HPC in Phase Change for Exascale Computing

Thomas Sterling

Professor of Informatics and Computing, Indiana University

Chief Scientist and Executive Associate Director
Center for Research in Extreme Scale Technologies (CREST)
School of Informatics and Computing
Indiana University

March 30, 2014

CENTER FOR RESEARCH IN EXTREME SCALE TECHNOLOGIES
INDIANA UNIVERSITY
Pervasive Technology Institute

Introduction

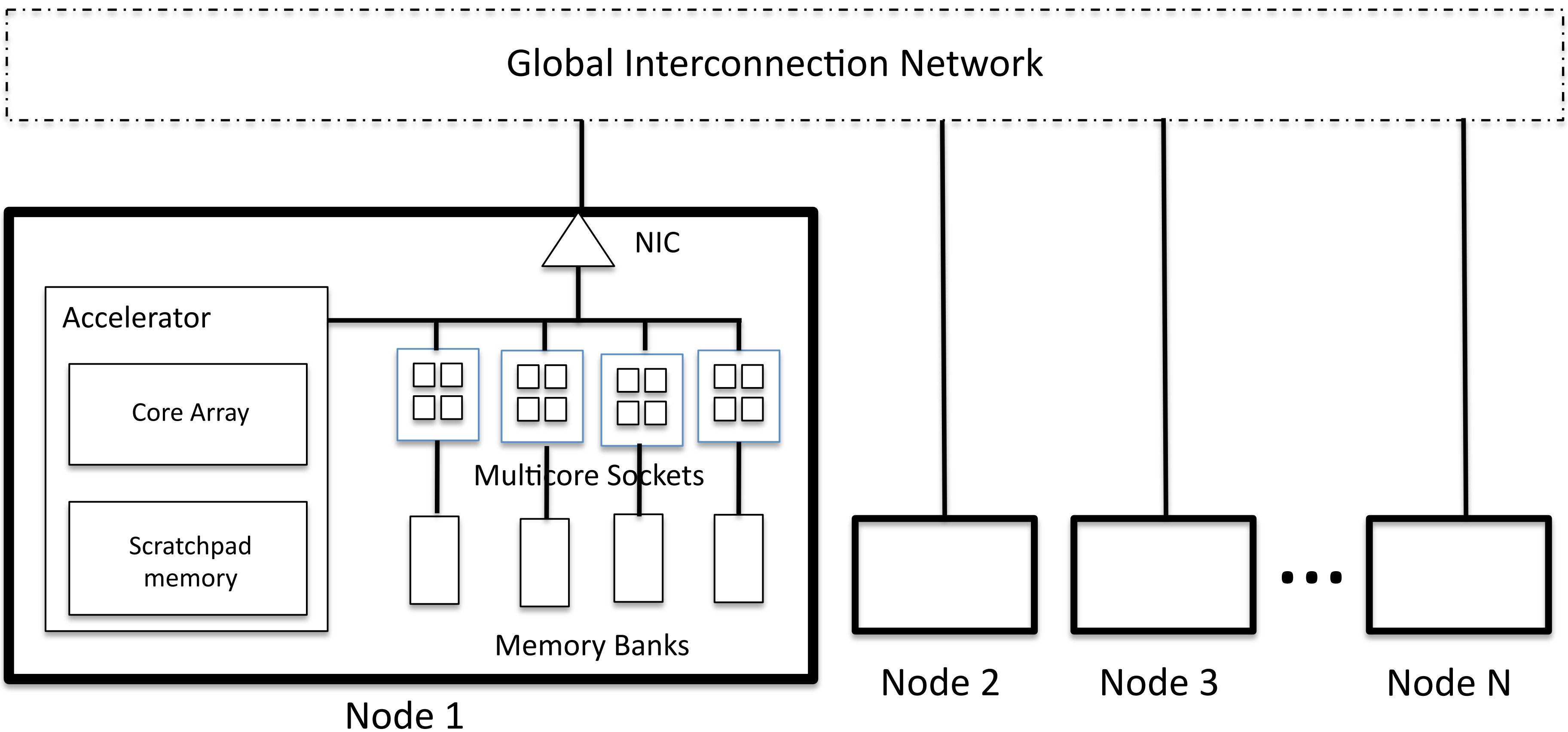
- Exascale computing will demand innovations greater than required for Petaflops, 6 years ago
  - Computer architecture
  - Parallel programming models
  - System software
- 2 Classes of Exascale computing
  - Evolutionary extensions of conventional heterogeneous multicore
  - Revolutionary runtime software based global address space
- Break from the past through a new execution model
  - To address starvation, latency, overhead, contention, energy, & reliability
  - Dynamic adaptive control of resource management and task scheduling
  - Achieve dramatic increase in efficiency and scalability with productivity

Conventional HPC

Tianhe-2
55 Petaflops peak performance
33.9 Petaflops Linpack Rmax
1,375 Terabytes memory
Intel Xeon Phi Accelerator
24 Mwatts power
NUDT deployed
Inspur manufacturer

Titan
27 Petaflops peak performance
17.5 Petaflops Linpack Rmax
693 Terabytes memory
NVIDIA Tesla Accelerator GPU
8.2 MWatts power
ORNL deployed
Cray manufacturer

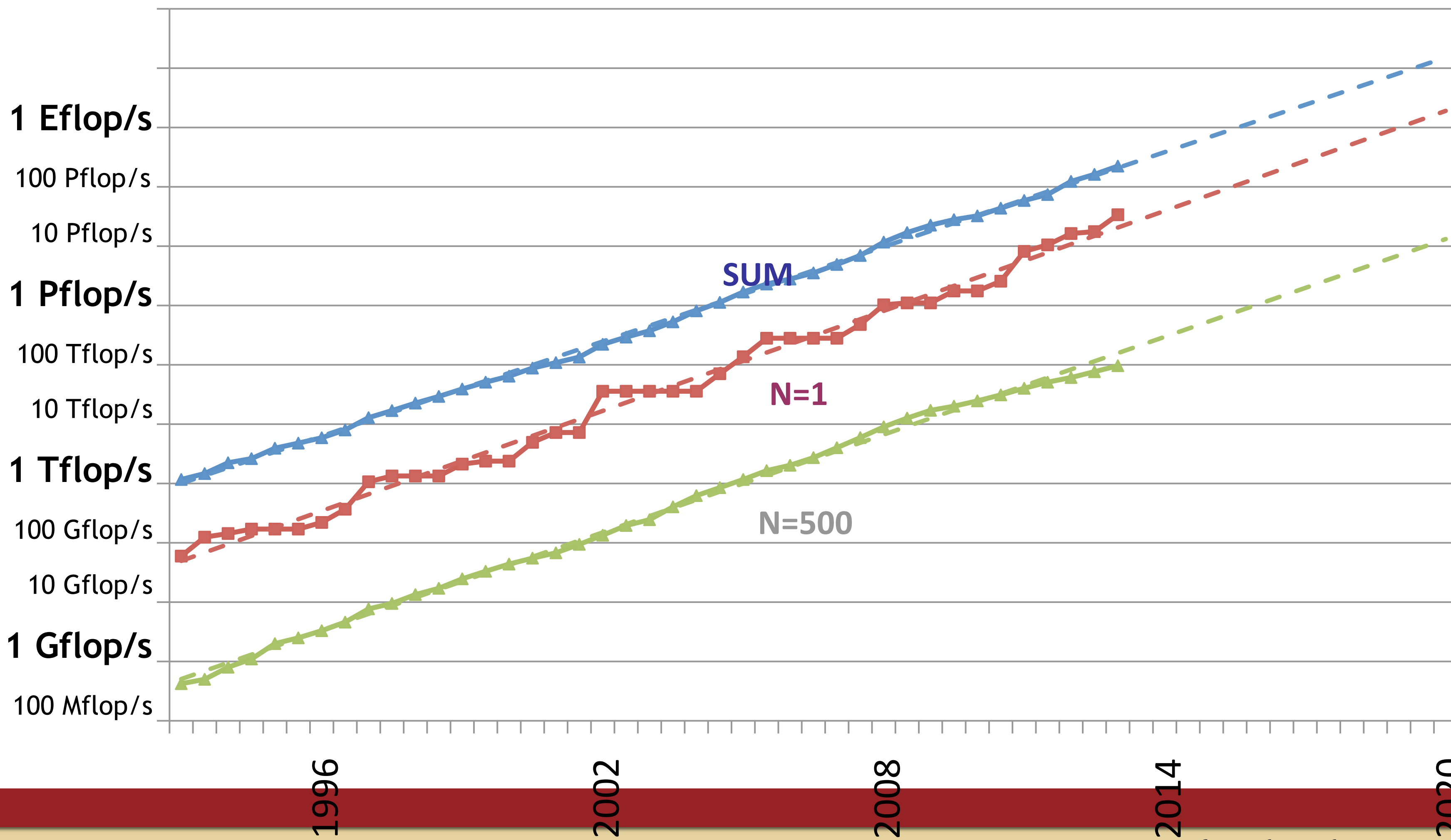
Conventional Heterogeneous Multicore System Architecture



