

高性能计算导论 | 超级计算机与高性能计算集群简介

(Introduction to HPC - Training Camp | Supercomputers and HPC Cluster Intro.)

李岳昆

Public Safety Institution of Big Data
Taiyuan University of Technology

2022 年 1 月 20 日



太原理工大学
TAIYUAN UNIVERSITY OF TECHNOLOGY



PUBLIC SAFETY
INSTITUTION OF BIG DATA

- 计算机架构史
- 并行计算机体系结构
- 并行计算访存模型
- 高性能计算集群简介
- 浪潮天梭 10000 集群架构与软件系统
- 集群节点架构分类
- 高性能计算网络互联
- 基准测试与 Top 500 榜单
- 分布式计算、网格计算、云计算

从算盘到编程计算机

人们最早设计了固定程序计算机 (Fixed Program Computer)，如：算盘、计算器等，这种计算机的特点是无法编程，只能解决固定的问题，通用性较差。这显然不符合计算机的设计初衷，如何设计出可编程的计算机？

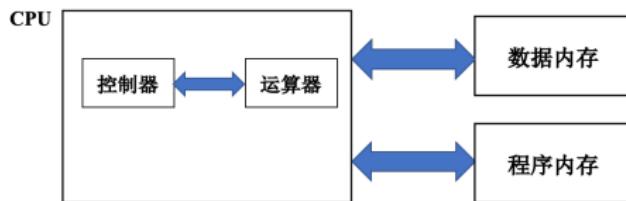


图: Harvard 架构：程序与指令存储在不同内存

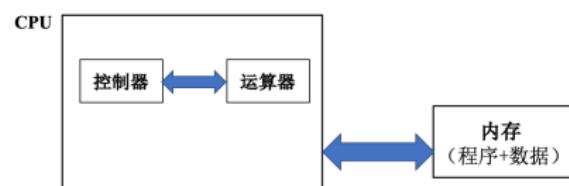


图: Princeton 架构：程序与指令存储在同一内存

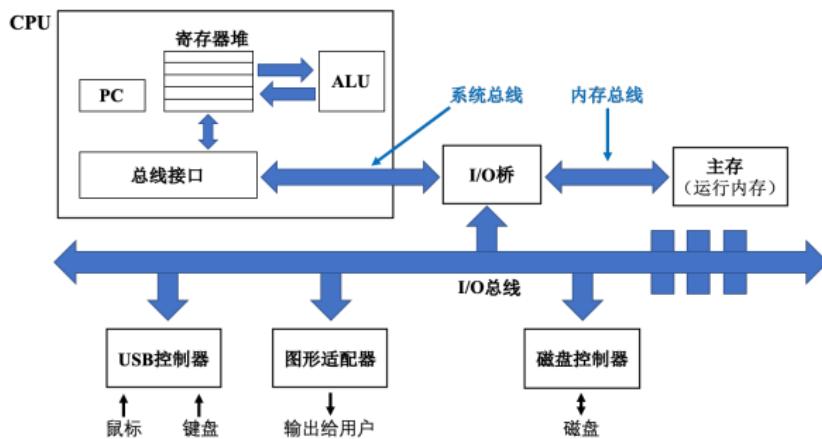
Harvard 架构

Harvard 架构设计的初衷正是为了减轻程序运行时 CPU 和存储器信息交换的瓶颈，其 CPU 通常具有较高的执行效率。

Princeton 架构 (冯·诺依曼架构)

尽管当代计算机是以冯·诺依曼架构为主，但这并不意味着 Harvard 架构是错的，事实上两者各有利弊。冯·诺依曼架构的瓶颈为：运算器的速度太快，CPU 与内存之间的路径太窄，以至于内存无法及时给运算器提供“材料”，这正是影响性能的致命因素。通常情况下，我们将冯·诺依曼架构的缺陷归结为访存墙 (Memory Wall)，即：

- 计算机具有单一的线性内存，指令和数据只有在使用时才进行隐式区分；
- 总性能受到内存的读写总线所能提供的延迟和带宽限制。



图：当代计算机架构

向量机

ENIAC 之后，大部分处理器都是将数据一个一个输入到运算器，这在某种程度上限制了运算器的发挥。为了追求更高性能，人们希望设计出一款用一条指令就能运算多条数据的机器，即数据以向量的形式被处理。于是向量机 (Vector Machine) 应运而生。

Seymour Cray 在 1972 年创建自己的公司 Cray Research Inc., 之后诞生这家公司第一个产品 Cray-1 超级计算机, 成为当时速度最快的计算机。与一次只能处理一个数据的标量处理器正相反, 向量处理器可以在特定工作环境中极大地提升性能, 尤其是在数值模拟或者相似领域, 这使得它远远超过同时代的其他机器。

超级计算机的雏形

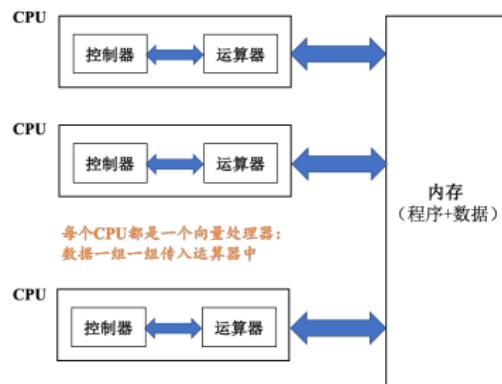
向量处理器最早出现于 20 世纪 70 年代早期，并在 70 年代到 90 年代期间成为超级计算机设计的主导方向。



图: Cray-1 与“超级计算机之父”Cray

并行向量机

随后的 70 年代到 90 年代之间，人们又基于向量机设计出了并行向量处理器（Parallel Vector Processors, PVP），即同时布置多个向量机并通过共享内存实现交互。并行向量处理器最大的特点是系统中拥有多个 CPU(即处理器)，同时每个处理器都是由专门定制的向量处理器（VP）组成。



图：并行向量处理器（每个 CPU 都是一个向量机）



图：Cray-2 代并行向量处理器

向量处理指令

由于 90 年代末常规处理器设计性能提升，而价格快速下降，基于向量处理器的超级计算机逐渐让出了主导地位，但这并不意味着向量处理器已经过时。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(单一指令、多重数据)。常见的例子有 VIS、MMX、SSE、AltiVec 和 AVX。向量处理技术也能在游戏主机硬件和图形加速硬件上看到。在 2000 年，IBM，东芝和索尼合作开发了 Cell 处理器，集成了一个标量处理器和八个向量处理器，应用在索尼的 PlayStation 3 游戏机和其他一些产品中。

表: Cray-1 向量机

处理器个数	处理器频率	内存大小	存储大小	性能
1	80MHz	8.39MB	303MB	160 MFLOPS

表: Cray-2 多处理器

处理器个数	处理器频率	内存大小	总性能
4	244MHz	256MB	1.9GFLOPS

分布式并行机

Cray-2 代并行机之后，一些厂商又提出了分布式并行机 (Parallel Processors, PP)：通过高性能网络连接多个分布式存储节点，每个节点由商用微处理芯片组成。

对称多处理机

与分布式并行机几乎同时代又出现了另一种并行计算机架构：对称多处理机 (Symmetric Multiprocessors, SMP)，它通过高性能网络连接多个高性能微处理芯片，芯片之间通过共享内存交互。

表: Intel Paragon XP/S 140 并行机与 SUN Ultra E10000 多处理机

	处理器个数	处理器频率	内存大小	访存带宽	网络带宽	总性能
Intel	3680	50 MHz	128 MB	400 MB/s	175 MB/s	143 GFLOPS
SUN	64	1 GFLOPS	250 MHz	64GB	12.8 GB/s	25 GFLOPS

