超级计算机结构综述

学号： B1910Z0363 姓名：唐大海

1 并行计算

1.1 并行计算模型

1.1.1 PRAM 模型

PRAM（Parallel Random Access Machine）模型，即并行随机存取机，也称之为共享存储器的 SIMD 模型，是一种抽象的并行计算模型。在这种模型中，假定存在着一个容量无限大的共享存储器；有有限或无限个功能相同的处理器，且其均具有简单的算术运算和逻辑判断功能；在任何时刻各处理器均可通过共享存储单元相互交换数据[1][2][3]。

优点： 特别适合并行算法的表达、分析和比较；使用简单；易于设计算法和稍加修改便可运行在不同的并行机上[1]。

缺点：需要指令按照锁步方式操作，特别费时。

1.1.2 APRAM 模型

APRAM（Asynchronous Parallel Random Access Machine）模型，即异步随机存取机，他是在 PRAM 模型上面修改而来的。其特点是每个处理器都有其本地存储器、局部时钟和局部程序；处理器之间的通信通过共享全局存储器；处理器任何时间依赖关系需要明确地在各处理器的程序中加入同步路障；一条命令在非确定但有限的时间内完成[4][5]。

优点：更接近实际的并行机，由于每个处理器有自己的本地存储，一定程度上缓和了数据访问的冲突，保留了 PRAM 模型的优点。

缺点：存在同步路障，且数据在共享存储方面有一部分的冲突，数据集中的部分对存储器带宽要求高。

1.1.3 BSP 模型

BSP（Bulk Synchronous Parallel）模型，即大同步模型。BSP 模型具有三个参数： 1. 处理器/存储器模块； 2. 施行处理器/存储器模块对之间点到点传递信息的选路器； 3. 执行以时间间隔 L 为周期的路障同步器。BSP 模型以三个参数：处理器数量（P），选路器吞吐率（带宽因子，g），全局同步时间间隔（L）来抽象计算时间， 是一个

分布存储的 MIMD 计算模型，具有：处理器与选路器分开，实现了硬件路障方式的全局同步等特点[6]。

1.1.4 LogP 模型

LogP 模型是一种分布存储的，点到点通信的多处理器机模型，其中通信网络由一组参数来描述。LogP 代表： 1) L（Latency），网络中消息从源到目的地所遭到的延迟； 2) o（Overhead）， 处理器发送或接收一条消息所需的额外开销； 3) g (Gap)，处理器可连续进行消息发送或者接收的最小时间间隔； 4) P （Processor）， 处理器模块数量。LogP 模型充分揭示了分布存储并行机的性能瓶颈，用 L、o 和 g 三个参数刻画了通信网络的特征[7][8]。

1.2 并行程序设计模型

1.2.1 数据并行模型

数据并行程序设计强调局部计算和数据选路操作，比较适合使用规则网络、模板和多维信号及图像数据集来求解细粒度的应用问题。数据并行操作是在编译时而不是在运行时完成的。数据并行模型具有：单线程，并行操作于聚合数据结构上，松散同步，全局命名空间隐式数据分配等特点[9]。

1.2.2 消息传递模型

在消息传递模型中，驻留在不同节点的进程可以通过网络传递消息相互通信，消息可以是指令、数据、同步信号或中断信号等。在消息传递并行程序中，用户必须明确地为进程分配数据和负载。消息传递模型具有：多线程，异步并行性，显式分配等特点[10]。

1.2.3 共享变量模型

共享变量模型中的各个处理器上的进程可以通过读写公共存储器中的共享变量相互通信，具有一个单一的全局内存地址空间[11][12]。

2 向量机

向量运算是一种较简单的并行计算,适用面很广,机器实现比较容易,使用也比较方便,因此向量计算机（向量机）获得了迅速发展。TIASC（1972 年）和 CDCSTAR－100 （1973 年）是世界上第一批向量巨型计算机（巨型机）。到 1982 年底,世界上约有 60 台巨型机,其中大多数是向量机，中国于 1983 年研制成功的每秒千万次的 757 机和亿次的“银河”

