The Future of Application Development and Exascale Computing

Thomas Sterling
Professor of Informatics and Computing, Indiana University
Chief Scientist and Associate Director
Center for Research in Extreme Scale Technologies (CREST)
Pervasive Technology Institute at Indiana University

February 27, 2014
Preamble:

• 1988 – A Speech by the Visionary
  – Supercomputing Conference (SC88) in Orlando, Florida
  – Seymour Cray gives Keynote Address

• Cray lays out roadmap for future of supercomputing
  – Technologies
  – Architectures
  – Timeline

• He was wrong about everything

• Cray never sold another machine

• 1996 – he dies in a car accident
1997 – the future is established

- ASCI Red – first Teraflops System, based on VLSI microprocessors
- NOW – the first cluster on the Top-500 list
- Beowulf – the first cluster to be awarded the Gordon Bell Prize
What Cray Got Right/Wrong

• Right
  – Father of (modern) supercomputing
  – Understood factors of performance degradation
    • Starvation
    • Latency
    • Overhead
    • Waiting for contention

• Wrong
  – Bet on the wrong technology
    • Assumed fast technology meant best supercomputing technology
    • Did not extrapolate technology projections correctly
  – Missed importance of density over raw clock
  – Missed importance of cost; did not value mass market
  – Time to market – mom&pop operation, cottage industry
Tianhe-2: Half-way to Exascale

- **China, 2013: the 30 PetaFLOPS dragon**
- Developed in cooperation between NUDT and Inspur for National Supercomputer Center in Guangzhou
- Peak performance of 54.9 PFLOPS
  - 16,000 nodes contain 32,000 Xeon Ivy Bridge processors and 48,000 Xeon Phi accelerators totaling 3,120,000 cores
  - 162 cabinets in 720m² footprint
  - Total 1.404 PB memory (88GB per node)
  - Each Xeon Phi board utilizes 57 cores for aggregate 1.003 TFLOPS at 1.1GHz clock
  - Proprietary TH Express-2 interconnect (fat tree with thirteen 576-port switches)
  - 12.4 PB parallel storage system
  - 17.6MW power consumption under load; 24MW including (water) cooling
  - 4096 SPARC V9 based Galaxy FT-1500 processors in front-end system

- Cosmology and Astronomy
- Weather and Climate
- Energy and Combustion
- Aerospace and Auto Crash
- Nuclear Reactors and Controlled Fusion
- Biology and Molecular Dynamics
- Medical and Drug Design
- Visualization and Entertainment
- Electronic Technology and Design
- Manufacturing and Distribution
- Chemistry
- Genomics
- Financial
- Crypto-Analysis and IDing
Progress in Magnetic Fusion Energy (MFE) Research

Data from Tokamak Experiments Worldwide

- TFTR (U.S.)
- JET (EUROPE)
- ITER

Fusion Power

<table>
<thead>
<tr>
<th>Kilowatts</th>
<th>Watts</th>
<th>Milliwatts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,000</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>100</td>
<td>10</td>
<td>1</td>
</tr>
</tbody>
</table>

D^+ + T^+ → 'He'' (3.5 MeV) + n (14.1 MeV)

Energy Multiplication About 450:1

Plasma self-heating

Deuterium-Tritium Fusion Reaction


Years

Courtesy of Bill Tang, Princeton

Center for Research in Extreme Scale Technologies
Indiana University
Pervasive Technology Institute
Elements of an MFE Integrated Model ➔ Complex Multi-scale, Multi-physics Processes

• Sawtooth Region (q < 1)
• Core confinement Region
• Magnetic Islands
• Edge Pedestal Region
• Scrape-off Layer
• Vacuum/Wall/Conductors/Antenna

Plasma-Wall Interactions  Atomic Physics  Radiative Transport  Energetic Particles  Heating & Current Drive

Core & Edge Transport  Plasma Turbulence  Large Scale Instabilities  MHD Equilibrium

Courtesy of Bill Tang, Princeton
# Progress in Turbulence Simulation Capability: Faster Computer ➔ Achievement of Improved Fusion Energy Physics Insights

