并行计算二十年

伍民友 上海交通大学计算机科学与工程系 上海 200030

1. 引言

在本文中我们简要回顾并行计算二十年的发展历程，而重点放在早年的发展。我们也讨论并行程序设计的发展潮流。高性能计算在尝试过所有可能性后，越来越清晰地分成两个主要潮流：超级计算和集群计算。人们等待了二十年，最终认识到这两种高性能计算都需要存在而且各得其所。回顾并行计算的发展历程尤其早年的历史，我们会发现得到这样的结论其实并不简单。

并行计算机的历史可以追溯到上个世纪五十年代中叶的IBM704。1985年以前已有建立并行计算机的尝试。Cray计算机早年的版本已经实现了流水线向量机，并为并行计算机奠定了基础。其他并行计算机，例如数据流机（dataflow）和处理器阵列机(systolic array)已经被提出并广泛研究。但是这些方案由于种种原因都不很成功，例如很差的灵活性、较高的代价和较低的效率。

1985年是并行计算发展史上的一个里程碑，因为这一年里涌现出一大批真正意义上的并行计算机。值得注意的是Hypercube和Thinking Machine。Hypercube是一种多指令多数据流(MIMD)机器而Thinking Machine制造单指令多数据流(SIMD)机器。从此开始，今天的并行计算机种类可分为大规模并行处理机（MPP）、共享存储器多向量处理机（SMP）和分布式共享存储机（DSM），这些都来自于可升级计算机潮流时政府支持的项目。不幸的是，这些探索带来体系结构的多样性，导致了标准平台和编程模块不兼容。每一个平台都有较低的容量、巨大的软件研发成本，最后无人问津。

本文追溯至1985年，审视今天的计算机是如何由早期的并行计算机演变而来，伴随着高性能计算机体系结构研究轰轰烈烈的开始和消亡。我们也总结二十年来并行编程设计的主要发展。这里并不是对并行计算历史作全面综合的回顾，仅是在某些重要发展领域海滩拾贝、抛砖引玉。

2. 并行计算机简史

我们从1985年起回顾并行计算机的发展历程。但是此前的某些进展催生了1985年真正意义并行计算机的诞生。早在1981年的加州理工学院，一个由Charles Seitz和Geoffrey Fox领导的团队开始研究超立方多计算机系统。1984年超级计算研究发展中心（CSRD）在伊利诺斯大学成立，但其分级共享存储计算机Cedar的研究工作从1982年就开始了。1984年耶鲁大学Josh Fisher团队创办的Multiflow研制成功了超长指令字（VLIW）超级计算机。

伊利诺斯大学CSRD中心由David Kuck创建并在1985年展示了Cedar超级计算机，它是全世界第一台大规模共享存储器计算机[GKLS83]。不过CSRD中心最有影响力的一个工作是Daniel D. Gajski在[GaPe85]阐述的并行编程原理，即并行编程分为三个阶段：（1）分割；（2）调度；（3）同步。一个程序要并行化必须首先分割成多个任务，然后分配到多个处理器，最后需要任务之间同步以保证正确的执行顺序。

由Charles Seitz和Geoffrey Fox共同研发的超立方计算机(Hypercube)是一种基于静态网络的分布式存储器并行计算机。超立方拓扑为处理器之间的通信提供了丰富的连接，这种体系结构后来被Intel超级计算机公司商品化。1985年Intel推出了第一款iPSC/1超立方计算机，后来演变为iPSC/2。iPSC/1机器包括128个80286处理器，并通过Ethernet控制器互联。这是第一款商业计算机允许科研人员或工程师在实际的机器上编制应用程序。后来NCUBE制造的nCUBE超立方计算机使用的是定制的类-VAX处理器。

超立方计算机代表了一类基于静态网络的分布式存储器机器，其中网络拓扑扮演了重要的角色。尽管超立方拓扑为处理器之间通信提供了丰富连接，它依然不能被看作可扩展机器，因为节点的度（一个节点与相邻节点的连结数）随着节点总数的增大而增加。后来Mesh和三维torus拓扑越来越流行，另一个可供选择的是胖树（fat tree）拓扑，它也提供了有效的内部互联。后来在1990年左右由于虫孔路由与高性能网络的实现，一些研究人员坚持认为拓扑并非那么重要。但是从近期高性能计算机的发展来看，拓扑对那些下面将要讨论的紧耦合问题依然重要。

