# HPCG and HPGMG benchmark tests on Multiple Program, Multiple Data (MPMD) mode on Blue Waters - a Cray XE6/XK7 hybrid system

JaeHyuk Kwack and Gregory H Bauer
National Center for Supercomputing Applications
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
e-mail: {jkwack2, gbauer}@illinois.edu

*Abstract*— **The High Performance Conjugate Gradients (HPCG) and High Performance Geometric Multi-Grid (HPGMG) benchmarks are alternatives to the traditional LINPACK benchmark (HPL) in measuring the performance of modern HPC platforms. We performed HPCG and HPGMG benchmark tests on a Cray XE6/XK7 hybrid supercomputer, Blue Waters at National Center for Supercomputing Applications (NCSA). The benchmarks were tested on CPU-based and GPU-enabled nodes separately, and then we analyzed characteristic parameters that affect their performance. Based on our analyses, we performed HPCG and HPGMG runs in Multiple Program, Multiple Data (MPMD) mode in Cray Linux Environment in order to measure their hybrid performance on both CPU-based and GPU-enabled nodes. We observed and analyzed several performance issues during those tests. Based on lessons learned from this study, we provide recommendations about how to optimize science applications on modern hybrid HPC platforms.**

***Keywords-HPC benchmark; MPMD mode; heterogeneous computing; GPU; hybrid HPC platforms***

## I. Introduction

Since 1979, LINPACK benchmark (HPL) [1] has been used to measure the performance of HPC systems and the TOP500 list has been accordingly updated to rank the world's most powerful supercomputer. However, there have been many opinions criticizing whether HPL is a good metric to measure modern HPC performance or not. One of the main criticisms is its low memory to flop (Byte/Flop) ratio. Byte/Flop ratios of many applications in molecular dynamics, weather forecasting, astrophysics, particle physics, structural analysis and fluid dynamics are between $10^{-1}$ to 10, while the ratio of HPL benchmark is less than $10^{-3}$. Since 2014, HPC community has shared the High Performance Conjugate Gradients (HPCG) [2,3] and High Performance Geometric Multi-Grid (HPGMG) [4,5] benchmark results in order to represent modern HPC performance for many science and engineering HPC applications, at least, in terms of Byte/Flop ratios.

This study is an extension of our continuing effort [6] to find out appropriate benchmarks for modern HPCs. In this study, we employed HPCG and HPGMG benchmarks to measure the performance of the Blue Waters system located at National Center for Supercomputing Application (NCSA). Blue Waters [7,8] is a Cray XE6/XK7 hybrid supercomputer with 22,640 CPU-based XE6 nodes and 4,228 GPU-enabled XK7 nodes. The XE6 dual-socket nodes are populated with 2 AMD Interlargos model 6276 CPU processors with a nominal clock speed of at least 2.3 GHz and 64 GB of physical memory, while the XK7 accelerator nodes are equipped with one Interlagos model 6276 CPU process and one NVIDIA GK110 "Kepler" accelerator K20X with 32 GB of CPU memory and 6 GB of GPU memory. We first tested HPCG and HPGMG on dual-socket CPU-based nodes and single-socket GPU-enabled nodes separately. After analyzing numerous configurations for the optimal performance, we moved on Multiple Program, Multiple Data (MPMD) runs to evaluate the performance on CPU-based XE6 nodes and GPU-enabled XK7 nodes together. We provide performance analyses results based on MPMD runs and our recommendations for other HPC applications on modern hybrid HPC platforms.

## II. HPCG Benchmark on Blue Waters

### A. General Description

The HPCG [2,3] implementation contains crucial computational and communicational patterns present in various methods for discretization and numerical solutions of PDEs, such as dense and sparse computations, dense and sparse collectives, multi-scale execution of kernels through truncated multi-grid V cycles, and data-driven parallelism for unstructured sparse triangular solves. It synthetically discretizes an elliptic Partial Differential Equation (PDE) in 3 dimensions with zero Dirichlet boundary conditions (BCs) and a synthetic right hand side (RHS) vector. A local domain, $n_x \times n_y \times n_z$ is distributed into all available MPI ranks with a process layout $np_x \times np_y \times np_z$; as a result, the global domain size becomes $(np_x \times n_x) \times (np_y \times n_y) \times (np_z \times n_z)$. The global sparse matrix is a symmetric positive definite matrix assembled by element-wise 27-point stencil operator. It is solved via the CG iterations including a local and symmetric Gauss-Seidel pre-conditioner, which computes a forward and a backward solve with a triangular matrix. The HPCG benchmark subsequently rewards investments in high-performance collective operations, local memory system performance and low latency cooperative threading [3]. At SC16, the 6th HPCG performance list was announced. Five systems in the top 10 are GPU- or XEON Phi-based systems, while other five systems are CPU-based systems.

This paper has been submitted as an article in a special issue of Concurrency and Computation Practice and Experience on the Cray User Group 2017.

In this section, we first present HPCG benchmark results on CPU-based XE nodes or GPU-enabled XK nodes. We tested them for several different configurations in terms of local block sizes, process layouts, number of threads, and number of nodes. Based on analyses of the numerous test results, we performed MPMD runs using XE and XK nodes together for HPCG benchmark. At the end of this section, we discuss load-balancing issues between XE and XK nodes, and provide recommendations for HPC applications on hybrid systems.

### B. *HPCG on CPU-based XE nodes*

We downloaded source files of HPCG revision 3.0 from the official HPCG GitHub repository (https://github.com /hpcg-benchmark/hpcg). We first built an executable with gcc/4.9.3 and cray-mpich/7.3.0 on Blue Waters (i.e., HPCG-CPU-GitHub-GNU), and then tested it for a variety of MPI-rank and thread combinations, and for various sizes of local blocks. Based on these results, we performed a scaling test with up to 4 % of Blue Waters XE nodes with HPCG-CPU-GitHub-GNU. In addition, we built the second executable binary with intel/16.0.3.210 with cray-mpich/7.3.0 on Blue Waters (i.e., HPCG-CPU-GitHub-Intel), and then conducted an additional scaling test with the new binary.

A single XE node on Blue Waters has 16 floating-point units (FPUs) with two integer cores sharing a single FPU (i.e., total 32 integer cores in an XE node). Depending on usage rates of FPUs, some applications may optimally perform by assigning a single core for a thread, while others may perform better with a single FPU per thread. Figure 1 and Table I show HPCG-CPU-GitHub-GNU performance for multiple threads on 4 XE nodes in weak scaling for MPI rank. Each MPI rank has $104^3$ local blocks; therefore, the global domain size as well as memory usage decreases as the number of threads increases. The red line shows HPCG performance when we assigned one integer core per thread. The blue line shows results when one FPU is assigned for a thread. The case with 128 MPI ranks and 1 integer core per thread shows the best performance, though other cases with 64 MPI ranks and 1 integer core or FPU per thread show similar results to the best case. It means HPCG on Blue Waters shows an optimal performance when its OpenMP threads share a single FPU, not multiple FPUs. In other cases where each MPI rank uses multiple FPUs through threading, HPCG performance noticeably drops even on the same number of XE nodes.

Figure 2 and Table II show HPCG-CPU-GitHub-GNU performance for multiple threads on 4 XE nodes in strong scaling for MPI rank. We configured that the global domain has $832×416^2$ blocks for all cases. As number of threads increases, we assigned larger blocks to each MPI rank. All cases, consequently, used around 103 GB in memory that is approximately 40% of memory on 4 XE nodes. When each MPI used a single FPU, HPCG showed the best performance as observed in the previous tests. The performance of strong scaling tests in Figure 2 shows a little bit more drops with many threads than weak scaling tests in Figure 1, but the difference between weak and strong scaling tests was not significant.

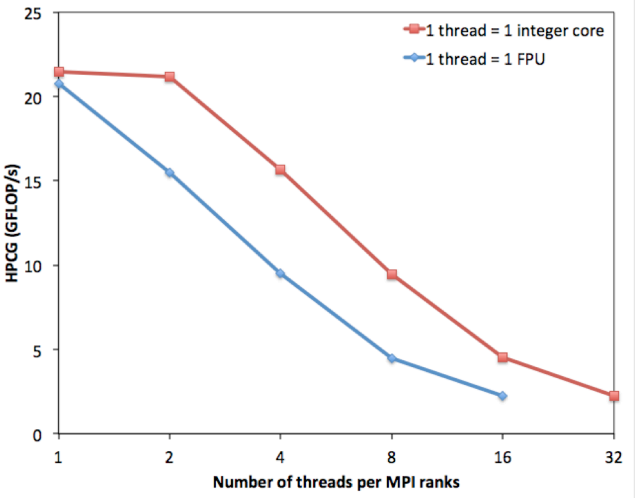

Figure 1. HPCG-CPU-GitHub-GNU for multiple threads on 4 XE nodes – weak scaling for MPI rank (the same number of equations per MPI rank)

TABLE I. HPCG-CPU-GITHUB-GNU RESULTS FOR MULTIPLE THREADS ON 4 XE NODES IN WEAK SCALING FOR MPI RANK

| Core type per thread | MPI rank | Number of threads | HPCG (GFLOP/s) | Number of Equations in total | Memory (GB) |
|---|---|---|---|---|---|
| 1 integer core per thread | 128 | 1 | 21.4798 | 143,982,592 | 102.99 |
| | 64 | 2 | 21.1808 | 71,991,296 | 51.50 |
| | 32 | 4 | 15.6547 | 35,995,648 | 25.75 |
| | 16 | 8 | 9.47234 | 17,997,824 | 12.87 |
| | 8 | 16 | 4.53633 | 8,998,912 | 6.44 |
| | 4 | 32 | 2.2442 | 4,499,456 | 3.22 |
| 1 FPU per thread | 64 | 1 | 20.7478 | 71,991,296 | 51.50 |
| | 32 | 2 | 15.5168 | 35,995,648 | 25.75 |
| | 16 | 4 | 9.50418 | 17,997,824 | 12.87 |
| | 8 | 8 | 4.45687 | 8,998,912 | 6.44 |
| | 4 | 16 | 2.24646 | 4,499,456 | 3.22 |

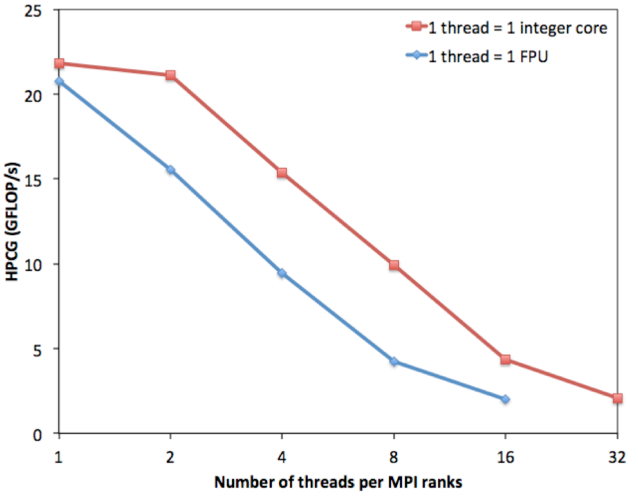

Figure 2. HPCG-CPU-GitHub-GNU for multiple threads on 4 XE nodes – strong scaling for MPI rank (the same number of equations in total)

TABLE II. HPCG-CPU-GITHUB-GNU RESULTS FOR MULTIPLE THREADS ON 4 XE NODES IN STRONG SCALING FOR MPI RANK

| Core type per thread | MPI rank | Number of threads | HPCG (GFLOP/s) | Number of Equations in total | Memory (GB) |
|---|---|---|---|---|---|
| 1 integer core per thread | 128 | 1 | 21.8334 | 143,982,592 | 102.99 |
| | 64 | 2 | 21.1425 | 143,982,592 | 102.98 |
| | 32 | 4 | 15.3969 | 143,982,592 | 102.96 |
| | 16 | 8 | 9.95385 | 143,982,592 | 102.95 |
| | 8 | 16 | 4.34153 | 143,982,592 | 102.94 |
| | 4 | 32 | 2.09014 | 143,982,592 | 102.92 |
| 1 FPU per thread | 64 | 1 | 20.74 | 143,982,592 | 102.98 |
| | 32 | 2 | 15.5413 | 143,982,592 | 102.96 |
| | 16 | 4 | 9.45932 | 143,982,592 | 102.95 |
| | 8 | 8 | 4.23567 | 143,982,592 | 102.94 |
| | 4 | 16 | 2.00099 | 143,982,592 | 102.92 |

