07 | 数据库选型：如何基于性能需求选择合适的数据库？

2021-06-01 尉刚强

《性能优化高手课》 课程介绍 >

讲述：尉刚强
时长 18:34 大小 17.01M

你好，我是尉刚强。

我们知道，在软件系统的性能建模分析设计中，并行架构设计、IO 模型设计、内存模型设计是最核心的三个维度，它们决定了最终产品的性能底座。而在互联网应用服务产品中，内存模型设计与 IO 模型设计的大部分职责，其实在很大程度上都沉淀到了数据库服务与消息中间件中来实现。

所以，这节课我会聚焦在数据库选型上，以一个真实的性能需求案例为引导，来带你了解在设计或性能优化阶段，如何寻找备选数据库并进行初步地分析、筛选，然后基于性能评估与软件设计的权衡，来进行数据库选型与方案设计的过程方法。

不过我们也要知道，数据库的种类非常繁多，要了解每一款数据库的功能与性能其实是不现实的，所以这节课我主要的目的就是**帮你建立基于性能的数据库选型思路**。这样，当你面对不熟悉的性能需求或问题时，也可以做到有的放矢，并能够借助这套数据库选型的过程方法，找到合适的数据库与设计方案。

现在，我先来给你介绍下这个性能需求案例。这是一个 SaaS 服务产品中的分析服务，主要功能是针对客户提交的数据进行查询搜索并生成报表数据。通过对产品线上的数据规模和特征分析，以及对系统的核心业务流程分析，我们可以挖掘出其最核心的性能需求：

数据规模	数据类型	字段规模	查询时延	分析时延
10~100亿条	结构化规则数据	200列左右	<500ms	<500ms

实际上，数据查询与搜索、报表生成都是互联网产品的核心业务场景，也是性能问题频发的重灾区。而在软件设计阶段从性能的角度出发，选择合适的数据库与架构设计，就可以在很大程度上避免出现这样的问题。

OK，在识别出典型的性能需求之后，我们接下来的工作就是寻找到所有满足这些条件的备选数据库列表了。

基于性能寻找备选数据库列表

在寻找备选数据库的过程中，我们首先应该关注的是**广度**。不过由于数据库的种类太多，而且人都存在认知的局限性，会很容易漏掉一些非常有价值的数据库产品，从而导致无法寻找到一个最佳的方案。所以这里，我给你推荐一个网站，这是一个专门收集和呈现数据库管理系统信息的数据库引擎排名，里面列举了超过 300 多种数据库产品，大部分的开源和商业数据库都在列。

可是，直接在这个数据库列表中进行大范围筛选，工作量还是比较巨大的，所以我们可以通过认识和学习数据库的大体分类，来提升筛选的效率。下面我就给你简单介绍下。

在业内，对数据库的通用分类主要包括 5 个大类：

1. **关系数据库**：以 MySQL、Oracle、PostgreSQL 为代表，这些是结构化的关系型数据库，主要基于 SQL 进行操作；
2. **文档数据库**：以 MongoDB、Cassandra、Elasticsearch 为代表，它支持灵活的半结构数据存储，如 JSON 等；
3. **时序数据库**：主要服务于监控、日志类的数据存储，它支持按照时间维度进行存储与分析；
4. **Key-Value 数据库**：以 Aerospike、Redis 等为代表，它支持典型数据结构的快速存储访问；
5. **图数据库**：支持图的存储，典型应用场景包括知识图谱、关键路径搜索等。

> 关于数据库更详细的种类及功能介绍，可以参考我之前写的文章资料。

那么，针对我们前面列举出的核心性能需求，可以发现其数据是标准结构化的，因此选择关系数据库就是一种让性能更加优越的实现方式。而对于关系数据库，按照典型应用场景我们可以分为 **OLTP**（On-Line Transaction Processing，联机事务处理）和 **OLAP**（On-Line Analytical Processing，联机分析处理）两种。

