计算机科学 2000Vol. 27№. 6

数据并行计算：概念、模型与系统*)

Data Parallel Computing: Concept, Models, and Systems

李晓明
（北京大学计算机科学技术系　北京 100871）

Abstract From the perspective of application developers, comparing a few programming styles on an example, this article describes the concept and programming models of data parallel computing, as well as ideas and implications of runtime support. Data parallel computing is referred to such a class of application of computing process: the same or similar operations are performed on each elements of the application data set. The main and the most natural programming model is SPMD (Single Program Multiple Data) for this kind of application on distributed memory systems. And SPMD programming model needs runtime support in order to obtain a good trade-off between execution efficiency and ease of programming.

Keywords Data parallel computing, SPMD parallel programming model, Runtime support

一、引言

并行计算，或者并行处理，指的是这样一种努力和相关的研究：利用多个具有计算能力的部件来共同完成一个计算工作，以获得比用一个部件来完成要快的效果。

这显然是一个很自然的想法。历史地看，几乎是自从有了计算机，就有了并行处理的想法和实践。在80年代后期到90年代初期，以寻求对人类面临的若干重大科学问题的解决为背景，并行计算的学术研究和工业实践达到了一个高潮。尽管自90年代中期以来，Internet 和 World Wide Web 风靡全球，成为信息技术领域人们关注的最大热点，计算机工业界已纷纷宣布实行以网络计算为背景的战略转移，计算技术学术界在其主导研究内容上也有了相应的转向，但"并行计算是未来高性能计算的主要途径"的认识依然是确立不变的。作为一个学术领域，它有着丰富的内涵；在计算技术的工业与社会应用中，它拥有一块不可或缺的市场。

特别地，在当今微处理器技术已十分成熟、国际互连网正蓬勃发展的年代，并行计算实际上是在一个更广阔的空间里得到发展和应用。我们可以从如下几个方面的认识来形成这个观点。

首先，网络应用、网上服务的发展对并行处理提出了新的课题。传统上，并行处理主要针对的是科学与工程计算问题，诸如大规模线性方程组的求解、天体系统演变的模拟等。现在人们迫切需要得到及时的网上信息服务和商贸服务，这不仅要求网络要畅通，服务所涉及的信息处理过程也必须足够快。并行处理技术有可能在这里发挥不可替代的作用。例如目前人们十分关注的数据挖掘，就是并行计算能在其中施展的重要商业应用[7]。我们从 McColl 的论文[6]可以看到，即使像 BSP 这样典型的面向科学计算的并行计算系统模型，也有可能用于构造有服务质量保证的 WWW 并行服务器系统。源于"深蓝"战胜国际象棋大师所涉及的计算活动，IBM 新创造了一个提法："深度计算"（deep computing），可以看作是传统高性能并行计算的一个升华。

再者，在80～90年代之间的百舸争流之后，尽管并不是所有技术问题都圆满解决了，但人们对并行计算技术有了比较成熟的认识。这种成熟包括：在现实微电子工艺和制造技术条件下，人们对什么是最有效的并行计算机系统的认识趋于一致，即"商品微处理器技术＋SMP 节点＋简单互连关系"；在并行程序设计方面，90 年代初形成了 HPF（High Performance Fortran）和 MPI（Message Passing Interface）两面旗帜，分别代表一种共享存储和消息传递程序设计模型。近十年实践下来，由于优化编译研制的难度，HPF 仍然有待于被工业界广泛接受，而 MPI 的应用越来越广、越来越大——尽管用它编起程序来要比 HPF 麻烦得多——从而消息传递成为当前主流并行程序设计模

*)本项研究受国家自然科学基金 69873004 和 863 高技术计划 863-306-ZT01-05-1 资助。李晓明　博士、教授、系主任。

型。这些一致和认同使人们能将注意力更多地集中在应用上。

另外，从国际上来看，对并行计算技术的研究和应用的开发正在向纵深发展。水平如何，仍然是综合国力的一个重要标志。以美国为例，petaflops 计划[3]，ASCI 计划，NSF 并行计算中心的调整，Computational Grid 的建设[2]等等，既反映了在新形势下对高性能计算长远的战略部署，也表现了以应用为驱动的强烈的务实态度。在另一方面，即使是瑞士那样的小国，其科学计算中心（Swiss Center for Scientific Computing）所开展的活动也使我们的确有一种紧迫感[10,11]。