中后期：分布式并行机与对称多处理机

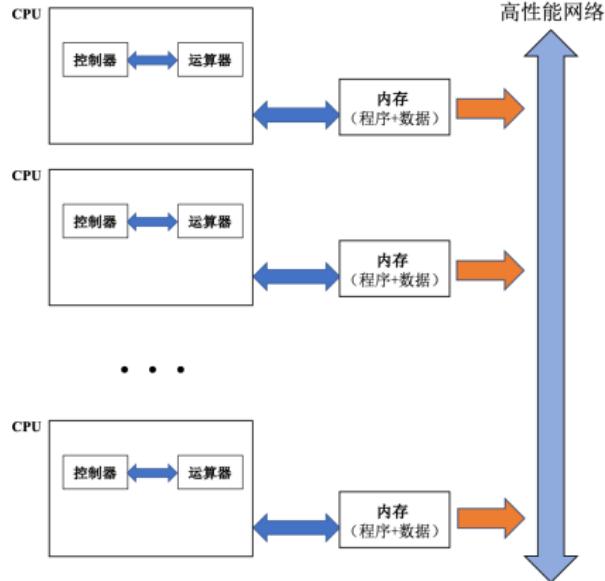


图: PP

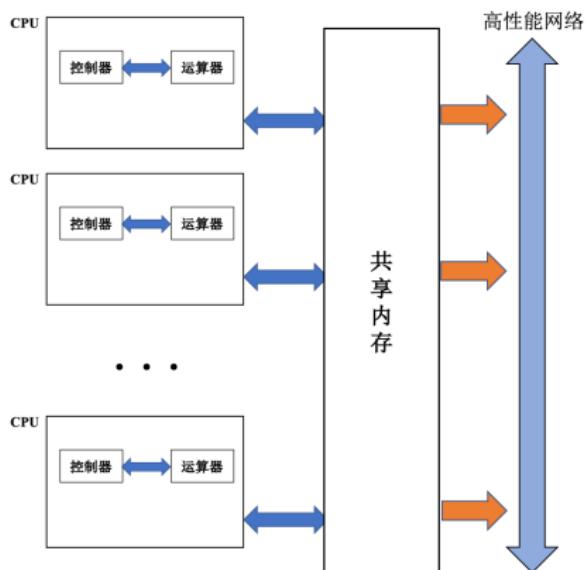


图: SMP

分布式共享并行机

另有分布式共享并行机 (Distributed Share Memory, DSM) 通过高性能网络连接多个高性能微处理芯片，每个芯片拥有局部内存，但所有局部内存都能实现全局共享。

SMP、MPP、DSM 和 COW 并行结构渐趋一致。

- 大量的节点通过高速网络互连起来；
 - 节点遵循 Shell 结构：用专门定制的 Shell 电路将商用微处理器和节点的其它部分（包括板级 Cache、局存、NIC 和 DISK）连接起来。优点是 CPU 升级只需要更换 Shell。

并行计算机体系种类总结：

- PVP(Parallel Vector Processor): 并行向量处理器
 - SMP(Symmetric Multiprocessors): 对称多处理机
 - MPP(Massively Parallel Processing): 大规模并行处理
 - DSM(Distributed Share Memory): 分布式共享并行机
 - COW(Cluster Of Workstation): 工作站集群

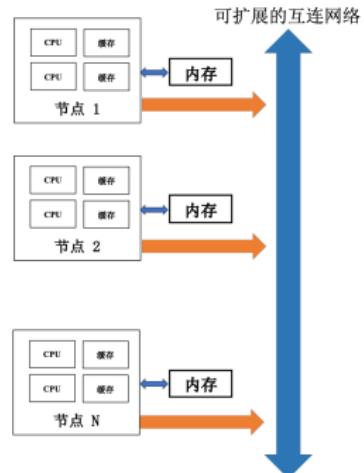


图: DSM



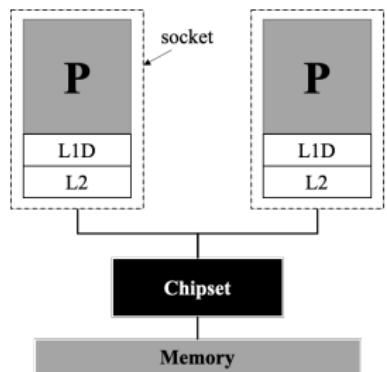
并行计算机体系结构对比

属性	PVP	SMP	MPP	DSM	COW
结构类型	MIMD	MIMD	MIMD	MIMD	MIMD
处理器类型	专用定制	商用	商用	商用	商用
互连网络	定制交叉开关	总线、交叉开关	定制网络	定制网络	商用网络(以太 ATM)
通信机制	共享变量	共享变量	消息传递	共享变量	消息传递
地址空间	单地址空间	单地址空间	多地址空间	单地址空间	多地址空间
系统存储器	集中共享	集中共享	分布非共享	分布共享	分布非共享
访存模型	UMA	UMA	NORMA	NUMA	NORMA
代表机器	Cray C-90, Cray T-90, 银河1号	IBM R50, SGI Power Challenge, 曙光1号	Intel Paragon, IBMSCP2, 曙光1000/2000	Stanford DASH, Cray T3D	Berkeley NOW, Alpha Farm

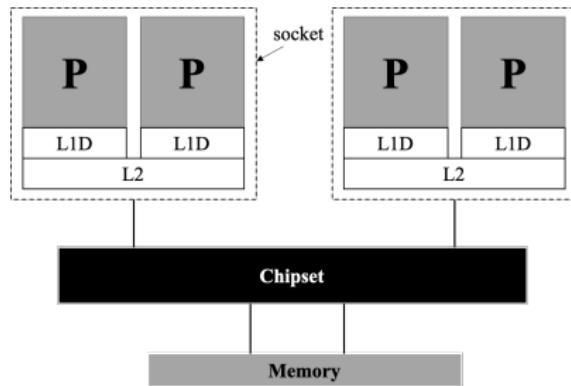
图: 五种结构特性一览表

一致存储器访问结构：UMA

所谓对称多处理器结构，是指服务器中多个 CPU 对称工作，无主次或从属关系。各 CPU 共享相同的物理内存，每个 CPU 访问内存中的任何地址所需时间是相同的，因此 SMP 也被称为一致存储器访问结构 (UMA: Uniform Memory Access)。对 SMP 服务器进行扩展的方式包括增加内存、使用更快的 CPU、增加 CPU、扩充 I/O(槽口数与总线数)以及添加更多的外部设备(通常是磁盘存储)。



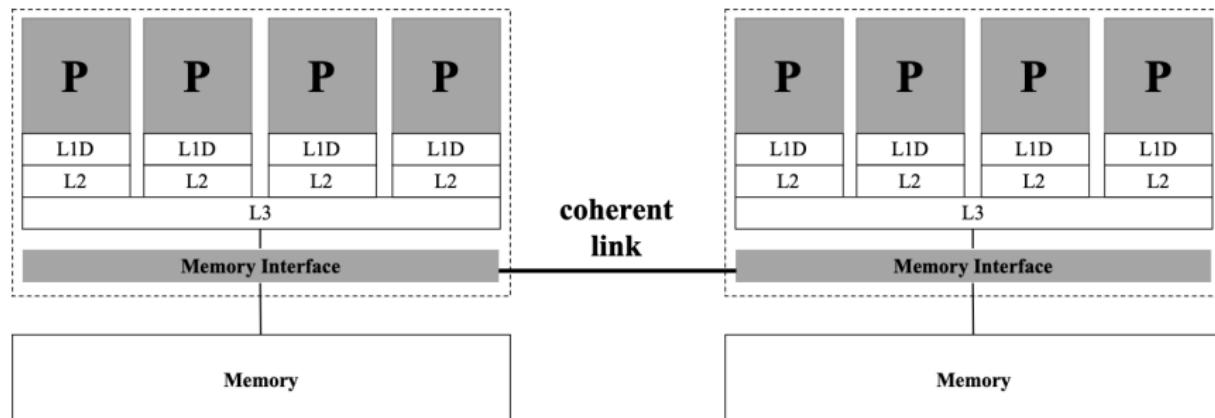
图：双路单核的 UMA 架构



图：双路双核的 UMA 架构

缓存一致性的非一致内存访问架构：cc-NUMA

上述 UMA 架构的问题在于：每个插槽内各处理器只共享 L2 级缓存，但并不共享内存，这会导致处理器和内存之间速度不匹配等问题。如何改进这种弊端？我们在每个插槽内部共享一个内存，并令各个插槽内共享的缓存数据同步，即著名的缓存一致性的非一致内存访问架构 (cache-coherent Nonuniform Memory Access, ccNUMA) 架构。



