2018 IEEE International Conference on Cluster Computing

GekkoFS—用于HPC应用程序的临时分布式文件系统

短论文

Marc-Andr´e Vef*, Nafiseh Moti*, Tim¨u*, Tommaso Tocci†, Ramon Nou†, Alberto Miranda†, Toni Cortes†§, Andr´e Brinkmann*

*约翰内斯古腾堡大学美因茨，美因茨,德国†巴塞罗那超级计算中心，巴塞罗那，西班牙§加泰罗尼亚理工大学，巴塞罗那，西班牙

*摘要:我们*提出了*GekkoFS*，一个临时的，高度可扩展的突发缓冲文件系统，专门针对数据密集型高性能计算(HPC)应用程序的新访问模式进行了优化。文件系统提供宽松的POSIX语义，只提供大多数(不是全部)应用程序实际需要的特性。它能够提供可扩展的I/O性能，并且已经为少量节点达到了数百万元数据操作，大大超过了通用并行文件系统的能力。

*索引术语-分布式*文件系统，HPC，突发缓冲区

I. 介绍

高性能计算(HPC)应用正在发生重大变化。传统的HPC应用已经被计算绑定，大规模的模拟，而今天的HPC社区正朝着生成、处理和分析大量实验数据的方向发展。这种被称为*数据驱动科学*的趋势正在影响许多不同的科学领域，其中一些领域由于新开发的技术[15]、[30]在解决以前无法解决的挑战方面取得了重大进展。

大多数数据驱动的工作负载都基于新的算法和数据结构，如图数据库，这对HPC文件系统提出了新的要求[22]，[39]。它们包括，例如，大量的元数据操作，数据同步，非连续和随机访问模式，以及小的I/O请求[9]，[22]。这些操作与过去的工作负载有很大的不同，过去的工作负载主要是在大文件上执行顺序I/O操作。它们不仅降低了数据驱动应用程序本身的速度，而且还会严重干扰正在并发访问共享存储系统[11]、[35]的其他应用程序。因此，传统的并行文件系统(PFS)不能有效地处理这些工作负载，数据驱动的应用程序会遭受I/O延迟延长、吞吐量降低和等待时间过长的问题。

基于软件的方法，例如，应用程序修改或中间件和高级库[12]，[19]，试图支持数据驱动的应用程序，以使新的访问模式与底层PFS的功能保持一致。然而，适应这样的软件通常是耗时的，很难与大数据和机器学习库相结合，或者有时(基于底层算法)只是不可能。

基于硬件的方法从磁盘(pfs的主要后端技术)转向基于nand的固态驱动器(ssd)。如今，许多超级计算机部署的ssd可以用作专用突发缓冲区[18]或*节点本地*突发缓冲区。为了获得较高的元数据性能，它们可以与*动态突发缓冲*文件系统[3]、[40]结合部署。

通常，与PFS相比，突发缓冲区文件系统在不修改应用程序的情况下提高了性能。因此，它们通常支持POSIX，它提供了大多数应用程序开发人员接受的标准语义。然而，强制执行POSIX会严重降低PFS的峰值性能[38]。此外，大多数科学应用程序并不需要许多POSIX特性[17]，特别是当它们可以独占访问文件系统时。类似的论点适用于其他高级功能，如容错或安全性。

在这项工作中，我们提出了*GekkoFS*，这是一种用于HPC应用程序的临时部署，高度可扩展的分布式文件系统，旨在加速对现代pfs具有挑战性的常见HPC工作负载的I/O操作。GekkoFS汇集了快速的节点本地存储资源，并提供了所有参与节点都可以访问的全局命名空间。它通过删除一些在分布式上下文中最损害I/O性能的语义来放松POSIX，并考虑了先前对HPC应用程序行为的研究[17]，以优化最常用的文件系统操作。

