Multicore Computing

Geilo Winter School in eScience, January 2008

Sverker Holmgren

Henrik Löf

Uppsala University

Sweden

Outline of Session 3

Examples:

- **Impact on algorithms:** Multigrid on multicores
- **Using a cluster of multicores and a hybrid programming model:** Computational genetics
- **Data locality is important:**
  - Large-scale FEM computation using conjugate gradients
  - Structured adaptive mesh refinement for a model problem on a NUMA system
- **A case for shared memory programming on multicores:** Reservoir flow simulation (streamlines)

Impact on Algorithms

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

The rules have changed!

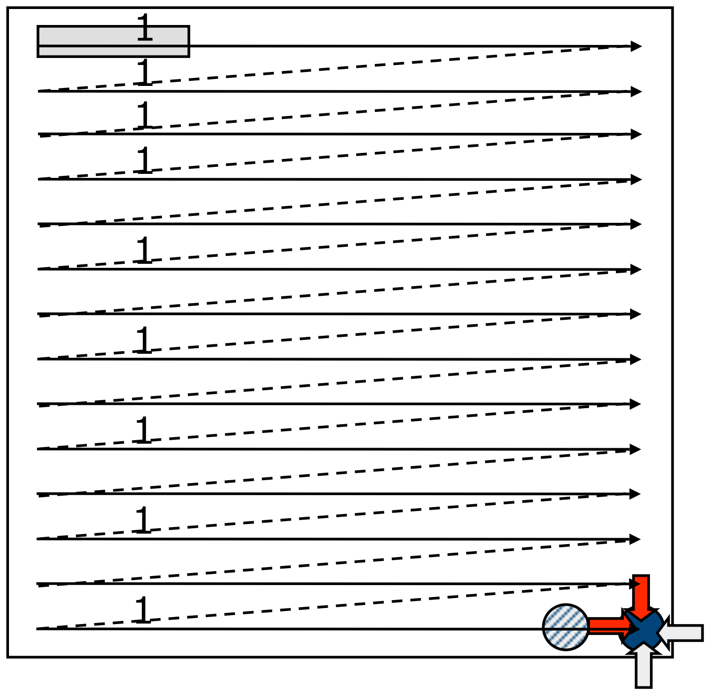
We need to question old algorithms and results

