计算机研究与发展
Journal of Computer Research and Development
ISSN 1000-1239/CN 11-1777/TP
43(7):1233~1239, 2006

EDCP——通用于任务复制资源分配算法的复制有效性检查过程

杨 娟[1,2] 白 云[1,2,3] 邱玉辉[1,2]
[1](西南大学计算机与信息科学学院 重庆 400715)
[2](西南大学智能软件与软件工程实验室 重庆 400715)
[3](西南大学电子信息工程学院 重庆 400715)
(hopper@swnu.edu.cn)

EDCP—A Duplication Checking Process Used in Duplication Based Resource Allocation Policies

Yang Juan[1,2], Bai Yun[1,2,3], and Qiu Yuhui[1,2]
[1](*Faculty of Computer and Information Science, Southwest University, Chongqing* 400715)
[2](*Intelligent Software and Software Engineering Laboratory, Southwest University, Chongqing* 400715)
[3](*School of Electronic and Information Engineering, Southwest University, Chongqing* 400715)

Abstract Optimizing the cost function of the resource allocation policy in heterogeneous systems is an important way to improve the systems' concurrency processing capability. Duplication based resource allocation policy has been proven to be effective in dealing with the data driving workloads. However, most of these policies overlook the trade offs between duplication cost and the incomes brought by it. The lack of an effective mechanism to balance these two aspects eventually hampers the performance improving of the whole system. EDCP, an effective duplication checking process is a checking process which can be generally used in any duplication based resource allocation policies under heterogeneous environment. EDCP can optimize the cost function by preventing the useless duplications without breaking the systems' whole benefit.

Key words cost function; resource allocation; effective duplication; data driving; workload

摘 要 在异构系统中最优化资源分配策略的代价函数是提高分布式系统自治并发处理性能的一个重要途径.基于复制的资源分配策略作为主流策略虽然在异构系统中可以对数据驱动的任务流资源分配上取得很好的效果,但由于大部分的基于复制的资源分配算法都忽略了任务复制本身耗费与其复制所取得的效益之间的平衡比较,因此产生了大量的无效复制,最终影响系统整体性能的提高.针对这种情况,提出了EDCP,可通用于异构系统中基于任务复制的资源分配策略的有效性复制检查过程.EDCP通过对复制的有效性检测,只复制能对系统性能有所提高的任务,减少了通信开销,并且在保证了系统整体效益的前提下最优化了代价函数.

关键词 代价函数;资源分配;复制有效性;数据驱动;任务流

中图法分类号 TP316.4

1 引 言

异构环境下的并发处理已经成为分布式系统研究领域的一个重要课题,因为各种各样的应用系统(如天气模拟计算、实时系统中的数据流控制、图像处理等)都潜在地要求在这样的背景下进行代价函数的构建.因此问题最终演变成了如何提高系统

收稿日期:2005-03-29;修回日期:2005-10-11

进行自治并发处理的能力. 系统自治并发性除了可以通过硬件性能改进提高,最主要的改进还是通过对所处理任务流的合理资源分配来实现.

资源分配的最优化问题根据所针对的任务流类型的不同[1]可分为事件驱动(event driving)和数据驱动(data driving)两大类. 事件驱动型的任务流其特点主要体现在任务流间没有数据依赖性[2,3],针对这种类型的资源分配策略主要指实时系统中的任务流控制. 由于任务间相对独立,因此,可通过限制满足条件构建其代价函数,并在解空间上应用各种启发式搜索策略来实现任务流的均衡负载[4]. 而数据驱动任务流则指任务间存在数据依赖关系,任务流可用 DAG 图准确描述[5~7],且数据驱动任务流的资源分配不仅仅局限于实时系统,其优化问题覆盖整个分布式系统. 因此,针对 DAG 图的资源分配成为资源分配问题的研究重点. 在此基础上构建的代价函数(cost function)也不局限于传统的最小化整体响应时间[2,3,6,7],还包括在保证任务流 QOS 限制(如实时系统中数据流从输入到输出必须要小于的时延限制[1])的基础上最大化系统容错性[8,9]以及最大化系统吞吐量[5]等. 而应用于数据驱动任务流的资源分配策略也成为资源分配领域的主流策略,主要有:

