云计算第九次实践报告

MF1332057 唐毅明， MF1332086 张磊， MF1332037 李辉

1 任务描述

本次实践的主要任务是设计实验来测试传统平台版本和云平台版本系统的差异。具体内容如下：

1. 扩展字典，按照实践8的要求，产生100万条文献记录。
2. 在非云平台版本和云平台版本的系统上，对比测试以下内容:
    1. 记录的插入。
    2. 倒排的建立。
    3. 检索的平均效率。
    4. 信息统计。
    5. 附件的写入和读取。
3. 根据内容2，自己设计其它测试内容。
4. 总结在两种平台上构建系统的异同、向云平台迁移过程中的体会，以及两种平台上所构建系统的差异。

2 测试

按照要求，我们修改了原先数据生成器的参数，生成了一个拥有100万条文献记录的文本文件。为方便起见每一行为一条文献记录。每个字段以 `|` 分隔。

2.1 记录的插入

对数据库做批量的插入操作有很多种选择。最简单也最容易想到方法是：

1. 从原始数据文件里读取每一行记录。
2. 利用分隔符解析记录里的每个字段。
3. 执行insert语句。
4. 如此执行上述步骤100万次。

如果数据集很小，那么上述操作会很快完成。但对于100万这样具有一定规模的数据集，速度就会非常慢。一方面，一行一行读数据如果buffer不够大，那么在读上会有开销。另一方面，一般情况下，直接用JDBC的插入API，默认实现不会把多个插入放到一个事务里做优化。这样导致插入100万条数据的时间完全不能忍。目测将近1小时（直接强制杀死了测试进程）。

一种优化方法是用batch API。这样100万条数据插入时间可以大幅减少。如果怕单次数据太大而程序崩溃，可以把100万条数据分隔成很多小块来处理。此外，你还可以用JDBC对SQL做预处理的API，这也能大幅提高效率。最终的结果是，单文献这张表插入100万条数据用机房电脑执行时间大约在150秒。

还有一点可以说的是，如果仅仅为了批量导入数据，对MySQL而言，有 `load data` 命令。它的效率会比 `insert` 高很多。究其根源主要是 MySQL 内部对于 `load` 和 `insert` 的处理机制不同。`load` 的处理机制是：在执行 `load` 之前，会关掉索引，当 `load` 全部执行完成后，再重新创建索引。`insert` 的处理机制是：每插入一条则更新一次数据库，更新一次索引。`load` 与 `insert` 的不同还体现在 `load` 省去了 SQL 语句解析，SQL引擎处理，而是直接生成数据块。

对HBase而言，我们对插入没有做什么优化，最后很轻松地在60秒以内把数据插完了。

通过对MySQL和HBase的记录插入测试，我们可以很明显地感觉到用HBase处理大批量数据插入可谓得心应手。MySQL如若不注意优化，很容易成为绊脚石。

2.2 倒排的建立

云平台下的倒排索引的实现可参见第七次实践。这里主要讲一下非云平台版本的实现。相比云平台下用MapReduce来做倒排，传统平台下我们不得不自己设计算法并实现。

我们用了HashMap来遍历数据库所有条目来生成倒排文件。简单地来说，就是分词后把单词做hash map的key，而条目id的集合为value。最后一次遍历就可以得到倒排文件，且仅仅花了18秒。相比而言云平台下建立倒排文件花费了至少220秒。

这样看来，通过算法优化，似乎传统平台下更快。但是，我们没有考虑到字典非常非常大，条目非常非常多的情况。在这种极端情况下，传统平台很容易会因为性能问题直接崩溃或者效率非常低下。云平台此时可以发挥强大的计算能力，把传统平台比下去。因此，简单的实验还不能得出云平台版本系统完全落败的结论。

2.3 检索的平均效率

从100万条数据数据做检索，非云平台版本平均花费时间为1.4s，而云平台版本在460ms。云平台版本的倒排索引使得它在这轮测试里胜出。我们也考虑一个问题：如果当数据库里的所有条目的标题和作者的长度特别短时，倒排索引就会太占优势了。因为我们用MySQL在标题和作者字段上非别建立缩影后，用 `like` 之类的条件做查询速度也是在1s内。考虑到倒排缩影占去的空间和维护成本，这时候就有点得不偿失了。倒排缩影还是比较适合大规模数据的情况。特别是做全文搜索时，显得非常有用。当然，实验只考虑了英文这种容易做分词的语言，若换成中文之类的语言，那结果可能又有所不同。