其中，OLTP 的事务能力更强，OLAP 则是分析性能更佳，但是事务能力偏弱。由于我们的业务性能需求需要对大规模的数据进行分析，而且时延要求极高，因此综合评定后，OLAP 的数据库性能会更加适合。

这样我们再通过前面的数据库引擎排名，可以搜索到的备选数据库列表如下（实际列表比较庞大，我没有全部列出，这里仅作示意）：

ClickHouse

Redshift

Greenplum

阿里云 AnalyticDB

MemSQL

Vertica

Hlive

SparkSQL

……

不过到这里，如果我们要对每一个备选数据库都进行系统的性能评估分析的话，那么成本还是会非常大。所以接下来，我们需要在这些备选数据库列表中进一步筛选一个子集，然后再进行深入的性能评估和分析。

理论评估筛选出待分析数据库

筛选待分析数据库的过程实际上会比较复杂，有些数据库可能你已经深入使用过，对它的功能与性能有深入的理解，那么是比较幸运的。不过如果你遇到的并不是非常熟悉的数据库的话，那你可以基于以下手段来进行初步的理论评估，识别或过滤掉一部分数据库产品：

1. 数据库官网提供的性能指标；
2. 数据库官网的存储模型与架构视图；
3. 第三方给出的性能测试分析报告；
4. 网上部分用户的性能测试分析结论。

相比较而言，**官网提供的性能指标、存储模型及架构视图是比较权威的**，你在选择数据库之前需要去深入了解学习。就拿我们前面提到的性能需求案例来说，基于官方文档的分析中，你会发现 Hlive 和 SparkSQL 数据库依赖 Map-Reduce，要动态创建分析请求的任务并上传时，准备时间会有些长，而当在业务中实时动态创建性能分析的请求时，性能很可能是不太理想的；而 MemSQL 开源版本有容量规模的限制，也很可能与产品业务的数据规模需求不一致，因此同样需要排除在外。

除此之外，**进一步理解和学习一些典型的数据库相关实现架构**，比如 MPP（Massively Parallel Processing）架构、Hadoop 生态架构等，也可以帮助你更好地识别数据库。当然，**仅从数据库的视角进行理论分析评估是不够的**，你可能还需要分析、评估潜在的外部性能影响因素，比如说，网络传输带宽的影响、跨地区或者跨云之间的传输时延开销，等等。

这里我给你举个例子，Redshift 和阿里云 ADB 都是云服务提供的数据库，如果你的产品业务是部署在阿里云上，那么使用 AWS 云上的 Redshift 就会引起额外的传输延迟抖动，甚至可能会影响产品的稳定性。

这样，当你的产品并没有极致的、严格的性能需求时，基于上述的搜索与理论评估，你就可以锁定到具体的数据库产品中，接下来重点工作就只需要进行原型验证，测试性能满足需求即可。

不过在不少的业务场景下，很多产品其实是有极致的性能要求的，而且这通常也是产品在行业内的核心竞争力体现。那么这时候，你就需要对备选数据库进行更深入的性能评估分析，甚至还需要调整软件设计与优化来应对。

好，针对前面我介绍的典型性能需求的案例，假设基于理论分析之后，我们筛选出了四个待分析的数据库，分别为 ClickHouse、 阿里云 ADB、 Redshift、 Greenplum（详细的筛选过程与业务相关性比较大，就不做深入介绍了，但使用的方法就是基于上述的手段），那我们接下来就可以针对这四个数据库做更深度的性能评估分析。

