

Systems Infrastructure for Data Science

Web Science Group

Uni Freiburg

WS 2013/14

Lecture X: Parallel Databases

Topics

- Motivation and Goals
- Architectures
- Data placement
- Query processing
- Load balancing

Motivation

- Large volume of data => Use disk and large main memory
- I/O bottleneck (or memory access bottleneck)
 - $\text{speed}(\text{disk}) \ll \text{speed}(\text{RAM}) \ll \text{speed}(\text{microprocessor})$
- Predictions
 - (Micro-) processor speed growth: 50 % per year (Moore's Law)
 - DRAM capacity growth: 4 x every three years
 - Disk throughput: 2 x in the last ten years
- Conclusion: the I/O bottleneck worsens
 - => Increase the I/O bandwidth through parallelism**

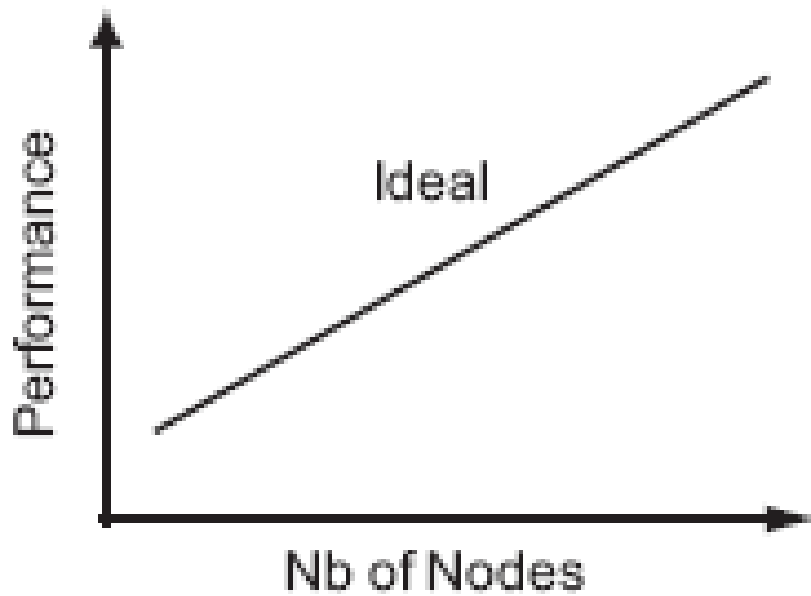
Motivation

- Also, Moore's Law doesn't quite apply any more because of the heat problem.
 - Recent trend:
 - Instead of fitting more chips on a single board, increase the number of processors.
- => The need for parallel processing**

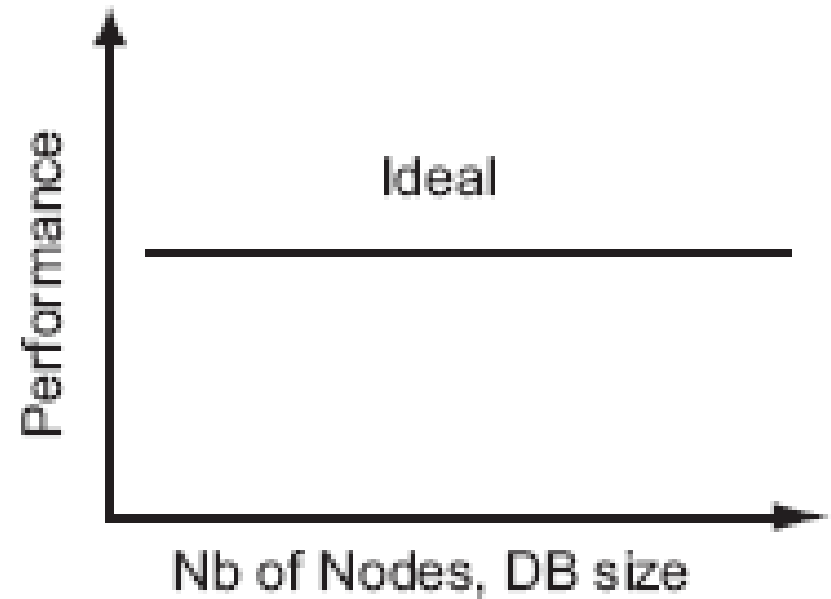
Goals

- I/O bottleneck
 - Increase the I/O bandwidth through parallelism
- Exploit multiple processors, multiple disks
 - Intra-query parallelism (for response time)
 - Inter-query parallelism (for throughput = # of transactions/second)
- High performance
 - Overhead
 - Load balancing
- High availability
 - Exploit the existing redundancy
 - Be careful about imbalance
- Extensibility
 - Speed-up and Scalability

Extensibility



(a) Linear speedup



(b) Linear scaleup

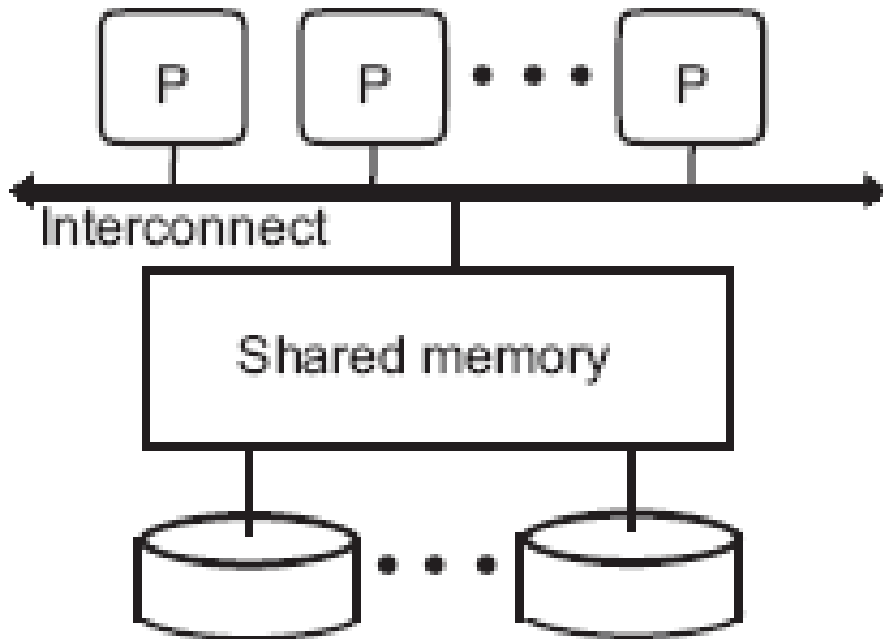
Today's Topics

- Parallel Databases
 - Motivation and Goals
 - Architectures
 - Data placement
 - Query processing
 - Load balancing

