通知他。这样每次你再去问的时候，他会直接把已经准备好的菜告诉你，你再去端。这就是事件驱动IO就绪通知的方式epoll。 epoll的出现，解决了select、poll的缺点：基于事件驱动的方式，避免了每次都要把所有fd都扫描一遍； epollwait只返回就绪的fd； epoll使用nmap内存映射技术避免了内存复制的开销。 epoll的fd数量上限是操作系统的最大文件句柄数目,这个数目一般和内存有关，通常远大于1024。目前，epoll是Linux2.6下最高效的IO复用方式，也是Nginx、Node的I O实现方式。而在freeBSD下，kqueue是另一种类似于epoll的IO复用方式。此外，对于IO复用还有一个水平触发和边缘触发的概念：水平触发，就是当就绪的fd未被用户进程处理后，下一次查询依旧会返回，这是select和poll的触发方式；边缘触发，就是无论就绪的fd是否被处理，下一次不再返回。理论上性能更高，但是实现相当复杂，并且任何意外的丢失事件都会造成请求处理错误。epoll默认使用水平触发，通过相应选项可以使用边缘触发。

3.4.4 信号驱动

上文的就餐方式还是需要你每次都去问一下饭菜状况。于是，你再次不耐烦了，就跟老板说，哪个饭菜好了就通知我一声吧。然后就自己坐在桌子那里干自己的事情。更甚者，你可以把手机号留给老板，自己出门，等饭菜好了直接发条短信给你。这就类似信号驱动的IO模型。流程如下：开启套接字信号驱动IO功能；系统调用sigaction执行信号处理函数（非阻塞，立刻返回）；数据就绪，生成sigio信号，通过信号回调通知应用来读取数据。此种io方式存在的一个很大的问题：Linux中信号队列是有限制的，如果超过这个数字问题就无法读取数据。

3.4.5 异步非阻塞

之前的就餐方式，到最后总是需要你自己去把饭菜端到餐桌。这下你也不耐烦了，于是就告诉老板，能不能饭好了直接端到你的面前或者送到你的家里(外卖)。这就是异步非阻塞IO了。对比信号驱动IO，异步IO的主要区别在于：信号驱动由内核告诉我们何时可以开始一个IO操作(数据在内核缓冲区中)，而异步IO则由内核通知IO操作何时已经完成(数据已经在用户空间中)。

异步IO又叫做事件驱动IO，在Unix中，POSIX1003.1标准为异步方式访问文件定义了一套库函数，定义了AIO的一系列接口。使用aioread或者aiowrite发起异步IO操作，使用aioerror检查正在运行的IO操作的状态。但是其实现没有通过内核而是使用了多线程阻塞。此外，还有Linux自己实现的Native AIO，依赖两个函数：iosubmit和iogetevents，虽然io是非阻塞的，但仍需要主动去获取读写的状态。需要特别注意的是：AIO是I/O处理模式，是一种接口标准，各家操作系统可以实现也可以不实现。目前Linux中AIO的内核实现只对文件IO有效，如果要实现真正的AIO，需要用户自己来实现。

3.4.6 网络编程模型

上文讲述了UNIX环境的五种IO模型。基于这五种模型，在Java中，随着NIO和NIO2.0(AIO)的引入，一般具有以下几种网络编程模型： BIO， NIO， AIO。

BIO是一个典型的网络编程模型，是通常我们实现一个服务端程序的过程，步骤如下：主线程accept请求阻塞；请求到达，创建新的线程来处理这个套接字，完成对客户端的响应。主线程继续accept下一个请求这种模型有一个很大的问题是：当客户端连接增多时，服务端创建的线程也会暴涨，系统性能会急剧下降。因此，在此模型的基础上，类似于 tomcat 的bio connector，采用的是线程池来避

免对于每一个客户端都创建一个线程。有些地方把这种方式叫做伪异步IO(把请求抛到线程池中异步等待处理)。

JDK1.4开始引入了NIO类库，这里的NIO指的是Non-blcok IO ，主要是使用Selector多路复用器来实现。 Selector在Linux等主流操作系统上是通过epoll实现的。 NIO的实现流程，类似于select：创建ServerSocketChannel监听客户端连接并绑定监听端口；设置为非阻塞模式；创建Reactor线程，创建多路复用器(Selector)并启动线程；将ServerSocketChannel注册到Reactor线程的Select or上；监听accept事件； Selector在线程run方法中无线循环轮询准备就绪的Key； Selector监听到新的客户端接入，处理新的请求，完成tc p三次握手，建立物理连接；将新的客户端连接注册到Selector上，监听读操作；读取客户端发送的网络消息。客户端发送的数据就绪则读取客户端请求，进行处理。相比BIO，NIO的编程非常复杂。

JDK1.7引入NIO2.0，提供了异步文件通道和异步套接字通道的实现。其底层在windows上是通过IOCP，在Linux上是通过epoll来实现的 (LinuxAsynchronousChannelProvider. java,UnixAsynchronousServerSocketChannelImpl.java)。

创建AsynchronousServerSocketChannel ，绑定监听端口调用AsynchronousServerSocketChannel的法，传入自己实现的CompletionHandler。包括上一步，都是非阻塞的连接传入，回调CompletionHandler的completed方法，在里面，调用AsynchronousSocketChannel的read方法，传入负责处理数据的CompletionHandler。数据就绪，触发负责处理数据的CompletionHandler的completed方法。继续做下一步处理即可。写入操作类似，也需要传入CompletionHandler。其编程模型相比NIO有了不少的简化。

3.5 本章小结

存储部分的数据库主要分为两大类：关系型数据库与 NoSQL 数据库。

第四章 系统设计

4.1 总体架构设计

jsql是一个可以替代mysql的分布式java数据库

...

