Parallel Database Systems

STAVROS HARIZOPOULOS
stavros@cs.cmu.edu

Outline

- Background
- Hardware architectures and performance metrics
- Parallel database techniques
- Gamma
- Bonus: NCR / Teradata
- Conclusions

Parallelism in DBMSs

- Demand for higher throughput
- Opportunity in relational data model
- Hardware trends and paradigms
- Enabling technologies

Information explosion

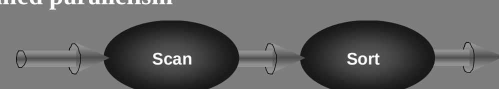
- Magnetic storage is cheaper than paper
- All information goes online
- We might record everything we
  - read: 10 MB/day
  - hear: 400 MB/day
  - see: 40 GB/day
- Data storage, organization, and analysis is a challenge

Relational DBMSs

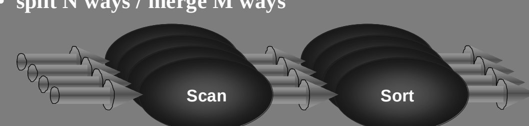
- Relational data model was universally adopted
- Relational queries are ideal for parallel execution
  - uniform operators apply to uniform data streams
  - consume 1-2 relations and produce a new relation
- *Dataflow approach* requires
  - messaging based systems
  - high speed interconnect

Relational DBMS parallelism

- Pipelined parallelism



- Partitioned parallelism
  - split N ways / merge M ways



Trends and paradigms

Mainframe increasingly expensive

Economy of scale

- off-the-shelf components

Trends in networks, storage, memory, and CPUs

- Bottlenecks shift, new issues arise

Enabling technologies

- Client-server / networking software

An idea whose time has passed?

Database machine: research in 1975 - 1985

Exotic technologies lead to failures..

- bubble memory
- head per track disks
- extra logic added on disk heads
- (note the comeback with *Active Disks*)

I/O traditionally assumed as bottleneck

Parallel DBMSs found their way

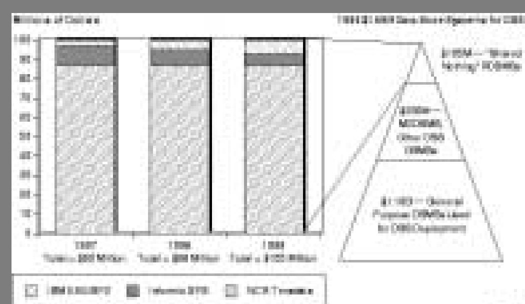
Academia

- Wisconsin: from DIRECT to GAMMA
- Berkeley: XPRS

Commercial

- Teradata started in 1984
- others: Tandem, Oracle, Informix, Navigator, DB2, Red-brick

The market today



Outline

Background

Hardware architectures and performance metrics

Parallel database techniques

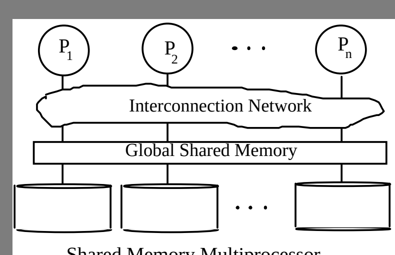
Gamma

Bonus: NCR / Teradata

Conclusions

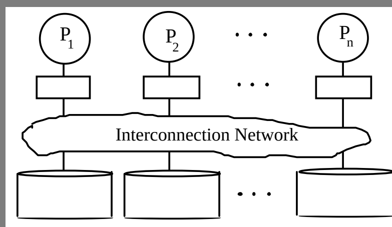
Shared-Memory

CPUs share direct access to global RAM and disks



Shared-Disk

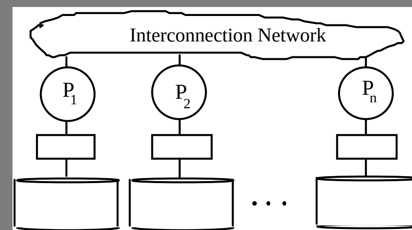
CPUs still share disks but have private RAM



Shared Disk Multiprocessor

Shared-Nothing

CPUs own RAM and disk(s)



Pros and cons

Shared systems don't scale

- resource contention
- interconnect bandwidth must equal #nodes
- concurrency problems

Fixes?

- private cache
- affinity scheduling

Pros and cons (cont'd)

Shared nothing scales better

- minimal resource sharing / contention
- low traffic on interconnect
- commodity components

Which is easier to

- program?
- build?
- scaleup?

