笔墨酷系统搜索系统分析

|  |  |  |
| --- | --- | --- |
| **日期** | **版本** | **更新** |
| 2013 / 7 / 22 | V1.0 | 初稿 |
| 2013 / 7 / 29 | V2.0 | 添加详细分析 |
| 2013 / 8 / 1 | V 3.0 | 添加详细分析2 |
| 2013/8/12 | V 0.5 | 添加全文检索功能 |
|  |  |  |
|  |  |  |
|  |  |  |

**目录**

[1开发语言分析 4](#_Toc364090182)

[1.1数据库访问比较 4](#_Toc364090183)

[1.2系统设计架构比较 4](#_Toc364090184)

[1.3跨平台性 5](#_Toc364090185)

[1.4开发成本比较 5](#_Toc364090186)

[1.5分布式多层架构比较 5](#_Toc364090187)

[2系统架构分析 5](#_Toc364090188)

[2.1 java开发工具 6](#_Toc364090189)

[2.2 项目框架搭建 6](#_Toc364090190)

[3系统角色分析 8](#_Toc364090191)

[4系统功能分析 10](#_Toc364090192)

[4.1 图书类别分类 11](#_Toc364090193)

[5系统数据量分析 12](#_Toc364090194)

[5.1书籍分类 12](#_Toc364090195)

[5.2书籍搜索 13](#_Toc364090196)

[5.2.1 MySQL的replication功能 13](#_Toc364090197)

[5.2.2 MySQL的横向扩展和Partitioning 14](#_Toc364090198)

[5.2.3 全文搜索引擎 15](#_Toc364090199)

[5.2.4 Sphinx与MySQL结合实现全文搜索 15](#_Toc364090200)

[5.2.5 MySQL+Sphinx+SphinxSE安装 16](#_Toc364090201)

[5.2.6 安装Sphinx全文检索服务器 18](#_Toc364090202)

[5.2.7 安装问题 20](#_Toc364090203)

[6系统并发量分析 21](#_Toc364090204)

[6.1系统并发分析方法 21](#_Toc364090205)

[6.2 举例说明 22](#_Toc364090206)

[6.3并发量过大处理方法 23](#_Toc364090207)

[6.4 使用Nginx和Tomcat实现负载均衡 23](#_Toc364090208)

[6.4.1 Windows下安装 23](#_Toc364090209)

[6.4.2 linux下安装 23](#_Toc364090210)

[6.4.3 Nginx和Tomcat整合 24](#_Toc364090211)

[6.5 使用Apache和Tomcat实现负载均衡 27](#_Toc364090212)

[7系统容灾分析 27](#_Toc364090213)

[7.1数据容灾 28](#_Toc364090214)

[7.2应用容灾 28](#_Toc364090215)



# 1开发语言分析

PHP是解释执行的服务器脚本语言。Java首先编译成字节码.class文件，然后在java虚拟机上解释执行。并且有JVM管理资源。

## 1.1数据库访问比较

Java通过JDBC来访问数据库，通过不同的数据库厂商提供的数据库驱动方便的访问数据库。

PHP对于不同的数据库采用不同的数据库访问接口，所以数据库访问代码的通用性不强。例如从MySQL数据库转到Oracle数据库只需要做很少的修改。而PHP则需要做大量的修改工作。

## 1.2系统设计架构比较

采用java的web开发技术，需要使用的是面向对象的系统设计方法，而PHP以前的版本是面向过程的开发方式，PHP虽然现在是面向对象的但是还保留大量面向过程的特性。所以用java进行开发前期需要做大量的系统分析和设计工作。但是后期的维护就比PHP要简单些。

## 1.3跨平台性

Java和PHP都是有很好的跨平台的特性。几乎都可以在不作任何修改的情况下运行在Linux或者Windows等不同的操作系统上。

## 1.4开发成本比较

PHP最经典的组合就是：PHP+MySQL+Apache。非常适合开发中小型的Web应用，开发的速度比较快。而且所有的软件都是开源免费的，可以减少投入。

Java的web应用服务器有免费的Tomcat、Jboss等，如果需要更好的商业化的服务有：Web Sphere和weblogic。

## 1.5分布式多层架构比较

PHP只能实现简单的分布式两层或三层的架构，而java在这方面就比较强大，可以实现多层的网络架构。数据库层（持久层）、应用（业务）逻辑层、表示逻辑层彼此分开，而且现在不同的层都已经有一些成熟的开发框架的支持。可以方便开发者高效、合理、科学的开发多层的商业应用。

# 2系统架构分析

暂时我们现在做的是图书云平台中的图书检索的项目，后期需要和一些其它系统进行融合，采用WebService能够很好的支持系统的集成和降低耦合。

系统是图书云平台，前端展现使用JAVASCRIPT+HTML+CSS，由于访问量大采用AJAX进行数据交互。后台采用CXF+Spring+JDBC+MySQL搭建WebService的架构开发JSON的数据接口，暂时采用MySQL数据库，如果不能达到需求可能会使用其他的NoSQL数据库。



如果访问量巨大的服务的问题，就是如何应对一台服务器无法承担的负载。近10年来的趋势表明，有两种方式来处理“横向扩展”、“纵向扩展”。“ 横向扩展”的意思是，通过增加服务器数量来提高系统整体的处理能力并分担负载。“纵向扩展”是通过提高硬件性能来提高处理能力的方法。

应对大规模数据量：计算机要从硬盘上读出数据并存储到内存上，然后CPU会从内存上取得数据进行处理。而且，从内存上取得的指令会缓存到更高速度的CPU缓存中。所以，数据要进过硬盘🡪内存🡪缓存🡪CPU 多层才能被处理。

