一个基于 OGSA 规范的网格计算过程表示模型*)

陈庆奎[1] 那丽春[2]

(上海理工大学 计算机工程学院 上海 200093)[1] (上海立信会计学院信息科学系 上海 201620)[2]

摘 要 提出一个基于网格服务树的网格计算过程的表示模型。对网格服务、网格计算过程、网格服务树、网格给出了形式化的定义,描述了一个网格计算过程表示模型,并给出了实现的机制和算法,最后给出利用 Web service 技术的模拟试验的方法。分析和实验结果表明,该模型有效地表示了基于 WAN 或 Internet 的网格计算过程,符合 OGSA 规范。该模型适合网格系统的构建、监测以及基于 Web service 机制的 Internet 的 QoS 管理等应用领域。

关键词 网格计算过程,网格,OGSA,网格服务树

Presentation Model of Grid Computing Process Based on OGSA

CHEN Qing-Kui[1] NA Li-Chun[2]

(Computer Engineering College, University of Shanghai for Science and Technology, Shanghai 200093)[1]

(Department of Information Science, Shanghai LIXIN University of Commerce, Shanghai 201620)[2]

Abstract A presentation model of the grid computing process based on grid service tree was discussed in this paper. A set of formal definitions, such as the grid service, the grid computing process, the grid service tree and the grid , was given . The formal presentation model about the grid computing process was provided. The implementing parallel algorithms of this model were studied. The analysis and experiment result show that this model effectively presents the grid computing process on WAN or Internet, and it is according with OGSA. This model can be fit for monitoring and building the grid system and managing QoS of Internet based on Web service.

Keywords Grid computing process, Grid, OGSA, Grid service tree

1 引言

随着信息技术的飞速发展和日益普及,海量信息的处理和高性能计算的需求越来越迫切,并且这些需求逐渐被国家建设的各个领域提出。寻求高性能价格比的海量信息处理技术、高性能计算技术已经成为产业界和学术界所面临的急需解决的主要问题。针对这一问题,网格[1]和数据网格[2]以其良好的自治性、自相似性、异构性、管理的多样性、强大的并行 I/O 能力、非常高的性能价格比等特性成为可行的最佳解决方案之一。获取这些强大计算、存储资源的必要基础是计算机网络,并且主要是分布广泛的 Internet 或 Intranet。然而,随着网络规模的扩大,网络资源的增加,系统的可控性、可靠性和资源的动态调动能力就会下降[3,4]。因此,基于 Internet 和 WAN、Intranet 的网络计算过程的监控问题成为研究的焦点[5~9]。文[5]给出面向并行和分布计算的可视化监测算法。文[6]构建了基于简单套接字的 C/S 模式的实时监控模型。文[7]对 TB 级规模的科学数据的 UDP 传输机制和过程进行了系统的研究。文[8,9]给出一个大规模数据并行性计算以及实时监测的过程模型。目前这些模型大多是基于传统的网络协议(UDP、TCP),并且往往和特定的应用系统密切相关,不适合通用的基于 Wan、Man 和 Internet 网格计算的应用。如今,结合 Web service[10] 技术的开放网格服务结构 OGSA[11,12] 已经成为目前学术界和工业界公认的标准。OGSA 是一个以服务为中心的模型,它通过虚拟的网格服务构建不同规模的服务资源,甚至是更广阔的虚拟组织。由于网格的广域性造成的网络延迟和异构性所带来的复杂性以及网络防火墙等安全机制等使传统的网络计算过程模型已经力不从心,新的网格计算过程的认识变得越来越重要。如何表示一个基于 OGSA 规范的网格计算过程对网格理论研究和网格应用建设都是一个重要的问题。粗粒度的并行非常适合广域网络上的网格计算。Web service 技术及其 SOAP[13] 协议为新的网格计算过程表示系统的实现奠定了基础。本文给出了一个基于 OGSA 规范的粗粒度网格计算过程表示模型,该模型可以跟踪粗粒度网格计算的过程,并把过程信息有机地组织起来,进而用 Web service 和 SOAP 技术模拟实现该模型。该模型可以适用于分析网格计算环境、网格的过程监测、Internet 的 QoS 分析等方面。

2 网格计算过程

按照 OGSA 规范,一个网格计算过程(本文所讨论的计算过程是基于粗粒度并行的)就是一个高级网格服务实现的过程。一个高级网格服务是由若干的其他网格所提供的服务实现的。

定义 1(网格服务树) 一棵网格服务树(Grid Service Tree)是一个三元组 $GST(GS, GS\_SET, R)$,其中 GS 为高级网格服务,$GS\_SET=\{GS_1, GS_2, \cdots, GS_n\}$ 一组低级服务,R 为规则集。GS 由一组低级服务 $GS_1, GS_2, \cdots, GS_n$ 按照规则集 R 实现。我们以 GS 为根结点,以 GS_1、GS_2、$\cdots$、GS_n 为子

