# Communication Characterization and Optimization of Applications Using Topology-Aware Task Mapping on Large Supercomputers

Sarat Sreepathi
Oak Ridge National Laboratory
Oak Ridge, TN, USA
sarat@ornl.gov

Ed D'Azevedo
Oak Ridge National Laboratory
Oak Ridge, TN, USA
dazevedoef@ornl.gov

Bobby Philip
Oak Ridge National Laboratory
Oak Ridge, TN, USA
philipb@ornl.gov

Patrick Worley
Oak Ridge National Laboratory
Oak Ridge, TN, USA
worleyph@ornl.gov

## ABSTRACT

On large supercomputers, the job scheduling systems may assign a non-contiguous node allocation for user applications depending on available resources. With parallel applications using MPI (Message Passing Interface), the default process ordering does not take into account the actual physical node layout available to the application. This contributes to non-locality in terms of physical network topology and impacts communication performance of the application. In order to mitigate such performance penalties, this work describes techniques to identify suitable task mapping that takes the layout of the allocated nodes as well as the application's communication behavior into account. During the first phase of this research, we instrumented and collected performance data to characterize communication behavior of critical US DOE (United States - Department of Energy) applications using an augmented version of the mpiP tool. Subsequently, we developed several reordering methods (spectral bisection, neighbor join tree etc.) to combine node layout and application communication data for optimized task placement. We developed a tool called mpiAproxy to facilitate detailed evaluation of the various reordering algorithms without requiring full application executions. This work presents a comprehensive performance evaluation (14,000 experiments) of the various task mapping techniques in lowering communication costs on Titan, the leadership class supercomputer at Oak Ridge National Laboratory.



*ICPE'16, March 12 - 18, 2016, Delft, Netherlands*



ACM ISBN 978-1-4503-4080-9/16/03…$15.00

DOI: http://dx.doi.org/10.1145/2851553.2851575

## Keywords

Communication Characterization; Reordering algorithms; Topology-Aware Optimization

## 1. INTRODUCTION

Modern leadership class supercomputers have a large number of processing elements (PEs) and the trends point to increasing complexity (beyond $O(1M)$ PEs). Moreover, such systems have a wide range of users with different requirements resulting in a plethora of job sizes and compute requirements. The job schedulers perform a critical role by allocating available resources to jobs based on numerous factors (e.g., queue type, request size, priority, wait time, other jobs etc. [11])

During typical operations, a job scheduler can assign a non-contiguous allocation to a user request depending on resource availability. In the worst case, an application could end up with an allocation where the nodes are sparsely scattered across the machine. This results in performance degradation due to long routes taken by communication messages and can be further impacted by network congestion. Even supercomputers with advanced interconnects exhibit scalability issues in such scenarios.

Task-mapping refers to the assignment of an application's tasks to available processing elements. With parallel applications using MPI (Message Passing Interface), the default task order is sequential. It does not consider the non-locality

This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan(http://energy.gov/downloads/doe-public-access-plan).

aspects of the actual physical node layout that was allocated to the application.

In order to mitigate such performance penalties, this work describes techniques to optimize task placement for the target application based on its behavior as well as network topology of the allocated nodes.

The primary contributions of this research are:

- Task mapping techniques that consider the target application's communication pattern and physical network topology to reduce communication costs (Section 2.2)
- `mpiAproxy`: A tool that simulates target application's communication behavior to estimate communication costs (Section 2.3).
- Detailed communication characterization of critical scientific applications from the Co-Design ecosystem and big science applications in the domains of nuclear fusion, climate and radiation diffusion. (Section 6)

The rest of the paper is organized as follows: We elaborate on the methodology for this study in Section 2. Related work is discussed in Section 3. As part of this research, we investigated communication behavior of several DOE applications. The applications are summarized in Section 4. Titan, the experiment platform is described in Section 5. We present a comprehensive performance evaluation of the various task mapping techniques along with communication characterization data in section 6 and conclude with future work.

# 2. METHODOLOGY

In this section, we detail the various phases of our experiment methodology. Section 2.1 presents an overview of our communication characterization tool. We discuss the details of the various reordering algorithms in section 2.2 and Section 2.3 describes the communication estimation tool that was developed to facilitate a detailed evaluation of the different methods.

## 2.1 Augmented mpiP Tool

For communication characterization, we used an augmented version of the mpiP [37] profiling library that was previously used in our application characterization studies [36]. We used this tool to perform a detailed analysis of an application's communication pattern. It was used to record the point-to-point communication data between communicating MPI tasks; message volume in `.mpiPpv` files and message counts in `.mpiPpc` files. Additionally, it generates histograms for the point-to-point and collective communication message sizes in `.mpiPpsh` and `.mpiPcsh` respectively.

The translation of collective operations into underlying point-to-point messages depends on the MPI implementation and the compute platform. For instance, a collection operation like MPI_Allreduce on Cray XK7 using MPICH could result in a set of point-to-point messages that could be different for the same operation on IBM BlueGene using OpenMPI. In contrast, the point-to-point communication pattern is platform-independent for a target application for a specified experiment configuration. This work focuses on on the point-to-point communication data and we intend to encompass collective communication data in the future.

## 2.2 Reordering Methods

Three graph based heuristics are compared for generating a mapping between the communication graph of the application and the network topology of the allocated nodes at run-time. The recursive spectral bisection and Reverse Cuthill-McKee (RCM) [17, 20] orderings were used in topology mapping in [23]. A bottom-up neigbor-join tree [24, 32] clustering heuristic commonly used in bioinformatics for the creation of phylogeneitc trees was also used. The general approach follows Hoefler and Snir [23] in generating an ordering (or permutation of the graph vertices) on the communication graph and an ordering for the network topology and matching the vertices of the two orderings to produce the topology-aware mapping.

The communication graph $C(i, j)$ is the volume of data sent from MPI process $i$ to process $j$. A threshold value (e.g. 95% of total message volume ) was used to ignore small transfers and obtain a sparse communication graph. The heuristics were computed using Matlab and completed in only a few seconds.

The network topology is obtained by querying the coordinates of the allocated compute nodes on the 3D torus network on Titan using `rca_get_meshcoord` function from Cray's Resiliency Communication Agent (RCA) library. Titan has a $25 \times 16 \times 24$ 3D torus network where 2 compute nodes share a network interface. However, the links in the "y" direction have half the bandwidth of the links in the "x" and "z" directions. The number of hops through the 3D network was used as an indicator of the communication cost with "y" direction hops being twice as costly. For details on Titan, the experiment platform see Section 5.

### 2.2.1 *Spectral bisection techniques*

The spectral bisection technique can be used to perform nested dissection reordering [21] to reduce the amount of fill-in in sparse direct Cholesky factorization of large symmetric positive matrices. The recursive spectral bisection method attempts to find a nearly equal partitioning of the graph with a small separator. The symmetrized communication graph $((C + \texttt{transpose}(C))/2)$ or the communication hop graph was used as the weighted Laplacian graph. The eigenvector corresponding to the second smallest eigenvalue of the weighted Laplacian graph (also called the Fiedler vector) was used to partition the graph into two clusters. The spectral bisection algorithm was recursively applied to the two remaining subgraphs. The idea being clusters of nodes that exchange high volume of data should be mapped to compute nodes that have high network connectivity. The METIS library [34] was used to compute a balanced partition in [23]. Here the matlab `eig()` is used for small matrices ($N \leq 512$) and `eigs()` eigensolver based on the Lanczos algorithm was used to find the Fiedler vector.

