The Gamma Database Machine

AUTHORS:

DEWITT, GHANDEHARIZADEH, SCHNEIDER, BRICKER, HSIAO, RASMUSSEN

PRESENTED BY:

JITIN DUA

Outline

- Motivation
- System Architecture
- Declustering
- Process Structure
- Query Processing
- Recovery
- Failure Management
- Performance Studies

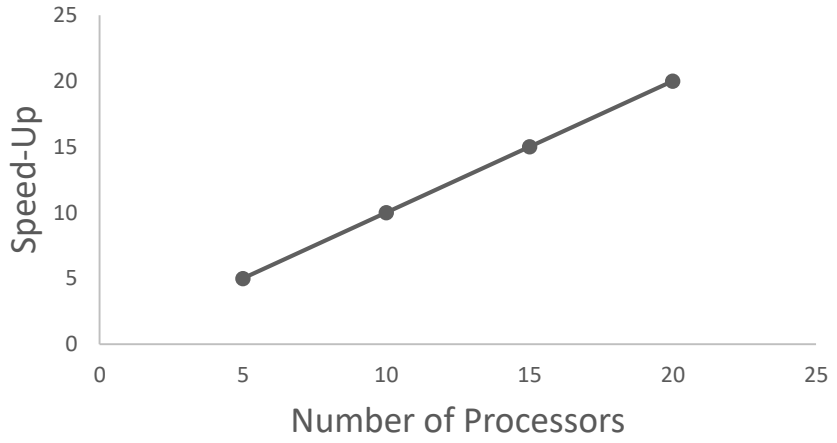
Motivation

Parallelizing Databases

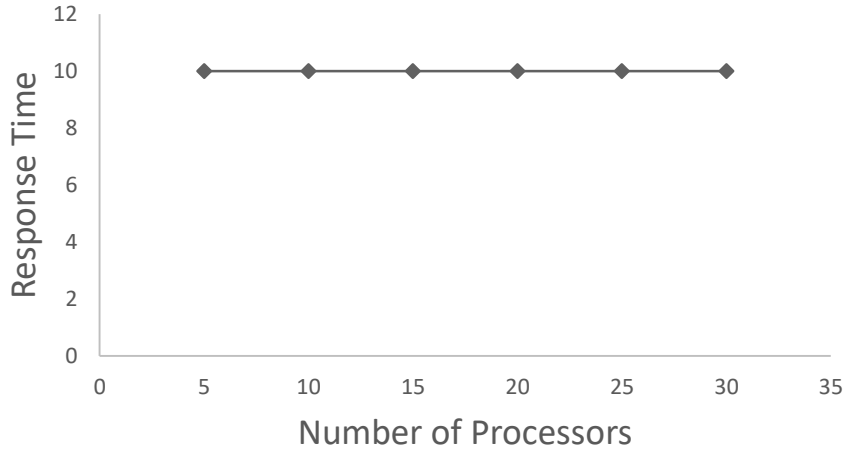
- Faster response time
- More storage
- Reduce System cost
- High Scalability

