

# Global Address Space Communication Techniques for Gyrokinetic Fusion Applications on Ultra-Scale Platforms

R Preiss<sup>1</sup>, N Wichmann<sup>2</sup>, B Long<sup>2</sup>, J Shalf<sup>1</sup>, S Ethier<sup>3</sup> and A Koniges<sup>1</sup>

<sup>1</sup> Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA

<sup>2</sup> CRAY Inc., St. Paul, MN 55101, USA

<sup>3</sup> Princeton Plasma Physics Laboratory, Princeton, NJ 08543, USA

E-mail: rpreiss1@lbl.gov

**Abstract.** Recent innovations in computer architecture, system architecture, and programming models require application codes from a variety of areas to be updated to fully exploit the technological potential on cutting-edge HPC systems, for example, systems with hardware support for the Partitioned Global Address Space (PGAS) model. We explore new parallel language constructs based on the PGAS model using the communication kernel of a real-world magnetic fusion simulation code. Novel single- and multithreaded communication algorithms using a one-sided messaging paradigm based on Coarrays are first evaluated in a benchmark suite and then integrated into the fusion code. Experiments on up to 131K processors on a Cray XE platform show that the performance of the heavily optimized original MPI communication kernel can be significantly improved by our new PGAS communication algorithms.

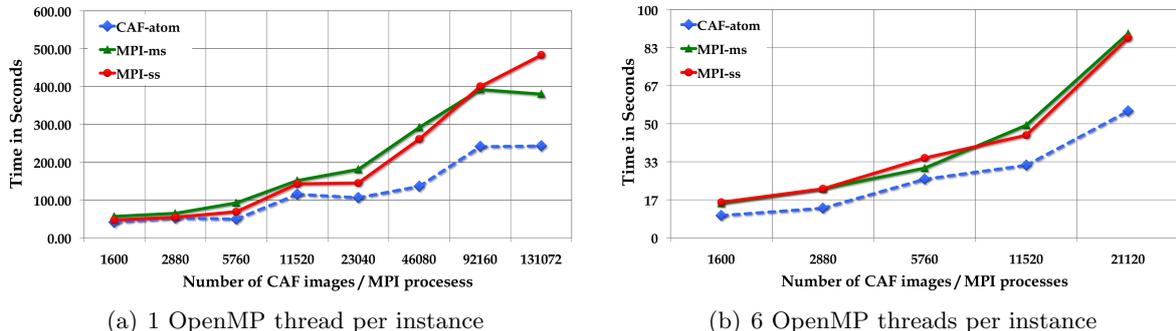
## 1. Introduction

The Gyrokinetic Tokamak Simulation (GTS [4]) code—a global 3D particle-in-cell (PIC) code with MPI and OpenMP support—has been selected among many of today’s fusion codes for the petascale postdoc program. GTS is a general geometry PIC code developed to study plasma microturbulence in toroidal, magnetic confinement devices called tokamaks. Microturbulence is a complex, nonlinear phenomenon that is believed to play a key role in the confinement of energy and particles in fusion plasmas [2], so understanding its characteristics is of utmost importance for the development of practical fusion energy.

Because of one of the levels implemented for parallelism in GTS, particles can move from one toroidal domain to another while they travel around the torus (the typical domain of a magnetic fusion reactor). This shift phase is the one we have studied the most so far in GTS since it represents the most communication-intensive routine in GTS and will gain in importance when scaling GTS to petascale or even exascale supercomputers.

## 2. Particle Shift Algorithms in GTS

At each time step, about 10% of the particles inside of a toroidal domain are communicated to adjacent toroidal neighbors, which translates to about 100 GB of data having to be communicated each time step shift is called in a 1-billion particle simulation run.



**Figure 1.** Weak scaling studies of a Coarray shifter and two MPI particle shifter algorithms with (a) no and (b) full (6 OpenMP threads per NUMA node) OpenMP support .

The original MPI particle shift algorithm, which implemented a nearest-neighbor communication pattern using MPI\_Send\_Recv functionalities, has been extensively researched in the past. This research resulted in a set of optimized, single-threaded MPI algorithms for shift containing, among others, nonblocking MPI send and receive operations, various MPI communication techniques (e.g., buffered send), and the usage of MPI data types (e.g., MPI\_Type\_create\_indexed\_block) to eliminate particle packing overheads. In addition—since GTS supports a hybrid MPI/OpenMP programming model—multithreaded MPI algorithms exploiting shared memory work-sharing constructs and novel communication and computation overlapping techniques using the newly introduced OpenMP tasks [3] have been developed over the years. However, all optimized MPI algorithms share the large bulk data transfers to exchange moving particles according to the rule that performance is optimized by sending fewer and larger messages.

