Encapsulation of Parallelism in the Volcano Query Processing System

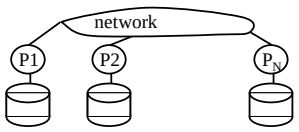
Goetz Graefe (1990)

The Volcano Query Processing System

- GAMMA
- Introduction
- Related Work
- The Volcano System
- Exchange Operator
- Conclusion

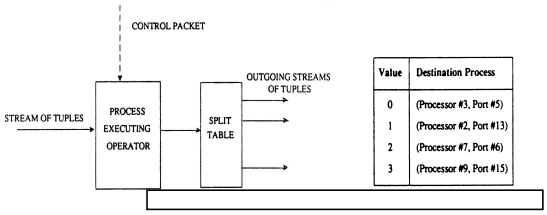
GAMMA

- Shared-nothing
- Hash-based parallel algorithms
- Horizontal partitioning (‘declustering’)

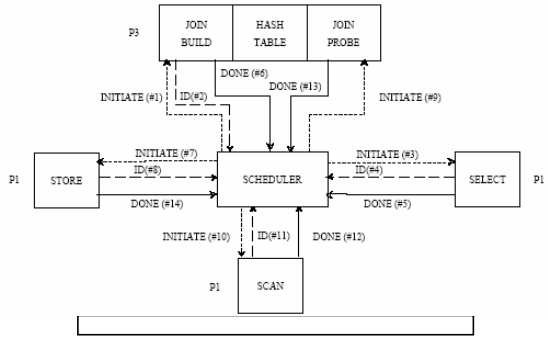


Split Table

Directs operator output to the appropriate node (e.g., by some hash value)



E.g.: Parallel Hash Join



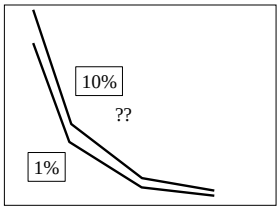
Experimental setup

- Wisconsin benchmark (100K, 1M, 10M tuples);
- tables: hash partitioned
- selections (1%, 10%) x (non-indexed, clustered index)
- joins
- wallclock time; speedup; scale-up

Selections

- non-indexed, 1%, 10%

response time



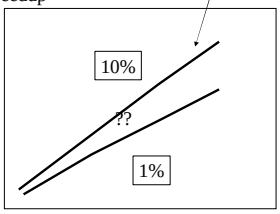
#processors

Selections

- non-indexed, 1%, 10%

speedup

superlinear! why?

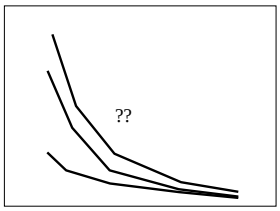


#processors

Selections

- clustered ind., 1%,
- clustered 10%
- non-clustered 1%

response time

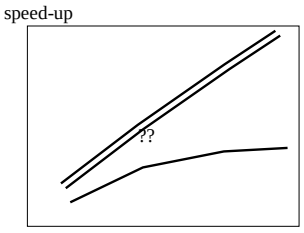


#processors

Selections

- clustered 1%,
- clustered 10%
- non-clustered 1%

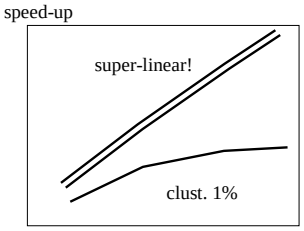
- super-linear?
- sub-linear?
- why?



Selections

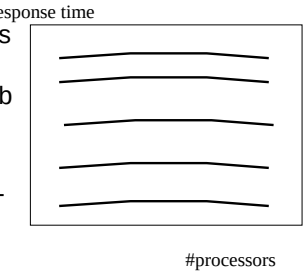
- clustered 1%,
- clustered 10%
- non-clustered 1%

- why super-linear?
- why sub-linear?



