计算机科学
COMPUTER SCIENCE

面向 Lustre 集群存储的错误日志分析及系统优化

程稳，李焱，曾令仿，王芳，唐士程，杨力平，冯丹，曾文君

引用本文

程稳，李焱，曾令仿，王芳，唐士程，杨力平，冯丹，曾文君．面向 Lustre 集群存储的错误日志分析及系统优化[J]．计算机科学，2022，49(10)：1-9.

CHENG Wen，LI Yan，ZENG Ling-fang，WANG Fang，TANG Shi-cheng，YANG Li-ping，FENG Dan，ZENG Wen-jun. Error Log Analysis and System Optimization for Lustre Cluster Storage[J]. Computer Science，2022，49(10)：1-9.

相似文章推荐（请使用火狐或 IE 浏览器查看文章）

Similar articles recommended (Please use Firefox or IE to view the article)

存在 CSI 估计错误的增强型 ELM 叠加 CSI 反馈方法

Enhanced ELM-based Superimposed CSI Feedback Method with CSI Estimation Errors

计算机科学，2022，49(6A)：632-638. https://doi.org/10.11896/jsjkx.210800036

一种面向电能量数据的联邦学习可靠性激励机制

Reliable Incentive Mechanism for Federated Learning of Electric Metering Data

计算机科学，2022，49(3)：31-38. https://doi.org/10.11896/jsjkx.210700195

分类学习算法的性能度量指标综述

Survey for Performance Measure Index of Classification Learning Algorithm

计算机科学，2021，48(8)：209-219. https://doi.org/10.11896/jsjkx.200900216

一种传输时限下认知无线电网络的动态广播策略

Dynamic Broadcasting Strategy in Cognitive Radio Networks Under Delivery Deadline

计算机科学，2021，48(7)：340-346. https://doi.org/10.11896/jsjkx.200900001

基于程序变异和高斯混合聚类的错误定位技术

Fault Localization Technology Based on Program Mutation and Gaussian Mixture Model

计算机科学，2021，48(6A)：572-574. https://doi.org/10.11896/jsjkx.200500121

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.220100134

面向 Lustre 集群存储的错误日志分析及系统优化

程 稳[1] 李 焱[2] 曾令仿[3] 王 芳[1] 唐士程[2] 杨力平[2] 冯 丹[1] 曾文君[2]
1 华中科技大学武汉光电国家研究中心信息存储系统教育部重点实验室暨数据存储系统与技术教育部工程研究中心 武汉 430074
2 深圳国家基因库 广东 深圳 518120
3 之江实验室 杭州 311121
(chengwen@hust.edu.cn)

摘 要 集群存储系统的错误日志信息有助于优化存储系统的可用性和稳定性。现有存储系统错误探究主要针对单机存储系统或集群存储系统的部分功能进行分析评估,缺乏在实际应用场景下,同一生产环境中,长时间、多视角的探究工作。新型功能模块的不断融入,使得集群存储系统日益庞杂,集群存储系统自身引发的错误层出不穷,给各类研发人员带来了困扰与挑战。针对以上问题,提出了面向 Lustre 集群存储的错误日志分析及系统优化策略,通过收集连续 1 673 天的错误日志,研究了近 2.26 GB 的 Lustre 错误日志,分析了多个版本 Lustre 错误的特点与问题,揭示了集群存储系统各方面的不足与错误,研究了不同 Lustre 版本错误的影响因素,总结了 Lustre 集群在实际生产环境中的常见错误,并给出了相应的解决方案。对 Lustre 系统研发有了许多新的见解,并总结了 14 个发现,最后通过采集 333 天的新增错误记录对 14 个发现进行了相关验证,给出了一些系统错误优化实例。相关测试表明,优化实例可以显著减少错误数量,提高系统的可用性和稳定性,研究结果和建议对集群存储系统本身的发展以及集群存储系统的运行和维护都有一定的参考价值。

关键词:Lustre 文件系统;日志分析;系统优化;错误;可靠性
中图法分类号 TP399

Error Log Analysis and System Optimization for Lustre Cluster Storage

CHENG Wen[1], LI Yan[2], ZENG Ling-fang[3], WANG Fang[1], TANG Shi-cheng[2], YANG Li-ping[2], FENG Dan[1] and ZENG Wen-jun[2]
1 Wuhan National Laboratory for Optoelectronics, Key Laboratory of Information Storage System, Engineering Research Center of Data Storage Systems and Technology, Huazhong University of Science and Technology, Ministry of Education of China, Wuhan 430074, China
2 China National GeneBank, BGI-Shenzhen, Shenzhen, Guangdong 518120, China
3 Zhejiang Lab, Hangzhou 311121, China

Abstract Cluster storage system error messages can help to optimize the availability and reliability of storage system. Previous research of storage system error analysis focuses on the local file system or a part of the cluster storage system. There is a lack of research on storage system error messages for a long-time and multi-dimension in practical applications. With the continuous integration of new functional modules, the cluster storage system is becoming more and more complex, and the errors caused by cluster storage system emerge endlessly, which brings troubles and challenges to the researcher and developer. To address the problems, we conduct a comprehensive study of the Lustre system error log. By collecting the error log in 1 673 consecutive days, we study nearly 2.26 GB of Lustre error logs, analyze the characteristics and problems of the Lustre system errors in multiple Lustre versions. We show that correlated errors between different subsystems and study the possible impacting factors on different Lustre versions. We also summarize the common errors in the Lustre system and show the corresponding solutions. We derive numerous new insights into the Lustre system development process and report 14 findings. Finally, we collect new error logs for 333 consecutive days to verify the 14 findings and give some cases about error optimization. Experimental results show that the error

到稿日期:2022-01-14 返修日期:2022-05-06
基金项目:国家自然科学基金重点项目(61832020);国家自然科学基金创新研究群体项目(61821003);之江实验室中心自设科研项目(2021DA0AM01)
This work was supported by the Key Program of National Natural Science Foundation of China(NSFC)(61832020), Creative Research Group Project of NSFC(61821003) and Center-initiated Research Project of Zhejiang Lab(2021DA0AM01).
通信作者:王芳(wangfang@hust.edu.cn)

optimization cases can significantly reduce the number of errors and improve the availability and stability of the system. Our results and suggestions should be useful for both the development of the cluster storage system themselves as well as the Lustre operation and maintenance.

