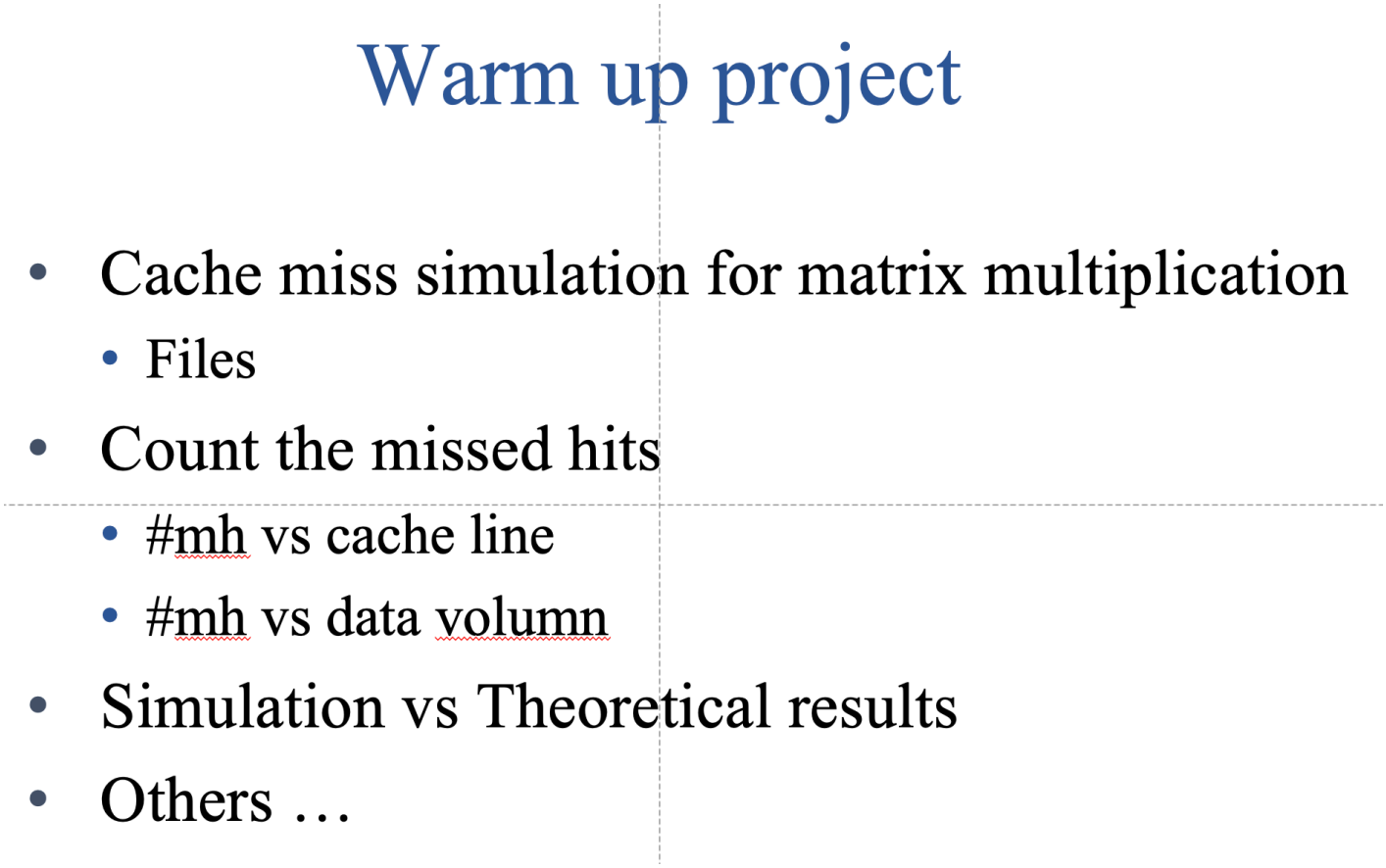
吕妍 | Modified Today

第一个实验：

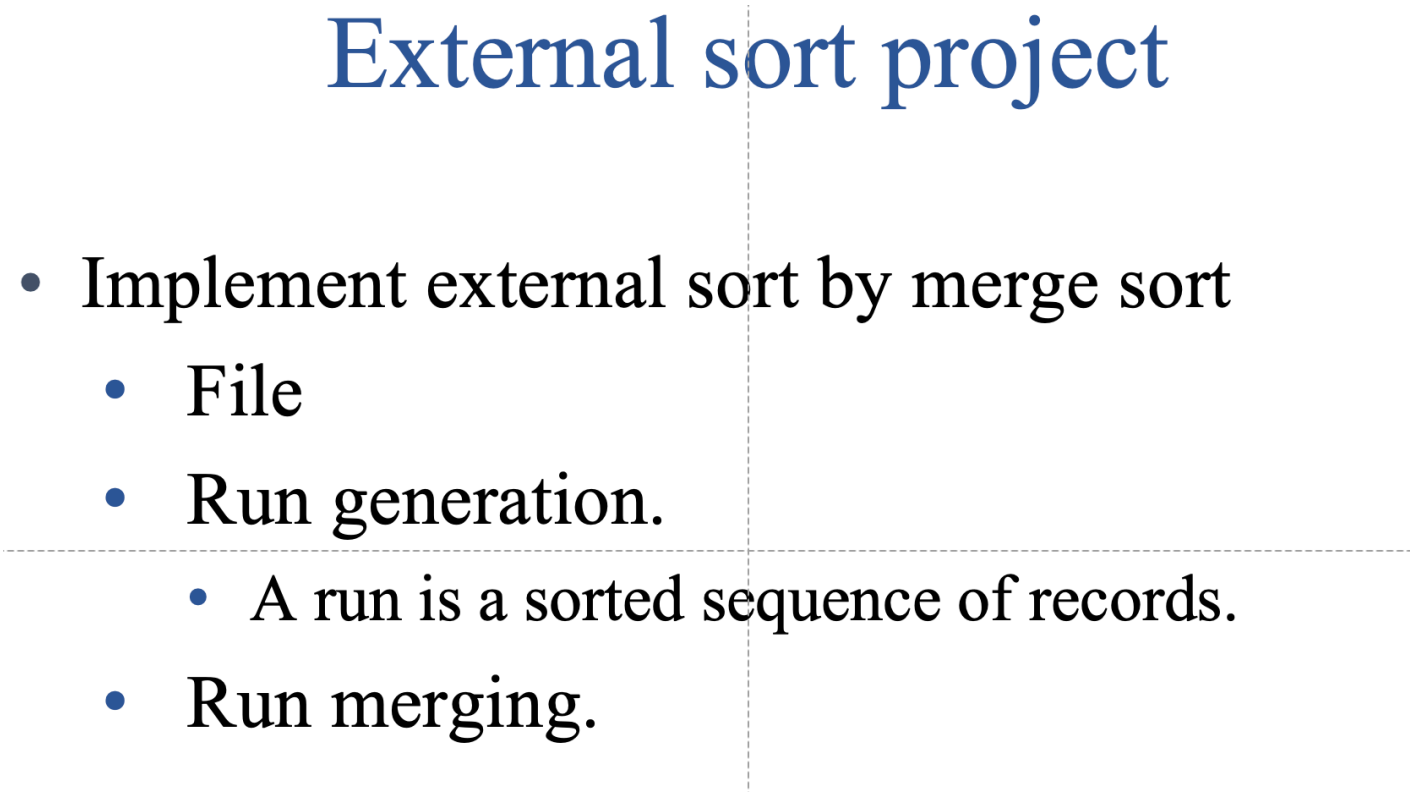


基本说明：

手动写出矩阵相乘的代码，来模拟cache miss 过程，要求：

1. 两个矩阵分别放在两个文件中
2. 尝试在不同大小的cache line情况下、不同矩阵大小情况下，采用不同的矩阵乘法的方式（不同顺序）计数cache miss的发生情况。
3. 将模拟出来（计数）的结果和理论分析出来的结果进行对比分析

