Parallel Query Execution

Parallelism

Why parallelism

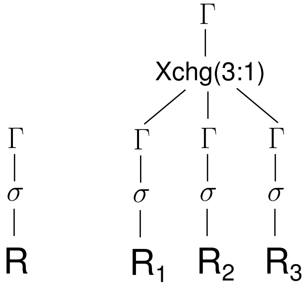
- allow multiple users at the same time
- better utilize hardware resources (CPU and IO)

Forms of parallelism

- inter-query parallelism: execute multiple queries concurrently
  - map each query to one process/thread
  - concurrency control mechanism isolates the queries
  - except for that parallelism is "for free"
- intra-query parallelism: parallelize a single query
  - horizontal (bushy) parallelism: execute independent sub plans in parallel (not very useful)
  - vertical parallelism: parallelize operators themselves

Vertical Parallelism: Exchange Operator

- implements iterator interface
- optimizer statically determines at query compile-time how many threads should run
- instantiates one query operator plan for each thread
- connects these with exchange operators, which encapsulate parallelism, start threads, and buffer data
- relational operator can remain (largely) unchanged



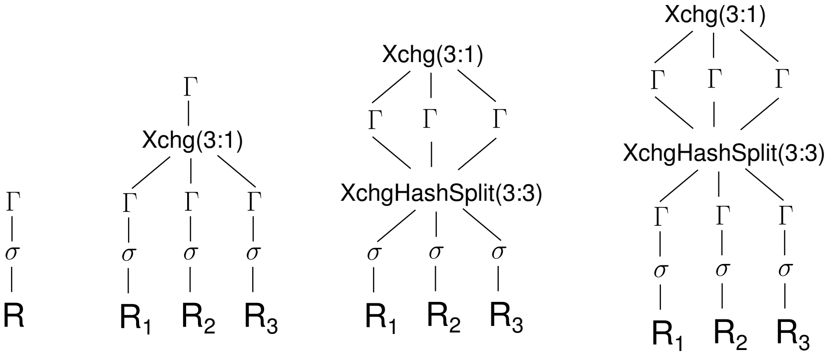
Exchange Operator Variants

- Xchg(N:M) N input pipelines, M output pipelines

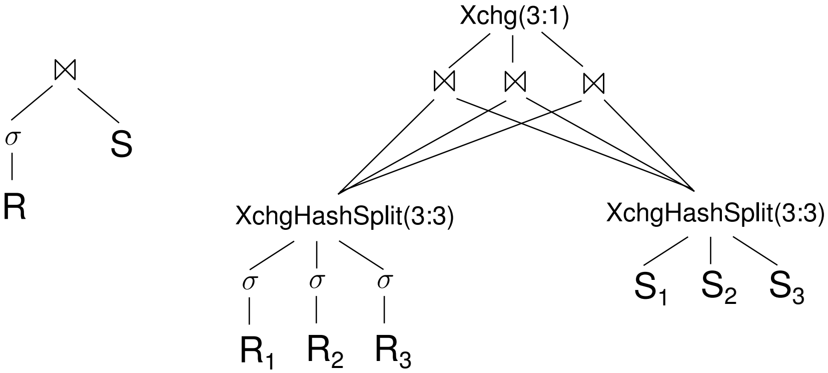
Many useful variants

- XchgUnion(N:1) specialization of Xchg
- XchgDynamicSplit(1:M) specialization of Xchg
- XchgHashSplit(N:M) split by hash values
- XchgBroadcast(N:M) send full input to all consumers
- XchgRangeSplit(N:M) partition by data ranges

Parallel Aggregation with Exchange Operators



Parallel Join with Exchange Operators



Disadvantages of Exchange Operators

- static work partitioning can cause load imbalances
- degree of parallelism cannot easily be changed mid-query (e.g., when a new query arrives)
- overhead:
  - thread oversubscription causes context switching
  - hash re-partitioning often does not pay off
  - exchange operators create additional copies of the tuples