Fast Functional Simulation with Parallel Embra

MoBS 2008

Bob Lantz
Computer Systems Laboratory
Stanford University
(advisor: Mendel Rosenblum)

Complete Machine Simulation

- "Simulators that ignore system effects...are likely to be so inaccurate as to be useless, even for CPU intensive benchmarks like SPECINT 2000." [Cain 2002]
- Publicly available examples: SimOS (-Alpha,-PPC,PHARMsim), SimICS (TFsim, SimFlex), Sparc-sulima, Mambo, M5, PTLsim
- Primary challenge: N-fold (or worse) slowdown for MP simulation!

MP Simulation Approaches

- Exploit Speed/Detail trade-off (e.g. SimOS)
  -> needs fast functional simulator
- Divide/parallelize in simulation *time* (DiST)
  -> needs fast functional simulator
- Statistical/sample in time (e.g. SMARTS, SimFlex)
  -> needs fast functional simulator
- Structural parallelism (we're going to use this to support the above 3 methods)

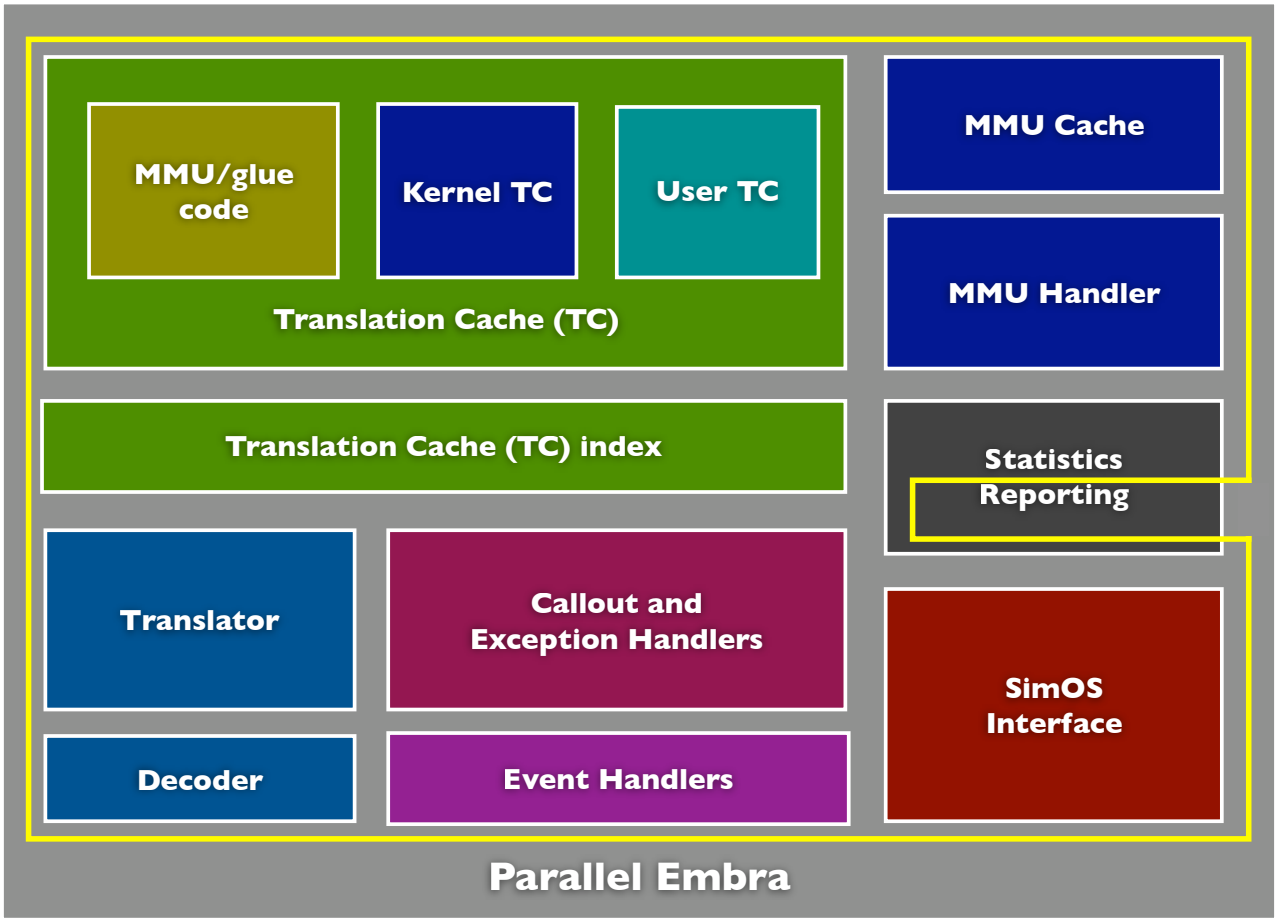
Other Benefits of Fast Functional Simulation