<table>
<thead>
<tr>
<th>GTC simulation name</th>
<th>PE#</th>
<th>Speed (TF)</th>
<th>Particle #</th>
<th>Time steps</th>
<th>Physics Discovery (Publication)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1998 Cray T3E NERSC</td>
<td>$10^2$</td>
<td>$10^{-1}$</td>
<td>$10^8$</td>
<td>$10^4$</td>
<td>Ion turbulence zonal flow (Science, 1998)</td>
</tr>
<tr>
<td>2002 IBM SP NERSC</td>
<td>$10^3$</td>
<td>$10^0$</td>
<td>$10^9$</td>
<td>$10^4$</td>
<td>Ion transport scaling (PRL, 2002)</td>
</tr>
<tr>
<td>2009 Jaguar/Cray XT5 ORNL</td>
<td>$10^5$</td>
<td>$10^3$</td>
<td>$10^{11}$</td>
<td>$10^5$</td>
<td>Electron transport scaling (PRL, 2009); EP-driven MHD modes</td>
</tr>
<tr>
<td>2012-13 Cray XT5 ➔ Titan ORNL</td>
<td>$10^5$</td>
<td>$10^4$</td>
<td>$10^{12}$</td>
<td>$10^5$</td>
<td>Kinetic-MHD: Turbulence + EP + MHD</td>
</tr>
<tr>
<td>2018 To Extreme Scale HPC Systems</td>
<td>$10^6$</td>
<td>$10^{13}$</td>
<td></td>
<td>$10^6$</td>
<td>Turbulence + EP + MHD + RF</td>
</tr>
</tbody>
</table>

* Example here of GTC code (Z. Lin, et al.) delivering production runs @ TF in 2002 and PF in 2009

Courtesy of Bill Tang, Princeton
Exaflops by 2019 (maybe)

Courtesy of Erich Strohmaier  LBNL
Practical Constraints for Exascale

- **Sustained Performance**
  - Exaflops
  - 100 Petabytes
  - 125 Petabytes/sec.

- **Cost**
  - Deployment – 8 billion Rubles
  - Operational support

- **Power**
  - Energy required to run the computer
  - Energy for cooling (remove heat from machine)
  - 20 Megawatts

- **Reliability**
  - One factor of availability

- **Generality**
  - How good is it across a range of problems

- **Usability**
  - How hard is it to program and manage

- **Size**
  - Floor space – 4,000 sq. meters
  - Access way for power and signal cabling
Where Does Performance Come From?

- **Device Technology**
  - Logic switching speed and device density
  - Memory capacity and access time
  - Communications bandwidth and latency

- **Computer Architecture**
  - Instruction issue rate
    - Execution pipelining
    - Reservation stations
    - Branch prediction
    - Cache management

- **Parallelism**
  - Parallelism – number of operations per cycle per processor
    - Instruction level parallelism (ILP)
    - Vector processing
  - Parallelism – number of processors per node
  - Parallelism – number of nodes in a system
A History of HPC
Clusters Dominate HPC System Architecture
Technology Demands new Response

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

Power (MW)

- Heavyweight
- Lightweight
- Heterogeneous

Courtesy of Peter Kogge, UND
* Decade of Canonical Systems: Rmax
Total Concurrency

TC (Flops/Cycle)

Heavyweight  Lightweight  Heterogeneous

Courtesy of Peter Kogge, UVP

Center for Research in Extreme Scale Technologies
Indiana University
Pervasive Technology Institute
Multi-core Intel Xeon

- EMT64 processors for servers and workstations
- Popular in many installations
  - Quad-core Clovertown was introduced in late 2006
    - Based on Core2 architecture
    - Two dual-core Woodcrest chips in one package
    - 48 GFLOPS peak at 3GHz
    - 582 mil. transistors in 65nm technology
    - 1333 MHz bus speed
    - 150W TDP at 3GHz, 80W at 2.33GHz or less

