Boise State University

Foundations and Future Directions for Exascale Computing

Thomas Sterling

Chief Scientist, CREST
Professor, School of Informatics and Computing
Indiana University

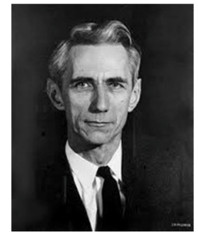
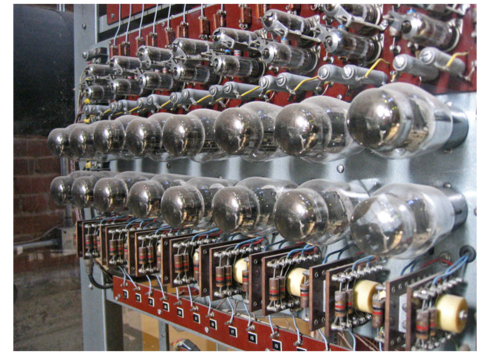
April 9, 2015

Introduction

- Exascale computing will demand innovations greater than required for Petaflops, 7 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
  - Dramatic increase in efficiency and scalability with productivity
  - Address starvation, latency, overhead, contention, energy, & reliability
  - Dynamic adaptive resource management and task scheduling
- Multicore design innovations to support $>10^9$ threads
  - Architecture mechanisms for cooperative computing with low overheads
  - Integrated core and memory for low latency and high bandwidth

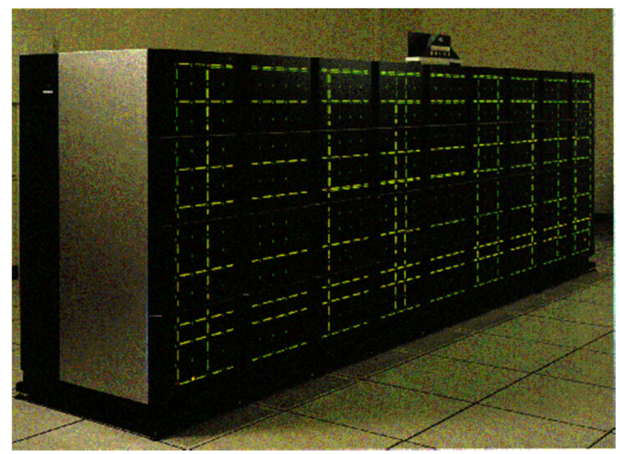
The von Neumann Age



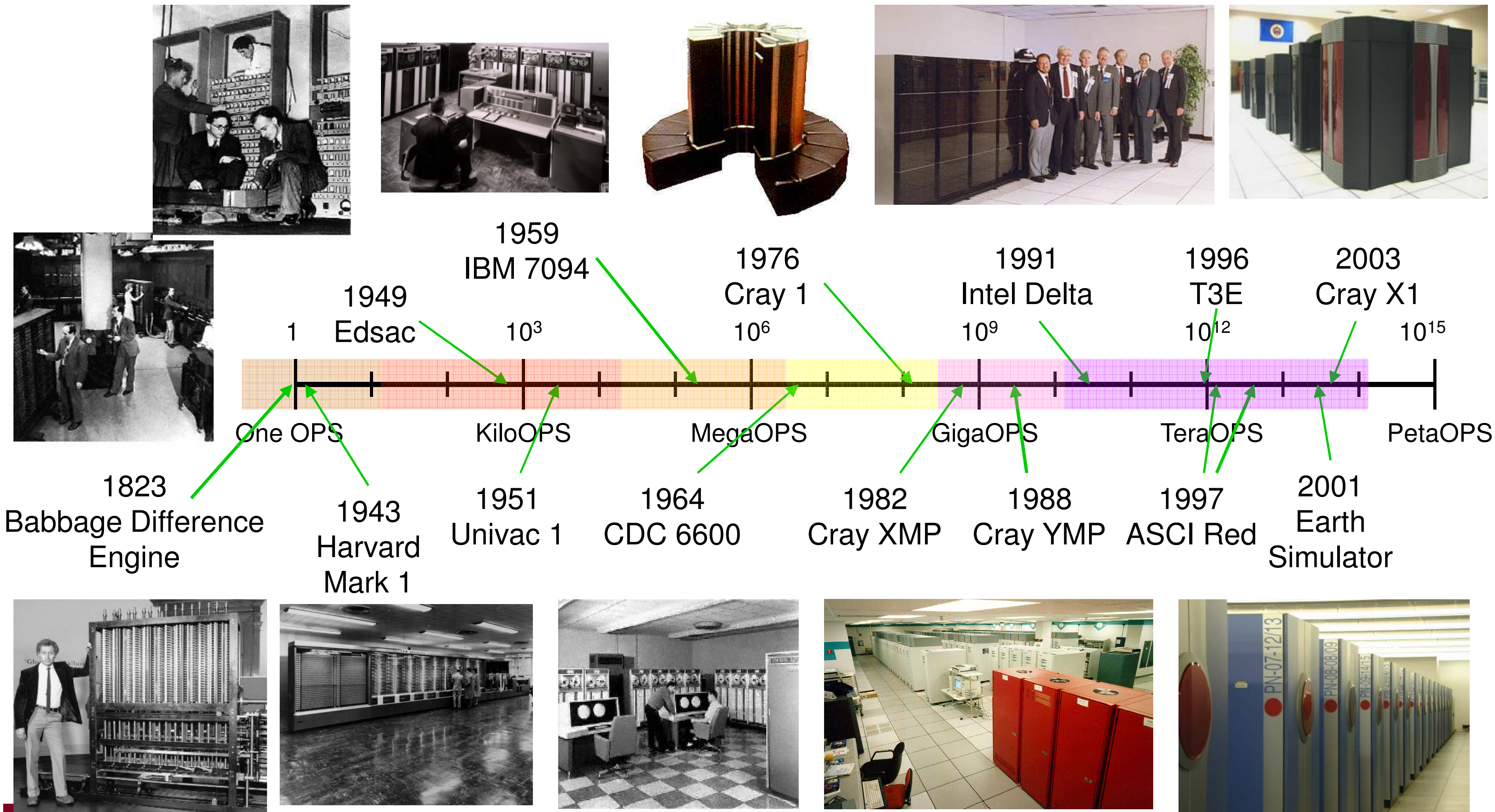
- Foundations:
  - Information Theory – Claude Shannon
  - Computabilty – Turing/Church
  - Cybernetics – Norbert Wiener
  - Stored Program Computer Architecture – von Neumann

- The von Neumann Shift: 1945 – 1960
  - Vacuum tubes, core memory
  - Technology assumptions
    - ALUs are most expensive components
    - Memory capacity and clock rate are scale drivers – mainframes
    - Data movement of secondary importance



- Von Neumann derivatives: 1960 – 2015
  - Semiconductor, Exploitation of parallelism
  - Out of order completion
  - Vector
  - SIMD
  - Multiprocessor (MIMD)
    - SMP
      - Maintain sequential consistency
    - MPP/Clusters
      - Ensemble computations with message passing