还有，高性能计算技术的广泛应用提出了对相应人才的大量需求。这方面一个重要的标志是对"Computational Science and Engineering（CSE）"学科的呼唤在全球范围都能听见。IEEE 也新增了相应的学术期刊。从大概 90 年代初开始，美国的不少大学，包括加州理工学院、斯坦福大学、芝加哥大学和锡拉丘茨大学等，都在张罗提供和 CSE 有关的学位。主要是由于教育观念上的原因，这种思想尚未被广泛接受，但人才市场的需求仍然是客观存在的。特别地，前述高性能计算应用范围的扩大也引起了相应人才培养内容的扩展：前些年主要谈的是 HPC，现在开始提"Internetics"[12]。

最后，国内在大型并行计算机系统的生产方面，一种稳定的格局也基本形成。除国防科大"银河"、中科院计算所"曙光"等系统的系列化发展外，以清华"探索 108"为代表的一类基于 PC 机群的系统（昵称为 Beowulf[5]）在一些院校和科研院所也相继攒建出来。以中科院软件所有关研究人员的工作为代表，大型真实的并行应用程序也由我们自己逐渐开发出来。北京大学等学校开出了并行程序设计等研究生课程[8]，迈出了高性能计算应用人才培养的第一步。

上述这些，试图说明这样一个中心问题，即并行计算不仅是一个引人入胜的科学领域，而且目前正处于一个新的发展时期。这个时期的显著标志就是广泛的应用。在因特网蓬勃发展的背景下，并行计算的用武之地将更加广阔。本文的目的就在于从应用（程序设计）的角度阐述有关的技术和概念。

二、与数据并行计算有关的一些概念

"数据并行"（data parallel）或"数据并行性"（data parallelism）的提法，最早大概出现在 1986 年 Hillis 的论文[1]中，指的是算法执行的过程具有对大量数据元素同时做相同或者类似操作的特征。在那篇论文中，作者针对他们设计的 CM-2，一种含有 65536 个处理单元的并行计算机，宣传数据并行的"普遍适用性"。文章通过例子试图说明：有些看来是串行的问题，也可以设计出适当的"数据并行算法"来得到求解过程的加速。文章最激进的观点是，只要应用问题具有大数据量的特征，就有可能用数据并行的办法来高效处理，而与问题的领域和类型关系不大。

从某个角度考虑，这个观点也是有道理的：如果一个计算问题涉及到一个很大的数据集合，难以想象对这个集合中的每一个元素进行各不相同的处理。例如在程序中，有一个很大的数组，通常是用循环来遍历数组的各个元素（从而更新它们的方式就相同或者相似），而不是用完全不同的赋值语句来一个个访问那些元素（从而有可能做完全不同的处理）。这在某种程度隐含着：如果问题的数据量大，则对那些数据的处理方式必然相似，于是就应该能找到求解该问题的数据并行算法。

不过我们这里从一个更直接的角度讨论数据并行性，认为数据并行首先是对一类应用问题的典型求解过程特性的一种刻画。那些应用问题有两个基本特征。一是数据量很大，特别地，如果该问题是对某种自然现象或客观事物的模拟（这样的模拟实际上是大规模并行计算的主要应用），则这种模拟的精度通常随数据规模的增大而提高；二是在典型的求解过程中（即人们已知的好算法中），经常出现一些"段落"（或者说该求解过程主要由这样一些段落构成），对大量相同类型的数据元素进行一系列相同或者类似的操作。我们称这样一类问题为数据并行问题。

这是从实际应用程序开发的角度中得到的表述，尽管不是很精确的定义，但所描述的情形是清楚的。例如，偏微分方程（场问题），常微分方程（粒子问题），蒙特卡罗方法（随机优化）和各种空间变换方法（线性代数，FFT 等），大概从数学模型的意义上覆盖了绝大多数科学和工程计算问题。它们常都表现了很强的上述数据并行性。在偏微分方程的数值求解方法中，典型内核是根据邻近点的值对数据域中的每一个点做同样方式的更新；而以 N 体问题为代表的粒子问题要根据每个粒子和其它所有粒子的相互作用情况，来决定系统下一时刻的状况，而那种相互作用的数学模型对每个粒子都是同样的。这些都属于数据并行描述的范畴。Fox 在对大量的科学和工程问题求解过程的研究中得出估计：科学和工程问题中约有 80% 可归纳为数据并行问题，或者说表现出了很强的数据并行性。

