



June 21, 2010

Challenges and Approaches for Exascale Computing: with a software perspective

Robert W. Wisniewski
Chief Software Architect
Blue Gene Supercomputer Research



© 2010 IBM Corporation

Why is Exascale Hard

- Where is the cliff
- Revolution



Why is Exascale Hard

- Is there a cliff
- Evolution vs Revolution

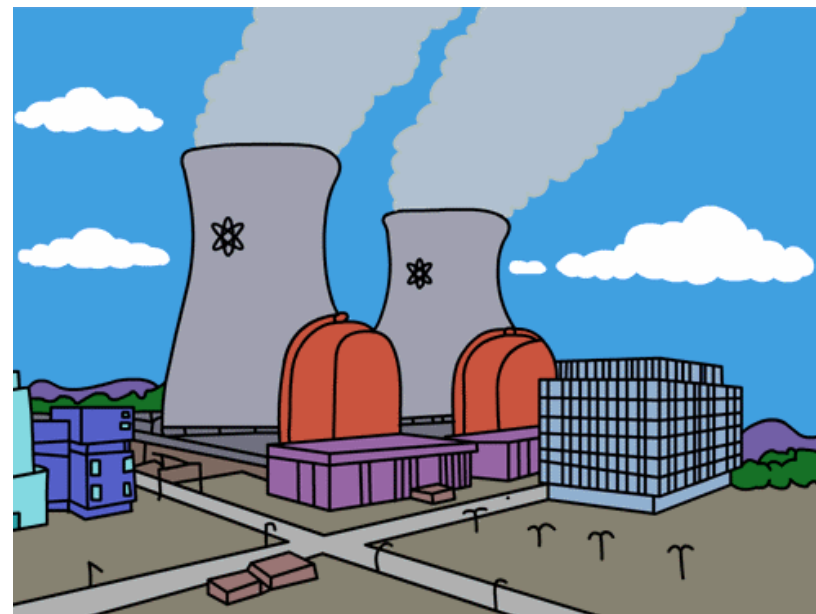


Outline

- **Review of technology trends with implications on software**
 - **Power**
 - **Frequency**
 - **Reliability**
 - **Bandwidths**
 - **Memory, network, I/O**
- **Overall approaches to address the challenges**
- **Concluding thoughts evolutionary versus revolutionary**

Power

- Straight line projection from Jaguar to Exaflop
 - 3GW
- Straight line projection from next generation to Exaflop
 - 200MW

