Wrangling Rogues: Managing Experimental Post-Moore Architectures

Will Powell, Jason Riedy
School of Computational Science and Engineering
Georgia Institute of Technology
Atlanta, Georgia 30322
{will.powell,jason.riedy}@cc.gatech.edu

Jeffrey S. Young, Tom Conte
School of Computer Science
Georgia Institute of Technology
Atlanta, Georgia 30322
{jyoung9,conte}@gatech.edu

Abstract—The Rogues Gallery is a new experimental testbed that is focused on tackling “rogue” architectures for the Post-Moore era of computing. While some of these devices have roots in the embedded and high-performance computing spaces, managing current and emerging technologies provides a challenge for system administration that are not always foresee in traditional data center environments.

We present an overview of the motivations and design of the initial Rogues Gallery testbed and cover some of the unique challenges that we have seen and foresee with upcoming hardware prototypes for future post-Moore research. Specifically, we cover the networking, identity management, scheduling of resources, and tools and sensor access aspects of the Rogues Gallery and techniques we have developed to manage these new platforms. We argue that current tools like the Slurm scheduler can support new rogues without major infrastructure changes.

Index Terms—testbed, post-Moore computing, identity management

I. CHALLENGES FOR POST-MOORE TESTBEDS

As we look to the end of easy and cost-effective transistor scaling, sometimes called the “post-Moore” era, we have reached a turning point in computer system design and usage. Accelerators like GPUs have created a pronounced shift in the high-performance computing and machine learning application spaces, but there is a wide variety of possible architectural choices for the post-Moore era, including memory-centric, neuromorphic, quantum, and reversible computing. These revolutionary research fields combined with alternative materials-based approaches to silicon-based hardware have given us a bewildering array of options and “rogue devices” for the coming post-Moore era but little guidance on how to evaluate potential hardware for tomorrow’s application needs.

One-off testbeds for each new Post-Moore technology increases the cost of evaluating them. We would like to provide a cohesive user environment that enables researchers to perform studies across different novel architectures that may be in different stages of acceptance by the wider computing community. For instance, devices based on FPGAs may be more “mainstream” in our current environment while super-cooled quantum computing devices are likely only available via national lab or industry testbeds.

The key challenges that this testbed needs to address are the following:

• Invest in rogues, but realize some technology may be short-lived. We cannot invest too much time in setting and integrating up a novel architecture when tools and software stacks may be in an unstable state of development. Rogues may be short-lived; they may not achieve their goals. Finding their limits is an important aspect of the Rogues Gallery.

• Physical hardware resources not dedicated to rogues should be kept to a minimum. As we explain in Section III, most functionality should be satisfied by VMs and containers rather than dedicating a physical server to each “rogue” piece of hardware.

• Collaboration and commiseration is key. Rogues need a community to succeed. If they cannot establish a community (users, vendors interested in providing updates, manageability from an IT perspective), they will disappear. We promote community with our testbed through a documentation wiki, mailing lists, and Slack channels for users as well as close contact with vendors, where appropriate.

• Licensing and appropriate identity management are tough but necessary challenges. Managing access to prototypes and licensed software are tricky for academic institutions, but we must work with collaborators like start-ups to provide a secure environment with appropriate separation of privileges to protect IP as necessary.

We aim to tackle these challenges with our testbed, the Rogues Gallery. Here we describe the initial design of the testbed as well as the high-level system management strategy that we use to maintain and tie together novel architectures. We use existing tools as much as possible; we need not re-invent the systems integration wheel to support early novel architectures. By their very nature, rogues are few in number. They are specialized but must not require much specialized administrative tooling.

A. The Rogues Gallery

The Rogues Gallery is a new concept to understand next-generation hardware with a focus on unorthodox and uncommon technologies. This project was initiated by Georgia Tech’s Center for Research into Novel Computing Hierarchies (CRNCH) in late 2017. The Gallery’s focus is to acquire new
and unique hardware (i.e. the aforementioned rogues) from vendors, research labs, and start-ups and make this hardware available to students, faculty, and industry collaborators within a managed data center environment. By exposing students and researchers to this set of unique hardware, we hope to foster cross-cutting discussions about hardware designs that will drive future performance improvements in computing long after the Moore’s Law era of “cheap transistors” ends.

