分布式数据库关键技术研究与应用

**院 系**：计算机科学与工程系

**导师**：张忠能

**学 号**：1110332066

**上海交通大学电子信息与电气工程学院**

**2014年8月**

# 摘要

当整合他们组织的数据库，考虑使用何种数据库类型时，许多公司都要做出艰难的决定。MySQL是世界上最大的开放源码的数据库，具有成本低，高的运行性能，是可靠和可扩展的网络数据库 (Vaswani, 2010)。另外这些特点同时有利于各种额外的高级功能，例如用于发展的高级管理工具和技术支持，及不同版本的管理应用。

诸如此类项目其所要达到的目的是对各类数据库的甄别来建立一项数据库模式，利用各种数据类型建立合理的数据表格。并且，利用在数据库里面使用索引及参照完整性，进而研究如何更加规范数据库的运行。但之前MySQL的发展中存在的一些规则对于研发者而言必须遵守。遵守这些规则的原因是因为能够帮助理解在MySQL系统中使用的约定和命令。

为了满足用户的要求，科学家考虑到用户的要求在各种各样方法，实现了一种分布式数据库，这样可以管理各个地方的节点并研究分布式数据库的一些关键技术。上述技术是指数据同步、安全性等等方面的。所谓数据同步实际上是构成了复制及维持众多数据库的重要操作，其隶属一种分散式的系统。这类系统的架构是基于一系列的递送服务器将报告反馈给中心数据库。能够利用递送服务器数量的增多来提升自身的规模。第二种分布式数据库技术有数据的安全，因为考虑到了数据同步，为了保持用户数据的安全，可以进行数据加密这样就保证用户与服务器都安全。

同时，当前许多应用程序都必须二十四小时一直运转，例如淘宝网，Facebook，人人网，ATM存取款机系统等，就上述类型的应用来说，一旦系统出现暂停，那么会产生众多难以解决的难题。所以分布式数据库和数据同步会保证用户可以保证二十小时得到所需要的信息，一个节点的故障不影响用户的操作。

笔者在这里对建立在MySQL基础上的分布式数据库中的各项关键技术加以研究。所谓关键技术其最重要作用是分析数据同步和文件加密。实现了高性能数据同步办法和在分布式数据库里面如何可以把用户的所有文件加密方法。设计并实现一款高性能、高可用性的分布式数据库架构，可以充分发挥MySQL Replication和文件加密这两种技术各自不同的优势。

关键词：分布式数可能，数据同步，文件加密，高性能，高可用性，MySQL，

RESEARCH AND APPLICATION OF KEY TECHNOLOGIES IN DISTRIBUTED DATABASE SYSTEMS

# ABSTRACT

Most companies are faced with tough decisions when considering which database technology to use when intending to integrate their organization’s database. MySQL is one of the world’s largest open source database, which enables cost effective delivery of high performing, reliable and scalable web based database (Vaswani, 2010). In addition to these characters one can as well benefit from various additional advance features such high management tools and technical support to develop, well managed applications different editions.

The main aim of this project is to determine the way of creating a database by identifying different type of data and how well to create data tables using different types of data. In this project I will analyze key technologies of distributed database system. Among the key technologies we will focus on MySQL database replication and file encryption based on distributed database. Furthermore one will learn how to normalize a database by implementing indexes and referential integrity in a database. But before the development of MySQL there are certain regulations that has to be followed by the developer. The reason for the regulation is to help understand how the conventions and commands are used in MySQL system.

Replication is considered to be the operation of copying and maintaining the objects of the multiple databases that belongs to a distributed system. The architecture of this kind of system is based around series of delivery servers that reports beck to the central database. The farms can be further scaled by adding more delivery severs (Zhiyuan, Hai & Xiaohui, 2005).

Meanwhile, a lot of applications have to keep in running state 24-hours a day without stopping, For example, a temporary shutdown in the ATM, medical & military monitor or stock processing system will lead to disastrous consequences.

The traditional method is to use the super computer system to work out the calculation task. But the price of super computer is too expensive, and the availability and scalability is not strong enough. Therefore, high performance and high availability database system has become the focus in the field of computer science.

**Keywords:** Distributed Database, Data Replication, Data encryption, High Performance, High, Availability, MySQL.

# 目录

[分布式数据库关键技术研究与应用 I](#_Toc401224259)

[摘要 II](#_Toc401224260)

[ABSTRACT III](#_Toc401224261)

[目录 IV](#_Toc401224262)

[第一章 绪论 1](#_Toc401224263)

[1.1背景 1](#_Toc401224264)

[1.2研究现状 1](#_Toc401224265)

[1.2.1关键技术 2](#_Toc401224266)

[1.2.2国外主流产品简介 2](#_Toc401224267)

[1.2.4国内现状与研究意义 4](#_Toc401224268)

[1.3研究内容与创新 4](#_Toc401224269)

[1.3.1研究内容 4](#_Toc401224270)

[1.3.2研究创新 5](#_Toc401224271)

[1.3.3数据库系统概念 6](#_Toc401224272)

[1.4 论文章节安排 8](#_Toc401224273)

[第二章 分布式数据库系统关键技术 5](#_Toc401224274)

[2.1数据同步 5](#_Toc401224275)

[2.1.1MySQL数据库同步 6](#_Toc401224276)

[2.2小结 7](#_Toc401224277)

[第三章 关键技术的研究和分析 8](#_Toc401224278)

[3.1 分布式数据库技术 8](#_Toc401224279)

[3.2同步（Replication）技术 9](#_Toc401224280)

[3.2.1复制解决的问题 9](#_Toc401224281)

[3.2.2复制如何工作 10](#_Toc401224282)

[3.2.3常见的同步架构 11](#_Toc401224283)

[3.3 基于分布式数据库的文件加密技术 12](#_Toc401224284)

[3.3.1 AES算法概述 13](#_Toc401224285)

[3.4常见应用场景和待解决的问题 14](#_Toc401224286)

[3.4.1用户典型使用场景分析 14](#_Toc401224287)

[3.4.2性能困境及解决方案概述 14](#_Toc401224288)

[3.5 本章小结 15](#_Toc401224289)

[第四章 系统架构设计与实现 16](#_Toc401224290)

[4.1分布式数据库的实现 16](#_Toc401224291)

[4.1.1应用场景分析 16](#_Toc401224292)

[4.1.2分布式数据库架构设计 17](#_Toc401224293)

[4.1.3分布式数据库架构实现 18](#_Toc401224294)

[4.1.4 功能实现及原理 20](#_Toc401224295)

[4.2 同步技术的实现 28](#_Toc401224296)

[4.2.1应用场景分析 28](#_Toc401224297)

[4.2.2同步架构设计 29](#_Toc401224298)

[4.2.3同步架构实现 31](#_Toc401224299)

[第五章 数据传输与加密处理 40](#_Toc401224300)

[5.1应用场景分析 40](#_Toc401224301)

[5.2文件加密架构设计 40](#_Toc401224302)

[5.3文件加密架构实现 42](#_Toc401224303)

[5.4 功能实现及原理 49](#_Toc401224304)

[5.5 本章小结 50](#_Toc401224305)

[第六章 总结与展望 51](#_Toc401224306)

[5.1研究总结 51](#_Toc401224307)

[5.2技术展望 52](#_Toc401224308)

[参考文献 53](#_Toc401224309)

# 第一章 绪论

## 1.1背景

科技进步尤其是智能终端技术的快速发展，现在网络愈发影响到普通人群的生活，各种娱乐、购物都逐渐在向在线方式过渡，这就催生了越来越多的数据库访问请求。各式查询请求不断地传送到数据库中，很多企业涉及到关键业务内容的数据库基本上每秒钟都可能要处理几千乃至上万次的Query请求，并且经常还需要进行Insert和Update操作，这就对数据库系统对操作请求的响应速度有比较高的要求；同时数据库作为现代企业的核心级部件，既存储了企业最核心的数据，包括客户信息、产品信息、订单数据等，而且还是企业核心业务的强依赖，几分钟的宕机就可能造成企业的大量收入损失和用户的强烈不满甚至流失，所以，数据库应当具备极高的可使用性，这是现实的需要。

正是在这样的前提下，传统的单机数据库系统正在面临着PB级数据集的存储和计算的难题，海量的信息和大量的Client请求，在这种情况下，数据库集群系统给处理过去DBMS遭遇的各式难题设计出了合理恰当的处理方式。

## 1.2研究现状

由于如今互联网时代，用户最常用的应用程序模型是B/S模型（浏览器/服务器模型），因此绝大多数电商信息管理平台都是采用此模型来实现的。电商信息管理平台有一个特点，如果今天的大多数的网页版有了那么大的成功，那是因为收到了用户的信任，因为用户可以随时随地访问信息。大多数这些平台的特点就是它们都采用分布式的框架来实现。对用户来说只要可以访问数据就可以了。但是对程序开发人员来说这是一个比较复杂的问题因为要考虑如何如果平台的一个站点遇到故障如何用户可以继续访问该站点的信息。因此，系统应当提升自身的可靠性。而为了达到上述目的，我们需要对分布式数据库相关的重要技术加以研究。关键技术包括数控同步，文件保密问题等技术。

### 1.2.1关键技术

研究各种复杂的分布式数据库技术，其实就是分析、比较其各自所包含的各项子技术性能和它们之间的协调运作能力，接下来主要就分布式数据库最为关键的核心技术加以简述，在此前提下也要注意其中的部分技术细节。

达到高性能要求的主要办法：

一、分布式数据库：在这里关键是寻找到好的方法以设计出高可靠性的分布式数据库。现在大多数的电商信息管理平台比如银行，SAP，一些网站都是采用这个技术。因为如果一个站点挂了，对用户来说没有任何影响，因为如果A的数据不能访问那么可以从B站点的数据库访问。

二、数据同步：该技术是分布式数据库的核心技术之一。假如作为分布式数据库缺少了该项技术，那么该数据库的系统也就失去了其最主要的价值。

三、文件加密：该技术也是分布式数据库的核心技术之一。这个主意是当用户和服务器互相转数据时，文件会不会被第三方人访问。该技术的主要作用是保证用户实时上传数据到服务器后对服务器加以加密。

### 1.2.2国外主流产品

当前主流的商业数据库产品都有自己的集群解决方案，下面做些简单介绍。

Oracle：

该产品为众多企业一级的用户设计了完整的集群方案，拥有利用集群技术实现多台互联微机处理信息的能力。而该产品中的Oracle RAC拥有了运用集群化硬件配置，向所有打包或定制应用程序提供极高的伸缩和可用性能的能力，而且作为单个系统，其自身极为简便易用。主要是可以从集群化配置的多节点来访问某一数据库，确保在软硬件出现故障时应用程序以及数据库的使用也不会出现问题，而且自身还具有硬件扩展的性能[3]。

此外，Oracle还提供了过程级复制和同步复制（Replication）功能，参与复制的节点之间的数据副本能够在任意时刻保持完全的一致性，保证了数据变化可以有效的运用到本地表以及其它节点数据副本里，一旦数据更新不成功那么全部事务要被完全回滚[4]。

SYBASE ASE[5]：

ASE性能的提升依赖于虚拟服务器架构，这一情况是SYBASE所拥有的独特的体系结构。其独立于操作系统以及有关软件能够确保ASE更加智能的对自身系统进行优化。让ASE性能得以提升以及成本被控制最重要的原因是其使用了专利技术的、能够进行自我调节的优化器以及查询引擎。其能够智能化调节各式各样的查询操作，而且能够自主过滤掉没有囊括有关信息的分区中的数据。AES还具有许多用于管理以及诊断数据库服务器的全新的特点，因此进一步的解决了自身成本。