Strengths of Incrementalism to Conventional Architectures

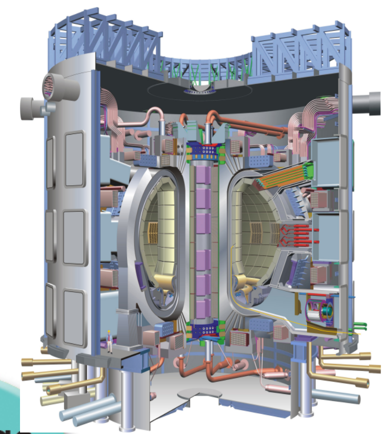
- Known (familiar) methods derived from years of experience
- Demonstrated effectiveness for some workloads
- Exploits COTS for economy of scale through mass market
- Continues usage of MPI programming framework and model
- Consistent with legacy codes
- Assumes incremental changes to codes for enhanced performance
- Rides Moore's Law for device density towards near nano-scale
- Distributed memory avoids complexity and overheads of SM
- Alternative approaches perceived as disruptive and unproven

Exaflops by 2019 (maybe)

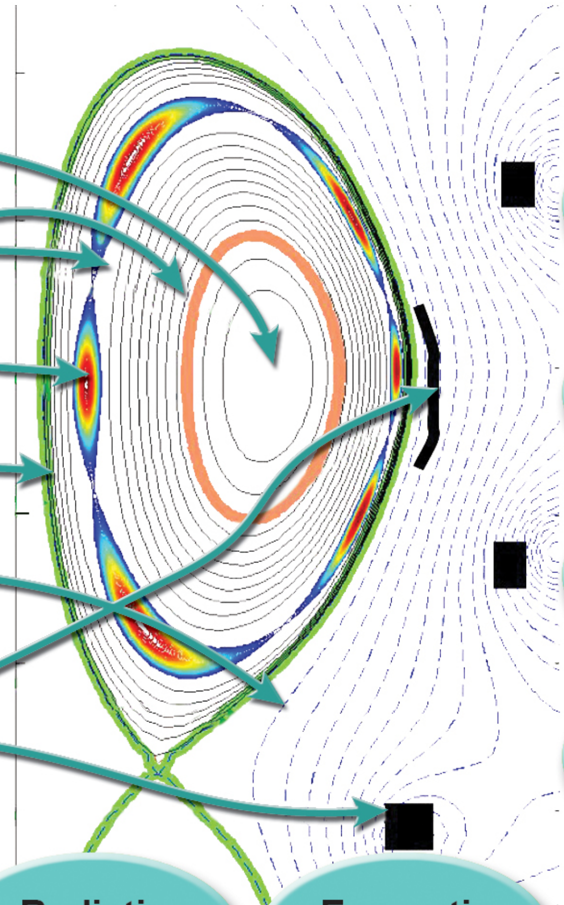


Courtesy of Erich Strohmaier LBNL

Elements of an MFE Integrated Model ➔ *Complex Multi-scale, Multi-physics Processes*



- Sawtooth Region (q < 1)
- Core confinement Region
- Magnetic Islands
- Edge Pedestal Region
- Scrape-off Layer
- Vacuum/Wall/ Conducctors/Antenna



