JOGR))) 利用 Java 对象组关系支持多模式并行编程

邓倩妮, 陆鑫达
(上海交通大学计算机科学与工程系, 上海 200030)

摘 要: 为支持在机群环境和 Grid 环境中实现 Master/Slave、分而治之、阶段并行等多种并行范例, 本文提出了 Master/Group/Slave 编程模式 JOGR, 利用 JOGR 可以实现机群环境下的动态数据分配和负载均衡. 此外, 我们还基于 WebServic 技术和 Java 语言实现了 JOGR. 实验结果表明 JOGR 能为并行程序的开发运行提供有效的支持.

关键词: WebService; 对象组; 集群计算; 格点计算; 并行编程范例
中图分类号: TP338.8 文献标识码: A 文章编号: 0372-2112 (2002) 12-1718-04

JOGR$$ Utilizing Java Object Group Relationship to Support Multi-Model Parallel Programming

DENG Qian-ni, LU Xin-da
(Dept. of Computer Sci. & Eng, Shanghai Jiaotong University, Shanghai 200030, China)

Abstract: Parallel programming paradigms to support multi-models in cluster and Grid environment, such as Master/Slave, Divide and Conquer, Phase Parallel and so forth this paper proposes a Master/Group/Slave programming model- JOGR with which we can realize data allocation and load-balance dynamically in cluster environment. Furthermore, based on WebService technology and Java language, we focus on the implementation of JOGR. Experimental results show that JOGR can provide effective support for developing and running parallel programs.

Key words: webService; object group; cluster computing; grid computing; parallel programming paradigm

1 引言

消息传递编程软件 PVM, MPI 在并行计算中有重要作用, 却无法适应 Web 计算和 Grid 计算的要求. 目前有一种趋势是利用 Java 语言、CORBA、WebService 等分布计算技术来实现并行编程, 如 JavaParty、WebFlow、JET 等. 图 1 反映了实现这类计算的原理.

这类系统主要有两层结构: 中间件层实现 Client/Server 模式下的分布对象互操作, 上端的应用层主要包括两类对象: Master 对象和 Slave 对象. 用一组 Web 服务中的 Server 来实现 Mater/Slave 并行编程模式下的 Slave, 这些 Slave 能够执行一个并行程序的不同计算部分; 再用一个 Web 服务中的 Client 来实现 Mater/Slave 并行编程模式下的主控程序 Master, 由 Master 创建一组线程, 每个线程负责与一个 Slave 对象交互, 数据分配和运算结果回收. 最后由 Master 对计算结果进行合并. 在这类系统中, 并行计算中的 Master(Web 服务中的 Client)请求一组 Slave(Web 服务中的 Server)并行执行程序中的不同部分, 用这种方法实现 Master/Slave 并行编程非常简单. 但仍然存在一些问题:

(1) 仅支持 Master/Slave 编程模式不足以表示各类并行计算问题.

(2) RMI 或 CORBA 技术创建分布对象的过程比较复杂, 需要生成 Skeleton 和 Stub 并手工启动, 不利于 Grid 环境和机群环境下的并行编程;

为解决第一个问题, 本文参考文献[2]提出了支持多种并行编程模式的对象组关系 JOGR. 为解决第二个问题, 相关项目[3]提出用 WebService 动态生成和查找服务的特性来支持 Grid 环境下的应用. 本文采用 GLUE 软件包[1]作为编程辅助手段正是出于同一目的.

2 GLUE 简介

GLUE 是由 The Mind Electric 公司开发的一个 Java 扩展包, 使用 GLUE 可以不考虑机器的位置分布性和异构性方便地构造、配置并无缝地调用网络上的各类服务, 因为 GLUE 支持 HTTP, XML, SOAP, WSDL 和 UDDI 等标准. 使用 GLUE, 不必象 CORBA 和 RMI 那样需要使用 stub 生成器和命令行工具来创建和调用远程对象, 仅需要将服务端对象发布出去并在客户端对服务端对象进行绑定即可.

