|  |  |
| --- | --- |
| 学号 | 2017282110214 |

**海量存储技术课程期末作业**

|  |
| --- |
| **LADS: Optimizing Data Transfers Using**  **Layout-Aware Data Scheduling读后感** |

|  |  |
| --- | --- |
| 院（系）名 称 ： | 计算机学院 |
| 专 业 名 称 ： | 计算机技术 |
| 学 生 姓 名 ： | 钟姗杉 |

二〇一七年十二月

目 录

[1 论文概述 1](#_Toc500619128)

[1.1 作者介绍 1](#_Toc500619129)

[1.2 论文背景 1](#_Toc500619130)

[1.3 论文大致内容以及整体架构 2](#_Toc500619131)

[1.3.1 论文大致内容 2](#_Toc500619132)

[1.3.2 整体架构 3](#_Toc500619133)

[1.4 阅读思路 3](#_Toc500619134)

[2 论文前期准备 4](#_Toc500619135)

[2.1 定义目标 4](#_Toc500619136)

[2.1.1 应用环境 4](#_Toc500619137)

[2.1.2 分布式文件系统 4](#_Toc500619138)

[2.2 确定问题以及优化目标 6](#_Toc500619139)

[2.2.1 核心问题 6](#_Toc500619140)

[2.2.2 Lustre中文件的存储 7](#_Toc500619141)

[2.2.3 具体问题分析 7](#_Toc500619142)

[2.2.4 问题对应解决方法 9](#_Toc500619143)

[3 论文实现部分 10](#_Toc500619144)

[3.1 LADS实现 10](#_Toc500619145)

[3.1.1 LADS架构 10](#_Toc500619146)

[3.1.2 LADS中数据的传输 11](#_Toc500619147)

[3.2 具体调度 12](#_Toc500619148)

[3.2.1 布局感知调度实现 12](#_Toc500619149)

[3.2.2 拥塞感知调度实现 13](#_Toc500619150)

[3.2.3 在不拥塞环境中基于对象与基于文件的方法 13](#_Toc500619151)

[3.2.4 在拥塞环境中基于对象与基于文件的方法 16](#_Toc500619152)

[3.3 进一步解决拥塞状况 17](#_Toc500619153)

[3.3.1 SSD上的对象缓存 17](#_Toc500619154)

[3.3.2 在拥塞环境下使用Flash进行基于发送端的缓存技术 18](#_Toc500619155)

[4 感想及问题 20](#_Toc500619156)

# 论文概述

我选择的论文是发自于2015年FAST会议的Optimizing Data Transfers Using Layout-Aware Data Scheduling。

## 作者介绍

这篇论文的作者有四名：Youngjae Kim, Scott Atchley, Geoffroy R. Vallée, and Galen M. Shipman，来自于美国的Oak Ridge National Laboratory(ORNL)。该实验室是能源与马丁·玛丽埃塔能源系统公司共同管理的多计划能源实验室。研究活动侧重如下四大领域：核聚变能源开发、磁聚变能源、生物与环境科学和基础能源科学（与国家能源目标有关的新知识开发）。这个实验室拥有着顶级的计算实验设施Oak Ridge Leadership Cpmputing Facility(OLCF)，其中的高性能计算机Titan，曾经在全球高性能计算计算机中排名第一，目前排名全球第三，前两名分别为中国的神威太湖之光和天河二号。

此次论文的选题与这个实验室工作内容有着紧密的联系：实验室的研究人员会将从实验设施产生的，或者在超级计算机上模拟而得的数据进行分析，以便于验证他们的结论，由于这些数据的来源可能是来自于其他组织，所以这一过程需要在地理上分散的组织之间移动数据，所以论文着重研究了如何提升传输数据的速率并实现了相应框架。

## 论文背景

正如上文介绍，实验室希望能加快不同组织设施之间数据的传输速度，特别是在面对如今日渐增长数据量。作者分析了目前人们采取的解决方法：网络运营商努力提升网络的吞吐量，存储需要传输数据的并行文件系统使用单独的服务器来并行地处理元数据和I/O，并且这使用几十到几百个I/O服务器，每个服务器都有数十到数百个磁盘，以提高性能和容量的可扩展性。

然而，网络速度达到太比特速度，PFS也达到了EB级，但仍然存在着较大的提升空间。

首先，这些网络改进仅有助于提高网络数据传输速率，而不是从源存储系统到接收存储系统的端到端数据传输速率，换句话说，就是道路畅通了，但是货车的装货卸货速度太慢了。其次PFS增大了硬件上的投入，增多了服务器，但是运行机制上存在着欠缺。依据大货车的类比，这个解决方法意识到了货车装卸货能力不高，雇佣更多工人参与装卸货物，但由于存储货物的工厂进出存在着限制导致工人不能同时进出工厂等工厂设施硬性要求，让这些工人的效率不能完全发挥。因此，作者就是根据工厂设施的硬性要求以及工人搬运货物时的机制入手考虑解决问题。

