复习课（七）| Dremel

2021-12-08 黄金

《大数据经典论文解读》 课程介绍 >

讲述：王惠

时长 10:11 大小 9.33M

你好，我是黄金。今天，我们一起来复习下 Dremel 的论文内容。

Dremel 介绍

Dremel 是一种可伸缩的、交互式即席查询系统，主要用于分析只读的嵌套数据集。只要几秒钟，它就能从万亿条记录中得到想要的聚合查询结果。

我们知道，Web 和科学计算中使用的数据常常是非关系型的，一般采用支持灵活扩展，可以不断嵌套的方式来表示。而 Dremel 为了支持这类数据的低延迟即席查询，提出了一嵌套数据的列式存储方案，这不仅减少了需要扫描的数据，还因为更廉价的压缩方式，降低了 CPU 的消耗。

并且，Dremel 还从搜索服务中借鉴了查询执行树的思想，像分布式搜索引擎查询数据一样，查询请求会通过树状结构下推到子节点，然后经过层层归并，返回最终结果。这种分而治之、并行计算的思路，就让 Dremel 降低到秒级延迟成为了可能。

那么，Dremel 论文主要介绍的，就是**嵌套数据的列式存储方案**和**多层查询执行树**。下面我们就一起来回顾下这篇论文的主要内容。

Google 工程师如何使用 Dremel？

我在刚开始读 Dremel 论文的时候，一直有一个疑问，Dremel 说自己是 MapReduce 的一个补充。但是我就想，MapReduce 分析数据要几个小时，Dremel 只要几秒钟，这分明是巨大的进步，怎么能说只是补充呢？

不过，当我读完论文的第 2 节 BACKGROUND，里面介绍了 Google 工程师如何分析数据，我才明白 Dremel 在数据生态管理中的地位。

让我们来认识一位叫 Alice 的 Google 工程师，有一天她有个新想法，想从网页中提取一种新信号。她使用 Dremel 执行了几个交互式命令，几秒钟之后她得到了查询结果，接着她又执行了几个其他的命令，来确保她的想法是正确的。这些查询过程需要快速返回结果，以便 Alice 能不断验证想法，调整思路。

不过，Dremel 返回的结果并不精确，也无法执行非常复杂的分析计算，要得到更精准的分析结果，Alice 还需要编写 FlumeJava 分析程序。这样，一旦 Alice 分析完所有问题，她就可以编写一条可以达到提取新信号目的的 SQL 查询语句，处理一直在新增的流式数据，并通过看板展示查询结果。最后，她可以把这份新数据注册到数据系统中，供其他工程师使用。

那么，通过了解 Alice 的工作，我们可以看到 Dremel 在数据分析中的作用，**它能帮助分析师快速验证想法，得出大致结论。但是想要获得精确的结果，或者分析实时的流式数据，还是需要其他技术的支持。**

嵌套数据的列式存储方案

接下来，就让我们认识下 Dremel 表示嵌套数据的列式存储方案。

列式存储就是把数据按列存储，使得程序在读取数据时，可以只加载指定列数据，而不用加载整行数据。不过，对于嵌套数据格式而言，采用列式存储之后的**麻烦在于，怎么还原回原始数据**。

比如，某个字段 A 是集合类型，它的元素是结构体，结构体的字段 B 允许为空，那么采用列式存储展平字段 A.B 以后，我们怎么知道一条原始记录，包含字段 A.B 的多少条记录呢。并且，如果字段 A 是其他嵌套字段的子字段，那问题就变得更复杂了。

Dremel 的解决方法，是为每个字段的每个值增加 Repetition Level 和 Definition Level，**让每个字段的值自描述所属的嵌套结构信息**，借助这些信息，Dremel 就能还原原始数据结构。

这里的 **Repetition Level，是指值所属的字段路径，第几个可重复字段最近发生了重复**。这个定义听起来有点绕，让我们来看个论文中的例子。

Figure 2 中定义了 Document 对象的 Schema，对于字段 Name.Language.Code 而言，字段路径指的是 A.B.C 这种形式，其中 Name 和 Language 是可重复字段，所以 Name.Language.Code 的值的 Repetition Level 是 0-2，0 比较特殊，它代表了一条原始记录的开始，之后的值的 Repetition Level 都会大于 0。

比如，值 en 所在的位置，最近发生重复的是字段 Language，所以它的 Repetition Level 为 2。值 en-gb 所在的位置，最近发生重复的是字段 Name，所以它的 Repetition Level 为 1。