ASE系统所具有的查询以及存储引擎其设计初衷是用来支持全新一代的网格计算以及集群技术。该系统综合了利用数据分区技术查询处理方式与处理集群问题的优化器技术，而且给事件驱动企业塑造了极为优秀的数据库平台。该系统和web services以及XML的架构能够降低系统内的彼此依赖程度，从而给予了应用开发更高的灵活度。[6]

Microsoft Cluster Service(MSCS) [7]：

在该产品的群集里MSCS软件能够实时的与四部在高速网络中运转的微机实现互联。一般来说，群集里的微机可以遵循“活动——活动”形式实现同类型的子系统以及功能的共享。换言之，就是全部集群微机能够自主利用共享负载形式共同进行工作，而且当其中某部微机发生故障的时候，其他的微机能够实时地承担故障微机的工作。MSCS最重要的作用是利用自己所拥有的容错能力提升应用程序的可使用性能。所谓容错能力是在有关处理当中把某节点里的故障应用程序转移到集群里正常节点中的一种能力。一旦故障排除，应用程序得以正常，那么集群将可以对原来的节点进行故障返回。MSCS可以确保不遗失所有同故障应用程序有关数据的同时，对集群里运转的应用程序加以故障恢复以及返回管理，而且可以在恢复时保证用户和应用程序状态。

在这里，SQL SERVER使用“出版物——订阅”形式对数据加以复制。源数据服务器称为出版服务器，尤其对数据进行发布。上述服务器将需要进行发布的数据的需要使用的更新内容的拷贝复制给分发服务器，而所谓分发服务器都拥有相应的分发数据库，对数据的各种变更加以存储，再由其将变更情况分发到订阅服务器。

### 1.2.4国内现状与研究意义

在了解了技术背景之后，我们了解到同国外的高新技术相比，我国大陆数据库集群技术相对落后，目前为止并无推出相应拥有自主知识产权的产品。

但这并不意味着在数据库集群领域国内便没有富有竞争力和创新性的研究成果。这是由于数据库集群要实现高度可使用性能，必须要精心设计负载均衡方案。因此，借鉴国际先进技术，研发相应的高度可使用性能的数据库集群负载均衡方案，其本身仍然是相当具有实际意义和商业价值的。

## 1.3研究内容与创新

### 1.3.1研究内容

企业规模的扩大必然会导致其分支遍布各地，而要想确保深处各地的企业分支都能够实时运用一致的数据信息，就必须要使用分布式数据库技术。

所谓分布式数据库指的是利用网络将物理上处于不同位置的各数据库单元相互连接构成的在逻辑上一致的数据库。这当中的数据库单元即是所谓的节点或者站点。该数据库拥有统一的管理系统，称之为分布式数据库管理系统。

该数据库有三个特征：第一，物理分散性，数据库内的数据分散保存在各场地上；第二，逻辑整体性，数据库逻辑上为整体，因此其实质上是完整集中的数据库系统；第三，场地自治性，不同场地中的数据受本地数据库管理系统DBMS管理，可以进行自我处理，可以独自实现该节点中的业务应用。

在该数据库里面需要思考的是，怎么才能确保外包数据可以被有效使用和安全保存，当然，依照国家有关隐私保护的法律法规确保数据不被外泄同样重要。安全数据库服务必须要应对许多难题，其中最为重要的就是隐私保护以及查询功能。合理运用密码技术从而达到两者间的高度平衡。

一方面要确保数据安全，另一方面要保证企业能够实时异地获取各项数据，大部分企业都使用了数据库同步技术来处理上述难题。不过上述办法在实际操作中也很困难，同步技术会因为网络、断电等情况导致并不能实时同步，从而导致数据遗失等。

现在众多的包括MySQL，Oracle，sybase，informix，Microsoft SQL server，Microsoft Access，Visual PoxPro等产品各以自己的特有的功能。

### 1.3.2研究创新

近年来数据库技术的应用正处于激烈变革的时期，当然这样说并非指过去的关系数据库技术最近有了新的重大进步，实际上是需要上述技术的客户需求在日益改变。大多情况下，客户不是仅仅寻找功能单一的软件，其需要更加高效、便捷、可控制、价格便宜的处理方案。

MySQL很早的时候就摆脱了单一数据库的行列，现在看来，已经不会有人将其看做一种玩具性质的软件。建立在MySQL技术之上的各类开源组件层出不穷，笔者在这里试图从下列几点介入来达成研究目的。

1. 实现分布式数据库：其由分布在不同节点之上的众多数据库系统构成，并为操作节点之上子数据库给出了合适的存取手段。该数据库的运用能够被当做整体数据库，不够其本质上还是处于在物理上分散的不同节点中的。理所当然的，处于不同节点中的子数据库其逻辑上存在关联。DDBS(Distributed Database system)里的数据分散在各个微机中，因此物理上有着极强的分布特性。尽管数据处于各个微机中，可是这只是表象，其本质还是构成了一个整体。在这里假设存在主数据库A城市数据库和从数据库B城市C城市···的数据库在主数据库当中保存部分数据，B、C等从数据库中保存的是同各自节点相关的数据。客户访问主数据库节点时，所需要请求的数据并不在主数据库里，而是在从数据库里，那么笔者在文中所要研究的分布式数据库技术就可以处理该难题。
2. 实现该类型数据库同步：大部分该类型数据库系统要想达到同步，主要运用通信程序以及数据复制技术。前者必须人工进行，而且数据更新不能达到实时动态。后者尽管使用简便，可是受限于网络情况，在进行上传下载时容易发生问题。本文主要是依照二期拟建网络环境，设计更为合理的分布式数据同步方案，要在确保通信环境的基础上，运用最优秀的数据库平台、数据处理以及通信等最新技术，研发全新同步机制，实现数据库数据动态自动与手动同步的目的，保证数据的完整。笔者在这里主要探讨MySQL的两种数据同步。
3. 对数据库内的文件实施加密处理。当前，电脑技术的应用已经满布于各个行业和领域，同时，Internet这一开放互通式网络的普及，人们越来越关注起在网络和电脑中的数据安全问题，尤其在保存和数据传播时。作为目前数据存储的主要依托和电商信息管理系统中最为重要的构成成分之一，数据库的安全尤为重要，因为其中保存了巨量的电子数据。由于这一特点，黑客作为网络安全的破坏者，往往将数据库作为攻击目标，而数据库的安全问题也被广泛关注，也是保障电商信息管理系统的重要一环。该文的主要目的就是探讨如何在分布式的数据库中做到文件的安全存储。如果有厉害的黑客的话，可以下载。但是下载的内容都是乱码。其主要的技术途径就是设计一套专门针对分布式数据库的信息安全服务系统以确保库内的数据安全。

## 1.4 论文章节安排

本文第一章节介绍研究背景、研究目标等相关内容。

第二章将重点探讨关于分布式数据库的相关基础性知识，例如MYSQL同步技术和文件加密技术。

第三章节主要针对符合本文研究方向的核心技术进行研究和分析，阐述了数据同步与文件加密技术实现方式和应用场景，继而表述本文所要研究解决的各个要点问题。

第四章节主要介绍了分布式数据库系统的关键技术的架构设计与实现，并对在分析和解决问题的过程中遇到的困难进行了解释和探讨。

第五章节即总结本文的主要研究内容并对将来的发展进行一定的展望。

# 第二章 理论与技术

## 2.1 分布式数据库系统概念

在阐述分布式数据库系统的概念之前，首先要明白何为数据库系统，它是由数据库（database）与管理数据的软件一同构成的的一体管理系统。它的发展是为了满足人们处理数据的要求，数据库系统是当前较为被大众适应的系统，它具备数据存储功能，并能够实现维护、传输及为其他系统提供数据的功能，换言之，它是一种数据存储器，是数据处理和管理的结合体。

对于database的研究，是一项集合电脑应用、专门数据管理系统和软件及理论知识为一体的工程，三者之间相互促进，电脑应用可以推动管理系统和软件的开发，而新的软件系统往往会激发创新理论领域，继而理论的突破又反作用与电脑和系统的进步。所以在计算机应用的突破上，database system的出现被视为起到了里程碑式的作用。计算机的主体功能随着数据库系统的出现从计算转为数据的处理，这一主体功能的转变促使计算机技术广泛地应用于各个行业以及普通大众的家庭生活中。在database system出现之前，对文件的持久处理并不是主要问题，但是对文件的快速访问一直限制着计算机技术的应用，即使许多的复杂的优化技术相应出现，但是不能普及到大众中得到应用。数据库系统的出现，为大众用户提供了一种相对简单的多的数据库语言。此外，计算机系统与数据库处理系统的是单独的，以便于不同或者多个用户对数据库的共享使用。故而数据库具有非常明显的独立和共享性能，这些特点使得数据应用的成本大大的降低，使得数据库系统的普及得到经济保障。可以说，正是由于数据库系统的引入，才让越来越多的大众计算机用户可以快速访问数据，并对数据存储、传输及应用得到实现，从而把计算机的应用从狭隘的研究室，搬进了越来越多的行业办公室和家庭之中。

根据不同数据库的大小，其系统也有一定的大小之分，大的系统诸如server、DB2和SQL等，而常见的中小系统有access和foxpro等。

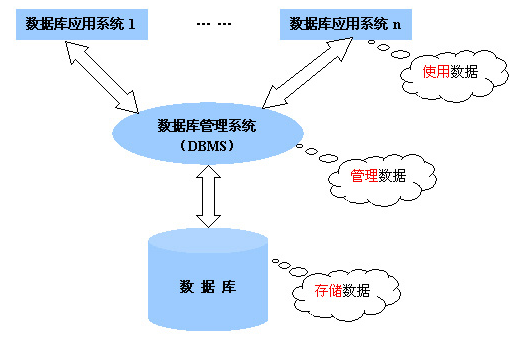


图 1-1 数据库系统原理图

Figure 1-1 Database Graph

数据库系统的分布式[8]类别可以被定义为由多个小型计算机系统和相应的配套数据库，以网络的形式实现之间连接，并构成统一的数据库，这样的分布式数据库系统，以小型计算机系统为重点之一，每个系统中都会被安装一份DBMS的copy副本，便于数据的处理。

明显的优劣势[9]：

1.降低费用：该数据库是可以拆分布置的，即为分布式。由于该系统的结构特殊性是能够帮助数据库分散安置的。各个用户都可以在其所在地的计算机上进行数据录入和查询，同时还可以帮助维护，进行局部的管理，从而减少了通信成本和购买适合集中使用的硬件和设备。由于一台计算器在分布式的数据库中所要处理的数据相对不多，所以其速度也会因为量少而增快。

2.提高系统整体可用性：因为其独特的单个计算机和数据库模式，没有了因为部分计算机和设备出现故障从而导致整个系统崩溃的情况。

3.其处理的能力和整个系统容量将非常便于扩展：分布式系统意味着组成部分的扩展性很强，在数据库中增加一个新的数据库，并不会对系统的运行产生影响。可以说这样的特征要比集中式的数据库扩展起来要方便的多，而且成本也远小一些。因为硬件的兼容性能有待验证，同时软件更新难度较大，往往集中式数据库系统的系统扩展和程序升级是代价高昂的。

4.事务性的管理成本较高：为了调解整个有相对独立个体组成的系统，其中的事务管理成本要相对高一些。

## 2.2 分布式数据库技术

数据库系统的分布式类别可以被定义为由多个小型计算机系统和相应的配套数据库，以网络的形式实现之间连接，并构成统一的数据库，这样的分布式数据库系统，以小型计算机系统为重点之一，每个系统中都会被安装一份DBMS的copy副本，便于数据的处理。

分布式数据库需要用到特有的软件系统，即为Distributed Software Systems，这是一种能够帮助数据库分散处理的系统，建立在网络连通的基础上，帮助多个计算机体系的整体结构处理任务。这个系统由多个子系统组成，分属各项功能。