## 论文大致内容以及整体架构

论文大致内容以及整体文章结构如下：

### 论文大致内容

论文专注于优化端到端数据传输，并调查应用程序，网络协议和存储系统在源主机和宿主机之间的交互，开发了Layout-Aware Data Scheduler（LADS），种使用CCI接口进行通信的PFS之间使用的批量数据移动框架。

其中LADS是使用文件的物理视图，而不是逻辑视图，所以它理解文件存储的底层结构，比如它知道文件由若干个数据对象组成，知道存储目标保存一数据对象，也知道存储服务器和存储目标之间的拓扑结构，这一性能显著区分了它与其他文件存储系统。因此，LADS能根据文件布局来调度文件数据进行传输，其中，LADS将所有读写操作对齐到PFS中的底层对象大小。 而且，LADS允许无序的对象传输（这么做的具体原因后便会有分析）。

在LADS中，作者实现了两大算法：一个是布局感知的I / O调度算法。主要思想是通过避免拥塞的服务器以及专注于空闲的服务器来减少由于拥塞的存储目标造成的停滞I /O时间。而其他现有的数据传输工具多个线程读取文件时，只专注于存储自己文件所在的服务器，无论服务器是否忙碌。另一个算法是拥塞感知I / O调度算法，它可以提高每个线程的数据处理速度，从而提高数据传输速率。

为了进一步避免缓解拥塞，尤其避免是在存储器上的I / O负载动态变化的环境中由于拥塞而导致存储目标工作缓慢的情况，作者在还实现和测试过程中还使用非易失性存储器（NVM）设备进行分层数据传输。

最后，论文比较了LADS与广泛使用的数据传输程序bbcp的性能。 具体来说，在对基于真实文件分发的工作负载的评估中，作者观察到当在一个节点上使用八个线程时，LADS产生的数据传输速率比bbcp高4-5倍。 此外，作者发现，使用少量的SSD，与没有SSD缓冲的基线相比，LADS可以将数据传输速率进一步提高37％，并且比提供额外的DRAM更具成本效益。

### 整体架构

第一章介绍背景需求；

第二章分析目标环境的系统架构以及相关文件系统，定义要解决的问题；

第三章实现LADS;

第四章测试评估LADS;

第五章介绍作者所做的相关工作

第六章总结

## 阅读思路

大致浏览文章的主要思想，确定着重了解学习第二，三，四章，其中第三章每提出一个实现方案，都在第四章的测试评估结构分析讨论，所以我打算先让第二章介绍作者如何发现问题以及解决问题的角度，然后让第三章分别介绍作者的实现方案以及对应评估实验结果，最后一章列出仍然无法想明白的问题。

# 论文前期准备

## 定义目标

分析应用环境并确定具体问题及目标。

### 应用环境

作者分析了他们实验室设施的整体系统架构。其与其他能源部实验机构一样，拥有大型高性能计算系统（例如OLCF的Titan）以及科学仪器（例如ORNL的SNS），可以生成大量的同步I/O操作。 HPC系统需要利用科学仪器产生的数据进行计算模拟实验，其工作时的具体任务有：计算模拟，然后是进程间通信，并定期通过I/O来保存当前运行状态或保存输出文件。计算模拟阶段启动时主要的任务是通过读取阶段来检索输入数据文件以及应用程序二进制文件和库文件。而科学仪器没有读取阶段，但具有写入其所捕获测量数据的阶段。这些测量是由周期性事件触发的，例如加速粒子撞击产生各种能量和子粒子的目标。仪器的探测器将捕获这些事件，并且必须在下一个事件前将数据移出设备。

为了保存上述设备产生或者运行需要的数据，这些实验机构的设备系统通常具有由高性能网络连接的大型并行文件系统（PFS）。因此他们实验室以OLCF为站点设置了PFS，可被多个HPC系统以及分析和可视化群集访问。在这种情况下，HPC系统是主要用户，集群是次要用户。考虑到运行这些较大资源的成本，要尽量避免由于次要用户的I/O而对HPC系统的性能产生负面影响。

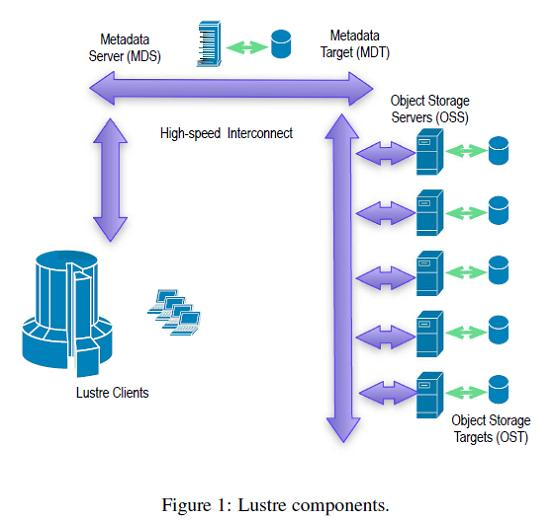
最后，大多数使用这些系统设施的科学家并不在该实验室内，而是来自其他能源部站点，大学以及一些商业实体。他们希望将实验所得数据转移回其所在机构以进一步分析数据。因此，每个能源部站点都有几个装置了PFS的数据传输节点（DTN），并连接到能源部的能源科学网络（ESnet），数据通过网络传到科学家所在站点，以便实现资源共享。与分析和可视化集群一样，DTN的使用不应该对大型HPC系统的I/O产生负面影响。