*)基金项目:国家自然基金(60573108)、上海自然科学基金(04ZR14100)、上海局管科技发展基金重点基金(04JG05071)、上海教委发展基金(04EB21)。陈庆奎 教授,主要研究领域为网格、计算机机群、并行数据库;那丽春 副教授,主要研究领域为网格技术、数据挖掘。

孙节点，规则集 R 为连接关系构造网格服务树 GST。如果一个低级服务 GS_i 在树种的不同位置出现两次以上，则把它当作两次独立的服务，每次服务在 GST 中有一个树节点与之对应，第二次以后的服务采用虚拟节点表示。如图 1 所示，其中 GS_2 使用两次，第二次以虚节点表示。

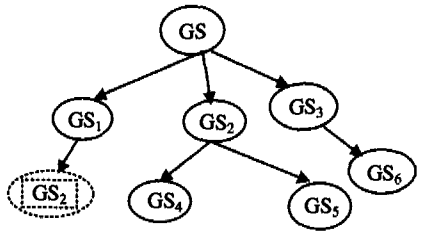

图 1　网格服务树

定义 2(网格计算过程)　一个网格计算过程 GCP(Grid Computing Process)就是按照先根顺序访问网格服务树 GST 所有节点的并发驱动、实现过程。全过程分三个过程：(1)服务驱动过程，父节点服务驱动子节点服务的过程，我们简记 DP(Driving Process)；(2)节点本地服务执行过程，本节点完成本地服务的逻辑功能，并且承担同步所有子服务工作，我们简记 CP(Computing Process)。(3)服务结果回收过程，子节点服务向父节点服务返回服务结果的过程，我们简记 RP (Receiving Process)。三个过程 DP、RP、CP 之间是异步并行进行，并且需要一定的同步机制。图 2 表示一个网格计算过程，图中虚有向线表示服务驱动过程，实有向线表示服务结果回收过程。

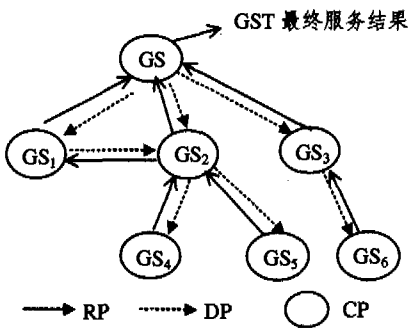

图 2　网格服务计算过程

定义 3(网格服务树节点)　一棵网格服务树 GST 的树节点(GST NODE)可以表示一个五元组 $GSTN$ ($GSTI$, GIP, DL, RL, CL)。其中 $GSTI$ 为网格服务树信息(GST Information)；GIP 为本服务节点所处的计算机的 IP 地址；DL 是本服务节点所驱动的其它子服务的列表(Drive List)，即该服务节点的所有子服务有序集合。CL 是本服务节点所完成的本地服务的进程列表(Computing List)，即该服务节点所有本地服务的有序集合。RL 为该服务节点接收其所有子服务的服务结果列表(Receive List)。

DL 可以表示为 DL(sid, dst)，其中 sid(Service ID)表示服务标志符，dst(Drive Start Time)表示服务 sid 的驱动起始时间；

CL 可以表示为 CL(cid, cst, cft, RSL)，其中 cid(Computing process ID)表示本地计算进程标志符，cst(Computing Start Time)表示 cid 计算的起始时间，cft(Computing Finished Time)表示 cid 计算的结束时间，RSL 是资源列表。

RL 可以表示为 RL(sid, rft, RSL)，其中 sid(Service ID)表示服务标志符，rft(Receive Finish Time)表示服务 sid 的服务结果回收结束时间，RSL 是资源列表。

RSL 的主要属性是：序号、资源类型、资源标志符、资源数量、使用时间；资源类型包括：CPU、Disk I/O、Communication、Memory、Disk。

3　网格计算过程表示模型(GCPPM)

3.1　GCPPM 体系结构

为了描述网格计算过程模型，首先给出模型涉及的主要部件。

(1)过程服务器 PS(Process Server)

过程服务器是负责管理、收集、协调、存储网格中的过程信息的计算机。在一个网格上，过程服务器可以采用集中式和分布式两种实现模式。本模型是基于多级分布机制的过程服务器模型。

设网格并行环境由 n 台计算机 $C_1, C_2, \cdots, C_n$ 通过 m 个物理网络 $N_1, N_2, \cdots, N_m$ 连接而成，每个物理网络中设置一台计算机为区域过程服务器 LPS_i(Local Process Server)($1 \leqslant i \leqslant m$)，整个网格上有一个全局过程服务器 GPS(Global Process Server)。

(2)过程信息库 PIB(Process Information Base)

