Combining Parallelism, Virtualization, Heterogeneity and Reliability: Some current HPC Research at The Australian National University

Peter Strazdins
Computer Systems Group,
Department of Computer Science,
The Australian National University

seminar at Sun Microsystems Laboratories, Menlo Park, 29 October 2008

(slides available from http://cs.anu.edu.au/~Peter.Strazdins/seminars)

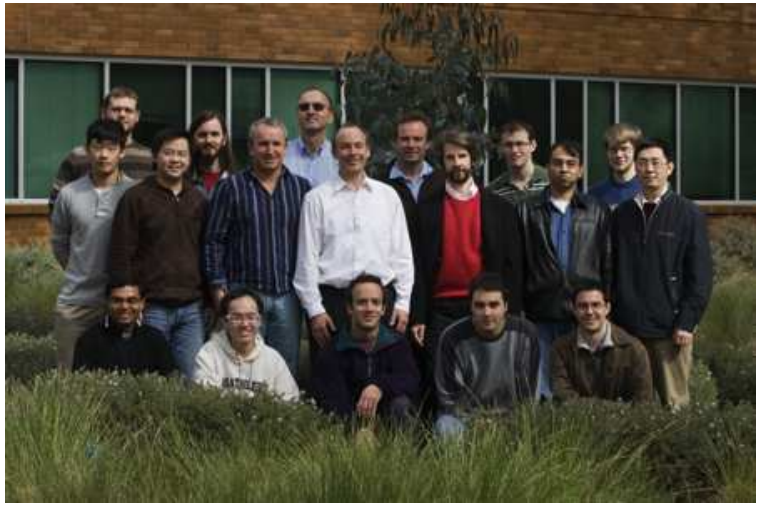
THE AUSTRALIAN NATIONAL UNIVERSITY

1 Overview

- overview of research teaching by the Computer Systems Group, ANU
- OpenMP for contemporary clusters
  - state of the art; handling of heterogeneity
  - utilization of advanced networking technologies
- high performance numerical computing on service-oriented middleware
  - provide fault-tolerance & handling of heterogeneity, *and* performance?
- virtualized HPC Clusters
  - motivations: data center with a heterogeneous cluster of sub-clusters
  - evaluating performance of communication configurations
  - framework for scheduling with virtual machine migration
- NUMA simulation tools and frameworks
  - UltraSPARC SMP: memory modelling validation, parallelization
  - Opteron: parallel Valgrind, NUMAGrind
- multicore computing (future plans)

2 Overview of Research at ANU's Computer Systems Group

- Robotics (Zimmer): autonomous submersibles & aircraft
- Bio-Engineering (Rendell, Blackburn)
  - multidisciplinary collaborative project with IBM Research
  - new tools for study of ion channel systems
  - involves X10 development
- Performance Analysis:
  - development of CC-NUMA simulator (Strazdins)
  - DaCapo: de facto standard Java benchmarking suite (Blackburn)

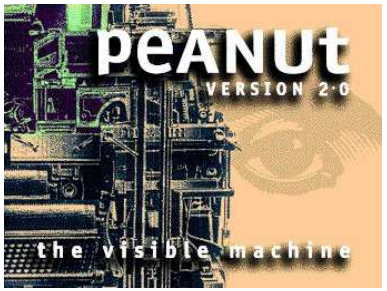


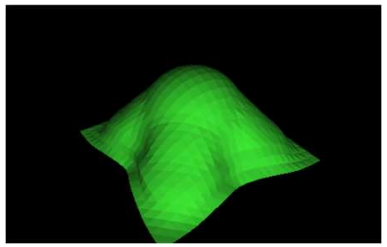
3 Overview of Research at ANU's Computer Systems Group (II)

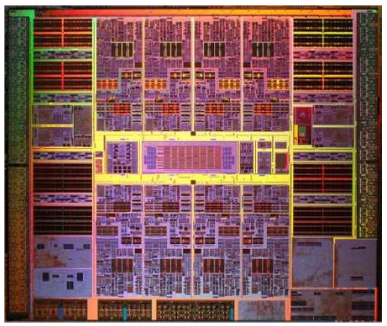
- Parallel Processing:
  - computational chemistry on cc-NUMA project (Rendell; Sun / Gaussian)
  - CBE/GPU implementations (Rendell, McCreath)
  - cluster OpenMP (Rendell, Strazdins; Intel)
  - numeric HPC on service-oriented architectures (Strazdins; Platform)
- Operating Systems:
  - application of virtualization to HPC (Strazdins)
  - Linux kernel development (McCreath)

4 Overview of Teaching in Computer Systems at ANU

- COMP2300 Introduction to Computer Systems: C, assembler, architecture and OS concepts
- COMP2310 Concurrent and Distributed Systems
- COMP3300 Operating Systems: inside the kernel!
- COMP3310 Computer Networks
- COMP3320 High Performance Scientific Computation
- COMP4330 Real-Time and Embedded Systems
- COMP4300 Parallel Systems
- COMP8320 Multicore Computing (for 2009)
- project courses at 3rd year, Hons & Masters levels
- specializations in Bachelors and Masters courses







5 Clusters at the Computer Systems Group

- saratoga: 8 ($\times$ 2 CPU) Opterons with GigaE
  jabberwocky heterogeneous cluster
  - hardware donated by Alexander Technology, mid 2006
    - VorpalBlades: 4 ($\times$ 2-4 CPU) Opterons with GigaE
    - SlithyToves: 4 ($\times$ 1-2 CPU) Opterons with GigE and Quadrics
    - TulgeyWood: remote node imaging file repository

  and its growing like the mythical Jabberwocky . . .
    - augmented with a 4-node quad-core Infiniband IB cluster (2008)

6 The Grid-Enabled Clusters Project

- scenario: a ‘compute center’ serving several users (with scientific applications)
  - heterogeneity is desirable, inevitable!
- aim: develop intelligent runtime environments
  - select where to execute a job, based on cost / benefit, current load
- approach:
  - develop performance-monitoring and characterization infrastructure into runtime environments
  - sophisticated scheduling strategies
  - use of virtualization by Xen to aid migration and operating system customization
    - migration for finding optimal sub-cluster, load balancing and consolidation

7 Communication Configurations of SMP Clusters with Multiple GigE Interfaces

- motivation:
  - low-end SMPs (esp. with multi-cores) are highly cost-effective for computational power
  - communication performance of COTS clusters has not kept up
  - many COTS motherboards support multiple I/O connections
    - e.g. IWILL DK8-HTX (Opteron) has three PCI-X / PCI buses and 14 device slots
    - adding network interface cards is at small relative cost
    - could these be used to balance a cluster's communication performance?
  - (Gigabit) Ethernet (GigE) is the most widely used interface: highly cost-effective
- can virtualized clusters also take advantage of this?

8 Background: Virtualization under Xen

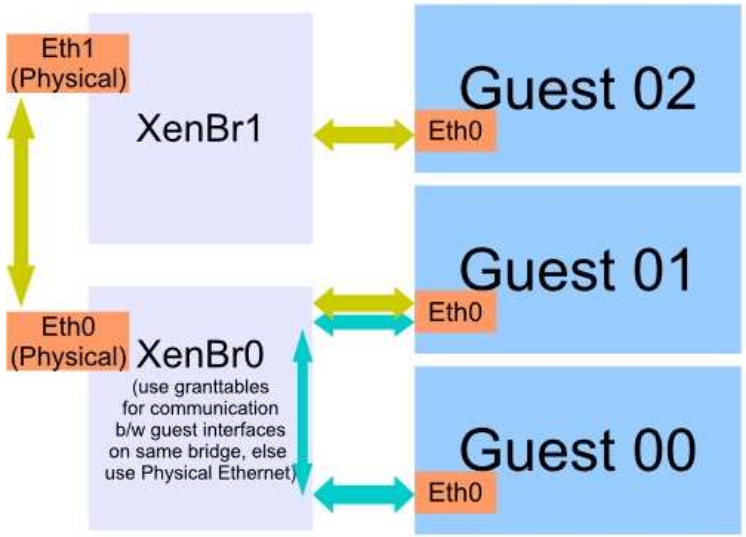
- virtualization offers many advantages (greater encapsulation), but typically comes at a significant performance penalty
- (OS) virtualization: a hypervisor (or virtual machine monitor, VMM) sits at a higher privilege level to the virtualized ('guest') OS
  - direct access to devices, page tables, privileged registers is by by calls to the VMM
- Xen uses para-virtualization: the parts of the (Linux) kernel dealing with the above must be modified ('XenoLinux')
  - offers both potentially high performance and functionality
- requires one special guest OS (domain0, the 'driver domain)':
  - manages (configures, creates, destroys) a number of guest OSs (guest VMs)
  - external communication occurs through a virtual interface:
    - data is transferred by pseudo device drivers to domain0

9 Background: TCP/IP Stack Processing under Linux SMP

- Linux SMP 2.6 has sophisticated TCP/IP stack processing
- uses 2 kinds of locks: connection-related and socket-related (more frequently accessed)
- 2 broad strategies:
  - message-parallel: ||ize the processing (of different segments) of a single transmission
  - connection-parallel: ||ize processing of messages using different sockets (generally superior performance)
- recent studies have shown parallelization incurs some overhead (lock manipulation and cache pollution)
  - explains the generally poor performance of channel bonding

10 Communication Configurations with Multiple Interfaces

- work with Muhammad Atif (formerly Richard Alexander and David Barr, XenHPC'06)
- under Xen, have a virtualized node one each CPU
- can have shared and separate bridges for communication



- can also export NICs to Xen Guests
- baseline of native Linux with one process per CPU (incoming messages for each process routed through separate NICs - MPICH)

11 Micro-Benchmarks: Inter-Domain Communication (2 nodes)

- used modified OSUbench, with MPICH and OpenMPI on VorpalBlades

Comfig/Pairs:	1	2	3	4
Linux-OMPI	125	94	114	123
Linux-MPICH	106	104	124	123
Exported Interfaces	125	112	80	110
Separate Bridges	125	129	125	160
Shared Bridge	126	125	128	149

Latency (μs) at 1 Byte

1	2	3	4
109	199	199	248
109	218	202	240
109	197	326	408
109	161	190	194
108	124	136	126

Latency (MB/s) at 4 MB

Comfig/Pairs:	1	2	3	4
Linux-OMPI	109	165	300	397
Linux-MPICH	105	186	305	366
Exported Interfaces	105	183	312	394
Separate Bridges	102	182	177	142
Shared Bridge	102	119	105	96

Avg. Bandwidth (MB/s), size $\geq$4K

1	2	3	4
124	296	337	408
124	273	350	361
121	183	370	415
115	183	177	168
115	119	105	111

Avg. Bi-Band. (MB/s), size $\geq$4K

12 Benchmarks: Intra-Domain Communication (2 nodes)

- OSU micro-benchmarks reveal heavy overheads from Xen:

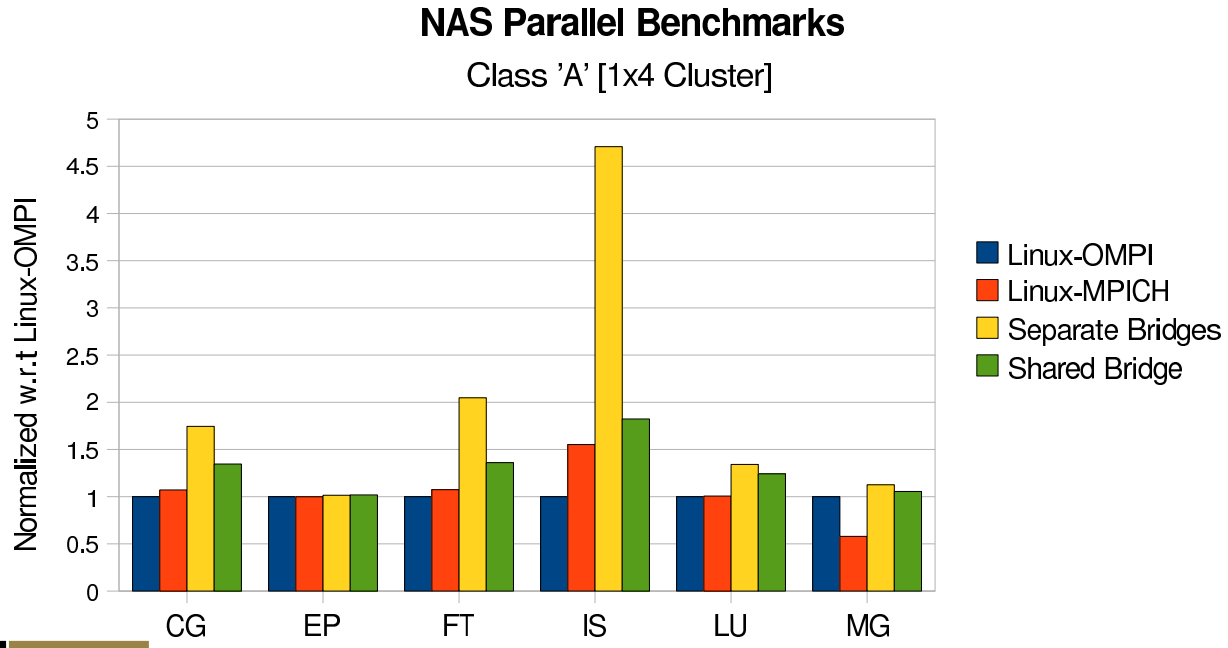
Comfig/Pairs:	1	2
Linux-OMPI	1250	1804
Exported Interfaces	87	196
Separate Bridges	84	94
Shared Bridge	344	300

Avg. Bandwidth (MB/s), size $\geq$4K

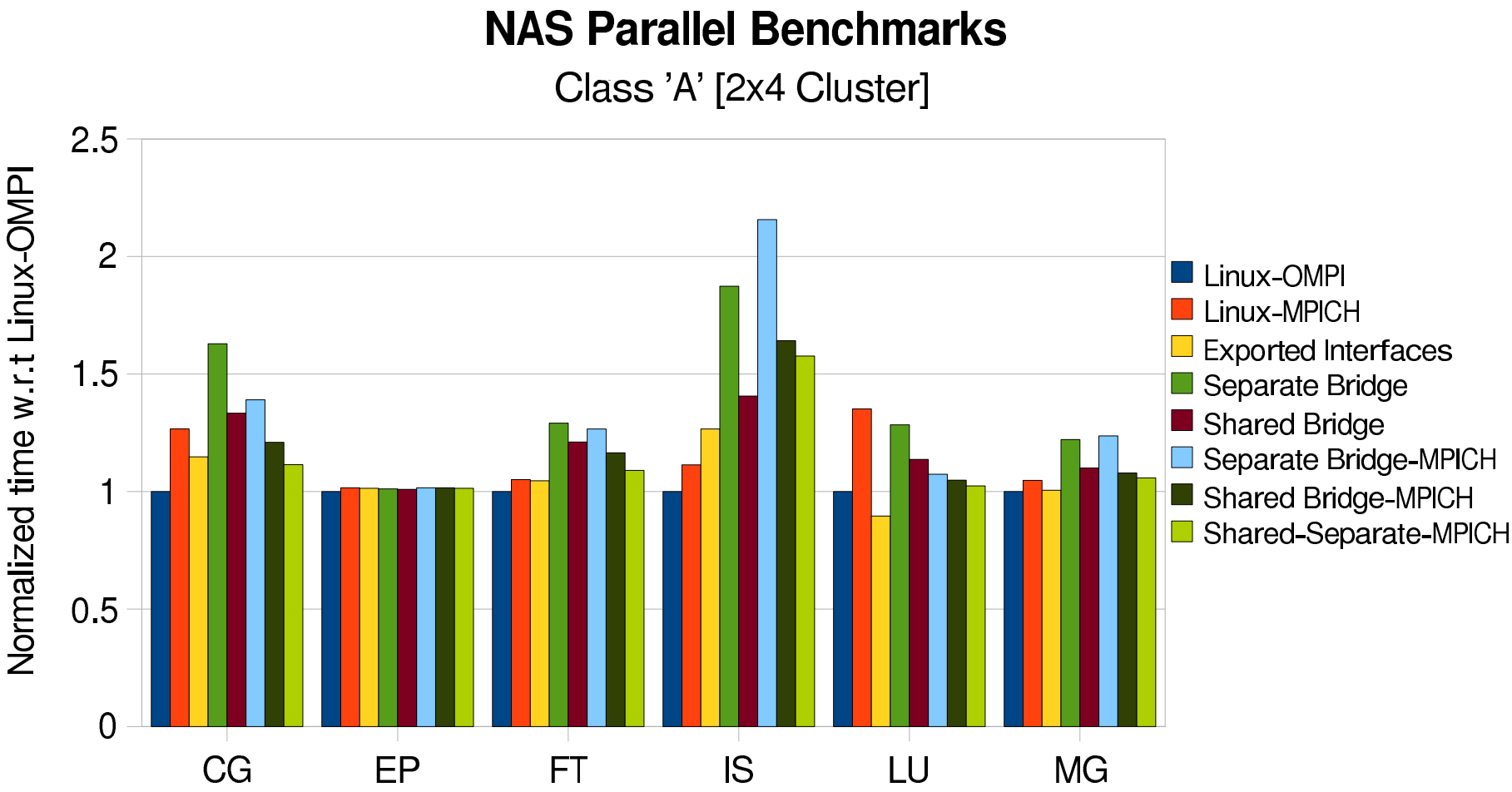
1	2
1163	1456
82	127
85	98
366	338

Avg. Bi-Band. (MB/s), size $\geq$4K

- using NAS Parallel Benchmarks (class A) on a 1×4 cluster:



13 Application-Level Micro-Benchmarks on a 2×4 Cluster



- weakened intra-domain performance in Xen seems hidden!

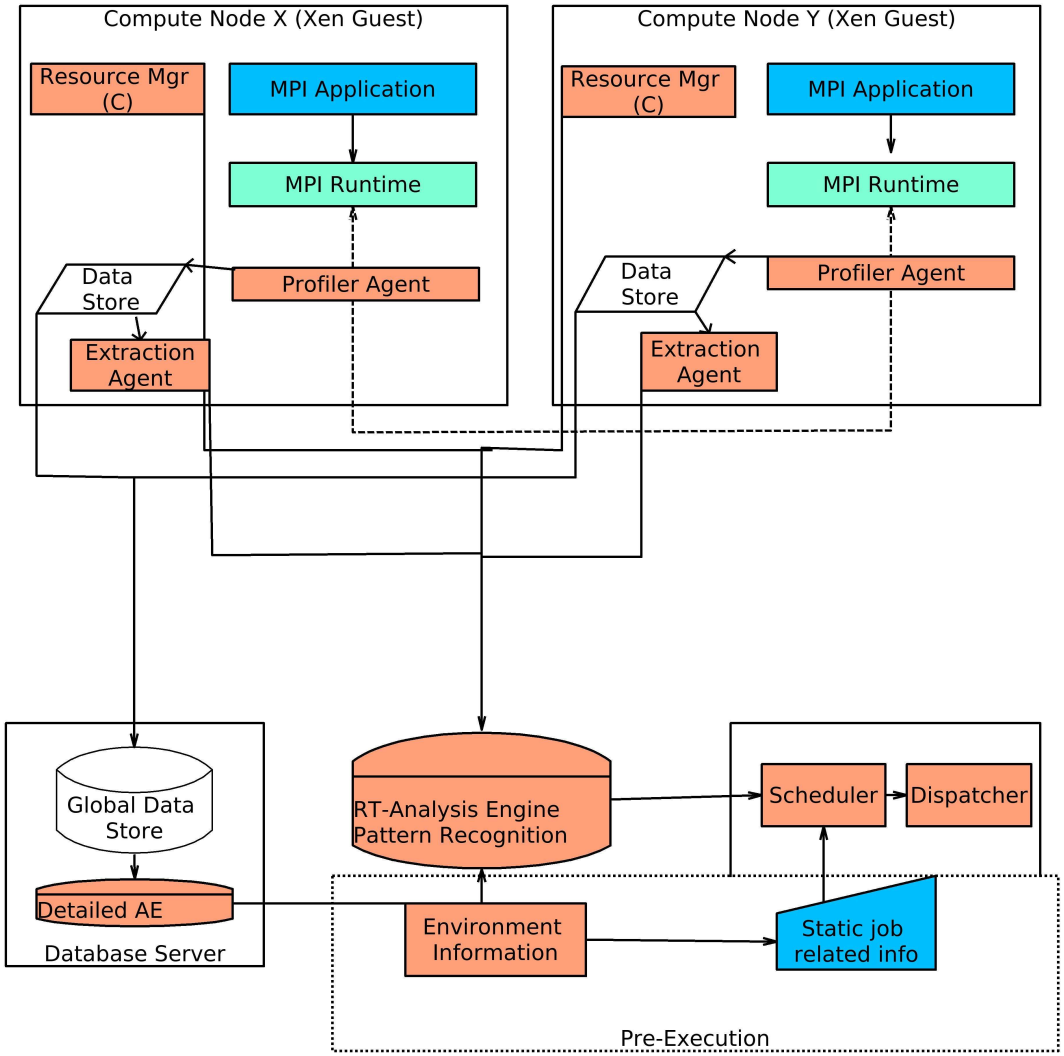
14 Attempts at Improving Xen Intra-Domain Performance

- XenSocket (IBM, 2007): for faster communication between Xen Guests
  - 1-way only, no polling or select
  - bare XenSocket comparable to native Linux (shared memory) transfer for OSU Bandwidth (only)
- attempt to add a Byte Transfer Layer module in OpenMPI
  - dynamically determine if guests are co-located
  - if so, set up alternate socket (pair) to TCP/IP
  - possible (in principle) to cope with migration
  - OpenMPI TCP/IP BTL is heavily callback implemented
- results:
  - only robust for blocking MPI send / receive
  - performance degraded once inside BTL
  - very difficult to work with OpenMPI BTLs!
- there must be a better way!

15 A Scheduling Framework for Heterogeneous Clusters

- Muhammad Atif's PhD topic
- virtualized Jabberwocky multi-cluster (filesystems with NFS, AoE)
- each parallel job runs on own virtualized nodes (one MPI process per VM), which can be migrated
- migration downtime varies with load, $<$ 75 ms
- wish to schedule to select optimal sub-cluster for job, or to load-balance, or to consolidate physical nodes
  - use xenoprofile to determine VM's (hence job's) CPU and memory usage statistics (need multiplexing!)
  - use a profiler within OpenMPI to determine communication characteristics
  - issue: need to determine what physical hosts VMs are running on!
- idea is to profile the running job, update runtime estimation engine about its progress, and migrate if worthwhile

16 Diagram of Scheduling Framework for Heterogeneous Clusters



17 Migration Experiments across Clusters

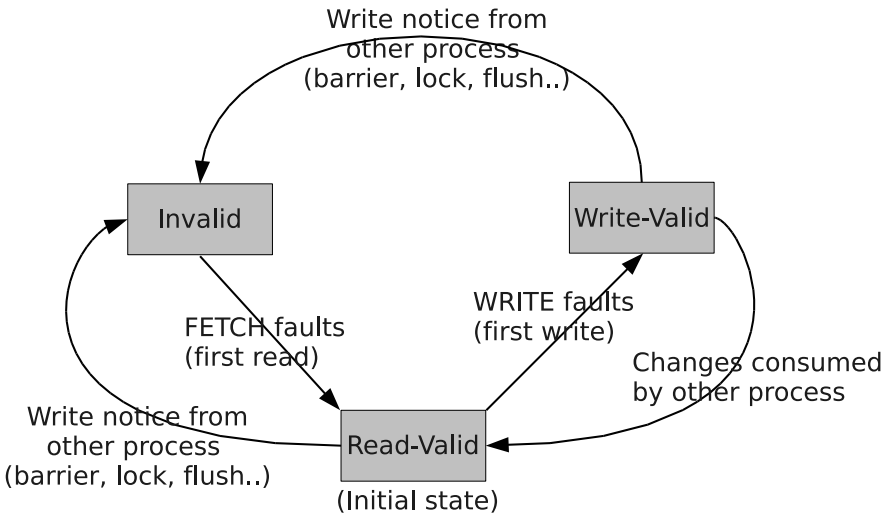
- need to determine overhead of migration
  - different from downtime!
  - need to measure impact on running application
  - as functions of VM memory size, job memory footprint, communication intensiveness, etc
- preliminary results on the HPL benchmark:
  - 15–20s per migration, little affected by memory footprint or load
  - further investigation needed!

