数据网格的动态读/写复制策略研究*)

张延松[1,2] 薛永生[1] 张 宇[1,2] 张剑达[3]

(厦门大学计算机科学系 厦门361005)[1] (哈尔滨金融高等专科学校 哈尔滨150040)[2]

(华侨大学计算机科学系 泉州362021)[3]

摘 要 探讨了在网格环境中数据复制的相关策略,包括动态只读复制策略和动态读/写复制策略。动态复制策略可以用来减少带宽消耗和访问延迟,不同的复制策略在不同的访问方式下有不同的性能表现。同时,提出了动态读/写复制策略的Fast Upload算法,算法支持数据传输过程中的并发复制,通过复制锁机制来维护复制与更新数据的一致性。

关键词 数据网格,动态复制策略,动态读/写复制策略,Fast upload

The Research on Dynamic Read/Write Replication Strategies of Data Grid

ZHANG Yan-Song[1,2] XUE Yong-Sheng[1] ZHANG Yu[1,2]

(Department of Computer Science, Xiamen University, Xiamen 361005)[1] (Harbin Financial Institute, Harbin 150040)[2]

(Department of Computer Science, HuaQiao University, Quanzhou 362021)[3]

Abstract In this paper, the replica strategies are discussed, including Dynamic read-only Replication Strategies and Dynamic read/write Replication Strategy. Dynamic Replication Strategies can reduce bandwidth consumption and access latency in accessing the huge amounts of data. The Dynamic read/write Replication Strategy named as "Fast Upload" is proposed. Fast Upload enables updating while data transporting, and the replica lock mechanism ensures the consistency of the data source and different replica.

Keywords Data grid, Dynamic replication strategies, Dynamic read/write replication strategies, Fast upload

计算机和网络技术的发展目标是为人们提供对资源的访问能力,作为第三代互联网技术代表的网格计算技术,其目标是将所有资源全面联通,最终实现网络虚拟环境下的资源共享和协同工作,消除信息孤岛和资源孤岛,形成网格(grid)或信息网格(information grid)。网格也是国家级高性能计算和信息服务的战略性基础设施[1],它不仅实现了对各种计算资源的访问,而且实现了对所有数据资源的统一访问。

在现代科学研究和应用领域中,数据量将达到几十TeraByte至PetaByte的级别。现有的数据管理体系结构、方法和技术已经不能满足地理上广泛分布的用户对高性能、大容量分布存储和分布处理能力的要求。因此,在计算网格的基础上人们提出了数据网格(Data Grid)的构想。在数据网格中对这些超级大型数据集的访问要占用大量网络带宽,同时也会造成很大的数据访问延迟,动态复制策略通过自动创建数据复制、自动分配和管理复制的机制来为数据网格中的用户提供就近访问策略,从而提供更高的数据访问性能。

1 数据网格的复制策略

网格有两个目标:数据共享和资源共享。GriPhyN项目将数据网格描述为异构的分层组织。产生数据的源被称为第一层,第二层是国家中心,第三层是区域中心,第四层,也是最高一层,由成百上千的客户端组成。

科学研究的数据量很大,形成了数据网格中的数据传输"瓶颈"。复制机制的目标是减少访问延迟和带宽消耗。通过对相同文件的多复制策略也可以起到负载平衡和提高数据可靠性的作用。复制管理与复制选择是数据网格的高层服务组件,它们提供注册、定位和管理多数据集的功能[2],包括:在新的位置创建可靠的大数据集的复制、通过信息服务的QoS评估选择性能最好的复制、根据应用的需要自动创建新的复制、删除过时的或访问率低的复制。

复制目录是数据网格系统的基础结构[3],它通过物理文件和逻辑文件的映射结构来跟踪文件的多个副本。

定义1(复制目录) RECA(LCollection, LLocation, LFile)为复制目录,其中:

LCollection表示逻辑集合:LCollection = { $LFile_1$, $LFile_2$, …, $LFile_i$, …}$i \in N$。集合中包含一组逻辑文件,集合为访问域或用户组提供了面向主题的数据访问机制。

LFile(property$_1$, property$_2$, …)表示逻辑文件:逻辑文件中包括相应的属性值,逻辑文件名遵循全局命名规则来保证在系统中的唯一性。

LLocation表示逻辑位置:

LLocation = {($LFile_1$, PhysicalLocation$_1$), …, ($LFile_1$, PhysicalLocation$_i$), …,

($LFile_2$, PhysicalLocation$_1$), …, ($LFile_2$,

*)本课题得到福建省自然基金资助(A0310008)和福建省高新技术研究开放计划重点项目资助(2003H043)。张延松 硕士研究生,讲师,研究方向为数据库技术、数据仓库、数据网格技术等。薛永生 教授,主要研究方向为数据库技术、数据仓库数据挖掘、网络技术等。张 宇 硕士研究生,讲师,主要研究方向为数据库技术、数据仓库、电子商务等。张剑达 硕士,助教,主要研究方向为数据库应用、网络技术等。

$PhysicalLocation_m$), …,

…,

($LFile_i$, $PhysicalLocation_1$), …, ($LFile_j$, $PhysicalLocation_n$), …,

…}$i, j, m, n \in N$, PhysicalLocation 为文件的物理位置。

位置对象包括逻辑文件名与复制的物理位置的映射，逻辑文件与物理位置的映射可以是一对多关系。

在复制目录上可以执行的操作有对逻辑集合、逻辑位置和逻辑文件的创建、删除；在逻辑集合和逻辑位置中插入或移除逻辑文件；监听逻辑集合和逻辑位置的内容；系统的核心功能是返回所有逻辑文件的物理位置。复制目录的功能形成了复制管理系统的基本功能，如创建、删除和管理复制。

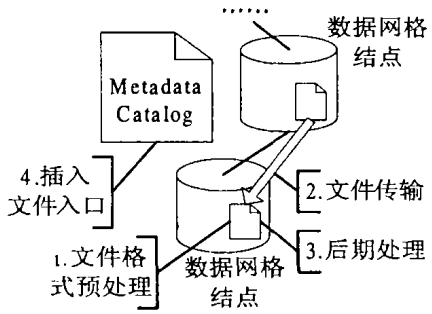

图1 复制创建步骤

将一个文件从一个存储位置复制到另一个存储位置需要以下几个步骤：

· 预处理过程：指定文件格式(有时可省略)，在目标位置数据库管理系统中创建新的 Schema。

· 文件传输过程：文件传输需要安全、高效、高速的传输机制，如 GridFTP 传输方式[4,5]。

· 后期处理：后期处理进行文件类型指定，将数据库文件与本地系统联连，并插入到内部文件目录中。

· 在复制目录中插入文件入口：包括对复制分配逻辑和物理名字，为复制指定数据网格的访问方式。

动态复制策略可以自动地根据用户访问的特点来创建、删除和管理文件复制。动态复制策略要解决三个问题：何时创建复制？哪一个文件应该创建复制？复制应该位于何处？

1.1 复制管理

复制本质上是对数据的缓存，为用户应用提供一个能够快速访问和处理远程数据的局部缓冲数据拷贝，避免大量数据远程传输到应用端。

定义2(复制管理系统) REPLICAMANAGER(DATASOURCE, RECA, USER, GRIDINFO, STRATEGY, REPLICA)为复制管理系统，它根据系统中相关对象的信息提供复制的创建、删除功能，并在复制目录中进行复制的注册、删除和复制选择功能。包括以下组成元素：

(1)DATASOURCE 为数据源，是数据网格系统中存储数据资源的实体。

(2)RECA 为复制目录对象。

(3)USER 为用户对象。

(4)GRIDINFO 为数据网格信息服务对象，提供数据网格的环境信息。

(5)STRATEGY 为复制策略。

(6)REPLICA 为复制对象。

定义3(复制创建) f_{create}(DATASOURCE, USER, GRIDINFO, STRATEGY, RECA, REPLICA)为复制管理系统根据数据源信息和网格服务信息响应用户的数据请求，按一定的复制策略创建数据集复制，并在复制目录中进行注册。

