Architectures for Extreme-Scale Computing

Josep Torrellas

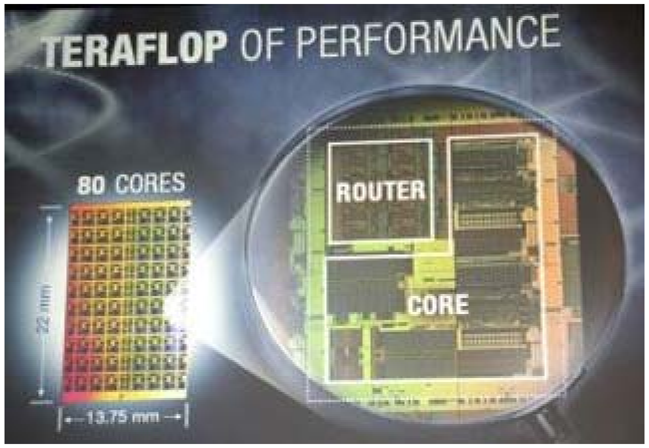
Department of Computer Science
University of Illinois at Urbana-Champaign
http://iacoma.cs.uiuc.edu

Exascale at Affordable Power

- Current petascale machines:
  - Footprint of 1/10th of a US football field
  - Consume over 10MW
    - 1MW == $1M per year
- Need Extreme Scale computing: 1000x more capable for the same power consumption and physical footprint
  - Exascale (10^{18}) datacenter: 20MW
  - Petascale (10^{15}) departmental server: 20KW
  - Terascale (10^{12}) portable device: 20W

Example

- Intel Polaris Chip (2007)
  - 80 very simple cores
  - At 5.7GHz, attained 2 TFLOPS (at 256W)



→It was bare-bones
→Power has to come down to less than 50W

Architectural Challenges in Extreme Scale

- Energy and power efficiency
- Concurrency and locality
- Resiliency
- Programmability

P. Kogge et al: “Exascale Computing Study: Technology Challenges in Achieving Exascale Systems, DARPA-IPTO, 2008

Energy and Power Efficiency

- Most formidable challenge
- Three orders of magnitude more energy efficient than current machines
- 20MW exascale / 20KW petascale / 20W terascale:
  - 50 Gops/Watt = 50×10^9 Ops/Watt
  - 20 pJoules/operation
- Reference: Intel CoreDuo mobile microprocessor 2006:
  - More than 10K pJ/instruction
- Problem is even harder:
  - Large machines spend most of the energy transferring data
  - Minimizing transport data, not ALU energy is the challenge

Evolutionary Approaches

- Design circuits for energy and power efficiency rather than speed
  - Low-swing on-chip interconnection network circuits
  - New memory layouts and bank organizations that minimize the capacitance switched per access
    - Minimize energy charging and discharging lines
    - At cost of bandwidth?
- Simplify the processor, shallow pipeline, less speculation
- Augment processing nodes with accelerators that are energy-efficient for some operations

$\rightarrow$ Not enough. Need new technologies

New Technologies Needed

- Near-Threshold Voltage operation
- Non-silicon memory
- 3D die stacking
- Efficient on-chip voltage conversion
- Photonic interconnects

Near-Threshold Voltage (NTV) Operation

- Most effective approach for energy efficiency:
  - Reduce supply voltage V_{dd} to a value slightly higher than V_{th}
  - Corresponds to V_{dd} of about 0.4V rather than current 1V
- Operation under NTV:
  - Reduces power consumption of gates by 100x
  - Increases their delay by 10x
  - Result: total energy savings by one order of magnitude
  - Additional drawbacks
    - Induces a 5x increase in gate delay variation
    - Induces several orders of magnitude increase in logic failures (especially in memories, less variation tolerant)

How to Deal with the NTV Challenges?

- If processors are slower → more processors
- Optimal V_{dd} for SRAMs is higher than for logic
  - Caches can cycle a few times faster than processors
  - Redesign cache hierarchy where processors share caches
  - Redesign processor to take multiple cycles to access registers
- Aggressive use of techniques that tolerate process variation
  - Adaptive body biasing
  - Variation-aware job scheduling
  - Novel designs of SRAM cells (8T rather than 6T)
- NTV mostly reduces dynamic power
  - Leakage power dominates
  - Super-aggressive techniques to power-gate chunks of logic