UPPSALA UNIVERSITET

Example: Multigrid on Multicore Systems

UPPSALA UNIVERSITET

Natural Order Gauss-Seidel



= sweep path

= previous

= current

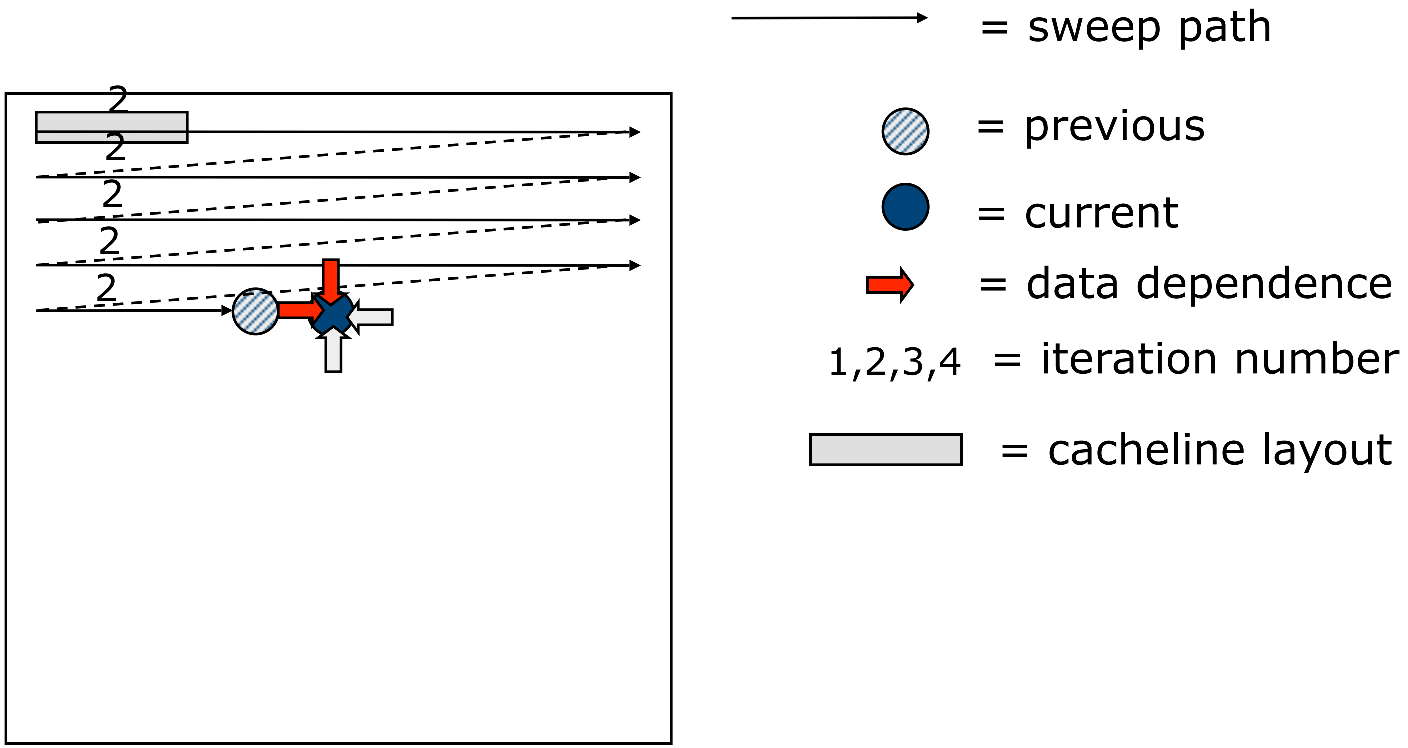
= data dependence

1,2,3,4 = iteration number

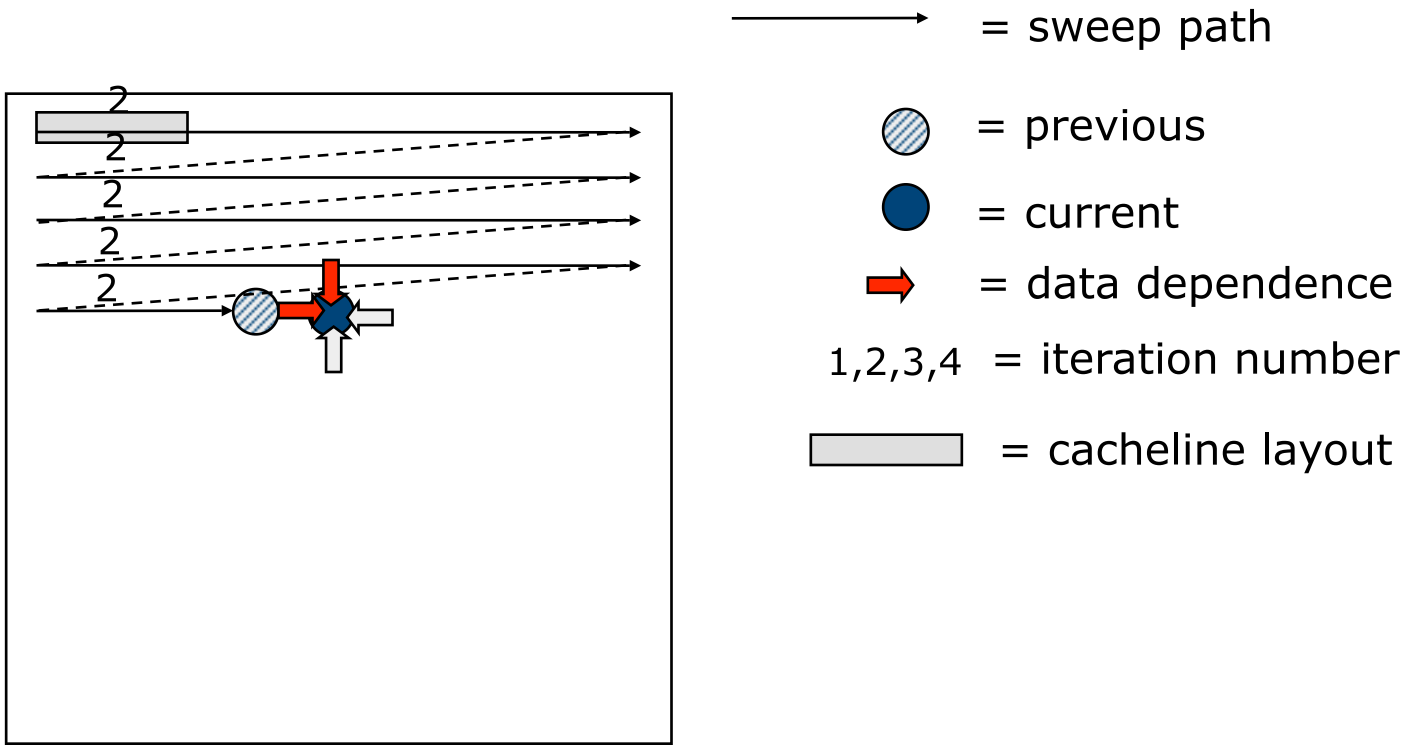
= cacheline layout

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

Natural Order Gauss-Seidel

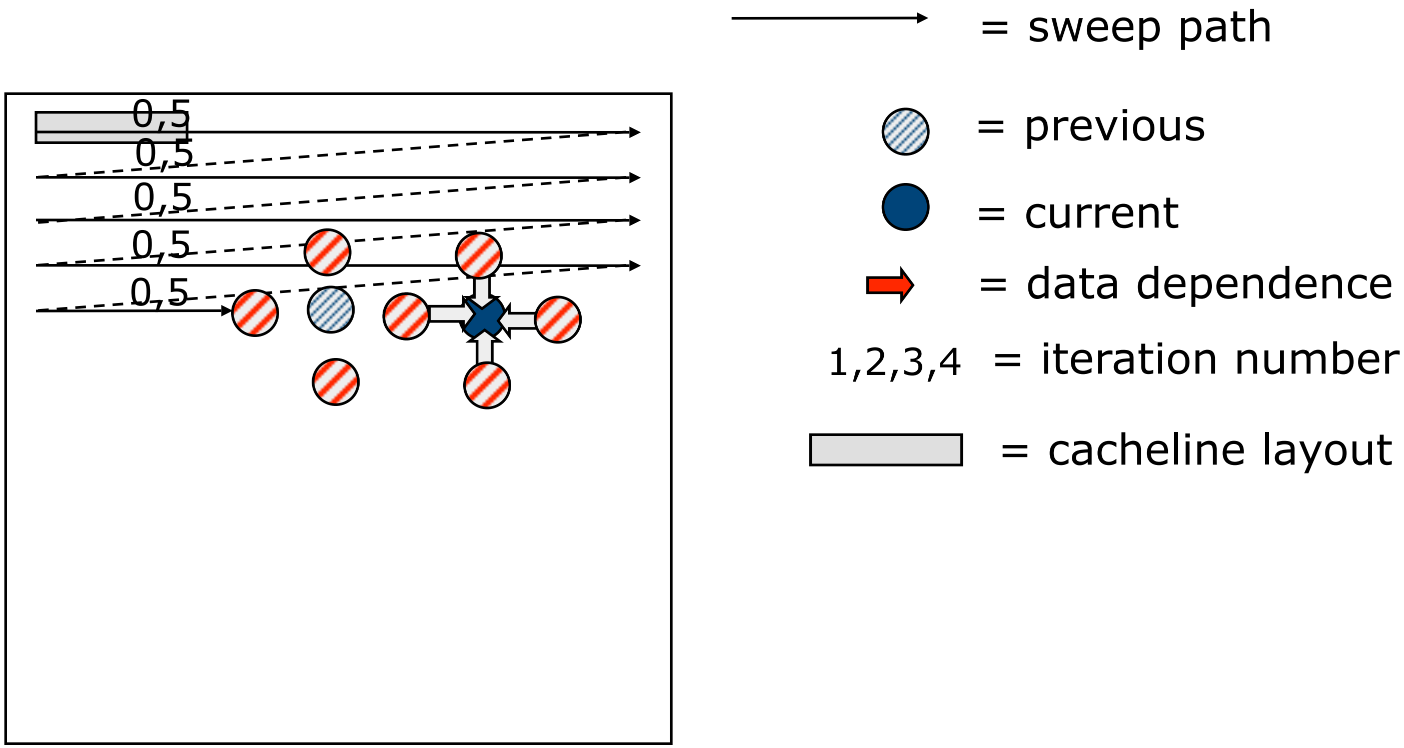


Natural Order Gauss-Seidel



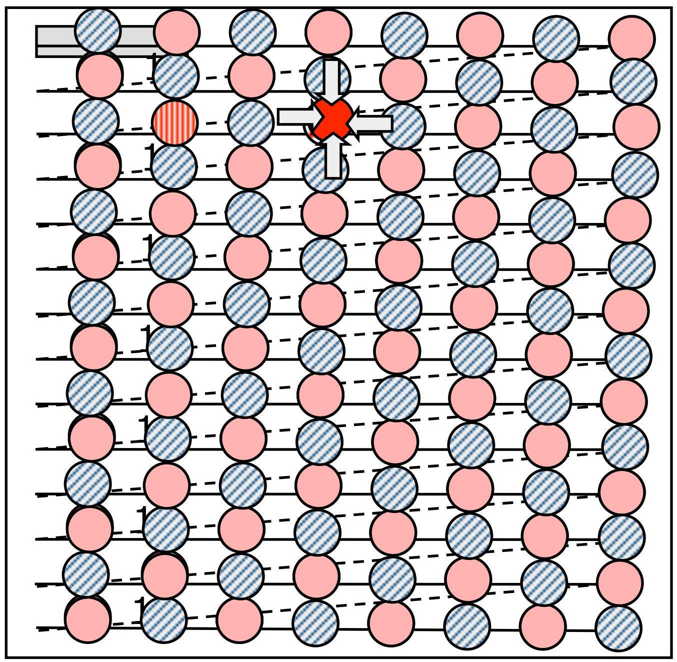
Data dependence ➔ Poor Parallelism☹

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



UPPSALA UNIVERSITET

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



= sweep path

= previous

= current

= data dependence

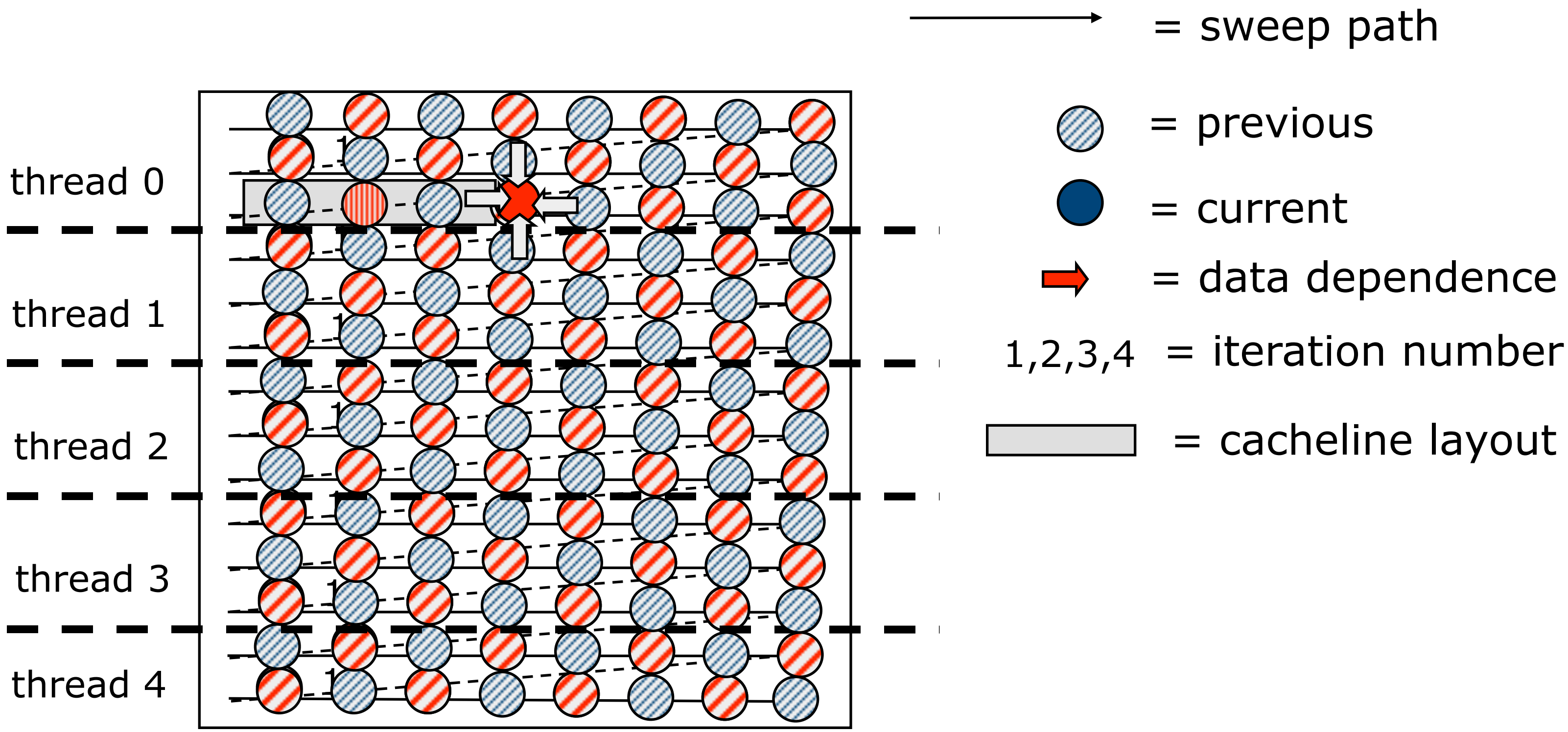
1,2,3,4 = iteration number

= cacheline layout

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

➔ great parallelism!!!

Red-Black Gauss-Seidel
Parallel version




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

Any Drawbacks?

UPPSALA UNIVERSITET

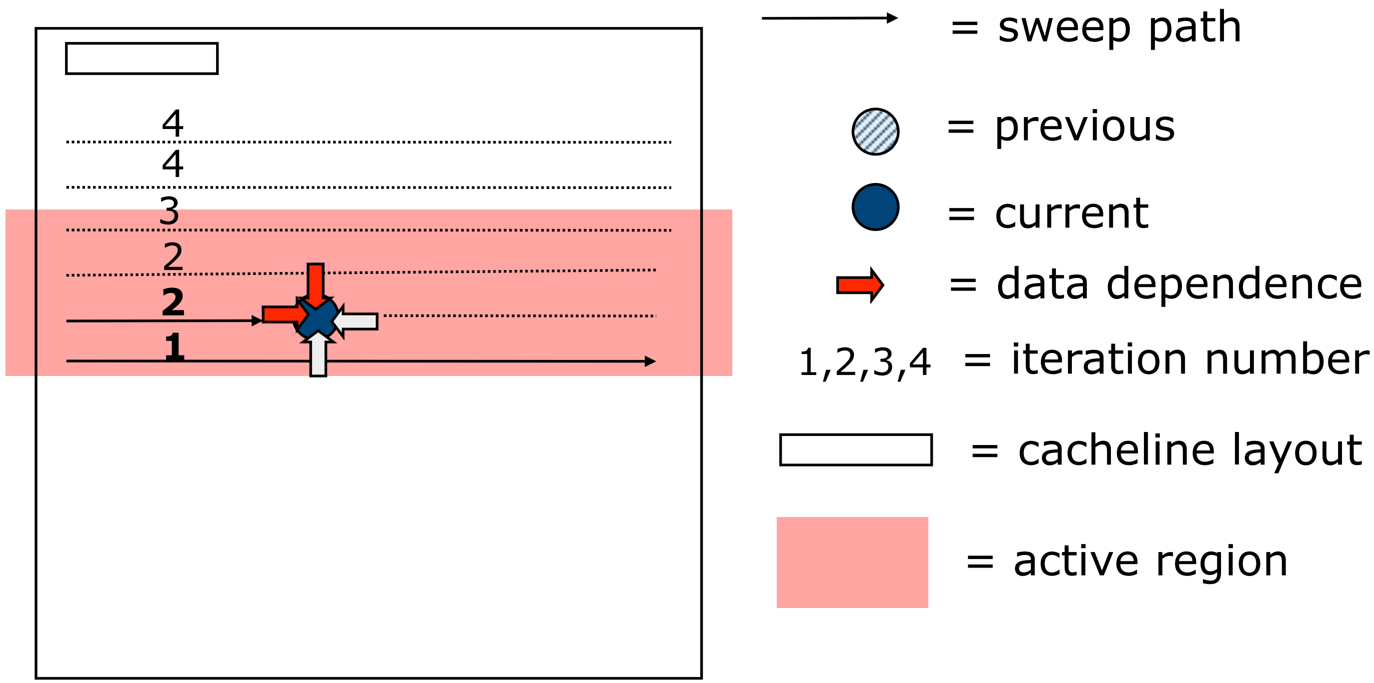
Any Drawbacks?

- 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 😐

Any Drawbacks?

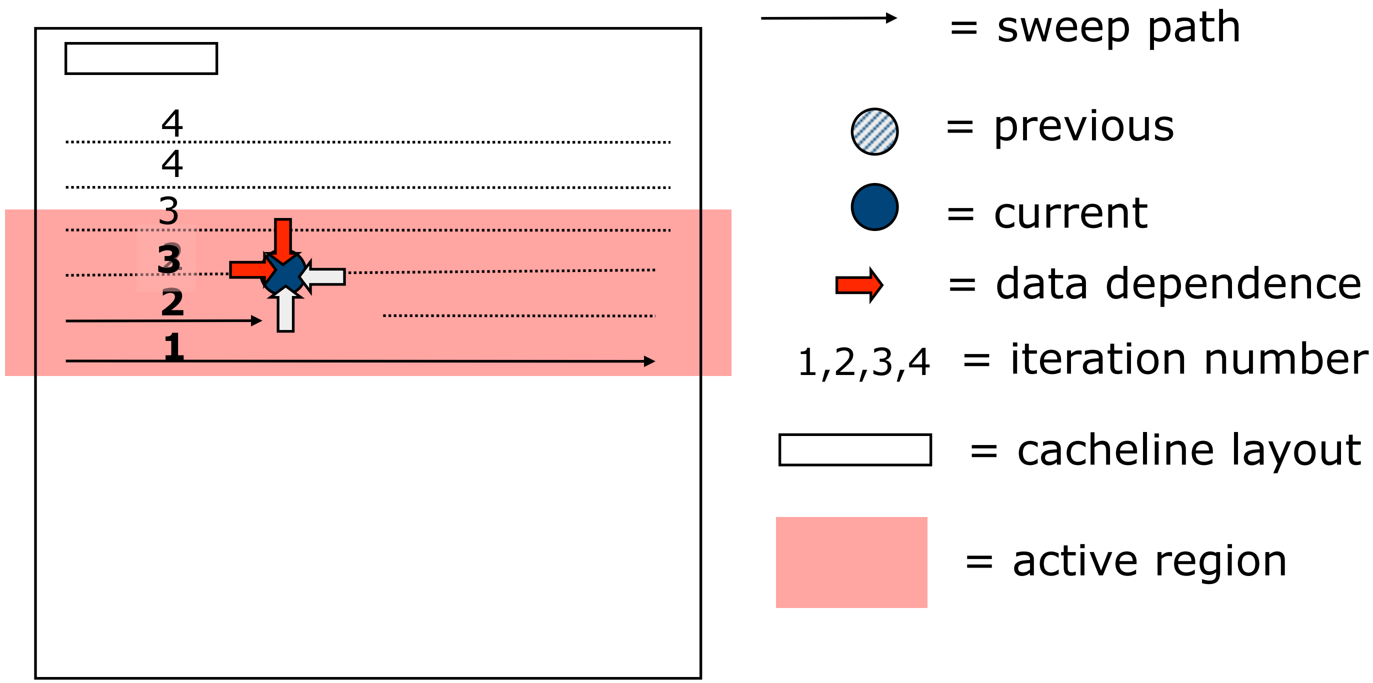
- Red-Black:
  - Each element will be brought into the cache **twice** per iteration ☹
- Natural Order:
  - Each element will be brought into the cache **once** per iteration 😐
- You can do better…
  - ➔ Temporal Blocking ☺

