A Keynote Presentation to LCSC 2006:

Multicore Beowulf Clusters for Petaflops-scale Computing

Thomas Sterling

Louisiana State University
California Institute of Technology
Oak Ridge National Laboratory

October 18, 2006

Center for Computation & Technology

AT LOUISIANA STATE UNIVERSITY

The new 512-host dual-Xeon cluster at LSU

Integrator: Atipa HPL benchmark: 2.1 Tflops

Petaflops-Year (Pfy)

- The amount of useful work performed on a single application by a Petaflops-scale (delivered performance) system in one year
  - Application dependent
- Equivalent to the Simulation of the evolution visible-lifetime of the Pinwheel Galaxy (M101)
  - Approximately 1 trillion stars
  - Approximately 100 billion years
  - Treated as an N-body tree code
  - Ignore dissipative medium
  - When the lights turn off due to:
    - Singularity assimilation
    - Kinetic Evaporation

- Equivalent to approximately 1 billion Standard Compute Days
  - A good days run on a desk top
    - 8 to 10 ours – overnight or from morning to lights-out
  - A moving target, changes with hardware evolution

Proteins: the Basic Building Blocks of Life

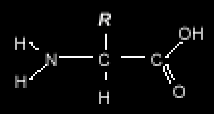
Function known for many proteins

- Structural: keratin (skin, hair, nail), collagen (tendon), fibrin (clot)
- Motive: actomyosin (muscle)
- Transport: Hemoglobin (blood)
- Catalysis: Enzymes

Precursor of fibrin. Fibrin polymerizes to form blood clots. Conversion of fibrinogen to fibrin is regulated via a cascade of factors to control blood clotting.

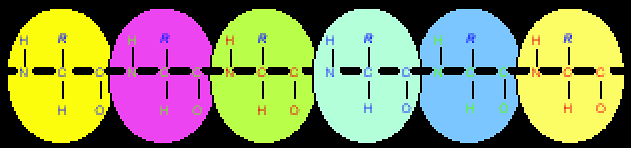
```
>3FIB:_ FIBRINOGEN GAMMA CHAIN
QIHDITGKDCQDIANKGAKQSGLYFIKPLKANQQFLVYCEIDG
SGNGWTVFQKRLDGSVDFKKNWIQYKEGFGHLSPTGTTEFWLG
NEKIHLISTQSAIPYALRVELEDWNGRTSTADYAMFKVGPEAD
KYRLTYAYFAGGDAGDAFDGFDFGDDPSDKFFTSHNGMQFSTW
DNDNDKFEGNCAEQDGSGWWMNKCHAGHLNGVYYQGGTYSKAS
TPNGYDNGIIWATWKTRWYSMKKTTMKIIPFNRL
```

Sequence known for "all" proteins (genome project)



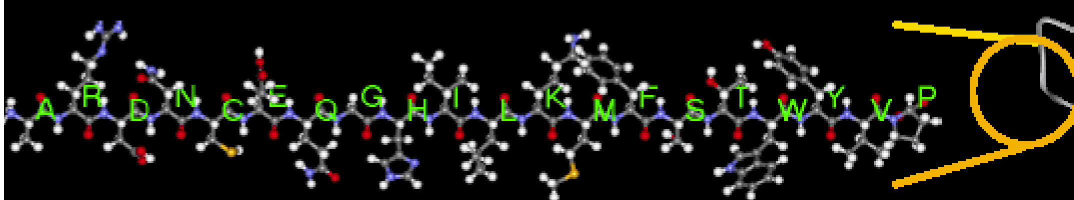
Amino acid

There are 20 natural amino acids with different physicochemical properties, such as: shape, volume, flexibility, hydrophobic, hydrophilic, charge



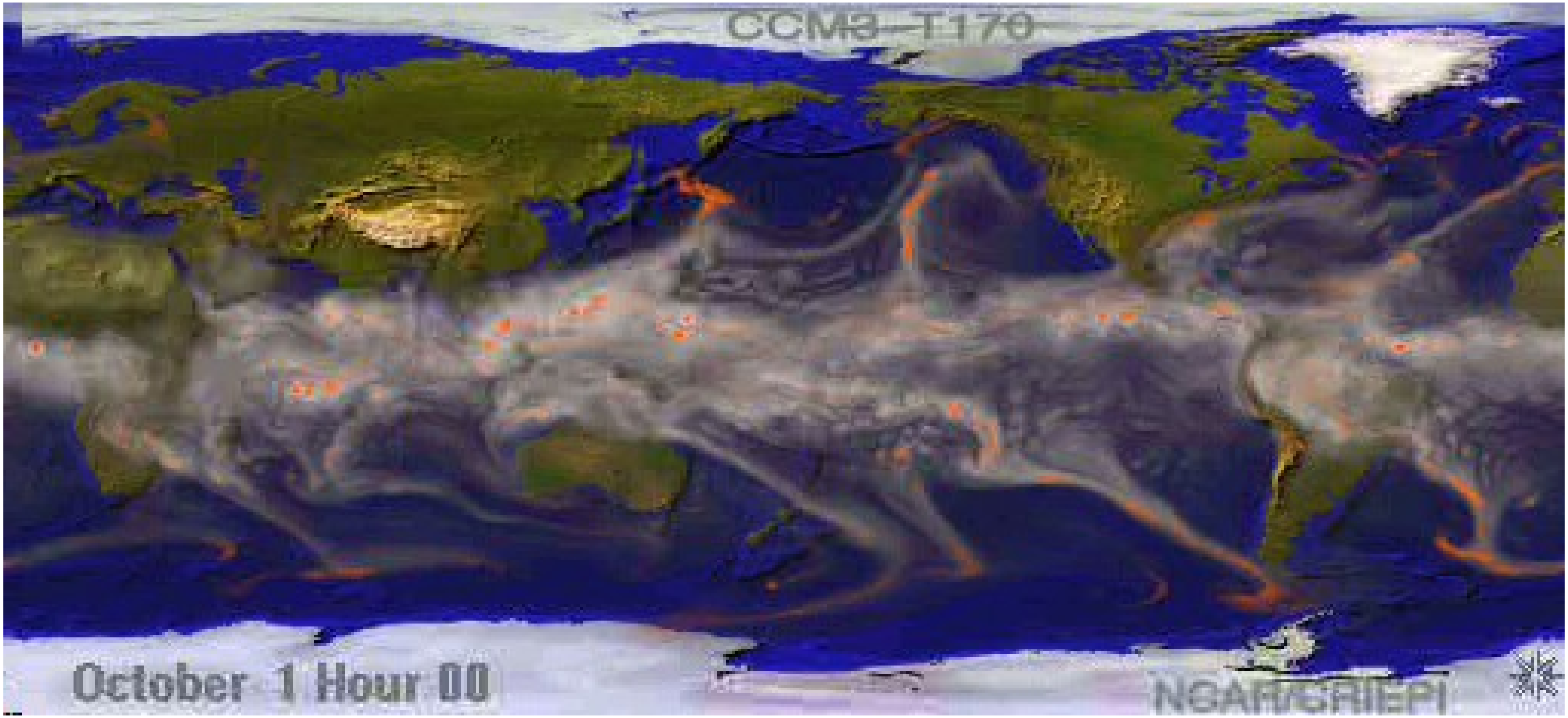
"Beads on a string"

Structure known for few proteins (crystalography)



