面向神威·太湖之光的 PETSc 可扩展异构并行算法及其性能优化

洪文杰[1)] 李肯立[1)] 全 哲[1)] 阳王东[1)] 李克勤[1)] 郝子宇[2)] 谢向辉[2)]

[1)](湖南大学信息科学与工程学院国家超级计算长沙中心 长沙 410082)

[2)](江南计算技术研究所 江苏 无锡 214125)

摘 要 共性数学库 PETSc(Portable, Extensible Toolkit for Scientific Computation)是高性能计算的基础模块,是超级计算机计算环境的基础算法库之一,其性能直接影响调用数学库的高性能数值计算应用的效率.面向国际上首台 100P 神威·太湖之光异构超级计算机,根据实际研究需要选取 PETSc 中两个典型用例 ex5(单节点线性求解方程组问题)和 ex19(多节点求解 2D 驱动腔问题)进行实验探究.对运行结果分析找到的热点函数主要为 PETSc 函数库中 7 个核心函数,针对这 7 个核心函数(主要包括向量运算与矩阵运算),提出和实现了其异构并行算法,并结合机器的异构体系结构提出了相应的性能优化方法.在超级计算机上的实验结果为:核心函数并行算法在 4 主核、256 从核的单节点上加速比最大可达到 16.4;多节点情况下,当输入规模为 16 384 时,8192 个节点相对于 256 节点的加速比为 32,且加速比随着异构处理器数目的增加接近线性增加,表明 PETSc 核心函数并行算法在神威·太湖之光超级计算机上具有良好的可扩展性.

关键词 并行算法设计;PETSc 数学库;可扩展性;神威·太湖之光

中图法分类号 TP301 **DOI 号** 10.11897/SP.J.1016.2017.02057

PETSc's Heterogeneous Parallel Algorithm Design and Performance Optimization on the Sunway TaihuLight System

HONG Wen-Jie[1)] LI Ken-Li[1)] QUAN Zhe[1)] YANG Wang-Dong[1)]
LI Ke-Qin[1)] HAO Zi-Yu[2)] XIE Xiang-Hui[2)]

[1)](*National Supercomputing Center of Changsha, College of Computer Science and Engineering, Hunan University, Changsha* 410082)

[2)](*Jiangnan Institute of Computing Technology, Wuxi, Jiangsu* 214125)

Abstract Large-scale scientific and engineering calculations such as hydrodynamic calculations, numerical weather forecasting, seismic data processing, genetic engineering, and high-dimensional differential equations are facing with the big performance challenge. Meanwhile, the High Performance Computing (HPC) platform has been significantly developed in recent years. The appearances of multi-core processors and heterogeneous computing platforms dramatically improve the performance of high-performance applications. To fully utilize the computing power of HPC systems, it is necessary to develop specific methodologies to optimize the performance of applications based on the system architecture. The Sunway TaihuLight supercomputer is presently

收稿日期:2016-10-21;在线出版日期:2017-06-12.本课题得到国家自然科学基金重点项目(61432005,91430214)、国家杰出青年科学基金(61625202)、国家自然科学基金(61602166)、数学工程与先进计算国家重点实验室开放基金课题、国家重点研发计划(2016YFB0201402,2016YFB0201900)资助.**洪文杰**,男,1993 年生,硕士研究生,中国计算机学会(CCF)学生会员,主要研究方向为高性能计算.E-mail:hwj_2015@hnu.edu.cn.**李肯立**(通信作者),男,1971 年生,博士,教授,博士生导师,中国计算机学会(CCF)杰出会员,IEEE 高级会员,研究领域为高性能计算、并行与分布式系统.E-mail:lkl@hnu.edu.cn.**全 哲**,男,1982 年生,博士,主要研究方向为人工智能、高性能计算.**阳王东**,男,1974 年生,博士,教授,主要研究领域为并行计算.**李克勤**,男,1963 年生,博士,教授,博士生导师,IEEE Fellow,研究领域为并行计算、分布式计算、云计算等.**郝子宇**,男,1978 年生,博士,高级工程师,主要研究方向为超级计算机体系结构、算法设计与优化.**谢向辉**,男,1958 年生,博士,高级工程师,中国计算机学会(CCF)高级会员,主要研究方向为计算机体系结构、高性能互联等.

ranked in the TOP500 list as the fastest supercomputer in the world, with a LINPACK benchmark rating of 93 petaflops. The Sunway TaihuLight uses a total of 40960 Chinese designed SW26010 multi-core 64-bit RISC processors. Portable, Extensible Toolkit for Scientific Computation (PETSc), an indispensable module of high performance computing, is one of basic algorithm libraries widely applied in many high-performance applications. Meanwhile, PETSc is also widely used in partial differential equations, sparse linear algebra and other related problems. The performance of PETSc directly affects the efficiency of applications invoking PETSc. In this paper, we use two most typical cases in PETSc according to actual research needs, that is ex5 (solving problems of linear systems on single node) and ex19 (solving problems of 2D driving cavity on multi nodes) to perform them on the Sunway TaihuLight supercomputer. With the analysis of experimental results, we figure out there are seven core functions including vector calculations and matrix calculations. First of all, for each core function, we do an in-depth research of its characteristics, parallel difficulties, optimizations for the bottlenecks. And then, we determine an appropriate heterogeneous parallel model for these functions on the SW26010 processor (there a total of four heterogeneous parallel model on the Sunway Taihulight). Finally, we figure out the best division strategy for task, determine the size of the data transferred, and design the parallel algorithm on the Sunway TaihuLight supercomputer. Furthermore, a series of novel performance optimization strategies is proposed according to the heterogeneous architecture of the Sunway TaihuLight system. These optimization methods mainly include the access optimization, eliminating data dependency and vectorization optimization. As the experimental results shown in this paper, our parallel algorithms of the seven core functions achieve the maximum speed up to 16.4 on one single node (contains 4 MPEs and 256 CPEs). In the case of run on multiple nodes, the acceleration ratio reaches 32 on 8192 nodes compared to 256 nodes, when the input data scale is up to 16384. Besides, the speedup presents an linear tendency with the increasing number of processors. This paper demonstrates that our parallel algorithms of PETSc have good scalability, reliability and security on the Sunway TaihuLight supercomputer, which provides the reference for the similar researches.

Keywords parallel algorithms; PETSc math library; extensibility; Sunway TaihuLight system

1 引 言

大规模的科学与工程计算诸如流体力学计算、数值天气预报、地震数据处理、基因工程、高维微分方程数值解仍是当前高性能计算亟待解决的挑战性问题. PETSc 库是一种可移植科学计算工具箱,是高性能计算环境的基础设施之一,广泛应用于偏微分方程、稀疏线性代数等相关问题的求解,是提高高性能计算应用程序计算效率的重要基础.

根据申威处理器特殊的异构体系结构和网络和存储特性,提出了 PETSc 库 7 个核心算法的并行算法,并提出了相应的性能优化方法.

神威·太湖之光超级计算机是由国家并行计算机工程技术研究中心研制的世界上首台运算速度超过十亿亿次的超级计算机. 如何在高性能计算应用中发挥超级计算机潜在的计算能力一直是高性能研究的主要挑战之一. 通常,多数应用都会通过调用开源的底层共性数学库来实现,而 PETSc 就是这一类数学库.