而 **Definition Level 是指值所属的字段路径，那些本可以不定义的子字段，有多少个出现在记录中，其中可以不定义的子字段包括可重复字段和可选字段**。这个定义听起来更绕，我们还是回到 Figure 2 的例子中。

对字段 Name.Language.Code 而言，Name 和 Language 是可重复字段，所以 Name.Language.Code 的值的 Definition Level 是 1-2。比如，第 3 行的 NULL 所在的位置，只有字段 Name 出现在记录中，所以它的 Definition Level 为 1。而对比字段表 Name.Language.Code 和 Name.Language.Country，我们可以看到前者如果值不为空，Definition Level 为 2，后者如果值不为空，Definition Level 为 3，这是因为 Code

是必须字段，不属于可以不定义的字段，而 Country 是可选字段，属于可以不定义的字段。

另外，在论文的第 4.3 节 Record Assembly，还描述了如何根据 Repetition Level 装配原始记录。Figure 4 给出了一个状态机，使得程序可以从字段 DocId 出发，根据值的下一个 Repetition Level，得知要读取哪一个字段，从而完成整个记录的读取。Figure 5 给出了另一个状态机，只包含字段 DocId 和字段 Name.Language.Country，说明了如何根据部分字段装配记录。

多层查询执行树

实际上，如果让 MapReduce 在列式存储上工作，查询速度也可以提升一个数量级。在论文的 Figure 10 中，就对比了 MapReduce-on-records 和 MapReduce-on-columns 的执行时间，执行相同的查询语句，MapReduce-on-records 需要读取 87TB 的数据，执行时间需要几个小时，而 MapReduce-on-columns 只需要读取 0.5TB 的数据，执行时间缩短到几分钟。

不过，Dremel 的查询速度会更进一步，又提升了一个数量级，它执行相同的查询语句，执行时间只需要几秒钟。

Dremel 在执行查询的时候，采用了**树状结构，分层聚合查询结果**。被查询的数据通常以**分片**的形式存储在分布式存储系统中，Dremel 会把查询计划分解到每个分片上执行，然后把查询结果聚合到一起，返回给客户端。

在这里，负责接受客户端请求，分解最初的查询计划，聚合最终的查询结果，并响应客户端的是**根服务器**；负责进一步分解查询计划，聚合中间结果的是**中间服务器**；负责具体执行查询计划的是**叶子服务器**。

其中，中间服务器可以有很多层。徐老师在第 17 讲中详细分析了一个用例，说明增加中间服务器可以显著减少查询执行时间。**中间服务器越多，并行查询和聚合的程度越高，查询速度就越快。**不过，随着中间服务器层数增加，它们之间的网络传输开销会不会也被成倍放大呢？

得益于分析查询的特点，每层中间服务器聚合之后，返回的数据量都在减小，所以实际上，传输数据的网络开销并不大。

小结

总结一下，我们今天一起复习总结了 Dremel 如此之快的两个原因：第一，Dremel 支持嵌套数据的列式存储，目的是减少查询需要读取的数据；第二，Dremel 采用树状结构，分层执行查询，目的是通过使用并行计算来加速查询。

不过，要把查询延迟降低到秒级，还是需要做很多工作的，Dremel 在 2020 年的回顾性论文中，总结了为降低延迟所做的其他工作，包括：

服务池化，避免类似 MapReduce 任务的冷启动时间；

投机执行，慢的服务器执行任务少一点，快的服务器执行任务多一点；

面向列的模式表示，只需要解析查询列的模式；

使用轻量级压缩平衡 CPU 和 IO；

近似结果，如果允许处理完 98% 的数据之后就返回查询结果，可以让延迟降低 2~3 倍；

查询延迟层，允许小查询快速返回，不会被大查询阻塞；

重用对文件的操作，比如一个作业的子任务读取了某个文件的元信息，其他子任务可以共享这个元信息，而不用重复读取；

保证容量，为重要的作业保证容量，不会被其他作业抢占；

自适应的查询缩放，可以根据查询语句动态构建多层查询执行树，自动识别应该用 2 层、3 层，还是更多中间层。

好了，咱们下一期复习课再见。

上一篇　复习课（六）| Hive

下一篇　25 | 从S4到Storm（一）：当分布式遇上实时计算

精选留言

写留言

> 由作者筛选后的优质留言将会公开显示，欢迎踊跃留言。