基于SW26010处理器的FT程序的性能优化

陶小涵 庞建民 高 伟 王 琦 姚金阳

(信息工程大学 郑州 450001) (数学工程与先进计算国家重点实验室 江苏 无锡 214125)

摘 要 "神威·太湖之光"是中国自主研发的超级计算机,其处理器芯片为国人自主研发的SW26010异构众核处理器,每个处理器内含有4个核组,每个核组包括1个主核和64个从核。NPB-FT程序的功能是利用快速傅立叶变换求解三维偏微分方程,其被广泛用于评测集群的计算和集合能力,因此选用FT程序对"神威·太湖之光"提供的多层次并行资源和体系架构的性能进行测试具有重要的意义。首先,利用加速线程库将程序改写为主从版本,使计算核心能够在从核上执行;其次,利用从核的寄存器通信以及主从核间的数据传输通道,消除FT程序中的数据转置过程;然后,实现了计算与通信隐藏,避免了核间通信时核内的计算资源处于空闲状态;最后,利用向量化和指令流水技术,提升程序的数据级并行和指令级并行。实验结果为:单核上3D-32规模的加速比为66,64核上3D-512规模的加速比为20,256核上3D-2048规模的加速比为46。

关键词 傅立叶变换,SW26010处理器,寄存器通信,通信隐藏

中图法分类号 TP301.6 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2019.04.050

Performance Optimization of FT Program Based on SW26010 Processor

TAO Xiao-han PANG Jian-min GAO Wei WANG Qi YAO Jin-yang

(Information Engineering University, Zhengzhou 450001, China)

(State Key Laboratory of Mathematical Engineering and Advanced Computing, Wuxi, Jiangsu 214125, China)

Abstract Sunway TaihuLight is a supercomputer independently developed by China. Its processor is SW26010 heterogeneous many-core processor, which is also independently developed by Chinese. Each processor includes four core-groups, and each core-group includes one management processing element(MPE) and 64 computing processing elements (CPEs). The function of NPB-FT program is to solve three-dimensional partial differential equations by using Fast Fourier Transform, and it is widely used in the evaluation of cluster computing and aggregation capabilities. Therefore, it is of great importance to use the FT program to analyze the multi-level parallel resources provided by Sunway TaihuLight and the performance of the architecture. First of all, the program is rewritten as master-slave version by accelerating athread library, so that the program core can be executed by the CPEs. Second, the data transposition process in the FT program is eliminated by using register communication of CPEs and the data transmission channel between the MPE and CPEs. Further, the computing and communication hiding are realized to avoid the computing resources in the core being in idle state while communicating between cores. Finally, the vectorization and instruction flow technology are used to enhance the program's data-level and instruction-level parallelism. The experimental results show that the 3D-32 program executing on a single core has an acceleration ratio of 66. The acceleration ratio of 3D-512 program executing on 64 cores is 20 while the acceleration ratio of 3D-2048 program executing on 256 cores is 46.

Keywords Fourier transform, SW26010 processor, Register communication, Communication hiding

1 引言

科学计算特别是高性能计算为现代科学研究提供了一种全新的手段,而"神威·太湖之光"超级计算机则为高性能计算提供了更高效的平台。作为世界首台峰值运算速度超过10亿亿次、并行规模超千万核的新型超级计算机,"神威·太湖之光"是中国自主研发的一台超级计算机,其突破了全系统高密度设计及高速互联技术。整机系统浮点峰值性能达到每秒12.5亿亿次,持续性能为每秒9.3亿亿次浮点运算,性能功耗比为60.5亿次/瓦,整机Linpack效率不低于70%[1]。

到稿日期:2018-02-11 返修日期:2018-04-03 本文受国家重点研发计划"高性能计算"重点专项(2016YFB0200503)资助。

陶小涵(1996-),男,博士生,主要研究方向为高性能计算、先进编译技术,E-mail:txh_0119@126.com;**庞建民**(1964-),男,博士,教授,博士生导师,主要研究方向为高性能计算、先进编译技术,E-mail:jianmin_pang@126.com(通信作者);**高 伟**(1988-),男,博士,主要研究方向为高性能计算、先进编译技术;**王 琦**(1992-),男,硕士生,主要研究方向为高性能计算、先进编译技术;**姚金阳**(1992-),男,硕士生,主要研究方向为高性能计算、先进编译技术。

其采用的 SW26010 异构众核处理器具有主从核并行模式，通过一个高性能的加速线程库来进行主从核任务分配，同时提供了 256 位 SIMD 向量化并行方式以及寄存器通信等核内部访存优化方式[2]，以实现程序的高效执行。

在超级计算机的性能评测方面，NPB(NAS Parallel Benchmark)是一个广泛使用的并行计算机评测基准程序。与 Linpack 不同，NPB 的目的不是测试系统最佳的浮点计算性能，而是测试系统运行用户实际应用程序的性能状况[3]。NPB 中的 9 个程序来自计算流体动力学应用软件，它们所采用的算法在很多领域都得到了应用，具有很强的代表性，NPB 的测试结果可以表现出一般应用程序的实际性能。其中，NPB-FT(fast Fourier Transform)程序利用快速傅立叶变换求解三维偏微分方程，主要被用来测试集合通信。快速傅立叶变换是一种离散傅立叶变换高效算法，它根据离散傅立叶变换的奇、偶、虚、实等特性，对离散傅立叶变换算法进行了改进[4]。本文采用了 3 种数据规模(3D-32 规模、3D-512 规模和 3D-2048 规模)的 FT 程序。

目前，在高性能计算领域已有研究者根据不同并行计算模型对 FT 程序进行了优化尝试，由文献[5]可知，采用 CPU/GPU 异构集群并行计算模型对 FT 程序进行优化，总体性能提高了 20%左右，但还没有研究利用主从核并行方式对 NPB-FT 测试程序进行优化尝试。FT 程序在 SW26010 处理器的一个主核上串行执行的效率极低，因此，针对“神威·太湖之光”特有的主从核结构，将“神威·太湖之光”强大的计算性能应用于并行计算机评测基准程序，需要调用从核对程序进行优化。基于上述研究初衷，本文对 FT 程序的过程和热点进行了分析，并通过寄存器通信、向量化等优化方法对程序热点函数进行了调优。以 3D-2048 规模为例，单主核串行执行 FT 程序需要 34862 s，调优后并行执行时间为 748 s，加速比达到 46 倍，3D-32 规模和 3D-512 规模的加速比分别可达到 66 倍和 20 倍，效果显著。