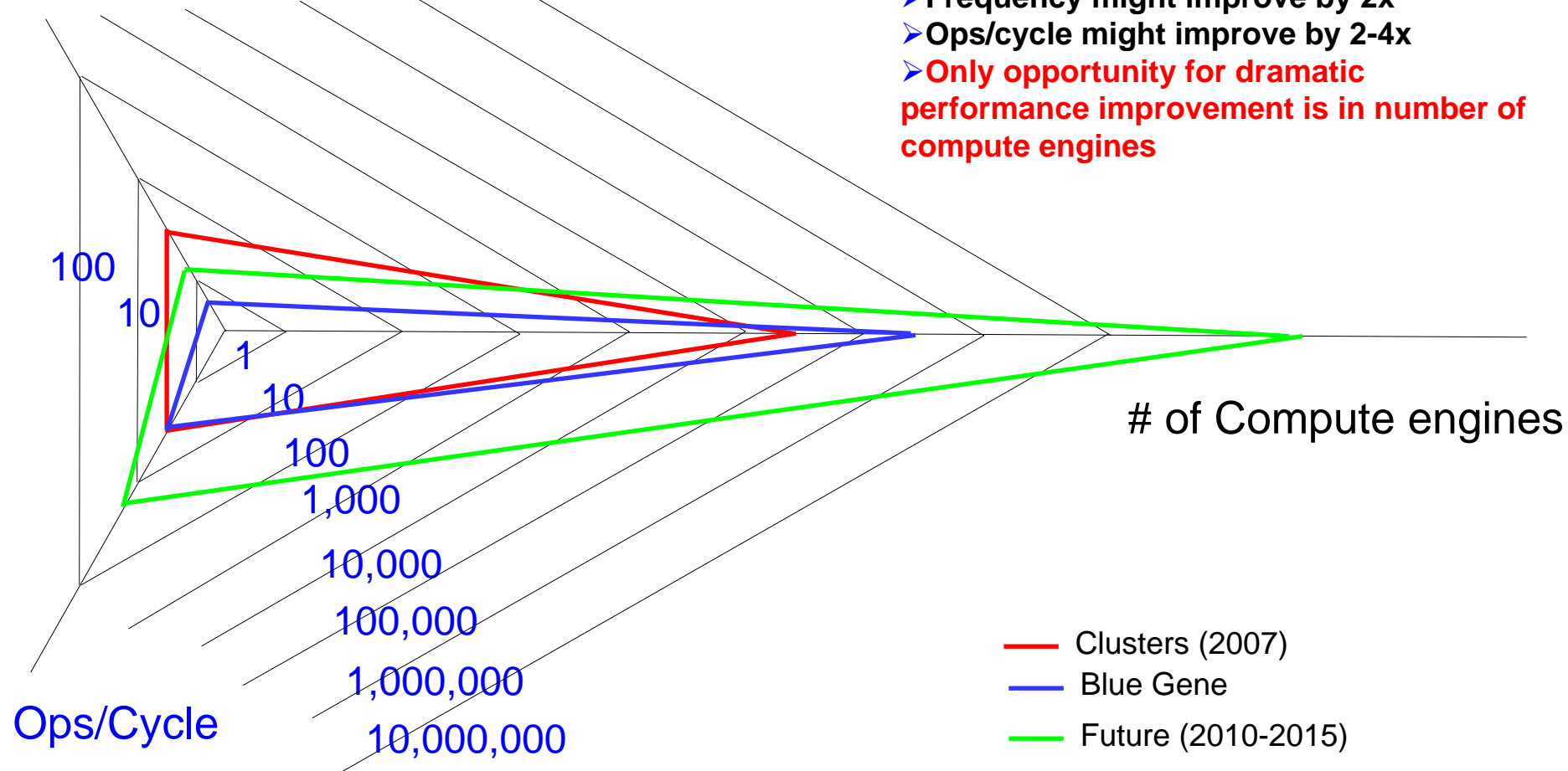


Increased Transistor Count is Driving Performance

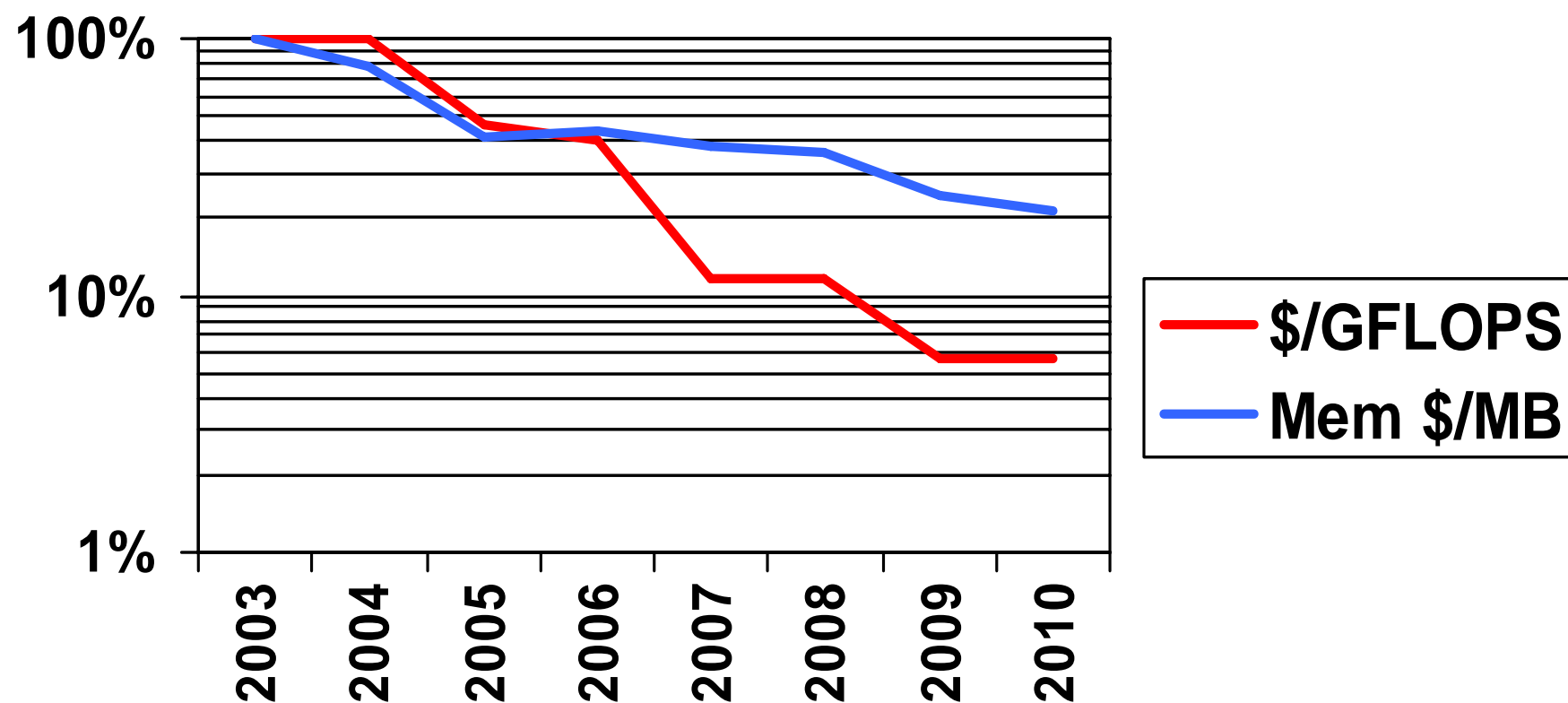
Frequency (GHZ)