图：双路四核的 ccNUMA 架构



其他并行计算机访存模型

除上述内容外，并行计算机还有一些常见模型：

- ① NUMA(Nonuniform Memory Access): 一致内存访问架构模型/非均匀存储访问
 - 被共享的存储器在物理上是分布在所有的处理器中的，其所有本地存储器的集合就组成了全局地址空间；
 - 处理器访问存储器的时间是不一样的：访问本地存储器 LM 或群内共享存储器 CSM 较快，而访问外地的存储器或全局共享存储器 GSM 较慢（此即非均匀存储访问名称的由来）；
 - 每台处理器照例可带私有高速缓存，外设也可以某种形式共享。
- ② COMA(Cache-Only Memory Access): 全高速缓存存储访问模型
 - 各处理器节点中没有存储层次结构，全部高速缓存组成了全局地址空间；
 - 利用分布的高速缓存目录 D 进行远程高速缓存的访问；
 - COMA 中的高速缓存容量一般都大于 2 级高速缓存容量；
 - 使用 COMA 时，数据开始时可任意分配，因为在运行时它最终会被迁移到要用到它们的地方。
- ③ NORMA(No-Remote Memory Access): 非远程存储访问模型
 - 所有存储器是私有的，仅能由其处理器访问；
 - 绝大多数 NORMA 都不支持远程存储器的访问。

高性能计算

高性能计算是一套计算性能强大、大规模存储空间、完整的软件系统、价格昂贵的计算机。它不仅面向科学计算，而且面向拥有大量数值分析、数据处理等更广泛的应用。因此它不同于超级计算机仅追求单一计算指标，而是追求全面的综合指标。

- 高性能计算集群 (High Performance Computing Cluster):

- 也称科学计算集群，通过多个普通节点的并行计算实现强大的计算功能。
- 分为计算密集型、通讯密集型与数据密集型。

- 高可靠性集群 (High Availability Cluster):

- HA 集群以最大限度保证系统可考性，减少宕机时间为目的，通过把出现故障的节点上的应用程序转移到没有出现故障的节点上来实现。

- 负载均衡集群 (Load Balance Cluster):

- 负载均衡集群使负载可以在计算机集群中尽可能平均地分摊处理。通常包括应用程序处理负载和网络流量负载。
- 这样的系统非常适合向使用同一组应用程序的大量用户提供服务。



图: TS 10k Cluster

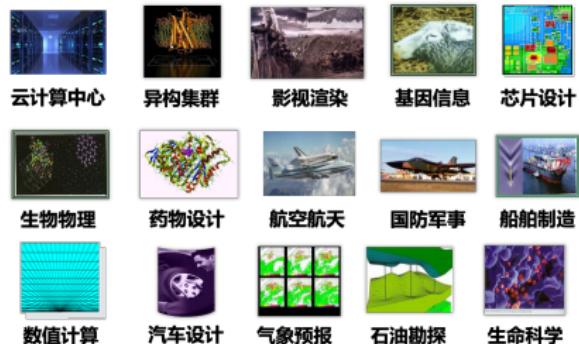


图: 高性能集群市场应用

- 生命科学: 基因排序、蛋白质折叠
- 计算化学: 量子化学、材料模拟
- 分子动力学: 分子运动模拟、可视化
- CFD & CAE: 流体动力学、计算机辅助工程
- 气象环境: 数值气象预报、环境污染监测
- 渲染可视化: 动漫渲染、影视制作
- 石油天然气: 油藏模拟、地震模拟
- 计算数学: 算法开发、数据可视化

高性能计算与并行计算

并行计算 (Parallel Computing) 是指同时使用多台计算机协同合作解决计算问题的过程，其主要目的是快速解决大型且复杂的计算问题 (规模和速度)。对比高性能计算并按照概念广泛程度排序，通常有：

高性能计算 > 并行计算 > 分布式计算 > 超级计算

管理节点: mu01, mu02, ...

管理节点是集群系统各种管理措施的控制节点，负责监控集群中各个节点和网络的运行状况。通常集群的管理软件也运行在这个节点上。由于现代服务器强大的处理能力，通常集群的作业调度程序（如 PBS）也可以运行在这个节点上。通常管理节点是计算网络中的关键点，如果它失效，所有的计算节点都会失效（已提交的作业很有可能会挂起）。所以管理节点也应该有硬件冗余保护。



- 管理节点对计算性能要求不高，一套集群配置一到两台双路机架式服务器即可满足。

图: 管理节点/登录节点

登录节点: ln01, ln02, ...

登录节点相当于用户访问集群系统的网关。用户通常登录到这个节点上编译并运行作业。为了保证用户节点的高可用性，登录节点应该采用硬件冗余的容错方法，如采用双机热备份。或至少应该采用 RAID(Redundant Array of Independent Disks) 技术保证。

浪潮天梭 10000 集群架构

计算节点: cu01, fat02, gpu03, knl04, ...

计算节点是整个集群系统的计算核心。它的功能就是使用本地的 CPU、内存、加速卡等资源完成计算任务。用户需要根据自身需求和预算来决定采用何种配置。

- ① 普通节点：出于预算考虑，集群的计算主力由配置相同的大量普通节点构成，一般为双路机架式/刀片式服务器，主要用于执行 MPI 并行运算。
 - ② 胖节点：用于 SMP(Symmetrical Multi-Processing) 运算或对内存需求特别大的 MPI(Message Passing interface) 运算等。
 - 集群规模越来越大，节能和节省空间成为客户的重要需求之一，刀片服务器已成为主流。
 - 胖节点一般采用四路、八路服务器，内存强大，单机计算能力强，价格比较昂贵。



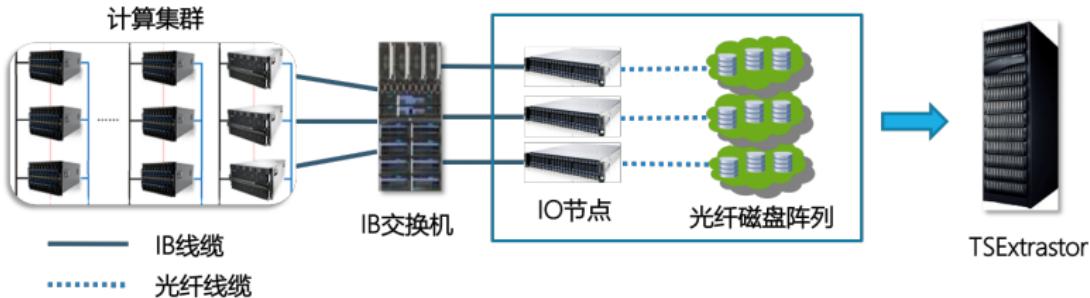
图：刀片服务器（左）与胖节点（右）



图：胖节点



浪潮天梭 10000 集群存储系统



图：集群架构与存储系统

高性能存储系统的解决方案

- 传统解决方案：硬件采用 I/O 节点 + 光纤磁盘阵列；软件采用 NFS 协议；网络接口使用以太网或 IB。
 - 升级解决方案：将光纤磁盘阵列升级为 IB 磁盘阵列，从而提高磁盘阵列的访问性能。
 - 终极解决方案：并行存储系统，实现多个访问通道和多个存储的并发读写和单一访问空间。