为了实现负载均衡，使用HPC RPC框架*Mercury*将所有数据和元数据分布在所有节点上[34]。该文件系统在用户空间中运行，任何用户都可以在20秒内轻松地在512节点的集群上部署。因此，它可以在许多临时场景中使用，例如，在一个计算作业的生命周期中，或在较长期的用例中，例如活动中。我们演示了轻量级的、高度分布式的文件系统GekkoFS如何在512节点集群上以每秒数千万次的元数据操作达到可扩展的数据和元数据性能，同时仍然为针对特定文件或目录的文件系统操作提供强大的一致性。

II. 相关工作

通用pfs，如GPFS、Lustre、BeeGFS或PVFS[4]、[14]、[27]、[31]、[32]提供长期存储

2168-9253/18/$31. 00 ©2018 IEEE
DOI 10.1109/CLUSTER.2018.00049

主要基于磁盘。相反，GekkoFS从快速节点本地ssd构建了一个短期的、独立的名称空间，该名称空间仅在作业或活动运行期间临时访问。因此，GekkoFS可以归类为节点本*地突发缓冲*文件系统，而*远程共享突发缓冲*文件系统使用专用的集中式I/O节点[40]，例如DDN的IME[1]。

一般来说，节点本地突发缓冲区是快速的中间存储系统，旨在减少PFS的负载和应用程序的I/O开销[18]。它们通常与运行计算作业的节点并置，但它们也可以依赖于后端PFS[3]，或者在某些情况下甚至由它直接管理[24]。BurstFS[40]可能是与我们最相关的工作，它是一个独立的突发缓冲文件系统，但与GekkoFS不同的是，它仅限于本地写入数据。BeeOND[14]可以在多个节点上创建与GekkoFS类似的作业临时文件系统。然而，与我们的文件系统相比，它是POSIX兼容的，我们的测量显示，它比BeeOND提供的元数据吞吐量要高得多[36]。

inode和相关目录块的管理是分布式环境中文件系统的主要可扩展性限制。通常，通用pfs将数据分布在所有可用的存储目标上。由于这种技术对数据很有效，它在处理元数据[5]、[28]时不能实现相同的吞吐量，尽管文件系统社区提出了各种技术来解决这个挑战[3]、[13]、[25]、[26]、[41]、[42]。性能限制可以归因于底层POSIX语义强制的串行化，当从多个进程在单个目录中创建大量文件时，这尤其会降低吞吐量。这种工作负载在HPC环境中很常见[3]，[24]，[25]，[37]，对于即将到来的数据科学应用来说可能会成为更大的挑战。GekkoFS建立在一种处理目录和用对象替换目录条目的新技术之上，对象存储在一个强一致的键值存储中，这有助于为数十亿个文件实现数千万个元数据操作。

III. 设计与实现

GekkoFS为特定用例的生命周期提供了一个用户空间文件系统，例如，在HPC作业的上下文中。该文件系统利用计算节点的可用本地存储来分发数据和元数据，并将它们的节点本地存储组合成一个单一的全局命名空间。

文件系统的主要目标集中在可扩展性和一致性上。因此，它应该扩展到任意数量的节点，以受益于当前和未来的存储和网络技术。此外，对于访问特定数据文件的文件系统操作，GekkoFS应该提供与POSIX相同的一致性。然而，例如，目录操作的一致性可以放宽。最后，GekkoFS应该是硬件独立的，以便有效地使用当今的网络技术以及用户可以访问的任何现代和未来的存储硬件。

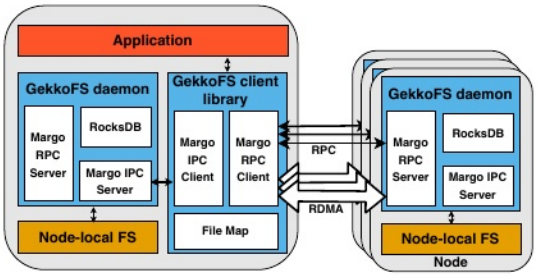

图1:GekkoFS架构

A. POSIX弛豫

与PVFS[6]和OrangeFS[21]类似，GekkoFS不提供复杂的全局锁定机制。从这个意义上说，应用程序应该负责确保不发生冲突，特别是w.r.t.重叠的文件区域。然而，缺乏分布式锁会对受影响的文件系统对象的数量事先未知的操作产生影响，例如ls -l命令调用的readdir()。在这*些间接文件系统操作中*，GekkoFS不保证返回目录的当前状态，并遵循最终一致性模型。此外，每个文件系统操作都是同步的，没有任何形式的缓存，以降低文件系统的复杂性，并允许对其原始性能能力进行评估。

