基于高性能微机群集的可扩展性的研究与设计

祝永志 田 甜
(曲阜师范大学计算机科学学院 日照 276826)

摘 要 可扩展性是并行计算系统的重要性能指标,虽然异构系统越来越普遍,但对其可扩展性的研究还很少。给出了一种既适合同构并行计算系统又适合异构并行计算系统的效率的定义,根据该定义对可扩展性进行了分析,得出了既适用于同构系统又适用于异构系统的等效率模型,并根据开销比得出了在某一效率常数保持一致的情况下系统规模和工作负载的变化情况。最后通过实验进行了分析,结果表明该模型可以对效率和可扩展性进行较好的评测,并能预测并行计算系统的高可扩展性。

关键词 并行计算,性能,加速比,等效率,高可扩展性

Design and Implementation of Scalability Based on High Performance PCs Cluster

ZHU Yong-zhi TIAN Tian
(College of Computer Science, Qufu Normal University, Rizhao 276826, China)

Abstract Scalability is a important concept for analyzing the performance of parallel computing systems. Heterogeneous systems have become more common, but few study for heterogeneous system's scalability. This paper first presented a definition of efficiency that can be fit for homogeneous parallel computing systems and heterogeneous parallel computing systems. According to this definition, this paper also presented a iso-efficiency model that can be applied to both heterogeneous systems and homogeneous systems. From this we can analyze the change circumstance of the system scale and the question scale in which the efficiency maintains consistent. Finally, the paper presented some experiments, the results show that this model can estimates preferably the efficiency and high scalability of parallel computing systems.

Keywords Parallel computing, Performance, Speedup, ISO-efficiency, High scalability

1 引言

并行计算技术的飞速发展,使得越来越多的科学计算问题能够通过并行程序设计得以解决。但在基础理论研究等众多领域,工作负载越来越大,需要更多的计算资源,所以并行计算系统必须随之扩展,以提高计算能力[1]。然而系统的效率并不是随着节点数目的增加而呈线性增长,当系统到达一定规模时会出现效率降低、执行时间难以预测等问题,在异构系统中这些问题更为严重。目前可扩展性研究主要集中在并行算法与并行系统相结合的可扩展性上[2],即研究如何随节点数目的增加而扩展工作负载,使得执行时间较合理且效率较高。

本文主要从效率方面分析同构、异构并行系统的可扩展性[3],在分析了现有的可扩展性模型的特点及其不足的基础上,研究了系统如何随着处理机数目的增长而扩展工作负载,使扩展前后的效率保持不变,并给出了一种新的可扩展性模型,试验证明使用该模型的并行计算系统具有高可扩展性。

2 并行计算系统可扩展模型

本文所指的并行计算系统是指由并行算法、并行处理机以及并行计算环境所构成的系统。并行计算系统的可扩展性是指系统随着处理机数目的增多使系统的计算性能得以增长的特性[4-8]。

可扩展性是设计并行算法和高性能并行机所追求的一个重要目标。可扩展性分析很重要,算法设计者能用其分析算法,使增加的处理器得以有效利用。同时可用其估计取得最佳加速比或其他性能参数,如 T_p 与 $P(T_p)^r$[9]等最佳处理器数。

在分析并行系统时,经常会碰到的一个概念就是工作负载,一个传统的定义是将工作负载定义为一个描述输入规模的参数,比如在矩阵乘法中,如果矩阵为 $n\times n$,那么工作负载就是 n。这种定义方法的缺点是当面对不同的问题时,对工作负载需要作出不同的解释。

工作负载的定义应该是,对所有的问题,工作负载加倍意味着需要执行的操作数也变为原来的两倍。因此,我们选择解决问题所需要的基本操作的总数量作为工作负载的定义。采用这个定义,$n\times n$ 的矩阵乘法的工作负载就是 $\Theta(n^3)$,而 $n\times n$的矩阵加法的工作负载是 $\Theta(n^2)$。为了使工作负载的定义对于给定的问题唯一,我们定义一个问题的工作负载为在

到稿日期:2010-01-12 返修日期:2010-04-22 本文受山东省高等学校重点实验研究项目(编号:2005-400)和曲阜师范大学校科研项目(编号:XJ0734)资助。

