第 33 卷 第 5 期 计 算 机 工 程 2007 年 3 月
Vol.33 No.5 Computer Engineering March 2007
·软件技术与数据库· 文章编号：1000—3428(2007)05—0068—03 文献标识码：A 中图分类号：TP338.8

TDCE:基于 Tspaces 的分布并行计算系统

胡 军，郭绍忠，周 蓓
(信息工程大学信息工程学院，郑州 450002)

摘 要：Tspaces 是一种新型的网络中间件。它为网络环境中各进程提供一种强大的共享存储机制来处理相互之间通信和同步。在 Tspaces 的基础上，构造了一个用于群机环境的并行计算系统 TDCE。TDCE 支持 SPMD 模式的并行程序，实验结果表明 TDCE 能以较小的系统配置和管理开销构建分布式计算平台，为并行程序的开发运行提供有效的支持，给出了系统 MPI 的对比测试结果并作了分析。
关键词：分布式计算；并行计算；Tspaces；元组空间

TDCE: Distributed and Parallel Computing System Based on Tspaces

HU Jun, GUO Shaozhong, ZHOU Bei
(College of Information Engineering, Information Engineering University, Zhengzhou 450002)

【Abstract】 Tspaces is a new network middleware, it provides a powerful shared-memory mechanism to handle communication and synchronization of processes in heterogeneous environment. This paper is based on Tspaces, constructs a parallel computing system TDCE used for computer cluster environment. TDCE sustains SPMD parallel programming, the testing result demonstrates TDCE can form a distributed computing platform with minimal system configuration and management overheads. It provides effective assistance for the developing and running of parallel program. It compares TDCE with MPI and analyzes the test result.
【Key words】 Distributed computing; Parallel computing; Tspaces; Tuplespace

传统的高性能计算都是基于大规模并行处理器、超级计算机或是高端的工作站集群通过高速网络互连实现的。这些资源价格昂贵而且大多都是面向特定的分布并行应用。对于某些通用的并行计算问题，如果能充分利用现有网络中的空闲资源进行计算将是一个不错的选择。不过该方式也存在以下挑战：(1)异构性：网络环境是典型的异构资源的集合，各台机器的配置、计算能力、可用软件和工具都千差万别，这些异构性对用户应该是不可见的。(2)干扰性：计算平台主要是针对空闲资源，一旦该空闲资源转入工作状态，平台上运行的计算不能对现行系统造成严重干扰。(3)系统的配置和管理开销：利用网络中的节点构建平台必然要对各节点的机器进行配置和管理，这将带来一笔额外的开销，如果开销过大会影响计算效率。(4)动态适应性：网络中的节点状态总是处于不停的变化之中，要保证可靠的计算提高系统的可扩展能力，平台必需具有很好的动态适应性，各种变化情况都能得到及时处理。(5)安全性：本地资源要提供给外部的计算使用，如何保证对资源安全可靠的访问，需要有相应的安全策略加以控制，同时平台自身的安全性也不容忽视。

1 相关工作

早期的分布式计算系统(如 Condor 或 SETI@Home)提供了在网络中异构资源上远程执行任务的功能，这为分布并行计算提供了很好的基础，但需要提供更多的功能以解决更复杂的问题。Condor 和 SETI@Home 是支持独立并行问题的示例，在这些系统中，可以独立计算每个任务，同时任务彼此之间没有任何关联。因此，这些系统缺乏处理关联问题的能力，而在实际科学计算中，一个问题往往很难分解成没有任何关联的任务集合。

基于消息传递的 MPI 通过在各个并行执行的部分之间传递消息来交换信息、协调步伐、控制执行。消息传递为编程者提供了更灵活的控制手段和表达并行的方法，一些用数据并行方法很难表达的并行算法都可以用消息传递模型来实现。灵活性和控制手段的多样化，是消息传递并行程序能提供高的执行效率的重要原因。然而，消息传递模型一方面为编程者提供了灵活性，另一方面它也将处理各个并行执行部分之间复杂的信息交换和协调、控制的任务交给了编程者，这在很大程度上增加了编程者的负担，也是消息传递编程模型编程级别较低的主要原因。

2 TSpaces 简介

TSpaces 是由 IBM Almaden Research Center 研究开发的一种网络中间件，它是带有数据库功能的网络通信缓冲区，它使用一种简单的、永久的数据仓库扩展消息传送系统，提供一种强大的机制来处理通信和同步。具有如下特点：