Over the next 8-10 years

- Frequency might improve by 2x
- Ops/cycle might improve by 2-4x
- **Only opportunity for dramatic performance improvement is in number of compute engines**



GFLOPS vs DRAM Price Reductions



Assumes constant uP price

Power, Transistor Count, Memory costs

- Will drive power efficient cores
 - Higher degrees of threading
 - Not targeted/optimized for single thread performance
 - ✓ Still can have familiar programming models

Bridging to Exascale (50x total sustained performance improvement from 20PF/s)

	200 PF/s (2015)	1 EF/s (2018)	
user {	2x	2x	Not significant memory growth
Increase in MPI Tasks			
system {	2.5x	1.5x	Modest growth in coherent threads
Increase in Threads/task			
Increase in Single thread performance			
• Frequency			
• SIMD effectiveness			
• Architecture tuning			
• Transparent techniques (speculation etc)			
Sustained performance increase	10x	5x	

** Not included:

Compiler improvements

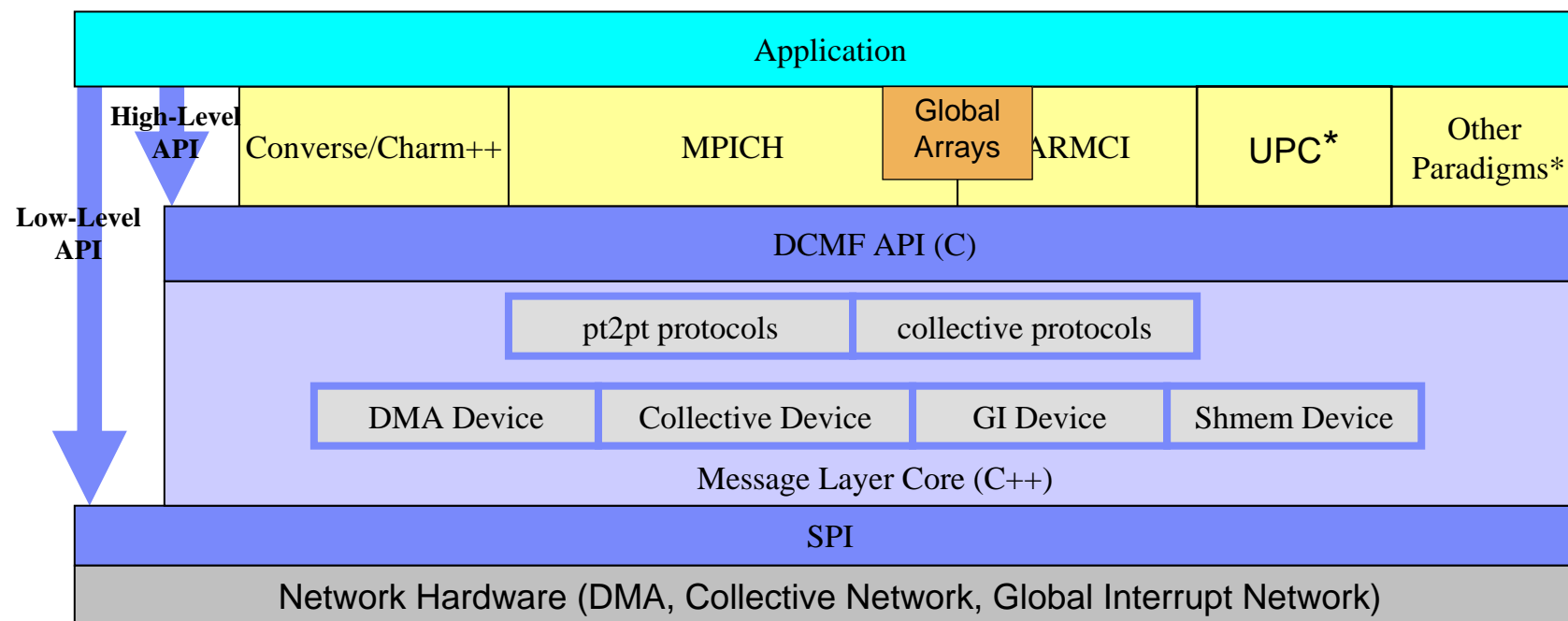
Network enhancements beyond scaling

Tools improvement

User performance tuning beyond threading and MPI growth

...