Parallel System Architectures

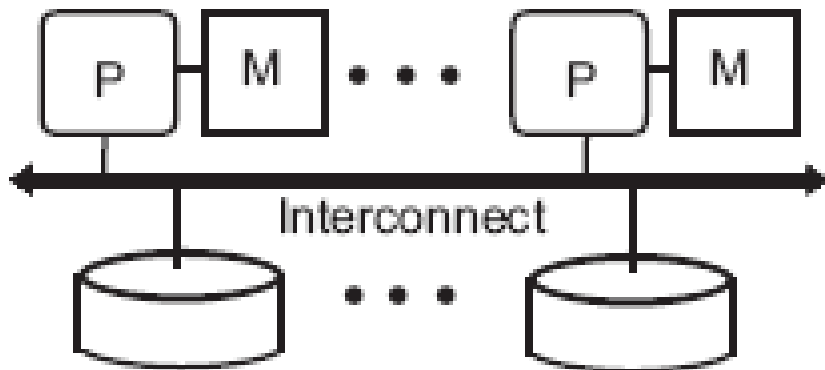
- Shared-Memory
- Shared-Disk
- Shared-Nothing
- Hybrid
 - NUMA
 - Cluster

Shared-Memory



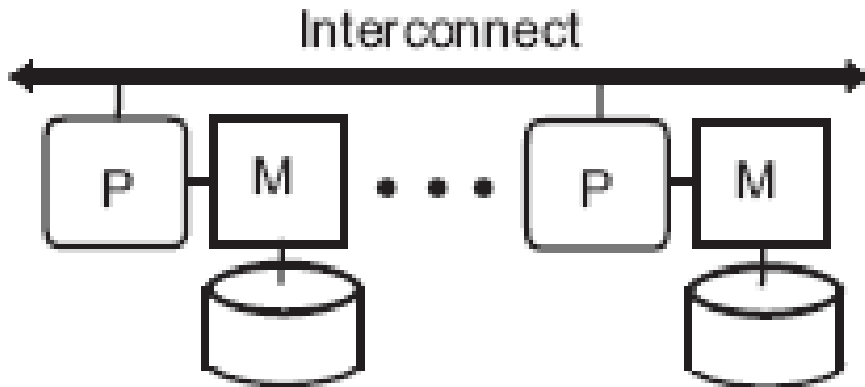
- Fast interconnect
- Single OS
- Advantages:
 - Simplicity
 - Easy load balancing
- Problems:
 - High cost (the interconnect)
 - Limited extensibility (~ 10 P's)
 - Low availability

Shared-Disk



- Separate OS per P-M
- Advantages:
 - Lower cost
 - Higher extensibility (~ 100 P-M's)
 - Load balancing
 - Availability
- Problems:
 - Complexity (cache consistency with lock-based protocols, 2PC, etc.)
 - Overhead
 - Disk bottleneck

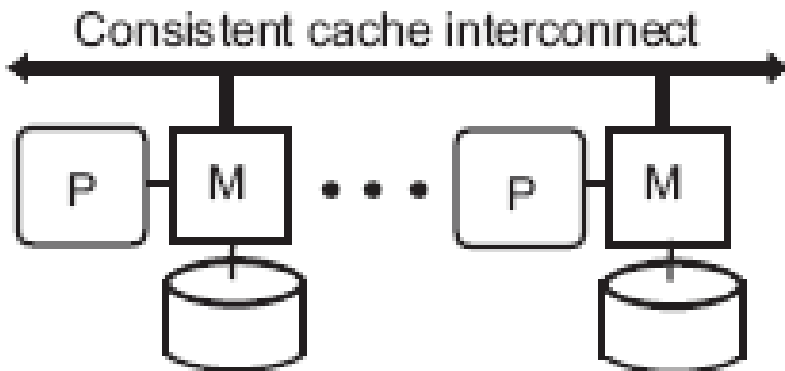
Shared-Nothing



- Separate OS per P-M-D
- Each node \sim site
- Advantages:
 - Extensibility and scalability
 - Lower cost
 - High availability
- Problems:
 - Complexity
 - Difficult load balancing

Hybrid Architectures

Non Uniform Memory Architecture (NUMA)



