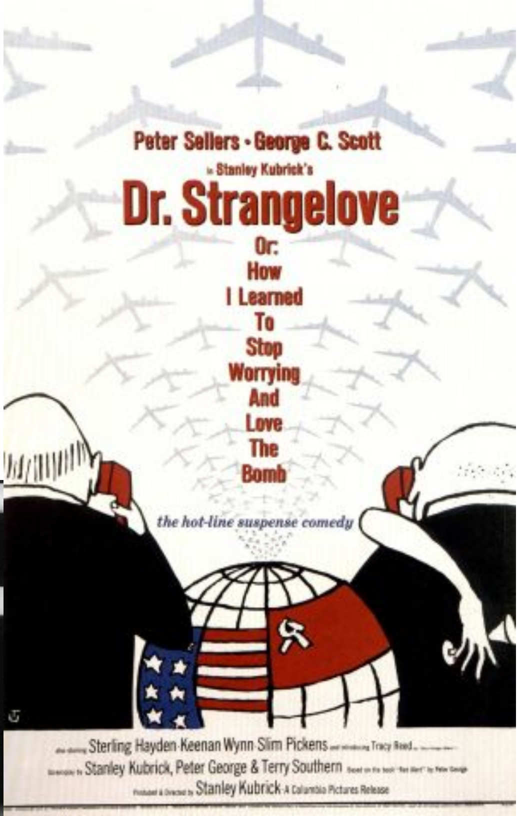
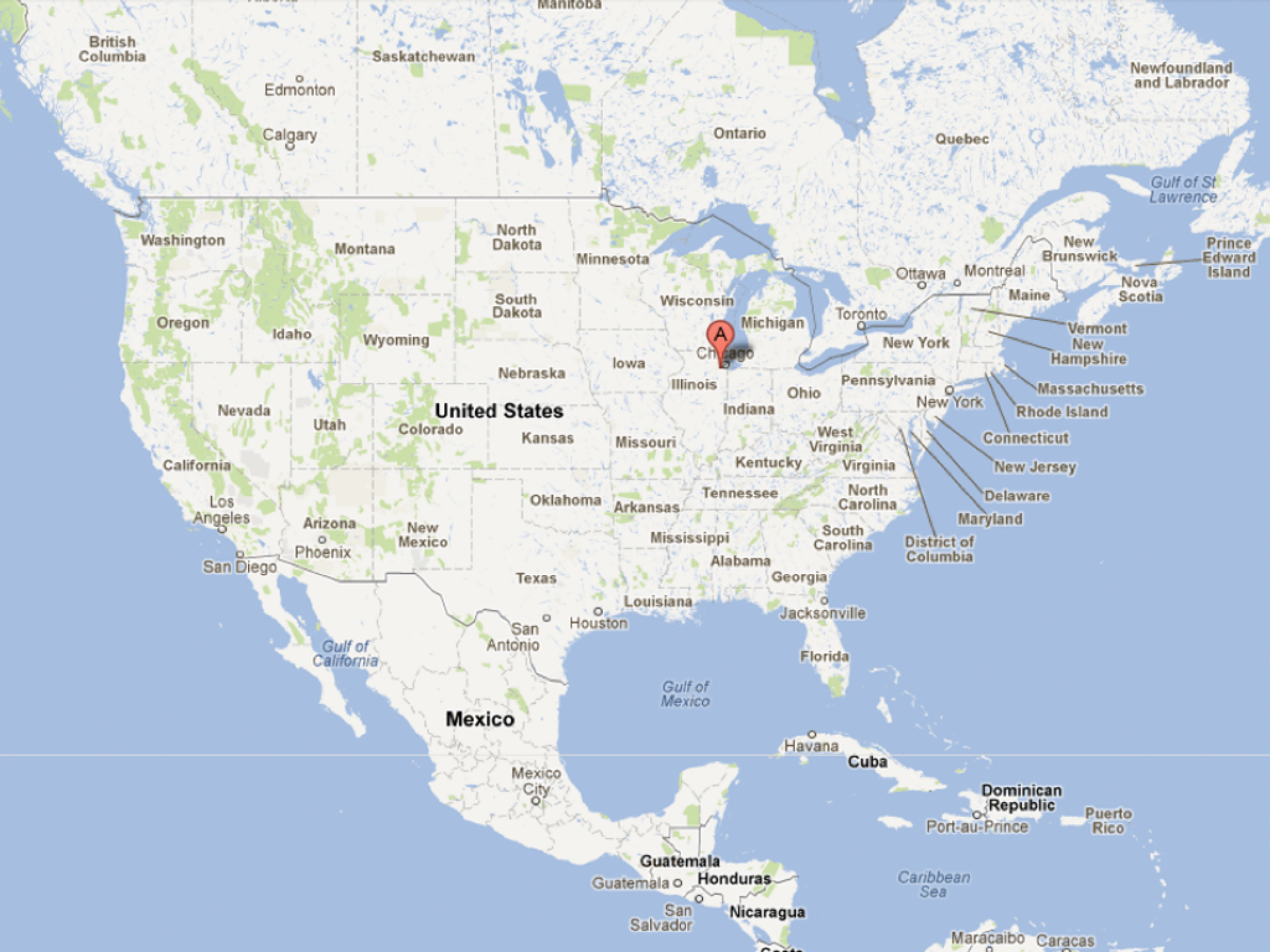
How I Learned to Stop Worrying about Exascale and Love MPI

Pavan Balaji

Computer Scientist and Group Lead

Argonne National Laboratory


Peter Sellers · George C. Scott
in Stanley Kubrick's
Dr. Strangelove
Or: How I Learned To Stop Worrying And Love The Bomb
the hot-line suspense comedy
Sterling Hayden · Keenan Wynn · Slim Pickens
Stanley Kubrick, Peter George & Terry Southern
Stanley Kubrick · A Columbia Pictures Release


Manitoba
British Columbia
Saskatchewan
Edmonton
Calgary
Ontario
Quebec
Newfoundland and Labrador
Gulf of St Lawrence
Washington
Montana
North Dakota
Minnesota
New Brunswick
Prince Edward Island
Ottawa
Montreal
Nova Scotia
Maine
South Dakota
Wisconsin
Oregon
Idaho
Wyoming
Toronto
Michigan
Vermont
A
Chicago
New York
New Hampshire
Iowa
Nebraska
Illinois
Pennsylvania
Ohio
Massachusetts
Nevada
Utah
United States
Colorado
Indiana
New York
Rhode Island
Kansas
Missouri
West Virginia
Connecticut
California
Kentucky
Virginia
New Jersey
Oklahoma
Tennessee
North Carolina
Delaware
Los Angeles
Arizona
Arkansas
Maryland
New Mexico
Mississippi
South Carolina
District of Columbia
Phoenix
Alabama
San Diego
Texas
Georgia
Louisiana
Houston
Jacksonville
San Antonio
Gulf of California
Florida
Gulf of Mexico
Mexico
Havana
Cuba
Mexico City
Dominican Republic
Puerto Rico
Port-au-Prince
Guatemala
Guatemala
Honduras
Caribbean Sea
San Salvador
Nicaragua
Maracaibo
Caracas

Argonne National Laboratory

About Argonne

- $675M operating budget
- 3,200 employees
- 1,450 scientists and engineers
- 750 Ph.D.s

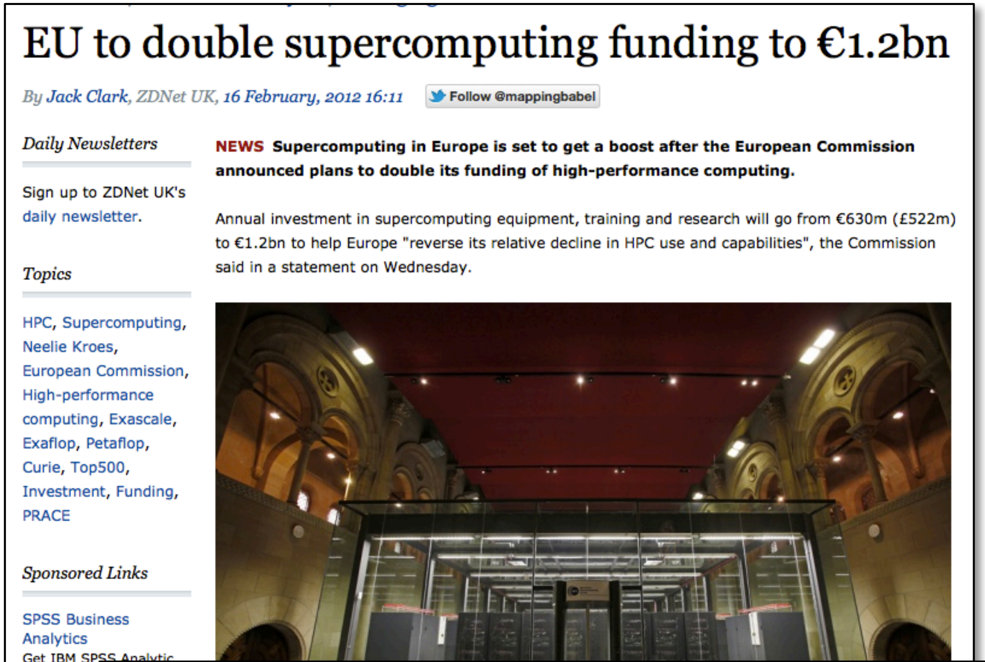
Argonne Research Highlights

- Sustainable and Renewable Energy
  - Argonne is one of the lead research institutes in biofuels and energy storage (battery) research
  - Other areas of research heavily carried out as well: e.g., study of high-energy states for electrons (used in solar cells)
- Non-fossil-fuel energy sources
  - Argonne is one of the primary laboratories for nuclear energy research
  - Recently funded project (CESAR) on new capabilities for modern nuclear reactors
    - E.g., Nondeterministic computations and their validation
- Bioinformatics and medicine
- Computational challenges for these projects are too large to fix today
  - Two-prong approach: faster hardware/software and better algorithms

Exascale is International
Discussions and plans almost 8 years in the making…

10^18 INTERNATIONAL EXASCALE SOFTWARE PROJECT

Europe is moving forward



EU to double supercomputing funding to €1.2bn

By Jack Clark, ZDNet UK, 16 February, 2012 16:11 Follow @mappingbabel

NEWS Supercomputing in Europe is set to get a boost after the European Commission announced plans to double its funding of high-performance computing.

Annual investment in supercomputing equipment, training and research will go from €630m (£522m) to €1.2bn to help Europe "reverse its relative decline in HPC use and capabilities", the Commission said in a statement on Wednesday.

Daily Newsletters

Sign up to ZDNet UK's daily newsletter.

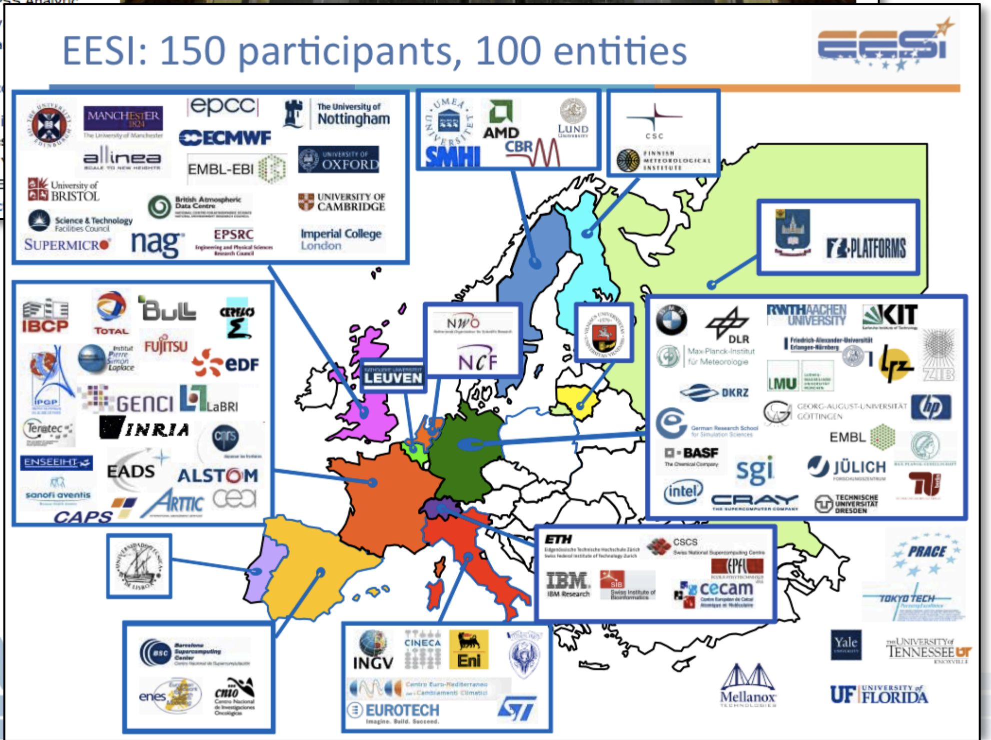
Topics

HPC, Supercomputing, Neelie Kroes, European Commission, High-performance computing, Exascale, Exaflop, Petaflop, Curie, Top500, Investment, Funding, PRACE

Sponsored Links

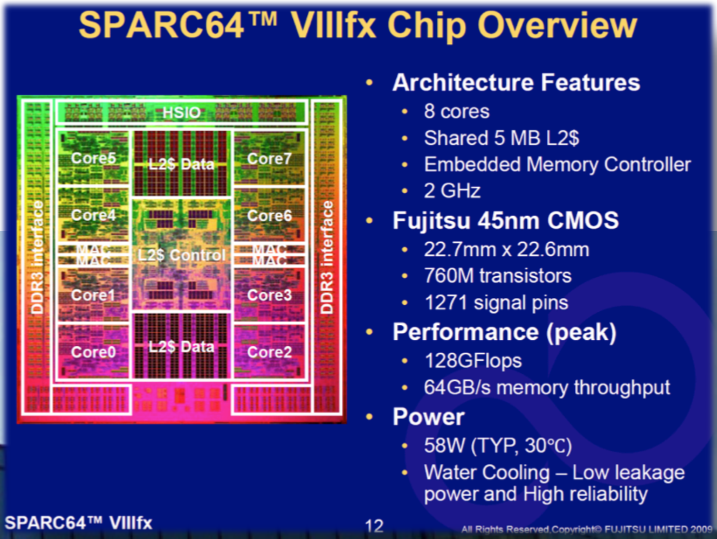
SPSS Business Analytics
Get IBM SPSS Analytic
Case Study
Top Compa
SPSS.
www.ibm.c

Foreigner i
Are Japanes
Increasing
Put Your YE
www.Objec



EESI: 150 participants, 100 entities

Japan has K computer…
is planning for exascale…



SPARC64™ VIIIfx Chip Overview

HSIO
Core5
L2$ Data
Core7
Core4
Core6
MAC
MAC
L2$ Control
MAC
MAC
Core1
Core3
Core0
L2$ Data
Core2
DDR3 interface
DDR3 interface

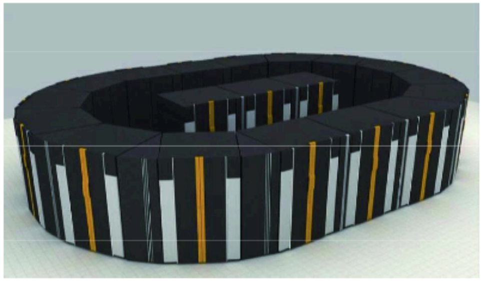
- **Architecture Features**
  - 8 cores
  - Shared 5 MB L2$
  - Embedded Memory Controller
  - 2 GHz
- **Fujitsu 45nm CMOS**
  - 22.7mm x 22.6mm
  - 760M transistors
  - 1271 signal pins
- **Performance (peak)**
  - 128GFlops
  - 64GB/s memory throughput
- **Power**
  - 58W (TYP, 30°C)
  - Water Cooling – Low leakage power and High reliability

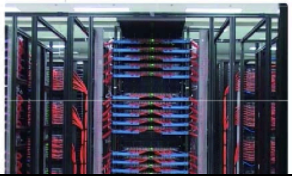
SPARC64™ VIIIfx 12

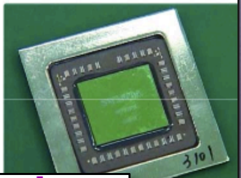
China: New Architectures and Big Systems

Sunway Bluelight

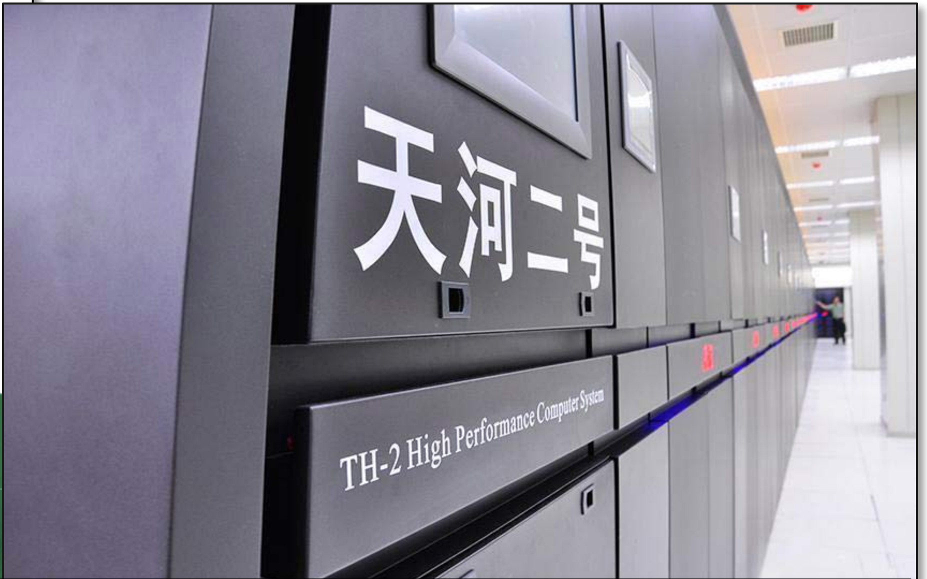
- National Supercomputing Center in Jinan, China
- Ranked 14th on TOP500 (November, 2011)
  - □ 1PF peak
  - □ Power：1074.00 kW
  - □ Cores：137200
  - □ Memory：139264 GB
  - □ Very compact system
    - 128TF/Rack
  - □ Implemented with domestic 16-core processors
  - □ Infiniband QDR 40Gbps
  - □ **Exploring possible architectures and key technologies for 10-Petascale computers**



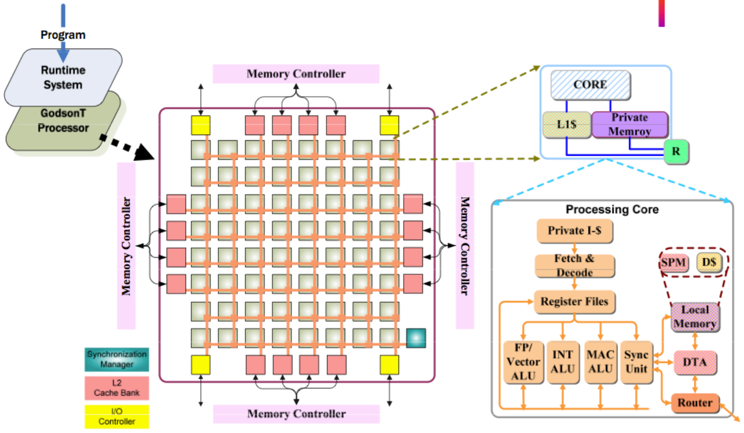




Tianhe-2 (2012)

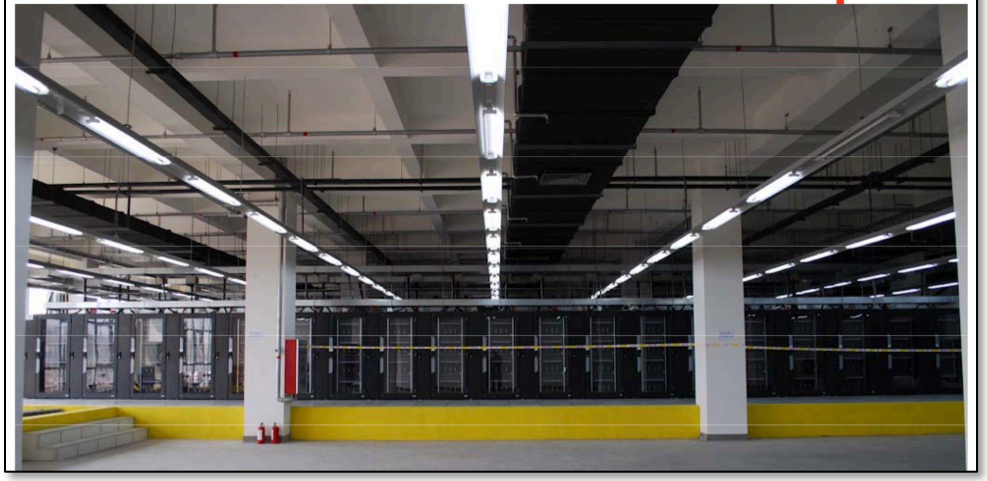


Architecture Overview of Godson-T



AMS 14 Godson | AICS 2012

Dawning Nebulae: 3PFlops (2010)

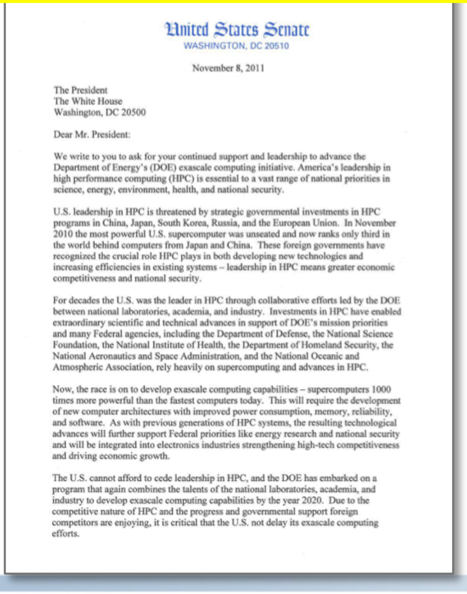


US: A grand challenge for the 21st century

Development of an Exascale Computing System is a Grand Challenge for the 21st Century

"[Development of] An "exascale" supercomputer capable of a million trillion calculations per second – dramatically increasing our ability to understand the world around us through simulation and slashing the time needed to design complex products such as therapeutics, advanced materials, and highly-efficient autos and aircraft."

