Future Node Architectures and their Implications for MPI

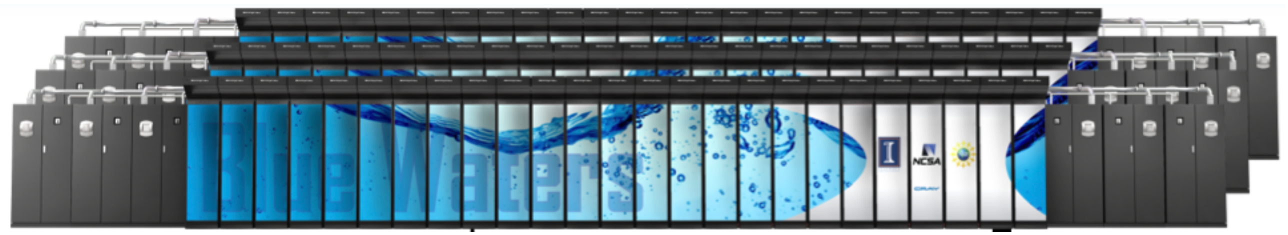
Rajeev Thakur

Deputy Director

Mathematics and Computer Science Division

Argonne National Laboratory

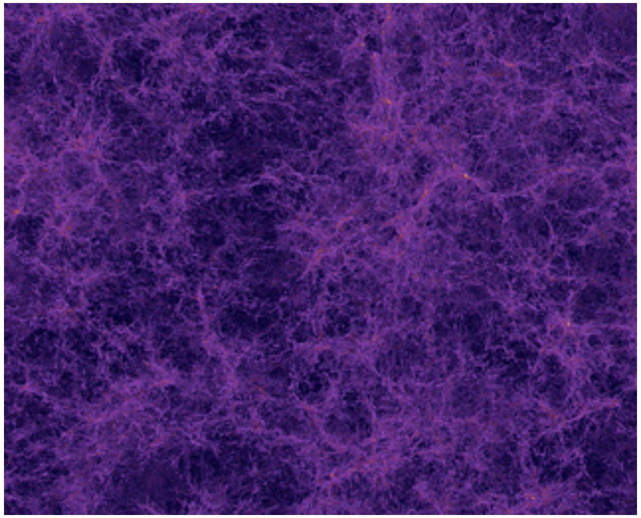
MPI Runs Successfully at Full Scale on the Largest Supercomputers of Today



A Few Application Successes

HACC Cosmology Code

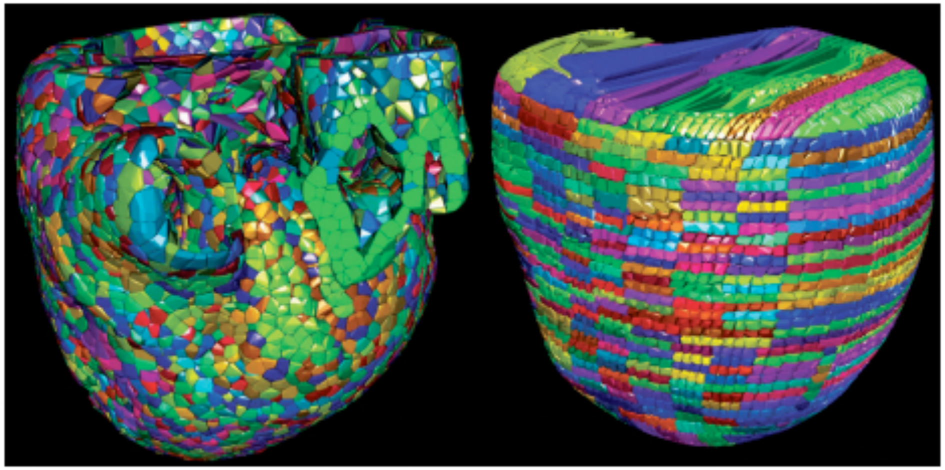
- HACC cosmology code from Argonne (Salman Habib) achieved **14 PFlops/s** on Sequoia (Blue Gene/Q at LLNL)
  - Ran on full Sequoia system using MPI + OpenMP hybrid
  - Used 16 MPI ranks * 4 OpenMP threads per rank on each node, which matches the architecture: 16 cores per node with 4 hardware threads each
  - **~ 6.3 million way concurrency: 1,572,864 MPI ranks * 4 threads/rank**
  - http://www.hpcwire.com/hpcwire/2012-11-29/sequoia_supercomputer_runs_cosmology_code_at_14_petaflops.html
  - SC12 Gordon Bell prize finalist



