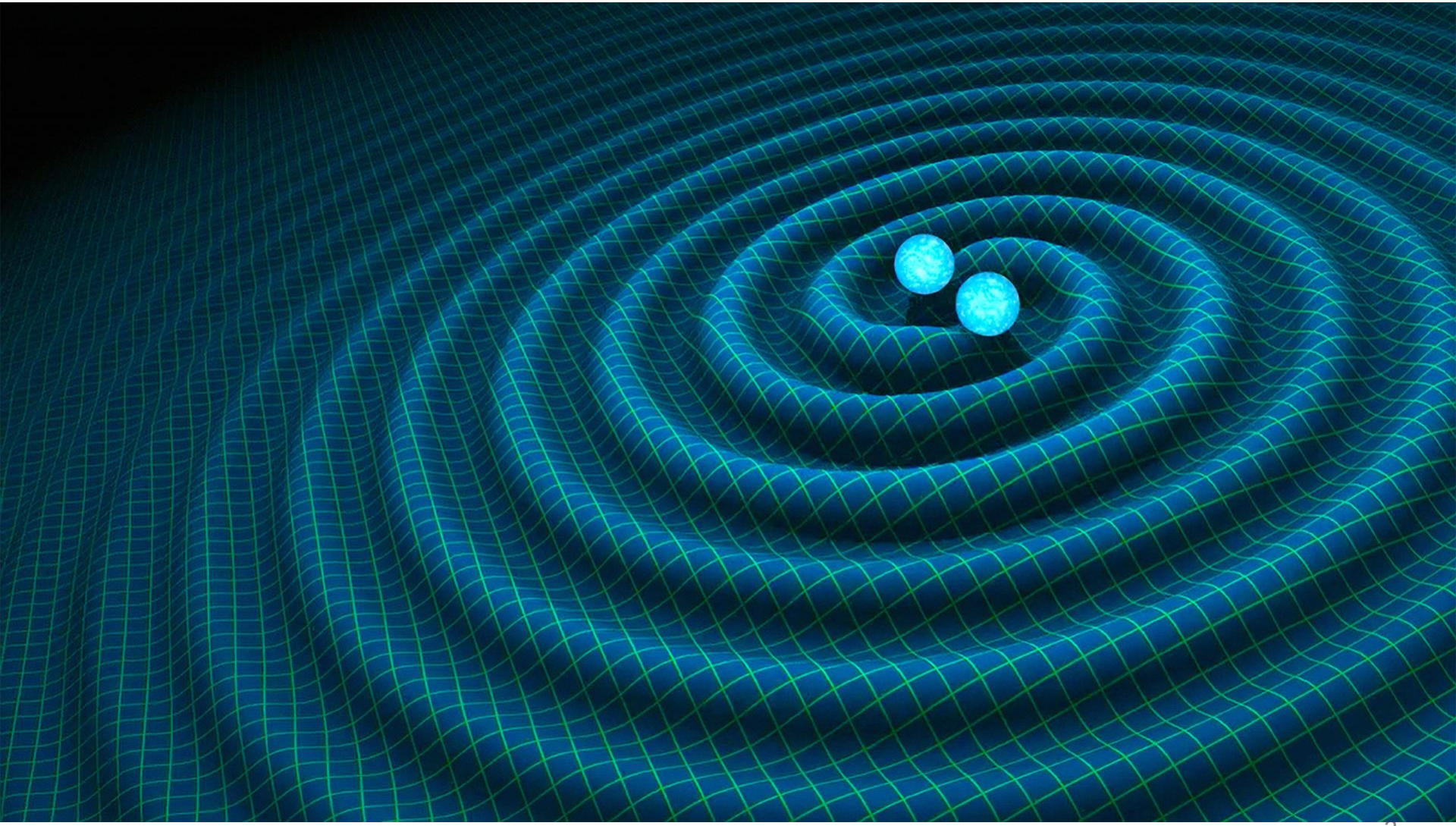
SOIC/ISE Colloquium Series:
Big Data and Big Simulation

CREST Research in Dynamic Adaptive Methods for Extreme Scale Computation

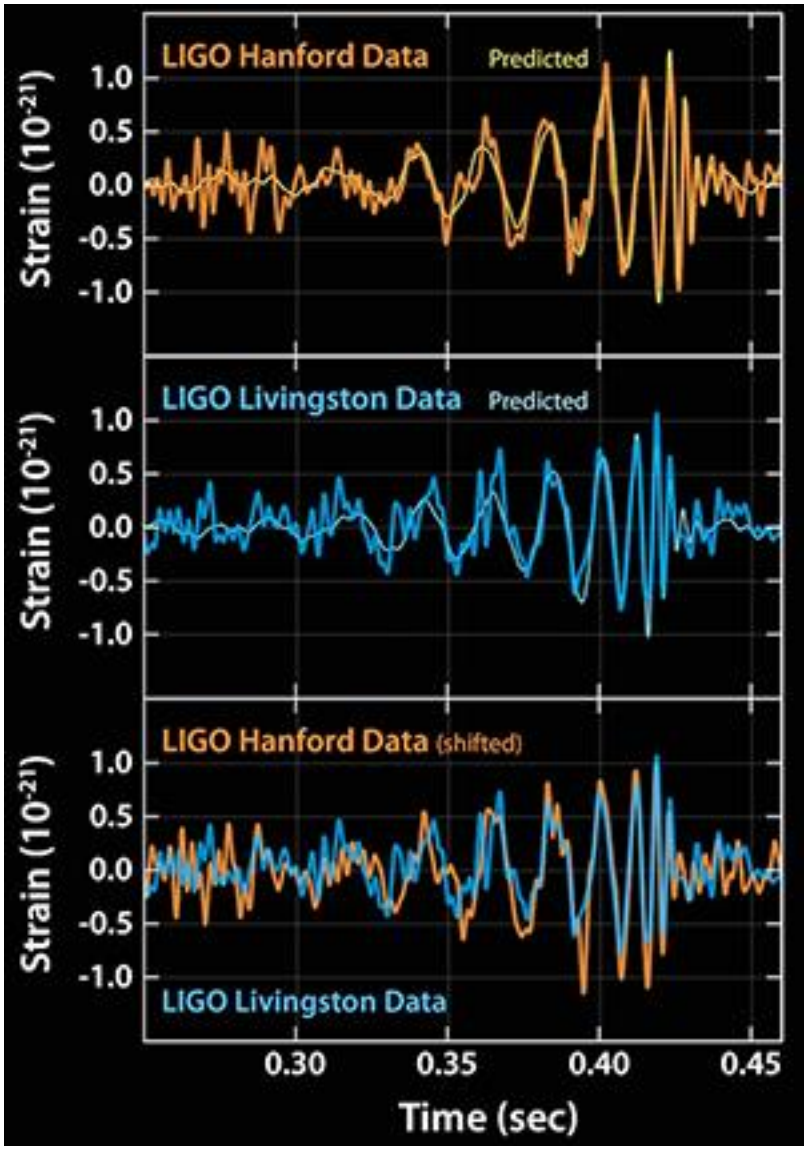
Thomas Sterling
Professor of Electrical Engineering
Director, Center for Research in Extreme Scale Technologies
School of Informatics and Computing
Indiana University

January 9, 2017



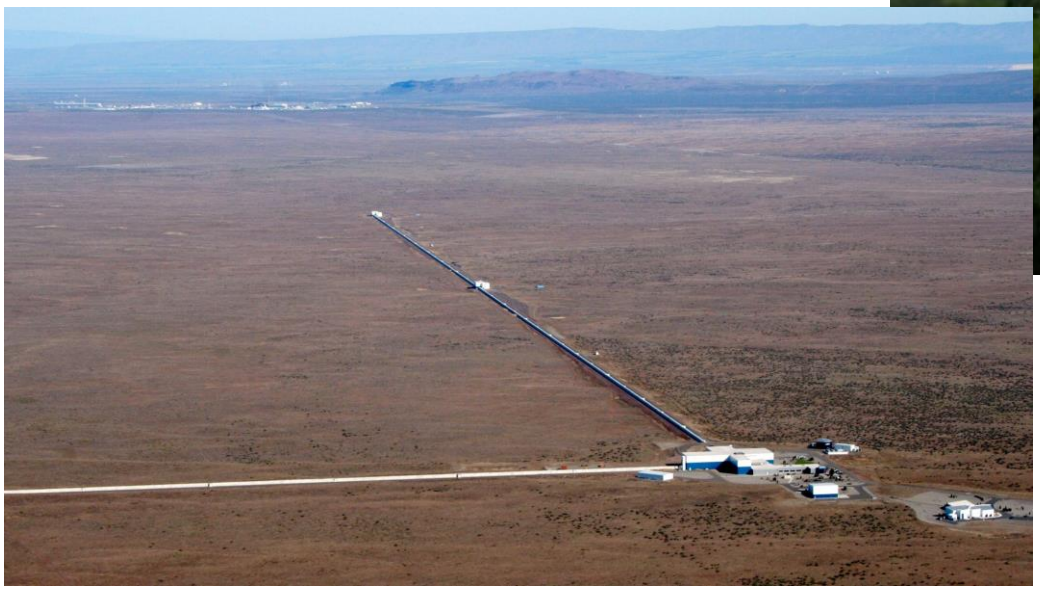
Discovery

- 14 September, 2015
- Combined objects of 29 and 36 solar masses
- Produced a black hole of 62 solar masses.
- Missing 3 solar masses converted to gravitational waves
- Travelled 1.3 billion years to Earth
- 50X all the power of all the stars in the universe