The Rogues Gallery is a testbed for unorthodox hardware that might otherwise exist only as first-generation start-up or research lab prototypes. The primary goal is to make unique hardware available to a wide audience of researchers and developers as soon as it is almost sufficiently stable for testing, application development, and benchmarking. The Rogues Gallery makes cutting-edge hardware available to a wide variety of researchers and application developers. Examples of such cutting-edge hardware include Emu Technology’s Chick, FPGA-based memory-centric platforms, Field Programmable Analog Devices (FPAAs), and others. Even companies like Intel and IBM are investigating novel hardware, Loihi and TrueNorth respectively.

The Rogues Gallery provides a coherent, managed environment for exposing new hardware and supporting software and tools to students, researchers, and collaborators. This environment allows users to perform critical architecture, systems, and HPC research, and enables them to train with new technologies so they can be more competitive in today’s rapidly changing job market. As part of this goal, CRNCH and Georgia Tech are committed to partnering with vendors to define a flexible access management model that allows for a functional and distinctive research experience for both Georgia Tech and external users, while protecting sensitive intellectual property and technologies.

Not all rogues become long-term products. Some fade away within a few years (or be acquired by companies that fail to productize the technology). The overall infrastructure of a testbed focused on rogues must minimize up-front investment to limit the cost of “just trying it out” with new technology. As these early-access and prototype platforms change, the infrastructure must also adapt.

And even if the technology is fantastic, rogues that do not develop communities do not last. In our opinion, initial communities grow by easing use and access. Again, people want to just try the system. Some, like the Emu Chick, require substantial thought around restructuring algorithms and data structures. Easing access permits eager users, often called students, to play with ideas. Georgia Tech has a history of providing such early access through efforts such as the Sony-Toshiba-IBM Center of Competence for the Cell Broadband Engine Processor [1], the Keeneland GPU cluster [2], and work with early high core count Intel-based platforms [3].

**II. INITIAL ROGUES**

We list our current rogues from least to most experimental.

---

**A. FPGAs for Memory System Exploration**

The Rogues Gallery currently has two types of reconfigurable platforms of note. Two Nallatech 385 PCIe boards provide access to a mid-grade Intel Arria 10 FPGA while Microns EX-700 PCIe backplane and AC-510 Xilinx-based module (shown in Figure 1) provides access to a Hybrid Memory Cube 1.0 [4] and FPGA prototype system. The Micron platform currently has upwards of 10 local users and has resulted in several recent publications [5], [6] focused on characterization and utilization of the 3D stacked memory component. Older technology available includes Convey-based FPGA boards [7], and we are currently in the process of installing a Nallatech 520N Stratix 10 PCIe FPGA. Development for both the Intel and Xilinx boards are supported by VMs with access to Intel Quartus and FPGA SDK 18.0, and Xilinx SDAccel 2016.4 software, which is supported by a common network share that is available from all Rogues Gallery VMs and physical machines.

**B. Emu Chick**

The Emu architecture (our Emu “Chick” box is shown in Figure 2) focuses on improved random-access bandwidth scalability by migrating lightweight Gossamer threads, or “threadlets”, to data and emphasizing fine-grained memory access. A general Emu system consists of the following processing elements, as illustrated in Figure 3:

- A common stationary processor runs the operating system (Linux) and manages storage and network devices.
- Nodelets combine narrowly banked memory with several highly multi-threaded, cache-less Gossamer cores to provide a memory-centric environment for migrating threads.
These elements are combined into nodes that are connected by a RapidIO fabric. The current generation of Emu systems include one stationary processor for each of the eight nodes built within a node. A more detailed description of the Emu architecture is available elsewhere [8], [9].

A launched Gossamer thread only performs local reads. Any remote read triggers a migration, which will transfer the context of the reading thread to a processor local to the memory channel containing the data. The Emu system keeps thread migration overhead to a minimum by limiting the size of a thread context to less than 200 bytes, implementing the transfer efficiently in hardware, and integrating migration throughout the architecture. Any operating system requests are forwarded to the stationary control processors through the service queue.