Goal

- Linear Speedup



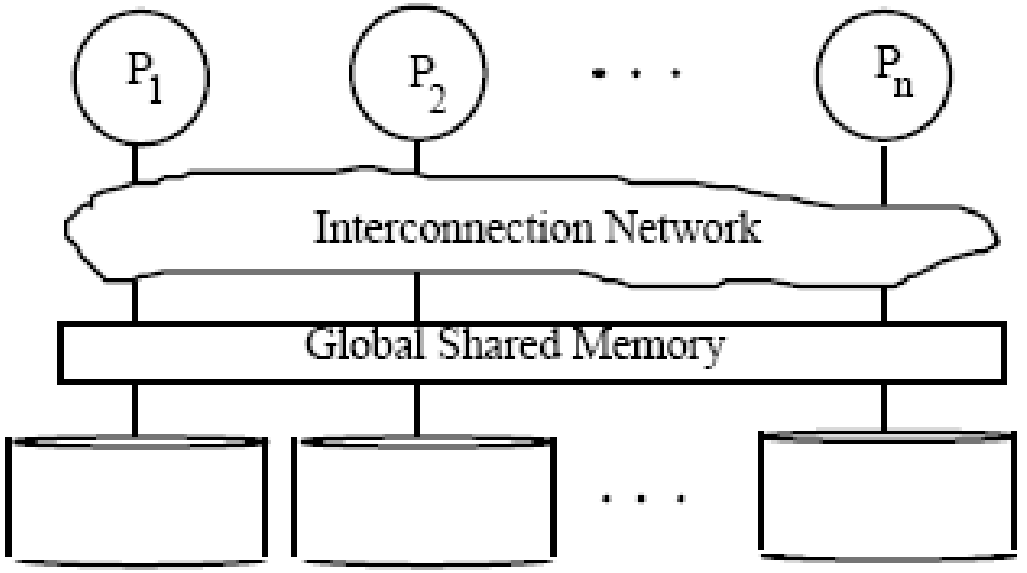
- Constant Scale-Up



System Architecture

Shared-memory architecture

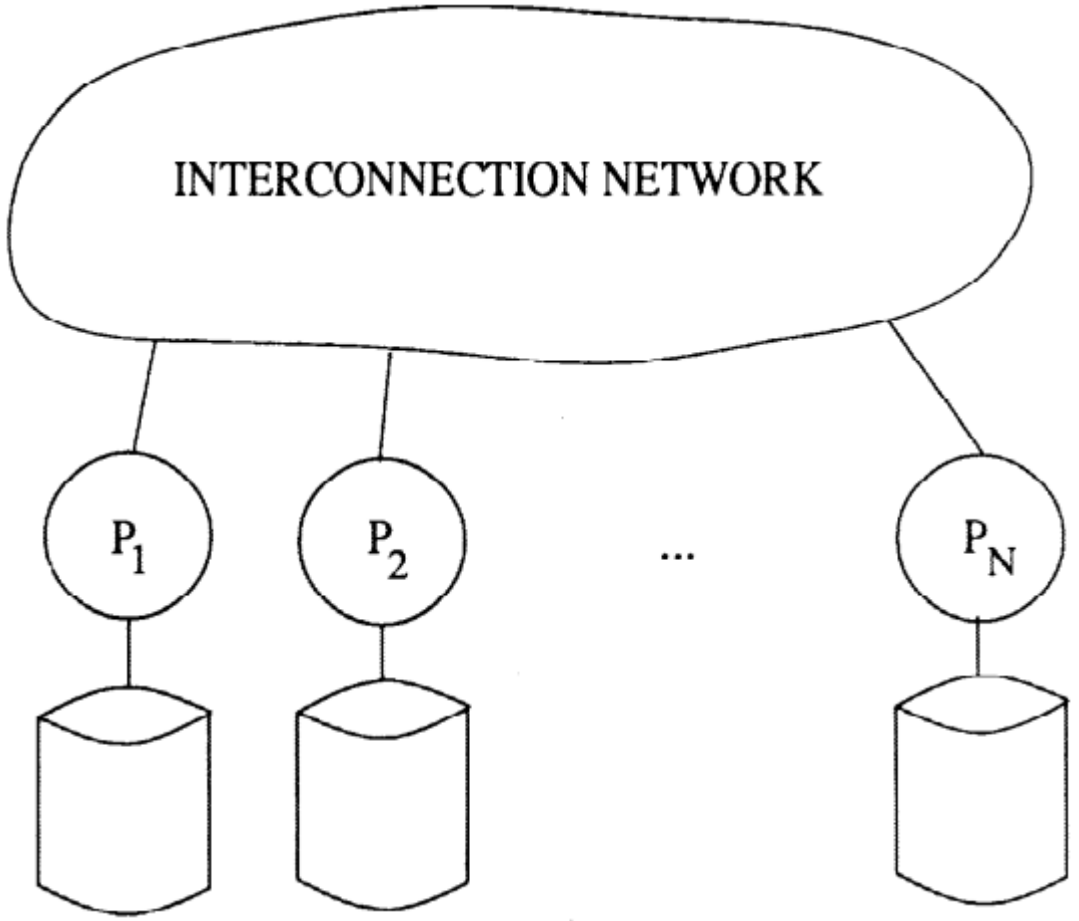
- Poor scalability
- Custom hardware/systems



Shared Memory Multiprocessor

Shared-nothing architecture

- Processors do not share memory
- Communication by messages through an interconnected network
- Commodity hardware (cheap!)



Declustering

Declustering

- Horizontal partitioning of database
- Attribute-less partitioning
  - Round robin
- Attribute Partitioning
  - Hash
  - Range

Round-robin

Name	Age	Salary
Peter	32	55K
Jeff	24	45K
Clark	43	60K
Kevin	54	140K
Anne	59	160K
Mike	18	38K

Name	Age	Salary
Peter	32	55K
Kevin	54	140K

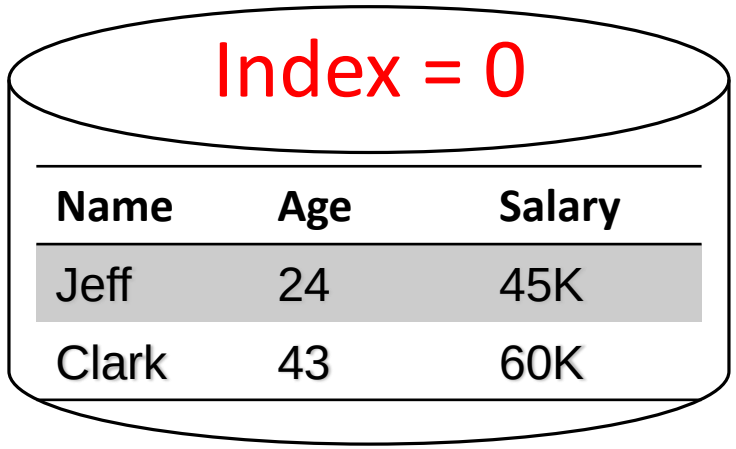
Name	Age	Salary
Jeff	24	45K
Anne	59	160K

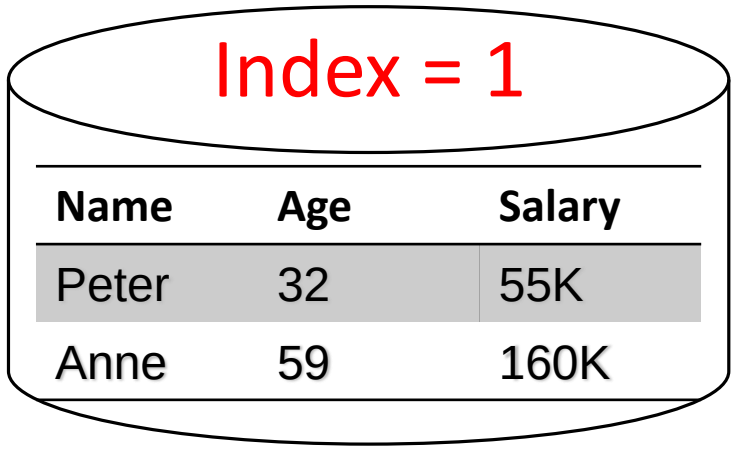
Name	Age	Salary
Clark	43	60K
Mike	18	38K