There are two variants of the spectral methods used in this work. In `SPECTRAL0`, an unweighted Laplacian matrix is used where all off-diagonal entries are equal. The Fiedler eigenvector is computed from the unweighted Laplacian matrix. In `SPECTRAL1`, a weighted laplacian matrix is used to where the weights are related to the MPI communication graph or the network topology.

### 2.2.2 *Reverse Cuthill-McKee (RCM) algorithm*

The RCM ordering is commonly used as a heuristic for reducing the matrix band-width before performing sparse

direct Cholesky factorization. It is a variant of breadth-first ordering starting from an extremal peripheral vertex in the graph. RCM was very fast but RCM ignores the numerical weight of edges and considers only the sparsity pattern of the matrix. The Matlab `rcm()` function was used.

#### 2.2.3 Neighbor-join tree methods

The Neighbor-join tree is a bottom up clustering method where given a distance or cost matrix, the algorithm finds a hierarchical clustering of nodes to form a tree. The method find the smallest distance, say $d(x, y)$ and merge the nodes to form a new super node $[x, y]$. The cost matrix is then updated where $distance([x, y], z) = \max(d(x, z), d(y, z))$. A post-order labeling of the leaf nodes (label all left sub-tree, then right subtree) is given as the final ordering. The algorithm has $O(n^3)$ costs in the worst case.

We used two different algorithms in this family. In `NJTREE1`, the cost matrix is computed as the number of hops needed through the 3D torus where the hops in the y-direction has twice the cost. In `NJTREE0`, the cost matrix is the original network topology matrix but with negative weights so that links with high bandwidth have smaller costs.

### 2.3 mpiAproxy: Communication Estimation

We developed a tool called mpiAproxy to facilitate detailed evaluation of the various reordering algorithms without requiring full application executions.

The tool takes as input the detailed communication data of the application collected by the augmented mpiP tool. First, the target application is linked with the mpiP libraries and executed. This generates the point-to-point communication data; aggregate message counts and volume between communicating processes.

The mpiAproxy tool computes the average message size between communicating neighbors and simulates the communication pattern to obtain an estimate of the application's communication costs. The execution time of the tool depends on the target application it is emulating, typical runs take under a few minutes. It does this by sending a representative number of messages between communicating tasks with the relevant message size and records the time.

The tool uses an asynchronous communication protocol to avoid deadlocks. This approximation does not capture all the intricacies of a complex application but presents a broad overview for a significantly lower computation cost. For instance, it does not not reflect any load imbalances that are present in the target application. However, it facilitates extensive experimentation of task mapping layouts as the cost of each experiment is small, thereby enabling a comprehensive search for an optimal layout.

## 3. RELATED WORK

Finding the optimal mapping between the communication graph and network topology may be viewed as determining the graph isomorphism problem, which is $\mathcal{NP}$-hard. Thus heuristics are needed to generate the topology mapping. One approach finds the mapping as the optimization of a cost function (such as the combination of hop-byte and dilation metrics), another approach uses graph-based algorithm to find a good mapping.

Sankaran et al [33] used a genetic algorithms for optimizing the mapping for two large-scale parallel S3D and LAMMPS on the Cray XK7 machine. Bhanot et al [13] used simulated annealing to optimize task layout of parallel applications SAGE and UMT2000 on the BlueGene/L machine. Bollinger and Midkiff [15] proposed process annealing for assigning tasks to processors and connection annealing for scheduling communication to reduce network contention.

Solomonik et al [35] considered mapping 2.5D dense matrix LU factorization algorithms onto the BlueGene/P system. However, optimization-based heuristics are quite costly and are appropriate only for applications with well characteristized fixed communication pattern (such as nearest neighbor communication for stencil computation on a 2D or 3D rectangular grid or dense matrix computation) and mapping to known network partitions such as the BlueGene machine.

Another approach is to use graph-based heuristic algorithms to determine the mapping. Ercal [19] considered recursive bisection in the context of a hypercube topology where at each step the algorithm finds a minimum cut of the communication graph while maintaining approximately equal load and recursively assigns the subgraphs onto subcubes. Hoefler and Snir [23] considered heuristics based on graph similarity for irregular communication patterns. They considered several heuristics including recursive bisection, Reverse Cuthill-McKee (RCM) and a greedy heuristic for picking the node with highest communication need and paired with its closest neighbor to minimize communication cost. In his PhD thesis, Bhatele [14] considered several heuristics for general communication graphs. Deveci at al. [18] proposed a geometric partitioning algorithm for task placement and demonstrated performance improvements for a structured finite difference application among others.

The breadth-first traversal (BFT) simply visit nodes by breadth-first order. Note BFT has some similarity to RCM. The max heap traversal starts with the node that has maximum number of neighbors and place it at center of 2D mesh. All unmapped neighbors are placed on the heap sorted by the number unmapped neighbors. At each step, the node with highest number of unmapped neighbors is placed close to the centroid of its mapped neighbors. Since the Cray XT batch system cannot guarantee a contiguous partition and the applications of interest (such as the XGC particle-in-cell code or climate simulation) have unstructured and not easily predictable communication patterns, the focus of this work is on exploration of fast graph-based heuristic that can be computed dynamically in a short amount of time.

## 4. APPLICATIONS OVERVIEW

### 4.1 Co-Design Benchmarks, Proxies and Applications

We have looked at a broad range of applications that are widely used in the Co-Design community. There are three Co-Design Centers under the purview of DOE, namely Exascale Co-Design Center for Materials in Extreme Environments (ExMatEx), Center for Exascale Simulation of Advanced Reactors (CESAR) and Center for and Exascale Simulation of Combustion in Turbulence (ExaCT). Additionally we studied the performance of several benchmarks associated with the CORAL (Collaboration of Oak Ridge, Argonne and Livermore)[16] acquistion effort to purchase next generation supercomputers.

#### 4.1.1 AMG2013

AMG2013 [1] is a parallel algebraic multigrid solver for linear systems arising from problems on unstructured grids. It has been derived directly from the BoomerAMG solver in the hypre library, a large linear solver library. The default problem is a Laplace type problem on an unstructured domain with various jumps and an anisotropy in one part.

AMG2013 is a highly synchronous code. The communications and computations patterns exhibit the surface-to-volume relationship common to many parallel scientific codes.

#### 4.1.2 BoxLibAMR

BoxLibMiniAMR is a proxy app developed by the ExaCT Co-Design center [2]. It uses a structured-grid Adaptive Mesh Refinement (AMR) approach to study combustion. It is being used to address challenges that arise in investigation of advanced low-emissions combustion systems.

