Parallel Query Scheduling and Optimization with Time- and Space-Shared Resources

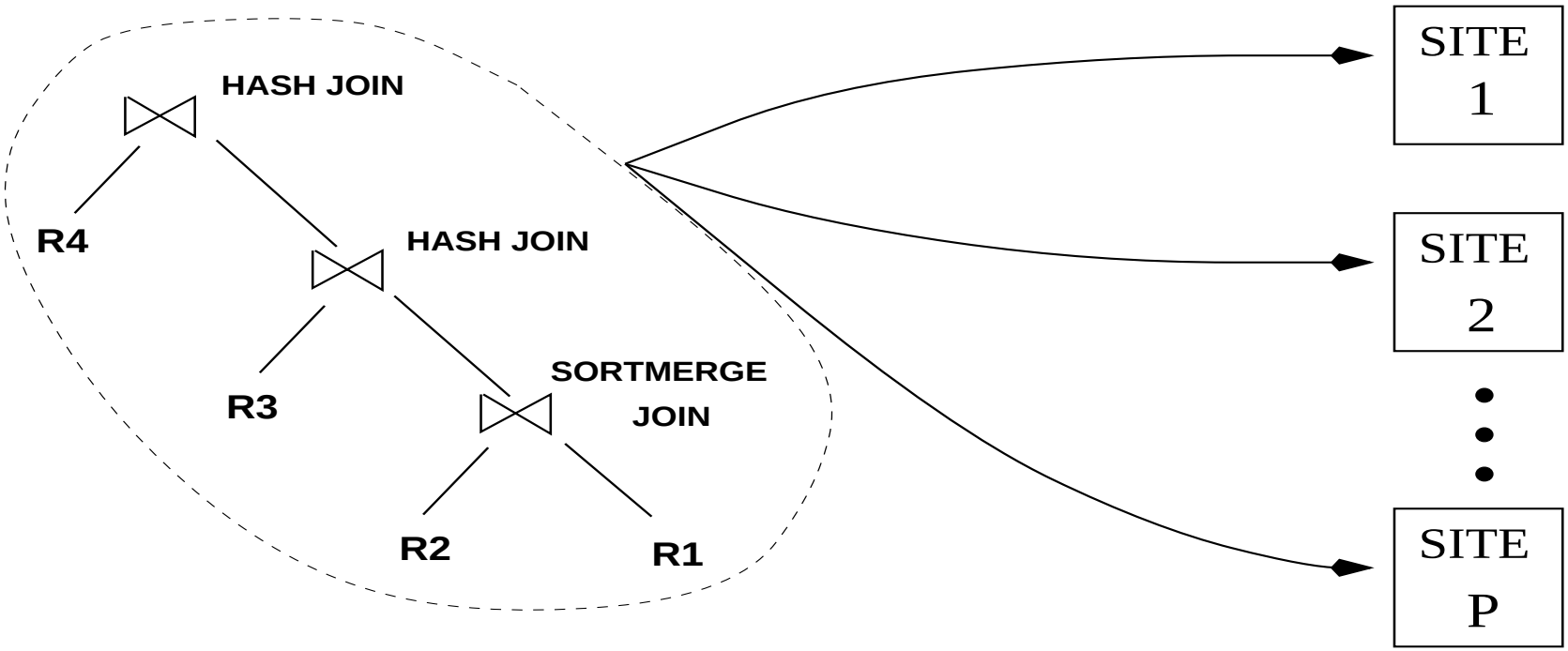
Minos N. Garofalakis
University of Wisconsin – Madison

[Joint work with Yannis Ioannidis]

Outline

- Introduction and Motivation
- Problem Formulation and Query Execution Model
- Heuristic Query Scheduling Algorithms
- Experimental Results (Cost Model & Detailed Simulation)
- Implications for Parallel Query Optimization
- Ongoing and Future Work

Introduction: Parallel Database Systems



- Exploit parallelism and available resources to reduce response times and increase system throughput
- Effective mappings of Query Execution Plans on parallel architectures

Introduction and Motivation

The Problem

- Architecture: Shared-Nothing or Hierarchical
- *Static* parallelization of set of complex query plans, based on size & cost estimates
- Objective: minimize query response time

Why?

- Most scalable architectural paradigms
- Part of cost-based query optimization (compile-time)
- Effective parallel query execution

Introduction and Motivation (cont.)

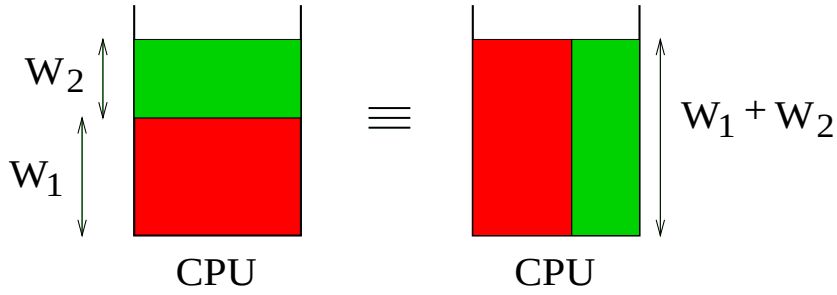
Sources of Complexity

- Various forms of intra/inter-operator parallelism
- Cost Model must account for communication overheads
- "Multi-dimensional" resource needs of DB operators (opportunities for resource sharing)
- Resources: *Time-Shared* (e.g, CPU(s)) or *Space-Shared* (e.g., memory)

- *NP-hard*, even for simple "one-dimensional" cases

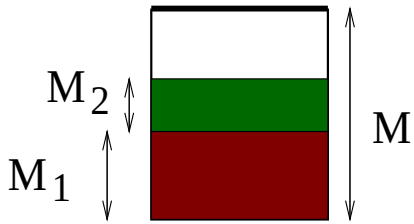
Goals Incorporate TS & SS dimensions; Effective scheduling heuristics; Implications for Query Optimization

Time-Shared vs. Space-Shared



Time-Shared (TS) Resources

- Sliced between operators at negligible overhead
- Effective time can be stretched over execution interval



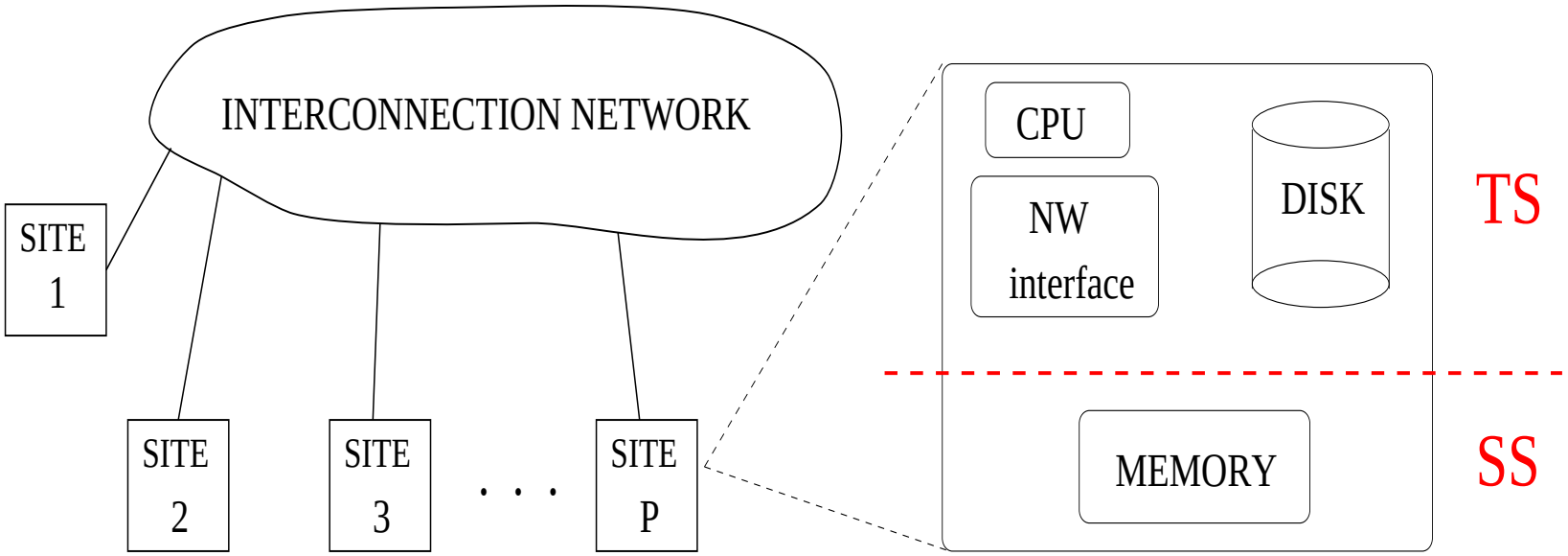
$M_1 + M_2 \leqslant M$

Space-Shared (SS)Resources

- Slicing implies prohibitive overhead
- Limited capacity that cannot be exceeded by executing operators

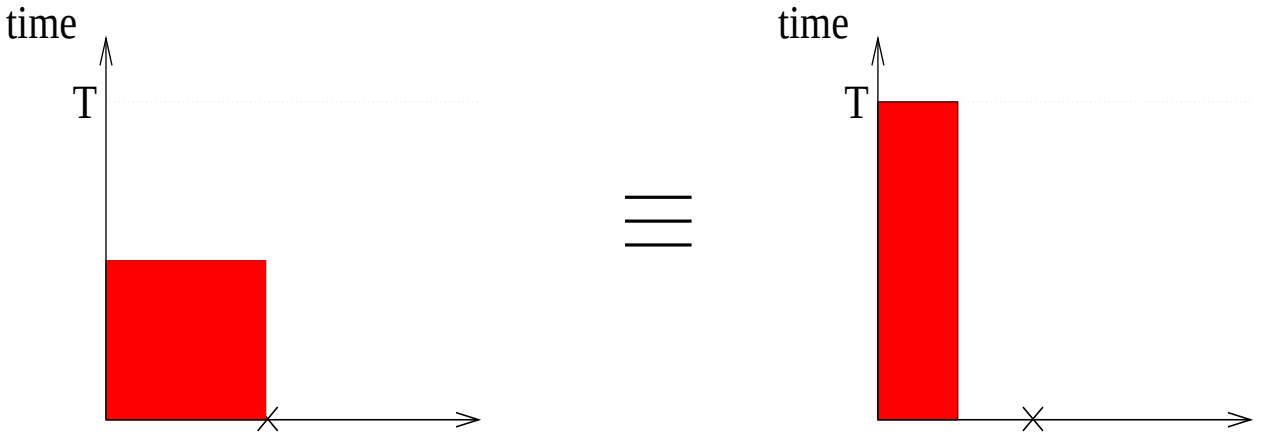
System Model

- Identical sites consisting of d TS and s SS resources



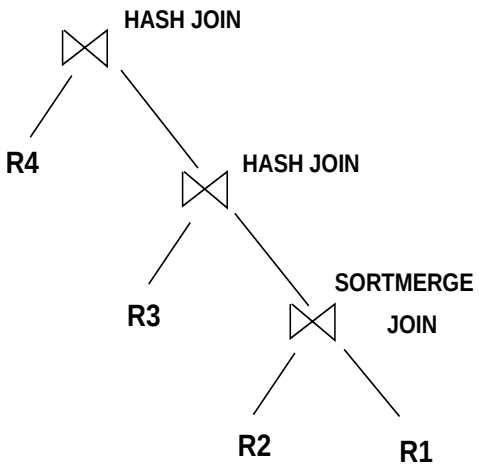
Main Assumptions

- No time-slicing overhead for TS resources
- Uniform TS resource usage

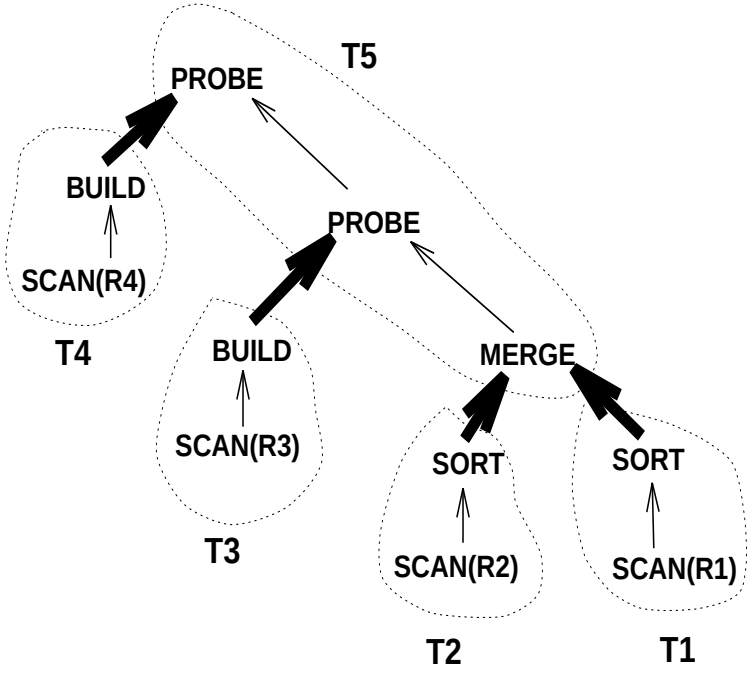


- Constant SS resource demands

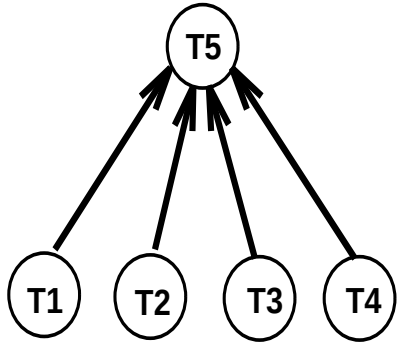
Query Plan Representation and Forms of Parallelism



