Multi-dimensional Resource Scheduling for Parallel Queries

Minos N. Garofalakis
Yannis E. Ioannidis
University of Wisconsin – Madison

Outline

- Introduction and Motivation
- Related Work
- Problem Formulation and Query Execution Model
- List Scheduling (LS) Heuristic Algorithms
- Experimental Results
- Ongoing and Future Work

Introduction and Motivation

The Problem

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

Why?

- SN paradigm is the most scalable
- 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)

NP-hard, even for "one-dimensional" operators and no communication costs

Goals General problem formulation; Fast and effective scheduling heuristics

Related 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]

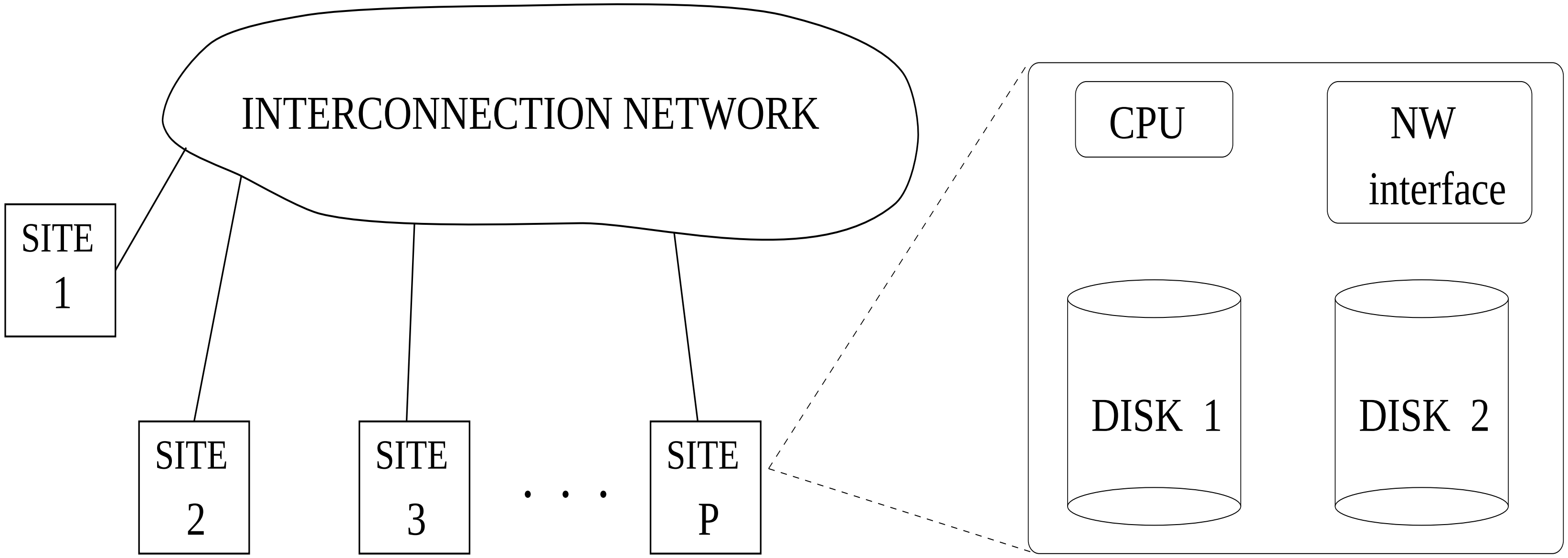
Exception: 2-d XPRS scheduler, Hong [SIGMOD'92] (Shared-Memory!)

