计算：课后作业，书上原题，PPT原题，3题，10′

OPENMP　MPI　CUDA　补充完整

程序设计：前面OPENMP圆周率题

基本　6道　SMP大型并行计算机系统

并行系统存储访问类型　共享存储

不同处理机　拓扑结构　节点度数　网络直径

常用加速比定理　古斯　阿姆达尔　受串行分量的限制

并行算法的指标

并行算法设计过程

划分通信组合映射

常见并行算法计算模型有哪几个

三种可扩放性度量标准

第7章　划分法，分治，流水线　基本技术

OPENMP　MPI　CUDA特点　使用场合

计算　4.6　4.7　求CPI、MIPS

阿姆达尔加速比计算　串行分量

十台机能加速到多少　极限能加速到多少　4.11

5.2 证明复杂度

Blent定理证明

PSRS均匀划分排序算法　在

步骤写出　结果

程序设计 MPI填空　、MPI基本函数　4.17　4.12

CUDA基本程序设计，两个数加法

CUDA分级内存（CUDA 内存、指令．　函数、线程有编号，

用了几个线程

写个OPEN MP的圆周率给串行写并行

下周四

并行计算

填空

(1) 大型机算机系统一般分为六类
① 单指令流多数据流计算机 SIMP
② 并行向量处理机 PVP
③ 对称多处理机 SMP
④ 大规模并行处理机 MMP
⑤ 工作站机群 COW
⑥ 分布共享存储多处理机 DSM

(2) 并行计算机访存类型
① 均匀存储访问 (UMA)
② 非均匀存储访问 (NUMA)
③ 全高速缓存存储访问 (COMA)
④ 高速缓存一致性非均匀存储访问 (CC-NUMA)
⑤ 非远程存储访问 (NORMA)

(3) 网络性能指标
① 节点度 — 射入或射出一个节点的边数
② 网络直径 — 网络中任何两个节点之间的最长距离
③ 对剖宽度 — 对分网络各半所必须移去的最少边数
④ 对剖带宽 — 每秒钟内，在最小的对剖平面上通过所有连线的最大字节数。
⑤ 对称 — 从任一节点观看网络都一样。

(4) 不同处理器拓扑结构

网络名称		网络规模 (N个节点)	节点度	网络直径	对剖宽度	对称
一维	线性阵列	N	2	$N-1$	1	
	环形	N	2	$\lfloor N/2 \rfloor$(双向)	2	✓
二维	2-D网孔	$(\sqrt{N} \times \sqrt{N})$	4	$2(\sqrt{N}-1)$	$\sqrt{N}$	
	Illiac网孔	$(\sqrt{N} \times \sqrt{N})$	4	$\sqrt{N}-1$	$2\sqrt{N}$	
	2D环绕	$(\sqrt{N} \times \sqrt{N})$	4	$2\lfloor \sqrt{N}/2 \rfloor$	$2\sqrt{N}$	✓
	二叉树	N	3	$2(\lceil \log N \rceil - 1)$	1	
	星形	N	$N-1$	2	$\lfloor N/2 \rfloor$	
三维	超立方	$N=2^n$	n	n	$N/2$	✓
	立方环	$N=k2^k$	3	$2k-1+\lfloor k/2 \rfloor$	$N/(2k)$	✓

(5) Amdahl定律[加速比定律]：问题规模一定

计算负载一定，增加处理器数来加快执行速度。

$S=\frac{W_s+W_p}{W_s+W_p/p}=\frac{f+(1-f)}{f+\frac{(1-f)}{p}}=\frac{p}{1+f(p-1)}$ 当$p\to\infty$时，$S=\frac{1}{f}$

