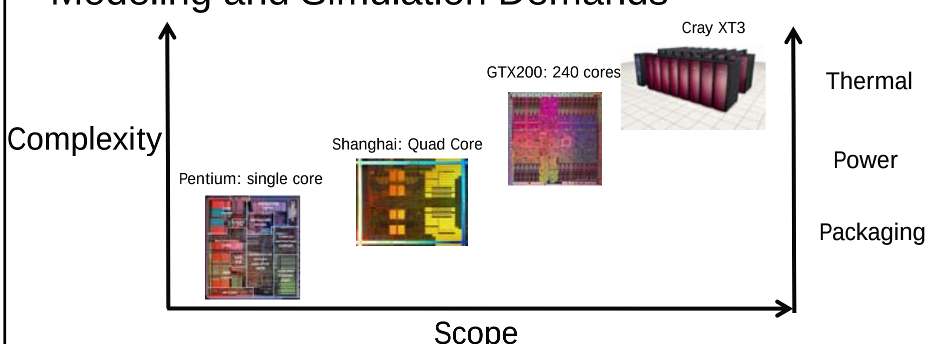
Scalable Simulation Frameworks

Sudhakar Yalamanchili, George Riley, Tom Conte, Hyesoon Kim
and many students….

Center for Experimental Research in Computer Systems
School of Electrical and Computer Engineering and School of Computer Science
Georgia Institute of Technology

Modeling and Simulation Demands



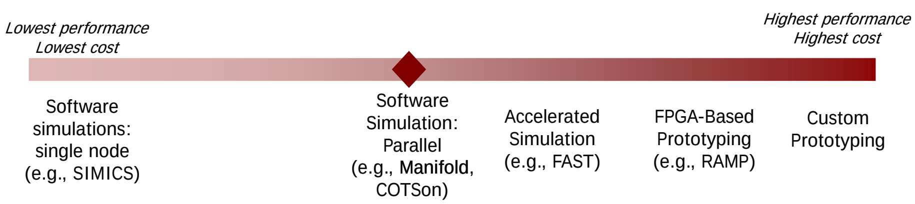
- System complexity is outpacing simulation capacity
  - Cannot perform analysis at scale
- The problem is getting worse faster → **Simulation Wall**
- Islands of simulators and simulation systems
  - Customized interactions
  - Difficult to leverage individual investments

Prioritized Major Challenges*

1. Cost of building a validated useful simulator
   - Composable
   - New methodologies for building simulators
2. Accuracy
   - Need for calibrated models
   - Methodologies for constructing calibrated models
3. Performance
   - Parallelism, multiscale, and hardware acceleration
4. Power and thermal models
5. Ease of use: Productivity and Management Tools
   - Visualization, deployment, debugging, etc.
   - Documentation & deployability

*From Outbrief: *Performance Prediction and Simulation for Exascale Interconnection Networks,* (Co-Chairs C. Janssen (SNL) and S. Yalamanchili (GT) Interconnect Workshop, DoE Institute for Advanced Architectures, July 2008

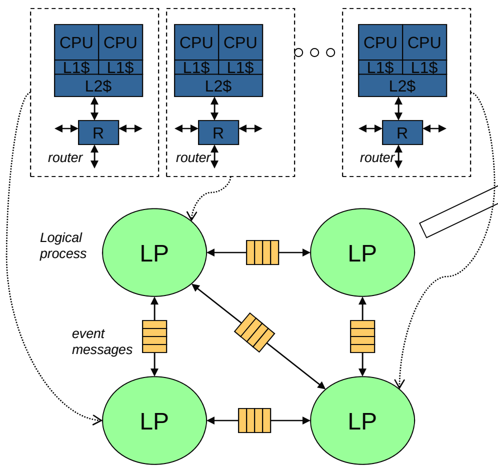
Spectrum of Solutions

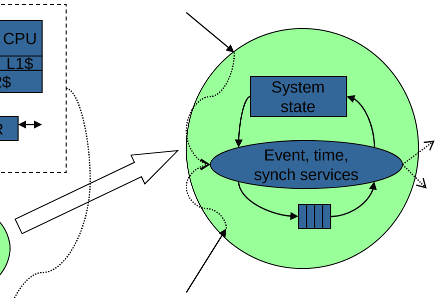


- Simple Premise: Use parallel machines to simulate/emulate parallel machines
- Leverage mature point tools via standardized API for common services
  - Event management, time management, synchronization
- Support Sandia's Structural Simulation Toolkit (SST)