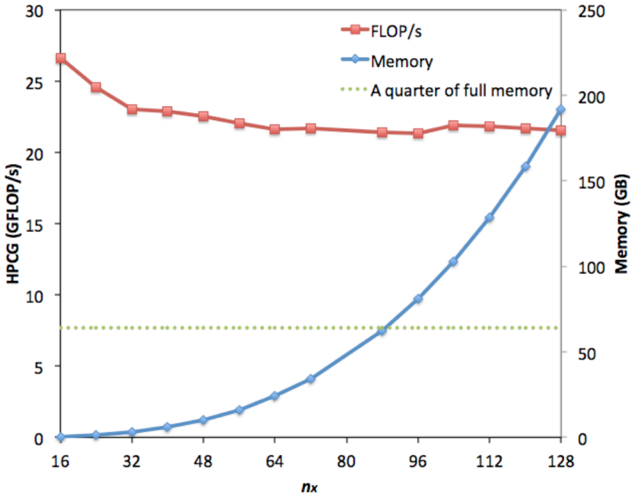

Figure 3. HPCG-CPU-GitHub-GNU performance for various local block sizes on 4 XE nodes

TABLE III. HPCG-CPU-GITHUB-GNU PERFORMANCE FOR VARIOUS LOCAL BLOCK SIZES ON 4 XE NODES

| $n_x$ | $n_y$ | $n_z$ | HPCG (GFLOP/s) | Number of equations in total | Memory (GB) |
|---|---|---|---|---|---|
| 16 | 16 | 16 | 26.5972 | 524,288 | 0.38 |
| 24 | 24 | 24 | 24.5402 | 1,769,472 | 1.27 |
| 32 | 32 | 32 | 23.0211 | 4,194,304 | 3.01 |
| 40 | 40 | 40 | 22.9008 | 8,192,000 | 5.87 |
| 48 | 48 | 48 | 22.5263 | 14,155,776 | 10.14 |
| 56 | 56 | 56 | 22.0341 | 22,478,848 | 16.09 |
| 64 | 64 | 64 | 21.6028 | 33,554,432 | 24.02 |
| 72 | 72 | 72 | 21.7075 | 47,775,744 | 34.19 |
| 88 | 88 | 88 | 21.3858 | 87,228,416 | 62.41 |
| 96 | 96 | 96 | 21.3235 | 113,246,208 | 81.01 |
| 104 | 104 | 104 | 21.8639 | 143,982,592 | 102.99 |
| 112 | 112 | 112 | 21.8343 | 179,830,784 | 128.63 |
| 120 | 120 | 120 | 21.6664 | 221,184,000 | 158.20 |
| 128 | 128 | 128 | 21.5274 | 268,435,456 | 191.98 |

Figure 3 and Table III present HPCG-CPU-GitHub-GNU performance for different local block sizes on 4 XE nodes. Since HPCG in the previous tests shows the best performance with 128 MPI ranks on 4 XE nodes, we again used the same configuration for this test. The *x*-axis shows number of local blocks along each direction (i.e., $n_x$=$n_y$=$n_z$) and *y*-axis shows HPCG results in GFLOP/s and corresponding memory usages in GB. As $n_x$ increases, the memory usage on 4 XE nodes exponentially grows (i.e., blue line in Figure 3). However, HPCG performance (i.e., red line in Figure 3) keeps constant after sharp drops for cases with very small numbers of equations. The green dotted line in Figure 3 shows a quarter of full memory on 4 XE nodes that is the required minimum memory usage for official HPCG results. Through this test, it turns out that local block size does not significantly affect HPCG performance once it satisfies the official HPCG requirement for memory usage.

TABLE IV. SCALING TEST FOR HPCG-CPU-GITHUB-GNU WITH UP TO 3.6% BLUE WATERS XE NODES

| XEs | MPI ranks | HPCG (GFLOP/s) | Number of equations in total | Memory (GB) | Efficiency (%) |
|---|---|---|---|---|---|
| 1 | 32 | 5.43104 | 35,995,648 | 25.75 | 100 |
| 2 | 64 | 11.355 | 71,991,296 | 51.50 | 105 |
| 4 | 128 | 21.9875 | 143,982,592 | 102.99 | 101 |
| 8 | 256 | 43.5148 | 287,965,184 | 205.99 | 100 |
| 16 | 512 | 86.6853 | 575,930,368 | 411.97 | 100 |
| 32 | 1024 | 170.412 | 1,151,860,736 | 823.94 | 98 |
| 50 | 1600 | 268.921 | 1,799,782,400 | 1,287.40 | 99 |
| 100 | 3200 | 538.826 | 3,599,564,800 | 2574.81 | 99 |
| 200 | 6400 | 1057.59 | 7,199,129,600 | 5149.62 | 97 |
| 400 | 12800 | 2154.29 | 14,398,259,200 | 10299.2 | 99 |
| 800 | 25600 | 4137.76 | 28,796,518,400 | 20598.50 | 95 |

TABLE V. SCALING TEST FOR HPCG-CPU-GITHUB-INTEL WITH UP TO 4.6% BLUE WATERS XE NODES

| XEs | MPI ranks | HPCG (GFLOP/s) | Number of equations in total | Memory (GB) | Efficiency (%) |
|---|---|---|---|---|---|
| 1 | 32 | 7.02902 | 35,995,648 | 25.75 | 100 |
| 2 | 64 | 13.0688 | 71,991,296 | 51.50 | 93 |
| 4 | 128 | 27.9278 | 143,982,592 | 102.99 | 99 |
| 8 | 256 | 55.3573 | 287,965,184 | 205.99 | 98 |
| 16 | 512 | 104.433 | 575,930,368 | 411.97 | 93 |
| 32 | 1024 | 218.144 | 1,151,860,736 | 823.94 | 97 |
| 64 | 2048 | 435.228 | 2,303,721,472 | 1,647.88 | 97 |
| 128 | 4096 | 817.343 | 4,607,442,944 | 3,295.75 | 91 |
| 256 | 8192 | 1646.64 | 9,214,885,888 | 6591.51 | 92 |
| 512 | 16384 | 3103.43 | 18,429,771,776 | 13183 | 86 |
| 1024 | 32768 | 6218.57 | 36,859,543,552 | 26366 | 86 |

We chose $n_x = n_y = n_z = 104$ for a scaling test for HPCG-CPU-GitHub-GNU, and conducted it with up to 800 XE nodes (i.e., around 3.6% of Blue Waters XE nodes). Based on our previous test results, we assigned 32 MPI ranks per node (i.e., 1 MPI rank to 1 integer core); therefore, number of MPI ranks used in this test was from 32 to 25,600. The memory usage was around 40% of assigned XE nodes. As shown in Table IV, HPCG-CPU-GitHub-GNU shows an impressive scaling on Blue Waters, and it keeps more than or equal to 95% of parallel efficiency in all of the cases.

We conducted an additional scaling test for HPCG-CPU-GitHub-Intel with up to 1,024 XE nodes (i.e., around 4.6%

of Blue Waters XE nodes) with the same configuration (i.e., $n_x = n_y = n_z = 104$ and 32 MPI/node). The range of MPI ranks is from 32 to 32,768 as shown in Table V. The parallel efficiency of HPCG-CPU-GitHub-Intel is more than or equal to 86%, which is lower than results of HPCG-CPU-GitHub-GNU. The overall performance of HPCG-CPU-GitHub-Intel, however, is much better than the performance of HPCG-CPU-GitHub-GNU, as compared in Figure 4.

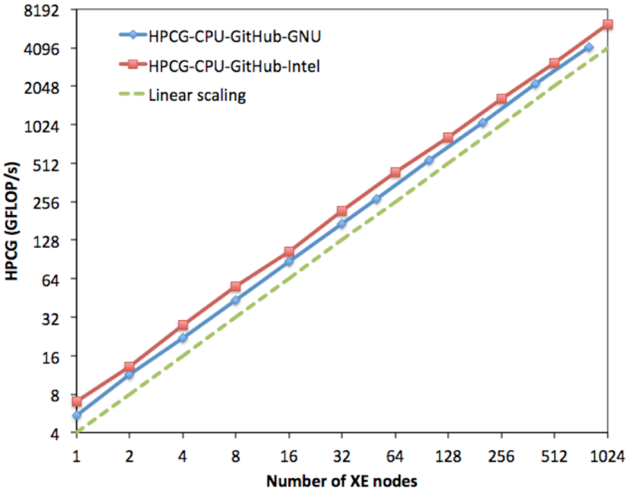

Figure 4. Scaling tests for HPCG-CPU-GitHubs up to 4.6% Blue Waters XE nodes

### C. *HPCG on GPU-enabled XK nodes*

The official HPCG webpage (i.e., www.hpcg-benchmark.org) provides HPCG 3.0 binary from NVIDIA optimized for NVIDIA GPUs. It requires CUDA and OpenMPI 1.6.5 libraries. However, in [6], it turned out that executable binaries built for the OpenMPI library could not perform optimally on Blue Waters. In this study, we instead employed a Cray-MPICH compatible HPCG binary provided by NIVIDA (i.e., HPCG-GPU-NVIDIA).

Each XK node on Blue Waters has one CPU processor with 32GB memory and one K20X GPU with 6 GB device memory. In this section, we assigned one MPI rank per GPU; as a result, one MPI rank was assigned to each node. Since GPUs do most of computations for HPCG, we kept an eye on GPU memory capacity for assigned XK nodes, instead of CPU memory capacity. Figure 5 and Table VI show HPCG-GPU-NVIDIA performance for various local block sizes on 8 XK nodes. The *x*-axis of Figure 5 is a local cubic block size along each direction (i.e., $n_x$, $n_y$, and $n_z$). As it grows, the memory used for data exponentially increases (i.e., red line in Figure 5). The green dotted line in Figure 5 shows the minimum required memory usage for the official HPCG results that is a quarter of GPU memory of 8 XK nodes. The blue line in Figure 5 shows HPCG results. The HPCG performance linearly increases from $24^3$ to $88^3$ of local blocks, since communication overheads between CPU and GPU in these cases rapidly tails off. After that, HPCG results look converged into some range. One interesting observation is that memory used for cases with $208^3$ and $216^3$ local cubes are larger than the total GPU memory of 8 XK nodes (i.e., 48 GB). It seems HPCG-GPU-NVIDIA somehow re-uses GPU memory during the computation. For the larger local domain than $216^3$, the out-of-memory error occurred. In summary, HPCG-GPU-NVIDIA shows the most optimal performance with $128^3$, $168^3$, and $176^3$ local blocks on 8 XK nodes.