This is in contrast to the strategy used for our new Coarray shift algorithms, which exploit the one-sided nature of the PGAS programming model. We implemented novel Coarray algorithms—that cannot be expressed in a two-sided message passing scheme like MPI-1—using more, but smaller messages with lower startup and completion costs. Building upon such lightweight one-sided communication techniques, we can efficiently spread out the communication over a longer period of time, resulting in a reduction of bandwidth requirements and a more sustained communication and computation overlap. Moreover, the expression of the one-sided messaging semantics as language constructs improves the legibility of the code and allows the compiler to apply communication optimizations. Hardware support for PGAS constructs and one-sided messaging, such as that provided by NERSC’s recent Cray XE6 Gemini interconnect, is essential to realize the performance potential of these new approaches. We also developed novel hybrid PGAS/OpenMP communication algorithms, which distribute the computational as well as the communication work load among OpenMP threads based on an advanced programming model that extends the classical hybrid distributed/shared memory model as used in the MPI algorithms. We note that we replace only the existing MPI communication kernel by a new algorithm using Coarrays and leave the rest of the physics simulation code unchanged, which still has MPI function calls in it.

Figure 1 presents the wall-clock runtime of a Coarray shifter implementation (*CAF-atom*) and of two MPI shift algorithms (*MPI-ms*, *MPI-ss*), running with no OpenMP support (Figure 1(a)) and full (i.e., 6 OpenMP threads per instance, on each NUMA node) OpenMP support (Figure 1(b)) on the Cray XE6 at NERSC. The algorithms are evaluated in a specially designed benchmark suite simulating GTS production run settings. Data for the single-threaded experiments was collected for concurrencies ranging from 1,600 up to 131,072 processor cores. For the multithreaded runs we run on 9,600 up to 126,720 processor cores on the Cray XE6

machine. All runtime numbers presented in Figure 1 are based on weak scaling experiments. We can observe in both tests—for the singlethreaded and the multithreaded runs of the shifter benchmark suite—a steady increase in runtime for shifting particles in a torus with increasing concurrencies, whereas one would expect a flat line along the x-axis for weak scaling experiments. This motivates optimizing this communication-intense GTS step to enable higher concurrencies as planned to model future nuclear fusion devices. Figure 1 shows that in both cases (OpenMP turned on or off) the Coarray implementation substantially outperforms the best MPI implementations, despite the extensive work in profiling and optimization of the communication layer of the GTS code.

Integration of the best Coarray and MPI shifter implementations to the GTS production code confirms that the standalone shifter communication benchmark correctly predicts the performance benefits of the particle shift phase for the full application code. This result proves that sending more frequent smaller messages enables the Coarray approach to outperform the message-passing implementations because of the enhanced communication and computation overlap as well as the better network bandwidth utilization. Employing a similar strategy of transmitting smaller more frequent messages is not practical in MPI and would require significant efforts in MPI-2, which would obscure semantics and science because of its semantic limitations [1].

### Acknowledgments

A majority of the work was supported by the Petascale Initiative in Computational Science at NERSC. Some additional research on this paper was supported by the Cray Center of Excellence at NERSC. Additionally, we are grateful for interactions with Nicholas Wright (NERSC) and for the extended computer time as well as the valuable support from NERSC. This work was supported by the Director, Office of Science, Advanced Scientific Computing Research, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

### References

- [1] Dan Bonachea and Jason Duell. Problems with using MPI 1.1 and 2.0 as compilation targets for parallel language implementations. *International Journal of High Performance Computing and Networking*, 1:91–99, August 2004.
- [2] S. Ethier, W. M. Tang, R. Walkup, and L. Oliker. Large-scale gyrokinetic particle simulation of microturbulence in magnetically confined fusion plasmas. *IBM Journal of Research and Development*, 52(1/2):105–115, 2008.
- [3] Robert Preissl, Alice Koniges, Stephan Ethier, Weixing Wang, and Nathan Wichmann. Overlapping communication with computation using OpenMP tasks on the GTS magnetic fusion code. *Scientific Programming*, 18:139–151, August 2010.
- [4] W. X. Wang, Z. Lin, W. M. Tang, W. W. Lee, S. Ethier, J. L. V. Lewandowski, G. Rewoldt, T. S. Hahm, and J. Manickam. Gyrokinetic Simulation of Global Turbulent Transport Properties in Tokamak Experiments. *Physics of Plasmas*, 13, 2006.