GekkoFS不支持移动或重命名操作或链接功能，因为HPC应用研究表明，在并行作业的执行过程中很少或根本不使用这些功能[17]。最后，GekkoFS不维护访问权限形式的安全管理，因为它已经隐式地遵循节点本地文件系统的安全协议。

B. 架构

GekkoFS的体系结构(参见图1)由两个主要组件组成:客户机库和服务器进程。使用GekkoFS的应用程序必须首先预加载客户端插入库，该库拦截所有文件系统操作，并在必要时将其转发给服务器(GekkoFS*守护进程*)。运行在每个文件系统节点上的GekkoFS守护进程接收来自客户机的转发的文件系统操作，并独立地处理它们，完成后发送响应。在接下来的段落中，我们将更详细地描述客户端和守护进程。

*a) GekkoFS客户端:*客户端由三个部分组成:1)拦截接口，捕获对GekkoFS的相关调用，并将不相关的调用转发到节点本地文件系统;2)一个文件映射，管理打开文件和目录的文件描述符，独立于内核;3)一个基于rpc的通信层，它将文件系统请求转发给本地/远程GekkoFS守护进程。

每个文件系统操作都通过RPC消息转发到特定的守护进程(由文件路径的散列确定)，并在此直接执行。换句话说，就是GekkoFS

使用伪随机分布在所有节点上传播数据和元数据，也称为*宽条纹*。由于每个客户机都能够独立地解析文件系统操作的负责节点，因此GekkoFS不需要跟踪元数据或数据所在位置的中央数据结构。为了实现大文件的均衡数据分布，数据请求在分布到各个文件系统节点之前，会被分割成大小相等的块。如果得到底层网络结构协议的支持，客户端将相关的块内存区域暴露给守护进程，通过*远程直接内存访问(*RDMA)进行访问。

b) GekkoFS守护进程:GekkoFS守护进程由三部分组成:1)用于存储元数据的键值存储(KV存储);2)从/向底层本地存储系统读写数据的I/O持久化层(每块一个文件);3)一个基于rpc的通信层，它接受本地和远程连接来处理文件系统操作。

每个守护进程运行单个本地RocksDB KV存储[10]。RocksDB针对NAND存储技术进行了优化，具有低延迟，适合GekkoFS的需求，因为ssd主要用作当今HPC集群中的节点本地存储。

对于通信层，我们利用了*Mercury* RPC框架[34]。它允许GekkoFS独立于网络，并在文件系统内有效地传输大数据。在GekkoFS中，水星是通过*Margo*库间接接口的，该库为水星的API提供了*argobots感知*包装器，目的是提供一个简单的多线程执行模型[7]，[33]。使用Margo允许GekkoFS守护进程最大限度地减少Margo进程线程和接受和处理RPC请求的处理程序的资源消耗[7]。

IV. 评价

我们基于各种未经修改的微基准测试评估了GekkoFS的性能，这些微基准测试捕获了HPC应用中常见的访问模式。我们的实验是在位于德国美因茨约翰内斯·古腾堡大学的*MOGON II*超级计算机上进行的。所有实验均在Intel 2630v4 Intel Broadwell处理器(每个处理器两个插槽)上进行。节点内主存容量为64 GiB ~ 512 GiB。MOGON II采用100gbit /s Intel Omni-Path，在所有计算节点之间建立胖树网络。此外，每个节点提供一个数据中心Intel SATA SSD DC S3700系列，作为计算作业中可用的划痕空间(XFS格式)。我们使用这些ssd来存储GekkoFS的数据和元数据，它使用512kib的内部块大小。

