Robert W. Wisniewski
Chief Software Architect
Blue Gene Supercomputer Research

IBM Exascale Software Initiatives
Overview

- Activities and investments in exascale
- Key challenges for achieving exascale
  - Thoughts on the co-design process
  - Important issues for working with open source software community
Exascale at IBM

- Work for more than two years on potential architecture – true codesign process
  - Technology exploration: optics, memory, fabrication, etc
  - Packaging
  - Hardware architecture
  - System software: control system, OS, messaging, compilers, performance tools, etc.
  - Application evaluation and workload analysis

- IBM has groups working across vertical stack

- Internal working groups
  - Overall meeting where all teams come together
  - Groups comprise people across teams
    - Ex: at application meetings, hw, system software in attendance
Sample of Current HPC and Exascale Relationships

- ANL
- Astron
- BSC
- Columbia
- CSCS
- EDF
- Edinburgh
- EPFL
- Exascale codesign centers
- Juelich EIC
- KEK
- LBNL
- LLNL
- Melbourne
- NCSA
- NEAMS
- ...

Key Software Challenges for Achieving Exascale

- Lots of technology and hardware challenges – focus here on software

- Technical
  - Already identified individual components: programming models, performance tools, OS
  - Richness of software ecosystem
Your Supercomputer 20 Years Ago

- Application
- Compiler
- Kernel
- Hardware
## Blue Gene Software Stack Openness

### I/O and Compute Nodes
- **Application**
  - Open Toolchain Runtime
  - XL Runtime
  - ESSL
  - MPI
  - Global Arrays
  - Charm++
  - MPI-IO
- **System**
  - DCMF (BG Message Stack)
  - CNK
  - Linux kernel
  - GPFS
  - Messaging SPIs
  - Node SPIs
  - totalviewd
- **Firmware**
  - Diags
  - Bootloader
  - Common Node Service
    - Hardware init, RAS, Recovery, Mailbox
- **HW**
  - Compute nodes
  - I/O nodes

### Service Node/Front End Nodes
- **User/Sched**
  - PerMon
  - BG-Nav
  - HPC Toolkit
  - ISV Schedulers, debuggers
  - Loadleveler
- **System**
  - High Level Control System (MMCS)
    - Partitioning, Job management and monitoring, RAS, Administrator interface
  - DB2
- **Firmware**
  - Low Level Control System
    - Power On/Off, Hardware probe, Hardware init, Parallel monitoring
    - Parallel boot, Mailbox
- **HW**
  - Link cards
  - Service card
  - SN
  - FENs

---

- Green: New open source reference implementation licensed under CPL.
- Yellow: New open source community under CPL license. Active IBM participation.
- Light green: Existing open source communities under various licenses. BG code will be contributed and/or new sub-community started.

© 2010 IBM Corporation
Key Software Challenges for Achieving Exascale

- Lots of technology and hardware challenges – focus here on software

- Technical
  - Already identified individual components: programming models, performance tools, OS
  - Richness of software ecosystem
    - Implication on scope of effort
    - Breadth of application base impacts software stack
    - Richness of programming model impacts software stack
  - Providing greater productivity while maintaining focus on performance
    - Affect individual component decision, OS, compiler, runtimes, libraries
    - Overall amount of productivity software means more effort and requires more support
Overview

- Activities and investments in exascale
- Key challenges for achieving exascale
  - Thoughts on the Exascale and the co-design process
  - Important issues for working with open source software community
Advantages of Co-Design across Application/Software/Hardware
(helping take advantage of multi-core environment)
Atomic Operations
(Iwarx stx on PowerPC)

- N round trips
  - Where N is the number of threads
  - For N=16 2500 → ~2500 cycles, 32→5000, 64→10000
Time for $N$ threads to synchronize on current generation demonstrating the need for improved scalability of atomic operations for next generation.
Use of Scalable Atomic ops in the Messaging Stack

- Handoff mode for communication bound applications

- Non-blocking send and receive operations handoff work
  - Provide function pointer and arguments and request is enqueued
  - Producer of work requests uses new primitive

- Worker threads, up to \( n \) per main sending/receiving execute handoff function
  - MPI processing
  - Message processing
  - Descriptor injected into messaging unit
  - Calls worker thread call new primitive

- Applied within the IBM Hardware, System Software, and Application teams
  - Needs to be applied across the community
Why is Exascale Hard