Performance metrics

Speedup

- add nodes to run faster a fixed problem

Scaleup

- add nodes to run at the same time a bigger problem

Transaction Scaleup

- more clients / servers, same response time

Barriers to linear throughput

Startup

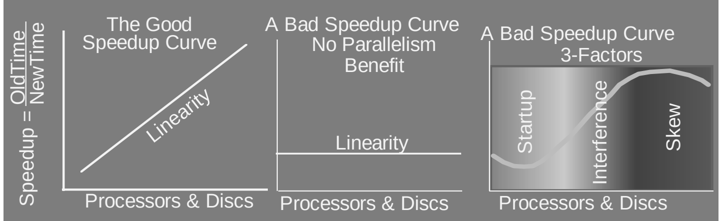
Interference

- even 1% increased contention limits speedup to 37

Skew

- at fine granularity variance can exceed mean service

Barriers to linear throughput



Outline

- Background
- Hardware architectures and performance metrics
- Parallel database techniques
- Gamma
- Bonus: NCR / Teradata
- Conclusions

Dataflow approach to SQL

- Relational properties
  - uniform data stream
  - relations are created, updated, queried via SQL
  - i.e. scan = select + project
- SQL benefits
  - data independence
  - non-procedural
  - can be executed as dataflow graph

Achieving parallelism

- Data partitioning of relations
- Pipelining relational operators
- Partitioned execution of relational operators

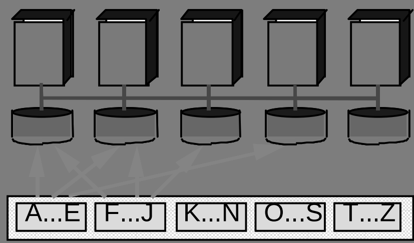
Limits to pipelining

- Pipelines are inherently short
- Some operators are not pipelineable
- Skew limits speedup

Data partitioning

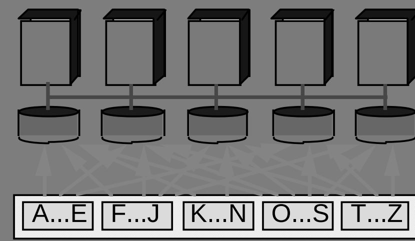
- I/O happens in parallel
- Three basic strategies
  - round robin
  - hash
  - range
- Partitioning helps both seq. and assoc. scans
- Further partitioning helps up to a point

Round-robin partitioning



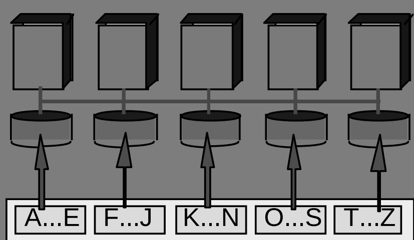
Good for sequential scans / spread load

Hash partitioning



Good for equijoins

Range partitioning



Good for range queries

Parallelizing Relational Operators

Reuse existing implementations

Shared-nothing helps

Only three mechanisms needed

- operator replication
- merge operator
- split operator

Result is linear speedup and scaleup

Parallelizing Relational Operators

Operator replication

- linear scaleup minus starting cost
- specialized operators (e.g. hash join - see GAMMA)

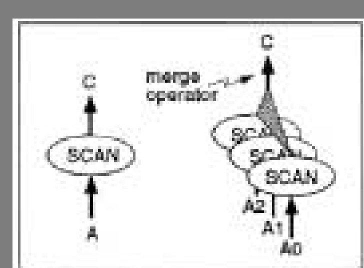
Merge operator

- combine many streams into one

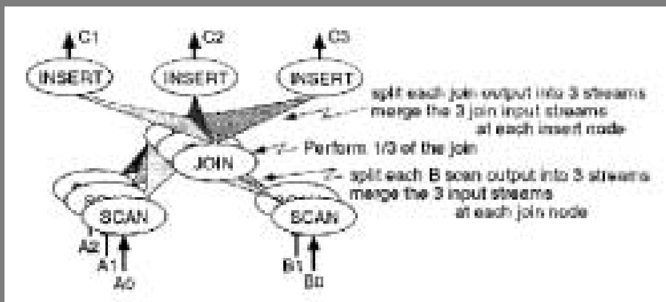
Split operator

- map from attr. values to destination processes

Operators on partitioned data (1)



Operators on partitioned data (2)



Summary

Exotic technologies yield to inexpensive hardware

Shared-nothing serves better parallel DBMSs

Potential of parallel database systems

Many implementations (successful: Teradata)

Research issues (at the end of the presentation)

Outline

Background

Hardware architectures and performance metrics

Parallel database techniques

Gamma

Bonus: NCR / Teradata

Conclusions

History

DIRECT 1977 - 84
- early database machine project
- showed parallelism useful for db apps

Flaws curtailed scalability
- shared memory
- central control of execution

Gamma 1984 - 92

Key ideas in Gamma

Shared-nothing

Hash-based parallel algorithms

Horizontal partitioning

Gamma hardware (v1.0)

17 VAX 11/750 processors

2 MB RAM per node

80 Mb/s token ring

Separate VAX running Unix (host)

