Multifaceted Approaches for Introducing a Hardware-thread Migratory Architecture

Aaron Jezghani
ajezghani3@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia

Jeffrey Young
jyoung9@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia

Vedavyas Mallela
vmallela6@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia

Will Powell
will.powell@cc.gatech.edu
Georgia Institute of Technology
Atlanta, Georgia

ABSTRACT

The challenges of HPC education span a wide array of targeted applications, ranging from developing a new generation of administrators and facilitators to maintain and support cluster resources and their respective user communities, to broadening the impact of HPC workflows by reaching non-traditional disciplines and training researchers in the best-practice tools and approaches when using such systems. Furthermore, standard x86 and GPU architectures are becoming untenable to scale to the needs of computational research, necessitating software and hardware co-development on less-familiar processors. While platforms such as Cerebras and SambaNova have matured to include common frameworks such as TensorFlow and PyTorch as well as robust APIs, and thus are amenable to production use cases and instructional material, other systems may lack such infrastructure maturity, impeding all but the most technically inclined developers from being able to leverage the system.

We present here our efforts and outcomes of providing a co-development and instructional platform for the Lucata Pathfinder thread-migratory system in the Rogues Gallery at Georgia Tech. Through a collection of user workflow management, co-development with the platform's engineers, community tutorials, undergraduate coursework, and student hires, we have been able to explore multiple facets of HPC education in a unique way that can serve as a viable template for others seeking to develop similar efforts.

KEYWORDS

HPC education, novel architecture workflows, CS curriculum, workforce development, community education

1 INTRODUCTION

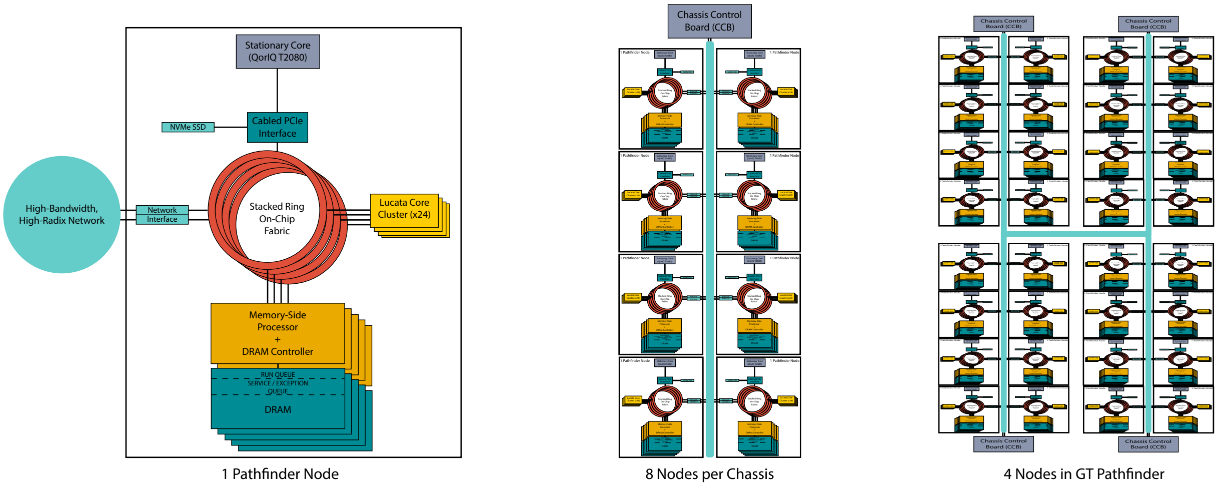
Despite overlaps with traditional computing, HPC education and training requires specialized skills, especially in terms of code scaling and target hardware dispatch. Target audiences for this type of training can include students [2, 4, 16], researchers [6, 21], and future administrators/facilitators [1, 22]. Approaches for training range from using traditional classroom instruction to condensed workshops or tutorials to unplugged activities to overcome the challenges of HPC training across the array of demographics and science domains, with a number of focus groups running annual workshops to aggregate best practices and pave the path forward; for example, see [10–12, 24].

Further complicating the state of HPC education is the ever-changing landscape of cluster infrastructure. Notably, the broad recognition of scalability challenges in the movement of data for modern computational research has led to the introduction of a new standard for a high-bandwidth, disaggregated architecture with shared memory access, Compute Express Link, or CXL [7]. Although systems capable of supporting the first generation of the CXL standard are just now deploying, major vendors have changed product offerings [8], and it is largely expected that broad adoption will follow the adoption of the 2.0 standard by 2025 [15]. Given the major shift in infrastructure, the need for a well-developed user community and management workforce is readily apparent.

Figure 1: An internal view of the Lucata Pathfinder chassis.

