并行性：计算机系统在同一时刻或者同一时间间隔内进行多种运算或操作。

同时性：两个或两个以上的事件在同一时刻发生。

并发性：两个或两个以上的事件在同一时间间隔内发生。

提高并行性的三个基本思想

1.时间重叠

引入时间因素，让多个处理过程在时间上相互错开，轮流重叠地使用同一套硬件设备的各个部分，以加快硬件周转而赢得速度。

2.资源重复

引入空间因素，以数量取胜。通过重复设置硬件资源，大幅度地提高计算机系统的性能。

3.资源共享

这是一种软件方法，它使多个任务按一定时间顺序轮流使用同一套硬件设备。如云计算中的虚拟机。

分布计算系统

分布计算系统是由多个相互连接的处理资源组成的计算系统，它们在整个系统的控制下可合作执行一个共同的任务，最少依赖于集中的程序、数据和硬件。这些处理资源可以是物理上相邻的，也可以是在地理上分散的。

分布计算系统和计算机网络系统有什么区别呢？

如果用户能说明他在使用哪一个计算机，则他是在使用一个计算机网络系统而不是分布式系统。一个真正的分布计算系统的用户不必知道他的程序在哪个机器上运行，他的文件在哪里存放，等等。使分布计算系统具有这种性质的是它的软件：分布式操作系统。

经常性原理

以经常性事件为重点

对经常发生的情况采用优化方法的原则进行选择，以得到更多的总体上的改进。

优化是指分配更多的资源、达到更高的性能或者分配更多的电能等。

程序的时间局部性

程序即将用到的信息很可能就是目前正在使用的信息（或刚刚用过不久的信息）。

程序的空间局部性

程序即将用到的信息很可能与目前正在使用的信息在空间上相邻或者临近。

Moore 定律

单位面积的晶体管数量每 18 个月增加一倍，价格则相反

Amdahl 定律

加快某部件执行速度所能获得的系统性能加速比，受限于该部件的执行时间占系统中总执行时间的百分比。如果仅仅对计算任务中的一部分做性能改进，则改进得越多，所得到的总体性能的提升就越有限。

可改进比例：在改进前的系统中，可改进部分的执行时间在总的执行时间中所占的比例。

部件加速比：可改进部分改进以后性能提高的倍数。它是改进前所需的执行时间与改进

后执行时间的比。

多处理机

当一个处理器不足以满足计算需求时，除了增强单个核心的计算性能（但这很难），最直观的方法就是增加核心数量（线程级并行，TLP）

我们称这种拥有多个处理机的结构为多处理机，其特点是多个处理机共用一个共有的内存，也称为共享内存模型。

UMA: 一致访存，所有处理器对内存的访问是一致的，可以有私有 cache。典型的 UMA：集中式共享存储器 SMP

NUMA: 非一致访存，处理器有各自的存储器。典型的 NUMA：分布式共享存储器 DSM

以上两种模型均具有统一的地址空间，也就是基于共享内存模型

并发

在操作系统中，线程与进程可以在单一处理器上(通过操作系统内核)实现，基本概念是对多道程序的快速切换(分配时间片)，是一种软件方式的伪并行，也被称为并发。

共享变量	消息传递
• 适合于SMP、DSM	• 适合于MPP、COW（不止）
• 单一地址空间	• 多地址空间
• 隐式通信	• 显式通信
• 在集群中，一般用于一个节点的多个核上	• 一般用于集群中的多个节点上

竞争

当程序的正确运行依赖于程序中各线程的特定时序时，就会出现竞态。这种依赖往往发生在多个线程对同一个资源的竞争中，尤其当其中存在写操作时。在内存上，这被称为数据竞争。

数据竞争的解决方法：利用同步，确保操作是原子性的，从而对操作进行排序，将资源与操作“保护”起来。

OpenMP

OpenMP 是基于线程的并行编程模型，一个共享的进程由多个线程组成。使用 FORK-JOIN 并行模型，主线程串行执行，直到编译制导并行域出现。

1.并行域结构

一个并行域就是一个能被多个线程并行执行的程序块，它是最基本的 OpenMP 并行结构。

2.任务划分结构

用来表明任务如何在多个线程间分配，任务划分结构将它所包含的代码划分给线程组的各成员来执行。它不产生新的线程，在任务划分结构的入口点没有路障，但在其结束处有一个隐含的路障。