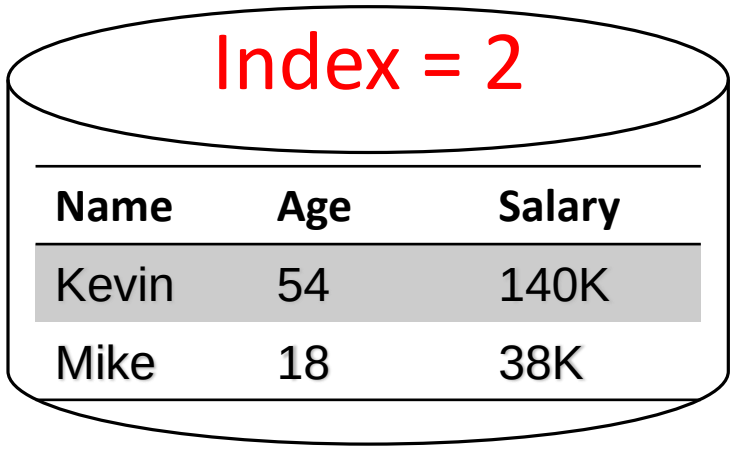
Hash based

Name	Age	Salary
Peter	32	55K
Jeff	24	45K
Clark	43	60K
Kevin	54	140K
Anne	59	160K
Mike	18	38K

salary % 3







Hash based

- Selections with equality predicates referencing the partitioning attribute are directed to a single node:
  - Retrieve Emp where salary = 60K

```
SELECT *
FROM   Emp
WHERE salary=60K
```

- Equality predicates referencing a non-partitioning attribute and range predicates are directed to all nodes:
  - Retrieve Emp where age = 20
  - Retrieve Emp where salary < 20K

```
SELECT *
FROM   Emp
WHERE salary<20K
```

Range based

Name	Age	Salary
Peter	32	55K
Jeff	24	45K
Clark	43	60K
Kevin	54	140K
Anne	59	160K
Mike	18	38K

< 50K

Name	Age	Salary
Jeff	24	45K
Mike	18	38K

50K – 100K

Name	Age	Salary
Peter	32	55K
Clark	43	60K

> 100K

Name	Age	Salary
Kevin	54	140K
Anne	59	160K

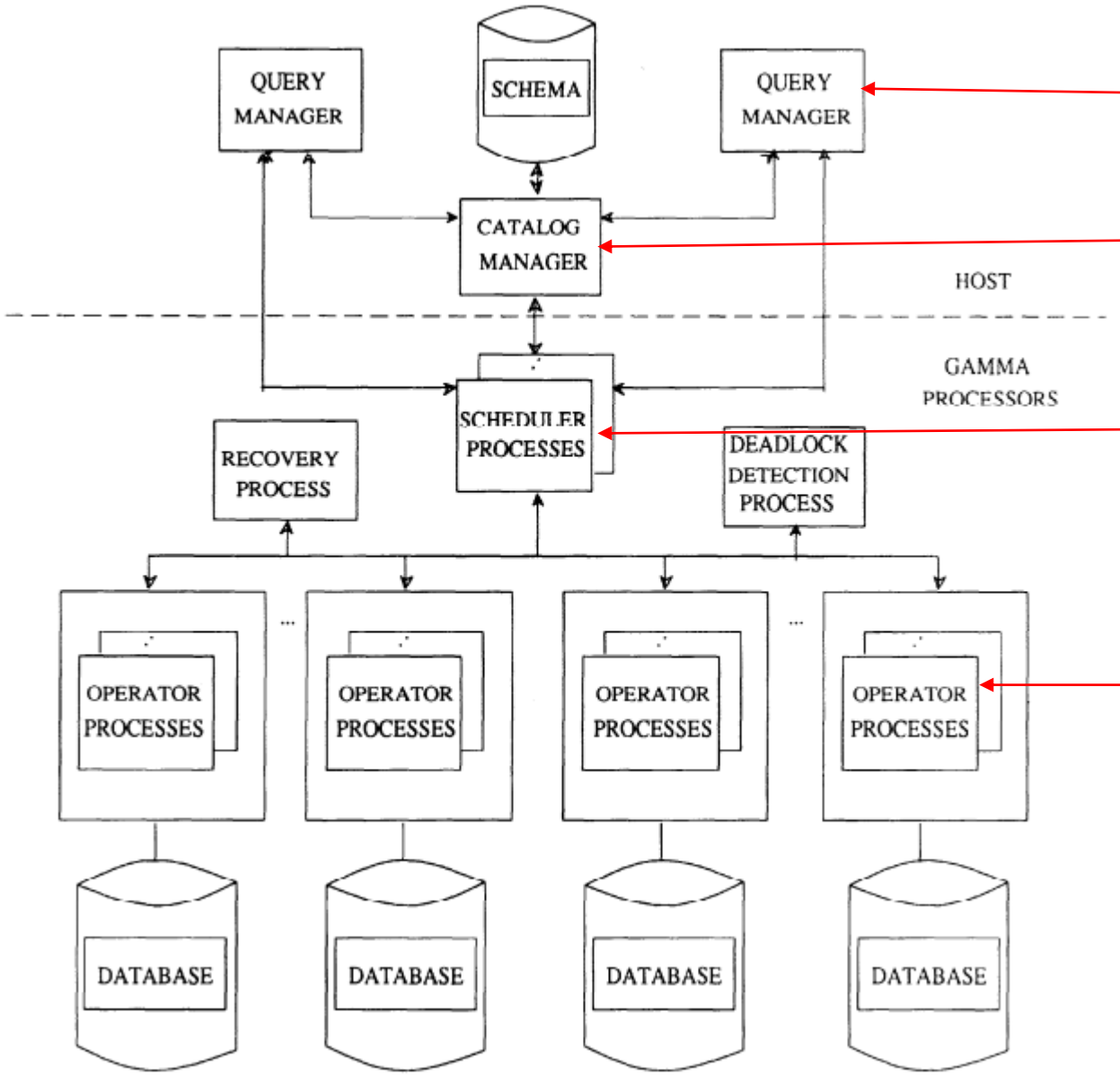
Range based

- Equality and range predicates referencing the partitioning attribute are directed to a subset of nodes:
  - Retrieve Emp where salary = 60K
  - Retrieve Emp where salary < 20K

  In the example, both queries are directed to one node.

