WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

CS 764: Topics in Database Management Systems
Lecture 12: Parallel DBMSs

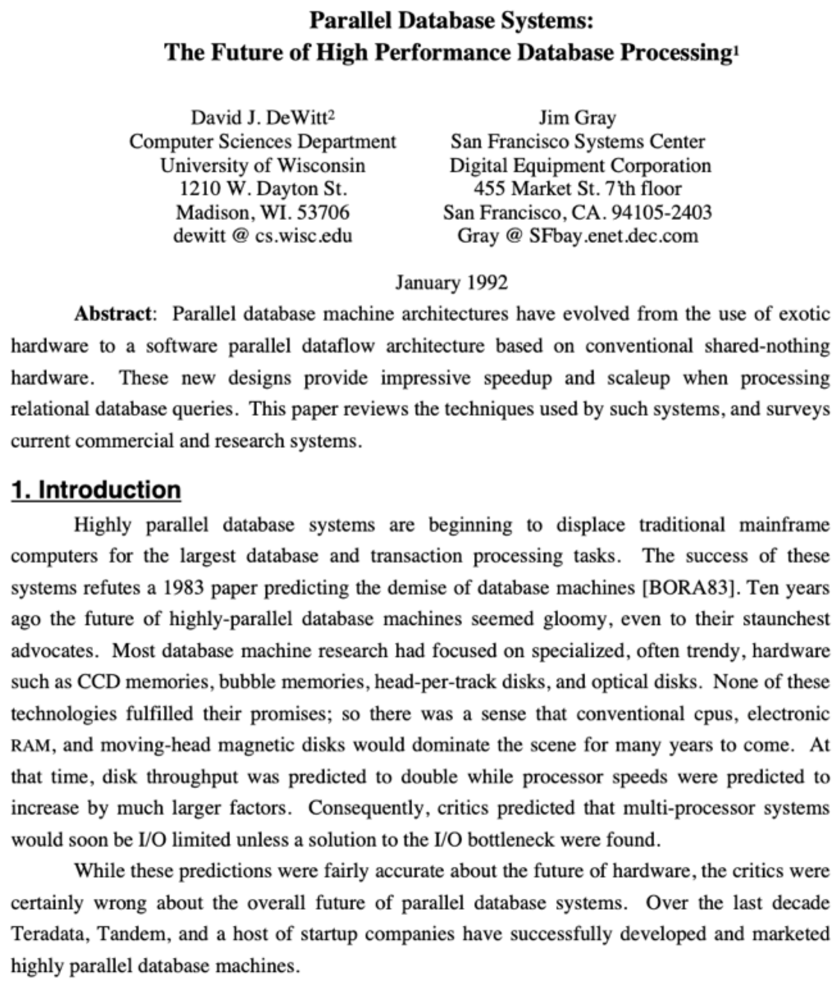
Xiangyao Yu
10/14/2020

Announcement

Class schedule

- 10/21: Last lecture included in exam
- 10/26: Guest lecture from Ippokratis Pandis (AWS)
- 10/28 and 11/2: Lectures become office hours
- 11/9 – 12/2: Lectures on state-of-the-art research in databases
- 12/7 and 12/9: DAWN workshop

Today's Paper: Parallel DBMSs



Parallel Database Systems:
The Future of High Performance Database Processing[1]

David J. DeWitt[2]
Computer Sciences Department
University of Wisconsin
1210 W. Dayton St.
Madison, WI. 53706
dewitt @ cs.wisc.edu

Jim Gray
San Francisco Systems Center
Digital Equipment Corporation
455 Market St. 7'th floor
San Francisco, CA. 94105-2403
Gray @ SFbay.enet.dec.com

January 1992

Abstract: Parallel database machine architectures have evolved from the use of exotic hardware to a software parallel dataflow architecture based on conventional shared-nothing hardware. These new designs provide impressive speedup and scaleup when processing relational database queries. This paper reviews the techniques used by such systems, and surveys current commercial and research systems.

1. Introduction

Highly parallel database systems are beginning to displace traditional mainframe computers for the largest database and transaction processing tasks. The success of these systems refutes a 1983 paper predicting the demise of database machines [BORA83]. Ten years ago the future of highly-parallel database machines seemed gloomy, even to their staunchest advocates. Most database machine research had focused on specialized, often trendy, hardware such as CCD memories, bubble memories, head-per-track disks, and optical disks. None of these technologies fulfilled their promises; so there was a sense that conventional cpus, electronic RAM, and moving-head magnetic disks would dominate the scene for many years to come. At that time, disk throughput was predicted to double while processor speeds were predicted to increase by much larger factors. Consequently, critics predicted that multi-processor systems would soon be I/O limited unless a solution to the I/O bottleneck were found.

While these predictions were fairly accurate about the future of hardware, the critics were certainly wrong about the overall future of parallel database systems. Over the last decade Teradata, Tandem, and a host of startup companies have successfully developed and marketed highly parallel database machines.

Communications of the ACM, 1992

Agenda

Parallelism metrics

Parallel architecture

Parallel OLAP operators

Parallel Database History

1980’s: database machines

- Specialized hardware to make databases run fast
- Special hardware cannot catch up with Moore’s Law

1980’s – 2010’s: shared-nothing architecture

- Connecting machines using a network

2010’s – future?

Scaling in Parallel Systems

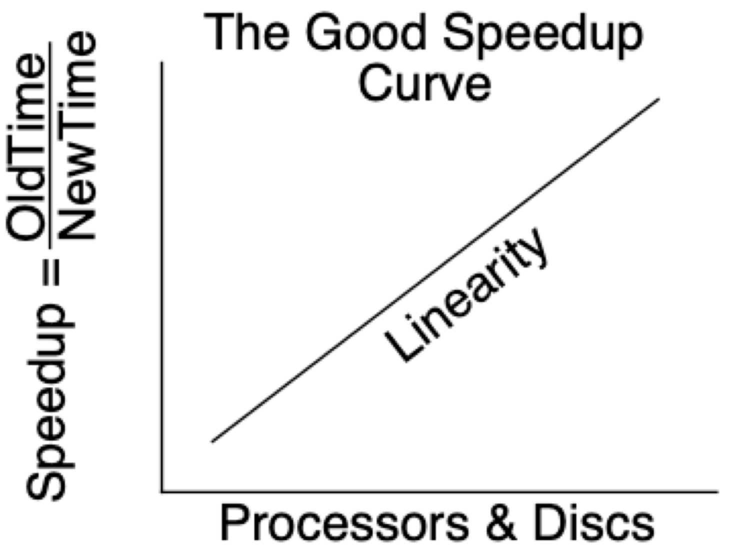
Linear speedup

- Twice as much hardware can perform the task in half the elapsed time
- $\text{Speedup} = \frac{small\ system\ elapsed\ time}{big\ system\ elapsed\ time}$
- Ideally speedup = N, where the big system is N times larger than the small system