Non-Silicon Memory

- Several choices of non silicon memory.
- One example is Phase Change Memory (PCM):
  - Uses storage element composed of two electrodes separated by a resistor and phase-change material, such as $Ge_2Sb_2Te_5$
  - Current through resistor heats the phase change material
  - Depending on the T conditions, changes between
    - Crystalline (low-resistivity)
    - Amorphous (high-resistivity)
  - This stores a bit.
  - It can also store multiple bits if multiple levels

Advantages of PCM

- Scalability with process technology
  - Heating contact area and heating current shrink with tech gen
  - PCM will enable denser, larger, and very energy-efficient main memories
  - DRAM: Non-scalable technology (needs big capacitors to store charge and, therefore, big transistors)
- Non-volatile:
  - Eliminates the leakage problem
  - Potentially support novel uses:
    - Fast, low-consumption checkpointing of the program state
    - Hybrid DRAM/PCM memories

Current Disadvantages of PCM

- Longer access latencies than DRAM
- Higher energy per access than DRAM (especially for writes)
- Limited lifetime in the number of writes
  - Need to play tricks not to wear-out certain memory areas

Things will likely improve

3D Die Stacking

- Main goal: reduce memory access power
- 3D stack may contain:
  - Only memory dies
  - Processor die and memory dies
- Advantages:
  - Eliminate energy-expensive data transfers
  - Enables high BW connection between memory and proc
    - Induce a reorganization of the cache hierarchy?
    - Enables high-bandwidth caches near the core
- Problems: hard to manufacture

Efficient On-Chip Voltage Conversion

- Provide several voltage islands on chip
  - Adapt the core(s) to the thread or the environment
  - Save a lot of power
- With on-chip voltage conversion:
  - Fast changes in voltages (in ns)
  - Power-efficient conversion of the voltages

Photonic Interconnects

- Optics have key properties that can be used for interconnects
  - Low-loss communication
  - Very large message BW enabled by wavelength parallelism
  - Low transport latencies (speed of light)
- This makes them great for long-range communication
  - End-to-end reductions in Energy/bit and time/access
- Extreme-scale machines will use them, especially for communication between far-away nodes in large machines
- Active research on photonics for on-chip interconnects
- Need to advance the interfaces between electronic and photonic signaling

Architectural Challenges in Extreme Scale

- Energy and power efficiency
- Concurrency and locality
- Resiliency
- Programmability

Concurrency and Locality

- Performance of extreme-scale machines will not be attained through higher frequencies
- Need to rely on more threads running concurrently
- Example:
  - 1GHz cores finishing one operation per cycle
  - Chip: 1,024 cores to attain one Tera-op
  - Server: 1M cores for a Peta-op
  - Data-center: 1B cores for an Exa-op
- In reality, a thread will often stall waiting for data → to hide the stall time, need several times more threads

Challenge

- Need memory hierarchies, synchronization primitives, network designs that support these concurrency levels
- At the same time must exploit high degrees of spatial/temporal locality (to keep energy constraints)

Supporting Fine-Grain Parallelism

- Efficient, scalable synch and communication primitives
  - Efficient point-to-point synch between two cores
    - Test-and-set and such
    - Producer-consumer with F/E bits
  - Low overhead dynamic hierarchical barriers
  - Broadcast updates
  - Collective operations
- Primitives for the creation, commit, and migration of lightweight tasks
  - Tasks created by compiler
  - Spawned with a single instruction
  - Managed with scalable queuing hardware

Structures that Minimize Data Movement

- Many-core chip organization based on clusters
  - A cluster shares some level of the cache hierarchy
  - Can be exploited by the compiler
- Simple compute engines in the mem controllers or in the L3 cache controllers → Processing in Memory (PIM)
  - Perform memory-intensive operations
  - Seek to avoid transfer of data mem → proc → mem
  - Typical operations: Ops on arrays or sets of data, recurrences, reductions, etc
- Mechanisms to prevent that caches move unnecessary data

Architectural Challenges in Extreme Scale

- Energy and power efficiency
- Concurrency and locality
- Resiliency
- Programmability

The Challenge of Resiliency

Many problems are getting worse:

- Spatial variations in process, voltage, and T, as well as wear-out
  - Relatively more acute as feature sizes decrease
- Smaller feature sizes imply less charge in storage elements
  - Elements more vulnerable to soft errors by particle impact
- Use of V_{dd} values close to V_{th} increases process variation

Increase the chances of permanent or transient error

Machines Have more Components

- Large machines will have many components, which increases the chance of faults
- Example: Exa-op supercomputer
  - 10-100 Pbytes of memory → tens or hundreds or millions of memory chips
  - Hundreds of exabytes of secondary storage → millions of disk drives