Core & Edge Transport

Plasma Turbulence

Large Scale Instabilities

MHD Equilibrium

Plasma-Wall Interactions

Atomic Physics

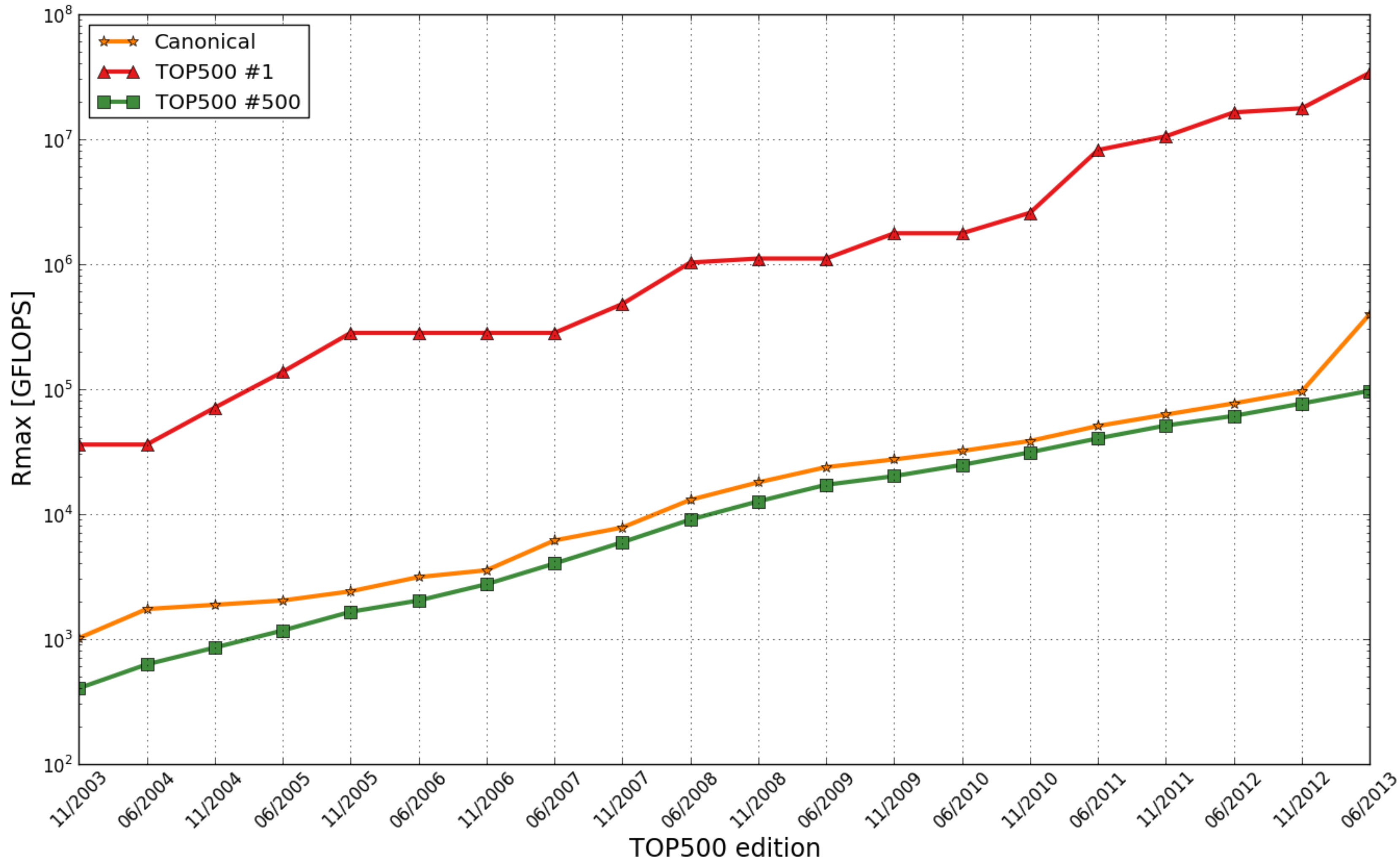
Radiative Transport

Energetic Particles

Heating & Current Drive

Courtesy of Bill Tang, Princeton

Decade of Canonical Systems: Rmax



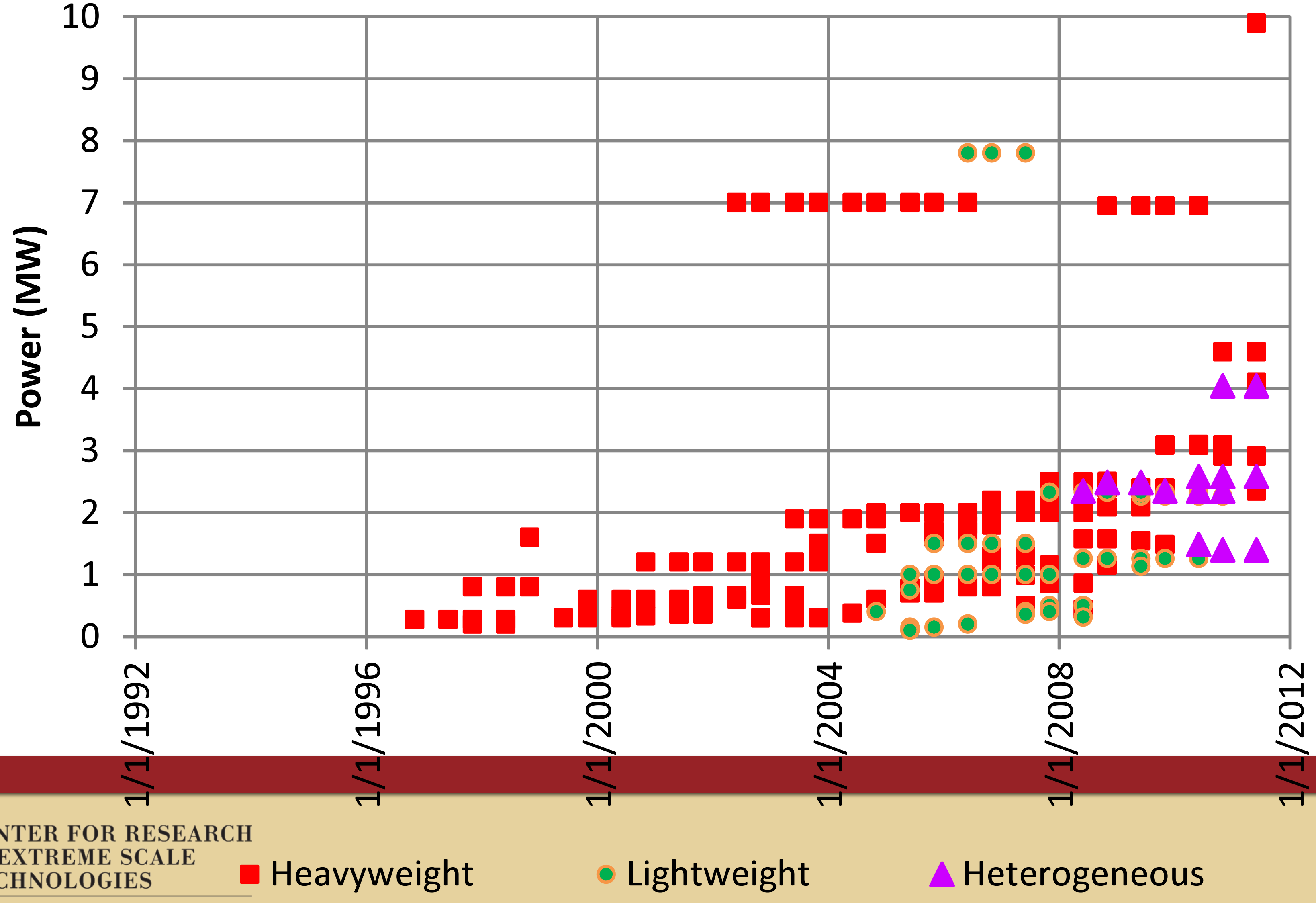
Courtesy of Maciej Brodowicz, Indiana University

Practical Constraints for Exascale

- Sustained Performance
  - Exaflops
  - 100 Petabytes
  - 125 Petabytes/sec.
- Cost
  - Deployment – 250 million $$
  - Operational support
- Power
  - Energy required to run the computer
  - Energy for cooling (remove heat from machine)
  - 20 Megawatts
- Reliability
  - One factor of availability
- Generality
  - How good is it across a range of problems
- Usability
  - How hard is it to program and manage
- Size
  - Floor space – 4,000 sq. meters
  - Access way for power and signal cabling

