Speaker: John Doe
Email: no_reply@example.com

CACHE & DYNAMODB

—

Notes

Introduction: Why Cache?

Scenario #1 So your system has been running pretty well, but you realized there are a few expensive SQL queries that slows down your whole application.

And you are wondering how we could mitigate this issue and reduce the latency!

Scenario #2 So your web application has been serving your customer pretty well. But you are expecting a LOT more requests tomorrow since your company will be launching a marketing campaign and driving in new customers.

- So why cache?
  - To reduce latency.
  - To increase throughput. 我们还注意到，许多请求使用的下游资源或查询结果相同，所以我们认为对这些数据进行缓存可以解决这个问题。
  - To reduce costs, because it can be cost effective. 随着调用量的增加，扩展数据库的成本会很高。
  - To take the pressure off your expensive backend services.
- When NOT to use cache?
  - Write-heavy system you have.
  - Requires strong consistency.
  - 先明白一个前提。就是如果对数据有强一致性要求，不能放缓存。我们所做的一切，只能保证最终一致性。另外，我们所做的方案其实从根本上来说，只能说降低不一致发生的概率，无法完全避免。因此，有强一致性要求的数据，不能放缓存。
- It doesn't come for free though 缓存令人欣喜令人忧

    - Data inconsistency: There are differences in data between the cache layer and the storage layer
    - Code maintenance cost: maintain the logic of the cache layer and storage layer at the same time.
    - Operation and maintenance costs
    - 缓存的数据必然随时间的推移而与源数据不一致。例如，相对静态或变化缓慢的数据可以缓存较长的时间。
    - 依赖于缓存的服务。
        - 缓存的地位已在不经意间从对服务的有益补充，提升到维持操
        - 作必要且关键组成部分。此问题的核心在于缓存引入的模式行为，根据给定的对象是否已缓存，其行为
        - 有所不同。一旦这种模式行为的分配发生意外变化，就可能导致灾难。
        - 我们在 Amazon 的构建和运营服务过程中体验了缓存的优势和带来的挑战。本文的其余部分将介绍我们学到的经验教训、最佳做法以及使用缓存的注意事项。
    -

Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan.

Now we encounter a problem.

- At vero eos et accusam et justo duo dolores et ea rebum
    - Ut wisi enim ad minim veniam.
    - Quis nostrud exerci tation ullamcorper.

本地缓存

- 表现形式
    - 例如springBoot上边的cache，就是属于application Server上边的本地cache
- 优点
    - 机上缓存（通常在进程内存中实施）相对而言比较快速且易于实施，并且只需最少的工作即可提供明显改善。
    - 机上缓存通常是在识别缓存需求之后实施和评估的第一个方法。
    - 与外部缓存相比，它们无额外的运营开销，因此在集成到现有服务时的风险相对较低。
    - 我们通常以内存中哈希表格的形式实施机上缓存，通过应用程序逻辑（例如，在完成服务调用后将结果明确放入缓存中）进行托管，或嵌入到服务客户端（例如，使用缓存 HTTP 客户端）中。
- 尽管内存中缓存具有诸多优势并且非常简单，但仍存在一些缺点。
    - 其一是队列中各个服务器中的已缓存数据不一致，出现缓存一致性问题。

        - 如果客户端重复发起调用，他们可以在第一次调用中使用更新的数据，在第二次调用中使用旧的数据，这取决于哪台服务器来处理请求。
    - 另一个缺点是下游负载现在与服务的队列大小成比例，因此随着服务器数量的增加，它仍可能超出从属服务的能力。
        - 我们发现，监控此情形的有效方式是发送缓存命中/未命中的指标，以及向下游服务提出的请求数。
    - 内存中缓存还容易受到“冷启动”问题的影响。
        - 如果新服务器启动时缓存完全为空，则会出现这些问题。
        - 空缓存会导致从属服务填充缓存时，向该从属服务发出的请求激增。在部署期间或在整个队列范围内清除缓存的其他情况下，这可能是一个重要的问题。缓存一致性和空缓存问题通常可以通过使用请求合并来解决（详情请见下文所述）。
    -

外部缓存

- 优点
    - 外部缓存可以解决我们刚才讨论过的许多问题。
    - 外部缓存将缓存的数据存储在单独的队列中，例如使用 Memcached 或 Redis。
    - 缓存一致性问题减少，因为外部缓存保留了队列中所有服务器使用的值。（请注意，这些问题并未完全消除，因为更新缓存时可能会出现故障。） 与内存中的缓存比，下游服务的总体 负载降低，与队列大小不成比例。
    - 在部署等事件期间不存在冷启动问题，因为在整个部署过程中，外部缓存仍处于填充状态。(缓存预热)
    - 最后，外部缓存比内存中的缓存提供更多的可用存储空间，从而减少因空间限制而导致缓存被移出的情况。
- 但是，需要考虑外部缓存的一些缺点。
    - 第一个考虑因素是整体系统复杂性和运行负载增加，因为还有额外的队列需要监控、管理和扩展。缓存队列的可用性特征与将其作为缓存的相关服务不同。
    - 缓存队列的可用性通常较低，例如，如果它不支持零停机时间升级，并且需要维护时段。

如果新服务器启动时缓存完全为空？

内存中缓存还容易受到“冷启动”问题的影响。如果新服务器启动时缓存完全为空，则会出现这些问题。

空缓存会导致从属服务填充缓存时，向该从属服务发出的请求激增。在部署期间或在整个队列范围内清除缓存的其他情况下，这可能是一个重要的问题。缓存一致性和空缓存问题通常可以通过使用请求合并

来解决（详情请见下文所述）。

Dfd

References:

Caching-challenges-and-strategies.pdf
https://d1.awsstatic.com/zh_CN/builderslibrary/pdfs/caching-challenges-and-strategies.pdf?did=ba_card-body&trk=ba_card-body

Redis Cache https://zhuanlan.zhihu.com/p/59168140

双写一致性专题 https://www.cnblogs.com/rjzheng/p/9041659.html