Ookami: Deployment and Initial Experiences

A. Burford, D. Oryspayev

Submitted to the PEARC '21: Practice and Experience in Advanced Research Computing Conference
to be held at Boston MA
July 18 - 22, 2021

Computational Science Initiative

Brookhaven National Laboratory

U.S. Department of Energy

USDOE Office of Science (SC), Advanced Scientific Computing Research (SC-21)

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or any third party's use or the results of such use of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof or its contractors or subcontractors. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

Ookami: Deployment and Initial Experiences

ANDREW BURFORD, ALAN C. CALDER, DAVID CARLSON, BARBARA CHAPMAN, FIRAT COŞKUN, TONY CURTIS, CATHERINE FELDMAN, ROBERT J. HARRISON, YAN KANG, BENJAMIN MICHALOWICZ, ERIC RAUT, EVA SIEGMANN, and DANIEL G. WOOD, Institute for Advanced Computational Science, Stony Brook University, USA

ROBERT L. DELEON, MATHEW JONES, NIKOLAY A. SIMAKOV, and JOSEPH P. WHITE, Center for Computational Research, University at Buffalo, USA

DOSSAY ORYSPAYEV, Brookhaven National Laboratory, USA

Ookami [3] is a computer technology testbed supported by the United States National Science Foundation. It provides researchers with access to the A64FX processor developed by Fujitsu [17] in collaboration with RIKΞN [35, 37] for the Japanese path to exascale computing, as deployed in Fugaku [36], the fastest computer in the world [34]. By focusing on crucial architectural details, the ARM-based, multi-core, 512-bit SIMD-vector processor with ultrahigh-bandwidth memory promises to retain familiar and successful programming models while achieving very high performance for a wide range of applications. We review relevant technology and system details, and the main body of the paper focuses on initial experiences with the hardware and software ecosystem for micro-benchmarks, mini-apps, and full applications, and starts to answer questions about where such technologies fit into the NSF ecosystem.

CCS Concepts: • **General and reference** → **Surveys and overviews**; • **Applied computing**; • **Hardware**;

Additional Key Words and Phrases: high-performance computing, computer systems, exascale

ACM Reference Format:
Andrew Burford, Alan C. Calder, David Carlson, Barbara Chapman, Fırat Coşkun, Tony Curtis, Catherine Feldman, Robert J. Harrison, Yan Kang, Benjamin Michalowicz, Eric Raut, Eva Siegmann, Daniel G. Wood, Robert L. DeLeon, Mathew Jones, Nikolay A. Simakov, Joseph P. White, and Dossay Oryspayev. 2018. Ookami: Deployment and Initial Experiences. In *PEARC '21: Practice & Experience in Advanced Research Computing (PEARC), Virtual.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Fugaku [36], the presently fastest computer in the world, and its associated processor technology [17] have achieved an *unprecedented sweep* of five major benchmarks [34] — Top 500 [43], Graph-500 [33], HPCG [42], HPL-AI [5], and, in 2019, Green-500 [16]. The Fujitsu Tofu-D communication fabric [1] certainly contributes to these successes. But central is the novel A64FX processor [17], which is a significant reversal in the global trend to supercomputers being powered by graphical processing units (GPUs) that typically burden the programmer with proprietary programming models and complex memory hierarchies. The Ookami test bed gives US researchers free access to evaluate this exciting processor technology. It is also an opportunity to explore the potential of this technology for deployment in multiple settings including at extreme scale in national supercomputer centers or on campus at the network edge.

The 48-core, 64-bit ARM processor is the first to deploy the Scalable Vector Extension (SVE) SIMD-vector instruction set, employing 512-bit wide vectors matched with 32 Gbyte of high-bandwidth memory (1 Tbyte/s). Designed specifically for leadership supercomputers, this processor+memory system promises to retain familiar and successful programming models while achieving very high performance for a wide range of applications. It supports 64/32/16-bit floating-point representations and fast partial dot-product of 8-bit integers to 32-bit results, and hence enables both HPC and big data. As demonstrated by the HPCG and Top500 benchmarks, A64FX offers strong performance on memory-intensive applications such as sparse-matrix solvers, while also being competitive on floating-point-intensive codes. We will see below that for some software this transformational performance is available nearly out of the box — MPI+OpenMP vectorized code should just compile and immediately run well, with additional performance possible from tuning.

The applications that will benefit most are those sensitive to memory bandwidth with a memory footprint less than 32 GB/node — these two features capture the vast majority of NSF's workload. In 2017, team members conducted a workload study [40] of NSF's computing resources including large-memory queues. The study found that 86% of all jobs (corresponding to 85% of cycles) will fit within the available memory per node (including O/S) and that the majority of jobs are memory-bandwidth intensive.

2 OOKAMI TECHNOLOGY AND SYSTEM ARCHITECTURE

2.1 System configuration

The Ookami HPE Apollo 80 system has 174 1.8 Ghz A64FX compute nodes each with 32GB of high-bandwidth memory and a 512 Gbyte SSD. The latter is a post-market addition made by the project. A high-performance Lustre filesystem employing the Cray ClusterStor E1000 provides about 0.8Pbyte storage. Nodes are connected by an HDR 200 Gbit/s Infiniband network configured as a full fat tree. Using a Y-cable, the HDR 100 Gbit/s NICs of two compute nodes are connected to each HDR 200 Gbit/s switch port. This recognizes the bandwidth limitation of the on-chip PCIe interface and also reduces cable density. The login nodes were originally x86 but are now dual socket ThunderX2 with 256 Gbyte memory to eliminate most cross-compilation issues. In addition, Ookami includes servers powered by dual-socket AMD Rome, Intel SkyLake, and Intel Haswell (two with NVIDIA V100 GPUs) for performance comparisons across architectures.

2.2 ARM background

ARM is a RISC processor designed by ARM Ltd. but specialized and manufactured by other companies leading to much greater diversity than exists for X86. Annual processor production of ARM is well over 20 billion, which dwarfs X86 at 3-400 million [38]. The development of 64-bit and other features with a deliberate push, including by the U.S. DOE FastForward program, made the processor relevant to HPC. More recently, systems on the Top500 [20] have been powered by the 64-bit Cavium ThunderX2 specifically optimized for HPC including 128-bit SIMD units and more cores. A detailed evaluation [24] of a Cray XC50 system (dual-socket ThunderX2), examined a wide range of applications relevant to NSF users with four compelling conclusions: (1) ARM can now provide performance competitive with mainstream processors; (2) with significant increases in price performance; (3) most benchmarks ran successfully out of the box; and (4) with no special tuning necessary for high performance. This closely mirrors our experience with A64FX,

Table 1. Available compilers.