The HACC code has been used to run one of the largest cosmological simulations ever, with 1.1 trillion particles

Cardioid Cardiac Modeling Code

- Cardioid cardiac modeling code (IBM & LLNL) achieved **12 Pflops/s** on Sequoia
  - Models a beating human heart at near-cellular resolution
  - Ran at scale on full system (96 racks)
  - Used MPI + threads hybrid: 1 MPI rank per node and 64 threads
  - OpenMP was used for thread creation only; all other thread choreography and synchronization used custom code, not OpenMP pragmas
  - http://nnsa.energy.gov/mediaroom/pressreleases/sequoia112812
  - SC12 Gordon Bell Prize finalist



ROSS Parallel Discrete-Event Simulator

- ROSS parallel discrete-event simulator run with 7.8 million MPI ranks on a Sequoia/Vulcan combined system
  - PI: Chris Carothers, Rensselaer Polytechnic Institute
  - Used 96 racks of Sequoia + 24 racks Vulcan (a BG/Q system at LLNL for industrial collaborators)
  - 1,966,080 cores * 4 MPI ranks/core = 7,864,320 MPI ranks
  - Ran the PHOLD benchmark and achieved the highest event rate ever reported by a parallel discrete event simulation (504 billion events/sec)
  - Paper: "*Warp Speed: Executing Time Warp on 1,966,080 Cores*," PADS 2013
  - News: http://news.rpi.edu/luwakkey/3173?destination=node/40108

- And there are other applications running at similar scales…

Experiments with over 100 million MPI processes

- Prof. Alan Wagner's group at Univ. of British Columbia has run programs with over 100 million MPI processes (on 6,480 processor cores)
- They use a modified version of MPICH they develop called FG-MPI (Fine-Grain MPI) that implements MPI processes as coroutines rather than operating system processes
- https://www.westgrid.ca/westgrid_news/2013-01-14/ubc_researchers_use_westgrid_explore_exascale_computing
- Intel recently funded this effort for further optimizations
- The authors plan to integrate it into regular MPICH, so other derivatives can use it too

Future Node Architectures

Reference: CAL Report

Abstract Machine Models and Proxy Architectures for Exascale Computing

Rev 1.1

J.A. Ang[1], R.F. Barrett[1], R.E. Benner[1], D. Burke[2],
C. Chan[2], D. Donofrio[2], S.D. Hammond[1],
K.S. Hemmert[1], S.M. Kelly[1], H. Le[1], V.J. Leung[1],
D.R. Resnick[1], A.F. Rodrigues[1],
J. Shalf[2], D. Stark[1], D. Unat[2], N.J. Wright[2]