并行文件系统——beegfs

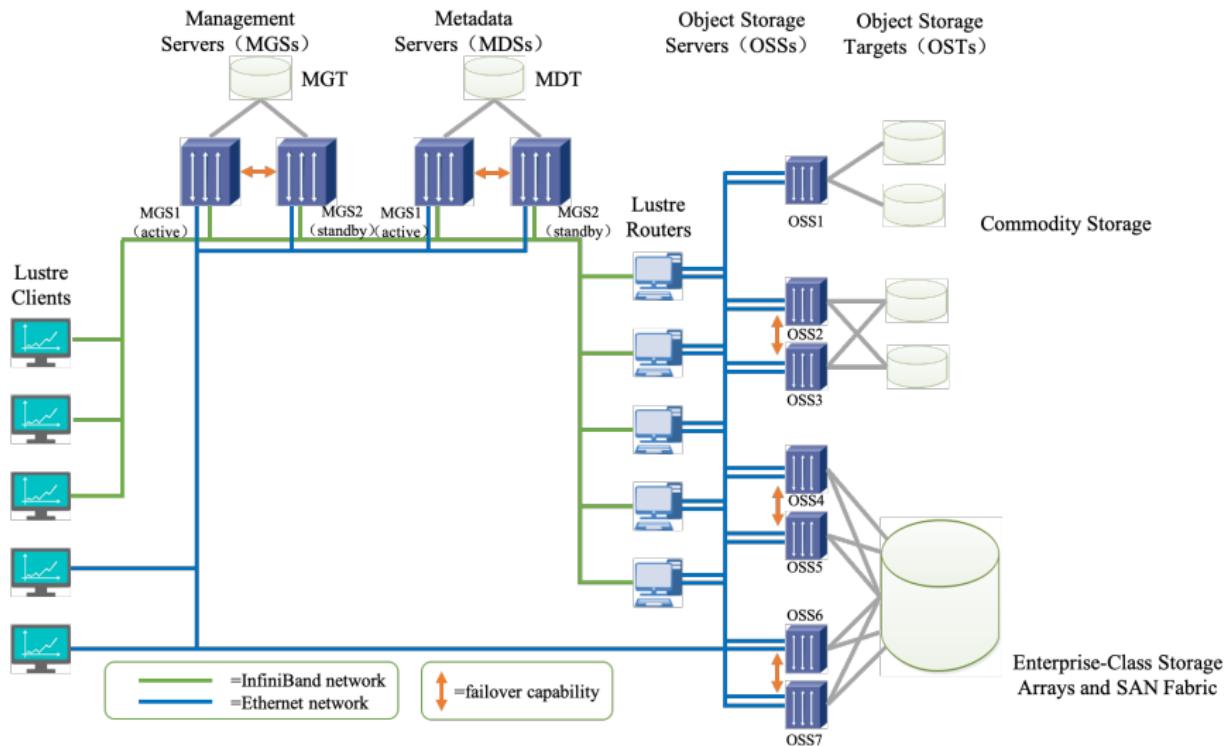


图: beegfs 架构

不同计算网络性能对比

千兆和万兆以太网络		Infiniband	
优点	缺点	优点	缺点
低延迟，高带宽	延迟比某些特定传输协议如InfiniBand, Myrinet高一些	极低延迟，高带宽	昂贵，距离短，线缆比较粗大而局限了布线。
有通用的标准，协议应用广泛	RNIC 10G比较新而且贵	成熟的传输管理：QoS和流量管理	IB的集中子网管理不如分布协议快(IEEE)
便宜	如果没有采用TOE和RDMA的话CPU负载高	比10G以太网便宜	比较新而且复杂的协议需要培训
RDMA, iWARP, TOE, iSCSI协议目前都已经可用	RDMA over IP 目前没有统一的协议	已经有基于IB的存储	有限的管理和支持工具
产品成熟灵活		协议本身支持RDMA和SRP	HCA需要同交换机近距离连接，升级和互操作检测都比较麻烦。
成熟的传输管理QoS，HA等		SDP允许IP跑在IB RDMA上	
布线简单		有工业标准	
管理及支持工具众多			

图：不同计算网络对比

天梭 TS10000 软件系统组成：

- ① RedHat/CentOS Linux 操作系统
- ② 浪潮集群监控管理软件
- ③ 集群作业调度系统
 - TSCE 作业调度系统
- ④ 集群并行环境
 - MPICH MPI 消息传递接口软件
 - Infiniband 专用 MPI
- ⑤ 集群开发环境
 - GNU C, C++, GNU F77/90
 - Intel 编译器 C/C++/Fortran, Intel 高性能数学库 MKL
 - Intel Vtune
 - Vampir(Intel Trace Analyzer/Trace Collector)

作业调度及集群监控软件 TSCE(天梭 Cluster Engine)：统一的集群使用与管理平台，通过该平台可以完成对集群的使用与管理工作。主要功能如下：

- 监控功能：包括集群整体监控、物理视图、资源监控、性能监控、节点监控
- 报警功能：支持界面、邮件、短信报警
- 管理功能：集群 shell、远程桌面、文件管理，用户管理、队列管理
- 大屏幕监控：可定制并动态添加监控页面，用于大屏幕展示
- 记账统计：消费记录查询、缴费记录、余额查询；统计集群历史使用状态

计算中心集群概况

整个集群设备包含：

- 1 台管理节点；
- 2 台登录节点；
- 50 台机架服务器作为计算节点，4 台 I/O 节点组成 beegfs 分布式文件系统；
- 多台 Infiniband 交换机，以及多台千兆以太网交换机。

目前集群队列：cu/fat 队列。各节队列核心数为：2200/192C，其中：

cu 队列：50 节点 \times 44 核心； fat 队列：1 节点 \times 192 核心。



图：集群架构

集群架构

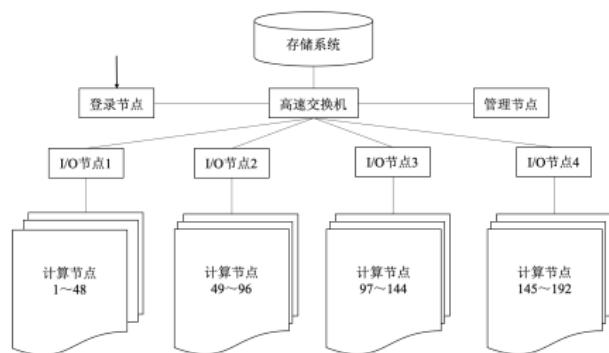
从硬件结构上，高性能计算系统包含计算节点、IO 节点、登录节点、管理节点、高速网络、存储系统等组成。计算节点是高性能集群中的最主要的计算能力的体现，目前，主流的计算节点有同构节点和异构节点两种类型。

- ④ 同构计算节点是指集群中每个计算节点完全由 CPU 计算资源组成，目前，在一个计算节点上可以支持单路、双路、四路、八路等 CPU 计算节点。Intel 和 AMD CPU 型号、参数详见：

<http://www.techpowerup.com/CPUdb>

- ② 异构计算技术从 80 年代中期产生，由于它能经济有效地获取高性能计算能力、可扩展性好、计算资源利用率高、发展潜力巨大，目前已成为并行/分布计算领域中的研究热点之一。异构计算的目的一般是加速和节能。目前，主流的异构计算有：

- CPU + GPU;
 - CPU + MIC;
 - CPU + FPGA.



