Planning for the Exascale Software Center

Robert Ross
Mathematics and Computer Science Division
Argonne National Laboratory
rross@mcs.anl.gov
Major Components of DOE Exascale Initiative

Platform R&D
2 Vendor Tracks
- Power
- Integration
- Risk Mitigation

Critical Technologies
(everyone benefits)
- Memory
- Nonvolatile storage
- Optics

Software and Environments
- Operating environment
- Systems Software
- System reliability
- Programming model

Co-design
- Performance models
- Simulators
- Applications integration with vendors

Platforms
- Early prototypes to ensure component integration and usefulness
- Risk mitigation for vendors – Non recoverable engineering cost

Exascale Initiative

Research and development, but also coordination and Quality Assurance.
The Changing Face of Extreme-Scale Platforms

Yesterday
- Seymour & team designs and hand builds set of computers
- Dozen HPC companies flourish: incompatible OS & components

Today
- Commodity components and Open Source move effort to integration
- Design-Build partnerships for extreme scale e.g.
  - LLNL/ANL/IBM
  - Sandia/ORNL/Cray
  - Fujitsu/NEC/Hitachi/Riken

Tomorrow
- Globally distributed teams, Diverse technology providers, Co-Design with applications, Open Source Software
  - e.g.: PRACE
Current HPC software approach needs a reboot

*It is our view that complex systems almost always fail in complex ways ... – Columbia Accident Investigation Board Report, August 2003*

- Software development uncoordinated with hardware features
  - (e.g., power mgmt, multicore tools, math libraries, advanced memory models)
- Only basic acceptance test software is delivered with platform
  - UPC, HPCToolkit, Optimized libraries, PAPI, can be YEARS late
- Vendors often “snapshot” key Open Source components and then deliver a stale code branch
- Community codes unprepared for sea change in architectures
- “Coordination via contract” is poor and only involves 2 parties
- No global evaluation of key missing components
Current ESC Planning Team

Coordinating PIs:
  Pete Beckman, Argonne National Laboratory
  Jack Dongarra, University of Tennessee

Pavan Balaji, Argonne National Laboratory
George Bosilca, University of Tennessee
Ron Brightwell, Sandia National Laboratory
Jonathan Carter, Lawrence Berkeley National Laboratory
Franck Cappello, University of Illinois
Barbara Chapman, University of Houston
James Demmel, University of California Berkeley
Al Geist, Oak Ridge National Laboratory
Bill Gropp, University of Illinois
Paul Hargrove, Lawrence Berkeley National Laboratory
Michael Heroux, Sandia National Laboratory
Kamil Iskra, Argonne National Laboratory

Rusty Lusk, Argonne National Laboratory
Allen Malony, University of Oregon
Arthur Barney Maccabe, Oak Ridge National Laboratory
Terry Moore, University of Tennessee
John Mellor-Crummey, Rice University
Robert Ross, Argonne National Laboratory
Marc Snir, University of Illinois
Rajeev Thakur, Argonne National Laboratory
Vinod Tipparaju, Oak Ridge National Laboratory
Jeff Vetter, Oak Ridge National Laboratory
Kathy Yelick, Lawrence Berkeley National Laboratory

Special thanks for feedback and help developing material:

Bronis de Supinski, Lawrence Livermore National Laboratory
Mark Seager, Lawrence Livermore National Laboratory
Pat McCormick, Los Alamos National Laboratory
Andy White, Los Alamos National Laboratory
Peg Williams, Cray
Peter Young, Cray

Robert Wisniewski, IBM
Al Gara, IBM
Bill Dally, Nvidia
Steve Parker, Nvidia
David Lombard, Intel
Ed Temple, Argonne National Laboratory
Exascale Software Center (ESC)

Goal: Ensure successful deployment of coordinated exascale software stack on Exascale Initiative platforms.

