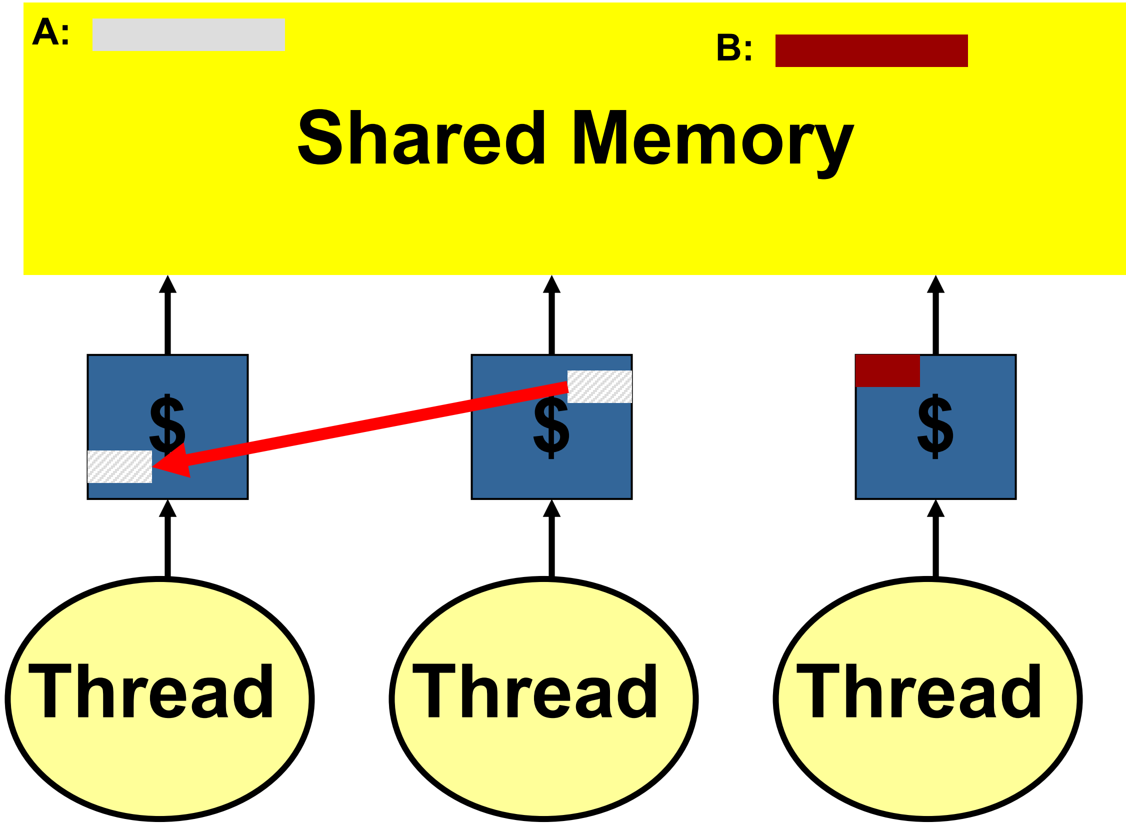
Multicore
from an Application's Perspective