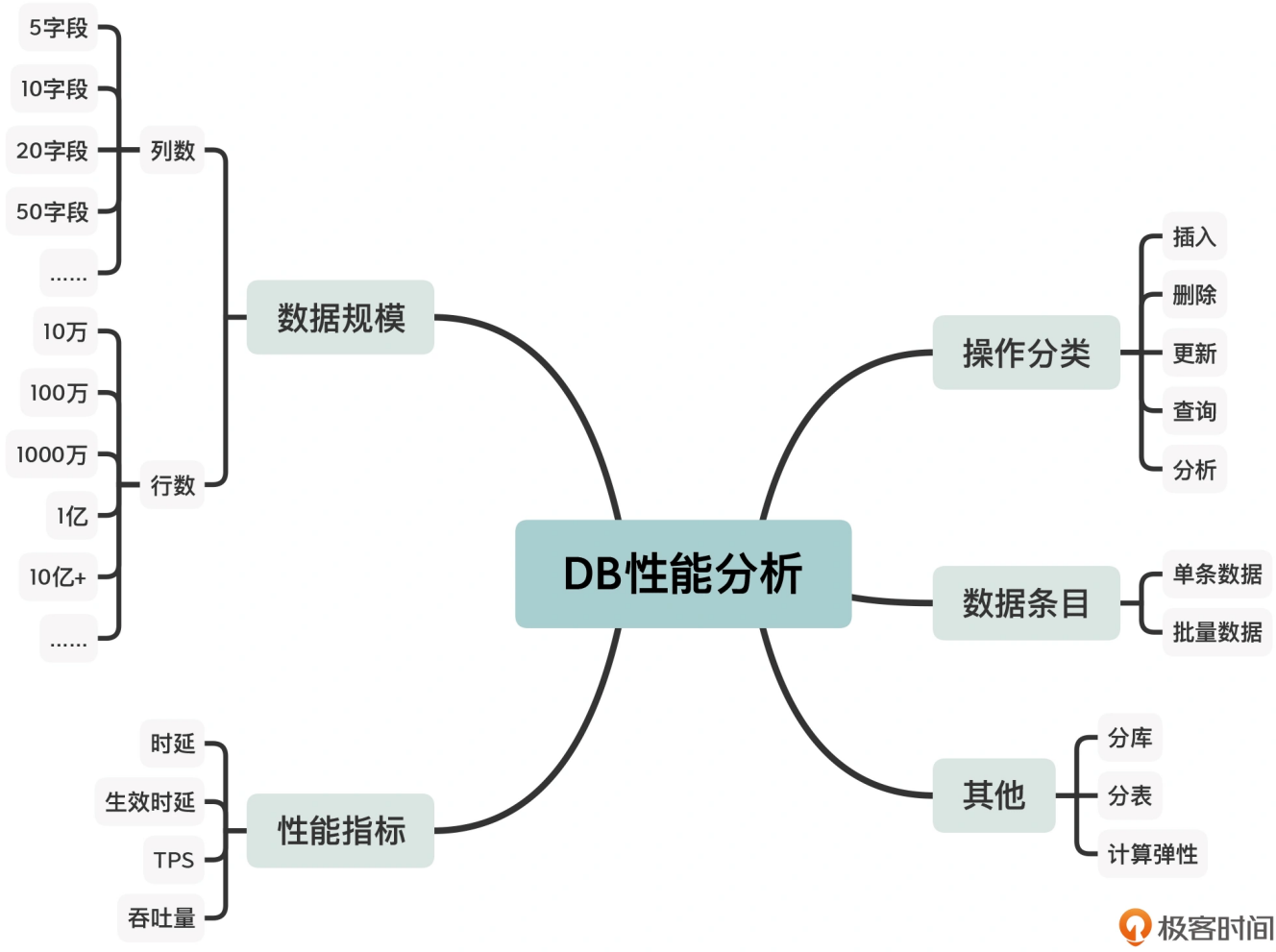
挖掘全量的性能需求点

对数据库的操作除了查询、分析操作外，还有插入、删除、更新等。因此在对数据库进行性能评估分析时，我们还需要考虑到各个性能维度，避免因为漏掉了个别的性能需求造成最终的评估结果无效。

可遗憾的是，在很多的产品业务中，因为性能需求考虑不完整，造成的性能评估结果不可靠的问题经常发生。比如，我前面介绍的性能优化案例中，原来的解决方案是引入 Elasticsearch 来优化数据库的查询性能，但这种方案会导致很多业务的复杂度增加，而且它只能解决一小部分现网的性能问题。

