



Presentation to the NASA Goddard Space Flight Center

The New Cluster Paradigm for Exascale Computing

Thomas Sterling

Professor of Informatics and Computing, Indiana University

Associate Director and Chief Scientist,

Center for Research in Extreme Scale Technologies

Adjunct Professor, Louisiana State University

CSRI Fellow, Sandia National Laboratory

November 30, 2011



Cluster 2004

The Cluster Agenda: First Achieve World Domination,
then Kick Ass

Dr. Thomas Sterling

Faculty Associate, California Institute of Technology
Principal Scientist, Jet Propulsion Laboratory





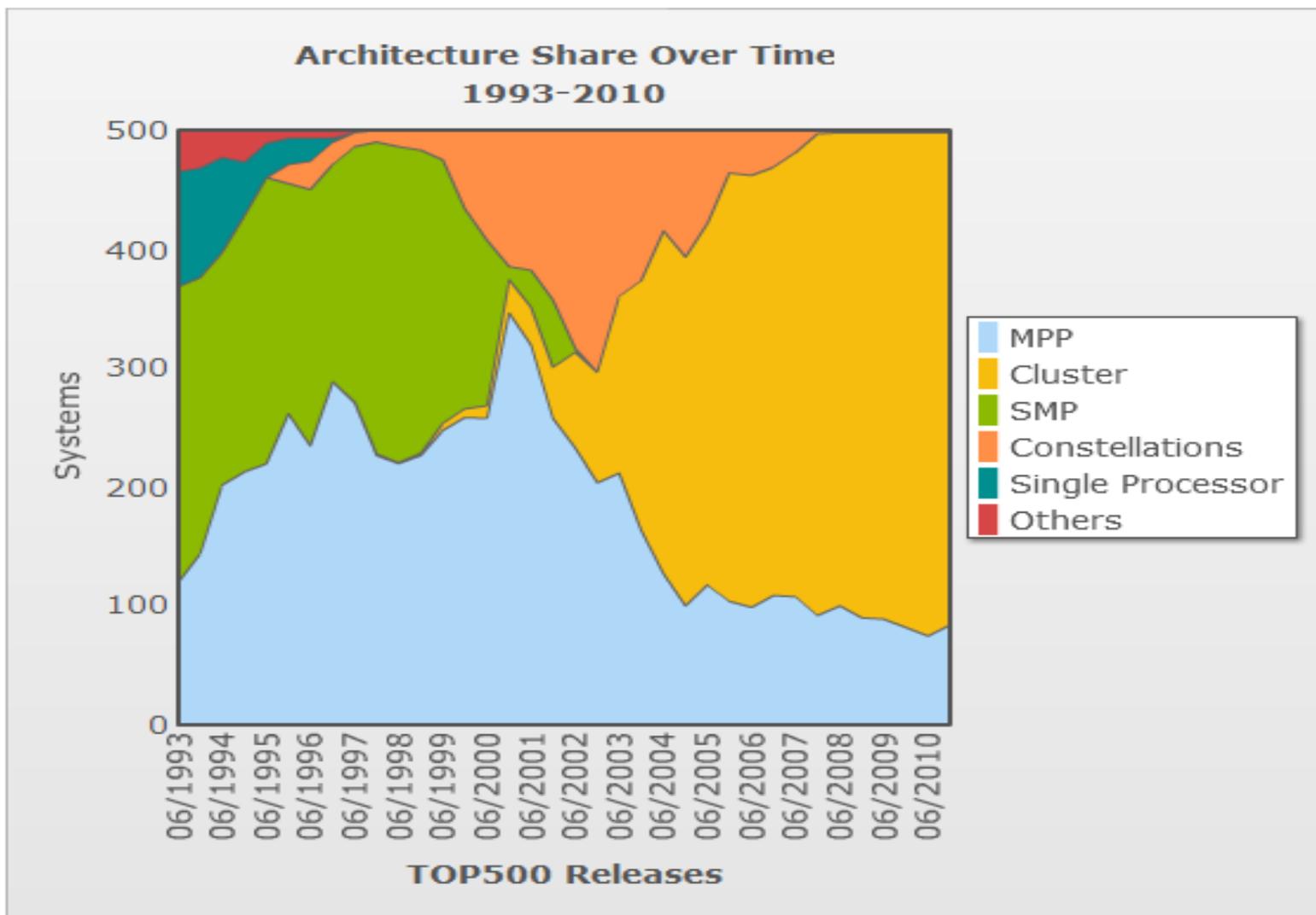
Petaflops Clusters

- #4 Nebulae
 - 1.27 Pflops
 - Dawning
 - China
- #5 Tsubame 2.0
 - 1.19 Pflops
 - NEC/HP
 - Japan
- #7 Pleiadies
 - 1.09 Pflops
 - SGI
 - USA
- #9 Tera-100
 - 1.05 Pflops
 - Bull SA
 - France





Clusters – part of a Paradigm Shift



Beowulf Project



- | | | |
|---------------------------|----------------------------------|-------------------------------------|
| ◆ Wiglaf – 1994 (GSFC) | ◆ Hrothgar – 1995 (GSFC) | ◆ Hyglac-1996 (Caltech) |
| ◆ 16 Intel 80486 100 MHz | ◆ 16 Intel Pentium 100 MHz | ◆ 16 Pentium Pro 200 MHz |
| ◆ VESA Local bus | ◆ PCI | ◆ PCI |
| ◆ 256 Mbytes memory | ◆ 1 Gbyte memory | ◆ 2 Gbytes memory |
| ◆ 6.4 Gbytes of disk | ◆ 6.4 Gbytes of disk | ◆ 49.6 Gbytes of disk |
| ◆ Dual 10 base-T Ethernet | ◆ 100 base-T Fast Ethernet (hub) | ◆ 100 base-T Fast Ethernet (switch) |
| ◆ 72 Mflops sustained | ◆ 240 Mflops sustained | ◆ 1.25 Gflops sustained |
| ◆ \$40K | ◆ \$46K | ◆ \$50K |

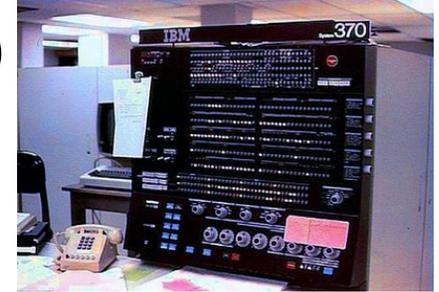
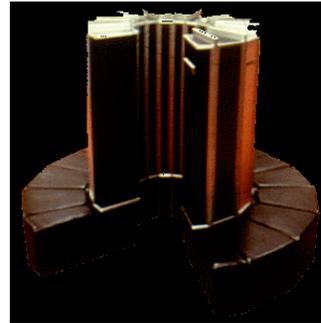
HPC in Phase Change



- Phase I: Sequential instruction execution (1950)

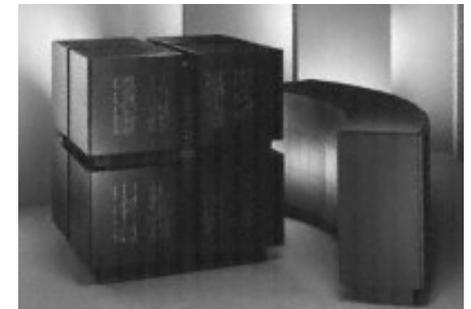
- Phase II: Sequential instruction issue (1965)

- 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)

