并行程序设计方法学

于策

Outline

- 并行算法设计
- PCAM方法学
- 并行程序设计模式
- 并行计算与软件工程

Outline

- 并行算法设计
- PCAM方法学
- 并行程序设计模式
- 并行计算与软件工程

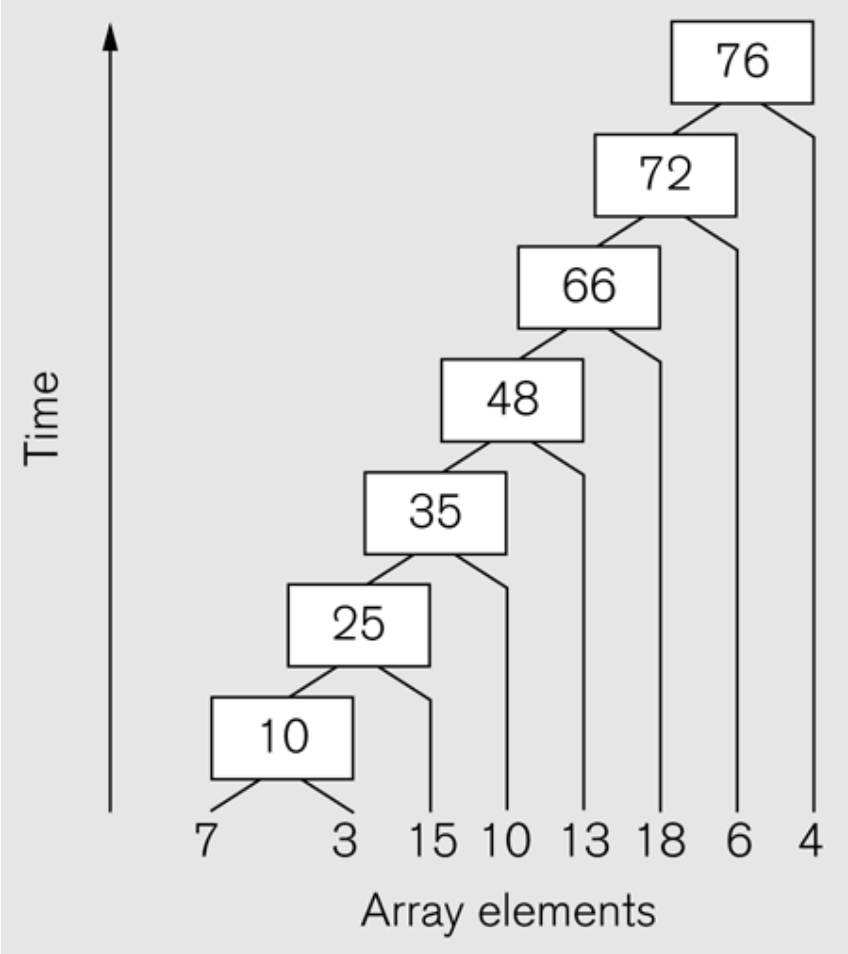
并行算法的一般设计方法

- 串行算法的直接并行化
- 从问题描述开始设计并行算法
- 借用已有算法求解新问题

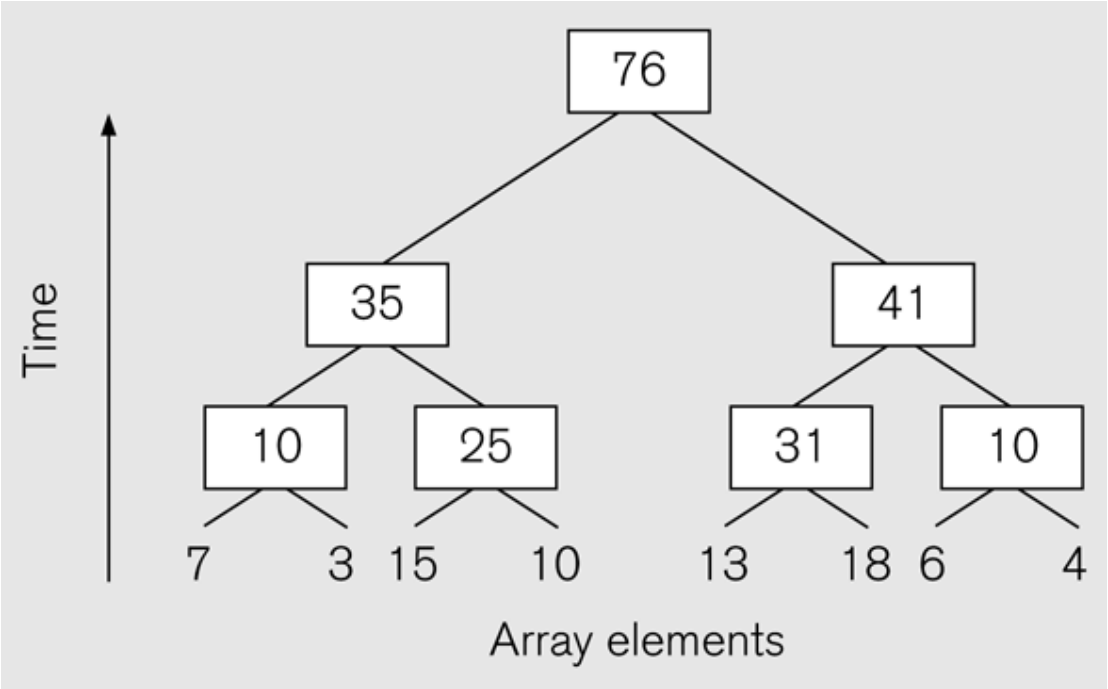
串行算法的直接并行化

- 方法描述
  - 发掘和利用现有串行算法中的并行性，直接将串行算法改造为并行算法。
- 评注
  - 由串行算法直接并行化的方法是并行算法设计的最常用方法之一；
  - 不是所有的串行算法都可以直接并行化；
  - 一个好的串行算法未必能并行化为一个好的并行算法；
  - 许多数值串行算法可以并行化为有效的数值并行算法。