Portable, Extensible Toolkit for Scientific Computation PETSc[①] 是美国 Argonne 国家实验室开发的可移植可扩展科学计算工具箱,目的是在高性能计算机上数值求解偏微分方程及相关问题. PETSc 是基于并行库(Message Passing Interface, MPI)和数学计算包(BLAS)实现. 它用 MPI 来实现并行所需进程之间的通信,而数学计算是通过调用

① Portable. Extensible Toolkit for Scientific Computation (PETSc). http://www.mcs.anl.gov/petsc 2016,5,25

BLAS 来实现. 最底层是基础库，向上依次是基础数据结构：主要为矩阵和向量；Krylov 子空间方法：一类用来求解形如 $\boldsymbol{Ax}=\boldsymbol{b}$ 方程的方法，预设条件子：包括雅可比矩阵、分块雅可比矩阵、SOR/SSOR 方法、不完全 Cholesky 分解和不完全 LU 分解等方法的集合；SNES 为大规模非线性方程提供高效的非精确或拟牛顿迭代解法.

神威·太湖之光是目前世界上最快的超级计算机，由我国自主研发的申威众核处理器为核心构建而成. PETSc 是高性能计算应用和数值计算应用常用的共性数学库，如何充分发挥神威·太湖之光在并行高性能计算应用中的计算潜力是当下面临的问题和挑战，对于一类以 PETSc 数学库作为底层调用的并行应用，其性能下限取决于 PETSc 数学库在当前体系结构下的优化效果. 因此，在神威·太湖之光上实现 PETSc 数学库的并行优化有着重要意义，为国产异构超算上实现自身配套的数值计算函数库进行探索实验. 由于 PETSc 数学库十分庞大，本文根据研究需要，对 PETSc 数学库的 7 个核心函数实现并行异构算法设计和优化.

文章的后续部分组织如下：第 2 节给出相关工作介绍，第 3 节介绍神威·太湖之光系统环境和申威体系结构，这是 PETSc 数学库并行算法设计的基础，核心函数的并行算法由第 4 节给出，第 5 节和第 6 节分别给出性能优化方法及实验结果的对比分析，最后是结论和未来研究工作的方向.

2　相关工作

Minden、Smith 等人研究 PETSc 函数库中所有向量运算函数和矩阵向量积函数在 GPU 上的实现[1]，实验结果表明：PETSc 中稀疏矩阵迭代求解时，加速瓶颈受限于 CPU 与 GPU 之间的传输带宽，而不取决于加速卡(GPU)个数，因此加速比最多只有几倍，要达到 100 倍或者更多几乎不可能.

Cuomo、Galletti 和 Giunta 等人在 CPU-GPU 集群上基于 PETSc 中抛物线模拟光流问题的偏微分方程设计了一种适应于异构计算环境并行算法，并软件实现[2]. 采用真实的 SAR 图像序列进行软件数值实验得到加速比为 1.95.

本文工作是在神威·太湖之光异构系统上，根据国产申威处理器特殊的异构结构，针对 PETSc 数学库中 7 个核心函数，设计实现并行算法并优化. 与前两者工作的区别在于：(1) 基于不同的体系结构；(2) 相同数学库下的不同应用实例的优化. 而归结到底层而言所有工作都是解决同一类问题：针对不同的异构体系架构，PETSc 函数库中迭代算法下的向量矩阵运算是否能在该异构环境下取得好的加速效益，充分发挥该异构处理器的性能，进而通过优化后的 PETSc 函数库解决一系列复杂科学问题.

根据研究需要，选取 PETSc 数学库自带标准实例 ex5① 和 ex19② 进行实验，二者调用了 PETSc 中常用核心函数，其中 ex5 为 KSP 下单节点运行的标准实例，利用牛顿迭代法求解两个线性系统，ex19 是 SNES 下可多节点 MPI 扩展的标准实例，利用 Jacobi 预处理的广义极小残量法(GMRES)求解 2D 驱动腔问题[3]的偏微分方程.

3　体系结构

3.1　硬件环境

神威·太湖之光的硬件性能指标如表 1 所示，主要包括性能、功耗等信息，其系统总体架构主要分为两大部分：高速计算系统和辅助计算系统，与高速计算系统和辅助计算系统对应的有在线存储和近线存储，再加上诸如众核 CPU 基础软件、并行操作系统环境、并行语言环境等众多软件系统就构成整个系统硬件设备. 表 1 也对比了太湖之光和天河二号的各项指标，可以看出太湖之光超级计算机的各项指标性能较天河二号都有巨大提高，这得益于国产自主研发的“申威 26010”芯片的设计和突破.

表 1　硬件性能指标对比

类别	神威·太湖之光	天河二号
峰值	125 PFLOPS	53.62 PFLOPS
Linpack 性能	93 PFLOPS	30.65 PFLOPS
Linpack 功耗	15.37 MW	17.6 MW
功耗性能比	6.05 GFLOPS/W	1.901 GFLOPS/W

在天河二号类似异构环境上，已开发出大量科学数值计算应用程序，进行并行模拟和优化研究[4-8]. 由于太湖之光的体系结构与天河二号存在明显不同，所以现有软件难以跨平台高效使用，因此针对太湖之光的特殊体系结构，移植和优化数值计算库成了当务之急.

① http://www.mcs.anl.gov/petsc/petsc-current/src/snes/examples/tutorials/ex5.c.html

② http://www.mcs.anl.gov/petsc/petsc-current/src/snes/examples/tutorials/ex19.c.html

3.2 申威体系结构

作为国产自主研发的芯片“申威 26010”,是国产芯片的重大突破,该芯片是异构众核处理器[9-10](如图 1).申威众核处理器采用片上融合的异构体系结构,由 4 个异构群构成,每个异构群包括一个主核、64 个从核构成的从核簇、异构群接口和存储控制器,整芯片共 260 个计算核心,众核处理器还集成系统接口总线,用于连接标准 PCIe 接口实现片间直连和互连,管理与维护接口实现系统管理、维护与测试. 4 个异构群和系统接口总线通过群间传输网络实现存储共享和通信.

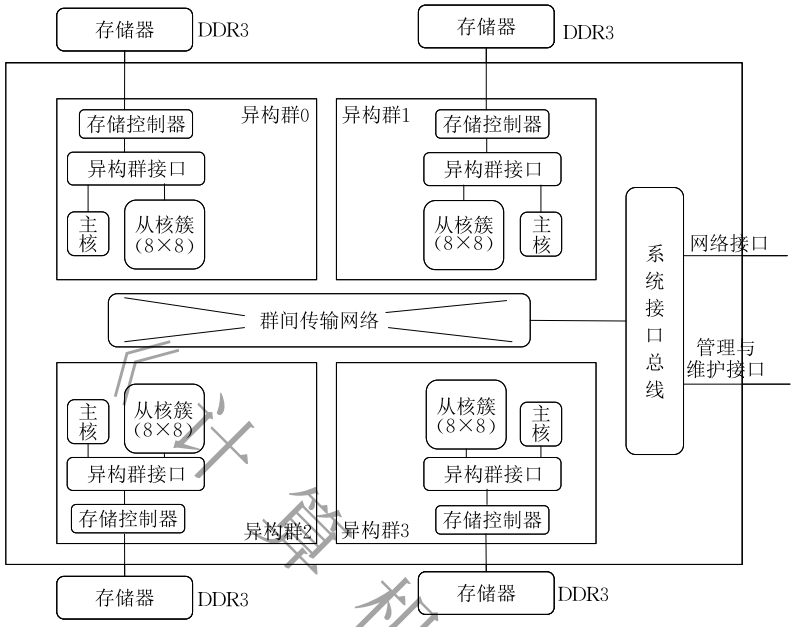


图 1 申威众核处理器结构图

主核主频为 1.5GHz,四译码七发射,每核组内存 8GB,L1cache 大小为 32KB,L2cache(数据 cache 和指令 cache 混合)大小为 256KB,运用高速暂存缓存器(SPM)[11-15],其常用于嵌入式实时系统. 从核主频为 1.5GHz,七发射,可以通过 gld/gst 直接离散访问主存,也可以通过 DMA 方式批量访问主存,从核阵列之间可以采用寄存器通信方式进行通信. 从核局部存储空间大小为 64KB,指令存储空间为 16KB. 每个主、从核核心都支持一个 256 位宽的向量化操作.

