How I Learned to Stop Worrying and Love New Models of Computation

Richard C. Murphy
Scalable Computer Architectures Department
Sandia National Laboratories
Affiliated Faculty, New Mexico State University

July 27, 2011

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

A Computer Architect's View of the World

The trouble with programmers is that you can never tell what a programmer is doing until it's too late.

- Seymour Cray

A Computer Architect's View of the World

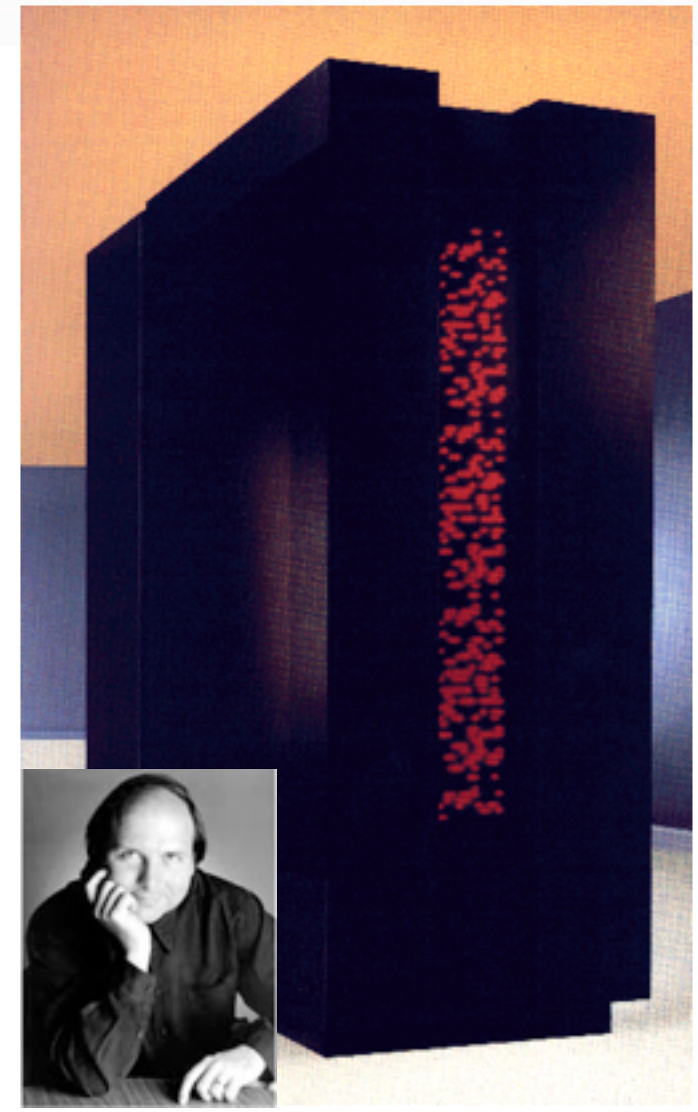
The trouble with programmers is that you can never tell what a programmer is doing until it's too late.