The Rogues Gallery is an NSF-funded novel architecture testbed designed to provide early experiences for a variety of resources [28], with training being one of its core tenets. Approaches based on near-memory computation (of which CXL is a subset), reconfigurable

https://doi.org/10.22369/issn.2153-4136/14/2/6


Stationary Core (QorIQ T2080)
Cabled PCIe Interface
NVMe SSD
High-Bandwidth, High-Radix Network
Network Interface
Stacked Ring On-Chip Fabric
Lucata Core Cluster (x24)
Memory-Side Processor + DRAM Controller
RUN QUEUE
SERVICE / EXCEPTION QUEUE
DRAM
Chassis Control Board (CCB)
1 Pathfinder Node
Chassis Control Board (CCB)

1 Pathfinder Node

8 Nodes per Chassis

4 Nodes in GT Pathfinder

Figure 2: The architecture of the Pathfinder. Each node consists of 24 Lucata cores, managed by a stationary core, with access to shared memory and a reconfigurable network. Each chassis contains 8 nodes, with 4 chassis total in the GT Pathfinder system.

hardware, advanced networking, and neuromorphic processing are hosted, allowing students and researchers to consider how these architectures can be leveraged, either using current software capabilities or through novel algorithm design, to tackle many of the problems in HPC. In this paper, we will discuess one particular platform, the Lucata Pathfinder, and how we leverage it to educate the community, students, and future administrators, as covered in Sections 4, 5, and 6, respectively.

2 THE LUCATA PATHFINDER SYSTEM

The Pathfinder advances the capabilities of the prototype Emu Chick architecture [9], an earlier prototype of migratory thread hardware.The Pathfinder-S is based on the concept of migrating threads rather than a deep cache hierarchy to improve locality and efficiency. The GT Pathfinder system is comprised of 4 chassis, shown in Figure 4, each of which contains 8 nodes to run compute. The primary compute elements, dubbed Lucata cores, execute these migratory threads and forego a traditional memory controller in favor of eight memory side processors (MSPs) that assist in accelerating specific in-memory operations like remote writes, addition, and atomics. There are 24 Lucata cores per node, and thus 192 Lucata cores per chassis and 768 for the GT Pathfinder system.

Figure 2 provides a high-level overview of the Pathfinder architecture as a single node, chassis, and multi-chassis view. The Lucata cores are managed by "stationary cores", which use a QorIQ T2080 CPU [17], with 4 multi-threaded PowerPC e6500 cores, and 8 GB of memory; the 8 boards in each chassis are managed by a chassis control board, which uses the same processor model. The nodes and chassis are connected via a reconfigurable, high-bandwidth, high-radix network for efficient scaling in problems such as sparse matrix operations and graph analysis algorithms.

The chassis control boards and stationary cores run Yocto Linux [18], with a custom API for Lucata core execution and thread management across the system. The custom kernel provides a subset of the standard utilities found for more common platforms, such as x86 with RHEL or Ubuntu OSes, but this has yet to prove to be a barrier in delivery of an operational system.

In addition to the Pathfinder, RG also hosts x86 machines to develop, validate, and compare code against dispatch to the target hardware. These servers range from a Intel Westmere Jupyter front-end with 64 cores and 1 TB of memory for lightweight simulation by many concurrent users to a set of four Intel Ice Lake nodes with 64 cores and 512 GB of memory for benchmark performance comparison. Other available architectures such as GPUs, FPGAs, and Arm CPUs can be used for such efforts, but have not yet been explored for these purposes.

3 SCHEDULER AND SOFTWARE INFRASTRUCTURE

Pathfinder access is handled via the Slurm cluster management and job scheduling system [25]. We chose to implement the Pathfinder as a federated cluster in the Rogues Gallery for several reasons, including the dynamic private network for the compute nodes, unsupported plugins leveraged on other hardware, and highly variable configurations thoughout the cluster. The federation allows for the Pathfinder Slurm instance to utilize a unique configuration while readily accepting jobs from users on the Rogues Gallery login nodes for a smoother integration. Additional system SSH configurations allow users to directly login to the compute nodes using the global system names and going through the gateway server.

In order to program code for execution on the Pathfinder, the Cilk parallel-programming model is utilized with the Lucata API [23]. The successor to Intel's cilk/cilk++ frameworks, OpenCilk provides a simple means for users to write deterministic parallel code as an extension to the LLVM compiler. Users are encouraged to run Jupyter notebooks on Hawksbill, from which they can explore the Lucata API and Cilk programming framework, followed by launching code for scaled execution on systems like the Frozones or the actual Pathfinder system.

