ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.17, No.6, June 2006, pp.1436−1447
DOI: 10.1360/jos171436

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

网格数据复本管理的动态自适应软件体系结构*

陈 磊+, 李三立

(清华大学 计算机科学与技术系,北京 100084)

Dynamic Self-Adapting Software Architecture for Replica Management in Grids

CHEN Lei+, LI San-Li

(Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)

+ Corresponding author: Phn: +86-10-62775830, E-mail: c-L03@mails.tsinghua.edu.cn, http://www.tsinghua.edu.cn

Chen L, Li SL. Dynamic self-adapting software architecture for replica management in grids. *Journal of Software*, 2006,17(6):1436−1447. http://www.jos.org.cn/1000-9825/17/1436.htm

Abstract: Single data replica management strategy can not meet the complexity and diverseness in data grid. This paper proposes a dynamic self-adapting replica management architecture (DSA-RM) driven by software architecture (SA). In DSA-RM, architecture descriptions are shared by the whole grid environment as a system kernel to schedule various replica management components that are adapted to various application and VO in data grid. By the abstraction of DSA-RM, the paper describes the formal definitions of replica management components and discusses the dynamic self-adapting regulation of component evolution and multiplication in dynamic environment. By these regulations, the DSA-RM's framework and algorithms of multiplication are designed. Experimental results show that the DSA-RM can effectively promote the performance of grid system. Finally, the project of applying the DSA-RM on Shanghai medical data grid is introduced.
Key words: software architecture; data grid; self-adapting; replica; DSA-RM

摘 要: 单一策略的复本管理不能适应网格系统复杂、多变的环境.提出了软件体系结构驱动的动态自适应数据复本管理架构 DSA-RM.在该架构中,软件体系结构描述作为核心元素被整个数据网格运行环境共享,适用于不同应用环境和虚拟组织的数据复本管理策略则可以按照描述规则封装为构件.DSA-RM 以该描述作为系统调度依据,运行各数据复本管理构件.通过对 DSA-RM 的抽象,形式化地描述了框架下的复本管理构件,讨论了动态环境下构件演化、复制过程的动态自适应规则,并按这些定义和规则设计了 DSA-RM 的实现框架和构件复制算法.通过性能分析以及对框架下多策略执行模拟验证表明,DSA-RM 可以有效提升网格系统的整体性能.最后,介绍了下一步在上海医学数据网格上的应用计划.
关键词: 软件体系结构;数据网格;自适应;复本;DSA-RM
中图法分类号: TP301 文献标识码: A

数据网格[1−4]的研究目的是解决如何在数据密集型计算应用中方便、高效地使用分布式异构数据资源的问题.为了提高系统的性能、可靠性及可用性,数据网格系统通常会利用复制技术在多个网格结点上建立同一

* Supported by the "SEC E-Institute: Shanghai High Institutions Grid" Project (上海高校网格 e-研究院资助)
Received 2006-01-09; Accepted 2006-03-13

数据的多个副本[5].数据副本管理主要负责副本的创建、迁移以及副本的一致性,并且可以考虑采用不同的体系结构,以适应不同应用环境的数据复本管理需求[6].例如,构建在相同网格中间件 Globus[11]基础上的 Grid Physics Network Project(GriPhyN)[7],Earth System Grid[8]以及北卡罗来那生物信息网格计划(NC BioGrid)[9]、欧洲的 BioGrid[10]等数据网格,均结合自身应用特点,采取了不同的数据复本管理策略.同样,为适应不同应用中数据复本管理策略的变化,基于应用特征的数据复本管理策略也成为数据网格中间件的研究热点.例如:Giggle[12]对于数据网格中大量只读数据的复本管理具有良好的性能;RLS[13]考虑数据一致性、缓存空间的限制、可靠性、数据更新效率以及查询效率等因素给出了数据复本管理的折衷方案;P-RLS[14]则利用 Peer-to-Peer[15]技术提高了 RLS 的效率;BHR[16]可根据各网格节点的接入带宽自动调整复本所在位置.

从软件体系结构[19]角度来看,数据复本管理策略的实现可视为数据复本管理平台中的可复用构件,可考虑采用构件的制作和集成技术[20]实现重用.此外,在实际应用中,同一网格节点可能位于多个对副本创建、迁移以及副本一致性要求不同的虚拟组织(VO)[17,18]中,且在运行过程中也可能由于网格节点增加以及数据属性变化(如由只读变为多节点可写)需要新的复本分布和一致性维护策略以提高系统性能.因此,数据复本管理平台的架构还具有动态性,主要表现在:(1) 自身结构的动态性.复本管理所扮演的角色由它在数据网格中所起的作用,即向外界提供的服务而定,例如,是否需要管理多写数据,是否需要提供数据一致性服务,是否需要向其他节点提供数据复本等.当向外界提供的服务即复本管理的能力或功能需要调整时,其自身结构必须发生变化以降低系统开销.(2) 交互结构的动态性.针对 VO 的不同特征、同一网格节点在不同 VO 中的不同角色以及数据网格运行环境的变化,网格节点上数据复本管理构件的通信对象及相互间的关系可能发生变化.因此,需要设计一种动态自适应体系结构[21−23],以满足在连续运营情况下的数据复本管理策略的变化.

针对上述需求,我们基于现有网格中间件平台设计了一种数据网格复本管理的动态自适应体系结构(dynamic self-adapting replica management architecture,简称 DSA-RM).在该架构中,数据复本管理中间件的软件体系结构描述作为核心元素被整个数据网格运行环境所共享,适用于不同应用环境和虚拟组织的各种数据复本管理策略则按照描述规则封装为构件.DSA-RM 以该描述作为系统调度依据,运行各数据复本管理构件的调度、执行和通信.

本文第 1 节用抽象代数理论,参考 Garlan & Shaw 软件体系结构模型[24]和软件体系结构抽象模型[25],形式化地描述 DSA-RM 体系结构,并在相关命题证明基础上讨论软件体系结构驱动的数据复本管理动态自适应规则.第 2 节介绍 DSA-RM 的实现框架和关键技术.第 3 节采用模拟方法验证多策略对整体性能的提升.最后总结全文并介绍 DSA-RM 在上海医学网格上的实现计划.

1 DSA-RM 体系结构

1.1 构架和连接件

DSA-RM 的构件定义参考了 Garlan & Shaw 软件体系结构模型[24]和软件体系结构抽象模型[25].定义构件 *Comp*=(*Ports*,*ImpBs*),其中:*Ports* 是构件接口集合;*ImpBs* 是构件的实现.*port*=(*ID*,*Publ*,*Extn*,*Prvt*,*Beha*,*Msg*,*Cons*,*NFun*),其中:*ID* 为构件标识;*Publ* 是 *port* 向外提供的功能的集合;*Extn* 是外部通过 *port* 向构件提供的功能的集合;*Prvt* 是 *port* 的私有属性集合;*Beha* 是 *port* 的行为语义描述;*Msg* 是 *port* 产生的消息集合;*Cons* 是 *port* 的行为约束;*NFun* 是 *port* 的非功能说明.*Cons* 可以表示为一个三元组(*init*,*pre-cond*,*post-cond*),分别表示组件的初始条件、前置条件和后置条件.定义连接件 *Con*=(*ID*,*Beha*,*Msg*,*NFun*,*Cons*,*Role*),其中:*ID* 为连接件的标识;*Beha* 是连接件行为语义的描述;*Msg* 是构件与各 *Role* 交互事件产生的消息的集合;*NFun* 为连接件的非功能描述;*Cons* 是连接件约束的集合;*Role* 是连接件与构件交互点的集合.