Sept 20th 2009
EXECUTIVE OFFICE OF THE PRESIDENT NATIONAL ECONOMIC COUNCIL OFFICE OF SCIENCE AND TECHNOLOGY POLICY



United States Senate
WASHINGTON, DC 20510

November 8, 2011

The President
The White House
Washington, DC 20500

Dear Mr. President:

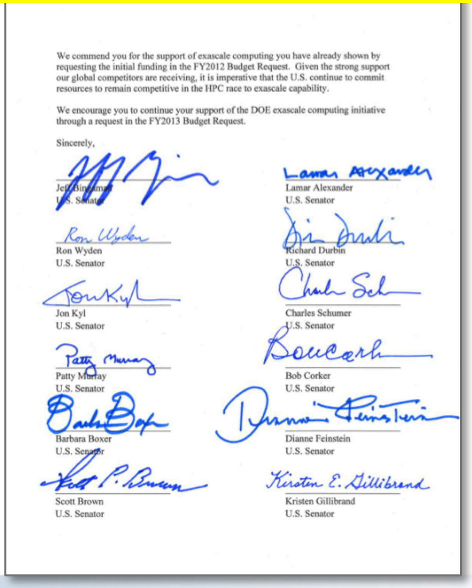
We write to you to ask for your continued support and leadership to advance the Department of Energy's (DOE) exascale computing initiative. America's leadership in high performance computing (HPC) is essential to a vast range of national priorities in science, energy, environment, health, and national security.

U.S. leadership in HPC is threatened by strategic governmental investments in HPC programs in China, Japan, South Korea, Russia, and the European Union. In November 2010 the most powerful U.S. supercomputer was unseated and now ranks only third in the world behind computers from Japan and China. These foreign governments have recognized the crucial role HPC plays in both developing new technologies and increasing efficiencies in existing systems – leadership in HPC means greater economic competitiveness and national security.

For decades the U.S. was the leader in HPC through collaborative efforts led by the DOE between national laboratories, academia, and industry. Investments in HPC have enabled extraordinary scientific and technical advances in support of DOE's mission priorities and many Federal agencies, including the Department of Defense, the National Science Foundation, the National Institute of Health, the Department of Homeland Security, the National Aeronautics and Space Administration, and the National Oceanic and Atmospheric Association, rely heavily on supercomputing and advances in HPC.

Now, the race is on to develop exascale computing capabilities – supercomputers 1000 times more powerful than the fastest computers today. This will require the development of new computer architectures with improved power consumption, memory, reliability, and software. As with previous generations of HPC systems, the resulting technological advances will further support Federal priorities like energy research and national security and will be integrated into electronics industries strengthening high-tech competitiveness and driving economic growth.

The U.S. cannot afford to cede leadership in HPC, and the DOE has embarked on a program that again combines the talents of the national laboratories, academia, and industry to develop exascale computing capabilities by the year 2020. Due to the competitive nature of HPC and the progress and governmental support foreign competitors are enjoying, it is critical that the U.S. not delay its exascale computing efforts.



We commend you for the support of exascale computing you have already shown by requesting the initial funding in the FY2012 Budget Request. Given the strong support our global competitors are receiving, it is imperative that the U.S. continue to commit resources to remain competitive in the HPC race to exascale capability.

We encourage you to continue your support of the DOE exascale computing initiative through a request in the FY2013 Budget Request.

Sincerely,

Jeff Bingaman
U.S. Senator

Lamar Alexander
U.S. Senator

Ron Wyden
U.S. Senator

Richard Durbin
U.S. Senator

Jon Kyl
U.S. Senator

Charles Schumer
U.S. Senator

Patty Murray
U.S. Senator

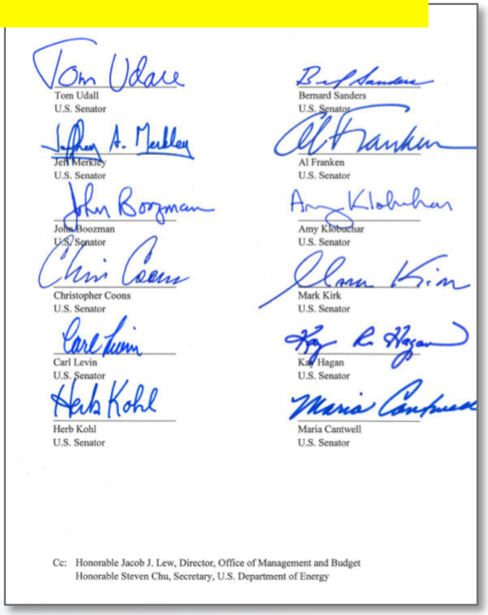
Bob Corker
U.S. Senator

Barbara Boxer
U.S. Senator

Dianne Feinstein
U.S. Senator

Scott Brown
U.S. Senator

Kristen Gillibrand
U.S. Senator



Tom Udall
U.S. Senator

Bernard Sanders
U.S. Senator

Jeff Merkley
U.S. Senator

Al Franken
U.S. Senator

John Boozman
U.S. Senator

Amy Klobuchar
U.S. Senator

Christopher Coons
U.S. Senator

Mark Kirk
U.S. Senator

Carl Levin
U.S. Senator

Kay Hagan
U.S. Senator

Herb Kohl
U.S. Senator

Maria Cantwell
U.S. Senator

Cc: Honorable Jacob J. Lew, Director, Office of Management and Budget
Honorable Steven Chu, Secretary, U.S. Department of Energy

Exascale Computing Trends

U.S. DOE Potential System Architecture Targets

System attributes	2010	"2015"		"2020"	
System peak	2 Peta	200-300 Petaflop/sec		1 Exaflop/sec	
Power	6 MW	15 MW		20-30 MW	
System memory	0.3 PB	5 PB		32-64 PB	
Node performance	125 GF	0.5 TF	7 TF	1 TF	10 TF
Node memory BW	25 GB/s	0.1TB/sec	1 TB/sec	0.4TB/sec	4 TB/sec
Node concurrency	12	O(100)	O(1,000)	O(1,000)	O(10,000)
System size (nodes)	18,700	50,000	5,000	1,000,000	100,000
Total Node Interconnect BW	1.5 GB/s	20 GB/sec		200GB/sec	
MTTI	days	O(1day)		O(1 day)	

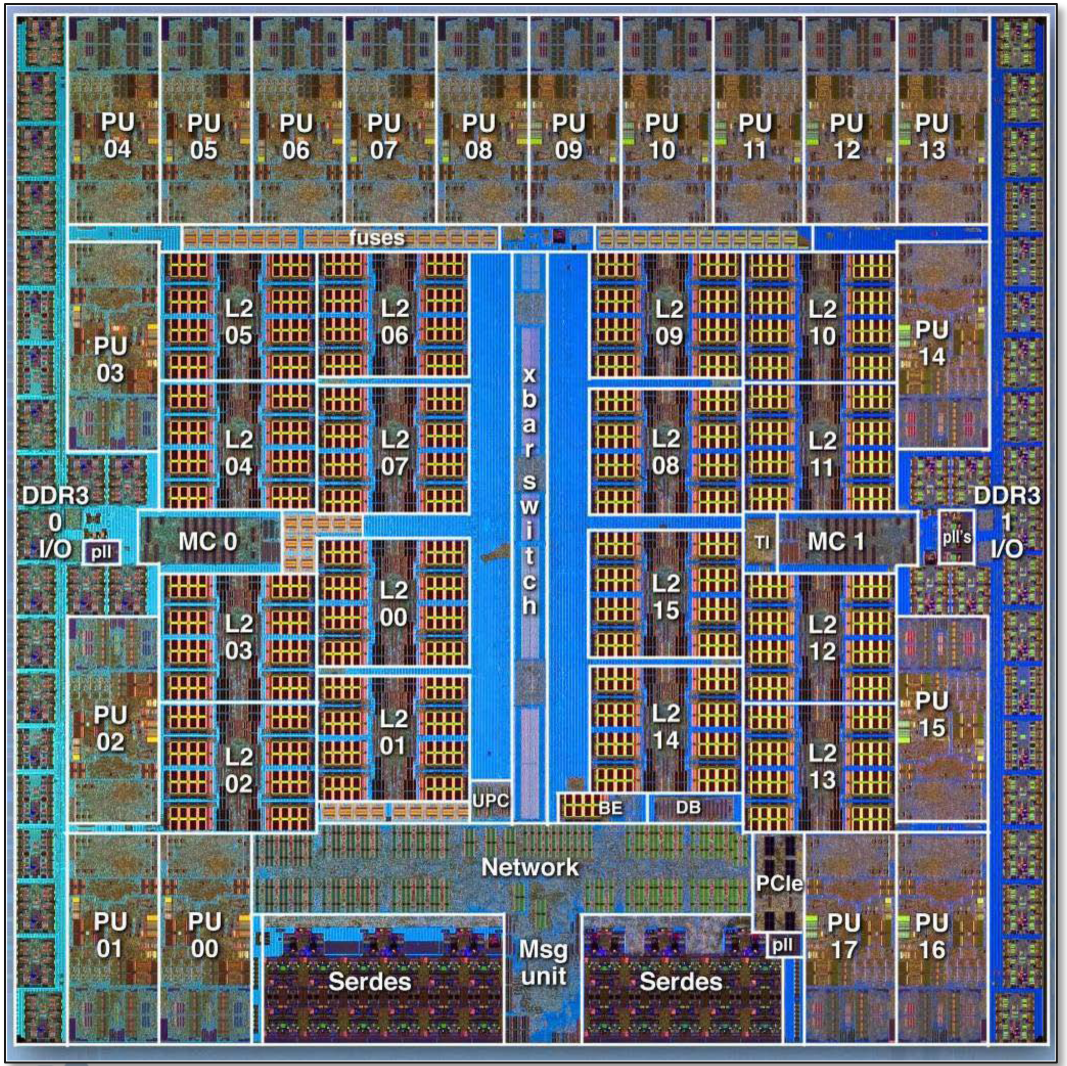
Courtesy Kathy Yelick (Lawrence Berkeley National Laboratory)

Mira: Argonne's Newest GREEN Supercomputer

- Blue Gene/Q System
  - 48 racks
  - 786,432 cores
  - 786 TB of memory
  - Peak flop rate: 10 PF
- Storage System
  - ~30 PB capability
    - 240GB/s bandwidth (GPFS)

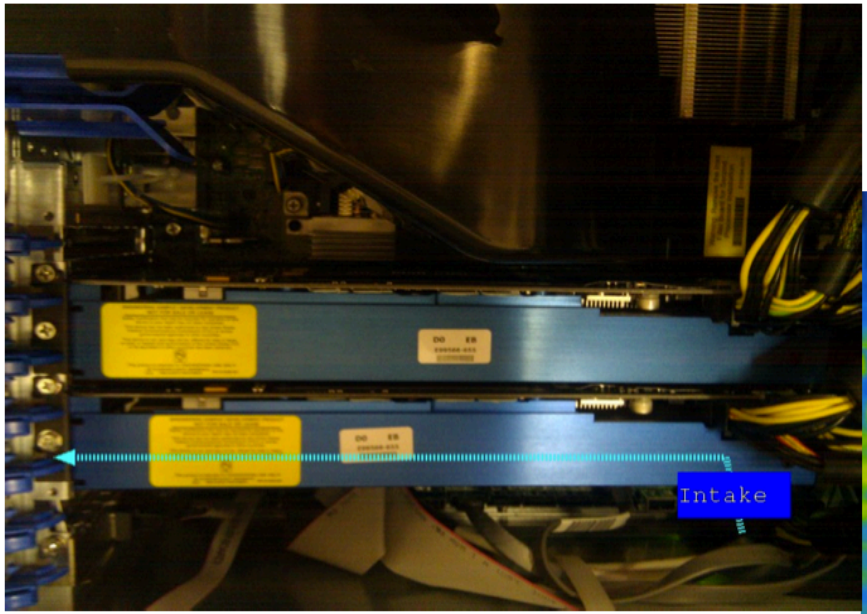
BlueGene/Q Compute chip

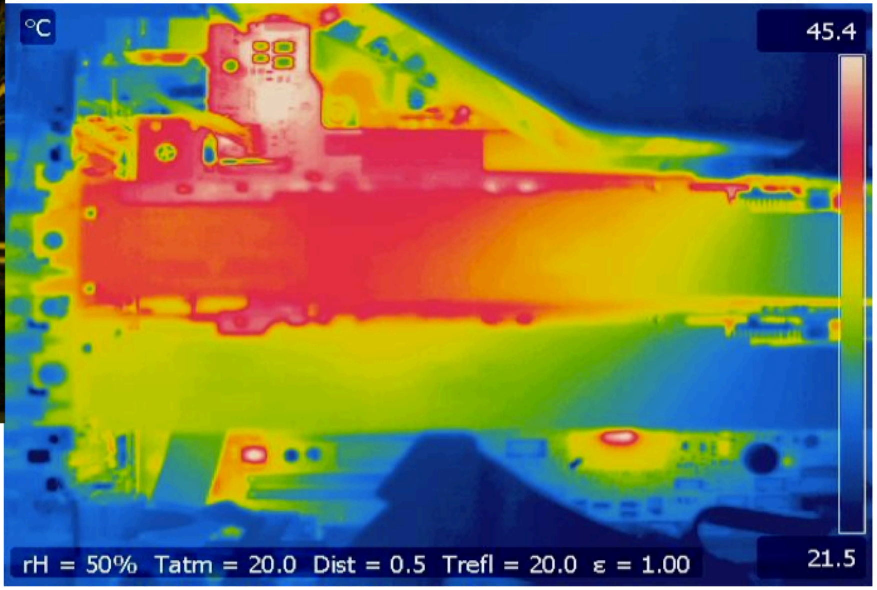
System-on-a-Chip design : integrates processors, memory and networking logic into a single chip



- **360 mm^2 Cu-45 technology (SOI)**
  - ~ 1.47 B transistors
- **16 user + 1 service PPC processors**
  - plus 1 redundant processor
  - all processors are symmetric
  - each 4-way multi-threaded
  - 64 bits
  - 1.6 GHz
  - L1 I/D cache = 16kB/16kB
  - L1 prefetch engines
  - each processor has Quad FPU (4-wide double precision, SIMD)
  - peak performance 204.8 GFLOPS @ 55 W
- **Central shared L2 cache: 32 MB**
  - eDRAM
  - multiversioned cache – will support transactional memory, speculative execution.
  - supports atomic ops
- **Dual memory controller**
  - 16 GB external DDR3 memory
  - 1.33 Gb/s
  - 2 * 16 byte-wide interface (+ECC)
- **Chip-to-chip networking**
  - Router logic integrated into BQC chip.
- **External IO**
  - PCIe Gen2 interface

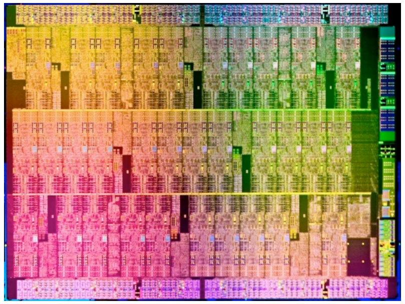
Exploring Power on Intel Knights Ferry



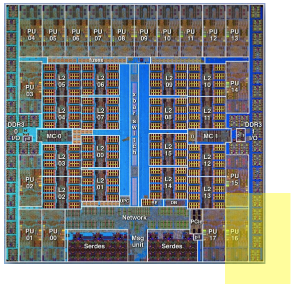


- Intel SS5520SC mother board
- Two D0 stepping KNF cards
- Cento OS 6.0
- alpha8-update

Key Changes:
Coherency, Power Management, Specialization



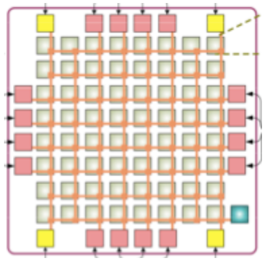
Intel: MIC



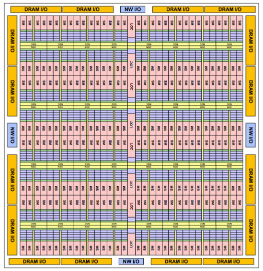
IBM: BG/Q

#18

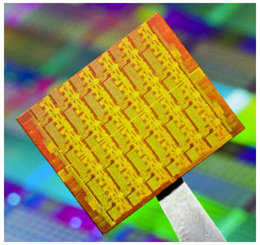
Power Constrained Memory Consistency



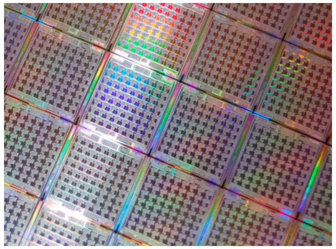
Godson T



Dally: Echelon



Intel: SCC

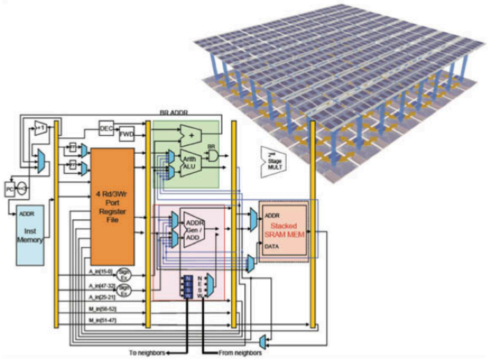


Tilera: GX

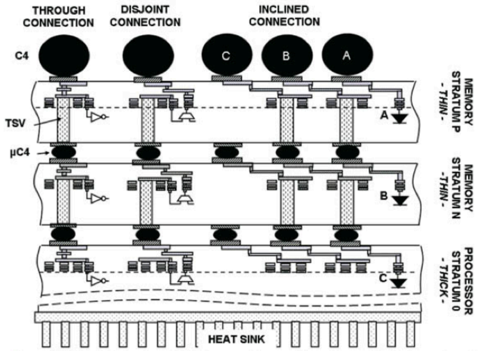
Extreme Specialization and Power Management

Chien: 10x10

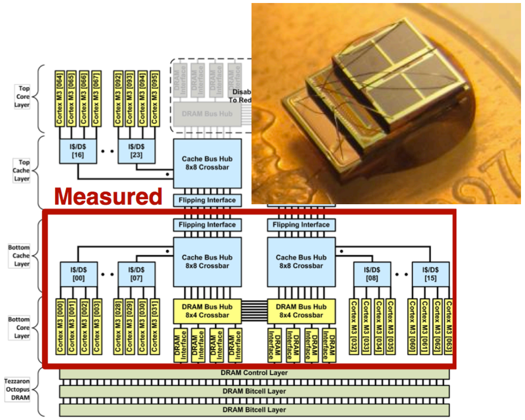
3D Chip Stacking: Fast, Close, (relatively) Small



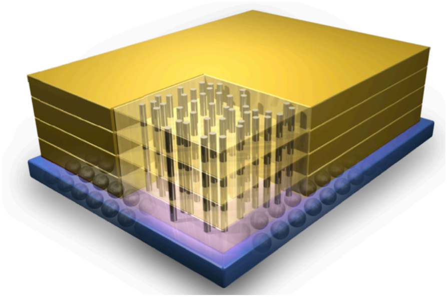
Georgia Tech


THROUGH CONNECTION
DISJOINT CONNECTION
INCLINED CONNECTION
C4
C
B
A
TSV
μC4
A
B
C
MEMORY STRATUM P - THIN -
MEMORY STRATUM N - THIN -
PROCESSOR STRATUM 0 - THICK -
HEAT SINK

IBM



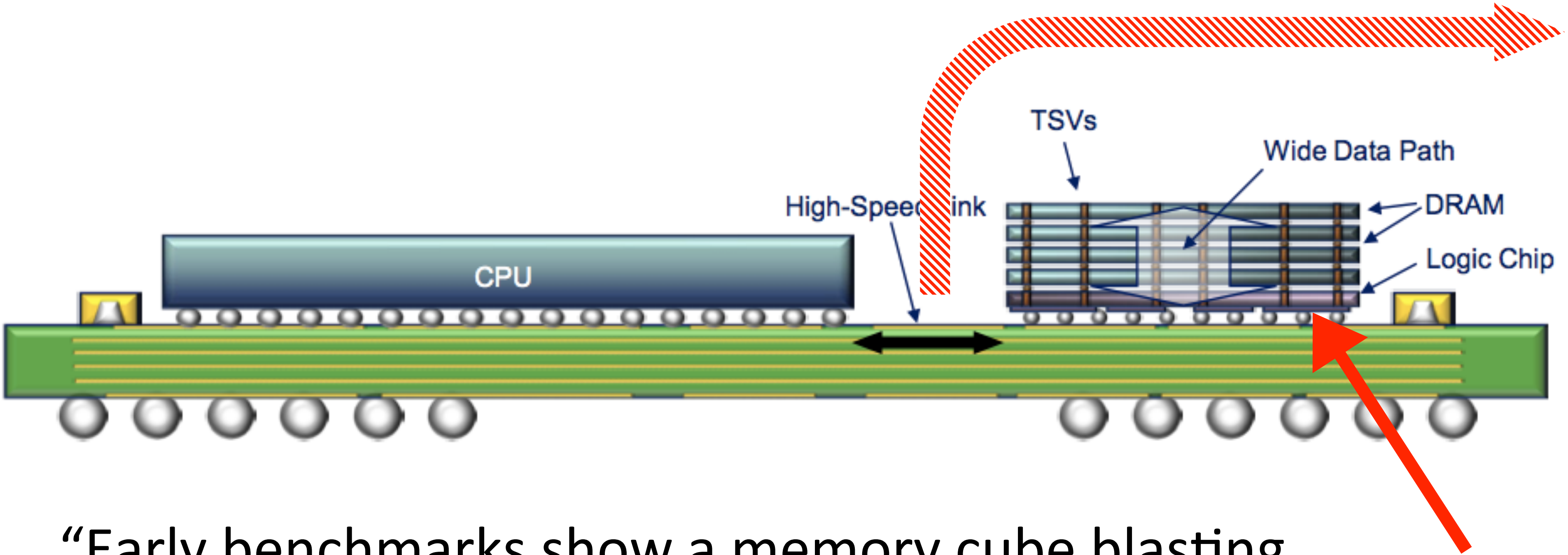
Univ of Michigan



Micron HMC

Micron Hybrid Memory Cube

Future on-module Interconnect pipe?



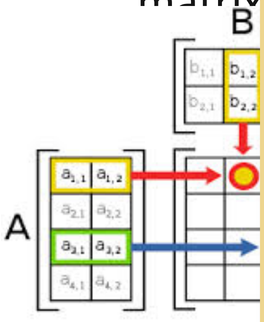
Logic!