这意味着随处理器数无限增大，并行计算所能达到的加速比上限为$\frac{1}{f}$

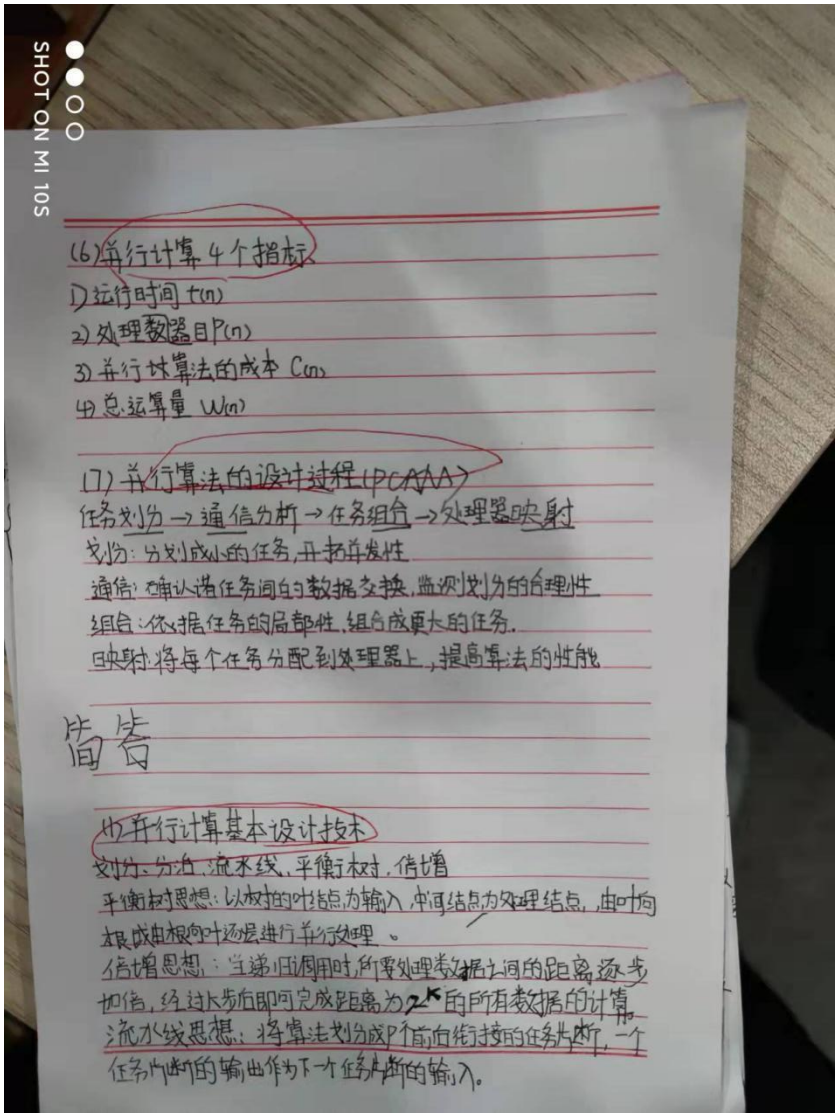
$S=\frac{W_s+W_p}{W_s+W_p/p+W_o}=\frac{p}{1+f(p-1)+\frac{W_o p}{W}}$ 当$p\to\infty$时，$S=\frac{1}{f+\frac{W_o}{W}}$

可见串行分量和并行分量容外开销越大，则加速比越小

Gustafson定律[加速比定律] 时间一定

$S=p-f(p-1)$

这意味着随处理器数目增加，加速比与处理器数成比例线性地增加。



(6)并行计算4个指标

1)运行时间 $t(n)$

2)处理器数目 $p(n)$

3)并行算法的成本 $C(n)$

4)总运算量 $W(n)$

(7)并行算法的设计过程(PCAAA)

任务划分→通信分析→任务组合→处理器映射

划分:分划成小的任务,开拓并发性

通信:确认满任务间的数据交换,监测划分的合理性

组合:依据任务的局部性,组合成更大的任务.