在每次实验迭代之前，重新启动GekkoFS守护进程(对于512个节点需要不到20秒的时间)，删除所有SSD内容，并刷新内核缓冲区、inode和dentry缓存。GekkoFS守护进程和被测试的应用程序被固定在单独的处理器套接字上，以确保文件系统和应用程序不会相互干扰。

A. 元数据的性能

我们使用未经修改的*mdtest*微基准[20]模拟了常见的元数据密集型HPC工作负载，以评估GekkoFS的元数据性能，并将其与Lustre并行文件系统进行比较。虽然GekkoFS和Lustre具有不同的目标，但我们指出了使用GekkoFS作为突发缓冲文件系统可以获得的性能。在我们的实验中，mdtest在单个目录中并行执行*创建*、*stat*和*删除*操作——这是许多HPC应用程序中的重要工作负载，也是通用PFS最困难的工作负载之一[37]。

GekkoFS上的每个操作在每个进程(每个节点16个进程)中使用100,000个零字节文件执行。从用户应用程序的角度来看，所有创建的文件都存储在单个目录中。然而，由于GekkoFS内部保持平面命名空间，因此创建的目录文件在概念上没有区别。这与传统的PFS形成了对比，如果工作负载分布在多个目录中，而不是在单个目录中，那么传统PFS的性能可能会更好。图2比较了GekkoFS和Lustre在多达512个节点的三种场景中的情况:文件创建、文件统计和文件删除。y轴描述了在对数尺度上特定工作负载每秒实现的相应操作。每个实验至少运行5次，每个数据点代表所有迭代的平均值。GekkoFS的工作负载扩展为每个进程10万个文件，而Lustre的工作负载固定为所有实验的400万个文件。我们固定了Lustre元数据实验的文件数量，因为Lustre在扩展到太多文件时检测到挂起的节点。

Lustre实验在两种配置下运行:所有进程在单个目录(单个目录)中运行，或者每个进程在自己的目录(唯一目录)中运行。此外，在系统可被其他应用程序访问的情况下，对Lustre的元数据性能进行了评估。

如图2所示，GekkoFS在所有场景中都大大优于Lustre，并且显示出接近线性扩展，无论Lustre进程是在单个目录中还是在孤立的目录中运行。与Lustre相比，GekkoFS在512个节点上实现了大约4600万次创建/秒(~ 1405倍)，4400万次统计/秒(~ 3559倍)和2200万次移除/秒(~453x)。标准偏差小于3.5%，这是作为均值的百分比计算的。

B. 数据性能

我们使用未修改的*IOR*[20]微基准来评估GekkoFS在两种场景下的顺序和随机访问模式的I/O性能:每个进程访问自己的文件(每进程一个文件)，所有进程访问单个文件(共享文件)。我们使用8 KiB、64 KiB、1 MiB和64 MiB*传输大小*来评估许多小型I/O访问和少数大型I/O请求的性能。我们在每个客户机上运行16个进程，每个进程总共写入和读取4 GiB。

