一种基于动态规划的并行构件资源选择算法

李建江[1],崔 健[1],严 林[1],李福林[1,2]
(1. 北京科技大学信息工程学院计算机科学与技术系,北京 100083;2. 中国白城兵器试验中心,吉林白城 137001)

摘 要: 当前,多种硬件体系结构并存,传统的并行构件资源管理却不能充分利用并行构件的属性以适应底层硬件的多样性,可扩展性比较差.在研究基于CCA规范的并行构件执行集成平台的基础上,本文提出了一种层次化并行构件资源管理模型,并提出了一种基于动态规划的并行构件资源选择算法.实验结果表明,相对于通常使用的CPU频率优先、CPU Cache值优先与通信优先策略,本文提出的基于动态规划的并行构件资源选择算法不仅能够更加精确地选择计算节点资源,而且具有良好的可扩展性.

关键词: 并行构件; 资源管理; 层次化模型; 动态规划
中图分类号: TP316.4 **文献标识码:** A **文章编号:** 0372-2112 (2011) 04-0887-07

A Resource Selection Algorithm for Parallel Component Based on Dynamic Programming

LI Jian-jiang[1], CUI Jian[1], YAN Lin[1], LI Fu-lin[1,2]
(1. *Department of Computer Science and Technology, School of Information Engineering, University of Science and Technology Beijing, Beijing* 100083, *China*; 2. *Baicheng Ordnance Test Center of China, Baicheng, Jilin* 137001, *China*)

Abstract: At present, various hardware architectures coexist, traditional parallel component resource management can not fully take advantage of parallel component properties to meet the diversity of the underlying hardware. At the same time the scalability of traditional parallel component is rather poor. In this paper, the research based on the specifications of CCA proposes a hierarchical model for parallel component resource management, then this paper mainly proposes the resource selection algorithm based on dynamic programming. Experimental results show that related to the tactics of CPU frequency priority, CPU Cache value priority and communication priority, the resource selection algorithm based on dynamic programming can select the resource of computing node more precise, and has a good scalability.

Key words: parallel component; resource management; hierarchical model; dynamic programming

1 引言

并行构件是指可部署在并行计算平台的构件,它能够显著提高软件的可重用性与可维护性.在科学计算领域,随着问题复杂性及规模不断增大,传统并行软件的复杂性不断增加,开发周期延长.并且由于并行体系结构各异,造成程序移植困难且效率不能保证,现有软件很难被重用.因此,利用并行构件技术对大型科学计算并行系统进行高效、快速的开发具有重要的现实意义.

本文针对分布内存、共享内存(包括:SMP与多核等)硬件体系结构的不同特点,首先在研究基于CCA规范的并行构件执行集成平台的基础上,提出一种层次化并行构件资源管理模型.通过资源属性映射机制,随着系统硬件不断扩展,该模型能实现较优的资源选择.

其次,在层次化并行构件资源管理模型的基础上,本文重点提出了一种基于动态规划的并行构件资源选择算法,实现并行构件属性与特定资源之间的映射,从而保证并行构件能较好地发挥底层硬件的性能.

最后,结合分子动力学模拟软件XMD的特点,构建XMD分子动力学模拟辐照损伤程序的并行构件(使之适应于多硬件体系结构),进行实际测试、性能对比分析与优化.

2 相关工作

美国国家实验室(包括:Argonne与Lawrence Berkeley等)和大学(包括:Indiana与Utah等)联合创建了CCA[1]

收稿日期:2010-04-29;修回日期:2010-07-22
基金项目:教育部重点基金(No.108008);国家863高技术研究发展计划(No.2008AA01Z109);北京市教委重点学科计算机系统结构(No.XK100080537)