同样是在1985年，另一台并行计算机Connection

Machine问世了。这台机器是Denny Hillis创建的Thinking Machines 公司制造的 [Hill85]。CM-2和CM-5分别于1987、1991年问世。Connection Machine是一种可拥有64k个处理器的单指令多数据流机器，网络拓扑是fat tree。次年，Maspar也推出了它的SIMD多处理器机器。SIMD体系结构参见图1，描绘了控制单元取指、解码指令并广播给处理器单元（PEs），其间布置了广播网络和常规网络。

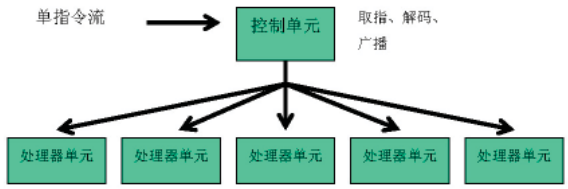


图1. SIMD 体系结构

SIMD机器的应用后来被证明是有限的。因此这并不是今天并行计算机的选项。但是Connection Machine给我们留下了可贵的经验。首先是连接网络。连接网络有两种：广播网络和常规网络。前者最初应用于单指令流的广播，亦可用于广播数据；后者是一种通用网络体系，为并行计算机提供了高性能和很好的灵活性。其次是编程语言里的FORALL结构。正如后面将要提到的，这种结构被高性能Fortran（HPF）继承。

1993年IBM推出了第一款基于其RISC RS/6000处理器的SP1系统。不同于前面的并行计算机，SP1使用动态网络作为内部连接设施。实际上有两种动态网络存在：Crossbar和Omega网络。Crossbar开关具有高灵活性、高性能和高价格，而Omega网络要廉价一些。几乎所有的日本并行计算机使用Crossbar而美国采用Omega网络或者其他多级开关网络。

绝大多数分布式存储器计算机需要人工编写并行应用程序。这种编程风格演变成今天的MPI编程。面向自动并行代码生成的一个重要发展是HPF语言。1990年由Syracuse和Rice大学Ken Kennedy和Geoffrey Fox领导的并行计算研究中心（CRPC）开始了一项研究工作，旨在利用Fortran为分布式计算机自动生成代码。Fortran D语言[FHKK+90]，即后来的高性能Fortran （HPF），允许用户利用FORALL结构描述并行化算法。它使用户可以在共享存储器环境下编写并行程序再自动编译为分布式存储器机器下的并行程序。次年第一款HPF编译器问世并应用于其后的大多数并行计算机[BCFH+94]。HPF的缺点在于很难编译复杂的下标结构。

使得并行编程简单化的一项独立工作是由李凯（Kai Li）发明的分布式共享存储器技术[Li88]。早在1986年还在耶鲁大学时李凯提出一种新的体系结构，允许用户编写分布式存储器机器的共享存储器程序。这一方法引起了90年代被称为可升级共享存储器的热潮，并最终导致了一大批商业化产品的诞生，如Cray T3D、T3E和SGI Origin 2000等。它的缺点是高效率必须以良好的程序局部性（locality）为前提。

经过并行计算领域科学家和工程师多年的研究工作，人们发现并非所有实际应用都需要复杂的并行体系结构和低延迟网络连接。事实上，绝大多数应用都是松散关联的，亦即它们能被分割成许多关系松散的部分。类似的应用包括药物发现、模拟，以及视频编解码和可视化等。实际上，如果每一部分可以运行几秒以上而无需通信，延迟就不是问题。因此，个人电脑甚至工作站网络（NoW）都能满足用户需求并获得高性能（因而称为高输出）。这一发现导致另一种超级计算的诞生：集群计算。

一种低成本、高性能的集群计算机Beowulfs逐渐商品化，成为科研工作者、工程师和商业应用的主流平台。Beowulf [Ster99]最终成为集群计算的一个标准化建议。这使得用户有了统一的平台和编程模型，不需要专用的处理器和网络，又有共同的操作系统和工具。这些优点使得高中生都有可能为超级计算的应用建造自己的计算机。建立并行计算机不再是高技术，当然我们此处指的是集群计算。

