Achieving Exascale Capabilities through Heterogeneous Computing

This article provides an overview of AMD's vision for exascale computing. The authors envision exascale computing nodes that compose integrated CPUs and GPUs, along with the hardware and software support to enable scientists to effectively run their scientific experiments on an exascale system. The authors discuss the challenges in building a heterogeneous exascale system and describe ongoing research efforts to realize AMD's exascale vision.

Michael J. Schulte
Mike Ignatowski
Gabriel H. Loh
Bradford M. Beckmann
William C. Brantley
Sudhanva Gurumurthi
Nuwan Jayasena
Indrani Paul
Steven K. Reinhardt
Gregory Rodgers
Advanced Micro Devices

Increasingly complex problems have demanded increasingly powerful computer systems. High-performance computing (HPC) has steadily pushed supercomputers to greater computational capabilities, with petascale computing (10^{15} floating-point operations per second [flops]) first being achieved in 2008.[1] The current fastest supercomputer is capable of 33.86 petaflops (see www.top500.org). These HPC systems perform massive computations to drive important scientific experiments leading to impactful discoveries, from designing more efficient fuels and engines to engineering safer bridges and buildings, from modeling global climate phenomena to exploring the origins of the universe.

The next HPC milestone is that of developing an exascale supercomputer that can achieve more than an exaflop (10^{18} or one billion, billion flops) on critical scientific computing applications. Such exascale supercomputers are envisioned to comprise at least 100,000 interconnected servers or nodes, implying that each node has an individual computing capability of greater than 10 teraflops (Tflops) on real applications. Note that a modern high-end discrete GPU today achieves a peak of only about three double-precision teraflops.

The challenges associated with exascale computing, however, extend far beyond merely achieving a certain number of floating-point calculations per second. To feed such high levels of computation, both the system's memory and internode communication bandwidths must increase dramatically beyond current levels. The energy efficiency of the supercomputer must improve by orders of magnitude to enable the operation within practical datacenter power-delivery capabilities of a few tens of megawatts. With more than 100,000 nodes, significant advances in resilience and reliability are also required to keep the overall machine up and running. In this article, we describe AMD Research's vision for exascale computing, and in particular, how we see heterogeneous computing as the path forward.

AMD's vision for exascale computing

An exascale system consists of the hardware implementing the computational resources, as well as the software needed to efficiently write, tune, and execute applications on the hardware.

Published by the IEEE Computer Society

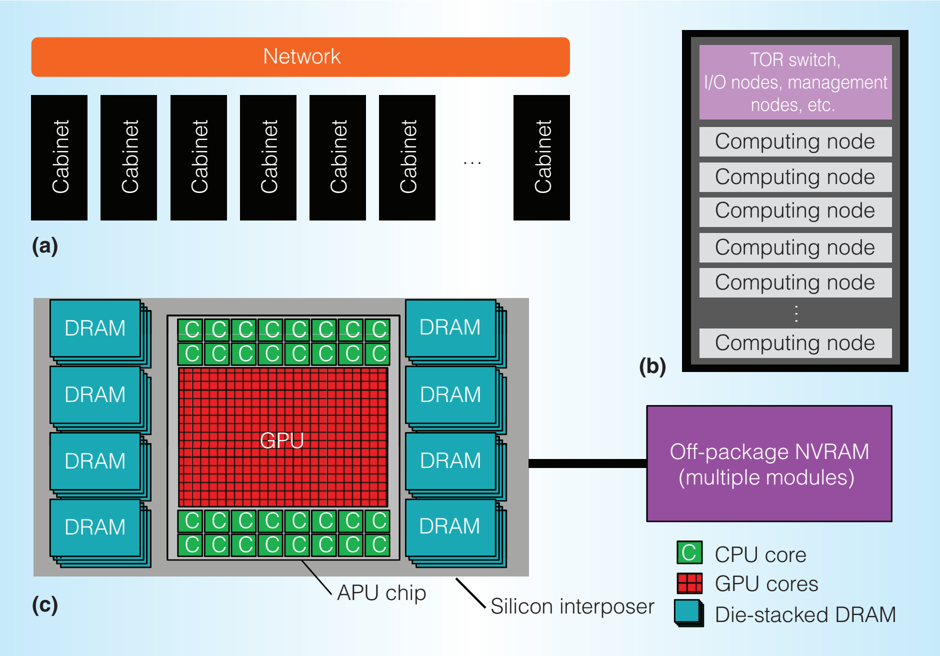

Figure 1. Hierarchy of hardware components in an exascale system. (a) A supercomputer comprises many cabinets that communicate through a network. (b) Each cabinet houses multiple computing nodes, along with additional components (such as the top-of-rack network switch and I/O nodes). (c) An example exascale heterogeneous processor (EHP) within a computing node comprising a silicon-interposer-based APU with multiple stacks of 3D DRAM. The EHP is coupled to a second level of off-package memory.