本文的主要贡献如下：

(1)在神威平台上对 FT 程序进行了一系列的分析和测试，找出了 FT 程序的性能瓶颈；

(2)针对神威处理器硬件的特性，对 3 种不同规模的 FT 程序进行了主从协同计算、数据传输、计算通信隐藏、寄存器通信等一系列的优化。

本文第 2 节介绍 SW26010 异构众核处理器；第 3 节介绍 NPB-FT 程序的流程和热点；第 4 节介绍主从协同计算、数据传输优化、计算通信隐藏及向量化等基于太湖之光的高效 FT 实现方法；第 5 节介绍性能测试与分析；最后总结全文。

2 SW26010 异构众核处理器

“神威·太湖之光”超级计算机采用的 SW26010 异构众核处理器，是国产自主研发的处理器，采用了片上计算阵列集群和分布式共享存储相结合的异构众核体系，使用 64 位自主申威指令系统[6]。

2.1 SW26010 处理器架构

每个 SW26010 处理器通过片上网络互连 4 个核组，每个核组包含一个主核、一个 8×8 的从核阵列，以及一个内存控制器。每个核组都有自己私有的内存空间，通过内存控制器与主核和从核相连，而所有的处理器均通过系统接口 SI 与其他处理器或者设备相连[7]。一块 SW26010 处理器和单个核组的结构如图 1 所示。

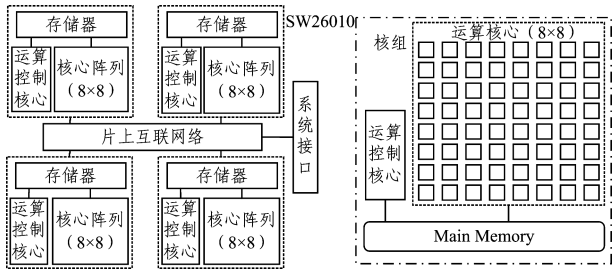

图 1 SW26010 处理器和单核组结构图

Fig. 1 General architecture of SW26010 processor and single core-group

主核是一个 64 位的 RISC 核心，可以在用户模式和系统模式下运行，支持完整的中断处理、内存管理、超标量和乱序执行等功能，因此主核是通信和管理的核心。主核拥有一个 32 kB 的 L1 指令 Cache、一个 32 kB 的 L1 数据 Cache 和一个 256 kB 的 L2Cache，同时支持 256 位的向量运算[8]。

从核也是一个 64 位的 RISC 核心，但是功能相对有限，其仅限于在用户模式下运行，同时不支持中断处理。从核的设计目标是实现计算能力的凝集，同时最小化微架构的复杂性。从核阵列按照 8×8 的结构排列，采用网络互连、支持低延迟的寄存器通信，每个从核拥有 16 kB 的 L1 指令 Cache 和一个 64 kB 的局部存储空间。

在主核与从核之间，从核可以通过 gld/gst 方式直接离散访问主存，即从核和主核共享一块 8 GB 的内存空间，也可以通过 DMA 方式批量访问主存。而从核阵列之间采用寄存器通信方式进行通信，由于从核在物理上是以 8×8 的阵列排布的，因此寄存器通信仅限于同行或者同列的从核之间进行通信，否则需要通过两次通信才能完成数据的交换。从核通过 put 函数可以进行同行或同列广播，也可以指定某一从核进行数据发送，当然也仅限于同行或同列的从核[9]。

2.2 主从加速并行方式

“神威·太湖之光”计算机系统支持主从加速并行方式等多种异构并行方法。这些方法中的程序都是由主核进入，然后由主核再调用从核进行加速。主从加速并行方式中，程序的计算核心通过 Athread 或者 OpenACC 被加载到从核上进行加速计算，而主核只完成应用程序的通信、I/O 和部分串行代码的计算。

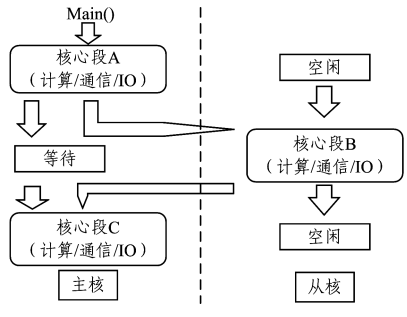

图 2 主从加速并行方式执行过程图

Fig. 2 Process of master-slave accelerated parallelism

图 2 给出了主从加速并行方式的执行过程。主核先执行

代码段 A，从核处于空等状态，然后由主核将代码段 B 加载到从核，由从核进行计算，主核此时处于空等状态，从核计算完成后返回主核，由主核再进行计算[10]。

2.3　编译、链接及运行

“神威·太湖之光”超级计算机使用神威系列编译环境，其基础编译环境包括 C、C++ 和 Fortran 编译器，针对 SW26010 异构众核处理器的主核、从核的程序编译和链接采用了同一编译命令的不同编译选项来实现[11]，具体如表 1 所列。

表 1　神威基础编译命令列表

Table 1　Command list of SW compiler

语言代码	主核代码	从核代码	混合链接
C	sw5cc-host	sw5cc-slave	sw5cc-hybrid
C++	sw5CC-host	不支持	sw5CC-hybrid
Fortran	sw5f90-host	sw5f90-slave	sw5f90-hybrid

在“神威·太湖之光”上，主从核的代码是分开进行编译的，然后再进行链接。最后将生成的可执行文件通过系统的作业管理系统进行提交，作业提交命令为 bsub。bsub 常用命令参数说明如表 2 所列。

表 2　bsub 命令常用参数说明

Table 2　Common parameters of bsub command

参数	说明
-I	提交交互式作业，使作业在作业提交窗口输出
-q	向指定的队列提交作业，必选项
-n	指定需要的所有主核数
-N	指定需要的节点个数
-np	指定每节点内使用的主核数