- Support interactive/on-line use
- OS development
- Software (including OS) development for unavailable hardware
- More flexible instrumentation/ introspection/modification than hardware
- Event counts, non-time-critical statistics; order-of-magnitude performance trends

Parallel Embra: a fast functional simulator

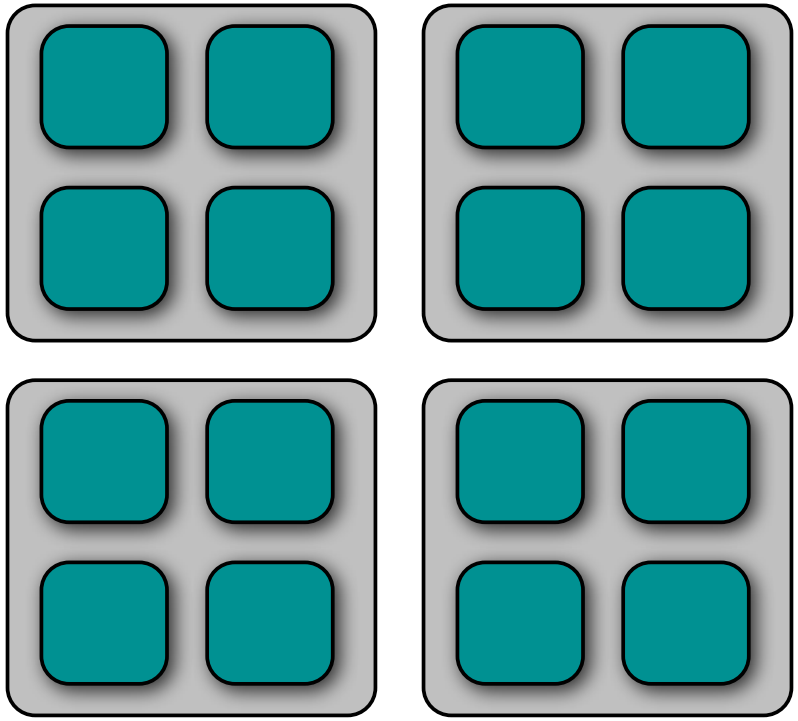
- Parallel, dynamic binary translation-based simulator on/for shared-memory MP
- Integrated into SimOS (now Parallel SimOS) simulator
- Key features:
  - full system
  - runs in user mode (address translation)
  - *directly utilizes underlying shared memory*
  - *loosely synchronized execution*

Parallel Embra Architecture



Parallel Organization

- Use thread-based parallelism, shared memory system of shared-memory multiprocessor
- Divide simulated hardware at host hardware granularity
- Preserves locality; parallelism and increased memory bandwidth reduce linear slowdown and resource exhaustion



The cost of speed: non-determinism

- Direct shared memory access for performance
- Exposes timing, loses determinism
- Lack of repeatability is a pain, but:
  - real systems non-deterministic
  - non-determinism is useful
  - compatible with repeatable mode
  - self-hostable

Parallel Simulator Synchronization

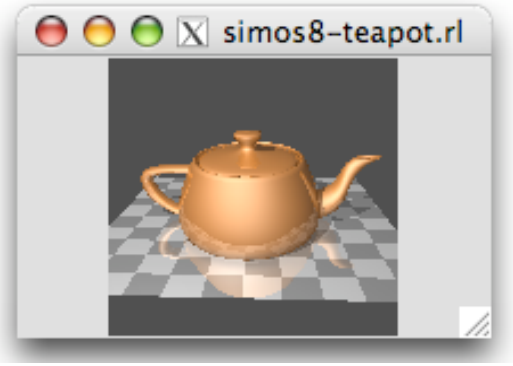
- Parallel data structure access must be synchronized
- Simulator resembles multiprocessor OS:
  - thread-based parallelism
  - fine-grained locking
  - non-blocking synchronization
  - painful to do!

Virtual CPU (non-) synchronization

- Address translation maps virtual to simulator memory, which is accessed *without* synchronization
- Configurable barrier-based synchronization to manage time/cycle skew
- But, this introduces delays/slowdown
- Solution: don't synchronize cycles (use global clock when needed)

But, is it (functionally) correct?