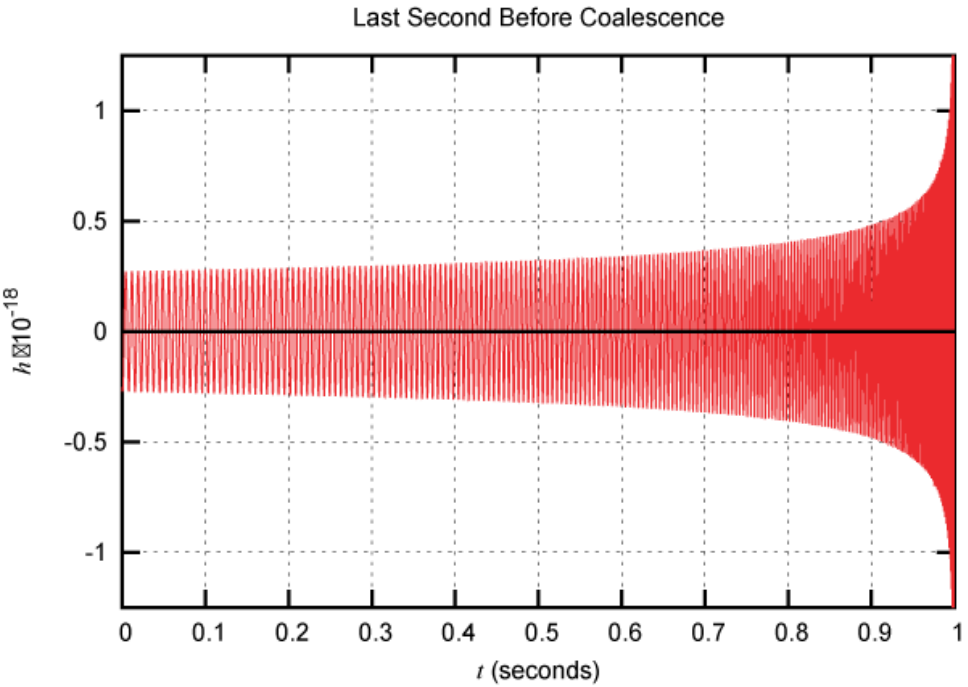
Laser Interferometric Gravitational-wave Observatory (LIGO)

Hanford, WA



Livingston, LA

LIGO Chirp Filter for Signal Target



CREST Research Thrust Areas

- Dynamic adaptive computation for efficiency and scalability
- ParalleX execution model to guide design and interoperability of cross-cutting system stack
- Runtime system development – HPX+
- Advanced network protocols, drivers, and NIC architecture
- Parallel programming intermediate representations
- Parallel applications in numeric and data centric domains
- Architectures
  - Edge functions for overhead reduction related to runtime system acceleration
  - Continuum Computer Architecture – ultra fine grain cellular elements
  - Network lightweight messaging
- Workforce development, education, mentorship

Technology Demands new Response

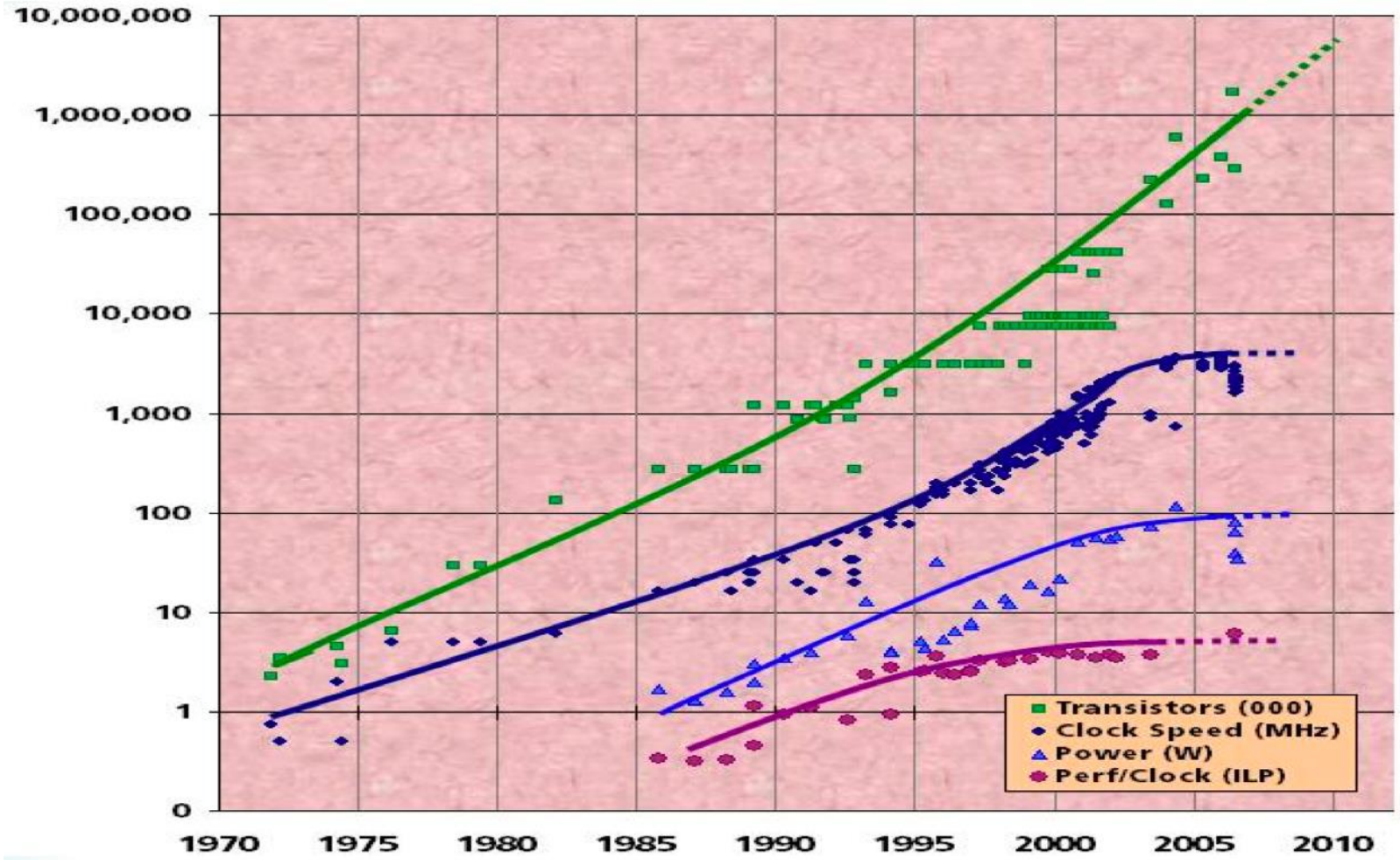


Figure courtesy of Kunle Olukotun, Lance Hammond, Herb Sutter, and Burton Smith

Technology Drivers towards Runtimes

- Sustained efficiencies < 10%
- Increasing sophistication of application domains
- Expanding scale and complexity of HPC system structures
- Moore's Law flat-lining and loss of Dennard scaling
- Starvation, latency, overhead, contention
- Asynchronous data movement and memory access
- Energy/power
- Changing priorities of component utilization versus availability
- Collision of parallel programming interfaces for user productivity
- Diversity of architecture forms, scales, generations requiring performance portability

Dynamic adaptive computation

- Avoid limitations of “ballistic” computing by “guided” control
- Exploit status information of system and computation at runtime for resource management and task scheduling
- Take advantage of over decomposition naturally
- Improve user productivity by unburdening of explicit control
- Enable performance portability through real-time adjustment to hardware architecture capabilities
- Expose and exploit lightweight parallelism through discovery from meta-data
- Requires:
  - Modification to compilation
  - Addition of runtime systems
  - Possible support through architecture enhancements
  - Consideration of parallel algorithms

CREST Engaged in Co-Design for Dynamic Adaptive Computational Systems

- Runtime systems only part of total system hierarchical structure
- Must be defined/derived in part by support for and interoperability with:
  - programming model
  - Compiler
  - Locality (node) OS
  - Processor core architecture
- Architecture will have to be designed to reduce overheads incurred by runtime systems; e.g.,:
  - Parcels to compute complexes
  - Global address translation
  - Context creation, switching, and garbage collection
  - Data and context redistribution for load balancing