图：高性能计算集群

随着超算应用增多和对计算量需求的增大，CPU 逐渐在某些领域中显露疲态。为解决这个问题，异构运算应运而生。异构是指与传统 CPU 不同架构的计算设备，如 GPU、MIC 等新型计算设备。这些新的计算设备通常拥有远高于 CPU 的并行计算能力。GPU 原本并非用于高性能计算，而是用于图形显示。但 GPU 所特有的硬件架构，天然适合高并发度的计算。但利用 GPU 编程，需要将通用问题转化为图形显示问题，因此编程门槛较高，很少有人利用 GPU 进行高性能计算并行应用开发。



图：“这里是石油巨头埃克森美孚 (Exxon-Mobil) 的地产，1996 年，这里来了一个新主人——美国电报和电话公司 (AT&T) 实验室”图为弗伦翰公园日落—选段摘自吴军博士：《浪潮之巅》

异构开发环境

异构开发环境有先进精简指令机 (advanced RISC machine, ARM)、现场可编程门阵列 (field programmable gate array, FPGA) 等非 x86 架构环境。

异构环境并非抛弃 CPU，而是 CPU 与心的计算原件相结合，并行进行计算。无论是 GPU 还是 MIC，都是以 PCI-E 接口与现有 x86 节点进行连接，以协处理器的方式与现有节点进行集成。因此在不破坏原有集群的情况下，增加一部分算力。当前，采用异构集群的高性能计算集群在 Top500 中越来越多，表明异构集群方案越来越多地被业界采用。

CPU 并行架构

一般来说，集群中每个节点都使用相同的配置，以方便管理和性能优化。节点一般为标准机架式服务器，配有双路或四路的多核服务器版 CPU(如 Intel Xeon 系列)，每个核心搭配 4GB 内存或更多。节点之间采用以太网线(千兆网线或万兆网线)或 InfiniBand 网线，通过相应的交换机互相连接。节点与节点属于同一个内网，并各自拥有不重复的内网 IP 地址，通常集群与外网物理隔离，以保证集群的安全性。由于上述高性能并行架构采用的是分布式内存结构，因此并行软件环境需要采用消息传递的网络通信方式，如：MPI。

MPI 环境配置步骤：

- ① 解压、安装：解压安装包后使用`./configure; make`命令安装。
- ② 放权：在整个集群的每一个节点机上都建立相同的账户名，使得 MPI 程序在相同的账户下运行。放权设置完成后，可以在任何一个节点，使用相同的账户名，不需要输入密码。
- ③ 运行：程序在不同节点放置在共享存储时最佳。使用哪些节点、每个节点内开启几个 MPI 进程，是由配置文件决定的，每个应用可以采用不同的节点配置文件，以便根据不同的应用选择不同的方案，避免闲置计算资源或达不到最好的利用效果。

软件部分	种类
编译器/调试工具	GNU、GDB
高性能函数库	Intel MKL (商业软件)
性能调试工具	gprof (编译时添加`-pg`) Intel VTune (商业软件)

图：CPU 并行架构软件环境

CPU+MIC 异构

异构环境中 GPU 和 MIC 是以附加板卡的形式附着于现有系统的，因此在硬件环境上，不需要对集群的整体结构进行任何改变，只需要在节点内部增加协处理器或将节点内部增加协处理器或将节点更换为可以使用协处理器的节点。

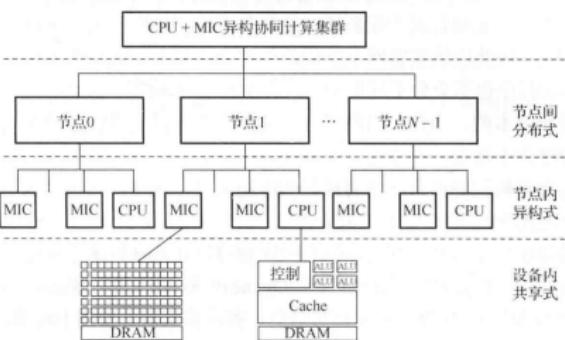


图: CPU+MIC 异构

还有三点特别之处需要注意：

- ① 传统并行应用环境中节点，不一定有多余或合适的 PCI-E 插槽，也就不一定能够附加 GPU 或 MIC 卡。但是异构并行开发环境中，其中的节点必须是传统并行环境节点中拥有 PCI-E 接口，可以增加 GPU 或 MIC 卡的类型。
- ② 由于增加了协处理器，所以节点的功耗需求大大增加，通常每增加一块协处理器，需要增加 300W 左右的电源供应。
- ③ 由于增加了协处理器，所以对节点的散热提出了更高的要求。在搭建异构集群时，不仅需要看节点是否能够支持协处理器，还要注意节点的散热能力。

不同异构方式的软件环境

通用软件环境与传统并行应用软件环境极为类似，不仅同样可以分为操作系统、并行环境等方面，而且异构模式下的通用软件环境，也基本与传统软件环境相同。在并行环境方面，一般需要选择 OpenMP、MPI 等通用并行库，如果支持 InfiniBand 硬件，则尽量使用 InfiniBand 驱动。在并行运行环境方面，MIC 支持标准的 OpenMP 和 MPI 库，但是如果想充分利用 MIC 的特性，则需要安装使用 Intel 的 MPI 库。

相关资源

- ① IDF：英特尔信息技术峰会是由 Intel 公司主办的技术讲座，在美国、中国等 7 个地区举办，每年分春秋两次。IDF 主要是由主题演讲、技术专题讲座和技术展示组成，主题演讲的演讲者均是 Intel 的高层人士，演讲的题目都是具有相当前瞻性；
- ② MIC 计算论坛；
- ③ 更多资源可以访问：英特尔开发者官网。

软件部分	种类
编程语言	C、C++、Fortran、OpenCL（最广泛）
编译器/调试工具	Intel 的 MIC 编译器/IDB
数学库	MIC 高性能数学库、Intel MKL
性能调优工具	Intel VTune（其他大部分 Intel 工具也都可以应用）

图：CPU+MIC 异构



运行库

运行时库包含异构程序运行时需要用到的库的集合，这些库调用了驱动程序的一些接口，使程序能够运用 GPU 硬件进行计算。需要特别注意的是，驱动程序必须在每个节点上安装，因为每个节点都有 GPU 卡，所以需要在每个节点的操作系统上配置驱动程序。而对于运行时库来说，仅在应用程序需要时才会用到，因此可以在共享目录中安装一份，以供全部节点使用。这种方式的好处是：减少安装和维护的工作量，可以统一管理，一次配置多次使用。但要注意的是对该目录下文件写权限的控制，以避免用户无意识地破坏文件，造成全局软件故障。为适应广泛的市场需要，针对不同的市场进行划分，NVIDIA 及其合作伙伴共同开发了多种多样的编程方式，有 CUDA C/C++、CUDA Fortran、OpenACC、HMPP、CUDA-x86、OpenCL、JCuda、PyCUDA、Direct Compute、MATLAB、Microsoft C++ AMP 等。

多线程编程

多线程编程语言可以分为两类，一种为编译指导，即在串行程序之前添加编译指令，指导编译器将串行指令自动编译为并行程序；另一种为显式线程模型，用户可以直接编写并行程序，显式地调用 GPU 线程。相对地，编译指导比较简单，有易于初学者的学习、方便串行程序并行化、开发周期短等优点。但是，因为编译器自动完成串行程序的并行化，所以对 GPU 的操作相对不够灵活。Nsight 是 NVIDIA 开发的一套集成了编译、调试与性能分析功能的开发环境，可以从官网下载。

网络性能指标：

- 节点度 (Node Degree)：射入或射出一个节点的边数。在单向网络中，入射和出射边之和称为节点度。
- 网络直径 (Network Diameter)：网络中任何两个节点之间的最长距离，即最大路径数。
- 对剖宽度/二分宽度 (Bisection Width)：对分网络各半所必须移去的最少边数
- 对剖带宽/二分带宽 (Bisection Bandwidth)：每秒钟内，在最小的对剖平面上通过所有连线的最大信息位 (或字节) 数
- 如果从任一节点观看网络都一样，则称网络为对称的 (Symmetry)

静态互联网络

静态互连网络：处理单元间有着固定连接的一类网络，在程序执行期间，这种点到点的链接保持不变；典型的静态网络有一维线性阵列、二维网孔、树连接、超立方网络、立方环、Round-Robin 交换网、蝶形网络等。

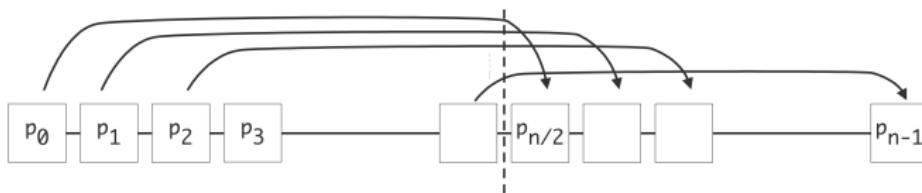
动态网络

用交换开关构成的，可按应用程序的要求动态地改变连接组态；典型的动态网络包括总线、交叉开关和多级互连网络等。

一维线性阵列 (1-D Linear Array)

