Invited Presentation to:
ECMWF Workshop 2010

Enabling Exascale Computing through the ParalleX Execution Model

Thomas Sterling

Arnaud & Edwards Professor, Department of Computer Science

Adjunct Professor, Department of Electrical and Computer Engineering

Faculty, Center for Computation and Technology

Louisiana State University

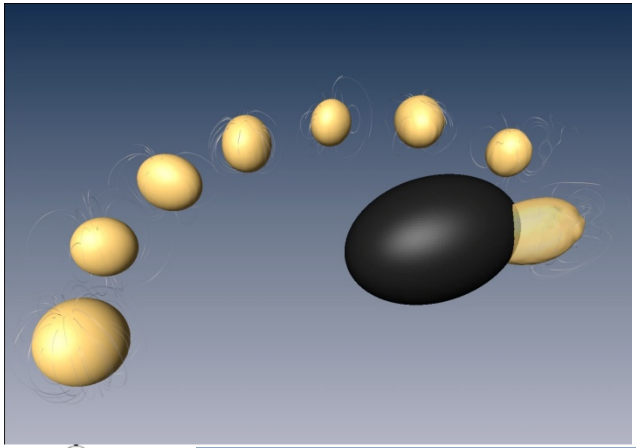
Distinguished Visiting Scientist, Oak Ridge National Laboratory

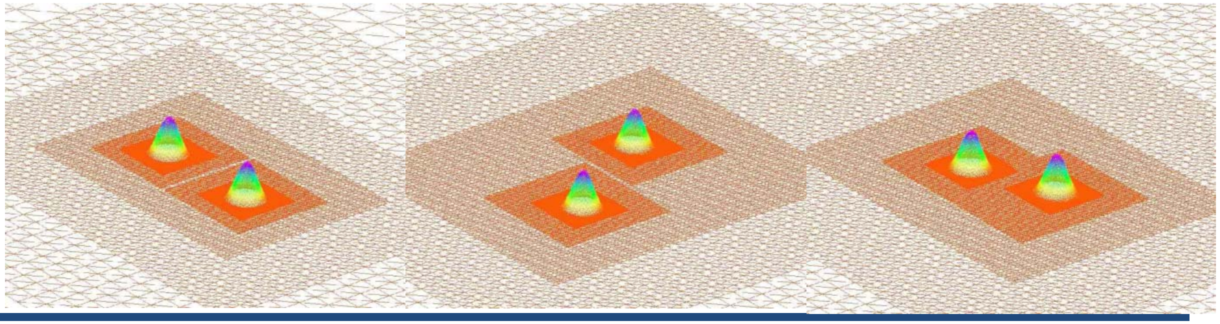
CSRI Fellow, Sandia National Laboratory

November 4, 2010

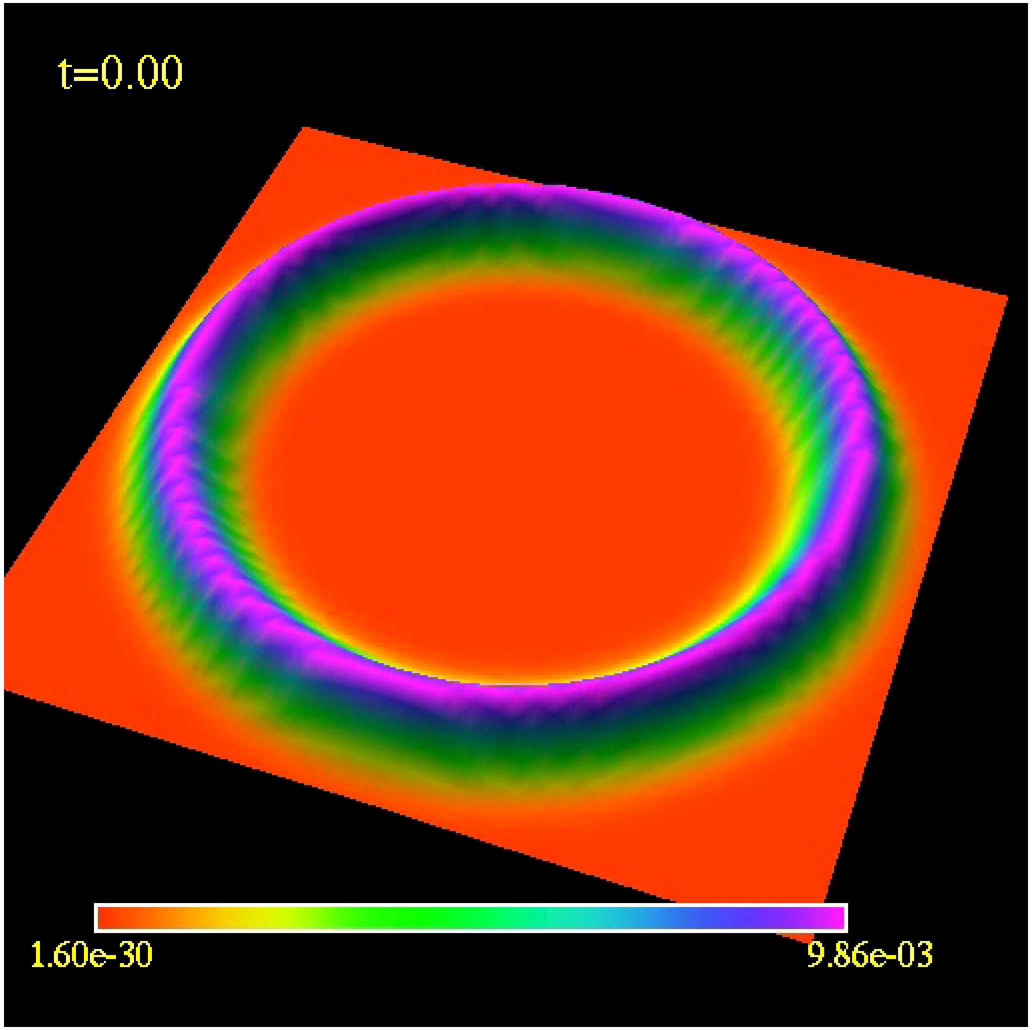
Application: Adaptive Mesh Refinement (AMR) for Astrophysics simulations

- Binary black hole and black hole neutron star mergers are LIGO candidates
- AMR simulations of black holes typically scale very poorly





Example: exploring critical collapse using Parallex based AMR with quad-precision.