JOIN TREE

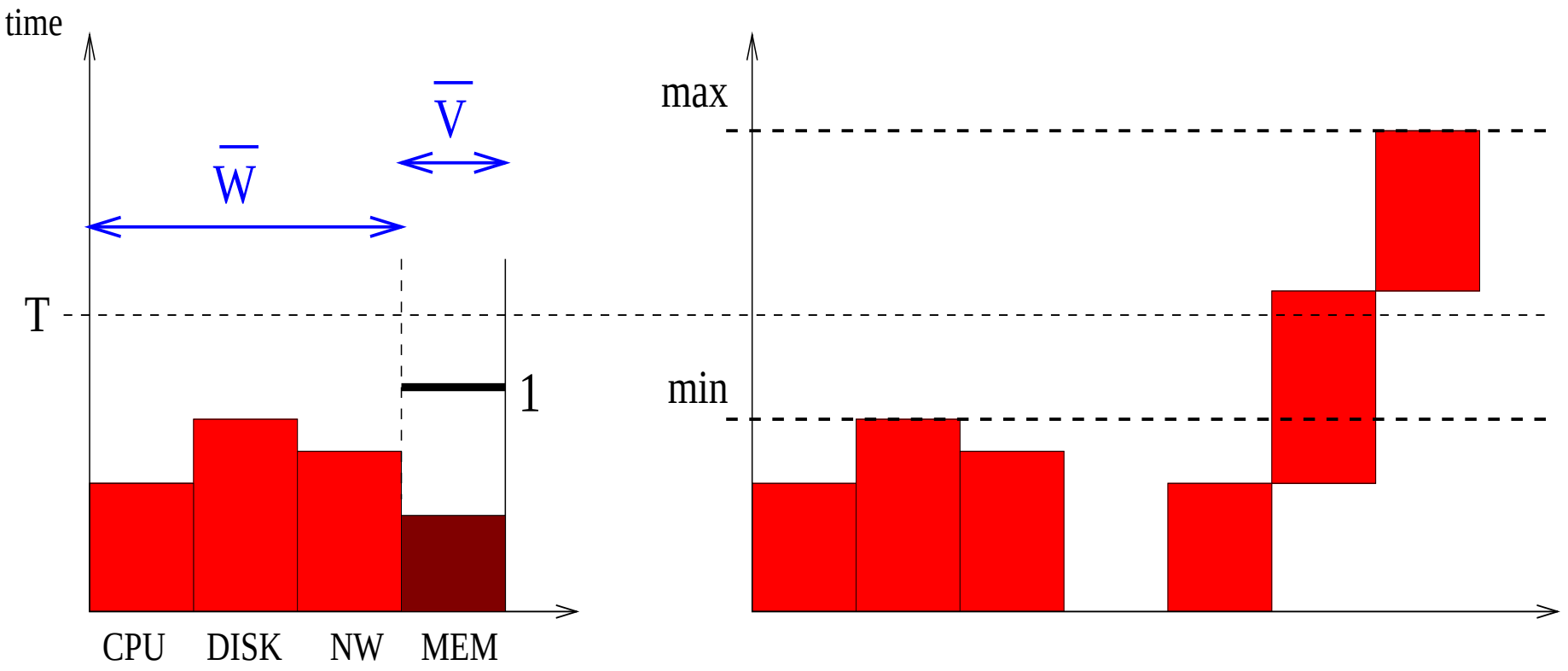


OPERATOR TREE



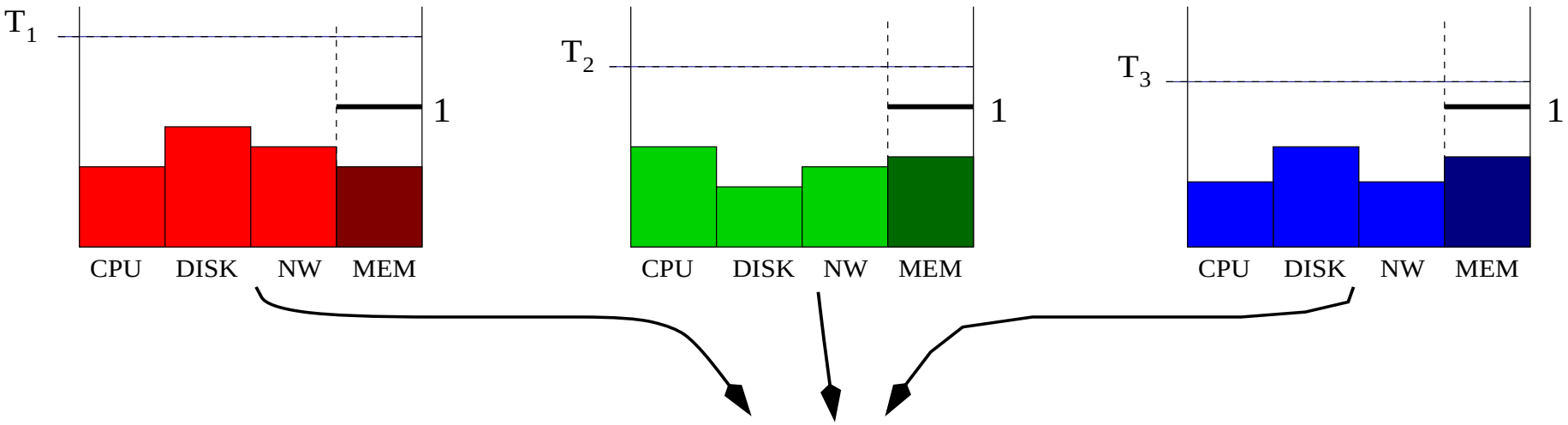
QUERY TASK TREE

Operator Cost Model

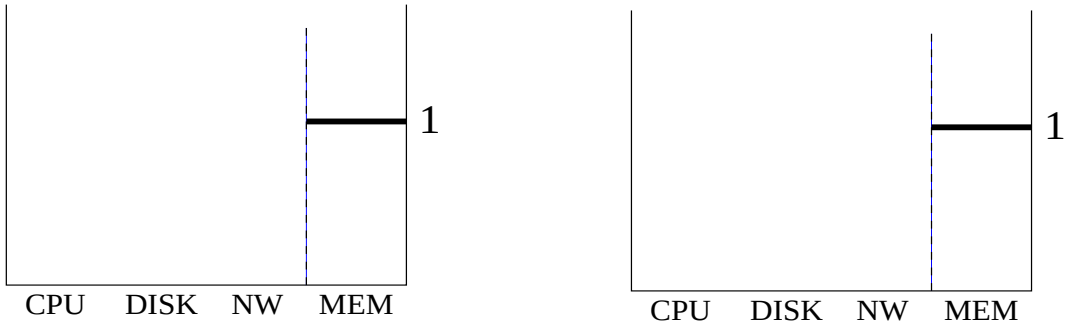


- Estimate *Work Vector* ($\overline{W}$) and *Demand Vector* ($\overline{V}$) components using conventional cost models and DB statistics
- Stand-alone execution time T = f($\overline{W}$, overlap), where $\overline{W}$ depends on $\overline{V}$

A Lower Bound on Response Time

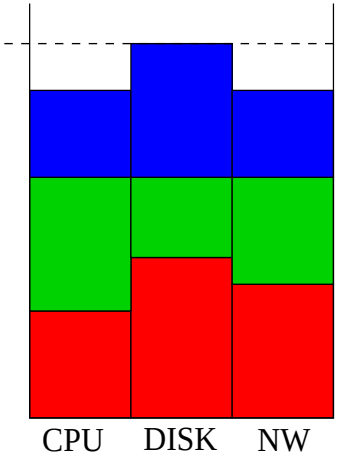


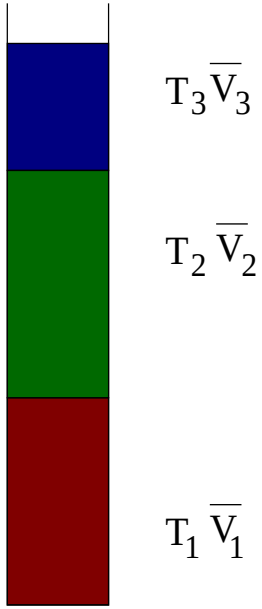
2 sites :



A Lower Bound on Response Time

- max { T_1 , T_2 , T_3 }: ‘‘critical path’’ of parallel execution





- $\dfrac{\text{max component } (\overline{W}_1 + \overline{W}_2 + \overline{W}_3)}{2}$

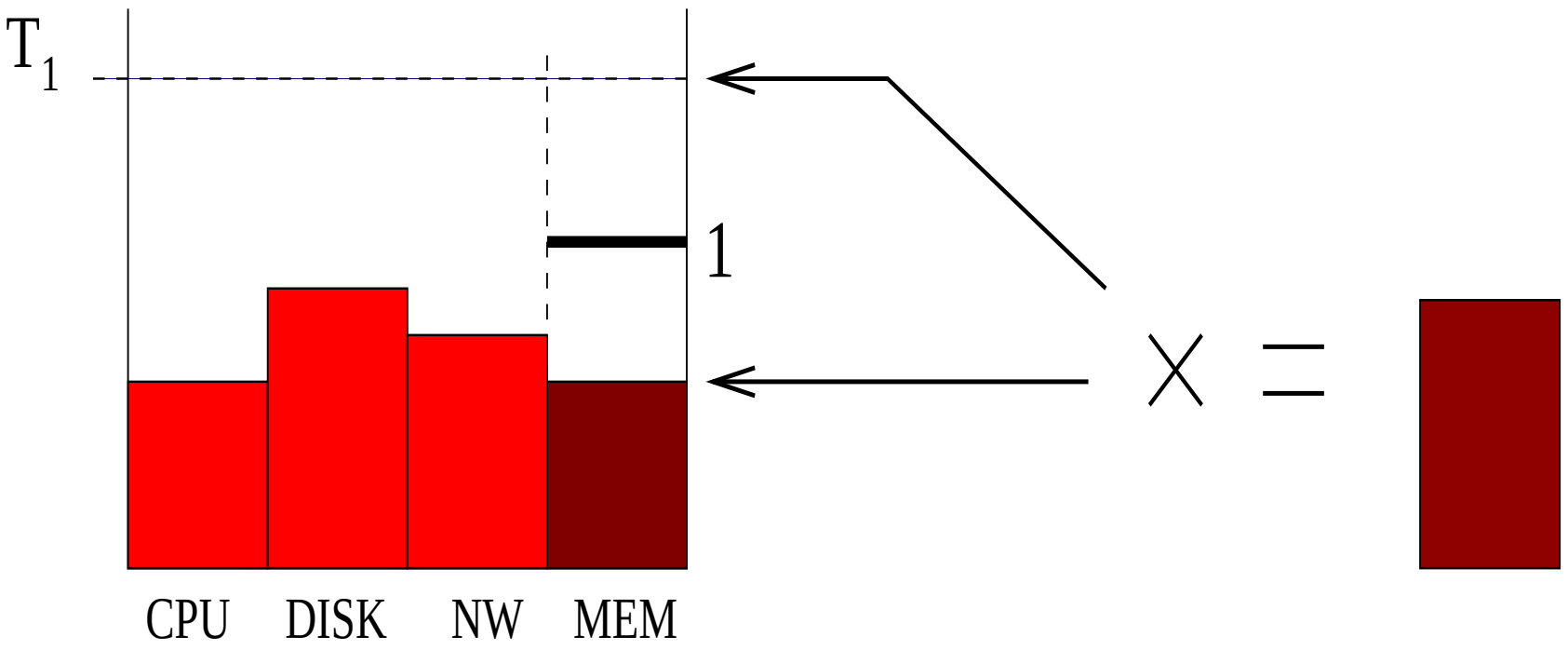
‘‘average work’’ per site (TS resources)

- $\dfrac{\text{max component } (T_1 \overline{V}_1 + T_2 \overline{V}_2 + T_3 \overline{V}_3)}{2}$