(b) Deterministic Scheduling Theory

- One-dimensional models of parallelization or globally accessible resources

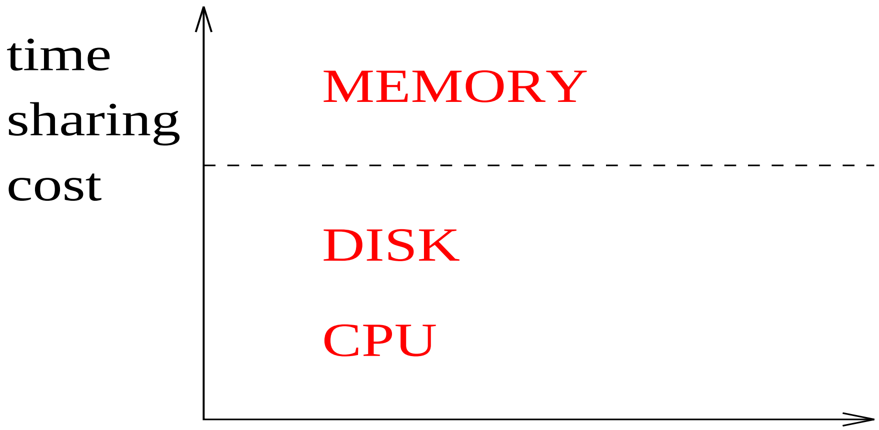
System Model

- Identical sites consisting of d *time-shareable* resources



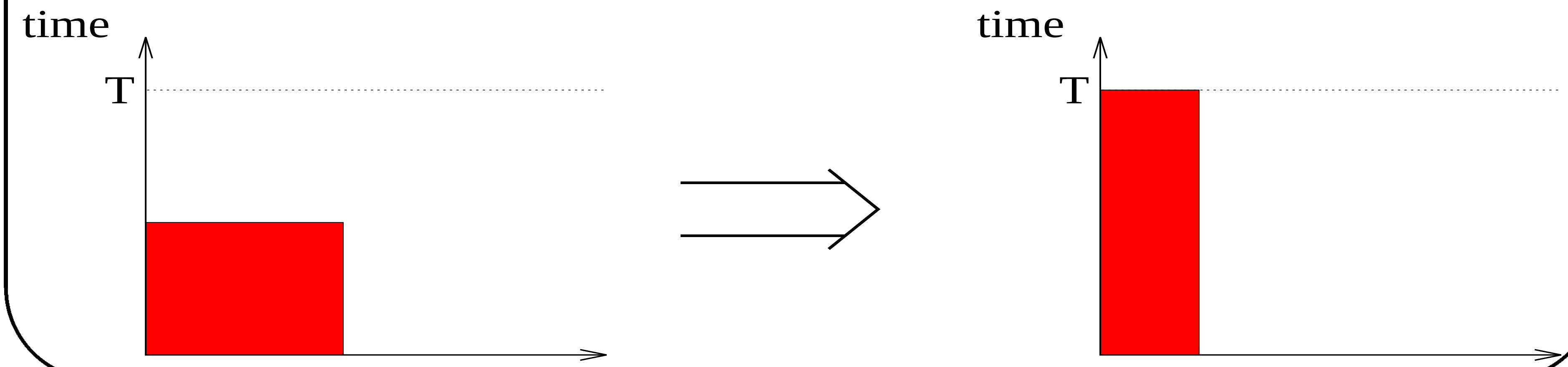
Main Assumptions

- No memory limitations

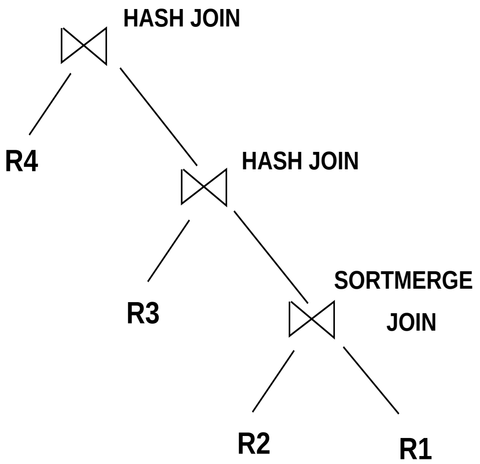


- No time-sharing overhead

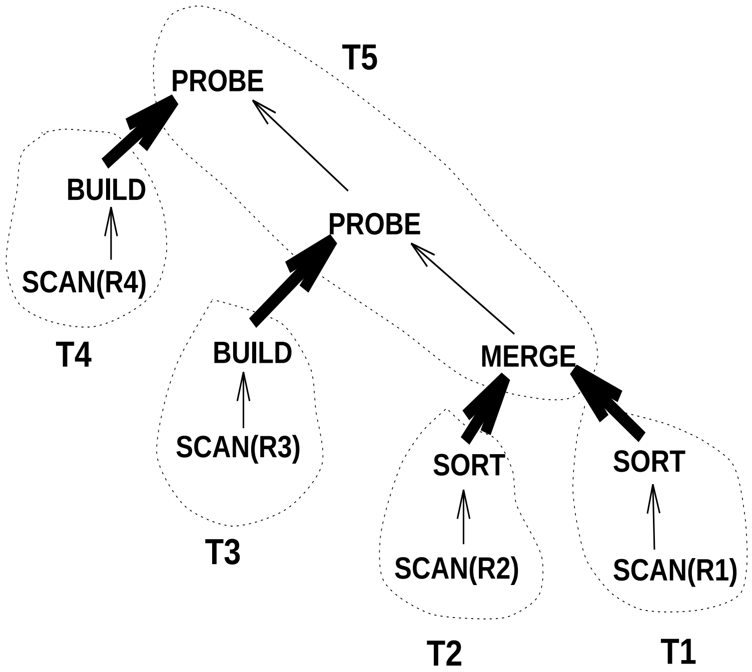
- Uniform resource usage



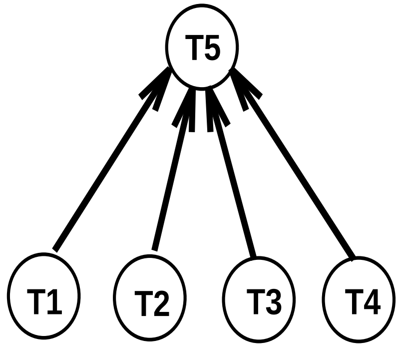
Query Plan Representation and Forms of Parallelism