Ultimately responsible for success of software

- Identify required software capabilities
- Identify gaps
- Design and develop open-source software components
  - Both: evolve existing components, develop new ones
  - Includes maintainability, support, verification
- Ensure functionality, stability, and performance
- Collaborate with platform vendors to integrate software
- Coordinate outreach to the broader open source
- Track development progress and milestones
Assumptions

- Several vendor platform partnerships
- ~2015 early scalability demonstration systems
  - Arch 2010-2011; System build 2015
- ~2018 exascale system
  - Arch 2014-2015; System build 2018
- Co-design centers provide initial applications
- ESC:
  - Partnership funding agencies, labs, and universities
  - Responsible for the common software environment for EI systems
  - All development will be open source
  - Some components will be integrated and supported by vendor, others will be provided atop basic platform, supported by ESC
  - Vendor-specific components will be part of their platform strategy
    - E.g.: system management, RAS, compiler, etc.
The Exascale Software Center in One Slide

 Scope
  – Deliver high quality system software for exascale platforms
    • ~2015, ~2018
  – Identify software gaps, research & develop solutions, test and support deployment
  – Increase the productivity and capability and reduce the risk of exascale deployments

 Cost:
  – Applied R&D: ~10-20 distributed teams of 3 to 7 people each
  – Large, primarily centralized QA, integration, and verification center

 Schedule overview
  – 2010 – Q1 2011: Planning and technical reviews
  – April 2011: Launch Exascale Software Center!
ESC Organization Chart

Advisory Committee
Vendor Partnerships
International Partners
Co-Design Council

ESC Management Director:
Deputy:

DOE Program HQ

Exascale Joint Management Council

Software Components
Programming Models
Operating Systems and Run Time
Application Programmer Tools
Numerical Libraries and Frameworks
Data Management and Analysis
System Management and Cybersecurity

QA
Testing, QA, HPC Center Integration, Support

Co-Design
Vendor Integration Team
Application Integration and Performance
Specific Challenges for the Center

- How does the Center participate in co-design activities? What are vendors and application teams looking for from the Center?
- Given resource constraints, how does the Center select components to be supported? What does the Center require of these components and the teams that develop them? How does the Center interact with these teams?
- How do we engage with the larger DOE, US, and international communities?
The Exascale Software Center and Co-Design Processes

Initial System Design

Platform Architects (vendors) → R&D Software Community → Co-Design Centers → Research → Initial Prototypes → Test & QA → Integration, Deployment, Support → Refining Design

Identify Needs → Identify Gaps → Research → Applied Research and Development
Co-Design Examples (current successes)

Co-Design Example: Math Libraries
- BLAS, Sca/LAPACK co-design: Well-known, huge success.
- Trilinos, PETSc:
  - Services:
    - Libraries: State-of-the-art libraries for scalable solvers, etc.
    - Framework: Building blocks for app implementations.
  - Vendor collaboration:
    - Cray-specific sparse kernels: avg 30% performance boost.
    - Trilinos, PETSc pre-compiled for Crays:
      - module load trilinos/10.4.1
      - module load petsc
    - “Heads-up” on trends:
      - Vendor: Early Sp kernels results for future systems.
      - Trilinos/PETSc teams: Perf-impacting app, algorithm trends.

MPICH Co-design with IBM and Cray
- The MPICH group has for years worked closely with both IBM and Cray in co-designing MPICH for Blue Gene and XT systems.
- Specific optimizations were recently added to MPICH to improve its multithreaded performance to attain the high multithreaded message rates needed for BG/Q.
- On the 32-bit BG/P architecture, we worked with IBM to make MPI_Aint as a 64-bit quantity to enable HDF-5 and other I/O libraries access files larger than 2GB correctly.
- Similarly, various optimizations and features, such as support for checkpoint-restart and improvements to the Nemesis communication layer, were added in collaboration with Cray to better support Cray systems.