基于权重的分配策略[2,10,11]. 这种方法将优先级赋给各个任务,并根据优先级进行资源分配,典型的算法是文献[2]中提出的 MMP 算法,但它也存在致命的问题. 例如:MMP 运用的前提是必须知道任务数为 λ 时其时间执行的上下限,而它们则受任务流性质的影响,无法准确确定. 另外,当任务流发生变化时,MMP 无法在保障系统整体利益最大的前提下最小化系统整体响应时间. 因此在此基础上产生了基于权重的复制策略[12],通过任务的复制保证了每次任务流的资源分配总是可以保证系统的整体利益,且最优化代价函数.

基于聚类的分配策略. 其主要思想是将一类在关系上相互依赖紧密的任务定位到同一个处理器上,这样可以有效地减少通信开销. 单独的聚类算法由于其使用情况单一已逐渐不再被使用,替代的则是基于聚类的任务复制策略[6,7,13]和扩散收敛策略[3]. 基于聚类的任务复制策略其主要思想是将相互依赖紧密的任务定位到同一处理器上后,再判断处理器的空闲情况而进行任务的复制. 这种方法的目的主要在于通过减少通信开销来提高整体效益. 基于聚类的扩散收敛策略则是通过收敛因子进行邻接节点间的任务传递控制,当收敛因子收敛时则说明系统整体已取得最大效益. 这种方法虽然通过收敛因子的设置最终可以达到负载平衡,但这种方法存在的最大问题在于无法模型化每个节点发送和接受的任务量,且这种方法无法判断任务间的数据依赖关系,因此不适用于数据驱动的任务流资源分配.

综上所述,基于权重和基于聚类的资源分配策略是最基本的两种资源分配方法,而将上述两种策略与复制结合起来则可以避免在复杂的解空间里搜索最优解(这是个 NP 完全问题[14]). 因为资源分配被分成了两个阶段,第 1 个阶段在初步应用基于权重或聚类的策略后得到一个近似最优解,第 2 个阶段则在此基础上通过任务复制在有限的解空间中得到最优解. 由于第 1 个阶段是传统的资源分配过程,因此大部分的基于复制的资源分配策略都将重点集中在了这个阶段中的任务的初始定位上,而忽略了第 2 阶段中任务复制的效率问题,即任务复制的低效率对系统整体性能的影响[6,7,13,15~19]. 认为只要进行可能的复制就可以提高整个系统的收益,而缺乏一个平衡任务复制所产生的耗费与其复制后可能会获得的收益之间关系的有效机制. 文献[12]虽然在复制过程中加入了复制有效性检查,但它只针对应用于同构系统的资源分配策略. 因此针对这个问题,我们提出了一种适用于异构系统下的所有基于复制的资源分配策略的复制有效性检查过程 EDCP. 无论基于复制的任务调度技术采用的是什么策略,由于它始终可以被分割成两个封闭的过程,即通过一个输入,产生一个输出,因此只需要在第 2 个阶段中加入复制有效性检查机制,且满足封闭特性就可以替代基于复制的资源分配策略的第 2 个阶段中的无差别复制过程来解决这个问题. EDCP 正是这样一个满足了这个特征且加入复制有效性判断机制的过程,它通过输入近似最优解序列,在传统的复制基础上进行复制有效性判断,输出一个复制次数最少,且 Makespan 最小的最优解序列. EDCP 的通用性体现在可以替代各种基于任务复制任务调度策略中的第 2 阶段,通过复制有效性检查提高复制的效率,达到真正改进整体效益的目的.