Performance Factors - SLOWER

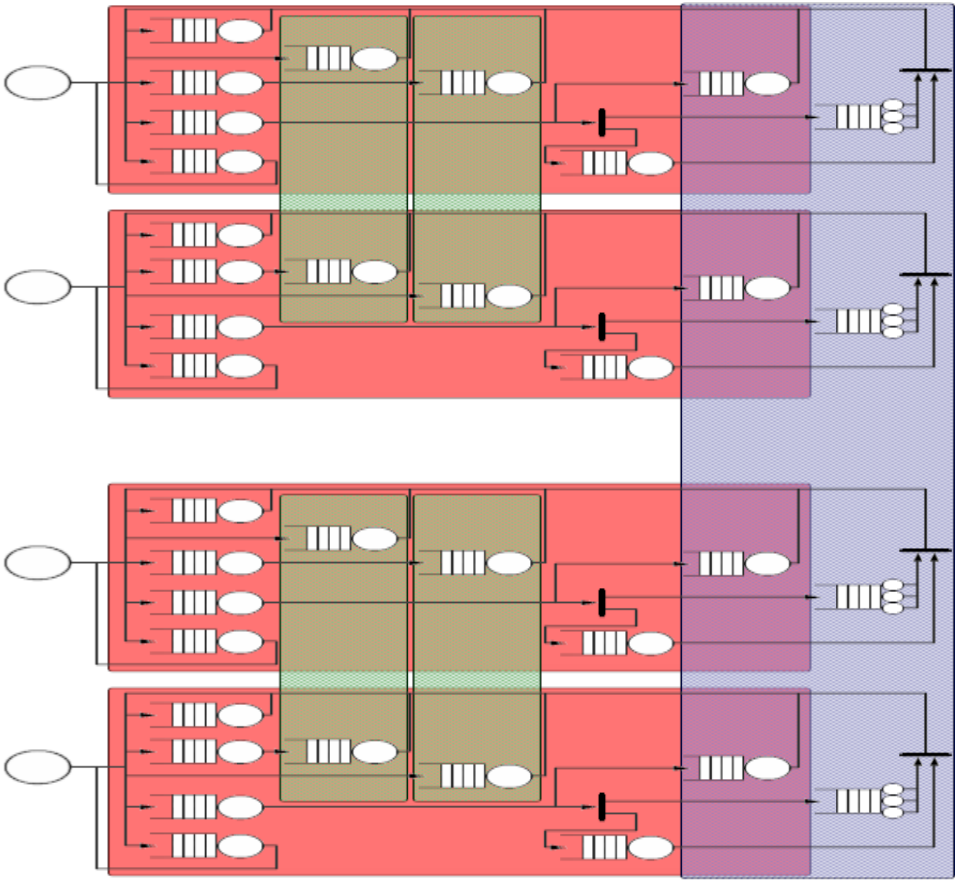
$$P = e(L,O,W) * S(s) * a(r) * U(E)$$

P – 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

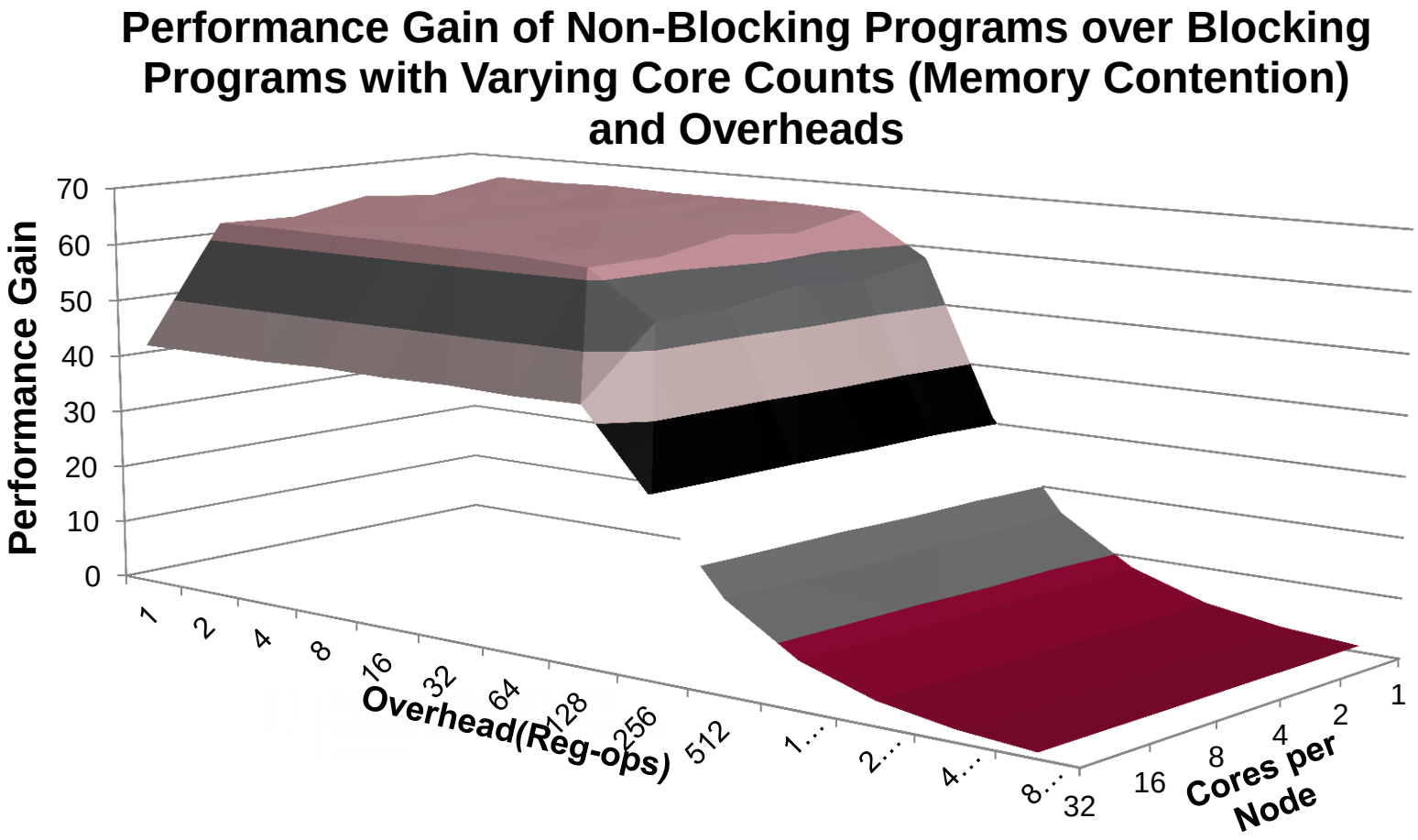
Performance Model, Full Example System

Modeling the full example system



- Example system:
  - 2 nodes,
  - 2 cores per node,
  - 2 memory banks per node
- Accounts for:
  - Functional unit workload
  - Memory workload/latency
  - Network overhead/latency
  - Context switch overhead
  - Lightweight task management (red regions can have one active task at a time)
  - Memory contention (green regions allow only a single memory access at a time)
  - Network contention (blue region represents bandwidth cap)
  - NUMA affinity of cores
- Assumes:
  - Balanced workload
  - Homogenous system
  - Flat network

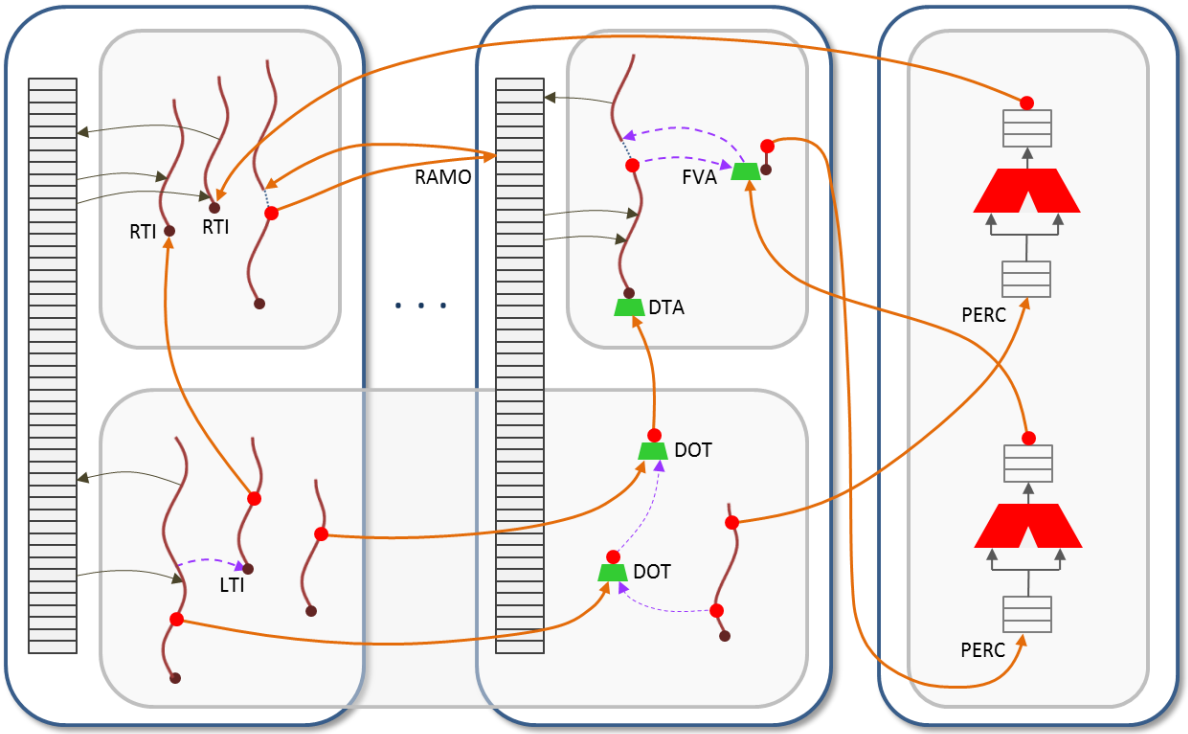
Gain with Respect to Cores per Node and Overhead; Latency of 8192 reg-ops, 64 Tasks per Core



