并行与分布式计算基础 Part1——basic

邓喻源 数学科学学院

什么是并行计算?

1.问题被分为能够并发执行部分
2.每个部分进一步被分为一系列指令
3.来自不同部分的指令可以同时在不同处理器上执行
4.需要全局协同机制

计算器硬件体系架构

1.**指令集架构(Instruction Set Architecture, ISA)**
主要指处理器所支持的机器语言(指令) 的种类、格式、长度等，以及内存与寄存器的抽象模型等，例子：x86, alpha, MIPS, RISC-V、ARM ...
2.**微架构(Micro-architecture, μarch)**
主要指ISA 的一种具体的处理器实现，比如处理器核数、缓存大小、流水线长度等，例子：Intel Xeon E5 处理器, ...
3.**系统架构(System Architecture)**
主要指与处理器不直接相关的其他部分，比如访存、I/O、网络、软件等。

计算能力与存储能力的度量

前缀	简称	量级	计算能力	存储能力
Kilo-	K	10^3	KiloFLOPS (KFLOPS)	KiloByte (KB)
Mega-	M	10^6	MegaFLOPS (MFLOPS)	MegaByte (MB)
Giga-	G	10^9	GigaFLOPS (GFLOPS)	GigaByte (GB)
Tera-	T	10^{12}	TeraFLOPS (TFLOPS)	TeraByte (TB)
Peta-	P	10^{15}	PetaFLOPS (PFLOPS)	PetaByte (PB)
Exa-	E	10^{18}	ExaFLOPS (EFLOPS)	ExaByte (EB)
Zetta-	Z	10^{21}	ZettaFLOPS (ZFLOPS)	ZettaByte (ZB)
Yotta-	Y	10^{24}	YottaFLOPS (YFLOPS)	YottaByte (YB)

其中，FLOPS (flops or flop/s) 指每秒浮点运算次数：floating point operations per second.

冯诺依曼结构

主要组成部分
控制单元：解释指令
处理单元：执行指令
内存：存储数据和指令
输入/输出：与外界交互

指令是执行步骤

四个步骤：取指、译码、执行、写回

提高处理器性能的手段

多级存储技术

小（快）=>大（慢）：寄存器=> 各级缓存=> ... => 内存!=>外存

其他手段

简化指令 指令级并行 数据级并行

并行计算的分类方式

福林分类：

根据（单/多）指令流+（单/多）数据流分类

SISD：传统串行任务（单用户单任务）。

SIMD：图像处理、深度学习的矩阵运算。

MISD：容错计算、加密。

MIMD：大规模并行处理任务（如超级计算、分布式数据库）

不同的架构

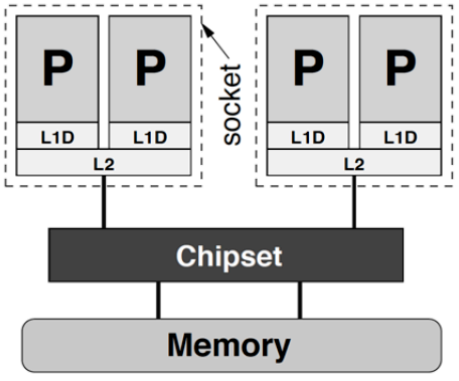
UMA（一致内存访问架构）

UMA 是共享内存并行机的一种内存访问架构，其特点是**所有处理器访问内存的速度相同。**在这种架构中，内存是全局共享的，各个处理器通过相同的路径访问内存。

几路几核?

一个插槽为一路，每个插槽有几核（核数不是总的，而是每个插槽的数量）

双路双核：



ccNUMA

ccNUMA结合了缓存一致性和非一致内存访问的特点。它通过高速互联将多个处理器节点和其各自的本地内存连接起来，每个节点访问本地内存的速度更快，而访问远程节点内存则需经过较高延迟。同时，系统通过缓存一致性协议（如 MESI 协议）确保多个处理器缓存中的数据一致性，从而保证程序运行的正确性。ccNUMA 具有良好的扩展性和性能优势，适用于高性能计算、大规模并行处理任务和多线程应用。

