doi:10.11887/j.cn.202002003　　http://journal.nudt.edu.cn

面向大规模集群的并行 I/O 用户层配置优化策略*

田鸿运[1],武林平[1],董　勇[2],景翠萍[1],罗红兵[1],莫则尧[1]
(1. 北京应用物理与计算数学研究所, 北京　100094; 2. 国防科技大学 计算机学院, 湖南 长沙　410073)

摘　要:影响应用 I/O 性能的关键因素主要有三个层次:包括应用的 I/O 接口实现、体系结构和文件系统组件的性能以及应用的 I/O 参数配置。从应用 I/O 配置优化的视角,分析了大规模集群并行 I/O 的配置调优空间,在此基础上,给出了一套大规模集群并行 I/O 性能特征测试分析方法。基于该方法,在某国产超级计算集群上开展了一系列 I/O 测试分析来刻画系统的 I/O 性能特征,进而指导并行应用程序的 I/O 配置优化。基于优化后的配置参数,在两类典型的并行 I/O 场景中,针对某类生产应用程序,8192 进程下的重启动数据写操作时间下降了 15%,4096 核的程序作业加载时间从 10 min 缩短到了 5 s。

关键词:并行 I/O 优化策略;Lustre 文件系统;大规模集群;传输数据量;条带数
中图分类号:TP316　　**文献标志码**:A　　**文章编号**:1001-2486(2020)02-023-08

User-level parallel I/O configuration optimize strategy toward large-scale cluster

TIAN Hongyun[1], *WU Linping*[1], *DONG Yong*[2], *JING Cuiping*[1], *LUO Hongbing*[1], *MO Zeyao*[1]
(1. Institute of Applied Physics and Computational Mathematics, Beijing 100094, China;
2. College of Computer Science and Technology, National University of Defense Technology, Changsha 410073, China)

Abstract: Three key factors exert big influence upon the application's I/O performance, including the I/O programming interface, the performance characteristic of I/O sub-system(both architecture and system software), and the I/O configuration parameters at user-level. From the user's perspective, this paper discussed the user-level parallel I/O configuration optimize space toward large scale cluster. Besides, we proposed a method of testing and analyzing the I/O characteristic of large scale cluster. Based on this method, the I/O performance portrait of a domestic super computer was built up and several user-level parallel I/O optimize suggestions were put forward. With these carefully selected I/O configuration parameters, the time of restart data write operation was cut down by 15 percent under 8192 processes in a real application environment, while the program's initial time is shortened from 10 minutes to 5 seconds at the scale of 4096 processes.

Keywords: parallel I/O optimize strategy; Lustre file system; large-scale cluster; transfer size; stripe count

近年来,我国的超级计算机系统数量持续呈现快速增长的态势,2018 年 11 月公布的 TOP500 榜单中,我国以 227 台数量排名第一,远超第二名(美国)的 109 台[1]。如何用好大规模系统,充分发挥系统的巨大算力,已经成为超级计算行业面临的共性挑战问题[2]。要解决好这个问题,需要应用、系统、厂商的科研人员大力协同。

当前,计算能力排名世界前十的超级计算机已经达到数十万至数千万处理器核(ABCI 391 680 核,Sunway TaihuLight 10 649 600 核),整机内存容量达到数百 TB ~ 数 PB(PizDaint 169 TB ~ Summit 2.8 PB)[1]。虽然 TOP500 榜单的 Linpack Benchmark 只关注计算和通信的性能,缺乏对 I/O 性能的考量,然而随着系统规模和应用并行规模的持续增长,应用对 I/O 性能的需求已经越来越迫切。为了应对系统故障、异常等因素引起的超大规模并行场景下的运行时可靠性问题,检查点技术应运而生[3]。应用必须每隔一定时间步就将中间计算结果存入文件系统,这将爆发大量的 I/O 写操作。此外,在程序计算完成后,应用科学家还需要对计算结果进行后处理分析(如:可视化分析),这将产生大量的 I/O 读操作。因此,应用科学家对于大规模并行场景下的 I/O 性能越来越关注。

要充分挖掘并行文件系统的性能,既需要应用层 I/O 实现的优化(如:使用并行 I/O 接口替

* **收稿日期**:2019-09-19
基金项目:国家重点研发计划资助项目(2018YFB0204003)
作者简介:田鸿运(1986—),男,重庆人,博士研究生,E-mail:tian_hongyun@iapcm.ac.cn;
莫则尧(通信作者),男,研究员,博士,博士生导师,E-mail:zeyao_mo@iapcm.ac.cn

代 POSIX 串行 I/O 接口，如 HDF5、MPI – I/O 等），也需要系统级的并行 I/O 组件优化（如：Filter Cache 技术[4]、MDDS 技术[5]、针对 MPI – I/O 的优化[6]等），同时，还需要根据系统 I/O 特征对应用的 I/O 进行参数配置优化，这项工作需要应用用户、程序开发者和系统管理员协作来完成。例如，我们的测试发现，在大规模并行场景下，并非所有的作业进程都参与 I/O 操作时性能最优，而是应该根据作业输出目录使用的条带化数量动态调整。因此，需要在应用 I/O 特征的牵引下对大规模集群上的并行 I/O 性能特征开展有针对性地测试分析，从而指导应用 I/O 配置参数优化。

