The SB-PRAM Project

work of W. J. Paul, P. Bach, M. Bosch, J. Fischer, C. Lichtenau, J. Röhrig, · · ·

Roman Dementiev / MPI für Informatik

dementiev@mpi-sb.mpg.de

The work reported here was done while all the authors were affiliated with Saarland University

Overview

- SB-PRAM architecture
- System
  - hardware
  - system software
  - applications
- Performance measurements

Shared memory emulation

- memory contention (caching, address hashing, combining)
- network contention (random routing, combining)
- memory latency (caches, multithreading)

Some shared memory architectures

	multi threading	address hashing	combining	caches
HEP	√	-	-	-
TERA	√	√	-	-
T3E	√	√	at memory	√
Ultra/RP3	-	√	√	√
SB-PRAM	√	√	√	-

SB-PRAM: some details

- 32 threads per processor
  - $th = 32 \cdot p$ threads with p processors
  - $th = 2048$ if $p = 64$
- butterfly network, multiprefix operations
- linear hash function
- Berkeley RISC instruction set with FPU

SB-PRAM: properties predicted

- p processors, cycle time t, v threads per processor
- user sees $v \cdot p$ processor PRAM with cycle time $v \cdot t$
  - $v = 3 \log p$
  - Ranade's analysis + simulations
- hardware effort $O(p \log p)$ [gates]
  - $p = 64$: network has *26 %* of gates of the machine

Processor board

- processor, 80 000 gates, 0.7 μm, 6.75 MHz
- sorting array, 37 000 gates
- 802 cm^2

Network board

- network chip:
  67 000 gates,
  0.8 μm, 27 MHz
- 1 chip $=$ 2 network nodes
- 1464 cm^2

Memory board

- 4 banks per card
- 1 bank: 16 MB
- 743 cm^2

Area $a(p)$

- $a(p) = 1173.5 \cdot p + 91.5 \cdot (p \log p)$
- $p = 64$: network = 32 %
- $p > 4096$: network = 50 %

Software

- a UNIX-like OS
- compiler for the language FORK
- extension of C and C++ with shared variables
- communication primitives for C: P4 library
- benchmark programs

Defining speedup and efficiency

- t_{ser}: wall clock time of fastest serial program on 1 thread
- t_{par}: p processors, $32 \cdot p$ threads
- 1 thread $= 1/32$ processor !
- $speedup = t_{ser}/(32 \cdot t_{par})$
- $eff = t_{ser}/(32 \cdot p \cdot t_{par})$

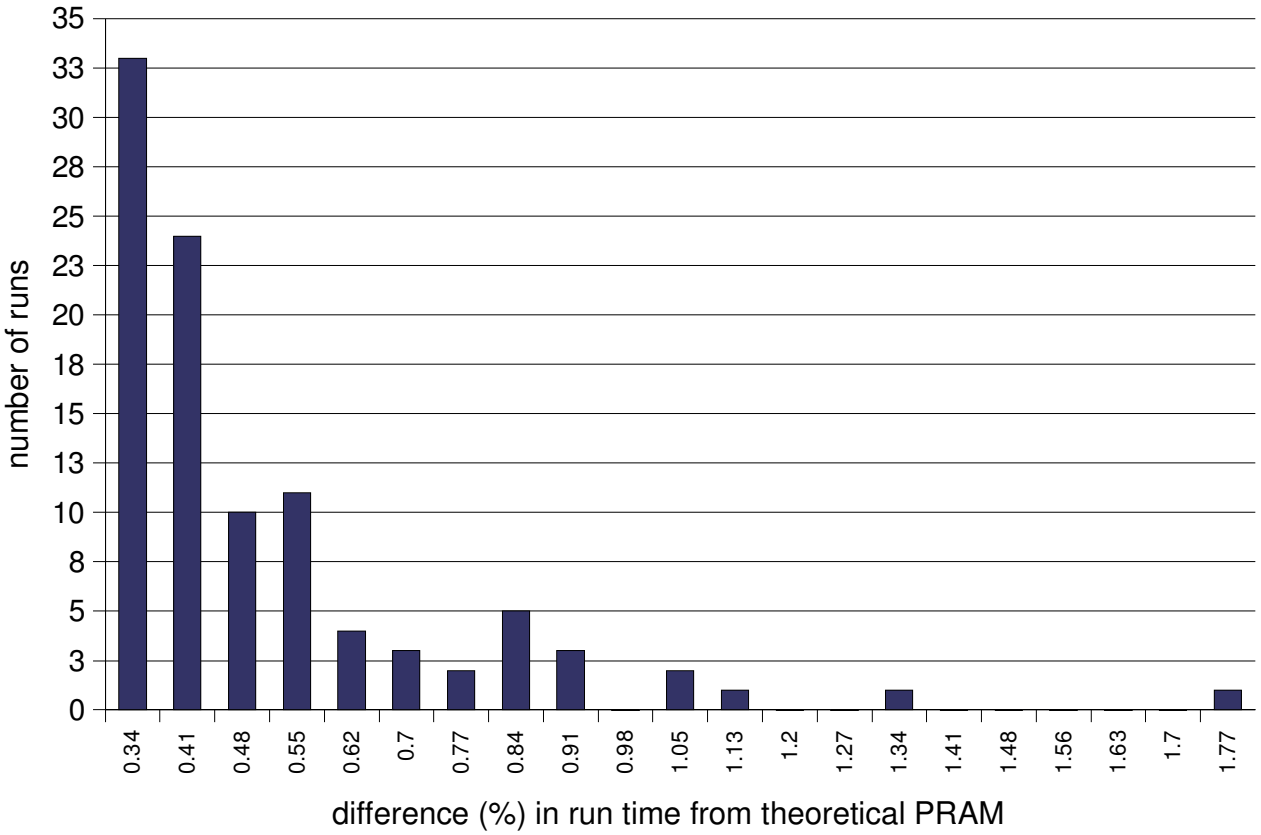
Measurements

- Radiosity, Locus Route: efficiency as good as on distributed machines
- PTHOR: speedup *2.5* with 16 processors (512 threads)
- MP3D:
  - $p = 32$: $speedup =$ *26*
  - $p = 64$: $speedup = 35$
  - $p = 64$, $80000\ particles$: $speedup =$ *45*

Measurements vs simulated run times

- Simulator simulates processor cycles, assumes no congestion of net and memory (e.g. theoretical PRAM)
- We run *all* benchmarks with 100 different random hash factors

Locus Route with random hash factors



Measurements vs simulated run times

- All runs with 64 processors, maximal observed relative difference:
  - 223 % with hash factor $h = 1$
  - *2.9 %* with 100 different random hash factors
  - *0.45 %* for synthetic programs, which consist *only* from random writes and/or reads

Conlusions

- we could make it work
- behaves as hoped for
  - 2.9% worse
- scales well
  - network area $= 50\ \%$ for $p > 4096$
- 1GHz processor would need optical interconnect between chips