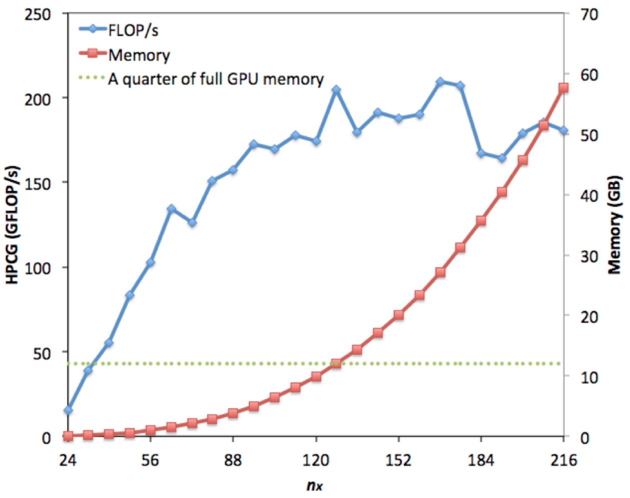

Figure 5. HPCG-GPU-NVIDIA for various local block sizes on 8 XK nodes

TABLE VI. HPCG-GPU-NVIDIA PERFORMANCE FOR VARIOUS LOCAL BLOCK SIZES ON 8 XK NODES

| $n_x$ | $n_y$ | $n_z$ | HPCG (GFLOP/s) | Number of equations in total | Memory (GB) |
|---|---|---|---|---|---|
| 24 | 24 | 24 | 15.5542 | 110,592 | 0.079 |
| 32 | 32 | 32 | 38.661 | 262,144 | 0.188 |
| 40 | 40 | 40 | 55.1535 | 512,000 | 0.367 |
| 48 | 48 | 48 | 83.3617 | 884,736 | 0.634 |
| 56 | 56 | 56 | 102.637 | 1,404,928 | 1.006 |
| 64 | 64 | 64 | 134.636 | 2,097,152 | 1.501 |
| 72 | 72 | 72 | 126.136 | 2,985,984 | 2.137 |
| 80 | 80 | 80 | 150.928 | 4,096,000 | 2.931 |
| 88 | 88 | 88 | 157.044 | 5,451,776 | 3.900 |
| 96 | 96 | 96 | 172.29 | 7,077,888 | 5.063 |
| 104 | 104 | 104 | 169.535 | 8,998,912 | 6.437 |
| 112 | 112 | 112 | 177.864 | 11,239,424 | 8.039 |
| 120 | 120 | 120 | 174.563 | 13,824,000 | 9.887 |
| 128 | 128 | 128 | 204.912 | 16,777,216 | 11.999 |
| 136 | 136 | 136 | 179.439 | 20,123,648 | 14.392 |
| 144 | 144 | 144 | 191.406 | 23,887,872 | 17.083 |
| 152 | 152 | 152 | 187.779 | 28,094,464 | 20.091 |
| 160 | 160 | 160 | 190.252 | 32,768,000 | 23.432 |
| 168 | 168 | 168 | 209.338 | 37,933,056 | 27.125 |
| 176 | 176 | 176 | 207.152 | 43,614,208 | 31.187 |
| 184 | 184 | 184 | 167.078 | 49,836,032 | 35.635 |
| 192 | 192 | 192 | 164.078 | 56,623,104 | 40.487 |
| 200 | 200 | 200 | 178.81 | 64,000,000 | 45.761 |
| 208 | 208 | 208 | 185.213 | 71,991,296 | 51.474 |
| 216 | 216 | 216 | 180.76 | 80,621,568 | 57.644 |

With $128^3$, $168^3$ and $176^3$ local blocks, we carried out scaling tests with up to 6.1% Blue Water XK nodes (i.e., 256 XK nodes). Table VII shows HPCG results in GFLOP/s and parallel efficiency in percentile for three cases. Figure 6 shows plots for number of XK nodes versus HPCG results for $128^3$, $168^3$ and $176^3$ local blocks. The purple dotted line describes a linear scaling. The case with $168^3$ local blocks shows the best HPCG performance on 8, 16 and 32 XK nodes, while the case with $128^3$ local blocks was the best on other numbers of XK nodes. The parallel efficiency is more than or equal to 83%, which is worse than the parallel efficiency of HPCG-CPU-GitHub-Intel (i.e., 92% with 256 XE nodes). We think this happens because of two reasons:

- Additional communicational overhead between the host (i.e., CPU) and the device (i.e., GPU) on XK nodes reduces the network performance for MPI communications.
- Higher HPCG performance of a single XK node (i.e., 27 GFLOP/s/node for GPU version in Table VII) than of a single XE node (i.e., 7 GFLOP/s/node for CPU version in Table V).

TABLE VII. SCALING TEST FOR HPCG-GPU-NVIDIA WITH UP TO 6.1% BLUE WATERS XK NODES

| XKs | MPI ranks | $n_x$=$n_y$=$n_z$= 128 | | $n_x$=$n_y$=$n_z$= 168 | | $n_x$=$n_y$=$n_z$= 176 | |
|---|---|---|---|---|---|---|---|
| | | *HPCG (GFLOP/s)* | *Efficiency (%)* | *HPCG (GFLOP/s)* | *Efficiency (%)* | *HPCG (GFLOP/s)* | *Efficiency (%)* |
| 1 | 1 | 27.029 | 100 | 23.544 | 100 | 23.734 | 100 |
| 2 | 2 | 53.353 | 99 | 46.811 | 99 | 48.960 | 103 |
| 4 | 4 | 103.84 | 96 | 93.085 | 99 | 95.674 | 101 |
| 8 | 8 | 204.87 | 95 | 209.33 | 111 | 207.19 | 109 |
| 16 | 16 | 397.87 | 92 | 415.67 | 110 | 412.30 | 109 |
| 32 | 32 | 790.06 | 91 | 827.76 | 110 | 692.05 | 91 |
| 64 | 64 | 1493.0 | 86 | 1347.1 | 89 | 1365.9 | 90 |
| 128 | 128 | 2941.0 | 85 | 2741.2 | 91 | 2724.6 | 90 |
| 256 | 256 | 5756.2 | 83 | 5365.2 | 89 | 5359.8 | 88 |

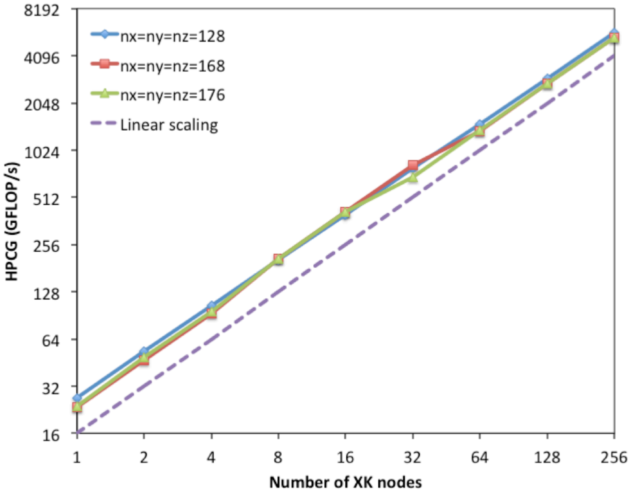

Figure 6. Scaling tests for HPCG-GPU-NVIDIA up to 6.1% Blue Waters XK nodes

### *D. HPCG on CPU-based XE and GPU-enabled XK nodes on MPMD mode*

For Multiple Program, Multiple Data (MPMD) mode, binary combinations for CPU-based XE nodes and GPU-enable XK nodes should be built from the same base code. In this section, we used two new binaries provided by NVIDIA: HPCG-MPMD-CPU for XE nodes and HPCG-MPMD-GPU for XK nodes.

Since the base code is slightly different from our previous studies on XE or XK node, we briefly performed a series of single node test for different local block sizes (e.g., $72^3$, $104^3$, $128^3$ and $128\times64^2$) and various combinations of MPI ranks and threads (e.g., 1 to 32 MPIs with a thread per an integer core on XE, and 1 to 16 MPIs on XK). Table VIII and Figure 7 shows HPCG performance of HPCG-MPMD-CPU on a single XE node and HPCG-MPMD-GPU on a single XK node. Because of different memory capacity of CPUs (i.e., 64 GB) on XE and GPU (i.e., 6 GB) on XK, some cases with a large number of MPI ranks and big local blocks did not complete due to the out-of-memory error. The best XE node performance was 8.3 GFLOP/s with $128\times64^2$ local blocks when 16 MPI ranks with 2 threads were assigned on the XE node. The best XK node performance was 22.8 GFLOP/s with $128\times64^2$ local blocks when 14 MPI ranks are assigned to the XK node. According to comparison of the best performances, HPCG on an XK node is 2.7 times faster than HPCG on an XE node. This difference can result in a serious load imbalance between XE and XK nodes; as a result, the performance on MPMD mode can be degraded.

Our approach to minimize the load imbalance between CPU-only XE and GPU-enable XK nodes is to synchronize XE nodes' HPCG performance per MPI rank with XK nodes'. Figure 8 presents HPCG performance per MPI rank of HPCG-MPMD-CPU on an XE node and HPCG-MPMD-GPU on an XK node. As number of MPI ranks increases, HPCG performance per MPI on the XK node linearly decreases. On the XE node, HPCG performance per MPI rank linearly increases as number of threads increase up to 8 threads per MPI rank.

TABLE VIII. HPCG PERFORMANCE OF HPCG-MPMD-CPU AND HPCG-MPMD-GPU ON A SINGLE XE OR XK NODE(UNIT: GFLOP/S)

| Node type | MPI ranks | Threads | nx=ny= nz=72 | nx=ny= nz=104 | nx=ny= nz=128 | nz=128, ny=nz=64 |
|---|---|---|---|---|---|---|
| XE | 1 | 32 | 1.0335 | 1.13907 | 1.39787 | 1.43499 |
| | 2 | 16 | 2.28724 | 2.2585 | 2.76631 | 2.98869 |
| | 4 | 8 | 5.3681 | 5.78568 | 7.21688 | 7.14705 |
| | 8 | 4 | 5.94444 | 6.4733 | 7.73771 | 7.88999 |
| | 16 | 2 | 7.04848 | 6.74828 | 8.02983 | 8.32372 |
| | 32 | 1 | 6.69432 | 6.31722 | OOM(1) | 7.97788 |
| XK | 1 | 1 | 12.7187 | 17.2333 | 22.9948 | 17.2444 |
| | 2 | 1 | 15.2734 | 18.4187 | 24.5339 | 20.31 |
| | 4 | 1 | 15.9005 | 18.9973 | 24.5516 | 21.899 |
| | 6 | 1 | 17.0141 | 19.2494 | OOM(1) | 22.3401 |
| | 8 | 1 | 16.0738 | OOM(1) | OOM(1) | 22.3905 |
| | 10 | 1 | 17.1923 | OOM(1) | OOM(1) | 22.7505 |
| | 12 | 1 | 17.3912 | OOM(1) | OOM(1) | 22.5492 |
| | 14 | 1 | 17.2224 | OOM(1) | OOM(1) | 22.7725 |
| | 16 | 1 | 17.4242 | OOM(1) | OOM(1) | OOM(1) |

(1) OOM : Out of Memory

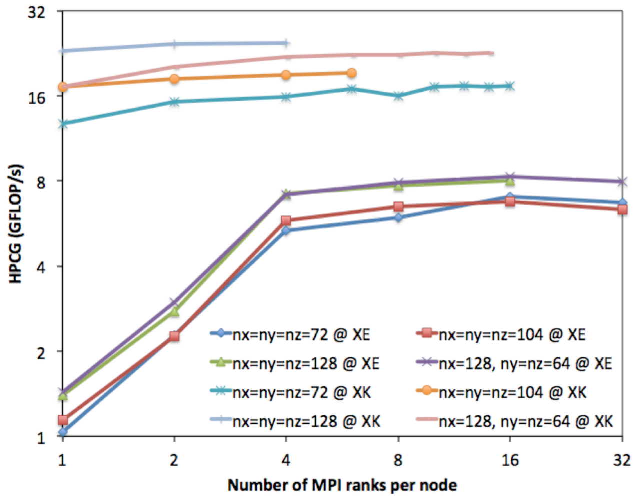

Figure 7. HPCG performance of HPCG-MPMD-CPU and HPCG-MPMD-GPU on a single XE or XK node

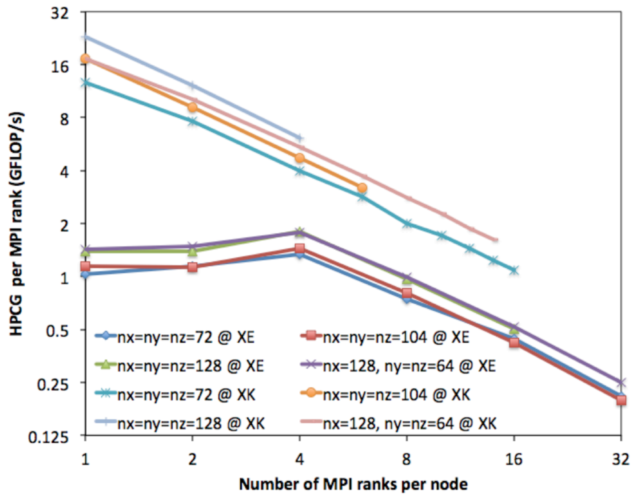

Figure 8. HPCG performance per MPI rank of HPCG-MPMD-CPU and HPCG-MPMD-GPU on a single XE or XK node

Before moving on MPMD runs, we performed scalability tests for HPCG-MPMD-CPU and HPCG-MPMD-GPU with 128×64$^2$ local block per MPI rank, as presented in Table IX. We considered two different combinations of MPI rank and thread on XE and three different MPI ranks per GPU on XK. In Figure 9, all of the cases show good scalability parallel to ideal linear scaling (i.e., orange dotted line). All cases on XK nodes performed similarly to each other except that results with 12 MPI/GPU on 16 and 32 XK nodes were relatively lower than others. On XE nodes, the case with 2 thread/MPI was 14 to 20% faster than the case with 8 thread/MPI. Figure 9 also shows that a XK node is approximately 4 times faster than an XE node, consistent with the ratios of XK to XE peak FLOPs and memory bandwidths.

