## 1　网格简介

### 1.1　网格概念

网格的概念源于电力网格，其含义是集合计算能力、数据库、通信网络、传感器、软件等资源，灵活共享，相互协作，继而形成的分布式基础设施[1]。从网络到网格，将会是信息技术的又一次飞跃。提出网格概念是为了建立一个统一的、自治而安全的分布式系统。网格提供的组织环境、统一调度、安全策略、容错机制等，将会给人们展示出一个强大而可靠的系统平台。

### 1.2　以服务为核心的传统网格体系

网格体系结构分为 4 个层次，从下往上分别是基础结构层、资源和连接性协议层、集合服务层、用户应用层，如图 1 所示[1]。整个架构以服务为核心[2~3]，通过实行各种资源组织方式和安全机制，使基础设施中的各种资源[4]，如计算机、存储介质、网络和传感器等成为可灵活使用的服务。服务请求方则通过查询目录代理得到适当的服务。原本大量的异质多态的网络资源形成了一个统一而强大的服务体系。

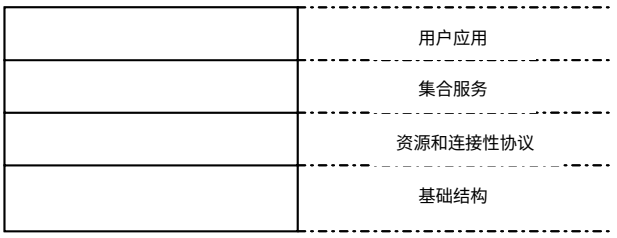


**图 1　网格体系结构**

## 2　新的基于请求的网格体系构想

### 2.1　移动网格须具备的特征

(1)脱机式计算

由于系统中的大多数成员都具有移动性，无法保证任何时间都能有效地接入通信网络中，也不一定会在长时间的移动过程中始终开着计算机设备，因此移动网格必须充分支持间断的通信。

(2)灵活的接入、退出机制

由于移动设备在移动的过程中会遇到各种复杂的情况，必然会经常地接入或退出网格系统，因此移动网格系统要支持灵活的接入、退出机制。

(3)虚拟的组织和成员

由于接入到网格系统的是各式各样的资源，它们是动态的、异质的。为了使所有的接入主体实现统一性，集合成为一个可用的整体，网格系统要支持虚拟的组织和成员架构。

### 2.2　新的基于请求的网格体系

为了适应移动网格的各种需求，本文以另一种思维方式提出了基于请求的网格体系。该体系以请求为核心，突破以往以服务为核心的网格组建模式。在基于请求的网格体系中，每个任务被描述成一条记录，这些记录可以通过作为中介的区域代理[5~6]送到满足资源要求的主机中执行，以此满足移动网格的脱机式计算等要求。基于该体系的设计思想，本文提出了移动网格模型。

## 3　移动网格模型

### 3.1　移动网格模型组织、成员的架构

模型中的组织成员由以下 4 部分构成（图 2）：

(1)区域：根据物理位置而划分的网格局部范围；

(2)区域代理：每个区域中用于承载请求的任务池，及起协调调度作用的主机；

(3)请求方：在区域中请求协助完成任务的主机；

(4)服务方：在区域中提供服务、帮助完成任务的主机。

区域代理通常是通信机站或者是与之有固定线路连接的计算机，它是整个区域的协调者，在区域内起着重要的作用；而请求方和服务方是各类主机。一个主机可以相继充当请求

**作者简介：**高理文(1981－)，男，硕士，主研方向：分布式系统；网格并行计算；姚耀文，教授

方和服务方，也可以同时扮演这 2 种角色。

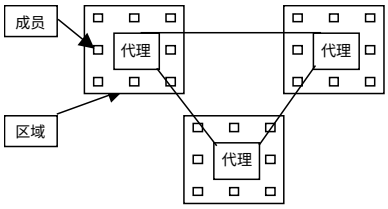


**图 2　组织成员架构**

### 3.2　虚拟组织、成员的关系

**定义 1** 代理从属关系。如果主机 a 在区域 d 中作为区域代理，承载该区域的请求任务缓冲池，负责该区域内的协调工作，那么就称主机 a 与区域 d 具有代理从属关系，记为：$a \overset{a}{\subset} d$。

**定义 2** 请求可达关系。如果请求方 r 可与区域代理 a 建立通信连接，并且可以把请求任务上传到任务池和下载已完成任务的结果，那么就称请求方 r 与区域代理 a 具有请求可达关系，记为：$r \overset{r}{\rightarrow} a$。