数组求和（串行）



数组求和（并行）



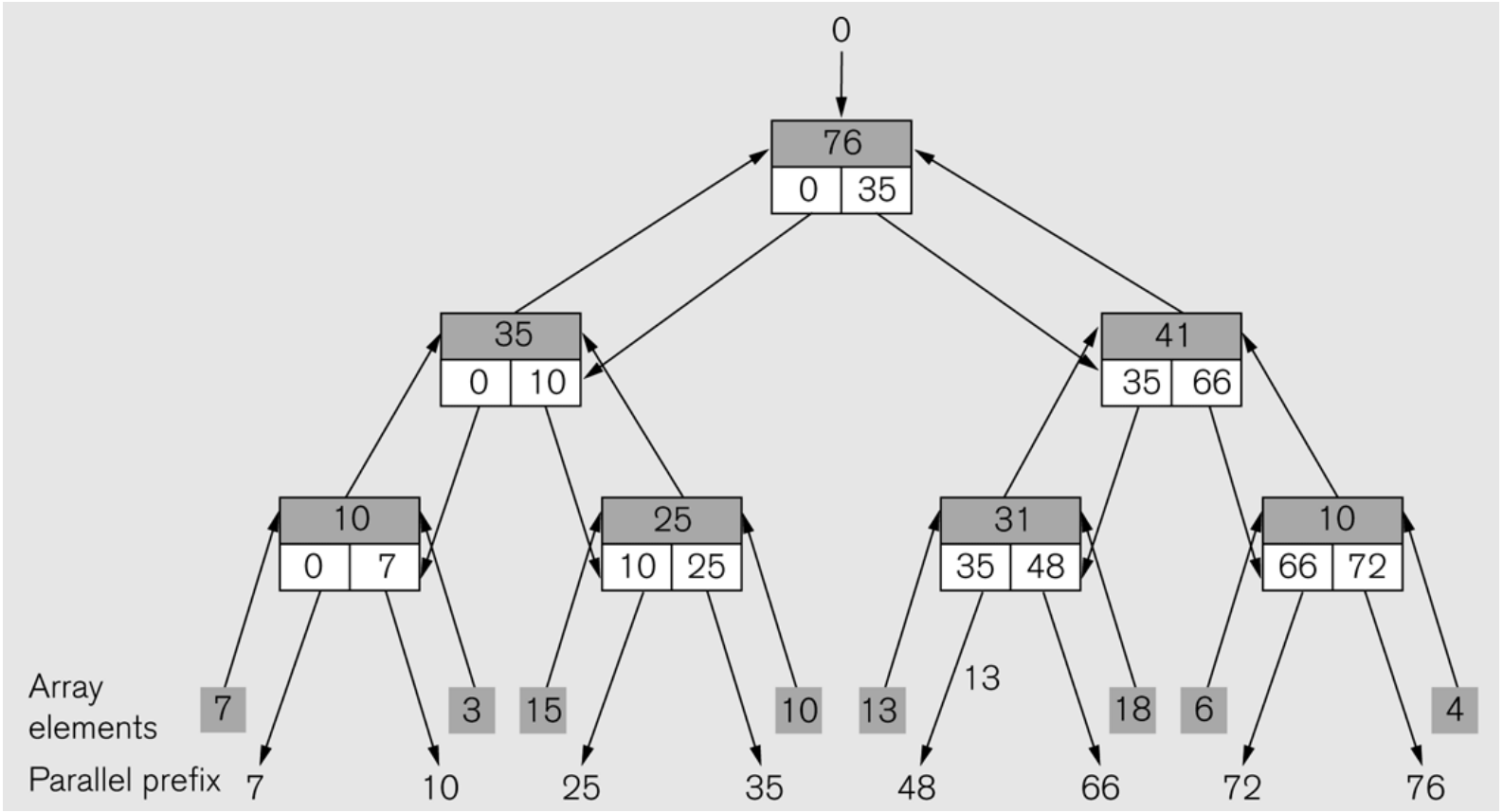
从问题描述开始设计并行算法

- 方法描述
  - 从问题本身描述出发，不考虑相应的串行算法，设计一个全新的并行算法。
- 评注
  - 挖掘问题的固有特性与并行的关系；
  - 设计全新的并行算法是一个挑战性和创造性的工作。

求前缀和

- 数组A = { a_0, a_1, a_2, … a_n}
- 求数组B = { b_0, b_1, b_2, … b_n}
  - 令b_i = sum(a_0 to a_i)

并行求前缀和


0
76
0 35
35
0 10
41
35 66
10
0 7
25
10 25
31
35 48
10
66 72
Array elements
7 3 15 10 13 13 18 6 4
Parallel prefix
7 10 25 35 48 66 72 76

借用已有算法求解新问题

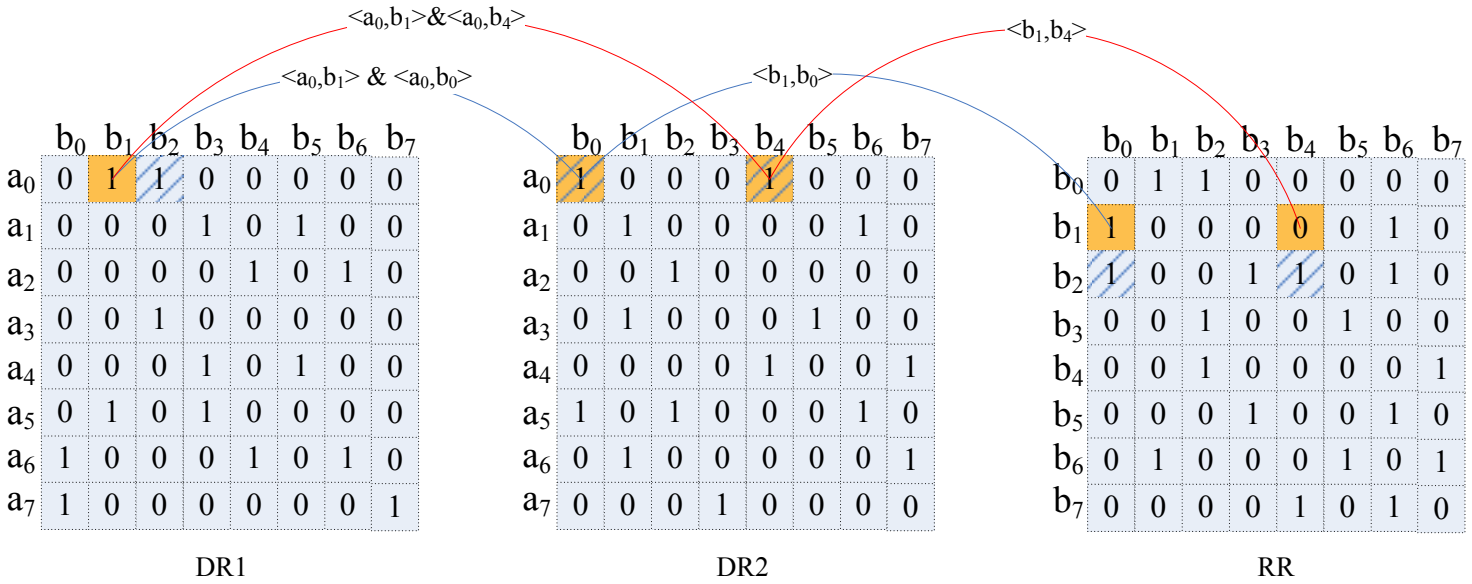
- 方法描述
  - 找出求解问题和某个已解决问题之间的联系；
  - 改造或利用已知算法应用到求解问题上。
- 评注
  - 这是一项创造性的工作。

3PCF（三点相关函数）

- 定义：
  - 点集D、R。
  - 定义D中的点为$a_i \in D$，R中的点为$b_i \in R$。
  - 距离：r_1、r_2、r_3、err
- 求：
  - 满足以下条件的三元组（空间中三角形）的数目
    - $<a_i, b_m, b_n>$，$|a_i-b_m|=r_1 \pm err$且$|a_i-b_n|=r_2 \pm err$且$|b_m-b_n|=r_3 \pm err$
- 原始解法：
  - 对于D中每一点a_i，在R中找到与之距离为r_1的点集R’，找到与之距离为r_2的点集R’’。在点集R’与R’’中，查找两点间距离为r_3的点组数目。累加。

3PCF

- 阶段一：
  - 遍历D中每一点，在R中找到与之相距为r1或r2的点，获得矩阵DR1和DR2；
  - 遍历R中每一点，在R中找到与之相距为r3的点，形成矩阵RR
- 阶段二： ResultMatrix = DR1×(DR2×RR^T)^T = DR1×RR×DR2^T



Outline

- 并行算法设计
- **PCAM方法学**
- 并行程序设计模式
- 并行计算与软件工程

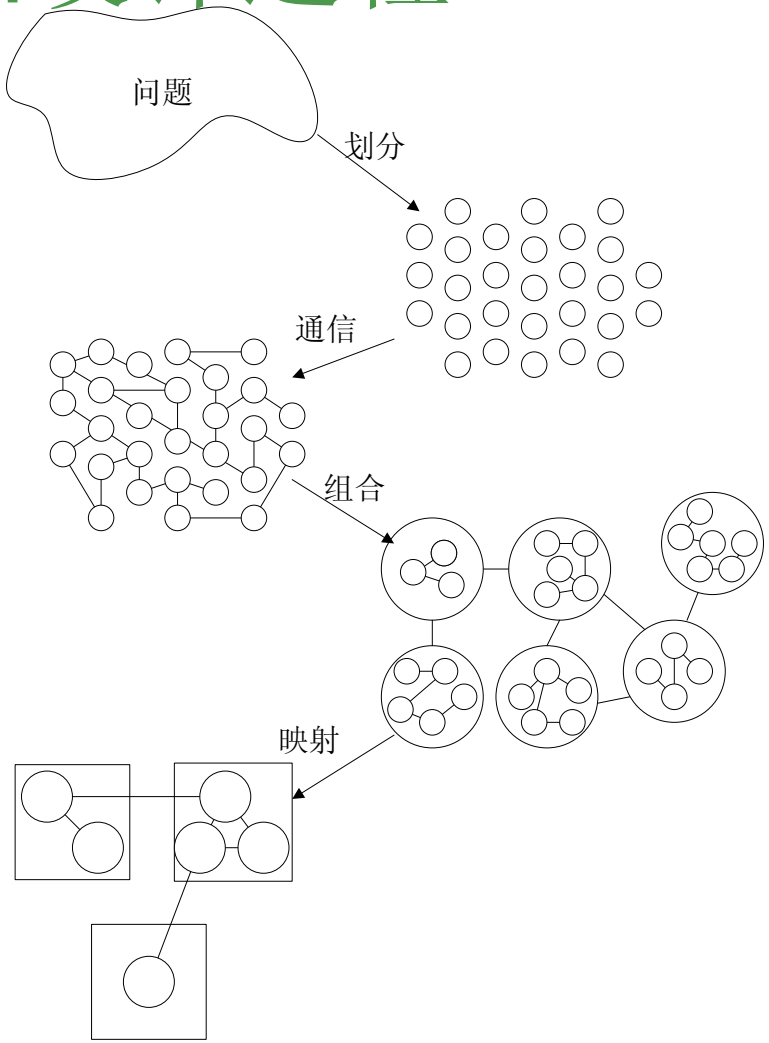
PCAM设计方法学

- 划分(Partitioning)
- 通讯(Communication)
- 组合(Agglomeration)
- 映射(Mapping)
- 小结

PCAM设计方法学

- 设计并行算法的四个阶段
  - 划分(Partitioning)
  - 通讯(Communication)
  - 组合(Agglomeration)
  - 映射(Mapping)
- 划分：分解成小的任务，开拓并发性；
- 通讯：确定诸任务间的数据交换，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。