Selections - scaleup

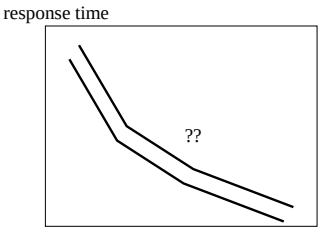
- response time vs processors, increasing the db size
- All queries: ~constant scale-up



Joins

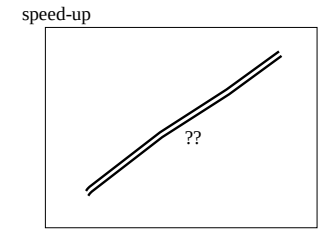
A join B

- part. = join attr
- part. attr != join attr



Joins

- joinAB
  - part. = join attr
  - part. attr != join attr



The Volcano Query Processing System

- GAMMA
- ➡ Introduction
- Related Work
- The Volcano System
- Exchange Operator
- Conclusion

Introduction

- Design and implementation of an extensible query processing system
- Should allow parallelizing of algorithms without reasoning about parallelism

Approach to Parallelism

- The exchange operator is used to parallelize query execution plans
- Volcano's mechanisms for operators and data exchange similar to commercial systems (System-R, Ingres)

Main Ideas behind Volcano

- Uniform interface extensible to new operators (iterator interface)
- Operator Model approach to parallelization
- Exchange operator used for parallelization

Questions Volcano Attempts to Address:

- How do we design an **extensible** system that is also efficient?
- How do we parallelize operators but **free** the programmer from reasoning about such parallelism?

Related Work

- Influenced by GAMMA (but departs radically in data exchange and parallelization)
  - Make a more extensible system
- Tandem Computer's parallel operator similar to the exchange operator

The Volcano Query Processing System

- GAMMA
- Introduction
- Related Work
- ➡ The Volcano System
- Exchange Operator
- Conclusion

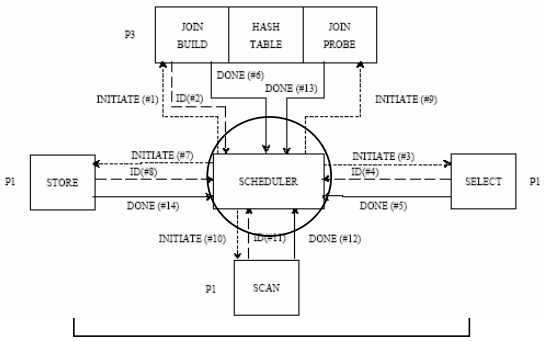
The Volcano System

- Study the design of an extensible system that is **also** very efficient
- Parallelization of query evaluation through the use of an operator
- Volcano system supports file systems, buffer management, sorting, B^+-trees, joins, (and many others)

The Volcano System (continued)

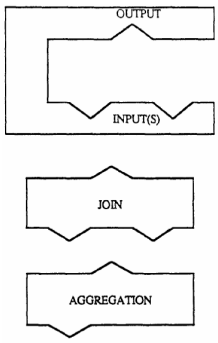
- Query Engine – provides operator building blocks with uniform (iterator) interface
  - Each block looks the same, operates on a constant stream of inputs
  - Algorithms are generic, state records capture specificity
- Query Optimizer - builds the query execution plan from operators, including the use of the Exchange parallelizing operator

GAMMA System



Bracket Model Advantages

- Generic process template for sending/receiving data
- Each operator wrapped within template, shielded from environment
- Template provides I/O service for data exchange



Bracket Model Pitfalls

- Template only executes one operator at a time
- Needs external scheduler to schedule operator
- Data exchange requires expensive network I/O or inter-procedure calls

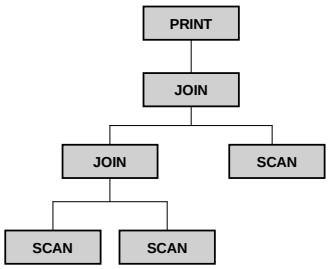
Bracket Model seen as unsuitable for an extensible system