More Examples from Co-Design with IBM

Hardware
- Added MMU
- # of cores
- # of chips
- Efficiency and safety improvements in the power module designs

Operating System
- Smart scheduler
- Reduced memory footprint
- Performance improvements (python, shr lib)
- Speculative multi-threading system programming interface

System Management
- Distributed control system
- New pervasive security model
- Open source, plug-in dynamic allocator
- Many RAS usability and performance improvements

MPI
- MPI scalability fixes (esp. memory)
- Fine-grained locking or lock-free methods for thread safety (open portable atomic lib)
- Derived datatype optimizations

Code Development & Tools
- Improved performance counters
- Improved behavior for TLS and TM to better match application needs
- Flexible programming model: MPI everywhere; flexible task/thread ratio
- Increased user level APIs

I/O
- Full size, standard PCI-e cards
- Debugger interfaces for IO nodes
- Persistent memory uses
- Page sizes and scalability improvements

Co-Design Examples: System Mgmt
- Blue Gene/Q resource allocation system
  - Collaboration between ANL, LLNL, and IBM
  - Supports dynamic allocation of network and node resources
- Blue Gene/P RAS real-time streaming interface
  - Supports real-time notifications of system failure events
  - Dovetails with existing RAS polling system from BG/L
  - Collaboration between ANL, LLNL, and IBM
Vendors and Co-Design

- Want something like ESC to coordinate and take *real* responsibility for features and milestones
  - Improved leverage over projects that are currently less responsive than needed
- Do not want “toss over the wall” strategy. “hardening” cannot be done by different team.
- Need to manage risk of final machine functionality, performance, stability and acceptance
- Key ESC models:
  - ESC developed -- vendor integrated and supported
  - ESC developed – ESC provided, and supported
- Formalized roles between ESC and Vendors for development, risk, support, and acceptance
- Feedback and progress tracking between ESC and vendors must be shared
- Application co-design centers should coordinate discussions of system software through ESC
- NDA material for roadmaps, across co-design centers, etc will be difficult to coordinate

<table>
<thead>
<tr>
<th>Examples of software package</th>
<th>Primary developer</th>
<th>First-level Support Provider</th>
<th>Second-level Support Provider</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 RAS, system mgmt, compilers</td>
<td>Vendor</td>
<td>Vendor</td>
<td>Vendor</td>
</tr>
<tr>
<td>2 OS, MPI, PAPI, math libraries</td>
<td>ESC</td>
<td>Vendor</td>
<td>ESC</td>
</tr>
<tr>
<td>3 Performance tools, I/O libraries</td>
<td>ESC</td>
<td>ESC</td>
<td>ESC</td>
</tr>
<tr>
<td>4 Perl, Python</td>
<td>Broader Community</td>
<td>Vendor</td>
<td>Broader Community</td>
</tr>
<tr>
<td>5 Eclipse IDE</td>
<td>Broader Community</td>
<td>Broader Community</td>
<td>Broader Community</td>
</tr>
</tbody>
</table>
Applications and Co-Design

- Want something like ESC to coordinate and take *real* responsibility for features and milestones
  - Improved leverage over projects that are currently less responsive than needed
- Want to know specifics about hardware and available software
- Applications will provide best estimates of needs for exascale science:
  - Data movement, memory sizes, programming models, etc
- Applications will test and evaluate prototype system software
- Need help managing risk of final machine functionality, performance, stability and acceptance
- Formalized roles between ESC and App Co-Design Centers for development, risk, support, and acceptance
- Feedback and progress tracking between ESC and App Co-Design Centers
- Coordinate discussions of system software through ESC
- NDA material for roadmaps, across co-design centers, etc will be difficult to coordinate
An Overabundance of Software

- Software and functionality fall into variety of categories:
  - I/O Storage
  - Math Libraries
  - Performance Tools
  - Etc.

