

一种基于任务复制方法的网格调度算法

吕桦¹, 钟诚¹, 李智^{1,2}

(1. 广西大学 计算机与电子信息学院, 广西南宁 530004;

2. 广西科技信息网络中心, 广西南宁 530012)

摘要:根据 Internet 上存在大量空闲主机的情况, 结合流行的 P2P 的思想给出了一种基于资源代理的网格系统模型, 该模型能提供超级计算能力给一般的用户。针对在像网格这样的大规模系统中部署大量监控组件的困难性, 设计了一种基于任务复制的调度算法。该算法无需任何有关环境的预测信息, 仅需知道任务的相对长度, 就可在执行过程中自动地适应网格的动态性, 并且具有容错功能。

关键词:网格; 任务调度; 任务复制; P2P

中图分类号: TP301.6

文献标识码: A

文章编号: 1673-629X(2006)08-0066-03

A New Grid Scheduling Algorithm Based on Task Replication

LÜ Hua¹, ZHONG Cheng¹, LI Zhi^{1,2}

(1. School of Computer and Electronics and Information, Guangxi University, Nanning 530004, China;

2. Guangxi Science and Technology Information Network Center, Nanning 530012, China)

Abstract: For the fact that there are a lot of underutilized hosts on the Internet, proposes a resource broker-based grid model by referring to the popular Peer-to-Peer idea, which can provide super computing capability to the ordinary users. Then, this paper presents a new scheduling algorithm based on task replication to solve the difficulty of monitoring a large-scale grid system. The algorithm doesn't require the forecasting information about the running environment. Only gives the relative length of the tasks, it can dynamically adapt to the varying grid environment, and it has the tolerant function.

Key words: grid; task scheduling; task replication; P2P

1 网格任务调度的相关研究

随着 Internet 的不断发展, 将地理上广泛分布的大量计算资源(包括超级计算机、集群、工作站、PC 等)组合起来进行大规模问题的求解变得日益普遍, 由此产生了所谓的“网格计算”^[1]。网格在本质上是一种具有巨大计算能力的异构共享分布式系统。如何将网格这种计算能力无缝地提供给用户依然是一个富有挑战性的问题。其中的一个主要困难在于如何在网格这样的异构共享分布式系统中调度一个大型的应用。由于网格的广泛分布性、异构性和动态性, 松散耦合的并行应用相比紧耦合的应用更适合在网格上运行。文中所考虑的正是这样的一类粗颗粒度应用——BoT(Bags-of-Tasks)应用。BoT 应用是这样的一类应用, 它由相互独立的任务组成, 任务可以以任何顺序执行, 并且任务间不需要通信。网格上任务调度的困难主要由网格的内在特点(即广泛分布性、异构性和动

态性)所决定。

众所周知, 异构计算环境中的调度问题是一个 NP 难的问题, 因此大量的启发式方法^[2,3]被应用于这一领域的研究。但与此同时还应看到, 网格绝不仅仅是一个异构计算环境。网格调度的困难更多地来自于它的动态性。自治资源的动态加入和离开, 网络带宽和网络延迟的变化, 由于共享所带来的资源竞争等等, 所有这一切使得网格中的应用所能访问的计算能力随时间动态变化。为了减少这些动态变化所带来的负面影响, 许多调度系统都采用了预测的方法^[4]。具有较准确预测信息的调度通常要优于那些已有的无任何预测信息的调度^[5]。最常使用的预测信息有主机负载、主机速度、任务长度、任务到达时间等。大多数基于预测的调度方法都事先假定这些信息是已知的。但是, 由于网格的广泛分布性, 很难获得整个网格系统的较为准确的参数信息。尽管已有一些工作(如 NWS^[6])致力于网格监控以帮助获得网格的动态信息, 但是, 要进行整个网格规模的监控还有许多困难^[7]。而且, 并不是所有的用户都愿意安装监控系统。例如, 正在兴起的一种形式的网格计算——公共资源计算^[8], 是利用家庭用户和办公室电脑捐献的空闲计算周期进行计算以更快地完成大规模计算任务。在这种情况下, 要求每个网格资

