



Computational Science and Engineering Software in the Multicore Era

Sverker Holmgren

(Many slides by Erik Hagersten)

Uppsala University
Sweden



Outline

- Conclusions
- Why? (Including a Crash-Course in Computer Architecture)
- Multicore Systems, Now and in the Future
- Impact on Performance
- Impact on CSE Software in General
- Impact on Algorithms
- Example: Multigrid on a Multicore Processor
(joint work with Erik Hagersten, Dan Wallin, Henrik Löf)
- Example: Genetics on a Cluster of Multicore Processors
(joint work with Mahen Jayawardena, Henrik Löf)
- Conclusions



Conclusions

- The Multicore Era is here
- There will be processors with both
 - ✱ "A few fat cores"
 - ✱ "Many thin cores"
- The cost of parallelism is dropping dramatically, but
 - ✱ Cache/thread is dropping
 - ✱ Memory bandwidth/thread is dropping
 - ✱ => The performance is determined by the flow of data
- The shift will have impact on CSE software

"For high-performance computing (HPC) applications, multicore processors introduces an additional layer of complexity, which will force users to go through a phase change in how they address parallelism or risk being left behind."



UPPSALA
UNIVERSITET

Uppsala University

Why?





Why multiple cores/threads?

- Instruction level parallelism is running out
- Wire delay is hurting
- Power dissipation is hurting
- The memory latency/bandwidth bottleneck is becoming even worse



Current Doubling/Halving Times

- Dynamic RAM Memory (bits per dollar) 1.5 years
- Average Transistor Price 1.6 years
- Microprocessor Cost per Transistor Cycle 1.1 years
- Total Bits Shipped 1.1 years
- Processor Performance in MIPS 1.8 years
- Transistors in Intel Microprocessors 2.0 years
- Microprocessor Clock Speed **2.7 years**



UPPSALA
UNIVERSITET

Uppsala University

A Four-Slide Crash- Course in Classical Computer Architecture



Performance so far?

Create and explore:

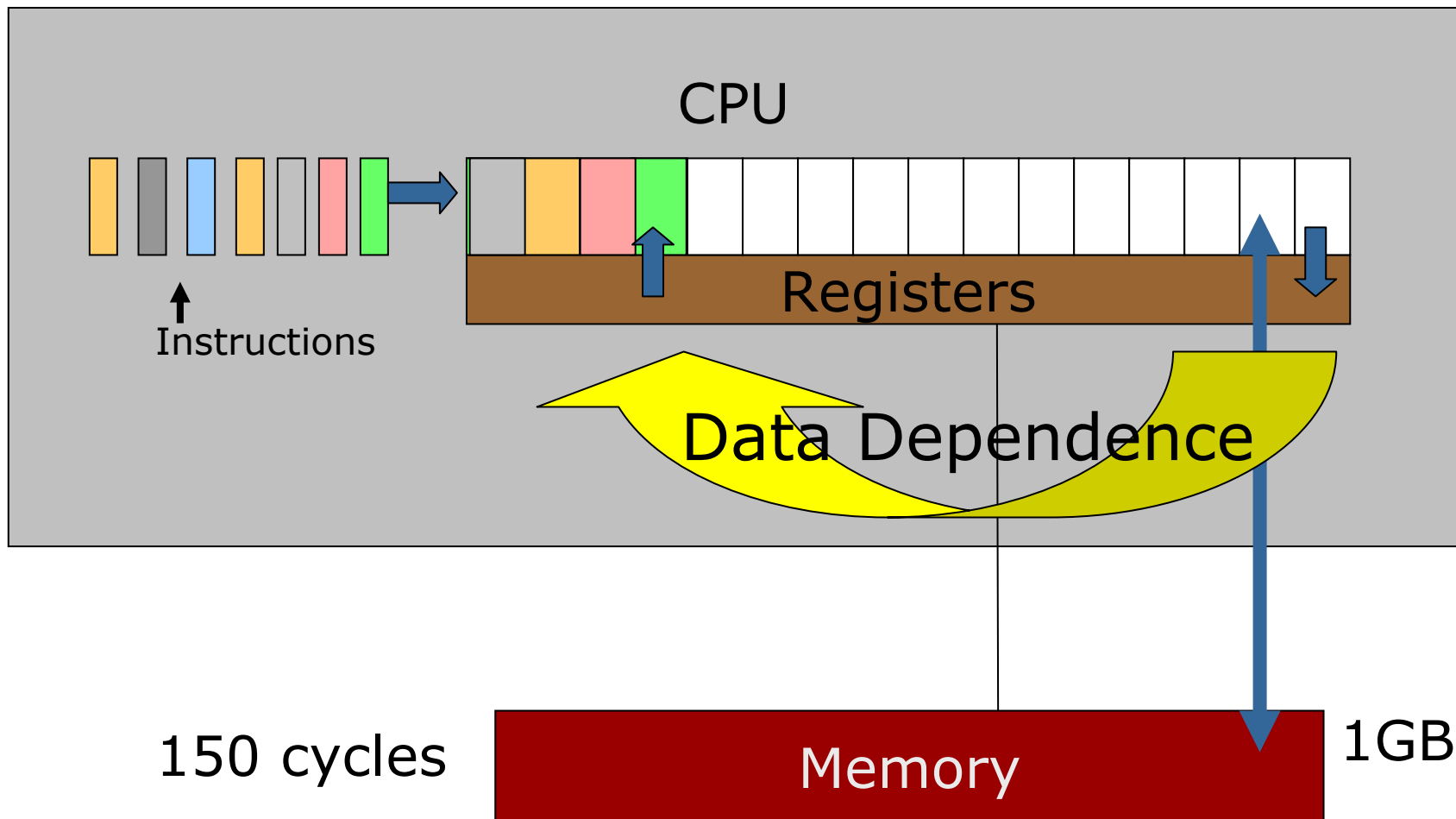
1. Locality of data

- Caches
- Temporal and spatial locality

2. Parallelism

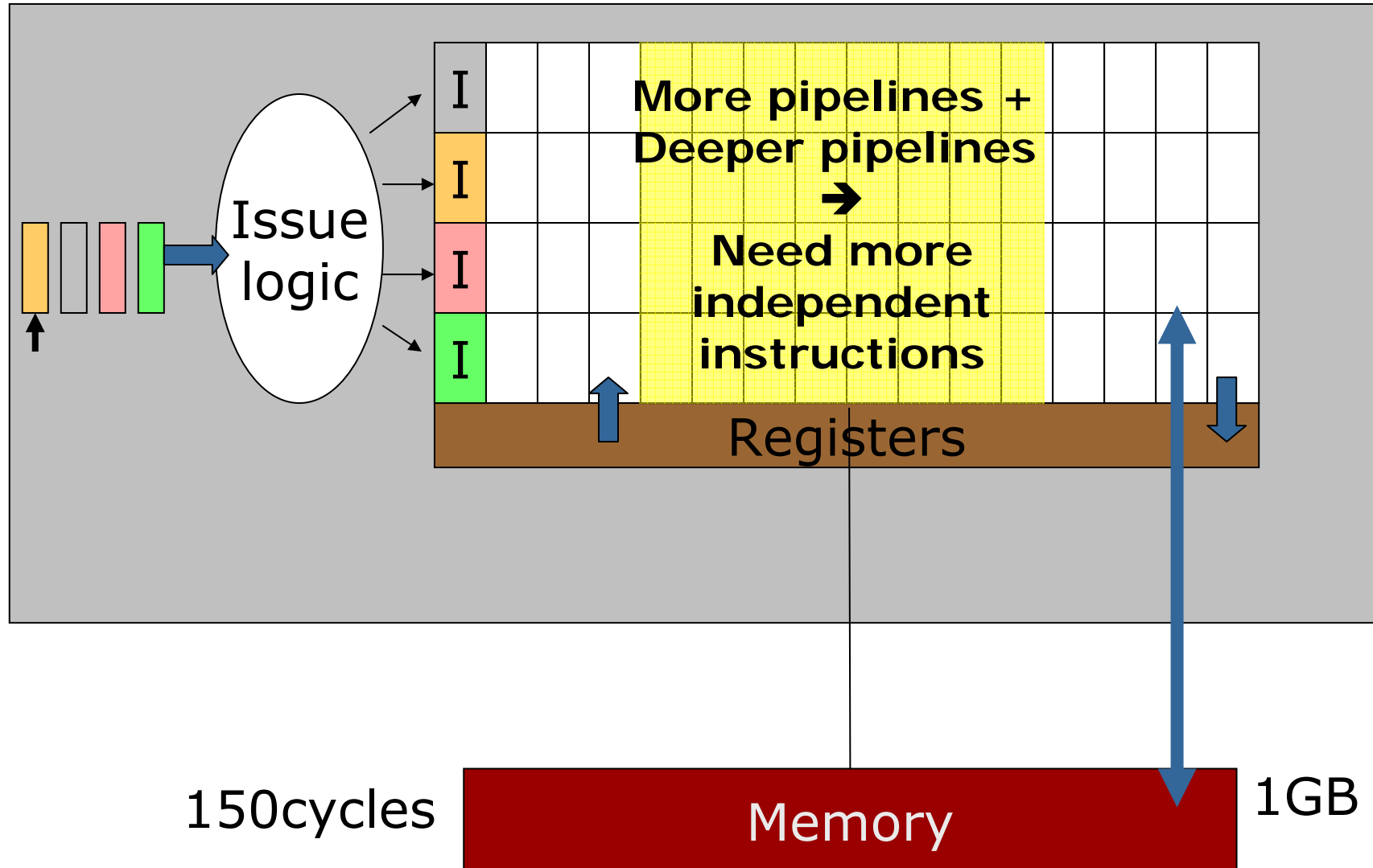
- Instruction level parallelism (ILP)
- Parallelism at the program/alg. level

Old Trend1: Deeper Pipelines (= Exploring ILP)



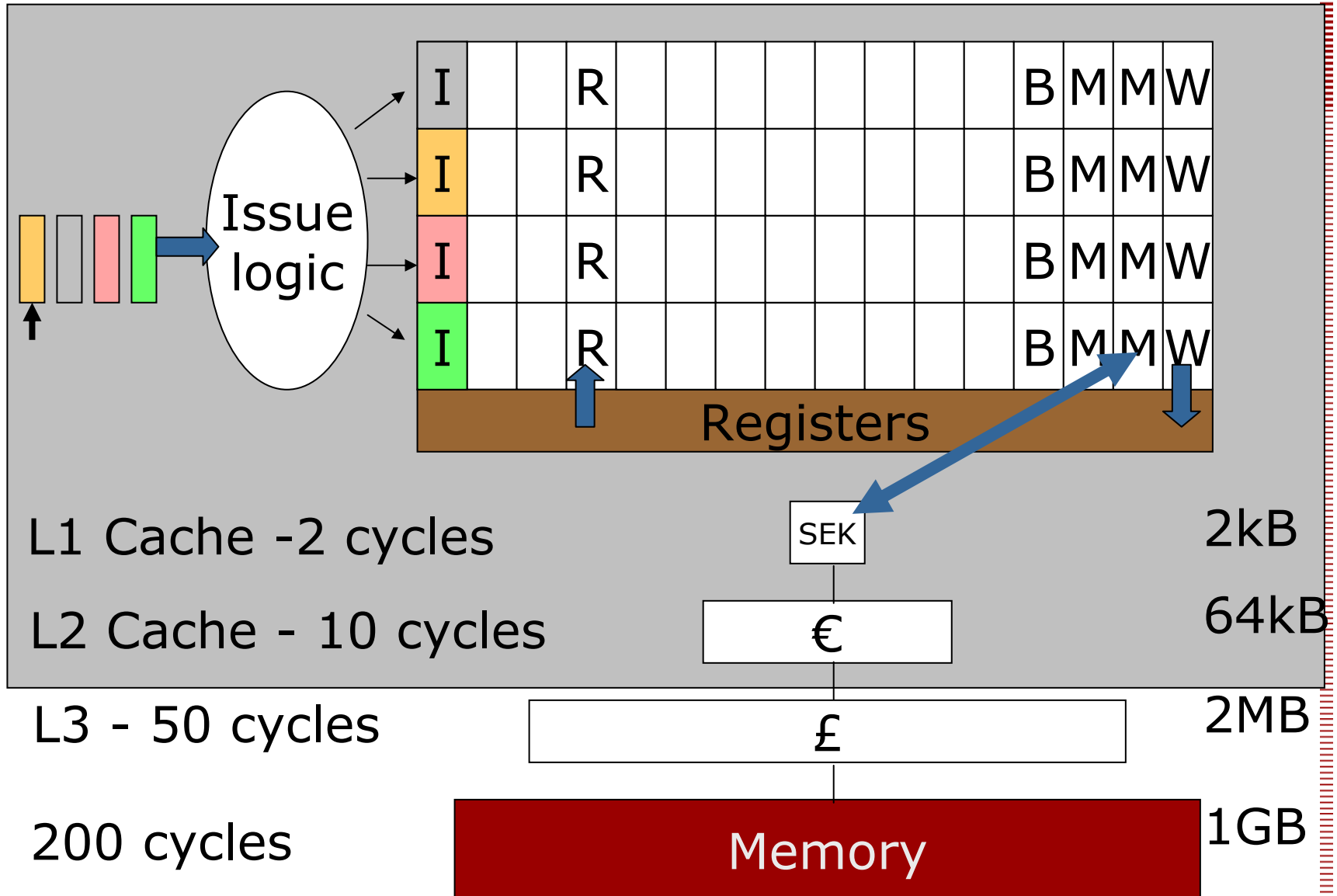


Old Trend 2: Wider Pipelines (= Exploring More ILP)



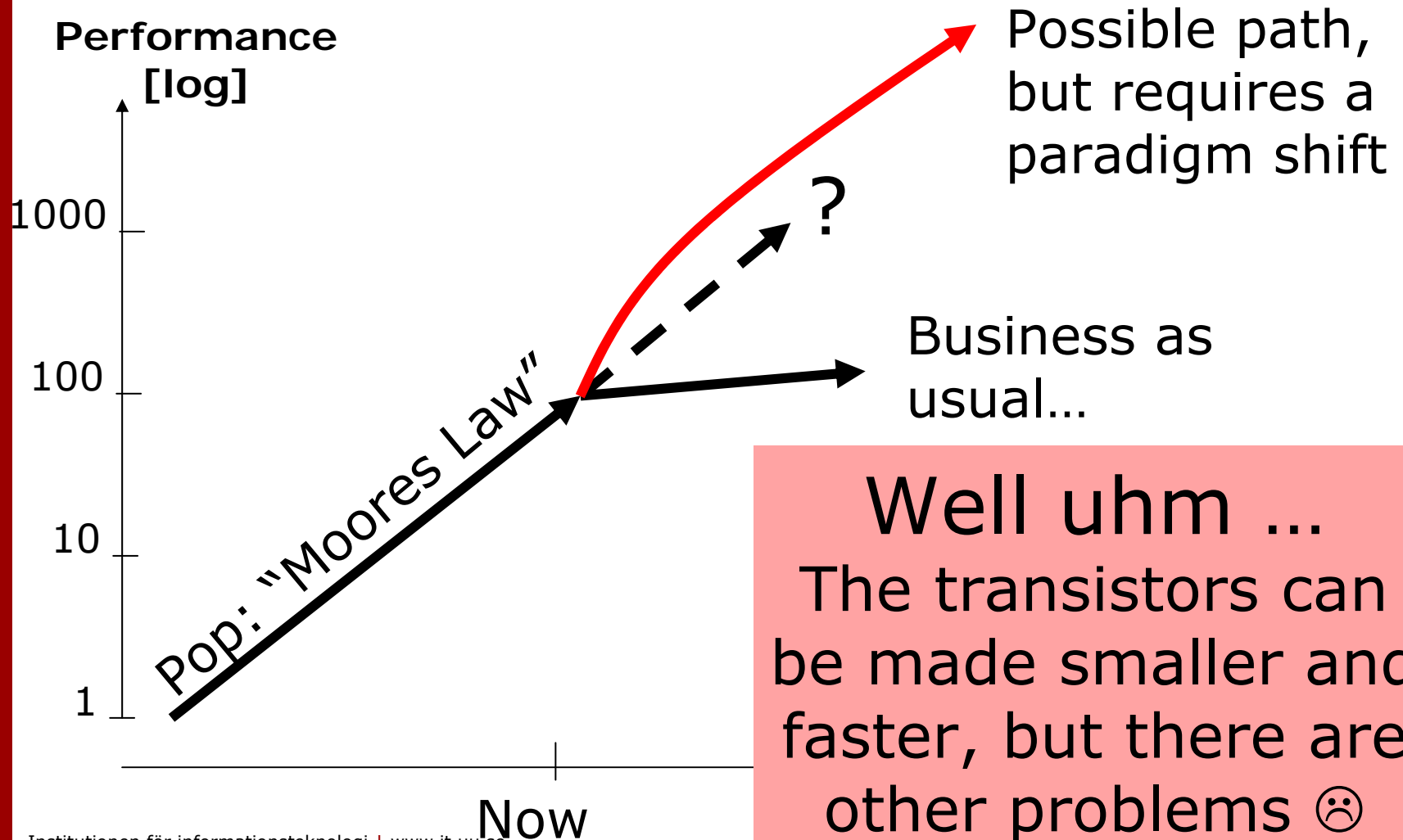


Old Trend3: Deeper Memory Hierarchy (= Exploring Locality of Data)





Are we hitting the wall now?