"Early benchmarks show a memory cube blasting data 12 times faster than DDR3-1333 SDRAM while using only about 10 percent of the power."

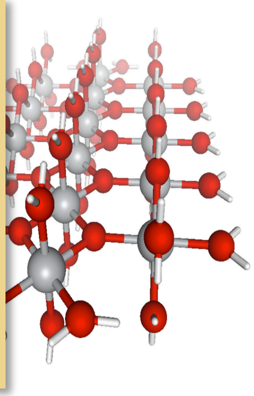
Irregular Computations

- **"Traditional" computations**
  - Organized around dense vectors or matrices
  - Regular data movement pattern, use MPI SEND/RECV or collectives
  - More local computation, less data movement
  - Example: stencil computation, matrix

- **Irregular computations**
  - Organized around graphs, sparse vectors, more "data driven" in nature
  - Data movement pattern is irregular and data-dependent
  - Growth rate of data movement is much faster than computation
  - Example: social network analysis, bioinformatics

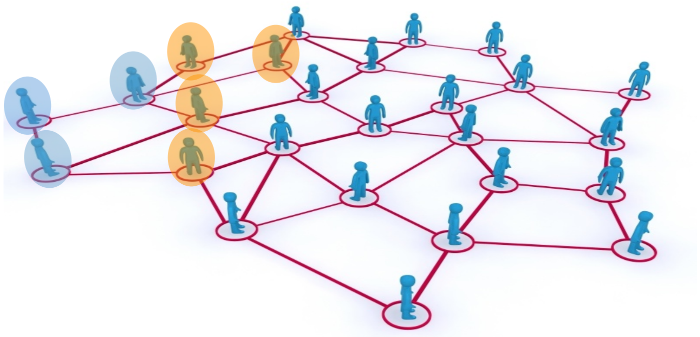


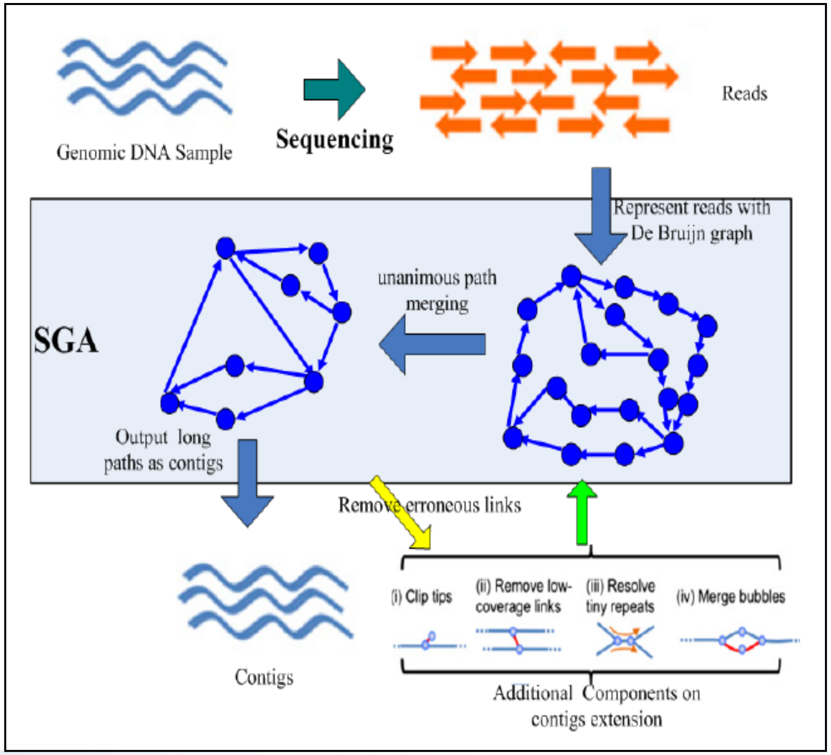
- **"New" irregular computations**
  - Increasing trend of applications moving from regular to irregular computation models
    - Computation complexity, data movement restrictions, etc.
  - Example: sparse matrix multiplication

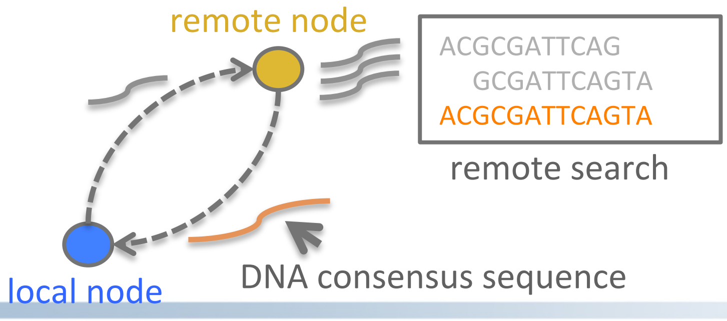


Example of Irregular Computations (1)

- Graph algorithms
  - Commonly used in social network analysis, like finding friends connections and recommendations
- DNA sequence assembly
  - Graph is different for various queries
  - Graph is dynamically changed throughout the execution
  - Fundamental operation: search for overlapping of sequences (send query sequence to target node; search through entire database on that node; return result sequence)




Genomic DNA Sample
Sequencing
Reads
Represent reads with De Bruijn graph
SGA
unanimous path merging
Output long paths as contigs
Remove erroneous links
Contigs
(i) Clip tips
(ii) Remove low-coverage links
(iii) Resolve tiny repeats
(iv) Merge bubbles
Additional Components on contigs extension



remote node

ACGCGATTCAG
GCGATTCAGTA
ACGCGATTCAGTA

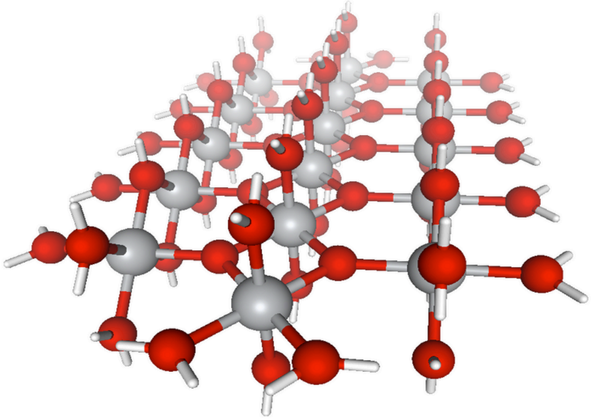
remote search

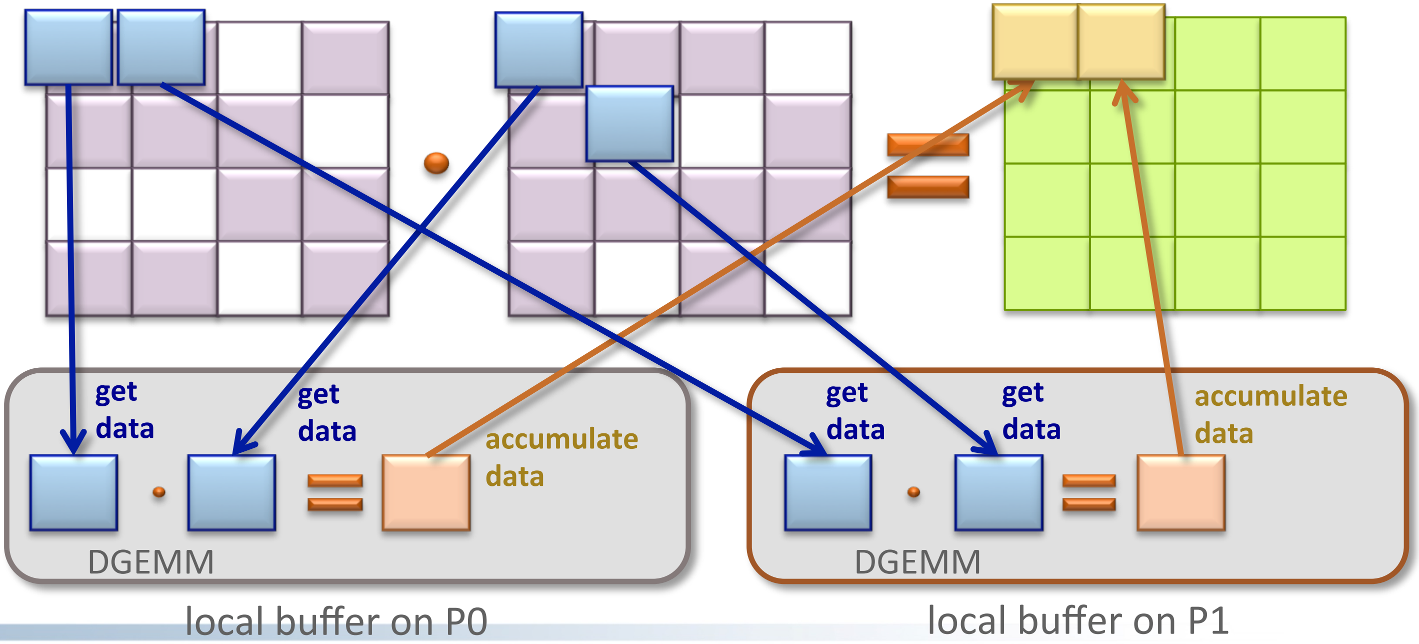
local node

DNA consensus sequence

Example of Irregular Computations (2)

- **NWChem: high performance computational chemistry**
  - Main computation: fetch data from remote processes, do some computation locally, then accumulate data to remote processes
  - Sparse matrix, symmetric matrix, etc.





Let’s talk MPI!

What is MPI?

- MPI: Message Passing Interface
  - The MPI Forum organized in 1992 with broad participation by:
    - Vendors: IBM, Intel, TMC, SGI, Convex, Meiko
    - Portability library writers: PVM, p4
    - Users: application scientists and library writers
    - MPI-1 finished in 18 months
  - Incorporates the best ideas in a “standard” way
    - Each function takes fixed arguments
    - Each function has fixed semantics
      - Standardizes what the MPI implementation provides and what the application can and cannot expect
      - Each system can implement it differently as long as the semantics match
- MPI is not…
  - a language or compiler specification
  - a specific implementation or product

MPI-1

- MPI-1 supports the classical message-passing programming model: basic point-to-point communication, collectives, datatypes, C/Fortran bindings, etc.
- MPI-1 was defined (1994) by a broadly based group of parallel computer vendors, computer scientists, and applications developers.
  - 2-year intensive process
- Implementations appeared quickly and now MPI is taken for granted as vendor-supported software on any parallel machine.
- Free, portable implementations exist for clusters and other environments (MPICH, Open MPI)

Following MPI Standards

- MPI-2 was released in 2000
  - Several additional features including MPI + threads, MPI-I/O, remote memory access, dynamic processes, C++/F90 bindings and others
- MPI-2.1 (2008) and MPI-2.2 (2009) were recently released with some corrections to the standard and small features
- MPI-3 (2012) added several new features to MPI
- The Standard itself:
  - at http://www.mpi-forum.org
  - All MPI official releases, in both postscript and HTML
- Other information on Web:
  - at http://www.mcs.anl.gov/mpi
  - pointers to lots of material including tutorials, a FAQ, other MPI pages

Status of MPI-3 Implementations

	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

Current Situation with Production Applications (1)

- The vast majority of DOE's production parallel scientific applications today use MPI
  - Increasing number use (MPI + OpenMP) hybrid
  - Some exploring (MPI + accelerator) hybrid
- Today's largest systems in terms of number of regular cores (excluding GPU cores)

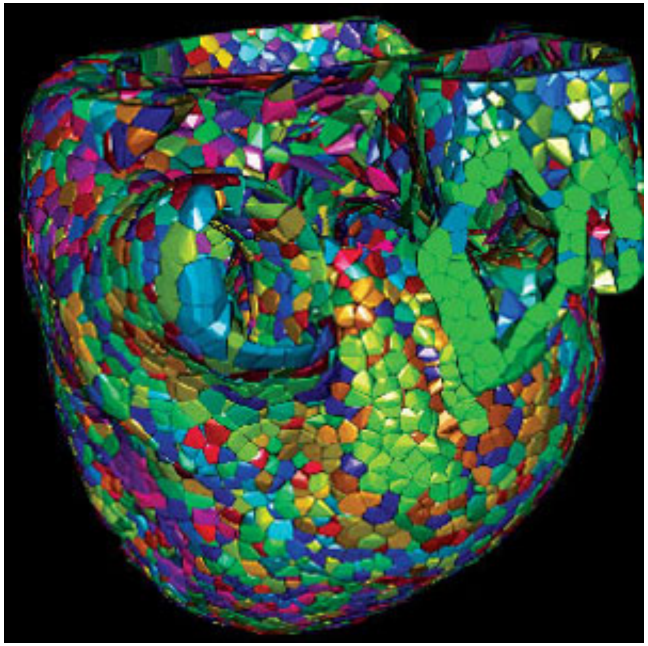
Sequoia (LLNL)	1,572,864 cores
Mira (ANL)	786,432 cores
K computer	705,024 cores
Jülich BG/Q	393,216 cores
Blue Waters	386,816 cores
Titan (ORNL)	299,008 cores

- **MPI already runs in production on systems with up to 1.6 million cores**

Current Situation with Production Applications (2)

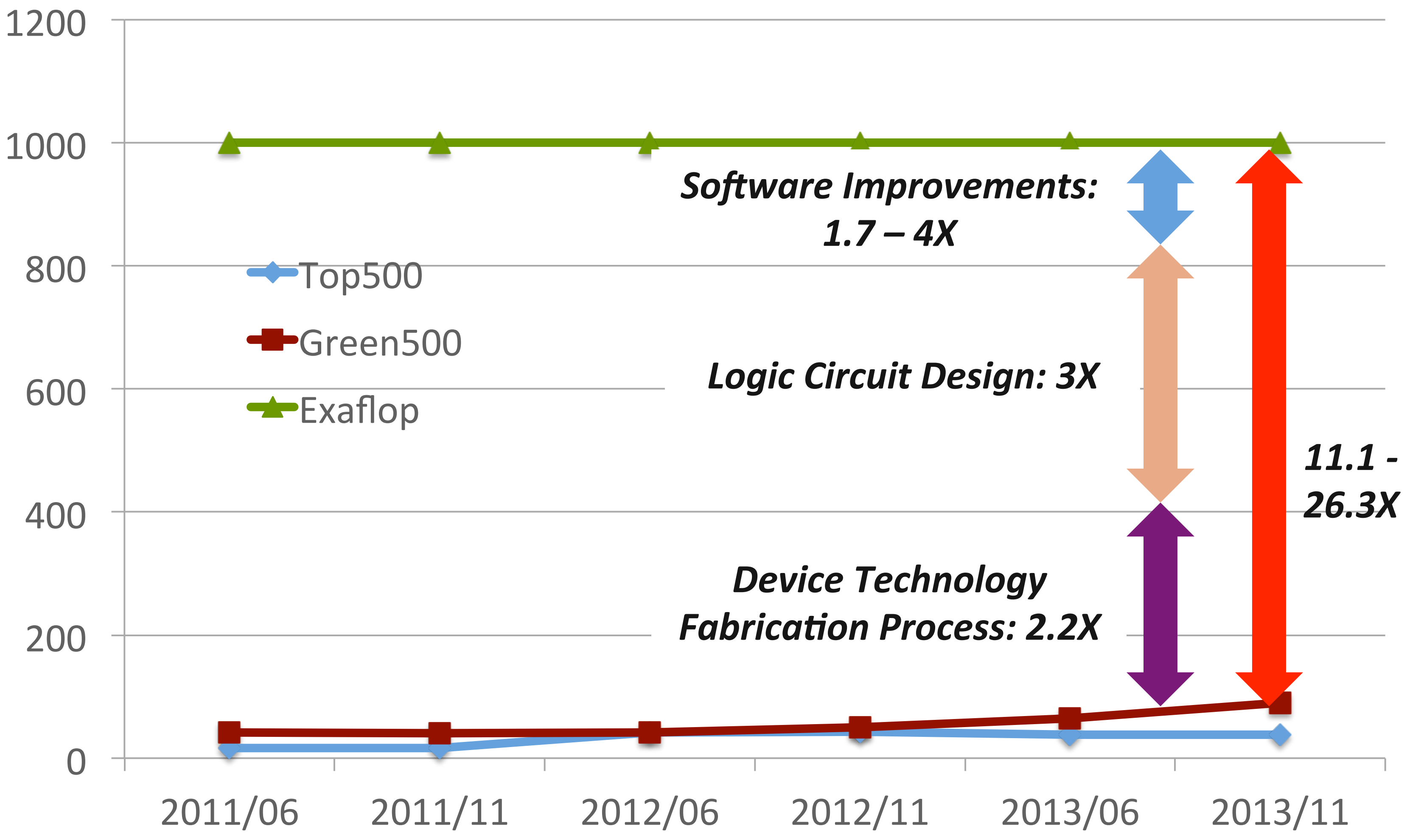
- IBM has successfully scaled the LAMMPS application to over 3 million MPI ranks
- Applications are running at scale on LLNL's Sequoia and achieving **12 to 14 petaflops** *sustained* performance
- HACC cosmology code from Argonne (PI: Salman Habib) achieved **14 petaflops** on Sequoia
  - Ran on full Sequoia system using MPI + OpenMP hybrid
  - Used 16 MPI ranks * 4 OpenMP threads on each node, which matches the hardware architecture: 16 cores per node with 4 hardware threads each
  - http://www.hpcwire.com/hpcwire/2012-11-29/sequoia_supercomputer_runs_cosmology_code_at_14_petaflops.html
  - SC12 Gordon Bell prize finalist

Current Situation with Production Applications (3)



- Cardioid cardiac modeling code (IBM & LLNL) achieved **12 petaflops** 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
- And there are other applications running at similar scales…

On the path to Exascale



MPI in the Exascale Era

- Under a lot of scrutiny (good!)
  - Lots of myths floating around (bad!)
- Push to get new programming models designed and developed for exascale
- The truth is that MPI today is a new programming model (compared to 2004) , and MPI in 2020 will be a new programming model (compared to today)
- Strengths of MPI
  - Composability
    - Ability to build tools and libraries above and around MPI
    - No “do everything under the sun” attitude
  - Continuous evolution
    - The standard incorporates best research ideas

MPI Myth #1:

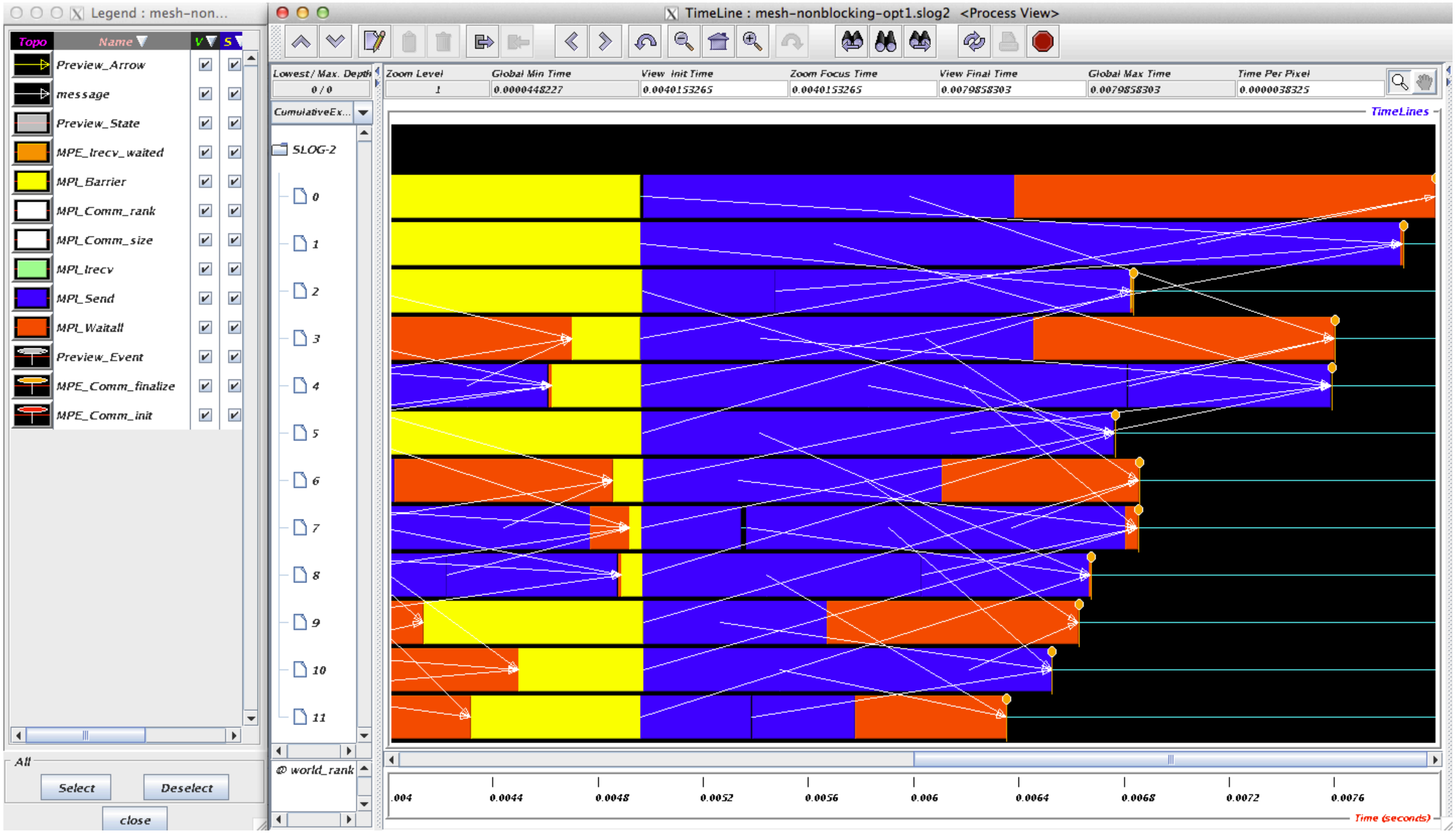
“MPI is bulk synchronous”

“MPI is too static”

Bulk Synchronous Programming

- Many current MPI applications work in a bulk synchronous fashion
  - Each process computes on its local data and all processes exchange data collectively
  - Reasons include (1) better cache/data locality compared to highly asynchronous models, and (2) easier to debug
  - Result: if there is any irregularity in their computation, it results in idleness

Sample Computation/Communication Timeline



Legend : mesh-non...

