# 概念

并行计算（Parallel computing）本身是个很泛的概念，简而言之，就是指的同时有若干计算执行的计算模式，并行计算的实现可能需要硬件、操作系统或者应用程序。

以下是一些并行计算中的一些概念，这其中某些概念相互之间有重合，某些概念则强调的是同一事物的不同方面特点，某些概念也并不非常精确，并随着时间、领域、语境略有变化。

### 分布式计算（Distributed computing）

分布式计算主要研究分布式系统（Distributed system）如何进行计算，分布式系统是一组通过网络连接的计算机，这些计算机通过网络进行消息传递，以达到同一目标。分布式系统把需要大量计算的工程数据分割成小块，分配给各个运算组件（计算机），再上传计算结果之后整合，得出结论。

分布式计算有3个重要特种：

1. concurrency of components（并发的运算组件）
2. lack of global clock（没有统一的时钟）
3. independent failure of components（单个计算组件的错误不会引起崩溃）

分布式计算的架构：

1. C/S架构
2. Three-tier：三层式架构（用户界面、逻辑、数据存储及访问）
3. N-tier：多层式架构
4. P2P

### 大规模并行（Massively parallel）

大规模并行指的是使用大量的处理器进行的并行计算。有几种硬件方式实现：

1. 网格计算（Grid computing）
2. 计算机集群（computer cluster）

大规模并行也可能指**大规模并行阵列（MPPA，Massively parallel processor array）**

参考《硬件->大规模并行阵列》

### 云计算（Cloud computing）

云计算本身是个商业上的概念，在此只做简单介绍，不做深入讨论。

云计算是一种基于[互联网](https://zh.wikipedia.org/wiki/互联网)的计算方式，通过这种方式，共享的软硬件资源和信息可以按需求提供给用户（计算机各种终端和其他设备）。

云计算的优势在于**经济性**（用户可根据峰谷值实时调整租用的计算资源，云提供商则可实时的将这些计算资源分配给各个用户避免空耗）以及搭建和维护的**简便性**（用户无需了解云中的基础设施细节）

云计算的部署模式：公有云、私有云、混合云

云计算的服务模式：主要有3个，**IaaS、PaaS、SaaS**。此外还有**MBaaS**（Mobile Backend）、**FaaS**（Function）、**DaaS**（Data）及**虚拟桌面**（Desktop as a service）

##### IaaS

基础设施即服务（Infrastructure as a service），消费者使用“基础计算资源”，如处理能力、存储空间、网络组件或中间件。消费者能掌控操作系统、存储空间、已部署的应用程序及网络组件（如防火墙、负载平衡器等），但并不掌控云基础架构。

例如：亚马逊AWS的EC2、阿里云的ESC。

##### PaaS

平台即服务（Platform as a service），介于IaaS和SaaS之间，消费者使用主机操作应用程序。消费者掌控运作应用程序的环境（也拥有主机部分掌控权），但并不掌控操作系统、硬件或运作的网络基础架构。平台通常是应用程序基础架构（即软件部署平台）。

例如：Google App Engine。

##### SaaS

软件即服务（Software as a service），这种服务模式中[云端](https://zh.wikipedia.org/wiki/云计算)集中式[托管](https://zh.wikipedia.org/wiki/互联网托管服务)软件及其相关的[数据](https://zh.wikipedia.org/wiki/数据)，软件仅需透过互联网，而不须安装即可使用。用户通常使用[瘦客户端](https://zh.wikipedia.org/wiki/精簡客戶端)访问SaaS。用户使用应用程序，但并不掌控操作系统、硬件或运作的网络基础架构。

例如：Microsoft CRM与Salesforce.com。

### 高性能计算（High-performance computing）

即指超级计算机（Supercomputer）。目前最快的超级计算机的算力已经超过100PFLOPS（10的17次方Float-point Operations per second），目前世界前500强的超级计算机全部使用Linux操作系统（Linux、CentOS、Cray Linux Environment、SUSE等）。

参见《硬件->多核处理器->大规模核心处理器》

参见《概念->多处理器处理》

### 多处理器处理（Multiprocessing）

指在一个单一计算机系统中，使用二个或二个以上的CPU，以及能够将计算工作分配给这些处理器。拥有这个能力的计算机系统，也被称为是多处理器系统（Multiprocessing system）。