Courtesy of Peter Kogge, UND

Total Power

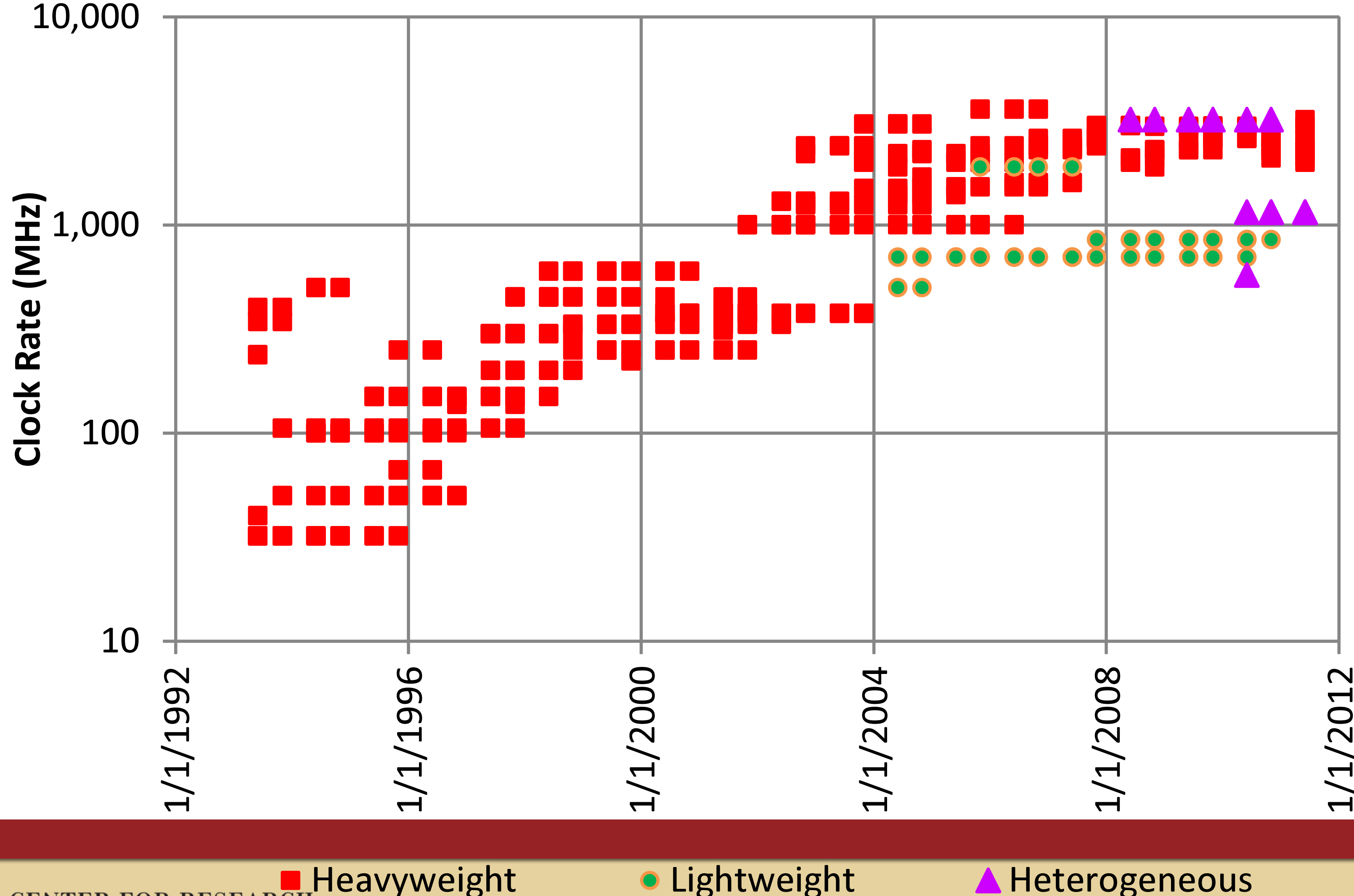


CENTER FOR RESEARCH IN EXTREME SCALE TECHNOLOGIES

INDIANA UNIVERSITY

Pervasive Technology Institute

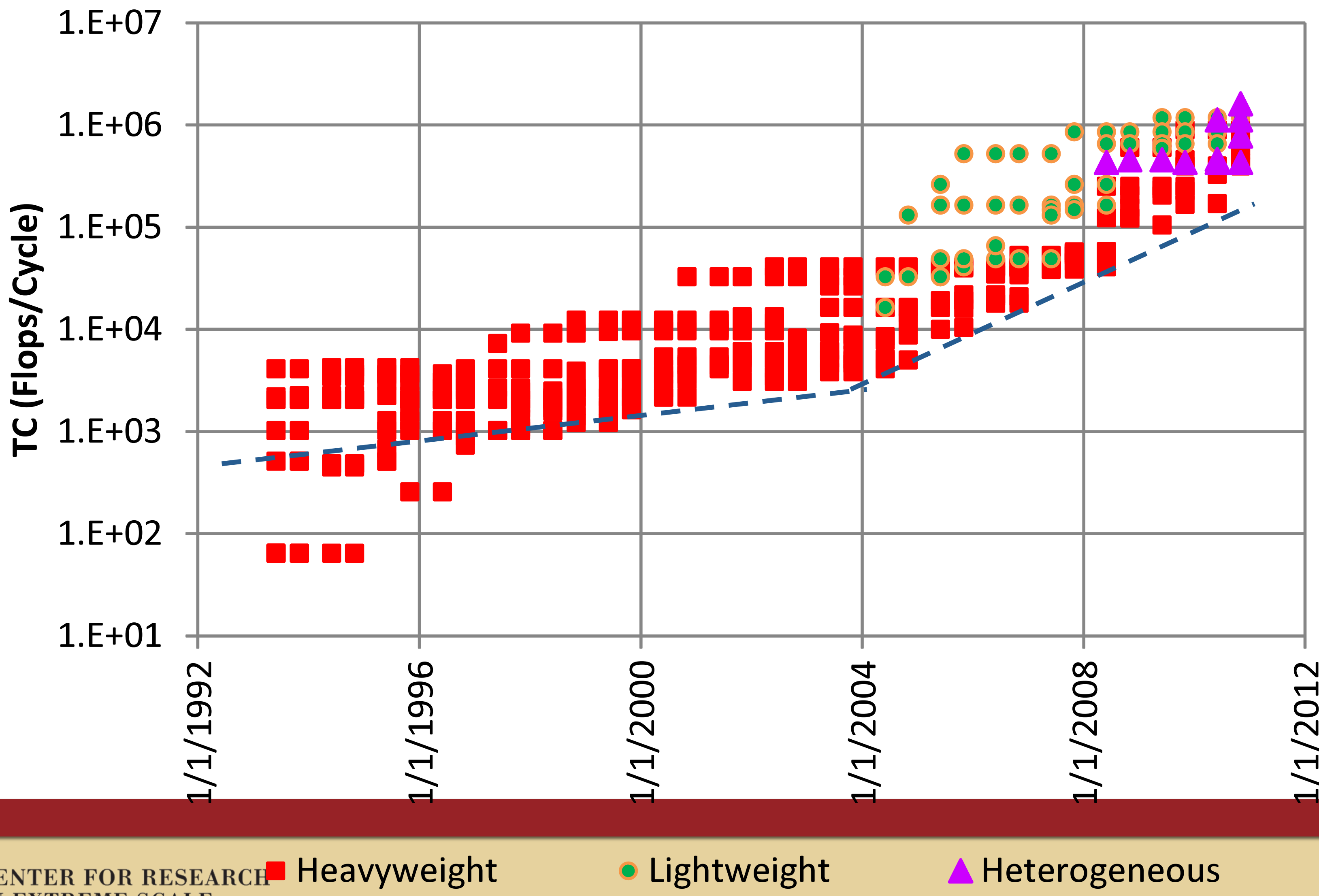
Clock Rate



CENTER FOR RESEARCH IN EXTREME SCALE TECHNOLOGIES
INDIANA UNIVERSITY
Pervasive Technology Institute

Courtesy of Peter Kogge, UND

Total Concurrency



CENTER FOR RESEARCH IN EXTREME SCALE TECHNOLOGIES
INDIANA UNIVERSITY
Pervasive Technology Institute

Courtesy of Peter Kogge, UND

Weaknesses of Conventional Approach

- < 10% efficiencies for many real-world applications
  - Doesn't exploit runtime information
  - Just guesses about the future
- Scaling limited due to inadequacies of exposed parallelism
  - Course grain and ILP, both limited
- Particularly bad for strong-scaled applications
  - Moore's Law is becoming irrelevant
  - HPC becoming special purpose as more apps are falling off the roadmap
- Energy already exceeding threshold of pain
- Check-point/restart times to exceed MTBF
  - Poor reliability
- Programming model an unmitigated disaster

Performance Factors - SLOWER

$$P = e(L,O,W) * s(S) * a(R) * U(E)$$

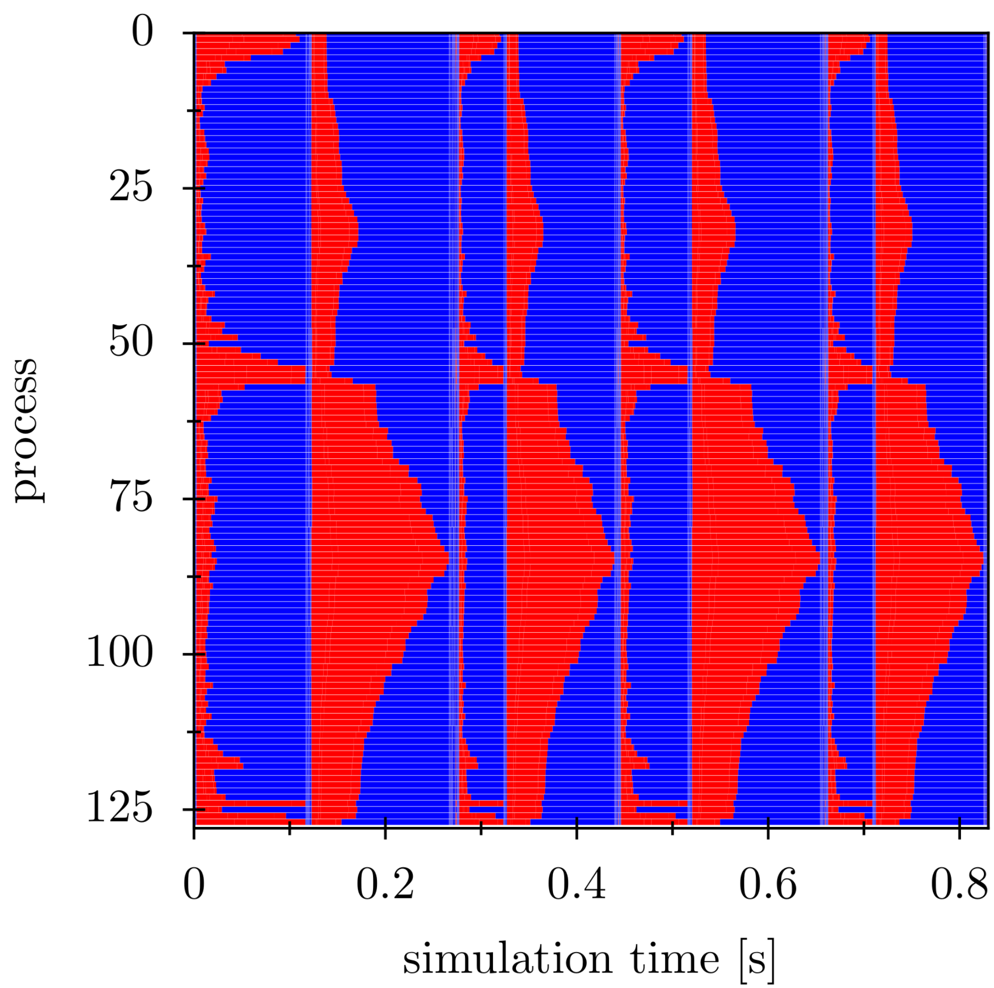
P – average performance (ops)
e – efficiency (0 < e < 1)
s – application's average parallelism,
a – availability (0 < a < 1)
U – normalization factor/compute unit
E – watts per average compute unit
R – reliability (0 < R < 1)

- Starvation
  - Insufficiency of concurrency of work
  - Impacts scalability and latency hiding
  - Effects programmability
- Latency
  - Time measured distance for remote access and services
  - Impacts efficiency
- Overhead
  - Critical time additional work to manage tasks & resources
  - Impacts efficiency and granularity for scalability
- Waiting for contention resolution
  - Delays due to simultaneous access requests to shared physical or logical resources

Conventional Practices

- Communicating Sequential Processes (CSP) model
  - MPI with BSP (Bulk Synchronous Parallel)
  - Closely matches underlying cluster and MPP system architectures
  - Static allocation
- Starvation
  - 1 coarse-grain process per processor core
  - ILP compiler and architecture driven
  - Assumes regular distributed work allocation
- Overhead
  - Avoidance – static scheduling
- Latency
  - Avoidance – exchange data after a lot of local work
- Contention
  - Minimize data transfers with respect to amount of work
  - Actually increases contention by forcing all communications into single phase

