讲师介绍

Hash　　QQ：805921455

从事Java软件研发十年。

前新浪支付核心成员、

咪咕视讯(中国移动)项目经理、

对分布式架构、高性能编程有深入的研究。

明天，你一定会感谢今天奋力拼搏的你

Redis除了缓存，
你可能不知道的实用功能

分布式高并发—缓存技术

目录

课程安排

01

memcached入门

概念、安装、使用命令

02

Memcached设计理念

简单而强大

03

Memcached应用场景

Redis VS Memcached如何选择？它有哪些应用场景？

04

Memcache集群

Memcached集群操作

目录

memcached入门

memcached简介

免费和开源，高性能，分布式内存对象缓存系统。通过减轻数据库负载来加速动态Web应用程序，Memcached简单而强大。

- 本质上就是一个内存Key-Value缓存
- 协议简单，使用文本行的协议
- 不支持数据持久化，服务器关闭后数据全部丢失
- 简洁强大，上手容易，便于快速开发
- 没有安全机制

Memcached项目地址：https://github.com/memcached/memcached/wiki

安装和测试

```
yum install libevent-devel
wget https://memcached.org/latest
mv latest memcached-1.5.12.tar.gz
tar -zxf memcached-1.5.12.tar.gz
cd memcached-1.5.12
./configure --prefix=/usr/local/memcached
make && sudo make install

# 启动并使用Memcached
cd =/usr/local/memcached/
./bin/memcached -m 64 -p 11211 -u root -vvv
# 使用memcached
# telnet memcached_server_ip port例如：
telnet localhost 11211
```

常用启动参数	说明
-p <num>	监听的TCP端口 （默认：11211）
-U <num>	监听的UDP端口 （默认：11211，0表示不监听）
-l <ip_addr>	监听的IP地址（默认：INADDR_ANY，所有地址）
-d	作为守护进程来运行
-u <username>	设定进程所属用户（仅root用户可以使用）
-m <num>	所有slab class可用内存的上限（默认：64MB）
-v	提示信息（在事件循环中打印错误/警告信息）
-vv	详细信息（还打印客户端命令/响应）
-vvv	超详细信息（还打印内部状态的变化）

常用命令

分组	命令	描述
存储命令	set	用于将 value 存储在指定的 key中。key已经存在，更新该key所对应的原来的数据。
	add	用于将 value存储在指定的 key中，存在则不更新。
	replace	替换已存在的 key的value，不存在，则替换失败。
	append	命令用于向已存在 key的 value后面追加数据
	prepend	向已存在 key的 value前面追加数据
	cas	比较和替换，比对后，没有被其他客户端修改的情况下才能写入。
检索命令	get	获取存储在 key 中的 value，不存在，则返回空。
	gets	获取带有CAS令牌存的value，若key不存在，则返回为空
删除	delete	删除已存在的 key
计算	incr/decr	对已存在的 key的数字值进行自增或自减操作
统计	stats	返回统计信息例如 PID(进程号)、版本号、连接数等
	stats items	显示各个 slab 中 item 的数目和存储时长(最后一次访问距离现在的秒数)
	stats slabs	显示各个slab的信息，包括chunk的大小、数目、使用情况等。
	stats sizes	显示所有item的大小和个数
清除	flush_all	清除所有内容

客户端使用

JAVA客户端（xmemcached）示例：

```
MemcachedClient client=new XMemcachedClient("host",11211);

//同步存储value到memcached，缓存超时为1小时，3600秒。
client.set("key",3600,someObject);
//从memcached获取key对应的value
Object someObject=client.get("key");

//从memcached获取key对应的value,操作超时2秒
someObject=client.get("key",2000);
//更新缓存的超时时间为10秒。
boolean success=client.touch("key",10);

//删除value
client.delete("key");
```