**定义 3** 服务可达关系。如果服务方 s 可与区域代理 a 建立通信连接，并且可以把请求任务下载到本地执行，然后把结果送到该代理中，那么就称服务方 s 与区域代理 a 具有服务可达关系，记为：$s \overset{s}{\rightarrow} a$。

**定义 4** 连通关系。如果区域代理$a_i$与区域代理$a_j$可以建立连接，并且可以互相交换任务缓冲池中的任务以及任务的结果，那么就称区域代理$a_i$与区域代理$a_j$具有连通关系，记为：$a_i \overset{L}{\leftrightarrow} a_j$。

**定义 5** 任务委托关系。如果请求方 r 通过本地的区域代理或者包括本地区域代理在内的若干个具有连通关系的区域代理的任务交换操作，可以间接地把请求任务送至服务方 s，服务方 s 完成该任务后，再经过相反的路径，返回结果给请求方 r，那么就称请求方 r 与服务方 s 具有任务委托关系，记为：$r \overset{rs}{\rightarrow} s$。

移动网格可以描述成一个七元组 MG=(D, A, R, S, FR, FS, F)。其中，D=(district1, district2,…)是区域的集合；A=(agent1, agent2,…)是整个网格系统中所有代理的集合；R=(request1, request2,…)是整个网格系统中所有请求方的集合；S=(server1, server2, … ) 是整个网格系统中所有服务方的集合；$FR=\{(r,a)\mid r \overset{L}{\rightarrow} a\}$，$r \in R$，$a \in A$，是网格系统中的请求可达关系(如定义 2)；$FS=\{(s,a)\mid s \overset{s}{\rightarrow} a\}$，$s \in S$，$a \in A$，是网格系统中的服务可达关系(如定义 3)；$F=\{(r,s)\mid r \overset{rs}{\rightarrow} s\}$，$r \in R$，$s \in S$，是网格系统中的任务委托关系(如定义 5)。

得出如下推论：

(1) $a \in A, a \overset{a}{\subset} d_i$，则对于 $\forall d_j \in D$，$d_j \neq d_i$，有 $a \overset{a}{\not\subset} d_j$（一个代理只能从属于一个区域）；

(2) $\forall a_i, a_j \in A, d \in D$，若满足 $a_i \overset{a}{\subset} d$，$a_j \overset{a}{\subset} d$，则 $a_j = a_i$，或者 ($a_j$ mirror $a_i \| a_j$ mirror $a_i$)（X mirror Y 表示 X 是 Y 的镜像站点，一个区域只能有一个正式代理站点和若干镜像代理站点）；

(3) $\forall a_i, a_j \in A$，若 $a_i \overset{L}{\leftrightarrow} a_j$ 且 $\exists r \in R, s \in S$ 满足 $r \overset{L}{\rightarrow} a_i$，$s \overset{s}{\rightarrow} a_j$，则 $r \overset{rs}{\rightarrow} s$（可定义 $a_i \overset{L}{\leftrightarrow} a_j$ 时，$(r, a_i) \dot{\times} (s, a_j) = (r, s)$ 为连接运算；$\dot{\times}$是连接运算符）。

### 3.3　移动网格模型的任务处理

任务处理借用了邮递员投递挂号邮件的机制。在每个区域代理中，设置了一个请求任务池（任务信箱）。请求方把要执行的任务放于该池中，空闲的服务方于池内寻找自己可以胜任的任务，若找到，则把该任务下载到本地执行，完成后把结果提交到任务池相应的位置中。整个过程中，请求方和服务方不必同时接入通信网络。此外，该模型中的各个请求方和服务方都是灵活而独立的。由于很多算法支持断点续传，因此各方主机甚至可以在任务的上传和下载的过程中退出网格系统。任务的处理如图 3 所示。