Keywords Lustre file system, Log analysis, System optimization, Bug, Reliability

集群存储系统承担着为 HPC 系统提供可靠服务的使命，但随着其规模的不断扩增，设备数量快速增长，故障处理与恢复已成为频繁事务，严重影响了系统的可用性和稳定性[1]。研究集群存储系统自身的错误，综合评估相关错误与设计的不足，优化集群存储系统，是大幅度提升系统性能、简化管理与维护的重要方法。但探究集群存储系统自身错误的工作比较少见，主要原因之一是：一个集群存储系统从开发到成熟需要很长时间(大约 10 年)，需要通过长时间且连续不断的研发、实际运用与优化来发现问题，进而提升系统的可用性与稳定性[2]。

存储系统整体探究类工作较少，但针对存储系统某一方面的不足进行优化的研究较多，如针对元数据可能会因硬件故障、软件错误、配置错误等而损坏的问题进行的研究，对 Lustre 文件系统检查器(Lustre File System Checker，LFSCK)的性能探究工作[3]。然而，局部优化由于其局限性，并不能代替对整个系统的研究[4]，这促使我们对整个系统优化研究工作的探索。由于 Lustre[5-7] 集群存储系统在 HPC 领域的广泛运用和其良好的设计理念，我们将它作为研究对象来探究集群存储系统的特点与问题。

在 Lustre 集群存储系统自身错误探究方面，有以下关键问题仍待探讨：1)错误分类与常见错误；2)子系统(client，Object Storage Server(OSS)和 Metadata Server(MDS))错误发生频次；3)不同版本相同或相似错误是否会重复出现；4)系统升级(不同版本)对错误的影响；5)发生错误的个数、时间点是否可以预测；6)错误一般的持续时间；7)根据已有错误记录，如何优化集群存储系统。这些问题的答案有助于集群存储系统在生产环境中实现高可用性和可靠性。

面对以上挑战，我们对国家基因库同一生产环境中，5 个 Lustre 集群存储系统的 client，MDS 和 OSS 连续 1 673 天的错误日志信息进行了采集，针对多个 Lustre 版本、多个 Lustre 集群存储系统错误的特点与问题，从时间和空间维度对错误的相关规律与特性进行了探索，分析并归纳了集群存储系统各方面的不足与错误，给出了常规解决方案，并对 7 个关键问题进行了探究，通过分析有以下发现。

(1)通过对采集到的错误日志进行分析归纳，根据错误特性，我们把集群存储系统的错误分为 3 类：性能问题、存储问题和配置问题。其中，存储问题最常见，配置问题出现的频次主要由运维人员对系统的熟悉程度决定。性能问题和存储问题会随着版本迭代或新功能的引入大幅度增多，而当存储问题得到缓解后，性能问题会凸显出来。从一个大版本的生命周期来看，配置问题主要出现在前期，性能问题主要出现在中后期，存储问题却一直存在，而且占比较高。

(2)通过统计与分析发现，OSS 是最常发生错误的子系统，MDS 次之，client 相对来说较稳定，出现错误次数最少。在错误日志存储空间占用方面，OSS 错误日志存储空间占比最高，MDS 错误日志存储空间占比最低。另外，不同子系统错误的持续时间不一致，OSS 是错误时间持续最长的子系统。

(3) 集群存储系统发生错误的个数与时间点不可预测，有爆发期也有沉静期，许多错误在不同的时间节点重复出现，并且系统采用 Lustre 的不同版本，相似错误会重复出现，对 Lustre 版本升级会直接影响到错误发现的数量与频率。

(4) Lustre 系统还在不断成熟完善中，系统并不稳定，服务器端版本升级较为频繁。client 节点除了自身硬件或配置引起的错误外，其他几乎都是由于 MDS 或 OSS 无法提供相应服务而引起的错误。Lustre 集群系统对 client 最新版本的需求并不强烈，长期维护发行版可满足用户的一般需求，仅当 client 版本无法适配服务端版本时，才会升级到相应版本。

(5) 基于已有数据分析，一个更新后的 Lustre 系统大概需要 71 天来进行运维与优化，才能较好地确保其可用性与稳定性。在实际生产环境中，系统运维人员对集群系统升级的态度谨慎，相比新特性、新功能，他们更关注系统的稳定性、可用性与可维护性。

1 Lustre 集群错误日志数据

1.1 Lustre 集群系统信息

我们对国家基因库 5 个 Lustre 集群存储系统的错误日志信息进行了采集，该 Lustre 集群存储系统的相关信息如表 1 所列，其中 Lustre 和其 client 的版本信息是该 Lustre 集群存储系统 2020 年 6 月份的主流版本信息。另外，集群存储系统 MDS 端都是用 ldiskfs 作为 MDT 的后端存储，OSS 端都是用 zfs 作为 OST 的后端存储；MDT 和 OST 一般采用的是默认条带策略(No-striping)；关于 OST 文件分布，其基于加权因子连续使用循环和加权分配器，QOS_PRIO_FREE 调优参数采用的是默认参数，即 91%；集群服务的 client 节点的数量一般约为 200。

表 1 Lustre 集群存储系统的基本信息

Table 1 Basic information of Lustre cluster storage system

	L_A	L_B	L_C	L_D	L_E
CPU	E5 2650v4	E5 2640v4	E5 2650v4	E5 2640v4	Gold 5115
CPU 核数	24	20	24	20	10
内存/GB	128	128	128	128	64
容量/PB	1.68	17.40	1.68	1.68	26.80
MDS 数量	2	2	2	2	2
OSS 数量	6	16	4	10	32
Lustre ver	2.10.6	2.10.6	2.10.6	2.10.6	2.10.8
Lustre ver	2.10.6	2.10.6	2.10.6	2.10.6	2.10.6
OS ver	CentOS7.3	CentOS7.3	CentOS7.3	CentOS7.3	CentOS7.3

1.2 错误日志与数据采集

Lustre 相关日志主要分为 3 种：系统日志、内核控制台

日志和 Lustre 调试日志。由于 Lustre 软件代码运行在内核上，因此 Lustre 的部分信息会被记录在 Linux 的系统日志（即“/var/log/messages”文件）中；内核控制台日志包含了当前节点的内核和内核模块信息，其可以在 Lustre 日志“/proc/sys/lnet/debug_path”文件中通过内核控制平台（dmesg）查看；Lustre 调试日志拥有关于 Lustre 短时间内的运转操作信息，可以通过 lctl dk filename 命令来捕获。通常错误信息会包括以下内容：时间、进程（进程 ID）、问题描述、正在通信的服务器节点等信息。了解相关错误信息，就可以进一步分析问题，解决错误，从而提升系统稳定性、可用性和可维护性，优化系统的性能。

我们主要收集了错误节点和其关联节点的 messages，dmesg 和 dklogs 信息，错误日志的总大小为 2 374 925 kB，其中对于错误发生的具体信息与时间段，根据错误日志中的错误提示符（LustreError，LNetError，Error，LBUG 等）来确定。实际收集的是 2016-01-01 至 2020-07-31 的错误日志记录，由于错误随机发生，已有错误记录始于 2016 年 3 月 29 号，截止于 2020 年 4 月 28 号。另外，在所有错误中，有一个问题（zfs suspend）多次重复出现，这个问题会使得系统冻结当前状态，重启服务器。这个问题太过频繁，并且其问题本质与解决方案较为简单。关于该问题的本质和解决方案将在 4.4 节进行详细讨论。为了更客观地表现错误发生的实际情况，第 3 节将错误统计中的 zfs suspend 作为 zfs 异常统计一次。最后，下文若不做特殊说明，则默认时间段是 2016-01-01 到 2020-07-31。