定义4(复制删除) f_{delete}(RECA, $REPLICA_x$)为复制删除。Access($REPLICA_i$)为复制 i 的访问量，Life($REPLICA_i$)为复制 i 的生命周期。

f_{delete}(RECA, REPLICAx), $REPLICA_x$ = Min(Access($REPLICA_1$), Access($REPLICA_2$), …);

if Access($REPLICA_i$) = Access($REPLICA_j$) = …

f_{delete}(RECA, $REPLICA_y$), $REPLICA_y$ = Max(Life($REPLICA_i$, $REPLICA_j$, …)。

文件替换策略首先考虑文件访问量，从文件列表将访问量最少的文件删除，文件的替换策略兼顾最少访问和文件生命周期，如果多个文件的有相当的低频度访问量，则删除最久的文件。

定义5(复制检索) f_{search}(DATASOURCE, USER, GRIDINFO, STRATEGY, RECA, REPLICA)为复制管理系统根据环境信息在复制目录中检索满足用户服务需求的复制。

数据复制之间的数据一致性和更新一直是分布式数据管理的难点，它与应用数据访问、产生、操作特性紧密相关。复制管理通过复制目录来实现对复制的组织和管理，复制目录可以采用集中式复制目录或分布式复制目录，以满足不同的性能要求和容错要求。数据网格包括多复制目录，面向不同的数据主题，也可能创建一个分级的复制目录来利用类似目录结构的相关逻辑集合。

1.2 复制选择和数据抽取

复制选择是数据网格提供的高层服务，是选择复制的过程，它提供优化的符合应用所需的性能，如绝对性能(速度)、代价或安全的数据访问。复制的选择可能会导致性能优于当前存在复制的新复制的创建。

定义6(复制选择) f_{select}(DATASOURCE, USER, GRIDINFO, STRATEGY, RECA, $REPLICA_{selected}$)为复制管理系统根据环境信息在复制目录中选择性能最好的复制响应用户的数据访问需求。

$REPLICA_{selected}$ = Max(NEWREPLICA(l_1, l_2, …), $REPLICA_1$(l_1, l_2, …), $REPLICA_2$(l_1, l_2, …), …), l_1, l_2, … 为复制选择所参考的环境信息，如网络性能、网络预留带宽、文件大小等参数；NEWREPLICA(l_1, l_2, …)为在当前环境指标下，生成新的复制的性能指标；Max()函数对应最好的复制性能指标。

复制选择服务选择一个能提供最佳访问性能的复制。同时，选择器可以评估创建一个新的复制是否会带来更好的性能。

复制选择功能需要具备调用能够了解文件结构并可以抽取所需要的数据子集的过滤和抽取程序。抽取出的子集具有自身的元数据和物理特征，作为一个文件提供给复制管理服务。

1.3 复制策略

数据复制提供更好的访问时间(本地数据访问)和容错性能。不同的复制策略在不同的用户数据访问模式下表现出不同的性能。文[6]中通过不同的文件的访问模式(最近文件访

问、邻近用户访问、邻近数据访问)对不同的复制策略进行性能测试。

主要的复制策略主要有:无复制或缓存(No Replication or Caching)、最佳客户端(Best Client)、层叠复制(Cascading Replication)、简单缓存(Plain Caching)、缓存 & 层叠复制(Caching plus Cascading Replication)、快速传播(Fast Spread)、T-value 复制策略。T-value 复制策略在文[7]中作了相关的性能分析,本文不作进行一步的比较分析。