G-S, temporal blocking

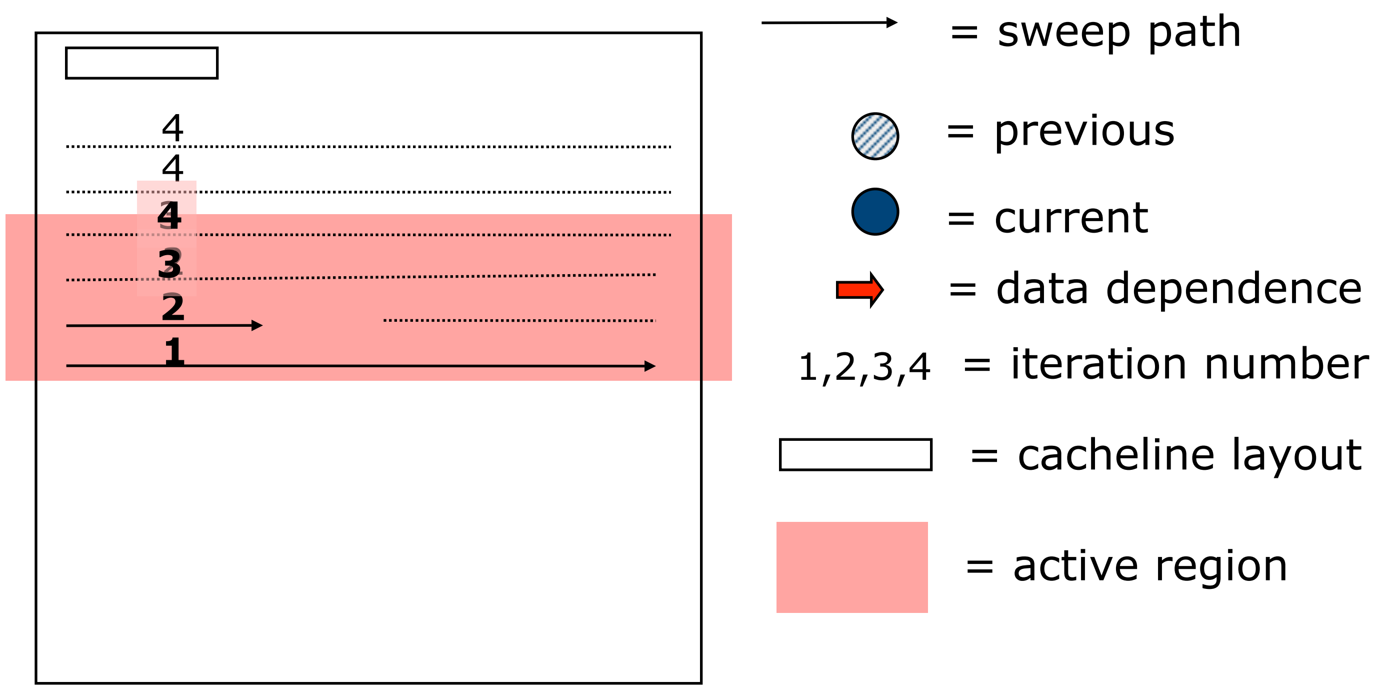


UPPSALA UNIVERSITET

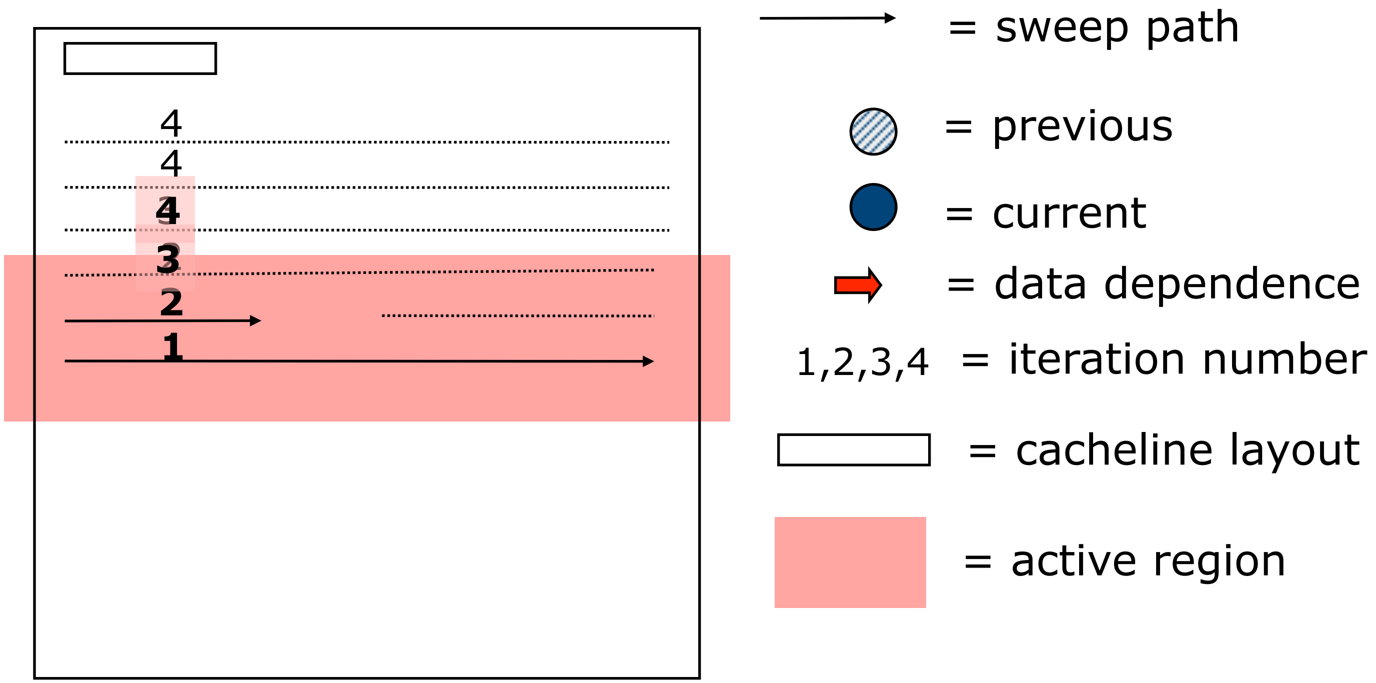
G-S, temporal blocking



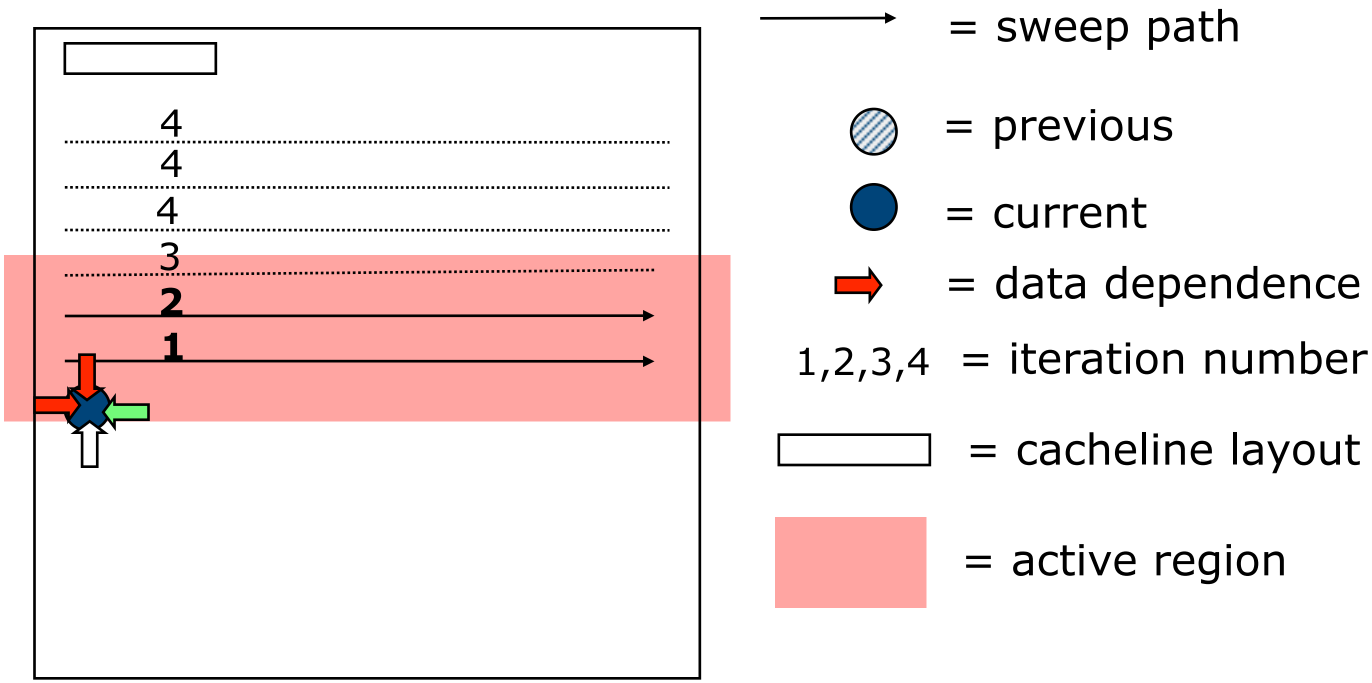
G-S, temporal blocking



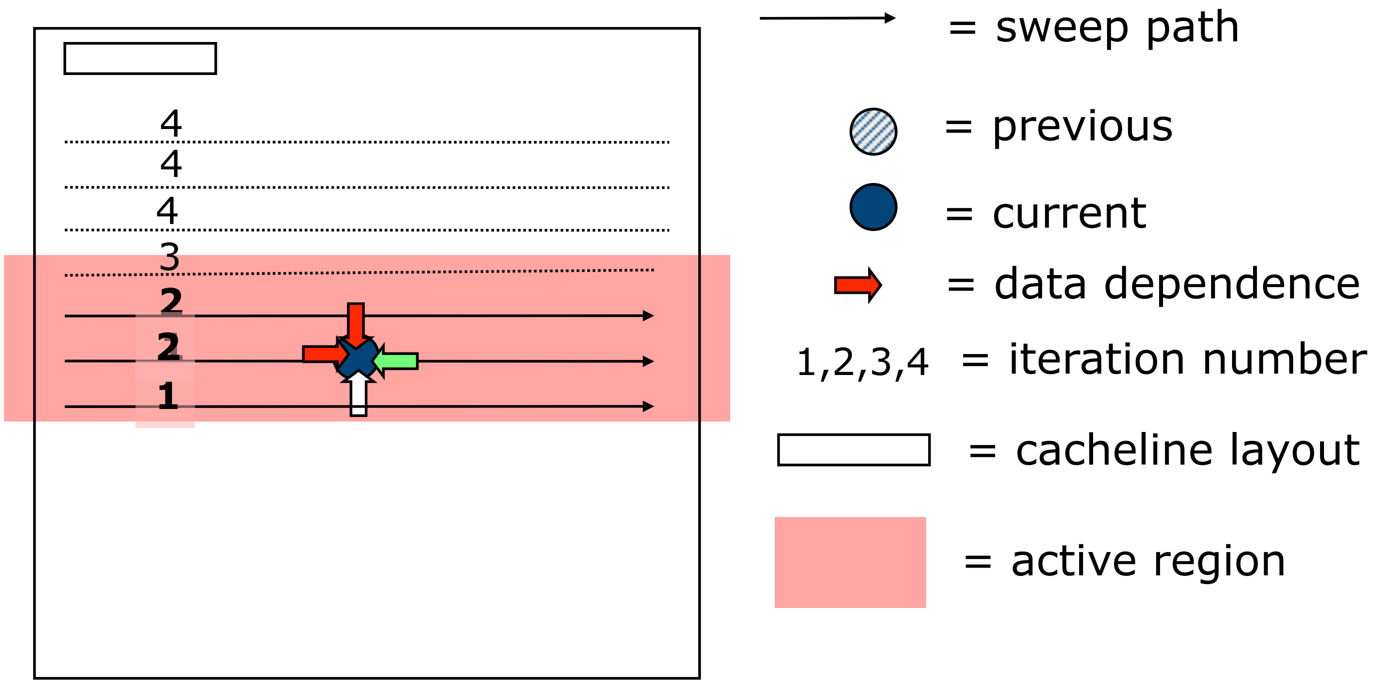
G-S, temporal blocking



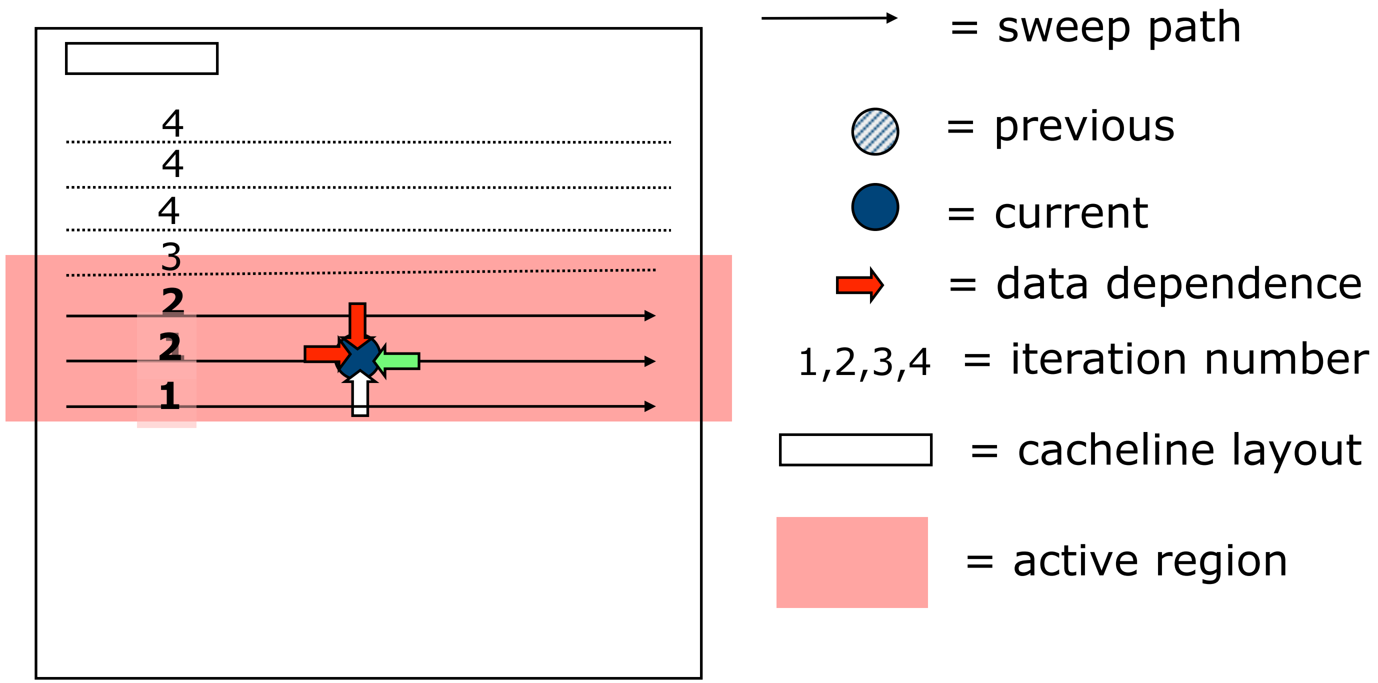
G-S, temporal blocking



G-S, temporal blocking

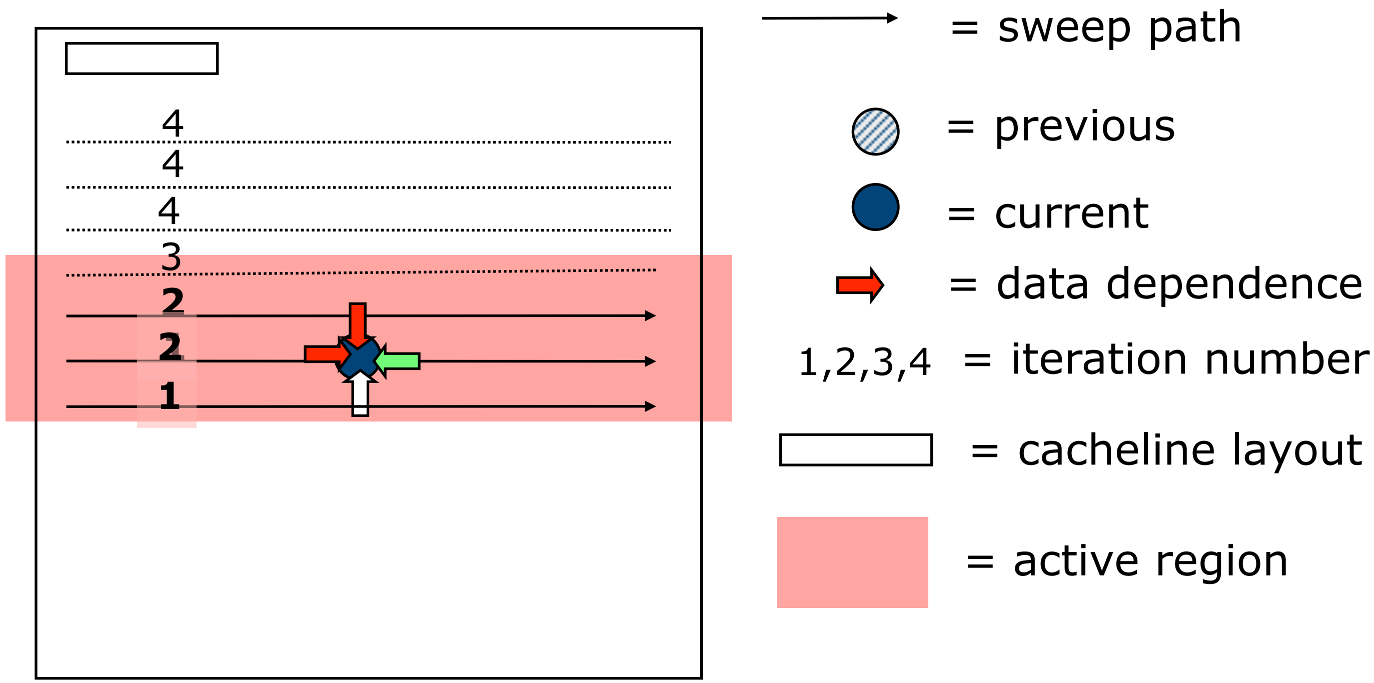


G-S, temporal blocking



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

G-S, temporal blocking

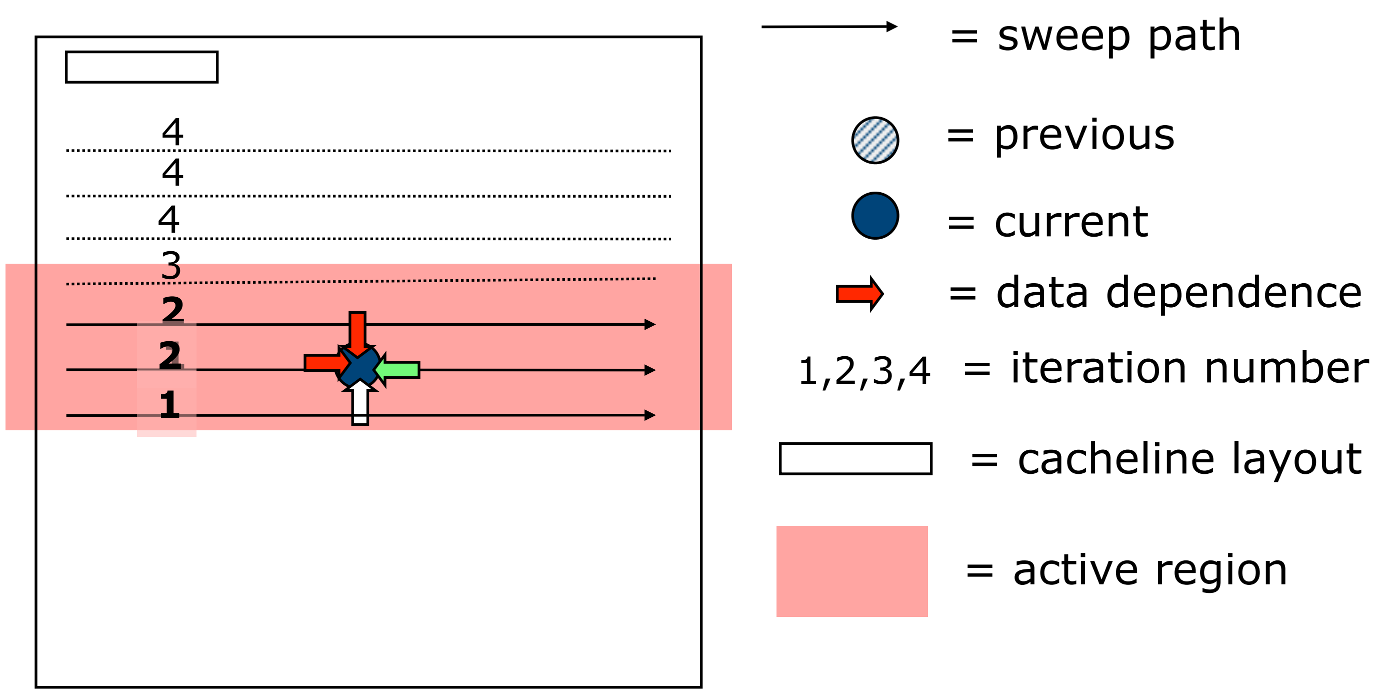