#### 4.1.3 HPCG

The HPCG (High Performance Conjugate Gradients) Benchmark project [3] is an new benchmark effort to create a more relevant metric for ranking HPC systems than the High Performance LINPACK (HPL) benchmark, that is currently used by the TOP500 listing. HPCG is designed to better match the computational, communication and data access patterns of a broad range of applications in contrast to HPL.

#### 4.1.4 LULESH

The Livermore Unstructured Lagrange Explicit Shock Hydrodynamics (LULESH) proxy application [4, 25] is being developed at Lawrence Livermore National Laboratory. Originally developed as one of five challenge problems for the DARPA UHPC program, it has since evolved and has received widespread use in DOE research programs as a mini-app representative of simplified 3D Lagrangian hydrodynamics on an unstructured mesh.

#### 4.1.5 MCB

Monte Carlo Benchmark (MCB) [5] is a Co-Design application developed at LLNL that is intended for use in exploring the computational performance of Monte Carlo algorithms on parallel architectures. It models the solution of a simple heuristic transport equation using a Monte Carlo technique.

#### 4.1.6 MultiGrid_C

MultiGrid_C is a proxy app developed by the ExaCT Co-Design center [7]. It is a finite-volume multigrid solver that supports different variants.

#### 4.1.7 Nek5000

Nek5000 [8] is a large application designed to simulate laminar, transitional, and turbulent incompressible or low Mach-number flows with heat transfer and species transport. It is also suitable for incompressible magnetohydrodynamics (MHD).

#### 4.1.8 Nekbone

The Nekbone mini-app [9] developed by the Center for Exascale Simulation of Advanced Reactors (CESAR) is used to to study the computationally intense linear solvers that account for a large percentage of Nek5000, the full application it is modeled after. It is also meant to emulate the communication behavior of Nek5000, specifically nearest-neighbor data exchanges and vector reductions.

### 4.2 Big Science Applications

In addition to benchmarks and proxy apps, we were specifically interested in large scientific applications that have intricate communication patterns that vary based on experiment configuration. Such applications manifest in hard to predict communication behavior that makes optimization efforts more challenging.

#### 4.2.1 XGC

XGC is a whole-device modeling code, with a special strength in modeling the edge plasma (and its effect on the core plasma) in tokamak fusion reactors at first-principles level. To be precise, XGC is a full-function 5D gyro kinetic particle-in-cell (PIC)/finite element code that is uniquely capable of simulating the tokamak plasma in realistic diverted magnetic geometry, as well as covering core plasma all the way to the magnetic axis, together with neutral particle recycling at material wall surface and their Monte-Carlo transport using atomic charge-exchange and ionization cross-sections.

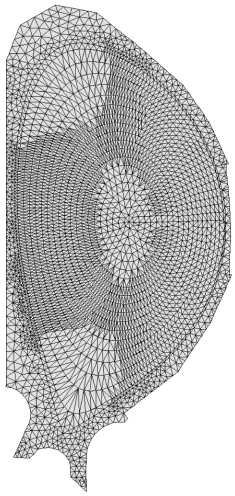

**Figure 1: XGC Triangular Mesh**

XGC is based upon the conventional particle-in-cell ODE solver method, with options such as electron fluid solves for kinetic-fluid hybrid physics. To be more specific, the Lagrangian ODE particle information is scattered to unstructured triangular mesh nodes on which the PDE electromagnetic force field equations are solved. The force field information is then gathered back to the particle locations for time-advancing the ODE cycle of the particles. Unlike the conventional particle-in-cell codes, all the dissipative physical processes are solved on 5D configuration-velocity space grid using finite element, finite difference schemes, which include the non-linear Fokker-Planck collisions, charge exchange, ionization, and radiation. Due to the enormity of the particle number required for full- function multiscale gyrokinetic simulation of tokamak plasma, over 95% of the computing time is spent on the ODE particle push that scales very well on extreme number of compute processors. The PDE solver parts of the code, which encapsulates the

long range data dependencies in the PIC method are more demanding in extreme scale computing. One advantage of the full-function method used in XGC is that it encapsulates these global data dependencies with a small fraction of the total work: Thus, solvers currently account for less than 2% of the total computing time.

The tokamak device is partition into multiple domains by poloidal planes. Each plane consists of the identical unstructured triangular mesh (see Figure1). The mesh is used for charge deposition and the calculation of the electric field by finite element method and resulting linear systems are solved using PETSc iterative linear solvers preconditioned with Hypre multi-grid library.

The XGC code can be configure to assign MPI tasks contiguously within the same plane (called `plane major` mode) or to assign MPI tasks contiguously in the toroidal direction across planes (called `inter-plane major` mode).

#### 4.2.2 MPAS-Ocean

The Model for Prediction Across Scales (MPAS) [6] is a collaborative project for developing atmosphere, ocean and other earth-system simulation components for use in climate, regional climate and weather studies. MPAS-Ocean, the ocean component of this effort is designed for the simulation of the ocean system across different time and spatial scales. Additionally, it is intended for the study of anthropogenic climate change.

#### 4.2.3 NRDF

NRDF is a 3D non-equilibrium radiation diffusion code [31] that solves the time dependent non-equilibrium radiation diffusion equations that are important for solving the transport of energy through radiation in optically thick regimes with applications in several fields including astrophysics and inertial confinement fusion. The associated initial boundary value problems that are encountered often exhibit a wide range of scales in space and time and are extremely challenging to solve. The non-equilibrium radiation diffusion problem is discretized on structured adaptive mesh refinement (SAMR) hierarchies which consist of nested refinement levels with each level a union of non-overlapping patches at the same resolution. A method of lines (MOL) approach is used with a cell centered finite volume spatial discretization followed by discretization in time. To solve the nonlinear systems arising at each timestep an inexact Newton method is used with GMRES for the linear solver. The linear system is preconditioned on SAMR grids with components that involve either a multilevel Fast Adaptive Composite Grid (FAC) solver or an asynchronous version of FAC (AFACx).

The Fast Adaptive Composite grid (FAC) method [30, 29] extends techniques from multigrid on uniform grids to locally refined grids. FAC solves problems on locally refined grids by combining smoothing on refinement levels with a coarse grid solve using an approximate solver, such as a V-cycle of multigrid. On parallel computing systems, the multiplicative nature of FAC introduces synchronization points during every correction step. Moreover, there is little opportunity to overlap communication with computation. These considerations led to development of asynchronous, or additive, versions of FAC that removes these synchronization points (AFAC, AFACx) [26, 27, 28].

## 5. TITAN: EXPERIMENT PLATFORM

We conducted our experiments on Titan [10], a Cray supercomputer installed at Oak Ridge National Laboratory. Titan is a hybrid-architecture Cray XK7 system with a theoretical peak performance exceeding 27 petaflops. It comprises of 18,688 compute nodes, wherein each node contains 16-core AMD Opteron CPUs and NVIDIA Kepler K20X GPUs for a total of 299,008 CPU cores and 18,688 GPUs. It has a total system memory of 710 terabytes, and utilizes Cray's high-performance Gemini network. Titan has a $25 \times 16 \times 24$ 3D torus network where 2 compute nodes share a network interface. As of June 2015, it is the second fastest supercomputer in the world according to the TOP500 list [12].

