25C03

Hierarchical Domain Decompositions for Particle-in-Cell Codes

Viktor K. Decyk and Tajendra V. Singh

Institute for Digital Research and Education
University of California, Los Angeles
Los Angeles, California, 90095, USA

The computational nodes of future supercomputers are becoming more and more powerful as the number of cores increases. To test the idea that such a hierarchy of computers might best be programmed with a hierarchy of domain decompositions, we have implemented a two level domain decomposition using OpenMP and MPI. Results show that for Particle-in-Cell codes with global field solves, such a hierarchy may work better than MPI alone.

1. Introduction

The most successful approach for parallelizing Particle-in-Cell (PIC) codes on large parallel computers is domain decomposition with Message-Passing (MPI). It is becoming evident that each node on a traditional parallel computer is becoming more powerful, with an increasing number of cores that share memory. In fact, the future computer is tending toward a hierarchy of parallel computers clustered together. Is a hierarchy of domain decompositions a useful approach to programming such systems?

Domain decomposition solves one important difficulty that arises in shared memory computers: the problem of data collisions, when two threads or processors attempt to update the same memory location simultaneously. In PIC codes this occurs primarily during the deposit step. The traditional approach to avoiding such problems on shared memory computers is to lock memory in some fashion. Such locks can be very slow, however, and few computer languages have native support for memory locks. On distributed memory computers, guard cells or ghost cells are used to avoid data collisions, and this technique can be used on shared memory computers as well, at the cost of using extra memory. Domain decomposition on shared memory machines can also improve memory performance, because it leads to better data locality, that is, the data needed by each thread is stored compactly together.

2. MPI/OpenMP domain decompositions

To evaluate new algorithms, we have implemented a number of 2D parallel skeleton PIC codes as part of the UPIC Framework [1]. In these codes, the physics procedures and the communication procedures which manage the data movement are separated. This is sometimes called Bulk Synchronous Programming (BSP). The physics procedures work only on local data. There are only 4 communications procedures that move data between domains: one to add/copy guard cells, another to transpose data, a particle manager to move particles between domains, and a field manager to move field data between domains.

To test this idea of hierarchical domain decompositions, we have implemented a two level domain hierarchy for PIC Codes. Each node uses a local domain decomposition implemented in OpenMP. These nested domains are then connected with each other using MPI. The physics procedures are written in OpenMP. The 4 communication procedures are written in both MPI and OpenMP. MPI part uses traditional message-passing to move data between local domain decompositions, and each node must be aware of what data it is sending and receiving. The OpenMP part uses simpler algorithms to move data within its local domain decomposition because it can safely read another thread's data directly. When data movement involves both MPI and OpenMP, the data is moved in stages.

3. Performance Results

Results of a 2D PIC code shown in Figure 1 indicate that on a single shared memory node, MPI is slightly faster than OpenMP.

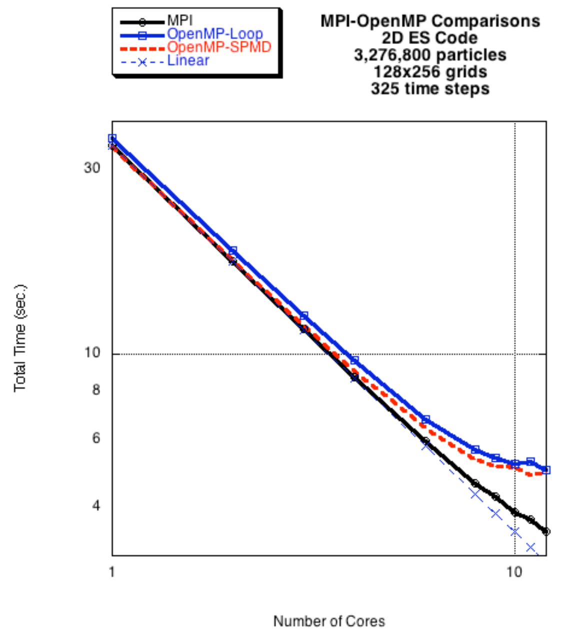

Figure 1. PIC Performance with MPI and mixed MPI/OpenMP on one shared memory node

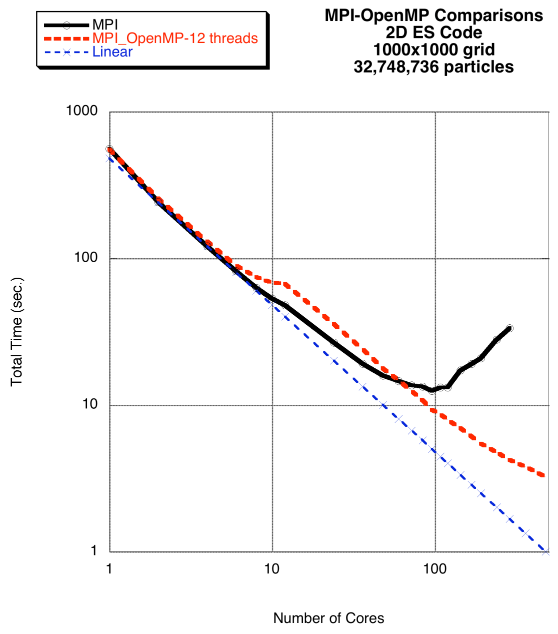

Figure 2. PIC Performance with MPI and mixed MPI/OpenMP on multiple shared memory nodes

On multiple shared memory nodes, there is some additional overhead with using nested domain decompositions. However, if there is substantial global communications, the hierarchical domain decomposition can outperform MPI alone because the global communications are staged. The most dramatic improvement came from the transpose used by the FFT. This improved the performance of a 2D parallel FFT by nearly a factor of three. The overall performance of a 2D electrostatic spectral PIC code, shown in Figure 2, improved by nearly a factor of 4.

We are currently applying this approach to a cluster of GPUs. Each GPU will have its own domain decomposition[2], and these will be connected by MPI.

Acknowledgements

This work is supported by the USDOE SciDAC program and UCLA IDRE.

References

[1] V. K. Decyk, "UPIC: A framework for massively parallel particle-in-cell codes," Computer Phys. Comm. 177, 95 (2007).

[2] Viktor K. Decyk and Tajendra V. Singh ,"Adaptable Particle-in-Cell algorithms for graphical processsing units," Computer Physics Communications 182, 641 (2011)