基于上述通用规则,我们对 DSA-RM 的构件及连接件定义如下:

定义 1. 设数据复本管理策略 P 的功能全集为 $FSall(P)$,$\forall f\in FSall(P)$,对构件 C,若$(\exists PortS\subseteq C.Ports)\wedge(f\in\cup PortS.Publ)$,则称 C 是 P 的实现构件,记作 $C=Comp(P)$.

根据定义 1,DSA-RM 的构件集合 $Comp=\{CompS,CompC\}$.$CompS$ 是数据复本管理策略的实现构件集

合,$CompC$ 表示用于管理和驱动 $CompS$ 的控制管理构件集合.

定义 2. 连接件集合 $Conn=\{ConnS,ConnM,ConnL\}$.$\forall conni\in Conn$,(1) $\exists S\in CompS\wedge(conni.Role\subseteq S.Ports)\Leftrightarrow conni\in ConnS$;(2) $\exists S_s\in CompS\wedge\exists S_c\in CompC\wedge(\exists R_i,R_j\in conni.Role\wedge R_i\in S_s.Ports\wedge R_j\in S_c.Ports)\Leftrightarrow conni\in ConnL$; (3) $\exists S_i,S_j\in CompC\wedge(conni.Role\subseteq S_i.Ports\cup R_j\in S_j.Ports)\Leftrightarrow conni\in ConnM$.

定义 3. $\forall compS_i\in CompS,compS_i.PortS=\{PortM,PortR\}$.$\forall porti\in compS_i.PortS$,(1) $porti\in ConnL.Role\Leftrightarrow porti\in PortM$;(2) $porti\in ConnS.Role\Leftrightarrow porti\in PortR$.

定义 4. $\forall S_i,S_j\in CompS,(Publ(S_i)\supseteq Publ(S_j))\wedge(Extn(S_i)\subseteq Extn(S_j))\wedge(Prvt(S_i)\subseteq Prvt(S_j))\wedge(Msg(S_i)=Msg(S_j))\wedge((Cons(S_i)\Rightarrow Cons(S_j))\Rightarrow(Beha(S_i)\Rightarrow Beha(S_j)))$,则称 S_j 是 S_i 的一个演化,记作 $S_i\rightarrow S_j$.

命题 1. $\forall S_i,S_j,S_k\in CompS,((S_i\rightarrow S_j)\wedge(S_j\rightarrow S_k))\Rightarrow(S_i\rightarrow S_k)$.

证明:

$$(S_i\rightarrow S_j)\Rightarrow(Publ(S_i)\supseteq Publ(S_j))\wedge(Extn(S_i)\subseteq Extn(S_j))\wedge(Prvt(S_i)\subseteq Prvt(S_j))\wedge(Msg(S_i)=Msg(S_j)) \tag{1}$$

$$(S_j\rightarrow S_k)\Rightarrow(Publ(S_j)\supseteq Publ(S_k))\wedge(Extn(S_j)\subseteq Extn(S_k))\wedge(Prvt(S_j)\subseteq Prvt(S_k))\wedge(Msg(S_j)=Msg(S_k)) \tag{2}$$

由式(1)、式(2)得

$$(Publ(S_i)\supseteq Publ(S_k))\wedge(Extn(S_i)\subseteq Extn(S_k))\wedge(Prvt(S_i)\subseteq Prvt(S_k))\wedge(Msg(S_i)=Msg(S_k)) \tag{3}$$

$$(S_i\rightarrow S_j)\Rightarrow(Cons(S_i)\Rightarrow Cons(S_j))\Rightarrow(Beha(S_i)\Rightarrow Beha(S_j)) \tag{4}$$

$$(S_j\rightarrow S_k)\Rightarrow(Cons(S_j)\Rightarrow Cons(S_k))\Rightarrow(Beha(S_j)\Rightarrow Beha(S_k)) \tag{5}$$

若 $(Cons(S_i)\Rightarrow Cons(S_j))\wedge(Cons(S_j)\Rightarrow Cons(S_k))=\text{true}$, 则 $(Cons(S_i)\Rightarrow Cons(S_k))=\text{true}$. 此时,由式(4)、式(5)得 $(Beha(S_i)\Rightarrow Beha(S_j))\wedge(Beha(S_j)\Rightarrow Beha(S_k))=\text{true}$,则$(Beha(S_i)\Rightarrow Beha(S_k))=\text{true}$.由此可得

$$(Cons(S_i)\Rightarrow Cons(S_k))\Rightarrow(Beha(S_i)\Rightarrow Beha(S_k)) \tag{6}$$

由式(3)、式(6)可得 $S_i\rightarrow S_k$,命题得证. □

定义 5. 设 $S\in CompS,C\in CompC$,如果$\exists x\in Publ(C)\wedge\exists y\in Extn(S)$使得$(x\Rightarrow y)\wedge(pre\text{-}cond(S))$,构件 C 通过 $ConnM$ 集合中的连接件向构件 S 发送消息激发(invocation)[26]构件 S 的 $Publ(S)$实现功能需求.称此操作为激发操作,记作 $Inv(C,S)$.若 $Inv(C,S_0)\wedge Inv(C,S_1)\wedge\ldots\wedge Inv(C,S_n)$,则记作 $Inv(C,\{S_0,S_1,\ldots,S_n\})$.

$Inv(C,S)$也是一个构件[26],满足如下性质:

$Publ(Inv(C,S))=Publ(C)\cup Publ(S)$;

$Extn(Inv(C,S))=Extn(C)\cup Extn(S)$;

$Priv(Inv(C,S))=Priv(C)\cup Priv(S)$;

$Beha(Inv(C,S))\Leftrightarrow Beha(C)\wedge Beha(S)$;

$Msg(Inv(C,S))=Msg(C)\cup Msg(S)$;

$Cons(Inv(C,S))\Leftrightarrow Cons(C)\wedge Cons(S)$;

$NFun(Inv(C,S))\Leftrightarrow NFun(C)\wedge NFun(S)$.