Compiler	Compiler names	Version	Vectorization Flag
GCC	gcc / g++ / gfortran	10.2.1	-mcpu=A64FX
Arm	armclang / armclang++ / armflang	20.3	-mcpu=A64FX -armpl=sve
LLVM	clang / clang++ / flang	11.0.0	-mccpu=A64FX
Cray	cc / CC / ftn	10.0.1	-h vec3
NVIDIA	nvc / nvc++ / nvfortran	21.2.0	-Mvect[=options]
Fujitsu (will be installed soon)	fcc / FCC / frt	4.5.0	-KSVE

with the exception that only well-vectorized applications attain a substantial fraction of peak speed on A64FX primarily due to (1) the 512-bit vector width that is 4x that of NEON, and (2) the relatively higher latency of many instructions.

2.3 Processor architecture

The A64FX processor comprises 4 non-uniform memory access (NUMA) regions or core memory groups (CMGs) with on-package, stacked, high-bandwidth memory (8 Gbyte). A CMG consists of 12 cores[1] with private L1 cache (64 Kbyte), a shared L2 cache (8 Mbyte), and a memory controller. The four CMGs keep cache coherency by ccNUMA with an on-chip directory. A cross-bar connection in a CMG maximizes high efficiency for throughput of the L2 cache (>115 GB/s/core read, >57 GB/s/core write). Out-of-order mechanisms in cores, caches, and memory controllers maximize the utilization of bandwidth at each level. The cache line is 256 bytes.

The A64FX core has a superscalar architecture with out-of-order execution and branch prediction. Peak execution rate is provided by issuing 2x512-bit wide SIMD FMA + predicate op + 4x ALU (shared with 2x address generation) instructions per cycle. The theoretical peak double precision speed in FLOP/s is computed as 1.8 GHz×2 FMA/cycle×2 FLOPs/FMA×8 64-bit words/vector = 57.6 GFLOP/s/core. Rates for other precisions are obtained by adjusting the number of words per vector. Machine learning applications are enabled by the FP16 high performance and the INT8 partial dot product that computes the elementwise product of vectors of 8-bit integers, reducing in groups of 4 into a vector of 32-bit results.

Each cycle 2x512-bit SIMD load or 1x512-bit SIMD store are supported and are enabled by the L1 data cache that can sustain 128 bytes/cycle even for unaligned SIMD load. The L1 "combined gather" doubles the gather (indirect) load's data throughput, when target elements are within a 128-byte aligned block for a pair of two registers, similar to the coalescing of out-of-order loads within a warp on some GPGPUs. The sector cache (scratchpad) feature is a valuable optimization for a library writer by enabling the programmer to modify how the cache replacement policy treats data, including forcing data to be kept in cache. There are 6 hardware barriers per CMG.

2.4 Scalable vector extension (SVE)

SVE supports vector widths from 128 to 2048 bits in 128-bit increments and was designed for HPC. With full per-lane predication, it superficially appears similar to Intel's AVX/AVX2/AVX512 instruction sets, however, there are important differences. SVE instructions can be generated to be vector-length agnostic (VLA), enabling a binary to run on any implementation regardless of vector width, and eliminating the need for loop tails. The instruction set also includes gather/scatter, vector partitioning, software-managed speculation, horizontal reductions, and scalarized intra-vector sub-loops, which all facilitate compilers vectorizing more loops.

[1] The processor in Fugaku includes an additional core for O/S services.

3 SOFTWARE ENVIRONMENT

The operating system on Ookami is the industry-standard and stable Linux distribution CentOS 8, and all features and tools work as expected. Indeed, outside of different performance tuning concerns one usually has little awareness of being on an A64FX platform. The industry standard Bright Cluster Manager, which has supported ARM for many years, is used to provision the hardware, operating system, and workload manager from a unified interface. SLURM (Simple Linux Utility for Resource Management, version 19.05.7) is employed to schedule jobs and manage the resources with nodes allocated exclusively to one job. Fair-share scheduling is presently used, rather than quotas, in order to encourage utilization. There are various partitions available covering different job duration, job sizes and architectures. Following our practice on other clusters at Stony Brook, each user has a personal home and scratch directory that are forced to be private. Each project has a shared directory accessible only by project members. The home and project directories are backed up daily.

3.1 Development environment

Users have access to a continuously growing software stack, organized into modules to simplify use, management, and version control. The available compilers are listed in Table 1. All of these, except for the NVIDIA and mainstream LLVM compilers, are capable of generating optimized SVE code. However, there are significant differences in capability and performance. We do not yet have substantial experience with the Fujitsu compiler. The ARM and Cray compilers can generate SVE instructions for loops containing standard math functions, with the Cray compiler typically generating faster code. However, the Cray C++ compiler front end is limited in its support for modern C++ standards, and, similarly for the ARM Fortan compiler front end. Only the ARM and recent GNU compilers support the SVE intrinsics, with the GNU compiler typically generating faster code. The ARM, Cray, and recent GNU compilers can all vectorize "if-tests" using the SVE predicated instruction set. Starting with version 10.2.1, the GNU chain is aware of the A64FX processor, and similarly for the upcoming LLVM version 11.0.0 that will add support for SVE intrinsics but will still lack SVE autovectorization.

Thus, for C and C++ the choice of compiler is complicated. Code using recent language standards or SVE intrinsics must use ARM or GNU, with the former recommended for kernels using math functions and the latter for SVE intrinsics. For FORTRAN, the Cray compiler is typically the best choice for both performance and standard compliance.

Other available modules include anaconda, archiconda, armpl, libsci, lapack, openblas, fftw, papi, cmake, etc. Various MPI versions are available, e.g. openmpi 4.0.5 and 4.1.0 and openshemem (Section 4.1.2).