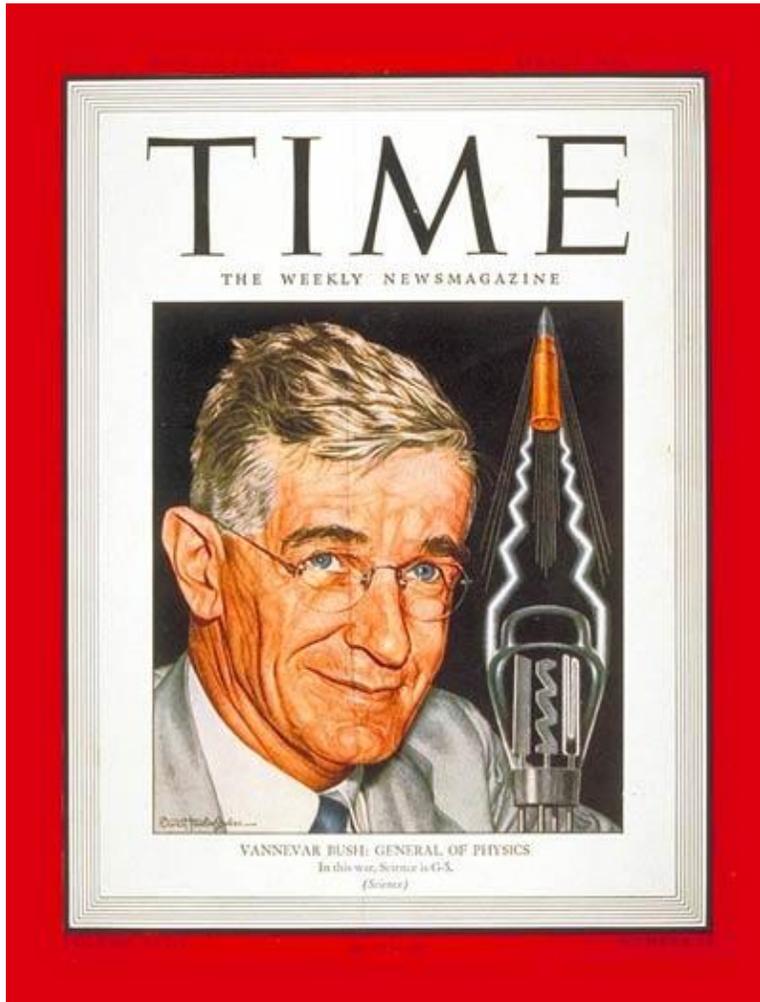
- MPP, clusters
- MPI, PVM



The first Supercomputer?

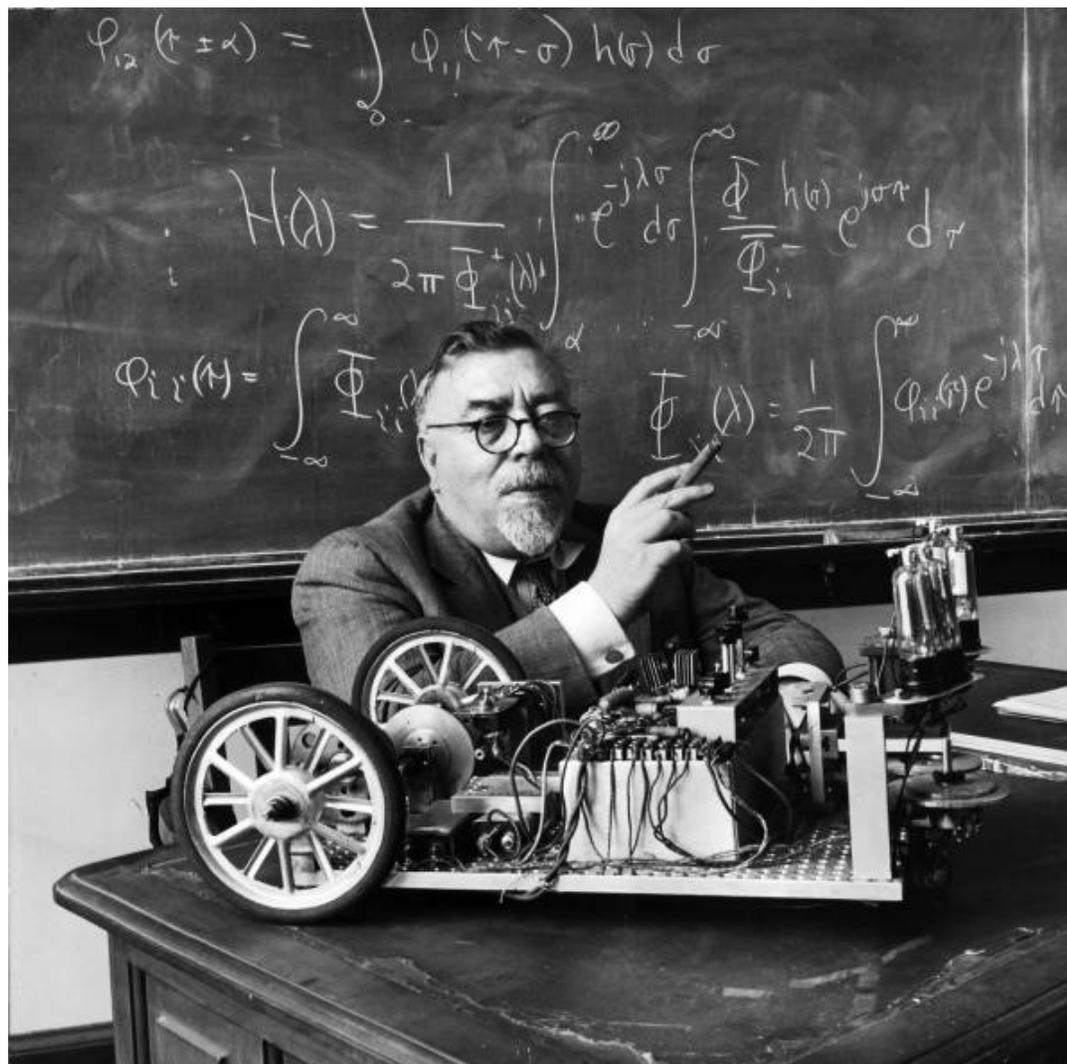
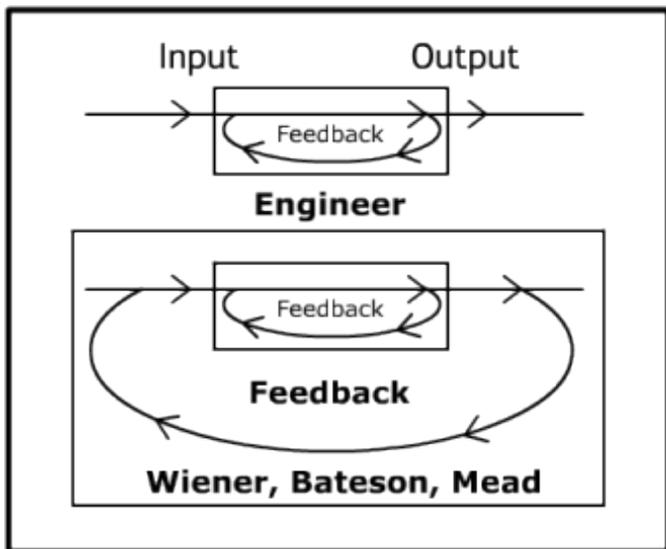


Computing in Paradigm Shift





The Essence of Computing – “Cyber”