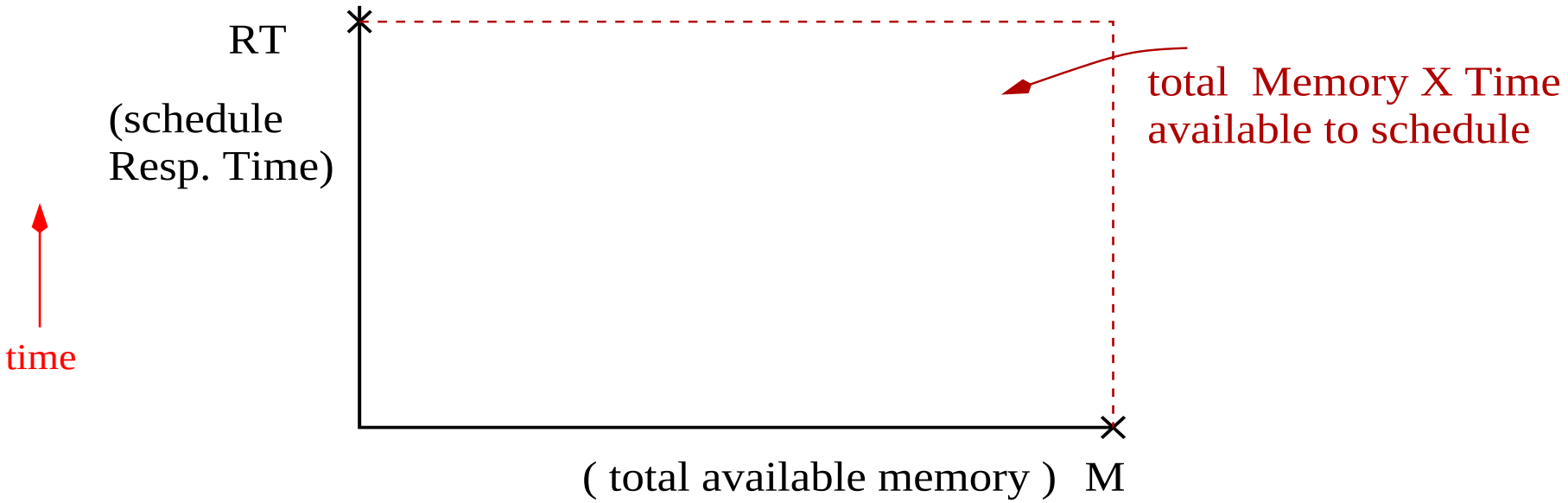
‘‘average volume’’ per site (SS resources)

LB = max { critical path , average work , average volume }

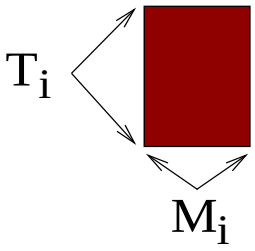
Operator Volume



More on the Volume Lower Bound



operator
"volumes" :



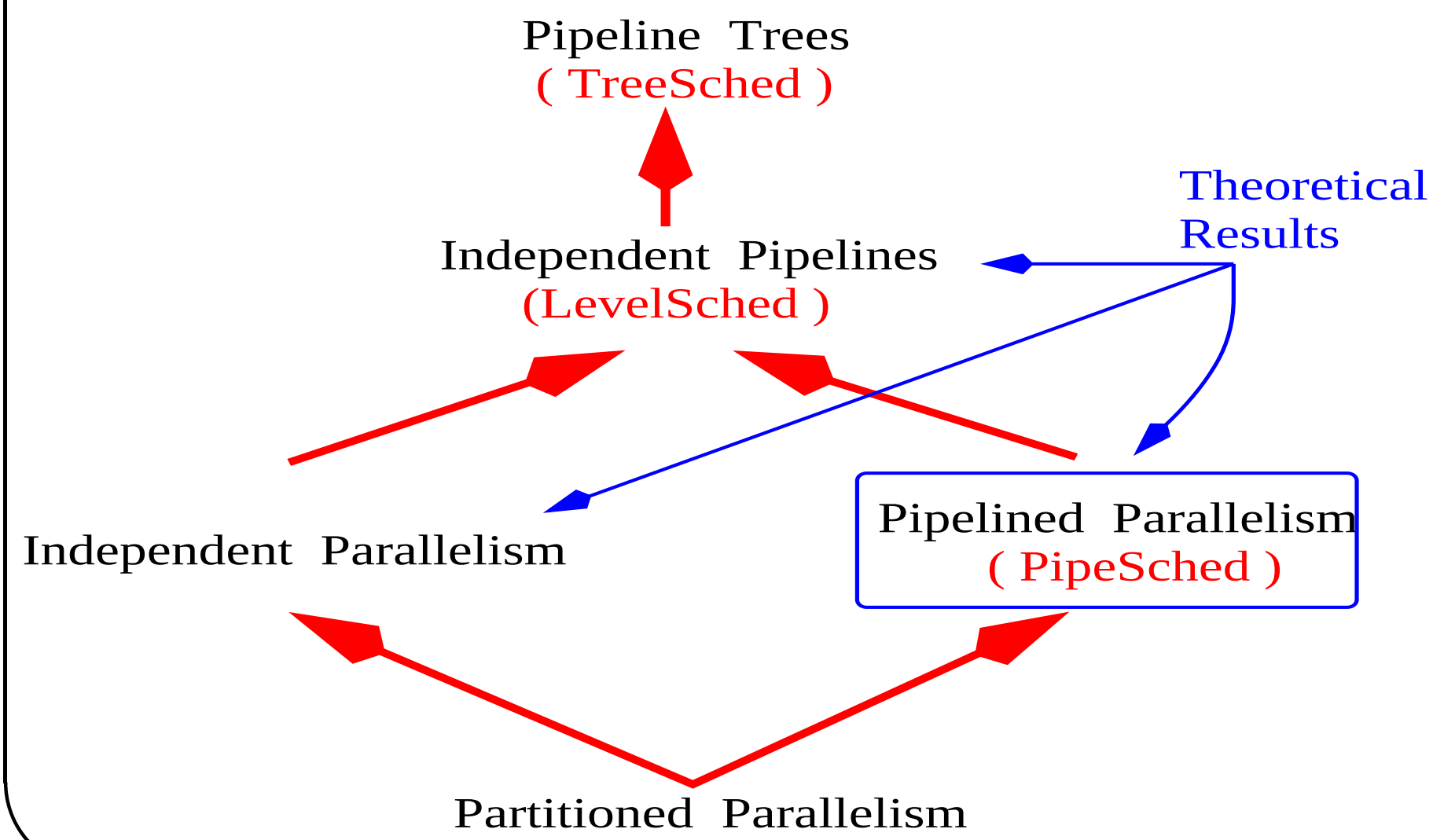
$$\sum T_i \, M_i \leq RT \cdot M$$

$$\Big|\Big|\Big|$$

$$RT \geq \frac{\sum T_i \, M_i}{M}$$

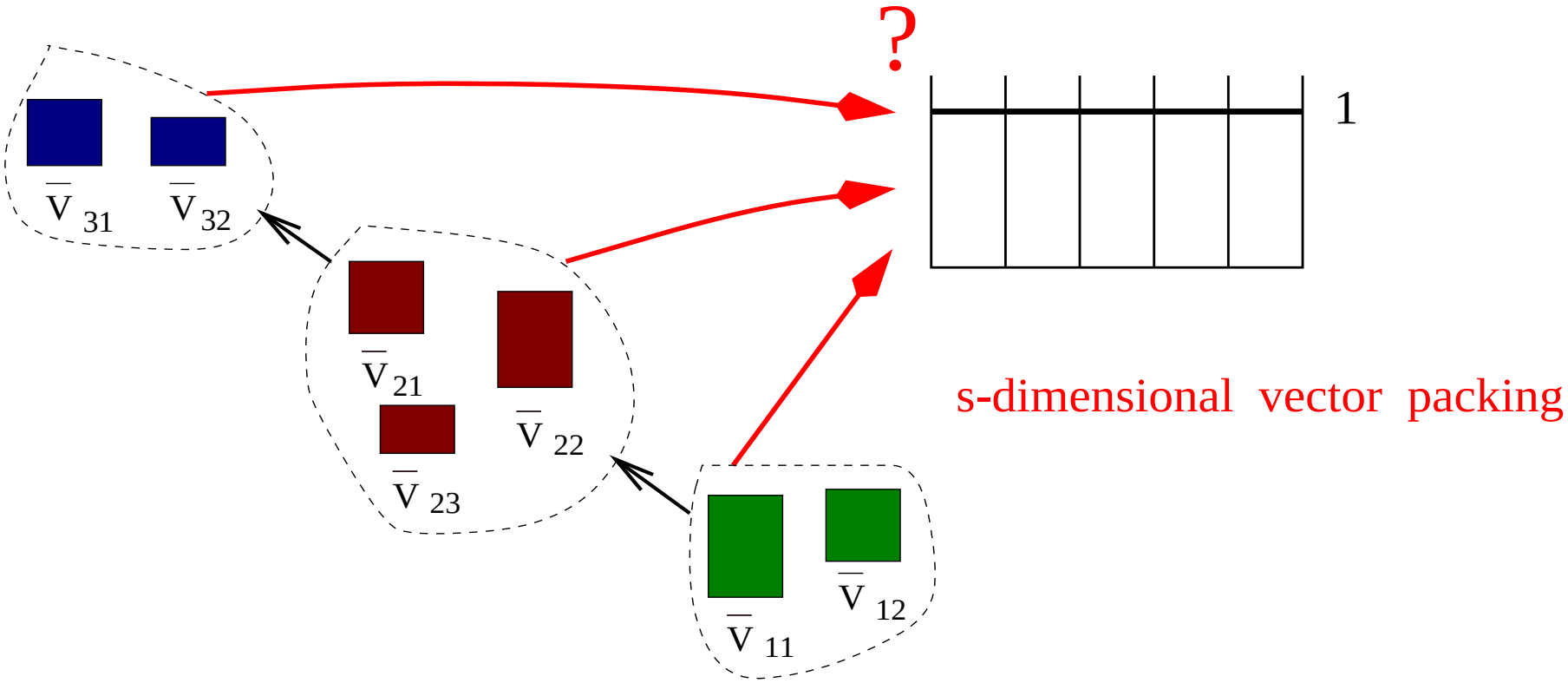
VOLUME
LB

Scheduling Algorithms



Pipelined Parallelism

- Co-scheduling requirement: NP-hard decision problem!



- λ-*granular clone*: all components of $\overline{V}$ are $\leq \lambda$ (≤ 1)
- Simple sufficient condition for schedulability of pipelines of λ-granular clones

Pipelined Parallelism (cont.)

Algorithm PipeSched: fast List-Scheduling Heuristic (à la Graham) for a schedulable pipeline

1. Sort all clones in decreasing order of

$$work\ density\ =\ \max\{\overline{W}\}\ /\ \max\{\overline{V}\}$$

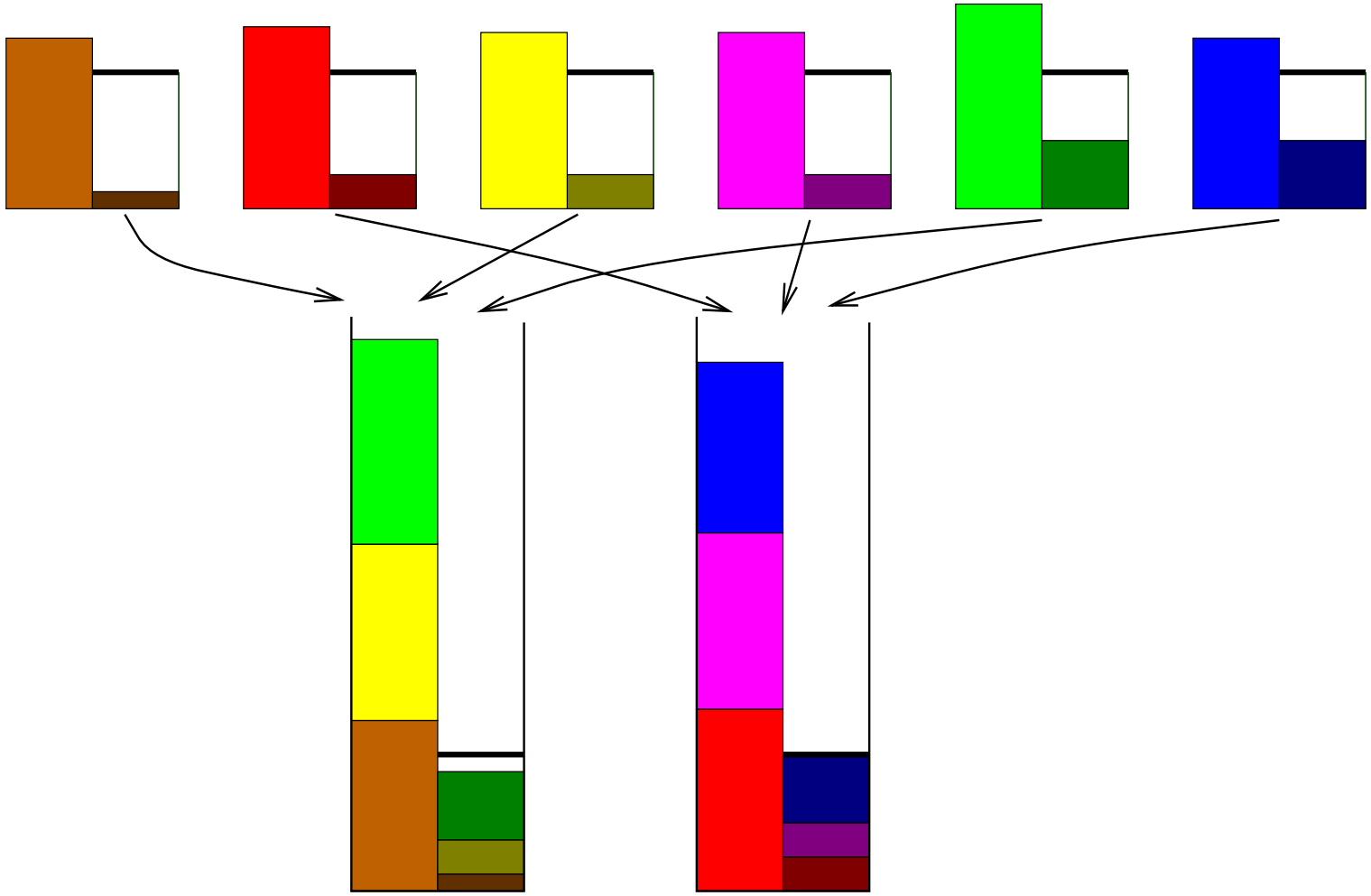
2. In that order, place each clone in the eligible site with the least amount of work

