**网站缓存技术方案**

目录

[一、缓存分析](#_Toc426452475)

[二、动态内容缓存分析](#_Toc426452476)

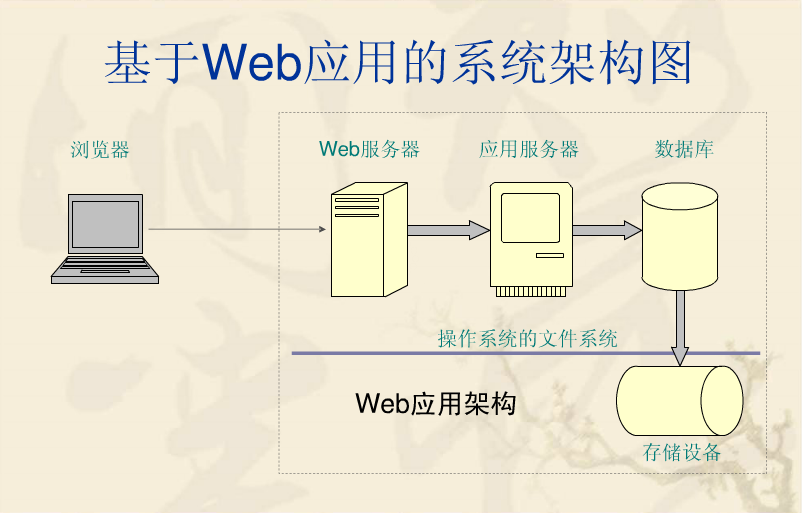
**[三、Redis缓存方案实施](#_Toc426452476)**[...............................................................................................................](#_Toc426452476)

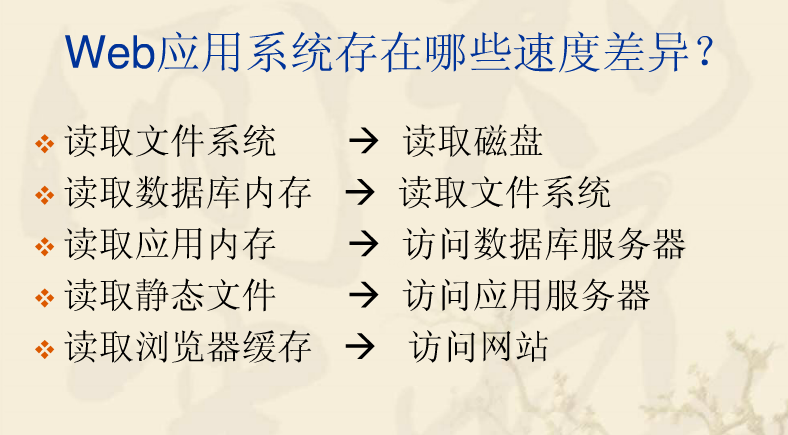
**一：缓存分析**

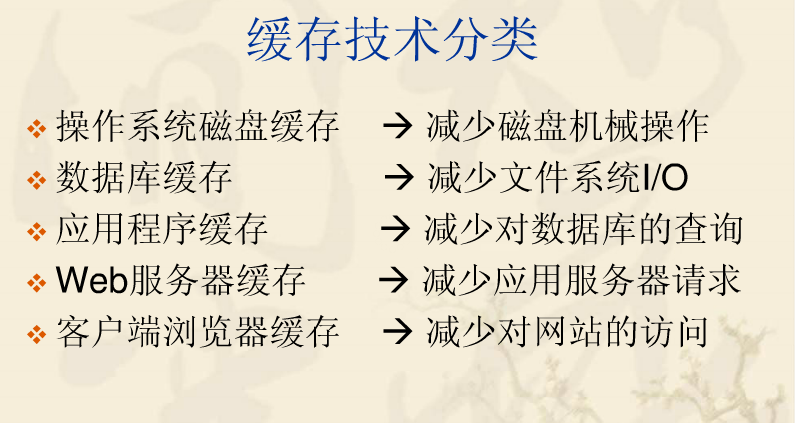
Cache是缓冲存储器，一种特殊的存储器子系统，其中复制了频繁使用的数据有利于快速访问。

凡是位于速度相差较大的两种硬件/软件之间，用于协调两者数据传输速度差异的结构，均可称之为Cache.