- Seymour Cray

OR

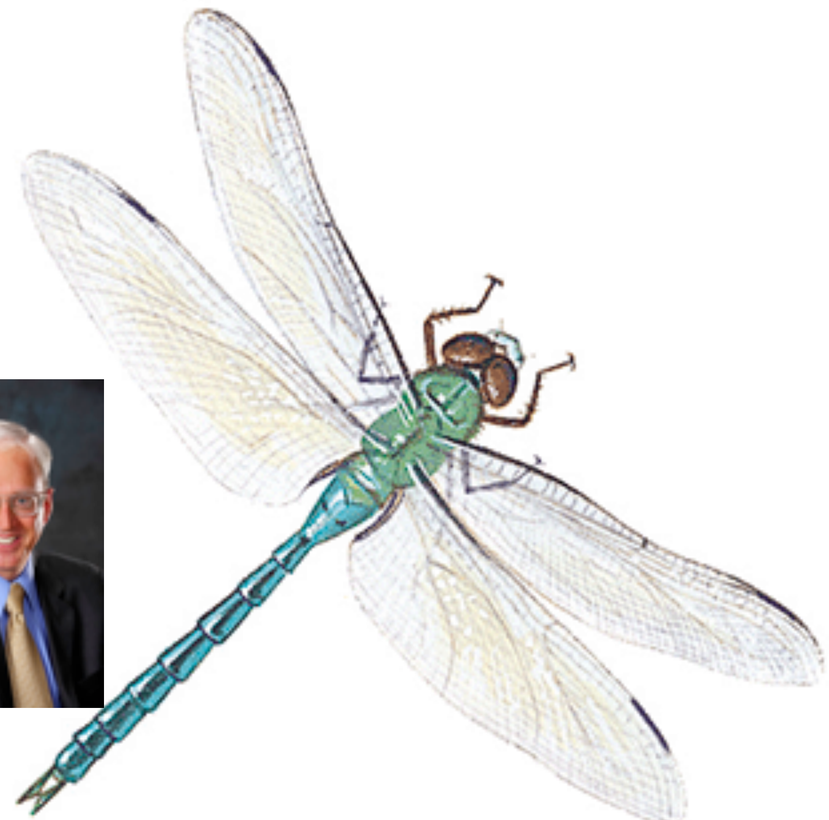
A(B(I)) = C(D(I))

DOE Has Over-constrained the Exascale Problem



Step 1: Choose your favorite MIT Alumni's Computer from the early 1990s as your compute node... errr swim lane