The Negative Impact of Global Barriers in Astrophysics Codes

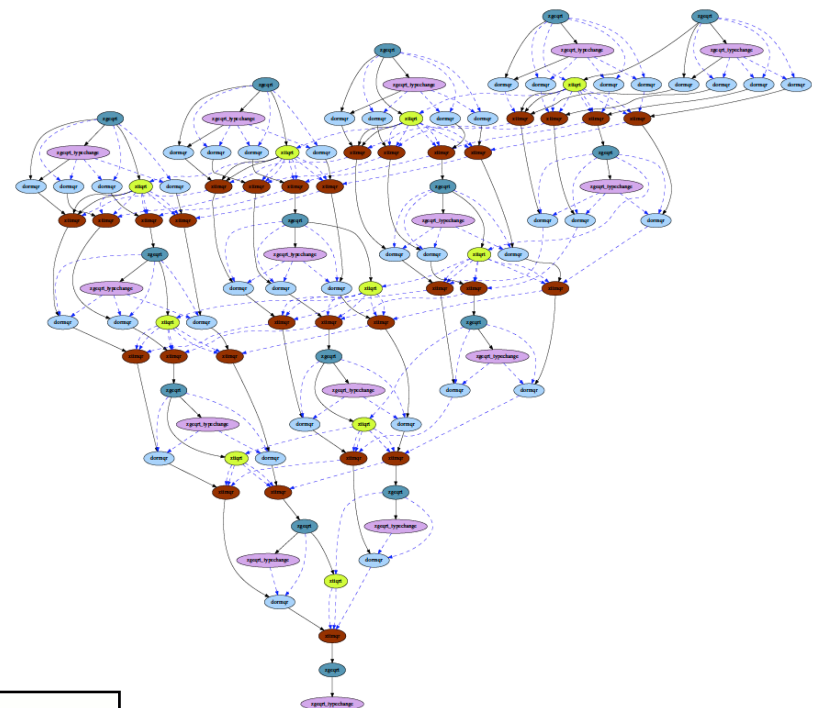


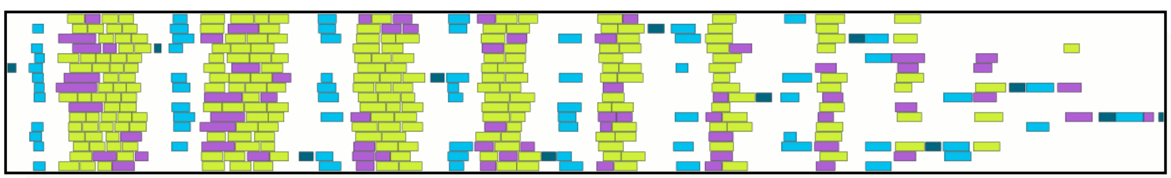
Computational phase diagram from the MPI based GADGET code (used for N-body and SPH simulations) using 1M particles over four time steps on 128 procs.

Red indicates computation
Blue indicates waiting for communication

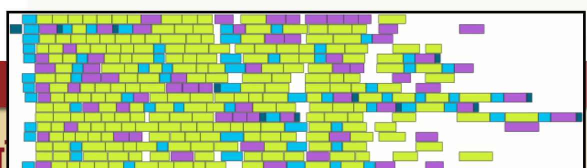
The Purpose of a QUARK Runtime

- Objectives
  - High utilization of each core
  - Scaling to large number of cores
  - Synchronization reducing algorithms
- Methodology
  - Dynamic DAG scheduling (QUARK)
  - Explicit parallelism
  - Implicit communication
  - Fine granularity / block data layout
- Arbitrary DAG with dynamic scheduling





Fork-join parallelism
Notice the synchronization penalty in the presence of heterogeneity.

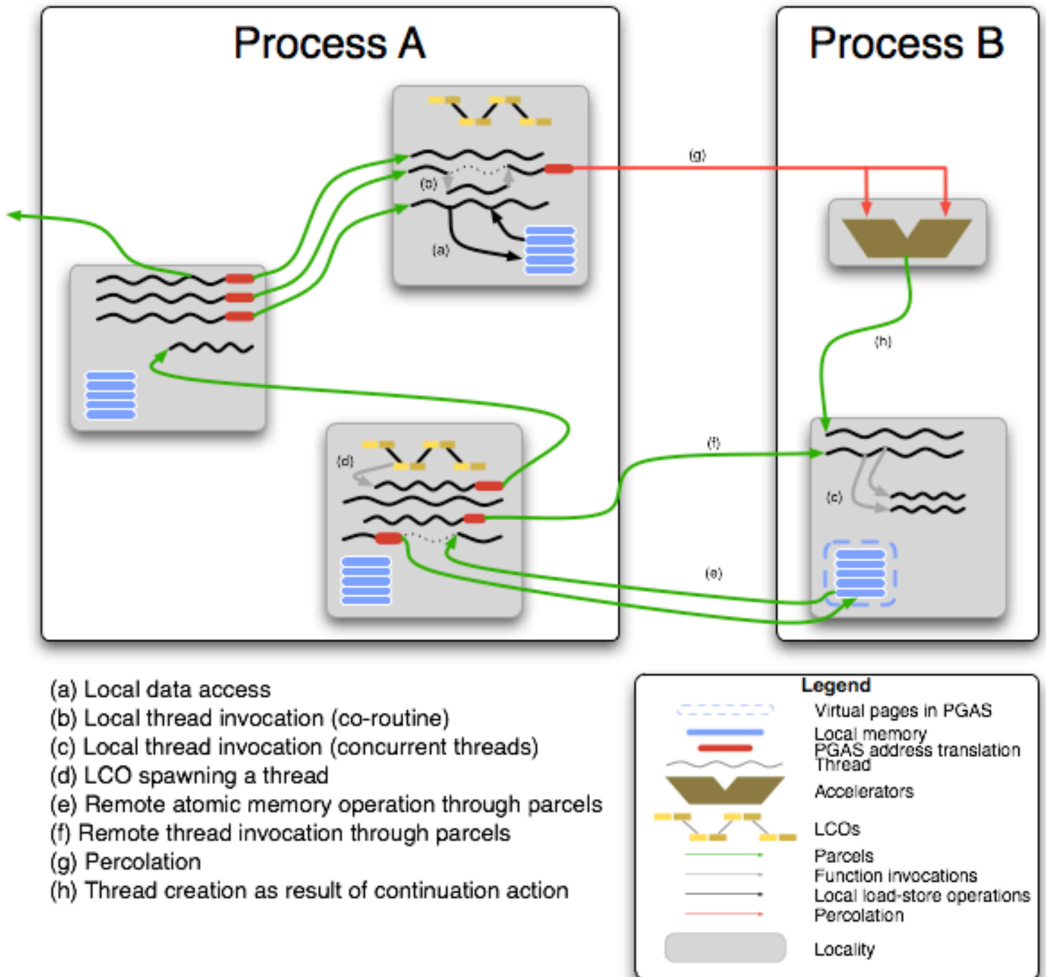


DAG scheduled parallelism

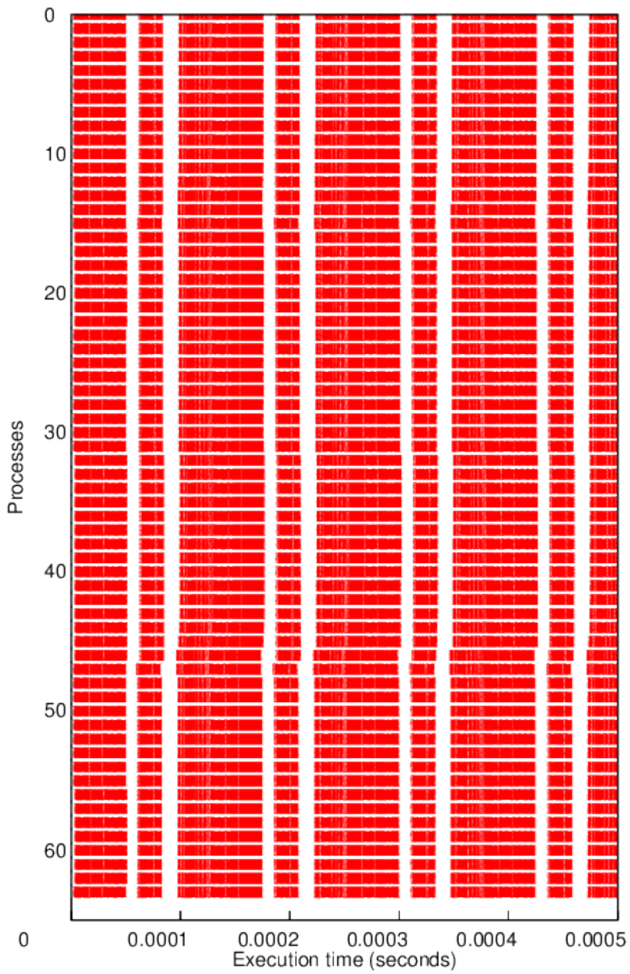
Courtesy of Jack Dongarra, UTK

ParalleX Execution Model

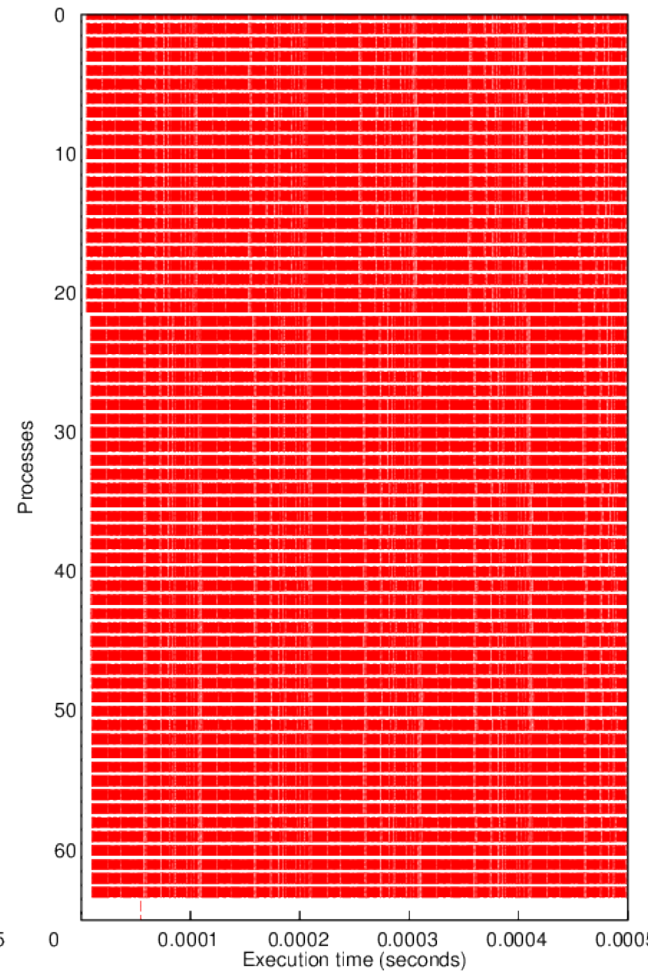
- Lightweight multi-threading
  - Divides work into smaller tasks
  - Increases concurrency
- Message-driven computation
  - Move work to data
  - Keeps work local, stops blocking
- Constraint-based synchronization
  - Declarative criteria for work
  - Event driven
  - Eliminates global barriers
- Data-directed execution
  - Merger of flow control and data structure
- Shared name space
  - Global address space
  - Simplifies random gathers



