基于代理的并行文件系统元数据优化与实现

易建亮 1，陈志广 1, 肖侬 2，卢宇彤 1

1国防科学技术大学计算机学院，　2国防科学技术大学高性能计算国家重点实验室

论文背景

并行文件系统的所有功能由元数据服务器和数据服务器实现。其中，元数据服务器是整个并行文件系统的核心。早期的高性能计算系统规模较小，并行文件系统元数据服务器承载的压力较低，一般采用单节点的元数据服务器即可满足整个计算系统的IO需求。随着计算系统规模的不断增大，尤其是百万核系统（天河二号包含312万个计算核心）的陆续出现，传统的单节点元数据服务器逐渐不能满足高并发访问需求。因此，各种并行文件系统逐步采用分布式元数据管理方式，提供可扩展的元数据服务。

鉴于计算系统规模不断增大，尤其是E级计算系统在不久的将来即将出现，并行文件系统的分布式元数据管理仍然面临以下两方面的挑战：

（1）高性能计算系统中计算节点数目不断增加，且经常出现大量节点同时产生并发IO请求的情况，传统的分布式元数据管理机制逐渐难以应对超大规模计算系统产生的并发元数据操作；

（2）对于高性能计算系统中运行的一个任务，参与该任务的所有进程往往在同一时间点向同一文件夹发出大量的IO请求，这会导致该文件夹下很重的元数据访问负载，而传统的基于子树划分的负载均衡机制并不能将同一文件夹下的元数据负载分散到多个元数据服务器上。

目标

本文针对以上两方面的挑战，结合高性能计算系统中经常出现属于同一任务的大量进程向同一文件夹发出并发元数据操作的负载特点，在并行文件系统中引入代理机制，通过代理上的元数据聚合和动态负载均衡，显著提升元数据操作的性能。实验表明，本文提出的基于代理的元数据优化机制具备优秀的平均文件I/O性能及可扩展的元数据管理方案，平均文件访问时延不到2ms。

元数据代理的并行文件系统架构

如图1所示，系统由四个组件构成：客户端、元数据服务器、代理服务器和数据服务器。

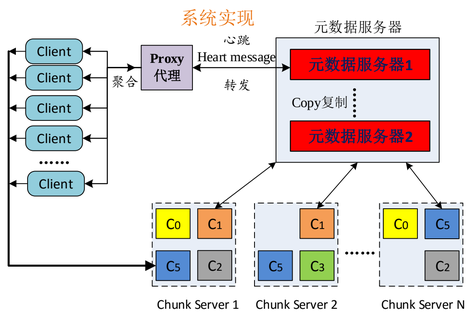

图1 系统架构图

Fig.1 System Architecture

本文提出的优化措施将在代理服务器上实现；数据服务器实现用户文件的本地存储。

元数据服务器（MDS）

- ➢存储每个文件/目录的关键元数据；
- ➢管理文件系统名字空间；
- ➢文件/目录和对象物理存储位置之间的映射。

代理服务器（Proxy）

- ➢实现客户端和元数据服务器之间的桥接；
- ➢接受客户端的IO请求，
- ➢经适当处理后转发到元数据服务器

元数据负载均衡

在分布式元数据管理方式中，为了防止热点出现，系统需要一套负载均衡机制，以实时监控节点状况，调整负载分配，达到均衡状态。

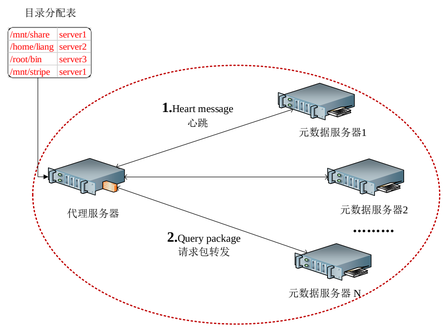

图2 负载均衡算法图解

Fig.2 Graph of load-balance algorithm

目录树分配表

为了使系统达到初始均衡状态，我们对目录树进行切分，将整个目录树划分成若干个子目录树结构（根据节点数量确定），并根据各节点配置情况，将这些子目录树分摊到相应节点，从而达到静态均衡。

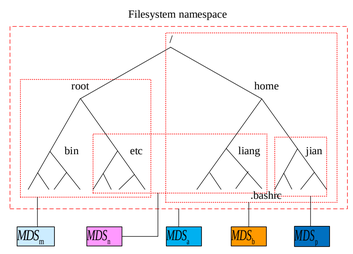

图3名称空间的分区。

Fig.3 Partitions of Namespace

表 1 目录树分配表。

Table1 Partition Table of Directory Subtree

目录子树路径	MDS 元数据服务器
/root/etc	MDSn
/root/bin	MDSm
/home/jian	MDSp
/home/liang	MDSb

负载计算方法

使用访问延迟来衡量某个服务器上的负载情况，对于连续I/O操作，则使用吞吐量比较合理

$$Latency_i(t) = (1-\alpha)\cdot Latency_i'(t) + \alpha \cdot Latency_i(t-1) \quad (1)$$