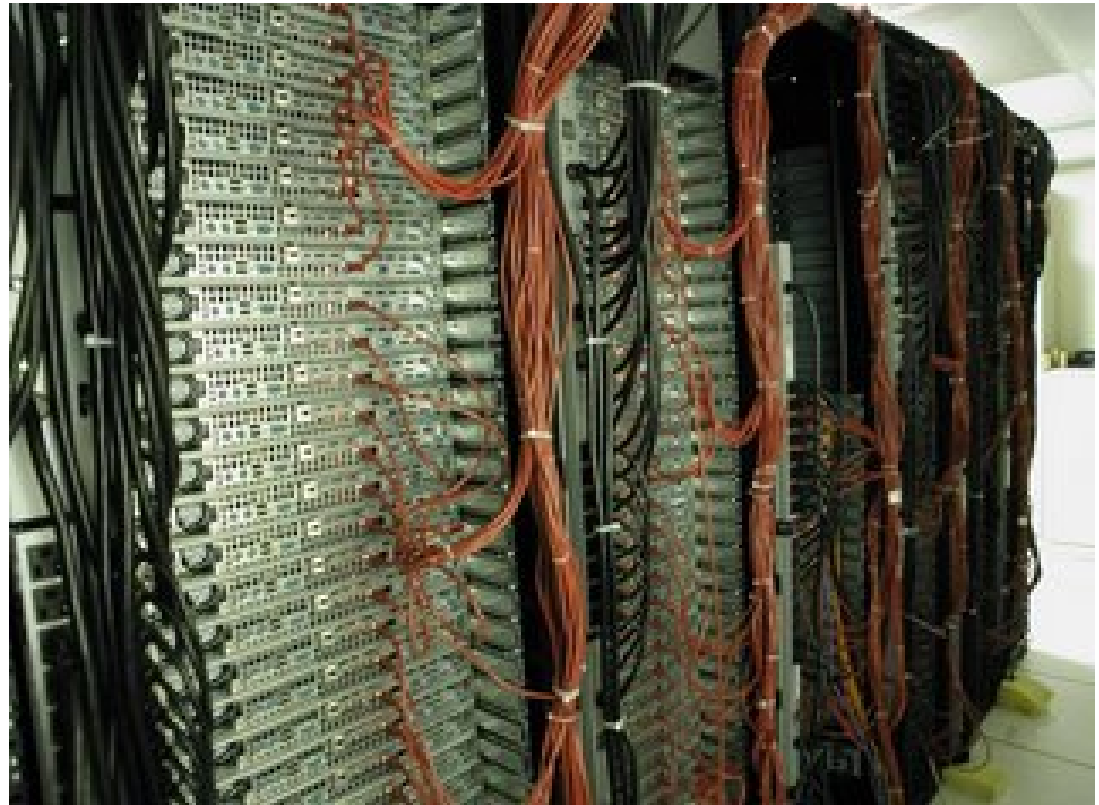
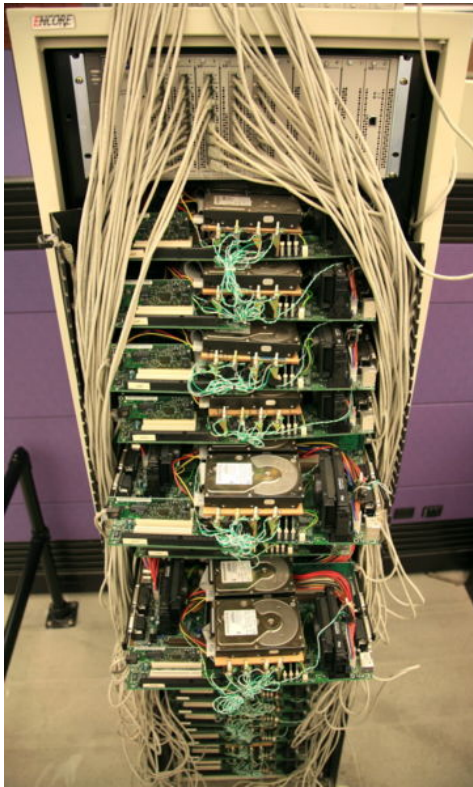
- Cache-coherent NUMA
- Any P can access to any M.
- More efficient cache consistency supported by interconnect hardware
- Memory access cost
 - Remote = 2-3 x Local

Hybrid Architectures Cluster

- Independent homogeneous server nodes at a single site
- Interconnect options
 - LAN (cheap, slower)
 - Myrinet, Infiniband, etc. (faster, low-latency)
- Shared-disk alternatives:
 - NAS (Network-Attached Storage) -> low throughput
 - SAN (Storage-Area Network) -> high cost of ownership
- Advantages of cluster architecture:
 - Flexible and efficient as shared-memory
 - Extensible and available as shared-disk/shared-nothing

The Google Cluster

- ~ 15,000 nodes of homogeneous commodity PCs [BDH'03]
- Currently: over 900,000 servers world-wide [Aug. 2011 news]



Parallel Architectures

Summary

- For small number of nodes:
 - Shared-memory -> load balancing
 - Shared-disk/Shared-nothing -> extensibility
 - SAN w/ Shared-disk -> simple administration
- For large number of nodes:
 - NUMA (~ 100 nodes)
 - Cluster (~ 1000 nodes)
 - Efficiency + Simplicity of Shared-memory
 - Extensibility + Cost of Shared-disk/Shared-nothing

Topics

- Parallel Databases
 - Motivation and Goals
 - Architectures
 - Data placement
 - Query processing
 - Load balancing

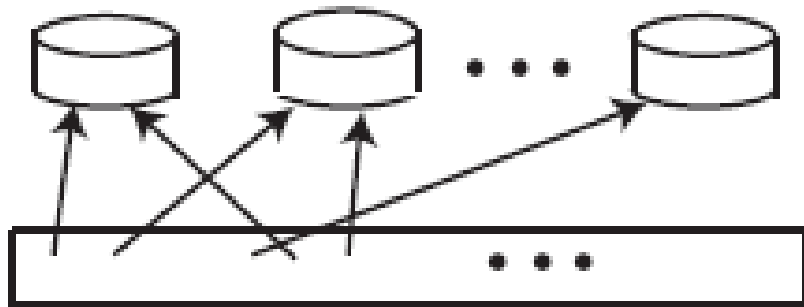
Parallel Data Placement

- Assume: shared-nothing (most general and common)
- To reduce communication costs, programs should be executed where the data reside.
- Similar to distributed DBMS's:
 - Fragmentation
- Differences:
 - Users are not associated with particular nodes.
 - Load balancing for large number of nodes is harder.
- How to place the data so that the system performance is maximized?
 - partitioning (min. response time) vs. clustering (min. total time)

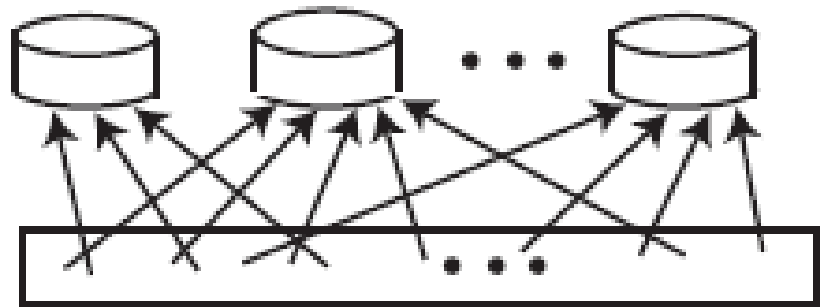
Data Partitioning

- Each relation is divided into n partitions that are mapped onto different disks.
- Implementation
 - Round-robin
 - Maps i -th element to node $i \bmod n$
 - Simple but only exact-match queries
 - Range
 - B-tree index
 - Supports range queries but large index
 - Hashing
 - Hash function
 - Only exact-match queries but small index