3.同步结构

同步结构用于控制执行过程中各线程的同步。

消息传递

在消息传递中，同步和异步是发送方对通信的两种不同处理方式

同步：发送方等待接收方完成接收并响应，然后开始处理其他工作

异步：发送方在发送消息后不等待响应，直接开始处理其他工作

MPI	OpenMP
多进程并行	多线程并行
消息传递	共享内存
数据分配方式显式	数据分配方式隐式
单机/多机	单机多核

MPI 消息

消息的内容，即信的内容，在 MPI 中称为消息缓冲，消息缓冲由三元组<起始地址，数据个数，数据类型>标识

消息的接收发送者，即信的地址，在 MPI 中成为消息封装，消息信封由三元组<源/目标进程，消息标签，通信域>标识

MPI 的四种点对点通信模式

通信模式指的是缓冲管理，以及发送方和接收方之间的同步方式。

共有下面四种通信模式：

标准通信模式 MPI_Send，它如何发送数据由具体实现决定

就绪发送 MPI_Rsend：只有当接收方已经开始接收，发送方才可以发送，否则发送出错

同步发送 MPI_Ssend：无论接收方是否启动都可以开始发送，但只有当接收方开始接收，函数才会返回

缓冲发送 MPI_Bsend：由用户定义、使用缓冲区，无论接收方操作是否启动，只要缓冲区可用都可以开启发送

MPI 的两类通信机制

阻塞通信与非阻塞通信

数据级并行——SIMD

更少的 Fetch 和 Decode 意味着什么？

——更少的器件，更低的能耗和时间开销

更多的 ALU 意味着什么？

——一次流水能处理更多数据，速度更快
增加数据寄存器的数量来一次存储更多数据，以减少存储器访问延迟

向量的计算方式

横向计算：数据相关：N 次　功能切换：2N 次
纵向计算：数据相关：1 次　功能切换：1 次
纵横计算：数据相关：1 次　功能切换：2 次

向量体系结构

向量体系结构应当具有很大的顺序寄存器堆 (Register File)，可加载更多向量元素以支持纵向计算。

向量体系结构从内存中收集散落的数据，将其放入寄存器堆中，并对寄存器堆中的数据们进行操作，然后将这些结果放回内存（一次传输一组数据，LD/ST 流水化）。

一条指令能够对一个向量的数据进行操作，也就对向量中诸多独立数据元素进行了操作（纵向计算，功能单元流水化）。

向量体系结构的性能优化

多车道技术、链接技术、编队技术、分段开采技术

三驾马车

在分布式领域，谷歌公司提出了三项技术来分别解决文件存储、结构化数据存储和分布式计算模型这三个关键问题，分别是：GFS（Google File System）、BigTable、MapReduce。

Hadoop：

Hadoop 包含了谷歌“三驾马车”的开源实现：
GFS　--　HDFS：分布式文件系统
MapReduce　--　Hadoop MapReduce：编程模型框架
BigTable　--　HBase

MapReduce：编程模型框架

一个 MR 作业通常将数据分为多个部分，每个部分分别由 map 操作生成中间值，然后由 reduce 操作对具有相同 key 的所有 value 进行汇总

用户编写的 Map 函数接受输入对，产出一系列中间 k-v 对。MR 库将同一个中间 key 对应的所有中间 value 收集起来，并发送给 Reduce。用户编写的 Reduce 函数接受中间 key，以及该 key 对应的所有 value 的集合，并将其整合为一个更小的集合。