Technology/Architecture/Programming Synergy

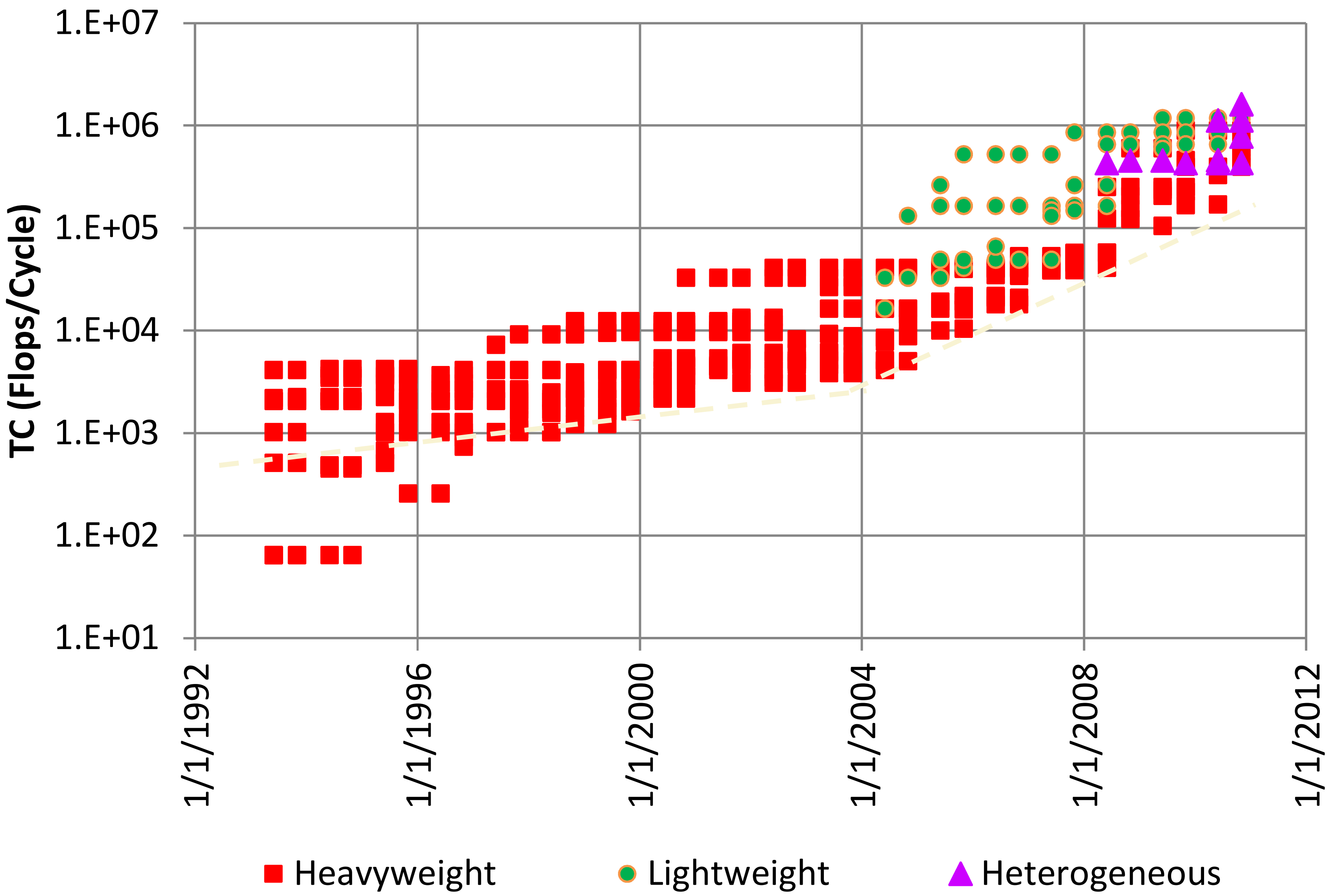


Conventional HPC

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

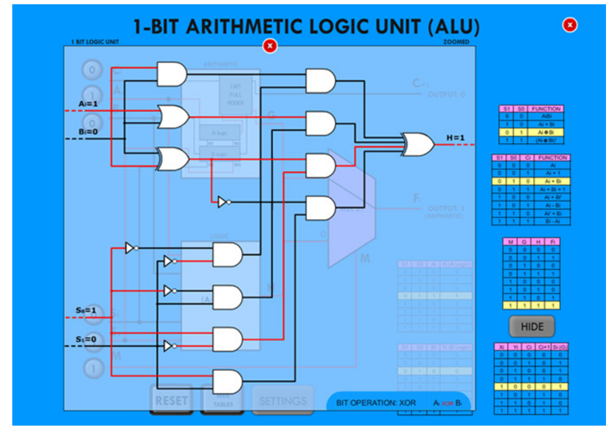
Total Concurrency



Courtesy of Peter Kogge,

Unquestioned Assumptions

- Floating point ALU optimized resource
- All architectures are von Neumann derivatives
- Control is sequential instruction issue, IP
- Node performance optimized
- Binary values
- 2-state Boolean logic
- Fixed-length instruction set architecture (ISA)
- Separation of CPU and main memory (von Neumann bottleneck)
- Silicon based semiconductor technology
- X86 dominated instruction set
- Checkpoint/restart for fault tolerance
- CSP/MPI (well, not unquestioned)



Head room, margins, potential innovations
Floating point ALU optimized resource

- Costs and burdens:
    - Cache hierarchy
    - Branch prediction
    - Speculative execution
    - Out of order flow control reservation stations, …
    - Prefetching, many simultaneous in-flight requests
- Alternatives:
    - Emphasis on memory access throughput
    - Response time to incidence of external messages
    - Scratch pad memory
    - Multi-threading
    - Dataflow ISA
    - Asynchronous flow control

Head room, margins, potential innovations
All architectures are von Neumann derivatives
Control is sequential instruction issue, IP

- Costs and burdens
  - Variants: out of order, vector, SIMD, MPPs and clusters
  - Flow control bottlenecks
  - Control state limited to program counters, fork-joins
  - Loss of operational precedence
  - Not effective in asynchronous operation
- Alternatives
  - DAGs
  - Dataflow
  - Systolic arrays
  - unums

Head room, margins, potential innovations

- Boolean logic, Binary values, bits
  - Limited to 2-state per spatial storage units
  - Boolean logic does not have to be limited to 2-state
  - Higher base may save space and energy
  - Single flux quantum storage: many incremental flux levels
- Fixed length instructions
  - Variable length instructions
  - Compression through Hamming codes
  - Tagged registers for typing – generic
  - Biased register access patters – accumulator
  - Average of 4 – 5 bits per instruction with full semantic richness
  - Convert instruction register to inner-loop register
  - Saves bits, bandwidth, energy
- X86 ISA
  - Treats rest of system as I/O devices
  - Doesn't recognize asynchrony of operation or message-driven computing

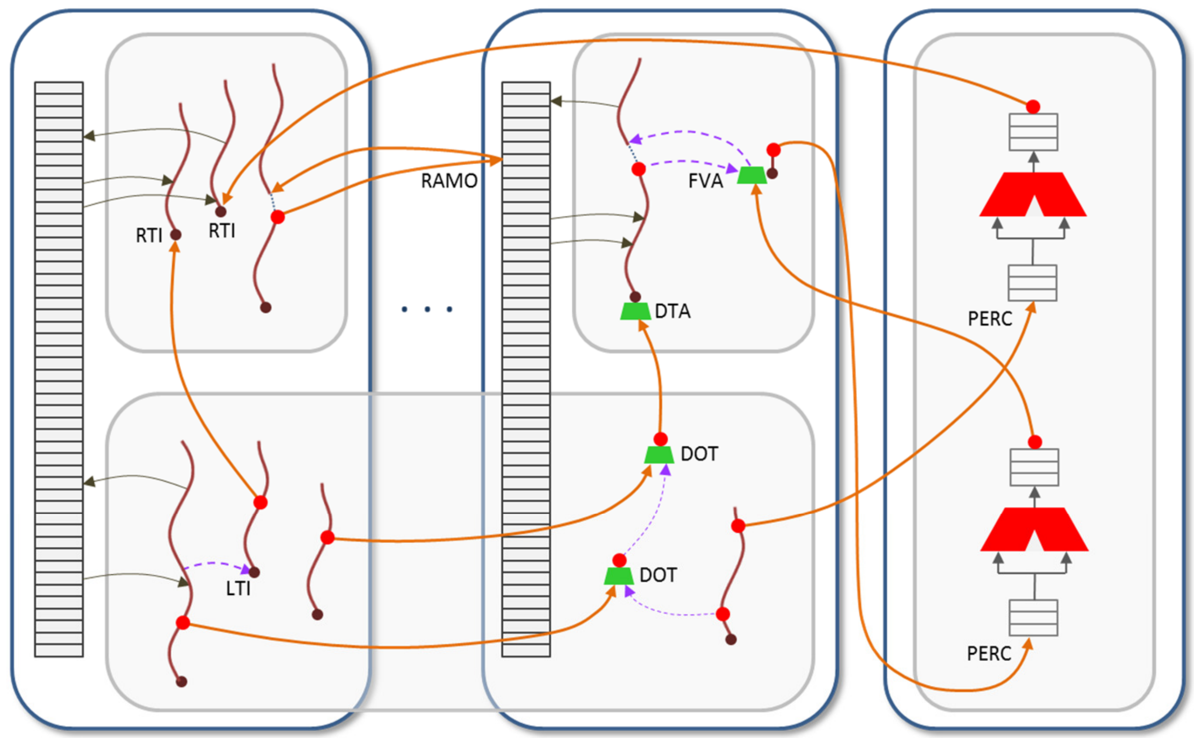
Head room, margins, potential innovations