机也都是向量机。
向量机适用于线性规划、傅里叶变换、滤波计算以及矩阵、线代数、偏微分方程、积分等数学问题的求解，主要解决气象研究与天气预报、航空航天飞行器设计、原子能与核反应研究、地球物理研究、地震分析、大型工程设计，以及社会和经济现象大规模模拟等领域的大型计算问题。
向量计算机以向量作为基本操作单位，操作数和结果都以向量的形式存在，包括纵向加工向量机和纵横加工向量机[13]。
向量一般配有向量汇编和向量高级语言，供用户编制能发挥向量机速度潜力的向量程序。只有研制和采用向量型并行算法，使程序中包含的向量运算越多、向量越长，运算速度才会越高。面向各种应用领域的向量的建立，能方便用户使用和提高向量机的解题效率。
典型的向量计算机有： Cray-1, Cray-2 等[14]。

3 同构计算机集群

集群系统以其良好的可扩展性和高性价比 ， 已成为高性能计算领域的主流体系结构 . 同构集群系统是一种主要的集群系统 ， 在同构集群系统中 ， 每一个计算节点的硬件配置和软件配置均相同。
集群计算机按功能和结构可以分成以下几类：
高可用性集群 High-availability (HA) clusters
负载均衡集群 Load balancing clusters
高性能计算集群 High-performance (HPC) clusters
网格计算 Grid computing

高可用性集群
一般是指当集群中有某个节点失效的情况下，其上的任务会自动转移到其他正常的节点上。还指可以将集群中的某节点进行离线维护再上线，该过程并不影响整个集群的运行。

负载均衡集群
负载均衡集群运行时一般通过一个或者多个前端负载均衡器将工作负载分发到后端的一组服务器上，从而达到整个系统的高性能和高可用性。这样的计算机集群有时也被称为服务器群（Server Farm）。 一般高可用性集群和负载均衡集群会使用类似的技术，或同时具有高可用性与负载均衡的特点。
Linux 虚拟服务器（LVS）项目在 Linux 操作系统上提供了最常用的负载均衡软件。

高性能计算集群
高性能计算集群采用将计算任务分配到集群的不同计算节点儿提高计算能力，因而主要应用在科学计算领域。比较流行的 HPC 采用 Linux 操作系统和其它一些免费软件来完成并行运算。这一集群配置通常被称为 Beowulf 集群。这类集群通常运行特定的程序以发挥 HPC cluster 的并行能力。这类程序一般应用特定的运行库，比如专为科学计算设计的 MPI 库。
HPC 集群特别适合于在计算中各计算节点之间发生大量数据通讯的计算作业，比如一个节点的中间结果或影响到其它节点计算结果的情况。

网格计算
网格计算或网格集群是一种与集群计算非常相关的技术。网格与传统集群的主要差别是网格是连接一组相关并不信任的计算机，它的运作更像一个计算公共设施而不是一个独立的计算机。还有，网格通常比集群支持更多不同类型的计算机集合[15]。
网格计算是针对有许多独立作业的工作任务作优化，在计算过程中作业间无需共享数据。网格主要服务于管理在独立执行工作的计算机间的作业分配。资源如存储可以被所有结点共享，但作业的中间结果不会影响在其他网格结点上作业的进展。
典型的同构超算有： 神威蓝光

4 异构计算机集群

异构计算（Heterogeneous computing）主要是指使用不同类型指令集和体系架构的计算单元组成系统的计算方式。常见的计算单元类别包括 CPU、GPU 等协处理器、DSP、ASIC、FPGA 等。

异构计算近年来得到更多关注，主要是因为通过提升 CPU 时钟频率和内核数量而提高计算能力的传统方式遇到了散热和能耗瓶颈。而与此同时，GPU 等专用计算单元虽然工作频率较低，具有更多的内核数和并行计算能力，总体性能-芯片面积比和性能-功耗比都很高，却远远没有得到充分利用[16]。