PIB 是过程服务器存放 GST 的数据库。最初把 GST 的初始框架存放在 PIB 中，在 GST 被执行完毕后，PIB 存放的是带有网格计算全过程信息的 GST。为了适合网格环境的扩展性，我们对每个 LPS_i ($1 \leqslant i \leqslant m$)和 GPS 都配置一个过程信息库，分别记为 PIB_i ($1 \leqslant i \leqslant m$)、PIB。

(3)全局时钟 GTS(Global Time System)

为了有效支持网格计算过程的表示，需要在网格建立一个全局时钟体系 GTS。有关 GTS 我们可以借鉴时钟同步协议。

(4)网格服务节点 GSN(Grid Service Node)

在网格中完成特定服务的计算机或服务资源。

(5)过程服务代理 PSA(Process Service Agent)

PSA 负责收集各个网格服务节点的服务过程信息，同时完成和 LPS 的信息互达功能。

(6)过程信息管理系统 PIMS(Process Information Management System)

提供网格计算过程的过程信息的存储管理、语义表示、过程监控、可视化表示等功能。

定义 4(网格计算过程表示模型 GCPPM)　一个网格计算过程表示模型(grid computing process presentation model)可以表示为一个七元组 $GCPPM$(GPS, $LPSS$, PIB, $PIBS$, GTS, $PSAS$, $PIMS$)，其中 GPS 为全局过程服务器，$LPSS$ 是区域过程服务器集合$\{ LPS_i \mid (1 \leqslant i \leqslant m)\}$；$PIB$ 是全局过程信息库；$PIBS$ 是区域过程信息库集合$\{ PIB_i \mid (1 \leqslant i \leqslant m)\}$；$GTS$ 是全局时钟体系；$PSAS$ 是过程服务代理集合$\{ PSA_i \mid (1 \leqslant i \leqslant p)\}$；$PIMS$ 是过程管理系统。一个 $GCPPM$ 的体系结构图如图 3 所示。

定义 5(过程网格 PG)　一个过程网格(process grid)可以表示为一个二元组 PG(GS, $GCPPM$)，其中 GS 表示网格 PG 的所有网格服务节点的集合$\{GSN_1, GSN_2, \cdots, GSN_k\}$，$GCPPM$ 表示网格 PG 的网格计算过程环境。假设 GS 中的每个 GSN_i ($1 \leqslant i \leqslant k$)可以提供多个网格服务。

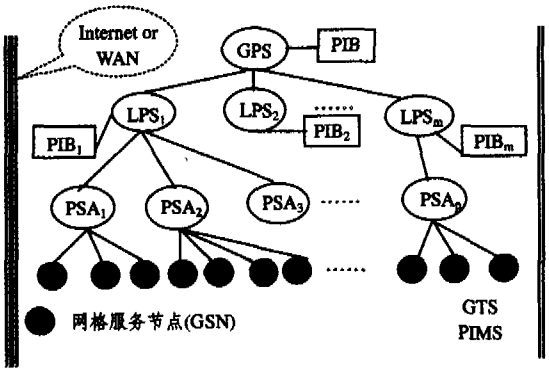

图 3　GCPPM 体系结构

3.2　GCPPM 并发执行过程

已知一个过程网格 *PG*(*GS*,*GCPPM*),一个 *GST* 在 *PG* 上被执行的过程是一个并发、异步驱动的过程,图 4 给出了 *PG* 的 *GCPPM* 的并发过程。

具体执行过程如下:

(1)应用代理 AA(applications agent)向网格 PG 提交网格服务树 *GST*,图 4 的〈1〉表示这一过程;

(2)GPS 并发地做(3)、(4);

(3)GPS 广播 *GST* 到所有的 LPS_i $(1\leqslant i\leqslant m)$,图 4 的〈2〉表示这一过程;

(4)GPS 通知网格 *PG* 驱动 *GST* 根服务,*GST* 被并发驱动执行,图 4 的〈4〉表示这一过程;

(5)所有的 LPS_i $(1\leqslant i\leqslant m)$ 并发广播 *GST* 到其所属 PSA,图 4 的〈3〉表示这一过程;

(6)所有 PSA 启动监视其管辖的网格服务节点(GSN)的进程;

(7)所有网格服务节点并行执行网格计算的三个过程:DP、CP、RP,并把每个过程的起始时间、资源使用、服务时间信息发送给其所对应的 PSA,图 4 的 PartC 和〈5〉表示这一过程;

(8)所有的 PSA_i $(1\leqslant i\leqslant p)$ 并发地收集其所辖网格服务节点的服务过程信息,并把收集到的过程信息发送给其所对应的 LPS,图 4 的〈6〉表示这一过程;

(9)所有的 LPS_i $(1\leqslant i\leqslant m)$ 并发地收集其所辖 PSA 提供的过程信息,并把收集到的过程信息发送给 GPS;把过程信息存放在本地数据库 PIB_i。图 4 的〈7〉表示这一过程;