举个例子来说, 下面两行代码可以发布一个 Java 对象供 SOAP 客户端调用:

收稿日期: 2001-12-26; 修回日期: 2002-04-28
基金项目: 国家自然科学基金(NO. 69773014)

```
public static void main( String[ ]args) throws Exception
{   HTTP. startup(/ http: / / localhost: 8004/ glue0 );
   Registry. publish(/ urn: exchange0 ,
new Exchange(    ));
}
```

下面这四行代码是一个客户端调用 bind 方法绑定指定路径上的 WSDL 文件提供的服务并调用服务提供的方法.

```
public static void main( String[ ]args) throws Exception
{   String url= / http: / / localhost: 8004/ glue/ urn: exchange. wsdl0 ;
IExchange exchange= (IExchange) Registry. bind ( url, IExchange.
class);
float rate= exchange. getRate(/ usa0 ,/ japan0 );
System. out. println(/ usa/ japan exchange rate= 0+ rate);
}
```

3 JOGR 工作原理

图 2 是 JOGR 的工作原理. 图中的方框表示对象, 带箭头的线条表示对象之间的请求/ 应答关系. 设计 JOGR 的目标在于能够用简便的方法支持多种模式(如分而治之、阶段并行等)的并行计算, 一个 JOGR 服务包括:

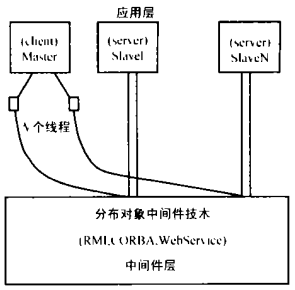

图 1　实现 Master/ Slave 并行计算模式的系统

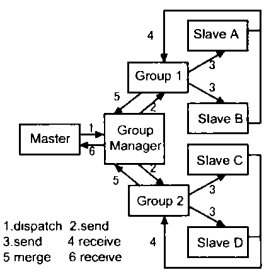

图 2　JOGR 的工作原理

* 一个 Master: 用户通过 Master 提交计算任务, 在 Master 端用户仅需简单描述数据划分和计算结果回收策略, 不须负责数据分派以及 Slave 的管理.

* 一个 GroupManager: GroupManager 负责接受 Master 提交的任务, 并根据 Master 中定义的数据划分策略动态查找 Group, 将不同功能的计算部分分配给不同的 Group, 最后根据 Master 中定义的结果回收策略从各个 Group 中搜集计算结果并返回给 Master.

* Group 对象: 所有的 Group 对象皆由 GroupManager 进行管理, 每个 Group 管理几个 Slave 对象, 同属一个 Group 的 Slave 对象执行的代码相同, 不同 Group 的 Slave 对象执行的代码不同.

* Slave 对象: Slave 对象负责接受 Group 传递来的划分好的数据以及运行代码, 将计算任务执行完成后返回给它的管理者: 某个 Group 对象.

例如, 要计算公式 $\frac{\sqrt{u_1^2+u_2^2+,+u_n^2}}{(A[1]*B[1]+A[2]*B[2]+,+A[m]*B[m])}$, 程序员仅需要在 Master 中指出将计算分成两组, 第一组计算 $x=u_1^2+u_2^2+,,u_n^2$, 第二组计算 $y=(A[1]*B[1]+A[2]*B[2]+,+A[m]*B[m])$, 最终的计算结果为 $\sqrt{x}/y$, 并指出各组计算的数据划分策略. 而数据的划分和各组子任务的计算和管理由 GroupManager、Group 和 Slave 对象协同操作, 实现相对于用户透明的并行计算. 这种服务的优点在于: 易编程, 可实现动态任务分配和负载平衡, 支持多种并行编程模式. 很显然地, 除了 Master/ Slave 模式外, 图 2 中的 JOGR 能支持/ 分而治之0 编程模式; 如果循环地执行图 2 中第 1 到第 6 步进行多级计算, 就能支持/ 阶段并行0 编程模式.

4 JOGR 的设计和实现

411 数据划分策略

为支持数据的动态分发, JOGR 通过 DataStructure 类来表示一个数据集及其划分策略.

```
public class DataStructure
{   static final int BROADCAST= 0;
static final int PER_ ELEMENT= 1;
static final int SAME_ SIZE= 2;
static final int DIFFERENT_ SIZE= 3;
  / / data dispatching strategy, range from 0~ 3
  int dds;
  int num_ of_ slave;
  String datatype; / / it can be any java data type
  byte[ ] data; / / the date set
  int[ ] data_ size;   }
```

图 3　DataStructure 类

这个数据结构支持以下数据划分策略:

* BROADCAST: 将整个数据集广播给组内所有的 Slave.

* SAME_ SIZE: 根据当前 Group 内 Slave 数目将数据平均分配给各个 Slave.