集群计算的一种简单形式是使用千兆以太网联接工作站或个人电脑。但是为了获得高性能和低延迟，高端用户可以使用Myrinet或InfiniBand集群体系结构。作为一种商品化的现货供应技术（COTS），集群计算能够如此简单的配置以至于人们在反问为什么我们需要多处理机。之所以我们依然需要这些专用的超级计算机，是因为存在某些紧耦合的应用，比如基因计算。集群计算适合于松耦合而不是紧耦合问题。其中的原因不在于有限的带宽，而是启动时间。当问题的各个任务之间频繁地交换数据时，更需要专用的超级计算机。正因为如此，在集群计算的时代，少数几家供应商（例如Compaq, Cray, IBM, Intel, SGI和Sun）依然生产专用的大规模并行计算机。

从二十年来对并行计算机结构的探索中，我们现在有了若干强有力的超级计算机，包括IBM蓝色基因（Blue Gene）和SGI Altix（NASA哥伦比亚）。IBM蓝色基因是世界上最快的机器，参见图2。它拥有0.7GHZ的处理器，和双重互联网络：3D torus和fat tree。此外，还有存取用的千兆以太网。安装有128k个处理器的Blue Gene/L安装在美国能源部（DOE ）Lawrence Livermore国家实

验室（LLNL），已经创造了Linpack 280.6 TFlop/s的世界纪录。这是仅有的一个超过100TFlop/s的系统。由SGI制造的安装在NASA/Ames Columbia系统达到51.87TFlop/s的性能，参见图3。这个SGI Altix机器有1万个1.5GHz的处理器，512个CPU集群共享处理器，其互联网络是SGI NUMAlink, Infiband 和10G的以太网。

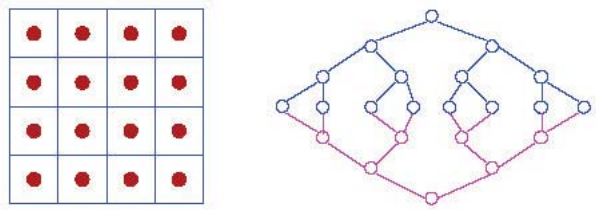
图2. IBM 蓝色基因

图3.美国国家航空航天局 哥伦比亚

探索并行计算新方法的二十年的努力究竟给我们留下了什么？所剩无几。所遗留的包括分布式存储计算机、可升级共享存储器系统、集群计算、静态互联网络、3D torus和fat tree、MPI和FORALL程序结构。集群计算体系结构潮流是Myrinet、InfiniBand或者千兆以太网互联的处理器；超级计算机的潮流是底部的共享存储器结构和顶部的分布式存储器结构。网络潮流是3D torus和fat tree，fat tree用于广播和归约，而3D torus用于通用的数据交换。编程潮流在人们的各种尝试失败后渐渐被MPI主宰而回归原始的手工编程。尽管类HPF的编程风格有益处，但是很难编译，因此很难发挥长处。看起来关于并行编程并不存在较好的解决方案。但是二十年里我们已经积累了很多经验和有趣的结论，这些将为我们思考下一代并行编程提供帮助。我们在下一节将讨论并行编程的方法。

3. 并行编程方法简论

大规模并行度主要有两种方法产生：数据并行（data parallel）和多递归并行(multi-recusive parallel)，如图4所示



(a) (b)

图4. (a) 数据并行; (b) 多递归并行

数据并行可以用并行语言的DOALL循环或FORALL结构表达。FORALL结构是一种灵活又富于表达力的结构，例如：

```
FORALL (i=1:N, j=1:M)
    Body
END FORALL
```

另一种大规模并行是多递归并行，可以用Prolog、Sisal或者C++递归子程序表达。多递归子程序调用例如：

```
subprogram {
  if (terminate) return result;
  else {
     x = subprogram;
     y = subprogram;
     return (operation(x,y));
      }
}
```

问题结构可以分类为规则型和不规则型。不同的人对此有不同的定义。在此，我们给出一种最简单的定义：如果没有一种简单方法分割一个问题以达到负载均衡，那么它是一个不规则问题；否则就是一个规则问题。问题结构同样可以分类为静态和动态结构。如果问题的结构变化不定，则是动态问题；否则是静态问题。综合这些分类，我们把问题结构分为静态规则的、静态不规则的和动态不规则的三种。

静态规则问题易于并行化，此类问题并行化后均有较好的性能而且实际上绝大多数并行性能较好的问题均属此类问题。静态不规则问题难于并行化。使用并行语言和编译器的方法能够解决此类问题。第一步是使用并行语言结构表达并行算法并分配数据以求均衡负载和同时最小化通信量。第二步，用编译器为并行系统自动生成代码。动态不规则问题是最难并行化的，上述使用并行语言和编译器的方法均不能奏效，不可避免地需要实