(Common Component Architecture)论坛,并提出了通用构件体系结构.Luc Courtrai 等人针对工作站集群,开发出 Concerto[2]软件平台,该平台可以支持并行构件的部署,通过动态改变构件行为可适应计算资源的变化.文献[3]定义和实现了 Concerto 平台的开放性与可扩展性框架.Pardico[4]项目在 OMG 组织的 CORBA 构件模型基础上实现并行构件,构造面向计算网格的构件模型.文献[5]通过 OVIS 项目,针对云计算的特点,从构件完好和资源利用的角度致力于收集和分析资源的度量,继而完成有利于应用需求的硬件资源分配.文献[6]针对高性能集群系统中容易出现的错误,提出了一种主动的分布式虚拟机结构策略来解决这些错误.K Uhlemann 等人针对高可用 HPC 任务和资源管理,提出了 JOSHUA[7]解决方案.JOSHUA 提供一个虚拟的同步环境,此环境使用基于 PBS 服务接口的外部复制.文献[8]针对自适应中间件平台提出一个动态资源管理框架.在该框架中,为了达到以一种更自然的方式获取资源的目的,计算资源能够直接再现.文献[9]提出了一种全局统一的层次化网格资源模型,该模型有效地屏蔽了广域网上资源的异构性,提高了资源的可扩展性.文献[10]提出新的基于组合双向拍卖的网格资源分配模型,并在此基础上提出新的资源分配及定价算法.文献[11]提出了一种灵活的移动节点间资源协作共享方案,包括基于可靠性理论的请求资源预测算法,以及基于排队论的理论层次型资源调度模型.文献[12]提出了基于线性规划的网格异构资源分配问题的建模和求解方法,同时提出了网格环境下对独立作业进行网格资源分配的网格服务架构.文献[13]使用模糊聚类方法,根据不同的性能指标要求,为不同应用选择不同的计算结点集合,这样可以明显改善应用性能.

上述研究主要针对的是并行构件体系结构规范、框架实现和计算网格资源管理.在计算网格中可以通过网格资源模型或中间件屏蔽资源的异构性,但是在并行构件研究中并没有提出有效的并行构件资源管理模型和方法,以适应底层硬件的多样性,充分发挥底层硬件的性能.同时,随着硬件系统的持续扩大,如何充分使用这些资源将会具有更重要的现实意义.为此,本文提出了一种基于 Ccaffeine[14]框架的层次化并行构件资源管理模型,同时在研究此层次化并行构件资源管理模型的基础上,提出了一种基于动态规划的并行构件资源选择算法来解决上述问题.

3 资源管理模型

3.1 并行构件执行集成平台

本文在基于 CCA 规范基础上衍生出一种框架,即并行构件执行集成平台(如图 1 所示),该集成平台由如下几个主要部分组成:Ccaffeine 框架、构件代理、资源管理、逻辑层、底层硬件.并行构件执行集成平台通过封装目前现有的并行构件规范,加入并行构件属性的形式化描述,使得在获得并行构件属性信息之后,能够为其选择合适的资源来执行.

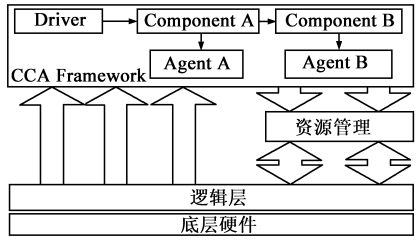

图1 并行构件执行集成平台

Ccaffeine 框架:用于高性能计算的一个轻量级 CCA 框架.Ccaffeine 框架基本实现了 CCA 规范的要求,相比其它的 CCA 扩展框架,其更加灵活.同时,Ccaffeine 能够在运行时通过 port 来动态连接不同构件,为并行构件的资源管理带来方便.

构件代理:为每个并行构件分别添加一个代理,构件代理提供了并行构件的执行要求.

资源管理:是并行构件执行集成平台的核心,资源管理的核心是一个高效的并行构件资源管理模型,该模型能够识别并行构件所运行的环境,并能基于并行构件的执行要求,利用基于动态规划的并行构件资源选择算法选取合适的资源来进行部署,从而在最大程度上缩短程序的运行时间.资源管理还包括并行构件的性能预测,并能够利用性能预测的结果对资源选择结果进行修正,进一步缩短程序的运行时间.

逻辑层:逻辑层的主要目的是屏蔽由异构 SMP 节点所组成的底层硬件,同时能够获得并行构件执行环境的资源部署信息,实现对并行构件的自动部署.

3.2 层次化并行构件资源管理模型

并行构件执行集成平台的资源环境层次结构可以分为站点、各站点下的计算节点和各计算节点所包含的具体资源.为了提高资源管理的效率,适应底层硬件的动态变化,本文基于并行构件执行集成平台的资源环境层次结构提出一种层次化并行构件资源管理模型.根据并行构件执行流程,我们把层次化并行构件资源管理模型分为两个层次,第一个层次为节点选择,第二个层次为本地资源选择.节点选择按照并行构件任务执行对资源的需求,为各个并行构件任务选择需满足特定条件(如:并行构件执行所需的进程数、并行构件执行开销等)的计算节点.本地资源选择能够细粒度地划分各个计算节点所接收的并行构件任务,并且根据适当的策略在计算节点内部合理、高效地进行调度.这种层次化并行构件资源管理模型一方面可以使资源

管理过程更加清晰,降低资源管理的复杂度,另一方面也能适应底层硬件系统的动态变化,提高并行构件执行集成平台的可靠性.