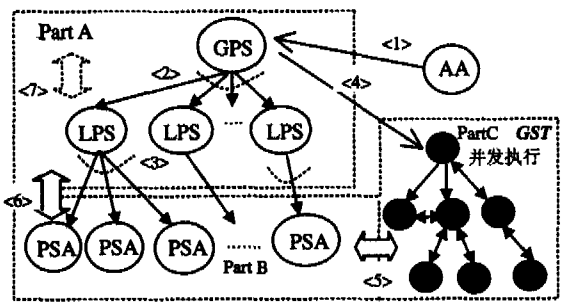

图 4　GCPVM 并发执行过程

(10)GPS 收集、监控所有 LPS 的过程信息并把过程信息存放到数据库 PIB;同时监测 *GST* 根节点的 RP 结果;如果 *GST* 根节点的 RP 信息收到,则把服务结果发送给 AA;结束 *GST* 计算过程。

(11)在整个 *GST* 被驱动执行过程中,*PIMS* 提取 *PIB* 信息并显示 *GST* 过程。

3.3　GCPPM 的主要算法

已知一个过程网格 *PG*(*GS*,*GCPPM*),GST_READY_QUEEN 是 *PG* 的网格服务树就绪队列;GST_STARTING_QUEEN 网格服务树启动队列;过程 Driving(*GSTN*)表示驱动一个以 GSTN 为根节点的网格服务树的计算过程;过程 Receiving(*GSTN*)表示异步接收以 *GSTN* 为根节点的网格服务树的服务结果。过程 Computing(*GSTN*)表示计算 *GSTN* 服务。finished(*GSTN*)表示判别以 *GSTN* 为根节点的网格服务树是否完成。Start(*p*)启动进程 *p*;*MSG* 为网格服务过程消息。

算法 1(GPS 算法)

在 GPS 上运行四个并发进程:网格服务接受进程、网格服务驱动进程、网格服务结果接收进程、接收服务过程信息进程。下面分别给出主要步骤。

a. 网格服务接受进程

(1)重复执行(2)~(4)步;

(2)接收应用代理 AA 提交的一个高级服务;

(3 根据服务信息构建网格服务树 *GST*;

(4)把新的 *GST* 加入网格服务树就绪队列 GST_READY_QUEEN;

b. 网格服务驱动进程

(1)重复执行(2)~(5)步;

(2)取 GST_READY_QUEEN 的一个网格服务树 *GST*;

(3)执行 Driving(*GST*) 和 Receiving(*GST*);

(4)把 *GST* 加入 GST_STARTING_QUEEN 对列;

(5)广播 *GST* 到所有的 LPS;

c. 网格服务结果接收进程

(1)对 GST_STARTING_QUEEN 的每个 *GST* 执行(2)步;

(2)如果 finished(*GST*)为真,则执行(3)~(5)步;

(3)发送 *GST* 的网格服务结果到 AA;

(4)从 GST_STARTING_QUEEN 中删除 *GST*;

(5)广播"Delete GST"命令到所有的 LPS;

d. 接收服务过程信息

(1)重复执行(2)~(5)步;

(2)从所有 LPS 接受 MSG;

(3)根据 MSG 确定 *GST*;

(4)存储 *GST* 到 PIB;

(5)通知 PIMS 刷新 *GST* 的可视化显示过程;

算法 2(LPS 算法)

LPS 算法在 *PG* 的所有 LPS 上运行。在每个 LPS_i $(1\leqslant i\leqslant m)$ 上都运行两个并发进程:传播网格服务树进程、接收服务过程信息进程。下面分别给出主要步骤。

a. 传播网格服务树进程

(1)重复执行(2)~(10)步;

(2)接收来自 GPS 的 *MSG*;

(3)If MSG 是"NEW GST" Then

(4)　存储 *GST* 到 PIB_i;

(5)　选播 *GST* 到 LPS_i 所属的 PSA;

(6)Endif;

(7)If MSG 是"Delete GST" Then

(8) 删除本地 *GST* 缓冲;
(9) 选播"Delete GST"到 LPS_i 所属的 PSA;
(10)Endif
b. 接收服务过程信息进程
(1)重复执行(2)~(5)步;
(2)从 LPS_i 的所有 PSA 接收 *MSG*;
(3)根据 MSG 确定 *GST*;
(4)存储 *GST* 到 PIB_i;
(5)发送 *MSG* 到 GPS;

算法 3(PSA 算法)