Classical microprocessors: Whatever it takes to run one program

Exploring ILP (instruction-level parallelism)

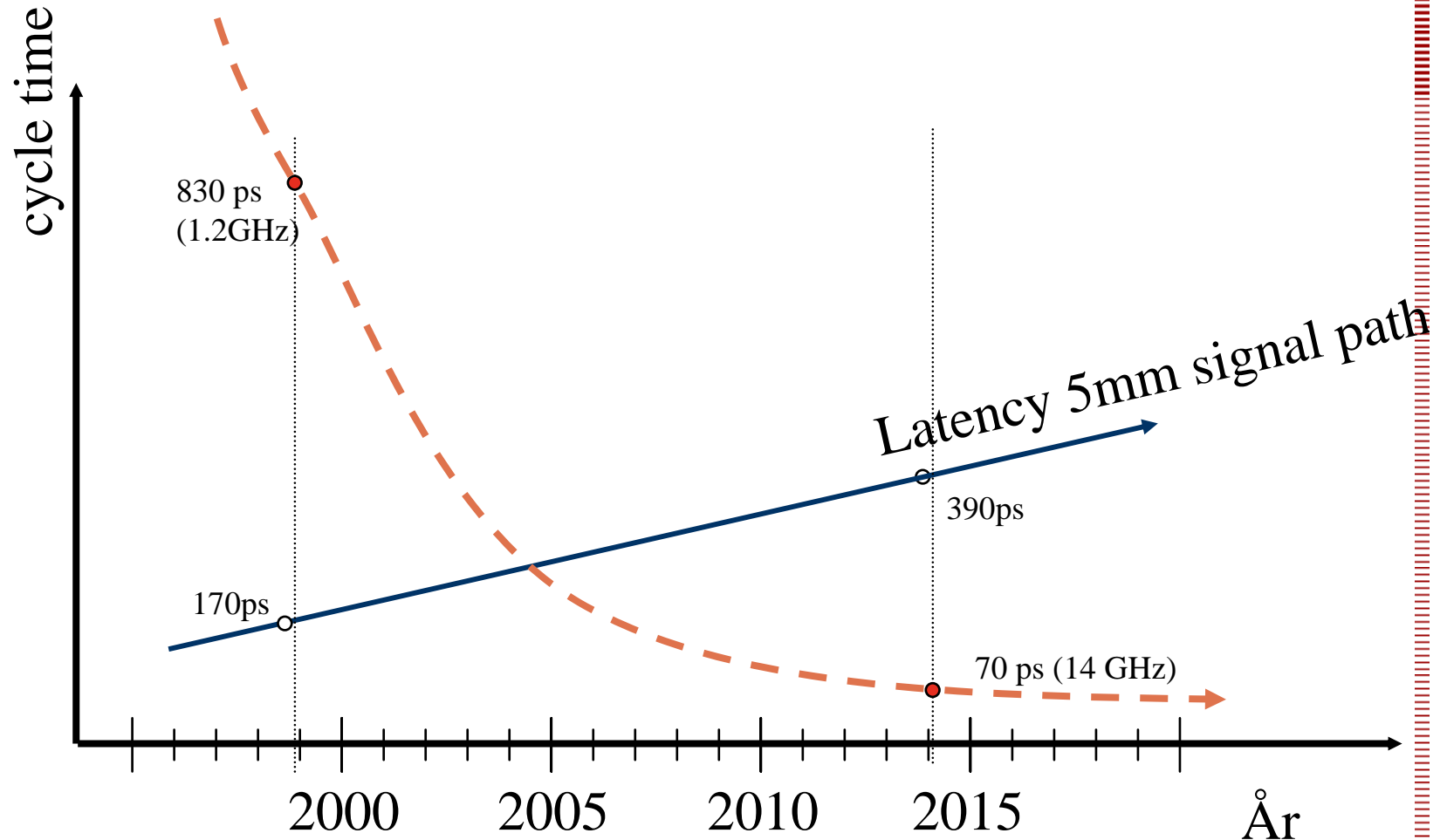
- Faster clocks → Deep pipelines
- Superscalar Pipelines
- Branch Prediction
- Out-of-Order Execution
- Trace Cache
- Speculation
- Predication
- Advanced Branch Target Cache
- Branch Target Cache
- Branch Target Cache

Bad News #1:
We have already explored most ILP
(instruction-level parallelism)



Bad News #2

Loong wire delay → slow CPUs



Quantitative data and trends according to V. Agarwal et al., ISCA 2000
Based on SIA (Semiconductor Industry Association) prediction, 1999



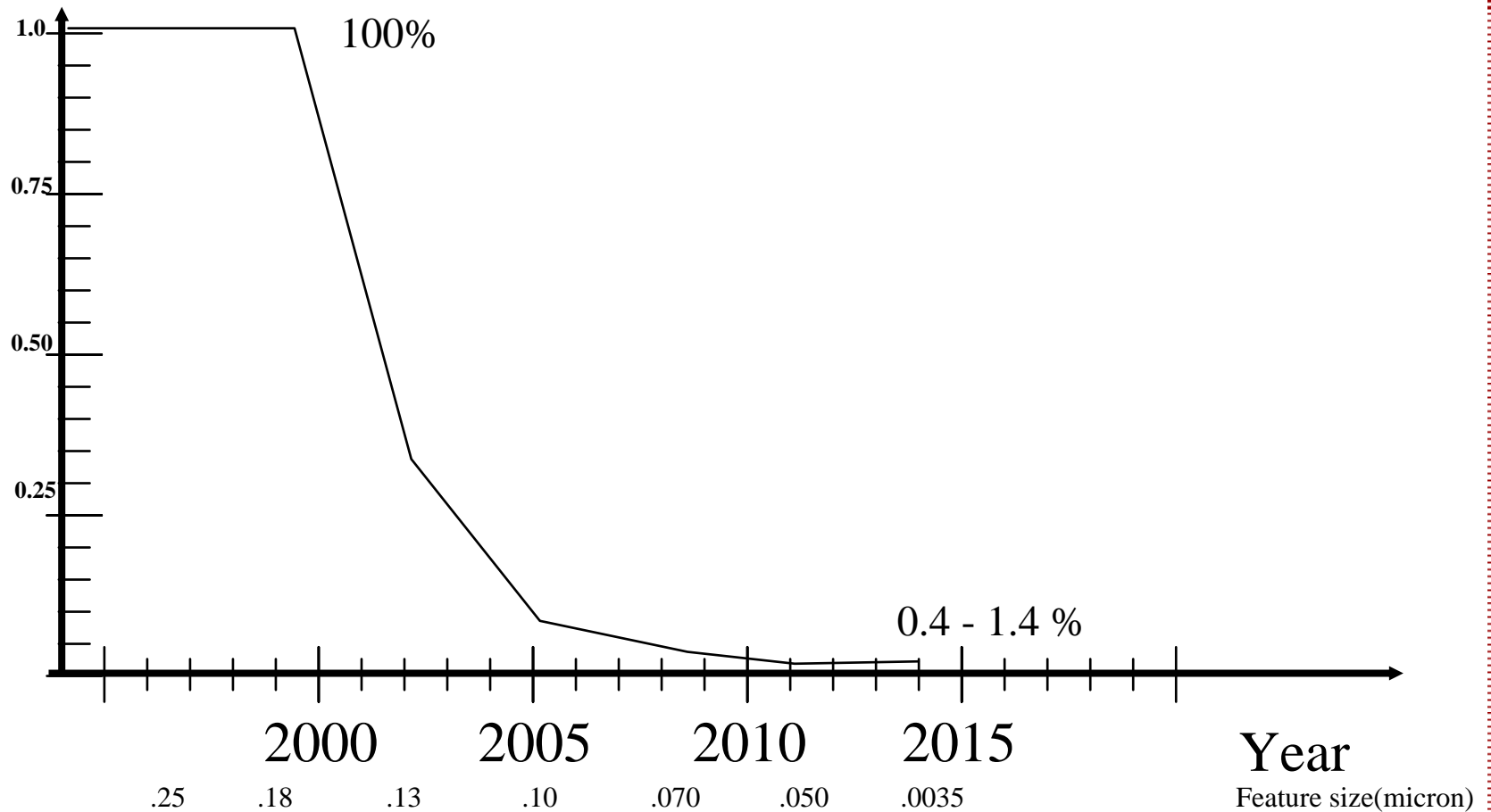
UPPSALA
UNIVERSITET

Uppsala University

Bad News #2

Loong wire delay → slow CPUs

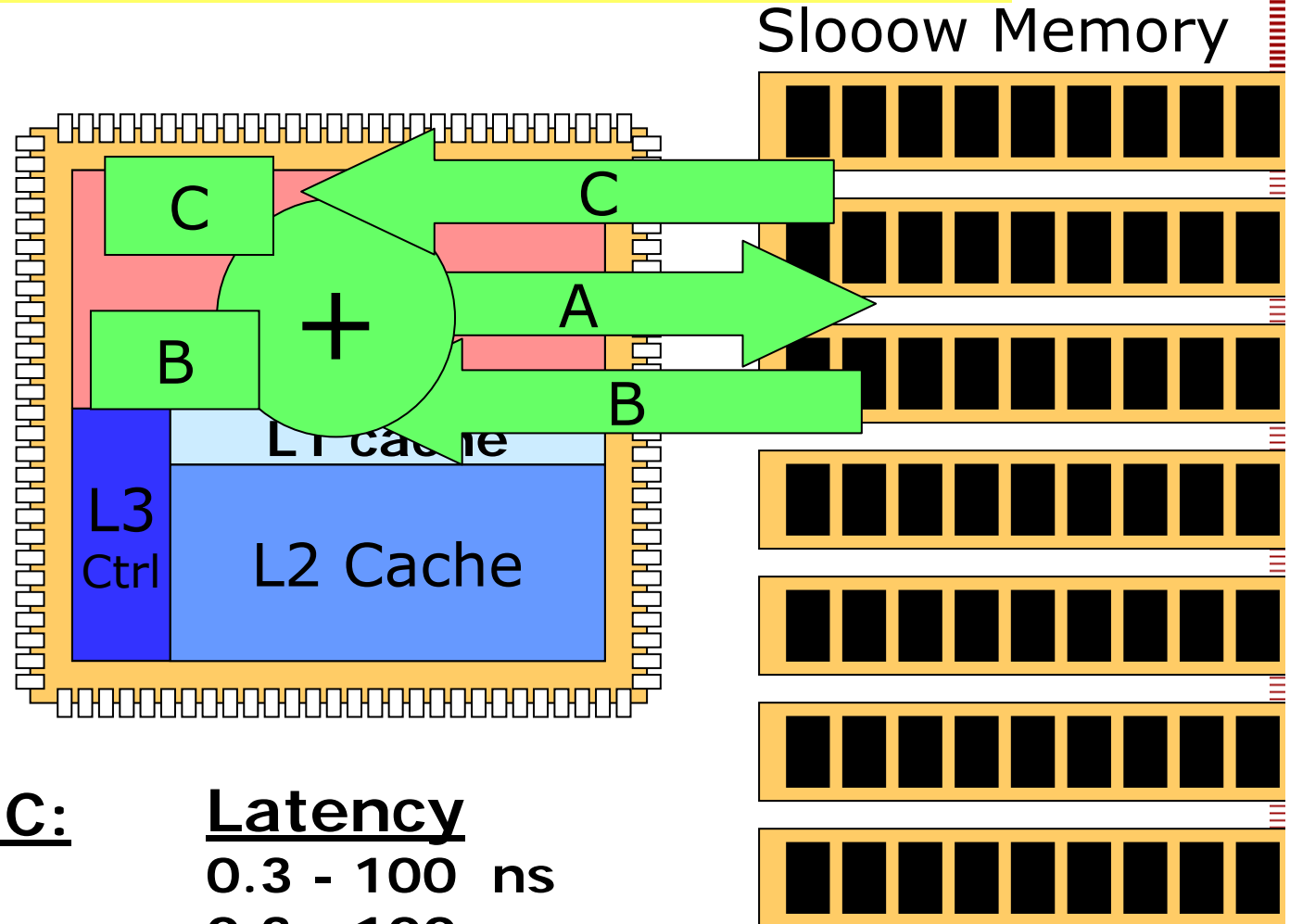
Span -- Fraction of chip reachable in one cycle





Bad News #3:

Memory latency/bandwidth is the bottleneck...



A = B + C:

Read B

Read C

Add B & C

WriteA

Latency

0.3 - 100 ns

0.3 - 100 ns

0.3 ns

0.3 - 100 ns



Bad News #4: Power is the limit

- Power consumption is the bottleneck
 - ✱ Cooling servers is hard
 - ✱ Battery lifetime for mobile computers
 - ✱ Energy is money
- Dynamic effect is proportional to
 - ~ Frequency
 - ~ Voltage²



Solving all the problems: exploring thread parallelism

#1: Running out of ILP

→ feed one CPU with instr. from many threads

#2: Wire delay is starting to hurt

→ Multiple small CPUs with private caches

#3: Memory is the bottleneck

→ memory accesses from many threads

#4: Power is the limit

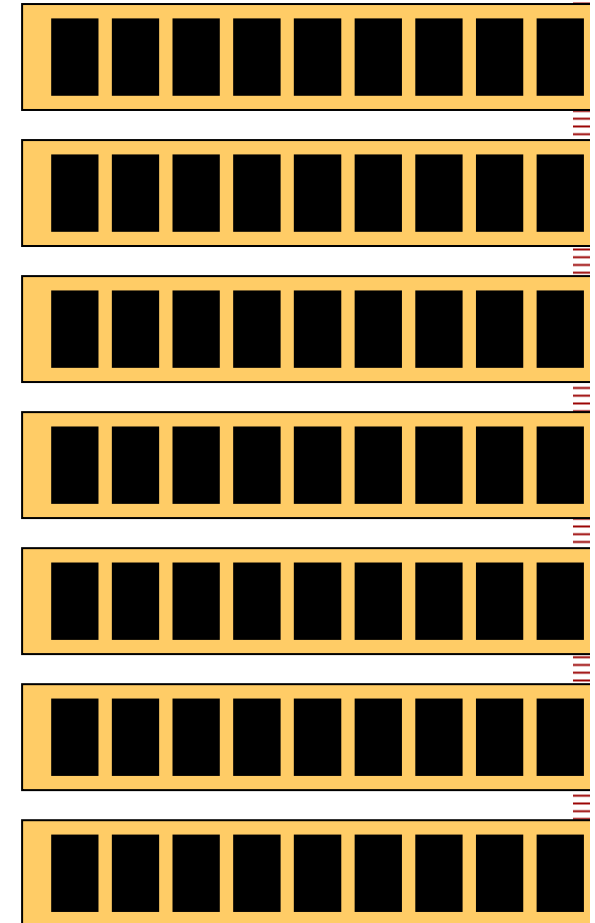
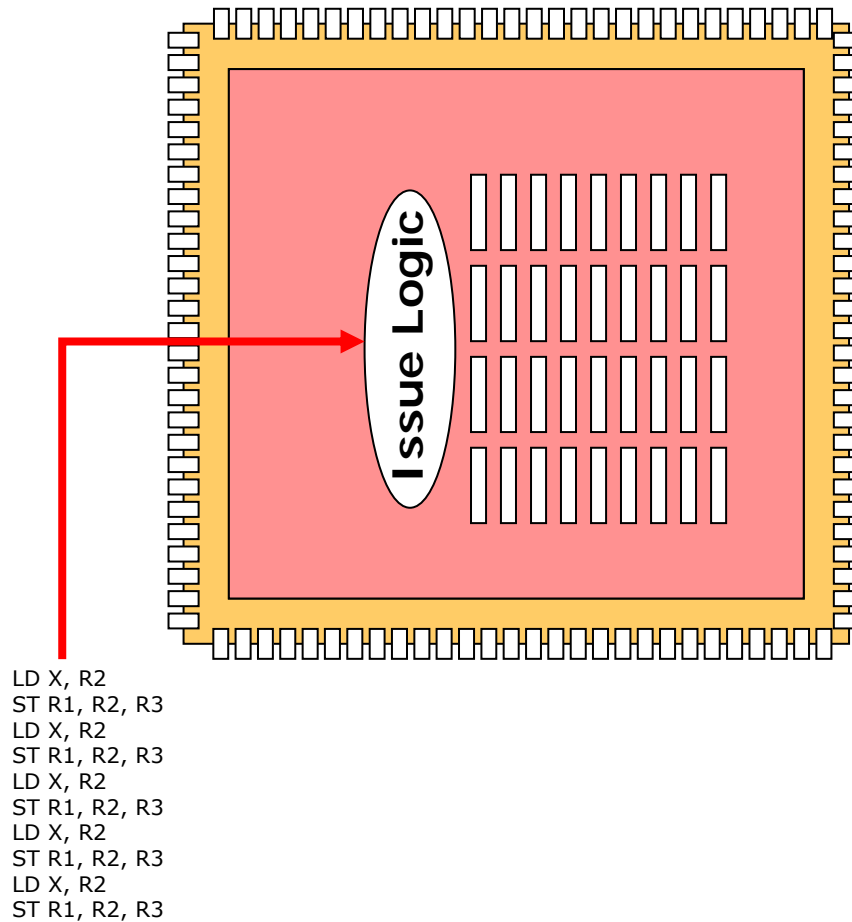
→ Lower the frequency → lower voltage



Bad News #1: Not enough ILP 1(2)

