Argonne NATIONAL LABORATORY

Exascale Software Center

Pete Beckman

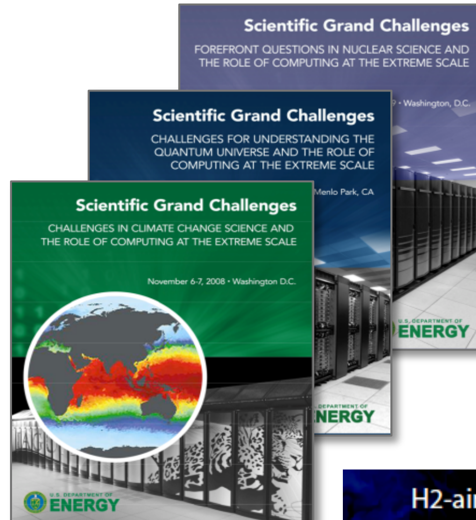
Director

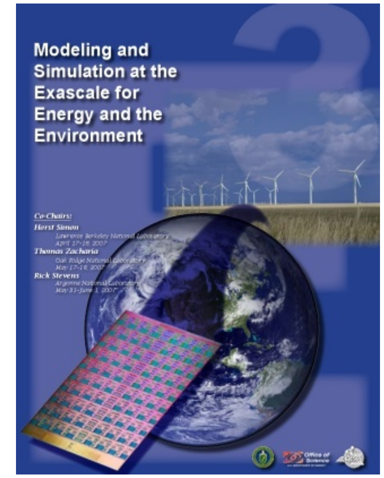
Exascale Technology and Computing Institute

Argonne National Laboratory

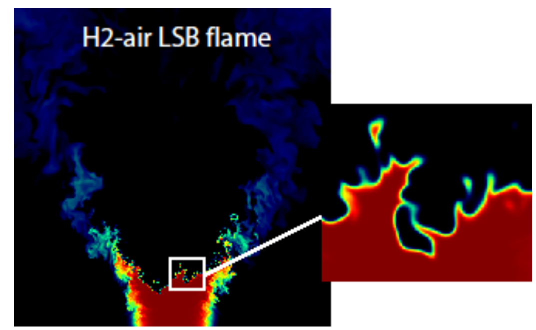
U.S. DEPARTMENT OF ENERGY

Exascale Applications and Technology





- Town Hall Meetings April-June 2007
- Scientific Grand Challenges Workshops November 2008 – October 2009
  - Climate Science (11/08),
  - High Energy Physics (12/08),
  - Nuclear Physics (1/09),
  - Fusion Energy (3/09),
  - Nuclear Energy (5/09),
  - Biology (8/09),
  - Material Science and Chemistry (8/09),
  - National Security (10/09) (with NNSA)
- Cross-cutting workshops
  - Architecture and Technology (12/09)
  - Architecture, Applied Math and CS (2/10)
- Meetings with industry (8/09, 11/09)
- External Panels
  - ASCAC Exascale Charge (FACA)
  - **Trivelpiece Panel**



"The key finding of the Panel is that there are compelling needs for exascale computing capability to support the DOE's missions in energy, national security, fundamental sciences, and the environment. The DOE has the necessary assets to initiate a program that would accelerate the development of such capability to meet its own needs and by so doing benefit other national interests. Failure to initiate an exascale program could lead to a loss of U. S. competitiveness in several critical technologies."

Trivelpiece Panel Report, January, 2010

Jack Dongarra
Pete Beckman
Terry Moore
Jean-Claude Andre
Jean-Yves Berthou
Taisuke Boku
Franck Cappello
Barbara Chapman
Xuebin Chi
Alok Choudhary
Sudip Dosanjh
Al Geist
Bill Gropp
Robert Harrison
Mark Hereld
Michael Heroux
Adolfy Hoisie
Koh Hotta
Yutaka Ishikawa
Fred Johnson
Sanjay Kale
Richard Kenway
Bill Kramer
Jesus Labarta
Bob Lucas
Barney Maccabe
Satoshi Matsuoka
Paul Messina
Bernd Mohr
Matthias Mueller
Wolfgang Nagel
Hiroshi Nakashima
Michael E. Papka
Dan Reed
Mitsuhisa Sato
Ed Seidel
John Shalf
David Skinner
Thomas Sterling
Rick Stevens
William Tang
John Taylor
Rajeev Thakur
Anne Trefethen
Marc Snir
Aad van der Steen
Fred Streitz
Bob Sugar
Shinji Sumimoto
Jeffrey Vetter
Robert Wisniewski
Kathy Yelick

SPONSORS

Office of Science
U.S. Department of Energy

NSF

ANR cea CEPBA CRAY EDF FUJITSU INRIA GENCI NVIDIA RIKEN 東京大学 The University of Tokyo 筑波大学

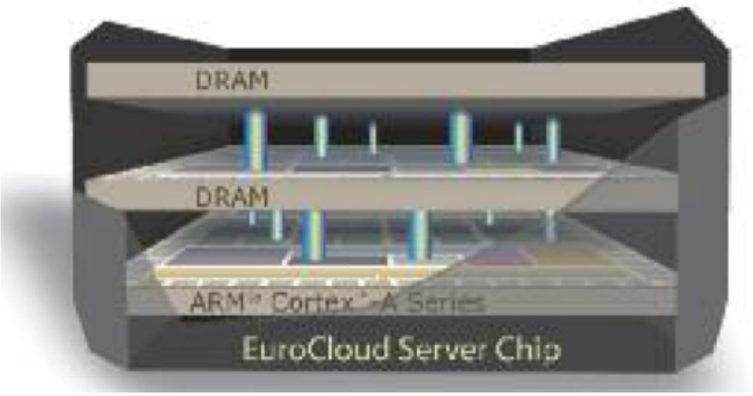
To be published in the January 2011 issue of The International Journal of High Performance Computing Applications

