计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2017.20150923
54(4):844-854,2017

面向科学计算可视化的两级并行数据读取加速方法

石 刘[1,2,3] 肖 丽[2,3] 曹立强[2,3] 莫则尧[2,3]
[1](中国舰船研究院 北京 100101)
[2](北京应用物理与计算数学研究所 北京 100088)
[3](中物院高性能数值模拟软件中心 北京 100088)
(shilbill@vip.sina.com)

Two Level Parallel Data Read Acceleration Method for Visualization in Scientific Computing

Shi Liu[1,2,3], Xiao Li[2,3], Cao Liqiang[2,3], and Mo Zeyao[2,3]
[1](*China Ship Research and Development Academy*, *Beijing* 100101)
[2](*Institute of Applied Physics and Computational Mathematics*, *Beijing* 100088)
[3](*CAEP Software Center for High Performance Numerical Simulation*, *Beijing* 100088)

Abstract In order to match the overall computing capability of super computer, the super computer's storage subsystem usually has good I/O performance scalability, which causes that, applications' I/O access concurrency under the best performance of the storage subsystem and the total compute core number (tens of thousands to several millions) of super computer are usually in the same order of magnitude; however, the process number (equals to the I/O access concurrency) used in visualization in scientific computing (ViSC) applications is usually relatively small (experientially set to 1% of used computing process number, typically several to hundreds). Therefore, the best I/O performance of the storage subsystem cannot be achieved. In this paper we propose a two level parallel data read-based acceleration method for ViSC applications. Multi threads parallel data accessing is introduced into the visualization process; the I/O access concurrency of the super computer's storage subsystem has been enhanced and visualization applications' data read rate has been promoted through the two level parallel read, i.e. the parallelism among multi processes and the parallelism among multi threads inner process. The test results show that, under various visualization process scales, the peak data read rate using two parallel mode is higher than that using single parallel mode by 33.5%-269.5%, while the mean data read rate using two parallel mode is higher than that using single parallel mode by 26.6%-232.2%; according to the diverse scientific computing applications and various process scales, based on two level parallel data read method, the overall peak running speed of visualization applications can be accelerated by 19.5%-225.7%, and the mean speed can be accelerated by 15.8%-197.6%.

Key words two level parallel data read; visualization in scientific computing; data access pattern; storage subsystem; I/O performance characteristics

---

收稿日期:2015-10-26;**修回日期**:2016-05-27
基金项目:国家自然科学基金重点项目(61232012);国家重点基础研究专项经费(2011CB309702);国家"八六三"高技术研究发展计划基金项目(2012AA01A309)
This work was supported by the Key Program of the National Natural Science Foundation of China (61232012), the Special Funds for National Key Basic Research Program of China (2011CB309702), and the National High Technology Research and Development Program of China (863 Program) (2012AA01A309).

摘　要　为了匹配超级计算机的整体计算能力,超级计算机存储子系统通常具有良好的I/O性能可扩展性,表现为:应用获得存储子系统最佳性能时的I/O访问并发度,与超级计算机系统总计算核数(可达数万至数百万)通常处于同一数量级.然而,科学计算可视化应用通常使用的进程数(等于I/O访问并发度)相对较小(经验上常设为计算进程数的1%,典型值为数个至数百个),因此无法充分发挥超级计算机存储子系统的最佳I/O性能.提出了一种面向科学计算可视化的两级并行数据读取加速方法,在可视化进程内部引入多线程并行数据读取,通过进程间和进程内两级并行,增加超级计算机存储子系统的I/O访问并发度,提升可视化应用数据读取速率.测试结果表明:在不同的可视化进程规模下,两级并行比单级并行峰值数据读取速率提高33.5%～269.5%,均值数据读取速率提高26.6%～232.2%;随着科学计算应用种类以及应用规模的变化,两级并行数据读取可使可视化应用整体峰值运行速度加速19.5%～225.7%,均值运行速度加速15.8%～197.6%.

关键词　两级并行数据读取;科学计算可视化;数据访问模式;存储子系统;I/O性能特征

中图法分类号　TP391

科学计算是以实际应用为牵引、以高性能计算机为依托而快速发展的一门交叉学科,被广泛应用于核武器、能源、信息、制造、材料和气候等诸多领域的科研和生产,对国防安全和国民经济发展起着非常积极的作用[1-2].科学计算可视化(visualization in scientific computing, ViSC)简称可视化,是运用计算机图形学和图像处理技术,将科学计算过程中产生的数据及计算结果转换为图像,在屏幕上显示出来并进行交互处理的技术,已成为科学研究的重要手段之一[3].

I/O是影响科学计算可视化应用性能的主要方面:在一些超过10万核的超大规模科学计算的可视化过程中,I/O时间超过可视化应用整体运行时间的90%①,而随着科学计算应用种类以及应用规模的变化,I/O时间占可视化应用整体运行时间的比例范围在65%～95%[4].

为了匹配超级计算机系统的整体计算能力,超级计算机的存储子系统通常被设计为具有良好的I/O性能可扩展性[5],表现为:应用获得存储子系统最佳性能时的I/O访问并发度,与超级计算机系统总计算核数通常处于同一数量级.

但是,相对于超级计算机系统超大的总计算核数(可达数万至数百万),科学计算可视化应用的进程规模(等于I/O访问并发度)则相对偏小(经验上通常设为科学计算应用进程规模的1%,典型值为数个至数百个),甚至远小于超级计算机系统的总计算核数(比如广州超算中心的天河2号超级计算机系统具有约300万个计算核心,而可视化应用典型进程规模与之相比,一般小几个数量级).因此,科学计算可视化应用往往无法充分利用超级计算机的I/O性能,影响可视化应用整体的运行效率.

本文提出了一种面向科学计算可视化的两级并行数据读取加速方法,其创新和贡献如下:1)根据可视化应用典型I/O进程规模,远小于超级计算机存储子系统发挥最佳性能时的I/O访问并发度这一特性,提出了面向科学计算可视化的两级并行数据读取加速方法;2)以典型科学计算可视化应用TeraVAP[6-7]为基础,设计并实现了功能独立的两级并行数据读取加速模块,验证了可视化两级并行数据读取方法的有效性.