Fastest Computer in the World

Dramatic Change in Technology Trends

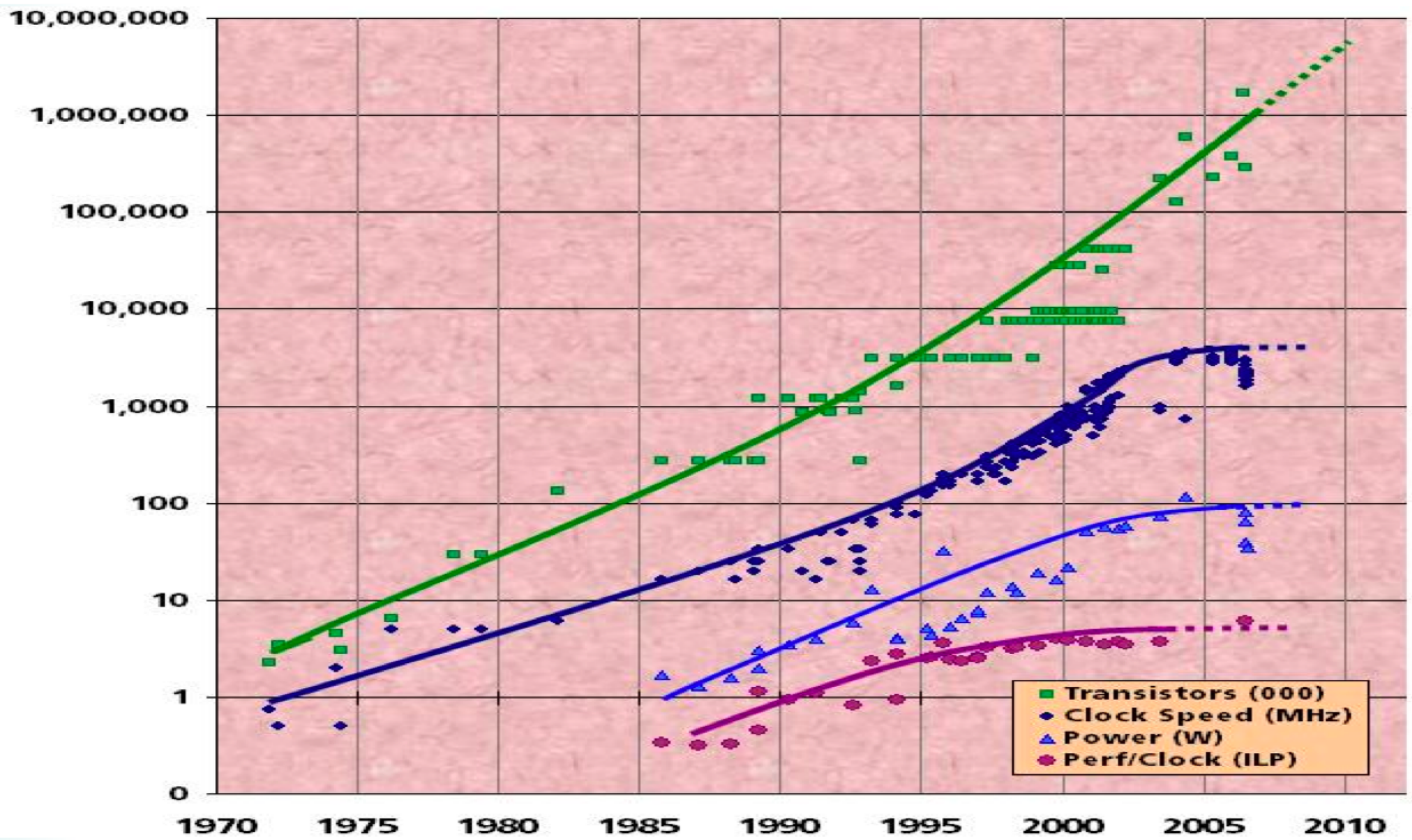
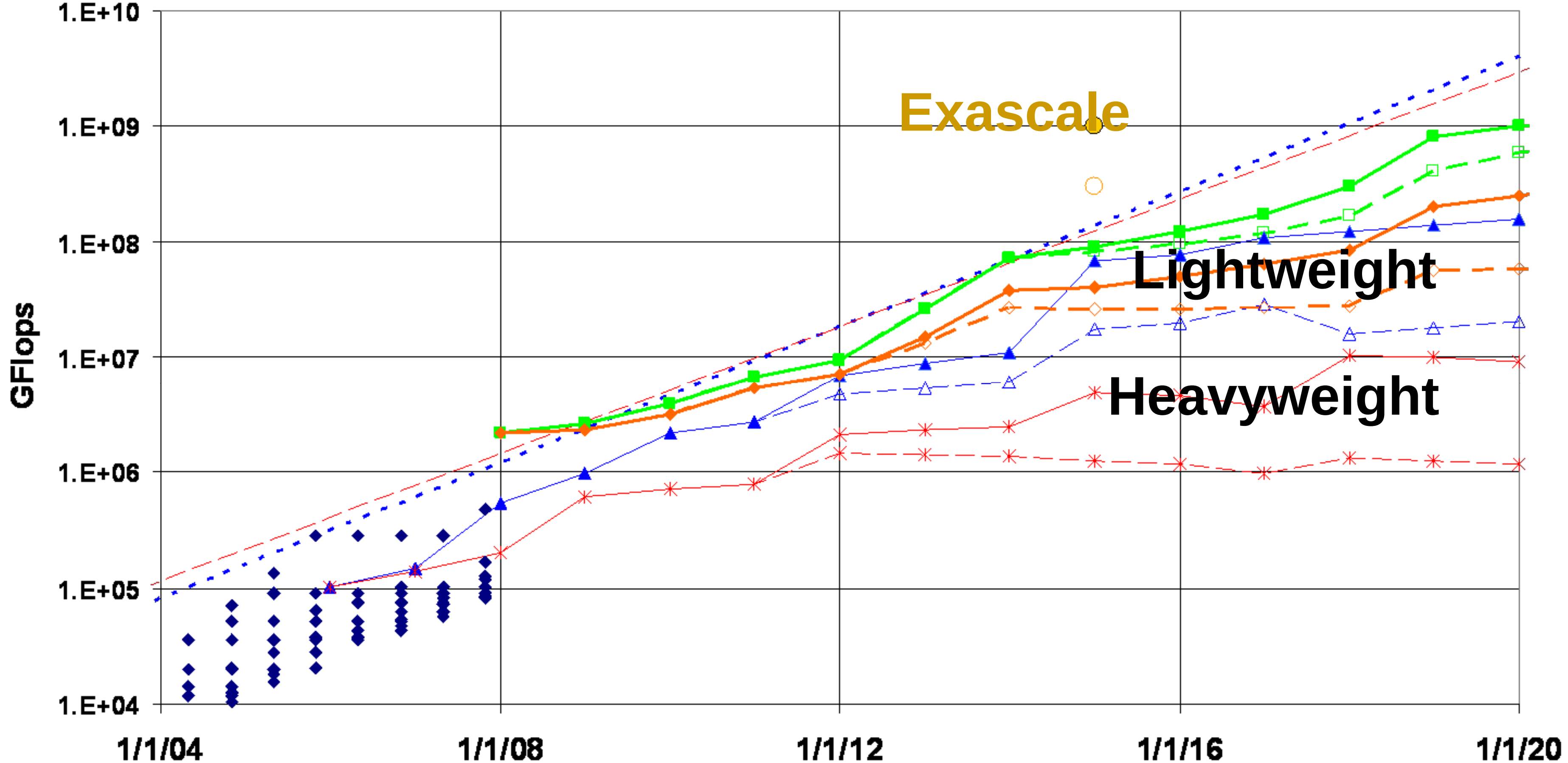


Figure courtesy of Kunle Olukotun, Lance Hammond, Herb Sutter, and Burton Smith

DARPA Exascale Technology Study



But not at 20 MW!

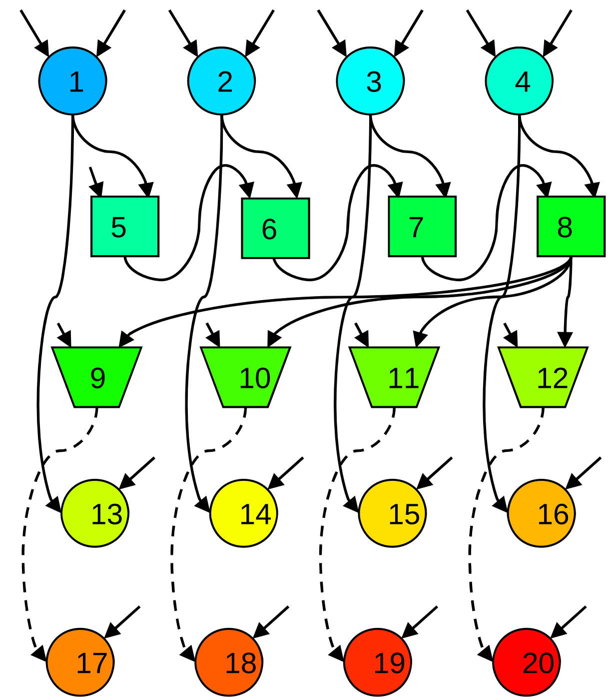
Courtesy of Peter Kogge, UND

StarSs: … taskified …

```
#pragma css task input(A, B) output(C)
void vadd3 (float A[BS], float B[BS],
            float C[BS]);
#pragma css task input(sum, A) inout(B)
void scale_add (float sum, float A[BS],
               float B[BS]);
#pragma css task input(A) inout(sum)
void accum (float A[BS], float *sum);
```

```
for (i=0; i<N; i+=BS)                // C=A+B
   vadd3 ( &A[i], &B[i], &C[i]);
...
for (i=0; i<N; i+=BS)               //
sum(C[i])
   accum (&C[i], &sum);
...
for (i=0; i<N; i+=BS)               // B=sum*A
   scale_add (sum, &E[i], &B[i]);
...
for (i=0; i<N; i+=BS)               // A=C+D
   vadd3 (&C[i], &D[i], &A[i]);
...
for (i=0; i<N; i+=BS)               // E=G+F
   vadd3 (&G[i], &F[i], &E[i]);
```