Supporting Different Shared Memory/Threading Models



Scope of Reliability Problem

- How often does a machine fail
 - Windows box
 - Once a day?
 - Once a month?
 - Once a year?
 - Linux box
 - Once a month?
 - Once a year?
 - Once a decade?

Back to RAS - Scope of Problem

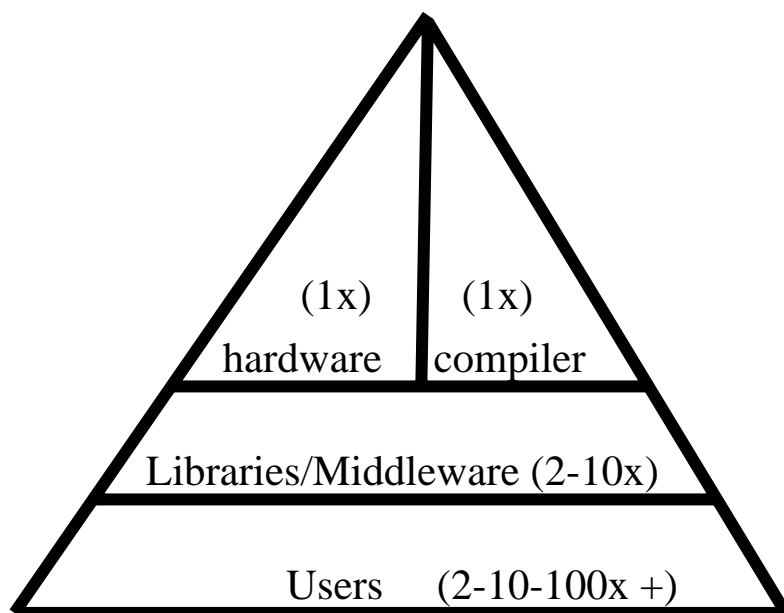
- 128 racks
- X
- 32 node board per rack
- X
- 32 nodes per node board
- X
- 16 cores per node
- =
- 2.1M cores

Scope of Problem

- A failure of once a year translates to
 - Whole machine
 - Every ~14 seconds if core based
 - Every ~4 minutes hour if node based
- A failure of once a decade translates to
 - Machine failing
 - Every ~2 minutes if core based
 - Every ~½ hour if node based
- A goal for an MTBF of 10 days translates to
 - Requiring individual node fails less than every 3600 years
 - Requiring individual cores fails less than every 61200 years

Reliability and impact on users

- 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. Not impossible, just expensive
 - 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

Future Bandwidth Challenges

- **Memory Bandwidth**

- As we add cores to a chip, careful design required to maintain memory bandwidth

- **Network Bandwidth**

- As we increase processor count, careful design required to maintain communication capability between processors

- **I/O Bandwidth**

- As we increase compute capability, careful design required to maintain communication capability into and out of the computer

Possible to address concerns must work on technology

- **Memory Bandwidth**

- Work with memory vendors

- **Network Bandwidth**

- Cost big factor accelerated technology can address

- **I/O Bandwidth**

- New hardware technologies with new software model

Outline

- Review of technology trends with implications on software
- Overall approaches to address the challenges
 - Overall philosophy
 - Productivity
 - Open source / community code
 - Co-design
- Concluding thoughts evolutionary versus revolutionary

Exascale High-Level Software Goals and Philosophy

- **Facilitate extreme scalability**
- **High reliability: a corollary of scalability**
- **Standards-based with consistency across IBM HPC**
- **Open source with community code inclusion where possible**
- **Facilitate high performance for unique hardware**
- **Facilitate new programming models**
- **Co-design between Hardware, System Software, and Applications**

Programmer Productivity

■ Develop applications

- Eclipse PTP (**Parallel Tools Platform**): remote development – **open source**
- Eclipse PLDT (**Parallel Tools Platform and Parallel Languages Development Tools**): open-source.
- Eclipse CDT (**C Development Tools**) - open-source
- Compilers: **OpenMP, UPC**; integrated with Eclipse
- Libraries: **MPI, LAPI, MASS, ESSL, Parallel ESSL**
- Job Execution Environment: **Eclipse plug-ins for PE and LoadLeveler** – open-source

■ Debug applications

- Parallel Debugger: **Petascale debugger**; integrated via Eclipse plug-in

■ Tune applications

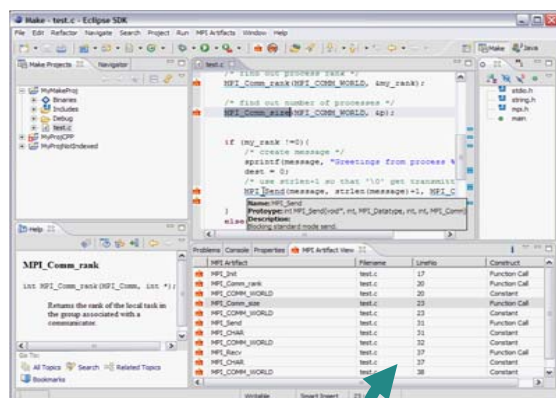
- HPCS Toolkit: **Automatic performance tuning**; integrated via Eclipse plug-in

