SCALING THE BATS–R–US MHD MODEL TO OVER 100,000 CORES WITH EFFICIENT HYBRID OPENMP AND MPI PARALLELIZATION

Allocation: GLCPC/360 Knh
PI: Gabor Toth[1]
Collaborator: Hongyang Zhou[1]

[1]University of Michigan

EXECUTIVE SUMMARY

This project aims to optimize and improve multilevel parallelization of the computationally most expensive components of the space weather modeling framework. One of the most important and computationally expensive models in the framework is the BATS–R–US magnetohydrodynamic (MHD) code. With pure MPI parallelization it is limited to about 32,000 cores owing to memory constraints. The research team has designed and implemented an efficient hybrid MPI + OpenMP parallelization. The main idea is to assign grid blocks consisting of hundreds of grid cells to each OpenMP thread. This is much easier to implement than a cell-by-cell multithreading approach, and it is also more efficient. The new version of the code can scale to over 100,000 cores on Blue Waters while maintaining high efficiency.

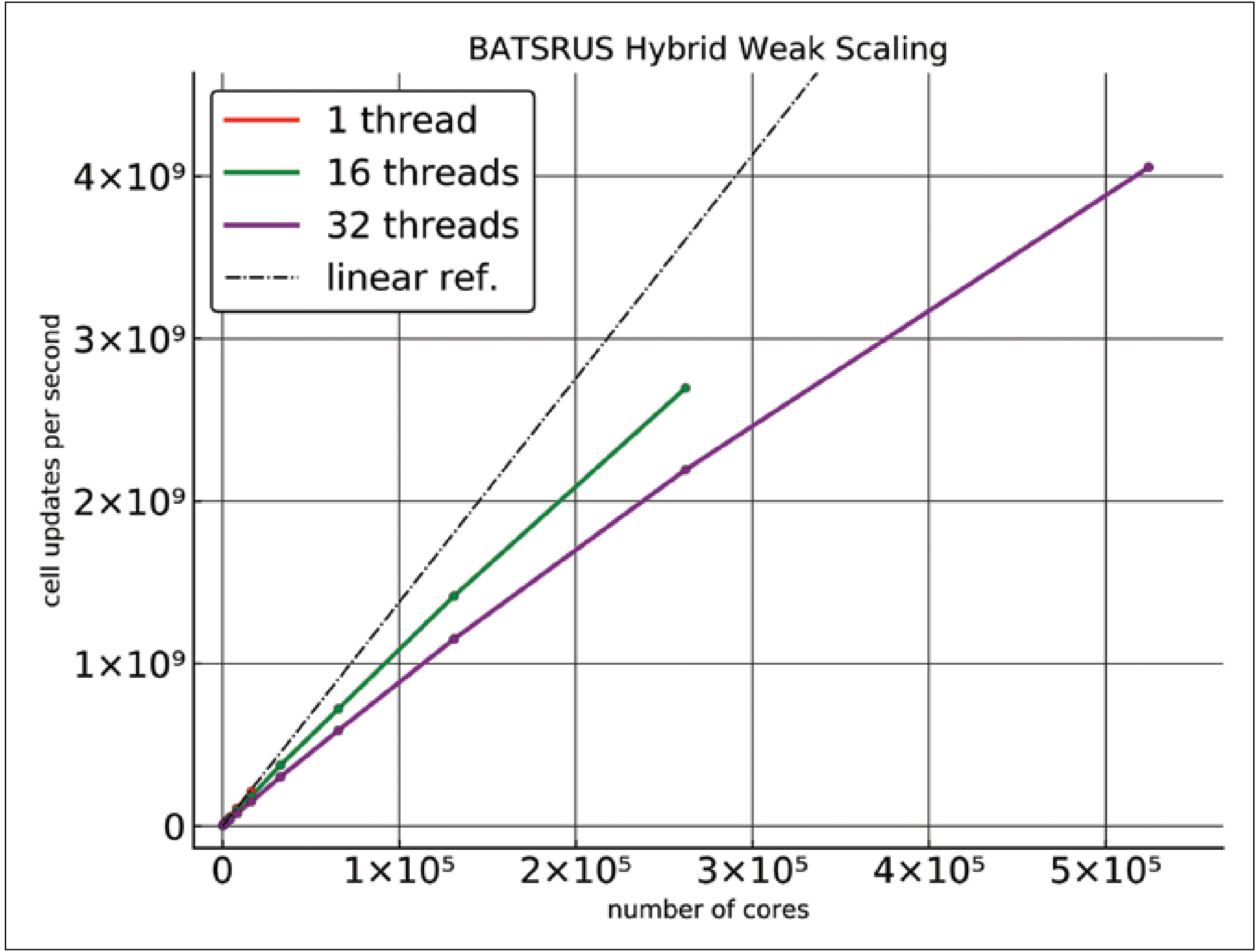

Figure 1: Weak scaling of BATS–R–US on Blue Waters. A 3D MHD problem is solved on a grid consisting of 131,072 grid cells per core using 8 x 8 x 8-cell blocks. There are 256 to 8,192 grid blocks per MPI process depending on the number of OpenMP threads. The dashed line indicates ideal linear scaling.