Erik Hagersten
Uppsala Universitet

Communication in an SMP

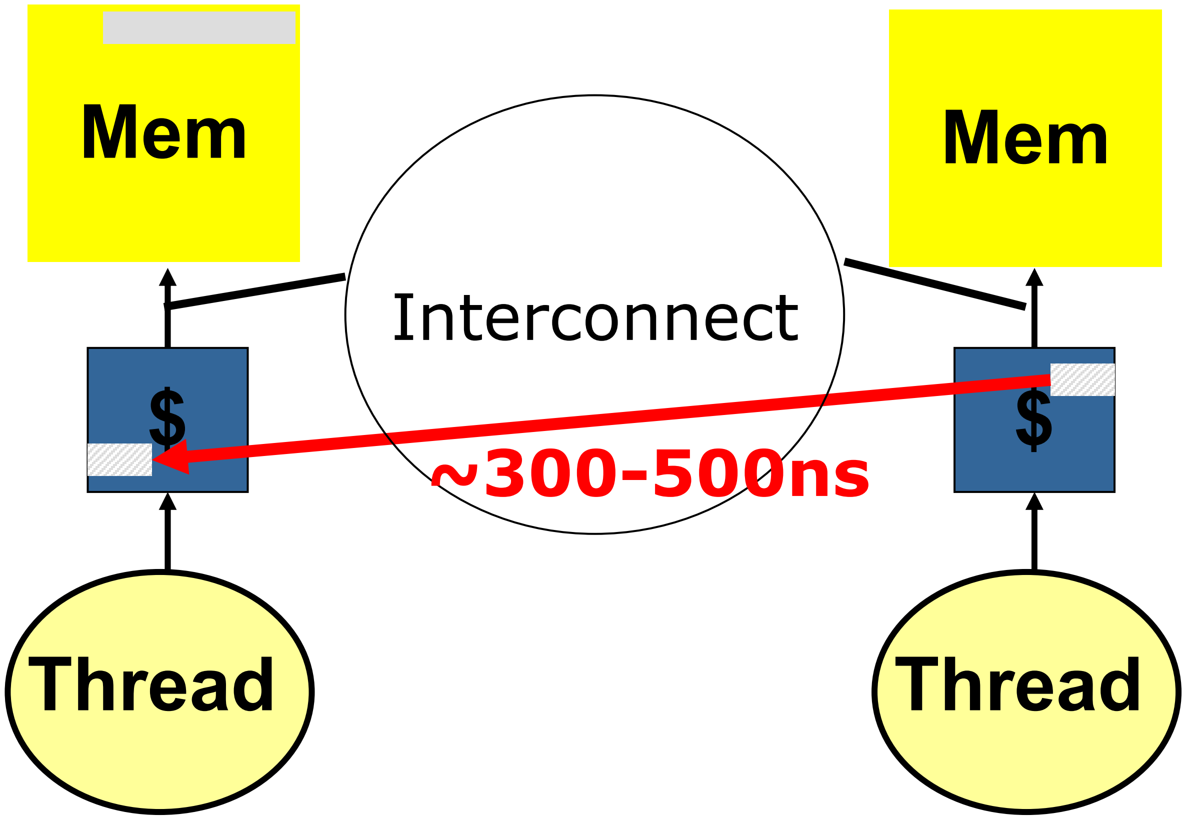


Read A
Read A
…
…
Read A

…
Read A
…
Write A

Read B
…
Read A

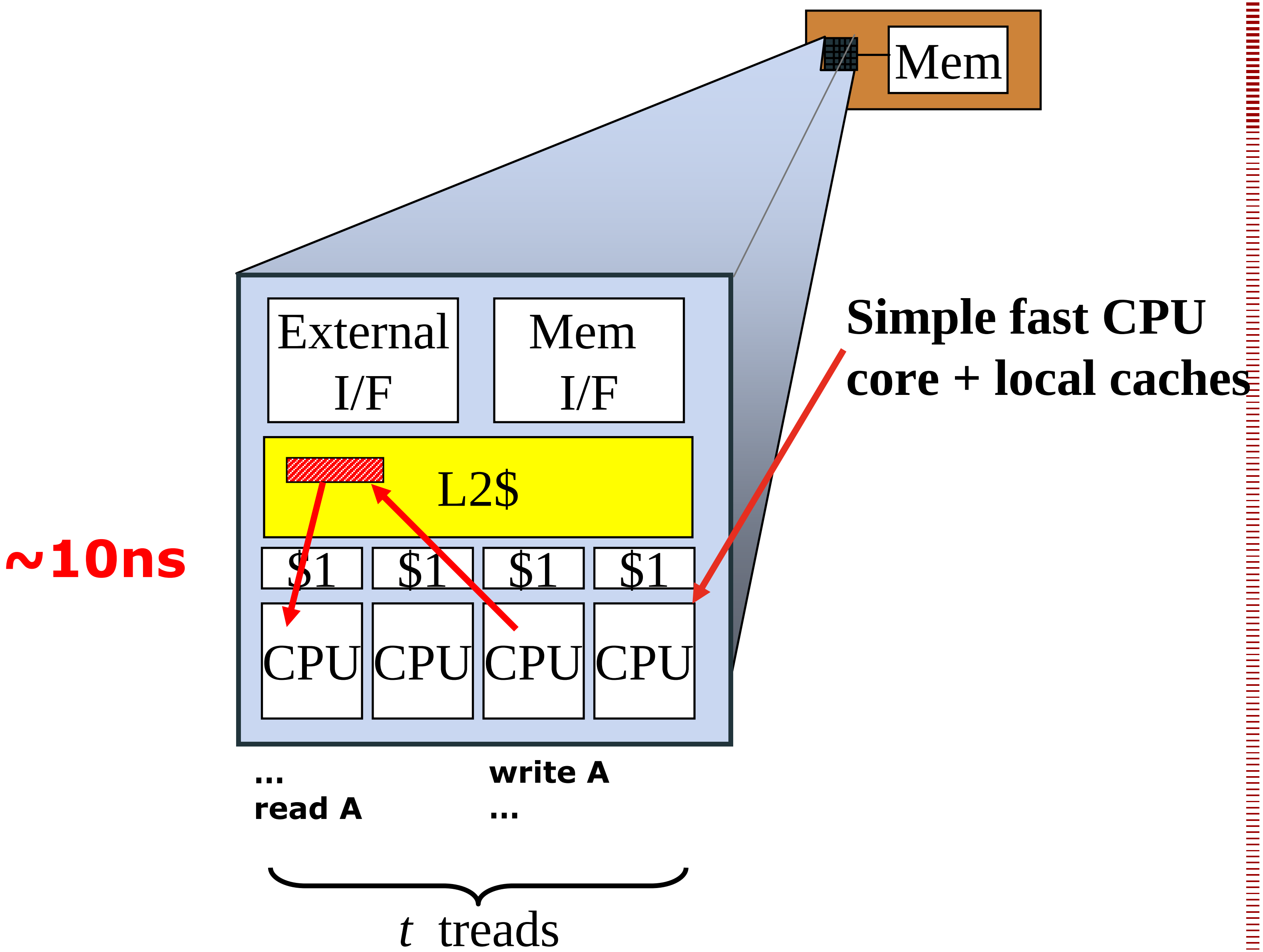
Communication in a NUMA → Wors



Read A
Read A
…
…
Read A

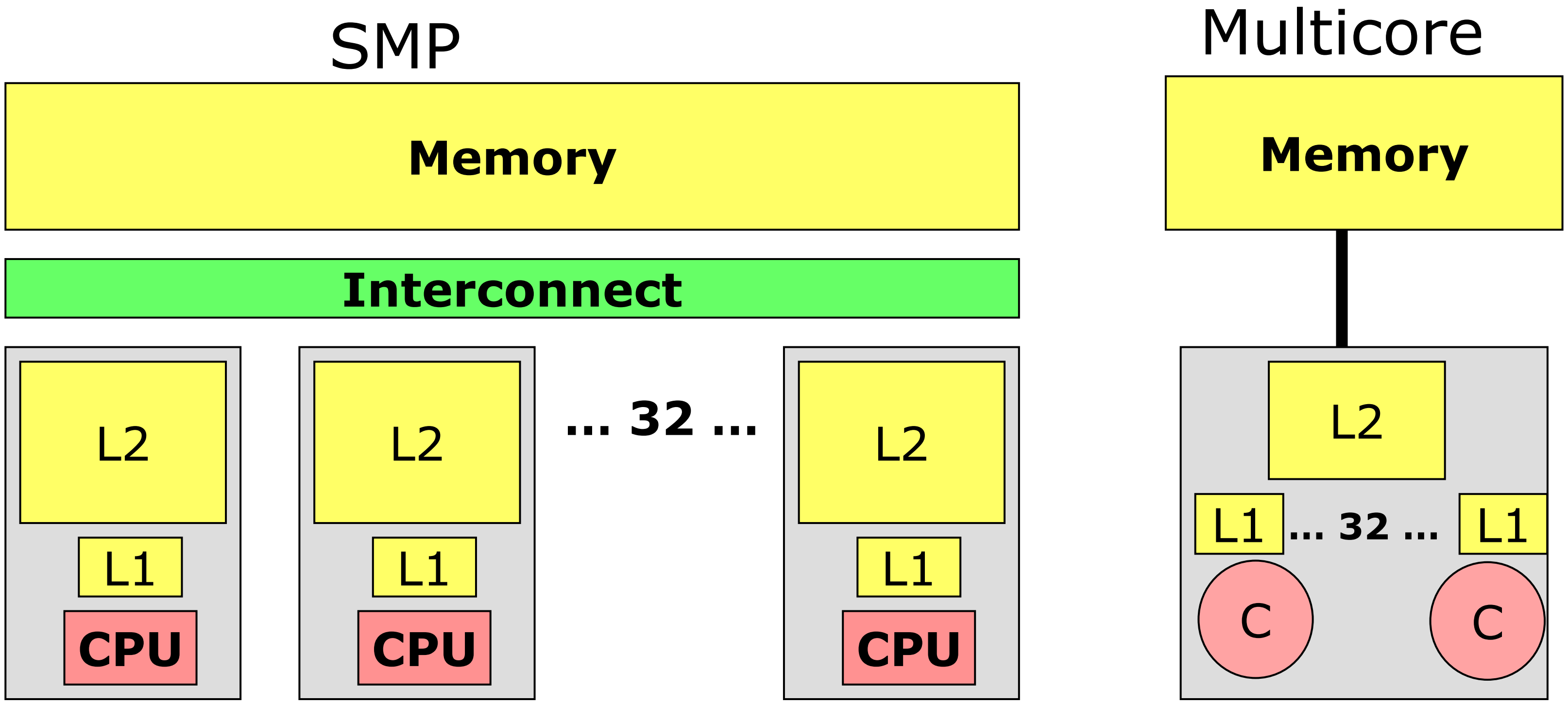
…
Read A
…
Write A

Communication in [some] Multicores



Uppsala University

Looks and Smells Like an SMP?



Well, how about:

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

→Gotta' Optimize For Locality!

UPPSALA UNIVERSITET

Uppsala University

Criteria for HPC Algorithms

- Past:
  - Minimize **communication**
  - Maximize scalability (1000s of CPUs)

- Multicores today:
  - Communication is “for free”
    [on some multicores]
  - Scalability is limited to 32 threads
  - The caches are tiny
  - Memory bandwidth is scarse

➔ Data locality is key!

(Both for Capacity and Capability Computing!)

Case Study: Gauss-Seidel on Multicores

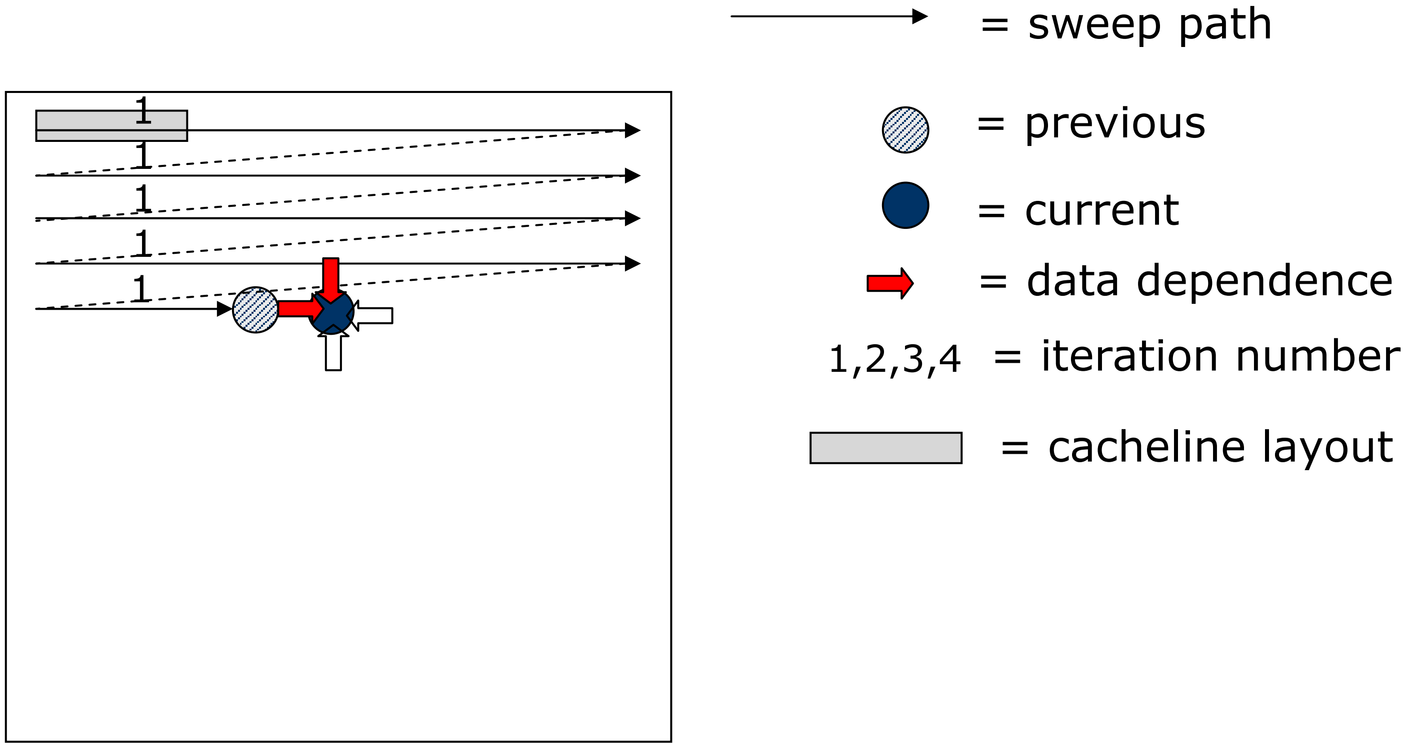
Erik Hagersten
Uppsala University
Sweden

Thanks: Dan Wallin(arch), Henrik Löf (sci comp) and Sverker Holmgren (sci comp)
From Wallin et al, ICS 2006