Figure 1 shows a conceptual depiction of the hierarchy of hardware components in an example exascale system. At the highest level (Figure 1a), the overall machine consists of many cabinets (for example, a couple hundred), all interconnected with a vendor-specified networking technology and topology. Each cabinet (Figure 1b) provides multiple computing nodes providing the computational and memory resources, as well as any other required networking, I/O, and system management facilities. In some systems, I/O and system management facilities are provided by dedicated I/O nodes. In this article, we focus on the architecture of the exascale computing nodes (Figure 1c), because that is where many of the heterogeneous computing concepts are most easily highlighted.

Each node consists of a high-performance APU that combines a balance of high-throughput general-purpose (GPGPU) pipelines for extreme high performance with excellent energy efficiency, coupled with high-performance multicore CPUs targeting single-thread performance. The GPUs provide the high throughput required for exascale levels of computation, whereas the CPU cores handle hard-to-parallelize code sections and provide support for legacy applications. We envision that the APU's heterogeneous computing capabilities will also be fed by a heterogeneous memory architecture. A combination of die-stacked dynamic RAM (DRAM) and high-capacity nonvolatile memory (NVM) deliver high bandwidth and low energy, while providing enough total memory capacity to handle the large problem sizes of mission-critical HPC applications. These computing and memory components connect to the other system nodes via a high-bandwidth, low-overhead network interface controller (NIC). Central to our vision of using heterogeneous computing to achieve effective exascale computing are robust, flexible, and easy-to-use software tools and programming environments. The Heterogeneous System Architecture (HSA) will provide open hardware and software interfaces, as well as the broader ecosystem, to enable HPC application programmers to unlock the computing capabilities of the underlying heterogeneous exascale system. In this article, we detail each primary component of our vision for a heterogeneous exascale system. Our intent is both to explain our vision for how to achieve exascale computing and to bring focus to critical technology areas where further research (for example, from academia and industry) can be of greatest value.

APUs for heterogeneous computing

We envision each node of our heterogeneous exascale system to use a high-performance APU, which we call an *exascale heterogeneous processor* (EHP), designed to achieve the scalability and performance targets for future exascale systems. Exascale computing requires a spectrum of carefully designed solutions working in concert to achieve all the system goals (for example, performance, programmability, energy efficiency, reliability, and scalability). The heterogeneous APU approach provides an effective balance of high-throughput, energy-efficient GPU resources coupled with CPU cores optimized for single-threaded performance.

The CPUs and GPUs must be architected cohesively and synergistically to make them easy to program and use. A critical part of our heterogeneous computing vision is that each EHP fully supports HSA,[2] which provides (among other things) a system architecture where all devices within a node (such as the CPU, GPU, and other accelerators) share a single, unified virtual memory space. This lets programmers write applications in which CPU and GPU code can freely exchange pointers without needing expensive memory transfers over PCI Express (PCIe), reformatting or marshalling of data structures, or complicated device-specific memory allocation. HSA also provides user-level task queues supported by the hardware, wherein any computing unit can generate work for any other unit. For example, a GPU can launch new tasks on the GPU itself, or even back to the CPU, without involving the operating system or complex drivers, whereas in most conventional (non-HSA) GPU-based heterogeneous computing, all control must flow through the CPU, which can lead to significant inefficiencies and harder-to-program code structures.

The EHP integrates significant CPU and GPU computational resources along with large quantities of in-package memory (such as 3D DRAM) to provide the memory bandwidth to feed the computation engines. Figure 1c shows a schematic view of a possible EHP architecture. The bulk of the computational horsepower comes from the GPU; we provision a large number of computing units (CUs) necessary to provide 10 Tflops or more of sustained throughput. A total computational throughput of exactly 1 exaflop can thus be achieved using a system with 100,000 EHP nodes. The integrated 3D DRAM delivers the bulk of the memory bandwidth, although additional off-package memory is required to provide adequate total memory capacity per node.