GekkoFS数据性能不与Lustre临时文件系统进行比较，因为在≤*10*个节点的情况下，已使用的Lustre分区的峰值性能已经达到12 GiB/s左右

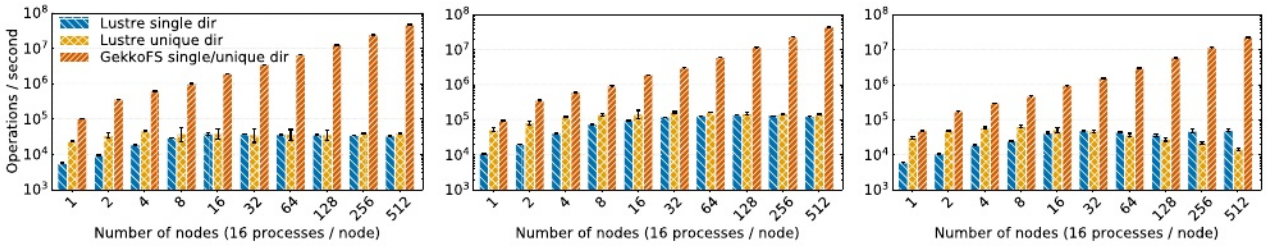

图2:与Lustre文件系统相比，GekkoFS的文件创建、统计和删除吞吐量在节点数量不断增加的情况下。

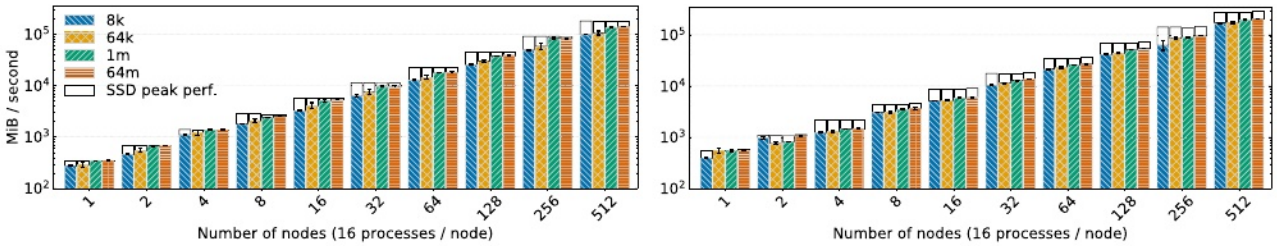

图3:与普通SSD峰值吞吐量相比，GekkoFS的每个进程在其自己的文件上操作的顺序吞吐量。

对于顺序I/O模式。此外，Lustre在更大规模的部署中呈线性扩展，可用的oss和ost更多[23]。

图3显示了GekkoFS的顺序I/O吞吐量(以MiB/s为单位)，代表了不同传输大小的节点数量不断增加时至少5次迭代的平均值。此外，还将每个数据点与所有聚合的SSD在给定节点配置下可以提供的峰值性能进行比较，显示为白色矩形，表示GekkoFS的SSD使用效率。总的来说，每个结果都表明GekkoFS接近线性可扩展性，在512个节点的传输大小为64 MiB的情况下，写和读操作可实现约141 GiB/s(约占聚合SSD峰值带宽的80%)和204 GiB/s(约占聚合SSD峰值带宽的70%)。在512个节点时，这意味着超过1300万的写IOPS和超过2200万的读IOPS，而对于传输大小为8 KiB的文件系统操作，平均延迟最多可以限制在700 μs。

对于每个进程文件的情况，对于大于文件系统块大小的传输大小，顺序和随机访问I/O吞吐量是相似的。这是由于传输大小大于块大小内部访问整个块文件，而较小的传输大小以随机偏移访问一个块。因此，大传输大小的随机访问在概念上与顺序访问是相同的。对于较小的传输大小，例如8 KiB，由于对块内位置的随机访问，512个节点的随机写和读吞吐量分别下降了大约33%和60%。

对于共享文件的情况，GekkoFS的同步和无缓存设计的缺点变得明显。实现了每秒不超过150K次的写操作。这是由于守护进程上的网络竞争，该守护进程维护共享文件的元数据，其大小需要不断更新。为了克服这个限制，我们添加了一个基本的客户端缓存，在一些写操作被发送到管理文件元数据的节点之前，本地缓存大小的更新。因此，用于顺序和随机访问的共享文件I/O吞吐量类似于每个进程的文件性能，因为守护进程上的块管理在这两种情况下在概念上是无关的。

V. 结论和致谢

我们介绍并评估了GekkoFS，这是一种新的突发缓冲文件系统，用于HPC应用程序，具有宽松的posix语义，允许它在少量节点上实现数百万元数据操作，并且在各种数据和元数据用例中接近线性可扩展性。接下来，我们计划从三个方向扩展GekkoFS:研究不同块大小的GekkoFS，评估缓存的好处，以及探索不同的数据分布模式。

这项工作由德国研究基金会通过ADA-FS项目资助，作为1648年优先计划的一部分。它还得到了西班牙科学与创新部(TIN2015 - 65316)、加泰罗尼亚政府(2014-SGR-1051)、欧盟地平线2020研究与创新计划(NEXTGenIO, 671951)和欧盟委员会大存储项目(h2020 - mca - itn -2014-642963)的支持。这项研究是利用美因茨约翰内斯·古腾堡大学提供的超级计算机MOGON II和服务进行的。

参考文献。

[1] Infinite Memory Engine. https://www. ddn. com/products/ime-flash-native-data-cache.