UPPSALA UNIVERSITET

Uppsala University

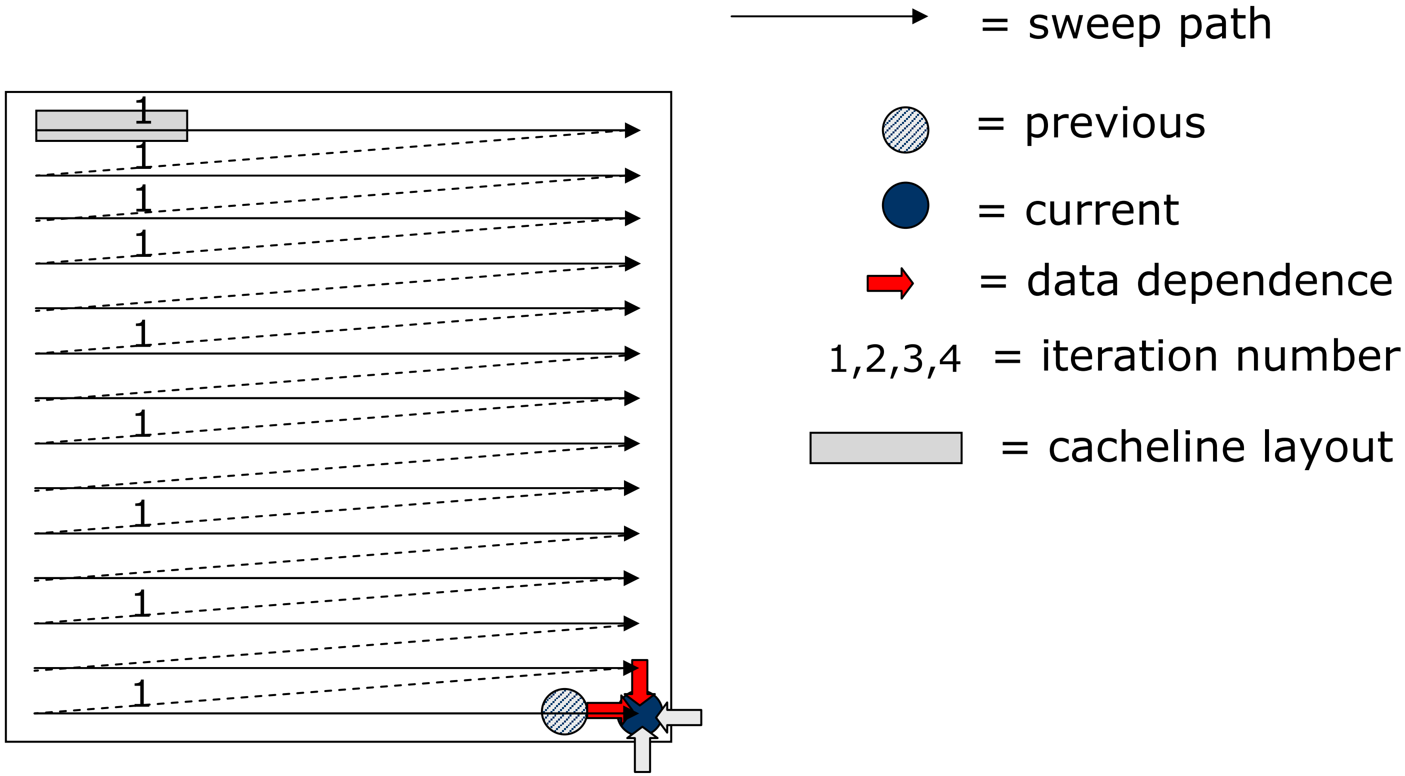
Natural Order Gauss-Seidel



UPPSALA UNIVERSITET

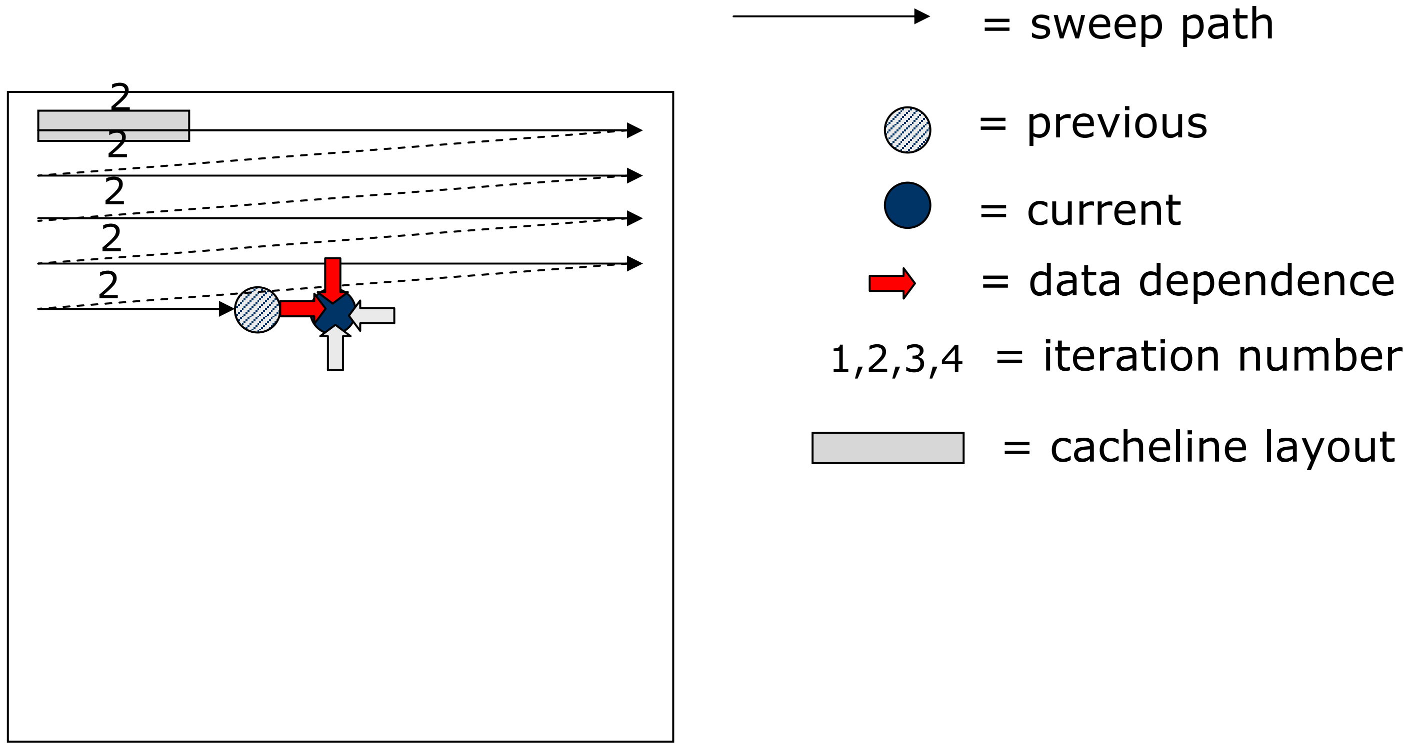
Uppsala University

Natural Order Gauss-Seidel



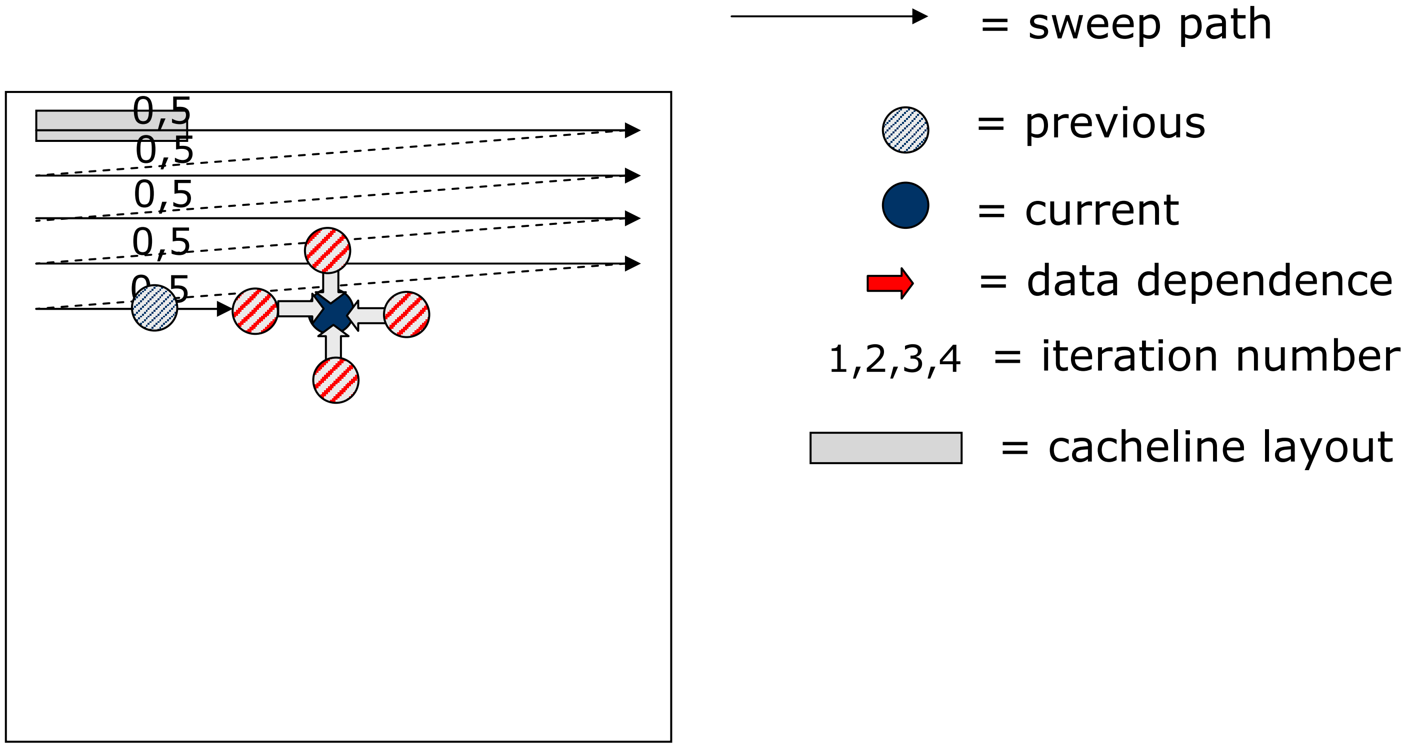
IF (convergence_test)
<done>
else
<iterate again>