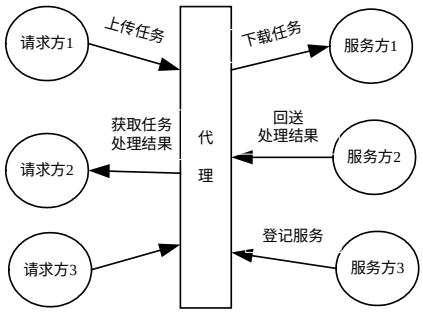


**图 3　任务处理示意图**

### 3.4　请求任务的表示

如图 4 所示，任务的数据模式从上往下分别是：请求方信息表，服务方信息表，请求任务表，其中包含以下字段：类型号(TypeId)，程序文件(ProFile)，参数 XML(ParaXml)，结果 XML(ResXml)，资源需求(Req)，请求主机名(ReName)，服务主机名(SerName)。


**图 4　任务的数据模式**

(1)TypeId = Hash(ProFile+ReName)，一个请求任务的类型由程序和请求主机号联合起来的哈稀值唯一确定。TypeId 由请求方生成。当请求方重复发出此任务时，无须再次进行哈稀运算，只须使用以前的运算结果即可。

(2)区域代理 3 个表都可以访问；请求方只可以访问请求方信息表和请求任务表；而服务方则只能访问服务方信息表和请求任务表。把这些字段分开建立 3 个表是为了实现服务方和请求方的相互透明性。请求方关心服务质量而不必理会由谁来完成此任务，服务方只负责完成所下载的任务，而不必知道这个任务由谁提交的。而中间的服务质量评定、服务计费和争议仲裁等由区域代理这个第三方来完成。

(3)资源需求描述了完成该项任务所需要的最少的资源，以便服务方匹配资源，寻找可完成的任务。

(4)每个程序文件均包含一个 ParaRead 的函数和一个 ParaWrite 的函数，分别负责解读 ParaXml 并把结果写道 ResXml 中。

## 4 移动网格模型原型系统的实现

原型系统分为请求方和服务方、区域代理 3 个大的模块。设计原型系统主要是为了证实移动网格模型的可行性，以及为正式系统的开发做准备，所以，按照模型的思想做了简单的设计实现。图 5 是原型系统的程序设计。

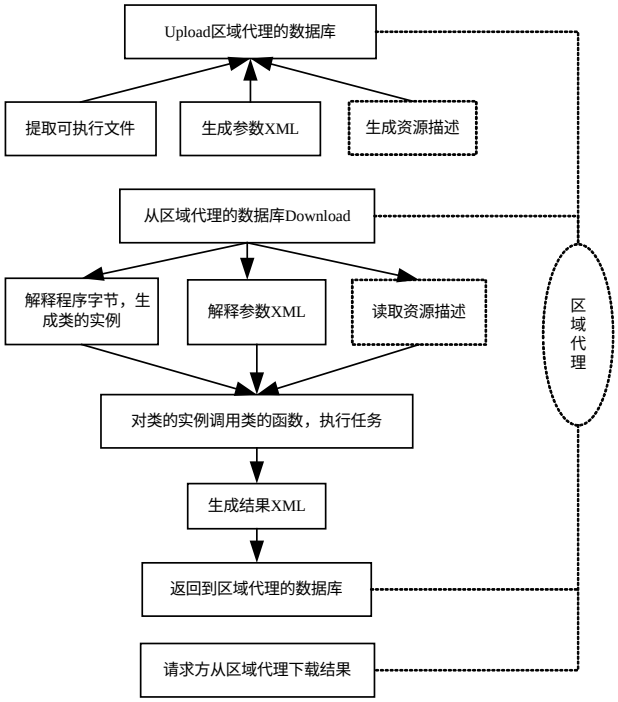


**图 5 原型系统设计**

本原型系统采用移植性很好的 Java 语言设计。在原型系统的编码实现过程中，主要存在 2 个技术难点：(1)如何智能地提取可执行的.class 文件的字节（提取可执行文件子模块）；(2)如何把从数据库下载的可执行程序字节转化成通常情况下的类，建立这个类的对象，以实现函数的调用来完成任务（解释程序字节，生成类的实例子模块）。解决了这 2 个技术难点，就可以实现程序的远程转载和执行的功能。其形式就像一些杀毒软件更新杀毒库，有的杀毒软件在更新病毒库后，连界面都改变了，也是这个道理。