相关知识点请参考网上一些文档知识。本文档为俱体主要针对Redis的动态缓存方案。



****

****

以上问题我们将重点针对数据库/动态缓存的分析。

**1：Ｗeb服务器缓存**

应用程序把动态文件生成的html文件缓存到文件服务器，以后用户请求动态文件，直接从文件服务器加载对应的静态缓存的html文件返回给用户，这里面主要节省了动态语言的执行时间和数据库访问时间。但是会增加了缓存框架的加载和缓存查找的时间。通用用cms生成静态html解析。如phpcms。

别外可以建议

采用**nginx**的缓存模块操作内存，减少io文件的读写。

采用nginx+lua处理前置逻辑。

采用**google**开源一个nginx加速器 **pagespeed**优化页面和代码。

**2: 利用反向代理服务器的缓存**

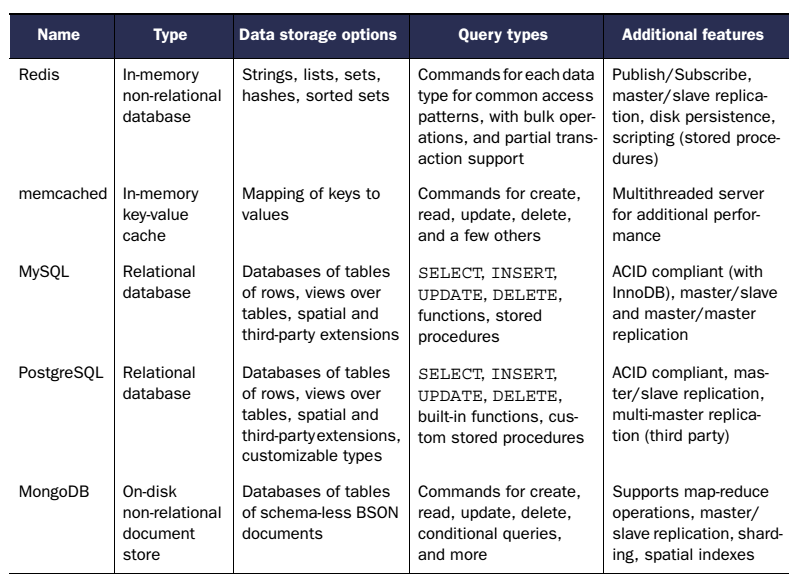
利用类似Nginx/haproxy的反向代理服务器，对请求的URL对应的输出的进行缓存。这个缓存和应用程序实现的动态页面缓存类似，只不过用反向代理充当了应用程序的缓存实现。主要节省了动态余元执行时间和数据库访问时间。

**3: 客户端浏览器缓存**

客户端浏览器缓存主要是通过在http头部增加Last-Modified,If-Modified-Since,Expires,Cache-Control等标识，和服务器进行协商，是否是采用客户的本机缓存来实现。

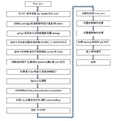
**二：动态缓存分析**

1:通过高速服务器Cache缓存数据库数据.



从以上各数据可知，对于我们产品最可行的技术方案有两种：

1. M**emcached/couchbase**       内存Key-Value Cache

[](http://www.baidu.com/link?url=mQkqeJOeMn0YgGVKVZCdagLGaSnRog5pAVdOtE8DsICb4nm4VaI3dwrDVu-y6Fy4LztuxtOmNJWHUueLQ-mDta)

memcache是一套分布式的高速缓存系统，由LiveJournal的Brad Fitzpatrick开发，但目前被许多网站使用以提升网站的访问速度，尤其对于一些大型的、需要频繁访问数据库的网站访问速度提升效果十分显著。

2.**Redis**          内存数据库

[](http://www.baidu.com/link?url=c5FDr-PU5R6x1FwtgnEO4hFE2EoagIRL5HgO5aW15p1KD37OaX97ZVgi34vOMYfvF4c5sVH6DVaWjF9IUGGWaa)

Redis是一个开源的使用ANSI C语言编写、支持网络、可基于内存亦可持久化的日志型、Key-Value数据库，并提供多种语言的API。

**后面我们会重点介绍Redis的服务架构。**

2:内存数据库.