In this case: 4 iterations (*steps*) per *sweep*.

σ = #steps per sweep

UPPSALA UNIVERSITET

G-S, temporal blocking



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

σ = #steps per sweep
(σ = 1,0 for natural order and σ = 0,5 for red-black)

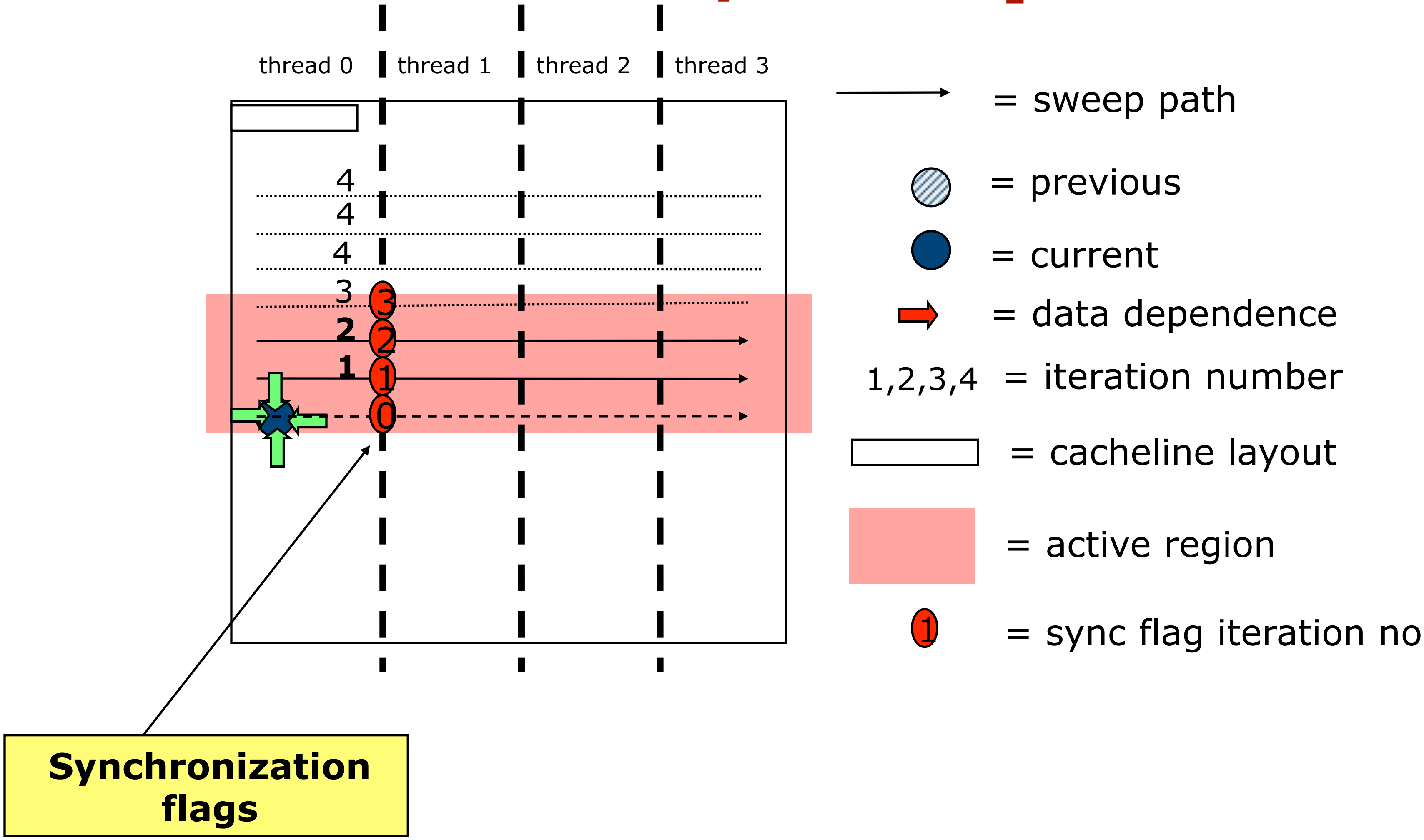
Sequential Execution time per step
(US4+, L1=16kB,L2=2MB,L3=32MB)

	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

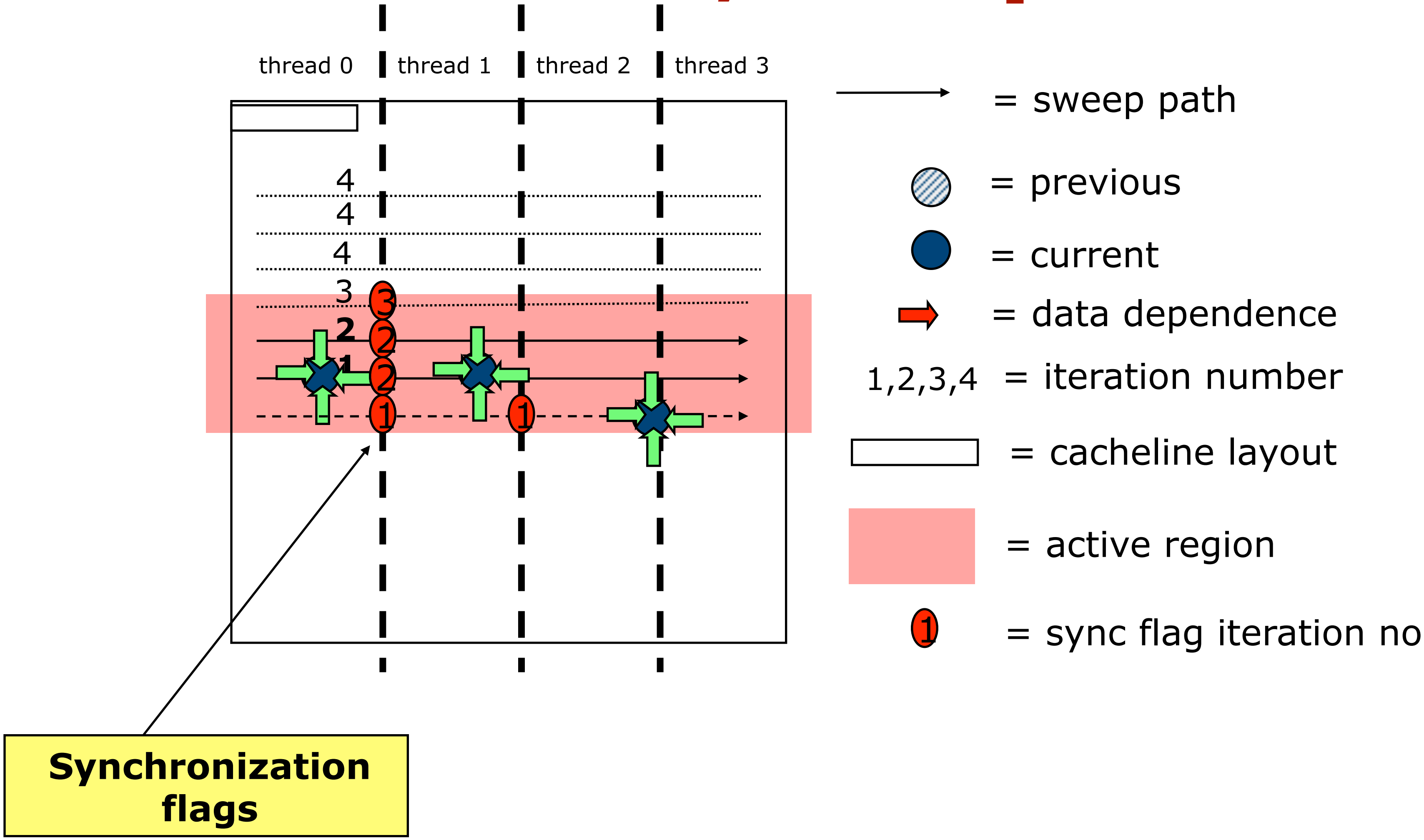
Required data size for each active region
(US4+, L1=16kB,L2=2MB,L3=32MB)