Theorem: Given a schedulable λ-granular pipeline, PipeSched produces a schedule that is within $d\left(1+\frac{s}{1-\lambda}\right)+1$ of optimal

A Simple Example ($d = 1,\ s = 1$)

1. Sort

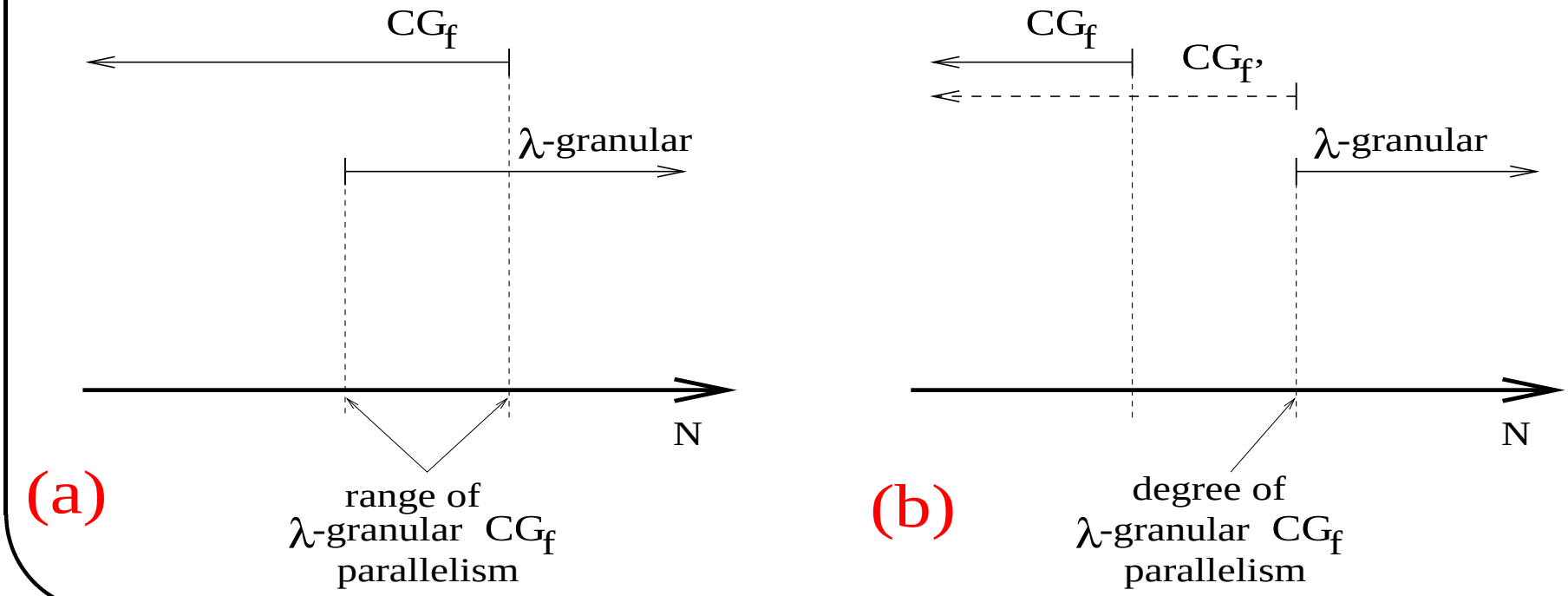
2. Place



Partitioned Parallelism

Degree of Parallelism? Tradeoff...

- Finer clones: satisfy SS resource demands, better load balancing
- Coarser clones: reduce communication costs (comm $\leq f \cdot$ work)
- Want to ensure λ-granularity for schedulability



Independent Pipelines

Algorithm LevelSched**:**

1. Group the pipelines into maximal schedulable subsets
2. Schedule each subset in a separate level ("shelf") using PipeSched

Theorem: LevelSched is within $O\left(\frac{s(d^2+s)}{1-\lambda}\right)$ of optimal

Cost Model Experiments

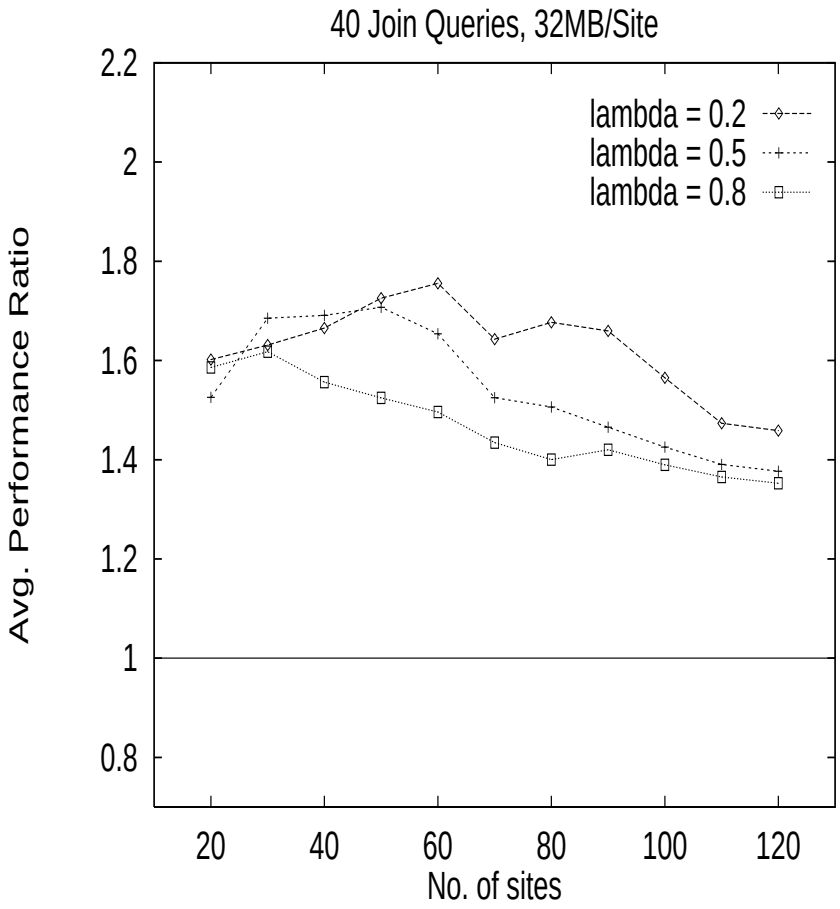
- TreeSched vs. lower bound on optimal response time (LB)
- *Average* effectiveness based on *cost model*

Experimental Assumptions

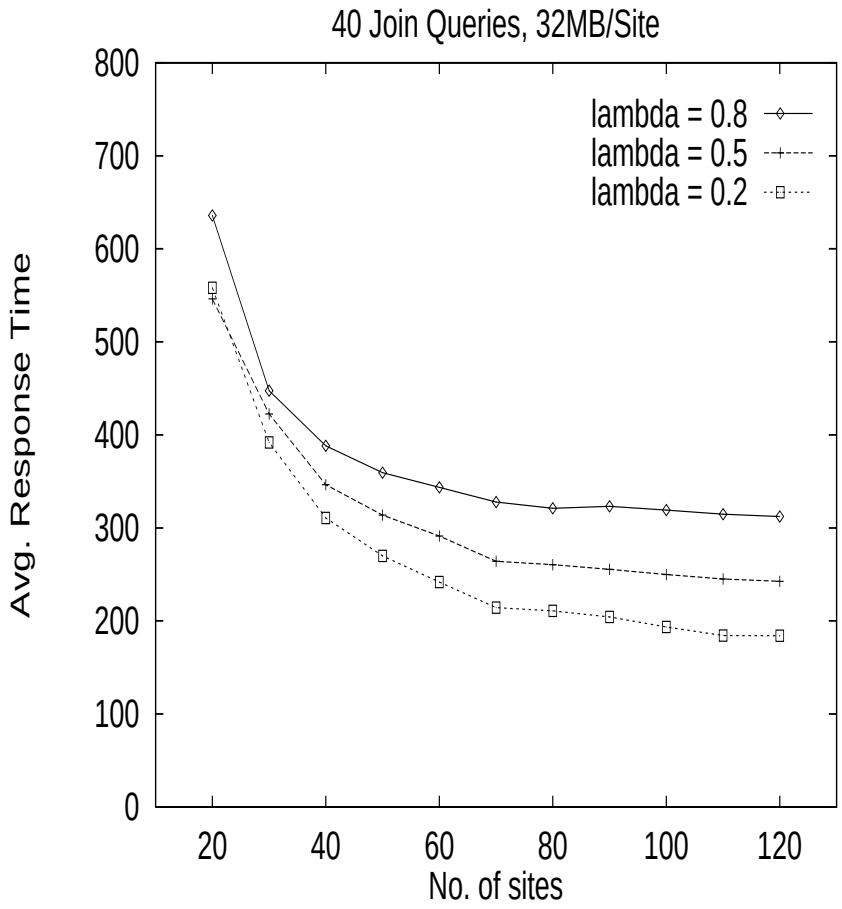
- Only simple hash-joins (one-phase pipelines)
- No execution skew (except for startup cost)
- Uniform resource overlap (single parameter)

Cost Model Experiments (cont.)

