并行计算
Parallel Computing

主讲人　　孙广中

Spring, 2018

第二篇 并行算法的设计

第五章 并行算法与并行计算模型

第六章 并行算法基本设计策略

第七章 并行算法常用设计技术

第八章 并行算法一般设计过程

第五章 并行算法与并行计算模型

5.1 并行算法的基础知识

5.1.1 并行算法的定义和分类

5.1.2 并行算法的表达

5.1.3 并行算法的复杂性度量

5.1.4 并行算法中的同步和通讯

5.2 并行计算模型

并行算法的定义和分类

- 并行算法的定义
  - 算法：略
  - 并行算法：一些可同时执行的诸进程的集合，这些进程互相作用和协调动作从而达到给定问题的求解。
- 并行算法的分类
  - 数值计算和非数值计算
  - 同步算法和异步算法
  - 分布算法
  - 确定算法和随机算法

第五章 并行算法与并行计算模型

5.1 并行算法的基础知识

5.1.1 并行算法的定义和分类

5.1.2 并行算法的表达

5.1.3 并行算法的复杂性度量

5.1.4 并行算法中的同步和通讯

5.2 并行计算模型

并行算法的表达

- 描述语言
  - 可以使用类Algol、类Pascal等；
  - 在描述语言中引入并行语句。
- 并行语句示例
  - Par-do语句

```
for i=1 to n par-do
    ……
end for
```

  - for all语句

```
for all Pi, where 0≤i≤k  do
  ……
end for
```

第五章 并行算法与并行计算模型

5.1 并行算法的基础知识

5.1.1 并行算法的定义和分类

5.1.2 并行算法的表达

5.1.3 并行算法的复杂性度量

5.1.4 并行算法中的同步和通讯

5.2 并行计算模型

并行算法的复杂性度量

- 串行算法的复杂性度量
  - 最坏情况下的复杂度(Worst-Case Complexity)
  - 期望复杂度(Expected Complexity)
- 并行算法的几个复杂性度量指标
  - 运行时间t(n): 包含计算时间和通讯时间，分别用计算时间步和选路时间步作单位。n为问题实例的输入规模。
  - 处理器数p(n)
  - 并行算法成本c(n): c(n)=t(n)p(n)
  - 成本最优性：若c(n)等于在最坏情形下串行算法所需要的时间，则并行算法是成本最优的。
  - 总运算量W(n): 并行算法求解问题时所完成的总的操作步数。

并行算法的复杂性度量

- Brent定理

  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t(n)=O(W(n)/p+T(n))时间内执行完毕。

  注：

  (1)揭示了并行算法工作量和运行时间的关系；

  (2)提供了并行算法的WT(Work-Time)表示方法；

  (3)告诉我们：设计并行算法时应尽可能将每个时间步的工作量均匀地分摊给p台处理器，使各处理器处于活跃状态。

第五章 并行算法与并行计算模型

5.1 并行算法的基础知识

5.1.1 并行算法的定义和分类

5.1.2 并行算法的表达

5.1.3 并行算法的复杂性度量

5.1.4 并行算法中的同步和通讯

5.2 并行计算模型

并行算法的同步

- 同步概念
  - 同步是在时间上强使各执行进程在某一点必须互相等待；
  - 可用软件、硬件和固件的办法来实现。
- 同步语句示例
  - 共享存储多处理器上求和算法

输入：A=(a_0,…,a_{n-1}),处理器数p

输出：S=Σa_i

```
Begin
    (1)S=0                                   (2.3) lock(S)
    (2)for all Pi where 0≤i≤p-1 do                 S=S+L
      (2.1) L=0                               (2.4) unlock(S)
      (2.2) for j=i to n step p do           end for
              L=L+aj                    End
          end for
      end for
```

Example: Critical Section

```
#define NUMTHREADS 4
CRITICAL_SECTION g_cs; // why does this have to be global?
int g_sum = 0;

DWORD WINAPI threadFunc(LPVOID arg )
{
   int mySum = bigComputation();
   EnterCriticalSection(&g_cs);
    g_sum += mySum;                  // threads access one at a time
   LeaveCriticalSection(&g_cs);
   return 0;
}
main() {
   HANDLE hThread[NUMTHREADS];
   InitializeCriticalSection(&g_cs);
   for (int i = 0; i < NUMTHREADS; i++)
     hThread[i] =
          CreateThread(NULL,0,threadFunc,NULL,0,NULL);
   WaitForMultipleObjects(NUMTHREADS, hThread, TRUE, INFINITE);
   DeleteCriticalSection(&g_cs);
}
```

