对象文件系统中元数据服务器集群的负载均衡研究

邵强 刘仲 窦勇

（国防科学技术大学计算机学院 长沙 410073）

（E-mail：byriver@tom.com tel:0731-4573664）

【摘 要】 集群系统是大型计算机系统发展的一种趋势，对象文件系统是其管理文件的一种有效方式。大型应用中，采用MDS集群管理元数据，本文针对MDS集群的特点提出了一种负载均衡方法，其采用hash表分配负载来分布管理集群负载分配。

【关键词】 集群系统；对象集群文件系统；负载均衡；hash表；负载分配

Research on Load Balancing of MDS Cluster in Objected File System

Abstract Cluster system is a trend of large-scale computer system development, and Objected File System is an efficient way to manage files. In large applications, meta-data are managed by MDS Cluster. In this paper, a method is put forward to manage the load balancing of MDS cluster in Objected File System. Aim at the characteristics of MDS cluster, this method adopt a hash table to assign the loads, and it is a method of load balance which assign the loads of the cluster in a distributed way.

Key words cluster system; Object-cluster file system; load balancing; hash table; load assignment

1 引言

随着计算机性能需求的提高，集群系统以其性价比优势逐渐成为一种大型系统发展的趋势[4]。对象集群文件系统以对象方式管理文件，将文件数据分为元数据和实际数据两部分，分别由不同的子系统（元数据服务器MDS和对象存储目标OST）[1][3]存储和管理，是集群系统管理文件的一种有效方式，具有高性能、可扩展、高可靠性的特点。

元数据作为“关于数据的数据”[1]，是文件系统的中心信息，正确、快速读取元数据是文件系统正常运转的保证，所以元数据的管理对于整个文件系统的可靠性、高性能等具有决定性作用。大型应用中，集群文件系统的元数据服务器是由多个计算机组成的计算机集群，其中的元数据管理具有更大的复杂性。MDS集群可以采用具有独立存储的分布式系统，集群要高效管理元数据，集群中的各MDS的负载均衡是一个很重要的方面。如果MDS集群中各服务器的负载不均衡，某些MDS空闲而某些MDS负载过重，就会导致整个MDS集群的服务性能降低，甚至有可能瘫痪。本文针对MDS集群的特点提出了一种负载均衡方案，实现MDS集群的负载均衡，并具有非常好的可扩展性。

2 MDS集群负载均衡评判指标

MDS集群的负载均衡方案的设计，主要考虑以下几方面的问题[5]：

- 性能：集群的服务负载按各MDS的计算机性能均衡地分配到各MDS上，减少client

的等待时间，提高服务速度。

- 可扩展性：文件系统是不断膨胀的，MDS 集群必须能够扩展。随着总服务负载的增加，需要加入新的 MDS，集群能够自动负载均衡，将部分服务负载分配到新加入的 MDS 上。
- 灵活性：负载均衡方案应能灵活地提供不同的应用需求，满足应用需求的不断变化。集群内部的调整要尽量不影响对外服务。负载均衡过程中需要进行迁移数据，迁移数据量要小，时间尽量短。
- 可靠性：负载均衡方案应能为服务器群提供完全的容错性和高可用性，不能因为某个负载均衡设备失效而导致整个负载均衡方案的设备。
- 易管理性：有灵活、直观和安全的管理方式，便于安装、配置、维护和监控，提高工作效率，避免差错。

3 负载均衡方案的设计

3.1 方案设计思想

目前有许多不同的负载均衡技术用以满足不同的应用需求：从负载均衡所采用的设备对象看包括软、硬件负载均衡；从负载均衡所处的网络层次看，负载均衡策略可以运用于网络结构的各个层次[5]。各种型负载均衡方法通常混合使用，例如，典型的集群解决方案 Linux 服务器集群系统——LVS（Linux Virtual Server）中采用 IP 负载均衡技术和基于内容请求分发技术[2]。

上面介绍的负载均衡技术，不管是软件或者硬件实现，多数是使用集中管理的方式将负载分发到各服务器，集中管理可能形成“瓶颈”或因为集中管理设备失效将引起整个集群性能降低或失效。在这里我们根据元数据访问的特点，提出由各服务器分布管理负载分发的负载均衡方法。