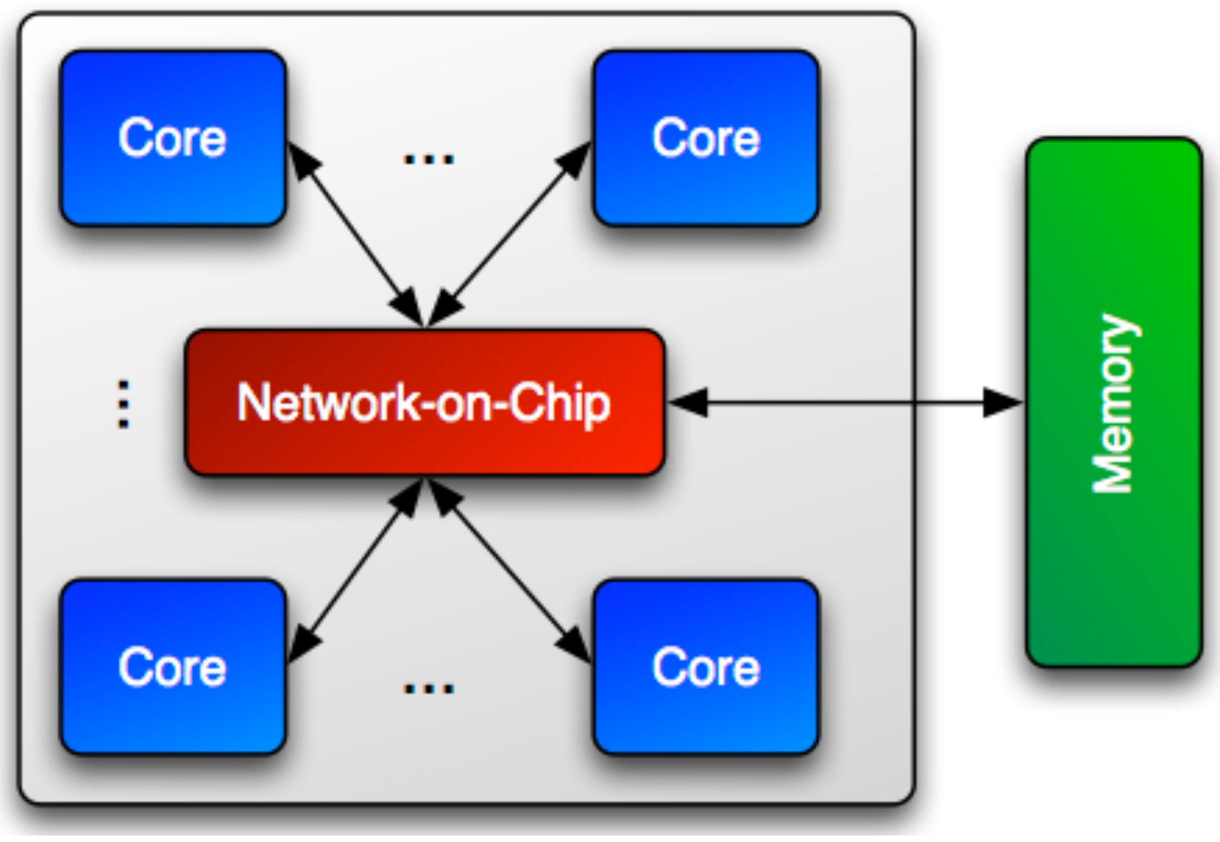
COMPUTER ARCHITECTURE LABORATORY
EXASCALE DESIGN SPACE EXPLORATION

Sandia National Laboratories, NM[1]
Lawrence Berkeley National Laboratory, CA[2]

May, 16 2014

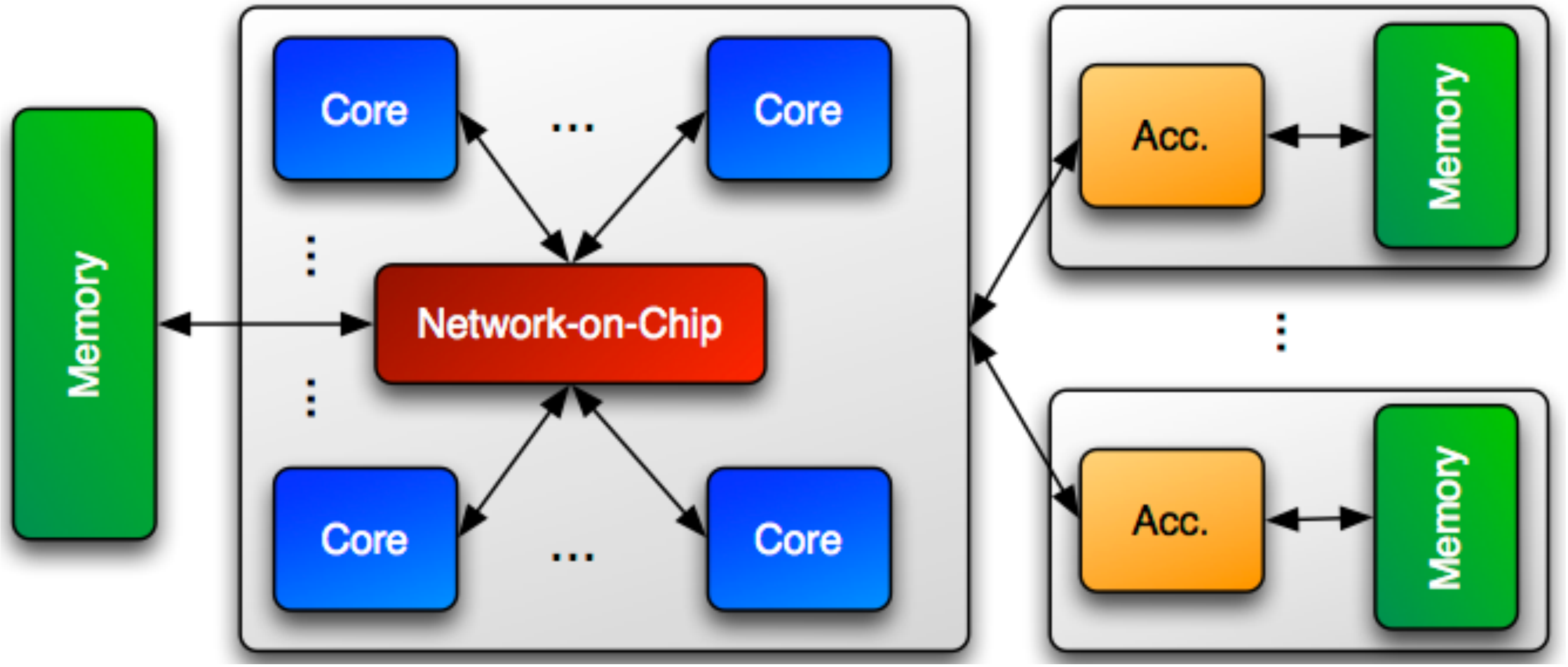
Available from http://www.cal-design.org/publications/publications2