文[8]中通过 P-随机测试、P1测试、P2测试三种方法对不同的复制策略进行测试。实验结果表明,如果网格用户的访问模式是完全随机模式,最好的复制策略是快速传播策略。如果有相当多的邻近用户访问模式,层叠策略作为一种复制策略优于其他策略,对存储空间的使用也较为合理。如果主要的目标是强调系统响应时间,层叠策略会更好一些,如果带宽消耗是首要考虑的,快速传播策略是更好的网格复制策略。

2 读写复制策略研究

目前的复制策略研究主要是面对只读数据复制,其重点放在复制重用、提高访问性能上。在这种策略下,数据的更新集中在服务器端进行,当服务器端的数据变化时,复制目录中的所有相关复制均变为无效复制,需要按用户的访问需求重新生成复制,增加了数据同步的代价。同时,由于数据量大,数据传输距离远,客户端的更新请求在服务器端完成需要较大的延迟和带宽占用,这些因素降低了数据网格的性能。本节讨论在数据网格环境中的动态读/写复制策略,以减小数据更新的代价,提高系统数据响应能力。

2.1 动态读/写复制策略更新机制原理与过程

我们提出动态读/写复制策略更新机制:复制管理策略相当于自底向上的 Fast Spread 策略,与客户端最近的读/写复制接受用户的数据更新,并沿着从客户端到服务器的路径逐级向上传输,当路径上存在相同数据源的复制时,在传输的同时完成该复制的更新,直到完成服务器端的数据更新为止。复制更新机制称为 Fast Upload。读/写复制更新机制分析如下:

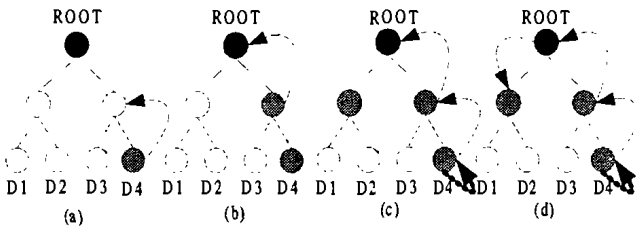

图2 Fast Upload 数据更新原理

当复制目录中没有相应数据的复制时(如图2 a 中所示),数据在客户端被更新,并沿目录的层次结构逐层向上传递,并在传递的同时在该节点生成复制(如图2 b 中所示),直到数据源所在的 ROOT 节点,完成数据更新操作。设客户端完成向本地复制的数据更新操作所需要的时间为 t,数据沿路径的传输时间为 T,数据更新的时间为 u,则在数据网格中完成由客户端向服务器端的数据更新需要的时间为 $t+T+u$。

当沿服务器到客户端的路径上和其他路径节点上存在着复制时(如图2 c 中所示),客户端完成向本地复制的数据更新操作,然后客户端可以进行其他的数据处理工作,由客户端更新的复制在系统后台完成逐级向上的更新操作。对客户端而言,数据更新需要的时间是 t,对系统而言,完成数据更新的时间为 $t+T+u$,客户端的数据更新时间被大大缩短。

在这种复制更新机制中,客户端数据更新沿着从客户端到服务器端的路径进行更新与传输,可以完成路径上的复制的同步更新,但对于不在当前路径上的同源复制来说,更新是不能够同步的,我们可以采取如下措施:

简单抛弃法:

```
(1)WHILE (Current node<>ROOT)
   //当复制更新未到达根结点
(2)  update(Current node);
     //更新当前结点上相应的复制
(3)  transfer(parent node);
     //向父结点进行数据传输
(4)  node=Up(node);
(5)END WHILE
(6)FOR EACH(REPLICA_i in RECA)
(7)IF (REPLICAi is old) f_delete(RECA, REPLICA_i);//将复制目录
   中不在更新路径上的未更新的复制删除。
```

在这种策略中,不在更新路径上的复制被简单地从复制目录中被删除,在后续的用户访问中按照复制的产生策略再重新动态生成新的复制,并分配到相应位置。非路径节点的更新代价为零,但需付出额外的复制生成代价。

逐层向上传播法(3.2算法):