18 OpenMP on Advanced Clusters

- with Jie Cai (also Jin Wong & Alistair Rendell)
- two DSM-based systems: Inter Cluster OpenMP (homeless) and Omni/Danui (home-based)
- development of page fault benchmarks
- page fault (segv) based performance models (tools can count segvs)
  - source code analysis technique to count page faults
  - performance valuation / validation on GigE / IB clusters
- heterogeneity modelling: p nodes, execution rates $r_1 \leq r_2 \leq \ldots r_p$
  - $\frac{pr_1T_1}{\Sigma_{i=1}^{p} r_i} \leq T \leq T_1$
  - for NAS CG.A/C, $T \approx T_1$; for BT.A/C, near-ideal $p = 2, 4, 8$; why?
- current / future work:
  - reduce segv cost (IB): use of RDMA / atomic ops / multi-rails
  - utilize of computation/communication overlap and prefetch
  - scheduling for heterogeneity

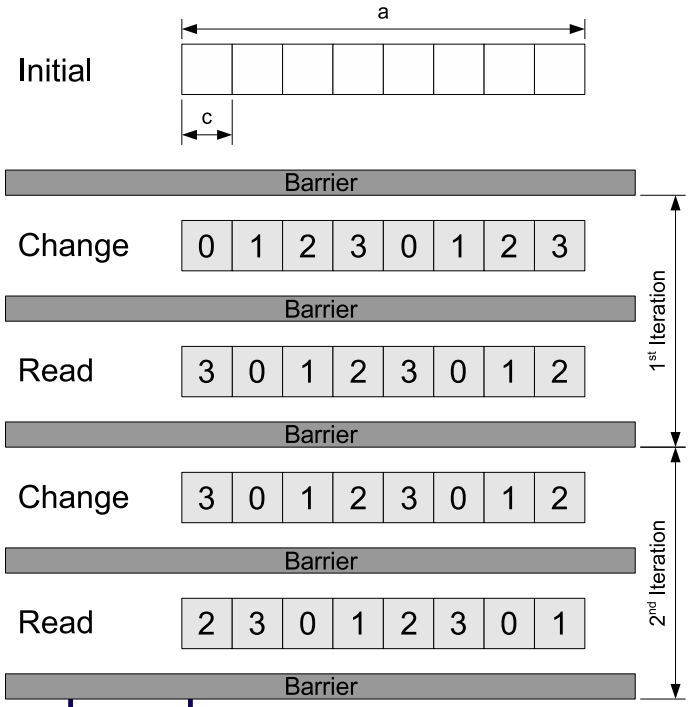
19 Intel Cluster OpenMP

- supported by Intel C/C++/Fortan compiler
- run-time system TMK based on TreadMarks
  - Lazy Release consistency protocol, homeless
  - has CAL communication layer, with TCP and uDAPL interfaces
  - memory is kept consistent through detecting and servicing different type of page-faults
    - write-fault involves making a 'twin' or original page, and subsequently a 'diff'
    - fetch fault involves gathering and applying diffs from previous synchronization point

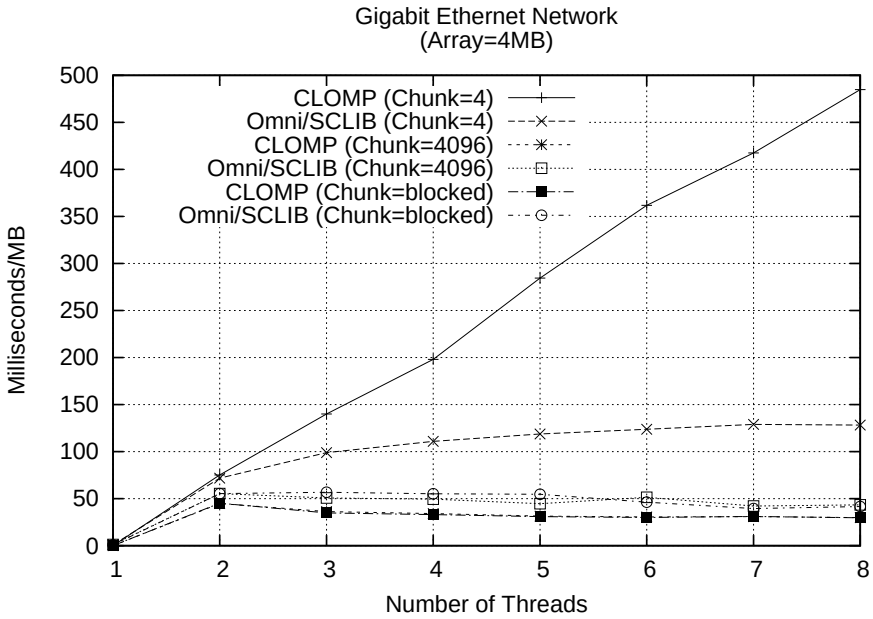


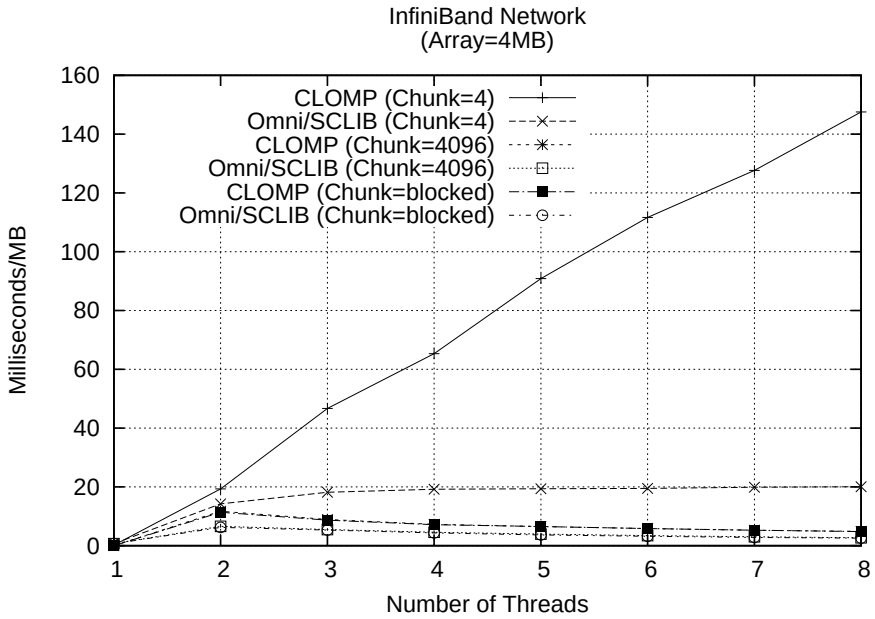
20 Memory Consistency Benchmarks for Cluster OpenMP

- no comparable benchmarks from shared-memory based systems
- generates memory consistency work-loads
  - modify some data
  - OpenMP barrier
  - access data written by another thread
- a reference time from using private data instead of shared
  - difference in elapsed time gives us the overhead
  - `kernel(array, arraysize, chunksize, id, nthreads)`
  - source: http://ccnuma.anu.edu.au/dsm/mcbench



21 Memory Consistency Benchmark Results on Clusters

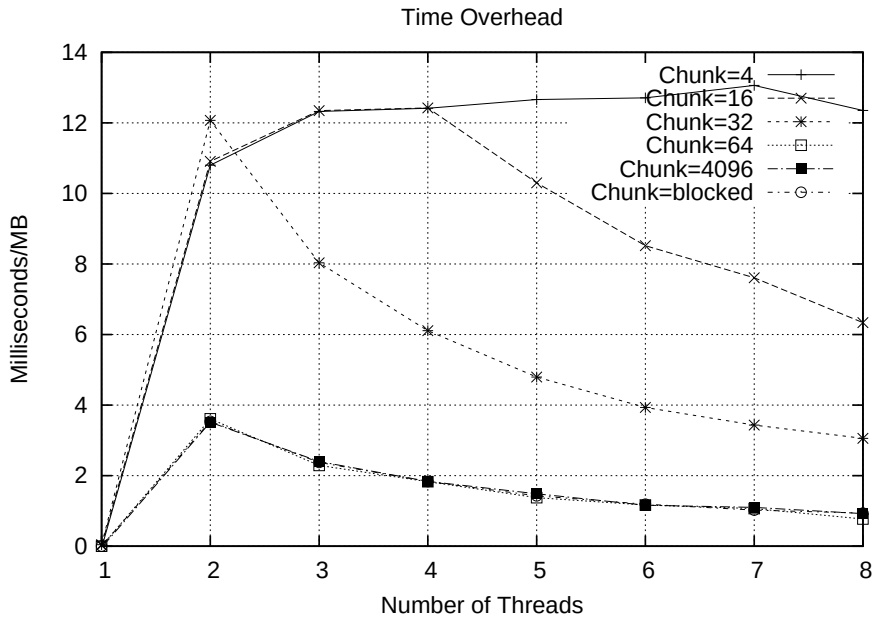


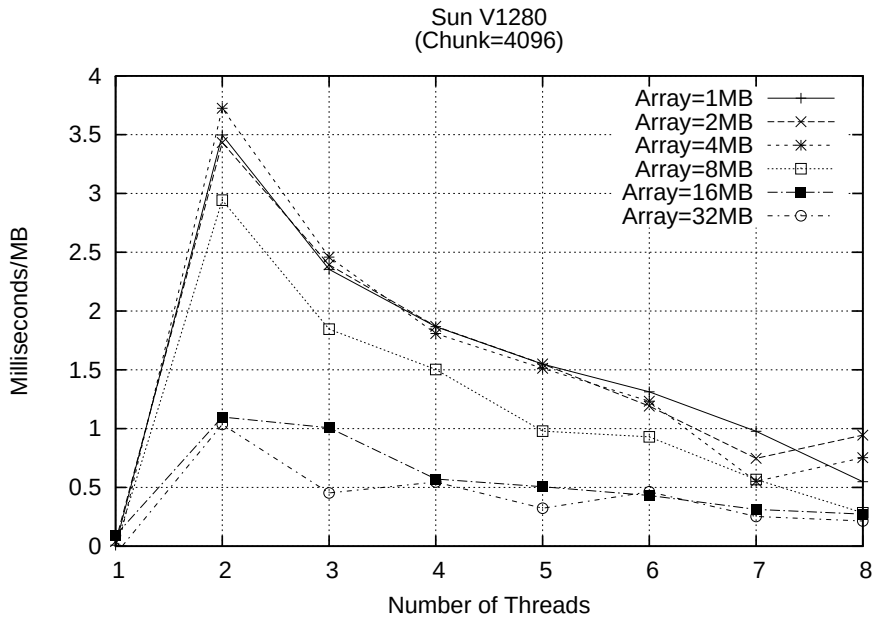


GigE cluster Infiniband cluster

- chunk sizes of 4 bytes, 4 KB and blocked (4 MB / threads)
- home-based DSM has advantage, especially for small chunk sizes
  - $O((p-1)/p)$ scaling
- both have $O((p-1)/p)$ scaling for 4 KB chunks

