Enabling Power Measurement and Control on Astra: The First Petascale Arm Supercomputer

Ryan E. Grant, Simon D. Hammond, James H. Laros III, Michael Levenhagen,
Stephen L. Olivier, Kevin Pedretti, H. Lee Ward, Andrew J. Younge
Center for Computing Research
Sandia National Laboratories
Albuquerque, New Mexico, USA
{regrant, sdhammo, jhlaros, mjleven, slolivi, ktpedre, lee, ajyoung}@sandia.gov

***Abstract*—Astra, deployed in 2018, was the first petascale supercomputer to utilize processors based on Arm's instruction set. The system was also the first under Sandia's *Vanguard* program which seeks to provide an evaluation vehicle for novel technologies that with refinement could be utilized in demanding, large-scale HPC environments. While the adoption of Arm-based processors was the flagship novel technology, several other important first-of-a-kind developments were used in the machine, including new approaches to cooling the data-center and machine, as well as several important innovations in the system imaging and development of a robust programming environment.**

This paper documents our experiences in building a measurement and control infrastructure for power consumption on the Astra machine. While this is often beyond the control of many users today, the accurate measurement, cataloging and evaluation of power, as our experiences show, is critical to the successful deployment of a large-scale platform. While such systems exist in part for other architectures, Astra required new development to support the novel Marvel ThunderX2 processor used in compute nodes. In addition to documenting the measurement of power during system bring up and for subsequent on-going routine use, we present results associated with controlling the power usage of the processor, an area which is becoming of progressively greater interest as data centers and supercomputing sites look to improve compute/energy efficiency and find additional sources for full system optimization.

I. Introduction

After many years of ecosystem development, server-class Arm processors are now readily available from multiple vendors as traditional commercial products and as infrastructure available for rent in the cloud. Platform enablement and standardization efforts [1], [2] have enabled a diverse set of system vendors and OEMs to field compatible Arm servers, and this in turn has enabled the development of a comprehensive and vibrant Arm software ecosystem. The results of this effort are demonstrated by Arm-based systems such as Astra [3] and Fugaku [4] appearing on the Top500 list of the world's fastest supercomputers for the first time, with Fugaku holding the top spot as of June 2020 [5].

Still, there are some areas such as power management that are not yet well understood or standardized for Arm servers. Even within a single vendor ecosystem, different processor models and product generations provide differing power management capabilities. This is amplified in a multi-vendor ecosystem such as Arm, where different vendors may employ completely different approaches. With the relatively recent entrance of Arm into the HPC space, it is important to examine the power management capabilities of existing platforms so that both common functionality and gaps can be identified. This is needed to drive platform standardization efforts in productive directions and to improve the power management capabilities of future Arm-based supercomputers.

This paper describes the power management capabilities of the Astra supercomputer based on the HPE Apollo 70 server product and Marvell ThunderX2 processor, detailing how these capabilities have been enabled and improved over the system's life. This has required close collaboration and co-design between Sandia, HPE, and Marvell, and a willingness to experiment with these settings to discover the various performance, power usage, and thermal tradeoffs. In particular, when problems have been identified, solutions have been implemented via firmware updates and other methods to improve the capabilities of the system in the field. Such improvements will be demonstrated through current Astra operational capabilities and the memory turbo mode evaluated in our experiments.

The remainder of this paper is organized as follows: Section II provides a brief overview of the Astra platform. Section III describes the platform's power measurement capabilities and is followed by Section IV, which describes the available power control mechanisms. Section V provides an empirical evaluation of several workloads operating under different power control configurations. Section VI covers related work. Final conclusions are presented in Section VII.

II. Astra Overview

This section provides a brief overview of the Astra supercomputer and its overall project goals. Astra was originally installed at Sandia National Laboratories in October 2018. Since then, it has undergone significant co-development and stabilization efforts and reached near-production status in October 2019. Astra is currently in use by the NNSA Tri-labs (Sandia, Los Alamos, and Lawrence Livermore national laboratories) to run important modeling and simulation workloads, with jobs running regularly at thousand node scales. A more detailed description of Astra can be found in previous work [3].

A. Hardware Platform

The Astra supercomputer system architecture is an amalgamation of industry standard parts including: the HPE Apollo 70 chassis and node design; Marvell ThunderX2 processors; Mellanox MLX5 EDR InfiniBand network; HPE Lustre storage servers; Red Hat Linux; standard 19-inch racks; power distribution units, and water to air cooling infrastructure. The overall system includes 2,592 compute nodes split into thirty-six racks of eighteen Apollo 70 chassis each, for a total of 72 nodes per rack. The compute nodes are connected by a fat-tree topology InfiniBand network with a 2:1 taper at the rack level. Each 36-port rack-level leaf switch connects 24 downlinks to compute nodes and 12 uplinks to director class core switches.

The Astra node architecture is comprised of two Marvell ThunderX2 28-core ARMv8 processors running at a base frequency of 2.0 GHz and maximum turbo frequency of 2.5 GHz. Each processor socket connects directly to eight DDR4-2666 DIMMs and to an EDR InfiniBand network interface via a dedicated 8x PCIe link. Additionally, each node includes its own Baseboard Management Controller (BMC) that implements the Intelligent Platform Management Interface (IPMI) protocol for out-of-band node management. The overall system is managed by a top-level management node running the HPE Performance Cluster Manager (HPCM) software, which interfaces with each node's BMC.

B. Project Goals

As a large-scale prototype platform, Astra was never intended to be a truly production system. Rather its primary goal was to prove out the viability of Arm for supporting real-world NNSA Tri-lab production applications and input problems. This meant gaining experience and confidence in the Arm ecosystem, both in terms of hardware and software, and demonstrating that it was ready to be used in future production supercomputer deployments. That goal has largely been accomplished, although further maturation efforts continue as they do for any new technology. In particular, improving Arm compiler toolchains and math libraries to better support NNSA workloads are continuing areas of focus.

A secondary goal was to leverage the Astra platform's prototype nature to conduct R&D activities. Areas of exploration have included in-platform container build support, new I/O filesystems, resilience studies, network congestion analysis, machine learning, and, the topic of this paper, power management. Little was known about how power measurement and control worked on Astra's Apollo 70 servers going into the project, and in practice this knowledge became critical for bringing up the system, as will be outlined in the following sections.

III. Power Measurement

When Astra was initially deployed, it was configured for maximum performance with features such as turbo boost and symmetric multithreading (SMT) enabled. Small scale testing appeared to work well, but large scale runs would quickly experience degraded performance after a few minutes. After investigation, the root cause was determined to be thermal throttling. Diagnosing this problem was not straightforward as there was not yet an infrastructure in place to collect and analyze power and temperature measurements throughout the system.