Homogeneous Many-core Processor Model



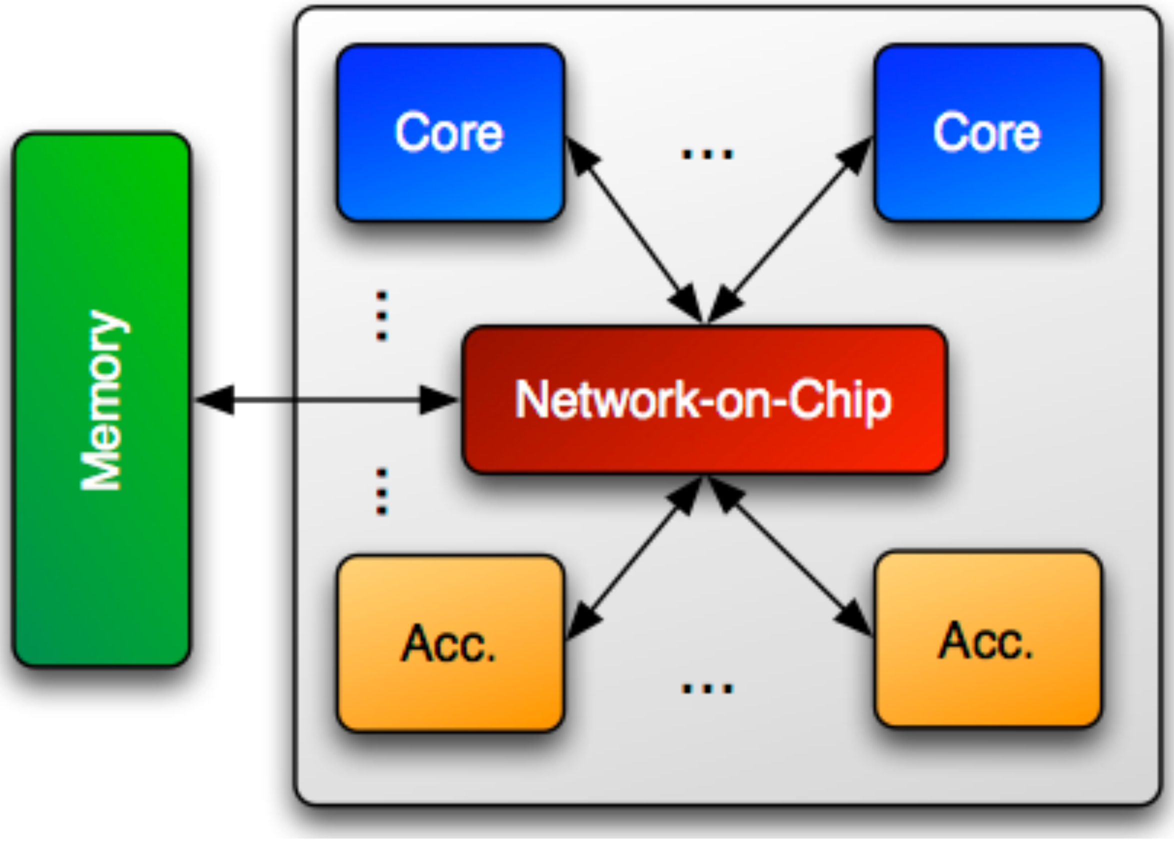
Source: CAL Report

Multicore CPU with Discrete Accelerators Model



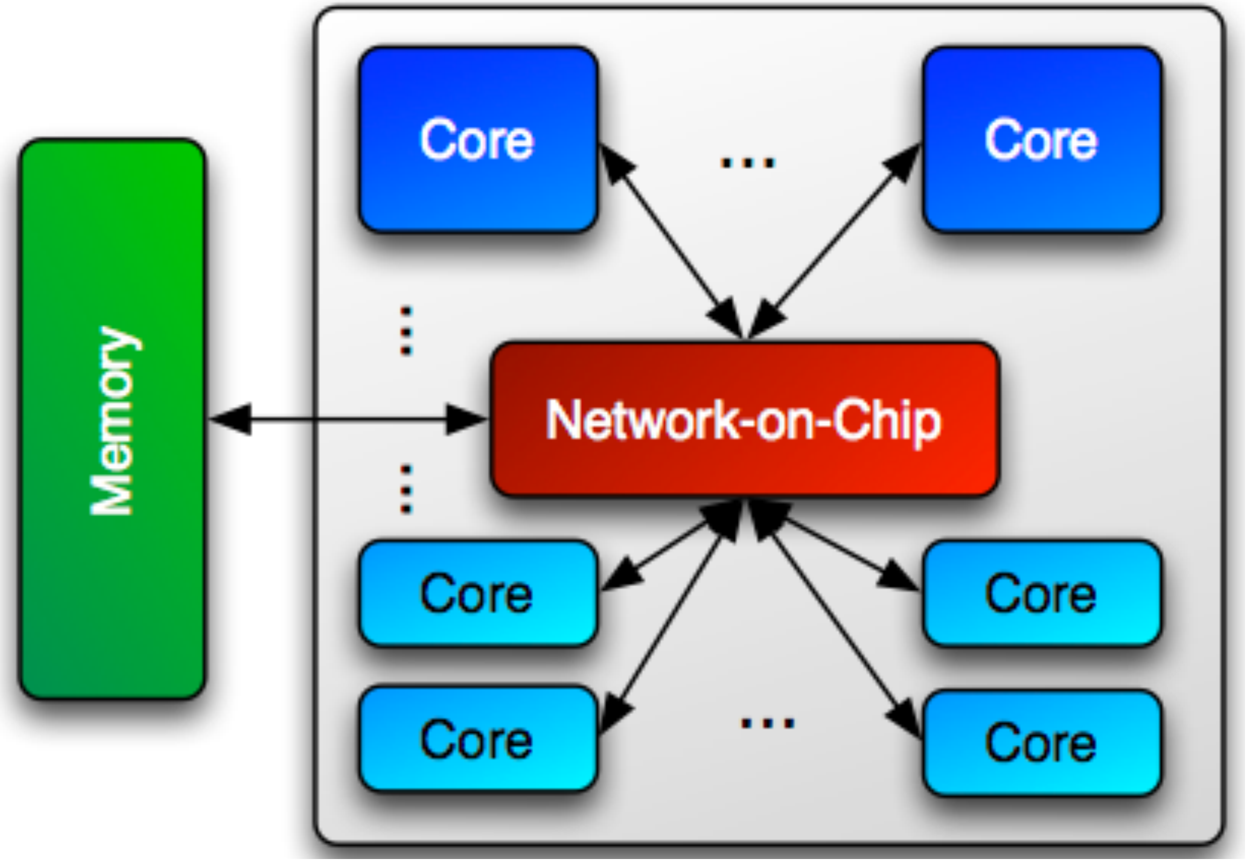
Source: CAL Report

Integrated CPU and Accelerators Model



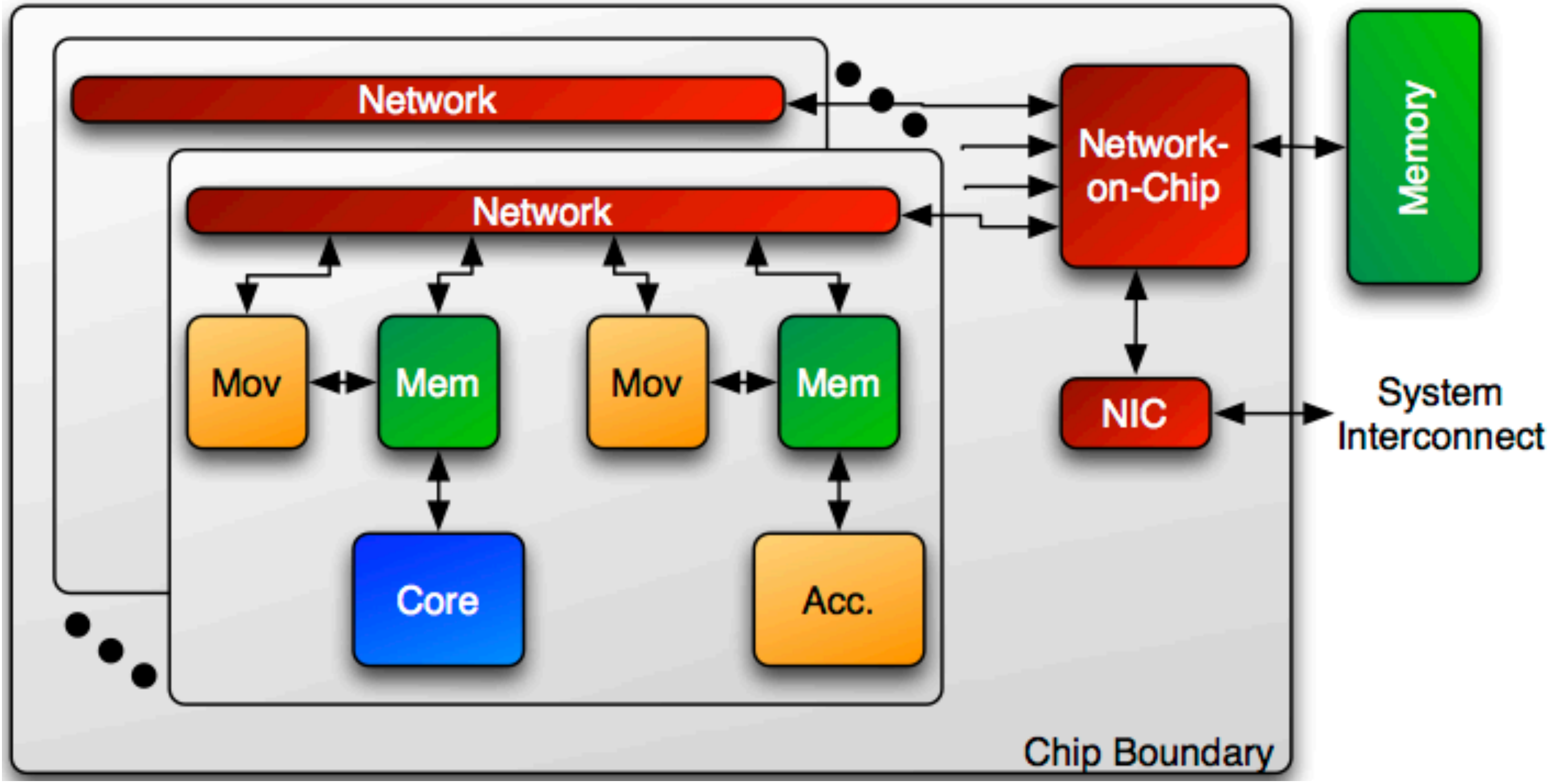
Source: CAL Report

Heterogeneous Multicore Model



Source: CAL Report

Performance-Flexible Multicore-Accelerator-Memory Model



Source: CAL Report

Design Parameters

	Processor Cores	GFlops/s per Proc Core	Accel Cores	Acc Count per Node	TFlops/s per Node	Node Count