DOE Has Over-constrained the Exascale Problem

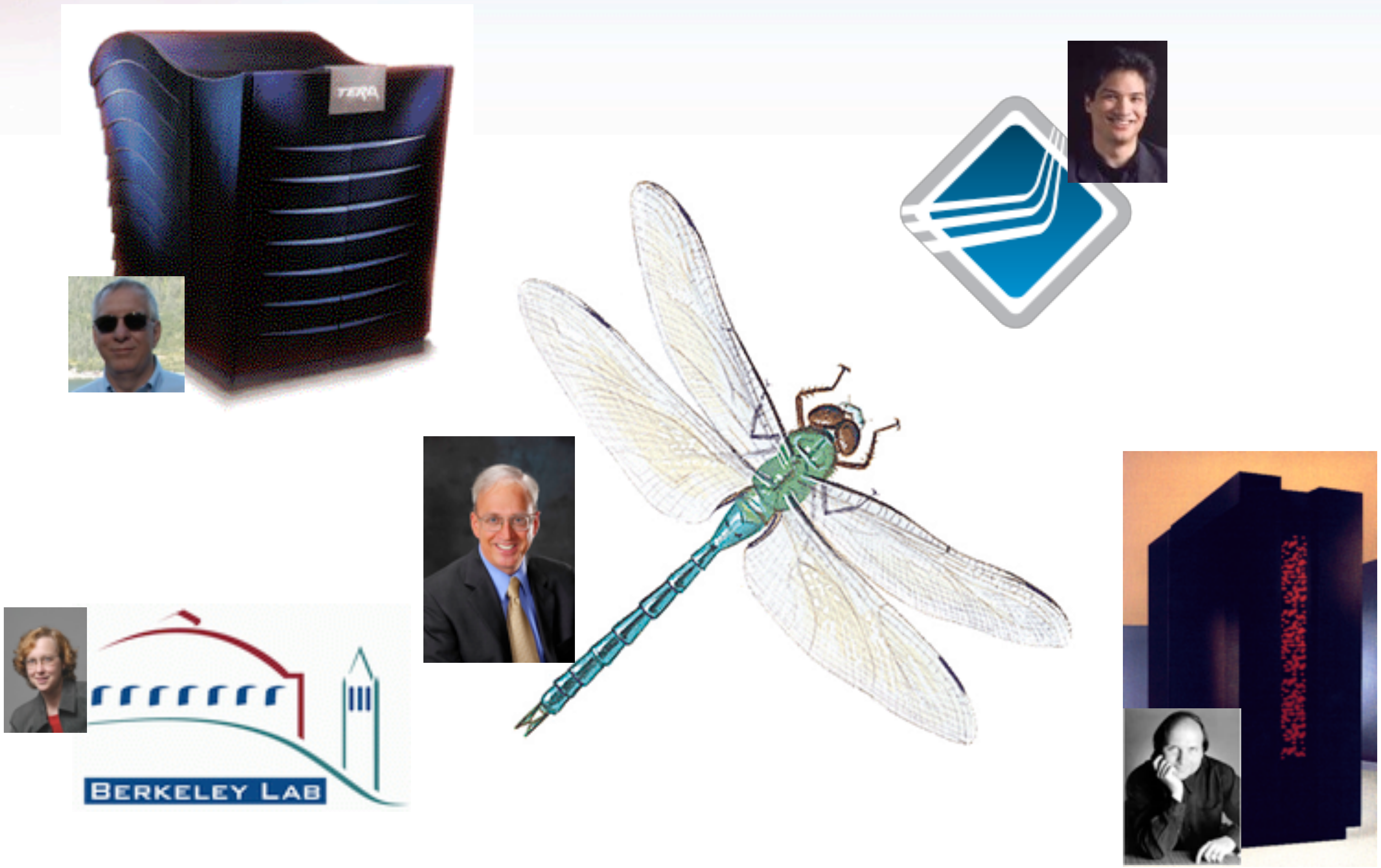


Step 2: Pick your favorite MIT Alumni's network topology and wire up whatever bandwidth you think you can afford

NNSA National Nuclear Security Administration

Sandia National Laboratories

DOE Has Over-constrained the Exascale Problem



Step 3: Mix in MPI... and a touch of your favorite MIT Alumni's Alternative Programming Model

NNSA
National Nuclear Security Administration

Sandia
National
Laboratories

DOE Has Over-constrained the Exascale Problem

We all know this approach is subject to criticism...
(I personally think it's a disaster)

Impact

- This results in a tremendously sub-par platform, capable (if you're lucky) of running LINPACK in the power envelope
- This eliminates the tight coupling between processor, memory, and network necessary to do codesign
  - How else can one do a system-wide power optimization?

- A new approach:
  - Codesign has to begin somewhere, either from the bottom-up (technology) or the top-down (applications)
  - Programming Methods allow us to introduce concrete parallel patterns for codesign
    - How well does it match the application writer's needs?
    - Can the hardware implement it in a performant, energy-efficient fashion?

NNSA National Nuclear Security Administration

Sandia National Laboratories

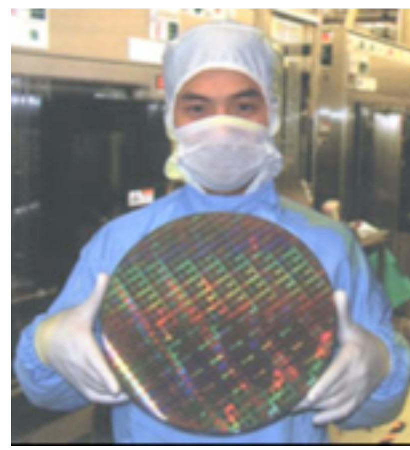
Everything Old is New Again (Vintage Computing?)

- Early Petaflops Effort (1996-1999)
  - NSF, DARPA, NASA, NSA
  - DOE stayed out because the mission need could be met with commodity (but we're paying the price now)
- One of 8 NSF-sponsored petaflops design points in a 6 month study
- We were able to get to petascale a decade later
  - Without addressing the fundamental energy issues
  - Without programming model innovation, which we know we need
  - Without broad agreement between government agencies
- Consider the power envelopes:
  - 2007 HTMT Design Point: 2.4 MW
    - Scaled (unfairly) by Moore's Law: < 1.2MW
  - 2008 Road Runner PF/s: 2.4 MW
  - 2008 Jaguar PF/s: 7 MW

Key concepts from HTMT drive today's Exascale research agenda (threads, message-driven computation, global shared memory)

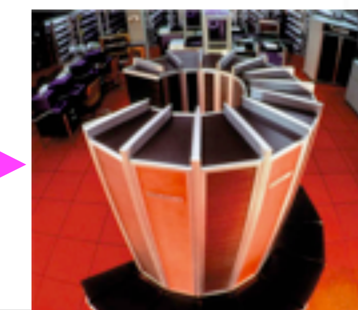
This is no time for Ease and Comfort...
It is time to Dare and Endure

Feature Freeze for the Processor
(We're done giving input!)