为了减少速度差，操作系统实用了各种各样的方式，从硬盘加载数据先放入内存中缓存，减少硬盘访问次数等。

自行构建基础设施的优点：

1. 硬件配置可以灵活调整；
2. 能够灵活应对服务的要求；
3. 可以控制瓶颈；

## 2.1 java开发工具

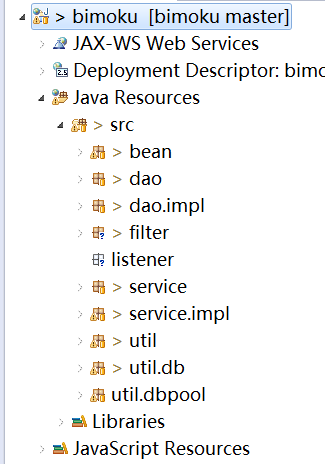
IDE：eclipse

JVM：jdk1.6

数据库：MySQL5.1

## 2.2 项目框架搭建

项目名称：bimoku



在src目录下：bean放数据库映射的实体Bean；dao包中放数据访问的接口；dao.impl中放数据访问的接口实现；filter包中放拦截器；listener包中放监听器；service放cxf开放服务的接口；service.impl中放cxf开放服务的接口实现；util中放一些工具类；util.db中放数据库访问工具类；util.dbpool中放数据库连接池实现。

1. 导入cxf和spring中需要的jar包；
2. 在web.xml中配置如下拦截器

|  |
| --- |
| <listener>  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  </listener>  <servlet>  <servlet-name>CXFServlet</servlet-name>  <servlet-class>  org.apache.cxf.transport.servlet.CXFServlet  </servlet-class>  </servlet>  <servlet-mapping>  <servlet-name>CXFServlet</servlet-name>  <url-pattern>/service/\*</url-pattern>  </servlet-mapping> |

1. 在WEB-INF中创建beans.xml文件

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:context=*"http://www.springframework.org/schema/context"*  xmlns:jaxws=*"http://cxf.apache.org/jaxws"*  xmlns:jaxrs=*"http://cxf.apache.org/jaxrs"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans-3.0.xsd*  *http://www.springframework.org/schema/context*  *http://www.springframework.org/schema/context/spring-context-3.0.xsd*  *http://cxf.apache.org/jaxws*  *http://cxf.apache.org/schemas/jaxws.xsd*  *http://cxf.apache.org/jaxrs*  *http://cxf.apache.org/schemas/jaxrs.xsd"*>  <import resource=*"classpath:META-INF/cxf/cxf.xml"* />  <import resource=*"classpath:META-INF/cxf/cxf-extension-soap.xml"* />  <import resource=*"classpath:META-INF/cxf/cxf-servlet.xml"* />  <bean id=*"restSample"* class=*"service.impl.HelloWodImpl"* />  <bean id=*"bookServiceImpl"* class=*"service.impl.BookImpl"*/>  <!-- 这里的地址很重要，客户端需要通过这个地址来访问WebService -->  <jaxrs:server id=*"restServiceContainer"* address=*"/rest"*>  <jaxrs:serviceBeans>  <ref bean=*"restSample"* />  <ref bean=*"bookServiceImpl"*/>  </jaxrs:serviceBeans>  <jaxrs:extensionMappings>  <entry key=*"json"* value=*"application/json"* />  <entry key=*"xml"* value=*"application/xml"* />  </jaxrs:extensionMappings>  <jaxrs:languageMappings>  <entry key=*"en"* value=*"en-gb"* />  </jaxrs:languageMappings>  </jaxrs:server>  </beans> |

# 3系统角色分析

图书云平台的图书检索项目中实体对象可以分为三类

包括系统管理员、用户、书籍、作者、出版社、平台游客





图书相关属性：

|  |  |  |  |
| --- | --- | --- | --- |
| 书名 |  |  |  |
| 作者 |  |  |  |
| 译者（~） |  |  |  |
| 出版社 |  |  |  |
| 版本 |  |  |  |
| isbn |  |  |  |
| 图书概述 |  | 内容介绍 | 按不同站点罗列 |
|  | 丛书介绍 |
|  | 编辑推荐 |
|  | 媒体推荐 |
|  | 作者介绍 |
|  | 译者介绍 |
| 书评 |  | 评论人 | 转发/收藏/评论（？） |
|  | 内容 |
|  | 时间 |
|  | 来自站点 |
|  | 评论人头像 |
| 笔记 |  | 评论人 | 转发/收藏/评论（？） |
|  | 内容 |
|  | 时间 |
|  | 来自站点 |
|  | 评论人头像 |
| 图书微博 |  | 评论人 | 转发/收藏/评论（？） |
|  | 内容 |
|  | 时间 |
|  | 来自站点 |
|  | 评论人头像 |
| 图书推荐 |  | 买过此书的人还买过 |  |
|  | 喜欢此书的人还买过 |  |
|  | 收藏此书的人还买过 |  |
|  | 搜索过此书的人还买过 |  |
|  | 本类五星图书 |  |
|  | 本类热销图书 |  |
| 想读人数 |  |  |  |
| 已读人数 |  |  |  |
| 在读人数 |  |  |  |
| 星级 |  |  |  |
| 目录 |  |  |  |
| 价格 |  |  |  |
| 封面图片 |  |  |  |

# 4系统功能分析

现在以用户使用系统为例分析系统的功能，用户在登录进入系统后首先能够看到的是所有的图书的列表，能够导航系统的图书导航包括“全部“、”新书“、”特价“、”销售排行榜“、”搜索排行榜“、“兴趣“、”社交网络“，选择类型后可以选择书籍的类型，类型包括“全部”、“小说”、”外语”、”艺术“、”考试“、”医学“、”文学“等。



搜索将会列出书籍的图篇和一些评论，同时能够在图书列表中对任何的图书进行评价，并能够点击查看详情、加为关注、喜欢、或者是比价。

加为关注后这本书将会在你受关注的书籍中出现，选择比价，将会列出不同网站该图书的价格。

当选择特价搜索出了全部搜索图书中的全部功能外还具有选择”3折以下“、”3-5折“、”5-7折“、”5元以下“、”5-10元“、”15-20元“、”15元以上“等选择，同时可以按照默认排序、评定级别排序、点评数排序、价格排序等。

当选择销售排行榜出现可以选择”近24小时畅销榜“、”近7天畅销榜“、”近30天畅销榜“等。还可以分类查看”图书畅销榜“、”新书畅销榜“、”特价书畅销榜“、”五星图书畅销榜“，显示图的每本书可以展示出该图书，现在有多少人正在读该图书，多少人读过、多少人想读、多少人关注该图书等，同时显示图书的图片、书名、作者、出版社、和本站的书价、亚马逊、京东、当当网的书价以及用户评定的平均等级。