Compute dependences @ task instantiation time

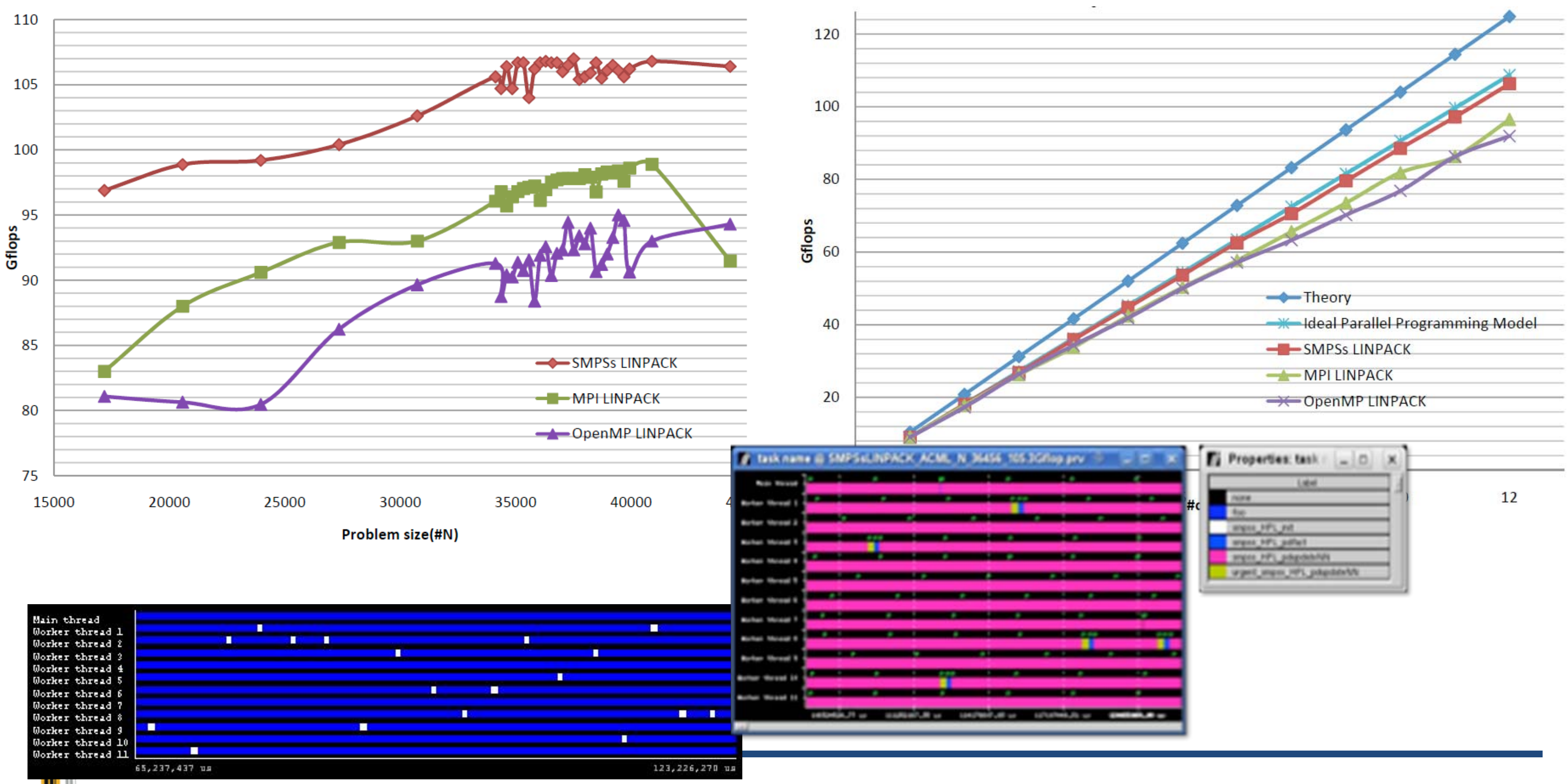


Color/number: order of task instantiation

Some antidependences covered by flow dependences not drawn

StarSs for SMP and multicores

- HPL Linpack: Comparison of SMPSs, OpenMP and MPI on a dual socket Istambul



Courtesy of Jesus Labarta, BSC

Runtime Solutions - Opportunities

- Adaptive scheduling
  - Load balancing
  - Contention avoidance, hot spots
- Lightweight mechanisms
  - Reduced overhead
- Finer granularity user threads
  - Increased concurrency for greater scalability
- Expanded synchronization semantics
  - Eliminate barriers, more intelligent control
- Runtime exploitation of Compile time programmer knowledge
  - Dedicated to specific application
- Adjusting to physical realities
  - Fault tolerance
  - Power management

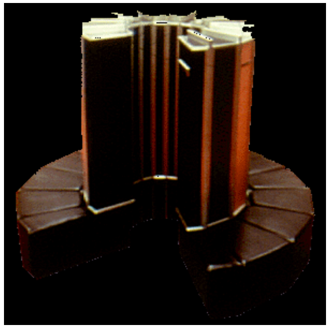
Performance Factors - SLOW

- Starvation
  - Insufficiency of parallelism
  - Either not enough work to do, or imbalance of workload
- Latency
  - Distance (in cycles) to remote resources
  - Avoid or hide
- Overhead
  - Critical path work required to manage tasks & resources
  - Imposes upper bound on scaling of fixed size workload
- Waiting for Contention
  - Delays incurred for shared access to resources
  - e.g., memory banks, network bandwidth, synchronization objects …

HPC in Phase Change

- Phase I: Sequential instruction execution (1950)
- Phase II: Sequential instruction issue (1965)

  - pipeline execution,
  - reservation stations,
  - ILP



- Phase III: Vector (1975)
  - pipelined arithmetic, registers, memory access
  - Cray

- Phase IV: SIMD (1985)
  - MasPar, CM-2
- Phase V: Communicating Sequential Processes (1990)
  - MPP, clusters
  - MPI, PVM

The Execution Model Imperative

- HPC in 6th Phase Change
  - Driven by technology opportunities and challenges
  - Historically, catalyzed by paradigm shift
- Guiding principles for governing system design and operation
  - Semantics, Mechanisms, Policies, Parameters, Metrics
- Enables holistic reasoning about concepts and tradeoffs
  - Serves for Exascale the role of *von Neumann architecture* for sequential
- Essential for co-design of all system layers
  - Architecture, runtime and operating system, programming models
  - Reduces design complexity from $O(N^2)$ to $O(N)$
- Empowers discrimination, commonality, portability
  - Establishes a phylum of UHPC class systems
- Decision chain
  - For reasoning towards optimization of design and operation

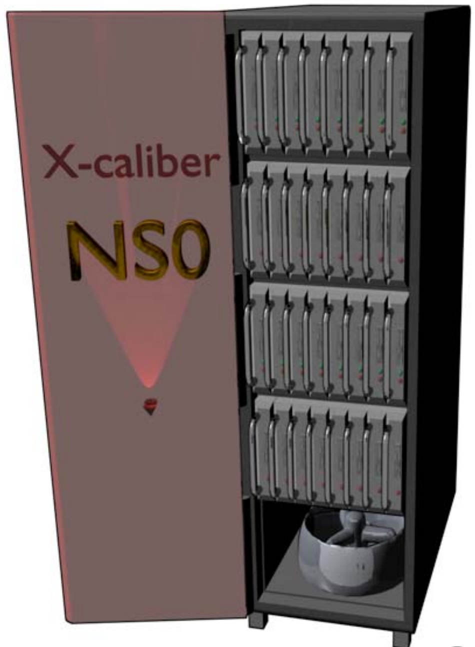
pgi0231 www.fotosearch.com

Decision Chain

- Axiom: an operation is performed at a certain place at a certain time to achieve a specified effect
- How did this happen?
- Every layer of the system contributed to the time/space/function event – the *decision chain*

- A program execution comprises the ensemble of such events across the system space and throughout the execution epoch
- There are many such paths that lead to a final result
- But not all minimize time and energy
- Understanding of the decision chain required for optimization
- Execution model required for understanding the decision chain