PCAM设计过程



PCAM设计方法学

- **划分(Partitioning)**
- 通讯(Communication)
- 组合(Agglomeration)
- 映射(Mapping)
- 小结

划分

- 方法描述
- 域分解
- 功能分解
- 划分判据

划分方法描述

- 充分开拓算法的并发性和可扩放性；
- 先进行数据分解(称域分解)，再进行计算功能的分解(称功能分解)；
- 使数据集和计算集互不相交；
- 划分阶段忽略处理器数目和目标机器的体系结构；
- 能分为两类划分：
  - 域分解(domain decomposition)
  - 功能分解(functional decomposition)

域分解

- 划分的对象是数据，可以是算法的输入数据、中间处理数据和输出数据；
- 将数据分解成大致相等的小数据片；
- 划分时考虑数据上的相应操作；
- 如果一个任务需要别的任务中的数据，则会产生任务间的通讯；

域分解

- 示例：三维网格的域分解，各格点上计算都是重复的。下图是三种分解方法：

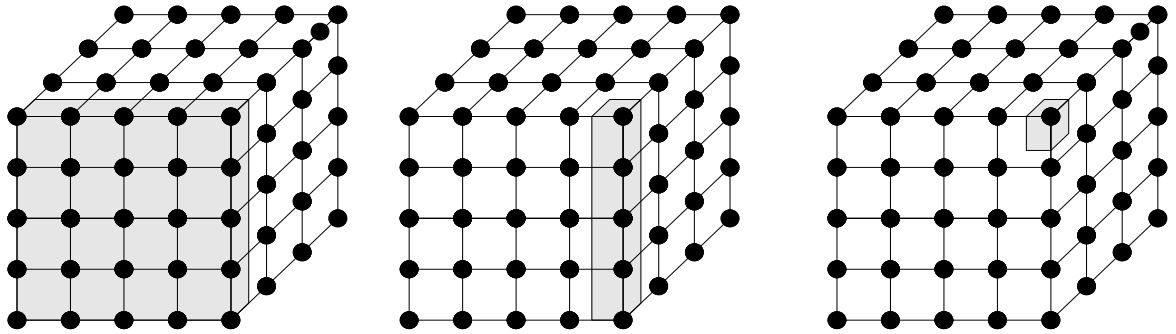


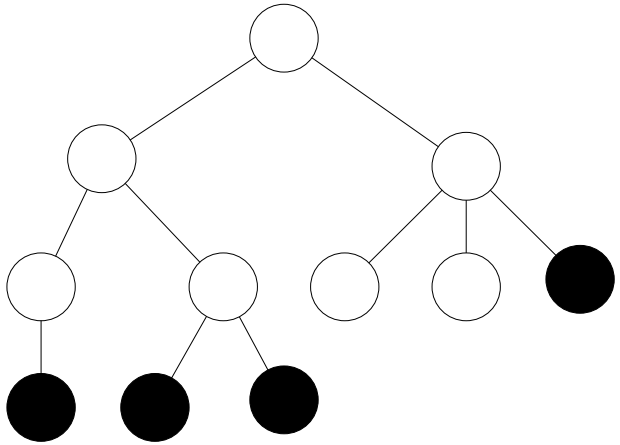
图7.2

功能分解

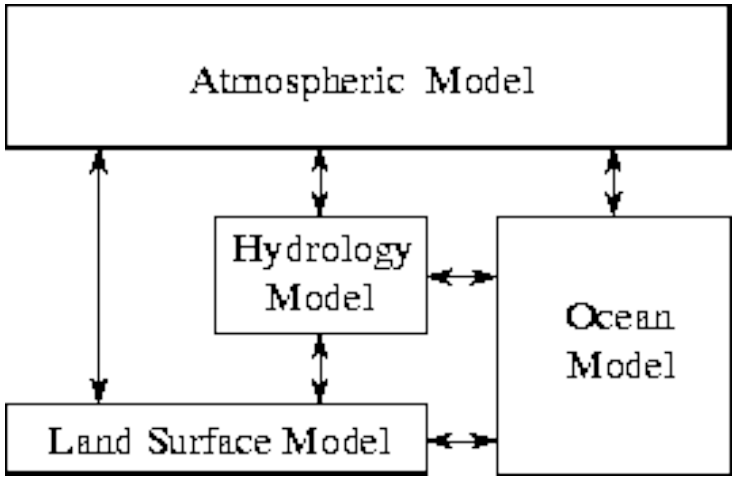
- 划分的对象是计算，将计算划分为不同的任务，其出发点不同于域分解；
- 划分后，研究不同任务所需的数据。如果这些数据不相交的，则划分是成功的；如果数据有相当的重叠， 意味着要重新进行域分解和功能分解；
- 功能分解是一种更深层次的分解。

功能分解

- 示例1：搜索树



- 示例2：气候模型



划分判据

- 划分是否具有灵活性？
- 划分是否避免了冗余计算和存储？
- 划分任务尺寸是否大致相当？
- 任务数与问题尺寸是否成比例？
- 功能分解是一种更深层次的分解，是否合理？

PCAM设计方法学

- 划分(Partitioning)
- **通讯(Communication)**
- 组合(Agglomeration)
- 映射(Mapping)
- 小结

通讯

- 方法描述
- 四种通讯模式
- 通讯判据

通讯方法描述

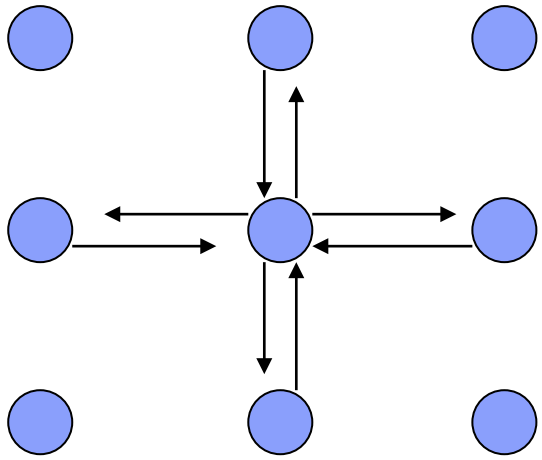
- 通讯是PCAM设计过程的重要阶段；
- 划分产生的诸任务，一般不能完全独立执行，需要在任务间进行数据交流；从而产生了通讯；
- 功能分解确定了诸任务之间的数据流；
- 诸任务是并发执行的，通讯则限制了这种并发性；