22 Memory Consistency Benchmark Results on a V1280 Ultra-SPARC SMP



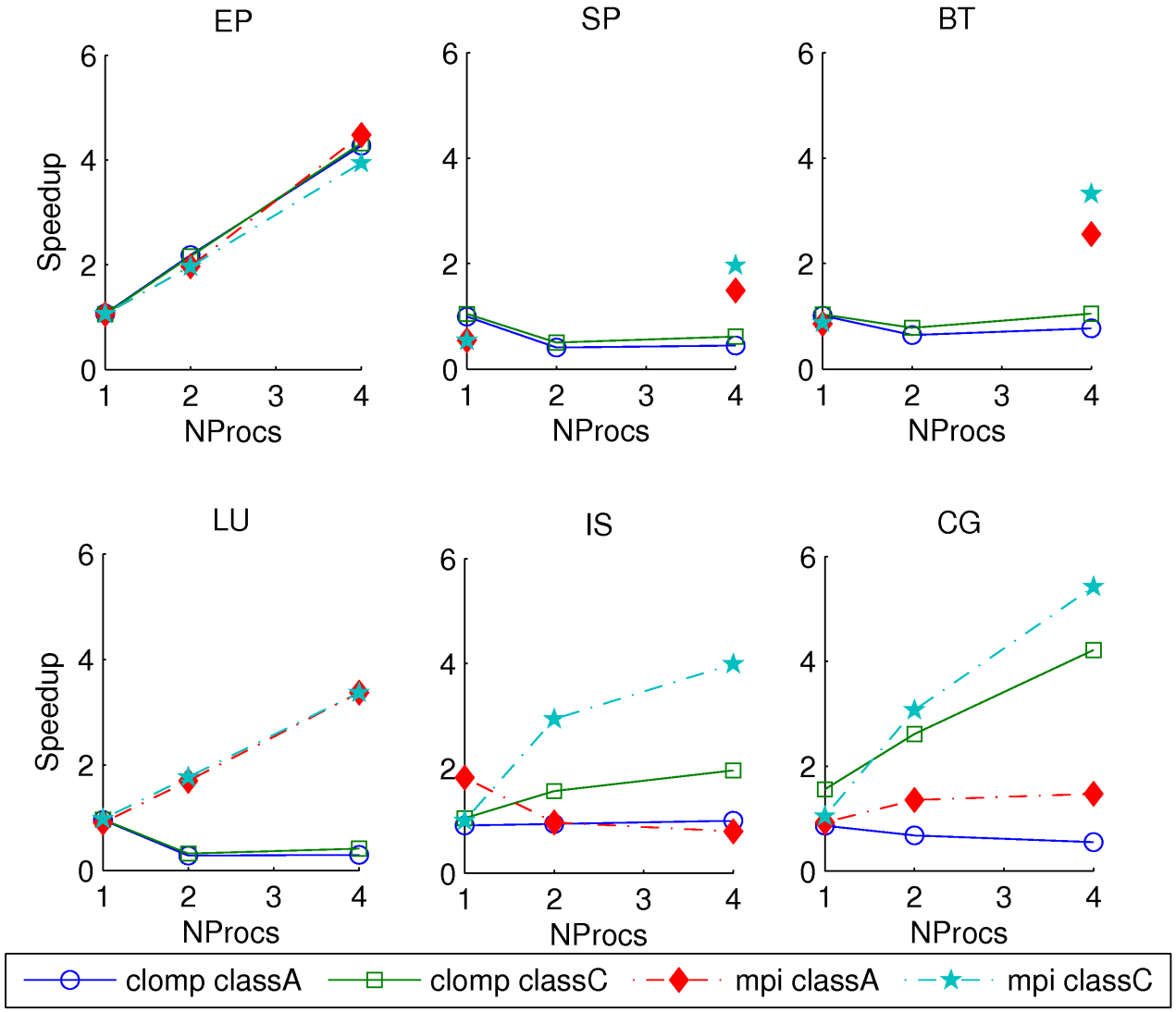


4 MB array,
chunk size from 4 B to (4 MB / threads)

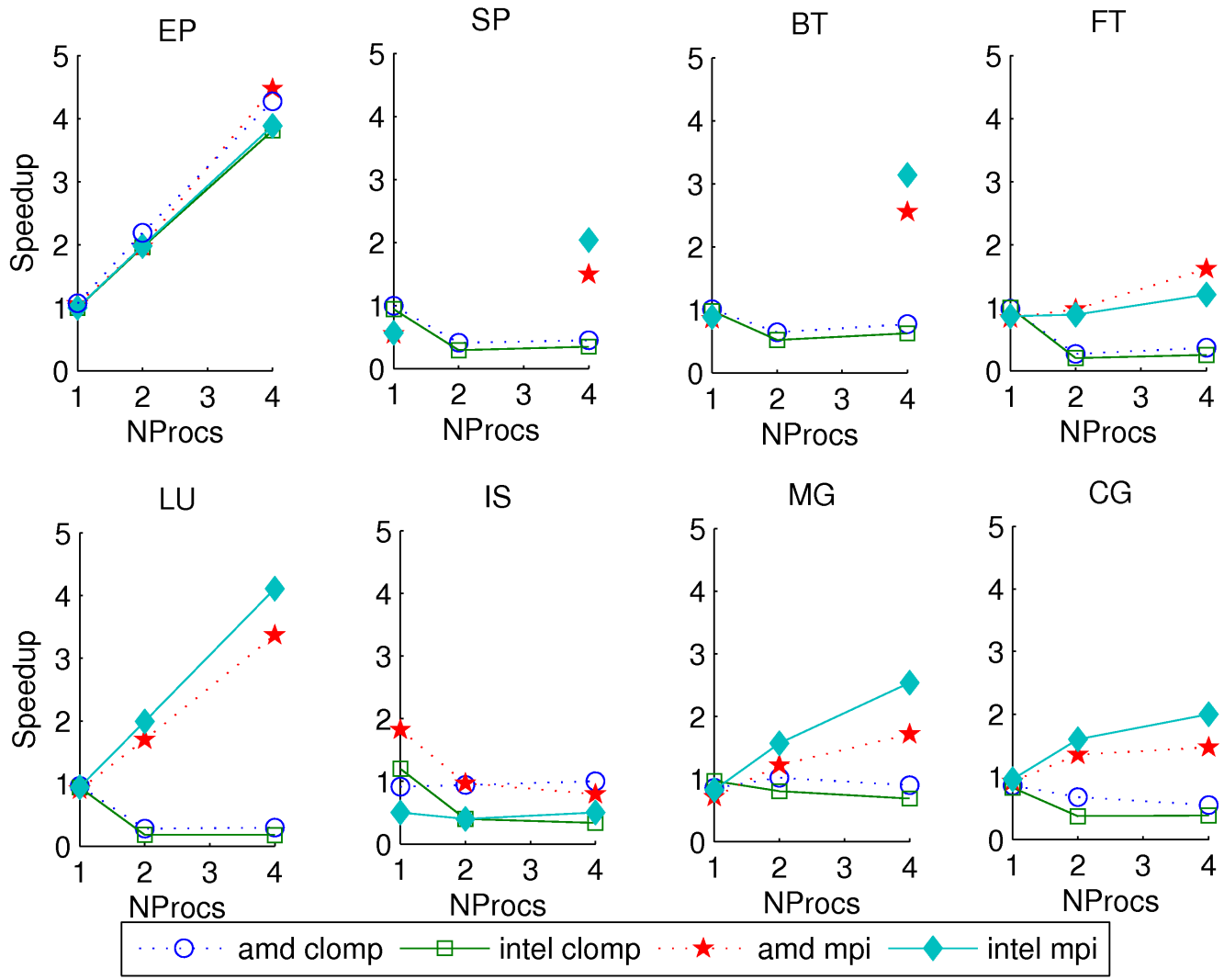
varying array size,
4 KB chunks

- 4 B chunk size shows effects of false sharing
- $O(1/p)$ scaling when chunk size set to unit of coherency
- rapacity cache misses obscure consistency misses

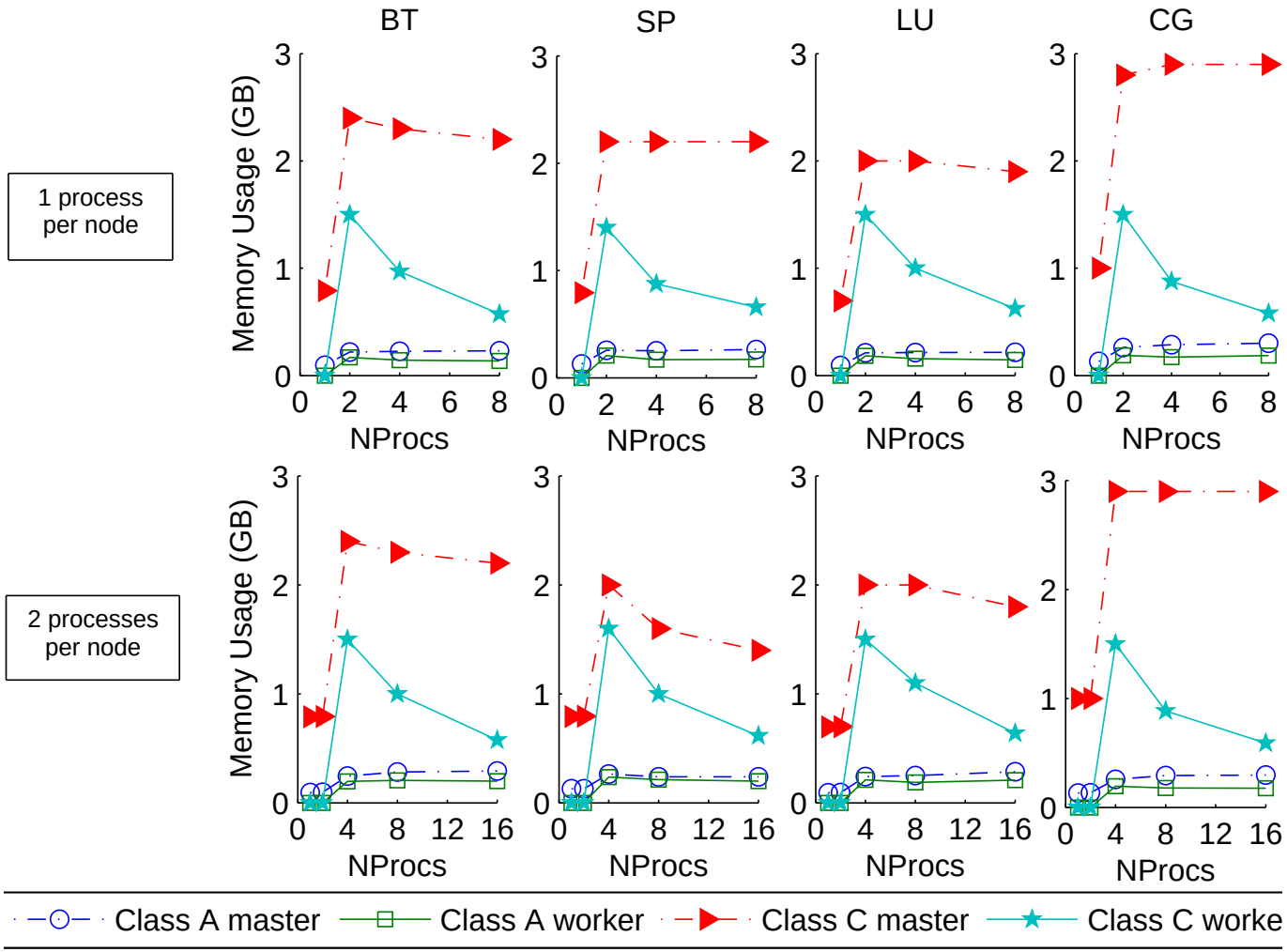
23 Performance of Cluster OpenMP: NAS Class A & C, GigE