- Predicates referencing a non-partitioning attribute are directed to all nodes.

Process Structure



Query parsing, optimization and compilation.

Schema information for each Database.

Schedules multi-site queries.

Processes queries.

Query Processing

Selection

Selection operation on set of relevant nodes

Aggregate

Each processor computes its piece of result in parallel.

The partial results are sent to a single site which combines it to get the final answer.

Update

Replace/delete/append.

Uses standard techniques, except when replace operator modifies the partitioning attribute.

Modified tuple is passed through a split table to determine destination node.

Join

StuID	Name
1	Peter
2	Jeff
3	Clark
4	Kevin

StuID	Grade	Sub
1	B	English
2	A	English
3	C	English
1	A	Science
2	B	Science
4	B	Science

StuID % 2

Partition on join attribute (StuID) into N buckets.

Join

StuID	Name
2	Jeff
4	Kevin

StuID	Name
1	Peter
3	Clark

StuID	Grade	Sub
2	A	English
2	B	Science
4	B	Science

StuID	Grade	Sub
1	B	English
3	C	English
1	A	Science

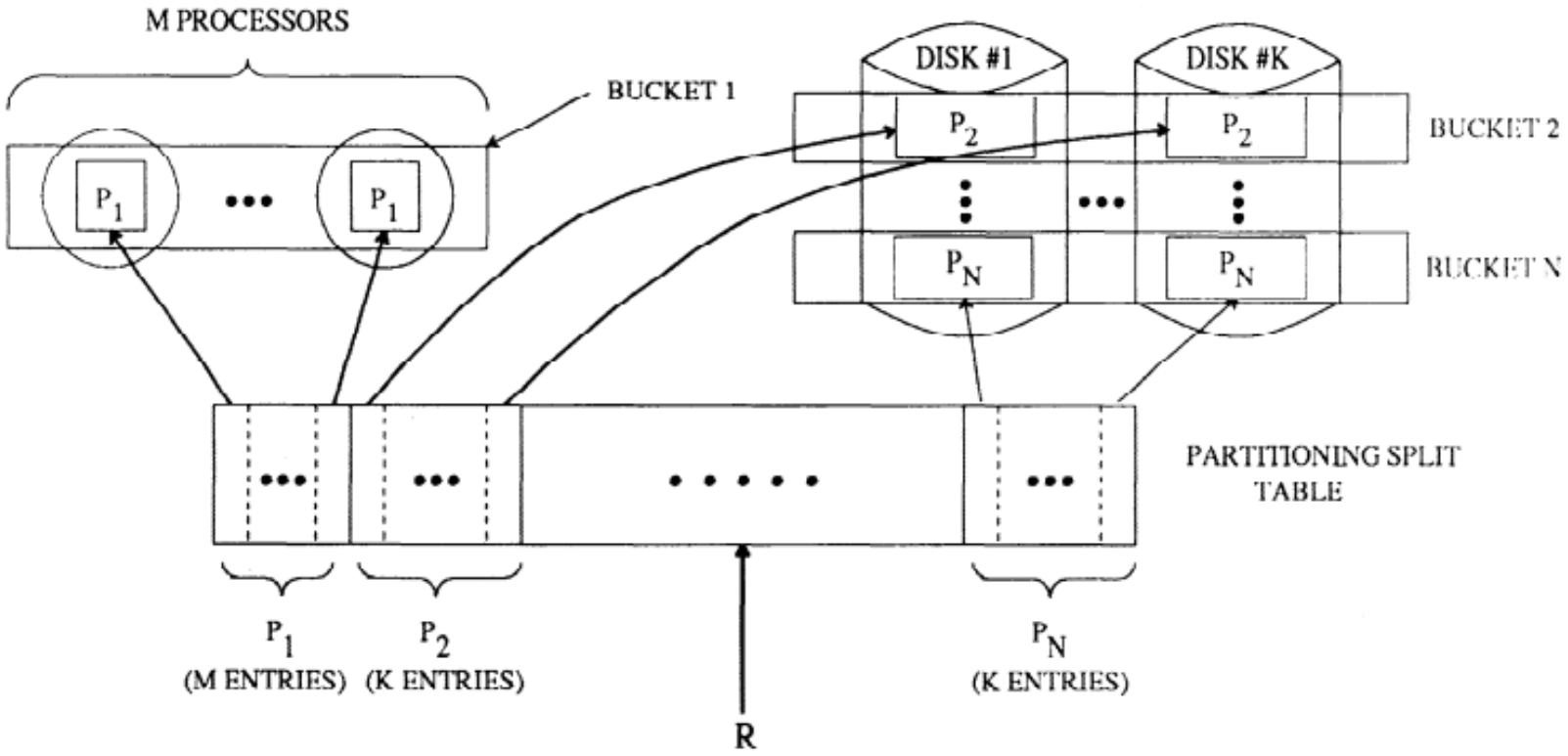
Process one bucket at a time using m processors and store relation using declustering.

Join

StuID	Name	Grade	Sub
2	Jeff	A	English
2	Jeff	B	Science
4	Kevin	B	Science

StuID	Name	Grade	Sub
1	Peter	B	English
3	Clark	C	English
1	Peter	A	Science

Join


M PROCESSORS
BUCKET 1
DISK #1
DISK #K
BUCKET 2
BUCKET N
PARTITIONING SPLIT TABLE
(M ENTRIES)
(K ENTRIES)
(K ENTRIES)
R

Recovery

Recovery

- Node-LSN pair as log index to identify which node created which log.
- If M log processors are used, Query processor i directs log to (i % M) log processor.
- ARIES algorithm.
  - Write ahead logging

Failure Management

Declustering Schemes

- Interleaved declustering
- Chained declustering

Interleaved Declustering

- Backup copy is constructed by:
  - Dividing nodes into clusters
  - Partition a primary fragment into the remaining nodes of the cluster.