(1)匿名性：TSpaces 通信是完全匿名的，数据以元组(tuple)的形式存储在元组空间中。Tuple 一旦提交成功，其使用情况与创建者无关。

(2)异步性：通信节点之间是通过访问共享区进行通信的，通信伙伴可以不必在一个特定的时刻相互通信。

(3)分布性：通信是发生在一些空间上彼此分隔的多台计算机之间，它们的通信是一种分布的方式，只要它们可以访问元组空间，就可以很好地合作。

(4)持久化存储：元组空间中的数据对象可能比产生它们的程序存活时间更长，而且元组空间本身可以看作是一个轻量级的数据库。

基金项目：军队基金资助重点项目
作者简介：胡 军(1979－)，男，硕士生，主研方向：分布式计算；郭绍忠，硕士、副教授；周 蓓，硕士、讲师
收稿日期：2006-10-11 **E-mail：**desertfox617@yahoo.com.cn

3 TDCE 体系结构

图 1 所示的是 TDCE 的体系结构，系统分 3 层。底层是 Tspaces，通过 Tspaces 可以实现不同机器之间的相互通信传递任务参数。系统的管理层由两部分组成：HTTP 服务器和守护进程。HTTP 服务器运行于系统的某台主机上，负责作业的管理、任务的调度、资源的管理。守护进程运行于系统每个计算节点上，主要负责系统的初始化和管理信息的处理。应用层由 Master 和 Worker 模块组成。当系统初始化后，Master 和 Worker 会根据各自的分工执行任务直到获得最终的结果。

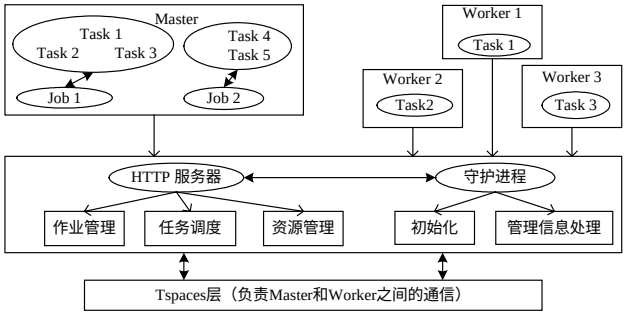

图 1 TDCE 系统体系结构

在一个异构的群机环境下构建这样一个系统，只需开启一个 HTTP 服务器，在每个计算节点上安装守护进程，由此可见系统配置开销不大而且十分方便。在这样一个结构中，其核心 Tspaces 是纯 Java 语言编写的，Java 的跨平台特性使我们的系统很好地解决了资源异构性的问题。整个系统的管理是由 HTTP 服务器集中实现的，它与各节点上的守护进程直接通信，实时监控系统状态，这种集中式管理易于实现，管理代价小且效率高。守护进程着重解决系统动态特性和干扰性问题，计算节点的加入撤出完全由守护进程控制。开启守护进程即自动加入平台，关闭则意味着退出计算，同时 Tspaces 的持久化存储和匿名通信功能保证了在计算节点异常情况下任务不会丢失，其它计算节点会继续执行未完成任务。通过给守护进程动态设置优先级，改变其对资源的占有率，最低限度降低对所在节点的干扰。安全方面引入角色和权限的概念，保证对资源的安全访问。容错方面采用了基于检查点的系统级容错机制，通过日志文件进行事务处理和恢复，解决了因主机故障或用户误操作所带来的计算数据丢失，使作业可以从断点开始运行，确保计算结果的正确性。

4 TDCE 的设计和实现

4.1 Master 模块

Master 即主控平台，主要负责计算任务的划分和结果回收。Master 按照用户的要求根据一定的划分算法调用(allocTask())，将每个 Job 划分若干个独立的计算任务，同时将这些任务(putTask())和任务参数(setTaskPara())放入共享区，等待回收结果(getResult())。

4.2 Worker 模块

在机群中，除运行 Master 的节点以外，其它机器均运行 Worker 作为计算节点。其主要工作是从共享区中获取任务(getTask())并执行任务，最后调用(putResult())将结果放入共享区。

4.3 HTTP 服务器

HTTP 服务器主要是提供 Web 界面，使得通过浏览器可以提交要执行的任务，查看作业运行时的信息，管理计算资源，其主要功能模块有以下几种：

(1)作业管理模块

作业管理模块为用户提供作业提交、作业队列的更新、作业实时监控、作业结果显示等功能。通过采用基于 Web 的实现使用户界面更加友好，提高计算平台与用户的交互性，同时减轻了 Worker 对 Master 直接访问的压力。在作业执行过程中，HTTP 服务器动态监控整个计算平台的运行状态，用户可以实时跟踪作业执行情况。