* PER_ ELEMENT: 将数据集中第一个元素给第一个 Slave, 第二个元素给第二个 Slave, 以此类推到组中的最后一个 Slave, 如果数据集中有多于 Slave 数目的元素, 对剩下的元素再按上面规则重新分配直到数据集中所有元素全部分配结束.

* DIFFERENT_ SIZE: 这种方式支持给同一组内各个 Slave 分配长度不同的数据元素, 通过 num_ of_ slave 变量和 data_ size[ ] 数组描述分配方案, data_ size[i] 表示分配给 Slave i 的数据元素的个数.

412 计算结果合并策略

为了对各个 Slave 的计算结果进行自动收集和合并, 采用 Recv_ DataStructure 类表示运算结果集及其自动合并策略, 归约后的结果存放在 result[ ] 中.

```
public class Recv_ DataStructure   {
String recvtype; int recvcounts;
byte[ ] recvbuf;
static final int NOTHING= 4;
static final int MAX= 0; static final int MIN= 1;
```

```
static final int SUM= 2; static final int PROD= 3;
int merge_ strategy; / / range from 0~ 4
String result_ type; / / type of merged results
int result_ counts; / / number of merged results
byte[] result;  }
```

图 4 Recv_ DataStructure 类

413 JOGR 中主要类的设计及它们之间的关系

JOGR 的主要类有: (1) GroupManger 类, 类中的 registerSlave (Slave s) 方法用于注册 Slave 并将 Slave 加入某一个 group 中, findAGroup() 方法用于查找 group 对象; (2) 作为一个 Master 类对象, 必须定义 disp_ a_ group_ of_ task() 方法用于发布任务, receive (int groupID, String result_ type, int result_ counts, byte[] result) 方法用于返回计算结果; (3) 作为一个 Group 类对象, 提供 dispatch(Master m, DataStructure ds, Recv_ DataStructure rds, byte[] code, int codeID) 方法提交一组程序相同处理数据不同的计算任务, joinGroup (Slave s) 方法把某个 Slave 加到该组中, leaveGroup(Slave s) 方法使某个 Slave 离开该组; (4) 作为一个 Slave 对象, 提供 beginTask(int groupID, int codeID, byte[] code) 方法启动一个计算任务.

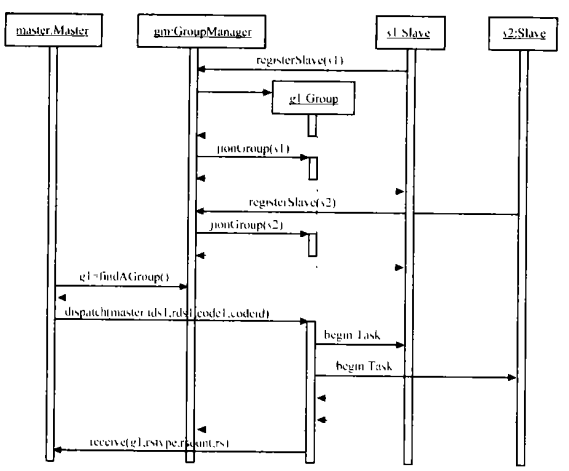

图 5 JOGR 中各个对象之间互操作的顺序图

图 5 中的 UML 顺序图表示 JOGR 进行一次组服务期间各个对象之间的互操作关系. 第一步, Slave1 和 Slave2 分别向 GroupManager 注册, GroupManager 创建 g1 组并将 Slaver1 和 Slave2 加入该组中; 第二步, Master 从 GroupManager 处获取 g1 并向 g1 派送一个任务; 第三步, g1 根据用户定义的数据划分策略向 Slave1 和 Slave2 派送计算子任务; 第四步, g1 从 Slave1 和 Slave2 处回收运算结果并按照用户定义的策略作合并; 第五步, Master 从 g1 处获得整个组计算的结果.

5 并行程序的实现和性能测试

本次实验采用了六台 Pentium Ó (667MHZ CPU, 128MB 内存, Windows NT 或 Win 98 操作系统), 由 100Base2T 网络接口与 BayStack 350T 交换机相连构成一个机群系统. 各台机器上均安装了 JDK1.3 和 Java 扩展包 GLUE. 为了检验 JOGR 的功能, 编制的 JOGR 并行程序如下.