主从核的异构体系设计提供了灵活的异构编程模型,有以下 4 种:

(1) 主从加速并行模式

主核主要完成无法用众核并行部分的计算以及通信,而在从核进行任务计算时,主核等待.

(2) 主从协同并行模式

主核和从核作为对等的个体进行并行计算,根据各自计算能力进行负载分配,共同完成核心段的计算.

(3) 主从异步并行模式

在从核进行加速计算的同时,主核完成其他计算、通信或 I/O 等操作,提高主从协作的并行效率.

(4) 主从动态并行模式

主核负责任务分配,从核负责取得新计算任务、完成计算和写回计算结果.

根据程序特点选取合适的异构编程模型可取得更好的加速效果.

主从核异构体系结构特点也引发了一些思考:(1) 程序单节点运行加速效果如何,多节点的 MPI 可扩展性如何?(2) 从核计算任务有规律的离散访问主存时,可调整数据顺序使得离散访问变成顺序批量访存实现访存优化,从核计算任务随机离散访存时,程序性能受访存影响如何?(3) 从核批量访问主存时,可设计数据传输和计算的异步优化,同时从核数量的变化对数据传输的影响如何?(4) 从核局存 64KB,表明每次从主存传输到从核局存数据有限,传输量对性能影响如何?(5) 核心七发射,意味需要充分利用流水线,支持 256 位宽向量化操作,表明可以采用向量化操作进行优化,较低主频体现更好的高效能[16].

4　并行算法设计

本节中，首先确定核心热点函数并分类，然后选取合适的异构编程模型对热点函数的异构并行算法进行阐述.

4.1　函数分类

移植 PETSc 函数库，并在主核运行上述两个实例，记录各核心函数时间，得到热点函数如表 2 所示.

表 2　热点函数

函数名称	函数名称
MatSolve	*MatMult*
VecMDot	*VecNormalize*
VecMAXPY	——

深入分析，这些核心热点函数可分为三类，第一类是 *MatSolve* 函数：存在数据依赖无法并行，如何对它进行优化，会在最后部分进行讨论；第二类是 *MatMult* 函数：虽然没有数据依赖但是存在访存瓶颈；第三类是后两行的函数：数据依赖较小，且主要是计算耗时，而访存耗时较小，具有良好的异构并行特点，所以后文介绍的优化工作也是主要针对这些函数. 本文选取 *VecMDot* 函数和 *MatMult* 函数作为并行算法设计的典型进行说明. 其中 *VecMDot* 函数实现多向量点积，*MatMult* 函数是实现稀疏矩阵向量乘(SpMV)，由于 SpMV 广泛应用于各种科学计算中，其有许多异构体系下的算法设计和研究[17-20].

针对这两个函数，选取的编程模型为主从异步并行模式(如图 2). 在并行算法设计时，需要写主核代码和从核代码，主核代码实现数据的读入、调用从核函数、数据通信和部分计算，从核代码实现任务划分、数据计算和结果写回.

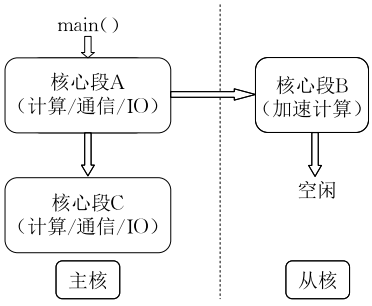


图 2　主从异步并行模式

4.2　异构并行算法设计

在这小节中，主要介绍 *VecMDot* 函数和 *MatMult* 函数的异构并行任务划分和基本算法.

VecMDot 函数任务划分如图 3，将向量 $\boldsymbol{x}$ 和向量 $\boldsymbol{y}[i]$以长度 T 均匀划分到每个从核进行计算，各从核计算结果写回主核并进行累加得到最终结果. 对于每个从核，有传输和计算两个过程，每次从主核传输大小为 L 的数据到从核进行计算，在经过 T/L 次上述过程后，不足 L 的余数部分($T\%L$)由从核直接访问主存计算(由于这部分数据量较小，所以直接访问主存的的时间消耗要小于数据在从核与主核之间的两次传递时间). 当所有 $\boldsymbol{y}[i]$全部计算完后，计算结果由从核返回给主核. 这种任务划分是考虑到输入数据是一个向量数组，先针对一个向量的并行，再进行多个向量的通用处理.

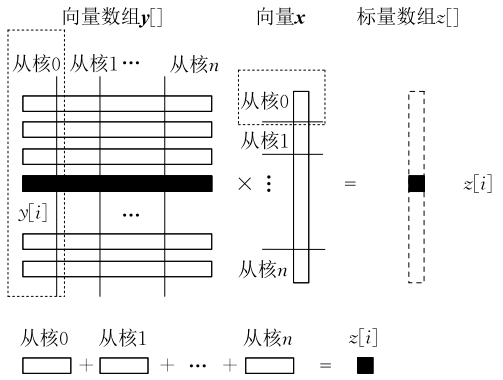


图 3　*VecMDot* 任务划分图

算法 1.　*VecMDot*.

输入：向量 ***xin***、向量数组 ***yin***[]、数组长度 *nv*

输出：标量数组 *z*[]

```
//主核代码
1.  PetscFunctionBegin
2.  VecGetArrayRead(xin,&x)
3.  For i=0,1,2  nv-1
4.    VecGetArrayRead(yin[i],&y[i]);
5.  End for
6.  athread_spawn(VecMDOT_Slave,&this_Arg)
7.  For i=0,1,2  nv-1
8.    z[i]=0.0;
9.    For j=0,1,2  THREADS_NUM-1
10.     z[i]+=zz[j+i*nv]
11.   End for
12. End for
13. PetscFunctionReturn(0);
//从核代码 VecMDOT_Slave
14. __thread_local type name;
15. my_id=get_row()×8+get_col();
16. start=my_id all/THREADS_NUM;
17. end=(my_id+1) all/THREADS_NUM;
18. k=start+(end-start)%L;
19. For i=0,1,2  nv-1
20.   z[i]=0.0;
```

```
21. End for
22. For start=start, start+1, start+2, …, start+k−1
23.   For j=0,1,2  nv−1
24.     z[j]=y[j][start]×x[start]+z[j];
25.   End for
26. End for
27. For i=start, start+L, start+2L, …, end
28.   athread_get 将数据从主核起始地址为 x+i,长度为 L 的一段传输到从核局存 xx[L]数组中
29.   For k=0,1,2  nv−1
30.     athread_get 将数据从主核起始地址为 y[k]+i,长度为 L 的一段传输到从核局存 yy[L]数组中
31.     For j=0,1,2  L−1
32.       z[k]=xx[j]×yy[j]+z[j];
33.     End for
34.   End for
35. End for
36. athread_put 将每个从核计算的结果 z[nv]传输到主核数组 zz[THREADS_NUM][nv]中
```

在算法 1 中 1～13 是在主核上运行,其中 6 是调用从核函数 *VecMDOT_Slave*,*this_Arg* 是结构体参数,保存从核计算所需的所有数据地址. 14～36 在从核上运行,14 是在从核局存为所有计算用到的变量开辟空间,所有变量必须用 *__thread_local* 声明存在于该从核局存中,15～18 是任务划分过程:*my_id* 是线程号、*all* 是向量长度、*THREADS_NUM* 是线程个数(从核数)、*L* 是传输数据的长度,19～36 是对划分的任务进行计算. 算法 1 根据如图 4 的任务划分方式,从主存中循环读取输入向量,并分成 *THREADS_NUM* 段,每段以长度 *L* 循环传输到每一个从核线程进行乘加运算,每个从核线程运算结果传输回主存,主核进行每个从核运算结果的累加得到最终结果.