In order to minimize load imbalance between XE and XK nodes during MPMD runs, we want to synchronize per-MPI performance on XE nodes with that on XK nodes. Table X and Figure 10 show HPCG performance per MPI rank of HPCG-MPMD-CPU and HPCG-MPMD-GPU. The per-MPI performance of 8 thread/MPI on XE nodes (i.e., 1.6 to 1.8 GFLOP/s) is much closer to cases of 10MPI/GPU (i.e., 2.0 to 2.3 GFLOP/s) and 12 MPI/GPU (i.e., 1.4 to 1.9 GFLOP/s) on XK nodes than the per-MPI performance of 2 thread/MPI on XE nodes (i.e., 0.45 to 0.52 GFLOP/s); therefore, we expect that 8 thread/MPI on XE nodes would perform more optimally for MPMD runs than 2 thread/MPI on XE nodes. Since the per-MPI performance of 12 MPI/GPU on XK nodes (i.e., 2.4 to 2.8 GFLOP/s) is too high to be synchronized with 8 thread/MPI on XE nodes, we did not consider it for further HPCG benchmarking on MPMD mode.

TABLE IX. HPCG PERFORMANCE OF HPCG-MPMD-CPU AND HPCG-MPMD-GPU WITH UP TO 128 XE OR XK NODES(UNIT: GFLOP/S)

| **Number of nodes** | **HPCG-HPMD-CPU** | | **HPCG-MPMD-GPU** | | |
|---|---|---|---|---|---|
| | ***2thr/mpi*[1]** | ***8thr/mpi*** | ***8mpi/gpu*[2]** | ***10mpi/gpu*** | ***12mpi/gpu*** |
| 1 | 8.31327 | 7.14167 | 22.2208 | 22.5616 | 22.3774 |
| 2 | 16.2669 | 14.1543 | 43.985 | 44.5186 | 44.4957 |
| 4 | 32.4941 | 28.0463 | 87.3184 | 88.2059 | 86.0493 |
| 8 | 65.1978 | 54.7415 | 170.352 | 175.311 | 170.902 |
| 16 | 124.731 | 108.783 | 339.408 | 338.977 | 282.692 |
| 32 | 248.387 | 216.612 | 648.375 | 692.816 | 525.783 |
| 64 | 493.589 | 409.541 | 1266.32 | 1273.18 | 1173.86 |
| 128 | 930.88 | 812.448 | 2498.76 | 2529.21 | 2245.37 |

(1) thr/mpi: number of threads per MPI rank, (2) mpi/gpu: number of MPI ranks per GPU

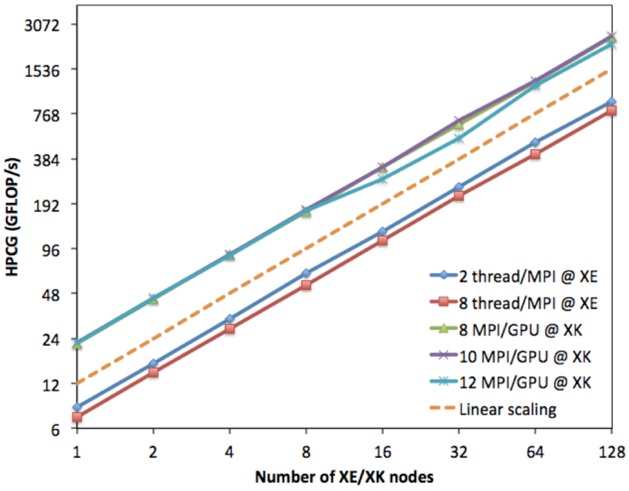

Figure 9. HPCG performance of HPCG-MPMD-CPU and HPCG-MPMD-GPU with up to 128 XE or XK nodes

TABLE X. HPCG PERFORMANCE PER MPI OF HPCG-MPMD-CPU AND HPCG-MPMD-GPU WITH UP TO 128 XE/XK NODES(UNIT: GFLOP/S)

| **Number of nodes** | **HPCG-HPMD-CPU** | | **HPCG-MPMD-GPU** | | |
|---|---|---|---|---|---|
| | ***2thr/mpi*[1]** | ***8thr/mpi*** | ***8mpi/gpu*[2]** | ***10mpi/gpu*** | ***12mpi/gpu*** |
| 1 | 0.5196 | 1.7854 | 2.7776 | 2.2562 | 1.8648 |
| 2 | 0.5083 | 1.7693 | 2.7491 | 2.2259 | 1.8540 |
| 4 | 0.5077 | 1.7529 | 2.7287 | 2.2051 | 1.7927 |
| 8 | 0.5094 | 1.7107 | 2.6618 | 2.1914 | 1.7802 |
| 16 | 0.4872 | 1.6997 | 2.6516 | 2.1186 | 1.4724 |
| 32 | 0.4851 | 1.6923 | 2.5327 | 2.1651 | 1.3692 |
| 64 | 0.4820 | 1.5998 | 2.4733 | 1.9893 | 1.5285 |
| 128 | 0.4545 | 1.5868 | 2.4402 | 1.9759 | 1.4618 |

(1) thr/mpi: number of threads per MPI rank, (2) mpi/gpu: number of MPI ranks per GPU

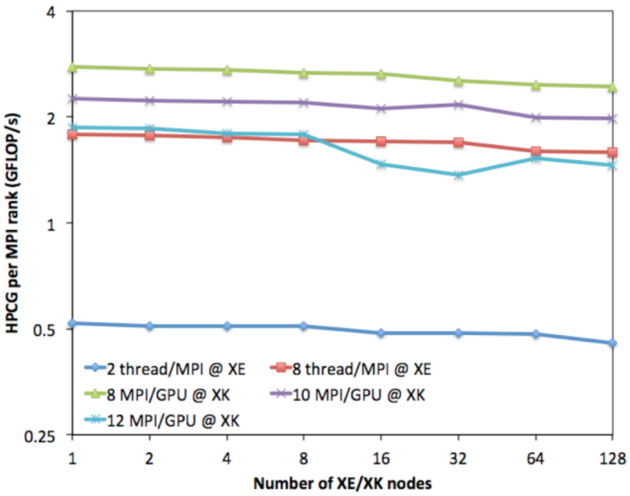

Figure 10. HPCG performance per MPI rank of HPCG-MPMD-CPU and HPCG-MPMD-GPU with up to 128 XE or XK nodes

We considered four cases for HPCG performance test on MPMD mode with up to 128 XE nodes and 128 XK nodes. Case I was tested with 8 thread/MPI on XE and 10 MPI/GPU on XK. In Cases II and III, we assigned 2 thread/MPI on XE with 10 MPI/GPU on XK and 12 MPI/GPU on XK, respectively. As a reference, the case IV used only CPUs on XE (i.e., 2 CPUs/node) and XK nodes (i.e., 1 CPU/node) with 2 thread/MPI. In Table XI and Figure 11, case I shows the best HPCG performance that is around 60% to 100% faster than other cases.

Table XII shows their per-MPI performance in GFLOP/s. The per-MPI performance of case I is similar to the per-MPI performance of 8 thread/MPI on XE nodes in Table X; besides, the per-MPI performance of cases II, III and IV is similar to the per-MPI performance of 2 thread/MPI on XE nodes in Table X. Since the per-MPI performance of GPUs was better than the per-MPI performance of CPUs in all cases, the overall performance was always bounded by the CPU performance. Due to the limited size of GPU memory, we could not assign more MPI rank per GPU in order to lower the per-MPI performance on XK nodes. In addition, we could not assign more threads per MPI ranks on XE node, because of the limited threading performance between NUMA cores on XE nodes. The followings are summary of HPCG performance on the MPMD mode:

- The overall HPCG performance on the MPMD mode is determined by the minimum per-MPI performance on XE and XK nodes.
- The HPCG performance on XK nodes is around 2.7 times higher than that on XE nodes.
- In order to decrease the per-MPI performance on XK nodes, multiple MPI ranks is assigned to GPUs, but the number of MPI ranks is bounded by the memory size of GPUs (i.e., 6 GB/GPU).
- In order to improve the per-MPI performance on XE nodes, multiple threads is assigned to CPUs; however, the maximum number of threads is bounded by the number of integer cores in NUMA core on XE nodes (i.e., 8 threads/MPI)
- As a result, the MPMD run with the combination of 8 thread/MPI on XE and 10 MPI/GPU on XK shows an optimal performance on Blue Waters.

TABLE XI. HPCG PERFORMANCE OF MPMD RUNS WITH UP TO 128 XE AND 128 XK NODES (UNIT: GFLOP/S)

| XE nodes | XK nodes | Case I[(1)] | Case II[(2)] | Case III[(3)] | Case IV[(4)] |
|---|---|---|---|---|---|
| 1 | 1 | 24.0831 | | 14.2045 | 12.2138 |
| 2 | 2 | 46.7258 | 26.6157 | 28.2871 | 24.3066 |
| 4 | 4 | 92.937 | 53.0355 | 55.8481 | 48.7609 |
| 8 | 8 | 182.796 | 105.505 | 106.092 | 93.1538 |
| 16 | 16 | 350.558 | 201.75 | 208.912 | 159.893 |
| 32 | 32 | 693.168 | 402.937 | 417.709 | 371.003 |
| 64 | 64 | 1366.53 | 803.817 | 815.194 | 725.243 |
| 128 | 128 | 2599.56 | 1571.16 | 1609.91 | 1461.7 |

(1) 8 thread/MPI @ XE + 10 MPI/GPU @ XK, (2) 2 thread/MPI @ XE + 10 MPI/GPU @ XK, (3) 2 thread/MPI @ XE + 12 MPI/GPU @ XK, (4) 2 thread/MPI @ XE and XK w/o GPU

Gray cell: HPCG could not initialize the global domain with the given number of MPI ranks.