图书概述展示图书的图片、图书的名称、图书评定级别、多少人评级、图书的简介、图书作者、图书译者、图书出版社、版本、喜欢、加入书架、想读、在读、已读、写书评、分享、加关注。显示本站架构和其他各站架构的对比。

同时需要显示图书概述，概述分为图书的介绍、当当网等各站的介绍、作者介绍、译者介绍、丛书介绍、编辑推荐和媒体推荐。展示查看该书的人搜索过其它那些书，以及自己搜索过的书籍。

## 4.1 图书类别分类

|  |  |  |
| --- | --- | --- |
|  | 小说 | 世界名著、作品集、四大名著、中国古典小说、中国当代小说、魔幻、外国小说、历史、影视小说、社会、侦探/悬疑/推理、中国近现代小说、武侠、情感、港澳台小说、都市、财经、惊悚/恐怖、科幻、英文原版书-小说、官场、军事、职场、乡土 |
|  | 文学 | 中国现当代随笔、文集、文学理论、中国古诗词、外国随笔、纪实文学、中国现当代诗歌、外国诗歌、戏剧、民间文学、中国古代随笔、英文原版书-文学、文学类考试 |
|  | 管理 | 一般管理学、市场/营销、金融/投资、创业企业与企业家、会计、电子商务、管理信息系统、生产与运作管理、商务沟通、战略管理、商业史传、MBA、管理音像、WTO、工具书、外文原版/影印版、英文原版书-管理、管理类职称考试 |
|  | 艺术 | 绘画、音乐、摄影、书法/篆刻、收藏/鉴赏、设计、小人书/连环画、影视/媒体艺术、艺术理论、世界各国艺术概况、戏剧艺术/舞台艺、人体艺术、工艺美术、建筑艺术、雕塑、民间艺术、舞蹈、艺术类考试、英文原版书-艺术 |
|  | 历史 | 历史普及读物、中国史、世界史、史家名著、文物考古、史料典籍、历史随笔、史学理论、地方史志、历史地理、名族史、专业史、…… |
|  |  |  |
|  |  |  |
|  |  |  |

# 5系统数据量分析

该系统图书云平台是大规模的系统，Google、Yahoo！所有的数据规模是达到TB、PB级别，这些是超大规模，全世界图书在10年统计1亿2千9百多万册，可见数据量还是很巨大的，但是当当网图书共有200多万，亚马逊700多万，3000多万从网上爬取到的数据就是4000万左右，还有一些书籍的评论相关等，书籍数据量在千万级别以上，评论的数据量更是比书籍多的多。光做图书这一块最大也就是GB级别也就是算个大规模系统。

当使用的数据量大于内存时，为了减少I/O交互数据，可以使用置换算法对项目进行优化，常用的置换算法：

（1）、最佳置换算法（OPT）（理想置换算法）：所选择的被淘汰页面将是以后永不使用的，或者是在最长时间内不再被访问的页面，这样可以保证获得最低的缺页率。

（2）、先进先出置换算法（FIFO）：优先淘汰最早进入的页面，亦即在内存中驻留时间最久的页面。

（3）、最近最久未使用（LRU）算法：选择最近最长时间未访问过的页面予以淘汰。

（4）、Clock置换算法（LRU算法的近似实现）：给每一帧关联一个附加位，称为使用位。

（5）、最少使用（LFU）置换算法。

（6）、工作集算法。

（7）、工作集时钟算法。

（8）、老化算法（非常类似LRU的有效算法）。

（9）、NRU（最近未使用）算法

（10）、第二次机会算法。

选取一种作为搜索对象库的置换算法，将数据库当成物理内存将JVM内存当成内存，减少与数据库的交互和批量操作数据库。

Google和PageRank算法大数据量检索：

Google的体系结构类似于传统的搜索引擎，它与传统的搜索引擎最大的不同处在于对网页进行了基于权威值的排序处理，使最重要的网页出现在结果的最前面。Google通过PageRank元算法计算出网页的PageRank值，从而决定网页在结果集中的出现位置，PageRank值越高的网页，在结果中出现的位置越前。

如果数据量过大那就要注意数据库的设计了，假设有3亿条的记录的话，假设给每一个记录增加大小为8字节的一列，那么数据量就会增加8\*3亿个字节。也就是3GB。表结构的设计要注意字段要设计的紧凑一些。

## 5.1书籍分类

书籍可以分类成“科学\*学问”、“电脑\*IT”两类，其他可以分为“政治”、“经济”等。一般的在使用爬虫抓取出来就是按照分类抓取出来这样是非常好的。如果有些书不明确类型，也可以人工判断类型，但是这样做是很费时间的。判断类别可以使用“贝叶斯过滤器判断类别”。贝叶斯过滤器接收文本作为输入，并使用朴素贝叶斯算法从概率上判断文章属于哪个类别。其特点是判断未知文章时，要利用以前已经分类的数据统计信息进行判断。

## 5.2书籍搜索

搜索主要处理3大问题：

（1）、需要搜索大量的数据。

（2）、需要高速搜索。

（3）、将”好“文章放在最前面。

其中第3个将”好”文章放在最前面，实际上是最困难的，为此需要进行“评分”处理。评分需要综合利用搜索文章所含的各种各样的信息。对于只能根据特定列进行排序，RDBMS来说，这种处理时不可能的。相反，自行创建查找索引，也就是说自己实现全文搜索引擎，就能自由选择评分算法等，搜索结果的排列方式也要比使用RDBMS灵活地多。图书云平台应该是需要自己创建索引列。

### 5.2.1 MySQL的replication功能

MySQL的基本功能中包括replication（复制）功能。所谓replication，就是确定master以及与之同步的slave服务器，再加上slave将master中写入的内容polling过来更新自身内容的功能。这样slave就是master的replica（复制品）。这样就可以准备多台内容相同的服务器。

通过master和salve的replication，准备好多台服务器之后，让应用程序服务器通过负载均衡器去处理查询slave。这样就能将查询分散到多台服务器上。

应用程序实现上应该只把select等读取之类的查询发送给负载均衡器，而更新应当直接发送给master。要是在slave上执行更新操作，slave和master的内容就无法同步。MySQL会检测到master和slave之间内容差异，并停止replication，这回导致系统故障。Slave可以采用LVS（linux系统自带的负载均衡器）实现查询的负载均衡。