例如，向高速计算系统队列 q_sw_expr 提交交互式作业 myjob，该作业使用 1 个节点，4 个主核并行，作业提交命令为：

bsub-I-q q_sw_expr-N 1-np 4./myjob

作业提交成功后，将显示一行包括 jobid 的提示信息，其中包括作业 id 号，如“Job〈102〉is submitted to queue〈q_sw_expr〉”，此时 jobid 就是 102，它是全局唯一的。一旦作业提交成功，用户对作业的终止、查询等操作就可以通过这个 jobid 来实现。

3　NPB-FT 程序的流程和热点

FT 测试程序即用快速傅立叶变换求解三维偏微分方程，本文采用了 3 种数据规模（3D-32 规模、3D-512 规模和 3D-2048 规模）的 FT 程序。

3.1　程序流程分析

FT 程序的主要流程如下。

（1）首先使用伪随机数产生器用种子 314159265 产生 $2\times n_1\times n_2\times n_3$ 个 64-bit 的随机浮点数以填充复数矩阵 $\boldsymbol{U}_{j,k,l}$，其中 $0\leqslant j<n_1$，$0\leqslant k<n_2$，$0\leqslant l<n_3$；

（2）对复数矩阵 $\boldsymbol{U}_{j,k,l}$ 做 3D-FFT（快速傅立叶变换），得出结果 V，令 $a=10^{-6}$，并令 $t=1$，然后进行后续计算；

（3）$W_{j,k,l}=\mathrm{e}^{-4a\pi^2(\bar{j}^2+\bar{k}^2+\bar{l}^2)tV_{j,k,l}}$；

（4）上式中，当 $0\leqslant j<n_1/2$ 时，$\bar{j}$ 取值为 j；当 $n_1/2\leqslant j<n_1$ 时，$\bar{j}$ 取值为 $j-n_1$。参数 $\bar{k}$，$\bar{l}$ 的取值方式与 $\bar{j}$ 的取值方式相似；

（5）然后使用 3D-FFT 方法对上式作反向快速傅立叶变换，求出结果矩阵 $\boldsymbol{X}$；

（6）最后计算复数校验值 $\sum_{j=0}^{1023}X_{q,r,s}$，其中 $q=j(\mathrm{mod}\ n_1)$，$r=3j(\mathrm{mod}\ n_2)$，$s=5j(\mathrm{mod}\ n_3)$，将 t 加 1 并重复步骤（3）—步骤（5），直到 $t=N$（N 为迭代次数，本文中 N 取 1000）。

在 FT 程序中，layout_type 分为 layout_0D 和 layout_1D 两种。其中，3D-32 规模对应 layout_0D 类型，而 3D-2048 和 3D-512 对应 layout_1D 类型。程序中由三维数组组成的数据按 z 平面阵列分布，每一个处理器存储一个或多个平面阵列，而 3D-FFT 在不同维度下用多个 1D-FFT 实现。首先 x 维度上实现 1D-FFT，其次在 y 维度上实现 1D-FFT。接着 layout_0D 类型直接实现在 z 维度上的 1D-FFT，在此类型中，将数据发送到从核后，由从核完成所有的计算，然后直接将最终计算结果发回主核即可。而对于 layout_1D 类型来说，需要进行一次 transpose 操作，最后是执行 z 维度上的 1D-FFT。在 transpose 操作步骤中，需要不断地执行数据由主核发送到从核，从核计算完毕后再将中间结果发送给主核这一过程，以便后续数据发到从核上进行计算。

反向 3D-FFT 即为将正向 3D-FFT 中的 x，y，z 3 个维度的逆序实现，其结构流程图如图 3 所示。

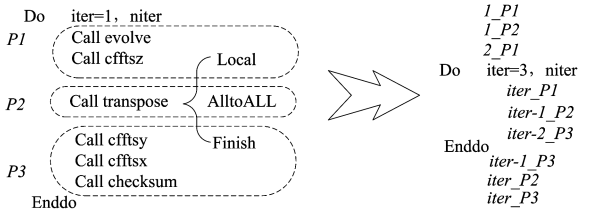

图 3　反向 3D-FFT 结构流程图

Fig. 3　Structure of negative 3D-FFT process

Transpose 操作分为 local，alltoall 和 finish 3 个过程。其中 local 为 1D-FFT 后的数组转置操作，同时在 Alltoall 通信结束后还需要继续进行转置操作，以便后续计算，此过程被称为 Finish。Local 和 Finish 的过程都是在主核上进行的，其过程就是矩阵转置，此时核组内的 64 个从核资源处于空闲状态，从核必须等主核把整个 transpose 过程执行完后才能进行后续的计算。

3.2　程序热点分析

程序的开销包括计算开销以及节点内和节点间的通信开销，节点间的通信又和超级计算机的拓扑结构有关，而节点内的通信则受内存带宽、Cache、I/O 总线的影响。

通过对 FT 程序执行过程的分析可知，程序在第 2 步需完成一次正向 3D-FFT 变换，在第 5 步需要完成 1000 次反向 3D-FFT 变换，在时间上相当于做 1001 次 3D-FFT 变换，此处即为高效实现方法中需要对程序着重优化的部分。

表 3　热点分析表

Table 3　Analysis of hot spots

函数名	所用时间百分比/%	调用次数
fftz2	54.52	923648
cffts1	11.39	44
cffts2	10.64	22
transpose2 local	6.61	22
transpose2 finish	5.99	22
evolve	3.35	20
cfftz	1.54	112640

在 3D-FFT 过程中，通过 gprof 工具分析 3D-32 规模 NPB-FT 串行程序的代码(见表 3)，发现计算子程序 fftz2 的时间占比为 54.52%，但在此执行期间调用次数较多，平均每次调用所需的时间较短，因此 fftz2 不是计算的瓶颈。通过更深层次的分析可以发现，两个计算子程序 cffts1 和 cffts2 都需要 11%左右的时间，而且产生的调用较少，即为计算密集部分，因此计算部分应尽量考虑优化 cffts1 和 cffts2 子程序。同时，在通信开销中，local 和 finish 两个数据转置的过程均需要 6%左右的时间，是通信开销部分的优化核心。