- Software executes in a number of domains within the system:
  - Service node, I/O nodes, compute nodes, login nodes, etc
## Examples: ORNL XT3 I/O and Math Libraries

### Software Requirements for HPC

<table>
<thead>
<tr>
<th>Site</th>
<th>System</th>
<th>Node Type</th>
<th>L1 Category</th>
<th>L2 Type</th>
<th>L3 Function</th>
<th>Package</th>
<th>Provider</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORNL</td>
<td>Cray XT3</td>
<td>All</td>
<td>App Support</td>
<td>Library</td>
<td>I/O &amp; Storage</td>
<td>HDF5_PAR</td>
<td>NCSA</td>
</tr>
<tr>
<td>ORNL</td>
<td>Cray XT3</td>
<td>All</td>
<td>App Support</td>
<td>Library</td>
<td>I/O &amp; Storage</td>
<td>HDF5_SERIAL</td>
<td>NCSA</td>
</tr>
<tr>
<td>ORNL</td>
<td>Cray XT3</td>
<td>All</td>
<td>App Support</td>
<td>Library</td>
<td>I/O &amp; Storage</td>
<td>netCDF</td>
<td>UCAR/Unidata</td>
</tr>
<tr>
<td>ORNL</td>
<td>Cray XT3</td>
<td>Compute</td>
<td>App Support</td>
<td>Library</td>
<td>Math</td>
<td>PetSC</td>
<td>ANL</td>
</tr>
<tr>
<td>ORNL</td>
<td>Cray XT3</td>
<td>Compute</td>
<td>App Support</td>
<td>Library</td>
<td>Math</td>
<td>Aztec</td>
<td>Sandia</td>
</tr>
<tr>
<td>ORNL</td>
<td>Cray XT3</td>
<td>Compute</td>
<td>App Support</td>
<td>Library</td>
<td>Math</td>
<td>BLAS</td>
<td>AMD</td>
</tr>
<tr>
<td>ORNL</td>
<td>Cray XT3</td>
<td>Compute</td>
<td>App Support</td>
<td>Library</td>
<td>Math</td>
<td>FFTPack</td>
<td>Netlib</td>
</tr>
<tr>
<td>ORNL</td>
<td>Cray XT3</td>
<td>Compute</td>
<td>App Support</td>
<td>Library</td>
<td>Math</td>
<td>FFTW</td>
<td>MIT</td>
</tr>
<tr>
<td>ORNL</td>
<td>Cray XT3</td>
<td>Compute</td>
<td>App Support</td>
<td>Library</td>
<td>Math</td>
<td>LAPACK</td>
<td>AMD</td>
</tr>
<tr>
<td>ORNL</td>
<td>Cray XT3</td>
<td>Compute</td>
<td>App Support</td>
<td>Library</td>
<td>Math</td>
<td>MUMPS</td>
<td>CERFACS</td>
</tr>
</tbody>
</table>

Note that this list is best viewed as a data table (menu Data >> Filter >> Autofilter), and Print...
### Examples: LLNL BG/P Tools

