Presentation to:
HPC User Forum at HLRS:

Trends and New Directions in HPC

Thomas Sterling, Ph.D

Arnaud and Edwards Professor of Computer Science

Louisiana State University

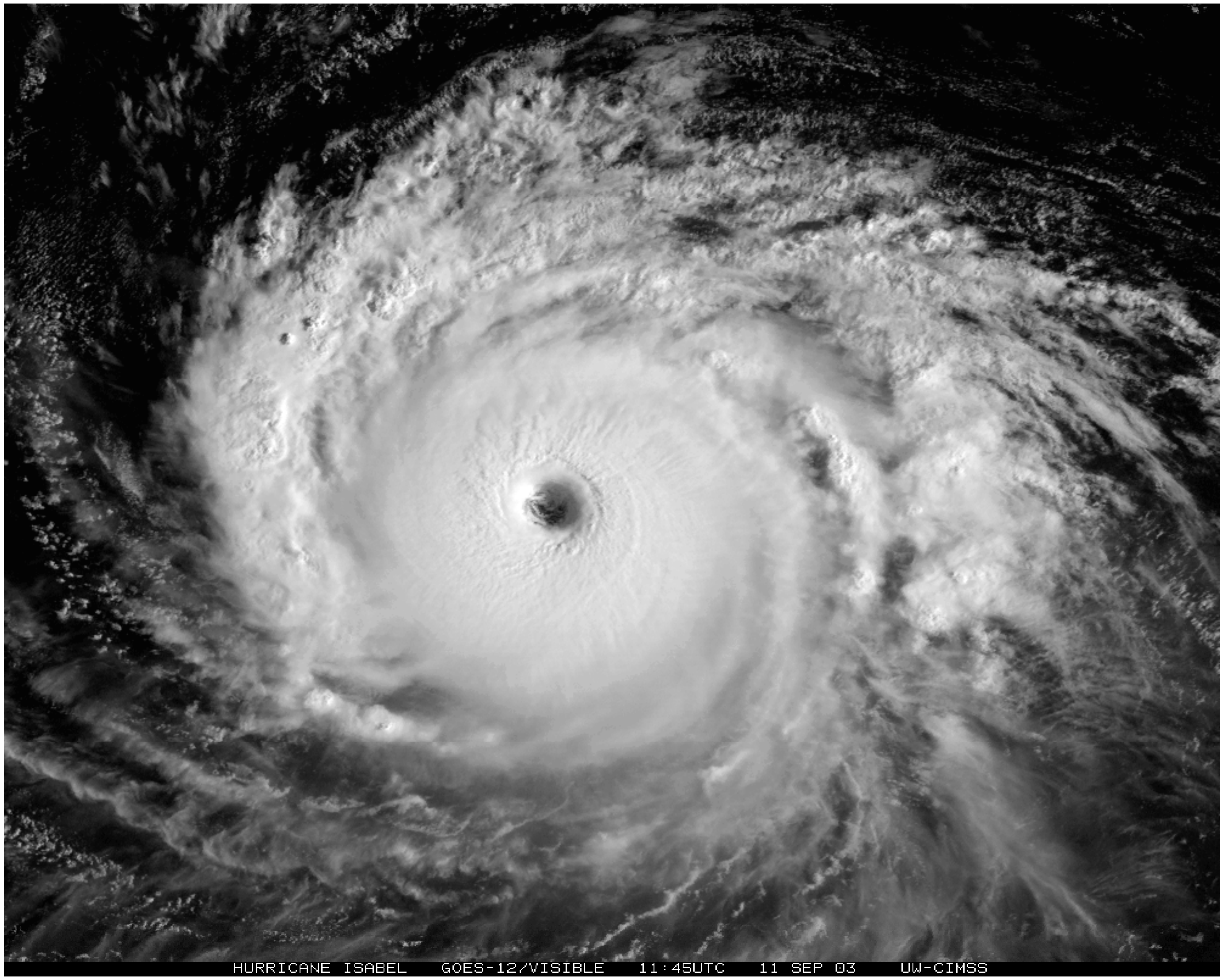
Visiting Associate, *California Institute of Technology*

Distinguished Visiting Scientist, *Oak Ridge National Laboratory*

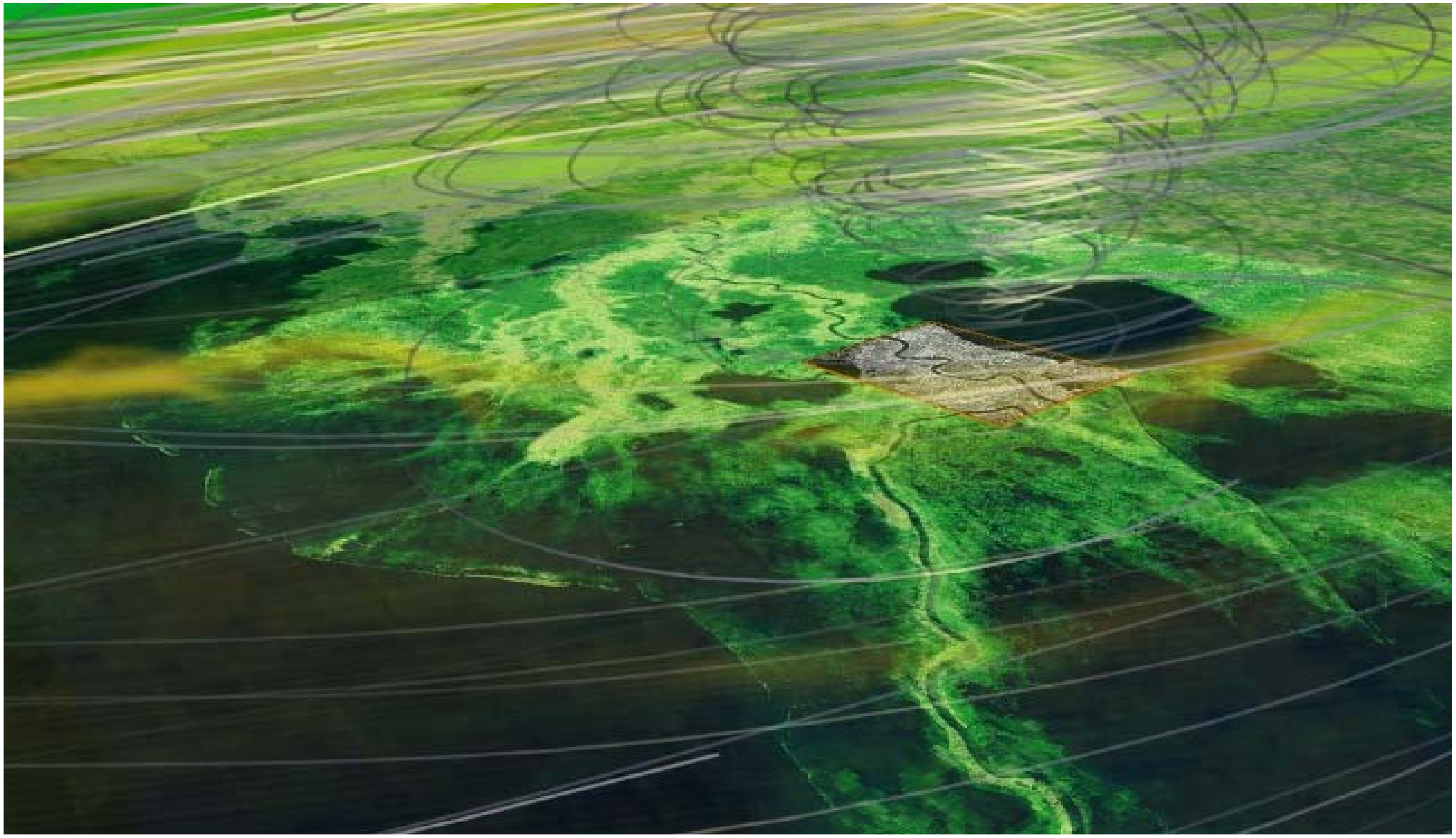
CSRI Fellow, *Sandia National Laboratory*