Global Climate Model Simulation

Precipitable Water and Precipitation Rate



Natural and Human Sources of Greenhouse Gases

Computing Needs and Realities

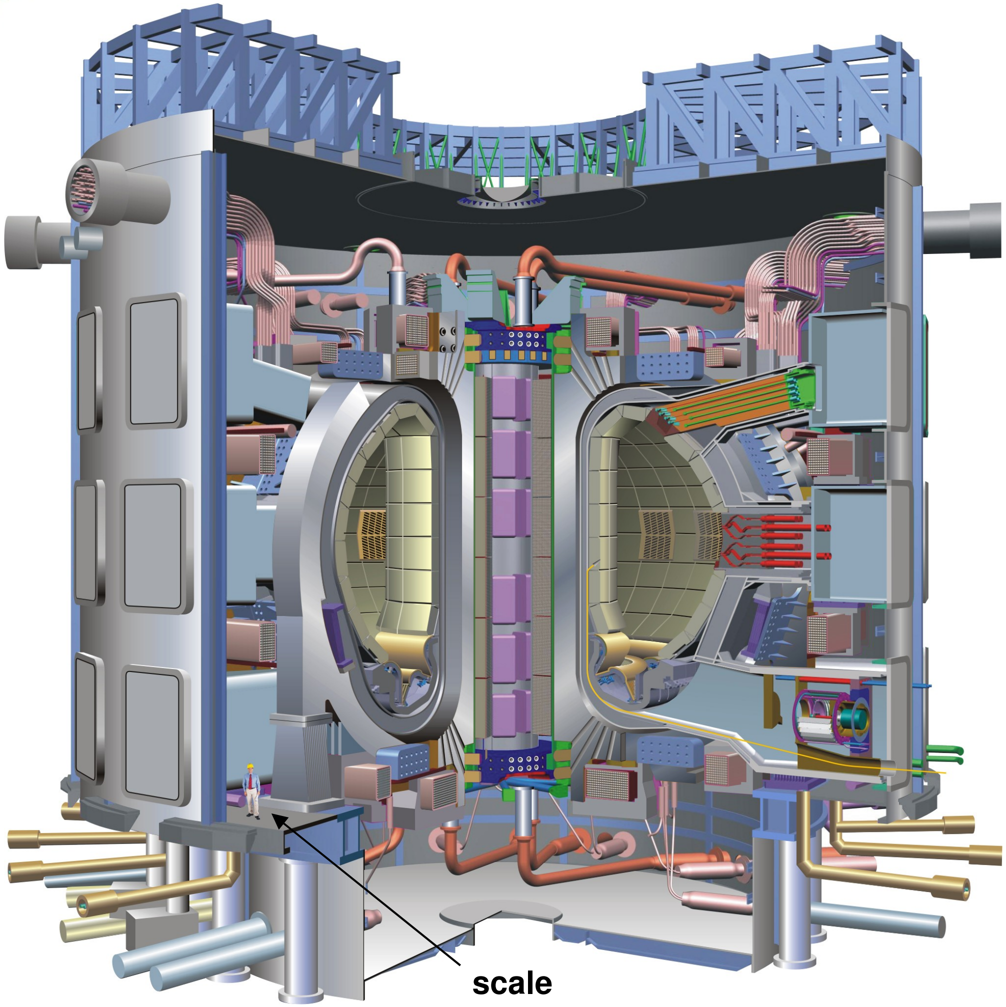
- Throughput required ~5 years/day for ensemble simulation (century/month)
- Long integration times/ensembles required for climate
  - non-deterministic problem with large natural variability
  - long equilibrium time scales for coupled systems
  - *computational capability 0th-order rate limiter*
- Quality of solutions are resolution and physics limited
  - balance horizontal and vertical resolution, and physics complexity
  - *computational capability 0th-order rate limiter*

Issue	*Motivation*	*Compute Factor*
Spatial resolution	Provide regional details	10^3-10^5
Model completeness	Add "new" science	10^2
New parameterizations	Upgrade to "better" science	10^2
Run length	Long-term implications	10^2
Ensembles, scenarios	Range of model variability	10
Total Compute Factor		10^{10}-10^{12}

