STEP East Coast Town Hall Report

A Sustainable Tools Ecosystem Project Event

Held at IBM's T.J. Watson Research Center in Yorktown Heights, NY

June 6-7, 2023

Recognition & Acknowledgements

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research program under Award Number DE-SC-0002844, program managers Hal Finkel and William Spotz.

Special thanks are due to IBM and East Coast Town Hall host, José Moreira, for generously providing facilities, resources and time for this event.

Finally, the writing team who created this report are appreciated and listed here.

Terry Jones, STEP PI

TABLE OF CONTENTS

1. Executive Summary & Key Findings

The Sustainable Tools Ecosystem Project (STEP) brings together a diverse community of High Performance Computing (HPC) tools developers and stakeholders to develop plans for the sustainability of the HPC tools ecosystem. We define HPC tools as the collection of tools and utilities for analyzing and optimizing application performance, identifying correctness problems, and debugging. These tools interact with hardware features, compilers, communication libraries, programming model runtime systems, and operating systems capabilities that support HPC Tools, as well as the many applications that use these tools. We define the HPC tools ecosystem as the broader ecosystem encompassing the collection of stakeholders, platform dependencies, and interactions that influence those tools. Given the nature of HPC tool development, it requires extensive hardware and application interaction and understanding, and it must adapt to evolving technology. The utilization of tools can greatly enhance the performance and effectiveness of supercomputers in advancing scientific discovery.

STEP is organizing a series of three town hall meetings during the summer of 2023. The main objective of these meetings is to develop a strategic action plan for DOE/ASCR, outlining a recommended approach to ensure the long-term sustainability of an HPC tools ecosystem. Participants will collaboratively explore the current HPC tools space and develop solutions to the sustainability challenges.

The first (East Coast) town hall was held on June 6–7, 2023, at IBM's T.J. Watson Research Center in Yorktown Heights, NY. It brought together 44 stakeholders from diverse communities including tool developers, vendors, facility operators, and application developers [Appendix 1.3 Workshop Attendees]. This town hall aimed to address a wide range of topics related to the first three tools sustainability challenges outlined in our proposal [ref]: exploding hardware complexity, exploding use cases, and coordination.

Expert plenary talks and panel discussions were organized to provide a comprehensive understanding of the topics at hand. This was followed by breakout room discussions, where participants were randomly assigned and had the opportunity to delve deeper into sustainability challenges and explore actionable solutions.

Based on the discussions, several key findings and recommendations emerged as recurring themes:

- **Finding:** Sustainability activities often require multi-year engagement (e.g., participation in standards development or vendor codesign with long product pipelines). At the same time, they also require agile responsiveness to unforeseen hardware and application trends.
    - **Recommendation:** The STEP center must facilitate both long-running, sustained engagement as well as timely responses to unanticipated needs.
- **Finding:** Point-to-point communication between stakeholders (e.g., a vendor, application team, or facility communicating with a single tool maintainer) hinders the sustainability of the community by fragmenting knowledge and limiting collective leverage.
    - **Recommendation:** The STEP center should establish continuous communication channels (options include alliance organizations, recurring meetings, and public forums) with the explicit goal of gathering and formulating requirements, engaging diverse expertise, and exchanging information for the benefit of the entire ecosystem. This may require new strategies to address intellectual property (IP) and non-disclosure agreement (NDA) concerns.
    - **Recommendation:** Establish a collaborative platform that is independent of any one facility to foster the exchange of best practices, mini-apps, case studies, and profiling data among various deployment sites.
- **Finding:** Application teams face numerous challenges that may prevent them from adopting and using tools effectively, including the need to prioritize certain inputs, a lack of awareness of pertinent tools, and the absence of sustainable communication channels with the tools community. We must also be mindful of the distinction between tool developers and tool users, two sub-communities with different objectives.
    - **Recommendation:** The STEP center should actively promote application engagement and communication by implementing various measures. A range of possible actions were identified, such as providing funding for ongoing engagement, employing organizational methods like curating test cases and application kernels, and leveraging technology methods such as AI-assisted guidance and lowering barriers to entry for tool usage.
    - **Recommendation:** Implement community-driven methods to incentivize the use of tools, such as offering compute hours for studies and manpower for assistance. Additionally, focus on improving on-ramps for new tool usage by streamlining the onboarding process and providing immediately actionable feedback.
    - **Recommendation:** Subsequent town halls should incorporate more targeted sessions that specifically seek feedback from the application community.
- **Finding:** The rapid pace of hardware innovation poses a looming challenge to the long-term sustainability of tools. The maturity of performance tools varies widely as applications and architectures increasingly rely on GPUs, accelerators, and other specialized technologies to

continue performance scaling, and from specialized accelerators to dispersed specialized accelerators within the architecture.

    - Recommendation: The STEP center should actively engage vendors, facilities, applications, hardware designers, and tool developers to help establish desired requirements, guidelines, testing methods, and early system access. These collaborative efforts will benefit all parties involved, fostering a culture of proactive collaboration and continuous improvement.
    - Recommendation: We should seek to collaborate with other communities that share similar concerns, including cloud and data center providers, as well as AI vendors, to combine our efforts and maximize our impact
    - Recommendation: The community should strive to establish clear correlations between root causes of problems, workflow patterns, and performance analysis within the broader context of network, storage, power, and other system-level metrics. New heterogeneity raises load-balancing issues with more sophisticated tradeoffs. It is crucial to engage early with vendor and application teams to initiate a dialogue on how we can achieve these objectives.

- Finding: Attendees highlighted numerous challenges related to interoperability, including difficulty in applying diverse tools in multi-step workflows, incompatible file formats, and redundant implementations of similar capabilities across different tools. It was noted that relying solely on formal standards is not a cure-all solution, and, in fact, can hinder productivity.
    - Recommendation: Produce and broadly adapt a tool portability layer (or layers) that enables tool developers to share common functionality, drawing inspiration from successful examples from other communities such as LLVM and Kokkos. Additionally, establish community guidelines to promote interoperability among stakeholders.
    - Recommendations: Develop methods to effectively document and streamline tool dependencies, accounting for subtle interactions such as required build options for underlying libraries, implicit dependencies, and performance dependencies. Utilize existing tools like Spack and E4S, where appropriate, to leverage their capabilities in this regard.

The subsequent sections of this document provide a detailed account of the outcomes from each plenary, breakout, and group discussion. Section 3 highlights the next steps based on these findings, along with key insights on enhancing the effectiveness of the upcoming two town halls in the series.

2. East Coast Town Hall Format & Content

2.1 Introduction

The STEP Town Halls topics are designed to follow a "hybrid" format, combining *iterative refinement* and *different broad themes/objectives*. The intent of each Town Hall is to cover defined broad themes/objectives in their entirety. For example, the East Coast Town Hall focused on challenges related to exploding hardware complexity, increasing diversity of use cases, and coordination among stakeholders, while the Midwest Town Hall will focus on challenges related to project management and identifying existing tool capabilities and users. Additionally, topics will be revisited in subsequent Town Halls to ensure that stakeholders who may have missed a previous one still have an opportunity to provide input on all the topics.

Each Town Hall consists of a variety of sessions that seek to clarify the important challenges and urgent gaps in the HPC tools ecosystem and generate concrete actions to move this area forward in the near and long-term future.

Breakout sessions explore critical challenges and opportunities in building a sustainable tools ecosystem relevant to application software. Dedicated session leads will facilitate productive conversations that yield actionable outcomes. The key discussion topics and proposed actions are documented, and the STEP team provides a comprehensive summary of the results obtained from the *tool survey* and *tool sustainability* sessions. Our intent for these town hall reports is to provide actionable tasks for immediate implementation as well as long-term strategies for achieving broader and sustained impact. We are designing the town halls to focus on a subset of the sustainability challenges, with some topics appearing in at least two of the three town halls to ensure coverage and diversity of viewpoints.

Collaborative efforts often face obstacles in engaging a core constituency, which leads to limited progress. In contrast, the STEP effort involves support from a substantial number of the HPC tool and national labs communities, demonstrating a shared recognition of the need to address these sustainability challenges. Further, many collaborative efforts lose momentum when internal priorities outweigh the benefits of collaboration. Our approach increases the collaborative gain by bringing together communities to collectively address challenges caused by their dependencies. As part of the objectives addressing the coordination challenge, we are committed to establishing mechanisms for ongoing communications among these communities even after the Town Halls have concluded.

Participation in the Town Halls is by invitation. Invitations were sent to senior HPC professionals selected from several categories, including: (a) tool developers; (b) facilities staff; (c) HPC vendors; (d) application developers; (e) those knowledgeable in diversity, equity and inclusion concerns.

2.2 Plenary Portion

Plenary Session 1: Exploding Hardware challenge

John Mellor-Crummey gave a detailed presentation on various challenges posed by the changing hardware landscape.

Even among emerging exascale systems, there is quite a bit of diversity. Emerging exacale systems include Frontier – a GPU-accelerated system based on AMD CPUs and GPUs, Aurora – a GPU-accelerated system based on Intel CPUs and GPUs, El Capitan – a system based on AMD APUs, NVIDIA's Helios – a GPU-accelerated system with 1024 Grace Hopper modules that pair an ARM-based multicore CPU with an NVIDIA GPU, and the Cerebras Andromeda – a cluster formed of 16 Cerebras WS-2 systems, each with a WSE-2 wafer-scale processor. Each of these architectures will perform more than 10^{18} operations per second.

