doi:10.11887/j.cn.201501008 http://journal.nudt.edu.cn

面向分层混合存储架构的协同式突发缓冲技术*

周恩强,张 伟,董 勇,卢宇彤
(国防科技大学 高性能计算国家重点实验室,湖南 长沙 410073)

摘 要:科学计算产生和分析的数据规模日益增长,高性能计算机的存储系统在体系架构和软件管理方法上面临重大挑战。针对天河-2系统的新型分层混合存储架构,提出一种由应用程序耦合的协同式突发缓冲技术来有效利用其存储资源优势。该方法采用运行时动态耦合的方法,将临近计算任务的分布式高速存储资源聚合成为一个的巨大的协同式突发缓冲区,通过采用文件命名空间投影的映射方法组织全局数据视图,利用位置亲和和感知数据意图的方法来挖掘空间局部性和时间局部性,并利用应用并发度感知的策略优化数据移动效率。天河-2系统的测试结果表明,该方法能够有效优化多种典型应用场景,可获得高可扩展的突发并行输出带宽和稳定的持续并行输出带宽,可显著提升数据分析场景的输入/输出性能,适合应用于大规模超级计算机的存储系统。

关键词:超级计算机;存储架构;并行文件系统;突发缓冲区
中图分类号:TP309 **文献标志码**:A **文章编号**:1001-2486(2015)01-047-06

Research on optimization towards hybrid and hierarchy storage architecture

ZHOU Enqiang, ZHANG Wei, DONG Yong, LU Yutong
(State Key Laboratory of High Performance Computing, National University of Defense Technology, Changsha 410073, China)

Abstract: Today's advancing modern science is generating and analyzing increasing scale of datasets and makes HPC storage system facing new challenges both on architecture and software approach. In order to exploit potential benefits of emerging hybrid and hierarchy storage architecture on Milky-2 system, a I/O middleware approach named application coupled burst buffer, is introduced to make full use of the solid state disk based in-system storage resources. Application coupled burst buffer aggregates distributed in-system storage close to running tasks into single namespace during application runtime and manages it as cooperative persistent burst buffer tightly coupled with its host application. To take full advantage of cooperative burst buffer, application coupled burst buffer uses an unified shadow namespace to map application data into physical in-system storage based on its real namespace of the host application. Besides that, application coupled burst buffer organizes data with locality aware layout and leverages application intent based replacement policy to fully exploit spatial and temporal locality. Furthermore, application coupled burst buffer employs concurrency aware policies to optimize data movement between different storage tiers. Evaluations on Milky-2 system show that application coupled burst buffer can improve the performance of typical data-intensive applications dramatically. It can achieve scalable burst I/O bandwidth and smooth sustained I/O bandwidth with high throughput solid state disk deployed and can be taken as an appropriate candidate for storage solution on emerging leadership supercomputer systems.

Key words: supercomputer; storage architecture; parallel file system; burst buffer

随着科学计算模拟精度逐渐提高,完整的科学模拟过程中产生和分析的数据量呈指数增长,计算瓶颈问题逐渐转化为I/O瓶颈问题。为了提供和计算能力相匹配的I/O能力,天河-2系统采用了新型的分层混合输入输出架构(Hybrid and Hierarchy I/O, H2IO)。H2IO采用混合存储、优势互补的方法,利用高速存储介质提供高I/O带宽和高可扩展性,利用传统存储介质提供大存储容量,但现有并行文件系统难以有效管理和利用这种架构,要求探索新的软件方法来充分发挥其架构的性能优势。

针对H2IO架构条件下的典型I/O问题,作者提出了基于运行时耦合的协同式突发缓存方法,在作业运行时将临近计算任务的加速存储资源聚合成为应用程序独占的、具有主动优化能力的全局协同式缓冲池,通过挖掘时空局部性和感

* **收稿日期**:2014-06-11
基金项目:国家自然科学基金资助项目(61120106005);国家"863"高技术研究发展计划基金资助项目(2012AA01A301)
作者简介:周恩强(1972—),男,浙江台州人,副研究员,硕士,硕士生导师,E-mail:eqzhou@nudt.edu.cn