收稿日期: 2006-01-18

基金项目: 广西科学基金资助项目(桂科基 0575014); 广西科技信息网络中心资助项目

作者简介: 吕桦(1980-), 女, 河南襄城人, 硕士研究生, 主要研究网络与并行计算; 钟诚, 博士, 教授, 主要研究网络与并行计算。

源安装类似的监控系统是不太可能的。鉴于此种情况,最近有文献提出通过任务复制而不是预测信息来适应网络的动态性^[5]。

2 基于资源代理的网络调度

随着计算机性能的不断提高以及高速互联网络的日益普及,现在 Internet 上存在着大量的空闲主机,这些主机的资源利用率很低,造成了计算能力的大量浪费。如能将这些闲置的计算周期聚集起来以满足大规模应用的计算需求,将是一件非常有意义的事情。目前已经有一些工作致力于这方面的研究,如已经成功开展的 SETI@home 项目^[9]和 Condor 项目^[10]。SETI@home 是使用 Internet 上的空闲处理周期来寻找外星人,它需要捐献周期者通过一个集中的网络站点来手工协调,并且它不能提供超级计算能力给一般的用户。Condor 虽然提供了自动调度功能,但是要求有一个集中的匹配服务。与此不同,文中借鉴了 P2P 互利互惠的理念,打破传统的 C/S 模式,给出了一种利用 Internet 上大量空闲主机的网格系统结构。该系统通过聚集互联网上的大量空闲处理周期来提供潜在的超级计算能力给每一个有大规模应用计算需求的参与者。

2.1 基于资源代理的网络系统结构

资源代理又称为元调度器,通常用于虚拟资源集接口。代理作为用户实体和相关资源集的媒介,提供任务的单一提交点,利用标准的网格管理协议(如 GRAM)将任务提交到一个或多个底层资源上执行。使用资源代理具有如下优点^[11]:

①资源虚拟化。代理的最基本的优点是它提供了资源集的简化视图,使得终端用户不需要了解一个虚拟组织可用资源的具体特征。

②策略实施。代理最重要的功能是为任务选定它应该使用的底层资源。尽管这个决策可能完全出于资源管理方面的考虑(如均匀分布负载,或获得最高的资源利用率),但是代理也为基于社区的策略提供控制中心。这个策略能够实施社区优先级、用户角色和成本模型等。

③协议转换。尽管网格协议(如 GRAM)定义了标准的远程作业提交方法,许多不支持网格协议的工具也是有用的。在这种情况下,代理除了正常的资源选择功能外,还可以作为协议转换器。

鉴于资源代理的诸多好处,使得它在资源管理特别是在分布式调度方面(如 Nimrod-G^[12])得到了广泛的应用。基于资源代理的网格系统的结构如图 1 所示。

其中各主要部件的作用简要介绍如下:

①用户生成网格作业并提交作业给资源代理。用户除了运行网格作业外还运行本地作业,并且优先运行本地作业,只是将空闲的计算周期贡献给网格作业使用。

②资源代理是该系统中的核心部分。每一用户都有一个专用的资源代理。当用户向其提交作业时,它负责通过网格信息服务(GIS)获得满足条件的资源列表,将作业分解为一组可以并行运行任务,利用调度算法对作业生成合适的调度,提交组成作业的各个任务,接收完成的任务并对结果进行组合以产生最终的作业结果返回给用户。也就是说资源代理全权负责作业的调度和执行,因此,在某种程度上,资源代理同时具备用户代理的功能。

③GIS 负责提供当前系统中可用的资源列表给资源代理。各个资源可通过软状态协议周期性地向 GIS 声明自己的存在。GIS 可采用 Globus 中 MDS 的组织方式^[13]。

