分布式系统作业

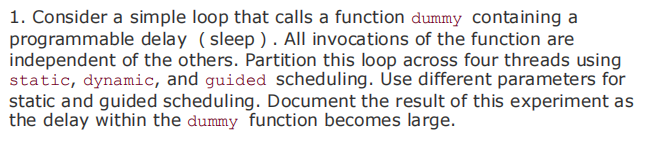
第5次作业

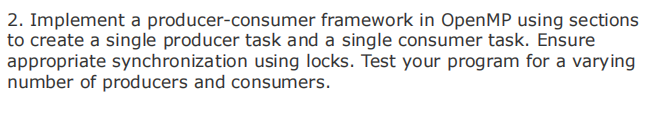
姓名：唐晨轩

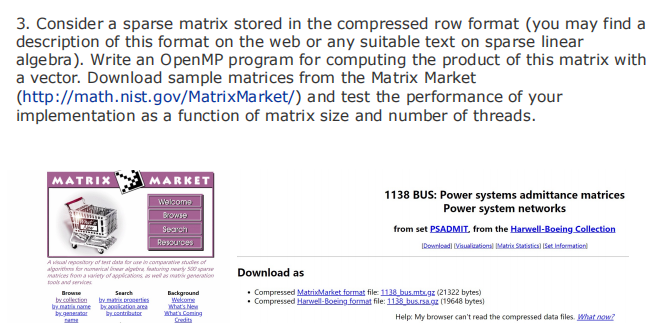
班级：人工智能与大数据

学号：19335182

1. 问题描述

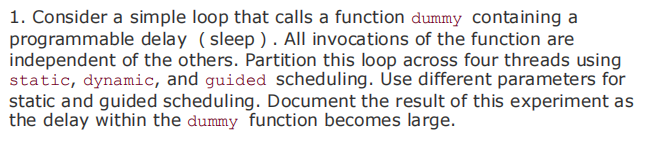






1. 解决方案

1.



具体代码见static.cpp，dynami.cpp和guided.cpp。

大致思路是，设计了一个12次的循环，在循环中调用了dummy造成适当延迟。观察三种调度函数的结果。

其中线程数量设置为4，chunksize设置为1/2。

运行结果如下：

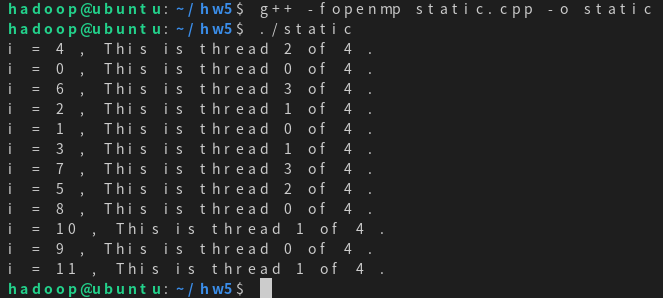
Static调度策略：

**当线程数目设置为4时：**

1. Chunksize = 1



1. Chunksize = 2



经过观察发现：

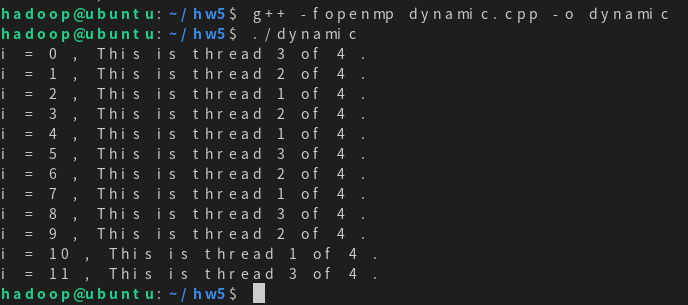
Chunksize=1时：静态调度策略，按轮转的方式，每次把1个循环分配给1个线程，所以i = 0/4/8是线程0完成的，i = 1/5/9是线程1完成的，i = 2/6/10是线程2完成的，i = 3/7/11是线程3完成的。