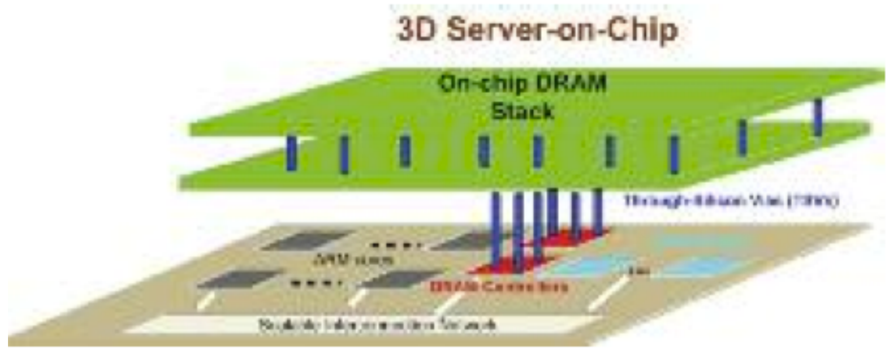
"We can only see a short distance ahead, but we can see plenty there that needs to be done."

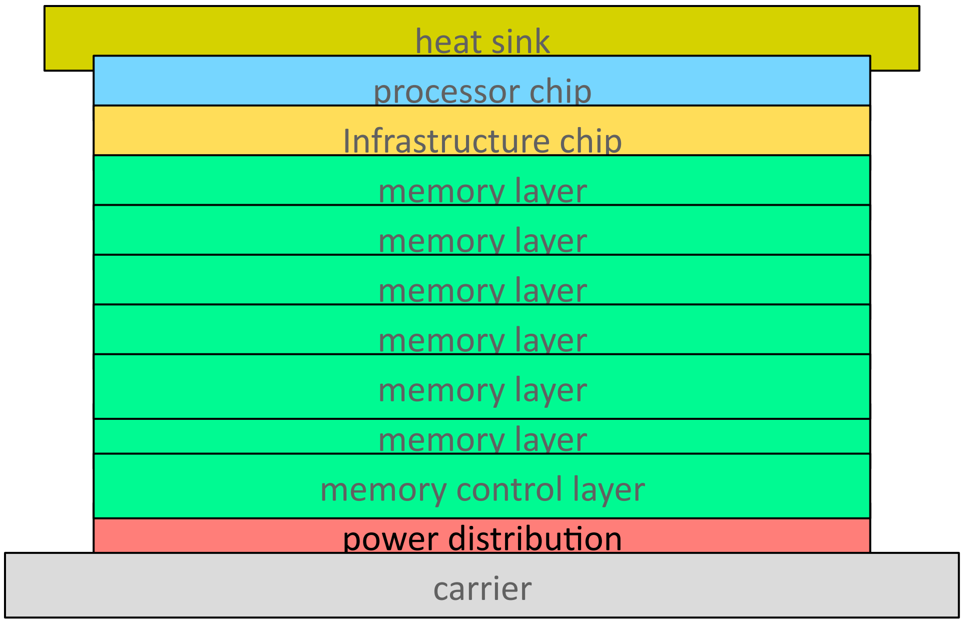
– *Alan Turing (1912 —1954)*

- www.exascale.org

Biggest Disruption: Node Architecture is Changing







- 100x – 1000x more cores
- Heterogeneous cores
- New programming model

- 3d stacked memory

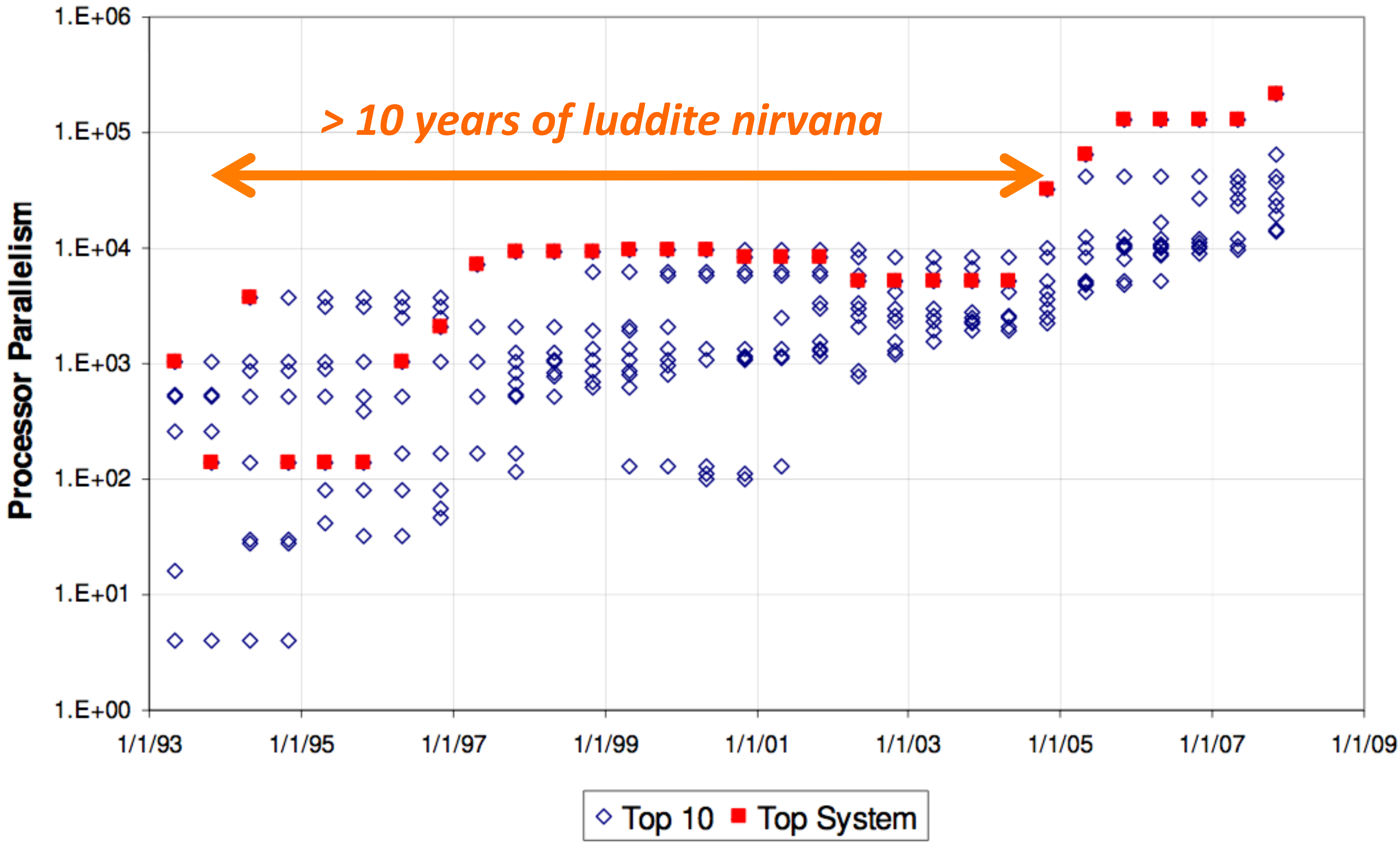
- Smart memory management

- Integration on package

COTS? No….

Potential System Architecture Targets

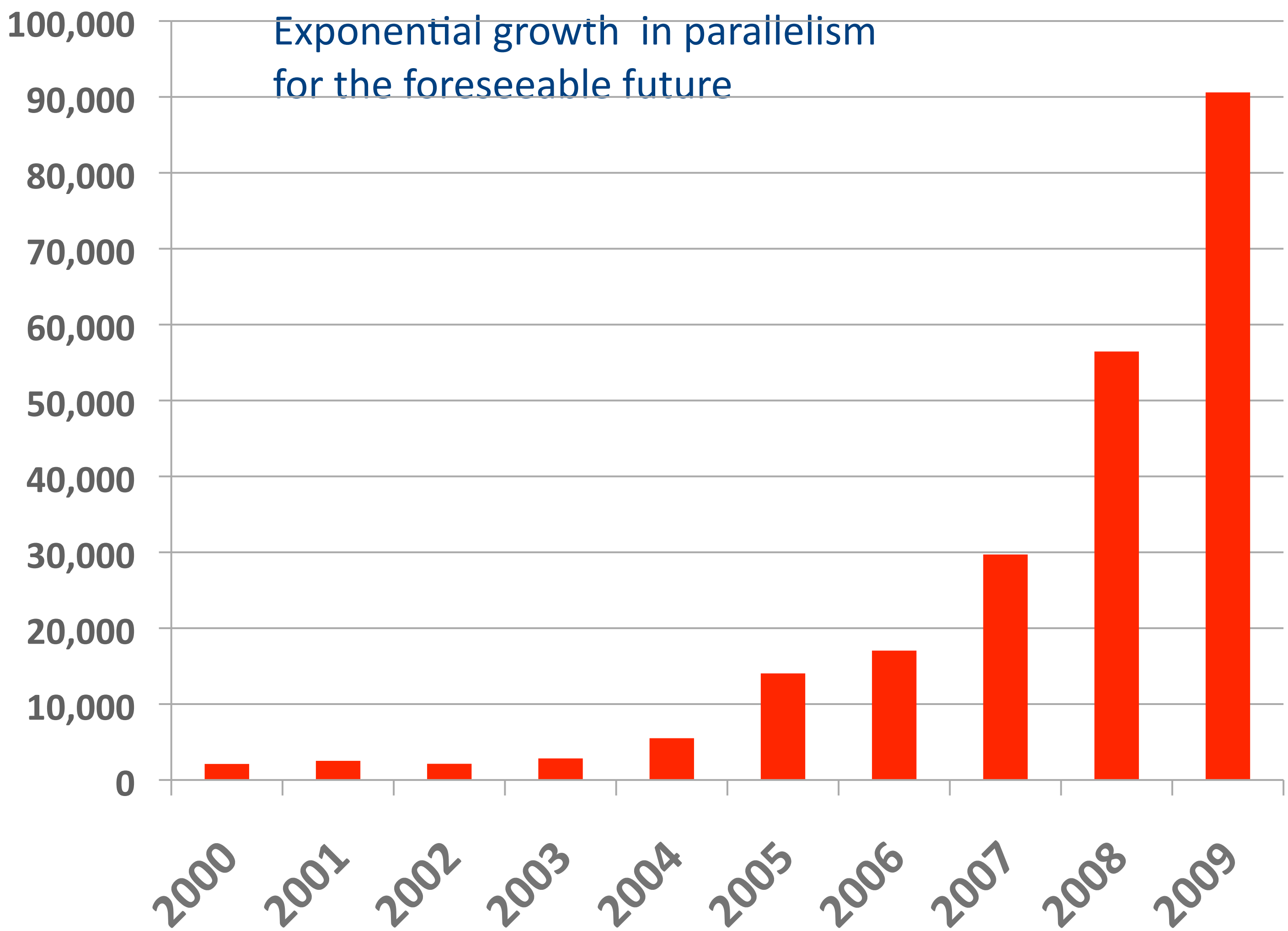
System attributes	2010	"2015"		"2018"		Difference Today & 2018
System peak	2 Pflop/s	200 Pflop/s		1 Eflop/sec		O(1000)
Power	6 MW	15 MW		~20 MW		
System memory	0.3 PB	5 PB		32-64 PB		O(100)
Node performance	125 GF	0.5 TF	7 TF	1 TF	10 TF	O(10) – O (100)
Node memory BW	25 GB/s	0.1 TB/sec	1 TB/sec	0.4 TB/sec	4 TB/sec	O(100)
Node concurrency	12	O(100)	O(1,000)	O(1,000)	O(10,000)	O(100) – O (1000)
Total Concurrency	225,000	$O(10^8)$		$O(10^9)$		O(10,000)
Total Node Interconnect BW	1.5 GB/s	20 GB/sec		200 GB/sec		O(100)
MTTI	days	O(1day)		O(1 day)		- O(10)