- Yes! Since hardware supports same memory consistency model as simulation
- Synchronization, memory access, other events aren't reordered inconsistently across real or virtual processors
- Also verified implementation by comparing results with hardware, sequential simulation



Performance: Test Configuration

Stanford FLASH Multiprocessor

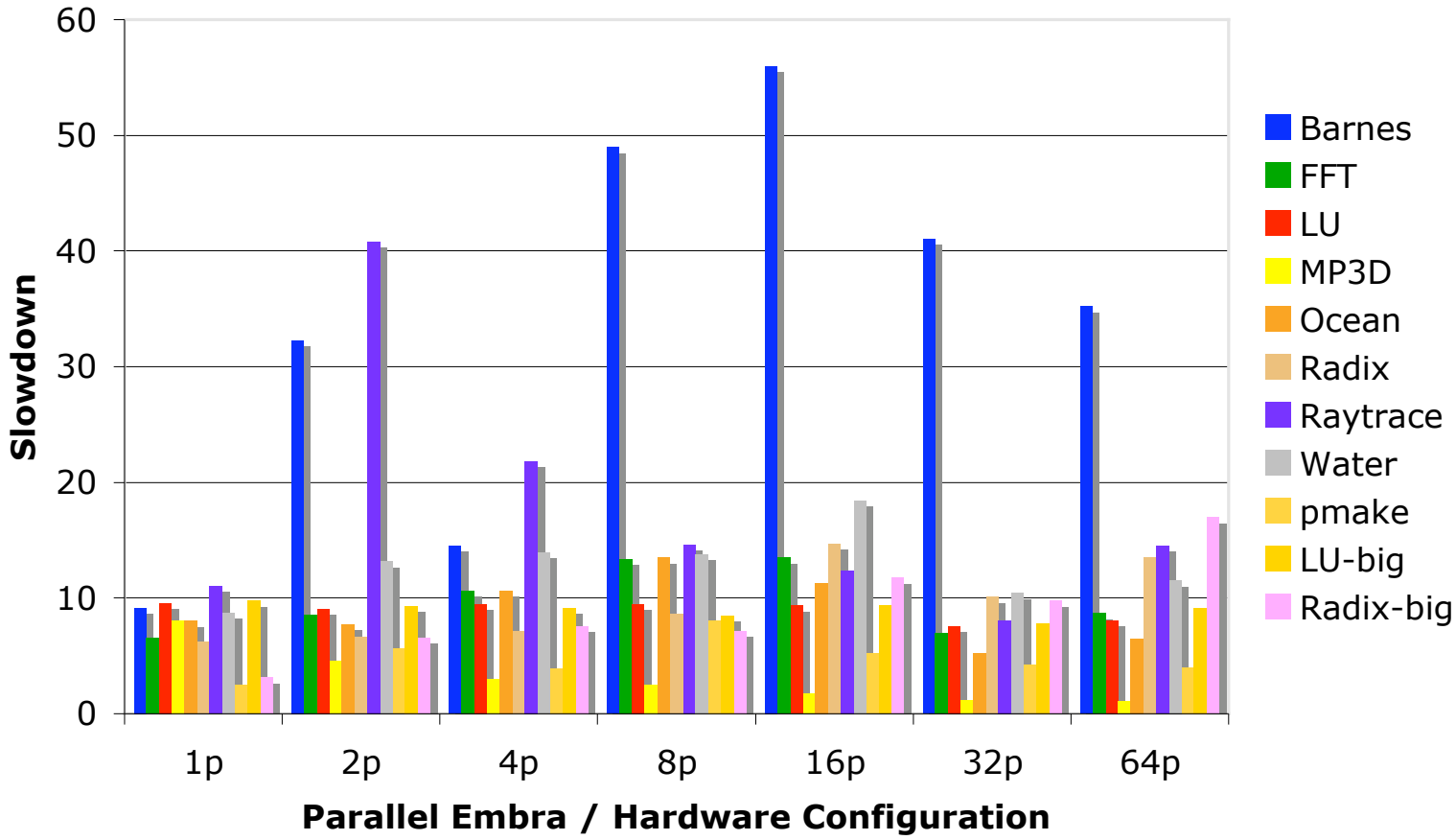
64 nodes
MIPS R10000, 225 Mhz
220 MB DRAM/node (14GB total)
flash1, flash32, flash64, etc.