Ref: ***A SCIENCE-BASED CASE FOR LARGE-SCALE SIMULATION***

Volume 2

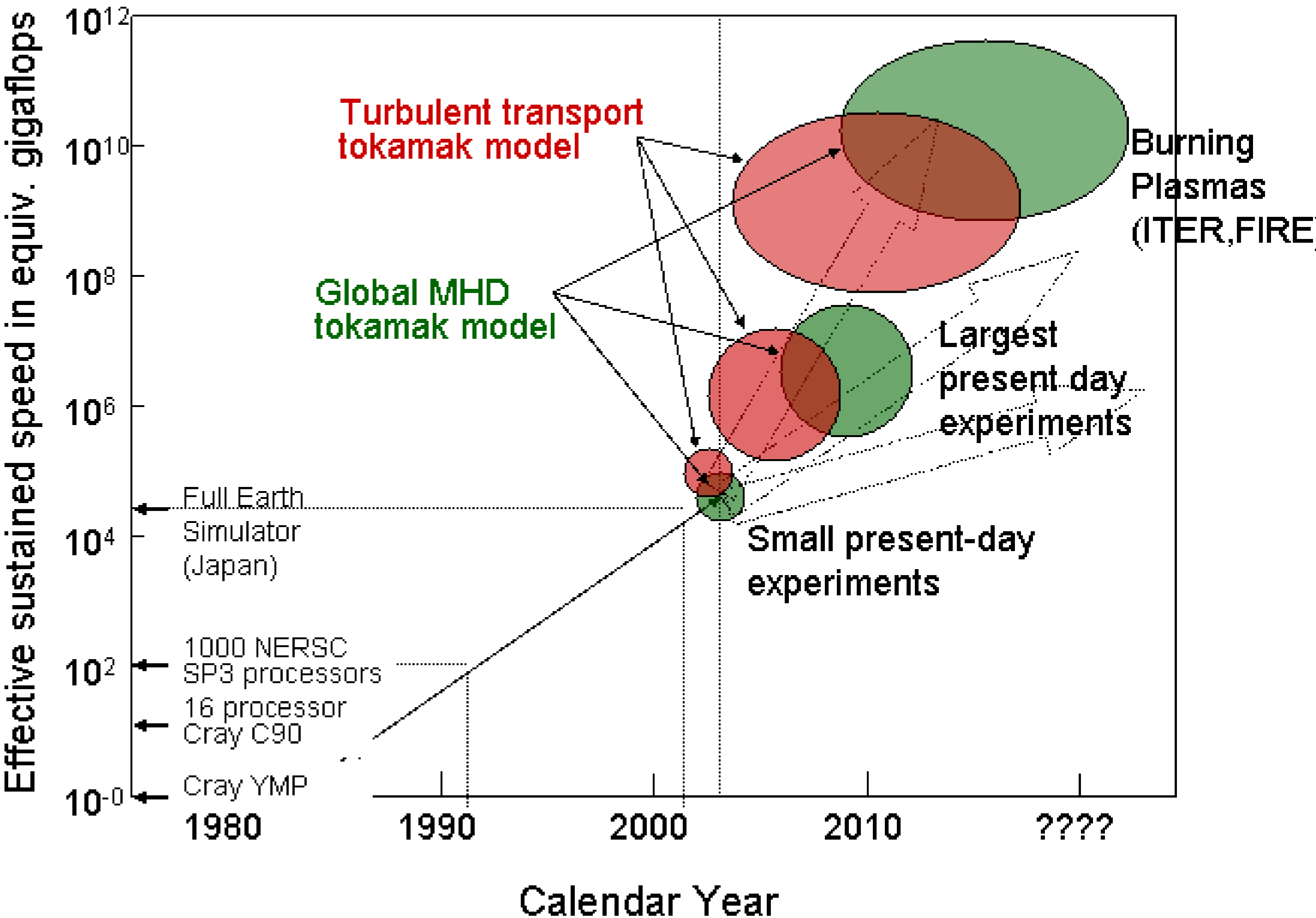
The U.S. is an official partner in ITER



- 500 MW fusion output
- Cost: $ 5-10 B
- To begin operation in 2015

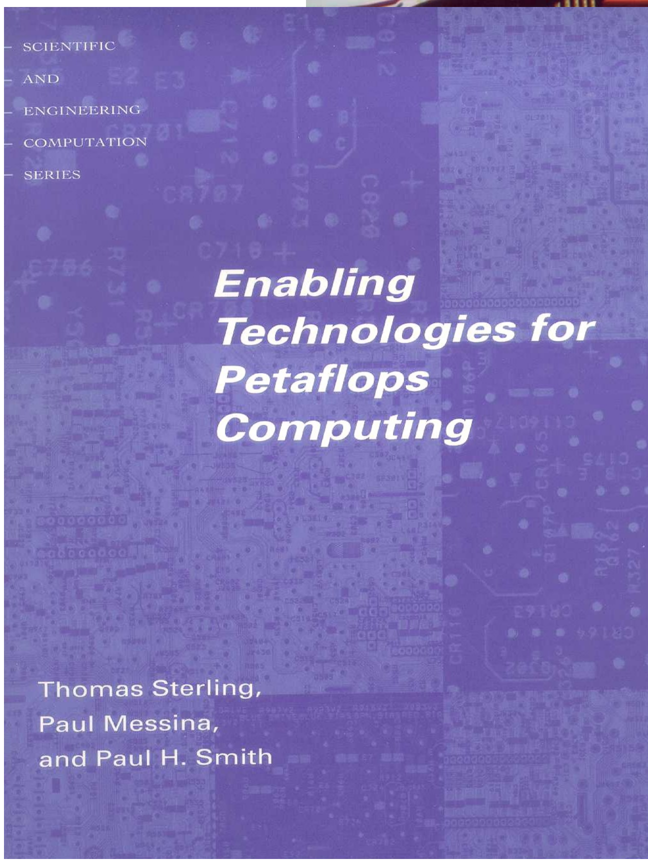
International
Thermonuclear
Experimental
Reactor:

- European Union
- Japan
- United States
- Russia
- Korea
- China



Effective sustained speed in equiv. gigaflops
10^{12}
10^{10}
10^{8}
10^{6}
10^{4}
10^{2}
10^{-0}
Turbulent transport
tokamak model
Global MHD
tokamak model
Burning
Plasmas
(ITER,FIRE)
Largest
present day
experiments
Small present-day
experiments
Full Earth
Simulator
(Japan)
1000 NERSC
SP3 processors
16 processor
Cray C90
Cray YMP
1980
1990
2000
2010
????
Calendar Year


SUPERCONDUCTOR


SCIENTIFIC
AND
ENGINEERING
COMPUTATION
SERIES
Enabling Technologies for Petaflops Computing
Thomas Sterling,
Paul Messina,
and Paul H. Smith

We didn’t get it all right

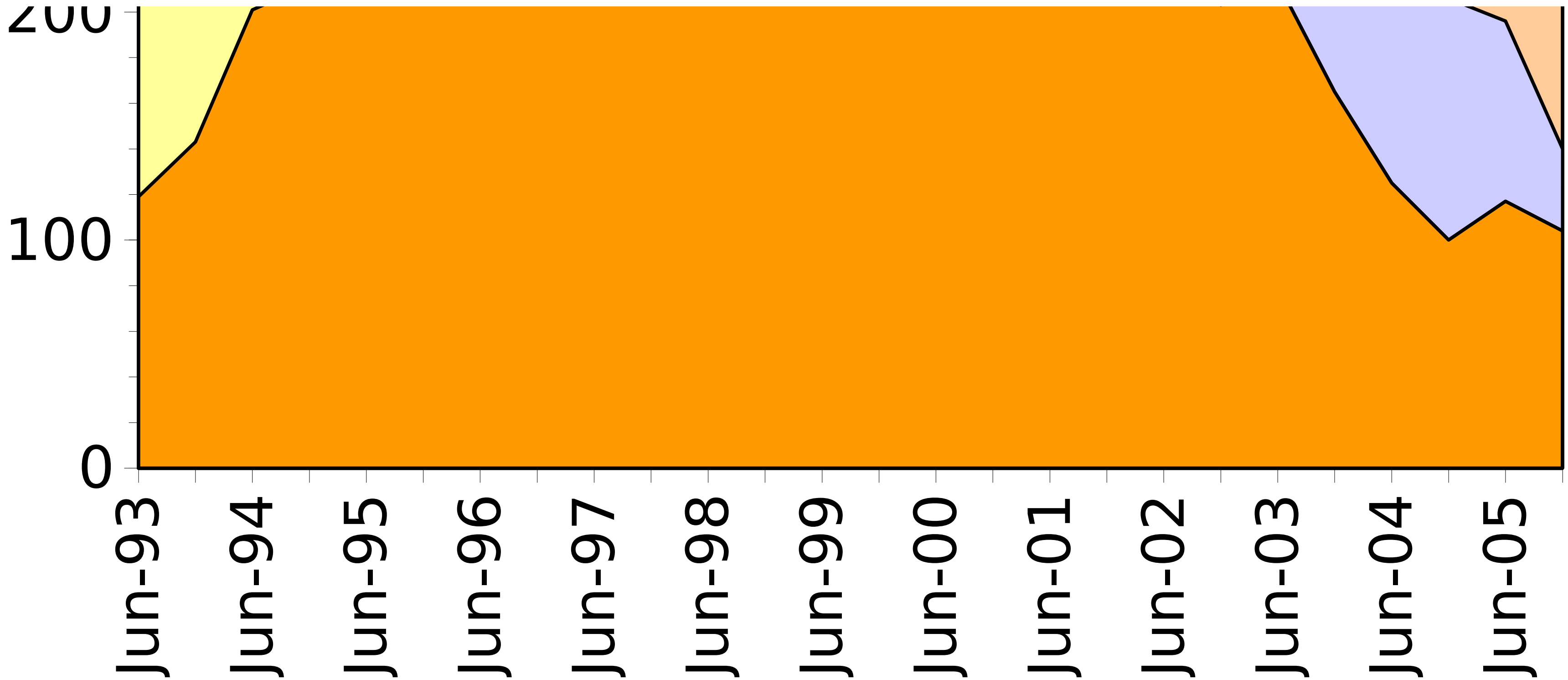
- Yes: We determined that a Pflops was feasible
- No: we thought it would take 5-7 years longer
- Yes: We didn’t think a new paradigm was needed
- No: we thought new technologies would be essential to complement (not replace) semiconductors
- Yes: We know silicon would continue on Moore’s law
- No: We figured 1 GHz clocks by 2007
- Yes: We considered off the shelf micros would be part of the equation
- No: We assumed custom architectures would dominate
- Yes: We identified the key software challenges
- No: We thought they would get fixed by now

