Hardware Session Overview

- Key Constraints
  - Power Budget
  - Procurement Budget
  - Delivery Schedule
- Focused on the Design Trade-offs possible
  - Node: Memory, power management, synchronization
  - Interconnect: Latency, BW, Message rates, Network topology
  - I/O: Great co-design opportunity
  - General: Programming models, HW acceleration for algorithms
- Collaboration Methods: How do we answer these questions?

Node tradeoffs

- Basic core design fixed (small changes possible)
- Heterogeneous nodes
  - Number of heavy-weight versus light-weight?
  - How important is single thread performance?
- There are significant advantages to explicitly manage data movement.
  - What mechanisms would be most useful?
- How many threads per coherence domain could you *effectively* utilize?
- How important is reproducibility (run times and bit level)?

Node tradeoffs

- What do you need in hardware to write resilient algorithms?
- How many levels of memory hierarchy can you deal with? Relative sizes?
- Direct application access to NVRAM (memory, disk or both)? Trade-off between capacity and latency?
- Will system software use NVRAM?
- Synchronization between cores?
- What types of power management tools can you use? Slow down cores, turn off memory channels, etc…
    - Do all cores need to run identically?
    - What information do you need?

Interconnect tradeoffs

- Relative importance of bandwidth, latency and messaging rate?
- How do we evaluate network topologies?
- How important is inter-node communication to performance?

IO

- What IO rates do we need to sustains for traditional model?
- How do we codesign a revolutionary approach to data analysis and resiliency?
- What operations could be moved to node-local persistent storage? What semantics are needed for accessing?

General Questions

- What hierarchical and hybrid programming models do we need to support?
- What types of acceleration an we put in hardware to support algorithms (fast collectives for krylov, atomics,)?
- Who is responsible for insuring a stable API for accessing these features?

Collaboration Methods

- Proprietary simulation/emulation tools are closely held.
  - Compact/proxy applications are useful for HW vendors.
- Simulation and emulation methods are needed to better understand application behavior.
  - A range of fidelities needed.
- Better tools for rapidly characterizing application behavior.
  - Extrapolating to future architectures.