四种通讯模式

- 局部/全局通讯
- 结构化/非结构化通讯
- 静态/动态通讯
- 同步/异步通讯

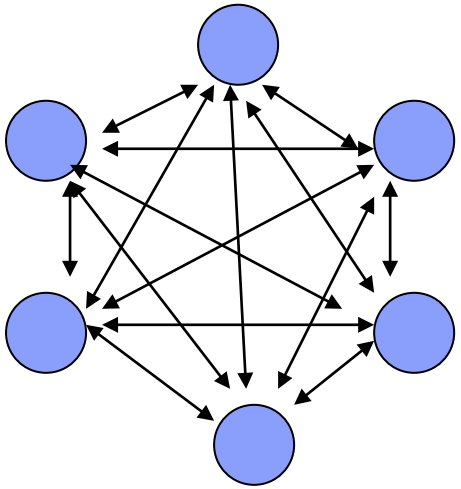
局部通讯

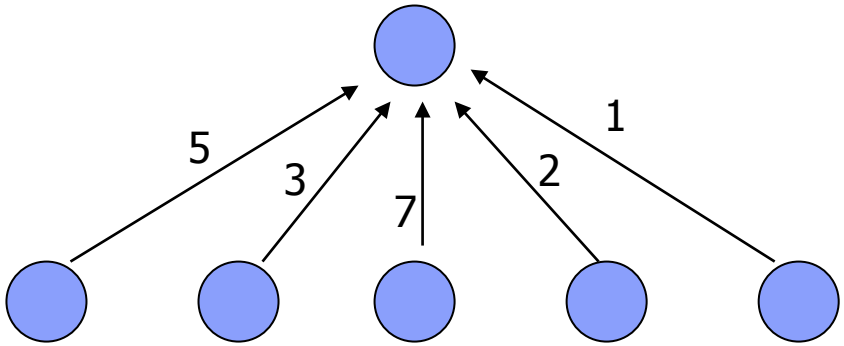
- 通讯限制在一个邻域内



全局通讯

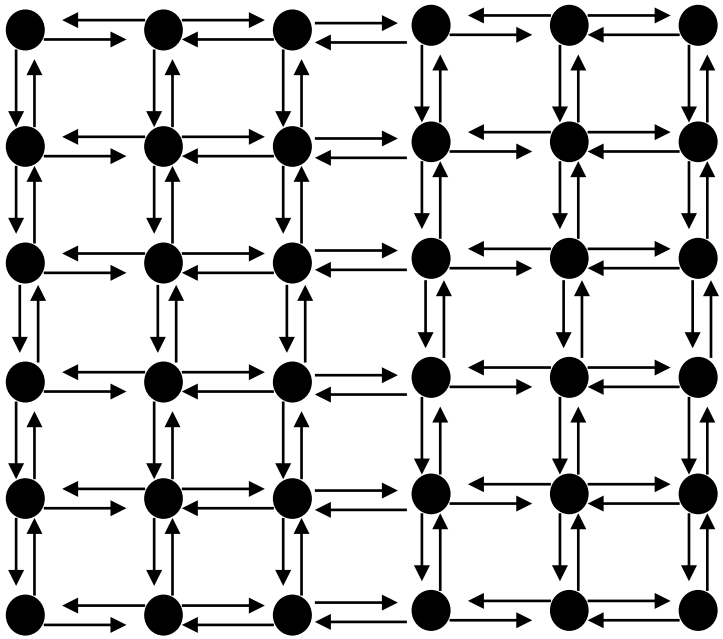
- 通讯非局部的
- 例如：
  - All to All
  - Master-Worker

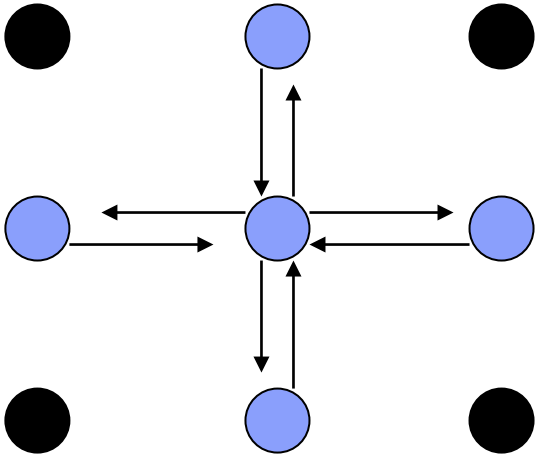




结构化通讯

- 每个任务的通讯模式是相同的；
- 下面是否存在一个相同通讯模式？





通讯判据

- 所有任务是否执行大致相当的通讯?
- 是否尽可能的局部通讯?
- 通讯操作是否能并行执行?
- 同步任务的计算能否并行执行?

PCAM设计方法学

- 划分(Partitioning)
- 通讯(Communication)
- **组合(Agglomeration)**
- 映射(Mapping)
- 小结

组合

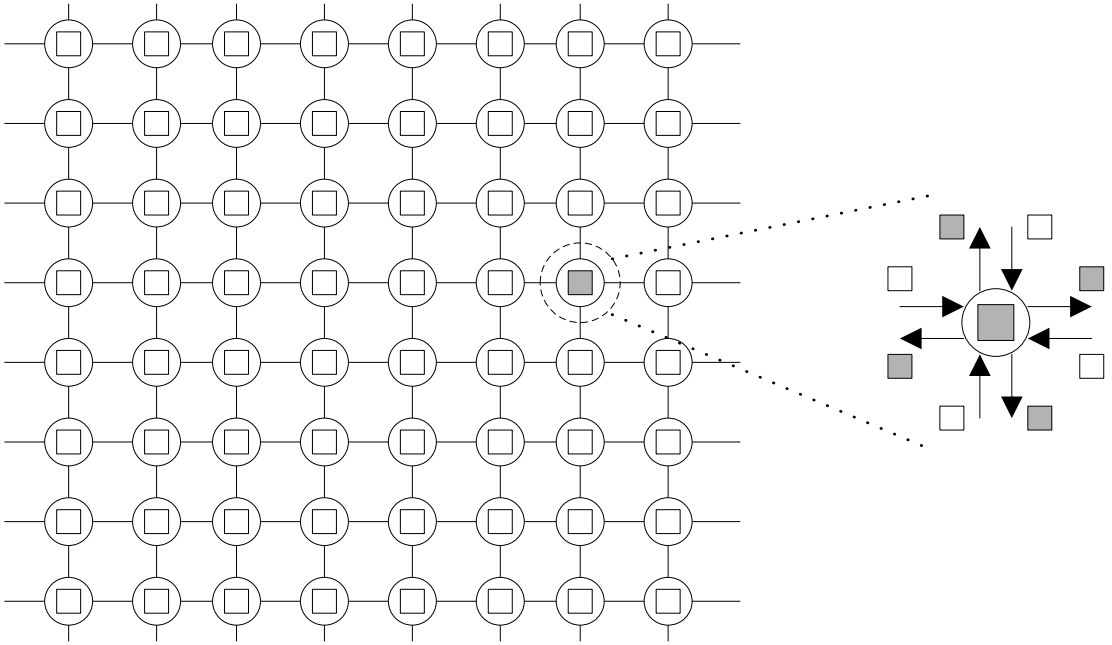
- 方法描述
- 表面-容积效应
- 重复计算
- 组合判据

方法描述

- 组合是由抽象到具体的过程，是将组合的任务能在一类并行机上有效的执行；
- 合并小尺寸任务，减少任务数。如果任务数恰好等于处理器数，则也完成了映射过程；
- 通过增加任务的粒度和重复计算，可以减少通讯成本；
- 保持映射和扩展的灵活性，降低软件工程成本；