Cybernetics in Action

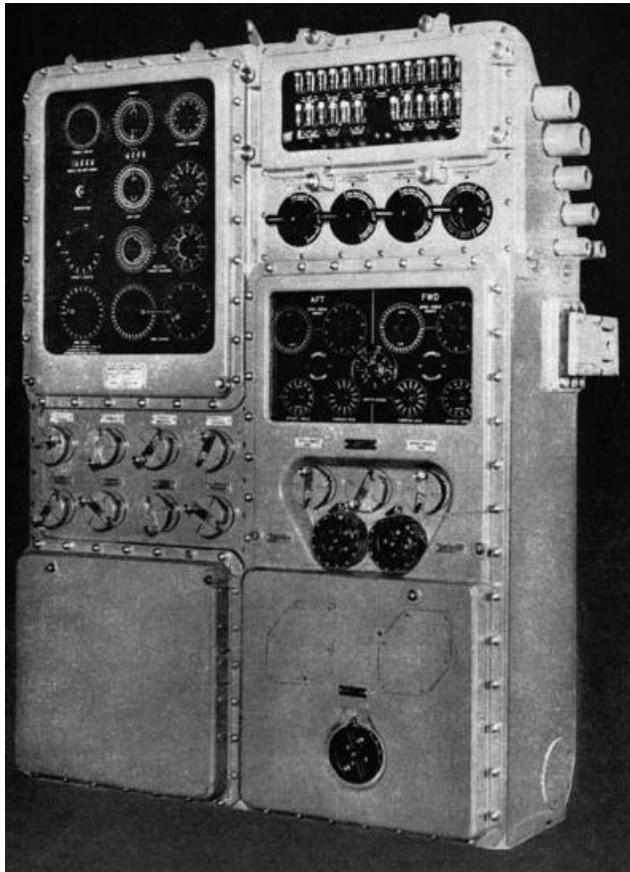


Photo # 80-G-441562 USS Pittsburgh anchored in Suda Bay, Crete, 8 May 1952



World's 1st Cluster



**DEPARTMENT OF COMPUTER SCIENCE @
LOUISIANA STATE UNIVERSITY**

Technology Demands new Response

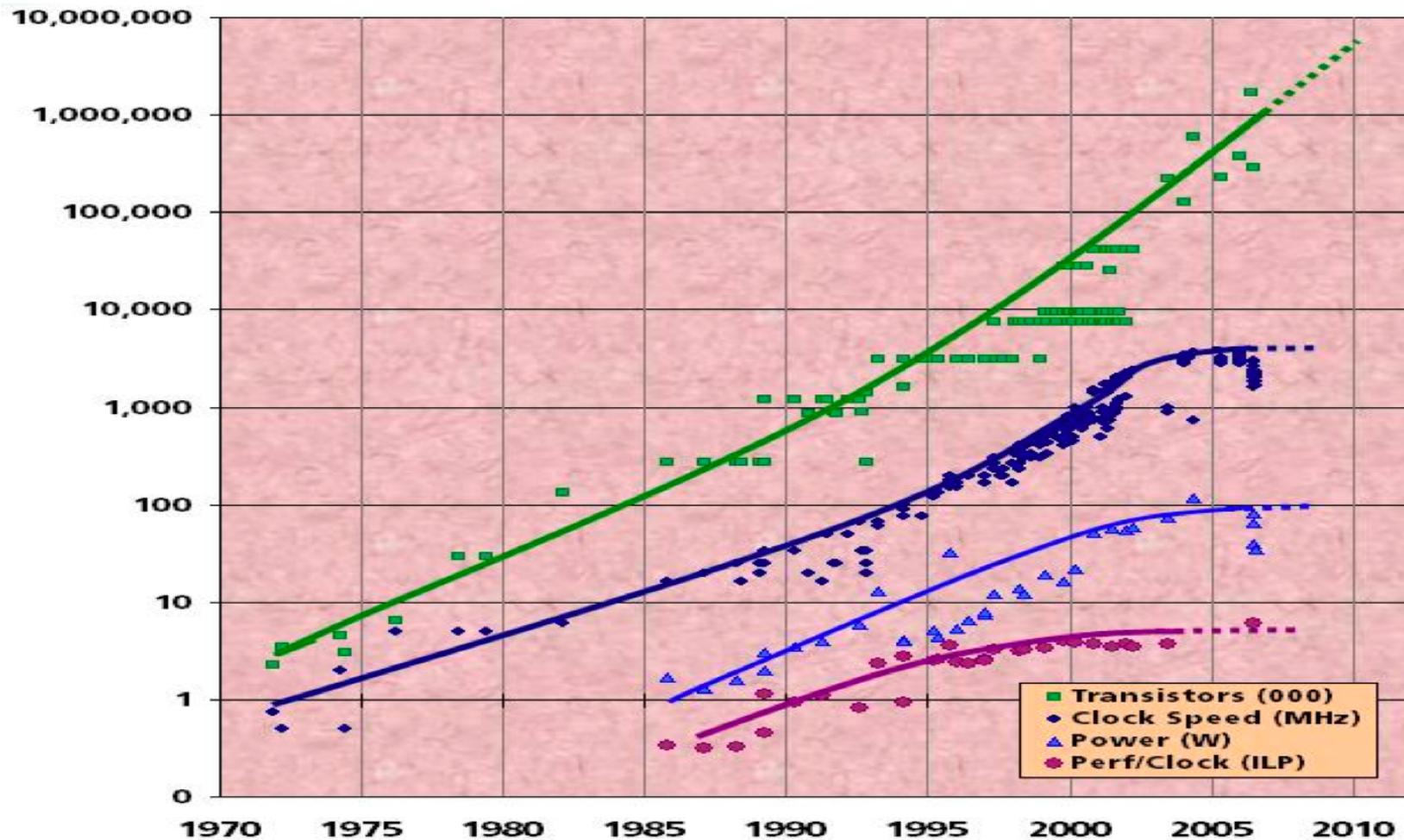


Figure courtesy of Kunle Olukotun, Lance Hammond, Herb Sutter, and Burton Smith



Strategic Requirements

- Performance
 - Efficiency
 - Scalability
 - Energy
 - Bounded power
 - Minimized energy
 - Reliability
 - Continued operation in the presence of faults
 - Programmability
 - System transparency
 - Portability across system classes, scales, and generations
 - Generality
 - STEM
 - Knowledge management and understanding
-



Tactical Performance Requirements



- 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

The Execution Model Imperative



- HPC in the 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



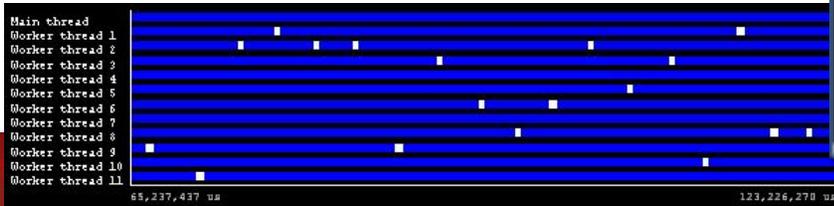
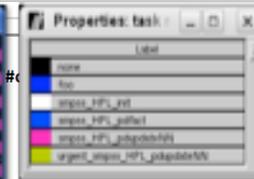
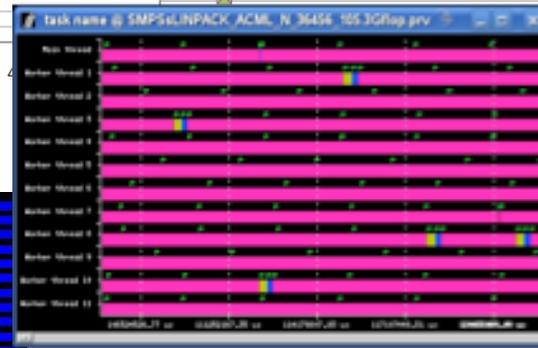
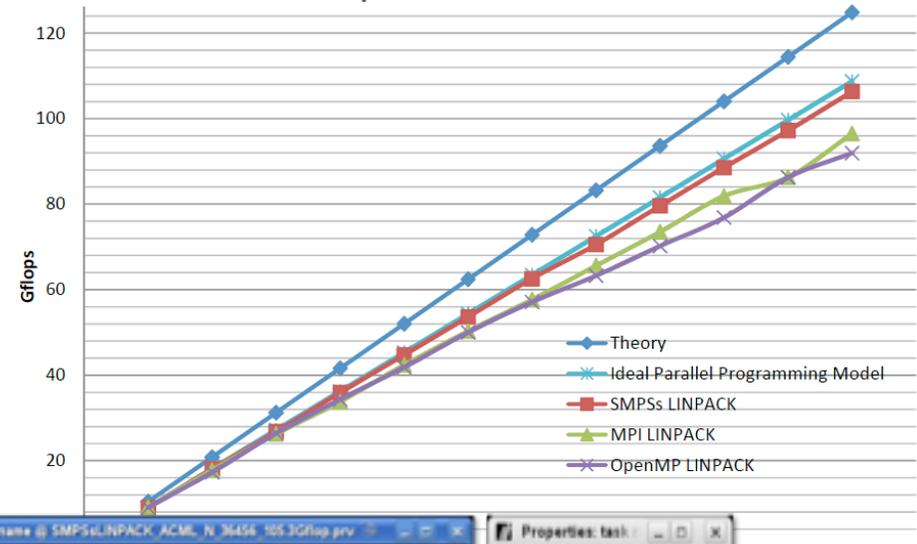
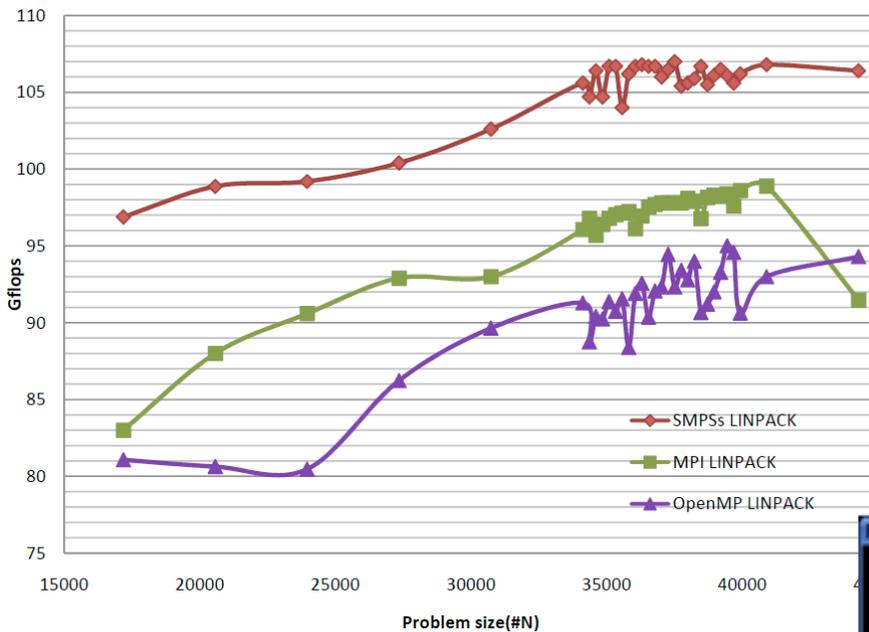
pgji0231 www.fotosearch.com