云计算的服务模式

IaaS（基础设施即服务）
PaaS（平台即服务）
SaaS（软件即服务）

虚拟化能做到什么

分区

可在一台物理机上运行多个操作系统，

可在虚拟机之间分配系统资源

隔离

可在硬件级别进行故障和安全隔离

可利用高级资源控制功能保持性能

封装

可将虚拟机的完整状态保存到文件中。

移动和复制虚拟机就像移动和复制文件一样轻松。

独立于硬件

可将任意虚拟机置备或迁移到任意物理服务器上。

虚拟化技术的分类

从资源类型上，虚拟化技术可以广义地分为

计算虚拟化（狭义的虚拟化）

存储虚拟化

网络虚拟化

狭义上的虚拟化，根据虚拟化程度还可分为

虚拟机（硬件虚拟化）

容器（OS 级虚拟化）

虚拟机

虚拟机就是通过在宿主机上“模拟”物理硬件来作为客户机，从而提供一个虚拟化的软件（操作系统）运行环境

Hypervisor 是虚拟机的引擎，负责硬件资源的分配等重要工作。根据 Hypervisor 可以将虚拟机划分为两类：

Type 1 裸金属（bare metal）：Hypervisor 直接运行在物理硬件上，典型的如 ESXi

Type 2 托管（hosted）：Hypervisor 完全运行在宿主操作系统之上

还可以将虚拟机技术分为两种：

全虚拟化：由 Hypervisor 将 guest 的特权指令捕获并进行处理

半虚拟化：修改 guest 操作系统，使其不使用特权指令

虚拟机编排 OpenStack：使用 OpenStack，可以在大型集群上对虚拟机进行管理

容器

容器是一种操作系统级虚拟化技术，它允许在同一个内核上分隔出多个用户空间实例。从应用程序的角度来看，每一个实例就是一台机器。

Docker 支持将容器打包为镜像（image），从而使得应用程序及其运行环境可以在任何地方运行。

容器编排 Kubernetes：K8s 是用来支持容器在集群上部署的软件

网络虚拟化

VLAN“虚拟局域网”，是在二层网络（数据链路层）划分和隔离广播域的技术。Overlay 技术将底层承载网络的复杂结构屏蔽，为上层提供一个清晰简洁的网络模型。

存储虚拟化

存储区域网络(SAN) 是一种专用的高速网络（也可以建构在传统以太网上），可提供对存储的块级网络访问。SAN 将存储设备呈现给主机，以使存储看起来是本地连接的。

在云上，或者说通过网络提供的存储服务一般有以下三种：文件存储、块存储、对象存储。

SOA 面向服务架构

将服务的思想引入到软件架构中，在 SOA 中，存在着许多独立的功能单元，被称为服务。服务之间相互依赖，通过通信的方式互相调用。

微服务通信方式

	gRPC	HTTP with JSON
序列化方式	ProtoBuf	JSON
编码格式	二进制	文本
底层协议	HTTP/2	HTTP
约定	需要	不需要
压缩	是	否

Sidecar

Sidecar 是一种设计模式，将我们刚刚所提到的配件们放到 Sidecar 中，进而将业务代码和其他辅助功能解耦合

状态

无状态，简单的说就是程序（段）没有记忆，每一次它的执行只和它当前的输入有关，与之前发生的事无关。

有状态就是指程序（段）存在记忆，每一次的执行可能会对未来的执行造成“副作用”（产生影响）。

为了解决无法扩展这一问题，我们的一个方案是使用 Redis。将有状态的 session 存储到 redis 中，剩下的服务就是无状态的，于是服务就拥有了可扩展性。

多处理器存储结构分类

非共享存储多处理器

每个处理器内存私有，逻辑上独立编址不共享，无 Cache 一致性问题

属于松散耦合系统，支持消息传递编程模型，如 OpenMPI。

多个处理器存在于多个计算机中，实质是多进程 MIMD 问题

COW 工作站机群/MPP 大规模并行处理器

共享存储多处理器

所有处理器内存共享，逻辑上统一编址共享，有 Cache 一致性问题

属于紧密耦合系统，支持共享存储编程模型。如 OpenMP

多个处理器存在于同一个计算机中，实质是多线程 MIMD 问题

UMA/NUMA

典型多核结构

专用 L1-Cache 结构、专用 L2-Cache 结构、共享 L2-Cache 结构、共享 L3-Cache 结构。

缓存一致性问题

共享数据进入 Cache，则同一存储块在多个处理器的 Cache 中有副本，当某个处理器对其 Cache 中的副本修改后，该副本与其他副本中的数据不一致，称为多处理机的 Cache 一致性问题。

Cache 一致性协议的分类

写作废协议（Write invalidate）vs 写更新协议

发生 WtHit 时，把所有其他副本全部作废，还是将新数据同步更新到所有副本

写直达协议 vs 替换写回协议

发生 RdMiss/WtMiss 时，去存储器获得最新数据，还是从远程 cache 获得最新数据

按写分配协议（经过 cache）vs 不按写分配协议（绕过 cache）

发生 wt 时，先调入本地 cache 再 wt，还是直接到数据所在处 wt

Cache 一致性协议的两种实现方式

两种方式采用不同技术来记录和传递共享数据块的状态

监听式协议

基于总线传递信息:1vN

存储器中数据块的共享状态信息分散保存在各个 Cache 中，物理上分布的各个存储器拼合成逻辑上统一的大存储器，物理上分布的 cache 状态信息拼合成逻辑上统一的 Cache Hot 状态。