所以，为了避免因性能需求考虑不完整，造成性能评估结果不可靠的事情发生，在进行深度性能评估前，挖掘出全量的性能需求点就显得尤为重要了。下面我就根据前面的性能案例，给你介绍下挖掘全量的性能需求点的过程。

首先针对案例中的性能需求，我们可以做进一步细化，会扩展出很多的性能场景，如下面的思维脑图所示：



从图上我们可以发现，各个树干上的叶子节点之间其实是可以组合的，比如说：

分库、1000 万条、20 列字段、插入、单条数据、时延；

分库、1000 万条、20 列字段、插入、单条数据、TPS；

分表、1000 万条、20 列字段、插入、单条数据、生效时延；

……

这样组合之后能生成很多种性能场景，因此你需要针对实际业务的性能需求进行一些等价类划分，分析出一些比较关键的场景，再进行接下来的性能评估分析。

那么在这个性能案例场景中，我们刚开始仅挖掘出了最核心且具有挑战性的性能需求，但产品对数据库的性能需求是包含很多维度的，所以我们可以进一步完善对插入、更新、删除的性能需求之后，再额外扩充一些具体的性能需求，如下表所示：

插入性能需求	更新性能需求	删除性能需求
单条数据插入：20个字段	单条数据更新：20个字段	/
生效时延要求：实时生效	生效时延要求：实时生效	
TPS: 500/s	TPS: 50/s	

此外，我们还可以根据这里扩展后的具体性能需求，基于理论评估分析再进一步地筛选待分析数据库，那么最终需要基于深度性能评估分析的数据库还能够再减少。

深度性能评估分析并发掘潜在性能冲突

到这里，我们已经完成了备选数据库的理论筛选过程，接下来就需要进行深度性能评估分析，也就是性能测试分析的过程。具体的性能测试理论与方法，我会在“性能看护”这个模块给你详细讲解，这里我们只需要关注下性能测试中比较重要的几点：

1. 测试数据要尽可能接近真实数据；
2. 尽量自动化执行；
3. 尽量代码脚本化执行。

在明确了性能测试的重点之后，还需要尽量获取真实的被测数据集。假设我们通过在现网中采集或构造仿真数据，创建了一个待评测数据集：

数据规模	字段数目
1亿条数据	20个字段

基于这个待评测数据集和一致化的业务分析请求，我们对上述四个数据库进行性能测试的对比分析，获取真实的性能评测数据如下：

查询和分析性能	
数据库	时延
ClickHouse	50~100ms
阿里云ADB	300~500ms
Greenplum	500~800ms
Redshift	1s~2s

从结果可以看出，在真实领域的业务场景下，ClickHouse 的时延仅为 50~100ms，有着非常卓越的性能。这样，在基于真实数据的性能评测之后，你会发现获取的性能对比结果与在网上获取的信息是不完全一致的，这是因为你的性能评估数据已经携带了领域数据特征，其分析请求也是与领域特性相关的。

此外，在这个阶段所做的性能评估分析，**还有一个重要的目标是发掘潜在的性能冲突点**。针对案例扩展出来的性能需求点，经过评估测试后我们会发现部分数据库产品在一些性能场景下，并不能满足需求，比如说：

Redshift	ClickHouse	阿里云ADB
单条数据插入TPS< 10TPS/s	单条数据插入TPS< 50TPS/s	插入数据的生效时延 5s~10s
数据插入时延 1~2s	/	/

如果你在数据库选型阶段忽略了这些性能冲突点，带来的后果往往是致命的，因为这可能会导致产品上线交付后，浪费很大的修改解决成本，甚至可能导致产品最终失败。

所以，如果没有百分百满足你所有性能需求的数据库产品，可以考虑通过软件设计方案去权衡解决发掘出的性能冲突点，不过这个方法只是在数据库选型不理想的场景下的一种弥补方法，希望你不会用到。

