并行计算性能的"双流"分析*)

Double-Stream Analyses for the Performance of Parallel Computation

乔香珍

(国家高性能计算中心(北京) 北京100080)

(国家智能计算机研究开发中心 北京100080)(中国科学院计算技术研究所 北京100080)

Abstract The generalized speed-up is estimated according to the "double-stream" analyses. The term "decreasing ratio" is used to describe the influence of the hierarchical memory and the characteristics of parallel application on the performance. The optimization principles for parallel computation are also given.

Keywords Parallel computation, Parallel performance, Double-stream analyses, Decreasing ration, Generalized speed-up

1 引言

CPU 或节点的重置,是并行机系统获得高性能的重要原因之一,这是早期并行算法研究的基本假设。然而,随着 CPU 性能按摩尔定律的不断攀升,提供高速的、与计算速度相匹配的供数能力成了高性能计算的更为突出的矛盾。这对高性能并行机系统尤为重要[1]。对于分布存储并行机系统,数据来源多样化,如:寄存器、Cache(高速缓存)、本节点内存、其它节点内存、本地磁盘、其它节点磁盘、"共享"磁盘,等等。这一多层次数据访问方式通称为层次存储访问技术[2]。在这一层次访问中,越靠近 CPU 的层次,存储访问速度越高。如 MIPS R8000芯片[3]的一级数据 Cache 访问速率是每时钟周期128位,由于流水线数据流的采用,使得 Cache 能以高带宽支持高速浮点计算。相比之下内存访问速度为每53时钟周期128字节,Cache 访问速率约是存储访问速率的6～7倍,因而在一般情况下,只有当计算所需数据已在寄存器或 Cache 中,CPU 的高速计算才有保证,其它情况都会引起计算速度的明显降低,因而近年不少研究者注意并讨论了存储访问层次对性能的影响[6~8]。

一般来说,一个并行机系统的峰值速度是该系统各节点峰值速度之和,而单节点的峰值速度是根据处理器的主频、CPU 内部件的并行性、流水、链接等特性推算出来的。其实峰值速度的推算暗含了两个重要假设,一是计算充分利用了 CPU 运算部件的并行性,二是计算所需数据已到位(即已在 CPU 的寄存器中,或在一级数据 Cache 中),显然对一般计算问题来讲,这两个假设是不现实的,因而实际计算问题达不到并行机峰值速度或相差甚远的现象是常见的,本文即从计算、供数两个方面分别对串、并行机系统及其计算问题性能作出定性描述,特别是将存储访问、通信、访盘等问题统一在"供数"速度之下讨论(简称之为"双流"-"计算"与"数据"流程分析),并用"降速因子"和"比例因子"的概念来描述计算问题(或应用软件)实际性能相对于计算机系统峰值性能的降低程度。对分布存储并行机系统,按双流分析的方法给出并行计算"广义加速比"的一个计算公式。在此基础上用"广义均衡性","广义瓶颈"和"广义局部性"静态分析应用软件在相应体系结构上的性能,并用曙光并行机的应用实例说明了优化原则和方法。

本文分别给出串行机和分布存储并行机系统及相应计算的"双流"分析,给出分布存储并行机系统上并行计算的"广义加速比"的一种计算方法,再给出优化原则和若干实例,最后是结论。

2 串行计算机系统和串行计算问题的"双流"分析

典型串行计算机系统性能特征可用图1-a 表示。对于 CPU 而言,其峰值速度(简记为 V_p)由主频周期 t_0 与器件流水并行性 p_0(p_0是大于等于1的正整数)决定,这是理想状态。往往计算所需数据在内存储器(图

*)本课题得到国家自然科学基金重点项目(编号69933020)和国家863高技术项目的资助。乔香珍 研究员,主要研究兴趣为并行处理,并行算法与应用。

中 mem)或磁盘(图中 disk)中，假设从内存或磁盘获取计算所需数据(以字为单位)的时间分别是 t_m 与 t_d，则有：$t_d > t_m > t_0$。这里 t_m 之值取决于存储访问带宽(B_m)、延迟(L_m)和访问冲突 C_m。类似地 t_d 依赖于 L_d、B_d 和 C_d。