知并发度来优化数据布局和并发访问,使其能够高效优化大规模突发数据输出和数据分析两种典型应用场景。

1 H2IO 架构及相关问题分析

天河 -2 采用传统磁盘存储和新型高速闪存类存储分层部署的 H2IO 混合存储架构。如图 1 所示,H2IO 架构在中转结点(I/O Node, ION)部署了自主设计的高速闪存阵列(Solid State Disk, SSD),若干个计算结点(Compute Node,CN)和临近的 ION 构成一个 I/O 紧耦合的物理分区,计算结点通过 ION 中转访问传统大容量磁盘阵列(Disk Array, DA)。该架构下两类存储相对于计算结点的 I/O 路径距离不同,因此计算耦合度不同。H2IO 的设计目标是让高并发密集 I/O 请求尽量直接访问临近的 SSD,使得慢速磁盘存储远离计算结点且松散关联,不再成为影响扩展性的障碍。这种架构平衡了带宽、容量和构建成本等需求,已经成为更大规模超级计算机的存储架构趋势,同时也带来使用上和存储管理上的复杂性,现有并行文件系统直接用于 H2IO 架构存在以下突出问题。

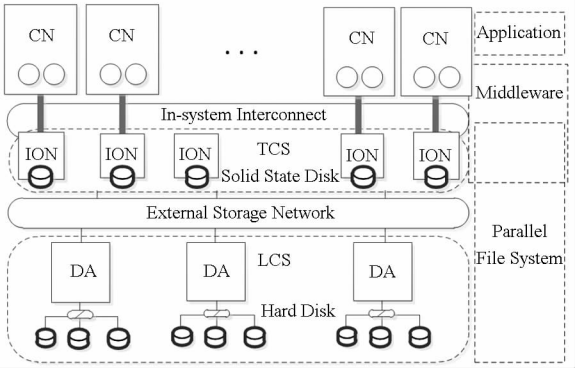

图 1 H2IO 架构和软件栈

Fig. 1 Architecture and software stack of H2IO

(1)扩展性问题。闪存阵列靠近计算结点部署,伴随计算规模扩展,具有规模大、资源高度分布的特征。如果采用传统的并行文件系统的方法,则文件系统作为全部闪存阵列的管理者,需要统一管理伴随计算规模扩展且分布化的存储空间。为了有效保证 I/O 服务的扩展性和共享访问的一致性,文件系统必须动态跟踪和控制数千甚至上万规模的软硬件存储资源[1]。统一管理的代价随系统规模增长而变大,引发的各种扩展性问题[2-4],不利于 H2IO 充分发挥应有的带宽优势。

(2)请求冲突问题[5-7]。应用程序可获得的实际带宽经常受限于局部请求竞争。大规模并发条件下系统的 I/O 负载具有明显的动态特征,底层文件系统的调度器难以准确控制 I/O 请求避免热点,并发的 I/O 请求会在硬件层(例如网络、控制器和磁盘)和软件层(例如锁服务、请求队列)产生明显竞争,I/O 请求完成时间的波动幅度变大。在大规模存储系统中,这种性能易变性相当普遍,导致并行任务间获得的 I/O 带宽不均衡,根据木桶原理,并行作业的完成时间会受到严重影响。

(3)时空局部性稀缺问题[3,6]。系统规模变大,挖掘局部性成为提高系统扩展能力的重要手段,现有并行文件系统假设存储资源是全局对称的,因此不感知存储位置的远近,而 H2IO 架构中不同位置的存储访问代价是不对称的,这导致请求访问缺乏空间局部性。现有文件系统为保证存储资源全局共享,在多任务并发条件下,允许 I/O 请求相互竞争,某种程度上破坏了时间局部性,时空局部性的稀缺会极大影响并发 I/O 的扩展性。实际上并行作业运行时仅共享访问自身的工作数据集,并且希望独占使用局部存储资源,全局共享的资源管理原则在某种程度上不利于挖掘局部性,影响大规模存储系统的可扩展性。