- Separation of CPU and main memory
  - Major bottleneck
  - Worse with multi/many core processor sockets
  - A driver for need for cache
  - Processor in Memory (PIM)
  - On-chip scratch pad memory
- Silicon based semiconductor technology
  - Moore's Law will flat-line by end of decade, ~ 5 nm feature size
  - Leakage current a challenge
  - Graphene of interest
  - Superconducting single flux quantum logic at 100 – 200 GHz, 100X energy advantage
- CSP/MPI (well, not unquestioned)
  - MPI + X, where X = OpenMP maybe
  - Fork-joins impose Amdahl bottlenecks
  - X could also be DAGs
  - Asynchronous Multi-Task execution models

Motivation for ParalleX Execution Model

- A crosscutting **execution model** to determine respective roles, responsibilities, and interoperability
- Exploit runtime information through introspection to discover parallelism for **scalability** and dynamically manage resources to demand for **efficiency**
- Expose limitations of conventional computer **architecture** and devise mechanisms for lower overhead and latency
- Serve as a **research** platform (HPX) to explore utility, generality, opportunity, and challenges/limitations
- Target and enabler for parallel **programming models**
- Operation in the presence of uncertainty of **asynchrony**
- First conceived in support of HTMT project and Cascade

Distinguishing Features of ParalleX/HPX



- Locality
- Process
- Local Memory
- LCO
- Accelerator

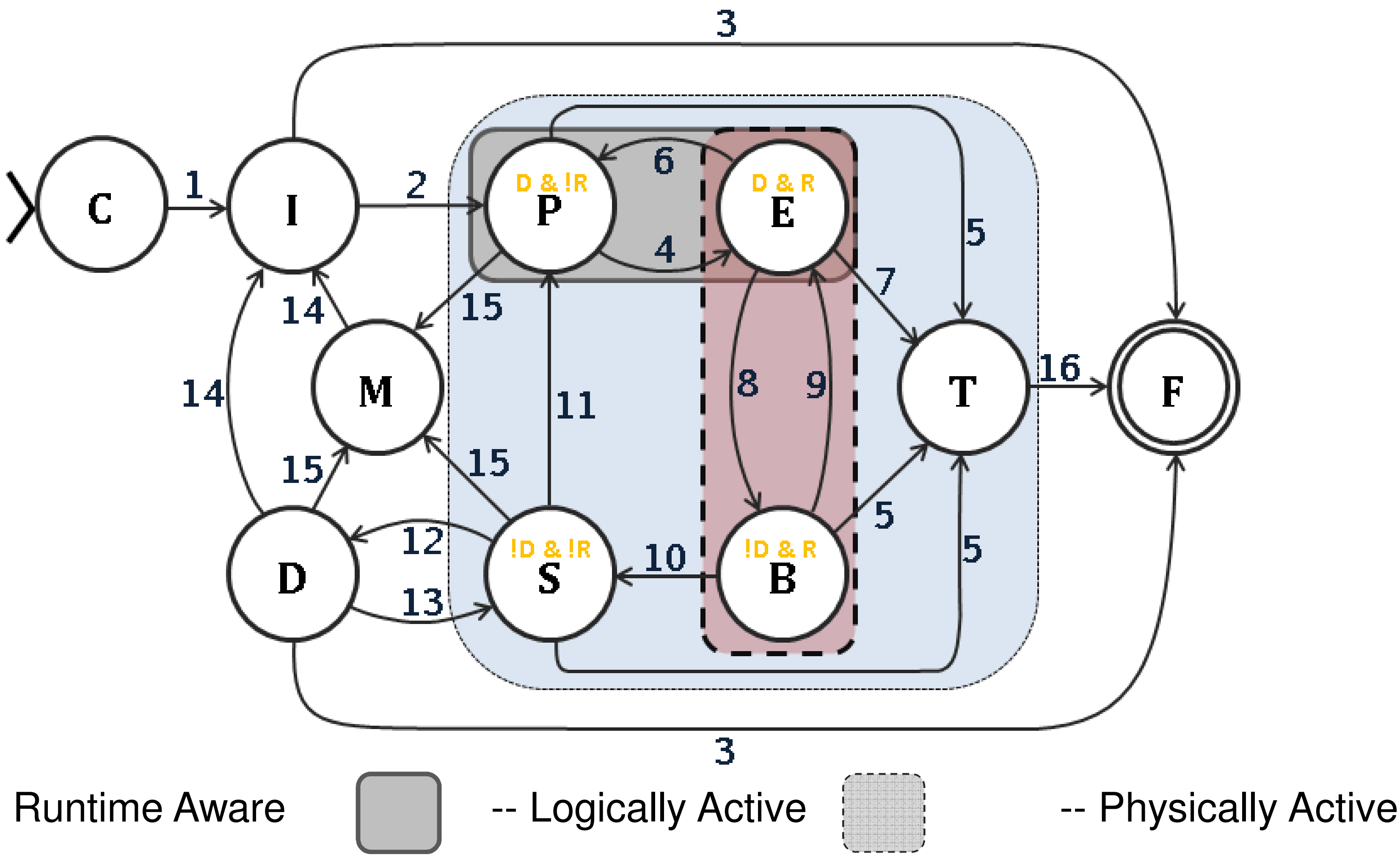
- Thread
- Suspended Thread
- Local Memory Access
- AGAS Address Lookup
- Local Action
- Parcel

LTI: local thread instantiation
RTI: remote thread instantiation
RAMO: remote atomic memory operation
DTA: depleted thread activation
DOT: dataflow object trigger
FVA: future value access
PERC: percolation