JOIN TREE

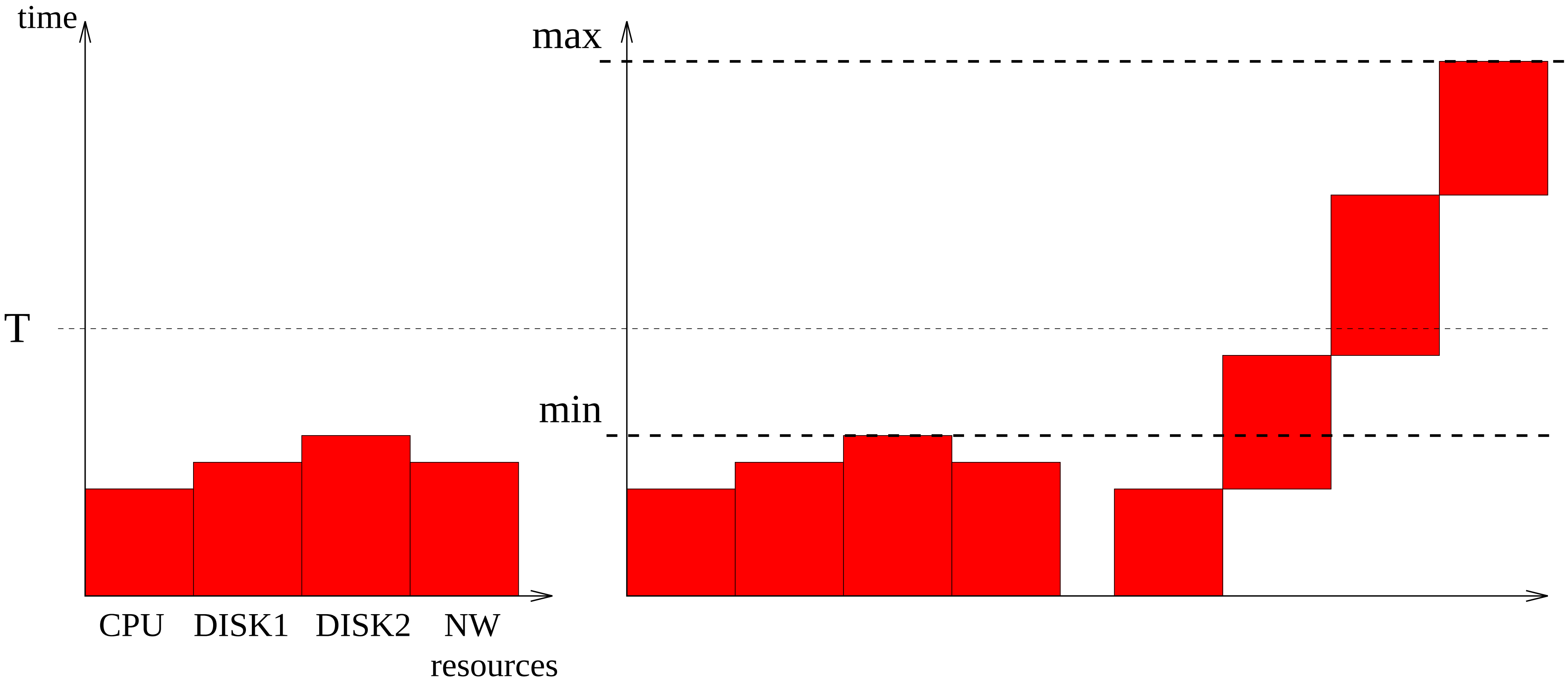


OPERATOR TREE



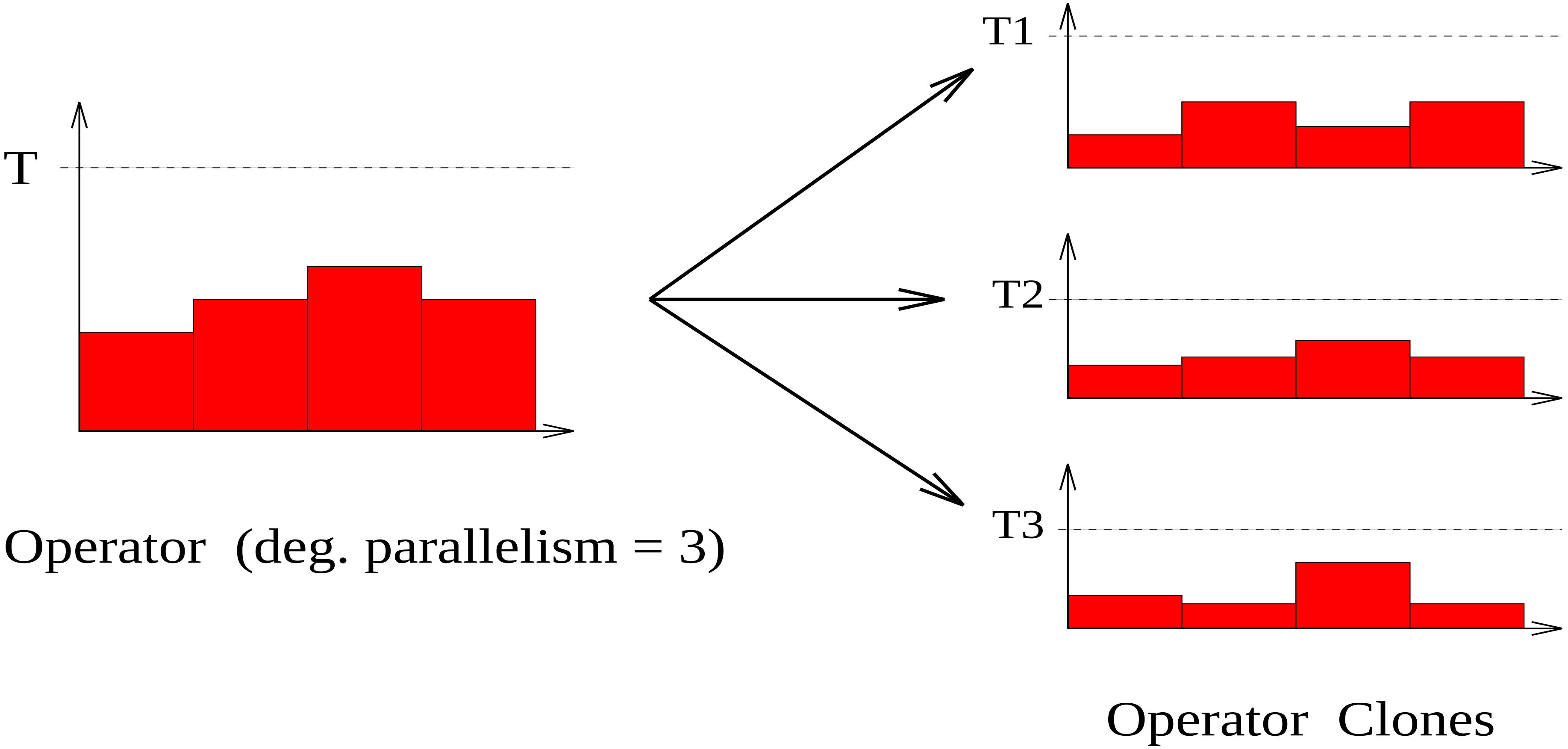
QUERY TASK TREE

Multi-dimensional Operator Cost Model