October 5, 2009

Gustav

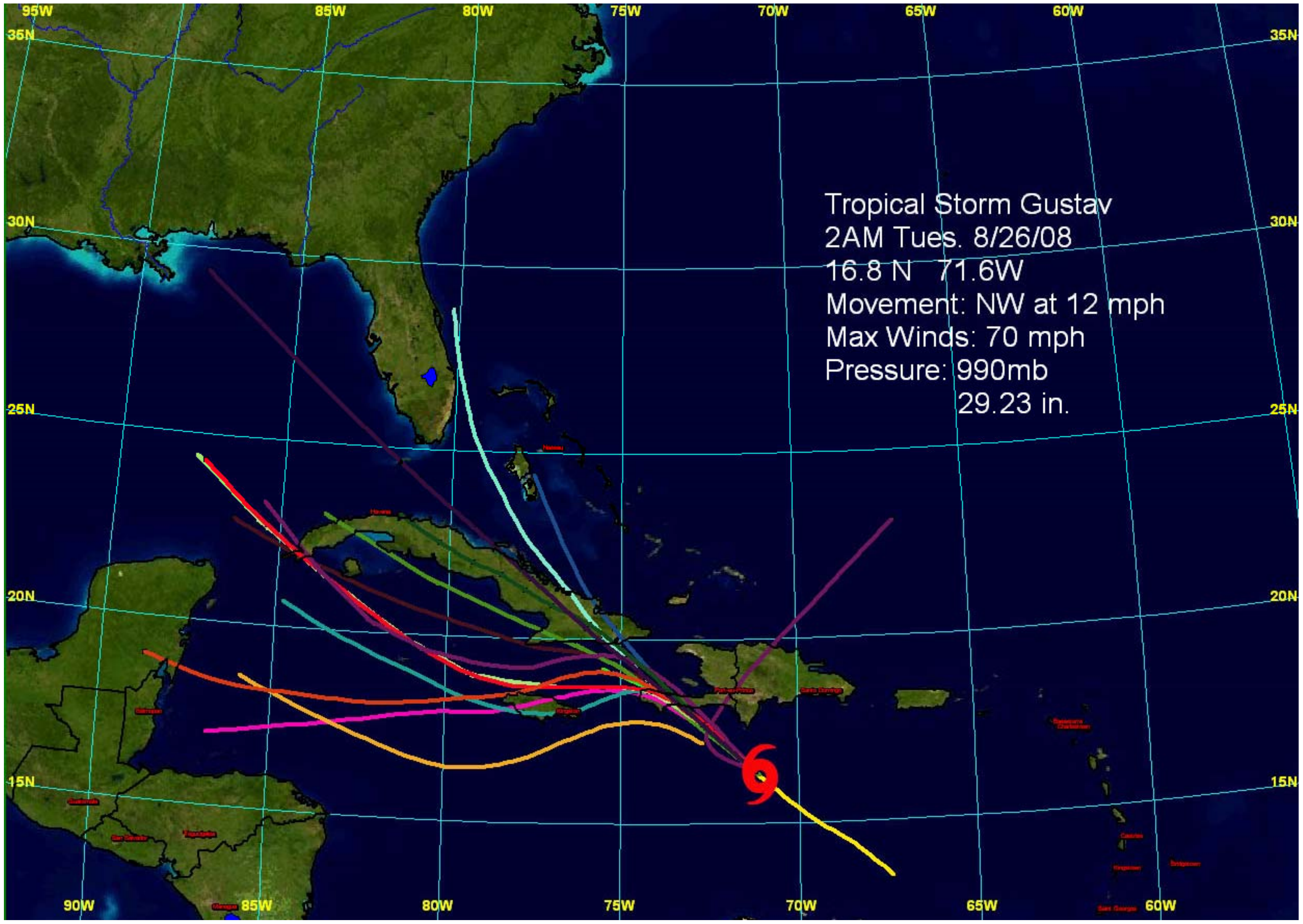

HURRICANE ISABEL GOES-12/VISIBLE 11:45UTC 11 SEP 03 UW-CIMSS

Simulating Hurricane Storm Surge



from Gabrielle Allen et al, LSU

Different Models – Time Critical Solutions



Key Points

- **HPC in Phase Change**
  - Technology pushes through punctuated equilibrium
- Role of new model of computation
  - Paradigm shift, adjusts to new set of needs
- ParalleX as an example
  - Includes some (not all) of the fundamental features needed
- Early results from ACS encouraged ParalleX project
  - Dynamic scheduling of lightweight user threads
  - Elimination of global barriers through lightweight synchronization