Overlapping computational phases for hydrodynamics



MPI



HPX

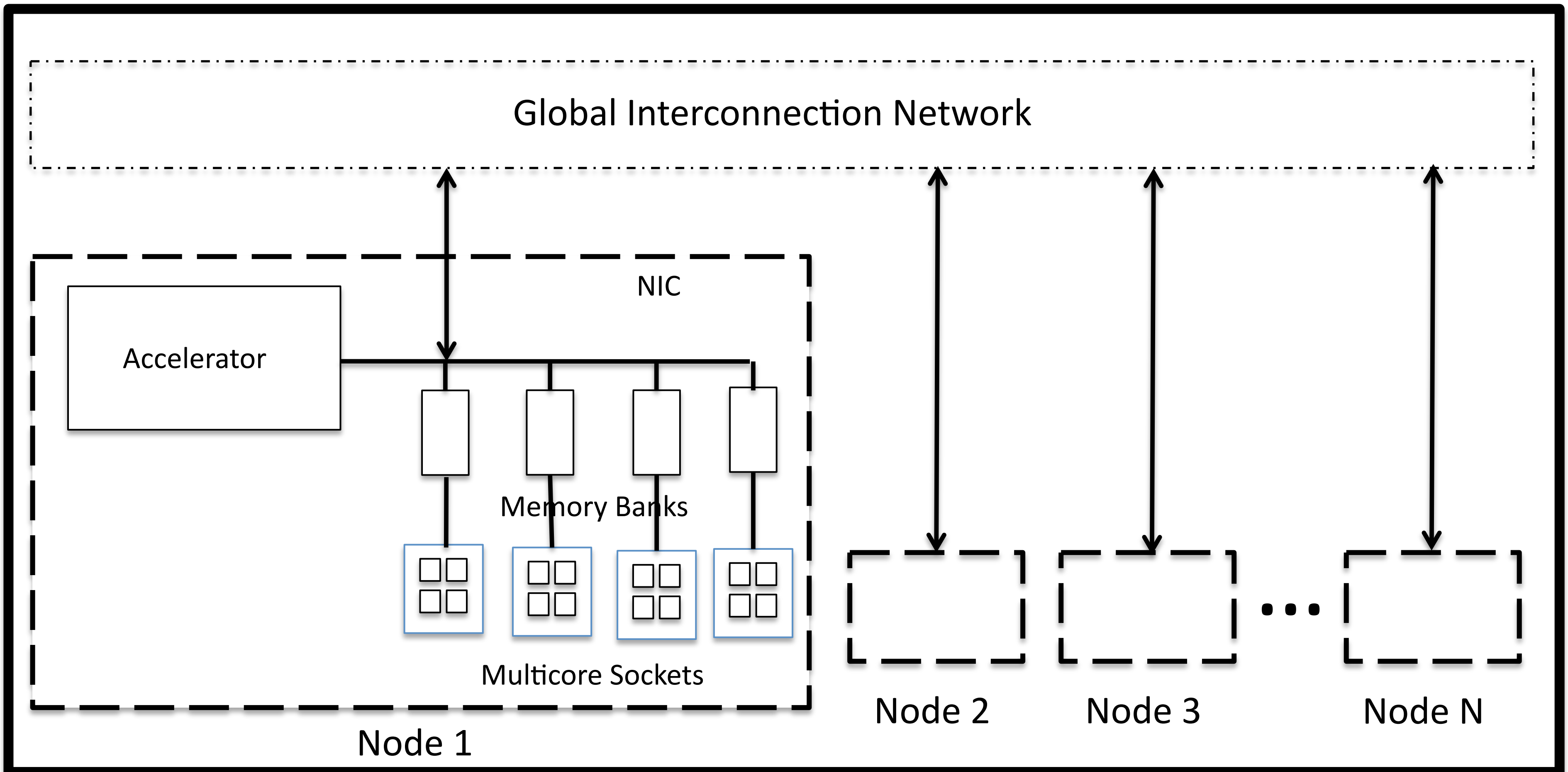
Computational phases for LULESH (mini-app for hydrodynamics codes).

Red indicates work

White indicates waiting for communication

Overdecomposition: MPI used 64 process while HPX used 1E3 threads spread across 64 cores.

Advanced GAS Exascale System Architecture



Exascale System Advances

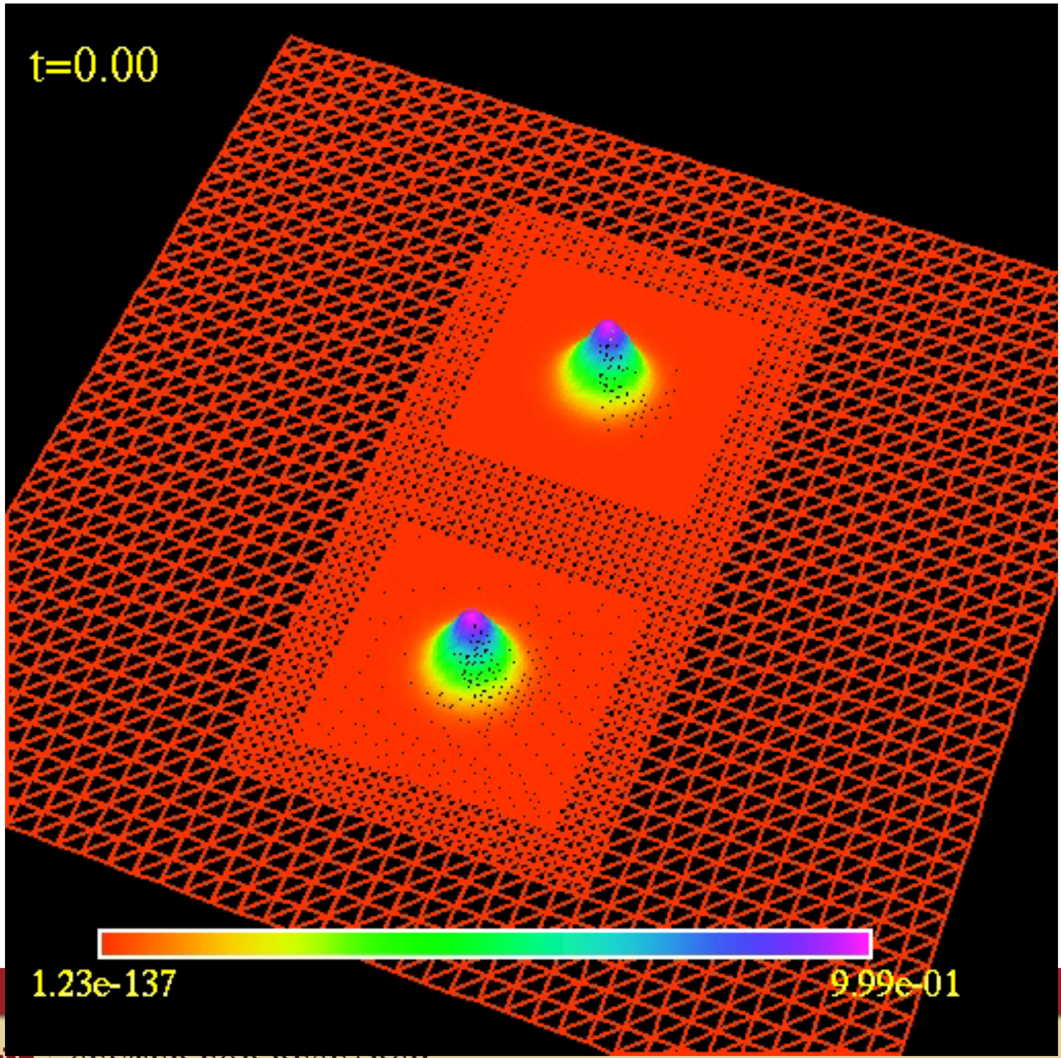
Conventional Incremental

- Change
- Static – scheduling & management
- Message passing
- Distributed memory
- Local view; rest is I/O
- Von Neumann bottleneck
- Bulk Synchronous Parallel (BSP)
- Speculative
- Synchronous
- Global barriers
- Separation of cores vs. NICs
- Explicit fixed processes
- Hybrid programming - monstrosity