### 分布式文件系统

这里我以一个广泛应用的分布式文件系统Luster为例来介绍该系统的系统架构。Lustre是一个以GNUGeneral Public为许可证的，开源的分布式并行文件系统，由Sun Microsystems Inc.公司开发和维护。Lustre为其客户端提供了包含对共享文件对象的并行存取能力在内的POSIX接口。根据Top500的数据，在全世界前30的超级计算机大多数都采用了Lustre文件系统。

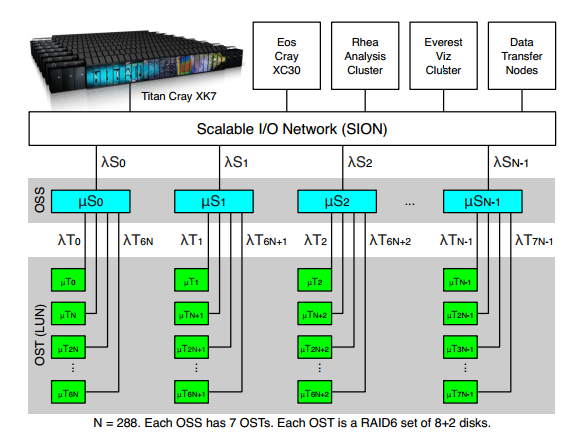
Lustre是一个面向对象的文件系统。它由三个部件组成：元数据服务器（Metadataservers, MDSs）、对象存储服务器（objectstorage servers, OSSs）和客户端。图1给出了文件系统的体系结构。Lustre使用块设备来作为文件数据和元数据的存储介质，每个块设备只能由一个Lustre服务管理。Lustre文件系统的容量是所有单个OST的容量之和。客户端通过POSIX I/O系统调用来并行访问和使用数据。

图1 Lustre文件系统的体系结构



OLCF的PFS是OLCF的第二代Lustre系统SpiderII。 其主要客户是Titan。 Titan有18688个直接挂载Spider的计算节点，而Titan的 I/O流量通过432个I/O节点，在Titan的Cray Gemini网络和OLCF的InfiniBand（IB）可扩展I/O网络（SION）之间充当（LNET）路由器。 除Titan外，Spider还与Eos，一个拥有744节点的Cray XC30系统，分析和可视化集群以及DTN共享。 图2提供了Spider的概述。目前，Spider文件系统可通过两个名称空间atlas1和atlas2访问，以实现负载平衡和容量管理。 每个命名空间有144个OSS，每个命名空间管理7个OST，每个命名空间总共有1008个OST。 每个OST代表10个（8 + 2）磁盘的RAID-6集合。

图2 OLCF center-wide PFS and clients

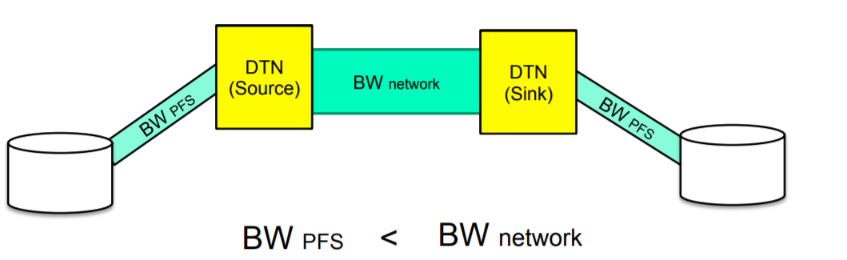


## 确定问题以及优化目标

### 核心问题

根据第一章分析的论文背景以及第二章介绍各大实验所所采用PFS系统结构，我们可以发现这一现象，如图3所示，连接着这些文件系统和广域网的数据传输节点DTN成为快速网络和相对较慢的系统之间的阻抗匹配的焦点。

图3 快速网络和较慢的系统之间的DTN



正因如此作者把关注点集中在PFS，并提出核心问题：如何在大数据传输中将未充分利用的PFS带宽充分利用。

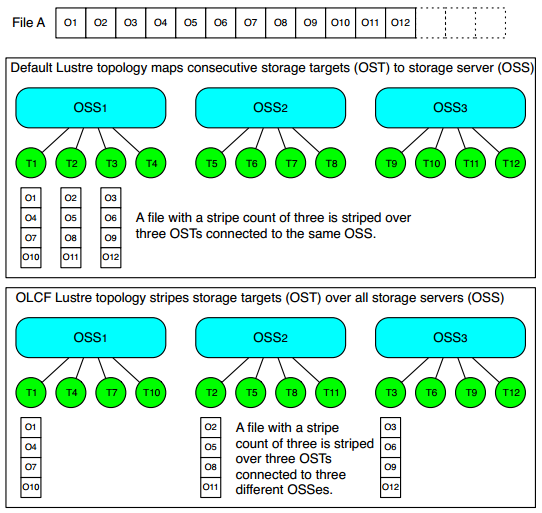
他们在观察研究中发现：PFS被视作单独的命名空间，然而，其并不是只有单独磁盘构成，而是具有服务器的多个磁盘的阵列；其次，PFS虽然正是为了并行I/O处理而设计出来的，但是传统数据传输工具不能充分利用PFS上I/O的可用并行性。

