基于动态优先级的数据复制反射式中间件研究*)

徐丽萍 卢炎生 袁 刚 刘 钧
(华中科技大学计算机科学与技术学院 武汉 430074)

摘 要 反射式中间件是当前中间件研究的热点,其特点是能够对中间件系统的灵活性和自适应性提供良好的支持。数据网格通常处于动态变化的环境和不同的用户需求之中,对灵活性和自适应性的支持是一个良好的数据网格和数据复制系统所必需的,通过动态配置数据和SE(storage element)的优先级,DPRM(Dynamic priority based Reflective Middleware for Data Replication)提供了这一支持。使用 OptorSim 仿真的结果表明,DPRM 有着良好的性能。

关键词 数据网格,适应性,数据复制

Research on Dynamic Priority Based Reflective Middleware for Data Replication

XU Li-ping LU Yan-sheng YUAN Gang LIU Jun
(School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China)

Abstract Reflective middleware is a hot topic at present, which enables a high degree of integration flexibility and adaptability. DRPM provides good flexibility and adaptability in face of environment varying and different user requirements which are absolutely necessary to Data Grid and Data Replication systems by dynamicly configuring the priority of data and SE. Results from OptorSim simulator are promising.

Keywords Data grid, Adaptability, Data replication

1 引言

计算网格被用来解决广域网中分布的资源共享、互联和互操作问题。随着数据密集型应用的飞速发展,数据呈爆炸式增长,如大型核强子碰撞器(LHC)、激光干涉仪重力波观测(LIGO)和 Sloan 数字天空观测(SDSS)等的应用每天要产生数以 Terabyte 计的数据,特别是 LHC 每年要产生数以 Pertabytes 计的数据,地理上广泛分布的科学家都希望能够快速访问、分析和使用这些庞大的广域分布的数据,数据网格技术应运而生,而且正成为网格技术的另外一个研究重点。

要确保对如此庞大的分布数据集的有效访问是对网格设计者的一个巨大挑战,而广域网的高延迟是对分布式数据集访问的主要障碍。数据复制技术通过把大量的副本数据存储在不同的结点上,大大提高了数据的可靠性,降低了对网络带宽的消耗,增加了系统的容错性。

在给定有限的数量和容量的存储设备的情况下,如何提高数据访问效率,是数据复制需要解决的一个重大问题。一个典型的数据网格环境中,许多用户共享有限的存储资源,因此对资源利用的优化非常重要。为数据副本和存储资源设定动态优先级可以达到这一目的。

当前对数据复制技术的研究大都基于静态或用户驱动的模式(user-driven)。然而,网格是一个高度自治并且动态变化的系统,资源的可靠性和网络的性能乃至数据访问请求都处在不断变化之中。因此,对灵活性和自适应性的支持是一个良好的数据复制系统所必需的,本文通过反射式中间件给出解决这一问题的一种方法。

2 相关工作

当前的数据复制技术大都基于静态或用户驱动的模式。如 LIGO 系统的数据复制和注册是基于拉方式(pull-based)的,把数据集复制到一个 LIGO 站点,然后通过复制定位服务(Replica Location Service)进行注册[1],这就是一种典型的用户驱动的模式。基于预测和代价的思想[2]可以一定程度上适应网络的动态变化和用户的动态需求,但这种预测是基于文件的内容相似性极大的假设($\forall i>0, |s_i|=|r_i-r_{i-1}|=1$)的,这种方法在实际运用中有着很大的局限性。HSM 系统中的复制优化服务体现了一定的反射思想[3],为了达到复制优化的目的,为每种存储器设计不同的代价估算器,并以 CEA (Component-Expert Architecture)构件的形式封装,便于动态替换。然而,其复制的代价模型是基于存储系统为主导的(即把网络代价作为次要因素),而且文中并没有涉及到反射的概念。

反射式中间件是一种通过因果相连的自表示(causally connected self-representation 或简称 CCSR)对自身行为提供自省和自适应性的中间件。反射式中间件通常包括元层和基层。元层实体负责建立和维护系统的自我描述,基层实体负责实现系统的业务功能,并将自描述信息映射到系统实际的状态和行为中。反射式中间件是当前中间件技术研究的一个热点,反射技术已经被引入各个领域。如 Lancaster 大学提出了一个可配置的多媒体中间件平台[4](他们现在仍在致力于 OPENCOM 模型的研究),Fassino 等人提出了一种基于构件(可动态配置)的操作系统 THINK[5]。然而,还没有用于数