Chunksize=2时：静态调度策略，按轮转的方式，每次把2个循环非陪给1个线程，所以i = 0/1/8/9是线程0完成的，i = 2/3/10/11是线程1完成的，i = 4/5是线程2完成的，i = 6/7是线程3完成的。

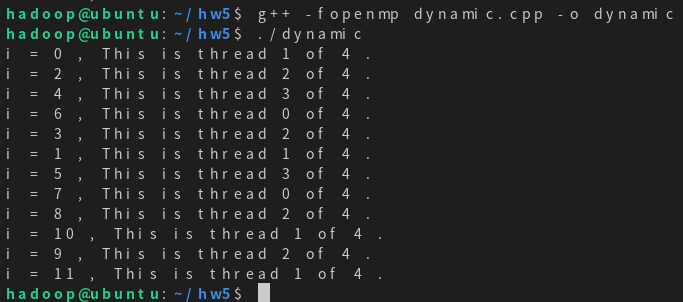
Dynamic调度策略：

当线程数目设置为4时：

1. Chunksize = 1



1. Chunksize = 2



经过观察发现：

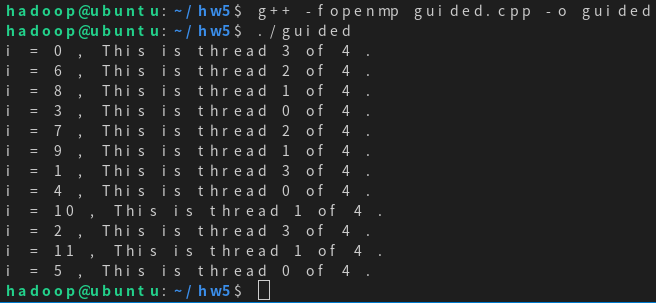
Chunksize=1时：动态调度策略，每次分配1个循环给当前空闲的线程，这种调度策略下的运行结果时不可预测的。

Chunksize=2时：动态调度策略，每次分配2个循环给当前空闲的线程，这种调度策略下的运行结果时不可预测的。

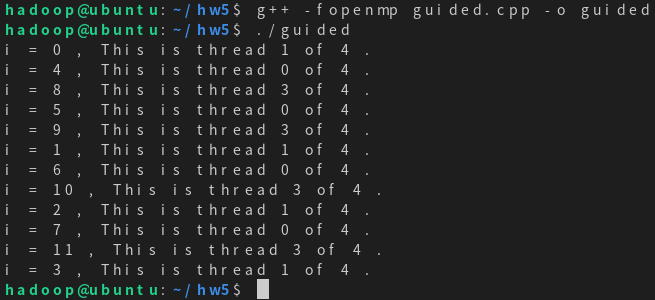
Guided调度策略：（较特殊所以修改了chunksize大小）

当线程数目设置为4时：

1. Chunksize = 2



1. Chunksize = 4

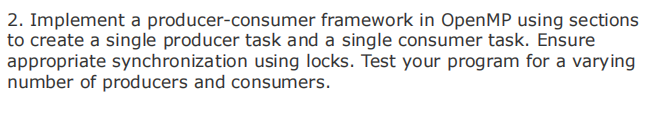


经过观察发现：

Chunksize=2时：guided调度策略，被分配的块的大小就是chunksize（除最后一块），chunksize为2，所以第一个块的大小是2，i = 0/1被分配给了线程3，当块完成后新块（此线程负责的新块）的大小变为原来的一半（1），线程3收到的下一个块就是i = 2，接下来的i = 3/4被分配给了线程0...

Chunksize=4时：guided调度策略，被分配的块的大小就是chunksize（除最后一块），chunksize为4，所以第一个快的大小是4，i = 0/1/2/3被分配给了线程1，i = 4/5/6/7被分配给了线程0，i = 8/9/10/11被分配给了线程3。

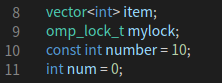
2.