- TreeSched / LB

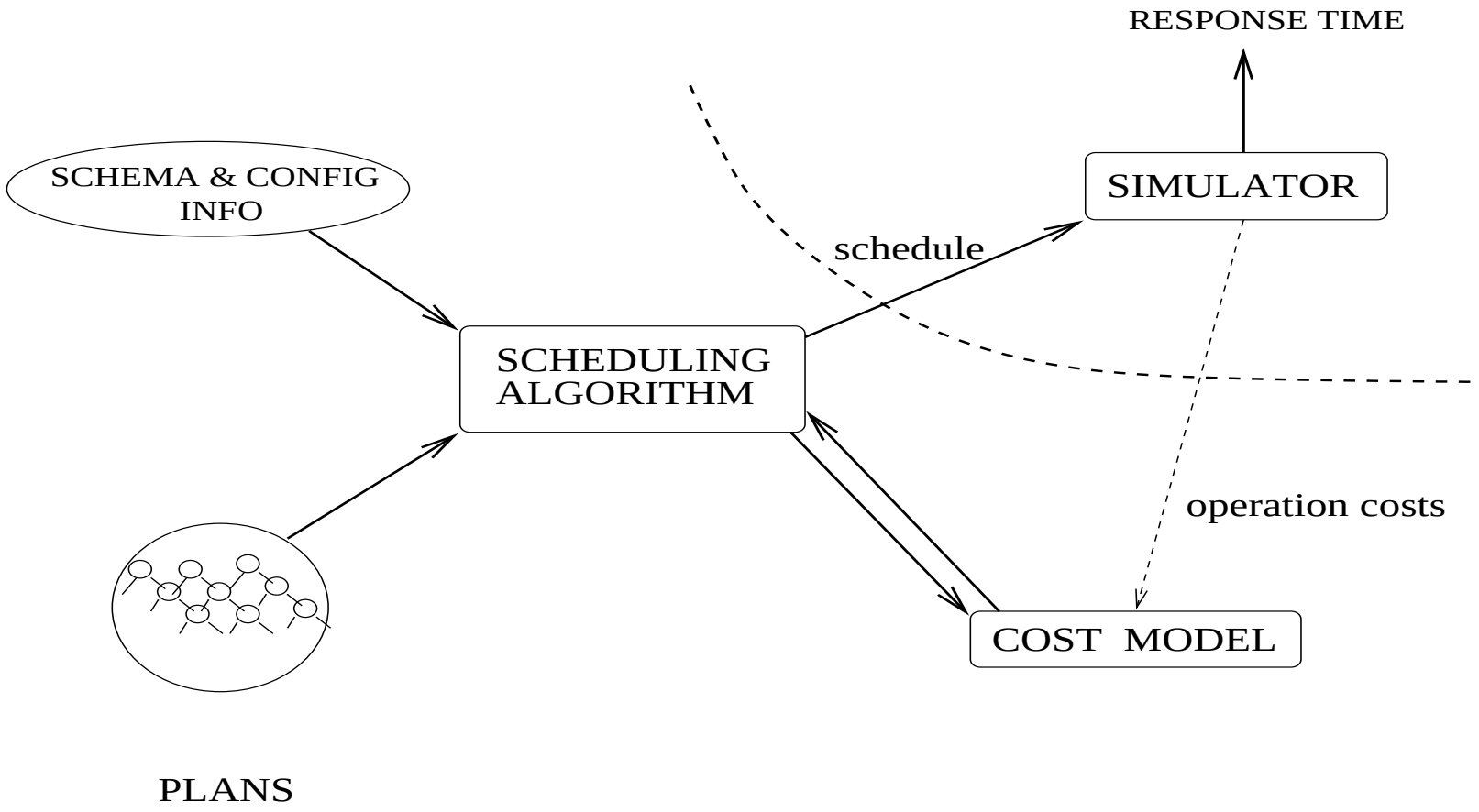


- TreeSched



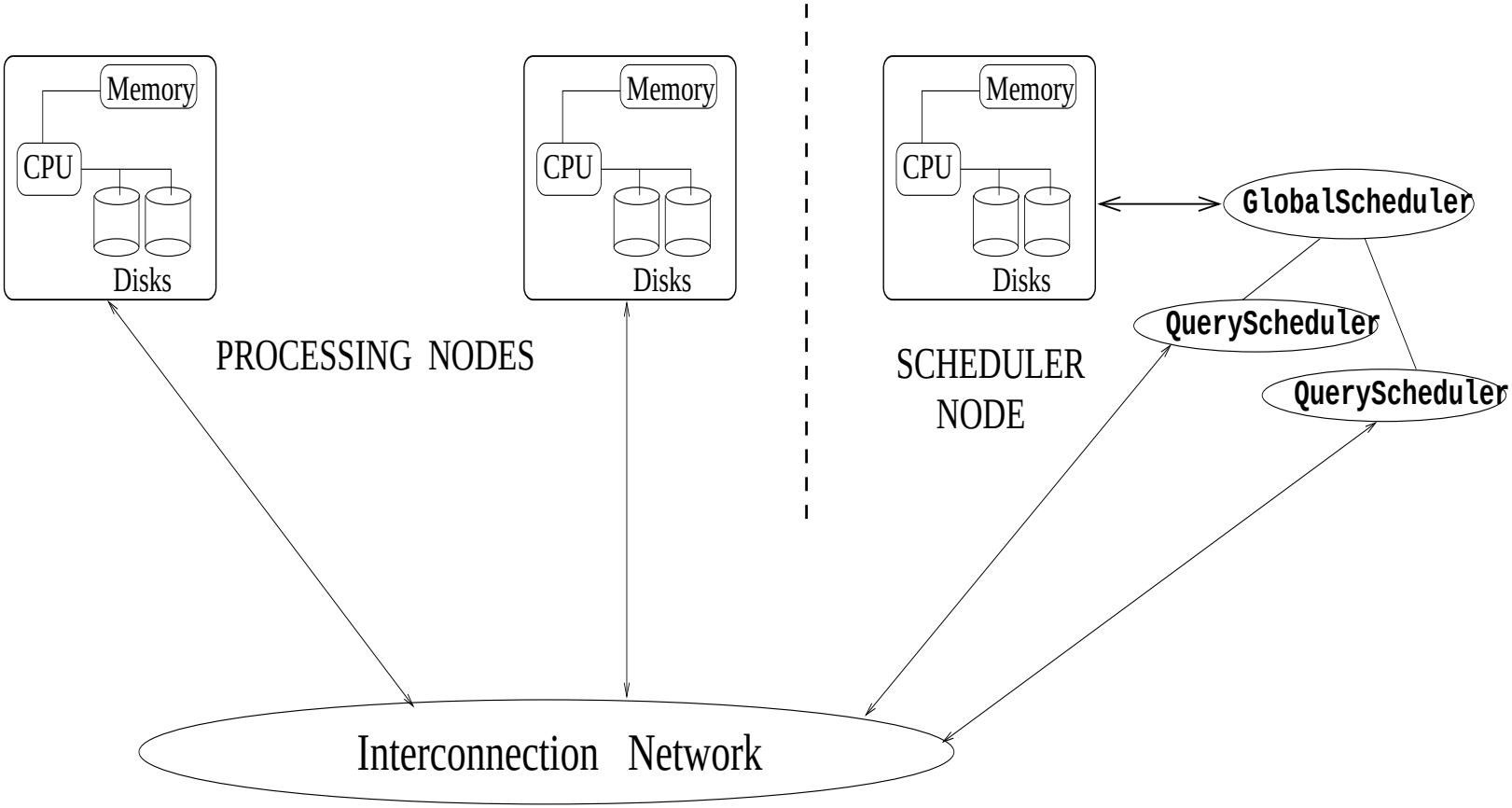
Simulation Experiments

- LevelSched on top of the ZetaSim detailed simulation model



Simulation Experiments (cont.)

- ZETASIM overview



Simulation Experiments: Testbed

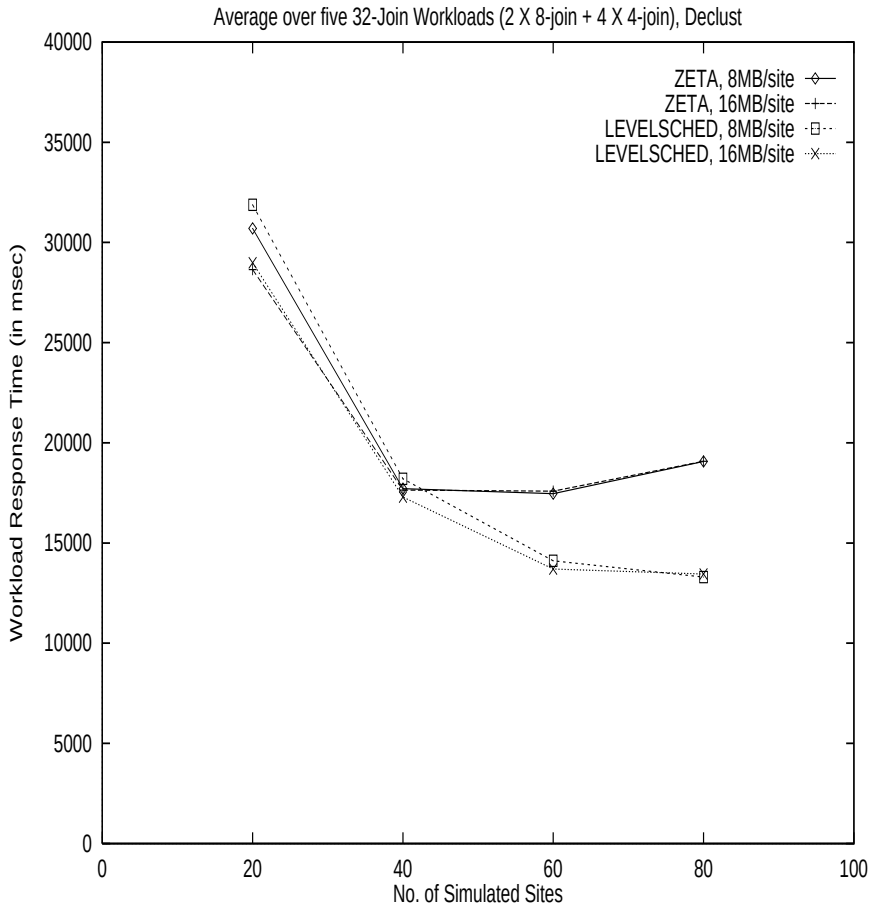
- LevelSched vs. (default) Zeta ("schedule join where build relation lives")
- Workload: Collection of right-deep join plans. Total no. of joins 8, 16, 32, and 64, spread across various combinations of query sizes (e.g., 32 = 2 X 8-join + 4 X 4-join)
- Metric: elapsed simulation time to schedule completion

Simulation Experiments: Testbed (cont.)

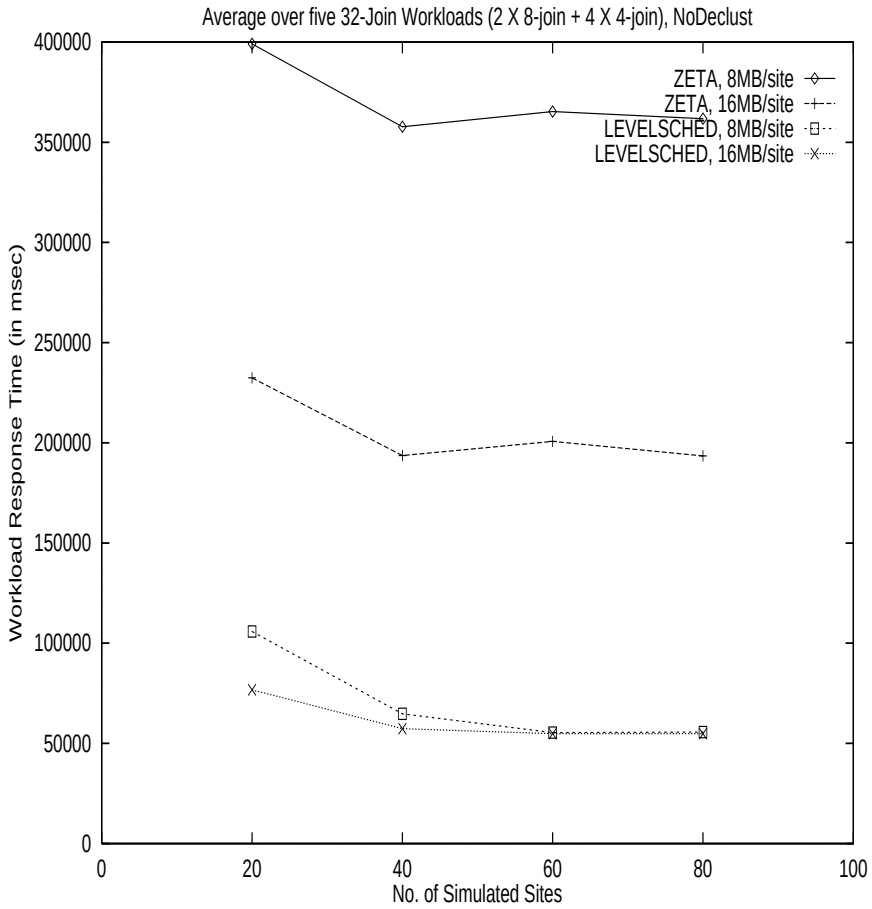
- Various data placement schemes:
  1. **Declust** (**NoDeclust**): full (no) relation declustering
  2. **Declust-1/4** (**NoDeclust-1/4**): as above, but on 1/4th of sites
  3. **Random**: degree of declustering and home randomly chosen
  4. **QueryBasedDeclust**: degree of declustering according to relation size, no overlaps between joins