1　背　　景

1.1　超级计算机存储子系统及性能特征

超级计算机上的存储子系统通常使用并行文件系统进行文件管理,科学计算环境下常见的并行文件系统包括Lustre[8],PVFS[9],GPFS[10]等.

并行文件系统的组件通常包括客户端、元数据服务器(metadata server, MDS)和存储服务器.并行文件系统通常使用存储服务器集群的设计,以使存储系统的容量和性能能够容易地进行横向扩展(scale-out):也即通过增加存储服务器的数量,即可容易地进行存储系统容量和性能的扩展.作为一个例子,图1给出了Lustre文件系统的整体架构[11],其中OSS(object storage service)servers即代表存储服务器集群.

① 数据来源于HPC China 2014会议Klask S A的大会报告《Co-designing for the 3 vs for data intensive computing》.

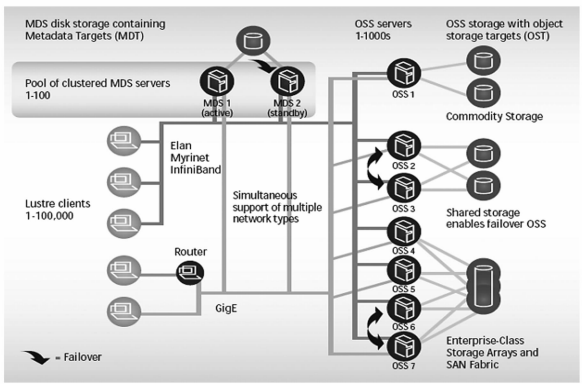

Fig. 1 Lustre file system architecture

图 1 Lustre 文件系统架构

存储服务器集群中通常包含较多的存储节点(存储服务器)以及大量的存储设备(磁盘),为了充分利用这些存储系统资源,应用通常需要较高的 I/O 访问并发度(总的访问线程数). 其外在表现为:超级计算机中的存储子系统通常具有良好的 I/O 性能可扩展性,并以此匹配超级计算机的超大总计算核数. 作为一个例子,IBM Blue Gene/P 是 2008 年计算性能世界排名前 5 的超级计算机系统,该系统具有 40 960 个 4 核计算节点,共计 163 840 个 CPU 核. 随着 I/O 进程(每进程 1 个线程,进程数等于线程数,也即等于 I/O 访问并发度)数量变化,IBM Blue Gene/P 的数据读取性能扩展性如图 2 所示(该测试直接基于底层 POSIX 接口,在全系统独占模式下进行).

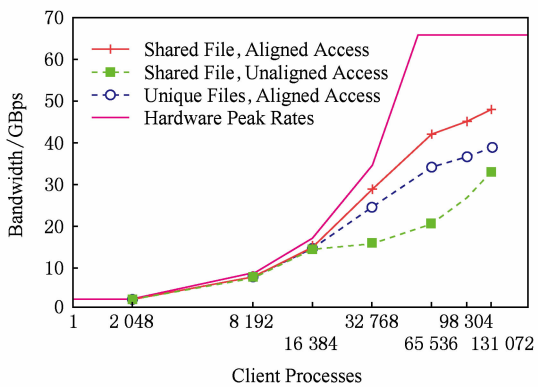

Fig. 2 Data read performance scalability of IBM Blue Gene/P system

图 2 IBM Blue Gene/P 系统数据读取性能扩展性

从图 2 可以看出,IBM Blue Gene/P 系统在 131072 进程时数据读取性能达到峰值[5]. 考虑到 IBM Blue Gene/P 系统总共具有 163 840 个 CPU 核,可见数据读取性能达到峰值时的 I/O 访问并发度(此例中总线程数等于总进程数)跟超级计算机系统的总计算核数接近,处于同一数量级.

1.2 科学计算可视化应用工作流程

科学计算可视化的主要工作包括并行数据读取、并行数据抽取和并行图形绘制 3 部分,其工作流程为:首先多个可视化进程并行读取科学计算结果数据集,然后多个进程进行并行数据抽取工作,最后多个进程并行进行图形绘制工作,如图 3 所示:

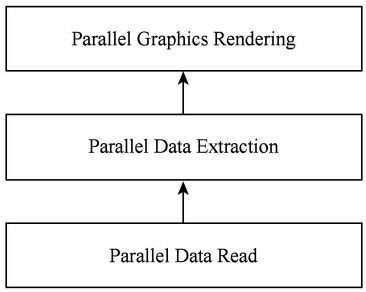

Fig. 3 Workflow of visualization in scientific computing

图 3 科学计算可视化工作流程

目前的可视化应用,图 3 中所示的 3 部分工作是紧耦合在一起的. 并行数据读取性能是影响科学计算可视化应用性能的最主要方面,其主要涉及到计算机数据存储方面的知识,而并行数据抽取和图形绘制则主要涉及计算机图形学方面的知识.

科学计算可视化应用通常采用一个适当的进程规模,这是因为:

1) 为了保证可视化应用具有一定的、用户可接受的性能,需要使用多个进程并行工作.

2) 可视化应用并行工作进程数量又不能太多,其原因有 3 点:

① 可视化应用是 I/O 密集型应用,增大进程规模则意味着增加计算核数,这导致在执行 I/O 的过程中,CPU 资源处于闲置状态,浪费功耗,能效降低;

② 对于打算重新购买超级计算机的用户而言,更大进程规模导致需要购买更多的计算节点资源,增大用户使用成本,而对于使用公共计算机资源的用户而言,更大进程规模导致排队等待时间变长、用户使用体验变差;

③ 程并行规模的增大,导致通信等开销的加大,在某些情况下反而可能导致一些可视化并行算法的效率降低.

目前,在这些因素(性能与功耗等)权衡下,经验上科学计算可视化应用的进程规模通常设为科学计算应用进程规模的 1%,典型值为数个至数百个.

1.3 科学计算可视化应用典型 I/O 访问模式

TeraVAP (terascale visualization & analysis platform)[6-7] 是一款典型的科学计算可视化应用,由北京应用物理与计算数学研究所研制,服务于基于

JASMIN(J parallel adaptive structured mesh application infrastructure)[12],JAUMIN(J parallel adaptive unstructured meshes applications infrastructure)[13],JCOGIN(J parallel mesh-free combinatory geometry infrastructure)[14]三个科学计算领域编程框架研制的科学计算应用.