并行算法的通讯（1）

- 通讯
  - 共享存储多处理器使用：global read(X,Y)和global write(X,Y)
  - 分布存储多计算机使用：send(X,i)和receive(Y,j)
- 通讯语句示例
  - 算法5.2 分布存储多计算机上矩阵向量乘算法

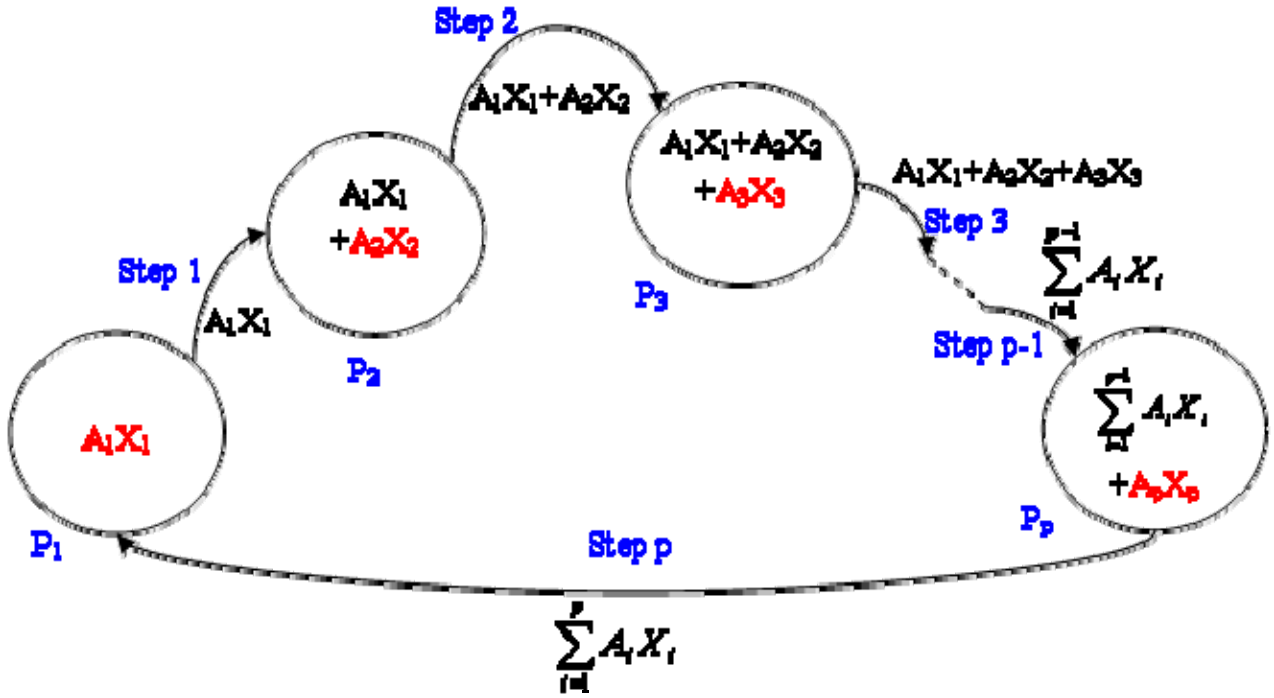
输入：处理器数p, A划分为B=A[1..n,(i-1)r+1..ir],
x划分为w=w[(i-1)r+1..ir] r=n/p, i=1~p

输出：P_1保存乘积AX

```
Begin
    (1) Compute z=Bw
    (2) if i=1 then y=0 else receive(y,left) endif
    (3) y=y+z
    (4) send(y,right)
    (5) if i=1 then receive(y,left)
End
```