Simulation Results

- Declust

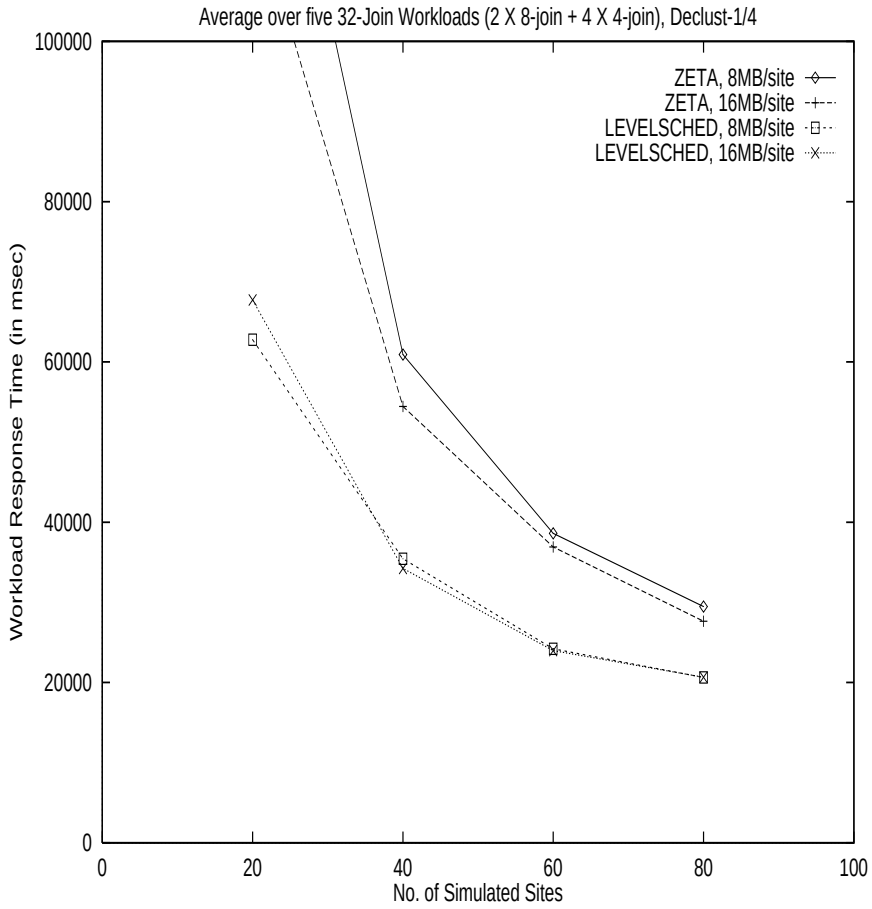


- NoDeclust

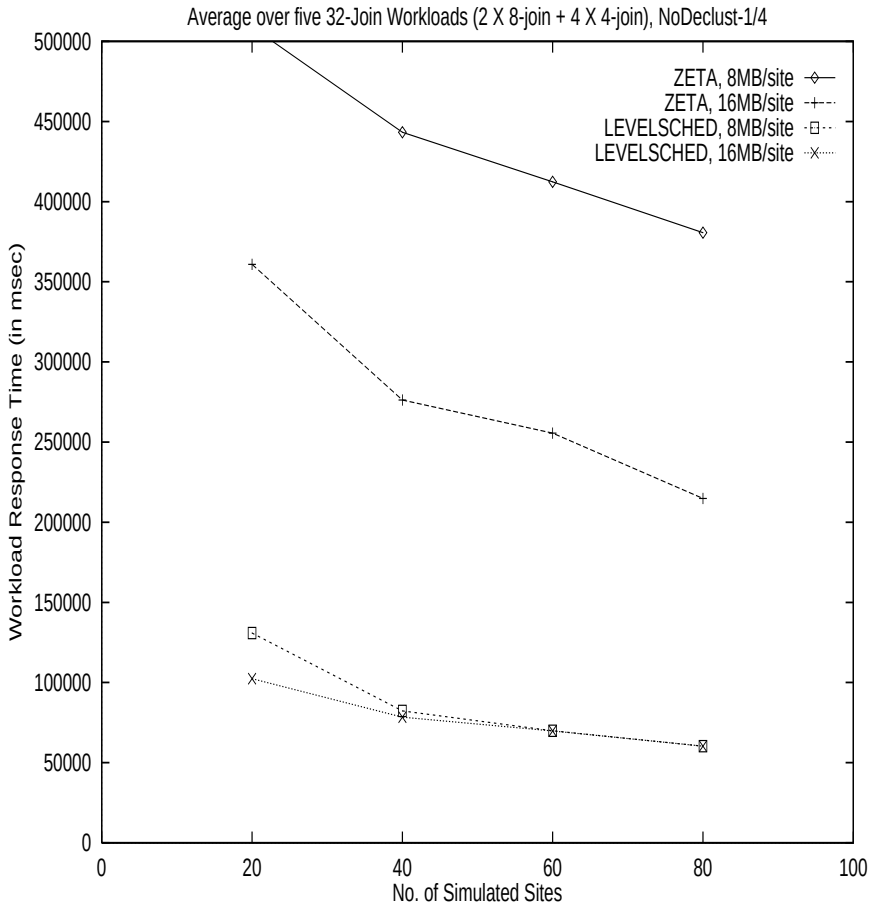


Simulation Results (cont.)

- Declust-1/4

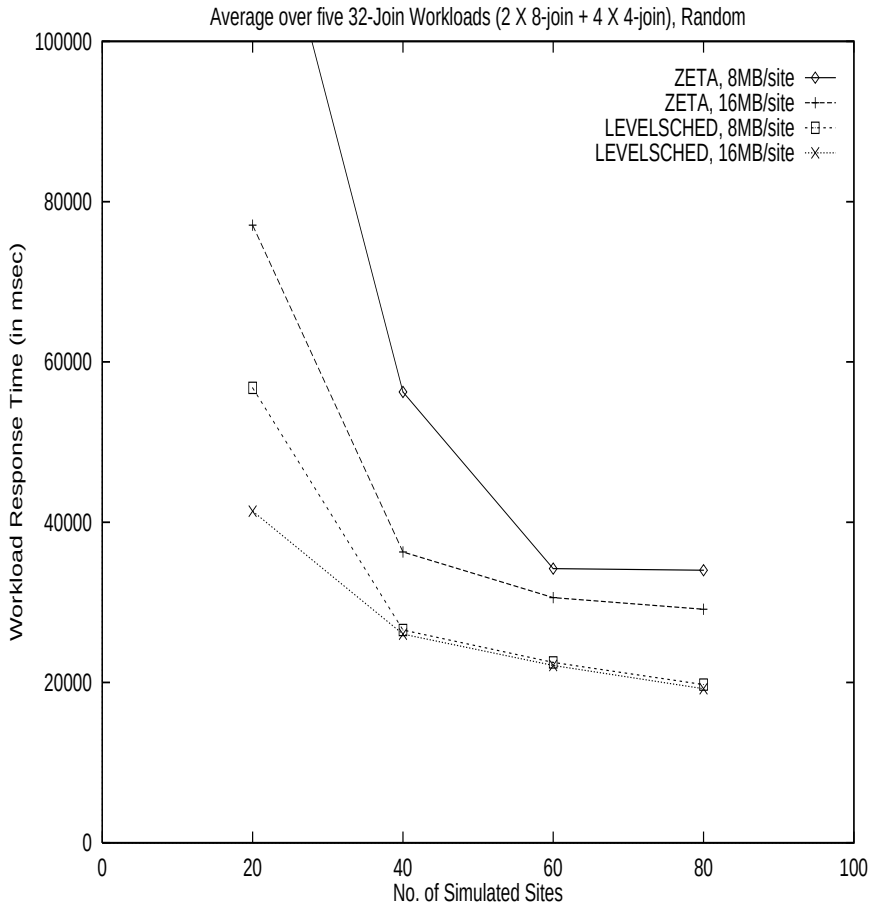


- NoDeclust-1/4

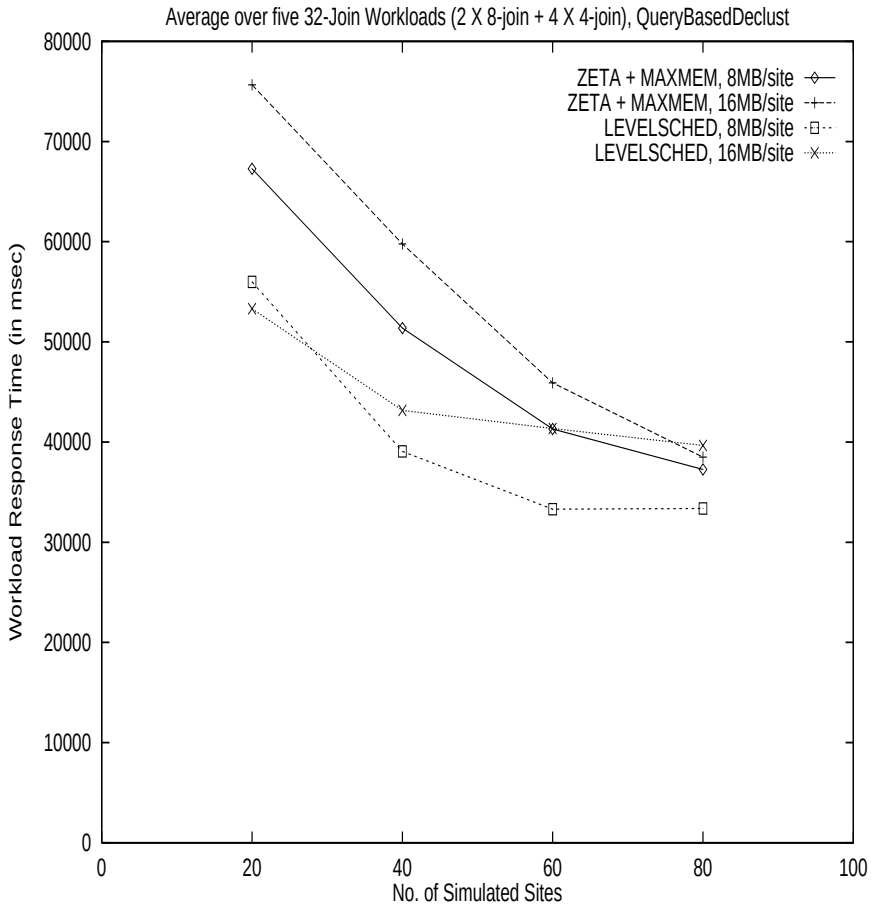


Simulation Results (cont.)

