

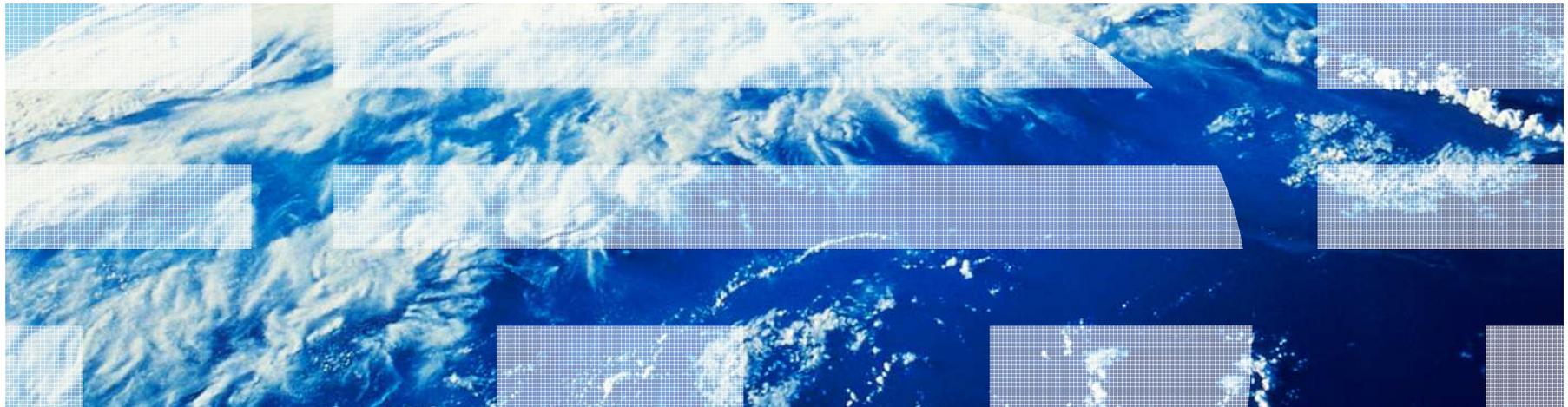
Robert W. Wisniewski

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

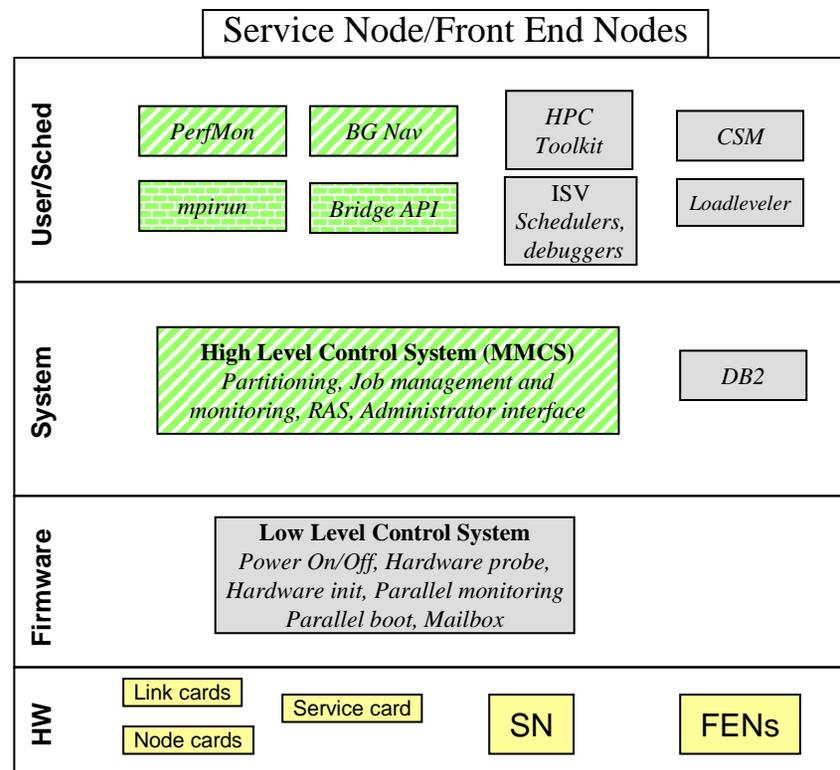
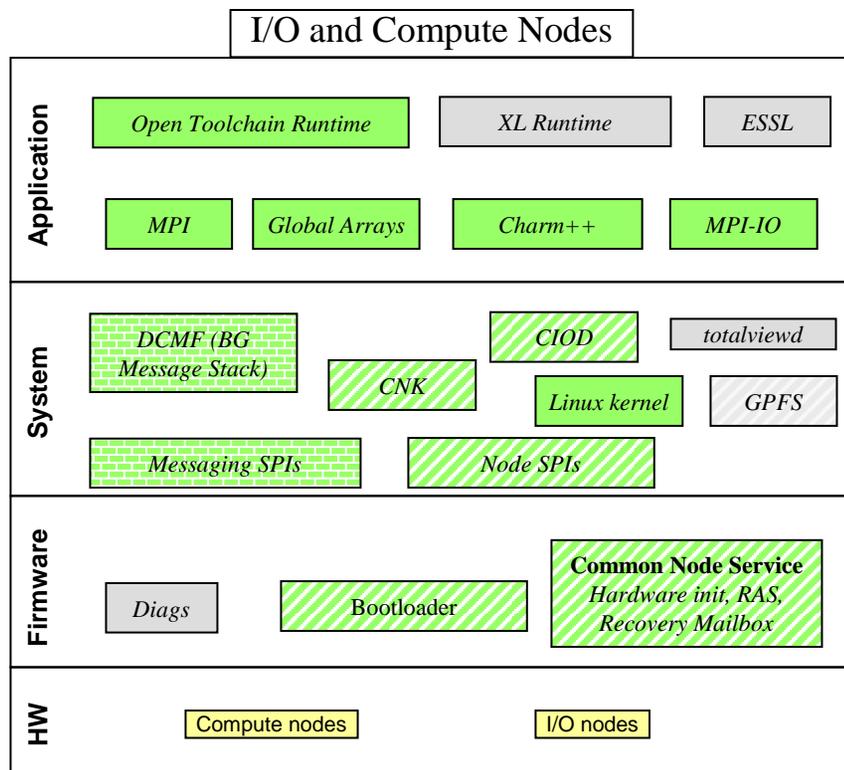
Application