映射:将每个任务分配到处理器上,提高算法的性能

简答

(1)并行计算基本设计技术

划分、分治、流水线、平衡树、倍增

平衡树思想:以双节的叶结点为输入,内部结点为处理结点,由叶向根或由根向叶逐层进行并行处理。

倍增思想:当递归调用时,所要处理数据之间的距离逐步加倍,经过k步后即可完成距离为2^k的所有数据的计算。

流水线思想:将算法划分成P个前后衔接的任务片断,一个任务片断的输出作为下一个任务片断的输入。

(2) 四种常见计算模型及它们的特点

1) PRAM(并行随机存取机器)是一种抽象的计算模型

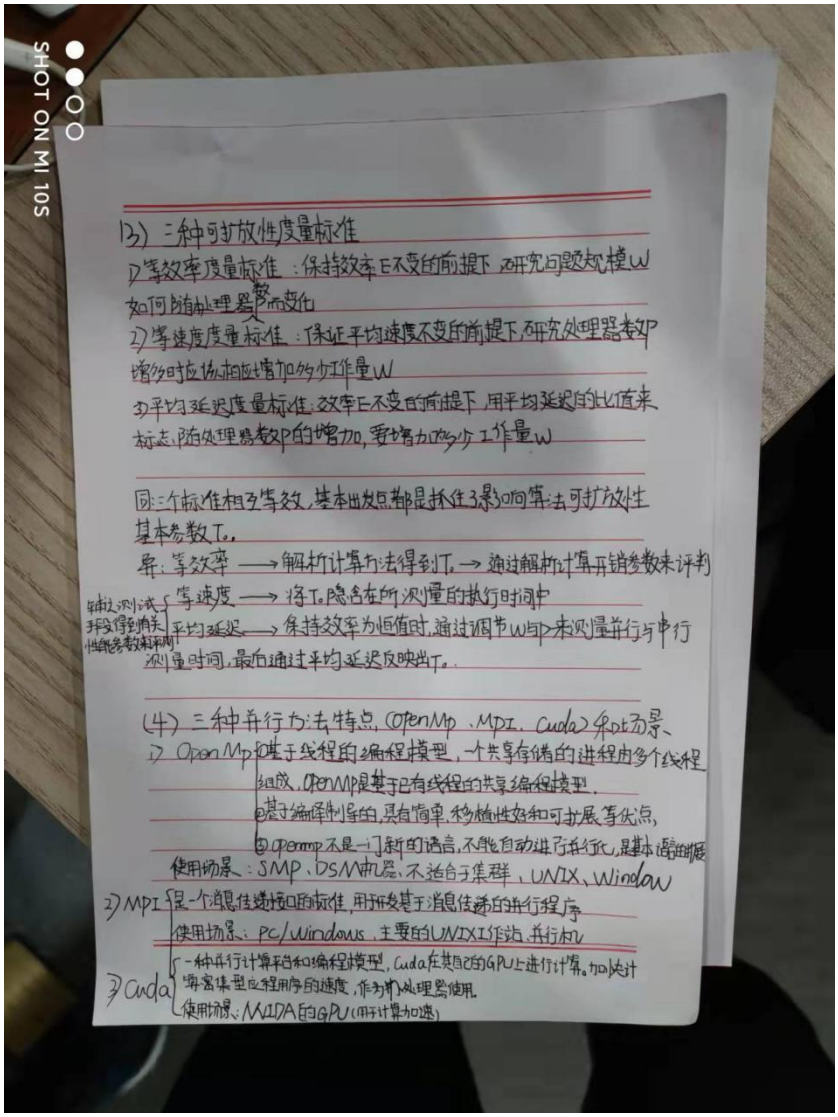
优点：①适合并行算法的表达、分析和比较，使用简单。
②易于设计算法，稍加修改便可运行在不同的并行机上。

缺点：①PRAM是一个同步模型，这意味着所有指令都按锁步的方式操作，用户感觉不到同步存在的开销。
②要求处理机器间通信无延迟，无限带宽和无开销，不现实

2) 异步PRAM模型
①每个处理器包有其局存，局部时钟和局部程序。
②处理器间通信经过共享全局存储器。
③无全局时钟。
④处理器任何时间依赖关系需要明确地在各处理器的程序中加入同步(路障)指令。
⑤一条指令可在非确定但有限的时间内完成

3) BSP模型
①处理器与选路器分开，强调了计算任务和通信任务分开。
②采用路障方式，从硬件实现的全局同步是在可控的粗粒度级。程序员无负担
③在分析BSP模型的性能时，假定局部操作可在一个时间步内完成，而在每一个超级步中，一个处理器至多发送或接收h条消息。
④为PRAM模型所设计的算法，均可用在每个BSP处理器上模拟以一些PRAM处理器的方法实现之。
⑤将计算划分为一个个超级步，有效避免了死锁。

4) logP模型
①抓住了网络与处理器之间的性能瓶颈，g反映了通信带宽，单位时间内最多有L/g个消息可能进行处理器间传送。
②处理机器间异步工作，并通过处理机间消息使通完成同步。
③消息延迟不确定，但延迟不太大。
④可预估算法实际运行时间。
⑤对编程模式有一定的反映。