目录式协议

基于互联网络传递信息：1v1

存储器中数据块的共享状态信息集中保存在本地目录中，物理上分布的各个存储器拼合成逻辑上统一的大存储器，物理上分布的目录拼合成逻辑上统一的大目录。

监听式协议

两种情况下 cache 块写回存储器：发生 M 块替换，以及 M 变 S 之前
两种情况下 Invalidate：收到 Invalidate，以及收到 WtMiss

本地Cache控制器监听本地CPU的数据请求地址，
根据该地址查询本地数据cache：

查询结果	Cache块当前状态	Cache控制器操作	Cache块状态变更
RdHit	I（不可能）		
RdHit	S	read data	S
RdHit	M	read data	M
RdMiss	I	send RdMiss to Bus, 调入新块, read data	S
RdMiss	S	send RdMiss to Bus, 调入新块替换旧S块, read data	S
RdMiss	M	向存储器写回旧M块, send RdMiss to Bus, 调入新块替换旧M块, read data	S
WtHit	I（不可能）		
WtHit	S	send Invalidate to Bus, write data	M
WtHit	M	write data	M
WtMiss	I	send WtMiss to Bus, 调入新块, write data	M
WtMiss	S	send WtMiss to Bus, 调入新块替换旧S块, write data	M
WtMiss	M	向存储器写回旧M块, send WtMiss to Bus, 调入新块替换旧M块, write data	M

本地Cache控制器通过总线监听其他远程Cache，
根据总线上的请求查询自己本地的数据cache：

查询结果	Cache块当前状态	Cache控制器操作	Cache块状态变更
RdHit（见不到）			
WtHit（见不到）			
RdMiss	I	Null	I
WtMiss	I	Null	I
Invalidate	I	Null	I
Invalidate	M（不可能）	Null	I
RdMiss	S	竞争向远程Cache传输该块	S
RdMiss	M	向目标存储器和远程Cache传输该块	S
WtMiss	S	竞争向远程Cache传输该块	I
WtMiss	M	向远程Cache传输该块	I
Invalidate	S	Null	I

目录式协议

节点间消息

源	目标	消息类型	备注 说明: 数据块K指的是区间内包含地址K的数据块。
A	B	RdMiss(K)	请求数据块K，并申请A加入共享集G
A	B	WtMiss(K)	请求数据块K，并申请A独占共享集G
B	A	Data(K)	返回所请求的数据（K是请求地址）
B	C	Invalidate(K)	请求C将数据块K作废
B	C	FetchS(K)	请求数据块K的唯一副本，并请求将C的副本置为共享
B	C	FetchI(K)	请求数据块K的唯一副本，并请求将C的副本置为无效
C	B	Quit(K)	C中的clean副本将被替换，申请退出共享集G
C	B	Update(K)	C中的dirty副本将被替换，或响应Fetch请求，用C的唯一副本更新存储器
C	B	Invalidate(K)	C中的clean副本将被写入，申请C独占共享集

本地Cache控制器监听本地CPU的数据请求地址，根据该地址查询本地数据cache：

Cache块当前状态	查询结果	Cache控制器操作	Cache块状态变更
I	RdHit（不可能）		
S	RdHit	read data	S
M	RdHit	read data	M
I	RdMiss	send RdMiss(K) to B, 待收到Data(K)后read data	S
S	RdMiss	此时clean副本将被替换，send Quit(K1) to B1, send RdMiss(K2) to B2, 待收到Data(K2)后read data	S
M	RdMiss	此时dirty副本将被替换，send Update(K1) to B1, send RdMiss(K2) to B2, 待收到Data(K2)后read data	S
I	WtHit（不可能）		
S	WtHit	send Invalidate(K) to B, write data	M
M	WtHit	write data	M
I	WtMiss	send WtMiss(K) to B, 待收到Data(K)后write data	M
S	WtMiss	此时clean副本将被替换，send Quit(K1) to B1, send WtMiss(K2) to B2, 待收到Data(K2)后write data	M
M	WtMiss	此时dirty副本将被替换，send Update(K1) to B1, send WtMiss(K2) to B2, 待收到Data(K2)后write data	M

远程节点C响应目录节点B，根据B的请求操作自己的数据cache：

Cache块当前状态	B的请求	Cache控制器操作	Cache块状态变更
I	Invalidate(K)（不可能）	Null	I
I	FetchS(K)（不可能）	Null	I
I	FetchI(K)（不可能）	Null	I
S	Invalidate(K)	Null	I
S	FetchS(K)（不需要，B自有数据）	Null	S
S	FetchI(K)（不需要，B自有数据）	Null	I
M	Invalidate(K)（不可能，自毁唯一副本）	Null	I
M	FetchS(K)	send Update(K) to B	S
M	FetchI(K)	send Update(K) to B	I