ParalleX Compute Complexes

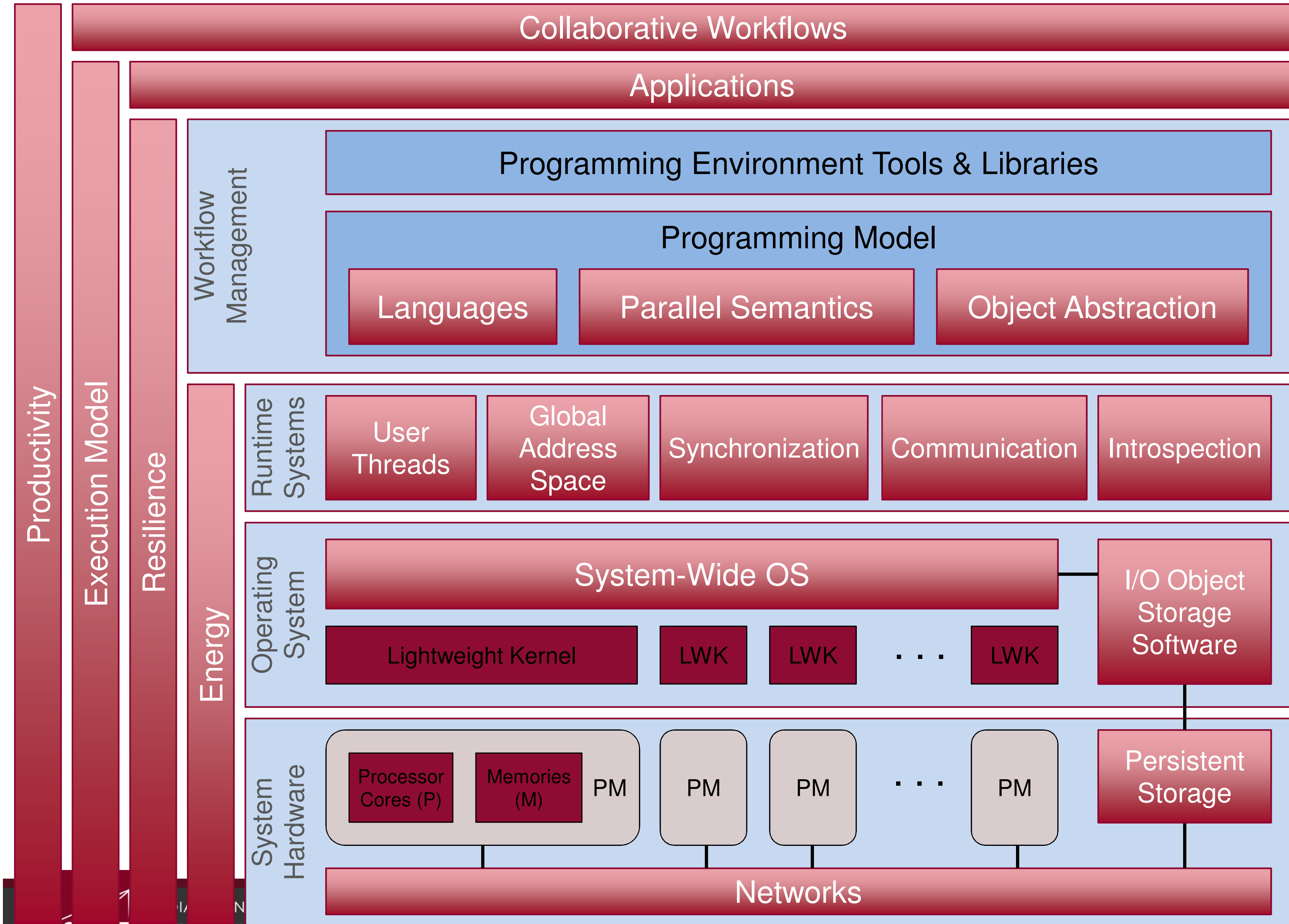
- Manifest as a variant of threads on conventional platforms
- Complexes are first-class objects
- Unbounded number of complex registers
- Preemptive, sometimes
- Internal static dataflow ILP
- Depleted complexes exhibit LCO synchronization semantics
- Can migrate as continuations
- State-machine definition in and out of runtime system

ParalleX Computation Complex



Extensions

- Power management
  - "Side-Path Energy Suppression"
  - Not funded
- Reliability
  - CVC-Microcheckpointing
  - Not funded
- Real-time
  - Semantics of time, progress to goal, priorities
  - NSF sponsored
- PXFS
  - Data driven mass storage
  - Unified name space
  - NSF sponsored
- PRIDE
  - System-wide operating system
  - Scales ParalleX processes up

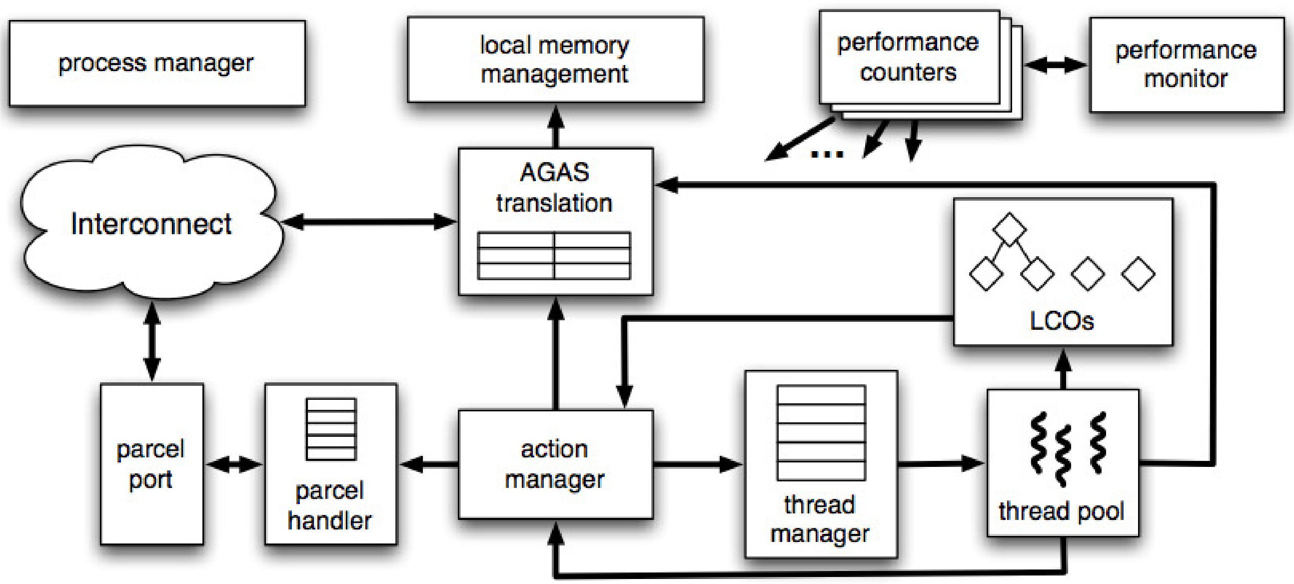

Collaborative Workflows
Applications
Productivity
Execution Model
Resilience
Energy
Workflow Management
Programming Environment Tools & Libraries
Programming Model
Languages
Parallel Semantics
Object Abstraction
Runtime Systems
User Threads
Global Address Space
Synchronization
Communication
Introspection
Operating System
System-Wide OS
I/O Object Storage Software
Lightweight Kernel
LWK
LWK
. . .
LWK
System Hardware
Processor Cores (P)
Memories (M)
PM
PM
PM
. . .
PM
Persistent Storage
Networks

Asynchronous Many-Task Runtime Systems

- Dharma – Sandia National Laboratories
- Legion – Stanford University
- Charm++ – University of Illinois
- Uintah – University of Utah
- STAPL – Texas A&M University
- HPX – Indiana University (also LSU)
- OCR – Rice University

HPX Runtime Design

- Current version of HPX provides the runtime infrastructure as defined by the ParalleX execution model
  - Compute Complexes (ParalleX Threads) and scheduling
  - Parcel Transport and Parcel Management for message-driven computation
  - Local Control Objects (LCOs) for synchronization
  - Active Global Address Space (AGAS) for system wide naming



HPX: Distinguishing Features (1)