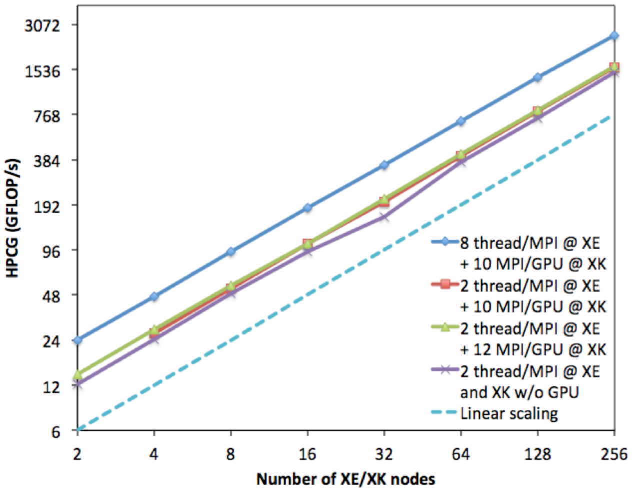

Figure 11. HPCG performance of MPMD runs with up to 128 XE nodes and 128 XK nodes

TABLE XII. HPCG PERFORMANCE PER MPI RANK OF MPMD RUNS WITH UP TO 128 XE AND 128 XK NODES (UNIT: GFLOP/S)

| XE nodes | XK nodes | Case I[(1)] | Case II[(2)] | Case III[(3)] | Case IV[(4)] |
|---|---|---|---|---|---|
| 1 | 1 | 1.7202 | | 0.5073 | 0.5089 |
| 2 | 2 | 1.6688 | 0.5118 | 0.5051 | 0.5064 |
| 4 | 4 | 1.6596 | 0.5100 | 0.4986 | 0.5079 |
| 8 | 8 | 1.6321 | 0.5072 | 0.4736 | 0.4852 |
| 16 | 16 | 1.5650 | 0.4850 | 0.4663 | 0.4164 |
| 32 | 32 | 1.5473 | 0.4843 | 0.4662 | 0.4831 |
| 64 | 64 | 1.5251 | 0.4831 | 0.4549 | 0.4722 |
| 128 | 128 | 1.4506 | 0.4721 | 0.4492 | 0.4758 |

(1) 8 thread/MPI @ XE + 10 MPI/GPU @ XK, (2) 2 thread/MPI @ XE + 10 MPI/GPU @ XK, (3) 2 thread/MPI @ XE + 12 MPI/GPU @ XK, (4) 2 thread/MPI @ XE and XK w/o GPU

Gray cell: HPCG could not initialize the global domain with the given number of MPI ranks.

## III. HPGMG BENCHMARK ON BLUE WATERS

### A. *General Description*

HPGMG [4,5] is another effort for HPC performance benchmarking based on geometric multi-grid methods with emphasis on community-driven development process, long-term durability, scale-free specification and scale-free communication. It provides two implementations, Finite Element (HPGMG-FE) and Finite Volume (HPGMG-FV) implementations; HPGMG-FE is compute-intensive and cache-intensive, while HPGMG-FV is memory bandwidth-intensive. HPGMG-FV has been used for the official list, and it solves an elliptic problem on isotropic Cartesian grids with fourth-order accuracy in the max norm. It calculates a flux term on each of the 6 faces on every cell in the entire domain. The fourth-order implementation [5] requires 4 times the floating-point operations, 3 times the MPI messages per smoother and 2 times the MPI message size without additional DRAM data movement compared to the second-order implementation [4] proposed originally. The solution process employs the Full Multi-grid (FMG) F-cycle that is a series of progressively deeper geometric multi-grid V-cycles. There are several dozen stencils that vary in shape and size, sweep per step, and the grid sizes vary exponentially. Coarse grid solution process can occur on a single core of a single node, and then the coarse grid solution is propagated to every thread in the system. At SC16, the 5th HPGMG-FV performance list was announced and it has two GPU-enabled systems. In the list, those GPU-enabled systems provide GPU-based and CPU-based performance separately.

In this section, we first performed HPGMG tests on CPU-based or GPU-enabled nodes on Blue Waters, and then analyzed several major components that affected the performance. We then conducted MPMD runs on CPU-based and GPU-enabled nodes together for HPGMG. We provide our analyses and recommendations for other geometric multi-grid applications on hybrid HPC platforms.

### B. *HPGMG on CPU-based XE nodes*

We downloaded HPGMG-FV source files from the official HPGMG Bitbucket repository (i.e., https://bitbucket.org/hpgmg/hpgmg/). We used them (i.e., commit: 8a2f0e1) to build a HPGMG binary for CPU-based XE nodes (i.e., HPGMG-CPU-8a2f0e1) with gcc/4.9.3 and cray-mpich/7.3.3 on Blue Waters. HPGMG-FV requires two input parameters

TABLE XIII. HPGMG PERFORMANCE ON 16 XE NODES (UNIT: GDOF/S)

| nthr | k | nB | | | | | | | | | | Max |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | *1* | *2* | *3* | *4* | *5* | *6* | *7* | *8* | *16* | *32* | |
| 1 | 4 | 0.0820 | 0.0956 | 0.0596 | 0.1489 | 0.1514 | 0.1367 | 0.1369 | 0.2007 | | | 0.4756 |
| | 5 | 0.2435 | 0.2517 | 0.1977 | 0.2931 | 0.2924 | 0.2864 | 0.2873 | 0.3320 | | | |
| | 6 | 0.3788 | 0.3731 | 0.3396 | 0.3741 | 0.3730 | 0.3826 | 0.3833 | 0.4127 | | | |
| | 7 | 0.4445 | 0.4344 | 0.4044 | 0.4131 | 0.4131 | 0.4293 | 0.4297 | 0.4588 | | | |
| | 8 | 0.4756 | OOM[1] | OOM | OOM | OOM | OOM | OOM | OOM | | | |
| | 9 | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | | | |
| 2 | 4 | 0.0453 | 0.0997 | 0.0696 | 0.1111 | 0.1111 | 0.0823 | 0.1608 | 0.1606 | | | 0.4804 |
| | 5 | 0.1623 | 0.2664 | 0.2184 | 0.2712 | 0.2711 | 0.2341 | 0.3134 | 0.3147 | | | |
| | 6 | 0.3109 | 0.4005 | 0.3670 | 0.3914 | 0.3915 | 0.3634 | 0.4081 | 0.4083 | | | |
| | 7 | 0.3768 | 0.4508 | 0.4269 | 0.4412 | 0.4410 | 0.4129 | 0.4490 | 0.4488 | | | |
| | 8 | 0.4041 | 0.4804 | OOM | OOM | OOM | OOM | OOM | OOM | | | |
| | 9 | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | | | |
| 4 | 4 | 0.0216 | 0.0298 | 0.0414 | 0.0726 | 0.0739 | 0.0540 | 0.0541 | 0.0872 | | | 0.4496 |
| | 5 | 0.0975 | 0.1216 | 0.1505 | 0.2177 | 0.2177 | 0.1817 | 0.1813 | 0.2322 | | | |
| | 6 | 0.2633 | 0.2680 | 0.2979 | 0.3653 | 0.3649 | 0.3347 | 0.3345 | 0.3613 | | | |
| | 7 | 0.3893 | 0.3503 | 0.3719 | 0.4271 | 0.4270 | 0.4039 | 0.4040 | 0.4185 | | | |
| | 8 | 0.4288 | 0.3765 | 0.3978 | 0.4493 | 0.4496 | OOM | OOM | OOM | | | |
| | 9 | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | | | |
| 8 | 4 | 0.0141 | 0.0159 | 0.0160 | 0.0214 | 0.0214 | 0.0346 | 0.0346 | 0.0581 | | | 0.4494 |
| | 5 | 0.0634 | 0.0733 | 0.0733 | 0.0919 | 0.0921 | 0.1278 | 0.1279 | 0.1826 | | | |
| | 6 | 0.2103 | 0.2237 | 0.2238 | 0.2377 | 0.2377 | 0.2776 | 0.2778 | 0.3416 | | | |
| | 7 | 0.3770 | 0.3728 | 0.3728 | 0.3444 | 0.3443 | 0.3720 | 0.3720 | 0.4267 | | | |
| | 8 | 0.4346 | 0.4230 | 0.4232 | 0.3744 | 0.3745 | 0.3984 | 0.3983 | 0.4494 | | | |
| | 9 | 0.4352 | OOM | OOM | OOM | OOM | OOM | OOM | OOM | | | |
| 16 | 4 | 0.0028 | 0.0081 | 0.0081 | 0.0103 | 0.0102 | 0.0103 | 0.0137 | 0.0137 | | | 0.3335 |
| | 5 | 0.0147 | 0.0380 | 0.0379 | 0.0486 | 0.0485 | 0.0484 | 0.0633 | 0.0637 | | | |
| | 6 | 0.0701 | 0.1319 | 0.1318 | 0.1465 | 0.1463 | 0.1463 | 0.1695 | 0.1694 | | | |
| | 7 | 0.1873 | 0.2597 | 0.2600 | 0.2615 | 0.2615 | 0.2614 | 0.2733 | 0.2732 | 0.3003 | 0.3006 | |
| | 8 | 0.2644 | 0.3239 | 0.3241 | 0.3160 | 0.3161 | 0.3161 | 0.3167 | 0.3168 | 0.3335 | OOM | |
| | 9 | 0.2706 | 0.3219 | 0.3221 | OOM | OOM | OOM | OOM | OOM | | | |
| 32 | 4 | 0.0012 | 0.0015 | 0.0015 | 0.0060 | 0.0051 | 0.0051 | 0.0051 | 0.0064 | | | 0.2495 |
| | 5 | 0.0056 | 0.0080 | 0.0081 | 0.0244 | 0.0245 | 0.0244 | 0.0244 | 0.0317 | | | |
| | 6 | 0.0258 | 0.0410 | 0.0411 | 0.0920 | 0.0921 | 0.0922 | 0.0921 | 0.1045 | | | |
| | 7 | 0.0732 | 0.1194 | 0.1196 | 0.1772 | 0.1774 | 0.1773 | 0.1773 | 0.1771 | 0.1864 | 0.2072 | |
| | 8 | 0.1132 | 0.1877 | 0.1875 | 0.2387 | 0.2386 | 0.2386 | 0.2387 | 0.2342 | 0.2371 | 0.2495 | |
| | 9 | 0.1146 | 0.1934 | 0.1932 | 0.2358 | 0.2357 | 0.2359 | 0.2357 | OOM | | | |

[1] OOM: Out-of-Memory, Configurations in gray cells were not tested.

(i.e., *k* and *nB* at the following bullets) to construct the global cubic geometry and distribute local boxes to each MPI ranks. In addition, number of thread per MPI rank (i.e., *nthr* at the following bullets) needs to be set to concurrently solve assigned local boxes on MPI ranks using the MPI-OpenMP hybrid implementation. The followings are notations for three parameters with their brief definitions:

- *k*: the log base 2 of the dimension of each box on the finest grid
- *nB*: target number of boxes per process: a loose bound on memory per process.
- *nthr*: number of thread for OpenMP

At first, we tested HPGMG-CPU-8a2f0e1 on 16 XE nodes for a large number of sets for three parameters (i.e., *k* = 4, 5, 6, 7, 8, 9, and *nB* = 1, 2, 3, 4, 5, 6, 7, 8, 16, 32, and *nthr* = 1, 2, 4, 8, 16, 32), and Table XIII presents HPGMG results for the finest resolution (i.e., 1h). Figure 12 shows plots for HPGMG performance vs. number of thread per MPI rank (i.e., *nthr*). Blue circles in Figure 12 expresses the maximum HPGMG performance for each *nthr*. They are very close to a red line in the figure that represents HPGMG results with $256^3$ boxes per thread (i.e., *k*=8 with *nthr*=*nB*). In the figure, there are four additional lines for $128^3$ boxes per thread (i.e., *k*=7 with *nthr*=*nB*), $64^3$ boxes per thread (i.e., *k*=6 with *nthr*=*nB*), $32^3$ boxes per thread (i.e., *k*=5 with *nthr*=*nB*), and $16^3$ boxes per thread (i.e., *k*=4 with *nthr*=*nB*). In the cases for $256^3$ and $128^3$ boxes per thread, we tested them with 16 thread/MPI rank and 32 thread/MPI rank. We observed significant performance loss on those cases due to inefficiencies in OpenMP implementations over NUMA nodes (i.e., 16 thread/MPI) or CPU sockets (i.e., 32 thread /MPI). We note that HPGMG performance improves as the number of boxes per thread increases (see Figure 13). This is understandable because, as local box sizes grow, local calculations for fine grid solutions become enriched and communication costs for coarse grid solutions can rapidly drop. Figure 14 shows plots for HPGMG performance vs. target number of boxes per MPI rank (i.e., *nB*). Depending on the ratio of actually distributed boxes per MPI rank that is usually less than *nB*, HPGMG performance fluctuates a little bit for the same size of local boxes; however, the performance difference between $128^3$ and $64^3$ local boxes is more obvious than the oscillation of the performance with the same size of local boxes.