并行三大定律

阿姆达尔定理

阿姆达尔定理可以表示为以下公式：

$$S = \frac{1}{(1-P) + \frac{P}{N}}$$

其中：S是并行化后的加速比（Speedup），P是程序中可并行化的部分比例（0 ≤ P ≤ 1），$1-P$是程序中必须串行执行(也就是无法并行化计算)的部分比例，N是并行处理器的数量。

要点：如果W是任务量，那么也就是说，有$(1-P)W$的任务只能使用一个处理器

古斯塔法森定律

设P是程序中不可并行化的部分比例。假设该任务的工作量可以随着处理器个数缩放，从而保持处理时间固定。则对任意n 个处理器，相比于1 个处理器，能够取得的加速比$S'(n)$不存在上界

要点：即$W_n = PW + (1-P)nW$,$S'(n) = W_n/W$

因为处理时间一样，所以在这里，衡量加速比的标准是W_n的多少

孙-倪定律

孙-倪定律是对古斯塔夫森定律的进一步扩展，考虑了**并行部分工作量扩展比例可能不是线性的情况。**它假设任务中可并行部分的工作量随处理器数量 n 的变化不一定严格线性，而是按照一个函数 $G(n)$ 进行缩放。

孙-倪定律的加速比 $S^*(n)$ 由以下公式给出：

$$S^*(n) = \frac{\alpha + (1-\alpha)G(n)}{\alpha + (1-\alpha)\frac{G(n)}{n}}$$

	加速比 $(n \to \infty)$	并行效率 $(n \to \infty)$
阿姆达尔	$S(n) \to \frac{1}{\alpha}$	$E(n) \to 0$
古斯塔法森	$S'(n) \to \infty$	$E'(n) \to 1-\alpha$
孙-倪 $(G(n) > O(n))$	$S^*(n) \to \infty$	$E^*(n) \to 1$

影响性能的主要因素是什么?

从计算、访存的关系考虑

标准：计算密度=flop/byte

Roofline模型衡量影响性能的是访问还是计算

从多级存储的角度考虑

需要考虑命中率/失效率对时间的影响

这就引入了单层和多层（只有一级缓存/有多级缓存）的AMAT模型。

单层：$AMAT = (1-r)Ts + r(T_s + T_M)$ 即为： $AMAT = T_s + rT_M$ 其中：T_s：缓存命中时间（Hit Time），即访问缓存所需的时间。T_M：缓存失效损失（Miss Penalty），即缓存失效后需要访问主内存的时间。r：缓存失效率（Miss Rate），表示缓存中未命中请求的比例。

多层：对于多级缓存（如 L1、L2、L3 缓存）的系统，访问内存的平均时间需要依次考虑每一级缓存的访问时间和失效率：

- **两层缓存**：

$$AMAT_2 = T_1 + r_1(T_2 + r_2 T_M)$$

化简为：

$AMAT_2 = T_1 + R_1 T_2 + R_2 T_M$

其中$R_1 = r_1$，$R_2 = r_1 r_2$ 是$L1$ 和$L2$ 的整体失效率。

- **三层缓存**：

$$AMAT_3 = T_1 + r_1[T_2 + r_2(T_3 + r_3 T_M)]$$

化简为：

$AMAT_3 = T_1 + R_1 T_2 + R_2 T_3 + R_3 T_M$

其中$R_1 = r_1$，$R_2 = r_1 r_2$，$R_3 = r_1 r_2 r_3$是$L1$ 、$L2$、$L3$ 的整体失效率。

注：**局部失效率**指该层次缓存失效的概率，**整体失效率**表示该层次缓存
以及其上层所有缓存同时失效的概率。

从多核并行角度考虑

PRAM模型：处理器共享连续内存空间，执行独立的指令，并且执行任意一种计算或者访存的操作时间开销都一样。

关键参数：处理器个数p，单位执行时间γ

PRAM忽略了太多机制，常常不太准确，因此有加强版本PHM

关于网络的问题

基本概念：

路由(router)：决定网络通信策略的算法或设备。
延迟(latency)：从一个计算节点到另一个计算节点的数据传输时间。

跳(hop)：拓扑网络上一点到另一点的最短距离。
网络直径(diameter)：拓扑网络上任意两个节点间的最大跳数。
二分宽度(bisection width)：将拓扑网络平分为二的最小切割数。