客户端支持的特性：集群下多服务器选择、节点权重配置、失败/故障转移、数据压缩、连接管理

Xmemcahed项目地址：https://github.com/killme2008/xmemcached/wiki

服务端配置

- 命令行参数

  查看memcached -h或man memcached获取最新文档

- init脚本

  如果通过yum应用商店安装，可以使用/etc/sysconfig/memcached文件进行参数配置

- 检查运行配置

  stats settings 查看运行中的memcached的配置（可以用telnet连接memcached进行测试）

目录

Memcached设计理念

Memcached设计理念

- **简单的键/值存储**

  服务器不关心您的数据是什么样的，只管数据存储

- **服务端功能简单，很多逻辑依赖客户端实现**

  客户端专注如何选择读取或写入的服务器，以及无法联系服务器时要执行的操作。

  服务器专注如何存储和管理何时清除或重用内存

- **Memcached实例之间没有通信机制**

- **每个命令的复杂度为 O(1)**

  慢速机器上的查询应该在1ms以下运行。高端服务器的吞吐量可以达到每秒数百万

- **缓存自动清除机制**

- **缓存失效机制**

Memcached性能

Memcached性能的关键是硬件，内部实现是hash表，读写操作都是O(1)。硬件好，几百万的QPS都是没问题的。

最大连接数限

内部基于事件机制(类似JAVA NIO)所以这个限制和nio类似，只要内存、操作系统参数进行调整，轻松几十万。

集群节点数量限制

理论是没限制的，但是节点越多，客户端需要建立的连接就会越多。

如果要存储的数据很多，优先考虑可以增加内存，成本太高的情况下，再增加节点。

memcached服务端没有分布式的功能，所以不论是集群还是主从备份，都需要第三方产品支持。

服务器硬件需要

- CPU要求

  CPU占用率低，默认为4个工作线程

- 内存要求

  memcached内容存在内存里面，所有内存使用率高

  建议memcached实例独占服务器，而不是混用。

  建议每个memcached实例内存大小都是一致的，如果不一致则需要进行权重调整。

- 网络要求

  根据项目传输的内容来定，网络越大越好，虽然通常10M就够用了。

  建议：项目往memcached传输的内容保持尽可能的小

这些应该在前期规划的时候尽量考虑进去

目录

Memcached应用场景

Redis与Mamcached

功能点	Mamcached	Redis
操作线程	支持多线程	单线程
数据支持类型	K/V类型，大V	支持多种数据类型，V 512M限制
存储方式	内存中	支持AOF、RDB持久化
Key淘汰过期	LRU	LRU、LFU、Random、Noeviction、TTL
安全	没做安全控制	有安全控制
集群	本身没有支撑	主从、分片集群支持高