$Inv(C,\{S,T\})$也是一个构件,满足如下性质:

$Publ(Inv(C,\{S,T\}))=Publ(C)\cup Publ(S)\cup Publ(T)$;

$Extn(Inv(C,\{S,T\}))=Extn(C)\cup Extn(S)\cup Extn(T)$;

$Priv(Inv(C,\{S,T\}))=Priv(C)\cup Priv(S)\cup Priv(T)$;

$Beha(Inv(C,\{S,T\}))\Leftrightarrow(Beha(C)\wedge Beha(S))\vee(Beha(C)\wedge Beha(T))$;

$Msg(Inv(C,\{S,T\}))=Msg(C)\cup Msg(S)\cup Msg(T)$;

$Cons(Inv(C,\{S,T\}))\Leftrightarrow(Cons(C)\wedge Cons(S))\vee(Cons(C)\wedge Cons(T))$;

$NFun(Inv(C,\{S,T\}))\Leftrightarrow(NFun(C)\wedge NFun(S))\vee(NFun(C)\wedge NFun(S))$.

命题 2. $Inv(C,S)\in CompS$.

证明:设 $S=Comp(F)$,按照定义 1 有$\forall f\in F,(\exists PortS\subseteq C.Ports)\wedge(f\in\cup PortS.Publ)$.按照定义 5 中 $Inv(C,S)$的性质,$\forall PortS,PortS.Publ\subseteq Publ(C)$,可知$(\exists PortI\subseteq Inv(C,S).Ports)\wedge(f\in\cup PortI.Publ)$.命题得证. □

命题 **3**. $Inv(C,\{S_0,S_1,\ldots,S_n\})\in CompS$.

证明:用数学归纳法.当 $n=0$ 时,命题成立.当 $n=1$ 时,按照定义 5 之 $Inv(C,\{S,T\})$的性质可知命题成立.设 $n=K$ 时命题成立,即 $Inv(C,\{S_0,S_1,\ldots,S_k\})\in CompS$,则当 $n=K+1$ 时,显然有 $Inv(C,S_{k+1})\in CompS$,可得 $Inv(C,\{S_0,S_1,\ldots,S_k\})\wedge Inv(C,S_{k+1})\in CompS$.命题得证. □

命题 **4**. 构件集合 $S=\{S_0,S_1,\ldots,S_n\}$,若$(S\subseteq CompS)\wedge(\forall S_i,S_j\in S,(S_i\rightarrow S_j)\wedge(S_j\rightarrow S_i))$成立,可构造 $P\in CompS$ 满足 $\forall S_j\in S,P\rightarrow S_j$.此时称 S 是 P 的演化状态集合.

证明:构件 $P=Inv(C,\{S_0,S_1,\ldots,S_n\})$,按照命题 2,显然 $P\in CompS$.

$$\begin{aligned}&(S\subseteq CompS)\wedge(\forall S_i,S_j\in S,(S_i\rightarrow S_j)\wedge(S_j\rightarrow S_i))\Rightarrow(Publ(S_i)=Publ(S_j))\wedge\\&(Extn(S_i)=Extn(S_j))\wedge(Prvt(S_i)=Prvt(S_j))\wedge(Msg(S_i)=Msg(S_j))\Rightarrow(Publ(P)=Publ(S_j))\wedge\\&(Extn(P)=Extn(S_j))\wedge(Prvt(P)=Prvt(S_j))(Msg(P)=Msg(S_j))\end{aligned}\tag{7}$$

按照定义 5 的性质有

$$(Cons(P)\Rightarrow Cons(S_j))\Rightarrow(Beha(P)\Rightarrow Beha(S_j))\tag{8}$$

成立.由式(7)、式(8)可得 $P\rightarrow S_j$.命题得证. □

1.2 动态自适应规则

DSA-RM 可看作一个数据复本管理构件 CompS 的管理模型,它包括所建立系统中各 CompS 元素在分布式网格系统各节点上生命周期内的状态描述、交互、演化、复制和共生的规则集合.DSA-RM 需要支持的体系结构变化包括两个方面:(1) 网格节点间数据复本管理关系的变化.例如,由于运行环境的变化导致 VO 中节点间从层次关系转变为 P2P 关系.(2) 网格节点内数据复本管理体系结构的变化.例如,新数据复本管理策略 S 在网格节点 M 上的部署、从网格节点 M 上删除一个数据复本管理策略 S.为便于描述,分别定义为 R-Chng 和 A-Chng.驱动体系结构变化的条件可分为两类:(1) 自治系统驱动,即 DSA-RM 监控到系统运行环境的变化,如网络可用带宽、网格节点数量、存储资源数量等,由此驱动 DSA-RM 调整数据复本管理策略以得到最优性能;(2) 应用驱动:网格节点用户调整 DSA-RM 运行环境,例如增加了新的应用、建立了新的 VO 等,这种驱动由网格用户直接发起,在全部节点或 VO 的部分节点上部署新的数据复本管理策略.为便于描述,分别定义为 D-Mnt 和 D-Appl.按上述定义,R-Chng⇒D-Mnt,A-Chng⇒D-Appl.

R-Chng 将引发可逆的数据复本管理构件演化,由第 1.1 节定义 4 和命题 1、命题 4 可知,R-Chng 演化可构造为同一数据复本管理构件的不同约束和行为.A-Chng 将引发不可逆数据复本管理构件的演化,由第 1.1 节定义 5 和命题 2、命题 3 可知,数据复本管理构件需要提供用于调度和激发的统一接口.

定义 **6**. 任一网格节点上的 CompC 全集是一个自适应系统,可描述为一个四元组 CompC=(Heart,Sensor,Evolve,Multiply).其中:Heart 为系统内核模块,负责存放传感器得到的最新信息与 CompS 构件进化的约束和行为,可以表示为三元组(CurInfo,Logic,Cond),CurInfo 表示一个 CompS 构件目前的状态信息,比如 CPU,Memory,Performance 等;Logic 表示构件将会产生的演化结果;Cond 表示构件产生演化必须满足的条件;Sensor 为负责系统内部监控的传感器集合,用于收集各构件运行期间的信息,将其反馈给 Heart 模块;Evolve 为演化逻辑功能模块集合,根据 Sensor 所提供的信息按照 CompS 指定的方式演化,以适应变化的环境;Multiply 为复制繁衍模块集合,能够在网格节点间完成 CompS 构件的复制.

由定义 6 可知,DSA-RM 满足如下性质:(1) 每个网格节点必定存在也仅存在一个 Heart;(2) 每个 CompS 构件有且只有一个 Sensor 跟踪其状态;(3) 每个 CompS 构件有且只有一个 Evolve 构件跟踪其演化状态;(4) 每个 CompS 有且只有一个 Multiply 用于其节点间的复制.

1.3 体系结构模型

结合第 1.1 节的形式化描述和命题及第 1.2 节动态自适应规则的定义和描述,DSA-RM 可表示如下:(1) DSA-RM 必须由一个内核、传感逻辑集合、演化逻辑集合和复制逻辑集合组成,其中演化逻辑和复制逻辑集合可分散在各 *CompS* 中;(2) DSA-RM 的演化和复制仅针对 *CompS* 构件,经有限次演化,系统仍为自适应系统.