由于文件的随机性，每个文件被访问的概率是相同的，具有相同数量文件的文件组被访问的概率是相同的，由于元数据是文件相同的抽象，大小相同，MDS 的服务负载可以转化为服务器上存储的数据量来表示。所以，把文件系统中的元数据按照数量均匀分配到各 MDS 上，client 对文件的访问就会随机均匀的分配到各服务器上，实现负载均衡的目的。

在对象文件系统中，以 OID 号唯一标识文件对象[1]，文件系统通过 OID 号访问和操作文件。OID 是一个二进制的 64 位正整数，每个文件分配一个全局统一的 OID。系统中 OID 随着文件系统的增大而增加，创建一个新的文件，将当前最大的 OID 加 1 分配给新的文件。对文件的 OID 号执行 hash 算法，将文件元数据均衡、分布存储于各 MDS，访问和操作文件数据时以 OID 号通过 hash 算法确定元数据存储的 MDS，client 可以正确连接存储元数据的 MDS 访问和操作元数据。

3.2 通过 hash 表分配负载

在实际应用过程中，随着计算机硬件技术的发展，并为了对已有计算机资源的充分利用，服务器集群中的计算机往往不一定是同构的，在考虑负载均衡时就会复杂些。考虑各服务器性能时，应当综合考虑 cpu 速度、内存大小以及硬盘空间大小等方面的信息得出结论。由于计算机的配置基本上是均衡的，计算机的各部件的配合不会出现很大的“浪费”或“瓶颈”现象，为了简化起见，在这里以硬盘存储空间作为考虑 MDS 性能的依据，数据量的分配的权值由硬盘存储空间作为参数计算得出。

假设有一组服务器 $S=\{S0, S1, \ldots, Sn-1\}$，$M(Si)$ 表示服务器的磁盘空间的大小，$W(Si)$表示服务器 Si 的权值。权值计算方式如下：以一定的磁盘大小为基数，假定为 base，如 5G，各服务器上元数据分配量的权值计算 $W(Si)=M(Si)/base$。为简化计算，可以以集群中最小的磁盘空间大小 Mmin 为基数，$W(Si)=M(Si)/Mmin$，那么最小的服务器权值为 1。当然，根据实际情况也可以采用另外的权值计算方法。

建立一个有 256 个桶（Bucket）的 Hash 表，一般来说服务器的数目会小于 256，当然表的大小也是可以调整的。算法的初始化是每个服务器连续占据 W(Si)个节点，将所有服务器顺序、循环地放置到 hash 表中，直到占满 hash 表的所有位置，服务器所在的 hash 表位置登记服务器的情况，包

服务器所在的 hash 表位置登记服务器的情况，包括编号，地址等。所有服务器的权值之和为 ΣW(Si)，那么 hash 表中服务器分配循环的遍数 num 为 256/ΣW(Si)，服务器 Si 占据的位置数为 W(Si)*num。数据分配时，对 OID 号运用 hash 算法按相同的概率将 OID 号分配到 hash 表的某个位置，则 OID 号的元数据存储于占据该 hash 表位置的服务器上。由于 OID 号是连续的正整数，可以采用 OID 号对 256 取模，通过取模的值读取 hash 表的相应位置所登记的信息可以确定数据应分配的服务器。每个 hash 表位置分配 OID 号的概率是相同的为 1/256，那么服务器 Si 存储某个数据的概率为 W(Si)*num*1/256＝W(Si)/ΣW(Si)，这样就实现了按权值在各服务器上分配数据的目的，即是按照权值将元数据集群的服务负载均衡分配到各服务器上。

```
//hash 表中节点定义
struct ServerNode
{    int number;                    //节点编号
     int flag;                      //数据迁移是否完成标志
     unsigned int address[32];   //登记的新服务器地址
     unsigned int old_addr[32];  //登记的老服务器地址
}
const int hash_len=256;
struct ServerNode hash[hash_len];
hash 表初始化算法：
输入：各服务器权值
输出：无
{
    if(0 号服务器)
    {    计算所有权值之和 sum_w;
         服务器顺序、循环登记到 hash 表中;    }
    广播;
}
按 OID 定位元数据存储 MDS 算法：
输入：oid
输出：MDS 信息
{
    n = oid% hash_len;
    return hash[n];
}
```