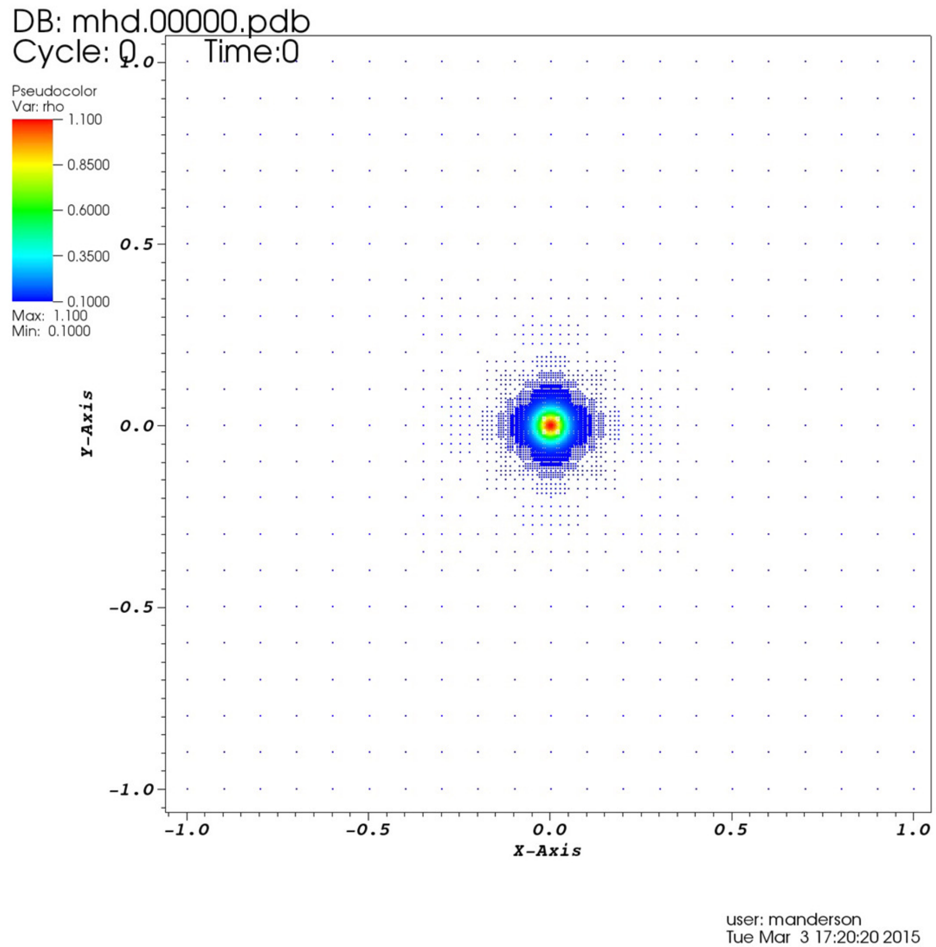
- Derived within the conceptual context of an execution model
- Derived within the context of the SLOWER performance model
- Global Name Space and active global address space
- ParalleX Processes
  - Span and share multiple hardware nodes
  - hierarchical (nested)
  - First-class objects
  - Support user and node OS requests for global services
  - Supports data decomposition
- Message-driven computation with continuations
  - Does not always return results to parent thread but migrates continuations
  - Percolation for moving work to resources such as GPUs
- Compute complexes extend beyond typical threads

HPX: Distinguishing Features (2)

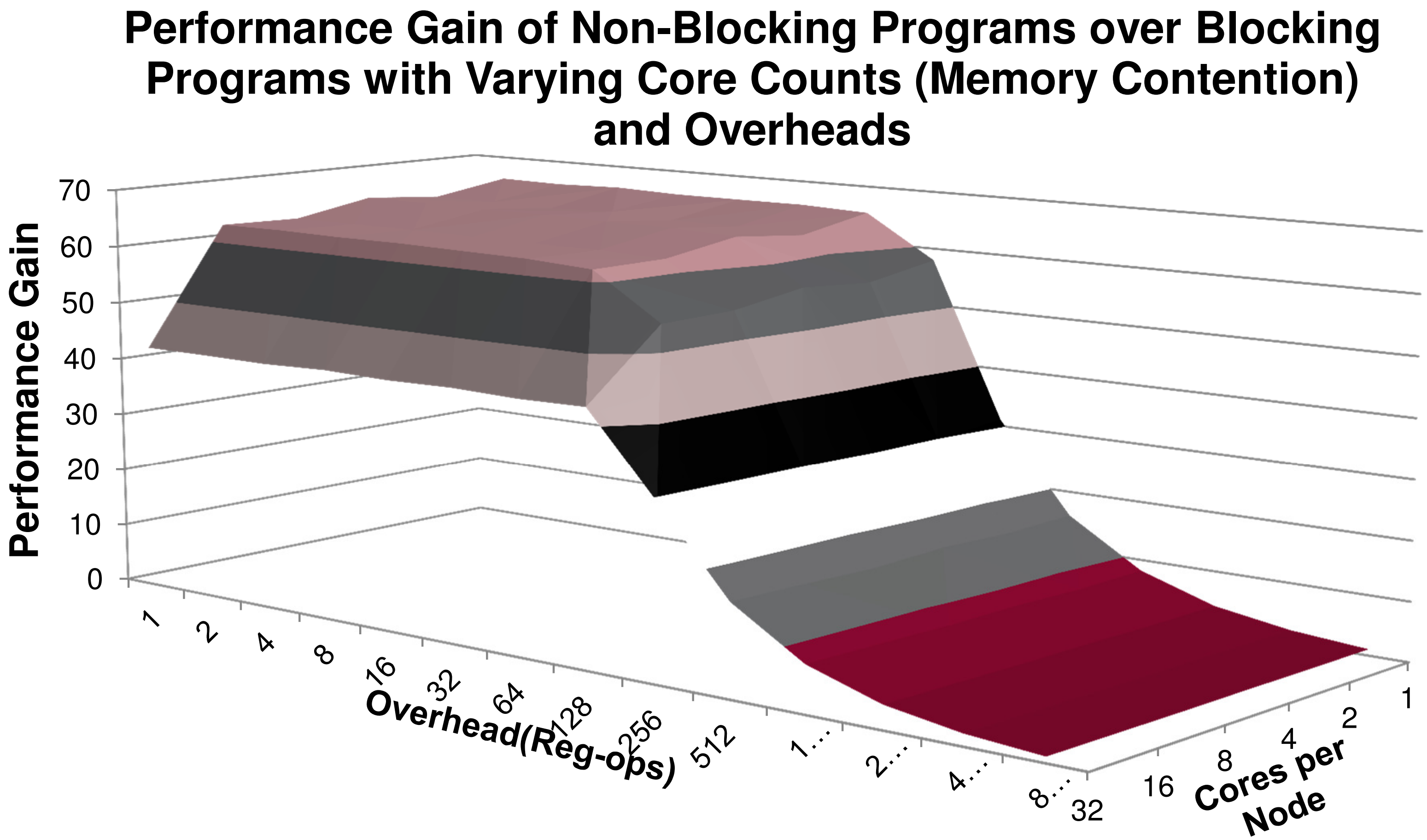
- Embedded data-structure (graphs) control objects
  - Resolves arbitration of simultaneous use requests
- Distributed parallel control state through dynamic graphs of continuations (futures and dataflow)
  - e.g., graph vertex/links insertion or deletion
- Copy semantics through Distributed Control Operations (DCO)
  - Distributed arbitration of access conflicts to structure elements
  - Graph structure changes
- Suspended (Depleted) threads (compute complexes) serve as control objects to build continuation graphs
  - Planning
  - Search spaces
- Responds to OS service requests for multi-node capabilities

AMR in HPX-5

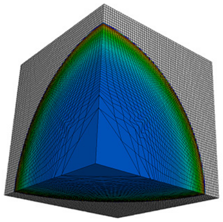
- Model of a short gamma ray burst
- Shows a fluid which is initially confined to a small high density high pressure region.
- Explodes creating spherical shock wave
- Colors indicate the density of the fluid
- 2nd inward moving wave collides at center then reflects back out
- 6 levels of refinement



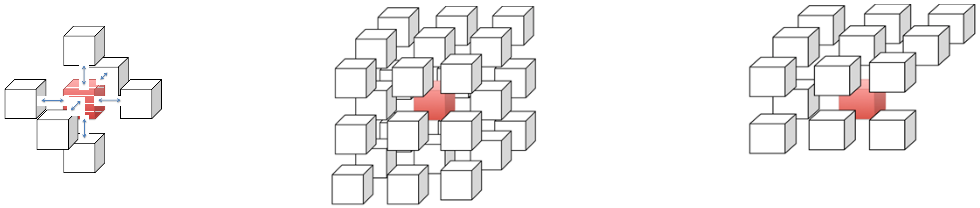
Gain with Respect to Cores per Node and Overhead;
Latency of 8192 reg-ops, 64 Tasks per Core