并行算法的通讯（2）

- 计算过程图示



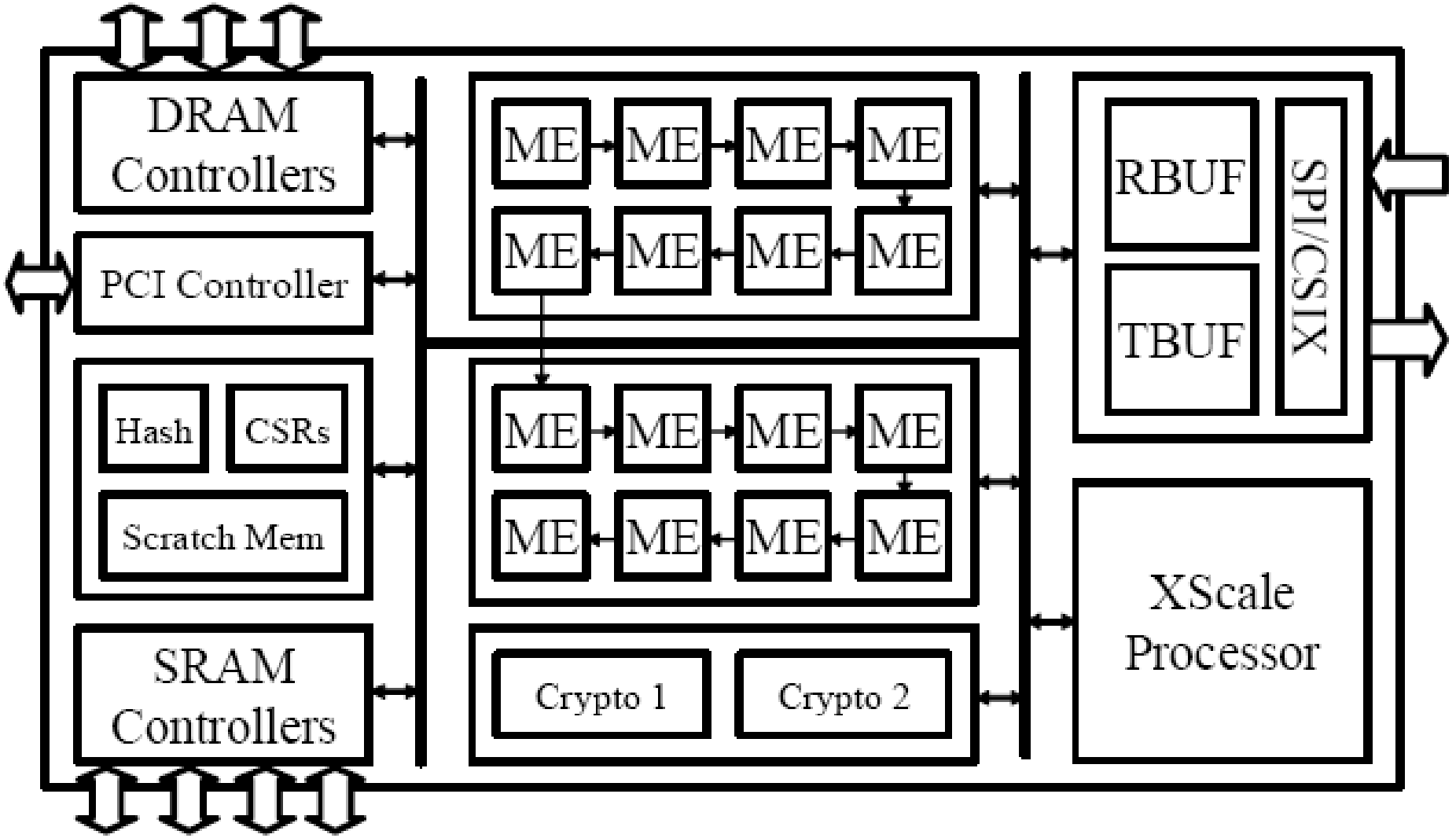
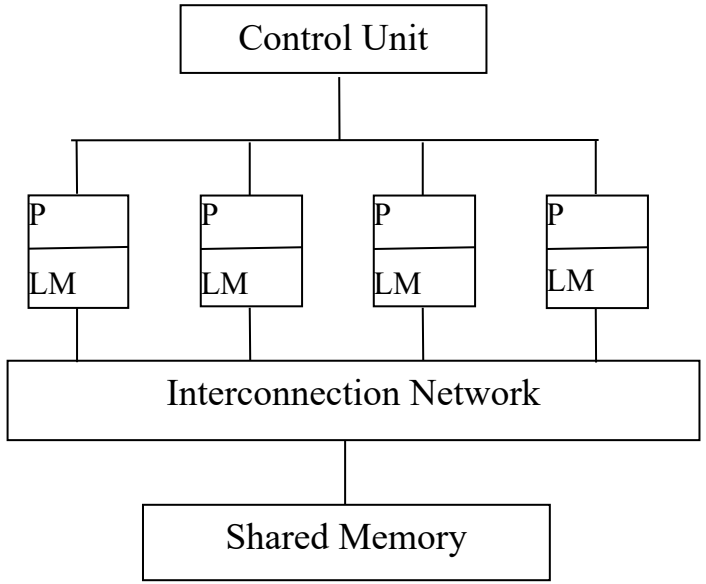