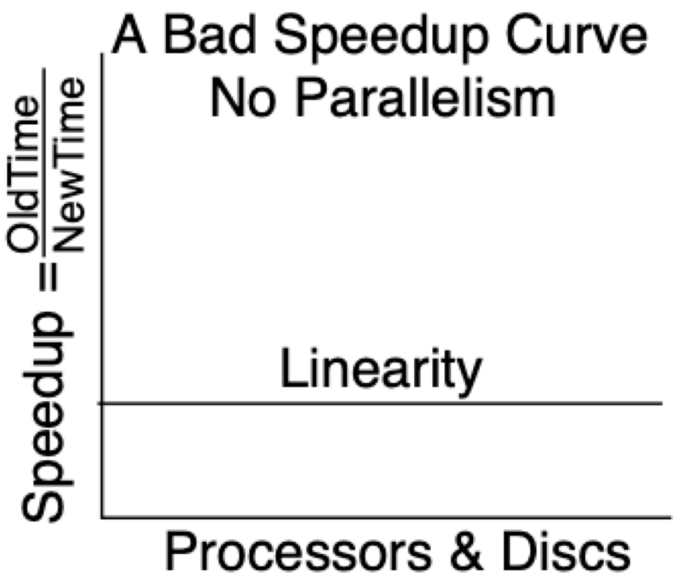
Linear scaleup

- Twice as much hardware can perform twice as large a task in the same elapsed time
- $\text{Scaleup} = \frac{small\ system\ elapsed\ time\ on\ small\ problem}{big\ system\ elapsed\ time\ on\ big\ problem}$
- Ideally scaleup = 1