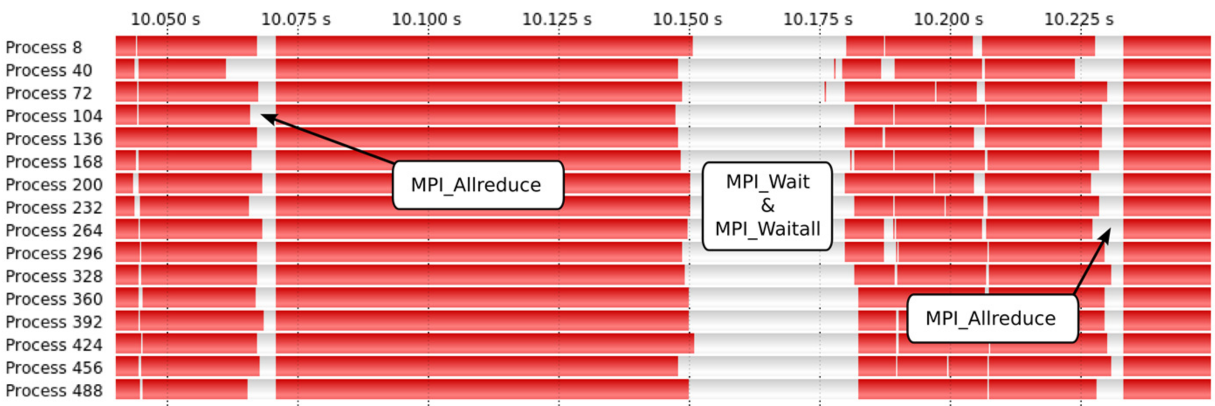
LULESH



- **L**ivermore **U**nstructured **L**agrangian **E**xplicit **S**hock **H**ydrodynamics evolving a Sedov blast wave
- Developed at the ASCR Co-design center at Livermore National Lab
- Contains three types of communication patterns: face adjacent, 26 neighbor, and 13 neighbor communications each timestep



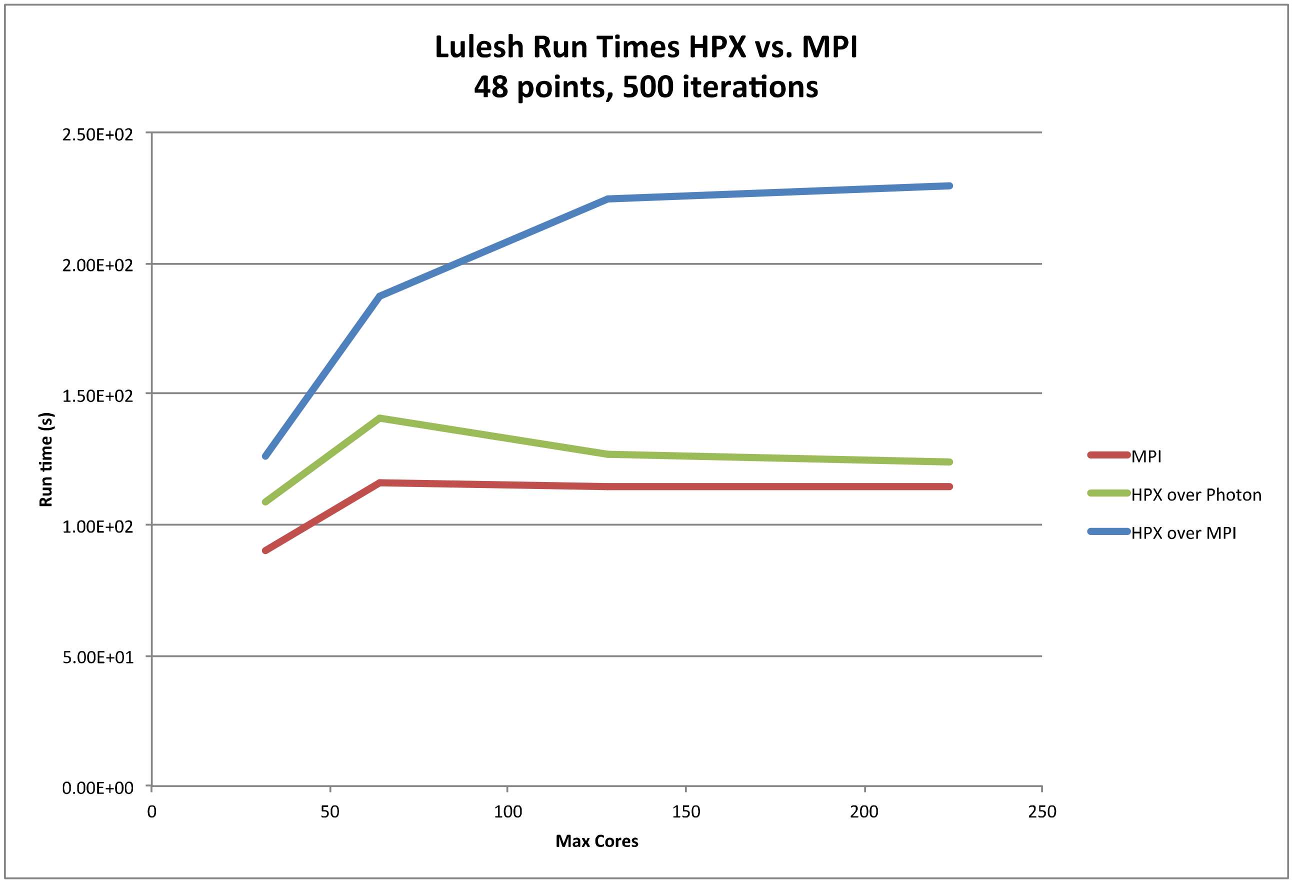
- LULESH is a static, regular computation – very well suited for MPI



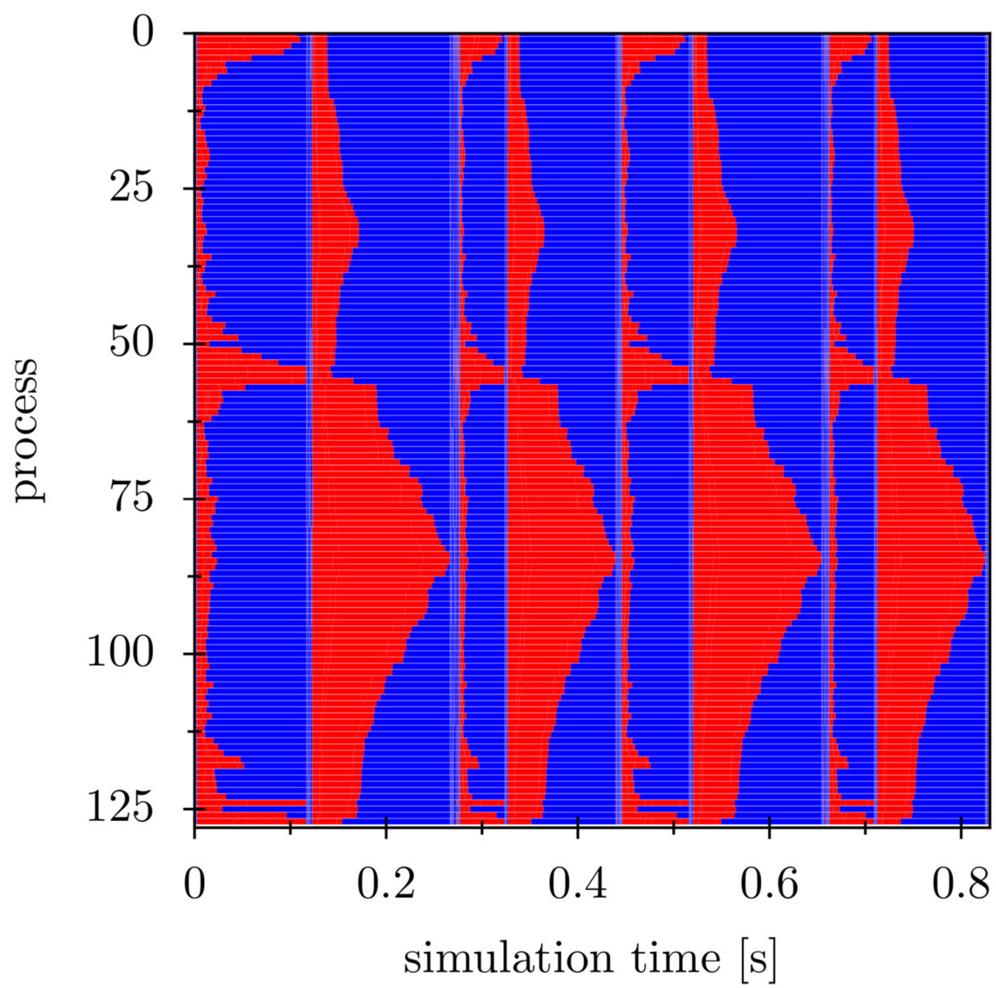
Red indicates computation
White indicates communicatioı