HPC Phases

I. Sequential instruction execution

II. Sequential instruction issue
   - pipeline execution,
   - reservation stations,
   - ILP

III. Vector
   - pipelined arithmetic, registers, memory access
   - Cray

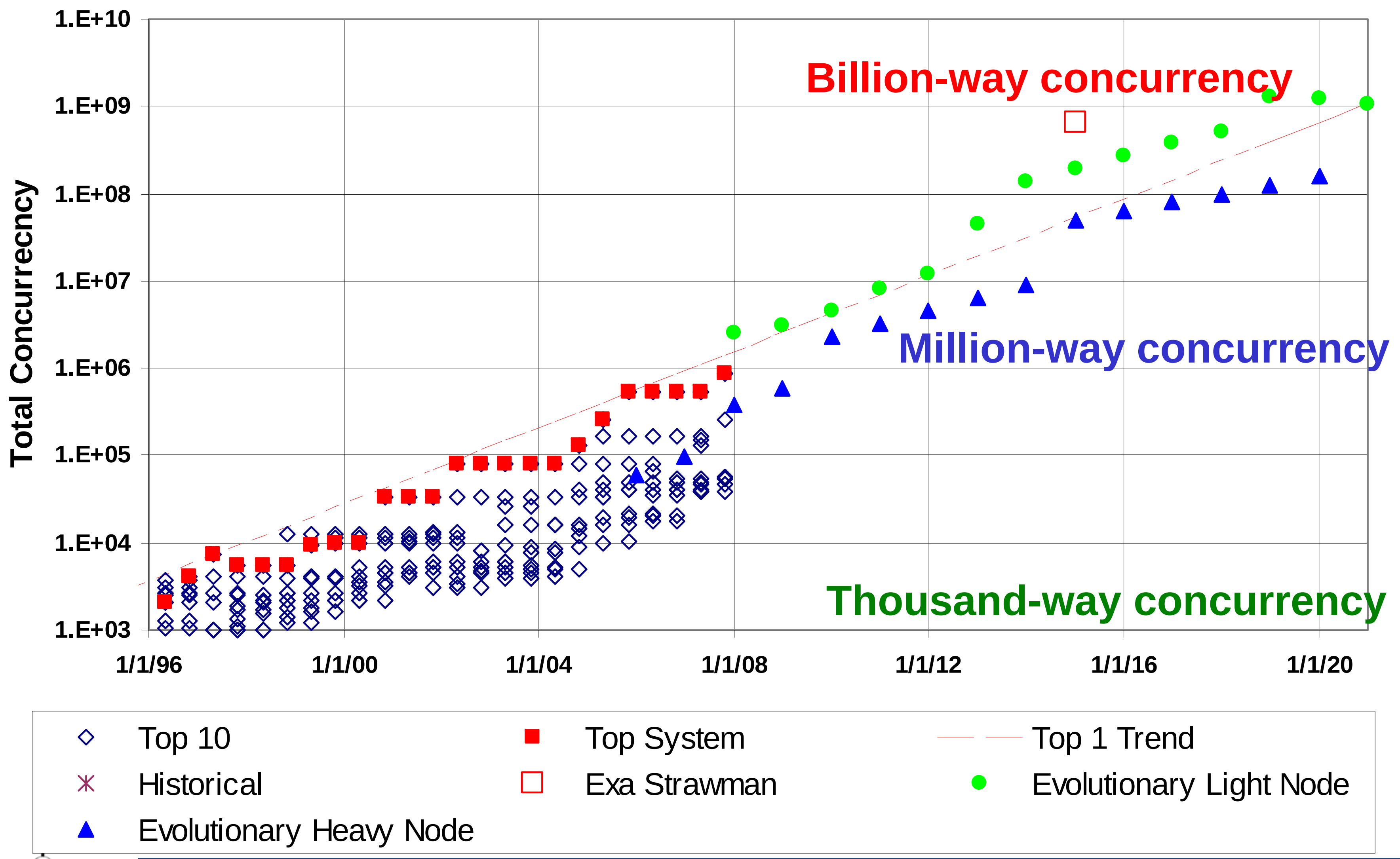
IV. SIMD
   - MasPar, CM-2

V. CSP
   - MPP, clusters
   - MPI

HPC in Phase Change

- Technology advances demand new system structures and operational modalities for optimality
- SLOW – sources of performance degradation
  - Starvation: insufficient parallelism
  - Latency: of access and action to remote resources
  - Overhead: of critical work for resource management
  - Waiting: for contention of access to shared resources
- Paradigm shift in organizing computing
  - Architecture
  - Programming models and compilation techniques
  - OS and runtime
- The 6th Phase
  - Scalability, efficiency, power, programmability, reliability