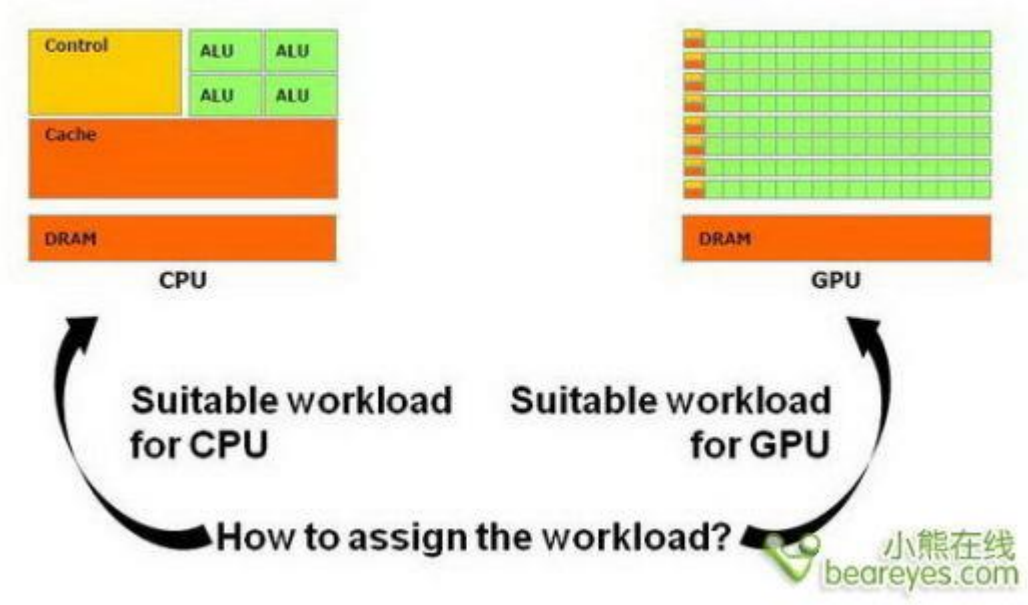
广义上，不同计算平台的各个层次上都存在异构现象，除硬件层的指令集、互联方式、内存层次之外，软件层中应用二进制接口、API、语言特性底层实现等的不同，对于上层应用和服务而言，都是异构的[17]。

从实现的角度来说，异构计算就是制定出一系列的软件与硬件的标准，让不同类型的计算设备能够共享计算的过程和结果。同时不断优化和加速计算的过程，使其具备更高的计算效能。

4.1 单节点跨 PCIe 总线

单节点跨 PICe 总线的异构计算平台的形式主要是主板上安装多块 CPU，CPU 作为主要的运行操作系统和进行调度的平台，而各种加速器件，如 GPU、Phi 或者 FPGA 则以 PCIe 插卡的形式对计算量大的任务进行加速计算。
这种方式极大地利用了各种加速卡的运算性能，使得计算任务能够被快速执行。但存在的问题就是 PCIe 的带宽限制和运算过程中的数据搬移有比较大的延迟。
单节点跨 PCIe 总线的异构计算平台根据加速卡的不同，又分为：CPU + GPU 异构计算平台，CPU + Phi 异构计算平台，CPU+FPGA 异构计算平台。

4.1.1 CPU + GPU 异构平台



在 CPU+GPU 异构计算平台中，GPU 作为加速卡对 CPU 分配的任务进行加速计算。当前使用的 GPU 加速卡厂商主要有 NVIDIA 和 AMD 两家公司的产品。
典型的 CPU+GPU 异构计算平台有： 中国的天河 1 超算，美国的 Titan 超算[18]等。

4.1.2 CPU + Xeon Phi 异构平台

至强融核（Xeon Phi）协处理器，是首款英特尔集成众核（Many Integrated Core，MIC）架构产品。用作高性能计算（HPC）的超级计算机或服务器的加速卡。最多 72 个处理器核心，每个核心拥有 4 个超线程，最多 288 个线程，超线程无法关闭。英特尔至强融核协处理器提供了类似于英特尔至强处理器编程环境的通用编程环境。多个英特尔至强融核协处理器可安装在单个主机系统中，这些协处理器可通过 PCIe 对等互连相互通信，不受主机的任何干扰。
典型的 CPU+Xeon Phi 异构计算平台有： 中国的天河 2 超算。

4.1.3 CPU + FPGA 异构平台

FPGA 由于其便于硬件编程的性质，近些年来被数据中心和云计算厂商所青睐。越来越多的图形识别、人工智能算法被证实在 FPGA 上取得了很大的加速效果，其能耗比相对于 GPU 异构平台也有很大的提升。目前在超算上面是否使用 FPGA 加速卡，也有相当一部分人进行了研究。相信在不久的将来，CPU+FPGA 异构平台能够大放异彩。

4.2 众核计算机

众核计算机在一个 CPU 中，采用了主核与协作小核的设计。对外看起来只是一个处理器，但是处理器里面各个分类的核心所作的事情都不一样。例如在神威系列的处理器里面，就采用了这种众核的设计，每个处理器芯片中有 260 个核心，采用大规模多核心并发运算的结构，其中 4 个为资源管理用途，称为 MPE（Management Processing Element，管理处理组件），采用对称多处理器的结构；另外 256 个作通用运算用途，每 64 个核心组成一个处理器核心阵列，共计 4 个阵列，合称为 CPE（Computing Processing Element，运算处理组件）。MPE 和 CPE 的连接布局类似于 Cell 的协处理器式、非对称多处理的布局（PPE+SPE），而 CPE 的阵列则与 Xeon Phi、GPGPU 等的流处理器形式相近[19]。
典型的众核计算机有：中国的神威太湖之光，以及基于 Intel 公司后期发行的 MIC 众核芯片设计的计算机。