<table>
<thead>
<tr>
<th>Site</th>
<th>System</th>
<th>Node Type</th>
<th>L1 Category</th>
<th>L2 Type</th>
<th>L3 Function</th>
<th>Package</th>
<th>Provider</th>
</tr>
</thead>
<tbody>
<tr>
<td>LLNL</td>
<td>BG/P</td>
<td>Service</td>
<td>Prog Env</td>
<td>Tool</td>
<td>Infrastructure</td>
<td>LaunchMON</td>
<td>LLNL (Open Source)</td>
</tr>
<tr>
<td>LLNL</td>
<td>BG/P</td>
<td>Service</td>
<td>Prog Env</td>
<td>Tool</td>
<td>Infrastructure</td>
<td>MRNet</td>
<td>University of Wisconsin</td>
</tr>
<tr>
<td>LLNL</td>
<td>BG/P</td>
<td>Service</td>
<td>Prog Env</td>
<td>Tool</td>
<td>Infrastructure</td>
<td>DynInst</td>
<td>University of Wisconsin</td>
</tr>
<tr>
<td>LLNL</td>
<td>BG/P</td>
<td>Service</td>
<td>Prog Env</td>
<td>Tool</td>
<td>Infrastructure</td>
<td>StackWalker</td>
<td>University of Wisconsin</td>
</tr>
<tr>
<td>LLNL</td>
<td>BG/P</td>
<td>Service</td>
<td>Prog Env</td>
<td>Tool</td>
<td>Infrastructure</td>
<td>secure VNC</td>
<td>Vaporware</td>
</tr>
<tr>
<td>LLNL</td>
<td>BG/P</td>
<td>Service</td>
<td>Prog Env</td>
<td>Tool</td>
<td>GUI</td>
<td>Tool Gear</td>
<td>LLNL (Open Source)</td>
</tr>
<tr>
<td>LLNL</td>
<td>BG/P</td>
<td>Service</td>
<td>Prog Env</td>
<td>Tool</td>
<td>GUI</td>
<td>tcl/tk</td>
<td>Open Source</td>
</tr>
<tr>
<td>LLNL</td>
<td>BG/P</td>
<td>Service</td>
<td>Prog Env</td>
<td>Tool</td>
<td>GUI</td>
<td>X11</td>
<td>Open Source</td>
</tr>
<tr>
<td>LLNL</td>
<td>BG/P</td>
<td>Service</td>
<td>Prog Env</td>
<td>Tool</td>
<td>GUI</td>
<td>Qt</td>
<td>TrollTech (Open Source)</td>
</tr>
<tr>
<td>LLNL</td>
<td>BG/P</td>
<td>Service</td>
<td>Prog Env</td>
<td>Tool</td>
<td>GUI</td>
<td>Performance Analysis</td>
<td>Paratools/Univ. of Oregon</td>
</tr>
<tr>
<td>LLNL</td>
<td>BG/P</td>
<td>Compute</td>
<td>Prog Env</td>
<td>Tool</td>
<td>Performance Analysis</td>
<td>HPM</td>
<td>Processor Vendor and Linux</td>
</tr>
<tr>
<td>LLNL</td>
<td>BG/P</td>
<td>Compute</td>
<td>Prog Env</td>
<td>Tool</td>
<td>Performance Analysis</td>
<td>PAPI</td>
<td>UTK (Open Source)</td>
</tr>
<tr>
<td>LLNL</td>
<td>BG/P</td>
<td>Compute</td>
<td>Prog Env</td>
<td>Tool</td>
<td>Performance Analysis</td>
<td>OTF</td>
<td>Paratools (Open Source)</td>
</tr>
<tr>
<td>LLNL</td>
<td>BG/P</td>
<td>Service</td>
<td>Prog Env</td>
<td>Tool</td>
<td>Performance Analysis</td>
<td>Vampir/VampirServer</td>
<td>Dresden Univ</td>
</tr>
<tr>
<td>LLNL</td>
<td>BG/P</td>
<td>Service</td>
<td>Prog Env</td>
<td>Tool</td>
<td>Performance Analysis</td>
<td>VampirTrace</td>
<td>Dresden Univ</td>
</tr>
<tr>
<td>LLNL</td>
<td>BG/P</td>
<td>Service</td>
<td>Prog Env</td>
<td>Tool</td>
<td>Tool version selection</td>
<td>dotkit</td>
<td>LLNL (Open Source)</td>
</tr>
<tr>
<td>LLNL</td>
<td>BG/P</td>
<td>Service</td>
<td>Prog Env</td>
<td>Tool</td>
<td>Editor</td>
<td>emacs</td>
<td>Open Source</td>
</tr>
<tr>
<td>LLNL</td>
<td>BG/P</td>
<td>Service</td>
<td>Prog Env</td>
<td>Tool</td>
<td>Editor</td>
<td>vim</td>
<td>Open Source</td>
</tr>
<tr>
<td>LLNL</td>
<td>BG/P</td>
<td>Compute</td>
<td>Prog Env</td>
<td>Tool</td>
<td>Performance Analysis</td>
<td>mpiP</td>
<td>LLNL/ORNL (Open Source)</td>
</tr>
<tr>
<td>LLNL</td>
<td>BG/P</td>
<td>Service</td>
<td>Prog Env</td>
<td>Tool</td>
<td>Source Code Control</td>
<td>svn</td>
<td>Open Source</td>
</tr>
<tr>
<td>LLNL</td>
<td>BG/P</td>
<td>Service</td>
<td>Prog Env</td>
<td>Tool</td>
<td>Source Code Control</td>
<td>cvs</td>
<td>Open Source</td>
</tr>
<tr>
<td>LLNL</td>
<td>BG/P</td>
<td>Service</td>
<td>Prog Env</td>
<td>Tool</td>
<td>Source Code Control</td>
<td>git</td>
<td>Open Source</td>
</tr>
</tbody>
</table>
### Examples: LLNL Visualization and Analysis