Machine

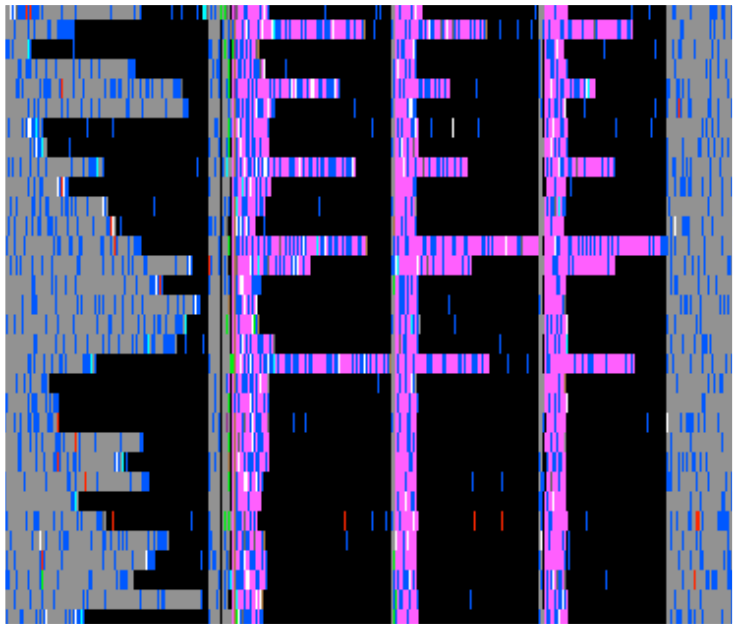
benchmark	*description*
Barnes	Hierarchical Barnes-Hut method for N-body problem
FFT	Fast Fourier Transform
LU	Lower/Upper matrix factorization
MP3D	Particle-based hypersonic wind tunnel simulation
Radix	Integer radix sort
Raytrace	Ray tracer
Ocean	Ocean currents simulation
Water	Water molecule simulation
pmake	Compile phase of Modified Andrew Benchmark
ptest	Simple benchmark for sanity check/peak performance

Workload

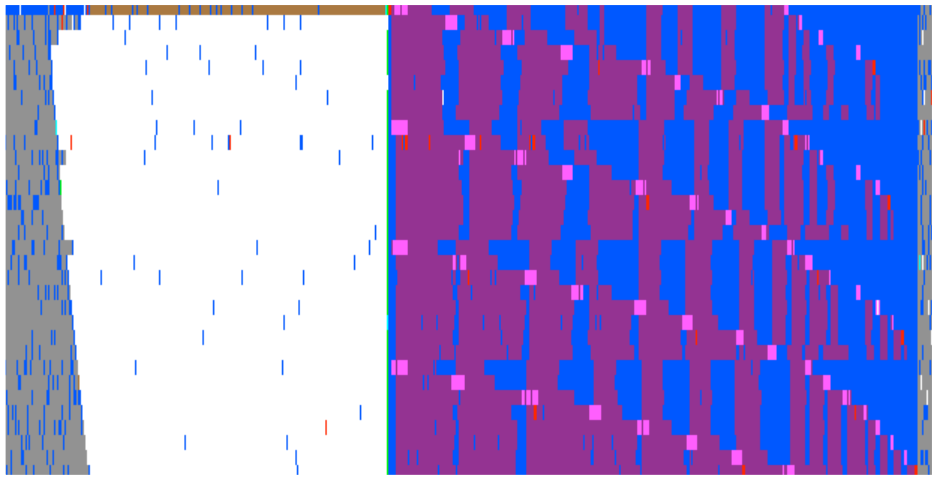
Performance: hardware self-relative slowdown