*)本文由国家部委基础研究计划(10104010201)资助。徐丽萍 博士,副教授,研究方向为软构件技术、中间件技术等;卢炎生 教授,博士生导师,研究方向为数据库与软件工程等;袁 刚 硕士研究生,研究方向为软构件技术、中间件技术等;刘 钧 硕士研究生,研究领域为移动计算、中间件技术等。

据网格的反射式中间件出现。

数据网格管理的是分布式的异构数据，面对的是动态变化的环境和不同的用户需求。反射式中间件通过对自表示的自省提供对系统的重配置，以达到运行时的灵活性和自适应性。为了适应网络的动态变化和用户的动态需求，将反射思想引入到数据网格，无疑是一个理想的方案。本文的研究仅限于数据复制，在未来的工作中，将进一步研究反射技术在数据网格其它领域中的应用。

3 DPRM 体系结构

DPRM 的体系结构如图 1 所示，主要功能部件有复制管理器、事件管理器、资源监控器、事件仓库、事件注册服务、复制管理客户端等。

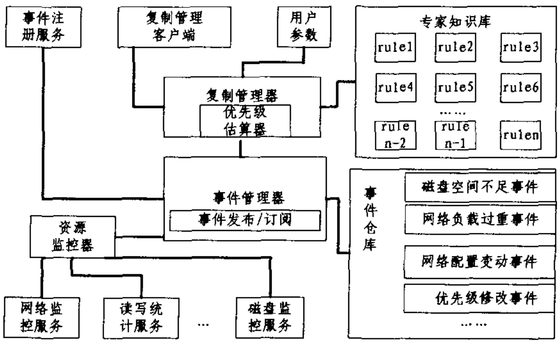

图 1 DPRM 的体系结构

当前许多数据复制模型都是基于用户驱动模型，为适应环境的动态变化，DPRM 采用了由资源监控器、事件管理器和事件仓库构成 DPRM 的事件发布/订阅模型。

事件订阅/发布模型是对系统上下文信息(context information)和系统自身元信息的抽象和约束，从而对系统进行调整。也就是说，事件订阅/发布模型提供了反射式中间件 DPRM 的自表示和自描述能力，因此，它构成系统的元层。事件注册服务是 DRPM 提供给外部的接口，主要为用户提供事件的注册功能。

4 DPRM 实现技术

4.1 事件管理器

事件管理器是事件发布/订阅模型的核心部件，在事件订阅/发布模型中起着枢纽的作用，它从资源监控器接收事件的发布，并接收事件仓库的事件订阅。事件经过事件管理器的注册后，就可以存入事件仓库，等候调度。比较重要的事件有磁盘空间不足事件、网络负载过重事件、网络配置变动事件、优先级修改事件等等。

4.2 资源监控器

在反射式体系结构 DPRM 中，资源监控器的职责是同基层的服务进行交互，它是系统的上下文感知部件，主要负责搜集必需的上下文信息和系统自身元信息。上下文信息主要包括副本的读写统计信息、网络流量信息、内存使用信息等。系统自身元信息主要包括磁盘空间信息、网络互联配置信息等。以上信息大都可以通过各自对应的服务进行搜集，如网络监控服务负责搜集网络流量信息，磁盘监控服务负责搜集磁盘的使用情况等。事实上，它还需要搜集一定的应用元数据信息(如副本的大小等)，所以资源监控器还需要和元数据服务进行交互，由于元数据服务不在本文讨论的范围之内，本文只是简单地认为这些必需元数据信息可以从系统接口获得。