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

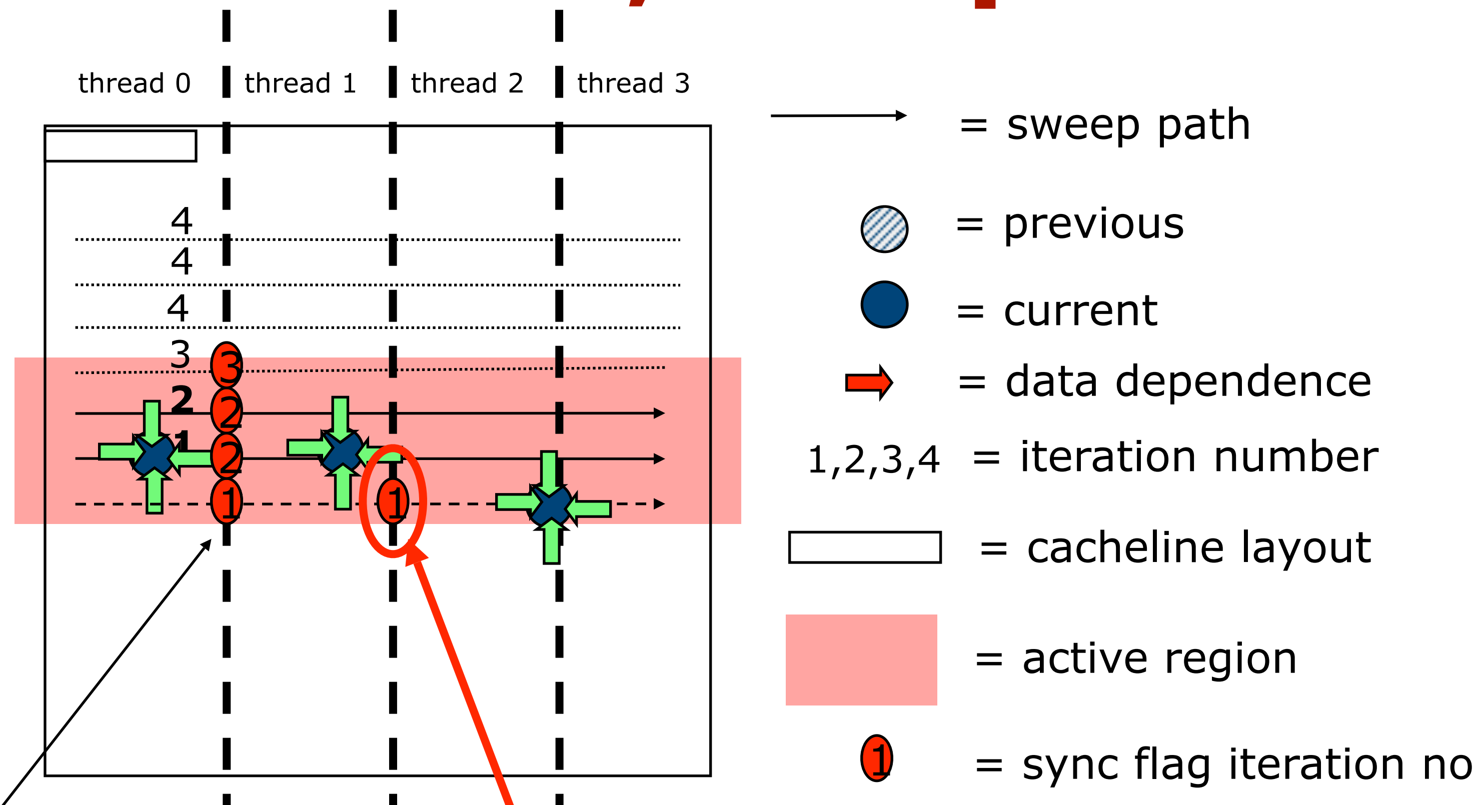


Parallel G-S, temp block



UPPSALA UNIVERSITET

Parallel G-S, temp block

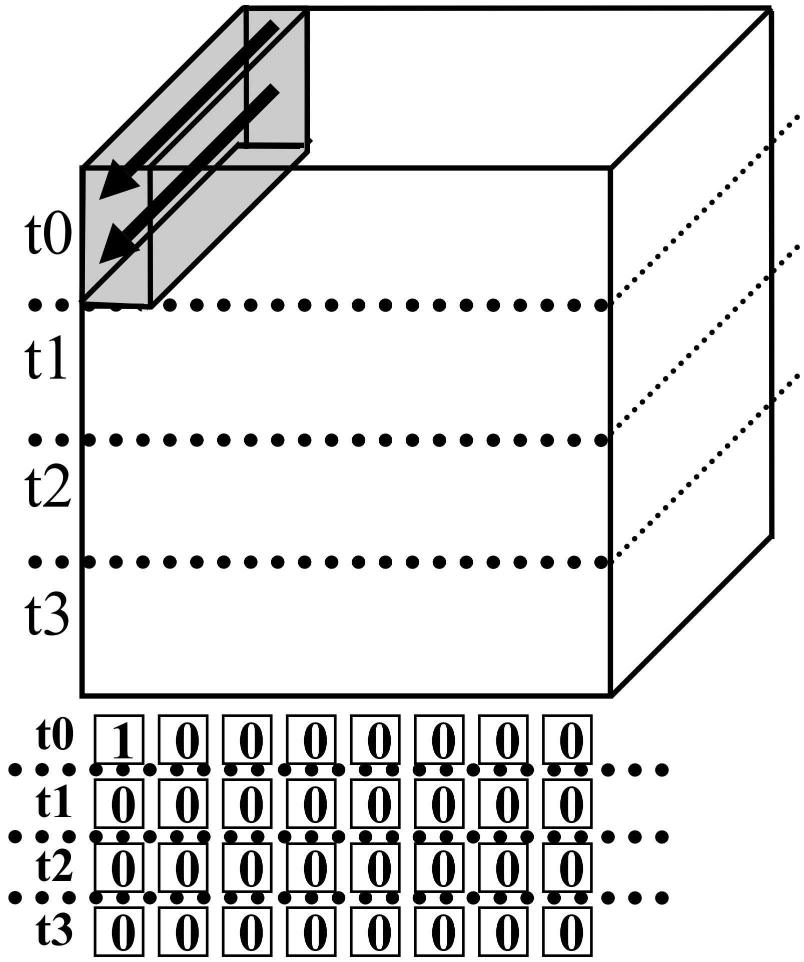


Synchronization flags

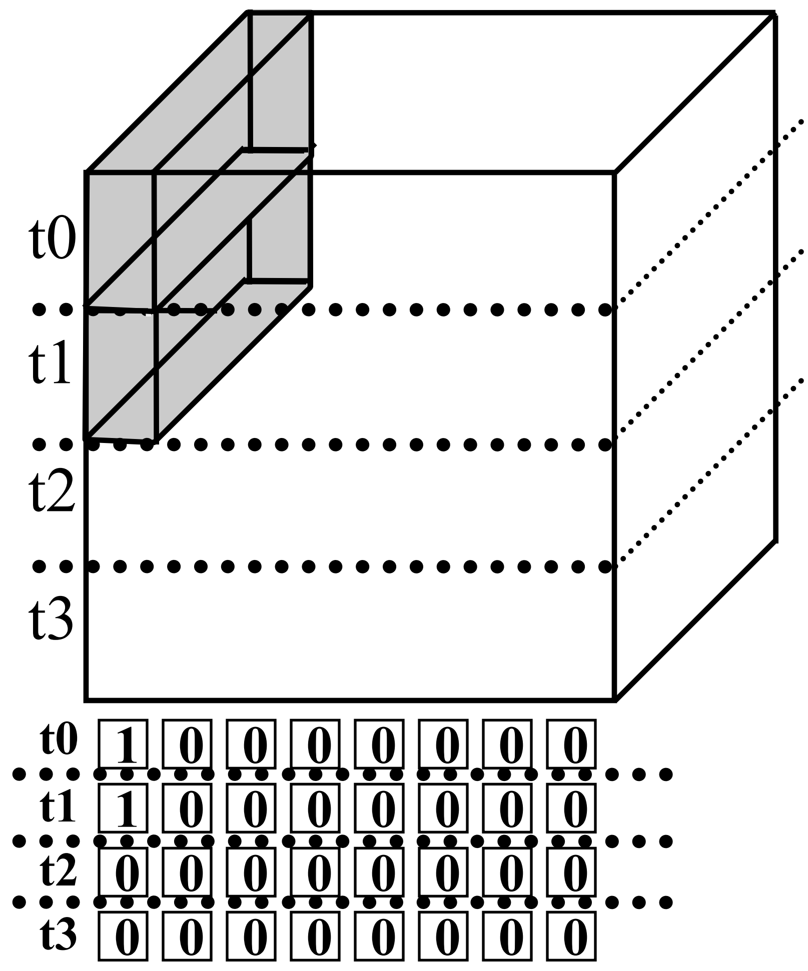
Wait until "lefty" is done:
Lots of communication