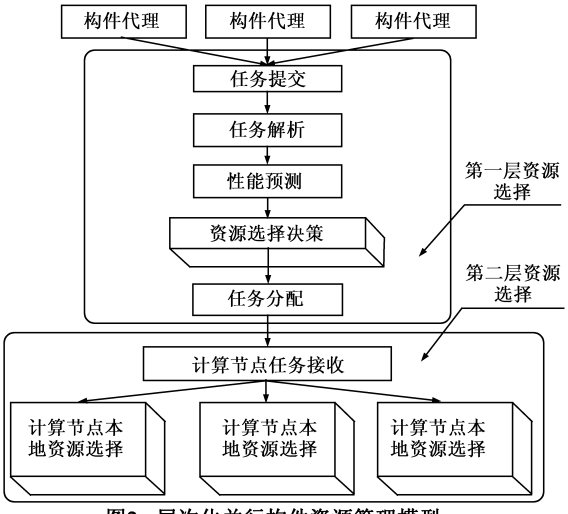

图2　层次化并行构件资源管理模型

图 2 给出了本文提出的层次化并行构件资源管理模型的体系结构.在图 2 中,第一层资源选择由任务提交、任务解析、性能预测、资源选择决策和任务分配组成.第二层资源选择由计算节点任务接收和计算节点本地资源选择组成.构件代理通过任务提交把并行构件任务提交给控制节点上的节点选择器.构件代理除了描述任务之外,还必须指定一些基本的任务要求,比如并行构件执行所需要的进程数等.节点选择器主要由任务解析、性能预测、资源选择决策和任务分配组成.任务解析主要对并行构件提交的任务进行解析,得出并行构件任务执行所需要的资源条件.性能预测主要是预测并行构件的性能,以便下一步能够利用性能预测的结果对资源选择结果进行修正.资源选择决策是第一层资源选择的核心,我们采用本文提出的 CNRS 算法来为每个并行构件任务选择满足要求的计算节点,提高并行构件的执行效率.任务分配负责将构件代理提交的任务分配到选定的计算节点执行.各计算节点内部的第二层资源选择主要由如图 1 中的逻辑层来处理.

4　并行构件的资源选择算法

4.1　并行构件的资源管理

图 3 显示了从资源识别到资源选择,再到最后并行构件部署的一个流程,其基本原理与过程如下:

(1)并行构件执行集成平台首先识别每个并行构件的属性说明书,并行构件属性说明书即为并行构件属性形式化定义的表现形式.本文使用与 Web 应用服务软件相类似的数据传输机制 XML,作为描述属性说明书的工具;

(2)逻辑层对识别到的运行环境资源信息进行编码处理,赋予其相应的 ID 值;

(3)资源管理利用基于动态规划的并行构件资源选择算法,根据收集到的并行构件属性值和运行环境的资源信息,选择合适的资源,同时会查询资源选择结果是否与性能预测反馈的结果冲突;

(4)若匹配结果不冲突,则直接将选择的资源信息传递给逻辑层;

(5)若匹配结果发生冲突,则修改并行构件属性说明书,用性能预测的结果修正并行构件原来的属性值;

(6)基于修改后的并行构件属性值,利用基于动态规划的并行构件资源选择算法重新进行资源的选择;

(7)将选择的结果与性能预测的结果再次比较;

(8)逻辑层自动完成并行构件在已选择资源上的部署.

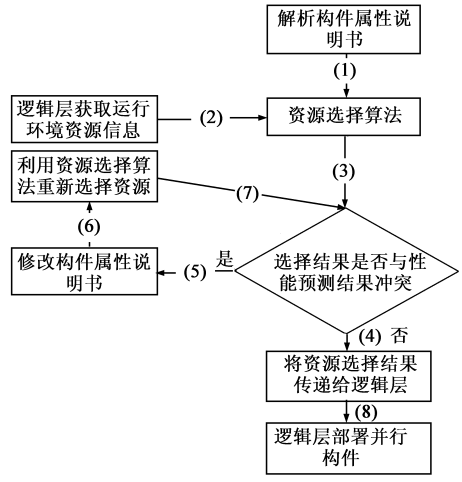

图3　资源管理流程图

4.2　基于动态规划的资源选择算法

资源选择主要是为每个并行构件任务选择满足要求的计算节点,提高并行构件的执行效率.基于所给定的层次化并行构件资源管理模型体系结构,本文提出了基于动态规划的并行构件资源选择算法,称之为计算节点资源选择(Computational Node Resource Selection, CNRS)算法.CNRS 算法根据每个并行构件任务的执行要求,为每个并行构件任务选择最能满足其执行需要的进程数和执行开销这两个衡量标准的计算节点.CNRS 算法的基本流程如下所示:

(1)构建任务集:各个构件代理提交任务之后,获得并行构件任务集 $T=\{t_1,t_2,\cdots,t_m\}$,m 表示并行构件任务的个数.

(2)分析并行构件任务集 $T=\{t_1,t_2,\cdots,t_m\}$ 中每个并行构件任务的执行条件:对于每个并行构件任务 t_i 均有一个条件向量 $\boldsymbol{TaskCond}_i(c_1,\cdots,c_k)$ 与之对应,每

个 ***TaskCond*** 有 k 个条件.这些条件可以是通信/计算比、要求的进程数 np 或 Cache 大小等.其中,通信/计算比(Communication-to-Computation Ratio,CCR)是由总通信时间除以总计算时间来得到.通信/计算比在并行构件属性说明书中的默认值为 0,即假设该应用程序部署在一个节点上,不需要参与通信,CCR 值可由用户指定.在执行并行构件任务时,CCR 越大,表明节点之间通信开销会越高.

(3)构建可用节点集:对于某个并行构件任务 t_i,如果所有计算节点的空闲 CPU 数之和大于或等于 np 且 Cache 大小、内存和硬盘等资源满足并行构件任务执行的要求,则由这些计算节点组成的集合就符合并行构件任务 t_i 的执行条件.在实际计算中,以上的几个属性都会和层次化并行构件资源管理模型中的性能预测结果进行交互,以获得更加准确的属性值.从而,最终得到一个满足并行构件任务 t_i 基本执行条件的计算节点集合 $nodeGroup\{node_1,\cdots,node_i,\cdots,node_k\}$.在 $node_i$ 中,i 表示此计算节点的节点号,$node$ 的值表示此计算节点内可用进程的个数.

(4)基于可用节点集,构造通信分组数组 $commGroup[N]$,N 为通信分组的个数.对于某个通信分组 $commGroup[n]=\{node_1,\cdots,node_i,\cdots,node_k\}$,考虑到网络状况的影响,节点之间的通信效率可能不稳定,因此,有必要构造出在某一时间段内通信状况较好的通信分组.具体方法是利用碰撞实验得出节点之间的通信时延.选中某一节点,该节点会发送数个信息单元到其它所有节点.其中响应时间 $respTime = recvTime - sendTime$,从而可以得到一条节点 i 关于响应时间的平滑曲线 $smoothCur_i$(如式(1)所示).

$$smoothCur_i=\frac{2*respTime_i+(i-1)*smoothCur_{i-1}}{i+1} \tag{1}$$

由式(1)可知:i 越大,$respTime_i$ 的平滑因子 $2/(i+1)$越小,那么说明近期加入的 $respTime_i$ 在整个平滑曲线中的权重越小,而已经存在的平滑曲线的权重因子$(i-1)/(i+1)$却越来越大.这样定义的目的,是为了保证 $smoothCur_i$ 的稳定性.$respTime_i$ 权重的缩小,$smoothCur_{i-1}$ 权重的放大,使平滑曲线受新加入的 $respTime_i$ 的影响相对较小,从而保证了平滑曲线 $smoothCur_i$ 的稳定性.如果所有节点的响应时间都相等,那么平滑曲线就是一条直线.如果新加入的节点响应时间较大,那么它偏离平滑曲线 $smoothCur$ 的距离会较远,从而可以迅速地反映出来.