2.4 信息统计

我们在传统平台下直接用 `select ... group by userId` 从100万条数据里统计一周内的评论，平均时间在16s。而云平台花费了51s。

2.5 附件的写入和读取

在非云平台下就简单的放到某个目录里，而云平台下会复杂许多。对于附件，我们也没有太多的经验。非云平台下的实现是将用户上传的附件保存在普通的文件系统上的某个目录里。而云平台版本则是将文件转存到HDFS下。从小文件的存储速度来看，两者差距并不大，一定程度上HDFS还慢一点。由于我们给系统上传文件的体积做过限制，所以从需求角度讲，并没有用HDFS的必要。

当然，云计算的拥护者们会列举出一大堆理由来说云计算的好，但所有这些前提是，我目前真的有必要用吗？大数据，高并发，高可用性这些词已经被用烂了，当有必要转向云平台时，人们必然会去的，不用你在后面推着。

2.6 其他测试内容

之前我们一直把注意力集中在插入和查询，但我们竟然没有考虑删除！因此，我们尝试对删除做一下测试。我们发现在传统平台版本系统下，删除一个条目非常容易，用 `delete` 就行，而且速度非常快在50ms以内。相比而言云平台版本就显得有些复杂，你得去取得id的集合，然后再把对应的条目删除。

3 比较、体会和总结

3.1 两种平台上构建系统的异同

先说相同点：

- 都是以数据为中心，对数据做操作；
- 对外表现的功能是一致的；
- 前台的事务逻辑是一致的。

也就是说实现对外的逻辑部分是相同的。无法避免的都要实现各种功能，而且很大程度上这部分的代码是可以复用的。

从课程实践来看，在两种平台上构建系统的主要区别在于数据层的实现。传统平台版本的数据层基于传统的关系型数据库管理系统MySQL，附件存储也是基于一般的文件系统，并且这一切都是单机的。而云平台版本的系统的数据层是NoSQL的，而且是分布式的。后台数据层的不同，导致相应的针对数据读写的实现也会有所不同。

这些异同，意味着两种版本系统不同的适用范围。数据量不大，对拓展性和性能要求也不苛刻时，显然传统版本系统更加适用。而当系统异常庞大，对规模和性能追求很高时，后者会发挥出强大的威力。

3.2 向云平台迁移过程中的体会

向云平台迁移最大的挑战就是数据层的修改。主要有以下事情：

- 原有数据的迁移

- 原有程序和新数据库的对接

因为NoSQL的引入，因此原有的关系型数据库的设计得推掉重新建立表。已有数据则要做迁移。如果是一个真实的上线项目，这会很困难，非常容易出错，一个不小心就会将整个站点置于不利的境地。课程实践的仅仅是“玩具项目”，且数据库结构没那么复杂，因此这一步还没有深刻体会。

让原有程序和新数据库对接倒是非常困难。整个数据层几乎是重写。还在事先做了MVC的架构，因此只是model模块改动比较大。学到的一课是，一定要设计一套稳定的接口，并且依赖接口。这样才能最大限度地保证灵活性！这一点我们做的不是太好。

3.3 其他

一些吐槽。从我个人角度来说，课程实践在前期做的传统平台软件在一定程度上没有体验出应有的意义：

1. 从技术上来说和云计算并没有多少关系。
2. 传统软件平台开发很费时间。
3. 这作业让我从0开始边学边用J2EE，浩然姐姐给本科生的J2EE课程要哭了。
4. 后来前面的东西抛掉了？反正没发现有多紧密。感觉就是被遗弃了。

总的来说，与其这样不伦不类。不如就专注云计算相关技术的实践。直入主题，刀刀入肉，从头到尾就做和云计算相关的实验。反正，我觉得后半段能学到的东西比我用J2EE实现一个玩具系统有意义多了。当然，这也是学生的个人喜好，比如我就偏执地反感实现xxx系统，讨厌J2EE这一套。因此上述观点有点主观啦。希望这门课越来越好！