祝永志 教授,硕士生导师,主要研究方向为网络与分布式系统。

单处理器上解决这个问题的最优串行算法的基本计算步骤的数目。此处所说的最优串行算法是指已知的性能最好的串行算法。

2.1 等效率可扩展模型

尽管可扩展性很重要，但目前还没有一个公认的评判标准，通常从3个不同角度进行可扩展性度量，即等效率、等速度和平均延迟方法。在等效率模型中，并行运行时间可以表示成工作负载、额外开销函数和处理器数目的函数，即：

$$T_p=\frac{T_e+T_0(W,p)}{P}=\frac{W+T_0(W,p)}{p}$$

则加速比可以表示成：

$$S=\frac{W}{T_p}=\frac{pW}{W+T_0(W,p)}$$

而效率可以表示成：

$$E=\frac{S}{p}=\frac{W}{W+T_0(W,p)}=\frac{1}{1+T_0(W,p)/W}$$

式中，T_e，W 表示工作负载，T_0 表示总的并行开销时间，p 表示处理器数目。

根据上面的效率函数，如果工作负载保持不变，当 p 增加时，总的额外开销会随着 p 的增加而增加，效率会降低。如果处理器数目保持不变，增大工作负载，那么对可扩展的并行系统，效率会提高，这是因为对固定的处理器数目 p，额外开销函数的增长速度要比 $\Theta(W)$ 慢，对这些并行系统，增加 p 时，可以通过增大工作负载 W 的方法来得到需要的系统效率。

对不同的并行系统，为了得到稳定的效率，当处理器数目增加时，W 必须以不同的速率相应增加。比如，在某些情形下，当处理器数目增长时，为了保持效率不变，W 需要以 p 的指数函数的速率增长。这样的系统其可扩展性很差，因为当使用大量的处理器时，并行系统很难得到良好的加速比，除非采用巨大的工作负载。另一方面，为了保持效率不变，如果相对 p 的增长，工作负载只需要以 p 的线性函数形式增长，那么这样的系统具有较高的可扩展性，这是因为当处理器的数目增加后，只需要采用合理的工作负载就可以得到较好的加速比。

对于可扩展的并行系统，效率可以维持在一个固定的值，只要 T_0/W 保持不变。设 E 为所需要的并行系统效率，根据前面的效率表达式，有：

$$E=\frac{1}{1+T_0(W,p)/W}\Leftrightarrow\frac{T_0(W,p)}{W}=\frac{1-E}{E}$$

$$W=\frac{E}{1-E}T_0(W,p)$$

令 $\lambda=E/(1-E)$，当给定效率后，这是一个常数。因为 T_0 是关于 W 和 p 的函数，则上面的表达式可以写成：

$$W=\lambda T_0(W,p)$$

这样，可以通过代数变换的形式，将 W 表示成 p 的函数，这个函数描述了为保持效率不变，当 p 改变时所需要的工作负载，我们把这个函数称为一个并行系统的等效率函数，这个等效率函数确定了一个并行系统为保持效率为常数而使加速比随处理器数目成比例的增长的容易程度，它描述了并行系统的可扩展能力。一个小的等效率函数意味着当处理器数目增长时，只需要较小的工作负载增长就可以达到处理器的充分利用，对应的并行系统当然具有很高的可扩展能力。相反的，一个大的等效率函数对应着一个可扩展性很差的并行系统。

按照等效率函数的定义，对于某一并行算法（或并行程序），为了运行效率保持不变，随着处理数目的增加，若只需增加较小的工作量（即加大工作负载），比如说 W 随 p 成线型或亚线型增长，则表示该算法具有良好的可扩展性；若需增加非常大的工作量，比如 W 随 p 成级数增长，则表示该算法是不可扩展的。对不可扩展的系统来说，等效率函数不存在，因为对这种并行系统，当处理器数目增加时，不管如何增加工作负载，都不可能使系统的效率保持为常数。

等效率函数揭示了并行算法和并行机结构共同影响下的计算性能。另外，等效率函数也说明为了维持效率是一个常数，当节点数变化时，工作负载也会相应地变化。然而等效率可扩展性评价准则把机器数 p 作为一个参数，显然不适合异构系统。