Natural Order Gauss-Seidel

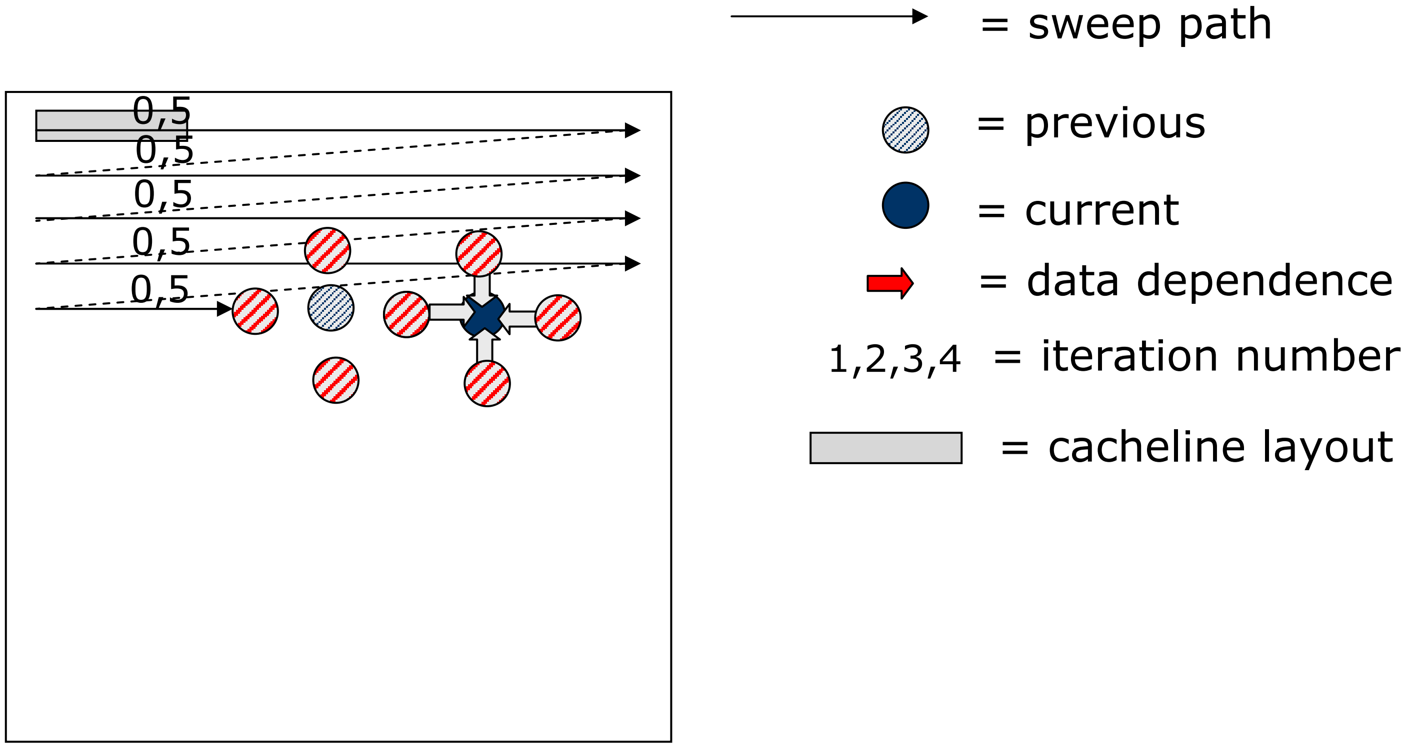


Data dependence ➔ Poor Parallelism☹

Red-Black Gauss-Seidel



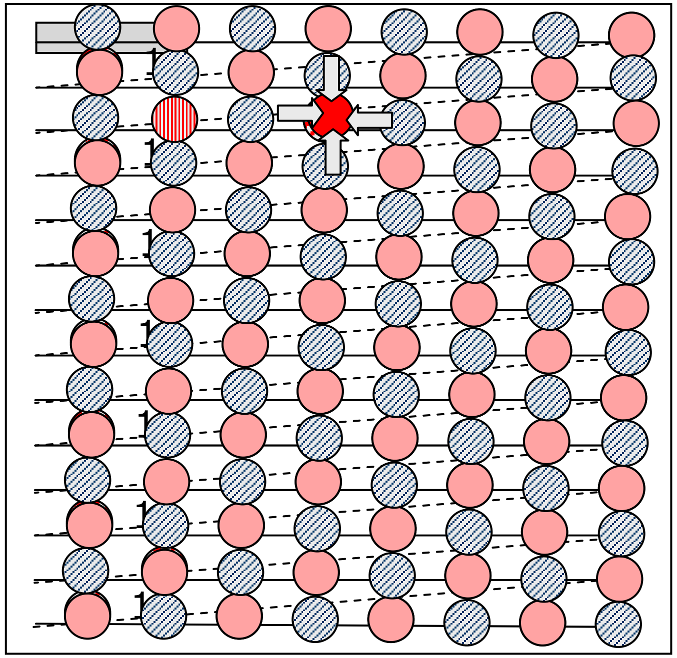
Red-Black Gauss-Seidel step 0,5: update the blacks

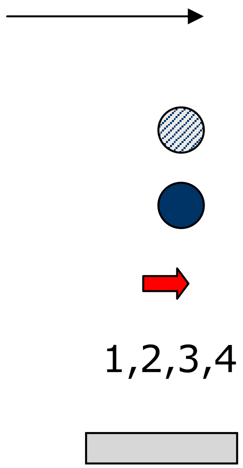


UPPSALA UNIVERSITET

Uppsala University

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

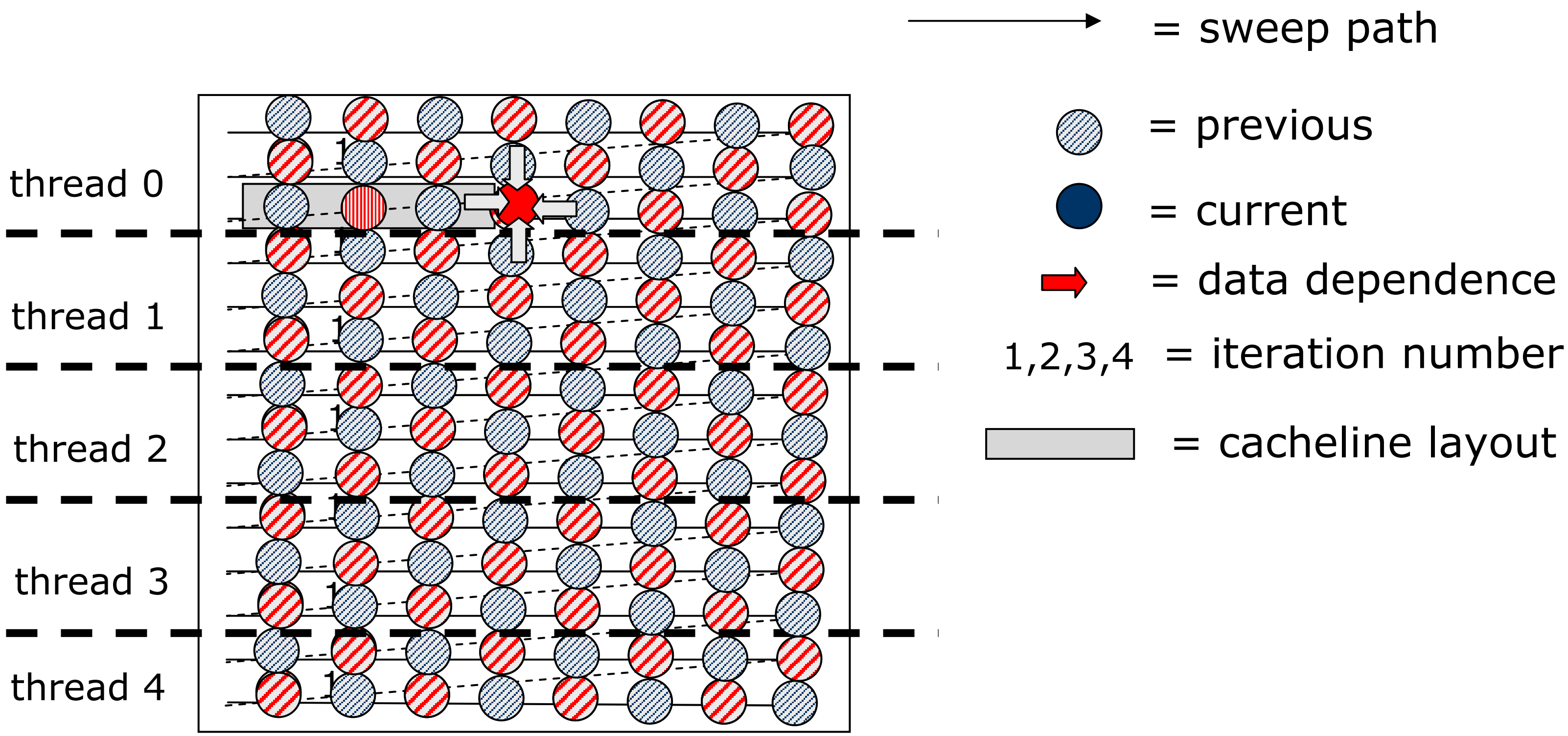




Update all blacks
<barrier>
Update all reds
<barrier>

➔ great parallelism!!!

