文章编号:1001 - 2486(2006)05 - 0042 - 05

基于镜像的高可用数据对象布局算法*

刘 仲,任 浩,周兴铭
(国防科技大学 计算机学院,湖南 长沙 410073)

摘 要:镜像是提高数据可用性的流行方法。借鉴 RAID 的方法,在算法一级实现数据的冗余分布,提出基于镜像的高可用数据对象布局算法。在数据对象和存储节点失效时,利用冗余数据重构数据对象和存储节点,有效保证存储系统的高可用性。采用马尔可夫激励模型对存储系统进行定量的可用性分析,计算结果表明该方法是有效的。

关键词:大规模存储系统;高可用性;数据对象布局
中图分类号:TP393 **文献标识码**:A

High Availability Data Objects Placement Algorithm with Mirroring

LIU Zhong, REN Hao, ZHOU Xing-ming
(College of Computer, National Univ. of Defense Technology, Changsha 410073, China)

Abstract: Mirroring is a popular technique for enhancing data availability. Based on this technique, this paper proposes a high-availability data objects placement algorithm with mirroring, which groups objects into redundancy sets by using RAID at the algorithm level. The redundancy allows us to reconstruct any corruption or failure of the data objects and storage nodes, thus efficiently ensuring the high availability of storage system. We quantify the availability of storage system by using Markov reward model, and the computing results indicate that the algorithm is efficient.

Key words: large-scale storage system; high availability; data objects placement

高可用性是大规模集群存储系统非常关键的技术指标之一。随着存储节点数量的增加,在增加存储系统容量和性能的同时,系统出现存储节点失效的概率也同步增加,并且由于磁盘容量的增加,导致磁盘失败时重建磁盘数据的时间更长,简单地使用 RAID 技术不能有效保证数据的可用性[1]。本文在基于动态区间映射的数据对象布局算法(dynamic interval mapping, DIM)[2-3]基础上,结合 RAID 的数据冗余思想,提出基于镜像的高可用变体算法。

1 DIM 算法介绍

在基于对象存储的大规模集群存储系统中,客户端通过对象 ID 访问数据对象。对任意的数据对象 x,客户端通过数据对象映射函数 $f(x)$确定数据对象所分布的存储节点地址。数据对象映射函数 f 是动态变化的,它随着映射表示 ρ 的改变而变化,而映射表示 ρ 随着存储系统的规模变化而变化。在基于动态区间映射的数据对象布局算法 F 中,数据对象映射 f 可以定义成函数 $F(x,\rho)$,即 $f(x)=F(x,\rho)$,其中 x 为数据对象集合 X 中的任意元素,ρ 是特定时期的映射表示。数据对象布局算法 $F(x,\rho)$根据输入的数据对象 x 和特定的映射表示ρ,返回相应的存储节点。对同一个数据对象 x,在不同的映射表示 ρ 下计算得到的存储节点 $F(x,\rho)$可能是不同的。不同时期的映射表示依次用 ρ_0, ρ_1, …, ρ_i 表示,下标 i 为映射表示版本号,允许客户端计算节点时使用的映射表示版本与存储节点当前使用的映射表示版本不同,客户端将通过视图校正算法自主学习,自动适应动态变化的存储系统规模。

* **收稿日期**:2006 - 06 - 01
基金项目:国家自然科学基金资助项目(60503042)
作者简介:刘仲(1971—),男,副教授,博士。

2　基于镜像的高可用数据对象布局算法

2.1　基本原理

镜像是实现数据高可用的重要方法。实现镜像的方法可分为两种,一种方法是传统的磁盘镜像[4],用设备虚拟化的形式使两个磁盘看起来就像一个磁盘,接受完全相同的数据,当一个磁盘驱动器失败时,系统仍然能够保证数据的可访问能力。该方法的好处是,它对数据对象布局算法是透明的,不需要在算法上作任何的修改。缺点是,价格昂贵,每个存储节点需要专门的硬件实现磁盘镜像,并且只适用于磁盘驱动器失效的情况,在由其他组件导致的存储节点失效时不能保证数据的可用性,比如因电源、CPU 等失效而导致的存储节点失效。另一种方法是在数据的管理中引入镜像机制,即在数据对象的布局中,数据对象及副本根据计算分布到两个不同的存储节点中,当存储其中一个数据对象的存储节点失效时,系统从保存副本的存储节点中获得数据,保证数据的可用性。这种方法的好处是,数据对象所分布的存储节点可以是两个异构的独立节点,存储节点可以是普通商用的 PC 机、工作站或 OSD,不需要昂贵的专门硬件实现镜像,不仅能够在磁盘驱动器失效时保证数据的可用性,而且在电源、CPU 等其他部件引起存储节点失效时可保证数据的可用性,甚至两个存储节点可分布到异地,在地震等自然灾害下依然保证数据的可用性。缺点是,需要专门的数据对象布局算法实现数据对象及其冗余数据的均衡分布、定位和重构。