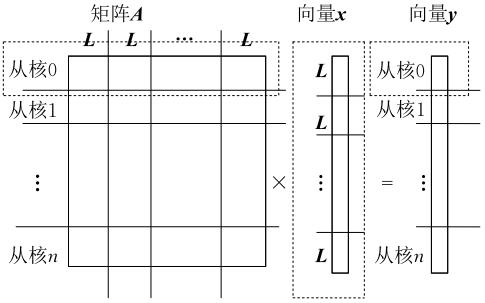


图 4 *MatMult* 任务划分图

MatMult 函数任务划分如图 4,矩阵 ***A*** 按 *T* 行均匀划分给每个从核,每个从核计算从主核传输过来大小为 *L* 数据段,循环传输、计算、再将每个从核计算结果写回主核内存.

算法 2 展示了稀疏矩阵向量乘的基本并行方法,矩阵 ***A*** 按行压缩格式存储,*aa* 是矩阵非零元素,*aj* 是非零元素所在列,*ii* 是索引. 在本文实验用例中,矩阵 ***A*** 是 Petsc 函数库生成的矩阵,其每行非零元素个数为 3 到 4 个,所以每个从核计算的负载基本一致.

算法 2. *MatMult.*

输入:矩阵 ***A***、向量 ***xx***、向量 ***yy***

输出:向量 ***yy***

```
//主核代码
1.  PetscFunctionBegin
2.  VecGetArray(yy,&y)
3.  VecGetArrayRead(xx,&x)
4.  athread_spawn(MatMult_Slave, &this_Arg)
5.  PetscFunctionReturn(0);
//从核代码 MatMult_Slave
6.  __thread_local type name;
7.  my_id=get_row()×8+get_col();
8.  start= my_id×all/THREADS_NUM;
9.  end =(my_id+1)×all/THREADS_NUM;
10. k=start+(end−start)%L;
11. For start=start, start+1, start+2, …, start+k−1
12.   N=ii[start+1]− ii[start];
13.   athread_get 将起始地址为 aa 和 aj,长度为 N 的数据从主核传输到从核的 mata 和 mataj 数组中
14.   aa+=N, aj+=N, sum=0.0;
15.   For j=0,1,2,3,…,N−1
16.     sum+=(mata[j])×x[mataj[j]];
17.   End for
18.   y[start]=sum;
19. End for
20. For i=start, start+L, start+2×L, start+3×L, …, end
21.   N=ii[i+1]−ii[i], now=0;
22.   athread_get 将起始地址为 aa 和 aj,长度为 N 的数据从主核传输到从核的 mata 和 mataj 数组中
23.   For j=0,1,2,3,…,L−1
24.     P=ii[i+j+1]−ii[i+j];
25.     For k=0,1,2,3,…,P−1
26.       matxx[k]=x[mataj[now+j]];
27.     End for
28.     matyy[k]=mata[now]×matxx[0];
29.     For k=1,2,3,…,P−1
30.       matyy[k]+=mata[now+k]×matxx[k];
31.     End for
32.     now+=P;
```

33. End for

34. $aa+=N$, $aj+=N$;

35. *athread_put* 将每个从核的 *matyy* 数组中的数据传输到主核首地址为 $\boldsymbol{y}+i$，长度为 L 的一段内存中

36. End for

在算法 2 中 1～5 在主核上运行，6～36 在从核上运行，主核只负责通信，计算全部由从核完成. 由于矩阵是稀疏格式的，在从核计算时对向量 $\boldsymbol{xx}$ 的访问为随机不连续访存，从核局存有限，因此无法将 $\boldsymbol{xx}$ 的数据直接传输到从核计算，只能通过从核访问主存获取向量 $\boldsymbol{xx}$ 对应的数据计算.

根据该主从核算法设计，主核处理乘法运算时间为 t_1，加法运算时间为 t_2，从核处理乘法运算和加法运算时间为 t_3 和 t_4，主从核一轮次通信时间为 t_p，传输 n 次，从核数目为 N，则并行算法时间复杂度如下：

$$t_{串}=t_1+t_2 \tag{1}$$

$$t_{并}=\frac{t_3+t_4}{N}+n\times t_p \tag{2}$$

$$Sp=\frac{t_{串}}{t_{并}}\approx\frac{t_1+t_2}{\frac{t_3+t_4}{N}+n\times t_p} \tag{3}$$

5　性能优化

本节主要介绍申威异构框架上的并行优化过程和方法，包括访存优化、去相关性优化和向量化优化.

5.1　访存优化

申威芯片没有采用传统的缓存技术而是每个异构群匹配一个 SPM(高速暂存器)，这样的设计带来了两个方面的影响. 相比缓存技术，SPM 方式给程序优化提供更多的主观能动性，但是给程序员更多编程挑战，大大增加程序的复杂性，所有内存访问操作的预取优化都要手动完成. 申威芯片提供 DMA 接口支持 SPM 与主存间的数据传输，在 128 B 数据对齐时，DMA 操作会达到最佳性能，与直接读取主存数据相比 DMA 操作的收益更大，由于库 PETSc 数学库中的大多数函数可以 SIMD 处理，所以手动循环预取的方法是采用隐藏数据访问延迟. 在每个异构群的并行区域内，内存的访问很大程度地限制了并行效率，因此利用异步 DMA 的内在特征进一步提高并行效率，运用双缓冲机制优化.

在并行算法的基础上加入双缓冲机制(异步)，可以实现计算和通信最大限度的相互隐藏取得良好加速效果. 双缓冲机制就是在计算核心的局部存储空间上申请 2 倍于所需数据大小的空间，以便存放两份同样大小且互为对方缓冲的数据. 通信双缓冲通过编程来控制和实现，具体过程如下：除了第一轮次(最后一轮次)读入(写回)数据的通信过程之外，当计算核心进行本轮次数据计算的同时，进行下一轮次(上一轮次)读入(写回)数据的通信. 此时计算核心数据通信部分开销分为两部分，一部分是不可隐藏部分，另外则是可以与计算开销相互隐藏部分. 其中不可隐藏部分开销为第一轮次读入与最后一轮次写回的数据通信开销之和，即单轮次通信部分的开销 t_p. 相应的可以与计算开销相互隐藏的数据通信开销为 $t_p\times(n-1)$，此时计算核心上计算开销与式(1)、(2)中一样，那么众核加速比的计算表达式具有如下的形式：

$$Sp=\frac{t_{串}}{t_{并}}\approx\frac{t_1+t_2}{\mathrm{Max}\left[\frac{t_3+t_4}{N},(n-1)\times t_p\right]+t_p} \tag{4}$$

该众核加速比计算表达式表明，计算核心计算开销和部分的通信开销实现了隐藏. 考虑计算密集型程序的情形，计算核心计算开销大于通信开销，而单轮次的通信开销又特别小，根据上面计算表达式得到众核加速比将会接近从核核心数 N. 如果将众核加速比与众核具有的计算核心个数之比定义为众核并行效率，那么此时众核并行效率也将接近于 1，即计算密集型更能够充分地发挥众核的优势.

