Multi Agent 在分布式测控系统动态任务调度中的实现

闫钧华,张焕春,经亚枝
YAN Jun-hua,ZHANG Huan-chun,JING Ya-zhi

南京航空航天大学 航天学院,南京 210016
College of Astronautics Engineering,Nanjing Univ. of Aeronautics and Astronautics,Nanjing 210016,China

YAN Jun-hua,ZHANG Huan-chun,JING Ya-zhi.Implementation of Multi Agent on dynamic task scheduling for distributed measurement and control system.Computer Engineering and Applications,2009,45(2):219-222.

Abstract: The implementation of Multi Agent is a key technology of a multi-agent-based dynamic task scheduling algorithm for distributed measurement and control system.The Java is adopted to realize Multi Agent of dynamic task scheduling for distributed measurement and control system according to function of Multi Agent.In the multi-agent-based dynamic task scheduling algorithm,the tasks are dynamically migrated by the mobile Agent in the running process of the system,according to the current status of load on each host.The Aglets system is used to exploit and run the mobile Agent so that the system efficiency is promoted effectively and dynamic task scheduling is attained.

Key words: distributed measurement and control system;dynamic task scheduling;Multi Agent;mobile Agent;Java;Aglets system

摘 要: Multi Agent 实现是基于 Multi Agent 的分布式测控系统动态任务调度算法实现的关键技术。采用 Java 作为开发工具,根据 Multi Agent 的功能,详细论述了 Multi Agent 在分布式测控系统动态任务调度中的实现。基于 Multi Agent 的动态任务调度算法根据各主机的负载状态,在系统运行过程中利用移动 Agent 动态迁移任务。文中详细论述了利用 Aglets 系统开发和执行移动 Agent,从而有效地提高了系统效率,实现了动态任务调度的目标。

关键词: 分布式测控系统;动态任务调度;Multi Agent;移动 Agent;Java;Aglets 系统
DOI: 10.3778/j.issn.1002-8331.2009.02.063 **文章编号:** 1002-8331(2009)02-0219-04 **文献标识码:** A **中图分类号:** TP316.4

分布式测控系统通过各处理机的协同合作,使其具有自治性、并行性、扩展性、透明性等一系列优点[1]。在分布式测控系统中,任务的到达是不可预测的动态过程,每个结点的负载大小是动态变化的。在某一时刻,一些计算机的负载极重,而另外一些计算机的负载却极为空闲,所以应根据系统当前的负载状况[2],采取有效的动态任务调度策略来平衡各结点(机)的负载,把当前重载计算机上的任务传送到轻载计算机上执行,从而使任务尽可能地并行执行,提高整个系统的资源利用率及效率。近年来,国内外积极开展了用 MAS(Multi Agent System)[3-4]来解决分布式测控系统动态任务调度问题的研究。Agent 是指分布式测控系统中模拟人类行为和关系,具有一定智能并能够持续自主运行和提供相应服务的计算实体,它具有自主性、交互性、反应性和主动性的特征[5-6]。如何将重载计算机上的任务迁移到轻载计算机上执行,是分布式测控系统动态任务调度的关键问题。20 世纪 90 年代初,随着互联网技术的迅速发展,一种新的网络技术,即移动 Agent 技术成为当前研究的一个新热点。移动 Agent 实际上是 Agent 技术与分布式计算技术的混血儿,它是一个能在异构网络中自主地从一台主机迁移到另一台主机,并可与其他 Agent 或资源交互的程序。本文将 MAS 理论和方法用于复杂的分布式测控系统动态任务调度问题求解,实现了基于 Multi Agent 的分布式测控系统动态任务调度算法,并且通过移动 Agent 实现了将重载机上的任务迁移到轻载机上执行。

1 基于 Multi Agent 的分布式测控系统体系结构

分布式测控系统采用 Client/Sever 体系结构,整个系统由服务器和主机构成,可以基本解决各种分布式计算问题,但也有一些缺点和不足。Client/Sever 体系结构其带宽耗费严重、网络负载不均衡、容错能力较差、响应速度较慢。分布式测控系统运行于开放环境中,系统中的有关各方为了协同工作,需信息共享,系统不仅需要考虑信息源的集成、安全性、网络带宽,还必须有效地面对各信息源的动态变化。此时,Client/Sever 难以胜任,而分布式对象技术和分布式人工智能技术的结合,为解决这一困难提供了有效途径。为了克服 Client/Sever 方案的不足之处,系统中引入了若干不同类型的静止 Agent 和移动 Agent。各种静止 Agent 分布在 Client 和 Sever 上,完成自己相应的功能。移动 Agent 是一种全新的分布式计算工具,它在重载计算机上产生,携带迁移任务移动至轻载计算机上执行。通过各静止 Agent 和移动 Agent,系统可以节省通信带宽,减少网络传输,并且可以成功解决现代分布式计算中日益突出的移动计算问