ParalleX Execution Model

- Execution model establishes principles for guiding design of system stack layers and governing their functionality, interfaces, and interoperation
- Paradigm shifts driven by advances in enabling technologies to exploit opportunities and fix problems
- Execution models capture computing paradigms
  - Von Neumann, Vector, SIMD, CSP
- Formal representation
  - PNNL-2 led EM2 project
  - Operational semantics specification
  - Prof. Jeremy Siek and Dr. Mateos Cimini
- Employed in
  - Sandia XPRESS Project
  - NNSA PSAAP-2 C-SWARM Project
  - PNNL EM2 project

Distinguishing Features of ParalleX/HPX+



Locality
Process
Local Memory
LCO
Accelerator

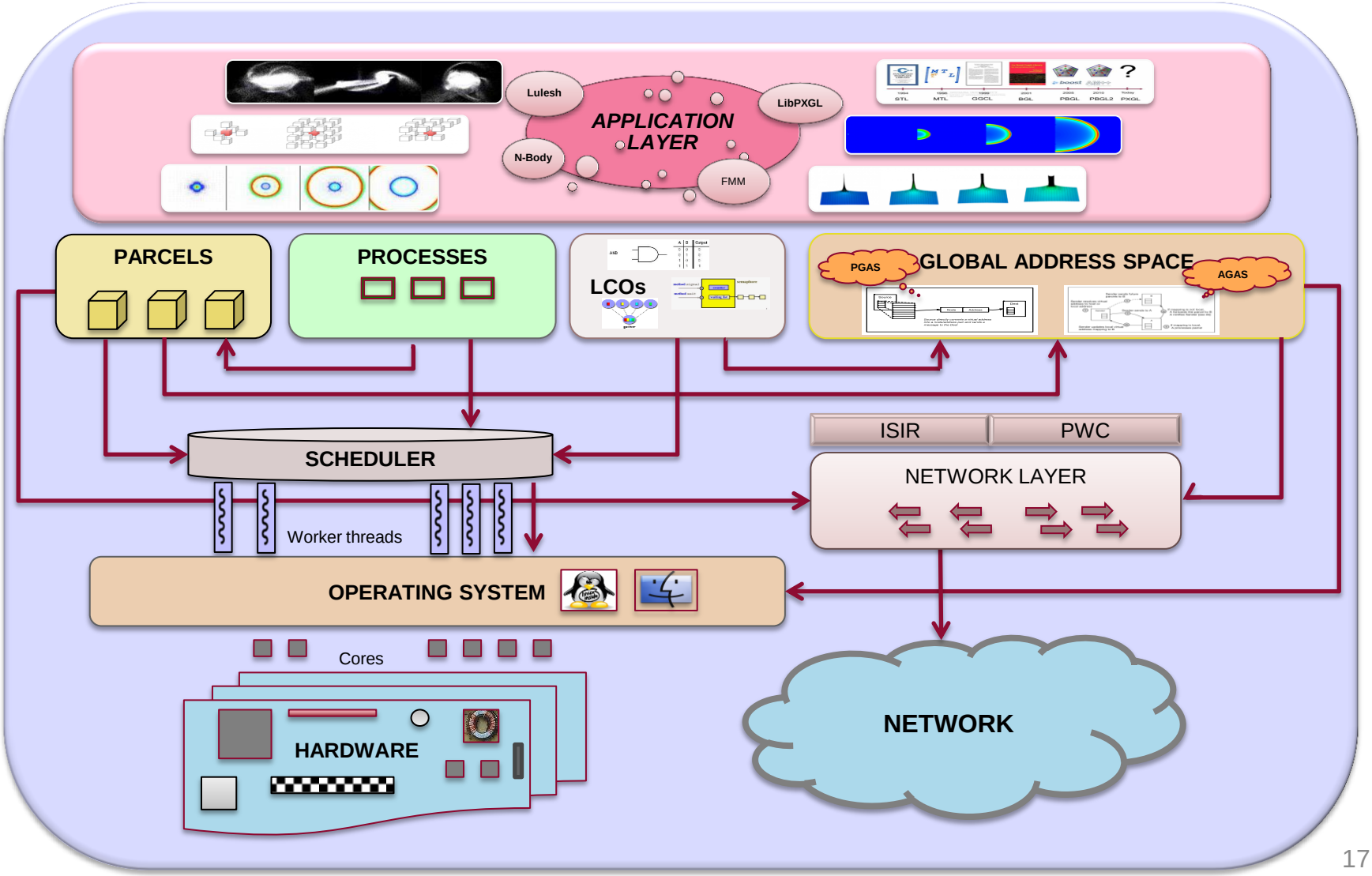
Thread
Suspended Thread
Local Memory Access
AGAS Address Lookup
Local Action
Parcel

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

HPX+: Runtime Software System Development

- First reduction to practice of ParalleX execution model
- Thread scheduler
- Global address system (AGAS)
- Message-driven computation
- Multi-nodal dynamic processes
- Futures/dataflow synchronization and continuation
- Percolation for heterogeneous computation
- Introspection data acquisition and policy-based control
- Load balancing hooks/stubs
- Low level intermediate representation for source to source compilation and heroic users/experimenters
- Drives architecture investigations

