Educating HPC Users in the use of advanced computing technology

Eva Siegmann, Alan Calder, Catherine Feldman, Robert J. Harrison
Institute for Advanced Computational Science, Stony Brook University, USA
Email: {eva.siegmann,alan.calder,catherine.feldman,robert.harrison}@stonybrook.edu

Abstract*—We examine a multi-modal approach to educating and training users of an advanced computing technology testbed at the Institute for Advanced Computational Science at Stony Brook University. Ookami [1] provides researchers worldwide with access to 176 Fujitsu A64FX compute nodes, this being the same processor technology powering the Japanese Fugaku supercomputer, the fastest computer in the world since June 2020. However, achieving high-performance on this Arm-based, leadership computing technology requires that users be familiar with details of computer architecture, performance analysis and modeling, and high-performance programming models that are commonly omitted in introductory programming courses. Indeed, regardless of their seniority, many of the testbed users are surprisingly unfamiliar with basic concepts such as vectorization, pipelining, latency/bandwidth, roofline models, computing energy/power, threads, and non-uniform memory access. These same concepts also pervade mainstream x86 technologies, so this is of widespread concern. Due to the national/global nature of our user community that is also very diverse in both discipline and experience, the inability to offer formal classes, and our experience that most people do not tend to read online documentation or training materials in sufficient depth, we have consciously employed multiple approaches that heavily emphasize (online) personal interactions and transfer of skills. Online documentation has been organized around best-practices and FAQs; twice-weekly hackathons and office hours via Zoom enable deep dives by both the team and the user community with multiple broad benefits; a Slack channel provides both real time and archived answers and discussions; and workshops, training and webinars target community needs as they arise. The perspective that these tools are being used in an educational setting rather than just for project communication makes them more effective and contributes to community success.

***Index Terms*—A64FX, high-performance computing, educational program**

I. Introduction

High-performance computing (HPC) is differentiated from mainstream computing by raising ”performance” to a level-one correctness concern. A correct program should not just produce a valid result, it should execute “efficiently.” Even defining HPC “performance” or “efficiency” is not straightforward, but everyone agrees that a national cyber-infrastructure costing several hundred million dollars should be used efficiently, and that is an adequate starting point for a conversation with users. Unfortunately, due to the diverse nature of scientific algorithms (so that most are not readily captured by optimized libraries) and the low-level of HPC programming models, this elevation of performance to a level-one concern means that most programmers *and users* must be aware of many details of system architecture and software as well as how these details impact performance. The impacts can be huge — a programmer can find well over a 100x speed up by switching from a serial implementation to a fully-pipelined, vectorized, and threaded version; and a user can find factors of 2-10, or more, by switching compilers or changing the options or environment variables controlling OpenMP and MPI. Black-box recipes and best practices do not fully address efficient utilization of even a single computer system, and, more significantly, do not produce a community prepared for today’s experimental hardware becoming mainstream or inevitable future changes in computer technology. As a computer testbed project funded by the National Science Foundation, we perceived from the outset a special responsibility for advancing communal interests of which Ookami is just a small part.

II. The Ookami Testbed

A. Technical Details

Ookami is an Apollo 80 system funded by the NSF (National Science Foundation) under grant OAC 1927880. It consists of

- 176 Fujitsu A64FX nodes running at 1.8 GHz each with 32GB of high-bandwidth (1 TB/s) memory and a 512 GB SSD
- 2 dual socket Thunder X2 (64 cores) each with 256 GB memory
- 1 dual socket Intel Skylake (36 cores) with 192 GB memory
- 1 dual socket AMD Milan (64 cores) with 512 GB memory
- 2 Nvidia V100 GPUs

Ookami runs CentOS8 and a high-performance Lustre filesystem provides around 800TB of storage served by a Cray ClusterStor E1000. The HDR 200 GB/sInfiniband network (100 GB/s host adaptors due to PCI limitations) is configured as a full fat tree. The two login nodes are each dual-socket ThunderX2 with 64 cores (256 threads) and 256 GB memory. Since they are also Arm-based, the issue of cross-compilation is reduced. The heart of the system is formed by the 176 A64FX compute nodes. A node consists of 48 cores (with private 64KB L1 cache) grouped into four core memory groups (CMG) of 12 cores that share an 8MB L2 cache is directly connected to on-package stacked, high-bandwidth memory (256GB/s). There is no L3 cache. The CMGs are connected via a crossbar, forming a non-uniform memory access (NUMA) architecture. The A64FX is the first production processor to implement the new ARM scalable vector extension (SVE) single-instruction multiple-data (SIMD) instruction set. A novel design feature of SVE is that it can be employed in a vector-length agnostic (VLA) manner so that binaries are portable between SVE hardware implementations that differ in the width of the vector registers.

The system uses a module environment which allows users to easily use different toolchains and libraries. Ookami offers access to the GNU, Arm, Cray, and Fujitsu compilers and toolchains. Different flavors of MPI are available (various versions of OpenMPI and MVAPICH). For job submission, the SLURM workload manager version 19.05.7 is in place.