Window of Opportunity

Move to production and FCS



Today 2012 2013 2014 2015 2016 2017 2018 2019 2020

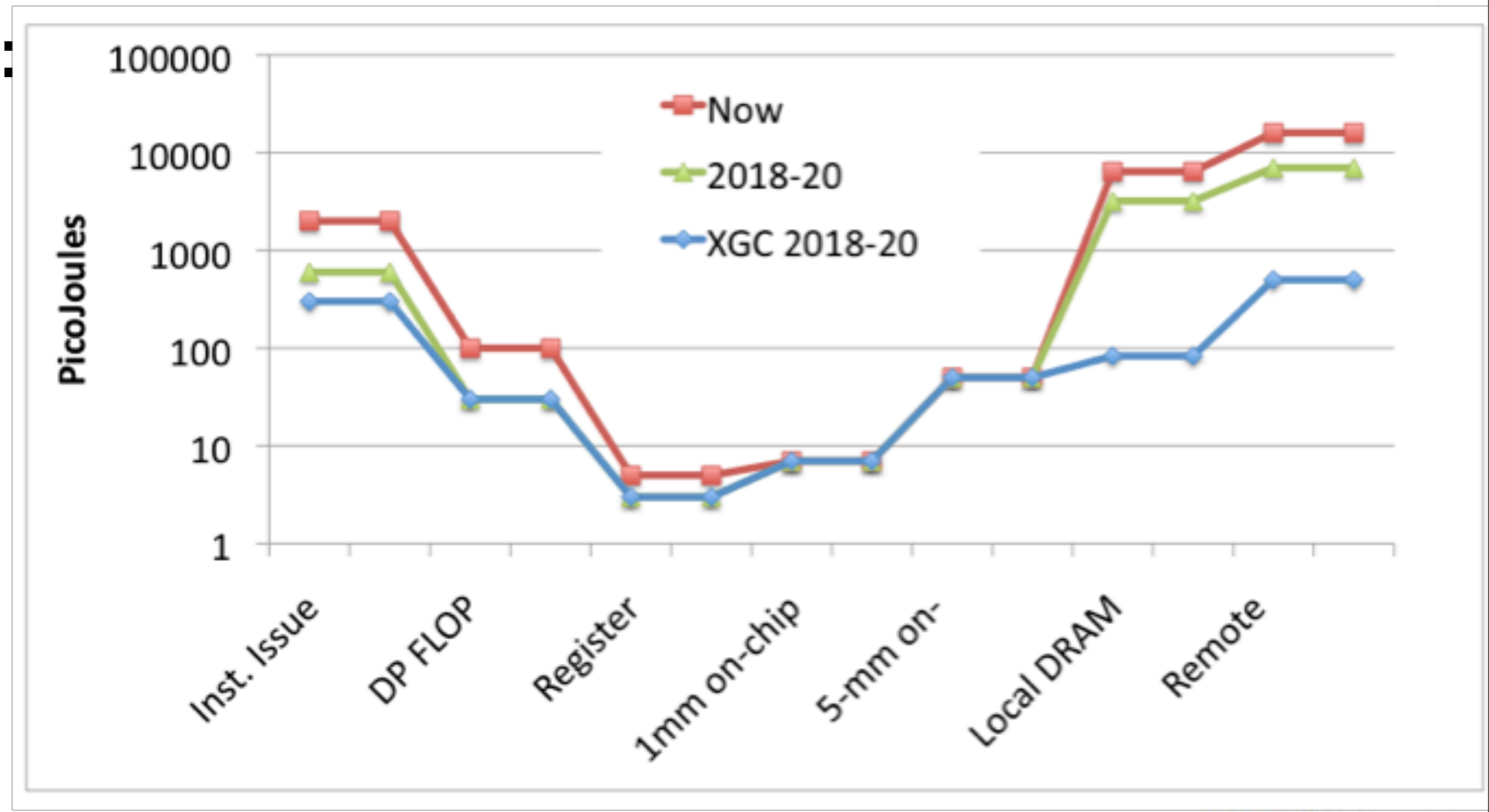
Who Endures?