The remainder of this section describes power measurement sources available on Astra, the in-band and out-of-band interfaces for accessing them, and the system monitoring infrastructure that was developed to help diagnose system-wide power and thermal anomalies.

A. Power Measurement Points

1) Processor-level: The Marvell ThunderX2 processor provides extensive on-die voltage, power, and temperature information. Per-core readings are provided for temperature and frequency, and package-level power usage is provided for Core, SRAM, Memory, and SOC (System On Chip) power domains. The total package power for the ThunderX2 processors used in Astra (28-core, 2.0 GHz, 150 W TDP) can be derived by summing the four power domains and adding 12 W. This additional 12W is an approximation of the unreported power used by other components in the package (Marvell, personal communication, January 23, 2019).

2) Chassis & Node-level: The HPE Apollo 70 architecture is comprised of a 4U chassis design which houses four dual-socket Arm server sleds/blades, an array of eight cooling fans, and two shared power supplies that power all of the components in the chassis. The chassis provides physical power measurement of the input AC power to each power supply and the total DC power output supplied to the components in the chassis. Each Arm server sled further provides a physical power measurement of its input DC power, enabling each Arm server to be measured individually.

3) Rack-level: At the rack-level, a single Raritan power distribution unit (PDU) converts three-phase 480VAC facility power to 270VAC power used by the Apollo 70 chassis. The PDU provides physical measurement of total power (active and apparent) and individual rms current readings for each of the PDU's 18 breakers. Additionally, the PDU provides an active energy counter that tracks the total energy consumed by the rack in joules.

4) System-level: Finally, at the full system level, the facility provides 480VAC power via two Starline Busway T5 1200Amp overhead bus bars, one dedicated to each row of Apollo 70 racks. Additionally, Liebert PDUs are used to supply 208VAC to the system's three network switch cabinets, two service node racks, and twelve cooling cabinets. Taken together, the utility grade power meters in these bus bars and PDUs capture the total power consumed by Astra. Notably, the full-system level 3 (highest quality level) power measurements supplied in the June 2019 Top500 submission for Astra were gathered using these measurement points [5].

B. In-band Interfaces

Marvell provides the `tx2mon` Linux utility for accessing the ThunderX2's internal power measurements. The tool consists of a Linux device driver, which must be rebuilt and/or ported for the specific Linux kernel version being used, and a userspace utility for accessing information exported by the device driver. The `tx2mon` utility was originally developed as an internal engineering tool by Marvell, however at the urging of Sandia and other customers, it was made available as open source tool [6].

Figure 1 shows an example of `tx2mon` output for an idle Apollo 70 compute node. Power usage information for each of the two ThunderX2 processors in the server is displayed (Node 0 and Node 1) via a curses-based interface similar to `top`. Output can also be written to a CSV-formatted text file. In addition to displaying real-time power usage, `tx2mon` also provides historical information on the count and duration of throttling events that have occurred since the server booted. This feature was added via a firmware update after Astra's initial deployment, and was helpful for investigating node performance issues.

A PowerAPI [7] plugin for `tx2mon` was developed and deployed on Astra in order to provide a portable in-band interface to access its information [8].

C. Out-of-band Interfaces

The IPMI protocol is the primary out-of-band interface provided for gathering power and environmental information. IPMI requests are serviced by the Apollo 70 chassis and server BMC and do not require any host involvement or other activity that would slow down application performance. IPMI traffic flows over an Ethernet management network to further avoid interfering with applications, which primarily utilize the system's high-speed InfiniBand network.

Figure 2 shows example output of the '`ipmitool sdr`" command used to gather a remote Apollo 70 server's sensor information. The server's current whole-node power usage is displayed near the end of the output as `NodeDCPower`. Notably, the Apollo 70 BMC does not expose an energy accumulation counter, instead requiring explicit polling to estimate a server's energy use. It would be a desirable feature to add to future Apollo systems, similar to the energy counter provided on Cray XC systems [9].

The SNMP protocol can be used to gather out-of-band power measurements from Astra's rack PDUs and cooling cabinets. This traffic flows over the same Ethernet management network used for IPMI.

A PowerAPI [7] plugin for accessing Apollo 70's out-of-band power information was developed. However, it has not been widely deployed because it requires administrator privilege. The Apollo 70 BMC does provide multiple role-based privilege levels, but unfortunately the power information is mapped to the administrator role rather than a user role.

D. System Monitoring

As previously mentioned, Astra faced significant challenges with thermal throttling early in its deployment. The Apollo 70 platform was new, resulting in little experience in debugging such systems. Furthermore, HPCM did not yet have effective support for monitoring the health of the system. These challenges were overcome by disabling turbo boost, SMT, and any other feature that increased server power usage. Additionally, Astra's cooling system was improved by adjusting water flow rates, fan speeds, and sealing air gaps. These changes enabled large-scale runs to complete successfully.

Still, there were lingering thermal issues and system behavior often changed as a result of firmware and configuration updates. To address this gap, HPE and Sandia collaborated to develop a system monitoring solution to periodically gather power and temperature information, store it in a time series database, and visually analyze it using Grafana, a web-based tool for constructing real-time monitoring dashboards. The system was effective in enabling the team to quickly identify and resolve overheating nodes and other thermal anomalies.

As an example, Figure 3 shows a Grafana heat map visualization of the inlet water temperatures over time to Astra's cooling racks, with each row representing a different rack and color representing temperature. This ad hoc plot was quickly generated as part of an investigation of why nodes were overheating and shutting off. The plot clearly shows the unexpected behavior: during mid-day water temperatures were rising considerably, as represented by the red areas in the plot. The problem was root caused as a faulty valve in the data center that wasn't properly alarming, and once fixed the system began operating normally again after five days of issues.

Another example is shown in Figure 4. In this case three clusters of compute nodes can be seen: 1) cool nodes that were not running a job, 2) normal nodes operating at expected levels, and 3) very hot nodes (red points) consuming upwards of 600 W, far beyond the expected maximum of 450 W. This phenomenon was eventually traced to a bug in a recent firmware update that left CPU operating voltages too high on a subset of nodes. Once the error was corrected, the overheating nodes operated normally. Issues like this would have likely been much harder to understand and resolve without the lightweight monitoring and analysis solution that was developed for Astra.