4.2 系统模块划分

下面是分布式环境下的部署图：

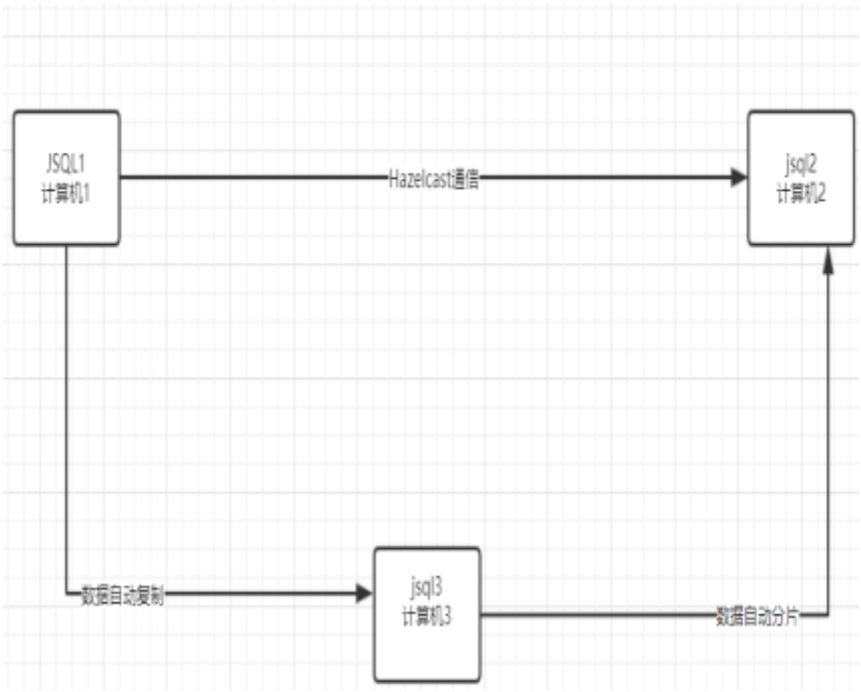

图 4-1 分布式环境下架构

图6-2是jsql的架构图

4.2.1 数据库模块设计

数据库主要是sqk到粗存

...

4.2.2 集群架构设计

主要用hazelcast

...

4.2.3 数据审计模块设计

数据库审计的

...

elasticsearch简介

...

grafana简介

...

审计的架构

...

...

4.3 各模块详细设计

本文主要是

...

4.3.1 数据库模块设计

主要分为3个部分

...

网络模块的设计

数据库主要是sqk到粗存

...

sql解析模块的设计

主要用hazelcast

...

存储引擎模块的设计

数据库审计的

...

4.3.2 集群架构的设计

主要用hazelcast

数据库审计的

...

4.3.3 数据审计架构的设计

...

审计数据库设计

数据库主要是sqk到粗存

...

审计管理器设计

主要用hazelcast

...

审计可视化模块的实现

数据库审计的

4.4 本章小结

分布式数据库

...

第五章 系统实现

本文主要是

...

5.1 数据库模块实现

主要分为3个部分

...

5.1.1 网络模块

数据库主要是sqk到粗存

...

5.1.2 sql解析模块

主要用hazelcast

...

5.1.3 存储引擎模块

数据库审计的

...

5.2 集群架构的实现

主要用hazelcast

...

5.2.1 通信模块

数据库主要是sqk到粗存

...

5.2.2 数据复制

主要用hazelcast

...

5.2.3 负载均衡

数据库审计的

...

5.3 数据审计架构的实现

...

5.3.1 审计数据库

数据库主要是sqk到粗存

...

5.3.2 审计管理器

主要用hazelcast

...

5.3.3 审计可视化模块的实现

数据库审计的

...

5.4 本章小结

分布式数据库

...

第六章 系统测试

...

6.1 测试环境

...

6.2 功能和性能测试

...

6.2.1 基本功能测试

……

6.2.2 数据库集群测试

……

6.3 数据库审计和可视化测试

...

6.4 本章小结

...

致 谢

在攻读硕士学位期间，首先衷心感谢我的导师曹晟老师...

参考文献

[1] 王浩刚, 聂在平. 基于mysql的分布式数据库的研究和实现[J]. 电子学报, 1999, 27(12):68–71

[2] X. F. Liu, B. Z. Wang, W. Shao. A marching-on-in-order scheme for exact attenuation constant extraction of lossy transmission lines[C]. China-Japan Joint Microwave Conference Proceedings, Chengdu, 2006, 527–529

[3] 竺可桢. 物理学[M]. 北京: 科学出版社, 1973, 56–60

[4] 陈念永. 基于mysql的分布式数据库的研究和实现[D]. 成都: 电子科技大学, 2001, 50–60

[5] 顾春. 基于mysql的分布式数据库的研究和实现[N]. 人民日报, 2012年3月31日

[6] 冯西桥. 基于mysql的分布式数据库的研究和实现[R]. 北京: 清华大学核能技术设计研究院, 1997年6月25日

[7] 肖珍新. 基于mysql的分布式数据库的研究和实现[P]. 中国, 实用新型专利, ZL201120085830.0, 2012年4月25日

[8] 中华人民共和国国家技术监督局. GB3100-3102. 中华人民共和国国家标准–量与单位[S]. 北京: 中国标准出版社, 1994年11月1日

[9] M. Clerc. Discrete particle swarm optimization: a fuzzy combinatorial box[EB/OL]. `http://clere.maurice.free.fr/pso/Fuzzy_Discrere_PSO/Fuzzy_DPSO.htm`, July 16, 2010