Compiler

Kernel

Hardware

Blue Gene Software Stack Openness



- New open source reference implementation licensed under CPL.
- New open source community under CPL license. Active IBM participation.
- Existing open source communities under various licenses. BG code will be contributed and/or new sub-community started..
- Closed. No source provided. Not buildable.
- Closed. Buildable source available

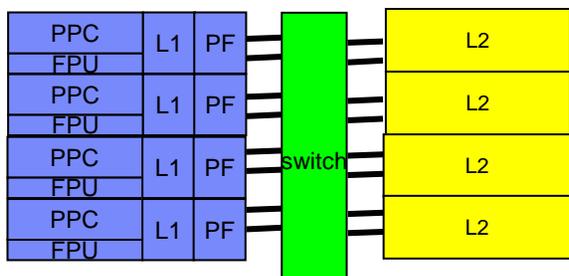
Key Software Challenges for Achieving Exascale

- 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

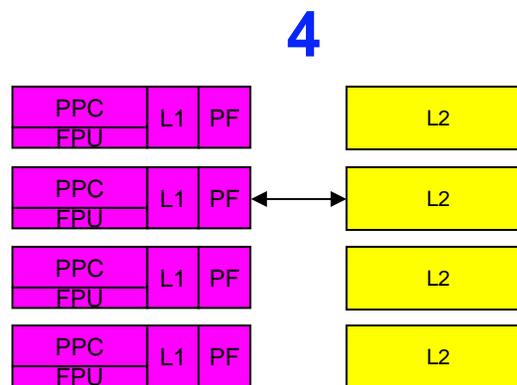
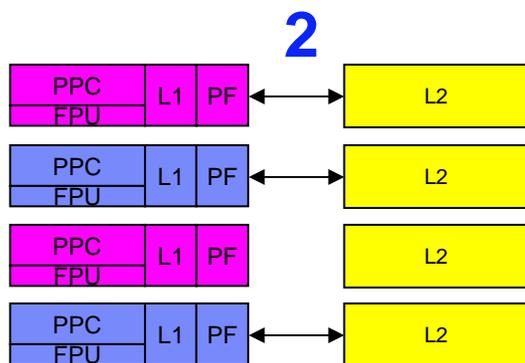
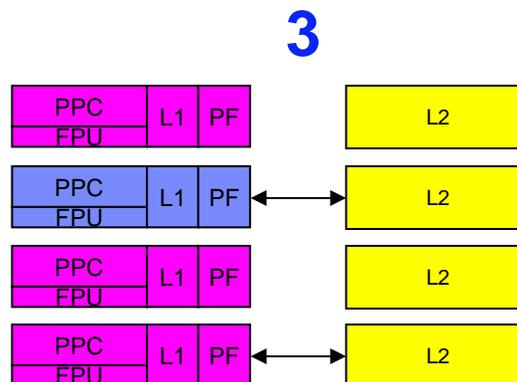
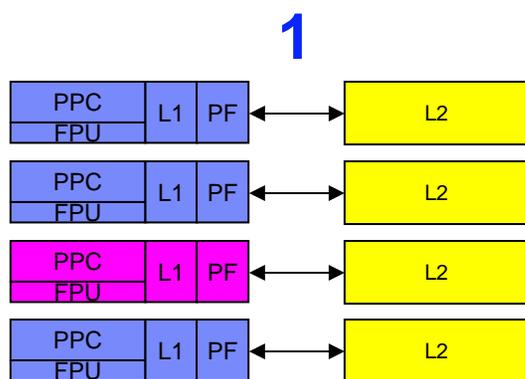
- Activities and investments in exascale
- Key challenges for achieving exascale
- Thoughts on the Exascale and the co-design process
- Important issues for working with open source software community