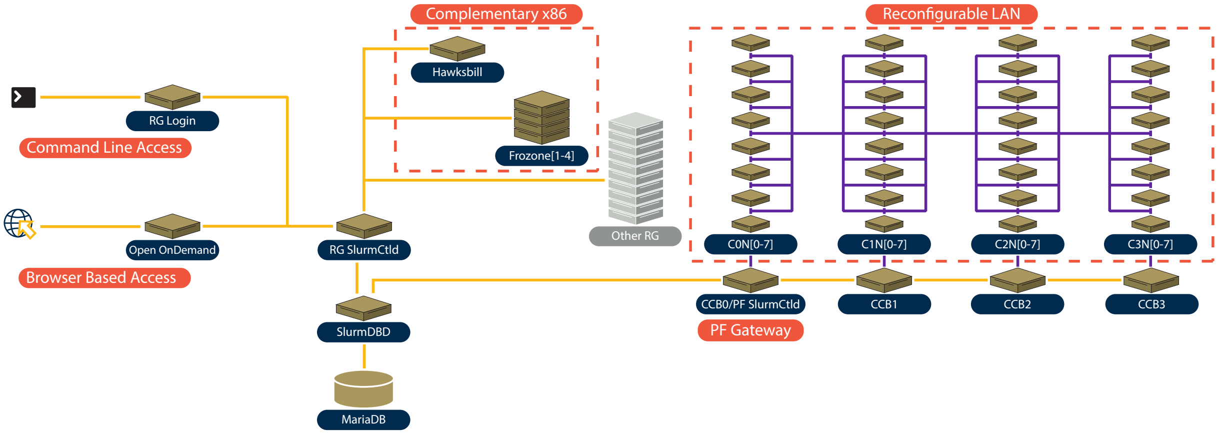


Figure 3: The Rogues Gallery is comprised of federated clusters, with a primary Slurm controller serving the majority of the hardware, including x86 servers that provide compute capabilities complementary to the Pathfinder system, and a secondary Slurm controller atop the Pathfinder and its reconfigurable LAN. Additional system-wide SSH configurations allow users to submit to either controller from common cluster access points.

4 COMMUNITY TUTORIAL EXPERIENCES

Tutorials focusing on the GT Emu Chick and Pathfinder systems have been presented at several conferences, either as components of a broader RG tutorial for the ASPLOS19 and PEARC19 conferences [19, 20], or as a dedicated presentation for the PEARC21 and HPEC22 conferences [26, 27] in an online-only format. Additionally, proposals were submitted to present tutorials at PEARC22 and SC22 but were not selected for the final program.

In all cases, the target audience was users who had some familiarity with high-performance or parallel computing but who might not be experts in computing with a system like the Lucata platform. As shown in Figure 5, the most recent HPEC22 tutorial debuted the usage of Jupyter notebooks as a mechanism for tutorial attendees to learn the basics of the Cilk-based Pathfinder workflow and to compile and run code for the Pathfinder simulator and the hardware via integrated Slurm commands. This new interface dramatically increased the number of attendees who interacted with the tutorial code samples to approximately 50% from earlier tutorials where just a few attendees downloaded and ran code samples on their laptops. This interface was created through an iterative process, which took several tutorial offerings to fully create and deploy from initial C-based code samples and slides that were used at the first tutorial back in 2019.

In general, each of these tutorials had anywhere from 10 to 25 attendees, with some outliers like PEARC22, which limited attendees to pre-registered sessions, which seemed to artifically limit the number of attendees who might join for part of the tutorial. Surveys indicated that most attendees were already working in some computing-related fields, and the more dedicated online and half-day tutorials like HPEC22 provided a more focused group of attendees.

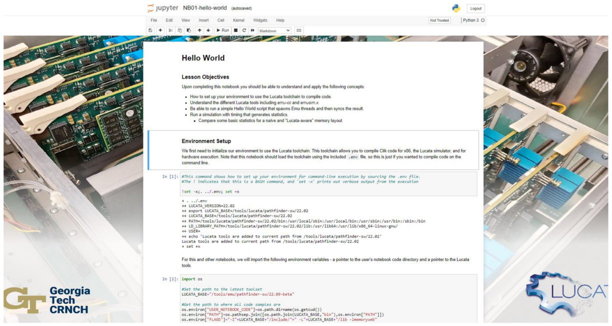
The key lessons learned from these tutorials were as follows: 1) The user interface and required background knowledge can dramatically increase/decrease engagement by attendees. 2) Tutorial venues with limited numbers of cross-scheduled tutorials and an option for a hybrid attendance mode seemed to increase audience attendance and possibly participation. 3) Most attendees had some familiarity with HPC or parallel computing concepts but required background knowledge to fully understand the tutorial's material.