511 P 值计算的并行程序

用 JOGR 计算 P 值只需在 Master 中将计算任务分派给一组 slave 即可, 数据分配策略为 BROADBAST, 结果合并策略为 SUM, 每个 slave 执行相同的代码计算部分矩阵分块, 然后由 Group 对象将计算结果相加返回给 Master 即可. 表 1 为组内不同 slave 数目的情况下计算指定精度 P 值的计算时间.

表 1 组内不同 Slave 数目情况下 P 值计算时间和加速比

Slave 对象数 目	P_{10000}		P_{35000}	
	时间(s)	加速比	时间(s)	加速比
1	25.776	-	305.386	-
2	13.933	1.85	161.580	1.89
3	10.607	2.43	120.706	2.53
4	8.235	3.13	91.984	3.32

表 2 不同 Slave 数目情况下字符串并行程序运算时间和加速比

Slave 对象数 目	总的运行时间(s)	加速比	Slave 运行时间(s)	
			最大值	加速比
1	2873.36	-	2821.77	-
2	1720.57	1.67	1631.08	1.73
3	1265.82	2.27	1205.89	2.34
4	954.60	3.01	892.97	3.16
5	780.81	3.68	710.77	3.97
6	689.06	4.17	636.97	4.43

512 字符串匹配并行求解

字符串匹配是从字符串文件中搜索给定模式的子串, 首先 Master 将搜索模式(一个固定长度的子串) 和部分字符串文件发送给各个 Slave, 各个 Slave 在分配到的部分字符串文件中搜索给定模式. 搜索时按照下列公式: $a_k * b^k + a_{k-1} * b^{k-1} + , + 1$ 进行计算, a_k 代表字符对应的 ASCII 值, b 表示字符集的大小, k 表示要搜索子串的长度, 将被检查的子串按上面公式计算出的值和目标子串按上面公式计算出的值相减所得差的绝对值反映了两个子串之间的匹配程度, 这个值越小说明被检查的子串越接近于要查找的子串. 使用 JOGR 进行字符串查找只需在 Master 端将计算任务分派给一组 slave, 数据分配策略为 SAME_ SIZE, 结果回收策略为 MIN. 表 2 为采用不同 Slave 数目在一个 150Kbytes 的字符文件中对一个存储在 12Kbytes 字符文件中的子串进行匹配的运算时间. 在表中/ 总的运行时间0 指从 Master 开始分派任务直到总的计算结果返回, 其中包括文件的传输时间, / Slave 运行时间0 指一个 Slave 计算部分任务时所花的时间, 这里我们取的是几个 Slave 运行时间中最大值.

从表 1 和表 2 可以看出, JOGR 能用简便的编程方式来解决计算密集型任务的并行计算并获得有效的加速比.

6 结论

本文主要介绍了 Java 对象组关系 JOGR 的设计和实现, 利用 WebService 的关键技术解决了通信透明性和对象互操作的问题. 在此基础上建立的 Java 对象组关系能方便地进行透

明的并行编程,实验结果表明 JOGR 能为并行程序的开发运行提供有效的支持.

将来的工作包括:(1)用 XML 语言对 JOGR 进行标准定义 (2)扩充目前支持集群计算环境的 JOGR 使其支持 Grid com2 puting 计算环境.

参考文献:

[1] The Mindelectic Corporation. GLUE API[Z]. 2001. http://www.the2mindelectric.com/products/glue.

[2] Markus Aleksy, Axe Korthaus. A CORBA2based object group service and a join service providing a transparent solution for parallel program2 ming[A]. International Symposium on Software Engineering for Parallel and Distributed Systems[C]. Proceedings, Limerick, IRELAND, 2000. 123- 134

[3] Ian Foster, Carl Kesselman, et al. The physiology of the grid, An open grid service architecture for distributed systems integration[R]. http://www.globus.org.

作者简介:

邓倩妮　女,1973 年生,广西柳州人,1996 年获上海交通大学硕士学位,现为上海交通大学在职博士生,讲师,主要研究领域为网格计算,分布计算,并行处理.

陆鑫达　男,上海交通大学计算机科学与工程系教授,博士生导师,1964 年哈尔滨工业大学计算机专业研究生毕业,1979~ 1981 年为英国纽卡舍尔大学计算机系访问学者,1987~ 1990 年为德国 GMD2FIRST 计算机研究所和柏林工业大学客座首席科学家,主要研究领域为异构网络计算,高性能计算,网格计算.