PSA 算法在 *PG* 的所有 PSA 上运行。
(1)重复执行(2)~(15)步;
(2)接收所有来自其对应的 LPS_i、GSN 的 *MSG*;
(3)If *MSG* 来自 GSN Then
(4) 根据 *MSG* 确定 *GST*;
(5) 刷新本地缓冲的 *GST* 信息;
(6) 发送 *MSG* 到该 PSA 的 LPS;
(7)Endif;
(8)If *MSG* 来自 LPS Then
(9) 根据 MSG 确定 *GST*;
(10) If *GST* 是一个新服务 then
(11) 本地缓冲 *GST*;
(12)Endif;
(11)If *MSG* 是"Delete GST" Then
(13) 删除本地 *GST* 缓冲;
(14) Endif
(15)Endif

算法 4(GSN 算法)

已知网格 *PG*(*GS*,*GCPPM*),假设一个 *GST*(*GS*,*GS_SET*,*R*)可以由网格 *PG* 提供的网格服务集实现。*PG* 中的 GSN_i($1 \leqslant i \leqslant k$)完成 *GST* 的 *GSTN* 所描述的网格服务,其过程由五个系统进程完成。这五个进程分别是:a. 远程子服务驱动进程 RSDP(Remote Service Driving Drocess);b. 本地子服务驱动进程 LSDP(Local Service Driving Drocess);c. 远程子服务监控进程 RSMP(Remote Service Monitor Process);d. 本地子服务监控进程 LSMP(Local Service Monitor Process);e. 同步监控进程 SMP(Synchronous Monitor Process)。

a. 远程子服务驱动进程 RSDP(Remote Service Driving Drocess)

输入:*GSTN* 的 *DL* 和 *RL* 列表;
(1)Count=0;/ * 成功驱动的远程子服务数目 * /
(2)$DLSet=\{Dl_1, Dl_2, \cdots, DL_n\}$;/ * DL_i 表示 *DL* 的第 *i* 个服务 * /
(3)While *DLSet*≠ϕ do/ * 驱动所有远程子服务 * /
(4) 从 *DLSet* 取一个服务 *VDL*(sid,dst);
(5) If 启动 Driving(*VDL*. *sid*)成功 Then
(6) 从 GTS 获取启动时间 StartTime;
(7) *VDL*. *DST*=StartTime;/ * 赋驱动时间 * /
(8) 启动 Receiving(*VDL*. *sid*);
(9) count ++;
(10) 申请一个回收列表节点 RL_{count};
(11) RL_{count}. *sid*= *VDL*. *sid*;
(12) RL_{count}. *RST*= *VDL*. *sid*;
(13) 根据 *VDL*, RL_{count} 更新 *GSTN*;
(14) *MSG*=(*GSTN*, *VDL*, RL_{count});/ * 构造消息 * /
(15) 发送 *MSG* 到其 PSA;
(16) *DLSet*=*DLSet*−{ *VDL* };
(17)Endif
(18)Endwhile;

b. 本地子服务驱动进程 LSDP(Local Service Driving Drocess)

输入:*GSTN* 的 *CL* 列表;
(1)$CLSet=\{Cl_1, Cl_2, \cdots, CL_n\}$;/ * CL_i 表示 *CL* 的第 *i* 个服务 * /
(2)While *CLSet*≠ ϕ do/ * 驱动所有本地子服务 * /
(3)从 *CLSet* 取一个服务 *VCL*(*cid*,*cst*, *cft*,*RSL*)
(4)If 启动 Computing(*VCL*. *cid*)成功 Then
(5) 从 GTS 获取启动时间 StartTime;
(6) *VCL*. *cst*=StartTime;/ * 赋驱动时间 * /
(7) 根据 *VCL* 更新 *GSTN*;
(8) *MSG*=(*GSTN*,*VCL*);/ * 构造消息 * /
(9) 发送 *MSG* 到其 PSA;
(10) *CLSet*=*CLSet*−{ *VCL* };
(11) Endif
(12)Endwhile;

c. 远程子服务监控进程 RSMP(Remote Service Monitor Process)

输入:*GSTN* 的 *RL* 列表;
(1)$RLSet=\{Rl_1, Rl_2, \cdots, RL_n\}$;/ * RL_i 表示 *RL* 的第 *i* 个服务 * /
(2)While *RLSet*≠ ϕ do/ * 监控所有本地子服务 * /
(3)从 *RLSet* 取一个服务 *VRL*(*sid*,*rft*,*RSL*);
(4)If 进程 Receiving(*VRL*. *sid*)结束 Then
(5) 从 GTS 获取结束时间 FinishedTime;
(6) *VCL*. *rft*FinishedTime;/ * 赋结束时间 * /
(7) 把 *VRL*. *sid* 花费的远程资源存储到 *VRL*. *RSL*;
(8) 根据 *VRL* 更新 *GSTN*;
(9) *MSG*=(*GSTN*,*VRL*);/ * 构造消息 * /
(10) 发送 *MSG* 到其 PSA;
(11) 发送 *VRL*. *sid* 的"服务结果"到进程 SMP;
(12) *RLSet*=*RLSet*−{*VRL*};
(13)Endif
(14)Endwhile;

d. 本地子服务监控进程 LSMP(Local service Monitor process)