图 1 给出 DSA-RM 的框架模型,包括两个方面:策略逻辑 CompS 和控制逻辑 CompC.CompS 是数据复本管理策略构件集合,每个 CompS 构件均可用于独立实现一套完整数据复本管理功能.CompS 通过统一接口接入 DSA-RM,并按照 DSA-RM 体系结构描述语言描述本构件的演化和复制逻辑.CompC 包括了 Heart,Sensor,Evolve,Multiply 构件.Sensor 通过 CompS 的统一接口和连接件监听 CompS 构件的运行状态.Sensor 还需要监视整个网格节点的运行状态及节点间的运行状态.Info 存储该节点的运行状况.Logic 统一存储 CompS 的演化和复制逻辑.Heart 负责演化逻辑的归并及演化和复制的触发.Evolve 和 Multiply 负责 CompS 的演化和复制,通过 Sensor 将演化或复制消息传递给相关的 CompS.

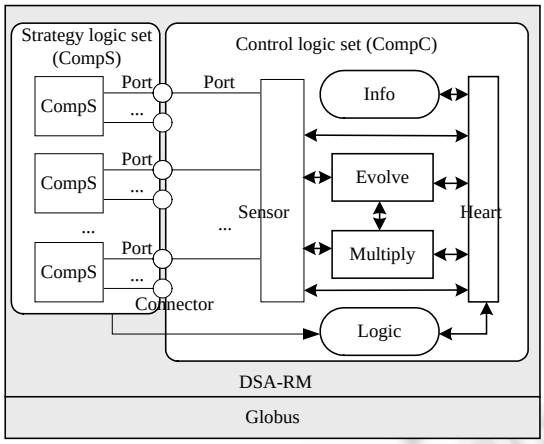

Fig.1 The Architecture of the DSA-RM

图 1 DSA-RM 体系结构模型

可用 XML 语言描述 DSA-RM 软件体系结构[28]的构件和连接件,以及 CompS 构件的演化规则及运行环境状态.例如,一个数据复本管理策略的构件描述如下:

```
〈Component name="pushStrategy"〉
  〈Port name="updateReplica"〉
    〈PubInterface name="update"/PubInterface〉
    〈PubInterface name="deleteOld"/PubInterface 〉
    〈Extn name="getReplica"〉
      〈Source name="ReplicaSource"〉
       〈Generator Location="192.168.0.12"/Generator〉
       〈Copy Location="192.168.0.13; 192.168.0.15"/Copy〉
      〈/Source〉
    〈/Extn〉
    〈Beha name="update"〉
        〈call name="com.replica.update"〉
          〈param name="replicaID"〉
            〈type value="int"/type〉
          〈/param〉
          …
          〈param name="content"〉
            〈type value="String"/type〉
          〈/param〉
      〈/call〉
    〈/Beha〉
    〈Msg name="Content"〉
    …
    〈/Msg〉
    〈Msg name="Return"〉
      〈Success value="1"/Success〉
      〈False value="0"/False〉
    〈/Msg〉
    …
    〈Cons name="Global"〉
      〈init name="State"〉
      …
      〈/init〉
      …
      〈pre-cond name="trigger"〉
    〈trigger name="outdate"〉
```

```
                〈outdate value=2006-01-20-20:00:00 /outdate〉
            〈/trigger〉
            …
            〈trigger name="msg"〉
                〈msg value="update"/msg〉
            〈/trigger〉
        〈/pre-cond〉
        …
        〈post-cond name="Other"〉
        …
        〈/post-cond〉
    〈/Cons〉
    〈Cons name="Update"〉
        〈init name="State"〉
          …
        〈/init〉
        …
        〈pre-cond name="trigger"〉
            〈trigger name="outdate"〉
                〈outdate value=2006-01-20-20:00:00 /outdate〉
            〈/trigger〉
            …
        〈/pre-cond〉
        …
        〈post-cond name="Other"〉
        …
        〈/post-cond〉
    〈/Cons〉
    〈Cons name="Update"〉
    …
    〈/Cons〉
  〈/Port〉
  〈Nfun name="detail"〉
  …
  〈/Nfun〉
〈/Component〉
```

2 DSA-RM 实现及性能分析

策略构件复制到节点之前,需要Heart为构件创造运行条件.网格节点的计算资源仅影响数据复本管理策略的执行时间,因此可以认为数据复本管理策略集合 S 在网格节点 N 上存在的充要条件为节点 N 为策略 S 分配了数据复本存储空间.此外,多策略可能共用数据项.因此,DSA-RM 的实现需要重点考虑两个问题:(1) 如何在多个节点间复制策略构件;(2) 如何管理同一节点上多个策略复本.

2.1 多节点策略复制

设网格节点集合 $N_G=\{N_0,N_1,\ldots,N_L\}$.DSA-RM 中任意网格节点 $N\in N_G$ 可表示为一个三元组 $N=(D,S,M)$,其中:$D=\{D_0,D_1,\ldots,D_k\}$为网格节点存储资源集合;$S=\{S_0,S_1,\ldots,S_m\}$为网格节点 N 上运行的数据复本管理构件集合;M 为一个 0/1 矩阵,表示数据复本管理策略集合 S 和网格节点存储资源 D 的映射关系.数据复本管理策略 S 在网格节点 N 上的状态可表示为一个五元组 $S=(ID,R,D,V,Q)$,$R=\{R_0,R_1,\ldots,R_n\}$表示当前通过策略 S 存储在节点 N 上的复本集合;$D=\{D_0,D_1,\ldots,D_k\}$表示节点 N 为策略 S 分配的存储资源集合;V 为一个 0/1 矩阵,表示 R 和 D 之间的映射关系;Q 表示节点 N 为策略 S 提供的存储空间质量保证,可表示为一个四元组(average,decrease,increase,allocate).其中:average 表示空间分配的平均值;decrease 和 increase 分布表示每次减少或增加空间时的数量;allocate 表示实际分配的空间数量.数据复本管理策略集合 S 在网格节点 N 上共生的充要条件为$\forall S_i\in S$,$\exists D_j\in N.D,m_{ij}=1$.

CompS 构件复制算法的伪代码如下:

Multiply(S_c,N_s,N_D) //S_c is the Component of CompS, N_s is the source node, N_D is the node set of destination.