- Westmere lineup (launched in 2010) enabled efficient multiprocessor configurations
  - 4- and 6-core version of Nehalem architecture
  - QPI for intra-node coherency traffic
  - 83.04 GFLOPS peak at 3.46GHz
  - 1.17 bil. transistors in 32nm (6-core)
  - 3-channel DDR3 memory controller at 1333MHz (32GB/s)
  - 130W TDP for 6-core at 3.47GHz (or 95W at 3.07GHz)
  - Westmere-EX increased number of cores to 10 per die
Nvidia Tesla

- GPGPU accelerator
- Major technology revisions: Tesla (original), Fermi, and Kepler
- C870 introduced in 2007 was based on GeForce 8 (G80) shader architecture
  - 128 thread processors at 1350 MHz
  - 518.4 single-precision GFLOPS (no dual precision)
  - 1.5 GB GDDR3 memory with 76.8 GB/s throughput
  - 170.9W TDP
- The newest K20X, based on GK110 architecture, launched in 2012
  - 2688 thread processors at 732 MHz
  - 3950 GFLOPS (single-precision), 1310 GFLOPS double-precision FMA
  - 6144 GB GDDR5 memory, throughput 250 GB/s
  - 235W TDP
Titan

- **USA, 2012:** *Jaguar avenged*
- Manufactured by Cray for ORNL
- Hybrid design with AMD Opterons and Nvidia Tesla GPGPUs
- Cray XK7 architecture:
  - 299,008 Opteron 6274 CPU cores
  - **18,688 Tesla K20 (Kepler) accelerators**
  - 710,144 GB memory (598,016 GB DDR3 attached to processors; 112,128 GB GDDR5 on GPU boards)
  - **17,590 TFLOPS** in HPL (#1 in November 2012)
  - Theoretical peak: 27,112.5 TFLOPS
  - Cray Gemini interconnect
  - 40 PB storage at aggregate 1.4 TB/s
  - 200 cabinets in 404m² area
  - $97 million contractual cost
Accelerators

- Intel Xeon Phi
- Clearspeed
- IBM Cell
- ATI Radeon
- Nvidia Kepler
- Nvidia Fermi

Courtesy of Erich Strohmaier  LBNL
IBM PowerPC

- Power efficient line of RISC cores
- Blue Gene/L ASIC used 32-bit PowerPC 440
  - Dual-core modules with two FPUs per core
  - Not cache-coherent
  - 5.6 GFLOPS at 700 MHz
  - IBM Cu-11 0.13 µm process, 95 mil. transistors
  - 12W power usage per ASIC
- Blue Gene/P utilized PowerPC 450 cores
  - Cache-coherent across ASIC (4 cores)
  - 13.6 GFLOPS at 850 MHz
  - 208 mil. transistors in IBM Cu-08 90nm process
  - 16W power
- Blue Gene/Q: 64-bit PowerPC A2
  - 18 cores
  - 204.8 GFLOPS at 1.6 GHz
  - 1.47 billion transistors in 45 nm
  - 55 W power draw
“Light Weight” Strawman

2 Nodes per “Compute Card.” Each node:
• A low power compute chip
• Some memory chips
• “Nothing Else”

System Architecture:
• Multiple Identical Boards/Rack
• Each board holds multiple Compute Cards
• “Nothing Else”
Sequoia

- **USA, 2012: BlueGene strikes back**
- Built by IBM for NNSA and installed at LLNL
- **20,123.7 TFLOPS peak performance**
  - Blue Gene/Q architecture
  - 1,572,864 total PowerPC A2 cores
  - 98,304 nodes in 96 racks occupy 280m²
  - 1,572,864 GB DDR3 memory
  - 5-D torus interconnect
  - 768 I/O nodes
  - 7890kW power, or 2.07 GFLOPS/W
  - Achieves 16,324.8 TFLOPS in HPL (#1 in June 2012), about 14 PFLOPS in HACC (cosmology simulation), and 12 PFLOPS in Cardioid code (electrophysiology)
Intel Many Integrated Core (MIC)