使用MySQL的replication是利用的冗余化，实现冗余化需要实现的最小服务器数量是4台，三台slave和一台master，slave为什么是需要三台呢，比如一台slave死机了，现在需要修复再次上线，那么意味着你必须停止一台slave来复制MySQL的数据，如果只有两台slave，一台坏了，你就必须停止服务，如果有三台，坏了一台，你复制数据时停止一台，还有一台可以运维。

### 5.2.2 MySQL的横向扩展和Partitioning

对于数据的处理是能放入到内存中就尽量放入到内存中如果不能放入到内存中，可以利用MySQL的Partitioning。

Partitioning就是表分割也就是讲A表和B表放在不同的服务器上。简单来说，Partitioning就是充分利用局部性进行分割，提高缓存利用效率，从而实现Partitioning的效果。其中最重要的一点就是以Partitioning为前提设计的系统将表分割开，用RDBMS的方式的话，对于一对多的关系经常使用JOIN查询将两张表连接起来。但是如果将表分割开了之后，也就是两张表不在同一个数据库，不在同一个服务器上怎样使用JOIN操作，这里需要注意的是如果是用where in操作不是省了一些麻烦了嘛。



### 5.2.3 全文搜索引擎

从全文搜索的架构上看有很多种。常见的有“grep类型”、“后缀类型”、“逆向索引类型”。

属于grep类型的，包括grep和Shunsaku。PFI的Sedue属于后缀类型<del>为PFI的Sedue</del>为PFI的，一些博客平台用的就是这种。逆向索引类型，以Google为代表的现代主流的搜索引擎基本上都采用这种架构。逆向索引由Dictionary（字典文件）和Postings（置入文件）两个基本要素组成。

创建Dictionary时有个问题就是怎样选择term。一个问题就是怎样处理语言中的单词。方法很多，可以使用事先确定的字典，可以用Aho-Corasick算法对单词进行切分，也可以使用语素分析。另一个问题是不进行分词，而是用n-gram方法将文字按照适当的单位切分，当做term处理，需要考虑到对象过大或者对象过小的问题。

Postings的创建方法，创建Postings就是某个文档包含单词的文档编号，也就是文档ID，但是有些时候还要保持term在文档中出现的位置，这种情况称为“Full Inverted Index”。保存出现的位置，就可以显示snippet，从而立即获知单词包含在文档的位置。

### 5.2.4 Sphinx与MySQL结合实现全文搜索

Sphinx是一个基于SQL的全文检索引擎，可以结合MySQL，PostgreSQL做全文搜索，他可以提供比数据库本身更专业的搜索功能，使得应用程序更容易实现专业化的全文检索。 Sphinx特别为一些脚本语言设计搜索API接口，如：PHP、Python、Perl、Ruby等，同时为MySQL也设计了一个存储引擎插件。

Sphinx单一索引最大可包含1亿条记录，在1千万条记录情况下的查询速度为毫秒级。Sphinx创建索引的速度为：创建100万条记录的索引只需 3～4分钟，创建1000万条记录的索引可以在50分钟内完成，而只包含最新10万条记录的增量索引，重建一次只需几十秒。

Sphinx的主要特性包括：

高速索引 (在新款CPU上,近10 MB/秒);  
高速搜索 (2-4G的文本量中平均查询速度不到0.1秒);  
高可用性 (单CPU上最大可支持100 GB的文本,100M文档);  
提供良好的相关性排名

支持分布式搜索;

提供文档摘要生成;

提供从MySQL内部的插件式存储引擎上搜索

支持布尔,短语, 和近义词查询;

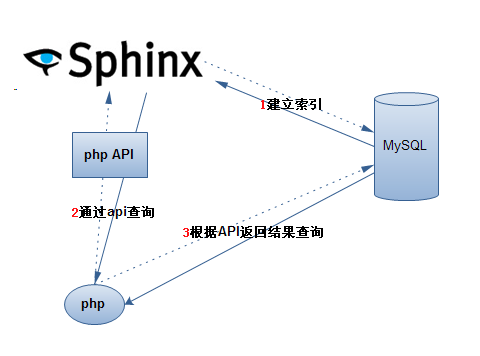
支持每个文档多个全文检索域(默认最大32个);

支持每个文档多属性;  
支持断词;

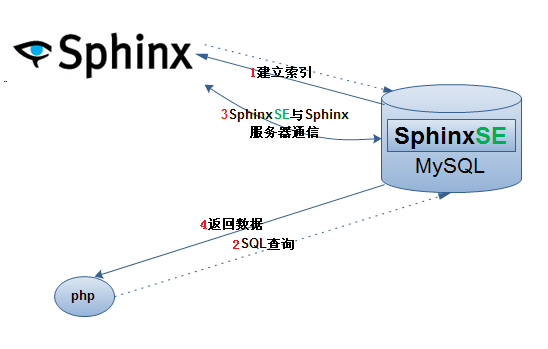
支持单字节编码与UTF-8编码;

Sphinx网站 ：<http://sphinxsearch.com/docs/2.1.1/about.html> 支持数据库MySQL。

原生MySQl存储引擎检索流程：



基于Sphinx存储引擎检索：



### 5.2.5 MySQL+Sphinx+SphinxSE安装

在开始安装之前需要安装一些必要的组件

|  |
| --- |
| yum -y install gcc g++ gcc-c++ libjpeg libjpeg-devel libpng libpng-devel freetype freetype-devel libxml2 libxml2-devel zlib zlib-devel glibc glibc-devel glib2 glib2-devel bzip2 bzip2-devel ncurses ncurses-devel curl curl-devel e2fsprogs e2fsprogs-devel krb5 krb5-devel libidn libidn-devel openssl openssl-devel openldap openldap-devel nss\_ldap openldap-clients openldap-servers patch libtool automake imake mysql-devel expat-devel |

（1）安装python支持

|  |
| --- |
| yum install –y python python-devel |

（2）编译安装LibMMSeg（LibMMSeg是为Sphinx全文搜索引擎设计的中文分词软件包，其在GPL协议下发行的中文分词法，采用Chin-Hao Tsai的MMSEG算法。LibMMSeg在本文中用来生成中文分词词库）。

|  |
| --- |
| wget <http://www.coreseek.com/uploads/sources/mmseg-0.7.3.tar.gz>  tar zxvf mmseg-0.7.3.tar.gz  cd mmseg-0.7.3  ./configure  make  make install |