IV. Power Control

This section describes the two main power control mechanisms available on Astra: 1) turbo modes and 2) OS CPU frequency control. This section describes their operation and the evaluation in Section V will present empirical results for several workloads running under different power control configurations.

A. Turbo Modes

As originally designed, the ThunderX2 processor provided three turbo modes:

- Disabled
- Autonomous
- Operating System Controlled (OSPM)

```
Node: 0  Snapshot: 182558986
Freq (Min/Max): 1000/2000 MHz     Temp Thresh (Soft/Max):  92.39/110.25 C

|Core  Temp   Freq |Core  Temp   Freq |Core  Temp   Freq |Core  Temp   Freq |
+-----------------+-----------------+-----------------+-----------------+
|  0:  58.89  2003 |  1:  58.34  2002 |  2:  58.34  2006 |  3:  57.78  2007 |
|  4:  56.66  2005 |  5:  58.89  2006 |  6:  56.66  2000 |  7:  57.78  2000 |
|  8:  58.34  2004 |  9:  57.78  2005 | 10:  58.89  2000 | 11:  56.66  2000 |
| 12:  54.99  2002 | 13:  58.34  2005 | 14:  60.01  2005 | 15:  58.89  2007 |
| 16:  56.10  2002 | 17:  58.89  2003 | 18:  57.22  2002 | 19:  57.78  2000 |
| 20:  57.78  2007 | 21:  58.34  2003 | 22:  56.66  2003 | 23:  57.78  2007 |
| 24:  57.22  2004 | 25:  58.89  2003 | 26:  56.66  2000 | 27:  57.78  2001 |

SOC Center Temp:  60.57 C
Voltage    Core:   0.74 V, SRAM:  0.90 V,  Mem:  0.93 V, SOC:  0.85 V
Power      Core:  18.97 W, SRAM:  0.27 W,  Mem: 13.41 W, SOC: 15.30 W
Frequency    Memnet: 2304 MHz


Throttling Active Events: None
Throttle Events     Temp:      0,    Power:     74,    External:      0
Throttle Durations  Temp:      0 ms, Power:  65547 ms, External:      0 ms

Node: 1  Snapshot: 181930636
Freq (Min/Max): 1000/2000 MHz     Temp Thresh (Soft/Max):  92.39/110.25 C

|Core  Temp   Freq |Core  Temp   Freq |Core  Temp   Freq |Core  Temp   Freq |
+-----------------+-----------------+-----------------+-----------------+
|  0:  51.08  2007 |  1:  51.08  2007 |  2:  49.41  2001 |  3:  49.41  2006 |
|  4:  52.20  2004 |  5:  51.08  2003 |  6:  50.52  2003 |  7:  50.52  2005 |
|  8:  51.08  2000 |  9:  52.20  2005 | 10:  51.64  2004 | 11:  48.85  2002 |
| 12:  51.08  2006 | 13:  50.52  2004 | 14:  52.75  2001 | 15:  48.29  2007 |
| 16:  51.64  2005 | 17:  53.31  2000 | 18:  51.08  2002 | 19:  53.87  2003 |
| 20:  51.64  2004 | 21:  51.64  2002 | 22:  48.85  2003 | 23:  49.96  2000 |
| 24:  51.08  2001 | 25:  51.08  2004 | 26:  51.08  2006 | 27:  49.41  2002 |

SOC Center Temp:  52.20 C
Voltage    Core:   0.74 V, SRAM:  0.90 V,  Mem:  0.93 V, SOC:  0.85 V
Power      Core:  19.47 W, SRAM:  0.45 W,  Mem: 14.14 W, SOC: 14.62 W
Frequency    Memnet: 2304 MHz


Throttling Active Events: None
Throttle Events     Temp:      0,    Power:     99,    External:      0
Throttle Durations  Temp:      0 ms, Power:  76150 ms, External:      0 ms


['q' to quit, any other key for refresh.]
```

Fig. 1. Example output from Marvell's `tx2mon` tool for displaying real-time ThunderX2 processor frequency, power, and thermal information.

```
Inlet ambient    | 24 degrees C      | ok
CPU 1            | 59 degrees C      | ok
CPU 2            | 54 degrees C      | ok
P1 DIMM 1-4      | 44 degrees C      | ok
P1 DIMM 5-8      | 42 degrees C      | ok
P2 DIMM 1-4      | 38 degrees C      | ok
P2 DIMM 5-8      | 35 degrees C      | ok
System Board     | 30 degrees C      | ok
LOM              | 68 degrees C      | ok
LOM Zone         | 54 degrees C      | ok
Sys Exhaust      | 53 degrees C      | ok
I/O              | 92 degrees C      | ok
I/O Zone         | 56 degrees C      | ok
Fabric Bd Zn     | 40 degrees C      | ok
P/S 1 Inlet      | 33 degrees C      | ok
P/S 2 Inlet      | 33 degrees C      | ok
Fan 1 PWM        | 35 percent        | ok
Fan 2 PWM        | 35 percent        | ok
Fan 5 PWM        | 35 percent        | ok
Fan 6 PWM        | 35 percent        | ok
InputACpower     | 800 Watts         | ok
TotalDCpower     | 752 Watts         | ok
NodeDCpower      | 184 Watts         | ok
```

Fig. 2. Example `ipmitool` tool output showing an idle Apollo 70's environmental information. The output is abridged to show the most relevant fields.

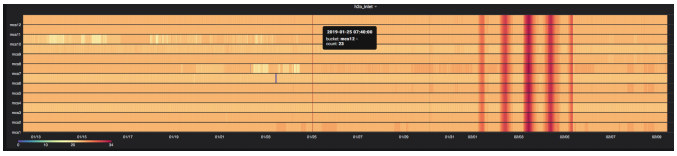


Fig. 3. Heat map visualization of inlet water temperatures over time. Each Astra cooling rack is represented by a horizontal row with time progressing on the X-axis and color indicating temperature (red=hot, yellow/orange=normal). The plot shows 5 days where water temperatures were unexpectedly elevated.

The disabled mode is self explanatory and results in the processor operating at its base CPU frequency of 2.0 GHz. The autonomous turbo mode enables CPU frequency to dynamically scale up to 2.5 GHz based on available thermal and power headroom. Finally the OSPM mode enables the OS to determine the CPU frequency, with settings available from 1.0 GHz to 2.5 GHz for Astra. If CPU frequency is set above the processor's base frequency, the CPU may operate at a frequency less than or equal to the desired setting based on thermal and power constraints.