NNSA National Nuclear Security Administration

Sandia National Laboratories

Given this grim picture...

- Anything you want from a programming model needs to appear today in:
  - –Chapel, X10, Fortress
  - –SHMEM, GASnet, UPC/CoArray Fortran
  - –Maybe MapReduce or some other *business* thing
- The problem is:



Chapel

- Background: Brad Chamberlain, Dave Callahan, and Burton Smith's view of how to program a petascale system
  - Burton was fundamentally right about the architecture (see my first slide) and the challenges of interconnecting them
  - Thinking that's decades ahead of the "multicore" community
- What does Chapel express (and how can it migrate into Fortran)?
  - Tasks (with threads as an execution vehicle)
  - Data-oriented synchronization (sorely missing from today's CPUs)
  - Rich arrays (multidimensional, strided, sparse, associative, unstructured)
    - User-specified layout and distribution
    - NOTE: the execution model has to understand more about data structures to meet energy/performance goals
  - Data-parallel constructs (forall, promotion of scalars, etc.) that are compatible with tasks ("multiresolution programming")

What about Fortress and X10?

- X10
  - There's something in Phasers that Vivek Sarkar can explain (and is beyond a hardware simpleton like me)
  - The name alone should establish it's importance
  - I'm not sure why we're doing synchronization for the sake of synchronization

- Fortress
  - The object model is interesting
  - Frosting
    - LaTeX style rendering
    - Support for units
    - Etc.
  - Probably more productivity oriented

What’s important from the hardware perspective?

- **Execution Model**
  - **We implement primitives, not high-level constructs in hardware**
  - **What is it?**
    - **Jack Dennis called it “an API for the machine” at the PRMHTS-2 workshop in April**
    - **This is insufficient, the current execution model is not “MPI”**
    - **It must also include *how the machine is programmed* (or a performance model?) AND a Memory Model**
      - **Today that’s MPI + BSP (+ TSO)**
  - **Five Elements**
    - **Concurrency**
    - **Coordnation**
    - **Movement (of data, of work)**
    - **Naming**
    - **Introspection**

What's the X in MPI+X?

- Again, too constrained a question... it should be "what comes next as a superset of MPI"?
- Nothing that exists today properly approximates "X"
- It's easier to point out out what X is not
  - We absolutely have to break BSP to make the energy numbers work out
- We have some ideas, but we have to apply codesign to figure it out... we're taking the following approach
  - Proffer an execution model (ParalleX)
  - Perform application, programming methods, architecture analysis of ParalleX
  - Use the key metrics of performance (in time and energy) and programmability to evaluate ParalleX
  - Repeat in a codesign loop (while concurrently optimizing across the stack)

NNSA National Nuclear Security Administration

Sandia National Laboratories

ParalleX

Element	ParalleX	GPUs	Stylized CSP	PGAS
Concurrency	Threads/Codelets	SIMD/lock-step threads	Ranks/ Processes	Processes
Coordination	Lightweight Control Objects (fine-grained)	Local Memory/ Explicit	BSP	BSP
Movement	of Work: Parcels of Data: PGAS and Bulk	Bulk Data Transfer (weak memory system)	Bulk Data Transfer	Data Only (load + store)
Naming	Global Name Space Global Address Space	Global Address Space	Explicit by Rank	Global Address Space
Introspection and Adaptivity	System Knowledge Graph/Dynamic	None/Static	None/Static	None/Static

Where does it depart from the roadmap?