上面是从数学模型的角度，看不同数学性质的问题都可能是具有数据并行特征的。从应用问题的计算过程的表现形式上，数据并行问题通常被分为规则问题与非规则问题两大类。这个分类又可以从应用问题和程序设计两个不同的层次（角度）来理解。前面谈到过，数据并行问题的特征之一是涉及到大量的数据，称

为问题的数据域(data domain)、例如一幅数字图像、飞行器周围的流场,宇宙空间中两两相互作用的星体等等。在程序中,这样的数据域几乎都是用一个或者多个数组来表达的。问题求解算法的执行常常就是通过循环语句的控制,对这种数组的某种遍历,按照特定的规则实施对该数据域中(该数组中)每个元素的状态更新。这种更新常常需要利用数据域中其它若干元素的值。如果数据域中一个元素和其它元素的这种关系能够"规则地"表现出来,例如 A(I)=A(I-1)+A(I+1),则称该问题为规则问题(regular problem),它反映了算法在该数据域上作用的某种规则性,例如一些典型的图像处理算法;否则,就称为"非规则"问题。非规则问题最常见的形式是用数组元素来引用数组元素,例如 A(B(I))。除了某些数据域(例如非规则流场)的性质会直接导致这种数组的引用方式外,涉及到稀疏矩阵的计算是导致非规则问题的一个重要来源。这是由于,为节省存储空间,稀疏矩阵的计算机表示常常是用两个一维数组来完成的[9],从而对于矩阵元素的引用就必须通过用数组元素做下标来完成。Fox 的研究认为,在数据并行问题中,非规则问题至少占 50%。

这种关于规则与非规则问题的分类有什么实际意义呢?对普通串行程序设计来说,意义不是那么大。因此我们在通常的程序设计课程中根本不提到这样的概念。对于 SMP 系统的并行程序设计,意义也不大。对于应用程序员来说,这种分类的意义主要体现在对分布存储系统的程序设计上。这样的计算机系统由若干"节点"按某种方式互连而成,每个节点有自己的处理器和存储器。处理器访问自己的存储器很快,而访问其它节点的存储器(也称为通信)很慢。并行计算中一种常见的有效技术称为"区域划分",指的是在考虑如何让多个节点来共同完成一个计算任务时,将上述数据域在节点的存储器之间进行分配,让每个节点负责其中一部分元素所涉及的计算任务(即那些元素状态的更新)。这种数据分配质量好坏的评判有几个方面,一是负载平衡,二是减少通信,三是降低运行管理开销。对于规则问题来说,数据访问的模式在编程序时是清楚的,因此程序员能够作出较好的抉择;但对于非规则问题,数据引用的具体信息只能在运行时得到,程序员难以确定他认为合适的数据分配方案。也就是说,A(I),I=1,2,…,N 给程序员的概念(依次访问数组 A 的各个元素)比 A(B(I)),I=1,2,…,N 给以的(A 的元素被访问的次序不清楚,有些元素可能被重复访问,等等)要清楚得多。因此,从并行计算技术的角度来看,处理非规则问题要困难得多。

三、并行程序设计模型

既然科学和工程实践中大量实际的计算问题都有数据并行的特征,研究高效解决它们的方法和工具就是有意义的。而且和许多引人入胜的科学问题一样,数据并行的概念表述简单,但同时有丰富的内涵,这就给我们展开了一幅主题鲜明,同时又有声有色地开展研究工作的画卷。从应用人员的角度看,如何向并行计算机系统表达求解问题的算法过程,也就是并行程序设计,是最直接,也是最基本的需要。任何满足这个需要的技术都要在两个最主要的因素之间寻求某种权衡,一是程序设计的方便性,二是程序运行的高效性。它们是在算法确定以后,求解一个问题所花时间的两个主要部分。下面的讨论针对上述分布存储计算机系统。需要指出的是,尽管"分布存储"和"共享存储"也用来刻画程序设计模型的性质,但它们和赖以运行的计算机系统结构模型是分布存储还是共享存储没有必然联系。