B. Users and Projects

Ookami is open to researchers worldwide working in both academia and industry. Users can apply for testbed or production projects. Testbed projects are primarily for porting and tuning codes and doing benchmark calculations to demonstrate capability. Those projects can use up to 15,000 node hours (1% of annual capacity). Once this is done and the performance of the application on the system is proven, users can move on to a production project in which they can use up to 150,000 node hours (10% of annual capacity). During the first 2 project years, starting in January 2021, testbed projects have higher priority on the system.

Since January 2021, Ookami has been open to the public. By August 2021, 53 projects were granted on the system. Most of these projects are from universities within the U.S. — 92.3% of all projects are from academia and 7.7% from industry. Although Ookami is mostly advertised within the U.S., it is open to researchers worldwide — 90.4% of projects are from institutions in the US and the rest from Europe (see figure 1).

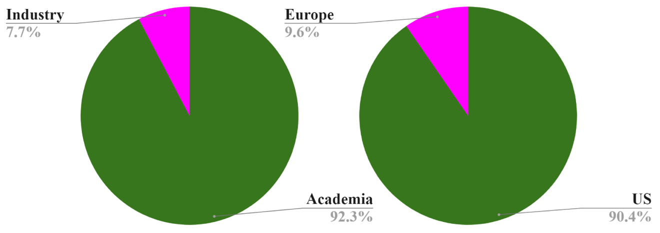

Fig. 1. Projects running on Ookami. (Status August 2021)

In total, Ookami has currently 166 users. The expertise of these users is wide-ranging. Some have been HPC users for decades while some are very new to scientific computing. The efforts in educating the users are laid-out to offer support to these various levels of expertise. Figure 2 shows some demographic information of the Ookami users.

C. Opportunities

The three compelling features of Ookami for campus-level science and engineering are the very high-bandwith memory,

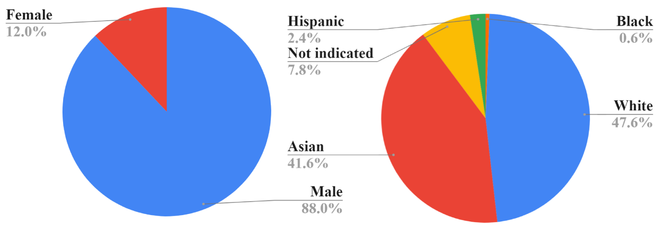

Fig. 2. Metrics of the 166 Ookami users. (Status August 2021)

the innovations in the SVE instruction set, and the remarkable scalability of the intra-node architecture that routinely demonstrates near linear speed up with thread count. At the leadership scale of Fugaku, power efficiency also becomes preeminent.

Most applications also just compile and run out of the box, allowing for the usual adjustment of compiler flags and libraries.

D. Challenges

Compilers and libraries: Achieving satisfactory performance requires users to dig deep into the different compilers and libraries. The GNU compiler that can generate excellent SVE code is let down by an unvectorized math library, forcing many users seeking performance to employ commercial compilers from either Fujitsu, Cray or ARM, all of which involve different compromises in terms of standard compliance, performance, morass of options, and correctness.

Vectorization and threading: Most users can rely upon compiler autovectorization, but even this rapidly requires understanding architectural details including predication and vector register width, instruction throughput, cache architecture, etc. Many additional details are necessary to fully understand performance differences relative to x86.

NUMA: Mapping of threads and processes is an end-user challenge, made worse by lack of standardization and shifting and eccentric choices by runtime developers.

Evolving best/correct practices: The immature (in terms of performance) software stack is constantly evolving in nontrivial ways, requiring users stay informed. This is unfamiliar to most users of x86 clusters that rarely receive significant updates.

A huge glossary: Of both concepts and relevant tools for analayzing and modeling performance.

Low bandwidth to users: Testbed users are not enrolled in class — we are competing for their attention and must provide immediately relevant, high-value, dense yet accessible material.

A few expert users just need access to the documentation and pointers to the FAQ and best practices, to which they rapidly start contributing. Nearly everyone else needs education in the foundations of computer architecture relevant to HPC, and then training in how to use tools and techniques. In the following we examine some of our activities in these directions.

III. New Opportunities for Education

A. Slack Channel

Normally regarded as a tool for team communication, this has become one of our most powerful educational tools. Sustainably monitored by both team members and growing cadre of users, we can immediately and directly respond to questions and, as appropriate, pivot the conversation to address either the fundamental issues or overarching concepts. For instance, a question of clock speed (Ookami's is fixed at 1.8 GHz) was deliberately morphed into a conversation about the different power efficiency concerns of edge and leadership computing, and the problems in benchmarking on x86 processors with dynamic clock speeds. Moreover, a "personal" conversation with a user routinely pulls in a wider audience that brings different/deeper technical perspectives and follow-up questions, and is publicly archived. It is routine that we refer users to these archived conversations. The average number of posts is 288 per month.