4 高效 FT 实现方法

通过对 FT 程序过程及开销热点的分析，针对优化核心代码，本节介绍了主从协同计算、主从核计算向量化、数据传输优化、计算与通信隐藏、寄存器通信及其他 FT 程序的高效实现方法。

4.1 主从协同计算

SW26010 加速线程库(Athread 库)是针对主从加速编程模型所设计的程序加速库，其目的是使用户能够方便、快捷地对核组内的线程进行控制和调度，从而更好地发挥核组内多从核执行加速性能[12]。一般来说，并发线程的应用程序无需考虑可用处理器的数量，但是在 SW26010 架构下，每个线程绑定一个从核资源，因此在使用 athread 库时需要考虑从核资源的分配状况。在调用线程库之前，必须显式检查核组内从核的可用资源状态。

主核加速线程库主要是提供主核程序使用的 athread 接口，用于控制线程的初始化、启动、结束等一系列操作。而从核加速线程库主要是提供从核程序的 athread 接口，用于从核线程的线程识别、中断发送等一系列操作。用户可以通过调用表 4 所列的接口实现相关功能[13]。

表 4 Athread 加速线程库接口

Table 4 Interface of Athread library

	函数接口	作用
主核	athread_init	加速线程库的初始化
	athread_spawn	添加新的受控线程组
	athread_join	显式阻塞调用该线程组
从核	athread_get_id	获得本地单线程的 ID 号
	athread_get_core	获得线程的物理从核号
	athread_get	从主核指定位置接收数据
	athread_put	向主核指定位置发送数据

利用 athread 编写的从核程序示例如下：

```
athread_init();
athread_enter64();
athread_spawn64(fft_init_slv,0 );
athread_join64()。
```

这个代码段表示函数 fft_init_slv 在每个从核上都进行一次执行，整个程序的数据流结构相对简单，首先利用伪随机数生成算法生成初始数据 U0，其次生成 evolve 时的数组 twiddle，然后生成傅立叶变换时的系数数组 u，这些数据均在主核上生成，通过调用 DMA 将其拷贝到从核上进行计算。这是因为主核的计算能力和从核的计算能力基本相当，所以程序的计算部分都应该尽可能地放到从核上，这样可以大大减少主核和从核之间的数据传输开销。

在将一次 fft 的过程分到从核上计算时，空间的管理是最重要的。本文的测试维度分别为 2048 和 512。当 fft 的长度为 2048 时，由于数据类型为浮点复数，因此一个数据元素的大小为 16 B。fft 的数组 u_0 大小为 2048 * 16 B/1024=32 kB，由于在进行 fft 的过程中需要开辟与 u_0 大小相一致的缓冲区 scratch，因此 scratch 的空间也为 32 kB，而 fft 过程的系数矩阵大小也为 32 kB，这 3 个数据空间需求的总和为 96 kB，已经远远超过从核局存的空间，导致一个从核不能完成一条 2048 长度数据 fft 的计算。在详细研究 fft 的过程后发现，可对 fft 的计算过程进行划分，利用两个从核共同完成一条 2048 长度数据的计算。具体计算流程如图 4 所示。

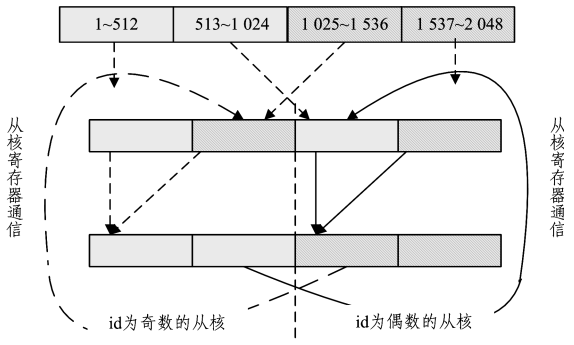

图 4 3D-2048 规模从核计算流程图

Fig. 4 Computation process by CPEs in 3D-2048

由于数据长度为 2048，因此 fft 共需要迭代 11 次。为便于后续描述，用从核 0 代替偶数核，用从核 1 代替奇数核。第一次迭代时数据需要从主核获取，在逻辑核号相邻的两个从核上进行计算，0 号核获取 1～512 和 1025～1536 这两部分数据，而 1 号核获取 513～1024 和 1537～2048 这两部分数据。完成第一次迭代计算后，0 号线程需要与 1 号线程交换数据，0 号线程利用寄存器通信将计算出来的 513～1024 这部分数据利用寄存器通信方式传输到 1 号线程的 1～512 部分，1 号线程利用从核寄存器通信将计算出来的 1～512 部分传输到 1 号线程的 513～1024 部分，经过这样的数据划分后，不仅 u_0 的数据空间减小为原来的一半，scratch 空间和系数矩阵 u 的空间也同样可以按照此方式进行划分，空间需要也变为原来的一半，因此采用 0 号线程和 1 号线程同时计算一条 fft 时总共数据需求不会超过从核的局存空间 64 kB，满足了计算的要求。

4.2 主从核计算向量化

由于原程序的编写语言为 Fortran，数据类型为浮点复数类型，而一般的编译器都不支持浮点复数类型的向量化，因此将原程序由 Fortran 程序改写为 C 程序，以便手工向量化。用 Fortran 编写的 FFT 核心代码段如下：

```
do j=1,ny
  x11=x(j,i11+k)
  x21=x(j,i12+k)
  y(j,i21+k)=x11+x21
  y(j,i22+k)=u1*(x11-x21)
enddo
```

上述代码段中的加、减、乘操作都为复数运算，都应该利

用复数运算法则进行运算，复数的加法和减法操作为对应的实部和实部进行加减、虚部和虚部进行加减，但是复数的乘法相对复杂，复数乘法的规则如下：

C. real＝a. real * b. real－a. imag * b. imag

C. imag＝a. real * b. imag＋a. imag * b. real

因此，对 fft 向量化的核心运算为复数乘法，由于复数在内存中是实部和虚部相邻而存，因此在进行复数加减法时可直接按照槽位进行加减，而不需要跨步访存指令或者混洗指令实现，但是在实现复数乘法向量化时一个思路是通过两次连续的向量 load 指令和一条混洗指令，将向量寄存器槽位内所有的数据都变为实部数据或者虚部数据，这样便于后面的计算，在写回数据时通过向量混洗指令将结果存入结果空间，这种实现方法引入了 6 条向量混洗指令，开销较大。