Performance Projection

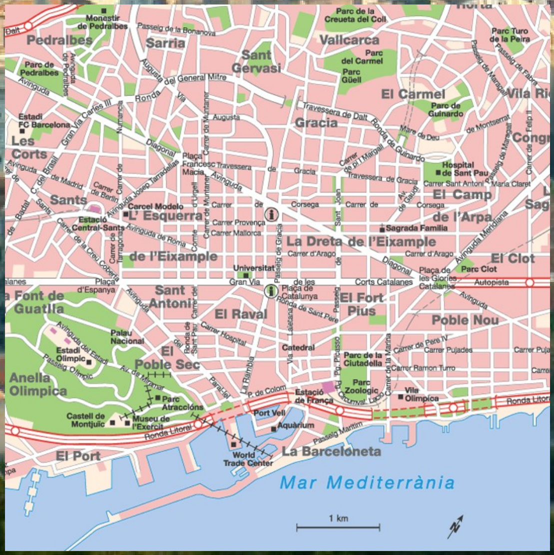
Earth Simulator and TSUBAME

Rank	Site	Computer	Processors	Year	R_{max}	R_{peak}
1	DOE/NNSA/LLNL United States	BlueGene/L - eServer Blue Gene Solution IBM	131072	2005	280600	367000
2	IBM Thomas J. Watson Research Center United States	BGW - eServer Blue Gene Solution IBM	40960	2005	91290	114688
3	DOE/NNSA/LLNL United States	ASC Purple - eServer pSeries p5 575 1.9 GHz IBM	12208	2006	75760	92781
4	NASA/Ames Research Center/NAS United States	Columbia - SGI Altix 1.5 GHz, Voltaire Infiniband SGI	10160	2004	51870	60960
5	Commissariat a l'Energie Atomique (CEA) France	Tera-10 - NovaScale 5160, Itanium2 1.6 GHz, Quadrics Bull SA	8704	2006	42900	55705.6
6	Sandia National Laboratories United States	Thunderbird - PowerEdge 1850, 3.6 GHz, Infiniband Dell	9024	2006	38270	64972.8
7	GSIC Center, Tokyo Institute of Technology Japan	TSUBAME Grid Cluster - Sun Fire X64 Cluster, Opteron 2.4/2.6 GHz, Infiniband NEC/Sun	10368	2006	38180	49868.8
8	Forschungszentrum Juelich (FZJ) Germany	JUBL - eServer Blue Gene Solution IBM	16384	2006	37330	45875
9	Sandia National Laboratories United States	Red Storm Cray XT3, 2.0 GHz Cray Inc.	10880	2005	36190	43520
10	The Earth Simulator Center Japan	Earth-Simulator NEC	5120	2002	35860	40960

Top-500 List: Architectures / Systems



Birth place of “Constellation” class Clusters – EuroPVMMPI’99

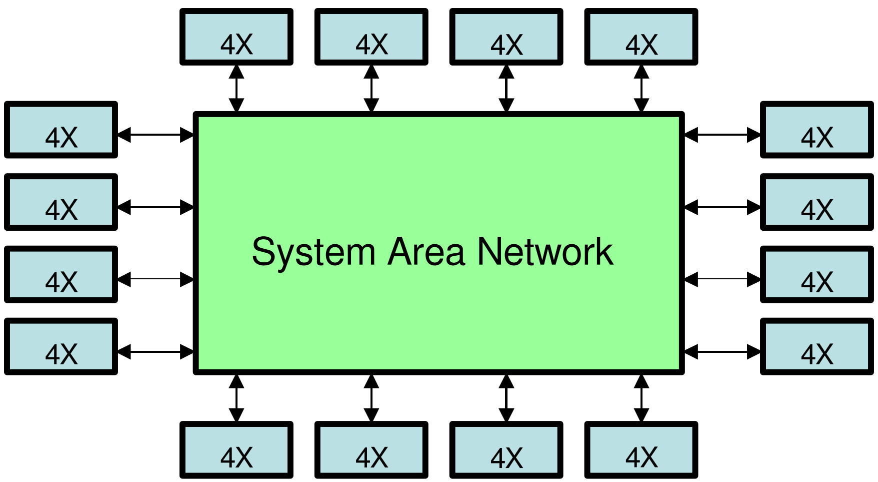


MareNostrum

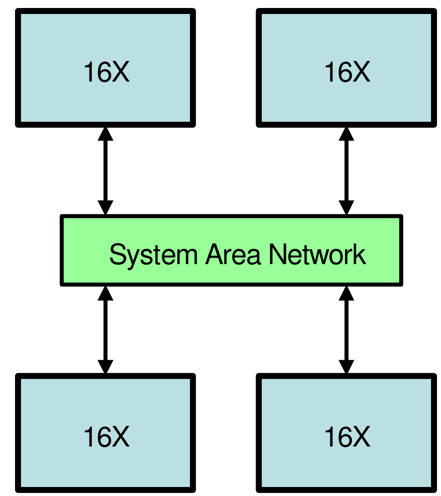
- One of the Largest Clusters in Europe
- Technical University of Catalonia
- IBM eServer BladeCenter JS20
- 31.4 Teraflops peak performance
- 2268 dual nodes
- PowerPC970 2.2 GHz
- Main memory 9 Terabytes
- Myrinet
- Linux

Commodity Clusters vs "Constellations"

- An ensemble of *N* nodes each comprising *p* computing elements
- The *p* elements are tightly bound shared memory (e.g., smp, dsm)
- The *N* nodes are loosely coupled, i.e., distributed memory
- *p* is greater than *N*
- Distinction is which layer gives us the most power through parallelism

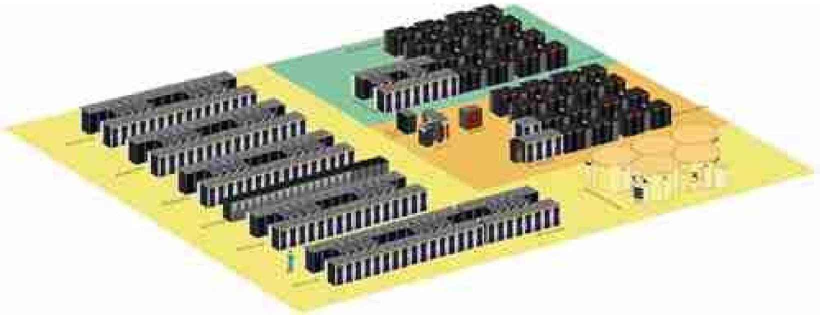


64 Processor Commodity Cluster



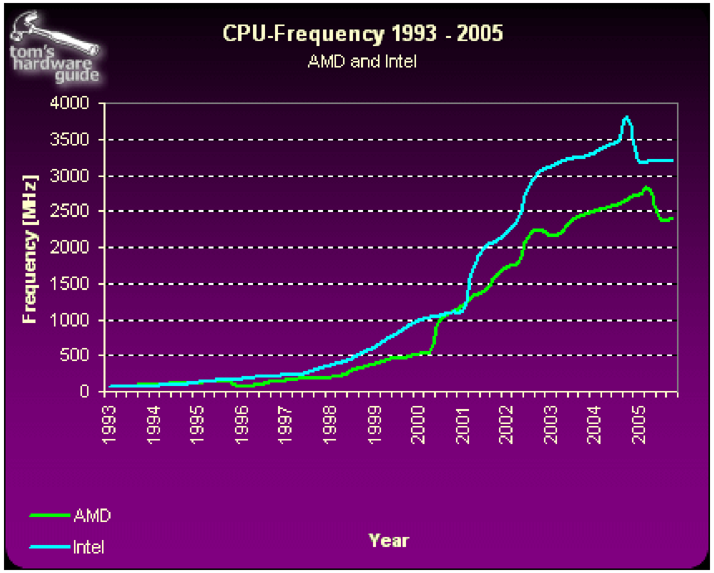
64 Processor Constellation

Columbia



- NASA's largest computer
- NASA Ames Research Center
- A Constellation
  - 20 nodes
  - SGI Altix 512 processor nodes
  - Total: 10,240 Intel Itanium-2 processors
- 400 Terabytes of RAID
- 2.5 Petabytes of silo farm tape storage