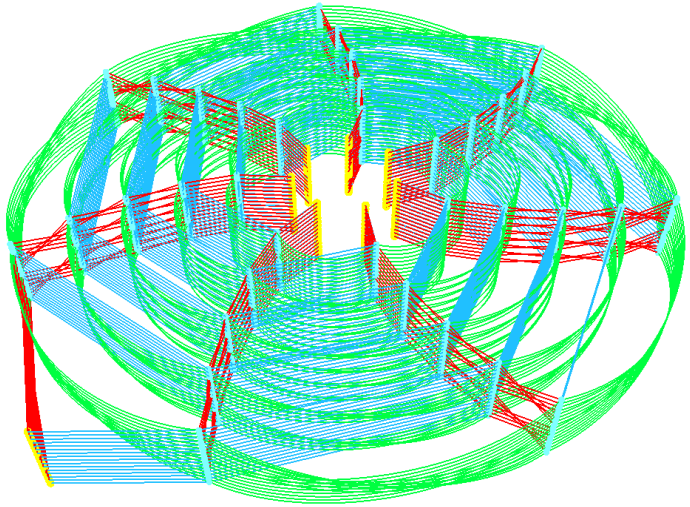
HPX+ Runtime Software Architecture



Courtesy of Jayashree Candadai, IU

Advanced System Area Networks

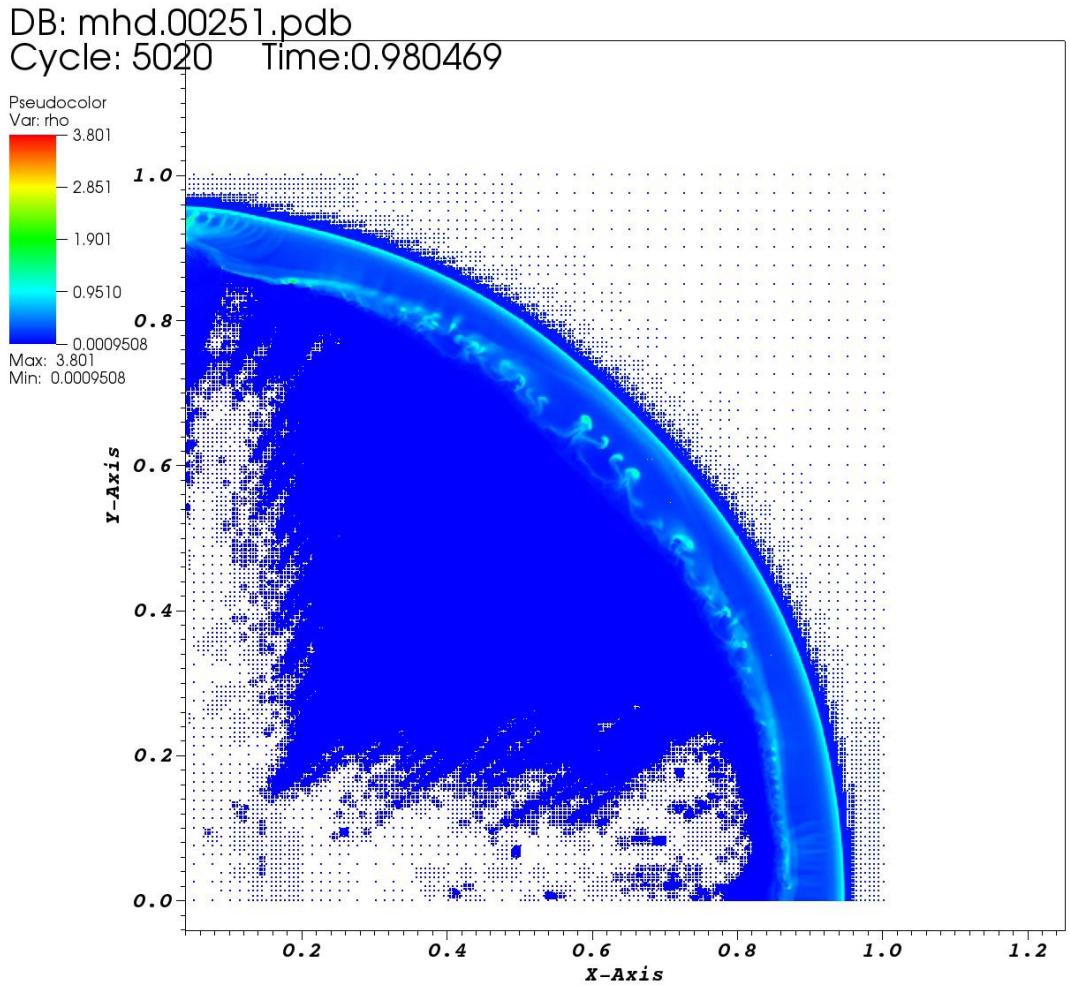
- Photon (Prof. Martin Swany, Ezra Kissel)
  - In house developed network protocol
  - Lightweight messaging
  - Put with completion
  - HPX+ built on top of it
- Parcels (Luke Dalessandro)
  - Advanced form of active messages in HPX
  - Message-driven computation
  - Migration of continuations
- Data Vortex with UITS
  - Small machine, DIET
  - Emphasis on lightweight messaging
  - Many in situ tests
  - Larger machines at PNNL & IDA



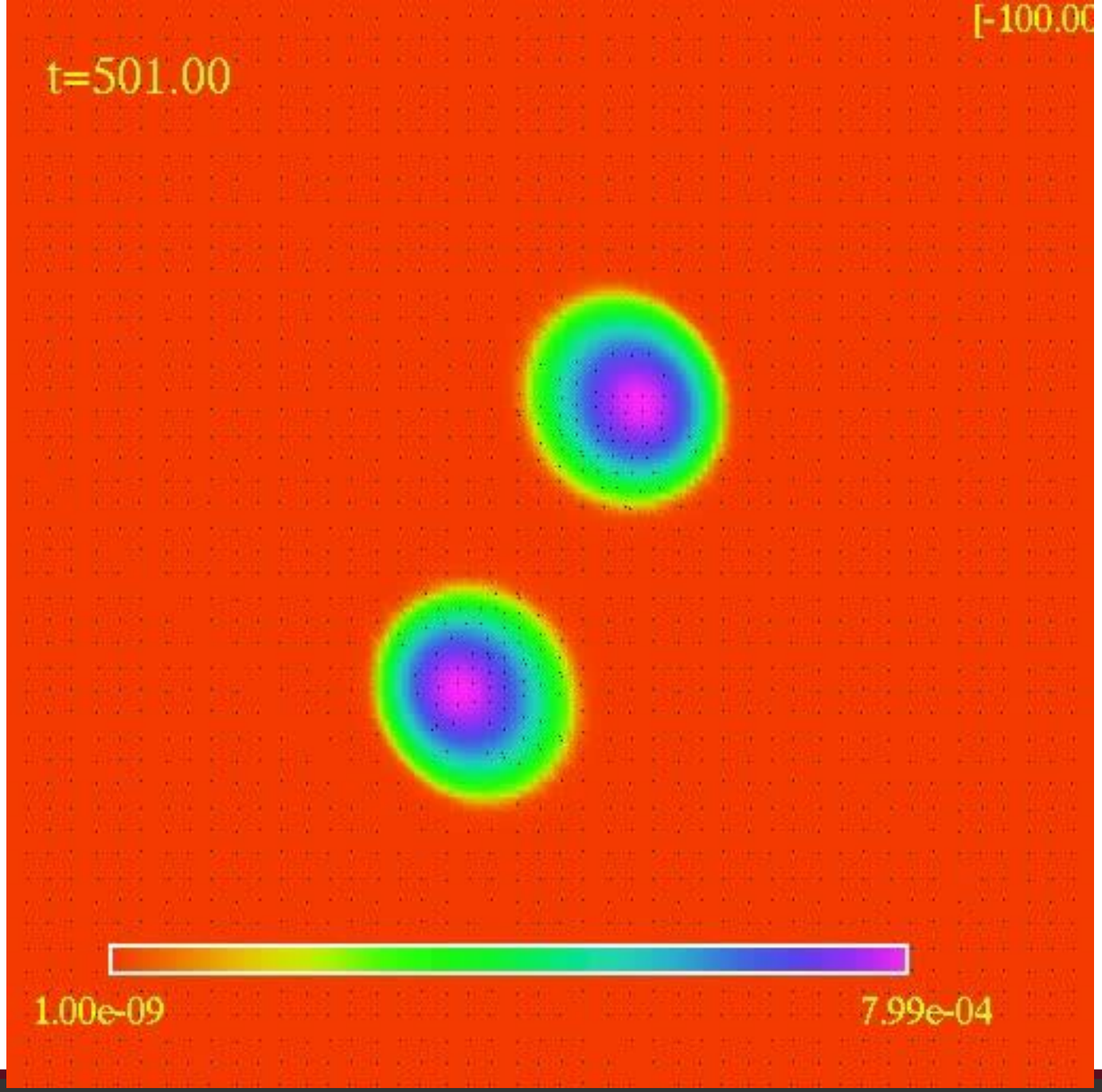
Adaptive Parallel Applications

- Adaptive mesh refinement (Matt Anderson)
- Fast multipole methods (DASHMM) (Bo Zhang)
- Barnes-Hut N-body (Jackson DeBuhr)
- Shock-wave material physics with V&V & UQ (C-SWARM)
- Wavelets (with Un. Notre Dame)
- Extremely Large Network processing (with Katy Borner)
- Brain Simulation (EPFL)
- Regular Applications
  - LULESH
  - Linpack
  - HPCG