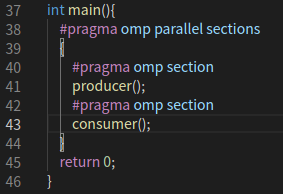
单个生产者单个消费者：

Sleep函数的作用是避免生产过快，导致生产结束了，消费的线程还没有开始创造，最终并行程序呈现出串行程序的效果。

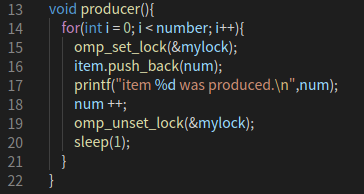
全局变量：



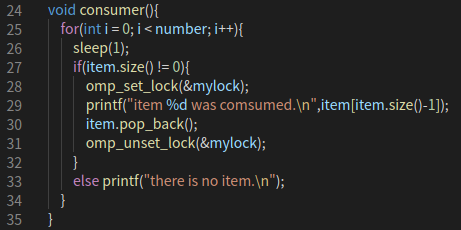
函数调用：



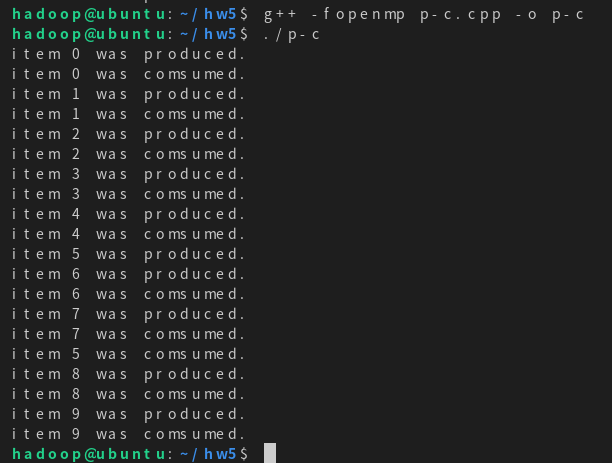
生产者函数：

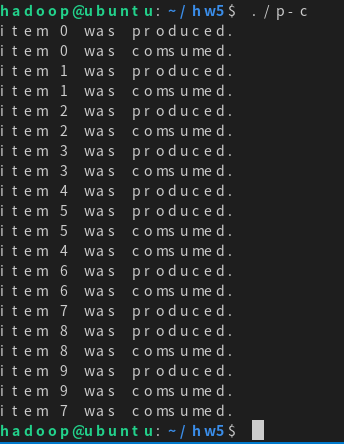


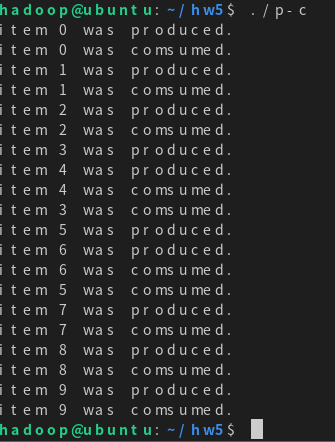
消费者函数：



运行结果：（经测试，不用锁会出现段错误）



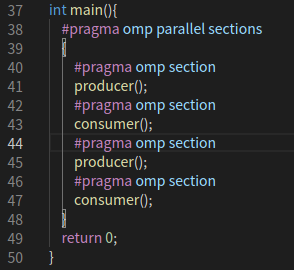