In the CPU design space, heterogeneity and complexity are increasing. Heterogeneous cores (energy-efficient single-threaded cores + performance multithreaded cores) are becoming increasingly common. Memory subsystems now include multiple levels of cache backed by on-package memory, off-package memory, or a mixture of both. Processors often include on-chip accelerators for SIMD computation, data movement, compression and cryptography, among others. In addition, several entities are building custom RISC-V-based processors with a variety of capabilities.

In the GPU space, there are new powerful discrete GPUs from AMD, Intel, and NVIDIA. GPUs from different vendors are rather different. NVIDIA GPUs use a SIMT-based organization. AMD GPUs execute a mix of scalar and SIMD vector operations. Intel GPUs also use SIMD instructions, with scalar instructions operating in a single SIMD lane. The memory hierarchies in GPUs from different vendors are both different and complex.

AMD's forthcoming MI300 APU will have a unified memory space and offer an unparalleled level of integration. However, it is a unique architecture with unique monitoring capabilities.

There are a wide range of special-purpose chips in the AI space, principally for ML training: the Habana Gaudi2 Training Processor, the Cerebras Wafer Scale Engine, the Sambanova Datascale, and Groq's GroqChip. These chips differ significantly from one another and also differ significantly from both CPUs and GPUs.

Today's systems use a profusion of interconnects ranging from PCIe-5, NVIDIA's NVLINK, AMD's Infinity Fabric, Intel's Ultra Path Interconnect to Cray Slingshot11 and Mellanox Infiniband. Communication on these interconnects includes one-sided explicit copies initiated by CPUs or GPUs, implicit copies caused by page faults between devices, as well as two-sided messaging and RDMA.

Across this broad landscape of architectures, tools want to know where an application spends its time, what resources it has at its disposal, is the program using the available resources well, if not why not, and where opportunities for improvement exist.

Mellor-Crummey offered some observations about today's emerging architectures and support for performance measurement and analysis:

- It is one thing to design a high performance chip; it is another to support measurement and analysis of performance losses.
- Since hardware support for measurement must be added at design time, hardware measurement capabilities typically must be planned years before an architecture will be delivered.
- Hardware counters are often available, plentiful, poorly documented, and difficult to interpret. Top-down models are necessary for application developers to readily understand performance losses using counters.
- Microprocessors by AMD, Intel, and IBM all offer support for precise sampling-based measurement at the instruction level
- Instruction-level measurement of GPUs is slowly emerging but its support is much more primitive and stall information is difficult to interpret
- On spatial AI architectures (e.g. Sambanova), performance instrumentation will need to be injected by the compiler.

Altogether, the wide range of compute engines, memory subsystems, and interconnects pose a daunting challenge for tools to measure, attribute, and interpret utilization and inefficiency, as well as identify opportunities for improvement. In most cases, performance analysis and bottleneck diagnosis typically require complex, architecture-dependent strategies. On many of the architectures, details about the architecture and its monitoring capabilities are closely held by vendors. In such cases, the only access to performance measurement capabilities will come through vendor libraries. Having vendors on the critical path for measurement APIs has proven to be a significant challenge.

In summary, for a wide range of reasons, building performance analysis tools for the spectrum of architectures of interest to the DOE is becoming increasingly difficult.

Plenary Session 2: Exploding use case challenge

The HPC tools community is currently facing an explosion of use cases, including: emerging applications in AI and ML, software and tools that integrate AI/ML into traditional programming models, dynamic resource provisioning and management, new and more complex workflow patterns, the expansion of HPC into more diverse problem domains including empirical support and experiment steering, and finally the growth of new provision possibilities such as cloud and commercial data center computing. The panel *Perspectives on Exploding Use Cases* was held to set the stage and give context on the challenges the tools community faces regarding the shifting landscape of applications and use cases.

Three panelists participated in this panel:

1) Kerstin Kleese Van Dam, Director of the Computational Science Initiative at Brookhaven National Laboratory
2) Amedeo Perazzo, Director of the Controls & Data Systems Division at the SLAC National Accelerator Laboratory
3) Sam Reeve, Computational Materials Scientist at Oak Ridge National Lab.

Each panelist prepared a short presentation and took questions regarding gaps and opportunities for the tools community related to the exploding use case challenge. The key findings and points of agreement are listed below:

1) Application workflows are becoming increasingly complex, and often rely on a variety of methods, programming models, languages, middleware, and machine architectures.
2) New tools from the programming language, compiler, runtime, systems, performance, and debugging communities are needed to address the exploding use case challenge and facilitate the scheduling and division of resources among tasks in diverse workflows.
3) There is an opportunity to integrate AI/ML into existing tools to address problems related to workflow analysis (i.e., understanding of application behavior) and resource management.
4) There is a need for better training and communication for application teams to use tools effectively. Fully automated approaches (e.g., dynamic resource management) are welcome if they can be shown to be effective.
5) One panelist noted that even relatively simple updates to task scheduling (e.g., by monitoring I/O utilization from each task and incorporating this information into scheduling) could improve workflow performance and efficiency.

Plenary Session 3: Coordination challenge

There is currently limited interoperability between the tools community and key stakeholders, including vendors and facility operators. There is also no ongoing forum for promoting coordination among applications, tools, systems, and platforms. This Coordination Challenge limits the adoption and reuse of tools, and ultimately curbs the benefits they provide.

Plenary Session 4: Plenary Discussion with Sli.do Polling

Terry Jones and Phil Carns held a plenary discussion which both (a) initiated a public discussion on the management challenges facing the tools ecosystem and (b) evaluated the sli.do web-based polling tool as a means to conduct real time audience driven topic steering in an auditorium setting.

The discussion determined that sli.do was effective in drawing out participation from a broad range of the audience. Sli.do demonstrated the ability to pull out comments from participants that are reluctant to jump into the fray when a minority of people take a passionate stance; we consider that very beneficial. Further, sli.do's real time voting proved an effective way to steer discussion toward topics felt more urgent by the audience as a whole.

The exchanges also provided guidance which STEP will incorporate into the design of the Midwest Town Hall. The audience migrated to the following topics as key:

- What other groups to prepare to reach out to (beyond initial bootstrapping from DOE/ASCR)
- Workforce sustainability efforts
- DEI efforts

2.3 Breakout Portion

This report section provides summaries for each breakout. The breakouts were grouped into 3 topic-based sessions – each session was subdivided into four facets of the session topic:

Breakouts Session 1: The Exploding Hardware Challenge

1. Support Obstacles (session lead: Devesh Tiwari, Northeastern University)
3. Coverage (session lead: Kshitij Doshi, Intel Corp)
4. Vendor Engagement (session lead: Heike Jagode, University of Tennessee)
5. Event Correlation (session lead: John Mellor-Crummey, Rice University)

Breakouts Session 2: The Exploding Use Case Challenge

1. Tool Outcomes (Session lead: Ann Gentile, Sandia National Laboratories)
2. New Capabilities (Session lead: Jim Brandt, Sandia National Laboratories)
3. User Interaction (Session lead: Sameer Shende, University of Oregon)
4. Gaps (Session lead: James Custer, Hewlett Packard Enterprise)

Breakouts Session 3: The Coordination Challenge

1. Convention Standards (Session lead: Tim Haines, University of Wisconsin)
2. Deployment (Session lead: Matt Legendre, Lawrence Livermore National Laboratory)
3. Cross Facility (Session lead: Jose Moreira, IBM)
5. Dependencies (Session lead: Mike Jantz, University of Tennessee)

In the summaries below we provide a synopsis of the breakouts along with their key takeaways and findings.

SESSION 1: Exploding hardware complexity challenge

Breakout 1.1: Support Obstacles

This breakout focused on supporting new hardware and platforms: The group considered two questions: (a) What are the biggest obstacles to supporting new hardware or platforms; (b) What mechanisms would be most helpful to address those obstacles from a community sustainability perspective.

For obstacles to supporting new hardware, several items were noted: the "reactive cycle" resulting from tools developers frequently only getting access and hands-on experience after the machine has been accepted at a facility. A second issue was identified as the time constraints for getting new machines online: often facilities are pushing to bring the machine to a *general availability* state as quickly as possible in an "all-hands-on-deck" style that makes normal coordination with key facilities and vendor personnel difficult. A third issue brought up for discussion was the lack of a coordinated forum for communication among stakeholders, resulting in fragmented availability of information. Finally challenges associated with the familiar human traits of denial and resistance to change were noted here.

For what mechanisms would be most helpful to address those obstacles from a community sustainability perspective, the subgroup identified three examples of possible mechanisms including (a) vendor->community briefings; (b) early and ongoing test access; (c) open communication channels. Several existing methods were noted as useful and perhaps could be amended to be even more effective. These existing methods include (a) training in that end users get frustrated if it's hard

to get up and running. So focused on getting users information and not what to do; (b) hackathons have shown success.

The subgroup also discussed the challenge of *what about when HPC is a small niche to the vendor*. It was noted by the NVIDIA representative that, with the emergence of Data Centers, the HPC market sector is once again competitive with the gaming market sector for many product spaces including GPUs. In instances where HPC remains insignificant, it was the sense of the subgroup that special measures from DOE like amending RFPs to include needed features and/or support may be required.