StarSs for SMP and multicores



- HPL Linpack: Comparison of SMPs, OpenMP and MPI on a dual socket Istanbul



Courtesy of Jesus Labarta, BSC



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);
```

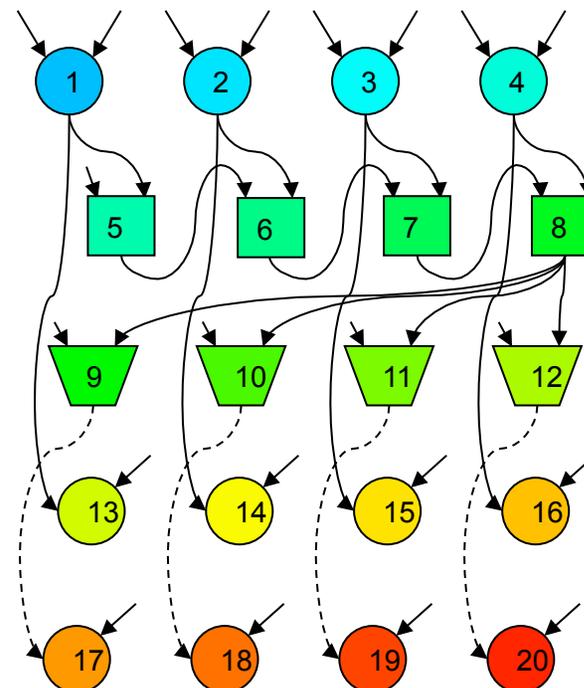


Compute dependences @ task instantiation time