可以利用跳、网络直径、二分宽度来衡量网络的性能

$\alpha - \beta$**模型**

网络通信时间由延迟α，带宽$\frac{1}{\beta}$，和消息长度L 决定（忽略拓扑架构）：T_{comm} =α +βL

推论：多条短消息不如一条长消息

这种模型准确的较高

BSP模型

基本假设：

每个处理器拥有一个独立的内存空间；
所有处理器可以通过一个公共网络采用点对点方式通信；
所有处理器可以通过该网络实现同步；
程序以超步(superstep) 为单位并行执行；
每个超步末进行栅栏同步，从而保证所有处理器同时进行下一个超步。

BSP 模型参数

- p：处理器个数；
- S：总超步数；
- g：每单位消息的单边通信时间（$1/g$= 通信带宽，由硬件决定）；
- ℓ：每次栅栏同步的时间（由硬件决定）；
- w_s：第 s 超步本地计算的最大时间；
- h_s：第 s 超步单边通信的最大消息量。

采用 BSP 预测程序执行总时间：

$$\text{Time}_{BSP} = \sum_{s=1}^{S} w_s + g \sum_{s=1}^{S} h_s + \ell S.$$

并行编程理论

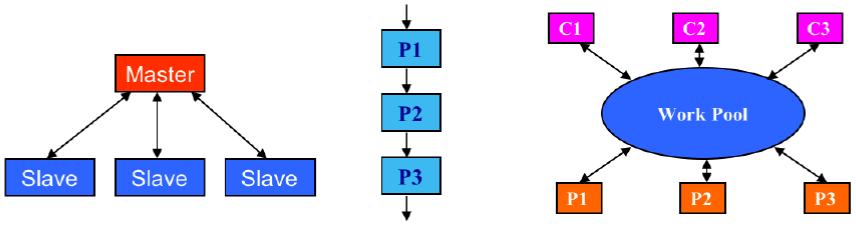
并行计算的原则和形式

八字原则：负载均衡，通信极小

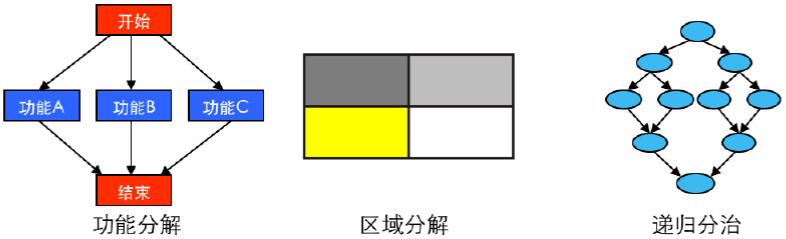
基本形式：主从并行、流水线并行、工作池并行、

功能分解、区域分解、递归分治

图解如下：



主从并行　　流水线并行　　工作池并行



功能分解　　区域分解　　递归分治

并行编程模型

自动并行

- 提交串行程序，由编译器自动实现并行；
- 希望十分渺茫。

共享内存并行编程

- 适用于共享内存并行机，以线程为并行单位；
- pthreads、OpenMP、CILK、TBB等。

消息传递

- 适用于共享内存和分布式并行机；
- PVM、MPI等。

数据并行

- CUDA/OpenCL、 Fortran 90、 PGAS、MapReduce等。

混合并行

- MPI + OpenMP、MPI + CUDA 等。

共享内存与消息传递

特性	共享内存	消息传递
内存访问	共享全局内存	各处理器拥有独立内存
通信方式	隐式，通过共享内存	显式，通过消息发送和接收
扩展性	受限于内存带宽，扩展性较差	良好的扩展性，适合大规模分布式系统
数据一致性	需要通过锁或同步机制解决	不需要一致性管理
通信开销	低	高，依赖网络性能
编程复杂度	较低，适合线程级并行	较高，需要显式管理数据传递和同步
适用场景	多核处理器、GPU 并行	分布式系统（如集群、超级计算机）