2 相关研究

Lustre[1]是超级计算机采用最多的并行文件系统,其通过对象存储思想来提高大文件访问带宽,但其元数据服务采用基于轮转调度的静态负载均衡策略,难以避免多作业多任务并行 I/O 可能产生的局部竞争,容易产生性能波动现象。Lustre 仅支持单层集中式存储架构,因此日本的 K 计算机采用两个独立的并行文件系统分别管理两层存储。文献[7]针对集中式存储架构下的网络拥塞问题,采用感知计算结点位置的文件布局技术,挖掘网络拓扑的局部性。GooleFS[8]和 HDFS[9]是分布式存储架构的文件系统代表,能够把高度分布化的存储资源聚合成单一空间,两者均采用随机调度的方法为单个文件分配数据块,由顶层作业调度器负责调度任务去挖掘数据访问局部性,其适合用于写少读多的大数据分析模型,不支持高速并发 I/O 的场景,本文主要关注高带宽并发 I/O。并行文件系统处于 I/O 栈底层,适合实现空间的统一管理和全局共享,但往往难以做到效率最优。

针对并行文件系统的不足,I/O 中间件被用来在文件系统之上优化 I/O 效率。ZIOD[10-11]用于优化蓝色基因系列并行计算机的 I/O 中转架构,利用中转结点的内存空间,采用请求聚合、流水等方法提高 I/O 请求中转的效率,但内存容量

过小且易失,使其难以用于大数据量突发和数据分析场景。文献[12]将固态盘集成到 I/O 中转架构中用作突发缓冲,研究其对于 HPC 应用的带宽提升效果和存储成本优势。ADIOS [13]利用分段传输(staging)的方法使用系统内的存储资源,其优化机理和突发缓冲类似,但两者都着重用于数据输出场景。PLFS[2]针对并行文件系统全局一致性语义引发的锁冲突问题,把单个大文件视图转化为多个独立的子文件,专门用于提高检查点文件的并行输出带宽。I/O 中间件的优化方法适合针对应用做定制优化,但不适合做存储空间的集中管理。

3　运行时耦合的协同式突发缓存技术

天河 -2 的 I/O 密集应用场景可归纳为以下 2 类:

(1)并行输出场景,即写密集型,例如用于任务容错的检查点数据、可视化数据和临时数据,数据量大,并发度高,具有明显突发 I/O 的特征;

(2)数据分析场景,计算任务读入数据进行数据分析的场景,需浏览输入数据集,属于并发读密集型,例如基因数据处理、地震数据处理。

突发缓冲[12](Burst Buffer,BB)是一种平滑带宽供需矛盾的技术手段,其核心思想是通过额外占用一组结点的存储资源作为 I/O 数据缓冲,快速吸收数据输出过程中大量突发的 I/O 请求,使得计算任务不必等待 I/O 请求抵达底层存储,即可返回继续计算,从而降低数据输出时间。BB 的思想源于大部分 HPC 应用的 I/O 输出模式具有突发特征[14],即短时间内需输出大量数据,并且“计算→I/O→计算”周期交替。BB 吸收数据后,利用任务的计算周期会异步将数据存入真实存储系统。

BB 的逻辑相对简单,BB 结点间一般无须构建数据文件之间的逻辑关联,没有分布式文件系统的统一管理开销,并行输出扩展性极佳,适合满足场景(1)的需求,但松散的 BB 架构使其不能用于大规模数据分析,对于场景(2)则存在局限性。

作者提出基于应用指导的方法(Application Coupled Burst Buffer,APBB)构建协同式 BB 池,使其能够支持以上两类典型应用模式,使得 H2IO 架构的作用能够充分发挥。如图 2 所示,APBB 根据存储和计算任务的耦合关系,将 H2IO 架构抽象为分层的存储模型,由两部分构成:紧耦合存储(Tightly Coupled Storage,TCS)和松耦合存储(Loosely Coupled Storage,LCS),其中 TCS 资源由应用程序指导中间件来使用,LCS 则由传统的并行文件系统管理。APBB 的核心思想是:(1)以计算作业为宿主,把作业任务临近的局部 TCS 看做私有突发缓冲,避免由并行文件系统统一管理 TCS 的代价,支持 BB 获得可扩展的高突发带宽;(2)将物理上松散分布的作业私有缓冲横向聚合成单一数据容器,按照应用的数据文件视图构造 BB 结点的协同关系,使其暂存大规模数据集,支持并发数据分析;(3)利用 BB 重组 I/O 请求,优化传统并行文件系统管理的 LCS 存储的性能。

