(a) Applications: What are the applications that motivate future systems? What advances in algorithms and programming systems will support these applications?

Applications:
- Simulations
  - Weather
  - Molecular (e.g. biological)
  - Neurological
- Cloud computing
- Big data

Algorithmic and programming advances:
- Domain-specific algorithmic advances
- I/O-efficient data structures and algorithms
- Programming models to ease management of concurrency

(b) Systems: What hardware architectures and distributed systems will support future applications? What challenges do we face in design and management?

Hardware architectures
- Multi-core with ever more cores
- Hardware transactions -- these may need to be modified to provide programmers a clear model so that they can use them for more than lock elision
- Persistent DRAM
- Secure “enclaves” may enable consolidation (and distribution) of computations that are currently too security-sensitive to be outsourced
- Extreme high-density storage (e.g. shingled disks)
- Scratchpad RAM

Challenges in design and management
- Increasing cores requires increased memory and disk bandwidth
- Persistent DRAM requires the ability to recover after buggy code corrupts the data. Transactions?
- Enclaves do not solve all security problems, e.g state rollback attacks. It’s not clear how to prevent these attacks.
- Shingled disks require changes to software for organizing data on disk.
- Scratchpad RAM requires new algorithms to take advantage of it.

(c) Technologies: What emerging technologies will change fundamental assumptions in hardware and software? What constraints disappear? What challenges arise?

Emerging technologies and their implications
- Write-optimized data structures will make indexing data on the fly extremely cheap.
  - Obstacles to maintaining indices will drop away
  - Point queries will become the bottleneck
- Persistent RAM will make achieving ACID semantics much, much cheaper

(d) Methodologies: How should we perform interdisciplinary research
that spans applications, systems, and technologies? How should abstraction layers evolve?

Interdisciplinary research methods

- One successful approach is, rather than ten people who are each specialists in one domain, form teams with people who straddle areas. Our team has a nearly pure theory person, a theory/systems person, and a theory applications person. Because we can all talk a common language, we can easily understand each other’s problems and recognize the significance of each other’s results.

Abstraction layers

Eric Brewer once said that, “Database designers strove for economy of semantics, whereas operating systems implementors strove for economy of mechanism.” In other words, databases provide a very clean and simple abstraction that is difficult to implement and, initially, had high performance costs. Operating systems largely chose simple, fast mechanisms and gave applications whichever semantics could be implemented most straightforwardly and with good performance. However, after over 40 years of work, it is now possible to implement ACID semantics with excellent performance.

This may be a good, or at least necessary, model for the future. Researchers should strive to define and implement “ideal” abstraction layers that are easy to use but potentially difficult to implement. While those “holy grail” abstraction layers are being optimized and improved over the years, practitioners can use fast but less clean abstractions to get work done today.

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

Parallelism is so crucial to increasing performance of computing systems, that XPS seems well positioned to be both a high pay-off and low-risk research program.