MooseFS 中 chunkserver 负载均衡算法研究*

艾云霄,谭跃生,王静宇
(内蒙古科技大学 信息工程学院,内蒙古 包头 014010)

摘 要:作为云存储的核心基础平台,分布式文件系统的重要性日益凸显。分布式文件系统中数据存储在多台计算机节点上,必然会出现负载均衡问题。首先,对 MooseFS 的系统架构进行了研究,然后分析了 MooseFS 分布式文件系统中 chunkserver 选择算法,研究了 chunkserver 算法的负载均衡性能,最后对其进行了改进。经过实验测试对比,实验结果显示改进算法能显著提高 chunkserver 的负载均衡性能。

关键词:分布式文件系统;MooseFS;数据存储;负载均衡
中图分类号:TP393 **文献标识码**:A **文章编号**:1674-7720(2013)05-0001-03

Chunkserver load balancing selection algorithm on MooseFS

Ai Yunxiao, Tan Yuesheng, Wang Jingyu
(College of Information Engineering, Mongolia University of Science and Technology, Baotou 014010, China)

Abstract: As the core foundation platform of the cloud storage, distributed file system is more and more important.In the distributed file system, data is stored in multiple computers, the appearance of the load balancing problem is inevitable. Firstly, research the architecture of the MooseFS system, and then research the load balancing performance of chunkserver algorithm. Finally, modify the chunkserver algorithm. After comparison of the experimental tests, the experimental results show that the modified algorithm can significantly improve the load-balanced performance of the chunkserver.

Key words: distributed file system; MooseFS; data storage; load balancing

随着云计算迅速发展,IT 界将进入“云”时代。然而,云计算[1]中会产生海量的数据存储,传统的文件系统已不能满足其性能要求。作为云存储的核心基础平台,分布式文件系统的重要性日益凸显。目前,互联网上应用最多的分布式文件系统有 GFS[2]、HDFS[3]、MooseFS 等。MooseFS 分布式文件系统,其设计思想来源于 google 文件系统,采用的是主从式服务器架构,通过将文件数据分成 64 MB 的 chunk 块分散存储在多台通过网络连接起来的计算机节点上,这种模式不可避免地存在一些节点分配的 chunk 块过多,而另外一些节点却是空闲的,导致系统的 chunkserver 数据块分配负载不均衡问题。

数据的负载均衡是分布式文件系统的核心之一,是否有好的负载均衡算法直接影响系统的性能,如果算法没有选择好,会导致负载严重失衡,使系统的性能不能得到充分的发挥。因此有必要研究 chunkserver 的数据块负载均衡选择算法,以解决 chunkserver 数据块分配的负载均衡问题。

*基金项目:内蒙古自然科学基金资助项目(2012MS0912);教育部春晖计划资助项目(Z2009-1-01044)

1 相关工作

负载均衡[4-5]的实现方法主要有静态模式和动态模式。静态模式是指在系统执行前,提前采取相应措施,把数据存储到各个节点上,尽可能地保证系统运行过程中不出现负载不均衡现象。动态模式是指在系统执行过程中,实时根据节点的存储状况来实现负载均衡。很显然,静态模式仍然还会有较高的概率出现系统负载不均衡现象,动态模式虽然实现起来比静态模式复杂,但是执行后效果好。MooseFS 分布式文件系统就是采用动态模式来实现 chunkserver 的负载均衡的。

负载的量化有多种标准,如 CPU 利用率、内存利用率等。目前,最常见的负载均衡算法有轮转法、随机法、散列法、最快响应法[3]等。轮转法,均衡器将新的请求轮流发给节点表中的下一个节点,是一种绝对平等。随机法,把伪随机算法产生的值赋给各节点,具有最大或最

小随机数的节点最有优先权，各个节点的机会也是均等的。散列法也叫哈希法，利用单射不可逆的 HASH 函数，按照某种规则将新的请求发送到某个节点。最快响应法，平衡器记录自身到每个节点的网络响应时间，并将下一个到达的连接请求分配给响应时间最短的节点。

本文以 chunkserver 上 chunk 块的多少作为负载均衡的指标。这里负载均衡是指各个 chunkserver 上 chunk 块数的多少大致相同，不会出现一些 chunkserver 上块数很多，而另外一些 chunkserver 上块数很少或是没有块数，造成一些 chunkserver 运行繁忙，而一些 chunkserver 处于空闲状态的不均衡现象。

2 MooseFS 的 chunkserver 负载均衡算法

Moose File System[6]是一个具备容错功能的网络分布式文件系统，它将数据分布在网络中的不同服务器上，MooseFS 通过 FUSE 使之看起来就是一个 Unix 的文件系统。即分布在各个范围的计算机将它们未使用的分区统一进行管理使用的一种文件系统。

2.1 MooseFS 文件系统架构

MooseFS 分布式文件系统主要由四部分组成[7]：

（1）管理服务器 managing server（master）：负责各个数据存储服务器的管理，文件读写调度，文件空间回收以及恢复，多节点拷贝。

（2）元数据日志服务器 Metalogger server（Metalogger）：负责备份 master 服务器的变化日志文件，文件类型为 changelog_ml.*.mfs，以便于在 master server 出问题的时候接替其进行工作。