表面-容积效应

- 通讯量与任务子集的表面成正比，计算量与任务子集的体积成正比；
- 增加重复计算有可能减少通讯量；

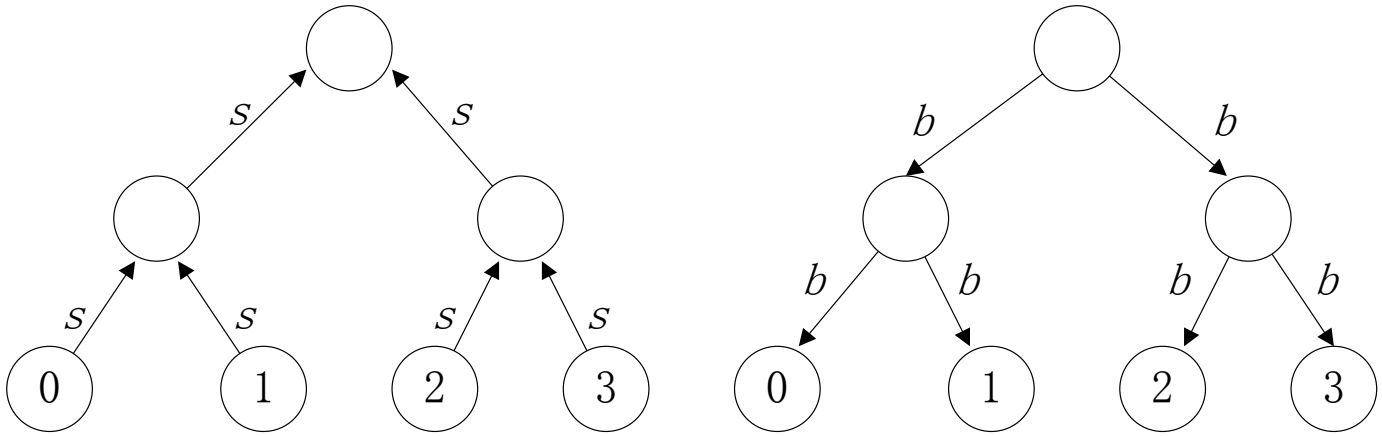


重复计算

- 重复计算减少通讯量，但增加了计算量，应保持恰当的平衡；
- 重复计算的目标应减少算法的总运算时间；

重复计算

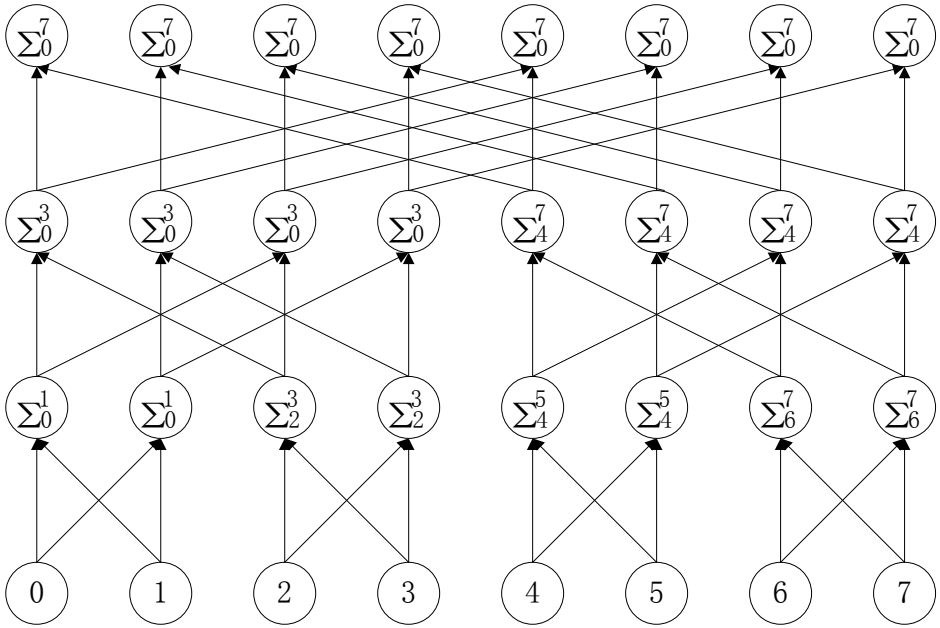
- 示例：二叉树上N个处理器求N个数的全和，要求每个处理器均保持全和。



二叉树上求和，共需2logN步

重复计算

- 示例：二叉树上N个处理器求N个数的全和，要求每个处理器均保持全和。



蝶式结构求和，使用了重复计算，共需logN步

组合判据

- 增加粒度是否减少了通讯成本？
- 重复计算是否已权衡了其得益？
- 是否保持了灵活性和可扩放性？
- 组合的任务数是否与问题尺寸成比例?
- 是否保持了类似的计算和通讯？
- 有没有减少并行执行的机会？

PCAM设计方法学

- 划分(Partitioning)
- 通讯(Communication)
- 组合(Agglomeration)
- **映射(Mapping)**
- 小结

映射

- 方法描述
- 负载平衡算法
- 任务调度算法
- 映射判据

方法描述

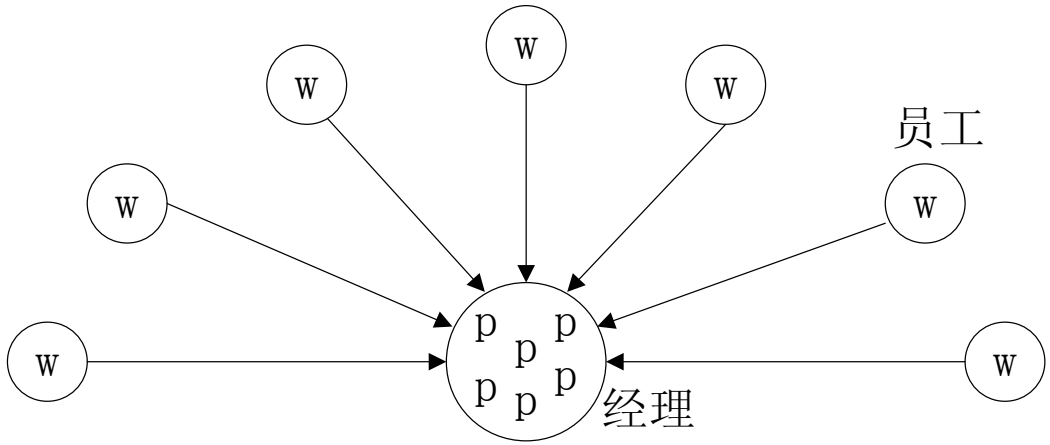
- 每个任务要映射到具体的处理器，定位到运行机器上；
- 任务数大于处理器数时，存在负载平衡和任务调度问题；
- 映射的目标：减少算法的执行时间
  - 并发的任务 → 不同的处理器
  - 任务之间存在高通讯的 → 同一处理器
- 映射实际是一种权衡，属于NP完全问题

负载平衡算法

- 静态的： 事先确定；
- 概率的： 随机确定；
- 动态的： 执行期间动态负载；
- 基于域分解的：
  - 递归对剖
  - 局部算法
  - 概率方法
  - 循环映射

任务调度算法

- 任务放在集中的或分散的任务池中，使用任务调度算法将池中的任务分配给特定的处理器。下面是两种常用调度模式：
- 经理/雇员模式
- 非集中模式



映射判据

- 采用集中式负载平衡方案，是否存在通讯瓶颈？
- 采用动态负载平衡方案，调度策略的成本如何？

PCAM设计方法学

- 划分(Partitioning)
- 通讯(Communication)
- 组合(Agglomeration)
- 映射(Mapping)
- 小结

小 结

- 划分
  - 域分解和功能分解
- 通讯
  - 任务间的数据交换
- 组合
  - 任务的合并使得算法更有效
- 映射
  - 将任务分配到处理器，并保持负载平衡

Outline

- 并行算法设计
- PCAM方法学
- **并行程序设计模式**
- 并行计算与软件工程

并行程序设计模式

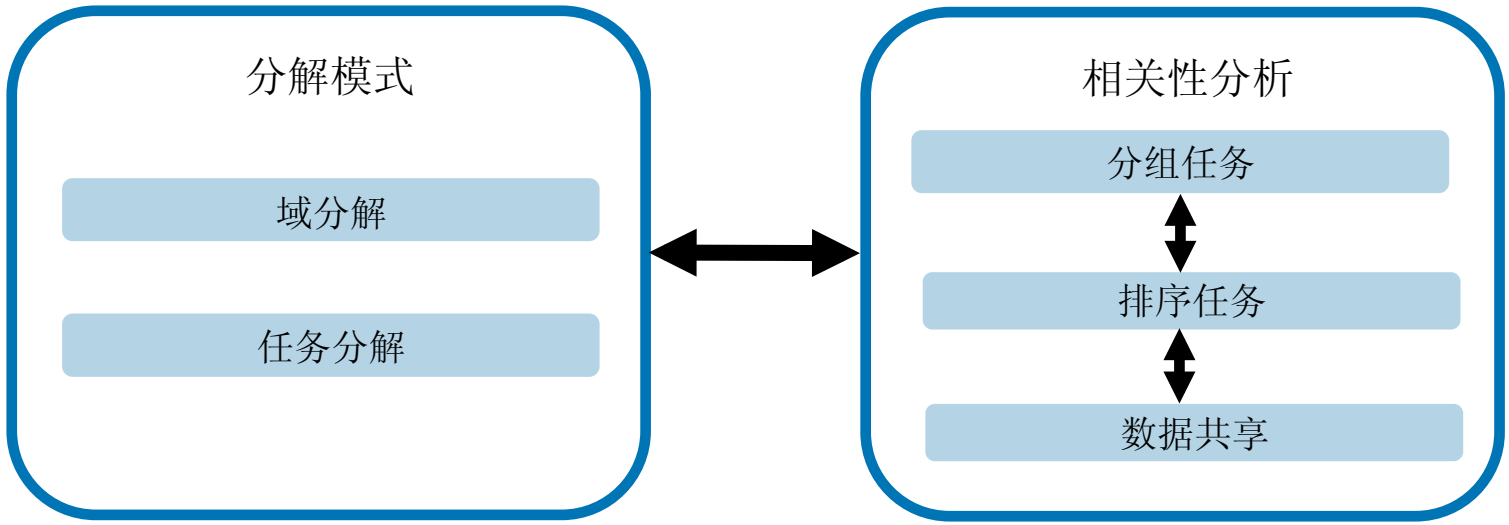
- 设计模式
  - 设计模式描述了在特定条件下解决重复出现问题的有效方法，能够将成功的经验记录下来，供面临相似问题的人参考。每一个模式都包括模式名称、问题、解决方案、效果四个基本要素。
- 专门针对并行程序设计的模式则是并行程序设计模式