Performance Tuning with HPCS Toolkit

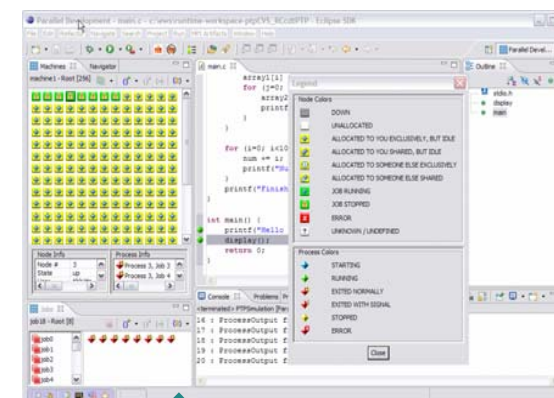
- Based on existing IBM HPC Toolkit for application tuning
- HPCS Toolkit is a set of productivity enhancement technologies
 - Performance Data Collection (extensible)
 - Scalable, dynamic, programmable
 - Completely binary: no source code modification to instrument application...
 - But retains ability to correlate all performance data with source code
 - Bottleneck Discovery (extensible)
 - Make sense of the performance data
 - Mines the performance data to extract bottlenecks
 - Solution Determination (extensible)
 - Make sense of the bottlenecks
 - Mines bottlenecks and suggests system solutions (hardware and/or software)
 - Assist compiler optimization (including custom code transformations)
 - Performance “Visualization” (extensible)
 - Performance Data / Bottleneck / Solution Information feedback to User
 - Output to other tools (e.g., Kojak analysis, Paraver visualization, Tau)