3.2 Performance analysis and debugging

Multiple toolchains enable the user to analyze and optimize their code. Arm Forge includes the Arm ddt debugger, the profiling tool ARM MAP and ARM performance reports for advanced reporting capabilities. The Cray toolchain includes CrayPat, Apprentice and Reveal tools for performance analysis/tuning, and the debugging tools gdb4hpc (general debugger), valgrind4hpc (for detecting memory leaks and errors in parallel applications) and atp (first-line tool to diagnose crashing applications). The GNU gdb command-line debugger is also available. The Fujitsu toolchain is not yet installed on Ookami, but will be provided in the near future.

4 INITIAL EXPERIENCE

Our standing joke is that the system is "ARM-less" (i.e., "harmless") in that standard-compliant applications in FORTRAN, C, or C++ simply compile and run out of the box, once mundane issues such as compiler flags and library paths have been addressed. This is due to (1) the standard and complete Linux distribution, (2) the extensive selection of standard-compliant tool chains, (3) a growing library of linear algebra and scientific kernels, and (4) the availability of multiple MPI implementations (Cray, MVAPICH, OpenMPI) all optimized for A64FX and SVE. More nuanced is what must be done to obtain high performance from the processor along with selection of the appropriate tool chain(s), as discussed above. Early concerns included that InfiniBand performance would be inferior due to the depth of instruction pipelines and cache architecture — these have proven unfounded (Section 4.1.2).

Performance measurements deliver highly reproducible results between runs on the same or different sets of nodes. For instance, whole-node DGEMM benchmarks run by HPE/Cray during the burn-in phase detected two slow nodes, which were replaced, and the performance of all other nodes had a spread reproducible less than 1%. This is due to the fixed clock speed and absence of turbo modes or thermal throttling. Multi-node runs are similarly reproducible due to fixed clock speed and the full fat tree that eliminates network contention.

The most important early discovery was the need to install an SSD on all nodes to store the root file system. As shipped, only two nodes targeted for software development and debugging had SSDs. All other nodes had a memory-resident system image — but this consumed ~14 Gbyte instead of the anticipated ~2 Gbyte. This turned out to be due to the 64 Kbyte page size, which is the minimum space consumed by even a 1 byte file. The SSDs open the door to new data-intensive applications, and in the near future may be upgraded from 512 Gbyte to 2 Tbyte.

So far the system has been very reliable. The two slow nodes were were replaced along with one NIC and 3 SSDs. Unscheduled outages have been due to power failures and non-A64FX-specific system software including a memory leak in a SLURM demon.

4.1 Micro and standard benchmarks

4.1.1 Single-node DGEMM and STREAM. Single-node DGEMM was measured by Cray on Ookami at 24.2 TFLOP/s, which is 88% of peak single processor speed. We are anticipating this to increase as Cray scilib is further optimized, and when we get access to the Fujitsu library. Fujitsu has reported 94% of peak single processor speed. At SBU, we have developed a code generator for AVX2, AVX512, NEON, and SVE instrinsics for the matrix-transpose times matrix

Table 2. OSU MPI Benchmarks

Benchmark	Msg Size	Est.	Measured
Two-sided uni-dir B/W	1MB	12.3 GB/s	12.3
Two-sided bi-dir B/W	1MB	24.4 GB/s	19.4
Two-sided latency	8B	1.8 µs	2.7
Put latency	8B	2.8 µs	3.9
Put BW	1MB	12 GB/s	12.3
Get latency	8B	3.8 µs	5.2
Get BW	1MB	9.5 GB/s	12.3
Injection Rate	1MB	12.3 GB/s	12.3
Saturation Point	1MB	1 thread	1

kernel that is important in high-order spectral-element methods. This code attains 53.24 GFLOP/s on a single core, which is 92.4% of peak speed.

Single-node STREAM TRIAD was measured by Cray on Ookami at 830 Gbyte/s, and a simple compiled DAXPY kernel with careful binding of threads+memory to cores achieved 840 Gbyte/s. Manually-tuned kernels were needed to achieve close to peak L1 bandwidth. About six threads are required to saturate the memory bandwidth of a single CMG. The observed 840 Gbyte/s is 84% of peak memory bandwidth, 93% of what we understand to be the L2-memory bandwidth, and is about 5x a dual-socket Intel Skylake. The measured single-core DAXPY bandwidth is 53 Gbyte/s, which is 2.5X Intel SkyLake. This memory bandwidth contributes greatly to the empirically observed strong thread scaling within a node.

4.1.2 OSU Benchmarks with MPI and OpenSHMEM. The Message Passing Interface [6, 14] (MPI) is a widely-used parallel programming library. It includes both two-sided rendezvous-based communication, and one-sided Remote Memory Access (RMA). OpenSHMEM [4, 28] is a Partitioned Global Address Space (PGAS) library that focuses on one-sided, decoupled, communication and uses RMA to expose memory directly to network interconnects such as Infiniband. The Ohio State University benchmark suite [44] tests bandwidth and latency for both point-to-point and collective operations using MPI and OpenSHMEM (UPC/UPC++ were not tested). We used Open-MPI [27] (version 4.0.5) as it provides both MPI and OpenSHMEM in the same package.

Of particular interest was the point-to-point latency and bandwidth performance as we wanted to investigate the thread-to-network behavior on the A64FX processor. Tests were between 2 nodes, 1 rank or processing element (PE) per node. In Table 2 are the results for the MPI one- and two-sided tests. The "put" and "get" latencies are slightly higher than anticipated. In Table 3 are the results for the OpenSHMEM tests. Included here are representative latencies recorded for the 1-sided RMA operations, the posting and fetching atomics, and swaps. The estimated time comes from comparisons made on a similar A64FX cluster [21] at Oak Ridge National Laboratory.

Installing the open-source Open-MPI and MVAPICH2 [26] worked "out of the box". Both are configured to use the Infiniband network and the on-node cross-memory-attach provided by KNEM [18] and XPMEM [45]. Our OpenSHMEM work involves a separate, reference, implementation of OpenSHMEM (although based on UCX [29] like Open-MPI). This installed without issue as we have worked with ARM on the OpenSHMEM project for a number of years, and it has components in common with Open-MPI, and can take advantage of the on-node modules KNEM and XPMEM. Compilation with the stock GCC, a new GCC with more SVE support, and the ARM compilers were also without issue. UCX includes a number of ARM-specific optimizations, e.g. low-power wait states, high-precision timers, and vectorized memory copy.