一个串行计算问题在图1.a 所示计算机系统上的实际性能可由图1.b 表示，首先是问题的计算性质(如运算类型，流水链接性，并行部件的可利用性等)的影响，我们将此总括为 P_0(P_0满足$1 \leqslant P_0 \leqslant p_0$)。即使在计算所需数据已到位的情况下，这些影响也使计算速度由峰值 V_p 降低为 V_0。

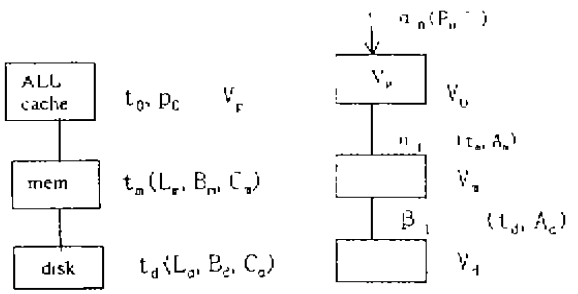

图1.a 串行机时间关系　图1.b 串行计算性能分析

这里我们引入"降速因子"的概念：

1)$\alpha_0 = V_0/V_p$：表示计算问题的纯计算性质引起的运算速度相对于峰值速度的降低程度，可以看出，$\alpha_0 = P_0/p_0 (0 < \alpha \leqslant 1)$，理想状态，$P_0 = p_0$，则有 $\alpha_0 = 1$，$V_0 = V_p$。

2)$\alpha_1 = V_m/V_0$：表示计算问题的访存速度相对于 V_0的降低程度。这里假设内存储器的供数速度为 V_m，其值依赖于 t_m-硬件性质和计算问题的存储访问性质 A_m。

3)$\beta_1 = V_d/V_0$：表示计算问题访盘速度相对于 V_0 的降低程度，这里 V_d 表示磁盘向内存储器的供数速度，其值决定于 t_d 和应用软件的访盘性质。

显然有：$0 < \alpha_1, \beta_1 < 1$。这里，我们引入以下记号：

N_0：一个计算问题 A_0中的总算术运算(浮点运算)量，T_0为 CPU 完成全部 N_0浮点运算需要的时间，$V_0 = N_0/T_0$。

N_m：在一个串行计算机上完成计算问题 A_0所需要的不能与算术运算相重叠的存储访问量，T_m 为完成存储访问量 N_m 所需要的时间，$V_m = N_m/T_m$。

N_d：在一个串行计算机上完成计算问题 A_0所需要的既不能与算术运算相重叠，又不能与存储访问相重叠的磁盘访问量，T_d 为完成存储访问量 N_d 所需要的时间，$V_d = N_d/T_d$。

于是，在一个串行计算机上完成计算问题 A_0所需要的总时间为：$T_1 = T_c + T_m + T_d$。

令：$S_m = N_m/N_0$，$S_d = N_d/N_0$，A_0在一个串行计算机上的实际性能是 $V_{rs} = N_0/T_1$。在以上假设下，A_0在串行机系统上能获得的实际速度 V_{rs}与 V_0的关系可由下述引理给出：

引理：根据以上假设，计算问题 A_0在串行机系统上能获得的实际速度 V_{rs}可表示为：

$$V_{rs} = V_0/(1 + S_m/\alpha_1 + S_d/\beta_1)$$

证明：$\because V_0 = N_0/T_0$，$V_{rs} = N_0/(T_0 + T_m + T_d)$，

$$\therefore V_{rs}/V_0 = T_0/(T_0 + T_m + T_d)$$
$$= (N_0/V_0)/(N_0/V_0 + N_m/V_m + N_d/V_d)$$
$$= 1/(1 + N_m/N_0 * V_0/V_m + N_d/N_0 * V_0/V_d)$$