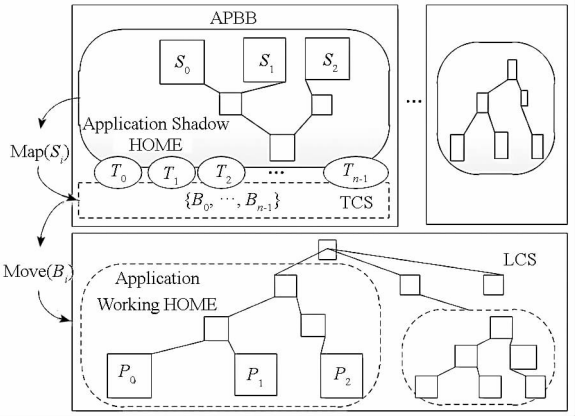

图 2　APBB 存储模型

Fig. 2　Storage model of APBB

4　APBB 关键设计

4.1　APBB 存储模型设计

APBB 没有像传统文件系统那样将所有 TCS 聚合成一个紧密的整体来提供服务,而是在应用层构造所需的存储视图。假设计算结点和 TCS 资源按比例均衡部署,单个作业内多个计算任务临近的 TCS 资源则定义为该应用的 APBB,该作业称为 APBB 的宿主。假设该作业的 I/O 并发度为 n,即 n 个任务并发 I/O,则 APBB 资源可细分并抽象为一个 TCS 资源集合 $B = \{B_0, \cdots, B_{n-1}\}$,$B_i$对应单个计算任务的突发缓冲且临近任务 T_i。

APBB 存储模型中,作业运行的主存储视图(即作业的工作目录树)处于 LCS 层,假设作业的数据集在该视图下对应 m 个文件,该视图可定义为文件路径集合 $P = \{p_0, \cdots, p_m\}$ 且 p_i具有唯一性。APBB 把 LCS 层的存储视图空间对等投影到 TCS 层,则构成应用程序的运行时影子加速空间 $S = \{S_0, \cdots, S_m\}$ 且 $S_i = P_i$,每个 APBB 存在映射规则 $Map(S_i)$,将 S_i映射到 TCS 物理资源 B_i,从而实现实体文件 P_i和突发缓冲物理资源的映射。APBB 通过数据移动规则 $Move(B_i)$,将缓冲数据 B_i推送到 LCS 层的实体文件 P_i。计算任务访问

某个文件时，APBB 根据该文件的 LCS 路径和映射规则，在 APBB 空间定位影子文件，假设 $Map(S_i)$ 无法在 APBB 中找到文件，则直接访问 LCS 层实体文件 P_i。基于以上规则，作业的数据集以混合且分层的形式分布在 H2IO 架构中，TCS 资源之间根据规则协同构成单一影子视图空间，I/O 请求存在三种时间代价，分别表示为临近 TCS 访问 C_{local}、非临近 TCS 访问 C_{remote} 和 LCS 层访问 C_{global}，且 $C_{local} < C_{remote} < C_{global}$。基于 TCS 存储的稀缺性、低延迟和高带宽特征，映射规则和移动规则是 APBB 发挥作用的关键。

APBB 由应用程序在运行时动态创建和拥有，具有以下优势：

(1) 应用成为 APBB 的宿主，无需文件系统对 TCS 进行复杂管理，整个 TCS 资源的管理开销分布到多个宿主作业，避免了单点管理常见的各类瓶颈问题，易于伴随计算规模扩展；