Coarse Grain Parallel Simulation

Example Modeled System

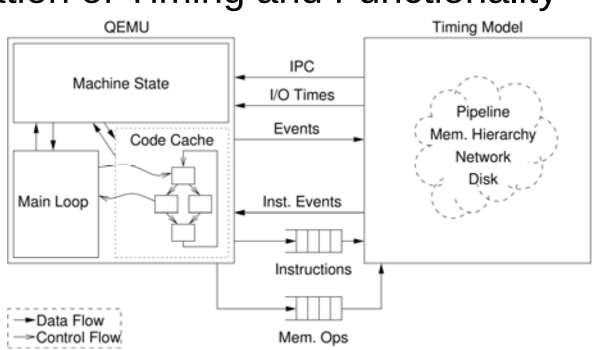




Key Challenges
1. Exploit program semantics
2. Exploit architecture behaviors

1. D. Burger and D. Wood, "Accuracy vs. Performance in Parallel Simulation of Interconnection Networks", *ICPP* 1995
2. A. Falcon P. Faraboschi D. Ortega, "An Adaptive Synchronization Technique for Parallel Simulation of Networked Clusters," *ISPASS*, 2008
3. Parallel SST, SNL

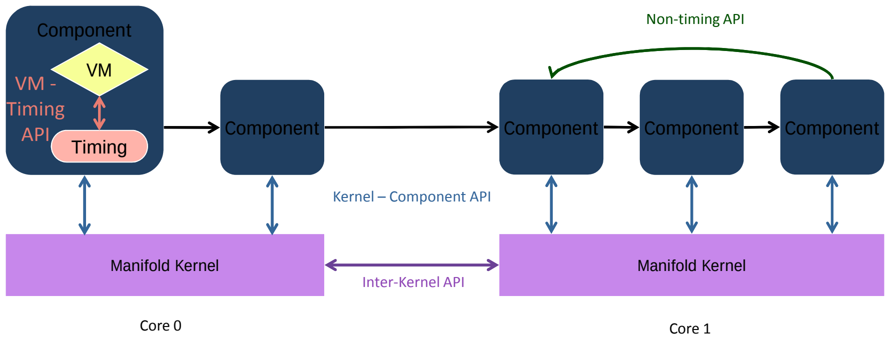
Separation of Timing and Functionality



- Granularity of the timing model & time synchronization
  - Sampling interface
- Direct interface into the code cache
- Timing model feedback
  - Synchronized advance of functional and timing models rather than roll-back and recovery
- Modeling I/O

Manifold Execution Model

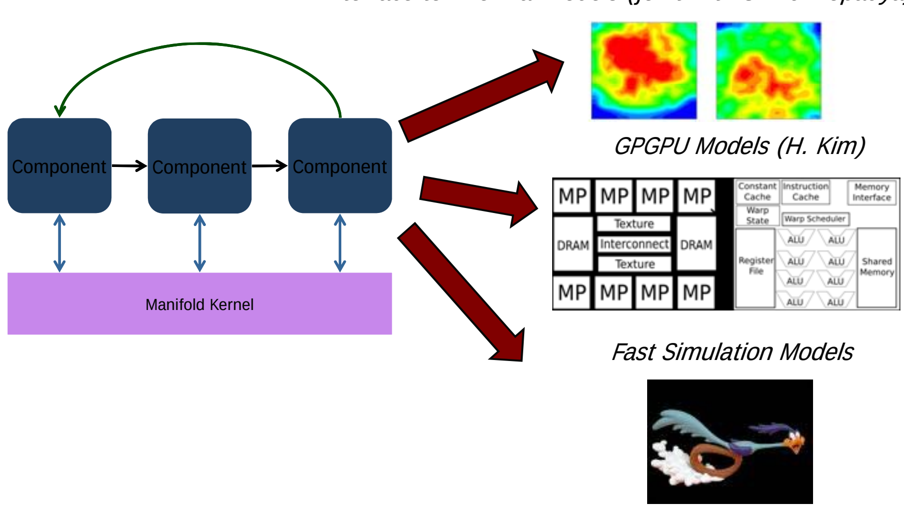
NSF Manifold: S. Yalamanchili, T. Conte, G. Riley