The highly multi-threaded Gossamer cores read only local memory and do not have caches, avoiding cache coherency traffic. Additionally, “memory-side processors” provide atomic read or write operations that can be used to access small amounts of data without triggering unnecessary thread migrations. A node’s memory size is relatively large with standard DDR4 chips (64 Gib) but with multiple, Narrow-Channel DRAM (NCDRAM) memory channels (8 channels with 8 bit interfaces to the host using FIFO ordering). Each DIMM has significantly smaller page and row sizes than typical DRAM, reducing the overall bandwidth but providing many more independent, narrower access channels. Because of this, it can sustain more simultaneous fine-grained accesses than a traditional system with fewer channels and the same peak memory bandwidth. Such fine-grained access patterns do not use the full transfer size of higher bandwidth designs, leading to very poor bandwidth utilization.

The Emu Chick provides an interesting use case as an “intermediate” level rogue in that it has a basic Linux kernel but not a full operating system, which allows for some deployment of packages but not full integration with traditional scheduler and management infrastructure.

C. FPAA: Field Programmable Analog Array

The Field Programmable Analog Array (FPAA) [10], developed at Georgia Tech by Dr. Jennifer Hasler’s group, implements a combined analog and digital board that can implement many analog and neuromorphic architectures [11], [12]. The FPAA combines ultra-low power processors, a 2D array of floating-gate analog plus digital blocks, with a driving 16-bit MSP430 microprocessor, and the exploration and development platform consists of a USB-attached board with multiple analog input ports (Figure 4). All of the development tools for the FPAA and MSP430 are provided as free and open-source software based on Scilab and XCos. The tools and environment are distributed as an Ubuntu 12.04 virtual machine and used for undergraduate coursework at Georgia Tech.

The rogue portion of the FPAA comes from its 2D array of combined floating-gate analog units plus programmable digital units along with its routing structure. Even manufactured with 350nm CMOS, the FPAA uses 23 µW to recognize the word “dark” in the TIMIT database [11]. Similarly, classifying acoustic signals from a knee joint requires 15.29 µW [12]. Both of these types of computations are performed in real time, so these power savings translate directly to energy savings and help to provide justification for further research in mixed analog and digital hardware for multiple applications. As the FPAA is a remote USB device, it provides some challenges in terms of how to manage it in the Rogues Gallery in terms of monitoring, scheduling, and maintenance.

D. Power Monitoring and Instrumentation

We envision three types of instrumentation and introspection that are available in the Rogues Gallery. These added devices will enable additional insight into power, thermal, and data characteristics of applications that are deployed on different “rogues” within the testbed. 1) The Rogues Gallery currently hosts a USB-addressable power device, the PowerMon [13] which provides 1 MHz readings on up to 6 ports and is connected to a small Tegra cluster containing TK1, TX1, and TX2 boards. Similarly, we currently host one WattProf [14] PCIe power sensor which provides a flexible tool to measure many types of system power and to correlate it with a segment of code using instrumented code regions. In both cases, we
typically have to splice a AC/DC adapter cable to allow for an in-line connection to either the PowerMon or WattProf sensor, but users can then interact with the device remotely via calls to a USB or PCIe device. 2) We hope to install wired thermal sensors in the near future that will be used to measure the thermal scaling of devices such as 3D stacked processors, as was done manually in recent work at GT [5] and which has been supported by projects like PowerInsight [15]. 3) We are also investigating incorporating network-based introspection using Mellanox programmable NICs (like the ConnectX 4-LX), which can be used to perform in-situ analysis of data as it moves across networked nodes or which can be used to support higher-level runtime research.

III. MANAGEMENT ISSUES

Figure 5 shows the management and networking outline of the Rogues gallery. Here we discuss different components of the testbed from the identity management, scheduling, tools support, and networking viewpoints. As demonstrated in the figure, many of the tools and development platforms are hosted on virtual machines while tools are made available via containers where available, as is the case with the Emu Chick simulator and toolchain and the FPAA development VM. The structure is complex representing the diversity of tools and platforms.