(2)资源管理模块

一般在组织内部机器资源是按照行政机构分别属于不同的行政管理单位，比如：机器 A 属于公司里的 1 部 1 科，或某部某科。因此，系统以行政单位为节点建立一个树型结构，如图 2 所示。

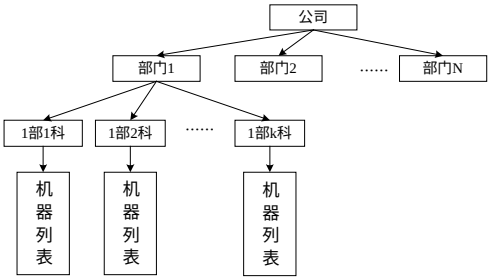

图 2 计算资源的树型结构

在这个结构中，按照角色与权限相对应的原则，以公司(管理员)的身份，部门的身份，科的身份登录将获得不同的资源调度权限。这种方式不仅便于管理而且也保证了对资源的安全访问。

(3)作业调度模块

作业调度采用了 FIFO 的调度策略，在 HTTP 服务器端维护一个等待作业队列。当有新的作业到来，首先加入到队尾等待执行。当系统中有空闲资源可用时，队首的作业被调度，同时判断其执行条件是否于与当前条件相符，如果不符，该作业继续等待，系统将调度下一个作业执行。当某个作业执行完毕，系统自动更新作业队列，将该作业从队列中删除。

4.4 守护进程

守护进程位于每个计算节点上，系统安装完毕后，守护进程会自动运行，它首先向 HTTP 服务器注册，使其所在计算节点成为系统中可用的计算资源。另外，当作业开始运行时，主控计算节点的守护进程还负责启动 TSpaces Server。在整个平台的运行期间，Master 和 Worker 以及 HTTP 服务器之间传递的一些管理信息，都是由守护进程发送和处理的。

5 性能测试与比较

5.1 测试问题描述

穷举文件口令。对于一个加密文件，穷举该文件口令。程序依据用户的设定，逐个穷举可能的口令，如果找到口令，程序中止，否则继续穷举。

5.2 性能测试

根据上述划分方法，选取口令长度 5 位，字符集大小为 90，分别测试在不同机器数和任务数下系统性能，测试机器：P4 2.8GHz，256MB，测试结果如表 1 和表 2 所示。

表 1 不同机器数下口令破解程序的执行时间和加速比

机器数	任务数	时间/s	加速比
1	90	15 613	—
10	90	1 632	9.568
20	90	817	19.12
50	90	328	47.65

表 2 不同任务数下口令破解程序执行时间和加速比

机器数	任务数	时间/s	加速比
10	90	1 632	—
10	902	1 954	0.835
10	903	5 106	0.319

从表 1 可以看出，程序运行性能随机器数目的增加而增加，程序获得了有效加速比，但随着计算力的进一步的增加，加速比增长滞后于机器的增长，说明计算力的增长要以任务总量为依据，否则，过量增长只能造成资源浪费。表 2 数据表明在计算力相同的情况下，当任务数超过某个范围后，加速比下降明显，这是由于在 TDCE 中 Master 和 Worker 之间的数据传递必须通过共享区转发，通信成为提高系统性能的瓶颈。从以上分析可以看出，系统对于解决口令破解这类穷举问题有着较好的性能，在问题规模一定的情况下，关键是要选择合适的任务粒度和相应的计算力资源，最大程度降低通信开销。

5.3 性能比较

对比测试中分别用 MPI 的 Windows 版 mpich.nt.1.2.5 环境下实现的系统和本系统进行了比较。

测试用例是解含错方程。该方程有 32 个未知元，采用 16 位穷举，16 位对折的方法进行求解。对于相同计算量的同一个问题，我们将用本系统的编程接口和 MPI 的不同实现分别在同一组机器上做了测试，结果如表 3 所示。性能比较如图 3 所示。

表 3 测试

参加计算的机器数目	MPI	本系统
2	15’15	13’59
4	7’37	6’54
8	4’4	3’30

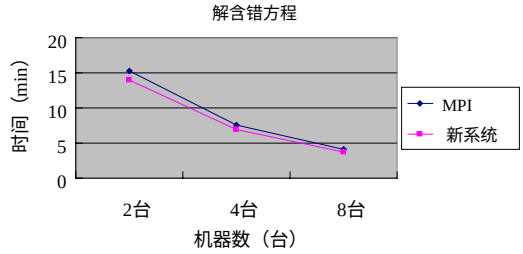

图 3 性能比较

测试结果显示系统在解决任务量大而且子任务间通信不太频繁的问题时，性能比 MPI 略优；同时 MPI 不支持运算过程中计算节点的动态加入和撤出，灵活性不够且不支持动态负载均衡，缺乏一套完善的容错机制。相比之下，本系统可以很好地解决这些问题。