Homogeneous M.C. Opt1	256	64	--	--	16	62,500
Homogeneous M.C. Opt2	64	250	--	--	16	62,500
Discrete Acc. Opt1	32	250	O(1000)	4	16C + 2A	55,000
Discrete Acc. Opt2	128	64	O(1000)	16	8C + 16A	41,000
Integrated Acc. Opt1	32	64	O(1000)	Integrated	30	33,000
Integrated Acc. Opt2	128	16	O(1000)	Integrated	30	33,000
Heterogeneous M.C. Opt1	16 / 192	250	--	--	16	62,500
Heterogeneous M.C. Opt2	32 / 128	64	--	--	16	62,500
Concept Opt1	128	50	128	Integrated	6	125,000
Concept Opt2	128	64	128	Integrated	8	125,000

Source: CAL Report

Cores Per Node and Total Number of Nodes

- For design purposes, let's use the maximum numbers of cores per node and total number of nodes from previous slide, and assume 8 hardware threads per core

Total number of nodes	125,000
Processor cores per node	256
Hardware threads per core	8
Hardware threads per node	2048
Total concurrency	256 million

Implications for MPI

Main Differences from Today's Largest Systems (for MPI)

Total number of nodes	O(125,000) => 1–4x
Hardware threads per node	O(2048) => 32–64x
Total concurrency	O(256 million) => 40–120x