目前 TeraVAP 已成功运用于数十个科学计算应用的后处理可视化数据分析,是一款典型的科学计算可视化应用,其数据访问模式为:多个进程并行读取科学计算结果总数据集的不同子数据集,如图 4 所示.这些子数据集之间交集为空,并集为总数据集.每个子数据集通常保存于一个数据文件中,总数据集的应用元数据信息保存于一个 summary 文件中.

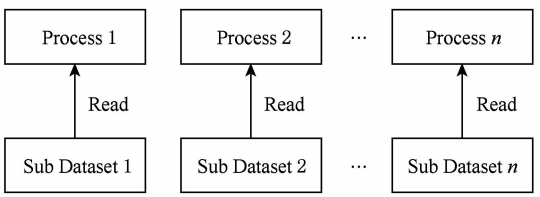

Fig. 4　Access pattern of visualization in scientific computing applications

图 4　科学计算可视化数据访问模式

TeraVAP 中每个进程读取的数据分为 2 级:第 1 级为网格片(patch)级,该级为逻辑概念,每个进程需要读取多个网格片;第 2 级为变量(variable)级,每个逻辑网格片上可以拥有多个变量,实际科学计算数据存储于变量中,如图 5 所示:

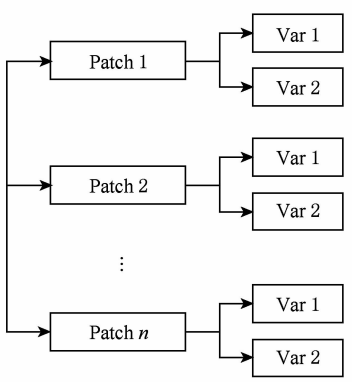

Fig. 5　Read data's organization diagram for each TeraVAP process

图 5　TeraVAP 每进程读取数据组织示意图

单个 TeraVAP 进程内部的数据读取的基本单位为一个变量,其访问流程为:以网格片(patch)为主循环、以变量(variable)为次循环串行读取数据,直到该进程被分配的所有网格片上的所有变量都读取完成,该进程数据读取过程结束,其流程如图 6 所示.这种数据访问模式意味着可视化应用对存储子系统的访问并发度(总 I/O 线程数)跟进程数相等.

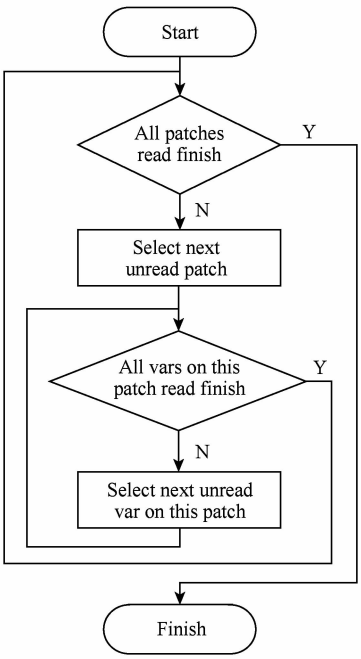

Fig. 6　Data read flow of one TeraVAP process

图 6　单个 TeraVAP 进程数据读取流程

可视化应用的进程规模通常较小(数个至数百个,远小于超级计算机系统总计算核数),并且由于进程数跟访问并发度(I/O 线程数)相等,因此,较低的访问并发度导致可视化应用无法获得较高的 I/O 性能(参见图 2).

2　问　　题

超级计算机存储子系统的 I/O 性能可描述如下:

$$P=F(N), \tag{1}$$

其中,P 代表可视化应用的 I/O 性能,N 代表可视化应用使用的 I/O 线程数,函数 F 代表随访问线程数变化的 I/O 性能.

F 的特点是:当 $N=N^*$ 时(N^* 通常跟超级计算机系统总计算核数处于同一数量级),P 达到最大值;当 $N<N^*$ 且逐渐增大时,F 是单调递增函数(参见图 2);当 $N>N^*$ 时,由于过分的竞争,性能会比峰值性能降低.

可视化应用通常使用较少的进程数 NP,也即可视化应用使用的 I/O 进程数 NP 通常远小于 N^*,即 $NP \ll N^*$.由于可视化应用采用单级进程间并行方式读取数据(每进程内线程数为 1),因此可视化应用的总 I/O 线程数 NT 等于总进程数 NP,即 $NT=NP$.

需要解决的问题是:如何使得在 NP 较小时得到较大的 P 值?

3 方法与实现

通过第 1 节和第 2 节介绍和分析可见:1)科学计算可视化应用所采用的进程数相对较小,通常为数个至数百个;2)设计良好的超级计算机存储子系统通常具有较好的 I/O 性能可扩展性,应用获得存储子系统最佳性能时的 I/O 访问并发度与超级计算机系统总计算核数通常处于同一数量级. 由于可视化应用的典型进程规模通常远小于超级计算机的总计算核数,因此科学计算可视化应用在超级计算机上无法充分利用 I/O 资源,从而无法获取较高的 I/O 性能. 本节介绍可视化应用的两级并行数据读取方法与实现.

3.1 两级并行数据读取方法

两级并行数据读取方法的思想是:

1) 在保持可视化进程规模不变的前提下,在进程间进行数据读取任务分配,使得进程间并行读取数据;

2) 在可视化进程内部引入多线程并行读取数据,使得进程内并行读取数据.

两级并行数据读取方法的流程如图 7 所示,通过进程间和进程内的两级并行数据读取方法,增加存储系统总的 I/O 访问并发度,提高可视化应用整体的读取性能.

图 7 中,如果 $NT=1$,则该进程内无需引入多线程并行,采用串行读取即可(如图 6 所示);如果 $NT>1$,则该进程内引入 NT 个线程并行读取数据.