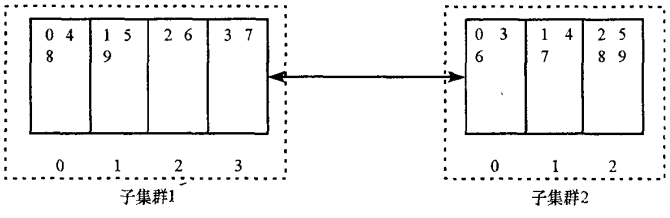

图 1　互为镜像的两个子集群

Fig.1　Two sub-cluster mirroring

基于镜像的数据对象布局算法的基本原理如图 1 所示,将所有节点分成两个独立的子集群,两个子集群互为镜像,任意的数据对象被更新到其中一个子集群中时,立即被更新到另一个子集群中。每个子集群独立地自主管理本子集群的数据对象布局,子集群 1 中的存储节点访问子集群 2 中的存储节点时,子集群 1 中的存储节点是作为子集群 2 中的客户端访问的,因此,子集群 1 中的存储节点除了本地的存储节点端映射表示视图外,作为客户端,同时保存有子集群 2 的本地客户端映射表示视图,并且在访问子集群 2 中的存储节点时通过视图校正算法更新子集群 2 的本地客户端映射表示视图。子集群 2 中的存储节点访问子集群 1 中的存储节点也是如此。

在正常情况下访问时,互为镜像的两个子集群服务的客户端集合是不同的,根据访问的客户端分为主从子集群。所有客户端事先根据相关规则(如网络距离、负载均衡等)分为两个不同集合,分别用 $C1$、和 $C2$ 表示,$C1 \cap C2 = \Phi$。其中子集群 1 称为 $C1$ 的主子集群,子集群 2 称为 $C1$ 的从子集群,与之对应的是,$C1$ 称为子集群 1 的主客户端集合,同时称为子集群 2 的从客户端集合。

2.2　客户端地址计算

客户端对每一个子集群都有本地的映射表示视图,两个映射表示视图可能是不同的,并且在访问不同的子集群中通过视图校正算法更新相应的本地映射表示视图。在正常访问情况下,客户端使用主子集群的本地映射表示视图,根据(1)式计算数据对象所分布的主子集群中的存储节点地址,将数据对象请求发送给该存储节点。

$$s' = F(x, \rho'_i) \tag{1}$$

若存储节点失效,则将存储节点失效的消息报告管理节点,并启动对从子集群的访问请求,即切换

到使用从子集群的本地映射表示视图，根据(1)式计算数据对象所分布的从子集群中的存储节点地址，将数据对象请求发送给该存储节点，若返回有视图校正信息，则根据(2)式的视图校正算法更新从子集群的本地映射表示视图。

$$\begin{cases} i' = i \\ \rho'_i = \rho_i \end{cases} \tag{2}$$

2.3 存储节点端地址计算

当接收到客户端的数据对象访问请求时，若是只读的数据对象访问请求，则根据(3)式计算服务该客户端数据对象请求的实际存储节点地址。

```
if  (i' ≠ i) then {                                    (3)
    s = F(x, ρ_i)
    if  (s 不是本节点) then
        将客户端的数据对象请求转发给存储节点 s
    else {
        接收并响应客户端请求
        将处理结果返回给客户端
        同时将新的映射表示 ρ_i 返回客户端
    }
}
else {
    接收并响应客户端请求
    将处理结果返回给客户端
}
```

当接收到存储节点端转发的客户端数据对象请求时，存储节点端根据(4)式响应该客户端的数据对象请求。

```
{                                                      (4)
    从转发过来的客户端数据对象请求中解析出客户端地址以及数据对象请求
    接收并响应客户端请求
    将处理结果直接返回给客户端
    同时将新的映射表示 ρ_i 直接返回客户端
}
```