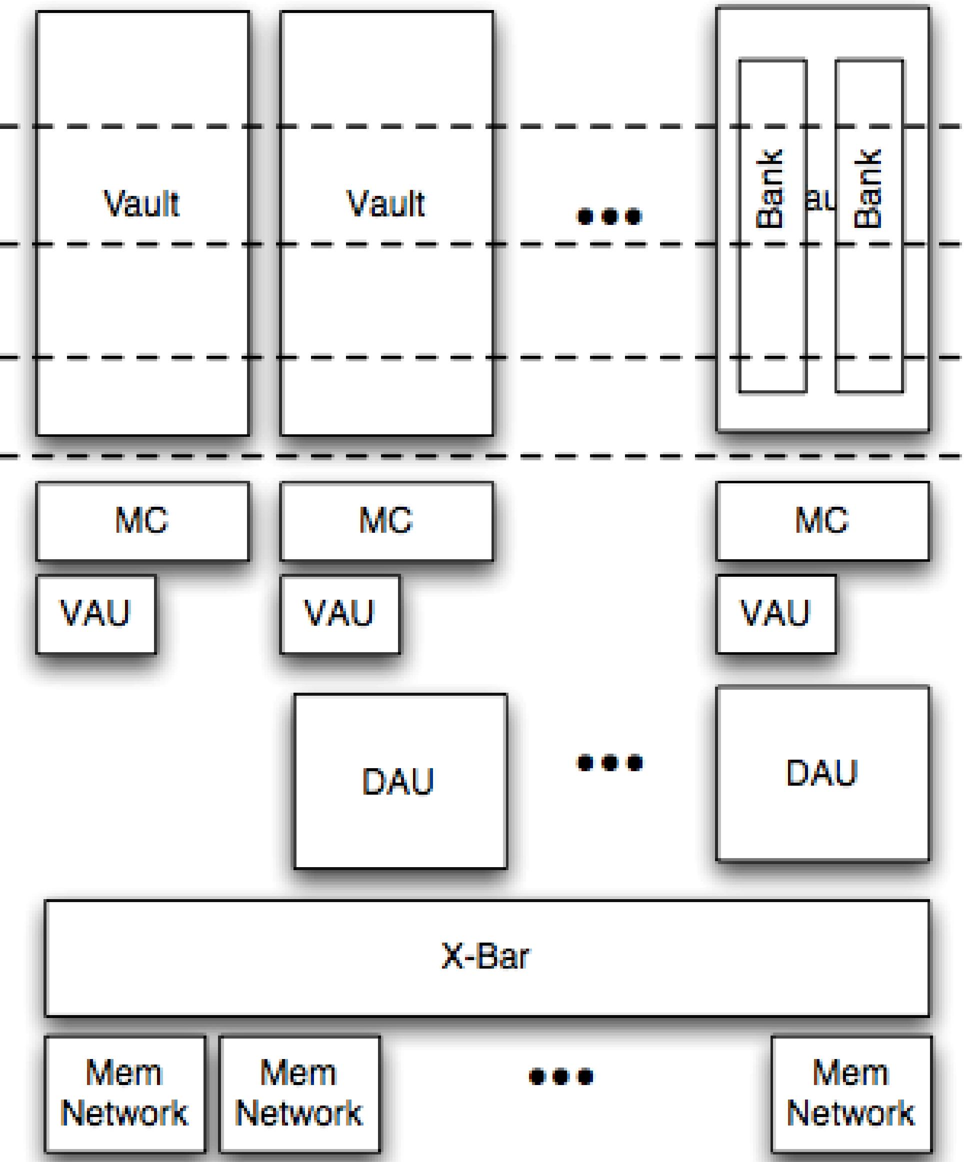
X-caliber System



- Rack Scale
  - Processing:128 Nodes, 1 (+) PF/s
  - Memory:
    - 128 TB DRAM
    - 0.4 PB/s Aggregate Bandwidth
  - NV Memory
    - 1 PB Phase Change Memory (addressable)
    - Additional 128 for Redundancy/RAID
  - Network
    - 0.13 PB/sec Injection, 0.06 PB/s Bisection

Deployment	Nodes	Topology	Compute	Mem BW	Injection BW	Bisection BW
Module	1	N/A	8 TF/s	3 TB/s	1 TB/s	N/A
Deployable Cage	22	All-to-All	176 TF/s	67.5 TB/s	22.5 TB/s	31 TB/s
Rack	128	Flat. Butterfly	1 PF/s	.4 PB/s	0.13 PB/s	0.066 PB/s
Group Cluster	512	Flat. Butterfly	4.1 PF/s	1.6 PB/s	0.52 PB/s	0.26 PB/s
National Resource	128k	Hier. All-to-All	1 EF/s	0.4 EB/s	0.13 EB/s	16.8 PB/s
Max Configuration	2048k	Hier. All-to-All	16 EF/s	6.4 EB/s	2.1 EB/s	0.26 EB/s

Memory System (M)



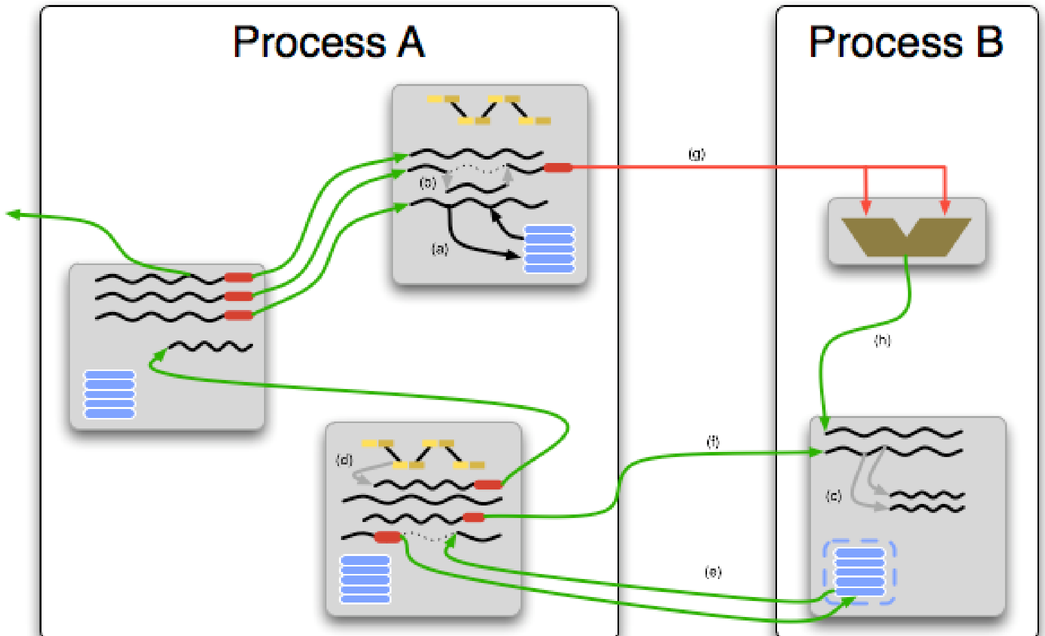
- Two computation Units
  - Right next to the DRAM vault memory controller (VAU)
  - To aggregate between DRAM vaults (DAU)
- "Memory Network" Centric
- Home-node for all addresses
  - Owns the "address"
  - Owns the "data"
  - Owns the "state" of the data
  - Can build "coherency"-like protocols via local operations
  - Can support PGAS-like operations
  - Can manage thread state locally

Sandia National Laboratories

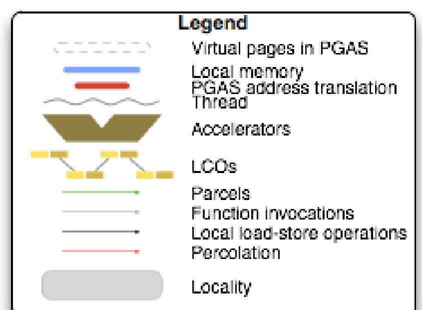
HPX Phase VI Parallel Execution Model

- Goals:
  - Guide Exascale system co-design for hardware, software, and programming
  - Dramatic gains in scalability, efficiency, and programmability
  - Framework for reliability, power management, security
  - Empower dynamic knowledge management and other graph-based problems
- Strategy:
  - Move work to data when appropriate; not always data to work
  - Dynamic adaptive resource and task management
  - work-queue split-phase transaction execution model for high utilization
  - Hierarchy name space for ease of data access with capabilities addressing for protection
- Constituent Components
  - Hierarchical Active Global Address Space, *AGAS*
  - *Parallel processes* spanning and overlapping multiple nodes
  - *Parcels* support message-driven computation and continuation migration
  - Local *computation complexes* (threads) with partial dataflow operations on private data
  - Local Control Objects, *LCO*, for lightweight synchronization and global parallel control state; includes dataflow and futures control
  - *Percolation* for efficient use of heterogeneous resources

ParalleX Model Components



(a) Local data access
(b) Local thread invocation (co-routine)
(c) Local thread invocation (concurrent threads)
(d) LCO spawning a thread
(e) Remote atomic memory operation through parcels
(f) Remote thread invocation through parcels
(g) Percolation
(h) Thread creation as result of continuation action



Multi-Grain Dataflow Multithreading: Computation Complexes (CC)