2 有效性复制检查过程(effective duplication check processing, EDCP)

2.1 定义一个 DAG 图

给定一个 DAG 图,用 6 元组〈T,F,E,P,τ,

C)来对其进行描述. 其中 T 为任务节点集合,$T=\{t_i \mid i\in[1,n]\}$,$F()$为映射函数,且 $F(t_i)=\delta_{t_i}$,用于将任务 t_i 映射到 t_i 本身被串行化后所产生的数据流量. t_i 所要进行的复制是处理器间的,因此其本身必然会产生由 t_i 性质所决定的不同大小的数据量. F 的主要作用是体现出不同 t_i 间的数据量差异. E 为表示节点间数据驱动关系的边的集合,$E=\{\langle t_i,t_j\rangle \mid t_i\in T,t_j\in T\}$,$(t_i,t_j)$表示任务节点 t_j 的执行依赖于 t_i 所产生的数据. P 是网络中可用的处理器集合,$P=\{p_i \mid i\in[1,m]\}$. $\tau=\tau(t_i,p_j)$,$t_i\in T$,$p_j\in P$ 表示任务节点 t_i 在 p_j 上执行的运行耗费. C 是一个 $m\times m$ 的对称矩阵,用于表示处理器间不完全连接的网络环境中处理器间的单位通信耗费.

$$C=\begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1m} \\ c_{21} & c_{22} & \cdots & c_{2m} \\ \square & \square & & \square \\ c_{m1} & c_{m2} & \cdots & c_{mm} \end{bmatrix},$$

$$c_{ij}=\begin{cases} 1, & p_i \text{ 和 } p_j \text{ 直接相连,且 } i\neq j, \\ f(p_i,p_j), & p_i \text{ 和 } p_j \text{ 间接相连,且 } i\neq j, \\ 0, & i=j, \end{cases}$$

p_i 和 p_j 是相互可达的,$f(p_i,p_j)$映射从 p_i 到 p_j 的最短路径上的耗费和.

2.2　相关参数描述

相关参数描述如表 1 所示:

Table 1　Description of the Relative Parameters

表 1　相关参数描述

Relative Parameters	Description
t_i	$t_i\in T,i\in[1,n]$, represents the task nodes.
p_i,p_j,p_g	$p_i,p_j,p_g\in P,i,j,g\in[1,m]$, refers to the available processors.
$\tau(t_i,p_j)$	The executing time of t_i on p_j.
$c(p_i,p_j)$	Unit communication cost between p_i and p_j at the given data flow.
d_{t_i}	Data flow produced by t_i after it being processed.
δ_{t_i}	Data flow produced by t_i during its replicating process.
$EST(t_i,p_j)$	The earliest possible executing time of t_i on processor p_j.
$DAT(t_i,p_j)$	The moment when all the predecessor data needed by t_i has arrived at p_j.
$MIIP(t_i)$	The bottleneck predecessor node of t_i which means $MIIP(t_i)$ is the most influencing node to decide the EST of t_i.
$G^S_{p_j},G^F_{p_j}$	The start time and the finish time of the first free slot on p_j which satisfied the constraint ②.

2.3　要满足的有效性复制约束条件

若任务 t_i 要从 p_o 复制到 p_j,则首先应满足下列约束条件:

①$EST(t_i,p_j)+\tau(t_i,p_j)+c(p_j,p_g)d_{t_i}<EST(t_i,p_o)+\tau(t_i,p_o)+c(p_o,p_g)d_{t_i}$.

p_o 表示 t_i 初始定位的处理器,p_j 表示 t_i 可能会进行复制目标处理器,p_g 表示 t_i 运行后产生的数据所要到达的目的处理器. 限制条件①指针对任务 t_i 的某个给定后继任务 t_j 来说,若 t_j 当前定位于处理器 p_g 上,则 t_i 复制后其产生的 t_j 所需的数据到达 p_g 的时刻要比复制前有所提高. 但这并不表示 t_j 的最早开始时刻会提前,因为 t_j 除了受多个前驱任务影响外,还受其空闲块开始时刻的影响. 即使 t_i 复制后使得它产生的数据到达 p_g 的时刻有所提前,但仍不保证它为有效复制. 其中:

$EST(t_i,p_j)=\max\{DAT(t_i,p_j),G^S_{p_j}\}+c(p_o,p_j)\delta_{t_i}$,且有

$EST(t_i,p_o)=\max\{DAT(t_i,p_j),G^S_{p_j}\}$,因为 $c(p_o,p_j)\delta_{t_i}=0$.