24 Performance of Cluster OpenMP: NAS Class A, IB



25 Memory Usage of Cluster OpenMP: NAS Class A



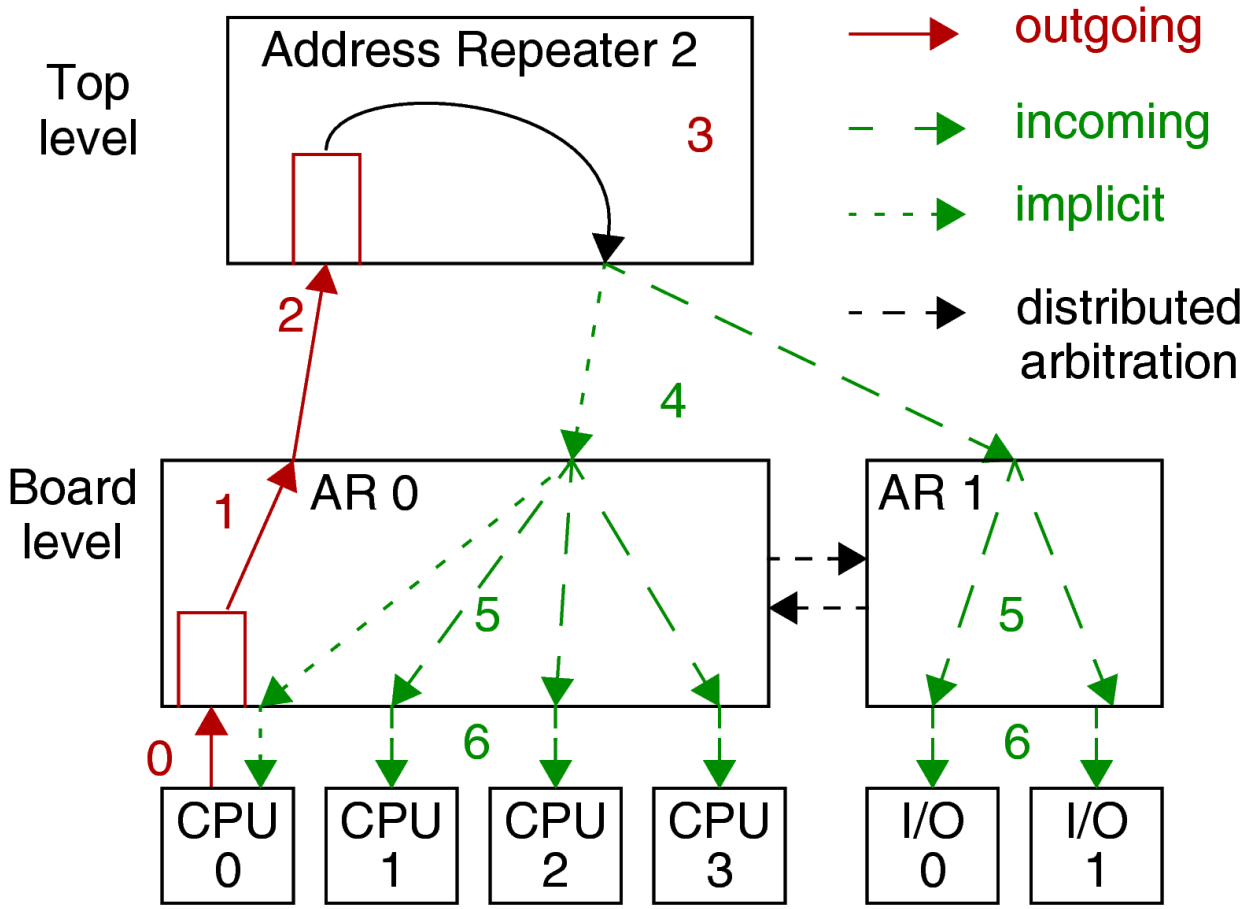
26 Approaches to Performance Evaluation

- Sun Microsystems donated a 12 CPU (900 MHz) UltraSPARC V1280 (SMP) to the ANU for the CC-NUMA Project (2003–)
  - 32KB I-Cache, 64KB D-Cache, 8MB E-cache
  - relies on hardware/software prefetch for performance
  - Sun FirePlane interconnect (150 MHz)
    - 'fat tree'-like address network, some NUMA effects
- benchmarks of interest: SCF Gaussian-like kernels in C++/OMP (by Joseph Antony)
  - primarily user-level, with memory effects of most interest
  - parallelize with special emphasis on data placement & thread affinity
  - use `libcpc` (CPC library) to obtain useful statistics
  - use simulation for more detailed information (e.g. E-cache miss hot-spots & their causes), or for analysis on larger/variant architectures
- OMP version of NAS Parallel Benchmarks also of interest

27 Sparc-Sulima: an accurate UltraSPARC SMP simulator

- execution-driven simulator with Fetch/Decode/Execute CPU simulator
  - captures both functional simulation *and* timing simulation
  - (almost) complete-machine
  - an efficient cycle-accurate CPU timing module is added
- emulate Solaris system calls at the trap level (Solemn, by Bill Clarke), including LWP traps for thread support
  permits simulation of unmodified (dynamically linked) binaries
- the CPU is connected to the memory system (caches and backplane) via a 'bridge'
  - can have a plain (fixed-latency) or fully pipelined Fireplane-style backplane
- simulator speed: slowdowns in range 500–1000 $\times$
- source code available from Sparc-Sulima home page
- (recent) team: Bill Clark, Andrew Over and Peter Strazdins

28 Target Architecture – UltraSPARC FirePlane Protocol



FirePlane address bus timing (from Alan Charlesworth, The Sun Fireplane System Interconnect, *ACM/IEEE Conference on Supercomputing*, Nov 2001.)

29 Target Architecture – FirePlane Protocol (cont)

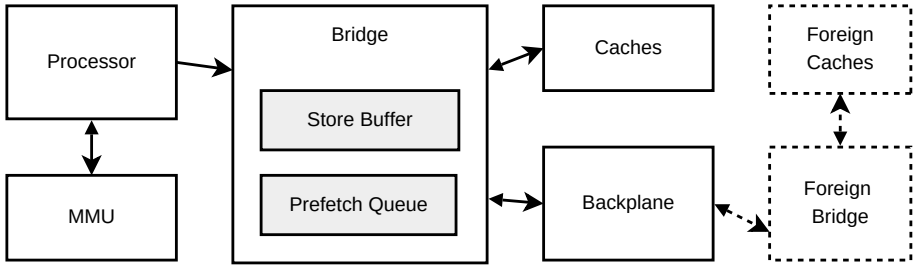
1. CPU 0 begins transaction (cycle 0)
2. system address repeater broadcasts address (cycle 3)
3. all CPUs snoop transaction's address (cycle 7)
   CPUs respond (cycle 11)
   CPUs see result (cycle 15)
4. owner initiates memory request (cycle 17)
5. data transfer begins (cycle 23)
   - completion varies depending on distance
   - 5 cycles for same CPU
   - 9 cycles for same board
   - 14 cycles for different board

note: here 'CPU' means 'the E-cache associated with the respective CPU'

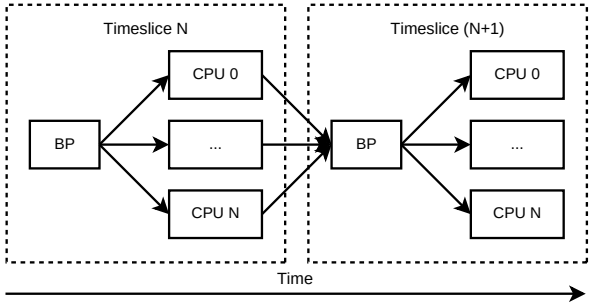
30 Sparc-Sulima: Pipelined Memory System Design

(Andrew Over's PhD topic)

- minimum latency between event & impact on foreign CPU (in the FirePlane) is 7 bus cycles – can apply parallel discrete event simulation techniques



bridge-based structure



run-loop (timeslice = 7*6 CPU cycles)

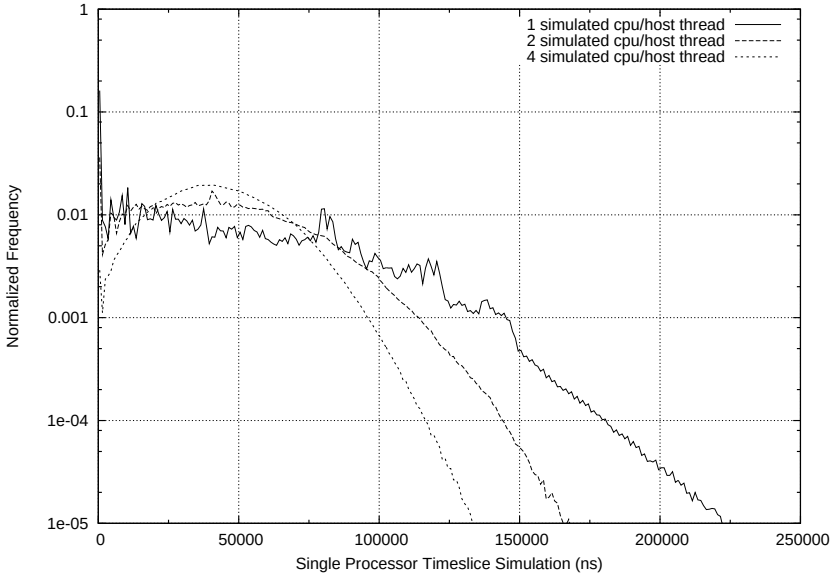
- asynchronous transactions facilitated by retry of load/store instructions, CPU event queues, and memory request data structures
- simulating the prefetch-cache and store buffer was particularly problem-atic
- added simulation overhead is typically 1.20 – 1.50
- scope for parallelization when running on an SMP host

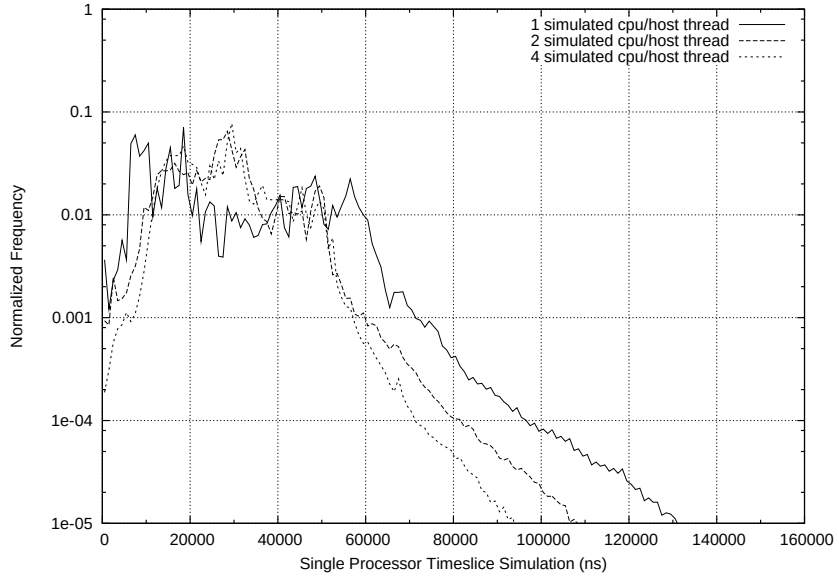
31 Parallelization of the Simulator

- on an SMP, CPUs interact with each other only through the memory system
- hence, natural to assign a separate thread to each simulated CPU
- two approaches:
  - using plain backplane model:
    - each simulated CPU runs for $s \geq 1$ simulated cycles – then synchronize (how big should s be?)
    - memory system access protected by an array of locks, based on the 'index' of the address
  - using pipelined backplane model: $s = 42$ (fixed)
    - backplane simulation phases must be serialized
    - appears to be a problem (Amdahl's Law) – but this is relatively fast
  - but there is a more serious and fundamental problem for both . . .

