EREW如何在O(logp)内模拟CRCW：

分发树：

一个超步：读一个过来，再写到SM中，因此可以以指数阶增长

当读多个不同数据时，要将处理器按照读取数据构造数据对，并统计读同一个数据的个数，构造不同的的分发树，构成分发森林

如何将求最大值算法改造成并行成本最优？

使用的处理器数量为p/logp，最底层的处理器（即第一次）每个处理器对logp个数使用串行算法球最大值，时间为O(logp)，之后向上传递是用平衡树的方法，因此平衡树的时间消耗为O(log(p/logp))可以化简为O(logp)；故第一阶段和第二阶段加在一起时间消耗仍为O(logp)，因此并行成本为O(p)等于串行算法的时间消耗，因此为并行成本最优（此处使用了级联的想法，即将最底层的串行块和上层的并行块进行连接来完成计算）

谈谈你所知道的并行计算与云计算的区别?

并行计算强调性能，大规模

云计算强调服务。

并行计算是一台计算机，配备有多处理机，多处理机之间进行合同协作计算，最终结果由一台计算机处理。云计算指计算机通过网络发送计算命令给服务器，让服务器执行计算任务并将结果返还给发送命令的计算机。并行计算是由单个用户完成的，云计算是没有用户参与，交给网络另一端的服务器完成的。云计算由并行计算发展而来，是并行计算的商业实现。

并行程序的描述应如何？与串行程序有什么不同?

循环描述并行化，任务分配分配，分配方式

信息同步，通讯

再论O(1)时间并行求解n个元素最大值算法，修改算法能否省去二维B数组? M数组呢？

不可省去二维B数组，但M数组可以。因为M数组可以用B数组的对角元代替。

CMP、SMP和Cluster在处理器、操作系统、并行编程上有什么不同? 其他还有什么更多的区别?

见第一章PPT 2-右上

如何设计一个2×2×2个处理器的度数均衡互连网络(节点度数均衡是指度数至多差1)，用最少的连边数获得度数小于等于3的互连网络并使网络直径最小?

XYZ三个轴各减去一条边，对剖宽度变为3

试给出环上收集(all-to-one)的CT选路算法，并画出8个节点环上的选路步骤示意图(收集到节点0)，以及推导环上的通信时间 。

将课本P58页上的图上的所有箭头反方向，并逆序标注顺序即可。因为发送包长的等比数列仅仅只是反过来而已，所以时间是相同的

序列x1, x2, … , xn前缀和串行算法的直接并行化可实现吗？又如何应用策略2和3进行并行化？

策略2：每个结果si单独计算，根据定义出发，第i个结果用i个x，相当于n个并行的求和，时间为O(logn)，使用的总处理器数量为O(n2)；模型为PRAM-CREW

策略3：将前缀和转化为线性方程组，并使用并行化的线性方程组求解算法进行求解

也可以是用平衡树的方式进行计算，详见书P199算法7.9.

倍增技术也可以做前缀和计算：从距离为1，2，4逐渐倍增（2的幂）

· Activity 11 
I. (Homework 2 一 Conte) 在 PRAM-CREW 模 型 上 ， 用 n 个 处 理 器 在 
0 （ 1) 时 间 内 求 出 数 组 A n 0 。 ， 1 。 ， 1 } ， 最 先 为 1 值 的 下 标 。 写 
出 并 行 伪 代 码 。 
乙 A 是 一 个 大 小 为 n 的 布 尔 数 组 ， 欲 求 出 最 小 的 下 标 i 且 4 [ 刂 为 真 ， 试 
设 计 一 个 常 数 时 间 的 PRAM 、 -CRCW 并 行 算 法 。 如 果 使 用 PRA 从 
CREW 模 型 ， 运 行 时 间 如 何 ？ 
Hint; 《 c 叩 丫 A[1enJ to B[1enJ 
乙 fo ： 1 to n 四 0d0 
if BC 刂 :true then 
fo 闫 ： i + 1 to n 四 0d0 
B[JJ=false 
endtor 
endif 
endfor 
for i:l to n 四 0d0 
if BC 刂 :true then 
return i 
endif 
endfor 