[2] The Open Group Base Specifications Issue 7(IEEE Std 1003.1-2008). http://pubs.opengroup.org/onlinepubs/9699919799/.

[3] J. Bent, G. A. Gibson, G. Grider, B. McClelland, P. Nowoczynski, J. Nunez, M. Polte, and M. Wingate, "PLFS: a checkpoint filesystem for parallel applications," in *Proceedings of the ACM/IEEE Conference on High Performance Computing (SC), November 14-20, Portland, Oregon, USA*, 2009.

[4] P. J. Braam and P. Schwan, "Lustre: The intergalactic file system," in *Ottawa Linux Symposium*, 2002, p. 50.

[5] P. Carns, Y. Yao, K. Harms, R. Latham, R. Ross, and K. Antypas, "Production i/o characterization on the cray xe6," in *Proceedings of the Cray User Group meeting*, vol. 2013, 2013.

[6] P. H. Carns, W. B. L. III, R. B. Ross, and R. Thakur, "PVFS: A parallel file system for linux clusters," in *4th Annual Linux Showcase & Conference 2000, Atlanta, Georgia, USA, October 10-14, 2000*, 2000.

[7] P. H. Carns, J. Jenkins, C. D. Cranor, S. Atchley, S. Seo, S. Snyder, and R. B. Ross, "Enabling NVM for data-intensive scientific services," in *4th Workshop on Interactions of NVM/Flash with Operating Systems and Workloads, INFLOW@OSDI 2016, Savannah, GA, USA, November 1, 2016.*, 2016.

[8] A. Choudhary, W.-k. Liao, K. Gao, A. Nisar, R. Ross, R. Thakur, and R. Latham, "Scalable i/o and analytics," in *Journal of Physics: Conference Series*, vol. 180, no. 1, 2009, p. 012048.

[9] P. Crandall, R. A. Aydt, A. A. Chien, and D. A. Reed, "Input/output characteristics of scalable parallel applications," in *Proceedings Supercomputing '95, San Diego, CA, USA, December 4-8, 1995*, 1995, p. 59.

[10] S. Dong, M. Callaghan, L. Galanis, D. Borthakur, T. Savor, and M. Strum, "Optimizing space amplification in rocksdb," in *CIDR 2017, 8th Biennial Conference on Innovative Data Systems Research, Chami-nade, CA, USA, January 8-11, 2017, Online Proceedings*, 2017.

[11] M. Dorier, G. Antoniu, R. B. Ross, D. Kimpe, and S. Ibrahim, "Calciom: Mitigating I/O interference in HPC systems through cross-application coordination," in *2014 IEEE 28th International Parallel and Distributed Processing Symposium, Phoenix, AZ, USA, May 19-23, 2014*, 2014, pp. 155–164.

[12] M. Folk, A. Cheng, and K. Yates, "Hdf5: A file format and i/o library for high performance computing applications," in *Proceedings of supercomputing*, vol. 99, 1999, pp. 5–33.

[13] W. Frings, F. Wolf, and V. Petkov, "Scalable massively parallel I/O to task-local files," in *Proceedings of the ACM/IEEE Conference on High Performance Computing (SC), November 14-20, Portland, Oregon, USA*, 2009.

[14] F. Herold, S. Breuner, and J. Heichler, "An introduction to beegfs," 2014, https://www.beegfs.io/docs/whitepapers/Introduction to BeeGFS by ThinkParQ.pdf.