B. Office Hours — what's in a name?

There are twice-weekly, 2-hour Ookami office hours that are virtual meetings via Zoom. Originally called hackathons, since the original focus was on deep code dives and actually writing code, we renamed them office hours to address poor attendance — it turned out that most users interpreted hackathon to be an invitation only or focused event, rather than being open to all.

In their current form, the Ookami office hours give users the opportunity to get in direct contact with the support team as well as with other users. Users can join and ask specific questions, do deep code dives, or just present their work on the system and get feedback from others. Experience and user-feedback show that these meetings are enormously helpful, and many users seem to greatly appreciate the sense of community it builds (as do we!). We also routinely use these time slots for scheduled seminars and training events. Again, approaching these as actual class office hours that serve as much a pedagogical purpose as user support makes the interaction much more impactful. For instance, rather than stopping at the discovery that the performance of a code is memory-bandwidth limited, we can introduce roofline performance models and provide pointers to further reading.

However, unlike Slack, the office hours are not recorded because large volumes of unstructured video are mostly useless. Thus, these conversation are ephemeral and different media must be employed to document key points.

C. Ticketing System

More conventional user support, installation requests, or reporting issues, are submitted via a ticket system at https://iacs.supportsystem.com/. Those tickets are handled by the cluster's support team. On average, users wait about a day for a response to their ticket, thus this mechanism is not appropriate for most other purposes.

D. Ookami Website

In addition to the above interactive components, we provide a project website, www.stonybrook.edu/ookami. It gives information about what Ookami is, how to get accounts on the system, the various projects which are using the system, the system status, upcoming events like office hours or webinars, and a detailed FAQ section. The FAQ section discusses the important topics and best practices, e.g., how to access the cluster, the module environment, compilers and toolchains, slurm, vectorization, and storage. It also gives users access to documentation like the A64FX specification, datasheet, and microarchitecture manual; Arm and Cray documentations; as well as recordings and material of any events hosted on Ookami.

IV. Webinars and Seminars

To educate users more deeply on specific topics, we routinely organize webinars and seminars from team and community members. These are primarily focused on enabling users to port and tune their codes for the A64FX architecture, but we also use these as opportunities to put A64FX and its software ecosystem in a broader context. In the following section, webinars held in the time from January till August 2021 are described. The materials and recordings of all webinars are available on the documentation section of the Ookami webpage.

A. SVE Hackathon

In February 2021, a SVE (scalable vector extension) hackathon held by Arm was hosted on Ookami. The goal of this event was to introduce SVE as a tool for enhancing scientific application codes, and to educate users in using performance engineering tools for SVE. Every participant had an account on Ookami and enough compute nodes were provided to ensure that everybody could participate in the hands-on exercises. This hackathon lasted three days and was a hybrid event with presentations and hands-on exercises. The first day gave introductions to the Arm ecosystem and the Fujitsu A64FX processor. This theoretical block was followed by hands-on exercises. Optimal code generation and differences in the various compilers were studied with the example of FMLA (floating-point fused multiply-add) and stream calculations. At the end of the first day, power measurements on A64fX were discussed. The second day covered the fundamentals of SVE. Participants had the chance to test various small test programs and use them for learning more on how to use SVE. The compilers and their capabilities of vectorizing codes were discussed in great detail. This topic was backed up with hands-on exercises. The last day of the hackathon was used to talk about the compiler intrinsics for SVE, complex arithmetics, data movement, and the Arm profiling tools.

A total of 31 users participated on the event. This number was very well suited for the hackathon as it was big enough to be a critical mass and small enough to still allow interaction between the users.

B. XDMoD

In March 2021, the team of the Center for Computational Research at the University at Buffalo presented their software XDMoD [2], which is monitoring the job performance on Ookami and also has the application kernel module installed. All users of Ookami can access the XDMoD instance and get detailed information on the system's overall utilization and the performance of their own jobs.

The performance data, which can be accessed via the online XDMoD interface [3] is collected via performance co-pilot (PCP) [4]. It can also handle the Arm SVE hardware counters, and time-series plots of the instruction rate are available in the job viewer. XDMoD runs daily benchmarks to actively monitor the performance. Instructions on how to set up these benchmarks and how to utilize the various compilers for those setups were given to the users.

This webinar educated Ookami users in accessing the XDMoD instance, monitoring the overall performance of the system, and getting detailed information on specific jobs.

C. TAU

In April 2021, a webinar about the TAU performance system [5] took place. TAU is a portable profiling and tracing toolkit for performance analysis of parallel programs written in Fortran, C, C++, UPC, Java, and Python. In this webinar, users learnt about the TAU architecture.