1 Lustre 并行文件系统

目前，在超级计算机上广泛使用的并行文件系统主要有 Lustre、GFS、GPFS、PVFS 等。其中，Lustre 占据绝对主导地位，在最近一期的 TOP500 榜单中，超过 70% 的超级计算机采用 Lustre 文件系统[1,7]。

1.1 Lustre 文件系统架构

Lustre 采用基于对象的存储技术，主要由元数据服务器（MetaData Server，MDS）、对象存储服务器（Object Storage Server，OSS）、客户端（Client）三部分组成。其中，MDS 用于存储数据的描述信息，管理命名空间和目标文件的存储地址，元数据的存储目标称为 MDT（metadata target）。为了增强元数据服务器的可靠性，Lustre 支持多个 MDS 访问同一套 MDT，分别用于元数据管理和容灾备份。OSS 进行实际的数据存储，提供文件 I/O 服务，每个 OSS 下可以管理多个对象存储目标服务器（Object Storage Target，OST）。Client 负责与 MDS 和 OSS 进行信息交互操作。

图 1 是典型的大规模集群上的 Lustre 文件系统架构示意图。为了满足大规模集群的 I/O 需求，Client、MDS、MDT、OSS、OST 分别部署在不同的节点上，并通过高速互联网络（如 InfiniBand、TH-Express2、Intel Omni-Path 等）进行连接。OST 下挂载多块硬盘（SAS、SSD、SATA），并使用 RAID 技术构成一个存储卷。在 MDS 的管理方面，Lustre 采用 active-standy 的方法，即双机热备。

1.2 Lustre 文件系统内的数据流

从应用的视角看，Lustre 文件系统内的数据流如图 2 所示。由于 Lustre 采用基于对象的存储技术，因此元数据和目标数据是分别存储管理的。如果不考虑 cache 的因素，用户的读 I/O 操作会

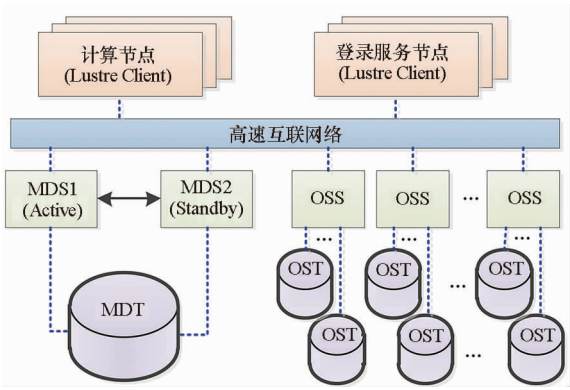


图 1 Lustre 文件系统架构示意图

Fig. 1 Lustre architectural diagram

先访问 MDS 获取相关文件的目录、权限等信息，再向 OSS 发起具体的读操作；用户的写 I/O 操作也会先通过 MDS 获取文件的元数据信息，再由客户端向 OSS 进行通信以实现数据的写操作。

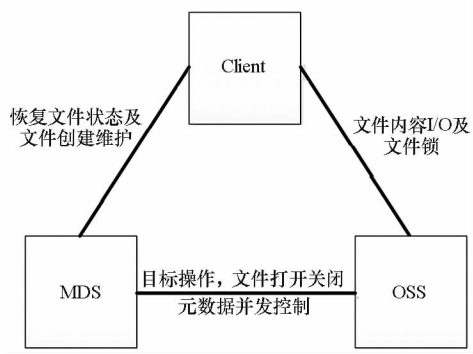


图 2 Lustre 文件系统内的数据流

Fig. 2 Dataflow diagram inside the Lustre file system

除了元数据和数据的分离，Lustre 还使用条带化技术将数据按照条带大小（stripe size）进行切分，再根据条带数量（stripe count）进行轮转分配，从而使数据文件均匀地分布到多个 OST 上，保持存储系统整体的负载均衡。

在 I/O 过程中，计算节点上的 I/O 数据通过高速通信端口进行传输，每个计算节点上的所有进程通常共享一个高速通信端口（虚拟通道），当计算节点上的所有 I/O 进程同时竞争该通道时，可能构成 I/O 性能的瓶颈。

2 应用特征分析与并行 I/O 配置优化空间

要开展大规模集群上的并行 I/O 配置参数调优，必须对应用当前的 I/O 模式有充分的调研和了解。科学计算数值模拟程序是大规模集群并行应用中的一类重要应用，本小节介绍科学计算数值模拟程序的 I/O 模式特征，分析这些 I/O 模式下的 Lustre 文件系统行为特征，进而研究大规模

集群上的并行 I/O 配置优化空间。

2.1　科学计算数值模拟程序的 I/O 特征

当前,北京应用物理与计算数学研究所的科学计算数值模拟程序在 I/O 方面主要表现出如下 5 个方面的特点:

1)编程接口:目前大多数科学计算数值模拟遗产程序还使用 Fortran 提供的文件操作编程接口,底层调用使用的是每个进程一个文件(one file per process)的 I/O 操作模式,少量程序使用了 HDF5 等并行 I/O 接口。在我们的调查中,还没有生产型的科学计算应用程序使用 MPI-I/O 的编程接口。