**Mongodb数据库**

**三：Redis缓存方案实施**

redis是一个key-value存储系统。和Memcached类似，它支持存储的value类型相对更多，包括string(字符串)、 list(链表)、set(集合)和zset(有序集合)。这些数据类型都支持push/pop、add/remove及取交集并集和差集及更丰富的操 作，而且这些操作都是原子性的。在此基础上，redis支持各种不同方式的排序。与memcached一样，为了保证效率，数据都是缓存在内存中。区别的 是redis会周期性的把更新的数据写入磁盘或者把修改操作写入追加的记录文件，并且在此基础上实现了master-slave(主从)同步。

Redis集群需要考虑到　数据分片、主从备份、平滑升级、单点故障等等问题。

**数据分片通常用三个模式**

1：客户端分片，通过客户端代码把数据hash分散到多个redis服务中。

代码层面需要编写服务器代码地址。

优点：分片算法在客户端，可以减少服务器计算。

缺点：服务器地址增加和减少，代码也需要更换。

2：中间件分片，通过如twemprox/codis 反向分散数据到多个redis服务。

Twemproxy 主要是分片功能，但下面的机器坏的时候，会就有部分会话报错。

Codis 是一个分布式 [Redis](http://www.oschina.net/p/redis" \t "http://www.oschina.net/p/_blank) 解决方案, 对于上层的应用来说, 连接到 Codis Proxy 和连接原生的 Redis Server 没有明显的区别 (不支持的命令列表), 上层应用可以像使用单机的 Redis 一样使用, Codis 底层会处理请求的转发, 不停机的数据迁移等工作, 所有后边的一切事情, 对于前面的客户端来说是透明的, 可以简单的认为后边连接的是一个内存无限大的 Redis 服务.

参考：<http://www.oschina.net/p/codis>

优点：可以平滑增删机器，支持分片分组。有简洁的管理后台UI

缺点：布置相对麻烦，需要引入zookeeper go 目前只支持redis2.8版本

## 3：Redis cluster集群方案。

## 3-1: Redis cluster的现状

Reids-cluster计划在redis3.0中推出目前redis支持的cluster特性

1):节点自动发现

2):slave->master 选举,集群容错

3):Hot resharding:在线分片

4):进群管理:cluster xxx

5):基于配置(nodes-port.conf)的集群管理

6):ASK 转向/MOVED 转向机制.

## 3-2: Redis cluster 架构

1) redis-cluster架构图



架构细节:

(1)所有的redis节点彼此互联(PING-PONG机制),内部使用二进制协议优化传输速度和带宽.

(2)节点的fail是通过集群中超过半数的节点检测失效时才生效.

(3)客户端与redis节点直连,不需要中间proxy层.客户端不需要连接集群所有节点,连接集群中任何一个可用节点即可

(4)redis-cluster把所有的物理节点映射到[0-16383]slot上,cluster 负责维护node<->slot<->value

2) redis-cluster选举:容错

(1)领着选举过程是集群中所有master参与,如果半数以上master节点与master节点通信超过(cluster-node-timeout),认为当前master节点挂掉.

(2):什么时候整个集群不可用(cluster\_state:fail),当集群不可用时,所有对集群的操作做都不可用，收到((error) CLUSTERDOWN The cluster is down)错误

a:如果集群任意master挂掉,且当前master没有slave.集群进入fail状态,也可以理解成进群的slot映射[0-16383]不完成时进入fail状

b:如果进群超过半数以上master挂掉，无论是否有slave集群进入fail状态.

实施

综合分析后，可行的服务端集群方案主要有两种

1：codis

2: redis cluster 3.0

<http://www.open-open.com/lib/view/open1428469036104.html>

<http://redis.io/topics/cluster-tutorial>

<http://blog.csdn.net/beyondlpf/article/details/51275106>　安装过程中的问

题

redis模块安装成功后再启动redis cluster 终于成功了。。

<http://blog.csdn.net/a67474506/article/details/50435845>

<http://ruby.sdutlinux.org/>　山东ruby源，不需要ssl证书

3：哨兵+分片 2.8

客户端方案

1: jedis

2: spring-data-redis

问题总结

安装redis集群过程遇到各种问题

总体，

1把ruby升级2.3.1 gem再升级

再gem install redis

2 注意redis.conf的 bind 0.0.0.0

3 不要全部使用自带的redis.conf文件下面为样例子

port 7000

cluster-enabled yes

cluster-config-file nodes.conf

cluster-node-timeout 5000

appendonly yes

bind 0.0.0.0

src/redis-trib.rb create --replicas 1 10.73.128.85:7000 10.73.128.85:7001 \

10.73.128.86:7000 10.73.128.86:7001 10.73.128.87:7000 10.73.128.87:7001

然后客户端验证

<http://blog.csdn.net/a67474506/article/details/50435845> 写得很精彩

redis-cli -c

Jedis too many cluster redirections

集群的ip必需是jedis能连接到的ip