→ feed one CPU with instr. from many threads

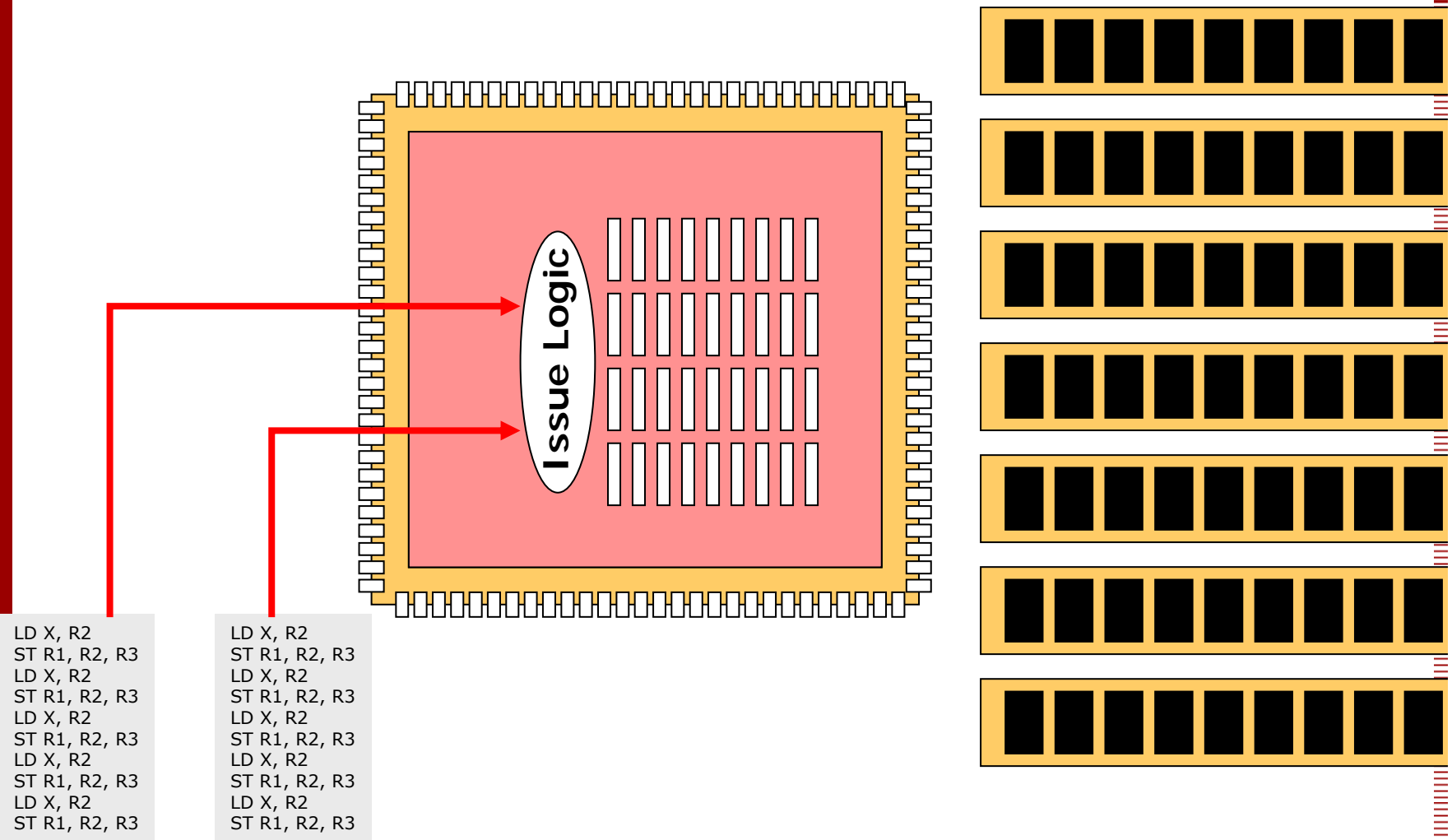
Sloooow Memory





Bad News #1: Not enough ILP 2(2)

→ feed one CPU with instr. from many threads





SMT is Not New ...

- Historical Examples (1984 ...)
Denelcor, HEP, Tera Computers [B. Smith]
 - Poor single-thread performance
 - Expensive

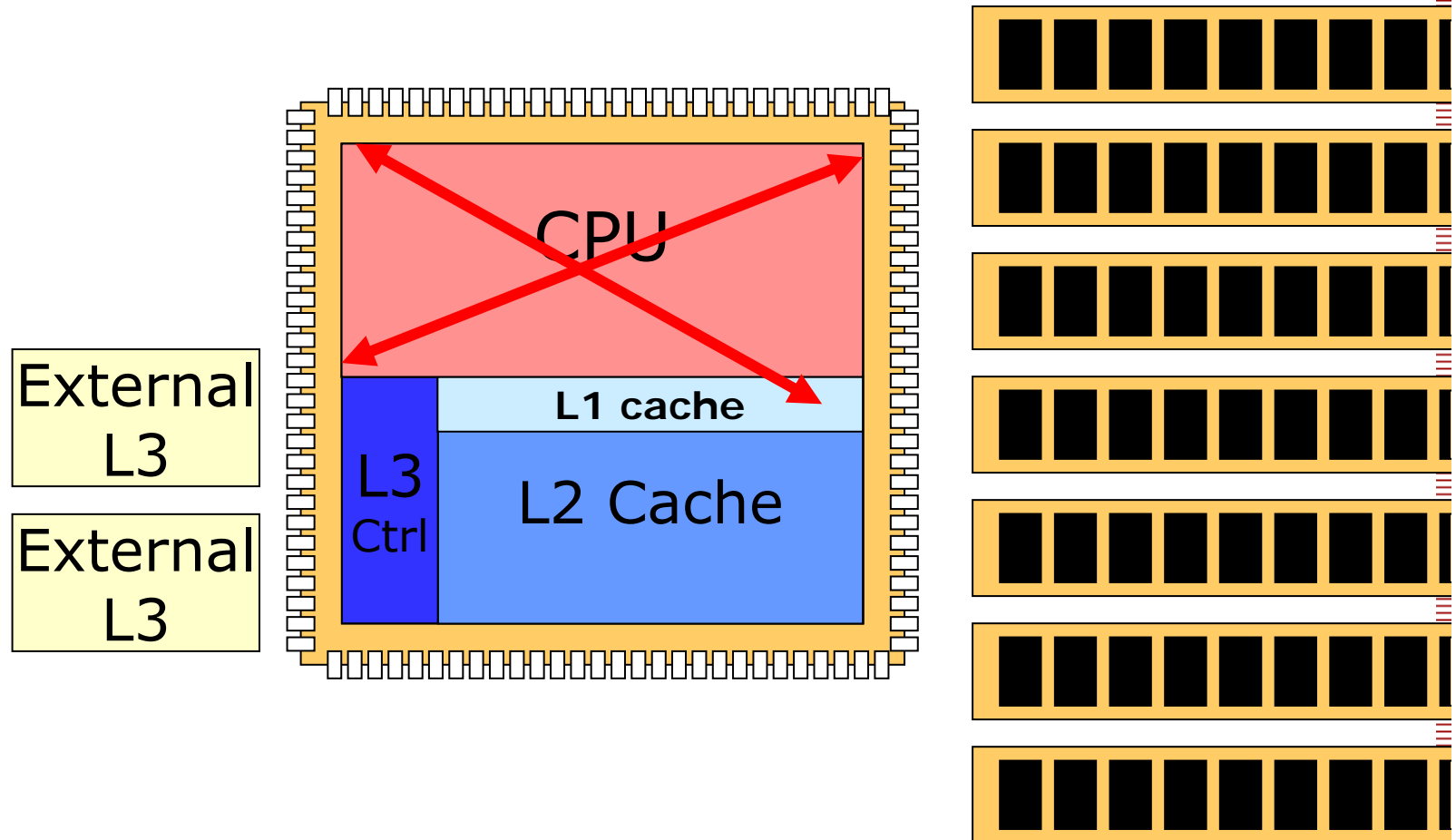


UPPSALA
UNIVERSITET

Uppsala University

Bad News #2: wire delay

→ Multiple small CPUs with private L1\$

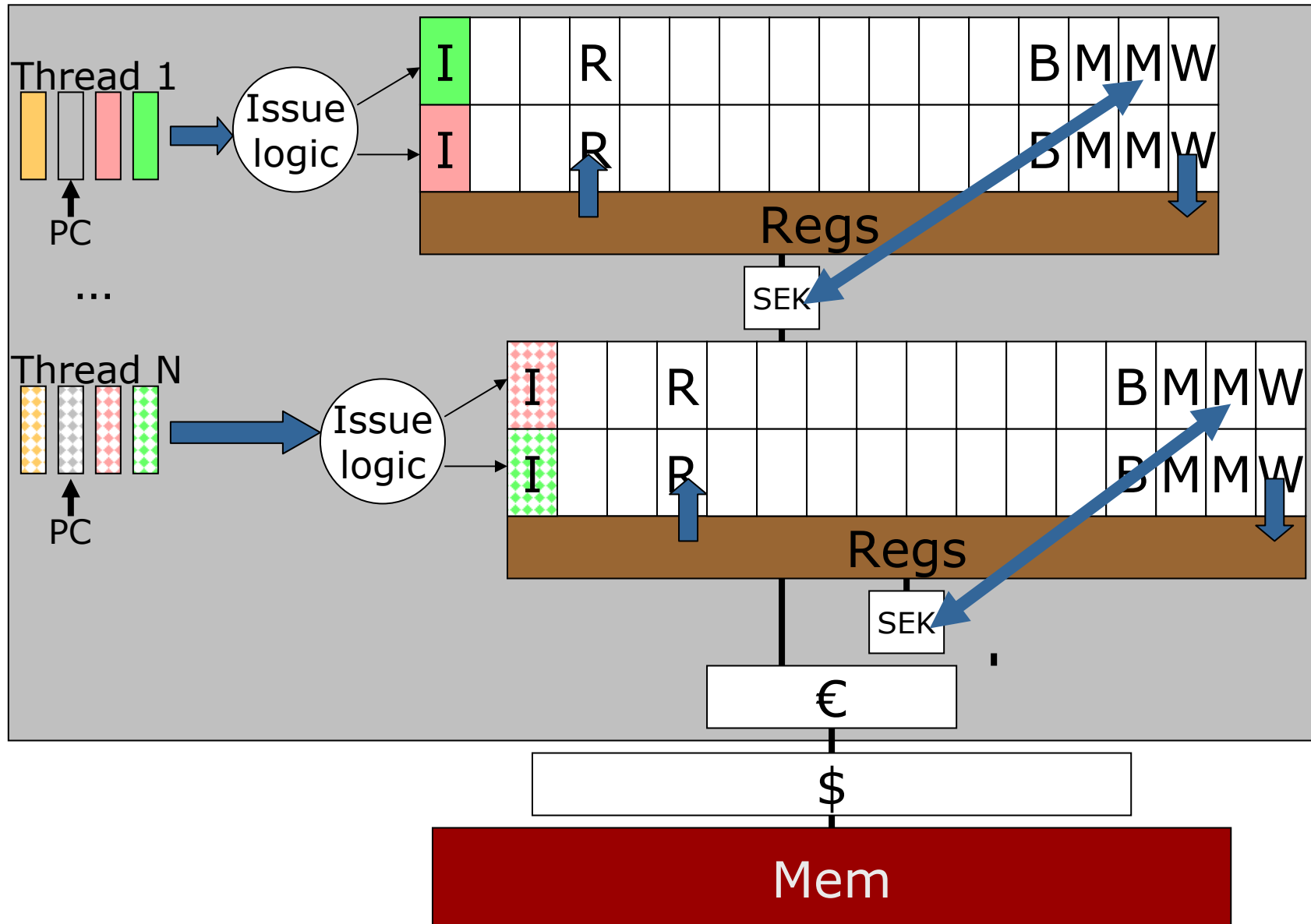




UPPSALA
UNIVERSITET

Uppsala University

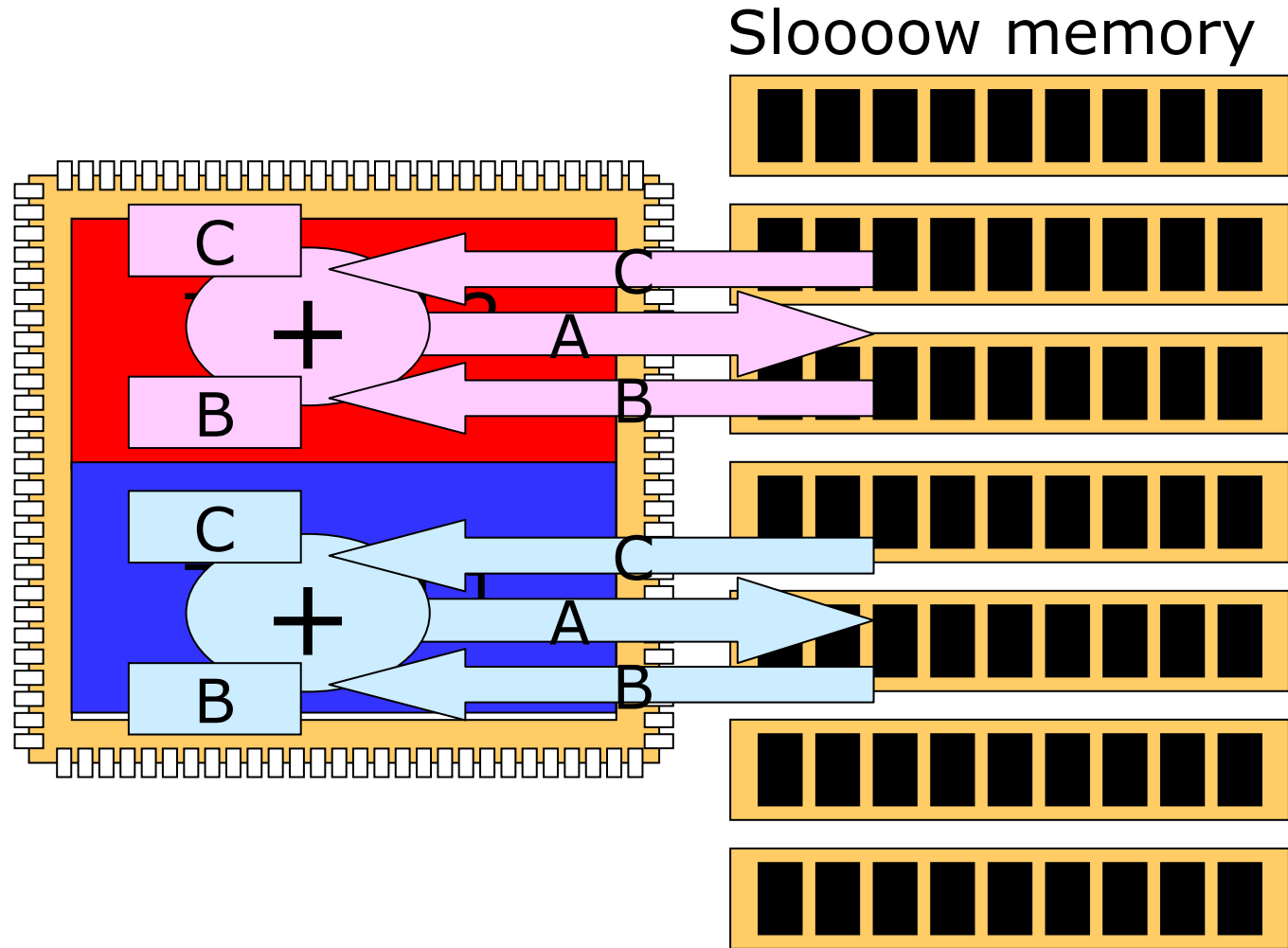
CMP: Chip Multiprocessor





Bad News #3: memory latency/bandwidth

→ memory accesses from many threads



Thread-Level Parallelism → Memory-Level Parallelism (MLP)



Bad News #4: Power consumption

→ Lower the frequency → lower voltage

$$P_{\text{dyn}} = C * f * V^2 \approx \text{area} * \text{freq} * \text{voltage}^2$$

CPU
freq=f

VS.

CPU
freq=f/2

CPU
freq=f/2

$$P_{\text{dyn}}(C, f, V) = CfV^2$$

$$P_{\text{dyn}}(2C, f/2, <V) < CfV^2$$

CPU
freq=f

VS.

CPU	CPU
CPU	CPU

freq = f/2

$$P_{\text{dyn}}(C, f, V) = CfV^2$$

$$P_{\text{dyn}}(C, f/2, <V) < \frac{1}{2} CfV^2$$



Why multiple cores/threads?