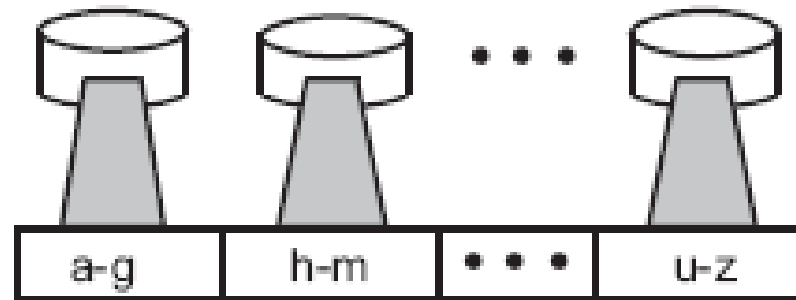
Full Partitioning Schemes



(a) Round-Robin



(b) Hashing



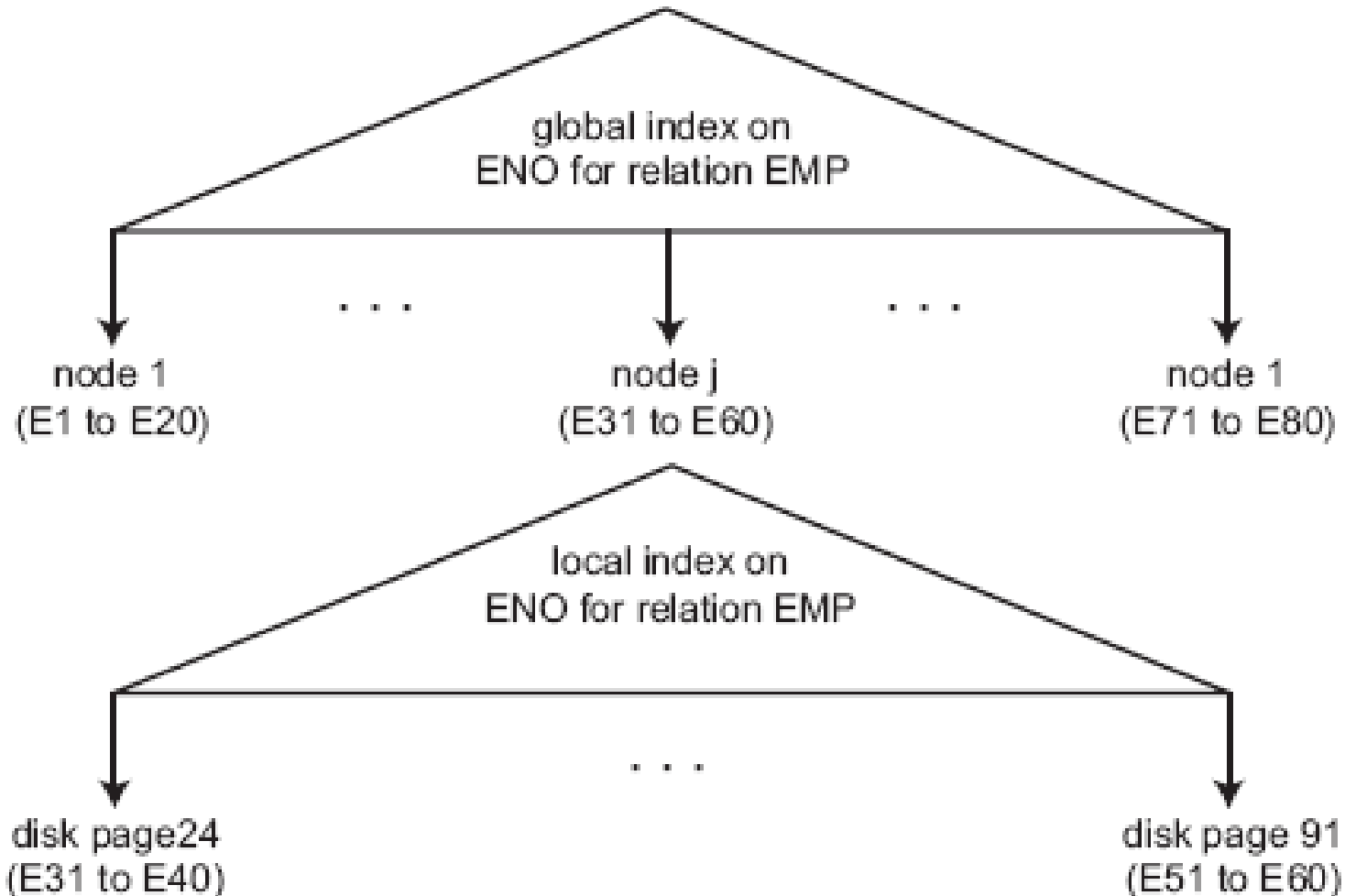
(c) Range

Variable Partitioning

- Each relation is partitioned across a certain number of nodes (instead of all), depending on its:
 - size
 - access frequency
- Periodic reorganization for load balancing
- Global index replicated on each node to provide associative access + Local indices

Global and Local Indices

Example



Replicated Data Partitioning for H/A

- High-Availability requires data replication
 - simple solution is mirrored disks
 - hurts load balancing when one node fails
 - more elaborate solutions achieve load balancing
 - interleaved partitioning (Teradata)
 - chained partitioning (Gamma)

Replicated Data Partitioning for H/A

Interleaved Partitioning

Node	1	2	3	4
Primary copy	R1	R2	R3	R4
Backup copy	r 2.3 r 3.2	r 1.1 r 3.3	r 1.2 r 2.1	r 1.3 r 2.2 r 3.1

Replicated Data Partitioning for H/A

Chained Partitioning

Node	1	2	3	4
Primary copy	R1	R2	R3	R4
Backup copy	r4	r1	r2	r3

Topics

- Parallel Databases
 - Motivation and Goals
 - Architectures
 - Data placement
 - Query processing
 - Load balancing