Red-Black Gauss-Seidel Parallel version



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

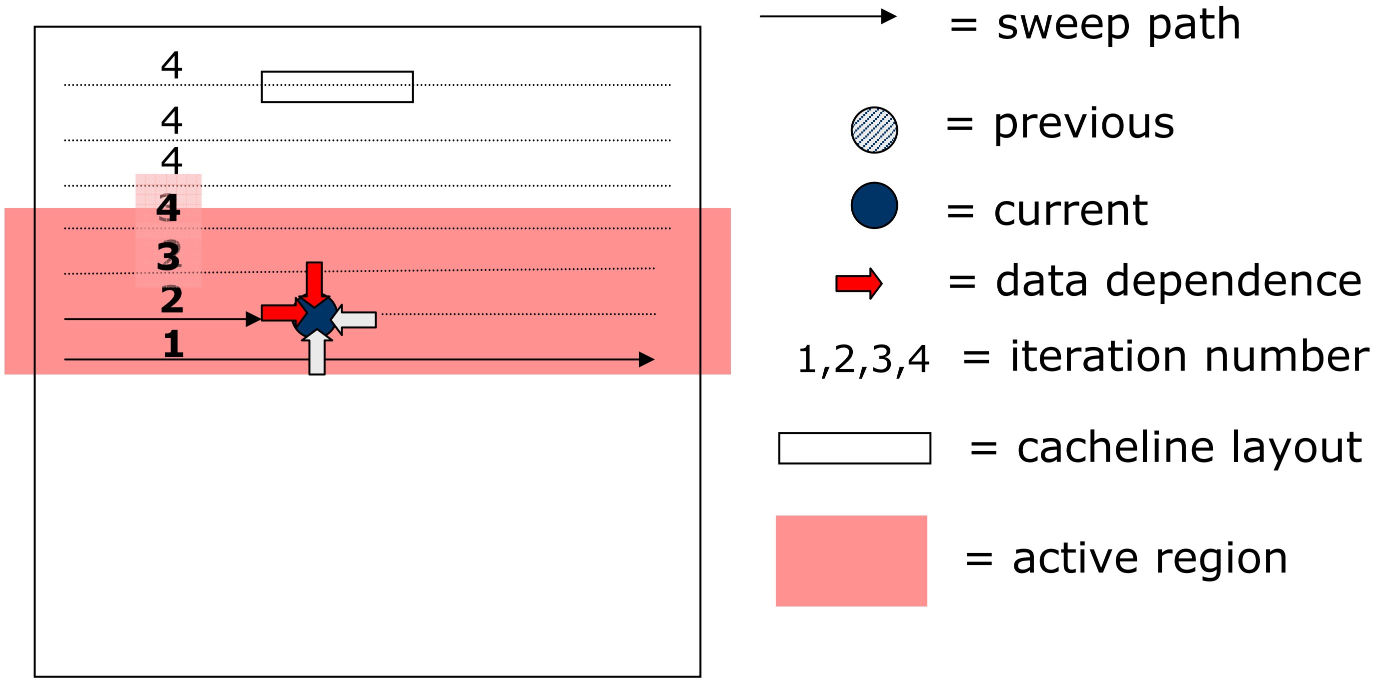
UPPSALA UNIVERSITET

Uppsala University

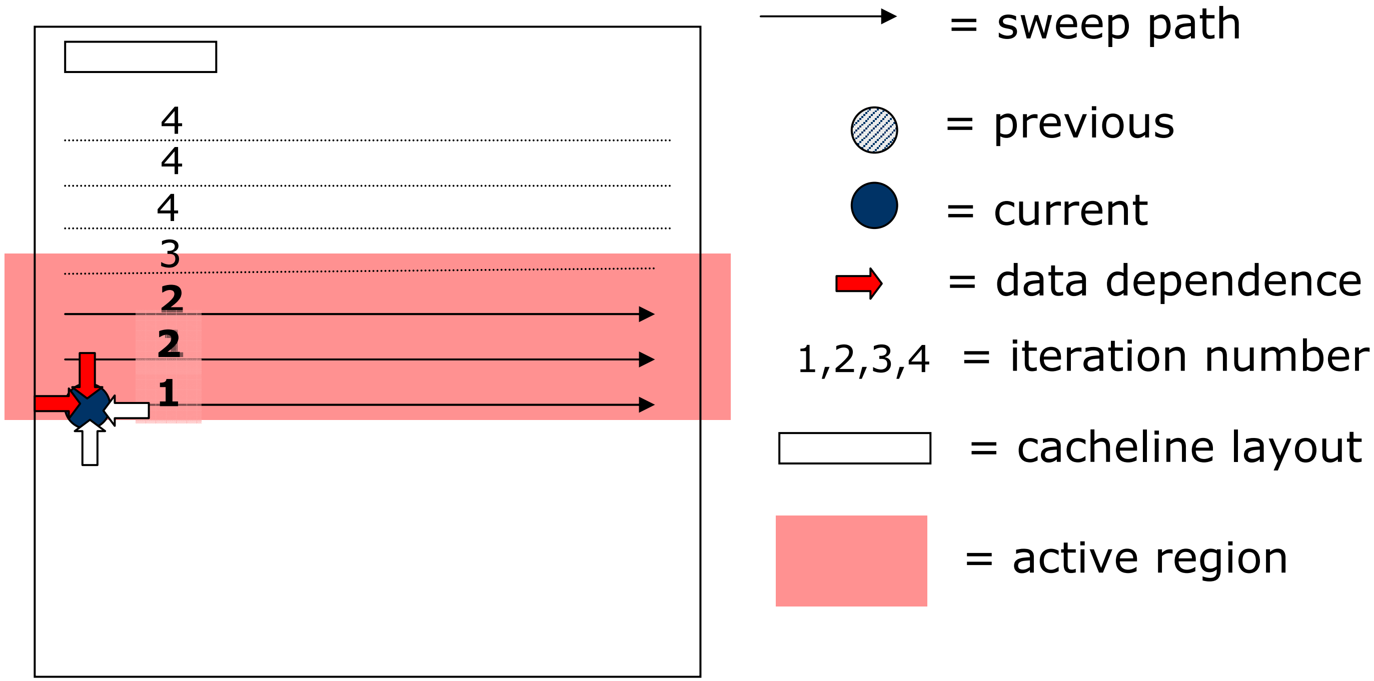
Any Drawbacks of the Red-Black?

- Poor Cache Locality of 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 even better…
  - ➔ Natural Order with Temporal Blocking ☺