4.3 复制管理器

复制管理器是 DPRM 的核心部件，是整个系统的决策器。事件管理器提交事件给复制管理器后，需要经过复制管理器的裁决，如优先级修改事件触发后，并不一定会修改某一个副本的优先级，是否真的修改需要复制管理器结合多方面的因素的考虑，如用户参数、专家知识库和代价估算器等。通过考虑多方面的因素，达到复制优化的目的。为了灵活适应数据网格中环境的动态变化以及用户的动态需求，DPRM 提供了用户参数接口，即运行时用户可以及时通过输入参数来改变系统的运行状态，系统会根据实际情况最大限度满足用户需求。复制管理器在进行复制决策时，通常会遇到多个事件同时产生的情况，如两个高优先级的副本都要求进行复制。在基于市场经济模型的复制策略中采取了反向拍卖(reverse auction)的机制，通过拍卖者的竞价解决冲突。这种拍卖机制的拍卖报文是通过 P2P 的方式进行洪泛传播的，这会造成很大的通信开销。DPRM 利用专家知识库(Expert knowledge Base)解决这一问题。专家知识库是一个预先定义好的规则集合，这些规则的设计来源于网格专家的实际经验。采用这种设计，其目的是为了保证 DPRM 的健壮性。这种设计借用了 CEA 体系结构[3]的思想。CEA 中，专家知识库的作用是消除多个构件之间的不兼容，这里则是用来消除多个事件之间的冲突。优先级估算器是复制管理器进行决策的另一个重要参考因素，它的作用是动态估算副本和 SE((Storage Element))的优先级。复制管理客户端是 DPRM 提供给外部的接口，借此用户可以很方便地了解到系统内部的运作情况，这种设计也是充分考虑到反射式中间件的开放性的。

4.4 优先级估算器

如图 2，6 个 SE 中，假设两个节点之间需要传送数据，那么这些数据经过 C 和 D 的概率是：$C_6^2-1/C_6^2=93.33\%$（仅仅 A 和 B 之间的数据传输不经过 C 和 D），所以，C 和 D 的利用率的使用是整个系统数据访问性能的决定性因素。本文将 C 和 D 这样的 SE 称为瓶颈 SE。DPRM 的任务之一就是使得瓶颈 SE 的利用率最大化。

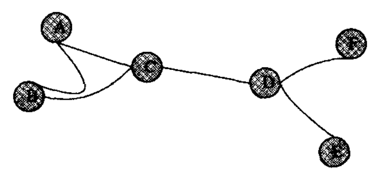

图 2 包括网格资源的网络拓扑结构

为了标识瓶颈 SE，DPRM 采取了运行时赋予它们更高的优先级的方法。每个数据对象 Di 的元数据项中都包含有动态优先级的子项，设为 P_{Di}，每个 SE 也设置一个动态优先级 P_{Si}，这两个动态优先级都由优先级估算器在运行时确定。

4.4.1 数据对象动态优先级 P_{Di} 的估算

对于 P_{Di}，优先级估算器应该考虑以下因素：

① 读统计 R_{Di}

R_{Di} 是系统对数据对象 Di 进行读操作的统计。对一个数据对象的读操作的频率是决定该数据对象是否有复制价值的重要标志，显然，对于读操作频繁的数据对象，应该赋予它更高的优先级。这可以从 DPRM 的读写统计服务中获得。

② 写统计 W_{Di}

W_{Di} 是系统对数据对象 Di 进行写操作的统计，同读操作

一样,写频率对于数据对象的动态优先级的设定至关重要。这也可以从 DPRM 的读写统计服务中获得。

③ 数据对象的大小 $SIZE_{Di}$

考虑数据对象大小是一个比较复杂的问题,一方面,将大文件进行复制一定会带来更多的时间和空间开销,另一方面,如果事先选择将大文件复制到恰当的 SE,将会大大提高系统的性能。因此,我们提出用平衡因子 λ 对其进行折中考虑,这里,λ 只和系统本身的元信息有关,由 DRPM 根据用户参数和专家知识库的规则动态确定。

④ 数据对象的最近访问时间 T_{Di}

最近访问时间无疑也是评估数据对象动态优先级必须考虑的因素。距离现在时刻最近的时间段内活动频繁的数据对象应该给予更高的优先级。

基于以上考虑,给出数据对象的动态优先级估算公式为:

$$P_{Di} = f1(R_{Di}, W_{Di}, \lambda SIZE_{Di}, T_{Di}) \quad (1)$$

4.4.2 SE 动态优先级 P_{Si} 的估算

对于 SE 优先级的确定,应该考虑以下因素:

① 网络拓扑结构

SE 处于网络中的什么位置,是决定其动态优先级的主要因素,本文中我们用函数 L 表示,例如图 2 中,可以设定:

$L(A)=L(B)=2$

$L(D)=5$

$L(E)=L(F)=1$

一旦网络拓扑结构确定,L 就不会改变。另外,DPRM 的网络监控服务会随时监控网络连接的变化,以便维持 L 的正确性。

② 网络流量统计 N

SE 上数据请求量越大,说明该 SE 被使用的频率很高,则应该被赋予高优先级。

③ SE 的存储容量 C

这个因素没有前两个因素那么重要,但也应该考虑。一般来说,网格设计者会把大容量的 SE 尽量设置到合适的位置。

基于以上考虑,给出 SE 的优先级估算公式为:

$$Psi = f2(L(Si), Nsi, Csi) \quad (2)$$

4.4.3 副本替换算法

在副本替换时,DPRM 采用了贪婪算法,其目的是最大化瓶颈 SE 上数据对象的优先级。当然,前提是必须满足专家知识库的规则。这种贪婪算法可以简单地表示如下:

if(Si has more space)

Replicate P_{Dj} to Si

else if($P_{Dj} >= P(P_{Si})$)

replace Dt with Dj

其中,Dj 为 DPRM 探测到的高优先级的数据对象 $Dt=\min\limits_{Di\in F_{Si}}(P_{Dj})$。$Si$ 为任意一个 SE, F_{Si} 为 Si 上的数据集合,$P(Psi)$ 是和 Psi 有关的一个函数,表示对于优先级不同的 SE,系统对其利用率要求不同。例如,对于瓶颈结点,其 Psi 较大,则尽量要求将高优先级的数据对象复制到该结点上。

设 Q 为 Si 替换前的状态,Q' 为替换后的状态,那么本次替换可以表示为:

$$T(Q, P_{Di}, P_{Si}, R_T) = Q' \quad (3)$$

其中,$R_T \subseteq R$,R 为专家知识库的规则集合,R_T 为本次替换 T 所对应的规则集合。设 U 为 SE 的利用率,则 U 应该是该 SE 上数据优先级之和的函数,即:

$$U = G(\textstyle\sum_{Di\in F_{Si}} P_{Di}) \quad (4)$$

$\sum_{Di\in F_{Si}} P_{Di}$ 值越大,U 就越大。这样经过替换 T 后,必然有 $U_{Q'} > U_Q$。经过若干次替换,由贪婪算法的特点可以知道,系统达到动态平衡时,瓶颈结点的 U 值应该是最大或者接近最大。

5 仿真及实验结果

为了评估 DPRM 的性能,我们采用 Optorsim 进行仿真。

5.1 仿真配置

如图 3,仿真实验中共有 9 个结点。每个 CE 具有相同的处理能力,每个任务运行 80 秒。每个 SE 具有 2GB 的存储能力。网络带宽为 100MB/s,工作负载分别为 100 个任务、500 个任务、1000 个任务。数据文件的大小为 100MB 和 1GB 之间。

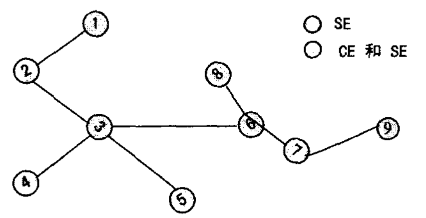

图 3 仿真实验拓扑图

5.2 仿真结果

为了说明问题,将 DPRM 和传统的 LRU(Least Recently Used)算法进行比较。比较了两个性能参数:任务完成时间和瓶颈 SE 利用率。

如图 4 所示,DPRM 的任务完成时间大大少于 LRU 算法。这是因为 LRU 算法只考虑了本地结点上文件的使用,当存储空间不够时,有可能把将来要用的文件删除。