并行程序设计模式

- 分析并发性
- 算法结构
- 支持结构
- 实现机制

分析并发性

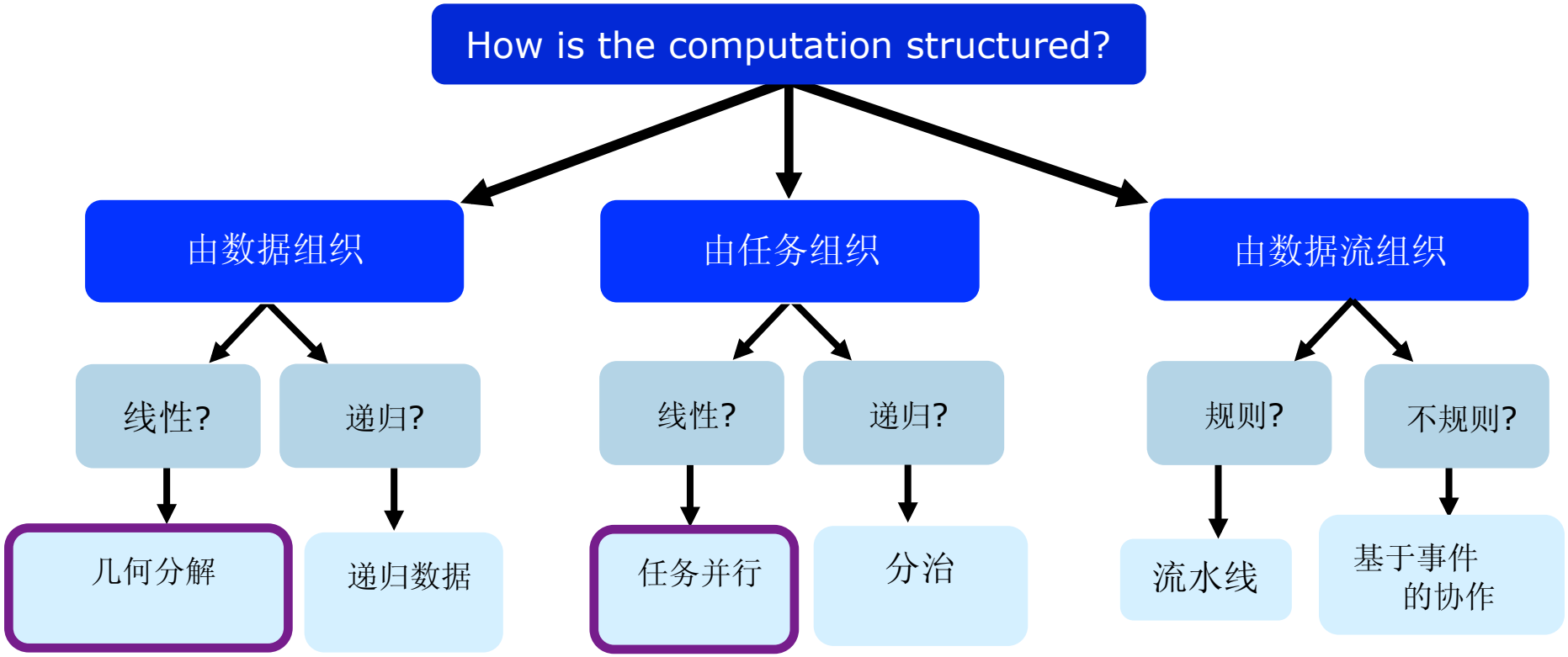
- 关注应用问题的宏观分析，完成问题的分解



算法结构

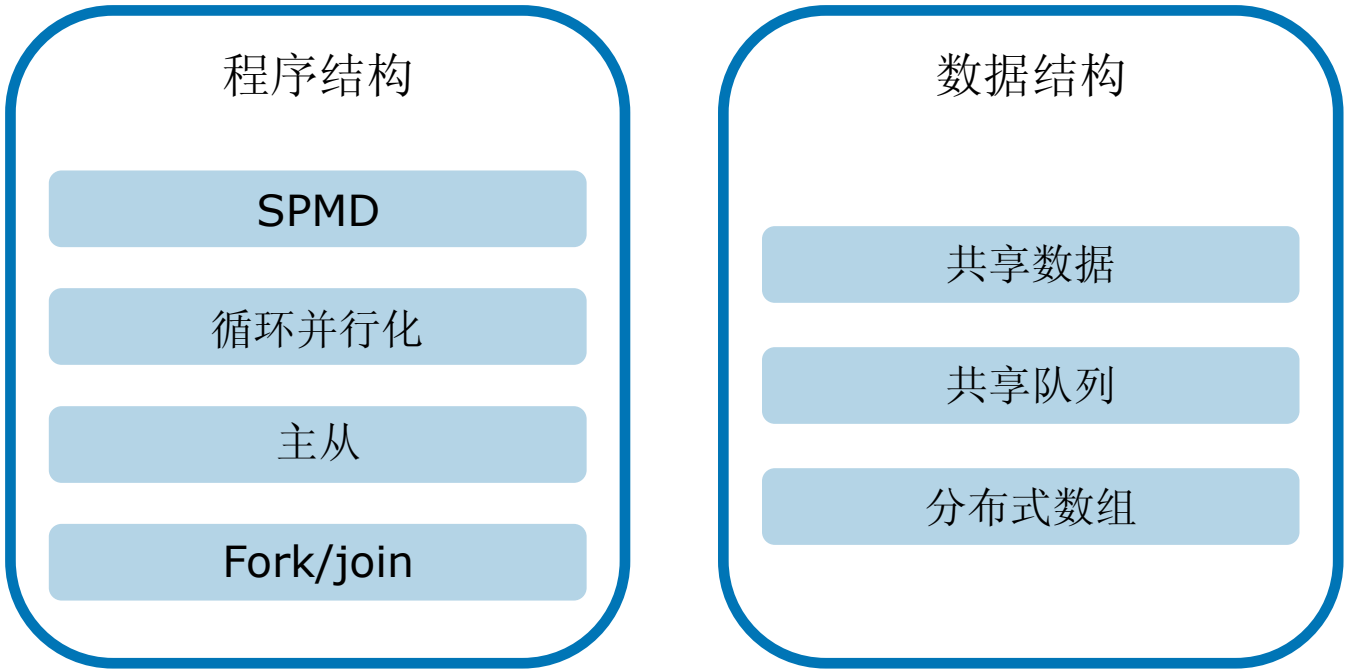
- 细化设计方案，使得设计更接近能够并行执行的程序

“算法结构”设计空间的决策树



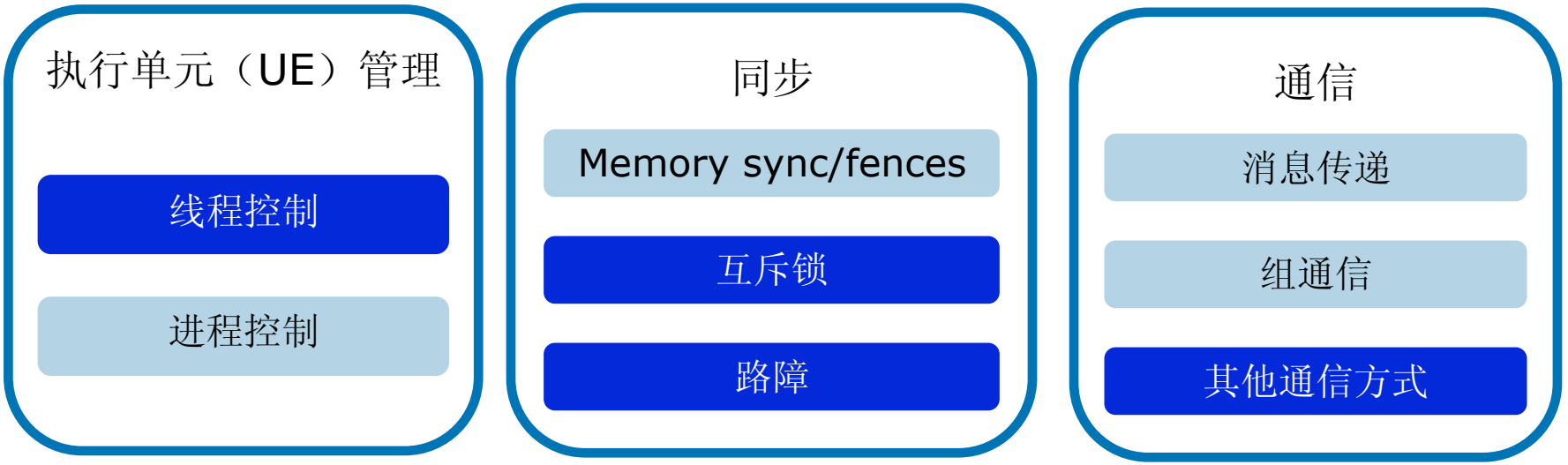
支持结构

- 算法转换为程序

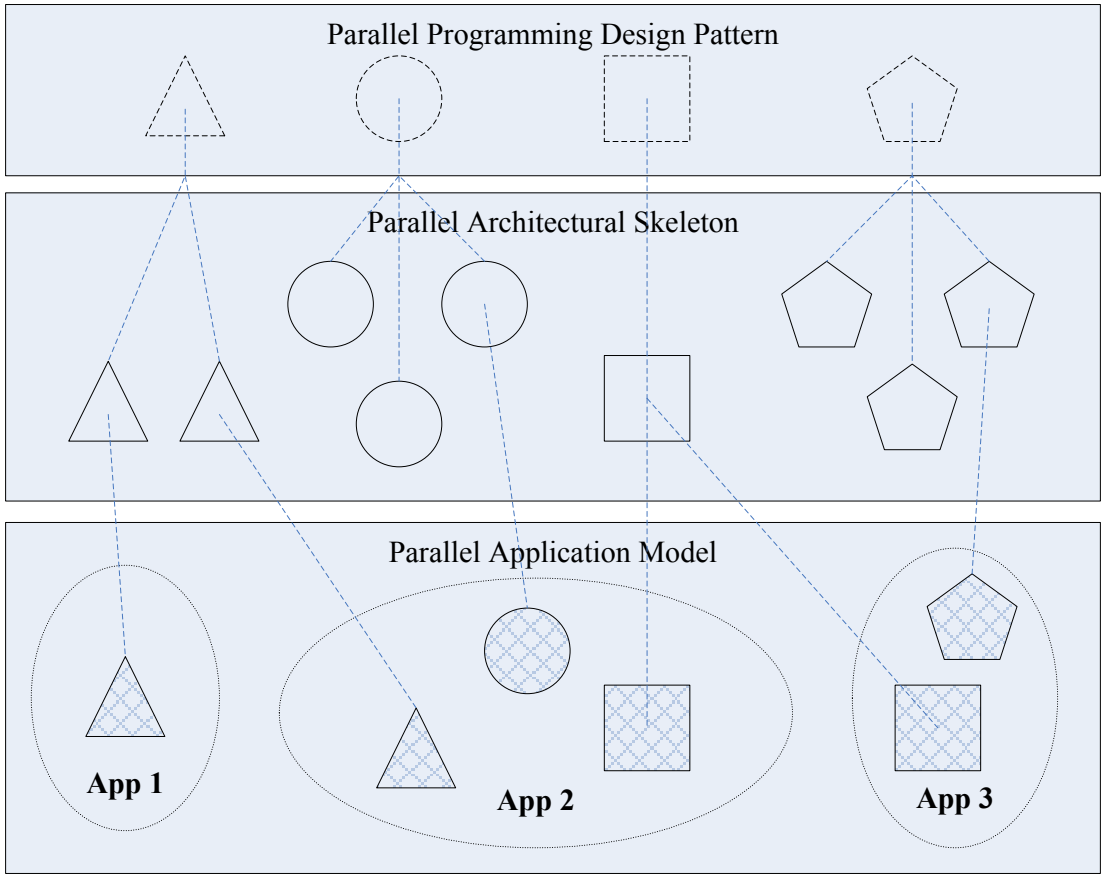


实现机制

- 将高层次空间中的模式映射到具体的并行计算环境，用于描述进程及线程的管理和数据交换等，一般不直接表现为模式。



模式复用


Parallel Programming Design Pattern
Parallel Architectural Skeleton
Parallel Application Model
App 1
App 2
App 3

Outline

- 并行算法设计
- PCAM方法学
