Parallel Optimization Methods for Direct Numerical Simulation of High Reynolds Number Wall Turbulence with a Grid Size of 100 Billion

Jiabin Xie
xiejb6@mail2.sysu.edu.cn
Sun Yat-sen University
Guangzhou, Guangdong, China

Guangnan Feng
fenggn3@mail2.sysu.edu.cn
Sun Yat-sen University
Guangzhou, Guangdong, China

Han Huang
huangh367@mail2.sysu.edu.cn
Sun Yat-sen University
Guangzhou, Guangdong, China

Junxuan Feng
junxuan.feng@nscc-gz.cn
Sun Yat-sen University
Guangzhou, Guangdong, China

Yutong Lu*
yutong.lu@nscc-gz.cn
Sun Yat-sen University
Guangzhou, Guangdong, China

Abstract

Direct numerical simulation (DNS) is a technique that directly solves the fluid Navier-Stokes equations with high spatial and temporal resolutions. However, its utility in studying high Reynolds number (*Re*) wall turbulence of particular interest is limited by the rapidly growing grid size (i.e., the memory and computation requirement) with Re^3.

We present PowerLLEL, a high-performance finite difference solver tailored for the challenging DNS of incompressible wall turbulence at extreme scales. An adaptive multi-level parallelization strategy is proposed to fully exploit the multi-level parallelism of various architectures and enhance computational performance. The communication performance of global transpose and halo exchange is significantly improved by a tridiagonal solver based on the parallel diagonal dominant (PDD) algorithm and three RDMA-implemented communication optimizations.

Strong scaling tests on the Tianhe-2A supercomputer show that PowerLLEL achieves nearly 92% parallel efficiency with up to 31,104 cores on a grid size of 143.3 billion.

CCS Concepts: • **Computing methodologies** → **Massively parallel and high-performance simulations**; • **Applied computing** → *Physics*.

Keywords: incompressible turbulence, direct numerical simulation, high performance computing

1 Multi-level Parallelization Strategy

Through reasonable abstraction and induction, we find that different system architectures have similar hierarchical architectures (Device-Processor-Compute Units), which makes it possible for us to propose an adaptive multi-level parallelization strategy. We propose a three-level parallelization strategy: Process->Thread->SIMD, corresponding to Pencil->Slab/Sub-pencil->X-segment, combined with an adaptive and load-balanced two-dimensional (2D) pencil domain decomposition. It can fully exploit the computing power of various architectures, including CPU, GPU, and heterogeneous many-core processors (for example MT-3000[2]).

2 PDD-based Tridiagonal Solver

In PowerLLEL, the tridiagonal solver is called multiple times to solve a series of tridiagonal systems of equations only coupled in the wall-normal (z) direction. The Poisson solver calls the tridiagonal solver only once, while the PSIP (partially semi-implicit time integration) step calls up to 6 times. However, in the "x-pencil" decomposition, the tridiagonal solver based on the conventional Thomas algorithm requires up to $2\times(1+6) = 14$ global transposes in a single time step. It leads to a large number of data packing/unpacking and all-to-all communications which hurts the overall computational performance and scalability.

To this end, we develop a parallel tridiagonal solver based on the parallel diagonal dominant (PDD) algorithm [3], which only needs two neighboring Sendrecv communications, to complete the solving locally in the current decomposition. Thus, the global transposes induced by the tridiagonal solver are completely avoided. With this optimization, we reduce the number of global transposes in the Poisson solver to two, which are only caused by FFT, while the PSIP no longer requires global transposes. This means that the number of global transposes in a time step is reduced from 16 to only 2 (FFT-induced only), which is a significant improvement in communication performance and scalability.

3 RDMA-based Communication Optimizations

The communication overhead of the PowerLLEL mainly comes from all-to-all communications and halo exchange. We propose three different RDMA-based optimizations, including pipeline overlap, round-robin transfer, and non-sync

*Corresponding author

RDMA. In pipeline overlap, the x-FFT and y-FFT computation are overlapped with the all-to-all computation. Once a slab of data finishes FFT computation, it posts all-to-all asynchronous send directly without waiting for FFT computation of all the data. Once a slab of data is received, it starts FFT computation directly without waiting for all the data. In round-robin transfer, data is sliced and sent to all the neighboring processes in a round-robin manner in halo exchange to prevent head-of-line blocking. As for non-sync RDMA, all the data in all-to-all communications and halo exchange are sent directly without an additional synchronization or memory copy because all the receive buffer is guaranteed to be ready. These RDMA-based optimizations remove all the synchronization costs in all-to-all communications and halo exchange, avoid potential congestion in the network, and hide the communication cost by computation.