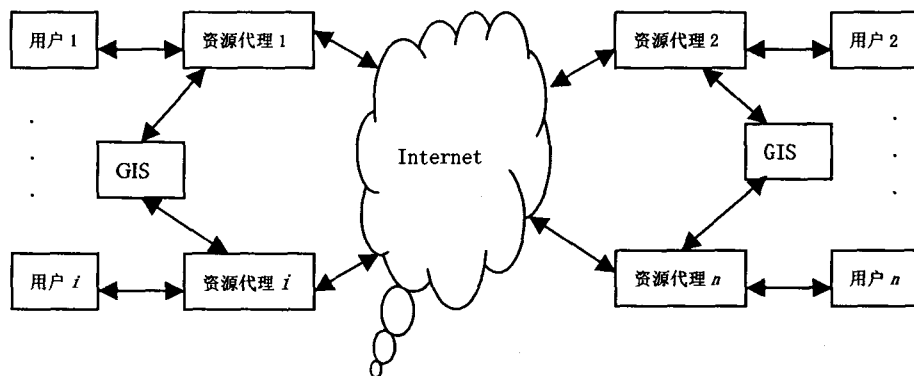


图1 基于资源代理的网络系统结构

2.2 基于任务复制方法的网络调度算法

文献[5]采用了任务复制方法,它所针对的是任务个数大于资源个数的情况,并且没有利用已有任务运行所反馈回来的有用信息。文献[14]提出了容错的调度策略,该策略松散地耦合作业调度和作业复制方案。当用户提交一个作业给局部资源管理器时,指定期望的副本数,每个副本驻留在一个单独的站点。各个副本周期性地通信以便作业在一个站点上失败时能转至另一个站点运行。与此不同,文中不对整个作业进行复制,而是对构成作业的各个任务进行复制,并且副本间不需要通信。

文中给出的基于资源代理的网格任务调度算法同样采用了任务复制的方法。但是文中的调度算法不需要有关资源的任何预测信息,仅需提供任务的相对长度信息。调度采用事件驱动的方式,调度事件发生时,采用的调度算法描述如下:

算法1 基于任务复制方法的网络调度算法

输入: 资源集合 $R = \{r_1, r_2, r_3, \dots, r_n\}$

任务集合 $T = \{t_1, t_2, t_3, \dots, t_m\}$

任务的相对长度集合 $L = \{l_1, l_2, l_3, \dots, l_m\}$

//算法开始

对 T 中任务根据其相对长度按降序排列;

$I=1$ //任务的下标

While(T 中有任务未完成) do

While (R 中有资源未利用) do

If (第 1 个任务已完成) then

I = I + 1;

If (I > m) then

I = 1;

End if

End if

将任务 I 提交给 R 中第一个未利用的资源(记为 J),
同时记录提交的时间 S_{ij} ;

I = I + 1;

If (I > m) then

I = 1;

End if

End while

接收各个完成的任务,记录其完成的资源号和完成时间;

对各个返回任务的资源 k(设其完成的任务为 x,任务的提交时间为 S_{xk} ,任务的完成时间为 E_{xk} ,任务的相对长度为 l_x),按 $(E_{xk} - S_{xk})/l_x$ 的升序排列;

End while

//算法结束

算法中对资源集 R 可用数组的形式存储,对数组中的每一元素记录其资源号,同时若非空闲则记录提交的任务号及其提交时间。寻找未利用资源时从前往后搜索,初始时资源是随机排列的,但随着任务的执行资源逐渐按其性能排序,所以提交的资源是当前未利用资源中最好的。

算法分析:对于文中给出的网格计算环境,一般情况下可以认为资源个数 n 是远远大于任务个数 m 的。这时,任务的副本数为 $\lceil n/m \rceil$,任务的多个副本同时执行使得任务的成功完成几率大大增加,而任务完成时间则为几个副本中最早完成的那个副本的结束时间。这就为调度提供了容错服务。而在资源个数不大于任务个数时,由于算法已将任务按相对长度的降序排列,随着任务的执行,对资源也逐渐根据其综合性能按降序排列,所以在此种情况下,文中给出的调度算法演化为动态 FPTLTF(Dynamic Fastest Processor to Largest Task First)^[5],只不过文中不是根据资源负载和速度的预测信息,而是根据任务实际运行所获得的反馈信息来评估资源性能的,这就能动态地适用网格资源可用性的变化,并且避免了进行网格实时性能监控的困难。

3 结束语

由于网格环境中的任务调度是众所周知的 NP 难问题,所以大量的启发式方法得到了应用。另外,为了应对网格环境内在的动态性,大多数调度算法采用了性能预测的方法。但在一个潜在的全球规模的网格环境中进行大规模的监控部署以获得实时的系统性能信息有许多困难。

文中针对 Internet 网上大量主机利用率极低的情况,结合流行的 P2P 的思想给出了一个基于资源代理的网格系统结构,并设计了用于资源代理中的基于任务复制的调