With $256^3$ boxes per thread, we performed scalability tests with up to 36% Blue Waters XE nodes (i.e., 2, 16, 128, 1024 and 8192 XE nodes). Table XIV presents selected input parameters (i.e., *nthr*, *k* and *nB*), number of employed XE nodes, number of MPI ranks, number of boxes assigned to each thread, total number of boxes, HPGMG performance for three resolutions (i.e., 1h, 2h and 4h), and parallel efficiency based on 2 XE-node performance of each *nthr*. On 8192 XE nodes, HPGMG-CPU-8a2f0e1 shows the best performance with 1 thread/MPI rank, and its parallel efficiency is 93% that is very impressive. In other cases, parallel efficiency is more than or equal to 87% on more than

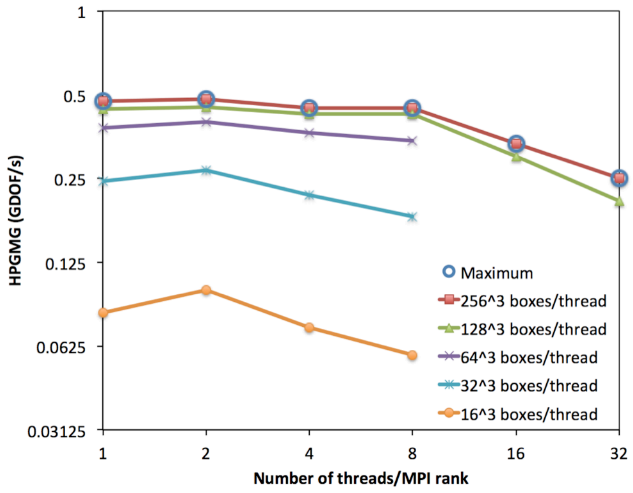

Figure 12. HPGMG performance vs. number of threads/MPI-rank of HPGMG-CPU-8a2f0e1 on 16 XE nodes

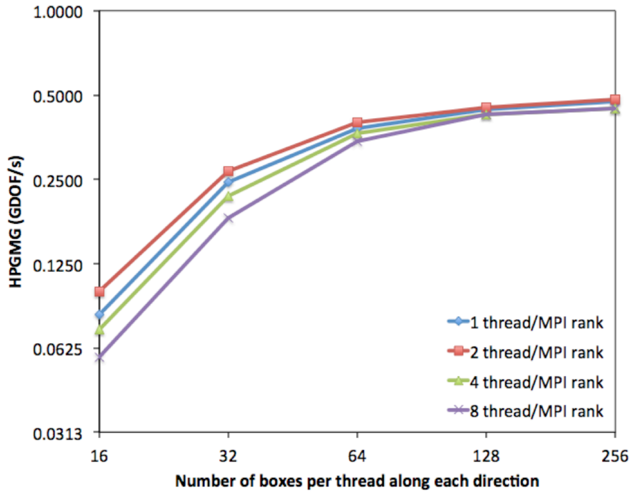

Figure 13. HPGMG performance vs. number of boxes/thread of HPGMG-CPU-8a2f0e1 on 16 XE nodes

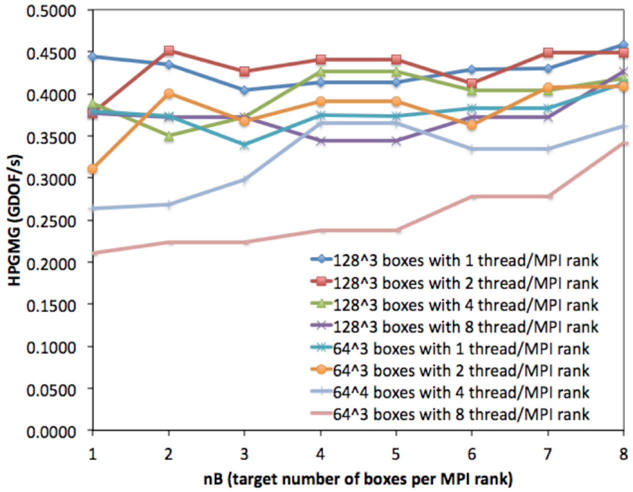

Figure 14. HPGMG performance vs. target number of boxes/MPI rank of HPGMG-CPU-8a2f0e1 on 16 XE nodes

TABLE XIV. HPGMG PERFORMANCE OF HPGMG-CPU-8A2F0E1 WITH UP TO 36% OF BLUE WATERS XE NODES (UNIT: GDOF/S)

| *nthr* | *k* | *nB* | XE nodes | Number of MPI ranks | Number of cells per thread | Number of cells in global | HPGMG (GDOF/s) | | | Parallel efficiency (%) | HPGMG /MPI rank |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | *1h* | *2h* | *4h* | | |
| 1 | 8 | 1 | 2 | 64 | $256^3$ | $1024^3$ | 0.06016 | 0.05753 | 0.04387 | 100 | 0.000940 |
| 1 | 8 | 1 | 16 | 512 | $256^3$ | $2048^3$ | 0.474 | 0.4441 | 0.3767 | 98 | 0.000926 |
| 1 | 8 | 1 | 128 | 4096 | $256^3$ | $4096^3$ | 3.762 | 3.466 | 2.752 | 98 | 0.000918 |
| 1 | 8 | 1 | 1024 | 32768 | $256^3$ | $8192^3$ | 29.35 | 26.18 | 18.62 | 95 | 0.000896 |
| 1 | 8 | 1 | 8192 | 262144 | $256^3$ | $16384^3$ | 229.6 | 198 | 126.3 | 93 | 0.000876 |
| 2 | 8 | 2 | 2 | 32 | $256^3$ | $1024^3$ | 0.06054 | 0.05777 | 0.05312 | 100 | 0.001892 |
| 2 | 8 | 2 | 16 | 256 | $256^3$ | $2048^3$ | 0.4708 | 0.4493 | 0.3952 | 97 | 0.001839 |
| 2 | 8 | 2 | 128 | 2048 | $256^3$ | $4096^3$ | 3.784 | 3.496 | 2.884 | 98 | 0.001848 |
| 2 | 8 | 2 | 1024 | 16384 | $256^3$ | $8192^3$ | 28.85 | 25.39 | 19.03 | 93 | 0.001761 |
| 2 | 8 | 2 | 8192 | 131072 | $256^3$ | $16384^3$ | 223.7 | 189.5 | 125.1 | 90 | 0.001707 |
| 4 | 8 | 4 | 2 | 16 | $256^3$ | $1024^3$ | 0.05672 | 0.05449 | 0.04811 | 100 | 0.003545 |
| 4 | 8 | 4 | 16 | 128 | $256^3$ | $2048^3$ | 0.4473 | 0.4245 | 0.3591 | 99 | 0.003495 |
| 4 | 8 | 4 | 128 | 1024 | $256^3$ | $4096^3$ | 3.526 | 3.286 | 2.654 | 97 | 0.003443 |
| 4 | 8 | 4 | 1024 | 8192 | $256^3$ | $8192^3$ | 26.63 | 23.3 | 17.08 | 92 | 0.003251 |
| 4 | 8 | 4 | 8192 | 65536 | $256^3$ | $16384^3$ | 205.9 | 172.8 | 112.5 | 89 | 0.003142 |
| 8 | 8 | 8 | 2 | 8 | $256^3$ | $1024^3$ | 0.05662 | 0.05462 | 0.04338 | 100 | 0.007078 |
| 8 | 8 | 8 | 16 | 64 | $256^3$ | $2048^3$ | 0.4485 | 0.4244 | 0.3367 | 99 | 0.007008 |
| 8 | 8 | 8 | 128 | 512 | $256^3$ | $4096^3$ | 3.503 | 2.312 | 2.463 | 97 | 0.006842 |
| 8 | 8 | 8 | 1024 | 4096 | $256^3$ | $8192^3$ | 26.3 | 22.86 | 15.59 | 91 | 0.006421 |
| 8 | 8 | 8 | 8192 | 32768 | $256^3$ | $16384^3$ | 201.5 | 165.7 | 100.4 | 87 | 0.006149 |

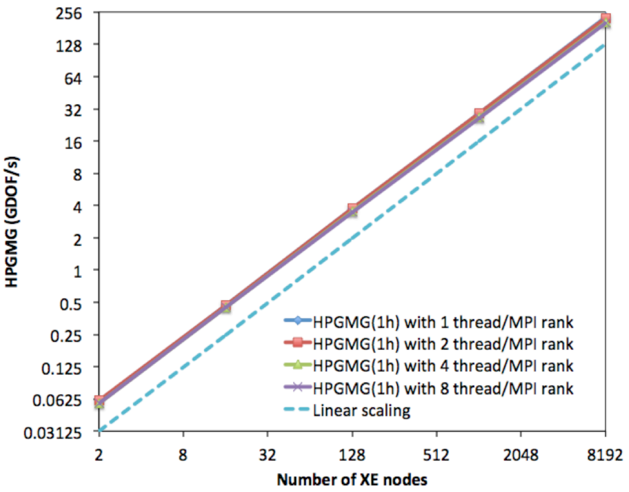

Figure 15. HPGMG performance for various number of thread per MPI rank with up to 36% of Blue Waters XE nodes

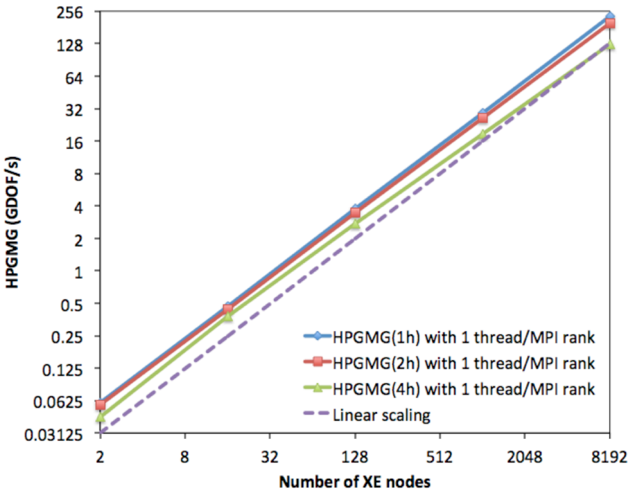

Figure 16. HPGMG performance in terms of resolutions (i.e., 1h, 2h and 4h) with up to 36% of Blue Waters XE nodes

one third of Blue Waters XE nodes. The HPGMG performance per MPI rank improves as the number of threads increases, and its range is between 0.9 and 7.1 MDOF/s. Figure 15 illustrates HPGMG performance for the finest resolution (i.e., 1h). Four solid lines show HPGMG performance with $256^3$ boxes per thread for 1 thread/MPI rank (i.e., blue line), 2 thread/MPI rank (i.e., red line), 4 thread/MPI rank (i.e., green line) and 8 thread/MPI rank (i.e., purple line). Light blue dotted line represents an ideal linear scaling. In most cases, blue or red line shows the best performance. Green and purple lines are lower than blue or read lines but the difference between the best and the worst cases is less than 11 %. Figure 16 shows HPGMG performance for different resolutions with 1 thread/MPI rank. Blue line is for the finest resolution (i.e., 1h), while red and green lines are for 2h and 4h, respectively. Purple dotted line shows an ideal linear scaling. Blue and red lines for 1h and 2h are almost parallel to the purple dotted line. The green line for 4h is sub-linear.