2)分组 I/O:一些科学计算数值模拟程序已经基于领域编程框架 JASMIN[8] 进行重构,利用该框架提供的合并输出功能,用户可将多个进程的输出数据合并到一个进程进行输出,用户可以在运行时设定进程合并输出的分组大小,从而控制用于 I/O 的进程数。缺省情况下分组大小为 1,即每个进程输出到独立的文件中。

3)数据大小:科学计算数值模拟程序 I/O 的数据主要包括计算结果数据(用于可视化后处理分析)和检查点数据(用于错误恢复、重启动),经过我们的调查发现,应用传输的文件数据大小波动范围大,单个进程 I/O 的文件大小在数 KB 至数百 MB 量级。

4)输出模式:大多数科学计算数值模拟程序采用向文件追加写入的方式进行写操作,计算过程产生数据之后随即进行写操作,单次写入的数据大小与源数据结构大小相关,无明显的特征;有一些开发者习惯于将输出数据通过归约操作汇集到某一个管理进程(通常是 0 号 MPI 进程),再由管理进程负责数据结果的输出。

5)读入特征:数据读操作主要发生在程序初始化和重启动时,在程序初始化时程序主要读入输入文件以及网格模型文件,输入文件的大小通常在几个 KB,而网格模型文件则可能在几百 MB 到几个 GB 量级,未来随着物理建模的精度进一步提高,还将使网格模型文件进一步增大;在程序重启动时,程序读入的是上一次退出时的内存镜像数据,重启动数据的大小与程序并行规模直接相关,数据大小可能在数百 GB 至 TB 量级。

2.2　系统视角下的 I/O 特征分析

IOR[9-10] 是由 LLNL 实验室开发的并行文件系统性能基准测试程序,支持对不同编程接口(POSIX、MPI-I/O、HDF5、parallelNetCDF)、不同 I/O 文件大小(file size)、不同数据传输尺寸(transfer size)、不同文件输出策略(SSF-Single shared file, FPP-File per process)、不同写入方式(direct I/O、buffer I/O)的 I/O 模式进行模拟,基于上述优良的特性,IOR 可以很好地模拟前述数值模拟应用的 I/O 特征。因此,本文以 IOR 为例分析系统管理视角下的应用 I/O 行为特征,在此基础上分析并行 I/O 配置优化空间。

如图 3 所示,在 IOR 中,transfer size 代表单次 I/O 操作的数据大小,对应于 POSIX 文件操作接口 read( )、write( )中单次数据的传输大小,每个 Block 包含若干个 transfer size,一个 Block 对应一个 I/O 进程,若干个 Block 构成一个 Segment,对应于数值模拟程序中的一个数据区。在读写数据时,多个 I/O 进程可以参与数据区中不同数据块的 I/O 操作,每个数据块的 I/O 操作被分解为若干个 transfer size 来完成。IOR 这种分层特征与数值模拟应用的 I/O 特征相似。

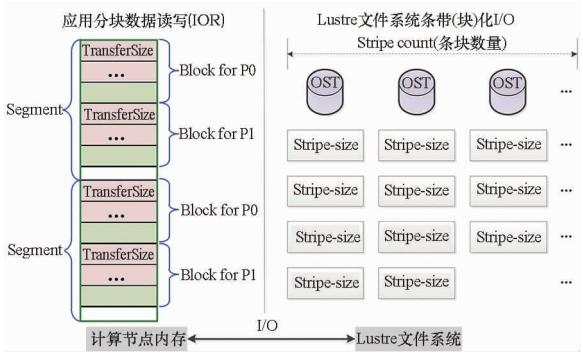


图 3　系统视角下的应用 I/O 过程

Fig. 3　Application I/O feature from system perspective

在 Lustre 文件系统的层面,应用传输的文件大小被按照条带切分,并将切分后的条带均匀地分布到不同的 OST 上。用户可以在程序运行前设置程序输出目录的条带数量(stripe count)和条带大小(stripe size)从而控制 Lustre 文件系统的 I/O 行为。IOR 的 transfer size 与 Lustre 文件系统的 stripe size 在大小上并没有直接关联,前者由程序实现决定,后者由用户或者系统管理员进行配置。

2.3　大规模集群上的并行 I/O 配置优化空间

从 2.2 节的分析可知,用户可以从如下 5 个方面进行并行 I/O 配置调优:

1)根据节点的并行 I/O 性能特征,选择每个进程单次 I/O 传输的数据量(对应 IOR 的 transfer size);

2)根据节点的并行 I/O 性能特征,调节每个

进程传输的数据量(对应 IOR 的 block size × segment_count);

3)根据节点的并行 I/O 性能特征,调节每个节点上用于 I/O 的进程数(process per node);

4)根据预估计的输出文件大小和文件数量,为输出目录配置合适的条带化大小(stripe size)和条带化数量(stripe count);

5)在大规模并行场景下,选择合适的客户端数量参与并行 I/O 操作,以获取最优的 I/O 性能。

3 大规模集群并行 I/O 性能特征测试与分析

Lustre 文件系统的性能受到很多因素的影响(例如:通信网络带宽、存储设备性能、Lustre 本身设置参数等),如果单纯进行系统级的 I/O 性能特征测试分析,则无法建立起它们与应用 I/O 性能优化间的关联关系。因此,需要在应用并行 I/O 配置优化空间的指导下,有针对性地开展系统 I/O 性能特征的测试分析。围绕 2.3 节所述的并行 I/O 配置优化空间,本节以某国产高性能计算机系统为例,介绍大规模集群并行 I/O 性能特征测试分析方法。