Node	Cluster 0				Cluster 1			
	0	1	2	3	4	5	6	7
Primary Copy	R0	R1	R2	R3	R4	R5	R6	R7
Backup Copy		r0.0	r0.1	r0.2		r4.0	r4.1	r4.2
	r1.2		r1.0	r1.1	r5.2		r5.0	r5.1
	r2.1	r2.2		r2.0	r6.1	r6.2		r6.0
	r3.0	r3.1	r3.2		r7.0	r7.1	r7.2	

Interleaved Declustering

- On failure, query load re-directed to backup nodes in cluster.

Load = 4/3

Node	Cluster 0				Cluster 1			
	0	1	2	3	4	5	6	7
Primary Copy	R0	R1	R2	R3	R4	R5	R6	R7
Backup Copy		r0.0	r0.1	r0.2		r4.0	r4.1	r4.2
	r1.2		r1.0	r1.1	r5.2		r5.0	r5.1
	r2.1	r2.2		r2.0	r6.1	r6.2		r6.0
	r3.0	r3.1	r3.2		r7.0	r7.1	r7.2	

Interleaved Declustering

- On failure, query load re-directed to backup nodes in cluster.
- Second failure before recovery in a cluster causes unavailability.
- Large cluster size improves failure load balancing but increases risk of data being unavailable.

Node	Cluster 0				Cluster 1			
	0	1	2	3	4	5	6	7
Primary Copy	R0	R1	R2	R3	R4	R5	R6	R7
Backup Copy		r0.0	r0.1	r0.2		r4.0	r4.1	r4.2
	r1.2		r1.0	r1.1	r5.2		r5.0	r5.1
	r2.1	r2.2		r2.0	r6.1	r6.2		r6.0
	r3.0	r3.1	r3.2		r7.0	r7.1	r7.2	

Data unavailable

Chained Declustering

Given a primary fragment Ri, its backup copy is assigned to node (i+1) mod M (M is the number of nodes in the relation cluster).

Node	0	1	2	3	4	5	6	7
Primary Copy	R0	R1	R2	R3	R4	R5	R6	R7
Backup Copy	r7	r0	r1	r2	r3	r4	r5	r6

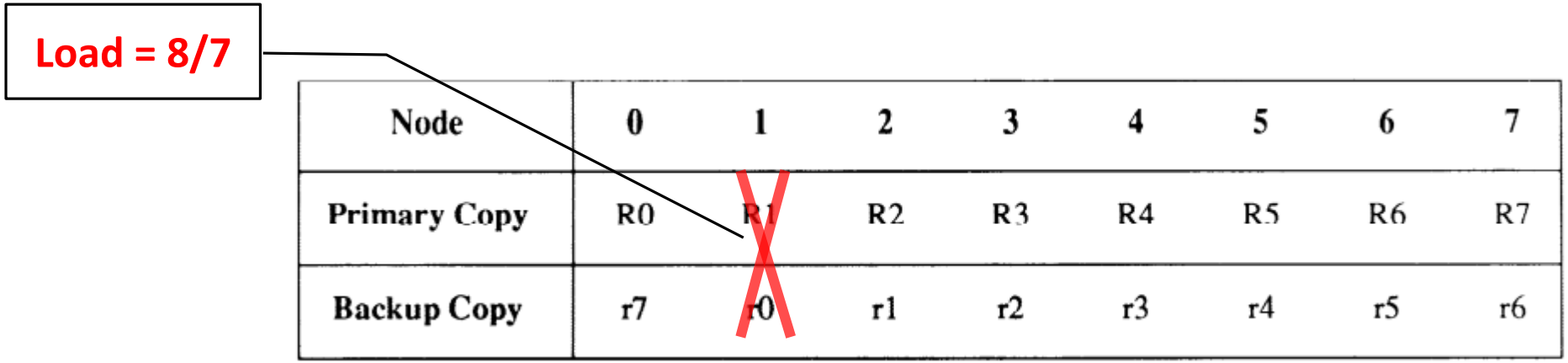
Chained Declustering

Node	0	1	2	3	4	5	6	7
Primary Copy	R0	R1	R2	R3	R4	R5	R6	R7
Backup Copy	r7	r0	r1	r2	r3	r4	r5	r6

Node	0	1	2	3	4	5	6	7
Primary Copy	R0	---	$\frac{1}{7}$R2	$\frac{2}{7}$R3	$\frac{3}{7}$R4	$\frac{4}{7}$R5	$\frac{5}{7}$R6	$\frac{6}{7}$R7
Backup Copy	$\frac{1}{7}$r7	---	r1	$\frac{6}{7}$r2	$\frac{5}{7}$r3	$\frac{4}{7}$r4	$\frac{3}{7}$r5	$\frac{2}{7}$r6

Chained Declustering

- On failure, query load re-directed to backup nodes in cluster.



Chained Declustering

- On failure, query load re-directed to backup nodes in cluster.
- Any two node failures in a relation cluster does not result in data un-availability.

Node	0	1	2	3	4	5	6	7
Primary Copy	R0	R1	R2	R3	R4	R5	R6	R7
Backup Copy	r7	r0	r1	r2	r3	r4	r5	r6