时系统支持。

面向系统的方法是一种解决动态不规则问题的可选方案。实时系统能够支持一般的应用并对体系结构细节和动态状态有更多的知识。因此开发精致的系统解决方案比反复地编写简单的用户代码有效的多。然而要想解决动态不规则问题，有很多难题有待于解决。首先，我们需要明白如何分割程序才能高效地并行化；其次，我们需要均衡负载、最小化通信量并合理安排并行操作的时间。在此，我们举一个二级结构的例子。第一级是程序分割为进程；第二级进程分割为线程。在第一级中，进程可以是可迁移的或不可迁移的。迁移（migration）是在执行过程中从一个处理器移动到另一个处理器。如果进程足够小则可以不迁移，即留在处理器直至执行完毕。在第二级中，线程可以是可中断的或不可中断的，如果线程足够小的话就可执行到底而不中断。图5就是这样一个模型。这一u线程模型是一个多线程的、信息驱动的、无迁移不中断的模型 [Shu95]。因为没有迁移和中断，这一模型有效并且开销较低。u进程(uProcess）是一种超轻型进程，u线程（uThread）是一种超轻型线程。u线程通过共享数据区（SDA）交换数据，u进程通过信息交换数据。如图6，u进程可被分配到不同处理器以均衡负载和运算时间。一旦u进程开始执行，它将不能被移动到其它处理器。同样，一旦u线程开始执行，它将直至计算结束。

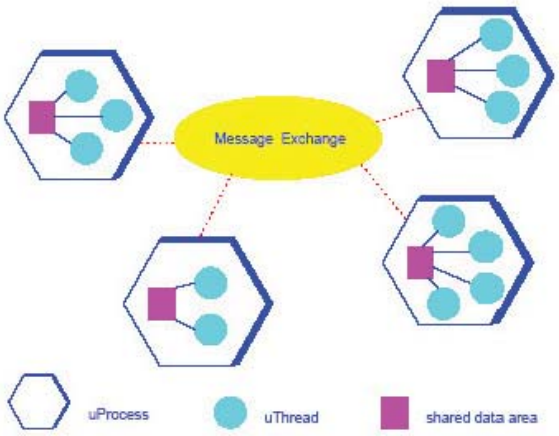


图 5. u进程、u线程和共享数据区SDA结构

最后，让我们总结一下不规则的应用问题的解决方法。第一步，用并行语言表达并行化，并用编译器分割程序为任务或者进程。这一步我们需要尽可能地保持并行度，同时最小化任务之间的通信量。第二步，负载调度和均衡。这一步我们需要均匀地分配负载，最大局部化，而且最小化通信量。此外，处理器空闲时间和开销也需要被最小化。调度可以是静态的、在运行之前执行，这适用于静态问题而且利用问题的特征知识达到全局最优；调度也可以是动态的，在运行过程中执行，这样能够基于负载信息动态地调节负载分配。但是负载信息收集与计算资源存在冲突关系。

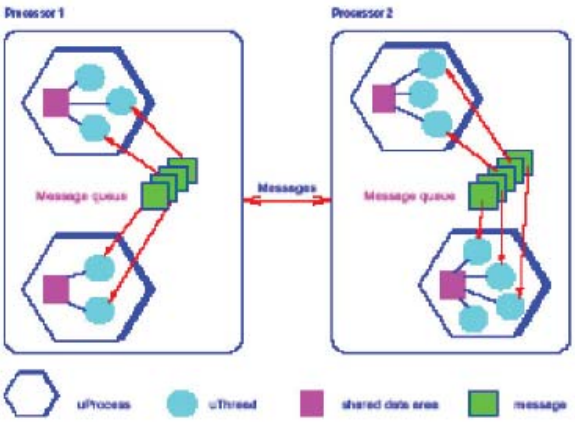


图6. 分布式存储器系统中的u线程

4. 总结

经过二十年并行体系结构与编程的研究，我们发现了很多问题的解决方案，却遗留了更多的问题有待于解决。在体系结构领域，研究汇聚于两大主流：集群计算和专用计算机，各有其应用领域。然而，在并行编程领域，情形是模糊和令人失望的。并行编程依然是个难题。实时系统还有很长的路要走。我们回归到用MPI为分布式存储器计算机的人工编程方案，毕竟其他编程风格的尝试都失败了。自动并行化依然是个梦，调试纠错还是个梦魇，路在何方？相对好的解决方案是提供一套有用的工具帮助用户并行编程，正如Daniel Gajski十六年前指出的那样。

