Annual Report for Blue Waters Professor Allocation

- **Project Information**
  - Title: Algorithms for extreme scale systems
  - PI: William Gropp, University of Illinois Urbana-Champaign
  - Collaborators: Luke Olson, University of Illinois Urbana-Champaign
  - Contact: wgropp@illinois.edu

- **Executive summary (150 words)**

Continued increases in the performance of large-scale systems will come from greater parallelism at all levels. At the node level, we see this both in the increasing number of cores per processor and the use of large numbers of simpler computing elements in GPGPUs. The largest systems must network tens of thousands of nodes together to achieve the performance required for the most challenging computations. Successfully using these systems requires new algorithms and new programming systems. My research looks at the effective use of extreme scale systems. Over the last year we have built on the new communication model that better fits the performance of multicore nodes to develop new algorithms for sparse matrix-vector products and better understand the behavior of non-blocking algorithms for the Conjugate Gradient method. We also developed a simple implementation of the MPI Cartesian topology routines that significantly outperforms the available implementations.

- **Description of research activities and results**
  - *Key Challenges:* At extreme scale, even small inefficiencies can cascade to limit the overall efficiency of an application. New algorithms and programming approaches are needed to address barriers to performance.
  - *Why it Matters*: This work directly targets current barriers to effective use of extreme scale systems by applications. For example, Krylov methods such as Conjugate Gradient are used in many applications currently being run on Blue Waters (MILC is one well-known example). Developing and demonstrating a more scalable version of this algorithm would immediately benefit those applications. In the longer term, the techniques that are developed will provide guidance for the development of highly scalable applications.
  - *Why Blue Waters:* Scalability research relies on the ability to run experiments at large scale, requiring tens of thousands of nodes and hundreds of thousands of processes and cores. Blue Waters provides one of the few available environments where such large-scale experiments can be run. In addition, only Blue Waters provides a highly capable I/O system, which we plan to use in developing improved approaches to extreme-scale I/O.
  - *Accomplishments:* We took advantage of the "max rate" performance model to develop new implementations of the MPI Cartesian topology routine that provides significantly better performance with an easy to implement method; in some cases the communication performance was 2-3x as fast. We have also continued development of scalable Krylov Methods and node-aware methods for optimized sparse matrix-vector computations.

A problem for applications is to write codes that run well without ad hoc, site-specific, non-portable tools or environment settings. One example is mapping processes to compute resources (that is assign processes to processors). MPI provides a good way to do this with the virtual process topology functions. For example, codes with a regular mesh should use MPI_Cart_create to produce an MPI communicator. By using this communicator, along with other MPI routines to determine neighboring processes in the communicator, an application should be able to run efficiently on any large-scale system.

Unfortunately, the process topology support in current MPI implementations are not very good, and applications must either forgo the performance or use ad hoc, non-portable techniques to achieve a good mapping. Such tools do exist for Blue Waters, but these do not provide the right solution to the problem. Applications should be able to rely on the features in MPI and not need to use non-standard, non-portable methods.

By using insight gained from our new performance model, we developed and refined an alternative implementation of MPI_Cart_create that provides a significant performance benefit, as show in Figure 1. Integrating this approach, which is itself highly portable, into existing MPI implementations would improve the performance of any application that uses the MPI Cartesian topology routines without requiring any non-standard, non-portable tools or code. Further, using simple information about the interconnect can further improve the performance of applications that use the Cartesian topology routines. This work received a “Best Paper” award at EuroMPI’18, and has been invited to a special issue of the Journal Parallel Computing. This work substantially improved on results shown in last year’s report; recent work included taking into account not only the nodes but the sockets within the node.

Additional work has been exploring the use of node aware organization of communication, particularly in sparse matrix-vector and sparse matrix-matrix multiplication. This work was also presented at EuroMPI’18, and work is continuing to both better understand the impact of load imbalance and the use of MPI 3 shared memory as an alternative to thread-parallelism.