- Estimate work vector components using conventional cost models and DB statistics
- T = function($\overline{W}$, overlap)

Partitioned Parallelism

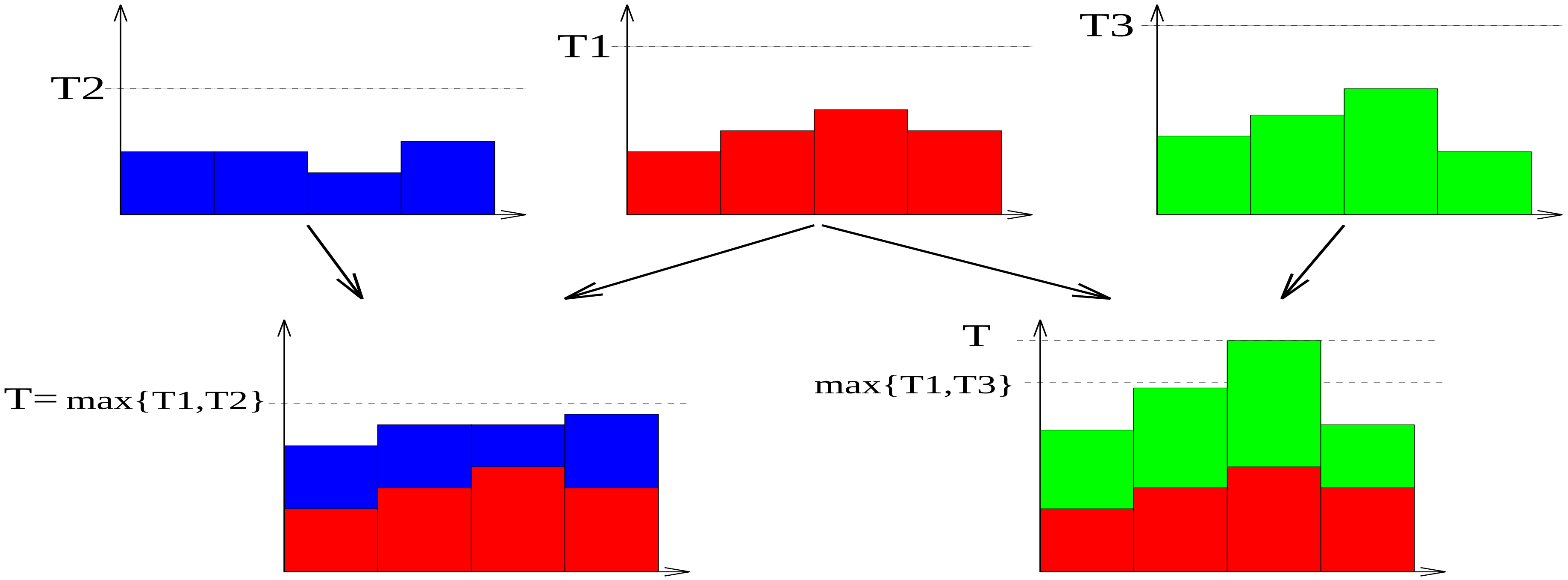


Degree of Parallelism?