Figure 4: An earlier Rogues Gallery tutorial at Georgia Tech.

5 FUTURE COMPUTING VIP'S NEAR-MEMORY SUBTEAM

The Future Computing with the Rogues Gallery VIP course introduces students to the various architectures in the Rogues Gallery and their target applications through a vertically-integrated, project-based approach [14]. In particular, the near-memory subteam is tasked with exploring the Pathfinder architecture and its use in HPC through a series of targeted benchmarks, including

- HPCG for sparse-matrix operations [13],
- pChase for exploring memory access latency [5], and
- breadth-first search for use in graph applications [3].


jupyter NB01-hello-world (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown
Georgia Tech CRNCH
LUCATA

Hello World

Lesson Objectives

Upon completing this notebook you should be able to understand and apply the following concepts:

- How to set up your environment to use the Lucata toolchain to compile code.
- Understand the different Lucata tools including *emu-cc* and *emusim.x*.
- Be able to run a simple Hello World script that spawns Emu threads and then syncs the result.
- Run a simulation with timing that generates statistics.
  - Compare some basic statistics for a naive and "Lucata-aware" memory layout.

Environment Setup

We first need to initialize our environment to use the Lucata toolchain. This toolchain allows you to compile Cilk code for x86, the Lucata simulator, and for hardware execution. Note that this notebook should load the toolchain using the included `.env` file, so this is just if you wanted to compile code on the command line.

```
In [1]: #This command shows how to set up your environment for command-line execution by sourcing the .env file.
        #The ! indicates that this is a BASH command, and `set -x` prints out verbose output from the execution

        !set -x;. ../.env; set +x
```

```
+ . ../.env
++ LUCATA_VERSION=22.02
++ export LUCATA_BASE=/tools/lucata/pathfinder-sw/22.02
++ LUCATA_BASE=/tools/lucata/pathfinder-sw/22.02
++ PATH=/tools/lucata/pathfinder-sw/22.02/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
++ LD_LIBRARY_PATH=/tools/lucata/pathfinder-sw/22.02/lib:/usr/lib64:/usr/lib/x86_64-linux-gnu/
++ USER=
++ echo 'Lucata tools are added to current path from /tools/lucata/pathfinder-sw/22.02'
Lucata tools are added to current path from /tools/lucata/pathfinder-sw/22.02
+ set +x
```

For this and other notebooks, we will import the following environment variables - a pointer to the user's notebook code directory and a pointer to the Lucata tools.

```
In [2]: import os

        #Set the path to the latest toolset
        LUCATA_BASE="/tools/emu/pathfinder-sw/22.09-beta"

        #Get the path to where all code samples are
        os.environ["USER_NOTEBOOK_CODE"]=os.path.dirname(os.getcwd())
        os.environ["PATH"]=os.pathsep.join([os.path.join(LUCATA_BASE,"bin"),os.environ["PATH"]])
        os.environ["FLAGS"]="-I"+LUCATA_BASE+"/include/"+" -L"+LUCATA_BASE+"/lib -lmemoryweb"
```

Figure 5: Screenshot from one of the tutorial demo Jupyter notebooks. To help reinforce the focus and attention of attendees, customized Jupyter styling was developed for the hands-on components.

The students in the subteam focus on one of the given applications in smaller project subteams, using the general class meeting each week to address common issues with Slurm, Cilk, or the Lucata API.

To help onboard new team members, the students have employed the OpenCilk documentation to understand the parallel-programming framework, and notebooks from the RG/Pathfinder tutorials to see practical uses for executing code on the system. Additionally, they develop a rich knowledgebase through their weekly notebook entries, which collectively provide a robust documentation of technical implementation details, references, and experiences that enable new students to contribute quickly to the team. As an example, student success in comparing serial code against Cilk-programmed multiplication of square matrices can be seen in Figure 6. As the students moved from toy models to benchmarking x86 and the Pathfinder systems, they engaged the Lucata engineers for code and algorithm development, and turned to the company repositories for components like

As an added benefit to the nature of the VIP program, students are able to use the opportunity to develop projects that can apply to other courses in their computer science curriculum. In particular, two students enrolled in CS3210: Design of Operating Systems chose the Pathfinder system to explore kernel functionality by attempting to write their own DMA driver to provide better performance via lower latency memory access for the Lucata cores, as depicted in Figure 7. As with the benchmark efforts, the students worked heavily alongside Lucata developers to understand the workflow to build the kernel from source. Although the semester ended with an incomplete kernel, as the networking components built incorrectly, the students gained considerable knowledge with regards to low-level activities to advance HPC capabilities on novel architectures.