```

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]);

```



Color/number: order of task instantiation

Some antidependencies covered by flow dependences not drawn



Courtesy of Jesus Labarta, BSC

Game Changer – Runtime System



- Runtime system
 - is: ephemeral, dedicated to and exists only with an application
 - is not: the OS, persistent and dedicated to the hardware system
- Moves us from *static* to *dynamic* operational regime
 - Exploits situational awareness for causality-driven adaptation
 - Guided-missile with continuous course correction rather than a fired projectile with fixed-trajectory
- Based on foundational assumption
 - Untapped system resources to be harvested
 - More computational work will yield reduced time and lower power
 - Opportunities for enhanced efficiencies discovered only in flight
 - New methods of control to deliver superior scalability
- “Undiscovered Country” – adding a dimension of systematics
 - Adding a new component to the system stack
 - Path-finding through the new trade-off space

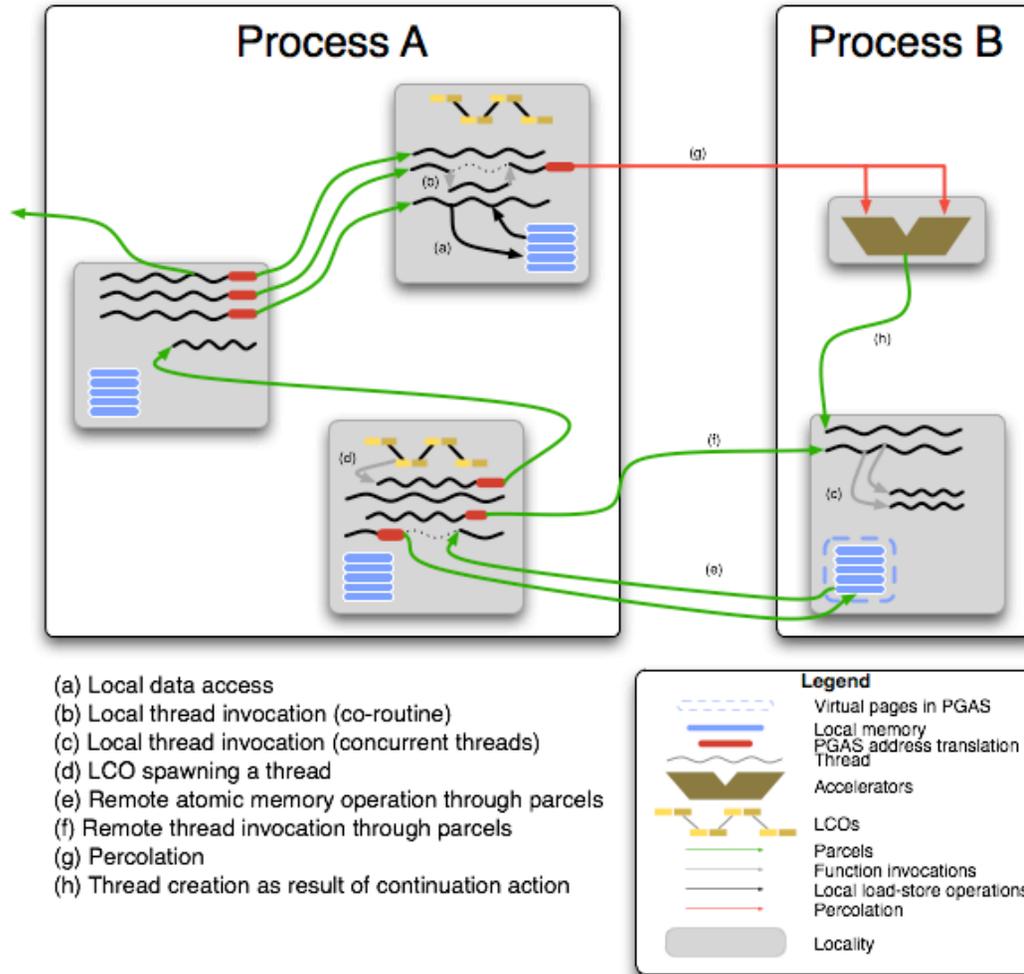


Concepts towards a new Paradigm



- Motivated by dynamic directed graphs
 - STEM
 - Knowledge management and understanding
- Split-phase transactions
 - Avoid blocking
- Message-driven computation
 - Move work to data
 - Parcels and Percolation
- Constraint-based synchronization
 - Declarative criteria for work