实验平台的每个计算节点包含 32 个处理器核,每个计算节点的内存为 128 GB,平均内存量为 4 GB/核,节点操作系统版本为 RHEL - 7.4,内核版本为 Linux - 3.10.0。实验平台的 Lustre 文件系统为 2.10,全系统共配置 96 个 OST。测试使用到了该集群其中的 256 个节点(8192 核),使用的 IOR 为 3.0.1,编译器版本是 Intel 17.0.4,MPI 版本为 3.2.1。

3.1 每个进程单次传输的最佳数据量

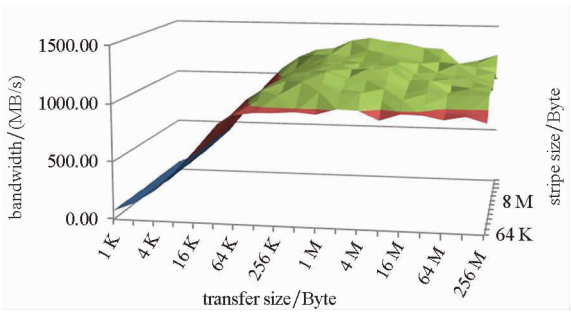
每个进程单次传输的数据量是指进程单次 I/O 操作调用 read/write 接口时,传入的 buf 长度,见图 4。buf 长度将会直接影响到数据传输时调用 I/O 接口的次数,从而对应用的 I/O 性能产生影响。当传输的数据量大时,单次传输很小的数据量将会引入过多的网络传输启动开销,从而使 I/O 性能下降。

```
ssize_t write(int fildes, const void *buf, size_t nbyte);
ssize_t read(intfildes, void *buf, size_t nbyte);
```

图 4 POSIX I/O 接口

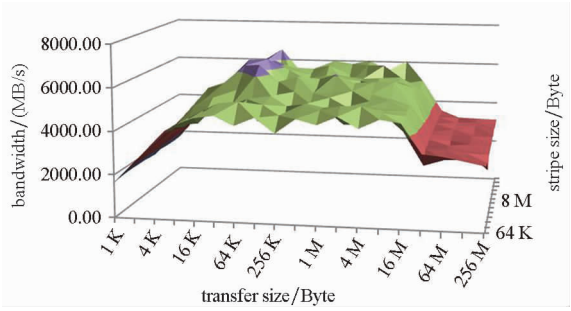
Fig. 4 POSIX I/O API

如图 5 所示,调节每个进程单次传输的数据量(transfer size)和程序输出目录的条带大小(stripe size),可以看到读/写带宽随 stripe size 的变化不太明显,但受 transfer size 影响很大。其中,写带宽随 transfer size 增大而增大,读带宽随 transfer size 增大也有明显增大,但是当 transfer size 大于 8 MB 后,再增大 transfer size 则读带宽出现了下降,性能下降的原因可能是由于节点上高速网卡读数据 vp 通道成了性能瓶颈。在目标测试平台上,基于读和写性能的综合考虑,每个进程单次传输的最佳数据量在 64 KB ~ 8 MB 之间。



(a) 写操作的 I/O 带宽

(a) Write bandwidth



(b) 读操作的 I/O 带宽

(b) Read bandwidth

图 5 传输尺寸和条带大小对 I/O 带宽的影响

Fig. 5 Influence of transfer size and stripe size to I/O bandwidth

3.2 每个进程传输的最佳数据总量

每个进程传输的数据总量是指每个进程负责传输的总数据量。当 I/O 进程传输的数据量很小时,无法充分利用网络带宽和磁盘带宽;当 I/O 进程传输的数据量过大时,又会因为对网络和存储资源的竞争而导致性能的下降,因此用户需要调节每个进程传输的数据总量。

如图 6 所示,当每个进程传输的数据总量小于 1 GB 时,读操作的性能受益于 vfs 的 page cache,获得了很高的读带宽,但是随着数据量的增大,缓存开始失效,读带宽迅速下降。写操作在每个进程传输的数据总量大于 64 MB 之后,基本达到饱和。可见,针对目标测试平台,每个进程传输的数据总量在 64 MB ~ 1 GB 之间是较好的

选择。

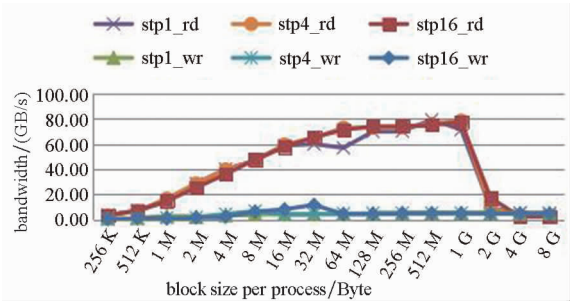


图 6　每个进程传输的数据总量对 I/O 带宽的影响

Fig. 6　Influence of block size per process to I/O bandwidth

3.3　每个节点的最优 I/O 进程数

由于单个计算节点上的 I/O 操作需要通过竞争网络资源来将数据传输到 I/O 节点（MDS、OST），因此单个计算节点上的每个进程传输数据总量（block size per process）和每个计算节点上的 I/O 进程数量（process per node）将会对 I/O 性能产生影响。