2.2 异构可扩展模型

异构系统中各节点的计算能力不同，因此需要对每个机器的计算能力进行计算能力的度量。常用的表征各节点的计算能力的方法有两种。

①用每秒中能完成的浮点操作数来表征各个节点的计算能力[10]，即“绝对计算能力”，然而该方法难以准确地预测浮点操作数目；

②选用标准节点[11]，其它节点的计算能力用相对于该标准节点的处理速度来表示，也称“相对计算能力”。

不同的标准节点的选取可以导致不同的性能模型。若采用系统中最快的节点作为标准节点，当系统中加入更快的节点时，则原来建立的模型将要做出调整。有的模型要求必须采用一个不受此系统的节点作为标准节点，这样在加入其它的节点时，原来的模型调整度不大。然而，标准节点的选取不受该系统的限制是没有必要的。本文给出的性能模型取消了这一限制，标准节点可以任意选取。

我们考虑这样的异构计算系统：它含 N 个节点，至少有两个节点的计算能力不同。指定某节点作为标准机，其他节点和整个系统的计算能力在此节点的基础上确定。假设某算法 A 的工作负载为 W，不考虑 I/O 时间的情况下，它在标准机上执行的时间 $T^r=WT_\theta$，其中，T_θ 表示执行单个浮点操作的平均时间，理论上是一常数。若 A 在第 i 个节点上执行的时间为 T^i，我们给出如下相关定义。

定义 1 计算系统中第 $i(0\leqslant i\leqslant N-1)$ 个节点的相对处理速度为 $V_i=\frac{T^r}{T^i}$。

定义 2 计算系统中所有节点的总计算能力用 V 表示，则 $V=\sum_{i=0}^{N-1}V_i$。对同构计算机群，总计算能力 $V=N\cdot V_i$。

定义 3 效率等于最优执行时间和实际执行时间的比值，即 $E=\frac{W/\sum_{i=0}^{N-1}V_i}{T_\phi}=\frac{W}{T_\phi\cdot V}$，其中实际执行时间 $T_\phi=\max_{i=0}^{N-1}T_i$，$V$ 的值依赖于系统结构及算法。V 也可以用另一种形式来度量，即表示成工作负载和时间的比值 $V=\frac{W}{T}$。因此 $T=\frac{W}{V}$ 可以表示最优执行时间。对同构系统有：

$$E=\frac{W}{T_\phi\cdot\sum_{i=0}^{N-1}V_i}=\frac{W}{T_\phi\cdot N\cdot V_i}=\frac{T^r}{T_\phi\cdot N}$$

定义 4 给定一个异构并行系统 $S(N,V,W)$，它有 N 个节点，总的计算能力为 V，总工作负载为 W，并给定另一个异构系统 $S'(N',V',W')$，其中 $V'>V$，如果当系统由 S 扩展到 S' 时，总能找到一个合适的 W'，使得 S 和 S' 的效率 E 保持一致，则称 S 是可扩展系统。

依据定义 3，如果其他参数不改变，只增加工作负载 W，则系统的效率会增加，反之则会降低。另外系统的最后反应时间取决于最慢的那个节点的执行时间，这和负载均衡算法有着必然的关系。本文旨在对可扩展性进行理论研究，在此假定负载均衡合理，有关异构系统负载均衡的问题将另文阐述。

依据上述定义，有关并行计算系统可扩展性的结论及其证明如下：

定理 1 给定一个异构系统 $S(N,V,W)$、一个扩展系统 $S'(N',V',W')$ 和一个固定的、可任意分解的工作负载 $W=(w_1,w_2,\cdots,w_N)$，如果工作负载相对各节点的计算能力被合理分配到各节点并且各节点开销时间为一常数，即 $T_i^0=C_0$，$i=1,\cdots,N$，如果 $W'=W\cdot\frac{V'\cdot C_0(N')}{V\cdot C_0(N)}$ 成立，则系统扩展前后两系统效率相同。

证明：要保持扩展前后效率 $E=E'$，即

$$\frac{W}{T_\phi\cdot V}=\frac{W'}{T'_\phi\cdot V'}\Leftrightarrow W'=W\frac{T'_\phi\cdot V'}{T_\phi\cdot V}$$