在这种更新策略中,数据在沿着路径逐层向上传递时,根据复制目录中的复制元数据信息,确认该节点的其他下层节点的复制位置,如无,继续向上传输,如有其他的下层节点,则在向上层节点进行数据传输的同时,向该下层节点也进行数据传输,完成该下层节点的复制更新,再由该下层节点自动完成其向下的传播路径中相关复制的同步更新(如图2 d 中所示)。在这种复制管理策略中,利用数据更新与传输的并行过程,可以缩短数据处理时间。

2.2 Fast Upload 算法设计

Fast Upload 算法实现复制的动态读/写复制更新。

CurrentReplica(x)表示当前节点 x 上的复制;

Up(x)表示节点 x 的父节点;

Down(x)表示节点 x 的子节点;

GridFTP(x,y,t)表示通过 GridFTP 工具由 x 复制向 y 复制进行更新,并加上时间戳 t;

Algorithm Fast_Upload:

输入:客户端节点 c

输出:各级节点上的复制

```
(1)Timestamp tag= Timestamp (c);
   //由客户端生成时间戳;
(2)updateclient(c, CurrentReplica (c));
   //客户端 c 向当前节点上的读/写复制进行数据更新
(3)Client upnode=Up(c);
   //将客户端的父节点保存在节点 upnode 中;
(4)GridFTP(CurrentReplica (c), CurrentReplica (upnode),tag);
   //从客户端节点 c 的复制向它的父节点 node 的复制进行更新;
(5)WHILE ((upnode<>ROOT)AND ( not Exist CurrentReplica
   (Down(ROOT)))
   //未到达根节点并且无其他子节点复制
(6)     Client downnode=upload;
        // downnode 用来跟踪下层节点;
(7)     DO
(8)     IF Exist CurrentReplica (Down(downnode))
        //如果当前节点的子节点存在同源复制
(9)        GridFTP (CurrentReplica (downnode), CurrentReplica
           (Down(downnode)),tag);
           //由当前节点的复制向其子节点的复制进行更新;
(10)       downnode= Down(downnode);
           //当前节点变为其下层节点;
(11)    END IF
(12)    UNTIL (downnode==null);
   //直到叶节点才停止向下层节点的复制更新;
(13)GridFTP(CurrentReplica (upnode), CurrentReplica (Up(upn-
   ode)),tag);
   //向上层节点进行复制更新;此向上更新过程可与 Do…until 过
   程并行操作,提高系统执行效率
(14)upnode=Up(c);
(15)END WHILE
```

GridFTP(x,y,tag)用来在两个复制之间进行更新,由于

是沿着客户端到服务器端的路径进行向上传输和更新，因此可以在传输文件的过程中进行并行的复制更新。

3 性能分析和相关问题

3.1 读/写复制策略的性能分析

若使用在服务器端进行数据更新、Fast Spread 策略进行复制更新的方式，数据更新包括以下三个过程：

(1)由客户端进行数据更新，数据沿路径向上传输；更新的时间 $t+T$；

(2)更新的数据到达根节点后在服务器端进行数据更新，时间为 u；

(3)由服务器端根据更新数据的内容将复制目录中相关的复制进行更新，时间为 U；

整个数据更新的时间为：$t+T+u+U$。

而使用 Fast Upload 方法从客户端向服务器端更新包括以下两个步骤：

(1)由客户端进行数据更新，更新数据沿路径向上传输，同时完成沿路径相关复制的更新，更新的时间为客户端数据更新的时间 t+数据传输时间 T+复制更新超出数据传输的时间 ε；

(2)更新的数据到达根节点后在服务器端进行数据更新，时间为 u；

整个数据更新的时间为：$t+T+u+\varepsilon$ ($\varepsilon<<U$)。

从更新过程的比较与分析可以看到，使用 Fast Upload 方法，数据在上传的同时完成了相关复制的更新，省去了自上而下的复制更新过程，数据更新的速度大大提高。

3.2 读/写复制策略的一致性问题

从客户端向服务器端的数据更新过程包括数据在网络中的传输过程与数据在服务器端的更新过程。在 Fast Upload 方法中，数据在传输的同时沿传输路径进行复制更新，这会造成不同复制之间的数据不一致及数据源与复制之间的数据不一致。

