京东商品单品页统一服务系统架构未公开的细节

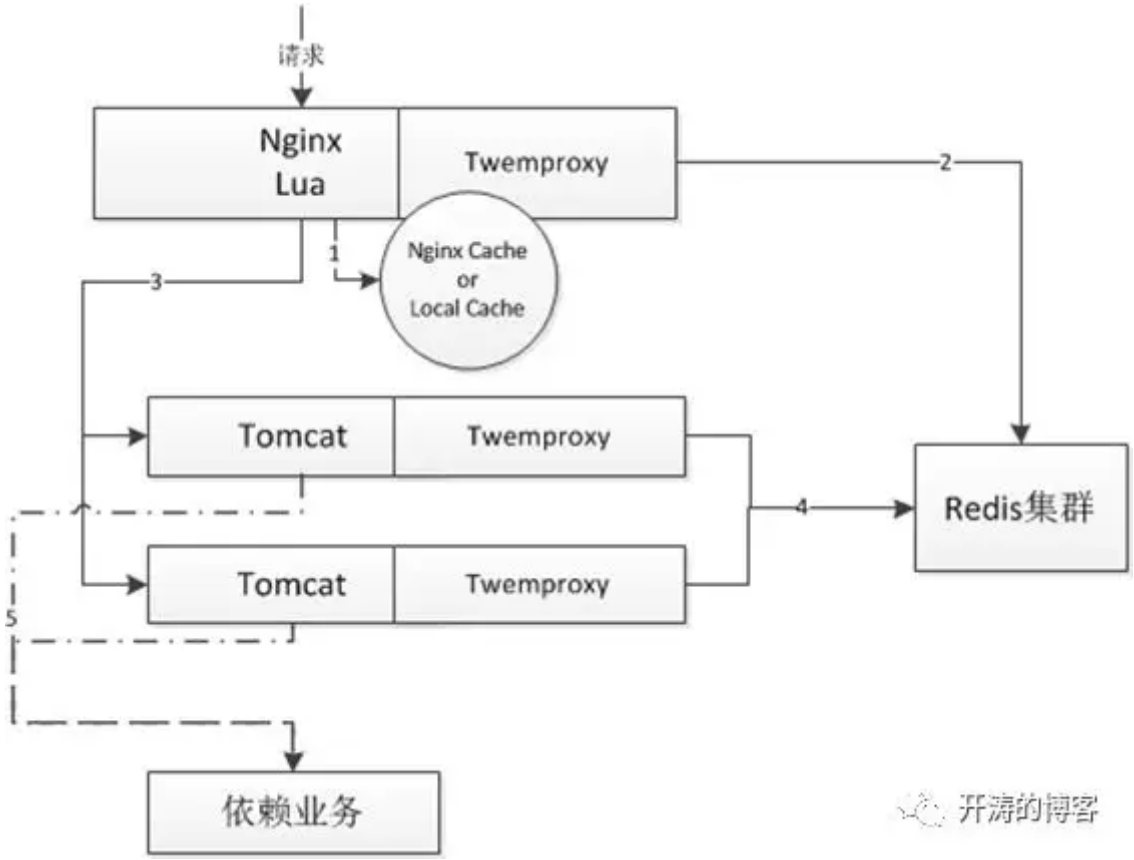
京东技术 2018-10-09

来这里找志同道合的小伙伴！

本文是《京东商品详情页服务闭环实践》中未公开的一些细节，是15年内部培训的 PPT，目前的内容也不过时，还适用现有系统架构设计。

PPT下载地址：https://pan.baidu.com/s/1K-Djkf6IFZ7qSEIlNqYPAw

架构



单品页依赖服务众多，分布在各个部门。问题：

- 服务质量没有监控数据
- 出现问题不能及时降级
- 接口调用分散化

- 域名重复解析，没有长连接的优势

1 架构总体原则

- 设计上无状态
- 使用 nginx+lua+tomcat7 架构
- 充分利用 localcache（proxycache or shared_dict orjava guava cache）
- 使用 localtwemproxy 做 redis 分片，且缓存分离（重要业务与其他业务分离）
- 分离线程池，核心业务与非核心业务分离，且有线程池监控和开关
- 异步化更新
- Redis 集群使用主从架构
- 使用 unixdomain socket 减少连接数
- 使用 keepalive 长连接
- 考虑好开关
- 缓存时间、是否调用后端服务、托底（托底恢复采用指数/随机数恢复机制）