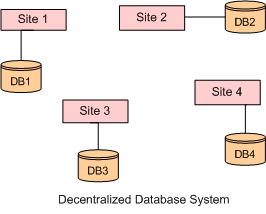


图3-1 分散数据库

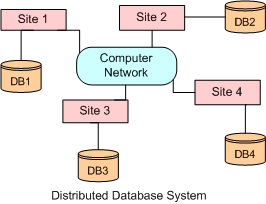


图3-2 分布式数据库

以上两个数据库可以看一般的数据库，每个站点都有自己的数据库，但是第二图，虽然每个站点都有自己的是数据库，可是所有的站点都有关联的。

其有以下特征：

1. 多数处理就地完成；

2．网络负责沟连布置在不同位置的计算机，并实现数据传输；

3.克服了中心数据库的弱点：降低了数据[传输](http://baike.baidu.com/view/389471.htm)代价；

4．整个系统的稳定性加强，即使有些部位和系统产生损坏，其余的仍可运行；

5.子数据库的分布相对开放，整个系统的扩充可行

6.调节性事务管理的成本较高

## 2.3 数据同步

所谓的数据同步，首先就要了解什么是数据库集群？而集群往往是指两个及以上个体的集合，而在这里就是指数据库服务器的集合成为一个类似于独个数据库的系统，这是一种映象，可以提供较为开放的数据信息以满足客户端的需求。这里有两个关键点：

1．需要一台以上的数据库服务器形成结合，不然不能叫做集群。

2. 透明的服务：集群的数据服务与单机的数据服务都是二进制的一种通讯协议。

从理论上来说，分布式与集群系统之间有一定的可比性。首先他们都是有多个单位组成，同时需要网络来将各类数据点结合在一起，形成一个计算逻辑上的统一体，而用户都能够从单一的数据点进行较为开放和透明的处理。但是它们仍具有明显的区别：

第一，两个系统的目的并不一样，集群系统的目的是为发挥计算机的集合优势，通过子数据库的共同并行处理，以期使得整个集群系统的性能得到加强。然而分布式系统建立的最终目的是一种分开的自主性和网内或者库内数据的开放性，并不需要强调系统的性能优势[11]。

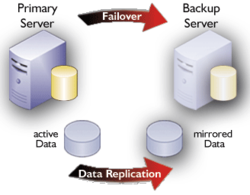


图 2-1 同步原理

Figure 2-1 Replication principle

第二，为了提高各个子节点的运算能力和数据处理能力，集群系统会采用较为高速的网络连接，通常是不低于100mb/s。因此节点之间的传输变得简单而又低廉，所以当一些节点的工作任务过于繁重的时候，可以通过网路传输给处于空闲状态的节点。从而实现系统的负载平衡。而相比较集群系统，分布式系统内的节点则主要强调相对独立的计算，所以在节点与节点之间的传输就不会被过于强调，一般采用较低速的局域或者广域网络进行节点连接。相比较而言，集群数据库的优势更加在集中协调，分散计算，对于一些较为大规模的服务器更为适用。

### 2.3.1 MySQL数据库同步

同步功能最早出现在MySQL3系统时，这一功能主要体现可以将数据从某个MySQL服务器拷贝到另外的服务器上。本章描述了MySQL的各种复制特性。阐述何为同步的概念，并探讨同步服务器的设置，与哪些为合适同步的服务器。此外还有这些概念里的一些问题和解决问题的方法介绍。MySQL 3.23.15及更高的版本支持单向同步。首先要明确一部服务器为主体，然后再选取一部乃至多部服务器为从属。主体服务器为master server,从属服务器为 slave server。例如，当master server将最新的文件内容更新到binlog即二进制日志中，同时创建并保证一个索引文件来保存这个日志的更新情况。该系统中，更新的内容会copy到slave服务器中。当一个新的slave与master相连接后，它就会自动搜索并告知master其日志更新的时间和位置，这样从服务器就可以按照更新记录开始更新工作，直到更新完毕后，等待主服务器的再次更新。

对于更新，必须强调一点，主服务器的明确是很重要的。虽然在一个服务器链中，可以做到同步共享，即从服务器也可以同时担任主服务器的功能，但是同步钱的原始更新工作必须保持在主服务器上完成，才能顺利的完成同步，否则会造成主从服务器的更新发生混乱，相互冲突。所以本文认为单向的同步过程是最为稳定的，而且更为容易管理。

* 有了master/slave机制后，就更稳健了。因为主服务器和从服务器之间可以切换，一点主发生问题，从可以直接代替其功能，并保证数据的稳定性。
* 部分数据的查询工作可以让从服务器进行一定分摊，以使系统的响应效率更为高效
* SELECT 查询就可以在slave上执行以减少master的负载。需要更新的数据要保证是在主服务器上完成的，这样主从服务器之间的同步才会更为稳定，但是一些不是属于更新内容的操作过多的时候，就需要从服务器来帮助主服务器来分担一些负载。
* 其他的优点还可以表现在独立于主服务器外的从器的拷贝功能。详情请看"5.7.1 Database Backups"。并系统如果一个数据库的节点不能使用，系统会自动使用该数据库的副本。

## 2.4 小结

本章节主要介绍了数据库系统的背景基础知识，数据库集群的概念及其相关的一些特点和核心技术，简单阐述了数据同步在分布式系统中的应用，以及MySQL的同步理念。综上所言，概括了数据库集群系统的一些应有的优异特点。

# 第三章 关键技术的研究和分析

本章的目的是研究高性能的分布数据库如何构造，并对业界比较成熟的分布式数据库相关技术，尤其是数据同步与文件加密系结构与高可用性支持、高性能支持、AES算法技术进行深入分析，在这样的研究上，研讨如何在MySQL数据库之上搭建高效率、高适用性的集群系统。

## 3.1 同步（Replication）技术

同步技术的原理主要是缓存设计的应用，在计算机的数据保存功能上，不一样的载体因为读写速度的不一而存在性能差别。所谓换成，即计算机在编写数据信息时，一般先保存其进入效率较高的介质中，一般为内存设备，而不是立即保存到效率较慢的介质，一般为外存设备。在引用读取时，系统也会首先查看内存中是否有数据备份，从而降低对外存的读取几率，以提高整个系统的运作效率。

### 3.1.1 复制解决的问题

数据库在体现其复制能力是表现出了一些优点：

1. 信息分布处理
2. 负荷协调
3. 备份能力较强
4. 较高的适用能力和稳定性

该类技术一般能够完善集群系统的相关性能，主要有如下3个方面[18]：

1. 可用性：多个数据节点组成的集群，即便有一个或者几个节点有损坏，其余的依然可以提供数据服务；
2. 性能：多个节点一般可以并行处理请求，从而避免单节点的性能瓶颈，一般至少可以提高读操作的并发性能；
3. 可扩展性：有限的处理能力一般都会发生在独个节点上，而增加其数量，是提升所在集群容纳量的有效途径。

### 3.1.2 复制如何工作

有3个环节[20]：

1.对于主服务器而言，数据更新，必须以事件的形式写入日志。

2.slave将master的binary log events拷贝到它的中继日志(relay log)；

3.从处理器更新relay log的时间记录，是改变其自身数据的起点，如3-1表述：



图3-3 Master节点的复制

Fig.3-3 Replication in the Master Node

该过程的第一部分就是master记录二进制日志。主服务器的binlog写入一般发生在数据更新之前，通过MySQL数据库，可以把事务内容写入二进制的日志文件，在这一过程中，哪怕事务语言同时交互运行。一旦binlog文件完成，主服务器才会让事务更新到储存中。

接下来，从服务器会首先copy来自主服务器的二进制日志到其relay log。首先，slave开始一个工作线程——I/O线程。这个线程是在主服务器中建立并打开一个相应的日常链接，然后通过二进制日子dumpprocessing读取主服务器的binlogevents，一旦连接到主服务器，就能够保持状态或称为sleep状态，等主服务器继续发生new events，从而继续写入 relay log。

SQL slave thread处理该过程的最后一步。这个线程的功能就是读取在relay log中的events，从而将从服务其里面的数据进行更新，保证与主的一致性。一般这个线程是与上述的I/O保持连接的，而relay log 一般就在OS Cache memory里，故而消耗比较少。

此外，在master中也有一个工作线程：与其余的MySQL的运作差不多，从服务器在主服务器里进行链接，从而运行主服务器的一个线程。Copy的过程往往会出现一个难以避免的障碍，copy是不能在主服务器更新的时候，在从服务器上同时进行。

### 3.1.3 常见的同步架构

独立的主服务器和一个或多个从服务器的节点：这样的组成在一对一的情况下完成copy system是最为基本并常见的。而从服务器除了对应主以外，其之间是无法沟通的。如下[21]：

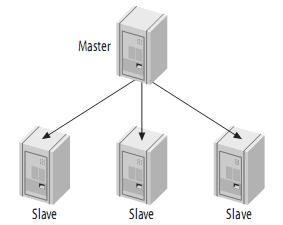


图3-4 一个master对应多个slaves

这种主从搭配的结构，主要是为了满足写程序较少，而读取较多的情况。从往往可以分担一些负荷来减轻主的运行。不过，较多的从服务器会让主在操作时产生满负荷或者超负荷运作，加之带宽的限制，就会形成较大的困扰。

此类结构相对简单，而又可以灵活运用，满足日常很多的应用中。另外在实际使用上还有一些建议如下：

(1) 不一样的从服务器的角色可以不一样，有的可以来索引，而有的可以作为引擎来存储。

(2) 用一个slave作为备用master，只进行复制；

(3) 往往可以使用一个较远的从服务器来担当备份的作用，一旦整个系统发生故障，可以从备份中得到还原

## 3.2 基于分布式数据库的文件加密技术

NDB Cluster[22] 是一种MySQL的较为特殊的版本，该类版本在分布式的环境下是较为实用的。它是一种引擎以用来存储。一个簇文件中能够承载较多的MySQL服务器。这个引擎一般实用在myqsl的第五代版本之后的二进制中，同时也在linux的rpm中担当存储驱动的角色。（注意，要想获得MySQL NDB Cluster 的功能，必须安装 MySQL-server 和 MySQL-max RPM）。

目前能够运行MySQL NDB Cluster 的操作系统有Linux、Mac OS X和Solaris（一些用户通报成功地在FreeBSD上运行了MySQL NDB Cluster ，但MySQL AB公司尚未正式支持该特性）。

目前，快速发展的international互联网使得分布式数据库得到极大的应用和高速的进步，也造成了一些应用中的安全隐患。在开放的环境下，去保障安全的分布式系统，较为苦难，这也就需要更多的探讨和研究。分布式的安全隐患相比集中式而言，有一些独特之处：

1.鉴于分布式数据库的独特结构，其必然会出现分布式的用户群，他们处在不同的节点，对于整个数据库系统而言，这些用户的身份确认较为繁琐。

2.数据库内的数据是是由其内不同用户上传，可以被其他用户分享的能够分享的，这就形成不一样的用户面对需要数据时的安全设置不同。只依靠用户的安全级别的不同来限制数据的访问权限，会导致一些泄露事件，尤其是一些没有授权的数据。

3.此类安全问题的两个常见类型是：一是自然出现的故障，如网络形成的限制和单个节点形成的障碍，这就需要去分析在发生该类故障时，如何应对，便于系统的稳定运行和恢复。而是黑客等造成的故障，这就需要去分析在黑客发起破坏性攻击时，如何保障系统的稳定运行和各类数据的保密工作。