并行机中最简单、最基本的互连方式，每个节点只与其左、右近邻相连，也叫二近邻连接。 N 个节点用 $N - 1$ 条边串接之，内节点度为2，直径为 $N - 1$ ，对剖宽度为1。当首、尾节点相连时可构成循环移位器，在拓扑结构上等同于环，环可以是单向的或双向的，其节点度恒为2，直径或为 $\lfloor N/2 \rfloor$ (双向环)或为 $N - 1$ (单向环)，对剖宽度为2

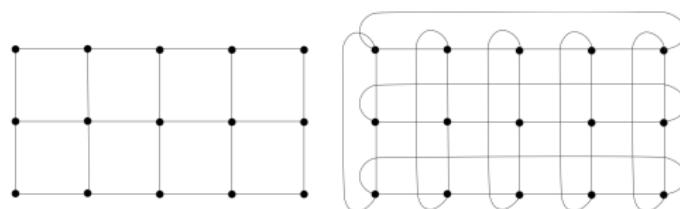
- 除了“一个消息从处理器 A 到处理器 B 需要多长时间”的问题外，我们还经常担心两个同时进行的消息之间的冲突：是否存在两个同时进行的消息需要使用同一网络链接的可能性？
- 在下图中，我们说明了如果每个处理器 p_i 在 $i < n/2$ 的情况下向 $p_{i+n/2}$ 发送消息会发生什么：会有 $n/2$ 的消息试图通过 $p_{n/2-1}$ 和 p_n 之间的线路。这种冲突被称为拥堵 (congestion) 或争夺 (contention)。显然，一台并行计算机的链接越多，发生拥堵的机会就越小。



图：由于同时发出的信息，对网络链接的争夺

$\sqrt{N} \times \sqrt{N}$ 二维网孔 (2-D Mesh)

每个节点只与其上、下、左、右的近邻相连（边界节点除外），节点度为 4，网络直径为 $2(\sqrt{N} - 1)$ ，对剖宽度为 N 。在垂直方向上带环绕，水平方向呈蛇状，就变成 Illiac 网孔，节点度恒为 4，网络直径为 $\sqrt{N} - 1$ ，而对剖宽度为 $2\sqrt{N}$ 。垂直和水平方向均带环绕，则变成了 2-D 环绕（2-D Torus），节点度恒为 4，网络直径为 $\lfloor N/2 \rfloor$ ，对剖宽度为 $2\sqrt{N}$ 。除了二维网孔外，二维组织方式还包括二叉树形式。



图：带有环形连接的二维网格

- 如果我们暂时接受物理模型需要近邻型通信，那么网状计算机是运行物理模拟的自然候选者。
 - 基于网格的设计通常有所谓的环绕或环形连接，它连接二维网格的左右两边，以及顶部和底部。
 - $n \times n \times n$ 的三维立方体的直径是多少？对剖宽度是多少？如果增加环绕环状的连接，会有什么变化？

超立方

一个 n -立方由 $N = 2^n$ 个顶点组成, n -立方的节点度为 n , 网络直径也是 n , 而对剖宽度为 $N/2$ 。

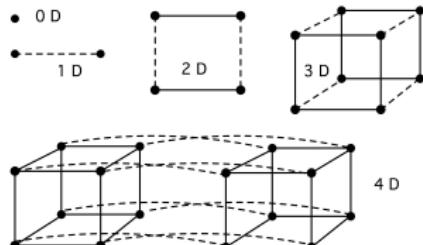


图: 超立方体

1D Gray code:	0 1
1D code and reflection:	0 1 : 1 0
2D Gray code:	append 0 and 1 bit: 0 0 : 1 1
2D code and reflection:	0 1 1 0 : 0 1 1 0
3D Gray code:	0 0 1 1 : 1 1 0 0 append 0 and 1 bit: 0 0 0 0 : 1 1 1 1

图: 格雷编码

- 用膨胀 (Dilation) 系数来描述嵌入的质量, 它是指被嵌入网络中的一条链路在所要嵌入的网络中对应所需的最大链路数, 若系数为 1, 则称为完美嵌入。

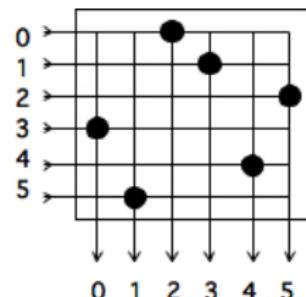
网络名称	网络规模	节点度	网络直径	对剖宽度	对称	链路数
线性阵列	N	2	$N - 1$	1	非	$N - 1$
环形	N	2	2	2	是	N
2-D 网孔	$(\sqrt{N} \times \sqrt{N})$	4	$\lfloor N/2 \rfloor$	\sqrt{N}	非	2
Illiac 网孔	$(\sqrt{N} \times \sqrt{N})$	4	2	$2\sqrt{N}$	非	$2N$
2-D 环绕	$(\sqrt{N} \times \sqrt{N})$	4	2	$2\sqrt{N}$	是	$2N$
二叉树	N	3	$2(\lceil \log N \rceil) - 1$	1	非	$N - 1$
星形	N	$N - 1$	2	$\lfloor N/2 \rfloor$	非	$N - 1$
超立方	$N = 2^n$	n	n	$N/2$	是	$nN/2$
立方体	$N = k \cdot 2^k$	3	$2k - 1 + \lfloor k/2 \rfloor$	$N/(2k)$	是	$3N/2$

图: 静态互连网络特性比较

交换机

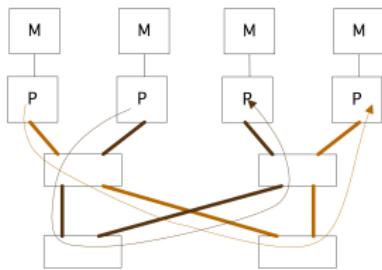
上面我们简要地讨论了完全连接的处理器。然而，通过在所有处理器之间制作大量的总线来进行连接是不切实际的。然而，还有另一种可能性，即通过将所有处理器连接到一个交换机 (switch) 或交换机网络 (switch network)。一些流行的网络设计是交叉开关 (Cross bar)、蝶形交换 (butterfly exchange) 和胖树 (fat tree)。

- 最简单的开关网络是一个交叉开关，由 n 水平线和垂直线组成，每个交叉点上都有一个开关元件，决定这些线是否连接在一起。如果我们把横线指定为输入，竖线指定为输出，这显然是让 n 输入映射到 n 输出的一种方式。每一个输入和输出的组合（有时称为“排列组合”）都是允许的。
- 这种类型的网络的一个优点是，没有任何连接可以阻挡另一个连接。主要的缺点是开关元素的数量是 n^2 ，是处理器数量 n 的一个快速增长的函数。



