2019 年春季学期并行计算期末考试

Edited by Lyncien

2019.06.10

一、 填空题 10 * 2%

1. 并行计算体系结构有 SIMD-SM/MIMD-SM/SIMD-DM/MIMD-DM，则 PRAM 模型是（1），APRAM 模型是（2），LogP 模型是（3）。
2. 令 W(n)是某并行算法 A 在运行时间 T(n)内所执行的运算量，则 A 使用 p 台处理器可在（4）时间内执行完毕。
3. 对于求最大值的算法，SIMD-EREW 结构上使用 n/2 个处理器可在（5）时间内完成，SIMD-CRCW 结构上使用 n^2 个处理器可在（6）时间内完成。
4. 高斯-赛德尔迭代法五点格式的 A 矩阵是（7）对角矩阵，并行化方法是（8）。
5. OpenMP 属于（9）并行编程模型，MPI 属于（10）并行编程模型

二、 简答题 4 * 5%

1. 解释概念 SIMD，SPMD，SMP，PCAM，Warp
2. MPI 为什么要使用消息标签？
3. 稀疏方程组的求解为什么使用迭代法（如共轭梯度法）而不是直接法（如高斯消元法）？
4. CUDA 中 CPU 与 GPU、线程块内、线程块间同步的方法与代码？

三、 综合题 4 * 15%

1. 阅读代码

```
1  #include <stdio.h>
2  #include <_____>
3  int main ()
4  {
5      int i, n;
6      float a[100], b[100], result;
7      /* Some initializations */
8      n = 100;
9      result = 0.0;
10     for (i=0; i < n; i++)
11     {
12         a[i] = i * 1.0;
13         b[i] = i * 2.0;
14     }
15     pragma omp _____
16     for (i=0; i < n; i++)
17     {
18         pragma omp _____
19         result = _____ + (a[i] * b[i]);
20     }
```

```
21      printf("Final result= %f\n",result);
22  }
```

（1） 补全代码并说明程序的功能

（2） 使用另一种方式实现 15-20 行的求和

2. 给出环上收集（all-to-one）的选路（CT）的算法，作出示意图，分析时间。

3. 对于 PRAM 下求 n 个数前缀和的算法（课本算法 7.9）

（1） 是否是并行成本最优？是 EREW/CREW/CRCW 中的哪一种？

（2） 给出使之并行成本最优改进方法的伪代码，并分析成本最优性。

4. 对于离散傅里叶变换

（1） 给出蝶式 FFT 算法的时间复杂度，可以使用哪种并行算法设计技术使之并行化？

（2） SIMD-BF 上的 FFT 算法，蝶形网络上每个处理器的 w 权因子有两种计算方法，比较分析它们的计算工作量。

个人答案（可能有误，仅供参考）

一、

1.（1）SIMD-SM （2）MIMD-SM （3）MIMD-DM

2.（4）O(W(n)/p+T(n))

3.（5）O(nlogn) （6）O(1)

4.（7）三 （8）红黑着色并行算法

5.（9）共享变量 （10）信息传递

二、

1.SIMD：单指令多数据流

SPMD：单线程多数据流

SMP：对称多处理机

PCMA：设计并行算法的四个阶段：划分(Partitioning)，通讯(Communication)，组合(Agglomeration)，映射(Mapping)的首字母

Warp：CUDA中每个线程块分为若干个组(称为warp)，每个warp包含32个线程，物理上以SIMD方式并行

2. 当发送者连续发送两个相同类型消息给同一个接收者，如果没有消息标签，接收者将无法区分这两个消息。添加标签使得服务进程可以对两个不同的用户进程分别处理，提高灵活性。

3.对于大型、稀疏线性方程组，迭代法比直接法简单、占用存储空间小；对于在有限步内无法得到问题的解时，迭代法可以在有限的迭代步数后，停止运算而得到足够好的近似解。

4. CPU与GPU：如果CPU在接下来的操作中需要用到GPU的计算结果，则CPU必须阻塞等待GPU执行完毕。可在kernel后添加一条同步语句cudaThreadSynchronize ()实现。

线程块内：__syncthreads()只有当同一个块内的所有线程都到达函数__syncthreads()时才会继续往下执行

线程块间：同一个grid中的不同线程块之间不能同步，即CUDA运行时库中没有提供此类函数。 但可以通过终止一个kernel来实现同步

三、

1.（1）

omp.h

parallel for

critical

result

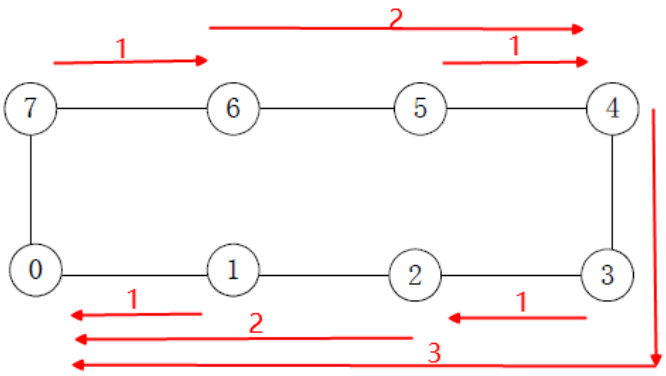
功能：向量点积

（2）并行归约方法

```
#pragma omp parallel for default(shared) private(i) reduction(+:result)
for (i=0; i < n; i++)
{
    result = result + (a[i] * b[i]);
}
```

2.

先发送至距离为 1 的处理器；
发送距离为 2 的处理器；
…
发送距离为 2^i 的处理器



时间：

$$t_{all-to-one}(CT) = \sum_{i=1}^{logp} t_s + 2^{i-1}mt_w + 2^{i-1}t_h = t_s logp + (p-1)(mt_w + t_h)$$

3.（1）不是并行成本最优，并行 c(n) = O(nlogn)，串行为 O(n)
CREW
（2）采用级联技术：先小范围串行后并行
处理器个数 p(n) = n / logn，每个处理器负责 n / (n / logn) = O(logn)
算法分为两阶段
阶段一：每个处理器串行求解 O(logn)个元素的和，花费时间 O(logn)
阶段二：再对 n / logn 个和应用平衡树求前缀和方法，
花费时间 O(log(n / logn)) = O(logn)
总时间为 O(logn)
并行成本 c(n) = p(n)t(n) = O(n/logn)O(logn) = O(n) = 串行成本，是最优

4.（1）T(n) = 2T(n/2) + O(n) => T(n) = O(nlogn)
可以用分治设计技术
（2）方法一：各处理器独立计算权因子
方法二：最后一行先计算权因子，然后各列处理器各自平方后即为上一行的对应列的权因子，logn 步后完成所有计算，除了最后一行，其余处理器权因子计算只需要一次乘法。因此比第一种方法更好。