333 MB Fujitsu drives at 8 processors

Gamma hardware (v1.0) issues

- 2K DB pages due to token ring
- Unibus congestion (network and I/O faster)
  - corrected with a backplane card
- VAX obsolete
- 2MB with no virtual memory was tight

Gamma hardware (v2.0) 1988

- iPSC/2 Intel hypercube
- 32 x386 processors
- 8MB of memory
- 330MB Maxtor drive / node (45KB cache)
- Routing modules
  - 2.8 Mb/s
  - full duplex, serial, reliable

Gamma software (v2.0)

- OS: NOSE
  - multiple, lightweight processes with shared memory
- Entire DB in one NX/2 process
- Details: renaming nodes
- 10% CPU used for copying
- Excessive interrupts during I/O

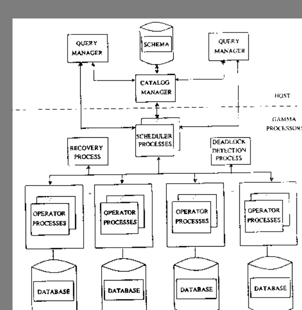
Storage organization

- Horizontal partitioning (user selectable)
  - round-robin
  - hashed
  - range partitioned
- Clustered index on different attributes
- Partition relations should have based on ‘heat’

Gamma process structure

- Catalog manager
  - repository for db schema
- Query manager
  - one associated with each user
- Scheduler processes
  - coordinates multi-site queries
- Operator processes
  - executes single relational operator

Gamma process structure (figure)

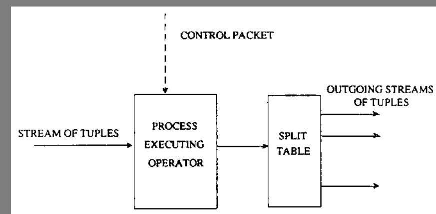


Query processing

- ad-hoc and embedded query interfaces
- standard parsing, optimization, code gen.
- left deep trees only
- hash joins only
- at most two join operators active simultaneously
- split tables

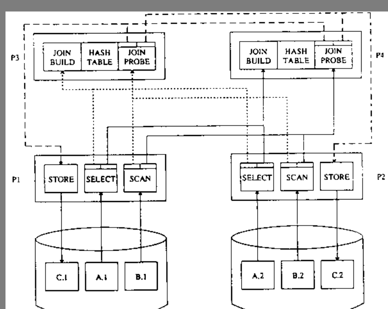
Split table

Directs operator output to appropriate node

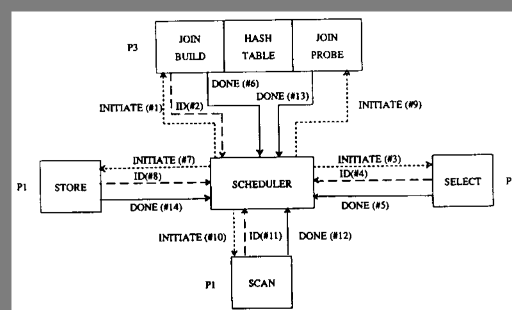


Value	Destination Process
0	(Processor #3, Port #5)
1	(Processor #2, Port #13)
2	(Processor #7, Port #6)
3	(Processor #9, Port #15)

Query processing: an example



Query processing: an example



Selections

Start selection operator on each node

Exclusion of nodes for hash and range partitioning

Throughput considerations

- one page read-ahead

Joins

Partition into buckets, join buckets

Implemented:

- sort-merge, Grace, Simple, Hybrid

Parallel Hybrid

Aggregation

- Compute partial results for each partition
- Hash on “group-by”attribute

Updates

- Standard techniques
- Update to partitioning attribute

Concurrency control

- 2PL
- Granularity: file and page
- Modes: S, X, IS, IX, SIX
- Local lock manager and deadlock detector
  - wait-for graph
- Centralized multi-site lock detector

Logging and recovery

- Log sequence number
  - LSN = f (node number, local sequence number)
- Processor i directs log records at log manager
  - (i mod M), where M = # log mgrs.
- Standard WAL protocol
  - local log manager reduces time waiting for log manager
- ARIES

Node failure

- Availability in spite of processor or disk fail
- Mirrored disk (Tandem)
- Interleaved declustering (Teradata)
- Chained declustering
- Load redirection results in 1/n increase

Node failure - declustering

	Cluster 0				Cluster 1			
Node	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	

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 failure - load redirection

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

Fig. 11. Fragment utilization with chained declustering after the failure of node 1 (relation cluster size = 8).

Performance experiments

Selection

- relation size
- speedup
- scaleup

Join

- (similar)

Aggregate

Update (no recovery)

Outline

Background

Hardware architectures and performance metrics

Parallel database techniques

Gamma

Bonus: NCR / Teradata

Conclusions

NCR / Teradata

Some research problems

Parallel query optimization

Concurrency

Physical database design

Scheduling (load balancing, priority)

Application program parallelism

On-line data reorganization