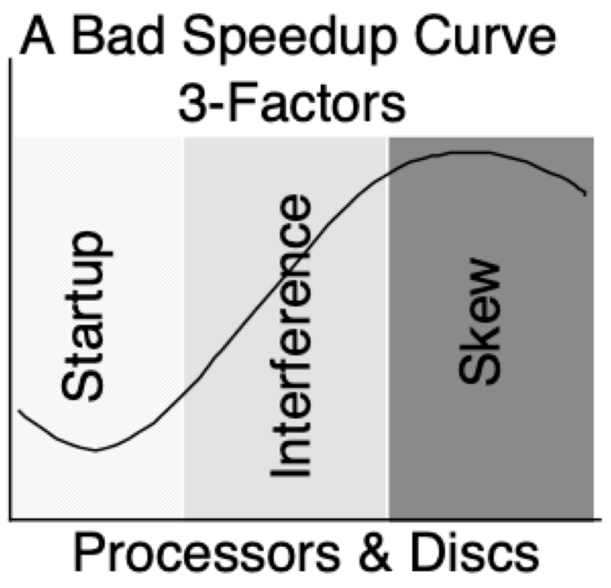
Scaling in Parallel Systems



Ideal speedup

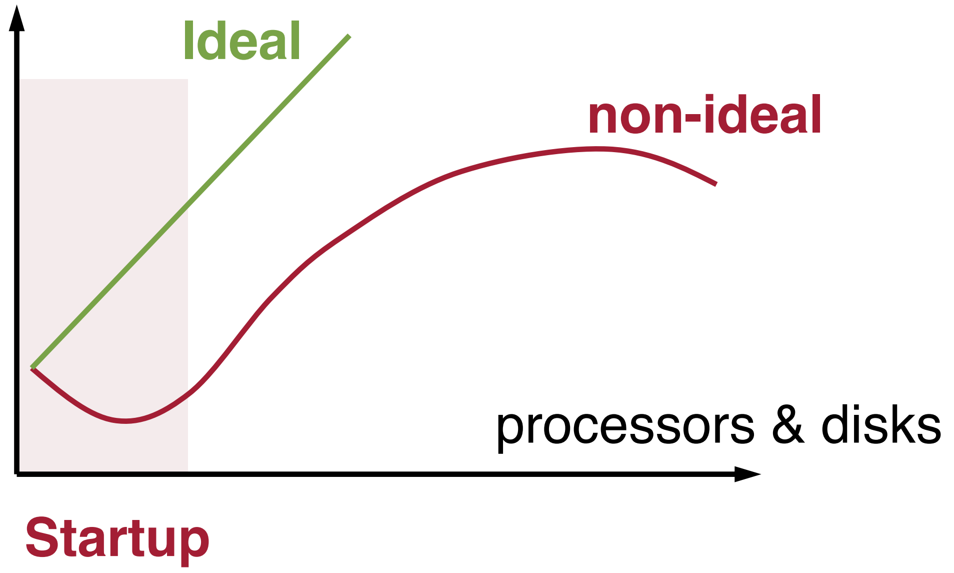


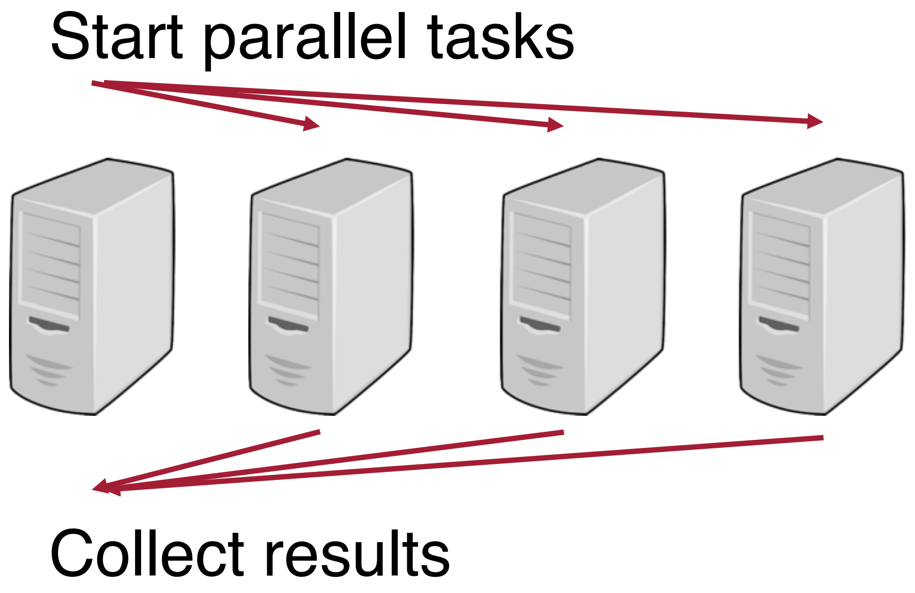
No speedup



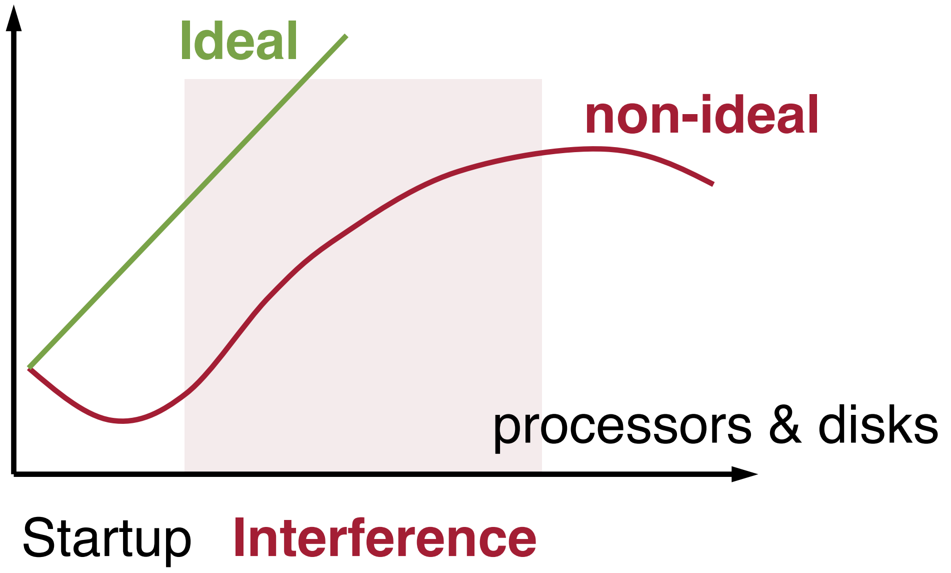
In practice

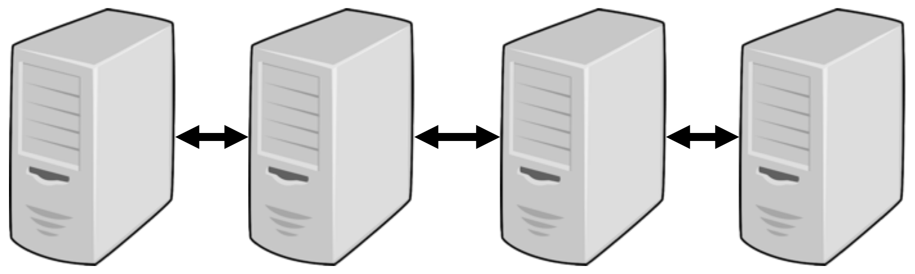
Threats to Parallelism





Threats to Parallelism

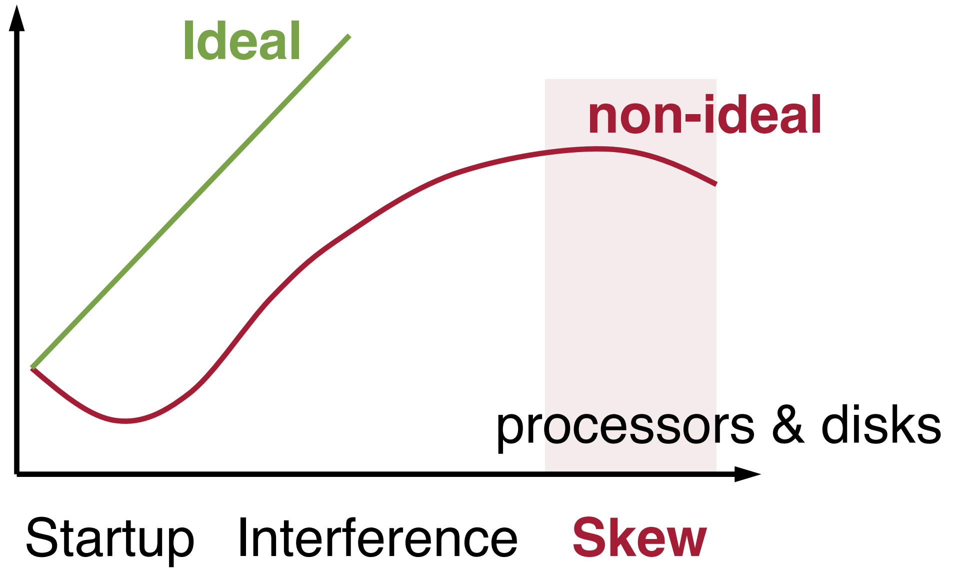


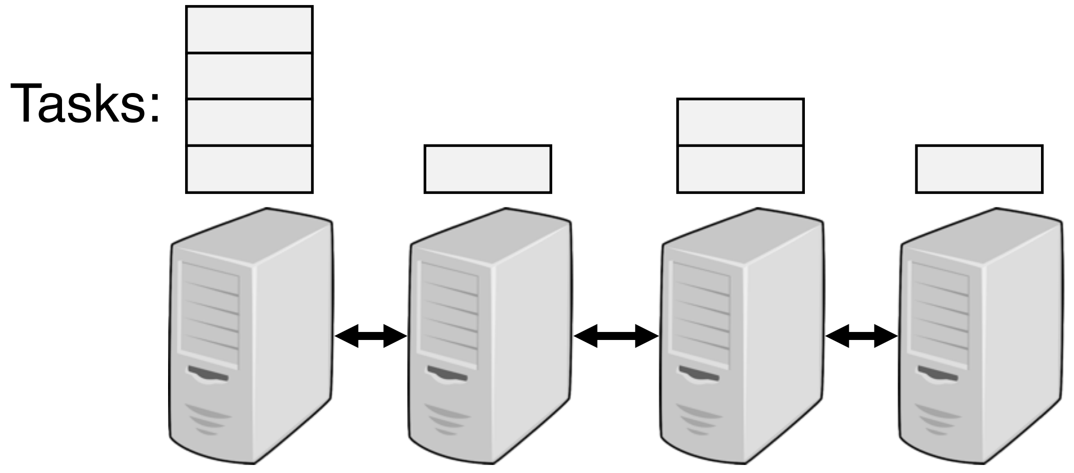


Examples of interference

- Shared hardware resources (e.g., memory, disk, network)
- Synchronization (e.g., locking)