The software environment for the reported experiments is as follows: Cray PGI programming environment (version 5.2.40) which uses PGI 15.3 compilers and Cray's MPICH implementation (version 7.2.2).

## 6. RESULTS

Large supercomputing systems exhibit a high degree of performance variability due to various factors, e.g., jitter, load imbalances, prevailing network traffic etc. Hence it is not prudent to compare best execution times or a few data points for each scenario to obtain an accurate performance comparison.

In this section, we present highlights from a large set of experiments that show the efficacy of the various reordering techniques in improving communication performance of target scientific applications. Each plot shows experiment data for 40 `mpiAproxy` executions for each reordering method (7 mappings) for a total of 280 data points.

`DEFAULT` refers to the baseline task mapping scenario where MPI ranks are sequentially assigned to the corresponding nodes. `NJTREE0, NJTREE1` are task mappings that are obtained from the heirarchical clustering/neighbor-join tree algorithms. `SPECTRAL0, SPECTRAL1` refer to the task mappings from the spectral methods. Finally, `RCM` refers to the Reverse CutHill McKee ordering and `RANDOM` is a random ordering of MPI ranks that acts a control for the experiments.

We utilized violin plots to accurately portray the efficacy, performance variability and robustness of the various scenarios to facilitate an informed comparison. The violin plots [22] are similar to box plots, except that they also show the probability density of the data at different values. The plot includes markers for the median and inter-quartile ranges of the data. Overlaid on this box plot is a kernel density estimation. The length of the violin plot reflects the degree of variability across experiments.

For each application, we present the point-to-point message volume diagram and summarize its communication behavior. Subsequently, we present the communication costs with various task mappings that are derived from the reordering algorithms. These communication costs are obtained by executing `mpiAproxy` with the target application's communication profile (point-to-point communication message count and volume data).

The first subsection focuses on the various benchmarks and proxy applications from the DOE Co-Design centers. The second section presents results for several large scientific applications.

## 6.1 Co-Design Benchmarks, Proxies and Applications

### 6.1.1 AMG2013

The communication behavior of AMG2013 is shown in Figure 2. Figure 3 shows the relative performance of the various methods. In this case, RCM performs slightly better than the default mapping.

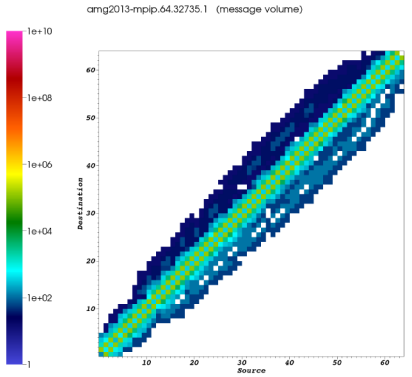


**Figure 2: AMG-3dlaplace: Point-to-point communication volume pattern with 64 MPI tasks**

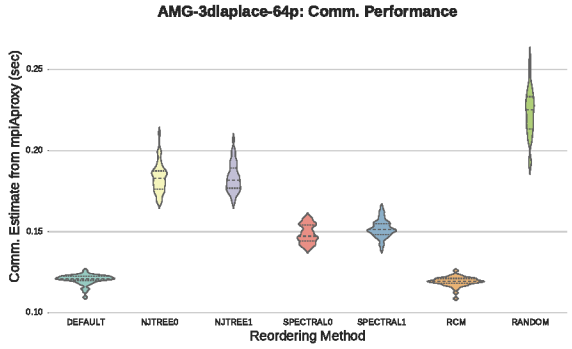


**Figure 3: Task Mapping: Communication Performance of AMG-3dlaplace with 64 MPI tasks**

### 6.1.2 BoxLibAMR

BoxLibAMR has a very dense communication pattern as shown in figure 4. Figure 5 shows the relative performance of the various methods. In this case, all methods outperform the default mapping as it is ill suited for this kind of dense communication that results in congestion.

### 6.1.3 HPCG

HPCG has a nearest neighbor communication pattern (Figure 6) with the dense communication concentrated along the diagonal. All reordering methods outperform the default mapping with `SPECTRAL0` providing the best mapping (Figure 7).

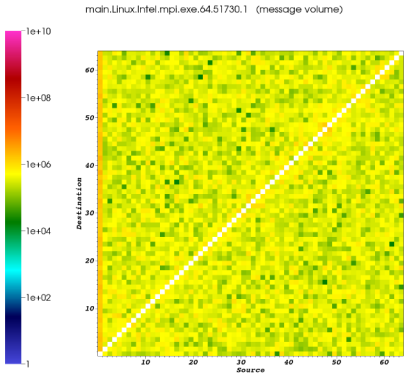


**Figure 4: BoxLibMiniAMR-inputs3d: Point-to-point communication volume pattern with 64 MPI tasks**

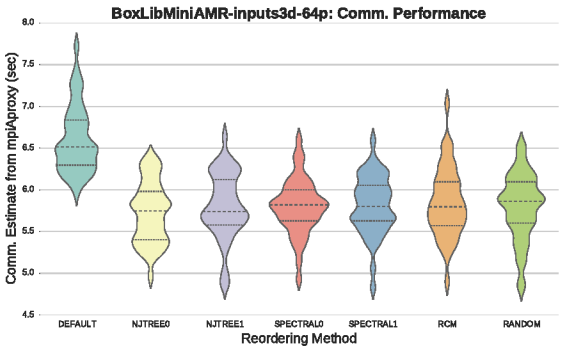


**Figure 5: Task Mapping: Communication Performance of BoxLibMiniAMR-inputs3d with 64 MPI tasks**

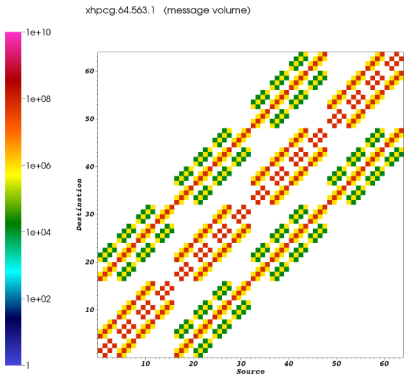


**Figure 6: HPCG: Point-to-point communication volume pattern with 64 MPI tasks**

### 6.1.4 LULESH

LULESH has a communication pattern (Figure 8) that appears similar to HPCG. The spectral methods deliver a good mapping in this scenario (Figure 9) with `SPECTRAL1` providing the best results. Please note that although LULESH and

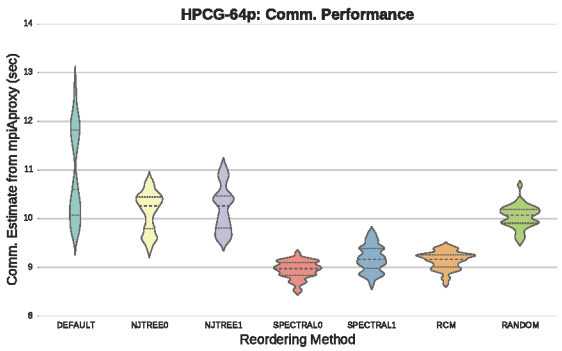


**Figure 7: Task Mapping: Communication Performance of HPCG with 64 MPI tasks**

HPCG *appear* to have similar patterns, the performance of the mapping algorithms differed due to the difference in the amount of off-diagonal communication.