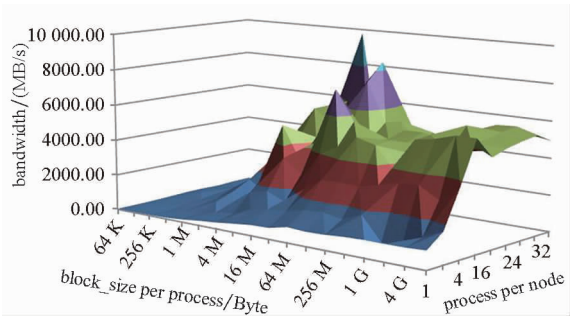
假设点到点的通信带宽为 x GB/s，每个 OST 可提供的 I/O 带宽为 y GB/s，为该作业分配的条带数（stripe count）为 n。则当 $x < y \cdot n$ 时，节点的通信带宽将成为 I/O 瓶颈。

在图 7 中，将输出目录的条带数设置为 4，并根据 3.1 节和 3.2 节的测试结果，将单次 I/O 的数据量（transfer size）设为 64 KB，条带大小设置为 64 KB，调节每个进程传输的数据总量（block size）以及节点上的 I/O 进程数（process）。从图 7 可以看出，当节点上的 I/O 进程数较少时，没有充分利用节点的网络带宽资源，I/O 带宽较低。单个计算节点的 I/O 进程数大于 8 之后，对计算节点的高速通信网络带宽基本达到饱和。进一步的测试数据表明，当单个进程传输的数据量（block size）为 8 MB，单个计算节点的 I/O 进程数为 32 时，写带宽达到了最大值 9.785 GB/s，此时的 I/O 写带宽性能值与该系统平台的网络点到点传输性能基本相当，表明此时 I/O 性能已经因为网络带宽达到了极限。

在目标测试平台上，每个进程的传输数据量（block size）较优选择在 4 MB ~ 2 GB 之间，相应地，每个节点上的较优 I/O 进程数（process）在 8 ~ 32 之间。综合来看，每个节点上的 I/O 进程数为 8 时始终能够获得较好的 I/O 性能。

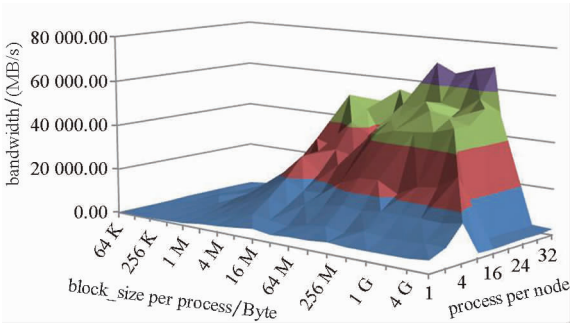
3.4　文件大小与条带大小、条带数量的关联分析

条带大小和条带数量对于 Lustre 文件系统的性能会产生较大影响，这在很多文献中都有提及[6,11-16]，但是它们的最优参数设置与文件大小的关系，文献中没有给出答案。为此，针对文件大小与条带大小（stripe size）、条带数量（stripe count）进行了测试分析。



（a）写操作的 I/O 带宽

（a）Write bandwidth



（b）读操作的 I/O 带宽

（b）Read bandwidth

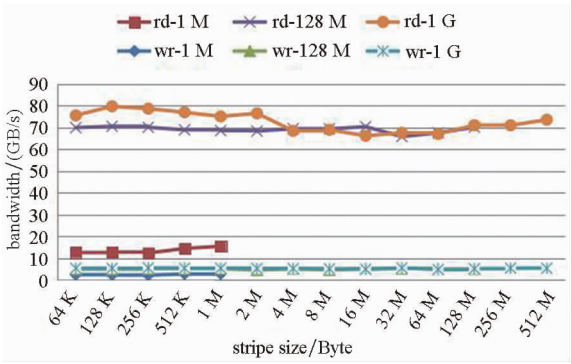
图 7　每个节点的进程数对 I/O 带宽的影响

Fig. 7　Influence of process per node to I/O bandwidth

对于小文件 I/O，条带大小和条带数量都无法带来直接的性能提升；对于大文件 I/O，条带过大则实际用于存储该文件的条带数会减少，从而无法充分利用存储带宽资源。本文分别针对条带大小和条带数量进行了测试。

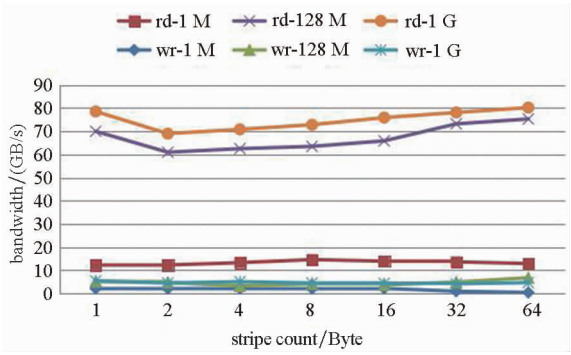
如图 8（a）所示，对于小文件（1 MB/进程）条带大小与该文件大小相当时 I/O 的性能较高；对于中等文件（128 MB/进程）和大文件（1 GB/进程），I/O 性能都随着条带增大而有所下降。如图 8（b）所示，条带数量对于 I/O 性能提升起到了一定的促进作用。图中，条带数为 2 相比条带数为 1 时性能出现了下降是因为传输的文件较小，数据分块导致的 I/O 性能下降。图 8（a）中部分曲线没有画满，是因为 stripe size 增大到文件大小时，再增大该参数已无意义。