On node N_s:{

N_s send *m-request-msg*=($S_c.ID$,$S_c.Q$,N_s,N_D) to every node in N_D;

Waiting *m-response-msg* sent back from N_D;}

On node each node $N_k \in N_D$ as receive *m-request-msg*:{

$$D^s=\bigcup_{i=0}^{k}\left(\left(\sum_{j=0}^{m} m_{ji}=0\right)?\{D_i\}:\phi\right);Surplus=Quantity(D^s);\ times=1;$$

If ($Surplus<S_c.Q.average$={

$$Surplus=Surplus+\sum_{i=0}^{m}\max(0,(S_i.Q(average-allocate)));$$

$$S'=\bigcup_{i=0}^{k}((S_i.Q.(allocate-average)>0)?\{S_i\}:\phi)$$

For each $S_i \in S'$ do $S_i.Q.allocate=S_i.Q.average$;}

While($Surplus<S_c.Q.average-times\times S_c.Q.decrease$=do{

$$Surplus=Surplus+\sum_{i=0}^{m}((S_i.Q(average-allocate)>times\times decrease)?decrease:0);$$

$$S'=\bigcup_{i=0}^{k}((S_i.Q.(allocate-average)>times\times decrease)?\{S_i\}:\phi);$$

For each $S_i \in S'$ do $S_i.Q.allocate=S_i.Q.average-S_i.Q.decrease$;}

For each $S_i \in S'$ do{

While($S_i.Q.allocate>Quantity(S_i.D)$) do{$S_i$ delete replica by strategy and merge storage distinct;};}

D'_s =Allocate(D^s,$Surplus$); // Allocate $D'_s \subseteq D^s$ for S_c use Knapsack algorithm

$N_k.S=N_k.S\cup S_c$;for each $D_j \in D'_s$ do{m_{cj}=1}; $S_c.Q.allocate=Surplus$;

Send *m-response-msg* to node N_s; // *m-response-msg*=($S_c.Q$,N_k);

Waiting *m-deploy-msg* sent back from N_s;}

On node N_s as receive *m-response-msg* from $N_k \in N_D$:{

Record info of N_s;Send *m-deploy-msg* to N_k} // *m-deploy-msg* is the binary code of S_c

On node each node $N_k \in N_D$ as receive *m-response-msg*:{Deploy the S_c}

当按上述算法向一个节点上部署新策略时,需要为该策略分配存储空间.当剩余空间不足时,需缩减其他策略的复本空间.缩减的方式采用轮询递减的方式,即每个策略(包括需复制的策略)按照定义的 decrease 的数值释放存储空间,其中公共复本不释放,直到获取新策略所需要的存储空间.由此,策略 S 在节点 N 上部署的时间复杂度为 $O(m\times k^2)$,其中:m 为节点 N 上复本管理构件的数量;k 为 N 上存储空间集合的元素数量.

2.2 多策略复本管理

网格节点为数据复本管理策略分配的存储空间直接影响数据访问效率[5−15].多策略共用复本可提高数据复本管理策略所获得的储存空间.此外,同一网格节点上的不同 CompS 构件可在节点内部互访数据以降低网络开销,从而提高性能.对网格节点 N,$\forall S_m,S_t \in N.S$,定义 $RShare(S_m,S_t)=S_m.R\cap S_t.R$ 表示策略 S_m 和 S_t 的公共复本.同一网格节点上任意两种策略的公共复本可存储在两个策略共用的存储资源中,即 $\forall S_m,S_t \in N.S$ 必然成立.定义 $DShare(S_m,S_t)=S_m.D\cap S_t.D$ 表示策略 S_m 和 S_t 在节点 N 上的共用存储资源集合.进而可把一个复本在网格节点 N 上的状态表示为一个三元组 $R=(gID,S^R,D)$,其中:gID 表示复本的全局唯一标识;S^R 是 R 在节点 N 上所属的策略集合;D 是复本存放的存储空间.在网格中,复本管理原子操作包括:读取、删除和更新 3 类.显然,当策略构件更新复本 R 时,将触发 $R.S^R$ 的所有策略按照策略内置的算法更新其他节点的复本;当删除策略 S_i 的复本 R 时,则将 S_i 从 $R.S^R$ 中删除,只有当 $R.S^R=\varnothing$时,才真正删除该复本;当策略 S_i 读取复本时,首先在本节点所有策略的复本中

搜索,然后再按照策略内置的复本搜索算法在全网格搜索.

2.3 性能分析

DSA-RM 将以缩减单节点上每个 CompS 构件的复本存储空间为代价,实现虚拟组织的数据复本管理的动态自适应.本节将分析多策略对数据复本访问的性能影响.

设在网格中部署了数据复本管理构件 S 的节点集为 $Gdeploy(S)=\bigcup_{i=0}^{L}((S\in N_i.S)?\{N_i\}:\phi)=\{N_0',N_1',\ldots,N_c'\}$.定义函数 $f_S(x)$为策略 S 在存储空间为 x 时的复本命中率,该函数显然是递增的.可根据 S 定义的复本访问优先顺序划分 $Gdeploy(S)$.

定义 7. 向量 $P=(P_0,P_1,\ldots,P_n)$,其中:$P.P_i=\{V_{i0},V_{i1},\ldots,V_{ia}\}$;$V_{ij}\subseteq Gdeploy(S)$.任取 $0\le i\le n$,若 P 同时满足条件:

(1) $(\forall V_{im},V_{in}\in P_i,V_{im}\cap V_{in}=\varnothing)\wedge\left(\bigcup_{j=0}^{a}V_{ij}=Gdeploy(S)\right)$;

(2) $\forall P_i,P_{i+1},(\exists V_{(i+1)n}\in P_{i+1})\wedge(\exists V_{im},V_{in}\in P_i)\wedge(V_{im}\cup V_{in}\subseteq V_{(i+1)n})$;

(3) $P_0=\{\{N_0'\},\{N_1'\},\ldots,\{N_c'\}\}$且 ${}^{I}P_n=\{\{N_0,N_1,\ldots,N_c\}\}$,则称 P 是 $Gdeploy(S)$的一个划分向量.

推论 1. $\forall N_i\in Gdeploy(S)$,存在向量 $Path(S,N_i)=(K_0,K_1,\ldots,K_n)$满足条件:任取 $0\le j\le n,N_i\in K_j\wedge K_j\subseteq P_j$,称 $Path(S,N_i)$为 N_i 基于策略 S 的路径.