Alongside the GPU resources, our conceptual EHP design also incorporates several CPU cores. For example, Figure 1c illustrates 32 cores (potentially supporting the x86 or ARM instruction set architectures) that can effectively execute serial portions of applications, non-performance-critical sections, or legacy applications for which programmer effort has not yet been expended to port to the integrated GPU. In addition to the computational units and in-package memory, the overall architecture must also provide a high-bandwidth, low-latency, and energy-efficient interconnection substrate (for example, a network on a chip [NoC]) to connect all of these components.

Beyond the envisioned heterogeneous system we have described, we should consider other candidate system organizations. For example, a node organization comprising only CPU cores (that is, no GPU acceleration) could provide programmability benefits. However, our analysis indicates that a CPU-only solution does not provide a sufficient performance-per-watt advantage to enable the target levels of execution throughput within the desired power budgets. Another approach would be to augment a conventional multicore CPU with external (discrete) GPU cards. We believe that the integrated GPU approach has the following advantages over external GPUs:

- lower overheads (both latency and energy) for communicating between the CPU and GPU for both data movement and launching tasks/kernels,

- easier dynamic power shifting between the CPU and GPU,
- lower overheads for cache coherence and synchronization among the CPU and GPU cache hierarchies that in turn improve programmability, and
- higher flops per m^3 (performance density).

There is much active research on other forms of acceleration, such as approximate computing, reconfigurable FPGAs, and neural net/neuromorphic computing, but thus far, successful usages of these techniques tend to focus on specific applications. For individual domains, these novel approaches may provide excellent performance and energy efficiency, but supercomputers execute a wide range of scientific computing, manufacturing, security, and other applications, and thus the exascale power-performance objectives must be achieved in a non-application-specific manner.

To realize an EHP that can support an exascale-class system, research is required in several areas. The first is in exploring the EHP's viable and effective physical organizations. The design illustrated in Figure 1c is but one possible example. Studies are required to determine the correct mix of interposer-based stacking (that is, 2.5D) and vertical stacking (that is, 3D), as well as to understand the tradeoffs among specific stacked topologies of logic chips, interposers, and memory stacks. At the same time, advances in the NoC are needed to support the distributed communications among the multiple silicon die that comprises the overall chip-stacked EHP. We have started exploring the implications of interposer-based system integration on the NoC's organization[3]; however, much research remains to be done on the design of scalable, multichip NoCs. The aggressive levels of computation being integrated within a single package require high performance, scalability, low energy, and resilience, all while maintaining acceptable engineering and manufacturing costs.

The massive computing resources provided by the GPU CUs and the multiple CPU cores share the EHP's hierarchical coherent memory system. To maintain HSA compatibility, the EHP supports a unified coherent address space and synchronization mechanisms accessible by all computing units within the EHP node. However, achieving such a memory system is beyond the capabilities of current coherence and synchronization mechanisms. Therefore, research is needed to investigate solutions to eliminate unnecessary communication and to localize necessary communication. To this end, we created a novel synchronization mechanism called QuickRelease, which enables GPU memory systems to efficiently support fine-grained, load-acquire, and store-release synchronization among GPU threads without sacrificing throughput.[4] We also defined a set of memory consistency models, called heterogeneous-race-free (HRF) memory models, that provide programmers with a well-defined framework to reason about large on-chip memory systems.[5] Collectively, QuickRelease and HRF reduce the synchronization overhead, but they do not deal with the high overhead of coherence probing. In that regard, we devised heterogeneous system coherence (HSC), which opportunistically manages coherence at a region (for example, 1 Kbyte) granularity, instead of a per-cache block (64 bytes) granularity.[6] Overall, these three techniques greatly streamline and improve the efficiency of the APU's communication. However, further opportunities for improvement still exist, and the computer architecture community should continue research in this domain.