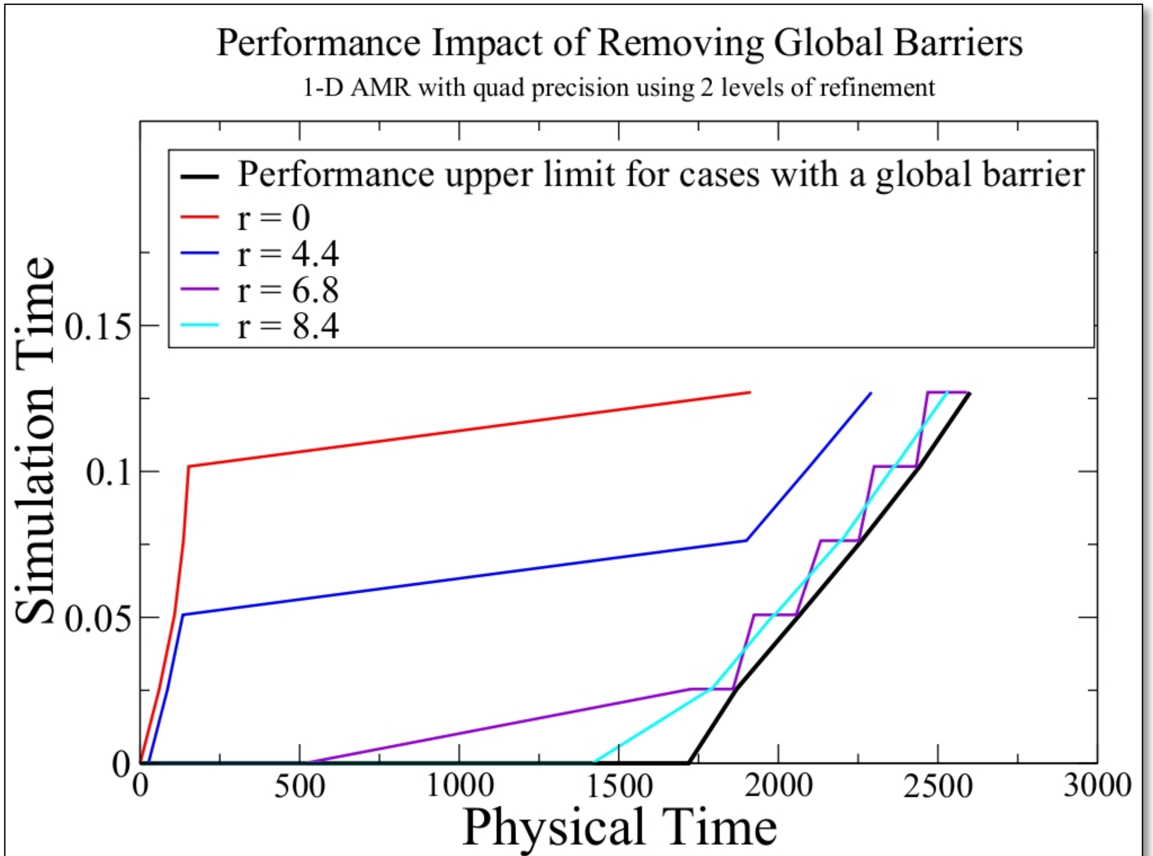
Advanced Revolutionary

- Change
- Dynamic – adaptive runtime control
- Message-driven
- Global name space
- System-wide object access
- Embedded memory processing
- Dataflow – overlapped phases
- Multi-threaded
- Asynchronous
- Futures – continuations migration
- Merged ISA for compute/comm.
- Meta-threads, depleted threads
- Unified programming model

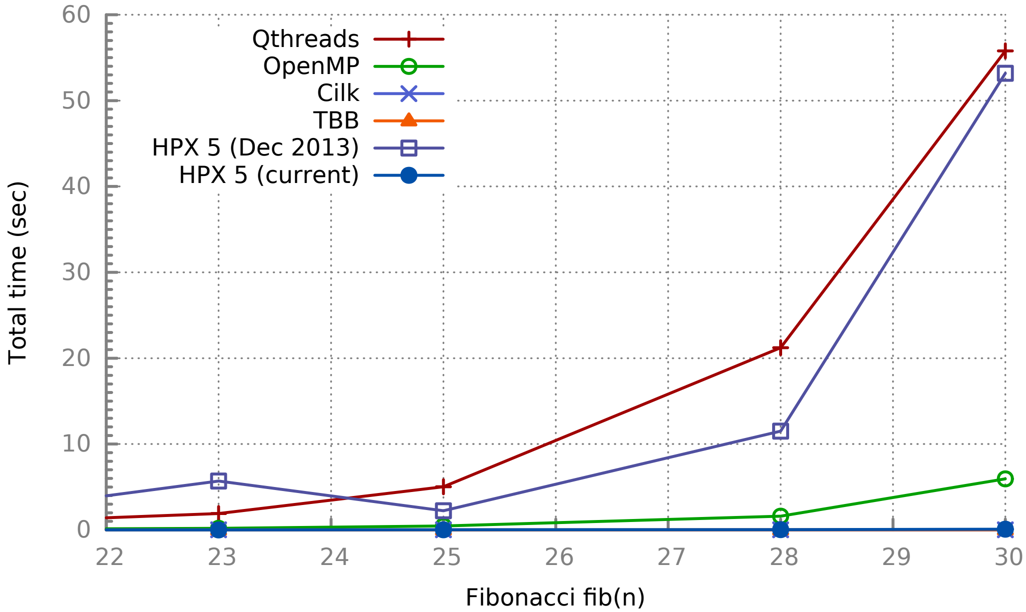
Dynamic load balancing via message-driven work-queue execution for Adaptive Mesh Refinement (AMR)



Application: Adaptive Mesh Refinement (AMR) for Astrophysics simulations

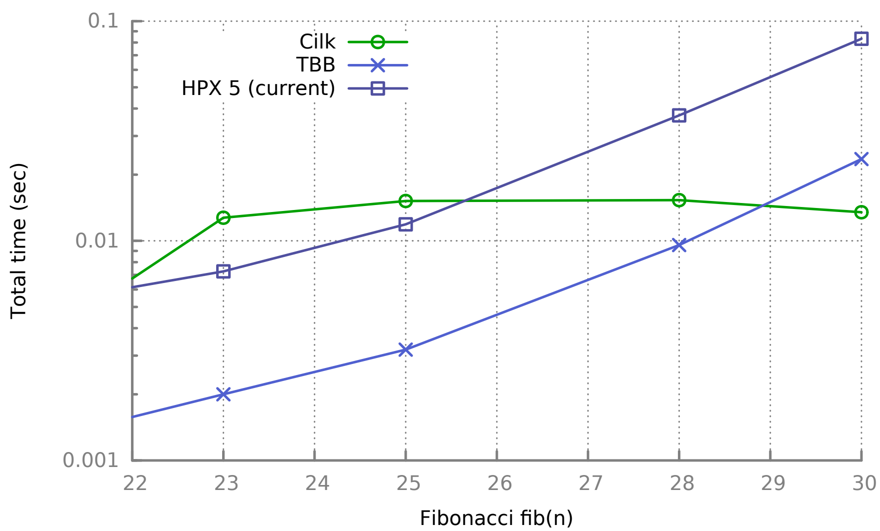


HPX-5 Development Progress



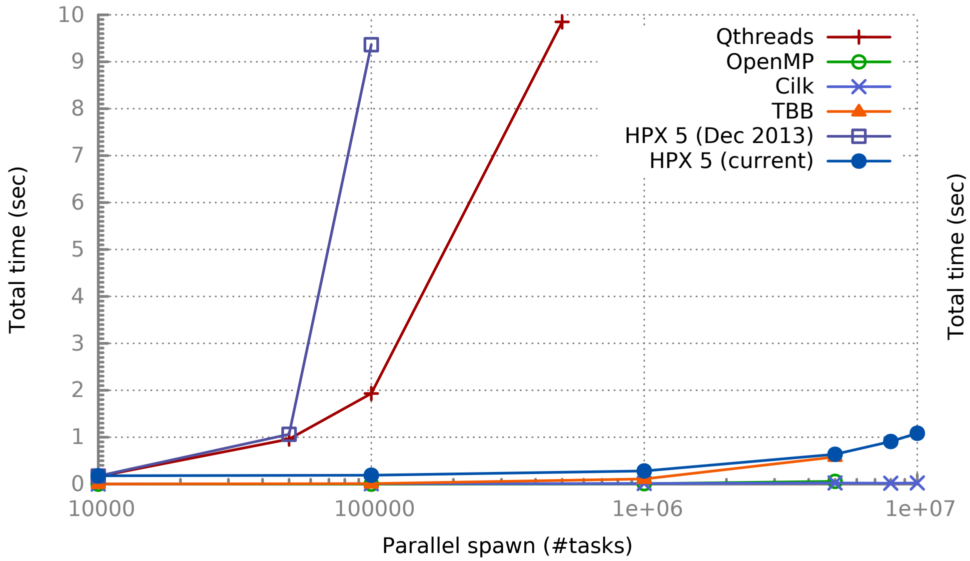
All cases run on 16 cores (1 locality)