##### 对称性

**SMP**和**ASMP**是两种不同对称性的多处理器系统，多处理器系统的对称性是由硬件和操作系统共同决定的。

**对称多处理器（Symmetric multiprocessor）**指的是系统无差别对待每个CPU，任何任务都可能分配到任何CPU上。

**非对称多处理器（Asymmetric multiprocessor）**相对于SMP，指的是系统中的每个CPU并非被平等对待（比如有的系统中有一个专门的CPU负责处理中断，有的系统使用一个专门的CPU运行操作系统内核等）

##### 耦合性

**紧耦合（Tightly-coupled）**指的是多个处理器以总线紧密连接，这些处理器可能使用共享内存。

**松耦合（Loosely-coupled）**的多个处理器之间通常是透过高速的网络设备进行消息传递，也被称作计算机集群（Computer cluster）

### 通用GPU（GPGPU）

通用图形处理器（General-purpose Graphics Processing Unit），是利用GPU来计算原本由[CPU](https://zh.wikipedia.org/wiki/中央处理器)处理的通用计算任务。这些通用计算任务通常与图形处理没有任何关系。由于现代GPU有强大的并行处理能力和可编程流水线，令GPU也可以处理非图形数据。特别是在面对[单指令流多数据流](https://zh.wikipedia.org/wiki/单指令流多数据流)（SIMD）且数据处理的运算量远大于数据调度和传输的需要时，GPGPU在性能上大大超越了传统的CPU。

# 费林分类法（Flynn’s taxonomy）

一种计算机结构的分类方式。

|  |  |  |
| --- | --- | --- |
|  | **单指令流** | **多指令流** |
| **单数据流** | [单指令流单数据流](https://zh.wikipedia.org/wiki/單指令流單數據流) （SISD） | [多指令流单数据流](https://zh.wikipedia.org/wiki/多指令流單數據流) （MISD） |
| **多数据流** | [单指令流多数据流](https://zh.wikipedia.org/wiki/单指令流多数据流) （SIMD） | [多指令流多数据流](https://zh.wikipedia.org/wiki/多指令流多数据流) （MIMD） |

### SISD

单指令流单数据流（Single Instruction stream Single Data stream），是一种单核单线程单指令流的体系结构。SISD也可以通过指令流水线获得并发计算的能力。

例如：标量处理器

### SIMD

单指令流多数据流（Single Instruction stream multiple data stream），是一种多个计算组件同时对不同的数据进行同样的操作的体系结构。这样的结构实现了数据级并行，但却不是并发的。

例如：向量超级计算机，GPU

##### SIMT

单指令流多线程（Single instruction, multiple threads），是SIMD和multithreading的组合。SIMT架构在一些GPU上已经实现。

### MISD

多指令流单数据流（Multiple instruction stream single data stream），是一种很少见的结构。MISD具有n个处理单元，按n条不同指令的要求对同一数据流及其中间结果进行不同的处理。一个处理单元的输出又作为另一个处理单元的输入。

严格意义上，采用这种结构的计算机至今没有出现，但是已经有了近似的例子。在指令[流水线](https://zh.wikipedia.org/wiki/流水线_(计算机))结构中，一条[指令](https://zh.wikipedia.org/wiki/指令)的执行过程被分为多个步骤，并且交给不同的硬件处理单元，以加快指令的执行速度。

### MIMD

多指令流多数据流（Multiple Instruction stream multiple data stream），多个处理器异步且独立的工作，任何时候，不同的处理器都可以使用不同的指令对不同的数据进行操作。

MIMD的实现很多，多处理器，多核心处理器，SoC，超线程，超标量处理器等等，这些都算是MIMD。

# 硬件（Hardware）

### 数据流架构（Dataflow architecture）

与**冯诺依曼架构（控制流架构）**相反，数据流架构并没有程序计数器（Program counter），程序的执行仅仅跟输入的数据有关。

迄今为止并没有商用的通用处理器采用此种架构（都用控制流架构），不过在专用硬件中已经得到实现，比如DSP、图形处理等。

### 指令流水线（pipelined processor）

参考《并行计算的层级->指令级并行->指令流水线》

### 标量处理器（Scalar processor）

标量处理器是最简单的一类[计算机](https://zh.wikipedia.org/wiki/计算机)[处理器](https://zh.wikipedia.org/wiki/处理器)。此类处理器在同一时间内只处理一条数据，如整数或者[浮点数](https://zh.wikipedia.org/wiki/浮点数)。标量处理器属于SISD的一种。