Profiling applications with TAU gives information about how much time is spent in each routine and loop, what is the contribution of each statement within a loop, time spent in OpenMP loops, data cache misses, extent of vectorization, memory usage, power usage, etc. In the webinar, hands-on exercises were used for allowing users to get familiar with the TAU interface.

As profiling is crucial in getting good performance, especially when using a new processor, this webinar was very helpful for the Ookami users.

D. Parallelware Analyzer

As the Ookami project is not only collaborating with academia but also with industry, a collaboration with the Spanish company Appentra evolved. They are developing the Parallelware Analyzer [6] and presented it to Ookami users in the form of a webinar in July 2021.

The Parallelware Analyzer is a static code analyzer specializing in performance. During this event, users learnt how they can use the software to create performance reports of their application. During a hands-on exercise it was shown how the profiler can identify performance bottlenecks. It can also modify the source code by adding OpenMP pragmas. The webinar used a MATMUL implementation as an example for showing Parallelware Analyzer's capabilities.

Due to the collaboration with Appentra, the software is available to all users of Ookami. Similarly, Ookami sends regular reports about the utilization of the software to Appentra. This allows them to tailor their software to the needs of HPC users.

E. OSACA and likwid

In July 2021, the team from the Erlangen National High Performance Computing Center at the Friedrich-Alexander University Erlangen-Nuremberg held a webinar and online training about their tools OSACA and likwid. This webinar started with an introduction to the SpMV (Sparse matrix-vector multiplication) theory and showed how the tools can be used for that.

likwid is [7] a toolsuite of command line applications. It consists of a variety of tools for e.g, showing thread, cache and NUMA topology; configuring and reading out hardware performance counters; pining threaded applications. OSACA (Open Source Architecture Code Analyzer) [8], [9] can analyze throughput and latency, and insert markers for deeper analysis.

As a result of this seminar and the associated training materials, multiple users have now started exploring use of these tools that were previously mostly only known to HPC experts.

V. Use Case - FLASH on Ookami

Due to the versatility of its applications, its long history of running on cutting-edge platforms, its limited ability to take advantage of accelerators such as GPUs, and the interests of the investigators, the FLASH code was chosen as one of the acceptance applications for Ookami. The research group tasked with porting FLASH to Ookami is mainly associated with the astrophysics community, and its members have differing levels of experience with HPC systems. Therefore, this task presented both many challenges and opportunities to learn. This use case seeks to demonstrate how the broad Ookami collaboration provides a supportive community experience that an application developer/user can expect when starting a project on Ookami.

A. The FLASH Code

The FLASH code is a highly parallelized, component-based scientific software package for addressing a variety of multi-scale, multi-physics applications [10]–[14]. While originally written to simulate astrophysical phenomena [15]–[19], such as the Type Ia supernovae currently under investigation by this research group, recent developments in FLASH have extended its capabilities [12], [20]. FLASH is extensively used in in modeling high powered laser experiments, including those at the Omega Laser Facility; high-energy density physics; fluid-structure interactions; and for more general problems [21]. The current release version of FLASH (4.6.2) is parallelized primarily through MPI, although some solvers have been modified to take advantage of threading [13]. Development toward a more general design for better allowing threading continues [22]–[24].

FLASH's success and development is intertwined with its history of being run on new architectures. FLASH was developed at the Flash Center for Computational Science at the University of Chicago that was originally funded by the Department of Energy's Accelerated Strategic Computing Initiative

[25], which allowed the group access to unclassified versions or partitions of the platforms at the Defense Program National Laboratories as they became available [26]. Highlights of the development of FLASH included winning the SC2000 Gordon Bell Prize, Special Category, for reactive fluid flow simulations using AMR that achieved 238 GFlops on 6420 processors of ASCI Red [27], performing a study of weakly compressible stirred-turbulence at an unprecedented resolution [19], [28] on the IBM BG/L machine to be commissioned at LLNL in 2005, and serving as one of of the formal acceptance tests for Intrepid, the IBM BG/P, and MIRA, the IBM BG/Q, machines at the Argonne Leadership Computing Facility [13].

B. Previous Experience

The P.I. of the astrophysics research group working with FLASH is one of its developers. The P.I. has written and incorporated modules with significant additional physics capability into the code, performed validation and verification studies, and played a large role in porting and tuning the code on the architectures highlighted above (see Section V-A). Also, a few decades ago, the P.I. has run dissertation simulations on traditional vector Cray machines, giving experience with vectorizing codes that proved surprisingly useful for this project.

The graduate student of the group had no prior experience in HPC. Until now, the student had only run FLASH on a cluster and made small changes to the problem, as the main focus was creating analysis and visualization scripts to compare simulations to observations. The student had only ever used the GCC compiler; had a very basic understanding of MPI; and had no knowledge of code profiling, porting code to new architectures, tuning code for optimization, vectorization, or even the parts of a computer.

In the following, the student describes in her own words the experience of starting to use Ookami.