- Instruction level parallelism is running out
 - ✦ Feed the CPU with instructions from many threads (Simultaneous MultiThreading, SMT)
- Wire delay is hurting now
 - ✦ Use the chip for multiple smaller CPU cores (Chip MultiProcessors, CMP)
- Power is a limit now
 - ✦ Lower the frequency (which enables lower voltage)
- Memory latency/bandwidth bottleneck
 - ✦ Access the memory from many threads



UPPSALA
UNIVERSITET

Uppsala University

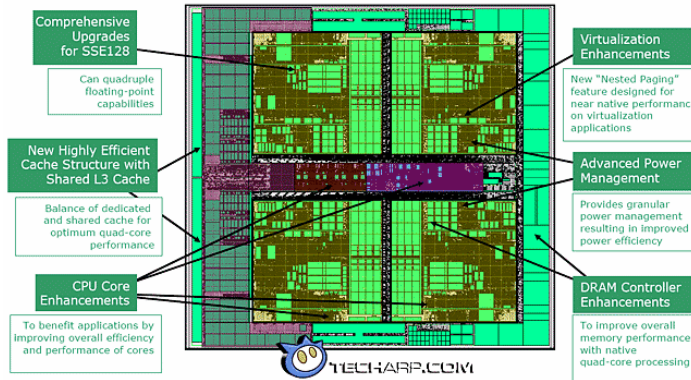
Multicore Systems, Now and in the Future



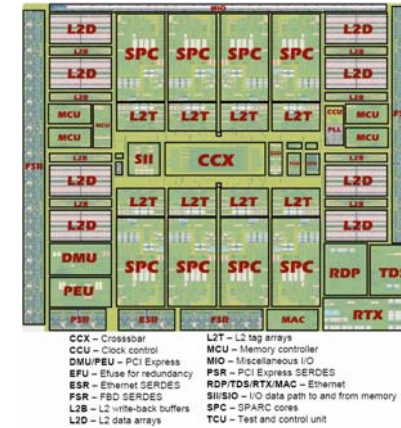
UPPSALA
UNIVERSITET

Uppsala University

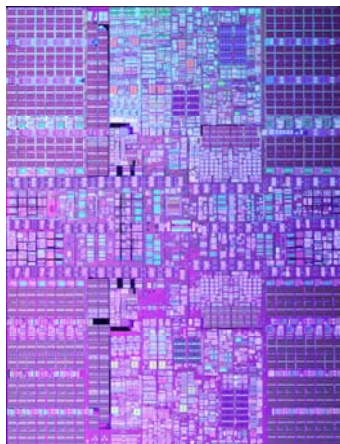
Some Current Multicore Proc.



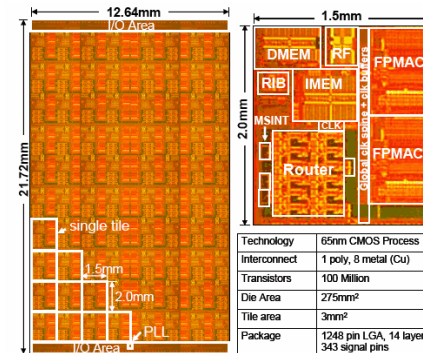
Quad-core AMD Opteron (Barcelona)



Eight-core, 64 thread Sparc T2 (Niagara 2)



Dual-core, four thread IBM Power6



80-core "Intel Teraflop Research Chip"



CMP Design Space

- Number of cores/chip
- Type of core (in-order, scalar, superscalar)
- Core clock frequency
- Number of cache levels and cache structure (L1,L2,L3,...; shared or local; associativity, ...)
- Threads per core (1 or SMT)
- Number and type of channels to memory, on-chip memory controller or not, ...



CMP Design Space

"Fat cores" \Leftrightarrow "Thin cores" ?

- In PCs today: A few fat, supercalar cores with rather high clock frequency
- In servers tomorrow: Many thin, in-order cores with mediocre clock frequency ??



Some Current Multicore Proc.

- **Intel Quad-core:** Two dual-core on a chip, 3.0 GHz, 2*4 MB L2, superscalar (August 13)
- **AMD Quad-core:** Monolithic, 2.0 GHz, Local L1, L2, shared L3 and memory controller on chip, superscalar (September 10)
- **IBM Power6:** Dual core, 2-way SMT. 4.7 GHz, 8MB L2, in-order (May 21)
- **Sun SPARC T2:** Eight-core, 8-way SMT, 1.4 GHz, 4MB L2, in-order (August 7)



The Multicore Era has been here for a while!

- *Intel have 10 projects in the works that contain four or more computing cores per chip* [Paul Otellini, Intel Chief Executive at IDF fall 2005]
- All major microprocessor vendors now focus on CMPs (with fat and/or with thin cores)
- Rumours of a 128-thread microprocessor in a 2048-thread single-box system in 2008



UPPSALA
UNIVERSITET

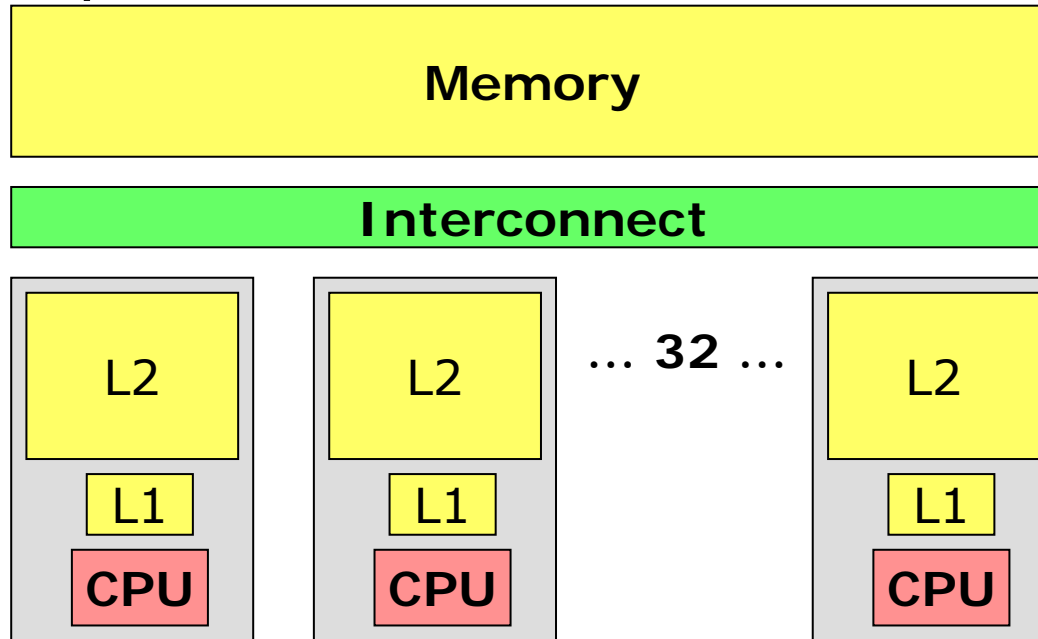
Uppsala University

Impact on Performance

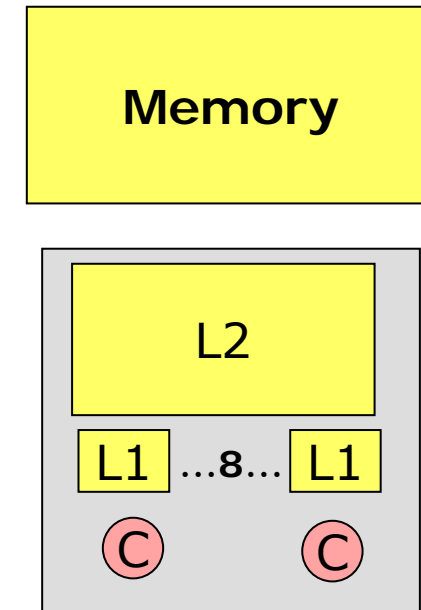


Isn't a CMP just an SMP on a chip?

Symmetric MultiProcessor, SMP



CMP



Well, how about:

- Cost of parallelism?
- Cache capacity per thread?
- Memory bandwidth per thread?
- Cost of thread communication?



Cost of parallelism

- *Individual processor [core] costs will drop so quickly that in the near future the world can view processors as a nearly free commodity* [IDC(*) presentation on HPC at International Supercomputing Conference, ICS07, Dresden, June 26-29 2007]

(*) IDC (International Data Corporation) is a major market research and analysis firm specializing in information technology, telecommunications and consumer technology.



Cache per Thread

- **Intel Quad-core:** 2 MB/thread L2
- **IBM Power6:** 2 MB/thread L2
- **AMD Quad-core:** 512 kB local L2, 512 kB/thread L3
- **Sun SPARC T2:** 64 kB/thread L2



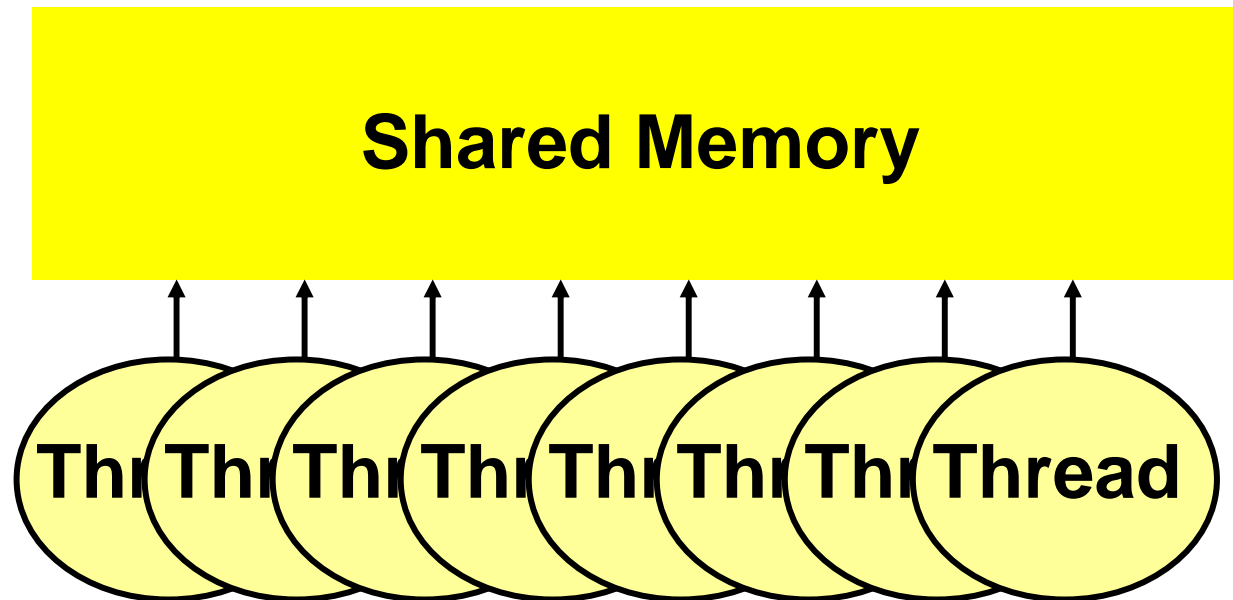
Memory Bandwidth

- *Memory bandwidth to a single processor core will decline at a fast rate* [IDC presentation on HPC]
- *The flow of data will become the major road block and hence the area of opportunity for performance speedups* [IDC...]



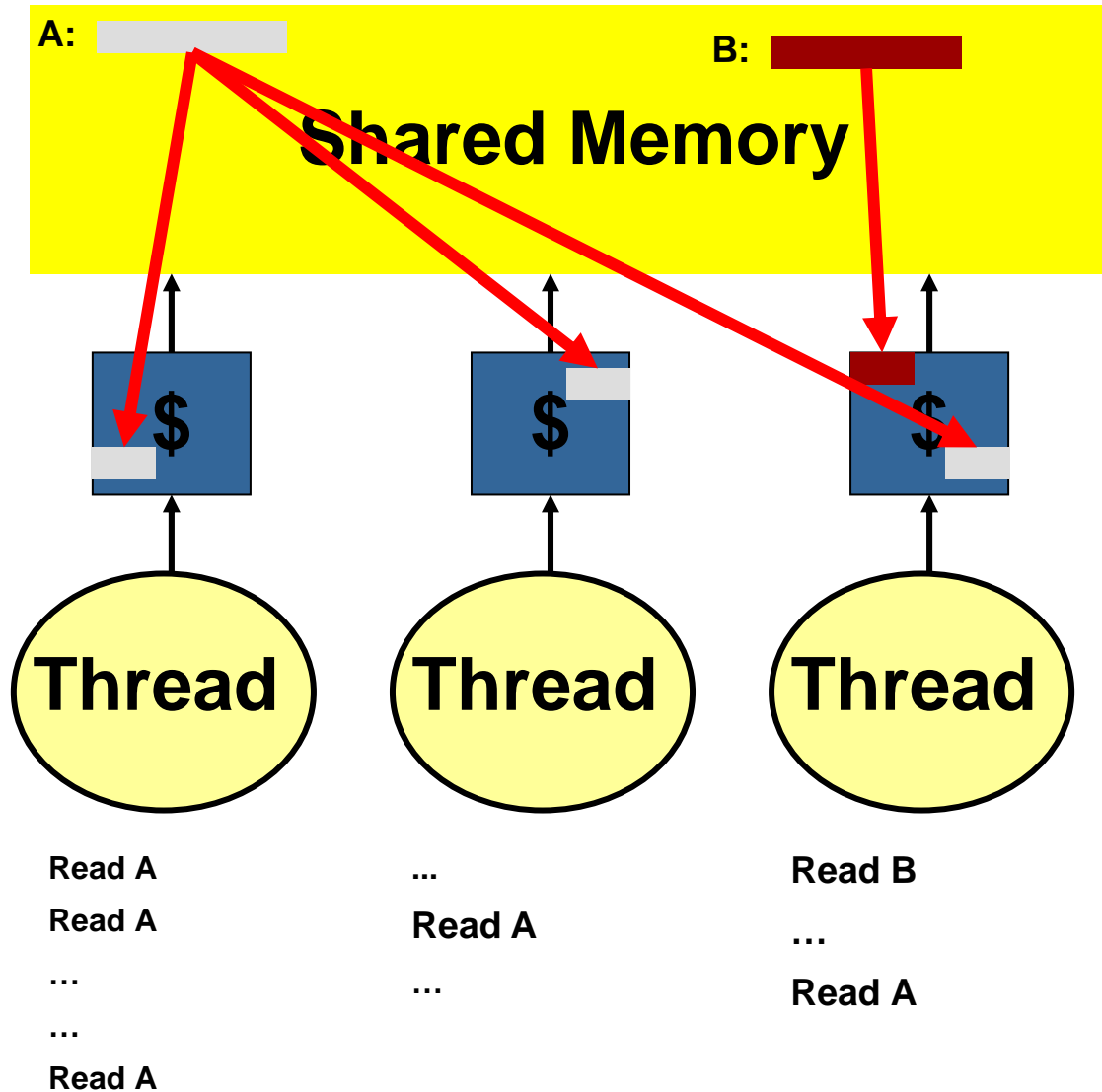
Cost of Thread Communication

Multithread programming model (e.g. OpenMP):



