Answers to Questions

A. **Applications:** What are the applications that motivate future systems? What advances in algorithms and programming systems will support these applications?

We see real-time reality-based simulations of physical processes as a driving force in future parallel systems. The advance in computing power available on every class of platform can challenge preconceived notions of applications are possible within a market segment: problems that formerly required a cluster can be done on an engineering workstation, problems that used to require a workstation can be done on a mobile device. Perhaps more importantly, applications that used to be off-line can be performed in real time and thus provide much greater value.

A major application-level technique will be new theory and algorithms that can partition work across heterogeneous processing elements for near-linear scaling with aggregate system capacity.

B. **Systems:** What hardware architectures and distributed systems will support future applications? What challenges do we face in design and management?

We see heterogeneous architectures as a major force in future parallel systems. A single homogeneous processing element or system design cannot provide high efficiency for every workload. Instead, we should design systems with either statically or dynamically heterogeneous resources, such as a mix of conventional processors and GPU cores.

A second direction for system design is dynamic reconfigurability for improved performance of efficiency. In a power-limited system, for example, some workloads may do better by reducing the size of the cache but increasing the CPU frequency. Others may benefit from disabling some processing elements to allow a single CPU to use more cache capacity.
A key challenge in this environment is managing data movement and bandwidth between processing elements; even in a shared-memory architecture, there may be a high synchronization cost to accessing data from different types of processing elements.

A challenge for system software will be managing and scheduling heterogeneous resources; to date operating systems have dealt with a single processor type and two levels of memory: DRAM and disk/flash. Future systems may have many more processor and memory types, requiring new techniques to derive maximum performance or efficiency.

A second challenge for software is power management in a power-limited system. As with heterogeneous systems, software must determine where to deploy power for maximum performance/efficiency, which can be different for each workload.

A third challenge for systems software is how to abstract the complexity and heterogeneity of a system: if too many low-level details are presented to programmers, code can be brittle and not portable, and difficult to write. If too few details are presented, it is difficult for programmers to take maximum advantage of the available hardware. For example, should the operating system abstract away multiple processor types by migrating threads between them, or should this be left to applications? Shared memory may be a convenient low-level abstraction, in that it has efficient hardware implementations that reduce software overhead, yet provides tight control over communication. Whether memory should be coherent or treated as PGAS is still an open question.

C. Technologies: What emerging technologies will change fundamental assumptions in hardware and software? What constraints disappear? What challenges arise?

We know of three new hardware technologies that may substantially change system design:

- **3D die stacking.** This will provide much more memory at high bandwidth and low latency, and can change how many memory-aware algorithms are written. Change assumptions about data management, communication In addition, it provides low-latency communication between different processor types, allowing more use of heterogeneity.

- **Non-volatile memory**, such as STT-MRAM, memristors, or PCRAM. These technologies can revolutionize storage, by providing byte-addressable persistent data directly to user-mode applications. This can substantially change how programs work with persistent data.

- **Silicon photonics.** Low-latency, high-bandwidth communication can change assumptions about the need for locality and how data is managed. It can allow new system designs that depend more on communication.
The major constraint we see growing even more in importance is power limitation, and the rise of dark or dim silicon. Such systems will fundamentally have to manage complicated power tradeoffs.

D. Methodologies: How should we perform interdisciplinary research that spans applications, systems, and technologies? How should abstraction layers evolve?

As a baseline approach, we advocate the crossing-the-chasm approach: pick representative candidate applications, look for tailored hardware and software approaches to improve the performance of those applications, and then generalize increasingly broadly.

We also advocate developing a new set of principles for designing abstractions: beyond the ability to independently evolve across the abstraction boundary, future systems should look at grey-box approaches that reveal performance information to upper levels of the software stack in order to enable platform-specific optimizations. The importance of performance predictability must be a consideration in abstraction design.

We also advocate decomposing existing mechanisms when possible, to reduce the overhead when the entire mechanism is not needed and to allow new uses. As an example, virtual memory address translation can be decomposed into a table lookup, a protection check, and an address computation.

Finally, it is important to learn from communities that have already shown success in integrating information from multiple levels of the stack, such as databases where theoretical results are heavily guided by the performance characteristics and capabilities of hardware. Similarly, the HPC community has a long tradition of integrating algorithm design with application and system design.

E. Risks: What are the risks that threaten the success of XPS research directions? How do we guard and hedge against these threats?

Software productivity will become a bigger issue, in that the gap between average software and high-performance software that fully utilizes hardware will continue to grow. Platforms that require algorithmic changes and cannot perform optimizations in a compiler/runtime/operating system require much more developer effort, and if there are multiple platforms, that effort may need to be repeated for each platform.

A second risk is the lack of benchmarks or established performance evaluation techniques for heterogeneous systems. Standardized benchmarks have allowed great progress in architecture and compilers by allowing independent evolution with a fixed interface. When a system expands to include software co-design, how can the cost of that co-design be factored into results? In
addition, how can system that use different amounts of hardware or power be effectively compared?