### Lustre中文件的存储

既然核心问题在PFC上，我们需先来了解其中文件是如何存储的。

在Lustre中，文件的数据存储在一组对象中，每个对象存储在单独的OST上。 OST到OSS的映射会影响文件对象的存储方式。图4显示了OST到OSS的映射如何物理地影响文件的对象位置。默认映射是将OST顺序分配给OSS。其中文件A被分为3组对象集合，每个OSS拥有4个OST，这些对象集合将分别存储在连接到一个OSS的3个OST上。然而OLCF所使用映射使得OST被分配给所有OSS。在图中，文件A的3组对象集合分配给三个OST，每个OST连接到不同的OSS。根据存储和网络硬件的选择，OSS或OST可能是瓶颈。为了通过最小化争用来提高I / O吞吐量，较高层需要这种信息。

图4 Lustre中文件存储方式



### 具体问题分析

既然了解了PFS中文件是如何存储的，我们可以发现PFS非常容易出现拥塞现象：

当多个I/O请求同一个服务器并超过该服务器的能力时，存储服务器会出现暂时拥塞，在这期间服务每个新请求的时间会增加。具体来说，例如当大型应用程序进入其I/O阶段（例如，写入状态点，在启动时读取共享库）或多个应用程序正在共同访问文件位于OST的一个子集上。虽然操作系统缓存和应用程序级存有时可以屏蔽许多应用程序的拥塞，但是数据传输工具不能从这些技术中受益。如果在传输的发送端出现拥塞，则发送端网络缓冲区将会逐渐耗尽并最终停止。另一方面，接收端的拥塞将导致接收端和发送端的缓冲区都被填满，最终导致发送端的I/O线程停止工作。

拥塞现象将会极大的降低数据传输带宽，因此需要寻找导致拥塞的原因：

（1）从系统自身角度分析——两级瓶颈

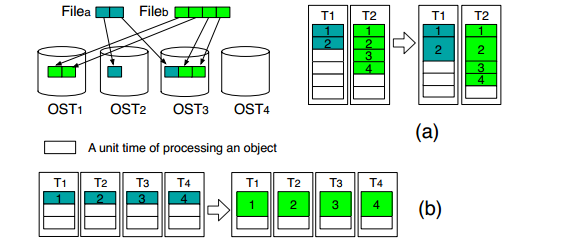
Lustre文件系统中通过OSS访问OST时，I / O瓶颈的可能位置。 对于OSSm，如果到达速率（λOSSm）大于其服务速率（μOSSm），服务器将开始溢出，成为瓶颈，其到来的服务将被延迟。如果连接到OSS的OST的数量大于与OSS的网络连接可处理的数量，则可能发生这种情况。为了避免这种情况，OLCF规定每个OSS的OST数量，使μOSSm>。然而即使λOSSm小于μOSSm，OST也可能成为瓶颈。例如，如果λOSTj大于μOSTj，则OSTj成为瓶颈。

（2）从使用者角度——基于文件的方法的传输

传统的文件传输工具依赖于文件的逻辑视图（称为基于文件的方法），它忽略了底层的文件系统架构。这样容易出现I/O线程竞争现象，我们可以分两种情况来看：第一种是一个I/O线程分配给一个完整的文件，此时直到整个文件被读取或写入前将会一直在这个文件上工作；第二种是使用多个线程在同一个文件上，直至文件操作完毕。

图5演示基于文件的方法（它不知道底层的文件系统布局）分别从上述两种情况演示该方法如何导致PFS中的I/O争用问题，我们假设每个OST可以在一个固定的服务时间内一次服务一个对象。 在图中，文件A被分布在OST2和OST3上，文件B被分布在OST1和OST3上。在图5（a）中，线程1（T1）和线程2（T2）分别尝试同时读取文件A和文件B。T1和T2读取不同的文件，但是T1和T2可以在访问同一个OST时互相干扰。当T1和T2完成访问OST3时，T1和T2分别减速25％和12.5％。在图5（b）中，所有四个线程访问同一个文件B的不同逻辑区域，但是，由于T1和T2都访问OST1，T3和T4都访问OST3。 因此，每个线程减慢了50％。 这个例子的结果表明，由于缺乏对文件对象物理布局的理解，OST竞争可能会增加。

图5 基于文件的方法的传输



### 问题对应解决方法

（1）针对两级瓶颈问题

为了使存储拥塞对整体I/O性能的影响最小化，作者尝试降低停滞I/O的存储占用率以及增加缓存来缓解，通过I/O线程不被分配给过载的服务器或目标的方法，避免对象存储服务器和对象存储目标瓶颈。

（2）针对基于文件的方法的传输的缺陷

从基于文件系统体系结构的物理角度来查看整个工作负载，根据文件的物理分布进行负载平衡，且让一个线程可以被分配给任何OST上的任何文件的一个对象，而不需要在另一个文件的对象之前传送一个特定文件的所有对象，允许无序的对象传输。