1. 编译安装MySQL5.1.26-rc、Sphinx、SphinxSE存储引擎

|  |
| --- |
| wget <http://blog.s135.com/soft/linux/nginx_php/mysql/mysql-5.1.26-rc.tar.gz>  tar zxvf mysql-5.1.26-rc.tar.gz  wget <http://www.sphinxsearch.com/downloads/sphinx-0.9.8-rc2.tar.gz>  wget <http://www.coreseek.com/uploads/sources/sphinx-0.98rc2.zhcn-support.patch>  wget <http://www.coreseek.com/uploads/sources/fix-crash-in-excerpts.patch>  tar zxvf sphinx-0.9.8.rc2.tar.gz  patch –p1 < ../sphinx-0.98rc2.zhcn-support.patch #补丁  patch –p1 <../fix-crash-in-excerpts.patch #补丁  cp –rf mysqlse ../mysql-5.1.26-rc/storage/sphinx  cd ../  cd mysql-5.1.26-rc/  sh BUILD/autorun.sh  ./configure --with-plugins= partition,innobase,myisammrg,sphinx --prefix=/usr/local/mysql / --enable-assembler --with-extra-charsets=complex --enable-thread-safe-client --with-big-tables --with-readline --with-ssl --with-embedded-server --enable-local-infile  make && make install  cd ../ |

启动MySQL数据库

|  |
| --- |
| cp support-files/my-medium.cnf /etc/my.cnf # 配置文件  cp support-files/mysql.server /etc/rc.d/mysqld # 添加 MySQL 服务控制  cd /usr/local/mysql  bin/mysql\_install\_db --user=mysql # 安装  bin/mysqld\_safe --user=mysql & # 测试安装是否成功  bin/mysql # 进入 MySQL 命令提示符  启动停止  /etc/rc.d/mysqld start  /etc/rc.d/mysqld stop  于是我们自己创建文件/etc/rc.local并给予执行权限。大致内容为：  #!/bin/sh  /usr/local/mysql/bin/mysqld\_safe --user=mysql &  或者  /etc/rc.d/mysqld start |

输入下列命令出现SPHINX表示SphinxSE已经移植到MySQL中去了。

|  |
| --- |
| show engines; |

### 5.2.6 安装Sphinx全文检索服务器

Sphinx默认不支持中文索引及检索，以前用Coreseek的补丁来解决，目前Coreseek不单独提供补丁，而基于Sphinx开发了Coreseek全文检索服务器，Coreseek应该是现在用的最多的Sphinx中文全文检索，它提供了为Sphinx设计的中文分词包LibMMSeg包含mmseg中文分词，其实coreseek-3.2.14.tar.gz中已经包含了sphinx，前面安装SphinxSE时也可以使用这个压缩包里的mysqlse。

安装autoconf

|  |
| --- |
| tar zxvf autoconf-2.64.tar.gz  cd autoconf-2.64  ./configure –prefix=/usr  make  make install |

安装Coreseek

|  |
| --- |
| tar zxvf coreseek-3.2.14.tar.gz  cd coreseek-3.2.14  cd mmseg-3.2.14/  ./bootstrap  ./configure –prefix=/usr/local/mmseg3  make  make install  cd ../csft-3.2.14/  sh buildconf.sh  ./configure --prefix=/usr/local/coreseek --without-python --without-unixodbc --with-mmseg --with-mmseg-includes=/usr/local/mmseg3/include/mmseg/ --with-mmseg-libs=/usr/local/mmseg3/lib/ --with-mysql --host=arm  make  make install  cd /usr/local/coreseek/etc |

进入配置目录通过命令ls可以看到3个文件

example.sql sphinx.conf.dist sphinx-min.conf.dist

其中example.sql是实例sql脚本我们将其导入到数据库中的test数据库中作为测试数据（会创建documents表和tags表）导入到MySQL中时注意创建表使用的编码使用utf8要不然不能对中文支持。

vi sphinx.conf

输入一些内容：

|  |
| --- |
| source src1  {  type = mysql  sql\_host = localhost  sql\_user = root  sql\_pass =12345678  sql\_db = test  sql\_port = 3306 # optional, default is 3306  sql\_sock = /tmp/mysql.sock  sql\_query\_pre = SET NAMES utf8  sql\_query = \  SELECT id, group\_id, UNIX\_TIMESTAMP(date\_added) AS date\_added, title, content \  FROM documents  sql\_attr\_uint = group\_id  sql\_attr\_timestamp = date\_added  sql\_query\_info = SELECT \* FROM documents WHERE id=$id  }  index test1  {  source = src1  path = /usr/local/coreseek/var/data/test1  docinfo = extern  charset\_type = zh\_cn.utf-8  mlock = 0  morphology = none  min\_word\_len = 1  html\_strip = 0  charset\_dictpath = /usr/local/mmseg3/etc/  ngram\_len = 0  }  indexer  {  mem\_limit = 32M  }  searchd  {  port = 9312  log = /usr/local/coreseek/var/log/searchd.log  query\_log = /usr/local/coreseek/var/log/query.log  read\_timeout = 5  max\_children = 30  pid\_file = /usr/local/coreseek/var/log/searchd.pid  max\_matches = 1000  seamless\_rotate = 1  preopen\_indexes = 0  unlink\_old = 1  } |

说明：代码段sorce src1{\*\*\*}代表数据源里面主要包含了数据库的配置信息，src1表示数据源名字，可以随便写。

代码段index test1{\*\*\*} 代表为那个数据源创建索引，与source \*\*\*是成对出现的，其中的source参数的值必须是某个数据源的名字。

创建索引表

|  |
| --- |
| CREATE TABLE t1  (  id INTEGER UNSIGNED NOT NULL,  weight INTEGER NOT NULL,  query VARCHAR(3072) NOT NULL,  group\_id INTEGER,  INDEX(query)  ) ENGINE=SPHINX CONNECTION="sphinx://localhost:9312/test1"; |

生成索引

|  |
| --- |
| /usr/local/coreseek/bin/indexer -c /usr/local/coreseek/etc/sphinx.conf --all |