- Components, virtual machines, models, and logical processes
- Scalable parallel kernel to manage progression of time
- APIs → key to integrating mature point tools
  - Introspector API not shown (later)

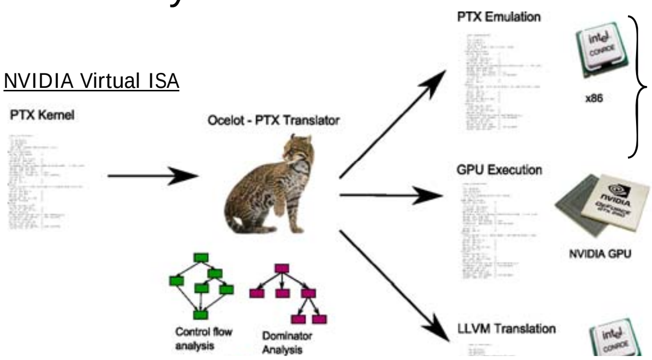
Extensions to the Core Framework

Interface to Thermal Models (joint with S. Mukhopadyay)



Ocelot: Dynamic Execution Infrastructure

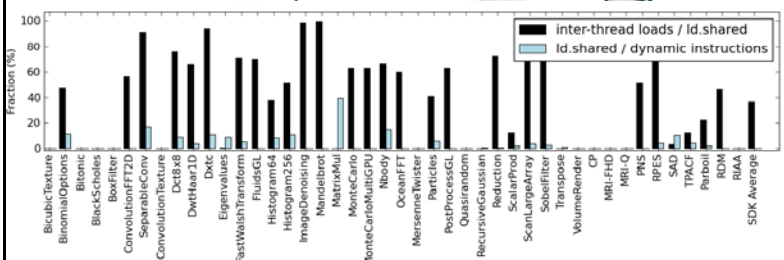
NVIDIA Virtual ISA



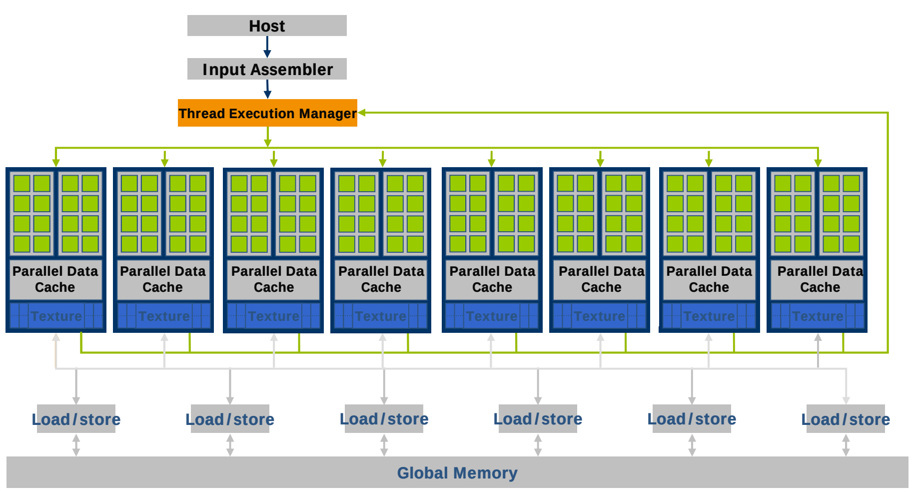
- PTX 1.4 compliant Emulation
- Validated on full CUDA SDK
- Open Source version released

(G. Diamos and A. Kerr)