(3) 三种可扩放性度量标准

1)等效率度量标准：保持效率E不变的前提下，研究问题规模W如何随处理器数p而变化

2)等速度度量标准：保证平均速度不变的前提下，研究处理器数p增多时应该相应增加多少工作量W

3)平均延迟度量标准：效率E不变的前提下，用平均延迟的比值来标志，随处理器数p的增加，要增加多少工作量W

同：三个标准相当于等效，基本出发点都是抓住影响算法可扩放性基本参数T。

异：等效率 → 解析计算方法得到T。→ 通过解析计算开销参数来评判

等速度 → 将T。隐含在所测量的执行时间中

平均延迟 → 保持效率为恒值时，通过调节W与p来测量并行与串行测量时间，最后通过平均延迟反映出T。

(4) 三种并行方法特点，(OpenMp、MPI、Cuda)分别适用场景、

1) OpenMp：基于线程的编程模型，一个共享存储的进程由多个线程组成，OpenMp是基于已有线程的共享编程模型，
基于编译制导的，具有简单、移植性好和可扩展等优点，
②openmp不是一门新的语言，不能自动进行并行化，是基本循环级
使用场景：SMP、DSM机器，不适合于集群、UNIX、Window

2) MPI：是一个消息传递接口的标准，用开发基于消息传递的并行程序
使用场景：PC/Windows，主要的UNIX工作站、并行机

3) Cuda：一种并行计算平台和编程模型，Cuda在其自己的GPU上进行计算。加快计算密集型应用程序的速度，作为协处理器使用
使用场景：NVIDA的GPU(用于计算加速)

并行计算：

~~① PVP：并行向量处理机~~

1. 大型并行机系统：
   - 单指令多数据流 (SIMD)
   - 并行向量处理机 (PVP)
   - 对称多处理机 (SMP)
   - 大规模并行处理机 (MPP)
   - 工作站机群 (COW)
   - 分布共享存储 (DSM)

   均为多指令多数据流

2. 并行计算存储访问类型：
   - 均匀存储访问 (UMA)
   - 非均匀存储访问 (NUMA)
   - 全高速缓存存储访问 (COMA)
   - 高速缓存一致性非均匀存储访问 (CC-NUMA)
   - 非远程存储访问 (NORMA)

3. 并行网络拓扑结构
   (网络直径，对剖宽度)

一维：线性阵列：N个结点用N-1条链路连接，直径 N-1，对剖宽度 1
(二近邻连接)

有环：单环：内度 2，直径 N-1

双环：内度 2，直径 N/2

二维：二维网孔

Illiac网孔

2-D环绕

直径 对剖宽度

2(N-1) N

√N - 1

2√N

		内节点度	直径	对剖宽度
一维	线性	2	$N-1$	1
	单向环	2	$N-1$	2
	双向环	2	$N/2$	2
二维	2-D网孔	4	$2(\sqrt{n}-1)$	$\sqrt{n}$
	Illiac网孔	4	$\sqrt{n}-1$	$2\sqrt{n}$
	2-D环绕	4	$2\lfloor\sqrt{n}/2\rfloor$	$2\sqrt{n}$.
树	二叉树	3.	$2(\lceil\log n\rceil-1)$	1
超立方	n立方 一个顶点有n个连接	n	n	n/2

4. 加速比定理，Amdahl，

p：并行中处理器数.
w：问题规模.
w_s：串行分量（不可并行化）$w_s=w_1$
w_p：并行分量
$w=w_s+w_p$
f：串行分量比例：$f=\frac{w_s}{w}$
$T_s=T_1$：串行执行时间.
T_p：并行执行时间.
S：加速比.
E：效率.

(4.1) Amdahl：问题规模固定，环固定.

$$S=\frac{w_s+w_p}{w_s+\frac{w_p}{p}}=\frac{1}{f+(1-f)/p}=\frac{p}{pf+1-f}=\frac{p}{(p-1)f+1}$$

当$p\to\infty$时，$S=\frac{1}{f}$

结论：
随着处理器数目的无限增大，并行系统所能达到的加速比上限为$\frac{1}{f}$.

$$S=\frac{w_s+w_p}{w_s+\frac{w_p}{p}}=\frac{f+(1-f)}{pf+(1-f)p}=\frac{p}{(p-1)f+1}$$

$p\to\infty$. $S\to\frac{1}{f}$

(4.2) Gustafson定律：时间固定，规模不固定。　(4.3) Sun和Ni定律

设 $W_{串} = W_s + W_p$.