Perhaps one of the biggest unexpected challenges encountered by students from this subteam was contention with Lucata development cycles. A combination of scheduling conflicts, in which the engineers had reserved time on the system and rendered it unavailable for student use, and a lack of understanding of the local network configuration, which caused problems if student code was built for a different setup, created disruptions throughout the semesters. Fortunately, both the Lucata personnel and students were patient, and both groups graciously considered the opportunity as a learning experience from which workflows and system tools could be improved.

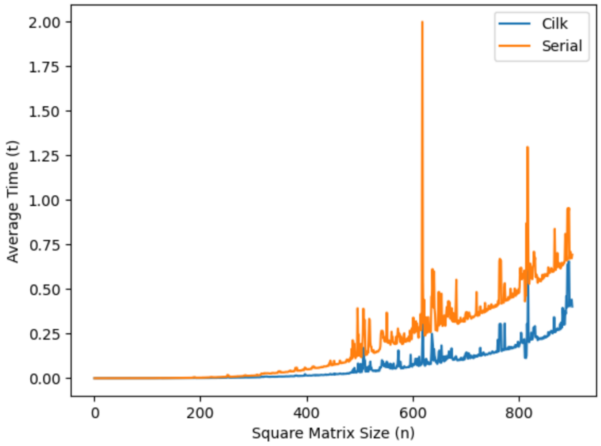


Figure 6: Benchmark comparing serial and Cilk-parallelized algorithms to perform square matrix multiplication. As with other parallel computing frameworks, the performance improvement requires sufficient problem size.

6 POTENTIAL PIPELINE FOR NEXT-GENERATION ADMINISTRATION AND FACILITATION

A recent initiative within the Rogues Gallery has been the hiring of a student worker to develop administrative utilities such as a production solution to publish the Pathfinder system configuration and facilitate access among the Lucata developers and research users. Unlike student research employees, whose work may utilize only familiar academic tools, or student administrative employees, whose work may only include a very targeted activity within the broader infrastructure, administrative employment on a near-production system such as the Pathfinder provides broad exposure to both the research workflows and the standard tools required to manage the systems. In this way,

The first project addressed by the student employee focuses on clarifying the complexity of the Lucata Pathfinder scheduling and reconfiguration mechanisms. Notably, the Pathfinder 4 chassis system can be reconfigured in up to eight valid configurations that include combinations of single-node, multi-node but single-chassiss, and multi-chassis.

To work around this complexity, the initial scheduling mechanism for the Pathfinder used a Google Calendar to block off time with specific configuration details. In 2022, Slurm was built for the Pathfinder system and was set up as a federated instance with the rest of the Rogues Gallery testbed. However, it was noted that the calendar interface seemed somewhat more intuitive to some of the engineers and developers working with the system.

Figure 8 shows a new approach that attempts to combine the best of both worlds by using Slurm to control and reconfigure

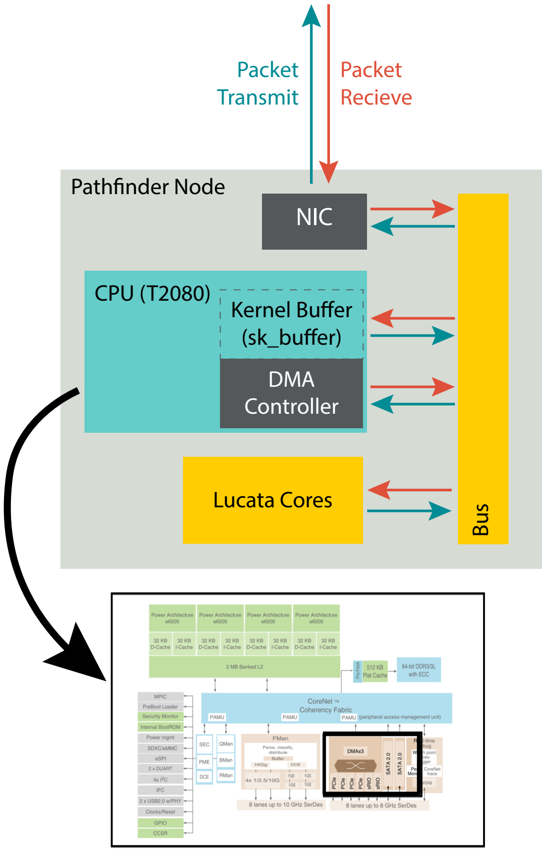


Figure 7: Schematic showing how a DMA driver would improve data throughput for the Lucata cores by bypassing processing in the stationary cores entirely. Despite the DMA controller being present on the T2080 CPU, the kernel currently provides no explicit support to move data directly between Lucata cores and DRAM.

the system and Google Calendar and Google Sheets to help share and update the state of the system. The Google Calendar API is used to read the state of the Pathfinder system as configured via Slurm and report it to a calendar that can be checked by Lucata engineers, researchers, and students. Likewise, Figure 9 shows the current status of the Pathfinder system with this example showing two chassis as combined "multi-chassis" instances in blue, a single-chassis instance in green, and single nodes in grey.

