

Parallel and Distributed Storage

Introduction

- Parallel machines have become quite common and affordable
 - prices of microprocessors, memory and disks have dropped sharply
- Data storage needs are growing increasingly large
 - user data at web-scale
 - 100's of millions of users, petabytes of data
 - transaction data are collected and stored for analysis.
 - multimedia objects like images/videos
- Parallel storage system requirements
 - storing large volumes of data
 - processing time-consuming decision-support queries
 - providing high throughput for transaction processing
 - Very high demands on **scalability** and **availability**

Parallel/Distributed Data Storage History

- 1980/1990s
 - Distributed database systems with tens of nodes
- 2000s:
 - Distributed file systems with 1000s of nodes
 - Millions of Large objects (100's of megabytes)
 - Web logs, images, videos, ...
 - Typically create/append only
 - Distributed data storage systems with 1000s of nodes
 - Billions to trillions of smaller (kilobyte to megabyte) objects
 - Social media posts, email, online purchases, ...
 - Inserts, updates, deletes
 - **Key-value stores**
- 2010s: Distributed database systems with 1000s of nodes

I/O Parallelism

- Reduce the time required to retrieve relations from disk by partitioning the relations on *multiple disks*, on *multiple nodes* (computers)
 - Our description focuses on parallelism across nodes
 - Same techniques can be used across disks on a node
- **Horizontal partitioning** – tuples of a relation are divided among many nodes such that some subset of tuple resides on each node.
 - Contrast with **vertical partitioning**, e.g. $r(A,B,C,D)$ with primary key A into $r1(A,B)$ and $r2(A,C,D)$
 - By default, the word partitioning refers to horizontal partitioning

I/O Parallelism

- Partitioning techniques (number of nodes = n):

Round-robin:

Send the i^{th} tuple inserted in the relation to node $i \bmod n$.

In a data center,

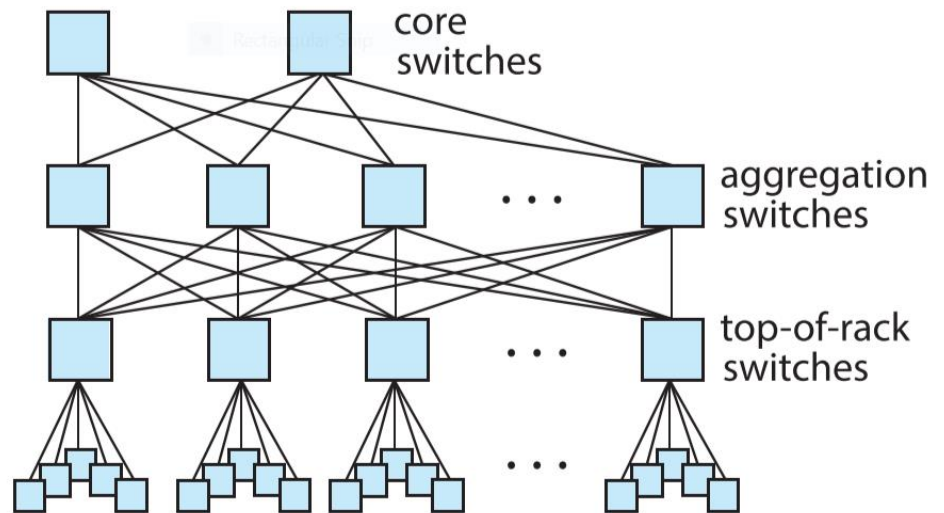
Nodes N0...N19 in Rack 1

Nodes N20...N39 in Rack 2

Total nodes $N = 40$

NID-Person relation has tuples
T0, T1, T2, T3 T39999

Question 4-1: Explain how tuples will be distributed into the nodes using round-robin technique.