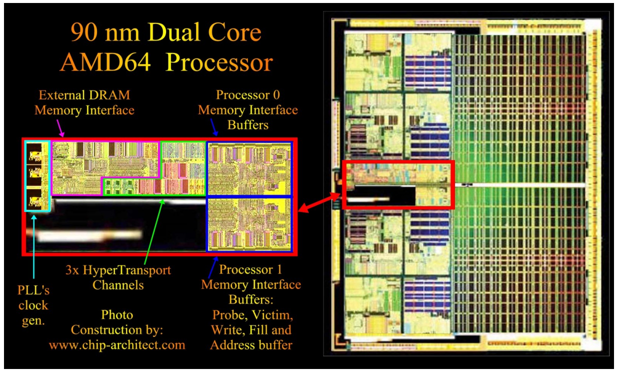
Driving Issues/Trends

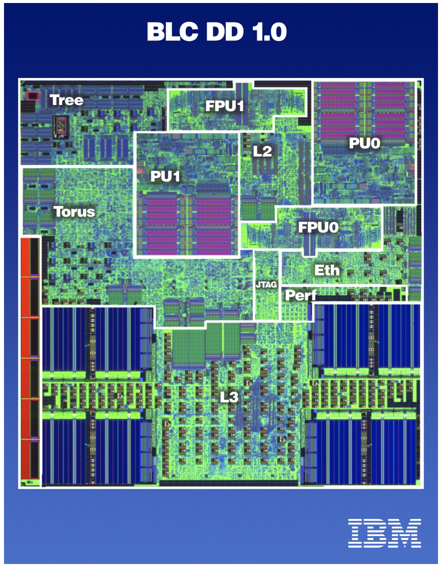


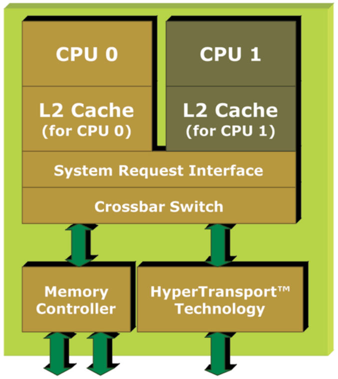
- Multicore
  - Now: 2
  - possibly 100's
  - will be million-way parallelism
- Heterogeneity
  - GPU
  - Clearspeed
  - Cell SPE
- Component I/O Pins
  - Off chip bandwidth not increasing with demand
    - Limited number of pins
    - Limited bandwidth per pin (pair)
  - Cache size per core may decline
  - Shared cache fragmentation
- System Interconnect
  - Node bandwidth not increasing proportionally to core demand
- Power
  - Mwatts at the high end = millions of Euros per year

Multi-Core

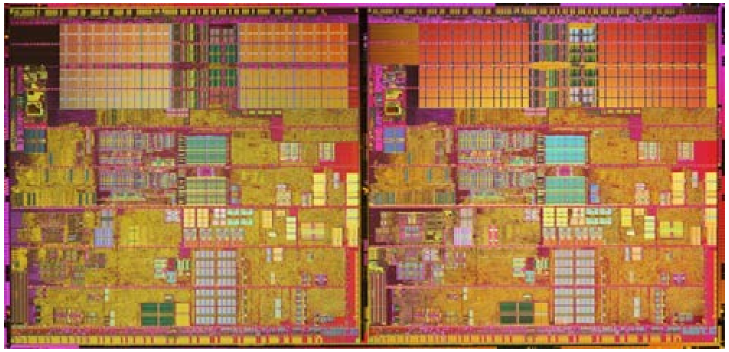
- Motivation for Multi-Core
  - Exploits increased feature-size and density
  - Increases functional units per chip (spatial efficiency)
  - Limits energy consumption per operation
  - Constrains growth in processor complexity
- Challenges resulting from multi-core
  - Relies on effective exploitation of multiple-thread parallelism
    - Need for parallel computing model and parallel programming model
  - Aggravates memory wall
    - Memory bandwidth
      - Way to get data out of memory banks
      - Way to get data into multi-core processor array
    - Memory latency
    - Fragments L3 cache
  - Pins become strangle point
    - Rate of pin growth projected to slow and flatten
    - Rate of bandwidth per pin (pair) projected to grow slowly
  - Requires mechanisms for efficient inter-processor coordination
    - Synchronization
    - Mutual exclusion
    - Context switching





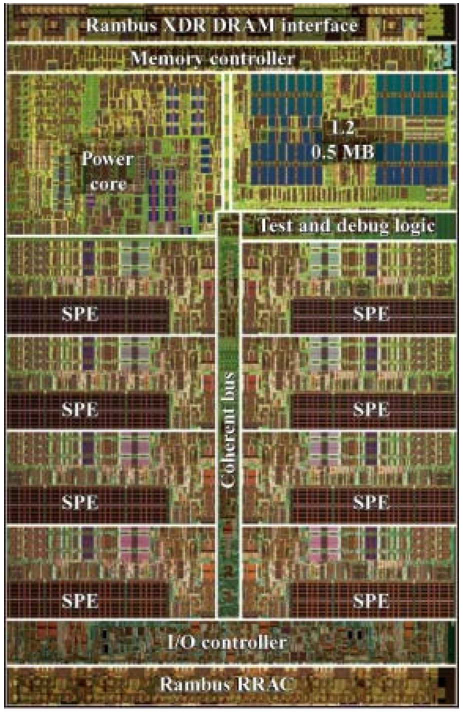


AMD Athlon™ 64 X2 Dual-Core Processor Design



Heterogeneous Architecture

- Combines different types of processors
  - Each optimized for a different operational modality
    - Performance > nX better than other n processor types
  - Synthesis favors superior performance
    - For complex computation exhibiting distinct modalities
- Conventional co-processors
  - Graphical processing units (GPU)
  - Network controllers (NIC)
  - Efforts underway to apply existing special purpose components to general applications
- Purpose-designed accelerators
  - Integrated to significantly speedup some critical aspect of one or more important classes of computation
  - IBM Cell architecture
  - ClearSpeed SIMD attached array processor



Implications, Ramifications, and Consequences

- *Constellations* will RULE!
  - Most parallelism in the nodes themselves
    - And did I mention multithreading, not much but some
  - By end of decade, perhaps
  - Starting at the middle and working up
  - Ultimately: Million processor (cores) systems
- Programming will be dominated by SMP/DSM nodes
  - NUMA is emerging on some vendor motherboards
  - Fragmented or disjoint programming?
    - Surely not MPI+OpenMP
    - Will we just ride MPI down to the cores and give up on SMP?
    - Or will MPI evolve to MPI-3?
      - One language to rule them all
- Linux will evolve
  - Adapt to multicore for the short term
  - Host something very different in the long term

Conventional Strategies to Address the Multi-Core Challenge

- Maintain status quo
  - Investment in current code stack
  - Investment in core design
- Increase L2/L3 cache size
  - Attempt to exploit existing temporal locality
- Increase chip I/O bandwidth
  - Reduce contention
  - Eventually embedded optical interfaces chip-to-chip
- Memory bandwidth aggregation through “weaver” chip
  - Balances processor data demand with memory supply rate
  - Enables and coordinates multiple overlapping memory banks
- Exploit job stream parallelism
  - Independent jobs
    - O/S scheduling
  - Concurrent parametric processes
    - Multiple instances of same job across parametric set
    - e.g., Condor
  - Coarse grain communicating sequential processes
    - Message passing; e.g., MPI
    - Barrier synchronization

Limitations of Conventional Incremental Approaches to MultiCore

- Its not just SMP on a chip
  - Cores on wrong side of the pins
  - Users expect to see performance gain on existing applications
- Highly sensitive to temporal locality
  - Fragile in the presence of memory latency
  - Uses up majority of chip area on caching
- Emphasizes ALU as precious resource
  - ALU low spatial cost
  - Memory bandwidth is pacing element for data intensive problems
- Low effective energy usage
  - Suffers from core complexity
- Does not address intrinsic problems of low efficiency
  - Just hoping to stay even with Moore's Law
  - Single digit sustained/peak performance
  - Bad when ALU is critical path element
- The Memory Wall is getting Worse!