### 超标量处理器（Superscalar processor）

参考《并行计算的层级->指令级并行->超标量》

### 并行向量处理机（Vector processor）

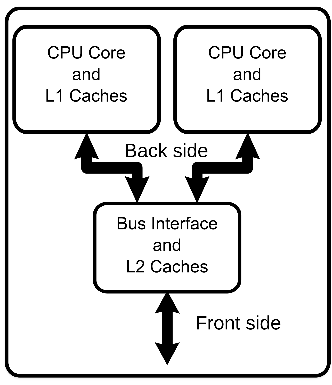
是一种实现了直接操作一维[数组](https://zh.wikipedia.org/wiki/數組)（向量）指令集的CPU。这与一次只能处理一个数据的[标量处理器](https://zh.wikipedia.org/wiki/标量处理器)正相反。现在，绝大多数商业化的CPU实现都能够提供某种形式的向量处理的指令，用来处理多个（向量化的）数据集，也就是所谓的SIMD

### 多核处理器（Multi-core processor）

是在单个计算组件（处理器）中，加入两个或以上的独立实体[中央处理单元](https://zh.wikipedia.org/wiki/中央處理單元)（核心，Core），只有两个核心的处理器，称为双核处理器（dual-core processor）。这些核心可以分别独立地运行程序指令，利用[并行计算](https://zh.wikipedia.org/wiki/平行計算)的能力加快程序的运行速度。“多核心”通常是对于中央处理器（CPU）而论的，但是某些时候它也指代数字信号处理器（[DSP](https://zh.wikipedia.org/wiki/DSP)）和系统芯片（[SoC](https://zh.wikipedia.org/wiki/系統單晶片)）。

现如今由于发展迅速，多核可能指代不同的架构，不过本质上都是同一个CPU中的部分组件有若干重复。比如，有的多核处理器中执行单元和L1 cache是独享的，而L2 cache是共享。

下图为一个双核处理器（独享的L1 cache和共享的L2 cache）



##### 大规模核心处理器（Manycore processor）

专门设计的**多核处理器**，核心数量从几十个到上千个，通常用于**超级计算机**。

目前世界排名第一的Summit是由9216个22核处理器（POWER R9）和27648个5120核GPU（Nvidia Telsa V100）组成，排名世界第三的神威-太湖之光是由40960个260核处理器（SW26010）组成。

虽然目前世界排名1-3的超级计算机都不是Intel/AMD架构，但是在超级计算机500强中的大部分计算机都是x86-64架构的CPU（比如Xeon）.

### 多处理器（Multiprocessor）

参见《概念->多处理器处理》

### 超线程（HT，Hyper-threading）

超线程是intel研发的一种**SMT**技术。

参考《多线程->交叉/同时》

### 内存

##### 共享内存

硬件中的共享内存指的是**多处理器**系统可以被若干CPU访问的内存，共享内存可能有以下几种类型：

UMA

**均匀访存（UMA，Uniform memory access）**，所有CPU对内存的访问是一致的，或者说访问时间和具体CPU无关。

NUMA

**非均匀访存（NUMA，Non-uniform memory access）**，对内存的访问时间根据具体内存位置和具体CPU不同而不同。CPU对自己的本地内存访问速度快于对其他CPU的内存（或共享的内存）。

COMA

**Cache-only memory architecture**，指的是CPU仅将本地内存当做cache使用，而不是像**NUMA**一样视为共享内存。

##### 分布式内存

分布式内存（Distributed memory）指的是多处理器系统中，每个CPU有自己的内存，所有的计算任务只能在本地内存中进行。如果需要远程数据，CPU必须要跟其他CPU通信。