致谢

本文由张文哲同学协助从英文翻译而成，特此致谢。

伍民友，上海交通大学计算机科学与工程系教授，网格计算中心首席科学家，新墨西哥大学研究教授。主要研究领域包括大规模集成电路设计，并行与分布式处理系统，多媒体视频系统，互联网与宽频接入网，及无线网络与传感器网络。已发表130多篇学术论文于国际著名期刊及国际学术会议上，SCI的引用总数超过300次。他的名字被载入美国名人录(2000−2001)及国际资讯科学名人录(1997及1999)。他于1996年被选为IEEE资深会员。

参考文献

[GKLS83] D. D. Gajski, D. J. Kuck, D. H. Lawrie, A. H. Sameh, “Cedar : A Large Scale Multiprocessor,” ICPP, pp. 524−529, 1983.

[GaPe85] D. D. Gajski and J−K. Peir, “Essential Issues in Multiprocessor Systems,” IEEE Computer, pp. 9−27, June 1985.

[Seit85] C. L. Seitz, “The cosmic cube,” Communications of the ACM, Vol. 28, Issue 1, pp. 22−33, January 1985.

[Hill85] W. D. Hillis, “The Connection Machine,” The MIT Press, 1985.

[FHKK+90] G.C. Fox, S. Hiranadani, K. Kennedy, C. Koelbel, U. Kremer, C.W. Tseng, and M.Y. Wu, “Fortran D Language Specifications,” Technical Report SCCS−42c, Syracuse University and Technical Reports TR90−079 and TR90−141, Rice University, December 1990.

[BCFH+94] Z. Bozkus, A. Choudhary, G.C. Fox, T. Haupt, S. Ranka, and M.Y. Wu, “Compiling Fortran 90D/HPF for Distributed Memory MIMD Computers,” Journal of Parallel and Distributed Computing, vol. 21, pp. 15−−26, April 1994.

[Li88] K. Li. “IVY: A Shared Virtual Memory System for Parallel Computing,” ICPP, pp.94−101, August 1988.

[Shu95] W. Shu, “Runtime Support for User−Level Ultralightweight Threads on Distributed−Memory Computers,” Journal of Supercomputing, vol. 9 (1/2), pp. 91−−103, 1995.

要 闻 集 锦

IBM公布突破性的Cell宽带引擎计算机

据www.supercomputingonline.com网站2006年2月8日消息报道，IBM公司在当日的新闻发布会上宣布推出一种基于Cell宽带引擎（Cell BE）技术的新型刀片式计算系统。这种基于Cell BE的计算系统是面向那些需要利用Cell BE处理器的强大计算能力和独特功能来处理大量图形密集型数值任务的商务应用。基于Power架构的Cell BE处理器是由IBM、索尼和东芝公司合作开发完成的，该处理器针对计算密集型任务和宽带媒体应用进行了优化，主要适用于计算机娱乐、电影以及其他形式的数字内容等领域。

IBM当日在美国纽约举行的新型BladeCenter产品发布会上对这种基于Cell BE处理器的系统进行了预展。该系统将依靠Cell BE处理器来加速3D绘图、压缩和加密等关键算法，以帮助用户生成并运行高度逼真的、令人感觉身临其境的实时应用。

Cell BE处理器突破性的多核结构和超高速通信能力能够为娱乐和多媒体应用提供大幅改善的实时响应。通过吸收IBM高端服务器中采用的先进多处理技术，Cell BE处理器能够高效地提供“类似于超级计算机的性能”，它尤其适用于多个行业中的高性能工作负载，例如数字媒体、医学成像、航空航天、国防以及通信等领域。

将BladeCenter系统与Cell BE处理器相结合，是IBM公司不断与客户合作以帮助他们使用IBM的技术和经验来实现突破性解决方案的一个例子。最近IBM宣布，它正与美国Mercury Computer Systems公司合作，帮助他们建立针对多种行业的、基于Cell BE处理器的解决方案。IBM还将继续通过Blade.org、Power.org和开放标准与更广泛的用户团体合作，为市场提供更多基于Cell BE处理器的解决方案。

（杜晓梅）