G-S, temporal blocking: several iterations per sweep

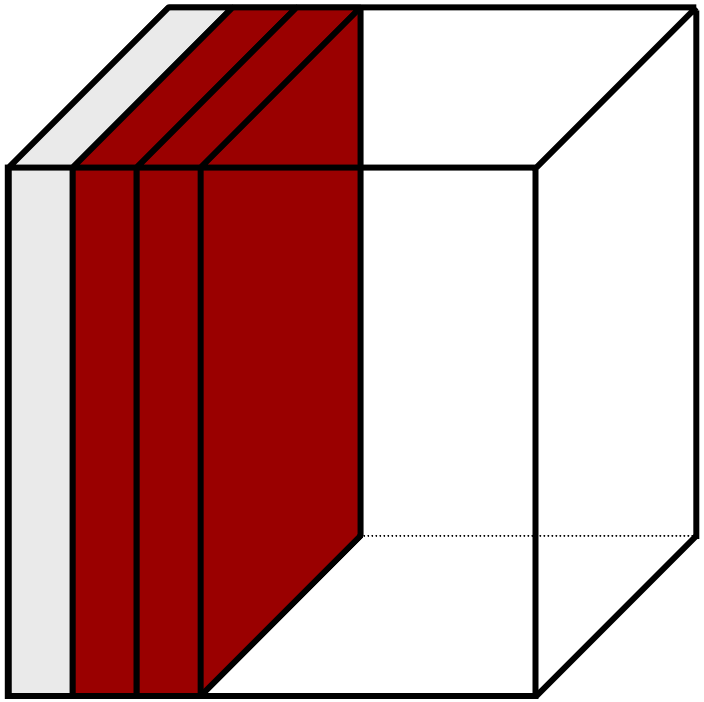


G-S, temporal blocking: several iterations per sweep

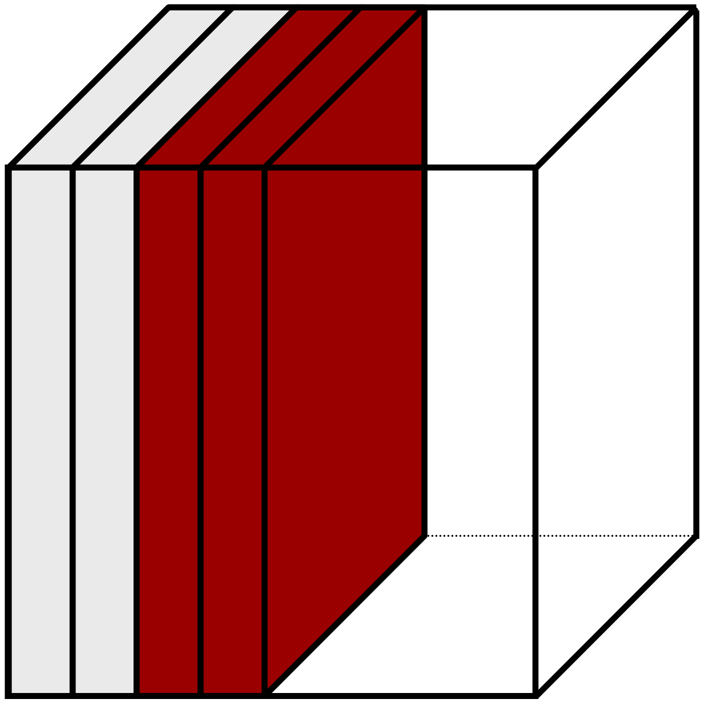


In this case: 4 iterations per "sweep". (σ = 4)
σ = 1,0 for natural order G-S
σ = 0,5 for red-black G-S