Advantages of Co-Design across Application/Software/Hardware (helping take advantage of multi-core environment)

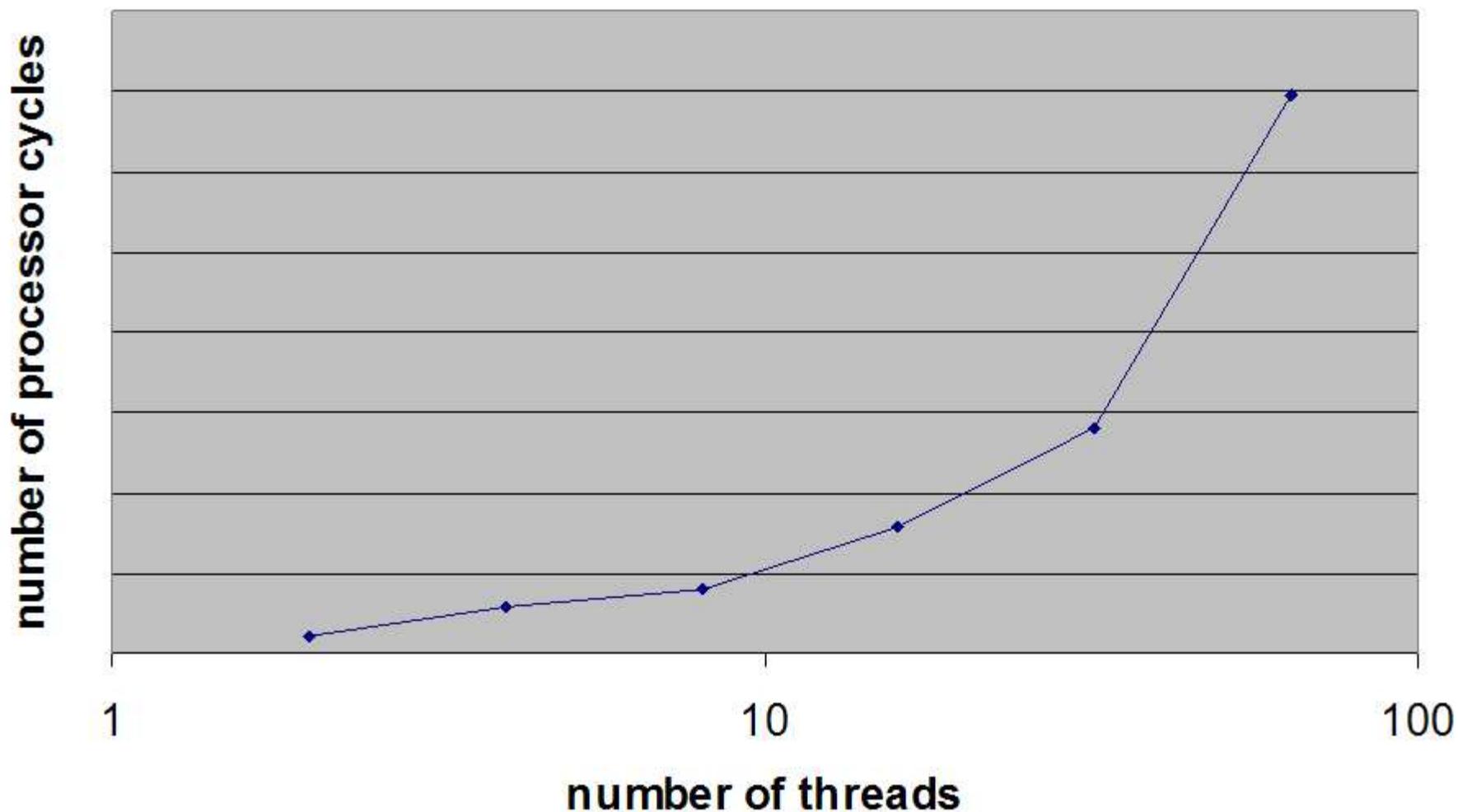


Atomic Operations (Iwax stx on PowerPC)