一方面,从计算角度看,由于数据读取过程并不是计算密集型的,因此进程内的多个线程可以运行于同一个计算核上,无需增加计算核数,即保持第 2 节问题中的进程数 NP 较小;另一方面,从存储角度看,进程内的多线程并行读取的引入,导致应用对存储子系统的访问并发度增加,使得可视化应用可以获取更大的读取性能,即第 2 节问题中的 P 值变大.

在两级并行数据读取方法中,一个需要解决的关键问题是:如何确定每个进程的线程数以使得可视化应用读取性能最大化? 需要考虑的因素包括 2 点:1)可视化应用独占超级计算机系统资源的情况. 不同的超级计算机系统,其 I/O 达到峰值的访问并发度不尽相同,如果可视化应用单级纯进程并行时的访问并发度,已经超过了特定超级计算机存储子系统获得最佳性能时的 I/O 访问并发度,则进程内引入多线程的两级并行读取技术,不但不会提升读取性能,反而会导致读取性能下降(过分竞争所致). 2)可视化应用与其他应用共享超级计算机系统资源的情况. 存储子系统是个多应用共享系统,因此其他应用(可能)已经对存储子系统产生了一定的 I/O 访问并发度,应该使得可视化应用采用两级并行数据读取之后,所有应用(包括可视化应用)对存储子系统产生的总访问并发度,不超过存储子系统达到峰值性能时的 I/O 访问并发度(避免过分竞争导致性能下降).

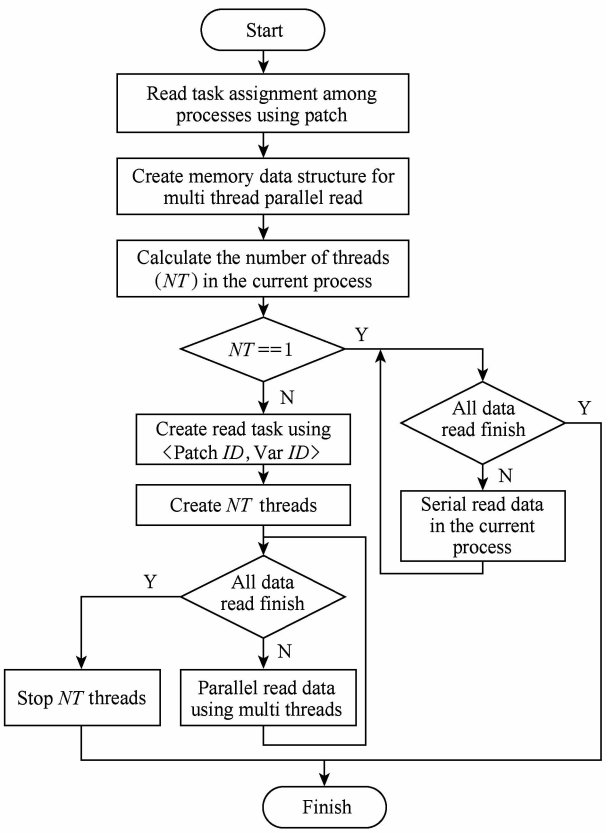

Fig. 7 Two level parallel data read flow of ViSC applications

图 7 科学计算可视化两级并行数据读取流程

假定存储子系统峰值 I/O 对应的访问并发度为 N^*,(除可视化应用外)所有其他应用产生的实时访问并发度为 NC,可视化进程规模为 NP,每进程的线程数为 NT,则计算特定进程内 NT 的算法流程如图 8 所示.

在图 8 中可以看出:1)判断当前计算机系统存储子系统 I/O 访问并发度是否超过存储子系统获取最大 I/O 性能时的访问并发度. 如果超过,可视化进程内线程数设为 1,即采用单级纯进程并行方式读

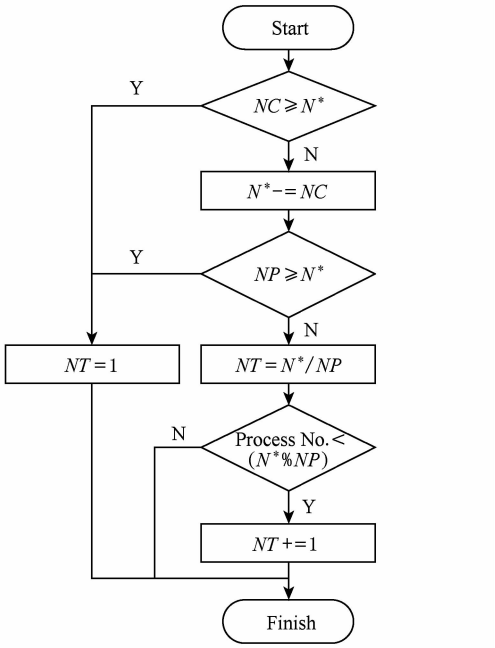

Fig. 8　Thread number calculation algorithm flow in one ViSC process

图 8　计算可视化进程内线程数算法流程

取数据;如果不超过,通过 N^*-NC 去除共享超级计算机系统资源情况下其他应用产生的实时 I/O 访问并发度. 2)可视化应用可看作独占超级计算机系统资源的情况,该超级计算机系统获取 I/O 峰值性能时的访问并发度即为 1)中的 N^*-NC,并以此计算每个进程内的线程数.

N^* 可通过测试形成映射表由用户配置或者根据存储子系统的软硬件配置,按照一定的随访问并发度变化的存储系统性能模型计算得到.

NC 可通过测试形成映射表由用户配置,或者根据存储子系统的实时负载状况,按照一定的随访问并发度变化的存储系统负载模型计算得到.

NC 是否大于 N^* 的判断可由存储子系统的实时负载状况确定.

3.2　可视化读取模块实现

我们以 TeraVAP 可视化平台为基础,独立出并行数据读取部分,实现了功能独立的可视化两级并行数据读取加速模块 TeraVAPReader.

为此,首先凝炼了 TeraVAPReader 与 TeraVAP 之间的数据读取接口,以方便实现可视化进程内的线程并行数据读取,如下:

1) 读取多网格片数据接口原型. *st_patches_raw_data* * ReadPatchData(IN char * *full_path_file_name*, IN int * *patch_id_list*, IN int *patch_count*, IN const char * *var_name_list*[], IN int *var_count*).