Furthermore, management of hardware may be complicated because man of the “rogues” might be temporary additions to the testbed, especially if they fail to build a local userbase or if a company decides to discontinue tool and compiler support. One important implication is that the start-up cost of incorporating a new rogue must be kept low. Currently, we plan to reevaluate the hardware composition of the testbed each year and confer with users and corporate colleagues to make sure that sysadmin resources are focused on an keeping an up-to-date set of rogues and associated software stacks.

We emphasize using existing tools although sometimes through an extra level of indirection. For example, we cannot fully integrate the Emu Chick into Slurm at this moment since each Chick node cannot run slurmd, but we can manage queue-based access to a VM used to access the Chick.

A. Networking Structure

The networking structure is designed to protect sensitive intellectual property and licenses as well as provide general security. Networked devices are attached to firewalling switches that control host and port accesses. Generally inbound access is allowed in the directions of the arrows in Figure 5 and only for ssh. Outbound access generally is not restricted to allow fetching public data sets from various online sources. The administrative nodes can access everything directly.

Some links effectively are managed by the Slurm scheduler via a PAM module. The FPGA development nodes and FPGA-equipped nodes can access each other if co-scheduled. This allows faster development cycles while also permitting users that need only the tools or only the hardware not to prevent other access.

One convenient aspect of completely unusual architectures under heavy development, like the Emu Chick, is that they do not present serious attack targets. Even if someone illicitly accessed the Chick, they would not find many tools to do damage. However, access is managed to prevent leaking system specifics to inappropriate parties.

B. Identity Management

The Rogues Gallery has two types of identity management that are tied in with its notion of scheduling unique devices. At a high-level, new users request access to the testbed via a publicly available webform tied to the local administrative mailing list. Once approved, users are set up with a new username and given access to specific subsets of resources. For example, we currently have groups for neuromorphic computing, reconfigurable devices, and the Emu Chick. Each has sub-groups for normal users and admins (with sudo access to related physical devices and VMs). This setup integrates with Georgia Tech’s campus authentication system (CAS) as well as our robust in-house role (GRS) and entitlement system (GTED). This allows for users to self-manage passwords while system administrators can define LDAP accessible groups for access control as well as POSIX groups at the filesystem level to control file/directory access — important on some network mounted file shares with various tools/data/documents throughout the lab.

More challenging is identity management for systems that have a limited idea of what a “user” might be. For example, while the Emu Chick has a basic Yocto-generated Linux OS [16] that runs on the controller node and compute nodes, it does not currently support the subset of packages needed to integrate with our LDAP-based authentication system for users. When the Chick initially arrived, it only supported root access. This is not uncommon for early hardware. Likewise, embedded style devices like standalone FPGA boards (such as the Arria
10 devkit) typically run a micro-kernel and have a limited concept of simultaneous user access or limited access to the FPGA fabric. These devices need to be accessible only through a fully featured front-end that can manage user identity. In the case of different FPGAs in the same system, this needs coupled to OS-level isolation like Linux control groups. We are testing the fine-grained controls as we roll out the scheduler to these resources.

C. Scheduling and Management of Resources

Currently, access to several of the Rogues Gallery resources are maintained using soft scheduling via the group’s mailing list and Slack channels. Other sites use calendaring systems. Resource slots are represented as conference rooms to be scheduled. These work well for small numbers of users and resources and require essentially no setup. The looser mechanisms do require a small, friendly user community.

As our user base grows, we are bringing the resources under control of the Slurm batch scheduler [17]. Slurm already supports generalized resources (GRES) to handle licenses and device allocation. Slurm also supports accounting which quantifies the user base for grants, etc. We use a single Slurm controller for all the devices, but each kind of device has its own partition. Currently with a small number of each type of hardware, we have not needed to worry about careful queue controls or complex solutions for co-scheduling heterogeneous resources.

Some systems are relatively easy to bring under Slurm control. The FPGA development and hardware systems are a somewhat typical case of allocating licenses and accelerator nodes. Our only complication is separating the development machines from the hardware machines. Many researchers start up needing only the compilation and development tools. Others have their generated FPGA bitstreams and only need the hardware for experiments. And some will need both for rapid final development turn-around. The resources need to be scheduled both separately and together, again a use already supported by Slurm. The pam_slurm_adopt module can clean up cross-connections for mutual allocations. We have scripts generating these configurations, but the setup is admittedly site specific and somewhat fragile.