Heterogeneous memory architectures

Exascale levels of computational capabilities are not likely to be useful if the overall system cannot provide enough memory bandwidth to keep the computation engines fed and a large enough memory capacity to tackle problem sizes of interest. Although exascale system requirements can vary from one supercomputer to another, the memory requirements are likely to be extremely challenging. For example, according to the US Department of Energy's objectives and requirements for exascale systems,[7] each node should support a memory bandwidth of 4 Tbytes per second (TBps) and a per-node memory capacity of 1 Tbyte or more. Even aggressive extrapolations of current memory solutions are unlikely to achieve the exascale targets in the specified timeframe. For example, current DRAM DIMM-based memory systems for CPUs typically support single-digit numbers of channels, providing a total node-level memory bandwidth of a few tens of Gbytes per second (GBps)—over 100× off from the exascale targets.

We believe that die-stacked memory within the processor package is required to achieve the exascale bandwidth requirements with the required energy efficiency, but it cannot deliver on the capacity targets. For example, current die-stacked memory bandwidth for the JEDEC high-bandwidth memory (HBM) standard is 128 GBps per DRAM stack. With an estimated eight stacks per EHP package, this provides a bandwidth of 1 TBps using current technology. Some combination of increasing signal rates, bus widths, and channel counts could potentially push the technology to the desired 4 TBps within the exascale timeframe. At the same time, HBM technology could provide 2 Gbytes of capacity in a four-layer stack of 4-Gbit chips. With eight stacks per EHP, this would provide only 16 Gbytes of fast in-package memory, which represents a 64× gap from the target of 1 Tbyte per node. Even with taller 3D stacks and higher DRAM densities (even without considering projected slowdowns in DRAM scaling), it is unlikely that the 64× gap will be closed by the time exascale systems are targeted to come online.

To address the conflicting objectives of bandwidth and capacity, our envisioned exascale memory architecture consists of a hierarchical organization (see Figure 1c). The first level consists of the eight stacks of DRAM on a silicon interposer accompanying the EHP chip providing an estimated 128 Gbytes of memory and 4 TBps of bandwidth (assuming only modest improvements in capacity and bandwidth compared to current technologies). A second level consists of off-package memory with a lower peak bandwidth of around 1 TBps. In the exascale timeframe, we anticipate that the second level could be implemented with emerging NVM technologies (such as phase change memory and memristors) that are projected to overtake DRAM as the economical, bulk memory technology of choice. The first-level DRAM provides high bandwidth and low energy-per-bit memory access, as well as buffering of store operations for the NVM layer. The second-level off-package memory provides the required per-node capacity at cost and power levels below those available with DRAM technology. For systems requiring even greater memory capacities, we envision that a third level of storage class memory (such as flash) could be incorporated in each node (not shown in the figure).

Research is needed to explore different mechanisms for using this heterogeneous memory organization. One option is to operate the fast die-stacked DRAM as a conventional, software-transparent, last-level cache that would allow the fast memory to be used for existing software without any additional programmer effort or changes to the operating systems or runtimes. Another option is to expose the heterogeneous memories to the software levels for explicit management. With such an approach, movement of data between the two memories would be under software direction to enable full control of data placement for performance tuning. It is an open research question as to what "software" should be making decisions. This software could take the form of modifications to the hypervisor, operating system, or runtimes, such that the management of the heterogeneous memories is largely transparent to the application programmer.[8] Alternatively, or additionally, the management of the memories could be directly exposed to the application programmer. Key research needs for heterogeneous memory architectures include the development of tools and infrastructure to understand application memory access patterns, characterization of application memory capacity requirements, and evaluation of the effectiveness of different management schemes in extracting locality for effective use of the heterogeneous memory organization.

Apart from the memory bandwidth and capacity challenges we have discussed, it is well documented that the memory system is also responsible for an increasingly large fraction of power consumption in modern servers. Even with aggressive assumptions about memory and interface technology improvements reducing total DRAM access energy from approximately 60 to 80 picojoules per bit for DDR3 to 2 picojoules per bit, we project that sustaining 4 TBps could consume more than half of the entire exascale system's power budget on memory accesses alone. Processing in memory (PIM) implemented via 3D die stacking allows some computation to be moved close to the data, enabling high memory bandwidth with reduced energy consumption per access. Although 3D die stacking is imminent across a range of memory designs in the industry, the use of die stacking for in-memory processing introduces a new set of architectural opportunities and challenges.