##### 分布式共享内存

分布式共享内存（DSM，Distributed shared memory）指的是多处理器系统中的内存虽然在物理上分开的，但是在逻辑上被组织在同一个共享的地址空间中。

### 大规模并行阵列（MPPA）

**Massive parallel processor array**，或者也叫**multi purpose processor array**，一种包含大量（几百个或几千个）并行CPU和内存的芯片，CPU之间通过可配置的通道进行连接。

MPPA用于高性能嵌入式系统，以及桌面电脑和服务器的硬件加速，比如视频压缩、图像处理等等。

### 计算机集群（computer cluster）

**计算机集群**是一种计算机系统，它通过一组[计算机](https://zh.wikipedia.org/wiki/计算机)[软件](https://zh.wikipedia.org/wiki/软件)/[硬件](https://zh.wikipedia.org/wiki/硬件)连接起来高度紧密地协作完成计算工作。在某种意义上，他们可以被看作是一台计算机。集群系统中的单个计算机通常称为节点，通常通过局域网连接，但也有其它的可能连接方式。

### 网格计算（Grid computer）

通过利用网络上分布的大量异构计算机的计算资源进行的并行计算。

**网格计算**和**计算机集群**的区别在于：

1.网格计算中的计算机是异构的

2.物理距离远大于计算机集群，计算机集群一般局限于数据中心

3.网格计算中每个节点的任务是不同的，而计算机集群中每个节点的任务是相同的

4.网格计算中的每个节点是不可信任的。

网格计算是**分布式计算**的一种，其主要特点在于各节点的工作载荷交互性很低（即任务相对独立），以及每个节点是不可信任的。

# 软件（Software）

### 进程（process）

进程是资源调度的最小单位，或者说每个进程拥有自己的独立地址空间。在线程出现之前，进程也是执行的基本单位，在线程出现之后，进程仅作为线程的容器。

### 线程（thread）

线程是OS能够执行调度的最小单位，包含在进程之中，同一个进程之中的线程共享地址空间。

线程分为**用户线程（User thread）**和**内核线程（Kernel thread）**，内核线程也叫轻量级进程（因为所有进程中的内核地址空间是一致的，因此内核线程就相当于进程）。线程通常指用户线程，由用户进程自行调度。

### 纤程/协程（fiber/coroutine）

纤程（Fiber）是最轻量化的调度单位，是一种用户态的线程。纤程比纤程更加轻量化，应用程序可以独立决定自己的纤程该如何调度，操作系统看不见它，也无法进行调度。

协程（Coroutine）即纤程，它们区别在于，协程是语言层面的概念，而纤程是系统层面的概念。

# 层级

### 位级并行（BLP，Bit-level parallelism）

位级并行即通过增加处理器字长来实现并行计算。比如在一个8位机上计算两个16位数相加，需要分别对高8位和低8位进行加法操作，之后再将两者相加；而在16位机上一次计算即可完成。

##### SWAR（SIMD Within A Register）

SWAR是一种在处理器寄存器中进行并行计算的技术，它将不相干但需要相同计算的subword组合起来放入寄存器，用同一次操作完成计算。通过一组寄存器及其对应的指令，大部分现代的通用处理器都支持一定程度的SWAR 。

Intel的**MMX**技术是SWAR早期的一种实现。

### 指令级并行（ILP，Instruction-level parallelism）

ILP指在同一个核心中，若干指令可以同时运行。也是一种衡量同时有多少指令可以同时运行的标准。

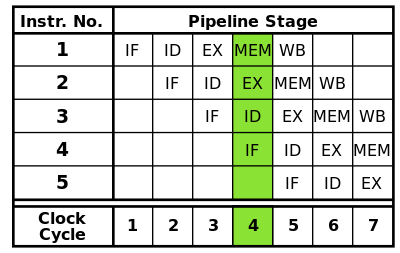
有两种方式可以实现ILP，硬件和软件

1. 硬件方式为动态并行，由处理器决定哪些指令可以同时运行。
2. 软件方式为静态并行，由编译器决定哪些指令可以同时运行。

实现ILP的主要技术有两种：**指令流水线**、**超标量**。除此之外还有一些提高效率的辅助技术，比如：**乱序执行**、**寄存器重命名**、**预测执行**、**分支预测**。