2：处理器要O(n2)，也可以类比求最大值；如果时间为O(logn)，则处理器只用O(n)，用平衡树，或者前缀和都可以完成计算

在超立方结构上，为什么Simple alg.比Cannon alg.要快？而Cannon alg.要比Fox alg.快？

Fox出发点，Cannon算法分了校准步和迭代步，实现会更麻烦；想要实现校准步与迭代步一致的办法；

简单分块中，仅做了两次量很大的通讯；而且存储容量很大

Cannon算法中，则做了数量更多的量较小的通讯

Cannon算法中在第一步做了全域通讯，但在迭代步中，仅做了局域性较小的局域通讯

Fox算法则一直在做局域性较大的通讯（行级别的通讯）。

简单并行分块乘法是使用多到多播送传输数据的。经过次播送，数据可以传输到所有处理器。而Cannon需要进行旋转，旋转次数共次，所需次数多，所以慢。

Fox是先旋转后多到多播送，且每一次旋转都要多到多播送一次，而Cannon不需要多到多播送，所以Fox慢。

如何将分布式Cannon alg.伪代码改造为共享存储式算法？并与Simple alg.比较不同之处。

本质，PPT中给的的分布式在旋转时，要一个个的传递分块矩阵，因此在共享存储模式下，要实现一步到位

**Cannon Alg.**

for all Pij par-do

{

Cij = 0

for k = 0 to sqrt(p) - 1 do

Cij += Ai, (i + j + k) mod sqrt(p) \* B(i + j + k) mod sqrt(p), j

}

**Simple Alg. //PRAM-CREW**

for all Pij par-do

{

Cij = 0

for k = 0 to sqrt(p) - 1 do

Cij += Ai, k \* Bk, j

}

对于上三角方程组求解问题，如果使用块行带状划分，其计算效果与前面的循环行带状相比如何?

在上三角方程组求解问题中，越靠下的地方运算负载越小（见第三层循环），越靠上的地方运算负载越大。块行带状划分会导致负载出现不均衡，最上面的一块运行得最慢，成为瓶颈。

证明串行FFT分治递归算法的正确性。

写出串行FFT蝶式递归算法的伪代码。

Procedure BF\_RFFT(A, B)

{

if n == 1 then:

b0 = a0

else

{

(1)

A[0] = (a0[0], … , an/2 - 1[0]), B[0] = (b0[0], … , bn/2 - 1[0])

A[1] = (a0[1], … , an/2 - 1[1]), B[1] = (b0[1], … , bn/2 - 1[1])

(2)

z = 1;

for j = 0 to n/2 - 1 do

{

aj[0] = aj + an/2 + j

aj[1] = (aj - an/2 + j) \* z

z = z \* ω

}

(3)

BF\_RFFT(A[0], B[0]);

BF\_RFFT(A[1], B[1]);

(4)

for j = 0 to n/2 - 1 do

b2j = bj[0]

b2j + 1 = bj[1]

}

}

SIMD-BF上的FFT算法在蝶形互连的分布式结构上ω权因子计算，除了各个节点独立计算方法之外，还有什么计算方法？该方法与前者的区别在哪里？

初始时: Pk,i读入ωexpr(k,i)，k=logn

若Pr+1,i已有ωexpr(r+1,i), 则Pr,i中的ωexpr(r,i)=ω2expr(r+1,i)

所以, 经过logn步就可以计算出每个ωexpr(r,i)

SIMD-BF上的FFT算法除了适应于蝶形互连的分布式结构，还有什么样的结构适合？

超立方体，可以嵌入到超立方体上。

LA：一维线性连接 Linear Array(Connected)

MC：二维网孔链接 Mesh Connected

TC：树形连接Tree Connected

HC：超立方连接Hypercube Connected

CCC：超立方环连接Cube Connected-Cycles

SE：洗牌交换连接Shuffle Exchange