证明:由定义 7 的条件(1)得:任取 $0\le i\le n,P$ 满足条件:$\bigcup_{j=0}^{a}V_{ij}=Gdeploy(S)$,则必存在 $0\le j\le a,N_i\in V_{ij}$,设 $K_i=V_{ij}$ 即构造出 $Path(S,N_i)$.推论 1 得证. □

可定义向量 $CostRate=(CR_0,CR_1,\ldots,CR_n)$,其中:$CR_i$ 是二元组$(cost_i,rate_i)$,$cost$ 是 K_{i-1} 与(K_i-K_{i-1})节点间集合的复本传输开销;$rate$ 是在 K_i 节点集合中命中但在 K_{i-1} 节点集合不命中的概率.显然,

$$rate_i=f_S(Quantity(K_i))-f_S(Quantity(K_{i-1}))=c(i),$$

$\Gamma(i)$体现了随缓存空间与命中率的关系.$costi=C(i)$.由此,访问一个复本的时间开销可表示为

$$T\approx\int_1^n(C(i)\times\Gamma(i)).$$

对于 $Path(S,N_0)$,如果存在节点 $N_j\in K_j$ 且在 N_j 上存在多个策略,则访问一个复本的时间开销为

$$T'\approx\int_1^n(C(i)\times\Gamma_{share}(i)).$$

设多策略状态和单策略状态二者所分配存储空间的差值为σ,则

$$\begin{aligned}\Delta T&=T_{share}-T\\&\approx\int_1^n(C(i)\times(\Gamma_{share}(i)-\Gamma(i)))\\&=\int_j^n(C(i)\times(\Gamma_{share}(i)-\Gamma(i)))\\&=\sigma\times\int_j^n(C(i)\times((f(Q(K_{i-1})))'-(f(Q(K_i)))').\end{aligned}$$

由此可得,复本访问接近均匀分布的策略,复本命中率随复本存储空间大小呈线性增长,则 $f(Q)$的导数接近 0,那么在节点上与其他策略共用复本空间对性能基本无影响;复本访问接近正态分布的策略,由于复本命中率随复本存储空间大小以渐近线方式增长,则如果在 N 基于 S 的路径 $Path(S,N)$中,K_i 集合存在多策略节点,则 i 值越大,$f(Q)$的导数越小,对性能影响越小.

3 模拟实验

本节将在 OptorSim[29]基础上模拟使用单一数据复本管理策略和使用多数据复本管理策略,并比较两种方式的数据访问开销.

设计如图 2 所示的网格拓扑环境.考虑 Internet 上的网络冲突,同一区域节点间带宽高于区域间节点带宽.实验中,考虑网络冲突设置带宽如下:通常的 Internet 带宽 K=10Mb;通常的 Intranet 带宽 M=100Mb;通常的局域

网带宽 G=1000Mb.设每个数据复本为 1MB,消息的大小为 0.01M,所访问复本的更新率为 10%.考虑为不同的 VO 提供不同的服务质量且各 VO 内节点间关系不同.图中黑色节点为数据产生主节点,即网格共用的所有数据产生和更新都由该节点完成.灰色节点属于 VO-W;白色节点属于 VO-R;标号为白色,底色为深灰色节点既属于 VO-W 也属于 VO-R.VO-W 节点的数据要求实时更新,且节点间还存在仅在该 VO 中可由多节点写的数据;VO-R 节点的数据只要求定时更新,且对不同的数据复本,要求不同.考虑两种均可运行在 VO-R 和 VO-W 的复本管理策略 Sr 和 Sw,差别在于:Sr 采用 Pull 的方式更新复本,即当数据复本过期时才去获取新的数据复本;Sw 采用 Push 的方式更新复本,即当主节点更新数据时,直接推向缓存了复本的节点,这要求主节点在更新数据前,首先写无效缓存的复本.模拟中,除主节点外,设每个节点的复本存储空间为 20G(通常是一个服务器 25%的存储空间);主节点共产生数据 1000G.考虑两种以缓存空间为自变量的复本命中率函数:(1) $f(x)=x/1000$,即数据访问为平均分布时,命中率为线性;(2) $g(x)=1-\frac{1}{\sqrt{x}\times e^{x/1000}+1}$,即数据访问为正态分布时,命中率为趋近 1 的渐近线.模拟两种复本命中率情况下:Sr 和 Sw 分别在 VO-R(Sr-VO-R)和 VO-W(Sw-VO-W)上使用(此时,将深灰色节点记入 VO-R)、Sr 在整个网格系统使用(Sr-G)、Sw 在整个网格系统使用(Sw-G)以及 Sr 和 Sw 同时在网格系统使用(Sr-Sw)的平均复本访问开销.

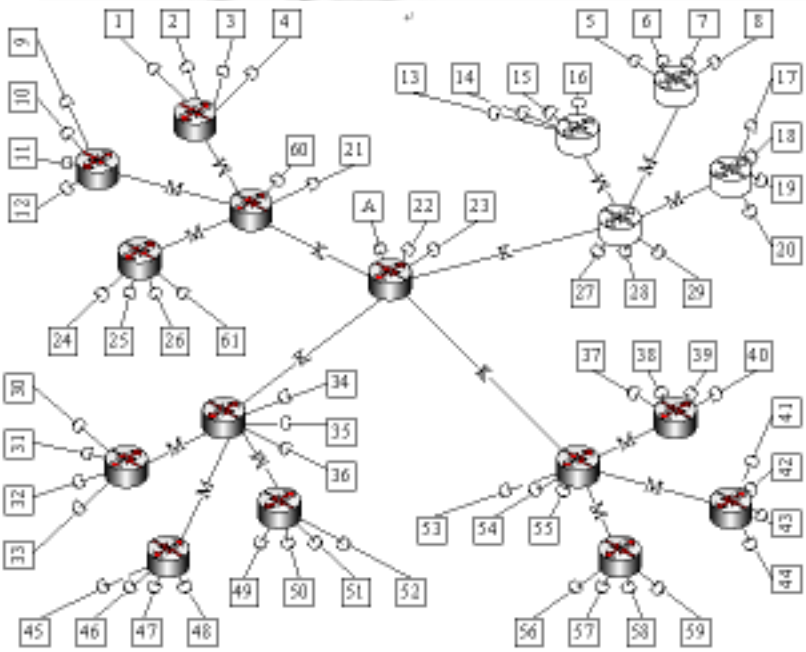

Fig. 2 Network topology used for simulations

图 2 模拟实验网络拓扑

图 3 给出了模拟测试结果.其中 X 轴表示每分钟平均每个节点的数据复本请求次数;Y 轴是响应时间,单位为 s.图 3(a)中($f(x)=x/1000$),复本缓存命中率低,由于网络冲突开销和写数据开销,在每个节点访问次数增加到 5 时,响应时间即开始接近指数性增长.采用混合策略(曲线⑤)的响应时间虽然高于 Sr 和 Sw 分别在 VO-R 和 VO-W 上使用(曲线①和曲线②),但只相当于 Sw 在整个网格系统使用(曲线④)以及 Sr 和 Sw 同时在网格系统使用(曲线③)的 50%.图 3(b)中$\left(g(x)=1-\frac{1}{\sqrt{x}\times e^{x/1000}+1}\right)$,复本缓存命中率较高,响应时间基本为线性增长.采用混合策略(曲线⑤)的响应时间虽然高于 Sr 和 Sw 分别在 VO-R 和 VO-W 上使用(曲线①和曲线②),但只相对于 Sw 在整个网格系统使用(曲线④)以及 Sr 和 Sw 同时在网格系统使用(曲线③)的 65%.