<table>
<thead>
<tr>
<th>Package</th>
<th>Provider</th>
<th>Support</th>
<th>Criticality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visit</td>
<td>LLNL (Open Source)</td>
<td>LLNL</td>
<td>1</td>
</tr>
<tr>
<td>OpenGL</td>
<td>Open Source</td>
<td>Community</td>
<td>1</td>
</tr>
<tr>
<td>EnSight</td>
<td>CEI</td>
<td>Licensing</td>
<td>2</td>
</tr>
<tr>
<td>ImageMagick</td>
<td>Open Source</td>
<td>Open Source</td>
<td>2</td>
</tr>
<tr>
<td>Tecplot</td>
<td>Tecplot, Inc.</td>
<td>Licensing</td>
<td>2</td>
</tr>
<tr>
<td>IDL</td>
<td>ITT Visual Informations Systems</td>
<td>Licensing</td>
<td>2</td>
</tr>
<tr>
<td>gnuplot</td>
<td>Open Source</td>
<td>Open Source</td>
<td>2</td>
</tr>
<tr>
<td>POV-Ray</td>
<td>Open Source</td>
<td>Community</td>
<td>2</td>
</tr>
<tr>
<td>RasMol</td>
<td>Open Source</td>
<td>Community</td>
<td>2</td>
</tr>
<tr>
<td>vmd</td>
<td>UIUC (Open Source)</td>
<td>UIUC</td>
<td>2</td>
</tr>
<tr>
<td>ParaView</td>
<td>Open Source</td>
<td>Community</td>
<td>2</td>
</tr>
<tr>
<td>NCAR</td>
<td>NCAR (Open Source)</td>
<td>NCAR</td>
<td>3</td>
</tr>
<tr>
<td>mplayer</td>
<td>Open Source</td>
<td>Community</td>
<td>3</td>
</tr>
<tr>
<td>Blockbuster</td>
<td>LLNL (Open Source)</td>
<td>LLNL</td>
<td>3</td>
</tr>
<tr>
<td>GIMP</td>
<td>Open Source</td>
<td>Community</td>
<td>3</td>
</tr>
<tr>
<td>xxdiff/tkdiff/meld</td>
<td>Open Source</td>
<td>Community</td>
<td>3</td>
</tr>
</tbody>
</table>
Breadth of Software Capabilities in ESC

- Initial set of types of software capabilities developed based on IESP report, DOE Exascale workshops, and conversations with vendors and application teams
  - Programming Models
  - Operating Systems and Runtime
  - Application Programmer Tools
  - Numerical Libraries and Frameworks
  - Data Management and Analysis
  - System Management and Cybersecurity

- Is this the minimum set that ESC should support?
- Where do we make the cut between vendor-supported and ESC-supported software?
- What can we rely on the co-design centers to support?
Selecting ESC Components to Provide Capability

- ESC is responsible for delivering successful software
  - Technical evaluation:
    - Criticality to successful deployment and key applications
    - Technical risk for achieving goal
  - Project team evaluation:
    - Team history of delivering high-quality, applied software
    - Management and institutional support

- ESC will make component selection and resource decisions based on criticality and risk
  - Continuous evaluations of progress; adjust resources

Technical Evaluation Matrix

<table>
<thead>
<tr>
<th></th>
<th>Low Risk</th>
<th>Moderate Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESC Supported</td>
<td>Important</td>
<td></td>
</tr>
<tr>
<td>Vendor Supported</td>
<td>Critical</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Most Critical</td>
<td></td>
</tr>
</tbody>
</table>
ESC Software Development