A key research question for PIM is the nature of the in-memory processors. In the context of HPC, data-parallel accelerators present a particularly compelling baseline design point, because they can exploit the bandwidth made available via die stacking, and they do so with improved energy efficiency. As a result, APUs in which GPU computing units provide a significant fraction of their peak computational power are a good match for this class of processors. Furthermore, the ability of today's APUs to support well-understood programming models (such as C/C++, Fortran, Java, OpenMP, Python, and OpenCL[2]) enable these in-memory processors to be used over a wide range of application domains.

Early evaluations show that even off-the-shelf GPU designs incorporated as PIM can yield 5 to 10× improvement in energy efficiency over traditional GPU implementations across different classes of GPGPU benchmarks, while performance benefits are also observed for memory-intensive computations.[9] Much of the energy-efficiency benefits arise from reduced data movement, tailoring the processor implementation to less-aggressive operating frequencies, and the use of low-power process technologies. Additional optimization of today's APU microarchitectures for further gains when used within in-memory implementations is a rich area for future research.

Overall, we envision a heterogeneous memory architecture as a key component required to support exascale-capable systems. The multiple levels of memory and PIM provide the necessary hardware foundations, but at the same time, they also open up new challenges for the programming environment and system software developers.

Programming challenges

A key requirement underpinning our vision of heterogeneous exascale computing is to greatly improve the programmability of large and heterogeneous systems. A supercomputer comprising 100,000 nodes (or more), wherein each node includes CPU and GPU resources, heterogeneous memories, PIM capabilities, network interfaces, and dynamic energy and resilience management features, can quickly become intractable for the application programmer to productively create new code (or even to port existing programs). Significant research investment is needed to explore the implications of exposing hardware features to the system software or programming model, abstract the underlying hardware from application programmers, and develop techniques to exploit the opportunities that come from leveraging HSA at the lower runtime levels.

Our vision strongly advocates for using HSA as a critical cornerstone to make heterogeneous exascale systems practical and effective. Although today's GPGPU programming languages let programmers exploit GPUs' vast computational power to accelerate algorithms with abundant regular, coarse-grained parallelism, many parallel algorithms remain which current GPUs cannot effectively execute, owing to their irregular and fine-grained parallelism. Meanwhile, CPUs are well suited for these codes by using complex speculation hardware to tolerate irregularity, but their restricted number of hardware threads limits their benefits, and a large portion of multithreaded applications do not match either of the two current design points. HSA addresses the gap between abundant regular parallelism and modest irregular parallelism by providing a shared CPU/GPU memory space, eliminating the need to copy data between CPU and GPU memories, and defining task queues that are operated on in user space, resulting in lower task-dispatch overheads than current GPU environments.

Today, GPGPU programmers must directly extract the data parallelism from their applications in order to run them efficiently on GPU hardware. AMD is pursuing programming enhancements that mitigate these challenges. For each computational kernel or loop, the compiler and runtime must target a mix of CPU cores and/or GPU computing units. We are developing enhancements to the compiler and runtime to implement dynamic policies that map computations to the different computing resources to improve performance while running within a constrained power budget. We are also actively exploring how to efficiently support mainstream programming languages (such as C, C++, and Fortran), as well as domain-specific languages (such as Scout) and programming systems (such as Legion) for effective and scalable deployment on large clusters of APUs.

Our proposed exascale vision not only provides heterogeneous computing, but it also introduces a heterogeneous memory architecture that has its own programmability needs. The PIM paradigm in particular presents new challenges and research opportunities. An important enabler for in-memory processing is the availability of effective and accessible programming models. Low-level APIs that allow high degrees of control and flexibility are necessary for runtime system and library developers. High-level abstractions that easily enable the distribution of data and computation across multiple in-memory processors with appropriate affinities are necessary for improving application developer productivity. Intermediate abstraction levels are likely to be of use for trading off system efficiency and programmer effort. Development of these key components of the software ecosystem will be crucial for broadly enabling PIM. Fortunately, many of the modern programming constructs developed for task parallelism, nonuniform memory architectures, and accelerators have close parallels in PIM and should ease the adoption of well-understood constructs to PIM programming.

HSA is also likely to be a key enabler in improving PIM's programmability, via its support for improved programmability of accelerators including a unified memory space and the option of cache coherence among the host and accelerators within a node. We believe extending such a model to in-memory processors (in contrast to, say, more constraining models where an in-memory processor can access only its own local memory stack and might not be cache coherent with the rest of the system) provides several significant benefits, such as

- the ease of debugging and profiling code running on the in-memory processors;
- the ability to gradually and selectively port performance-critical, memory-bound segments of code to PIM without the need for explicit memory copies or other additional data movement;
- the enablement of programmers to focus their performance optimization efforts on critical loops even within the selected code ported to PIM (and let code between loops rely heavily on hardware support for communication and coherence among the host APU and in-memory processors); and
- the enablement of autotuning tools that can profile code on both the host and PIM and optimize the placement of data and computation.