第二种实现向量化的方法是数据在向量寄存器内还是按照实部和虚部相邻的方式，这样可以利用更少的 shuffle 实现向量化[14]，但是参与复数乘法运算的另一个复数是循环不变量，其可以在循环外先用 loade 指令组成向量，然后在循环内作为一个不变量使用，复数乘法运算过程如图 5 所示。

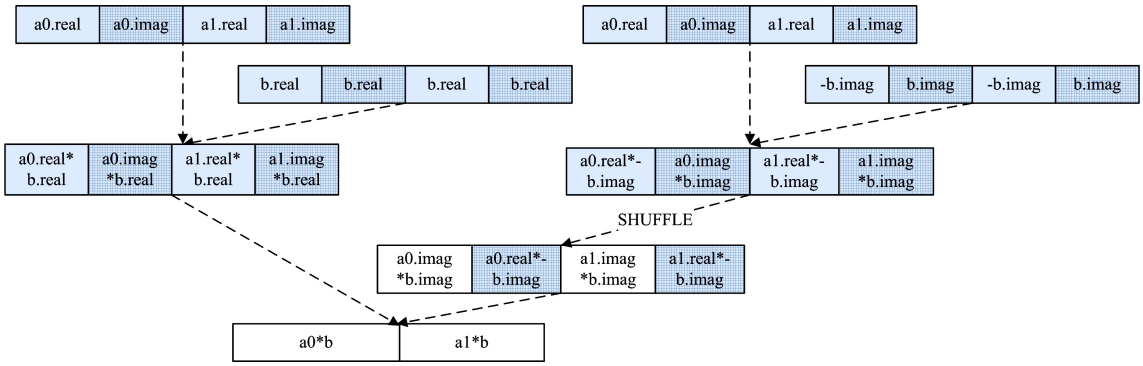

图 5　复数乘法运算过程示意图

Fig. 5　Process of complex multiplication

由于在进行 fft 时不仅要执行乘法运算，而且还需要执行加法运算，将生成的结果写入两个不同的位置，并且系数矩阵 u 为循环不变量，因此第二种实现方法的效果比第一种更好。最后生成向量化的代码片段如下：

```
for(k=0;k <lk;k++)
{
  idx1=(i11+k) * 4;
  idx2=(i12+k) * 4;
  idx3=(i21+k) * 4;
  idx4=(i22+k) * 4;
  simd_load(vx1,&x[idx1]);
  simd_load(vx2,&x[idx2]);
  vy1=vx1+vx2;
  simd_store(vy1,&y[idx3]);
  vy2=vx1-vx2;
  tp1=vu1 * vy2;
  tp2=vu2 * vy2 * vfu1;
  vy3=simd_vshff(tp2,tp2,177);
  vy3=tp1+vy3;
  simd_store(vy3,&y[idx4]);
}
```

单个计算核心内部除了采用向量执行计算外，还采用了循环展开策略，循环展开通过多次复制循环体代码来实现，它能够增大指令调度的空间，减少循环分支指令的开销，是一种加快程序的执行速度的优化方法[15]。展开过度或展开非常大的循环时，可能导致代码篇幅增加。如果展开后的循环不能再放入缓存，这不仅不能带来性能的提升，反而会降低程序的执行速度，因此需要选择合适的循环展开因子，循环展开因子一般为 2 的整数次幂，经过反复测试发现本程序循环展开因子为 2 时可以获得最好的程序性能。

4.3　数据传输优化

根据 3.1 节对程序的分析可知，程序完成 Z 方向的 FFT 变换后需要进行转置操作，此过程被称为 Local。同时在 Alltoall 通信结束后还需要继续进行转置操作，以方便后续的计算，此过程被称为 Finish。Local 和 Finish 的过程都是在主核上进行的，其过程就是矩阵转置，此时核组内的 64 个从核资源处于空闲状态，从核必须等主核把整个 transpose 过程执行完后才能进行后续的计算。

为了避免主核在进行数组转置操作时从核处于空闲状态，本文提出了数据传输优化。具体地，在从核上完成 Z 方向的 FFT 变换后，将计算结果回传给主核的同时，按照矩阵转置后的状态写回主核内存，消除主核上的 Local 过程。同时，在计算 Y 方向的 FFT 变换前，将数据不连续地发送到从核上，同样为了完成矩阵转置的过程，消除了主核上的 Finish 过程，整个示例如图 6 所示。

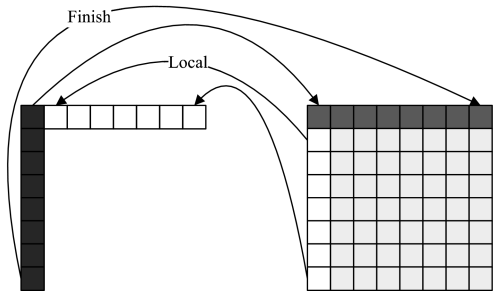

图 6　利用跨步传输消除主核上的矩阵转置

Fig. 6　Elimination of matrix transposition on MPEs using stride transmission

图 6 给出了利用跨步数据传输消除主核上的矩阵转置过程。如果每计算完一条数据后就将结果发回到主核上，那么每存储一个数据，就跨过一定的步长再存储第二个数据，此种

方式开销较大，为此通过寄存器通信将目标区域连续的数据合并到一定长度后，再统一写回到主存中，这样跨步虽然没变，但是一次存入的数据增多，降低了程序的执行开销。理论上，连续数据的长度越长性能就越好，但是如果目标连续的数据通过寄存器通信方式都变得连续，则寄存器通信的压力会变大。经过多次测试后可知，当连续数据的长度为 8 时程序的执行开销最小，这是由于当数据长度为 8 时，寄存器仅需同行或者同列进行一次通信就可完成数据的组合，如果长度多于 8 个，就需要二次通信才能将数据从源从核传输到目标从核。