 - Event driven
- Data-directed execution
 - Merger of flow control and data structure
- Shared name space

ParalleX Execution Model Components



A Quick ParalleX Review



1. Synchronous Domains
2. AGAS – Active Global Address Space
3. ParalleX Processes – with capabilities protection
4. Computational Complexes – threads & fine grain dataflow
5. Local Control Objects – synchronization and global distributed control state
6. Distributed control operation – global mutable data structures
7. Parcels – message-driven execution and continuation migration
8. Percolation – heterogeneous control
9. Micro-checkpointing – compute-validate-commit
10. Self-aware – introspection and declarative management

Constraint-based Synchronization

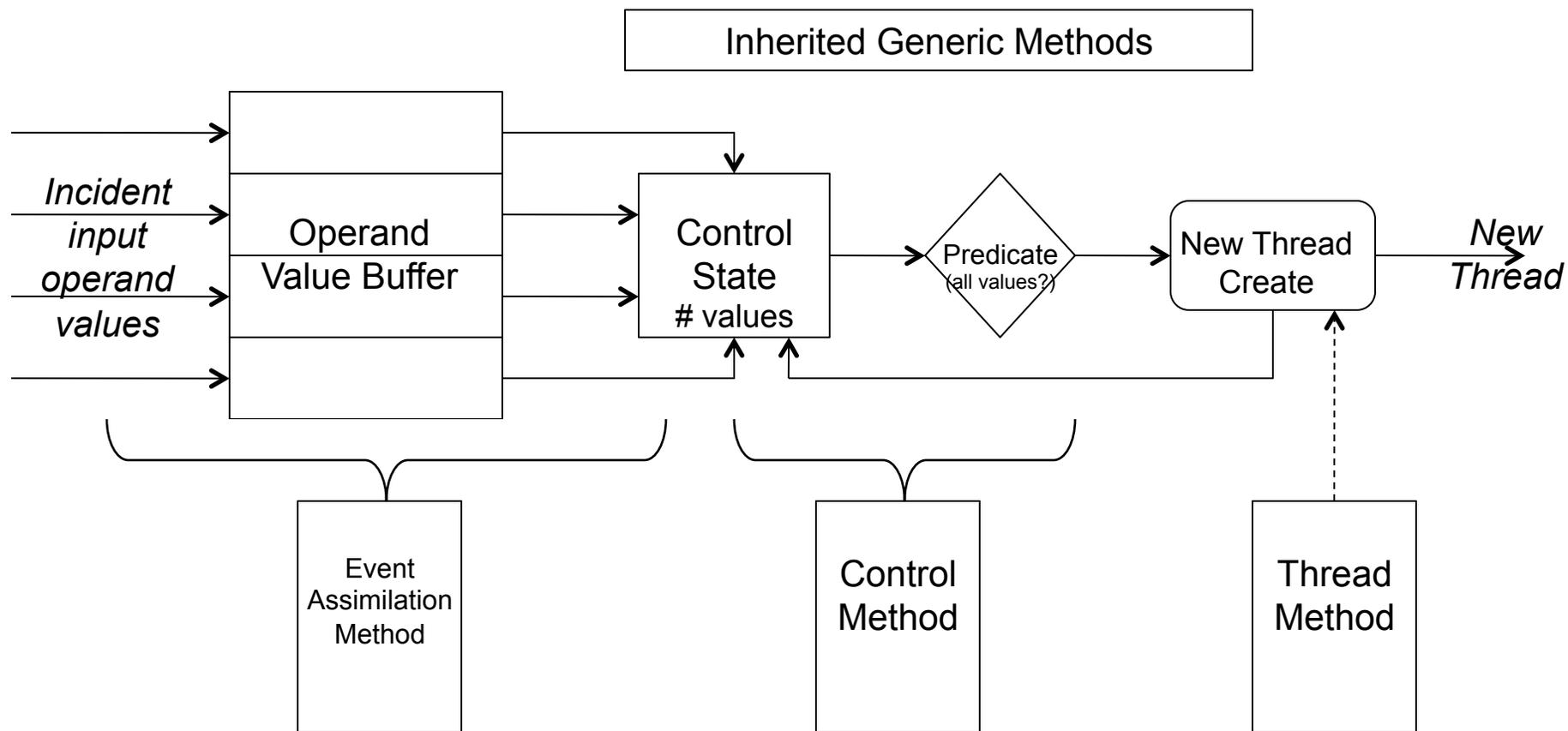


- Supports Dynamic-Adaptive Task Scheduling
- Declarative Semantics for Continuation of Execution
 - Defines conditions for work to be performed
 - Not imperative code by user
- Establishes Criteria for Task Instantiation
- Supports DAG flow control representation
- Examples:
 - Dataflow
 - Futures





Dataflow LCO



Motivation for Message-Driven Computation

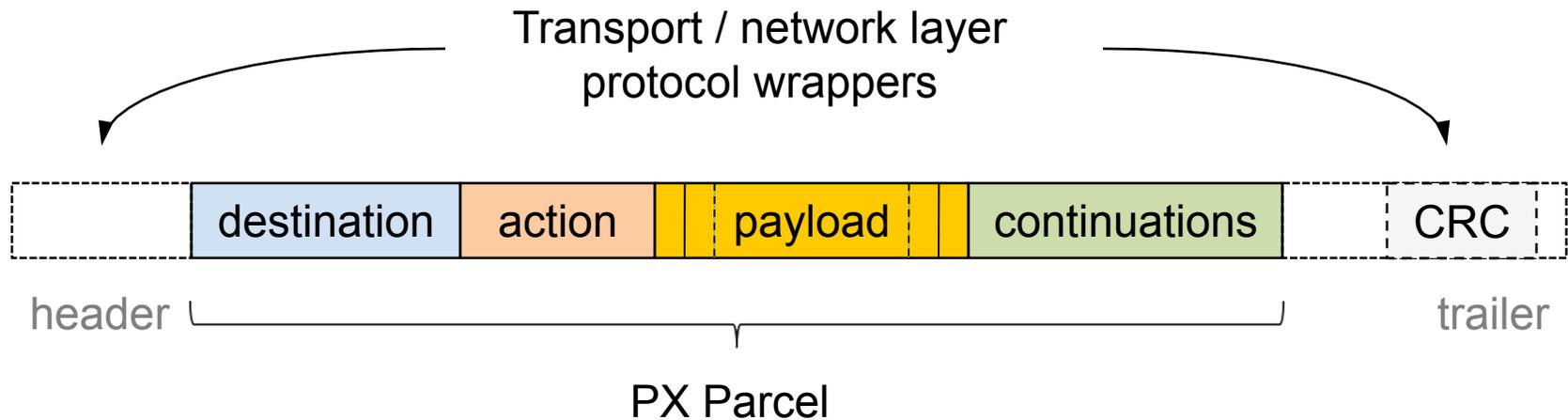


- To achieve high scalability, efficiency, programmability
- To enable new models of computation
 - e.g., ParalleX
- To facilitate conventional models of computation
 - e.g., MPI
- Hide latency