在构建分组时,每个节点 $node_i$ 都会考虑自己与其它节点关于响应时间的平滑曲线,如果其中某一个节点 $node_j$ 的响应时间值与 $node_i$ 关于响应时间的平滑曲线偏离较多,那么 $node_i$ 就不会与 $node_j$ 分在同一组中.偏离阈值可以在并行构件属性说明书里面特意指定,也可以利用默认值进行筛选.默认值一般设定为平滑曲线的 10%,这是在实验过程中得到的比较合理的经验值.这时,将所有与 $node_i$ 通信不好的节点都挑选出来,并在这些挑选出来的节点中再次按照上述方法构造新的通信分组,直至所有的节点都被分组.并且每一个通信分组 $commGroup[n]$中,属于同一通信分组的所有计算节点的空闲 CPU 数之和大于或等于 np.另外,当通信分组数组中每一个通信分组的空闲 CPU 数的总和都小于并行构件任务 t_i 所要求的进程数 np 的时候,则需要合并某些 $respTime$ 相差较小的通信的分组,使其满足并行构件任务的要求,即每一个通信分组的空闲 CPU 数的总和都大于或者等于并行构件任务 t_i 所要求的进程数 np.

(5)评估节点的性能:调用评测函数(如式(3)所示)来评估每个节点的性能.假设 a_i 代表并行构件属性说明书中的通信/计算比,那么 a 就是所有并行构件任务综合的通信/计算比,可以通过式(2)来得到.接下来每个节点都会根据该 a 值来计算本节点的性能.本节点的性能还与 CPU 的频率、内存大小和 cache 大小等其它因素相关,如式(3)所示.

$$a=\sum_{k=1}^{m}a_i \tag{2}$$

$$prf_i=\left(\frac{m}{a+m}*\frac{CpuHz_i}{CpuHz_{\max}}+\frac{a}{a+m}*\frac{cacheSize_i}{cacheSize_{\max}}+\beta*\frac{memSize_i}{memSize_{\max}}\right)\times 100 \tag{3}$$

其中,prf_i 是第 i 个节点的评估性能,$CpuHz_i$ 为第 i 个节点的 CPU 主频,$CpuHz_{\max}$为集群中 CPU 的最高主频;$cacheSize_i$ 为第 i 个节点的 $cache$ 大小,$cacheSize_{\max}$为集群中 cache 大小的最大值;$memSize_i$ 为第 i 个节点的内存大小,$memSize_{\max}$为集群中内存大小的最大值;a_i 由构件代理根据并行构件的属性说明书给出;β 是考虑 $memSize$ 对程序性能影响的参数,默认为 0,除非用户在并行构件属性说明书里面特意指定;m 是计算节点的总数.

(6)选择合适的计算节点:本文提出的基于动态规划的并行构件资源选择算法是一个与背包问题类似的组合优化 NP 完全问题.如果选择一个节点,那么该节点上的所有可用进程都将被选择,因此,资源的选择与否就可以简化为一个 0/1 背包问题[15].考虑到不同通信分组之间的通信开销,本文的最优值搜索分别针对各个通信分组进行.然后在不同通信分组中选择式(4)的值最大者为最优值.下面是对这一问题的详细描述:

对于某一通信分组 $commGroup[n]=\{node_1,\cdots,node_i,\cdots,node_k\}$,假设 $commGroup[n]$里面的节点总数为 k,每个通信分组对应一个描述该通信分组内各个节

点是否被选择的一维向量 $\boldsymbol{x}=\{x_1,\cdots,x_i,\cdots,x_k\}$,$x_i\in\{0,1\}$,$1\leqslant i\leqslant k$,$proc_i$ 是该通信分组内每个节点的可用进程数,prf_i 是该通信分组内每个节点的评估性能,np 是构件任务 t_i 要求的执行进程数.则此问题可被描述如下(如式(4)和式(5)所示):

$$\sum_{i=1}^{k} prf_i * x_i \tag{4}$$

$$\begin{cases}\sum_{i=1}^{k} proc_i * x_i \leqslant np\\ x_i\in\{0,1\},1\leqslant i\leqslant k\end{cases} \tag{5}$$

式(6)显示了该问题的递归定义.定义一个二维数组 $f(i,j)$,$f(i,j)$指在 i 个节点上要求使用 j 个进程时,能使式(4)取的最大值.因而,式(6)中的 $f(k,np)$就是全局最优值.

$$\begin{cases}f(i,j)=\max\{f(i-1,j),f(i-1,j-proc_i)+prf_i*x_i\}\\ f_{all}=\sum_{i=1}^{k} prf_i*x_i,j=0\end{cases} \tag{6}$$

其中 $1\leqslant i\leqslant k$,$1\leqslant j\leqslant np$,$proc_i\leqslant j$

如果 $j=0$,则意味着用户已经选择了全部可用的节点进行计算,那么其最优解就是所有选择节点对应的 prf_i 的累加和.如果 $1\leqslant i\leqslant k$,$1\leqslant j\leqslant np$,则意味着本文提出的层次化并行构件资源管理模型会根据 CNRS 算法来选择最优解.