由于 $T_\phi=\max_{i=0}^{N-1}(T_i^0+T_i^\gamma)$，$T_i^0=C_0$，即 $T_\phi=\max_{i=0}^{N-1}(C_0+T_i^\gamma)$。由于我们假定负载均衡合理，因此节点上的计算时间 T_i^γ 的值可以表示成总工作负载 W 和总计算能力 V 的比值，故实际执行时间 $T_\phi=\max_{i=0}^{N-1}(C_0+W/V)$，可表示成 N 的函数即 $T_\phi=C_0(N)$。同理 $T_\phi'=C_0(N')$，所以 $W'=W\cdot\frac{V'\cdot C_0(N')}{V\cdot C_0(N)}$。

证毕。

定理 2 设因系统规模大小不同导致的各节点开销时间不同且分别为 $T_i^0=C_0+C_1N$，即节点的固定开销时间加上由于系统规模大小不同导致的开销时间，要保持系统扩展前后效率不变，则必须满足下面的条件

$$W'=W\frac{C_0(N')\cdot V'+C_1N'\cdot V'}{C_0(N')\cdot V+C_1N\cdot V}$$

证明：要保持扩展前后效率 $E=E'$，即

$$\frac{W}{T_\phi\cdot V}=\frac{W'}{T_\phi'\cdot V'}\Leftrightarrow W'=W\frac{T_\phi'\cdot V'}{T_\phi\cdot V},$$

由于 $T_i^\gamma=W/V$，因此，

$$T_\phi=\max_{i=0}^{N-1}(T_i^0+T_i^\gamma)=\max_{i=0}^{N-1}(C_0+C_1N+T_i^\gamma)$$

$$=C_0(N)+C_1N$$

同理 $T_\phi'=C_0(N')+C_1N'$，所以有

$$W'=W\frac{(C_0(N')+C_1N')\cdot V'}{(C_0(N)+C_1N)\cdot V}$$

$$=W\frac{C_0(N')\cdot V'+C_1N'\cdot V'}{C_0(N)\cdot V+C_1N\cdot V}$$

证毕。

定理 3 除了由于系统规模大小不同导致的开销时间差异外，开销时间还与各节点及各节点对应的工作负载有关。设各节点的开销时间为 $T_i^0=C_0+K_1\frac{W}{V}V_i$，即开销时间等于固定开销时间加上工作负载带来的开销时间。要使扩展前后效率不变，则必须满足：

$$W'=W\frac{C_0(N')V'+C_1W\max_{i=0}^{N'-1}V'_i}{C_0(N)V+C_1W\max_{i=0}^{N-1}V_i}$$

证明：与定理 2 的证明相类似，因为

$$T_\phi=\max_{i=0}^{N-1}(T_i^0+T_i^\gamma)=\max_{i=0}^{N-1}(C_0+K_1\frac{W}{V}V_i+T_i^\gamma)$$

$$=C_0(N)+C_1\frac{W}{V}\max_{i=0}^{N-1}V_i$$

同理推得，$T_\phi'=C_0(N')+C_1\frac{W}{V}\max_{i=0}^{N'-1}V_i'$

要保持扩展前后效率不变，则

$$W'=W\frac{T_\phi'V'}{T_RV}=W\frac{(C_0(N')+C_1\frac{W}{V}\max_{i=0}^{N'-1}V'_i)V'}{(C_0(N)+C_1\frac{W}{V}\max_{i=0}^{N-1}V_i)V}$$

$$=W\frac{C_0(N')V'+C_1W\max_{i=0}^{N'-1}V'_i}{C_0(N)V+C_1W\max_{i=0}^{N-1}V_i}$$

证毕。

定理 4 如果可扩展并行计算系统节点开销时间由 3 部分组成：

a) 每个节点的固有开销时间或者初始化时间 C_0；

b) 由于系统规模导致的开销时间 C_1N；

c) 由于工作负载导致的开销时间 $C_2\frac{W}{V}V_i$。

要使扩展前后效率不变，则必须满足：

$$W'=W\frac{(C_0+C_1N')V'+C_2WV'_{max}}{(C_0+C_1N)V+C_2WV_{max}}$$

$$=W\frac{C_0V'+C_1N'V'+C_2WV'_{max}}{C_0V+C_1NV+C_2WV_{max}}$$

证明：由条件可知，各节点的开销时间为：

$$T_i^0=C_0+C_1N+C_2\frac{W}{V}V_i$$

令 $C_0(N)=C_0+C_1N$，$\max_{i=0}^{N-1}V_i=V_{max}$，则由定理 3，得：

$$W'=W\frac{(C_0+C_1N')V'+C_2WV'_{max}}{(C_0+C_1N)V+C_2WV_{max}}$$

$$=W\frac{C_0V'+C_1N'V'+C_2WV'_{max}}{C_0V+C_1NV+C_2WV_{max}}$$

证毕。

3 实验分析及结论

PC 机群包括 1 个服务节点和 24 个计算节点。节点配置情况如下：

N_0-N_{15} 节点：

CPU：Intel Pentium 4 2.66GHz

内存：DDR 256M

硬盘：7200R 40G

$N_{16}-N_{23}$ 节点：

CPU：奔腾 D 3.0 GHz(2048kB 全速二级缓存)