- Based on Larrabee many core architecture
- Official branding: Xeon Phi
- Modified Pentium P54C cores
  - Up to 61 cores per die, 4 threads per core
  - Up to 1220 GFLOPS at 1.25 GHz
  - Max. 8 GB GDDR5 memory at 5.5 GHz, 512 bit bus
  - 22nm technology, approx. 5 billion transistors
  - 225-300W TDP
  - Available as PCIe x16 board
- Main processing component of Tianhe-2 and TACC Stampede
Energy per Flop

Energy per Flop (pJ)


Courtesy of Peter Kogge, UND
Energy Projections

- Heavyweight
- Lightweight
- Heterogeneous

- Heavyweight - Scaled
- Lightweight - Scaled
- Heterogeneous - Scaled
- Heavyweight - Constant
- Lightweight - Constant
- Historical
- CMOS Projection - Hi Perf
- CMOS Projection - Low Power

Courtesy of Peter Kogge, UND
Performance Projections

- Heavyweight
- Lightweight
- Heterogeneous
- Heavyweight - Scaled
- Lightweight - Scaled
- Heavyweight - Constant
- Lightweight - Constant

Courtesy of Peter Kogge, UND
Dally’s 2-3-4 Rule for Power Improvement

- Stated at ISC-13 Keynote address
- Needs a factor of 25X energy efficiency improvement
- 2.2X through device technology fabrication process
- 3X through logic circuit design
- 4X through architecture
The Negative Impact of Global Barriers in Astrophysics Codes

Computational phase diagram from the MPI based GADGET code (used for N-body and SPH simulations) using 1M particles over four timesteps on 128 procs.

Red indicates computation
Blue indicates waiting for communication
Performance Factors - SLOWER

\[ P = e(L,O,W) \times S(s) \times a(r) \times U(E) \]

- **Performance** (P)
  - efficiency \( e \) (0 < e < 1)
  - application’s average parallelism \( s \)
  - availability \( a \) (0 < a < 1)
  - normalization factor/compute unit \( U \)
  - watts per average compute unit \( E \)
  - reliability \( r \) (0 < r < 1)

**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
When to Jump, if ever?

• In the beginning:
  – Clusters of single-core nodes
  – CSP/MPI/BSP served well

• Then came SMPs
  – Different breed with shared memory, MESI
  – OpenMP

• And dual-core chips

• Clusters of Multi-core SMPs

• Suddenly: (Oh Nooo!) MPI+X
  – e.g., X = OpenMP
Sources of Asynchrony for Exascale

- Scale
- Increase range of network latencies and opportunities for packet collisions and routing variations
- Deeper memory hierarchy
- Scheduling conflicts for threads to cores
- Active response to errors
- Variable instruction rate from clock and voltage change
- Finer grain threads as normalization factor of time delays
Goals of a New Execution Model 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
The Purpose of a QUARK Runtime

• Objectives
  – High utilization of each core
  – Scaling to large number of cores
  – Synchronization reducing algorithms

• Methodology
  – Dynamic DAG scheduling (QUARK)
  – Explicit parallelism
  – Implicit communication
  – Fine granularity / block data layout

• Arbitrary DAG with dynamic scheduling

Fork-join parallelism
Notice the synchronization penalty in the presence of heterogeneity.
X-Stack Portfolio

• **DEGAS** (Kathy Yelick)
  Hierarchical and resilient programming models, compilers and runtime support.

• **Traleika** (Shekhar Borkar)
  Exascale programming system, execution model and runtime, applications, and architecture explorations, with open and shared simulation infrastructure.

• **D-TEC** (Dan Quinlan)
  Complete software stack solution, from DSLs to compilers to optimized runtime systems.

• **XPRESS** (Ron Brightwell)
  Software architecture and interfaces that exploit the ParalleX execution model, prototyping several of its key components.

• **DvnAX** (Rishi Khan)
  Novel programming models, dynamic adaptive execution models and runtime systems.