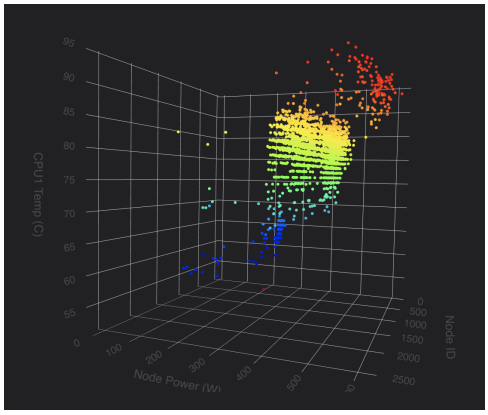


Fig. 4. Power vs. Temperature visualization of Astra's 2,592 compute nodes. Each node is represented by a point with color indicating the node's temperature (red=hot, blue=cool).

B. Memory Turbo

The turbo modes are implemented by an embedded Arm power control processor on the ThunderX2 package. This enables them to be updated and extended through firmware updates. In March of 2019, a new turbo algorithm called memory turbo was co-developed and deployed on Astra in order to boost memory bandwidth while still staying within the system's thermal constraints. Memory turbo is not a distinct turbo mode, but rather an option that can be enabled for Autonomous or OSPM turbo modes.

With memory turbo enabled, CPU frequency remains fixed while memory frequency is allowed to increase to its maximum value (2.3 GHz for Astra) based on thermal and power constraints. The 15% increase in memory frequency (2.0 GHz to 2.3 GHz) resulted in Astra's per-node memory bandwidth improving by 13.6% (220 GB/s to 250 GB/s).

C. OS Frequency Control

Once memory turbo was deployed there was still a desired power management feature missing—in-band operating system control of CPU frequency. To enable this, the ThunderX2 must be configured for OSPM turbo mode with Collaborative Processor Performance Control (CPPC) mode enabled. Initial testing in 2018 with CPPC enabled had resulted in degraded performance, which was due to an OS/firmware interaction bug resulting in CPU frequency getting stuck at a low setting. By mid 2019 that issue had been resolved and testing showed that CPPC mode could be enabled without adverse effects.

Figure 5 shows the behavior of OS frequency control (i.e., OSPM Turbo / CPPC enabled) with and without memory turbo enabled. For convenience, throughout this paper the configuration with memory turbo enabled is called MEMTURBO and the mode with it disabled is called CPUTURBO. In the CPUTURBO configuration, CPU frequency and memory frequency increase in unison up to 2.3 GHz and then CPU frequency can further increase to 2.5 GHz. In the MEMTURBO configuration, memory frequency remains fixed at 2.3 GHz and CPU frequency cannot increase beyond a cap of 2.0 GHz.

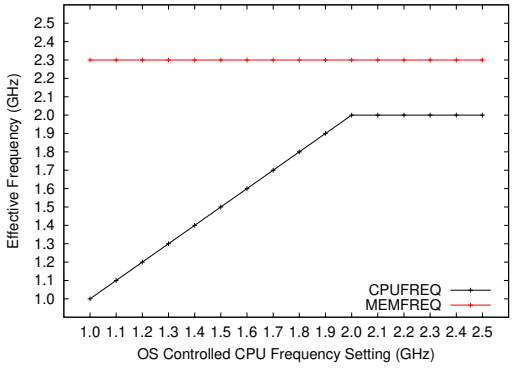
Stated differently, the CPUTURBO configuration allows the OS to set a CPU frequency that may result in thermal throttling for some workloads, particularly those that are CPU or memory intensive and that run on a large number of nodes. MEMTURBO has been carefully engineered to not allow this to happen for any OS frequency control setting. Intelligent system software could potentially take advantage of CPUTURBO to improve performance for some workloads while proactively managing thermals to avoid exceeding limits.

V. EVALUATION

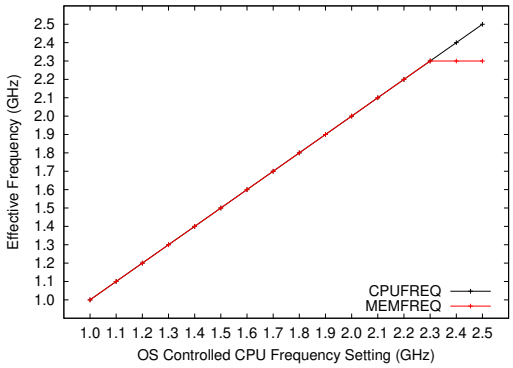
This section presents empirical results of a study comparing two turbo configurations of interest, MEMTURBO and CPUTURBO. The goal of the study was to better characterize the performance, power, and thermal implications of switching from the baseline MEMTURBO configuration to CPUTURBO. In particular, an important question was whether real applications such as Sandia's SPARC CFD simulation code could benefit from CPUTURBO without exceeding thermal constraints.

A. Testing Procedure

The experiments were conducted on Stria, which is a small-scale testbed system with hardware and software identical to



(a) MEMTURBO



(b) CPUTURBO

Fig. 5. Comparison of MEMTURBO vs. CPUTURBO for the ThunderX2 processors used in Astra. Effective frequencies above 2.0 GHz may be reduced during operation by the turbo algorithm due to power and thermal constraints.

Astra. The single-node HPL, STREAM, and HPCG experiments were run using the same physical node to eliminate part-to-part variability and location differences. The four-node SPARC experiments were run on the same set of nodes for the same reasons.

The test nodes were rebooted into either MEMTURBO or CPUTURBO configurations. Then for each configuration, each workload was run at different OS CPU frequency control settings, sweeping a range of settings from 1.0 GHz to 2.0 GHz for MEMTURBO and 1.0 GHz to 2.5 GHz for CPUTURBO (see Figure 5). Particular attention was paid to CPUTURBO settings above the 2.0 GHz baseline, since those were the settings which had potential for increased performance.

While each workload was running, the `tx2mon` tool was used to collect in-band power and temperature information at 1 Hz, writing results to a CSV file. Additionally, the Astra system monitoring infrastructure collected out-of-band information that could later be accessed via SQL queries and Grafana visualizations. The collected in-band information was useful for getting a fine grained picture of how ThunderX2 processor-level power and temperatures were changing over time, while the out-of-band data provided a coarse-grained view of node-level power usage and thermals.

A several minute cool down period was enforced between each run in order to start out from a neutral thermal environment.

B. Workloads

The following workloads were used in the evaluation:

1) Idle Power: This test simply measured the power consumption of an idle node that was not running a workload. Out-of-band samples were taken for approximately 10 minutes and the average power reported.