Figure 2: Organization of the IXP 2850 NP.

PRAM模型

- 基本概念
  - 由Fortune和Wyllie1978年提出，又称SIMD-SM模型。有一个集中的共享存储器和一个指令控制器，通过SM的R/W交换数据，隐式同步计算。
- 结构图



PRAM模型

- 分类
  - PRAM-CRCW并发读并发写
    - CPRAM-CRCW(Common PRAM-CRCW)：仅允许写入相同数据
    - PPRAM-CRCW(Priority PRAM-CRCW)：仅允许优先级最高的处理器写入
    - APRAM-CRCW(Arbitrary PRAM-CRCW)：允许任意处理器自由写入
  - PRAM-CREW并发读互斥写
  - PRAM-EREW互斥读互斥写
- 计算能力比较
  - PRAM-CRCW是最强的计算模型，PRAM-EREW可logp倍模拟PRAM-CREW和PRAM-CRCW

$$T_{EREW} \geq T_{CREW} \geq T_{CRCW}$$

$$T_{EREW} = O(T_{CREW} \cdot \log p) = O(T_{CRCW} \cdot \log p)$$

PRAM模型

- 优点
  - 适合并行算法表示和复杂性分析，易于使用，隐藏了并行机的通讯、同步等细节。
- 缺点
  - 不适合MIMD并行机，忽略了SM的竞争、通讯延迟等因素

5.1 并行算法的基础知识
5.2 并行计算模型
- 5.2.1 PRAM模型
- 5.2.2 异步PRAM模型
- 5.2.3 BSP模型
- 5.2.4 logP模型

APRAM模型

- 基本概念
  - 又称分相（Phase）PRAM或MIMD-SM。每个处理器有其局部存储器、局部时钟、局部程序；无全局时钟，各处理器异步执行；处理器通过SM进行通讯；处理器间依赖关系，需在并行程序中显式地加入同步路障。
- 指令类型