- Successful applied R&D teams are built around clear goal of delivering working, supported packages
- Good software hygiene can’t be someone else’s job
- ESC must **work with successful teams existing processes** or in some cases, boot new teams within institutions with excellent history of deployed software
  - Probably not feasible to launch new team at site without history of software success
- Formal plans and milestones and reviews are necessary for each component
- Co-design feedback and risk-based assessments work well with spiral development discipline for software (common in R&D)

---

**Classic “Waterfall” model**

**“Spiral” model**
Required Processes for ESC Components

- Formulation of clear deliverables with specific targets for functionality, performance, and stability
- Defined team management plan and risk tracking
- Documented software development plans
  - QA (unit tests, integration, etc)
  - Performance testing
  - Documentation, support
  - Bug and new feature tracking
- Resource accounting
- Technical review schedule
- Release schedule
- Integration plan
Distributed Project Staffing Approach

- “ESC Component Teams” should be located where their center of mass has demonstrated success
  - E.g: Math libraries at UTK, Performance tools at UOregon and Rice

- Each Component Team will have at least one “embedded” QA and testing staff member provided by ESC
  - Position will be held by professional QA/build engineer (i.e., not a student or postdoc)
  - Candidates will be approved by ESC director of QA and have performance appraisal “matrix input”

- Each site must have local ESC team members responsible for integration
  - Will belong to production computing division, not R&D division

- QA, integration, and support team will be primarily at one site

- Resources dedicated to collaboration and software development infrastructure is required
Community Engagement

IESP Activities
- European Exascale Initiative
- Japanese Exascale Initiative

ASCR/NNSA Institutes
- Domain Science Institutes
- Computer Science Institutes

ESC

Co-design Centers
- Applications Co-design
- Hardware Co-design

Third Party Software
Next Steps in ESC Planning

- Develop software planning documents:
  - Definition of review materials
  - Formal review in April 2011
- Build application co-design liaisons, develop plan for jointly evaluating key software
- Build links to IESP organizational plan
- Begin technical evaluation and ranking of key software components
- Link to NSF, NASA, DARPA, and other groups
Acknowledgments: ESC Planning Team

Coordinating PIs:

- Pete Beckman, Argonne National Laboratory
- Jack Dongarra, University of Tennessee

Pavan Balaji, Argonne National Laboratory
George Bosilca, University of Tennessee
Ron Brightwell, Sandia National Laboratory
Jonathan Carter, Lawrence Berkeley National Laboratory
Franck Cappello, University of Illinois
Barbara Chapman, University of Houston
James Demmel, University of California Berkeley
Al Geist, Oak Ridge National Laboratory
Bill Gropp, University of Illinois
Paul Hargrove, Lawrence Berkeley National Laboratory
Michael Heroux, Sandia National Laboratory
Kamil Iskra, Argonne National Laboratory

Rusty Lusk, Argonne National Laboratory
Allen Malony, University of Oregon
Arthur Barney Maccabe, Oak Ridge National Laboratory
Terry Moore, University of Tennessee
John Mellor-Crummey, Rice University
Robert Ross, Argonne National Laboratory
Marc Snir, University of Illinois
Rajeev Thakur, Argonne National Laboratory
Vinod Tipparaju, Oak Ridge National Laboratory
Jeff Vetter, Oak Ridge National Laboratory
Kathy Yelick, Lawrence Berkeley National Laboratory

Special thanks for feedback and help developing material:

Bronis de Supinski, Lawrence Livermore National Laboratory
Mark Seager, Lawrence Livermore National Laboratory
Pat McCormick, Los Alamos National Laboratory
Andy White, Los Alamos National Laboratory
Peg Williams, Cray
Peter Young, Cray

Robert Wisniewski, IBM
Al Gara, IBM
Bill Dally, Nvidia
Steve Parker, Nvidia
David Lombard, Intel
Ed Temple, Argonne National Laboratory