1.3　局限性

本文工作的局限性如下：

（1）部分数据丢失。错误导致系统崩溃后可能会发生日志记录丢失的情况。崩溃错误是瞬时的，对分析中的错误类别、数量、错误变化趋势等没有影响，但对日志数据量大小和错误持续时长有一定影响。

（2）单一生产环境。我们仅收集了与 Lustre 系统相关的错误日志信息，应用场景主要是生物信息领域，可能会有一定的局限性。但长时间、多版本的 Lustre 系统的错误日志信息本身具有代表性，对 Lustre 集群存储系统的观察可以很容易地扩展到其他集群系统。

（3）应用联动分析。为了确保集群存储系统的通用性、安全性与可靠性等，集群存储系统仅为应用提供了一些抽象接口，从应用角度来看，表现为对集群系统的读写数据量大小与频率等。关于应用负载分析已有较多研究，并未发现应用与集群错误之间的直接联系，因此当前本文主要关注集群存储系统的自身错误，并没有结合上层应用进行联动分析。

2　相关工作

针对文件系统，采集相关数据并分析系统缺陷与演变，根据对象的不同，提出的相关优化方案的研究主要分为两大类：局部优化和整体优化。

（1）局部优化研究。现有的大部分工作都是局部优化研究类，一般研究存储系统某方面的问题，通常探索已有解决方案的不足，重新设计方案或优化已有方案。Lockwood 等[8]针对现有 HPC 领域 I/O 需求变化问题，量化分析了两个集群设施的全年 I/O 数据变化规律，总结了关于系统 I/O 与性能的相互关系，并对 HPC 存储从业者提出了一些建议。Dai 等[3]针对元数据损坏问题，根据 HPC 负载生命周期数据，优化 Lustre 文件系统检查器（LFSCK）的性能。Ji 等[9]针对移动设备端 Ext4 文件碎片与读写性能问题，探究了碎片化问题的起因及其影响，并给出了优化文件系统碎片的设想。总的来说，针对局部问题进行研究的主要不足是不能考虑到相互或多方影响，缺乏全局视角，难以确保给整个系统带来优化提升。

（2）整体优化研究。由于各方面因素的影响，对存储系统整体研究的工作一般较少，该部分研究工作主要是针对整个存储系统进行探索分析。Aghayev 等[10]针对 Ceph 的后端存储系统性能与设计限制，重新设计了一个全新的存储后端 BlueStore，并给出了分布式系统和其存储后端的设计建议。Agrawal 等[11]针对 Windows 个人电脑文件系统元数据的变迁问题，给出了将来个人电脑文件系统各方面的发展与设计建议。Lu 等[12-13]针对 Linux 的 bug 问题，对 2003（Linux 2.6.0）－2011（Linux 2.6.39）年 6 个主要文件系统提交的 patch 进行了分析处理，第一次量化了 Linux 文件系统 bug 的相关问题与趋势。总的来说，针对整个存储系统进行的研究工作较少，通常来自不同领域，大都没有交集，缺乏某些重要领域的研究工作，如面向 HPC 领域集群存储系统错误的相关探索研究。

本文主要研究的是同一生产环境中，多个 Lustre 集群存储系统的错误日志信息，这样可以大幅度减少/避免不同应用负载/硬件配置对我们日志探究的影响，使发现更具有实际意义。在采集错误日志的过程中，对该系统实际运行过程中产生的错误信息都进行了收集整理，最大程度地使采集的错误信息具有普遍性与代表性。另外，为了确保所获得的错误信息中错误发生时系统状态、报错信息、错误持续时间等信息的完整性与正确性，对错误节点（如被 MDS 驱逐的 client 节点）、关联节点（与错误节点相关的节点，如驱逐 client 的 MDS 节点）的 3 类日志信息，即 messages，dmesg 和 dklogs 进行了综合对比分析与提炼，其中 Lustre 相关错误信息的提炼主要是根据 LustreError，LNetError，Error，LBUG 等关键字确定的，而且系统一直有专业人员维护管理，历经了多个内核版本，因此我们的发现更加具有实际意义与价值。

3　错误日志分析

3.1　错误分类

我们采集的错误日志记录来源于 3 个子系统，即 Lustre client，OSS 和 MDS。我们对错误日志进行了分析归纳，根据错误特征把错误分为了以下 3 类：性能问题、存储问题和配置问题。表 2 列出了错误记录、错误描述及其出现次数等信息。从表 2 可以看出，常见错误主要有：zfs（Zettabyte File System）性能异常（zfs suspend）、quota 设置问题、Client 负载过重卡死、OSS recovery 问题、硬件设置和网络配置问题。

表 2 错误分类

Table 2 Classification of errors

错误类型	描述	错误	次数
性能问题	系统反应缓慢或无响应，性能下降但存储服务可用	zfs 性能异常(zfs suspend)	11
		Client 负载过重卡死	8
		MDS 负载过重卡死	4
		Client 反应缓慢	2
		OSS 负载过重卡死	1
		OSS 反应缓慢	1
存储问题	存储服务不可用、数据丢失、无法访问存储服务器	OSS Recovery 问题	6
		Kernel Panic	4
		Input/Output 错误	3
		zpool 脑裂数据丢失	2
		无法读取某些文件	2
		MDT ChangLog 问题	2
		空指针	2
		OSS 无响应	1
配置问题	人为因素占主要部分，设备或系统配置问题导致服务不可用或资源异常损耗	quota 设置不合理	8
		硬件设置问题	5
		网络设置问题	5
		Linux 系统配置问题	3
		Timeout 设置问题	2
		pool 设置问题	1

下面将结合时间和空间维度，进一步探究错误日志的相关规律与特性。

3.2 错误个数变化趋势

表 3 列出了采集的错误日志记录个数、错误类型与错误发生时间段等信息。从表 3 可以看出，相比其他年份，2016 年和 2020 年错误出现的次数较少；对于不同的子系统，OSS 出现错误的次数最多，client 出现错误的次数最少；虽然各类问题的占比比较均等，但相对来说，存储问题的占比高达 39.7%，并且各类错误出现的时间不可预测。在性能问题、存储问题和配置问题 3 类错误中，配置问题的占比为 30.1%，错误种类繁多，很难准确定位错误根源，但可以采用一些方案来减少配置问题重复出现，具体方案将会在 4.2 节进行讨论。

表 3 错误个数

Table 3 Number of errors

年份	时间段	不同对象错误记录个数									总数
		MDS			OSS			Client			
		性能	存储	配置	性能	存储	配置	性能	存储	配置	
2016	29 Mar—27 Apr		1			1		1	1		4
2017	18 Jan—29 Dec	3	3	4	2	3	1	4	1	2	23
2018	20 Jan—9 Dec	2	2	1	4	2	3	1	2	2	19
2019	14 Jan—5 Dec	1	2	3	1	8	5	2	1	1	24
2020	25 Apr—28 Apr					2	1				3
单个问题合计		6	8	8	7	16	10	8	5	5	73
错误记录合计			22			33			18		