Hybrid Application Development Environment

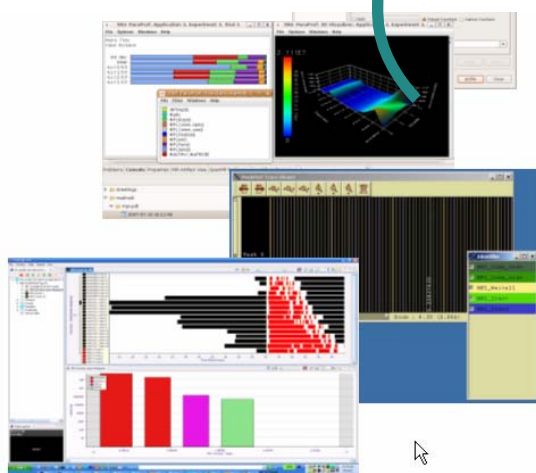
Coding & Analysis Tools



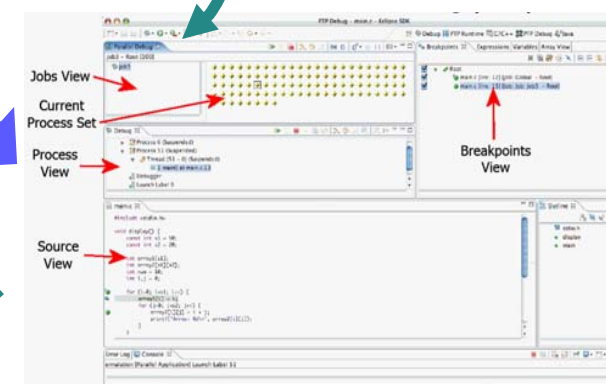
Launching & Monitoring Tools



eclipse

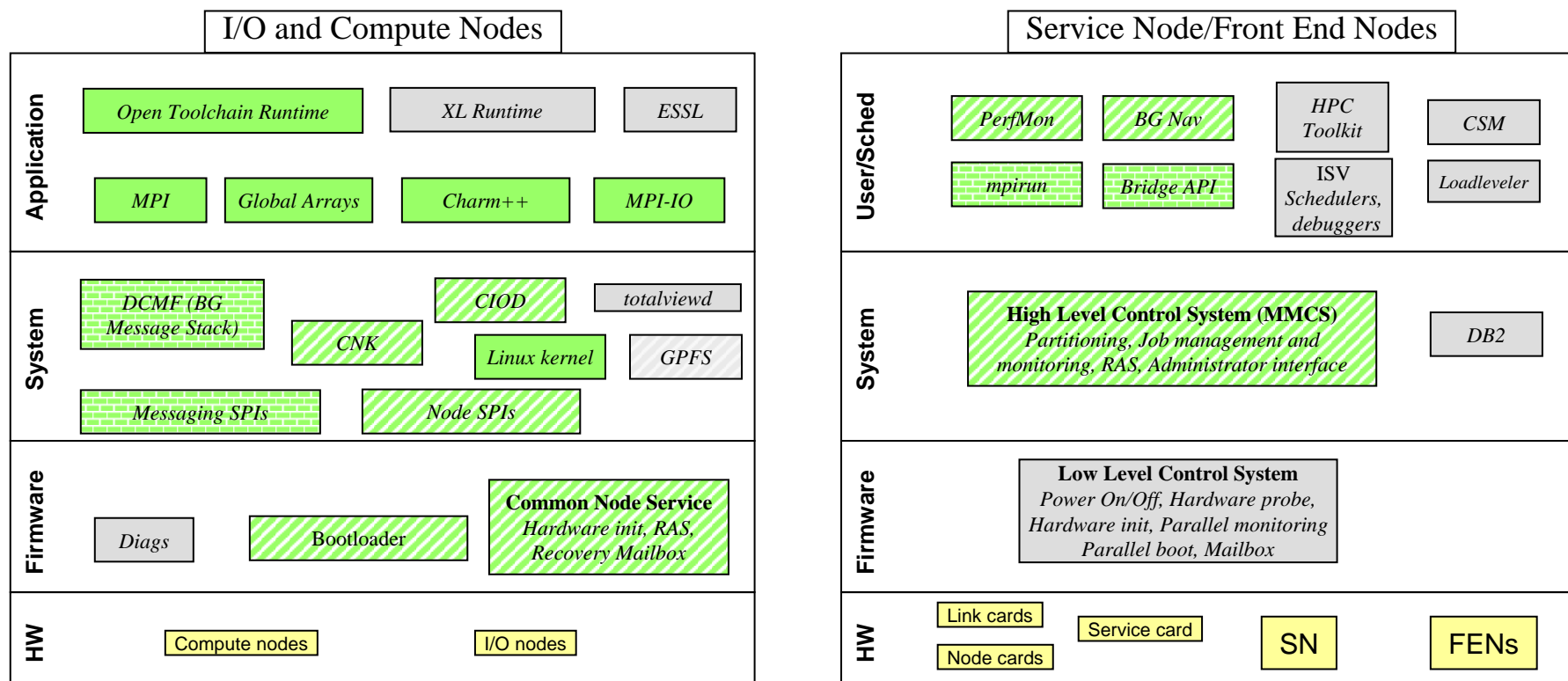







Performance Tuning Tools



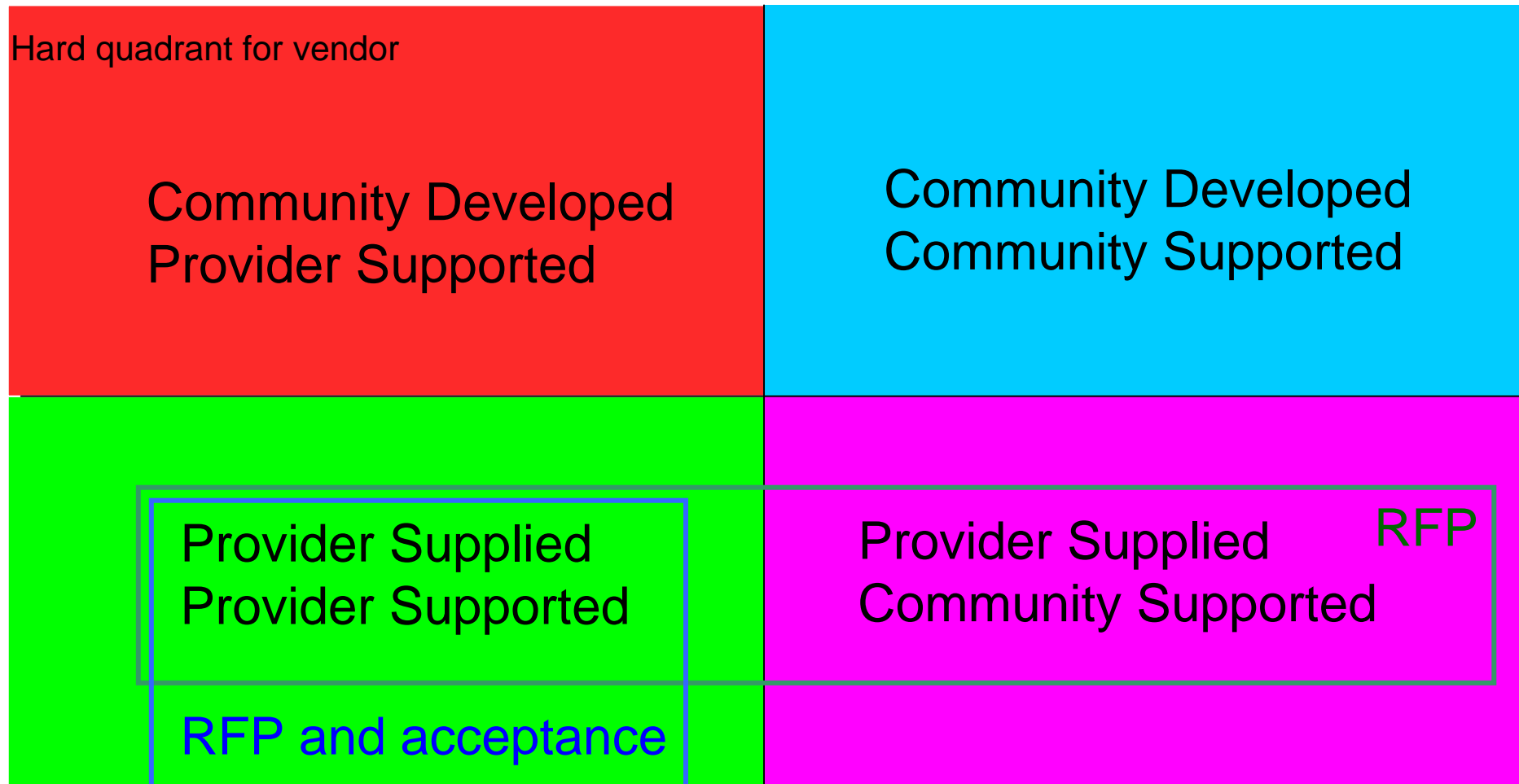
Debugging Tools

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

Suggested Model for software



*developed implies who implemented

*supplied could be co-developed

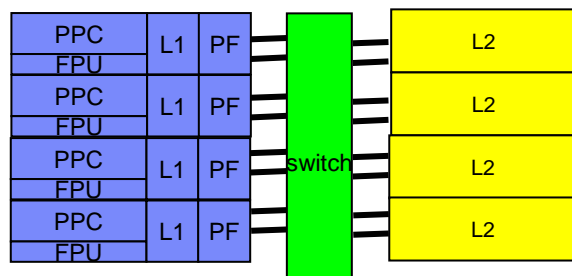
Requirement	Open Source	Open Source with formal support	Open Software	Collaborative Development	Co-Development	Proprietary Development	Proprietary Development with Escrow
Community							