# 论文实现部分

基于以上问题以及解决方案，作者提出了LADS架构，并实现了改进的并行性，网络可移植性，以及拥塞调度。这里我将首先介绍LADS的实现，接着介绍几个调度算法并结合对应测试结果分析

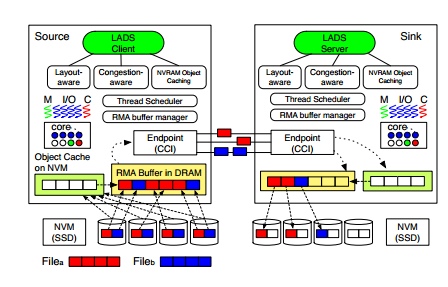
## LADS实现

### LADS架构

LADS由以下线程组成：主线程保持传输状态，而I/O线程读写文件对象到PFS。 通信线程负责接收端和发送端之间的所有数据传输。在具体实现中，有一个主线程，一个可配置数量I/O线程和一个通信线程。因为I/O线程使用阻塞调用，所以允许比内核更多的线程（即超额订阅）。由于我们可以超额订阅内核，线程和I/O线程在空闲或等待资源时会阻塞。通信线程永远不会阻塞，并总是尝试进行通信。通信线程生成驱动应用程序的大部分事件。如果通信线程需要一个无法立即获得的资源（例如缓冲区），它将在主线程请求队列中排队请求并唤醒主线程。

几个I/O优化技术已经在LADS中实现。在该图中布局感知的技术可以优化I / O线程访问的数据的单位大小，使之在底层文件系统中达到对象大小，并且在服务器拥塞时改善了停滞的I/O时间。OST拥塞感知算法可以避免拥塞的服务器。当RMA缓冲区被缓存对象全部用完时,NVM可以用作扩展内存区域。通信线程在发送端和接收端，使用CCI，保留内存区域给在发送端和接收端的通信线程之间的RMA传输。在发送端，如果RMA缓冲区已满，主线程会通知I/O线程使用NVM缓冲区，而不是直接从PFS复制对象到RMA缓冲区，因此，它允许在扩展内存区域预加载在NVM上。在接收端，如果RMA缓冲器已满，同样，扩展的NVM区域可以在PFS之前用作中间缓冲器，以避免拖延网络传输。

图6 LADS架构



### LADS中数据的传输

对于传输文件，首先发送端和接收端进程需要初始化某些状态，产生线程并交换一些信息。主线程初始化其工作队列，等待队列和I / O线程的等待队列。

通信线程打开CCI端点，分配其RMA缓冲区并将其注册到CCI，并打开与远程对等体的连接。 发送端通信线程发送其最大对象大小，RMA缓冲区中的对象数量以及RMA缓冲区的内存句柄。 接收端通信线程接受连接请求，触发送端上的CCI连接事件。 I/O线程只是等待其他线程。

在CCI初始化步骤之后，数据传输将在发送端和接收端处遵循这些步骤：

步骤1.对于每个文件，发送端的主线程将打开文件，确定文件的长度和布局（即存储对象的大小和它们所在的OST），并生成一个NEW FILE请求并将请求排在通信线程的工作队列上。通信线程为每个存储对象生成NEW BLOCK请求，并将该请求排入适当的OST工作队列。通信线程将封送NEW FILE请求并将其发送到接收器。

步骤2.在接收端，通信线程将收到NEW FILE请求，并将其列入主线程的工作队列，并将其唤醒。 主线程将打开该文件，将文件描述符添加到请求中，将请求类型更改为FILE ID，并将请求排列在通信线程的工作队列上。通信线程将把它出队并发送给发送端。

步骤3.在发送端中，当通信线程收到FILE ID消息时，它将唤醒N个I/O线程，其中N是文件被分割的OST的数量。 一个I/O线程首先保留一个注册到RMA的CCI的缓冲区。 然后确定应该访问哪个OST队列，然后将第一个NEW BLOCK请求出队。 它使用pread( )函数将数据读入RMA缓冲区。当读取完成时，它将请求排入通信线程的工作队列。 通信线程编组请求并将其发送到接收端。 值得注意的是，发送端的通信线程的工作队列将打乱NEW FILE和NEW BLOCK请求，从而重叠FILE ID交换和块请求。

步骤4.在接收端，通信线程收到请求，并尝试预留一个RMA缓冲区。 如果成功，它会启动一个RMA的读取程序。否则，将请求排入主线程的工作队列并唤醒主线程。主线程将在RMA缓冲区的等待队列中休眠，直到缓冲区被释放。 然后，它将在通信线程的队列上排队请求，通信线程将会读取到RMA中。

步骤5.在接收端处，当RMA读取完成时，它发送一个BLOCK DONE消息回发送端。接收端的通信线程通过该块的文件偏移量来确定适当的OST，并将其排列在OST的工作队列上。然后它唤醒一个I/O线程。I/O线程寻找下一个要服务的OST，将请求出队，调用pwrite( )函数将数据写入磁盘。写入完成后，它将释放RMA缓冲区，以便通信线程可以启动另一个RMA读取。