4.4 计算与通信隐藏

以 3D-512 规模为例说明计算与通信隐藏的问题，经过其他手段进行优化后，程序总共的运行时间为 42 s，其中从核计算加上主、从核之间数据传输的时间共约为 12 s，而核组间通信时间约为 30 s，程序的运行结构流程图如图 7 所示。

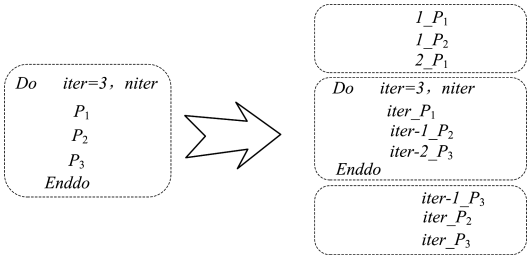

图 7 计算与通信隐藏示意图

Fig. 7 Diagram of computation and communication hiding

整个程序的核心部分是一个循环，迭代次数为 niter，循环内依次调用函数 evolve，cfftsz，transpose，cfftsy，cfftsx 和 checksum，其中 evolve 表示数据更新的过程，cfftsz，cfftsy 和 cfftsx 分别表示 Z 方向、Y 方向和 X 方向的傅立叶变换，checksum 表示对运算结果进行校验。transpose 又分为 Local，AlltoALL 和 Finish 3 个步骤，AlltoALL 表示核组间的通信，Local 和 Finish 表示核组内的数据转置。可以将程序的结构流程分为 3 个部分，P_1 包含 evolve，cfftsz 和 Local 3 个部分，P_2 仅包含 AlltoALL 部分，P_3 包含 Finish，cfftsy，cfftsx 和 checksum 4 个部分。P_1 和 P_3 部分利用核组内的从核计算资源和主从核之间的数据传输通道，P_2 部分仅用核组间的通信部件，其中 P_1 和 P_3 部分所占的时间共约 12 s，而 P_2 部分所占的时间约为 30 s。

在未做计算与通信隐藏前，P_1，P_2 和 P_3 这 3 个部分顺序执行，也是就说只有执行完第 i 次迭代 P_1 部分，然后才执行第 i 次迭代的 P_2 部分，最后执行完第 i 部分的 P_3 部分。而在执行第 i 次迭代 P_1 部分和 P_3 部分时，使用了核组内的计算和数据传输通道，核组间的通信部件处于空闲状态。同时，在执行第 i 次迭代的 P_2 部分时，使用了核组间的通信部件，而核组内的计算和数据传输通道处于空闲状态，计算与通信的隐藏就是为了解决这个问题。具体地，计算与通信隐藏优化将程序结构分为预处理、循环主体和后处理 3 个阶段，在预处理阶段完成第 1 次迭代的 P_1 和 P_2 的执行以及第 2 次迭代的 P_1 的执行，这样进入到循环主体后进行第 iter-2 迭代的 P_3 部分的执行、第 iter-1 次迭代的 P_2 部分的执行以及第 iter 次迭代的 P_1 部分的执行，这样就形成了流水操作，实现了计算和通信同时执行，避免了核组内计算和数据传输通道空闲或者核组间通信部件的空闲。

如果将计算与通信进行流水并行，理论上程序的运行时间会缩短为 30 s，即计算占用的时间 12 s 被通信占用的时间 30 s 完全隐藏，但是由于计算结束后需要往主核主存写入数据，而通信过程结束后也需要往主核的主存写入数据，它们之间会竞争存储控制总线，因此计算与通信隐藏后实际的运行时间比理论时间长 30 s。最后经过计算与通信隐藏优化后，程序执行时间为 37 s。

经过上述优化后，程序执行时间主要花费在核组间的 AlltoALL 通信上，即制约程序性能提升的主要瓶颈为通信，为此我们提出了两个优化策略。第一个策略是开启三级并行，因为 SW26010 处理器的一个 CPU 内包含 4 个核组，4 个核组间使用 OpenMP 级并行，这样相当于将原来的 64 个进程间的 AlltoALL 通信分为两级，一个是 16 个进程间的 AlltoALL 通信，另一个是在 CPU 内按照共享内存的方式共享数据。第二个策略仍然采用 64 个进程，但是每个 CPU 上的 4 个进程选出一个代表，这样先进行 16 个进程间的通信，相当于每个 CPU 上选出一个代表进程，然后进行 CPU 内部 4 个进程间的 AlltoALL 通信，经尝试后可知这两种方法都没有性能提升，主要是因为 AlltoALL 通信时通信消息太大，这种采用两级进行 AlltoALL 通信的思路不仅不能获得性能的提升，反而降低了程序的性能。

4.5 寄存器通信

从核和从核之间可以通过寄存器实现数据的交换。由于从核在物理上是以 8 * 8 的阵列排布的，因此寄存器通信仅限于同行或者同列的从核之间进行通信，如果是在不同行且不同列的从核之间交换数据，则需要进行两次通信才能完成数据的交换[16]。寄存器通信的方式如图 8 所示。

从核通过 PUT 函数可以同行或同列广播，也可以指定某一从核号进行数据发送，当然也仅限于同行或同列的从核。此外，寄存器通信一次传输的数据为 256 bit，在传输较大的数据时可以使用循环传递，但是这样会占用很多寄存器。因此使用从核寄存器时需要做一定的权衡，否则会适得其反。另外，在使用寄存器通信时，我们还需要选择合适的通信方式，避免同一时刻多个从核与同一个从核进行数据交换，这样会造成拥塞，因为寄存器通信是阻塞式的，所以一旦造成拥塞，势必会引起整个程序的低效执行。