启动服务

|  |
| --- |
| /usr/local/coreseek/bin/searchd -c /usr/local/coreseek/etc/sphinx.conf |

**注意**：1.每次插入数据的时候都需要重新生成分词库。

2.外网不能访问需要设置访问权限

|  |
| --- |
| grant select,update,insert,delete on \*.\* to root@192.168.1.12 identified by "root";  flush privileges; |

3.MySQL中root用户密码修改

|  |
| --- |
| update user set password=PASSWORD('123456') where user='root';  flush privileges; |

### 5.2.7 安装问题

问题1：如果sh BUILD/autorun.sh

但sphinx就是不会出现在configure –h里面，需要执行sh BUILD/cleanup 再执行sh BUILD/autorun.sh 然后执行./configure –h现在就可以看到sphinx了。

问题2：如果编译mysql是报错查看是否安装ncurses安装包

可以执行：yum list|grep ncurses

yum –y install ncurses-devel

yum install ncurses-devel

然后再执行./configure。

问题3 在安装LibMMSeg需要先执行yum install mysql-devel libxml2-devel expat-devel

问题4 在安装MMSeg的时候出现错误提示为：css/UnigramCorpusReader.cpp:89: error: 'strncmp' was not declared in this scope

手动修改了src/css/UnigramCorpusReader.cpp

在上面添加了一句

#include <string.h>

然后再开始编译安装就可以了。

问题4 编译mmseg提示cannot find input file:src/Makefile.in

|  |
| --- |
| aclocal libtoolize --force 运行后有一个错误，不用管它。 automake --add-missing autoconf autoheader make clean ./configure --prefix=/usr/local/mmseg3 通过了 make make install 安装完成后，mmseg使用的词典和配置文件，自动安装在/usr/local/mmseg3/etc中 |

# 6系统并发量分析

该项目并没有实际的运维过，暂时预定是2000的并发量处理，实际的运维压力还要等待第一个版本上线后实际运维确定用户量的大小。

## 6.1系统并发分析方法

与并发用户数的概念还包括“并发用户数”、“系统用户数”和“同时在线用户数”，下面用一个实际的例子来说明它们之间的差别。

假设有一个系统，该系统有2000个使用用户。这就是说，可能使用该系统的用户总数是2000名，这个概念就是“系统用户数”，该系统有一个“在线统计”功能（系统用一个全局变量计数所有登录的用户），从在线统计功能中得到，最高峰时有500人在线（这个500就是一般所说的“同时在线人数”），那么，系统的并发用户数是多少呢？

根据我们队业务并发用户的定义，这500就是整个系统使用时最大的业务并发数。当然，500这个数值只是表面在最高峰时刻有500个用户登录了系统，并不表示实际服务器承受的压力，因为服务器承受的压力还与具体用户访问模式相关。例如，在500个“同时使用系统”的用户中，考察某一个时间点，在这个时间上，假设其中40%的用户在较有兴致地看系统公告（注意：“看”这个动作时不会对服务端产生任何负担的），20%的用户在填写复杂的表格（对用户填写的表格来说，只有在“提交”的时刻才会向服务器发送请求，填写过程是不对服务端构成压力的），20%部分用户在发呆（也就是什么也没做），剩下的20%用户在不停地从一个页跳转到另一个页面—在这种场景下，可以说，只有20%用户真正对服务器构成了压力，因此，从上面的例子中可以看出，服务器承受的压力不只取决于业务并发用户数，还取决用户的业务场景。

在实际的性能测试工作中，测试人员一般比较关心的是业务并发用户数，也就是从业务角度关注究竟应该设置多少个并发数比较合理，因此，在后面的讨论中，也是主要针对业务并发用户数进行讨论，而且，为了方便，直接将业务并发用户数称为并发用户数

1. 计算平均的并发用户数：C=nL/T
2. 并发用户数峰值：C‘≈C+3根号C

公式（1）中，C是平均的并发用户数；n是login session的数量；L是login session的平均长度；T指考察的时间段长度。

公式（2）则给出了并发用户数峰值的计算方式中，其中,C’指并发用户数的峰值，C就是公式（1）中得到的平均的并发用户数。该公式的得出是假设用户的login session产生符合泊松分布而估算得到的。

## 6.2 举例说明

         假设有一个系统，该系统有3000个用户，平均每天大约有400个用户要访问该系统，对一个典型用户来说，一天之内用户从登录到退出该系统的平均时间为4小时，在一天的时间内，用户只在8小时内使用该系统。  
则根据公式（1）和公式（2），可以得到：  
               C = 400\*4/8 = 200

               C’≈200+3\*根号200 = 242

            F=VU \* R / T

其中F为吞吐量，VU表示虚拟用户个数，R表示每个虚拟用户发出的请求数，T表示性能测试所用的时间

R = T / TS

TS为用户思考时间

计算思考时间的一般步骤：

A、 首先计算出系统的并发用户数

C=nL / T      F=R×C  
B、 统计出系统平均的吞吐量  
F=VU \* R / T R×C = VU \* R / T  
C、 统计出平均每个用户发出的请求数量  
R=u\*C\*T/VU  
D、根据公式计算出思考时间  
TS=T/R  
缺陷检测有效性百分比DDE=TDFT/(TDFC+TDFT)×100%  
其中:TDFT=测试过程中发现的全部缺陷(即由测试组发现的),TDFC=客户发现的全部缺陷(在版本交付后一个标准点开始测量,如,半年以后)

缺陷排除有效性百分比DRE=(TDCT/TDFT)×100%  
其中:TDCT=测试中改正的全部缺陷,TDFT=测试过程中发现的全部缺陷

测试用例设计效率百分比TDE=(TDFT/NTC)×100%  
其中:TDFT=测试过程中发现的全部缺陷,NTC=运行的测试用例数

以下公式较适用于白盒测试  
功能覆盖率= 至少被执行一次的测试功能点数/ 测试功能点总数 （功能点）  
需求覆盖率= 被验证到的需求数量 /总的需求数量 （需求）  
覆盖率= 至少被执行一次的测试用例数/ 应执行的测试用例总数 （测试用例）  
语句覆盖率= 至少被执行一次的语句数量/ 有效的程序代码行数  
判定覆盖率= 判定结果被评价的次数 / 判定结果总数  
条件覆盖率= 条件操作数值至少被评价一次的数量 / 条件操作数值的总数  
判定条件覆盖率= 条件操作数值或判定结果至少被评价一次的数量/(条件操作数值总数+判定结果总数)  
上下文判定覆盖率= 上下文内已执行的判定分支数和/(上下文数\*上下文内的判定分支总数)  
基于状态的上下文入口覆盖率= 累加每个状态内执行到的方法数/(状态数\*类内方法总数)  
分支条件组合覆盖率= 被评测到的分支条件组合数/分支条件组合数  
路径覆盖率= 至少被执行一次的路径数/程序总路径数