- Producer/Consumer Flag
- Sharing of data values

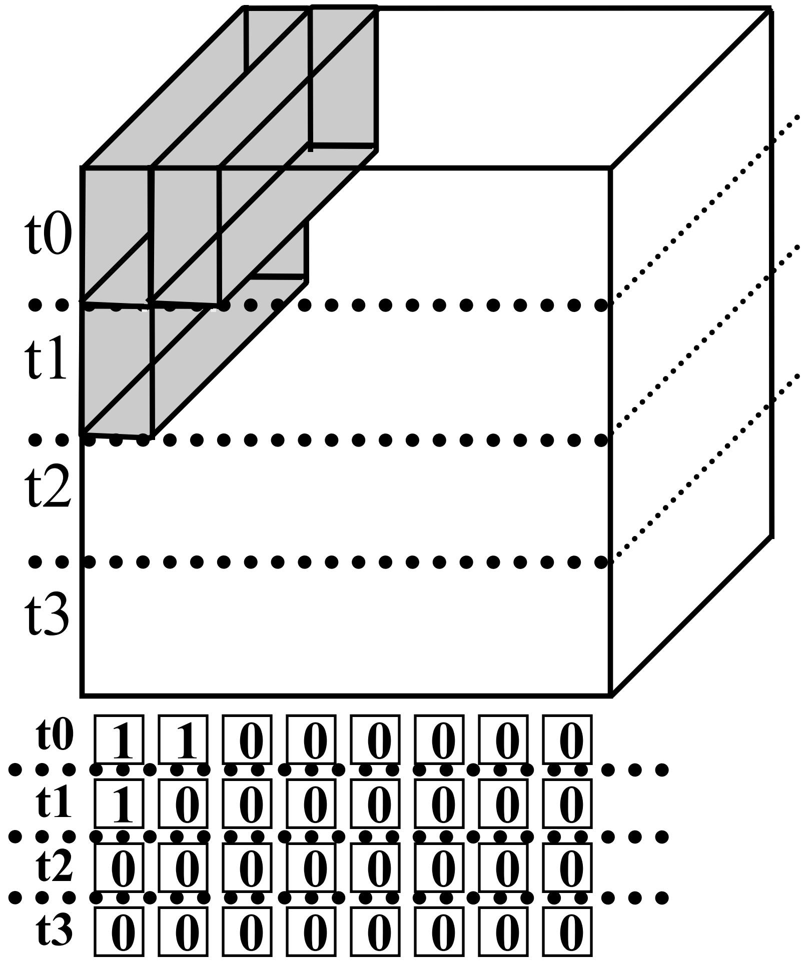
Parallel Natural-order G-S in 3D



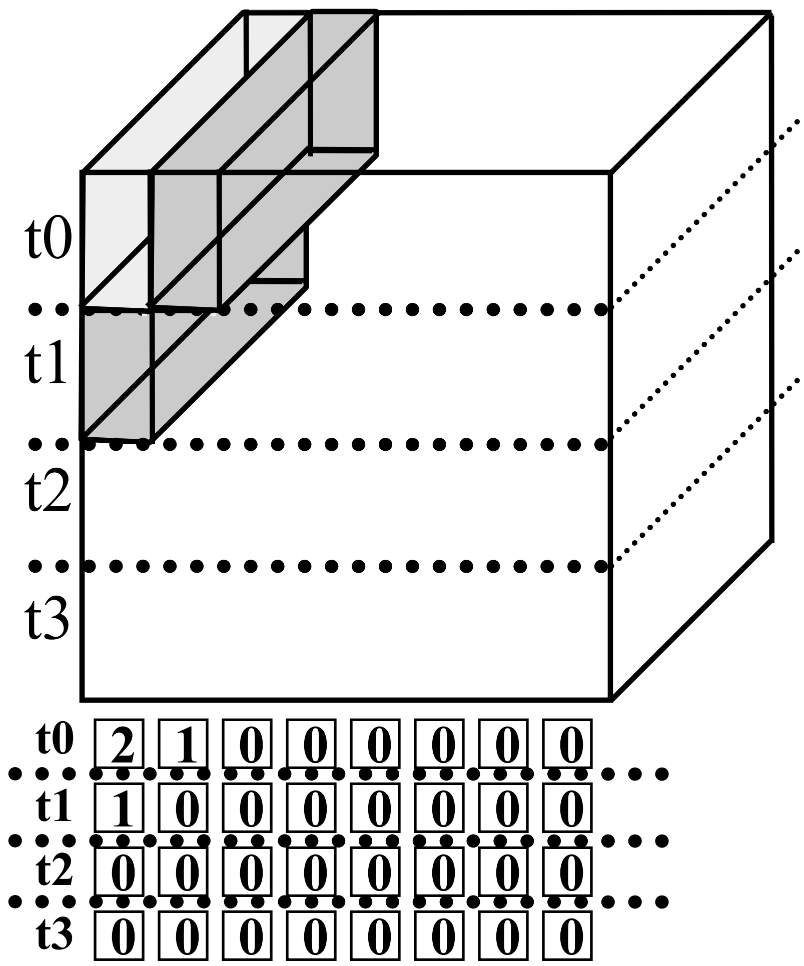
Parallel Natural-order G-S in 3D



Parallel Natural-order G-S in 3D

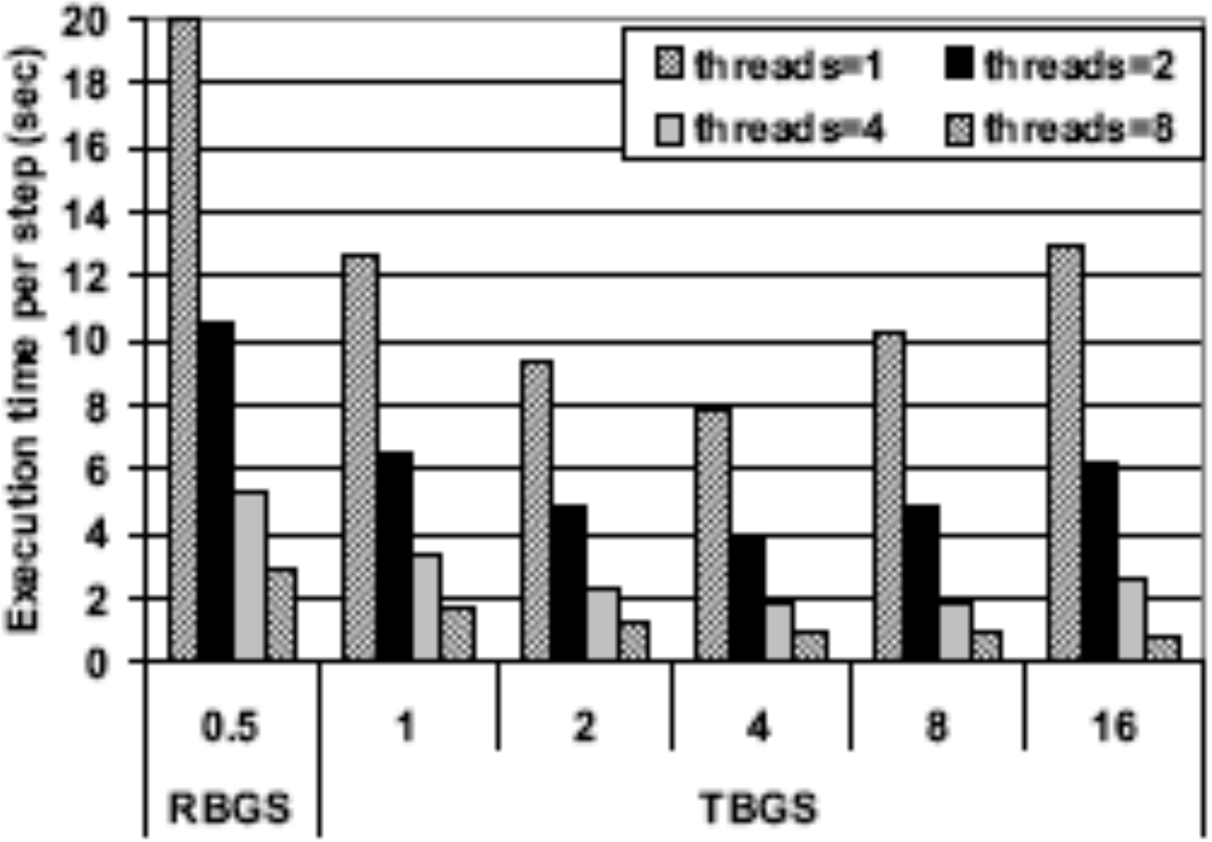


Parallel Natural-order G-S in 3D



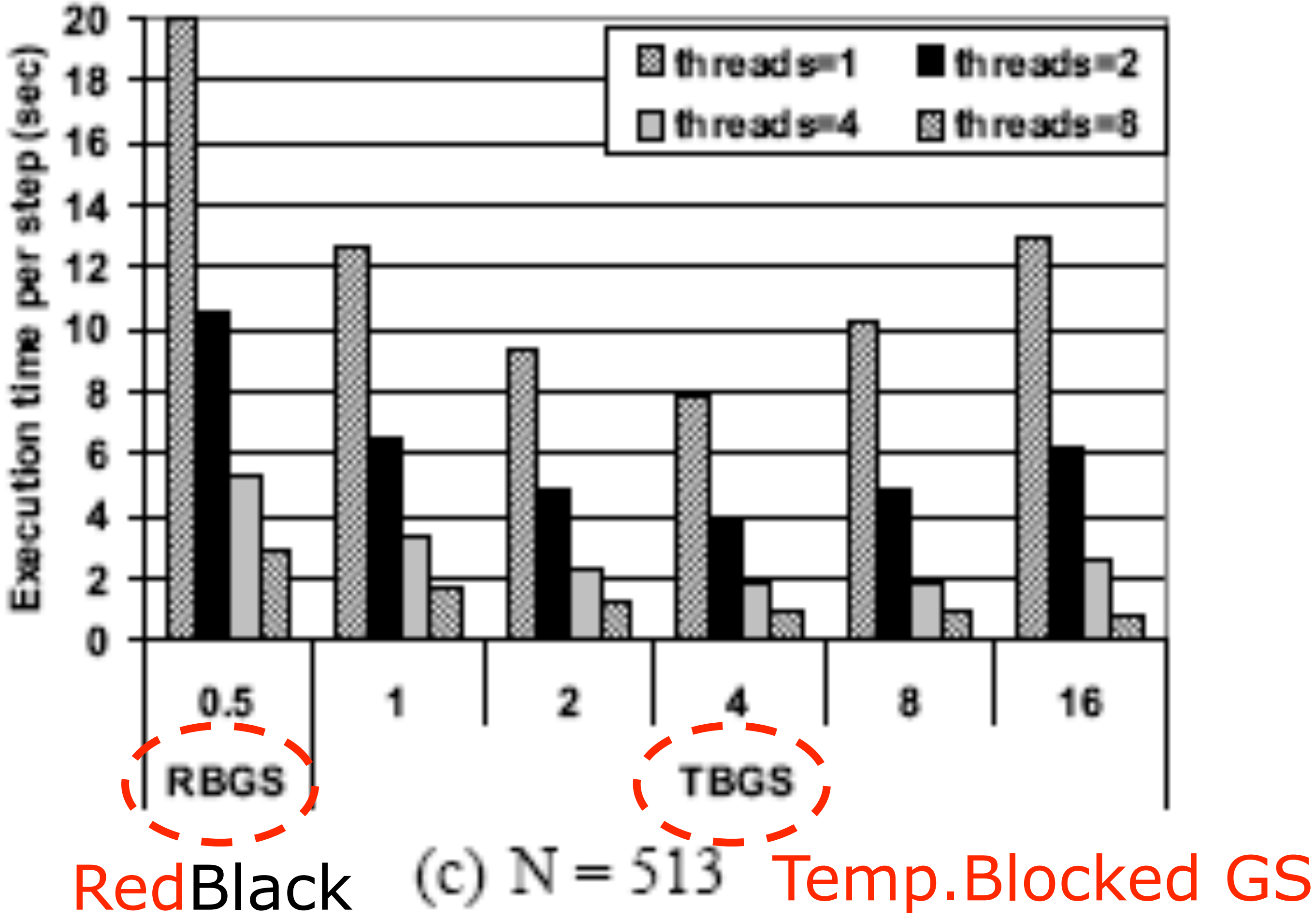
Parallel Execution Time

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

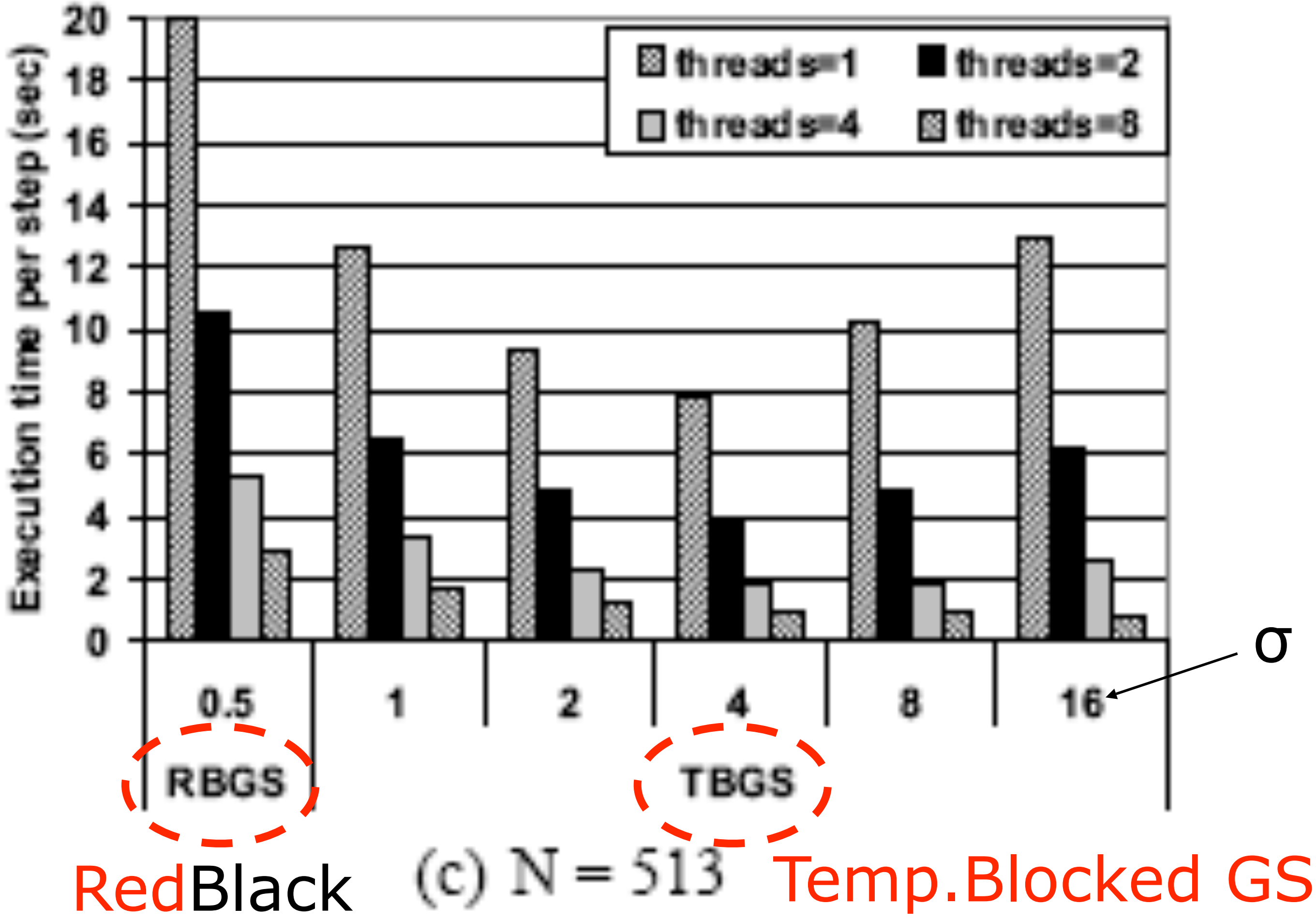


(c) N = 513

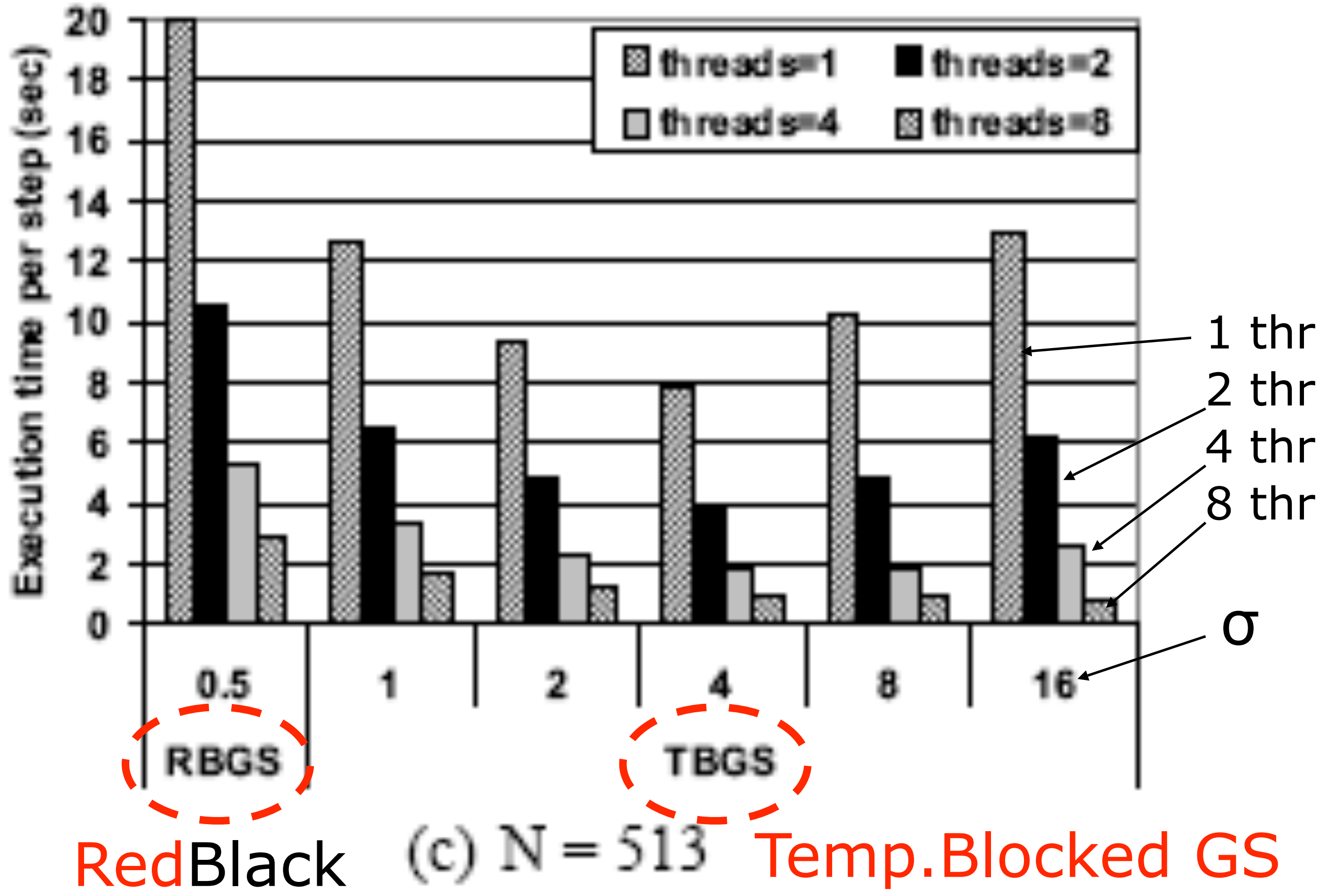
Parallel Execution Time
(Sun Fire 15k, 1.5 MHz US4+, $=p16kB/s2MB/s32MB)