RESEARCH CHALLENGE

While pure MPI parallelization can be near optimal in terms of speed, the total number of cores used can be limited by the memory usage. In our BATS–R–US MHD code [1], for example, most but not all of the data are distributed over the processors. In particular, the description of the adaptive tree of the grid blocks is stored on all the MPI processes so that finding neighbor blocks or finding the grid block covering some spatial location can be done efficiently. The size of this array grows with the problem size and eventually can become a bottleneck. This issue can be mitigated by using OpenMP parallelization because the amount of memory occupied by the repeated arrays on a given node will be reduced proportionally with the number of MPI processes per node. Correspondingly, one can scale the model to a larger number of cores. With dozens of cores per node on current architectures this difference is significant, by a factor of 15 to 30 or even more.

Adding OpenMP into a complex MPI parallel code is not simple. BATS–R–US consists of about 250,000 lines of Fortran 90+ code without counting comments and empty lines. The workload is distributed over a large fraction of the source code. The goal is to add OpenMP parallelization that obtains good performance with a reasonable development time investment.

The team's primary goal is to allow scaling BATS–R–US to hundreds of thousands of cores on new computers to solve large problems efficiently. Further, the researchers' work and experience should help in making similar projects run smoothly and efficiently.

METHODS & CODES

The researchers' multiphysics code BATS–R–US can solve various partial differential equations with a large variety of numerical schemes on a block adaptive grid. The grid blocks consist of a fixed number of grid cells, typically 4 x 4 x 4 to 8 x 8 x 8 cells, although their physical size may vary. The code loops over grid blocks and performs various computations such as calculating fluxes and sources for each grid cell and then updates the grid cell value, etc. There are two possible implementation strategies for OpenMP: (1) parallelize the loops over grid cells (fine-grained) or (2) parallelize loops over grid blocks (coarse-grained).

The team chose the coarse-grained option for several reasons. In the overall algorithm, the loops over the cells contain less work than the loops over the blocks. There are many more loops over cells than loops over blocks, so a fine-grained approach would require adding more OpenMP parallel sections. For many loops over the grid cells the work per cell may be insufficient to make the OpenMP parallelization efficient given the overhead of starting and closing the multithreaded section. A significant difficulty with the coarse-grained approach is finding and resolving race conditions. It took significant effort for the team to find the appropriate tool: Intel's Inspector turned out to be invaluable.

RESULTS & IMPACT

The team's strategy required relatively modest code changes: only 609 OpenMP directive lines were added to the 250,000-line source code. Most of the changes were declaring variables as thread private or moving module variables into subroutines when convenient. The team also learned to look out for variable initializations, which make local subroutine variables behave as shared by default.

Fig. 1 shows that by using the OpenMP + MPI parallelization, BATS–R–US can run on more than 500,000 cores. Running 32 threads per node requires communication between the two sockets; still, the performance is about 50% of the ideal scaling, which is quite reasonable. With 16 threads per node BATS–R–US can scale up to approximately 250,000 cores and obtain around 80% of the ideal performance. In comparison, the pure MPI parallelization can run only up to about 16,000 cores before running out of memory. The researchers obtained similar results when running the code with an implicit solver.

The team also learned that for some compilers, switching on the OpenMP library can severely impact code performance (a slowdown of up to a factor of three) even if only one thread is used per MPI process. It is important to make sure that the code is portable and performs well for a compiler that runs efficiently with the OpenMP library. The researchers also found that the pinning of the OpenMP threads (assigning them to the proper CPU cores with respect to the MPI processes) can be quite complicated and the proper settings vary from platform to platform and even from compiler to compiler. A simple C code reporting of which core is used by a certain thread and MPI process is invaluable to verify that the pinning works as expected.

WHY BLUE WATERS

Blue Waters provided a platform with the appropriate hardware, software, and computing environment to make good progress with this project. On most systems it is practically impossible to run on more than about 10,000 cores with reasonable turnaround times. On Blue Waters, the team could scale up to about 500,000 cores. The variety of compilers allowed for testing the efficiency of the hybrid parallelization comprehensively and identifying some of the not-so-well-known problems.

PUBLICATIONS & DATA SETS

H. Zhou and G. Toth, "Efficient OpenMP parallelization to a complex MPI parallel magnetohydrodynamics code," *J. Parallel Distributed Comput.*, vol. 139, pp. 65–74, May 2020.