输入:*GSTN* 的 *CL* 列表;
(1)$CLSet=\{CL_1, CL_2, \cdots, CL_n\}$;/ * CL_i 表示 *CL* 的第 *i* 个服务 * /
(2)While *CLSet*≠ϕ do/ * 监控所有本地子服务 * /
(3)从 *CLSet* 取一个服务 *VCL*(*cid*,*cst*,*cft*,*RSL*);
(4)If 进程 Computing(*VCL*. *cid*)结束 Then
(5) 从 GTS 获取结束时间 FinishedTime;
(6) *VCL*. *cft*=FinishedTime;/ * 赋结束时间 * /
(7) 计算 *VCL*. *cid* 所需本地资源并记入 *VCL*. *RSL*;
(8) 根据 *VCL* 更新 *GSTN*;
(9) *MSG*=(*GSTN*,*VCL*);/ * 构造消息 * /
(10) 发送 *MSG* 到其 PSA;

(11) 发送 $VCL.sid$ 的"服务结果"到进程 SMP;
(12) $CLSet=CLSet-\{VCL\}$;
(13)Endif
(14)Endwhile;
e. 同步监控进程 SMP(Synchronous Monitor process)
输入:$GSTN$ 的 CL 列表;
(1)$CLSet=\{CL_1,CL_2,\cdots,CL_n\}$;/* CL_i 表示 CL 的第 i 个服务 */
(2)$RLSet=\{Rl_1,Rl_2,\cdots,RL_n\}$;/* RL_i 表示 RL 的第 i 个服务 */
(3)While $CLSet\neq\phi$ and $RLSet\neq\phi$ do /* 监控所有本地、远程子服务 */
(4) MSG=ReceiveMassage(LSMP, RSMP);/* 接收 LSMP、RSMP 进程发来的消息 */
(5) 把 MSG 输入同步区,并通知等待该 MSG 的本地子服务;
(6) If MSG 来自 LSMP Then
(7) 根据 MSG 确定 CL_i;
(8) $CLSet=CLSet-\{CL_i\}$;
(9) Endif
(10) If MSG 来自 RSMP Then
(11) 根据 MSG 确定 RL_i;
(12) $RLSet=RLSet-\{RL_i\}$;
(13) Endif
(14)Endwhile
(15)得到 $GSTN$ 最终服务结果 $GSTNResult$;
(16)$MSG=(GSTN,GSTNResult)$;/* 构造消息 */
(17)If $GSTN$ 是 GST 的根节点 Then
(18) 发送 MSG 到 GPS;
(19)Else
(20) 发送 MSG 到其 $GSTN$ 的父网格服务节点;
(21)Endif
(22)发送"GSTN 执行结束"到其 PSA;
(23)GSTN 执行结束;

分析:根据 GCPPM 的实现机制可以看出,一个网格服务 GST 在 GCPPM 中被执行结束后,GST 中的信息完全记载了该网格计算的过程信息。

4 实现

4.1 模拟网格

构建了由 10 台 PC 机和三个物理网络 A、B、C 构成的广域网络来模拟一个实验网格。A 物理网络由 4 台 PC 构成,其中一台作 GPS,B、C 物理网由三台 PC 构成。A 物理网络由固定 IP 地址的局域网络联入 Internet,B、C 网络由 FTTB 拨号联入 Internet。A、B、C 分别处于城市的三个不同的行政区,即 A、B、C 由 WAN 连接构成。WAN 的通信数度在 128k/s~512k/s 范围内变化。

每个物理网络上选一台 PC 当作 LPS,所有 PC 都运行一个 PSA。每台 PC 都是网格服务节点(GSN)。

4.2 网格服务示例

设一共有 n 个基本网格服务 $\{GS_1,GS_2,\cdots,GS_k\}$,对每个服务 $GS_i(1\leqslant i\leqslant k)$ 执行算法如下:

```
Int GS(int i)
{   Wait_Random(i)/*表示网络延迟*/
  Sleep(i);/*等待 i 个时间段(秒),代表本地计算时间,*/
  Return i;/*返回的服务结果*/
}
```

忽略了服务使用的其它资源,仅仅探讨服务过程的时间因素。

模拟由 $\{GS_1,GS_2,\cdots,GS_k\}$ 运用加(+)、减(-)、乘(*)、除(/)运算构成的复合算数运算服务。例:一个高级网格服务是:$GS=GS_1+(GS_2+GS_3*(GS_6/GS_8)*7)-GS_7$。显然可以用 GST 表示 GS。当服务的个数 K 大于网格服务节点的个数时,采用轮转法在网络 A、B、C 的计算机上分配服务。实验分别对 k=3、7、15、31、63、127 时进行。实验是针对一个均衡蛇型 GST 进行的,均衡蛇形 GST 构造如下:首先构造一个 $K(k=3、7、15、31、63、127)$个节点的均衡二叉树,然后把 k 个 $GS_1,GS_2,\cdots,GS_k$ 按先根顺序蛇型分布在二叉树上。