能够利用回溯方法[15]确定递归产生式的 x_i 值.由式(6)的递归产生式可得到一个解空间,这个解空间至少包含问题的一个解(可能是最优的).把解空间组织成树,然后按照深度优先的方法从起始节点进行搜索,且利用限界函数以避免移动到不可能产生解的子空间.则每个通信分组都对应于一个 $\boldsymbol{x}=\{x_1,\cdots,x_i,\cdots,x_k\}$,$x_i\in\{0,1\}$,$1\leqslant i\leqslant k$.然后,针对每个通信分组分别计算式(4)的值.在对应式(4)的值最大的通信分组中,$x_i=1$ 的节点即为本文提出的 CNRS 算法所选定的计算节点.

5　实验及性能分析

本文实验平台的底层硬件是由多个异构 SMP 系统组成的集群环境,以 XMD 分子动力学模拟辐照损伤程序的并行构件为测试对象.XMD 是美国康涅狄格州立大学 Jon Rifkin 编写的分子动力学模拟程序[16],通过改变粒子个数、迭代次数,可以改变其计算规模.在本文中,为其给定 4 种不同的计算规模,如表 1 所示.考虑到粒子个数的增加会导致 XMD 计算规模呈几何程度的增长,因而在设置粒子个数为 121 时,将迭代次数设置为 5 次.从后面的实验结果上来看,即便这样,Class D 的计算规模还是远大于其它的计算规模.

表 1　测试规模分类

参数设置 / 计算规模	粒子个数	迭代次数
Class A	31	100
Class B	51	100
Class C	81	100
Class D	121	5

根据部署并行构件执行的一般情况,在对比资源选择策略上,将本文提出的 CNRS 算法与一些常用的人为指定资源选择策略进行对比,这些策略分别是 CPU 频率优先、CPU Cache 值优先与通信优先.其中,

CPU 频率优先:在测试中手动选定 CPU 频率最高的节点作为运算资源.

CPU Cache 值优先:XMD 应用程序对 CPU Cache 的依赖程度很高.因而,在该测试中手动选择那些 CPU Cache 值最好的节点作为运算资源.

通信优先:是指尽可能地选择可用进程数最多的节点,尽可能将计算进程放置在一个节点中.即:手动选择剩余可用进程数最多的节点作为计算资源.

5.1　正确性测试

首先,验证本文提出的 CNRS 算法的正确性,以及验证本文提出的 CNRS 算法相对于通常使用的 CPU 频率优先、CPU Cache 值优先与通信优先策略的优点.在第一次执行程序时,由人工手动输入并行构件属性说明书.在多次执行之后,利用性能预测结果对并行构件属性值进行修正之后验证运行结果.如果 CNRS 算法选择一个节点,则选择该节点内的所有可用进程,这与通信优先策略类似,但又有所不同.CNRS 算法所选择的进程是经过选择的节点内的所有进程,而通信优先是所选择的进程可用进程数最多的节点内的所有进程.

实际测试结果(如图 4 所示)表明,根据初始化的构件属性说明书,CNRS 算法的选择结果与通信优先策略选择的结果一致(如图 4 中修改前测试结果所示).考虑到 XMD 程序的特性,随着迭代次数的增加,对 CPU Cache 值大小的要求将不断增加;反之,如果迭代次数较少的话,对 CPU Cache 值的要求就没那么高,从而优先考虑那些 CPU 频率较高的节点资源.所以,当计算规模为 Class D 级时,CPU 频率优先的策略选择结果明显优于其它策略的选择结果.

在层次化并行构件资源管理模型中,利用性能预测结果,在多次运行程序之后,能够针对构件程序的输入参数修正并行构件的属性参数,而 XMD 程序的相关输入参数就是粒子大小和迭代次数.