Topo	Name	V	S
	Preview_Arrow	☑	☑
	message	☑	☑
	Preview_State	☑	☑
🟧	MPE_Irecv_waited	☑	☑
🟨	MPI_Barrier	☑	☑
⬜	MPI_Comm_rank	☑	☑
⬜	MPI_Comm_size	☑	☑
🟩	MPI_Irecv	☑	☑
🟦	MPI_Send	☑	☑
🟥	MPI_Waitall	☑	☑
	Preview_Event	☑	☑
	MPE_Comm_finalize	☑	☑
	MPE_Comm_init	☑	☑

TimeLine : mesh-nonblocking-opt1.slog2 <Process View>

Lowest/Max. Depth	Zoom Level	Global Min Time	View Init Time	Zoom Focus Time	View Final Time	Global Max Time	Time Per Pixel
0 / 0	1	0.0000448227	0.0040153265	0.0040153265	0.0079858303	0.0079858303	0.0000038325

BSP in MPI

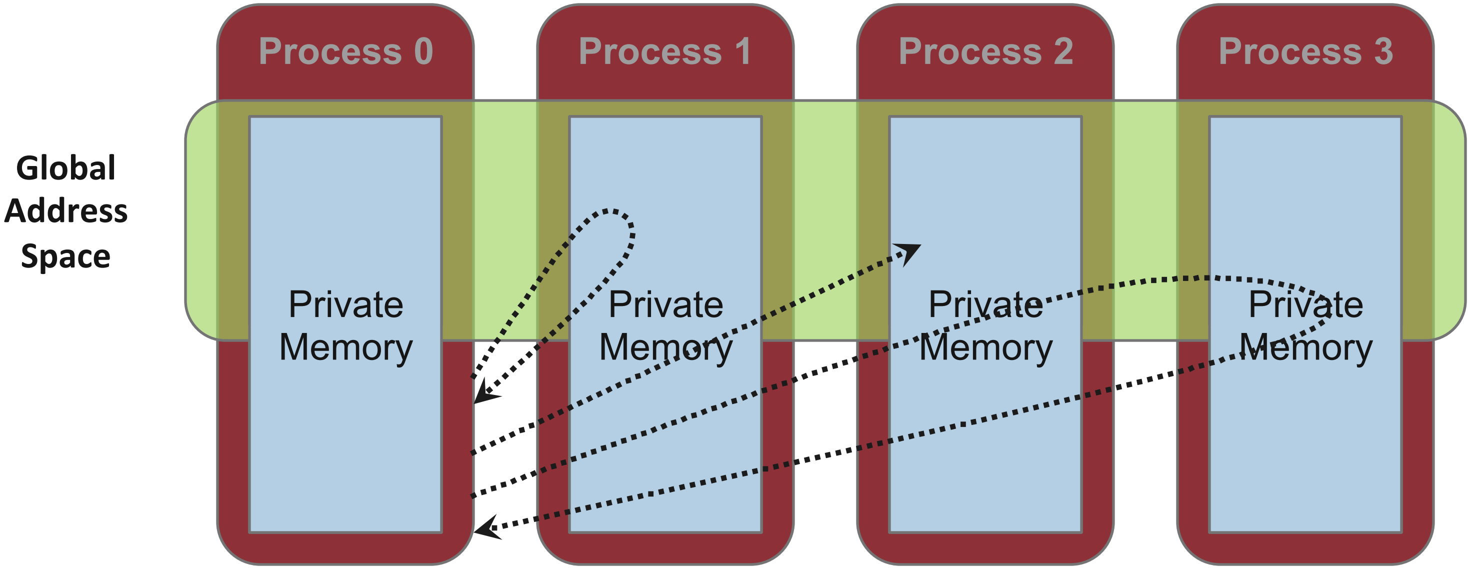
- MPI does not mandate or even recommend bulk synchronous programming
- Asynchrony (within reason) is good
- The reason why applications are bulk synchronous is because they want to (e.g., because of simplicity), not because MPI requires them to

Improved RMA Interface

- Substantial extensions to the MPI-2 RMA interface
- New window creation routines:
  - MPI_Win_allocate: MPI allocates the memory associated with the window (instead of the user passing allocated memory)
  - MPI_Win_create_dynamic: Creates a window without memory attached. User can dynamically attach and detach memory to/from the window by calling MPI_Win_attach and MPI_Win_detach
  - MPI_Win_allocate_shared: Creates a window of shared memory (within a node) that can be can be accessed simultaneously by direct load/store accesses as well as RMA ops
- New atomic read-modify-write operations
  - MPI_Get_accumulate
  - MPI_Fetch_and_op (simplified version of Get_accumulate)
  - MPI_Compare_and_swap

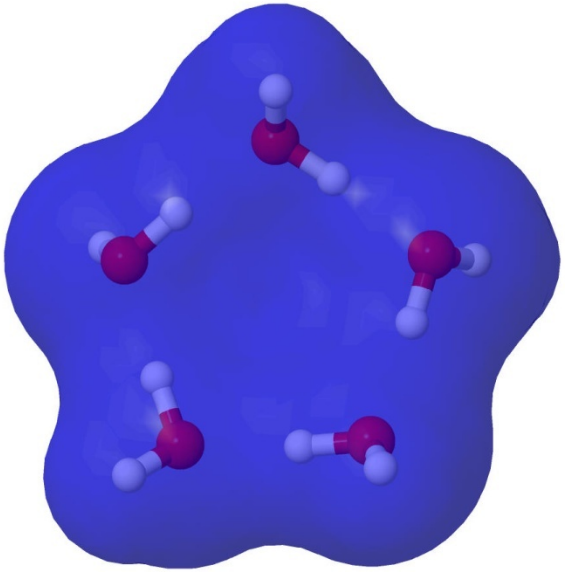
One-sided Communication

- The basic idea of one-sided communication models is to decouple data movement with process synchronization
  - Should be able move data without requiring that the remote process synchronize
  - Each process exposes a part of its memory to other processes
  - Other processes can directly read from or write to this memory

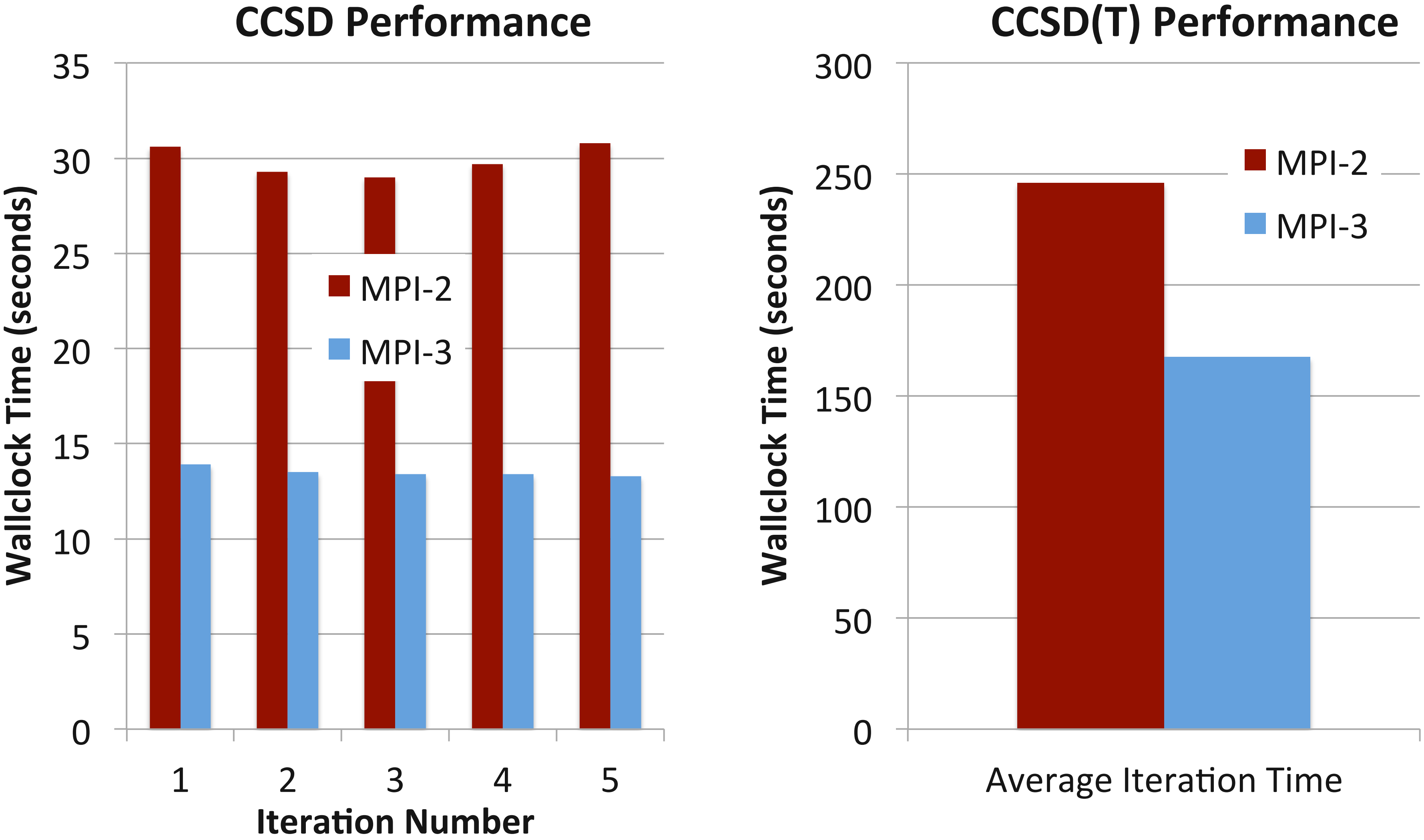


Use Case: Distributed Shared Arrays

- Quantum Monte Carlo: Ensemble data
  - Represents initial quantum state
  - Spline representation, cubic basis functions
  - Large(100+ GB), read-only table of coeff.
  - Accesses are random
- Coupled cluster simulations
  - Evolving quantum state of the system
  - Very large, tables of coefficients
  - Table$_t$ read-only, Table$_{t+1}$ accumulate-only
  - Accesses are non-local/overlapping
- Global Arrays PGAS programming model
  - Can be supported with passive mode RMA [Dinan et al., IPDPS'12]



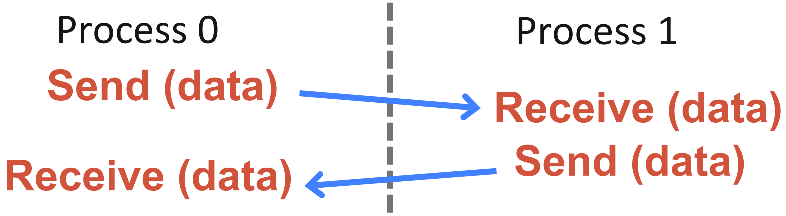
Case-study: NWChem over MPI-3



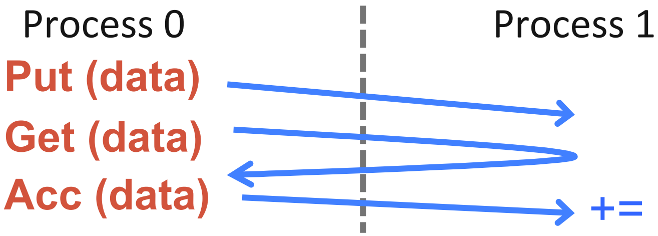
Courtesy Jeff Hammond, Argonne National Laboratory

Message Passing Models

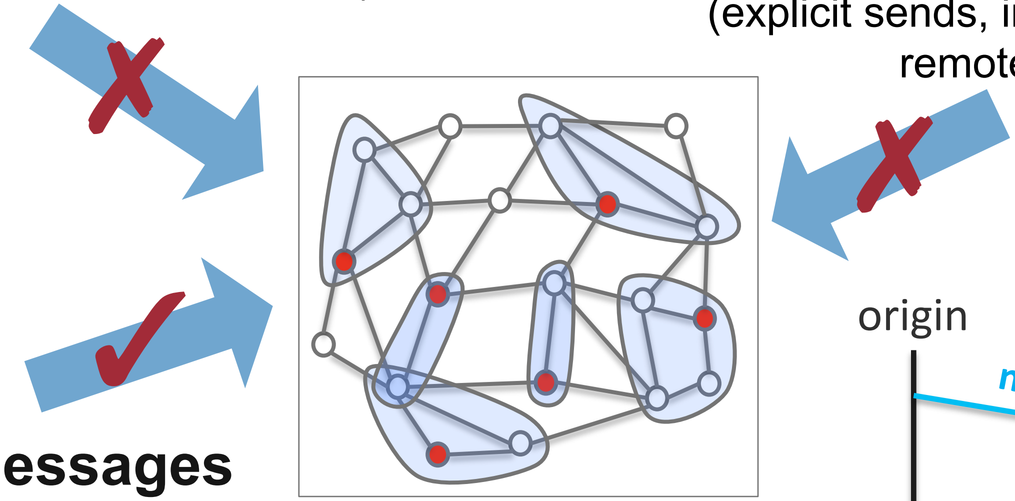
- Current MPI is not well-suitable to data-intensive applications



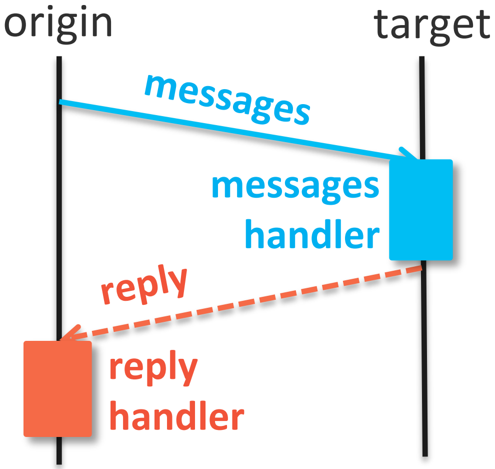
two-sided communication
(explicit sends and receives)



one-sided (RMA) communication
(explicit sends, implicit receives, simple remote operations)

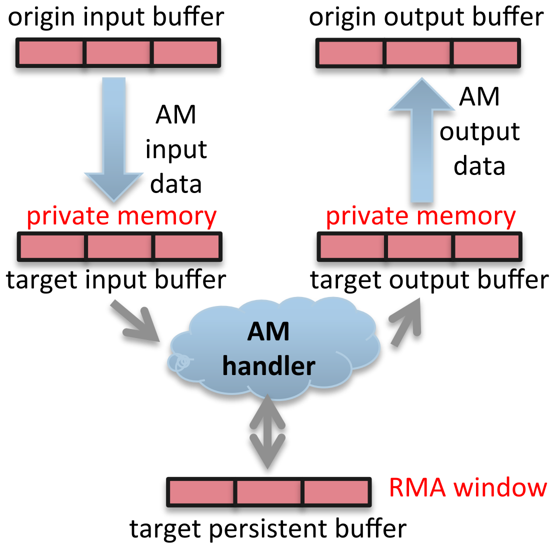


- Active Messages
  - Sender explicitly sends message
  - Upon message's arrival, message handler is triggered, receiver is not explicitly involved
  - User-defined operations on remote process



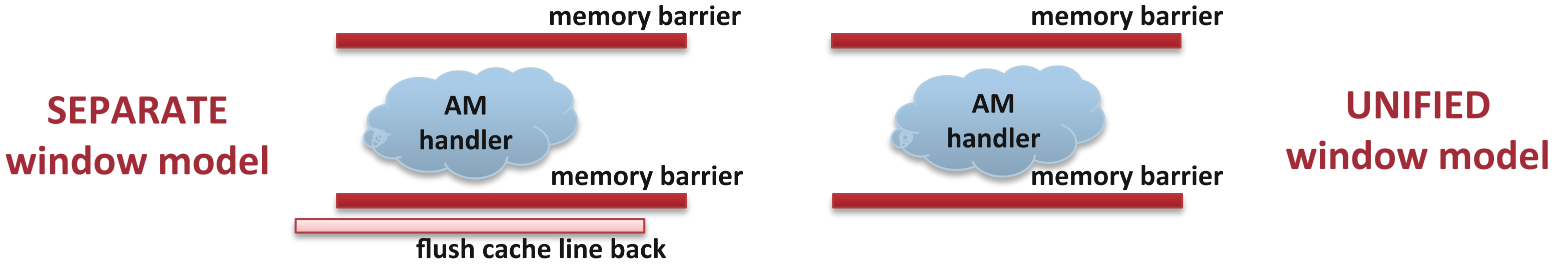
Generalized and MPI-Interoperable AM

MPI-AM: an MPI-interoperable framework that can dynamically manage data movement and user-defined remote computation.



MPI-AM workflow

- **Streaming AMs**
  - define "**segment**"--- minimum number of elements for AM execution
  - achieve pipeline effect and reduce temporary buffer requirement
- **Explicit and implicit buffer management**
  - **system buffers**: eager protocol, not always enough
  - **user buffers**: rendezvous protocol, guarantee correct execution
- **Correctness semantics**
  - **Memory consistency**
    - MPI runtime must ensure consistency of window

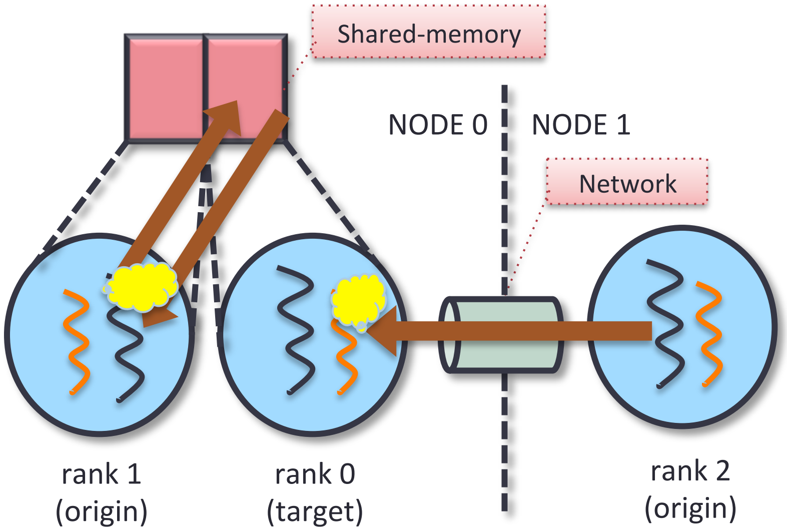


  - **Three different type of ordering**
  - **Concurrency**: by default, MPI runtime behaves "**as if**" AMs are executed in sequential order. User can release concurrency by setting MPI assert.

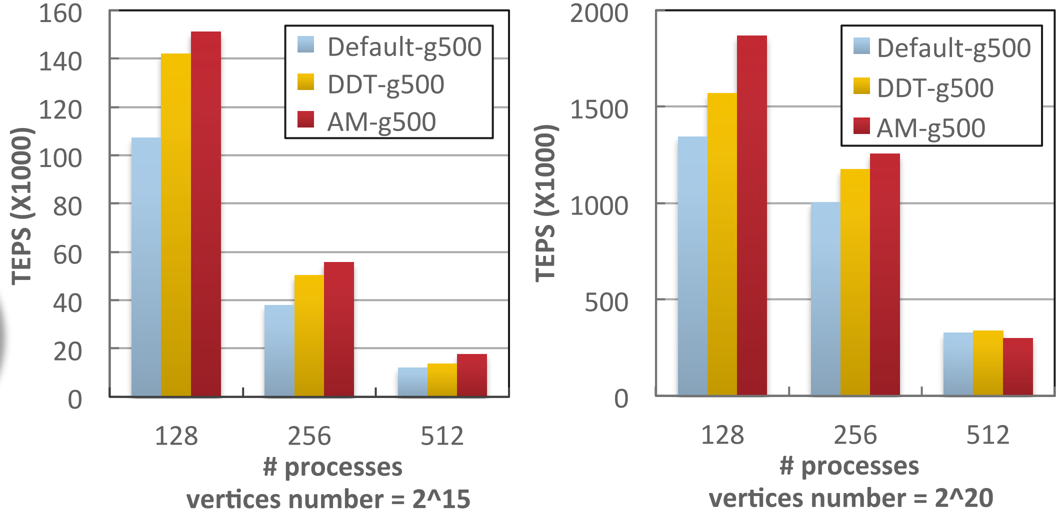
[ICPADS 2013] X. Zhao, P. Balaji, W. Gropp, R. Thakur, *"MPI-Interoperable and Generalized Active Messages"*, in proceedings of ICPADS' 13

Asynchronous and MPI-Interoperable AM

- **Supporting asynchronous AMs internally from MPI library**
  - Inter-node messages: **spawn a thread in network module**
    - Block waiting for AM
    - Separate sockets for AM and other MPI messages
  - Intra-node messages: **"origin computation"**
    - Processes on the same node allocate window on a shared-memory region
    - Origin process directly fetches data from target process's memory, completes computation locally and writes data back to target process's memory



Design of asynchronous AMs



Graph500 results, strong scaling (gathered on Fusion cluster at ANL, 320 nodes, 2560 cores, QDR InfiniBand)

[CCGrid 2013] X. Zhao, D. Buntinas, J. Zounmevo, J. Dinan, D. Goodell, P. Balaji, R. Thakur, A. Afsahi, W. Gropp, *"Towards Asynchronous and MPI-Interoperable Active Messages"*, in proceedings of CCGrid' 13

MPI Myth #2:

“MPI cannot deal with manycore systems”

MPI+Threads Hybrid Programming

- One of the most successful models in used today
- Hybrid programming vs. a single unified programming model
  - The number of models we program to should not be too large, but a small collection of standardized programming models which interoperate with each other is not a bad thing
  - MPI+OpenMP has demonstrated this successfully

Why is this:

better than this?