实验二：

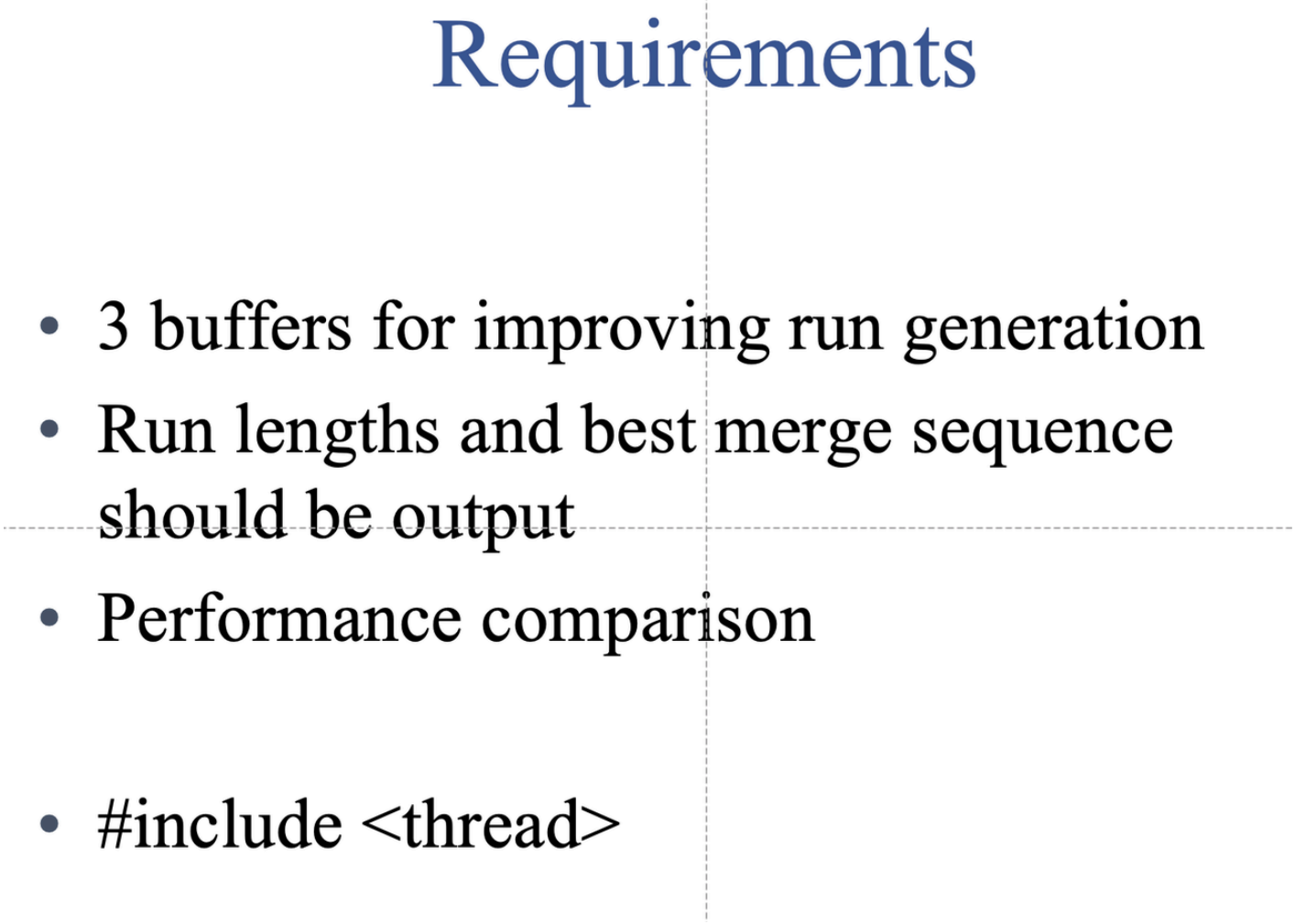


基本说明：

实现一个外部归并排序算法，要求：

1. 待排序 序列 和 排序后的序列都要存放在文件中，不能一次性读入或者写出。需要模拟内存较小但待排序文件很大这种情况
2. 生成顺串（可以用任何内部排序算法，此时，单个顺串大小应该设置为内存可以放的下的），生成的顺串也写入文件（模拟内存装不下所有顺串的情况）
3. 归并顺串（可以用最简单的两两归并）
4. 可以探索
    a. 不同大小的顺串设置、不同的归并路数（不同k）对排序速度的影响。
    b. 也可以记录IO次数。看看外部排序的效率瓶颈在哪里。
    c. 等等