作者简介: 闫钧华(1972-),女,博士,讲师,研究方向为分布式系统、计算机测控技术;张焕春(1940-),男,教授,博士生导师,研究方向为智能仪器、计算机测控技术等;经亚枝(1942-),女,副教授,研究方向为微机应用、计算机测控技术等。

收稿日期: 2008-01-02 **修回日期:** 2008-03-17

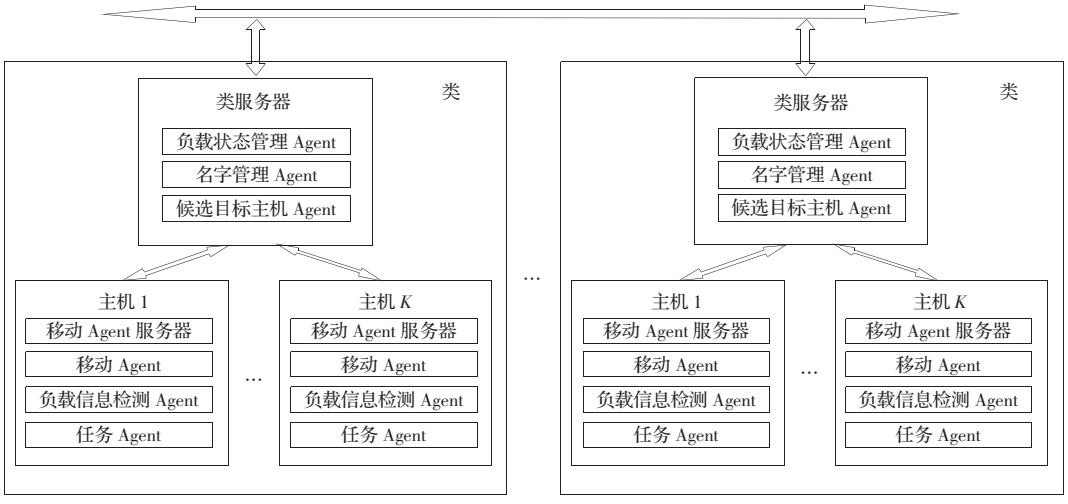

图 1　分布式测控系统体系结构

题,实现系统的负载平衡。

分布式测控系统中有大量不同种类的任务,它们大致可分为:测试任务、控制任务、分析决策任务、操作任务、显示打印任务、保护任务等 6 类。根据各主机上运行的任务,将主机归类。每一类都有一台类服务器,类服务器负责为本类主机提供负载状态管理、任务命名和名字管理、候选目标主机管理等。这些服务是通过运行在类服务器上的各种功能不同的 Agent 来实现的。在各主机上也运行着若干种 Agent,它们完成了各不相同的功能。分布式测控系统体系结构如图 1 所示。

在分布式测控系统体系结构中,类服务器上的负载状态管理 Agent、名字管理 Agent、候选目标主机 Agent,采用了多 Agent 系统(MAS)的结构形式中的分布式,如图 2 所示。

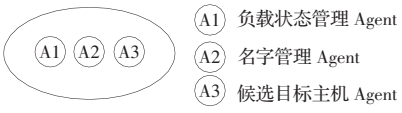

图 2　分布式 MAS

主机上的移动 Agent 服务器、移动 Agent、负载信息检测 Agent、任务 Agent,采用了多 Agent 系统(MAS)的结构形式中的混合式,如图 3 所示。

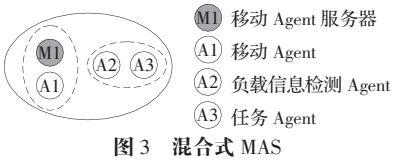

图 3　混合式 MAS

2　分布式测控系统 Multi Agent 的功能

Multi Agent 在分布式测控系统中的分布如图 1 所示。在类服务器上有:负载状态管理 Agent、名字管理 Agent、候选目标主机 Agent。在主机上有:移动 Agent 服务器、移动 Agent、负载信息检测 Agent、任务 Agent。它们的功能分别为:

(1)负载状态管理 Agent:负责收集类中各主机的负载状态信息,即获取各主机的负载状态指标<$Load_C$,$Load_M$>。在负载状态管理 Agent 中保存有各主机的负载上限<$Load_C Max$,$Load_M Max$>与负载下限<$Load_C Min$,$Load_M Min$>值,并且可根据各主机负载状态指标,判定各主机的负载状态。

(2)名字管理 Agent:负责为类中各主机上的所有任务提供命名和名字管理,采用全局的、与位置无关的命名原则,并且任务迁移后为任务提供定位服务。

(3)候选目标主机 Agent:负责记录系统中轻载主机。

(4)移动 Agent 服务器:移动 Agent 服务器是移动 Agent 的运行平台,为移动 Agent 提供基本服务,包括创建、传输、执行等。移动 Agent 的移动和任务求解能力很大程度上决定于移动 Agent 服务器所提供的服务,一般应包括生命周期服务、事件服务、安全服务、应用服务、并行服务等基本服务。

在每一个主机上运行着一个移动 Agent 服务器,协助用户自动地生成移动 Agent。移动 Agent 服务器允许多个移动 Agent 的运行,这可通过让不同的移动 Agent 在不同的端口地址上运行来实现。移动 Agent 可以携带迁移任务移动至轻载主机,利用轻载主机所提供的计算环境及资源,利用本地操作的优势快速而高效地完成迁移任务。当系统中的重载主机需要向轻载主机迁移任务时, 请求移动 Agent 服务器为其生成一个移动 Agent,请求原语格式如下[7]:

```
Request-Create-Agent{
SourceHost Address 1;
GoalHost Address2;
UserTask TaskNumber;}
```

其中,SourceHost 后的参数给出重载主机的地址,GoalHost 后的参数给出轻载主机的地址,UserTask 后的参数给出迁移任务的任务号。

(5)移动 Agent:携带迁移任务移动至轻载主机,利用轻载主机所提供的计算环境及资源,利用本地操作的优势快速而高效地完成迁移任务。

(6)负载信息检测 Agent:每台主机上都运行有唯一的负载信息检测 Agent,负责检测主机的负载信息,即获得各主机的负载状态指标<$Load_C$,$Load_M$>的值。

(7)任务 Agent:记录到达主机的每一个任务,并为它们排队和编号,当一个任务执行完后,就从任务队列中删除这个任务。当需要迁移任务时,就按 FIFO(First In First Out)原则迁移任务队列中的任务。

3　基于 Multi Agent 的分布式测控系统动态任务调度算法

本分布式测控系统采用接收者启动动态任务调度策略。当

一个主机成为轻载机时,它就向本类服务器发出请求,启动动态任务调度算法。如果本类中有重载机,则将重载机上的任务利用移动 Agent 迁移到轻载机上。如果本类中没有重载机,则向其它类服务器发出请求,启动动态任务调度算法。如果其它类中有重载机,则将重载机上的任务利用移动 Agent 迁移到轻载机上。如果其它类中没有重载机,则中止请求,等待一段时间后,再重新发出任务请求。

类服务器上的负载状态管理 Agent,每隔一定的周期时间t,向本类内各主机上的负载信息检测 Agent 发送查询消息,收集本类内各主机的负载信息,并根据各主机的负载信息判定各主机的负载状态。当某主机的负载状态为轻载时,即启动动态任务调度算法。分布式测控系统动态任务调度算法流程图如图 4 所示。