Four levels of MPI Thread Safety

- MPI_THREAD_SINGLE
  - MPI only, no threads
- MPI_THREAD_FUNNELED
  - Outside OpenMP parallel region, or OpenMP master region
- MPI_THREAD_SERIALIZED
  - Outside OpenMP parallel region, or OpenMP single region, or critical region
- MPI_THREAD_MULTIPLE
  - Any thread is allowed to make MPI calls at any time

```
#pragma omp parallel for
for (i = 0; i < N; i++) {
        uu[i] = (u[i] + u[i - 1] +  u[i + 1])/5.0;
}

MPI_Function ( );
```

```
#pragma omp parallel
{
        /* user computation */
        #pragma omp single
        MPI_Function ();
}
```

```
#pragma omp parallel
{
        /* user computation */
        #pragma omp critical
        MPI_Function ();
}
```

Problem: Idle Resources during MPI Calls

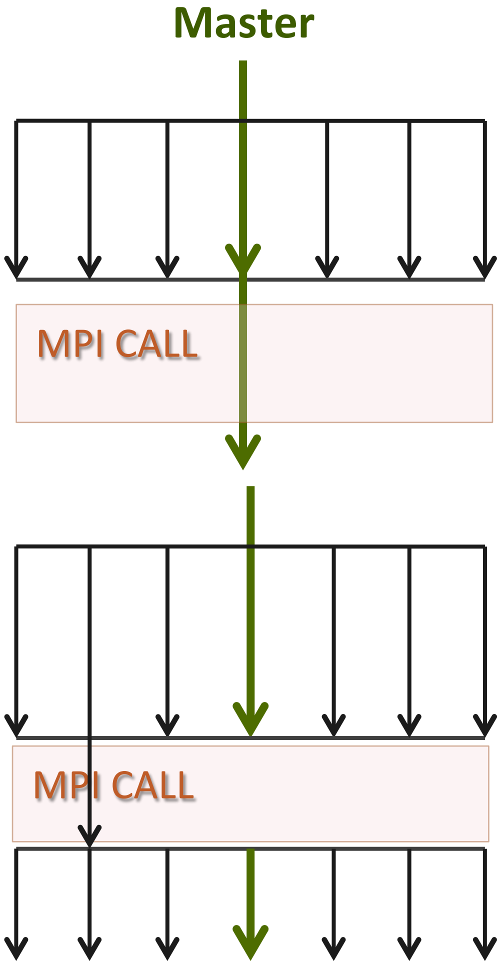
- Threads are only active in the computation phase
- Threads are IDLE during MPI calls

```
#pragma omp parallel for
for (i = 0; i < N; i++) {
        uu[i] = (u[i] + u[i - 1] +  u[i + 1])/5.0;
}

MPI_Function ( );
```

(a) Funneled mode

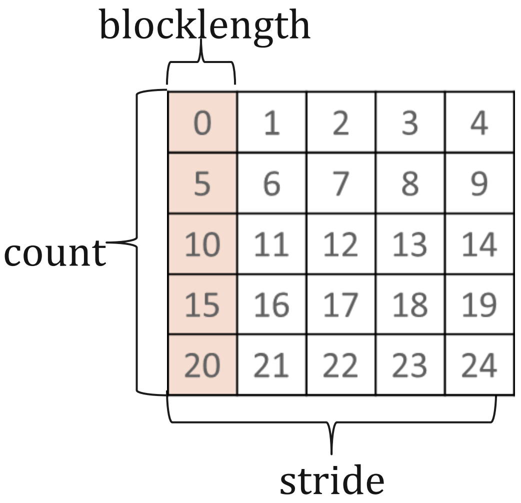
```
#pragma omp parallel
{
        /* user computation */

        #pragma omp single
        MPI_Function ();
}
```

(b) Serialized mode



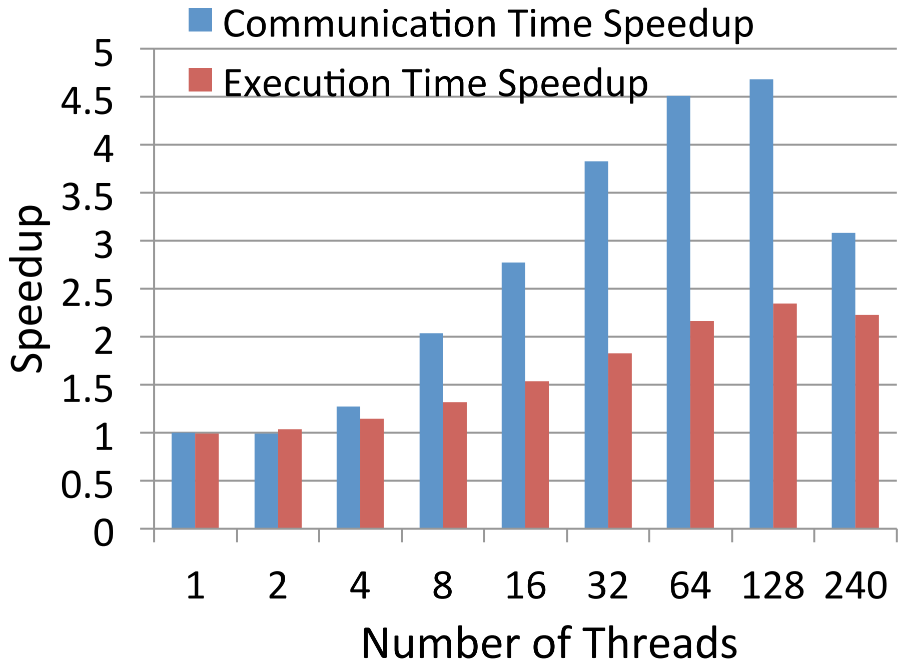
Derived Data Type Packing Processing

- MPI_Pack / MPI_Unpack
- Communication using Derived Data Type
  - Transfer non-contiguous data
  - Pack / unpack data internally



0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24

```
#pragma omp parallel for
for (i=0;  i<count;  i++){
        dest[i] = src[i * stride];
}
```

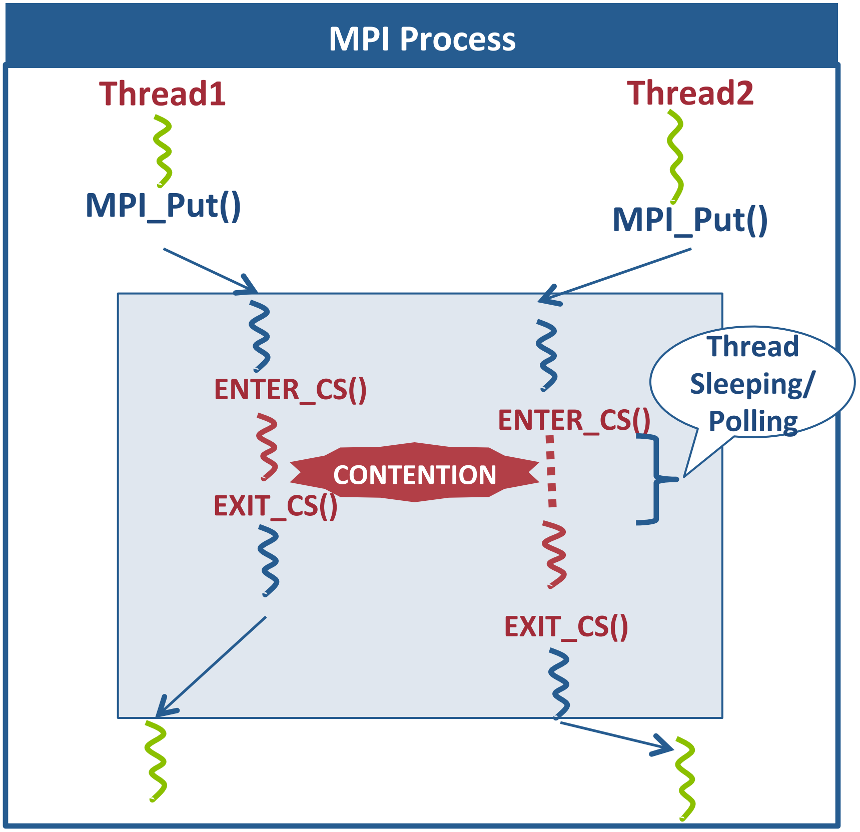


Hybrid MPI+OpenMP NAS Parallel MG benchmark

Contention in a Multithreaded MPI Model

```
MPI_Init_thread(…,MPI_THREAD_MULTIPLE,…);
.
.
#pragma omp parallel
{
    /* Do Work */
    MPI_Put();
    /* Do Work */
}
```

- Multithreaded MPI
  - Threads can make MPI calls concurrently
  - Thread-safety is necessary



Thread-safety can be ensured by:

- **Critical Sections** (Locks)

→ Possible Contention !

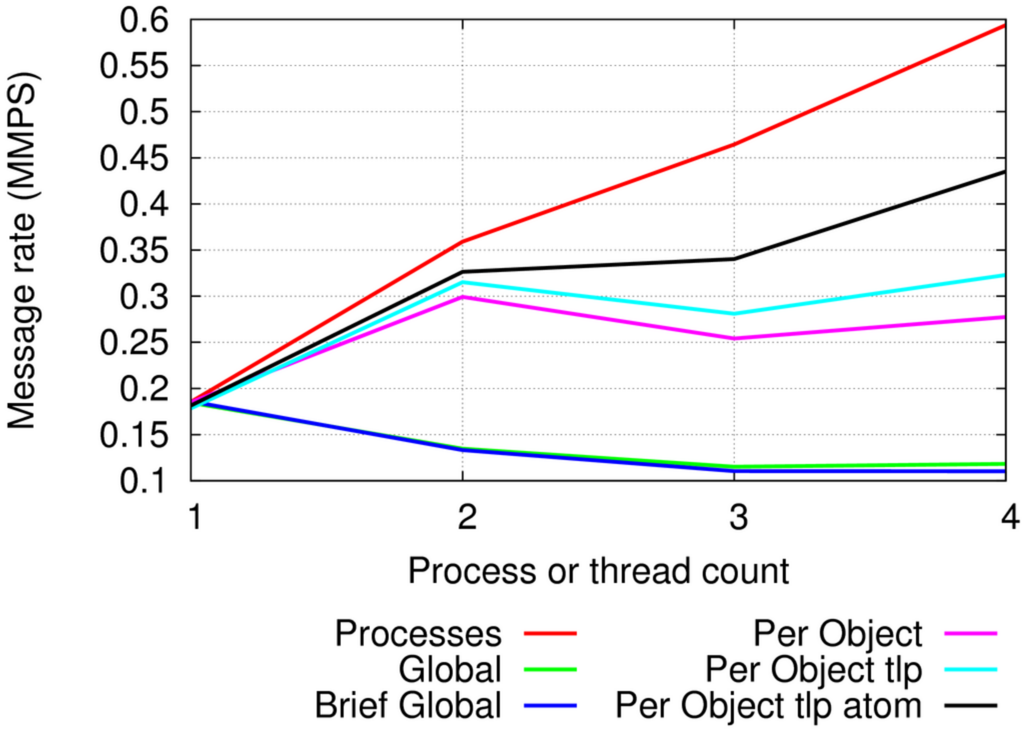
- Using **Lock-Free** algorithms

→ **Non trivial !**

→ **Still does memory barriers**

Several Optimizations Later…

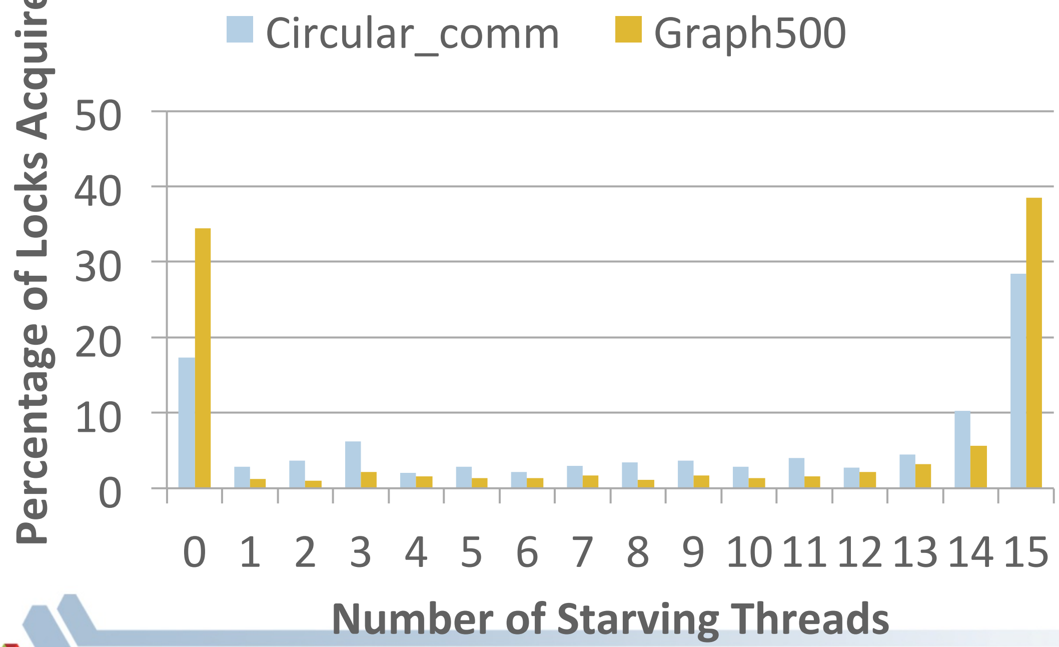
- Reduction of lock granularities
- Thread-local pools to reduce sharing
- Per-object locks
- Some atomic operations (reference counts)
- But the performance scaling was still suboptimal



Hidden Evil: Lock Monopolization (Starvation)

- Implementing critical sections with spin-locks or mutexes
- Watch out: **no fairness** guarantee!

Starvation measurement with 16 processes and 16 threads/nodes

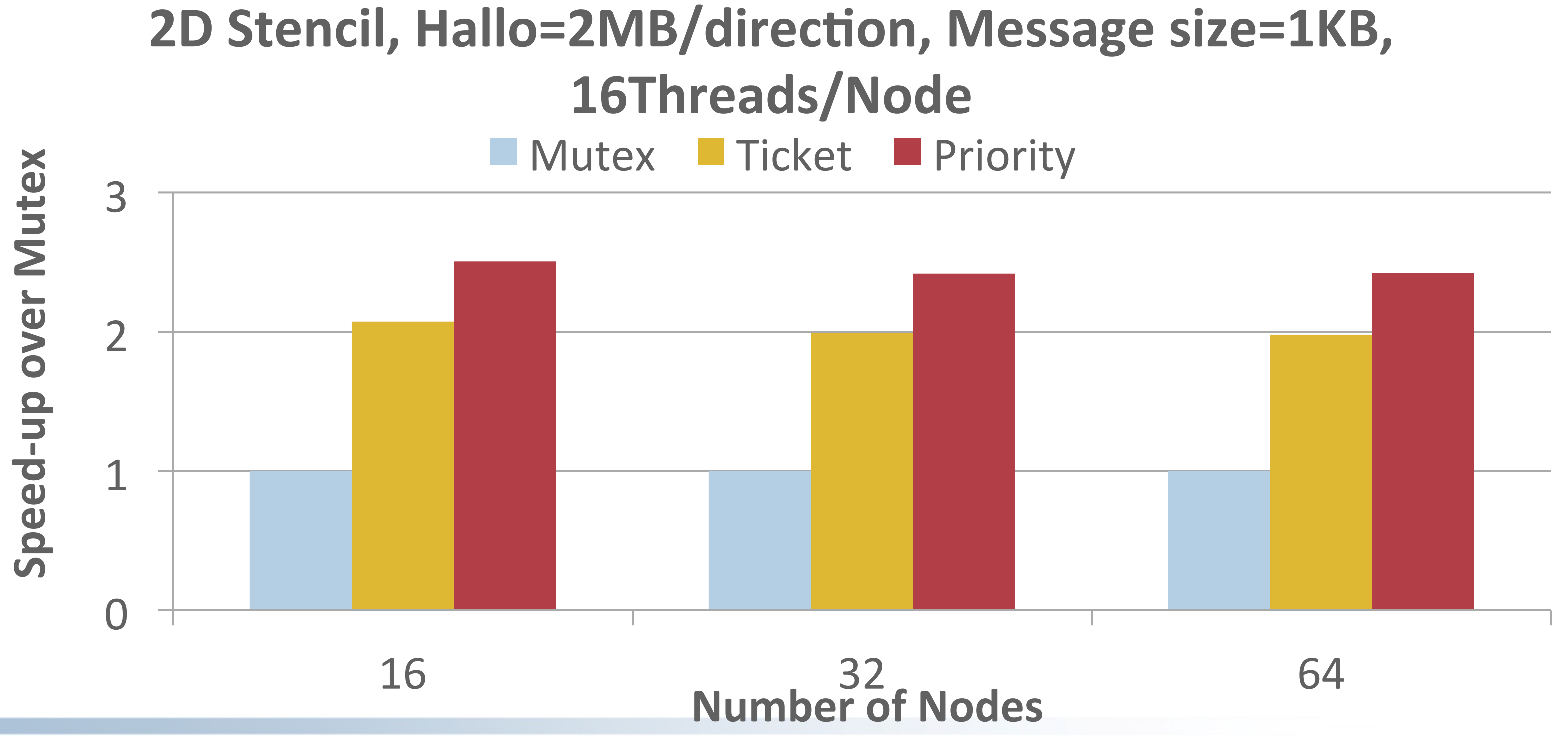


Starvation Detection Algorithm

```
int waiting_threads = 0;
int last_holder;

acquire_lock(L)
{
  bool lock_acquired = false;
  try_lock(L, lock_acquired)
  if ((lock_acquired)                    &&
     (my_thread_id == last_holder) &&
     (waiting_threads > 0))
    STARVATION_CASE;
  else if (!lock_acquired)
  {
    atomic_incr(waiting_threads);
     lock(L);
    atomic_decr(waiting_threads);
  }
  last_holder = my_thread_id;
  return;
}
```

Priority Locking Scheme

- 3 basic locks:
  - One for mutual exclusion in each priority level
  - Another for high priority threads to **block** lower ones
- Watch out: do not forget **fairness** in the **same priority level**
  - Use **exclusively** FIFO locks (Ticket)

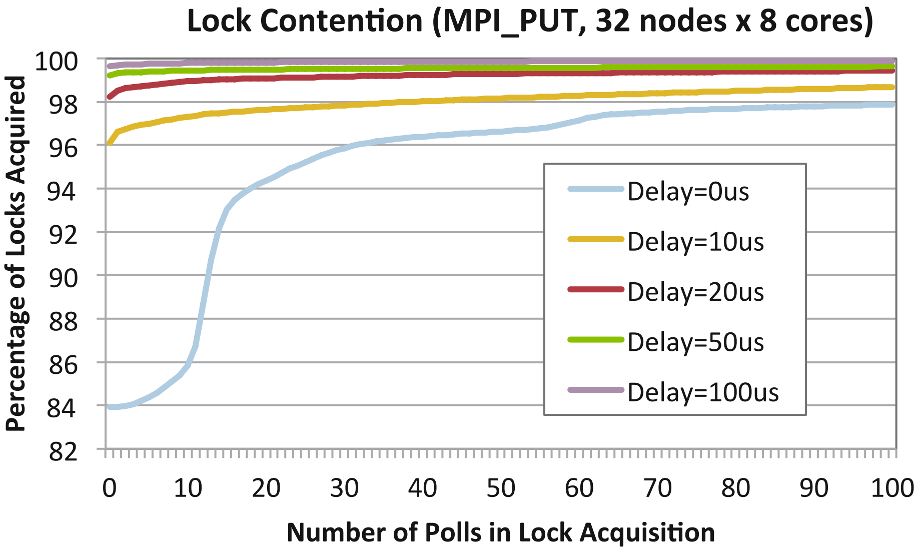


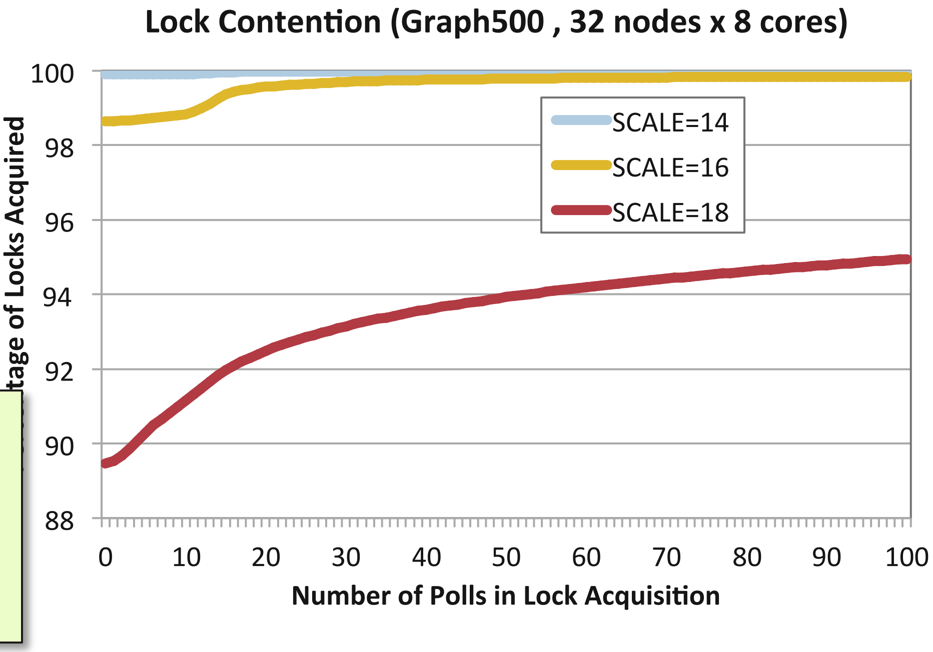
Does Fixing Lock Contention Solve the Problem?