- Coarse Granularity Restriction
- Pre-processing Greedy Selection Step

Pipelined and Independent Parallelism

- Collection of clones to be executed concurrently
- Schedule: mapping of clones to sites s.t. clones of the same operator are mapped to distinct sites *(placement constraint)*
- Effects of resource sharing



- Response Time: slowest clone or most heavily congested resource

Problem Statement

- Find a **packing** of the d-dimensional clone (work) vectors of M operators in P d-dimensional bins (system sites) that
  (a) satisfies the placement constraint, and
  (b) minimizes the required bin capacity (max resource usage)

- Instance of d-dimensional *bin-design problem* (dual of bin-packing)

An Algorithm for Independent Query Tasks

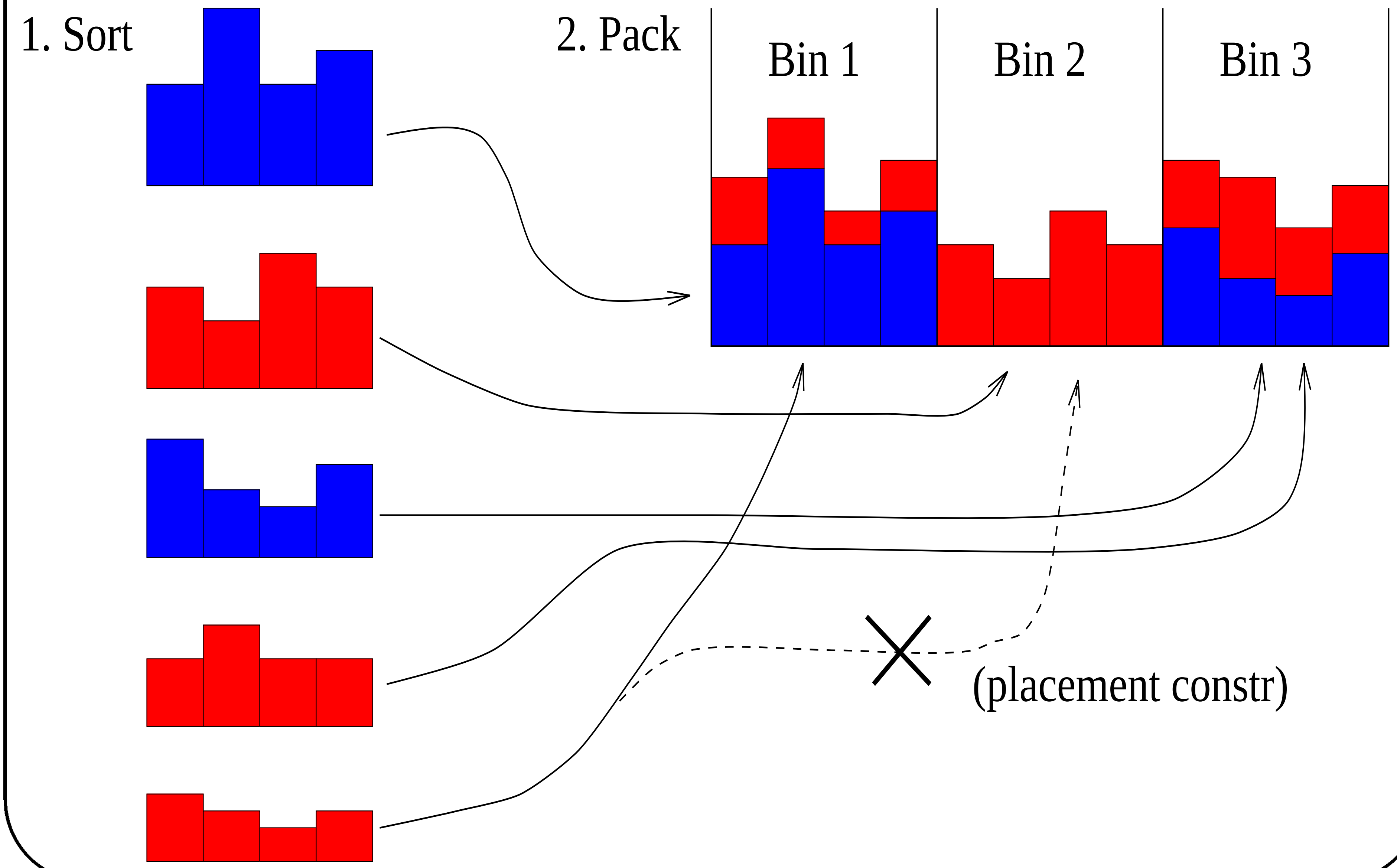
OperatorSchedule: List Scheduling Heuristic (a-la Graham)

1. Sort work vectors in decreasing order of max component
2. In that order, place each vector in least filled eligible bin

Theorem:

OperatorSchedule runs in time $O(MP(M + \log P))$ and produces a provably near-optimal schedule (within $2d + 1$ of optimal).