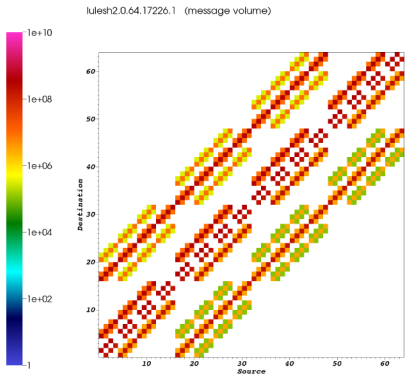


**Figure 8: LULESH: Point-to-point communication volume pattern with 64 MPI tasks**

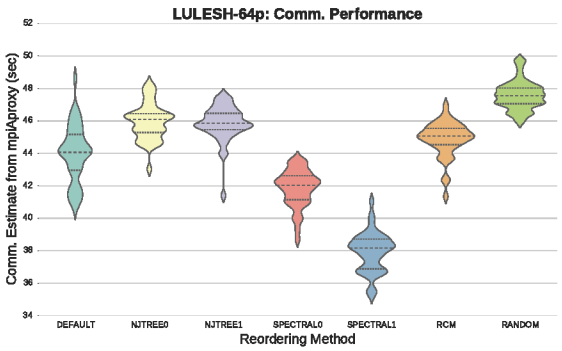


**Figure 9: Task Mapping: Communication Performance of LULESH with 64 MPI tasks**

### 6.1.5 *MCB*

MCB has an interesting scatter pattern originating from the origin (Figure 10). All algorithms outperform the default mapping with the spectral methods and RCM providing the best mapping (Figure 11).

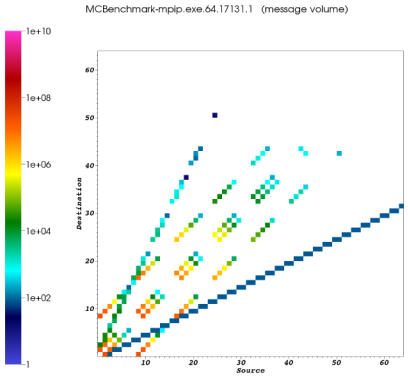


**Figure 10: MCB: Point-to-point communication volume pattern with 64 MPI tasks**

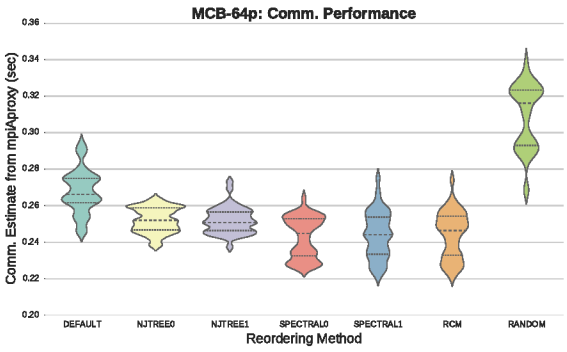


**Figure 11: Task Mapping: Communication Performance of MCB with 64 MPI tasks**

### 6.1.6 *MultiGrid_C*

MultiGrid_C for solving a 3d problem with 512 cells communicates mostly along the diagonal as well as pockets of nearby neighbors (Figure 12). In this case, the default mapping itself seems optimal (Figure 13).

### 6.1.7 *Nek5000*

Nek5000 communication pattern with the `vortex` problem configuration is shown in Figure 14. Again the default mapping itself seems optimal and at par with the spectral techniques (Figure 14)

### 6.1.8 *NEKBONE*

Figure 16 shows the communication pattern of NEKBONE with the `mgrid` problem. The default mapping is better for this scenario (Figure 17).