Threats to Parallelism





Some nodes take more time to execute the assigned tasks, e.g.,

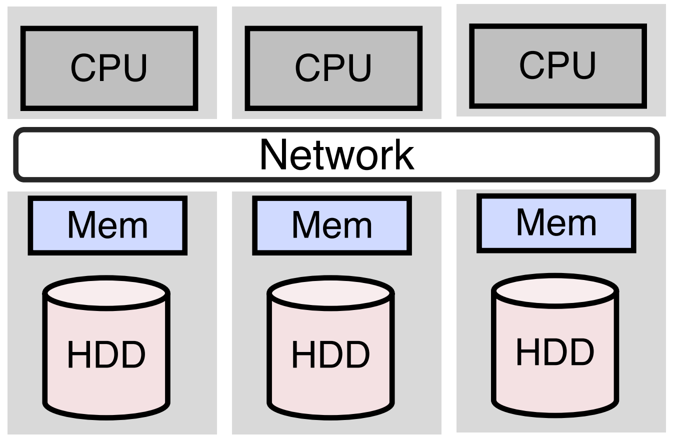
- More tasks assigned
- More computational intensive tasks assigned
- Node has slower hardware

Design Spectrum

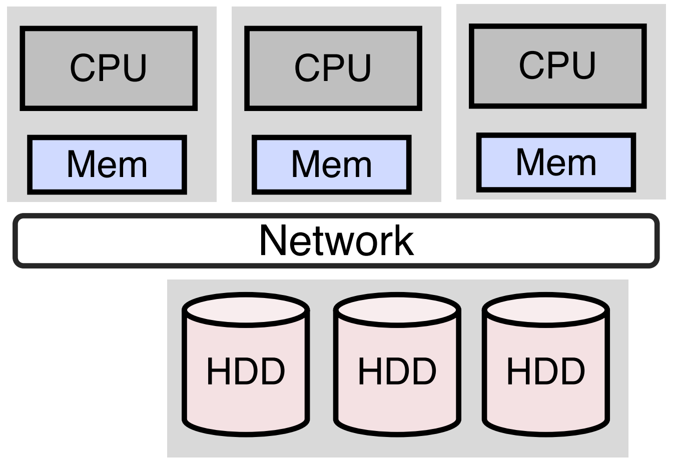
Shared-memory

Shared-disk

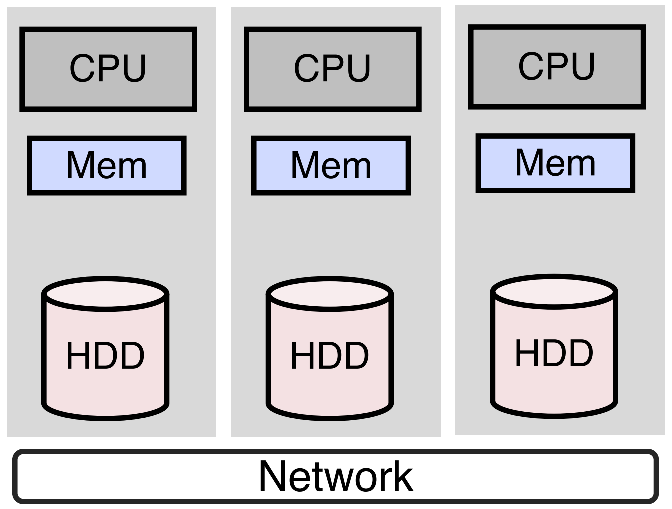
Shared-nothing



Shared Memory



Shared Disk



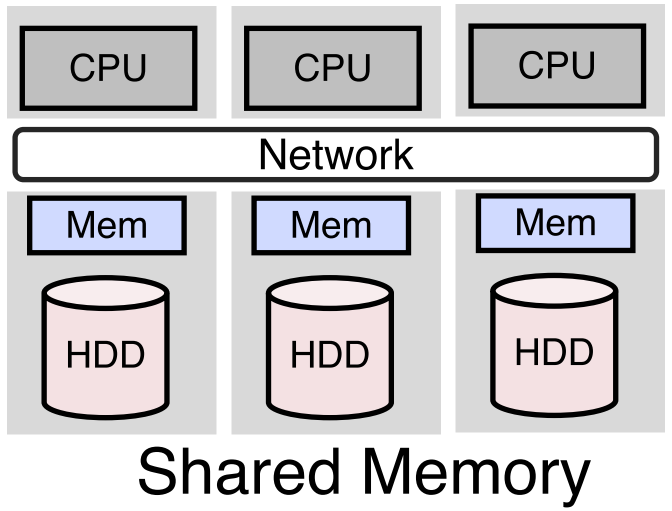
Shared Nothing

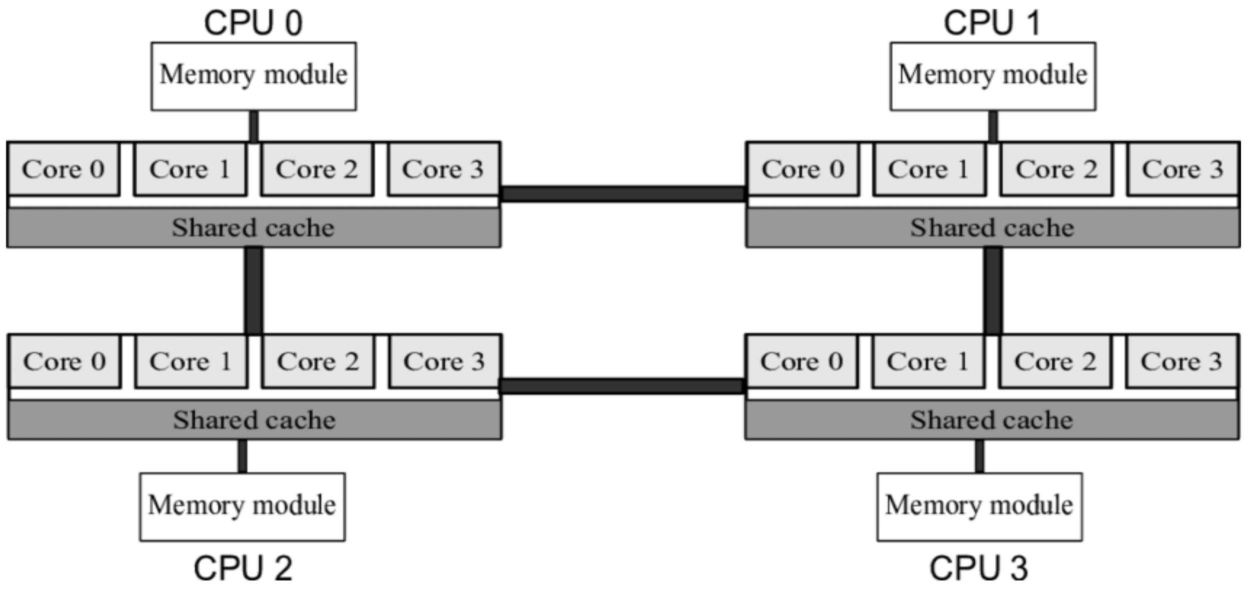
Design Spectrum – Shared Memory (SM)