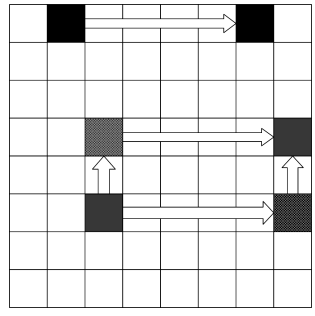

图 8 寄存器通信方式示意图

Fig. 8 Structure of register communication

在 3 个算例的运算过程中，多次用到了寄存器通信实现

数据在从核之间的高效通信。如果不使用从核之间的寄存器通信，那么就需要从核先将数据传给主核，再将数据由主核发回到从核上，由于从核和主核之间数据传输的开销较大，因此如果不利用从核之间的寄存器通信，那么从核之间数据交换的开销将非常大。在对 FFT 程序的 3 个规模算例进行计算时，使用到了寄存器通信优化。首先 3D-2048 规模在进行傅立叶变换时由于局存的空间有限，需要两个从核运算同一条数据；其次，在数据转置过程优化也就是数据传输优化过程中，同样用到了寄存器通信优化；然后，在计算完 Z 方向的傅立叶变换后，不需要将数据传回到主核上，直接进行后续的 Y 方向和 X 方向的傅立叶变换，此时从核间的数据转换利用从核寄存器通信即可实现。以计算为例，详细说明寄存器通信的流程。

4.6　其他优化

其他优化包括指令流水优化、数据对齐优化等，除此之外还做了内联替换、常数传播等优化。

(1)指令流水优化。处理器内含有两个指令发射通道，每个发射通道上能够发射的指令种类是不一致的，简单举例如流水线 1 可以发射加、减、乘、除等浮点运行，而流水线 2 可以发射加、减等整型运算，因此一组指令内应该最好既有浮点运算又有整型运算，而不应该仅有浮点计算或者仅有整型运算，这样才能将两条指令发射通道都充分利用起来，理论上可以将程序的性能提升 2 倍，但是实际上往往达不到这样的效果，其原因包括两个方面：1)程序中浮点运算和整型运算的指令数可能并不相等，也就是不能保证在每发射一条浮点运算指令的同时发射一条整型运算指令；2)在重新对指令进行排序时，为了满足正确性要求，还需要指令间的相互依赖关系。以上两个原因也是在对指令进行重排时要考虑的两个关键要素，除此之外还应该考虑每个指令所占的节拍数。指令流水优化需要在汇编代码上进行，不仅需要程序员了解程序的依赖关系，还需要熟悉底层指令集体系结构的相关知识，因此实施指令流水优化的难度较大，FFT 程序指令流水获得了 1.26 倍的加速效果。

(2)数据对齐优化。当且仅当 $A \bmod n=0$ 时，内存访问为对齐访问，其中 A 为内存地址，n 为访存数据的字节数。数据访问的对齐实质是指被访问数据与硬件部件的对齐。指令集希望这些数据的访问是对齐的，这样它们就可以快速读取这些数据，不对齐的数据访问需要额外的开销。为保证数据在引用时是对齐的，可以在数组声明时加上对齐指示关键字。

5　性能测试与分析

本次实验将从 3D-32 规模、3D-512 规模和 3D-2048 规模对 NPB-FT 程序进行优化，不同的规模考查了 FT 程序的不同方面，其中 3D-32 规模算例在单核组上执行，而 3D-512 规模和 3D-2048 规模算例分别在 64 和 256 个核组上执行。

不同规模的优化效果如图 9 所示。

3D-32 规模中，将数据发送到从核后，从核完成所有的计算然后直接将最终计算结果发回主核即可，因此相对于 3D-512 和 3D-2048 两个规模，3D-32 规模不需要进行数据传输优化，同时由于不存在核组之间的通信，计算与通信隐藏优化方法也不需要。图 9 中，在利用从核计算、主从核计算向量化、从核寄存器通信及其他优化策略对 FT 程序进行优化后，性能可提升 66.90 倍。

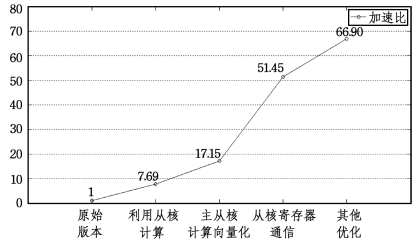

图 9　3D-32 规模下 FT 程序的优化效果

Fig. 9　Optimization result of FT program under scale of 3D-32

而在 3D-512 和 3D-2048 两个测试规模中，数据需要被不断地发送到从核，从核计算完毕后再将中间结果发送给主核，以便后续数据发到从核上进行计算，因此会通过 transpose 函数进行矩阵转置操作，故需要用到数据传输优化及计算与通信隐藏优化方法。

3D-512 规模和 3D-2048 规模下 FT 程序的优化效果如图 10、图 11 所示。从图 10 和图 11 可以看出，在利用从核计算、主从核计算向量化、数据传输优化、计算与通信隐藏、从核寄存器通信及其他优化策略对 FT 程序进行优化后，其性能可分别提升 20.00 倍和 46.58 倍。同时，由图中的加速比增长趋势可以看出，由于 3D-2048 规模核组间的通信次数更多，通信时间更长，因此相较于 3D-512 规模，其数据传输优化方法和计算与通信隐藏优化策略有着更为显著的优化效果。

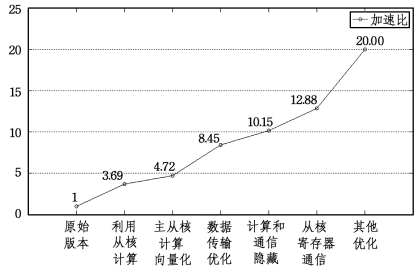

图 10　3D-512 规模下 FT 程序的优化效果

Fig. 10　Optimization result of FT program under scale of 3D-512

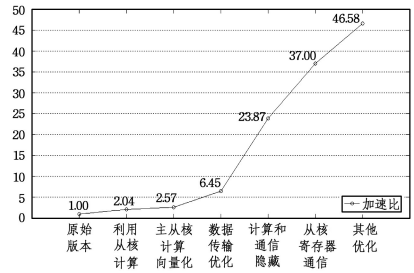

图 11　3D-2048 规模下 FT 程序的优化效果

Fig. 11　Optimization result of FT program under scale of 3D-2048

理论上，利用从核计算的加速比应为 64 倍，但实际上，任务的切分、依赖的消除需要时间；另外，由于从核所需要的计算数据需要从主核上获取，因此也需要通信时间；由于从核局部存储空间的限制，使得从核无法从局部存储空间获取计算