### 3.2.1 AES算法概述

随着电脑的广泛使用与各个行业和家庭，以及开放网络的形成，涉及到较大内容的数据安全保存和传播问题。而数据库作为承载数据的核心部件，极为可能成为网络黑客关照的对象，故而在研究数据库的安全处理上，是整个系统中信息安全管理中的重要一环。这就造成了一个必须要去研究和处理的问题。一些企业在这一方面都有一定的防护措施，例如访问限制、局域网专用或者设定防火墙等安全系统。此外，在数据库设备供应商提供的产品内也具备一定的安全防护性能，例如DBMs的生产商。其运用的技术有针对数据库的备份，以便恢复的功能，还有一些关于审计或者安全策略的应用。然后，这样的处置并不能满足需要，明文数据是最大的隐患。

AES，即为高级加密标准，是一种高级的密钥生成算法，是美国政府使用的一种算法。以往的DES加密算法，有着诸多漏洞，被AES所取代，目前已在全球范围内得到通用。这种算法经历了5个年份的检测，最终被NIST于01年发布，并在次年的5月份正式成为可用的一种新的标准。

这一类的密码，是一种相对应的分组密钥结构。所采用的密钥有三个主要的位数，分别是一百二十八、一百九十八和二百五十六位，而分组密钥的长度则以一百二十八位为主，能够应用到各种各样的软硬件中。五年的测试里，美国国家标准与技术研究院于98年进行了第一次测试，形成候选的算法，共有十五个，并在99年完成了第二次测试。紧接着在两千年正式采用比利时的专家祖安和文森特的算法，即为密钥生成算法，AES。虽然当时大量的应用DES技术是建立在大量的元件产出的基础上，但是其安全性能已被突破，高级的3级DES密码是当时增加数据安全时的最后选择，但是DES迟早要被AES代替。 分组密码其实本不是在理论上密码研究的最新概念，下一代需要开发的密码算法，应该是流密码。

AES 算法基于排列和置换运算。所谓排列，是一种顺序的再一次调整，而置换则是数据单元之间的代替和交换，在AES环节，会出现较多种不一样的方式来计算上述安排。

作为一种反复执行的、采用对称分组形式的密码，这种密钥有三个主要的位数，分别是一百二十八、一百九十八和二百五十六位，加之一百二十八位的加密，变得更加安全。相比一般的密码中的密钥组成，此类密码，以对称而又相同的密钥进行加解密，以处理数据。该类算法即为本位所要应用到分布式系统中的来确保数据安全的密码。

## 3.3 常见应用场景和待解决的问题

当前高性能数据库系统主要应用于大型商业Web系统中，例如电商、支付、CRM、ATM，广告等大型业务系统中，其应用场景具备高度可概括性，本段落将对系统所需解决的用户需求问题以及解决方案做简单的解释和阐述，对于具体的解决方案和实现方法将在本文第四章节做具体介绍。

### 3.3.1 用户典型使用场景分析

以普通用户的网上购物行为举例，其主要的使用过程如下：

1. 打开网站，选择相应的商品种类，进入该种类下的商品集合页面；
2. 按照自己的偏好，设置一些筛选条件，然后可能会从不同纬度进行多次排序，最后对多个商品点击查看其详情；

3.产品的色彩、大小以及各类附加配件等属性；；

4.之后要么对该类产品进行确认订单后直接买下，要么先收藏抑或放进购物车中；

5.填写订单，读取用户账号信息，折算各种优惠条件，最后在线支付完成。

在如上这些步骤中，还需要持续记录每位用户的访问路径，实时分析并动态展现当前某特定范围内，最热、最近、购买最多的商品，并推荐给符合该商品的定位的潜在用户。

### 3.3.2 性能困境及解决方案概述

通过上一部分，我们知道在普通用户轻松在线购物的同时，网站系统的后台程序正在进行一系列运算，并且对整体系统而言，最大的性能困境就是海量的大并发读写操作。

各类购物网站，其主要的业务往往包括用户界面、货物陈列、订单管理、支付界面和其他一些例如红包的辅助功能，在各个不同的环节有着不同的业务，这些业务成为网站的单个模块，模块的建立和各个接口的设置对于应用而言较为要紧，通常会遵循聚合性高、连接口收缩的的基本要素，这样可以提高整个系统的可用性。在一些模块的功能有大有小，将较小的安排在一起，放大较大模块的应用，进行单独安置。

在这个系统的运行中，会有较多的难度，一般体现在单个的数据库与分布式的库中数据是否能够同步。这样比如说一个用户在修改或者进行一些操作，是否可以进行同步，这个问题用分布式数据库的一个很重要技术“事务”技术来解决这个问题。还有文件加密。加密处理时，假若数据能够做到同步，那么文件是否也能可以同时完成。这些问题都在本文里面介绍了的很清楚。

## 3.4 本章小结

本章节主要内容是针对符合本文研究方向的核心技术进行了研究和分析，阐述了其技术的实现方式和应用场景；并以电商网站应对用户的在线购物行为举例，揭示了本文所要解决的主要技术问题，并给出了概述性的阐述。

# 第四章 系统架构设计与实现

## 4.1分布式数据库的实现

从目前互联网技术以及人们消费习惯的发展情况看，用户使用在线网购的行为习惯正在逐步养成并大规模普及。此时，电商网站内在处理货品展现的时候，急需解决的问题就是应对用户查看货品信息的海量大并发请求，笔者会在下文中详述通过运用MySQL，达到构建和实现分布式数据库框架之目标。

### 4.1.1应用场景分析

在该应用场景下，现有的传统系统架构面临着巨大的并发请求压力以及性能瓶颈，急需解决的主要问题如下：

1. 电商类网站往往存在大量用户和大量商品，这部分用户最经常的使用流程就是打开静态页面选中某一分类进行商品查询，然后点击查看商品详情。这就意味着系统在任意时间都面临着大量的数据查询请求，并且同时伴随着高并发状态，这对系统主数据库，尤其是其主存储器都是致命的威胁。
2. 大量商品本身已经会对查询造成相当大的压力，并且这些商品可能还会伴随着一些附加属性，例如颜色、尺寸、附件等，这些从属性数据需要和商品本身交叉关联后，动态实时查询其实际的库存量信息，这对系统的读操作又带来了进一步威胁。
3. 在同一段时间内，网站的用户数量其实是可控的，即正常情况下短时间内不会产生大量新注册用户（用户量的自然增长速度往往会稳定在一个固定水平）；但此时系统内产生的订单数往往不可控，并且订单量一定是呈递增趋势的，这就导致了“查询订单”这个读操作的性能表现会越来越差。
4. 网站需要始终呈可用状态，尤其是购物类网站如果经常出现全站的访问困难将会对其口碑造成无法估量的消极影响，进而产生用户放弃使用该网站或者转投其他网站，造成客户外流的结果。

### 4.1.2分布式数据库架构设计

因为笔者设计和建构的模型系统是以B/S模型为基础的，因此本文采用HTML+CSS+Javascript作为它的前台语言，并将PHP作为解析前台中的用户操并负责与数据库通信的语言，数据库则采用了MySQL。MySQL的特点是成本低廉，运行速度快且占用空间小，特别是其还兼有源代码开放功能，所以笔者以此来作为构建管理系统的方式。

首先，笔者构建的管理系统由两个部分组成，一是中央数据库，二是N个归属不同地方的基层站点。中央数据库主要管理管理员和注册用户的账号信息和每一个地方数据库的备份规则。每一个地方站点管理两个数据库，一个是用户发布在本地的消息数据库正本，另一个则是用户在异地发布的消息数据库的副本。具体的数据库模型为如下。

中央数据库

表4-1用户数据库表

1. **tb\_account (uid,name,password,email,regdate,photo,description,type)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 域名 | Uid | Name | Password | Email | Regdate | Photo | Description | Type |
| 含义 | 编号 | 账号名 | 密码 | 邮箱 | 注册日期 | 照片 | 说明 | 用户类型 |

其中，type=1表示系统管理员，type=0表示普通用户。当用户注册时，系统会自动分配给他权限。如果type为一就是管理员权限，如果type为0就是普通用户权限。

表4-2 词典表

1. **tb\_dictionary (tid,siteno,version)**

|  |  |  |  |
| --- | --- | --- | --- |
| 域名 | Tid | Siteno | Version |
| 含义 | 编号 | 地方编号  （1：北京，2：上海，3：广州） | 版本  (1：正本，2:副本) |

其中，Version表示目前已经建成并可以使用地方基层站点数据库。Siteno表示用户正在使用的数据库站点。在本文中假定了1为北京，2为上海，3为广州。Version里面如果1就是说明我们正在使用的是正本数据库，如果为2就是说前段时间1数据库（正本）不能使用，系统就自动分配2（副本）数据库。

【注意】

第一次运行系统时，根据具体情况需要填该表,以便系统运行时参考它。

例如；

表4-3 案例表

|  |  |  |
| --- | --- | --- |
| Tid | Siteno | Version |
| 1 | 1 | 1 |
| 2 | 2 | 1 |
| 3 | 3 | 2 |

如上表所示，北京(1)，上海(2)的数据库正本可用，但是广州(3)的数据库正本不能用，因此只能用副本（广州数据表的副本在北京站点）。

表4-4 文件表

1. **tb\_accesslist(mid,filename,description,admin,users,timestamp)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 域名 | Mid | filename | description | admin | users | Timestamp |
| 含义 | 编号 | 文件名 | 文件描述 | 上传文件的管理员 | 可以访问的用户列表 | 上传日期 |

Users（用户列表）以#来隔开。例如，管理员选择UserA,UserB,UserC，则Users域为UserA#UserB#UserC。

地方站点

有两个数据库；本地的正本(main\_db)，异地的副本(backup\_db)。所有数据库都维护两个表。例如，北京数据库的表为如下；

表4-5 发布消息表

1. **tb\_message (mid,title,content,price,contact,images,author,timestamp)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 域名 | Mid | Title | Content | price | Contact | Images | Author | timestamp |
| 含义 | 编号 | 标题 | 消息正文 | 价格 | 联系方式 | 图片 | 发布者 | 发布日期 |

保存用户在北京发布的消息。 用户可以发布N个图片。Images保存用户发布的图片名。**用户上传的图片保存在文件体系而非数据库里。**多个图片名以#隔开。例如，用户发布了图片img1.jpg,img2.jpg,img3.jpg，Images为img1.jpg#img2.jpg#img3.jpg.