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

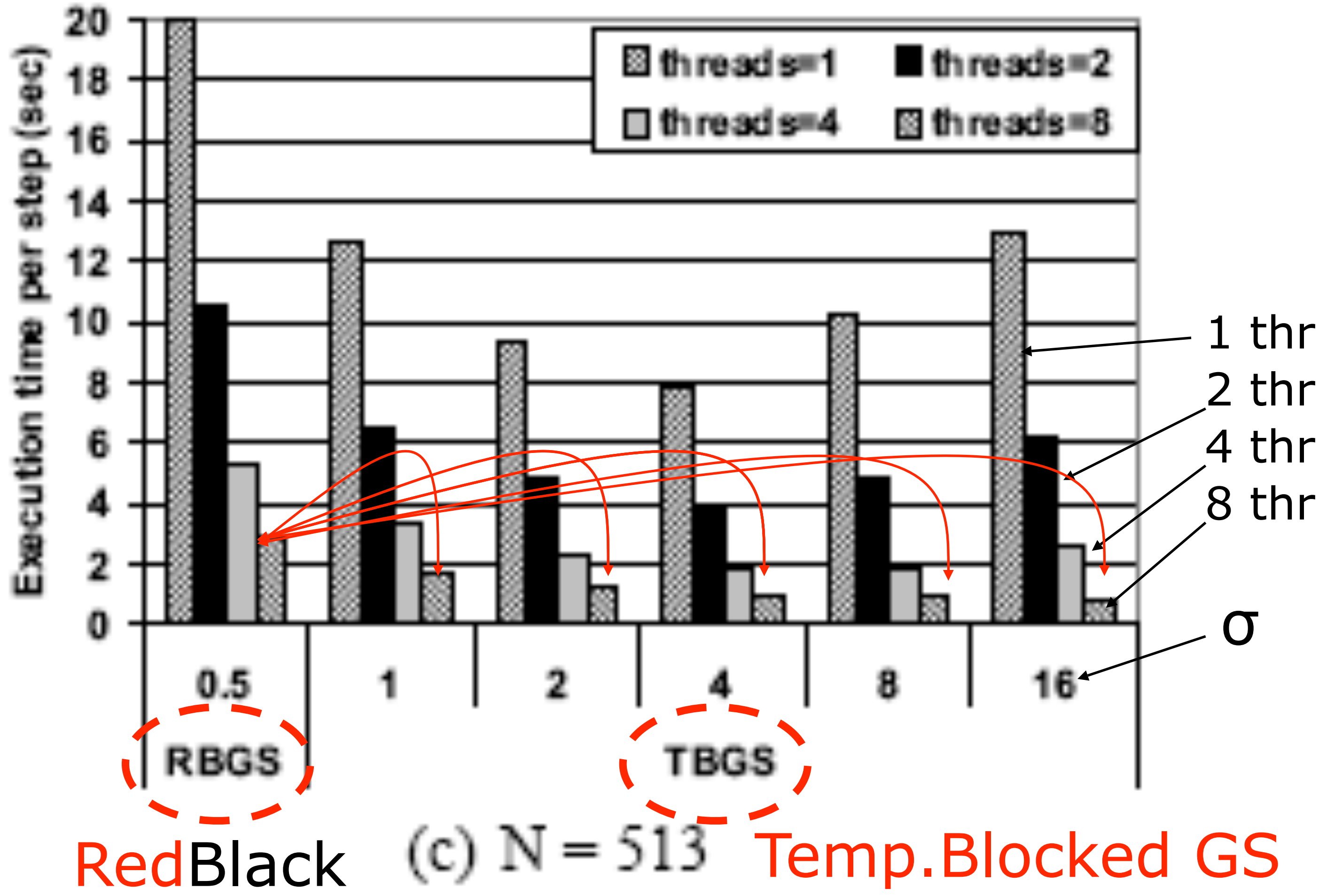


Parallel Execution Time
(Sun Fire 15k, 1.5 MHz US4+, $=p16kB/s2MB/s32MB)



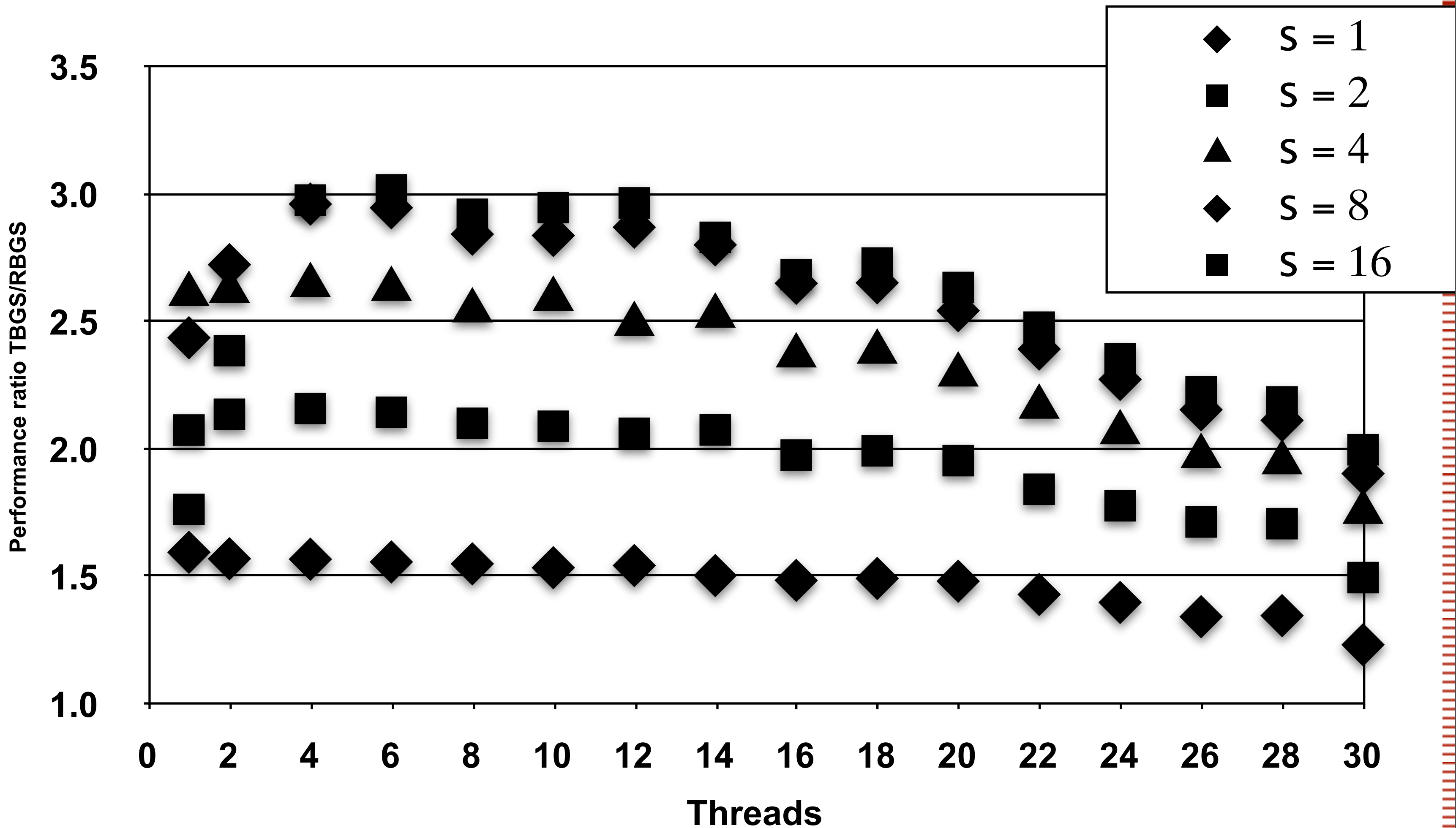
Parallel Execution Time

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

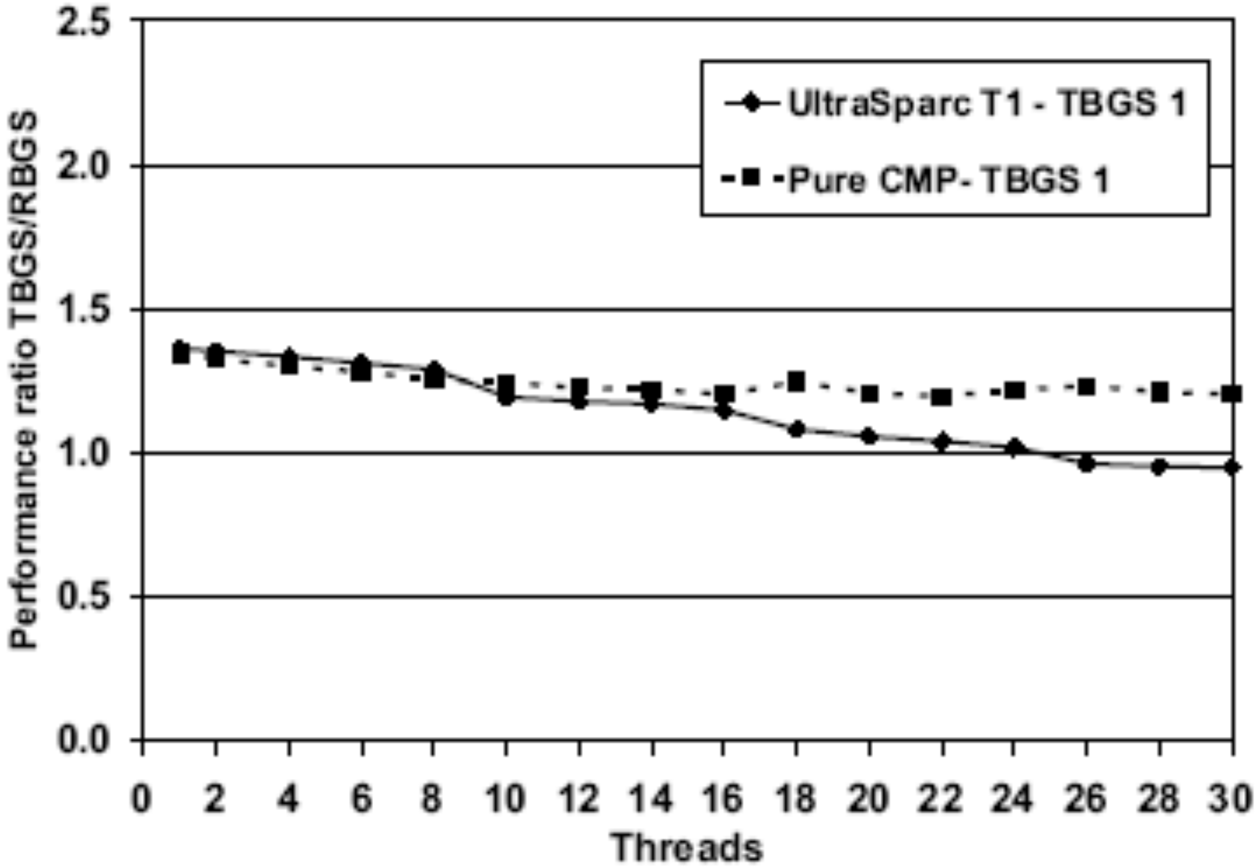


(c) N = 513

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



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



Gauss-Seidel Smoother in Multigrid

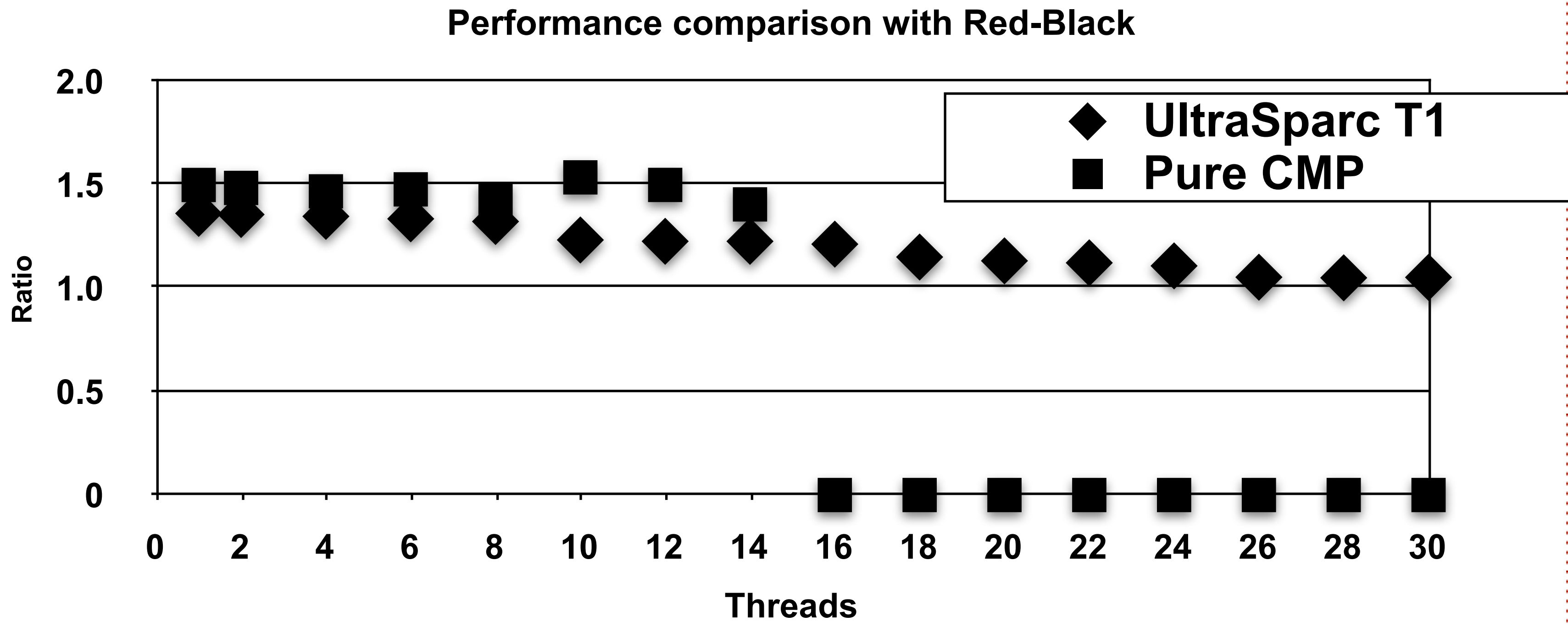
- Algorithm shown so far is rarely used alone.
- G-S as a Smoother in multigrid
  - Iterative algorithm for 3D Poisson eq.
  - More efficient smother cuts #iterations

	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.