What is required

- Global name spaces; both data and active tasks
- Rich parallelism semantics and granularity
  - Diversity of forms
  - Tremendous increase in amount
- Support for sparse data parallelism
- Latency hiding
- Low overhead mechanisms
  - Synchronization
  - Scheduling
- Affinity semantics
- Do not rely on:
  - Direct control of hardware mechanisms
  - Direct management and allocation of hardware resources
  - Direct choreographing of physical data and task locality

ParalleX: a Parallel Execution Model

- Exposes parallelism in diverse forms and granularities
  - Greatly increases available parallelism for speedup
  - Matches more algorithms
  - Exploits intrinsic parallelism of sparse data
- Exploits split transaction processing
  - Decouples computation and communication
  - Moves work to data, not just data to work
- Intrinsics for latency hiding
  - Multithreading
  - Message driven computation
- Efficient lightweight synchronization overhead
  - Register synchronization
  - *Futures* hardware support
  - Lightweight objects
  - Fine grain mutual exclusion
- Provides for global data and task name spaces
  - Efficient remote memory accesses (e.g. shmem)
  - Lightweight atomic memory operations
- Affinity attribute specifiers
  - Automatic locality management
  - Rapid load balancing

Split Phase Transactions

- A transaction is a set of interdependent actions on exchanged values
- Transactions are divided between successive phases
- All actions of a transaction phase are relatively local
  - Assigned to a given execution element
  - Operations perform on local state for low latency
- Phases are divided at stages of remote access or service request
  - Thus, asynchronous phasing at split
- No waiting for response to remote resources

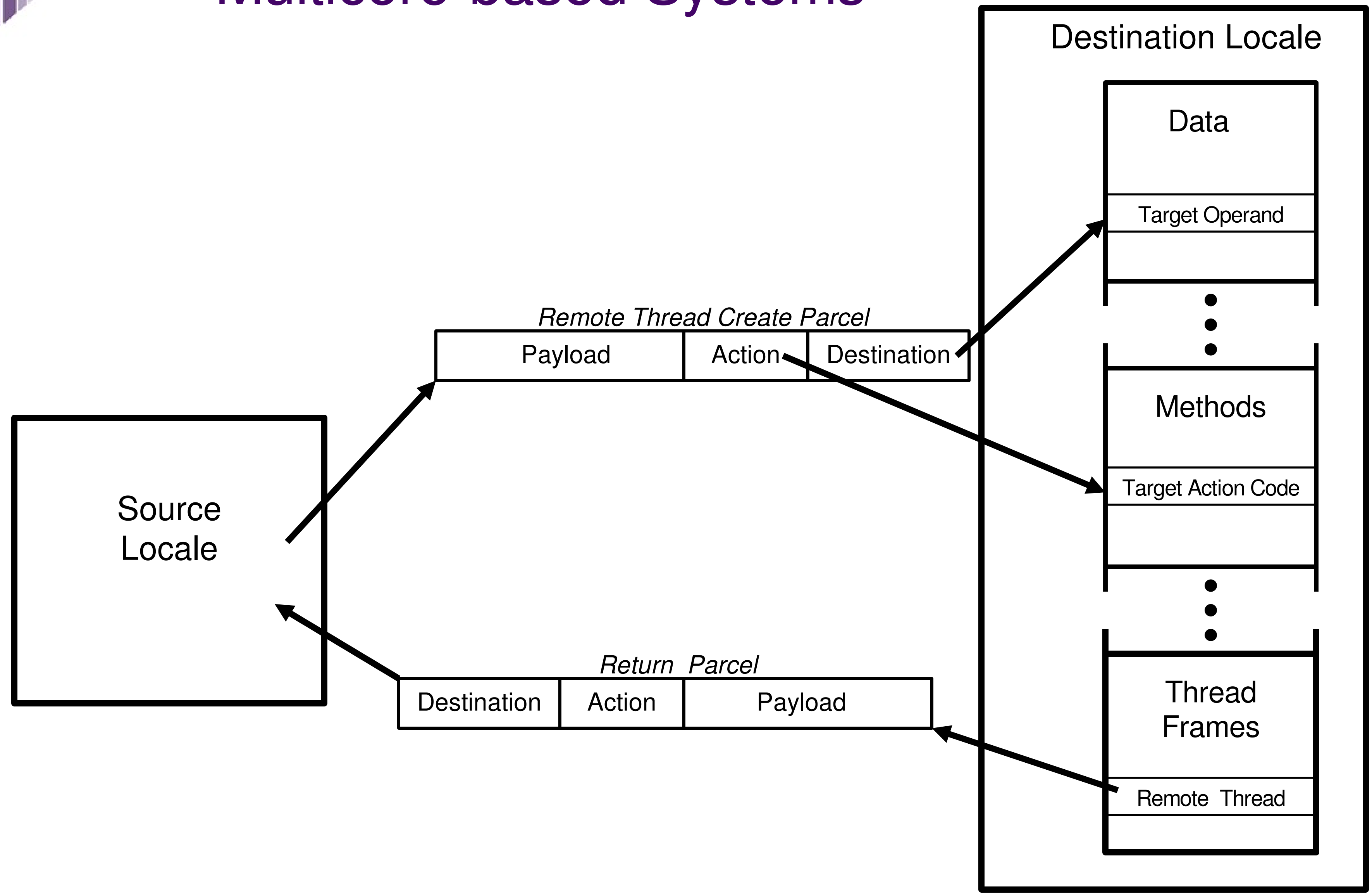
Localities

- A “locality” is a contiguous physical domain
- Guarantees compound atomic operations on local state
- Manages intra-locality latencies
- Exposes diverse temporal locality attributes
- Divides the world into synchronous and asynchronous
- System comprises a set of mutually exclusive, collectively exhaustive localities
- A first class object
- An attribute of other objects
- Heterogeneous
- Specific inalienable properties

Parcels

- Enables message-driven computation
- Messages that specify function to be performed on a named element
- Moves work and data between objects in different localities
- Parcels are not first-class objects
- Exists in the world of “parcel sets”
  - First-class objects
  - Transfer between parcel sets is atomic, invariant, and unobservable
- Major semantic content
  - Destination object
  - Action to be performed on targeted object
  - Operands for function to be performed
  - Continuation specifier

Parcels for Latency Hiding in Multicore-based Systems



Latency Hiding with Parcels

Idle Time with respect to Degree of Parallelism

Multi-Grain Multithreading

- Threads are collections of related operations that perform on locally shared data
- 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

Percolation Pre-Staging

- An important latency hiding and scheduling technique
- Overhead functions are not necessarily done optimally by high speed processors
- Moves data and task specification to local temporary storage of an execution element by external means
- Minimum overhead at execution site
- Almost no remote accesses
- Cycle: dispatch/prestage/execute/commit/control update
- High speed execution element operates on work queue
- Processors are dumb, memory is smart
- Good for accelerators, functional elements, precious resources