表4-6 记录用户操作的表

1. **tmp\_message(mid,author,timestamp,flag)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 域名 | Mid | Author | Timestamp | Flag |
| 含义 | 编号 | 发布者 | 发布日期 | 1:添加消息，2：删除消息，3：删除用户 |

用户在该城市（例如：北京）的所有数据痕迹都要保存并同步导入新副本。这个表在副本和正本都有。因为同步的时候要首先判断我们正在使用的是正本还是副本，然后检查这个表是否两个站点一致然后继续同步。

核心功能实现:

1. 注册，登陆功能

登陆功能相对来说比较简单，故略之。

注册包括：

* 首先，后台（php）分析用户提交的注册（登录）表单。
* 其次，后台（php）查看账号有没有冲突（对账号数据库的sql查询），如果不冲突就添加账号。由于两个用户同时可以用相同的用户名注册，因此冲突查询和添加查询要满足**原子性**。原子性可通过数据库自带的 Transaction来进行运作。
  + - 1. 管理员功能；添加用户，发布消息，删除消息，删除用户功能

这些功能都没有太大的难度。对敏感用户和不合规用户进行删除处理时，除了删除该用户所使用的ID账号，同时也需要将用户发布、评论过的全部信息一并删除。显然，这两个操作必须要满足原子性的，要么两个删除操作都成功，要么两个删除操作都失败。

需要注意的是，为了实现同步，所有涉及到用户和消息的操作都需要记录下来。因而进行消息上线发布的操作时，也需要将其记录到本地（如：北京）的tmp\_message表里。

* + - 1. 注册用户功能；发布消息，删除消息功能

这些功能都跟管理员功能差不多，故略之。

### 4.1.3分布式数据库架构实现

由于此业务场景的单一性，我们通过在实验环境搭建该架构的最小实现即可验证其可行性，其最小实现架构如下图：

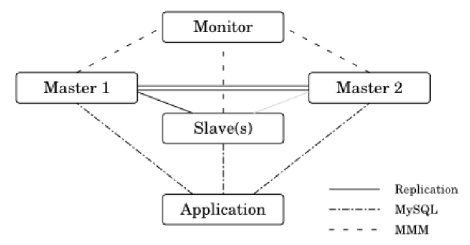


图4-1双主复制结构

Fig.4-1 Double Master Node Replication

具体服务器清单如下：

表4-6 服务器角色列表

Table4-6 Server Role List

|  |  |  |  |
| --- | --- | --- | --- |
| 主机名 | IP地址 | 角色 | Server\_Id |
| DB1  DB2  DB3 | 192.168.24.128  192.168.24.129  192.168.24.130 | Main\_DB  master1  slave1 | 1  2  3 |

先修改配置文件，其中db1 修改conn\_account.php，内容如下：

1. /\*
2. This is for the user/admin account homeIp
3. $homeIP is the IP Address of Account DB.
4. \*/
5. $homeIP="192.168.0.108";
6. $conn = @MySQL\_connect($homeIP,"root","")
7. or die("连接数据库失败：" . MySQL\_error());
8. MySQL\_select\_db("account\_db", $conn);
9. MySQL\_query("set character set 'utf-8'");
10. MySQL\_query("set names 'utf-8'");
11. unset($homeIP);
12. ?>

Db1 站点1 conn\_data1.php，有以下内容

1. <?php
2. $homeIP="192.168.0.108"; //main server database
3. $masterIP="192.168.0.109"; //master database
4. $slaveIP="192.168.24.128"; //slave database
5. //This is for the site1
6. //First Read (@)Info from Account DB.
7. $home = @MySQL\_connect($homeIP,"root","") or
8. die("连接H数据库失败：" . MySQL\_error());
9. MySQL\_select\_db("account\_db", $home) or
10. die("打开H数据库失败");
11. MySQL\_query("BEGIN",$home);
12. $res1=MySQL\_query("select version from tb\_dictionary where siteno=1",$home);

站点1，站点2和站点3的代码是一样的。只是masterIP和slaveIP会有变化。

### 功能实现及原理

1. 注册，登录。

首先，后台（php）分析用户提交的注册（登录）表单。

其次，后台（php）查看账号有没有冲突（对账号数据库的sql查询），如果不冲突就添加账号。由于两个用户同时可以用相同的用户名注册，因此冲突查询和添加查询要满足**原子性**。原子性可通过数据库自带的Transaction来进行运作。

1. 管理员系统

图4-管理员系统功能

* + 1. **添加用户**：管理员可以添加用户
    2. **运行同步文件**：只有管理员可以运行同步文件
    3. **删除用户**：管理员可以删除用户。同时可以删除几个用户。需要注意的是，在进行用户删除操作时，同时也需要将用户发布、评论过的全部信息一并删除，而且此过程需要符合**原子性**。
    4. **修改用户：**管理员可以修改用户
    5. 删除消息：管理员可以按站点删除用户发布了的消息。
    6. 上传文件：管理员可以按用户的请求给用户分配文件。管理员选择几个用户并且只让这些用户查看通知文件（其他没有选的用户没法查看）。系统管理员在上传有关文件时，可以授权哪些用户可以查看到该上传文件。

【注意】

管理员上传的文件首先经过文件加密，然后再放到文件系统里。

* + 1. 删除文件：管理员可以删除文件。

【注意】：为了实现同步，所有涉及到用户和消息的操作都需要记录下来。所以，删除用户的相关操作痕迹也需要记录到本地（如：北京）的tmp\_message表里。

* + 1. 以过滤消息。

系统管理员能通过其用户ID名称以及城市分布来查看消息或者过滤信息。

【注意】

为了实现同步，所有涉及到用户和消息的操作都需要记录下来。因此，过滤消息的操作也要记录到本市（例如：北京）的tmp\_message表里。

1. 用户系统



图4- 用户系统功能

如果只想查看消息，就不登陆也能查看所有用户在该城市发布的信息。如果想管理消息（发布或删除），就需要登录。

* 1. 发布消息。

用户发布的信息就保存到本市的数据库（或副本）。

【注意】

为了实现同步，所有涉及到用户和消息的操作都需要记录下来。所以有关的消息发布操作也需要同时记录到本地（如：北京）的tmp\_message表里。

* 1. 删除消息。

普通用户可通过指定的城市，来查阅在其他城市上线发布的消息并且可以选择删除该消息。

【注意】

为了实现同步，所有涉及到用户和消息的操作都需要记录下来。因此，删除消息的操作也要记录到本市（例如：北京）的tmp\_message表里。

## 4.2 同步技术的实现

在建立分布式数据库过程中，信息同步是最为关键的技术要求，如果无法实现信息同步，分布式数据库的实际使用价值便大打折扣。笔者在下文中将重点论述在分布式数据库中怎样进行构建与实现同步。

### 4.2.1应用场景分析

由于如今互联网时代，用户最常用的应用程序模型是B/S模型（浏览器/服务器模型），因此绝大多数公司，政府都是采用此模型来实现的很多需要的功能。 现在无论是政府，或者公司大部分都通过网站方式来给用户看到他们，用户更理解他们。典型的代表为58同城，赶集网和百姓网，淘宝网。。。这些管理网站必须要实现如下几个功能；

* 1. 用户管理功能

可以把用户分类，首先，系统管理员拥有最高的权限，能够管理或者删除注册用户和普通用户的账号以及其在系统中发布的各类消息。注册用户则可以在网站上发帖以及阅读其他用户发布的信息。普通用户实际上是没有对应的用户账号，他的主要功能只是在线查看别人发布的消息。

* 1. 管理员功能
  + 添加注册用户
  + 删除注册用户

需要注意的是， 管理员删除这些用户的同时，也要把他们之前发布的所有消息一起删除。

* + 过滤有害消息

如果其中含有有害信息的，管理员可以将该消息过滤处理。

* + 发布通知文件

运行信息管理网站的时候，常常需要将一些通知文件发送到几个特定的用户。例如，如果管理员查看系统中发布的信息的时候，发现几个固定的用户经常发布一些含有不良信息的消息。在这种情况下，管理员将警告文件（或警告通知）发送给这几个用户。再如，如果系统需要升级或维护的时候，系统管理员需要将这些消息通知到所有用户。**该功能是本文新提出的功能。**

* 1. 注册用户功能
     + 发布新的消息

发布消息的功能是需要注册并登陆之后才能使用。发布信息时也要将信息的描述，图片，联系方式，价格等一些参数一起发布。

* + - 删除自己发布的消息

删除消息的功能也是需要注册并登陆之后才能使用。

* 1. 一般用户功能
     + 查看已发布的消息

考虑到在大都数信息管理网站用户查看已发布的信息是不需要登陆到系统，甚至不需要注册账号的，因此实际上本文中的该功能也是不需要用户账号的。

经过上述分析梳理，管理系统所实现的功能可以简要总结为下图所示。

信息管理系统

一般用户功能

管理员功能

注册用户功能

查看消息

查看消息

删除消息

发布消息

发布文件

查看并过滤消息

删除用户

添加用户

注册 登陆

**图4.2 本系统主要功能**

### 4.2.2同步架构设计

实现分布式架构的信息管理系统时，最主要考虑的是不同数据库之间的同步问题。为实现数据库同步技术达成的目的，笔者将在所有的基层站点同时维护两个数据库；一个是用户在本地发布的信息的正本，另一个则是用户在其他站点发布的信息的副本。但是，这里有几个问题需要解决；首先每一个地方数据库的副本放到哪一个地方站点是在哪儿如何规定的？第二，在信息同步时，同步的内容是数据库全部信息或者局限于需要更替的信息？最后何时进行同步？

首先，由于地方数据库的备份规则与具体的地方数据库的内容无关，因此可以在中央数据库维护一张表，用它来描述具体的备份规则。例如，在广州和上海两个地方站点维护北京数据库的副本。这样，就有了一个正本和两个副本。出于便于分析研究的目的，笔者假设只存在一个正本、一个副本的情形。无论维护多少个副本，其同步原理是一样的。与此同时，分布式数据库的系统中，无法排除在任一节点上存在产生错误的几率。例如，上述例子中，北京站点的北京数据库突然不能用，则为了不影响用户使用北京站点，必须将北京站点和北京数据库的副本（在上海）一起绑定，进而使得整个分布式数据库系统的高容错性（Redundncy）得以实现。这样，分布式数据库中一个节点的故障对用户来说是透明的，以后用户的所有操作（发布信息，删除信息等）都是在副本数据库上进行的。因此备份规则还要描述备份的方向，也就是说，将正本数据库的内容同步到副本数据库，还是将副本数据库同步到正本数据库。在北京站点读取中央数据库的备份规则之后，如果备份规则规定的版本（例如，如果是正本）数据库连接失败，则说明数据库很可能不能使用，因此立刻将备份规则改为副本。因而，系统中全部用户均能够在副本数据库开展有效操作。

其次，在正本和副本数据库之间进行同步时，最好只将更新的内容同步到副本中，而不是同步整个数据库的内容。虽然这样可以减少不必要的开销的同时还可以提高同步的效率，但是进行同步时如何让系统知道哪些信息是被更新的也是一个富有挑战性的问题。为了解决这个问题，可以在不同的地方数据库都维护一张表，它是专门用来记录用户在该城市的所有记录（添加信息，删除信息，删除用户等）。实际上，该表相当于的日志文件(Log file)，它记录那些被更新的内容。因而我们在进行信息同步的时候，可以以该日志文件为蓝本，从而进行有关信息同步操作。理论上，同步操作完毕之后，正本数据库和副本数据库维护的信息是相同的，因此同步结束之后，必须将日志表清空。

第三，数据库与数据库之间开展信息同步要以强化信息数据整体系统的容错性为方针；即使分布式系统中的一个节点挂了，用户也可以通过它的副本数据库来管理信息。备份方式一种是提供一个接口，管理员可以通过此接口来手动调用同步功能。使用该方法，系统管理员就能选择任何时间段或者时间节点来启动信息同步的功能。但是，由于管理员要手动进行同步操作，因此使用该方法的话，管理员必须要负责整个系统的容错性。第二种方法是系统自动同步，也就是说，系统选择一个时间段来进行同步操作。但是，考虑到在信息网站中24小时内随时都有可能发生新的用户操作，因此适当的选择数据库同步的时间尤为重要。例如，由于凌晨3-4点用户的操作两相对来说比较少，因此可以在每天凌晨3点钟进行同步操作。

由于每一个站点负责用户在该站点发布的信息而且用户发布的信息可能是在正本数据库或者在副本数据库，因此可以在每一个站点后台实现相应的同步功能。例如，北京站点可以实现北京数据库的正本与副本之间的同步，上海站点则负责上海数据库正本与副本之间的同步。这样，将整个系统的同步所带来的大量工作量可以有效分摊在各个地方站点中进行。

需要申明的是，以上所有程序及操作都必须符合原子性这一条件。这些包括地方站点读取并更新中央数据库的备份规则表的操作，涉及到同步的所有操作（包括读取日志表，读取备份的源数据库并写到目标数据库）等等。

中央数据库

登录、注册

用户

广州站点(3)

上海站点(2)

北京站点(1)

发布、删除消息

数据库3(正本)

数据库2(正本)

数据库1(正本)

数据库2(副本)

数据库1(副本)

数据库3(副本)