4.3 实现技术

Web service 技术已经成为开发下一代互联网络的主要技术之一,其基于服务的思想恰好与 OGSA 规范相吻合。利用 Microsoft . net 的 C# 和 Web service 框架实现了本文提供的模型。GCPPM 中的 GPS、LPS、PSA、AA 以及 PG 的所有服务都用 Web 服务模式开发,所有部件间通信是利用 SOAP 协议进行的。

4.4 全局时钟 GTS

应用一个免费软件 WebTime 模拟了 GTS,在模拟网格的每个计算机上安装 WebTime 完成时钟同步软件。

4.5 实验

实验的目的是检测该模型的运行过程。实验针对 6 个服务(k=3、7、15、31、63、127)在 3 种不同情况下完成的过程进行了监测,这 3 种情况是:(1)采用 UDDI 注册中心;(2)不采用 UDDI 注册中心,网格中的所有 GSN 在初始化阶段,到 GPS 获取所有提供服务 $GS_1,GS_2,\cdots,GS_k$ 的计算机的 IP 地址表;(3)应用随机函数模拟网络延迟。每个服务分别执行 10 次,然后取平均值作为最终结果。图 5 显示了通过 *GCPPM* 获取的网格服务最终平均完成时间的情况,横轴表示 6 个服务,纵轴表示 3 种情况下服务完成的时间。

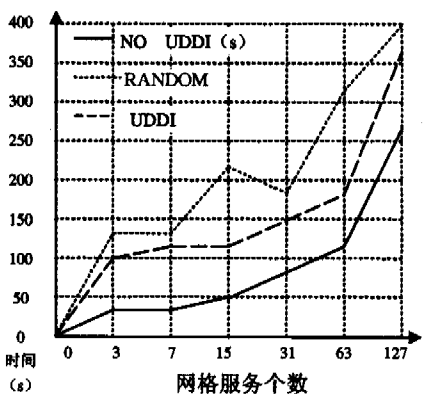

图 5 对比实验结果

结束语 基于 Internet 或 WAN 的网格计算系统,有效的计算过程表示模型非常重要。该模型采用多级分布的过程服务器、分布的 PSA 机制以及 Web 时钟体系,使该模型更适合 WAN 和 Internet,具有良好的可扩展性。由于篇幅限制,本文忽略了过程服务容错问题,相关的研究结果将陆续发表。本文所讨论的网格计算过程表示模型适合基于 WAN 或 Internet 的网格系统的监测、过程分析和服务性能分析等方面,

还可以用于互联网络上的 Web service 的 QoS 分析监测等方面。

参 考 文 献

1 Foster I, Kesselman C. The Grid: Blueprint for Future Computing Infrastructure [M]. San Francisco, USA: Morgan Kaufmann Publishers, 1999

2 Segal B. Grid Computing: The European Data Project [A]. In: IEEE Nuclear Science Symposium and Medical Imaging Conference [C], Lyon, 2000. 15～20

3 Siweiorek D P, Swarz R S. The Theory and Practice of Reliable System Design. Digital Press, Digital Equipment Coporation, ISBN 0-932376-13-4, 1983

4 leon J. Fail-safe PVM. PVM Users Griup Meeting, Oak Ridge, TN, May 1994

5 Wittenbrink C M. Survey of Parallel Volume rendering Algorithms. In: Proc. of Int. Conf. On Parallel Distributed Processing Techniques and Applications [C]. 1998. 1329～1336

6 Mahovsky J, Benedicenti L. An Architecture for Java-Based Real-time Distributed Visualization. IEEE Transactions on Visualization and Computer Graphics [J]. 2003, 9(4): 570～579

7 Bethel E W, Shalf J. Cactus and Visapult: An Ultra-High performance Grid-Distributed Visualization Architecture Using Connectionless Protocols. IEEE Computer Graphics and Applications [J]. 2003, 23(2): 51～59

8 Li K, Malony A, Bell R, et al. A Framework for Online Performance Analysis and Visualization of Large-Scale Parallel Applications. International Conference on Parallel Proceeding and Applied Mathematics [C], Czestochowa, Poland, September 2003

9 Brunst H, malony A, Shende S, et al. Online Remote Trace Analysis of Parallel Applications on High-performance Cluster. International Symposium on High-performance Computing [C], October 2003