2 Twemproxy+Redis

- 使用 localtwemproxy 做 redis 分片，且缓存分离（重要业务与其他业务分离）Redis 集群使用集中式主从架构/ 考虑复制缓冲区大小
- 考虑使用 unix domain socket/ 套接字放内存文件系统

- 考虑使用 HashTag
- Redis 考虑缓存驱逐策略：maxmemory-policy allkeys-lru maxmemory-samples 10

3 Nginx keep alive

- 使用长连接，并限制长连接数量

4 Nginx timeout

- 设置超时

5 Nginx proxy cache

- 使用内存文件系统进行 Nginx Proxy Cache

- 注意响应头对 cache 时间的影响

> Parameters of caching can also beset directly in the response header. This has higher priority than setting ofcaching time using the directive.
>
> •The “X-Accel-Expires” header field sets caching time of a response in seconds. Thezero value disables caching for a response. If the value starts with the @ prefix, it sets an absolute time in seconds since Epoch, up to which the response maybe cached.
>
> •If the header does not include the “X-Accel-Expires” field, parameters of cachingmay be set in the header fields “Expires” or “Cache-Control”.
>
> •If the header includes the “Set-Cookie” field, such a response will not be cached.
>
> •If the header includes the “Vary” field with the special value “*”,such a response will not be cached (1.7.7).
>
> •If the header includes the “Vary”field with another value, such a response will be cached taking into accountthe corresponding request header fields (1.7.7).

- 数据有问题不要缓存

6 Nginx DNS

- proxy_pass 时使用 Local DNS 解析

- 可能解析到多个 server（nslookup），会自动 next upstream
- Nginx plus 支持 upstream 的动态解析

7 Nginx Gzip

- 根据自己需求设置 gzip_comp_level、gzip_min_length、gzip_types

8 Nginx upstream

- upstream 检查

- upstream 策略 ip_hash

- upstream 策略 hash key [consistent]

consistent_key 是根据流量负载动态计算的，如根据 ip 计算：

```
local newval, err = ip_log:incr(ip, 1)
local toDelay = false
-- 单IP恶意请求 分流到固定机组
if newval > 50 then
    toDelay = true
    ngx_var.consistent_key = ngx_var.consistent_key .. '_' .. newval
    --要在set_uri前执行,否则执行不到
end
```

9 Nginx Real ip

10 Nginx client header

限定请求头和请求体大小，在 proxy pass 到后端服务时不传输请求头和请求体，减少网络交互。

11 Nginx limit

- limit request
- limit connection / limit rate
- ip 白名单/黑名单
- user-agent 白名单/黑名单
- Token 限流
- 漏桶算法令牌桶算法（JavaGuava rate limit）
- Delay 限速

参考聊聊高并发系统之限流特技-1、聊聊高并发系统之限流特技-2

12 Nginx+Lua shared_dict

- 使用共享字典做 local cache

13 Nginx+Lua 接口合并

- 请求时使用 method 参数表示请求哪个服务

- 数据过滤逻辑前置，不合法直接403（防止XSS）

- 封装调用逻辑，参数顺序等固定，提升缓存命中率

- 通过 Nginx 子请求（ngx.location.capture_multi）进行合并
- 只对原子接口进行 Cache

- 通过一层代理重试或者记录 UMP 日志

14 Nginx+Lua 记录日志

- 记录日志还可以通过

Java 架构

- 异步非阻塞事件模型

从 Servlet3 开始支持异步模型，Tomcat7/Jetty8 开始支持，相同的概念是Jetty6 的 Continuations。我们可以把处理过程分解为一个个的事件。