## 6.3并发量过大处理方法

如果系统的并发量过大可以采用两种方法处理

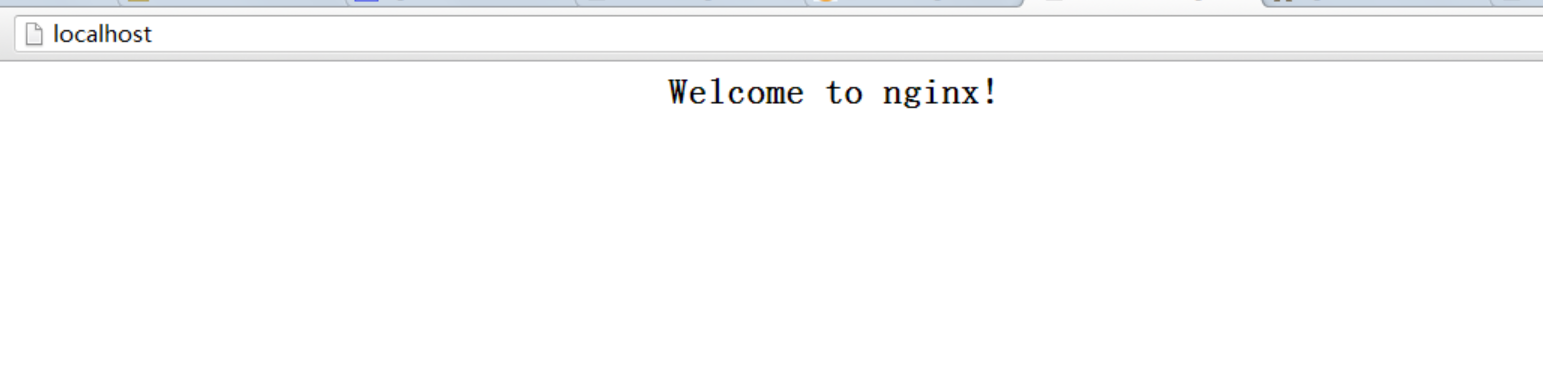
## 6.4 使用Nginx和Tomcat实现负载均衡

Nginx 是一款高性能的，轻量级的HTTP Web服务器和反向代理服务器及电子邮件IMAP/POP3/SMTP代理服务器。

### 6.4.1 Windows下安装

（1）首先去官网下载nginx的windows版本，官网下载地址：<http://nginx.org/download/ngix-1.0.11.zip>

（2）解压包到根目录下，然后cd到目录下运行 start nginx就可以启动。在浏览器中输入：<http://localhost> 就可以看到nginx的欢迎界面了，如果要停止nginx服务器可以输入：nginx –s stop 停止nginx；输入：nginx –s reload 重新加载配置文件；输入：nginx –s quit 退出nginx.



（3）配置nginx配置文件。

### 6.4.2 linux下安装

### 6.4.3 Nginx和Tomcat整合

使用nginx负载均衡的话，可以修改配置http节点如下：

|  |
| --- |
| #设定http服务器，利用它的反向代理功能提供负载均衡支持  http{  #设定mime类型，类型由mime.type文件定义  Include /etc/nginx/mime.types;  default\_type application/octet-stream;  设定日志格式  access\_log /var/log/nginx/access.log;  #  #设定负载均衡的服务器列表  Upstream mysvr{  #weight参数表示权值，权值越高被分配到的几率越大  Server 192.168.8.1x:3128 weight=5;  #本机上的Squid开启3128端口，不是必须要squid  server 192.168.8.2x:80 weight=1;  server 192.168.8.3x:80 weight=6;  }  Upstream mysvr2{  #weight参数表示权值，权值越高被分配到的几率越大  Server 192.168.8.x:80 weight=1；  Server 192.168.8.x:80 weight=6;  }  #第一个虚拟服务器  Server{  #侦听192.168.8.x的80端口  listen 80;  server\_name 192.168.8.x;  #对aspx后缀的进行负载均衡请求  Location ~.\*\.aspx${  #定义服务器的默认网站根目录位置  root /root;  #定义首页索引文件的名称  index index.php index.html index.htm  #请求转向mysvr定义的服务器列表  Proxy\_pass <http://mysvr>;  #以下是一些反向代理的配置可删除。  proxy\_redirect off;  #后端的Web服务器可以通过X-Forwarded-For获取用户真实IP  proxy\_set\_header Host $host;  proxy\_set\_header X-Real-IP $remote\_addr;  proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  #允许客户端请求的最大单文件字节数  client\_max\_body\_size 10m;  #缓冲区代理缓冲用户端请求的最大字节数，  client\_body\_buffer\_size 128k;  #nginx跟后端服务器连接超时时间(代理连接超时)  proxy\_connect\_timeout 90;  #连接成功后，后端服务器响应时间(代理接收超时)  proxy\_read\_timeout 90;  #设置代理服务器（nginx）保存用户头信息的缓冲区大小  proxy\_buffer\_size 4k;  #proxy\_buffers缓冲区，网页平均在32k以下的话，这样设置  proxy\_buffers 4 32k;  #高负荷下缓冲大小（proxy\_buffers\*2）  proxy\_busy\_buffers\_size 64k;  #设定缓存文件夹大小，大于这个值，将从upstream服务器传  proxy\_temp\_file\_write\_size 64k;  }  }  } |

Nginx和Tomcat负载均衡实现session共享，为了实现多个session共享，使用开源的Memcached-Session-Manager 框架，框架网址：<https://code.google.com/p/memcached-session-manager/>。