No single solution can fully address this challenge
Need combination of techniques at different levels

Techniques for Resiliency

	Detection/Isolation	Correction/Recovery
HW	-Hardened latches -Parity/ECC - Testing circuitry -Detector of aging (ring,etc) -Watchdog timers -DIVA checkers -HW redundancy	Efficient checkpointing Micro-rollback Network reconfiguration Core spares/core disabling/core salvaging Core pairing Thermal management
OS/runtime	Resiliency module to diagnose and test	Scheduling threads around errors Core salvaging Virtualization
Compiler	Embed checksums in the basic blocks of the code	Compiler support for Intelligent checkpointing
Appl & Progr System	Applications that check themselves	Appl does its own checkpointing at key points Use transactional model

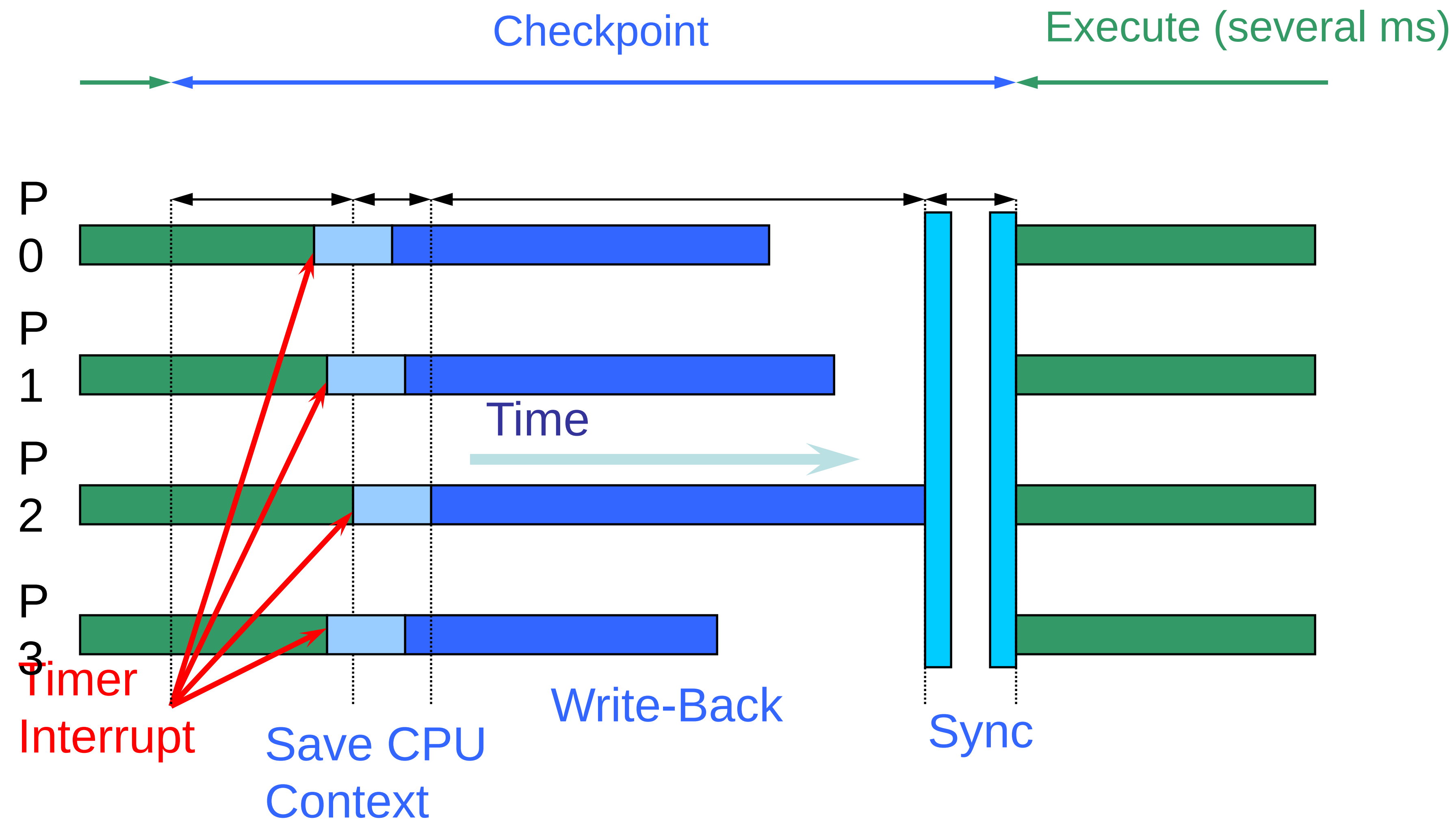
ReVive: Incremental In-Memory Checkpointing [ISCA02]