若是需要更新数据对象的访问请求，该存储节点在完成客户端的访问请求以后，必须将数据对象的更新同步到镜像的子集群相应的存储节点中。因此，在子集群的存储节点中，除了本地的存储节点端映射表示视图外，作为客户端，同时保存有镜像子集群的本地客户端映射表示视图，并且在访问镜像子集群中的存储节点时通过视图校正算法更新镜像子集群的本地客户端映射表示视图。存储节点将数据对象同步到镜像子集群时，它作为镜像子集群的客户端，使用镜像子集群的本地客户端映射表示视图，根据(1)式计算数据对象所分布的镜像子集群中的存储节点地址，将数据对象同步请求发送给该存储节点。若返回有视图校正信息，则根据(2)式的视图校正算法更新镜像子集群的本地客户端映射表示视图。

当碰到数据对象不能读取的错误时，使用镜像子集群的本地客户端映射表示视图，根据(1)式计算数据对象所分布的镜像子集群中的存储节点地址，请求恢复该数据对象，将数据对象更新存储后返回响应请求给客户端。

2.4 数据恢复方法

2.4.1 存储节点恢复

假定失效的是子集群 1 中的存储节点为 s_i,空闲存储节点为 s,子集群 1 中的存储节点映射表示为 ρ,子集群 2 中的存储节点映射表示为 ρ'。依照如下步骤恢复存储节点 s_i 中的数据:

(1) 对子集群 2 中的所有存储节点,检查子集群 1 的本地客户端映射表示视图是否与 ρ 一致,若不一致,则更新本地客户端映射表示;

(2) 对子集群 2 中的所有存储节点,扫描本存储节点的所有数据对象,查找满足条件:$s_i = F(x,\rho)$ 的数据对象 x;

(3) 将满足条件的所有数据对象 x 发送到空闲存储节点 s;

(4) 空闲存储节点 s 编号替换为 s_i,恢复该存储节点的最新映射表示 ρ 以及其他配置信息。

2.4.2 数据对象恢复

假定访问数据对象 x,失效的数据存储节点为 s_i,依照如下步骤恢复数据对象:

(1) 若存储节点端碰到数据对象不能读取的错误时,使用镜像子集群的本地客户端映射表示视图,根据(1)式计算数据对象所分布的镜像子集群中的存储节点地址,请求恢复该数据对象,将数据对象更新存储后返回响应请求给客户端。

(2) 若客户端在访问满足条件 $s_i = F(x,\rho)$ 的数据对象 x 时,则切换使用从子集群的本地映射表示视图,根据(1)式计算数据对象所分布的从子集群中的存储节点地址,将数据对象请求发送给该存储节点,若返回有视图校正信息,则根据(2)式的视图校正算法更新从子集群的本地映射表示视图;

(3) 若存储节点端接收到客户端的数据访问请求,根据(3)式计算服务该客户端数据对象请求的实际存储节点地址满足条件 $s_i = F(x,\rho)$,则使用镜像子集群的本地客户端映射表示视图,根据(1)式计算数据对象所分布的镜像子集群中的存储节点地址,将数据对象访问转发给该存储节点,由该存储节点处理该数据对象访问请求。

2.5 高可用性分析

对于系统的可靠性和可用性分析,一般采用马尔可夫激励(Markov reward model, MRM)模型[5-6]。MRM 模型把目标系统的状态 S 分为两类:

$$S = A \cup N$$

其中,A 表示不可靠状态集合,N 表示可靠状态集合。

若只考虑可靠状态集合 N,可以通过公式(5)和(6)计算出系统到达不可靠状态前的平均时间。

$$\boldsymbol{L}_N(\infty)\boldsymbol{Q}_N = -\boldsymbol{\pi}_N(0) \tag{5}$$

$$MTTDL = \sum_{i\in N} L_i(\infty) \tag{6}$$

其中,$\boldsymbol{L}_N(\infty)$为到达不可靠状态前的平均时间向量,$\boldsymbol{Q}_N$ 为状态转移概率矩阵,$\boldsymbol{\pi}_N(0)$为初始时的状态概率向量,MTTDL 为系统的平均无数据丢失时间(mean time to data loss)。