发现 1：OSS 最常发生错误，需要优化；MDS 次之，需要关注；client 相对来说较稳定，其出现错误的次数最少。

发现 2：许多错误可以减少或避免，如配置问题引发的错误。

通过对表 3 中每年的错误记录进行分析可知，针对错误发生的个数、时间点预测问题，从数据中可以看出，全年的错误发生时间点比较随机，几乎每个月系统都会发生错误，错误发生的时间节点很难预测。

发现 3：系统采用 Lustre 不同版本，相似错误会重复出现。

发现 4：对 Lustre 版本升级会直接影响到错误发现的数量与频率。

3.3 不同子系统错误日志大小

表 4 列出了每年错误日志的大小。由表 4 可知，OSS 错误日志大小占比最高，MDS 错误日志大小占比最低。另外，MDS 每年的错误日志数量虽然占比较高(MDS 平均每年约有 4.73 个错误记录，OSS 平均每年约有 7.1 个错误记录，Client 平均每年约有 3.87 个错误记录)，但其错误日志大小较小。OSS 错误数量与其错误记录总体大小占比都是最高的。

表 4 错误日志大小

Table 4 Size of error log

年份		2016	2017	2018	2019	2020	合计/Gb
日志大小/kB	MDS	804	67092	4521	21232	0	0.71
	OSS	1111	492749	545408	280125	13540	10.71
	Client	92980	815794	29291	10278	0	7.24

发现 5：OSS 错误日志大小占比最高，MDS 错误日志大小占比最低。

3.4 Lustre 版本与错误分布规律

表 5 列出了采集的错误日志记录个数、错误类型与 Lustre 相关版本等信息。

表 5 Lustre 版本与错误个数

Table 5 Lustre version and number of errors

Lustre 版本	不同对象错误记录个数									错误总数
	MDS			OSS			Client			
	性能	存储	配置	性能	存储	配置	性能	存储	配置	
2.7		1			1		1	1		4
2.8.0	3	1	2	2			3			11
2.9.0		2	2		3	1	1	1	2	12
2.10.2	1	2	2	3	3	4	3	2	1	21
2.10.6	1	1	1		6	5	1	1	1	17
2.10.8		1	2		3	2				8

发现 6：系统发生错误的个数与时间点不可预测，有爆发期也有沉静期，许多错误在不同的时间节点重复出现。

图 1 给出了服务器端 Lustre 版本与 client 系统版本变迁图。从图中可以看出，client 系统仅升级了 3 次，其中重大升级仅一次；Lustre 系统升级较频繁，集群系统升级的时间节点与相应 Lustre 版本的发行时间有较长的时间间隔。

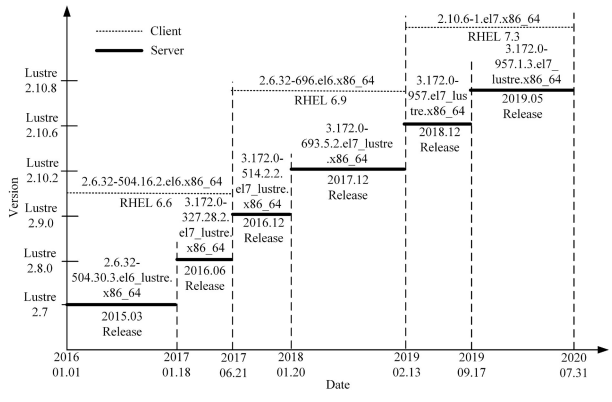


图 1 服务器端 Lustre 版本与 client 系统版本变迁图

Fig. 1 Version upgrade of Lustre server and client

发现 7：在实际生产环境中，Lustre 集群系统对 client 最新版本的需求并不强烈，长期维护发行版可满足用户的一般需求，仅当 client 版本无法适配服务端版本时，才会升级到相应版本。

发现 8：Lustre 系统还在不断成熟完善中，系统并不稳定，服务器端版本升级较为频繁。

发现 9：在实际生产环境中，系统运维人员对集群系统升级的态度谨慎，相比新特性、新功能，人们更关注系统的稳定性、可用性与可维护性。

client 版本的升级时间点和服务器端是重叠的，在分析 Lustre 版本变迁对错误发生的影响时，主要考虑服务端版本升级时间点即可。因此，仅当 client 版本无法适配服务端版本时，才会升级到相应版本。另外，通过对错误记录起始时间和错误根源的分析，client 节点自身硬件或配置引起的错误较少，大多数错误与 MDS 或 OSS 相关。

发现 10：client 节点除了自身硬件或配置引起的错误外，其他几乎都是由于 MDS 和 OSS 无法提供相应服务而引起的错误。

图 2 给出了错误累计分布信息的概况，其中采集时长可以代表系统优化运维时间，90% 错误记录间隔时长可以代表系统的稳定程度。从图 2 可以看出，一般情况下，运维优化的时间越长（红色点柱），错误间隔时间越长（黄色竖条柱），最大错误间隔也越长（蓝色波纹柱），但 Lustre 2.10.2 和 Lustre 2.10.6 版本的错误记录却有较大幅度的上升，错误发生比较频繁。综合表 5 和图 3 可以知道，Lustre 2.10.2 和 Lustre 2.10.6 版本的错误记录增长主要来自存储错误和配置错误。

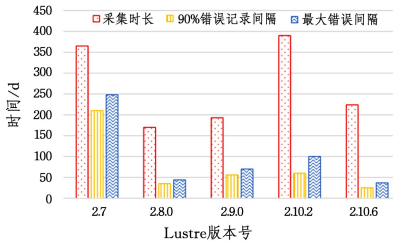


图 2　Lustre 版本与错误分布（电子版为彩图）

Fig. 2　Lustre version and error distribution

表 6 列出了 Lustre 集群（版本从 2.7 到 2.10）的研发情况[14]，可以看出，Lustre 服务器 2.10 以下版本更新主要是已有功能的优化和扩充，而 Lustre 2.10 及以后版本引入了较多存储和配置方面的新功能，如 Progressive File Layouts，Multi-Rail LNet 等。

表 6　Lustre 研发路线图

Table 6　Lustre research and development

版本号	研发重点
Lustre 2.7	动态 LNET 配置，UID/GID 节点图
Lustre 2.8.0	LFSCK 阶段 4-性能调优，多个元数据 RPCs，DNE Phase Iib，Kerberos Revival
Lustre 2.9.0	文件系统默认的 OST 池，UID/GID 映射，子目录挂载，服务器端的建议和提示，大块 IO，共享密钥加密
Lustre 2.10	多轨 Lnet，TBF 策略优化，用户空间快照，文件布局改进，渐进的文件布局，项目配额，NRS 延迟策略，无补丁服务器
Lustre 2.12	Lazy size on MDT，配置“保存/编辑/还原”功能