- App checkpointing: traditionally high-overhead in high-end computing
- What is costly about checkpointing:
  1. Access to disk
  2. Moving large chunks of data
- What ReVive proposes:
  – In-memory, incremental checkpointing
  – Use PCM (non-volatile memory) to avoid disk access
- Result: Scalable checkpointing scheme:
  – Very low overhead during error-free operation
  – System downtime in an error is kept to a minimum
  – Compatible with off-the-shelf processors/caches/mem modules
  – Transparent to software (app, compiler, VMM, OS) → productivity

Basic In-Memory Incremental Checkpointing

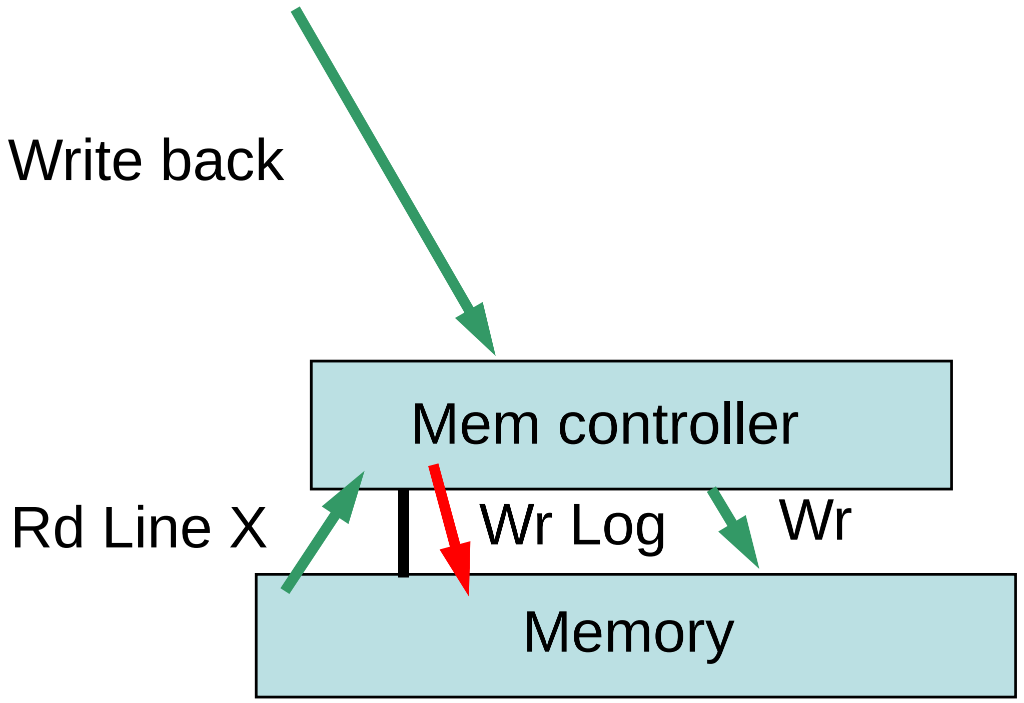
- Periodically establish a global checkpoint by:
  - Interrupting all the processors
  - Write-back dirty data from caches, save processor context
  - Main memory is the checkpointed state of the program
- Between Checkpoints:
  - When program execution modifies memory for 1st time, memory controller saves the previous value in a log
  - Leaves a trail of updates that enables restoring to previous checkpoint