首先是数据并行程序设计模型。从实用的角度讲,程序设计模型表现为一组一般性规则和约定,确定了程序员写程序时的感受和写出的程序的一种风格。例如,FORTRAN 77,HPF,和 MPI 体现了三种不同的程序设计模型。从数据访问的角度看,即分别为随机共享存储模型、非一致访问的共享存储模型和消息通信模型。讲 FORTRAN 77 是随机共享存储程序设计模型,指的是当程序员编程序的时候,他的头脑中无形地有一个统一的存储器,所用到的变量都从那个存储器中来,在访问时间上没有区别。称 HPF 是非一致访问的共享存储程序设计模型,指的是当程序员用 HPF 编程序的时候,一方面他看到的是一个统一的逻辑存储空间(表现在当他引用一个数组元素时,只需给出该元素在数组中的下标),另一方面他还关心一个计算所用的数据是否都在同一个物理存储中;他知道如果数据不在一起,则要发生通信,从而执行效率降低。而利用 MPI 编程的时候,程序员在程序中引用的只有本地数据;如果需要用到其他节点中的数据,必须首先显式将那些数据通信过来,成为了本地数据,然后才能引用之。

SPMD(Single Program Multiple Data,单程序多数据)是当前在分布存储系统上,针对数据并行问题求解最常用的程序设计模型。它在数据并行计算领域的地位相当于 client/server 模型在分布计算领域的地位。其基本观念是将问题的数据分布到参与计算的处理器上,让每个处理器拥有数据的一部分;写出一个程序来,让这个程序在所有参与计算的处理器上运行,使之集体等价于某个全局程序的运行效果。由于数据并

行问题的第二个特点——对数据做相同或者类似的操作——这种思想的有效实现成为可能(否则程序中会有太多的情况分枝,以至于失去"单程序"的意义)。SPMD的概念大概是Frederica Darema或Geoffrey Fox首先提出的(在1997年6月的一次DARPA PI会议上,有一发言人在台上提起Fox发明了SPMD,Fox在台下立刻说是Frederica发明的,她当时也在台下,没有表示异议)。

SPMD程序设计模型的命名反映其两个基本特征。首先,程序员看到的是一个分布的数据空间,他要将应用问题的数据分布到参与计算的处理器中。例如,应用问题要求的数据可能是一个100×100的数组,现在有4个处理器来参与计算,程序员要考虑是将这100×100数组分成4个50×50,还是4个100×25,或者是4个25×100的数组分别赋予各个处理器。这种数据分布隐含有两个直接的要求,一是显式通信:当某处理器中一个元素的计算需要另一个处理器中的某个元素参与时,需要先进行通信将该元素取过来才能计算;二是下标转换:应用问题的串行求解算法通常是针对全局数据(逻辑上的全局数组)的,要将这个算法转换为并行的,由多个处理器来协同执行,每个处理器操作的都是本地数据(本地数组)。于是:要使算法能正确执行,有一个数组下标从全局到局部,局部到全局的转换问题。这种转换常常还体现在循环控制变量的范围中。举个例子来说,假设应用问题本身要求做的是A(I)=I,I=1,2,…,100。这件事分给两个处理器(分别用pid=0和1代表)来做,每个处理器负责50个元素。它们执行的节点程序就会类似于A(I)=I+50*pid,I=1,2,…,50。循环控制变量都是从1到50,但赋值语句的右手部不能再是简单的I了。这些都是程序员要考虑处理的。换句话说,程序员面对的是由若干局部数据空间构成的一个全局空间,要时刻关心数据的引用在这两种空间中的对应关系,以及数据在不同局部空间之间的运动。其次,尽管有多个处理器参与计算,但程序员只写一个程序,同样这一个程序在所有处理器上执行。这一个程序不仅要实现算法所要求的计算,还要在处理器之间自我协调,达到算法所要求的总体效果。这里有一个重要的问题要解决:各个处理器所从事的计算只是"相同或者相似"的,也就是说是可能有所不同的。我们需要有一个办法来使运行着相同程序的不同处理器实际做的事情有所不同。典型的方法是在程序中让每个处理器能够判断自己的标志(可能是一个编号),让这个标志成为区别处理器行为的依据。上述例子中的pid就是这个概念的具体体现。