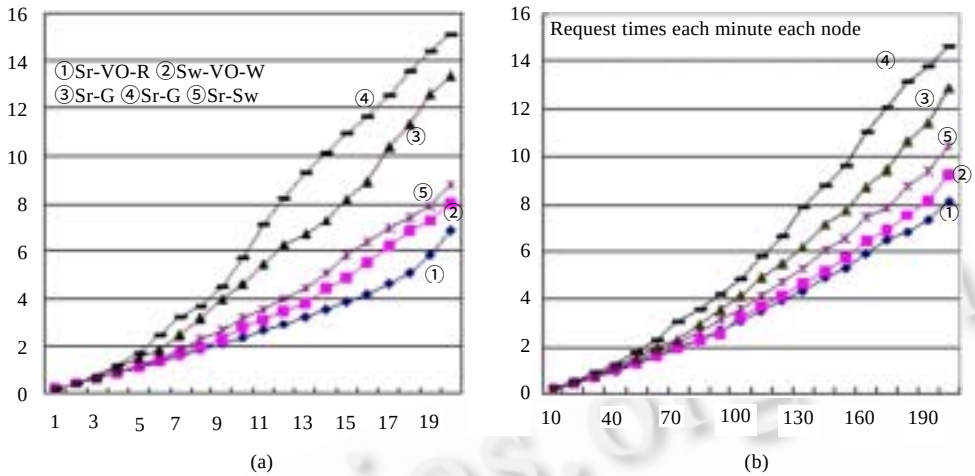

Fig.3 Average response time of Sr-VO-R, Sr-VO-R, Sw-VO-W, Sr-G, Sr-G, Sr-Sw

图 3 Sr-VO-R,Sr-VO-R,Sw-VO-W,Sr-G,Sr-G,Sr-Sw 的平均响应时间

选取节点 33,34,41,考察对单个节点的影响.对 $f(x)=x/1000$,测试上述节点每分钟请求 5,10,15,20 个复本的平均响应时间;对 $g(x)=1-\frac{1}{\sqrt{x}\times e^{x/1000}+1}$,测试每分钟请求 50,100,150,200 个复本的平均响应时间.比较内容包括:节点 33 运用 Sr-VO-R 和 Sr-Sw 的响应时间;节点 34 运用 Sr-VO-R,Sw-VO-W 和 Sr-Sw 的响应时间;节点 41 运用 Sw-VO-W 和 Sr-Sw 的响应时间.图 4 给出了模拟测试结果.如图 4 所示,同时部署了 Sr 策略和 Sw 策略的节点 44 受性能影响最大,其他节点几乎不受影响.

4 小结和下一步的工作

软件体系结构为实现网格复本管理自适应环境提供了新的解决思路.本文提出了软件体系结构驱动的可动态自适应系统运行环境的数据复本管理架构 DSA-RM,并在形式化描述 DSA-RM 体系结构和相关命题证明的基础上,讨论了适用于数据网格复本管理的软件体系结构驱动的动态自适应规则;设计了 DSA-RM 的实现框架和关键技术,并给出了构件描述方法;最后,采用模拟验证了多策略对整体性能的提升.DSA-RM 架构的主要优点在于:(1) 网格用户可按照 DSA-RM 模型标准为新的 VO 开发复本管理策略并动态部署;(2) 网格系统可根据服务质量动态调整复本管理策略以满足应用需求;(3) 网格系统可根据应用环境变化触发数据复本管理构件的演化以提高系统整体效率.

下一步,我们计划在上海医学网格中测试应用 DSA-RM.上海医学网格的研究分为两个阶段:第 1 阶段主要是上海耳鼻喉医学网格的研究与实现,该网格部署在上海、北京两地,主要包括耳鼻喉临床辅助诊断与辅助治疗和耳鼻喉流行病学调查研究两大功能;第 2 阶段将进入数据量更大的乳腺癌医学网格的研究实现.当前,耳鼻喉医学网格已建设完成,第 2 阶段已进入设计阶段.我们计划首先在已建设完成的耳鼻喉医学网格上应用测试 DSA-RM,然后将其在乳腺癌医学网格上推广实现.

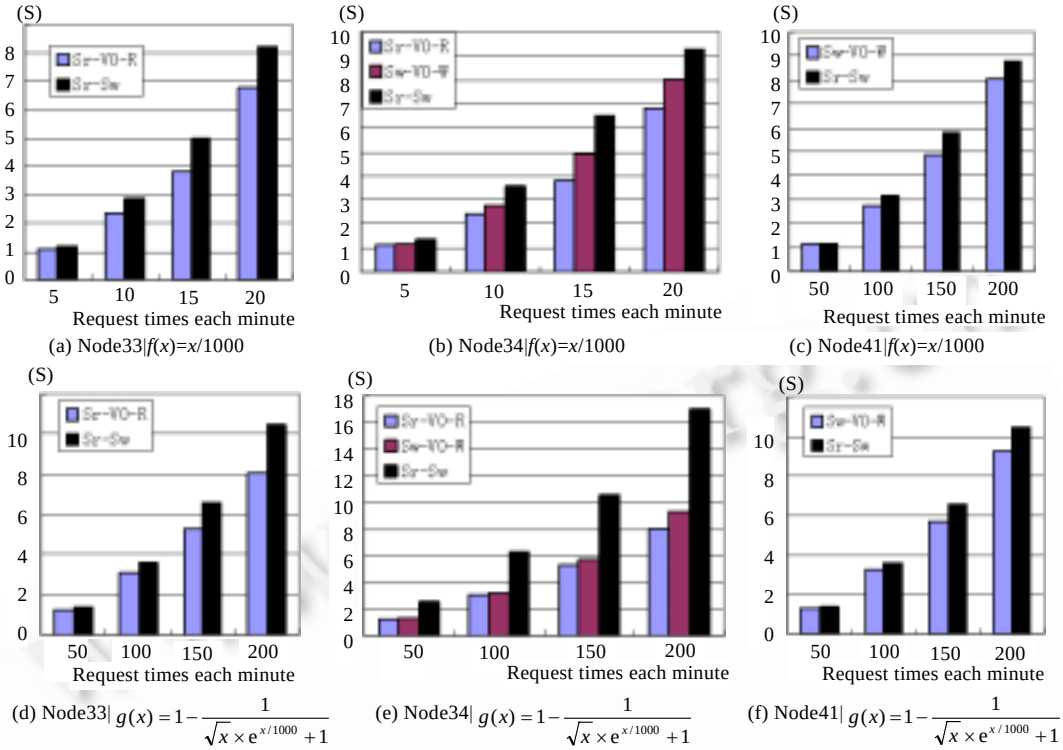

Fig.4 Compare response time of node 33, 34, 41 between mix and single strategy

图 4 比较节点 33,34,41 使用混合策略和单一策略的平均响应时间

References:

[1] Foster I, Kesselman C, Tuecke S. The anatomy of the grid: Enabling scalable virtual organizations. Int'l Journal of High Performance Computing Applications, 2001,15:200−222.

[2] Chervenak A, Foster I, Kesselman C, Salisbury C, Tuecke S. The data grid: Towards an architecture for the distributed management and analysis of large scientific data sets. Journal of Network and Computer Applications, 2001,23(3):187−200.