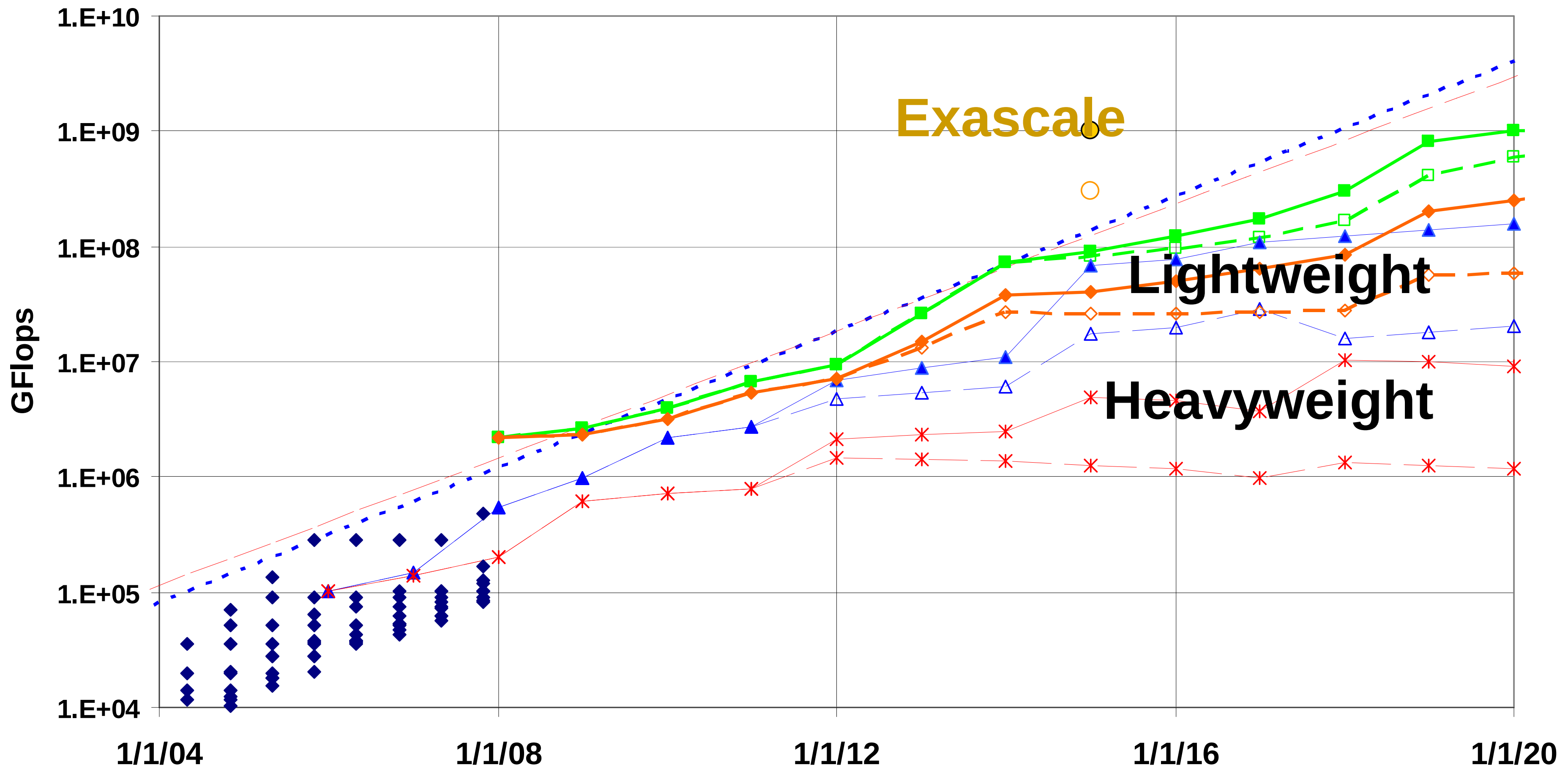
Data Center Total Concurrency



LSU
CENTER FOR COMPUTATION & TECHNOLOGY

Courtesy of Peter Kogge, UND

Data Center Performance Projections



But not at 20 MW!

CENTER FOR COMPUTATION & TECHNOLOGY

Courtesy of Peter Kogge, UND

Strategic Requirements

- Scalability
- Efficiency
- Storage capacity
- Power consumption
- Reliability, and
- Programmability

Exascale Design Point

- Feature size of 22 to 11 nanometers, CMOS in 2018
- Total average of 25 Pico-joules per floating point operation
- Approximately 10 billion-way parallelism
- 100 million to 1 billion cores
- Clock rates of 1 to 2 GHz (this is approximate with a possible error of a factor of 2)
- Multi-threaded fine grain parallelism of 10 to 100 way concurrency per core
- 100’s of cores per die (varies dramatically depending on core type, and other factors)
- Global address space without cache coherence; extensions to PGAS
- 128 Petabytes capacity mix of DRAM and nonvolatile memory
- Explicitly managed high speed buffer caches; part of deep memory hierarchy
- Optical communications for distances > 10 centimeters, possibly inter-socket
- Optical bandwidth of 1 Terabit per second
- System-wide latencies on the order of 10’s of thousands of cycles
- Active power management to eliminate wasted energy by unused cores
- Fault tolerance by means of graceful degradation and dynamically reconfigurable structures
- Hardware rapid thread context switching
- Hardware message to thread conversion for message-driven computation
- Hardware lightweight synchronization
- 3-D packaging of dies for stacks of 4 to 10 dies each

Key Points

- HPC in Phase Change
  - Technology pushes through punctuated equilibrium
- **Role of new model of computation**
  - Paradigm shift, adjusts to new set of needs
- ParalleX as an example
  - Includes some (not all) of the fundamental features needed
- Early results from ACS encouraged ParalleX project
  - Dynamic scheduling of lightweight user threads
  - Elimination of global barriers through lightweight synchronization

Purpose of Models of Computation

- Not just an academic intellectual exercise
- Critical <u>tool</u> for realizing effective Exascale applications
- To address challenges imposed by technology changes
- To exploit opportunities delivered by technology advances
- Reflects change to (VIth) HPC phase
- Facilitates co-design of separate but interoperable system layers
- Supports new methods
- Frees previous constraints, “deadly embrace” of conventional practices

Attributes of an Execution Model (1)

- A single representation of a system class
  - All such systems share the same specifications
  - All layers of the system are implicitly represented
    - Architecture, OS, runtime, compiler, programming models
    - Not necessarily explicitly exposed
    - Effects of interoperability as an emergent property
  - Systems may be distinguished by implementation
- Strategies
  - Approach to solving perceived problems to effective operation
  - Approach to exploiting the new capabilities of advanced technologies
- Semantics
  - Named and manipulated entities
  - Action categories
    - Threads, processes, functions, operations, atomic
  - Parallelism
    - Forms, granularity, synchronization, ordering, eager/lazy evaluation

Attributes of an Execution Model (2)

- Controlling physical resources
  - Managing power
  - Reconfiguring for graceful degradation
  - Self-aware status monitoring and load balancing response
- Policies
  - Invariants of operational properties specified
  - Implementation methods to achieve properties unspecified
  - Leaves flexibility
    - Adapting alternative technologies
    - Distinctive balance points between hardware and software
  - Permits differentiation

Attributes of an Execution Model (3)

- Abstract to physical relationships
  - Address translation
  - Routing
  - Protection
  - Distribution versus locality management
  - Dynamic adaptive migration
  - Moving work to data, not just data to work
    - Traversing continuations
- Determines commonality
  - Portability
  - Stability across generations
  - ISV application software product targets

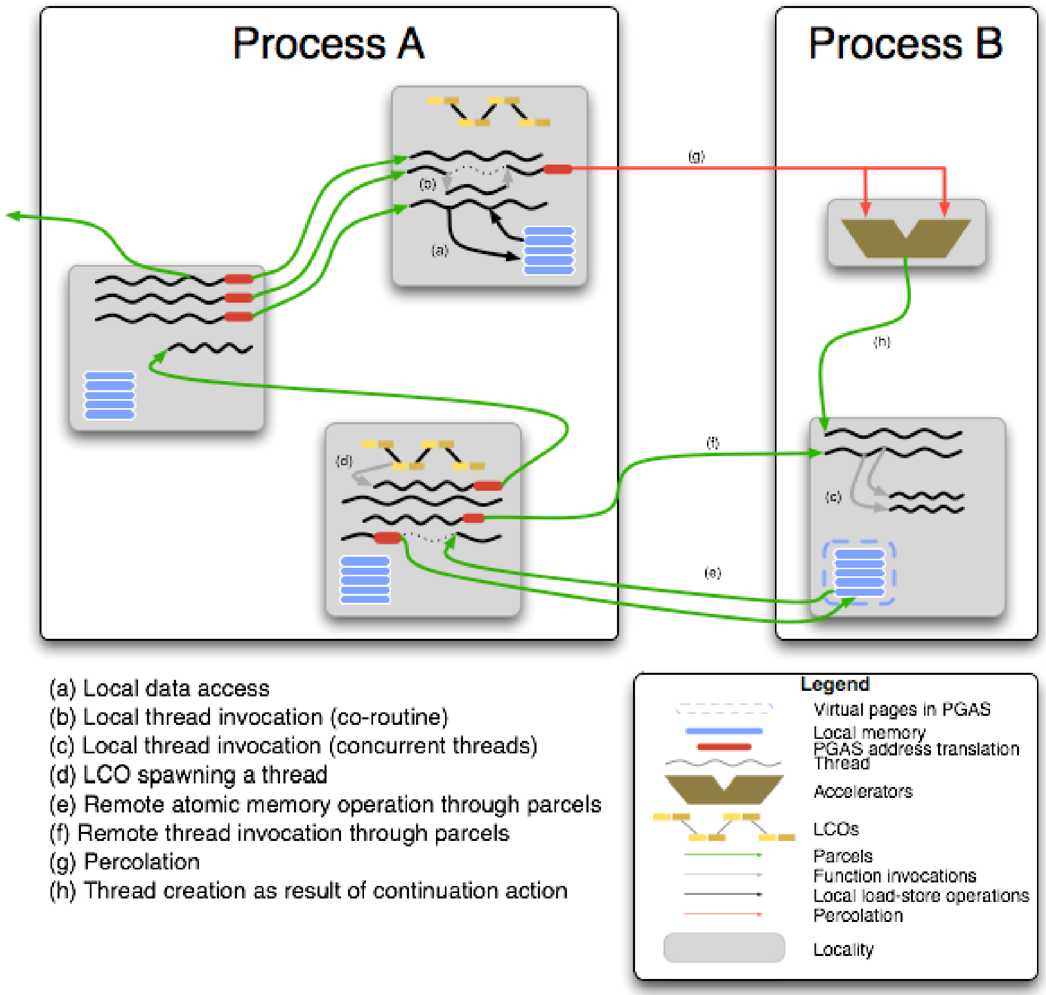
Goals of a New Model of Parallel Computation for Exascale