4.1.3 HPC Challenge Benchmark suite. (HPCC) [11] combines a large number of benchmarks allowing a convenient way to test multiple subsystems of HPC resource in a single run. Several of the tests rely on linear algebra and FFT

Table 3. OSU OpenSHMEM Benchmarks: Latency

Benchmark	Est.	Measured
Put (8B)	5.4 µs	4.5
Get (8B)	4.2 µs	4.1
Atomics (add)	0.0 µs	0.8
Fetching atomics (fadd)	5.2 µs	5.2
Swap	4.4 µs	4.3

libraries that are often used within scientific and engineering applications. To benchmark these important libraries, we concentrated on matrix-matrix multiplication, High-Performance LINPACK (HPL) [30], and Fast Fourier Transformation (FFT) benchmarks. Several linear algebra libraries are available on Ookami: Cray LibSci, ARM Perfomance Library (ARMPL), and OpenBLAS. The first two already have SVE support, while the last one does not. These libraries' performance is shown in Figure 1.A-B. ARMPL has the best performance while OpenBLAS has the smallest due to lack of SVE support.

To HPCC we added a test of the most recent version of FFTW as well as other libraries providing the FFTW-3 API. Within the FFT benchmark, we compared ARMPL and several version of FFTW library: official (no SVE support), provided by Dolbeau, Cray, and Fujitsu. The results are shown in Figure 1.C. ARMPL (20.3) show extremely low FFT performance, suggesting falling back to a completely non-SIMD version. The Fujitsu version of FFTW shows the best results, followed by Dolbeau and Cray. It is worth mentioning that Dolbeau developed his version of FFTW without access to actual hardware using emulators; and two other versions with SVE support are most likely based on his work.

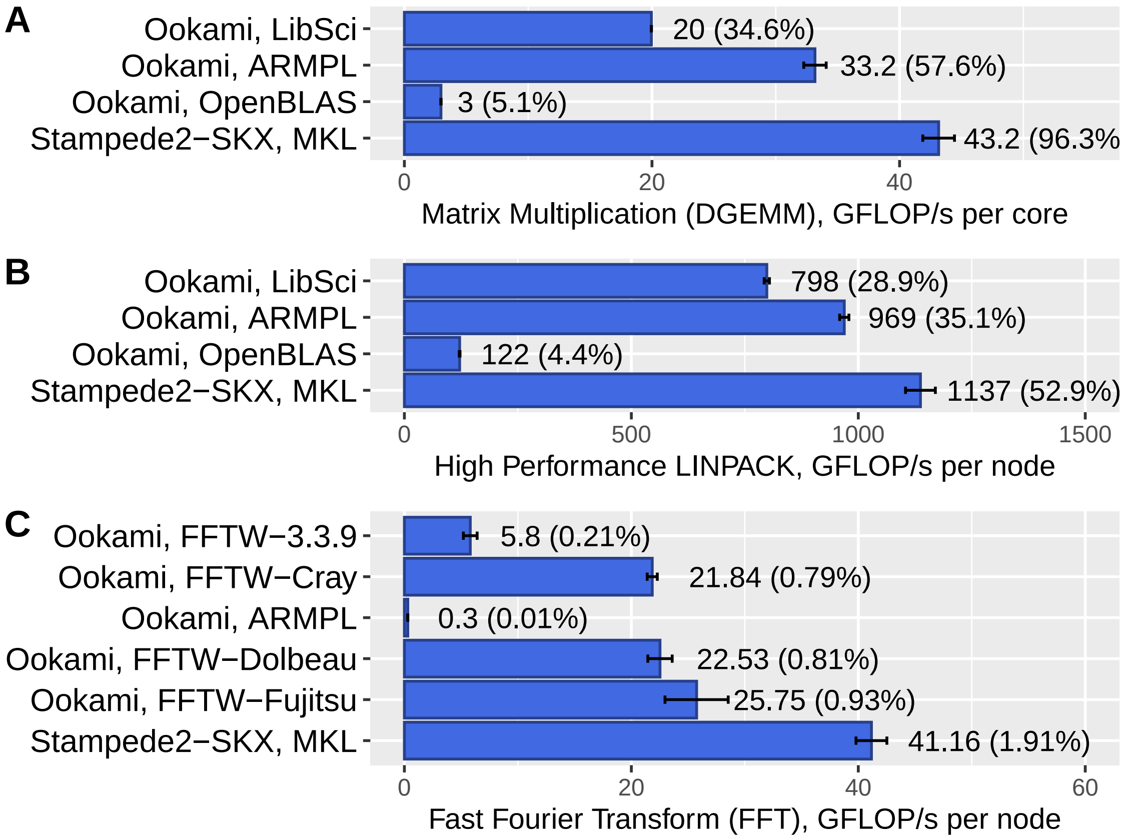

Fig. 1. HPCC benchmark. Performance is shown in GFLOP/s, a number next to the bar corresponds to absolute value, and value in paranthesis corresponds to percentage of theoretical FLOP/s. A) Matrix-matrix multiplication, executed on a single core, B) High Performance LINPACK (HPL), executed on a single node, C) FFT, executed on a single node

4.2 Mini-apps and Applications

4.2.1 Minimod. Minimod [25] is a seismic modeling mini-app developed by Total. Minimod extracts the stencil computation from a production seismic imaging application. The stencil is used to numerically solve the acoustic wave equation. The goal of Minimod is to offer a benchmark to test new and emerging hardware and programming models for geophysics applications.

We evaluated the Minimod OpenMP loop-based and task-based implementations [32] on Ookami. Results are shown in Figure 2 for the task-based implementation. While armgcc shows a performance improvement over armclang at low thread counts, at higher counts the advantage is mostly eliminated.