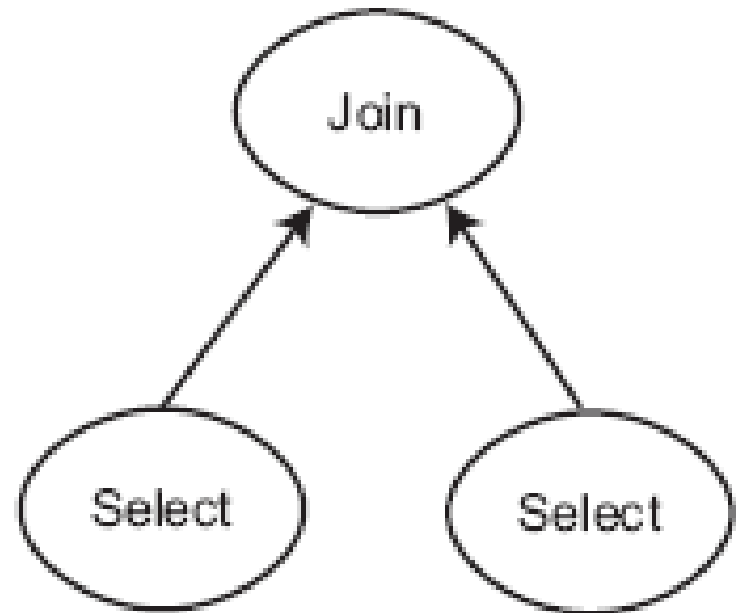
Parallel Query Processing

- Query parallelism
 - inter-query
 - intra-query
 - inter-operator
 - intra-operator

Inter-operator Parallelism

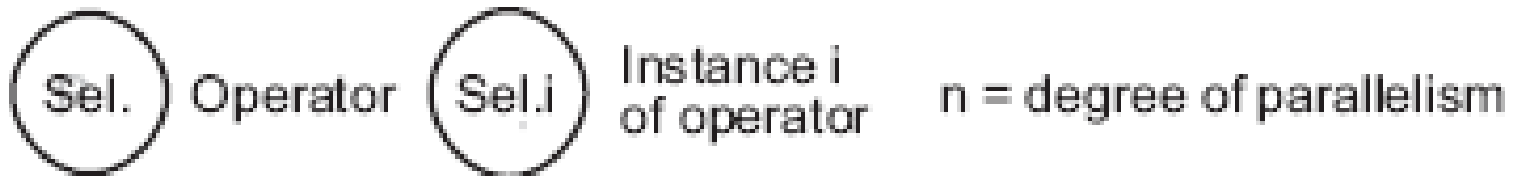
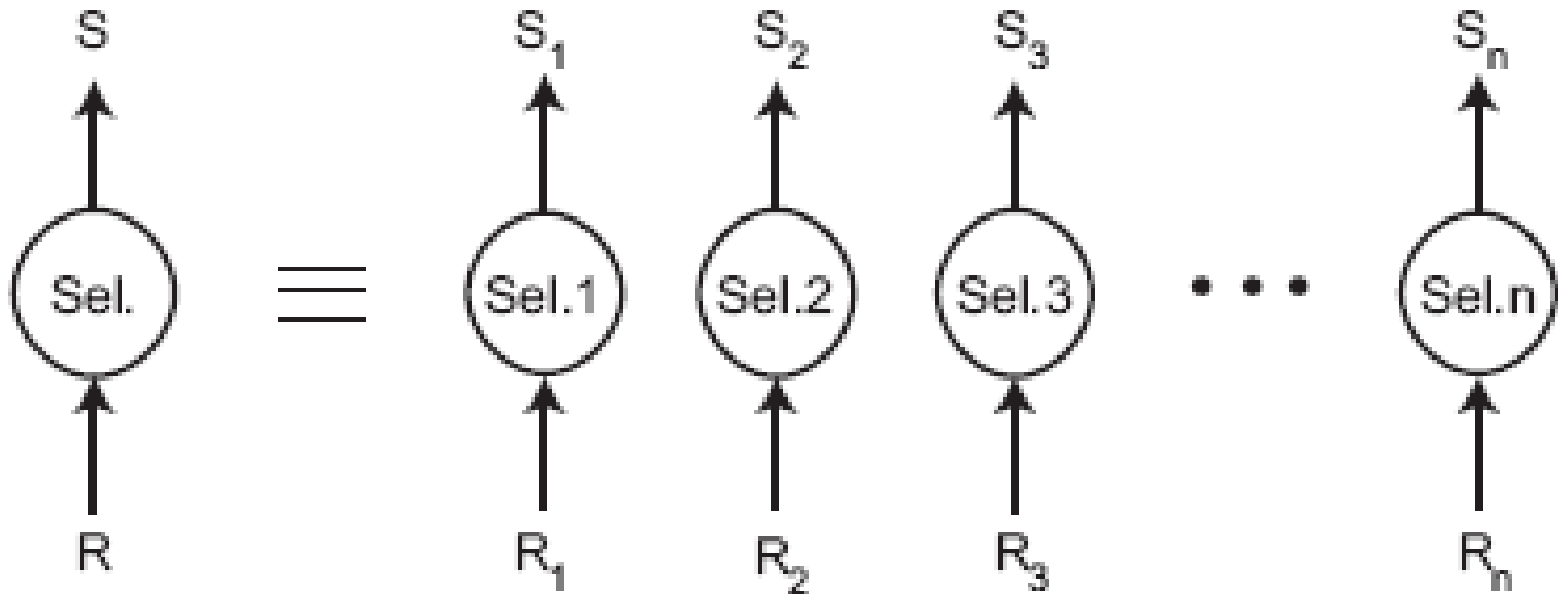
Example

- Pipeline parallelism
 - Join and Select execute in parallel.
- Independent parallelism
 - The two Select's execute in parallel.