Checkpoint Creation Timeline

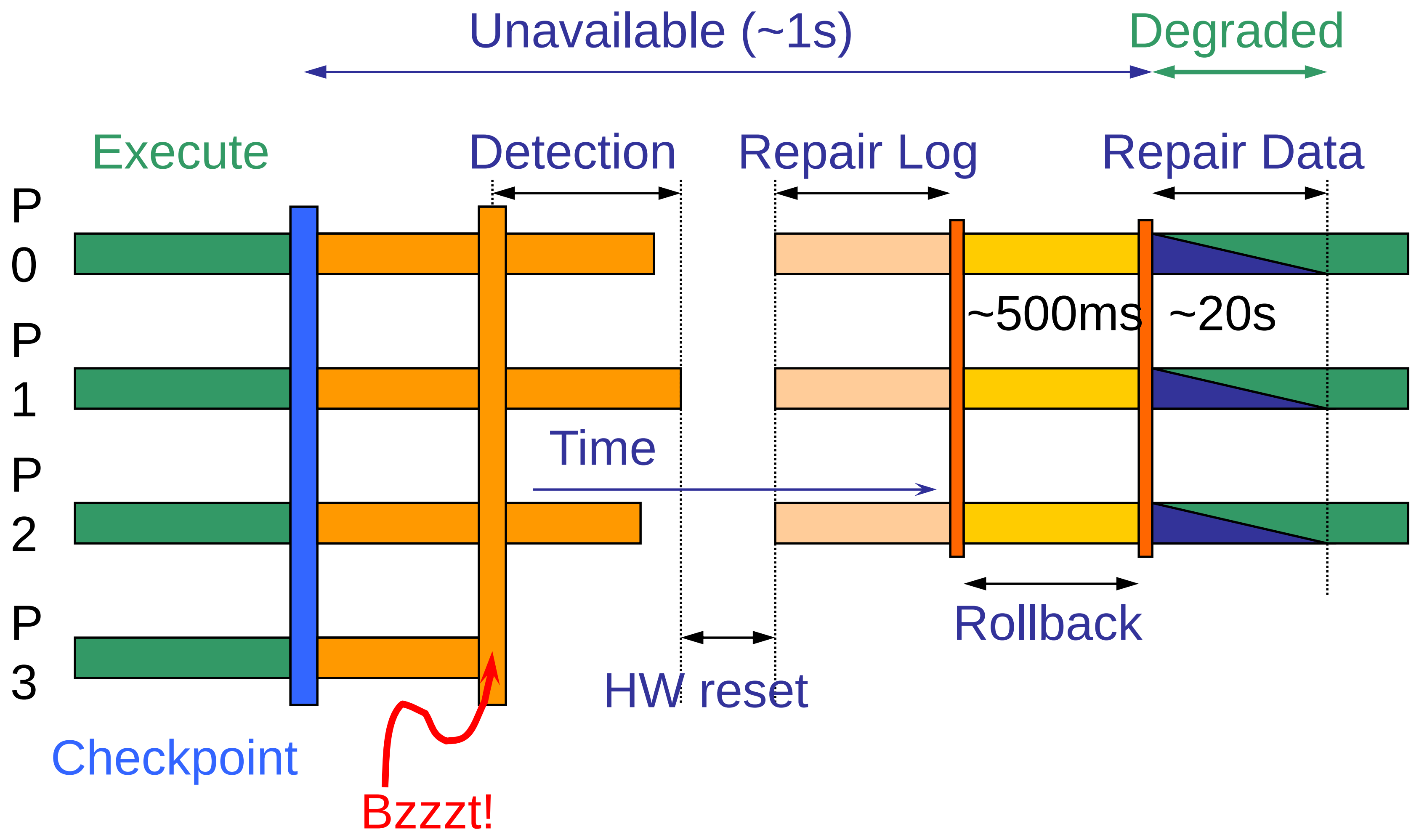


Between Checkpoints: Logging in HW

- Only needed for the first write back or displacement of the line since last ckpt



Recovery from Permanent Node Loss



Summary

- Low cost: HW changes only to memory controllers
- Low performance overhead in error-free operation: < 5%
- Low recovery overhead: ~1 second
- High availability:
  - Recovers from system-wide transients
  - If memory is RAID-ed: recovers from permanent loss of one node
- Transparent to SW

Architectural Challenges in Extreme Scale

- Energy and power efficiency
- Concurrency and locality
- Resiliency
- Programmability

Programmability

- Programming highly-concurrent machines has required heroic efforts
- Extreme-scale architectures, with emphasis on power-efficiency, may make it worse
  - Low Vdd requires more concurrency to attain same perf
  - Carefully manage locality and minimize communication