- Spin-lock based critical sections
- Contention metric: **Wasted Polls**
- Test scenarios:
  - Micro-benchmarks
  - HPC applications

```
#pragma omp parallel
{
  for(i=0; i< NITER; i++)
  {
   MPI_Put();
   /*Delay X us*/
   Delay(X)
  }
}
```

“When you have eliminated the impossible, whatever remains, however improbable, must be the truth.” – Sherlock Holmes, Sign of Four, Sir Arthur Conan Doyle



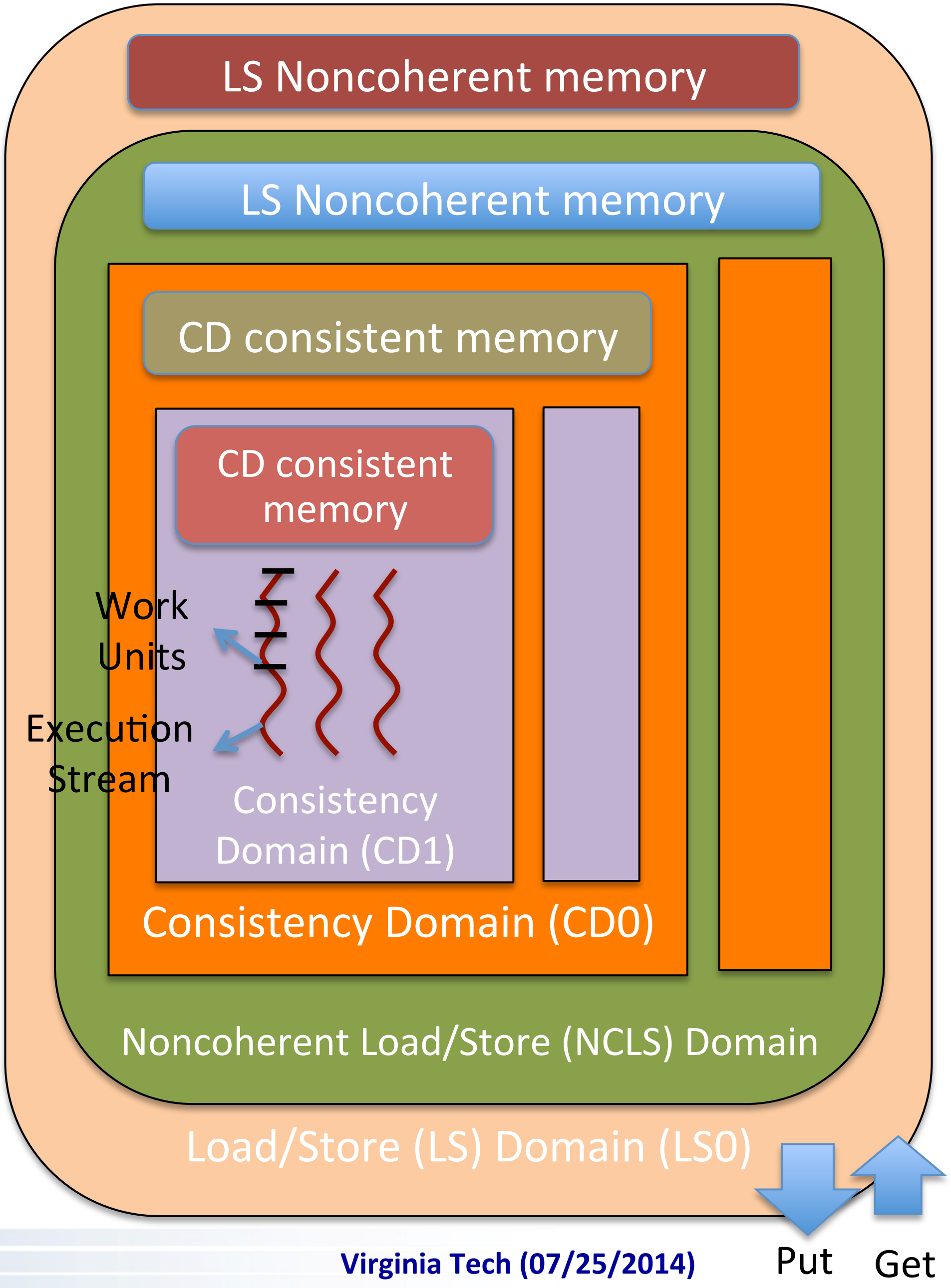


Hybrid MPI+OpenMP (or other threading models)

- Thread execution model exposed to applications is too simplistic
- OpenMP threads can be pthreads (i.e., can execute concurrently) or user-level threads such as qthreads (i.e., might or might not execute concurrently)
  - Not exposed to users/MPI library
- What does this mean to MPI?
  - MPI runtime never knows when two threads can execute concurrently and when they cannot
  - Always has to perform locks and memory consistency calls (memory barriers) even when switching between user-level threads

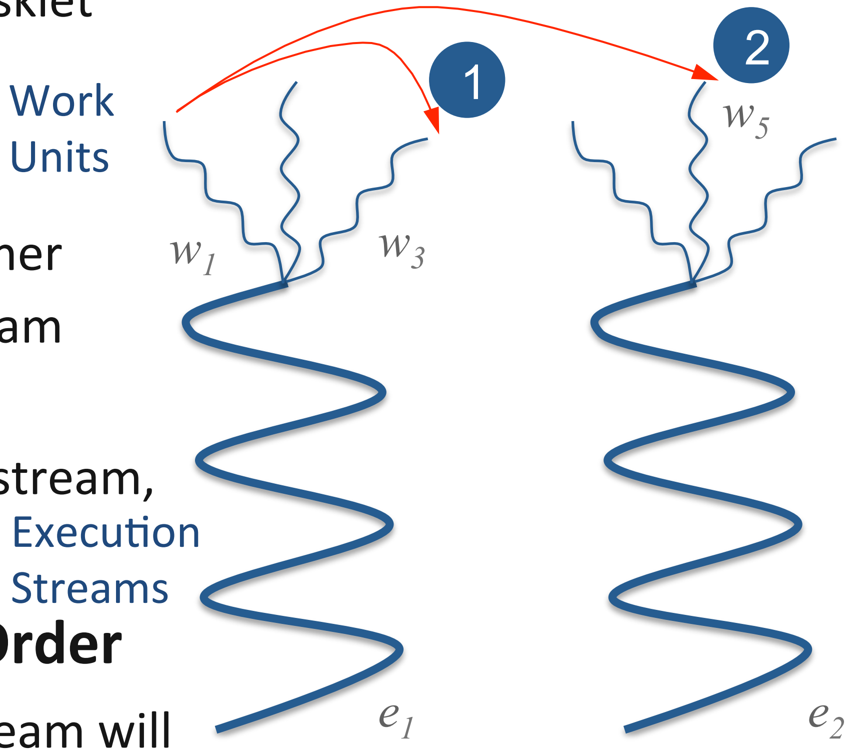
Argobots: Integrated Computation and Data Movement with Lightweight Work Units

- Execution Model
  - **Execution stream**: a thread executed on a hardware processing element
  - **Work units**: a user level thread or a tasklet with a function pointer
- Memory Model
  - **Memory Domains**: A memory consistency call on a big domain also impacts all internal domains
  - **Synchronization**: explicit & implicit memory consistency calls
  - **Network**: PUT/GET, atomic ops



Argobots Ongoing Works: Fine-grained Context-aware Thread Library

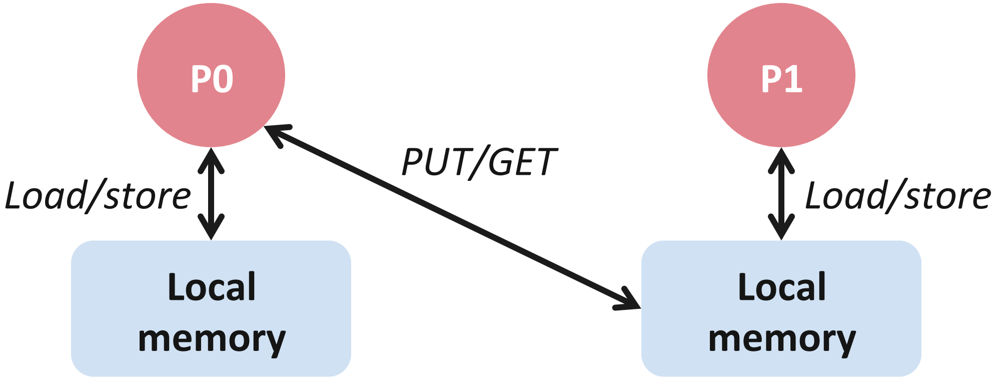
- **Two Level of Threads**
  - execution stream: a normal thread
  - work unit: a user level thread or a tasklet with a function pointer
- **Avoid Unnecessary lock/unlock**
  - Case 1: switch the execution to another work unit in the same execution stream **without unlock**
  - Case 2: switch to another execution stream, call unlock
- **Scheduling Work Units in Batch Order**
  - work units in the same execution stream will be batch executed



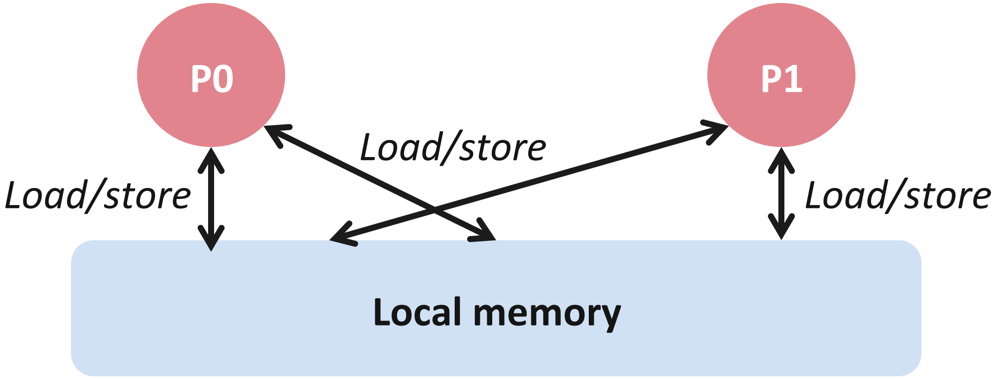
Hybrid Programming with Shared Memory

- MPI-3 allows different processes to allocate shared memory through MPI
  - MPI_Win_allocate_shared
- Uses many of the concepts of one-sided communication
- Applications can do hybrid programming using MPI or load/store accesses on the shared memory window
- Other MPI functions can be used to synchronize access to shared memory regions
- Can be simpler to program than threads

Regular RMA windows vs. Shared memory windows



Traditional RMA windows

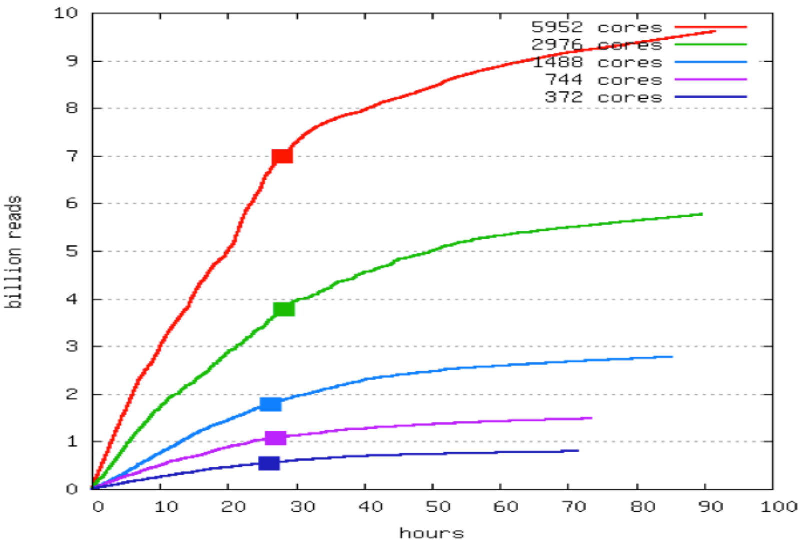


Shared memory windows

- Shared memory windows allow application processes to directly perform load/store accesses on all of the window memory
  - E.g., x[100] = 10
- All of the existing RMA functions can also be used on such memory for more advanced semantics such as atomic operations
- Can be very useful when processes want to use threads only to get access to all of the memory on the node
  - You can create a shared memory window and put your shared data

Case Study: Genome Assembly

Terabase Assembly on Cray XE6



Courtesy Fangfang Xia, Argonne National Laboratory

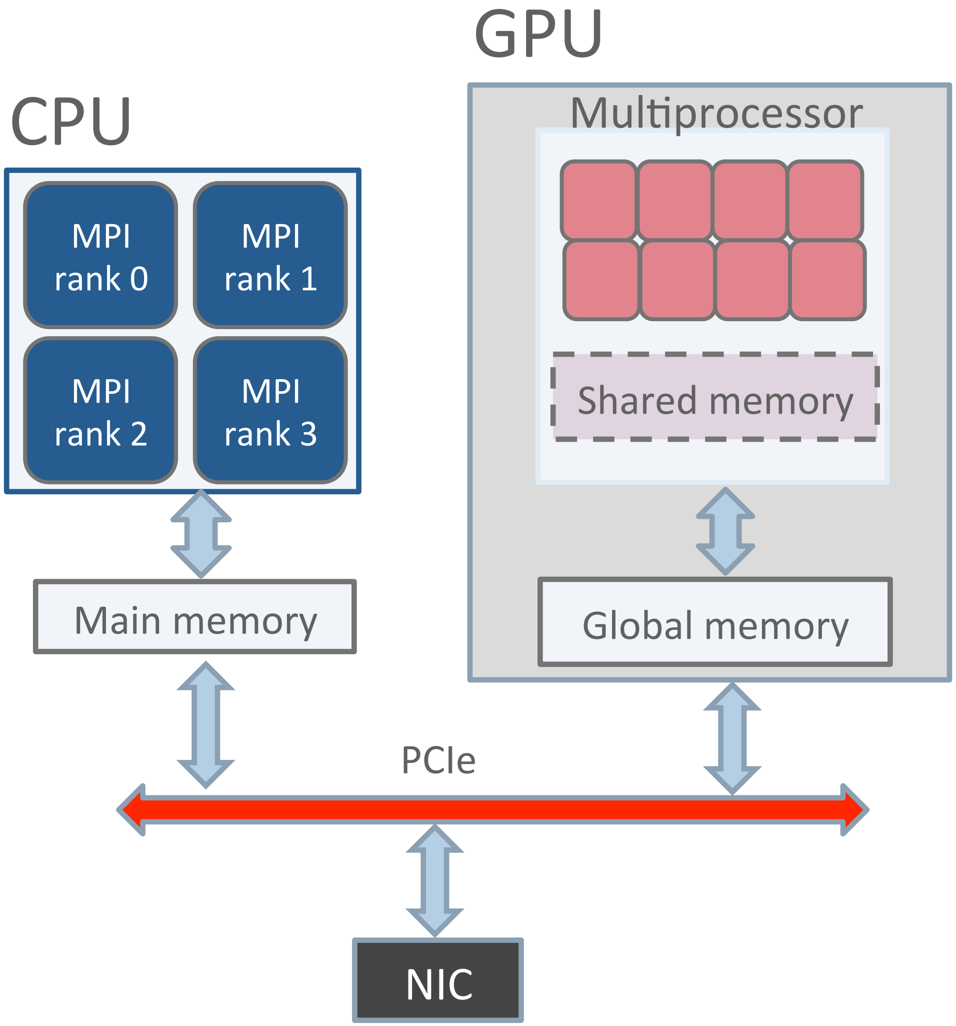
- Largest genome assembly to date: 2.3TB dataset performed with MPI-3 shared memory capability
  - First terascale genome assembly
- Very simple optimization: place all of the node dataset in shared memory and access as read-only data
- Could not use threads because all MPI calls face lock overheads

MPI Myth #3:

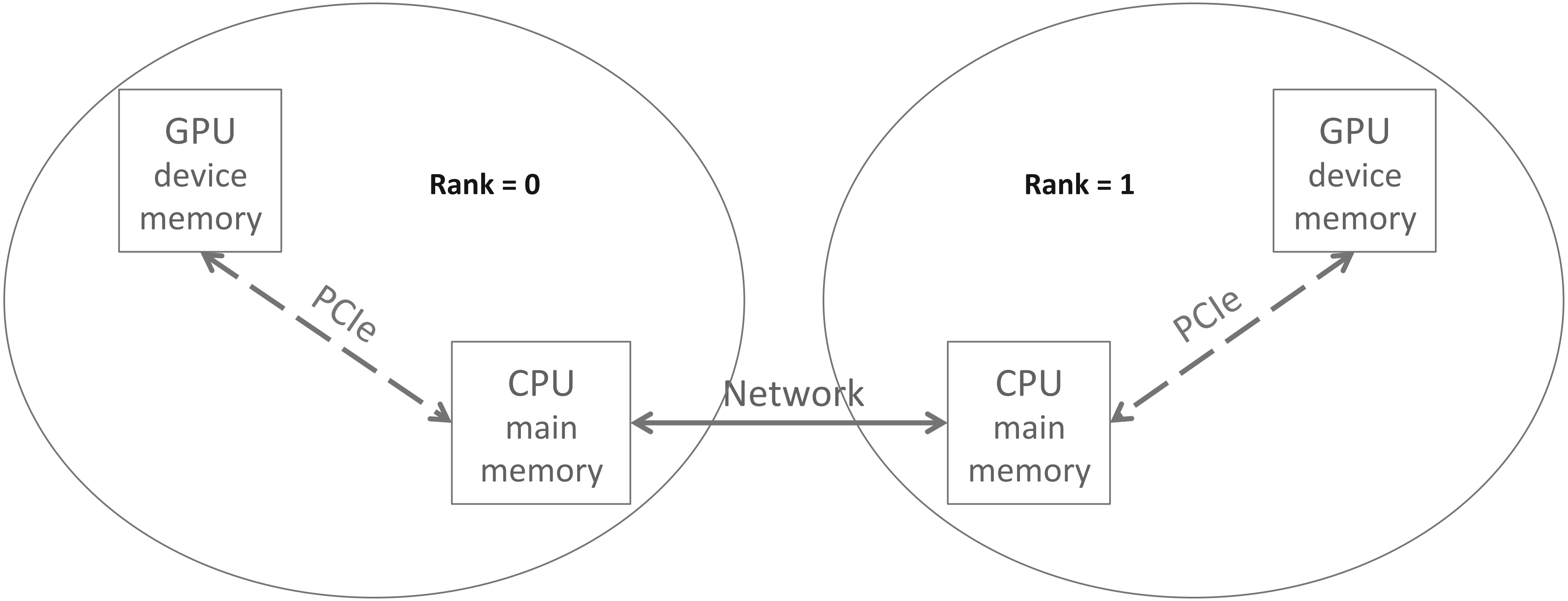
“MPI cannot deal with accelerators”

Example Heterogeneous Architecture: Accelerator Clusters

- Graphics Processing Units (GPUs)
  - Many-core architecture for high performance and efficiency (FLOPs, FLOPs/Watt, FLOPs/$)
  - Prog. Models: CUDA, OpenCL, OpenACC
  - Explicitly managed global memory and separate address spaces
- CPU clusters
  - Most popular parallel prog. model: Message Passing Interface (MPI)
  - Host memory only
- Disjoint Memory Spaces!



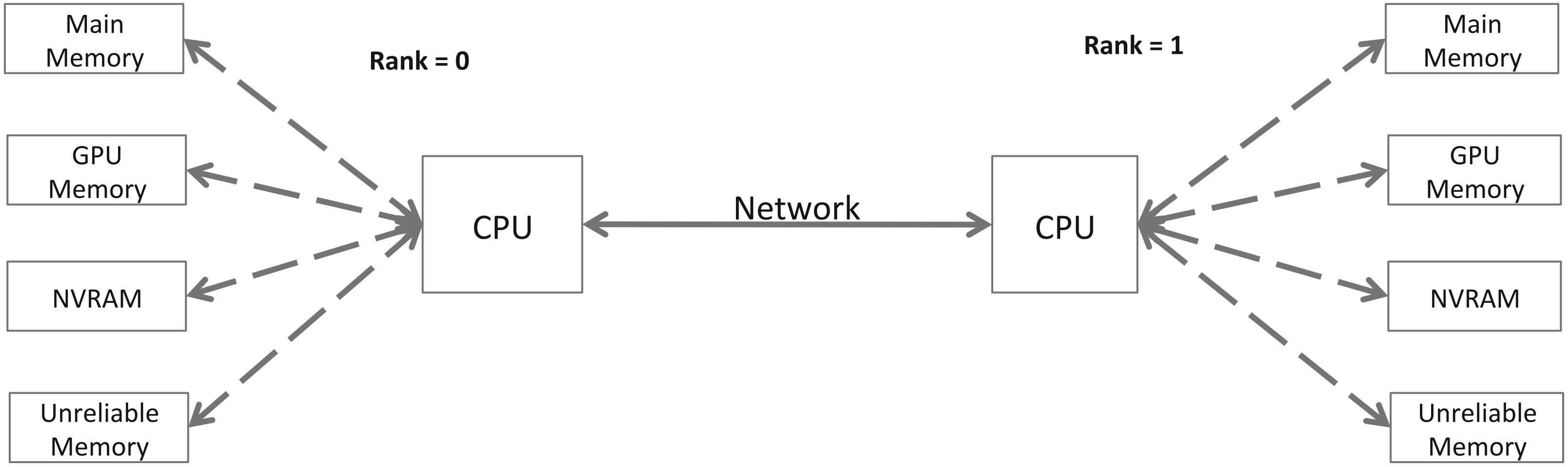
Programming Heterogeneous Memory Systems (e.g: MPI+CUDA)



```
if(rank == 0)
{
  cudaMemcpy(host_buf, dev_buf, D2H)
  MPI_Send(host_buf, .. ..)
}
```

```
if(rank == 1)
{
  MPI_Recv(host_buf, .. ..)
  cudaMemcpy(dev_buf, host_buf, H2D)
}
```

MPI-ACC: A Model for Unified Data Movement



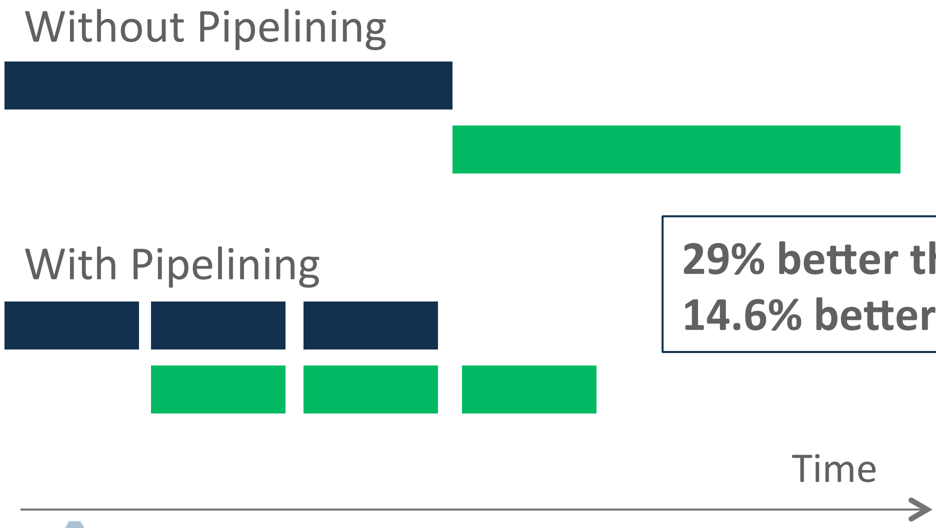
```
if(rank == 0)
{
  MPI_Send(any_buf, .. ..);
}
```