- 并行程序设计模式
- **并行计算与软件工程**

逻辑与计算分离的软件工程方法

- 软件工程的发展趋势是系统功能的细化以及模块接口的标准化。
- 并行计算硬件技术的发展趋势也表明了软件体系结构功能细化和模块接口标准化的迫切要求，当一块处理器内封装成百上千的计算核心，运行在其之上的软件必须要由分工明确众多模块构成，方能够充分利用硬件的资源。
- 将软件系统业务逻辑中需要且能够并行化的部分分离，从而使并行化的工作转化为这些可并行计算部分的内部问题，使用专用的工具和平台进行设计与开发。分离可并行化计算的工作贯穿于软件工程过程的各个阶段。

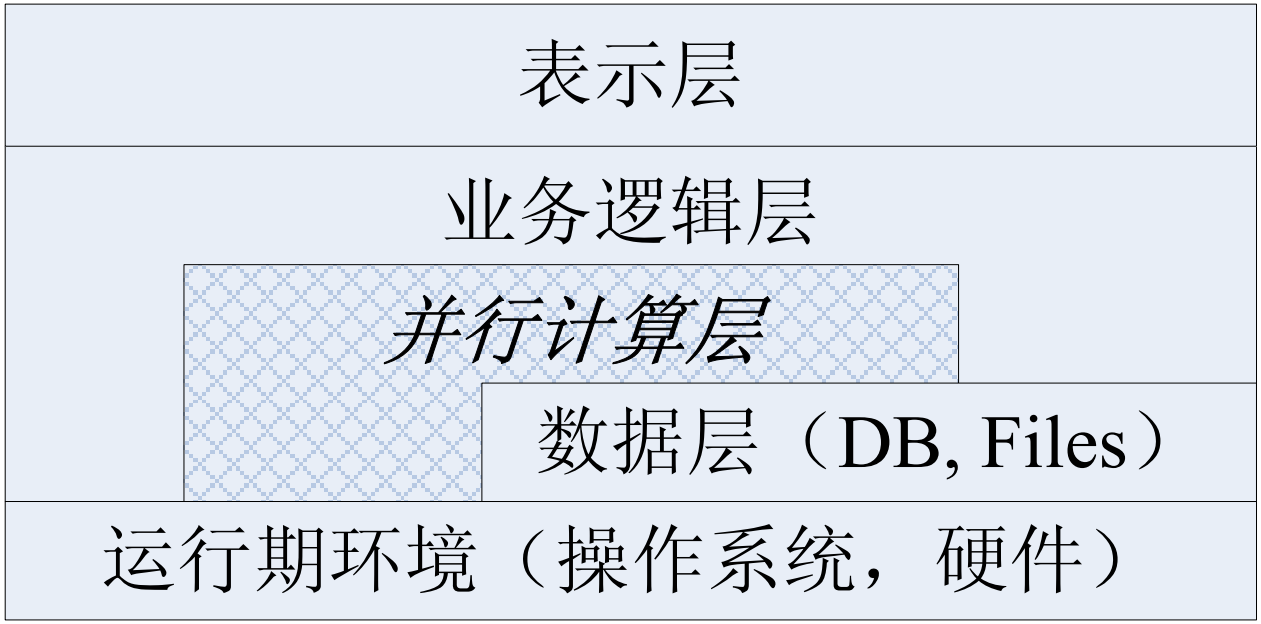
PCO，Parallel Computing Offload

- 业务逻辑与可并行化计算分离，是指在软件的设计与实现过程中，对软件系统的体系结构进行明确划分，使得软件中需要且能够进行并行计算的部分成为独立模块。其中，“业务逻辑”实现软件的各项具体功能，接收用户的输入数据，进行处理，并生成结果保存或返回给用户；“可并行化计算”是“业务逻辑”处理过程中的具体环节，通常计算量较大且能够并行化。