Element	ParalleX	GPUs	MPI + BSP	PGAS + BSP
Concurrency	Threads/Codelets	SIMD/lock-step threads	Ranks/ Processes	Processes
Coordination	Lightweight Control Objects (fine-grained)	Local Memory/ Explicit	BSP	BSP
Movement	of Work: Parcels of Data: PGAS and Bulk	Bulk Data Transfer (weak memory system)	Bulk Data Transfer	Data Only (load + store)
Naming	Global Name Space Global Address Space	Global Address Space	Explicit by Rank	Global Address Space
Introspection and Adaptivity	System Knowledge Graph/Dynamic	None/Static	None/Static	None/Static

Some Problematic Attitudes from People I Deeply Respect

“We only have to worry about the on-node programming model for exascale...”

- If I have to break the BSP model for hardware reasons your local problem just became a global problem
- As Culler observed in 1992:
  - “a high performance network is required to minimize the communication time and it sits 90% idle [to achieve a high computation to communication ratio]”
  - How do we smooth this out in the most energy efficient manor?
- Today’s model may allow principally local “thinking” but at the cost of energy which we can no longer afford
  - Worse, the long distance links we’re talking about are the most energy expensive in the system
  - This may be THE optimization problem for exascale
- What are the right processor/memory/NIC interactions?

“My physics code works great in the current model”

- We run the codes that the platforms support
  - What science are we missing out on because we can’t run the algorithms in an MPI+BSP model? Is it important?
- Good example: Graphs
  - Fundamental to mathematics
  - Potentially represent a much larger market than all of HPC (business analytics, medical informatics, cybersecurity)
  - Don’t run well today on any computer, even for simple analytics
  - Google is based on what you can do with a graph using MapReduce
    - Surely the lamest of programming models
    - Very limited in the kinds of analytics (PageRank)
    - Worth billions of dollars
  - Jackie Chen

Our job in HPC is to be at the leading edge, not the trailing edge!

“Why are you experimenting with all those failed (DoD) programming models like work moving and ActiveMessages?”

- Most Active Messages implementations are too limiting
  - Things changed since Culler’s 1992 paper and we likely want to create threads upon the receipt of an AM rather than “integrate the code into the running computation”
- Maybe the J-machine WAS hard to program, but we’ve got two decades of improved compilers and hardware
- Low-cost thread creation from the NIC (or better, work-queue based processors) solves a critical MPI problem
- Making processors more J-machine like will fix several commercial threading problems
  - The work queue model already exists in many OS and runtime interfaces (TBB, qthreads, etc.)
  - The runtime needs the flexibility to manage parallelism for energy efficiency

Tighter NIC/Processor Integration is CRITICAL for energy efficiency, but do we know how to program it?

“Why Failed Models (continued)?”

- Analysis of SNL Physics and Informatics Applications shows
  - 1-2 orders of magnitude more concurrency exposed by work moving
  - 5-400x improvement in data movement over the application suite
    - THIS IS WHERE YOUR ENERGY GOES!
  - Reduced thread state size (15% of a modern register file)
- Significant room for data partitioning improvement
  - 5-10x reduction in energy possible vs. programmer partitionings
    - (Today’s partitionings look more like random)
- Cheap synchronization required to make it work
  - Today that synchronization happens in the (power hungry) reservation stations of a modern processor
  - It needs to be programmer exposed and processor controlled
- Perfect example of how programming methods work impacts architecture!

"You want new... we've got PGAS... isn't that sufficient?"

- PGAS isn't new, it's actually quite mature with many implementations
- Which PGAS are you talking about (CoArrays, UPC, etc.)?
  - Each of them brings a slightly different and nuanced set of thinking
  - Are you using an ActiveMessages based implementation or something else?
  - What kind of something else (they're non-standard)?
  - What EXACTLY should we support in hardware?
- As far as I know there's been no analysis of PGAS vs. MPI in terms of energy-performance, and our time-performance measures are at best anecdotal
- PGAS also isn't the focus of Chapel, which is where you'll get your "new" stuff

“We know how to do data movement well, you should base your architecture on that”