2) STREAM: The STREAM benchmark [10] is designed to measure memory bandwidth of a computing system. It is a suite of four simple kernels: Copy, Scale, Add and Triad. The kernels are applied elementwise to arrays that are meant to be the larger of 1M elements or at least 4X the size of the last level cache for the machine under test. The performance results for the STREAM benchmark reported in this paper are for the Triad kernel. The figure of merit is sustained memory bandwidth (GB/s).

The STREAM test was configured to use 105 GB of memory and to run for approximately nine minutes in the baseline MEMTURBO 2.0 GHz configuration. OpenMP was used to parallelize across the node's 56 cores.

3) HPL: High Performance Linpack (HPL) [11] is frequently used for benchmarking of HPC systems, most notably in the semi-annual "Top 500" supercomputer list. The problem is to solve a dense linear system of double precision values using LU factorization with partial pivoting. The figure of merit is GFLOPS (billion floating operations per second), and overall performance typically depends heavily on the floating-point arithmetic capability of the system under test.

The input problem was setup to fill approximately 100 GB of the node's 128 GB main memory and all 56 cores were utilized placing one MPI processes per core.

4) HPCG: High Performance Conjugate Gradient (HPCG) [12], [13] is intended to complement HPL for benchmarking of HPC systems by emphasizing sparse rather than dense linear algebra. It performs a fixed number of conjugate gradient (CG) iterations. Major operations in the computation include sparse matrix-vector multiplication, vector updates, global dot products, and multigrid preconditioning of the matrix using a symmetric Gauss-Seidel smoother. The figure of merit is GFLOPS (billion floating operations per second), weighted to amortize any performance-optimizing operations, and overall performance typically depends heavily on the memory bandwidth of the system under test.

HPE's optimized version of HPCG was used for testing and it was configured for a local problem size of 104x104x104. The test was run in an MPI-only configuration with 56 MPI processes running, one per physical core.

5) SPARC: Sandia's Parallel Aerosciences Research Code (SPARC) [14], [15] is a best-in-class computational fluid dynamics (CFD) simulation capability that has been written from new to exploit next-generation high-performance computing architectures. The latest code, developed over a period of five years under the NNSA's Advanced Technology Development and Mitigation (ATDM) sub-program to explore future application programming approaches, comprises hundreds of thousands of lines in latest language standard C++, utilizing Sandia's Kokkos-based performance portable programming model and Trilinos high-performance, scalable solver framework. In recent studies, scaling to thousands of nodes on the Los Alamos (Haswell/KNL) Trinity [16], Lawrence Livermore Sierra (IBM POWER/NVIDIA Volta GPU) [17] and Sandia Astra (Arm ThunderX2) [18], [19] platforms has been demonstrated with the numerically intensive routines optimized to exploit wide-SIMD units, multi-core threading and GPU-based acceleration [20], [21].

Execution of the code proceeds through a series of iterations, each of which progresses simulated time. During each iteration a series of steps are executed which traverse the mesh permitting values and computation on the mesh elements, followed by a sequence of assembly (construction of a non-linear/linear system) and solve (solution of the system) phases. While the specifics will vary depending on the input problem and configuration of study, profiling of these phases shows the mesh traversal and assembly phases exhibit more latency and cache-based activities leading to more intense use of the processor core, while the solve phases will more often present aggressive use of the memory subsystem and memory bandwidth.

The HIFiRE-1 [22] input problem was used for testing. This problem was too large to fit in a single node's memory, so four nodes were used. SPARC was configured for MPI-only mode, with one MPI process running per physical core (224 MPI processes total). It is worth noting that the SPARC application is proprietary to Sandia and source code is not available publicly. However, its relevance to Sandia's mission

TABLE I
IDLE POWER MEASUREMENTS FOR AN ASTRA COMPUTE NODE

OS Controlled CPU Frequency (GHz)	Per Socket / Whole Node MemTurbo (W)	CpuTurbo (W)
1.0	37 / 152	30 / 136
1.2	38 / 152	31 / 136
2.0 (baseline)	48 / 176	46 / 168
2.5	N/A	62 / 208

capabilities as an important real-world application provided the motivation for examining its power usage and control characteristics.

C. Idle Power Measurements

Table I shows the measured idle power draw for the different configurations. Surprisingly, CPU frequency control does have an affect on idle power draw. The results suggest that the ThunderX2 processor does not go into a common idle sleep state when no workload is running. This behavior may be an area for improvement in future processors. The MEMTURBO configuration has slightly higher idle power (5-10%) at the same OS frequency due to the memory clock running at a fixed 2.3 GHz.

D. Performance

Table II shows the absolute performance obtained for the STREAM TRIAD benchmark at each configuration tested. There is a significant difference in behavior observed between MEMTURBO and CPUTURBO for this workload. With MEMTURBO, memory bandwidth is mostly unaffected by CPU frequency, while CPUTURBO is significantly impacted until around 2.2 GHz. This behavior closely matches the expected behavior shown in Figure 5, and suggests that memory bandwidth sensitive applications should prefer MEMTURBO.

Tables III and IV give the absolute performance obtained for the HPL and HPCG workloads, respectively. In this case, the difference between the turbo configurations is less pronounced but still noticeable. Unlike with STREAM, CPUTURBO does lead to higher performance for these workloads at frequencies above the 2.0 GHz baseline. For HPL, up to 11% higher performance is observed, while for HPCG approximately 7% higher performance is observed. These results suggest that, depending on power and thermal results, enabling CPUTURBO may be desirable for these workloads.

Finally, Table V shows the total runtime of the SPARC application. At the baseline 2.0 GHz, MEMTURBO is approximately 4% faster, likely due to the lower memory frequency of CPUTURBO at this operating point. However, at higher CPU frequencies, CPUTURBO provides up to 9% higher performance. This result suggests that SPARC is sensitive to both memory bandwidth and compute performance.

E. Power Usage

Figure 6 shows how power usage scales for each of the workloads, normalized to the performance of MEMTURBO 2.0 GHz. Here again, the STREAM workload demonstrates the biggest difference in behavior between the two turbo configurations. The other workloads scale similarly to one another, with the increase in performance provided by CPUTURBO requiring additional power consumption. For the SPARC application, achieving 9% higher performance at 2.5 GHz requires an average power draw that is 25% higher. This tradeoff may be considered acceptable as long as temperatures do not rise significantly enough to result in thermal throttling for large-scale runs.