• **X-Tune** (Mary Hall)
  Unified autotuning framework that integrates programmer-directed and compiler-directed autotuning.

• **GVR** (Andrew Chien)
  Global view data model for architecture support for resilience.

• **CORVETTE** (Koushik Sen)
  Automated bug finding methods to eliminate non-determinism in program execution and to make concurrency bugs and floating point behavior reproducible.

• **iLEEC** (Milind Kulkarni)
  Semantics-aware, extensible optimizing compiler that treats compilation as an optimization problem.
HPX

• Active global address space (AGAS) over PGAS
• Message driven computation over message passing
• Lightweight control objects over global barriers
• Latency hiding over latency avoidance
• Adaptive locality control over static data distribution
• Moving work to data over moving data to work
• Fine grained parallelism of lightweight threads instead of Communicating Sequential Processes
## Exascale System Advances

<table>
<thead>
<tr>
<th>Conventional Incremental</th>
<th>Advanced Revolutionary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change</td>
<td>Change</td>
</tr>
<tr>
<td>Static – scheduling &amp; management</td>
<td>Dynamic – adaptive runtime control</td>
</tr>
<tr>
<td>Message passing</td>
<td>Message driven</td>
</tr>
<tr>
<td>Distributed memory</td>
<td>Global name space</td>
</tr>
<tr>
<td>Local view; rest is I/O</td>
<td>System-wide object access</td>
</tr>
<tr>
<td>Von Neumann bottleneck</td>
<td>Embedded memory processing</td>
</tr>
<tr>
<td>Bulk Synchronous Parallel (BSP)</td>
<td>Dataflow – overlapped phases</td>
</tr>
<tr>
<td>Speculative</td>
<td>Multi-threaded</td>
</tr>
<tr>
<td>Synchronous</td>
<td>Asynchronous</td>
</tr>
<tr>
<td>Global barriers</td>
<td>Futures – continuations migration</td>
</tr>
<tr>
<td>Separation of cores vs. NICs</td>
<td>Merged ISA for compute/comm.</td>
</tr>
<tr>
<td>Explicit fixed processes</td>
<td>Meta-threads, depleted threads</td>
</tr>
<tr>
<td>Hybrid programming - monstrosity</td>
<td>Unified programming model</td>
</tr>
</tbody>
</table>
Focus and Premise

• Execution model
  – Computing strategy to exploit emerging enabling technologies
  – Abstract machine model to attack sources of performance degradation
  – Guides co-design of system component layers
  – Governs cooperative operation among system component layers

• ParalleX execution model
  – Addresses requirements for extreme scale computing
  – Delivers significant advances in efficiency and scalability for some problems
  – Supports runtime software for dynamic adaptive introspective computing

• Insufficient as defined for future real-world exascale
  – Productivity, Resilience, Power, Portability
  – ParalleX II: an integrated strategy to fault tolerance, energy, debug, realtime
ParalleX Addresses Critical Challenges (1)

• Starvation
  – Lightweight threads for additional level of parallelism
  – Lightweight threads with rapid context switching for non-blocking
  – Low overhead for finer granularity and more parallelism
  – Parallelism discovery at runtime through data-directed execution
  – Overlap of successive phases of computation for more parallelism

• Latency
  – Lightweight thread context switching for non-blocking
  – Overlap computation and communication to limit effects
  – Message-driven computation to reduce latency to put work near data
  – Reduce number and size of global messages
ParalleX Addresses Critical Challenges (2)

• Overhead
  – Eliminates (mostly) global barriers
  – However, ultimately will require hardware support in the limit
  – Uses synchronization objects exhibiting high semantic power
  – Reduces context switching time
  – Not all actions require thread instantiation

• Waiting due to contention
  – Adaptive resource allocation with redundant resources
    • Like hardware for threads
  – Eliminates polling and reduces # of sources of synch contacts
ParalleX Compute Complex

-- Runtime Aware
-- Logically Active
-- Physically Active
Parcels may utilize underlying communication protocol fields to minimize the message footprint (e.g. destination address, checksum)
Generic LCO