- Serve as a **discipline** to govern future scalable system architectures, programming methods, and runtime
- **Latency** hiding at all system distances
  - Latency mitigating architectures
- Exploit **parallelism** in diversity of forms and granularity
- Provide a framework for efficient **fine grain synchronization** and scheduling (dispatch)
- Enable optimized runtime **adaptive resource management** and task scheduling for dynamic load balancing
- Support **full virtualization** for **fault tolerance** and power management, and continuous optimization
- Self aware infrastructure for **power management**
- **Semantics** of failure response for **graceful degradation**
- Complexity of operation as an emergent behavior from **simplicity of design**, high replication, and local adaptation for global optima in time and space

Key Points

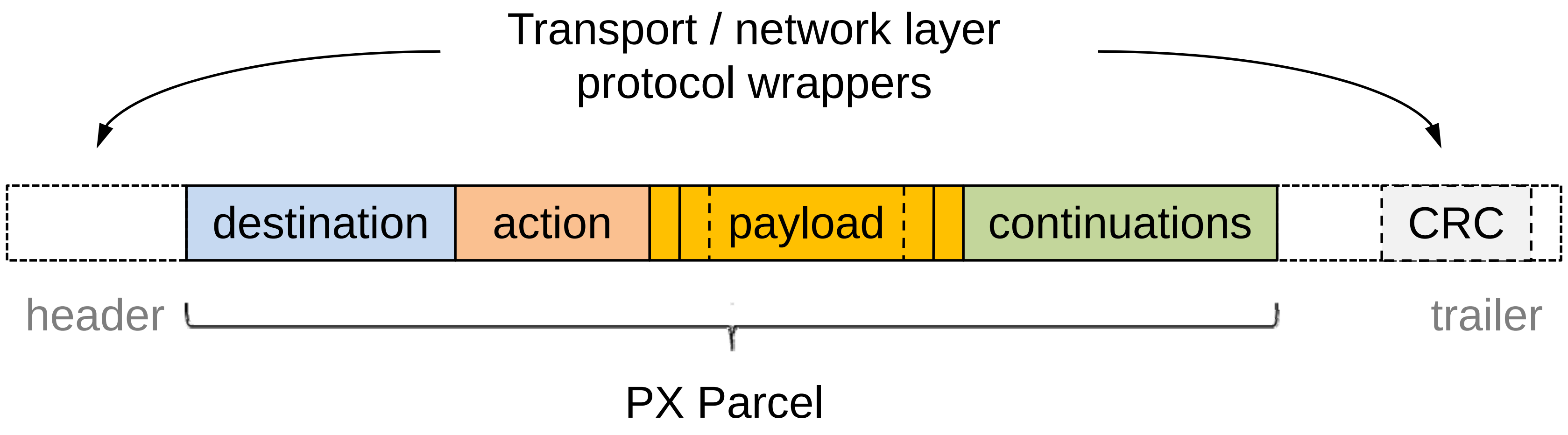
- HPC in Phase Change
  - Technology pushes through punctuated equilibrium
- Role of new model of computation
  - Paradigm shift, adjusts to new set of needs
- **ParalleX as an example**
  - Includes some (not all) of the fundamental features needed
- Early results from ACS encouraged ParalleX project
  - Dynamic scheduling of lightweight user threads
  - Elimination of global barriers through lightweight synchronization