关于系统版本升级对错误的影响，我们有以下观察：新功能的引入是错误大幅度增长的主要原因，运维人员对新功能的熟悉程度也对错误的发生有一定影响，如 Lustre 2.10.8 小版本更新，错误发生次数大幅度减少。另外，一个版本随着时间的迁移出现配置问题的可能性会大幅度减小。

发现 11：配置问题出现的频次主要由运维人员对系统的熟悉程度决定，性能问题和存储问题会随着版本迭代、新功能的引入大幅度增多，而当存储问题得到缓解后，性能问题可能会凸显出来。

发现 12：从一个大版本的生命周期来看，配置问题主要出现在前期，性能问题主要出现在中后期，存储问题一直存在，而且占比较高。

根据每次采集时长确定平均采集时长和平均 90% 错误记录间隔占比，综合它们可以计算得出所有版本平均 90% 错误记录间隔时长，表达式如下：

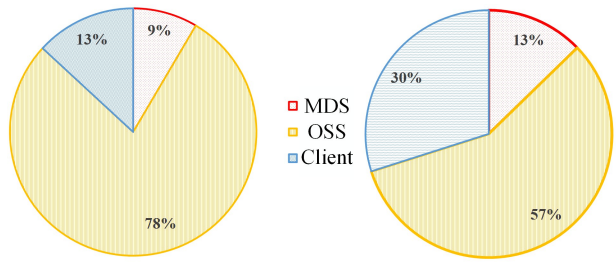
$$SUM_{\text{average\_collect\_time}} * SUM_{\text{average\_90\%\_error}} \tag{1}$$

其中，$SUM_{\text{average\_collect\_time}}$ 表示所有 Lustre 版本日志采集的平均总时长，$SUM_{\text{average\_90\%\_error}}$ 表示 90% 错误在相应日志采集时长下的平均占比。根据本文的数据，我们可以计算出 Lustre 集群的平均 90% 错误记录间隔时长为 71 天。

发现 13：根据我们采集到的错误记录数据，一个 Lustre 系统升级后，需要 71 天左右的时间来进行运维与优化，才能较好地确保其可用性与稳定性。

3.5　错误持续时间分布

我们对不同对象故障时长和平均故障时长进行统计发现，平均故障时长最短 0.78 h，最长 497.71 h。图 3 给出了不同对象累计故障时长和平均故障时长占比图，从图中可以看出，OSS 累计故障时长和平均故障时长占比最高。通过对错误持续时间数据进行分析发现，错误持续时间从 1 h 左右到 20 多天不等，错误发生后，并不一定马上给集群系统带来影响，相关维护人员可能并没有即刻得到反馈，给后续问题溯源带来了挑战。



(a) 累计故障时长　　(b) 平均故障时长

图 3　不同对象故障时长占比

Fig. 3　Proportion of error time of different subsystems

发现 14：不同子系统错误持续时间不一致，OSS 是错误时间持续最长的部件，需要持续关注。

4　Lustre 系统性能优化与实例检验

本节将讨论相关发现的意义、常见错误的解决方案、错误优化、发现检验与错误实例等。

4.1 常见错误分析与讨论

根据第3节的分析可知，Lustre系统在实际生产过程中出现频率较高的几类错误主要包括：zfs性能异常(zfs suspend)、quota设置问题、Client负载过重卡死、OSS recovery问题、硬件设置和网络配置问题。下文分别针对以上6种常见错误的前4种进行分析说明，并给出相关解决方案。

4.1.1 ZFS性能异常

在采集的错误记录中，Lustre系统的OSS端是用zfs作为OST后端存储。zfs拥有众多实用功能，可以很好地契合实际生产环境，如开源、COW事务模型、RAID、在线fsck数据错误检测、数据去重与数据压缩等。但在实际生产环境中，zfs会因为各种原因导致suspend。在我们的环境中，zfs主要有两个问题：1)一些默认设置不太合理，需要根据负载环境进行相应调整，如在采集到的错误记录中，99%的zpool错误故障是MMP(Massively Parallel Processing)导致的，默认MMP超时只有5s，如果单个设备写入失败，则会导致整个zfs suspend，需要根据实际负载重新设置MMP超时阈值；2)硬件适配问题，如OSS重启后，可能仅仅因为一个慢设备使得Multi-Host超时，导致zfs suspend，建议在使用新硬件前对硬盘提前进行简单测试，这样可以最大程度地减少不必要的错误。

4.1.2 配额设置问题

Lustre系统通过quota功能可以方便地实现对OST池的使用限制，达到应用隔离、减少资源争用、满足应用QoS的目的。但不合理的quota设置会导致诸多问题，如LDLM资源问题、系统与网络资源过度消耗等。由于问题多样，实质性解决方案几乎都要契合当时的生产环境，这里给出了一些缓解quota设置问题的建议：1)尽可能给予系统更多的资源；2)资源预留；3)采用双网模式(如以太网和IB网)进行非侵入式监控。

4.1.3 客户端卡死问题

client负载过重会导致卡死，该问题比较普遍，主要是因为client数量通常是服务器端数量的几十倍或上百倍，其故障率也相应上升。client卡死的主要原因是本地资源紧张，该问题的解决方案较为简单：1)采用更高性能的硬件设备或给予client更多的资源；2)构建多套Lustre，减少冲击和干扰。相比OSS，client问题相对简单，而且client对稳定性的要求不像服务器端那么严格，因此client可以采用新型存储技术(如PCC技术[15-16])来大幅度提升系统性能，减少client卡死问题。

4.1.4 对象服务器恢复问题

OSS会因为各种原因重启，重启后为了确保数据的完整性与一致性，需要对相关数据进行恢复。在OSS恢复时可能会遇到两个错误问题：1)zpool脑裂；2)数据丢失。脑裂问题已有相关解决方案，如引入第三个观察方并采用3阶段提交(Three-Phase Commit)。设备故障后，进行数据恢复的时间越长，数据丢失的风险就越大，建议采用如下策略确保数据安全：1)构建冗余阵列，周期性存储快照和多副本相结合；2)时常通过文件系统管理工具进行日常运维，如Lustre的lfsck工具、zfs的scrub命令等；3)采用资源预留与I/O重定向相结合的快速数据重构技术，缩短恢复时间，减小数据丢失概率。数据被误删，可以按照如下步骤尝试恢复：1)MDS挂载ldiskfs，然后用debugfs或者ext4 recovery恢复文件的元数据信息；2)在客户端用lfs getstripe获取相关文件的布局信息；3)对OST使用ext4修复工具进行修复；4)最后把需要恢复的对象恢复到OST指定目录。