图 4 中修改后测试结果显示了 XMD 程序在不同运算规模下,在利用性能预测的结果修正并行构件属性值之后的运行时间.当计算规模为 Class A-Class C 时,本文提出的 CNRS 算法选择与通信优先策略相同的计算资源,当计算规模为 Class D 时,本文提出的 CNRS 算法

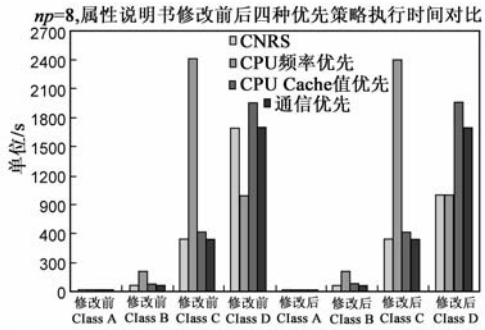

图4 *np*=8,属性说明书修改前后四种优先策略执行时间对比

利用更新后的构件属性值,其执行时间明显小于对应属性值被修改之前的执行时间,选择了与 CPU 频率优先策略相同的节点资源.

通过对比属性值修正前后 XMD 程序的执行时间,不难看出利用性能预测的反馈机制,本文提出的基于动态规划的并行构件资源选择算法能够更加精准地选择节点资源,提高并行构件执行性能,缩短并行构件执行时间.

5.2 可扩展性测试

针对 XMD 的四种计算规模,通过逐步增加执行进程数,分别对比了 CNRS 算法、CPU 频率优先、CPU Cache 值优先和通信优先四种方案的可扩展性.

如图 5 所示,当 $np=2$ 的时候,通信优先策略和 CPU Cache 值优先策略的节点资源选择结果是一样的.由于只设置了 2 个进程,CNRS 算法选择的结果比较少,但选择结果是正确的.当 $np=4$ 时,通信优先策略与 CPU Cache 值优先策略的选择结果出现分歧.通过图 5 可以看出,CNRS 算法依然选择了最佳的节点资源组合.当 $np=8$ 且计算规模为 Class A-Class C 的时候,CNRS 算法继续选择与通信优先策略相同的资源,同时可以看出此时通信优先策略优于 CPU Cache 值优先策略,可见在 XMD 程序中,通信/计算比 CCR 还是很大的.当计算规模为 Class D 时 CNRS 算法选择的结果与

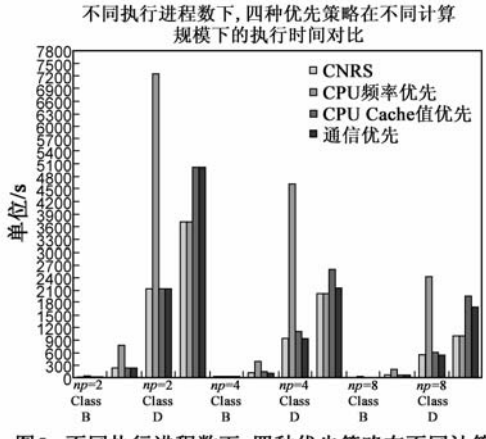

图5 不同执行进程数下,四种优先策略在不同计算规模下的执行时间对比

CPU 频率优先选择的结果相同.

随着进程数的增加,CNRS 算法总能够选择合适的节点资源作为运算节点,这体现了 CNRS 算法在可扩展性上的优势.因此,通过本文提出的层次化并行构件资源管理模型与基于动态规划的并行构件资源选择算法,并行构件能够动态地选择较合适的计算资源.

6 结论

本文提出了一种层次化并行构件资源管理模型,随着底层硬件的不断扩展,该模型通过资源属性映射机制和并行构件性能预测反馈机制,都能找到较优的资源来部署.通过研究此模型,重点提出了一种基于动态规划的并行构件资源选择算法,该算法实现了并行构件属性与特定资源之间的映射规则,使得并行构件能够较好地发挥底层硬件的性能.实验结果表明,相对于通常使用的 CPU 频率优先、CPU Cache 值优先与通信优先策略,本文提出的基于动态规划的并行构件资源选择算法,能够更精确地选择计算节点资源.同时随着进程数的增加,该算法总能够选择合适的计算节点资源作为计算节点,表明该算法具有良好的可扩展性.本文仅考虑了影响计算节点性能的部分因素,而实际影响计算节点性能的因素很多.在下一步的工作中将对此做进一步研究.

参考文献:

[1] D E Bernholdt, B A Allan, R Armstrong, et al. A component architecture for high-performance scientific computing[J]. International Journal of High Performance Computing Applications, 2006, 20(2): 163 – 202.

[2] L Courtrai, F Guidec, N L Sommer, et al. Resource management for parallel adaptive components[A]. Proceedings of the 17th International Symposium on Parallel and Distributed Processing[C]. Los Alamitos: IEEE computer society, 2003. 134 – 141.