Breakout 1.2: Hardware coverage

The purpose of this breakout room discussion was to collect information, identify challenges, and identify possible actionable solutions to how to achieve tools sustainability in terms of hardware coverage. The discussion can be can be categorized into the following themes:

Biggest hardware coverage gaps: The breakout group discussed a variety of perceived gaps according to time frame. The area of greatest immediate concern is *GPU tool support*; GPU tooling is not perceived to be as capable as conventional CPU tooling for performance optimization at this point in time, despite the wide-spread reliance on GPU for computational capability on machines being deployed today. In the near-term horizon (in less than five years), we anticipate greater accelerator hardware specialization and availability of FPGA devices to exacerbate this challenge. In the longer-term horizon (greater than five years) we anticipate proliferation of specialized accelerators throughout the system architecture and along the data path to be the next frontier of hardware coverage. There are additional challenges in instrumentation of network, storage, and conventional CPU hardware, but accelerators pose the greatest unknown for sustainability.

In addition to specific hardware technology gaps, the breakout group identified two cross-cutting hardware coverage concerns: power management across devices and capabilities, and load balancing across heterogeneous capabilities. These challenges will become increasingly important across the tools community in future years.

How to be more agile in hardware support: The breakout group attendees were in agreement that hardware specialization is outpacing the ability of the tools community to react. We identified three potential actionable paths forward to address this challenge. The first is availability of contingency funding to address urgent, unforeseen needs. However, this mechanism must have clearly defined, transparent policies to govern its use. Secondly, the HPC community at large would benefit from an explicit mechanism to identify the need for HPC tooling work as part of funded proposals (either as

an FOA response appendix or some other method). At present it is difficult to support engineering work needed to reach a science objective. Thirdly, greater transparency in vendor communication would make it easier for vendors and open-source tool developers to work together without fear of duplicate effort or conflicting goals.

User concerns in identifying appropriate tools for their hardware: The rapid expansion of hardware diversity presents a challenge for users in how to find the "right tool for the job" as they execute codes on new platforms. This becomes an even greater concern for scientific workflows, which explicitly use more than one hardware resource in their pipeline. This is a difficult challenge without clear-cut actionable solutions. More coherent top-down documentation, perhaps with machine-aided guidance could perhaps help but is difficult to maintain. Generally more sharing among the tools community could be helpful as well. Examples in other fields include "leaderboard" style competitions for well-documented use cases [TODO: cite examples] and open sharing of profiling data from diverse hardware [TODO: cite examples] to expand access for researchers who want to experiment with new tool concepts. This presents a social challenge more than a technical challenge due to concerns about reputational impact in data sharing and tool competition.

Recommendations for tool sustainability in the context of hardware complexity: The breakout group identified a set of ideas for how to facilitate tool sustainability across emerging hardware at the conclusion of the discussion. Ideas put forward included the following:

- Contingency funding: Availability of contingency funding, with transparent, well defined governance policies, would help the community react more quickly to emerging hardware.
- Standardizing subsets of counters: it is agreed that broad standardization of hardware counters is impractical. However, more standards on subsets of high-level informative hardware counters would provide an early onramp for tool support while still permitting vendor innovation.
- Considering alternatives to counters for hardware instrumentation: identifying methods for instrumentation (for example, a visual X-ray like display of hot spots and bottlenecks) that are still informative without revealing proprietary hardware information.
- Availability of "mini apps for tooling": mini apps have been successful in platform design and acceptance testing; similar mini apps oriented towards demonstrating tooling are likely to also be effective.
- Early access to hardware, in any form, is always welcome, though it is understood that this is a logistical challenge that requires close coordination between vendors and developers.
- Open-source hardware designs: open-source hardware, as in the RISC V community [TODO: cite], may open the door for more rapid tool support. However, the vendors must still play a key role in developing tooling even in fully open source designs.

- FOA mandates that support HPC tools: Identifying tool engineering needs to support a scientific activity or mandates to show how tools helped advance the productivity over time, would be a welcome way to share responsibility for tooling effort.
- Community coordination to ensure coverage: greater community interaction in any form (top down documentation, leader-board style open competitions, sharing of profiling data) would make it easier for users to find the right tool for their hardware problem.

Breakout 1.3: Vendor Engagement

The Vendor Engagement breakout session aimed to gather insights, identify challenges, and explore practical and constructive strategies to ensure the sustainability of HPC tools. The session focused on two significant aspects: exploring how vendors can contribute to tool sustainability and determining how tools can support vendors in this initiative. Subsequently, for each of these overarching questions, a thorough assessment of more specific and detailed aspects was conducted.

When discussing vendors, it is crucial to include hardware designers as well. Since performance is a critical aspect deeply rooted at the lowest level of the hardware, it is imperative for performance tools to be actively involved in the early stages of the hardware development design cycle.

Vendors can contribute to the sustainability of tools by considering the following aspects:

1. To enhance vendor engagement, it is recommended to consider increased buy-in and acceptance of input from tools:

As part of a collaborative understanding, vendors should take the opportunity to proactively disclose their forthcoming vendor-specific software releases to tool developers in advance. An even more advantageous arrangement would involve granting access to pre-release software. Such access would empower the tools community to promptly implement support for new features, APIs, and other essential components, conduct thorough early-stage testing, and provide timely feedback to the vendors prior to the official software releases.

Furthermore, vendors could offer access to pre-release hardware for tool developers. This access would enable them to develop performance monitoring capabilities tailored to the unique characteristics of novel hardware. It would also facilitate early-stage testing, allowing developers to identify and address any potential issues, and provide constructive feedback to the vendors before the hardware's official release.

In addition, we need to explore strategies and approaches to effectively convey the importance of open-source tools to vendors and facilitate their understanding of the community's demand for these tools, particularly emphasizing the benefits of portability.

Potential outcomes and benefits to vendors may include:

1. Vendors and facilities can greatly benefit from seeking external feedback from tools to enhance their systems and promote increased safety measures.
2. Relieving the burden on vendors to prepare tools for "acceptance tests.": Relying solely on the passing of "acceptance tests" as the only metric of success may not provide a comprehensive evaluation, particularly if conducted by vendors or facility staff rather than the developers of the specific tool. There is a notable risk that "acceptance tests" may pass without adequately testing the tool's support for specific new hardware components of the system. Therefore, it is important to consider additional metrics and involve the tool developers in the testing process to ensure a thorough assessment of all critical tool features for a particular architecture. To support that, vendors and/or facilities could offer financial assistance to developers to ensure that tools are adequately prepared for “acceptance tests.”
3. Timely availability of tools on new machines: This initiative ensures that tools are promptly accessible on newly released machines.
4. Application teams derive significant advantages from the expeditious availability of tools, enabling them to effectively evaluate and optimize the performance of their codes on these newly introduced machines.

Establishment of a "Vendors-Tools Alliance" website and workshop:

The contribution of software and hardware feedback from multiple tools can lead to a significant influx of emails, potentially containing overlapping and redundant input. Such a situation may pose challenges to effectively managing the communication flow. To tackle this matter, we suggest implementing a recurring workshop-style event, such as the "Vendors-Tools Alliance" workshop, aimed at managing the aforementioned issue. This event would provide a platform for addressing open issues directly and facilitating in-person discussions between technical team members from vendors and tool developers. To optimize the efficiency of the process, the implementation of a dedicated "Vendors-Tools Alliance" website can serve as a centralized hub for collecting and consolidating issues and feedback from various tools. This platform would encompass a wide range of submissions, including (but not limited to) links to problem reports submitted to vendor software repositories, concerns related to hardware features, as well as any disruptive elements identified within software and/or hardware components.

2. Vendor adoption of key tools:

Vendor adoption of key tools raises essential questions regarding its implications. Such as:

What does "adoption" precisely entail, and what does it not imply?

- Does it mean vendors commit to actively contributing to a tool?
- Does it involve taking ownership of the tool or specific parts of it?
- Does it require vendors to create pull requests (PRs) to incorporate updates reflecting changes in their APIs?
- Does it encompass vendors developing support for new vendor features, be it in software or hardware?

Is it necessary to define the criteria for determining which tools are considered "key tools"?:

- The responsibility of building a list of critical infrastructure software should ideally involve collaboration among multiple stakeholders, including the open-source community, the Department of Energy (DOE), vendors, and facilities. Each of these stakeholders brings valuable perspectives and expertise to the table.

Additionally, the viability of vendor adoption may vary based on the architecture of each tool. Lastly, it is crucial to ascertain whether vendor adoption is desirable from the perspective of tool developers.

3. Non-Recurring Engineering Efforts in Support of Tools:

An alternative approach to the more aggressive concept of "vendor adoption" is the idea of Non-Recurring Engineering (NRE) efforts in support of tools. The feasibility of this approach depends on the specific structural design of the tool in question. Tools could potentially offer an abstraction layer, allowing vendors to concentrate on the code components that pertain to their vendor-specific support. This approach would eliminate the need for vendors to redesign or fully comprehend other aspects of the tool's codebase. An example of successful implementation can be observed in PAPI, where its component design enables such abstraction. However, the viability of this method varies across different tools and heavily depends on their current architectural design. Furthermore, the desirability of pursuing NRE efforts may differ among various tools. For instance, in the case of PAPI, the "PAPI framework and its independent components" have been intentionally designed to enable anyone from the community to develop their own components. These components have the option to be integrated into the official PAPI release, which prompts the consideration of maintenance responsibilities. Alternatively, they can be utilized exclusively by the original developers and their

corresponding user bases. HPE, for example, has released specific PAPI components solely within their HPE-PAPI version.