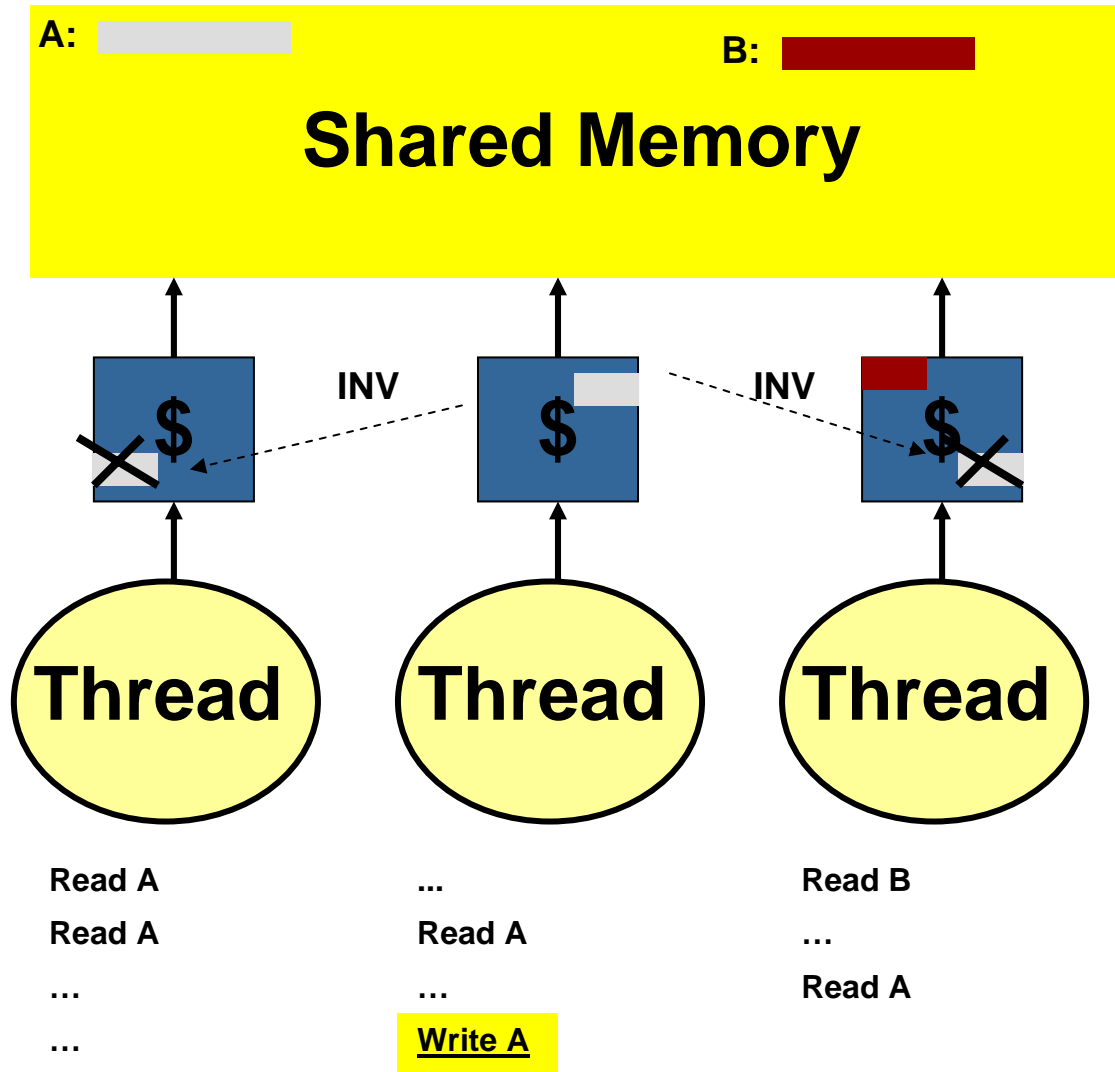


Caches: Automatic Replication of Data



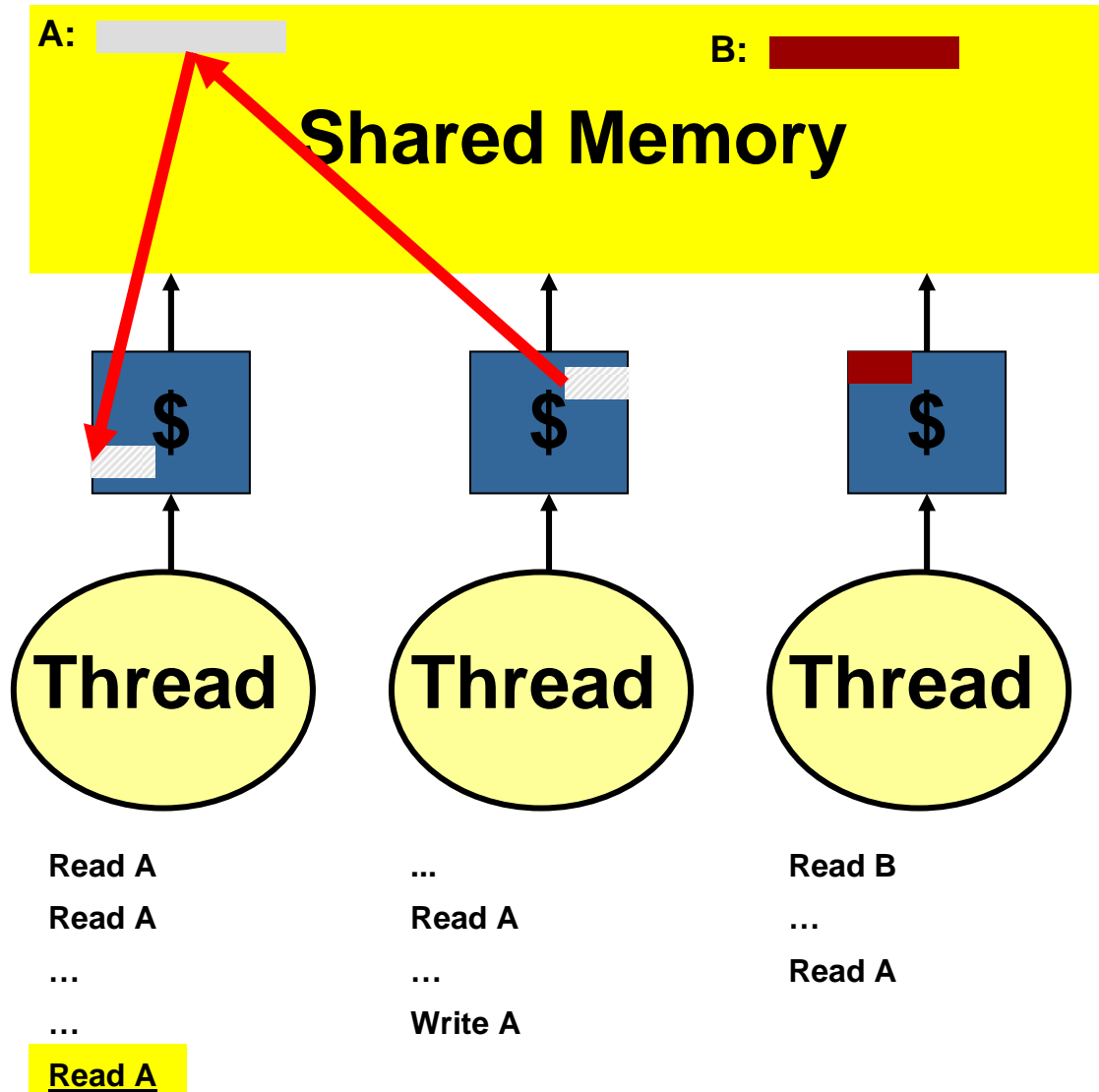


The Cache Coherent Memory System





The Cache Coherent Cache-to-cache

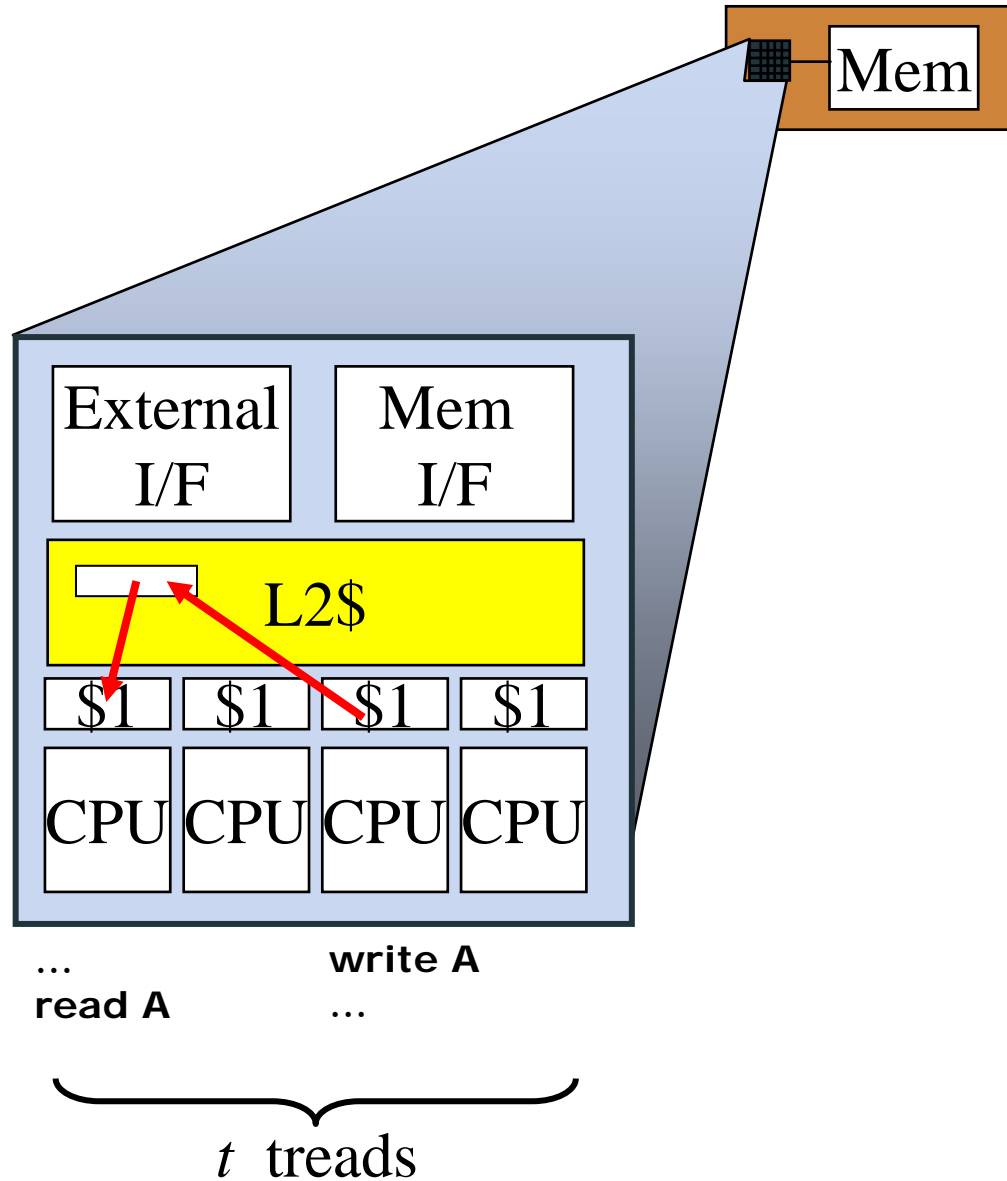




UPPSALA
UNIVERSITET

Uppsala University

Communication in a CMP





Do we need to bother?

IDC presentation again ...

For high-performance computing (HPC) applications, [multicores] introduces an additional layer of complexity, which will force users to go through a phase change in how they address parallelism or risk being left behind.

Users who find new methodologies for dealing with the added level of parallelism could see rapid evolution in capability as processor cores become dramatically less expensive



UPPSALA
UNIVERSITET

Uppsala University

Similar Thoughts are Shared by Others

The Matlab News&Notes Newsletter,
Summer 2007:

Cleve's Corner: Parallel MATLAB

The proliferation of multicore systems and clusters sets the stage for parallel computing with MATLAB.

Programming Patterns: Maximizing Code Performance by Optimizing Memory Access

Improve memory usage and increase code speed by understanding how MATLAB stores and accesses data.

...



UPPSALA
UNIVERSITET

Uppsala University

Impact on CSE Software in General ?



UPPSALA
UNIVERSITET

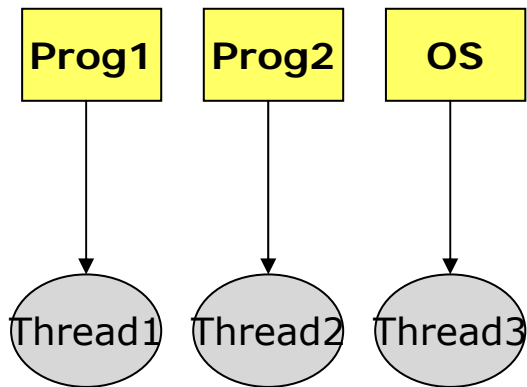
Uppsala University

Some Thoughts ...

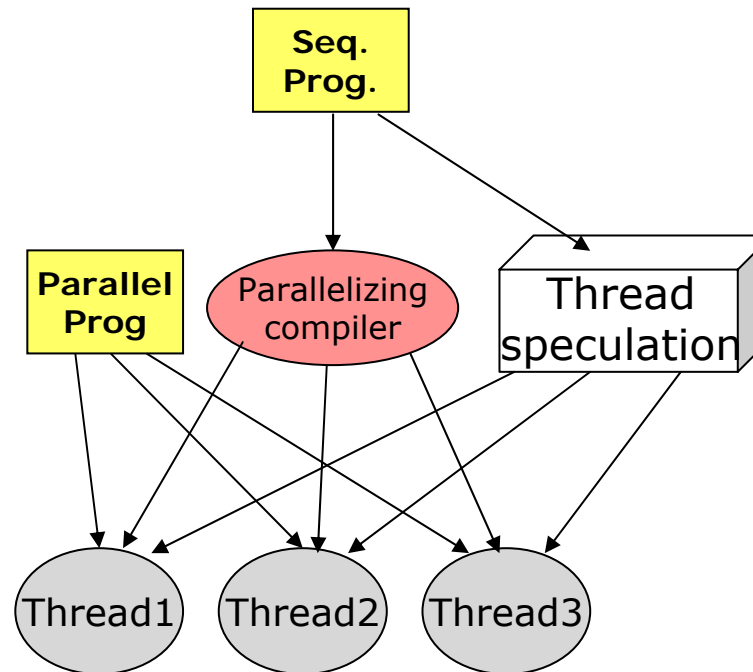
- Parallel programming will be used for applications where performance is important
 - ✿ Parallel programming will become mainstream
 - ✿ Large efforts will be put into tools, programming environments etc



How do you use the parallelism?



Capacity Computing



Capability Computing



Some Thoughts ...

- Can you not just run different jobs/processes on different threads for increased capacity? Remember:
 - ✱ Less cache/thread
 - ✱ Less memory bandwidth/thread
 - ✱ (Can get some help from adaptive scheduling etc.)



More Thoughts ...

- Better tools for performance analysis and optimization are indeed needed
- Better tools for testing, verification and debugging of parallel codes are also needed

For CSE software:

- Autotuning for critical kernels
- Libraries, re-use of code and implementations becomes even more important
- Another motivation for lifting the level of abstraction



Thoughts on Programming CMP systems

- Hybrid programming models will be used (Message passing + multithreading)
- What is really the right programming model?
- OpenMP? What about data placement?
- PGAS?
- Can hardware be of assistance? (e.g. transactional memory)
- Scalability!?! (2048 threads in a single-box server in 2008?)



UPPSALA
UNIVERSITET

Uppsala University

Impact on Algorithms

For performance, we need to understand the interaction between algorithms and architecture.

The rules have changed!

Do we need to question old algorithms and results?



Criteria for algorithm design

- Pre-CMP:
 - ✱ Minimize communication
 - ✱ Data locality is important
 - ✱ Maximize scalability for large-scale applications

- Within a CMP chip today:
 - ✱ (On-chip) communication is almost to free
 - ✱ Data locality is even more important
 - ✱ Scalability to 2-32 threads

- In a multi-CMP system tomorrow:
 - ✱ Communication is sometimes almost free (on-chip), sometimes very expensive (between chips)
 - ✱ Data locality (minimizing of-chip references) is the key to efficiency
 - ✱ “Hierarchical scalability”

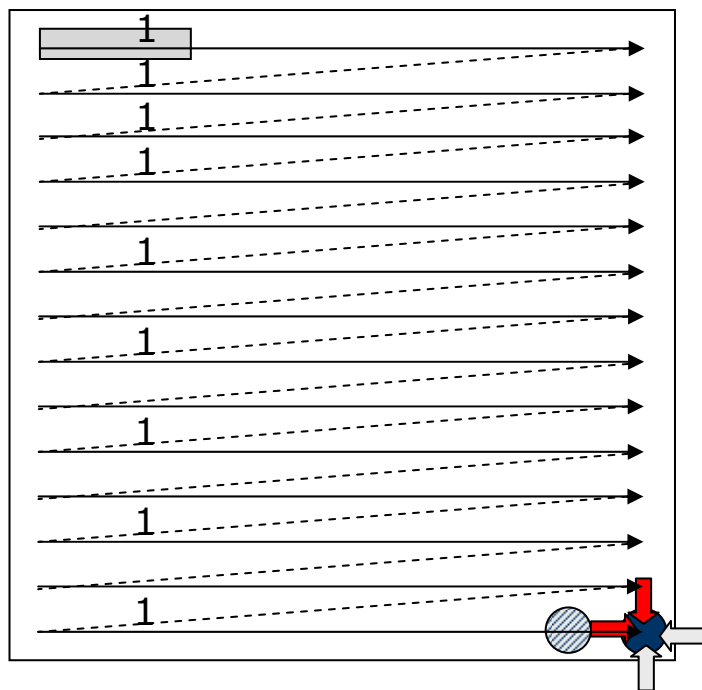


UPPSALA
UNIVERSITET

Uppsala University

Example: Multigrid on Multicore Systems

Natural Order Gauss-Seidel

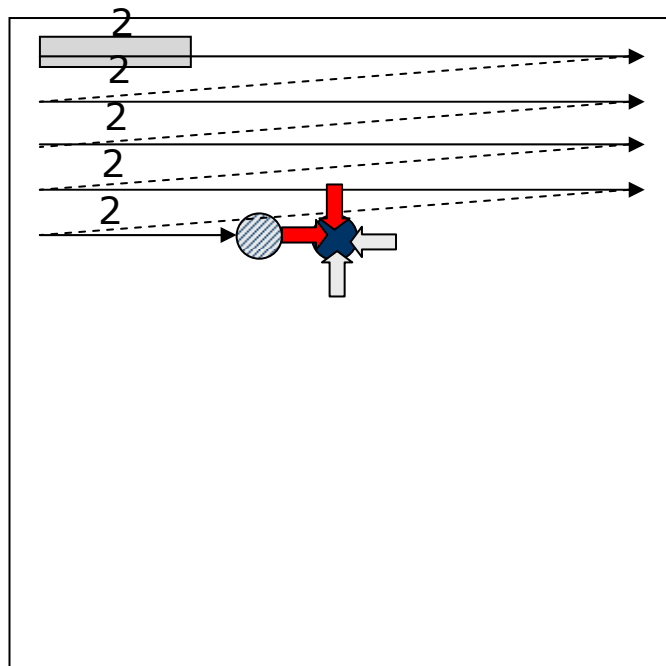


- = sweep path
- ⊘ = previous
- = current
- = data dependence
- 1,2,3,4 = iteration number
- ▭ = cacheline layout

```

IF (convergence_test)
  <done>
else
  <iterate again>
  
```

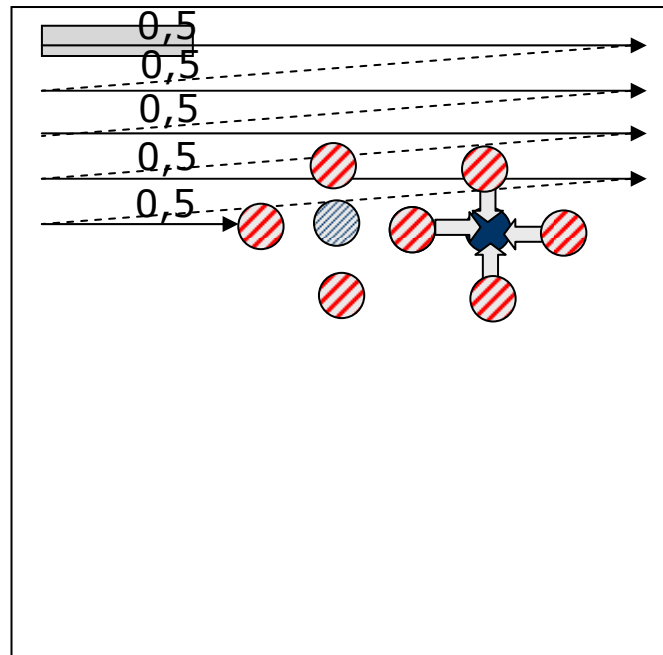
Natural Order Gauss-Seidel



- = sweep path
- ⊙ = previous
- = current
- ➔ = data dependence
- 1,2,3,4 = iteration number
- ▭ = cacheline layout