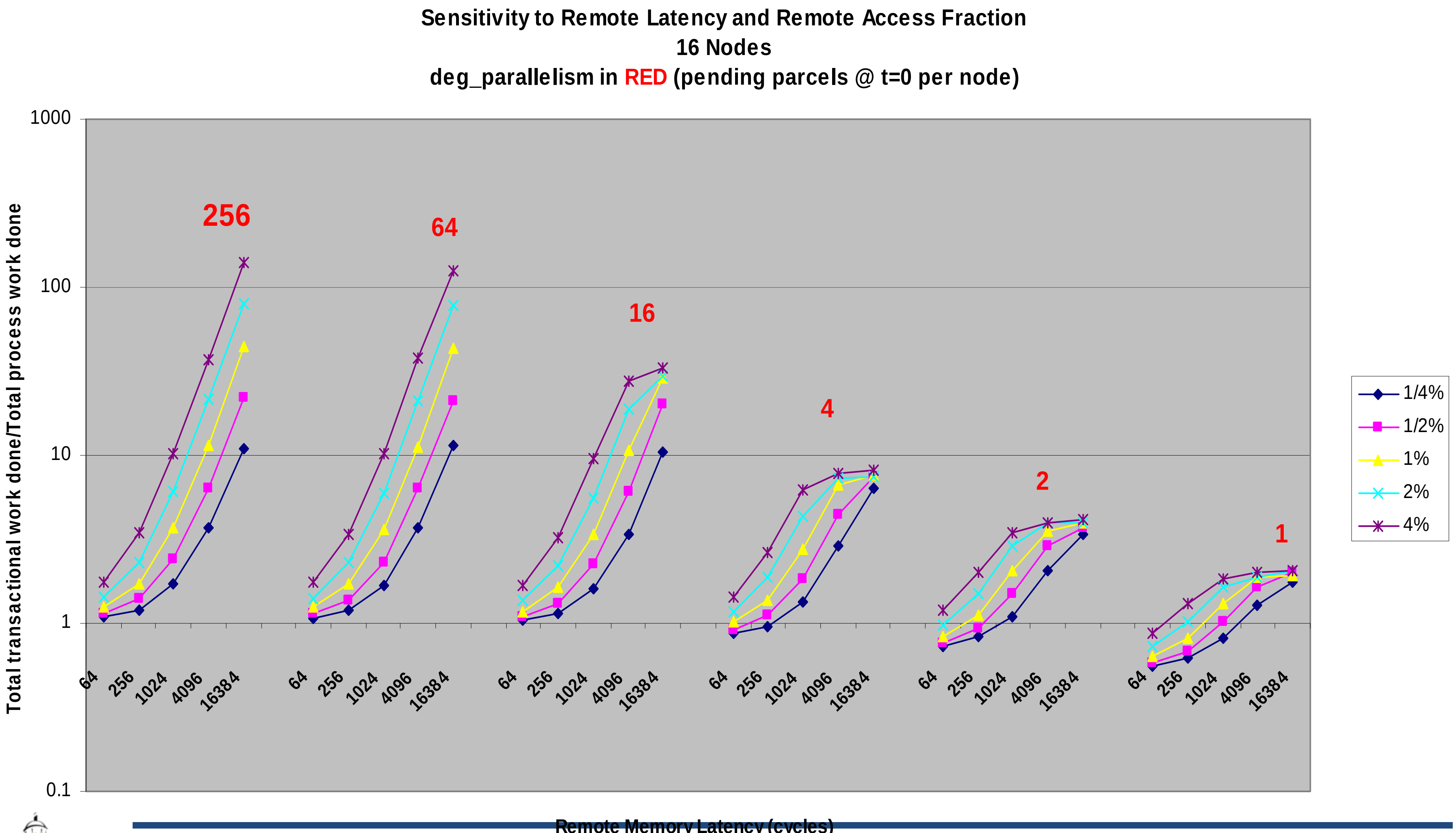
- Complexes are collections of related operations that perform on locally shared data
- Complex is a continuation combined with local environment
  - Modifies local named data state and temporaries
  - Updates intra-thread and inter-thread control state
- Does not assume sequential execution
  - Other flow control for intra-thread operations possible
- Complex can realize transaction phase
- Complex does not assume dedicated execution resources
- Complex is first class object identified in global name space
- Complex is ephemeral

Motivation for Message-Driven Computation

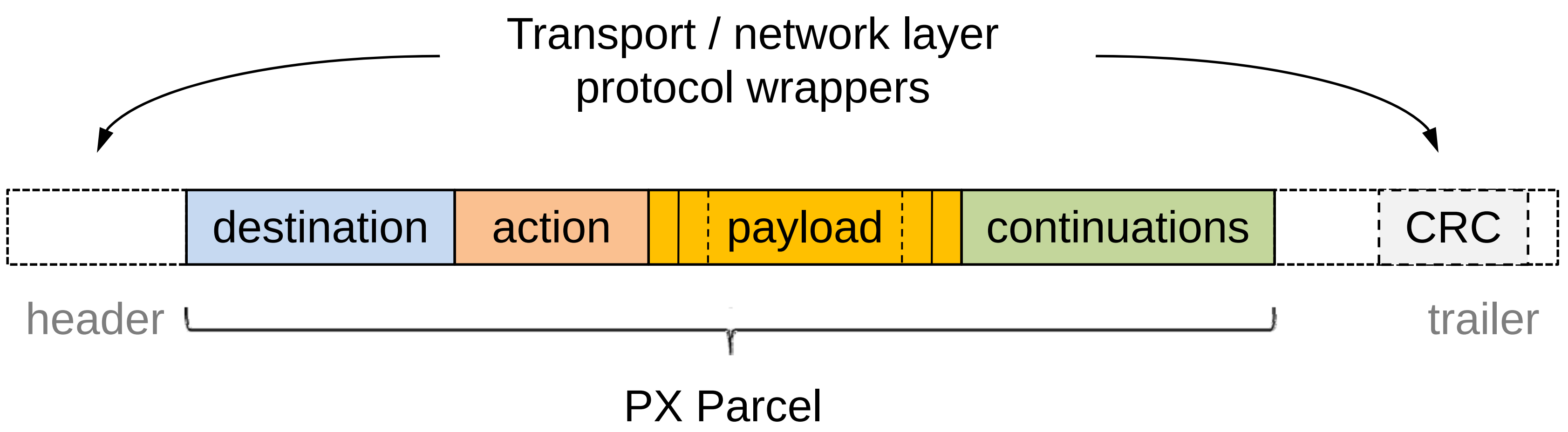
- To achieve high scalability, efficiency, programmability
- To enable new models of computation
  - e.g., ParalleX
- To facilitate conventional models of computation
  - e.g., MPI
- Hide latency
  - Support overlap of communication with computation
  - Move work to data, not always data to work
- Work-queue model of computing
  - Segregate physical resource from abstract task
  - Circumvent blocking of resource utilization
- Support asynchrony of operation
- Maintain symmetry of semantics between synchronous and asynchronous operation