步骤6.当发送端的通信线程接收到BLOCK DONE消息时，它释放RMA缓冲区并唤醒一个I/O线程。这种模式一直持续到所有的文件块都被传送完毕。当所有块被写入时，源发送FILE DONE消息并关闭文件。当接收器收到该消息时，也会关闭该文件。

## 具体调度

### 布局感知调度实现

在PFS中，文件存储为对象集合，并存储在多个服务器中，以提高整体I/O吞吐量。访问PFS的最佳方法是应用程序发出大量请求，以获得跨多个服务器并行访问的好处。根据传统文件访问方法中，正如前一章所述单个文件分配单个线程访问，该线程将请求N个对象，并且可以同时并行读取M个对象（假设M <N，并且该文件被划分到M个服务器上）。但是，如果其中一台服务器拥塞，请求的持续时间由最慢的服务器决定。所以N个对象请求的吞吐量是由拥塞服务器的对象吞吐量决定的。在LADS中，由于作者将所有I/O访问与对象边界对齐，所以不用单个线程来请求N个对象，而是有N个线程分别从不同的服务器请求一个对象。如果其中一个请求被拥塞的服务器延迟，则N-1线程可以自由地向其他服务器发出新的请求。到缓慢服务器的请求完成时能检索到更多个对象。

虽然对齐访问技术旨在减少I/O停滞时间并提高整体吞吐量，但并不指定向哪个服务器发送请求。 大多数数据传输工具试图一次移动一个文件（例如bbcp，XDD）或一个小的子集（例如GridFTP）。然而，在PFS中，单个文件被划分到N个服务器上。在ORNL的Atlas文件系统的情况下，默认是四个服务器。虽然文件系统可能有数百个存储服务器，但大部分数据移动工具一次访问它们的一小部分。 如果其中一台服务器拥挤，整个性能将在拥塞的时期受到影响。

### 拥塞感知调度实现

对于拥塞感知I /O调度，我们试图避免拥塞的存储服务器。给定一组文件，确定发送端中读取所有对象所在的位置，或者确定在接收端上写入到哪些服务器的对象上。然后，根据对象的位置安排访问，而不是基于文件。论文在特定的OST队列中排队请求一个特定的对象。然后，I/O线程以循环方式选择一个队列，并将第一个请求出队。如果另一个线程正在访问OST，则其他线程跳过该队列并转到下一个。如果一个OST拥塞，一个线程可能会停顿，但其他线程可以自由移动到其他非拥塞的服务器。

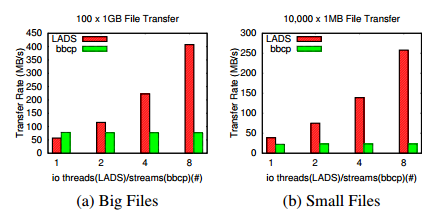
所有OST上的基本OST队列和简单的循环调度能够提高整体I/O性能。然后，通过实施启发式算法来检测和避免拥塞的OST，从而将布局感知调度扩展到拥塞感知。该算法可以为选择下一个I/O线程将工作的存储目标做出主动决策。该算法使用基于阈值的节流机制来进一步减少对HPC系统使用PFS的影响。例如，当在发送端读取时，I/O线程从其相应的服务器读取对象并记录读取时间，并且在预设时间窗口时间（W）期间计算多个对象读取时间的平均值。如果W期间的平均读取时间大于预设的阈值（T），则将服务器标记为拥塞。该算法告诉线程他们应该跳过拥塞的服务器M次。因此，I/O线程可以在短时间内避开拥塞的服务器，从而减少I/O延迟时间。

### 在不拥塞环境中基于对象与基于文件的方法

论文在不拥塞环境下比较了LADS与bbcp文件传输。在LADS中，作者动态调节I/O线程的数量，可以最大限度地提高数据传输节点上的CPU利用率，然而却使用单个通信线程。LADS使用CCI的Verbes传输接口，也就是本身使用底层的InfiniBand互连。在bbcp中，作者只能调整TCP / IP流的数量来提高性能（bbcp总是使用单个I/O线程），这些流通过相同的InfiniBand互连，但使用支持传统套接字的IPoIB接口。论文在确保网络永远不是测试的瓶颈的前提下进行测试。在bbcp中，论文使用带宽延迟乘积的公式计算TCP窗口大小（W）：使用ping时间（Tping）和网络带宽（Bnet）如下：W = Tping×Bnet，据此在测试中设置中使用了10 MB的TCP窗口大小。论文也通过改变块大小来测试bbcp，但是由于1 MB和4 MB之间几乎没有性能差异，所以论文用bbcp测试的块大小为1 MB来显示结果。

就数据传输速率而言，在图7(a) (b)中，随着I/O线程数量的增加，LADS显示出几乎完美的线性缩放比例，而bbcp随着TCP / IP流增加提高并不显著。 bbcp是通过基于文件的数据传输协议实现的，其中文件逐个传输，多个TCP流在同一个文件上运行。因此，瓶颈是由PFS扫描文件的宽度决定的。论文还发现，使用bbcp时，与存储的I/O带宽相比，网络速度适度缓慢时，多个TCP / IP流才能提供性能提升。总体而言，除了LADS传输使用一个I/O线程作为大文件集的结果之外，图7中所有测试用例的LADS都显着优于bbcp。在这种情况下，论文认为bbcp正在受益于测试平台预读。 LADS不会从中受益，因为I/O队列的循环访问可能意味着在下次访问此OST时正在访问来自不同文件的对象，这样会失去了预读的好处。由于这个原因，OLCF生产系统禁用预读。