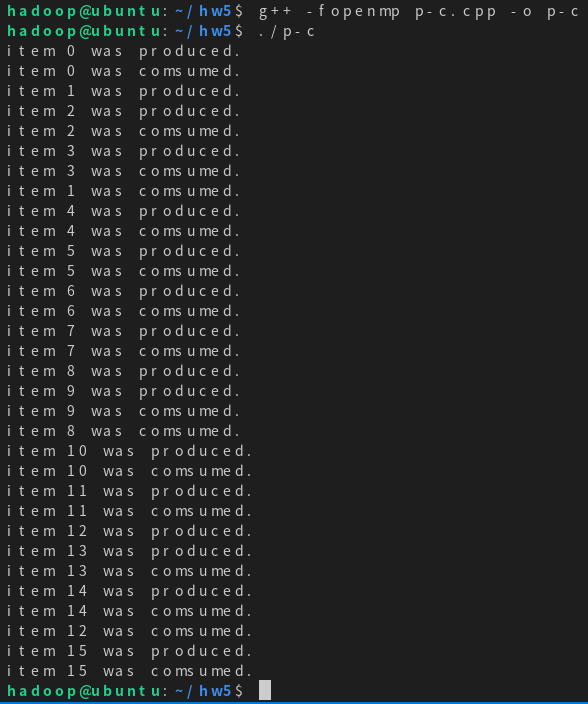
多个生产者多个消费者：

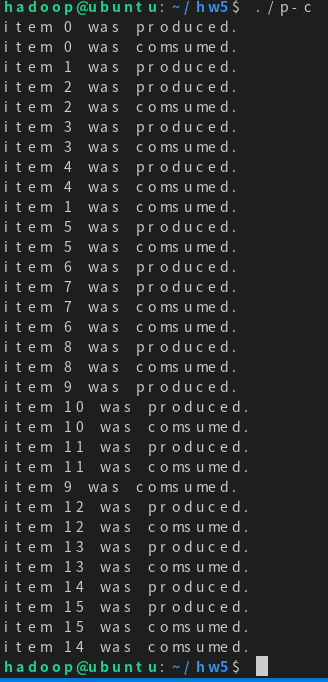
多添加了几个并行任务

函数调用：



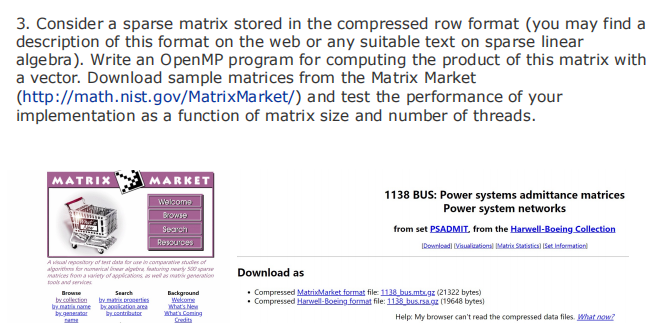
运行结果：（数字太大不好截图，所以我把number从10修改为8）





可以看到，无论是针对单个生产者单个消费者，或者多个生产者多个消费者，程序的执行都没有出错，互斥锁起到了很好的效果。

3.

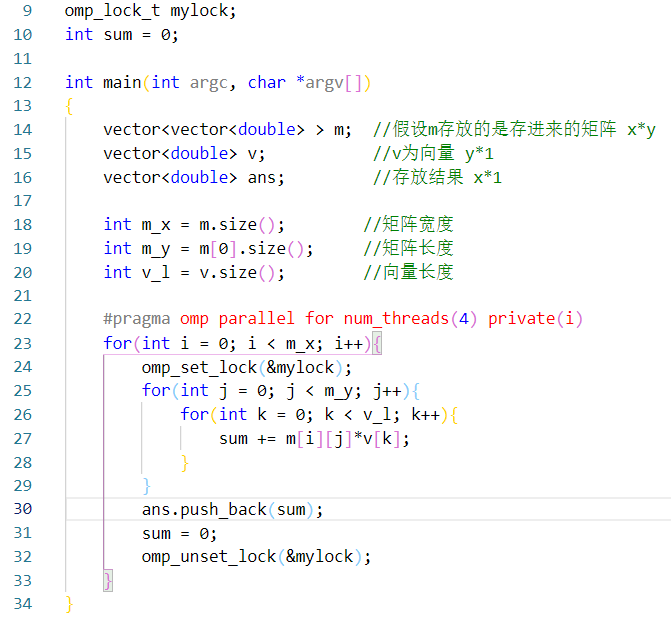


先从Matrix Market网站上下载对应的1138 BUS的mtx.gz文件并且解压为mtx文件：



大致思路为：

先读取1138\_bus.mtx文件，得到矩阵。然后通过openmp并行编程，完成矩阵乘向量的运算。关键代码如下：



遗憾的是我没有完成m矩阵的读入，所以这个作业没有完全完成。