不同的程序设计模型不一定是互斥的,它们可能反映的是同一事物的不同侧面,或者不同层次。例如,SPMD可能基于消息传递之上,而消息传递和共享存储也可以相互模拟。

四、数据并行程序设计与运行支持系统

模型只是确定了一个概念、一个框架和若干基本要素,它需要有一定的系统支持才能得以应用。我们称这种系统支持为模型的实现。模型的实现有两种基本方式:程序设计语言和运行支持库。例如HPF是用语言的方式实现它所代表的模型,而MPI是以运行库的形式实现的消息传递模型。同一个模型可能有不同的实现方式,例如HPF和HPC++是两种体现同一模型的语言;MPI和PVM是两种体现消息传递模型的运行库。下面,我们以Laplace方程的Jacobi迭代求解内核为例,看不同程序设计模型的具体体现。首先,在单处理器计算机上,这个内核的FORTRAN 77代码为:

```
REAL A(N+2,N+2),B(N+2,N+2)
...
初始化数据域A的边界,并拷贝到工作区B中
DO K=1,NSTEP
    DO I=2,N+1
        DO J=2,N+1
            A(I,J)=(B(I-1,J)+B(I+1,J)+B(I,J-1)
                   +B(I,J+1))*0.25
        END DO
    END DO
    将A的数据拷贝到B
END DO
```

计算过程是简单清楚的,即让数据域内部的每个元素用周围4个元素的平均值更新,迭代若干次,更新的时候所用的都是上一次迭代产生的值。

现在考虑用某个4处理器的分布存储系统做并行计算。遗憾的是,FORTRAN 77程序在分布存储系统上的并行化仍然是人们努力追求而尚未达到的目标(也就是说,我们一般地不指望上述FORTRAN 77程序能被自动并行化编译在分布存储并行计算机系统上运行)。如果是用HPF编程,则我们可以有等价的:

```
      REAL A(N+2,N+2)
! HPF$ PROCESSORS P(2,2)
! HPF$ DISTRIBUTE A (BLOCK, BLOCK) ONTO P
      ...
      初始化A的边界
   DO K=1,NSTEP
         FORALL(I=2:N+1,J=2:N+1)
           A(I,J)=(A(I-1,J)+A(I+1,J)+A(I,J
                  -1)+A(I,J+1))*0.25
      END DO
```

这里讲的是将数组A在两个维上切分成相等的4块,每个处理器分得一块。但在程序的所有其它地方,对数组元素的引用都是全局的。除此以外,我们还看到HPF提供的FORALL的并行处理语义(对所有I,J,算完赋值语句右手部后再进行并行赋值)免去了程序员用临时数组B的必要。尽管他在程序中没有安排,但程序员能够认识到,在执行这个FORALL语句时,处理器之间会有一些通信发生(在数据的边界)。这样的程序也必须有特别的编译器处理才能在上述并行计算机系统上运行。由于程序中显式给出了一些关于数据划分和并行计算的说明,我们可以指望这样的编译

器是可能的(这正是 HPF 语言提出的基点). 最后,如果我们利用 MPI 来写一个运行于上述 4 处理器系统的 SPMD 节点程序,则

```
REAL A(N/2+2,N/2+2),B(N/2+2,N/2+2)
...
初始化数据域 A 的边界,并拷贝到工作区 B 中
DO K=1,NSTEP
    CALL MPI_SEND(有关参数):将自己数组 A 的数据一
        个边界送给相邻的处理器
    CALL MPI_RECV(有关参数):从该相邻的处理器接收
        数据边界到自己 B 的适当行或列
    CALL MPI_SEND(有关参数):将自己数组 A 的数据另
        一个边界送给相邻的处理器
    CALL MPI_RECV(有关参数):从该相邻的处理器接收
        数据边界到自己 B 的适当行或列
    DO I=2,N/2+1
        DO J=2,N/2+1
            A(I,J)=(B(I-1,J)+B(I+1,J)+B(I,J-1)
                +B(I,J+1)) * 0.25
        END DO
    END DO
    将 A 的数据拷贝到 B
END DO
```

同上面两个程序相比,这个程序更像第一个:它本身就是 FORTRAN 77程序,只不过是节点程序而已。节点程序的含义是,它是直接运行在多处理器计算机系统中节点处理器上的程序。作为节点程序,它的数组是局部数组,其大小只是问题数组的四分之一(多的一行一列是为了程序设计方便增加的"阴影区",ghostarea),程序中所有操作都是针对局部数据进行的,其次,这里有显式通信的调用(MPI_SEND,MPI_RECV)。虽然显式通信部分略写了,但我们能够想象得到,要以 SPMD 的风格精确描述这些通信并不是一件直截了当的事情(例如,每个处理器的"相邻处理器"是各不相同的)。