Latency Hiding with Parcels

with respect to System Diameter in cycles



Parcel Structure

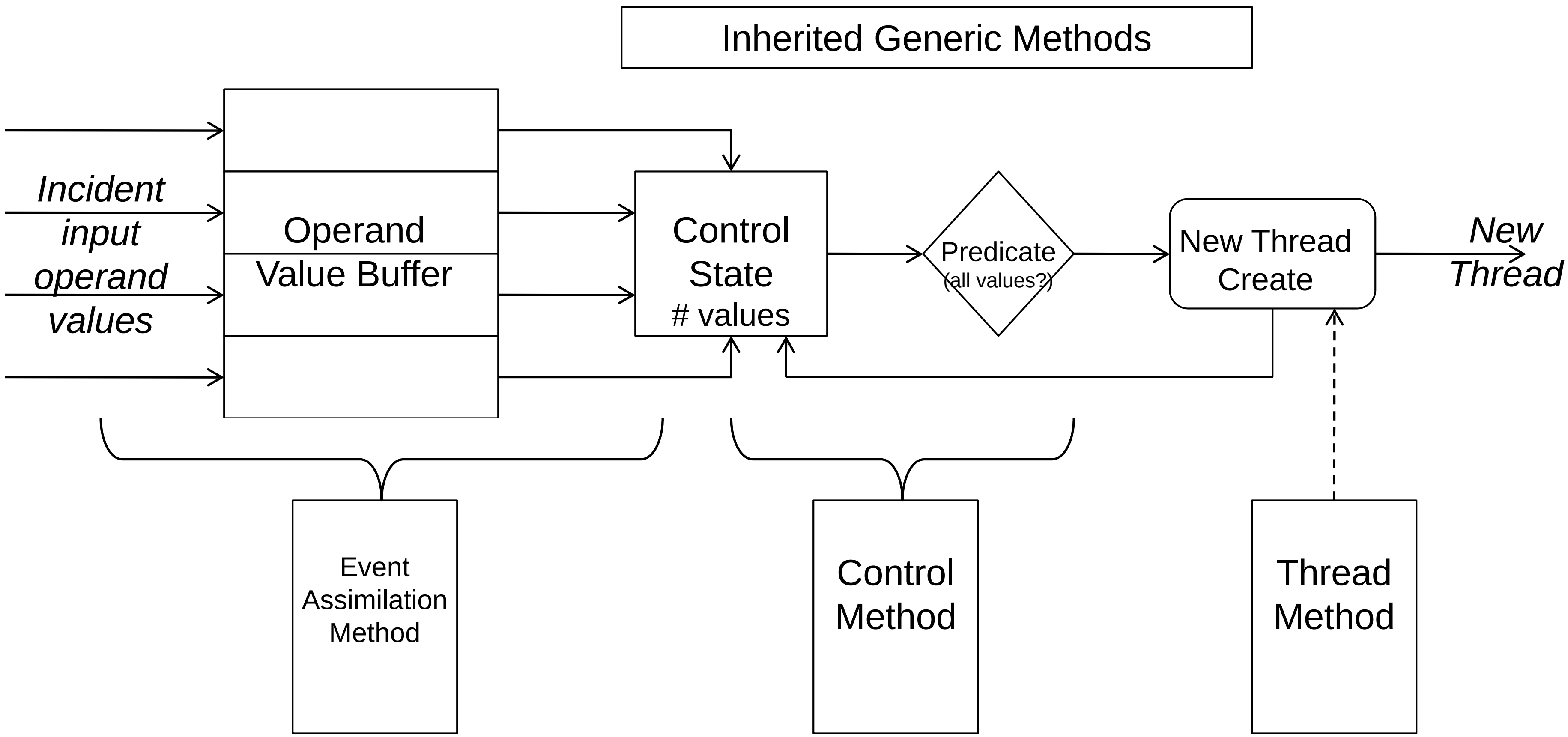


Parcels may utilize underlying communication protocol fields to minimize the message footprint (e.g. destination address, checksum)

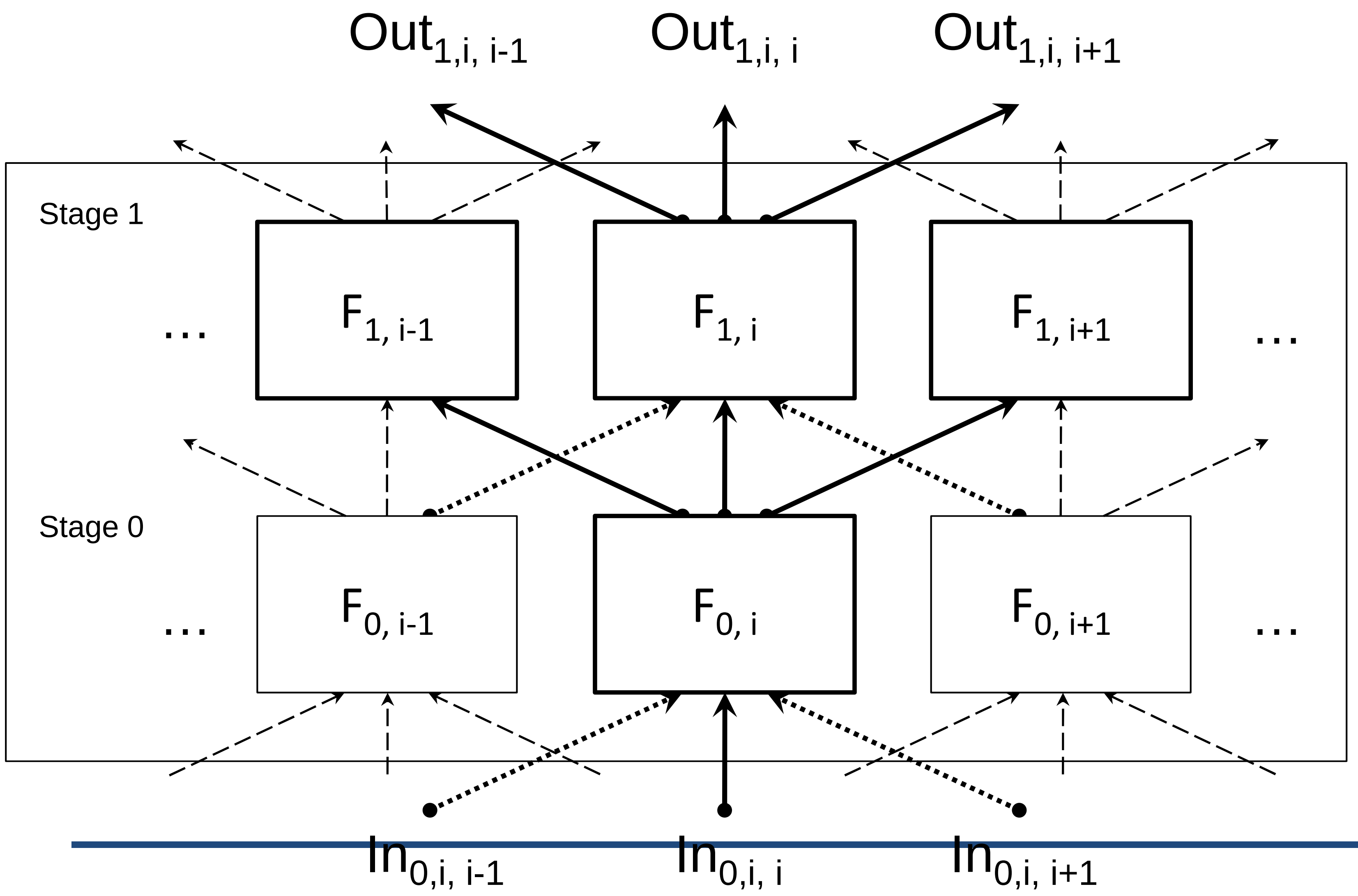
Local Control Objects

- A number of forms of synchronization are incorporated into the semantics
- Support message-driven remote thread instantiation
- Finite State Machines (FSM)
- In-memory synchronization
  - Control state is in the name space of the machine
  - Producer-consumer in memory
  - Local mutual exclusion protection
  - Synchronization mechanisms as well as state are presumed to be intrinsic to memory
- Basic synchronization objects:
  - Mutexes
  - Semaphores
  - Events
  - Full-Empty bits
  - Data flow
  - Futures
  - …
- User-defined (custom) LCOs