②$G^F_{k,p_j}-G^S_{k,p_j}\geqslant\tau(t_i,p_j)+c(p_o,p_j)\delta_{t_i}$.

限制条件②表示,若要将 t_i 从 p_o 上复制到 p_j,那么 p_j 上至少要有一个空闲块支持其传输和执行的时间. 若 p_j 上的第 k 个空闲块是顺序找到的第 1 个满足条件的空闲时间块,则这个空闲块要满足限制条件②.

2.4　有效性复制检查算法(effective duplication checking algorithm, EDCA)

所有的基于复制的资源分配算法(无论是优先级算法还是聚类算法),都必然会经历将 DAG 转化为一个有序的任务序列集合及为这个任务序列挑选最合适的处理器两个阶段. 第 1 个阶段的结果为产生一个近似最优解集合,通过各种启发式算法将任

务节点定位到可能最适合的处理器上,最坏的情况是 $\varepsilon=\{(t_{k_1},p_r),\cdots,(t_{k_n},p_r)\}$,$k_i\in[1,n]$,$r\in[1,m]$即所有的任务节点都定位在同一个处理器上顺序排队执行(串行),而最好的情况则是已经成为了最优解,其 Makespan 最小. 第 2 个阶段是通过将处理器上等待执行的任务复制到其他有空闲时间槽的处理器上并发处理来实现产生最优解的目的. EDCP 是一个加入了复制有效性判断机制的过程,它通过输入一个近似最优解序列,在传统的复制基础上进行复制有效性判断,输出一个复制次数最少,且 Makespan 最小的最优解序列. EDCP 的通用性体现在可以替代各种基于任务复制任务调度技术中的第 2 个阶段. 下面引入 EDCP 中的主体算法 EDCA,现假设 DAG 已被转换成为一个有序对集合:$\varepsilon=\{(t_{k_1},p_{r_1}),\cdots,(t_{k_n},p_{r_n})\}$,$k_i\in[1,n]$,$r_i\in[1,m]$即已对任务集合中的每个任务节点进行了初始的处理器分配. ε 产生的依据可能是基于权重的关键路径策略[16,20],也可以是基于聚类的插入产生策略[13]. 对该集合中的每一个任务 t_{k_i} 执行有效性复制检查算法:

```
D_Check (t_i, p_j)
{start: A = find EST(t_i, p_j);
 M_i = MIIP(t_i); /* (1) */
 EST'(t_i, p_j) = {(A, p_o)}; /* (2) */
   for all p_i ∈ P and p_i ≠ p_o
    {if (M_i, p_i) satisfy the constraint ①
      {EST*(t_i, p_j) = EST(M_i, p_i) + τ(M_i, p_i) + c(p_i, p_j)d_Mi;
      if EST*(t_i, p_j) < A
         (EST*(t_i, p_j), p_i) → EST'(t_i, p_j); /* (3) */;
      }}
  sorting the elements in setting EST'(t_i, p_j) as the ascending value of EST*(t_j, p_j);
  while (setting EST'(t_i, p_j) is not empty)
   {p^c_Mi = p_i in the first pair (EST*(t_i, p_j), p_i) of EST'(t_i, p_j);
    if p^c_Mi satisfy the constraint ② and p^c_Mi ≠ p_o
     {copy M_i to p^c_Mi;
      turn to start;}
    else if p^c_Mi = p_o
          return fail; /* (4) */
          else delete the pair in which p^c_Mi resides from the setting EST'(t_i, p_j);}}
```

(1) M_i is supposed to be originally allocated to processor p_o.

(2) $EST'(t_i,p_j)$ is a setting used to store the sequential pair $(EST(t_i,p_j),p_k)$, which means if the duplication of M_i from processor p_o to p_k can improve the t_i's EST, add pair $(EST(t_i,p_j),p_k)$ into setting $EST'(t_i,p_j)$.