ParalleX Model Components



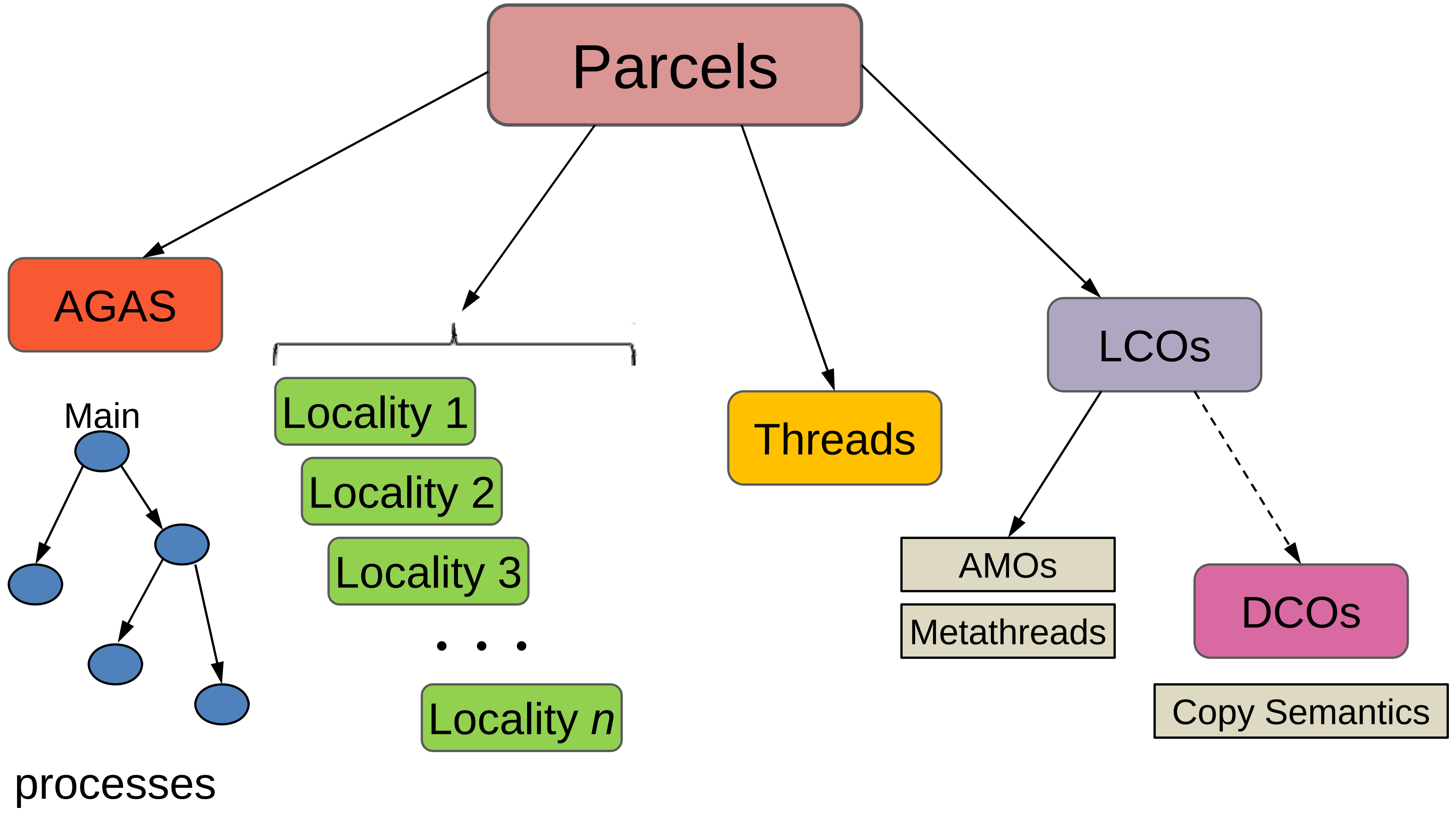
(a) Local data access
(b) Local thread invocation (co-routine)
(c) Local thread invocation (concurrent threads)
(d) LCO spawning a thread
(e) Remote atomic memory operation through parcels
(f) Remote thread invocation through parcels
(g) Percolation
(h) Thread creation as result of continuation action

Parcel Structure



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

Parcel Interaction with the System



Parcel Destination

- Application specific addressing:
  - Global virtual address of target (recipient) object
  - Current implementation: HPX GID (global ID), 128-bit
- System specific addressing:
  - Physical address of a hardware resource
  - Supports direct access to register space or state machine manipulation
  - May be required for percolation

Parcel Actions

- Data movement
  - Block read and write
  - Lightweight scalar load/store
- Synchronization
  - Atomic Memory Operations
  - Basic LCOs
- Thread manipulation
  - Thread instantiation
  - Thread register access
  - Thread control and state management
- Direct hardware access
  - Counters
  - Physical memory
  - State machines

LCOs

- A number of forms of synchronization are incorporated into the semantics
- Support message-driven remote thread instantiation
- 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

Multithreading

- Threads are collections of related operations that perform on locally exchanged data values
- A thread is a continuation combined with a 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
- Thread can realize transaction phase
- Thread does not assume dedicated execution resources
- Thread is first class object identified in global name space
- Thread is ephemeral

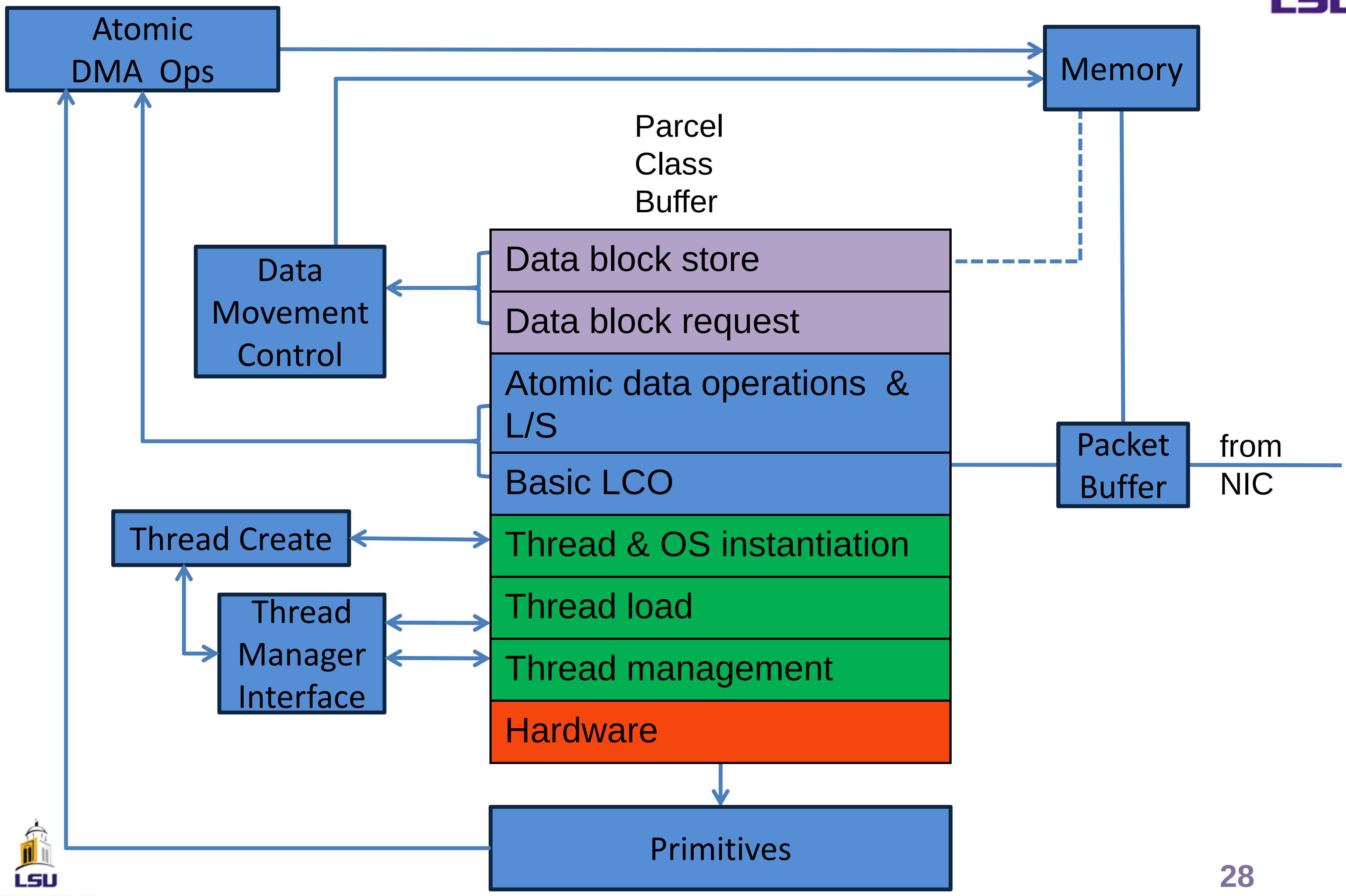
Parcel Payload

- Lightweight
  - Remote function arguments (includes LCO actions)
  - AMO operands
  - Scalar load / store
- Heavyweight
  - Action-dependent
  - Migrating object state
  - Page relocation
  - Some percolation instances