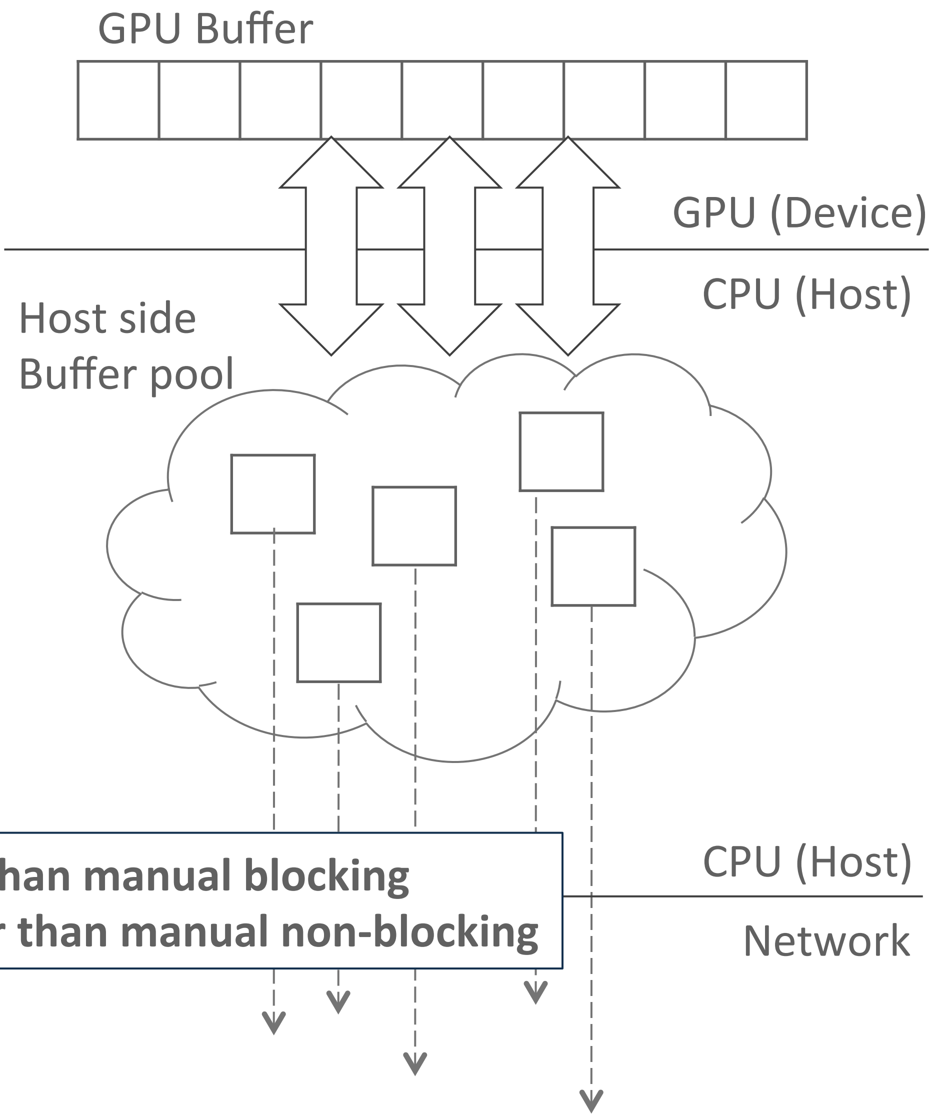
```
if(rank == 1)
{
  MPI_Recv(any_buf, .. ..);
}
```

"MPI-ACC: An Integrated and Extensible Approach to Data Movement in Accelerator-Based Systems", Ashwin Aji, James S. Dinan, Darius T. Buntinas, Pavan Balaji, Wu-chun Feng, Keith R. Bisset and Rajeev S. Thakur. IEEE International Conference on High Performance Computing and Communications (HPCC), 2012

MPI-ACC Optimizations: Pipelined Data Movement

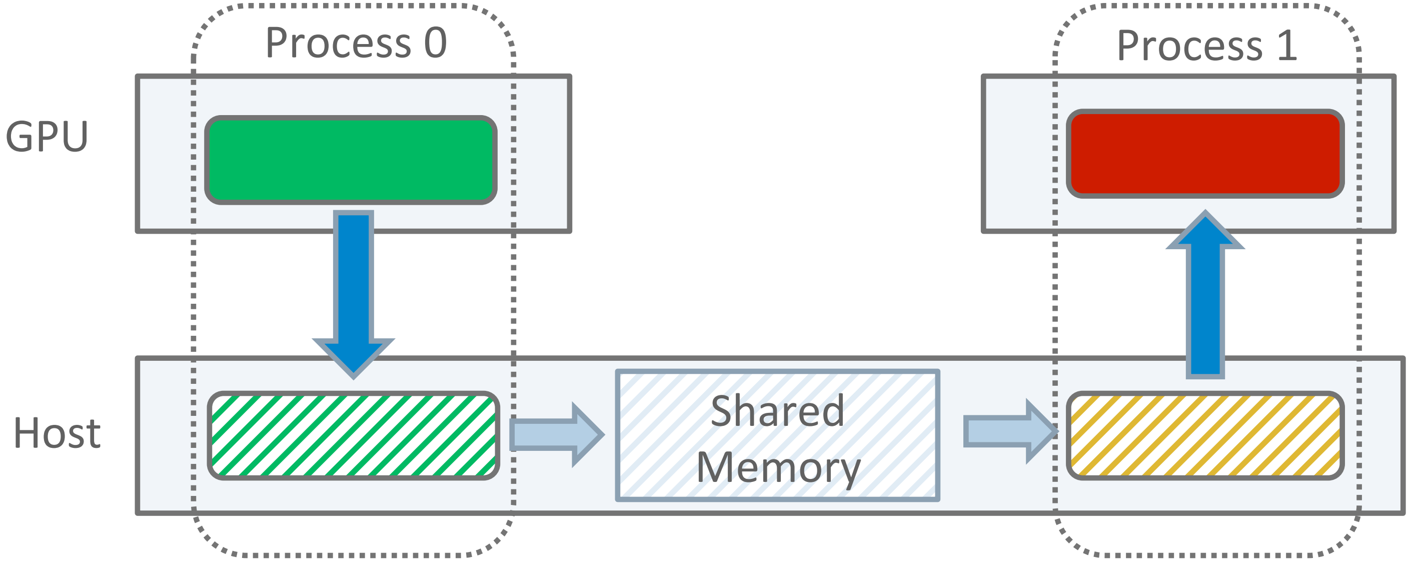
- Host buffers instantiated during MPI_Init and destroyed during MPI_Finalize
- Classic double-buffering technique
- Intercepted the MPI progress engine
- When possible (e.g., newer CUDA), multiple streams are used for improved DMA utilization
- Architecture-specific optimizations: GPU Direct





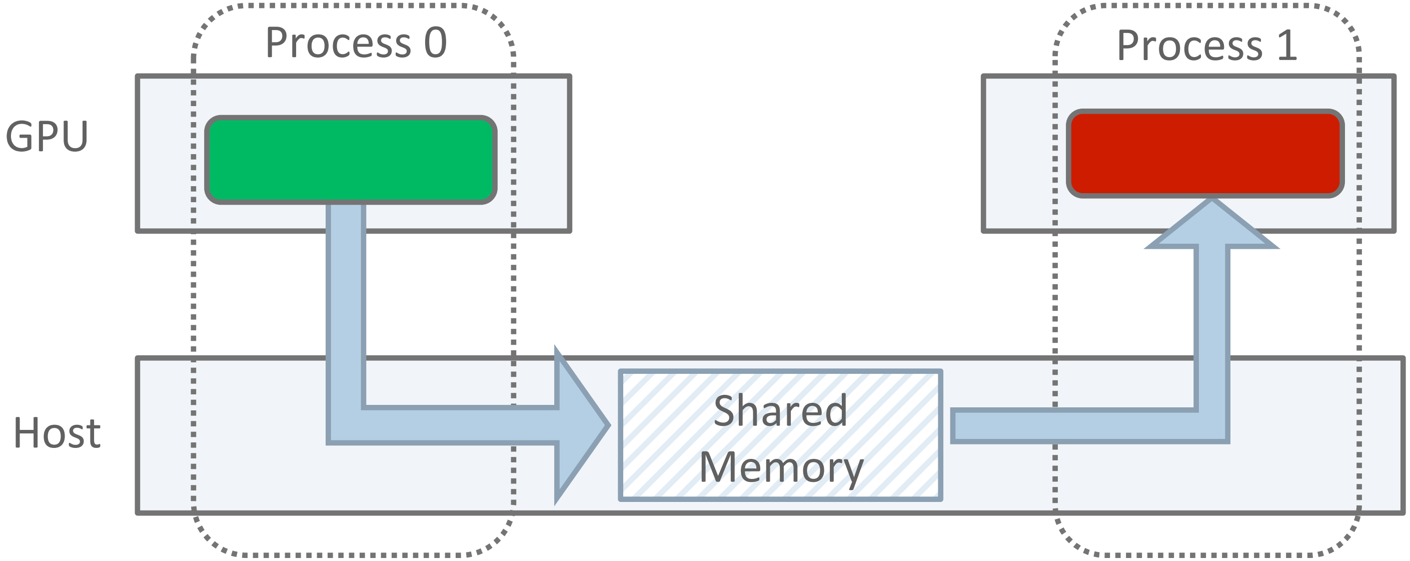
29% better than manual blocking
14.6% better than manual non-blocking

Traditional Intranode Communication



- Communication without accelerator integration
  - 2 PCIe data copies + 2 main memory copies
  - Transfers are serialized

Eliminating Extra Copies



- Integration allows direct transfer into shared memory buffer
- LMT: sender and receiver drive transfer concurrently
  - Pipeline data transfer
  - Full utilization of PCIe links

"Optimizing GPU-to-GPU intra-node communication in MPI", Feng Ji, James S. Dinan, Darius T. Buntinas, Pavan Balaji, Xiaosong Ma and Wu-chun Feng. Workshop on Accelerators and Hybrid Exascale Systems (AsHES); in conjunction with the IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2012

GPU Direct and CUDAIPC optimizations

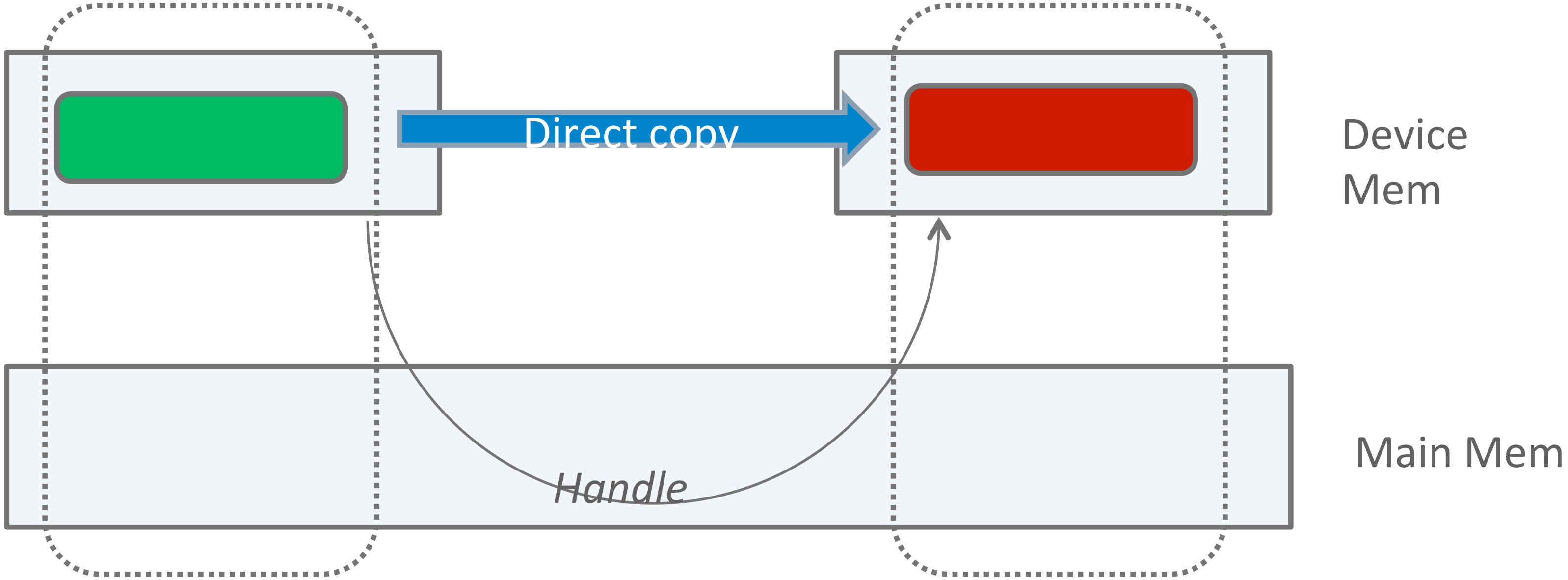
- GPUDirect: DMA-driven peer GPU copy
- CUDAIPC: exporting a GPU buffer to a different process

Process 0

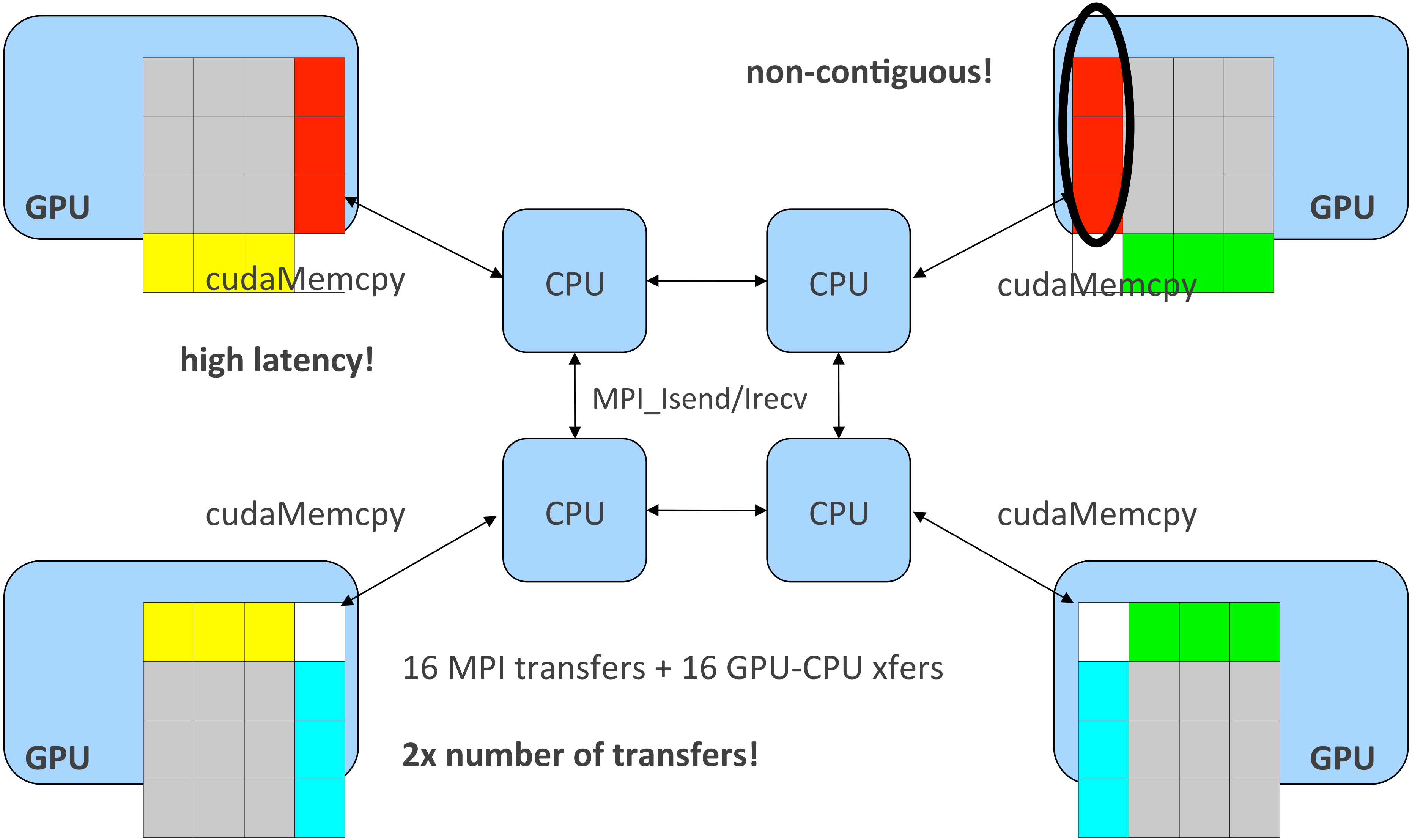
cudaIpcGetMemHandle(&*handle*, d_ptr);

Process 1

cudaIpcOpenMemHandl(&d_ptr_src, *handle*);
cudaMemcpy(d_ptr, d_ptr_src, …);



MPI + GPU Example - Stencil Computation



non-contiguous!

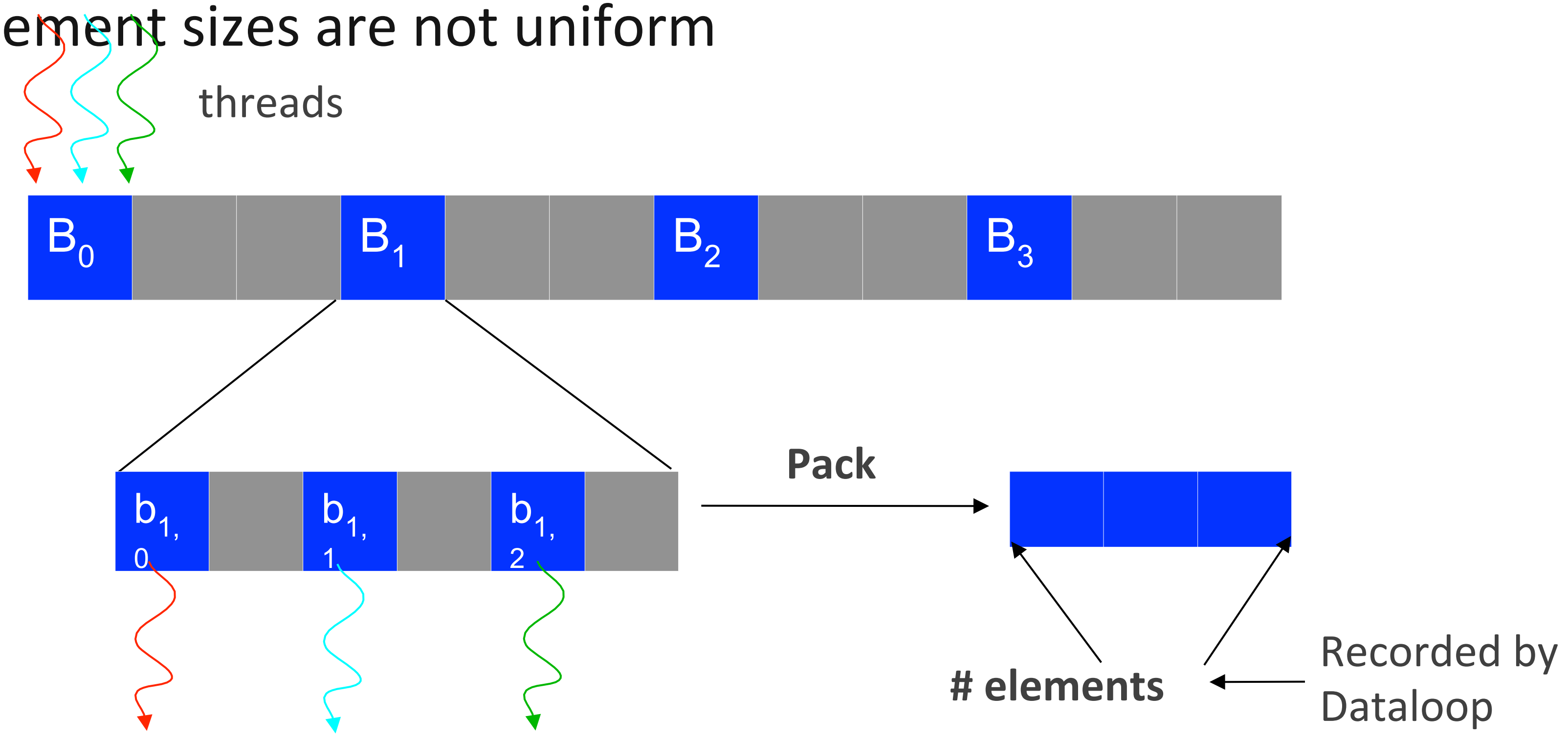
high latency!

16 MPI transfers + 16 GPU-CPU xfers

2x number of transfers!

GPU optimizations for Data Packing

- Element-wise traversal by different threads
- Embarrassingly parallel problem, except for structs, where element sizes are not uniform

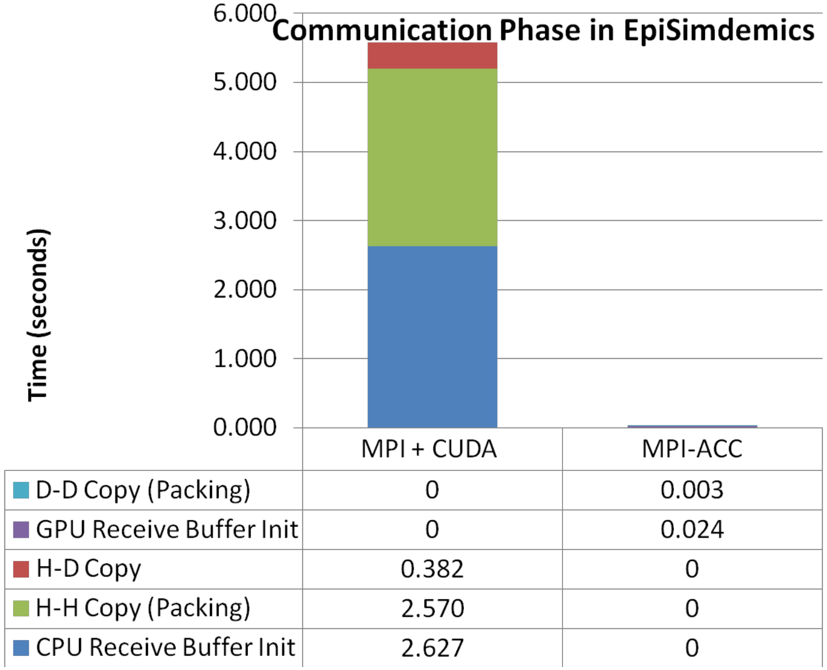


traverse by **element #**, read/write using **extent/size**

MPI-ACC with EpiSimdemics

- Enables GPU pointers in MPI calls
  - (CUDA & OpenCL); generic support for heterogeneous memory subsystems is coming
- Coding productivity + performance
- Generic approach, feature independent (UVA not needed)
- Datatype and communicator attrs
  - Pointer location
  - Streams / queues

Evaluating Epidemiology Simulation with MPI-ACC



	MPI + CUDA	MPI-ACC
🟦 D-D Copy (Packing)	0	0.003
🟪 GPU Receive Buffer Init	0	0.024
🟥 H-D Copy	0.382	0
🟩 H-H Copy (Packing)	2.570	0
🟦 CPU Receive Buffer Init	2.627	0

AM Aji, LS Panwar, F Ji, M Chabbi, K Murthy, P Balaji, KR Bisset, JS Dinan, W Feng, J Mellor-Crummey, X Ma, and RS Thakur. *“On the efficacy of GPU-integrated MPI for scientific applications”*, in HPDC 2013.