另外由于从核计算核心专属的局存空间不大、延迟小. 因此在针对存在不可避免的大量离散访存的程序进行众核算法设计时，就可以利用众核的特点来避免离散访存并获得理想加速比. 因此在针对存在大量离散访存的程序段的众核并行设计过程中：首先在计算核心上将原来离散的数组调整成方便通信的读入和写回的存储顺序，然后计算核心进行通信读入数据、计算和通信写回数据，最后计算核心将写回的数据再次调整回原来的存储顺序，这种算法是 Gather-Scatter[21]，尽管相较于原来的算法，增加了前后两个数组存储顺序调整的过程，但由于上述过程都是在计算核心上来完成的，两个存储顺序调整所导致的计算核心开销增加得不大. 另外，计算方面则由于计算核心对专属局部存储空间访问延迟小使得计算核心计算开销大大减小.

但是，双缓冲机制的加入会造成片上存储的减小，因此针对不同计算任务，首先尝试不加入双缓冲机制，通过测试找到最优的数据通信量，然后，以该

通信量作为双缓冲时的通信量.如果此时片上存储空间仍未满,则为理想结果;否则若片上存储空间不够,在保证 128B 对齐的前提下,以原通信量的 1/2,1/4,1/8,…作为通信减量依次改变数据通信量大小,直到测试找到最好的数据通信量为止.

5.2 消除相关性

在申威处理器上,每个从核核心的计算是七级流水,因此在算法设计时要摒除相关性才能充分利用流水线,以矩阵向量乘为例(如图 5 所示),除去相关性就要调整循环次序,而后对矩阵 $\boldsymbol{A}$ 的访问顺序也随之改变,宏观来看对矩阵 $\boldsymbol{A}$ 的访问是不连续的.因此在算法优化时,每次是取矩阵前 N 行数据从主核传输到从核进行流水计算,这样对每个从核的局存,矩阵 $\boldsymbol{A}$ 计算访问从核局存是连续的,所以若有 $CoreNumber$ 个核心,相当于每个核心计算 $N/CoreNumber$ 行的数据.

```
!有相关性
do i=1,im
 do j=1,jm/8
  y[i]=A[i][j]×x[j]+y[i]
  y[i]=A[i][j+1]×x[j+1]+y[i]
  …
  y[i]=A[i][j+7]×x[j+7]+y[i]
 end do
end do
```

```
!无相关性
do j=1,jm
 do i=1,im/8
  y[i]=A[i][j]×x[j]+y[i]
  y[i+1]=A[i+1][j]×x[j]+y[i+1]
  …
  y[i+7]=A[i+7][j]×x[j]+y[i+7]
 end do
end do
```

图 5 相关性的除去实例

5.3 向量化操作

申威处理器每个主、从核核心都支持一个 256 位宽的向量化操作,具体如下:

(1) 32×8 的定点运算

一次操作处理 8 个 32 位的定点运算

(2) 256×1 的定点运算

一次操作处理 1 个 256 位长整型运算

(3) 64×4 的单精度浮点运算

一次操作处理 4 个单精度浮点运算

(4) 64×4 的双精度浮点运算:

一次操作处理 4 个双精度浮点运算

利用向量化操作来优化时要注意对界问题,因为不对界的 Load/Store 会引发异常,操作系统收到异常信号后会将这些 Load/Store 拆分成标准类型的 Load/Store,这样会大大的降低性能.在申威处理器构架中,向量类型的数据项在内存中除了 floatv4 类型是 16 字节对界,其余都是 32 字节对界.

解决对界问题后,对程序进行向量化优化时要考虑程序形式对向量化的限制,有如下 3 种:

(1) 相关性限制

没有数据相关才能被向量化,数据相关性分析包括寻找内存访问可能重叠的条件.给定程序的两个访问,相关的条件一是访问的变量在内存中是同一个(或者重叠)区域的别名,二是对数组访问,分析下标的关系是否可能相等.

(2) 循环结构限制

循环可以是常用的 for,while-do 结构,也可以是使用 goto 和标号组成的,但是循环必须只有一个入口和一个出口时才能被向量化.

(3) 循环退出条件限制

循环退出条件决定循环执行的迭代次数.向量化需要迭代次数是可以计算的,也就是说迭代次数必须是常数、循环不变量以及外层循环控制变量的线性函数.

确定无上述 3 个限制条件的循环才可向量化,向量化时,可以利用循环分裂、循环展开、循环融合、变量替换和操作替换等手段来实现,而为了更好的向量化就要合理使用数组:程序中对数组的操作需要尽量的连续访问;更有效的使用 Cache,向量化与 Cache 对程序的要求是一样的,都是访问内存的连续区域,即访问同一个 Cache 行;尽量将数据转换成扩展数据类型进行向量化,这样可以更简便快速实现和调试,另外扩展数据类型的操作还提供了乘加等更加方便快速的运算操作.

6 实验结果和对比

6.1 从核性能测试结果

单个申威众核处理器就是太湖之光超级计算机的一个节点,在单节点情况下使用基准程序对主从核性能进行评估,得到主核性能为单个从核性能的 4 倍左右,再根据式(3)和式(4),对于单节点若忽略 t_p,那么理论加速比 Sp 为单节点参与运算的从核个数除以 4,即全片理论加速比为 16,主核访存带宽为 9.918GB/s.

从图 6 中看出随着从核个数增加,时间下降速

度减缓，从核数目越大时间减缓趋势越小. 这是由于单节点上可运算规模(由内存大小决定)限制了从核计算能力的充分发挥，从核数达到 16 时，矩阵规模使得从核计算能力接近饱和.

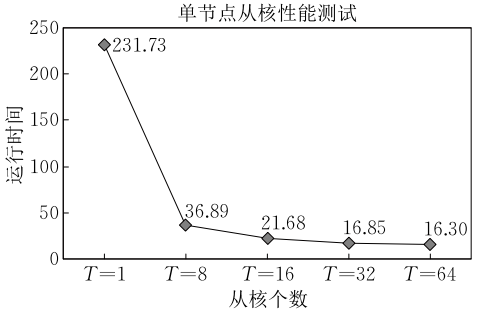


图 6　单节点从核性能测试结果

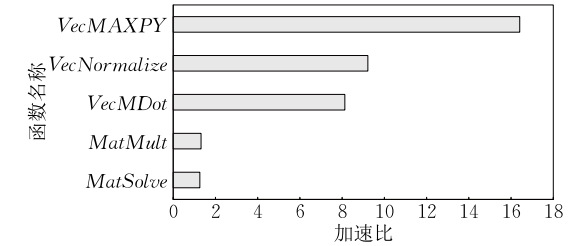
从核访存带宽测试便于分析访存瓶颈，测试结果如表 3 所示.

表 3　带宽测试结果

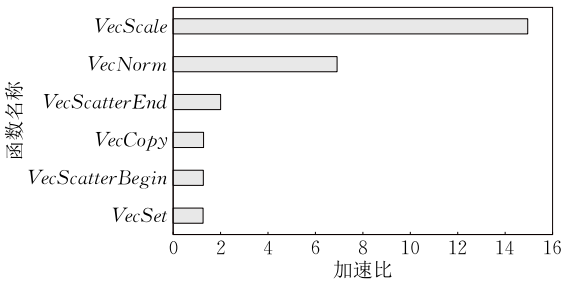
	从核访存带宽	
粒度/B	1pebw/(GB·s^{-1})	64pebw/(GB·s^{-1})
8	0.05	0.99
16	0.10	1.99
32	0.20	3.92
64	0.41	7.96
128	0.80	15.77
256	1.47	28.88

6.2　单节点优化结果

编译并行优化后的 PETSc 函数库，运行 ex5，记录优化后的核心函数运行时间，与优化前对比得到热点函数的加速比如图 7 所示.