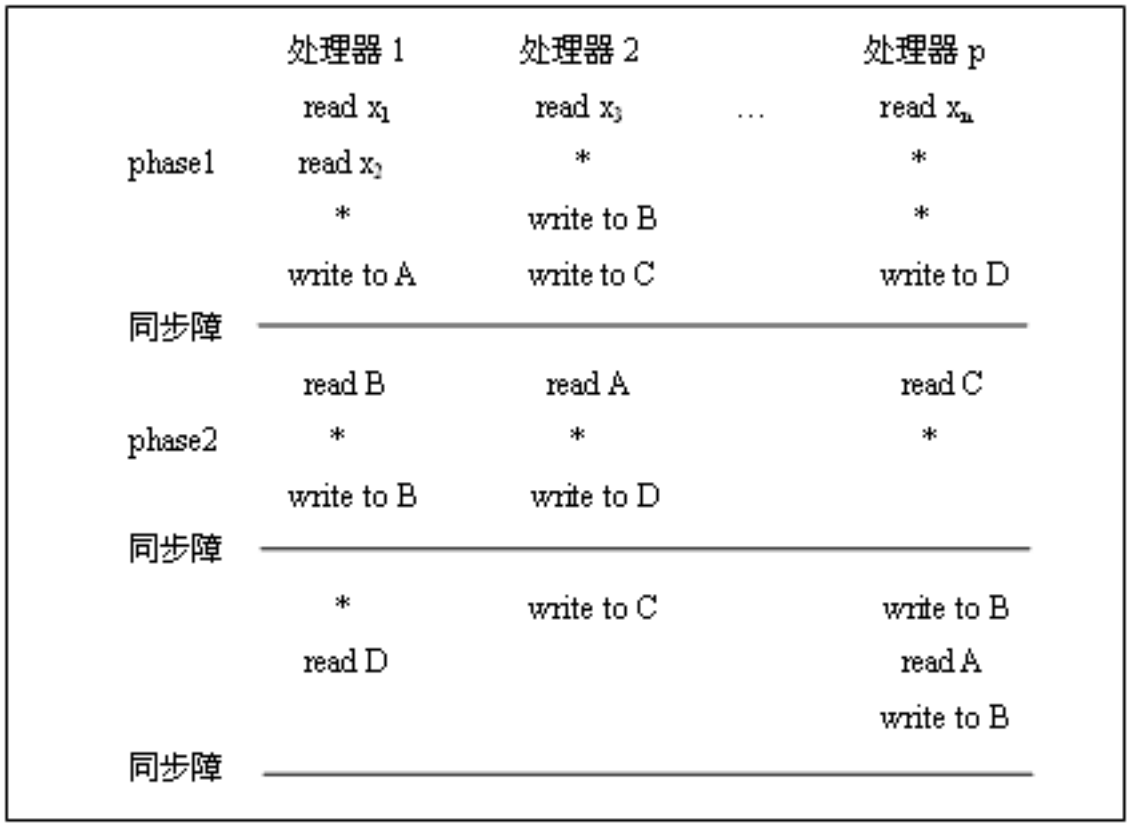
(1)全局读　　(2)全局写

(3)局部操作　　(4)同步

APRAM模型

- 计算过程

  由同步障分开的全局相组成



APRAM模型

- 计算时间

  设局部操作为单位时间；全局读/写平均时间为d，d随着处理器数目的增加而增加；同步路障时间为B=B(p)非降函数。

  满足关系 $2 \le d \le B \le p$ ； $B = B(p) = O(d \log p)$ 或 $O(d \log p / \log d)$

  令 t_{ph} 为全局相内各处理器执行时间最长者，则APRAM上的计算时间为

$$T = \sum t_{ph} + B \times \text{同步障次数}$$

- 优缺点

  易编程和分析算法的复杂度，但与现实相差较远，其上并行算法非常有限，也不适合MIMD-DM模型。

5.1 并行算法的基础知识
5.2 并行计算模型
- 5.2.1 PRAM模型
- 5.2.2 异步PRAM模型
- 5.2.3 BSP模型
- 5.2.4 logP模型

BSP模型

- 基本概念
  - 由Valiant(1990)提出的，“块”同步模型，是一种异步MIMD-DM模型，支持消息传递系统，块内异步并行，块间显式同步。
- 模型参数
  - *p*：处理器数(带有存储器)
  - *l*：同步障时间(Barrier synchronization time)
  - *g*：带宽因子(time steps/packet)=1/bandwidth