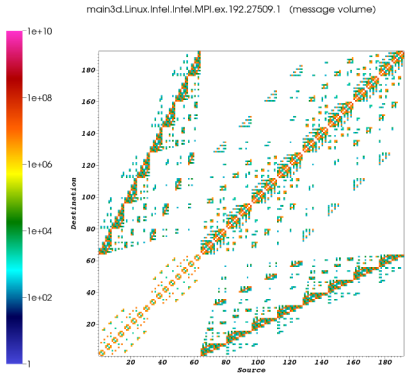


**Figure 12: MultiGrid_C-3d-512cells: Point-to-point communication volume pattern with 192 MPI tasks**

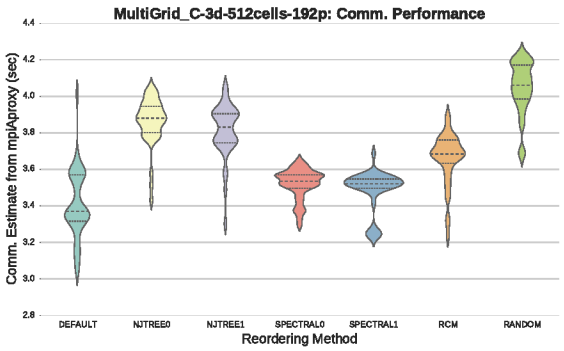


**Figure 13: Task Mapping: Communication Performance of MultiGrid_C-3d-512cells with 192 MPI tasks**

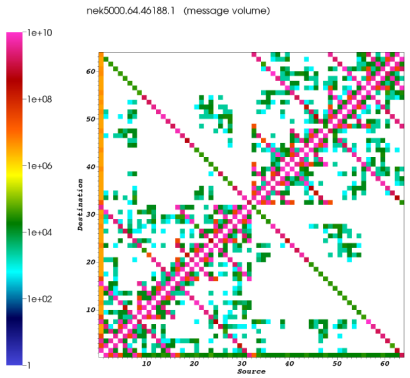


**Figure 14: Nek5000-vortex: Point-to-point communication volume pattern with 64 MPI tasks**

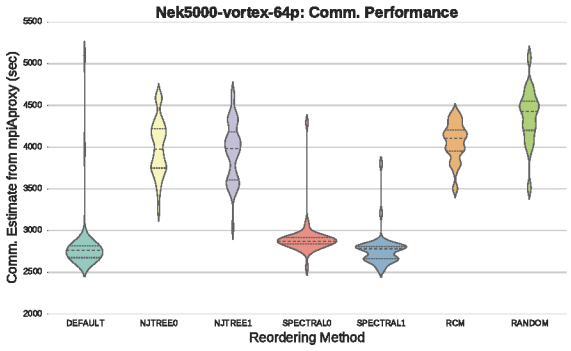


**Figure 15: Task Mapping: Communication Performance of Nek5000-vortex with 64 MPI tasks**

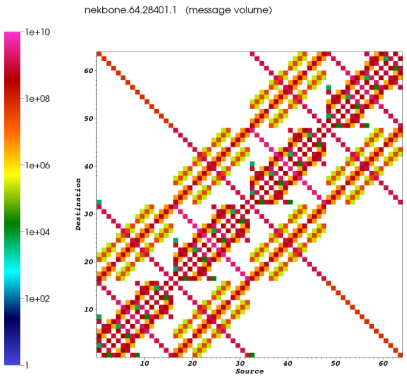


**Figure 16: NEKBONE-mgrid: Point-to-point communication volume pattern with 64 MPI tasks**

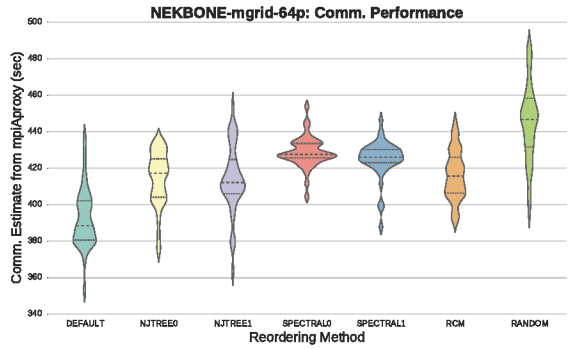


**Figure 17: Task Mapping: Communication Performance of NEKBONE-mgrid with 64 MPI tasks**

## 6.2 Big Science Applications

### *6.2.1 XGC*

As part of this work, we have performed an in-depth analysis of a nuclear fusion code, XGC using different problem configurations (interplane/planemajor ordering and poloidal decomposition). Figure 18 shows the point-to-point communication volume without poloidal decomposition and Figure 20 with poloidal decomposition. Both experiment configurations used `interplanemajor` ordering. There is a marked increase in communication volume with poloidal decomposition.

In the first case, the tree methods perform better (Figure

19) whereas the spectral methods (Figure 21) fare well with poloidal decomposition. The *SPECTRAL0* method consistently provides a superior task mapping for the poloidal decomposition configuration.

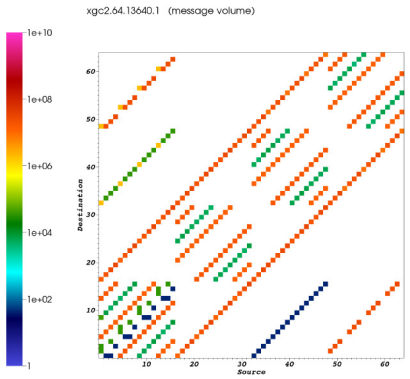


**Figure 18: XGC-interplanemajor: Point-to-point communication volume pattern with 64 MPI tasks**

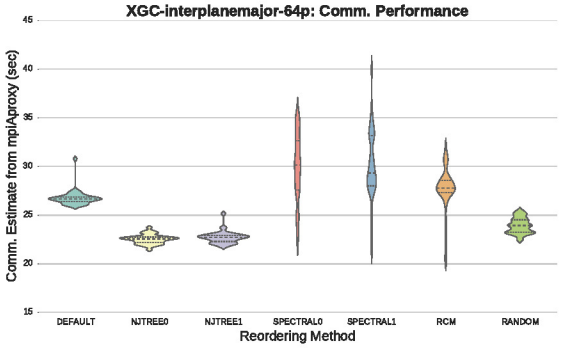


**Figure 19: Task Mapping: Communication Performance of XGC-interplanemajor with 64 MPI tasks**

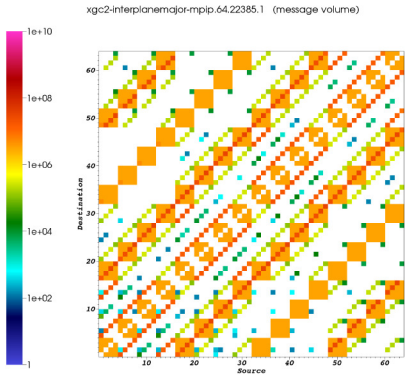


**Figure 20: XGC-interplanemajor-poldecomp: Point-to-point communication volume pattern with 64 MPI tasks**

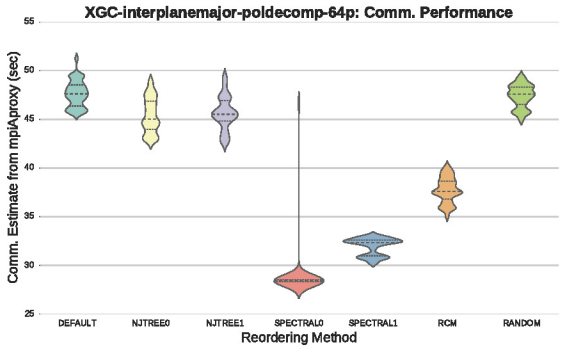


**Figure 21: Task Mapping: Communication Performance of XGC-interplanemajor-poldecomp with 64 MPI tasks**

At larger scale, Figure 22 shows the communication behavior at 2048 cores. Again, `SPECTRAL0` provides the optimal mapping in this scenario (Figure 23).

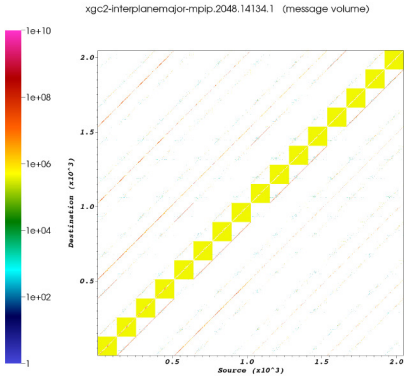


**Figure 22: XGC-interplanemajor-poldecomp-10ts: Point-to-point communication volume pattern with 2048 MPI tasks**

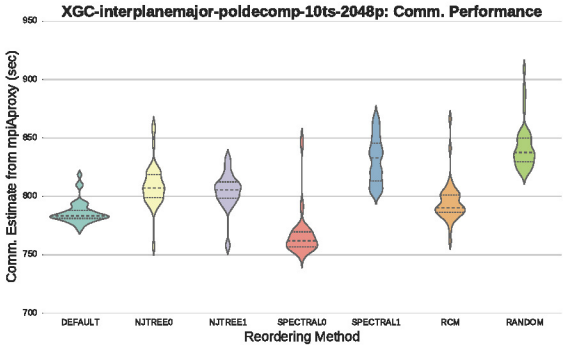


**Figure 23: Task Mapping: Communication Performance of XGC-interplanemajor-poldecomp-10ts with 2048 MPI tasks**

### 6.2.2 *MPAS-Ocean*

The communication behavior of MPAS-Ocean using 128 processes is shown in Figure 24. The different task mapping methods outperform the default ordering with the tree methods showing the best performance (Figure 25).