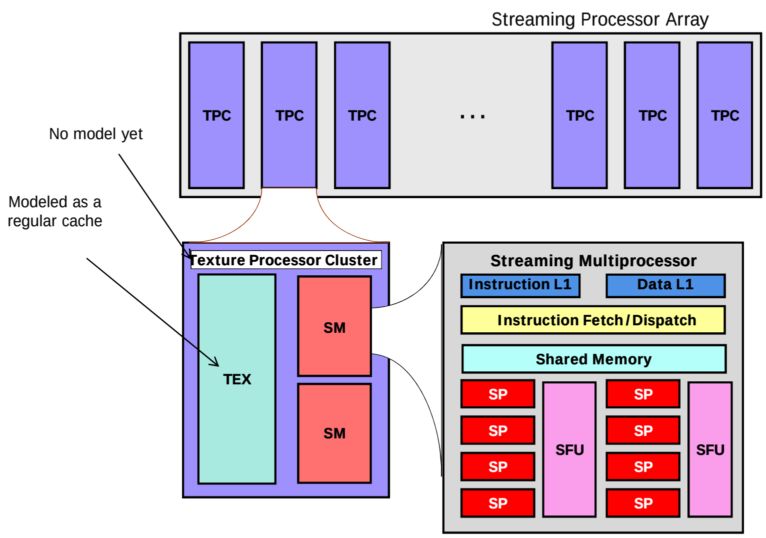
Vector (SSE) and AMD backends are in progress



Simulating the Tesla Architecture – MACSim (H. Kim)



MACSIM’s GPU SM Core



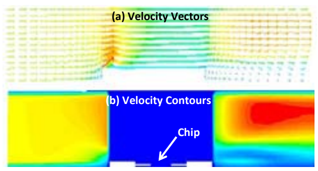
Coordinated Power and Thermal Modeling

S. Mukhopadhyay(ECE), S. Kumar(ME), and S. Yalamanchili(ECE)

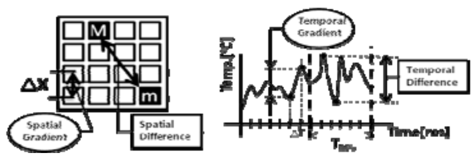
- Scope of architecture simulation has changed
- Need on-line (simulation) analysis of performance, power, and thermal behaviors
  - Architectural techniques for energy management driven by on-line analysis of the thermal field
- Need to couple physics of heat flow with detailed architecture simulation
  - System energy optimization requires trade-offs between compute energy and cooling energy
  - Coupling these models is a challenge!

S. Mukhopadhyay’s work was supported in part by Intel and a IBM Faculty Partnership Award

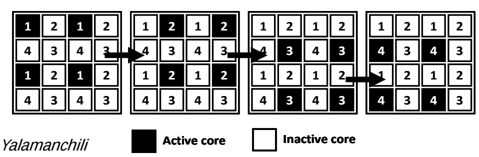
Modeling and Simulation Challenges



S. Kumar



S. Mukhopadhyay

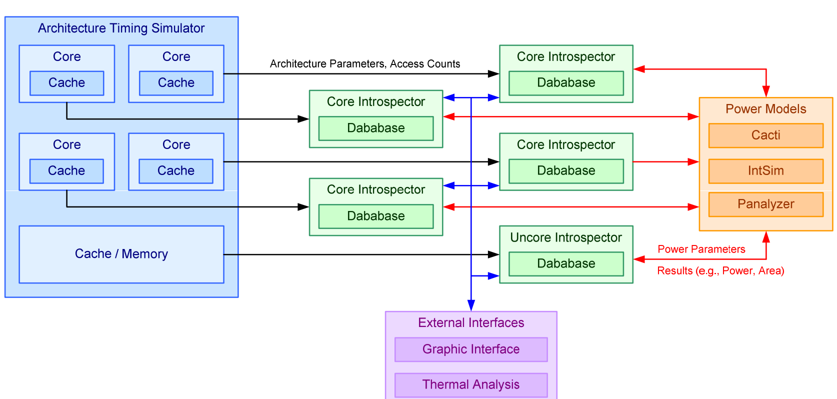


S. Yalamanchili

- How do you couple different simulation models?
  - Differing physical time scales
  - Reduced order computational techniques
- How do we model coordinated mechanisms in a single environment?
  - Time management
- What interfaces do we provide?
  - Manifold/SNL API extensions?