Ratio on US T1 & Pure CMP, N=129

$\sigma = 1$



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

Fastest Red-Black vs. fastes 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

Example: Genetics on a Cluster of Multicore Processors

Where in the Genome are the Important Genes?

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

UPPSALA UNIVERSITET

Quantitative Trait Loci

QTL = A position in the genome affecting a quantitative trait

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

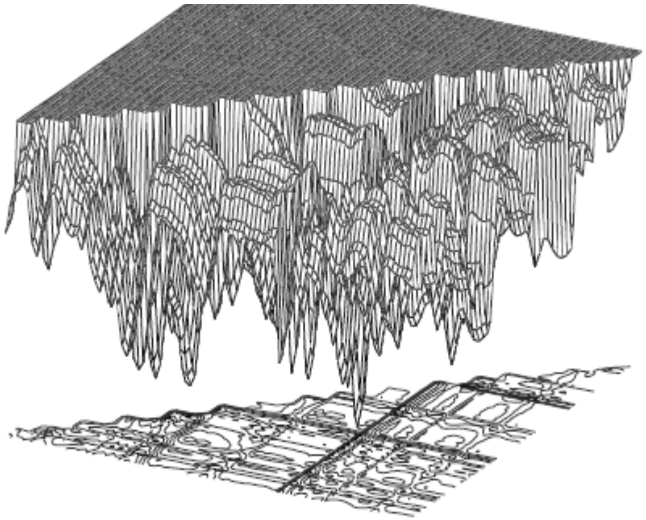
QTL model = Statistical model relating genotypes (genetic composition) to phenotypes (trait values) for an experimental population.

Mapping of *d* QTL

Fit model to data at various combinations of *d* positions. The set of QTL positions giving the best model fit is the most likely to be true.

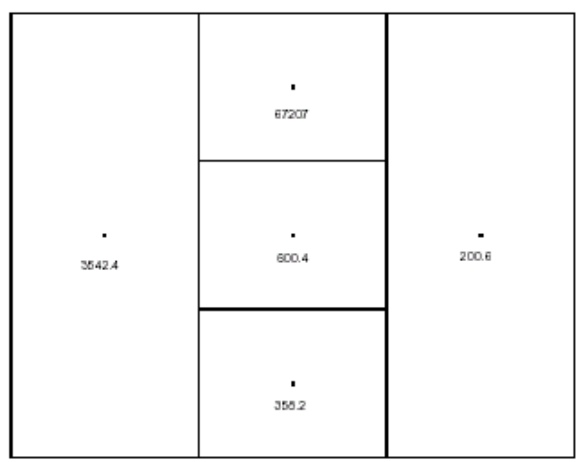
D-dimensional global optimization problem

$\bar{x}$

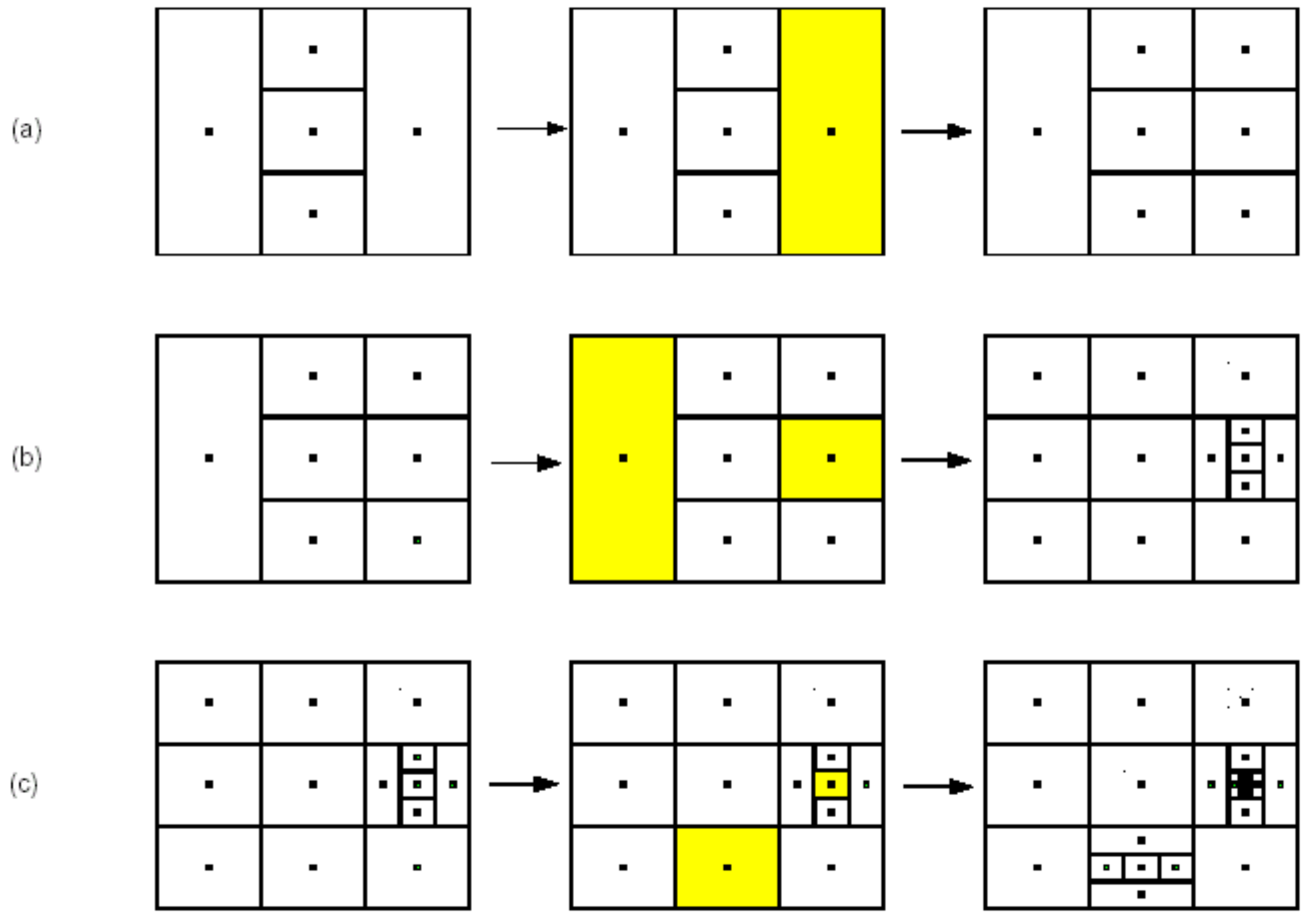


DIRECT

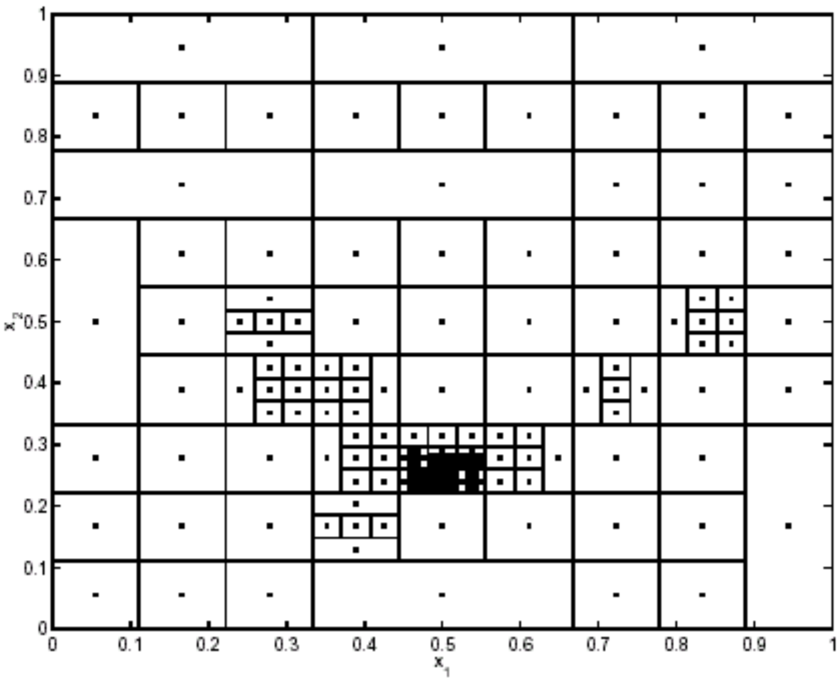
- Find global minimum of non-convex function in a rectangular domain
- Modification of Lipschitz optimization
  - but the Lipschitz constant is unknown
- DIvide search space in RECTangles and evaluate objective function at centers.



DIRECT cont...



DIRECT cont...



How to Parallelize DIRECT

- Domain decomposition
  - Parallelize over regions in the search space
- Convex hull
  - Parallelize over the rectangles divided in each iteration
- Objective function
  - Parallelize within the objective function evaluations

New version of DIRECT

- Initiate by evaluating the objective function in the center of each cc-box
  - Retains the analytic convergence properties
  - Creates a set of C^d “independent” optimization problems
- Parallel scheme
  - Several DIRECT in sets of cc-boxes
  - Exhaustive search over the results from the DIRECT searches
  - NOTE: Communication and synchronization is removed from the original algorithm!

Implementation

UPPSALA UNIVERSITET

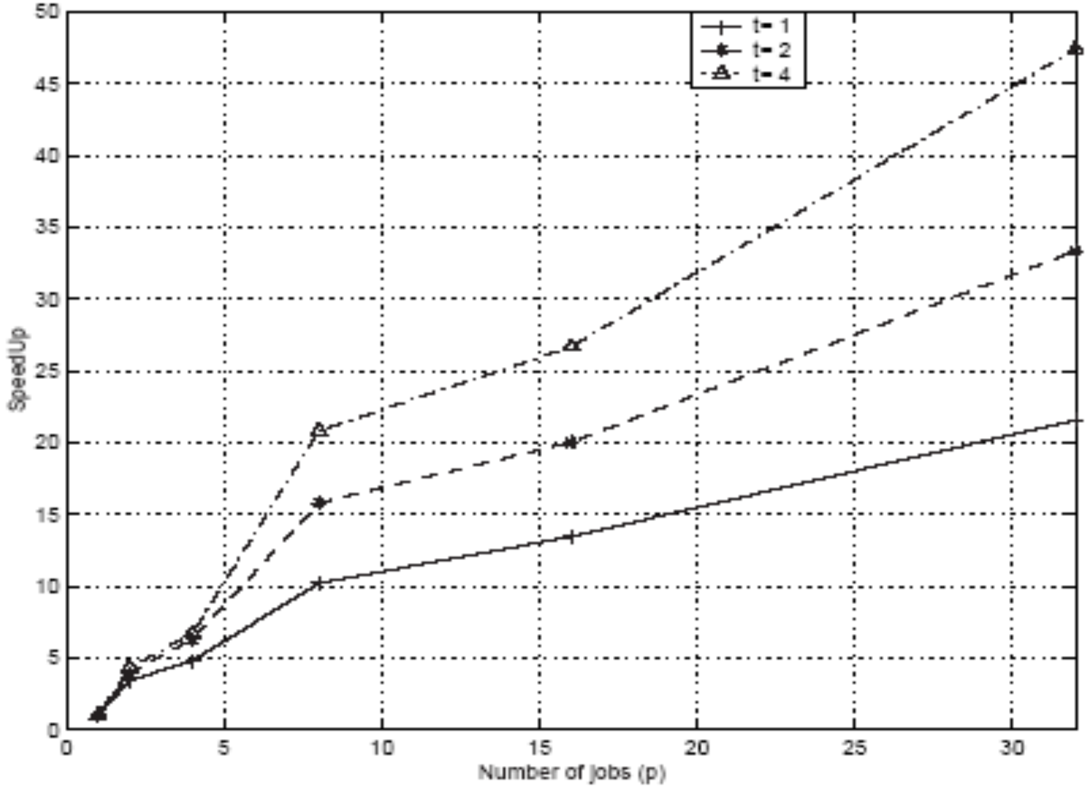
- Submit one serial job for each set of chromosome combination boxes (cc-box)
- Compare the local minima

Earlier: Execution on the SweGrid system and clusters

NOW: Including OpenMP parallelization of the convex hull computations within each grid/cluster job.

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

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



- More details: *Grids and Clusters with Multi-Core Nodes: A Genetics Application Perspective,* Mahen Jayawardena, Henrik Löf, Sverker Holmgren, Future presentation at the SIAM Conference on Parallel Processing for Scientific Computing, Atlanta, March 2008.