综合来看，在目标测试平台上：①增大条块数量能够带来一定的 I/O 性能提升；②对于小/中/大文件的 I/O 操作而言，条带大小设为 1MB 是一个较优的选择。



（a）条带大小对 I/O 性能的影响

（a）Influence of stripe size to I/O performance



（b）条带数量对 I/O 性能的影响

（b）Influence of stripe count to I/O performance

图 8　条带大小和条带数量对 I/O 带宽的影响

Fig. 8　Influence of stripe size and stripe count to I/O bandwidth

3.5　大规模作业的最优 I/O 进程数分析

根据前面的调研，科学计算数值模拟程序大多采用每个进程一个文件的 I/O 模式。然而，在大规模并行场景下，所有进程都参与 I/O 操作显然不是一个最优的选择。这主要是因为：①所有进程都参与 I/O 会导致节点上高速网络资源的竞争；②所有进程都参与 I/O 操作，会产生大量的元数据操作请求，进而加大 MDS 的响应开销；③所有进程都参与 I/O 操作，则每个 OST 都要响应全部作业进程，这会超出 OST 的响应能力。其中，对于节点上高速通信网络资源的竞争，3.3 节已有分析。

对于并行文件的操作开销，图 9 给出了目标测试平台的统计结果。可以看出，在目标测试平台上，当作业进程数大于 1024 之后，作业并行开关文件的开销迅速增大，对于数万核进程并行的作业，如果采用每个进程一个文件的 I/O 操作模式，则其并行 I/O 性能势必会受到很大的影响。

为了考察并行 I/O 进程数对 I/O 性能的影响，我们在目标测试平台上，将作业输出目录的条带数量固定设置为 16，然后对比不同 I/O 进程数

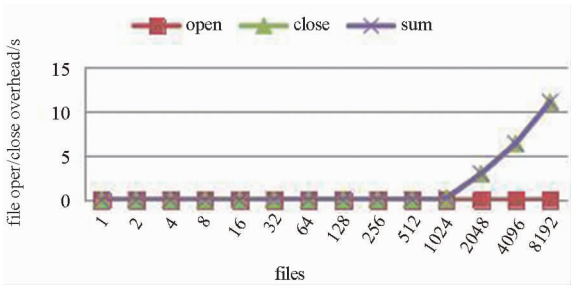


图 9　并行文件开关操作开销

Fig. 9　Overhead of parallel file open/close operation

量下的 I/O 写操作带宽。对于不同大小的文件，8192 进程时的 I/O 带宽都比 4096 时的 I/O 带宽出现了下降，如图 10 所示，表明此时的 I/O 进程数已经超过了 OST 的响应能力。对于每个进程 1 GB 的 I/O 数据量而言，最优的并行 I/O 进程数是 256。

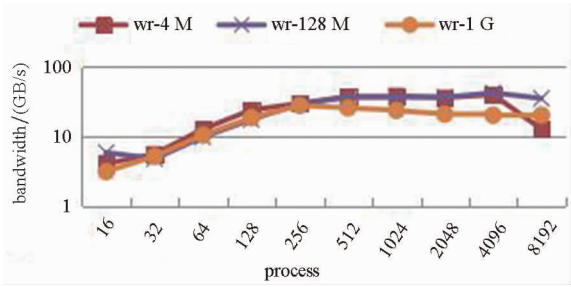


图 10　并行 I/O 进程对 I/O 性能的影响

Fig. 10　Influence of process number to I/O bandwidth

4　大规模集群上的并行 I/O 配置优化策略及应用验证

结合前述测试分析，我们给出目标测试平台的并行 I/O 配置优化策略，如下：

1）每个进程单次传输的数据量对 I/O 性能有较大的影响，建议控制单次数据传输量在 64 KB ~ 8 MB 之间。

2）每个进程传输的数据量对 I/O 性能有较大的影响，建议单进程数据传输总量在 8 MB ~ 1 GB之间。

3）受限于节点的高速通信网络带宽，建议每个节点的 I/O 进程数在 8 个左右。

4）通过预估计需要进行 I/O 操作的文件大小，设置合理的条带化大小和条带化数量，可以提升 I/O 性能。对于小文件，建议条带化设置为 1，对于中等大小的文件，建议条带化设置为 4，对于大文件，建议条带化设置为 16。考虑到实际生产环境下的存储可靠性以及多个大规模作业之间可能产生的影响，不建议对于大文件设置过多的条带数。

5)考虑到大规模并行 I/O 时的文件操作开销,建议大规模并行时的 I/O 进程数不超过 1000;考虑到条带数量设置导致的文件系统对应用并行 I/O 的响应能力限制,建议大规模并行作业的 I/O 进程数量不超作业输出目录条带数(stripe count)的 16 倍(经验值)。对于基于结构网格编程框架的程序,建议使用合并输出功能,调节合并输出的分组大小,将单个计算节点上的 I/O 进程数控制在 8 左右,将总体的 I/O 进程数控制在 1000 以下。