Performance issues: Imbalance, Serial phases

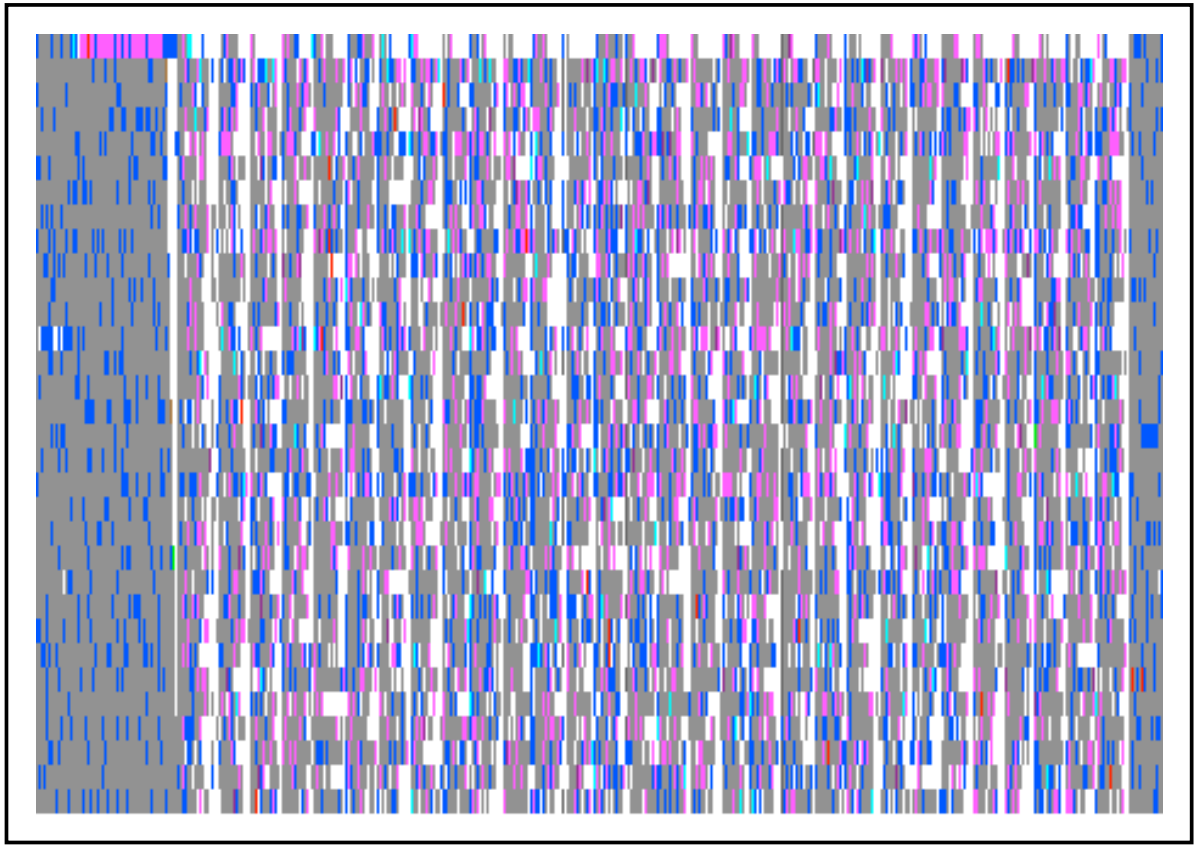


Barnes-Flash32