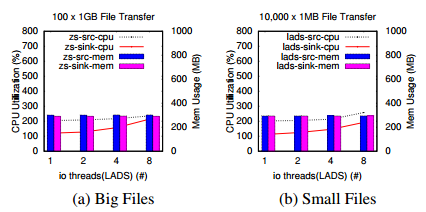
图7 传输大文件和小文件集的比较速率



对于及资源占用率，LADS使用DIRECT IO作为发送端的读取操作，以尽量减少CPU和内存的资源利用率，而接收器使用缓冲的I/O写入。如图8(a) (b)所示，LADS适度地使用系统资源，随着I/O线程数量的增加，CPU利用率仅略有增加。虽然I/O线程越多将涉及更多的Lustre文件系统元数据服务请求和更多的I/O，但是，总的来说，LADS利用InfiniBand网卡的卸载能力并管理好系统资源，即使在8个I/O线程中，CPU利用率仍然相对较低。内存使用从不变化，几乎恒定在280-300 MB，而发送端和接收端使用了256 MB的RMA缓冲区，占了大部分的内存使用量。

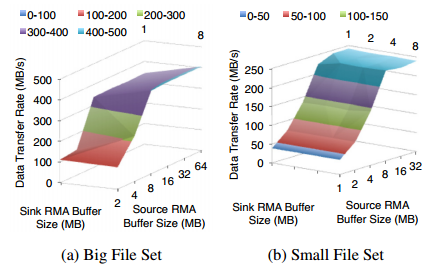
相反，在bbcp中，我们可以发现CPU和内存的使用率非常低。例如，对于小文件工作负载，当使用八个流时，其内存使用率和CPU利用率分别小于2 MB和5％。 对于大文件工作负载，使用相同的配置，bbcp的内存使用率和CPU利用率最多分别大约为30MB和40％。这正是因为bbcp采用基于文件的方法，导致磁盘I / O是瓶颈，所以主机资源不能被充分利用。

图8 传输大文件和小文件集的比较资源占用率



讨论到资源占用率，论文开始探究如RMA要设计多大才合理。图9显示了在LADS上发送端和接收端的可用RMA缓冲区大小的影响。在发送端，一个较大的RMA缓冲区减少了I/O线程在RMA缓冲区中等待一个缓冲位的时间，从而提高了数据传输速率。在接收端，在I/O线程忙于向OST写入数据块的同时，RMA缓冲区的容量可以保存更多的数据，减少I/O线程等待的时间。有趣的是，随着RMA缓冲器尺寸的增加，LADS的性能并不总是提高。具体来说，作者发现：一方面，接收器处设置几个RMA缓冲区的缓冲位（几兆字节）足以达到最大数据传输速率。另一方面，随着在发送端增加RMA缓冲区大小，LADS性能得到改善。这是因为在接收端允许Buffered IO，因此写入到磁盘可以是快速的，而在发送端，磁盘读取带宽是瓶颈，因为DIRECT IO用于在发送端存储器上读取数据。针对以上发现，论文决定在发送端添加更多的RMA缓冲位以提高数据读取性能比增加接收器处的RMA缓冲器大小更有利。

图9 传输大文件和小文件集的比较RMA大小



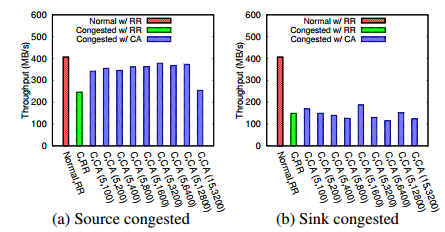
### 在拥塞环境中基于对象与基于文件的方法

论文在拥塞环境下比较了LADS与bbcp文件传输。图10显示了在正常和存储拥塞环境中总共传输100 GB数据的运行时间比较结果。

对于图10（a），我们测试了各种参数设置，以查看当发送端存储部分拥塞时我们的轮询调度算法的有效性。总的来说，我们看到，在遇到拥塞时，轮询调度算法性能比拥塞感知调度算法性能提高35％。性能改善的范围可以根据拥塞阈值的函数以及在拥塞服务器上跳过的次数来确定。我们注意到，如果阈值设置得太大，或者如果拥塞服务器的跳过次数设置得太小或太大，则该算法可能做出误判的决定，否定了避免拥塞磁盘带来的性能增益。

图10（b）显示了接收端PFS的拥塞时的情况，我们可以发现拥塞对于接收端影响远远高于发送端服务器拥塞的情况。令人惊讶的是，拥塞感知调度算法几乎不会改善性能，显示的执行时间与轮询调度算法一样高。这主要是因为拥塞感知调度算法未能检测到拥塞的服务器。拥塞算法测量每个对象的I/O服务时间，但是在测试中接收端使用Buffered IO阻止了它准确地测量OST的实际拥塞水平。作者经过评估证实，大部分预测是误报，经常错误分配I/O线程到繁忙或过载的OST。