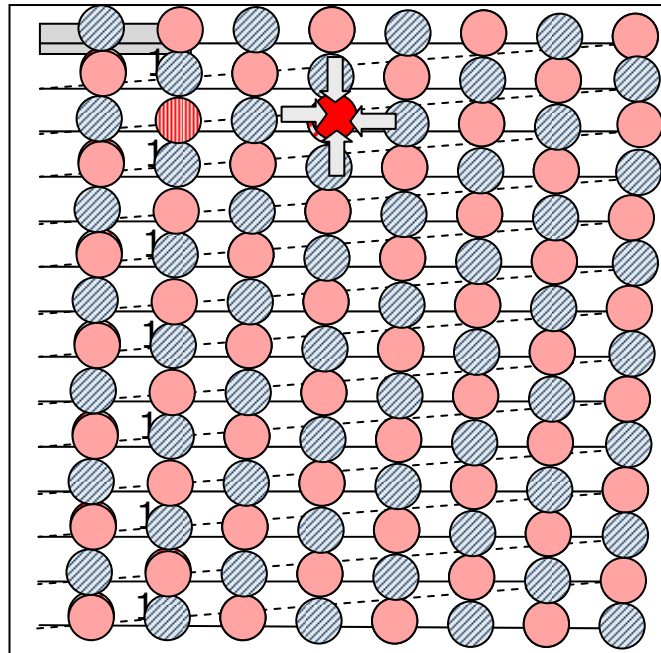
Data dependence → Poor Parallelism ☹️

Red-Black Gauss-Seidel step 0,5: update the blacks



- = sweep path
- ⊘ = previous
- = current
- = data dependence
- 1,2,3,4 = iteration number
- ▭ = cacheline layout

Red-Black Gauss-Seidel step 1,0 update the reds



- = sweep path
- = previous
- = current
- = data dependence
- 1,2,3,4 = iteration number
- ▭ = cacheline layout

Update all blacks

<barrier>

Update all reds

<barrier>

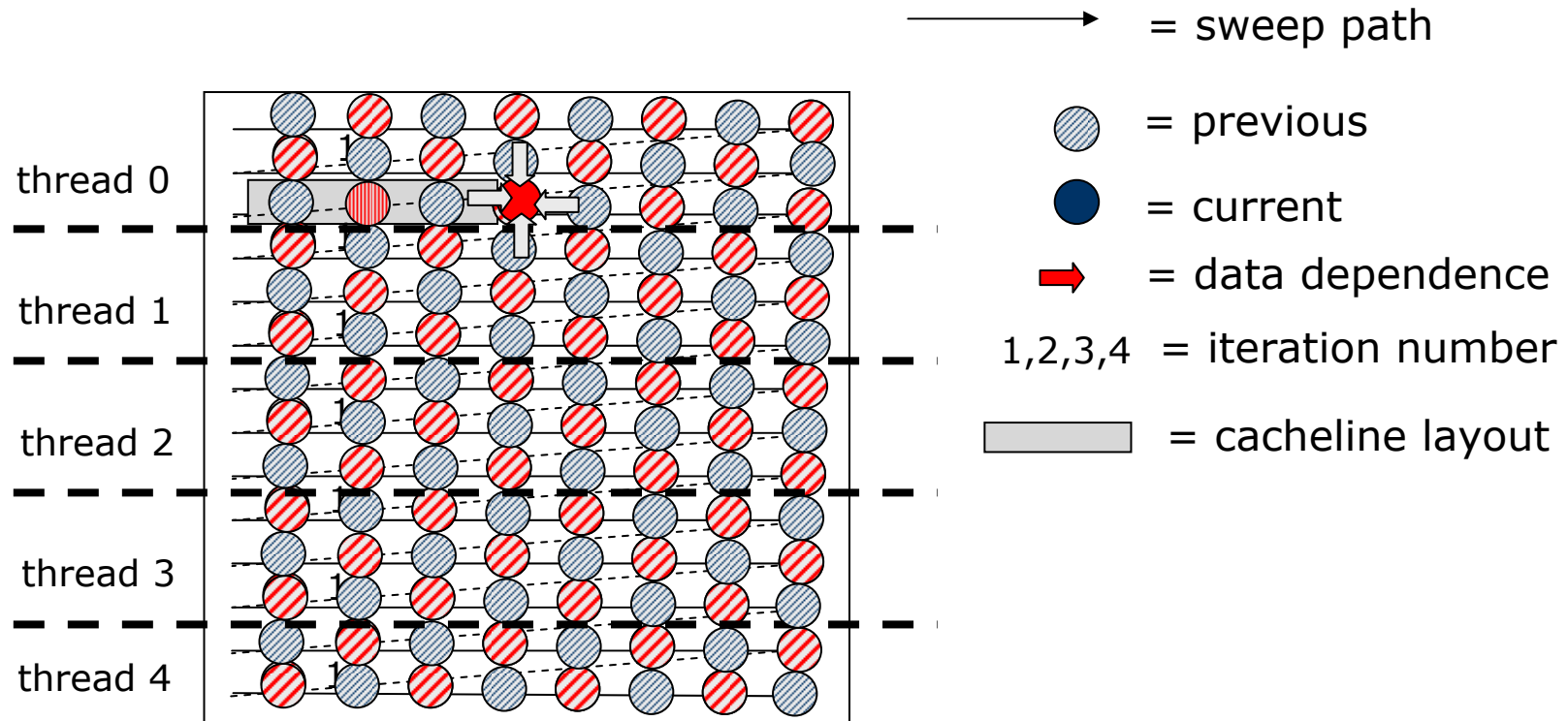
→ great parallelism!!!



UPPSALA
UNIVERSITET

Uppsala University

Red-Black Gauss-Seidel Parallel version



```

IN PARALLELL {
  Update all blacks
  <barrier>
  Update all reds
  <barrier>

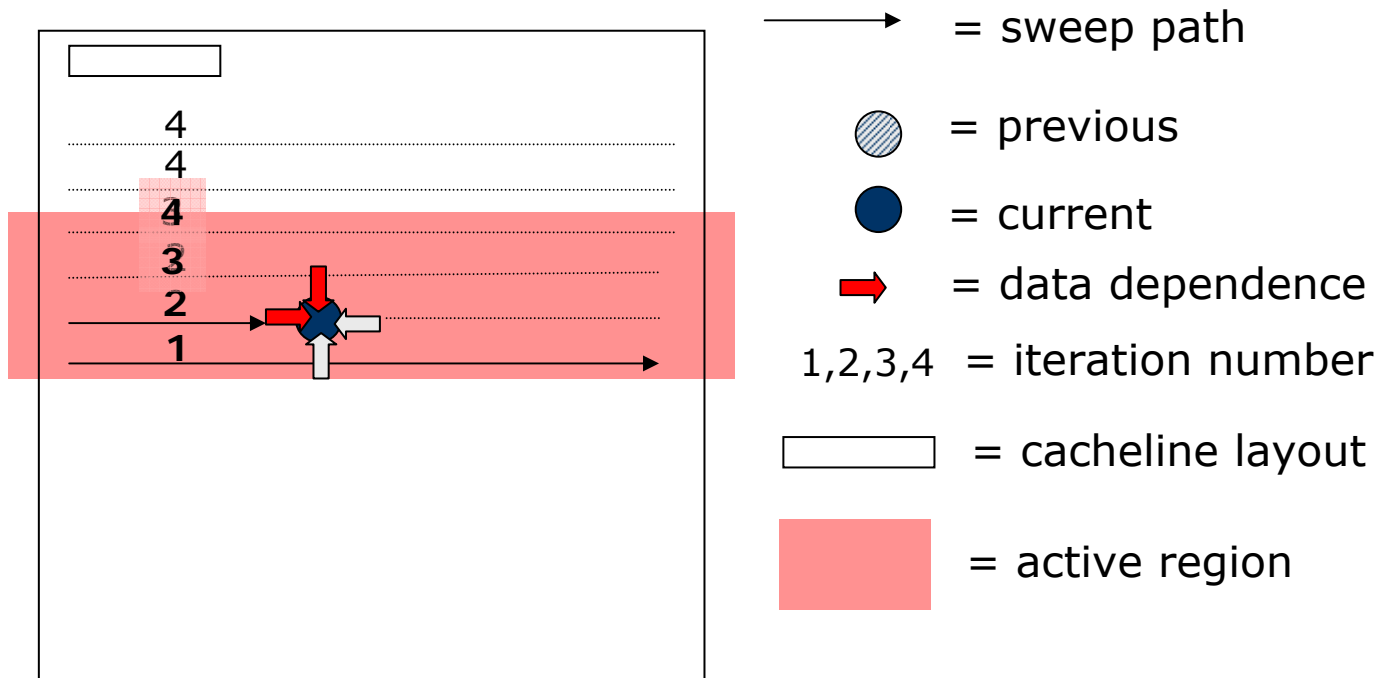
```



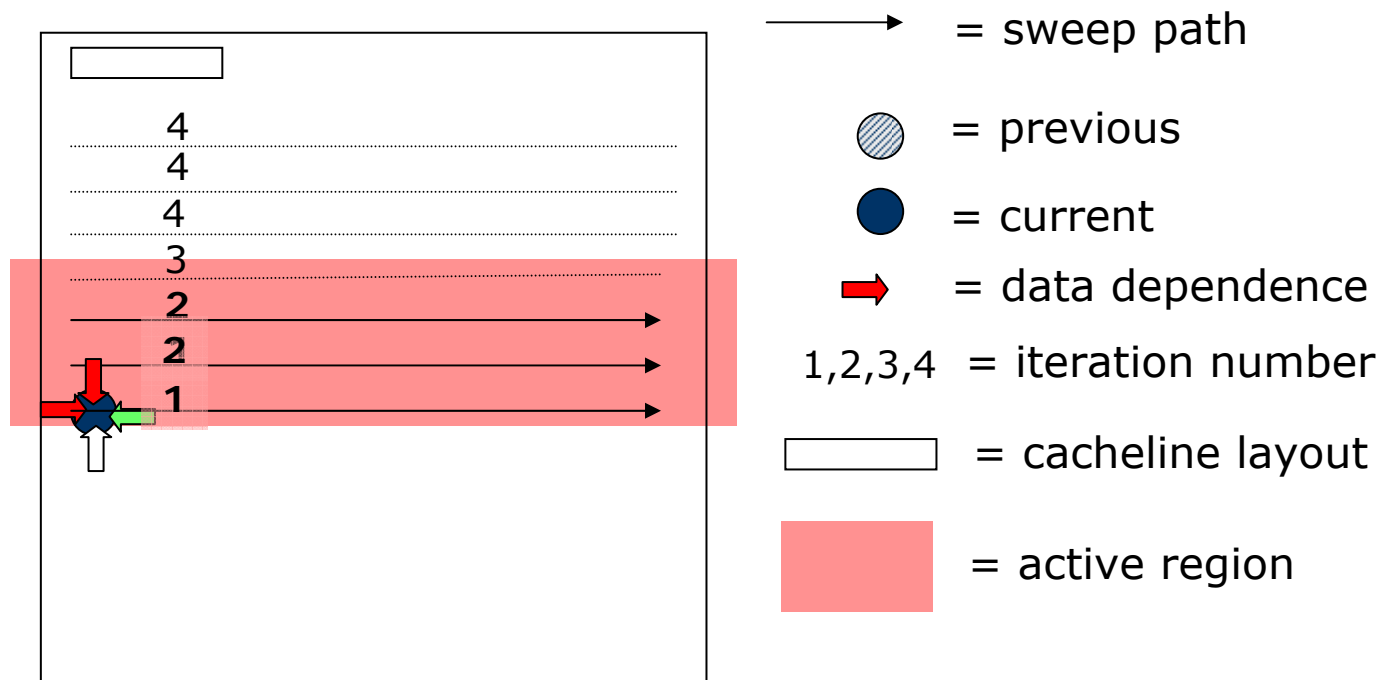
Any Drawbacks?

- Red-Black:
 - ✱ Each element will be brought into the cache **twice** per iteration ☹️
- Natural Order:
 - ✱ Each element will be brought into the cache **once** per iteration ☺️
- You can do better...
 - ➔ **Temporal Blocking** ☺️

G-S, temporal blocking



G-S, temporal blocking



In this case: 4 iterations (*steps*) per *sweep*.

$\sigma = \# \text{steps per sweep}$

($\sigma = 1,0$ for natural order and $\sigma = 0,5$ for red-black)



Sequential Execution time per step (US4+, L1=16kB,L2=2MB,L3=32MB)

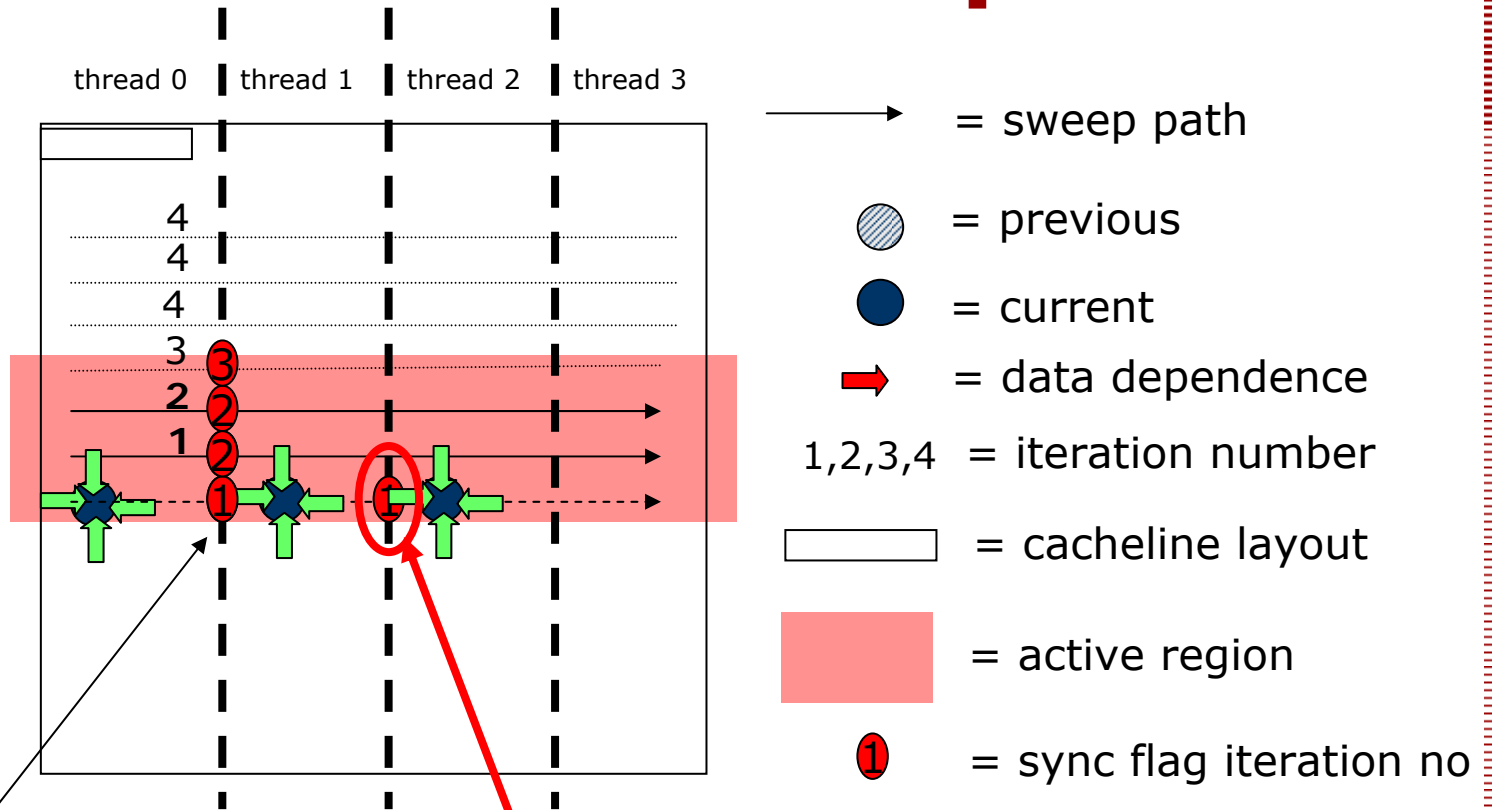
	RBGS	TBGS	TBGS	TBGS	TBGS	TBGS
σ	0.5	1	2	4	8	16
N=129	0.091	0.091	0.076	0.067	0.076	0.079
N=257	2.476	1.573	1.104	0.869	0.752	0.694
N=513	19.93	12.64	9.419	7.827	10.30	12.95



Required data size for each active region (US4+, L1=16kB,L2=2MB,L3=32MB)

σ	1	2	4	8	16
N=129	0.5 MB	0.76 MB	1.3 MB	2.3 MB	4.3 MB
N=257	2.0 MB	3.0 MB	5.0 MB	9.1 MB	17 MB
N=513	8.0 MB	12 MB	20 MB	36 MB	68 MB