内存：DDRII，512 MB

硬盘：80 GB SATA2(7200 转)

网卡：Realtek RTL8169

交换机：Cisco Catalyst 2950

消息传递接口采用 MPICH2。整个系统采用 VC 语言开

发[12,13]。

3.1 相对处理速度的确定

在同构环境下，各节点的权重值相等。在异构环境下，节点的权重值通过相对处理速度来确定。根据 2.2 节中的定义，它是某一固定算法在节点上的执行时间和此算法在标准机上执行时间的比值。在本文所构建的实验环境下，各节点的权重值确定如下。

单个任务在标准机上的执行时间为 T^s，本文任意取了一台机器(编号为 1)测得 $T^s=0.3312$ 秒(由执行 1024 次任务的总时间除以 1024 获得)，在其他机器上测得时间、相对处理速度及其权重值(之和为 1)如表 1 所列。由于每次实验所用节点数不同，故执行每次实验时的权重值会有所不同，但其总和为 1。

表 1 实验测得的各节点相对处理速度

节点编号	单个任务执行时间 T^s(秒)	相对处理速度	权重 w_i
1	0.3312	1.0000	0.0615
2	0.2903	1.1409	0.0539
3	0.3014	1.0989	0.0559
4	0.4231	0.7828	0.0785
5	0.2986	1.1092	0.0554
6	0.4017	0.8245	0.0746
7	0.3731	0.8877	0.0693
8	0.2869	1.1544	0.0533
9	0.2856	1.1597	0.0530
10	0.4315	0.7676	0.0801
11	0.3562	0.9298	0.0661
12	0.3578	0.9257	0.0664
13	0.2996	1.1055	0.0556
14	0.3651	0.9071	0.0678
15	0.3022	1.0960	0.0561
16	0.2829	1.1707	0.0525

3.2 可扩展性实验

实验的目的如下：

(1)检验本文效率模型能否同时适合同构系统和异构系统；

(2)检验传统的等效率模型不能适用于异构系统；

(3)检验模型能够预测计算机群系统的可扩展性。

在 2.2 节中把各节点的开销时间定义为：$T_i^0=C_0+C_1N+C_2\frac{W}{V}V_i, i=0,\cdots,N-1$，其中 N 为系统规模即节点数量，W 为工作负载即任务数，V 为系统的总计算能力，V_i 是当前节点的相对处理速度。根据上述对实验内容的分析可知，这 4 个参数都能够量化，而实际执行时间 T_i^0 可以由程序执行过程中取得，因此，C_0，C_1，C_2 可以由实验测得。在本文的实验环境下，执行程序后测得 $C_0=0.3$，$C_1=2.4$，$C_2=3.3$。所以节点开销时间可以表示成 $T_i^0=0.3+2.4N+3.3\frac{W}{V}V_i$，因此可以直接计算出节点上任务的执行时间，取各节点执行时间的最大值即可得到任务的执行总时间。

因此，可扩展模型也可以变为：

$$W'=W\frac{0.3V'+2.4N'V'+3.3WV'_{\max}}{0.3V+2.4NV+3.3WV_{\max}}$$

并行计算实验环境确定后，V，V'，$V_{\max}$，$V'_{\max}$ 可以根据 N，N'确定，因此公式共有 4 个未知数，即 W，W'，N，N'。

上面关系式就是扩展前后保持系统效率不变的方程，确定 W，N，N'的值后，可以求得 W'的值。因此本文从理论上可求得系统扩展后工作负载应扩展到多少，才能使效率保持不变。为了验证理论的正确性，下面将提供实验数据及分析。