**图4.3 分布式数据库框架模型**

对一般用户来说，将用户发布的消息存放到哪一个地方节点（正本或副本）应该是透明的。因此，信息系统必须授权给一般用户能正常运作的数据库句柄，只有如此，用户在将来的操作中，才可以在该数据库里操作。前面已经提到中央数据库中有一个表是专门负责管理地方数据库的备份规则的，而该表所记录所有基层站点记录使用的数据库是正本还是副本需要明确，如果是正本，则说明该站点用的数据库是在本站点，如果是副本，则说明其他站点中保存该站点用的数据库的备份。

当用户打开地方站点的网页时，系统要将该站点用的数据库中的消息数据返回给用户。筛选主要方式为：

* 经由中央信息数据库的备份法则表来查阅当前数据库的版本。为了方便，现在假设；城市是北京，可用的版本为‘正本’。
* 毗连正本信息数据库

如果毗连正常，则将数据库的毗连句柄(Handle)反馈给用户。

如果连接失败，意味着正本数据库挂掉了，就试图连接副本数据库。

* 如果副本数据库连接成功，就把副本的句柄返回给用户，并将备份规则表的值改为‘副本’。
* 如果副本数据库连接失败，就意味着没有可用的数据库（实际上是几乎不可能的），系统不能运行。

经过以上两个程序，基本上能确定反馈给一般用户的数据库句柄是可靠的。

### 4.2.3同步架构实现

本文介绍了同步问题的实现与应用。同步分布式数据库的工作原理如下图所示：

1. 数据库选择

b. 读取日志数据表

c. 依次执行日志表中的记录

d. 清空日志数据表

*图4-4同步流程图1*



*图4-5：同步流程图2*

**以上图示表明**：

* 信息同步的前提条件是正本和副本数据库运行正常，若其中任何一个数据库无法正常使用，便不能开展信息同步工作。从中央数据库读取备份规则表(tb\_dictionary)。该表记录当前可用的数据库版本（正本或副本）。如果是正本，那么信息同步是按照正本到副本的方向进行的；如果是副本，则说明同步的方向是从副本到正本。
* 当前可用的数据库中有一个日志表，它记录用户在该地点的所有操作（添加消息，删除消息，删除用户）。由于之前的用户操作都是在该数据库上进行的，因此同步时只需将这些新的操作同步到副本数据库即可。
* 日记表通常有以下3种记录情况：添加和删除两个大类，其中删除又包括删除记录或者删除用户。假设同步的方向是从正本到副本。如果是【添加记录】;则会逐条阅读正本数据库中的信息记录并一一对应复制至副本数据库;如果是【删除记录】，则根据日志表中的用户名和时间戳来查找相应的副本数据库中的记录，并删除它;如果是【删除用户】，则从副本数据库删除所有与日志表的记录匹配的用户的消息。
* 信息同步完成后，会清空有关日志数据，所以空的日志数据表表明了正本、副本数据库的内容一致。

以下代码就是上面的流程图的实现代码sync\_daemon.php：

1. <?php
2. //First Read (@)Info from Account DB.
3. $homeIP="192.168.24.128"; //main server database
4. $masterIP="192.168.24.129"; //master database
5. $slaveIP="192.168.24.129"; //slave database
6. $home= @MySQL\_connect($homeIP,"root","") or
7. die("连接H数据库失败：" . MySQL\_error());
8. MySQL\_select\_db("account\_db", $home) or
9. die("打开H数据库失败");
10. MySQL\_query("BEGIN",$home);
11. $res1=MySQL\_query("select version from tb\_dictionary where siteno=1");
12. $fromDB=$toDB=$res2=NULL;
13. $fromDBName=$toDBName=NULL;
14. $flag=-1;
15. if ($row=MySQL\_fetch\_assoc($res1)){
16. if ($row['version']==1){
17. //主数据库连接
18. $fromDB=@MySQL\_connect($masterIP,"root","");
19. if ($fromDB){
20. $fromDBName="main\_db";
21. //MySQL\_select\_db("main\_db",$fromDB);
22. //从数据库连接
23. $toDB=@MySQL\_connect($slaveIP,"root","");
24. if (!$toDB){
25. //从数据库连接失败,不能进行备份.
26. MySQL\_close($fromDB);
27. echo("Oops!,No available Slave DB!");
28. $flag=-2;
29. }
30. else{
31. $toDBName="backup\_db";//MySQL\_select\_db("backup\_db",$toDB);
32. }
33. }
34. else{
35. //主数据库连接失败
36. MySQL\_close($fromDB);
37. //将(@)标志赋给从数据库，并退出
38. $flag=2;
39. }
40. }
41. else{
42. //从数据库连接
43. $fromDB=@MySQL\_connect($slaveIP,"root","");
44. if ($fromDB){
45. $fromDBName="backup\_db";//MySQL\_select\_db("backup\_db",$fromDB);
46. //主数据库连接
47. $toDB=@MySQL\_connect($masterIP,"root","");
48. if (!$toDB){
49. //主数据库连接失败,不能进行备份.
50. MySQL\_close($fromDB);
51. echo("Oops!,No available Slave DB!");
52. $flag=-2;
53. }
54. else{
55. $toDBName="main\_db";//MySQL\_select\_db("main\_db",$toDB);
56. }
57. }
58. else{
59. //从数据库连接失败
60. MySQL\_close($fromDB);
61. //将(@)标志赋给主数据库，并退出
62. $flag=1;
63. }
64. }
66. }
67. //情况1: flag==-2(待同步的数据库出问题)
68. if ($flag==-2){
69. if ($res1){
70. MySQL\_query("COMMIT");
71. }
72. else{
73. MySQL\_query("ROLLBACK");
74. }
75. MySQL\_query("END");
76. if ($home!=$fromDB&&$home!=$toDB)
77. MySQL\_close($home);
78. exit();
79. }
80. //情况1：flag=1 or 2(from数据库连接失败)
81. if ($flag>0){
82. $res2=MySQL\_query(
83. "update tb\_dictionary set version=$flag where siteno=1",$home);
84. }
85. if ($res1&&($res2||$flag==-1)){
86. MySQL\_query("COMMIT",$home);
87. }
88. else{
89. MySQL\_query("ROLLBACK",$home);
90. }
91. MySQL\_query("END",$home);
92. if (!$fromDB||!$toDB) exit();
93. //正常情况：fromDB,toDB都能用
94. //AGAIN.........
95. MySQL\_select\_db($fromDBName,$fromDB);
96. MySQL\_select\_db($toDBName,$toDB);
97. MySQL\_query("set names 'utf8'",$fromDB);
98. MySQL\_query("set names 'utf8'",$toDB);
99. //锁住两个表。
100. $allowCommit=true;
101. MySQL\_query("BEGIN",$fromDB);
102. MySQL\_query("BEGIN",$toDB);
103. $res1=MySQL\_query("select \* from tmp\_message",$fromDB);
104. if (!$res1){
105. $allowCommit=false;
106. }
107. while ($row=MySQL\_fetch\_assoc($res1)){
108. $author=$row['author'];
109. $timestamp=$row['timestamp'];
110. $flag=$row['flag'];
111. switch($flag){
112. case 1: //添加信息
113. $res2=MySQL\_query("select \* from tb\_message where
114. author='$author' and timestamp='$timestamp'",$fromDB);
115. if (!$res2){
116. $allowCommit=false;
117. }
118. if ($row=MySQL\_fetch\_assoc($res2)){
119. $title=$row['title'];
120. $content=$row['content'];
121. $price=$row['price'];
122. $contact=$row['contact'];
123. $imgdata=$row['images'];
124. MySQL\_query("insert into tb\_message values
125. (NULL,'$title','$content','$price',
126. '$contact','$imgdata','$author','$timestamp');",$toDB) or
127. $allowCommit=false;
128. }
129. break;
130. case 2: //删除信息
131. $res2=MySQL\_query("delete from tb\_message where
132. author='$author' and timestamp='$timestamp'",$toDB) or
133. $allowCommit=false;
134. break;
135. case 3: //删除用户
136. $res2=MySQL\_query("delete from tb\_message where
137. author='$author'",$toDB) or
138. $allowCommit=false;
139. break;
140. default:
141. break;
142. }
143. }
144. //tmp\_message表清空.
145. MySQL\_query("truncate table tmp\_message",$fromDB) or $allowCommit=false;
146. MySQL\_query("truncate table tmp\_message",$toDB) or $allowCommit=false;
147. if ($allowCommit){
148. MySQL\_query("COMMIT",$fromDB);
149. MySQL\_query("COMMIT",$toDB);
150. }
151. else{
152. MySQL\_query("ROLLBACK",$fromDB);
153. MySQL\_query("ROLLBACK",$toDB);
154. }
155. MySQL\_query("END",$fromDB);
156. MySQL\_query("END",$toDB);
157. //释放变量
158. unset($title);
159. unset($author);
160. unset($content);
161. unset($timestamp);
162. unset($flag);
163. unset($res1);
164. unset($res2);
165. //关闭句柄
166. MySQL\_close($fromDB);
167. MySQL\_close($toDB);
168. echo 'Success!';
169. ?>

# 第五章 数据传输与加密处理

当今世界对那些能够发现，处理和交流我们日常生活信息及商业数据的技术需求是非常大的。相较于动态数据及静态数据，保护知识产权，个人身份和其他敏感信息则更为重要。

这种保护的很大一部分是通过密码学来实现的。密码学是一种代码加密学科，是将明文从可理解的形式转化为不可理解的密文，从而确保了数据在不安全的或公共/共享信道中的保密性。密码保护可防止数据被未经许可方的授权进行使用或变更。从传统意义上来说，密码技术面临的挑战是计算的复杂性及计算成本的昂贵性。

一般来说，一个密码系统是通过数学和算法流程将可读的文本编码转换为加密的密文然后再将密文转换为原文。用于加密/解密的算法被称为密码。密码操作通常是由一个或一组密钥控制的，许多因素如：数据的保密性，完整性，真实性和性能都决定了终端用户不同的加密标准。

## 5.1应用场景分析

当前市场不仅仅对数据库的整体性能要求越来越高，分布式数据库的稳定和安全也逐渐成为引人注目的问题和因素。

为什么密码学在当今享有如此高的市场热度？尤其在企业备受推崇？首先，自2005年1月以来，美国有超过3.45亿包含敏感的个人信息的记录受到安全性攻击，并且越来越迅猛，攻击的复杂度也越来越高，也越来越难以察觉。目前，安全性攻击已从多台计算机上随机攻击发展到有目的性的针对一些高价值的银行或政府系统的敏感的账务或个人身份信息进行攻击。在今天的高度虚拟化的计算环境中，多个虚拟机共享相同的硬件资源，如同装了许多鸡蛋的篮子一样，硬件资源需要更多的安全保护。加密技术为它们提供了深层次的保护，这样，即使系统受损，信息丢失，通过对称和非对称的加密方案，信息仍为不可用模式。加密技术也为诸如健康保险携带和责任法案（HIPAA），Sarbanes-Oxley 法令（SOX），支付卡行业（PCI）条例这些对美国公司有深远影响的法令提供越来越为重要的数据保护。需要注意的是，仅当数据在公共网络传输时，需要对HIPAA进行加密，但是，HiTECH法案增加了大量的安全性布署要求以加强没有加密时HIPPA的安全性。

当前一个流行的加密标准是高级加密标准（AES），2001年美国政府采用了该标准。目前，它被广泛用于软件生态系统中以保护网络流量，个人数据，和企业IT基础设施。AES的应用包括商业安全，数据库和存储系统中的数据安全，虚拟主机的安全迁移以及全磁盘加密。根据IDC的数据加密使用调查，使用最为广泛的应用是企业数据库和档案备份。全磁盘加密也受到了很多的关注。