上述策略的具体配置数值可能因为各系统平台的配置不同而不同,但该策略的优化思路可以推广应用到其他同类超算系统平台。基于上述优化策略,针对两类典型并行 I/O 场景进行了测试验证:

1)场景 1:大规模并行程序加载初始化。针对某生产应用程序,通过:①增大条带化数量(条带数从 1 增大到 16),从而增大响应读操作请求的 OST 数量;②调节参与读操作的 I/O 进程数(每个节点指派 1 个读数据文件的 I/O 进程),减小多进程读取同一文件时引入的文件锁竞争。使得 4096 核的作业加载时间从 10 min 降到了 5 s。

2)场景 2:大规模并行程序写重启动数据。针对某生产应用程序,通过:①设置作业输出目录的条带数量(条带数从 4 增大到 16);②调节参与写操作的 I/O 进程数(每个计算节点 8 个 I/O 进程),减少并行文件操作开销。使得 8192 进程下的重启动数据写操作时间下降了 15%。

5　讨论

近年来,针对大规模集群上的并行 I/O 性能优化已有大量的研究工作,这些工作主要集中在并行 I/O 算法接口的实现优化以及针对 Lustre 文件系统的性能优化两个方面。

在并行 I/O 接口优化方面,文献[6]对 MPI-I/O 的聚合操作(Collective)进行了优化,文献[17]提出了一种针对 MPI-I/O 的自动调优框架,文献[18]针对 HDF5 的读操作使用增加数据块视图方法进行了优化。目前在调研的核科学数值模拟应用中,还没有生产型应用直接使用 MPI-I/O 的方式实现并行 I/O。

在 Lustre 文件系统组件性能优化方面,文献[4]针对 Lustre 文件系统上的小文件 I/O 性能低下问题,提出了一种“Filter Cache”方法;文献[5]针对 Lustre 文件系统的元数据性能瓶颈问题,采用分层的元数据管理方法,设计实现了元数据代理 MDDS,提升元数据访问性能;文献[19]基于 TH-2 的高性能互连网络,实现了针对该机文件系统的高速通信模块 FSE 优化,获得了更高的数据访问带宽和元数据性能。

然而,从并行 I/O 配置参数优化的角度开展大规模集群上的并行 I/O 调优还不多见,在这方面,与本文研究切入点最为接近的是文献[15-16,20],其中文献[16,20]分别针对 Jaguar 和 RedStrom 超级计算机系统上的并行文件系统性能展开了测试和分析,文献[15]则从应用 I/O 特征的角度提出了在 Jaguar 系统上的并行 I/O 优化建议。国内也有研究者开展 Lustre 文件系统的性能测试研究[11-14],但都仅限于在小规模下开展实验测试,其研究结论无法推广适用于大规模集群。

6　结论

本文从科学计算数值模拟应用程序的 I/O 特征出发,从应用 I/O 配置优化的视角来看应用程序 I/O 过程中的应用行为和系统行为,进而提出了大规模集群上的并行 I/O 配置调优空间,给出了从 5 个方面开展配置参数优化的调优策略,在此基础上,在某国产超算系统平台上开展了上述 5 个方面的调优分析测试研究,并将优化策略在两类典型的应用 I/O 场景中进行了验证。对于大规模并行程序加载初始化以及程序写重启动数据两类典型 I/O 场景,提出的优化策略都分别取得了明显的效果。

本文提出的并行 I/O 配置参数优化策略,可以推广应用到其他同架构的超算系统平台,对于大规模集群上的用户层并行 I/O 配置调优具有借鉴意义。还需说明的是,本文提出的部分并行 I/O 配置参数优化策略,需要应用开发人员和系统研究人员共同参与完成,例如调整每个计算节点上用于 I/O 的进程数量这一策略,对于基于领域编程框架的应用程序,可以使用框架提供的聚合输出功能,在模型配置文件中调节聚合输出的分组 I/O 大小,而对于非框架应用程序,则需要在程序代码上进行改进。要实现科学计算数值模拟应用程序的 I/O 参数全部可动态调节,还需要应用开发人员和系统研究人员的持续共同努力。

参考文献(References)

[1] Dongarra J, Luszczek P. TOP500[R/OL]. Encyclopedia of Parallel Computing. [2018-11-25]. https://www.top500.org/lists/2018/11.

[2] 张云泉. 2018 年中国高性能计算机发展现状分析与展望[J]. 计算机科学, 2019, 46(1): 1-5.