F. Thermal Behavior

Finally, Figure 7 plots the package-level temperature measurements gathered by tx2mon for each workloads run at each configuration. It takes approximately three minutes for thermal conditions to stabilize, and there appears to be some overshoot while fans and water flows ramp up.

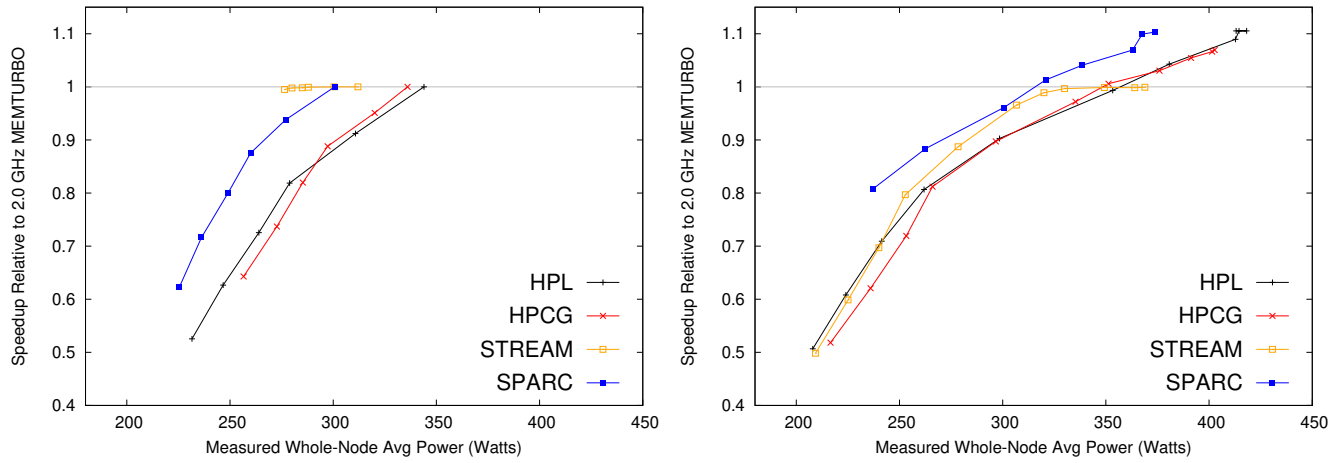
In general, the CPUTURBO configuration is observed to result in significantly higher temperatures compared to MEMTURBO. For the CPUTURBO 2.5 GHz configuration, the workloads result in 9% to 12% higher average temperatures compared to MEMTURBO 2.0 GHz. In absolute terms, this a 6° C to 9° C difference.

TABLE II
STREAM PERFORMANCE UNDER OS CPU FREQUENCY CONTROL

OS Controlled CPU Frequency (GHz)	STREAM TRIAD GB/s MEMTURBO	CPUTURBO
1.0	246	123
1.2	246	148
1.4	246	172
1.6	247	197
1.8	247	219
2.0 (baseline)	247	238
2.1	–	244
2.2	–	246
2.3	–	246
2.4	–	246
2.5	–	247

TABLE III
HPL PERFORMANCE UNDER OS CPU FREQUENCY CONTROL

OS Controlled CPU Frequency (GHz)	HPL GFLOPS MEMTURBO	CPUTURBO
1.0	394	380
1.2	470	456
1.4	544	532
1.6	614	605
1.8	684	677
2.0 (baseline)	750	745
2.1	–	782
2.2	–	817
2.3	–	829
2.4	–	829
2.5	–	829



(a) MEMTURBO (b) CPUTURBO

Fig. 6. Controlling CPU frequency to characterize Power vs. Performance. Each point represents a run of the specified workload at a different static CPU frequency control setting. The right-most point on each curve represents the highest possible setting, 2.0 GHz for MEMTURBO and 2.5 GHz for CPUTURBO.

When Astra was initially deployed, the thermal margin was measured to be approximately 1° C; however, this has since been improved significantly, possibly to as high as 10° C. A node-level temperature increase of 9° C may be sustainable, but it is borderline. These results suggest that further experimentation will be needed to determine if the increased temperatures observed for CPUTURBO can be sustained at full-system scales.

VI. Related Work

Yokoyama et al. provide a wide survey of early research on Arm processors in HPC, including Systems on Chip (SoC) and Scalable Vector Extensions (SVE) [23]. Puzovic et al.'s early energy efficiency study of Arm for HPC used the first-generation ThunderX processor [24]. Hammond et al. benchmark the second-generation (ThunderX2) processor's memory and compute performance against Intel Xeon processors [25]. Jackson et. al compare 32 nodes of ThunderX2 processors to similarly-sized systems of other architectures on a diverse set of application workloads [26]. McIntosh-Smith et al. describe a series of experiments with ThunderX2 in a Cray early-access system [27] and later a production Cray XC50 system [28]. Calore et al. measure power using *tx2mon* for computations running on a single ThunderX2 node to calculate energy efficiency of Lattice Boltzmann and Lattice Quantum ChromoDynamics applications, but their experiments do not vary the frequency of the processor [29]. Kodama et al. evaluate the power control mechanisms of the Fujitsu A64FX processor used in Fugaku for the DGEMM and STREAM microbenchmarks [30]. A dynamic scheduler was developed for non-HPC ARM CPUs for the Montblanc project [31]. Similarly, application performance with a power and energy study was done on a early prototype ARM system, also from the Montblanc project [32].

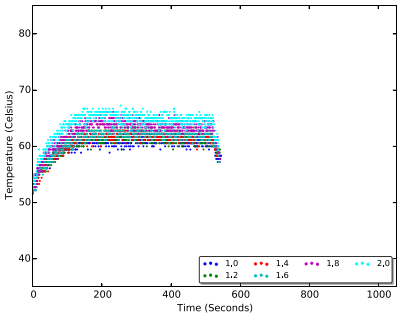
VII. Conclusions

This paper has described the enablement of power measurement and control on the Astra supercomputer. An overview of the platform's capabilities in these areas has been provided along with a discussion of how they have evolved over time and been critical to diagnosing and resolving system-wide thermal issues. Results from an empirical investigation of Astra's power control mechanisms were presented, characterizing the performance, power usage, and thermal impacts on several workloads including the Sandia SPARC application.