### C. HPGMG on GPU-enabled XK nodes

NVIDIA has shared HPGMG with GPU acceleration via Bitbucket (i.e., https://bitbucket. org/nsakharnykh/hpgmg-cuda). It is based on HPGMG v0.3 compliant MPI version 2, OpenMP version 3 and CUDA version 7. Among multi-grid F-cycles, top levels with fine resolutions run on GPUs, while bottom levels with coarse resolutions run on host CPUs that has less latency than GPUs. We downloaded HPGMG-CUDA source file (commit: 02c7ea214ce8) and built an HPGMG-CUDA binary (i.e., HPGMG-GPU-02c7ea2) with CUDA 7.5 and cray-mpich/7.3.3 on Blue Waters.

TABLE XV. HPGMG PERFORMANCE OF HPGMG-GPU-02C7EA2 ON 16 XK NODES (UNIT: GDOF/S)

| nMPI /GPU[(1)] | k | nB | XK nodes | nMPI[(2)] | Boxes /MPI[(3)] | Total boxes | HPGMG (1h) |
|---|---|---|---|---|---|---|---|
| 1 | 8 | 1 | 16 | 16 | 1 | $512^3$ | 0.3304 |
| 1 | 8 | 2 | 16 | 16 | 1.688 | $768^3$ | 0.5639 |
| 1 | 8 | 3 | 16 | 16 | 1.688 | $768^3$ | 0.5636 |
| 2 | 8 | 1 | 16 | 32 | 1.688 | $768^3$ | 0.589 |
| 3 | 8 | 1 | 16 | 48 | 0.562 | $768^3$ | 0.5909 |
| 1 | 7 | 8 | 16 | 16 | 7.812 | $640^3$ | 0.545 |
| 1 | 7 | 9 | 16 | 16 | 7.812 | $640^3$ | 0.5447 |
| 1 | 7 | 14 | 16 | 16 | 13.5 | $768^3$ | 0.6195 |
| 1 | 7 | 16 | 16 | 16 | 13.5 | $768^3$ | 0.6194 |
| 7 | 7 | 2 | 16 | 112 | 1.929 | $768^3$ | 0.5837 |
| 7 | 7 | 3 | 16 | 112 | 1.929 | $768^3$ | 0.5829 |
| 1 | 6 | 112 | 16 | 16 | 108 | $768^3$ | 0.6433 |
| 1 | 6 | 108 | 16 | 16 | 108 | $768^3$ | 0.6433 |
| 2 | 6 | 54 | 16 | 32 | 54 | $768^3$ | 0.6442 |
| 4 | 6 | 27 | 16 | 64 | 27 | $768^3$ | 0.646 |
| 8 | 6 | 14 | 16 | 128 | 13.5 | $768^3$ | 0.5937 |

(1) Number of MPI rank assigned to each GPU

(2) Number of total MPI rank, (3) Number of boxes per MPI rank

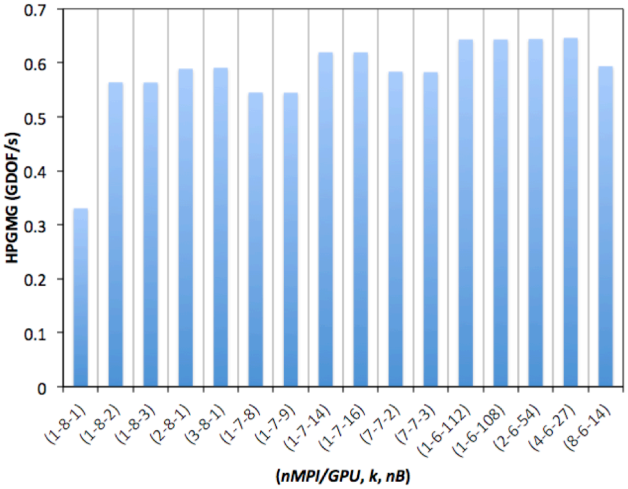

Figure 17. Performance of HPGMG-GPU-02c7ea2 on 16 XK nodes

In this section, we consider an additional parameter, number of MPI rank per GPU (i.e., *nMPI/GPU*) that affects the HPGMG-CUDA performance. Table XV and Figure 17 present the performance of HPGMG-GPU-02c7ea2 on 16 XK nodes for several sets of parameters (i.e., *nMPI/GPU*, *k* and *nB*). Similarly to our discussion in the previous section, HPGMG performance on XK nodes is proportional to total number of boxes. The sets with *k*=6 show the best HPGMG performance on 16 XK nodes.

We performed scalability tests for HPGMG-GPU-02c7ea2 with up to 24% Blue Waters XK nodes with the most optimal configurations (i.e., *k*=6, *nMPI/GPU* = 1, 2, 4 and *nB* = 108, 54, 27) from the tests on 16 XK nodes. While cases with *nMPI/GPU*=2 and 4 on 1024 XK nodes failed to create levels due to "malloc failed" error, other cases ran smoothly, as presented in Table XVI and Figure 18. Parallel efficiency drops very rapidly as number of XK nodes increases. In the worst case, it is 63% on 1024 XK nodes, which is much lower than the worst case on 8192 XE nodes (i.e., 87% in Table XIV). The blue dotted line in Figure 18 represents an ideal linear scaling, and all cases in Figure 18 show sub-linear scaling. We note that the HPGMG performance drops between different resolutions (i.e., 1h vs. 2h vs. 4h) are significantly large compared to our XE scalability tests (see Figure 16). It looks smaller local blocks on XK nodes than on XE nodes results in the huge performance drops. It follows the relation of HPGMG performance and number of boxes per thread shown in Figure 13. HPGMG performance per MPI rank decreases as *nMPI/GPU* increases. It ranges from 9.7 to 32 MDOF/s.

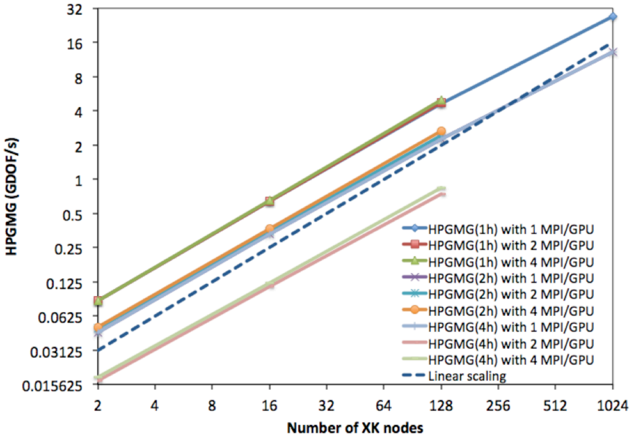

Figure 18. HPGMG performance of HPGMG-GPU-02c7ea2 with up to 24% Blue Waters XK nodes

TABLE XVI. HPGMG PERFORMANCE OF HPGMG-GPU-02C7EA2 WITH UP TO 24% OF BLUE WATERS XK NODES (UNIT: GDOF/S)

| *nMPI/ GPU*[(1)] | *k* | *nB* | XK nodes | Number of MPI ranks | Number of cells per MPI ranks | Number of cells in global | HPGMG (GDOF/s) | | | Parallel efficiency (%) | HPGMG /MPI rank |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | *1h* | *2h* | *4h* | | |
| 1 | 6 | 108 | 2 | 2 | $108 \times 64^3$ | $384^3$ | 0.08475 | 0.04496 | 0.0144 | 100% | 0.042375 |
| 1 | 6 | 108 | 16 | 16 | $108 \times 64^3$ | $768^3$ | 0.6433 | 0.3325 | 0.1043 | 95% | 0.040206 |
| 1 | 6 | 108 | 128 | 128 | $108 \times 64^3$ | $1536^3$ | 4.611 | 2.271 | 0.697 | 85% | 0.036023 |
| 1 | 6 | 108 | 1024 | 1024 | $108 \times 64^3$ | $3072^3$ | 27.24 | 13.11 | 4.104 | 63% | 0.026602 |
| 2 | 6 | 54 | 2 | 4 | $54 \times 64^3$ | $384^3$ | 0.08539 | 0.0486 | 0.01677 | 100% | 0.021348 |
| 2 | 6 | 54 | 16 | 32 | $54 \times 64^3$ | $768^3$ | 0.6446 | 0.3463 | 0.1133 | 94% | 0.020144 |
| 2 | 6 | 54 | 128 | 256 | $54 \times 64^3$ | $1536^3$ | 4.734 | 2.419 | 0.7401 | 87% | 0.018492 |
| 2 | 6 | 54 | 1024 | 2048 | $54 \times 64^3$ | $3072^3$ | malloc failed - create_level/level->my_boxes | | | | |
| 4 | 6 | 27 | 2 | 8 | $27 \times 64^3$ | $384^3$ | 0.08531 | 0.05022 | 0.01801 | 100% | 0.010664 |
| 4 | 6 | 27 | 16 | 64 | $27 \times 64^3$ | $768^3$ | 0.6492 | 0.366 | 0.1217 | 95% | 0.010144 |
| 4 | 6 | 27 | 128 | 512 | $27 \times 64^3$ | $1536^3$ | 4.972 | 2.675 | 0.8368 | 91% | 0.009711 |
| 4 | 6 | 27 | 1024 | 4096 | $27 \times 64^3$ | $3072^3$ | malloc failed - create_level/level->my_boxes | | | | |

(1) Number of MPI rank assigned to each GPU

### *D. HPGMG on CPU-based XE and GPU-enabled XK nodes on MPMD mode*

MPMD mode requires the same base code for CPU-based and GPU-enabled executable binaries; therefore, we downloaded HPGMG v0.3 (commit: 26f74a2120ce) that is an older version than HPGMC-CPU-8a2f0e1. We then built an additional CPU based binary (i.e., HPGMG-CPU-26f74a2) for our MPMD runs. As we discussed in the previous section for HPCG MPMD mode tests, we wanted to synchronize HPGMG performance per MPI rank on XE and XK nodes. For that, we needed to increase *nthr* (number of thread) to improve the per-MPI performance of CPU-based binary. We also increased *nMPI/GPU* (number of MPI rank per GPU) in order to reduce the per-MPI performance of GPU-enabled binary. Since the GPU memory per XK node (i.e., 6 GB) was much smaller than CPU memory per XE node (i.e., 64 GB), we also reduced the box size assigned to each XE node to the level of each XK node. After several parametric tests, we selected a set of parameters (i.e., *nthr*=8, *nMPI/GPU*=7, *k*=7 and *nB*=2) for an optimal MPMD run by synchronizing per-MPI performances on XE and XK nodes. As references, we selected two sets of parameters; one is for the best CPU-based binary (i.e., *nthr*=1, *k*=8 and *nB*=1) and the other is for the best GPU-enable binary (i.e., *nMPI/GPU*=1, *k*=6 and *nB*=108).

Table XVII, Figures 19 and 20 show HPGMG performance on MPMD mode, XE-only mode and XK-only mode. We tested with 16 XE and 16 XK nodes, and then with 128 XE and 128 XK nodes. Unfortunately, one of the reference sets for the best CPU-based binary (i.e., *nthr*=1, *k*=8 and *nB*=1) encountered out-of-memory errors. It seems the recent HPGMG (commit: 8a2f0e1123c7) is better at the memory usage for the same size of problems than HPGMG v0.3 (commit: 26f74a2120ce).