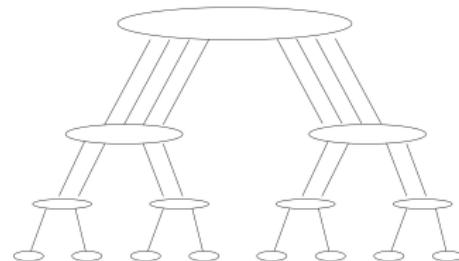
图：连接 6 个输入和 6 个输出的简单交叉开关

- 蝶形交换网络由小型交换元件构成，随着处理器数量的增加，层级的数量也随之增加。



图：通过蝴蝶交换网的两条独立路线

- 胖树是一个树状网络，每一级都有相同的总带宽，这样就不会出现这种拥堵问题：根部实际上会有 2^{k-1} 条进线和出线连接。



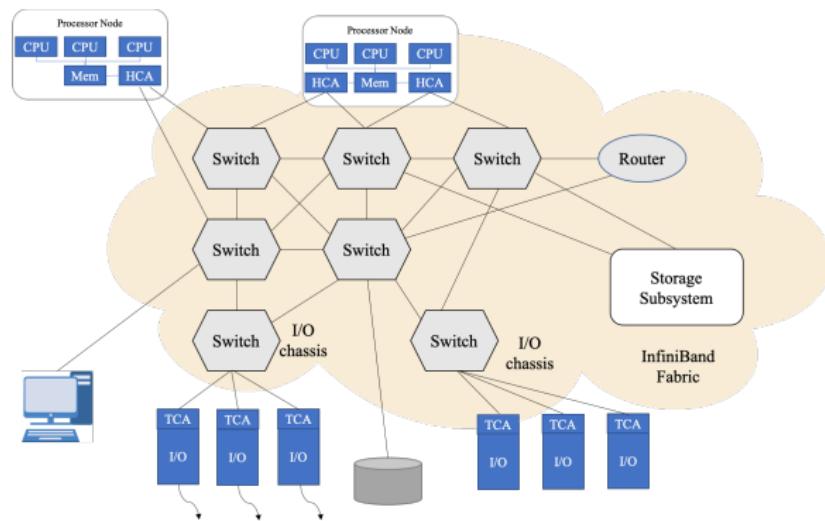
图：具有三级互连的胖树

其他动态互联网络

除上述内容外，互联网络还有级间互连 (Interstage Connection): 均匀 Round-Robin、蝶网、多路均匀 Round-Robin、交叉开关、立方连接等连接。 n 输入的 Ω 需要 $\log_2 n$ 级 2×2 开关，在 Illinois 大学的 Cedar[2] 多处理机系统中采用了 Ω 网络。

InfiniBand 网络

InfiniBand(简称 IB)是一个统一的互联结构，既可以处理存储 I/O、网络 I/O，也能够处理进程间通信(IPC)。它可以将磁盘阵列、SANs、LANs、服务器和集群服务器进行互联，也可以连接外部网络(比如 WAN、VPN、互联网)。设计 InfiniBand 的目的主要是用于企业数据中心，大型的或小型的。目标主要是实现高的可靠性、可用性、可扩展性和高的性能。InfiniBand 可以在相对短的距离内提供高带宽、低延迟的传输，而且在单个或多个互联网络中支持冗余的 I/O 通道，因此能保持数据中心在局部故障时仍能运转。



InfiniBand 组成单元主要分为四类：

- HCA(Host Channel Adapter): 连接内存控制器和 TCA 的桥梁
- TCA(Target Channel Adapter): 将 I/O 设备 (例如网卡、SCSI 控制器) 的数字信号打包发送给 HCA
- InfiniBand link: 连接 HCA 和 TCA 的光纤, InfiniBand 架构允许硬件厂家以 1 条、4 条、12 条光纤 3 种方式连结 TCA 和 HCA
- 交换机和路由器: 无论是 HCA 还是 TCA, 其实质都是一个主机适配器, 它是一个具备一定保护功能的可编程 DMA(Direct Memory Access, 直接内存存取) 引擎。

IB 工作模式共有 7 种, 分别为:
(1)SRD(Single Data Rate): 单倍数据率, 即 8Gb/s; (2) DDR (Double Data Rate): 双倍数据率, 即 16Gb/s;
(3)QDR(Quad Data Rate): 四倍数据率, 即 32Gb/s; (4)FDR (Fourteen Data Rate): 十四倍数据率, 56Gb/s;
(5)EDR(Enhanced Data Rate): 100 Gb/s; (6)HDR(High Data Rate): 200 Gb/s; (7)NDR(Next Data Rate): 1000 Gb/s+。

应用场景

在高并发和高性能计算应用场景中, 当客户对带宽和时延都有较高的要求时, 可以采用 IB 组网: 前端和后端网络均采用 IB 组网, 或前端网络采用 10Gb 以太网, 后端网络采用 IB。由于 IB 具有高带宽、低延时、高可靠以及满足集群无限扩展能力的特点, 并采用 RDMA 技术和专用协议卸载引擎, 所以能为存储客户提供足够的带宽和更低的响应时延。

MPI 规范的标准化工作是由 MPI 论坛完成的，其已经成为并行程序设计事实上的工业标准。最新的规范是 MPI3.0，基于 MPI 规范的实现软件包括 MPICH 和 OpenMPI。MPICH 由美国阿贡国家实验室和密西西比州立大学联合开发，具有很好的可移植性。MVAPICH2、Intel MPI、Platform MPI 都是基于 MPICH 开发的。OpenMPI 由多家高校、研究机构、公司共同维护的开源 MPI 实现。

Eager 协议

该模式下发送进程将主动发送信息到接收进程，而不会考虑接受进程是否有能力接受信息。这就要求接受进程预先准备足够的缓存空间来接受发送过来的信息。Eager 协议只有非常小的启动负载，非常适合对延迟要求高的小消息发送。Eager 协议下，可以采用 'InfiniBand Send/Recv' 或 RDMA 方式发送消息来实现最佳性能。

Rendezvous 协议

与 Eager 模式相反，该模式下 Rendezvous 协议会在接收端协调缓存来接受信息。通常适用于发送比较大的消息。该情况下，发送进程自己不能确认接收进程能够有足够的缓存来接受要发送的信息，必须要借助协议和接收端协调缓存之后才会发送信息。

- 在 8 个服务器节点时，InfiniBand 能够提供双倍于以太网的性能；16 个节点时，基于 InfiniBand 的 NAMD 性能是以太网性能的 1.5 倍。



基准测试 (Benchmark)

利用测试程序对系统进行整体计算能力进行评价

- Linapck 测试：采用主元高斯消去法求解双精度稠密线性代数方程组，单位为 flops。
 - HPL：针对大规模并行计算系统的测试，其名称为 High Performance Linpack(HPL)，是第一个标准的公开版本并行 Linpack 测试软件包。用于 TOP500 与国内 TOP100 排名依据。



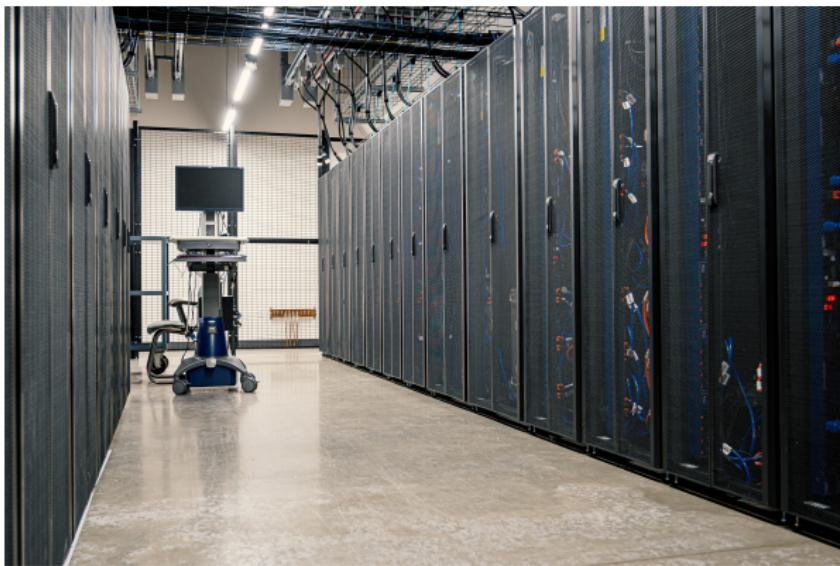
超级计算机榜单与 Top 500

Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	Supercomputer Fugaku - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442,010.0	537,212.0	29,899
2	Summit - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148,600.0	200,794.9	10,096
3	Sierra - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94,640.0	125,712.0	7,438
4	Sunway TaihuLight - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway, NRCPC National Supercomputing Center in Wuxi China	10,649,600	93,014.6	125,435.9	15,371
5	Perlmutter - HPE Cray EX235n, AMD EPYC 7763 64C 2.45GHz, NVIDIA A100 SXM4 40 GB, Slingshot-10, HPE DOE/SC/LBNL/NERSC United States	761,856	70,870.0	93,750.0	2,589