 - Support overlap of communication with computation
 - Move work to data, not always data to work
- Work-queue 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





Parcel Structure

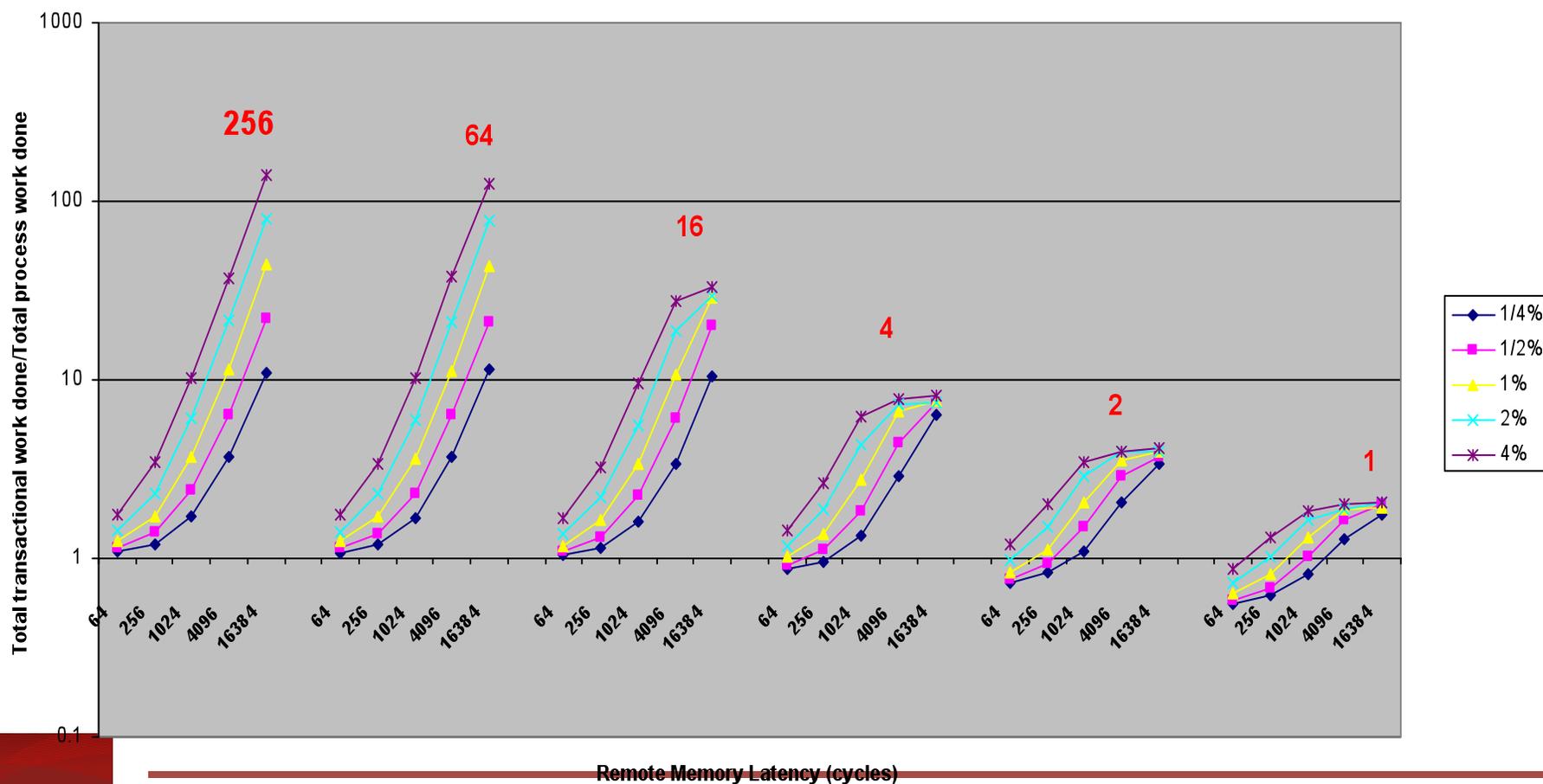


Parcels may utilize underlying communication protocol fields to minimize the message footprint (e.g. destination address, checksum)



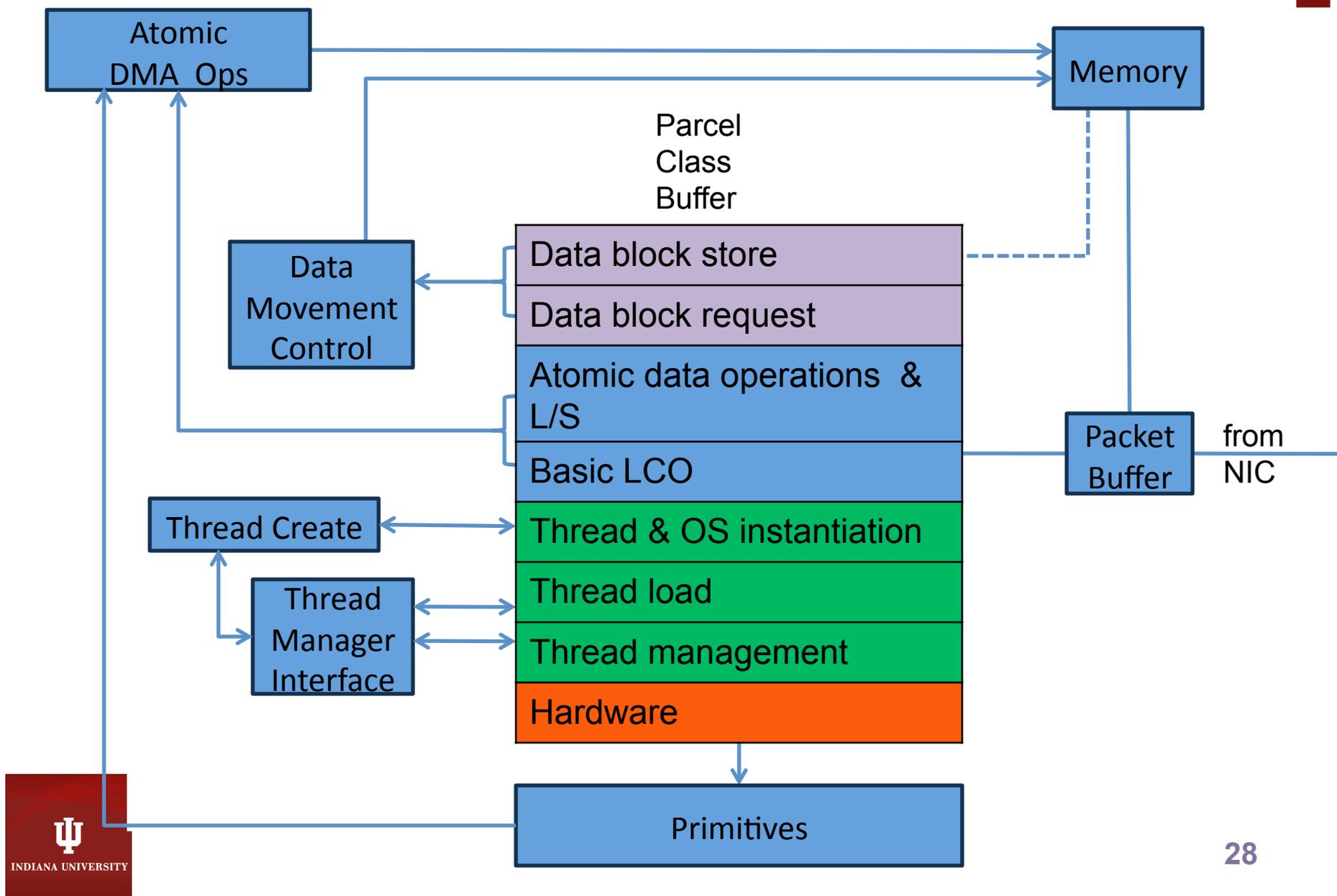
Latency Hiding with Parcels with respect to System Diameter in cycles

Sensitivity to Remote Latency and Remote Access Fraction
16 Nodes
deg_parallelism in RED (pending parcels @ $t=0$ per node)





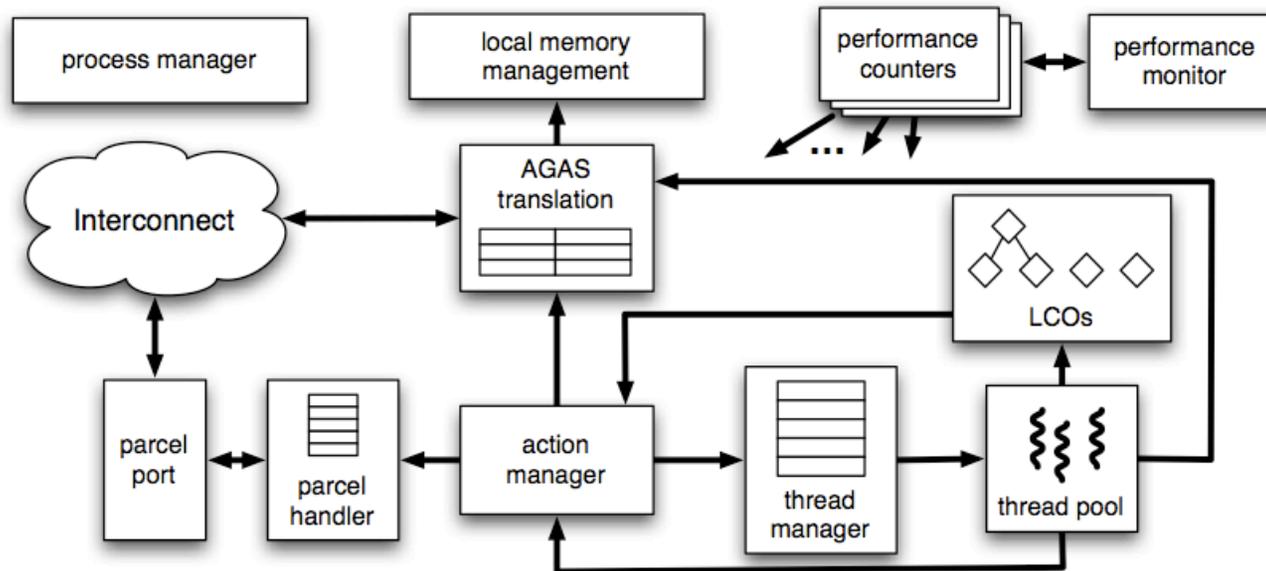
Parcel Handler Implementation



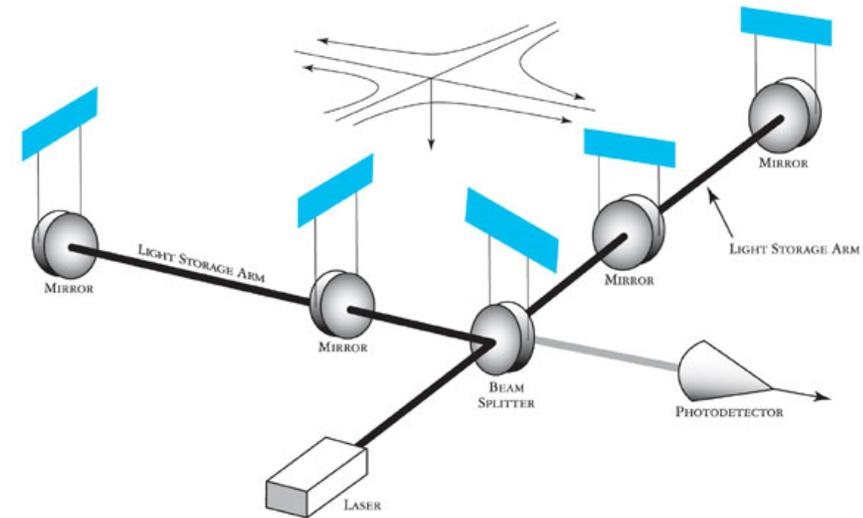


HPX Runtime Design

- Current version of HPX provides the following infrastructure as defined by the ParalleX execution model
 - Complexes (ParalleX Threads) and ParalleX Thread Management
 - Parcel Transport and Parcel Management
 - Local Control Objects (LCOs)
 - Active Global Address Space (AGAS)



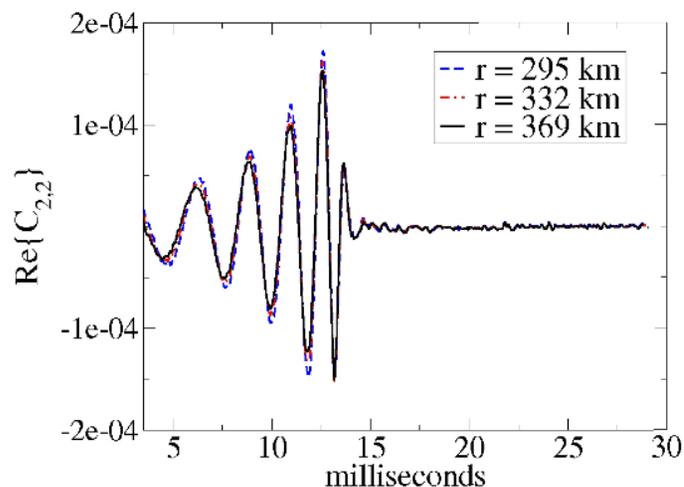
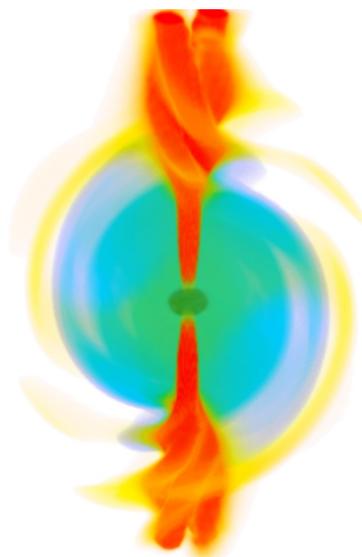
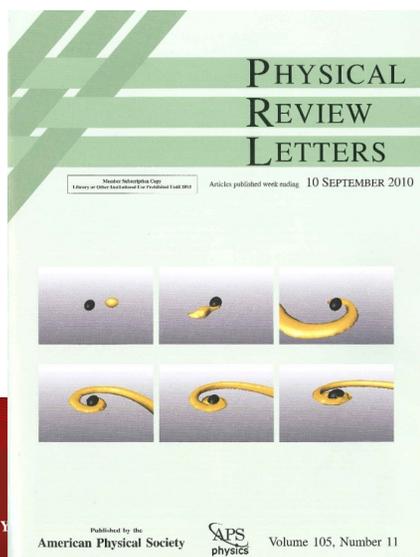
LIGO: Laser Interferometer Gravitational Observatory



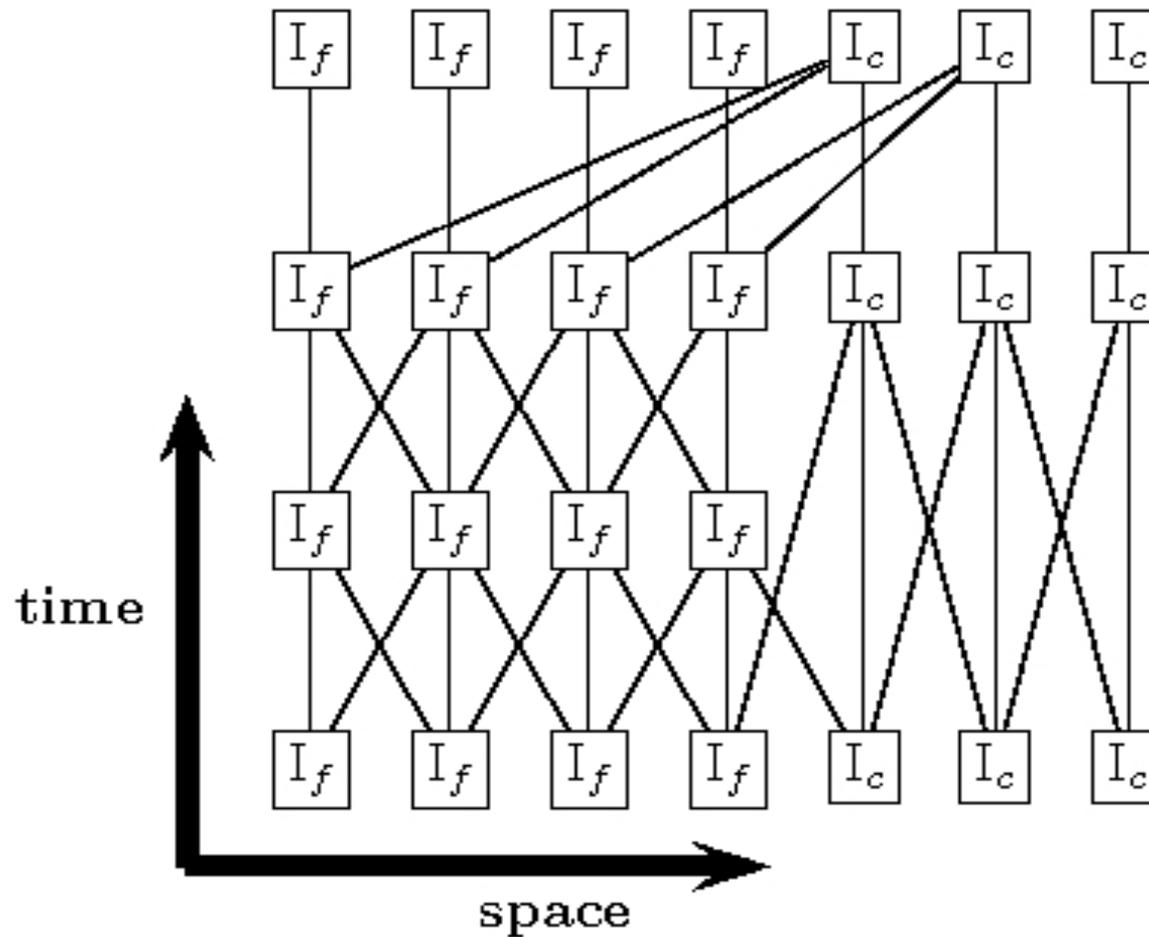


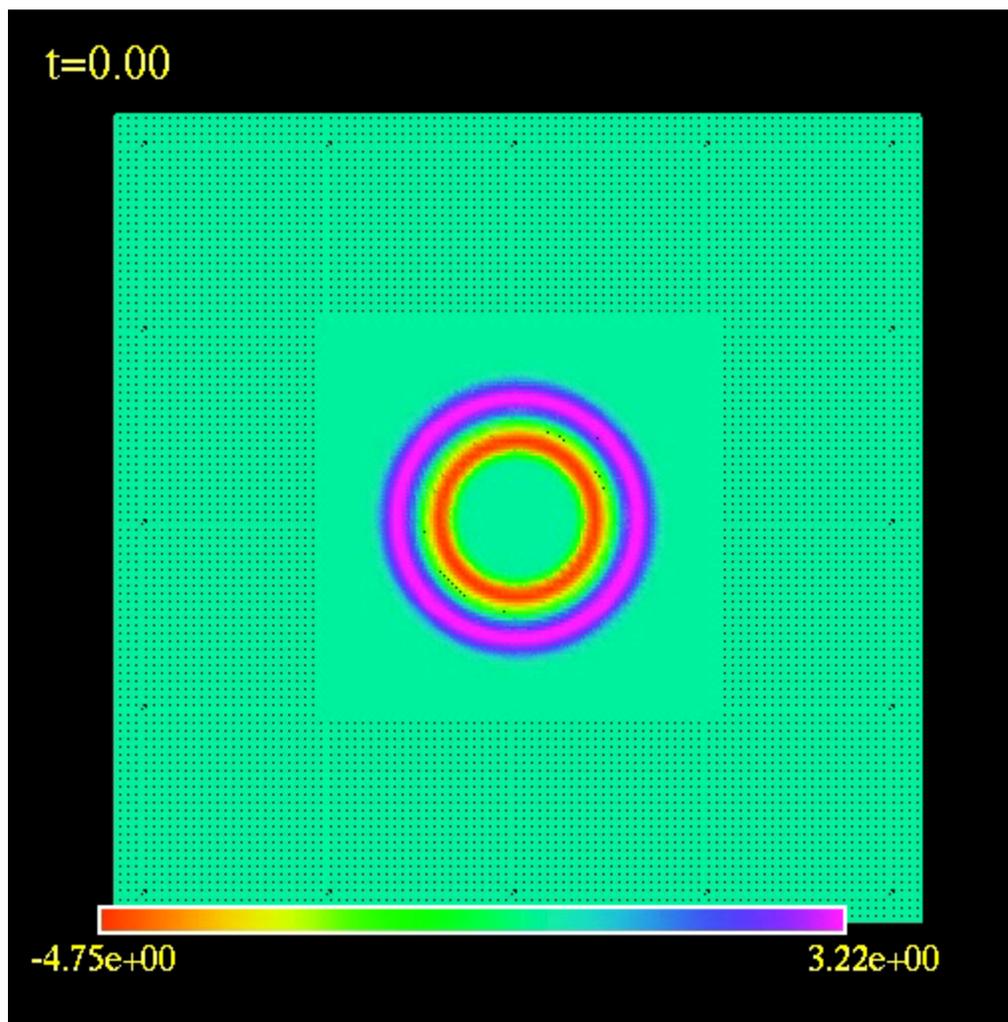
Adaptive Mesh Refinement

- Why Adaptive Mesh Refinement?
- The AMR dataflow in ParalleX
- The impact of granularity