The power monitor does not need scheduled, but the resources being monitored do. The resources like the Tegra nodes do not need to be accessible to anything but the power monitor, so this odd case could use simple reservations and not run anything remotely. Right now, however, we do not worry about anything beyond ensuring only the allocated user can access the appropriate Tegra node. The Tegra nodes support PAM, so that works out of the box. We could build a more complex system that carefully controls access, but we have yet to see a need.

Other systems are more complicated. While Slurm cannot run on the Emu Chick directly, it can manage allocating the Chick’s nodes via GRES on the Emu development node. This still requires cooperation between users; we cannot control access to individual nodes without more effort than it is worth. Also, users need root access to the control node to reboot individual nodes. Ideally, Slurm could reconfigure the Chick between the eight single nodes and one eight-node configurations, but that reconfiguration is not sufficiently stable to script (at the time of writing). But we can manage multi-tenant access to the Chick, one user per node, along side single-user access to the entire Chick through different Slurm partitions. This is similar to sharing a multi-core server while still permitting whole-node jobs. These are issues with many first-generation hardware platforms. The risks are worth the benefits in an academic environment.

And then there are the FPAAAs (Section II-C). These are USB-connected devices. Our plan is to manage access to the FPAA host (possibly with something as simple as a Raspberry Pi) as with other accelerated systems. But the current tools assume a direct USB connection. We are experimenting with tunneling USB/IP for access and permitting power toggling via uhubctl. Remote power control is important here. Having to be present to power-cycle the experimental device would greatly limit its usability. Also, foreign students increasingly require an escort into shared machine rooms. And experimenting with these tools very much is a student task. Otherwise this may require too much (paid) effort compared to the gain of a possibly short-lived device; the next generation of hardware is in progress. The students gain the experience of an early system where not everything can be installed from a package manager.

D. Tool support

VMs for tools and users are currently provisioned via a RHEV 3.5 cluster of 12 nodes (each with dual E5-2620 CPUs and 192 GiB of DRAM), with 2 nodes specifically serving research VMs like those used by the Rogues Gallery. The standard image for most of our testbed research is Ubuntu 16.04.5 LTS with one or two CentOS VMs as needed for specific toolsets.

Early hardware may require its own, specific OS and dependency stack. New hardware companies rarely can invest in wide software support. Virtual machines and light-weight containers come to the rescue. We use Singularity [18] to package the Emu toolset and VirtualBox [19] for the FPAA tools.

Singularity wraps the Emu compiler and simulator in a manner convenient for command-line build systems. We can offload compilation and simulation to laptops and other platforms that are less resource-constrained than the VMs. Only users with access to the Emu development node have access to the Singularity image. Because of Emu’s rapid development, we do not worry about old versions “in the wild” after students graduate; they provide no useful information outside of what is publicly available.

The Georgia Tech FPAA tools [20] currently consist of a graphical environment using Scilab and Xcos. These need USB

\footnote{1http://usbip.sourceforge.net/}

\footnote{2https://github.com/mvp/uhubctl}
access to the FPAA, so an environment like VirtualBox is more appropriate to encompass the GUI, compiler tools, simulation, and example code. As mentioned in Section III-C, we are experimenting with methods for remote USB access. Here a major unsolved aspect is providing appropriate physical input to an FPAA device. We can replay data via the digital interface, but that rarely excites students as much as real device data.

FPAA tools often have license restrictions that prevent wrapping everything into nice container images. However, these are commercial tools that almost always support at least one of the major, current enterprise Linux distributions.

We manage the specific tools on a mid-range, dual-socket Intel Xeon system (IBM System x3650 M4) with 192GiB of RAM, a 250GB NVMe drive, and a 500GB ZFS raidz-1 volume. Resources remotely NFS mount the specific tools. This does require administrative support for deploying new tools, but managing a single host is simpler than managing tools on each individual resource. Currently tool versioning is provided by a top-level symbolic link in each resource’s top-level directory while keeping older versions accessible when useful; our needs are relatively simple.