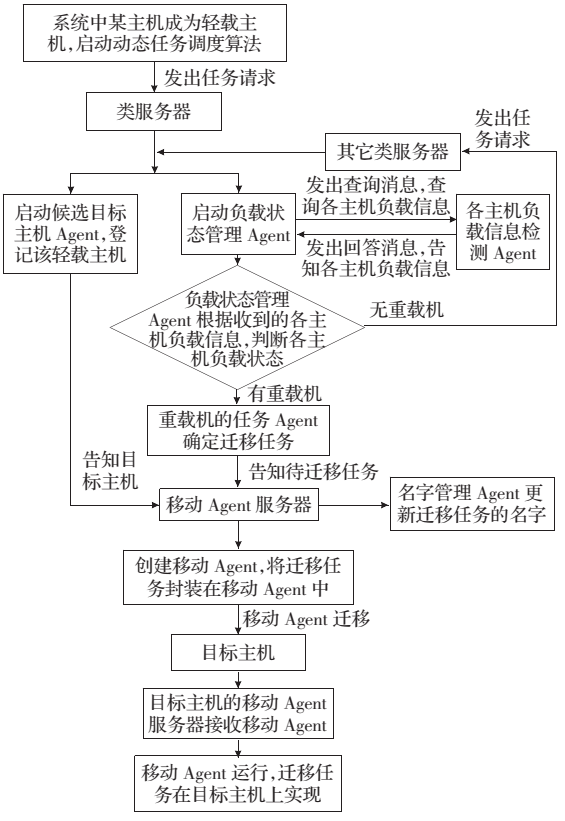

图 4　分布式测控系统动态任务调度算法流程图

若系统中同时有若干台轻载主机时,可以将它们都记录在候选目标主机 Agent 中,将重载机上的不同任务同时迁移至各轻载主机,从而很好地实现系统动态任务调度的目标。

4　Multi Agent 在分布式测控系统动态任务调度中的实现

基于 Multi Agent 的分布式测控系统动态任务调度算法能够有效地将重载机上的任务迁移到轻载机上执行,达到系统动态任务调度的目标。其中的关键是根据分布式测控系统中 Multi Agent 的功能,在分布式测控系统动态任务调度中实现 Multi Agent。本系统采用 Java 作为开发工具[8],这是因为 Java 具有面向对象、分布式、健壮安全、可移植等多种特性;Java 提供了一个异构和分布式的体系结构,这正与分布式测控系统的出发点不谋而合;Java 语言的多线程机制使得同一台主机上可以有多个 Agent 独立运行,而内建的同步机制又使得这些 Agent 之间可以相互通信;Java 语言由于其内建的平台无关性、目标序列化、动态类装载等机制,为设计移动 Agent 提供了一些独一无二的特点。

4.1　负载状态管理 Agent

根据负载状态管理 Agent 的功能,它在分布式测控系统动态任务调度中的实现如图 5 所示。

```
//导入所需各类库
public class LoadStatusManagementAglet extends Aglet {
 public void run(){
  collectHostsLoadStatusInformation();//收集类中各主机的负载状态信息,获取各主机的负载状态指标 Load=<Load_C,Load_M>。
 judgeHostsLoadStatus ();//根据各主机负载状态指标, 判定各主机的负载状态。
  }
 public void collectHostsLoadStatusInformation()//收集类中各主机负载状态信息的方法
  {
    …}
 public int judgeHostsLoadStatus()//判定各主机负载状态的方法
  {
    …}
  }
```

图 5　负载状态管理 Agent 的实现

4.2　名字管理 Agent

根据名字管理 Agent 的功能,它在分布式测控系统动态任务调度中的实现如图 6 所示。

```
//导入所需各类库
public class NameManagementAglet extends Aglet {
 public void run(){
  Aglet.handleMessage(msg);//接收并处理收到的信息
  }
 public boolean handleMessage(Message msg)//消息处理方法
 {if(msg.sameKind("nameTask" )){
     nameTask();      //任务命名
     }else if(msg.sameKind("renameTask" )){
     renameTask();  //任务名字更新
     }else if(msg.sameKind("orientateTask" )){
     orientateTask();//任务定位
     }else{ return false;
     }return true;
 }
 public String nameTask()//任务命名的方法
  {
   …}
 public String renameTask()//任务名字更新的方法
  {
   …}
 public String orientateTask()//任务定位的方法
  {
   …}
 }
```

图 6　名字管理 Agent 的实现

4.3　候选目标主机 Agent

根据候选目标主机 Agent 的功能,它在分布式测控系统动态任务调度中的实现如图 7 所示。

```
//导入所需各类库
public class CandidateGoalHostsAglet extends Aglet{
 public void run(){
  Aglet.handleMessage(msg);//接收并处理收到的消息
  }
 public boolean handleMessage(Message msg)//消息处理方法
 {if(msg.sameKind("recordLightLoadHost" )){
     recordLightLoadHost();  //记录轻载主机
 }else if(msg.sameKind("tellGoalHost" )){
     tellGoalHost();          //告知目标主机
 }else{ return false;
 }return true;
}
 public String recordLightLoadHost()//记录轻载主机的方法
 {
 …}
 public String tellGoalHost()//告知目标主机的方法
  {
  …}
 }
```