度算法。该算法的优点是无需进行性能预测,而是通过任务复制的方法来自动地适应网格的动态性,同时还具有容错的功能。

参考文献:

- [1] Buyya R, Chapin S, DiNucci D. Architectural Models for Resource Management in the Grid[A]. In Proc of 1st IEEE/ACM International Workshop on Grid computing (GRID 2000), LNCS, Vol. 1971 [C]. [s. l.]: Springer - Verlag, 2000. 18 - 35.
- [2] Braun T D, Siegel H J, Beck N, et al. A comparison study of static mapping heuristics for a class of meta-tasks on heterogeneous computing systems[A]. In Proc of 8th IEEE Heterogeneous Computing Workshop (HCW'99)[C]. [s. l.]: IEEE Computer Society Press, 1999. 15 - 29.
- [3] Xu Zhihong, Hou Xiangdan, Sun Jizhou. Ant algorithm-based task scheduling in grid computing[A]. In Proc of 2003 - Canadian Conf on Electrical and Computer Engineering, vol. 2 [C]. [s. l.]: IEEE Computer Society Press, 2003. 1107 - 1110.
- [4] Berman F, Wolski R. Adaptive Computing on the Grid Using AppLeS[J]. IEEE Transactions on Parallel and Distributed Systems, 2003, 14(4): 369 - 382.
- [5] Paranhos D, Cirne W, Brasileiro F. Trading cycles for information: Using replication to schedule bag-of-tasks applications on computational grids[A]. In Proc of international Conference on Parallel and Distributed Computing (Euro - Par), LNCS, Vol. 2790 [C]. [s. l.]: Springer - Verlag, 2003. 169 - 180.
- [6] Wolski R. Dynamically forecasting network performance using the network weather service[J]. Cluster Computing, 1998, 1(1): 119 - 132.
- [7] Francis P, Jamin S, Paxson V, et al. An Architecture for a global Internet host distance estimation service[A]. Proc of IEEE INFOCOM, Vol. 1 [C]. [s. l.]: IEEE Press, 1999. 210 - 217.
- [8] Fujimoto N, Hagihara K. A Comparison among Grid Scheduling Algorithms for Independent Coarse-Grained Tasks[A]. Proceedings of the 2004 Symposium on Applications and the Internet - Workshops (SAINT 2004 Workshops) [C]. [s. l.]: IEEE Computer Society Press, 2004. 674 - 680.
- [9] Anderson D, Cobb J, Korpela E, et al. SETI@home: An Experiment in Public-Resource Computing[J]. Communications of the ACM, 2002, 45(11): 56 - 61.
- [10] Litzkow M, Livny M, Mutka M W. Condor - A Hunter of Idle Workstations[A]. In Proceedings of 8th International Conference on Distributed Computing Systems [C]. [s. l.]: IEEE Computer Society Press, 1988. 104 - 111.
- [11] Foster I, Kesselman C. 网格计算(第 2 版)[M]. 金海,袁平鹏,石柯,译.北京:电子工业出版社,2004.

工作,待程序运行完毕后则可利用 pfmon 生成的数据,按照事件计数器与性能指标的映射关系进行计算从而得到对于程序的全局性信息。

这种方法的好处在于:一方面可使部分分析过程自动实现,减轻用户手工分析的负担;另一方面,将软件开发人员在机器模型、性能模型、程序性能优化等方面的专业知识融入到数据分析中,能够得到一些更具指导意义的性能指标,对程序性能优化更具指导意义。

2)从局部出发,将数据进行逆向映射,对应到源代码级别。

基于这种思想开发的工具是针对体现程序性能中的某一种事件进行分析的,并建立程序源代码语句与被监视的事件的关联,有以下几个方面可以利用来实现从事件到源代码的精确定位。

首先,可以按照事件地址寄存器 EARs 中收集的 IP 地址对采样样本进行排序和分类以找出出现频率最高的 IP 地址^[5],在大多数情况下,这些地址就代表着反映性能瓶颈的程序中的“热区(hot spots)”所在。比如,如果监视的是指令 Cache Miss 事件,那么 EARs 的 IP 地址中出现的最多的就应该是 Cache 中缺失最频繁的指令。按照程序局部性原理,80% 的程序执行时间是集中在 20% 的代码段上,因此,确定程序的“热区”对于改进程序性能是非常重要的一个环节。