E. Monitoring the Rogues Gallery

An instance of OpenNMS monitors the Rogues Gallery. Alerts are sent to support personnel regarding any change in the availability of systems in case something unexpected/unscheduled occurs. We also use monitored power distribution units for reports/trends as well as alerts if power usage gets dangerously high (intentionally or not).

However, things are complicated with some of the more novel systems in the Rogues Gallery. For example, with the Emu Chick, we would like to monitor the system management board but not be notified every time a user resets an internal node unless there is an issue. We are investigating how to use tools like ssh-ping to supplement OpenNMS queries for the management node without interfering with other system uses. The FPAA provides a similar challenge in that it is a USB device attached to a related physical host. We would like to check the USB device’s online status without interfering with any active activity and also communicate the availability of the resource to our Slurm scheduler. Our current target for performing this type of monitoring is to extend LBNL’s Node Health Check [21] script for Slurm to support monitoring (and possibly restarting) specific USB devices.

Monitoring individual FPGA resources without disturbing running experiments similarly is complicated. Currently these devices must be managed as either USB or PCIe devices using node-health scripts. Platforms with an embedded ARM core like Xilinx’s Zync board or the Intel Arria10 DevKit provide a basic Linux-enabled core, but it is not clear that these on-board processors can host meaningful monitoring and/or perform communication with a global monitor or scheduler.

IV. Reproducibility and Replicability

Many major research venues push for reproducible, or at least replicable, experiments. In general this is a wonderful direction. For early, novel platforms, this may not be easily achievable. Some of our platforms, like the Emu Chick, fix show-stopping bugs with frequent software and firmware releases. Reverting to earlier versions to reproduce experiments requires system-wide pain.

Other Emu Chick installations have used our benchmarks and test codes, and this has helped Emu Technology identify hardware and software issues. Some are available online¹ to help reproduce comparative results on other platforms. Others are not generally available and would not be useful anyways. The platform’s software interface is changing as well, in part due to research undertaken over the past year using the Rogues Gallery.

The high-initial-effort “Review of Computational Results Artifacts” option in [22] still is possible and worth-while. Instilling these ideas in students will pay off over the long run but does require more initial effort than last-minute preparation permits. And sometimes last-minute preparation allows results on newer system versions that provide important new results. Re-running all timer precision tests may not be feasible. Balancing between replication and deadlines is an open issue.

V. Future Plans

Clear future plans include extending the infrastructure and making new system integration easier. Collaborating with other rogue-like centers, including CENATE at Pacific Northwest National Lab and ExC/L at Oak Ridge National Lab, could expose abstractions useful across all such efforts. Adopting new platforms and combining them with upcoming technology like large-scale nonvolatile memory is on the roadmap. A recent NSF award opens more pathways for exposing this infrastructure to a wider community through XSEDE [23]. Integrating the Globus tools with the rogue architectures will be an interesting challenge but will permit larger scaling and data movement to the host VMs and shared storage that support tools and front-end testing. But there are more future possibilities out there, including evaluating platforms using real-world but sensitive data and providing a gateway to remote rogue systems.

Many interesting applications for these novel platforms involve sensitive data in areas like corporate networks [24] and health care [25]. A secure infrastructure to pipe sensitive data through our testbed would benefit both application and hardware / software designers. Application designers learn what upcoming hardware could be useful while hardware and software designers would discover which of their assumptions apply to specific, real-world application areas.

Current VLAN, VXLAN, and overlay network methods combined with some assurance for wiping data may suffice for non-regulated industry applications. Per-VM memory and storage encryption combined with the Science DMZ design [26] could assist with data transfer. However, for some levels of certification, particularly in health care, this may not be feasible. Identifying specific roadblocks could help new

¹[https://github.com/eheinf6/emu-microbench](https://github.com/eheinf6/emu-microbench)
platform developers engineer around them and possibly provide regulatory agencies guidance otherwise unavailable.

We also have not discussed the deployment of “far-off” devices like those in the quantum computing space in our current testbed. Many existing quantum devices require extensive physical hosting requirements that are out of reach for nearly all facilities. However, we envision hosting common sets of quantum tools like those provided by the ProjectQ team [27] and engaging with smaller quantum start-ups to provide remote access for researchers, perhaps allowing our testbed to become a gateway for this hardware or even a “Rogues Grid.”