根据IDC的加密使用调查，自2005以来，超过9千万用户已经开始重视潜在的个人信息安全漏洞。3. 例如：早上，我们打开配有WI-FI的便携电脑，属性窗口显示的是经企业授权加密的AES-CCMP数据；在局域网内工作或在安全的网站进行购物，在你的浏览器上启动锁定图标，这些都表明了安全套接字层（SSL）为您提供了一个安全的连接，SSL是一个密码协议，为网络通信如Internet通信提供数据的安全和完整性，SSL以及最近的传输层安全（TLS）协议为端到端的网络段连接提供安全加密。

## 5.2文件加密架构设计

文件加密构架方法一般有两种，一是对称加密法，二是非对称加密法。

对称加密：加密和解密用相同的密钥。DES和AES是典型的对称密码。由于现实中要加密的内容(例如文件)大小比较大，因此通常采用分组密码或序列密码的形式。笔者在此使用采用分组密码，所以只对分组密码进行论述，其他不作展开。

分组密码是把固定长度的明文模块，通过技术转换，变成等长的密文，这个转换过程需要使用密钥进行管控，同理，我们可以依靠同一密钥，进行反向转换来解密。

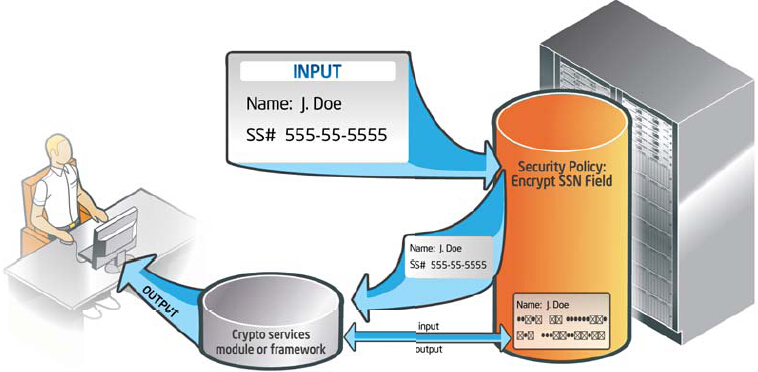


图5-1 Application-level encryption AES usage model

显然，从在个人层面上讲，安全性问题充斥在我们的日常生活中，那么在企业及其服务器层面中呢？许多政府和公司服务器中含有大量的个人可识别信息以及财务信息，这是分布式客户的要求，这使得服务器级别的加密尤为重要，特别是当下恶意代码（恶意软件）越来越多的攻击各类计算机。

同样的，黑客的攻击变得更加复杂和难以检测。更甚者，攻击的类型正不断转变。过去，许多黑客事后留名以炫耀自己的技术以及受感染的计算机数量之多，现在，这种类型的黑客正被那些出于金钱和参与有组织犯罪的黑客所取代，他们的目标不是尽可能多地去感染计算机，而是悄无声息的感染那些有高价值的电脑，这些电脑可能是银行以及能够访问金融和或个人信息的机构。加密技术是最好、最后的一道防御手段，即使系统受损，信息受到窃取，加密技术也能让信息保留在不可用状态。常见的分组密码有：密码分组连接（CBC），电子编码本（ECB），密文反馈（CFB）等三种。

非对称加密法，也可以称作公钥加密，是指加密解密用两个密钥：公开密钥和私有密钥。如果使用公开密钥进加密，那么就需要使用相对应的私有密钥解密；同理，如果使用私有密钥加密，那么就需要使用相对应的公有密钥解密。RSA加密法即是经典的公钥加密。

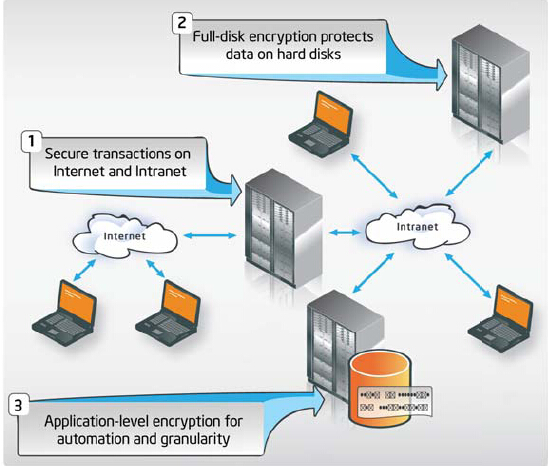


图5-2 Three AES usage models

以下图示即是本文加密、解密流程:

管理员

发布文件

通知文件

加密/解密

(黑匣子)

文件系统

注册用户

查看文件

通知文件

**图5.3 文件加密解密视图**

## 5.3文件加密架构实现

在分布式数据库的时间里，介绍了加密的实现问题。系统用户需要发送相应的请求给系统管理员，管理员在收到并清除用户请求指令后，就给有关用户上传相关的文件。当然，管理员所上传的所有文件都是加密的，只有相对应的用户才能够下载到该文件，其他用户则无法下载和使用。



图5-4 文件管理

以下为aes.php文件。在本文件实现了文件加密算法

1. <?php
2. /\*\*
3. \* Pure-PHP implementation of AES.
4. \*
5. \* Uses mcrypt, if available/possible, and an internal implementation, otherwise.
6. \*
7. \* PHP versions 4 and 5
8. \*
9. \* If {@link Crypt\_AES::setKeyLength() setKeyLength()} isn't called, it'll be calculated from
10. \* {@link Crypt\_AES::setKey() setKey()}. ie. if the key is 128-bits, the key length will be 128-bits. If it's 136-bits
11. \* it'll be null-padded to 192-bits and 192 bits will be the key length until {@link Crypt\_AES::setKey() setKey()}
12. \* is called, again, at which point, it'll be recalculated.
13. \*
14. \* Since Crypt\_AES extends Crypt\_Rijndael, some functions are available to be called that, in the context of AES, don't
15. \* make a whole lot of sense. {@link Crypt\_AES::setBlockLength() setBlockLength()}, for instance. Calling that function,
16. \* however possible, won't do anything (AES has a fixed block length whereas Rijndael has a variable one).
17. \*
18. \* Here's a short example of how to use this library:
19. \* <code>
20. \* <?php
21. \* include('Crypt/AES.php');
22. \*
23. \* $aes = new Crypt\_AES();
24. \*
25. \* $aes->setKey('abcdefghijklmnop');
26. \*
27. \* $size = 10 \* 1024;
28. \* $plaintext = '';
29. \* for ($i = 0; $i < $size; $i++) {
30. \* $plaintext.= 'a';
31. \* }
32. \*
33. \* echo $aes->decrypt($aes->encrypt($plaintext));
34. \* ?>
35. \* </code>
36. \*
37. \* LICENSE: Permission is hereby granted, free of charge, to any person obtaining a copy
38. \* of this software and associated documentation files (the "Software"), to deal
39. \* in the Software without restriction, including without limitation the rights
40. \* to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
41. \* copies of the Software, and to permit persons to whom the Software is
42. \* furnished to do so, subject to the following conditions:
43. \*
44. \* The above copyright notice and this permission notice shall be included in
45. \* all copies or substantial portions of the Software.
46. \*
47. \* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
48. \* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
49. \* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
50. \* AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
51. \* LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
52. \* OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
53. \* THE SOFTWARE.
54. \*
55. \* @category Crypt
56. \* @package Crypt\_AES
57. \* @author Jim Wigginton <terrafrost@php.net>
58. \* @copyright MMVIII Jim Wigginton
59. \* @license http://www.opensource.org/licenses/mit-license.html MIT License
60. \* @link http://phpseclib.sourceforge.net
61. \*/
62. /\*\*
63. \* Include Crypt\_Rijndael
64. \*/
65. if (!class\_exists('Crypt\_Rijndael')) {
66. include\_once 'Rijndael.php';
67. }
68. /\*\*#@+
69. \* @access public
70. \* @see Crypt\_AES::encrypt()
71. \* @see Crypt\_AES::decrypt()
72. \*/
73. /\*\*
74. \* Encrypt / decrypt using the Counter mode.
75. \*
76. \* Set to -1 since that's what Crypt/Random.php uses to index the CTR mode.
77. \*
78. \* @link http://en.wikipedia.org/wiki/Block\_cipher\_modes\_of\_operation#Counter\_.28CTR.29
79. \*/
80. define('CRYPT\_AES\_MODE\_CTR', CRYPT\_MODE\_CTR);
81. /\*\*
82. \* Encrypt / decrypt using the Electronic Code Book mode.
83. \*
84. \* @link http://en.wikipedia.org/wiki/Block\_cipher\_modes\_of\_operation#Electronic\_codebook\_.28ECB.29
85. \*/
86. define('CRYPT\_AES\_MODE\_ECB', CRYPT\_MODE\_ECB);
87. /\*\*
88. \* Encrypt / decrypt using the Code Book Chaining mode.
89. \*
90. \* @link http://en.wikipedia.org/wiki/Block\_cipher\_modes\_of\_operation#Cipher-block\_chaining\_.28CBC.29
91. \*/
92. define('CRYPT\_AES\_MODE\_CBC', CRYPT\_MODE\_CBC);
93. /\*\*
94. \* Encrypt / decrypt using the Cipher Feedback mode.
95. \*
96. \* @link http://en.wikipedia.org/wiki/Block\_cipher\_modes\_of\_operation#Cipher\_feedback\_.28CFB.29
97. \*/
98. define('CRYPT\_AES\_MODE\_CFB', CRYPT\_MODE\_CFB);
99. /\*\*
100. \* Encrypt / decrypt using the Cipher Feedback mode.
101. \*
102. \* @link http://en.wikipedia.org/wiki/Block\_cipher\_modes\_of\_operation#Output\_feedback\_.28OFB.29
103. \*/
104. define('CRYPT\_AES\_MODE\_OFB', CRYPT\_MODE\_OFB);
105. /\*\*#@-\*/
106. /\*\*#@+
107. \* @access private
108. \* @see Crypt\_AES::Crypt\_AES()
109. \*/
110. /\*\*
111. \* Toggles the internal implementation
112. \*/
113. define('CRYPT\_AES\_MODE\_INTERNAL', CRYPT\_MODE\_INTERNAL);
114. /\*\*
115. \* Toggles the mcrypt implementation
116. \*/
117. define('CRYPT\_AES\_MODE\_MCRYPT', CRYPT\_MODE\_MCRYPT);
118. /\*\*#@-\*/
119. /\*\*
120. \* Pure-PHP implementation of AES.
121. \*
122. \* @package Crypt\_AES
123. \* @author Jim Wigginton <terrafrost@php.net>
124. \* @version 0.1.0
125. \* @access public
126. \*/
127. class Crypt\_AES extends Crypt\_Rijndael
128. {
129. /\*\*
130. \* The namespace used by the cipher for its constants.
131. \*
132. \* @see Crypt\_Base::const\_namespace
133. \* @var String
134. \* @access private
135. \*/
136. var $const\_namespace = 'AES';
137. /\*\*
138. \* Default Constructor.
139. \*
140. \* Determines whether or not the mcrypt extension should be used.
141. \*
142. \* $mode could be:
143. \*
144. \* - CRYPT\_AES\_MODE\_ECB
145. \*
146. \* - CRYPT\_AES\_MODE\_CBC
147. \*
148. \* - CRYPT\_AES\_MODE\_CTR
149. \*
150. \* - CRYPT\_AES\_MODE\_CFB
151. \*
152. \* - CRYPT\_AES\_MODE\_OFB
153. \*
154. \* If not explictly set, CRYPT\_AES\_MODE\_CBC will be used.
155. \*
156. \* @see Crypt\_Rijndael::Crypt\_Rijndael()
157. \* @see Crypt\_Base::Crypt\_Base()
158. \* @param optional Integer $mode
159. \* @access public
160. \*/
161. function Crypt\_AES($mode = CRYPT\_AES\_MODE\_CBC)
162. {
163. parent::Crypt\_Rijndael($mode);
164. }
165. /\*\*
166. \* Dummy function
167. \*
168. \* Since Crypt\_AES extends Crypt\_Rijndael, this function is, technically, available, but it doesn't do anything.
169. \*
170. \* @see Crypt\_Rijndael::setBlockLength()
171. \* @access public
172. \* @param Integer $length
173. \*/
174. function setBlockLength($length)
175. {
176. return;
177. }
178. }