LULESH

- Dynamic techniques can match MPI performance, even for static, uniform computations!



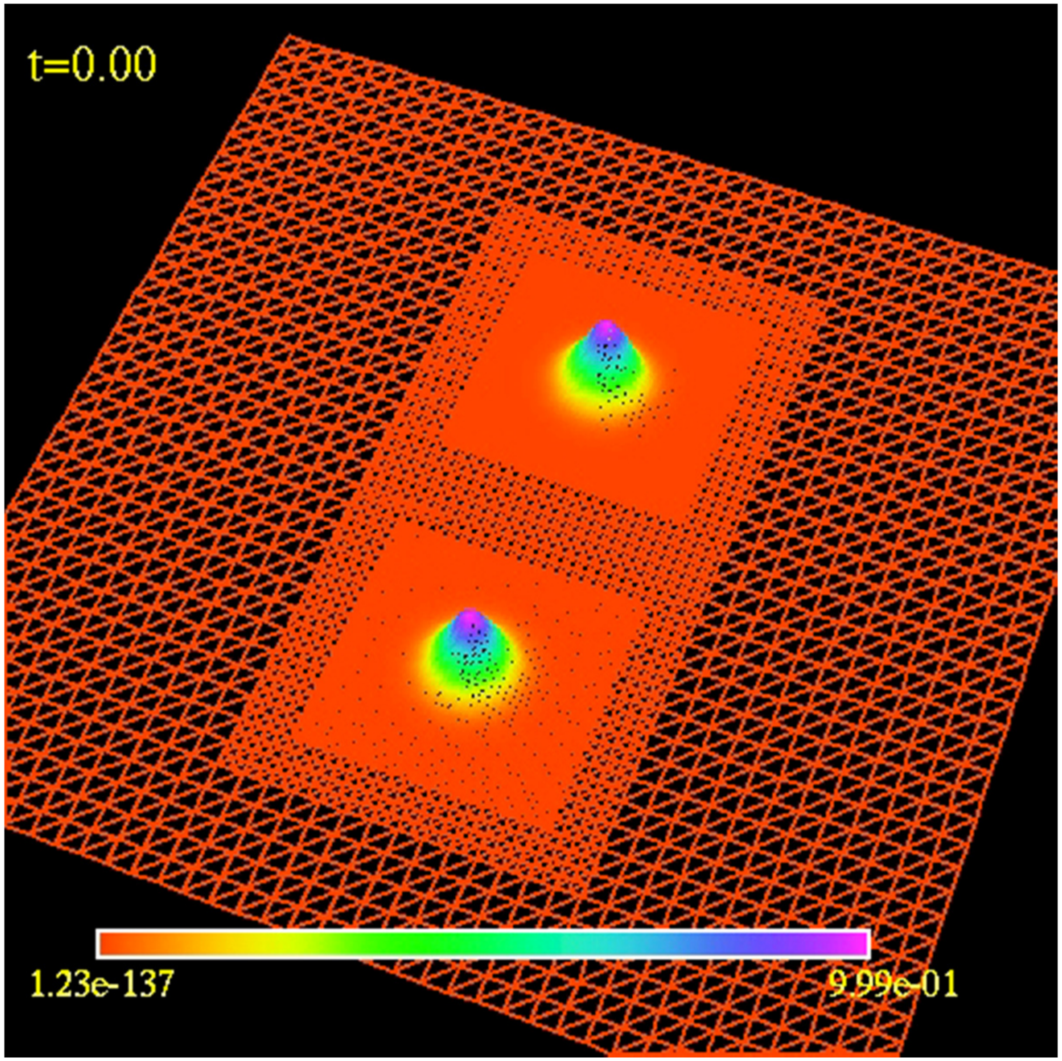
The Negative Impact of Global Barriers in Astrophysics Codes



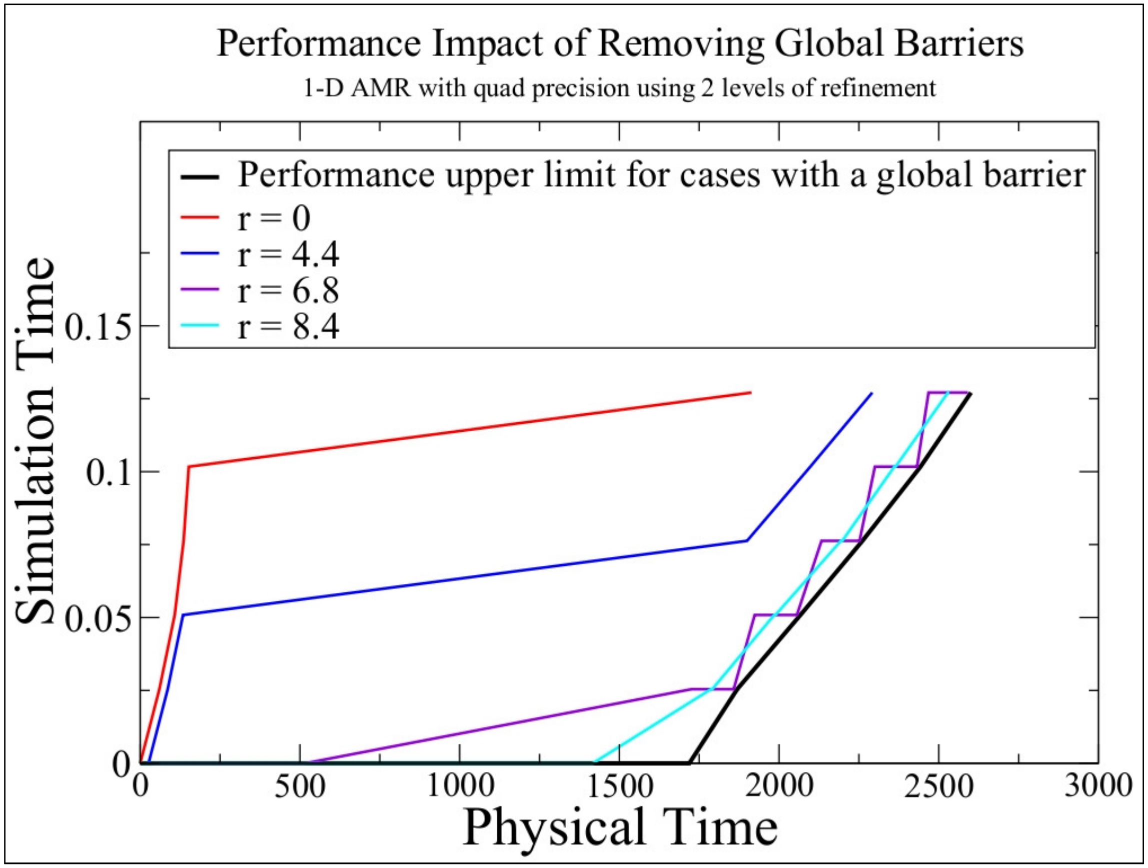
Computational phase diagram from the MPI based GADGET code (used for N-body and SPH simulations) using 1M particles over four time steps on 128 procs.