4.2 配置错误问题

常见错误中的第五种硬件设置和第六种网络配置问题都可以被看作是配置问题。配置问题发生的频次主要由运维人员对当前系统版本的熟悉程度决定，因此提高运维人员的相关知识储备与实际操作水准可以减少配置问题。另外，还有一些其他方案：在安装构建系统时，为了方便后续管理与错误排查，记录各个硬件信息，用硬件信息代替常规默认集成方案，如在创建zpool时，使用硬盘编号来构建池；在集成到系统之前，对硬件进行简单的测试，并且同一个系统中采用相同批次与型号的硬件设施，以减少后续不同类型的错误；采用一体化运维管理工具来进行相关参数设置，减少由于运维人员疏忽而引起的配置错误问题；针对网络配置问题，建议采用PCC等技术来减少子系统之间的网络交互，从而达到问题隔离的目的。

4.3 对象存储服务器错误优化

通过对错误日志变化规律的探究，发现版本大更新是OSS错误的主要来源之一。存储相关功能的迭代更新相比新功能大规模引入对系统的稳定性更友好。OSS相关错误主要包括性能与存储两方面。性能问题主要是负载过重，无响应等；存储错误主要是存储服务不可用、数据丢失等。

性能问题的主要解决思路是减轻OSS端负载，我们推荐两种解决方案：1)PCC技术，PCC可以缓存单client读写文件到本地文件系统，并且可以删除OSS端相应的数据；2)Data on MDT(DOM)技术[17]，DOM可以提高小文件的读写性能，而且会大幅度减少client与OSS端的RPC与IO交互。

存储问题的主要解决思路是提供数据安全与管理。Lustre系统一般采用第三方加以保护，相关技术虽然能提供较好的数据保护功能，但没有与Lustre系统优化适配，可能会引起错误/性能问题，建议采用面向Lustre研发的数据保护技术来降低数据丢失的风险，以确保系统性能。现有的数据保护(Lustre 2.11版本)与管理方案有：File Level Redundancy(FLR)，Erasure Coding(EC)，Data Placement Policy(DPP)等。

4.4 发现检验与实例研究

本节将对Lustre错误相关发现进行检验，并给出一些减少错误、提升系统稳定性的具体实例。

4.4.1 验证数据与系统信息

我们新采集了同一套系统中错误节点及其关联节点的messages，dmesg和dklogs信息，采集时间段为2020-08-01到2021-06-30，共333天的错误日志信息。5个Lustre集群在硬件上无增删，最大程度地减少了由硬件不确定引起的系统错误，软件主要是操作系统和Lustre版本的迭代更新，更新信息如表7所列。

表 7　Lustre 集群版本更新信息

Table 7　Update information of Lustre cluster

版本	集群				
	L_A	L_B	L_C	L_D	L_E
Lustre ver	2.10.6－>2.12.6				2.10.8
Client ver	2.10.6－>2.12.6				
OS ver	CentOS 7.3－>7.9				CentOS 7.6

新采集的错误日志信息有如下特点：1）新采集的错误日志与第 3 节分析的错误日志一脉相承，时间上无缝对接，两者都来自于同一套系统；2）新采集的错误日志记录的时长、集群版本等信息足够提供验证支撑；3）新采集的错误日志包含 3 个同时运行的 Lustre 版本，有助于探究复杂环境下 Lustre 系统的错误，并验证发现是否正确。

4.4.2　发现检验

根据关键问题和错误发现的实质性内容对前文的 14 个发现进行了分类，表 8 列出了相关发现的分类。本文把相关发现分为 3 类：错误的一般特点、错误发生的规律性和错误与系统的稳定性。下文将根据新采集的错误记录对现有发现进行检验。

表 8　发现分类

Table 8　Classification of findings

发现分类	具体发现
错误的一般特点	发现 2、发现 5、发现 10、发现 14
错误发生的规律性	发现 3、发现 6、发现 11、发现 12
错误与系统的稳定性	发现 1、发现 4、发现 7、发现 8、发现 9、发现 13

由发现内容可知，错误的一般特点、发现 7 和发现 9 都是常识性认知，可以较容易验证其正确性。下面主要针对错误发生的规律性（发现 3、发现 6、发现 11、发现 12）和错误与系统稳定性（发现 1、发现 4、发现 8、发现 13）相关发现进行检验。

2021 年 1 月到 2021 年 6 月集群主要存在 3 个 Lustre 版本（lustre 2.10.8，lustre 2.12.6 和 lustre 2.12.0），根据采集到的错误数据，2021 年版本更新初期，配置错误发生比例较高，随着运维与优化，性能问题慢慢出现，而存储问题一直存在。另外，以前版本发生的某些错误重复出现，如 zfs 性能异常，并且有新的存储错误出现，如多台 OSS 同时重启、zfstxg sync hang 等。

表 9 列出了 2020 年和 2021 年新增错误记录信息。可以看出，2021 年 6 个月的时间错误发生了 46 次，而 2020 年 5 个月的时间仅发生了 8 次错误。错误发生的主要子系统是 OSS，并且其主要错误是配置错误，其次是存储错误。

表 9　2020 年和 2021 年新增错误日志记录信息

Table 9　New error log information in 2020 and 2021

年份	时间段	不同对象错误记录个数									错误总数
		MDS			OSS			Client			
		性能	存储	配置	性能	存储	配置	性能	存储	配置	
2020	25Aug—23 Dec	1			3	2	2				8
2021	18 Jan—28 Jun	1			1	11	16		4	13	46
单个问题合计		2	0	0	4	13	18	0	4	13	54
错误记录合计		2			35			17			

表 10 列出了不同版本新增错误记录信息，可以看出，新升级的 Lustre 2.12 版本（Lustre 2.12.0 和 Lustre2.12.6）错误发生总次数是 37 次，是 Lustre 2.10.8 版本错误发生次数（17 次）的 2.18 倍。

表 10　不同版本新增错误日志记录

Table 10　New error log of different versions

Lustre 版本	不同对象错误记录个数									总数
	MDS			OSS			Client			
	性能	存储	配置	性能	存储	配置	性能	存储	配置	
2.10.8	1			3	8	2		3		17
2.12.0					2	13			10	25
2.12.6	1			1	3	3		2	3	12

通过对新错误记录各表格的分析可以较好地验证错误发生的规律性（发现 3、发现 6、发现 11、发现 12），另外错误与系统的稳定性中发现 4 和发现 8 也可以得到印证。下面将主要针对发现 1 和发现 13 进行相关检验。

关于 MDS，OSS 和 Client 各自发生错误的次数，新增错误记录与发现 1 的不同之处是：新增错误记录中 OSS 发生错误次数最多，Client 发生错误次数次之，MDS 仅发生了两次性能问题；而发现 1 给出了 OSS 错误次数最多、MDS 发生错误次数次之、Client 发生错误次数最少也最稳定的推论。探究发现：新增错误记录 OSS 发生的错误大多数为配置错误，而且 Client 发生的错误几乎都与 OSS 相关，大部分都是版本不匹配引起的。另外，由表 6 研发路线可知，Lustre 2.12 仅是 MDS 常规性能优化，而 Lustre 2.10 主要针对 OSS 研发了许多新功能部件，结合表 12 可知，MDS 常规性能优化使其错误发生次数较少，而新功能研发使得 OSS 错误发生次数一直居高不下。