仔细对比这三个程序,体会它们之间的等价性和差别是很有意义的。一般地,SPMD 程序不需要特别的编译器,但需要运行支持,这种运行支持就体现在对特定的库函数的调用上,如同上述 MPI_SEND,MPI_REVC。取决于对程序设计模型实现的层次,对 SPMD 程序的运行支持可能在一个很宽的范围内定位,系统设计的技术和艺术就体现在这种定位中。目标还是那么一个:在程序设计的方便性和程序运行的效率之间获得最好的权衡。直接用 SEND/RECV 这样的点对点通信具有很大的灵活性,但层次较低,程序设计不方便。因此,现在做 SPMD 运行支持的一个倾向是提供较高层次的库函数。例如,考虑到上述节点程序中 SEND/RECV 所代表的数据通信模式具有一定的普遍性,我们有可能用一个函数来概括它们,从而节点程序变成:

```
REAL A(N/2+2,N/2+2),B(N/2+2,N/2+2)
...
初始化数据域 A 的边界,并拷贝到工作区 B 中
DO K=1,NSTEP
    CALL EDGE2GHOST(有关参数)
    DO I=2,N/2+1
        DO J=2,N/2+1
            A(I,J)=(B(I-1,J)+B(I+1,J)+B(I,J-1)
                +B(I,J+1)) * 0.25
        END DO
    END DO
    将 A 的数据拷贝到 B
END DO
```

对程序员来说,这种支持就使得编程容易,而得到的程序可读性也就高多了。

总结 在本文,我们介绍了与数据并行计算有关的一些主要概念,贯穿了我们关于并行处理在计算机科学技术领域中地位的一些认识和观点。它们包括两个主要方面。第一,对以大规模并行计算为技术核心的高性能计算技术的追求是一个永恒的课题。对高性能计算技术掌握和应用的能力是一个国家综合国力的一个重要标志。在当今社会信息化建设的浪潮中,并行计算应用的领域不仅没有缩小,而且有了新的扩充——从几乎仅限于科学与工程计算的王国,正向商业数据和信息处理迈进。第二,随着人们对并行处理系统结构认识的相对稳定,并行计算技术研究与发展策略本身也就更注重应用牵引,要用真实的应用带动技术的集成和发展。对于在我国以计算机专业为背景从事并行处理技术研究的人们来说,这实际上是一个新的挑战,要求学计算机技术的人们也懂一些适合于并行处理的应用,习惯于从应用需求的角度来研究技术。为此,我们向有进一步兴趣的读者推荐参考文[7],其中的二、三两章值得仔细研读。

参 考 文 献

1 Hillis M T, Steele G L. Data parallel algorithms. Communications of the ACM, 1986,29:1170～1183
2 Alliance/NCSA. ACCESS, From Supercomputers to the Grid. Vol. 11, No. 1 Fall/Winter 1999
3 Sterling T, MacDonald M J. The Proceedings of The Petaflops Frontier Workshop. 1995(Feb)
4 Fox G C, Williams R D, Messina P C. Parallel Computing Works! San Francisco, Morgan Kaufmann Publishers, Inc., 1994
5 李晓明. 一种值得注意的新型计算环境. 计算机世界(专家观察),1999. 5 3
6 McColl W F. Foundations of Time-Critical Scalable Computing. In: Proc. of the XV. IFIP World Computer Congress (Fundamentals—Foundations of Computer Science), 1998(Aug):93～105
7 Culler D E, Singh J P, Gupta A. Parallel Computer Architecture (A Hardware/Software Approach). San Francisco, Morgan Kaufmann Publishers, Inc., 1999, Chapter 2～3
8 李晓明. 北京大学并行程序设计课程站点. 1998-2000. Available at: http://arch.cs.pku.edu.cn/p2
9 Press W H, et al. Numerical Recipes in FORTRAN. 2nd Ed. Cambridge University Press, 1992. 71～76
10 CrosSCuts. Newsletter of Swiss Center for Scientific Computing, 1999,18(2), Available at: http://www.cscs.ch
11 SPEEDUP. Journal of SPEEDUP Society, 1998, 12(1) Available at: http://www.speedup.ch
12 Fox G C. Internetics: Technologies, Applications and Academic Fields. In: A. J. G. Hey, ed. Feynman and Computation, exploring the limits of computers, 241～256