此框架的优势：

1. 支持Tomcat6和Tomcat7；
2. 操作粘性或不粘性Session；
3. 没有单点故障；
4. Tomcat的故障转移；
5. Memcached的故障转移；
6. 可插拔的Session序列化；
7. 支持异步保存Session；
8. Session发送到Memcached如果他确实修改了
9. JMX管理和监控；

以下操作步骤的前提是你已经安装了，Tomcat和Memcached，和已经配置了Nginx的负载，如果这些你都没做，请先安装和配置。

下载jar包：memcached-session-manager-1.5.1.jar,memcached-session-manager-tc6-1.5.1.jar。jar包的下载地址：<http://code.google.com/p/memcached-session-manager/downloads/list>

把下载的jar包放在%TOMCAT\_HOMT%\lib中。配置配置%TOMCAT\_HOME%\config\context.xml或者是在server.xml配置文件中的<host></host>中添加配置，详细配置参看官方文档。

启动Tomcat查看session是否共享成功。

说明： 1、 session存储到memchached实现方案时。他主要功能是修改tomcat的session存储机制，使之能够把session序列化存放到memcached中。

       2、Manager标签属性说明：

             className

                    此属性是必须的。

             memcachedNodes

                        此属性是必须的。这个属性必须包含你所有运行的memcached节点。每个节点的定义格式为<id>:<host>:<port>。

                   多个之间用空格或半角逗号隔开(如：memcachedNodes="n1:localhost:11211,n2:localhost:11212")。

                       如果你设置单个memcache节点<id>是可选的，所以它允许设置为<host>:<port>（memcachedNodes="localhost:11211"）。

             failoverNodes

                      可选项，属性只能用在非粘连Session机制中。

                      此属性必须包含memcached节点的Id，此节点是Tomcat作为备份使用。多个之间用空格或逗号隔开

              memcachedProtocol

                   可选项，默认为text。出属性指明memcached使用的存储协议。只支持text或者binary。

              sticky 可选项，默认为true。

                    指定使用粘性的还是非粘性的Session机制。

              lockingMode 可选项， 此属性只对非粘性Session有用，默认为none。

                     指定非粘性Session的锁定策略。他的只有

                        (1)、none:从来不加锁

                        (2)、all: 当请求时对Session锁定，直到请求结束

                        (3)、auto:对只读的request不加锁，对非只读的request加锁

                      (4)、uriPattern:<regexp>: 使用正则表达式来比较requestRUI + "?" + queryString来决定是否加锁，

             requestUriIgnorePattern  可选项

                        此属性是那些不能改备份Session的请求的正则表达式。如果像css,javascript,图片等静态文件被同一个Tomcat和同一个应用上下文来提供，这些

                   请求也会通过memcached-session-manager。但是这些请求在一个http会话中几乎没什么改变，所以他们没必要触发Session备份。所以那些静态文件

                   没必要触发Session备份，你就可以使用此属性定义。此属性必须符合java regex正则规范。

            sessionBackupAsync 可选项，默认true

                        指定Session是否应该被异步保存到Memcached中。 如果被设置为true，backupThreadCount设置起作用，如果设置false，通过sessionBackupTimeout

                   设置的过期时间起作用。

            backupThreadCount 可选项，默认为CPU内核数。

                       用来异步保存Session的线程数(如果sessionBackupAsync="true")。

            sessionBackupTimeout  可选项，默认100，单位毫秒

                       设置备份一个Session所用的时间，如果操作超过时间那么保存失败。此属性只在sessionBackupAsync="false"是起作用。默认100毫秒

            sessionAttributeFilter 可选项 从1.5.0版本有

                       此属性是用来控制Session中的那个属性值保存到Memcached中的正则表达式。郑则表达式被用来匹配Session中属性名称。如

                  sessionAttributeFilter="^(userName|sessionHistory)$" 指定了只有"userName"和"sessionHistory"属性保存到Memcached中。

                  依赖于选择的序列化策略。

            transcoderFactoryClass 可选，默认为 de.javakaffee.web.msm.JavaSerializationTranscoderFactory

                       此属性值是创建序列化和反序列化保存到Memcached中的Session的编码转换器的工厂类名。这个指定的类必须实现了de.javakaffee.web.msm.TranscoderFactory

                 和提供一个无参的构造方法。例如其他的有效的实现在其他packages/jars中提供如：msm-kryo-serializer,msm-xstrea-serializer和msm-javolution-serializer.

            copyCollectionsForSerialization 可选项，默认false。

            customConverter 可选项

            enableStatistics 可选项，默认true

                   用来指定是否进行统计。

            enabled 可选项，默认true

                    指定Session保存到Memcached中是否可用和是否可以通过JMX进行改变。只用于粘性Session。

## 6.5 使用Apache和Tomcat实现负载均衡

<http://www.kankanews.com/ICkengine/archives/20036.shtml>

# 7系统容灾分析

主要介绍在出现一些突发现象将如何处理，比如计算机在遭受到火灾、水灾计算机病毒、掉电、网络/通信失败、硬件/软件错误和人为操作错误等如何保证用户和系统的数据的安全性。

## 7.1数据容灾

数据容灾，就是指定建立一个异地的数据系统，该系统是本地关键应用数据的一个可用复制。在本地数据及整个应用系统出现灾难时，系统至少在异地保存有一份可用的关键业务的数据。该数据可以是本地产生数据的完全实时复制，也可以比本地数据略微落后，但一定是可用的。采用的主要技术室数据备份和数据复制技术。

数据容灾技术，有称为异地数据复制技术，按照其实现的技术方式来说，主要分为同步传输和异步传输方式，另外，也有如“半同步”这样的方式。半同步传输方式基本与同步传输方式相同，只有在Read占I/O比重较大时，相对同步传输方式，可以略微提高I/O的速度。

## 7.2应用容灾

是在数据容灾的基础上，在异地建立一套完整的与本地生产系统相当的备份应用系统（可以项目备份）。建立这样一个系统是相当比较复杂的，不仅需要一份可用的数据复制，还要有包括网络、主机、应用、甚至IP等资源，以及各资源之间的良好协调。主要的技术包括负载均衡、集群技术。数据容灾是应用容灾的技术，应用容灾是数据容灾的目标。