Tools can take the following actions to assist vendors in providing sustained support for the tools:

As part of a collaborative understanding, it is advantageous for tool developers to engage in proactive communication with vendors prior to releasing vendor-specific support within their toolsets. This approach gives vendors an opportunity to conduct testing and provide feedback.

Furthermore, tools can capitalize on the recommendations mentioned earlier in the section regarding how vendors can enhance their acceptance of input from tools. These recommendations include the following considerations:
Tools can leverage their access to pre-released vendor software, allowing for early adoption, rigorous testing, and the ability to provide feedback prior to official releases. Furthermore, tools can utilize their access to pre-release hardware to develop robust performance monitoring capabilities tailored to specific new hardware features, perform comprehensive testing, and report helpful insights back to vendors. It is crucial to emphasize that, as a code of practice, it is essential that all feedback and input provided adhere to the principles of being Constructive, Concise, and Current (C^3).

Breakout 1.4: Event Correlation

Full-featured tools for correctness, debugging, and performance require that program activity be correlated back to the program source code.

- Correctness tools need to be able to attribute problems to calling contexts, functions, source lines, and program variables accessed in these contexts.
- Debugging tools need to be able to correlate program behaviors with code positions and values of program variables.
- Performance analysis tools need to correlate program behaviors to source code contexts including calling contexts, functions, and source lines. Performance tools that provide feedback about data usage patterns need to associate them with program variables. Performance tools also need to be able to correlate performance information with resources so that a performance analyst can assess the utilization of resources.

Key kinds of correlations include correlating metrics with source code positions. The simplest is correlating a metric, such as a dynamic instruction count, with a source code position. Data-oriented

metrics can be correlated with multiple source code positions, such as data declarations, storage allocation points, or code initiating data movement.

The difficulty of correlating program activity with source code positions depends upon the kind of measurement and the measurement methodology.

The simplest case is when a tool collects measurements by making calls to an instrumentation library such as LLNL's Caliper [1]. Such calls can collect many kinds of useful information, e.g. timings, execution rates, hardware counter measurements for an interval of execution, or semantic information such as tensor sizes used. Instrumentation-based measurements are trivial to correlate: they can be mapped to the source code position where instrumentation library calls are made.

Today, Roofline models [9] are extraordinarily popular for assessing the performance of computations on CPUs and GPUs to ascertain (1) whether they are bandwidth bound or compute bound, and (2) how the performance of a computation relates to the relevant performance ceilings associated with available bandwidth or peak operation rates. While coarse-grain Roofline metrics can be enormously useful in understanding how close an application or function is to achieving the relevant peak performance possible, the reason for the gap between achieved performance and peak performance is often difficult to understand because it is not correlated with the operations that cause performance to fall short of the achievable peak. For deeper insight than Roofline models can provide, hardware mechanisms that can monitor and attribute performance metrics at the level of individual operations are needed.

A more difficult challenge for both hardware and tools is to measure and correlate events associated with individual operations. On hardware such as CPUs and GPUs, operations are represented by instructions. On FPGAs or dataflow architectures, operations map directly to functional units and data flow directly to wires and switches.

On CPUs, one can count hardware events that represent work, utilization, inefficiency, or monitor data accesses and then trigger interrupts when a particular event threshold is reached. On modern parallel hardware, where many instructions can be in flight at once, sophisticated hardware support is needed to measure and attribute metrics to instructions in flight. There are several well developed mechanisms for measuring and accurately attributing events on CPUs that support out-of-order instruction execution, including IBM's marked instructions [11], AMD's instruction-based sampling [12], Intel's precise event-based sampling [4], and Intel's load latency facility [13].

On GPUs, hardware mechanisms for fine-grain performance measurement are less well developed. While NVIDIA GPUs provide hardware support for PC sampling [10] as will AMD's MI300. These

samples occur at regular intervals rather than being triggered by specific events of interest such as cache misses. As a result, GPU architectures have under-developed mechanisms for correlating events with their root causes. As a result, GPUs attribute stalls where they are observed rather than to their root causes. For instance, when a load causes a stall, the stall is attributed to the instruction that waits for the load to complete rather than the load itself. Another weakness in GPU capabilities for event correlation is that they cannot precisely identify from what level in the memory hierarchy a load was satisfied. Having more sophisticated hardware capabilities for correlating events to their root causes would (a) make it easier for tools to provide more accurate attribution of events to code and data and (b) simplify the process for doing so.

On non-instruction-based architectures such as Field Programmable Gate Arrays (FPGAs) and dataflow architectures, monitoring will be in the form of functional units added to a hardware configuration and data movement to record measurements made by these functional units. Correlating measurements made by functional units added to a hardware configuration to monitor its operation must be done by associating the measurements by such counters with the source code position associated with the operations that the counters are counting.

For instruction-based architectures, mappings from instructions and data values to source code positions are typically encoded in DWARF - a debugging information file format [2]. To support understandable correlation of events to source code positions for optimized code arising from inlined functions or templates, compilers need to associate instructions with their full inlined call chains. Members of the breakout group believe that DWARF lacks mechanisms for associating operations mapped to functional units and data movement mapped to switches back to source code positions. There may be a need for developing new mechanisms to support correlation of events in such architectures back to source code positions.

In the breakout group, there was little discussion of mechanisms for associating network and I/O performance back to source code positions. Some kinds of semantic information, e.g. communication or I/O event counts and bytes transferred, back to source code positions using instrumentation-based measurement. However, there is likely a need for better mechanisms for measuring and correlating quantitative metrics of network and I/O performance such as operation latency, resource utilization, and congestion utilization and congestion as well as I/O metrics such as request latency, bandwidth utilization with network and I/O operations.

Besides correlating hardware events with source code positions, the breakout group noted that today architectures often provide a daunting number of counters. For instance, one can count 382 distinct events on the AMD MI250X and 12,335 events for HPE/Cray's Cassini network interface chip for Slingshot-11 [8]. Often, there is a glaring lack of information about how to use the available counters

to obtain insight into application performance. On CPUs, hierarchical cycle accounting [7], IBM's CPI stack model [5] and Intel's top-down model [6] explain how to use counters to obtain insight at multiple levels of detail. Such hierarchical models are essential guides for how to use hardware counters to gain insight into program performance.

A conclusion of the breakout group is that the status quo is not working very well. Today, vendor offerings in response to procurements often fall short of meeting the needs for both tool developers and end users in the area of hardware and software support for measuring and attributing performance metrics. When new architectures emerge, (1) they often lack necessary hardware and software mechanisms for monitoring hardware events and correlating them with source code positions, and (2) they often lack sufficient documentation, such as top-down models, that inform how to use the available hardware mechanisms for measurement and event correlation. A key contributing cause of this is a failure of procurements to identify what kinds of measurements would be desirable and what kinds of correlations are needed sufficiently early in the procurement process. Often, subject matter experts in the performance tools domain aren't involved until procurements are finalized and NRE contracts are in place; at that point it is too late to ask for something that hasn't already been offered.

To improve this situation and provide a hardware and software ecosystem that makes it possible and easier to construct high-quality tools with useful event correlation, the breakout group came up with several recommendations.

1. Looking forward, it would be much better for subject matter experts to identify best practices and gaps for measurement and event correlation independently of procurements, publish a white paper describing needs, so that vendors can work to meet these needs when developing next-generation technologies before ever offering them in a procurement. When a vendor responds to a particular procurement, it is typically too late to add new hardware mechanisms for measurement and event correlation to a proposed architecture. Hardware development is typically a 5-year process; thus the time to effect changes in hardware that will be offered in response to a call for bids for a procurement is to have hardware designers begin to develop and incorporate support for desired mechanisms before the DOE publishes a procurement.
2. Since the HPC community is only one of the relevant stakeholders that care about performance (e.g. cloud providers, companies that employ massive data centers to provide services such as Google and Facebook, as well providers of AI training and inference appliances), the breakout group recommends joining forces with these other community stakeholders to

    a. identify common needs and communicate them to technology developers of hardware and software so that tools can measure important performance metrics and correlate them to source code positions so that application and framework developers can obtain insights about the root causes of work, inefficiency, utilization, and data access patterns in their codes, and
    b. Identify gaps where research into new methods is needed to improve the ability of hardware to measure and correlate key metrics with program source code.
3. Procurements can then reference an existing white paper that can provide a detailed wish list for measurement and analysis capabilities needed to support tools and their rationale. This should provide a much better mechanism than having a procurement simply asking a vendor to identify how they propose to address the need for performance insights with insufficient guidance about what would be desirable or particularly responsive to community needs.

SESSION 2: Exploding Use Case challenge

Breakout 2.1: Use case complexity: Tool Outcomes

This breakout focused on what the tools and tool ecosystem provide. In this first town hall, the terminology of "tool outcomes" was used, to distinguish that concept from "capabilities" or "functionalities", but it was agreed that new terminology should be developed to appropriately capture the intent.

The "exploding use case" sustainability challenge refers to the situation that as architectures and computing paradigms increase, there is an increasing demand for tooling and increasing dependencies that cannot be met by current tool development practices. To address this demand, then, this breakout focused on identifying possibilities in the four following areas:

1. Most **important "outcomes"** needed from the tools ecosystem to support the current and evolving computing infrastructures
2. New and increased **design requirements** that are placed on the tool ecosystem as a result of these needed "outcomes" and evolving compute infrastructure
3. Positive, constructive, **actionable paths forward for sustainability of the tools ecosystem** that transcend any single software project or use case as a result of new and increased "outcomes" and design requirements
4. How a STEP Center could further these paths forward as part of a **compelling and timely follow-on Community Sustainability Proposal**

Highlights of the breakout findings and additional notes are given below.