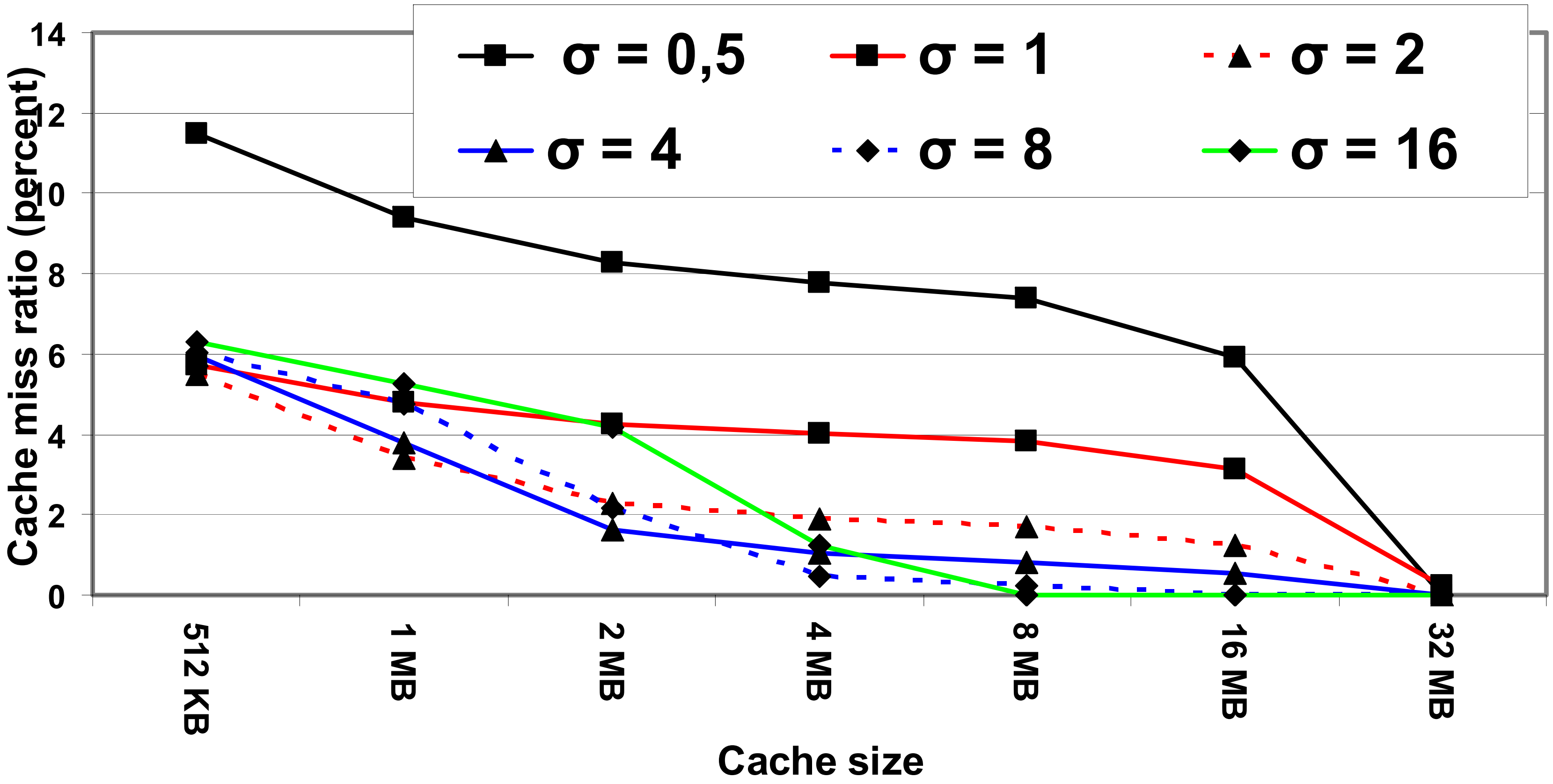
G-S 3D, σ=2



G-S 3D, σ=2

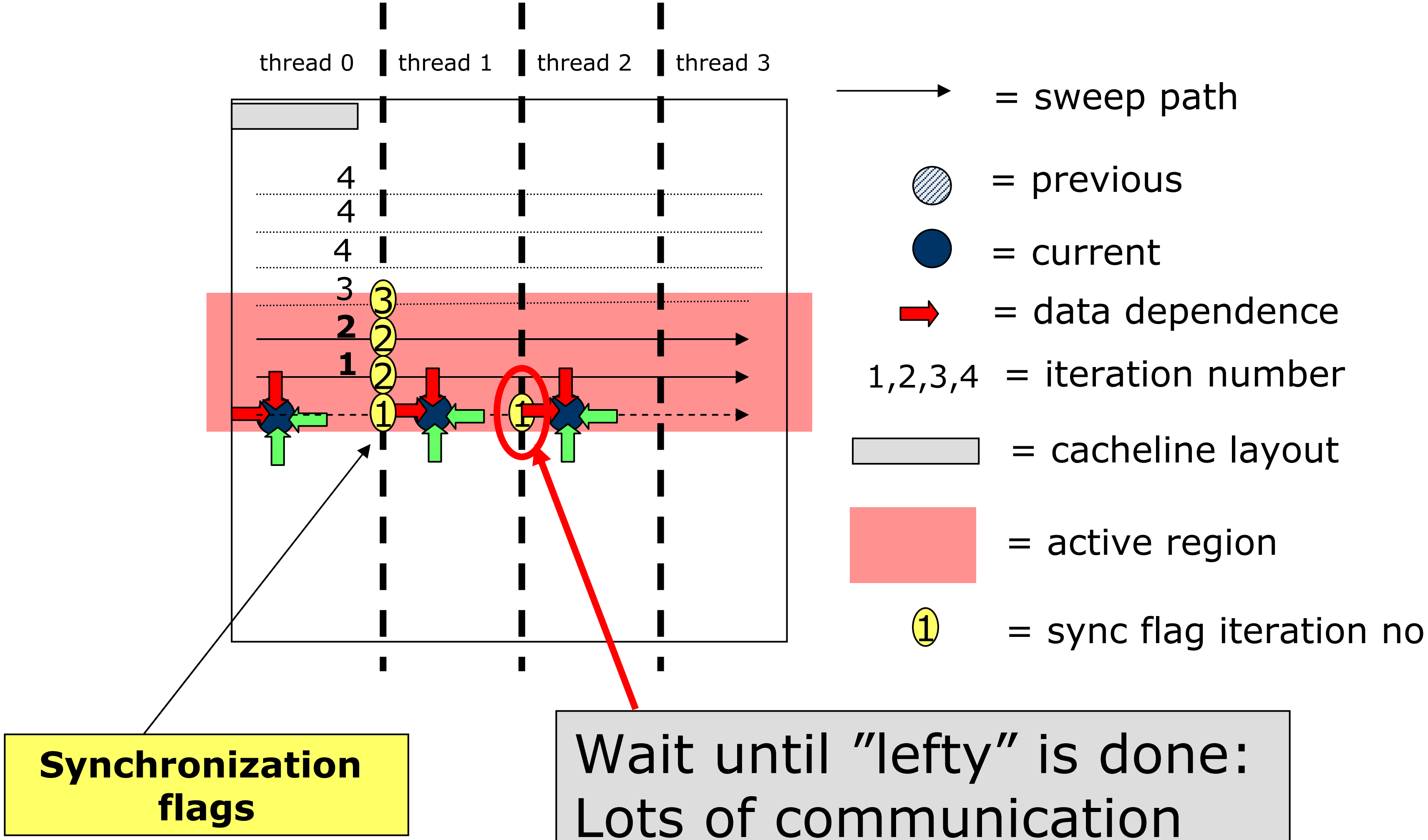


Acumem Graph, 3D N=129



Miss ratio ~ Memory bandwidth

Parallel G-S, temporal blocked

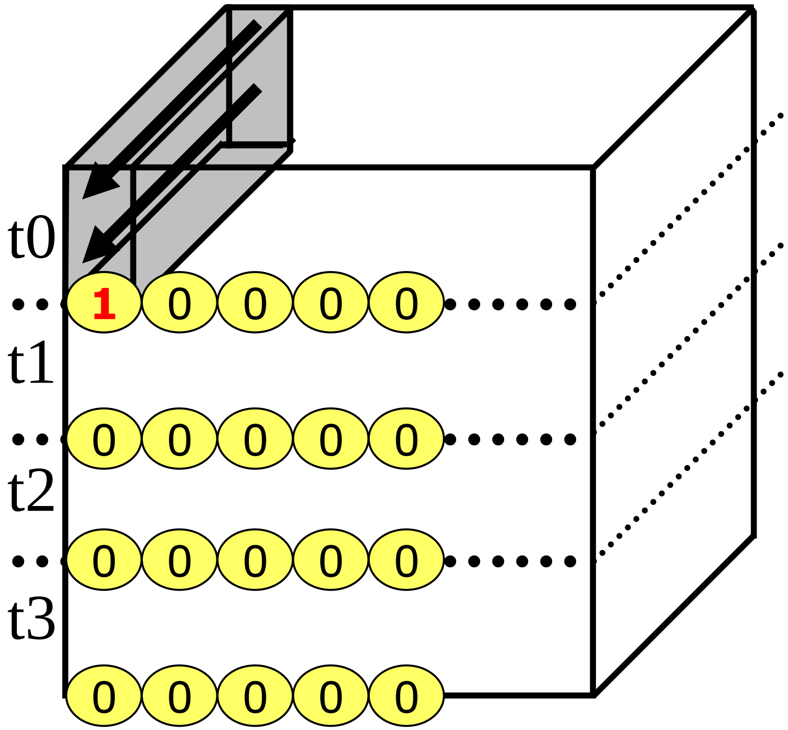


Wait until "lefty" is done:
Lots of communication
- Producer/Consumer flag
- Sharing of data values

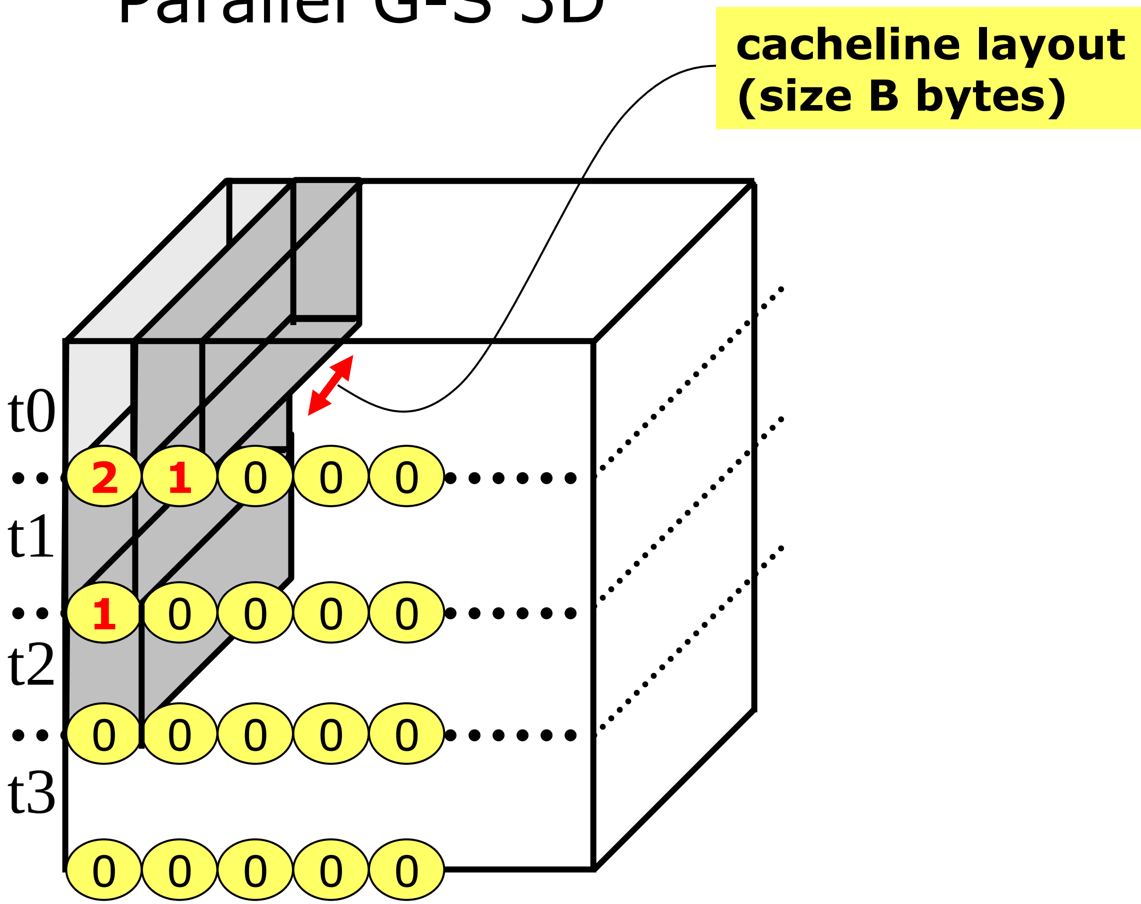
Uppsala University

UPPSALA UNIVERSITET

Parallel G-S 3D



Parallel G-S 3D



Stratup cost = (#threads-1)/(Nσ)

Communication:

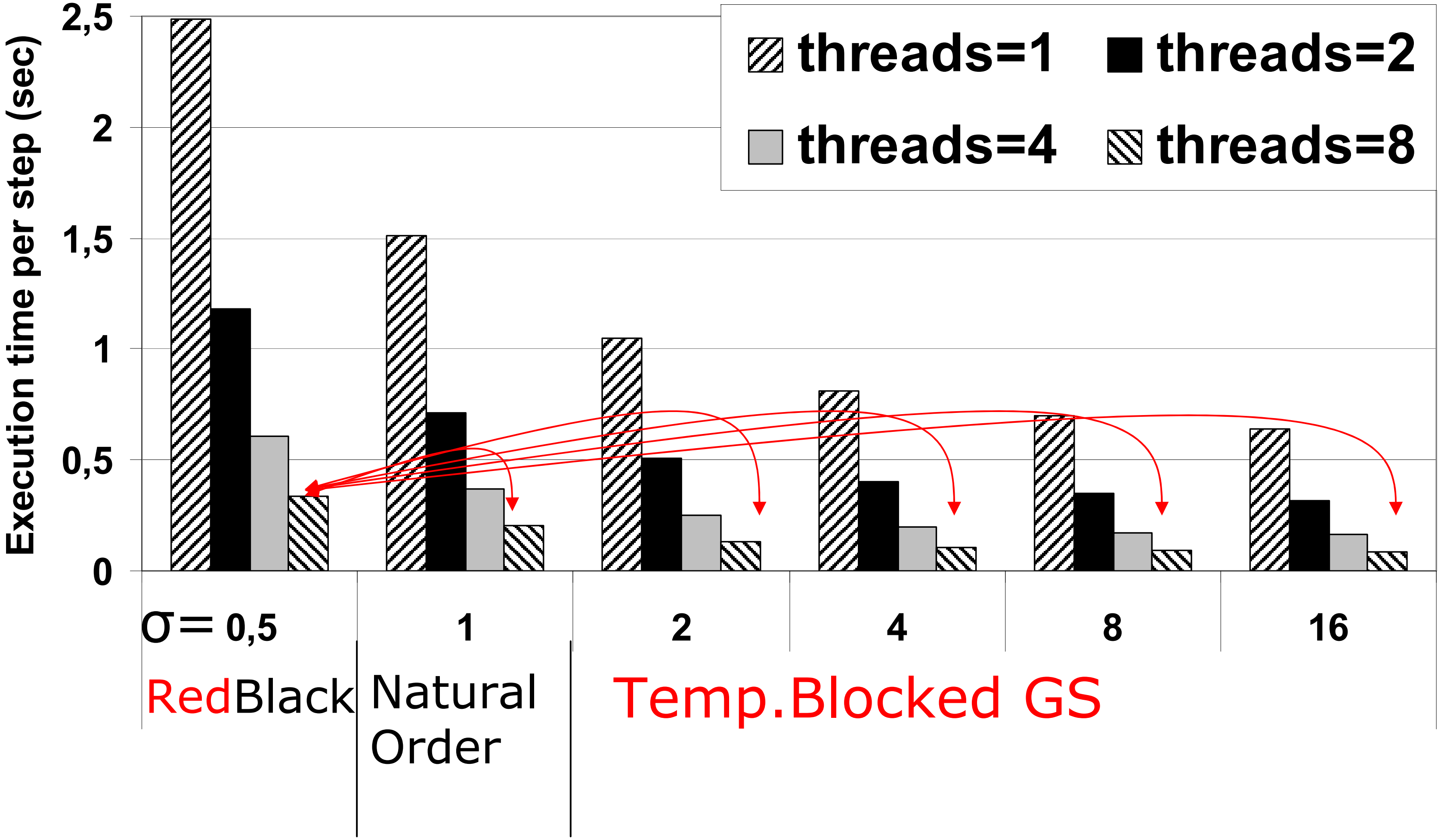
•one flag synchronization per N^2/#threads ops