在以上接口原型中,*full_path_file_name* 表示数据文件所对应应用元数据文件 summary 的文件名,*patch_id_list* 表示多个网格片,*patch_count* 表示需要读取的网格片数量,*var_name_list* 表示网格片上需要读取的变量列表,*var_count* 表示需要读取的变量个数.

2) 设计了新的数据结构,用于在单个进程内供多线程并行读取并保存数据. 接口返回值类型 *st_patches_raw_data* 即为该数据结构,其由多个结构体组合定义而成,用于保存单个可视化进程内部的多个网格片上的多个变量数据,内存结构示意如图 9 所示. 需要指出的是,由于每个变量都有各自独立的内存存储空间,相互之间不会干扰,因此该结构

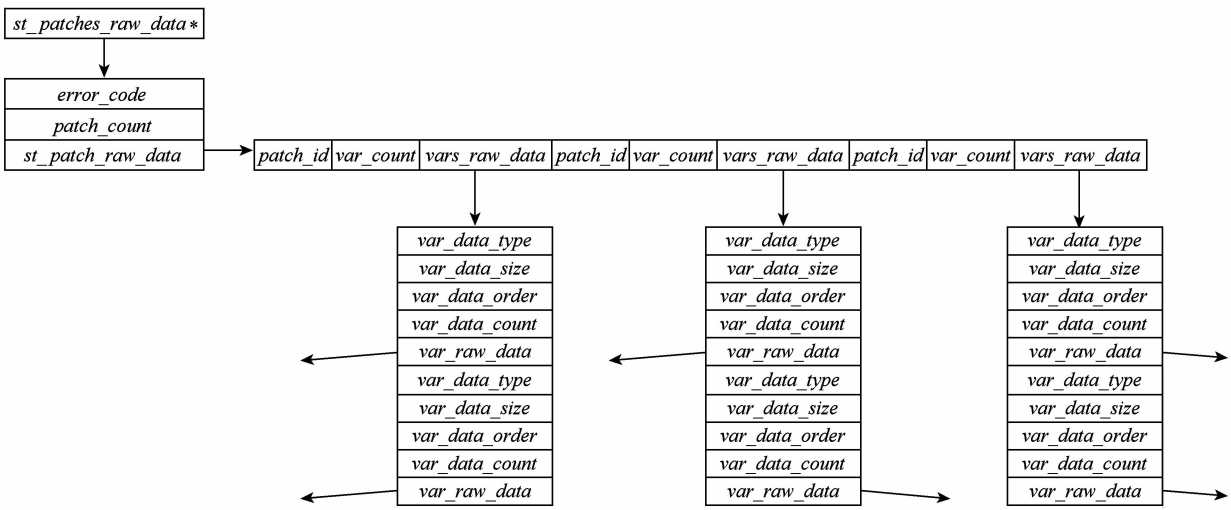

Fig. 9　Memory data structure for multi threads parallel read data in one ViSC process

图 9　可视化进程内多线程并行读取数据的内存数据结构

采用无锁设计,避免了多个线程并行访问时的同步开销.其中,*error_code* 域用于保存多线程并行读取数据的错误码,*patch_count* 域表示该进程需要读取的网格片数量,*patch_id* 域表示网格片全局编号,*var_count* 域表示网格片上的变量数,*var_raw_data* 域用于存放 ViSC 应用实际读取的单个变量的原始数据,其他以"var_"开头域为读取变量数据的辅助信息(应用元数据,例如变量字节数、变量数据类型等).

3) 针对典型可视化应用 TeraVAP 的数据组织特点,基于单生产者多消费者模型,设计了针对性的两级并行数据读取算法,步骤如下:

步骤 1. 以网格片为单位,进行多个可视化进程间的读取任务分配,达到进程间并行数据读取. 如图 10 所示:

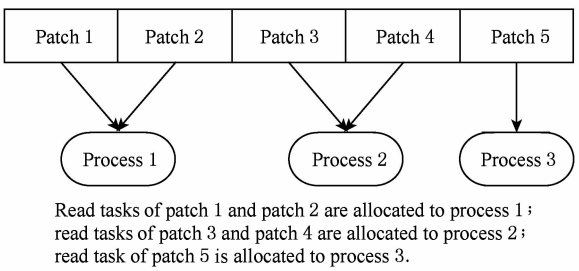

Fig. 10 Patch-based read task allocating diagram among ViSC processes

图 10 可视化进程间以网格片为单位的读取任务分配

步骤 2. 每个可视化进程内,以单个网格片上的单个变量读取作为一个读取任务,一次性生产出该可视化进程的所有读取任务.

步骤 3. 每个可视化进程内启动多个读取线程,采用多消费者模型逐个获取读取任务并进行实际数据读取,达到进程内并行数据读取. 如图 11 所示.

算法步骤 1 和步骤 3 联合,实现了两级并行数据读取. 当所有进程内的读取任务处理完成,整个可视化应用的并行数据读取过程结束,进入并行数据抽取阶段.

假定单个可视化进程内所有网格片上的所有变量总数为 n,则该算法的时间复杂度和空间复杂度都为 $O(n)$.

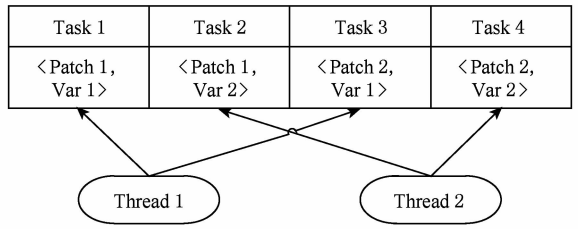

Fig. 11 Multi consumers model-based multi threads parallel data read diagram in one ViSC process

图 11 可视化进程内基于多消费者模型的多线程并行数据读取

4) 实现了功能独立的可视化两级并行数据读取加速模块. 进程间任务分配以网格片为单位,进程内创建所有读取任务以变量为单位(以〈Patch *ID*, Var *ID*〉为键值,如图 7 所示). 采用用户配置的方式设置 N^* 和 NC(如图 8 所示),并由此计算出当前进程所需线程数 NT. 进程间以网格片为单位的多 MPI 进程并行数据读取,与进程内以变量为单位的多 pThread 线程并行数据读取一起,共同实现了两级并行数据读取.