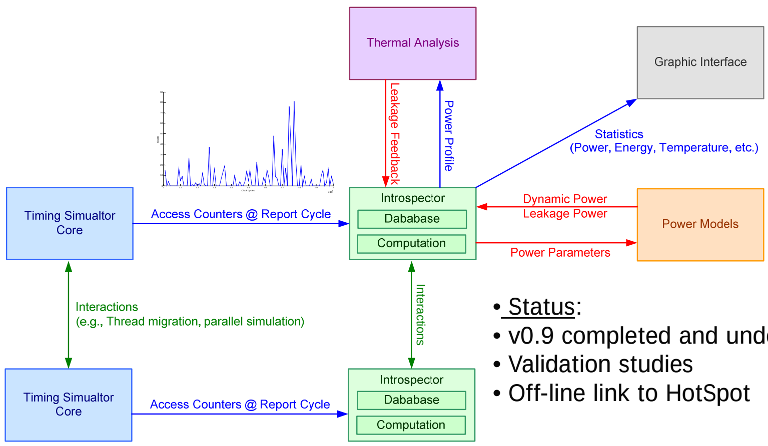
Introspector API

- An Introspector is a pseudo component that monitors a set of model components
- An Introspector provides a uniform interface to physical properties of of a monitored component



Introspector Functionality: Example

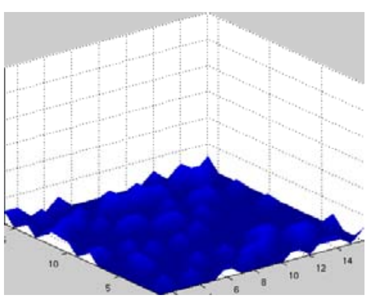
- Timing simulator reports the access counts of the components to the Introspector
  - User selected models
- The Introspector records energy values based on selected models



- Status:
- v0.9 completed and under test
- Validation studies
- Off-line link to HotSpot

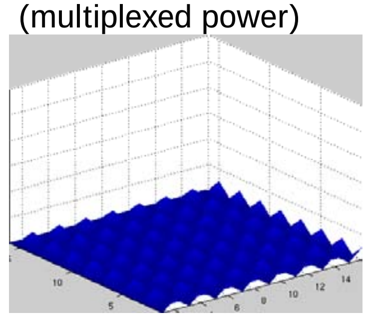
Unmanaged Thermal Behavior

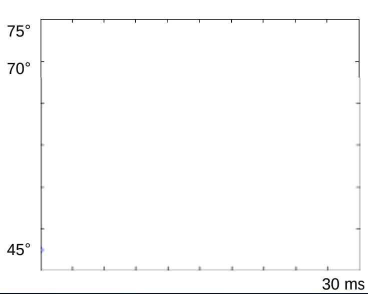
Managed Thermal Behavior (multiplexed power)



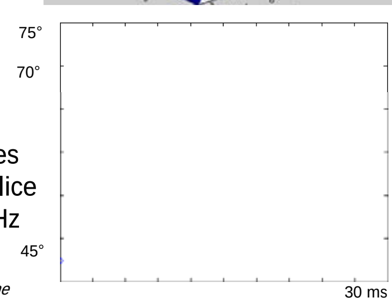
Spatial gradient: 10.5°@0.75mm
Temporal gradient: 2.5°@100Kcyles

Spatial gradient: 2.5°@0.75mm
Temporal gradient: 1.99°@100Kcyles





- 64 on-tiles
- 256 total tiles
- 100K time slice interval@3GHz



Courtesy: Nikil Sathe

Some Research Problems

- Integrating interconnection and memory subsystems
  - #memory controllers, address space mappings, routing, topology
- Energy management (with S. Mukhopadhyay)
  - Application signatures → runtime management
  - Integration of non-volatile memory technologies
  - Integrated package-level power and thermal management
- Algorithm-architecture analysis
  - Data mining and (fast) model construction

Concluding Remarks

- Integration with Sandia SST in Q4/2010
  - X86 processor models
  - Execute code generated by stock compilers and boot (version) Linux
- Keep simulation capacity scaling with Moore's Law
- Coordinated with the Manifold Project (NSF)
- A focus on validation infrastructure in 2011
  - Leverage the GreenIT infrastructure

Thank You