今天，好像什么也没干。。。

先补充昨晚做的个测试，之前和一个前辈争论过一个问题，几十G大小的文件，每行是一个字符串统计最高频的词，我说按服务器内存尽可能多的塞进内存后建堆，前辈说其实都不用占什么内存只需要一行一行读再进行操作就行，但是我直觉上觉得会更慢，就说这样做可能io会慢，连续读取内存可能会比不断io了再内存操作然后又io这样循环强，前辈说都是读那么多的东西io的时间是一样的，所以操作的时间应该是相等且不占内存。我想了想就没争了，决定回去写个demo看看。

然后一直忘记这事，昨晚下班时想起来就读取了30M的两个文件测试（同时读两个是以防磁盘有缓存干扰结果），中途想到因为计算机有很多进程可能在不停io所以磁头会移来移去，这样的话一行一行读肯定会更慢，但是突然又想起来连续读一个大文件的话也不一定是能连续读这个文件，其他进程的读写可能也会在中间移动磁头，而且文件也不定是在一块连续的空间上，这点就算扯平把。

因为没有一样大的文件，所以我分步读取了较小的那个文件，然后按小文件的大小一次性读取了较大文件中的那么多个字节。第一次实验我极端地按一字节分布读，然后一次性读了大文件，发现时间上分步读差了很多，毫秒级的话差了好几个数量级，正当我自鸣得意地时候，学长说把分步读地内存再调大一些，然后慢慢加大，发现分步读的时间降得好快，到簇大小4K的时候时间就持平了，而到了8K的时候分步读大部分时间下还更快？我这个就非常郁闷了，后来分析，可能是因为指令条数分部读的指令会多出几个数量级，分步地越小差的越多，当分步读10字节时，指令条数差了100.24\*1024\*30条，300万行条代码，编译出来近亿级别？大概占100ms？但是差距不止100ms这个量级，应该还是涉及到io问题。连续读大文件更慢的原因可能是无法找到连续那么大地一片空间，需要挪动内存。不过才30M也不至于这样，如果再大的话这个影响可能明显一些。

但是今天想起来了缓存的影响，今天又做了一次实验，分步读8K每次和一次性读完，分步读需要1788ms而一次性读的时间为746ms，然而再次读的时候两者就都快了不少，都变成了2，30ms，说明之前的实验结果还是严重被缓存干扰了，并不只是分别读两个文件，而连续读也会对缓存造成影响。

这是第三天的测试，从学长那里拿到一个测试工具，按不同读取大小的文件进行读取，磁盘的io瓶颈在16K左右，低的0.5K时读取文件的速度超级慢，所以如果是按行读取，速度应该是慢的。接下来要研究的是，为什么会是这样。

今天干了啥？把原来本里的代码和文档同步到git上，在公司的本上把东西同步过来。然后fix了一下午的eclipse报错，各种错，无穷无尽，最后还是还原了所有操作然后把原来本上的配置文件全copy来设置，一步一步改好设置才部署进tomcat中。总结是，maven用不好就是个坑。