- 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

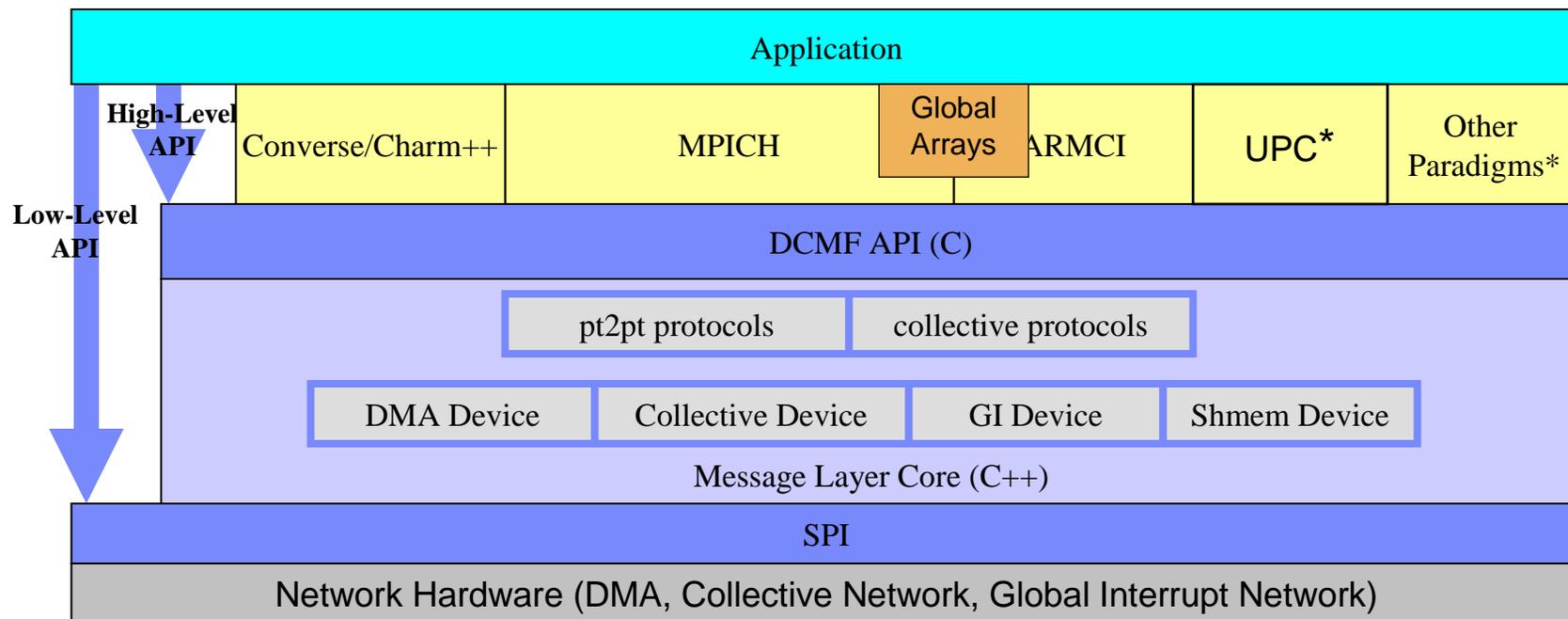


Why is Exascale Hard

- Is there a cliff
- Evolution vs Revolution

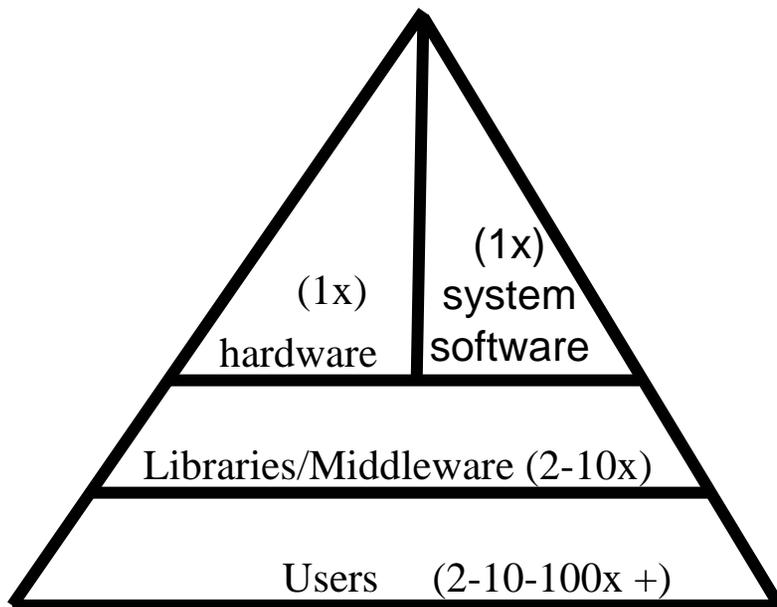


Supporting Different Shared Memory/Threading Models



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

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

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 unparalleled 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