为保持时间不变，则 $W_{并} = W_s + p \cdot W_p$

则加速比 $S' = \dfrac{W_s + p \cdot W_p}{W_s + W_p} = f + p(1-f) = p - f(p-1)$

结论：随着处理器数目的增加，加速几乎与处理器数成比例的线性增加，串行比例 f 不再是瓶颈。

5. 并行算法关心的几个指标：
   ① CPU和存储器的某些基本性能指标。
   ② 通信开销。
   ③ 机器的成本，价格与性能价格比。

6. 并行算法设计过程：
   划分、通信、组合、映射。（简称PCAM）
   划分 → 任务划分　通信 → 通信分析　组合 → 任务组合　映射 → 处理器映射。

7. 并行计算模型 and 特点。（PRST）
   ① PRAM：并行随机存取机器
   ② 异步PRAM模型：
   ③ BSP模型
   ④ LogP模型

8. 可扩放性评测标准

可扩放性：计算机系统性能随处理器数个而按比例提高的能力.

可扩放性三个标准：等效率度量标准、等速度度量标准、平均延迟度量标准.

9. 并行算法设计技术.

划分设计技术、分治设计技术、平衡树设计技术、倍增设计技术.
流水线设计技术.

均匀划分. 方根划分.
对数划分. 功能划分.

10. 习题 4.6. 4.7. 4.11

5.2
brent证明.

11. PSRS. 排序结果(每步). P191.

5.2 证明: $\frac{1}{2}n^2-3n=\Theta(n^2)$

设 $C_1=\frac{1}{4}$

$\frac{1}{2}n_0^2-3n_0-\frac{1}{4}n_0^2=0$

$n_0=0$ 或 12.

当 $n\geq n_0=12$ 时.

$f(n)-C_1g(n)\geq 0$.

取 $C_2=10$ 时, 当 $n\geq n_0$ 时.

$10n^2-\frac{1}{2}n^2+3n^2\geq 0$.

即 $f(n)-C_2g(n)\leq 0$.

∴ 当 $n_0=12$, $C_1=\frac{1}{4}$, $C_2=10$ 时,

满足 $n\geq n_0$, $C_1g(n)\leq f(n)\leq C_2g(n)$

即 $\frac{1}{2}n^2-3n=\Theta(n^2)$

程序：

1. ~~MPI 6个基本函数~~

① 启动：int MPI_Init(int *argc, char ***argv)

② ~~结束~~：int MPI_Finalize(void)

③ ~~获取进程编号~~：int MPI_Comm_rank(MPI_Comm comm, int *rank)

④ ~~获取进程数~~：int MPI_Comm_size(MPI_Comm comm, int *size)

⑤ ~~消息发送~~：int MPI_Send(void *buf, int count, MPI_Datatype ~~datatype~~ datatype, int dest, int tag, MPI_Comm comm)

⑥ ~~消息接收~~：

2. MPI：并行.

```
MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Send(&senddata, 1, MPI_INT, 1, tag, MPI_COMM_WORLD);
MPI_Recv(&recvdata, 1, MPI_INT, 0, tag, MPI_COMM_WORLD, &status);
MPI_Finalize();
```

3. ~~OpenMP~~：

OpenMP：计算π.

```
#include <omp.h>
static long num_steps=100000;
#define NUM_THREADS 2
void main()
{  int i;
   double x,pi,sum=0;
   double step=1.0/(double)num_steps;
   omp_set_num_threads(NUM_THREADS);
   #pragma omp parallel for reduction(+:sum) private(i,x)
   for(i=0;i<num_steps;i++)
   {  x=(i+0.5)*step;  sum=sum+4.0/(1.0+x*x); }
   pi=sum*step;
}
```

3. CUDA.

```
__global__ void add(int *a, int *b, int *c)
{   *c = *a + *b; }.

int main(void)
{   int a=1, b=2, c;
    int *d_a, *d_b, *d_c;

    cudaMalloc((void **)&d_a, sizeof(int));
    cudaMalloc((void **)&d_b, sizeof(int));
    &d_c, ~~~~~~~~~~

    cudaMemcpy(d_a, &a, sizeof(int), cudaMemcpyHostToDevice);
    (d_b, &b, ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~);

    add<<<1,1>>>(d_a, d_b, d_c);

    cudaMemcpy(&c, d_c, sizeof(int), cudaMemcpyDeviceToHost);

    cudaFree(d_a);
    cudaFree(d_b);
    ~~~~(d_c);

    return 0;
}.
```