图 7 候选目标主机 Agent 的实现

4.4 负载信息检测 Agent

根据负载信息检测 Agent 的功能,它在分布式测控系统动态任务调度中的实现如图 8 所示。

```
//导入所需各类库
public class LoadInformationMeasurementAglet extends Aglet{
 public void run(){
  measureHostLoadStatusInformation();//检测主机的负载状态信息,获得主机的负载状态指标 Load=<Load_C,Load_M>的值。
  Aglet.handleMessage(msg);//接收并处理收到的消息
  }
 public float measureHostLoadStatusInformation()//检测主机负载状态信息的方法
 {
  …}
 public boolean handleMessage(Message msg)//消息处理方法
{if(msg.sameKind ("inquireHostLoadStatusInformation" )){
    tellHostLoadStatusInformation();//告知主机负载状态信息
 }else {return false;
 }return true;
}
 public float tellHostLoadStatusInformation(); //告知主机负载状态信息的方法
  {
  …}
 }
```

图 8 负载信息检测 Agent 的实现

4.5 任务 Agent

根据任务 Agent 的功能,它在分布式测控系统动态任务调度中的实现如图 9 所示。

4.6 移动 Agent

本分布式测控系统采用 Aglets 系统来开发和执行移动 Agent,从而实现将重载机上的任务迁移至轻载机上去运行。Aglets 是由 IBM 日本东京研究开发中心用纯 Java 开发的移动 Agent 技术,是目前最为成功和全面的系统[9-10]。Aglets 系统中的 Aglet 即是移动 Agent,它是可以在不同的主机之间移动的 Java 对象。当需要将重载机上的任务迁移至轻载机上执行时,移动 Agent—Aglet 服务器 Tahiti 创建移动 Agent—Aglet, 将迁移任务封装在 Aglet 中,并将此 Aglet 发送至目标主机。目标主机上的 Tahiti 服务器可接收并运行 Aglet, 从而使迁移任务在目标主机上执行。下面通过一个 MobileTaskAglet 来迁移一个数据处理任务 DataProcessing,如图 10 所示。DataProcessing 任务存放于 package task 中。

```
//导入所需各类库
public class TaskAglet extends Aglet{
 public void run(){
  recordTask();//记录到达主机的每一个任务
  Aglet.handleMessage(msg);//接收并处理收到的消息
  }
 public void recordTasks()//记录到达主机的每一个任务的方法
 {
 …}
public boolean handleMessage(Message msg)//消息处理方法
{if(msg.sameKind("migrateTask" )){
    tellMigratoryTask();  //告知迁移任务
}else if(msg.sameKind("completeTask" )){
    deleteTask();//任务执行完后,删除这个任务
}else {return false;
 }return true;
}
public String tellMigratoryTask()//告知迁移任务的方法
 {
  …}
public void deleteTask()//删除任务的方法
 {
  …}
}
```

图 9 任务 Agent 的实现

```
package task
//导入所需各类库
public class MobileTaskAglet extends Aglet{
 public void onCreation(Object init){  //初始化 MobileTaskAglet
 createGUI(); //创建 GUI 来控制 Aglet
 home=getAgletContext().getHostingURL().toString()//初始化变量,获得重载主机地址
 …}
 public void run(){
 createMobileTaskAglet(); //创建 MobileTaskAglet
 …}
 public void createMobileTaskAglet()  //创建 MobileTaskAglet 方法
 {AgletProxy p=getAgletContext().createAglet(null,
 "task.DataProcessing.MobileTaskAglet",null);//将 DataProcessing 任务封装于 MobileTaskAglet 中
 p.dispatch(new URL(addr));//将 MobileTaskAglet 派遣至轻载主机
}}
```

图 10 移动 Agent 的实现

5 结束语

分布式测控系统动态任务调度算法是利用多 Agent 系统的行为显现能力来实现的,采用移动 Agent 这一新型的分布式计算工具来迁移任务,节省了通信带宽,减少了网络传输,借助轻载主机的计算环境及资源,利用本地操作的优势快速而高效地完成迁移任务。动态任务调度实现的关键是根据分布式测控系统中 Multi Agent 的功能, 在分布式测控系统动态任务调度

(下转 236 页)