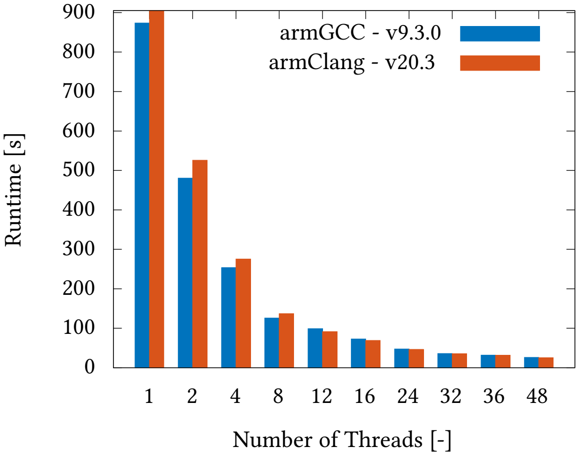

Fig. 2. Minimod: Runtime results between 2 different compilers with SVE optimizations on Ookami

We also ported [31] Minimod to use the Legion task-based parallel programming framework [2] and compared the result to MPI. We also compared the performance to Summit using only Summit's POWER9 CPUs. Throughput is shown in Figure 3. Although the MPI version of Minimod on Ookami (MVAPICH 2.3.4) achieves throughput and speedup comparable to Summit, the Legion version (Figure 3) suffers a performance penalty in both weak scaling and strong scaling — we are still investigating the cause.

4.2.2 SWIM. SWIM [41] is a weather forecasting benchmark (FORTRAN OpenMP) that solves the shallow-water equations using finite difference, and tracks FP speed, bandwidth, and cache characteristics, and is now included in the SPEC CPU2000 Benchmark suite. In our experiment, we used the default test problem `swim.ref.in` as the sample problem on two different clusters of different architectures. One is the A64FX cluster "Ookami" and the other is a SkyLake x86 cluster from Stony Brook Exasca||ab. Seven different compilers were tested on Ookami. To avoid clutter and data overlap, we have chosen 3 representatives: ARM's GNU-based compiler based on GCC 9.3.0 , ARM's LLVM-based compiler based on 9.0.1, and Cray 10.0.1. The times are shown in Figure 4. Times on the X86 cluster using GNU compiler 10.2.1 and on Ookami using Cray 10.0.1 are shown in Fig 5. As the results shown in Fig 4, the Cray Compiler obtained the best performance on Ookami. Ookami has a clear performance advantage with a single socket of A64FX being 4.3x faster at 32 threads compared with dual socket SkyLake x86. We ascribe the slight slow down at 48 threads on SkyLake to oversubscription (there are only 40 physical cores).

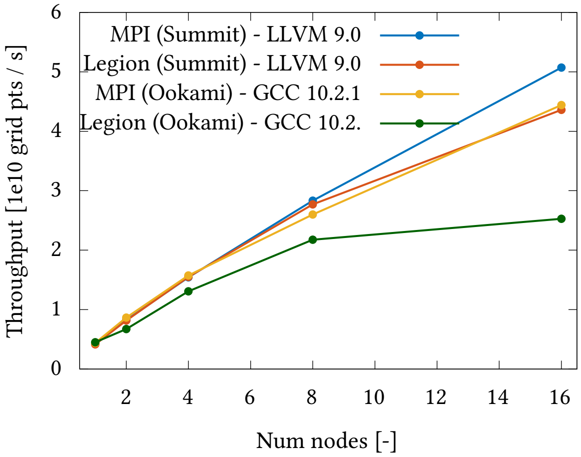

Fig. 3. Minimod: Comparison of Legion and MPI throughput on Ookami and Summit

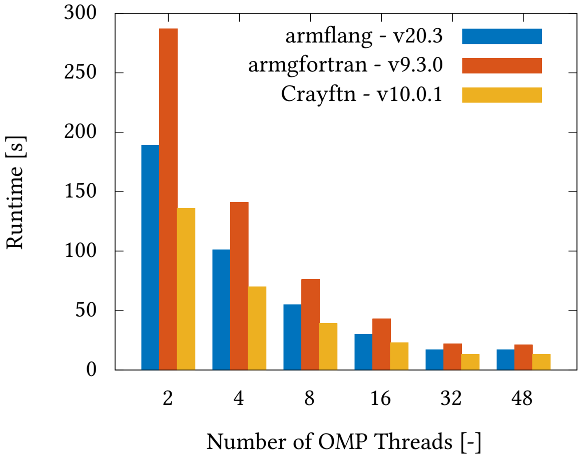

Fig. 4. SWIM: Runtime results for three compilers with SVE optimizations on Ookami.

4.2.3 PENNANT.

PENNANT is an unstructured mesh, MPI+OpenMP, physics mini-application [13] designed for advanced architecture research. It is a challenge the A64FX processor due to the lack of locality and the architectures 256 byte cache line. This is particularly evident with the largest of the mesh inputs, each having at least 3 billion points and sides, with at least 11 billion edges. MPI handles inter-node gather/scatters and reductions, while OpenMP is used for intra-node activities like mesh initialization, summing energy usage, and data allocation. To explore the scalability of PENNANT on A64FX and compare its performance to other processors, we tested one of PENNANT's smaller tests (LeblancBig).

MPI scaling experiments were run on the Ookami and SBU SeaWulf clusters, using the Cray (Ookami), Intel, and GNU (SeaWulf) compiler toolchains. Tests used an 8 node configuration, running from 2 to 32 MPI ranks/node, and

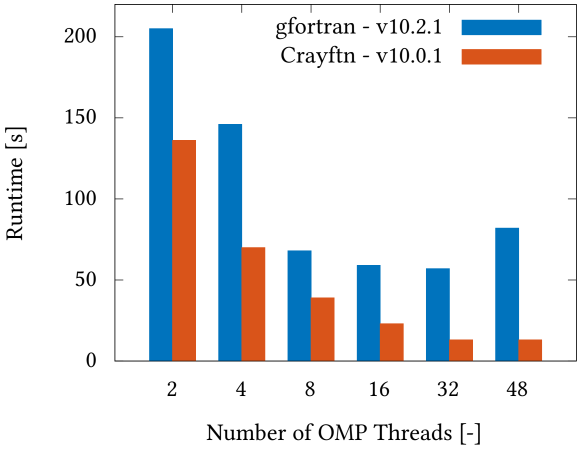

Fig. 5. SWIM: Runtime results for the Cray compiler on A64FX and the GNU compiler on dual socket SkyLake x86.

with one OpenMP thread per process. While run-times begin converging at larger process counts, the Intel system is faster, as shown in Figure 6. We also tested single-node, OpenMP scaling on Ookami, comparing it to an Intel Haswell processor, with experiments using thread counts from 1 to 64 and using the GNU compiler on both platforms to facilitate direct comparison. The runtime spike from 11 seconds to nearly 600 once more than 48 OpenMP threads are requested is attributed to oversubscription, which is handled less gracefully by A64FX due to the lack of hyperthreading and current settings for O/S thread scheduling policies .