系统扩展时，新服务器加入系统，以新集群中所有服务器的权值之和计算服务器在 hash 表中的遍数，将 hash 表后端原有服务器多余的遍数去掉，用新服务器占据 hash 表的这些位置，并将新的服务器登记到 hash 表中。原有服务器占据 hash 表相应位置所分配的数据迁移到新服务器中，并且，扩展后的集群按照新登记的 hash 表服务器将新数据分配到占据该位置的服务器上。再进行扩展时，仍然是重新计算遍数，前面的服务器所占位置往前收缩，后面的服务器所占位置往后收缩，将新服务器插入 hash 表中。服务器在 hash 表中分布示意图如图 1 所示。通过这种方式进行扩展，可以保证原有的服务器扩展前后分配的数据变化不大，仅需要把分配给新服务器的数据迁移到新服务器上即可，减小系统扩展时系统内部调整的代价。

```
系统扩展 hash 表调整算法：
输入：各服务器权值
输出：无
{
    m0：系统中原有服务器最大编号
    计算所有权值之和 sum_w,所有扩展服务器权值和 sum_w1；
    计算遍数 num＝hash_len/sum_w;
    从表尾连续顺序登记扩展服务器进 hash 表；
}
算法 enlarge                //MDS
输入：无
输出：无
{    if(新加入 MDS )
     {    发信息给系统 0 号服务器，请求加入系统;
          wait();
          收 0 号服务器消息(权值计算 base，各 MDS 权值，hash 表内容)；
          计算自身权值，更新 hash 表;
          广播 hash 表;    }
          if(是 0 号 MDS&&收到新服务器加入消息)
               发送消息到新 MDS(权值计算 base，各 MDS 权值，hash 表内容)；
     if(非新加入和 0 号 MDS)
     {    接受消息，更新 hash 表，权值表；}
```

```
    数据迁移 immigrate();
}
算法 immigrate
输入：hash 表
输出：无
{for(i=m-1;i>=0;i--)        //m 为总的服务器数量
{   查找编号为 i 的 MDS 是否需要数据传输;
    if(需要)
    {   if(迁移要求者)
        {   发送消息到数据源，要求传输数据以及数据特征(hash 表位置);
            wait();
            while(数据未传输完)
            {接受数据；}
            置该项数据标志为不需传递;
            广播，更新 hash 表项；}
        if(数据源)
        {   接受数据传递消息;
            定位需传输数据;
            while(数据未传输完){发送数据；}
            发送数据传递完消息；  }
}}
```

Hash 表大小的选择要考虑服务器集群可能扩展的大小，同时要在共享 hash 表付出的代价和负载是否完全均衡之间权衡。由于系统中所有服务器的权值之和不一定刚好整除 hash 表大小，hash 表越大，而且每个位置在全局中所占的比例较小，不能整除而多余的位置由某些服务器占据使服务器多分配的数据所占比例就会更小，可以更好地实现负载均衡；但是，在服务器集群中需要共享 hash 表，hash 表越大共享 hash 表所付出的代价越大。

当服务器集群中有服务器需要换出时，比如某个服务器要进行维修和升级时，可以设置该服务器的权值为 0，系统重新分配数据，则该服务器上不再分配数据，通过数据迁移，该服务器上的数据将迁移到其他服务器上，这时可以将该服务器换出。