Parcel Continuations

- Enables migration of flow control across global space
- Format: list of arbitrary LCOs
- Accept result(s) returned by parcel-invoked action
- Continuation types
  - Return the value to the requestor
  - Standard LCO evaluation
    - Spawn local computation
    - Propagate the result to another locality via parcel
  - Perform a system call

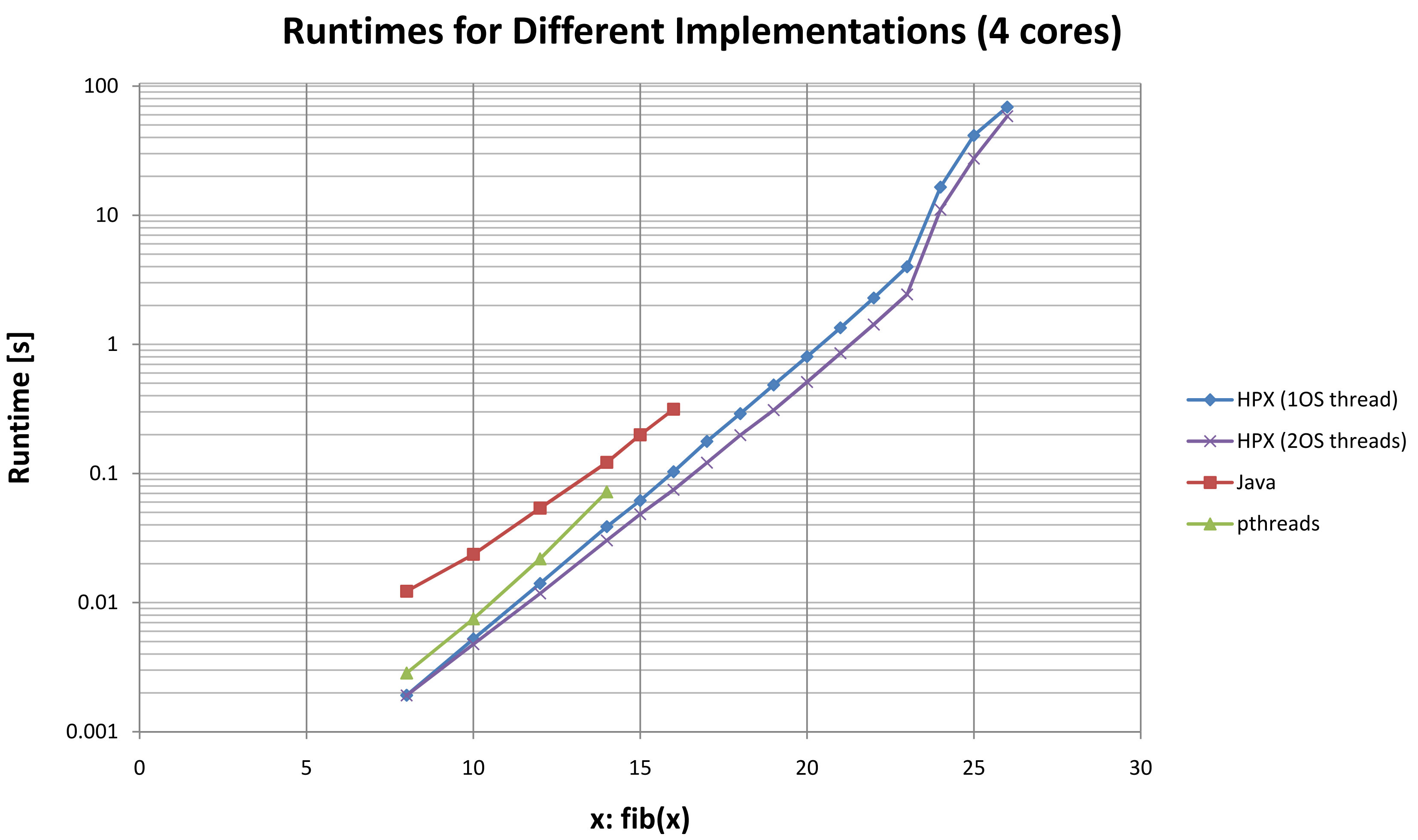
Parcel Handler Implementation



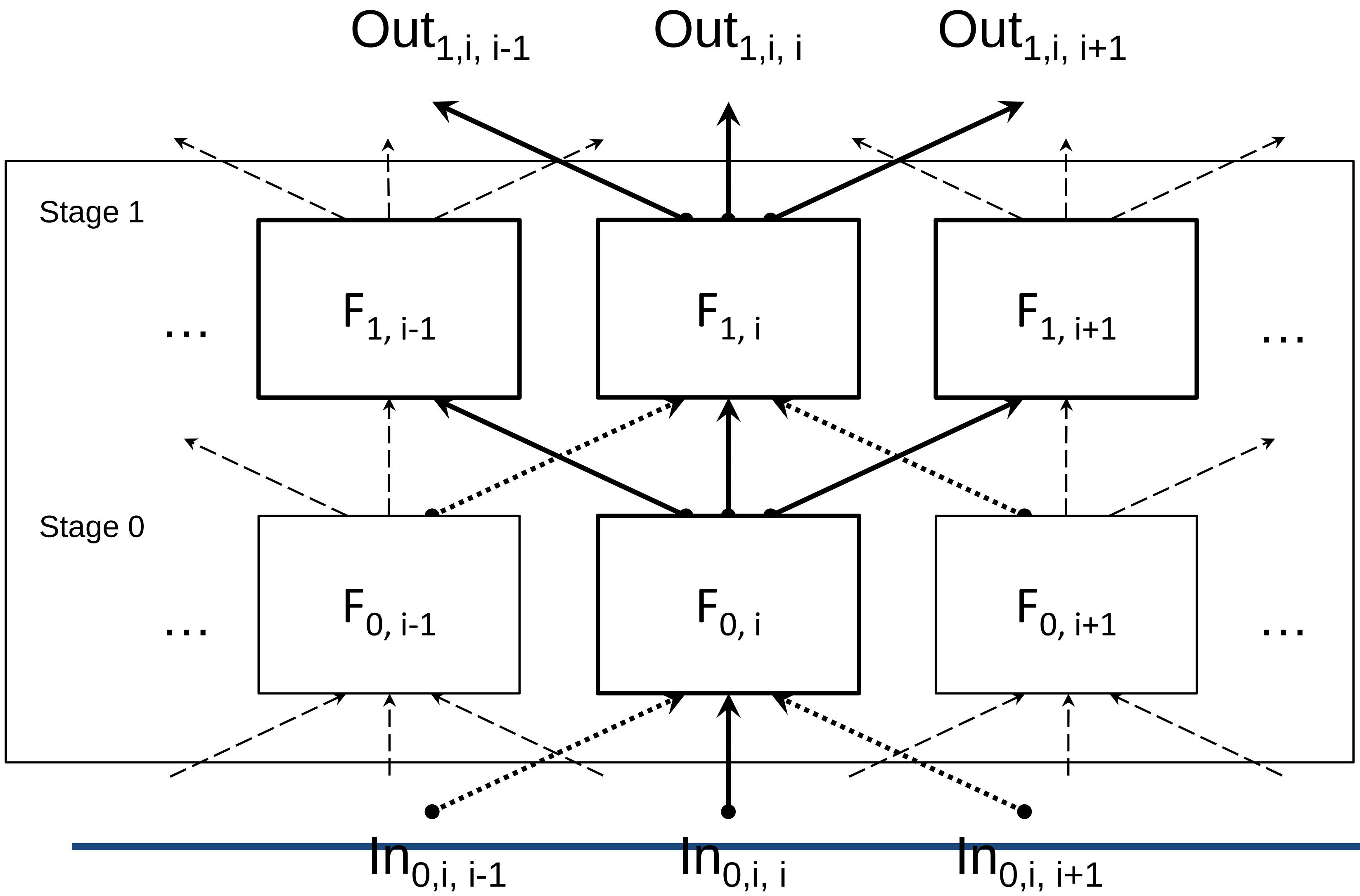
Key Points

- HPC in Phase Change
  - Technology pushes through punctuated equilibrium
- Role of new model of computation
  - Paradigm shift, adjusts to new set of needs
- ParalleX as an example
  - Includes some (not all) of the fundamental features needed
- **Early results from ACS encouraged ParalleX project**
  - Dynamic scheduling of lightweight user threads