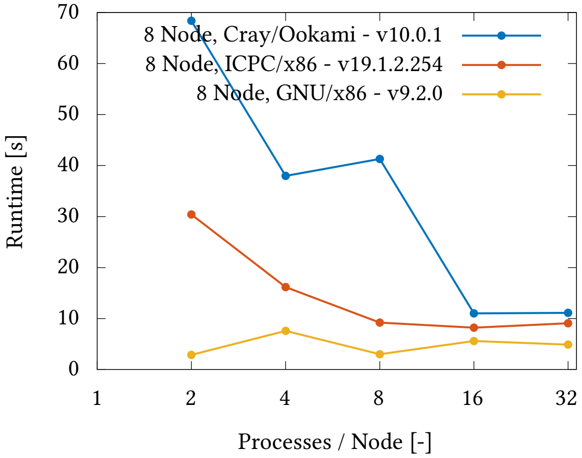

Fig. 6. PENNANT: Run-time vs. no. cores between 3 different compilers with optimizations

4.2.4 GROMACS. [22] is a molecular dynamics simulation package for modeling biomolecular systems such as proteins, DNA, RNA, and lipid membrane. The code extensively utilizes features of the C++17 language. Due to this, we were only able to build binaries with GCC(10.2.0) but not with ARM(20.3) and Cray(10.0.1) compilers and with other versions of GCC failing due to language constructs. SVE support was introduced to GROMACS in 2021 version by specialists from Research Organization for Information Science and Technology (Japan). Before that, GROMACS had optimization for ARM NEON. Below, we compare Ookami no-SIMD/NEON/SVE performances with no-SIMD/SSE2/AVX512 on dual socket 2 Intel Xeon Platinum 8160 (TACC Stampede-2). From the GROMACS benchmark set[19] we report results for a protein embedded into a membrane, consisting of 87k atoms.

On a single Ookami node, the performance for no-SIMD, NEON and SVE versions are 3.6, 13.8 and 20.2 ns/day, respectively. On Stampede-2 system the performance for no-SIMD, SSE2 and AVX512 versions are 12.7, 47.8 and 75.6 ns/day. In both cases (Ookami and Stampede 2) the gain from utilizing wider SIMD is very similar, a fascinating, 3.8 times performance increase by moving from no-SIMD to 128-bit wide SIMD and moderate 50% increase by moving from 128-bits wide to 512-bits wide SIMD. What makes it interesting is that although Ookami and Stampede 2 theoretical flops are similar and aforementioned similar scaling on SIMD width increase, the absolute performance of Ookami is significantly smaller. It is not clear whether this is due to architectural differences or immature compiler support (including the inability of GCC to vectorize math functions).

4.2.5 FLASH. The FLASH code is a large, component-based package written primarily in modern FORTRAN that has a wide user base addressing a variety of multi-scale, multi-physics applications, especially astrophysics [7, 8, 15]. The current release version of FLASH (4.6.2) is parallelized using MPI, and utilizes PARAMESH [23] to support its block-structured Adaptive Mesh Refinement (AMR) scheme. FLASH is able to tune the amount of memory per processor it uses, and be paused and restarted on a different number of cores. All of these features make FLASH a promising platform to take advantage of the A64FX's HBM, NUMA architecture, and SVE instructions.

We successfully tested multiple combinations of compilers and MPI implementations (MVAPICH 2.3.4 and OpenMPI 4.0.5), initially without optimization to assess ease of porting and correctness. The GCC (9.2.0 and 10.2.1) and NVIDIA (20.9) compilers worked out of the box, while using the ARM (20.3.0) and Cray (10.0.3) compilers just required setting proper options and MPI environment variables. We were also able to demonstrate strong scaling of our supernova simulation without optimization on 12–140 cores using the GCC 9.2.0 compiler with MVAPICH 2.3.4. However, the SeaWulf cluster currently provides over double the speedup. Our next goal is to enable SVE, however the ARM and Cray compilers generate runtime errors, and the GCC compiler provides no significant speedup. The application chosen for this study was an explosive astrophysical event known as a thermonuclear or Type Ia supernova. We are exploring the dimmest of these events, known as a Type Iax, which is thought to be produced by a pure deflagration (subsonic burning front) occurring in a special "hybrid" white dwarf [10, 12], which leaves behind a bound remnant.

4.3 XDMoD

Ookami is monitored with XDMoD [9], with the XDMoD job performance and application kernel modules also installed. The XDMoD instance is hosted in CCR's cloud computing cluster at the University at Buffalo. Ookami users can access the XDMoD instance using their Stony Brook accounts via Globus Auth. The default XDMoD install worked without requiring any customization. However, we did modify the XDMoD software to monitor A64FX-specific metrics. Performance data are collected using performance co-pilot (PCP) [39]. We used the PCP version supplied with CentOS 8 but recompiled the performance monitoring module against the latest version of `libpfm4` to log data from the A64FX hardware counters. We also backported the latest version of the module that records information from Lustre because Ookami uses a newer version of the Lustre driver than is supplied with CentOS. The XDMoD job summarization software was updated to handle the ARM SVE hardware counters, and time-series plots of the instruction rate are available in XDMoD's job viewer. Figure 7.A shows an example plot of the SVE instruction rate exported from the XDMoD job viewer.