1. New and Increasing Tool Ecosystem "Outcomes":
Many of the long-held visions of the tools community continue as goals. Overall, the tools ecosystem seeks to increase efficiency, performance and throughput through approaches such as autotuning, dynamic resource management decisions and actions, and diagnostic insights for both human-in-the-loop and automated response.

The breakout noted that no single "outcome", tool, or approach can meet the needs of all cases. Necessary "outcomes" may have tradeoffs in accuracy or fidelity of resultant insights vs the required latency and processing involved. For example, it may be more important to get a quick, partial answer at the edge vs needing full insight into the complex and time-dependent interplay of I/O, network, storage, and compute resource state and utilization.

2. New and Increased Design Requirements:
Insights into the considerations in the increase in computing architectures and paradigms have been discussed in the earlier, architecture breakouts, and we avoid repeating them here.
Instead, we choose to highlight the complexity due to the increase in the "user base" and "interaction base" of tools. While output for human-in-the-loop (e.g., application users, application developers, and system administrators) processing continues to be a valid target, tool "outcomes" are increasingly meant to target other tools, system software and middleware for interaction and response. A flexible, aggregate tool ecosystem, as opposed to single tools, provides the "outcomes" necessary to increase efficiency, performance, and throughput. No single tool can meet the needs; multiple variants and implementations of functionalities are required to meet the tradeoffs mentioned earlier.

As a result, there is an increase in the complexity of output formats (e.g., human ingestible vs machine ingestible) and in choice of data/analysis presentation (e.g., what a system administrator wants to know vs what an application developer wants to know). There is increased development necessary and knowledge-base required to support the increasing architectural diversity and the expanding execution environment. Increases in compute capability, analytics, and advanced possibilities for tooling (such as siting tools in situ with compute processing and system software and middleware) increase the possibilities for ecosystem success, if the sustainability challenges can be addressed.

3. Actionable Paths Forward for Sustainability:
The breakout proposed two concurrently actionable paths, which vary based on amount of possible commonalities and intrusion into current development paths:

- STEP Tools Portability Layer (STPL) inspired by Performance Portability Layers (e.g., Kokkos, Raja): A STPL would consist of implementations, with well-known interfaces, of underlying needs of tools as architectures and requirements evolve. Multiple concurrent functionalities and implementations would be included in STPL to address requirements and tradeoffs. Tools' outcomes would be exposed consistent with the STPL interfaces. STPL would therefore both facilitate tool development and increase interoperability among tools, so that outcomes can be achieved by combinations of tools. Sustainability is achieved through modular and aggregate development.
- Expert advisors for tool developers: Common advice to developers can increase sustainability by helping to ensure that tool development evolves in similar ways and by reducing the global knowledge needed for development.

We note that there have been efforts exploring development of common components for tools before with mixed success. Key challenges have been in the ability to identify and develop sufficient commonalities to make such an approach worthwhile, while still enabling the necessary specificity and level of detail necessary for the various use cases. In addition, design and implementation are key to meeting performance (e.g., memory footprint, latency, minimizing data processing and data transport) requirements. Success of Performance Portability Libraries in these regards is inspiring, however.

Relatedly, we advise that future breakouts expand on the question of "What does sustainability look like for this topic?" Given the time limitations of the breakout as well as the logical direction of approach (new outcomes->requirements->actions), we believe there may be additional actions worth exploring.

4. Proposed Activities in a STEP Center:

STEP is a seedling project for a longer-term, community sustainability proposal, with an anticipated call of fall 2023.

The breakout believes that STEP could contribute to a compelling follow-on proposal on that timeline in the following ways:

- STEP will commit to the Center developing a STPL
- STEP members will specify what functionalities are needed from a STPL and the requirements. An exploratory prototype could be developed by using existing tool needs to identify obvious high-value functionality. Development of a prototype, overcoming challenges in previous attempts at common components, would increase ASCR and developers' confidence in this approach.

- STEP would commit to changing our tools to adopt a reasonable STPL. Identifying interoperability possibilities enabled by STPL and articulating the gains by existing interoperability would incentive community adoption.

As a note, we advise that while the focus of this breakout was on identifying and overcoming the challenges in Sustainability of the Tool Ecosystem, a compelling proposal should be sure to articulate how increases in efficiency, performance, and throughput enabled by the Tools Ecosystem benefit the user and the ultimate science.

Breakout 2.2: Use case complexity: New Capabilities

The New Capabilities breakout session examined new and emerging capabilities across the HPC landscape, including algorithms, applications, software, and hardware, as well as the challenges and opportunities faced by the tools community with respect to new capabilities. The main points of discussion in this session are described below. In each case, the group also identified specific requirements and concerns that the tools community will need to address to take advantage of the new capabilities.

In addition to multiprocessing and conventional accelerators (e.g., GPUs), there are further opportunities to enhance application performance by exploiting functional elements and capabilities in emerging architectures (e.g., smart NICs, smart switches, storage systems, and accelerators for data compression, encryption, and movement). To leverage these capabilities, the tools community will need to incentivize vendors to a) expose performance related information from these elements as well as the more traditional ones, and b) to provide insight into what the data they expose means and how best to use it.

Some architectures will be able to exploit data flow graphs that are compiled directly into executable files for greater efficiency. However, there is a need for tools that can help guide this process by monitoring and integrating performance measurements into the data flow graph itself.

The information and metrics that can be collected on computing infrastructure (e.g., switches, NICs, CPUs, GPUs, and memory) are exploding and provide unprecedented opportunity for insight into hardware state. However, the community needs vendors to provide insight into what the data they expose means and how best to use it. In many systems, the new data that can be collected is largely undocumented and may not be useful. Without such insight and guidance from vendors, the extra information may only increase the burden on tools to collect and store data with little or no added benefit.

Several emerging technologies, such as smart NICs, smart switches, burst to cloud, and others provide opportunities to offload processing to compute and infrastructure elements other than conventional processing elements. However, current tools cannot easily track execution as it flows through these offload technologies, in many cases. Hence, the tools community needs to extend or create new capabilities that are able to conduct performance analysis of alternative processing elements.

Security needs to be integrated into all aspects of computing and data movement. Raw (unencrypted) data can enable reverse engineering of applications and other problems if left unaddressed. Encryption and role-based access needs to be enabled everywhere (e.g., application data, performance counter data, monitoring data, data transmitted from one computational element to another) not only as it traverses infrastructure but also within compute nodes. Moreover, secure computing scenarios are growing, including in Cloud, and many tools will need to be updated to interoperate with new security features and constraints.

The tools community should explore alignments between DOE needs and Cloud and AI/ML technologies as it searches for new opportunities for improvement in debugging, efficiency, gaining performance insight, and abilities to perform dynamic reconfiguration of workflow elements and resources. ChatGPT has proven to be a great productivity tool in writing and debugging code and may well be able to provide similar insights in the analysis of data from performance and monitoring tools. However, it is important to note that cloud providers can change technologies quickly as they respond to market forces while the DOE mission space and associated codes have significant inertia.

In many cases, teams and institutions create tools as they require functionality and there is little to no (re)use of other tools. There is a need for better infrastructure and processes to encourage tool reuse and maintenance. One potential approach could be to create a “tools library” (low level libraries and utilities that higher level tools can build on) as a “sustainability” step. Maintenance of, and even extension to, a well defined set of functionality and interfaces is more sustainable than maintenance of an ever increasing number of tools that do roughly the same thing. An example of a tool with good reuse and maintainability is PCM, but this tool only operates on Intel architectures.

Some other suggestions targeting a sustainable tools ecosystem that were mentioned during this breakout are listed below:

1. The creation of a common tooling infrastructure would enable better reusability and maintenance. Vendors could still differentiate on performance. Examples of past successes include LLVM (software that has become widely used by vendors) and the UCX consortium.

2. The creation of a common tooling API would enable facilities to leverage performance information at the system level for elastic resource allocation and increasing system utilization. Vendors that target this API would have a better financial argument going into a procurement. Datacenters would also likely benefit from an API where they can maximize their own allocations at a more granular level.
3. This project could potentially fund vendor staff for some development involving IP to sidestep NDA problems. This would be more sustainable than each tool developer having to go through the NDA process with each vendor and each new technology.
4. This project should focus on tools of direct benefit to the users. Vendors invest in tools because it builds value for their users. DOE could fund technologies which build value for the DOE user rather than "tools".
5. There is also an opportunity to provide value to the larger, more diverse community that is currently served by vendor tools. For instance, this project could develop a model where the vendors collaborate with the DOE on building common tool components, (e.g. a reliable x86 stack unwinder).

Breakout 2.3: Use case complexity: User Interactions

This breakout group discussed how we can increase engagement with application teams. Here are some key takeaways. The group focused on two questions:

How can we improve bidirectional interaction between application teams and tool developers?

1. Applications teams have different priorities (Science vs Performance), and do not require performance evaluation studies to be included in the INCITE/ALCC allocation submissions.
2. However, Leadership Computing Facilities (LCFs) do need to report on scaling data and reasonable use of on-node resources
3. There are ways to incentivize the use of performance evaluation tools (e.g., NERSC introduced a free day – where user allocations were not charged for runs with Darshan).
4. We can improve tool usability with upfront analysis of performance data (e.g., time spent waiting at barriers within collective operations, recommendations).
5. Using a collection of results and issues (e.g., with Slack or a database of issues to sort through) can help.
6. The applications teams could also have someone else run their codes and evaluate performance.
7. Interaction with catalysts, performance tool experts can help improve usage of performance evaluation tools and engagement with the tool developers.