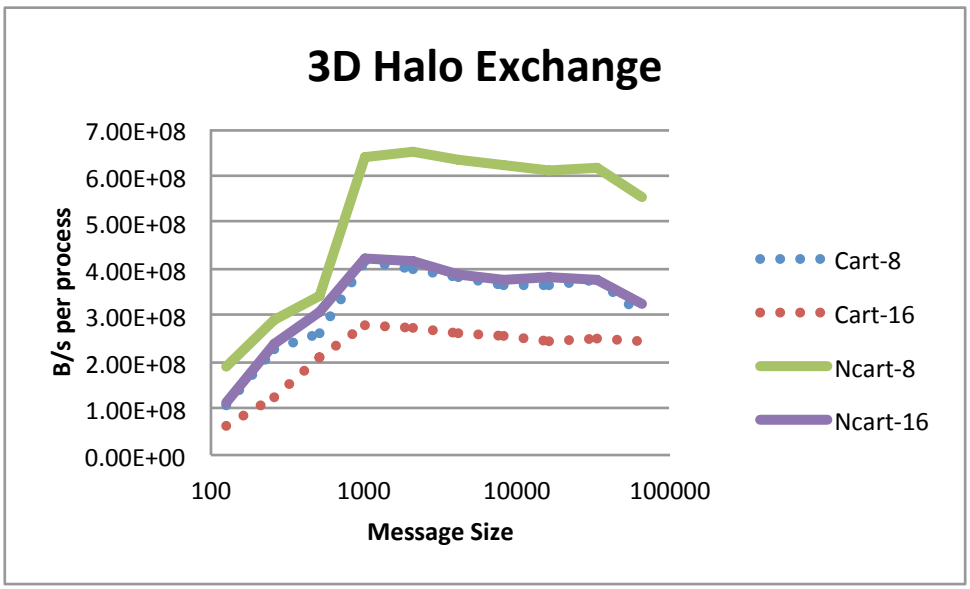

Figure 1: Communication performance for a 3D Halo exchange on Blue Waters, comparing the Cray MPI implementation of MPI_Cart_create with our node-aware version (the "Ncart" lines), showing significantly increased performance.

- **List of publications and presentations associated with this work**

Publications

Using Node Information to Implement MPI Cartesian Topologies, Gropp, William D., Proceedings of the 25th European MPI Users' Group Meeting, 18:1–18:9, 2018.

Improving Performance Models for Irregular Point-to-Point Communication, Bienz, Amanda, Gropp, William D., and Olson, Luke N., Proceedings of the 25th European MPI Users' Group Meeting, 7:1–7:8, 2018.

Presentations

These are some of the presentations that included reference to Blue Waters:

- Using Node Information to Implement MPI Cartesian Topologies, EuroMPI'18, Barcelona, Spain, September 2018. **Best paper.**
- Managing Code Transformations for Better Performance Portability, at Workshop on Clusters, Clouds, and Data for Scientific Computing (CCDSC) 2018, Lyon, France, September, 2018.
- Thinking about Parallelism and Programming, keynote for 4th European Workshop on Parallel and Distributed Computing Education for Undergraduate Students (Euro-EDUPAR) at EuroPar'18, Turin, Italy, August, 2018.
- Challenges for Developing \& Supporting HPC Applications, Session (chaired) at ISC18, Frankfurt, Germany, 2018

Plan for 2019

These projects have made good progress over the last year and are expected to expand their need for scalability studies. In addition, based on the experience with poor I/O performance, we are looking at parallel I/O approaches. The research efforts for the next year include

1. Optimizing parallel sparse matrix-vector product by optimizing intra- and inter-node communication
2. Parallel I/O autotuning and adaptivity
3. Communication optimized Krylov methods.

Most of these experiments study behavior at scale and typically need only a short run time but with 10,000-20,000 nodes. In order to produce timings at scale that are consistent, reproducible, and accurate, a typical run may require anywhere from a few minutes to 30 minutes per test. Thus, tests at scale may require 1,000-10,000 node-hours each. Because these tests are being used to evaluate different algorithms, most of which are developed as a result of evaluating the results of experiments on Blue Waters and at scale, it is difficult to determine a priori the amount of time that will be needed. Over the past year, we were careful to limit the scale for tests in order to limit the amount of resources consumed; as a result, we used a little under 20,000 node hours. In the upcoming year, I expect several projects to run at full scale by the end of the year. If each of the 3 projects requires 2 tests at full scale (each taking 20 minutes at 10,000 nodes), along with a sequence of scaling tests (another 50%) and some development time, 40,000 node hours would be needed. Depending on the progress of the algorithm development efforts, more time (as much as the 245,000 node-hour allocation) or less may be required. An exact estimate simply is not possible for this type of basic computer science research. For a specific request, 40,000 node hours should be sufficient; however, the option for more time, up to the original allocation, is highly desirable.

Few other resources will be needed. While some IO scalability studies may require files in the multi-Terabyte range, these will be temporary files. Similarly, little networking is expected.

Estimated distribution of time: Q1: 25%, Q2: 25%; Q3: 25%; Q4: 25%

Rationale for the distribution: Most projects are now far enough along that a uniform distribution of time is most likely. However, scaling runs are likely to consume the largest fraction of the time, and these are hard to predict. In addition, the Krylov methods project is in support of completing the PhD dissertation of Paul Eller, and should be completed in 2019.