所需要的全部数据，导致有些数据需要通过直接访问主核内存来获取，这大大增加了计算时间，以至于无法获得理论加速比。同时，本次测试中主从核计算向量化的理论加速比应为4倍，但实际上向量化中 load，store 等操作会延长执行时间，使得主从核计算向量化同样无法得到理论上的加速比。

结束语 本文针对国产自主研发的 SW26010 处理器的特殊架构，通过重写 NPB-FT 程序并利用 athread 加速线程库，使其能够更好地运行在“神威·太湖之光”超级计算机上，充分利用 SW26010 处理器的从核资源，从而获得较好的性能；同时，针对 NPT-FT 程序并结合 SW26010 处理器的硬件特性进行了一系列的优化，极大地提高了 NPB-FT 程序在神威平台上的运行速度，其中 3D-32 规模的速度提升了 66 倍，3D-512 规模的速度提升了 20 倍，3D-2048 规模的速度提升了 46 倍左右。但是，在多节点执行过程中，依然存在通信时间冗长的问题。下一步工作是优化核组间的通信以及负载均衡策略，以进一步提升 NPB-FT 程序在“神威·太湖之光”上的性能。

参 考 文 献

[1] DONGARRA J. Report on the Sunway Taihu Light System: UIEECS-16-742[R]. Knoxville: University of Tennessee, 2016.

[2] HONG W J, LI K L, QUAN Z, et al. PETSc's Heterogeneous Parallel Algorithm Design and Performance Optimization on the Sunway TaihuLight System[J]. Chinese Journal of Computers, 2017, 40(9): 2057-2069. (in Chinese)
洪文杰，李肯立，全哲，等. 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[J]. 计算机学报，2017，40(9)：2057-2069.

[3] YUAN W, ZHANG Y Q, SUN J C, et al. Perfomance Analysis of NPB Benchmark on Domestic Tera-Scale Cluster Systems[J]. Journal of Computer Research and Development, 2005, 42(6): 1079-1084. (in Chinese)
袁伟，张云泉，孙家昶，等. 国产万亿次机群系统 NPB 性能测试分析[J]. 计算机研究与发展，2005，42(6)：1079-1084.

[4] FANG W, SUN G Z, WU C, et al. A Parallel Algorithm of Three-Dimensional Fast Fourier Transform [J]. Journal of Computer Research and Development, 2011, 48(3): 440-446. (in Chinese)
方维，孙广中，吴超，等. 一种三维快速傅里叶变换并行算法[J]. 计算机研究与发展，2011，48(3)：440-446.

[5] WU Y W. Research on Parallel Computing Model for CPU/GPU Heterogeneous System[D]. Changsha: National University of Defense Technology, 2012. (in Chinese)
吴勇文. CPU/GPU 异构集群并行计算模型研究[D]. 长沙：国防科学技术大学，2012.

[6] CHAO Y. Peta-scale fully-implicit solver for nonhydrostatic atmospheric dynamics with 8.5M Cores[C]// Proc. of SC'16, 2016.

[7] ZHENG F, XU Y, LI H L, et al. A homegrown many-core processor architecture for high-performance computing[J]. SCIENTIA SINICA Informations, 2015, 45(4): 523-534. (in Chinese)
郑方，许勇，李宏亮，等. 一种面向高性能计算的自主众核处理器结构[J]. 中国科学：信息科学，2015，45(4)：523-534.

[8] DONGARRA J. Sunway Taihu Light super-computer makes its appearance[J]. National Science Review, 2016, 3(3): 265-266.

[9] YAO W J, CHEN J S, SU Z C, et al. Porting and optimizing of NAMD on Sunway Tai huLight System[J]. Computer Engineering & Science, 2017, 39(6): 1022-1030. (in Chinese)
姚文军，陈俊仕，苏志超，等. 基于神威太湖之光的 NAMD 软件的移植与优化[J]. 计算机工程与科学，2017，39(6)：1022-1030.

[10] FU H H, LIAO J F, YANG J Z, et al. The Sunway TaihuLight supercomputer: system and applications[J]. Science China Information Sciences, 2016, 59(7): 072001: 1-072001: 16.

[11] YAO W J. Implementation and Optimization of Molecular Dynamics Application on Sunway TaihuLight Supercomputer[D]. Hefei: University of Science and Technology of China, 2017. (in Chinese)
姚文军. 神威·太湖之光上分子动力学软件的实现与优化[D]. 合肥：中国科学技术大学，2017.

[12] ZHAO M T, LIU Y, LIU R, et al. Acceleration of histogram of oriented gradient (HOG) based on Sunway many-core processor [J]. Computer Engineering & Science, 2017, 39(4): 611-618. (in Chinese)
赵美婷，刘轶，刘锐，等. 基于申威众核处理器的 HOG 特征提取算法并行加速[J]. 计算机工程与科学，2017，39(4)：611-618.

[13] WU M C, HUANG L, LIU Y, et al. An OpenCL Compiler for the Homegrown Heterogeneous Many-cor Processor on the Sunway TaihuLight Supercomputer [J]. Chinese Journal of Computers, 2018, 41(10): 2236-2250. (in Chinese)
伍明川，黄磊，刘颖，等. 面向神威·太湖之光的国产异构众核处理器 OpenCL 编译系统[J]. 计算机学报，2018，41(10)：2236-2250.

[14] SCHLEGEL B, GEMULLA R, LEHNER W. Fast integer compression using SIMD instructions[C]// International Workshop on Data Management on New Hardware. ACM, 2010: 34-40.

[15] STOJANOV A, TOSKOV I, ROMPF T, et al. SIMD intrinsics on managed language runtimes[C]// International Symposium. 2018: 2-15.

[16] MENG D L, WEN M H, WEI J W, et al. Porting and Optimizing OpenFOAM on Sunway TaihuLight System[J]. Computer Science, 2017, 44(10): 64-70. (in Chinese)
孟德龙，文敏华，韦建文，林新华. 神威太湖之光上 OpenFOAM 的移植与优化[J]. 计算机科学，2017，44(10)：64-70.