(e) tree-like topology

- Partitioning techniques (number of nodes = n):

Hash partitioning:

- Choose one or more attributes as the partitioning attributes.
- Choose hash function h with range $0 \dots n - 1$
- Let i denote result of hash function h applied to the partitioning attribute value of a tuple. Send tuple to node i .

In a data center,

Nodes $N_0 \dots N_{19}$ in Rack 1

Nodes $N_{20} \dots N_{39}$ in Rack 2

Total nodes $N = 40$

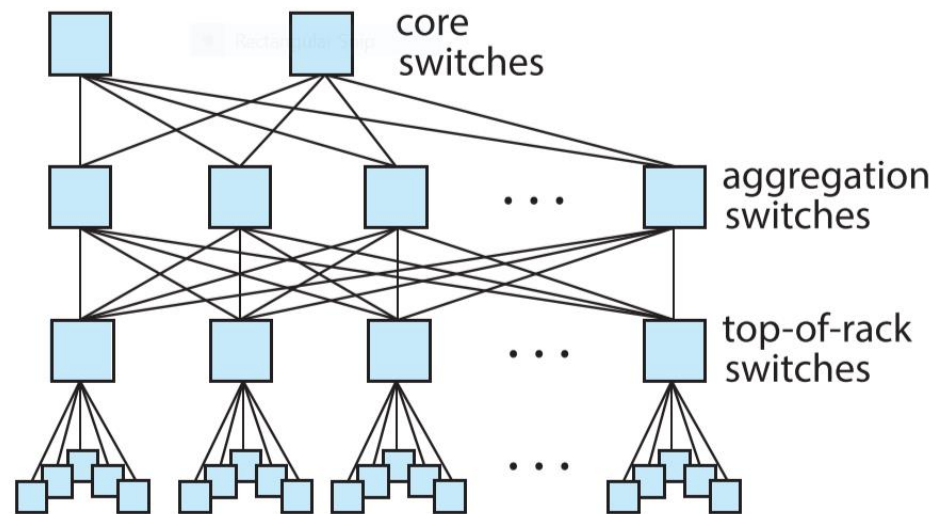
NID-Person(NID, name, DOB, street, city, district)

The tuples are:

$T_0, T_1, T_2, T_3 \dots T_{39999}$

Question 4-2: Explain how tuples will be distributed into the nodes using hash partitioning technique using

- NID
- Street, city, district.



(e) tree-like topology

Range Partitioning

Data Partitioning among Nodes

Range partitioning technique:

- Choose an attribute as the partitioning attribute.
- A partitioning vector $[v_0, v_1, \dots, v_{n-2}]$ is chosen.
- Let v be the partitioning attribute value of a tuple.
- Tuples such that $v_i \leq v < v_{i+1}$ go to node $i+1$.
- Tuples with $v < v_0$ go to node 0 and tuples with $v \geq v_{n-2}$ go to node $n-1$.
 - E.g., with a partitioning vector $[5, 11]$
 - a tuple with partitioning attribute value of 2 will go to node 0,
 - a tuple with value 8 will go to node 1, while
 - a tuple with value 20 will go to node 2.

In a data center,

Nodes N0...N9 in Rack 1

Nodes N20...N19 in Rack 2

Total nodes $N = 20$

Student(Id, name, DOB, street, city, district)

The tuples are:

T0, T1, T2, T3 T399

Id ranges from 202105001 to 202105400

Question 4-3

- a. Design a partition vector on Id for storage of student relation
- b. Find partitions P0, P1, P19

Range Partitioning

Data Partitioning among Nodes

Example: There are 4 nodes: N1, N2, N3 and N4 in a parallel database system. The schema of the relation person is as follows:

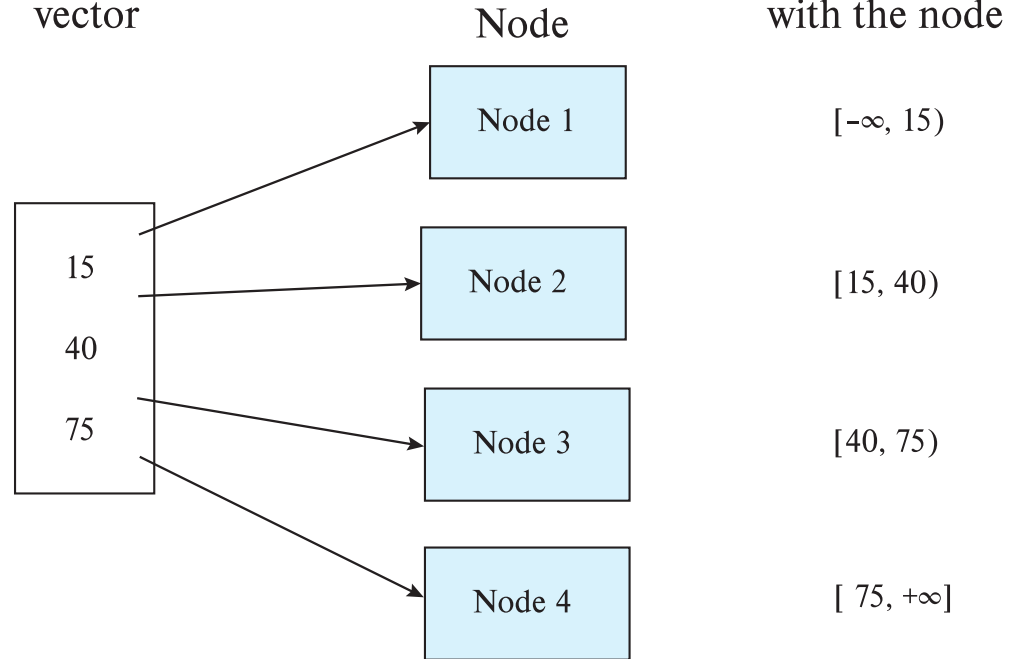
Person(NID, Name, Thana, District, Age)

Partition vector

$P[f\text{-age}] = [30, 50, 75]$

Perform range partitioning of person relation into 4 nodes.

Range partitioning
vector



Partitions:

Partition 1, age < 30 in Node N1

Partition 2, $30 \leq \text{age} < 50$ in Node N2

Partition 3, $50 \leq \text{age} < 75$ in Node N3

Partition 4, age ≥ 75 in Node N4

Range Partitioning

Data Partitioning among Nodes

Example: There are 4 nodes: N1, N2, N3 and N4 in a parallel database system. The schema of the relation **parents** is as follows:

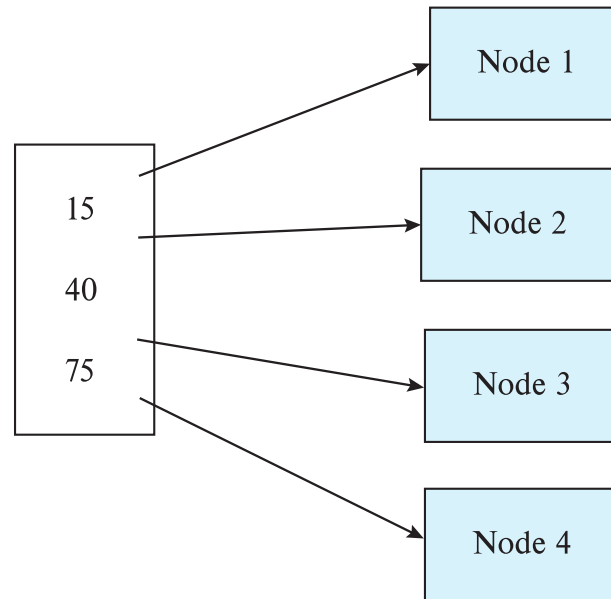
parents(m-NID, f-NID, c-NID, f-age)

Partition vector

$P[f\text{-age}] = [30, 50, 75]$

Perform range partitioning of parents relation into 4 nodes.