在某个时刻t，观测服务器Si上的访问延迟为Latencyi(t)。

设在时刻t-1到时刻t这段时间内，服务器Si上的平均访问延迟为Latencyi(t)，其中i代表第i个节点，t为时间。

$$AvgLatency(t) = \sum_{i=1}^{n} w_i(t-1)\cdot Latency_i(t) \quad (2)$$

$\{w_1(t), w_2(t), \ldots, w_n(t)\}$为各节点分配权值。

负载均衡策略

集合A包含负载较轻的节点，集合B包含负载较重的节点。我们的目标是动态地调整服务器负载，使得集合B中服务器将部分负载转移到集合A中服务器上，并调整目录分配表条目，达到动态负载均衡的目的。

- ➢ 当Latencyi(t)<AvgLatencyi(t)时，将服务器Si归为集合A；
- ➢ 当Latencyi(t)>AvgLatencyi(t)时，将服务器Si归为集合B。

$$f(x) = \begin{cases} 0 \ldots \psi(A) > \psi(B) \\ 1 \ldots . \psi(A) \le \psi(B) \end{cases}$$

其中，f(x)=0代表集群负载均衡状态，f(x)=1则代表集群失衡状态。 代表集合A，B中元素个数。

副本策略

将每一组配对中前一个节点上的目录树复制一份到配对后一个节点上，如下图所示：

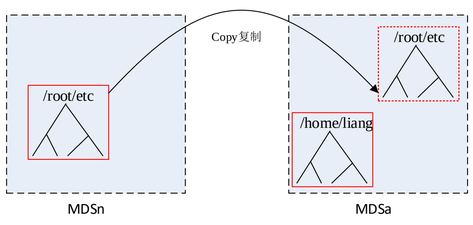

图4 目录树复制

Fig.4 Copy of directories subtree

由于元数据规模很小，在1PB数据量的文件系统中，理论上元数据量不超过2GB，分在每个节点上的元数据较少，所以复制过程并不会对系统性能造成重大影响。

元数据代理的系统实现

系统由三个组件构成：客户端、元数据服务器和代理服务器。如图5所示

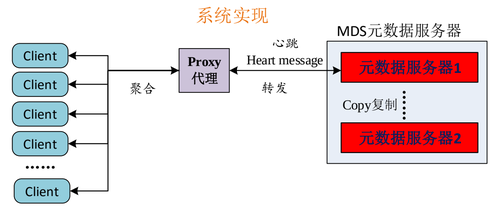

图5 系统实现结构

Fig.5 The architecture of system

如图5所示，元数据服务器（MDS）存储每个文件/目录的关键元数据，主要功能包括：管理文件系统名字空间等；代理服务器（Proxy）维护目录分配表，实现元数据服务器集群动态负载均衡。

实现步骤

如图5，具体包含如下步骤：

1、应用程序调用Client端API接口，发出文件Open操作请求；

2、请求通过网络传送给代理服务器端；

3、代理服务器负责接收来自Client端的文件请求，并将收集到的请求根据目录分配表进行分组，并将每个装满的分组打包成请求包，对应元数据服务器的处理包；

4、代理服务器上的转发器选择将各请求包转发给对应元数据服务器进行处理；

5、元数据服务器将请求包“解包”成相应文件请求，对每个请求执行相应的操作；

6、元数据服务器将处理结果分发给相应Client端。

性能测试及扩展性评估

元数据更新延迟

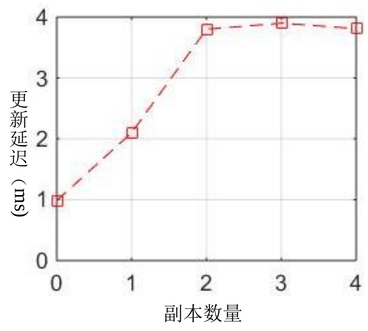

图6 元数据更新延迟

Fig.6 Delay of Metadata Update

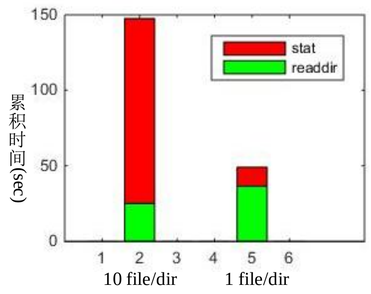

图7 ls命令累积时间

Fig.7 Cumulative Time of ls Command

性能测试及扩展性评估

扩展性评估

:

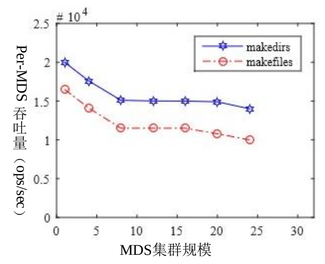

图9 随着集群规模的扩展，单节点IOPS的变化情况。

Fig.9 Per-MDS TPS along with Cluster’s Scale

从图中可以看出，MDS集群节点平均吞吐量从单个节点时的2000 IOPS降到了24个节点时的1380 IOPS。当节点规模24时，单节点下降到理想情况的69%，验证了系统具有较好的扩展性。