OPERATORSCHEDULE in Action

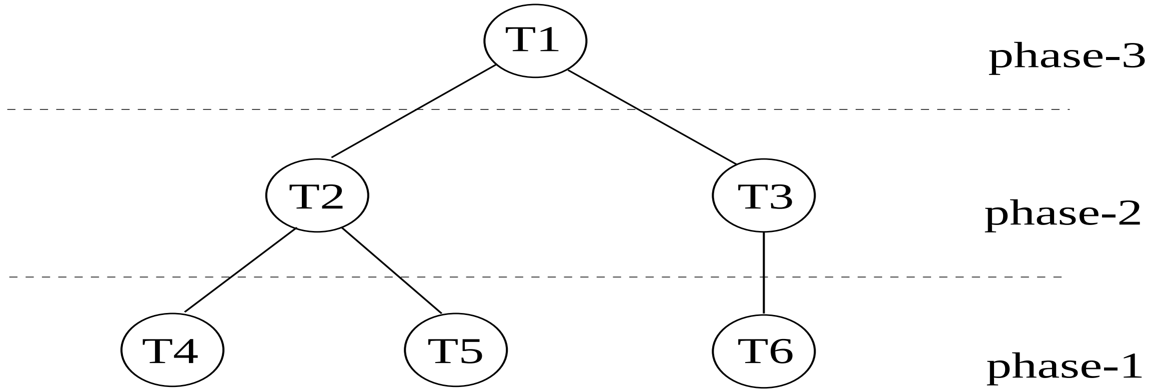


An Algorithm for Query Task Trees

Handling Data Dependencies?

TREESCHEDULE:

1. Split query task tree in phases ("shelves")
2. Schedule each phase using OPERATORSCHEDULE



- Complexity $O(JP(J + \log P))$
- Theoretical performance bounds? Open problem...

Experimental Results

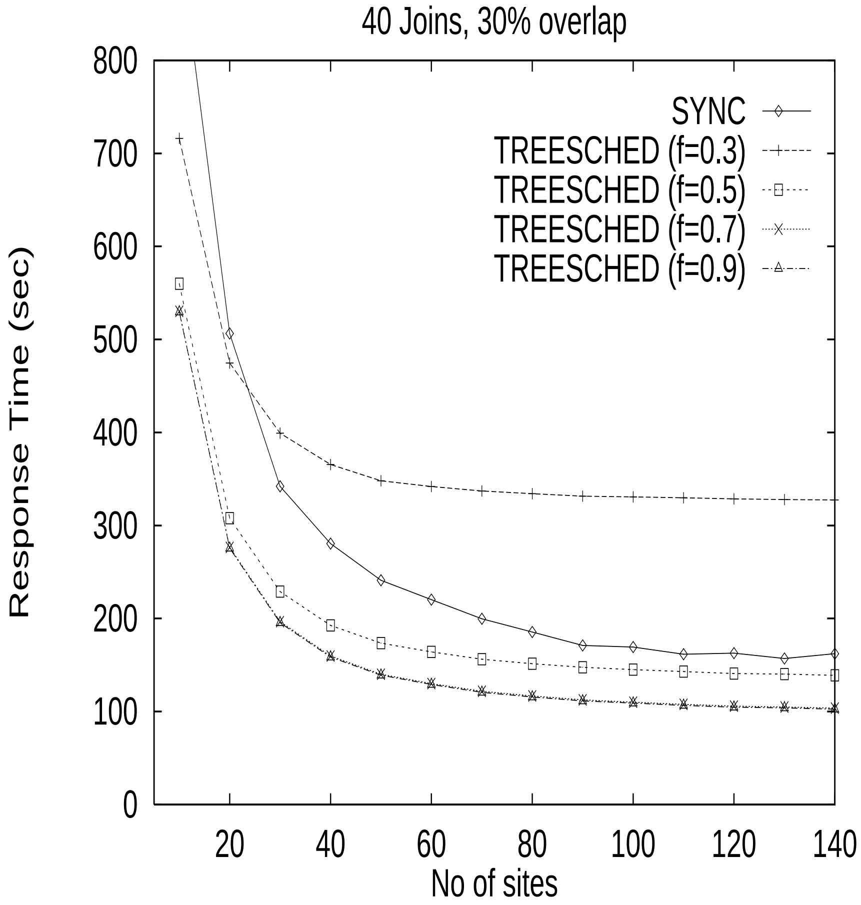
- TreeSchedule vs SYNCHRONOUS (Hsiao et al. [SIGMOD'94]) and lower bound on optimal
- *Average* effectiveness based on *cost model*