Identifying efficient hardware implementations that enable HSA-like features for in-memory processors, while minimizing hardware and energy costs, are also important areas of future research.

Although HSA provides a strong foundation for parallel programming within the EHP node, exascale programs must scale across entire systems. Existing programs typically use the message passing interface for internode communication. We expect that this model will continue to be needed and should be well supported by the EHP NIC and software stack. However, research is also needed to explore ways in which HSA can be extended to provide a more uniform task-based programming model spanning both intra- and internode parallelism (for example, by extending HSA to work within a distributed Partitioned Global Address Space (PGAS) environment). Envisioning the NIC itself as a coherent HSA-enabled accelerator could be a key aspect of this approach. We believe this direction aligns well with distributed task-based programming models, such as those embodied by Legion and Chapel.

Dealing with physical constraints

The construction of an effective EHP node requires that the design stays within various physical constraints. Some of these constraints are mainstream, such as power and thermals. Other constraints are more specific (but not exclusive) to the HPC domain, such as high levels of resilience and reliability.

The strategy to use GPU computing units as the primary throughout engines for an exascale system is heavily influenced by their superior power-performance efficiency compared to CPU cores. However, even with heterogeneous computing, additional research to further reduce the power consumed by GPUs (and the rest of the system) is still critical to ultimately achieve a scalable and practical exascale supercomputer. As the industry moves to increasingly smaller feature sizes, performance scaling will become dominated by the physics of the computing environment and in particular by the transient behavior of interactions between power delivery, power management, and thermal fields. Although tightly coupled heterogeneous processors benefit from reduced communication latency, higher power efficiency, and improved performance, they also give rise to new management challenges, including thermal coupling and performance coupling due to phenomena such as physical asymmetry in thermal and power signatures between the diverse elements and asymmetry in performance. These coupling effects are fundamental to the operation of heterogeneous architectures. Power-performance tradeoffs in heterogeneous processors are determined by coupled behaviors between major components, owing to their on-die integration, programming model, and processor physics.

To this end, our research focuses on in-depth characterization of the management challenges in heterogeneous processors executing emerging HPC workloads to understand sensitivities and scaling trends of applications against various hardware characteristics. Our research also identifies runtime metrics, APIs, and predictors that can capture these various coupling effects in heterogeneous processors and identify performance and power sensitivities to different hardware characteristics. We encapsulate these runtime metrics and sensitivity predictors into sophisticated power management algorithms that coordinate power states among different components of the heterogeneous processor to optimize for a node-level objective under various physical constraints of the processor.[10] Such coordinated power management will be the key to manage future heterogeneous processors operating under strict power and thermal constraints and will meet exascale systems' high performance and power efficiency goals.

In addition to power and thermals, many high-availability servers have additional requirements for resilience and reliability. However, the aggressive use of heterogeneous computing in an exascale context greatly increases the need for new mechanisms to ensure reliable operation. Even a small probability of a node-level failure, when multiplied across 100,000 nodes, can easily result in a mean-time-to-failure rate of a few hours! The usage of GPU-based computing for most of the system's flops also introduces new challenges, because conventional GPU architectures do not typically provide the same levels of reliability, availability, and serviceability (RAS) as mission-critical CPU designs. For example, data corruption for a graphics application might lead to a minor color perturbation of a single pixel in a single frame, which would likely go unnoticed by the end user; however, as GPU resources are deployed for HPC applications, error detection and correction become critical design considerations. The inclusion of new memory technologies (such as 3D DRAM and NVM) also require research efforts to devise effective RAS management techniques suited to their unique attributes. Finally, architects need design and analysis tools to aid RAS design decisions for heterogeneous platforms used in exascale systems. Resiliency and reliability research for heterogeneous exascale systems should attack two main goals: minimizing the occurrence of silent errors in the exascale node and reducing the RAS solution's overall cost.