What is the right vehicle to make this sustainable across more than just pair-wise discussions between a single application team and a single tool team?

1. Interactions need to be between groups of people who are performance oriented, and specific application teams. This is similar to the liaison process for INCITE proposals at OLCF/ALCF
2. Hackathons, training workshops can help drive this interaction and are deemed very useful.
3. There is a clear distinction between users and developers of applications. Developers are often unaware of the performance bottlenecks until the users report it. Developers may not have run the code at scale.
4. Key to tool usage is to have simple tools that require no change to source code, build, or even the application binary. Just modifications (with examples) for launching the tool on un-modified binary should help drive the tool usage.
5. We need to make tool usage more intuitive including automating the analysis of collected performance data.
6. We need to bridge the gap between performance data collected and actionable intelligence and present it in a clearly understandable way.

Breakout 2.4: Use case complexity: gaps

This breakout group examined the main gaps in HPC tool support for the variety of use cases that scientific applications require. The primary gaps identified in this session are listed below:

1. There is a lack of documentation and support that make it difficult for novices to understand and use performance tools. As a result, the barrier of entry for users to adopt a new tool is very high.
2. There is a lack of tooling support for many modern languages. Many in the community have begun using C++, which still needs better tool support. For instance, tracking data flow in C/C++ applications is difficult with current tooling infrastructure. There is also demand for tools that support languages and runtimes with additional features, such as Rust, Lua, Go, and Julia.
3. There is a need for better tool support for performance analysis of task-based parallelism, complex workflows, and cloud-based applications.
4. There are gaps in reporting and visualizations provided by tools. Most tools create custom output that might be difficult for new users to understand and make actionable.
5. There is a lack of coordination between vendors and the open source tools community. Earlier access to vendor-specific interfaces and software libraries with functions to interpret performance data could enhance open source tool development.
6. The community currently lacks tooling support for quantum computing simulations.

A few specific solutions and practices were also discussed as promising approaches to address these gaps.

1. Some tools provide annotations in their output (e.g., “time printing”) and many users have found this feature helpful for understanding tool output and gaining insight from tools.
2. Developing tools that use a common interface or “portability layer” can help address differences across vendors.
3. Better communication among vendors and tool developers could reduce redundant implementations and enable more effective tool solutions.
4. Comprehensive documentation of common tools and their capabilities (with examples) could help new users and students understand the current tool landscape and how to collect important information and conduct analyses with modern tools.

SESSION 3: Coordination challenge

Breakout 3.1: Coordination Challenge: Conventions and Standards

At the highest level, we consider the two key elements of interoperability between tools to be the ability to 1) pass data between them without the user needing to explicitly be involved and 2) allow two or more tools to be executed on a single program simultaneously without interference. Data-passing can manifest as the kind of producer/consumer model commonly seen in pipeline workflows where data is moved “down” the pipeline via a common file format. We label this category of interfaces as loosely coupled since the consumer does not strictly require knowledge of the upstream application- only the format of the data. Conversely, a tightly coupled interface is one in which that knowledge must be present. The most common example of this type of interface is consuming data in a machine-dependent manner by compiling and linking against a third-party application. In both cases, there must be a contract between the two tools as to the structure and content of the data.

It is tempting to assert that both types of interfaces should be regulated by an authoritative standard (or standards) so that all tools can interoperate seamlessly. However, we know from experience that such standards do not always produce interoperable tools. A notable example in the HPC space is the incompatibility between some implementations of the MPI standard. It has also become more difficult over time to define even the most basic of terminology in some fields. For example, the meaning of FLOPs must now consider the size of the data type being used (e.g., f8 versus f64). It is also well known that hardware-level tooling provided by HPC vendors suffer from poor documentation and unstable interfaces and are generally difficult to use correctly- particularly in the earliest stages of their development. Getting vendors to unify their interfaces based on a dictated standard would require immense force to change their momentum.

Even outside of the vendors, HPC technologies move forward at breakneck speeds. New concepts and properties are created frequently for software, hardware, and their interactions. The manual process of creating standards for them is very labor-intensive and not scalable. Differing use cases (e.g., was the API developed to access data only when there is an issue? Was it developed to change/control things? Was it developed to be used to access a single counter vs collecting timestamped counters synchronously) drive heterogeneity so high as to make encapsulating it in a single standard a very high bar to achieve.

On the second point of interoperability, users want to run multiple tools simultaneously on a single execution of their program. In this case, a single tool can no longer expect to be the only actor (e.g., some tools write into a limited set of buffers and overwrite other writer's output). The number of interactions grows much faster than any standardization process could possibly keep up with. Yet, these interactions are important at every level of tooling. For example, a tool developer may need to debug an error in their own tool by running both their tool and a memory analysis tool like valgrind. Such interactions could also be across domains of concern with a GPU performance tool being used simultaneously with a file I/O tool.

To this end, we suggest the tools community not engage in constructing rigid standards which stumble over the issues above. Rather, the community should cooperatively construct a set of guidelines which illuminate the desired types of information tools need from vendors in order to produce actionable information to users. This kind of sharing of deeply technical knowledge is an excellent use for the vendor engagement activities the co-design center for STEP could cultivate. Importantly, the guidelines must not be so broad as to have no meaning, but not so narrow as to only apply to a single hardware vendor.

Challenges and proposed solutions:

- Proprietary issues with early access to hardware. Tool developers cannot publish their capabilities until the vendors release the runtime/hardware details.
    - Multi-way NDAs are available, but are hard to effectively utilize. No known solution at this time.
- Cost and time for implementation of the guidelines
    - This can be alleviated by parallelizing over the community.
- Different data for different components (i.e., is there a rational consistent subset of data we need?)

    - This is the central tenant of the guidelines. We need to aggregate the cross-cutting needs from the lowest details for tool developers to the science-oriented questions of the experimentalists.
- HPC evolves very quickly
    - Standards committees are too slow. An agile, community-driven approach can be effective in weeks/months rather than years.
- Hardware vendors are not easily swayed
    - By forming a unified voice across many aspects of the tools community, it may be possible to bring larger bargaining power when it comes to getting vendors to implement desired features.
- Who will be the guiders of the guidelines?
    - TBD. There could be a (volunteer?) group who edits an official document.
- Where will the document live?
    - The doc source could live in a public source-controlled repository (e.g., GitHub), and the official docs could be published on the ascr-step.org site.

Breakout 3.2: Coordination Challenge: Deployment

The deployment breakout group discussed challenges and strategies around deploying tools at facilities. It focused on what the tool community could do to ease facility deployments, requirements from facilities, and how users at facilities access deployed tools. The group developed recommendations around testing for tools, mechanisms for tool deployments, and roles where organizations like STEP could contribute to easier deployment.

Tools are particularly challenging to deploy. Unlike applications, they lack many of the standards that allow one to abstract away hardware. Deployments must frequently be customized for specific systems. Within a system, a tool deployment may have to support multiple software toolchains. Frequently tools need to be installed in a cross-product for each MPI, Compiler, and GPU Driver version that users may require. And within a facility there may be multiple system types each with their own hardware and toolchains.

These challenges make tool deployments an immense amount of work. Expecting tool developers to support deployments across many facilities without dedicated resources is unreasonable. One possible role for a tools software sustainability effort could involve organizing and supporting tool deployments at facilities. Facilities could provide information about systems, needed deployments, and accounts. The software sustainability effort could provide resources and coordination to tools teams who deploy software.

Spack can significantly ease the effort involved in deploying tools across many facilities and system types, but it does not completely eliminate the challenges. Tools can have detailed dependencies into hardware resources, schedulers, facilities policies, networks, system software versioning, and other low-level details. Spack does not always model systems at these level. Software deployments with Spack may still require an expert to map between systems and the necessary spack variants and flags. Another role for a tools software sustainability organization may be in extending spack with features and models needed by tools. Spack could also be extended to allow users to request tool deployments that are guaranteed to work with spack-deployed applications, or allow instrumentation-based tools to instrument applications during spack build time.

(TODO: Write about tool testing for validating tool deployments.)

Breakout 3.3: Coordination Challenge: Facility Coordination

The Coordination Challenge workgroup on Cross-Facility Collaboration met on June 7th with the goals of (1) identifying the major hurdles in having effective cross-facility collaboration and (2) recommending the best approaches to achieve that effective collaboration.

In addition to specific recommendations we make, we believe that a long term project on software sustainability should continuously strive to develop and follow a collection of best practices for cross-facility collaboration. Our mission here is to establish an initial set of good practices, which will then evolve throughout the project.

A major hurdle identified by the workgroup is the complexity and diversity of non-disclosure agreements (NDA) imposed by the various vendors working with the facilities. The vendors will each have their own NDA requirement and will often require a separate and different NDA for each party collaborating on a specific topic. For example, Party X may have an NDA with vendor Y to work on MPI. And yet, the same vendor Y may require a new NDA from the same party X if there is an updated MPI requirement. Things can get even more complicated when multiple vendors are present in the same contract with a facility, as is common today.