通过以上分析可知，新增错误记录中 Client 错误的发生次数比 MDS 错误的发生次数多的原因为：1）Client 发生的错误几乎都是与 OSS 相关联的错误，其自身表现稳定；2）MDS 在 Lustre 2.10 和 Lustre 2.12 版本中更新的功能是常规的优化提升，减少了错误发生的次数。因此，新功能研发使得 OSS 错误占比较高，Client 与 OSS 的关联错误提高了 Client 发生错误的概率，MDS 小范围的优化提升减小了 MDS 发生错误的概率，最终导致新增错误记录中 Client 发生错误的次数明显多于 MDS 发生错误的次数。如果仅考虑自身错误次数，新增错误记录的 Client 几乎没有发生错误，MDS 发生了两次性能问题，因此新增错误记录可以验证发现 1。

图 4 给出了新增错误累计分布的信息概况。根据每次的采集时长确定平均采集时长和平均 90%错误记录间隔占比，综合它们可以计算得出所有版本平均 90%错误记录间隔时长，由式（1）可以计算出新增错误记录的平均 90%错误记录间隔时长是 68 天。新增错误记录计算得出的运维与优化天数（68 天）与发现 13（71 天）相差不大，发现 13 可以得到相应验证。

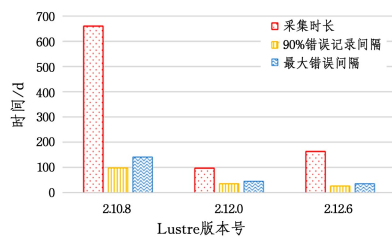


图 4　新增错误累计分布信息概况

Fig. 4　New Lustre version and error distribution

本小节对 14 个发现进行了检验，检验结果都能较好地支持本文的 14 个发现。下面将介绍通过减少错误次数来提升系统稳定性的具体实例。

4.4.3 错误实例研究

为了较好地探究和体现 Lustre 集群的错误规律，第 3 节的错误日志分析中仅把 zfs suspend 作为 zfs 性能异常统计过一次，但在实际采集到的数据中，zfs suspend 导致服务器重启的情况司空见惯。下面将对该问题进行详细探究，给出问题本质、解决方案与其效果。值得注意的是，2016 年初到 2018 年底，zfs suspend 导致服务器重启的情况几乎每两个星期发生一次，但直到 2019 年之后才开始尝试采用实质性的解决方案(硬盘固件升级与 zfs 默认参数调整)。下文将主要集中对 2019 年 1 月 1 日到 2021 年 6 月 30 日之间的 zfs suspend 问题进行探讨。

采集的错误日志中，zfs suspend 是一个常见的、重复性的错误，2019 年 1 月 1 日到 2021 年 6 月 30 日之间 zfs suspend 总共发生了 40 次，优化或解决 zfs suspend 问题十分必要。通过分析可知，产生 zfs suspend 的原因主要有两方面：1)读写延迟异常的设备；2)默认参数设置过于敏感。下面针对这两方面问题给出实例分析、测试与相关优化效果。

在实际生产环境中，服务器经常会因为单个设备不正常而触发 SCSI 重置，导致 I/O 暂停，进而导致 zfs suspend。设备读写带宽或延迟异常，这类异常设备通常被称为慢设备。图 5 给出了一个慢设备引发 zfs suspend 的实例。从实例中可以看出，慢设备的延迟可以高达 1 min 2.095 s，远远高于 zfs 默认设定的阈值(5 s)。慢设备在生产环境中比较常见，常规解决方案是固件升级。为了验证固件升级的有效性，对实际生产环境中的一块慢设备进行了固件升级(希捷 TT54－>希捷 TT56)，并对慢设备升级前后进行了 fio 测试。

```
# time sg_inq -d /dev/xxx
standard INQUIRY:
  PQual=0 Device_type=0 RMB=0 version=0x06 [SPC-4]
  [AERC=0] [TrmTsk=0] NormACA=0 HiSUP=1 Resp_data_format=2
  SCCS=0 ACC=0 TPGS=0 3PC=0 Protect=1 [BQue=0]
  EncServ=0 MultiP=0 [MChngr=0] [ACKREQQ=0] Addr16=0
  [RelAdr=0] WBus16=0 Sync=0 Linked=0 [TranDis=0] CmdQue=1
  [SPI:Clocking=0x0 QAS=0 IUS=0]
    length=144 (0x90) Peripheral device type:disk
Vendor identification:SMC
Product identification:SMC2108
Product revision level:2.12
Unit serial number:0082a75c147479702200da7806800403
        inquiry:pass-through requested 252 bytes but got 12 bytes

real    1m2.095s
user    0m0.000s
sys     0m0.001s
WARNING:MMP writes to pool 'xxx' have not succeeded in over 159s; suspending pool
WARNING:Pool 'xxx' has encountered an uncorrectable I/O failure and has been suspended.
```

图 5 慢设备引发 zfs suspend 实例

Fig. 5 Example of slow device triggers zfs suspend

图 6 给出了一个慢设备升级前后和一个正常设备的 fio 读延迟。从图中可以看出，sdb 表现正常，最高延迟仅 0.11 s，升级前慢设备 sda1 读延迟表现异常，延迟高达 2.82 s，升级后慢设备 sda2 读延迟 2.24 s。虽然慢设备固件升级后读延迟有所下降，但相比正常设备，读延迟还是偏高。以上测试表明，固件升级难以较好地解决慢设备问题。

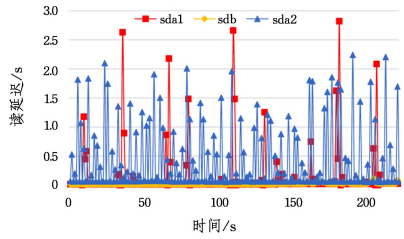


图 6 慢设备固件升级前后读延迟测试

Fig. 6 Read latency of slow device before and after firmware upgrade

从以上的实例可以看出，硬件优化并不能很好地解决 zfs suspend 问题，这促使我们对软件相关优化进行探究。经过对 zfs suspend 错误溯源与 openzfs 源码分析发现，问题的实质是 MMP 默认参数设置对延迟写入过于敏感，并且 MMP 超时阈值是固定值，没有考虑当前系统的 I/O 负载。通过分析 openzfs 源码可知，zfs_multihost_interval 的默认值是 1 000 ms，而 zfs_multihost_fail_intervals 的默认设置是 5。如果系统在 zfs_multihost_fail_intervals * zfs_multihost_interval ms 未成功执行一个 mmp write，则会引发 zpool suspend。在实际生产环境中，服务器 block dev 访问通常需要 30 s，5 s 默认设置并不合适。基于以上分析，适当增大 zpool 的超时默认值有助于减少不合理的 zpool suspend。