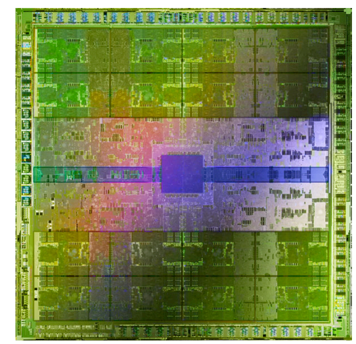


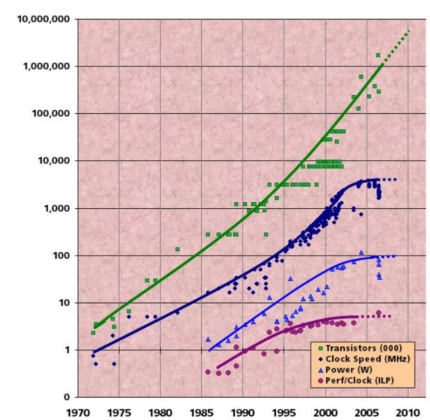
Source: DARPA Exascale Report

Average Number of Cores Per Supercomputer

Top20 of the Top500







"It took a decade to be able to efficiently utilize a 10X increase in processor parallelism, to expect that 1000X can be handled in less than that is a long stretch"

A Thought Experiment

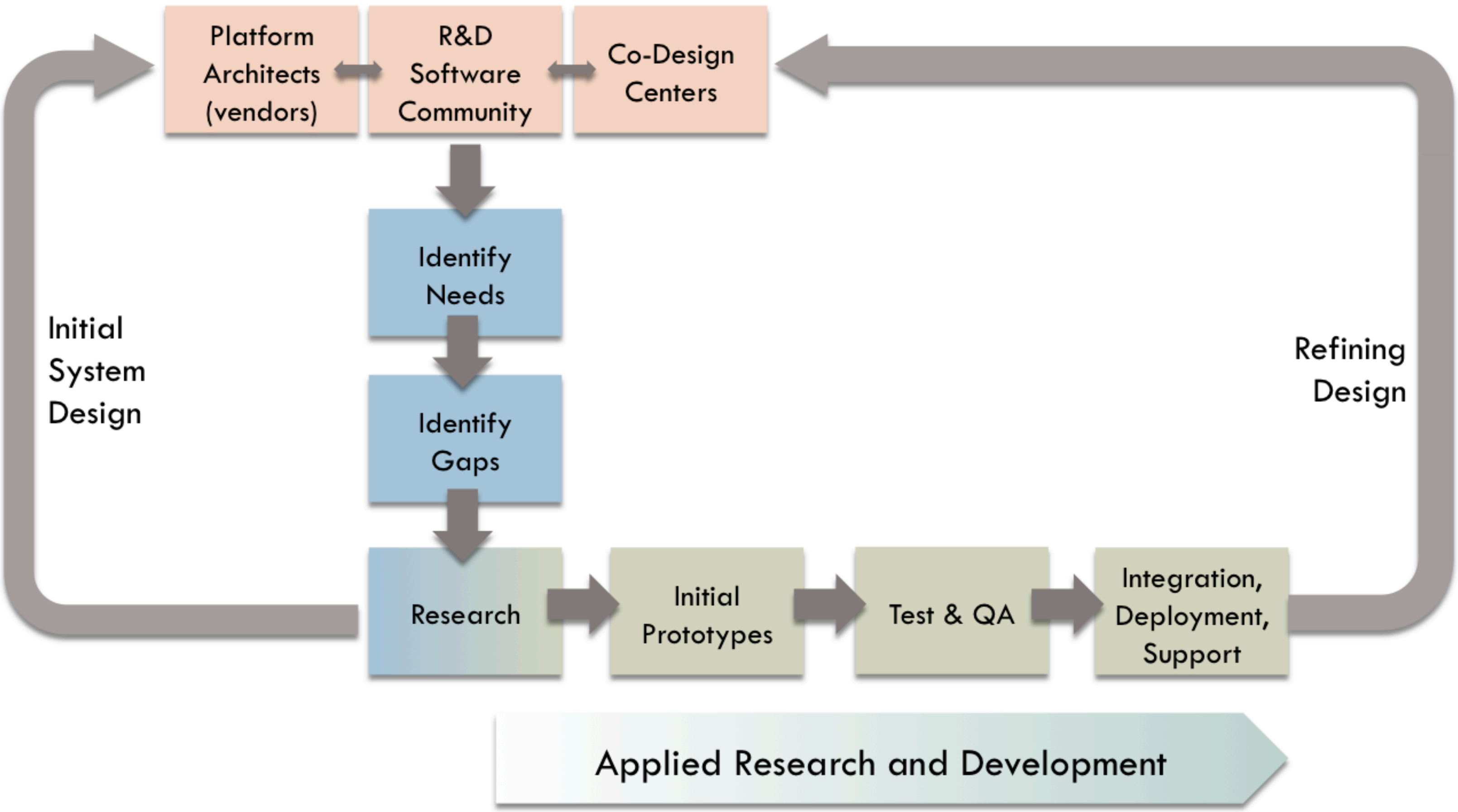
lim

cores → ∞

power → ∞

mem/core → 0

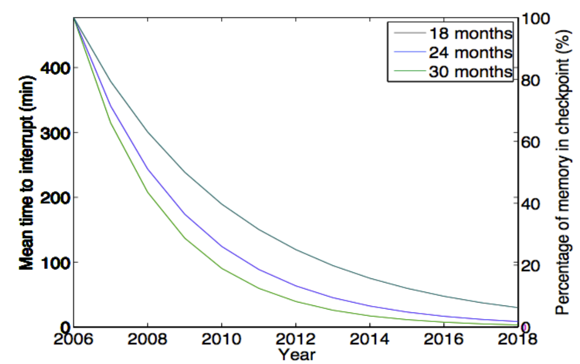
Co-design and System Software



Example: The Big Challenges for System Software

- Parallelism
  - Programming Model
  - MPI + ?
- System scale
  - Operating system and run-time
  - Communications and run-time libraries
  - I/O and file systems
  - Tools, math libraries,
- Fault management
  - Coordination across components
  - Programming model issues? TBD…
- Power management
  - Never to be managed by user, but smarter system software