One potential enhancement to this setup would be to actually interpret and allow inputs from the Google Calendar to drive Slurm jobs for the Pathfinder or even eventually for other less complicated platforms. A key note here is that while the Lucata system might seem like a unique and one-off type system with a very bespoke scheduler, we are starting to see similar setups in systems with NVIDIA Multi-Instance GPUs (MIGs) which currently are not compatible with current Slurm installations, due to their reconfigurability. It is likely that other cutting edge systems could benefit from thinking about user interfaces in similar fashions to improve both utilization and user experiences.

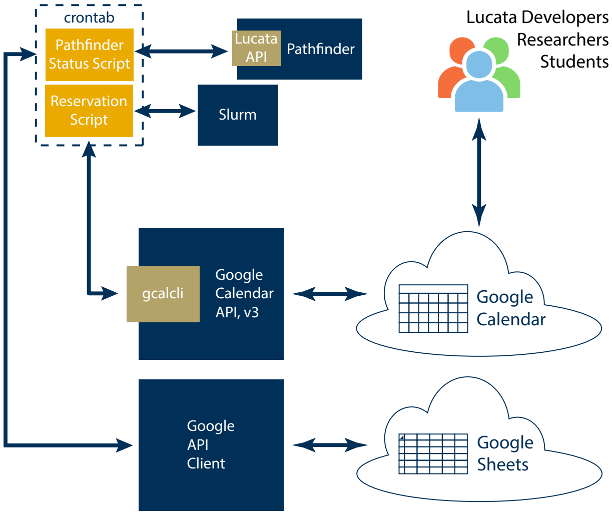
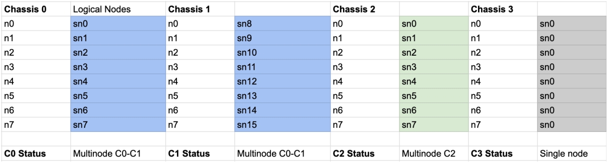


Figure 8: Automated status update.



Chassis 0	Logical Nodes	Chassis 1		Chassis 2		Chassis 3	
n0	sn0	n0	sn8	n0	sn0	n0	sn0
n1	sn1	n1	sn9	n1	sn1	n1	sn0
n2	sn2	n2	sn10	n2	sn2	n2	sn0
n3	sn3	n3	sn11	n3	sn3	n3	sn0
n4	sn4	n4	sn12	n4	sn4	n4	sn0
n5	sn5	n5	sn13	n5	sn5	n5	sn0
n6	sn6	n6	sn14	n6	sn6	n6	sn0
n7	sn7	n7	sn15	n7	sn7	n7	sn0
C0 Status	Multinode C0-C1	**C1 Status**	Multinode C0-C1	**C2 Status**	Multinode C2	**C3 Status**	Single node

Figure 9: Cluster Configuration Example with Google Sheet.

Although we recognize that asking student employees to "fix" interface issues like this for a unique system has some shortcomings, we feel that near-production system administration fills a specific need in the training process. Broader-reach workforce development initiatives churn out larger numbers of trained individuals, but with less depth of knowledge due to the nature of the training exercises and limited time for the training. At the other end of the spectrum, student researchers managing isolated systems may become experts in the systems they manage, but are less likely to use industry-standard tools in accordance with best practices. By working on a near-production system, students are exposed to real-world problems and the tools used to monitor, mitigate, and manage them, providing a nice balance between richness and exposure. Furthermore, as multiple near-production systems may exist on a given campus, such a workforce model can broaden its impact.

7 CONCLUSIONS

The deployment of extremely novel and unique near-memory migratory thread systems like the Lucata Chick and Pathfinder has provided both new opportunities for training as well as challenges from an educational and maintenance standpoint.

From an educational perspective, the Lucata systems have provided an extremely compelling high-performance platform that has been used to support local coursework at Georgia Tech, tutorial development at architecture and HPC conferences, and proving grounds for testing of new tools and techniques to support next-generation HPC ecosystems. Figuring out the best way to engage students and non-experts has required iterative and collaborative work with GT researchers as well as strong support from the vendor.

Some takeaways that could improve this process would be an earlier focus on smarter tooling and interfaces to support new users including Slurm scheduling instead of ad-hoc scheduling, automated scripts to reconfigure the system, and Jupyter notebooks for teaching and tutorials. Furthermore, matching tutorial materials and scope with the correct conference audience has proven to be important to get good turn out and engagement with such a unique platform.