32 Simulator Parallelization - Load Imbalance Issues

- load imbalance in the time required to simulate s cycles
  - exacerbated by optimizations in Sparc-Sulima: 'skip' stall cycles, 'caching' of calculations (e.g. instruction decoding and address translation)
- histograms of time to simulate $s = 42$ cycles on NAS OMP `ft.S` & `is.W`:

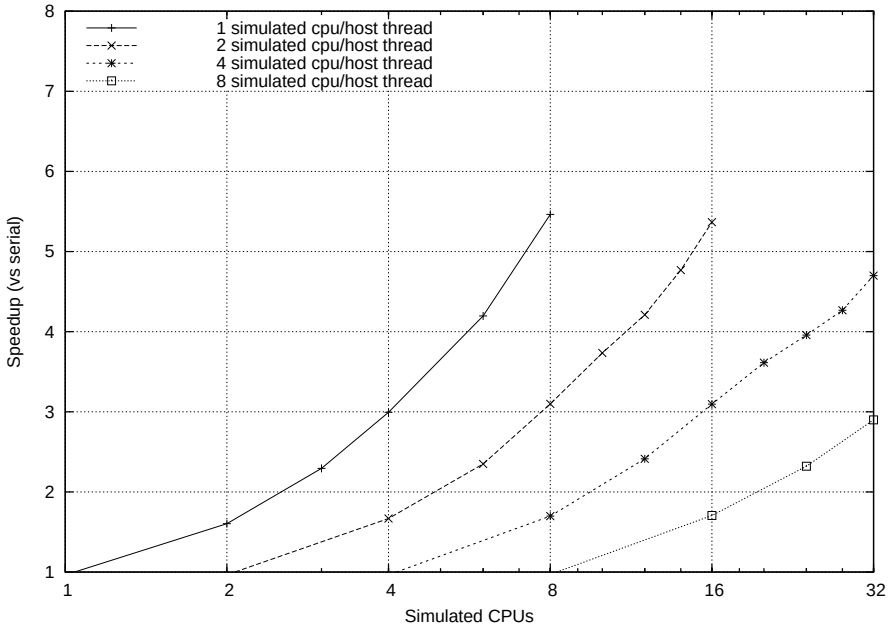


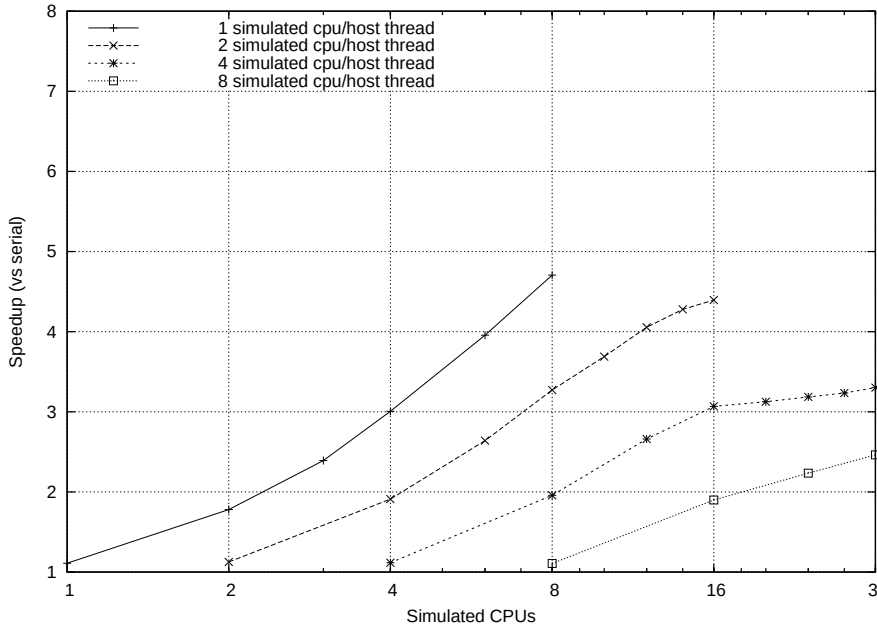


- for the plain backplane model, an accuracy analysis on the NPB indicated that $32 \leq s \leq 256$ was optimal speed–accuracy tradeoff

33 Parallelization - Results

- speedup for NAS OMP `ft.S` of plain ($s = 256$) and pipelined model:

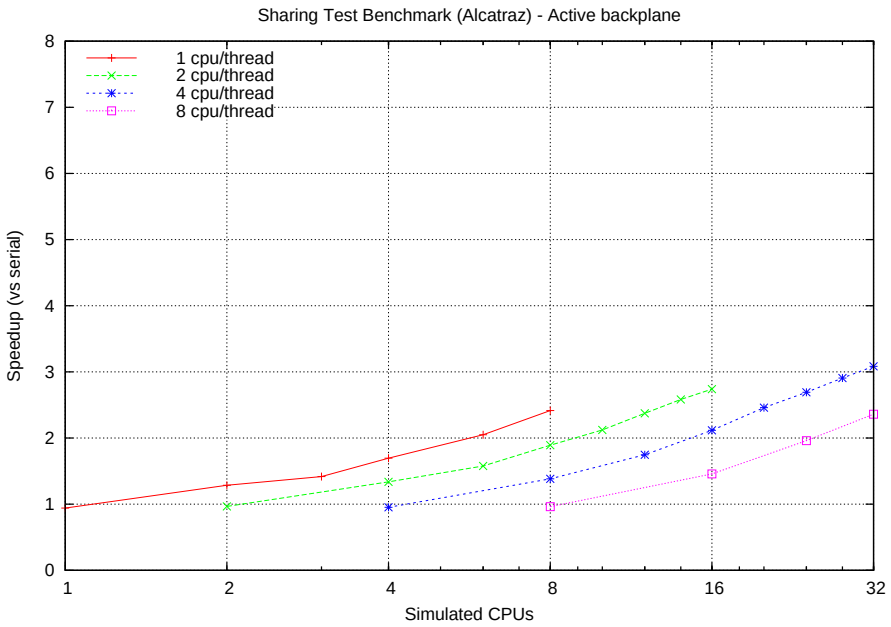




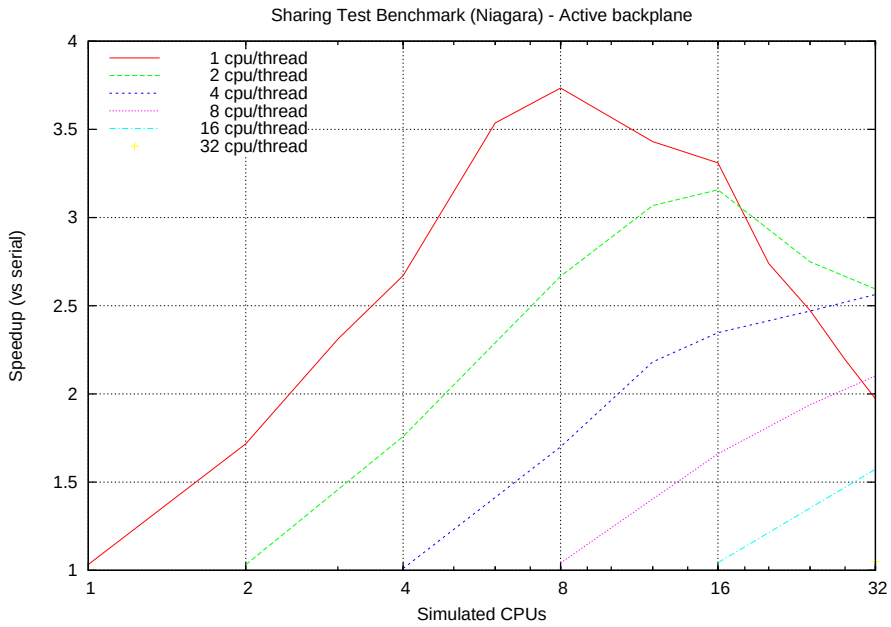
- speedups dependent on application (instruction mix and frequency of stall events)

34 Multicore as a Parallel Simulator Host

- synthetic benchmark with extensive memory traffic (hard to ||ize!)
  - small shared region used for writes
  - large private region used for reads
  - different reference stream for each thread



(a) V1280

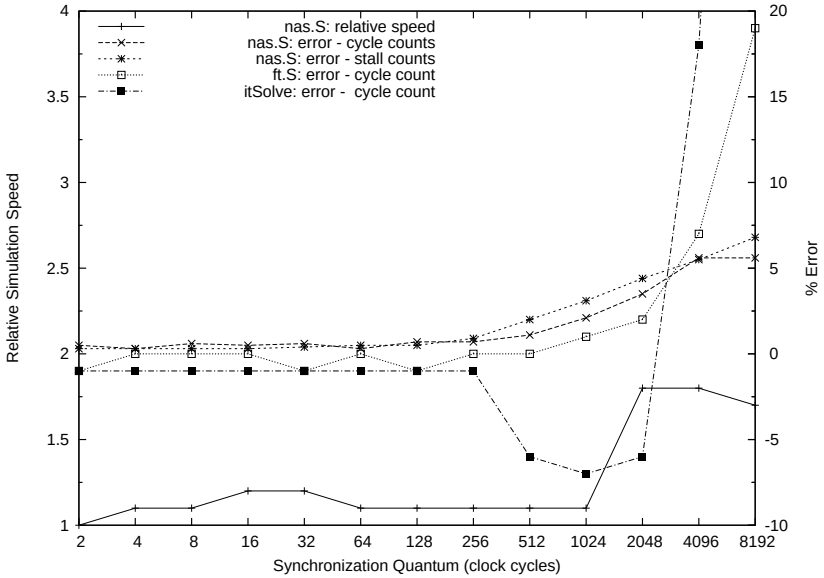


(b) Niagara-1

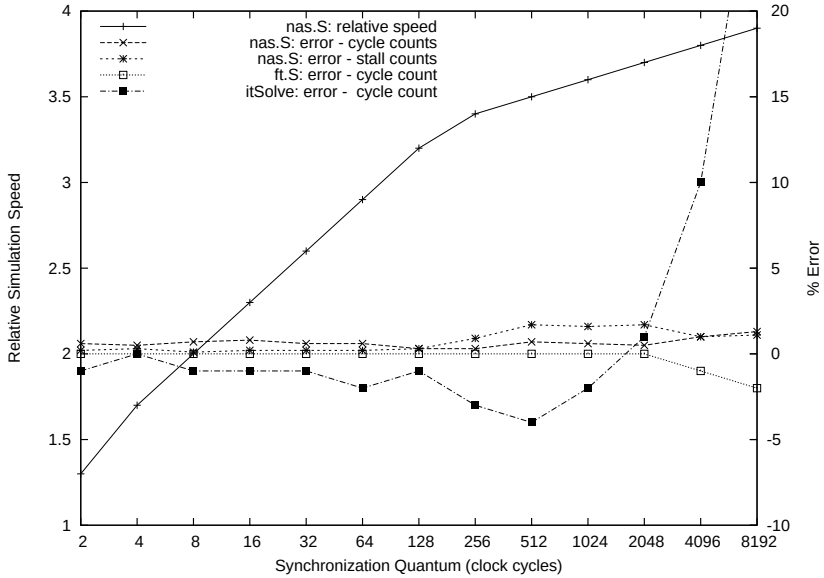
- on NAS etc, both showed better speedups

35 Multiprocessor Simulation - Stability Issues

- relationship between synchronization quantum, speedup and accuracy for simulation of a 4 CPU UltraSPARC III



serial simulation



parallel simulation (1 CPU per thread)

(error and simulation speed are relative to a simulation quantum of 1)

- for a given simulation quantum, parallel simulation is more accurate!
- frequently synchronizing applications are generally the most sensitive

36 Simulator Validation Methodology

- verifying simulator accuracy is critical for useful performance analysis
  - essential in any kind of performance modelling!
- validation is an ongoing issue in field of simulation
- microbenchmarks: used to verify timing of a multitude of single events
- application-level: by the OpenMP version of the NAS Parallel Benchmarks
  - use of hardware event counters (via UltraSPARC CPC library)
  - √ permits a deeper-level of validation than mere execution time
  - √ also provides breakdown of stall cycles (e.g. D/E-cache miss, store buffer)
  - × hardware counters are not 100% accurate;
    also ambiguously/incompletely specified (e.g. stall cycle attribution)