"With petascale computers only a year or two away there is a pressing need to anticipate and compensate …"



Programming at Exascale
What do we expect from the Message Layer?

- Systems with the largest core counts in June 2010 Top500 list

Jülich BG/P	294,912 cores
Oak Ridge Cray XT5	224,162 cores
LLNL BG/L	212,992 cores
Argonne BG/P	163,840 cores
LLNL BG/P (Dawn)	147,456 cores

- MPI already runs successfully on these systems
- In a couple of years, we will have systems with more than a million cores
- MPI will need enhancements, but will keep scaling, provided we improve key parts
- On exascale, MPI is likely to be used as part of a hybrid programming model much more so than it is today
  - MPI being used to communicate between "address spaces"
  - With some other "shared-memory" programming model (OpenMP, UPC, CUDA, OpenCL) for programming within an address space
- How can MPI support efficient "hybrid" programming on exascale systems?

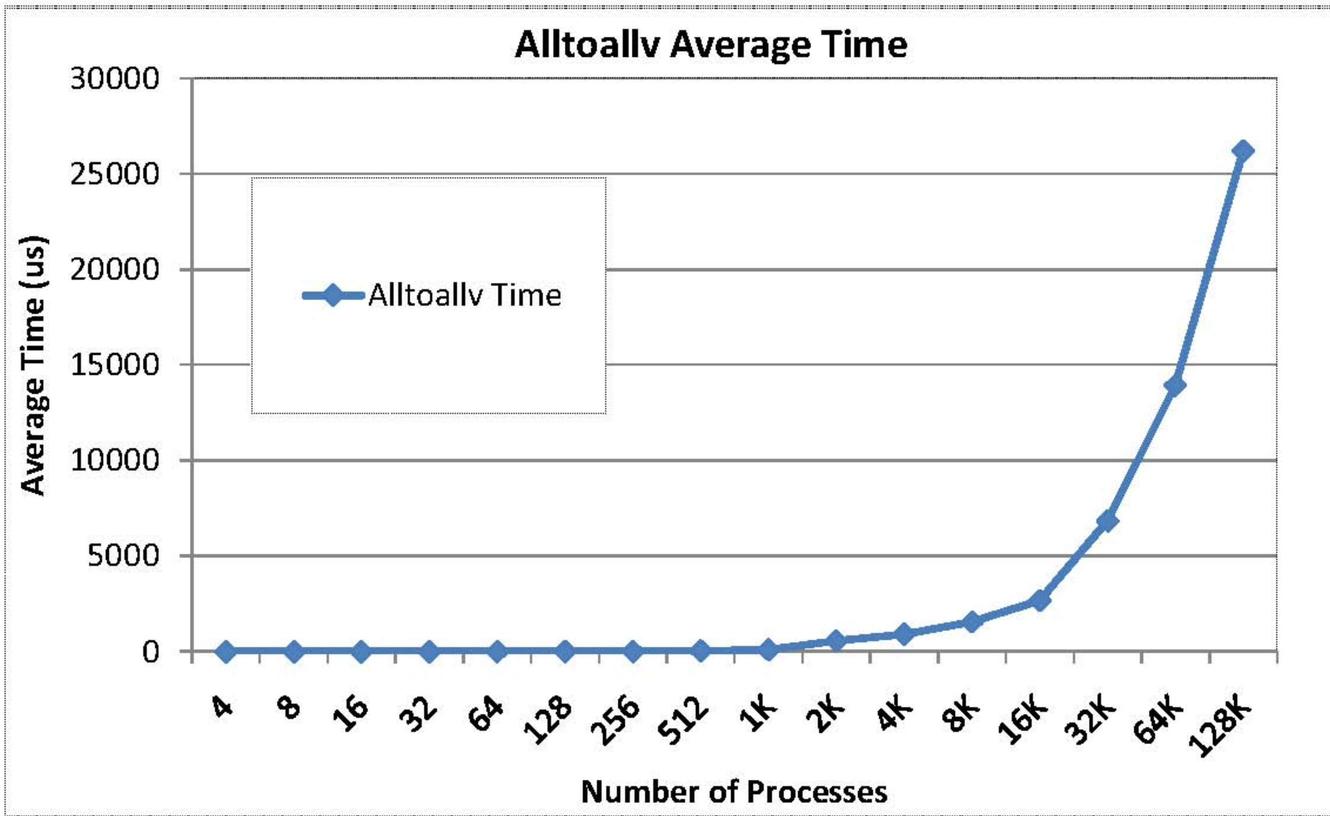
Scaling MPI to Exascale

- Although the original designers of MPI were not thinking of exascale, MPI was always intended and designed with scalability in mind. For example:
  - A design goal was to enable implementations that maintain very little global state per process
  - Another design goal was to require very little memory management within MPI (all memory for communication can be in user space)
  - MPI defines many operations as *collective* (called by a group of processes), which enables scalable efficient implementations
- Nonetheless, some parts of the MPI specification may need to be fixed for exascale
  - Being addressed by the MPI Forum in MPI-3