Our recommendation is to establish a common NDA framework that all vendors have to agree to and will facilitate collaboration across parties. A party that signs the NDA for a project is then entitled to collaborate with other parties in that project. Vendors must not impose additional NDAs beyond the common one.

Another hurdle identified is that quite often the use cases that are worth sharing across facilities and among the various parties are those use cases that proved problematic. That is, they caused machine

crashes, or had low performance, or were very difficult to deploy. There is a natural reluctance of people and institutions to expose the “dirty laundry”, but the problematic use cases are precisely the ones that can be most helpful to other facilities and parties involved.

Our recommendation is to establish an independent group (not affiliated with a specific facility) to experiment with different use cases and report all aspects of that experience. Both good outcomes and difficulties should be reported.

We must also develop processes and guidelines for what is useful to share among the various facilities. One recommended approach is to standardize on a set of use cases, represented by the DOE Mini-apps. By focusing on the Mini-apps, we can create a knowledge base with results from experiences on multiple facilities. Because of the commonality and pervasiveness of those experiences, they are likely to be relevant to multiple parties.

Once a large collection of results are collected (for example, from running a variety of Mini-apps use cases in various facilities) it starts to get difficult to find relevant information in the collection. There are a variety of new technologies that can be used to facilitate that process. In particular, we recommend an experiment in using an Open Source large language model that can be hosted by the project. The large language model would continuously ingest data produced by the various facilities and would serve as a knowledge base that could be queried by the participants.

We recommend the use of two standard tools for collaboration and communication among multiple parties in different facilities. The first is Slack, which is an on-line communication tool. It can be configured with multiple channels for specific interests and keeps a record of the conversations. Contents from the conversations can be ingested by the large language model mentioned above.

The other recommended standard tool is github. It supports sharing of large files and keeps a history of changes and conversations. Data from github can also be ingested in the large language model.

As projects evolve, the size of shared information can grow very large, easily to multiple terabytes. We can use existing services such as Zenodo to share large data sets across facilities.

Even in the most collaborative environment there is still need and value in both anonymizing data and providing access control. For example, keeping use cases anonymous may avoid some bias in running/not running them. Correspondingly, the ability to keep some results private until properly curated may encourage more people to publish those results. We need to develop a mechanism for anonymizing data but still share experiences in a useful way. We recommend considering the approach adopted by anonymous.4open.science.

Breakout 3.4: Coordination Challenge: Dependencies

The dependencies breakout focused on identifying concerns and issues regarding dependencies among tools and their users, as well as potential solutions for these problems.

The concerns and issues regarding dependencies / interoperability that were discussed are as follows:

1. Some participants noted that many tools employ the same API to interface with an application. For example, multiple tools might dynamically link against the same application using the LD_PRELOAD environment variable. If there are conflicts between how the tools operate and how the application expects them to operate, this can lead to crashes and undefined behavior. Sometimes the presence or severity of these issues depends on the order with which tools are linked into the application. In many cases, these issues can be resolved, but there are cases where fixes are undocumented or difficult to coordinate.
2. Some tools require custom libraries, compiled with the appropriate versions and compilation flags, to work properly. For example, many tools have use cases that map memory addresses back to application source code or logical data objects. For this case, such tools typically rely on system or third-party libraries (e.g., DWARF [2], binutils [3]). However, these libraries must be compiled with the -fPIC flag to execute this capability properly. Since most systems do not provide versions of these libraries with the -fPIC flag enabled, these tools typically require a custom copy of these libraries for the address mapping capability to work properly.
3. The OMPT interface, which provides facilities for using tools with OpenMP, does not support multiple tools connecting to the same interface.
4. Some panelists noted that many tools and applications have performance dependencies. For example, the performance of a tool or application can shift as the library software it uses is updated or changed over time. Others noted that the collective operations used by MPI applications, which have a significant impact on overall performance, depend on environment variables that might not be consistent across different platforms and environments. Another panelist noted that filesystem operations (e.g., stat) can have very different performance depending on the underlying filesystem.
5. Some panelists noted that security policies are another dependence for many tools. As security policies are updated or changed, the space of capabilities that tools can provide are also changed.
6. Other dependencies that were noted by this panel include: BIOS settings, firmware, and environment variables.

The breakout group also discussed potential solutions for the problem of dependencies:

1. Group noted that there are some tools that handle dependencies well. Examples include:
    a. e4s builds a large set of tools with the appropriate versions, matching the full software stack, at whichever centers support it.
    b. SPACK facilitates building and deployment of complex HPC applications by enabling users to specify and automatically build all relevant software dependencies. However, some panelists noted that SPACK is both a blessing and a curse because minor errors or issues with SPACK configuration files could potentially cause unnecessary recompilation of dozens of software packages.
    c. There are also tools for snapshotting system environments, including environment variables, BIOS settings, etc.
2. The general consensus of the panel was that there exist workable solutions for handling dependencies between tools and application software. However, they also noted that having a consistent software stack is very important for ensuring dependency problems do not arise. Additionally, the system environment settings, including environment variables and BIOS options, are also important to record and control. However, the panel noted there are not well known or standard techniques for keeping the system environment consistent.

Additionally, the group briefly discussed whether tools should or should not be allowed to be enabled by default at system centers. Some panelists noted that having some tools enabled by default could provide very valuable information about application behavior, resource utilization, and performance. While there was agreement that this information could potentially be collected with very low (perhaps imperceptible) overhead, others noted that it is still very important for performance engineers to be able to turn off tool operation for some applications and experiments for certain use cases (e.g., debugging).

3. Next Steps

3.1 Propose STEP center initiatives

This first STEP town hall explored a broad range of topics and identified key cross-cutting findings and recommendations as summarized in Section 1. In our next town hall meeting, we will propose a finer-grained breakdown of initiatives within STEP that align with those findings and provide a more concrete framework for discussion.

3.2 Refine town hall processes

The STEP East Coast Town Hall was highly successful in its objectives. As the first of a series of three town halls, it also provided a unique opportunity to solicit feedback so that we can refine and improve our town hall process.

We identified the following logistical improvements to consider for future town halls:

- Allow additional time for breakout reporting and discussion
- Provide printed materials (e.g., schedule, room assignments, facility maps) to in-person attendees
- Provide earlier guidance to attendees
- Expand representation to include additional communities, particularly from debugging tools, cloud providers, and more vendors.
- Assign explicit scribes for breakout sessions and panels

The charge questions for the first town hall were purposefully broad and cross-cutting in order to collect the concerns of the HPC tool ecosystem and potential directions for sustainability. In subsequent town halls, we will refine the breakout topics and charge questions to develop more concrete strategies for the operation of the STEP center. We identified the need for the following topical coverage in particular:

- Refinement and clarification of the scope of the STEP center
- Structure for distinct initiatives within the STEP center
- Models for sustainability
- Technical coverage of power management and debugging tools

3.3 Convene next town hall

The second of three STEP town halls will be held on July 11-12 on the campus of Iowa State University. This town hall will shift discussion focus to more heavily feature the management challenge identified in the original STEP proposal. As in the first town hall, it will assemble a collection of experts from the tools community including not only tools developers but also application, facility, and vendor stakeholders to develop actionable collaborative solutions to the challenge of sustainability in the tools ecosystem.

3.4 Conduct community survey

Attendees of the first town hall identified a clear need for two distinct surveys. We plan to complete both of the following surveys and make them publicly available prior to the next town hall.

- User-facing: assesses which tools are used, usability concerns, how tool expertise is acquired within the user community, what capabilities and overheads are desired, and objectives.
- Tool-facing: assesses shared dependencies, readiness levels, and points of contact.

4. References

1. David Boehme, Todd Gamblin, David Beckingsale, Peer-Timo Bremer, Alfredo Gimenez, Matthew LeGendre, Olga Pearce, and Martin Schulz. 2016. Caliper: performance introspection for HPC software stacks. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '16). IEEE Press, Article 47, 1–11.
2. DWARF Version 5. https://dwarfstd.org/dwarf5std.html.
3. GNU Binutils. https://www.gnu.org/software/binutils.
4. Brinkley Sprunt. Pentium 4 performance-monitoring features. IEEE Micro, 22(4):72-82, 2002.
5. M. Srinivas, B. Sinharoy, R. J. Eickemeyer, R. Raghavan, S. Kunkel, T. Chen, W. Maron, D. Flemming, A. Blanchard, P. Seshadri, J. W. Kellington, A. Mericas, A. E. Petruski, V. R. Indukuru, and S. Reyes. IBM POWER7 performance modeling, verification, and evaluation. IBM Journal of Research and Development, 55(3):4:1-4:19, May 2011.
6. A. Yasim. A top-down method for performance analysis and counters architecture. In IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), pages 35-44, March 2014.
7. A. Nowak, D. Levinthal and W. Zwaenepoel, "Hierarchical cycle accounting: a new method for application performance tuning," *2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, Philadelphia, PA, USA, 2015, pp. 112-123, doi: 10.1109/ISPASS.2015.7095790.
8. Keith Underwood - Cassini: The Slingshot NIC Hardware and Software Enabling Ex
9. ascale. Eighth International Workshop on Communication Architectures for HPC, Big Data, Deep Learning and Clouds at Extreme Scale (Exacomm2023), May 25, 2023. https://youtu.be/yxPrDBSgK1Y.
10. Samuel Williams, Andrew Waterman, and David Patterson. 2009. Roofline: an insightful visual performance model for multicore architectures. Commun. ACM 52, 4 (April 2009), 65–76. https://doi.org/10.1145/1498765.1498785
11. NVIDIA Corporation. CUDA Profiler - Release 12.1. April 14, 2023. https://docs.nvidia.com/cuda/pdf/CUDA_Profiler_Users_Guide.pdf
12. IBM Corporation. Power ISA Version 2.07B. April 9, 2015.
13. Paul J. Drongowski. Instruction-Based Sampling: A New Performance Analysis Technique for AMD Family 10h Processors. November 16, 2007. https://www.amd.com/system/files/TechDocs/AMD_IBS_paper_EN.pdf
14. Intel 64 and IA-32 Architectures Software Developer's Manual. Document Number: 252046-043. June 2014. http://kib.kiev.ua/x86docs/Intel/SDMs/252046-043.pdf.