VI. SUMMARY

The CRNCH Rogues Gallery at Georgia Tech lets people play with new and novel architectures, up to administrative requirements like licensing, nationality limitations, and overall network security considerations. Reducing the intellectual cost of trying new systems brings far more applications, sometimes unexpected, into the platforms’ evaluations. The Rogues Gallery assists with porting software and otherwise building a community. Eventually, such actions give rogues some respectability as commercial products. Even in the Rogues Gallery, we have found significant success. Some systems have external users investigating their application. Experiments have produced multiple academic publications. One Ph.D. student graduated and joined one of the rogue companies. Tools developed using the rogues assist outside users and are being adopted by the platforms’ producers.

There are multiple similar architecture and system evaluation labs, each with their own special benefits. The Rogues Gallery’s lie in our focus on flexible infrastructure and, most importantly, our students.

VII. ACKNOWLEDGMENTS

This work is supported in part by Micron’s and Intel’s hardware donations, the NSF XScala project (ACI-1339745), the NSF SuperSTARLU project (OAC-1710371), and IARPA. Thanks also to colleagues at GTRI including David Ediger and Jason Poovey and users like Vinip Sachdeva for ideas and assistance with scheduling and FPGA support using Slurm. In addition, thanks to Eric Hein, Janice McMahon and the rest of the team at Emu for their assistance in setting up and supporting the Emu system, we thank reviewers for their attentive comments.