（3）数据存储服务器 data servers（chunkservers）：负责连接管理服务器，听从管理服务器调度，提供存储空间，并为客户提供数据传输。

（4）客户机挂载使用 client computers：通过 fuse 内核接口挂接远程管理服务器上所管理的数据存储服务器，使共享的文件系统和本地 unix 文件系统的使用效果类似。

2.2 chunkserver 负载均衡算法

在 MFS 系统中，当客户端向数据存储服务器上传文件时，这些被上传的文件被划分成 64 MB 大小的 chunk 块，然后再根据 chunkserver 选择算法被存储在数据存储服务器上。如果 chunk 块被均衡分配，则系统不会出现一些 chunkserver 运行繁忙，而一些 chunkserver 处于空闲状态的现象，提高了用户访问系统的速度。

MFS 源代码中定义了 matoceeerventry 结构体，用来描述 chunkserver 的信息。在这个结构体中有一个 carry 变量，它是 MFS 中数据存储时分布算法的核心。MFS 中每台 chunkserver 会有自己的 carry 值，在选择 chunkserver 时会将每台 chunkserver 按照 carry 值从大到小做快速排序，优先选择 carry 值大的chunkserver 来使用。算法流程图如图 1 所示。其中，allcnt 表示 mfs 中可用的 chunkserver 的个数，availcnt 表示 mfs 中当前可以直接存储数据的

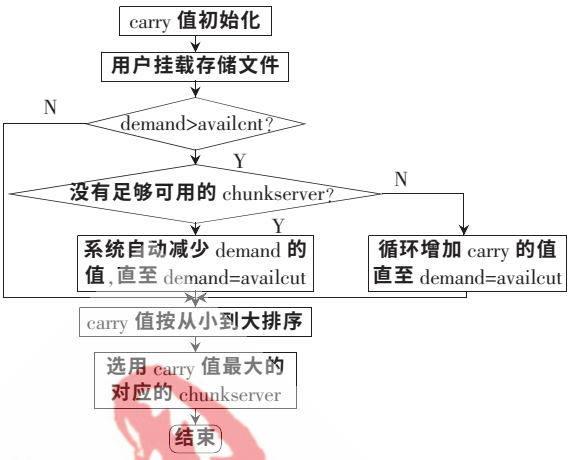

图 1 chunkserver 负载均衡算法流程图

chunkserver 的个数，demand 表示当前文件的副本数目。

MFS 系统启动时，通过 rndu32 () 函数为每一个 chunkserver 随机产生一个大于 0 且小于 1 的 carry 值。系统运行时，每台 chunkserver 的 carry 值的变化满足以下规律[8]：

（1）仅当 carry 值大于 1 时，才可以向此 chunkserver 中存储数据，并将此 chunkserver 的 carry 值减 1。

（2）当 demand>availcnt 时，循环增加每台 chunkserver 对应的 carry 变量的值，直到满足 demand<availcnt 时为止。

（3）变量 carry 每次增加的增量为本台 chunkserver 的总空间与系统中总空间最大 chunkserver 的总空间的比值。

根据以上算法的分析可知，在 MFS 系统中，数据并不是均匀地分配到各台 chunkserver 上的，而是 chunkserver 总空间大的，分配到的数据就多，即分配到 chunkserver 上的数据与此 chunkserver 的总空间大小成正比。如果 chunkserver 的总空间大小相同，则数据被均匀分配到 chunkserver 上，表 1 为随机生成 500 个、1 000 个、1 500 个、2 000 个文件时，chunk 块在各个 chunkserver 上的分布，测试结果显示，数据被均匀分配到各个 chunkserver 上。

表 1 chunk 块的分布情况测试

测试轮次	1	2	3	4
chunkserver1	167	334	500	666
chunkserver2	167	333	500	667
chunkserver3	166	333	500	667

2.3 改进的 chunkserver 负载均衡算法

在 MFS 系统中，如果 chunkserver 的总空间大小差别很大，就会造成总空间大的 chunkserver 被多次选择，chunk 块数多，而总空间小的 chunkserver 很少或几乎不被选择，chunk 块数少，造成 chunk 块分布不均衡。在图 1 整个算法流程图中循环增加可直接存储数据的 chunkserver 的个数，即增加 carry 的值直至 demand = availcnt 是负载均衡算法的核心部分，而其中 carry 的增加量 servtab[allcnt].w 如何计算是算法的关键问题。增加可直接存储数据的 chunkserver 的流程图如图 2 所示，算法实现代码如下：

```
while(availcnt<denand){
    availcnt-0;
    for(i=0;i<allcnt;i++){
        carry =servtab  [i].carry +servtab
        [i].w;
        servtab[i].carry=carry;
        servtab[i].ptr->carry=carry;
        if(carry>=1.0){
          avalicnt++;
         }
    }
}
```