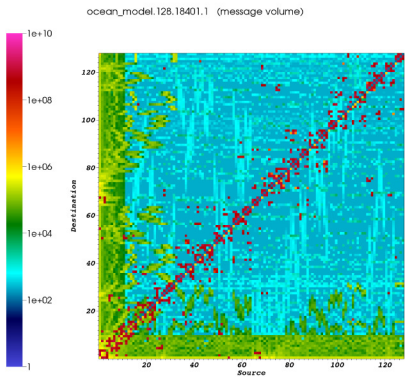


**Figure 24: MPAS-default-partitioning: Point-to-point communication volume pattern with 128 MPI tasks**

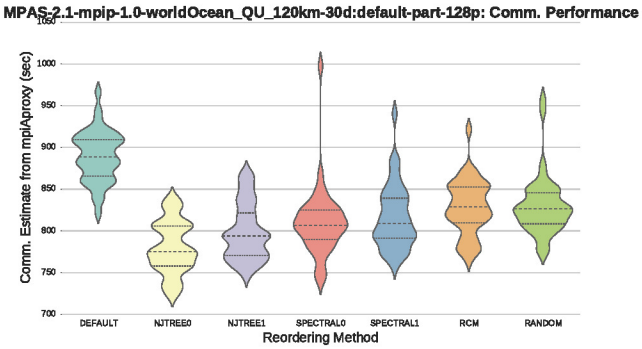


**Figure 25: Task Mapping: Communication Performance of MPAS-default-part with 128 MPI tasks**

At larger scale, MPAS-Ocean replicates the pattern from the smaller run with mutually exclusive subdomains (Figure 26). The tree methods still outperform other mappings (Figure 27)

### 6.2.3 *NRDF*

Finally, we did a detailed characterization of NRDF in various AMR (Adaptive Mesh Refinement) configurations. We present the highlights here. Figure 28 illustrates the communication pattern of NRDF using the AFACx algorithm in `256b31` configuration. The tree methods provide the best performance for this problem (Figure 29).

The communication pattern of NRDF using the FAC algorithm is shown in Figure 30. It is substantially different from the AFACx algorithm in the same configuration (`256b31`). All reordering algorithms provide a better mapping than the default in this case (Figure 31).

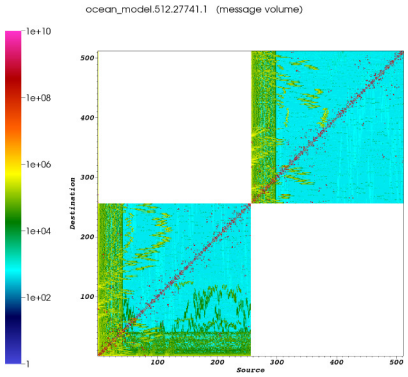


**Figure 26: MPAS-60km-30d: Point-to-point communication volume pattern with 512 MPI tasks**

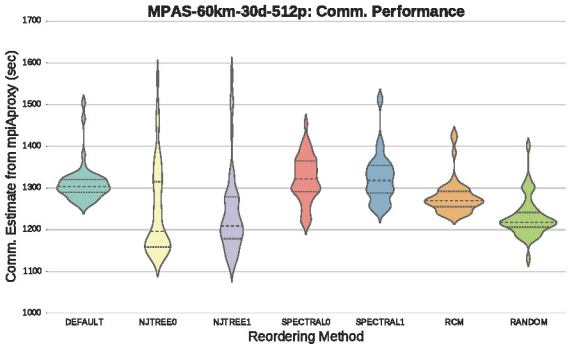


**Figure 27: Task Mapping: Communication Performance of MPAS-60km-30d with 512 MPI tasks**

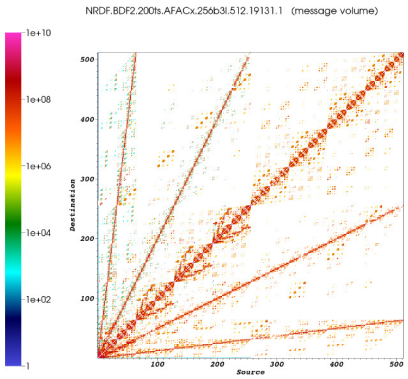


**Figure 28: NRDF-AFACx-256b3l: Point-to-point communication volume pattern with 512 MPI tasks**

## 7. CONCLUSIONS

This paper presented various task mapping algorithms that combine the insights from the application's behavior with the network topology information to provide an efficient task assignment. We developed a communication estimation tool, mpiAproxy that simulates the target application to provide a good approximation of communication

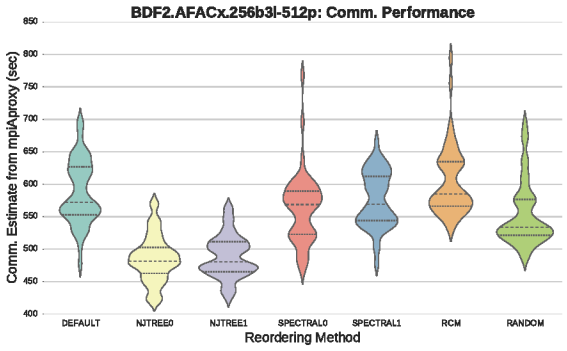


**Figure 29: Task Mapping: Communication Performance of NRDF-AFACx-256b3l with 512 MPI tasks**

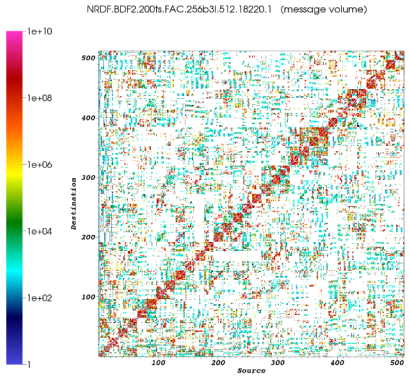


**Figure 30: NRDF-FAC-256b3l: Point-to-point communication volume pattern with 512 MPI tasks**

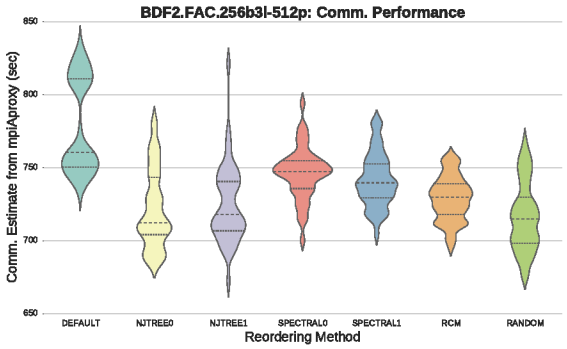


**Figure 31: Task Mapping: Communication Performance of NRDF-FAC-256b3l with 512 MPI tasks**

costs. We presented detailed communication characterization data for several scientific applications from the DOE Co-Design ecosystem as well as large scientific applications from the domains of nuclear fusion, climate and radiation diffusion. Finally, we presented a comprehensive performance evaluation (14,000 experiments) to understand the efficacy of the various task mapping techniques to optimize communication of our target application set.

The results demonstrate that our methods were able to extract significant performance improvements for a diverse pool of applications ranging from benchmarks and proxy apps to full scientific applications. We believe that the robustness of the techniques is amply reflected by the violin plots (that show the probability density of the data at different values) while accounting for system variability on modern supercomputers.

## 7.1 Future Work

We would like to extend this work to applications with multiple components that exhibit different communication patterns. For instance, the Community Earth Systems Model (CESM) comprises of several different component with vastly different communication patterns. Finding an optimal task mapping while taking the inter-component interactions into account is an open research problem.