[3] Y Maheo, F Guidec, L Courtrai. Towards resource-aware parallel java components[A]. Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications[C]. Las Vegas: CSREA Press, 2004. 1006 – 1012.

[4] A Denis, C Pérez, T Priol, et al. Padico: A component-based software infrastructure for grid computing[A]. Proceedings of the 17th International Parallel and Distributed Processing Symposium[C]. Los Alamitos: IEEE computer society, 2003. 2 – 8.

[5] J Brandt, A Gentile, J Mayo, et al. Resource monitoring and management with OVIS to enable HPC in cloud computing environments[A]. Proceedings of the 2009 IEEE International Symposium on Parallel and Distributed Processing[C]. Piscat-

away:IEEE computer society,2009.1 – 8.

[6] Song Fu,Cheng-Zhong Xu. Proactive resource management for failure resilient high performance computing cluster[A]. Proceedings of the 2009 International Conference on Availability, Reliability and Security[C]. Piscataway:IEEE computer society,2009.257 – 264.

[7] K Uhlemann,C Engelmann,S L Scott. JOSHUA:Symmetric active/active replication for available HPC job and resource management[A]. Proceedings of the 2006 IEEE International Conference on Cluster Computing[C]. Piscataway:IEEE computer society,2006.1 – 10.

[8] H A Duran,G S Blair. A resource management framework for adaptive middleware[A]. Proceedings of the Third IEEE International Symposium on Object-Oriented Real-Time Distributed Computing[C]. Los Alamitos:IEEE computer society,2000.206 – 209.

[9] 杨广文,武永卫,朱晶.一种全局统一的层次化网格资源模型[J].计算机研究与发展,2003,40(12):1763 – 1769.
Yang Guangwen,Wu Yongwei,Zhu Jing. A global uniform hierarchical grid resource model[J]. Journal of Computer Research and Development,2003,40(12):1763 – 1769.(in Chinese)

[10] 李立,刘元安,马晓雷.基于组合双向拍卖的网格资源分配[J].电子学报,2009,37(1):165 – 169.
Li Li,Liu Yuanan,Ma Xiaolei. Grid resource allocation based on the combinatorial double auction[J]. Acta Electronica Sinica,2009,37(1):165 – 169.(in Chinese)

[11] 牛新征,周明天,佘堃.一种应用于移动 P2P 网络的资源协作共享策略[J].电子学报,2010,38(1):18 – 24.
Niu Xinzheng,Zhou Mingtian,She Kun. A cooperative sharing scheme for resources in mobile P2P networks[J]. Acta Electronica Sinica,2010,38(1):18 – 24.(in Chinese)

[12] 林东岱,姜中华.一种网格资源优化分配方案[J].电子学报,2008,36(5):875 – 879.
Lin Dongdai,Jiang Zhonghua. A grid resource optimal allocation scheme[J]. Acta Electronica Sinica,2008,36(5):875 – 879.(in Chinese)

[13] 桂小林,王庆江,龚文强,钱德沛.面向网格计算的机器选择算法研究[J].计算机研究与发展,2004,41(12):2189 – 2194.
Gui Xiaolin,Wang Qingjiang,Gong Wenqiang,Qian Depei. Study of a machine selection algorithm for grid computing[J]. Journal of Computer Research and Development,2004,41(12):2189 – 2194.(in Chinese)

[14] B A Allan,R C Armstrong,A P Wolfe,et al. The CCA core specification in a distributed memory SPMD framework[J]. Concurrency and Computation:Practice and Experience,2002,14(5):323 – 345.

[15] 张景成,戴光明.基于 0/1 背包问题的算法探究[J].电脑知识与技术(学术交流),2007,2(11):1388 – 1389,1411.
Zhang Jingcheng,Dai Guangming. On the algorithm of the 0/1 knapsack problem[J]. Computer Knowledge And Technology(Academic Exchange),2007,2(11):1388 – 1389,1411.(in Chinese)

[16] Li Bin,P C Clapp,J A Rifkin,et al. Molecular dynamics simulation of stick-slip[J]. Journal of Applied Physics,2001,90(6):3090 – 3094.

作者简介:

李建江　男,1971 年 10 月出生于四川广安,博士,副教授,CCF 会员,主要研究方向为高性能计算、并行编译和并行软件工程.
E-mail:jianjiangli@gmail.com

崔　健　男,1986 年 8 月出生于江苏苏州,硕士生,主要研究方向为高性能计算、并行软件工程.
E-mail:cuijian613@yahoo.com.cn