使用记号 α_1、β_1和 S_m、S_d，则有：$V_{rs} = V_0/(1 + S_m/\alpha_1 + S_d/\beta_1)$，证毕。

容易看出：1)$S_m = S_d = 0$时，$V_{rs} = V_0$；2)V_{rs}是 S_m 和 S_d 的减函数，即 α_1、β_1固定时，S_m 或 S_d 增加使 V_{rs}减少；3)V_{rs}是 α_1和 β_1的增函数，即 S_m、S_d 固定时，α_1、β_1的提高使 V_{rs}提高。

该引理告诉我们，在串行机系统上一个计算问题的实际性能依赖于多种因素：计算问题的算术运算性质；存储访问特点(存储访问量和数据访问性质)；计算与存储访问的可重叠性；硬件结构性质(运算和存储两方面性能)。

3 分布存储并行机系统及其并行计算的"双流"分析

一个典型的由 p 个处理器(或计算节点)组成的同构分布存储并行机系统的性能特征由图2给出[9]。实际上它是由 p 个独立的类似于图1.a 所示单机系统通过互联网(图中 network 部分)构成的。系统没有共享主存储器，但包含一个共享磁盘(图中 C. Dusk)系统，共享磁盘系统通过 I/O 系统和互联网与各节点相连。

图中 node0部分给出了结构性能的有关参数，t_0、p_0，t_m，t_d 含义同串行机(见图1.a)，网络通信时间由 t_c 给出，它决定于通信延迟(L_c一软、硬件总延迟)，带宽(B_c)和网络冲突(C_c)。共享磁盘 I/O 时间由 t_{cd}给出，它取决于延迟(L_{cd})，带宽(B_{cd})，盘冲突(C_{cd})和网络有关参数(C_{net})，现在我们来分析分布存储并行机系统并行应用程序的性能。图2中 node(p-1)与网络、共享磁盘部分列出有关参数。对于单计算节点，V_0，α_c，α_1，β_1、V_m，V_d 含义同图1 b，类似于2节的分析，假设并行应用程序在单个节点上的运算速度为 V_n，从并行机系统的整个并行计算来看，性能受以下因素影响。

1)并行性，在理想状况下，一个并行算法的性能应是相应串行算法性能的 p 倍，但在算法本身并行度为 P_1的情况下，其速度将为 $P_1 * V_{rs}$。

2)对其它节点内存储器的访问，实际上这是通过节点间通信实现的。假设通信速度是 V_c（V_c 依赖于 t_c，通信量与通信性质），则有由于消息传递引起的通信相对于单节点计算的降速因子 $\alpha_2=V_c/V_{ns}$。

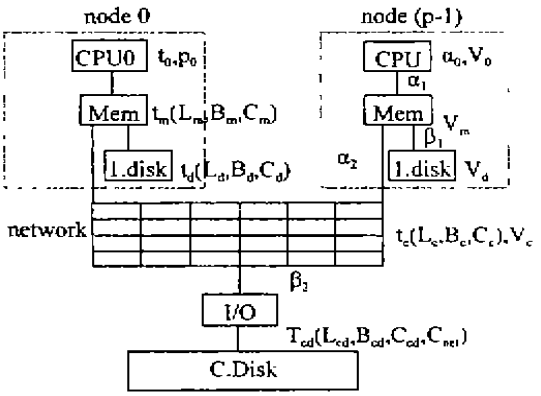

图2　分布存储并行机系统

3)对共享磁盘的访问则需通过网络通信和 I/O 处理机实现。假设共享磁盘访问速度是 V_{cd}（依赖于 t_{cd} 与应用程序访盘性质），则有由于对共享磁盘访问引起的访盘相对于单节点计算的降速因子 $\beta_2=V_{cd}/V_{ns}$。一般有 $V_{cd}<V_c$，$V_{cd}<V_d$。

为使问题简化，又不失一般性，我们假设应用程序中不包含对其它节点磁盘的访问，p 个处理机组成的并行机系统共同完成一个并行计算问题 W_p，引入以下记号：

N_{op}：一个计算节点所完成 W_p 的部分计算所包含的总算术运算量。为简单起见，假设 $N_{op}=N_0/p$，这里 N_0 为 W_p 相应的串行计算的总算术运算量，T_{p0} 为每个计算节点完成 N_{op} 浮点运算需要时间的最大值。