[15] T. Hey, S. Tansley, and K. M. Tolle, Eds., *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research, 2009.

[16] R. Latham, R. B. Ross, and R. Thakur, "The impact of file systems on MPI-IO scalability," in *Recent Advances in Parallel Virtual Machine and Message Passing Interface, 11th European PVM/MPI Users' Group Meeting, Budapest, Hungary, September 19-22, 2004, Proceedings*, 2004, pp. 87–96.

[17] P. H. Lensing, T. Cortes, J. Hughes, and A. Brinkmann, "File system scalability with highly decentralized metadata on independent storage devices," in *IEEE/ACM 16th International Symposium on Cluster, Cloud and Grid Computing (CCGrid), Cartagena, Colombia, May 16-19*, 2016, pp. 366–375.

[18] N. Liu, J. Cope, P. H. Carns, C. D. Carothers, R. B. Ross, G. Grider, A. Crume, and C. Maltzahn, "On the role of burst buffers in leadership-class storage systems," in *IEEE 28th Symposium on Mass Storage Systems and Technologies, MSST 2012, April 16-20, 2012, Asilomar Conference Grounds, Pacific Grove, CA, USA*, 2012, pp. 1–11.

[19] J. F. Lofstead, S. Klasky, K. Schwan, N. Podhorszki, and C. Jin, "Flexible IO and integration for scientific codes through the adaptable IO system (ADIOS)," in *6th International Workshop on Challenges of Large Applications in Distributed Environments, CLADE@HPDC 2008, Boston, MA, USA, June 23, 2008*, 2008, pp. 15–24.

[20] "Mdtest metadata benchmark and ior data benchmark," 2018, https://github.com/hpc/ior.

[21] M. Moore, D. Bonnie, B. Ligon, M. Marshall, W. Ligon, N. Mills, E. Quarles, S. Sampson, S. Yang, and B. Wilson, "Orangefs: Advancing pvfs," *FAST poster session*, 2011.

[22] N. Nieuwejaar, D. Kotz, A. Purakayastha, C. S. Ellis, and M. L. Best, "File-access characteristics of parallel scientific workloads," *IEEE Trans. Parallel Distrib. Syst.*, vol. 7, no. 10, pp. 1075–1089, 1996.

[23] S. Oral, D. A. Dillow, D. Fuller, J. Hill, D. Leverman, S. S. Vazhkudai, F. Wang, Y. K. , J. Rogers, J. James Simmons, and R. Miller, "Olcfs 1 tb/s, next-generation lustre file system," in *Proceedings of Cray User Group Conference (CUG 2013)*, 2013.

[24] S. Oral and G. Shah, "Spectrum scale enhancements for coral. presentation slides at supercomputing 16." 2016, http://files.gpfsug.org/presentations/2016/SC16/11 Sarp Oral Gautam Shah Spectrum Scale Enhancements for CORAL v2.pdf.

[25] S. Patil and G. A. Gibson, "Scale and concurrency of GIGA+: file system directories with millions of files," in *9th USENIX Conference on File and Storage Technologies, San Jose, CA, USA, February 15-17, 2011*, 2011, pp. 177–190.

[26] S. Patil, K. Ren, and G. Gibson, "A case for scaling HPC metadata performance through de-specialization," in *2012 SC Companion: High Performance Computing, Networking Storage and Analysis, Salt Lake City, UT, USA, November 10-16, 2012*, 2012, pp. 30–35.

[27] Y. Qian, X. Li, S. Ihara, L. Zeng, J. Kaiser, T. S¨uß, and A. Brinkmann, "A configurable rule based classful token bucket filter network request scheduler for the lustre file system," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC), Denver, CO, USA, November 12 - 17*, 2017, pp. 6:1–6:12.

[28] K. Ren, Q. Zheng, S. Patil, and G. A. Gibson, "Indexfs: Scaling file system metadata performance with stateless caching and bulk insertion," in *International Conference for High Performance Computing, Network-ing, Storage and Analysis, SC 2014, New Orleans, LA, USA, November 16-21, 2014*, 2014, pp. 237–248.