(3) Adding pair $(EST^*(t_i,p_j),p_i)$ into setting $EST'(t_i,p_j)$.

(4) No effective duplication can be executed, return fail.

D_Check()算法主要为产生的任务及其初始分配的处理器序列进行任务复制的有效性检查,对于产生的 ε 来说,每个任务 t_i 都有其瓶颈前驱节点 $MIIP(t_i)$,*D_Check*()算法为这个任务找到满足复制约束条件的可能处理器,并从中挑选出可以使 $EST(t_i,p_j)$时刻提前最多的处理器作为 $MIIP(t_i)$的复制目标处理器. 这个过程反复执行,直到 $MIIP(t_i)$无法再通过复制使 $EST(t_i,p_j)$值提前为止. 在这个过程中,由于加入了两个约束条件的判断,任意一个条件不满足都可以将这个看起来可能会改进系统性能的复制拦截下来,从而提高复制的效率.

3 EDCP 应用于样例算法

3.1 TANH 和 TDS 算法

我们选用 TANH 算法作为应用实例算法,采用 TANH 算法的主要原因是它是最近性能最好的基于任务复制任务调度算法. TANH 算法是基于聚类的任务复制算法,是在初始聚类阶段对经典 TDS 算法进行改进了的任务复制算法,但它在任务复制阶段仍采用了 TDS 算法中的无差别复制. TANH 算法主要分为 3 个步骤:①计算每个任务节点 t_i 的 $EST(t_i,p_j)$,$MIIP(t_i)$等其他参数. ②产生初始聚类及任务序列. ③执行所有可能的复制. 其中第③步所执行的复制策略原理为:对于产生的初始聚类中的一个给定的任务节点 t_i,其任务序列中的前驱任务若不是它的瓶颈节点,就将这个节点开始的一直处于紧耦合状态的任

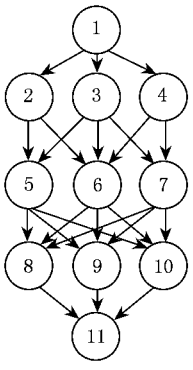

Fig. 1 Example DAG.

图 1 DAG 示例

务所组成的任务链复制到另一个处理器上. 反复执行这个过程,直到入口节点为止. 在这个复制过程中,由于复制缺乏有效性检查,很多时候复制的结果反而还会增大其 Makespan 值. 若任务本身由复制所产生的数据量 δ_{t_i} 与其运行后产生的数据量 d_{t_i} 处于同一个数量级时,无差别复制就会使整个系统的通信耗费迅速增长,并最终影响任务完成的实际时间. 图 1 为样例 DAG 图,图 2 则显示了 TANH 和 TDS 算法中的无差别复制过程.

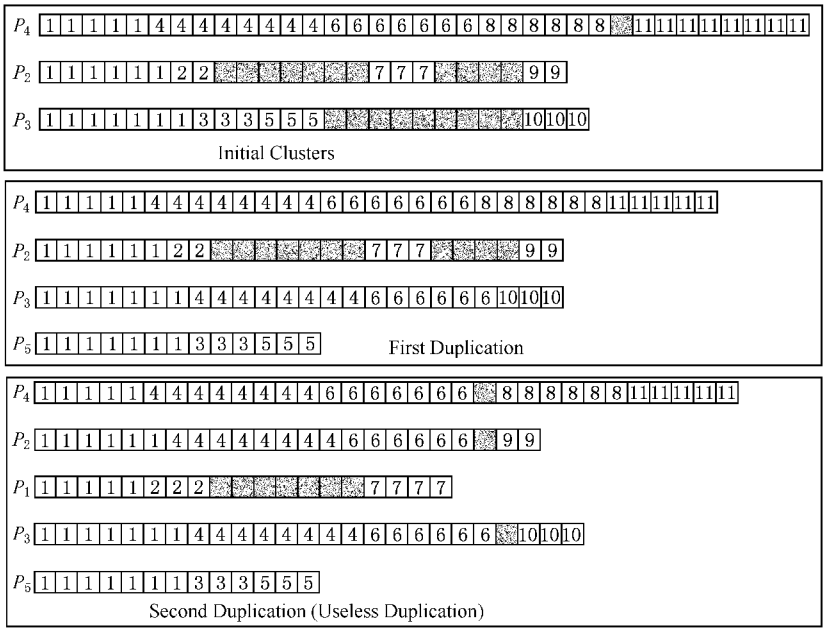

Fig. 2　Duplication process in TDS.