N'_m：一个计算节点所完成 W_p 的部分计算所需要的不能与算术运算 N_{op} 相重叠的局部存储访问量，T_m 为每个计算节点完成存储访问量 N_m 所需要时间的最大值。

N'_d：一个计算节点所完成 W_p 的部分计算所需要的既不能与算术运算 N_{op} 相重叠，又不能与局部存储访问 N'_m 相重叠的局部磁盘访问量，T_d 为每个计算节点完成局部磁盘访问所需要时间的最大值。

N_c：一个计算节点为完成 W_p 的部分计算所需要的不能与 N_{op}，N'_m，N_d 相重叠的计算节点间的通信量，T_c 为每个计算节点完成通信 N_c 所需要时间的最大值。

N_{cd}：一个计算节点为完成 W_p 的部分计算所需要的不能与 N_{op}，N_m，N'_d 和 N_c 相重叠的共享磁盘访问量，T_{cd} 为每个计算节点完成磁盘访问 N_{cd} 所需要时间的最大值。

同样采用第二节中的符号 N_0，N_m，N_d，S_m，S_d，α_1 和 β_1，并记：$S_c=N_c/N_0$，$S_{cd}=N_{cd}/N_0$，$S'_m=N'_m/N_0$，$S'_d=N'_d/N_0$。

假设 W_p 相应的串行计算所需要的计算时间为 T_1，且 $T_1=T_0+T_m+T_d$，相应的性能为 V_{ns}。每个计算节点完成 W_p 所需要时间的最大值为：$T_p=T_{p0}+T'_m+T'_d+T_c+T_{cd}$，则完成 W_p 相应的速度为 $V_{rp}=N_0/T_p$。在评价一个具体应用问题在并行机系统上的性能时，并行加速比仍是实用、简单的一个参数。实用中有多种并行加速比，如果不考虑所有层次上的存储访问开销，其理论并行加速比为：$S_p=T_1/T_{p0}$。该加速比仅考虑了算术运算性质，为分析分布存储并行机上应用问题的实际性能，我们使用广义加速比的概念[10]：

并行计算问题 W_p 的广义加速比 S_g 定义为 $S_g=W_p$ 的并行执行速度/W_p 的串行执行速度。据此定义，我们有：$S_g=V_{rp}/V_{ns}$。相应的效率为：$E_g=S_g/p$。按照以上分析，分布存储并行机系统上的"广义加速比"可用下述定理计算：

定理　根据以上假设，在分布存储并行机系统上 S_g 与 S_p 的关系可表述为：

$$S_g=S_p*(1+S_m/\alpha_1+S_d/\beta_1)/(1+S_p*(S'_m/\alpha_1+S'_d/\beta_1+S_c/\alpha_2+S_{cd}/\beta_2))。$$

证明　$\because V_{0p}=N_0/T_{p0}$，$V_0=N_0/T_0$，

$\therefore S_p=T_0/T_{p0}=V_{0p}/V_0$；

$\because V_{rp}=N_0/T_p$，且 $V_{ns}=N_0/T_1$

$\therefore S_g=V_{rp}/V_{ns}$

$=(T_0+T_m+T_d)/(T_{p0}+T'_m+T'_d+T_c+T_{cd})$

$=(N_0/V_0+N_m/V_m+N_d/V_d)/(N_0/V_{0p}+N'_m/V_m+N'_d/V_d+N_c/V_c+N_{cd}/V_{cd})$

上式的分子与分母同乘以 V_0/N_0，则有：

$$S_g=(1+N_m/N_0*V_0/V_m+V_0/V_d*N_d/N_0)/(V_0+V_{0p}+N'_m/N_0*V_0/V_m+N'_d/N_0*V_0/V_d+N_c/N_0*V_0/V_c+N_{cd}/N_0*V_0/V_{cd})$$

$$=(1+S_m/\alpha_1+S_d/\beta_1)/(S_p^{-1}+S'_m/\alpha_1+S'_d/\beta_1+S_c/\alpha_2+S_{cd}/\beta_2)$$

$$=S_p*(1+S_m/\alpha_1+S_d/\beta_1)/(1+S_p*(S'_m/\alpha_1+S'_d/\beta_1+S_c/\alpha_2+S_{cd}/\beta_2))。$$