Intra-operator Parallelism

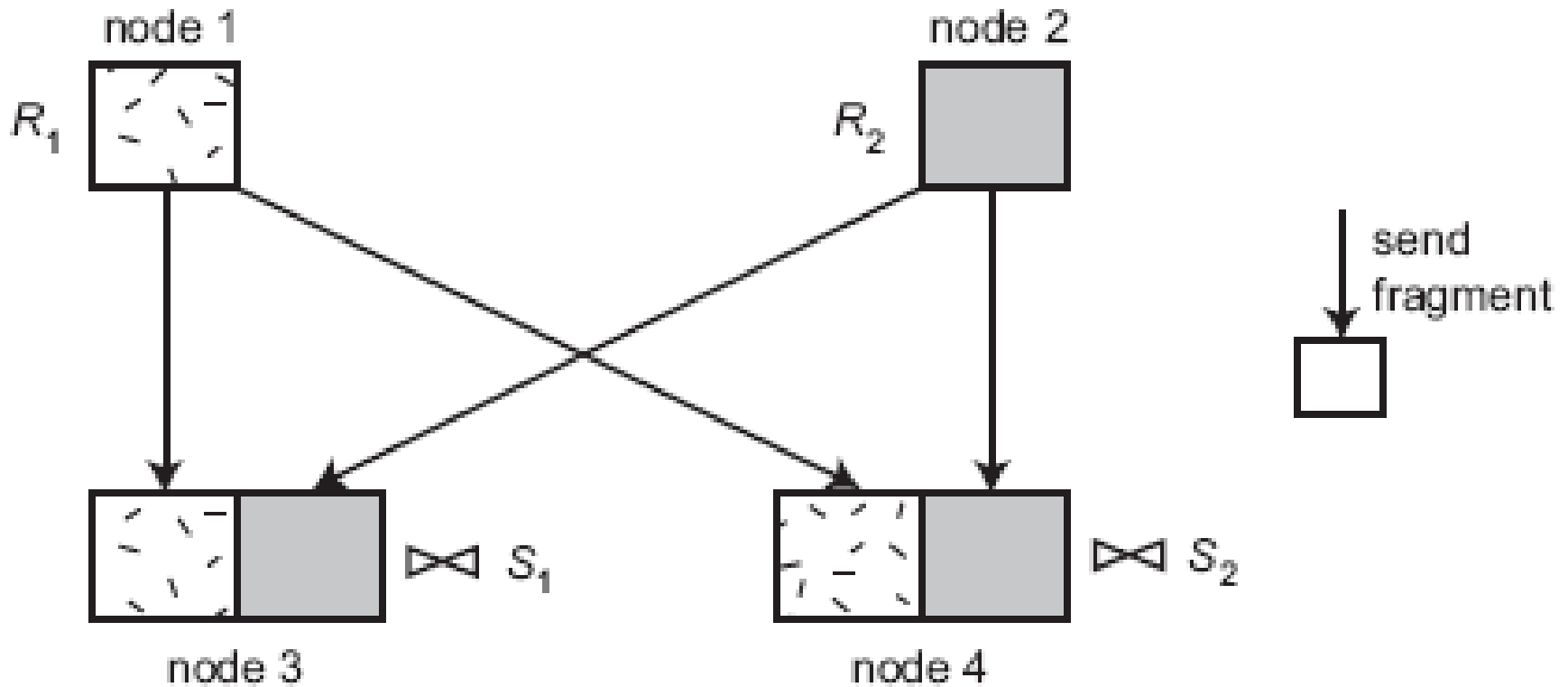
Example



Parallel Join Processing

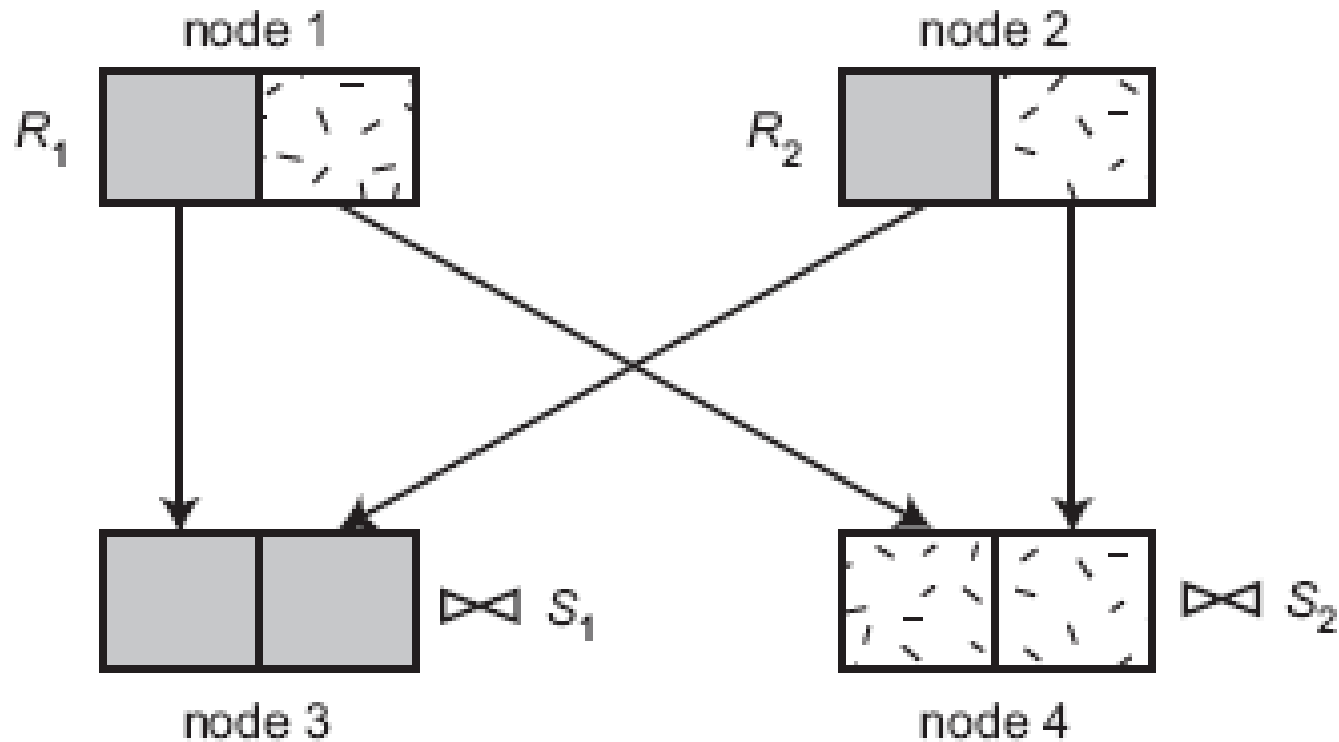
- Three basic algorithms for intra-operator parallelism:
 - Parallel Nested Loop Join:
 - no special assumptions
 - Parallel Associative Join:
 - assumption: one relation is declustered on join attribute + equi-join
 - Parallel Hash Join:
 - assumption: equi-join
- They also apply to other complex operators such as duplicate elimination, union, intersection, etc. with minor adaptation.