4 Performance Evaluation

The performance evaluation is performed on the Tianhe-2A supercomputer[1]. Figure 1 shows the changes in the wall-clock time per time step and the corresponding speedup when the above optimizations are applied in turn. The baseline is the naive implementation with MPI only. Firstly, we use OpenMP to fully exploit the multi-level parallelism. The solver achieves nearly 2x speedup when using 18,432 cores. At this point, the solver bottleneck is no longer computation, but communication. Therefore, we adopted the PDD-based tridiagonal solver and RDMA-based communication optimizations to reduce the communication overhead as much as possible, and the solver finally obtain up to 10.6x speedup when using 18,432 cores.

Results of strong scaling tests are shown in Figure 2. PowerLLEL achieves approximately 95% parallel efficiency when scaling from 2,304 cores to 18,432 cores, with a grid size of 16.3 billion. It also exhibits linear strong scaling performance up to 31,104 cores with nearly 92% parallel efficiency, on a larger grid with 143.3 billion points. From the timing breakdown, we can further analyze the scalability of different computation/communication tasks.

Weak scalability reflects the computational efficiency of a parallel program when computational resources grow with the problem size. The results of weak scaling tests are shown in Figure 3. When the number of cores and the grid size are increased by 16 times (from 1,152 to 18,432 and from 3.1 billion to 48.9 billion), the wall-clock time per time step of PowerLLEL increases by about 20%, and the parallel efficiency is about 83%.

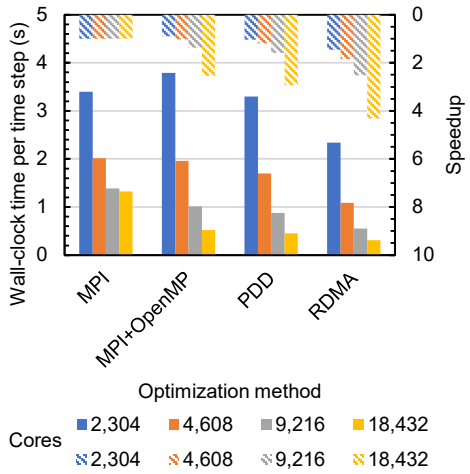

Figure 1. Speedup of the proposed optimization methods.

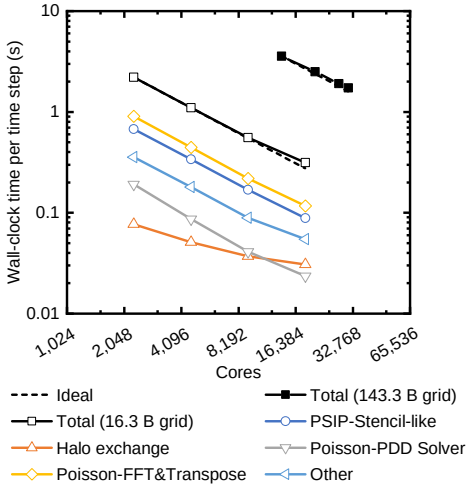

Figure 2. Strong scaling test results.

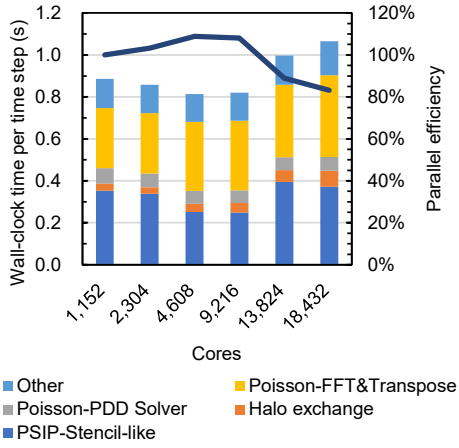

Figure 3. Weak scaling test results (2.66 M grid points per core).

References

[1] Xiangke Liao, Liquan Xiao, Canqun Yang, and Yutong Lu. 2014. MilkyWay-2 supercomputer: system and application. *Frontiers of Computer Science* 8, 3 (June 2014), 345–356. https://doi.org/10.1007/s11704-014-3501-3

[2] Kai Lu, Yaohua Wang, Yang Guo, Chun Huang, Sheng Liu, Ruibo Wang, Jianbin Fang, Tao Tang, Zhaoyun Chen, Biwei Liu, Zhong Liu, Yuanwu Lei, and Haiyan Sun. 2022. MT-3000: a heterogeneous multi-zone processor for HPC. *CCF Transactions on High Performance Computing* 4, 2 (June 2022), 150–164. https://doi.org/10.1007/s42514-022-00095-y

[3] Xian-He Sun, Hong Zhang Sun, and Lionel M. Ni. 1989. Parallel algorithms for solution of tridiagonal systems on multicomputers. Association for Computing Machinery, Crete, Greece, 303–312. https://doi.org/10.1145/318789.318822