- No, we don’t... but we might if we got the processor to let us hook in at the right place
- How do you define well?
  - Yes, we support BSP applications
  - No, we are by no means energy-efficient (which is THE defining problem for exascale)
  - Large data transfers force us into the BSP model, whereas small data transfers promote asynchrony
- What we need for small messages (MPI, PGAS, work moving, etc.) is the same from the network
  - Higher message rates
  - Tighter integration between processor, memory, and NIC
  - Work sharing between the NIC, Processor, and Memory
    - Do the operation at the lowest energy place
  - LOTS more bandwidth than exists on the roadmap

Commodity Nodes are More Cost Effective

- Really? Power costs \$1M/Megawatt-Year
  - –\$100-\$250M to buy the machine
  - –At 125 MW in 2018, \$625M for power (not including cooling, etc.)
  - –And we probably want 2-3 of them: \$1.7-\$2.7B
- So what's that worth?
  - –If you can reduce power by 50% for a single machine for less than \$300M, it's probably worth it!
  - –If you can reduce power by 75% for \$450M, it's probably worth it!
- We've never built a commodity node for a capability machine (this is DOE mythology)
  - –At Sandia: nCube, Intel Paragon, ASCI Red, Red Storm
  - –At LLNL: BlueGene (full custom except for the core!)
  - –At LANL: RoadRunner

NNSA National Nuclear Security Administration

Sandia National Laboratories

Worldwide Impact

"Total power used by servers [in 2005] represented ... an amount comparable to that for color televisions. "

-ESTIMATING TOTAL POWER CONSUMPTION BY SERVERS IN THE U.S. AND THE WORLD, Jonathan G. Koomey

3741e9 KW-Hrs	Total US power consumption
* 3-4%	used by computers (>2% servers, >1% household computer use)
= 112 - 150e9 KW-Hrs	US Computer power consumption
* $0.1 $/KW-Hr	Retail cost, US Average 2009
= $11 - $15	Billion US$ in compute power
* 3-5	in 2005 US was roughly 1/3 of servers, by power. This has probably decreased
= $33 - $75	Billion US$ in worldwide computer power
=  - 	Yearly GDP of Qatar to Burma

Worldwide Impact

"Total power used by servers [in 2005] represented ... an amount comparable to that for color televisions. "

-ESTIMATING TOTAL POWER CONSUMPTION BY SERVERS IN THE U.S. AND THE WORLD, Jonathan G. Koomey

3741e9 KW-Hrs	Total US power consumption
* 3-4%	used by computers (>2% servers, >1% household computer use)
= 112 - 150e9 KW-Hrs	US Computer power consumption
* $0.1 $/KW-Hr	Retail cost, US Average 2009
= $11 - $15	Billion US$ in compute power
* 3-5	in 2005 US was roughly 1/3 of servers, by power.
= $33 ( ) - $75 ( )	Billion US$ in worldwide computer power
* 15-35%	DRAM memory power
= $5 ( ) - $25 ( )	BIllion in US$ in DRAM power

Conclusions

- Being wimps about actually taking on the risk necessary to build a meaningful exascale platform does not benefit:
  - National Competitiveness
  - National Security
  - DOE or other government agencies
  - Anybody's application base
- We need a tightly integrated processor, NIC, memory model
  - And a programming model that takes advantage of that
- We need to get started NOW, especially on the software part

NNSA National Nuclear Security Administration

Sandia National Laboratories

A Call to Action

- We need to focus on execution models and exposing low-energy (and energy-aware) capabilities to the programmer
- We need to manage massively more parallelism
  - 3-orders of magnitude just to get to exascale
  - (by my estimate) 2-orders of magnitude more to do what we do today at lower energy
- Mechanisms need to be fine-grained
- There are three major hurdles:
  - We have threads, but they don't interoperate well (fix synchronization)
  - Today's message rates are insanely low
  - BSP may occupy the memory system 100% of the time, but has spiky processor and NIC properties (bad for energy)

NNSA National Nuclear Security Administration

Sandia National Laboratories

Thank you!