Data available

Chained Declustering

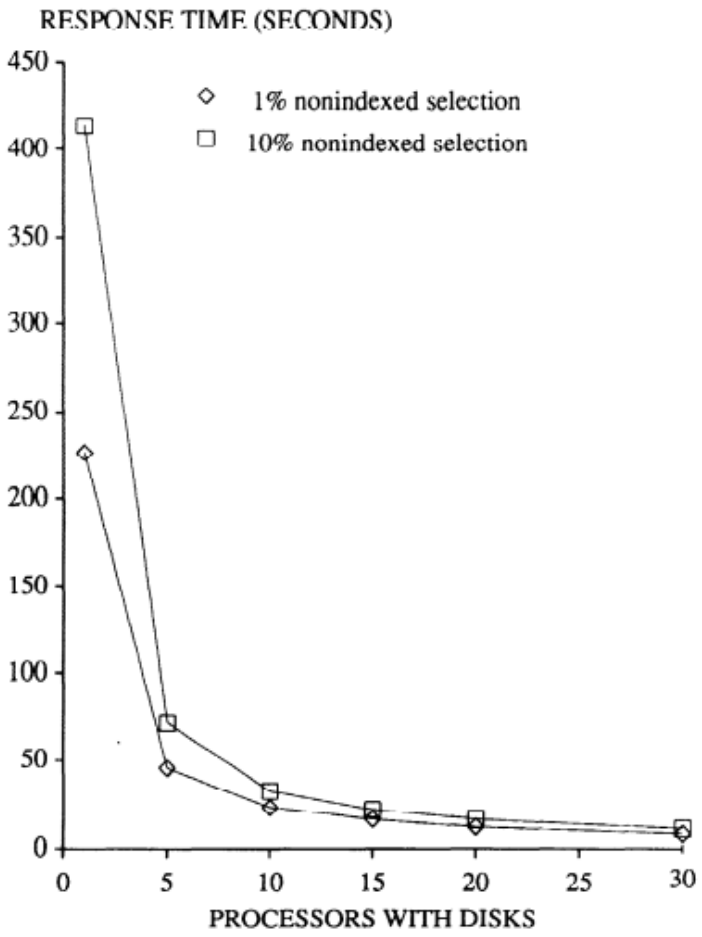
- On failure, query load re-directed to backup nodes in cluster.
- Any two node failures in a relation cluster does not result in data un-availability.
- Two adjacent nodes must fail in order for data to become unavailable. Can have higher cluster size.

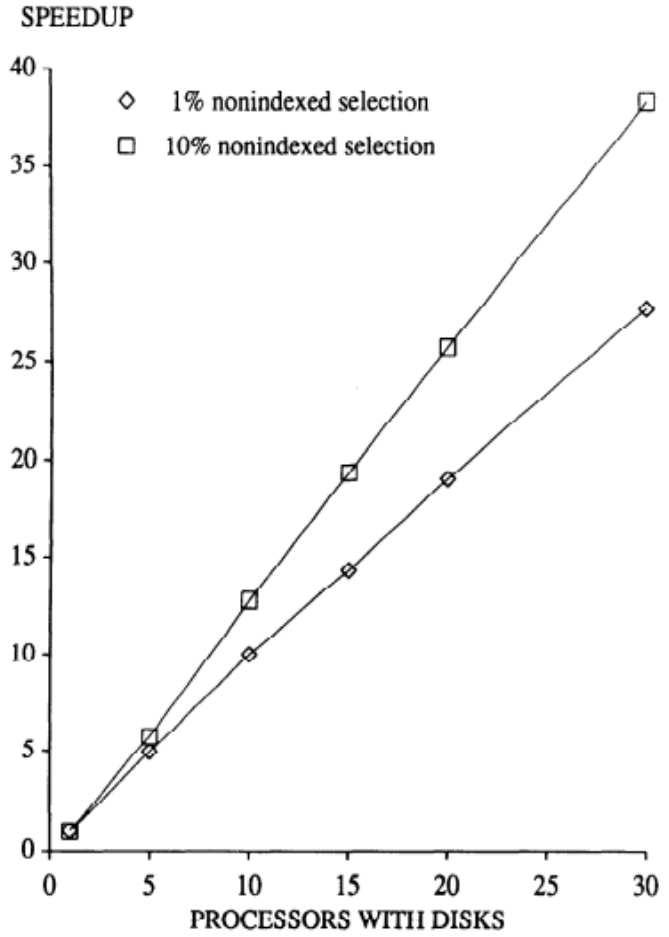
Node	0	1	2	3	4	5	6	7
Primary Copy	R0	R1	R2	R3	R4	R5	R6	R7
Backup Copy	r7	r0	r1	r2	r3	r4	r5	r6