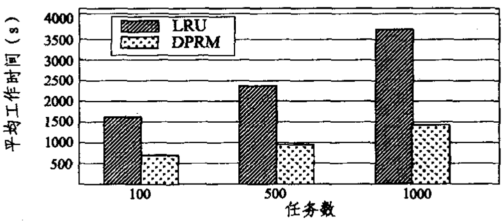

图 4 DPRM 和 LRU 的平均工作时间

而 DPRM 中,高优先级结点优先级会考虑整个系统,高优先级文件很有可能会因为其在系统中优先级高而保存,因此,当另一个任务需要该文件时,能够从高优先级结点快速得到。

如图 5 所示,稳态时 DPRM 的瓶颈 SE 利用率达到 80%,这也是因为在 DPRM 中高优先级结点保存了当前的高优先级文件,所以提高了文件的重用率。

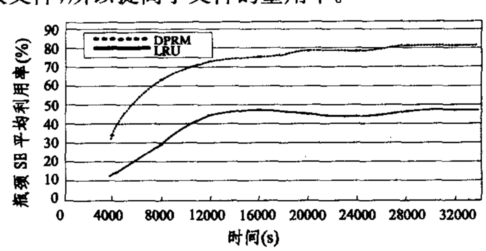

图 5 运行 1000 个任务瓶颈 SE 平均利用率

结束语 本文将反射式中间件引入到数据网格中,在数据复制方面取得了一定进展。通过 DPRM 的上下文感知部

件对动态环境的感知，搜集数据复制所需要的各种参数，从而作出正确的复制决策。由于反射式中间件能够适应动态变化的特征，其在数据网格中的其他领域必然也有用武之地，未来的工作中，我们将会将反射技术应用到元数据管理、数据传输等方面。

参考文献

[1] Chervenak A, Kesselman C, Schuler R, et al. Design and Implementation of a Data Replication Service

[2] Teng M, Junzhou L. A prediction-based and cost-based replica replacement algorithm research and simulation. IEEE, 2005

[3] Stockinger K, Stokinger H, Duka L, et al. Access Cost Estimation for unified Grid Storage Systems. IEEE, 2005

[4] Coulson G, Blair G S, Davies N, et al. Supporting Mobile Multimedia Applications Through Adaptive Middleware. IEEE, 1999

[5] Fassino J P, Stefani J B, Lawall j, et al. THINK: A Software Framework for Component-based Operating System Kernels// Usenix Aunual Technical Conference. Monterey (USA), June 2002

(上接第 269 页)

[9] Lopes C V, Ngo T C. Unit-Testing Aspectual Behavior[Position Paper]// 1st Workshop on Testing Aspect-Oriented Programs, 4th International Conference on Aspect-Oriented Software Development. New York: ACM Press, 2005

[10] Mortensen M, Alexander R T. Adequate Testing of Aspect-Oriented Programs. Colorado State University: Fort Collins, 2004

[11] Mortensen M, Alexander R T. An Approach for Adequate Testing of AspectJ Programs// 1st Workshop on Testing Aspect-Oriented Programs, 4th International Conference on Aspect-Oriented Software Development. New York: ACM Press, 2005

[12] Lemos O A L, et al. Testing Aspect-Oriented Programming Pointcut Descriptors// 2nd Workshop on Testing Aspect-Oriented Programs (WTAOP 2006). New York: ACM Press, 2006

[13] Ceccato M, Tonella P, Ricca F. Is AOP code easier or harder to test than OO Pcode // Workshop on Testing Aspect-Oriented Programs, 4th International Conference on Aspect-Oriented Software Development. New York: ACM Press, 2005

[14] Xu W, Xu D. A Model-Based Approach to Test Generation for Aspect-Oriented Programs// 1st Workshop on Testing Aspect-Oriented Programs, 4th International Conference on Aspect-Oriented Software Development. New York: ACM Press, 2005

[15] Zhao J. Tool Support for Unit Testing of Aspect-Oriented Software// OOPSLA'2002 Workshop on Tools for Aspect-Oriented Software Development. New York: ACM Press, 2002

[16] Zhao J. Data-Flow-Based Unit Testing of Aspect-Oriented Programs// 27th Annual International Computer Software and Applications Conference (COMPSAC'03). Boston: IEEE Computer Society, 2003