基于简化负载均衡分配，在各个 Client 程序中设置了一个标准大小的任务 T(本文选取两个固定阶的矩阵相乘)，Server 节点向各节点发送所要执行的任务数 n(任务 T 执行的次数)。

本实验依据各节点的权重值(见表 2)对任务数进行分配。比如结点数为 4、任务数为 1024 时，在同构环境下，因为节点计算能力相同，所以 4 个节点按权重(0.25，0.25，0.25，0.25)平分任务，即 1024(256，256，256，256)。在异构环境下，如 4 个节点的相对处理速度为(0.2，0.3，0.1，0.4)、任务数为 1024 时根据权重值进行分配，即 1024(205，307，102，410)，对不能整除的部分四舍五入，但要保证任务全部被分配给计算节点。程序运行时会自动获得节点的权重值。

下面是实验获得的数据及分析。

表 2 同构系统中分别使用传统模型和本文异构模型的执行时间、效率

节点数	任务数	实际执行时间(秒)	E1(%)	E2(%)
2	2	1.021	0.613	0.622
	4	1.674	0.780	0.801
	8	3.216	0.784	0.803
	16	7.028	0.893	0.898
	32	12.156	0.897	0.901
	64	25.223	0.908	0.902
	128	52.652	0.913	0.903
	256	102.43	0.915	0.909
	512	201.357	0.916	0.912
	1024	401.121	0.917	0.916
4	4	1.165	0.598	0.601
	8	2.106	0.724	0.762
	16	3.086	0.798	0.803
	32	7.033	0.824	0.835
	64	11.276	0.896	0.901
	128	25.117	0.905	0.903
	256	51.035	0.908	0.910
	512	102.02	0.915	0.914
	1024	198.725	0.916	0.917
8	8	1.312	0.432	0.442
	16	2.134	0.698	0.701
	32	3.471	0.805	0.802
	64	7.335	0.813	0.816
	128	13.116	0.823	0.832
	256	27.840	0.897	0.901
	512	52.316	0.902	0.905
	1024	103.431	0.906	0.907
16	16	1.805	0.465	0.465
	32	2.765	0.527	0.523
	64	4.132	0.665	0.655
	128	9.103	0.678	0.677
	256	15.032	0.834	0.841
	512	27.138	0.869	0.866
	1024	54.609	0.876	0.878
	2048	112.523	0.902	0.909
	4096	206.128	0.913	0.914

表 2 是在同构环境下测得的数据，提供了在不同的节点数和工作负载下的执行时间、使用传统等效率模型得出的效率 E_1、使用本文的异构效率模型得出的效率 E_2。其中 E_2 计算时使用异构环境下的效率计算方法。由数据可以看出，在同构环境下，使用传统的等效率模型和使用本文的异构可扩展模型，其效率是大体一致的，因此可以说本文的模型适用于同

构环境。图 1 是使用传统模型在同构环境下所得的效率曲线。

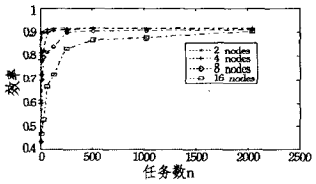

图 1　传统模型在同构环境下的效率曲线

表 3 的部分数据取自表 2，其中理论任务数由 3.2 节中的公式计算得出。它显示了在同构环境下保持效率在 0.9 左右、节点数从 2 增加到 16 时需要增加到的工作负载。节点为 24 时，是对任务数、执行时间做出的预测，保持效率为0.897，任务数应达到 3135.2。另外可以看出实际和理论任务数是相吻合的，因此证明了本文可扩展模型的有用性和有效性。

表 3　指定效率为 0.9 时的同构计算机群规模和工作负载

节点数	实际任务数	理论任务数	实际执行时间	E1(%)	E2(%)
2	16	16	6.763	0.898	0.902
4	64	67.2	13.324	0.901	0.904
8	256	267.7	26.132	0.905	0.906
16	1024	1031.1	53.026	0.901	0.903
24	——	3135.2	105.327	——	0.897