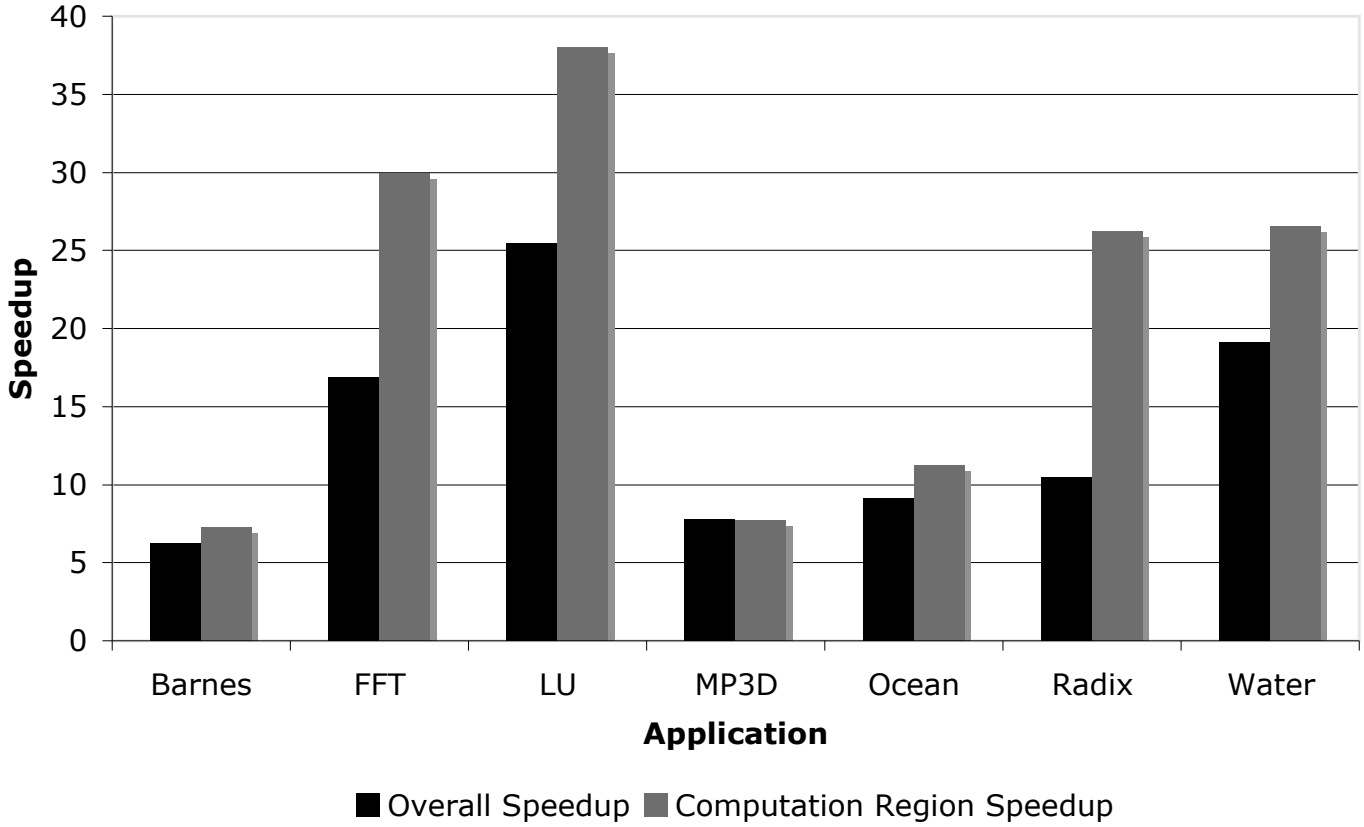
LU-Flash32

Performance issues: Contention

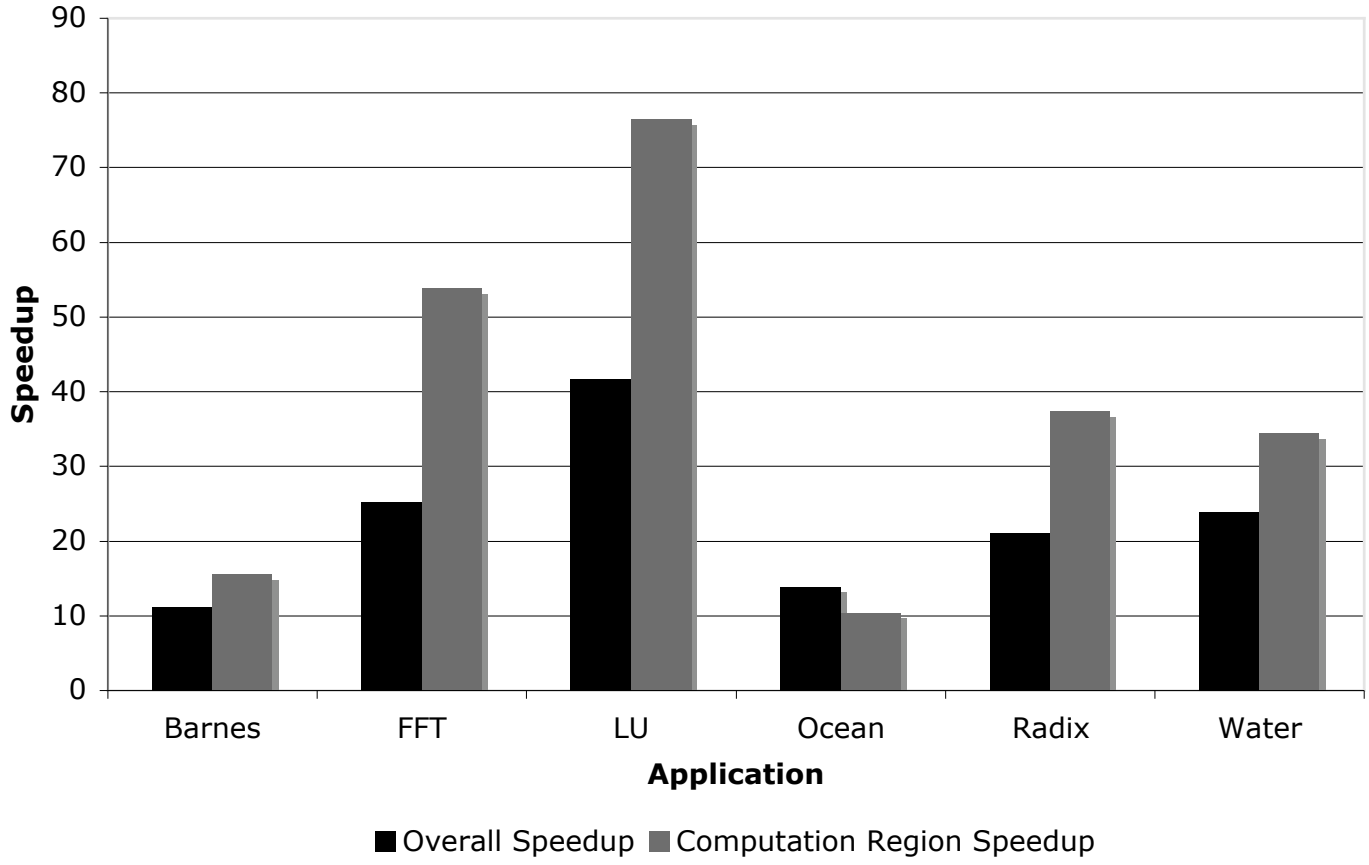