10 OGSA. http://www.gridforum.org/ogsi-wg/drafts/GS_Spec_draft03_2002-07-17.pdf

11 Foster I, Kesselman C, Nick J, et al. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. January, 2002. http://www.globus.org/research/papers/ogsa.pdf

12 Web Service. http://www.w3.org/2003/ws

13 Soap. phttp://www.w3.org/TR/Soap

(上接第 61 页)

的可能性也稍稍提高。从图中还可以看出延迟-立即方式和立即-等待方式的更新度比较接近，而立即-立即方式是三种中更新度最高的。

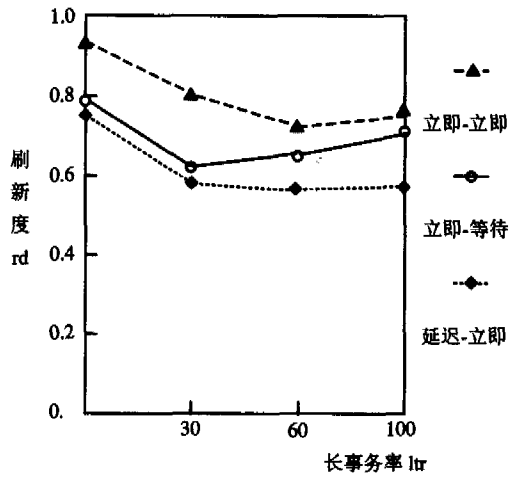

图 6 高更新事务到达率 f-1 图

我们接着考察主节点在高的更新事务到达率的情况下的刷新度，如图 6 所示。从图中可以看出，当 $ltr=0$ 时，三种策略的刷新度都比低更新事务到达率的情况下的刷新度低，这是因为，对于短的频繁到达的短事务而言，当更新事务 T_i 在主节点的主版本 R 上提交后，刷新事务 Ti 被成功提交给辅助节点副版本 r 前，T_{i+1}、T_{i+2}、T_{i+3}…已经提交给 R 的可能性极高。在高事务到达率的情况下，网络相对比较拥塞，刷新度受网络的影响比较大，延迟传播方式比立即传播方式所需的传播时间更长，因此当长事务率较高后，立即-等待比延迟-立即方式的更新度要高，而且差距加大，这同在低到达率情况下的状态差别比较大。

总结 本文提出了一种“温和一致性代理复制机制 MCARM)”，该机制采用了主节点的复制管理器(RM)与辅助节点的 MSS-Agent 协调工作的架构，吸取严格一致性协议和弱一致性协议的优势，又避开其局限性和复杂性，更好地适应移动计算环境的要求，并能与缓存失效策略 CISBMA[6] 协同工作，保证了移动用户的应用需求，具有较强的实际应用价值。

参 考 文 献

1 Agrawal D, Bernstein A J. A nonblocking quorum consensus protocol for replicated data. IEEE Trans. Parallel Distrib. Syst. April, 1991. 171～179

2 Bernstein P A, Goodman N. Concurrency control and recovery in database systems. ACM Comput. Surv, 1981, 13(2): 185～221

3 Gifford D. Weighted voting for replicated data. In: Proc. 7th ACM-SIGOPS Symposium on OS Principles, Pacific Grove, CA, Dec, 1979. 150～159

4 Gray J, Helland P. The dangers of replication and a solution. Proc. ACM SIGMOD Int. Conf. on Magnagement of Data, Montreal, Canada. ACM Press, New York, June 1996, 25(2): 173～182

5 吴劲，卢显良，任立勇，等. 移动计算环境中基于移动代理的数据管理体系结构. 计算机科学，2005，32(5)：76～78

6 吴劲，卢显良，任立勇. 在移动计算环境中基于移动代理的缓存失效方案. 计算机科学，2003，30(4)：82～84

7 Ceri S, Owicki S. On the use of optimistic methods for concurrency control in distributed databases. In: Proceedings 6th Berkeley Workshop on Distributed Data Management and Computer Networks, Berkeley, Calif, 1982. 117～130

8 Pacitti E, Simon E. Update propagation strategies to improve freshnes in lazy master replicated databases. the VLDB Journal, 2000. 305～318

9 Hadzilacos V, Toueg S. A modular approach to fault-tolerant broadcasts and related problems: [Technical Report TR-94-1425]. Dept. of Computer Science, Cornell University, Ithaca, N. Y., 1994

10 Phatak S H, Badrinath B R. Multiversion reconciliation for mobile database. In: Proc the 15th International Conference on Data Engineering, Sydney, Australia, 1999. 582～589

11 Didriksen T, Galindo-Legaria C A, Dahle E. Database de-centralization: A practical approach. In: Proceedings of the 21st VLDB conference, Sep 1995. 654～665

12 Son S H, Zhang F J. Real-time replication control for distributed database systems: Algorithms and their performance. In: Proceedings of the 4th International Conference on Database Systems for Advanced Applications, Apr 1995. 214～221