实验三：

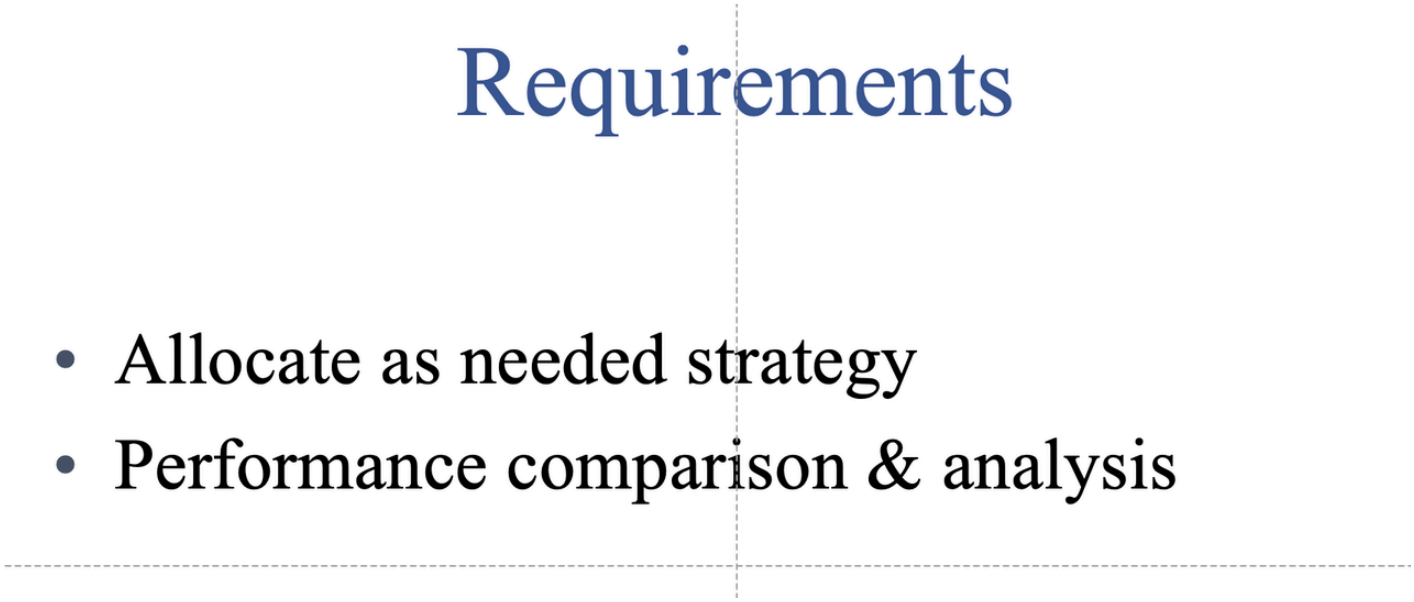


基本说明：

改进实验二的外部归并排序算法，使用loser tree来生成更大的顺串来提高排序效率。要求

- 跟课堂上演示的不同，本次实验要求只使用3个buffers （1个跟loser tree通信，另外两个分别读写）
- 必须要额外输出顺串大小
- 使用哈夫曼树来确定最佳的归并顺序（可以两两归并，也可以k>2），这个归并顺序也要输出出来
- 最终排序效率需要跟实验二的进行对比
- 可以探索：
  - 顺串的个数（例如，平均个数、在最差情况下顺串的个数）
  - 不同的归并顺序对结果的影响，哈夫曼树是否一定是最快的?
  - 等等

实验四：



基本说明：

进一步改进实验三的外部归并排序算法，在使用loser tree来生成更大顺串的同时，探索如何用跟大的k进行归并。要求：

- 使用一个尽可能大的k进行k-way merge
- 具体归并依然使用loser tree进行
- 为每一个顺串分配一个buffer，这个buffer为loser tree提供数据
- 额外一个input buffer挂在最快被消耗完的那一个顺串的buffer后面
- 剩下的k-1个buffer放在pool里面，随用随调。
- 可以探索：
  - 跟实验三进行对比，看看更大的k是否带来性能的提升
  - 不同的k选择进行对比，看看k越大，是否效率就越高
  - 是否还有进一步的优化空间来提升速度
  - 等等