Range partitioning
vector



Range associated
with the node

$[-\infty, 15)$

$[15, 40)$

$[40, 75)$

$[75, +\infty]$

Partitions:

Partition 1, f-age < 30 in Node N1

Partition 2, $30 \leq f\text{-age} < 50$ in Node N2

Partition 3, $50 \leq f\text{-age} < 75$ in Node N3

Partition 4, f-age ≥ 75 in Node N4

Data Partitioning among Nodes

Partitions:

Person relation

Partition 1, age <30 in Node N1

Partition 2, 30≤age <50 in Node N2

Partition 3, 50≤age <75 in Node N3

Partition 4, age ≥75 in Node N4

Partitions:

Parents relation

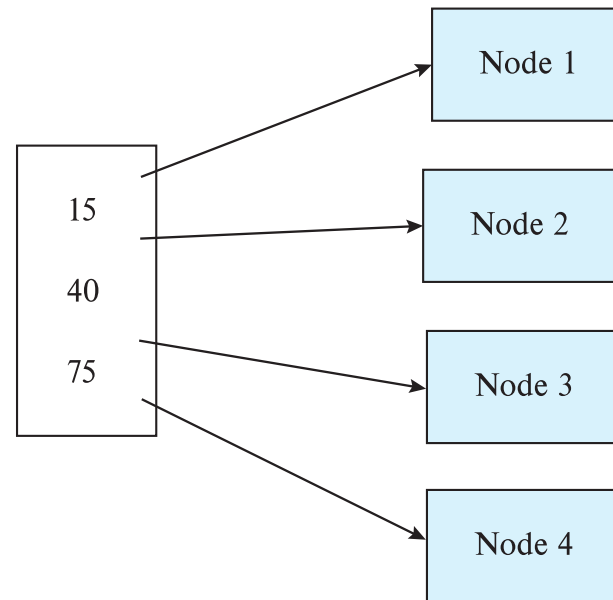
Partition 1, f-age <30 in Node N1

Partition 2, 30≤f-age <50 in Node N2

Partition 3, 50≤f-age <75 in Node N3

Partition 4, f-age ≥75 in Node N4

Range partitioning
vector



Range associated
with the node

$[-\infty, 15)$

$[15, 40)$

$[40, 75)$

$[75, +\infty]$

Person(NID, Name, Thana, District, Age), parents(m-NID, f-NID, c-NID, f-age)
process the query1: SELECT * FROM person r, parents s WHERE r.age = s.f-age

Question 4-4:

- Explain how query 1 will be executed using 4 nodes?
- Comments on speed-up and scale-up for the system.

Comparison of Partitioning Techniques

- Evaluate how well partitioning techniques support the following types of data access:

1. Scanning the entire relation.

2. Locating a tuple associatively – **point queries**.

- E.g., $r.A = 25$.

3. Locating all tuples such that the value of a given attribute lies within a specified range – **range queries**.

- E.g., $10 \leq r.A < 25$.

Select * from person

Select * from person where NID = 1234567890

Select * from person where NID > 1234567890

Question5-1: Explain the performance of Round robin partitioning technique for

- Scanning the entire relation
- Point query
- Range query

Comparison of Partitioning Techniques (Cont.)

Round robin:

- Best suited for sequential scan of entire relation on each query.
 - All nodes have almost an equal number of tuples; retrieval work is thus well balanced between nodes.
- All queries must be processed at all nodes

Comparison of Partitioning Techniques

- Evaluate how well partitioning techniques support the following types of data access:

1. Scanning the entire relation.

2. Locating a tuple associatively – **point queries**.

- E.g., $r.A = 25$.

3. Locating all tuples such that the value of a given attribute lies within a specified range – **range queries**.

- E.g., $10 \leq r.A < 25$.