##### 指令流水线（Instruction piplining）

指令流水线技术是一种使得多条指令可以部分重叠执行的技术：一条指令的执行被分成了多个阶段，每个阶段需要处理器的不同部分完成，如果没有指令流水，则除了当前执行的部分之外，其他部分是被闲置的。指令流水技术通过使用处理器不同部分执行不同的指令，使得若干指令可以同时执行，从而减少了CPU的闲置 ，提高并行计算能力。



上图是一个典型的RISC机器的五级流水线：

IF：Instruction Fetch，指令获取

ID：Instruction Decode and register fetch，指令解码和寄存器读取

EX：Execution，指令执行

MEM：Memory access，内存访问

WR：Register wirteback，寄存器写回

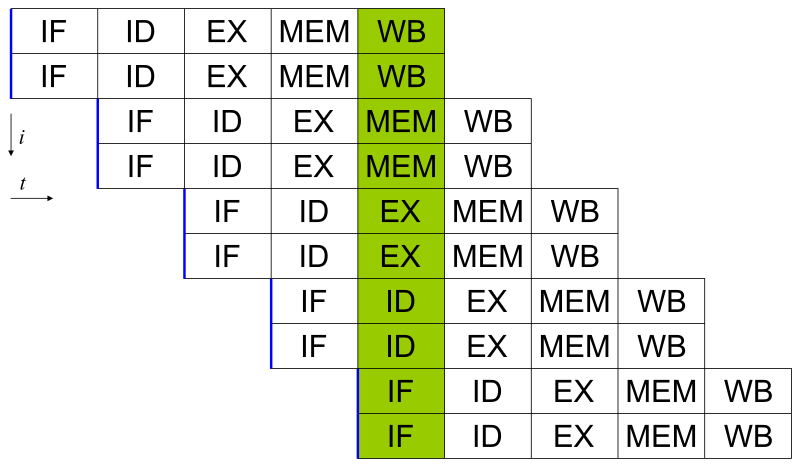
很多处理器的流水线能达到5级、7级甚至31级。

##### 超标量（Superscalar）

超标量处理器实现了指令级并行。处理器的内核中一般有多个功能单元，比如算数逻辑单元（ALU）、位移单元、乘法器，超标量的CPU在一个时钟周期可以分派多条指令在不同的功能单元中执行。

相较于标量处理器（scalar processor），超标量处理器在每个时钟周期可以执行多于一条指令（仅实现了指令流水的标量处理器在每个时钟周期也仅能完成一条指令的执行）。

超标量和指令流水是实现ILP的不同技术，下图是一个超标量指令流水线：



##### 乱序执行（Out-of-order execution）

乱序执行指的是处理器在一个由输入数据可用性所决定的顺序中执行指令，而不是由程序的原始数据所决定。在这种方式下，可以避免因为获取下一条程序指令所引起的处理器等待，取而代之的处理下一条可以立即执行的指令。比如：

1. c = b+a

2. e = c + d

3. h = f + g

以上指令序列中，1和3可并发执行，而2则必须等1的结果才可执行。

##### 寄存器重命名（Register renaming）

寄存器重命名是一种技术，通过去除**寄存器重用**而造成的**伪数据依赖**（False data dependency），从而增强指令级并行。

某些在源代码级别前后互不依赖的语句，在经过编译器编译之后，却可能因为使用到相同的通用寄存器，形成伪数据依赖，从而使得指令流水/超标量无法实现。通过将同一个通用寄存器映射为不同的物理寄存器，可以去除这种伪数据依赖。

##### 预测执行（Speculative execution）

预测执行指的是根据现有信息，利用CPU空转时间提前执行一些将来可能用得上，也可能用不上的指令（比如if-else后的语句）。如果指令执行完成后发现用不上，系统会抛弃计算结果，并回退执行期间造成的副作用（如缓存）。

##### 分支预测（Branch prediction）

条件分支指令通常具有两路后续执行分支。即不采取跳转，顺序执行后面紧挨JMP的指令；以及采取跳转到另一块程序内存去执行那里的指令。是否条件跳转，只有在该分支指令在指令流水线中通过了执行阶段才能确定下来。