证毕。

该定理说明了：

1）$S_c=S_{cd}=S_m=S_d=S'_m=S'_d=0$ 时，$S_g=S_p$，即只有在既无通信，又无对共享磁盘的访问时，应用问题的广义并行加速比才仅仅依赖于算法并行性；

2）比值 S_m/α_1，S_d/β_1，S'_m/α_1，S'_d/β_1，S_c/α_2 和 S_{cd}/β_2 决定广义并行加速比 S_g。这意味着在分布存储并行机系

统上，应用问题的广义并行加速比取决于硬件性能和应用问题的计算与存储访问两方面的特性：

(1)S_g是比值S_m/α_1和S_d/β_1的增函数，即在其它量固定的情况下，S_m/α_1和S_d/β_1的提高使S_g提高。这说明了两个问题：对于同一个计算问题，在存储访问速度相对于CPU计算速度降低较小(α_1和β_1的值较大)的并行机系统上可能获得较大的并行加速比，这是应用中常遇到的现象；对于同一个并行计算问题，如果相应串行计算的存储访问效率较低，则可能获得较大的并行加速比，甚至于"超线性加速比"，这在应用中也是不罕见的。为了防止在分析中，"CPU速度越低，并行加速比越大"(上述第一种现象)和"串行计算的效率越低，并行加速比越大"(上述第二种现象)的不正常现象发生，这里我们假设所用串行计算是已作了较好计算和存储访问优化的，

(2)S_g是S_m、S_d、S_c和S_{cd}的减函数，即通信、共享磁盘与相应存储访问量增加时、应用问题的并行加速比降低；

(3)在S_m、S_d、S_c和S_{cd}固定的情况下，α_1、β_1、α_2和β_2的增加使S_g增加，即提高系统通信、存储访问和I/O效率是提高并行应用软件性能的有效手段。

4 公式分析与并行应用软件优化原则

根据上述公式，一个并行应用问题在分布存储并行机系统上的实际性能依赖于降速因子(α_1、β_1、α_2和β_2)。较大的降速因子(即相应部件的存储访问速度相对于CPU计算速度降低不大)引起较大的加速比，这些降速因子不仅仅依赖于并行机系统的硬件特性，也依赖于应用问题的存储访问特性。如访问是顺序或非顺序，是否有体冲突，等等。关于这方面的优化方法，已有很多讨论[1,3,7,8]，这里不再赘述，下面我们主要讨论局部性的影响。另一方面，根据第三章中的公式，一个并行应用问题在分布存储并行机系统上的实际性能依赖于比例因子(S'_m、S'_d、S'_c和S'_{cd})。较大的比例因子引起加速比的降低。这些比例因子不仅仅依赖于应用问题相应存储访问量与计算量之比，也依赖于应用问题中的相应存储访问与计算的可重叠性。这里我们主要从局部性、存储访问量与计算之比值和重叠性，广义瓶颈原则，广义均衡性等方面讨论分布存储并行机系统上的优化原则，并给出曙光天潮分布存储并行机系统[5]的应用实例。

4.1 广义瓶颈原则

按Amdahl定律，一个问题的运算速度不是由其最快部分，而是由其最慢部分(瓶颈)决定，因而分析一个应用问题性能的关键是找出最耗时部分，加以并行化和优化，才可能达到较好的效果，这里我们不仅要从计算角度去分析，也要从供数速度分析，这包括存储访问、通信、磁盘I/O等等。即整体分析各级供数量和计算量的关系，找出广义瓶颈点、加以优化。这里我们给出一个计算量占主导的计算实例。