(2) APBB 绑定应用，资源竞争局限在作业内部，易于根据并行作业内部任务分布的天然均衡性实施有效的并发控制，避免请求竞争；

(3) APBB 可抽象为宿主应用的数据容器，容器内数据在存储空间和时间周期上具有关联性，容器内数据访问具有访问局部性。这些优势都有利于在大规模系统条件下发挥 TCS 的带宽优势。

4.2 基于并发度划分的影子空间映射

每个 APBB 拥有独立的影子空间，由 APBB 负责映射到 TCS 资源，以避免文件系统层资源调度的盲目性。映射规则是避免竞争的关键。APBB 采用基于任务域划分的映射策略避免访问竞争，即保证每个任务映射到对等的 TCS 资源。APBB 假设以下条件成立：(1) H2IO 架构下计算结点均衡匹配 TCS 资源；(2) 程序设计保证应用层的 I/O 请求均衡分布到并行作业的多个任务中。基于以上假设，即存在“I/O 请求→任务→结点→TCS”的均衡映射关系，映射函数 $Map(S_i)$ 仅需判断 S_i 的任务索引，即可将 S_i 均衡映射到 TCS 资源，避免不同任务竞争相同资源 B_i。APBB 使得应用程序能够指导存储资源的分配，资源的使用具有确定性，避免盲目调度产生的 I/O 冲突。

APBB 支持两类映射，一种称为独立映射，即任务的数据访问完全不相关，APBB 仅需构造任务自身的影子空间，每个任务有自己的私有映射函数 $Map(S_i)$，实现一对一的映射，这类映射适合加速并行输出场景。另一种为协同共享映射，即构造任务间可共享数据的协同缓冲空间，此时 APBB 的任务间采用一致的规则 $Map(S_i)$，保证 APBB 协同空间的视图一致性。传统并行文件系统的数据布局方法，例如条带、哈希、轮转和随机等数据布局规则，均可应用于构造共享映射。

4.3 基于并发度收敛的数据移动机制

当持续向 TCS 注入数据时，APBB 的可用容量逐渐降低，此时应用程序仅能获得 APBB 的持续 I/O带宽，并受限于将数据迁移到 LCS 的效率。

LCS 存储具有并发能力弱的特点[15]，适度并发才能保持其最优性能。APBB 是 I/O 请求被动接收者，如果仅采用请求聚合并转发[10-11]的方式，无法收敛并发度。APBB 根据预定义的 LCS 最佳并发度优化移动规则 $Move(B_i)$，在请求聚合重组的同时主动收缩并发度。假设作业 I/O 并发度为 n，LCS 最优并发度为 m，TCS 最优并发度为 k(通常 $m < k < n$)，则 APBB 设置 $p = \min[m,k]$ 个队列，并均分到 ION 上，每个队列由一个搬运工负责。由此 APBB 的移动规则将单次 n 个并发请求收敛为 n/p 次且每次 p 个并发请求的适度数据移动过程，该并发度适合从 TCS 向 LCS 移动数据。

4.4 局部性优化

APBB 把空间局部性 L_s 定义为对一组数据集的所有请求访问临近 TCS 的概率，把时间局部性 L_t 定义为一个时间窗口内请求访问 TCS 的概率。

APBB 通过优化映射规则来提高空间局部性。对于独立映射，APBB 在吸收突发数据输出时采用亲和调度策略，直接将任务 T_i 的文件数据分配到 APBB 资源集合中最近的 B_i 中，保证每个突发写请求代价均为 C_{local}。对于共享映射，APBB 采用两级映射的方法，文件数据依然亲和分配到最近的 B_i 中，但增加一级索引数据，索引数据按照共享映射规则分布，可以存放在远端 B_i。

为了实现时间局部性最大化，APBB 设计了基于数据意图的数据淘汰规则，根据应用程序的暗示确定数据是否被淘汰出缓冲。APBB 根据应用程序使用缓冲的意图，赋予每个 APBB 一个意图属性，用于定义需暂缓淘汰的数据，使得待分析的数据尽量停留在 TCS，使得后续处理处理流程中的读请求能落入 C_{local} 或 C_{remote} 代价范畴。通常数据分析过程中数据会分布在多个 ION 上，读取过程中尽管难以保证 C_{local} 的概率，但即使所有请求都落入 C_{remote} 范围，基于 TCS 的低延迟访问特性，也能使其性能远好于 LCS 访问。