Does not want to be limited to a fully proprietary solution	X	X	X	X	?		
Flexibility to replace components of the 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

Open Source Summary

- Co-Development (joint ownership and responsibility with a formal agreement) meets all requirements
- Open source with formal (paid) support agreements meet all but one requirement

Advantages of Software/Hardware Co-Design

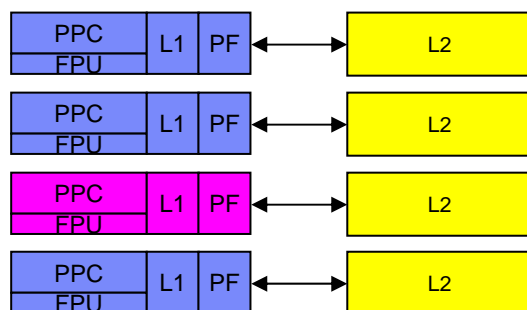
(helping take advantage of multi-core environment)



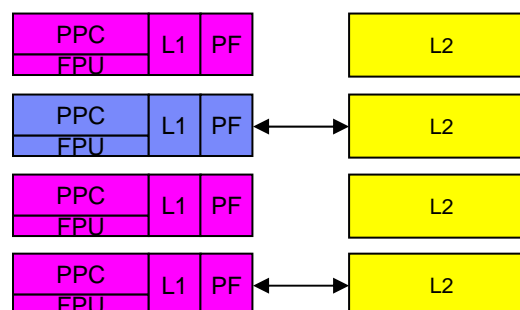
Atomic Operations (lwarx stx on PowerPC)

- N round trips
 - Where N is the number of threads
 - For N=16 2500 → ~2500 cycles, 32 → 5000, 64 → 10000