C. The Learning Process

The resources provided by the Ookami collaboration allowed us to easily share information, learn new techniques, and create a strong community. Ookami Office Hours provided real-time help from experts in different fields. Over Zoom, we could easily share our screen and be provided with instant feedback when debugging. This greatly sped up the process of porting our code; we learned from our more experienced collaborators instead of rediscovering information on our own. And when faced with a new problem, it was able to be solved with suggestions from every member on the call, who brought their different viewpoints together. This more personal way of collaborating led to even greater involvement in our project: a graduate student on the Ookami team in Stony Brook's computer science department helped us perform scaling studies and MPI implementation comparisons, and the P.I. of the Ookami project set out to help us with vectorization. Getting to meet our collaborators face-to-face allowed us to get to know them, making Office Hours a comfortable and enjoyable environment. The welcoming and supportive atmosphere made it easy to ask questions, from basic questions about how to log in to more complicated ones such as vectorizing specific loops. We were also enthusiastically shown helpful tips to speed up our workflow, such as saving our loaded modules for easy restoration and our new favorite command line shortcuts `Ctrl-r` and `!`.

In between the twice-weekly Office Hours, the active Slack channel was an invaluable resource. Every member of the Ookami collaboration is on the Ookami Slack channel. As Slack is commonly used by many research groups, it was a familiar way of communicating. Due to this and the welcoming and supportive atmosphere, the Slack channel is widely used by the collaboration. Important information, such as upcoming events and FAQs, are pinned to the channel for easy access. We are able to search through previous messages for answers to our questions before posting new ones, and it usually takes less than an hour for someone to respond in the group chat. Through Slack, we can easily share screenshots, error messages, code snippets, and even entire files if necessary. This allowed other collaborators to write their own mini-examples to test the feature in question, and see if the issue was with the machine or with the user. In the former case, issues could be easily reported through the ticketing system, but more often than not the core Ookami team would quickly respond on Slack and address the issue. This kind of support was important on Ookami, and worked both ways, as the testbed would also undergo frequent changes and the software stack would evolve. Changes such as updating and moving modules; adding queues; changing compiler licenses; or implementing data limits and storage requirements were promptly reported through the Slack channel, which provided the users with instant feedback. Additionally, Slack gives the option to message any member of the team individually, allowing for deeper discussions and encouraging collaboration.

As the Ookami user base grew, so did the opportunities for formal training in the form of webinars. We were able to learn to use software and tools from their creators and developers, who were excited that their work was being used on a new platform and were consequently open to feedback, suggestions, and questions. These workshops helped us learn to use the tools that we are currently using to profile FLASH. The workshops had varying levels of assumed previous knowledge, which is at times too high for an application developer/user in another field. However, questions are encouraged, and the other resources mentioned above provided mediums to discuss what was learned by others in more general terms. The recorded videos available on the website are also helpful resources to review.

Overall, the Ookami group has created a communication model that makes users feel welcome and comfortable, encourages and enables collaboration, and is able to be scaled to many users. Both Office hours and the Slack channel benefit from having a large user base; the more questions that are asked and the more people there are to answer, the better. Any common issues and solutions that the group finds are documented on Ookami's growing website, where FAQs and

Getting Started Guides are continuously updated to to make the user experience as smooth as possible.

D. Ongoing and Future Work

We have been able to run FLASH with each of Ookami's available compilers and MPI implementations, shown in Table I. The main challenge we faced while the software stack was evolving was how to pair each compiler with the properly compiled MPI implementation, an issue that was solved by many of our collaborators both at Office Hours and over Slack, and is now documented on the website for other users to learn from. We have chosen optimal compiler flags, performed basic profiling, and completed an initial scaling study. The next version of FLASH is being prepared for release, and we hope to contribute to it by vectorizing its main routines, which we are identifying by profiling the code with Arm MAP.

TABLE I
SUMMARY OF COMPILERS, MPI IMPLEMENTATIONS, AND ATTEMPTS AT USING SVE TESTED WITH THE FLASH CODE.

Compiler	MPI Implementation	Additional SVE Flags
GCC 9.2.0	MVAPICH 2.3.4	None
	Open-MPI 4.0.5	
GCC 10.2.1	MVAPICH 2.3.4	`-O3 -mcpu=a64fx`
	Open-MPI 4.0.5	
NVIDIA 20.9	Open-MPI 4.0.5	None
Cray 10.0.3	MVAPICH 2.3.4	Load `cce-sve` module
	Open-MPI 4.0.5	
Arm 20.3.0	MVAPICH 2.3.4	`-O3 -mcpu=a64fx`
	Open-MPI 4.0.5	

Though our interactions with other members of the Ookami group, we were asked to collaborate with another research group in Stony Brook's computer science department, who is interested in comparing the interconnects of Ookami and Fugaku. This group had recently completed a study of OpenMP applications on both of these platforms [29], and the next step is to similarly study the performance of applications that use MPI, including FLASH. We are performing strong and weak scaling studies with FLASH's test problems and taking a deeper look at communication patterns. This collaboration brings the experience of two fields together, and both of our groups benefit from the knowledge of the other.