Memcached应用场景

应用场景：

1、 数据查询缓存：将数据库中的数据加载到memcahced，提供程序的访问速度。

2、 计数器的场景：通过incr/decr命令实现评论数量、点击数统计、操作次数等等场景。

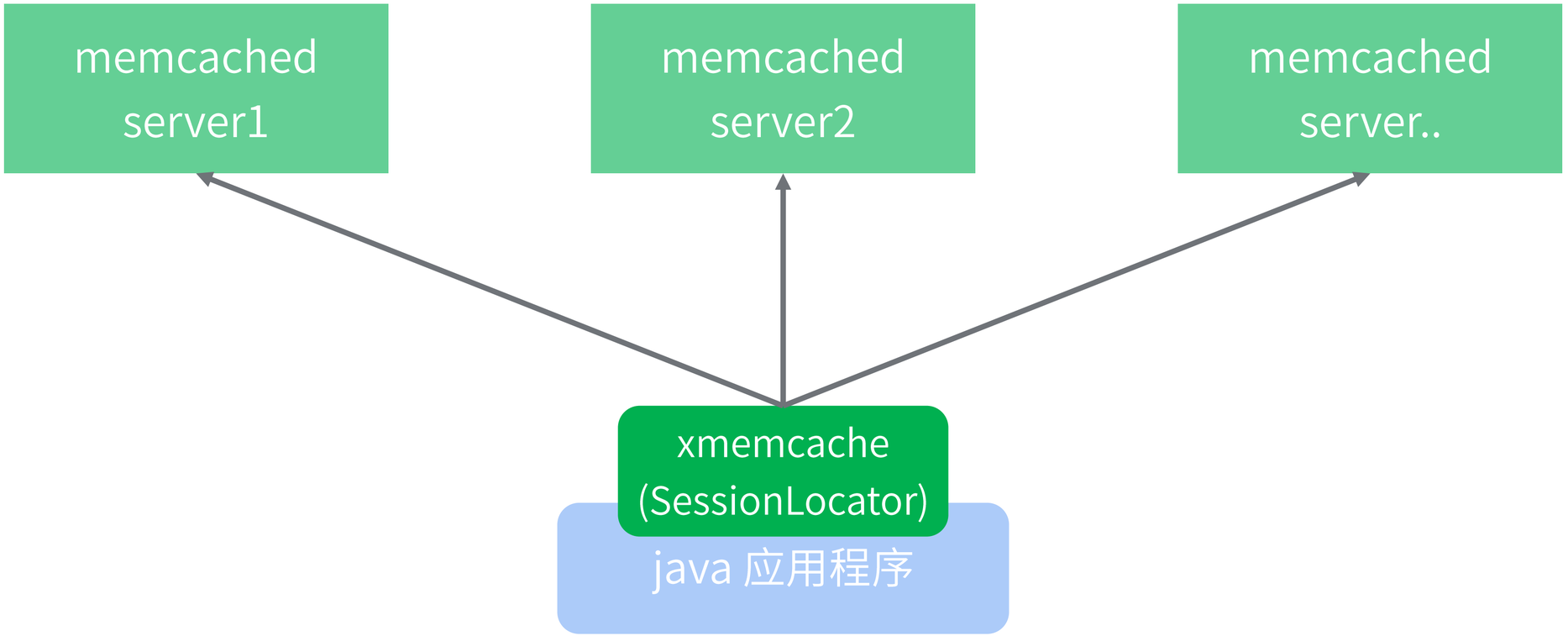
3、 乐观锁实现：例如计划任务多实例部署的场景下，通过CAS实现不重复执行。

4、 防止重复处理：CAS命令

目录

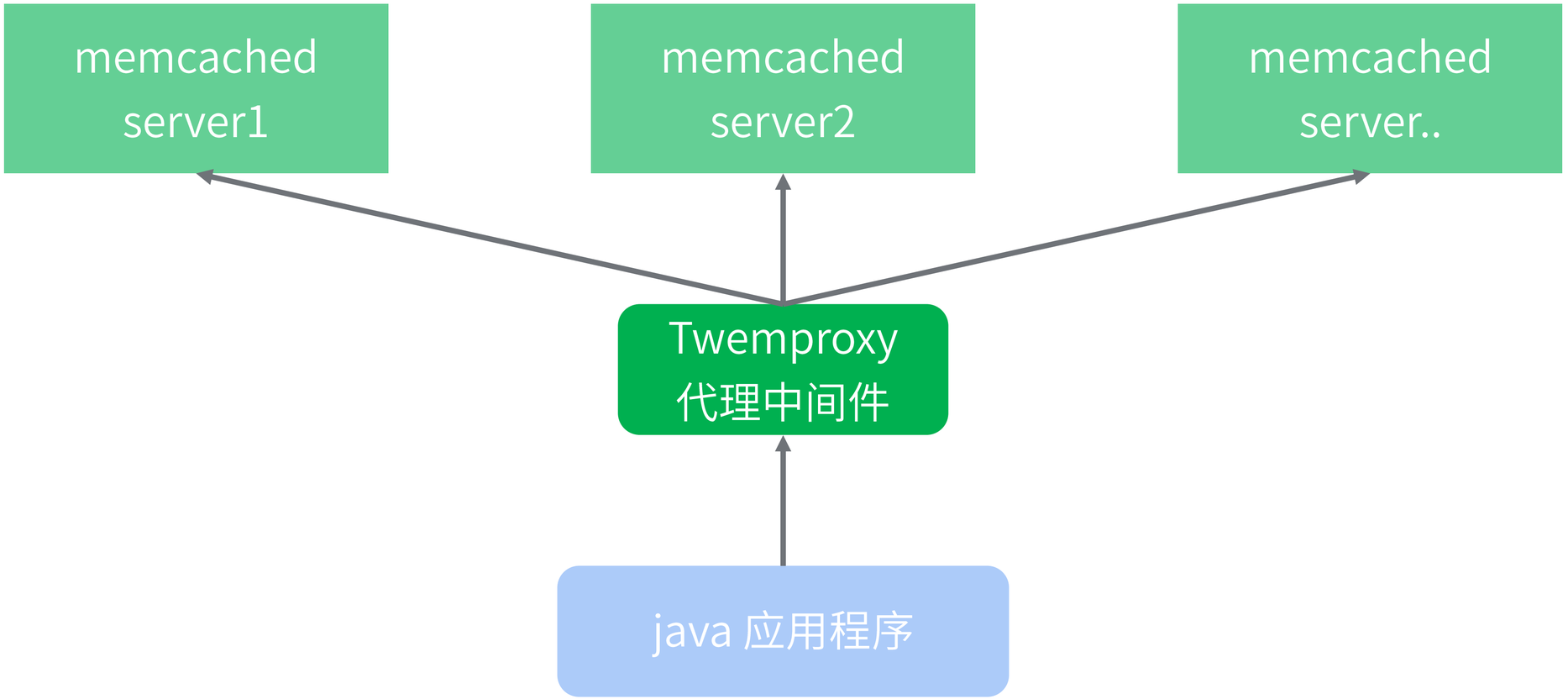
Memcache集群

集群方案 – 客户端支持



Java客户端根据key，通过hash取模或者其他一致性算法，最终选择一个server进行连接。

集群方案 – Twemproxy



推特开源的中间件，实现memcached代理。

对于JAVA应用程序就像使用一个普通的memcached一样

Memcached监控命令

监控命令	类型
stats	返回memcached的统计信息
stats settings	正在运行的memcached的设置
stats items	每个slab class的存储信息
stats sizes	存储在缓存中的所有item的常规大小和计数
stats sizes_enable	启动直方图的形式展示sizes信息
stats sizes_disable	禁用直方图
stats slabs	已激活的slab的信息
stats conns	连接信息
stats reset	清空统计数据

谢谢观看