图 2　TDS 中的复制过程

3.2　用 EDCP 替代 TANH 中的复制过程

将 TANH 算法中的第③步用 EDCP 替代,为简单起见,我们也只考虑 TANH 算法中所假设的理想情况,即任务本身并不产生任何耗费,$\delta_{t_i}=0$ 且 d_{t_i} 也不对通信耗费产生影响,即 $d_{t_i}\equiv 1$,网络中处理器间均直接相连,即 $c(p_i,p_j)\equiv 2, p_i,p_j\in P$,对产生的任务序列对 $\{(11,p_4),(9,p_2),(10,p_3),(8,p_4),(5,p_3),(7,p_2),(6,p_4),(2,p_2),(3,p_3),(4,p_4),(1,p_4)\}$ 有如表 2 所示的结果：

Table 2　Applying EDCP on TANH

表 2　TANH 上应用 EDCP 的结果

t_i	*MIIP*	have passed EDC	Garget processor	Makespan
11	10	×	ø	32
9	6	×	ø	32
10	6	√	p_3	31
8	6	×	ø	31
5	3	×	ø	31
7	4	×	ø	31
6	4	×	ø	31
2	1	×	ø	31
3	1	×	ø	31
4	1	×	ø	31
1	ø	×	ø	31

当 EDCP 检查到 $MMIP(10)=6$ 时,发现当它复制到处理器 p_3 上可以缩短整体 Makespan,且此时为最优结果,因此以后的所有任务都无法通过有效性复制检查.

4　模拟实验

一个复制算法的效果可以用 ACT/ECT 比值来进行衡量(ACT—actual completion time, ECT—earlist completion time),在用 EDCP 替代传统复制过程后,我们同样可以通过 ACT/ECT 的比值来反映 EDCP 在资源分配算法中所起的作用,即在增加了算法复杂度的前提下,是否可以保证其最优性. 我们通过 *CCR* 的变化来观察 ACT/ECT 的变化情况. *CCR* = average edge cost/average task cost, *CCR* 值的变化表示了通信耗费与与任务运行耗费之间的关系.

将 EDCP 过程应用于一个随机产生的有 100 个节点的 Fork-Join 图,并与样例算法进行比较,因为 Fork-Join 图是构成实际任务 DAG 图的一个基本框架,因此可以通过 Fork-Join 图判断资源分配算法的最优性.

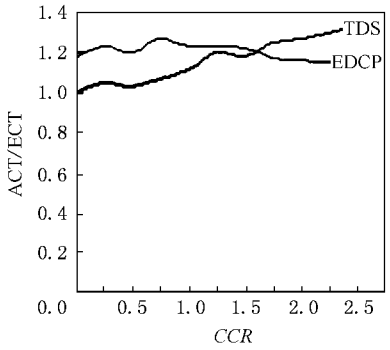

Fig. 3 Performance comparison TDS/EDCP.

图 3 TDS/EDCP 性能对比

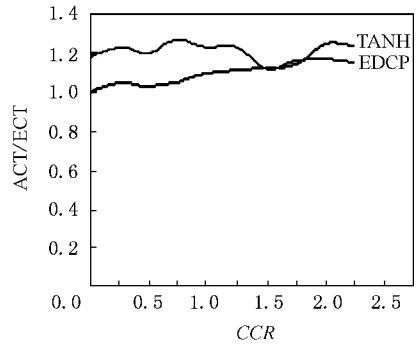

Fig. 4 Performance comparison TANH/EDCP.