Fine-grain event driven synchronization: breaking the *barrier*

- A number of forms of synchronization are incorporated into the semantics
- Message-driven remote thread instantiation
- Lightweight objects
  - Data flow
  - Futures
- In-memory synchronization
  - Control state is in the name space of the machine
  - Producer-consumer in memory
    - e.g., empty/full bits
  - Local mutual exclusion protection
  - Synchronization mechanisms as well as state are presumed to be intrinsic to memory
- Directed trees and graphs
  - Low cost traversal

Global name space

- User variables
- Synchronization variables and objects
- Threads as first-class objects
- Moves virtual named elements in physical space
- Parcel sets
- Process
  - First class object
  - Specifies a broad task
  - Defines a distributed environment
    - Spans multiple localities
    - Need not be contiguous

Beyond current scope

- Policies not specified
  - Execution order
  - Language and language syntax
  - What's special about hardware
  - Runtime vs. OS responsibilities
  - Load balancing
- What's missing
  - Affinity, colocation
  - Fault intrinsics
  - Meta threads
  - I/O
  - Many details

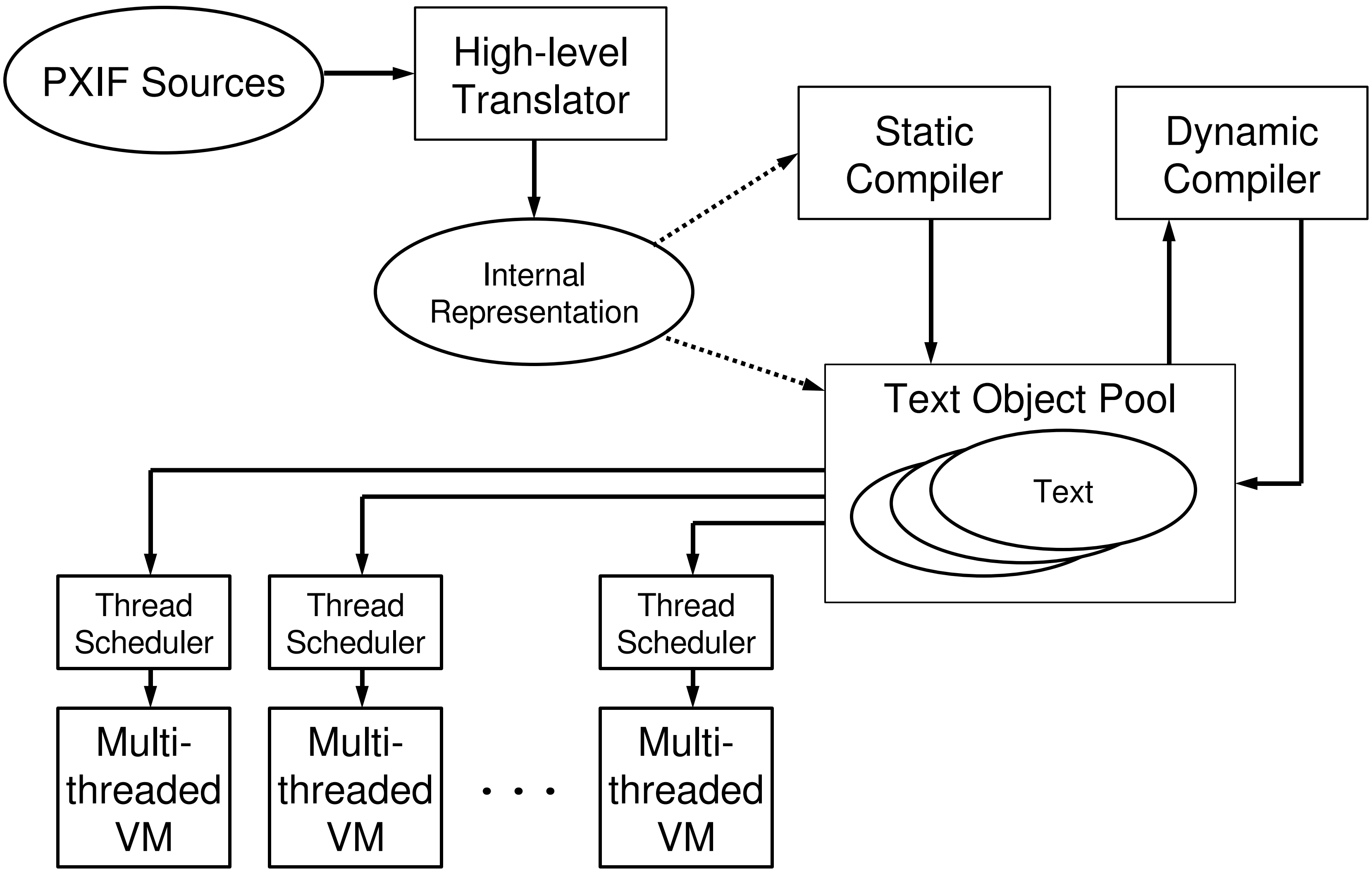
PXIF – ParalleX Intermediate-form

- Not a programming language
- Provides command line-like hooks to relate to and control all elements and actions of ParalleX execution
- Lists of actions and operands
  - ‘(<action> <target object> <function operands>)’
  - Some special forms (sadly)
- Create, delete, and move objects
- Invoke, terminate, migrate actions
- Syntaxless syntax
  - Currently uses a prefix notation but is amenable to other forms as long as one-on-one isomorphism
  - Note that MPI has more than one syntax
- In-work
  - Not finished
  - Subject to change
  - But great progress