Parallel Nested Loop Join



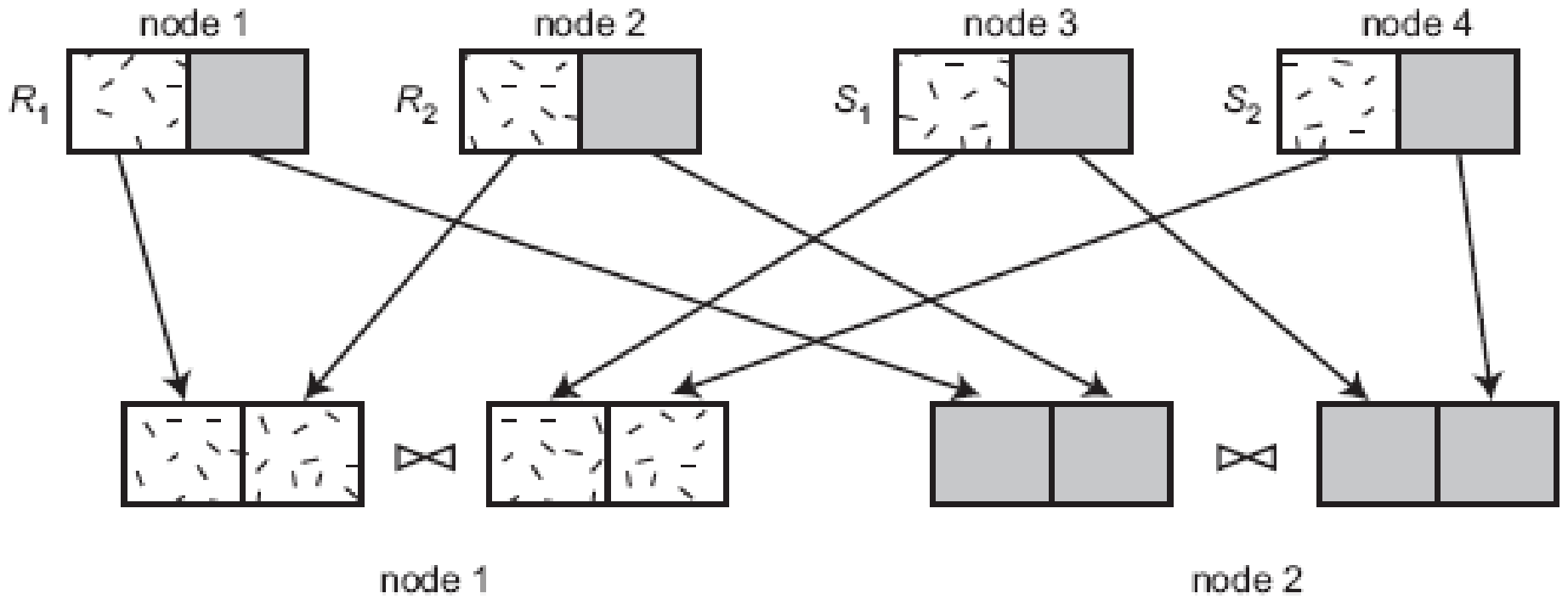
$$R \triangleright \triangleleft S \rightarrow \bigcup_{i=1}^n R \triangleright \triangleleft S_i$$

Parallel Associative Join



$$R \triangleright \triangleleft S \rightarrow \bigcup_{i=1}^n (R_i \triangleright \triangleleft S_i)$$

Parallel Hash Join



$$R \bowtie S \rightarrow \bigcup_{i=1}^p (R_i \bowtie S_i)$$

Which one to use?

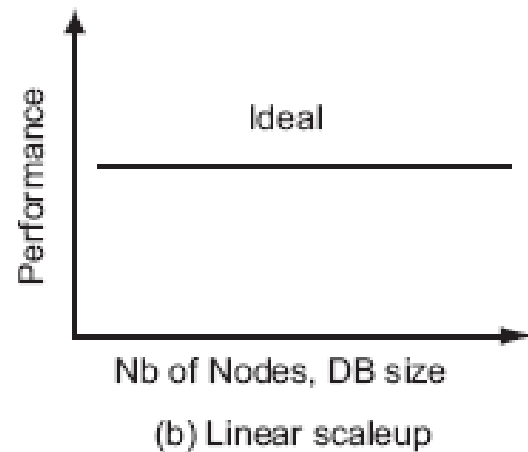
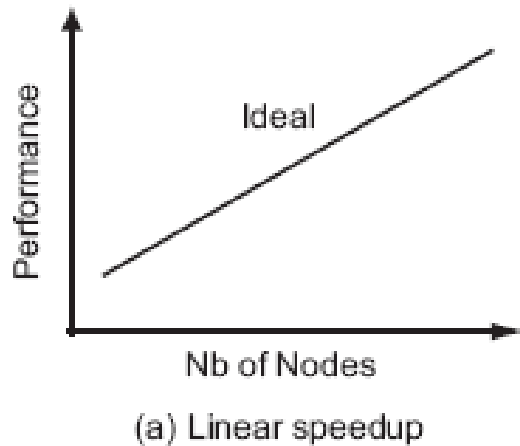
- Use Parallel Associative Join where applicable (i.e., equi-join + partitioning based on the join attribute).
- Otherwise, compute total communication + processing cost for Parallel Nested Loop Join and Parallel Hash Join, and use the one with the smaller cost.