在管理员界面，管理员上传所有文件都经过加密uploadFile.php：

1. <?php
2. header('content-type:text/html;charset=utf-8');
3. session\_start();
4. ini\_set("display\_errors", 0);
5. if (!isset($\_SESSION['adminname'])){
6. header("location: signin.html");
7. }
8. if (isset($\_POST['uploadFile'])){
9. $users=$\_POST['users'];
10. $admin=$\_SESSION['adminname'];
11. $description=addslashes($\_POST['file\_description']);
12. $timestamp=date('Y-m-d H:i:s');
13. $userStr="";
14. $temp = explode(".", $\_FILES["file"]["name"]);
15. $extension = end($temp);
16. $fileName=$admin.MD5($timestamp).'.'.$extension;
17. $fileContent=file\_get\_contents($\_FILES["file"]["tmp\_name"]);
19. //Encryption.
20. include('../security/Crypt/AES.php');
22. $cipher = new Crypt\_AES(CRYPT\_AES\_MODE\_CFB);
23. $key='abcdefghijklmnopijklmnop'.MD5($admin.'xyz'.MD5($timestamp));
24. $cipher->setKey($key);
25. file\_put\_contents("../files/".$fileName,$cipher->encrypt($fileContent));
27. for ($i=0;$i<count($users);$i++){
28. $userStr.=addslashes($users[$i]);
29. if ($i<count($users)-1){
30. $userStr.="#";
31. }
32. }
33. include("../common/conn\_account.php");
34. MySQL\_query("insert into tb\_accesslist values
35. (NULL,'$fileName','$description','$admin','$userStr','$timestamp');");
36. MySQL\_close($conn);
38. //释放变量
39. unset($userStr); unset($users);
40. unset($timestamp); unset($fileName);
41. header("location: welcome.php");
42. }
43. ?>

用户下载文件时，进行解密downloadFile.php：

1. <?php
2. if (isset($\_POST['downloadFile'])){
3. // place this code inside a php file and call it f.e. "download.php"
4. $path = "../files/";
5. $files=$\_POST['files'];
6. //zipFilesAndDownload($files,"kkc.zip",$path);
7. foreach ($files as $file){
8. $temp=explode('#',$file);//$temp=$filename#$admin#$timestamp
9. $fullPath= $path.addslashes($temp[0]);
11. if ($fd = fopen ($fullPath, "r")) {
12. $fsize = filesize($fullPath);
13. $path\_parts = pathinfo($fullPath);
14. $ext = strtolower($path\_parts["extension"]);
15. switch ($ext) {
16. case "pdf":
17. header("Content-type: application/pdf"); // add here more headers for diff. extensions
18. header("Content-Disposition: attachment; filename=\"".$path\_parts["basename"]."\""); // use 'attachment' to force a download
19. break;
20. default;
21. header("Content-type: application/octet-stream");
22. header("Content-Disposition: filename=\"".$path\_parts["basename"]."\"");
23. }
24. header("Content-length: $fsize");
25. header("Cache-control: private"); //use this to open files directly
27. //Decryption.
28. include('../security/Crypt/AES.php');
30. $cipher = new Crypt\_AES(CRYPT\_AES\_MODE\_CFB);
31. $key='abcdefghijklmnopijklmnop'.MD5($temp[1].'xyz'.MD5($temp[2]));
32. $cipher->setKey($key);
33. // keys are null-padded to the closest valid size
34. // longer than the longest key and it's truncated
35. //$cipher->setKeyLength(128);
36. $buffer="";
37. while(!feof($fd)) {
38. $buffer.= fread($fd, 2048);
39. // echo $buffer;
40. }
41. echo $cipher->decrypt($buffer);
42. //echo $buffer;
43. }
44. fclose ($fd);
45. }
46. exit;
47. header("location: welcome.php");
48. }
49. ?>

## 5.4 功能实现及原理

本文实现了文件加密问题并实现了一个分布式数据库的可靠强大的文件加密系统。首先用户发请求给管理员，管理员按请求要求上传文档。所有上传的文档都是经过加密的。管理员上传好了文档，系统自动通知用户，用户就可以下载文档。



图5-4 文件管理

管理员可以上传文件并给一个特定的用户，所有上传的文件都是经过加密。

## 5.5 本章小结

笔者在本章中，重点解释了分布式数据库的核心技术，研究了该数据库的优点和现实价值。介绍了MySQL Replication复制集群和文件加密技术2种各有特点的机群技术，在结合实际业务场景的情况下的架构表现，并通过具体的架构实现过程来验证了其特有的性能表现。

最后，通过技术模拟和论证突破了构建完整分布式数据库必要的核心技术。

# 第六章 总结与展望

## 5.1研究总结

在当前这个时代，各种各样基于互联网的应用，SNS、网上购物、Blog、Weibo、在线游戏……已经完全融入了我们的日常生活。可以断言，如果现代人们的生活离开了网络，现代社会的生活和工作将难以为继。现代生活中人们依靠网络开展交流，基于网络而开展的各种活动也产生了超级巨量的信息，所以人们常说这是一个信息爆炸的时代，人们使用网络而产生的大量数据信息则需要存储在数据库中。

由于人类电子科技技术的不断进步，计算机的功能愈加丰富和强大，尤其是Intel CPU技术的更新换代，引导着PC电脑和服务的不断升级，性能也越来越可靠，互联网交往的争夺导致数据量的爆炸式增长，因而自然而然地产生了对于数据库的技术更新的要求，人们关注的目光也自然投向了MySQL。正是在这样的前提下，促使我选择了MySQL技术作为切入点来研究可以支撑现下符合用户使用需求的大规模高性能高可用性数据库集群技术。

并且目前确实有越来越多的商业公司甚至是顶级的互联网公司正在将业务数据从商业数据库系统往开源数据库系统上迁移，而其中大部分公司都和我一样不约而同得选择了迁往MySQL技术平台，其主要原因就是MySQL数据库是基于GPL开源协议的开源数据库软件，使用者可以非常透彻地了解到所有功能的实现方式；并且MySQL数据库本身所展现出来的优异的性能和卓越的稳定性也给了使用者们相当大的信心。

因此，本论文通过横向对比多种分布式数据库的核心技术和优缺点，对分布式数据库进行了研究分析和理论阐述，从而将研究重点确定在了MySQL Replicaion和文件加密这两种关键技术上；并结合实际应用场景进行了需求痛点分析从而最终确认了所需解决的主要技术问题：

* 1. 虽然msql有自带的同步技术，可是这个技术不够强。比如如果一个站点挂了，无法进行自动分配挂了的站点给他的副本数据库。虽然在本文实现了该功能还是需要提高，比如如果需要同步100GB的文件，是否可以。；
  2. 文件加密难题目前尽管可以利用传统的方式进行处理，不过用户下载文件时，不需要任何密码，这样如果一个黑客偷了用户的密码，就可以轻松的下载文件。

经过几个月时间的反复理论探讨和设计，最终实现了符合以上需求的集群架构体系。

## 5.2技术展望

现在，MySQL数据库的装机量接近1500万，占有全球数据库25%的市场份额，互联网公司80%的市场份额，特别近几年Web 2.0兴起，更是引爆了MySQL技术的发展，但目前来看其主要使用群体还是互联网公司，因为这是基于MySQL数据库系统自身之特点和市场客观需要。在这一市场群体方面，MySQL数据库肯定会对其他商业数据库的市场份额造成一定的影响；至于高端企业级应用方面，从MySQL自身的定位及市场策略来看，短期内应该不会对其他商业数据库造成太大的威胁，但再往后就不一定了。

并且随着Oracle公司收购SUN， Oracle和MySQL这两款业界最优秀的数据库产品归于一家公司所有，我们相信这一商业收购在技术上对MySQL技术的发展是有相当的积极作用的。

Oracle收购了MySQL后，很多人担心可能以后

技术上，对于大数据量高并发的Web应用来说，本论文所设计的这个集群架构体系中，在线DDL功能的增强确实是非常重要且紧急的需求；并且我们也发现其在充分利用多核并发能力方面如果能有更好的突破，将会让很多使用者节省大量的硬件成本，所以这也是一个急需提升的性能问题。除此之外，在线物理备份的功能也是十分重要的，同时让一个Slave节点从多个Master节点同时进行复制，以提高维护便利性对于数据库的使用和维护也至关重要。当然，我们也坚信，这些问题或者难题在将来的某个时间必能得到克服和突破。

相信在Oracle这样一家拥有大量顶级技术专家的公司的带领下，MySQL将会越来越优秀，越来越受到广大使用者的青睐。

# 参考文献

[1]. S. Hariri, M. Parashar - Tools and Environments for Parallel and Distributed Computing, John Wiley & Sons, 2004

[2]. W. Jia, W. Zhou - Distributed Network Systems. From Concepts to Implementations, Springer, 2005

[3]. A.S. Tanenbaum, M. van Steen - Distributed Systems. Principles and Paradigms, Prentice Hall, 2002.

[4]. Buretta, M. 1997. Data Replication: Tools and Techniques for Managing Distributed Information. New York: Wiley.

[5]. Thompson, C. 1997. “Database Replication: Comparing Three Leading DBMS Vendors’ Approaches to Replication.” DBMS 10,5 (May): 76–84.

[6]. 陈卫，数据库加密密钥的分配和管理技术[J]．清华大学学报：自

然科学版，1994，34(1)：99—102．

[7]. 藏一奇，尚杰，陈卫等一种新的数据库加密密钥管理方案[J]．清

华大学学报：自然科学版，1995，35(4)：43—47．

[8]. 崔国华，汤学明数据库中加密机制的实施研究[J]．密码与信息，

2O01(2)：84～9O

[9]. 戴一奇，尚杰，苏中民．密文数据库的快速检索[J]．清华大学学报：

自然科学版，l997，37(4)：24—27．

[10]. Tamassia Goodrich. Data Stuctures and Algorithms in JAVA. Wiley, 4th edition, 2005.

[11]. Horstmann. Big Java. Wiley, 4rd edition, 2007.

[12]. Ross Kurose. Computer Networking - A Top-Down Approach Featuring the Internet. Addison-Wesley, 4th edition, 2008.

[13]. Peeger. Software Engineering Theory and Practice. Prentice Hall, 3rd edition, 2006.

[14]. A. Adya, W. Bolosky,M. Castro, and G. Cermak et al. Farsite: Federated, available, and reliable storage for an incompletely trusted environment. In Proceedings of the fifth symposium on operating systems design and implementation, 2002.

[15]. Y. Amir, D. Dolev, P. Melliar-Smith, and L. Moser. Robust and efficient replication using group communication, 1994.

[16]. J. Welch H. Attiya. Distributed Computing. McGraw Hill International, Ltd., 1998.

[17].LRWIS Bil,BERG Daniel. Multithreaded programming with Pthreads

[18]. 张雄．分布式数据库数据同步的研究与应用【D】．华中科技大学，2008．

[19]. 张湘辉．分布式数据库数据复制技术研究与应用【D】．武汉大学，2004．

[20]. 蒋春曦．MS sQL seⅣer复制技术及实现【J】．电脑知识与技术，2004

18(02)：2l-22．

[21]. 吴旭，张静．SQL Server7．0的数据同步复制技术【J】．计算机时代，1999 ，33(04)：27．28．