Overall, the engagement with the community and collaboration with local researchers and the vendor has provided Georgia Tech with an incomparable resource and experience for teaching and researching high-performance computing applications. Future efforts will focus on extending the scope of applications that can be run using tutorial-style notebooks and in supporting added tools and features to improve the user experience when using the Pathfinder system as part of the testbed.

ACKNOWLEDGMENTS

This research was supported by the NSF MRI award #1828187: “MRI: Acquisition of an HPC System for Data-Driven Discovery in Computational Astrophysics, Biology, Chemistry, and Materials Science." Additionally, this research was supported in part through research infrastructure and services provided by the Rogues Gallery testbed hosted by the Center for Research into Novel Computing Hierarchies (CRNCH) at Georgia Tech. The Rogues Gallery testbed is primarily supported by the National Science Foundation (NSF) under NSF Award Number #2016701. Any opinions, findings and conclusions, or recommendations expressed in this material are those of the author(s), and do not necessarily reflect those of the NSF.

REFERENCES

[1] David Akin, Mehmet Belgin, Timothy A. Bouvet, Neil C. Bright, Stephen Harrell, Brian Haymore, Michael Jennings, Rich Knepper, Daniel LaPine, Fang Cherry Liu, Amiya Maji, Henry Neeman, Resa Reynolds, Andrew H. Sherman, Michael Showerman, Jenett Tillotson, John Towns, George Turner, and Brett Zimmerman. 2017. Linux Clusters Institute Workshops: Building the HPC and Research Computing Systems Professionals Workforce. In *Proceedings of the HPC Systems Professionals Workshop (HPCSYSPROS'17)*. Association for Computing Machinery, New York, NY, USA, Article 4, 8 pages. https://doi.org/10.1145/3155105.3155108

[2] A. Antonov, N. Popova, and Vl. Voevodin. 2018. Computational science and HPC education for graduate students: Paving the way to exascale. *J. Parallel and Distrib. Comput.* 118 (2018), 157–165. https://doi.org/10.1016/j.jpdc.2018.02.023

[3] David A. Bader, John Feo, John Gilbert, Jeremy Kepner, David Koester, Eugene Loh, Kamesh Madduri, Bill Mann, and Theresa Meuse. 2007. HPCS Scalable Synthetic Compact Applications #2 Graph Analysis (SSCA#2 v2.2 Specification). online. http://www.graphanalysis.org/benchmark/HPCS-SSCA2_Graph-Theory_v2.2.pdf

[4] Fabio Banchelli and Filippo Mantovani. 2018. Filling the gap between education and industry: evidence-based methods for introducing undergraduate students to HPC. In *2018 IEEE/ACM Workshop on Education for High-Performance Computing (EduHPC)*. 41–50. https://doi.org/10.1109/EduHPC.2018.00008

[5] Tim Besard and Steven Noonan. 2013. pChase. github. https://github.com/maleadt/pChase

[6] Caughlin Bohn. 2022. HPC Outreach and Education at Nebraska. In *Practice and Experience in Advanced Research Computing (PEARC '22)*. Association for Computing Machinery, New York, NY, USA, Article 63, 4 pages. https://doi.org/10.1145/3491418.3535128

[7] The Compute Express Link Consortium. 2022. About CXL. https://www.computeexpresslink.org/about-cxl

[8] Intel Corporation. 2022. Migration from Direct-Attached Intel Optane Persistent Memory to CXL-Attached Memory. Technical Brief. https://www.intel.com/content/dam/www/central-libraries/us/en/documents/2022-11/optane-pmem-to-cxl-tech-brief.pdf

[9] Timothy Dysart, Peter Kogge, Martin Deneroff, Eric Bovell, Preston Briggs, Jay Brockman, Kenneth Jacobsen, Yujen Juan, Shannon Kuntz, Richard Lethin, Janice McMahon, Chandra Pawar, Martin Perrigo, Sarah Rucker, John Ruttenberg, Max Ruttenberg, and Steve Stein. 2016. Highly Scalable Near Memory Processing with Migrating Threads on the Emu System Architecture. In *2016 6th Workshop on Irregular Applications: Architecture and Algorithms (IA3)*. 2–9. https://doi.org/10.1109/IA3.2016.007

[10] SIGHPC Education. 2023. Sixth Workshop on Strategies for Enhancing HPC Education and Training (SEHET23). https://sighpceducation.acm.org/events/sehet23_cfp/

[11] EduHPC. 2022. EduHPC-22: Workshop on Education for High-Performance Computing. https://tcpp.cs.gsu.edu/curriculum/?q=eduhpc22