- Where is the cliff
- Revolution
Why is Exascale Hard

- Is there a cliff
- Evolution vs Revolution
**Bridging to Exascale – Threading/MPI Counts Achievable (50x total sustained performance improvement from 20PF/s)**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Increase in MPI Tasks</td>
<td>2.5x</td>
<td>1.5x</td>
</tr>
<tr>
<td>Increase in Threads/task</td>
<td>2.5x</td>
<td>1.5x</td>
</tr>
<tr>
<td>Increase in Single thread performance</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Frequency</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- SIMD effectiveness</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Architecture tuning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Transparent techniques (speculation etc)</td>
<td>2.4x</td>
<td>1.5x</td>
</tr>
<tr>
<td>Sustained performance increase</td>
<td>15x</td>
<td>3.33x</td>
</tr>
</tbody>
</table>

**Not included:**
- Compiler improvements
- Network enhancements beyond scaling
- Tools improvement
- User performance tuning beyond threading and MPI growth

Not significant memory growth
Modest growth in coherent threads
Supporting Different Shared Memory/Threading Models

- Converse/Charm++
- MPICH
- Global Arrays
- ARMCI
- UPC*
- Other Paradigms*

- DCMF API (C)
  - pt2pt protocols
  - collective protocols
  - DMA Device
  - Collective Device
  - GI Device
  - Shmem Device
  - Message Layer Core (C++)

- SPI

- Network Hardware (DMA, Collective Network, Global Interrupt Network)
Reliability and impact on users – higher up pyramid less user impact

- The vast amount of silicon will make reliability a more difficult challenge.
- Multiple potential directions
  - Work with the user community to determine
    - Option 1) Leave it to system hardware and software to guarantee correctness
    - Option 2) Leave it to the users to deal with potential hardware faults

- Key to scalability is to keep it simple and predictable
- Keep reliability complexity away from the user as that is where the real cost is
- Use hardware/software to perform local recovery at system level
Key Co-Development Models for Achieving Exascale

- For software intended to run on Vendor machines (should be other research initiative also)
  - Interlocking milestones
  - Understanding vendor need for product readiness
  - Vendors sit on review panel for milestone approval

- Funded vendor participation in key components

- IP agreements

- Stable model for API definition with vendor participation
<table>
<thead>
<tr>
<th>Requirement</th>
<th>Open Source</th>
<th>Open Source with formal support</th>
<th>Open Software</th>
<th>Collaborative Development</th>
<th>Co-Development</th>
<th>Proprietary Development</th>
<th>Proprietary Development with Escrow</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Community</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Does not want to be limited to a fully proprietary solution</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flexibility to replace components of the stack</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Open API</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Leverage Government investment</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Protect Government investment</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Applications have common environment</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Scientists need to how their devices work for reproducibility</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td><strong>Provider</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Not held responsible for components that they do not have control over</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Protect other provider proprietary information</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Facility</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Level of Quality</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Best Value</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

**Co Development Required:** example BG/Q MPICH
Exascale is Hard but Achievable

- There should be no cliff
  - Maybe a few drop-offs

- Investment is needed
  - Early, lead-time key

- Evolution
  - With strategic revolution
Exascale SW and Applications - Key Messages

- **Exascale Software delivery will have challenges - but it is tractable:**
  - Challenges due to scale, reliability, power, and application diversity and integration
  - Much expertise has been gained from our experience with tera and petascale systems
    - Threading, heterogeneity, single thread performance, power, memory usage, and reliability

- **Co-Design and collaborative development are key components in achieving Exascale**
  - IBM has an unparalledled track record in co-design & will be active participant in initiatives

- **Use models will shift from HPC-based research around individual applications to integrated workflows addressing complete research and industrial systems**
  - Moving to “commercial grade” requirements: code development, maintenance, verification
  - Interface standards, software frameworks will be increasingly important
  - Data scales and heterogeneity increasingly will add analytics type workloads into workflows

- **Existing programming models will migrate and must be supported efficiently**
  - New programming models should be explored as appropriate - but beware of the “silver bullet”

- **Open Source and community provided software will play a significant role**
  - Must be managed carefully to ensure vendor product schedules and delivery are maintained
    - Interlocking milestones
  - Platform enabling software must remain proprietary
  - Some key opportunities for Open Source contribution include:
    - Frameworks, ‘Revolutionary’ programming models, Performance tools, e.g. Visualizers
Backup
Potential Areas for Strategic Revolution

- **File Systems**
  - Effective meta data, parallel, object-based
- **Debuggers**
  - Radically new techniques would be valuable
- **Tools**
  - Maybe not a revolution but much more emphasis
- **CPO – Continuous Program Optimization**
  - Dynamically evaluate
- **Programming Model**
  - Go forward with one of the existing proposals