在远程数据更新的数据传输延迟阶段，客户端、复制、服务器端的数据不一致难以避免，我们通过在数据传输阶段使复制目录中的相关复制暂时失效的方法来统一管理分布的复制的一致性。我们通过复制锁机制来解决复制不一致问题：

(1)客户端进行数据更新时，将数据更新到本地复制中，并按 Fast Upload 方法向上传播；

(2)在复制目录中将更新数据对应的所有复制锁定，并标记为无效，不接受用户的数据访问，并且不再响应同类复制的创建；

(3)按 Fast Upload 方法逐层传输数据并更新相应的复制；

(4)当数据到达服务器所在的根节点并完成数据更新时，复制目录中标记为无效的复制重新标记为有效，接受数据访问和复制管理。

3.3 其他相关问题

数据更新的有效性问题。在服务器端进行数据更新时，可以进行数据完整性与一致性验证，并可利用事务机制来提高数据更新的可靠性。而 Fast Upload 算法是自底向上的数据更新策略，很难起到服务器端的数据有效性验证作用。

数据安全与权限问题。数据的更新需要一定的安全权限，复制目录中可以将复制和用户访问权限以元数据的方式结合在一起，在用户访问时进行安全性检查。读/写复制的安全管理增加了复制管理的复杂性。

系统的复制和更新机制与很多因素相关，考虑到科学数据的访问特性以及数据查询与更新的比例，需要在只读复制机制与读/写复制机制之间进行效果与代价的折衷权衡，选取适合的复制策略，也可以使只读复制机制与读/写复制机制共存，处理不同类型的数据访问。

结束语 复制策略是数据网格中提高数据访问效率的有效方案，但复制策略也固有其算法的复杂性和代价评估的复杂性，需要一个有效的评估机制来控制复制策略的实施。只读复制策略适用于大多数的科学数据，为系统提供了简单高效的复制策略。而动态的读/写复制策略提供了高效的数据更新机制，可以在网格系统中完成对数据的快速更新，并自动完成相关复制的同步更新。由于网格环境本身固有的复杂性，读/写复制策略要与网格其他相关管理策略结合起来形成有效的复制管理机制。

参 考 文 献

1 王意洁，卢锡城. 网格中的信息计算技术[J]. 计算机科学，2003，30(2)：38～39

2 Chervenak A，Foster I，Kesselman C，et al. The Data Grid：Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets[J]. Journal of Network and Computer Applications，2001，23：187～200

3 Stockinger H，Samar A，Allcock B，et al. Holtman3，Brian Tierney. File and Object Replication in Data Grids[J]. Journal of Cluster Computing，2002，5(3)：305～314

4 Allcock W，Bester J，et al. GridFTP Protocol Specification[EB/OL]. http://www.globus.org/research/ . GGF GridFTP Working Group Document，Sep. 2002

5 Allcock B，et al. Data Management and Transfer in High Performance Computational Grid Environments[J]. Nefedova，D. Quesnal，S. Tuecke. Parallel Computing Journal，2002，28 (5)：749～771

6 Ranganathan K，Foster I. Identifying Dynamic Replication Strategies for a High-Performance Data Grid[J]. In：Intl. Conf. on Computing in High Energy and Nuclear Physics，Beijing，Sep. 2001

7 庞丽萍，陈勇. 网格环境下数据副本创建策略[J]. http://www.chinagrid.net. 2004-1

8 Ranganathan K，Foster I. Design and Evaluation of Dynamic Replication Strategies for a High-Performance Data Grid[J]. In：Intl. Conf. on Computing in High Energy and Nuclear Physics，Beijing，2001