Data unavailable

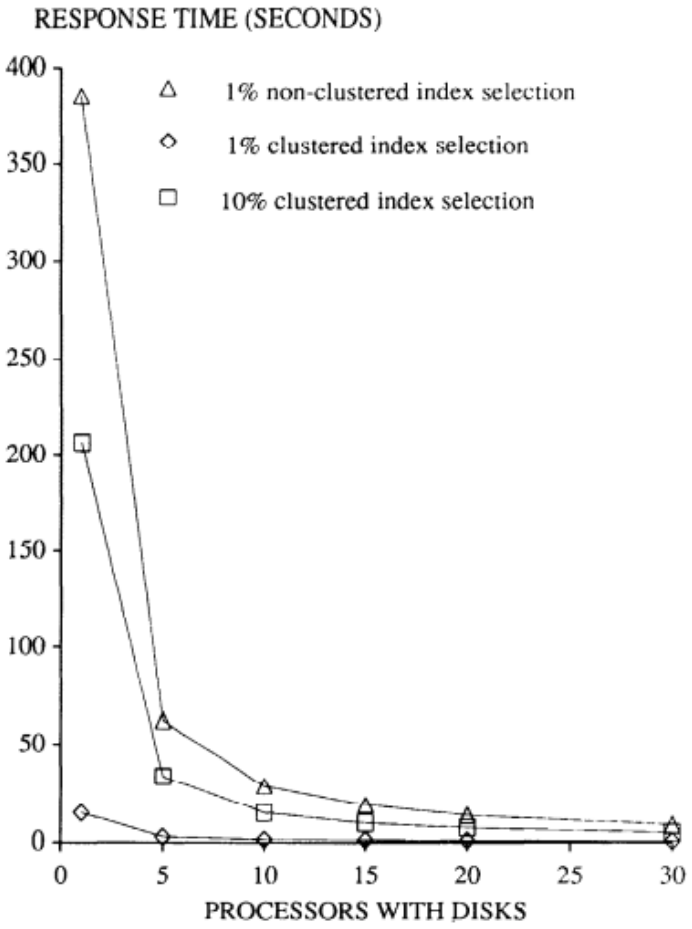
Performance Studies

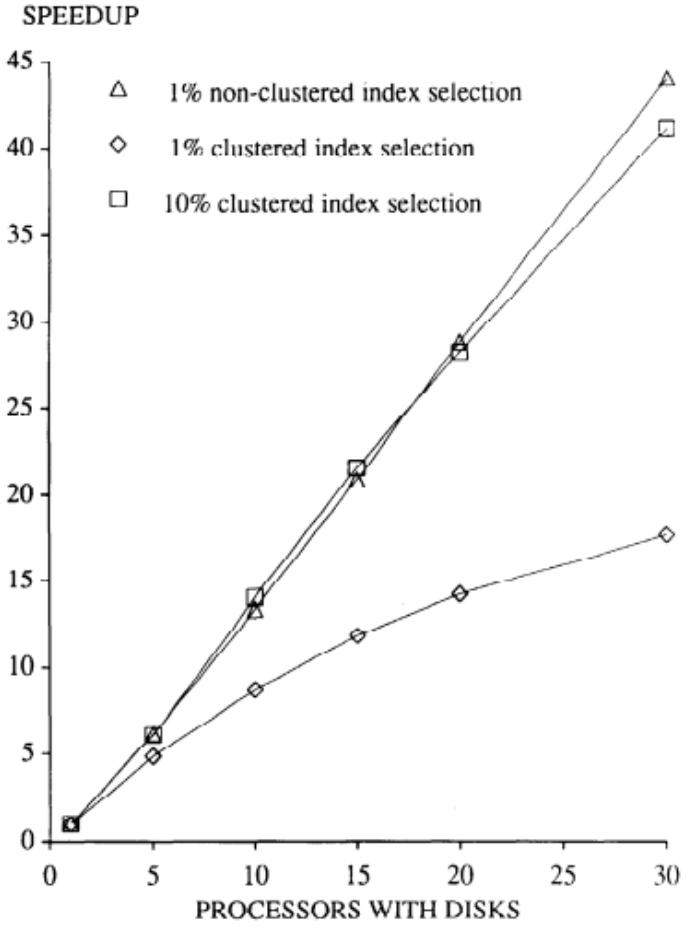
Speedup (Selection)



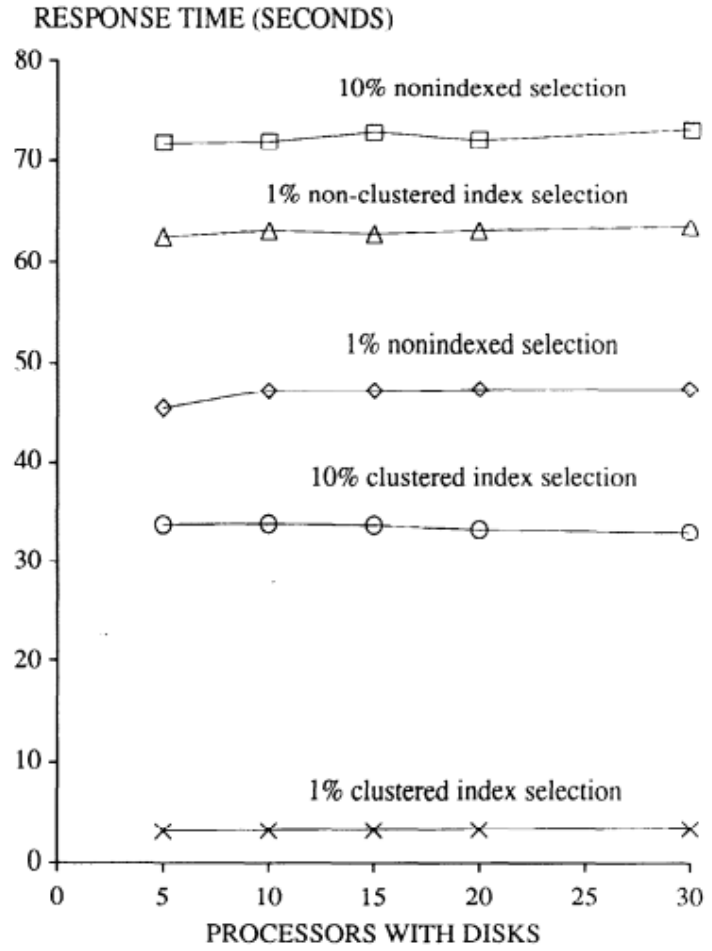


Speedup (Selection)