All processors share direct access to a common global memory and to all disks

- Does not scale beyond a single server

Example: multicore processors



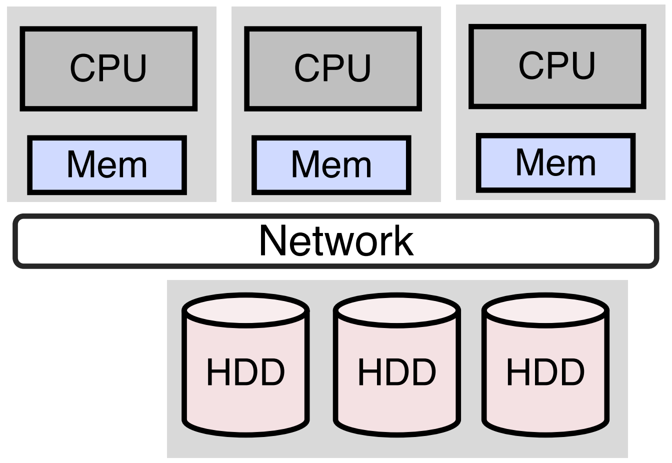


Design Spectrum – Shared Disk (SD)

Each processor has a private memory but has direct access to all disks

- Does not scale beyond tens of servers

Example: Network attached storage (NAS) and storage area network (SAN)



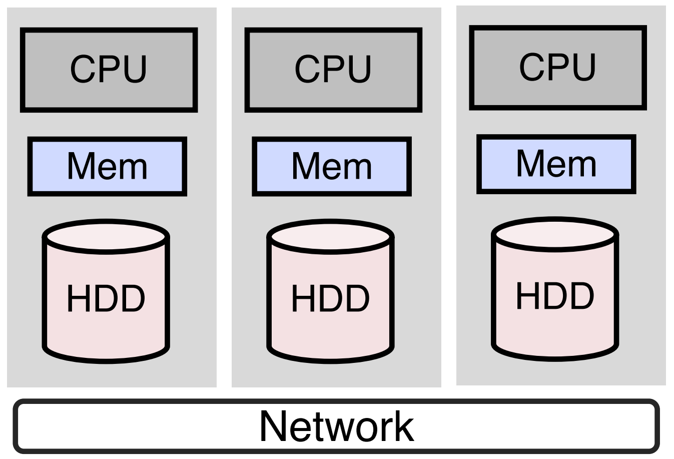
Shared Disk

Design Spectrum – Shared Nothing (SN)

Each memory and disk is owned by some processor that acts as a server for that data

- Scales to thousands of servers and beyond

Important optimization goal: minimize network data transfer



Shared Nothing

Legacy Software

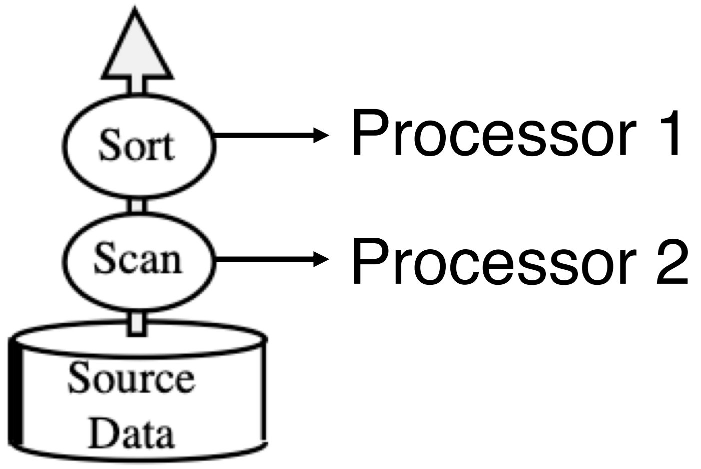
Old uni-processor software must be rewritten to benefit from parallelism

Most database programs are written in relational language SQL

- Can make SQL work on parallel hardware without rewriting
- Benefits of a high-level programming interface

Pipelined Parallelism

Pipelined parallelism: pipeline of operators



Advantages

- Avoid writing intermediate results back to disk

Disadvantages

- Small number of stages in a query
- Blocking operators: e.g., sort and aggregation
- Different speed: scan faster than join. Slowest operator becomes the bottleneck

Partitioned Parallelism

Round-robin partitioning

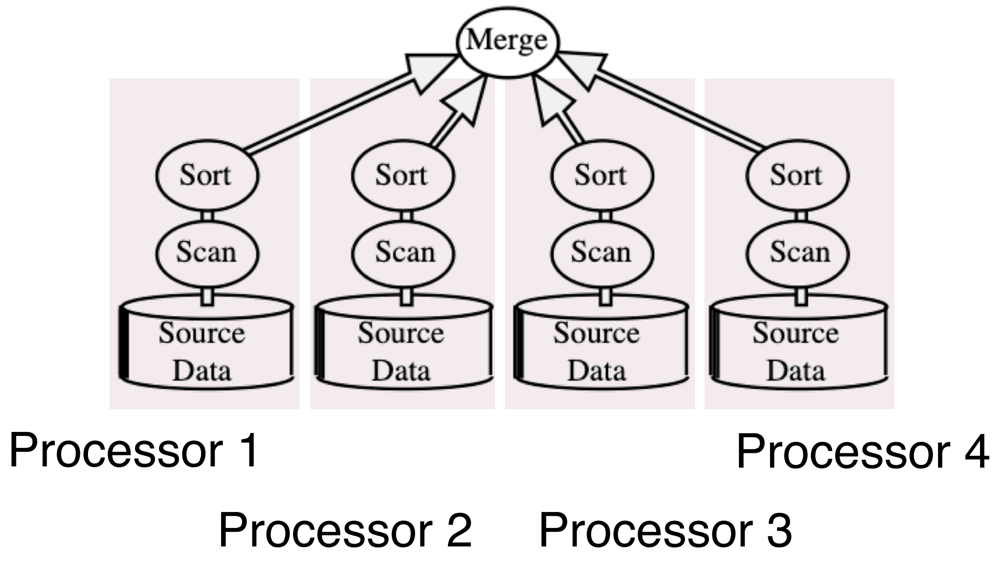
- map tuple *i* to disk (*i mode n*)