The following research activities address the first goal:

- the development of fault detection and recovery and repair techniques that ensure a low silent error rate and sufficiently large mean time to interrupt at the application level,

- field studies of the nature of SRAM and DRAM failures that occur in large production supercomputers, and
- the development of health monitoring tools.[11]

Any reliability techniques developed must account for the impact of the exascale node's heterogeneity. One promising research direction to reduce the overall cost of RAS is to develop software-based techniques for fault detection and recovery. Elevating RAS to the software stack removes the burden of implementing RAS from the hardware designer and also reduces the amount of die area dedicated solely for reliability. Software-based reliability could also be implemented on commodity hardware, further lowering costs. An example of this approach is compiler-managed implementation of redundant multithreading (RMT) for GPUs. RMT uses redundant execution, instead of hardware duplication or coding techniques, to detect faults. We developed an initial implementation and evaluation of this technique on commercial AMD Radeon GPU hardware.[12] We have been developing simulation and runtime tools that aid in understanding the impact of faults for applications that run on heterogeneous hardware.[13] These tools help hardware designers prioritize RAS decisions to meet the reliability targets at low cost, help application developers gain insights into the impact of program structure on resilience, and guide software optimizations to enhance overall system resilience.

For AMD, solving the research problems surrounding exascale computing is not merely an abstract intellectual exercise. Many of AMD's current products and technology capabilities play directly into the longer-term exascale vision, and in a complementary fashion, innovations developed on the path to exascale computing will accelerate the introduction of new capabilities into its non-HPC commercial offerings. Although this article focuses on the exascale-related aspects of heterogeneous computing, AMD is committed to heterogeneous computing as the key technology for higher performance and lower power from servers to desktops, from game consoles to embedded systems, from supercomputers to laptops. The combination of heterogeneous hardware architectures, along with open system architectures (HSA) and the accompanying software ecosystems, will pave the way for the computing systems of the future. MICRO

Acknowledgments

AMD, the AMD Arrow logo, and combinations thereof are trademarks of Advanced Micro Devices. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

References

1. J. Fildes, "Supercomputer Sets Petaflop Pace," *BBC News*, 9 June 2008; http://news.bbc.co.uk/2/hi/technology/7443557.stm.
2. HSA Foundation, "Heterogeneous System Architecture (HSA): Architecture and Algorithms," Int'l Symp. Computer Architecture tutorial, 2014; www.slideshare.net/hsafoundation/isca-2014-heterogeneous-system-architecture-hsa-architecture-and-algorithms-tutorial.
3. N. Enright Jerger et al., "NoC Architectures for Silicon Interposer Systems: Why Pay for More Wires When You Can Get Them (From Your Interposer) for Free?," *Proc. 47th Ann. IEEE/ACM Int'l Symp. Microarchitecture*, 2014, pp. 458–470.
4. B.A. Hechtman et al., "QuickRelease: A Throughput-Oriented Approach to Release Consistency on GPUs," *Proc. IEEE 20th Int'l Symp. High Performance Computer Architecture*, 2014, pp. 189–200.
5. D.R. Hower et al., "Heterogeneous-Race-Free Memory Models," *Proc. 19th Int'l Conf. Architectural Support for Programming Languages and Operating Systems*, 2014, pp. 427–440.
6. J. Power et al., "Heterogeneous System Coherence for Integrated CPU-GPU Systems," *Proc. 46th Ann. Int'l Symp. Microarchitecture*, 2013, pp. 457–467.
7. "FastForward 2 R&D Draft Statement of Work," report LLNL-PROP-652542, Lawrence Livermore Nat'l Lab., 2014; https://asc.llnl.gov/fastforward/rfp/04_DraftSOW_04-03-2014.pdf.

8. M. Meswani et al., "Heterogeneous Memory Architectures: A HW/SW Approach for Mixing Die-stacked and Off-package Memories," *Proc. IEEE 21st Int'l Symp. High Performance Computer Architecture*, 2015, pp. 126–136.
9. D.P. Zhang et al., "TOP-PIM: Throughput-Oriented Programmable Processing in Memory," *Proc. 23rd Int'l Symp. High-Performance Parallel and Distributed Computing*, 2014, pp. 85–98.
10. I. Paul et al., "Cooperative Boosting: Needy versus Greedy Power Management," *Proc. 40th Ann. Int'l Symp. Computer Architecture*, 2013, pp. 285–296.
11. V. Sridharan et al., "Memory Errors in Modern Systems: The Good, the Bad, and the Ugly," *Proc. 20th Int'l Conf. Architectural Support for Programming Languages and Operating Systems*, 2015, pp. 297–310.
12. J. Wadden et al., "Real-World Design and Evaluation of Compiler-Managed GPU Redundant Multithreading," *Proc. 41st Ann. Int'l Symp. Computer Architecture*, 2014, pp. 73–84.
13. B. Fang et al., "GPU-Qin: A Methodology for Evaluating the Error Resilience of GPGPU Applications," *Proc. IEEE Int'l Symp. Performance Analysis of Software and Systems*, 2014, pp. 221–230.