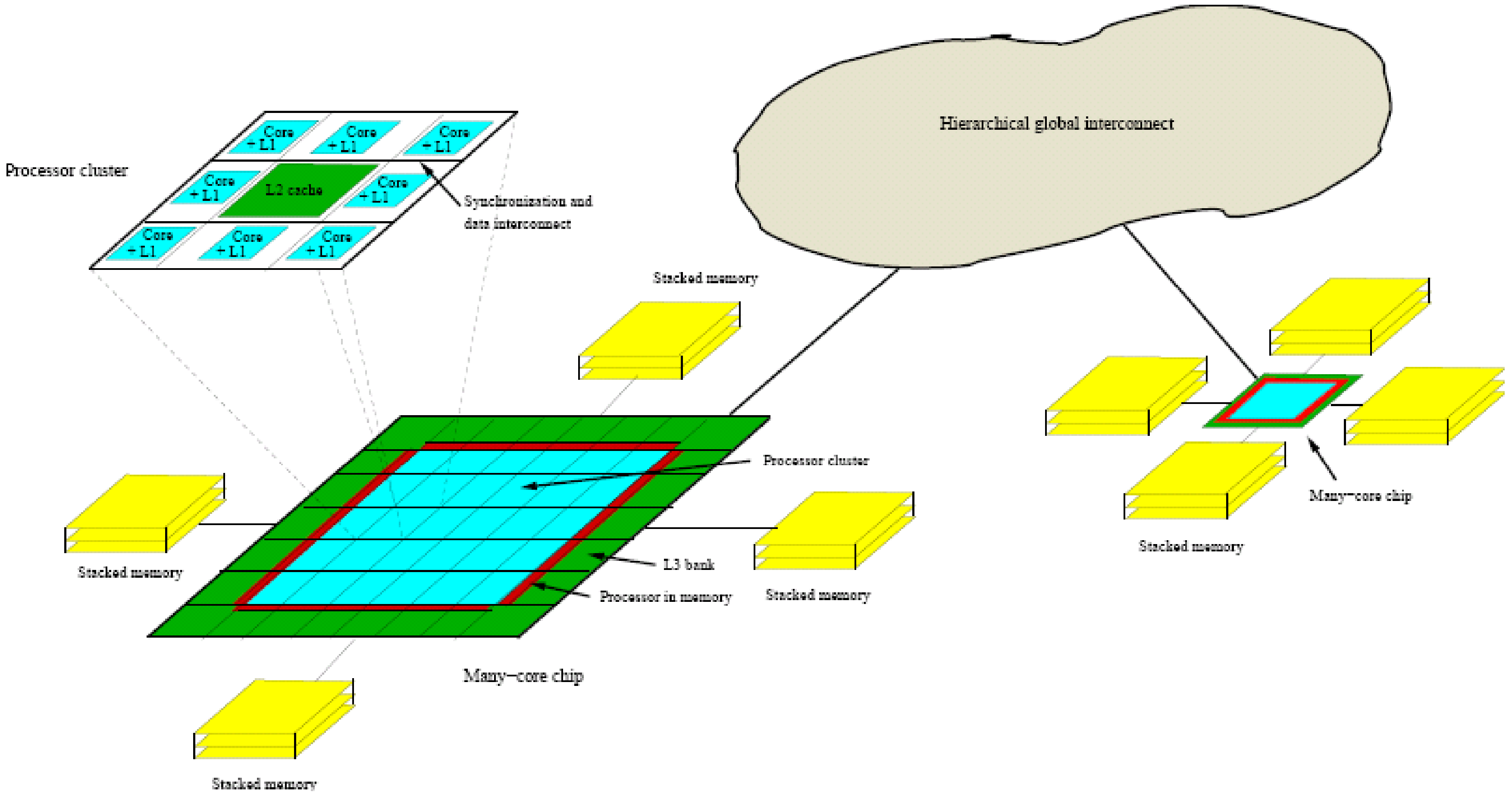
How to Enhance Programmability

- Programmers need
  - Express high-degree of parallelism
  - Not preclude careful locality management and communication minimization
- General approach:
  - Program in a high level programming model
    - Likely data parallel
  - Rely on an intelligent translation layer to map code to HW
    - Compiler needs to map tasks and subtasks to localities
- Need autotuning because the architecture is complex

Architectural Features for Programmability

- Machine provides a single address space to the software
- Hardware mechanisms to support compiler optimizations and high-level languages
  - Mechanisms to detect inter-thread dependences
  - Mechanisms to manage cache in software
- Hardware features to detect data transfer patterns and eliminate or optimize the transfers.
  - Primitives for prefetching, multicast-update, move computation to data
  - Efficient synch primitives
- Autotuning: performance or energy monitoring hardware:
  - Counters, signatures, trace buffers

Thrifty Extreme Scale Architecture at UIUC



Funded by DOE-Exascale

Conclusion

- High performance architectures present major challenges
- Need to advance enabling technologies
  - Near V_{th} operation, optics, 3D-stacking, non-silicon memories, accelerators
- Need to involve all the layers of the computing stack
  - Programming models, compilers, runtime, arch, CAD
- Applications researchers part of the team