37 Validation: Microbenchmarks

- e.g. cache-to-cache transfer microbenchmark:

Processor A		Processor B	
1: st	%g0, [A]	1: call	_cache_flush
call	_barrier	call	_barrier
		rd	%tick, %l0
		ld	[A], %g0
		rd	%tick, %l1
		sub	%l1, %l0, %l0
call	_barrier	call	_barrier
ba	1b	ba	1b

- also D/E Cache load/store hit/miss (various cache states/CPU pairs), atomic instr'n latency, store bandwidth, memory access (various regions), RAW, etc
- preferable to (possibly erroneous, out-of-date) data sheets
- provided valuable data, with several surprising & undocumented effects

38 Multiprocessor Validation: NAS Benchmarks (S-class)

- p threads; ratio of total cycles target: simulator (pipelined)

p	BT	FT	IS	LU	LU-hp	MG	SP	UA	Avg	RMS
1	1.02	1.05	0.99	0.97	1.01	0.94	0.99	0.87	0.99	6%
2	0.99	0.97	0.97	0.89	0.76	0.81	0.88	0.93	0.91	12%
4	0.96	0.91	0.92	0.86	0.70	0.78	0.88	0.81	0.85	17%
8	0.93	0.86	0.67	0.87	0.61	0.74	0.73	1.58	0.86	24%

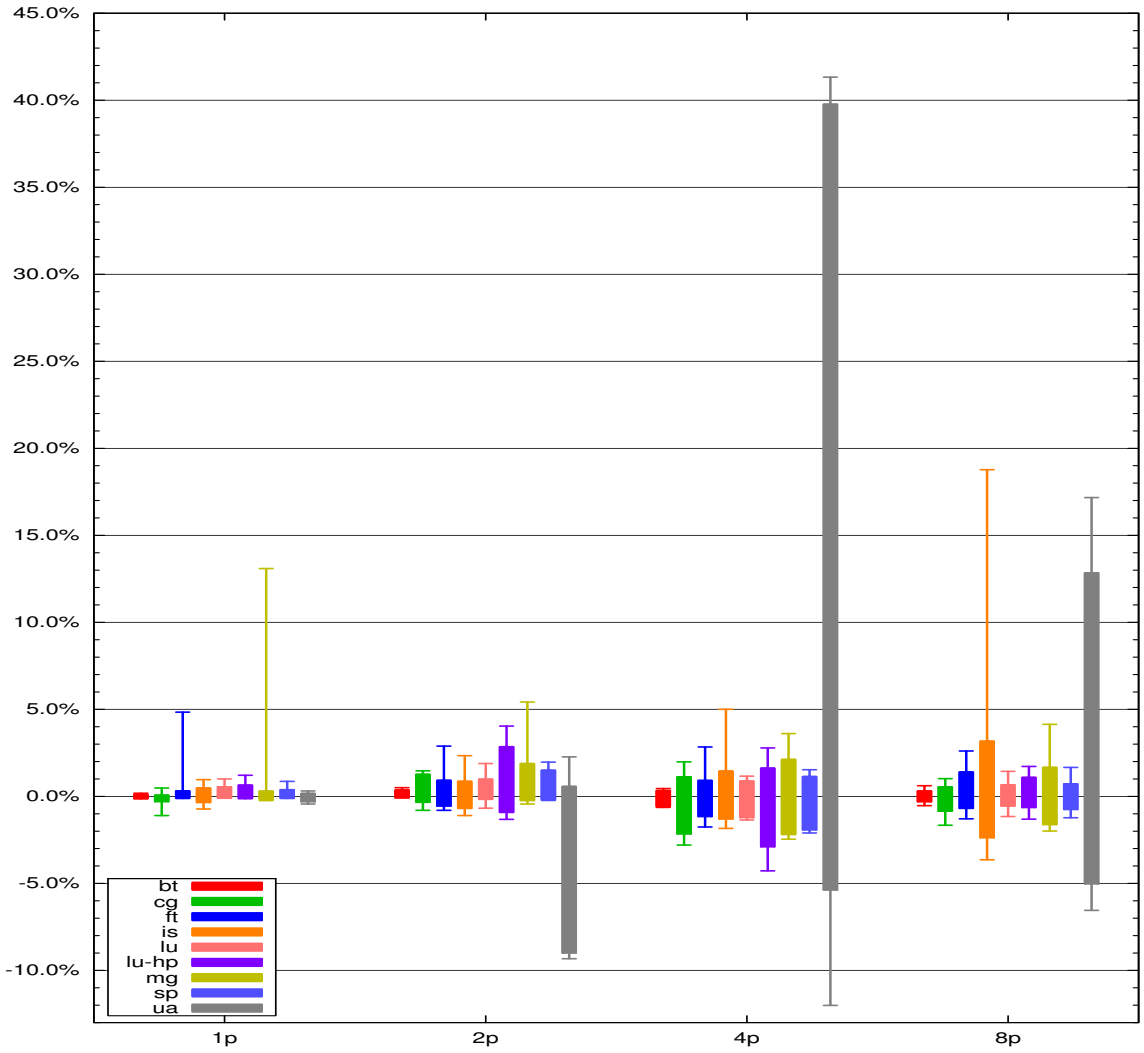
- simulator is generally optimistic; slightly better results for W-class
  - non-pipelined sim. slightly more optimistic (RMS: 6%, 1%, 22%, 30%)
  - as p increases, instability on host does (esp. UA)
  - errors in counts of total instructions generally matched cycle counts
  - E-cache events (stalls, copy-back and invalidate event counts) generally agreed closely
- issues:
  - modelling the random replacement policy (D-cache problematic)
  - store buffer and prefetch also difficult

39 Multiprocessor Validation (II)

- kernel-level effects (scheduling, cache pollution) were (initially) suspected for inaccuracies
- strong correlation between error and number of thread-related syscalls
  - is kernel-level simulation essential for accurate SMP simulation?
  - *also* between time spent in spin-loops!
- recently tried modified (non-sleeping) `libmtsk`, also `gcc`-compiled (+non-sleeping)
  - no clear correlation between user-level only and unmodified workloads

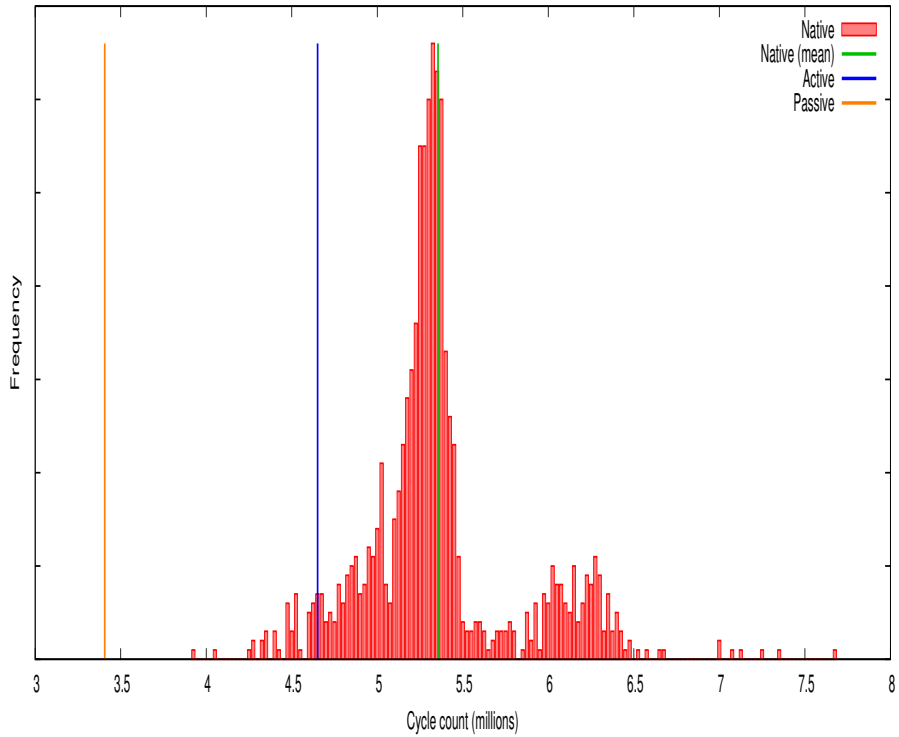
40 Parallel Workload Stability - NAS

- on a quiesced V1280 (under processor sets), there was a variation in cycle counts between repeated runs
- larger variation in other event counts
  - e.g. 10% variation in CG ($p = 1$) for store buffer stalls
- UA is particularly problematic

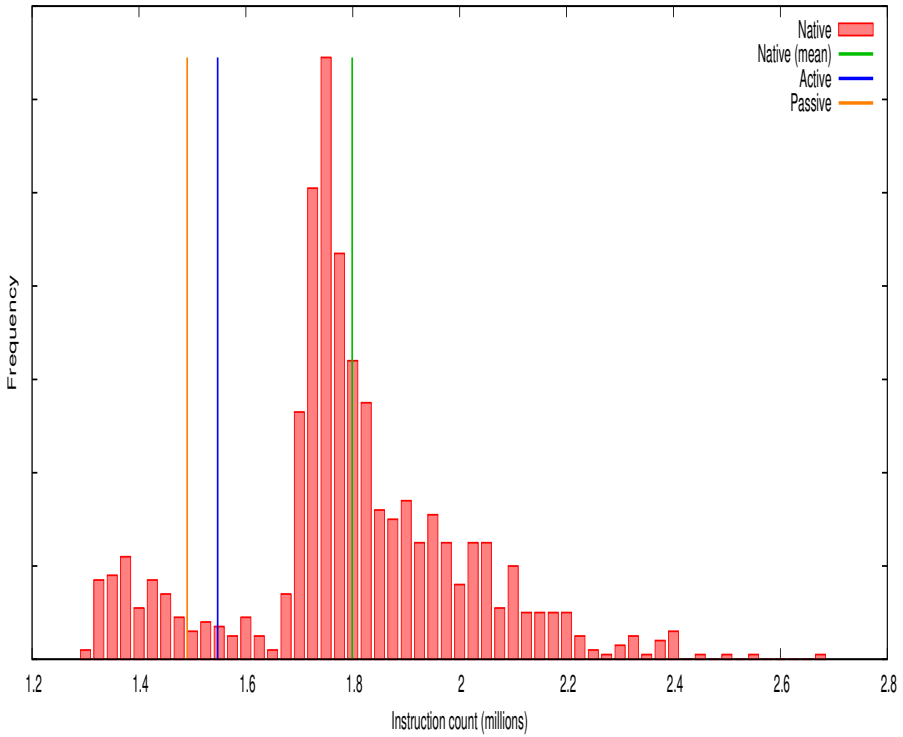


41 Parallel Workload Stability - Barriers

- OpenMP barrier microbenchmark: variability on the V1280



(a) cycle count



(b) instruction count

- which behavior should the simulator be modelling?

42 Computational Chemistry on NUMA Opteron Clusters

- Sun Microsystem's high-end HPC focus is has shifted on clusters of 'fat' Opteron nodes
  - i.e. 2–4 core, 16-way SMP; NUMA effects due to HyperTransport
  - e.g. the 10,480 CPU cluster at the Tokyo Institute of Technology
- accordingly, CC-NUMA project's Phase II (2007–2009) is oriented to this platform (X4600, donated by Sun)
  - algorithm development and performance evaluation of CC (electronic structure methods)
  - development and use of simulation tools for this platform
    - based on the x86-based Valgrind simulator infrastructure – fast!
    - add similar cycle-accurate CPU and memory models
    - also model the cluster communication network
    - and parallelize it all!
  - project of PhD student Danny Robson (started Sep 07)

43 Modelling of Multi-core NUMA Clusters: Approach

- implement as a Valgrind 'module' invoked upon memory references
- develop configurable cache & memory system models (MicroLib?)
  - need also to model page coloring and NUMA effects
- accurate CPU simulation: can be based on basic blocks - fast!
- for multi-CPUs: Valgrind's scheduling to be controlled by the module
  - must accurately maintain simulated time to co-ordinate simulated threads
  - parallelization of a Valgrind based system a further challenge!
- cluster-level: traps corresp. to comms. invoke a cluster-level scheduler
  - such as distributed implementation provides an interesting challenge!
  - construct performance models of the communication network of various levels of accuracy (e.g. contention)
- alternative infrastructure, SimNow!, has so far shown to be limited in accuracy and extensibility
- validation: may have to rely on microbenchmarks more . . .