[29] D. Ritchie and K. Thompson, "The UNIX time-sharing system (reprint)," *Commun. ACM*, vol. 26, no. 1, pp. 84–89, 1983.

[30] R. Ross, R. Thakur, and A. Choudhary, "Achievements and challenges for i/o in computational science," in *Journal of Physics: Conference Series*, vol. 16, no. 1, 2005, p. 501.

[31] R. B. Ross and R. Latham, "PVFS - PVFS: a parallel file system," in *Proceedings of the ACM/IEEE SC2006 Conference on High Performance Networking and Computing, November 11-17, 2006, Tampa, FL, USA*, 2006, p. 34.

[32] F. B. Schmuck and R. L. Haskin, "GPFS: A shared-disk file system for large computing clusters," in *Proceedings of the FAST '02 Conference on File and Storage Technologies, January 28-30, Monterey, California, USA*, 2002, pp. 231–244.

[33] S. Seo, A. Amer, P. Balaji, C. Bordage, G. Bosilca, A. Brooks, P. H. Carns, A. Castell´o, D. Genet, T. H´erault, S. Iwasaki, P. Jindal, L. V. Kal´e, S. Krishnamoorthy, J. Lifflander, H. Lu, E. Meneses, M. Snir, Y. Sun, K. Taura, and P. H. Beckman, "Argobots: A lightweight low-level threading and tasking framework," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 3, pp. 512–526, 2018.

[34] J. Soumagne, D. Kimpe, J. A. Zounmevo, M. Chaarawi, Q. Koziol, A. Afsahi, and R. B. Ross, "Mercury: Enabling remote procedure call for high-performance computing," in *2013 IEEE International Conference on Cluster Computing, CLUSTER 2013, Indianapolis, IN, USA, September 23-27, 2013*, 2013, pp. 1–8.

[35] S. Thapaliya, P. Bangalore, J. F. Lofstead, K. Mohror, and A. Moody, "Managing I/O interference in a shared burst buffer system," in *45th International Conference on Parallel Processing, ICPP 2016, Philadelphia, PA, USA, August 16-19, 2016*, 2016, pp. 416–425.

[36] Thinkparq and BeeGFS, "Beegfs the leading parallel cluster file system," 2018, https://www.beegfs.io/docs/BeeGFS Flyer.pdf.

[37] M.-A. Vef, V. Tarasov, D. Hildebrand, and A. Brinkmann, "Challenges and solutions for tracing storage systems: A case study with spectrum scale," *ACM Trans. Storage*, vol. 14, no. 2, pp. 18:1–18:24, 2018.

[38] M. Vilayannur, P. Nath, and A. Sivasubramaniam, "Providing tunable consistency for a parallel file store," in *Proceedings of the FAST '05 Conference on File and Storage Technologies, December 13-16, 2005, San Francisco, California, USA*, 2005.

[39] F. Wang, Q. Xin, B. Hong, S. A. Brandt, E. Miller, D. Long, and T. McLarty, "File system workload analysis for large scale scien-tific computing applications," Lawrence Livermore National Laboratory (LLNL), Livermore, CA, Tech. Rep., 2004.

[40] T. Wang, K. Mohror, A. Moody, K. Sato, and W. Yu, "An ephemeral burst-buffer file system for scientific applications," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2016, Salt Lake City, UT, USA, November 13-18, 2016*, 2016, pp. 807–818.

[41] J. Xing, J. Xiong, N. Sun, and J. Ma, "Adaptive and scalable metadata management to support a trillion files," in *Proceedings of the ACM/IEEE Conference on High Performance Computing, SC 2009, November 14-20, 2009, Portland, Oregon, USA*, 2009.

[42] S. Yang, W. B. Ligon III, and E. C. Quarles, "Scalable distributed directory implementation on orange file system," *Proc. IEEE Intl. Wrkshp. Storage Network Architecture and Parallel I/Os (SNAPI)*, 2011.