BSP模型

- 计算过程

  由若干超级步组成，

  每个超级步计算模式为左图

- 优缺点

  强调了计算和通讯的分离，提供了一个编程环境，易于程序复杂性分析。但需要显式同步机制，限制至多h条消息的传递等。

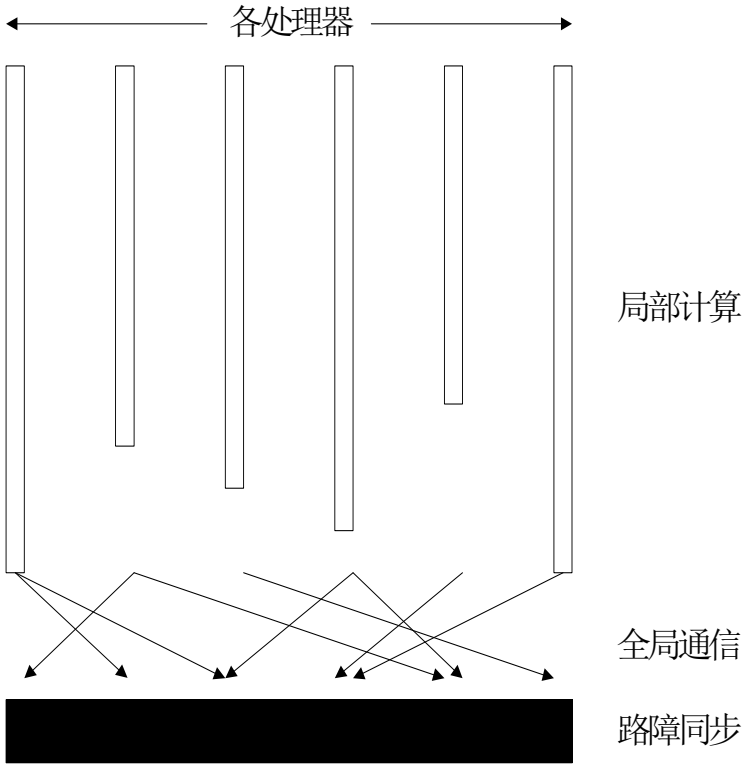


图4.3

5.1 并行算法的基础知识
5.2 并行计算模型
- 5.2.1 PRAM模型
- 5.2.2 异步PRAM模型
- 5.2.3 BSP模型
- 5.2.4 logP模型

logP模型

- 基本概念
  - 由Culler(1993)年提出的，是一种分布存储的、点到点通讯的多处理机模型，其中通讯由一组参数描述，实行隐式同步。
- 模型参数
  - *L* ：network latency
  - *o* ：communication overhead
  - *g* ：gap=1/bandwidth
  - *P* ：#processors