目录节点B响应A，根据A发来的请求操作自己的目录信息：

目录项当前状态	收到消息类型	B的操作	目录项状态变更
U	RdMiss	send Data(K) to A, G={A}	S
U	WtMiss	send Data(K) to A, G={A}	E
S	RdMiss	send Data(K) to A, G+={A}	S
S	WtMiss	send Data(K) to A, send Invalidate(K) to all Cs in G , G={A}	E
E	RdMiss	此时C有唯一副本，send FetchS(K) to C, 待收到Update(K)后，update mem，send Data(K) to A, G+={A}	S
E	WtMiss	此时C有唯一副本，send FetchI(K) to C, 待收到Update(K)后，send Data(K) to A, G={A}	E

目录节点B响应C，根据C发来的请求操作自己的目录信息：

目录项当前状态	收到消息类型	B的操作	目录项状态变更
U	Quit(K) (不可能)	Null	U
U	Update(K)(不可能)	Null	U
U	Invalidate(K)(不可能)	Null	U
S	Quit(K)	G-={C}，# of G ?= 0	S/U
S	Update(K)(不可能)	S表示存储器中有新值，无需fetch，排除了FetchS和FetchI；C中的dirty副本将被替换时，此目录项状态应该为M，而不该为S，因此S的情况下不会收到此消息	S
S	Invalidate(K)	G={C}，send Invalidate(K) to all other Cs in G	E
E	Quit(K) (不可能)	Null	E
E	Update(K)	E表示存储器中为旧值，C中为新值， if fetchS then G={C}, state S if fetchI then G={}, state U if C中的dirty副本将被替换，update mem and state U	S/U
E	Invalidate(K)(不可能)	Null	E

互连结构

概念

互连结构是由开关元件按照一定拓扑结构连接并按照一定控制方式工作而形成的结构，用以实现计算机系统内多个功能部件间的相互连接。

三要素

互连拓扑：描述连接通路的拓扑结构

开关元件：描述连接通路的开关状态

控制方式：描述连接通路的操作规则

分类

根据互连网络中连接通路是否能够被共享，可以分为静态互连网络和动态互连网络。

静态互连网络，网络节点固定地与开关单元相连，以建立节点与节点之间的被动连接通路。（一维，环网，带弦环，树形，网格型，超立方）

动态互连网络，网络节点只与位于互连网络边界上的开关单元相连，开关单元之间可以按照应用需求动态的改变连接组态，以建立节点与节点之间的主动可控连接通路。（总线，交叉开关，多级互连）

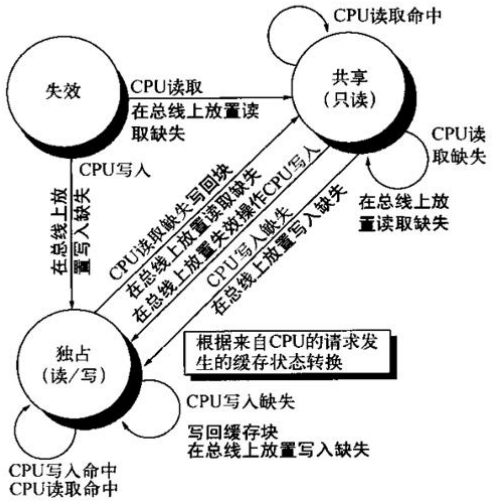
加速比定律

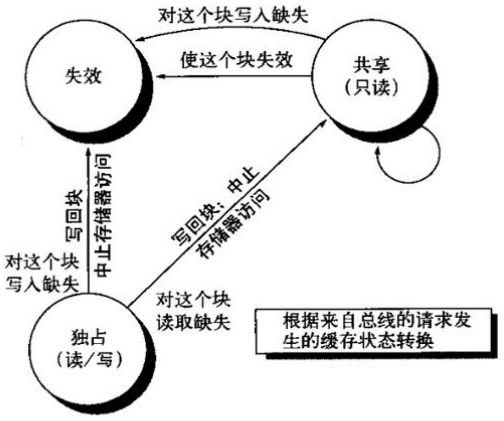
Amdahl 定律：当计算负载固定时适用

Gustafson 定律：当问题具备可扩放性时适用

Sun&Ni 定律：当存储器受限时适用

监听式协议的状态转换图





目录式协议的状态转换图