分支预测猜测两路分支中哪一路最可能发生，然后预测执行（speculative execution）这一路的指令，来避免流水线停顿造成的时间浪费。如果后来发现分支预测错误，那么流水线中推测执行的那些中间结果全部放弃，重新获取正确的分支路线上的指令开始执行，这招致了程序执行的延迟。

静态预测

静态预测不依赖于代码执行的动态历史信息，仅依赖于分支指令自身。最简单的单方向静态预测总是预测条件跳转发生（或者不发生）。一些处理器允许分支预测提示出现在代码中（比如C/C++中的likely和unlikely宏），比如Intel奔腾4，但之后的Intel处理器不支持。

动态预测

一条条件跳转指令第一次遇到，还没有任何信息可以去预测分支。此后保持这条指令是采取还是不采取跳转的历史记录，就可以作为再遇到这条指令时猜测最可能的分支。这就是动态预测。

### 任务并行（TP ，Task parallelism）

也叫功能并行（Function parallelism）或控制并行（Control parallelism）。

任务并行关注的是多处理器（或者多核心处理器）上任务（进程或线程）的分配。任务并行是由操作系统内核中的调度器（Scheduler）实现的。

##### 线程级并行（Thread-level parallelism）

线程级并行是任务并行的一种，实现了一个应用程序内多线程的并行执行。

### 数据并行（DP ，Data parallelism）

是一种[多处理器](https://zh.wikipedia.org/wiki/多處理器)的[平行运算](https://zh.wikipedia.org/wiki/平行運算)模式，将资料分配到不同的平行运算节点中。资料平行把大的任务化解成若干个相同的子任务，处理起来比[任务平行](https://zh.wikipedia.org/wiki/任務平行)简单。

### 内存级并行（MLP ，Memory-level parallelism）

[并行计算](https://zh.wikipedia.org/wiki/平行計算)技术的一种，是电脑架构的一种，能够同时进行数个内存操作，特别是在[快取](https://zh.wikipedia.org/wiki/CPU快取)未命中（cache miss），或[转译后备缓冲区](https://zh.wikipedia.org/wiki/轉譯後備緩衝區)未命中（TLB miss）时。

在单核心处理器架构下，内存层级平行可以被视为是一种特殊的[指令层级平行](https://zh.wikipedia.org/wiki/指令層級平行)（ILP）。它也经常在[超标量](https://zh.wikipedia.org/wiki/超純量)架构下出现。

### Loop-level parallelism

并行计算技术的一种，对循环的并行，当一段程序被循环执行的时候，多个线程同时对若干或者所有的索引执行循环体。

# 多线程（Multithreading）

Multithreading指的是一个CPU（或者多核CPU的一个核）并发的执行多个线程的能力（需要OS支持）。

### 交叉/同时（Temporal/Simultaneous SMT）

交叉多线程（Temporal multithreading）和同时多线程（SMT，Simultaneously multithreading）是多线程的两种主要形式。在交叉多线程中，同一时间只有一个线程的指令可以在**某个CPU核**的任何指定的流水线阶段执行。

而在同时多线程中，多于一个线程的指令可以在同一个CPU核的任何指定的流水线阶段执行。SMT的一个实现是intel的**超线程技术（HT，Hyper-threading）。**

完成SMT在基本的处理器架构上没有太多改变：所需的主要添加是在一个周期中从多个线程获取指令的能力，以及一个更大的寄存器文件来保存来自多个线程的数据。并发线程的数量可以由芯片设计者决定。常见模式是每个CPU核心有两个并发线程，但一些处理器的每个核心支持最多八个并发线程。