MP3D-Flash32

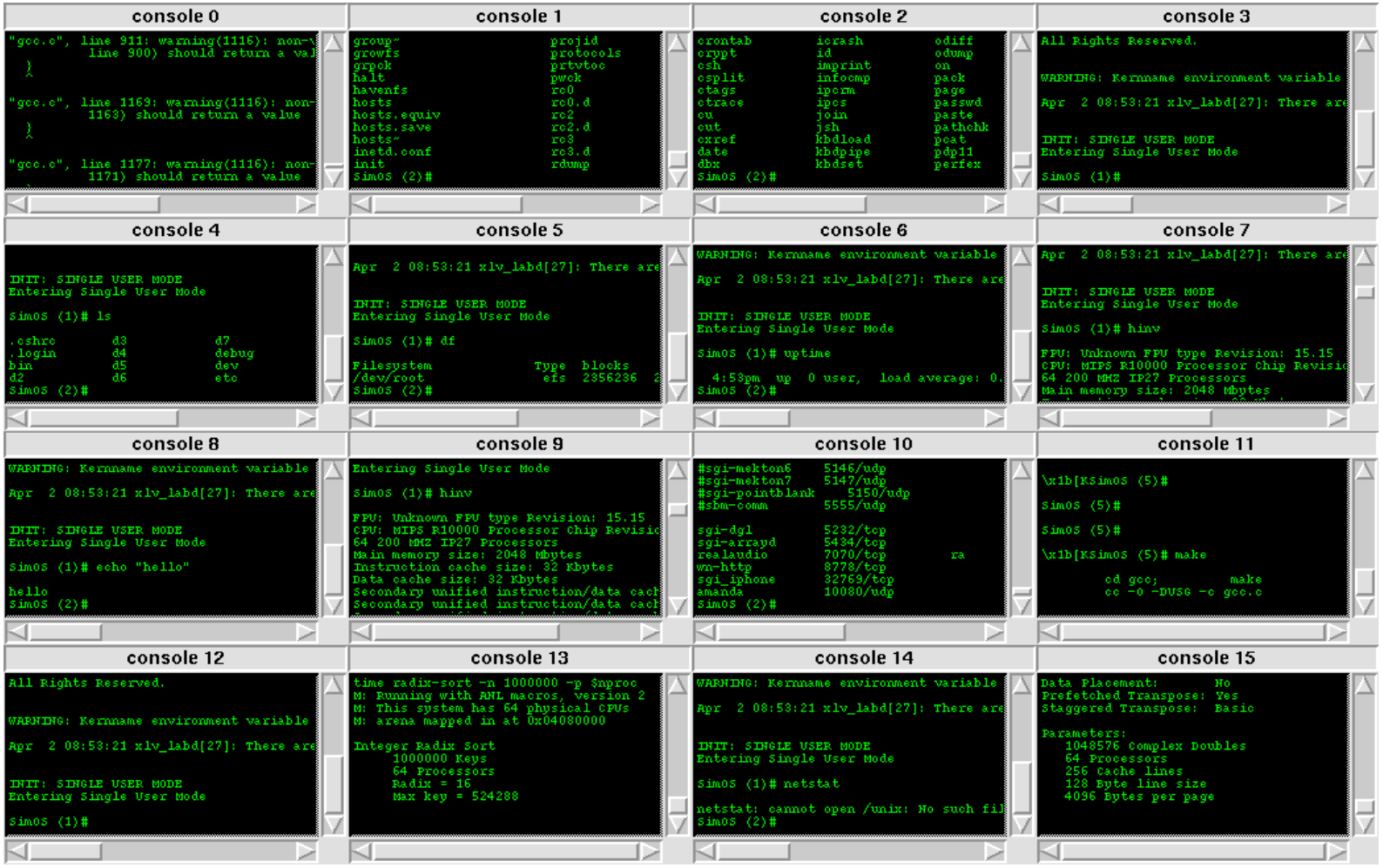
Performance: Self-relative Speedup (32p)