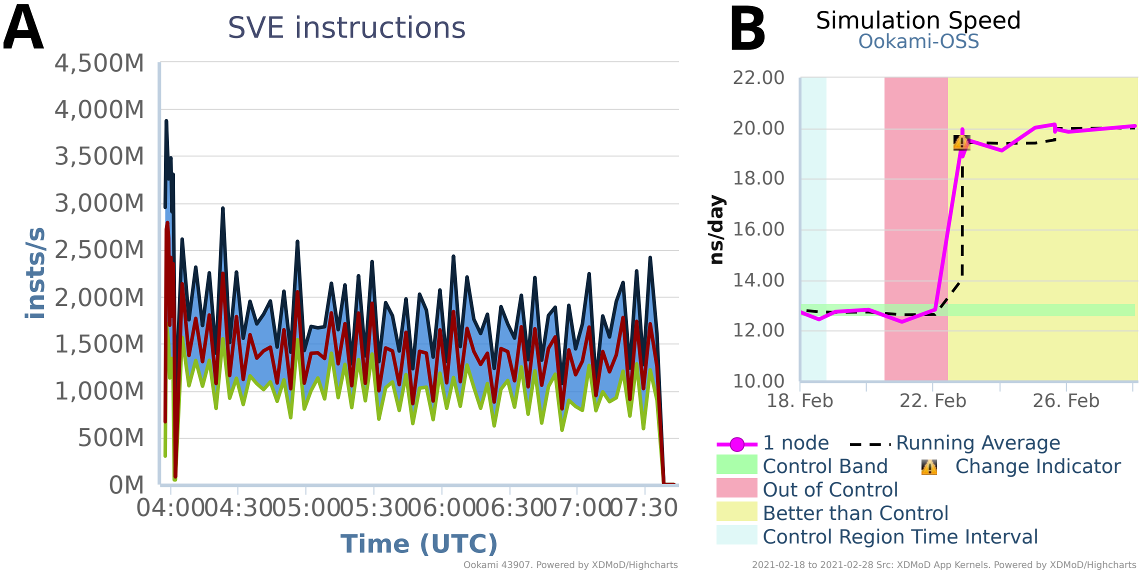

Fig. 7. A) XDMoD job viewer plot of the SVE instruction rate for a 150 node run of the MADNESS software. At each time point the SVE instruction rate for the compute nodes with the maximum, median and minimum value is shown. B) Daily execution of GROMACS application kernel shows the performance improvement due to version update from 2020.4 to 2021 (first release with SVE support).

Within XDMoD, application kernels are used to proactively monitor HPC resource performance by daily benchmarks. For Ookami, a goal is to see how the performance of benchmarks and real applications change as the compiler toolchains improve. Figure 7.B illustrates a performance improvement for the GROMACS application. We plan to add a power data collector so that we can analyse job level power usage in XDMoD, and also adding new job-efficiency analytics that take into account the extra ARM hardware counters, e.g., to identify floating-point intensive codes not using SVE.

5 SUMMARY

In conclusion, we have had very positive initial experience with A64FX. It should be viewed as a "leadership processor" that trades high performance and high power efficiency on a large class of well-vectorized scientific applications for reduced performance (especially if not vectorized) and reduced applicability (primarily due to memory capacity) on more general codes. The peak processor vector speed and peak memory bandwidth are indeed readily accessible to compiled codes that are well vectorized and pay attention to localizing memory references within a CMG. The latter is readily accomplished by running four multi-threaded MPI processes per node, with one per CMG. So far, it is living up to the expectation that most such software can deliver great performance out of the box. However, the relative immaturity of the SVE software ecosystem (again noting that we do not yet have the Fujitsu stack) makes it hard to generalize this statement.

ACKNOWLEDGMENTS

Ookami is supported by the National Science Foundation (NSF) grant OAC 1927880, XDMoD by NSF OAC 1445806, and Seawulf by NSF OAC 1531492. OpenSHMEM is funded through Los Alamos National Laboratory, Grant #367958. FLASH was developed in part by the US Department of Energy (DOE) NNSA-ASC and OSC-ASCR-supported Flash Center for Computational Science at the University of Chicago. Work involving supernovae research was supported in part by the DOE under grant DE-FG02-87ER40317. We would like to thank Total Exploration and Production Research and Technologies for their support of experimentation using MiniMod. Use of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory was supported by the Office of Science of the DOE under Contract No. DE-AC05-00OR22725. Use of XSEDE systems is supported by XSEDE grant TG-CCR120014. We also thank the MVAPICH and Open-MPI teams for their assistance in tuning and deploying their software.

REFERENCES

[1] Y Ajima. 2019. The Tofu Interconnect D for the Supercomputer Fugaku. https://www.fujitsu.com/global/Images/the-tofu-interconnect-d-for-supercomputer-fugaku.pdf

[2] M. Bauer, S. Treichler, E. Slaughter, and A. Aiken. 2012. Legion: Expressing locality and independence with logical regions. In *SC '12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. 1–11. https://doi.org/10.1109/SC.2012.71

[3] IACS Stony Brook. 2020. *Ookami*. https://www.stonybrook.edu/commcms/ookami/

[4] Barbara Chapman, Tony Curtis, Swaroop Pophale, Stephen Poole, Jeff Kuehn, Chuck Koelbel, and Lauren Smith. 2010. Introducing OpenSHMEM: SHMEM for the PGAS community. In *Proceedings of the Fourth Conference on Partitioned Global Address Space Programming Model*. 1–3.

[5] J Dongarra, P Luszczek, and Y Tsai. 2021. *HPL-AI benchmark*. https://icl.bitbucket.io/hpl-ai/

[6] J. Dongarra, S. Otto, M. Snir, and D. Walker. 1995. An Introduction to the MPI Standard.

[7] A.C. Calder et al. 2002. On Validating an Astrophysical Simulation Code. *The Astrophysical Journal Supplement Series* 143 (2002), 201–229.

[8] A. Dubey et al. 2013. The Software development process of FLASH, a multiphysics simulation code. In *2013 5th International Workshop on Software Engineering for Computational Science and Engineering (SE-CSE)*. 1–8. https://doi.org/10.1109/SECSE.2013.6615093

[9] J. Palmer et al. 2015. Open XDMoD: A tool for the comprehensive management of high-performance computing resources. *Computing in Science and Engineering* 17, 4 (2015), 52–62. https://doi.org/10.1109/MCSE.2015.68

[10] M. Kromer et al. 2015. Deflagrations in hybrid CONe white dwarfs: a route to explain the faint Type Iax supernova 2008ha. *Monthly Notices of the Royal Astronomical Society* 450, 3 (05 2015), 3045–3053. https://doi.org/10.1093/mnras/stv886 arXiv:https://academic.oup.com/mnras/article-pdf/450/3/3045/18513746/stv886.pdf

[11] P. Luszczek et al. 2006. The HPC Challenge (HPCC) Benchmark Suite. In *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing* (Tampa, Florida) *(SC '06)*. Association for Computing Machinery, New York, NY, USA, 213–es. https://doi.org/10.1145/1188455.1188677

[12] R.J. Foley et al. 2013. TYPE Iax SUPERNOVAE: A NEW CLASS OF STELLAR EXPLOSION. *The Astrophysical Journal* 767, 1 (mar 2013), 57. https://doi.org/10.1088/0004-637x/767/1/57

[13] Charles R. Ferenbaugh. [n.d.]. *PENNANT: An Unstructured Mesh Mini-App for Advanced Architecture Research*. https://www.osti.gov/biblio/1079561-pennant-unstructured-mesh-mini-app-advanced-architecture-research

[14] MPI Standardization Forum. 1994-2021. *MPI Standardization Forum Website*. https://www.mpi-forum.org/

[15] B. Fryxell, K. Olson, P. Ricker, F. X. Timmes, M. Zingale, D. Q. Lamb, P. MacNeice, R. Rosner, J. W. Truran, and H. Tufo. 2000. FLASH: An Adaptive Mesh Hydrodynamics Code for Modeling Astrophysical Thermonuclear Flashes. *The Astrophysical Journal Supplement Series* 131 (2000), 273–334.