Instead of just asking questions, we find that we are now able to answer them. We contribute to documentation and presentations about getting new users started on Ookami, especially those with no previous knowledge of HPC. As we learn, we add to our continuously updated and detailed document about how to port and run FLASH on Ookami. It contains links to the Ookami website for more general information as well as more specific instructions for running our code, including editing Makefiles and loading the proper modules. It also shows all of the results from our scaling studies, compiler comparisons, and lessons learned along the way. We shared this both internally with our group and with the Ookami support team. This not only helps ease new group members in to using Ookami to do science with FLASH, but also provides a good resource for the support team to see what types of tests we are performing and what questions we most often ask.

A great success for the entire Ookami community was seeing how helpful these resources were to a brand new user in our group. A new summer intern in the IACS's NSF-funded REU (Research Experiences for Undergraduates) program, who had no previous experience with HPC, was able to quickly come up to speed with running FLASH on Ookami by reading the FLASH document we wrote as well as the FAQ pages on the website. The student was also involved on Slack, where answers to problems were quickly answered. This summer, this undergrad used 14,664 CPU hours for production-quality FLASH runs on Ookami. This shows that not only did we learn how to use Ookami, but we were also able to explain it to a new student who had no experience at all in HPC. While not a formal assessment, this success demonstrates that we truly understand what we have learned, and that the Ookami team's efforts at educating our group were extremely well done. We hope to be able to bring our newfound experience to the Ookami community and any application developers who wish to join us.

VI. ASSESSMENT

Assessment of the documentation and training materials consists of surveys of new users, continuous monitoring of discussion and tickets for improving documentation and training materials, and will include surveys of users that are presently being formulated.

Assessment of the Ookami project educational materials begins with surveys sent to new users approximately two weeks after they receive their accounts. This survey aims to get a feeling of how users get by the account creation process, the initial login and compiles, and their experience with the software environment. Users are asked about their overall experience with Ookami. Figure 3 shows their answers. All users confirm that their initial experience is either very good or good.

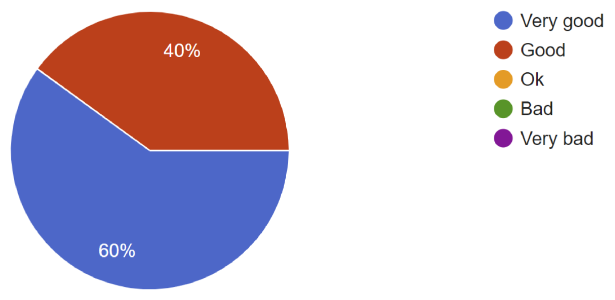

Fig. 3. Answers of users about their overall experience with Ookami.

Asked what they like about the system users highlight: the new architecture; the high bandwidth memory; the easy allocation process; very supportive staff; Slack channel; hackathons and webinars. The users also get the opportunity to mention anything they are missing. Just 5 out of 166 answered this question. They mentioned that they would like to have specific software on Ookami, e.g for visualization. Users who did not yet attend any events (webinar, hackathon, office hours) mentioned that they do not have time. Therefore, all webinars and hackathons are recorded and the recordings together with the slides and other material are available on the Ookami webpage.

Another aspect of improving training materials and documentation is the continuous monitoring the slack channel discussion and tickets, as well as direct feedback from users, by research staff. Improvements and additions to documentation and training materials are thus continuously made following interactions with users.

Similarly, research staff also monitor the slack channel, bug reports, and tickets for deficiencies in documentation and training materials and for deficiencies in management procedures. An example was the confusion caused early on by the rapidly changing software stack. Ookami uses modules to allow users to modify their environments as needed for their applications. Lists of modules, some of which were hidden, were rapidly changing when Ookami first came on line, and that resulted in considerable confusion among the users. As a result, the pace of changes was slowed and additional attention was paid to documenting module lists and the software stack.

Finally, plans are for formal surveys of users. Questions will include the ease of finding information (e.g the usefulness of FAQs), the overall utility of each category of training material (Slack channel, documentation, ticket submission, Office Hours, and webinars), and opportunities for direct feedback from users. Surveys are expected to be completed and sent to users by end of the calendar year 2021.

VII. Summary and Conclusion

The Ookami testbed provides researchers access to a new computing technology, A64FX. The expertise of the users ranges from experienced HPC users to undergraduate students, e.g., students funded by the NSF REU program. The office hours, the slack channel, the ticketing system and the website are in place since the beginning aiming to provide support to all users, independent of their level of expertise. During the first weeks and months it turned out that the usage of those different support channels depends a lot on the users' personal preferences. Some prefer a personal conversation during the office hours while others favor written communication, or just the supply of documentation on the website. The level of expertise does not have a significant impact on what kind of support users prefer. The topics of the webinars we are offering is based on the feedback we get from user interaction.