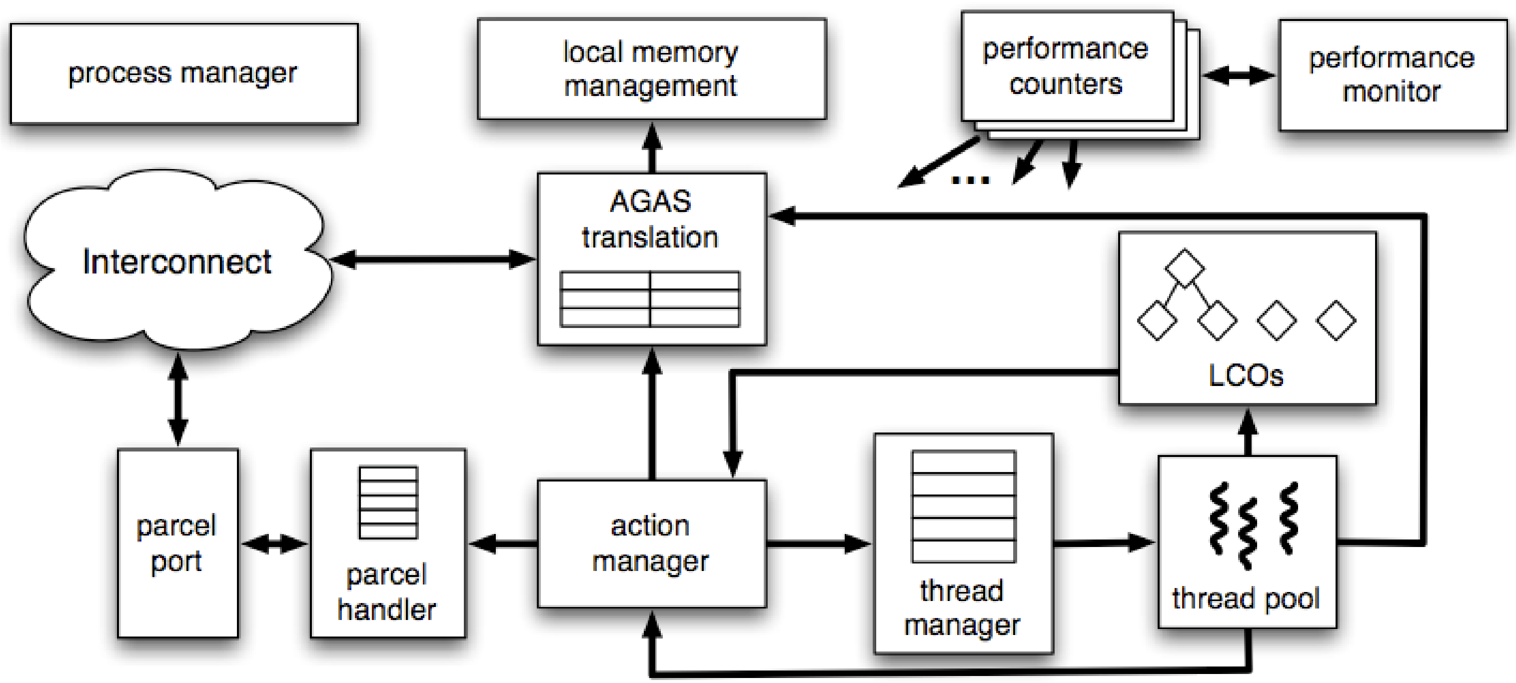
Dataflow LCO



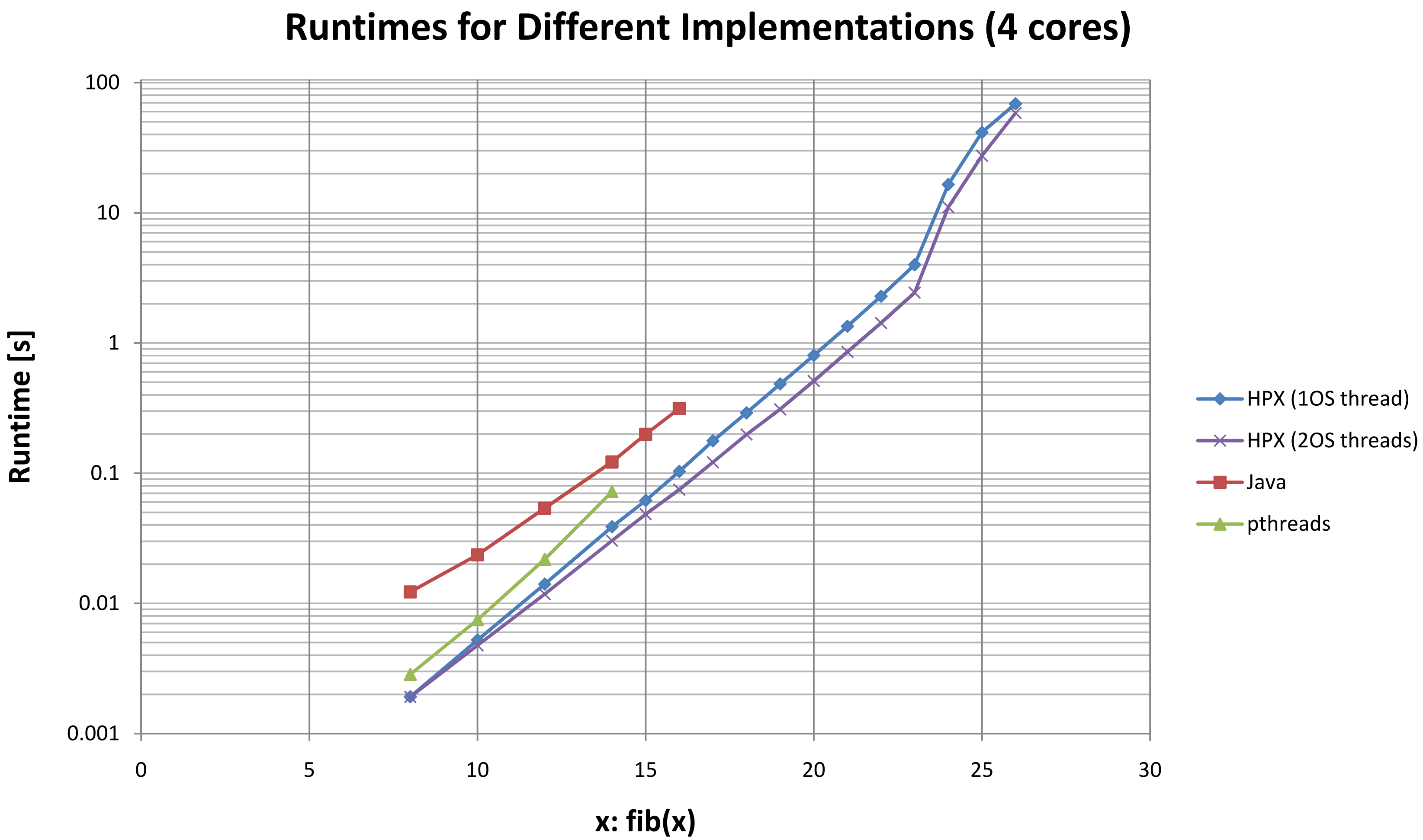
Using HPX for AMR



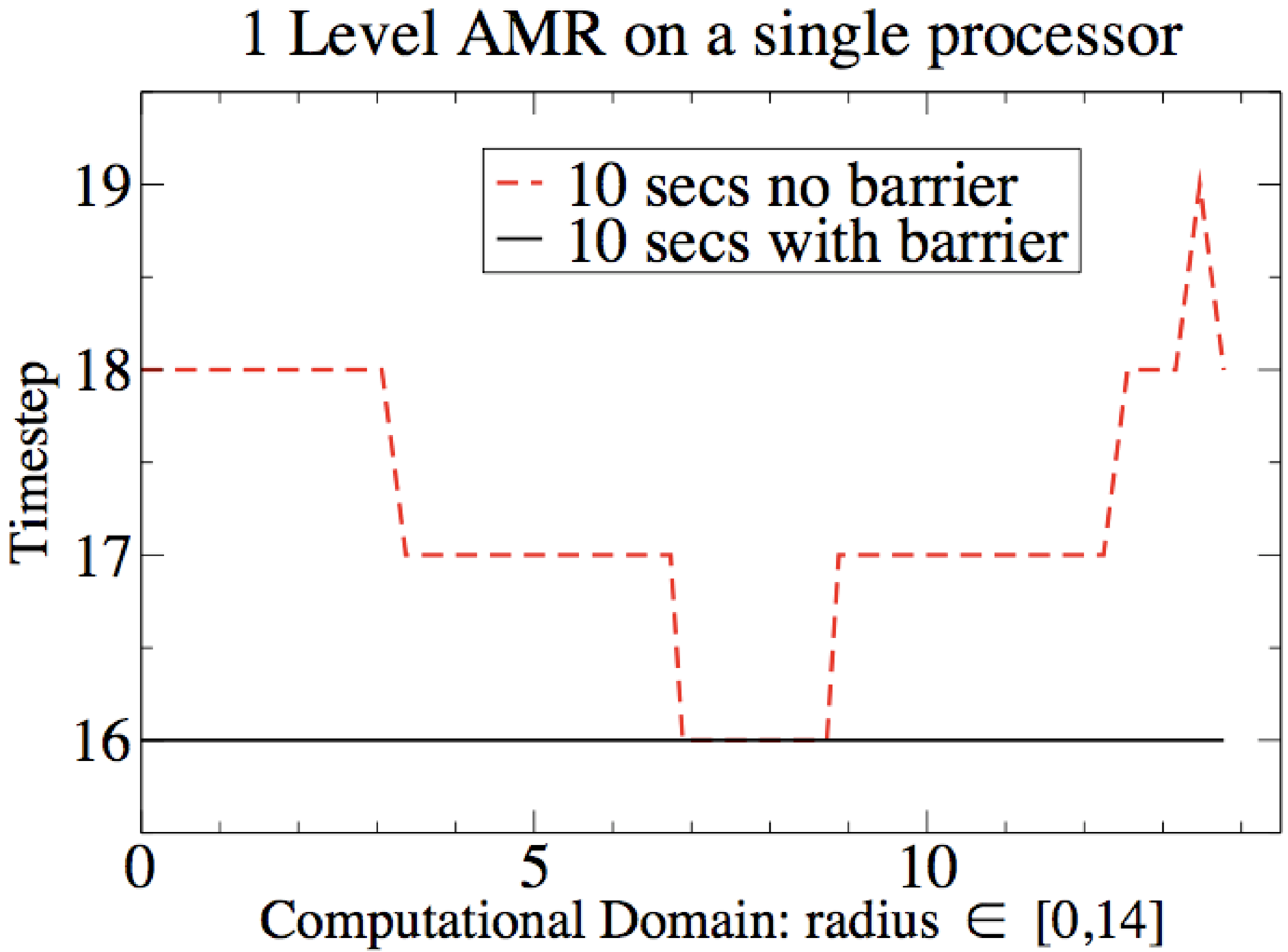
HPX Runtime System



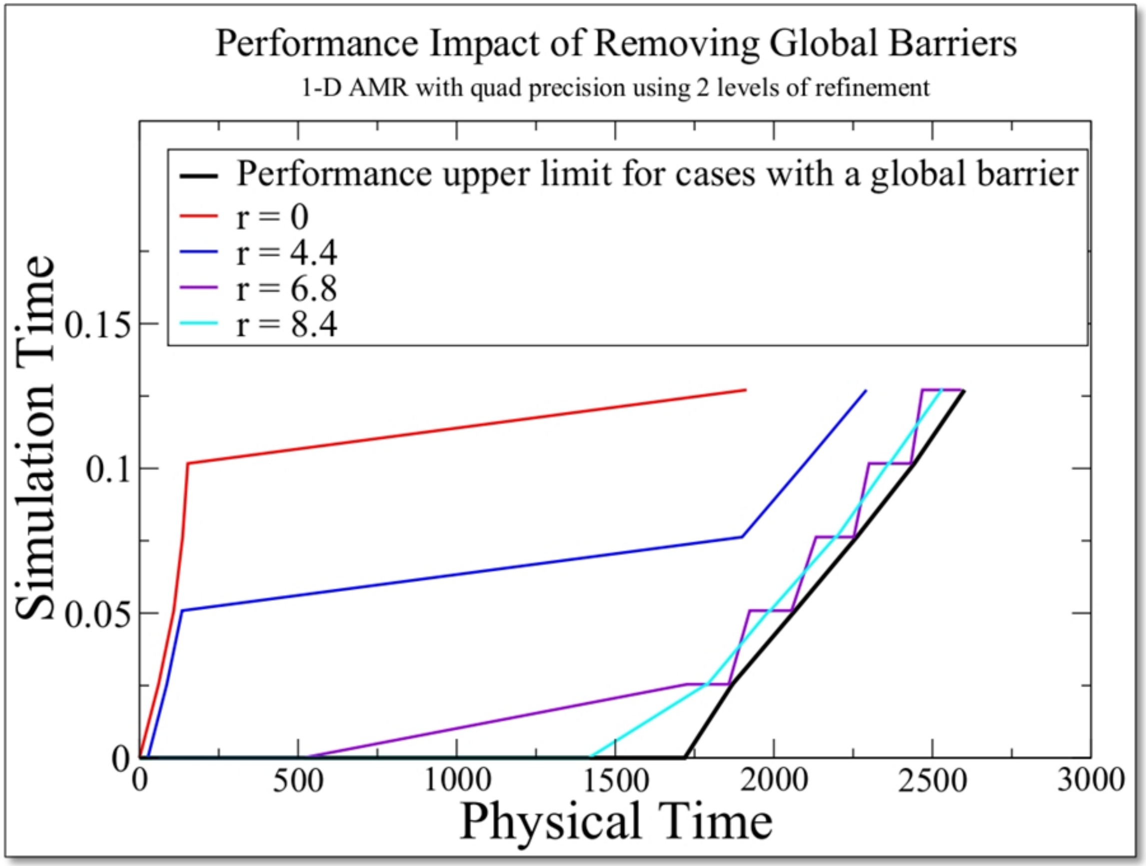
Fibonacci Sequence



Using HPX for Variable Threads


1 Level AMR on a single processor
10 secs no barrier
10 secs with barrier
Timestep
16
17
18
19
0
5
10
Computational Domain: radius ∈ [0,14]

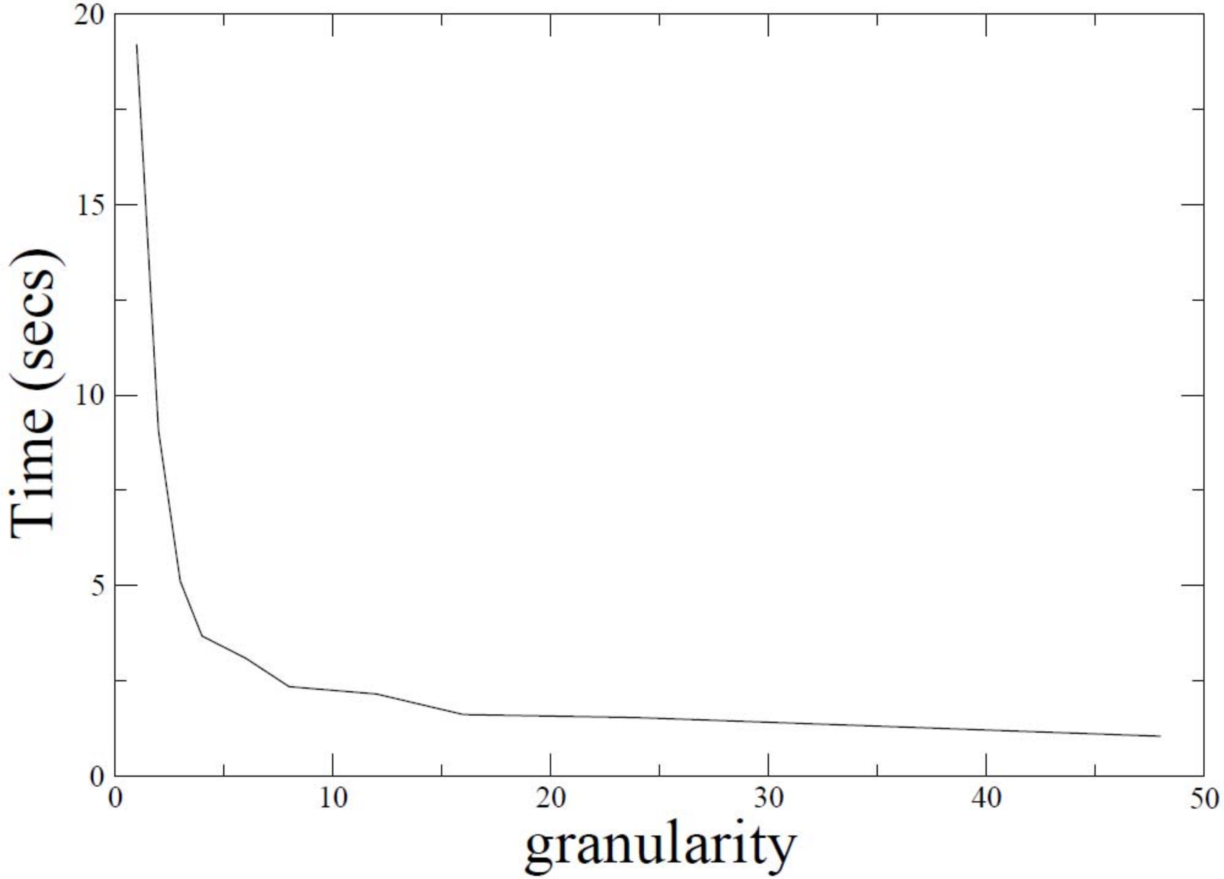
Application: Adaptive Mesh Refinement (AMR) for Astrophysics simulations



- ParalleX based AMR removes all global computation barriers, including the timestep barrier (so not all points have to reach the same timestep in order to proceed computing)

LSU
CENTER FOR COMPUTATION & TECHNOLOGY

AMR Granularity



Conclusions

- The future of HPC demands innovative response to technology challenges and application opportunities
- HPC is entering Phase VI requiring a new model of computation
  - Attack starvation, latency, overhead, & waiting for contention (SLOW)
  - Dynamic adaptive resource management & task scheduling
  - Dynamic graph-based applications for knowledge management (AI)
- ParalleX represents an experimental step
  - Dynamic, overlap/multiphase message-driven execution
- Large scale runtime experiments required to guide progress
  - Application driven
  - Stimulate work in Architecture and Programming Models
  - ParalleX provides an experimental model with HPX reference implementation