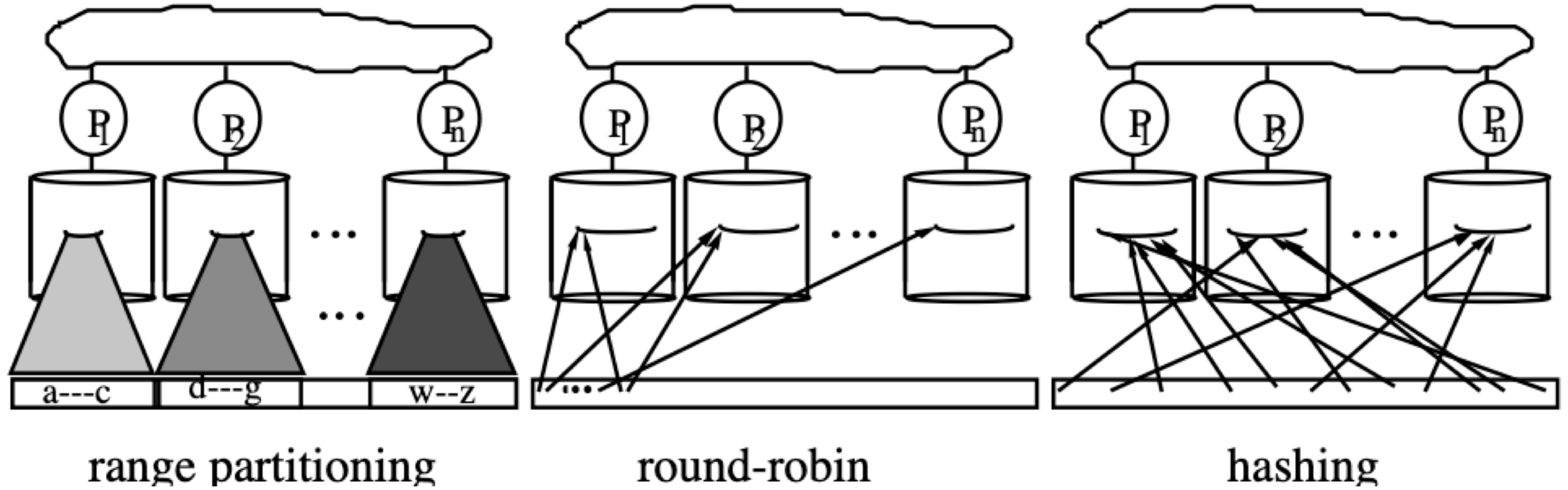
Hash partitioning

- map tuple i based on a hash function

Range partitioning

- map contiguous attribute ranges to disks
- benefits from clustering but suffers from skew

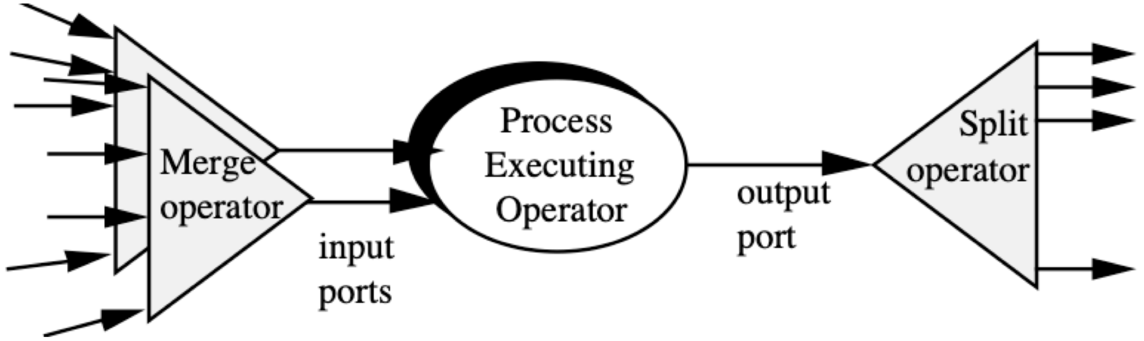


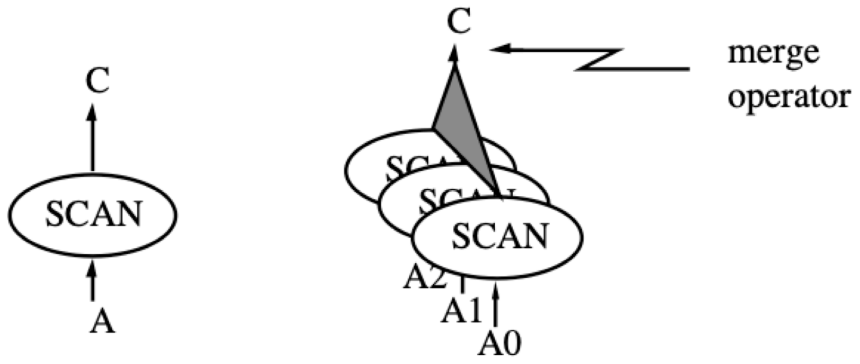


Parallelism within Relational Operators

Parallel data streams so that sequential operator code is not modified

- Each operator has a set of input and output ports
- Partition and merge these ports to sequential ports so that an operator is not aware of parallelism