Incident Events

Event Buffer

Control State

Predicate

Thread Create

New Thread

Inherited Generic Methods

Event Assimilation Method

Control Method

Thread Method
OpenX Software Architecture

- Legacy Applications
  - OpenMP
  - MPI
- New Model Applications
  - Domain Specific Language
  - Domain Specific Active Library
- Compiler
  - XPI
- Metaprogramming Framework
- Runtime System Instances
  - AGAS
    - name space processor
  - Lightweight Threads
    - context manager
  - Parcels
    - message driven computation
  - PRIME MEDIUM
    - Interace / Control
- Operating System
- Distributed Framework
- Hardware Architecture
  - +10^6 nodes x 10^3 cores / node + integration network

X-Stack Review
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)
Overlapping computational phases for hydrodynamics for LULESH (mini-app for hydrodynamics codes).

Red indicates work

White indicates waiting for communication

Overdecomposition: MPI used 64 processes while HPX used 1E3 threads spread across 64 cores.
Dynamic load balancing via message-driven work-queue execution for Adaptive Mesh Refinement (AMR)
Application: Adaptive Mesh Refinement (AMR) for Astrophysics simulations

Performance Impact of Removing Global Barriers
1-D AMR with quad precision using 2 levels of refinement

- Performance upper limit for cases with a global barrier
- $r = 0$
- $r = 4.4$
- $r = 6.8$
- $r = 8.4$

Simulation Time vs. Physical Time
Micro-Checkpointing
Compute-Validate-Commit Cycle

• Fault management
  – Error detection, Isolation, Diagnosis, Reconfiguration
  – Restart

• Fault zones
  – Identify and retain precedent constraints (micro-checkpointing)
  – Compute up to global mutable effects
  – Validate correctness
  – Hierarchical and overlapping

• Fault detection at many levels, no single-level solution
  – Application, library, compiler
  – Runtime and OS
  – Hardware architecture
Extreme Debugging

• Software errors at extreme scale are largely indistinguishable from hardware or transient errors

• Conventional practices (including the venerable ‘printf’) cannot manage the quadrillions of threads manifest during the execution on future exascale systems

• Bugs are insidious
  – Repeatability is poor
  – May be data or control dependent
  – May occur in OS, runtime, or libraries rather than user code

• Challenges include:
  – Error prevention, detection, isolation, diagnosis, correction, recovery
Side-Path Energy Suppression

• Power and energy, related but different
  – Power imposes operational constraints
  – Point power determines speed of operation
  – Energy equated to cost
  – Energy is a resource that can be managed

• Critical path of executing thread sequence
  – Determines time to solution and average system performance
  – For fastest performance, should receive highest power

• Side-path threads can complete early providing precedent results
  – Either run at slower clock rate or turn off physical threads when done
  – Suppresses energy consumption non-critical path computation

• Determining optimal trace path a priori
  – Approximate
  – Subject to change due to scheduling effects
Real-Time

• Not usually associated with HPC
• Important for many embedded applications and services
  – How to perform time-critical functionality
  – Exploit parallelism when too much work for single-threaded outcome
• Key to introspective control of exascale computing systems
  – Add time-domain semantics to control parameters
  – Predict progress-to-goal for adaptive resource management
  – Establish time-based utility function for productivity
• Expand ParalleX to incorporate event-driven state definition
  – Partial ordering
  – Constrained time elasticity between events
  – Treating of time errors
Conclusions

• **ParalleX** execution model is a framework for co-design and interoperability of future exascale system layer components to enable dynamic adaptive resource management and task scheduling through introspective methods for dramatic improvements in efficiency and scalability for some problems.

• ParalleX is **inadequate** to fully serve the needs of exascale computing as it does not address **practical problems** of: **fault tolerance, energy reduction, debugging, and real-time response**

• These four issues, ordinarily treated separately, are recognized to be **interrelated** in requirements and solution structures.

• A new generation of execution model (**ParalleX II**) is under development to incorporate an integrated