分布式计算、网格计算与云计算

- 分布式计算可以追溯到来自大型数据库服务器，如航空公司的预订系统，它必须被许多旅行社同时访问。对于足够大的数据库访问量，单台服务器是不够的，因此发明了远程过程调用 (remote procedure call) 的机制，中央服务器将调用不同 (远程) 机器上的代码 (有关的过程)。远程调用可能涉及数据的传输，数据可能已经在远程机器上，或者有一些机制使两台机器的数据保持同步。这就产生了存储区域网络 (Storage Area Network, SAN)。比分布式数据库系统晚了一代，网络服务器不得不处理同样的问题，即许多人同时访问必须表现得像一个单一的服务器。
- 可以在远程服务器和电网之间做一个类比，前者在需要的地方提供计算能力，后者在需要的地方提供电力。这导致了网格计算或实用计算的出现，美国国家科学基金会拥有的 Teragrid 就是一个例子。网格计算最初是作为一种连接计算机的方式，通过局域网 (Local Area Network, LAN) 或广域网 (Wide Area Network, WAN)，通常是互联网连接起来。这些机器本身可以是平行的，而且通常由不同的机构拥有。最近，它被视为一种通过网络共享资源的方式，包括数据集、软件资源和科学仪器。



应用场景

按需提供服务的模式对企业很有吸引力，这些企业越来越多地使用云服务。它的优点是不需要最初的货币和时间投资，也不需要对设备的类型和大小做出决定。目前，云服务主要集中在数据库和办公应用上，云资源使用场景的大致有：

- 扩展：在这里，云资源被用作一个平台，可以根据用户需求进行扩展。分为平台即服务 (PaS)、基础设施即服务 (IaaS)、软件即服务 (SaaS)。
- 批量处理：用户有大量的数据需要以批处理模式进行处理，这种模式是 MapReduce 计算的良好候选者。
- 存储：大多数云供应商都提供数据库服务，让用户不需要维护自己的数据库。
- 同步化：这种模式在商业用户应用中很受欢迎。Netflix 和亚马逊的 Kindle 允许用户消费在线内容 (分别是流媒体电影和电子书)；暂停内容后，他们可以从任何其他平台恢复。苹果公司最近的 iCloud 为办公应用中的数据提供了同步，但与 Google Docs 不同的是，这些应用不是“在云中”，而是在用户机器上。

人工智能领域

- 人工智能：人工智能是一门基于计算机科学，生物学，心理学，神经科学，数学和哲学等学科的科学和技术。人工智能是一门科学，这门科学让机器做人类需要智能才能完成的事。
- 机器学习：一种实现人工智能的方法。机器学习最基本的做法，是使用算法来解析数据、从中学，然后对真实世界中的事件做出决策和预测。机器学习是用大量的数据来“训练”，通过各种算法从数据中学习如何完成任务。
- 深度学习：一种实现机器学习的技术。它将现实世界表示为嵌套的层次概念体系（由较简单概念间的联系定义复杂概念，从一般抽象概括到高级抽象表示），从而获得强大的性能与灵活性。

声音识别 $f(\text{[声波图]}) = \text{"How are you"}$

图像识别 $f(\text{[猫的照片]}) = \text{"Cat"}$

人工智能 $f(\text{[手写数字]}) = \text{"5-5"}$

场景智能 $f(\text{["Hi"]}_{\text{(what the user said)}}) = \text{["Hello"]}_{\text{(system response)}}$

常用深度学习框架：

- ① Tensorflow
- ② MXNet
- ③ Caffe
- ④ Pytorch

图：深度学习应用场景

- ① Python: Python 是一种计算机程序设计语言。是一种面向对象的动态类型语言，最初被设计用于编写自动化脚本 (shell)，随着版本的不断更新和语言新功能的添加，越来越多被用于独立的、大型项目的开发。
- ② TensorFlow: Tensorflow 是 google 开发的一个深度学习框架软件，同时它也是 python 的一个库，支持 python 语言开发相应程序。
- ③ MXNet: MXNet 是一个十分优秀的深度学习框架。目前包含了许多语言接口，如 Python、C++、Scala、R 等。目前，MXNet 版本已经更新到 1.3.0。浪潮的 AIStation 仅使用 Python 接口，MXNet 也是 python 的一个库。
- ④ Caffe: Caffe 是一种常用的深度学习框架，主要应用在视频、图像处理方面的应用上。Caffe 纯粹的 C++/CUDA 架构，支持命令行、Python 和 MATLAB 接口；可以在 CPU 和 GPU 直接无缝切换。
- ⑤ PyTorch: 它是由 Torch7 团队开发，是一个以 Python 优先的深度学习框架，不仅能够实现强大的 GPU 加速，同时还支持动态神经网络，这是很多主流深度学习框架比如 Tensorflow 等都不支持的。
- ⑥ 飞桨 (PaddlePaddle) 以百度多年的深度学习技术研究和业务应用为基础，集深度学习核心框架、基础模型库、端到端开发套件、工具组件和服务平台于一体，2016 年正式开源，是全面开源开放、技术领先、功能完备的产业级深度学习平台。飞桨源于产业实践，始终致力于与产业深入融合。目前飞桨已广泛应用于工业、农业、服务业等，服务 150 多万开发者，与合作伙伴一起帮助越来越多的行业完成 AI 赋能。

Docker 技术

Docker 是基于容器技术的轻量级虚拟化解决方案。其特点为：秒级启动，秒级停止；空间资源占用极少；可以实现进程级别的隔离且易于迁移。基本元素包含：镜像，容器，仓库，数据卷

- 镜像 image：镜像用于创建容器，一个镜像中包含一个操作系统及需要安装的应用程序。image 相当于 container 的模板，container 初始创建后里面有什软件完全取决于它使用什么 image。一个镜像可以创建出多个运行的虚拟主机且相互独立。
- 容器 container：可以把每个 container 看做是一个独立的虚拟主机，container 的创建通常有一个 image 作为其模板。类比成虚拟机的话可以理解为 image 就是虚拟机的镜像，而 container 就是一个正在运行的虚拟机。可以把容器看成是一个简易版的操作系统和运行在其中的应用程序
- 仓库 registry：存放镜像的仓库，仓库分为公开仓库和私有仓库，Push 镜像到仓库，从仓库 pull 下镜像。通过网络连接到私有仓库 registry 后使用 pull 命令从仓库中获取镜像。本集群中在 gpu01 节点搭建本地镜像仓库。
- 数据卷 volumes：Docker 管理宿主机文件系统的一部分，默认位于 /var/lib/docker/volumes 目录中

- ① Victor Eijkhout, Introduction to High Performance Scientific Computing. October 5, 2021, 2020
- ② 杨丽凤, 曹锐, 王娜等. 大学计算机基础与计算思维.
- ③ Kai Hwang,Zhiwei Xu,"Scalable Parallel Computing",McGraw- Hill,1998
- ④ J.JaJa,"Introduction to Parallel Algorithms",Addison Wesley,1992
- ⑤ V.Kumar etal, “Intro to Parallel Computing”, Benjamin/Cummings, 1994
- ⑥ 陈国良, 并行算法的设计与分析 (第 3 版), 高等教育出版社, 2009
- ⑦ 陈国良等, 并行计算机体系结构, 高等教育出版社, 2002
- ⑧ 陈国良等, 并行算法实践, 高等教育出版社, 2003
- ⑨ J.Dongarra etal, “Sourcebook of Parallel Computing”(莫则尧等译), 电子工业出版社, 2005
- ⑩ Shameem Akhter, et. al. 著, 李宝峰等译. 多核程序设计技术, 电子工业出版社, 2007
- ⑪ Richard Gerber, et. al. 著, 王涛等译. 软件优化技术, 电子工业出版社, 2007
- ⑫ 超算竞赛导引编写组, 超算竞赛导引, 科学出版社, 2016