Operator Model

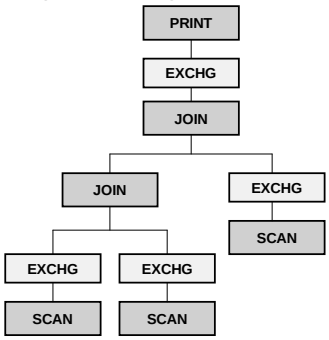
- Query Execution Engine provides parallelism mechanisms
- Query Optimizer decides on policy
- Single operator (Exchange) provides parallelism
  - Iterator interface
  - Data exchange through shared memory “port”
  - Inserted into points of the query plan

Exchange operator enables parallelism

Simple Query Execution Plan



Inserting Exchange into a QEP



The Volcano Query Processing System

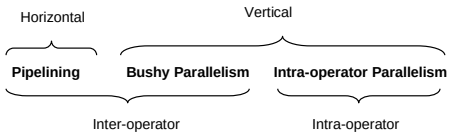
- GAMMA
- Introduction
- Related Work
- The Volcano System
- ➡ Exchange Operator
- Conclusion

The Exchange Operator

- Exchange operator creates shared port, forks child process
- Child process produces data to the port
  - Aggregate tuples into packets, write packet
  - Allows N queued packets to accumulate
- Parent process consumes data

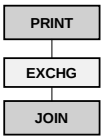
The Exchange Operator (continued)

- Captures both vertical and horizontal parallelism (or inter and intra operator parallelism)
- What are 3 of the ways we can exploit parallelism?

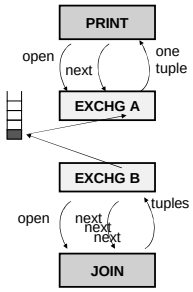


Vertical Parallelism

- The Exchange operator provides pipelining between processes
  - Calling *open* (EXCHNG) creates new process, and shared port
  - Exchange operator in parent process (PRINT) receives data from IPC
  - Exchange operator in child process (JOIN) produces packets of tuples to port



In detail…



1. ***Open*** **procedure in EXCHNG A called**
2. **New process (EXCHNG B) forked, port created**
3. **EXCHNG B calls *open* for JOIN, and so on**
4. ***Next*** **procedure in EXCHNG A called, Waits for data from port**
5. **EXCHNG B calls *Next* in JOIN, keeps producing tuples, put into port as packet**
6. **EXCHNG A reads packet, returns tuple per next call to PRINT**

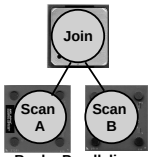
Dataflow

- Demand Driven
  - Iterators, lazy evaluation
  - Only produce a tuple when *next* is called
- Data Driven
  - Exchange operators, eager evaluation
  - Eagerly call *next*, produce tuples, and write to port

Exchange operator decouples the flow of data

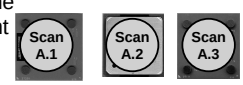
Horizontal Parallelism

- Bushy parallelism
  - Different CPU's execute different subtrees of a complex query tree



Bushy Parallelism

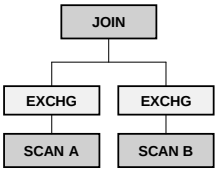
- Intra-operator parallelism
  - Several CPU's perform the same operator on different subsets of data



Intra-operator Parallelism

Bushy Parallelism in Exchange

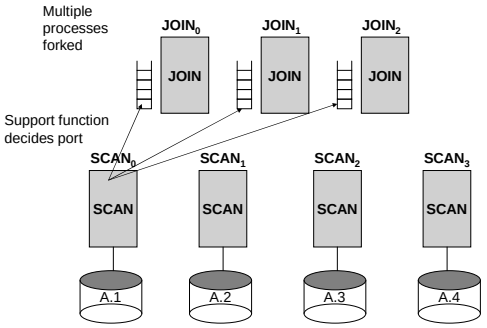
- Bushy parallelism implemented by inserting one or more Exchange operators into a query tree
- SCAN A and B now operate in parallel to produce tuples to the JOIN