5 APBB 原型实现

APBB 基于 H2FS 的基础支撑模块实现。

H2FS 是天河 - 2 上支撑多目标的 I/O 软件栈平台,其核心组件可以用来构建 I/O 中间件。如图 3 所示,APBB 基于 H2FS 的 RPC 模块作为计算任务向缓冲传输数据的高速通道,通过 H2FS 的 provider 存储模块在 SSD 上存放数据。每个 APBB 的公共信息存放在全局共享的存储空间内,记录 APBB 的私有属性和状态,例如 APBB 标识、作业标识、任务数、ION 集合、映射规则和数据意图等信息。

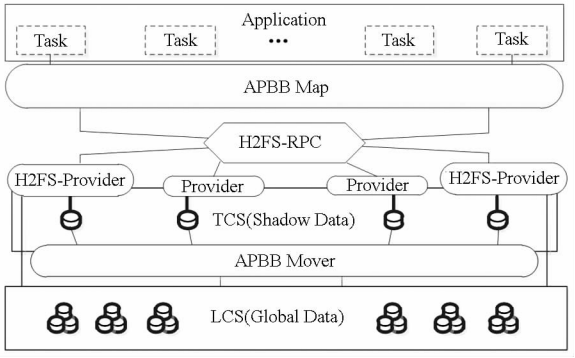

图 3　APBB 原型实现框架

Fig. 3　Prototype implementation framework of APBB

APBB 以动态库的形态存在,采用符合 POSIX 标准的文件接口作为缓冲入口,应用程序无须修改即可使用。Map 和 Move 规则分别嵌入到计算任务和 provider 的运行上下文,通过公共信息可配置。

作业创建 APBB 时通过聚合通信收集该作业内的 ION 集合,在 SSD 上分配并创建 TCS 资源集合 B 对应的数据文件,完成每个任务 T_i 到 B_i 的映射。影子空间的文件由数据和元数据构成,根据映射规则分布在多个 ION 上。每个 ION 在 APBB 中对应一个内部索引,元数据通过该索引记录数据文件所在位置。计算任务和 ION 间使用基于 RDMA 的用户态零拷贝传输协议,因此计算任务和 APBB 之间可以获得接近 MPI 点点通信的高传输带宽。

6　性能评估

使用 IOR 并行 I/O 基准测试程序模拟多种应用场景,评估 APBB 的突发写带宽扩展能力、持续写带宽能力和数据读带宽能力。测试平台为天河 - 2 系统的 HPC 分区,共 8192 个计算结点通过自主设计的高速网络互联[16]连接 ION,点点单向 MPI 带宽大于 6GB/s。测试中 TCS 为 256 个 ION,单个 ION 配置单个PCI-E SSD,持续带宽约 2.5GB/s。LCS 为 Lustre 集群存储系统,包括 64 个存储服务器连接 32 套盘阵,单套盘阵 60 个 SATA 磁盘,持续带宽约 2GB/s。

6.1　突发带宽扩展性评估

测试中模拟高并发度条件下的并发 I/O 负载,每个计算任务顺序读写一个大文件,数据总量不变,扩展任务数至 8192。测试中对应了 128 个 ION,分别评估 APBB 独立映射和协同共享映射的突发带宽扩展能力,共享映射采用两级散列映射。

如图 4 所示,两种映射模式在任务数小于 256 时,均表现出较好的扩展性,当任务数大于 256 时,因为文件数随任务数增加,文件变小,独立模式优于共享模式,这说明共享模式下 ION 之间维护映射索引代价对小文件性能存在部分性能影响,但对 HPC 应用中常见的大文件写带宽则影响不大。