Select * from person

Select * from person where NID = 1234567890

Select * from person where NID > 1234567890

Question5-2: Explain the performance of hash partitioning technique for

- Scanning the entire relation
- Point query
- Range query

Comparison of Partitioning Techniques (Cont.)

Hash partitioning:

- Good for sequential access
 - Assuming hash function is good, and partitioning attributes form a key, tuples will be equally distributed between nodes
- Good for point queries on partitioning attribute
 - Can lookup single node, leaving others available for answering other queries.
- Range queries inefficient, must be processed at all nodes

Comparison of Partitioning Techniques

- Evaluate how well partitioning techniques support the following types of data access:

1. Scanning the entire relation.

2. Locating a tuple associatively – **point queries**.

- E.g., $r.A = 25$.

3. Locating all tuples such that the value of a given attribute lies within a specified range – **range queries**.

- E.g., $10 \leq r.A < 25$.

Select * from person

Select * from person where NID = 1234567890

Select * from person where NID > 1234567890

Question5-3: Explain the performance of range partitioning technique for

- Scanning the entire relation
- Point query
- Range query

Comparison of Partitioning Techniques (Cont.)

Range partitioning:

- Provides data clustering by partitioning attribute value.
 - Good for sequential access
 - Good for point queries on partitioning attribute: only one node needs to be accessed.
- For range queries on partitioning attribute, one to a few nodes may need to be accessed
 - Remaining nodes are available for other queries.
 - Good if result tuples are from one to a few blocks.
 - But if many blocks are to be fetched, they are still fetched from one to a few nodes, and potential parallelism in disk access is wasted
 - Example of **execution skew**.

Handling Small Relations

- Partitioning not useful for small relations which fit into a single disk block or a small number of disk blocks
 - Instead, assign the relation to a single node, or
 - Replicate relation at all nodes
- For medium sized relations, choose how many nodes to partition across based on size of relation
- Large relations typically partitioned across all available nodes.

Types of Skew

- **Data-distribution skew:** some nodes have many tuples, while others may have fewer tuples. Could occur due to
 - **Attribute-value skew.**
 - Some partitioning-attribute values appear in many tuples
 - E.g., partitioning on age in ecommerce database
Game related DB, $P[\text{age}] = [10, 20, 30, 40, 50, 60, 70]$
Comments on the partition vector???
 - All the tuples with the same value for the partitioning attribute end up in the same partition.
 - Can occur with range-partitioning and hash-partitioning.

Types of Skew

- **Partition skew.**
 - Imbalance, even without attribute –value skew
 - Badly chosen range-partition vector may assign too many tuples to some partitions and too few to others.
 - Less likely with hash-partitioning

Types of Skew (Cont.)

- Note that **execution skew** can occur even without data distribution skew
 - E.g. relation range-partitioned on date, and most queries access tuples with recent dates
- Data-distribution skew can be avoided with range-partitioning by creating **balanced range-partitioning vectors**
- We assume for now that partitioning is **static**, that is partitioning vector is created once and not changed
 - Any change requires **repartitioning**
 - **Dynamic partitioning** once allows partition vector to be changed in a continuous manner
 - More on this later