图 7 给出了 2019 年 1 月 1 日到 2021 年 6 月 30 日 Lustre 集群发生 zfs suspend 错误次数的累计图，其中黄、红、绿 3 条竖线分别给出了 zfs_multihost_interval 参数设置为 10000、硬件固件升级(希捷 TT54－>希捷 TT56)和 zfs_multihost_interval 参数设置为 17000 的时间点。

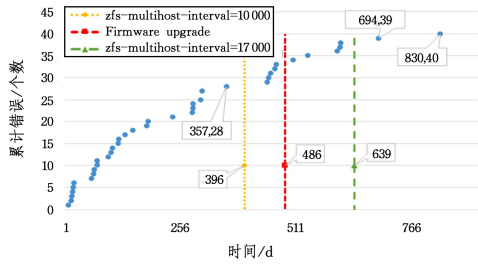


图 7 zfs suspend 错误累计与 zfs 参数设置(电子版为彩图)

Fig. 7 Error cumulation of zfs suspend and zfs parameter setting

2019 年 1 月 1 日到 2020 年 2 月 1 日之间，zfs_multihost_interval 参数是默认参数 1 000，固件是希捷 TT54，这期间集群发生 zfs suspend 的次数是 28；2020 年 2 月 1 日到 2020 年 10 月 1 日之间，根据对慢设备的测试，把 zfs_multihost_interval 的默认参数值修改为 10000，并且在 2020 年 5 月 1 日对慢设备的固件进行了升级(希捷 TT54－>希捷 TT56)，这期间集群发生 zfs suspend 的次数是 10，参数修改后到慢设备升级

前集群发生 zfs suspend 的次数是 5，慢设备升级后到参数再次修改前，集群发生 zfs suspend 的次数也是 5，可以看出参数设置与固件升级都有助于减少 zfs suspend 发生的次数；2020 年 10 月 1 日到 2021 年 6 月 30 日之间，根据对慢设备的测试和实际负载预估，把 zfs_multihost_interval 的默认参数值修改为 17 000，修改后，这期间集群发生 zfs suspend 的次数是 2，zfs suspend 的发生次数已大幅度减少。

通过以上实际测试可以看出，固件升级与参数设置对 zfs suspend 错误有较大影响，通过固件升级缩短硬盘时延或重调 zfs 相关参数以增加超时阈值，都是减少 zfs suspend、确保系统可维护性、提升系统稳定性的有效且简便的优化方法。

结束语　针对集群存储系统可用性和稳定性等问题，采集了实际应用场景下，同一生产环境中 5 个 Lustre 集群存储系统 1 673 天的错误日志信息，在确保错误日志数据普遍性、代表性、完整性与正确性的基础上，根据错误特性，将错误日志分类，归纳了常见错误与其解决方案，并在时间和空间维度对错误的相关规律与特性进行了探索分析，探究了 7 个关键的待探讨问题。通过对相关数据的分析，总结了 14 个发现。另外，新采集了 333 天新增错误日志信息对 14 个发现进行了检验，并给出了错误优化实例。相关测试表明，优化实例与相关方案可以显著减少错误发生的次数，确保集群存储系统的可维护性，提升系统稳定性。

参 考 文 献

[1] RASHAWN L K, TATYANA M, SUPADA L, et al. Preeti, Valgrind, AVX-512, and Intel HPC Analysis Tools[OL]. https://slidetodoc.com/valgrind-avx512-and-intel-hpc-analysis-tools-7/.

[2] AGHAYEV A, WEIL S, KUCHNIK M, et al. File systems unfit as distributed storage backends: lessons from 10 years of Ceph evolution[C]// The 27th ACM Symposium on Operating Systems Principles. 2019: 353-369.

[3] DAI D, GATLA O R, ZHENG M. A Performance Study of Lustre File System Checker: Bottlenecks and Potentials[C]// The 35th Symposium on Mass Storage Systems and Technologies (MSST). IEEE, 2019: 7-13.

[4] CIORBA F M. The importance and need for system monitoring and analysis in HPC operations and research[J]. arXiv: 1807.03112, 2018.

[5] BRAAM P. The Lustre storage architecture[J]. arXiv: 1903.01955. 2019.

[6] Lustre[OL]. https://www.lustre.org/.

[7] BRIM M J, LOTHIAN J K. Monitoring extreme-scale Lustre toolkit[OL]. http://arxiv.org/abs/1504.06836.

[8] LOCKWOOD G K, SNYDER S, WANG T, et al. A Year in the Life of a Parallel File System[C]// Proceedings of the International Conference for High Performance Computing. 2018.

[9] JI C, CHANG L P, SHI L, et al. An empirical study of file-system fragmentation in mobile storage systems[C]// The 8th {USENIX} Workshop on Hot Topics in Storage and File Systems (HotStorage 16). 2016.

[10] AGHAYEV A, WEIL S, KUCHNIK M, et al. File systems unfit as distributed storage backends: lessons from 10 years of Ceph evolution[C]// The 27th ACM Symposium on Operating Systems Principles. 2019: 353-369.

[11] AGRAWAL N, BOLOSKY W J, DOUCEUR J R, et al. A five-year study of file-system metadata[J]. ACM Transactions on Storage (TOS), 2007, 3(3): 9: 1-9: 32.

[12] LU L, ARPACI-DUSSEAU A C, ARPACI-DUSSEAU R H, et al. A study of Linux file system evolution[C]// The Presented as part of the 11th {USENIX} Conference on File and Storage Technologies ({FAST} 13). 2013: 31-44.

[13] LU L, ARPACI-DUSSEAU A C, ARPACI-DUSSEAU R H, et al. A study of Linux file system evolution[J]. ACM Transactions on Storage (TOS), 2014, 10(1): 1-32.

[14] Lustre Projects[OL]. https://wiki.lustre.org/Projects.

[15] QIAN Y J, LI X, IHARA S, et al. LPCC: Hierarchical Persistent Client Caching for Lustre[C]// Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. New York, NY, USA, 2019, 88: 1-14.

[16] CHENG W, LI C Y, ZENG L F, et al. NVMM-oriented Hierarchical Persistent Client Caching for Lustre[J]. ACM Transactions on Storage (TOS), 2021, 17(1): 1-22.

[17] Lustre. Data on MDT Solution Architecture[OL]. http://wiki.lustre.org/Data_on_MDT_Solution_Architecture.

CHENG Wen, born in 1989, Ph.D candidate, is a student member of China Computer Federation. His main research interests include distributed storage system, data mining, system optimization, and data privacy protection.

WANG Fang, born in 1972. Ph.D, professor, Ph.D supervisor, is a senior member of China Computer Federation. Her main research interests include distributed storage system, parallel file system, new storage system based on non-volatile memory devices, software defined storage, and large-scale graph data storage and processing.

（责任编辑：喻藜）