### SpMT（Speculative Multithreading）

又叫TLS（Thread Level Speculation），一种对接下来可能会执行的线程进行提前预测，并预测执行其代码的技术。

### 抢占式/协作式（Cooperative/Preemptive）

**协作式（Cooperative）**，或**非抢占式（None-preemptive）**多任务环境中，下一个进程被调度的前提是当前进程主动放弃时间片。

而在**抢占式（Preemptive）**环境中，操作系统完全决定调度方案，操作系统可以剥夺耗时长的任务的时间片，提供给其他进程。

### 硬件侦测（Hardware scout）

指的是一种无需应用程序或者操作系统干预，由硬件（处理器）来分发线程执行数据预取任务，从而加速执行的技术。

# 概念区分

### 预测执行和分支预测

分支预测和预测执行是两个相连的阶段，先（分支）预测，再（预测）执行

### 并行、并发

并行计算（Parallel computing）和并发计算（Concurrent computing）在很大一部分上是概念重合的，但也有区别：

1.相对于并行，并发强调的是多个**指令**同时进行，因此SIMD算并行**不算并发**，位级并行也**不算并发**。

2.相对于并发，并行则强调的是**同时**进行，因此时分多任务操作系统**算并发不算并行**，多个客户端访问服务器**算并发不算并行**。

### 标量、向量、超标量

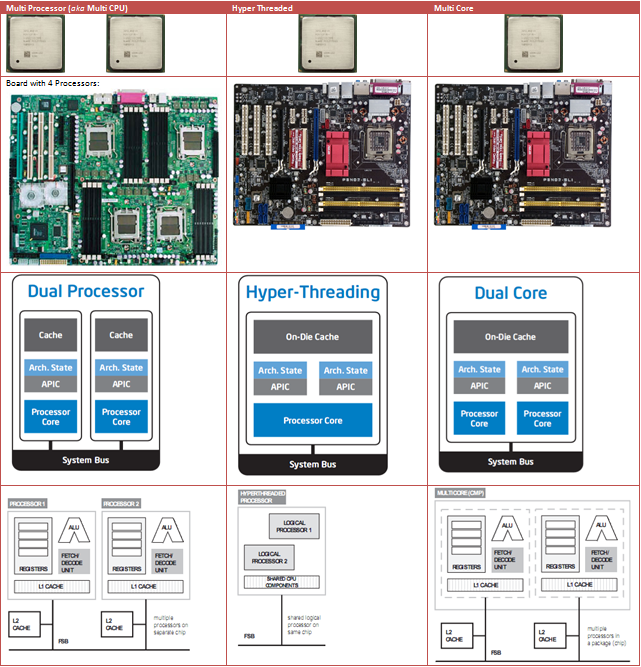
标量、向量、超标量分别对应于费林分类法中的SISD、SIMD和MIMD

### Multi-core、Manycore

**Manycore processor**是**multi-core processor**的一个特化，为高性能计算而设计。

### 多核、多处理器、超线程

**多处理器**指的是多个CPU的系统，从外观上看即为独立的芯片；**多核处理器**指的是CPU中某些计算组件（执行单元）有重复，形成若干核心；通常每个CPU核心中只能有一个线程执行，**超线程**指的是在同一个核心中可以有若干线程被执行。请参考下图：



### 进程、线程

进程（Process）和线程（Thread）都是操作系统范畴的概念，进程和线程都表示一个独立的执行绪（sequence of execution），进程中可以包含若干线程。其最主要的区别在于，进程**不共享地址空间**（Address space），而同一个进程中的线程**共享地址空间**。

# 概念合并

SISD-标量处理器

SIMD-GPU-数据级并行-并行向量处理器

MISD-指令流水线

MIMD-超标量处理器-任务并行-多核-多处理器

SMT-超线程

分布式计算-网格计算

高性能计算：manycore，multi-core，多处理器，计算机集群

# 一台土鳖计算机的并行化

一个8位的单核心单线程CPU电脑，同时只能执行一条指令，运行单任务操作系统，阻塞在任何IO操作上，不可做任务切换，可以并行/并发的地方有：

1. 8位 → 64位（位级并行）
2. 程序中不使用协程 → 程序中使用协程（纤程）
3. 执行一条指令 → 同时执行多条指令（指令流水线）
4. 每个时钟周期执行一条指令 → 每个时钟周期同时执行多条指令（超标量）
5. 单线程核心 → 超线程核心（SMT）
6. 单核心CPU → 多核CPU（多核处理器、Manycore processor）
7. 单CPU → 多CPU系统（多处理器）
8. 仅CPU运行 → CPU+GPU运行（GPGPU）
9. 单任务操作系统 → 时分多任务操作系统（并发计算）
10. 一台电脑 → 多台电脑（分布式计算、计算机集群、网格计算）