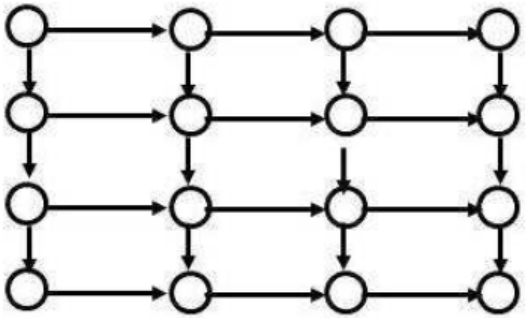
分离原则

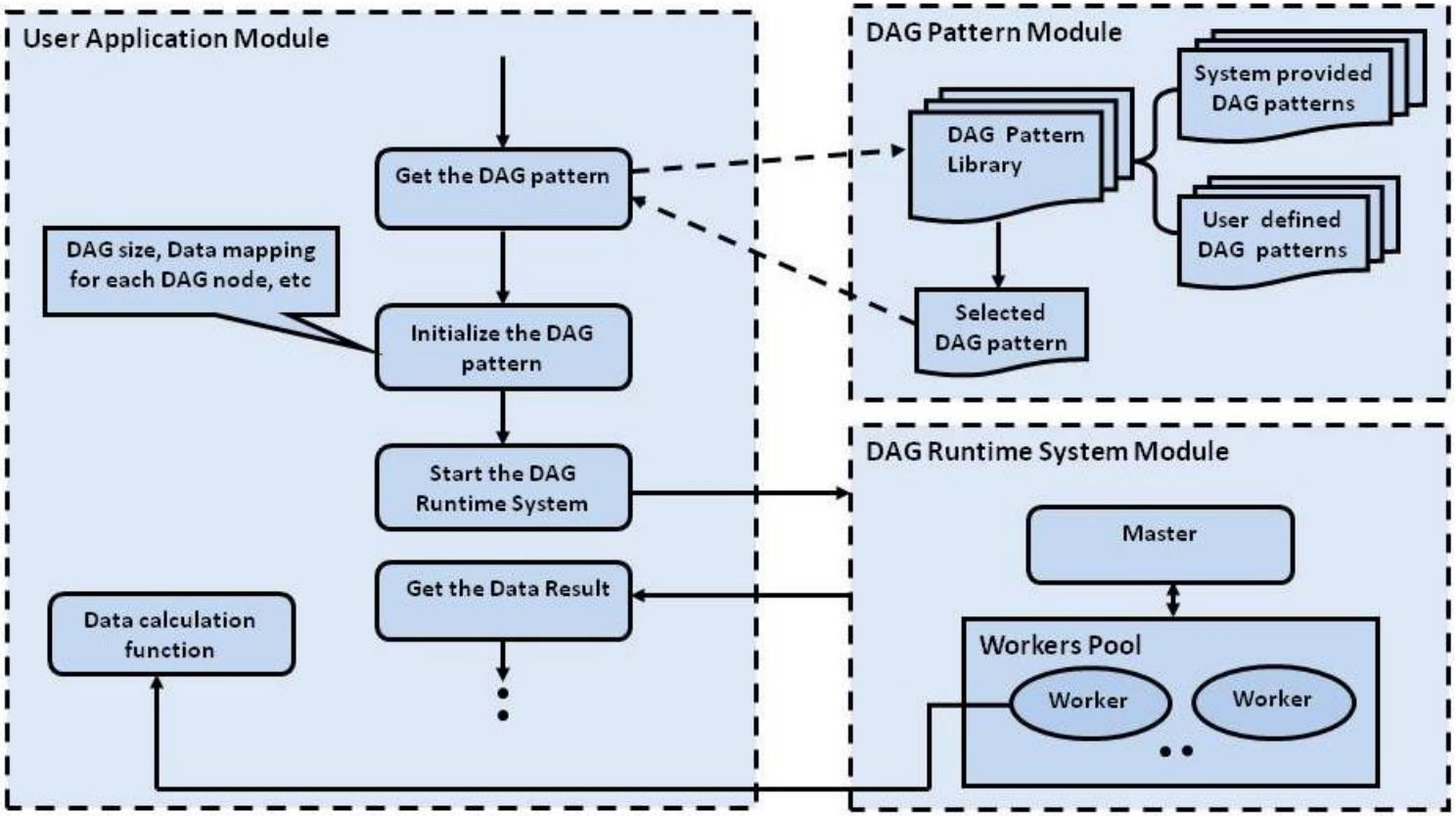
- 在需求分析阶段明确软件的目标运行环境。
- 设计软件总体体系结构的软件架构师要精通并行计算技术。
- 安排专人设计实现软件中并行计算部分。
- 为并行计算部分设计详尽的日志机制。
- 单独为并行计算部分设计测试用例以及维护方案。
- 并行计算模块的设计实现需要支持封装、复用、互操作及自治。

PCO软件体系结构层次参考模型



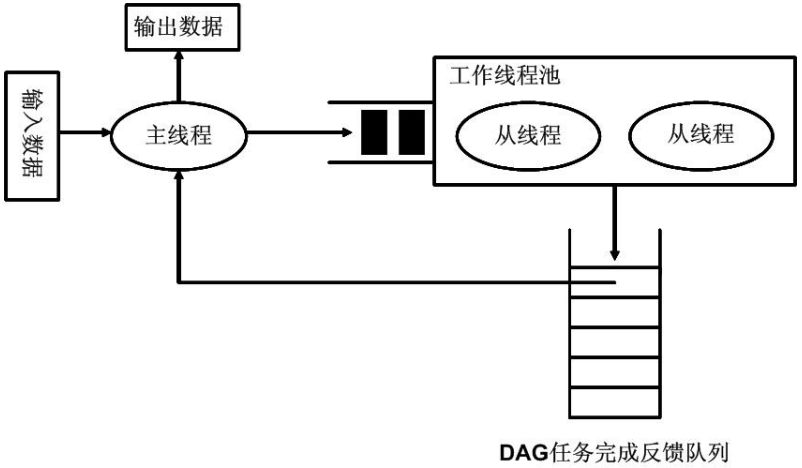
DAG Data Driven Model

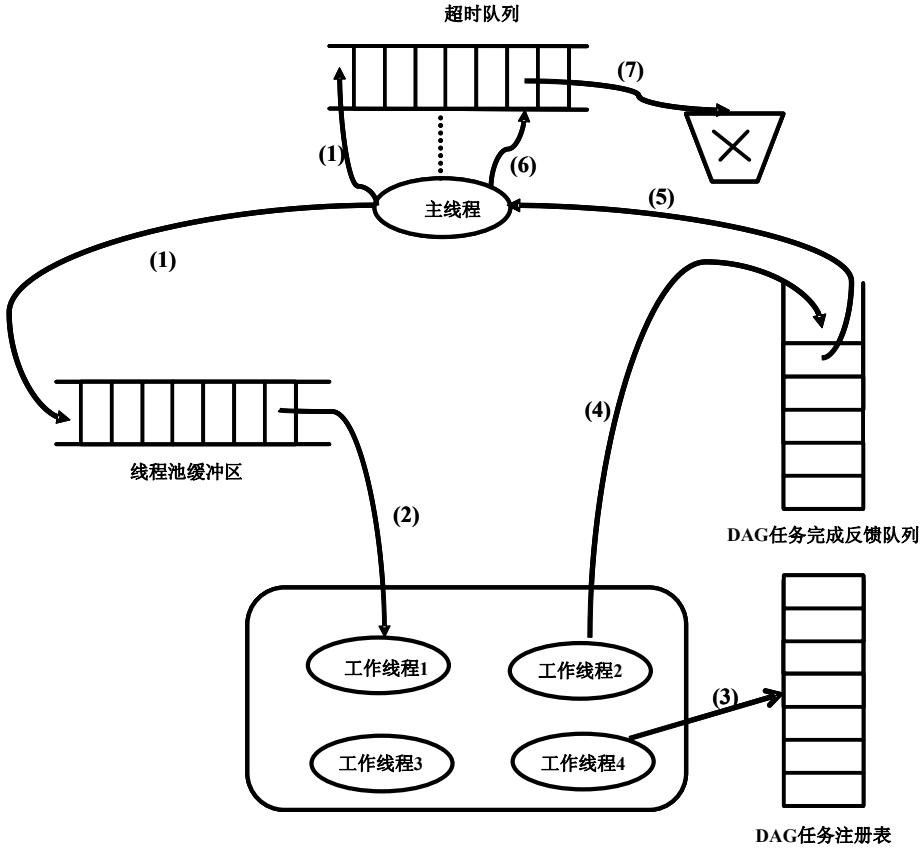




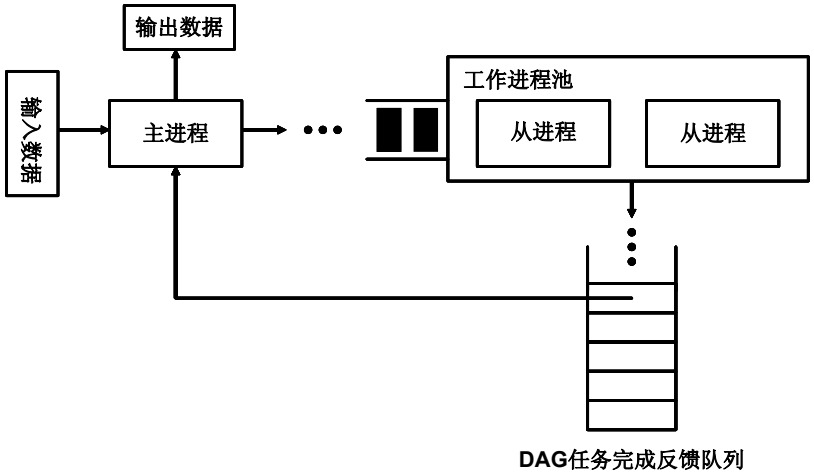
EasyPDP

- 用于多线程并行计算环境
- 动态任务调度

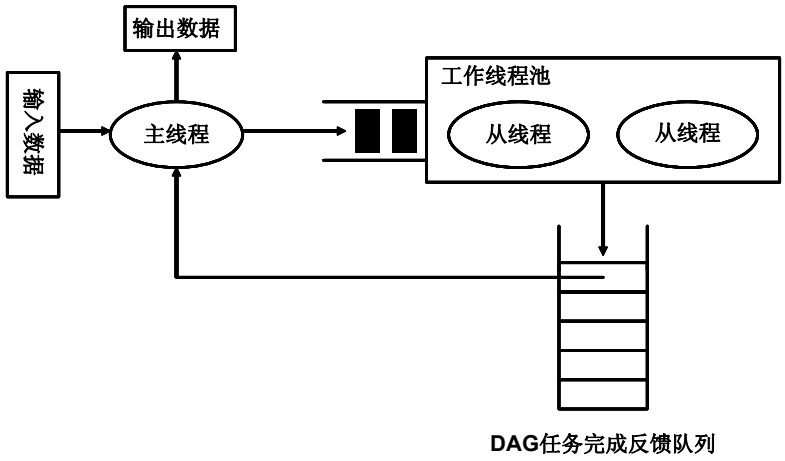




EasyHPS



(a). 进程之间的数据控制流



(b). 进程内部的数据控制流