Topics

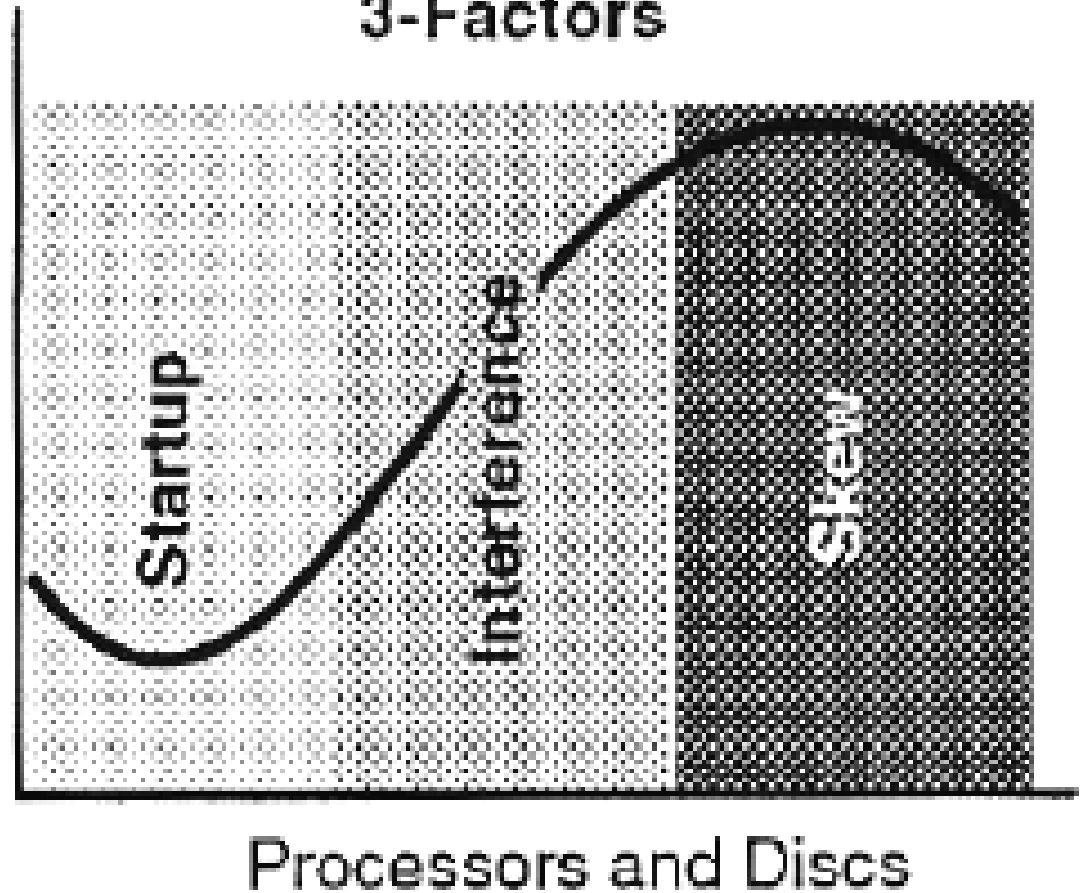
- Parallel Databases
 - Motivation and Goals
 - Architectures
 - Data placement
 - Query processing
 - Load balancing

Three Barriers to Extensibility

Ideal Curves



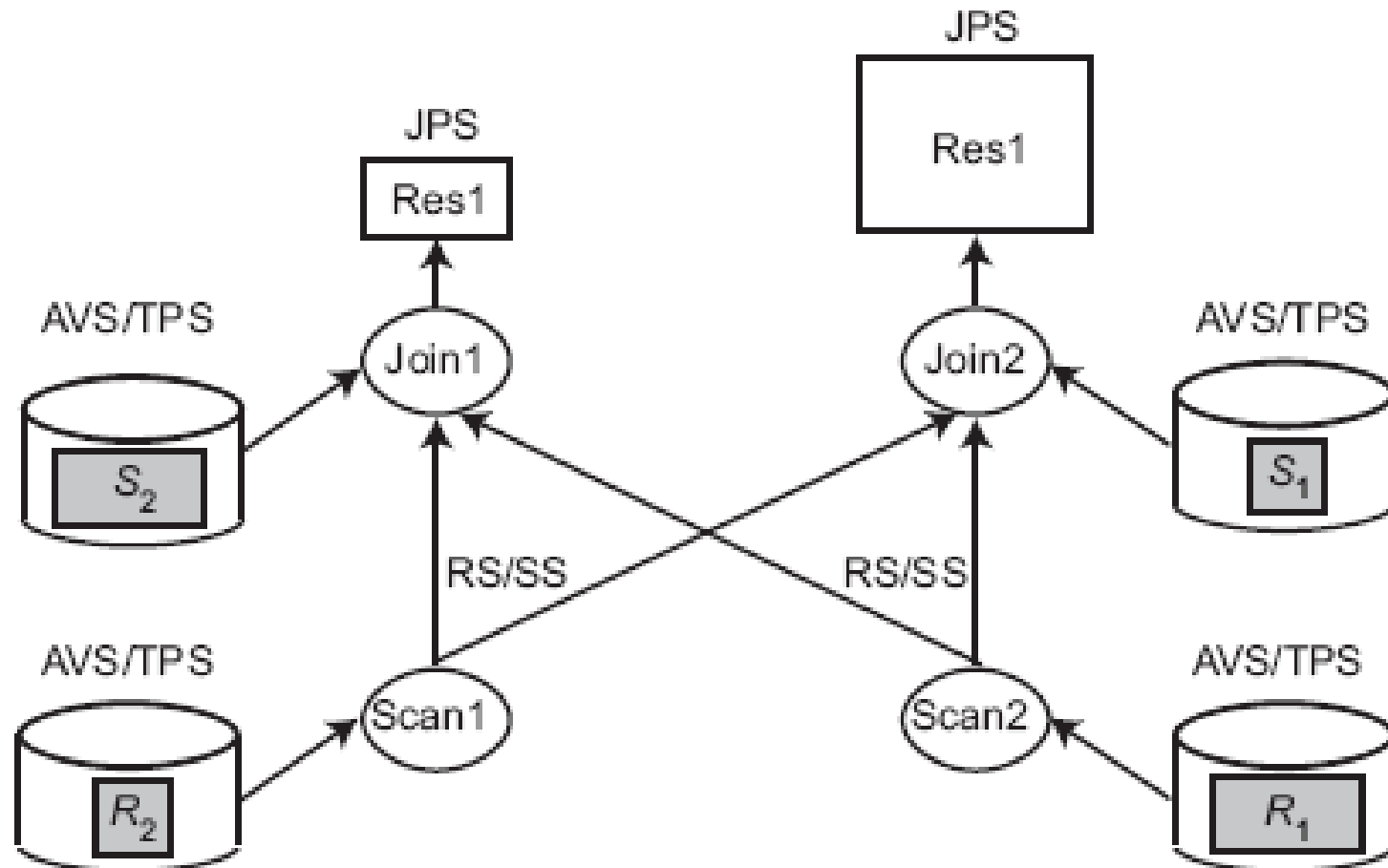
A Bad Speedup Curve 3-Factors



Load Balancing

- Skewed data distributions in intra-operator parallelism make load balancing harder.
 - Attribute Value Skew (AVS)
 - Tuple Placement Skew (TPS)
 - Selectivity Skew (SS)
 - Redistribution Skew (RS)
 - Join Product Skew (JPS)

Data Skew Example



Load Balancing Techniques

- Intra-operator load balancing
 - Adaptive techniques (adapt to skew by dynamic load reallocation)
 - Specialized techniques (switch between specialized parallel join algorithms that can deal with skew)
- Inter-operator load balancing (increase pipeline parallelism)
- Intra-query load balancing (combine the two)