Induction training of Dev Team

Cache

By PuGong

Numbers Everyone Should Know

L1 cache reference	0.5 ns
Branch mispredict	5 ns
L2 cache reference	7 ns
Mutex lock/unlock	100 ns
Main memory reference	100 ns
Compress 1K bytes with Zippy	10,000 ns
Send 2K bytes over 1 Gbps network	20,000 ns
Read 1 MB sequentially from memory	250,000 ns
Round trip within same datacenter	500,000 ns
Disk seek	10,000,000 ns
Read 1 MB sequentially from network	10,000,000 ns
Read 1 MB sequentially from disk	30,000,000 ns
Send packet CA->Netherlands->CA	150,000,000 ns

Google

List

- Product / Develop / Operation
- Software lifecycle
- ITIL
- Source Control
- Evolution of A Website's Architecture
- AuAz & SSO
- ***Cache***
- Message Queue
- Storage
- Database and SQL
- NoSQL & New SQL
- TOGAF & 4+1 Arch View
- 测试
- 发布
- 监控

What’s Cache

- Cache定义：
  - a component that transparently stores data so that future requests for that data can be served faster
- Cache适合
  - 访问频繁
  - 数据变化不频繁/对数据不一致不敏感
- Cache算法及更新规则
  - LRU/LFU/FIFO
  - 主动更新/被动更新
- Cache系统是不可靠的，必须考虑failure情况下如何处理

Cache

- Cache类型
  - 进程内缓存：本地缓存，速度快，受限于本地硬件
  - 进程外缓存：中央缓存/分布式缓存，可扩展性强，维护方便，需要额外的网络以及序列化/反序列化开销。常用Memcached / Redis
- Cache Design
  - 类型选择
  - Key-value设计: Key的粒度/复杂度/冲突；Value大小
  - 监控：CPU/Memory/Hit Ratio etc

Cache适用场景

- 本地缓存的适用场景
  - 缓存数据需要进行大量计算后给出
  - 典型场景：线路规划, 推荐引擎等
  - Pitfalls: Cache build / Lazy Loading
- Memcached / Redis
  - 热点数据，简单查询
  - 典型场景：查询结果缓存，等
  - Pitfalls: Cache Build/ Value太小/Value超过1M
- 两级缓存 (本地+Memcached / Redis)
  - 是否还有必要?

Cache Sharding

- Redis Cluster
- Consistent Hashing

Cache常见问题 I

- Lazy Loading/被动更新
  - 不要用阻塞组流程模式，否则设置/更新缓存失败导致主流程崩溃
  - 建议方式：另起线程更新，主线程正常处理
- 主动更新 Cache Get失败
  - 后果：直接从数据库获取，增加数据库压力
  - 可能原因：key设计不合理；缓存失效等
  - 建议方针：异步设置标志或者通知缓存更新应用，由其决定是否更新

Cache常见问题 II

- Cache Set失败
  - 后果：设置缓存失败，应用之后的读取依然到数据库
  - 可能原因：key设计不合理；内存不足等
  - 建议方针：监控可用内存，及时将过期或者Hit Rate低的缓存清理掉；合理设置缓存尺寸；缓存内容可以根据情况进行压缩
- Cache 监控
  - 使用AppInternals / Metrics API
  - Memory
  - CPU
  - Cache hit ratio

Cache常见问题 III

- Cache 重建
  - 后果：瞬时对数据库造成压力，可能会造成雪崩效应
  - 可能原因：应用发布/系统维护等(本地缓存)；系统维护/服务器故障(Memcached)；
- 建议方针：
  - 避免集中重建：发布工艺调整：设置重建时间；增加重建锁，同时只允许指定数据量的服务器重建
  - Cross cache server sync：从已建好的Cache Server创建，减少db压力。需要Server之间有通讯
  - Local/Share file for Cache persistency：从缓存文件创建，类同Cross cache server sync，不需要Server之间的通讯

Reference

- Consistent Hashing： http://xiexiejiao.cn/java/memcached-consistent-hashing.html
- Redis Source Code: https://github.com/antirez/redis