高能物理中格点规范理论的数值模拟包含大规模的数值计算[13]，这一计算的基本思想是将连续时空离散化为一个四维超立方点阵，实际计算中格点体系大小直接决定了计算量的大小，若作大规模数值模拟计算，整个应用问题的计算量将非常庞大，按第2节中的方法分析，此问题的计算量占整个问题的主导地位，因此计算部分的并行化是提高性能的关键。由于计算中格点计算之间的相关性质：即任一点的计算只与超立方体中"相邻"点之值有关，可采用红-黑格法[14]将整个格点分类，同类格点的计算是可并行的，通信只产生于不同类格点的计算之间，并仅需进行"边界"交换，这类方法的优点：1)并行计算量与原串行计算量"相容"[11]；2)通信次数和通信量都较小，并保持了局部性，即通信中产生于相邻节点之间。更进一步，重新安排计算次序、即先计算需要在处理机间进行数据交换的"边界"点，以便于通信与计算重叠。曙光天潮D2000分布存储并行机系统实际计算结果也说明此方法的并行效率较高，以墙钟计算，在处理机个数增加时，并行加速效率在90%以上[13]。

4.2 均衡性

"负载平衡"是保证并行计算性能的一个重要手段，这里的"负载平衡"往往是指各计算节点上计算时间的均衡。事实上，各节点计算时间不仅仅由计算量决定，也由供数速率决定，即由内存访问性质、通信等等因素决定，因此我们这里的均衡性既包括计算量的均衡，也包括各节点计算中的存储访问、通信、访盘等的相对均衡。在曙光天潮D2000上合成孔径雷达成像的试算就是这样一个实例。该计算包含大量的二维FFT计算，原用户采用的方法是对二维FFT的并行化。二维FFT的并行化计算由三步组成：X-方向的多个一维FFT的并行计算；二维数组的转置；Y-方向的多个一维FFT的并行计算。从计算量方面来看，该计算达到了"负载平衡"。但由于二维数组的转置包含了较多的各节点间的数据通信量、且不均衡，因而效率不高[7]。针对问题特性，我们作出以下改进，一是改原来的"横向分割"为"纵向分割"，二是不对二维FFT本身并行，而是根据计算问题的物理性质，采用被分割的数据块之间的并行。这样一来，在保证计算均衡性的基础上，不仅减少了通信量，也改进了通信的均衡性。试算结果表明新方法的并行效率在74%到97%之间[13]。

4.3 局部性

局部性是一个相对的概念，这里是指Cache访问

相对于内存访问的局部性，多机系统中本地内存访问相对于对其它节点内存访问(通信)的局部性，本地内存访问相对于访盘的局部性，本地磁盘访问相对于共享磁盘访问的局部性，等等，即各层存储访问相对其上层存储访问的局部性，也就是应尽量实现靠近CPU层次的存储访问，以提高整体效率。这里我们给出一个大I/O的计算问题的实例。

美国NCAR的气候模式ccm3.6[12]是一个全球大气环流谱模式，该模式可进行气候模拟和预测研究。因考虑到对不同计算机系统的可移植性，本模式提供了多种选项供用户选择，其中一个选项是根据计算机系统本身所带内存大小，选择将模式运行的中间结果放在磁盘(out core)或内存中(in core)，由于减少了内存和磁盘数据的交换，后者可提高整体运行速度。实测结果表明，整体运行速度的提高在19%与25%之间(见表1)。

表1 磁盘访问对运算速度的影响

np	T1	T2	(T1−T2)/T1(%)
1	76092	61500	19.18
2	45558	34800	23.61
4	27120	21240	21.68
8	17628	13380	24.10
16	12210	9090	25.55
32	8478	6780	20.03
64	5460	4260	21.98

表中，np：处理器个数，T1：中间结果放在磁盘的情况下，ccm3.6一个实例的整体运行时间，T2为中间结果放在内存储器的情况下，同样计算的运行时间。时间以秒为单位。