Wavelet Adaptive Multiresoultion



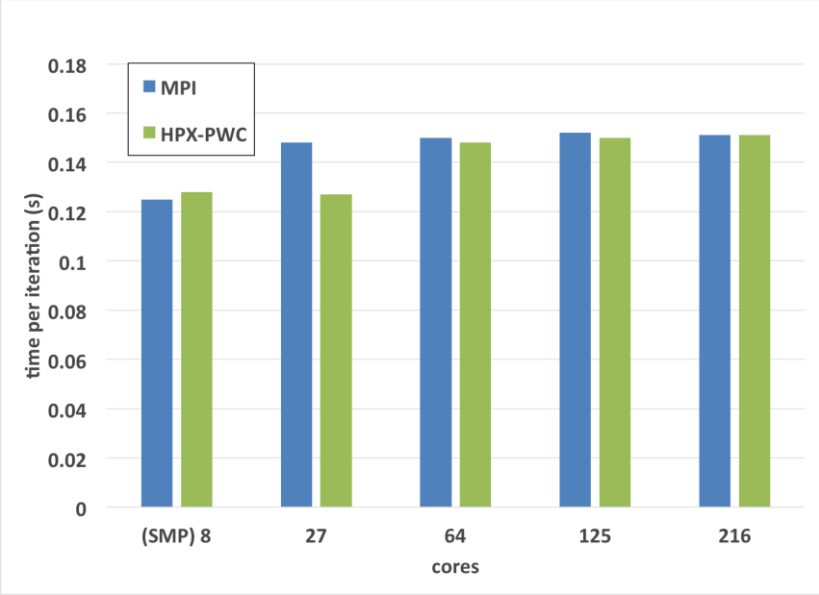
Courtesy of Matt Anderson, IU

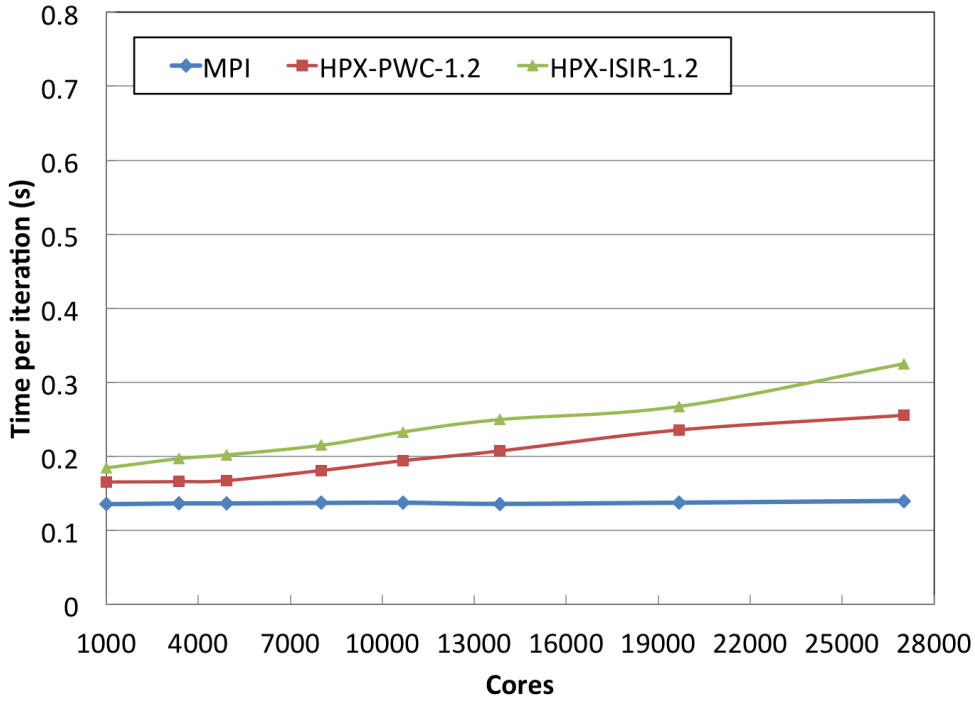

t=501.00
[-100.00
1.00e-09
7.99e-04

Not All Apps benefit from Runtimes

- One size does not fit all
- Applications with key properties best served by CSP
  - with uniform and regular execution,
  - with mostly local data access,
  - Static data structures
  - with coarse granularity
- Scheduling to be determined at compile/load time
- Data structure and distribution static
- Runtime overhead costs detrimental
  - It should be smart enough to know when to get out of the way
- Active scheduling policies can have deleterious effects

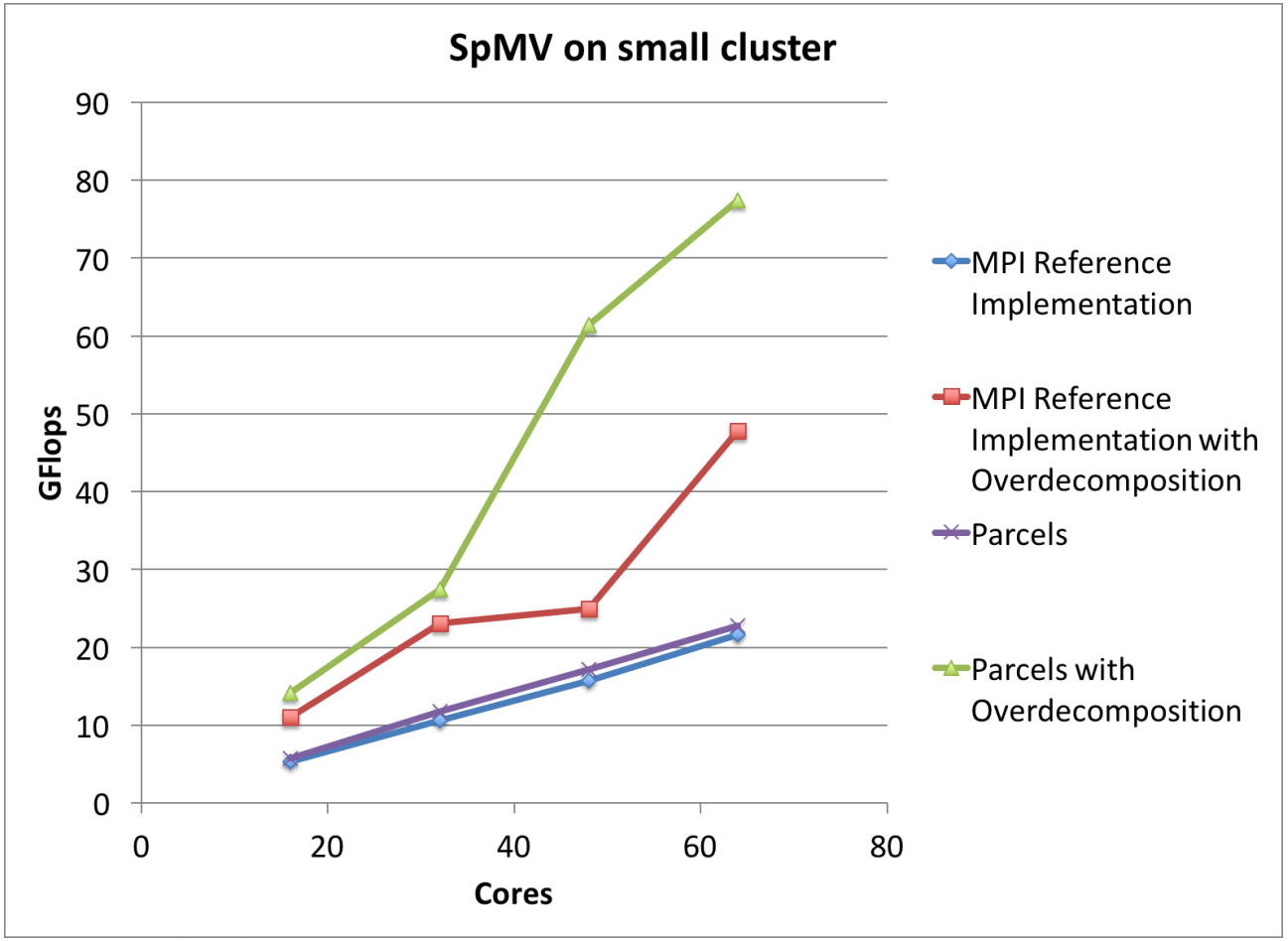
LULESH HPX+ Performance





Courtesy of Matt AndersonIU

SpMV in HPCG

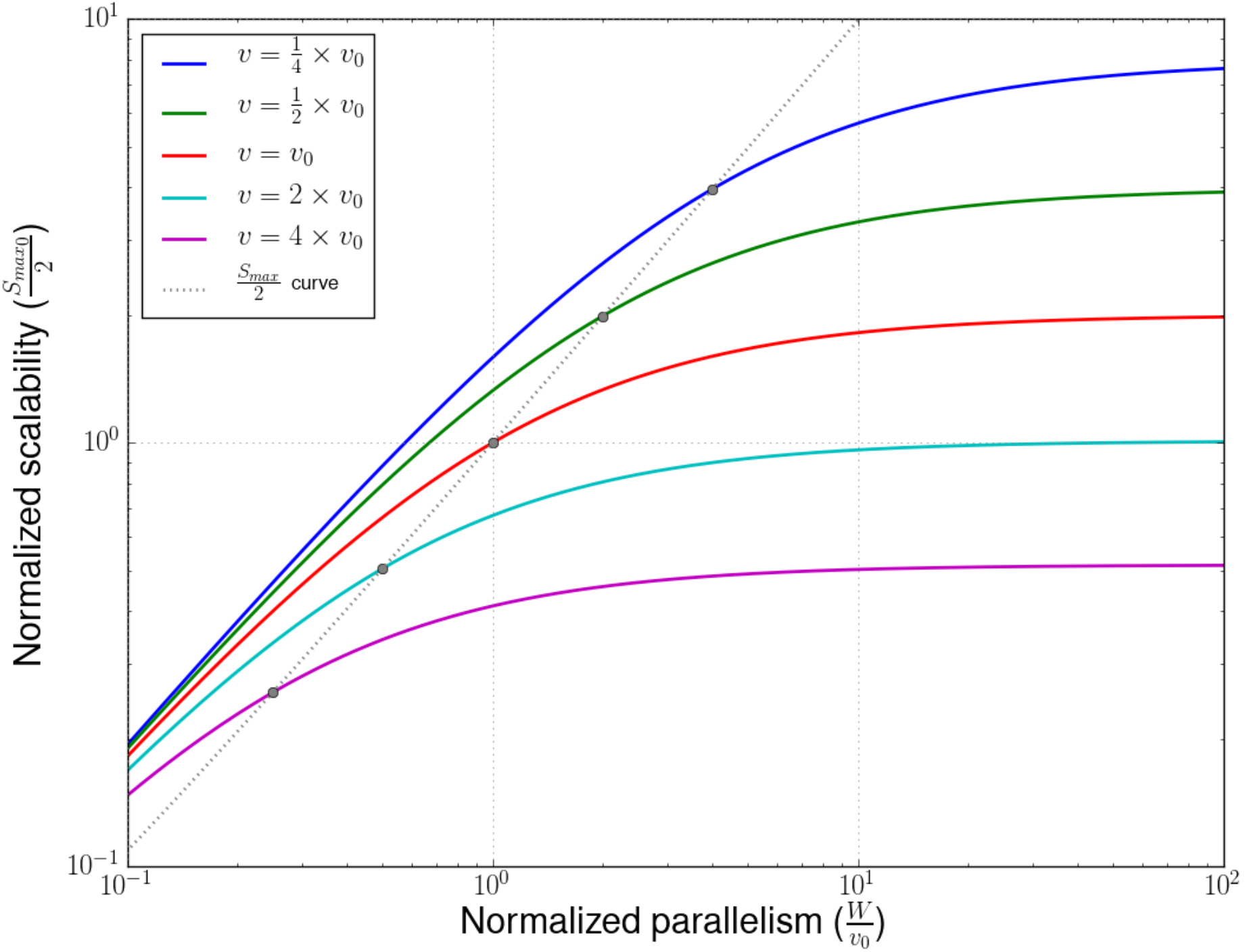

SpMV on small cluster
GFlops
90
80
70
60
50
40
30
20
10
0
0
20
40
60
80
Cores
MPI Reference Implementation
MPI Reference Implementation with Overdecomposition
Parcels
Parcels with Overdecomposition