图 2 是在异构环境下采用传统的效率模型的效率曲线，可以看出效率是不连续的。因此在异构环境下，传统的效率模型不适用。

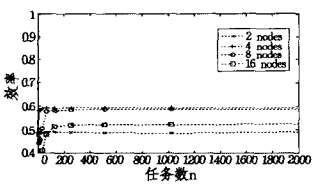

图 2　传统效率模型在异构环境下的效率曲线

表 4 是在异构环境下测得的数据，可以看出采用传统等效率模型计算出的效率 E_1 和 E_2 是不一致的，也说明传统的等效率模型不适用于异构环境。另外表 4 还表明了在异构环境下保持效率在 0.9 左右不变时需保持多大的计算机群规模和工作负载，节点数为 24 时，任务数、执行时间以及效率是由模型预测所得到的。由实际和理论任务值相吻合可以证明本文可扩展模型可以预测异构环境的可扩展性。图 3 是在异构环境下使用本文异构模型所得的效率曲线，与同构环境下使用传统模型得到的效率曲线相比较，可以看出两者都具有较好的连续性，因此可以认为模型能够较好地应用于异构系统。

表 4　指定效率为 0.9 时的异构计算机群规模和工作负载

节点数	实际任务数	理论任务数	实际执行时间	E1(%)	E2(%)
4	16	16	7.23	0.429	0.903
6	64	69.4	14.15	0.587	0.905
12	256	268.1	24.67	0.608	0.908
24	——	1109.6	50.73	——	0.911

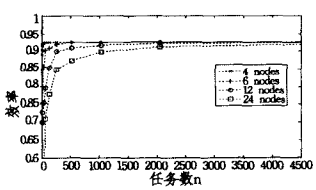

图 3　异构环境下使用异构模型所得的效率曲线

本实验首先根据可扩展模型的理论及具体实验环境，得出了使扩展前后效率保持不变的公式，从理论上对工作负载如何扩展进行了量化。接着实施了具体实验，用于验证理论的正确性，结果显示理论和实践是相吻合的。因此，系统规模扩展时，本文的可扩展模型可以指导工作负载应如何扩展，并能够预测执行时间，达到了预测系统可扩展性的目的。

结束语　本文阐述了并行计算系统可扩展性理论问题，指出了传统可扩展模型在异构系统中的瓶颈，建立了既适合同构系统又适合异构系统的等效率模型。该模型能直观地反映并行系统的可扩展性，能定量地研究与分析等效率可扩展问题，能全面地反映算法、处理机以及计算环境对可扩展性的影响。

本文的实验分析结果表明，该等效率可扩展模型对并行计算系统的性能研究具有很好的指导意义。

参 考 文 献

[1] Hossfeld F. Teraflops computing: a challenge to parallel numerics[C]// ACPC'99. 1999:1-12

[2] Dongarra J. High performance computing, numerical kernels, and performance measurement[C]// HPC-Asia'2000. Beijing, May 2000

[3] Sterling T, Becher D J, Savarese D. Beowulf: a parallel workstation for scientific computation[C]// 1995 International Conference on Parallel Processing. 1995

[4] Chen Jun, Li Xiao-mei. A practical scalability metric[C]// The HPC Asia'2000. Singapore, 2000

[5] Xun S. Scalability versus execution time in scalable systems [J]. Journal of Parallel and Distributed Computing, 2002, 62(2): 173-192

[6] Grama A, Gupta A, Han E. Parallel algorithm scalability issues in petaflops architectures [EB/OL]. http://www.cs.umn.edu/kumar, 2003/2007

[7] Sun X, Rover D. Scalability of parallel algorithm-machine combinations [J]. IEEE Trans, Parallel and Distributed System, 1994, 5(6): 599-613

[8] Ji Yong-chang, An Hong, et al. A scalability metric for algorithm-machine on NOW and MPP[C]// Proc. HPC Asia'2000. 2000: 405-407

[9] Alan H K, Horace P F. Measuring parallel processor performance [J]. Communications of ACM, 1990, 33(5): 539-543

[10] Barbosa J, Tavanes J, Padilha A J. Linear algebra algorithm in a heterogeneous cluster of personal computers [C]// 9th Heterogeneous Computing. Workshop(HCW'2000). Cancun, Mexico, May 2000

[11] Clematis A, Corana A. Modeling performance of heterogeneous parallel computing systems [J]. Parallel Computing, 1999, 25(9): 1131-1146

[12] Gropp W. Using MPI-2: Advanced Features of the Message-Passing Interface [M]. America: MIT Press, 1999

[13] Wilkinson B, Allen M. 并行程序设计[M]. 陆鑫达，等译. 北京：机械工业出版社，2002: 280-320