To enable all users, independent of their level of expertise or personal preferences, to port and tune their codes for A64FX, to use Ookami in an efficient way, a multi-modal education and training strategy is employed, which assessment suggests is greatly appreciated by users.

Acknowledgment

Ookami [1] is supported by the National Science Foundation (NSF) grant OAC 1927880, XDMoD by NSF OAC 1445806.

The FLASH code was developed in part by the DOE NNSA ASC and DOE Office of Science ASCR supported Flash Center for Computational Science at the University of Chicago. Our work with the FLASH code was supported in part by the US Department of Energy under grant DE-FG02-87ER40317. The authors acknowledge *REU Site: Data + Computing = Discovery* NSF Award 1950052 that supported some of the research with FLASH.

References

[1] Stony Brook University, "Ookami." https://www.stonybrook.edu/commcms/ookami/, 2020.

[2] J. T. Palmer, S. M. Gallo, T. R. Furlani, M. D. Jones, R. L. DeLeon, J. P. White, N. Simakov, A. K. Patra, J. Sperhac, T. Yearke, R. Rathsam, M. Innus, C. D. Cornelius, J. C. Browne, W. L. Barth, and R. T. Evans, "Open xdmod: A tool for the comprehensive management of high-performance computing resources," *Computing in Science Engineering*, vol. 17, no. 4, pp. 52–62, 2015.

[3] University at Buffalo Center for Computational Research. https://ookami.ccr.xdmod.org/index.php/, 2021.

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

[5] S. S. Shende and A. D. Malony, "The tau parallel performance system," *The International Journal of High Performance Computing Applications*, vol. 20, no. 2, pp. 287–311, 2006.

[6] Appentra Solutions, S.L. https://www.appentra.com/products/parallelware-analyzer/, 2021.

[7] Thomas Gruber, Jan Eitzinger, Georg Hager, and Gerhard Wellein, ""likwid"." https://zenodo.org/record/4983493#.YS1_WIhKhPZ, 2021.

[8] J. Laukemann, J. Hammer, J. Hofmann, G. Hager, and G. Wellein, "Automated instruction stream throughput prediction for intel and amd microarchitectures," in *2018 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)*, pp. 121–131, 2018.

[9] J. Laukemann, J. Hammer, G. Hager, and G. Wellein, "Automatic throughput and critical path analysis of x86 and arm assembly kernels," in *2019 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)*, pp. 1–6, 2019.

[10] B. Fryxell, K. Olson, P. Ricker, F. X. Timmes, M. Zingale, D. Q. Lamb, P. MacNeice, R. Rosner, J. W. Truran, and H. Tufo, "FLASH: An adaptive mesh hydrodynamics code for modeling astrophysical thermonuclear flashes," *The Astrophysical Journal Supplement Series*, vol. 131, pp. 273–334, 2000.

[11] A. C. Calder, B. Fryxell, T. Plewa, R. Rosner, L. J. Dursi, V. G. Weirs, T. Dupont, H. F. Robey, J. O. Kane, B. A. Remington, R. P. Drake, G. Dimonte, M. Zingale, F. X. Timmes, K. Olson, P. Ricker, P. MacNeice, and H. M. Tufo, "On validating an astrophysical simulation code," *The Astrophysical Journal Supplement Series*, vol. 143, pp. 201–229, 2002.

[12] A. Dubey, K. Antypas, A. Calder, C. Daley, B. Fryxell, J. Gallagher, D. Lamb, D. Lee, K. Olson, L. Reid, P. Rich, P. Ricker, K. Riley, R. Rosner, A. Siegel, N. Taylor, K. Weide, F. Timmes, N. Vladimirova, and J. Zuhone, "Evolution of flash, a multi-physics scientific simulation code for high-performance computing," *International Journal of High Performance Computing Applications*, vol. 28, no. 2, pp. 225–237, 2014.

[13] A. Dubey, A. Calder, R. Fisher, C. Graziani, G. Jordan, D. Lamb, L. Reid, D. Townsley, and K. Weide, "Pragmatic optimizations for better scientific utilization of large supercomputers.," *International Journal of High Performance Computing Applications*, vol. 27, no. 3, pp. 360–373, 2013.

[14] A. Dubey, K. Antypas, A. Calder, B. Fryxell, D. Lamb, P. Ricker, L. Reid, K. Riley, R. Rosner, A. Siegel, F. Timmes, N. Vladimirova, and K. Weide, "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)*, pp. 1–8, 2013.

[15] M. Zingale, F. X. Timmes, B. Fryxell, D. Q. Lamb, K. Olson, A. C. Calder, L. J. Dursi, P. Ricker, R. Rosner, P. MacNeice, and H. M. Tufo, "Helium detonations on neutron stars," *Astrophysical Journal Supplement Series*, vol. 133, p. 195, 2001.