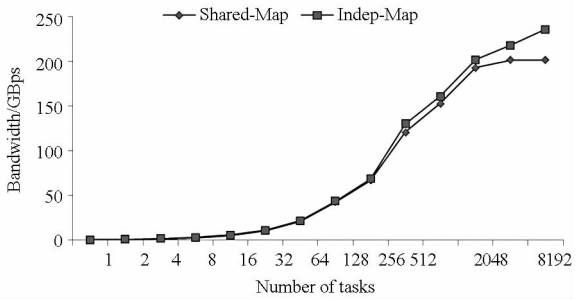

图 4　APBB 突发带宽

Fig. 4　Burst output bandwidth of APBB

此时独立模式比共享模式表现出更好的扩展能力。两者的曲线波动都不大,突发带宽均能够伴随随任务数扩展,表明 APBB 基于任务域的映射方法在并发 I/O 条件下具有较好的均衡性,对于场景 1 能带来大幅性能提升。

6.2　持续带宽能力评估

将 APBB 意图设为即时移动数据,并在文件关闭时等待迁移完成,对比测试 APBB 的持续输出带宽能力,测试数据量随任务数倍增。测试中 LCS 缩减为两套盘阵,增加请求竞争概率,以评估并发度收缩移动策略在高竞争条件的效果。

如图 5 所示,当任务数超过 256 时,大量请求竞争 LCS 资源,导致 Lustre 带宽下降明显。APBB 则表现较好,尽管数据移动的代价使其最大带宽低于直接访问 Lustre,但性能没有剧烈波动,这表明 APBB 的数据移动优化可避免高并发度带来的性能易变性,虽然多了 1 次数据移动,但抑制了 I/O 过程的过度并发,避免了请求竞争,有利于优化并行文件系统的性能波动问题。任务数小于 16 时 APBB 的持续带宽甚至高于 Lustre,这是 APBB 的请求聚合机制对性能带来的好处。

6.3　APBB 数据分析能力评估

测试中将 APBB 缓冲数据的意图设为待分

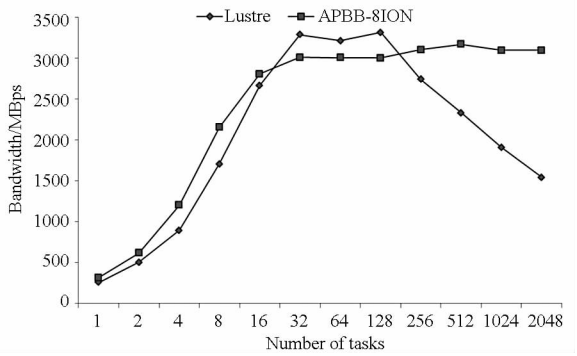

图5 APBB 持续写带宽

Fig. 5 Sustained output bandwidth of APBB

析,在 SSD 中保持前面写入的数据,对比评估 APBB 对于数据分析场景的好处。测试中按全系统计算和存储资源的比例,配置了 256 个计算结点和 2 套盘阵,并分别将数据映射到不同数量的 ION 上,评估 ION 个数对于读带宽的影响。如图 6 所示,4 个 ION 的持续带宽远超过 Lustre 系统,8 个 ION 情况下(即正常比例),APBB 的性能相对于磁盘系统性能提升 4 倍以上,并且随着 ION 数量增加,说明了 APBB 的影子空间映射技术充分地利用了固态盘资源来加速并发读取过程,有利于大规模数据分析场景。