- Random

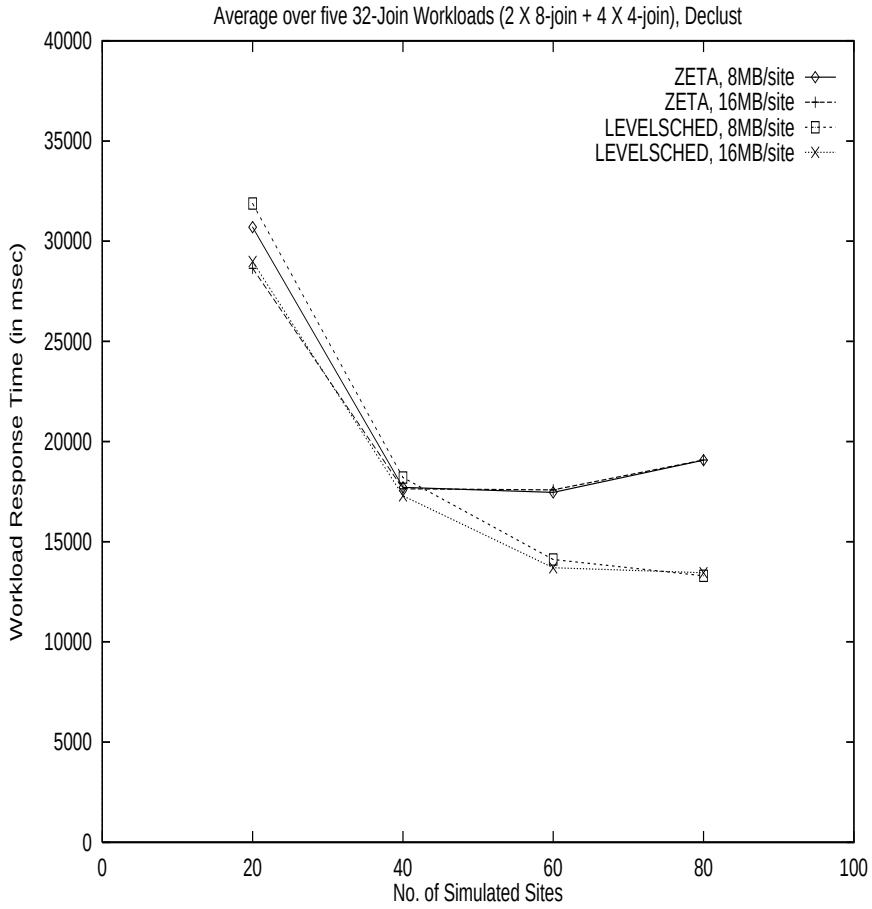


- QueryBasedDeclust

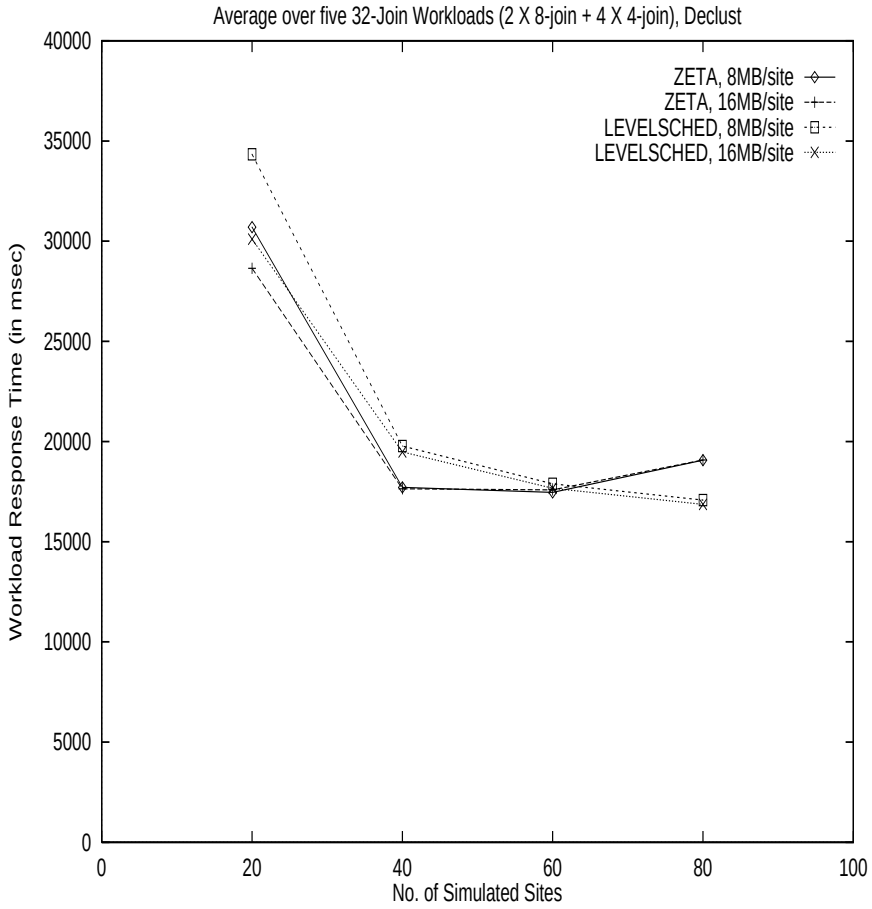


Effect of Coarse Granularity Parameter f : Declust

- $f = 0.01$

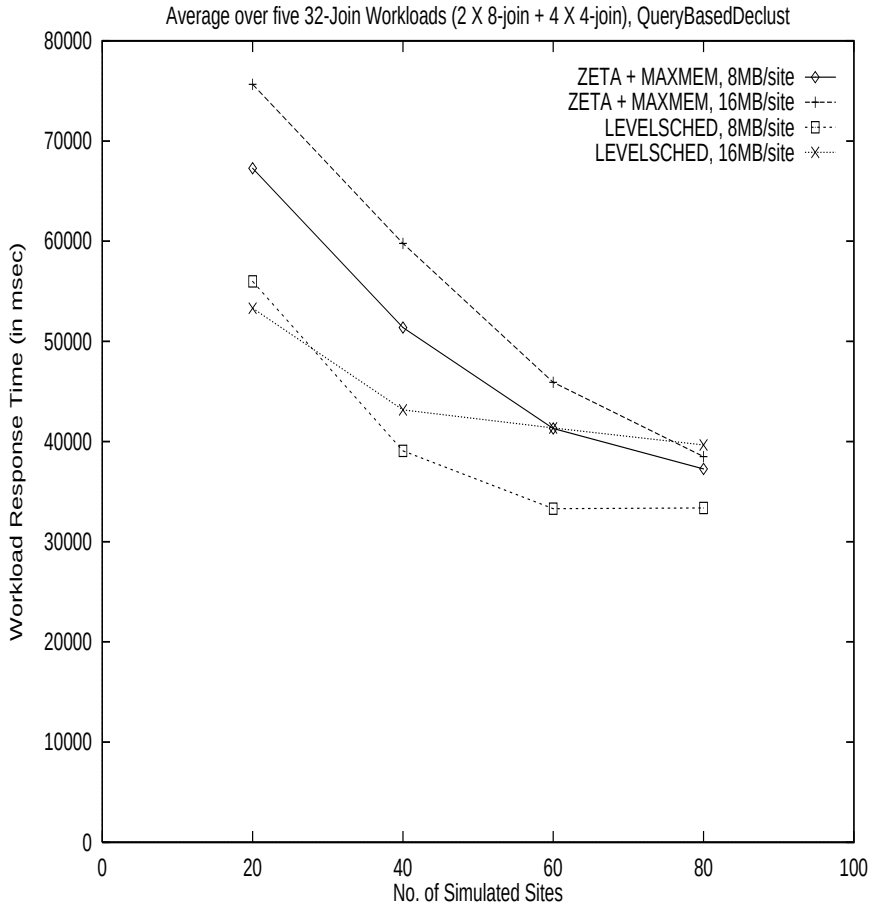


- $f = 0.001$

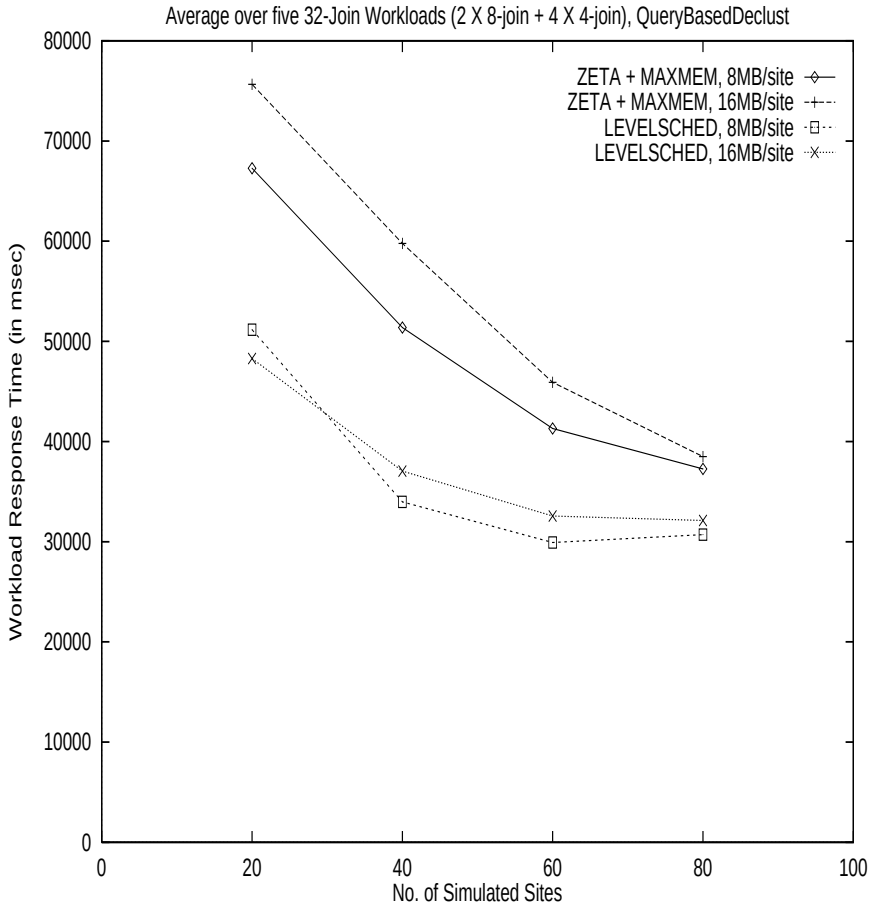


Effect of Granularity Param. f : QueryBasedDeclust

- $f = 0.01$

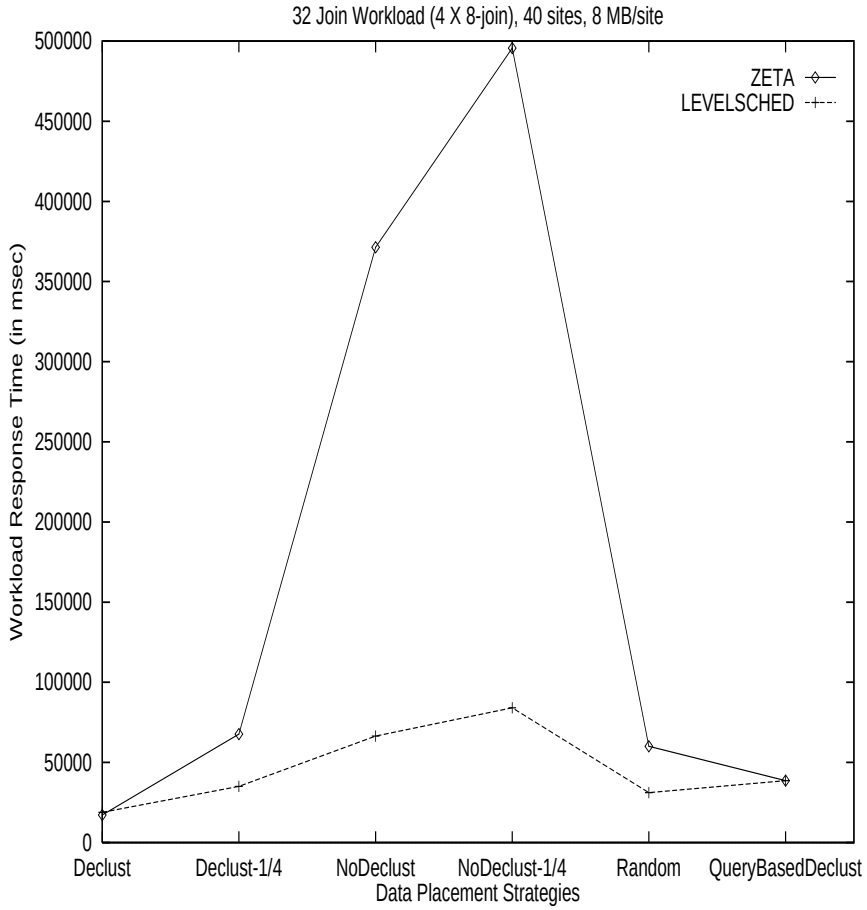


- $f = 0.001$

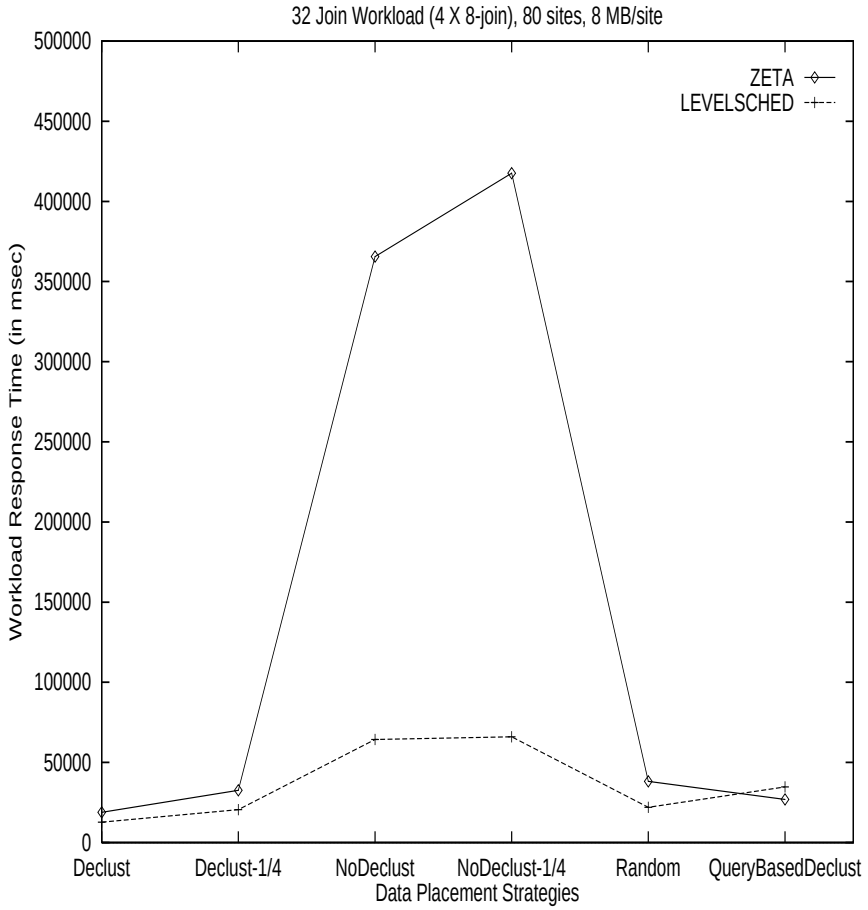


Effect of Initial Data Placement

- 40 sites, 8MB/site

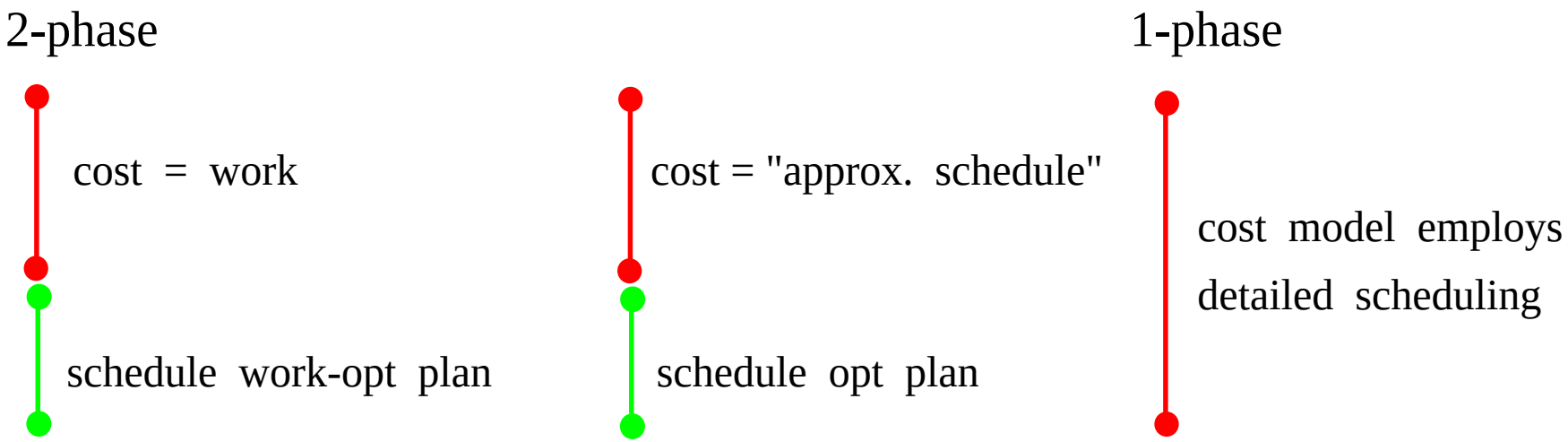


- 80 sites, 8MB/site



Parallel Query Optimization

Cost Model: "plan quality" by abstracting execution environment



- [Ganguly et al., PODS'96]: max { critical path, average work }
- Our results: *average volume* captures the effects of SS resources
- $<$ *critical path, average work , average volume* $>$: crucial bulk parameters of a parallel plan for efficient optimization