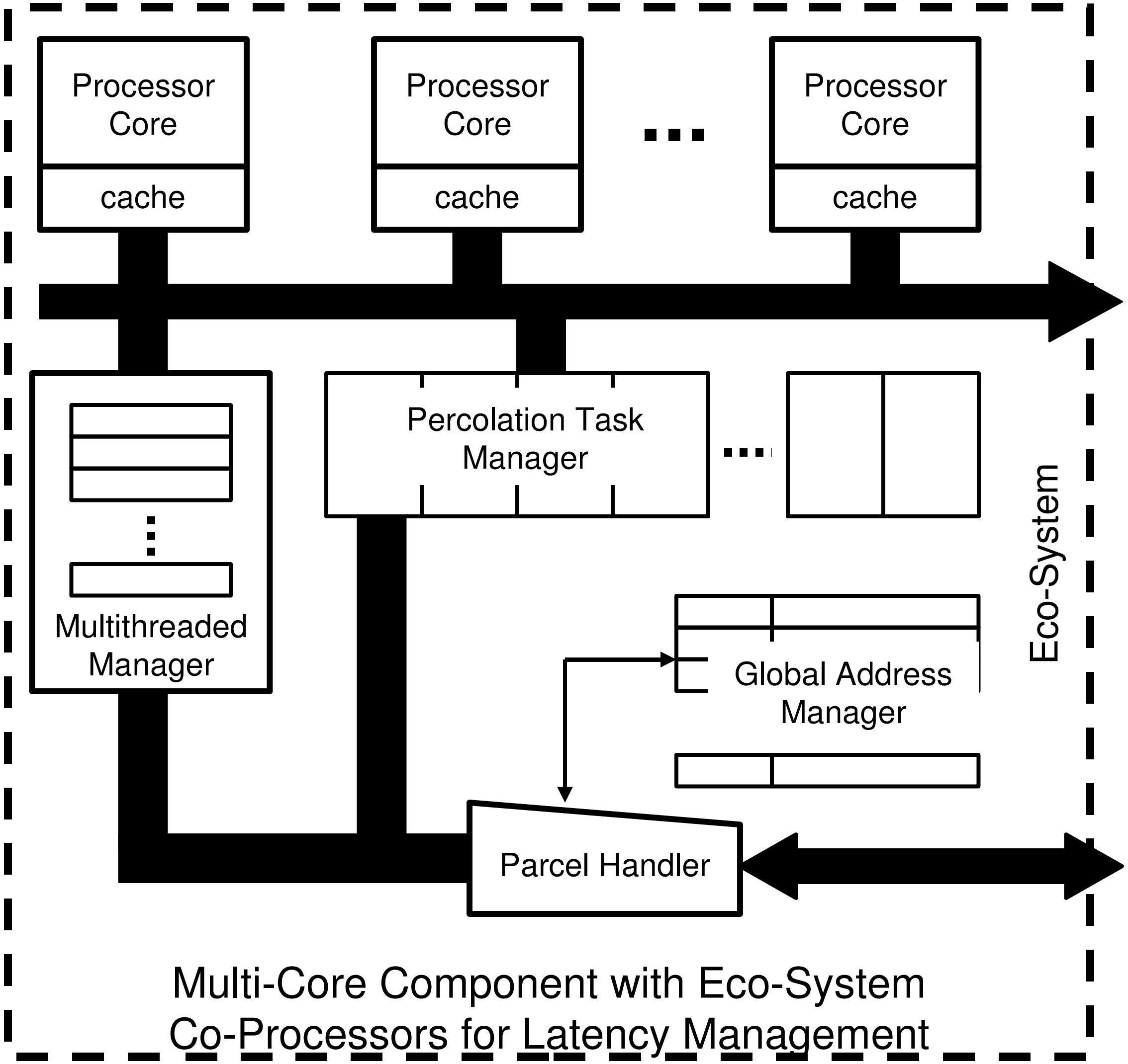
Reference Implementation

- Goal
  - Validation of semantics and PXI formulation
    - Correctness
    - Completeness
  - Early testbed for experimentation and algorithm development
  - Executable reference for future PXI implementations by external collaborators
- Strategy
  - Facilitates development of PXIF syntax specification
  - Employ rapid prototyping software development environment
  - Incremental design
    - Replace existing functions with PXI-specific modules
    - Refinement of ParalleX concepts and PXIF formalism

PXIF from Sources to Execution



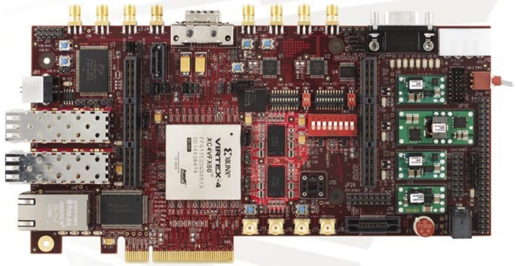
Can't Change the Architecture but if I could:

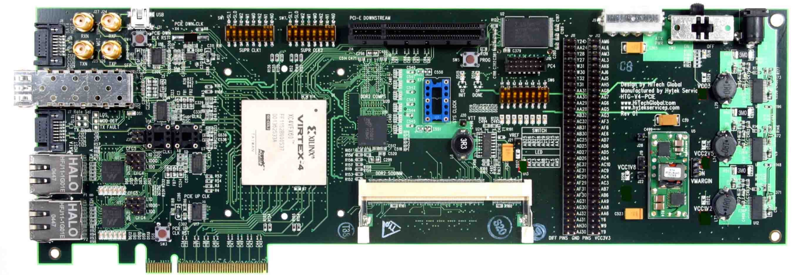


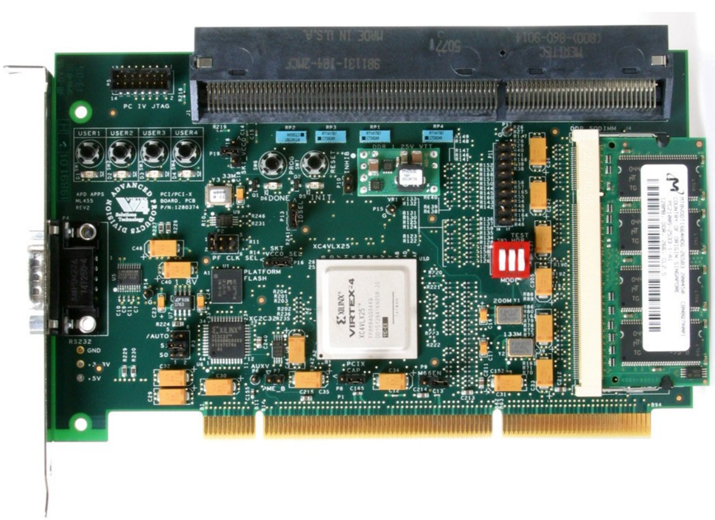
But there may be a way …And I am psyched!

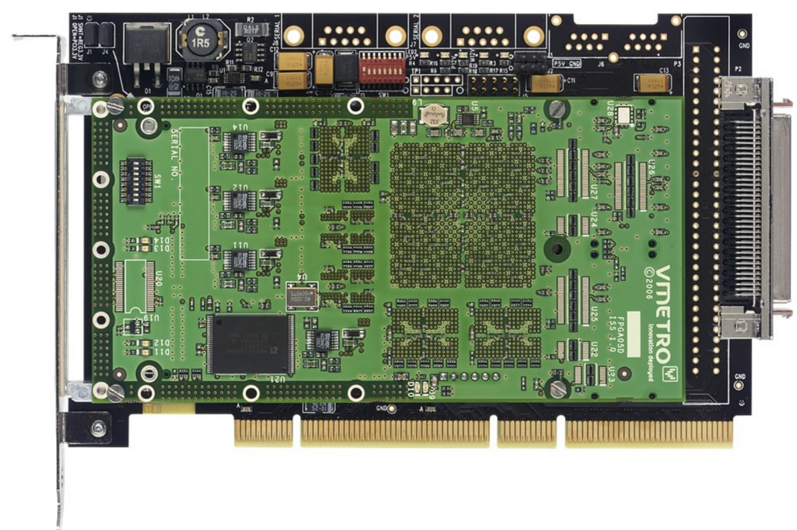
- Cluster implementers and users are scum sucking bottom feeders
  - We take what we can get and do what we can
  - Has been a good strategy for many purposes for a decade
- We have not controlled architecture
  - Although we now control much of the software stack
  - Otherwise we’ve adopted available components
- But there may be a way and I am psyched!
  - A new line of products incorporating FPGA technology
  - Integrated with conventional nodes via industry standard interfaces

FPGA attached boards: a new opportunity for advanced execution models









FPGA ParalleX Accelerator

- Based on prior work performed on MIND architecture as part of Caltech/JPL Gilgamesh project
- Goal: enhance scalability and efficiency
  - Hide system wide latency
  - Reduce parallelism control overhead
- Design FPGA-based hardware drivers and co-processors to support ParalleX model
  - Parcel message-driven computation handler
  - Medium grained multithread execution scheduler
  - Global address translation support
  - Percolation pre-staging task manager
  - (possibly) local control object synchronization acceleration

The Changing Cluster Agenda

- 1994 – would they work, could they be useful
- 1997 – could we build & program them to be practical
- 2000 – will they scale & can we manage them reliably
- 2003 – can we win
- 2004 – world domination
  - we inherit all the problems of HEC parallel computing that were not solved by previous generations of systems
- Today –
  - As always, track rapidly advancing technology
  - Harness multicore and heterogeneous components
  - Advance execution and programming models and methods to address scalability, programmability, and power within the domain of increasing system complexity