(a) 热点函数加速比



(b) 非热点函数加速比

图 7　热点函数和非热点函数加速比

从图 7 中可以看出：第三类函数单节点并行优化效果良好，*VecMAXPY* 函数基本达到理论加速比，而 *MatMult* 函数和 *MatSolve* 函数加速比较低. 同热点函数一样，非热点函数只有一部分适合并行，由于其所占时间比重较小，深入优化意义不大.

对比总运行时间，得到整体加速比为 2，表明单节点情形下并行优化具有较好加速效果.

6.3　可扩展性测试结果

当节点数较多，输入规模要达到一定大小才能体现效果，因此以 256 节点下不同输入规模的运行结果做基准时间，而规模由输入的 x 和 y 确定. 运行 ex19 并行可扩展程序，选取 4 个热点函数可扩展结果进行展示说明，如图 8～图 11 所示(其中图例为规模，2048 表示 x 和 y 都为 2048；横坐标表示节点数；纵坐标表示不同规模输入在该节点数下运行时间相对于 256 个节点运行时间的加速比)：

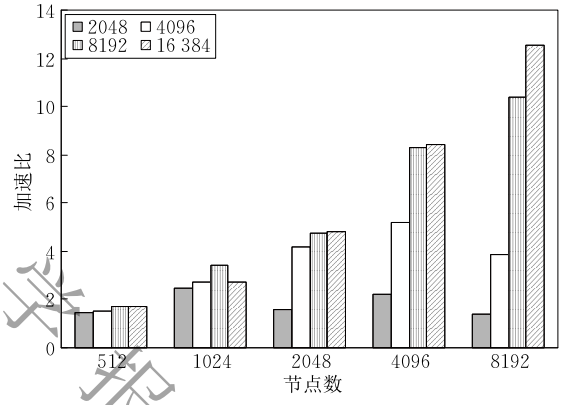


图 8　*VecMDot* 函数的可扩展性测试加速比

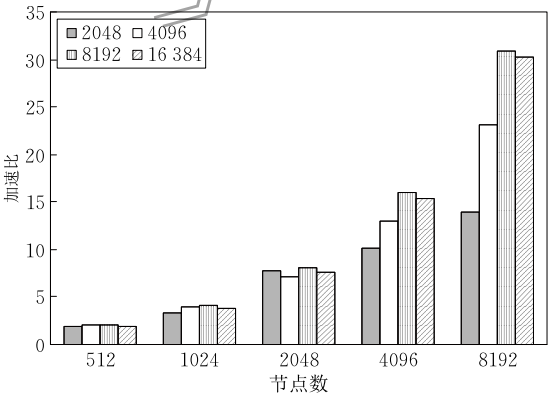


图 9　*MatMult* 函数的可扩展性测试加速比

从 4 个函数可扩展性测试结果中，可以看出 4 个热点函数在不同数据规模时，加速比随着计算节点数的增加呈线性增加的趋势，更具体的表现为加速比随着节点数成倍增加而近乎成倍增加，说明具有良好的可扩展性.

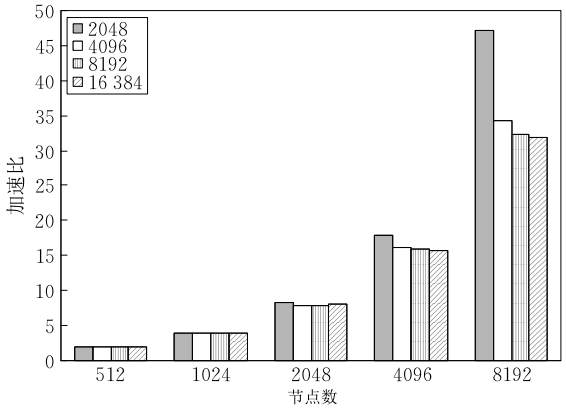


图 10 *VecMAXPY* 函数的可扩展性测试加速比

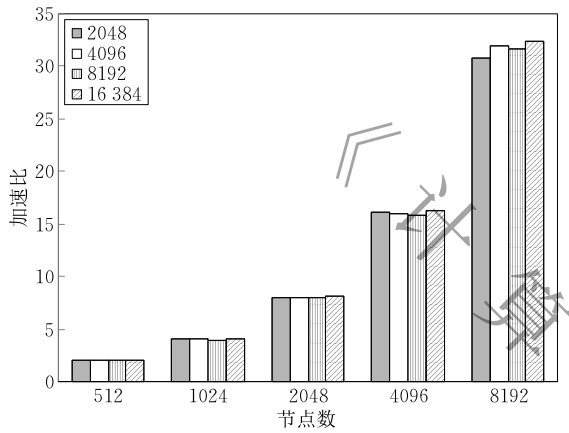


图 11 *VecMatSolve* 函数的可扩展性测试加速比

为更好分析性能随节点变化的关系，给出 PETSc 数学库在该程序应用下除去准备时间（准备时间是指解法器开始求解前给矩阵染色预处理时间）的总时间随节点变化的图像，如图 12 所示.

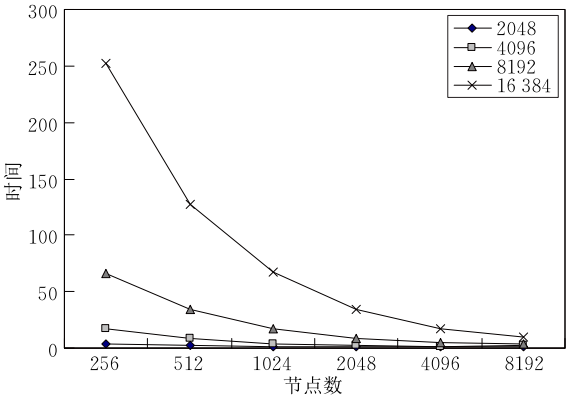


图 12 整体程序的可扩展性测试结果

图 12 中显示了整体程序运行时间随着计算节点数变化的趋势，运行时间随着节点数的增加而线性降低. 需要特别指出的是，图中的运行时间是在减去矩阵预处理后的计算时间，预处理过程使用染色算法对矩阵进行压缩，以缩短迭代所需时间.

6.4 相关工作对比

Cuomo，Galletti 和 Giunta 等人用 10 帧组成的 SAR 图像序列进行测试，其对应于时间间隔为 15 min 的天气状况地图，每个图像大小是 555×845，同 ex19 一样也是利用 Jacobi 预处理的广义极小残量法求解偏微分方程. 实验环境是 Quadro K5000 GPU 和 Intel Core i7 CPU（2.8 GHz，8 GB Cache）搭建的集群[2]. 本文用程序 ex19 测试进行对比，其中输入数据规模大小为 $x=1024$，$y=1024$，结果如表 4，由于应用程序和输入数据规模不同，对比运行时间没有意义，只需要分析加速比情况. 其中 Intel CPU 是多核 CPU，一个 CPU 看成一个节点，每个申威节点只启动单个主核计算，两者单节点加速比都为 1.

表 4 Inter CPU 与申威主核对比

节点数	Intel CPU		申威	
	时间	加速比	时间	加速比
1	259	1.00	46.20	1.00
2	131	1.98	23.40	1.97
4	88	2.94	12.10	3.83
8	60	4.31	6.36	7.26

表 5 中一个 CPU＋GPU 作为一个节点，每个申威节点所有主从核都启动计算，CPU＋GPU 加速比$_1$为 n 个 GPU＋CPU 节点运行时间除以单个 CPU 运行时间，加速比$_2$为 n 个 GPU＋CPU 节点运行时间除以单个 GPU＋CPU 节点运行时间，申威加速比$_1$为 n 个节点主核＋从核运行时间除以单节点主核运行时间，加速比$_2$为 n 个节点主核＋从核运行时间除以单节点主核＋从核运行时间.