4 实验结果与分析

为了验证两级并行数据读取加速方法,我们使用科学计算应用 LinAdvSL(单层均匀矩形结构网格上求解线性对流方程)在一台曙光计算机以及广州超算中心的天河 2 超级计算机上分别对该方法进行了测试. 测试方法为:1)通过 LinAdvSL 3D 应用产生测试数据集;2)使用 TeraVAPReader 分别进行单级并行和两级并行数据读取性能测试(I/O 性能只跟数据总量和变量总数相关,采用 LinAdvSL 单个应用产生测试数据集即可). 表 1 显示了测试的软硬件环境.

Table 1 Software and Hardware for Test

表 1 测试软硬件环境

Test Machine	Computing Node Hardware Configuration	Storage Subsystem Hardware Configuration	Test Software Environment
Sugon	4 cores, 16 GB memory, 20 Gbps ib	Lustre, 26 I/O nodes, 78 TB data volumn for users	Linux, JASMIN, HDF5, MPI, pThread, TeraVAPReader
Tianhe2-b	24 cores, 64 GB memory, 40 Gbps ib	Lustre, 3.4 PB data volumn for users	Linux, JASMIN, HDF5, MPI, pThread, TeraVAPReader

表 1 中需要说明的是 HDF5[15] 库采用的是 1.8.6 版本,在编译时通过--enable-threadsafe 选项打开 HDF5 对 pThread 库的多线程并行访问支持,未使用 MPI-IO[16].

表 2 显示了测试所采用的数据集:

Table 2　Dataset for Test

表 2　测试采用的数据集

Test Machine	Application for Data Generation	Data Volumn of One Step/MB	Total Patch Number	Variable Number on Patch
Sugon	LinAdvSL 3D	7 837	1 274	1
Tianhe2-b	LinAdvSL 3D	106 126	14 450	1

我们通过在每个进程内变动读取线程数的方法来进行数据读取性能测试.每种测试进行 5 次,结果取平均值.进程内线程数等于 1 代表单级并行数据读取,进程内线程数大于 1 代表两级并行数据读取.测试过程中,使用的计算核数等于进程数;当进程内的线程数变化时,使用的计算核数不变.

曙光集群的存储子系统规模相对较小,因此在曙光集群上进行了小规模进程数(进程数为 1～8,每进程的线程数为 1,4,8)的性能测试.图 12 显示了使用不同进程线程数在曙光集群上的性能测试结果,可以看出:当进程数小于 4 时,单级并行数据读取速率反而比两级并行数据读取速率高,这是因为两级并行引入多线程的开销所致;随着进程规模逐渐增大,两级并行的性能优势逐渐增加,掩盖了其多线程开销,最终导致两级并行的数据读取速率超过单级并行数据读取速率,并在 8 进程时两级并行与单级并行数据读取速率差值达到最大.

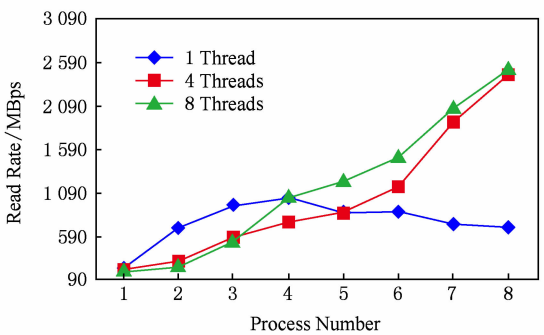

Fig. 12　Data read rate of Sugon cluster using different process number and thread number

图 12　曙光集群使用不同进程线程数的数据读取速率

图 12 中单级并行数据读取时(1 线程情况,进程内采用图 6 中串行读取方式),当进程数超过 4 以后,随着进程数的增加,数据读取性能反而有所下降,这与图 2 中的 I/O 性能曲线规律并不一致.我们的测试中使用了 HDF5 库,而图 2 的测试中并未使用该库,我们分析认为,这种不一致应该跟 HDF5 库中进程和线程 2 种模式下的数据读取算法相关.

图 13 显示了曙光集群 8 进程下使用不同线程数的数据读取速率,可以看出,在 8 进程下,两级并行比单级并行峰值数据读取速率提高 269.5%(7 线程 vs 1 线程),均值数据读取速率(2～8 线程数据读取速率的平均值 vs 1 线程)提高 232.2%.从图 13 中也可以看出,单个计算节点的 I/O 带宽具有上限,也就是进程内的线程数并不是越多越好.

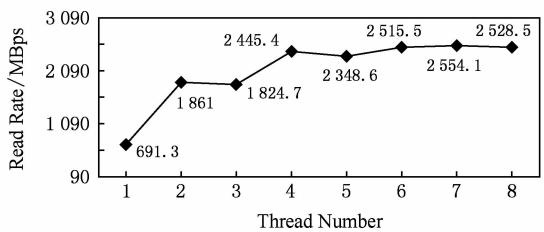

Fig. 13　Data read rate of Sugon cluster under 8 processes

图 13　曙光集群 8 进程数据读取速率

在天河 2-b 集群上进行了大规模进程数(进程数为 128～1024,每进程的线程数为 1～64)的性能测试,图 14 显示了使用不同进程线程数的性能测试结果.由于天河 2-b 集群是目前计算性能世界排名第一的超级计算机系统,其上同时运行着较多的应用软件,这些应用软件之间共享存储子系统资源,彼此之间会造成 I/O 性能干扰.因此,图 14 的 I/O 性能曲线特征没有图 2 那样明显,但从图中仍然可以看出,两级并行数据读取速率超过单级并行.