Factors Affecting MPI Scalability

- ***A nonscalable MPI function is one whose time or memory consumption per process increase linearly (or worse) with the total number of processes***
- For example
  - If memory consumption of MPI_Comm_dup increases linearly with the no. of processes, it is not scalable
  - If time taken by MPI_Comm_spawn increases linearly or more with the no. of processes being spawned, it indicates a nonscalable implementation of the function
- The goal should be to use constructs that require only constant space per process

Zero-byte MPI_Alltoallv time on BG/P



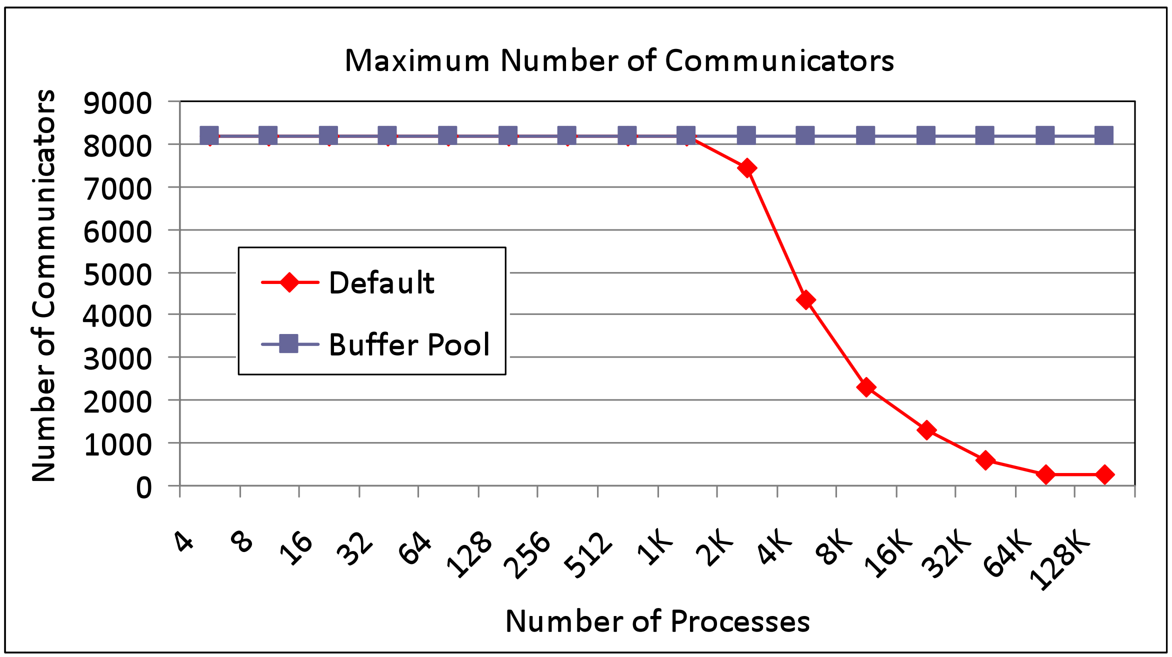
- This is just the time to scan the parameter array to determine it is all 0 bytes. No communication performed.

Other Issues in the MPI Specification

- Graph Topology
  - In MPI 2.1 and earlier, requires the entire graph to be specified on each process
  - Already fixed in MPI 2.2 – new distributed graph topology functions
  - But existing applications must switch to the new interface
- One-sided communication
  - Synchronization functions turn out to be expensive
  - Being addressed by RMA working group of MPI-3
- Representation of process ranks
  - Explicit representation of process ranks in some functions, such as MPI_Group_incl and MPI_Group_excl
  - Concise representations should be considered
- Fault tolerance…

Communicator Memory Consumption Fixed

- Looking at the source code, we found that IBM's MPI really only needed one buffer per thread instead of one buffer per new communicator
- Since there are only four threads on the BG/P, we fixed the problem by allocating a fixed buffer pool within MPI
- We provided IBM with a patch that fixed the problem and enabled NEK5000 to run at full scale