基于性能的软件设计权衡

幸运的话，你可以找到各个方面都能满足性能需求的数据库产品，然而实际中面对的业务通常都是复杂多变的，而且如今在数百种数据库不断迭代更新的场景下，确实存在找不到那个性能完美匹配的数据库的情况。

那么面对这个问题，有没有什么好的解决办法呢？就我的经验来看，当出现性能冲突点时，我们可以在软件设计层面，考虑一些折中的手段来规避和解决问题，这里我给你举几个例子。

假设受制于数据库的单条数据插入 TPS 的上限为 10 条，而产品业务每秒插入的数据是 500 条，那我们是不是就没有办法选择这款数据库产品了？

并不是，针对这个场景，我们可以通过批处理模式（即将很多个单次处理操作整合起来，一次性执行处理的性能优化模式，我在第 11 讲中会给你详细介绍），将每秒内的 500 条数据整合到一起插入（注意，在使用这个性能模式时，我们还需要构建单独的业务代码逻辑来实现这个功能）。

不过，这种实现方法也可能会带来一些潜在的问题，比如说：

1. 插入数据的潜在生效时延被人为拉长了；
2. 插入数据在整合批量插入的间隔时间内，又发生了变更，需要开发复杂的逻辑来处理这种场景。

所以你需要明白的是，**软件设计上的权衡是有利弊的**，在通过提升某些操作上的性能来解决性能冲突的同时，势必会在某些点上引起性能的下降。这个时候，你还需要评估设计权衡导致的这些性能下降点是否可以被产品业务所接受，才能确定该设计权衡的可行性。

我再举个例子，**有些 OLAP 分析数据库不提供删除操作，但当产品业务需求有删除操作时，是否有其他变通实现的手段呢？**答案也是有的，我们可以将删除操作转换为对数据条目的状态位更新操作，然后通过状态位标记软删除来实现业务中删除能力。

再比如说，**MongoDB 数据库不提供跨文档操作的事务一致性机制，但在产品业务中存在这样的事务性要求时，要怎么解决呢？**如果业务中存在的事务逻辑的切换频率非常低，其实有些逻辑是可以通过加锁来实现的，或者是可以通过实现事务机制来支撑。

以上我举的这几个例子，其实并不能完全列举出，各种软件设计与实现手法来规避数据库性能冲突的权衡手段，因为针对每个数据库产品或多或少都能找到这样一些规避手段。

总而言之，你在实际的业务场景下，如果碰到了数据库的性能冲突时，要知道还可以通过软件设计实现权衡来解决的思路与方法，以此帮助你更好地实现系统极限的性能需求。

小结

互联网时代是一个不断制造数据的时代，各行各业的产品都具有特色的数据分析需求，而最终产品的性能表现都会承载在数据库上来实现。然而，在对数据库选型时，我们其实很难完全剥离功能来谈性能，而是需要基于功能与性能一起综合决定。

完整且系统地介绍所有数据库的功能与性能，是一个非常庞大的工程，所以今天我是**站在性能的立场**去给你讲解数据库选型的过程和方法，以及过程中需要注意的一些事项，希望你能够掌握基于性能的数据库选型中的理论筛选、深度性能评估、软件设计权衡的思路与方法，从而能够在产品性能设计与优化中选择一款合适的数据库。

思考题

在你的业务场景中，有没有哪个性能瓶颈是受制于数据库且还没有很好的解决办法的？学完今天的内容，你觉得可以怎样解决呢？欢迎给我留言，分享你的思考和看法。如果觉得有收获，也欢迎你把今天的内容分享给更多的朋友。

上一篇 06 | 通信设计：请不要让消息通信拖垮了系统的整体性能

下一篇 08 | 可监控设计：如何利用eBPF来指导可监控设计？

精选留言 (1)

写留言

li3huo

2021-06-02

单纯读或写的性能提升可以采取增加 buffer 把多次写入合并为单次/把频繁读的公共内容写入 cache 的方式来提升吞吐量

展开

作者回复: 哈哈，你总结的很精炼！