With the first set of parameters for the synchronized per-MPI performance (i.e., *nthr*=8, *nMPI/GPU*=7, *k*=7 and *nB*=2), it turned out HPGMG performance on MPMD mode was similar to summation of HPGMG performances of XE-only and XK-only modes. On 16 XE and 16 XK nodes, HPGMG at each MPI rank (i.e., 4.516 MDOF/s) was

TABLE XVII. BOXESHPGMG PERFORMANCE OF HPGMG-CPU-26F74A2 AND HPGMG-GPU-02C7EA2 ON MPMD MODE (UNIT: GDOF/S)

| **Run type (*nthr*, *nMPI/GPU*, *k*, *nB*)** | **XE nodes** | **XK nodes** | **nMPI @XE(1)** | **nMPI @XK(2)** | **Number of cells** | **Boxes /MPI(3)** | **HPGMG (GDOF/s)** | | | **HPGMG /MPI rank(4)** |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | ***1h*** | ***2h*** | ***4h*** | |
| MPMD (8,7,7,2) | 16 | 16 | 64 | 112 | 896 | 1.949 | 0.7949 | 0.2997 | 0.0768 | 0.004516 |
| XE only (8,7,7,2) | 16 | | 64 | | 640 | 1.953 | 0.3525 | 0.2084 | 0.063 | 0.005508 |
| XK only (8,7,7,2) | | 16 | | 112 | 768 | 1.929 | 0.4828 | 0.1578 | 0.03365 | 0.004311 |
| MPMD (2,1,6,108) | 16 | 16 | 256 | 16 | 1792 | 80.706 | 0.4002 | 0.3257 | 0.2294 | 0.001471 |
| XE only (2,1,6,108) | 16 | | 256 | | 768 | 85.75 | 0.426 | 0.3437 | 0.2363 | 0.001664 |
| XK only (2,1,6,108) | | 16 | | 16 | 1792 | 108 | 0.6433 | 0.3325 | 0.1043 | 0.040206 |
| MPMD (1,1,8,1) | 16 | 16 | 512 | 16 | 2048 | 0.97 | Out of memory | | | |
| MPMD (8,7,7,2) | 128 | 128 | 512 | 896 | 1792 | 1.949 | 5.818 | 1.917 | 0.433 | 0.004132 |
| XE only (8,7,7,2) | 128 | | 512 | | 1280 | 1.953 | 2.618 | 1.417 | 0.3976 | 0.005113 |
| XK only (8,7,7,2) | | 128 | | 896 | 1536 | 1.929 | 3.557 | 1.026 | 0.2039 | 0.003970 |
| MPMD (2,1,6,108) | 128 | 128 | 2048 | 128 | 3584 | 80.706 | 3.171 | 2.539 | 1.69 | 0.001457 |
| XE only (2,1,6,108) | 128 | | 2048 | | 1536 | 85.75 | 3.373 | 2.682 | 1.764 | 0.001647 |
| XK only (2,1,6,108) | | 128 | | 128 | 3584 | 108 | 4.611 | 2.271 | 0.697 | 0.036023 |
| MPMD (1,1,8,1) | | 128 | | 128 | 3584 | 108 | Out of memory | | | |

(1) nMPI/XE: number of MPI ranks assigned to XE nodes, (2) nMPI/XE: number of MPI ranks assigned to XK nodes,
(3) Boxes/MPI: number of boxes ($2^k$ cells) assigned to each MPI rank, (4) HPGMG/MPI rank: HPGMG performance per MPI rank

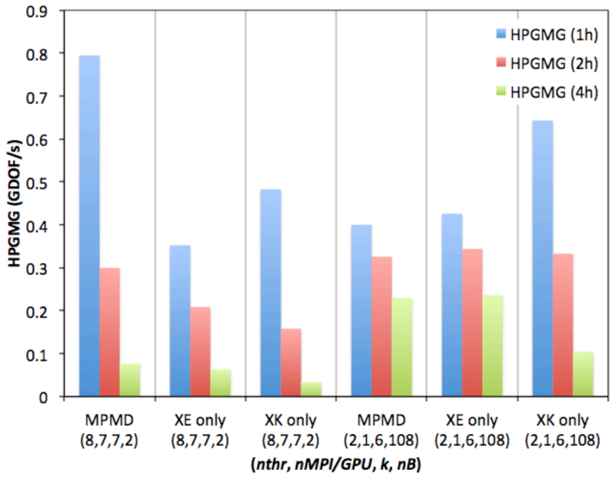

Figure 19. HPGMG performance on MPMD mode with 16 XE nodes and 16 XK nodes

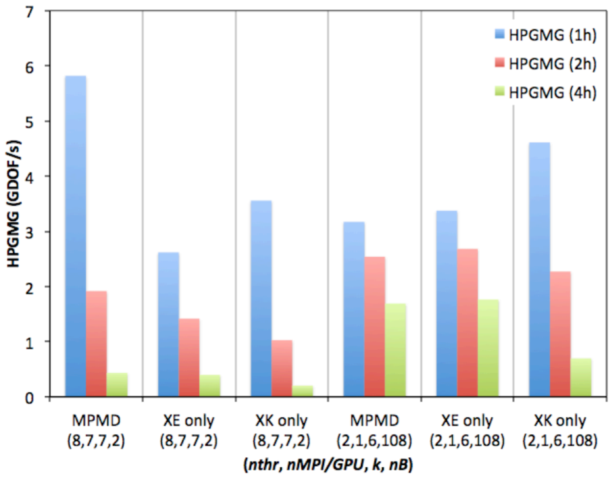

Figure 20. HPGMG performance on MPMD mode with 128 XE nodes and 128 XK nodes

between those of XE-only (i.e., 5.508 MDOF/s) and XK-only (i.e., 4.311 MDOF/s) modes. The per-MPI performance of MPMD mode on 128 XE and 128 XK nodes (i.e., 4.132 MDOF/s) was also between those of XE-only (i.e., 5.113 MDOF/s) and XK-only (i.e., 3.97 MDOF/s).

HPGMG performances on MPMD mode with the second set of parameters for the best GPU-enable performance (i.e., *nMPI/GPU*=1, *k*=6 and *nB*=108) were much lower than those with the first set of parameters. They were even lower than HPGMG performances on XE-only or XK-only mode. It is because the performances on MPMD mode were bounded by lower per-MPI performance that was of CPU based binary with the second set of parameters. On 16 XE and 16 XK nodes, the per-MPI performance on MPMD mode (i.e., 1.471MDOF/s) was slightly lower than that on XE-only mode (i.e., 1.664 MDOF/s) and much lower than that on XK-only mode (i.e., 40.21 MDOF/s). The HPGMG performance per MPI rank on MPMD mode with 128 XE and 128 XK nodes (i.e., 1.457 MDOF/s) was also lower than that on XE-only mode (i.e., 1.647 MDOF/s) and much lower than that on XK-only mode (i.e., 36.02 MDOF/s).

## IV. Concluding remarks

HPCG and HPGMG benchmarking are new metrics to measure modern HPC performance. They include many featured aspects representing important science and engineering HPC applications popularly used in modern HPC community. We performed HPCG and HPGMG benchmarking tests on CPU-based XE nodes, GPU-enabled XK nodes, and both XE and XK nodes with MPMD mode on Blue Waters.

HPCG benchmarking on CPU-based XE nodes showed consistent performance for various numbers of equations per MPI process, and the parallel efficiency kept more than or equal to 86% on tested cases (up to 4.6% Blue Waters XE nodes, 1024 XE nodes). We tested HPCG on GPU-enabled XK nodes with up to 6.1% Blue Waters XE nodes (i.e., 256 XK nodes). Its performance increased exponentially with number of equations per MPI process at the beginning, and then it converged to a certain level. The parallel efficiency of HPCG on XK nodes was more than or equal to 88% that was a little bit lower than that on the same number of XE nodes (i.e., 92%). HPCG performance on XK nodes was around 2.7 times faster than HPCG on the same number of XE nodes. For an optimal performance on MPMD mode, we tried to synchronize HPCG performance per MPI process on XE and XK nodes. For that, we increased number of threads on XE nodes and increased number of MPI ranks per GPU on XK nodes. As a result, we selected a set of parameters with $128\times64^3$ local blocks per MPI process, 8 threads per MPI process on XE nodes and 10 MPI processes per GPU on XK nodes. It resulted in around 60% to 100% better performance than other reference cases that were chosen for the best CPU-based performance or the best GPU-based performance.

HPGMG performance on XE nodes rapidly increased with number of cells assigned to each compute node, and then it was converged. Its OpenMP implementation was efficient enough to keep the performance at a certain level with threading in a NUMA core. The parallel efficiency of HPGMG on XE nodes is impressive, at more than 87% with up to 36% of the available XE nodes (i.e., 8192 XE nodes). HPGMG on GPU-enabled XK nodes was sensitive to number of cells per MPI process, since the assignable number of cells on GPU was not big enough for HPGMG performance to be converged. Its performance on 16 XK nodes was around 34% faster than HPGMG on 16 XE nodes, but the parallel efficiency on XK nodes rapidly dropped to 63% on 1024 XK nodes (i.e., 24% Blue Waters XK nodes). That was much lower than the parallel efficiency of HPGMG on 1024 XE nodes (i.e., 91%). For MPMD mode, we again tried to synchronize HPGMG performance per MPI process on XE and XK nodes. After several parametric studies, we selected an optimal set of parameters (i.e., 8 thread per MPI process on XE nodes, 7 MPI processes per GPU on XK nodes and $2\times128^3$ cells per MPI process). It turned out that HPGMG on MPMD mode with the selected set of parameters shows around 100% better performance than a reference case that was optimized for the best HPGMG performance with GPU-enabled executable binary.

In summary, we performed HPCG and HPGMG benchmarking tests on Blue Waters with various configurations. We analyzed each benchmark on CPU-based nodes, GPU-enabled nodes, and both of heterogeneous nodes with MPMD mode. In both benchmarks, the performance on GPU-enabled nodes was better than those on CPU-based nodes. To obtain an optimal performance on MPMD mode, we tried to synchronize performance per MPI process on CPU-based and GPU-enable nodes. It turned out the synchronization of the per-MPI performance resulted in 60% to 100% better performance than other reference cases. We hope this study will help Cray users optimize their applications for science and engineering projects on modern hybrid HPC systems.

## Acknowledgment

This study is part of the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (awards OCI-0725070 and ACI-1238993) and the state of Illinois. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign and its National Center for Supercomputing Applications. We thank Massimiliano Fatica and Mauron Bisson at NVIDIA for providing HPCG binaries used in this paper.  We also thank Samuel Williams at LBNL, Nikolai Sakharnykh at NVIDIA and Vladimir Marjanovic at HLRS for sharing your precious experience on HPGMG benchmarking with us.

## References

[1] J. Dongarra, J. Bunch, C. Moler and G.W. Stewart “LINPACK Users Guide,” SIAM, Philadelphia, PA, 1979.

[2] M. Heroux, J. Dongarra and P. Luszczek “HPCG Technical Specification,” Sandia National Laboratories Technical Report, SAND2013-8752, October, 2013.

[3] J. Dongarra, M. Heroux and P. Luszczek “HPCG Benchmark: a New Metric for Ranking High Performance Computing Systems,” Technical Report, Electrical Engineering and Computer Sciente

Department, Knoxville, Tennessee, UT-EECS-15-736, November, 2015.

[4] M. Adams, J. Brown, J. Shalf, B. Straalen, E. Strohmaier and S. Williams, “HPGMG 1.0: A Benchmark for Ranking High Performance Computing Systems,” LBNL Technical Report, 2014, LBNL 6630E.

[5] S. Williams, “4th Order HPGMG-FV Implementation,” HPGMG BoF, Supercomputing, November 2015.

[6] J. Kwack, G. Bauer and S. Koric, “Performance Test of Parallel Linear Equation Solvers on Blue Waters – Cray XE6/XK7 system,” Preceedings of the Cray Users Group Meeting (CUG2016), London, England, May 2016.

[7] B. Bode, M. Butler, T. Dunning, W. Gropp, T. Hoe-fler, W. Hwu, and W. Kramer (alphabetical), “The Blue Waters Super-System for Super-Science,” Contemporary HPC Architectures, Jeffery Vetter editor. Sitka Publications, November 2012.Edited by Jeffrey S . Vetter, Chapman and Hall/CRC 2013, Print ISBN: 978-1-4665-6834-1, eBook ISBN: 978-1-4665-6835-8.

[8] W. Kramer, M. Butler, G. Bauer, K. Chadalavada and C. Mendes, “Blue Waters Parallel I/O Storage Sub-system,” High Performance Parallel I/O, Prabhat and Quincey Koziol editors, CRC Publications, Taylor and Francis Group, Boca Raton FL, 2015, Hardback Print ISBN 13:978-1-4665-8234-7.