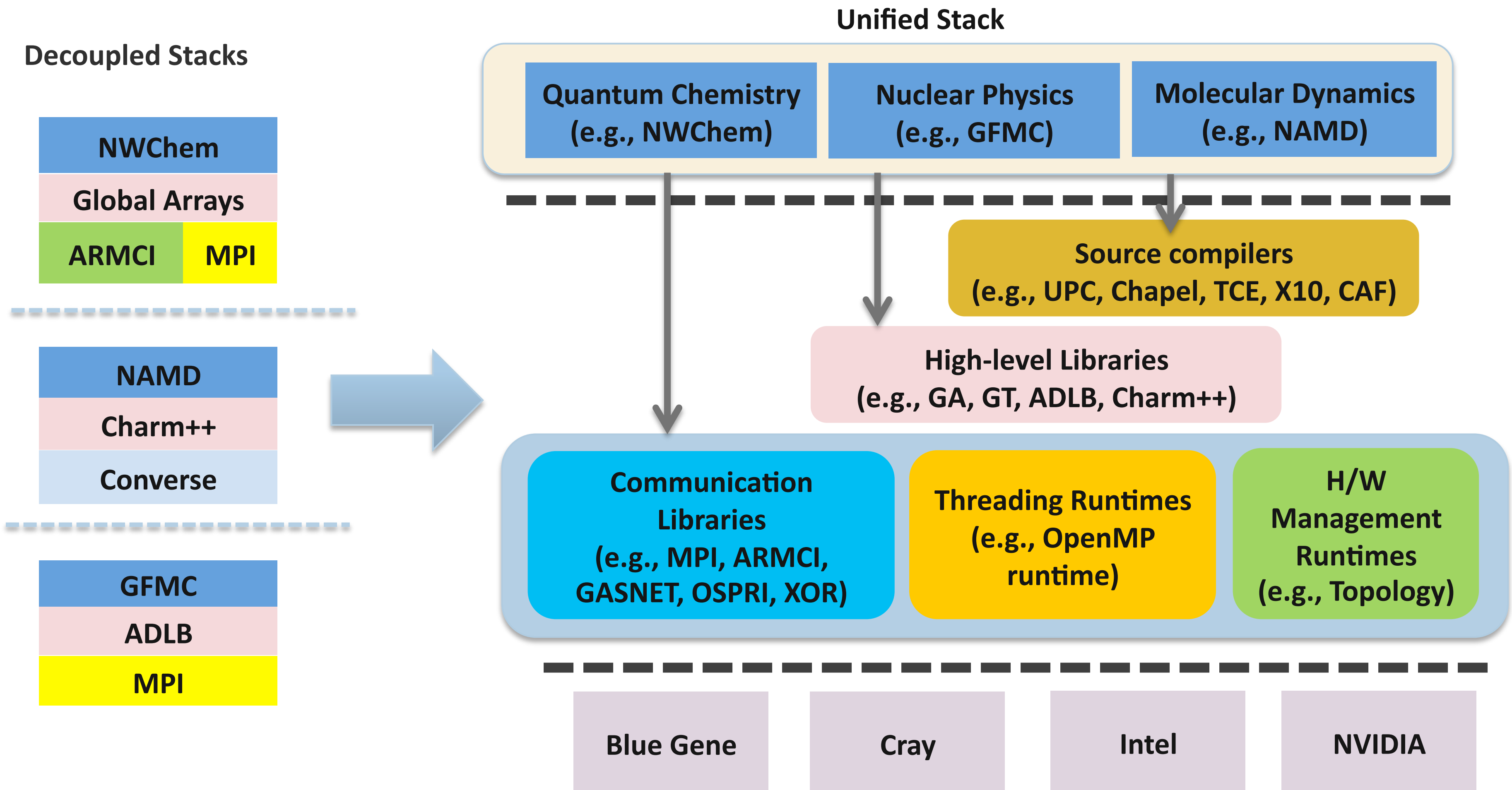
The Problem and the Fix

- MPI_Comm_split does an allgather of the colors and keys from all processes, followed by a local sort of the keys for the same color
- In the case where all ranks pass the same color, the data set to be sorted is of size *p*
- The local sort used a simple bubble sort algorithm, which is O(p^2)
  - The code did have a FIXME comment acknowledging this
- Simply switching the local sort to use quicksort, which is O(*plgp*), fixed the problem

	OLD	NEW
16,384 procs	1.5 sec	0.105 sec
32,768 procs	6.3 sec	0.126 sec
65,536 procs	25.3 sec	0.168 sec
131,072 procs	101.2 sec	0.255 sec

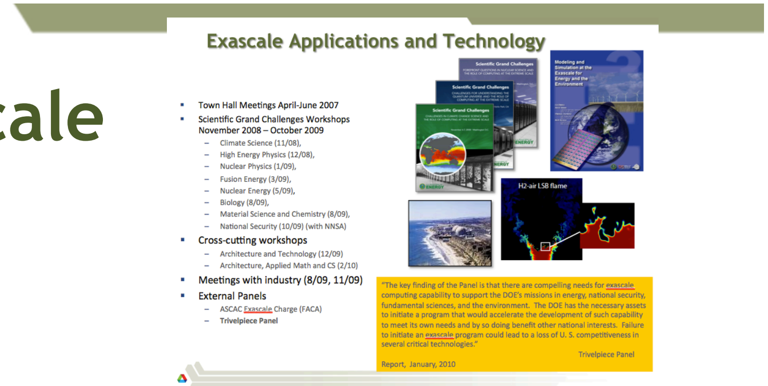
- At this scale, there is a big difference between p^2 and *plgp*!

Programming Models and Runtime Systems



The key is to provide a unified architecture with multiple levels of capabilities and ALLOW APPLICATIONS TO BREAK THE LAYERING→ transition path for applications!

Context: DOE Planning for Exascale



Platforms
- Systems: 2015
- Systems: 2018

Cross-cutting Technologies

Co-Design Application Teams

Exascale Software

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

Current HPC Software Ecosystem is Chaotic

- Software development uncoordinated with hardware features
  - (e.g., power mgmt, multicore tools, math libraries, advanced memory models)
- No global evaluation of key missing components
- 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
  - Counterexample: A models that work: partnerships with vendors
- Community codes unprepared for sea change in architectures
- Coordination via contract is poor and only involves 2 parties