Red indicates computation
Blue indicates waiting for communication

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



Application: Adaptive Mesh Refinement (AMR) for Astrophysics simulations

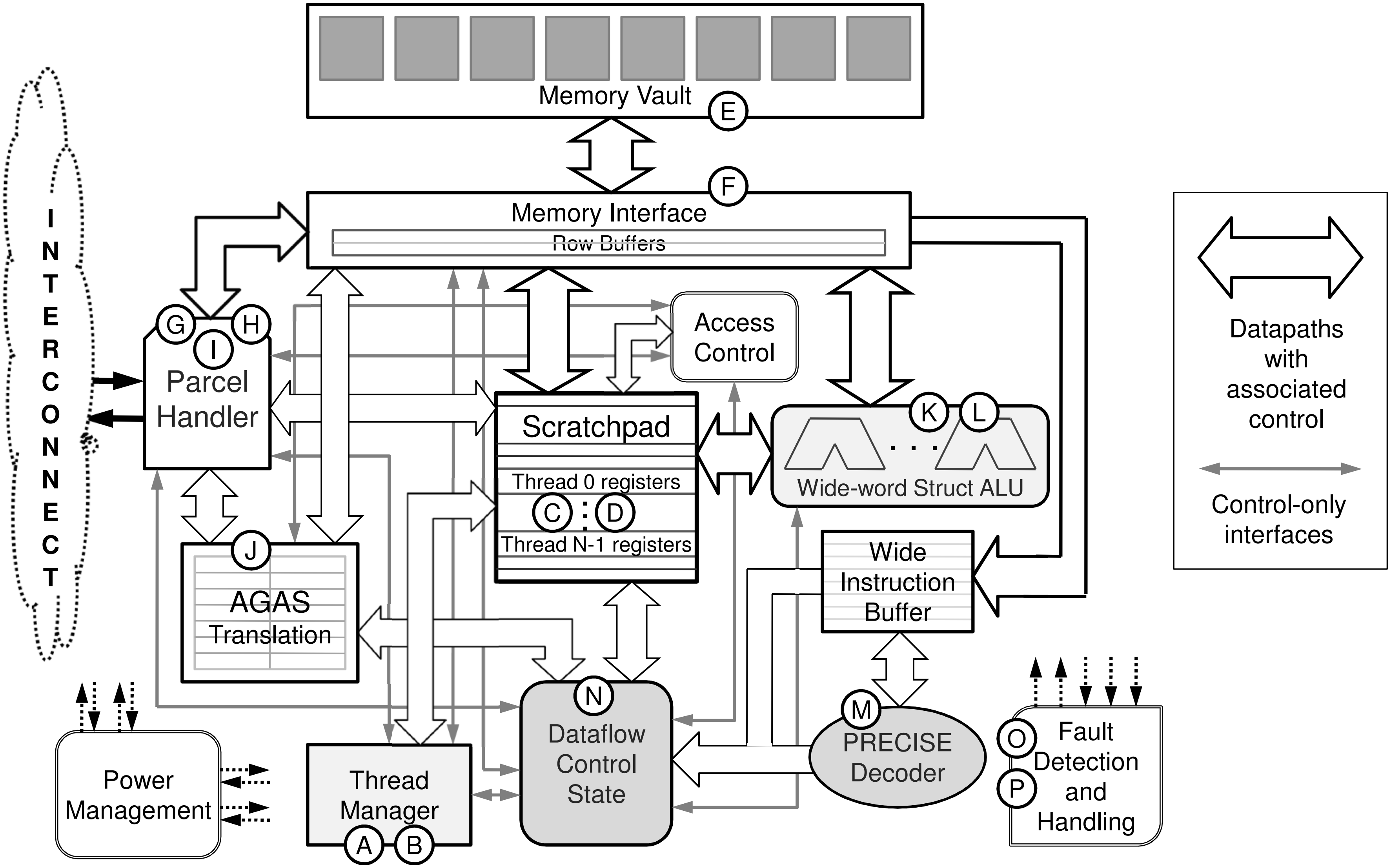


Towards the Global Core Architecture

- A new class of core architectures required
  - Designed for extreme scale
  - Extreme replication
  - Useful for diverse computing domains
- Special requirements
  - Emphasis on interoperability with the other Billion cores
  - De-emphasis on ALU utilization, need availability and hyper-control state
  - Ultra low overheads for
    - Global address translation
    - User thread context creation, termination, and switching including preemption
    - Message-driven computation and networking
  - Minimization of local memory latency, possibly PIM structures
- Operational properties monitoring
  - Fault detection and reconfiguration control
  - Energy/power measurement and modulation
  - Resource utilization and availability

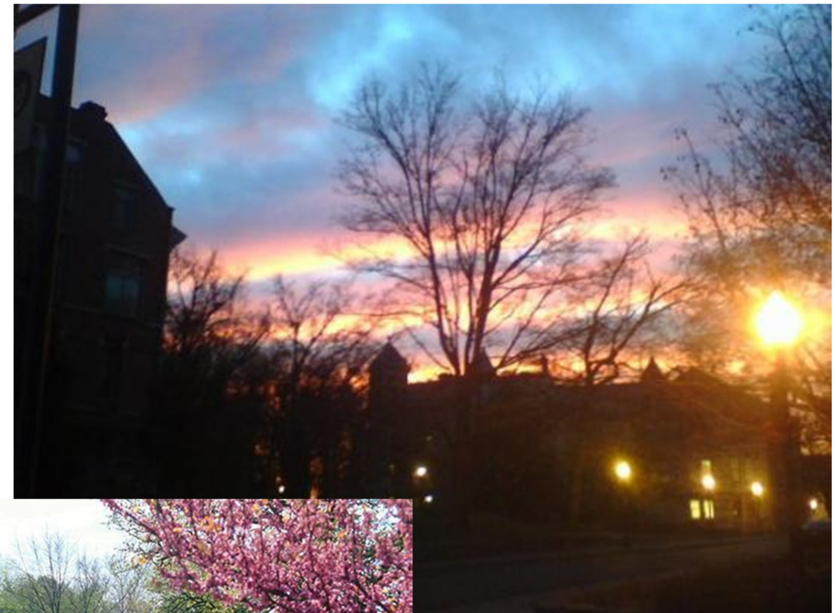
Other Architecture Innovation Opportunities

- Variable width binary instruction encoding
  - Register type tagging
  - Compression
  - Generic operations
  - Non uniform register access ordering
- Wide word ALU
  - High throughput
  - In flight compound atomic multi-field operations
  - Software flexibility with hardware performance
- Dataflow fine grain parallelism
  - Replaces complex ILP mechanisms
  - Latency hiding
  - Hardware support for control flow
- Advanced optical interconnect and 3-D stacking


Memory Vault
E
F
Memory Interface
Row Buffers
I
N
T
E
R
C
O
N
N
E
C
T
G
H
I
Parcel Handler
Access Control
Scratchpad
Thread 0 registers
C
D
Thread N-1 registers
K
L
Wide-word Struct ALU
J
AGAS Translation
Wide Instruction Buffer
N
Dataflow Control State
M
PRECISE Decoder
Power Management
Thread Manager
A
B
O
P
Fault Detection and Handling
Datapaths with associated control
Control-only interfaces

Conclusions

- New high density functional structures are required at end of Moore's Law and are emerging
- Reactive Runtime systems supported by innovations in hardware architecture mechanisms will exploit extremes of parallelism at high efficiency
- Neo-Digital age advances beyond von Neumann architectures to maximize execution concurrency and react to uncertainties of asynchrony




CardCow.com