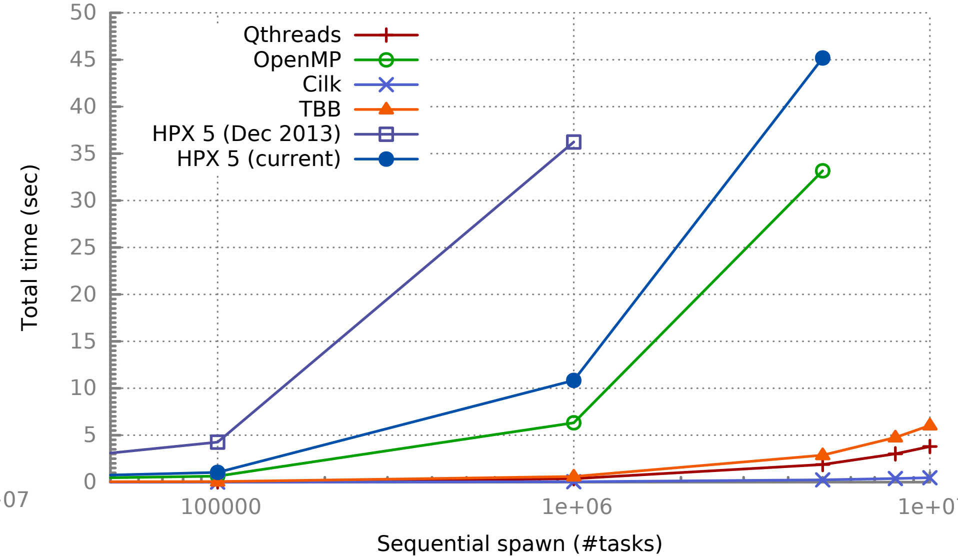
Zoom-in on best performers



Courtesy of Matt Anderson, Indiana University

HPX-5 Development Progress

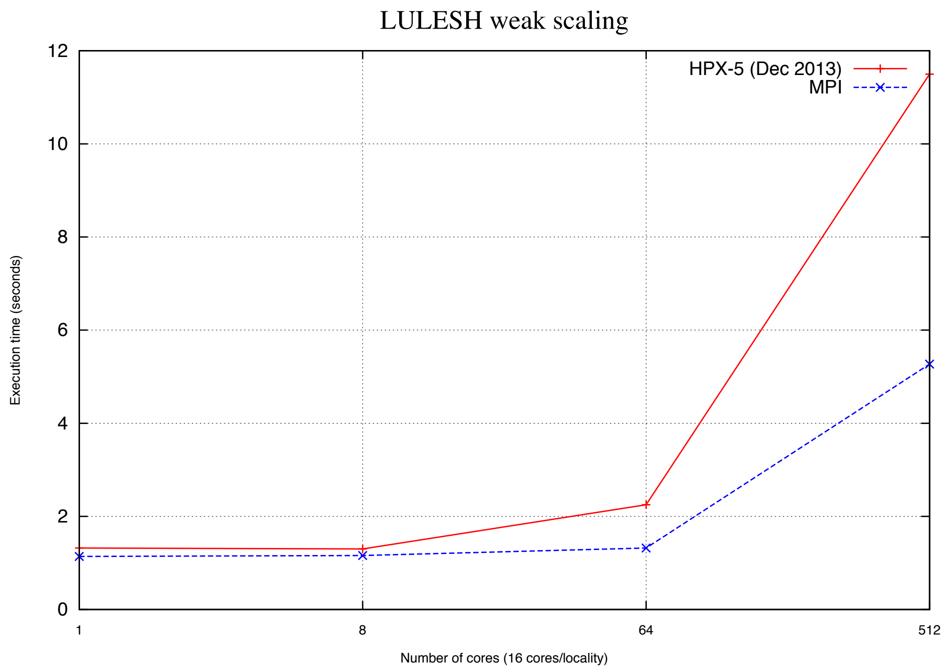


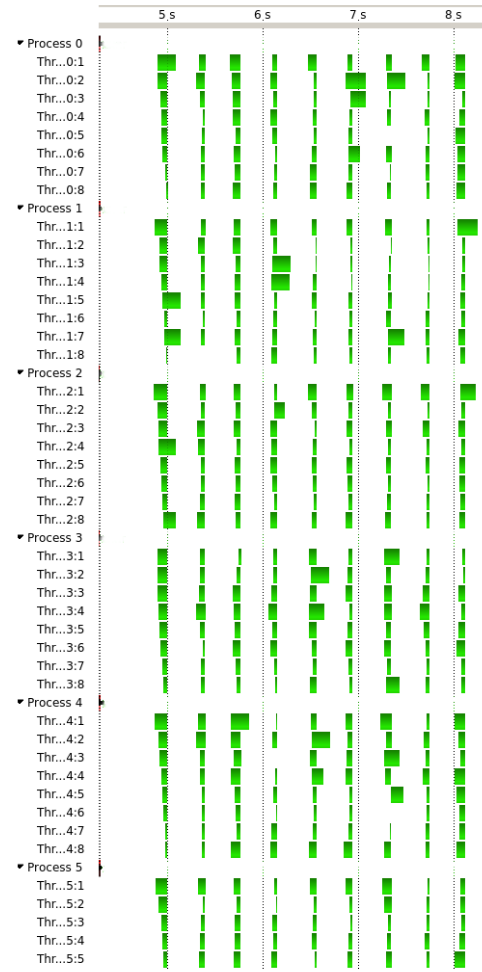


CENTER FOR RESEARCH IN EXTREME SCALE TECHNOLOGIES
INDIANA UNIVERSITY
Pervasive Technology Institute

Courtesy of Matt Anderson, Indiana University

LULESH in HPX-5 (current status)



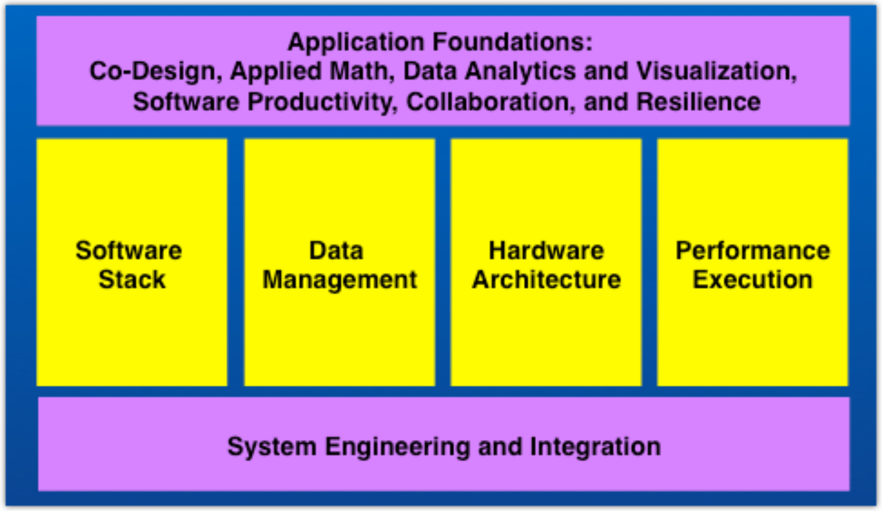


Replacing MPI calls with HPX-5 calls but not changing the LULESH algorithm results in worse weak scaling than just using MPI currently. The time spent waiting for communication is shown in phases at the right in green.

Courtesy of Matt Anderson, Indiana University

Research Breakdown toward Exascale

- Software Stack
- Performance Execution
- Data Management
- Hardware Architecture
- System Engineering and Integration
- Co-design
- Applied Math
- Resiliency
- Data visualization
- Productivity
- Collaboration



Conclusions

- HPC is in a (6th) phase change
- Ultra high scale computing of the next decade will require a new model of computation to effectively exploit new technologies and guide system co-design
- ParalleX is an example of an experimental execution model that addresses key challenges to Exascale
- Early experiments prove encouraging for enhancing scaling of graph-based numeric intensive and knowledge management applications