For the Total Number of Nodes

- 125,000 nodes is not a problem
  - Sequoia already has 98,304 nodes
  - Sequoia and Vulcan combined together have 122,880 nodes and have been run as a single system (see slide 7)

- Scaling the number of nodes requires scaling in the distributed memory sense. We know what that requires and, in general, how to do it
  - Better, scalable collectives
  - Scalable data structures
  - Efficient RMA
  - Resilience
  - …

For the 32-64x Number of Hardware Threads/Node

- Needs MPI+X hybrid or even MPI+X+Y
  - Applications are already realizing this
- X/Y can be any of OpenMP, Pthreads, OpenACC, CUDA, …
- X can also be MPI as some people have recognized
  - MPI-3 has added support for shared-memory programming
  - See "**MPI+MPI: A New, Hybrid Approach to Parallel Programming with MPI Plus Shared Memory Computing**," Hoefler et al., *Computing,* 2013
- Needs better support from MPI for hybrid programming
  - Such as the "endpoints" proposal being discussed in the MPI Forum (see slide 23)

For the 40-120x Total Concurrency

- It should be considered as two sub-problems and solved accordingly
  - How many MPI processes
  - How many threads per MPI process
- For example, 256 million total concurrency could be implemented as
  - 16 million MPI processes and 16 threads per process, or
  - 4 million MPI processes and 64 threads per process
- Both cases are manageable by
  - Improving MPI implementations to use memory scalably (memory-efficient data structures)
  - Using MPI-3 shared-memory constructs and the new endpoints proposal for efficient hybrid programming
  - For a particular application, picking the best performing combination of (n x m), where n = number of MPI processes, m = threads per process
- The new fault tolerance proposal in MPI will help support the increased need for resilience (see slide 24)

Better Hybrid Programming: Extending MPI to Support Multiple Endpoints Per Process

- In MPI today, each process has a single communication endpoint (rank in MPI_COMM_WORLD)
- Multiple threads of a process communicate through that single endpoint, requiring the implementation to use locks etc., which are expensive
- MPI Forum is discussing a proposal (for MPI-4) that allows a process to have multiple endpoints
- Threads within a process can attach to different endpoints and communicate through those endpoints as if they are separate ranks
- The MPI implementation can avoid using locks if each thread communicates on a separate endpoint
- This allows the MPI standard to support "MPI + X" more efficiently without specifying what X is

Improved Support for Fault Tolerance

- MPI always had support for error handlers and allows implementations to return an error code and remain alive
- MPI Forum working on additional support for MPI-4
- Current proposal handles fail-stop process failures (not silent data corruption or Byzantine failures)
  - If a communication operation fails because the other process has failed, the function returns error code MPI_ERR_PROC_FAILED
  - User can call MPI_Comm_shrink to create a new communicator that excludes failed processes
  - Collective communication can be performed on the new communicator
  - Lots of other details in the proposal…

What's New in MPI-3 (Released Sept 2012)

- Many enhancements for scalability, such as distributed graph topologies, support for symmetric memory allocation in RMA, nonblocking collectives
- Major improvements to one-sided communication, including
  - Atomic operations, such as compare-and-swap and fetch-and-add
  - New memory model with simplified consistency semantics
  - Support for allocating and accessing shared memory within a node
- Nonblocking collectives
  - MPI_Ibcast, MPI_Ibarrier, MPI_Ireduce, and all other collectives
- Neighborhood collectives
  - Communication among nearest neighbors (e.g., stencil) can be expressed as a collective communication
- Extensive interface for tools to portably access performance variables, or set control variables, in an MPI implementation
- Bindings for Fortran 2008
- Many other miscellaneous items…