ZHANG Yunquan. Analysis and prospect of China′s high performance computer development in 2018[J]. Computer Science, 2019, 46(1): 1 – 5. (in Chinese)

[3] 谢旻, 卢宇彤, 周恩强, 等. 基于 Lustre 文件系统的 MPI 检查点系统实现技术与性能测试[J]. 计算机研究与发展, 2007, 44(10): 1709 – 1716.
XIE Min, LU Yutong, ZHOU Enqiang, et al. Implementation and evaluation of MPI checkpointing system over Lustre file system[J]. Journal of Computer Research and Development, 2007, 44(10): 1709 – 1716. (in Chinese)

[4] 李柱, 周恩强, 廖湘科. Filter Cache:一种提高 Lustre I/O 性能的方法[J]. 计算机研究与发展, 2009, 46(Z2): 71 – 77.
LI Zhu, ZHOU Enqiang, LIAO Xiangke. Filter Cache: a way to Improve the I/O Performance of Lustre[J]. Journal of Computer Research and Development. 2008, 46(Z2): 71 – 77. (in Chinese)

[5] 陈起, 陈左宁, 蒋金虎. MDDS:一种面向高性能计算的并行文件系统元数据性能提升方法[J]. 计算机研究与发展, 2014, 51(8): 1663 – 1670.
CHEN Qi, CHEN Zuoning, JIANG Jinhu. MDDS: a metadata performance improvement method for HPC parallel file system[J]. Journal of Computer Research and Development, 2014,51(8): 1663 – 1670. (in Chinese)

[6] 庄园. 基于 Lustre 文件系统 MPI – I/O 优化的改进与实现[D]. 济南: 山东大学, 2017.
ZHUANG Yuan. Optimization of MPI – I/O based on Lustre file system[D]. Ji′nan: Shandong University, 2017. (in Chinese)

[7] Kosta L, Hunter H, George G. Measuring I/O performance of Lustre and the temporary file system for tradespace applications on HPC systems[C]//Proceedings of the SouthEast Conference. ACM, 2017: 187 – 190.

[8] 莫则尧. 高性能数值模拟编程框架研究进展[J]. 科研信息化技术与应用, 2015, 6(4): 11 – 19.
MO Zeyao. Research progress of high performance numerical simulation programming framework[J]. Journal of Research Information Technology and Application, 2015, 6(4): 11 – 19. (in Chinese)

[9] Shan H Z, Antypas K, Shalf J. Characterizing and predicting the I/O performance of HPC applications using a parameterized synthetic benchmark[C]// Proceedings of the ACM/IEEE Conference on High Performance Computing, Austin, Texas, USA, 2008.

[10] Shan H Z, Shalf J, Using IOR to analyze the I/O performance of HPC platforms[R]. In Cray User Group (CUG), Seattle, WA, 2007.

[11] 董勇, 周恩强, 陈娟. 基于 Infiniband 技术构建高性能分布式文件系统 – Lustre[J]. 计算机工程与应用, 2005, 41(22): 103 – 107.
DONG Yong, ZHOU Enqiang, CHEN Juan. Construct high performance distributed file system based on Infiniband technology[J]. Computer Engineering and Application, 2005, 41(22): 103 – 107. (in Chinese)

[12] 梁军, 聂瑞华. 面向对象存储的文件系统 Lustre[J]. 计算机工程与设计, 2015, 36(6): 1666 – 1669.
LIANG Jun, NIE Ruihua. Lustre file system based on object Storage. [J] Computer Engineering and Design, 2015, 36(6): 1666 – 1669. (in Chinese)

[13] 李亮, 聂瑞华. 高性能计算平台的 IO 性能测试与分析[J]. 计算机与现代化, 2011(6): 160 – 165.
LI Liang, NIE Ruihua. Test and analysis of IO performance on high performance computing platform[J]. Computer and Modernization, 2011(6): 160 – 165. (in Chinese)

[14] 刘昊晨. Lustre 设置优化分析[J]. 计算机光盘软件与应用, 2012(20): 71 – 73.
LIU Haochen. Optimization analysis of Lustre settings[J]. Computer CD Software and Application, 2012(20): 71 – 73. (in Chinese)

[15] Larkin L, Fahey M R. Guidelines for efficient parallel I/O on the cray XT3/XT4[R]. In Cray User Group (CUG), Seattle, WA, 2007.

[16] Yu W K, Vetter J, Oral S. Performance characterization and optimization of parallel I/O on the cray XT[C]//Proceedings of the 22nd IEEE International Parallel & Distributed Processing Symposium (IPDPS′08), Miami, Florida USA, 2008.

[17] 刘伟峰. 机群环境下并行 I/O 操作优化研究[D]. 济南: 山东大学, 2016.
LIU Weifeng. Optimization research on parallel I/O operation in cluster environment[D]. Ji′nan: Shandong University, 2016. (in Chinese)

[18] 沈卫超, 曹立强, 夏芳, 等. 面向数值模拟数据的 HDF5 性能优化[J]. 计算机研究与发展, 2012(S1): 314 – 318.
SHEN Weichao, CAO Liqiang, XIA Fang, et al. Performance optimization of HDF5 for numerical simulation data[J]. Computer Research and Development, 2012(S1): 314 – 318. (in Chinese)

[19] 董勇, 周恩强, 卢宇彤, 等. 基于天河 2 高速互连网络实现混合层次文件系统 H2FS 高速通信[J]. 计算机学报, 2017(9): 1961 – 1979.
DONG Yong, ZHOU Enqiang, LU Yutong, et al. H2FS: high speed communication based on Tianhe 2 high performance interconnection network[J]. Journal of Computer Science, 2017(9): 1961 – 1979. (in Chinese)

[20] Laros J H, Ward H, Klundt R, et al. Red storm IO performance analysis[C]// Proceedings of IEEE International Conference on Cluster Computing, Austin, TX, USA, 2007.