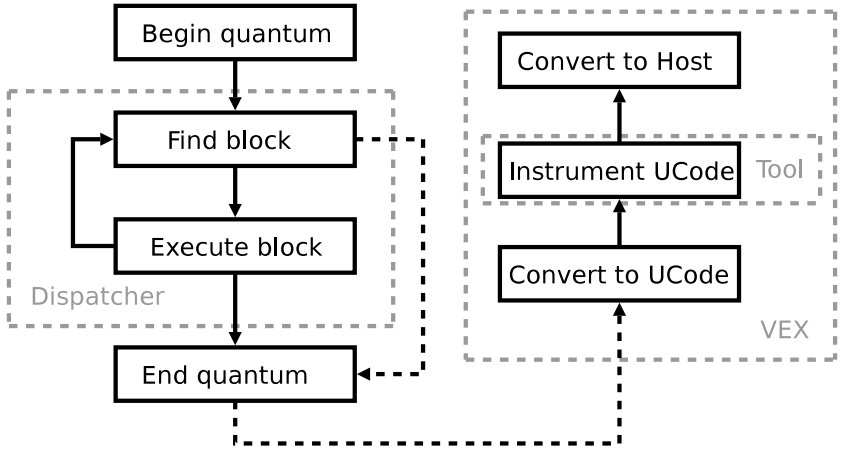
44 Overview of Valgrind

- Valgrind dynamic binary translation and instrumentation framework – user-level only!
  - can run threaded programs (serialized), but is not intended to be parallelized!
- operates on basic blocks of an unmodified executable, which are translated first to UCode for later instrumentation

```
0x4001D5E:  subq 2199099(%rip),%rax

------ IMark(0x4001D5E, 7) ------
PUT(168) = 0x4001D5E:I64
t22 = Add64(0x4001D65:I64,0x218E3B:I64)
t21 = GET:I64(0)
t20 = LDle:I64(t22)
t19 = Sub64(t21,t20)
PUT(128) = 0x8:I64
PUT(136) = t21
PUT(144) = t20
PUT(0) = t19
```

(a) UCode translation



translate, instrument & dispatch loop

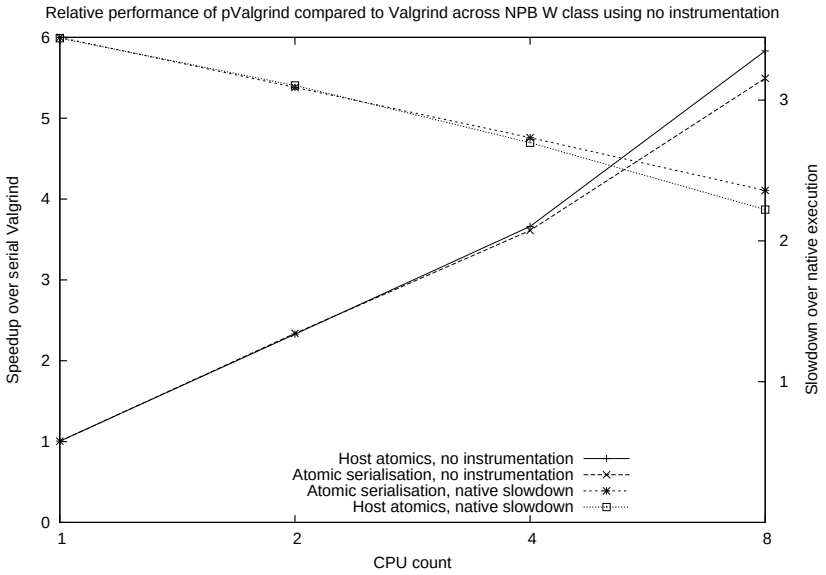
45 Parallelization of Valgrind

- aim: ||ize 'core' + some demo tools for AMD-64 (paper for ISPA'08)
- issue: flushing of blocks in the translation cache:
  - what if another thread is accessing it?
  - put it in a pending deletion list
  - then (atomically) clear pointer in the fast translation table
- issue: (translations of) atomic instructions must be executed atomically!
  - e.g. if atomic decrement was translated into read, subtract, write:

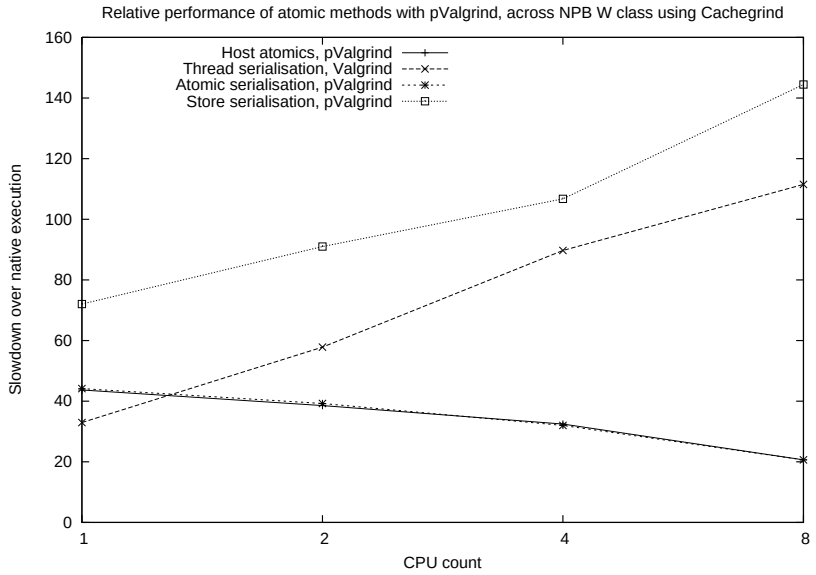
time	thread 1	thread 2	thread 3
1	`read [m1], r1`		
2		`read [m1], r2`	
3	`sub r1, 1, r1`		`write r3, [m1]`
4		`sub r2, 1, r12`	
5	`write r1, [m1]`		
6		`write r2, [m1]`	
	(atomic lost)		(store lost)

46 Parallel Valgrind: Performance

- locks for atomics & stores to same location is safest but much too slow!
- locking only atomics safe enough in practice
- alt. use atomics on (AMD-64) hosts (less generic)
- geometric means on NPB.W (slowdown over native execution)



(a) pValgrind only (with speedups)



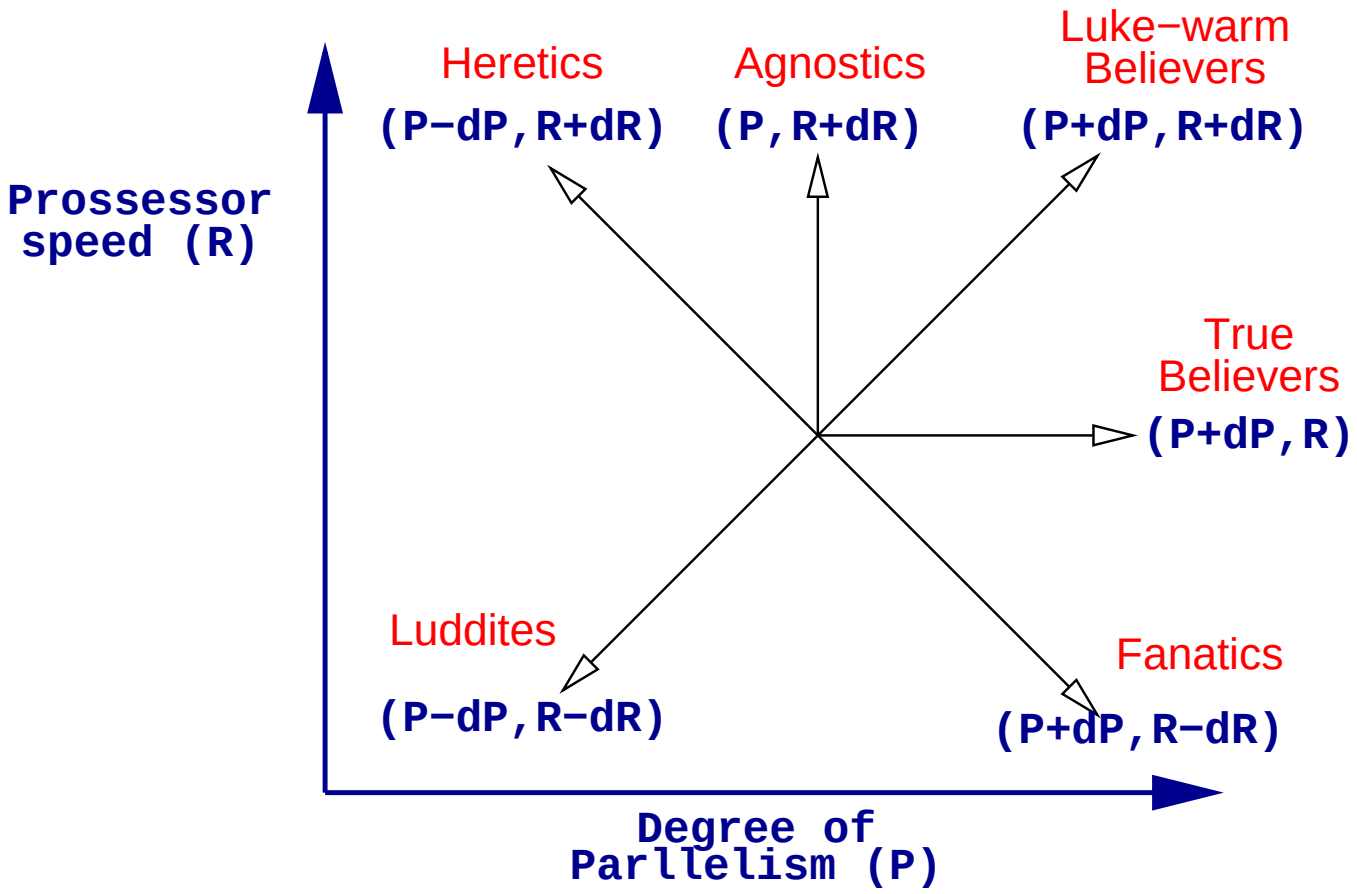
(b) adding pCachegrind tool

- code available from http://ccnuma.anu.edu.au/pvalgrind/
- (parallel) NUMAgrind is currently being validated

47 Simulation Work – Conclusions

- accurate and reasonably efficient simulation of a modern NUMA memory system can be achieved
  - entails hard work, and limited by lack of accurate and complete documentation of the target system
  - hardware event counter based validation methodology was reasonably successful, but some issues remain
  - reasonably efficient parallelization has been achieved
  - have also analysed some computational chemistry applications with it!
  - validation, and (simulator & host) stability issues need further investigation
- are extending the performance evaluation methodology and simulation tools to clusters
  - useful to improve cluster performance instrumentation (develop network-level hardware event counter infrastructure)

48 Demography of Parallel Computing



(origin unknown)

- mid 90's: advent of the killer micros
- mod 2000's: advent of multicore

49 Multicore Computing (Future Work)

with recent donation of a mavericks, a T5120 UltraSPARC T2 processor:

- integrate into our teaching program via a logical domain wallaman
- possible research topics:
  - hardware threading / LDom evaluation; T2 BLAS
  - evaluation of emerging multicore programming paradigms (+Haskell!)
  - a T2 version of the Sparc-Sulima simulator
  - investigate OS issues: scheduling and scalability: TM support
  - large-scale multicore simulation: validation, performance and stability issues
  - software-hardware co-design for heterogeneous multicore for (virtualized) HPC