图 4 TANH/EDCP 性能对比

从图 3 和图 4 中可以看出,当 CCR 值较大时,EDCP 所取得的 ACT/ECT 比值比 TDS 和 TANH 算法更趋近于 1,而当 CCR 值较小时,EDCP 仿佛在性能上还差于 TDS 和 TANH 算法. 这是因为当 $CCR \to 0$ 时,表示通信耗费$\ll$任务运行的耗费,即表示无效复制产生的通信耗费不会影响系统的整体性能提高,而有效性复制检查则增大了系统的计算耗费,因此当 CCR 很小时系统性能没有得到改进反而有所降低. 但这与我们设计 EDCP 的初衷不相矛盾,我们要解决的就是当通信耗费与任务执行耗费相当时任务复制的耗费与收益间的平衡问题. 因此当 CCR 增大时,用 EDCP 替代无差别复制后产生的结果明显比 TDS 和 TANH 算法有提高.

5 结 论

EDCP 是一个可应用于异构系统中基于任务复制资源分配算法的复制有效性检查过程,EDCP 通过对复制的有效性检测,只复制能对系统性能有所提高的任务,通过减少通信开销来达到在保证了系统整体效益的前提下最优化代价函数的目的. EDCP 的时间复杂度在最坏情况下为 $O(m \times n)$,即当处理环境为理想的全连接的 mesh 网络时. 但我们可以通过减少可用处理器数量来降低复杂度(例如,假设处理环境为非完全连接的,即每个节点的邻近节点数量远小于任务数. 这是分布式系统中最实际的网络环境,也是大多数基于聚类资源分配算法采取的策略). 第 4 节的实验也显示了 EDCP 对资源分配策略性能的改进.

参 考 文 献

1 AI-Jaroodi, Hong Jiang, David Swanson. A comparative study of parallel and distributed Java projects for heterogeneous systems [C]. In: Proc. 16th Int'l Parallel and Distributed Processing Symposium. Los Alamitos: IEEE Computer Society Press, 2002

2 Leonidas Georgiadis, Christos Nikolaou, *et al*. A fair workload allocation policy for heterogeneous systems [J]. Journal of Parallel and Distributed Computing, 2004, 64(1): 507~519

3 Tiberiu Rotaru, Hans-Heinrich Nageli. Dynamic load balancing by diffusion in heterogeneous systems [J]. Journal of Parallel and Distributed Computing, 2004, 64(4): 481~497

4 Muhammad Kafil, Ishfaq Ahmad. Optimal task assignment in heterogeneous distributed computing systems [J]. IEEE Concurrency on Complex Distributed Systems, 1998, 6(3): 42~51

5 Shoukat Ali, Jong-Kook Kim, *et al*. Greedy Heuristics for resource allocation in dynamic distributed real-time heterogeneous computing systems [C]. The 2002 Int'l Conf. Parallel and Distributed Processing Techniques and Applications (PDPTA '02). Las Vegas, NV, 2002

6 Binoy Ravindran, Peng Li, *et al*. Proactive resource allocation for asynchronous real-time distributed systems in the presence of processor failures [J]. Journal of Parallel and Distributed Computing, 2003, 63(12): 1219~1242

7 Samantha Ranaweera, Dharma P. Agrawal. A task duplication based scheduling algorithm for heterogeneous systems [C]. In: Proc. 16th International Parallel and Distributed Processing Symposium. Los Alamitos: IEEE Computer Society Press, 2002. 445~450

8 Ladislau Boloni, Dan C. Marinescu. Robust scheduling of metaprograms [J]. Journal of Scheduling, 2002, 5(5): 395~412

9 Shoukat Ali, Maciejewski, *et al*. Measuring the robustness of a resource allocation [J]. IEEE Trans. Parallel and Distributed Systems, 2004, 15(7): 630~641

10 Wang Yongyan, Wang Qiang, *et al*. A real-time scheduling algorithm based on priority table and its implementation [J]. Journal of Software, 2004, 15(3): 360~370 (in Chinese)
(王永炎, 王强, 等. 基于优先级别表的实时调度算法及其实现[J]. 软件学报, 2004, 15(3): 360~370)