Appendix 1: Attendees

Appendix 1.1 Workshop Organizers

First Name	Last Name	Affiliation
Jim	Brandt	Sandia National Laboratories
Phil	Carns	Argonne National Laboratoy
James	Custer	Hewlett Packard Enterprise
Kshitij	Doshi	Intel
Ann	Gentile	Sandia National Laboratories
Tim	Haines	University of Wisconsin
Heike	Jagode	University of Tennessee
Mike	Jantz	University of Tennessee
Terry	Jones	Oak Ridge National Laboratory
Matt	Legendre	Los Alamos National Laboratory
Keith	Lowery	Advanced Micro Devices
John	Mellor-Crummey	Rice University
Bart	Miller	University of Wisconsin
José	Moreira	IBM
Erdal	Mutlu	Pacific Northwest National Laboratory
Phil	Roth	Oak Ridge National Laboratory
Sameer	Shende	University of Oregon
Galen	Shipman	Los Alamos National Laboratory
Shane	Snyder	Argonne National Laboratory
Devesh	Tiwari	Northeastern University
Theresa	Windus	Ames National Laboratory

Appendix 1.2 Writing Leads

This report was created at an accelerated pace with the help and diligence of the following people. Each person led a breakout summary and provided information for the Executive Summary and findings. In addition, people listed in **bold font** remained at the event for an additional day to help expedite this writing of the report.

First Name	Last Name	Affiliation
Jim	**Brandt**	**Sandia National Laboratories**
Phil	**Carns**	**Argonne National Laboratory**
James	Custer	Hewlett Packard Enterprise
Kshitij	Doshi	Intel
Ann	Gentile	Sandia National Laboratories
Tim	**Haines**	**University of Wisconsin**
Heike	Jagode	University of Tennessee
Mike	**Jantz**	**University of Tennessee**
Terry	**Jones**	**Oak Ridge National Laboratory**
Matt	Legendre	Los Alamos National Laboratory
John	**Mellor-Crummey**	**Rice University**
José	**Moreira**	**IBM**
Sameer	Shende	University of Oregon

Appendix 1.3 Workshop Attendees

The following list of people physically attended the East Coast Town Hall. This report would not be possible without the time and commitment from this distinguished group of HPC experts and tools ecosystem veterans. Their insights and input affirm that a sustainable tools ecosystem founded on openness and coordination can be realized.

First Name	Last Name	Affiliation
Jean Luca	Bez	Lawrence Berkeley National Laboratory
George	Bosilca	University of Tennessee
Jim	Brandt	Sandia National Laboratories
Tim	Burke	Los Alamos National Laboratory
Phil	Carns	Argonne National Laboratoy
James	Custer	Hewlett Packard Enterprise
Kshitij	Doshi	Intel
Ben	Forget	Massachusetts Institute of Technology
Hubertus	Franke	IBM
Ann	Gentile	Sandia National Laboratories
Jennifer	Green	Los Alamos National Laboratory
Tim	Haines	University of Wisconsin
Clayton	Hughes	Sandia National Laboratories
Heike	Jagode	University of Tennessee
Mike	Jantz	University of Tennessee
Terry	Jones	Oak Ridge National Laboratory
Kirk	Jordan	IBM
Chulwoo	Jung	Brookhaven National Laboratory
Christoph	Junghans	Los Alamos National Laboratory
Kerstin	Kleese Van Dam	Brookhaven National Laboratory
JaeHyuk	Kwack	Argonne National Laboratory

Matt	Legendre	Los Alamos National Laboratory
Ang	Li	Pacific Northwest National Laboratory
Chunhua	Liao	Los Alamos National Laboratory
Wei-Keng	Liao	Northwestern University
John	Linford	NVIDIA
Ray	Loy	Argonne National Laboratory
John	Mellor-Crummey	Rice University
David	Montoya	Trenza Synergy
José	Moreira	IBM
Vitali	Morozov	Argonne National Laboratory
Matthew	Norman	Oak Ridge National Laboratory
Swann	Perarnau	Argonne National Laboratory
Amedeo	Perazzo	SLAC Stanford Linear Accelerator Laboratory
Jonathan	Pietarila Graham	Los Alamos National Laboratory
Sam	Reeve	Oak Ridge National Laboratory
Robert	Rutherford	Pacific Northwest National Laboratory
Aaron	Scheinberg	Jubilee Development
Karl	Schulz	Advanced Micro Devices
Sameer	Shende	University of Oregon
Nathan	Tallent	Pacific Northwest National Laboratory
Devesh	Tiwari	Northeastern University
Rene	Van Oostrum	Advanced Micro Devices
Gwen	Voskuilen	Sandia National Laboratories
Huihuo	Zheng	Argonne National Laboratory

Appendix 2: Agenda

Day 1 (June 6th, 2023)

Time	Topic
9:00 - 9:15	Introduction and Logistics (Terry Jones, ORNL)
9:15 - 10:00	Opening Remarks (Hal Finkel, DOE)
10:00 - 10:30	Break
10:30 - 11:15	Plenary: Perspectives on The Exploding Hardware Problem (John Mellor-Crummey, Rice University)
11:15 - 12:00	Panel: Perspectives on The Exploding Use Case Problem (Moderator: Mike Jantz, Univ. of Tennessee) ● Amadeo Perazzo (SLAC) ● Kerstin Kleese Van Dam (BNL) ● Sam Reeve (ORNL)
12:00 - 12:15	Guidance, Logistics and Desired Outcomes for Breakouts (Terry Jones, ORNL)
12:15 - 1:30	Working Lunch (Provided)
1:30 - 3:00	Breakouts Session 1: The Exploding Hardware Challenge 1. Support Obstacles (session lead: Devesh, Northeastern University) 2. Coverage (session lead: Kshitij Doshi, Intel Corp) 3. Vendor Engagement (session lead: Heike Jagode, University of Tennessee) 4. Event Correlation (session lead: John Mellor-Crummey, Rice University)
3:00 - 3:30	Break
3:30 - 5:00	Breakouts Session 2: The Exploding Use Case Challenge 1. Tool Outcomes (Session lead: Tim Haines, University of Wisconsin) 2. New Capabilities (Session lead: Jim Brandt, Sandia National Laboratories) 3. User Interaction (Session lead: Sameer Shende, University of Oregon) 4. Gaps (Session lead: James Custer, Hewlett Packard Enterprise)
5:00 - 5:30	Closing and Guidance for Day 2

Day 2 (June 7th, 2023)

Time	Topic
8:45 - 8:50	Introduction and Day 2 Logistics (Terry Jones, ORNL)
8:50 - 9:10	Report out summaries from Session 1: The Exploding Hardware Challenge 1. Support Obstacles (session lead: Devesh, Northeastern University) 2. Coverage (session lead: Kshitij Doshi, Intel Corp) 3. Vendor Engagement (session lead: Heike Jagode, University of Tennessee) 4. Event Correlation (session lead: John Mellor-Crummey, Rice University)
9:10 - 9:30	Report out summaries from Session 2: The Exploding Use Case Challenge 1. Tool Outcomes (Session lead: Tim Haines, University of Wisconsin) 2. New Capabilities (Session lead: Jim Brandt, Sandia National Laboratories) 3. User Interaction (Session lead: Sameer Shende, University of Oregon) 4. Gaps (Session lead: James Custer, Hewlett Packard Enterprise)
9:30 - 10:15	Panel: Perspectives on The Coordination Challenge (Moderator: José Moreira, IBM) ● Matthew Legendre (LLNL) ● JaeHyuk Kwack (ANL) ● Dave Montoya (Trenza Synergy)
10:15 - 10:45	Break
10:45 - 12:15	Breakouts Session 3: The Coordination Challenge 1. Convention Standards (Session lead: Tim Haines, University of Wisconsin) 2. Deployment (Session lead: Matt Legendre, Lawrence Livermore National Laboratory) 3. Cross Facility (Session lead: José Moreira, IBM) 4. Dependencies (Session lead: Mike Jantz, University of Tennessee)
12:15 - 1:30	Working Lunch (provided)
1:30 - 3:00	Report out summaries for Session 3: The Coordination Challenge 1. Convention Standards (Session lead: Tim Haines, University of Wisconsin) 2. Deployment (Session lead: Matt Legendre, Lawrence Livermore National Laboratory) 3. Cross Facility (Session lead: José Moreira, IBM) 4. Dependencies (Session lead: Mike Jantz, University of Tennessee)
3:00 - 3:30	Break
3:30 - 4:15	Plenary Discussion with Sli.do Polling (Terry Jones, ORNL & Phil Carns Argonne)
4:15 - 4:30	Closing Remarks / Adjourn (Terry Jones, ORNL)