Question 6-1: Explain the various reasons and types of skews in

- a. Hash partitioning
- b. Range partitioning

Handling Skew in Range-Partitioning

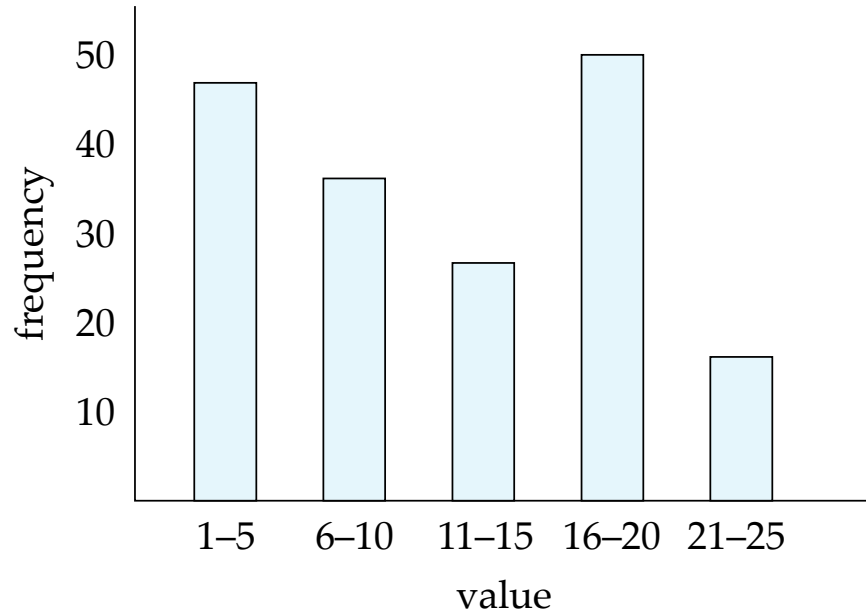
- To create a **balanced partitioning vector**
 - Sort the relation on the partitioning attribute.
 - Construct the partition vector by scanning the relation in sorted order as follows.
 - After every $1/n^{th}$ of the relation has been read, the value of the partitioning attribute of the next tuple is added to the partition vector.
 - n denotes the number of partitions to be constructed.
 - **Imbalances can result if duplicates are present in partitioning attributes.**

Handling Skew in Range-Partitioning

- To reduce cost
 - Partitioning vector can be created using a random sample of tuples
 - Alternatively histograms can be used to create the partitioning vector

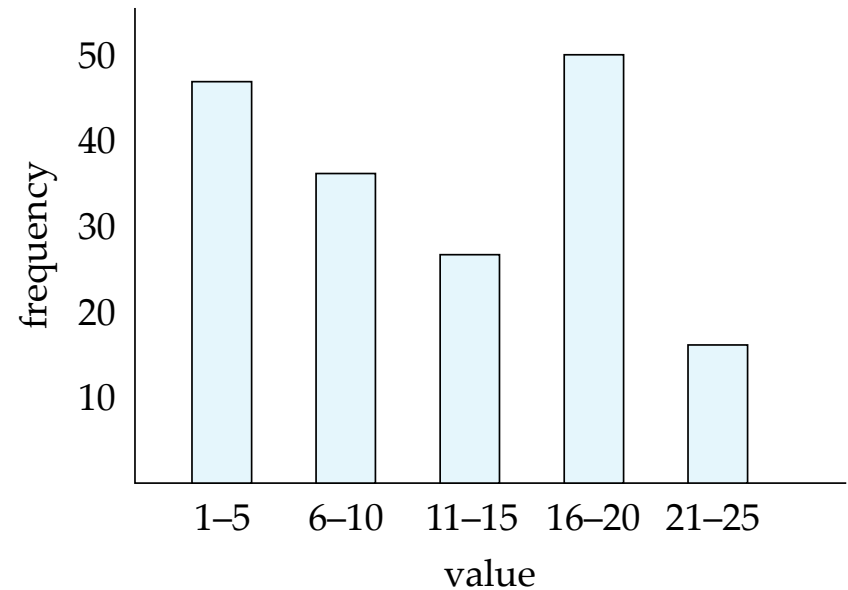
Histograms

- Histogram on attribute *age* of relation *person*



- **Equi-width** histograms
- **Equi-depth** histograms
 - break up range such that each range has (approximately) the same number of tuples
 - E.g. (4, 8, 14, 19)
- Assume uniform distribution within each range of the histogram
- Create partitioning vector for required number of partitions based on histogram

Histograms



Question 6-2:

- What is the type of the given histogram?
- Define an approximate partition vector using the histogram for a parallel system with nodes N0, N1, N2 and N3.
- Draw the approximate histogram of your partition vector.