[16] Fujitsu. 2019. *Fujitsu Green-500 award*. https://bit.ly/382Ls9Y

[17] Fujitsu. 2021. *A64FX*. https://www.fujitsu.com/global/products/computing/servers/supercomputer/a64fx/

[18] Inria. 2021. *KNEM Website*. https://knem.gitlabpages.inria.fr/

[19] C. Kutzner, S. Páll, M. Fechner, A. Esztermann, B.L.. de Groot, and H. Grubmüller. 2019. More bang for your buck: Improved use of GPU nodes for GROMACS 2018. *Journal of Computational Chemistry* 40, 27 (2019), 2418–2431. https://doi.org/10.1002/jcc.26011 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/jcc.26011

[20] Sandia National Laboratories. 2018. *Astra, ARM-based supercomputer*. https://www.scientific-computing.com/news/sandia-labs-supercomputer-fastest-arm-based-top500-system

[21] Oak Ridge National Laboratory. 2020-2021. *ORNL Wombat Cluster*. https://www.olcf.ornl.gov/olcf-resources/compute-systems/wombat/

[22] Lindahl, Abraham, Hess, and van der Spoel. 2021. *GROMACS 2021 Source code*. https://doi.org/10.5281/zenodo.4457626

[23] P. MacNeice, C. Olson, K. M. Mobarry, R. de Fainchtein, and C. Packer. 1999. PARAMESH: A Parallel Adaptive Mesh Refinement Community Toolkit. *NASA Tech. Rep. CR-1999-209483* (1999).

[24] S. McIntosh-Smith, J. Price, T. Deakin, and A. Poenaru. 2019. A performance analysis of the first generation of HPC-optimized Arm processors. *Concurrency and Computation: Practice and Experience* 31, 16 (2019), e5110.

[25] J. Meng, A. Atle, H. Calandra, and M. Araya-Polo. 2020. Minimod: A Finite Difference solver for Seismic Modeling. *arXiv* (2020). arXiv:2007.06048 [cs.DC] https://arxiv.org/abs/2007.06048

[26] MVAPICH2. 2001-2021. *MVAPICH2 Website.* https://mvapich.cse.ohio-state.edu/

[27] Open-MPI. 2004-2021. *Open-MPI Website.* https://www.open-mpi.org/

[28] OpenSHMEM. 2011-2021. *OpenSHMEM Website.* http://www.openshmem.org/

[29] OpenUCX. [n.d.]. *OpenUCX Website.* https://www.openucx.org/

[30] A. Petitet, R. C. Whaley, J. Dongarra, and A. Cleary. [n.d.]. HPL - A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers. http://netlib.org/benchmark/hpl.

[31] E. Raut, J. Anderson, M. Araya-Polo, and J. Meng. 2021. Porting and Evaluation of a Distributed Task-driven Stencil-based Application. In *Proceedings of the Twelfth International Workshop on Programming Models and Applications for Multicores and Manycores* (Virtual Event, Republic of Korea) *(PMAM '21)*. Association for Computing Machinery, New York, NY, USA, 10 pages. https://doi.org/10.1145/3448290.3448559

[32] E. Raut, J. Meng, M. Araya-Polo, and B. Chapman. 2020. Evaluating Performance of OpenMP Tasks in a Seismic Stencil Application. In *OpenMP: Portable Multi-Level Parallelism on Modern Systems*, K. Milfeld, B.R. de Supinski, L. Koesterke, and J. Klinkenberg (Eds.). Springer International Publishing, Cham, 67–81.

[33] RIKEN. 2020. *Fugagku Graph-500.* https://www.r-ccs.riken.jp/en/award/20201117_graph500

[34] RIKEN. 2020. *Fugaku performance award.* https://www.riken.jp/en/news_pubs/news/2020/20200623_1/

[35] RIKEN. 2021. *Center for Computational Science.* https://www.r-ccs.riken.jp/en/

[36] RIKEN. 2021. *Fugaku supercomputer.* https://www.r-ccs.riken.jp/en/fugaku

[37] RIKEN. 2021. *RIKEN.* https://www.riken.jp/en/

[38] M. Sato. 2019. *Overview of the Post-K processor.* http://www.jicfus.jp/jp/wp-content/uploads/2018/11/msato-190109.pdf

[39] Silicon Graphics Inc, Aconex, and Red Hat. 2000. Performance Co-Pilot (PCP). https://pcp.io.

[40] Nikolay A. Simakov, Joseph P. White, Robert L. DeLeon, Steven M. Gallo, Matthew D. Jones, Jeffrey T. Palmer, Benjamin Plessinger, and Thomas R. Furlani. 2018. A Workload Analysis of NSF's Innovative HPC Resources Using XDMoD. arXiv:1801.04306 [cs.DC]

[41] SPEC. [n.d.]. *SWIM benchmark page.* https://www.spec.org/cpu2000/CFP2000/171.swim/docs/171.swim.html

[42] HPCG team. 2021. *HPCG benchmark.* https://www.hpcg-benchmark.org/

[43] Top500òrg. 2020. *Fugaku Top-500 award.* https://bit.ly/2RWivXo

[44] Ohio State University. 2001-2021. *Ohio State University Micro-Benchmarks.* https://mvapich.cse.ohio-state.edu/benchmarks/

[45] XPMEM. [n.d.]. *XPMEM Website.* https://github.com/hjelmn/xpmem