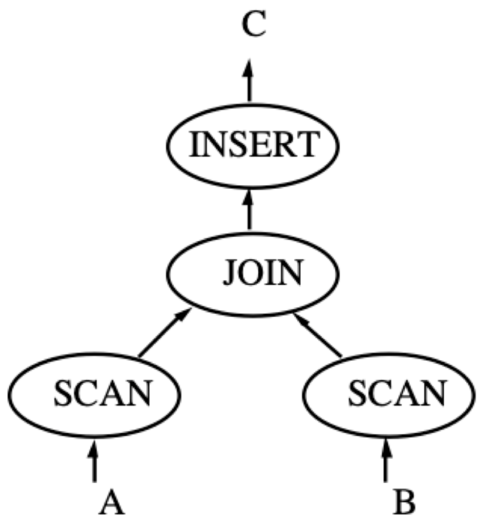


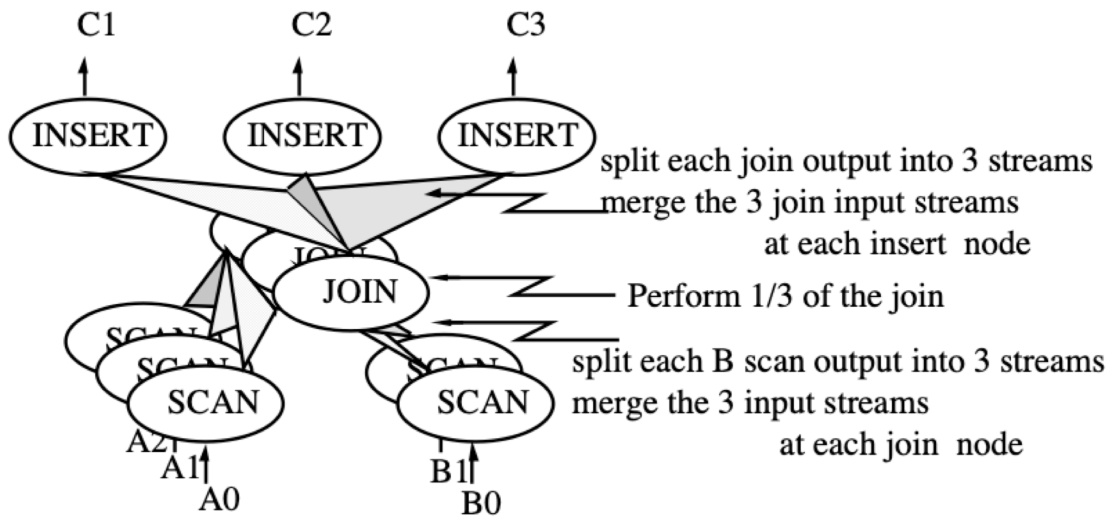
Parallelism within Relational Operators

Parallel data streams so that operator code is not modified

- Each operator has a set of input and output ports
- Partition and merge these ports to sequential ports so that an operator is not aware of parallelism

```
insert into C
  select    *
  from      A, B
  where     A.x = B.y;
```



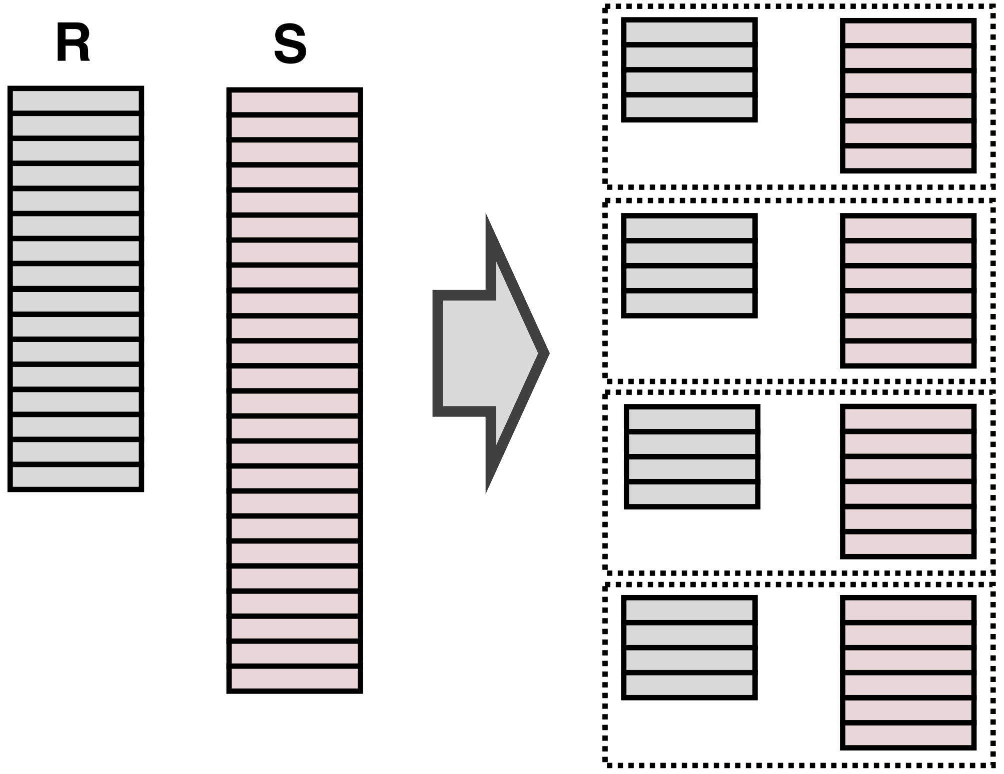


C1
C2
C3
INSERT
INSERT
INSERT
split each join output into 3 streams
merge the 3 join input streams at each insert node
JOIN
Perform 1/3 of the join
SCAN
SCAN
split each B scan output into 3 streams
merge the 3 input streams at each join node
A2
A1
A0
B1
B0

Specialized Parallel Operators

Parallel join algorithms

- Parallel sort-merge join
- Parallel hash join (e.g., radix join)