提取可执行文件子模块部分实现代码：

```
byte[ ] getExecData(String name) throws IOException {
byte[ ] data=null;    //定义一个字节数组变量，用以返回可执行
//字节数据
int length = 0;    //定义文件长度，初始化为 0
try {
length = this.FileLength(name);    //调用成员函数，得到.class
//可执行文件长度
URL url=this.getClass().getResource(name);    //获取可执行文件
//的 URL 地址
InputStream inputStream=url.openStream();    //根据 URL 打开输
//入流
data = new byte[length];    //赋予字节数组实际的内存空间
inputStream.read(data);    //从输入流读可执行字节到 data 数组
inputStream.close();    //关闭输入流
}
catch (Exception e)
{
//捕获异常
e.printStackTrace();    //打印调试栈，便于找到从哪里出现异常
System.out.print(e.getMessage());    //打印异常信息
System.out.print("error in getClassData");    //补充说明异常是出
//自于软件中的哪个函数
}
return data;    //返回可执行字节
}
```

解释程序字节子模块部分实现代码：

```
public Object getClassObject(byte[] data)
{
Object myObject=null;    //定义一个对象变量，初始化为空
try {
Class myClass=this.defineClass(data,0,data.length);    //从可执行
//字节载入类
this.resolveClass(myClass);    //运行时解释这个类，Java 是解释
//型语言
myObject=myClass.newInstance();    //实例化成一个对象
}
catch (Exception e)
{    //捕获异常
e.printStackTrace();    //打印调试栈，便于找到从哪里出现异常
System.out.print(e.getMessage());    //打印异常信息
System.out.print("error in getClassObject");    //补充说明异常出
//自于软件中的哪个函数
}
return myObject;
}
```

虽然整个系统的任务都是动态载入然后解释运行，但由于普通的 Java 程序其实也经过相同的过程，只不过这些工作平时由 JDK 隐式完成了，因此系统的运行速度和普通的需要读写数据库的 Java 程序相近。

## 参考文献


1 Foster I, Kesselman C. The Grid 2: Blueprint for a New Computing Infrastructure[M]. [S. l.]: Morgan Kaufmann Publisher, 2003.

2 Buyya R, Abramson D, Giddy J. A Case for Economy Grid Architecture for Service-oriented grid Computing[C]//Proc. of the 15th International Conference on Parallel and Distributed Processing Symposium. 2001.

3 Abawajy J H. Grid Accounting Service Infrastructure for Service-oriented Grid Computing Systems[C]//Proc. of the 1st Int'l Workshop on Scientific Applications of Grid Computing. 2004.

4 Luo Junzhou, Ji Peng, Wang Xiaozhi, et.al. Resource Management and Task Scheduling in Grid Computing[C]//Proc. of the 8th International Conf. on Computer Supported Cooperative Work in Design. 2004.

5 Ding Shunli, Yuan Jingbo, Ju Jiubin. An Algorithm for Agent-based Task Scheduling in Grid Environments[C]//Proc. of the 3rd Int'l Conference on Machine Learning and Cybernetics. 2004.

6 Thompson C W. Agents, Grids, and Middleware[J]. Internet Computing, 2004, 8(5): 97-99.

7 Jin Xiaolong, Liu Jiming, Yang Zhen. Modeling Agent-based Task Handling in a Peer-to-peer Grid[C]//Proc. of the 2004 IEEE/WIC/ACM International Conference on Intelligent Agent Technology. 2004: 288-294.

8 Nahrstedt K, Steinmetz R. Resource Management in Networked Multimedia Systems[J]. IEEE Computer, 1995, 28(5): 52-63.