4.3 其它

最新的登顶的美国超级计算机顶点（Summit），采用的是 CPU+GPU 的异构加速形式。但是 GPU 的通信，采用了 NVIDIA 公司自己研发的 NVLink 技术[20]，在 GPU 与 GPU 之间相互通过 NVLink 传输数据，有效地降低了 GPU 上的数据需要通过 PCIe 总线到高速节点间网络进行传输的时延消耗和带宽消耗。

[1] 陈国良，吴俊敏等. 并行计算机体系结构, 高等教育出版社, 2002.9, 43-55
[2] Fich F E. The complexity of computation on the parallel random access machine[M]. Department of Computer Science, University of Toronto, 1993.
[3] Stockmeyer L, Vishkin U. Simulation of parallel random access machines by circuits[J]. SIAM Journal on Computing, 1984, 13(2): 409-422.
[4] Savage J E. Models of computation[M]. Reading, MA: Addison-Wesley, 1998.
[5] Gibbons P B, Matias Y, Ramachandran V. The Queue-Read Queue-Write PRAM model: Accounting for contention in parallel algorithms[J]. SIAM Journal on Computing, 1997: 638-648.
[6] Tiskin A. The bulk-synchronous parallel random access machine[J]. Theoretical computer science, 1998, 196(1-2): 109-130.
[7] Culler D, Karp R, Patterson D, et al. LogP: Towards a realistic model of parallel computation[C]//ACM Sigplan Notices. ACM, 1993, 28(7): 1-12.
[8] Alexandrov A, Ionescu M F, Schauser K E, et al. LogGP: Incorporating Long Messages into the LogP Model---One step closer towards a realistic model for parallel computation[J]. 1995.
[9] Gordon M I, Thies W, Amarasinghe S. Exploiting coarse-grained task, data, and pipeline parallelism in stream programs[J]. ACM SIGARCH Computer Architecture News, 2006, 34(5): 151-162.
[10] Braverman M, Ellen F, Oshman R, et al. A tight bound for set disjointness in the message-passing model[C]//2013 IEEE 54th Annual Symposium on Foundations of Computer Science. IEEE, 2013: 668-677.
[11] LeBlanc T J. Shared Memory Versus Message-Passing in a Tightly-Coupled Multiprocessor: A Case Study[C]//Proceedings of the International Conference on Parallel Processing. IEEE, 1986: 463-466.
[12] Adve S V, Gharachorloo K. Shared memory consistency models: A tutorial[J]. computer, 1996, 29(12): 66-76.
[13] Demjanenko V. Vector processor architecture and methods performed therein: U.S. Patent Application 10/467,225[P]. 2004-4-15.
[14] Bailey D H. Extra high speed matrix multiplication on the Cray-2[J]. SIAM Journal on Scientific and Statistical Computing, 1988, 9(3): 603-607.
[15] Mavriplis D J, Pirzadeh S. Large-scale parallel unstructured mesh computations for three-dimensional high-lift analysis[J]. Journal of aircraft, 1999, 36(6): 987-998.
[16] Liu H T, Silvester J A. Dynamic resource allocation scheme for distributed heterogeneous computer systems: U.S. Patent 5,031,089[P]. 1991-7-9.
[17] Jain R. A delay-based approach for congestion avoidance in interconnected heterogeneous computer networks[J]. ACM SIGCOMM Computer Communication Review, 1989, 19(5): 56-71.
[18] Mondragon O H, Bridges P G, Levy S, et al. Understanding performance interference in next-generation HPC systems[C]//SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE, 2016: 384-395.
[19] 洪文杰, 李肯立, 全哲, 等. 面向神威· 太湖之光的 PETSc 可扩展异构并行算法及其性能优化[J]. 计算机学报, 2017, 40(9): 2057-2069.

[20] Kahle J A, Moreno J, Dreps D. 2.1 Summit and Sierra: Designing AI/HPC Supercomputers[C]//2019 IEEE International Solid-State Circuits Conference-(ISSCC). IEEE, 2019: 42-43.