Michael J. Schulte is a senior fellow at Advanced Micro Devices, where he's the principal investigator on AMD's FastForward 2 Node Architecture Project. His research interests include power-efficient processor design, heterogeneous processor architectures, and high-performance computing. Schulte has a PhD in electrical engineering from the University of Texas at Austin. Contact him at michael.schulte@amd.com.

Mike Ignatowski is a fellow at Advanced Micro Devices, where he's the principal investigator for AMD's FastForward 2 Memory Technology Project. His research interests include emerging memory technology and architectures. Ignatowski has an MS in computer engineering from the University of Michigan. Contact him at mike.ignatowski@amd.com.

Gabriel H. Loh is a fellow at Advanced Micro Devices. His research interests include computer architecture, processor microarchitecture, emerging technologies, and 3D die stacking. Loh has a PhD in computer science from Yale University. He's a senior member of IEEE and an ACM distinguished scientist. Contact him at gabriel.loh@amd.com.

Bradford M. Beckmann is a senior member of the technical staff at Advanced Micro Devices, where he directs research in the areas of heterogeneous (CPU+GPU) architectures, cache coherence protocols, memory constancy models, on-chip networks, and simulation. Beckmann has a PhD in computer science from the University of Wisconsin–Madison. Contact him at brad.beckmann@amd.com.

William C. Brantley is an AMD Fellow at Advanced Micro Devices, where he leads AMD's exascale projects funded by the Department of Energy. His research interests include high-performance computing, system performance, and exascale. Brantley has a PhD in electrical engineering from Carnegie Mellon University. He's a member of the ACM and a life member of IEEE. Contact him at bill.brantley@amd.com.

Sudhanva Gurumurthi is a senior member of the technical staff at Advanced Micro Devices, where he directs research on resiliency and reliability. He is also a visiting associate professor in the Department of Computer Science at the University of Virginia. His research interests include resiliency and reliability, datacenter architectures, and energy-efficient design. Gurumurthi has a PhD in computer science and engineering from Pennsylvania State University. He's a senior member of IEEE and the ACM. Contact him at sudhanva.gurumurthi@amd.com.

Nuwan Jayasena is a principal member of the technical staff at Advanced Micro Devices, where he leads research efforts on processing in memory and multilevel memories. His research interests include memory systems, heterogeneous computing, processor microarchitecture, and emerging

technologies and applications. Jayasena has a PhD in electrical engineering from Stanford University. He's a senior member of IEEE. Contact him at nuwan.jayasena@amd.com.

Indrani Paul is a senior member of the technical staff at Advanced Micro Devices, where she leads research on energy utilization in AMD's exascale projects funded by the Department of Energy. Her research interests include power-efficient design, power management and modeling, emerging workloads, and heterogeneous architectures. Paul has a PhD in electrical and computer engineering from the Georgia Institute of Technology. She's a member of IEEE. Contact her at indrani.paul@amd.com.

Steven K. Reinhardt is a senior fellow at Advanced Micro Devices, where he is the principal investigator and technical lead on AMD's DesignForward exascale R&D project. His research interests include high-speed networking, system architecture, and system simulation. Reinhardt has a PhD in computer science from the University of Wisconsin–Madison. He's an IEEE Fellow and an ACM Distinguished Scientist. Contact him at steve.reinhardt@amd.com.

Gregory Rodgers is a principal member of the technical staff at Advanced Micro Devices, where he leads the exascale programmability team. He is also a visiting professor in the Department of Computer and Information Technology at Purdue University. His research interests include heterogeneous system architectures and runtimes. Rodgers has a PhD in computer science from Pennsylvania State University. Contact him at gregory.rodgers@amd.com.