  注：*L*和*g*反映了通讯网络的容量

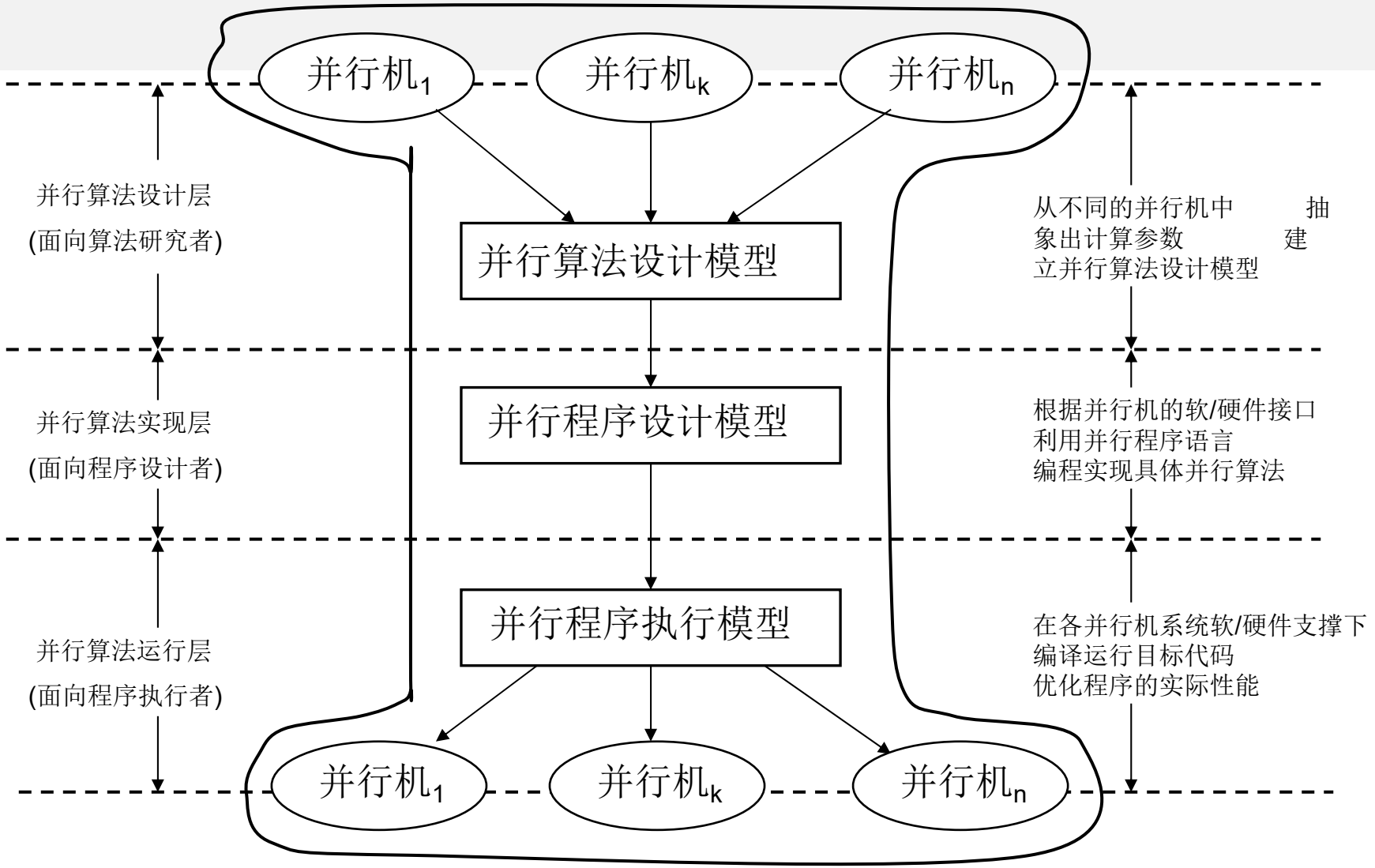
logP模型

- 优缺点

  捕捉了并行机的通讯瓶颈，隐藏了并行机的网络拓扑、路由、协议，可以应用到共享存储、消息传递、数据并行的编程模型中；但难以进行算法描述、设计和分析。

- **BSP vs. LogP**
  - BSP→LogP：BSP块同步→BSP子集同步→BSP进程对同步 = LogP
  - BSP可以常数因子模拟LogP，LogP可以对数因子模拟BSP
  - BSP = LogP+Barriers – Overhead
  - BSP提供了更方便的程设环境，LogP更好地利用了机器资源
  - BSP似乎更简单、方便和符合结构化编程

LogP上的多播



- 三层并行计算模型从几何形状上看，呈现哑铃形状：从不同的并行计算机来(抽象计算参数建立模型)，经过不同的加工后，又回到不同的并行计算机中去(运行代码，求解问题)。

分层模型对照表

名称	并行算法设计模型	并行程序设计模型	并行程序执行模型
面向对象	算法设计者	编程者	程序运行者
作用	算法设计者和机器结构设计者之间桥梁	程序设计者与计算机软/硬之间接口	编译设计者与系统实现者之间接口
关注点	•算法正确性 •低时、空开销	•确保算法正确语义 •正确编程实现	优化程序执行性能
要素	•机器计算参数 •计算行为 •计算复杂度函数	•程序结构(编程模式) •数据结构(共享/分布) •可扩展，通用泛化	•机器性能参数 •运行时系统行为 •性能指标

分层模型对照表（续）

名称	并行算法设计模型	并行程序设计模型	并行程序执行模型
方法学	设计方法（划分，分治，流水线，平衡树，…）	编程风范（SPMD,循环并行,主从法MPMD,Fork/Join, 放牧法,流水线法,…）	执行模式（线程/进程产生,管理与撤消;同步;通信)
复杂度	算法步数	高级语言条数	机器指令条数
支撑条件	•硬件平台 •软件支撑 •算法理论	•并行语言 •工具环境 •应用编程接口API	•编译器 •OS •运行时系统 •硬件结构(CPU,Memory,I/O)
现有模型	PRAM, APRAM, BSP, logP, NHBL, UMH,DRAM(h)	OpenMP, MPI, HPF	?

并行计算模型

- A view from Berkery
  - 计算数学、计算物理、计算机软件、计算机硬件等学科20余名教授，近半年时间研讨
  - http://view.eecs.berkeley.edu/wiki/Main_Page