REFERENCES

[1] D. A. Bader and V. Agarwal, “FFTC: Fastest fourier transform for the IBM Cell Broadband Engine,” Lecture Notes in Computer Science, pp. 172–184, 2007.
[2] J. S. Vetter, R. Glassbrook, J. Dongarra, K. Schwan, B. Loftis, S. McNally, J. Meredith, J. Rogers, P. Roth, K. Spafford et al., “Keeneland: Bringing heterogeneous GPU computing to the computational science community,” Computing in Science & Engineering, vol. 13, no. 5, pp. 90–95, 2011.
[3] J. Riedy, H. Meyerhenke, D. A. Bader, D. Ediger, and T. G. Mattson, “Analysis of streaming social networks and graphs on multicore architectures,” in International Conference on Acoustics, Speech and Signal Processing (ICASSP), Kyoto, Japan, Mar. 2012.
[4] Hybrid Memory Cube Specification 1.0, Hybrid Memory Cube Consortium, 2013.
[5] R. Hadidi, B. Asgari, B. A. Mudassar, S. Mukhopadhyay, S. Yalamanchili, and H. Kim, “Demystifying the characteristics of 3D-stacked memories: A case study for Hybrid Memory Cube,” in 2017 IEEE International Symposium on Workload Characterization (IISWC), Oct 2017, pp. 66–75.
[6] R. Hadidi, B. Asgari, J. Young, B. A. Mudassar, K. Garg, T. Krishna, and H. Kim, “Performance implications of NoCs on 3D-stacked memories: Insights from the Hybrid Memory Cube,” in 2018 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), April 2018, pp. 99–108.
[7] K. K. Nagar and J. D. Bakos, “A sparse matrix personality for the Convoy HC-1,” in 2011 IEEE 19th Annual International Symposium on Field-Programmable Custom Computing Machines, May 2011, pp. 1–8.
[8] T. Dysart, P. Kogge, M. Deneroff, E. Bovell, P. Briggs, J. Brockman, K. Jacobsen, Y. Jia, and J. Lethin, “Highly scalable near memory processing with migrating threads on the Emu system architecture,” in Workshop on Irregular Applications: Architecture and Algorithms (IRAS), IEEE, 2016, pp. 2–9.
[9] E. Hein, T. Conte, J. S. Young, S. Esvar, J. Li, P. Lavin, R. Vuduc, and J. Riedy, “An initial characterization of the Emu Chick,” in The Eighth International Workshop on Accelerators and Hybrid Exascale Systems (AsHES), May 2018.
[10] S. George, S. Kim, S. Shah, J. Hasler, M. Collins, F. Adil, R. Wunderlich, S. Nease, and S. Ramakrishnan, “A programmable and configurable miniature FPA SoC,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 24, no. 6, pp. 2253–2261, June 2016.
[11] J. Hasler and H. Marr, “Finding a roadmap to achieve large neuromorphic hardware systems,” Frontiers in Neuroscience, vol. 7, p. 118, 2013.
[12] S. Shah, C. N. Teague, O. T. Inan, and J. Hasler, “A proof-of-concept classifier for acoustic signals from the knee joint on a FPA,” in 2016 IEEE SENSORS, Oct. 2016, pp. 1–1.
[13] D. Bedard, M. Y. Lim, R. Fowler, and A. Porterfield, “PowerMon: Fine-grained and integrated power monitoring for commodity computer systems,” in Proceedings of the IEEE SoutheastCon 2010 (SoutheastCon), March 2010, pp. 479–484.
[14] M. Rashiti, G. Sabin, D. Vansickle, and B. Norris, “WattProf: A flexible platform for fine-grained HPC power profiling,” in 2015 IEEE International Conference on Cluster Computing, Sept 2015, pp. 698–705.
[15] J. H. Laros, P. Pokorny, and D. DeBonis, “PowerInsight — a commodity power measurement capability,” 2013 International Green Computing Conference Proceedings, Jun. 2013.
[16] O. Salvador and D. Angolini, Embedded Linux Development with Yocto Project. Packt Publishing, 2014.
[17] M. A. Jette, A. B. Yoo, and M. Grondona, “SLURM: Simple Linux Utility for Resource Management,” in In Lecture Notes in Computer Science: Proceedings of Job Scheduling Strategies for Parallel Processing (JSSPP) 2003. Springer-Verlag, 2002, pp. 44–60.
[18] G. M. Kurtzer, V. Sochat, and M. W. Bauer, “Simplicity: Scientific containers for mobility of compute,” PLOS ONE, vol. 12, no. 5, p. e0177459, May 2017.
[19] P. Dash, “Getting started with Oracle VM VirtualBox,” 2013.
[20] M. Collins, J. Hasler, and S. George, “An open-source tool set enabling analog-digital-software co-design,” Journal of Low Power Electronics and Applications, vol. 6, no. 1, 2016.
[21] M. Jennings, “Node health check,” https://github.com/mej/nhc, 2018.
[22] M. A. Heroux, “Editorial: ACM TOMS replicated computational results initiative,” ACM Trans. Math. Softw., vol. 41, no. 3, pp. 13:1–13:5, Jun. 2015.
[23] J. Towns, T. Cockerill, M. Dahon, I. Foster, K. Gaither, A. Grimshaw, V. Hazlewood, S. Lathrop, D. Lifka, G. D. Peterson, and et al., “XSEDE: Accelerating scientific discovery,” Computing in Science & Engineering, vol. 16, no. 5, p. 6274, Sep. 2014.
[24] T. E. Senator, H. G. Goldberg, A. Memory, W. T. Young, B. Rees, R. Pierce, D. Huang, M. Reardon, D. A. Bader, E. Chow, I. Essa, J. Jones, V. Bettadapura, D. H. Chau, O. Green, O. Kaya, A. Zakrzewska, E. Briscoe, R. I. L. Mappus, R. McCell, L. Weiss, T. G. Dietterich, A. Fern, W.-K. Wong, S. Das, A. Eimatt, J. Irvine, J.-Y. Lee, D. Kourra, C. Faloutsos, D. Corkill, L. Friedland, A. Gentzel, and D. Jensen, “Detecting insider threats in a real corporate database of computer usage activity,” in Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ser. KDD ’13. New York, NY, USA: ACM, 2013, pp. 1393–1401.
[25] J. Duke, “Precision medicine at Georgia Tech: Introduction to the Health Data Analytics Platform,” Georgia Institute of Technology, Center for
[26] D. Eli, R. Lauren, T. Brian, H. Mary, and Z. Jason, “The Science DMZ: A network design pattern for data-intensive science,” *Scientific Programming*, vol. 22, no. 2, pp. 173–185, 2014.

[27] D. S. Steiger, T. Hänner, and M. Troyer, “ProjectQ: an open source software framework for quantum computing,” *Quantum*, vol. 2, p. 49, 2018.