Parallel G-S, temp block



Synchronization flags

Wait until "lefty" is done:
Lots of communication

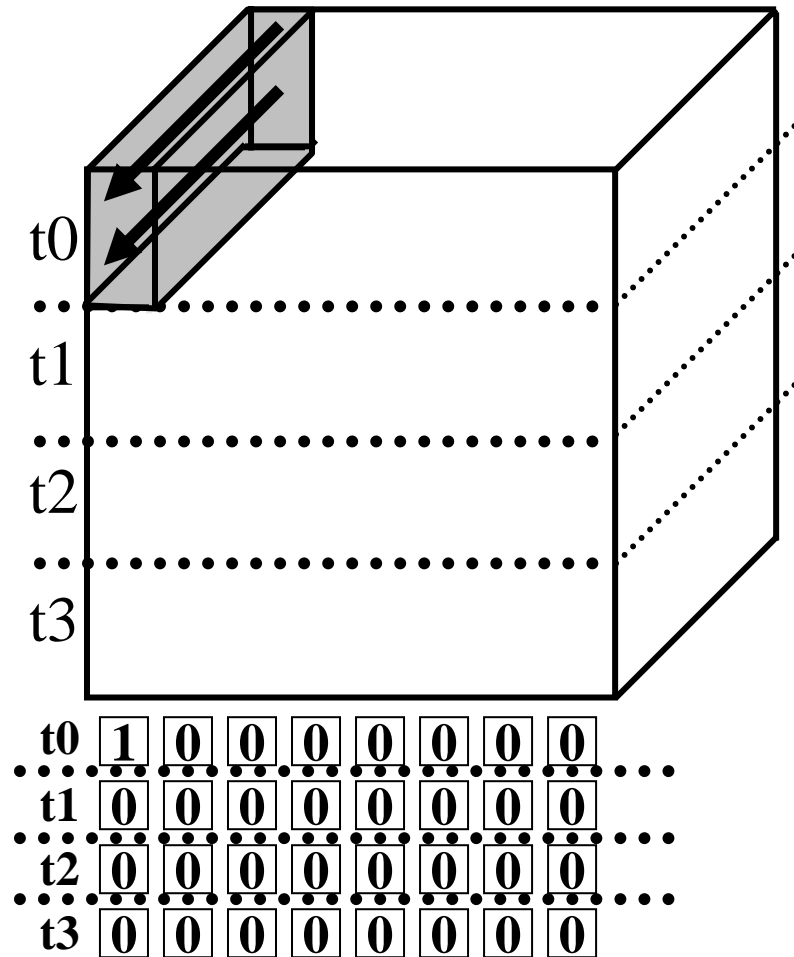
- Producer/Consumer Flag
- Sharing of data values



UPPSALA
UNIVERSITET

Uppsala University

Parallel Natural-order G-S in 3D

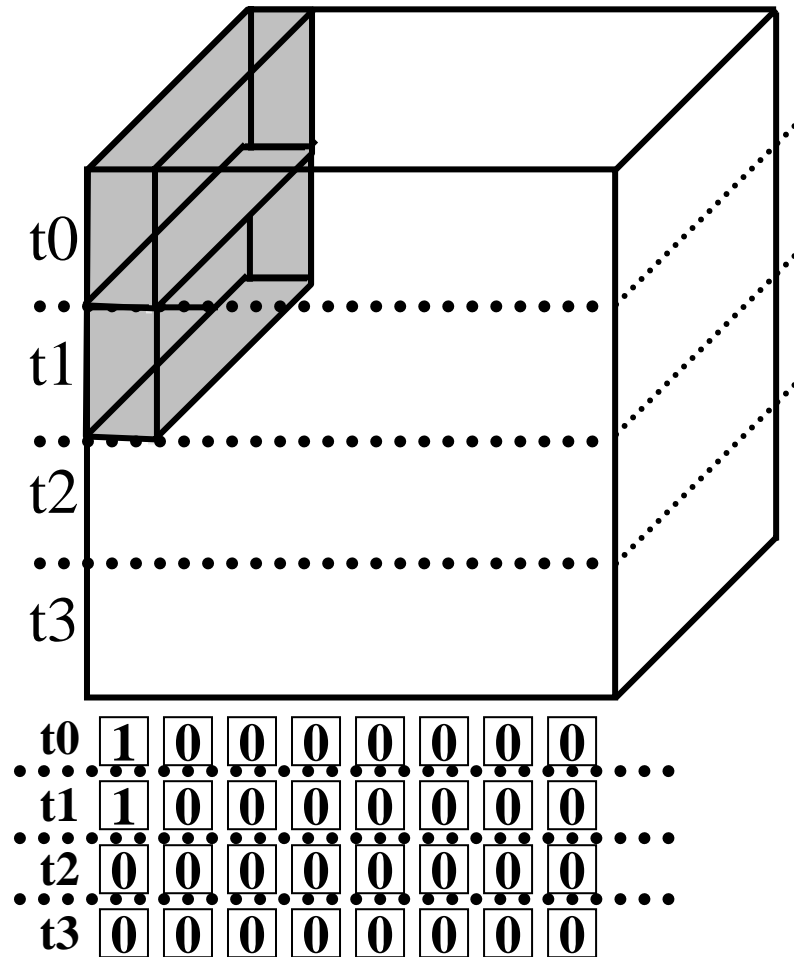




UPPSALA
UNIVERSITET

Uppsala University

Parallel Natural-order G-S in 3D

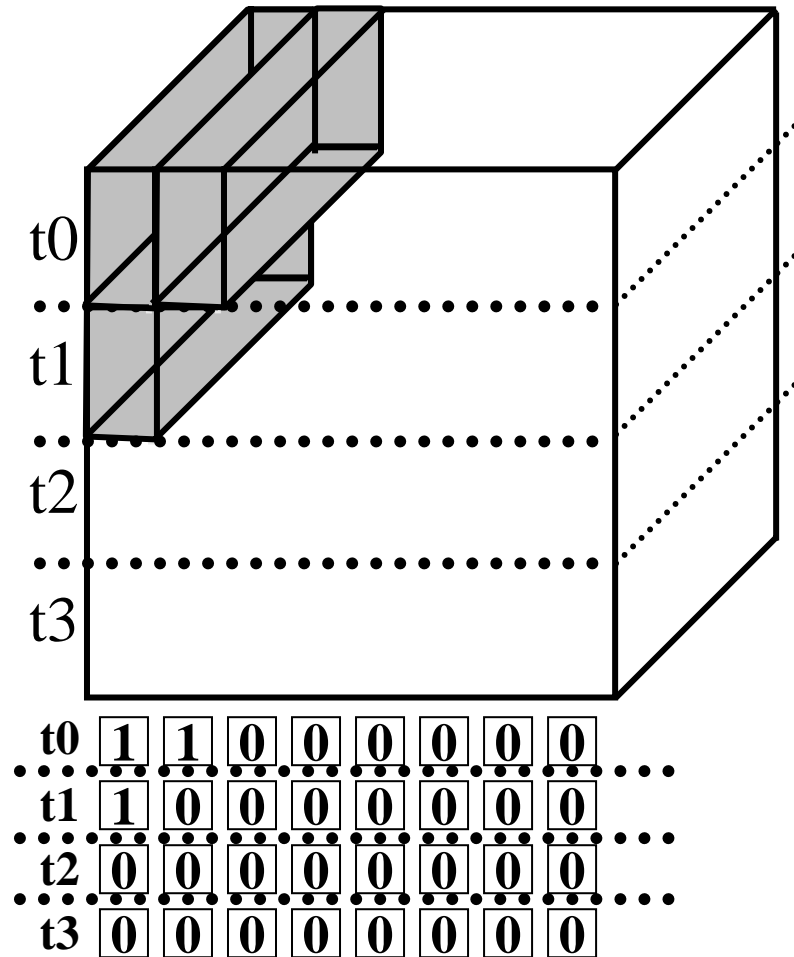




UPPSALA
UNIVERSITET

Uppsala University

Parallel Natural-order G-S in 3D

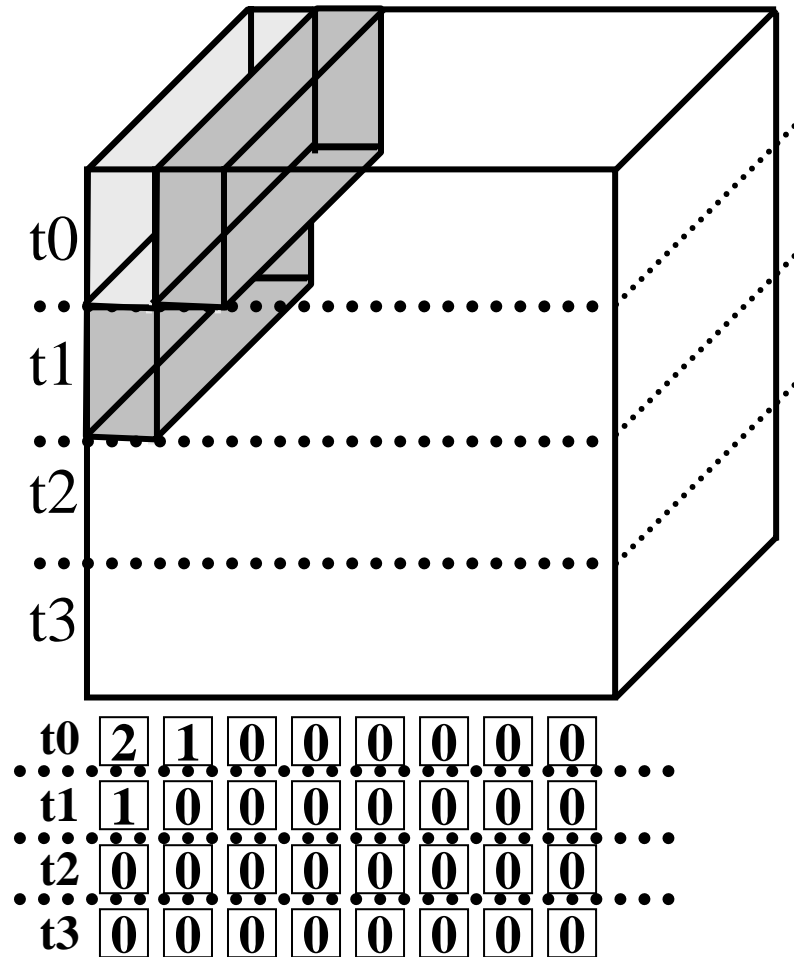




UPPSALA
UNIVERSITET

Uppsala University

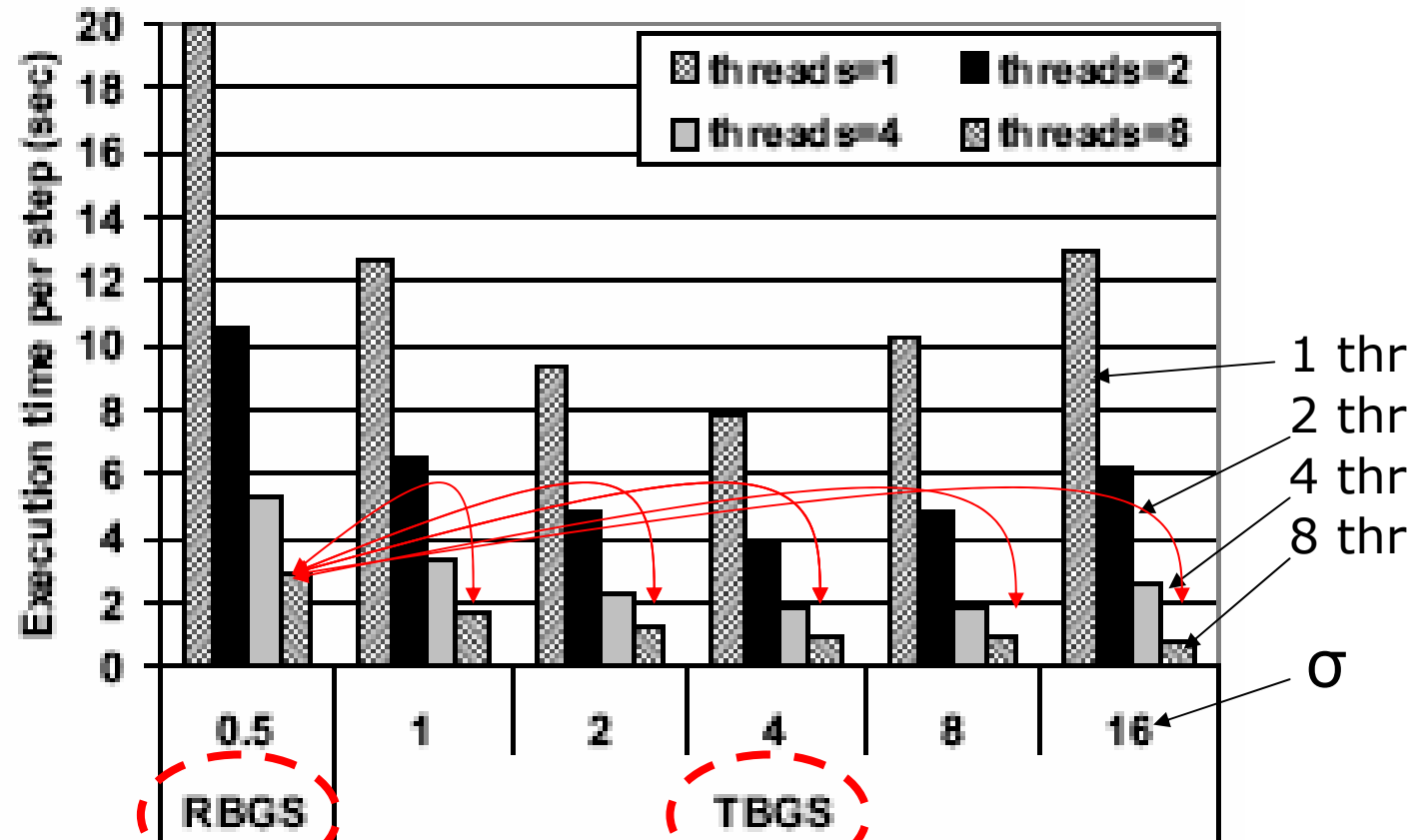
Parallel Natural-order G-S in 3D





Parallel Execution Time

(Sun Fire 15k, 1.5 MHz US4+, \$=p16kB/s2MB/s32MB)