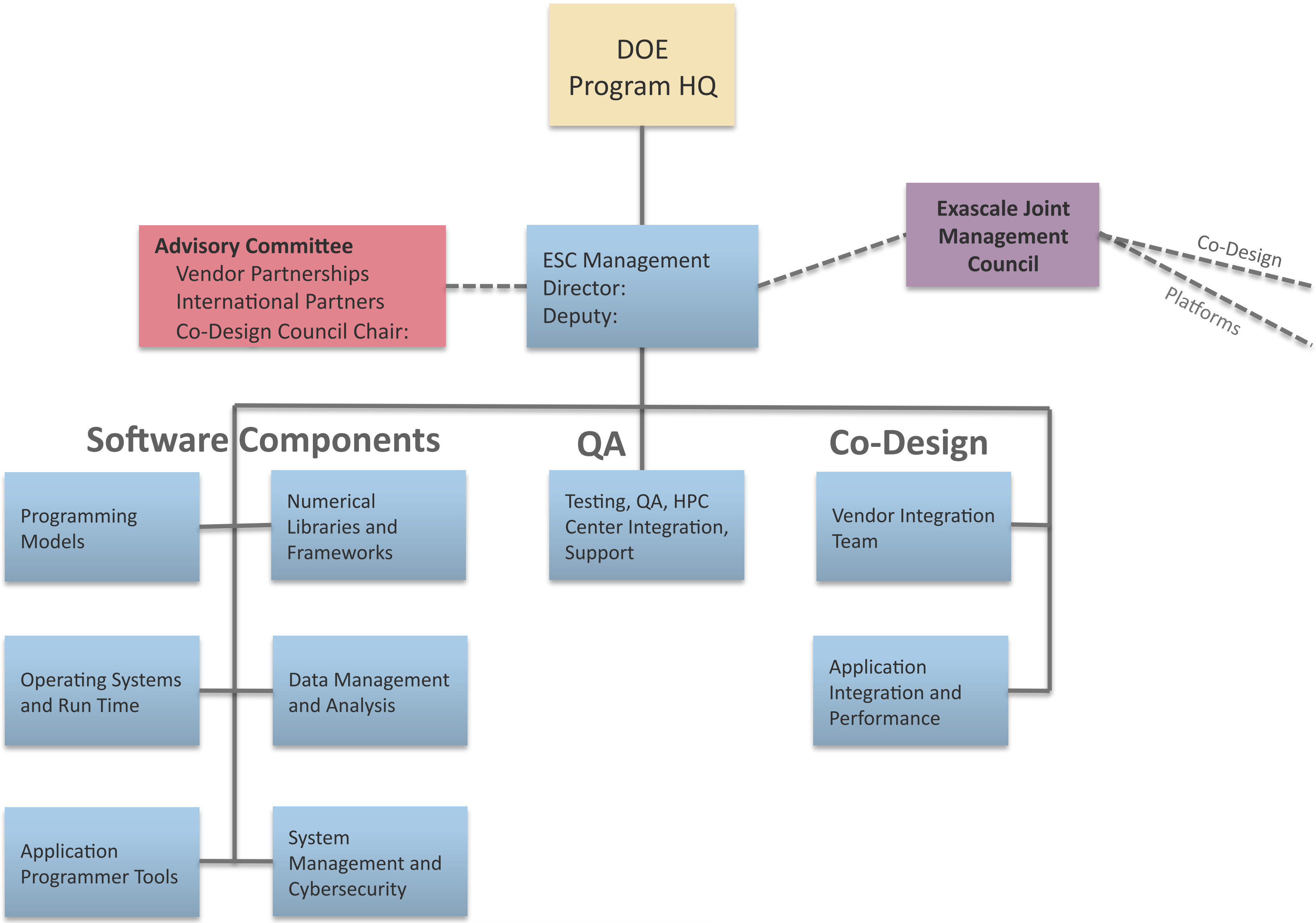
Exascale Software Center (ESC)

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

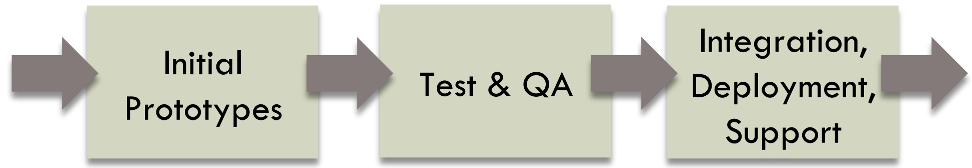
Ultimately responsible for success of software

- 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

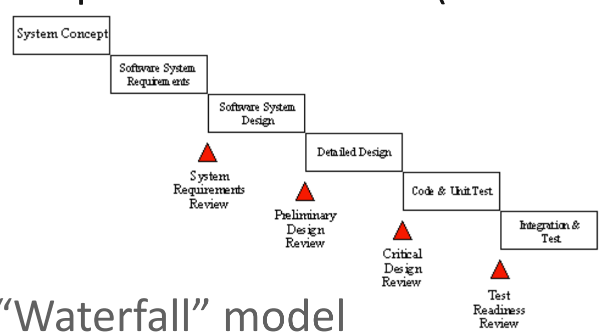
ESC Organization Chart



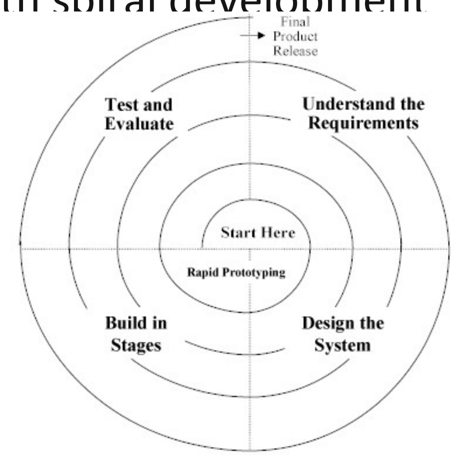
ESC Software Development



- Successful applied R&D teams are built around clear goal of delivering working, supported packages
- Need balanced risk portfolio… evolution and revolution
- 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

Example Component Evaluation Criteria

- Criticality
  - Base component already deployed on petascale systems?
  - Required for new architecture feature or to address key unresolved petascale issue?
  - Is an "exa-only" problem, component not otherwise ready?
  - Already in use, or planned use by key applications?
  - Requires important vendor integration activities?
- Technical Risk
  - Is path to scaling/extending component well understood?
  - Do existing examples demonstrate feasibility?
  - Can development be done incrementally, or is whole functionality required for success?
  - Could multiple packages provide functionality?
- Project Team History
  - How long as team been delivering related software?
  - Have they demonstrated successful software engineering discipline?
  - Is applied research and support for developed software part of their ethos?
- Management and Institutional Support
  - Is the leadership invested in exascale?
  - Is the team part of a larger organization that provides support for applied development?
  - What is the institutional commitment to the ESC?

		Low Risk	Moderate Risk
ESC Supported	Important		
Vendor Supported	Critical		
	Most Critical		

Evolution & Revolution

- We can’t continue simple evolution
- We can’t reinvent everything
- Path forward is exciting!
  - Will leverage existing billions in HPC software and applications
  - Will encourage and reward disruptive change
    - Dynamic run-time systems and programming models
    - Billion-way parallelism with load balancing and graph execution models
    - Run-time code morphing
- Balance…