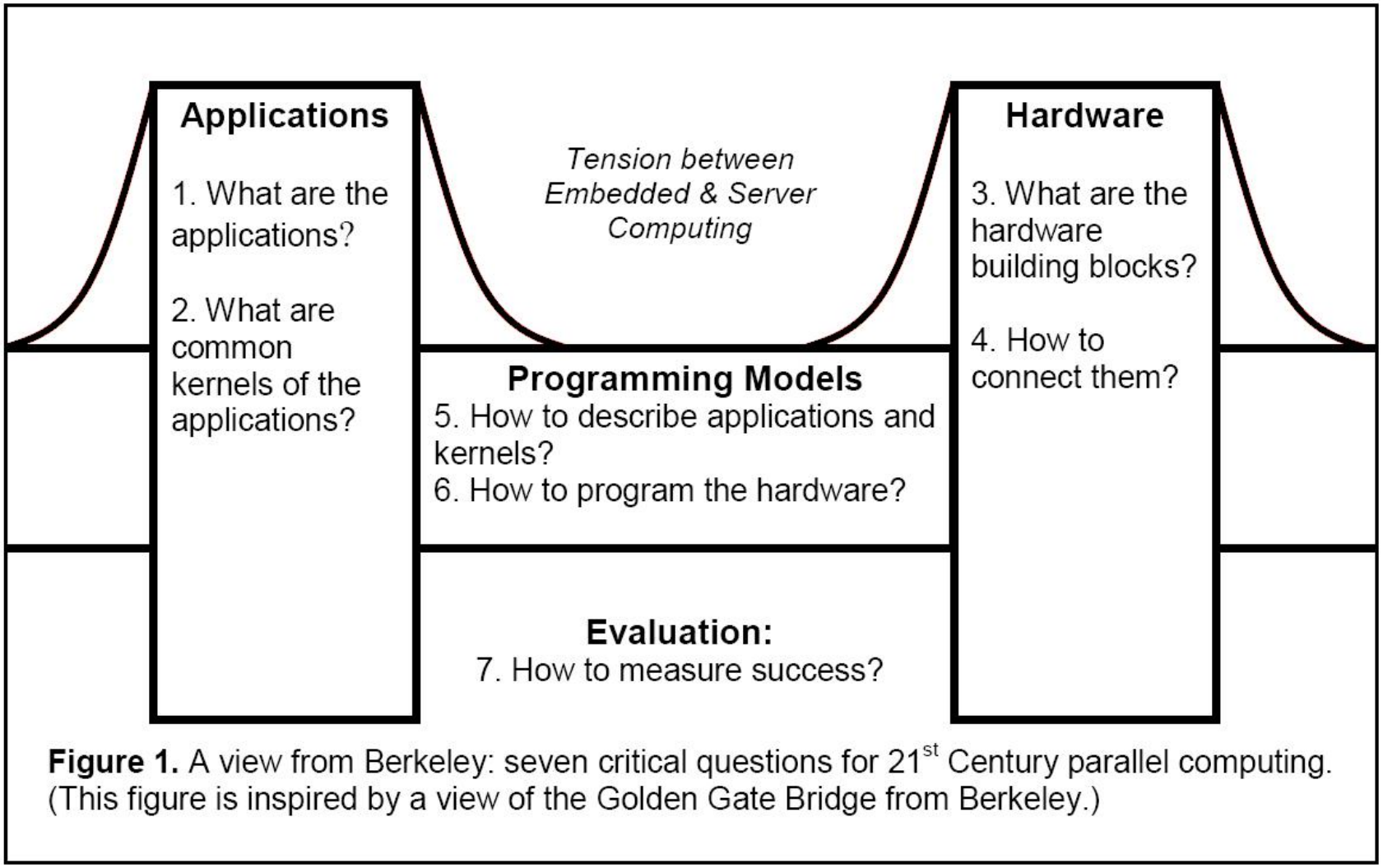


Figure 1. A view from Berkeley: seven critical questions for 21st Century parallel computing. (This figure is inspired by a view of the Golden Gate Bridge from Berkeley.)

参考文献

1. David Culler, Richard Karp, David Patterson, Abhijit Sahay, Klaus Erik Schauser, Eunice Santos, Ramesh Subramonian, and Thorsten von Eicken. **LogP: towards a realistic model of parallel computation**. In Proceedings of the fourth ACM SIGPLAN symposium on Principles and practice of parallel programming (PPOPP '93). ACM, New York, NY, USA, 1-12
2. 计永昶 , 卜添, 并行播送和求和算法在几种实际计算模型上的设计和分析, 中国科学技术大学学报, 1996 (2) :195-203
3. 顾乃杰, 李伟, 刘婧， 基于斐波那契序列的多点播送算法，计算机学报， 2002， 25(4):365—372.
4. 陈国良,苗乾坤,孙广中等， 分层并行计算模型， 中国科学技术大学学报 , 2008 , 38 (7) :841-847.

作业

- 之前作业解释

- 第五章作业