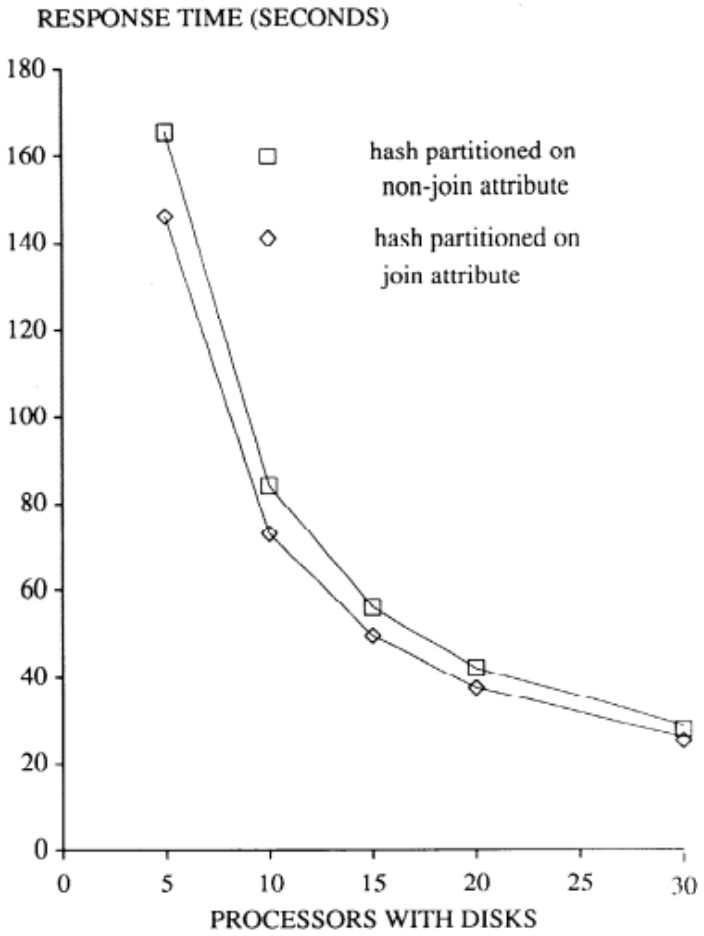


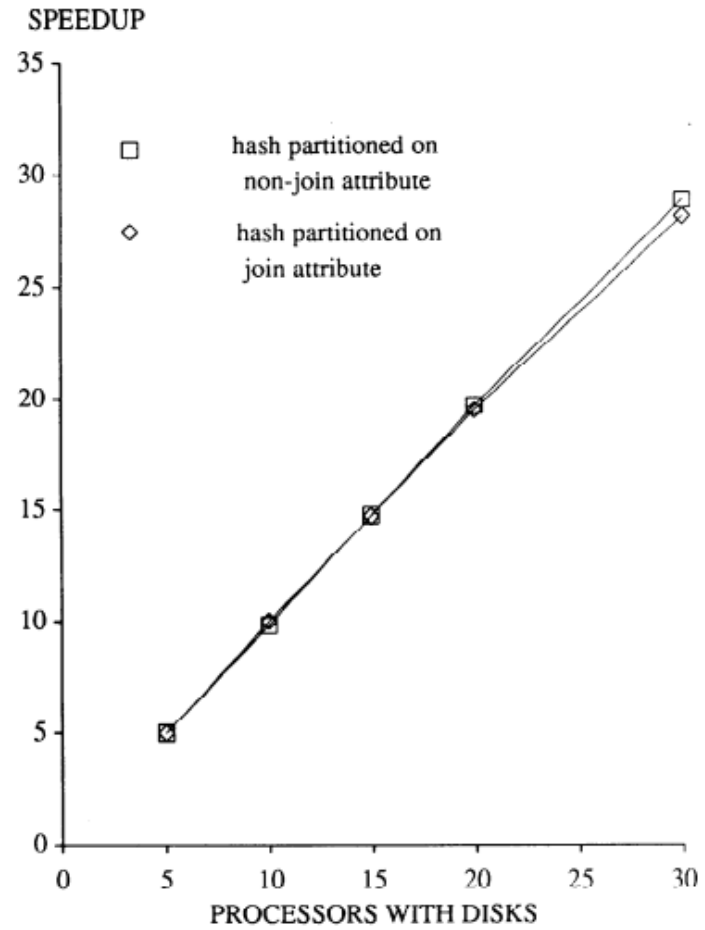


Scaleup (Selection)

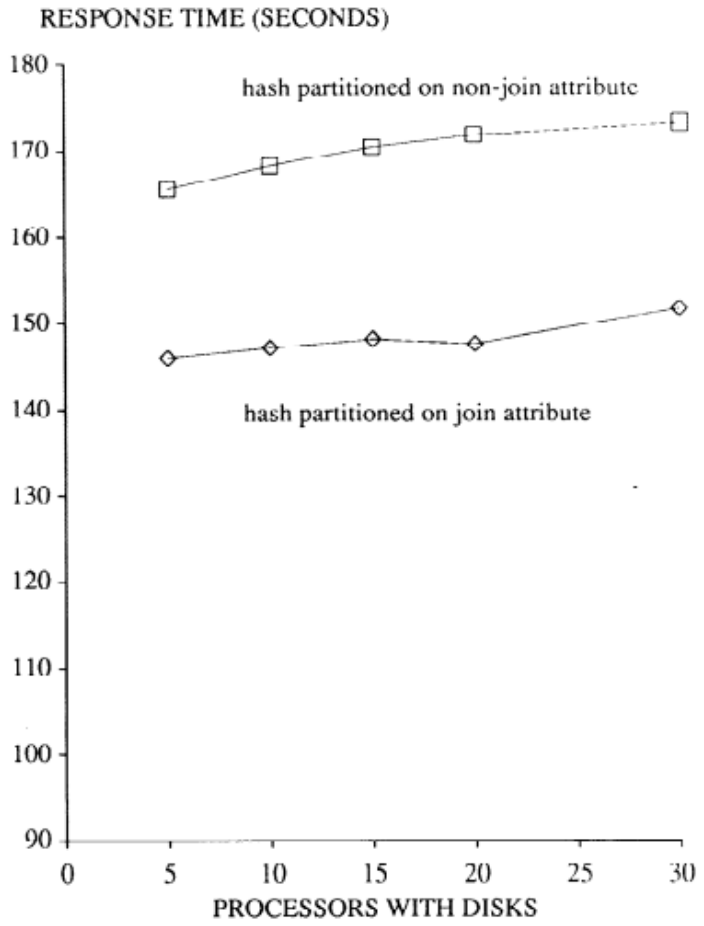


Speedup (Join)





Scaleup (Join)



Conclusion

- Shared-nothing architecture
  - Declustering (horizontal partitioning)
  - Hash based parallel algorithms, ex. Join
  - Dataflow scheduling techniques
- Performance
  - Linear Speedup
  - Constant Scaleup
- Efficient recovery and failure management.

References:

- The Gamma Database Machine, DeWitt, Ghandeharizadeh, Schneider, Bricker, Hsiao, Rasmussen
- web.eecs.umich.edu/~michjc/eecs584

Questions?

Thank You!