Inter-operator Parallelism in Exchange

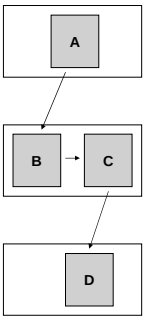
- Requires data partitioning
- Can have multiple ports
- Support function used to decide which port a packet is sent
  - Can implement round robin, range, hash

In detail…



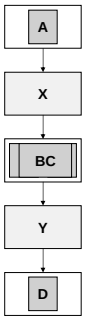
Example of Intra-operator parallelism

- Consider a query with four operators: A, B, C, D
  - A calls B's iterator methods
  - B calls C's iterator methods
  - C calls D's iterator methods
- Assume there are three processing groups: A, BC, D

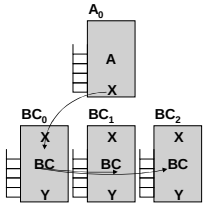


Example (continued)

- Exchange operators need to be inserted between A, BC, and D
- B and C run in the same process, and pass records through simple procedure calls
- A has process A_0
- BC has processes BC_0 BC_1 BC_2
- D has processes D_0 D_1 D_2 D_3



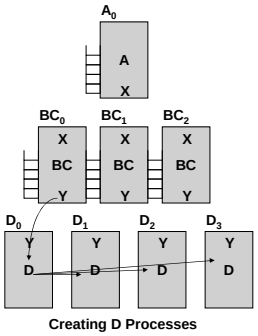
Example (continued)



- A calls X's open, close, next procedures instead of B's (without knowledge of process boundaries)
- X creates a port with one input queue for A_0 and forks BC_0
- BC_0 forks rest. BC group wait for Y to initialize 3 input queues

Creating BC Processes

Example (continued)



Creating D Processes

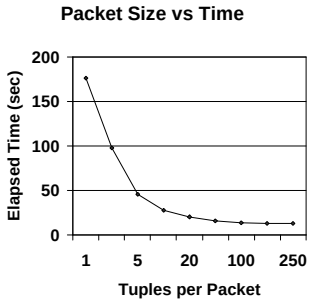
Overhead and Performance

- Overhead of iterator procedure calls small (but not insignificant)
- Pipelining in Exchange improve performance

	Time for 100k tuples
Read	20.28 seconds
Read using iterator	28.00 seconds
Reading using iterator and pipeline	16.21 seconds

Packet Size

- Size of packets can be changed to make overhead negligible



Summary

- Volcano Query Processing System is a flexible and extensible
- The Exchange operator allows vertical, bushy, and intra-operator parallel without exposing parallelism to other operators
- Novel Exchange operator decouples data flow, enables vertical, bushy, and intra-operator parallelism
- Operator model and exchange operator allows operators to schedule each other

Comparisons

GAMMA	Volcano
Shared-nothing	Shared-memory
Bracket Model	Operator Model
Central system schedules operators	Operators schedule operators
Split/Merge Tables	Exchange Operator

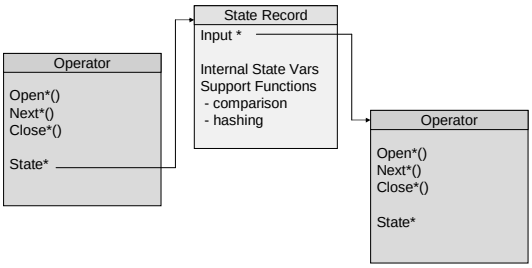
… anything else?

Conclusion

- Operator model supports self-scheduling parallel query evaluation in an extensible database system
- Can exploit many types of parallelism “for free” programmers do not need to reason about parallel algorithms

Questions?

Operator Structure



Anonymous Inputs or Streams
Operator does not need to know what operator produces its input