Problems caused by HPC Runtime

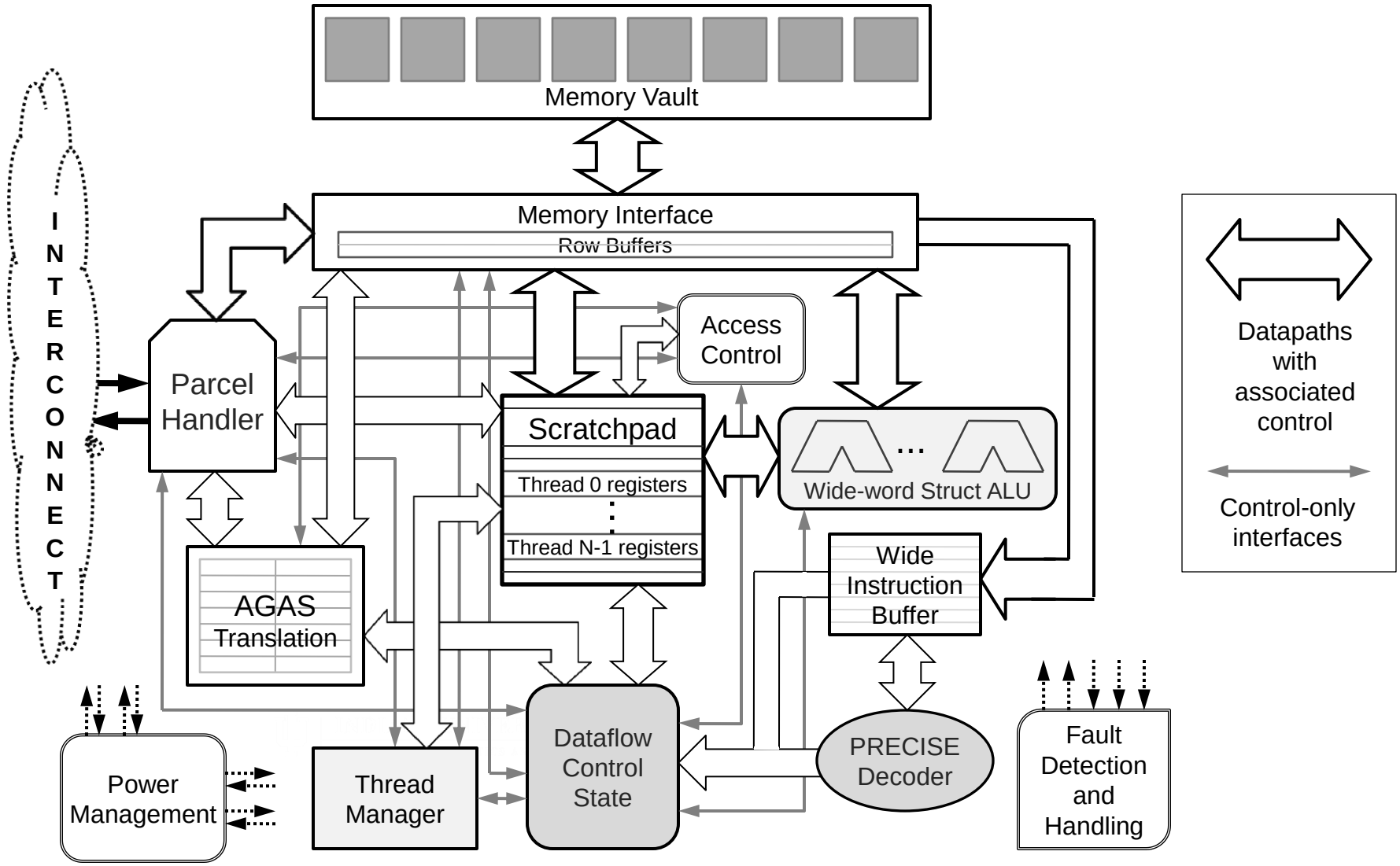
- Experimental
  - Issues for performance, robustness, deployment
  - Possible exception: Charm++ is mature software
- Impose additional problems
  - increased system software complexity
- Added overheads,
  - Paradox: to reduce time, add work
  - Time and energy costs of task scheduling and resource management
- Uncertainty about programming interfaces
  - New execution models cross-cutting of system layers
- Support for legacy codes
  - Continuity of working codes on future machines
- Workload interoperability such as libraries
  - Separately developed functions, filters, solvers,

Architecture for Runtime Acceleration

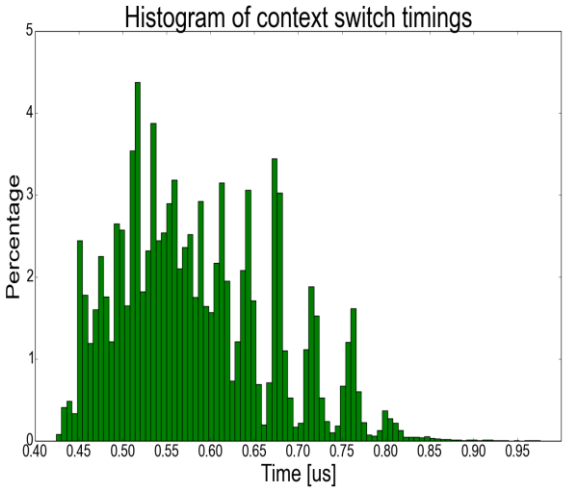
- Reduction of overheads for runtime mechanisms
- Reduced overheads permit finer grained parallelism
- Example mechanisms feasible with conventional cores
  - Thread create & terminate
  - Thread context switch
  - Thread queue management
  - Parcel send/receive/complete and queuing
  - Global address translation
- Mechanisms disruptive to cores
- FPGAs can perform many of the required runtime functions

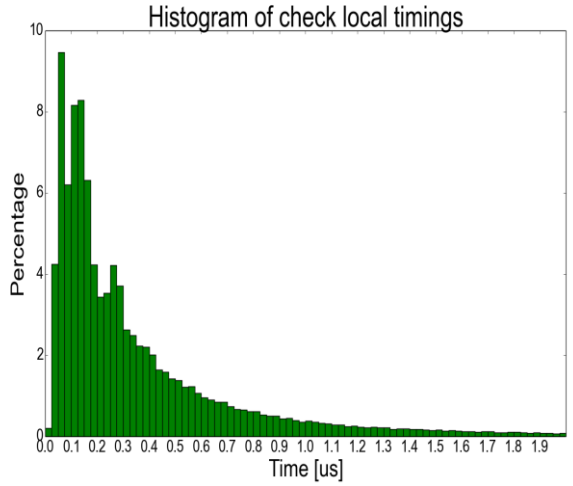

$v = \frac{1}{4} \times v_0$
$v = \frac{1}{2} \times v_0$
$v = v_0$
$v = 2 \times v_0$
$v = 4 \times v_0$
$\frac{S_{max}}{2}$ curve
Normalized scalability ($\frac{S_{max0}}{2}$)
Normalized parallelism ($\frac{W}{v_0}$)
10^{1}
10^{0}
10^{-1}
10^{-1}
10^{0}
10^{1}
10^{2}