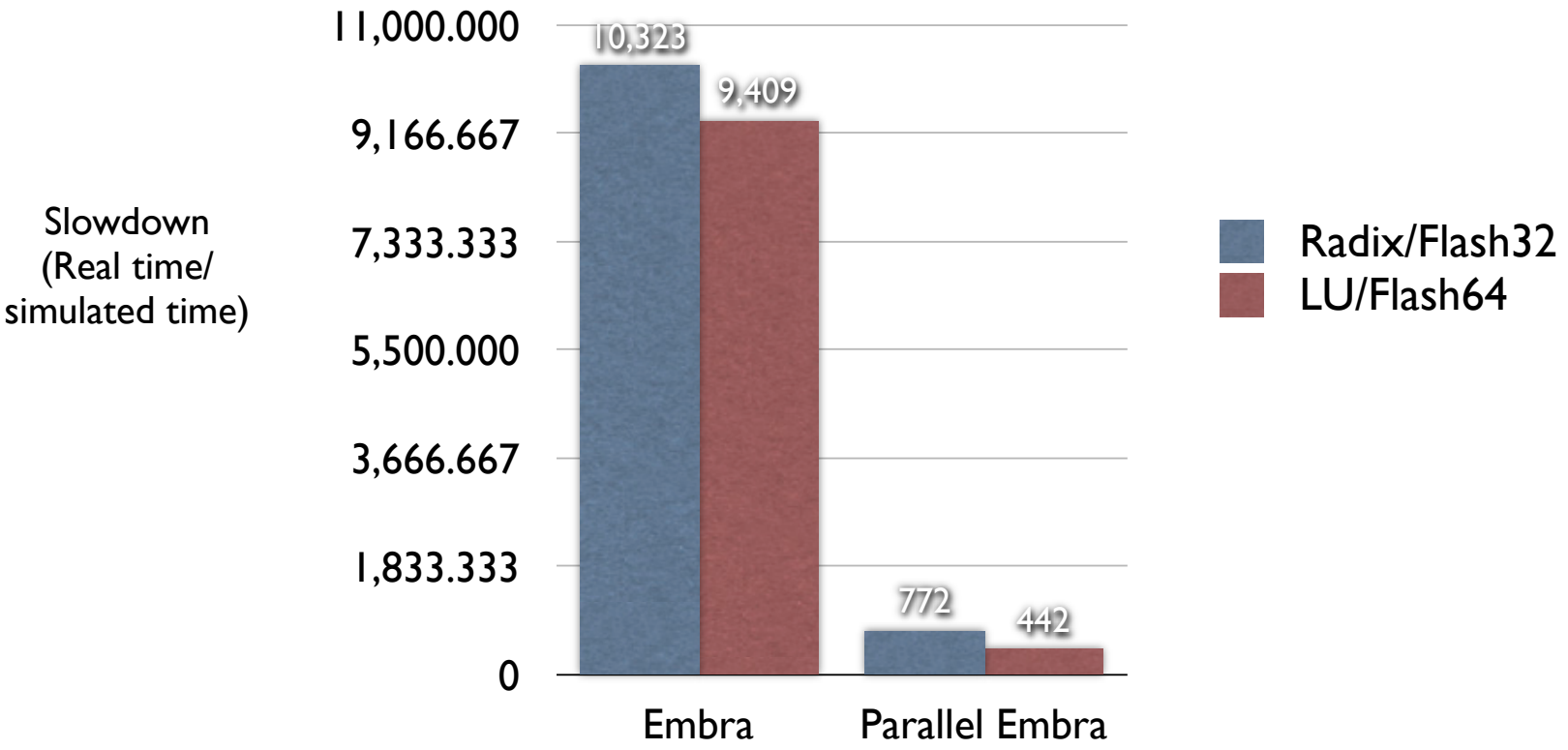
Performance: Self-relative Speedup (64p)



Scalability: 1024p system boot



Scalability: 1024p performance



Hours or days rather than weeks

Related Work

- Parallel Simulation
  - many parallel simulators (WWT...)
  - few parallel complete machine simulators (Mambo, Sparc-sulima: more detail)
- Direct Execution (SimOS, PTLsim)
- Workload Reduction (WCWS)
- Hardware-based simulation (RAMP, FAST, ProtoFlex, WARP workshop!)

Parallel Embra Conclusions

- A complementary and practical approach to parallel functional simulation
  - thread-based parallelism
  - divide at granularity of host hardware
  - use underlying shared memory system
  - avoid synchronizing virtual time
- Scales complete machine functional simulation up to 64p hosts, 1024p targets
- Supports speed/detail, division in time, sampling/statistical simulation methods

Parallel Embra Conclusions

Exciting developments in Speed vs. Detail/ Flexibility in parallel complete machine simulation!

↑ Faster Simulation Speed
Parallel Direct Execution (e.g. PTLsim)
Parallel with Paging (e.g. Parallel Embra)
Parallel with Caches (e.g. Sparc-sulima passive)
Parallel Deterministic (e.g. Sparc-sulima active)
Parallel Microarchitecture (future work)
↓ Greater Simulation Detail/Flexibility