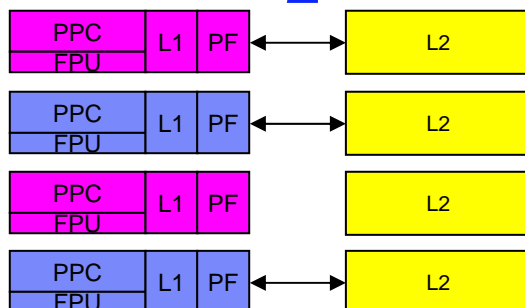
1



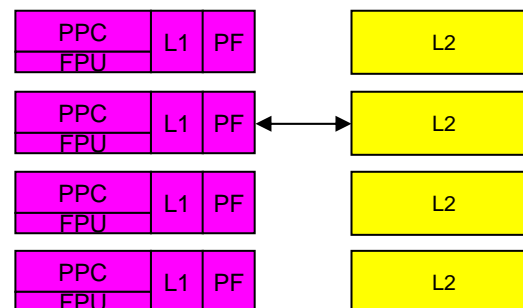
3



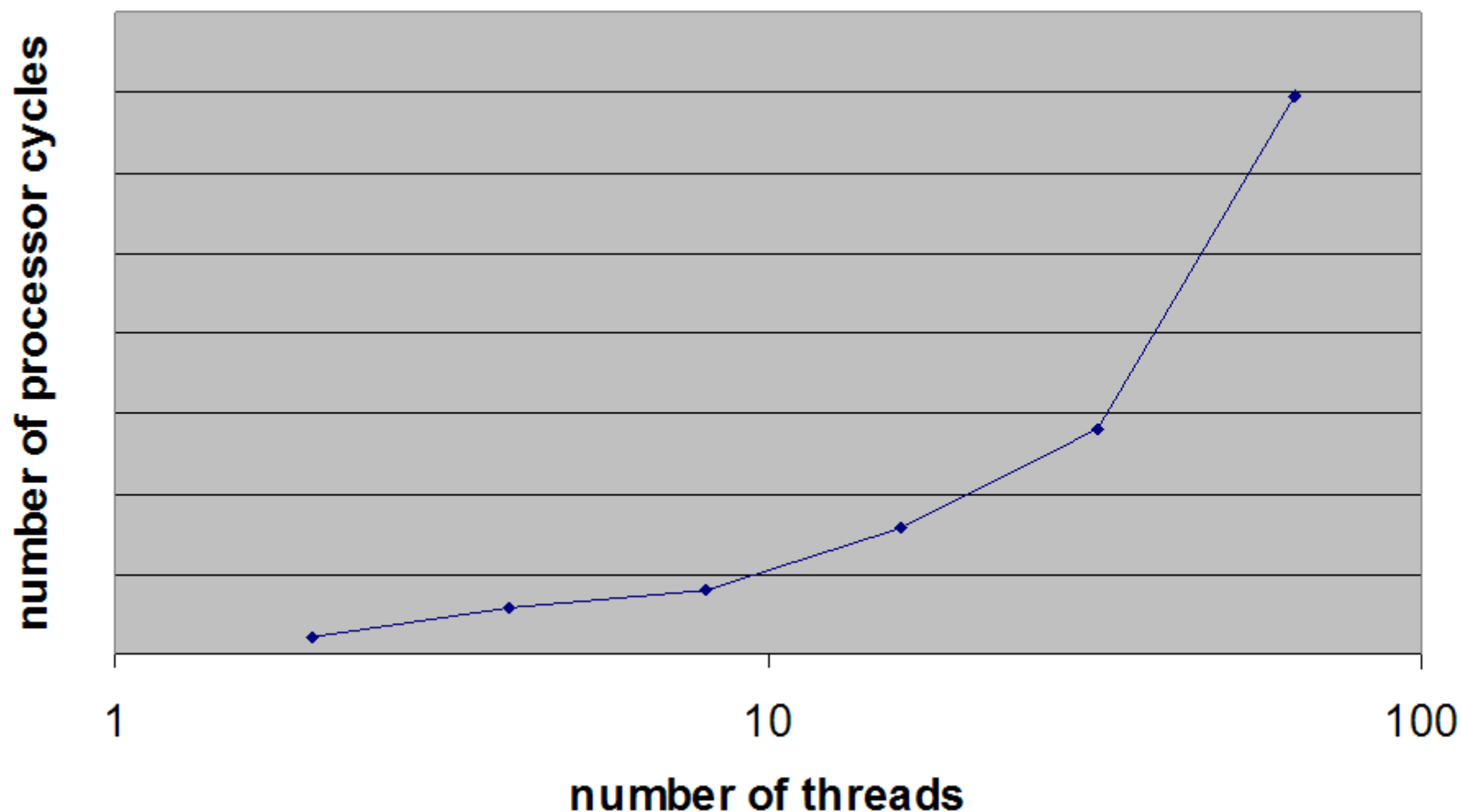
2



4



Time for N threads to synchronize on current generation demonstrating the need for improved scalable 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 `fetch_inc`
- 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 `store_dec`
- Applied within the IBM Hardware, System Software, and Application teams
 - Needs to be applied across the community

Going Forward

- **Hardware trends are posing challenges to HPC software**
 - **Co-design important to handle hardware-trend induced challenges**
- **Progress has been made in the software stack**
 - **More progress needs to be made in the software stack**
- **There are paths forward from where we are to exascale**
 - **Investment needs to be made soon to get there by goal date**
- **Overall evolution with strategically targeted revolution**

Exascale is Hard but Achievable

- There should be no cliff
 - Maybe a few drop-offs
- Investment is needed
 - Early now lead-time key
- Evolution
 - With strategic revolution