[16] L. J. Dursi, M. Zingale, A. C. Calder, B. Fryxell, F. X. Timmes, N. Vladimirova, R. Rosner, A. Caceres, D. Q. Lamb, K. Olson, P. M. Ricker, K. Riley, A. Siegel, and J. W. Truran, "The Response of Model and Astrophysical Thermonuclear Flames to Curvature and Stretch," *Astrophysical Journal*, vol. 595, pp. 955–979, Oct. 2003.

[17] A. Alexakis, A. C. Calder, L. J. Dursi, R. Rosner, J. W. Truran, B. Fryxell, M. Zingale, F. X. Timmes, K. Olson, and P. Ricker, "On the nonlinear evolution of wind-driven gravity waves," *Physics of Fluids*, vol. 16, pp. 3256–3268, Sept. 2004.

[18] A. Alexakis, A. C. Calder, A. Heger, E. F. Brown, L. J. Dursi, J. W. Truran, R. Rosner, D. Q. Lamb, F. X. Timmes, B. Fryxell, M. Zingale, P. M. Ricker, and K. Olson, "On heavy element enrichment in classical novae," *Astrophysical Journal*, vol. 602, pp. 931–937, 2004.

[19] R. Fisher, S. Abarzhi, S. M. Antypas, K.and Asida, A. C. C alder, F. Cattaneo, P. Constantin, A. Dubey, I. Foster, J. B. Gallagher, M. K. Ganapath, C. C. Glendenin, L. Kadanoff, D. Q. Lam b, S. Needham, M. Papka, T. Plewa, L. Reid, P. Rich, K. Riley, and D. Sheeler, "Tera-scale Turbulence Computation on BG/L Using the FLASH3 Code," *IBM J. Res. & Dev.*, vol. 52, pp. 127–136, jan 2008.

[20] P. Tzeferacos, M. Fatenejad, N. Flocke, C. Graziani, G. Gregori, D. Q. Lamb, D. Lee, J. Meinecke, A. Scopatz, and K. Weide, "FLASH MHD simulations of experiments that study shock-generated magnetic fields," *High Energy Density Physics*, vol. 17, pp. 24–31, Dec. 2015.

[21] J. O'Neal, K. Weide, and A. Dubey, "Experience report: Refactoring the mesh interface in flash, a multiphysics software," in *WSSSPE6.1, colocated with eScience 2018, Amsterdam, Netherlands*, 2018.

[22] C. Daley, J. Bachan, S. Couch, A. Dubey, M. Fatenejad, B. Gallagher, D. Lee, and K. Weide, "Adding shared memory parallelism to FLASH for many-core architectures," in *TACC-Intel Highly Parallel Computing Symposium*, April 2012. Poster.

[23] A. Dubey, "Programming abstractions for orchestration of hpc scientific computing." https://chapel-lang.org/CHIUW2019.html, 2019. Keynote, Chapel User's Group Meeting.

[24] A. Dubey, "Dynamic resource management, an application perspective." https://project.inria.fr/resourcearbitration/program/, 2019. Invited talk, RADR, co-located with IPDPS.

[25] U. O. of Defense Programs, "Accelerated strategic computing initiative (asci) program plan [fy2000]," 1 2000.

[26] R. Rosner, A. C. Calder, L. J. Dursi, B. Fryxell, D. Q. Lamb, J. C. Niemeyer, K. Olson, P. Ricker, F. X. Timmes, J. W. Truran, H. Tufo, Y. Young, M. Zingale, E. Lusk, and R. Stevens, "Flash Code: Studying Astrophysical Thermonuclear Flashes," *Computing in Science and Engineering*, vol. 2, p. 33, Mar. 2000.

[27] A. C. Calder, B. C. Curtis, L. J. Dursi, B. Fryxell, G. Henry, P. MacNeice, K. Olson, P. Ricker, R. Rosner, F. X. Timmes, H. M. Tufo, J. W. Truran, and M. Zingale, "High-performance reactive fluid flow simulations using adaptive mesh refinement on thousands of processors," in *Proceedings of Supercomputing 2000*, p. http://sc2000.org, 2000.

[28] K. Antypas, A. C. Calder, A. Dubey, R. T. Fisher, M. K. Ganapathy, B. Gallagher, L. B. Reid, K. Riley, D. J. Sheeler, and N. T. Taylor, "Scientific applications on the massively parallel BG/L machine," in *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications & Conference on Real-Time Computing Systems and Applications, PDPTA 2006, Las Vegas, Nevada, USA, June 26-29, 2006, Volume 1* (H. R. Arabnia, ed.), pp. 292–298, CSREA Press, 2006.

[29] B. Michalowicz, E. Raut, Y. Kang, T. Curtis, B. Chapman, and D. Oryspayev, "Comparing the behavior of openmp implementations with various applications on two different fujitsu a64fx platforms," in *Practice and Experience in Advanced Research Computing*, PEARC '21, (New York, NY, USA), Association for Computing Machinery, 2021.