冗余集包括同一数据对象的两个副本,被分布到不同的两个存储节点中。定义目标系统冗余集的系统状态如下:

- 状态 0:两个数据对象均可用,数据对象可用;
- 状态 1:其中一个数据对象失效,数据对象可用;
- 状态 2:两个数据对象同时失效,数据对象不可用。

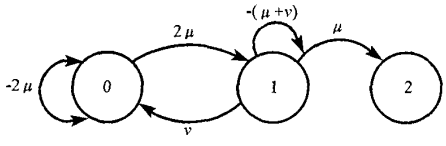

图 2　冗余集状态转移模型

Fig.2　State transition model of redundant set

图 2 显示冗余集的状态转移模型,其中 μ 和 ν 分别表示冗余集的失效率和修复率。系统的可靠状

态集合包括状态0和1;不可靠状态集合包括状态2。

可靠状态集合 N 的状态转移概率矩阵和初始时的状态概率向量分别为:

$$Q_N = \begin{pmatrix} -2\mu & 2\mu \\ \nu & -(\mu+\nu) \end{pmatrix}, \quad \pi_N(0) = (1 \quad 0)$$

由公式(5)、(6)求得:

$$MTTDL = \frac{3\mu+\nu}{2\mu^2}$$

假定存储系统中的参数定义见表1,在冗余集中,有:

$$\mu = \frac{1}{MTTF_{\text{disk}}}, \quad \nu = \frac{\gamma}{R}$$

由于 $\nu \gg \mu$,所以有:

$$MTTDL_{\text{冗余集}} = \frac{3\mu+\nu}{2\mu^2} \approx \frac{\nu}{2\mu^2} = \frac{\gamma \cdot MTTF_{\text{disk}}^2}{2R}$$

存储系统包含的冗余集合数量为 $S = \frac{T}{R}$,并且 $\frac{1}{MTTDL_{\text{冗余集}}}$ 很小,简化计算后得到:

$$MTTDL_{\text{存储系统}} = \frac{1}{1-\left(1-\frac{1}{MTTDL_{\text{冗余集}}}\right)^S} \approx \frac{MTTDL_{\text{冗余集}}}{S} = \frac{\gamma \cdot MTTF_{\text{disk}}^2}{2T}$$

按照表1的参数计算,基于镜像方法的存储系统的平均无数据丢失时间大约为57年。

表1 存储系统参数

Tab.1 Parameters of storage system

参数	意义	取值
T	存储系统总的存储数据容量	1PB
R	单个冗余集合的存储数据容量	可变
S	存储系统包含的冗余集合数量	T/R
$MTTF_{\text{disk}}$	磁盘的平均无故障时间	10^5 h
γ	故障修复率	100GB/h

3 小结

在DIM算法基础上,结合RAID的数据冗余思想提出基于镜像的高可用数据对象布局算法,通过容错的数据对象布局算法实现数据的冗余分布,在数据对象或存储节点失效时,利用冗余数据重构数据对象,保证存储系统的高可用性;存储节点失效时,重建失效磁盘的数据能够通过并行分布的方式实现,提高了数据重建的效率,降低了重建期间同时发生另一个磁盘失效的概率。采用马尔可夫激励模型进行定量的可用性分析,计算结果表明该方法能够有效保证存储系统的高可用性。

参考文献:

[1] Xin Q, Miller E L, Long D D E, et al. Reliability Mechanisms for Very Large Storage Systems[C]. Moore R, eds. Proceedings of the 20th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies. Washington: IEEE Computer Society, 2003:146-156.

[2] 刘仲,周兴铭.基于动态区间映射的数据对象布局算法[J].软件学报,2005,16(11):1886-1893.

[3] Liu Z, Zhou X M. An Adaptive Data Objects Placement Algorithm for Non-uniform Capacities[C]. The 3rd International Conference on Grid and Cooperative Computing, Wuhan, Lecture Notes in Computer Science, 2004, 3251:423-430.

[4] Holland M C. On-line Data Reconstruction in Redundant Disk Arrays[D]. Carnegie Mellon University, 1994.

[5] Bolch G, Greiner S, Meer H, et al. Queueing Networks and Markov Chains[M]. John Wiley & Sons, Inc., 1998.

[6] 柳新民,邱静,刘冠军.BIT系统的三态马尔可夫模型分析[J].国防科技大学学报,2004,26(1):84-88.