The current implementation of our algorithms is not amenable for scaling to large core counts. We intend to address that by porting to a programming language that is more suitable for high performance computing.

During this research, we have identified several opportunities to customize algorithms specifically to a target application. Especially for NRDF, we are interested in algorithmic exploration that is intended to further reduce communication while retaining solution fidelity.

## Acknowledgments

Support for this work was provided through the Scientific Discovery through Advanced Computing (SciDAC) program funded by the U.S. Department of Energy Office of Advanced Scientific Computing Research (ASCR). Early communication characterization work was partially supported by the Oxbow project, another ASCR program. Awards of computer time was provided by the Innovative and Novel Computational Impact on Theory and Experiment (INCITE) program. This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725.

## 8. REFERENCES

[1] AMG2013 - parallel algebraic multigrid solver. https://codesign.llnl.gov/amg2013.php, 2015.

[2] BoxLibAMR- Block Structured AMR for Combusion Studies. http://exactcodesign.org/sample-page/s3dboxlib/, 2015.

[3] HPCG:High Performance Conjugate Gradients Benchmark. http://www.hpcg-benchmark.org, 2015.

[4] LULESH: Livermore Unstructured Lagrangian Explicit Shock Hydrodynamics. https://codesign.llnl.gov/lulesh.php, 2015.

[5] MCB - Monte Carlo Benchmark. https://codesign.llnl.gov/mcb.php, 2015.

[6] MPAS: Model for Prediction Across Scales. https://mpas-dev.github.io/, 2015.

[7] MultiGridC. http://exactcodesign.org/proxy-app-software/, 2015.

[8] Nek500. http://nek5000.mcs.anl.gov/, 2015.

[9] NEKBONE. https://cesar.mcs.anl.gov/content/software/thermal_hydraulics, 2015.

[10] Titan - Cray XK7 Supercomputer at Oak Ridge National Laboratory. https://www.olcf.ornl.gov/computing-resources/titan-cray-xk7/, 2015.

[11] Titan - Job Scheduling Policy. https://www.olcf.ornl.gov/kb_articles/titan-scheduling-policy/, 2015.

[12] TOP500 - Top 500 Supercomputer Sites in the World - June 2015. http://top500.org/lists/2015/06/, 2015.

[13] G. Bhanot, A. Gara, P. Heidelberger, E. Lawless, J. C. Sexton, and R. Walkup. Optimizing task layout on the Blue Gene/L supercomputer. *IBM Journal of Research and Development*, 40:489–500, 2005.

[14] A. Bhatele. *Automating topology aware mapping for supercomputers*. PhD thesis, University of Illinois at Urbana-Champaign, 2010.

[15] S. W. Bollinger and S. F. Midkiff. Processor and link assignment in multicomputers using simulated annealing. In *Proceedings of the International Conference on Parallel Processing, ICPP '88, The Pennsylvania State University, University Park, PA, USA, August 1988. Volume 1: Architecture.*, pages 1–7, 1988.

[16] CORAL Collaboration. CORAL Request for Proposal B604142, 2015.

[17] E. Cuthill and J. McKee. Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 1969 24th National Conference*, ACM '69, pages 157–172, New York, NY, USA, 1969. ACM.

[18] M. Deveci, K. Kaya, B. Uçar, and Ü. V. Çatalyürek. Fast and high quality topology-aware task mapping. In *Proceedings of 29th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2014.

[19] F. Ercal, J. Ramanujam, and P. Sadayappan. Task allocation onto a hypercube by recursive mincut bipartitioning. In *Proceedings of the 3rd Conference on Hypercube Concurrent Computers and Applications*, pages 210–221. ACM Press, 1988.

[20] A. George and J. W. Liu. *Computer Solution of Large Sparse Positive Definite*. Prentice Hall Professional Technical Reference, 1981.

[21] J. A. George. Nested dissection of a regular finite element mesh. *SIAM Journal on Numerical Analysis*, 10(2):345–363, 1973.

[22] J. L. Hintze and R. D. Nelson. Violin plots: A box plot-density trace synergism. *The American Statistician*, 52(2):181–184, May 1998.

[23] T. Hoefler and M. Snir. Generic topology mapping strategies for large-scale parallel architectures. In *Proceedings of the international conference on Supercomputing*, pages 75–84. ACM, 2011.

[24] S. C. Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3), 1967.

[25] I. Karlin, J. Keasler, and R. Neely. Lulesh 2.0 updates and changes. Technical Report LLNL-TR-641973, August 2013.

[26] B. Lee, S. McCormick, B. Philip, and D. Quinlan. Asynchronous fast adaptive composite-grid methods: numerical results. *SIAM journal on scientific computing*, 25(2), 2004.

[27] B. Lee, S. McCormick, B. Philip, and D. Quinlan. Asynchronous fast adaptive composite-grid methods for elliptic problems: Theoretical foundations. *SIAM journal on numerical analysis*, 42(1), 2005.

[28] S. McCormick and D. Quinlan. Asynchronous multilevel adaptive methods for solving partial differential equations on multiprocessors: Performance results* 1. *Parallel computing*, 12(2), 1989.

[29] S. F. McCormick. *Multilevel Adaptive Methods for Partial Differential Equations*. SIAM, Philadelphia, PA, 1989.

[30] S. F. McCormick and J. W. Thomas. The Fast Adaptive Composite grid (FAC) method for elliptic equations. *Math. Comp.*, 46:439–456, 1986.

[31] B. Philip, Z. Wang, M. Berrill, M. Birke, and M. Pernice. Dynamic implicit 3d adaptive mesh refinement for non-equilibrium radiation diffusion. *Journal of Computational Physics*, 262:17 – 37, 2014.

[32] N. Saitou and M. Nei. The neighbor-joining method: A new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.*, 4(4):406–425, 1987.

[33] R. Sankaran, J. Angel, and W. M. Brown. Genetic algorithm based task reordering to improve the performance of batch scheduled massively parallel scientific applications. *Concurrency and Computation: Practice and Experience*, 2015.

[34] K. Schloegel, G. Karypis, and V. Kumar. Parallel static and dynamic multi-constraint graph partitioning. *Concurrency and Computation: Practice and Experience*, 14(3):219–240, 2002.

[35] E. Solomonik, A. Bhatele, and J. Demmel. Improving communication performance in dense linear algebra via topology aware collectives. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '11, pages 1–11, New York, NY, USA, 2011. ACM.

[36] S. Sreepathi, M. L. Grodowitz, R. Lim, P. Taffet, P. C. Roth, J. Meredith, S. Lee, D. Li, and J. Vetter. Application Characterization Using Oxbow Toolkit and PADS Infrastructure. In *Proceedings of the 1st International Workshop on Hardware-Software Co-Design for High Performance Computing*, Co-HPC '14, pages 55–63. IEEE Press, 2014.

[37] J. Vetter and C. Chambreau. mpip: Lightweight, scalable mpi profiling, 2004. URL http://mpip.sourceforge. net.