EMP Structure



Time Required to Accomplish Runtime Overheads





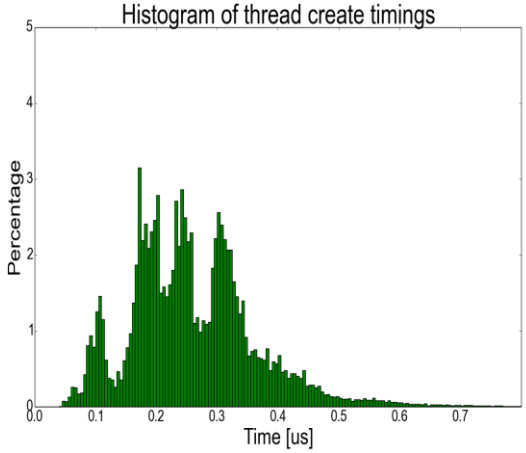
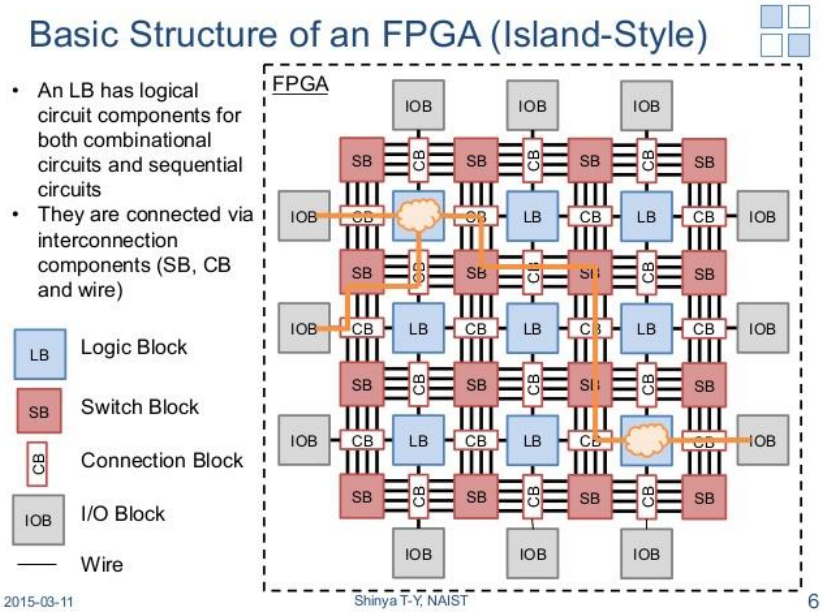


Chart courtesy of Daniel Kogler, IU

Field Programmable Gate Arrays

- Large scale integration with VLSI components
- Reconfigurable for generalized logic circuit synthesis
- Slower that custom logic
- Fast enough to handle message and memory traffic at peak speeds
- Rapid prototyping and small run product delivery
- Includes industry standard interfaces and functional units
- Updatable with design improvements and new functions

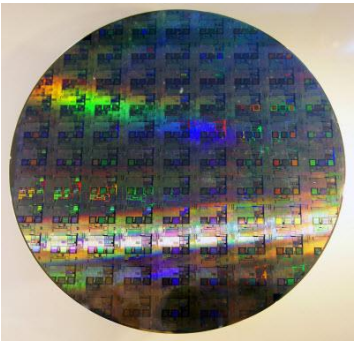


FPGA Early Proof-of-Concept

- FPGAs may permit early implementation and testing of some of these concepts
- Those requiring core intrinsics may be beyond this technology because of separation of control path
- Makes possible a vehicle of technology transfer using industry standard interfaces
  - Physical FPGAs
  - Abstract VHDL/Verilog design specifications
- Integrated multi-components with FPGA layer E.g., Intel
  - NICs and FPGAs
- Time constants comparable to message incident rate and main memory access rates
  - In spite of lower clock rates and device densities

Extreme Scale Parallel Computer Architecture

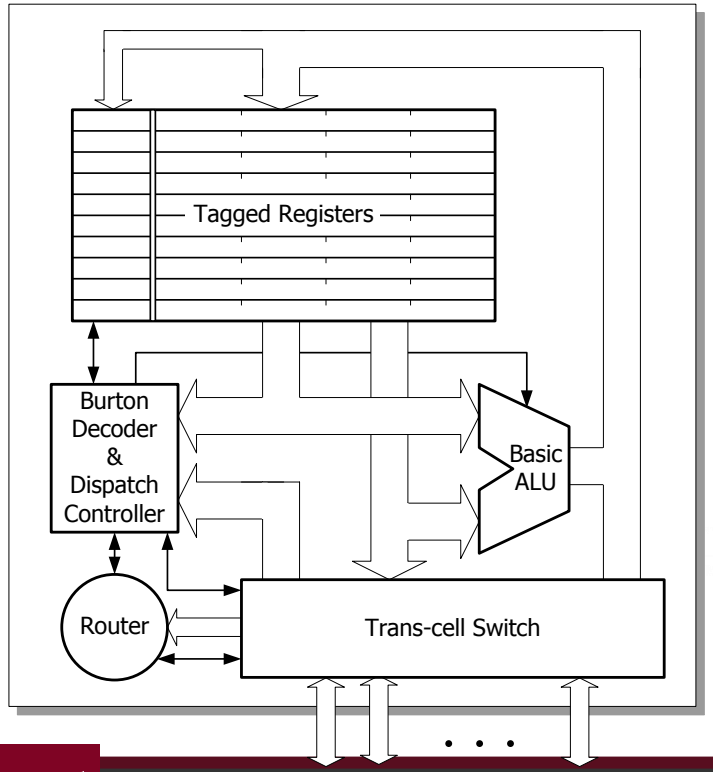
- Continuum Computer Architecture (CCA) (Maciek Brodowicz)
  - Initial work at Caltech/CACR under DARPA sponsorship
  - Exploratory concepts, not needed with enabling technologies
- Architecture in post Moore's Law era
  - Investigates the limits of lightweight homogeneous structures
    - Ultra simple in design
  - Non von Neumann architecture
    - Eliminates FPU as critical optimization
    - Eliminates sequential issue
    - Eliminates separation of processing and memory

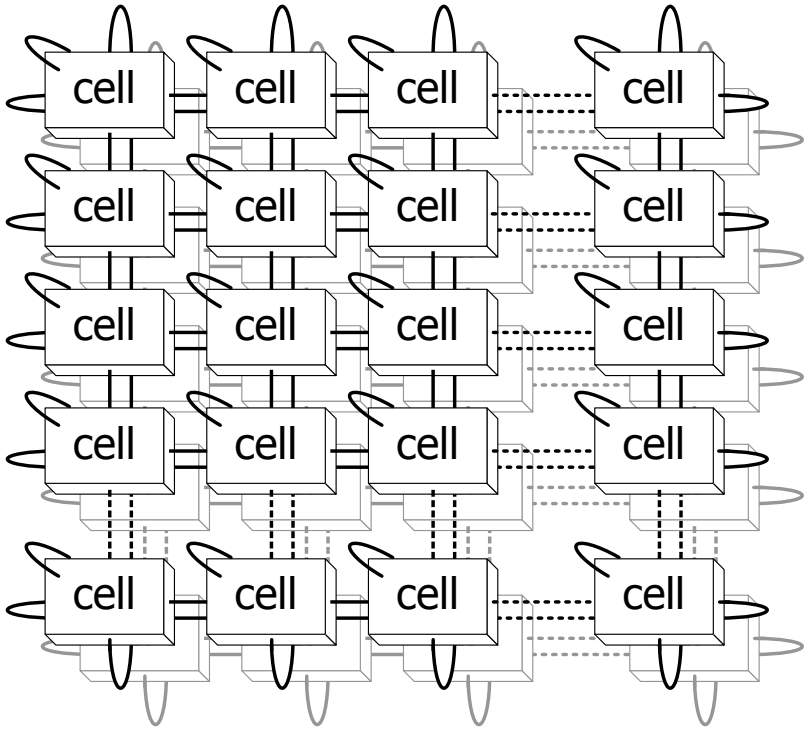


- ParalleX as guiding principles of parallelism and asynchronous control
  - Embeds much of HPX functionality in hardware as primitives
  - Ideal for parallelism discovery from graph structure meta-data

CCA Structure: *Simultac Fonton*

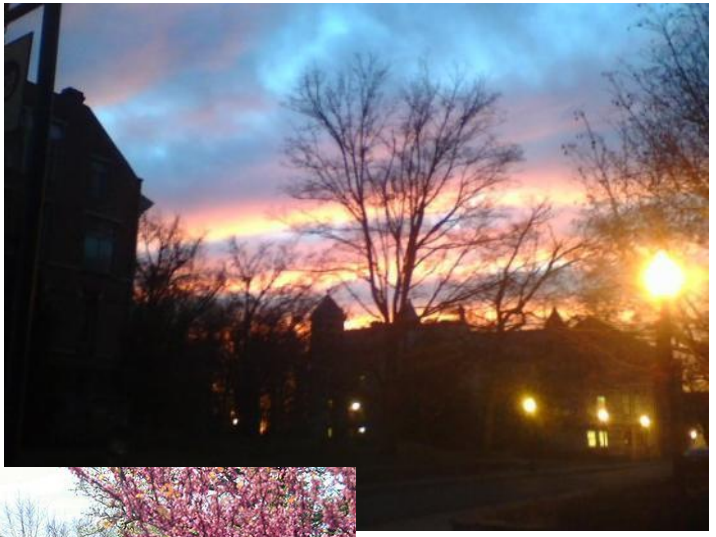
- Small block of fully associative tagged memory
- Basic logical and arithmetic unit
- Instruction register directs control to set data paths
- Nearest neighbor communications with switching





Workforce Development, Education, & Mentorship

- "Introduction to High Performance Computation"
- "Operating Systems"
- Graduate student research support
  - Faculty advisors
  - Substantial student desk spaces, computer laboratories
- Outreach
  - Conference tutorials (supported by UITS)
  - Textbook
    - "High Performance Computing - Modern Systems and Methods"
    - Publisher Morgan-Kaufmann – July, 2017
- ISE evolution
  - 2 Faculty
  - 3 Research Scientists
  - Planning
    - Curriculum, spaces, laboratories




CardCow.com