6 结束语

本文主要介绍了分布计算环境 TDCE 设计和实现，该环境是基于 Tspaces 技术实现的。由于 Tspaces 提供了简单而高效的共享内存访问机制，解决了各计算结点之间通信透明性的问题。TDCE 提供了简单易懂的 API，支持用户方便地编写 SPMD 模式并行程序。实验结果表明 TDCE 已经能为并行程序的开发运行提供有效的支持。

我们下一步的工作包括：(1)用户界面优化，提供程序设计向导；(2)改进共享区通信瓶颈问题，进一步提高并行程序的计算性能。

参考文献

1 Tspaces Documentation[DB/OL]. http://www.almaden.ibm.com/cs/TSpaces.

2 Sun Microsystems. JavaSpaces[DB/OL]. http://www.javasoft.com/products/javaspaces/specs/.

3 Jyoti B, Manish P. Adaptive Cluster Computing Using Java Spaces[C]//Proceedings of the IEEE International Conference on Cluster Computing. 2001: 323-330.

4 Lizkow M, Livney M, Mukta M. Condor: A Hunter of Idle Workstations[C]//Proceedings of the 8th International Conference on Distributed Computing Systems. 1998.

5 Baratloo A, Karaul M, Karl H, et al. An Infrastructure for Network Computing with Java Applets[J]. Concurrency: Practice and Experience, 1998, 10(11/13): 1029-1041.

6 Kai H. Advanced Computer Architecture: Parallelism, Scalability, Programmability[M]. 北京: 机械工业出版社, 1993.

7 Marko B. Java 与分布式系统[M]. 曹学军, 译. 北京: 机械工业出版社, 2003.

8 陈国良. 并行计算——结构, 算法, 编程[M]. 北京: 高等教育出版社, 1999.

9 都志辉. 高性能计算并行编程技术——MPI 并行程序设计[M]. 北京: 清华大学出版社, 2001.

10 庞丽萍, 唐晓辉, 羌卫中. 分布式计算模式及其软件开发包[J]. 华中科技大学学报, 2005, 33(4): 10-12.

（上接第 46 页）

参考文献

1 刘 霞, 李明树, 王 青, 等. 软件体系结构分析与评价方法评述[J]. 计算机研究与发展, 2005, 42(7): 1247-1254.

2 胡红雷, 毋国庆, 梁正平, 等. 软件体系结构评估方法的研究[J]. 计算机应用研究, 2004, 21(6): 11-14.

3 Dobrica L, Niemela E. A Survey on Software Architecture Analysis Methods[J]. IEEE Transactions on Software Engineering, 2002, 28(7): 638-653.

4 Ali-babar M, Zhu L, Jeffery R. A Framework for Classifying and Comparing Software Architecture Evaluation Methods[C]// Proceedings of the Australian Software Engineering Conference. 2004.

5 Ali-babar M, Gorton I. Comparison of Scenario-based Software Architecture Evaluation Methods[C]//Proceedings of the 11th Asia-Pacific Software Engineering Conference. 2004.

6 Kazman R, Bass L, Abowd G, et al. SAAM: A Method for Analyzing the Properties of Software Architectures[C]//Proceedings of the 16th International Conference on Software Engineering. 1994.

7 Lung C H, Bot B, Kalaichelvan K, et al. An Approach to Software Architecture Analysis for Evolution and Reusability[C]//Proceedings of CASCON. 1997.

8 Bengtsson P, Bosch J. Scenario-based Software Architecture Reengineering[C]//Proc. of the 5th International Conference on Software Reuse. 1998.

9 Kazman R, Klein M, Barbacci M, et al. The Architecture Tradeoff Analysis Method[C]//Proceedings of IEEE on ICECCS'98. 1998.

10 Bengtsson P, Lassing N, Bosch J, et al. Architecture-level Modifiability Analysis[J]. The Journal of Systems and Software, 2004, 69(1): 129-147.

11 Clements P C. Active Reviews for Intermediate Designs[Z]. 2000-08. http://www.sei. cmu.edu/publications/pubweb.html.