表 5 CPU＋GPU 异构与主从核异构对比

节点数	CPU＋GPU			主核＋从核		
	时间	加速比$_1$	加速比$_2$	时间	加速比$_1$	加速比$_2$
1	50	5.18	1.00	35.60	1.30	1.00
2	29	8.90	1.72	18.50	2.49	1.92
4	20	12.95	2.50	9.59	4.82	3.71
8	13	19.93	3.85	5.00	9.23	7.12

根据表 4、表 5 画出加速比图像如图 13～图 15 所示.

对比 CPU 和主核，表明申威主核可扩展性比 CPU 可扩展性好，对比 CPU＋GPU_1 与主核＋从核_1，表明 CPU＋GPU 对单 CPU 的加速效果十分明显，单节点的主从核加速性能不及 CPU＋GPU，而 CPU＋GPU_2 和主核＋从核_2 的对比表明的主从核加速模式的可扩展性较 CPU＋GPU 有明显优势.

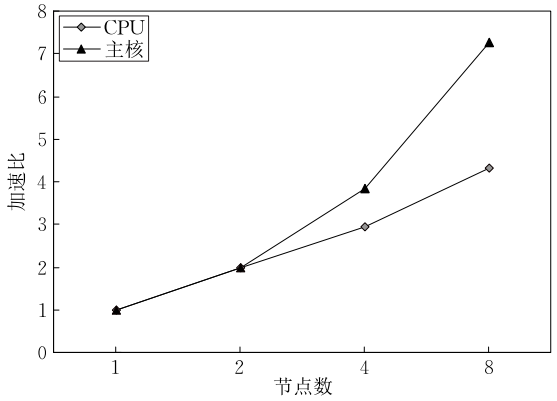


图 13　CPU 与主核加速比对比

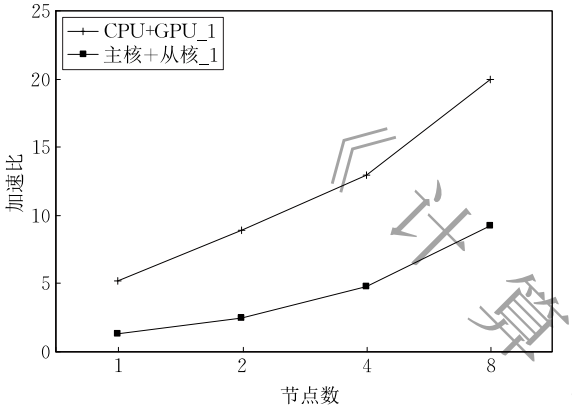


图 14　CPU+GPU_1 与主核+从核_1 加速比对比

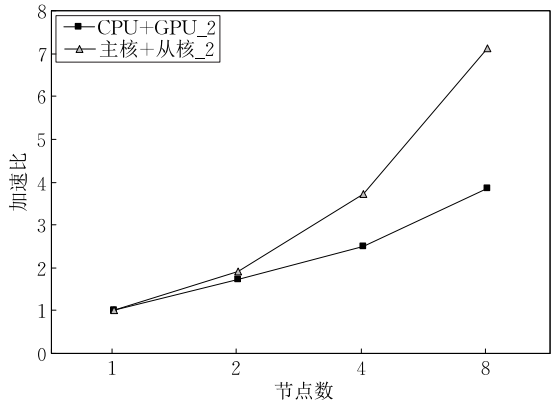


图 15　CPU+GPU_2 与主核+从核_2 加速比对比

7　结　论

根据 PETSc 函数库下的应用程序在申威异构处理框架的移植优化结果对比，可以得到以下结论：

(1) 单节点情况

对于没有数据依赖，且主要是计算耗时，而访存耗时较小的函数，其性能加速比基本达到理论加速比，表明申威异构体系下有良好的并行性. 对于不连续离散访存的 *MatMult* 函数，其计算所需访存结果如下(其中整型长度 32 B，浮点型长度 64 B，循环次数为 51 次，总时间为 0.3121 s)：

通过表 6 的测试数据计算得到访存带宽如表 7. 由于函数对于浮点数的访问为不连续随机访问，因此每次访存取一个浮点数，访存粒度为 64；从表 4 中可知：访存粒度为 64 时最大带宽为 7.96 GB/s，但是表 7 中显示计算所需带宽为 12.97 GB/s，所以 *MatMult* 函数受异构体系结构从核访存限制无法进一步优化. 而 *MatSolve* 函数不适合单节点并行优化，只能做主核优化.

表 6　*MatMult* 函数访存测试结果

	整型访问次数	浮点访问次数
每个循环	780 288	1 331 200
总计	39 794 688	67 891 200

表 7　访存带宽计算

访存总计/GB	平均每秒访存/(GB·s^{-1})	浮点访存/GB	平均每秒浮点访存/(GB·s^{-1})
4.395 263 67	1.311 57E-08	4.046 630 86	12.965 814 99

(2) 分布式存储下的多节点 MPI 情形

分布式存储下的多节点 MPI 可扩展效果良好，对于所有函数其加速比随节点数的成倍增加近乎成倍增加，基本达到线性加速.

(3) 单节点从核个数性能和主从核传输量影响

在单节点情形下，程序运行时间会随从核个数增加而减少，但减少速度变得越来越缓慢，这是由于单节点上可运算规模(由内存大小决定)限制了从核计算能力的充分发挥，从核数达到 16 时，矩阵规模使得从核计算能力接近饱和. 而主从核传输量的大小不发生极大变化时，对程序性能影响不大.

(4) CPU-GPU 集群对比

由申威芯片核心频率较低，GPU 线程数也远大于申威从核数，单个申威节点性能将无法和单个 CPU+GPU 相比，但多节点扩展情况申威有明显优势，表明 PETSc 数学库下的应用程序在申威体系结构上有更好的可扩展性.

综上，PESTc 数学库作为科学数值计算的共性数学库，用来求解一系列复杂科学问题，其有大量的矩阵向量运算函数，这些函数都有良好的并行性，能充分发挥申威异构处理器的并行性能，而其他不易单节点并行的函数也能高效的在太湖之光超级计算机上多节点扩展. 因此，实现一个适应于太湖之光体系结构的类 PETSc 数学库的共性数学库，十分有意义. 其能够充分利用太湖之光的优势去更快地解决当前大规模的科学工程数值计算问题.

8 未来工作

下一步工作将在以下方向开展：

(1) PETSc 函数库中用来求解科学数值计算问题的矩阵格式经常用到稀疏格式，将稀疏矩阵转化为稠密矩阵进行并行计算代价较大，拟考虑设计适应神威体系结构的稀疏矩阵 BLAS 库，供 PESTc 调用，以提高并行优化效果.

(2) PETSc 函数库中不同的解法器求解同一问题步骤和迭代方法不同，调用的底层核心函数也不同，将针对不同解法器进行性能优化工作.