[3] Baru C, Moore R, Rajasekar A, Wan M. The SDSC storage resource broker: In: Proc. of the Conf. on the IBM Centre for Advanced Studies on Collaborative Research (CASCON'98). Toronto: IEEE Computer Society Press, 1998.

[4] Allcock B, Bester J, Chervenak AL, Foster I, Kesselman C, Meder S, Nefedova V, Quesnel D, Tuecke S. Data management and transfer in high performance computational grid environments. Parallel Computing Journal, 2002,28(5):749−771.

[5] Stockinger H, Samar A, Allcock B, Foster I, Holtman K, Tierney B. File and object replication in data grids. Journal of Cluster Computing, 2002,5(3):305−314.

[6] Foster I, Kesselman C. A data grid reference architecture. Technical Report, GriPhyN-2001-12, 2001. http://www.griphyn.org

[7] Deelman E, Kesselman C, Mehta G, Meshkat L, Pearlman L, Blackburn K, Ehrens P, Lazzarini A, Williams R, Koranda S. GriPhyN and LIGO, building a virtual data Grid for gravitational wave scientists. In: Proc. of the 11th IEEE Int'l Symp. on High Performance Distributed Computing HPDC-11 2002 (HPDC'02). Washington: IEEE Computer Society Press, 2002. 225−234.

[8] Bernholdt D, Bharathi S, Brown D, Chanchio K, Chen M, Chervenak A, Cinquini L, Drach B, Foster I, Fox P, Garcia J, Kesselman C, Markel R, Middleton D, Nefedova V, Pouchard L, Shoshani A, Sim A, Strand G. The earth system grid: Supporting the next generation of climate modeling research. Proc. of the IEEE, 2005,93(3):485−495.

[9] 2006. http://www.ncbiogrid.org

[10] 2006. http://biogrid.icm.edu.pl

[11] 2006. http://www.globus.org

[12] Chervenak A, Deelman E, Foster I, Guy L, Hoschek W, Iamnitchi A, Kesselman C, Kunszt P, Ripeanu M, Schwartzkopf B, Stockinger H, Stockinger K, Tierney B. Giggle: A Framework for Constructing Scalable Replica Location Services. In: Proc. of the IEEE/ACM SC2002 Conf. (SC'02). Washington: IEEE Computer Society Press, 2002. 1−17.

[13] Chervenak AL, Palavalli N, Bharathi S, Kesselman C, Schwartzkopf R. Performance and Scalability of a Replica Location Service. In: Proc. of the 13th IEEE Int'l Symp. on High Performance Distributed Computing, 2004. 182−191.
[14] Cai M, Chervenak A, Frank M. A peer-to-peer replica location service based on a distributed Hash table. In: Proc. of the ACM/IEEE SC 2004 Conf. Pittsburgh: IEEE Computer Society, 2004. 56−56.
[15] Stoica I, Morris R, Karger D, Kaashoek MF, Balakrishnan H. Chord: A scalable peer-to-peer lookup service for internet applications. IEEE/ACM Trans. on Networking, 2003,11(1):17−32.
[16] Park SM, Kim JH, Ko YB, Yoon WS. Dynamic data grid replication strategy based on Internet hierarchy. In: Li M et al. eds. Proc. of the 2nd Int'l Workshop on Grid and Cooperative Computing (GCC 2003). LNCS 3033, 2004. 838−846.
[17] Foster I, Kesselman C, Tuecke S. The anatomy of grid: Enabling scalable virtual organizations. Int'l Journal of Supercomputer Applications, 2001,15(3):200−222.
[18] Shang EF, Du ZH. Efficient grid service location mechanism based on virtual organization and the small-world theory. Journal of Computer Research and Development, 2003,40(12):1743−1748 (in Chinese with English abstract).
[19] Shaw M, Garlan D. Software Architecture: Perspectives on An Emerging Discipline. New Jersey: Prentice Hall, 1996.
[20] Zhang SK, Zhang WJ, Chang X, Wang LF, Yang FQ. Building and assembling reusable components based on software architecture. Journal of Software, 2001,12(9):1351−1358 (in Chinese with English abstract).
[21] Sun CA, Jin MZ, Liu C. Overviews on software architecture research. Journal of Software, 2002,13(7):1228−1237 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/13/1228.pdf
[22] Sam M, Ieven LD, Nico J. Self-Adapting concurrency: The DMonA architecture. In: Proc. of the 1st Workshop on Self Healing Systems (WOSS'02). Charleston: ACM press, 2002. 43−48.
[23] Reill YDA, Tal EBB, Laws A. An instrumentation and control based approach for distributed application management and adaptation. In: Proc. of the 1st Workshop on Self Healing Systems (WOSS'02). Charleston: ACM press, 2002. 61−66.
[24] Garlan ID, Shaw M. An introduction to software architecture. Technique Report, CMU/SEI-94-TR-21, Carnegie Mellon University, 1994.
[25] Gao Y, Wang GR, Zhao HQ. An abstract model of software architecture. Chinese Journal of Computers, 2002,25(7):730−736 (in Chinese with English abstract).
[26] Taylor RN, Medvidovic N, Anderson KM, Dubrow DL. A component and message based architectural style for GUI software. IEEE Trans. on Software Engineering, 1996,22(6):390−406.
[27] Medvidovic N, Taylor RN. A classification and comparison framework for software architecture description languages. IEEE Trans. on Software Engineering, 2000,26(1):483−491.
[28] Zhao WY, Zhang Z. A research on XML based architecture description language-XBA. ACTA Electronica Sinica, 2002,12(A): 2036−2039 (in Chinese with English abstract).
[29] Bell WH, Cameron DG, Millar AP, Capozza L, Stockinger K, OptorSim FZ. A grid simulator for studying dynamic data replication strategies. Int'l Journal of High Performance Computing Applications, 2003,17(4):403−416.

附中文参考文献:

[18] 尚尔凡,都志辉.基于虚拟组织和小世界模型的高效网格服务定位机制.计算机研究与发展,2003,40(12):1743−1748.
[20] 张世琨,张文娟,常欣,王立福,杨芙清.基于软件体系结构的可复用构件制作和组装.软件学报,2001,12(9):1351−1358.
[21] 孙昌爱,金茂忠,刘超.软件体系结构研究综述.软件学报,2002,13(7):1228−1237. http://www.jos.org.cn/1000-9825/13/1228.pdf
[25] 高原,王国仁,赵会群.软件体系结构的抽象模型.计算机学报,2002,25(7):730−736.
[28] 赵文耘,张志.基于 XML 的构架描述语言 XBA 的研究.电子学报,2002,12(A):2036−2039.

陈磊(1976-),男,河南信阳人,博士生,主要研究领域为网格计算技术,高性能计算技术.

李三立(1935-),男,教授,博士生导师,中国科学院院士,CCF 高级会员,主要研究领域为网格计算技术,高性能计算技术.