其次,可以将收集的采样样本与分支路径缓冲的内容联系起来,以建立与被监视事件相符合的程序执行路径。

最后,可以利用在程序编译过程中生成的反映执行过程信息的调试符号表(debugging symbol table)来将 IP 地址映射为源代码中的相应位置,也就是指出产生被监视事件的语句的行号。可以通过 Linux 下的工具软件 GDB 将调试符号表保存为文件,然后对文件进行分析,通过从 EARs 中读取的 IP 地址与调试符号表中的内容进行对照匹配,得到产生 EARs 中 IP 地址的指令与源程序中某一行语句的对应关系,这将有助于定位那些产生瓶颈的代码和数据结构并分析产生瓶颈的原因。这种开发工具的处理流程如图 3 所示。

这两种开发方向各有利弊,第一种全局性的数据分析方法虽然只能概略性地指出程序性能瓶颈存在于哪个方

面,但其实现过程简单;而第二种局部性的方法能够具体精确地指出产生性能瓶颈的语句所在,但其实现过程较为烦琐,开发周期长,因此很难评说这两种开发方向的优劣,只能说,应该在面对不同的应用环境下可以选用不同的开发方向,当然,两者的结合将是更加理想的状态,只不过其开发更为复杂。

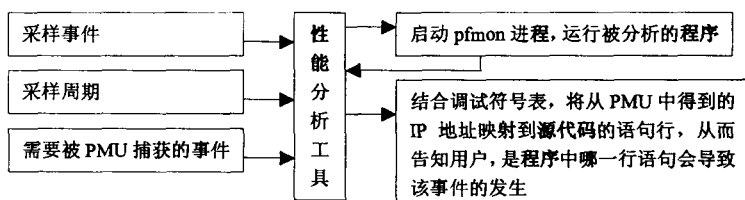


图 3 对数据进行逆向映射的工具的组织结构

4 结束语

程序性能分析对于程序优化的意义将随着计算机结构的不断更新发展而显得越来越重大。中国古语云:工欲善其事,必先利其器。因此,开发适应用户需求的性能分析工具必将极大地推进程序性能分析工作。文中介绍的基于 pfmon 的开发方法,加以变通,必将能够适应 IA-64 家族中的其余微处理器。

参考文献:

- [1] Intel Corporation. Intel Itanium 2 Processor Reference Manual for Software Development and Optimization [EB/OL]. <http://developer.intel.com/design/itanium2/manuals/index.htm>. 2002-06.
- [2] Jarp S. A Methodology for using the Itanium 2 Performance Counters for Bottleneck Analysis, Tech - Report HP Labs [EB/OL]. <http://www.gelato.org/pdf/Performance-counters-final.pdf>, 2002-08-27.
- [3] Mosberger D, Eranian S. IA-64 linux kernel: Design and Implementation[M]. [s.l.]: Prentice Hall PTR, 2002.
- [4] HP Labs. Perfmon Project web site[EB/OL]. <http://www.hpl.hp.com/research/linux/perfmon/>, 2005-02.
- [5] Lingamneni A. PerfView - A Tool for Source-Level Performance Analysis on the Itanium Processor Family[EB/OL]. <http://dynopt.dtc.umn.edu/documents/perfview-ananth-planb-report.pdf>. 2004-06.

(上接第 68 页)

- [12] Buyya R, Abramson D, Giddy J. Nimrod/G: An Architecture of a Resource Management and Scheduling System in a Global Computational Grid[A]. In Proceedings of the 4th International Conference and Exhibition on High Performance Computing in Asia - Pacific Region (HPC ASIA 2000)[C]. [s.l.]: IEEE Computer Society Press, 2000. 283-289.
- [13] Czajkowski K, Fitzgerald S, Foster I, et al. Grid Information Services for Distributed Resource Sharing[A]. In Proceedings

of the 10th IEEE International Symposium on High-Performance Distributed Computing (HPDC-10)[C]. [s.l.]: IEEE Press, 2001. 181-195.

- [14] Abawajy J H. Fault-Tolerant Scheduling Policy for Grid Computing Systems[A]. In Proceedings of the IEEE 18th International Parallel and Distributed processing Symposium (IPDPS'04)[C]. [s.l.]: IEEE Computer Society Press, 2004. 238-244.