AM Aji, P Balaji, JS Dinan, W Feng, and RS Thakur. “*Synchronization and ordering semantics in hybrid MPI+GPU programming”*. In AsHES 2013.

J Jenkins, JS Dinan, P Balaji, NF Samatova, and RS Thakur. “*Enabling fast, noncontiguous GPU data movement in hybrid MPI+GPU environments.* In Cluster 2012.

AM Aji, JS Dinan, DT Buntinas, P Balaji, W Feng, KR Bisset, and RS Thakur. “*MPI-ACC: an integrated and extensible approach on data movement in accelerator-based systems”*. In HPCC 2012.

F Ji, AM Aji, JS Dinan, DT Buntinas, P Balaji, RS Thakur, W Feng, and X Ma. *DMA-assisted, intranode communication in GPU accelerated systems.* In HPCC 2012.

F Ji, AM Aji, JS Dinan, DT Buntinas, P Balaji, W Feng, and X Ma. “*Efficient Intranode Communication in GPU-accelerated systems”*. In IPDPSW 2012.

MPI Myth #4:

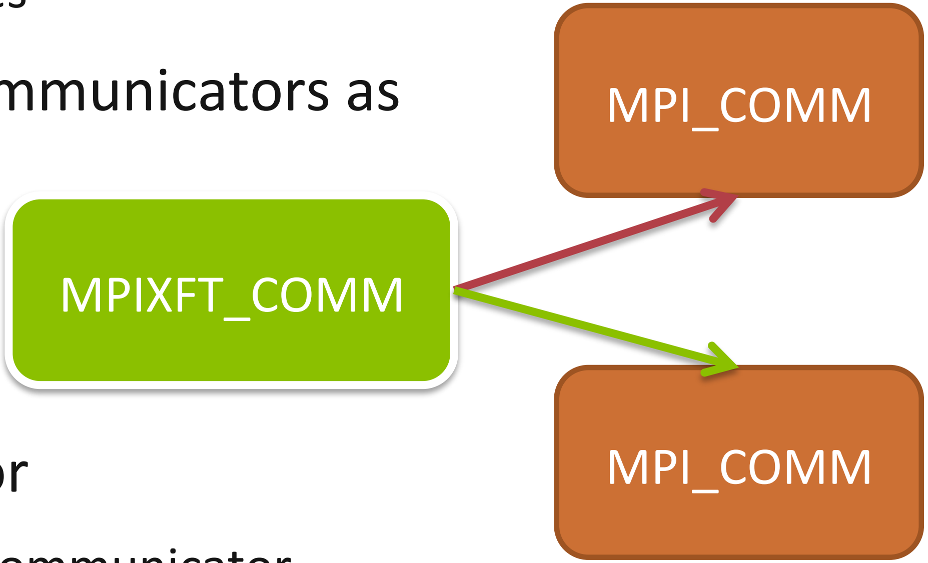
“MPI cannot deal with faults”

CPU (Total System) Failures

- Generally will result in a process failure from the perspective of other (off-node) processes
- Need to recover/repair lots of parts of the system
  - Communication library (MPI)
  - Computational capacity (if necessary)
  - Data
    - C/R, ABFT, Natural Fault Tolerance, etc.
- MPI-3 has the theoretical ability to deal with faults, but the user has to do a bunch of bookkeeping to make that happen
  - New communicator has to be created
  - All requests have to be kept track of and migrated to the new communicator
  - Need to watch out for failure messages from other processes

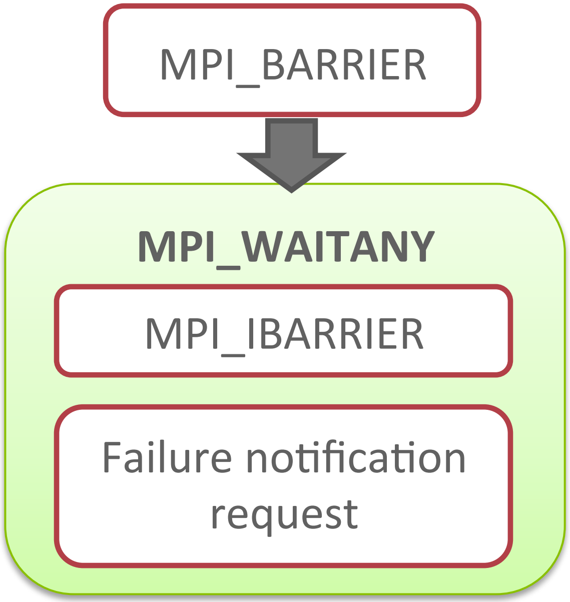
MPIXFT: MPI-3 based library for FT bookkeeping

- Lightweight virtualization infrastructure
  - Gives users virtual communicators and requests and internally manages the real ones
- Automatically repairs MPI communicators as failures occur
  - Handles running in n-1 model



- Virtualizes MPI Communicator
  - User gets an MPIXFT wrapper communicator
  - On failure, the underlying MPI communicator is replaced with a new, working communicator

MPIXFT Design



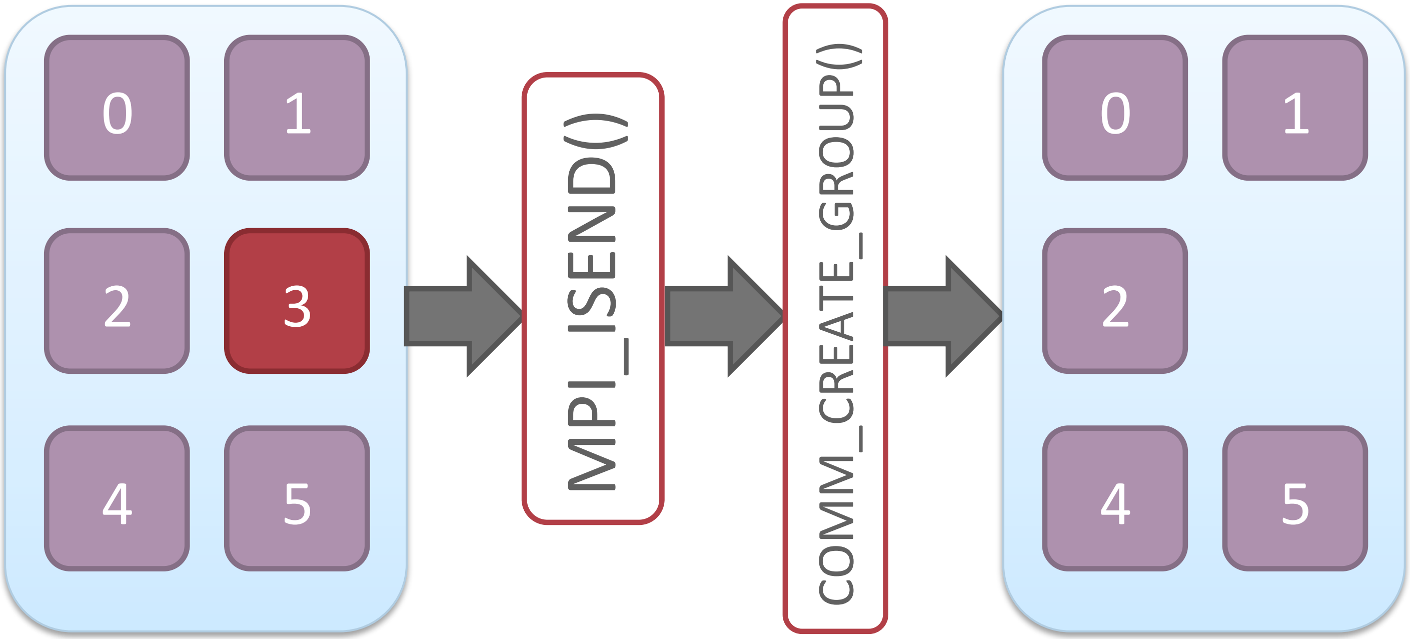
MPI_BARRIER

MPI_WAITANY

MPI_IBARRIER

Failure notification request

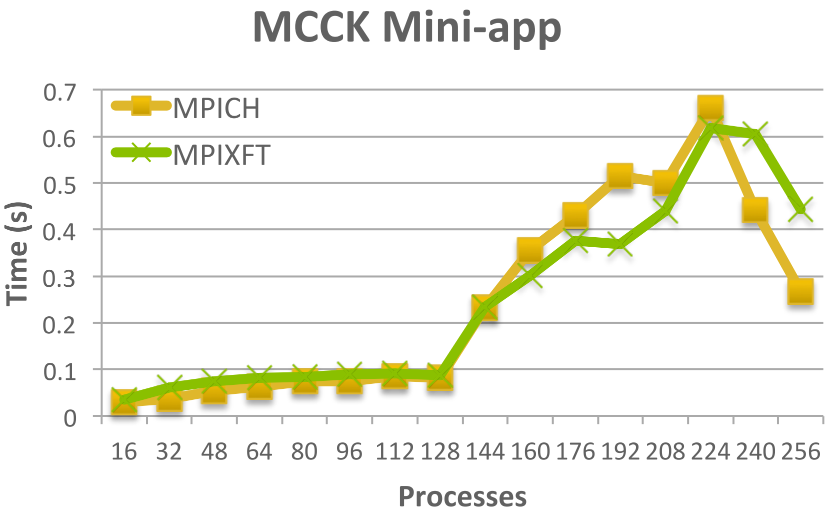
- Possible because of new MPI-3 capabilities
  - Non-blocking equivalents for (almost) everything
  - MPI_COMM_CREATE_GROUP



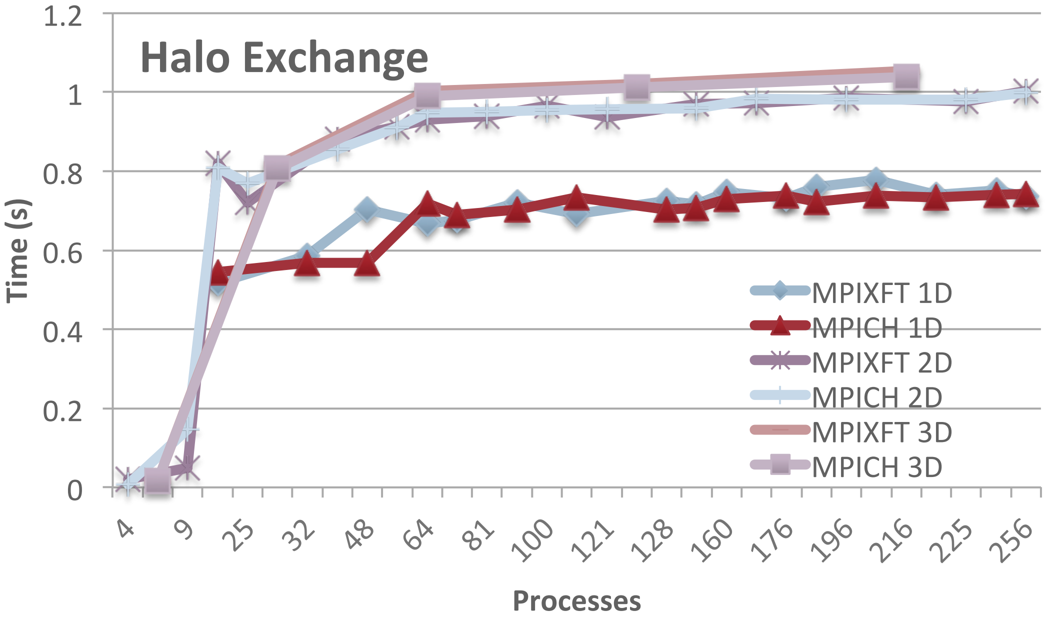
MPI_ISEND()

COMM_CREATE_GROUP()

MIPXFT Results



- MCCK Mini-app
  - Domain decomposition communication kernel
  - Overhead within standard deviation



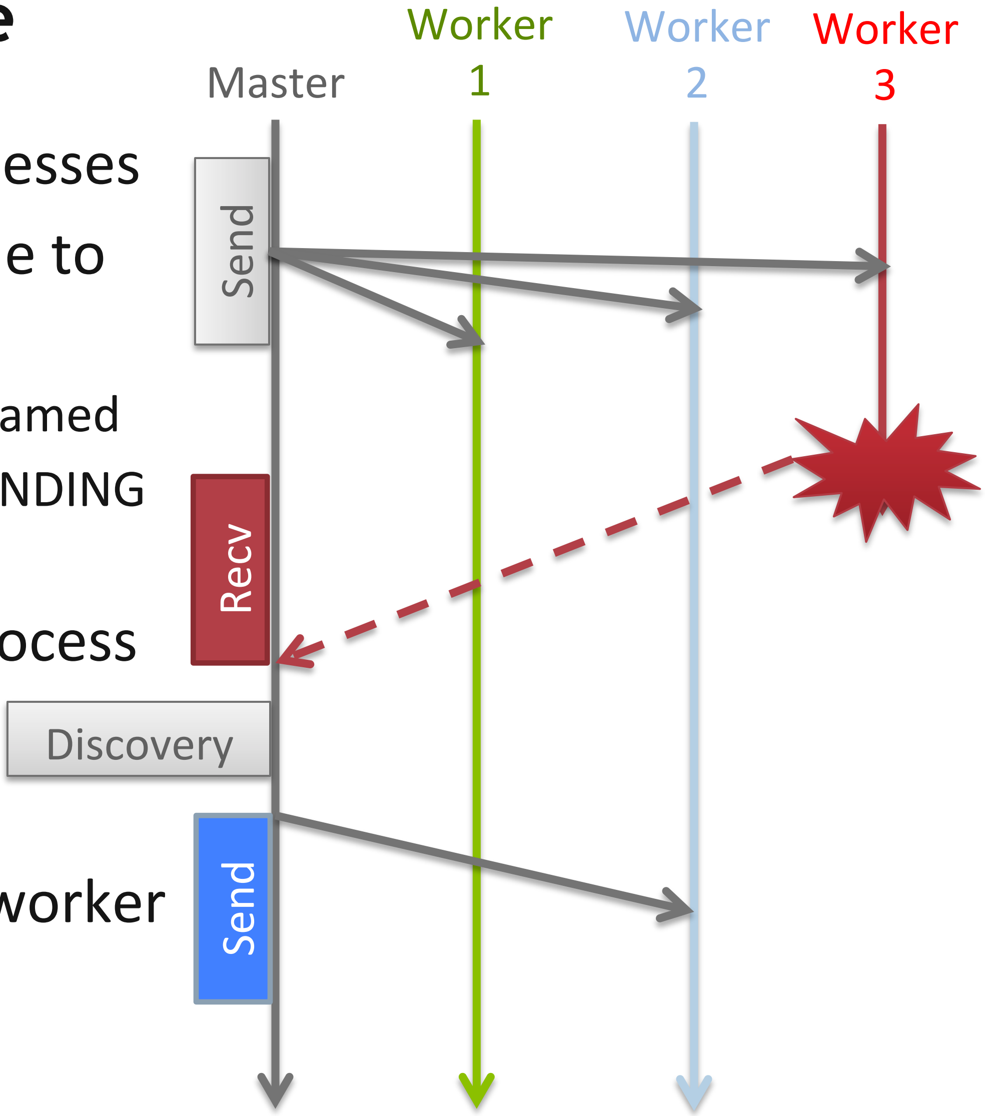
- Halo Exchange (1D, 2D, 3D)
  - Up to 6 outstanding requests at a time
  - Very low overhead

User Level Failure Mitigation

- Proposed change to MPI Standard for MPI-4
- Repair MPI after process failure
  - Enable more custom recovery than MPIXFT
- Does not pick a particular recovery technique as better or worse than others
- Introduce minimal changes to MPI
- Works around performance problems with MPIXFT
- Treat process failure as fail-stop failures
  - Transient failures are masked as fail-stop
- Ability to notify remaining processes on errors

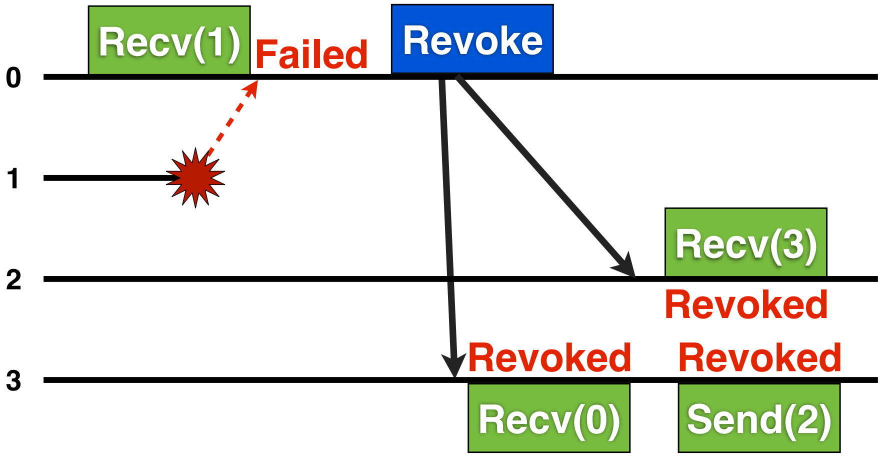
Recovery with only notification Master/Worker Example

- Post work to multiple processes
- MPI_Recv returns error due to failure
  - MPI_ERR_PROC_FAILED if named
  - MPI_ERR_PROC_FAILED_PENDING if wildcard
- Master discovers which process has failed with ACK/GET_ACKED
- Master reassigns work to worker 2



Failure Propagation

- When necessary, manual propagation is available.
  - **MPI_Comm_revoke**(MPI_Comm comm)
    - Interrupts all non-local MPI calls on all processes in comm.
    - Once revoked, all non-local MPI calls on all processes in comm will return MPI_ERR_REVOKED.
      - Exceptions are MPI_COMM_SHRINK and MPI_COMM_AGREE (later)
  - Necessary for deadlock prevention
- Often unnecessary
  - Let the application discover the error as it impacts correct completion of an operation.



MPI Myth #5:

"MPI is too hard to program"

Productivity

- Well, that one's actually true ☺
  - It's meant to be a low-level portable runtime on top of which higher-level programming models should be developed
- A programming model has to pick a tradeoff between programmability, portability, and performance
  - MPI has chosen to be a high-performance/portable programming model
  - Focus has been on completeness and ability to help real and complex applications meet their computational needs
- ***MPI's goal is not to make simple programs easy to write, but to make complex programs possible to write***

Take Away

- MPI has a lot to offer for Exascale systems
  - MPI-3 and MPI-4 incorporate some of the research ideas
  - MPI implementations moving ahead with newer ideas for Exascale
  - Several optimizations inside implementations, and new functionality
- The work is not done, still a long way to go
  - But a start-from-scratch approach is neither practical nor necessary
  - Invest in orthogonal technologies that work with MPI (MPI+X)
- I don’t know what tomorrow’s scientific computing language will look like, but I know it will be called Fortran
- I don’t know what tomorrow’s parallel programming model will look like, but I know it will be called MPI (+X)

Funding Acknowledgments

Funding Grant Providers

Argonne NATIONAL LABORATORY

DEPARTMENT OF ENERGY · UNITED STATES OF AMERICA

NSF

U.S. DEPARTMENT OF ENERGY Office of Science

NNSA National Nuclear Security Administration

Sandia National Laboratories

Microsoft

NCSA

Infrastructure Providers

Argonne NATIONAL LABORATORY

DEPARTMENT OF ENERGY · UNITED STATES OF AMERICA

NSF

U.S. DEPARTMENT OF ENERGY Office of Science

Microsoft

Office of Science INCITE Advancing America's Science and Industrial Competitiveness

NCSA

intel

IBM

Programming Models and Runtime Systems Group

Group Lead

- Pavan Balaji (computer scientist)

Current Staff Members

- Wesley Bland (postdoc)
- Huiwei Lu (postdoc)
- Antonio Pena (postdoc)
- Ken Raffenetti (developer)
- Sangmin Seo (postdoc)
- Junchao Zhang (postdoc)

Other Current Affiliates

- Xiaomin Zhu (visiting scholar)

Past Staff Members

- Darius T. Buntinas (developer)
- James S. Dinan (postdoc)
- David J. Goodell (developer)
- Ralf Gunter (research associate)
- Yuqing Xiong (visiting scholar)

Current and Past Students

- Ashwin Aji (Ph.D.)
- Md. Humayun Arafat (Ph.D.)
- Alex Brooks (Ph.D.)
- James S. Dinan (Ph.D.)
- Piotr Fidkowski (Ph.D.)
- Priyanka Ghosh (Ph.D.)
- Sayan Ghosh (Ph.D.)
- Jichi Guo (Ph.D.)
- Yanfei Guo (Ph.D.)
- Amer Halim (Ph.D.)
- Marius Horga (M.S.)
- John Jenkins (Ph.D.)
- Feng Ji (Ph.D.)
- Ping Lai (Ph.D.)
- Palden Lama (Ph.D.)
- Yan Li (Ph.D.)
- Huiwei Lu (Ph.D.)
- Ganesh Narayanaswamy (M.S.)
- Qingpeng Niu (Ph.D.)
- Ziaul Haque Olive (Ph.D.)
- David Ozog (Ph.D.)
- Sreeram Potluri (Ph.D.)
- Li Rao (M.S.)
- Gopal Santhanaraman (Ph.D.)
- Thomas Scogland (Ph.D.)
- Min Si (Ph.D.)
- Brian Skjerven (Ph.D.)
- Rajesh Sudarsan (Ph.D.)
- Lukasz Wesolowski (Ph.D.)
- Shucai Xiao (Ph.D.)
- Chaoran Yang (Ph.D.)
- Boyu Zhang (Ph.D.)
- Xiuxia Zhang (Ph.D.)
- Xin Zhao (Ph.D.)

Advisory Board

- Pete Beckman (senior scientist)
- Rusty Lusk (retired, STA)
- Marc Snir (division director)
- Rajeev Thakur (deputy division director)

Web: http://www.mcs.anl.gov/~balaji   Email: balaji@mcs.anl.gov

Group website: http://collab.mcs.anl.gov/PMRS

U.S. DEPARTMENT OF ENERGY