  - Elimination of global barriers through lightweight synchronization

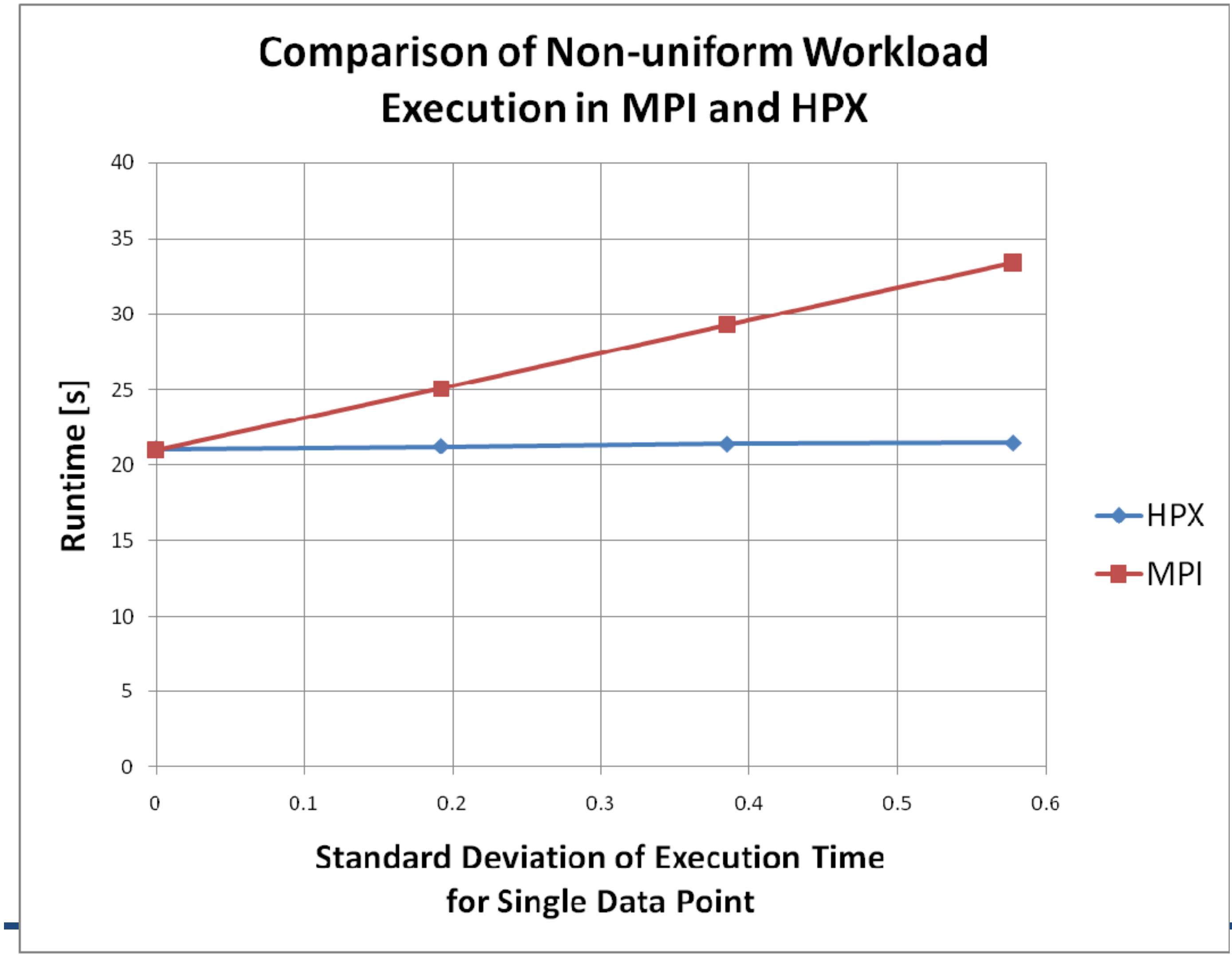
Fibonacci Sequence



Using HPX for AMR



Using HPX for AMR



Meanwhile, back on Earth

- MPI-n
  - Better alternative?
  - Opens up intra-process parallelism
    - UPC, GASnet inside?
  - Backwards compatible with > decade of legacy codes
- Megalopolis computing
  - Concurrent operation of multiple interoperating activities
  - Disparate life cycles (sources, languages) for code modules
- ParalleX supercharging for Exascale in a future real-world
  - Scaling through data-driven execution (?)
    - Merges metadata and control flow
    - Dynamic graph based problems
  - Self-aware operation for dynamic scheduling & power management
  - *compute-validate-commit* cycle for in situ micro-shedding fault tolerance