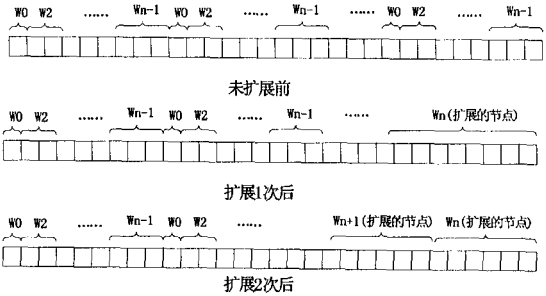

图 1 服务器在 hash 表中分布示意图

3.3 模拟测试

为了验证本方案的效果，采用模拟程序进行测试。本测试中，主要是检验采用本负载均衡方案是否能达到负载均衡的目的且效果如何。具体方法如下：在 MPI 平台上，以若干处理机模拟 MDS 和 client，元数据按负载均衡方案在各 MDS 上分配，client 随机发出文件处理请求，各 MDS 接受 client 发出的请求并处理，观察各 MDS 处理的请求数并计算所占比例来检验效果。表 1 为多次测试总的效果对比，表 2 权值为 1 和 3 的两个 MDS 组成的集群运行过程负载分配情况。

表 1 几种集群配置条件下总体负载分配效果

序号	参数		理想值	测试值
	MDS 数量(m)	权值 W(Si)		
1	2	1,1	50%,50%	50.6%,49.4%
2	2	1,3	25%,75%	25.8%,74.2%
3	3	1,2,3	16.67%,33.33%,50%	17.78%,35.57%,46.65%
4	4	1,2,3,4	10%,20%,30%,40%	10.45%,23.27%,27.22%,39.05%

表 2 权值为 1 和 3 的两个 MDS 组成的集群运行过程负载分配情况

总请求数	各 MDS 处理请求数及比例		总请求数	各 MDS 处理请求数及比例	
	0 号 W=1	1 号 W=3		0 号 W=1	1 号 W=3
170	43，25.29%	127，74.71%	280	72，25.71%	208，74.29%
180	45，25%	135，75%	290	75，25.86%	215，74.14%
190	48，25.26%	142，74.74%	300	77，25.67%	223，74.33%
200	50，25%	150，75%	310	80，25.81%	230，74.19%
210	51，24.29%	159，75.71%	320	84，26.25%	236，73.75%
220	54，24.55%	166，75.45%	330	85，25.76%	245，74.24%
230	58，25.22%	172，74.78%	340	86，25.29%	254，74.71%
240	63，26.25%	177，73.75%	350	89，25.43%	261，74.57%
250	65，26%	185，74%	360	92，25.56%	268，74.44%，
260	67，25.77%	193，74.23%	370	94，25.41%	276，74.59%
270	69，25.56%	201，74.44%	380	100，26.32%	280，73.68%

从上述两表可以看出，运用本方案的 MDS 集群中，负载量总体上和运行过程中都是按照权值在各 MDS 上的，较好地实现了负载均衡。

4 小结

本文针对元数据服务的特点提出了一种分布管理负载分发的负载均衡方法。该方法能克服集中管理负载均衡方法的缺点，并具有以下特点：能很好地实现负载均衡，hash 表大小为 hash_len，则每个服务器的负载差别不会大于 1/hash_len，若 hash 表大小为 256 时，仅为 0.39%；具有很好的扩展性，可以根据总负载量的大小任意加入新的服务器；扩展前后数据重叠大，系统扩展过程中，需要迁移的数据查找很方便而且数据迁移量小，扩展调整的代价不大，对集群对外服务的影响小；通过设置权值，可以将某个服务器换出系统。

参 考 文 献

[1] Peter J. Braam(with others)：The Lustre Storage Architecture，Cluster File Systems, Inc. http://www.lustre.org/docs/lustre.pdf ，03/04/2004

[2] Wensong Zhang：Linux Virtual Server for Scalable Network Services， http://www.LinuxVirtualServer.org/

[3] 章文嵩，王召福，刘仲：基于对象存储的集群文件系统 CFSlight 设计与实现，大连理工大学学报，2003 Vol.43 No.z1

[4] 许广斌：基于linux的集群系统 http://www-900.ibm.com/developerWorks/cn/linux/cluster/linux_cluster/part1/index.shtml

[5] 刘爱洁：负载均衡技术浅析，信息产业部北京邮电设计院第七届新技术论坛

对象文件系统中元数据服务器集群的负载均衡研究

作者： 邵强， 刘仲， 窦勇
作者单位： 国防科学技术大学计算机学院,长沙,410073