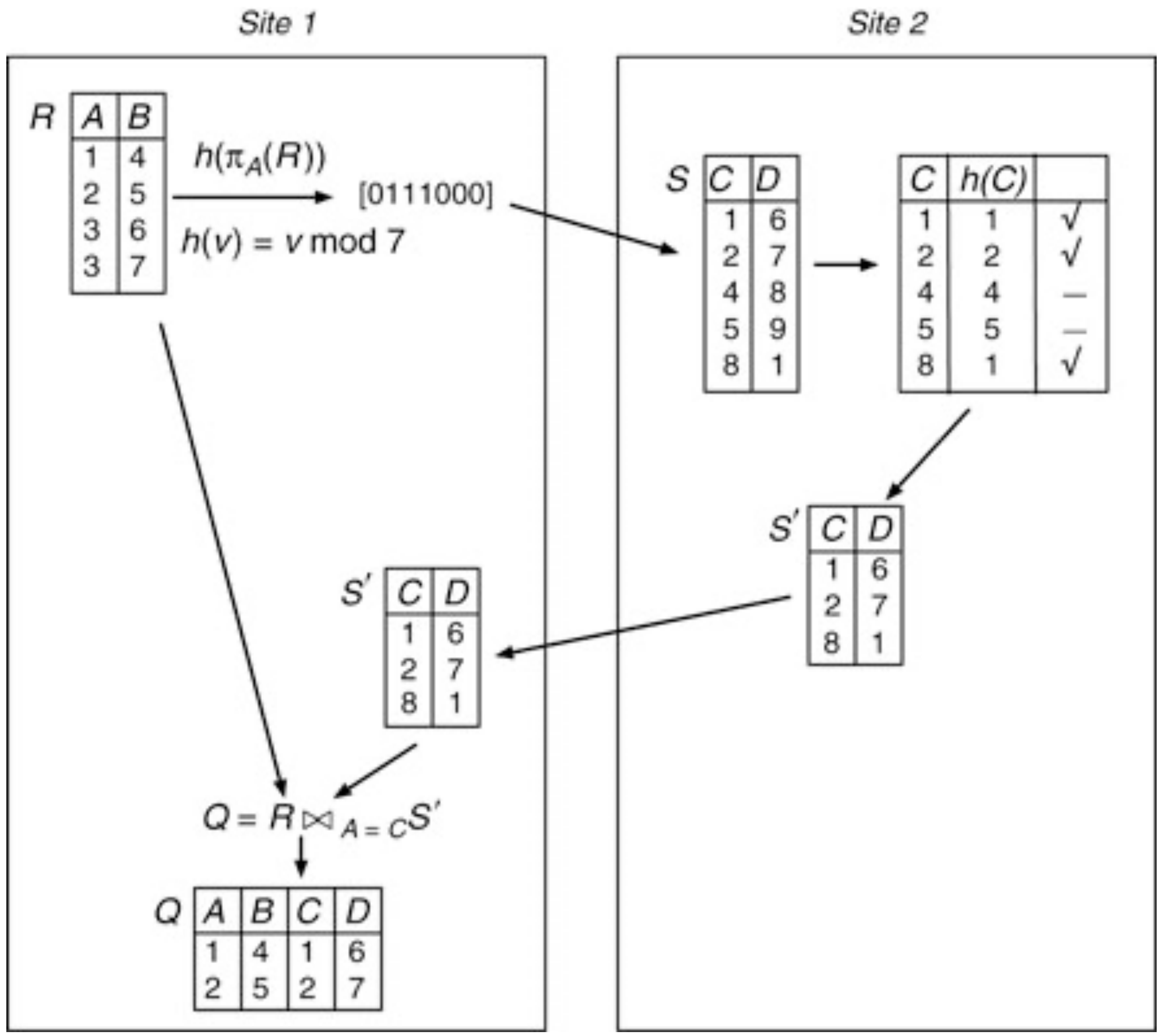


Specialized Parallel Operators

Semi-join

- Example:

```
SELECT *
 FROM T1, T2
 WHERE T1.A = T2.C
```

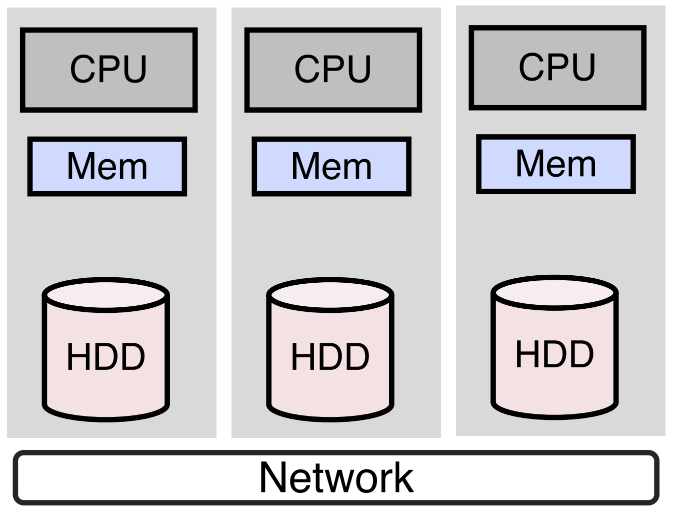


* Source: Sattler KU. (2009) Semijoin. Encyclopedia of Database Systems.

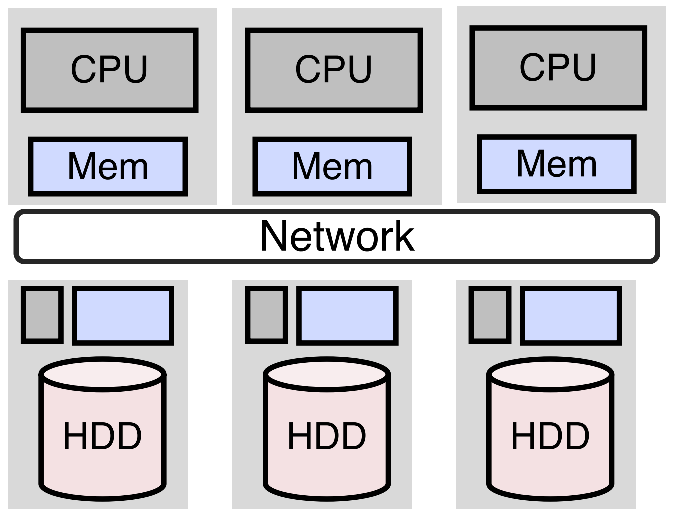
2010’s – Future

Cloud databases – Storage disaggregation

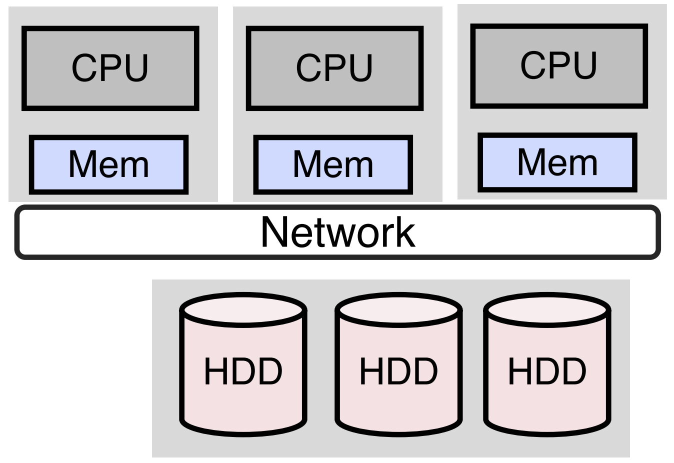
- Lower management cost
- Independent scaling of computation and storage



Shared Nothing



Storage Disaggregation



Shared Disk

Q/A – Parallel DBMSs

Parallel vs. distributed vs. cloud DBMS?

Valid for modern databases?

Batch processing for OLTP workloads?

Change of storage technology affects OLTP performance?

Will things change with the end of Moore’s law?

Extra challenges in the cloud?

Discussion

SQL, as a simple and high-level interface, enables database optimization across the hardware and software layers. Can you think of other examples of such high-level interfaces that enables flexible optimizations?

Can you think of any optimization opportunities for the storage-disaggregation architecture for OLTP or OLAP workloads?

Before Next Lecture

Look for teammates for the course project ☺

Submit discussion summary to https://wisc-cs764-f20.hotcrp.com

- Title: **Lecture 12 discussion. group ##**
- Authors: Names of students who joined the discussion

Deadline: Thursday 11:59pm

Submit review before next lecture

- Michael Stonebraker, et al., Mariposa: A Wide-Area Distributed Database System. VLDB 1996