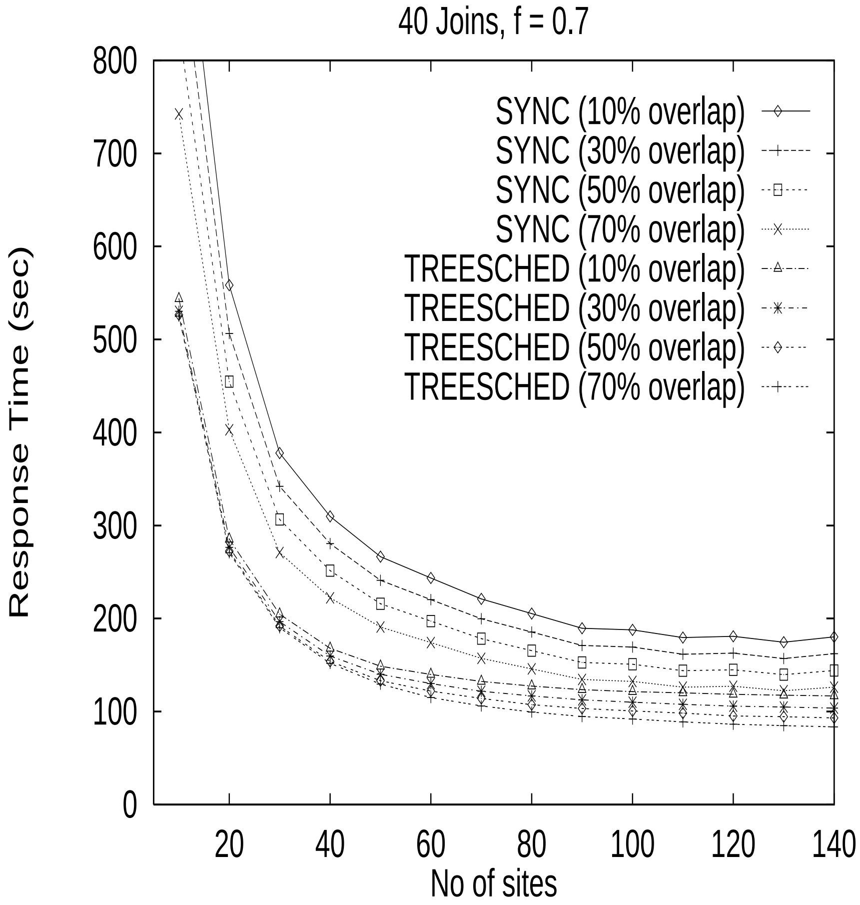
Experimental Assumptions

- Only hash-joins (due to SYNCHRONOUS)
- No execution skew (except for startup cost)
- Uniform resource overlap (single parameter)

Experimental Results (cont.)