致 谢 感谢国家超级计算无锡中心给予作者实验测试环境！

参 考 文 献

[1] Minden V, Smith B, Knepley M G. Preliminary implementation of PETSc using GPUs//Yuen D A, Wang L, Chi X, et al, eds. GPU Solutions to Multi-scale Problems in Science and Engineering. Berlin Heidelberg, Germany: Springer, 2013: 131-140

[2] Cuomo S, Galletti A, Giunta G, et al. Toward a multi-level parallel framework on GPU cluster with PetSC-CUDA for PDE-based optical flow computation. Procedia Computer Science, 2015, 51(1): 170-179

[3] Bruneau C H, Saad M. The 2D lid-driven cavity problem revisited. Computers & Fluids, 2006, 35(3): 326-348

[4] Li Zhi-Hui, Jiang Xin-Yu, Wu Jun-Lin, et al. Study on high performance parallel algorithm for spacecraft re-entry aerodynamics in the whole of flow regimes using Boltzmann model equation. Chinese Journal of Computers, 2016, 39(9): 1801-1811(in Chinese)
(李志辉，蒋新宇，吴俊林等. 求解 Boltzmann 模型方程高性能并行算法在航天跨流域空气动力学应用研究. 计算机学报，2016，39(9)：1801-1811)

[5] Lu Lin-Sheng, Dong Chao-Qun, Li Zhi-Hui. Research on parallelization of multiphase space numerical simulation. Computer Science, 2003, 30(3): 129-137(in Chinese)
(陆林生，董超群，李志辉. 多相空间数值模拟并行化研究. 计算机科学，2003，30(3)：129-137)

[6] Bian Yi, Yuan Fang, Guo Jun-Xia, et al. CPU+GPU heterogeneous computing orientated multi-objective test case prioritization. Journal of Software, 2014, 27(4): 943-954(in Chinese)
(边毅，袁方，郭俊霞等. 面向 CPU+GPU 异构计算的多目标测试用例优先排序. 软件学报，2014，27(4)：943-954)

[7] Ma A G, Tan C F, Cheng Y. Parallel implementation of neural network model for quadratic programming on GPGPU// Proceedings of the International Conference on Fuzzy Systems and Neural Computing (FSNC'11). Hong Kong, China, 2011: 241-245

[8] Shimokawabe T, Aoki T, Takaki T, et al. Peta-scale phase-field simulation for dendritic solidification on the TSUBAME 2.0 supercomputer//Proceedings of the Conference on High PERFORMANCE Computing Networking, Storage and Analysis. Seattle, USA, 2011: 1-11

[9] Fu H, Liao J, Yang J, et al. The Sunway TaihuLight supercomputer: System and applications. Science China Information Sciences, 2016, 59(7): 1-16

[10] Dongarra J. Report on the Sunway TaihuLight system. USA: Oak Ridge National Laboratory, Department of Electrical Engineering and Computer Science, University of Tennessee, Technical Report: UT-EECS-16-742, 2016

[11] Egger B, Lee J, Shin H. Scratchpad memory management in a multitasking environment.//Proceedings of the ACM & IEEE International Conference on Embedded Software. Atlanta, USA, 2008: 265-274

[12] Baker M A, Panda A, Ghadge N, et al. A performance model and code overlay generator for scratchpad enhanced embedded processors//Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis, CODES+ISSS. Scottsdale, USA, 2010: 287-296

[13] Bai K, Lu J, Shrivastava A, et al. CMSM: An efficient and effective code management for software managed multicores// Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis. Montreal, Canada, 2013: 1-9

[14] Bai K, Shrivastava A. A software-only scheme for managing heap data on limited local memory (LLM) multicore processors. ACM Transactions on Embedded Computing Systems, 2013, 13(1): 171-190

[15] Baker M A, Panda A, Ghadge N, et al. A performance model and code overlay generator for scratchpad enhanced embedded processors//Proceedings of the IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis. Scottsdale, USA, 2011: 287-296

[16] Zheng Fang, Zhang Kun, Wu Gui-Ming, et al. Architecture techniques of many-core processor for energy-efficient in high performance computing. Chinese Journal of Computers, 2014, 37(10): 2176-2186(in Chinese)
(郑方，张昆，邬贵明等. 面向高性能计算的众核处理器结构级高能效技术. 计算机学报，2014，37(10)：2176-2186)

[17] Li K, Yang W, Li K. Performance analysis and optimization for SpMV on GPU using probabilistic modeling. IEEE Transactions on Parallel & Distributed Systems, 2015, 26(1): 196-205

[18] Li K, Yang W, Li K. A hybrid parallel solving algorithm on

GPU for quasi-tridiagonal system of linear equations. IEEE Transactions on Parallel & Distributed Systems, 2016, 27(10): 2795-2808

[19] Yang W, Li K, Mo Z, et al. Performance optimization using partitioned SpMV on GPUs and multicore CPUs. IEEE Transactions on Computers, 2015, 64(9): 2623-2636

[20] Bell N, Garland M. Efficient sparse matrix-vector multiplication on CUDA. California, USA: Nvidia Corporation, Technical Report: NVR-2008-004, 2008

[21] Han H, Jung H, Eom H, et al. Scatter-gather-merge: An efficient star-join query processing algorithm for data-parallel frameworks. Cluster Computing, 2011, 14(2): 183-197

HONG Wen-Jie, born in 1993, M. S. candidate. His main research interests focus on high-performance computing.

LI Ken-Li, born in 1971, Ph. D., professor. Ph. D. supervisor. His research interests mainly include high-performance computing, parallel and distributed systems.

QUAN Zhe, born in 1982, Ph. D. His research interests include artificial intelligence and high-performance computing.

YANG Wang-Dong, born in 1974, Ph. D., professor. His research interests mainly focus on parallel computing.

LI Ke-Qin, born in 1963, Ph. D., professor, Ph. D. supervisor, IEEE fellow. His current research interests include parallel computing distributed computing, and cloud computing.

HAO Zi-Yu, born in 1978, Ph. D., senior engineer. His current research interests include high performance computer architecture, the design and optimization of algorithms.

XIE Xiang-Hui, born in 1958, Ph. D., senior engineer. His research interests include computer architecture and high-performance internet.

Background

The Sunway TaihuLight supercomputer is developed by the National Research Center for Parallel Computer Engineering and Technology, the first supercomputer equipped with the chip of "Shen Wei 26010" as the processor, which is independently developed by China, and the first supercomputer which can perform 100p calculations per second, ranking the first in the Top500 supercomputer in June this year. The rapid development of supercomputer hardware in China has highlighted the lag of the development of high-performance computing software. How to develop the potential computing power of supercomputer in high-performance computing applications has always been one of the main challenges in high-performance research. Often, most applications are implemented by calling the math library of the open source, and PETSc is the kind of math library. Therefore, the problem of the PETSc library's transplantation and optimization in the supercomputer has become an urgent problem.

The Sunway TaihuLight supercomputer has the fastest operating rate in the world, with the core of Shen Wei many-core processor, which is independently developed by China. PETSc is a math library which commonly used in high-performance computing application and umerical calculation, then how to give full play to the computing potential of the Sunway TaihuLight supercomputer in the parallel high-performance computing applications is the problem and challenge we facing now. For the kind of parallel application which uses the PETSc library as the underlying call, its upper-sided performance is determined by the optimization effects of the PETSc math library under the current architecture. Therefore, it is very important to achieve the parallel optimization of PETSc math library on the Sunway TaihuLight supercomputer. Then the experiment to realize the related numerical calculation function library on domestic heterogeneous supercomputing could be carried out. Because the PETSc math library is of large content, this paper designs and optimizes the parallel heterogeneous algorithm on the seven core functions of PETSc math library according to the requirement of this research.

This research is supported by the National Natural Science Foundation of China (Nos. 61432005, 91430214), the National Science Fund for Distinguished Young Scholars (No. 61625202), the National Natural Science Foundation of China Youth Project (No. 61602166), the State Key Laboratory of Mathematical Engineering and Advanced Computing, Open Fund Project, National Key Research and Development Program National Key Research and Development Program (Nos. 2016YFB0201402, 2016YFB0201900).