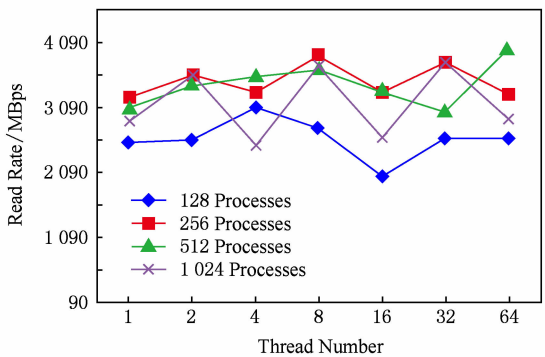

Fig. 14　Data read rate of Tianhe2-b cluster using different process number and thread number

图 14　天河 2-b 集群使用不同进程线程数的数据读取速率

图 15 显示了不同进程规模下天河 2-b 集群两级并行比单级并行数据读取速率峰值提升比例:不同进程下两级并行比单级并行峰值数据读取速率

提高 33.5%(1024 进程下),均值数据读取速率(128～1024 进程下数据读取速率提高比例的平均值)提高 26.6%.

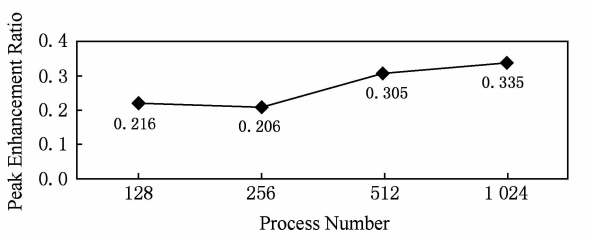

Fig. 15 Two level parallel-based data read peak enhancement of Tianhe2-b cluster compared with one level parallel under different process scales

图 15 天河 2-b 集群不同进程规模下两级并行比单级并行数据读取速率峰值提升比例

对比图 15 和图 13 可以发现,相比于小规模进程下大幅度的性能提升,大规模进程下两级并行比单级并行数据读取速率提升比例变小.一方面原因是,大规模进程下单级并行数据读取速率已经提高,两级并行数据读取性能能够提升的空间缩小了(参见图 2 随访问并发度变化的 I/O 性能曲线);另一方面可能原因是,测试时系统中由其他应用产生的访问并发度已经较高,剩余的性能提升空间已不大.

综合图 12～15 可以看出,在不同的进程规模下,两级并行比单级并行峰值数据读取速率提高 33.5%～269.5%,均值数据读取速率提高26.6%～232.2%,可视化应用 I/O 数据读取速率得到显著提升,验证了两级并行数据读取加速方法的有效性.

按照 I/O 时间占可视化应用整体运行时间的比例范围在 65%～95%这一数据[4],结合上述数据读取速率提高的比例范围,通过计算可得,跟单级并行数据读取相比,两级并行数据读取加速可视化应用整体运行速度的效果,如表 3 所示:

Table 3 Overall Running Speed Enhancement of ViSC Applications

表 3 可视化应用整体运行速度提升比例 %

I/O Proportion in ViSC Running Time	Minimum Value of Peak Read Rate Enhancement Proportion	Maximum Value of Peak Read Rate Enhancement Proportion	Minimum Value of Average Read Rate Enhancement Proportion	Maximum Value of Average Read Rate Enhancement Proportion
65	19.5	90.2	15.8	83.3
75	23.2	120.8	18.7	110.2
85	27.1	163.2	21.7	146.4
95	31.3	225.7	25.0	197.6

表 3 中分别计算出了可视化应用整体峰值和均值运行速度提升比例的最小值和最大值,可以看出,随着科学计算应用种类以及应用规模的变化,两级并行数据读取可使可视化应用整体峰值运行速度加速 19.5%～225.7%,均值运行速度加速 15.8%～197.6%.

5 相关研究

ADIOS(adaptable I/O system)[17]项目由美国橡树岭国家实验室计算科学国家中心牵头,联合劳伦斯伯克利国家实验室科学数据管理中心以及美国国防部等单位针对科学计算应用提出应用级 I/O 框架.该项目采用可扩展的架构,集成了多种 I/O 访问策略并且定制了针对科学计算应用的专用数据格式,以尽量获取最大 I/O 带宽,从而减轻科学计算及可视化应用运行过程中所遭遇的 I/O 性能瓶颈问题.该项目采用基于希尔伯特空间填充曲线的方式进行数据分块,以提高可视化数据读取效率,该方法主要针对直交平面这种数据读取模式提出.

DCPL(dual channel parallel I/O library)[18]通过预取可视化应用元数据、减少应用元数据读取次数的方法,提升可视化应用的读取性能.由于下层的 DCPL 库并不了解上层可视化应用的语意,元数据预取可能失败.

两阶段 I/O(two phase I/O)与数据过滤(data sieving)[19]等方法可提高小块数据的 I/O 性能.这些技术通常实现在如 MPI-IO 等底层数据库/数据中间件中,不在可视化应用 I/O 必经的路径中[20].

I/O 转发可扩展层技术(I/O forwarding scalability layer)[21]用于高层 I/O 库和文件系统之间,使用专用 I/O 节点聚合、转发高层 I/O 调用,提高读取效率

并增强可扩展性.该转发技术层是针对特定体系结构研发,目前基于 Lustre 文件系统的许多超级计算机都未配备该转发层.

6　未来工作

本文分析了超级计算机存储子系统的 I/O 性能特征,介绍了科学计算可视化应用的典型 I/O 访问模式,提出了面向科学计算可视化的两级并行数据读取加速方法,并以典型科学计算可视化应用 TeraVAP 为基础,设计并实现了功能独立的两级并行数据读取加速模块.测试和分析表明,两级并行数据读取方法可显著提升可视化应用 I/O 读取速率,从而加速可视化应用整体运行速度.

目前在单个可视化进程内部,启动多少个线程进行数据读取加速是由参数 N^* 和 NC 决定的(如图 8 所示),而在目前的实现中,N^* 和 NC 是通过测试形成映射表由用户配置的方式确定的.采用用户配置的方式确定 N^* 和 NC,需要用户具有比较专业的知识,对超级计算机存储子系统的性能较为熟悉,这对普通用户使用可视化应用造成了一定的困难.考虑到可视化应用具有跨不同超级计算机平台运行的需求,未来需要研究基于存储体系结构感知和运行时存储子系统负载感知的 I/O 性能自动优化方法:通过自动识别超级计算机存储子系统的软硬件配置以自动确定参数 N^*,通过自动感知存储子系统的实时负载以自动确定参数 NC.通过 N^* 和 NC 的自动确定,自动设置进程内最优的多线程参数 NT,达到可视化应用 I/O 性能自动优化的目标.