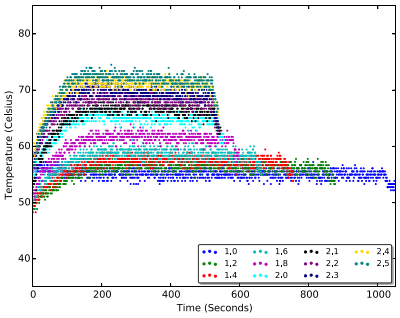
A key question of the evaluation was whether the Astra system could be configured to operate in the CPUTURBO configuration without causing thermal limits to be exceeded. Results were mixed. For SPARC, an observed 9% performance improvement was accompanied by 25% higher node-level power usage and 8° C higher processor package temperature. While the performance improvement is attractive, the temperature increase is close to the system's thermal margin

TABLE IV
HPCG PERFORMANCE UNDER OS CPU FREQUENCY CONTROL

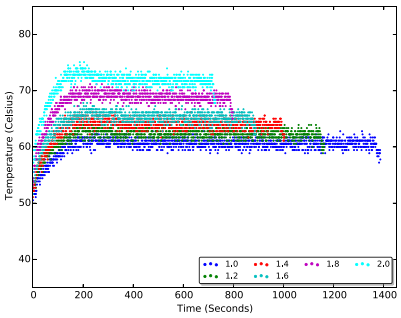
OS Controlled CPU Frequency (GHz)	HPCG GFLOPS MEMTURBO	HPCG GFLOPS CPUTURBO
1.0	25.3	20.4
1.2	29.0	24.4
1.4	32.3	28.3
1.6	35.0	32.0
1.8	37.5	35.3
2.0 (baseline)	39.4	38.3
2.1	–	39.6
2.2	–	40.6
2.3	–	41.5
2.4	–	42.0
2.5	–	42.1



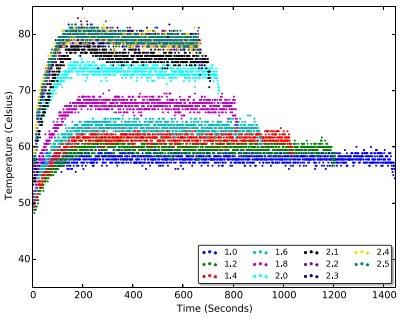
(a) STREAM MEMTURBO



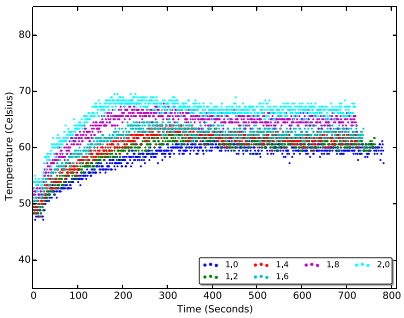
(b) STREAM CPUTURBO



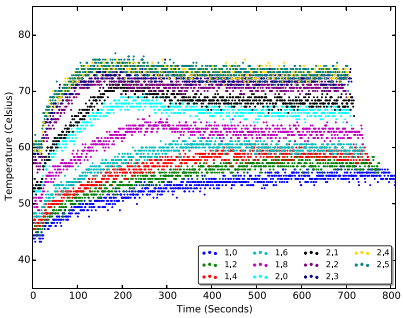
(c) HPL MEMTURBO



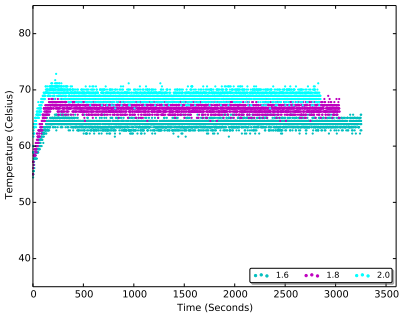
(d) HPL CPUTURBO



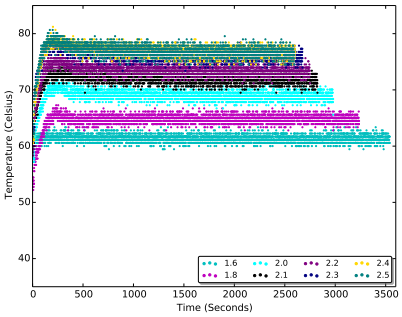
(e) HPCG MEMTURBO



(f) HPCG CPUTURBO



(g) SPARC MEMTURBO



(h) SPARC CPUTURBO

Fig. 7. Time vs. Temperature plots for workloads running on Astra configured for MEMTURBO (left column) or CPUTURBO (right column). Each plot shows multiple runs of the same workload, with each run represented as a unique color run with the OS CPU frequency control setting given in the key, in GHz. This allows visual analysis of both the total execution of the run, shown on the X-axis, and the time-varying thermal behavior, shown on the Y-axis.

TABLE V
SPARC PERFORMANCE UNDER OS CPU FREQUENCY CONTROL

OS Controlled CPU Frequency (GHz)	SPARC Runtime (s) MEMTURBO	CPUTURBO
1.6	3255	3530
1.8	3039	3226
2.0 (baseline)	2849	2965
2.1	–	2813
2.2	–	2738
2.3	–	2665
2.4	–	2592
2.5	–	2583

and may not be sustainable. Future work will entail large-scale experiments on Astra to further characterize the optimal operating settings and determine the viability of using the CPUTURBO configuration for sustained production usage. While the findings described herein may be specific to Astra and similar architectures, we believe they outline a new basis for the analysis and optimization of thermal-limited HPC systems, which may become commonplace in the exascale computing era.

ACKNOWLEDGEMENTS

Vanguard is intentionally a program rooted in collaboration between the NNSA and selected technology providers. We are grateful for all of the support and collaboration we have received in the deployment of Astra from HPE, Marvell, Arm and Mellanox.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

REFERENCES

[1] "Arm Server Base System Architecture (SBSA), Version 6.0," Arm Limited, Platform Design Document DEN0029D, 2020.

[2] "Arm Server Base Boot Requirements (SBBR), Version 1.2," Arm Limited, Platform Design Document DEN0044E, 2020.

[3] K. Pedretti, A. J. Younge, S. D. Hammond, J. H. Laros, M. L. Curry, M. J. Aguilar, R. J. Hoekstra, and R. Brightwell, "Chronicles of Astra: Challenges and Lessons from the First Petascale Arm Supercomputer," in *SC20: Proc. Int. Conference for High Performance Computing, Networking, Storage and Analysis*, November 2020.

[4] M. Sato *et al.*, "Co-Design for A64FX Manycore Processor and Fugaku," in *SC20: Proc. Int. Conference for High Performance Computing, Networking, Storage and Analysis*, November 2020.

[5] J. Dongarra, H. Meuer, and E. Strohmaier, "Top 500 Supercomputers," Website, November 2019. [Online]. Available: https://www.top500.org/

[6] "tx2mon," https://github.com/jchandra-cavm/tx2mon.git.