11 G. C. Sih, E. A. Lee. A compile time scheduling heuristic for interconnection-constrained heterogeneous processors architectures [J]. IEEE Trans. Parallel and Distributed Systems, 1993, 4(2): 175～187

12 Savina Bansal, Padam Kumar, *et al*. An improved duplication strategy for scheduling precedence constrained graphs in multiprocessor systems [J]. IEEE Trans. Parallel and Distributed Systems, 2003, 14(6): 533～544

13 Rashmi Bajaj, Dharma P. Agrawal. Improving scheduling of tasks in a heterogeneous environment [J]. IEEE Trans. Parallel and Distributed Systems, 2004, 15(2): 107～118

14 R. L. Graham, L. E. Lawler, *et al*. Optimization and approximation in deterministic sequencing and scheduling: A survey [J]. Annals of Discrete Math., 1979, 5: 287～326

15 S. Darbha, D. P. Agrawal. Optimal scheduling algorithm for distributed memory machines [J]. IEEE Trans. Parallel and Distributed Systems, 1998, 9(1): 87～95

16 I. Ahmad, K. Yu-Kwong. On exploiting task duplication in parallel program scheduling [J]. IEEE Trans. Parallel and Distributed Systems, 1998, 9(9): 381～422

17 C. I. Park, Y. Y. Choe. An optimal scheduling algorithm based on task duplication [J]. IEEE Trans. Computers, 2002, l51(4): 444～448

18 I. Ahmad, Y. K. Kwok. On exploiting task duplication in parallel program scheduling [J]. IEEE Trans. Parallel and Distributed Systems, 1998, 9(9): 872～892

19 Y. C. Chung, S. Ranka. Application and performance analysis of a compile-time optimization approach for list scheduling algorithms on distributed-memory multiprocessors [C]. In: Proc. Super-Computing. Los Alamitos: IEEE Computer Society Press, 1992. 512～521

20 H. EI. Rewini, T. G. Lewis, *et al*. Task Scheduling in Parallel and Distributed Systems [M]. New Jersey: Prentice Hall, 1994

Yang Juan, born in 1979. Received her master's degree in 2004. Now Ph. D. candidate. Her main research interests include distributed computing, distributed AI, etc.

杨娟,1979 年生,硕士,博士研究生,主要研究方向为分布式计算、分布式人工智能等.

Bai Yun, born in 1978. Received his master's degree in 2003. Now Ph. D. candidate. His main research interests include distributed computing, MAS, Pervasive Computing, etc.

白云,1978 年生,硕士,博士研究生,主要研究方向为 MAS、分布式计算、普适计算等.

Qiu Yuhui, born in 1938. Professor and Ph. D. supervisor of the Faculty of Computer and Information Science in South-Western University, (SWU), China. Director of the Institute of Artificial Intelligence, SWU; Member of Professional Association; senior member of CCF. Vice-Director of Chinese Association of Artificial Intelligence; IEEE senior member; ACM professional member. His main research interests include discrete mathematics, compiler principle, logical foundation of artificial intelligence, expert systems, automatic reasoning, machine learning, DAI, mobile computing, etc.

邱玉辉,1938 年生,教授,博士生导师,西南大学人工智能研究所所长,国家教育部科学技术委员会信息学部委员,中国计算机学会高级会员,中国人工智能学会副理事长,美国 IEEE 高级会员,主要研究方向为非单调推理、近似推理、神经网络、机器学习和分布式人工智能的教学和研究工作.

Research Background

The reasonably allocating of the resources needed by the workloads in the application systems is the main method to improve the concurrency capability of the system. Duplication based resource allocation policy has been proven to be effective in dealing with the data driving workloads. However, the lack of an effective mechanism to balance these two aspects eventually hampers the improving of the whole system's performance. EDCP is a checking process which can be generally used in any duplication based resource allocation policies under heterogeneous environment.