参　考　文　献

[1] Zhu Shaoping. A brief report on scientific computing [J]. Physics, 2009, 38(8): 545-551 (in Chinese)
(朱少平. 浅谈科学计算[J]. 物理, 2009, 38(8): 545-551)

[2] Zhang Linbo, Chi Xuebin, Mo Zeyao, et al. Introduction to Parallel Computing [M]. Beijing: Tsinghua University Press, 2006 (in Chinese)
(张林波, 迟学斌, 莫则尧, 等. 并行计算导论[M]. 北京: 清华大学出版社, 2006)

[3] Tang Zesheng, Sun Yankui, Deng Junhui. Advances in the study of visualization in scientific computing [J]. Journal of Tsinghua University: Science and Technology, 2001, 41(4/5): 199-202 (in Chinese)
(唐泽圣, 孙延奎, 邓俊辉. 科学计算可视化理论与应用研究进展[J]. 清华大学学报: 自然科学版, 2001, 41(4/5): 199-202)

[4] Bethel E W, Johnson C, Ahern S, et al. Occam's razor and petascale visual data analysis [J]. Journal of Physics: Conference Series, 2009, 180(1): 1-18

[5] Lang S, Carns P, Latham R, et al. I/O performance challenges at leadershiop scale [C] //Proc of Int Conf for High Performance Computing, Networking, Storage & Analysis (SC'09). Piscataway, NJ: IEEE, 2009: 1-12

[6] Xiao Li, Cao Xiaolin, Wang Huawei, et al. Large-scale data visual analysis for numerical simulation of laser fusion [J]. Journal of Computer-Aided Design and Computer Graphics, 2014, 26(5): 675-686 (in Chinese)
(肖丽, 曹小林, 王华维, 等. 激光聚变数值模拟中的大规模数据可视分析[J]. 计算机辅助设计与图形学学报, 2014, 26(5): 675-686)

[7] Xiao Li, Ai Zhiwei, Cao Xiaolin. A patch-based data reorganization method for coupling large-scale simulations and parallel visualization [J]. Trans on Edutainment IX, 2013, 7544(9): 278-289

[8] Braam P J, Schwan P. Lustre: The intergalactic file system [C/OL] //Proc of Ottawa Linux Symp. [2015-10-10]. https://www.kernel.org/doc/ols/2002/ols2002-pages-50-54.pdf

[9] Carns P H, Ligon Ⅲ W B, Ross R B, et al. PVFS: A parallel file system for Linux clusters [C] //Proc of the 4th Annual Linux Showcase and Conf. Berkeley, CA: USENIX Association, 2000: 317-327

[10] Schmuck F, Haskin R. GPFS: A shared-disk file system for large computing clusters [C] //Proc of the Conf on File and Storage Technologies (FAST'02). Berkeley, CA: USENIX Association, 2002: 231-244

[11] Sun Microsystems Inc. Lustre file system: High-performance storage architecture and scalable cluster file system white paper [EB/OL]. [2015-10-16]. http://www.csee.ogi.edu/~zak/cs506-pslc/lustrefilesystem.pdf

[12] Mo Zeyao, Zhang Aiqing, Cao Xiaolin, et al. JASMIN: A parallel software infrastructure for scientific computing [J]. Frontiers of Computer Science in China, 2010, 4(4): 480-488

[13] Zhao Weibo, Liu Qingkai, Qin Guiming. Parallel computation for finite element method based on hierarchical data structure [J]. Computer Engineering & Science, 2012, 34(8): 107-111

[14] Zhang Baoyin, Li Gang, Deng Li. JCOGIN: A combinatorial geometry monte carlo particle transport infrastructure [J]. High Power Laser and Particle Beams, 2013, 25(1): 173-176

[15] HDF Group. Hierarchical data format 5 [EB/OL]. [2015-10-15]. http://www.hdfgroup.org/HDF5

[16] Message Passing Interface Forum. MPI-2: Extensions to the message-passing interface [EB/OL]. [2015-10-18]. http://www.mpi-forum.org/docs/docs.html

[17] Lofstead J F, Klasky S, Schwan K, et al. Flexible IO and integration for scientific codes through the adaptable IO system (ADIOS)[C] //Proc of CLADE 2008 at HPDC. New York: ACM, 2008: 792-798

[18] Cao Liqiang, Mo Zeyao, Shen Weichao, et al. Dual channel parallel I/O: A fast data access for scientific computing [J]. Chinese Journal of Computers, 2015, 38(5): 1035-1043 (in Chinese)
(曹立强, 莫则尧, 沈卫超, 等. 科学计算双路并行 I/O 优化方法[J]. 计算机学报, 2015, 38(5): 1035-1043)

[19] Thakur R, Gropp W, Lusk E. Data sieving and collective I/O in ROMIO [C] //Proc of the 7th Symp on the Frontiers of Massively Parallel Computation. Piscataway, NJ: IEEE, 1999: 182-189

[20] Behzad B, Luu H V T, Huchette J, et al. Taming parallel I/O complexity with auto-tuning [C] //Proc of Int Conf for High Performance Computing, Networking, Storage & Analysis (SC'13). Piscataway, NJ: IEEE, 2013: 1-12

[21] Vishwanath V, Hereld M, Iskra K, et al. Accelerating I/O forwarding in IBM blue Gene/P systems [C] //Proc of Int Conf for High Performance Computing, Networking, Storage & Analysis(SC'10). Piscataway, NJ: IEEE, 2010: 1-10

Shi Liu, born in 1982. PhD and assistant researcher. His main research interests include storage system, file system, cooperative caching system and application level I/O technologies.

Xiao Li, born in 1971. Master, researcher. Member of CCF. Her main research interests include visualization for engineering and science computing, visual analysis and parallel visualization.

Cao Liqiang, born in 1976. PhD and associate researcher. His main research interests include high volume data management in scientific computing and efficiently parallel I/O in HPC.

Mo Zeyao, born in 1971. Professor and PhD supervisor. His main research interests include algorithm, system and application in high performance computing.