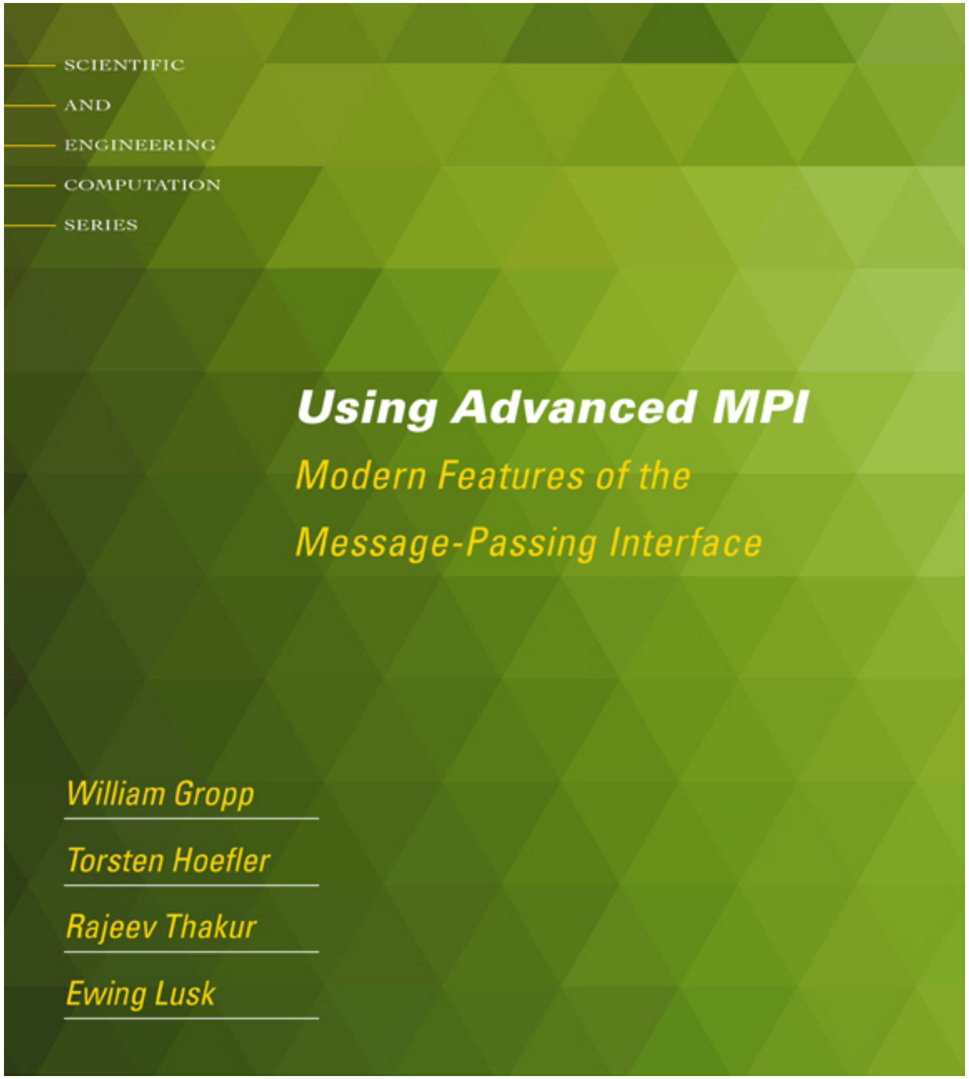
MPI-3 Implementations are already available

	MPICH	MVAPICH	Open MPI	Cray MPI	Tianhe MPI	Intel MPI	IBM BG/Q MPI [1]	IBM PE MPICH [2]	IBM Platform	SGI MPI	Fujitsu MPI	MS MPI
NB collectives	✔	✔	✔	✔	✔	✔	✔	Q4 '14	✔	✔	Q3 '14	
Neighborhood collectives	✔	✔	✔	✔	✔	✔	✔	Q4 '14	Q3 '15	✔	Q2 '15	
RMA	✔	✔	✔	✔	✔	✔	✔	Q4 '14	Q3 '15	✔	Q2 '15	
Shared memory	✔	✔	✔	✔	✔	✔	✔	Q4 '14	Q3 '15	✔	Q2 '15	
Tools Interface	✔	✔	✔	✔		✔	✔ [3]	Q4 '14	Q3 '15	✔	Q2 '15	
Non-collective comm. create	✔	✔	✔	✔	✔	✔	✔	Q4 '14	Q3 '15	✔	Q2 '15	
F08 Bindings	✔		✔	Q4 '14		Q4 '14	✔	Q4 '14	Q3 '15	Q3 '14	Q2 '15	
New Datatypes	✔	✔	✔	✔	✔	✔	✔	Q4 '14	Q3 '15	✔	Q2 '15	
Large Counts	✔	✔	✔	✔	✔	✔	✔	Q4 '14	Q3 '15	✔	Q2 '15	
Matched Probe	✔	✔	✔	✔	✔	✔	✔	Q4 '14	Q3 '15	✔	Q3 '14	

Release dates are estimates and are subject to change at any time.
Empty cells indicate no *publicly announced* plan to implement/support that feature.

[1] Open source, but unsupported   [2] Beta release   [3] No MPI_T variables exposed

New Tutorial Book on MPI-3 to be out by SC14



All four authors are in the room!