RedBlack

(c) N = 513

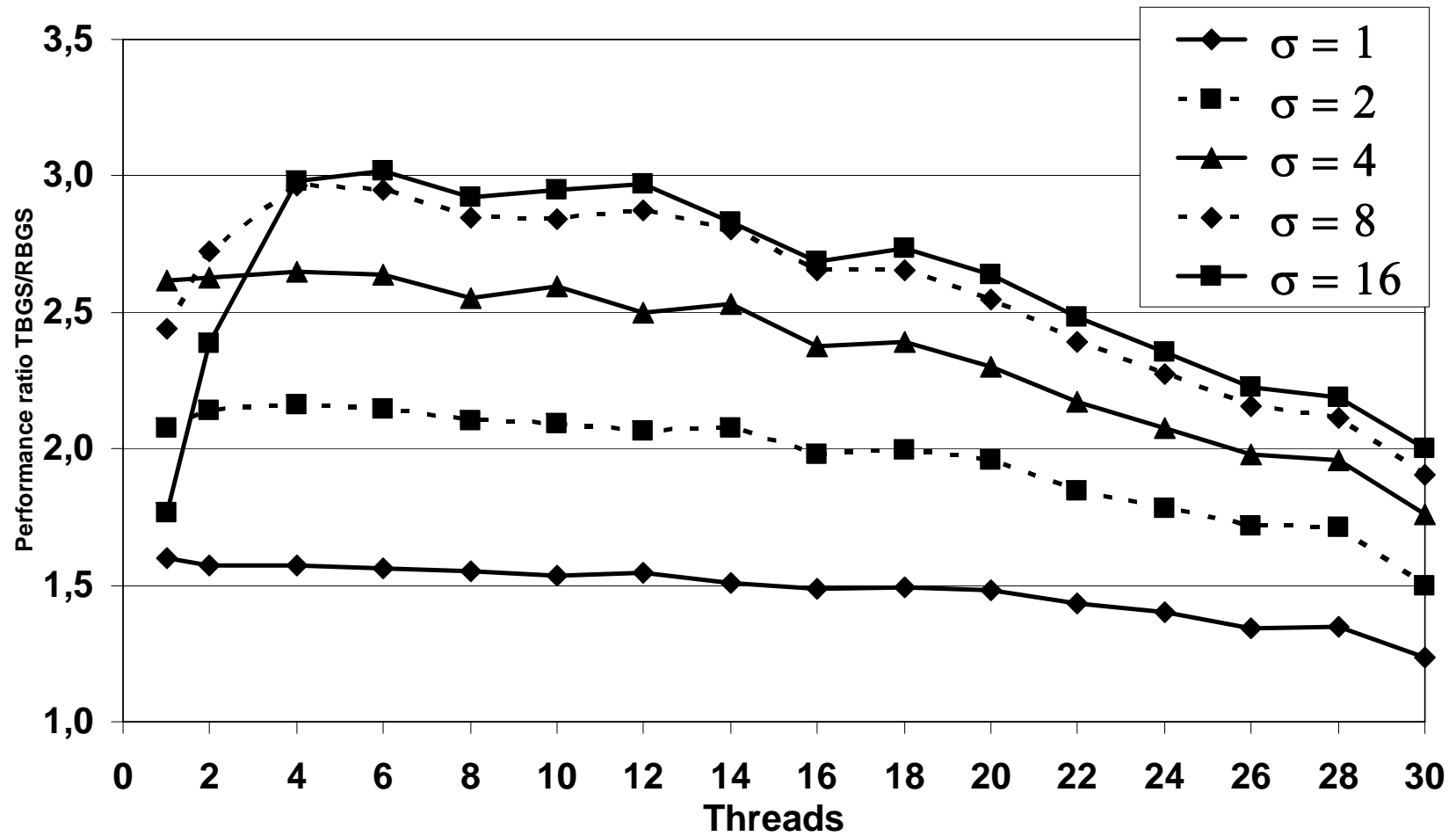
Temp.Blocked GS



UPPSALA
UNIVERSITET

Uppsala University

Performance comparison with Red-Black Sun E15 K (SMP, Single-core CPUs), N=257

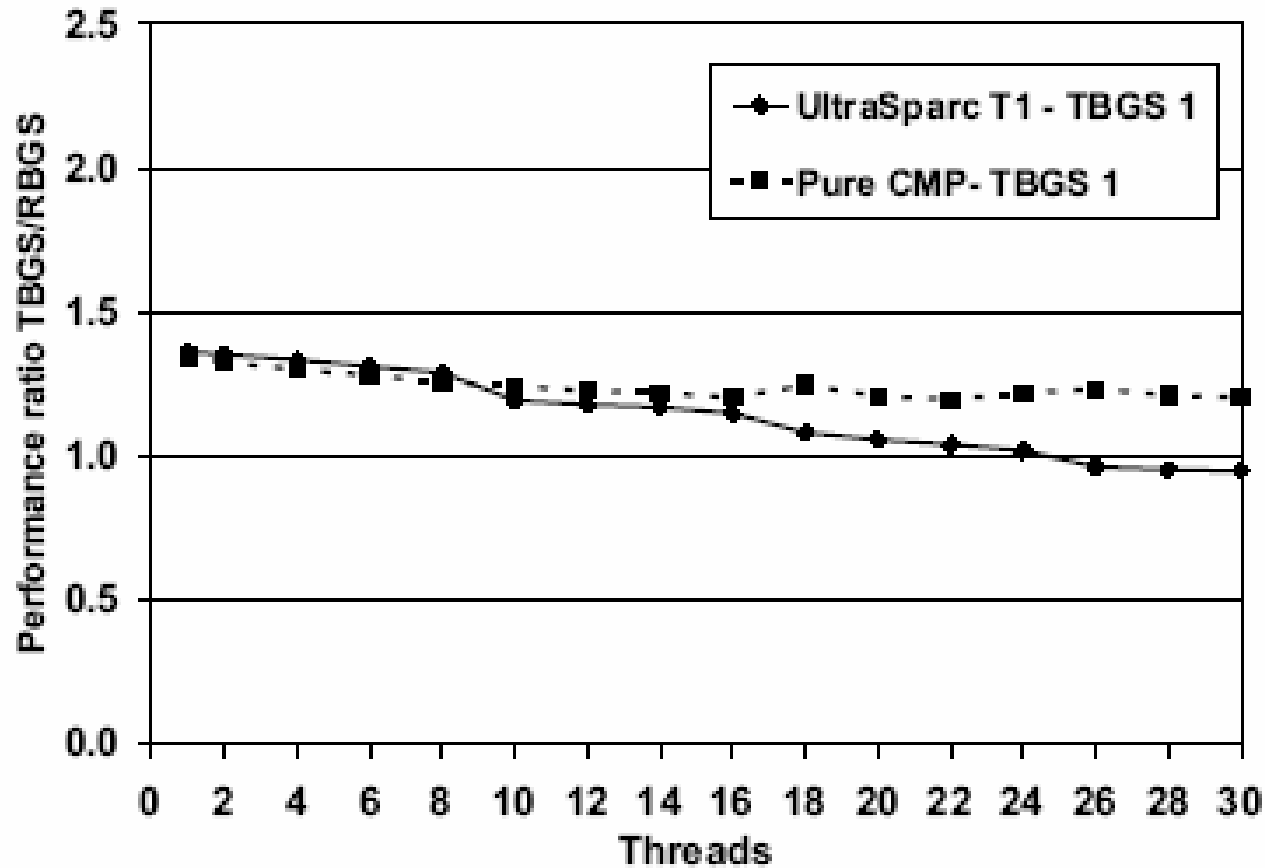




UPPSALA
UNIVERSITET

Uppsala University

Performance comparison with Red-Black Sun T1 (Niagara 1) and simulated pure CMP, N=129





Gauss-Seidel Smoother in Multigrid

- Algorithm shown so far is rarely used alone.
- G-S as a Smoother in multigrid
 - ✱ Iterative algorithm for 3D Poisson eq.
 - ✱ More efficient smoother cuts #iterations

	γ	1	2	3	4	5	6	7
RBGS	N=129	11	8	7	7	6	6	6
	N=257	11	8	7	7	6	6	6
	N=513	12	8	7	7	6	6	6
TBGS	N=129	13	9	7	7	6	6	6
	N=257	13	9	7	7	6	6	6
	N=513	13	9	7	7	6	6	6

Table 3. Number of required multigrid v-cycles to reach convergence for different values of γ .



Execution time

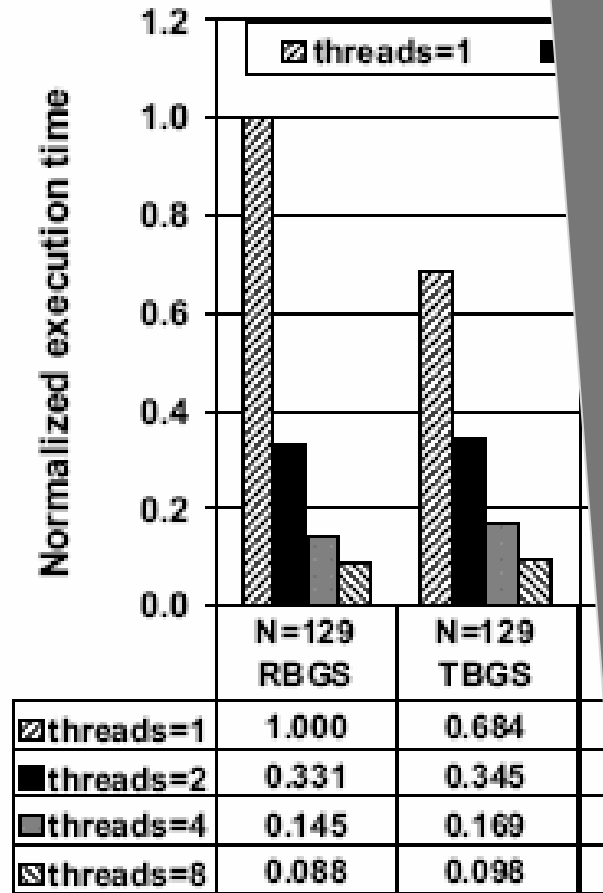


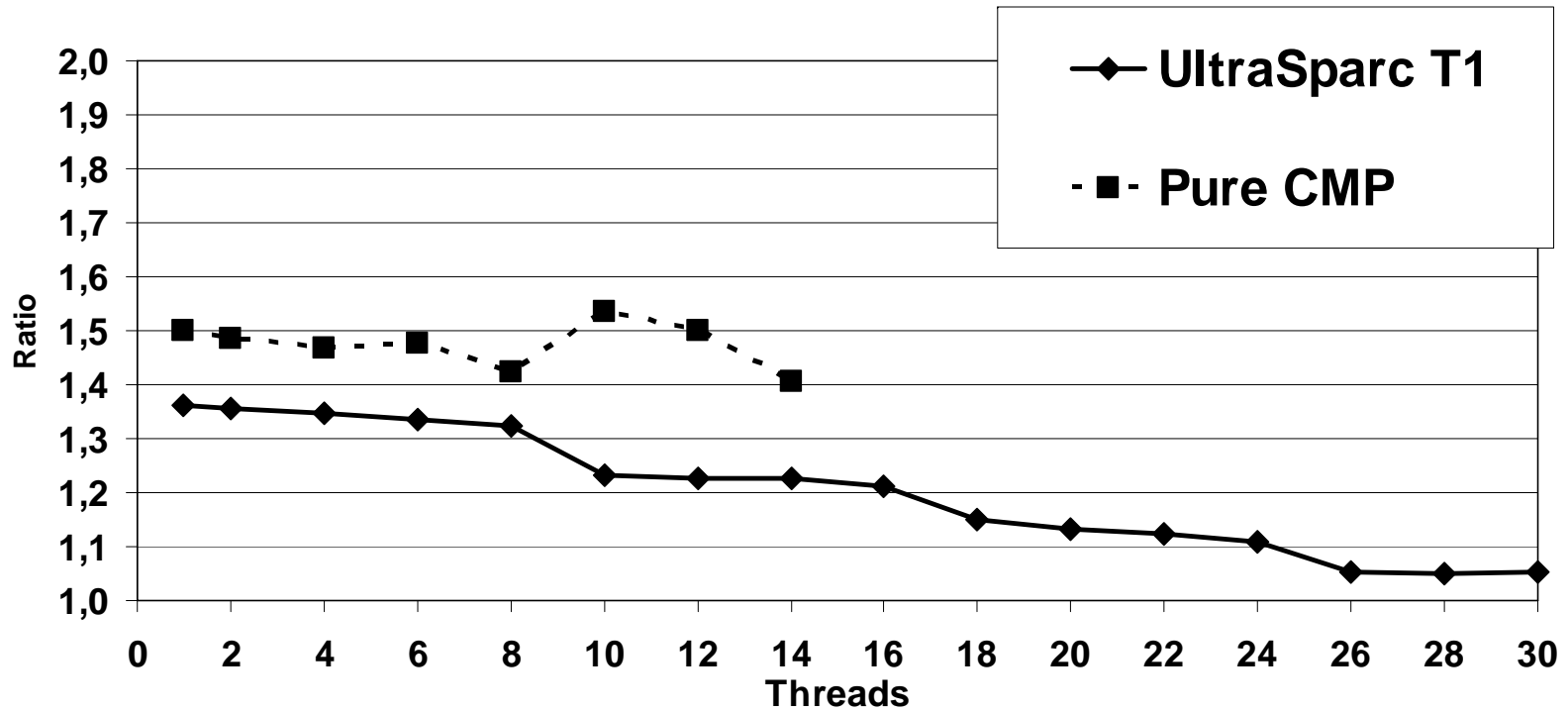
Figure 10. Execution time of the RBGS- and TBGS-smoothers. The execution time is normalized to the RBGS multigrid solver.



Ratio on US T1 & Pure CMP, N=129

$$\sigma = 1$$

Performance comparison with Red-Black



Multigrid Performance comparison with Red-Black (Sun E15 K)

Fastest Red-Black vs. fastest TB Natural

threads	N=129	N=257	N=513
1	1.46	1.57	1.55
2	0.96	1.59	1.58
4	0.86	1.60	1.66
8	0.90	1.62	1.63

Table 4. Relative speedup of the multigrid solver with TBGS smoothing compared to the RBGS-multigrid solver.

More details: See D. Wallin, H. Löf, E. Hagersten, S. Holmgren, *Multigrid and Gauss-Seidel smoothers revisited: Parallelization on chip multiprocessors*, Proc. of the 20th ACM International Conference on Supercomputing (ICS 2006), pp 145-155.



UPPSALA
UNIVERSITET

Uppsala University

Example: Genetics on a Cluster of Multicore Processors



UPPSALA
UNIVERSITET

Uppsala University

Where in the Genome are the Important Genes?



Quantitative Trait Loci (QTL) analysis may give (part of) the answer.



Quantitative Trait Loci

QTL = A position in the genome affecting a quantitative trait

Quantitative trait = A trait that is measured on a continuous scale (e.g. Body weight, blood pressure, crop yield, ...)

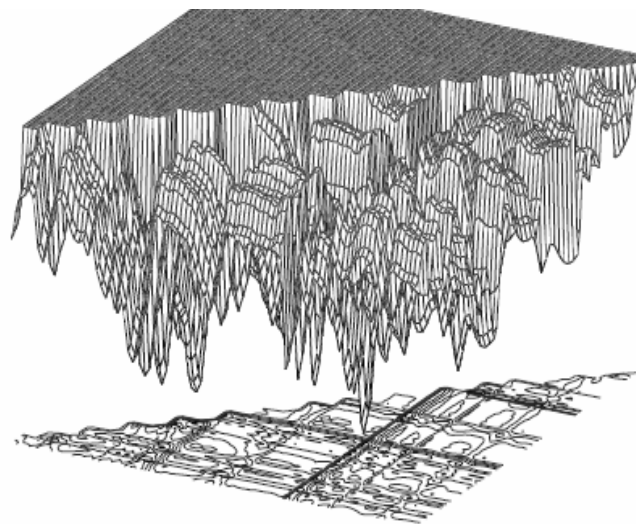
QTL model = Statistical model relating genotypes (genetic composition) to phenotypes (trait values) for an experimental population.



Mapping of d QTL

Fit model to data at various combinations of d positions. The set of QTL positions giving smallest $RSS(x)$ is the most likely to be true.

D-dimensional global optimization problem





The global search

We use a modified version of the DIRECT global optimization method (branch-without-bound).

Parallel version using a naive "domain decomposition" scheme based on chromosomes:

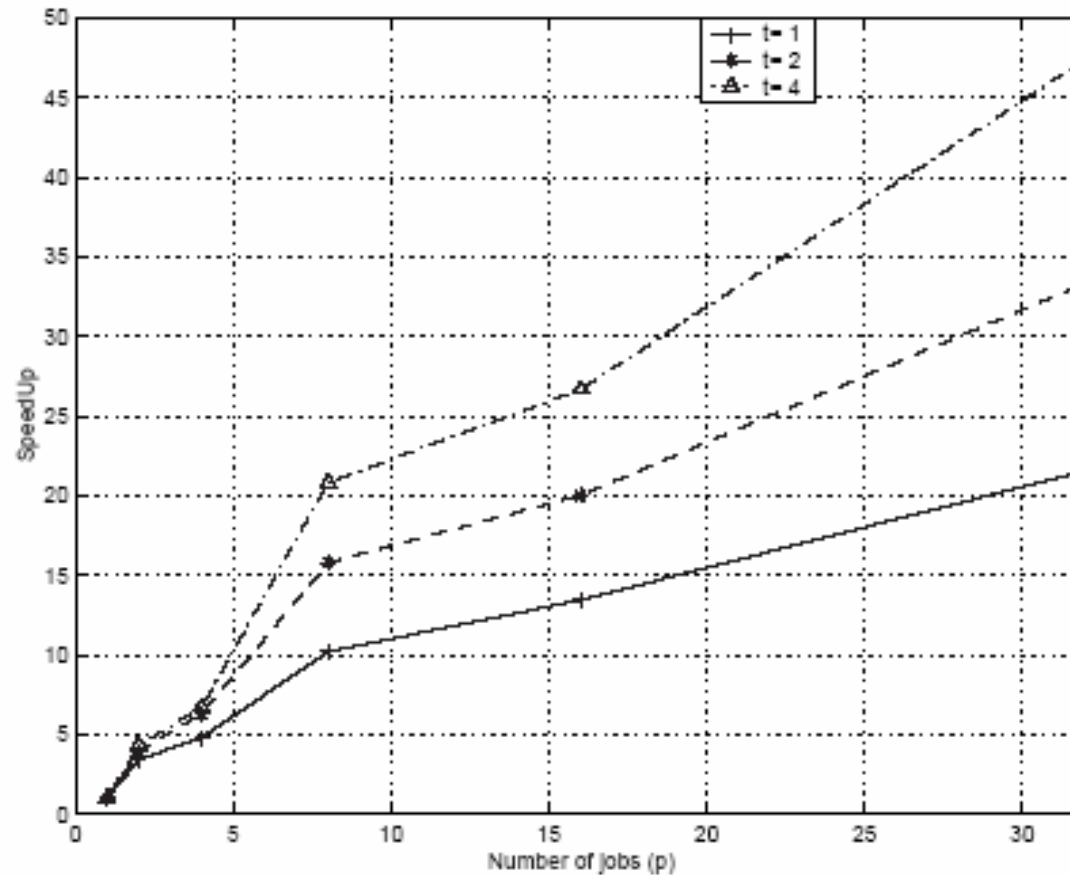
- Submit one serial job for each partition
- Compare the local minima

Earlier: Execution on the SweGrid system

NOW: Including OpenMP parallelization within each job.

Results on new SweGrid Cluster (Nodes with 2 dual-core Opteron)

Fig. 6. SpeedUp for data partitioning and multithreading - 3D





UPPSALA
UNIVERSITET

Uppsala University

Results on Niagara T2

*See [Grids and Clusters with Multi-Core Nodes: A Genetics Application Perspective](#),
Mahen Jayawardena, Henrik Löf and Sverker
Holmgren, to be published soon ...*



Conclusions

- The Multicore Era is here
- Impact on CSE algorithms? Multicore processors change the rules for:
 - ✱ Cost of parallelism
 - ✱ Cache capacity per thread
 - ✱ Memory bandwidth per thread
 - ✱ Cost of thread communication
- For example, today's algorithms are based on the assumption that communication is expensive. In the future, the cost of communication may sometimes be "zero"
- Should we redesign algorithms to face these changes?
- Preliminary results: Yes, we should!