•one communication miss per B*N/#threads bytes

Uppsala University

UPPSALA UNIVERSITET

Parallel Executiontime

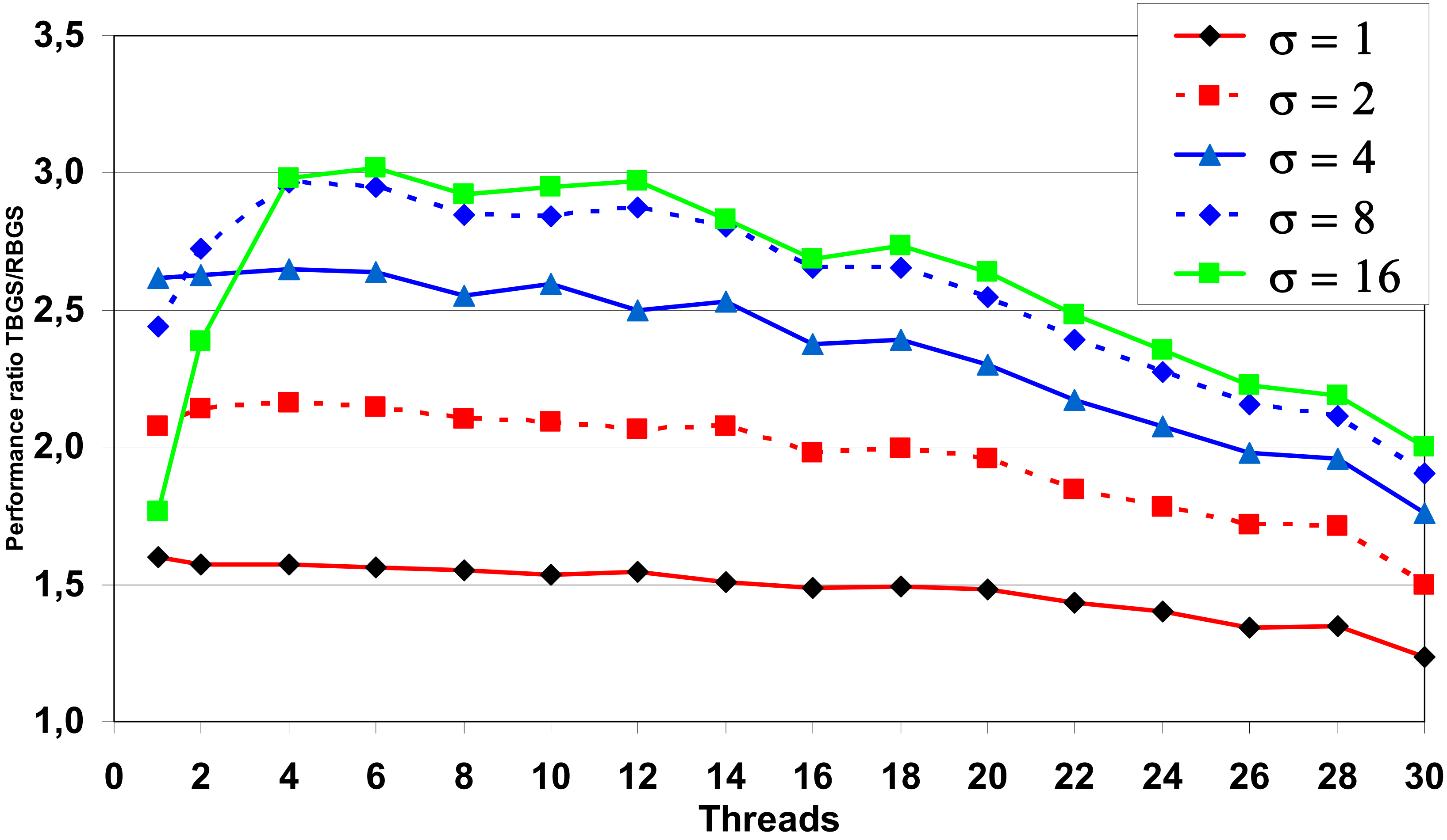


Uppsala University

UPPSALA UNIVERSITET

Performance comparison with Red-Black σ = 0,5 N = 257, 32 threads

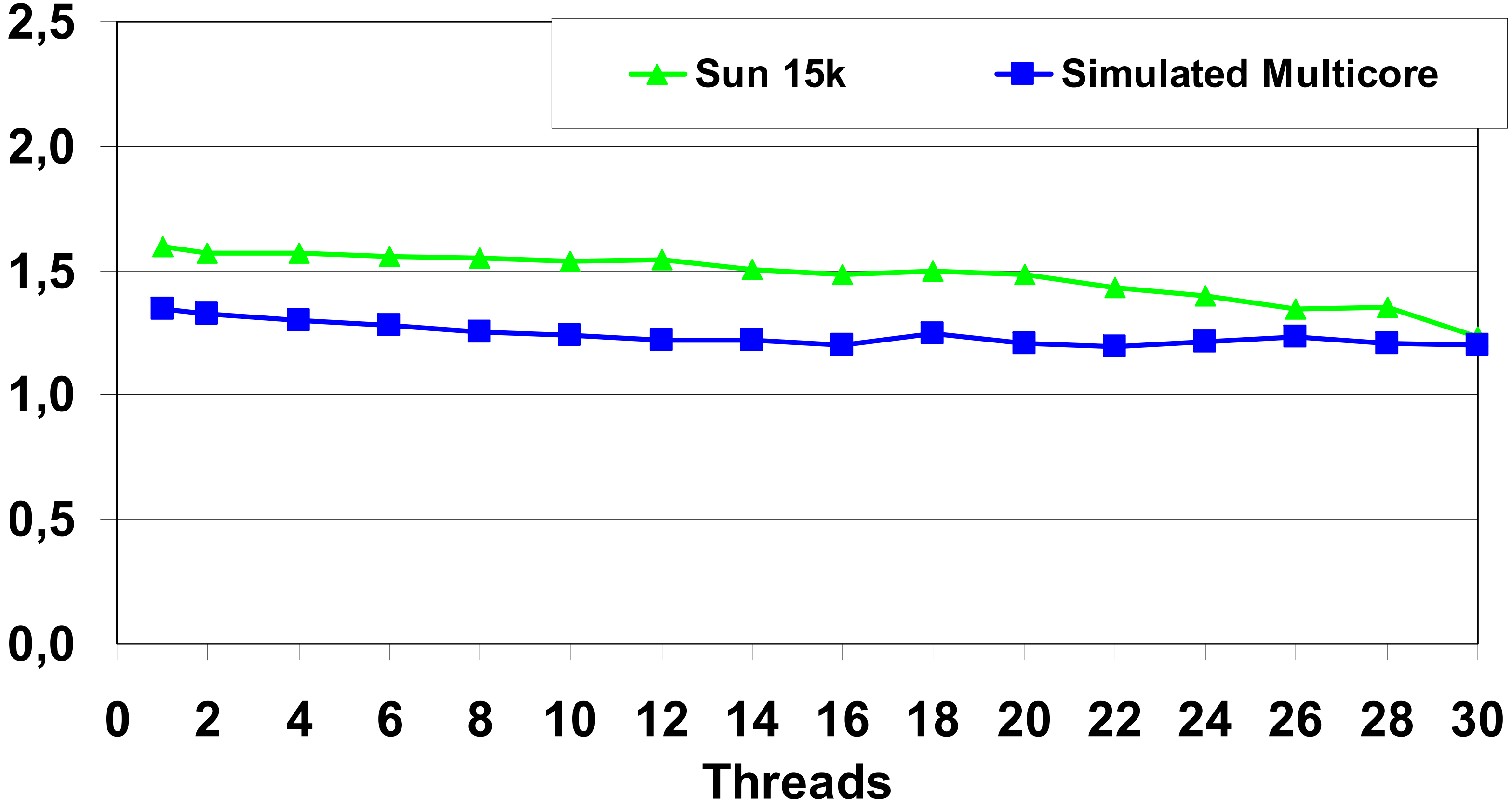
(Sun E15 K, US IIIcu = SMP!!)



Multicore Simulation
σ = 1

UPPSALA UNIVERSITET

Uppsala University



UPPSALA UNIVERSITET

Uppsala University

Using Gauss-Seidel Smoother in a Multigrid

- G-S Important part of many real apps!
- EX: G-S as a Smoother in “Multigrid”
  - Iterative algorithm
  - More efficient smother cuts #iterations

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.

One slide summary

- Today's algorithms assume expensive communication
- The communication cost of [some] multicores is close to zero
- Locality is becoming key to performance [again]

➔ Redesign HPC algorithms to face this fact!
(For both Capacity and Capability computing)

We show: * 3x performance gain
* ~30x less bandwidth

Is it time to revisit more algorithms?

Uppsala University

UPPSALA UNIVERSITET