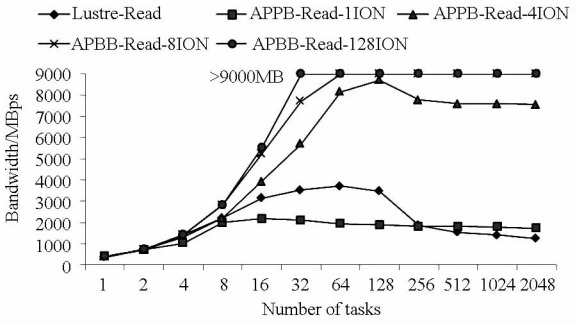

图6 APBB 读带宽

Fig. 6 Input bandwidth of APBB

7 结论

本文从中间件的角度着手优化使用 H2IO 架构中的加速资源,相对于传统文件系统单个名字空间的方法,APBB 专注系统中的计算存储动态耦合、协同数据访问和数据整体移动等方面的效率,简化了 H2IO 中多层次存储的使用复杂性,使其可以伴随计算规模水平扩展。APBB 采用运行时动态耦合的方法根据应用程序的暗示均衡分配使用缓冲资源,避免了竞争对性能的干扰,提高 SSD 资源的并发访问带宽。APBB 基于主动收缩并发度和提高时空局部性的方法,高效地在存储层次间移动数据,支持大数据集的并行分析。天河 -2 系统中的测试结果表明,该方法具有极佳的扩展性,能利用加速存储提供高 I/O 带宽。APBB 影子空间的方法,屏蔽了混合架构的层次,可被并行应用透明使用,可让大量的应用模式直接收益。

参考文献(References)

[1] Braam P. Lustre file system[EB/OL]. 2005[2014 - 01 - 18]. http://www.lustre.org.

[2] Bent J, Gibson G, Grider G, et al. PLFS: a checkpoint filesystem for parallel applications[C]//Proceedings of the 2009 International Conference for High Performance Computing, Networking, Storage and Analysis, 2009:1 - 12.

[3] Seelam S, Kerstens A, Teller P J. Throttling I/O streams to accelerate file-IO performance[C]//Proceedings of Third International Conference, High Performance Computing and Communications, 2007: 718 - 731.

[4] Chen Y. Towards scalable I/O architecture for exascale systems[C]//Proceedings of the 2011 ACM International Workshop on Many Task Computing on Grids and Supercomputers, 2011: 43 - 48.

[5] Xie B, Chase J, Dillow D, et al. Characterizing output bottlenecks in a supercomputer[C]//Proceedings of the 2012 International Conference for High Performance Computing, Networking, Storage and Analysis, 2012: 1 - 11.

[6] Lofstead J, Liu Q, et al. Managing variability in the IO performance of petascale storage systems[C]//Proceedings of the 2010 International Conference for High Performance Computing, Networking, Storage and Analysis, 2010: 1 - 12.

[7] Dillow D A, Shipman G M, Oral S, et al. I/O congestion avoidance via routing and object placement[C]//Proceedings of Cray User Group Conference (CUG), 2011.

[8] Ghemawat S, Gobioff H, Lueng S, et al. The google file system[C]//Proceedings of the 9th ACM Symposium on Operating System Principles, 2003:29 - 43.

[9] White T. Hadoop: the definitive guide[M]. Cambridge, 2009.

[10] Iskra K, Romein J W, Yoshii K, et al. ZOID: I/O-forwarding infrastructure for petascale architectures[C]//Proceedings of the 13th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, 2008: 153 - 162.

[11] Ali N, Carns P, Iskra K, et al. Scalable I/O forwarding framework for high-performance computing systems[C]//Proceedings of the Conference on Cluster Computing and Workshops, Piscataway, 2009: 1 - 10.

[12] Liu N, Cope J, Carns P, et al. On the role of burst buffers in leadership-class storage systems[C]//Proceedings of the 28th Symposium on Mass Storage Systems and Technologies, 2012: 1 - 11.

[13] Lofstead J F, Klasky S, Schwan K, et al. Flexible io and integration for scientific codes through the adaptable I/O system[C]//Proceedings of the 6th International Workshop on Challenges of Large Applications in Distributed Environments, 2008: 15 - 24.

[14] Carns P, Harms K, Allcock W, et al. Understanding and improving computational science storage access through continuous characterization[J]. ACM Transactions on Storage, 2011, 7(3): 8 - 22.

[15] Polte M, Simsa J, Gibson G. Comparing performance of solid state devices and mechanical disks[C]//Proceedings of the 3rd Workshop of Petascale Data Storage, Austin, 2008.

[16] Xie M, Lu Y, Liu L, et al. Implementation and evaluation of network interface and message passing services for TianHe-1A supercomputer[C]//Proceedings of 19th Annual Symposium on High Performance Interconnects, 2011: 78 - 86.