[12] EduPAR. 2023. EduPar-23: 13th NSF/TCPP Workshop on Parallel and Distributed Computing Education. https://tcpp.cs.gsu.edu/curriculum/?q=edupar23

[13] Michael Allen Heroux and Jack. Dongarra. 2013. Toward a new metric for ranking high performance computing systems. (6 2013). https://doi.org/10.2172/1089988

[14] Aaron Jezghani, Jeffrey Young, Will Powell, Eric J. Coulter, and Ronald Rahaman. 2023. Future Computing with the Rogues Gallery. [online] EduPar23 preprint. https://tcpp.cs.gsu.edu/curriculum/?q=system/files/Jezghani%20paper.pdf

[15] Astera Labs. 2022. Compute Express Link CXL 2.0 Specification Released the Big One. News Brief. https://www.asteralabs.com/news/compute-express-link-cxl-2-0-specification-released-the-big-one/

[16] Julia Mullen, Lauren Milechin, and Dennis Milechin. 2021. Teaching and learning HPC through serious games. *J. Parallel and Distrib. Comput.* 158 (2021), 115–125. https://doi.org/10.1016/j.jpdc.2021.07.014

[17] NXP. 2023. QorIQ T2080 and T2081 Multicore Communications Processors. [online] Product Page. https://www.nxp.com/products/processors-and-microcontrollers/power-architecture/qoriq-communication-processors/t-series/qoriq-t2080-and-t2081-multicore-communications-processors:T2080

[18] The Yocto Project. 2023. Yocto Project (webpage). https://www.yoctoproject.org/

[19] Jason Riedy, Will Powell, Jeffrey Young, Tom Conte, and Vivek Sarkar. 2019. Rogues Gallery: Addressing Post-Moore Computing. https://crnch-rg.gitlab.io/pearc-2019/

[20] Jason Riedy, Jeffrey Young, Tom Conte, and Vivek Sarkar. 2019. Programming Novel Architectures in the Post-Moore Era with the Rogues Gallery. https://crnch-rg.gitlab.io/asplos-2019/

[21] Semir Sarajlic, Neranjan Edirisinghe, Yuriy Lukinov, Michael Walters, Brock Davis, and Gregori Faroux. 2016. Orion: Discovery Environment for HPC Research and Bridging XSEDE Resources. In *Proceedings of the XSEDE16 Conference on Diversity, Big Data, and Science at Scale (XSEDE16)*. Association for Computing Machinery, New York, NY, USA, Article 54, 5 pages. https://doi.org/10.1145/2949550.2952770

[22] Semir Sarajlic, Neranjan Edirisinghe, Yubao Wu, Yi Jiang, and Gregori Faroux. 2017. Training-Based Workforce Development in Advanced Computing for Research and Education (ACoRE). In *Proceedings of the Practice and Experience in Advanced Research Computing 2017 on Sustainability, Success and Impact (PEARC17)*. Association for Computing Machinery, New York, NY, USA, Article 71, 4 pages. https://doi.org/10.1145/3093338.3104178

[23] Tao B. Schardl and I-Ting Angelina Lee. 2023. OpenCilk: A Modular and Extensible Software Infrastructure for Fast Task-Parallel Code. In *Proceedings of the 28th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming (PPoPP '23)*. Association for Computing Machinery, New York, NY, USA, 189–203. https://doi.org/10.1145/3572848.3577509

[24] MIT Supercloud. 2022. Scaling HPC Education. https://supercloud.mit.edu/scaling-hpc-education

[25] Andy B Yoo, Morris A Jette, and Mark Grondona. 2003. Slurm: Simple linux utility for resource management. In *Job Scheduling Strategies for Parallel Processing: 9th International Workshop, JSSPP 2003, Seattle, WA, USA, June 24, 2003. Revised Paper 9*. Springer, 44–60.

[26] Jeffrey Young, Patrick Lavin, Jason Riedy, Srinivas Eswar, Janice McMahon, Aaron Jezghani, Will Powell, and Darryl Bailey. 2022. Exploring Graph Analysis for HPC with Near-Memory Accelerators. https://doi.org/10.5281/zenodo.7117251

[27] Jeffrey Young, Semir Sarajlic, Will Powell, Janice McMahon, and Jason Riedy. 2021. Lucata Pathfinder-S Tutorial: Next-generation Computationwith the Rogues Gallery. https://crnch-rg.gitlab.io/pearc-2019/

[28] Jeffrey S. Young, Jason Riedy, Thomas M. Conte, Vivek Sarkar, Prasanth Chatarasi, and Sriseshan Srikanth. 2019. Experimental Insights from the Rogues Gallery. In *2019 IEEE International Conference on Rebooting Computing (ICRC)*. 1–8. https://doi.org/10.1109/ICRC.2019.8914707