4.4 存储访问相对于计算量的比例

减少层次存储访问量相对于计算的比例是提高应用软件性能的重要手段，第4.2节也是减少通信/计算比例，提高并行应用软件性能的一个例子。这里我们给出减少存储/计算比例，提高应用软件性能的一个例子-矩阵乘法。一般从计算复杂性和空间复杂性来分析，两个大小为n×n的矩阵相乘，所需算术运算量为O(n^3)，存储空间为O(n^2)。存储/计算比例为1:n。在一个实际计算机系统上，该计算可通过下述三重循环完成：

```
do 10 j=1,n
do 10 j=1,n
do 10 k=1,n
      C(i,j)=C(i,j)+A(i,k)*B(k,j)
10 continue
```

若n相当大，则对每一个(i,j,k)循环进行一次乘加计算的同时，需要两次访问内存储器，事实上造成了存储/计算比例为1:1，极大地影响了计算性能。上述计算有多种优化方法，其中之一即分块算法。在采用较合适子块大小的情况下[8]，能真正实现存储/计算比例为1:n。我们在SGI Power 4D/240上对矩阵乘法的分块算法和常用的中积算法进行了比较，性能改进在20%左右，实算结果见表2。

表2 存储/计算比例对运算速度的影响

n	T_1(S)	T_2(S)	$(T_1-T_2)/T_1$(%)
256	6.15	5.58	9.27
512	57.14	43.82	23.31
1024	454.28	353.61	22.16
2048	4173.50	3235.44	22.48

n：矩阵阶数；T_1：矩阵乘法的中积算法运行时间；T_2 分块矩阵乘法的计算时间。

4.5 重叠性

这里包括存储访问与计算的重叠性，通信与计算的重叠性，访盘与计算的重叠性，和各级存储访问间的重叠性，这里我们给出一个重新安排计算次序以便于通信与计算重叠，从而提高计算效率的一个例子，Linpack是应用广泛的一种数值计算方法。在分布存储式并行计算机系统上，往往采用分块算法以达到负载平衡并提高存储访问效率。设待解方程组为Ax=b，其中A是n×n矩阵，x,b是n×1向量。设n=m*q，在该方法中，A被分割成列块，即$A=(A_0,A_1,\cdots,A_{q-1})$，其中$A_0,A_1,\cdots,A_{q-1}$为n×m矩阵。为达到负载平衡，往往采用"卷帘"式存储方案。若计算中实际采用的子块长为nb，则计算由以下子步组成。设有一置换阵P，使得

$$PA=\begin{pmatrix}A_{00} & A_{01}\\ A_{10} & A_{11}\end{pmatrix}=\begin{pmatrix}L_{00} & \varnothing\\ L_{10} & I\end{pmatrix}*\begin{pmatrix}U_{00} & U_{01}\\ \varnothing & \tilde{A}_{11}\end{pmatrix}$$

这里L_{00}是(nb×nb)单位下三角阵，U_{00}是(nb×nb)上三角阵，则有：

(1)$L_{00}*U_{00}=A_{00}$；(2)$L_{10}*U_{00}=A_{10}$；(3)$L_{00}*U_{01}=A_{01}$；(4)$\tilde{A}_{11}=A_{11}-L_{10}*U_{01}$。

该计算是易并行的，需注意的是，在该分割方式中，L_{10}往往驻留在一个计算节点中，但其值被所有节点所需用于(4)的计算中。在(4)能真正进行计算前，L_{10}应广播到参与(4)计算的所有计算节点中。于是，当n很大时，将有相当长的通信等待时间。为避免这一现象的发生，我们将(1)与(4)的计算相结合，将L_{10}的求解与$\tilde{A}_{11}$的修正相结合，即每次先作一个带有nb个右端项的nb阶方程组求解(L_{10}的部分求解)，结果广播到需要该值的计算节点，并修正$\tilde{A}_{11}$的一个条块。如此反复，直到修正完全部$\tilde{A}_{11}$。在矩阵阶很大的情况下，对$\tilde{A}_{11}$一个条块的修正可与下一个广播通信相重叠，

从而缩短了总处理时间，提高了计算效率[13]。

结论 本文采用"双流"分析方法描述了串、并行计算机系统的性能特征，用"降速因子"和"比例因子"的概念描述了应用软件性能相对于计算机峰值速度的降低程度，给出了分布存储并行机系统"广义加速比"的一个计算方法，并给出并行应用软件的若干优化方法和实例。实践说明该分析方法与优化原则对并行应用软件的优化是十分重要且可行的。