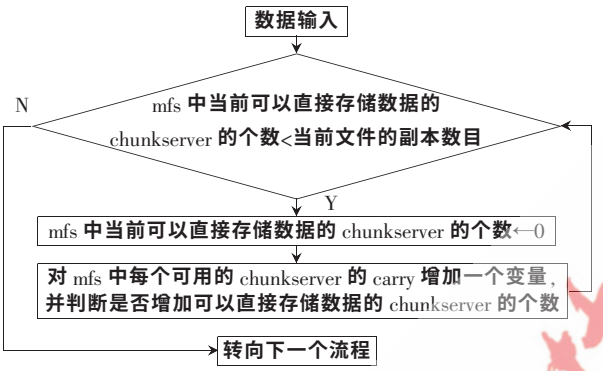

图 2　chunkserver 增加算法流程图

在原算法中 carry 的增加量 servtab [allcnt].w = (double)eptr->totalspace/(double)maxtotalspace，就是把本台 chunkserver 的总空间与系统中总空间最大 chunkserver 的总空间的比值作为 carry 变量的增加量。而改进后 carry 的增加量 servtab [allcnt].w=((double)maxtotalspace-(double)eptr->usedspace)/(double)maxtotalspace，就是把系统中总空间最大 chunkserver 的总空间减去本台 chunkserver 已用去的空间大小后与系统中总空间最大 chunkserver 的总空间的比值作为 carry 变量的增加量。

2.4 对改进负载均衡算法的测试

本测试的实验环境是在 VMware 里虚拟出 5 台虚拟机，1 台 master，3 台 chunkserver，1 台 client。其中，3 台 chunkserver 的硬盘大小分别为 5 GB，8 GB，11 GB，其他配置均相同。测试的主要目的是检测改进的算法是否能将数据均匀地存储到各台 chunkserver 上，此时系统的冗余备份设置为 1。

client 的挂载目录为/mnt/mfs/test。测试脚本为：

```
#! /bin/bash
for((i=0;i<1000;i++))
do
   dd  if=/dev/zero  of  test"$i"  bs="$RANDOM"
   count=1
   cp  test"$i"/mnt/mfs/test
done
```

利用测试脚本随机生成 1 000 个随机文件，然后上传到 MFS 系统中。算法改进前后 chunk 块的分布情况如表 2 和表 3 所示。

表 2　算法改进前 chunk 的分布情况

测试轮次	1	2	3	4
chunkserver1	199	200	202	198
chunkserver2	326	328	330	323
chunkserver3	526	504	509	531

表 3　算法改进后 chunk 的分布情况

测试轮次	1	2	3	4
chunkserver1	317	343	351	345
chunkserver2	311	339	348	341
chunkserver3	314	375	347	357

实验分别对改进前和改进后做了 4 次测试。从测试结果可以看出，算法改进前 chunkserver 硬盘容量越大，其上数据的分布就越多，这种情况容易导致各台 chunkserver 上的访问压力不一样，使系统性能不能达到最优。算法改进后，数据在 chunkserver 上基本是平均分配，各台 chunkserver 访问压力也基本一致，避免了总空间大的 chunkerver 总被不停地访问，而总空间小的 chunkserver 被闲置，使系统性能得到了优化。

本文对 MooseFS 分布式文件系统进行了分析，针对 chunkserver 选择算法存在负载不均衡的不足进行了改进，避免出现系统中总空间大的 chunkserver 上存储 chunk 块数多，访问量大，而总空间小的 chunkserver 上存储的 chunk 块数少或没有 chunk 块存数而处于闲置状态。通过实验测试，改进后达到了预期的效果，chunk 块在各个 chunkserver 上分布均衡，系统性能得到优化。

参考文献

[1] 王德政，申山宏，周宁宁.云计算环境下的数据存储[J].计算机技术与发展，2011，21(4)：81-82.
[2] GHEMAWAT S，GOBIOFF H，LEUNG S T. The Google file system[C]. Proceedings of the 19th ACM Symposium on Operating Systems Principles.Lake George，New York：2003：29-43.
[3] APACHE HADOOP.Hadoop[EB/OL].[2009-03-06].(2012-03-19)http://hadoop.apache.org/.
[4] 谭支鹏.对象存储系统副本管理研究[D].武汉：华中科技大学，2008.
[5] 张聪萍，尹建伟.分布式文件系统的动态负载均衡算法[J].小型微型计算机系统，2011，32(7)：1424-1426.
[6] 百度文库.MFS 文档[DB/OL].2010.http://wenku.baidu.com/view/320b56260722192e4536f61b.html.
[7] 51CTO 博客.MooseFS 介绍[DB/OL].2011.http://haiquan517.blog.51cto.com/165507/526252.
[8] mfs（mooseFS）深入分析（chunkserver 选择算法）[DB/OL].2011.http://www.oratea.net/?p=285#comment-481.

（收稿日期：2012-12-13）

作者简介：

艾云霄，女，1987 年生，硕士研究生，主要研究方向：计算机网络和云计算。

谭跃生，男，1959 年生，教授，硕士生导师，主要研究方向：网格计算和计算机网络。

王静宇，男，1976 年生，副教授，硕士生导师，主要研究方向：网格与云计算安全有关理论和技术。