通过这种将请求划分为事件方式我们可以进行更多的控制。如，我们可以为不同的业务再建立不同的线程池进行控制：

即我们只依赖 tomcat 线程池进行请求的解析，对于请求的处理我们交给我们自己的线程池去完成；这样 tomcat 线程池就不是我们的瓶颈，造成现在无法优化的状况。

通过使用这种异步化事件模型，我们可以提高整体的吞吐量，不让慢速的 A 业务处理影响到其他业务处理。慢的还是慢，但是不影响其他的业务。

通过这种将请求划分为事件方式我们可以进行更多的控制。如，我们可以为不同的业务再建立不同的线程池进行控制：

1 Java Tomcat

- start.sh

```
export JAVA_OPTS="-Djava.library.path=/usr/local/lib -server  -XX:-UseConcMarkSwe
```

-XX:+UseConcMarkSweepGC 表示使用 CMS

-XX:+CMSParallelRemarkEnabled 表示并行 remark

-XX:+UseCMSCompactAtFullCollection 表示在 FGC 之后进行压缩，因为CMS 默认不压缩空间的

-XX:CMSInitiatingOccupancyFraction=80 设置阀值为80%，默认为68%

-XX:SoftRefLRUPolicyMSPerMBsoftly reachable objects will remain alive for some amount of time after thelast time they were referenced. The default value is one second of lifetime perfree megabyte in the heap

-XX:NewRatio年轻代（包括Eden和两个Survivor区）与年老代的比值（不包括持久代）

-XX:SurvivorRatio Eden 区与 Survivor 区的大小比值

- server.xml

```
<Connector port="1601" asyncTimeout="10000" acceptCount="10240" maxConnections="1
```

以 Tomcat 6 为例，其 Connector 有几个关键配置：

BIOm 实现：

acceptCount：在超过最大连接数时，可接受的排队数量；超过这个值就直接拒绝连接；默认100；
maxThreads：tomcat 可创建的最大线程数，没线程处理一个请求，它决定了 tomcat 最大线程阀值；默认200；
minSpareThreads：最小备用线程数，即 tomcat 一启动就创建的线程数；默认25；(使用 Executor 时配置)
maxQueueSize：最大备用线程数，一旦创建的线程超过这个值 tomcat 就会关闭不活动的线程；默认 Integer.MAX_VALUE；(使用 Executor 时配置)

NIO 实现（继承如上的配置）：

acceptorThreadCount：接受连接的线程数；默认1，可以根据 CPU 核数调整；如果没有问题默认1个即可，基本不需要改；
pollerThreadCount：运行选择事件的线程个数；默认每核一个；
processorCache：协议处理器缓存 Http11NioProcessor 对象的个数，目的是提高性能，默认200，建议其值接近 maxThreads。

对于 tomcat7 的相关配置可以参考官网 http://tomcat.apache.org/tomcat-7.0-doc/config/http.html；核心差不多。

2 Java servlet3

3 Java thread pool

- 线程池并发执行任务获取数据

- 异步更新缓存

4 Java cache

- local cache guava

- 批量接口时，对单个数据进行缓存；首先查询单个缓存，然后对 miss 数据进行批量获取，最后合并为结果

5 其他

- 域名分区：客户端同域连接限制，进行域名分区：c.3.cn c1.3.cn c2.3.cn

- 充分使用 CPU，比如绑定 CPU 核数
- 考虑减少连接数
- 考虑使用内存文件系统
- 考虑大内存或企业级 SSD
- 全部使用弹性云

PPT下载地址：**https://pan.baidu.com/s/1K-Djkf6IFZ7qSEIlNqYPAw**。

----------END----------

下面的内容同样精彩

点击图片即可阅读

京东技术
---关注技术的公众号

长按识别二维码关注