[17] Zhou Y, Richardson D, Ziv H. Towards a Practical Approach to Test Aspect-Oriented Software // 2004 Workshop on Testing Component-Based Systems (TECOS2004). Germany: GI, 2004

[18] Lemos O A L, Maldonado J C, Masiero P C. Structural Unit Testing of AspectJ Programs // 1st Workshop on Testing Aspect-Oriented Programs, 4th International Conference on Aspect-Oriented Software Development. New York: ACM Press, 2005

[19] Mackinnon T, Freeman S, Craig P. Endo-Testing: Unit Testing with Mock Objects // eXtreme Programming and Flexible Processes in Software Engineering - XP2000. Boston: Addison-Wesley, 2000

[20] Yamazaki Y, et al. A Unit Testing Framework for Aspects without Weaving// 1st Workshop on Testing Aspect-Oriented Programs, 4th International Conference on Aspect-Oriented Software Development. New York: ACM Press, 2005

[21] Massicotte P, Badri M, Badri L. Generating Aspects-Classes Integration Testing Sequences a Collaboration Diagram Based Strategy// 2005 Third ACIS Int'l Conference on Software Engineering Research, Management and Applications (SERA'05). Boston: IEEE Computer Society, 2005

[22] Badri M L, Badri, Naha M. A Use Case Driven Testing Process: Towards a Formal Approach Based on UML Collaboration Diagrams// FATES 2003. Heidelberg: Springer-Verlag, 2003

[23] Xie T, et al. Automated Test Generation for AspectJ Programs // Workshop on Testing Aspect-Oriented Programs, 4th International Conference on Aspect-Oriented Software Development. 2005

[24] Massicotte P, Badri L, Badri M. Aspects-Classes Integration Testing Strategy: An Incremental Approach. Lecture Notes in Computer Science, 2006, 3843: 158-173

[25] Souter A L, Pollock D S a L L. Testing with Respect to Concerns// International Conference on Software Maintenance (ICSM 2003). IEEE Computer Society, 2003

[26] Xu D, Xu W. State-Based Incremental Testing of Aspect-Oriented Programs // AOSD 2006. Bonn, Germany: New York: ACM Press, 2006

[27] Zhao J, Xie T, Li N. Towards Regression Test Selection for AspectJ Programs // 2nd Workshop on Testing Aspect-Oriented Programs (WTAOP 2006). New York: ACM Press, 2006

[28] Harrold M J, et al. Regression Test Selection for Java Software // ACM Conference on Object-Oriented Programming, Systems, Languages, Applications (OOPSLA2001). New York: ACM Press, 2001

[29] Xu G. A Regression Tests Selection Technique for Aspect-Oriented Programs // 2nd Workshop on Testing Aspect-Oriented Programs (WTAOP 2006). New York: ACM Press

[30] JUnit. JUnit. 2004 [cited 2007 March]. Available at: http://www.junit.org

[31] Massol V. Junit in Action 中文版. 鲍志云, 译. 北京: 电子工业出版社, 2005

[32] Parasoft. Jtest manuals version 4. 5. 2007 [cited April]. Available at: http://www.parasoft.com/

[33] Miles R. aUnit: Unit Testing for Cross-Cutting Concerns. 2004 [cited March]. Available at: http://aunit.sourceforge.net/

[34] Xie T, et al. Detecting Redundant Unit Tests for AspectJ Programs. University of Washington Department of Computer Science and Engineering: Seattle, WA, 2004: 23

[35] Xie T, Zhao J. A Framework and Tool Supports for Generating Test Inputs of AspectJ Programs // AOSD 2006. New York: ACM Press, 2006

[36] Xie T, et al. Detecting Redundant Unit Tests for AspectJ Programs// 17th IEEE International Conference on Software Reliability Engineering (ISSRE'06). Boston: IEEE Computer Society, 2006

[37] Xie T, et al. Automated Test Generation for AspectJ Programs // 1st Workshop on Testing Aspect-Oriented Programs, 4th International Conference on Aspect-Oriented Software Development. New York: ACM Press, 2005

[38] Katz S. A Survey of Verification and Static Analysis for Aspects. AOSD-Europe eu network of excellence: Israel, 2005