- 3-D Test problem: comparisons between MPI based publicly available AMR toolkits and ParalleX AMR
- Impact of nedmalloc, concur, tcmalloc, and jmalloc



Constraint based Synchronization for AMR

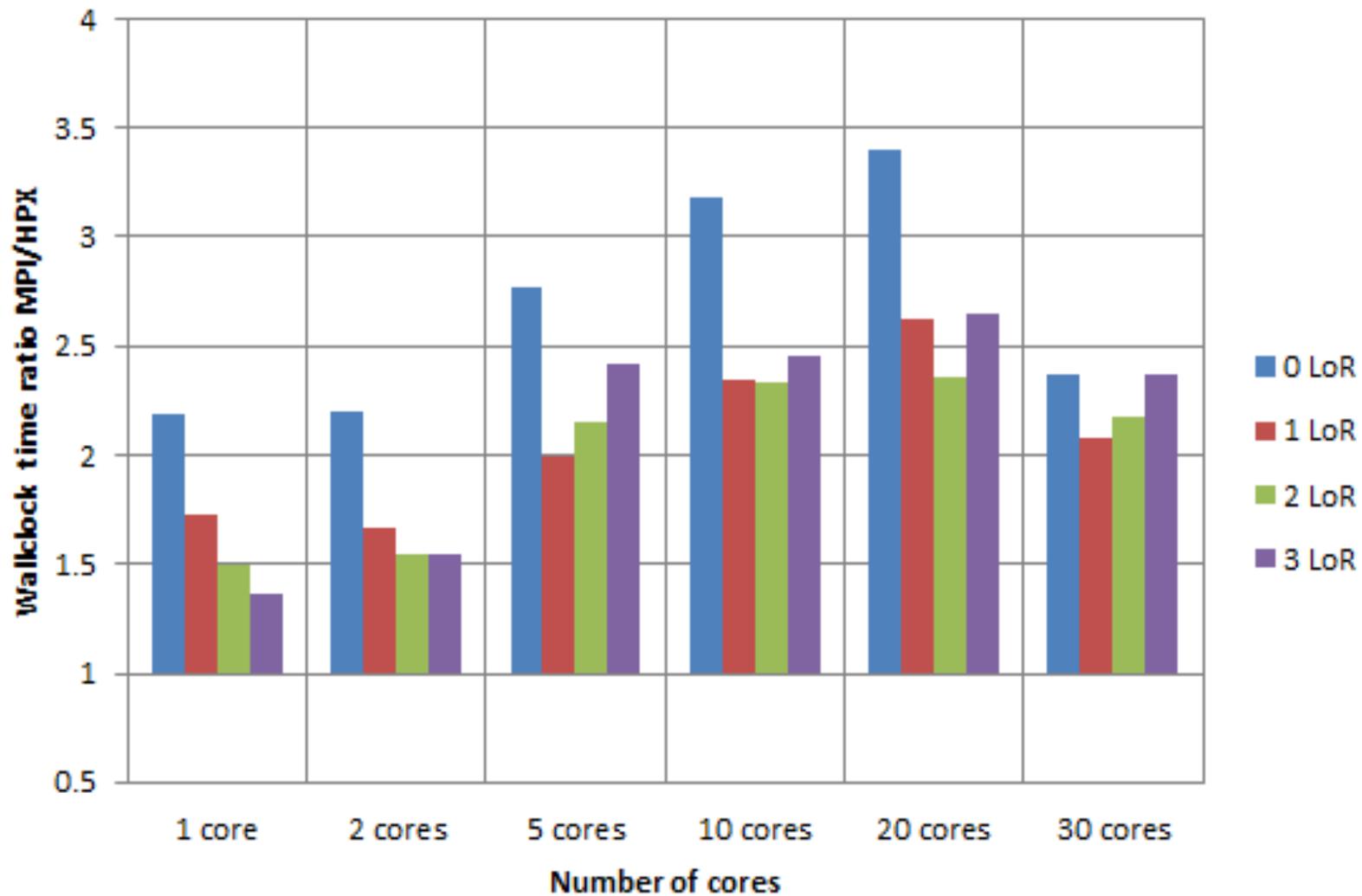




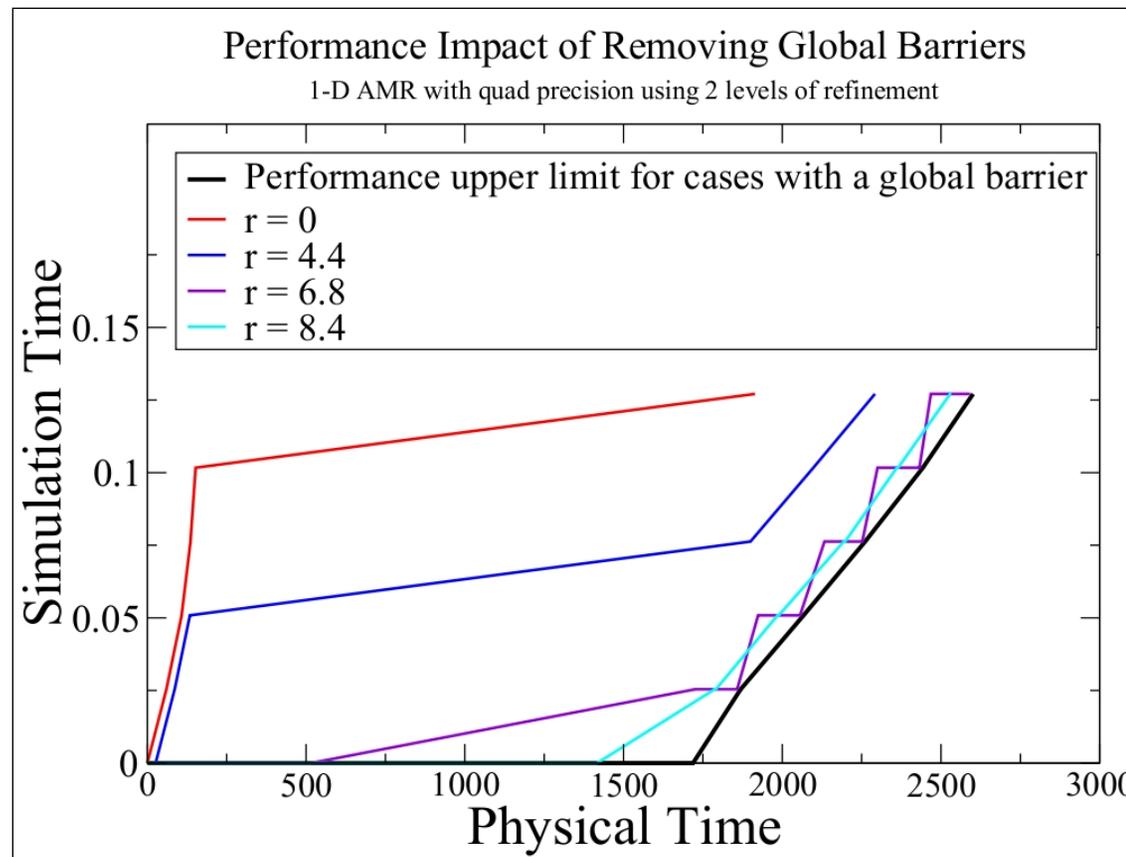


Wallclock time ratio MPI/HPX

(Depending on levels of AMR refinement - LoR, pollux.cct.lsu.edu, 32 cores)



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)



XPI: a ParalleX API

- Low-level programming syntax
 - MPI touch&feel
 - C bindings
- Source to source translation layer
 - Targets HPX library calls
 - Not implemented
- XPI_thread
- XPI_process
- XPI_parcel
- XPI_LCO



Conclusions

- Cluster Computing is going through a phase transition; a paradigm shift to exploit many core
- Needs a new execution model
 - Attack starvation, latency, overhead, & waiting for contention (SLOW)
 - Dynamic adaptive resource management & task scheduling
 - Dynamic graph-based applications for knowledge management (AI)
- ParalleX provides basis for new system methods & components
 - Programming models
 - Runtime systems
 - Architecture changes
- Commodity clusters continue to leverage their major strength
 - Economy of scale through mass market of component elements
 - System software and tools from high end and low end computing
 - Leader of commercial, scientific, and defense computing needs