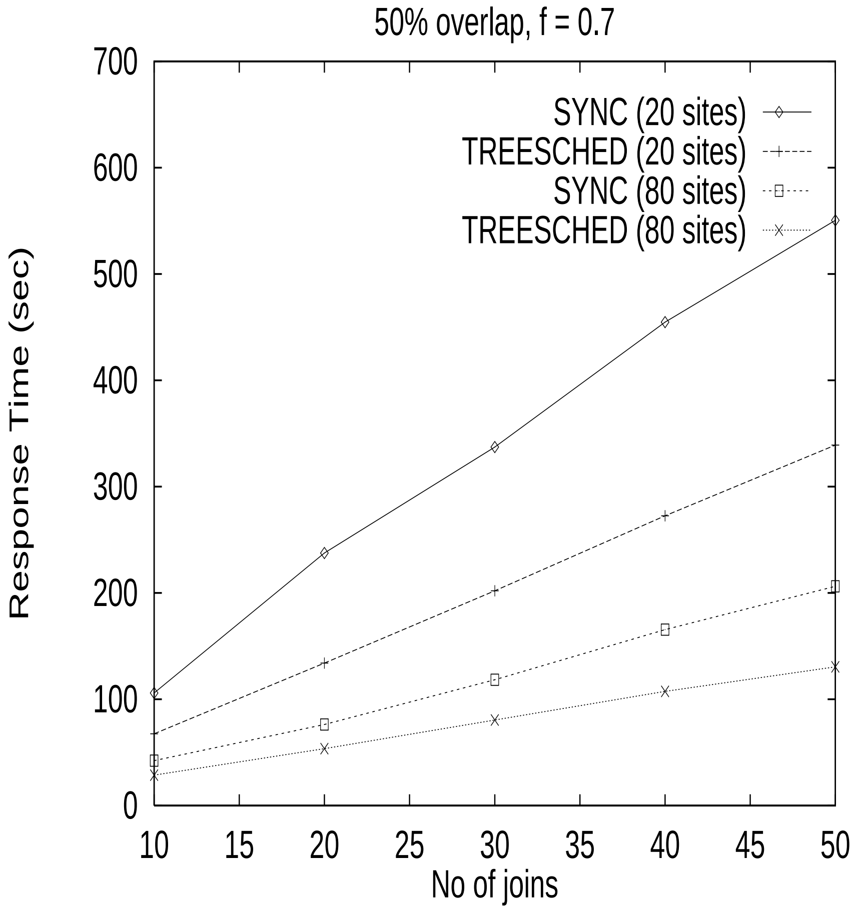
- TREESCHEDULE vs SYNCHRONOUS

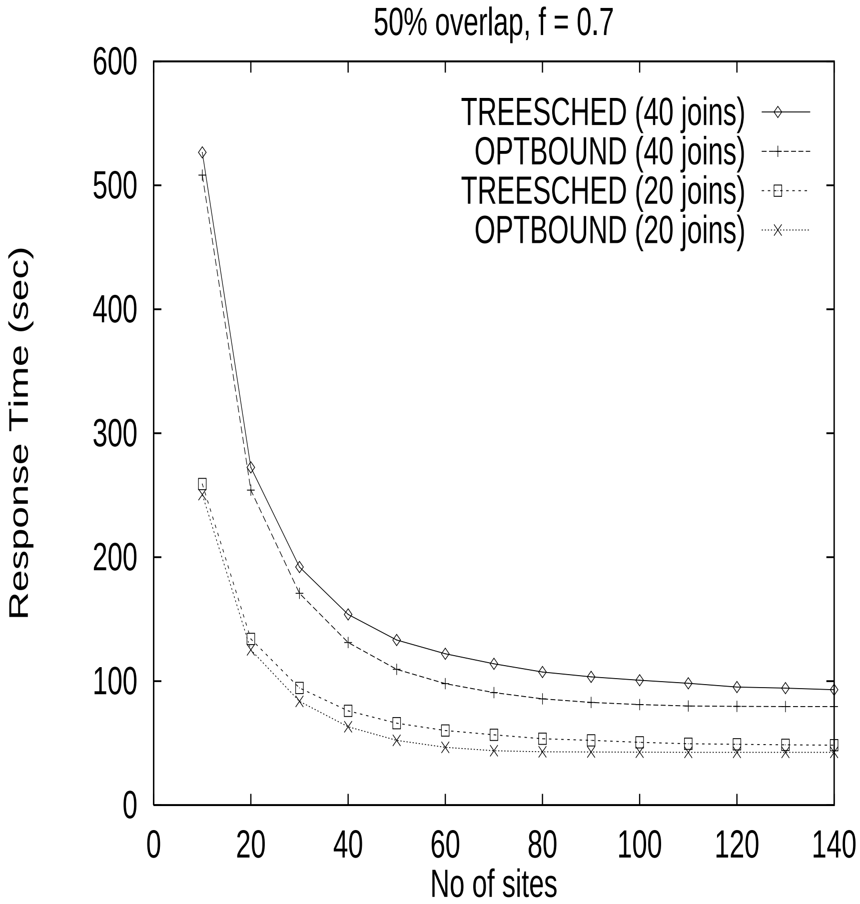




Experimental Results (cont.)

- TreeSchedule vs SYNCHRONOUS
- TreeSchedule vs Optimal





Ongoing and Future Work

- Extend framework to incorporate "non-preemptable" resources (e.g., memory)
- Introduce different degrees of "resource preemptability" (e.g., disk vs CPU)
- Verify cost model and algorithms in real SN system
- Examine applicability to scheduling in multimedia storage servers, Garofalakis and Ioannidis [ACM Comp. Surveys'95]

Experimental Testbed

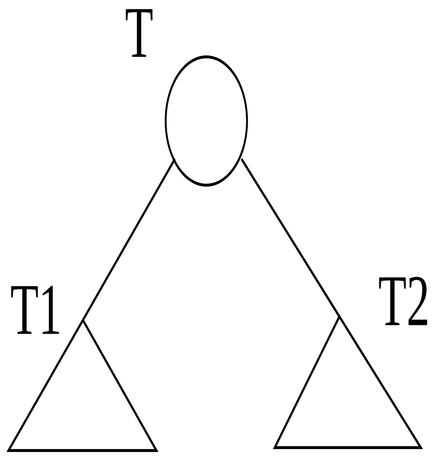
- 3-dimensional sites (CPU, Disk, NW interface)
- Tree queries 10,20,30,40,50 joins (20 random query graphs/size)
- Overlap: 10% - 70%, Coarse Granularity: 0.3 - 0.9
- Comparison metric: Average plan response time

Experiment Parameter Settings

Configuration/Catalog Parameters	**Value**
Number of Sites	10 - 140
CPU Speed	1 MIPS
Effective Disk Service Time per page	20 msec
Startup Cost per site (α)	15 msec
Network Transfer Cost per byte (β)	0.6 μsec
Tuple Size	128 bytes
Page Size	40 tuples
Relation Size	10^3 - 10^5 tuples
CPU Cost Parameters	**No. of Instr.**
Read Page from Disk	5000
Write Page to Disk	5000
Extract Tuple	300
Hash Tuple	100
Probe Hash Table	200

The SYNCHRONOUS Algorithm (Hsiao et al.)

- Partition processors in proportion to "cumulative work" (no resource sharing)



$$P(T1) = \left\lfloor \frac{CW(T1)}{CW(T1) + CW(T2)} \right\rfloor * P(T)$$

$$P(T2) = P(T) - P(T1)$$

Deterministic Scheduling Theory

- Multiprocessor scheduling: mapping DAGs of computation-intensive tasks on parallel machines
- Simple model of parallelization, inadequate for multi-dimensional DB operators
- Only very recent work on multi-resource parallel task scheduling: Shachnai and Turek [Unpubl.], Chakrabarti and Muthukrishnan [SPAA'96]
  Both assume globally accessible resources (not true for SN)