[7] R. E. Grant, M. Levenhagen, S. L. Olivier, D. DeBonis, K. T. Pedretti, and J. H. Laros III, "Standardizing power monitoring and control at exascale," *Computer*, vol. 49, no. 10, pp. 38–46, 2016.

[8] "PowerAPI Reference Implementation," https://github.com/pwrapi/pwrapi-ref.

[9] *Monitoring and Managing Power Consumption on the Cray XC System*, https://pubs.cray.com/bundle/Monitoring_and_Managing_Power_Consumption_on_the_Cray_XC_System_S-0043/page/User_Access_to_Power_Management_Data.html, Cray Inc., September 2015.

[10] J. D. McCalpin, "Memory Bandwidth and Machine Balance in Current High Performance Computers," in *IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter*, December 1995.

[11] J. J. Dongarra, "The LINPACK Benchmark: An Explanation," in *Proc. Int. Conference on Supercomputing (ICS)*, 1987.

[12] M. A. Heroux and J. Dongarra, "Toward a New Metric for Ranking High Performance Computing Systems," Sandia National Laboratories, Tech. Rep. SAND2013-4744, 2013.

[13] J. Dongarra, M. A. Heroux, and P. Luszczek, "High-performance conjugate-gradient benchmark: A new metric for ranking high-performance computing systems," *Int. Journal of High Performance Computing Applications*, vol. 30, no. 1, pp. 3–10, 2016.

[14] M. Howard, A. Bradley, S. W. Bova, J. Overfelt, R. Wagnild, D. Dinzl, M. Hoemmen, and A. Klinvex, "Towards Performance Portability in a Compressible CFD Code," in *Proc. 23rd AIAA Computational Fluid Dynamics Conference*, 2017.

[15] J. Smith, D. W. Kuntz, S. J. Beresh, and K. M. Casper, "Aerosciences Research at Sandia National Labs," May 2016.

[16] D. W. Doerfler, "Trinity: Next-Generation Supercomputer for the ASC Program," in *HPC User Forum*, April 2014.

[17] S. S. Vazhkudai *et al.*, "The Design, Deployment, and Evaluation of the CORAL Pre-Exascale Systems," in *International Conference for High Performance Computing, Networking, Storage and Analysis (SC18)*, 2018, pp. 661–672.

[18] J. H. Laros, III, "Vanguard: Sandia's Advanced Architecture Technology Prototype Program," *NNSA/ASC Seminar Series 2019*, January 2019.

[19] J. Laros, K. Pedretti, S. Hammond, M. Aguilar, M. Curry, R. Grant, R. Hoekstra, R. Klundt, S. Monk, J. Ogden, S. Olivier, R. Scott, H. Ward, and A. Younge, "FY18 L2 Milestone 8759 Report: Vanguard Astra and ATSE – an ARM-based Advanced Architecture Prototype System and Software Environment," Sandia National Laboratories, NM., Tech. Rep. SAND2018-9999, September 2018.

[20] M. Howard and T. C. Fisher, "SPARC Deep Dive," *Presentation to the JOWOG-34 Applied Computer Sciences Meeting (2019)*, January 2019.

[21] R. M. Wagnild, N. Bitter, J. A. Fike, and M. Howard, "Direct Numerical Simulation of Hypersonic Turbulent Boundary Layer Flow using SPARC: Initial Evaluation," Sandia National Laboratories, NM., Tech. Rep. SAND2019-11158, September 2019.

[22] M. MacLean, T. Wadhams, M. Holden, and H. Johnson, "Ground Test Studies of the HIFiRE-1 Transition Experiment Part 2: Computational Analysis," *Journal of Spacecraft and Rockets*, vol. 45, no. 6, pp. 1149–1164, 2008.

[23] D. Yokoyama, B. Schulze, F. Borges, and G. Mc Evoy, "The Survey on ARM Processors for HPC," *The Journal of Supercomputing*, vol. 75, no. 10, pp. 7003–7036, 2019.

[24] M. Puzović, S. Manne, S. GalOn, and M. Ono, "Quantifying energy use in dense shared memory hpc node," in *2016 4th International Workshop on Energy Efficient Supercomputing (E2SC)*, Nov 2016, pp. 16–23.

[25] S. D. Hammond, C. Hughes, M. J. Levenhagen, C. T. Vaughan, A. J. Younge, B. Schwaller, M. J. Aguilar, K. Pedretti, and J. H. Laros, "Evaluating the Marvell ThunderX2 Server Processor for HPC Workloads," in *The 6th Special Session on High-Performance Computing Benchmarking and Optimization (HPBench'19)*, July 2019.

[26] A. Jackson, A. Turner, M. Weiland, N. Johnson, O. Perks, and M. Parsons, "Evaluating the arm ecosystem for high performance computing," in *Proceedings of the Platform for Advanced Scientific Computing Conference*, ser. PASC '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3324989.3325722

[27] S. McIntosh-Smith, J. Price, T. Deakin, and A. Poenaru, "Comparative benchmarking of the first generation of hpc-optimised arm processors on isambard," in *Cray User Group*, 5 2018.

[28] S. McIntosh-Smith, J. Price, A. Poenaru, and T. Deakin, "Scaling Results From the First Generation of Arm-based Supercomputers," in *CUG 2019 proceedings*, 5 2019.

[29] E. Calore, A. Gabbana, S. F. Schifano, and R. Tripiccione, "ThunderX2 Performance and Energy-Efficiency for HPC Workloads," *Computation*, vol. 8, no. 1, 2020.

[30] Y. Kodama, T. Odajima, E. Arima, and M. Sato, "Evaluation of Power Management Control on the Supercomputer Fugaku," in *Energy Efficient HPC State of the Practice Workshop (EE HPC SOP 2020)*, September 2020.

[31] D. Rajagopal, D. Tafani, Y. Georgiou, D. Glesser, and M. Ott, "A novel approach for job scheduling optimizations under power cap for arm and intel hpc systems," in *2017 IEEE 24th International Conference on High Performance Computing (HiPC)*. IEEE, 2017, pp. 142–151.

[32] E. Francesquini, M. Castro, P. H. Penna, F. Dupros, H. C. Freitas, P. O. Navaux, and J.-F. Méhaut, "On the Energy Efficiency and Performance of Irregular Application Executions on Multicore, NUMA and Manycore Platforms," *Journal of Parallel and Distributed Computing*, vol. 76, pp. 32–48, 2015.