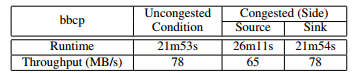
图10 拥塞情况下比较拥塞调度算法和轮询调度算法



论文还测量了拥塞状况下的bbcp的吞吐量，结果如表1所示。我们可以发现bbcp在运行时的运行时间拥塞与正常情况下的情况没有多大区别，这很可能是由于OS缓冲区缓存和bbcp较慢的通信吞吐量。很显然，用于写入的Buffered IO应该能够隐藏磁盘写入延迟。另一方面，当来发送端发生拥塞时，bbcp的运行时间比正常情况下增加了19％。而且，bbcp使用套接字引起额外的副本，包括用户对内核上下文切换以及TCP / IP堆栈处理。较慢的网络吞吐量掩盖了接收端磁盘拥塞。LADS明确地从可用的零复制网络中受益。

表1 bbcp的运行时间和吞吐量

拥挤的环境



## 进一步解决拥塞状况

在LADS架构图中我们可以发现作者采取其他手段来缓解拥塞状况。

### SSD上的对象缓存

在接收端发生广泛拥塞（即每个I / O线程正在访问拥塞的服务器）的情况下，新到达的对象将快速填充RMA缓冲器。 接收端将阻塞来自发送端的RMA读取请求的流水线，导致发送端的RMA缓冲区被填充。这样一来，发送端的I / O线程将会停止工作，因为它们没有要读取的缓冲区。为了减轻这一点，引入了新的接口来将NVM用作RMA的扩展的存储区域，在RMA缓冲区已满的情况下继续读取对象。

当服务一个新的请求时，一个I/O线程试图保留一个RMA缓冲区。如果其中，没有缓冲区，则会尝试在SSD缓冲区中保留一个。如果成功，则会读取SSD缓冲区，将请求排入SSD队列，并唤醒SSD线程。SSD线程然后尝试获取RMA缓冲区。 如果尝试不成功，它会休眠等待RMA缓冲区被释放。当一个缓冲区被释放时，它会唤醒，获得RMA缓冲区，将数据复制到RMA缓冲区并将请求排到通信线程的工作队列上。最后，通信线程编组NEW BLOCK并将其发送到接收端。

我们可以将相同的发送端SSD缓存算法应用于接收端SSD缓存，然而，当Buffered IO操作被允许时，接收端缓存对提高数据传输速率的作用不大。因为算法无法考虑到操作系统缓冲区缓存的作用而无法正确检测拥塞的服务器。使用Direct IO进行写操作是可能的，并且允许我们的算法检测拥塞的服务器，但是直接的Direct IO性能更差，我们选择不使用它来进行接收端SSD缓存

### 在拥塞环境下使用Flash进行基于发送端的缓存技术

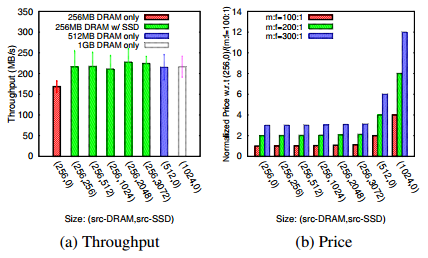
当接收端过载时，LADS的数据传输吞吐量显着下降。 这种情况下，发送端的RMA缓冲区变满，这会阻止I/O线程读取其他对象。因此，论文使用Flash进行基于发送端的缓存技术。 这种基于发送端的SSD缓冲利用了Flash上的可用缓冲区，比DRAM慢，但比HDD快，用于加载要传输的数据块。

为了评估它，作者通过存储拥塞周期之间插入10秒的空闲时间，修改了图10(b)所用的超载工作负载。在接收端的无拥塞期间，发送端可以将缓冲的数据从SSD缓冲区复制到RMA缓冲区。为了公平评估，接收端设置为只使用256 MB RMA缓冲区，发送端和接收端的8个I/O线程都不使用拥塞感知算法。

图11(a)显示了使用这种技术的结果。我们观察到吞吐量随着通信可用内存的增加而增加。但是，参考图11(b)，将DRAM的尺寸加倍是非常昂贵的，并且使用更便宜的Flash可以实现相同的吞吐量。

图11 bbcp的运行时间和吞吐量

拥挤的环境



# 感想及问题

其实刚开始阅读的时候觉得还挺困难的，因为这篇论文运用了Lustre文件系统，存在一些名词和技术流程上的疑点，并且还涉及到文件传输工具的原理，所以导致一些具体细节不能彻底读懂，但是读到核心思想就非常明了，就是从硬件层面去理解避免拥塞。

虽然核心思想了解了，但还是有许多不明白的问题：

（1）LADS允许无序的对象传输，那么前期必须得确定好每个文件的对象存储的位置，这其中耗费的时间占据的比重大不大？

（2）LADS中接收端不能采用的是Directed IO, 作者解释是这样会让性能变差，这是为什么？