Conclusions

- Comprehensive formal approach to multi-dimensional query scheduling in parallel systems
- Efficient, near-optimal heuristic algorithms
- Experimental results (cost model and detailed simulation) verified effectiveness
- Novel, effective cost model for parallel query optimization capturing TS & SS dimensions

Ongoing and Future Work

- *Malleable* memory demands
- More complex pipelining strategies that work with limited memory
- Different degrees of "Time-Shareability" (e.g., disk vs. CPU)
- Validate scheduling algorithms and optimizer cost model in a real system

Prior Work

(a) DB Research

- One-dimensional model of resource allocation (scalar "work" metric)
- Limited forms of parallel plans (often no bushy tree parallelism)

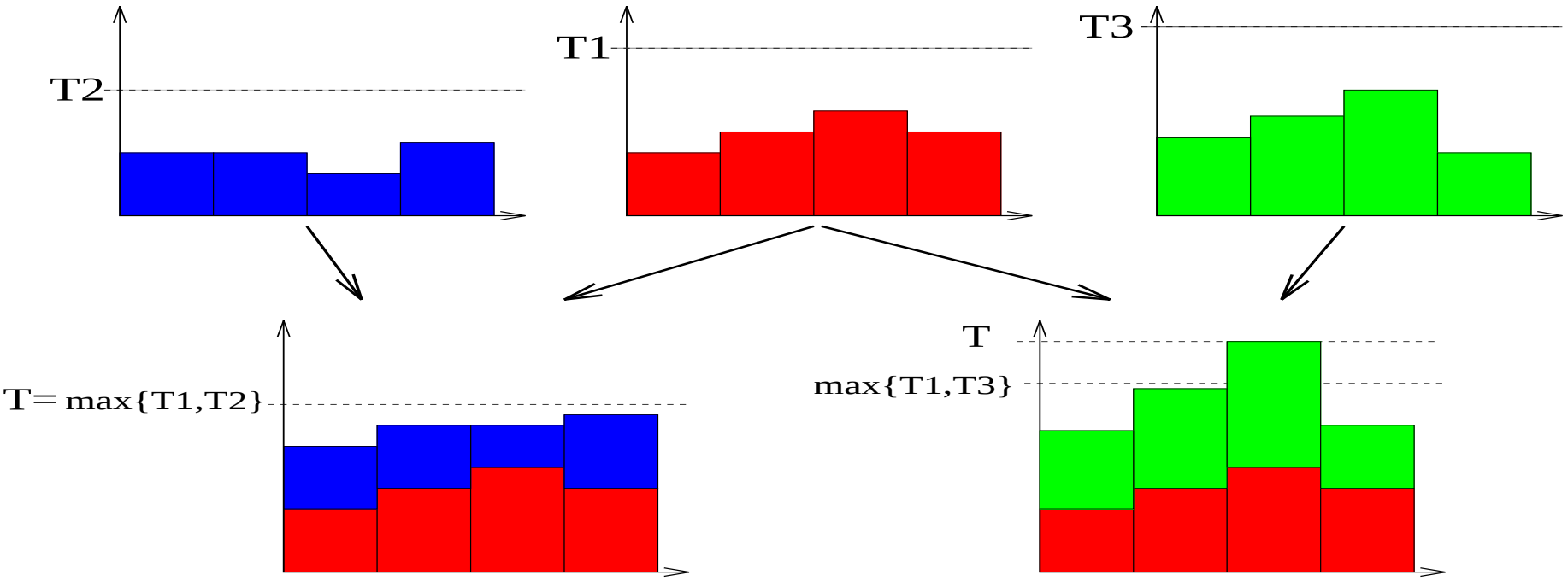
Schneider and DeWitt [VLDB'90], Lo et al. [SIGMOD'93], Shekita et al. [VLDB'93], Hsiao et al. [SIGMOD'94], Hasan and Motwani [VLDB'94], Chekuri et al. [PODS'95]

(b) Deterministic Scheduling Theory

- Multiprocessor scheduling: mapping DAGs of computation-intensive tasks on parallel machines
- One-dimensional models of parallelization or globally accessible resources

Concurrent Execution

- Feasible *only if* total SS requirements ≤ 1
- Effects of TS resource sharing



- Response Time: slowest clone or most heavily congested resource

Independent Parallelism

- *Compatible clones*: total SS resource demands ≤ 1
- Schedule: partition clones into compatible subsets; map onto sites
- Response time in a compatible partition is determined by TS components
- Algorithm OpSched: simple near-optimal heuristic

Query Task Trees: Handling Data Dependencies

Algorithm TreeSched:

- Maintain dynamic list of enabled ("ready") pipelines starting from leaves
- Ensure resource dependencies are maintained (e.g., `build` $\rightarrow$ `probe`)
- Same ideas can be applied for scheduling tasks from multiple queries *on-line* (inter-query parallelism)

Cost Model Experiments: Testbed

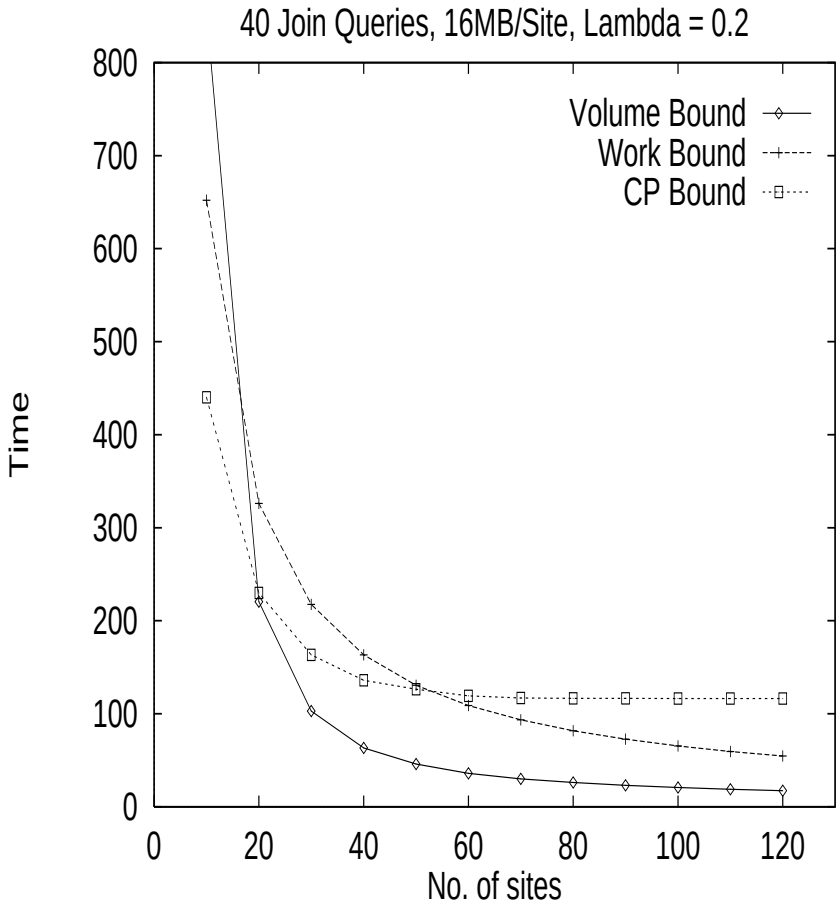
- Sites with d=3 (CPU, Disk, NW interface) and s=1 (Memory)
- Tree queries 10,20,30,40,50 joins (20 random query graphs/size)
- Memory/site: 16MB - 64MB, λ = 0.1 - 0.9
  Overlap: 10% - 70%, Coarse Granularity: 0.3 - 0.9
- Performance metrics: Average performance ratio, Average schedule response time

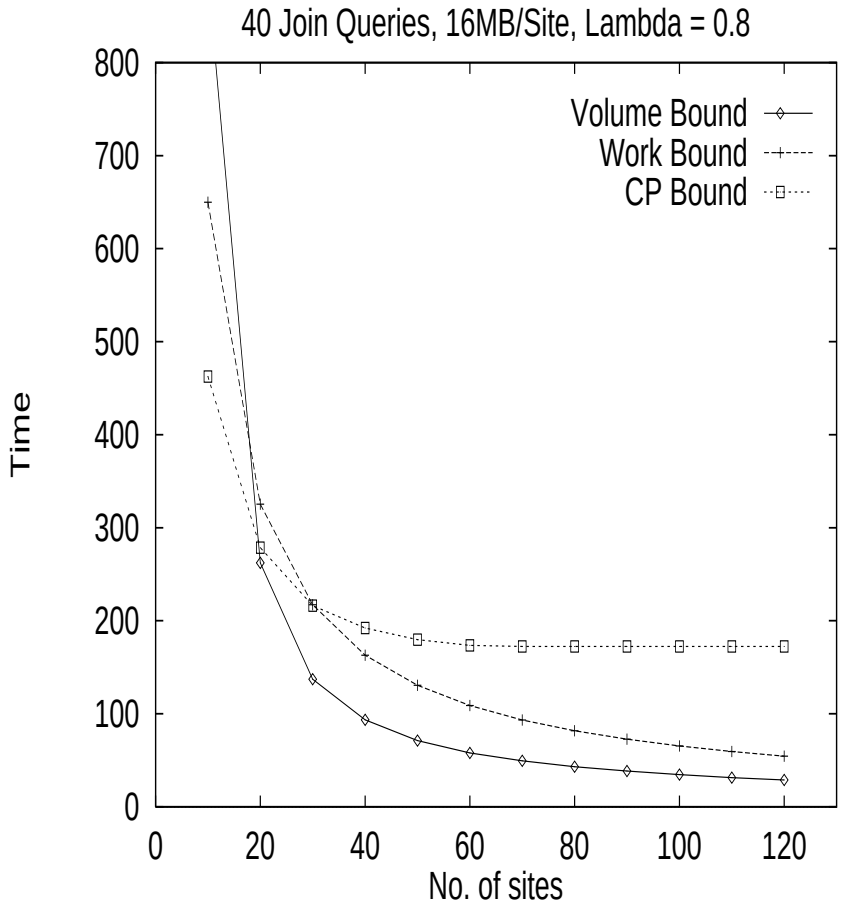
Cost Model Parameter Settings

Configuration/DB Catalog Params	**Value**
Number of Sites	10 – 120
CPU Speed	10 MIPS
Memory per Site	16 – 64 MB
Effective Disk Service Time per page	5 msec
Startup Cost per site (α)	15 msec
Network Transfer Cost per byte (β)	0.3 μsec
Fudge Factor (F)	1.4
Tuple Size	128 bytes
Page Size	32 tuples
Relation Size	10^4 – 10^6 tuples
CPU Cost Parameters	**No. Instr.**
Read Page from Disk	5000
Write Page to Disk	5000
Extract Tuple	300
Hash Tuple	100
Probe Hash Table	200

Experimental Results (more...)

- Lower Bound Components, $\lambda = 0.2$, 0.8





Simulation Parameter Settings - I

Configuration/Site Parameters	Value
Number of Sites	20 – 80
CPU Speed	20 MIPS
Number of Disks per Site	2
Memory per Site	8 – 16 MB
Page Size	8 KB
Latency for 8K Message	1.8 msec
Disk Seek Factor	0.617
Disk Rotation Time	16.667 msec
Disk Settle Time	2.0 msec
Disk Transfer Rate	3.09 MB/sec
Disk Cache Context Size	4 pages
Disk Cache Size	8 contexts
Disk Cylinder Size	83 pages
Buffer Manager	3-level LRU

Simulation Parameter Settings - II

CPU Cost Parameters	No. Instr.
Initiate Select	20000
Initiate Join	40000
Initiate Store	10000
Terminate Select	5000
Terminate Join	10000
Terminate Store	5000
Read Tuple	300
Write Tuple into Output Buffer	100
Probe Hash Table	200
Insert Tuple in Hash Table	100
Hash Tuple Using Split Table	500
Apply a Predicate	100
Copy 8K Message to Memory	10000
Message Protocol Costs	1000

Simulation Schema & Workload Parammeters

Schema/Workload Parameters	Value
Tuple Size	200 bytes
Base Relation Size	1K – 100K tuples
Join Attribute Distribution	Uniform
Workload Size (Total No. of Joins)	8, 16, 32
Right-Deep Query Size	2, 4, 8 joins
Coarse Granularity Parameter (f)	0.001 – 0.1
Memory Granularity Parameter(λ)	0.15 – 0.25
Join Probability	1.0
Projection Factor	0.5