参考文献

1 Cremonesi P, Rosti E, et al. Performance evaluation of parallel systems. Parallel Computing, 1999, 25(4): 1677～1698

2 Culler D E, Singh J P, Gupta A. Parallel Architecutre, A hardware/software approach. San Francisco, Morgan Kaufmann, 1999

3 SGI Inc. Power challenge technical report. Silicon Graphics Inc, Mountain View, CA. Tech Rep Pwr CHALL (9617) 1996

4 Amdahl G M. Validity of the single-processor approach to achieving large scale computing capabilities. In: Proc AFIPS 1967 Spring Joint Computer Conf. Vol. 30, Atlantic City, New Jersey, April 1967. 483～485

5 Patterson D A, Gibson G, Katz R H. A case for redundant array of inexpensive disks (RAID). In: ACM SIGMOD conference, Chicago, Illinois, 1988. 109～116

6 Alpern B, Carter L. Towards a model for portable parallel performance: exposing the memory hierarchy. In: workshop on portability and performance for parallel processing, Southhampton, England, Wiley. July 1993

7 Qiao Xiangzhen, et al. Cache optimization in scientific computing. ACM Symp. on Applied Computing '99. San Antonio, Texas. 1999. 548～552

8 乔香珍. 并行计算时间模型和并行机系统性能. 计算机学报, 1998, 21(5): 413～418

9 NCIC. 曙光2000超级服务器. 北京曙光信息产业公司, 2000, 1

10 Sun, Gustafson J L. Toward a better parallel performance metric, Parallel Computing, 1991, 17(2): 1093～1109

11 Hockney R W, Jesshope C R. Parallel Computers, architecutre, programming and algorithms. Bristol, UK. Adam Hilger Ltd, 1988

12 Kluzek E B. User's Guide to NCAR CCM3. 6: [NCAR technical report] 1996

13 NCIC. 曙光2000超级服务器测试报告. 北京曙光信息产业公司, 2000, 1

（上接第23页）

为2的幂时，CP2算法有可能会超过 Bisection 算法（因为 Bisection 的二分法在这时对工作量的分配比较不均，而 CP2算法不受 P 值的影响）。如表2所示。

综上所述，在较多的情况下（如表中打勾的情况），CP2算法的效率要好于 Bisection 算法。而且，CP2算法求解时间较短的优势随着 DVE 世界中 Cell 数目的增大会更加明显，在大尺寸 DVE 的情况下这更具有实际意义。

·CP2算法的另一特点 Bisection 的每次执行都需要较长时间，且每次执行会重新分配所有的 Cell，相应地也就需要重新建立所有客户与服务器之间的连接，因此如果用来在系统运行时动态分区的话，会引起较长的延迟，所以它更适合于事先估计 Avatar 分布，在静态执行分区划分之后再运行 DVE 系统的情况。而 CP2算法另一优点则是可以逐渐得到更好结果，它每执行一代竞争的时间很短，每次只需在各服务器之间调整少量 Cell，非常适合于在 DVE 系统运行的时候不断地执行，随着 Avatar 在 Cell 之间的移动而动态地调整划分。所以 CP2算法比 Bisection 算法要更加适合于 Avatar 分布长期变化的虚拟环境系统。

参考文献

1 潘志庚，姜晓红，张明敏，石教英. 分布式虚拟环境综述. 软件学报, 2000, 11(4): 461～467

2 Lui J C S, et al. Dynamic Partitioning for a Distributed Virtual Environment. Department of Computer Science & Engineering, The Chinese University of Hong Kong

3 Fei Chan Ming, Yan Oldfield So King, Shing Peter Tam Tsz. DVE partition under uniform and non-uniform avatars distribution. Feb. 1998. Department of Computer Science & Engineering, The Chinese University of Hong Kong

4 史忠植. 高级人工智能. 第十一章 P249