Other Relational Operations in Parallel Database System

Selection $\sigma_\theta(r)$

- If θ is of the form $a_i = v$, where a_i is an attribute and v a value.
    - If r is partitioned on a_i the selection is performed at a single node.

Given schema person (NID, name, street, city, district, DOB, income)

Nodes are N1, N2 ……. N64

Partitioning attribute is district and each node contains tuples of a district.

Q1: Select * from person where district = ‘Dhaka’

Q2: Select * from person where district = ‘Sherpur’

Question 12-1:

a. Why each of the queries Q1 and Q2 will be processed in single node?
b. Compare the processing time of Q1 and Q2 and suggest possible improvement.

Other Relational Operations in Parallel Database System

Selection $\sigma_\theta(r)$

- If θ is of the form l <= a_i <= u (i.e., θ is a range selection) and the relation has been range-partitioned on a_i
  - Selection is performed at each node whose partition overlaps with the specified range of values.

Given schema person (NID, name, street, city, district, DOB, income)

Nodes are N1, N2 ……. N64

Partitioning attribute is NID and there is no skew.

Q21: Select * from person where NID > 4567891230 and NID < 8678912340

N10 - 4500000000 <NID < 5000000000

N30 - 8678912340 <NID < 8978912340

Question 12-2: Explain how Q21 will be processed.

Other Relational Operations in Parallel Database System

Selection σ_θ(r)

- In all other cases: the selection is performed in parallel at all the nodes.

Given schema person (NID, name, street, city, district, DOB, income)

Nodes are N1, N2 ……. N64

Partitioning attribute is district and each node contains tuples of a district.

Q1: Select * from person

Q2: Select * from person where DOB > ’01-jan-2000’

Question 12-3:

Why each of the queries Q1 and Q2 will be processed in all nodes in parallel?

Other Relational Operations (Cont.)

- **Duplicate elimination**
  - Perform by using either of the parallel sort techniques
    - eliminate duplicates as soon as they are found during sorting.
  - Can also partition the tuples (using either range- or hash- partitioning) and perform duplicate elimination locally at each node.

Select distinct (name, street, city) from person

Select distinct (income) from person

Other Relational Operations (Cont.)

- **Projection**
  - Projection without duplicate elimination can be performed as tuples are read from disk, in parallel.
  - If duplicate elimination is required, any of the above duplicate elimination techniques can be used.

Grouping/Aggregation

- **Step 1**: Partition the relation on the grouping attributes
- **Step 2**: Compute the aggregate values locally at each node.

Given schema person (NID, name, street, city, district, DOB, income)

Person relation has 160 million tuples. Nodes are N1, N2 ……. N64

Partitioning attribute is district. Repartition on income requires to transfer 80% tuples from each node.

Q31: Select income, count(NID) from person group by income

Question 12-4:

a. Explain how this aggregate query will be processed?
b. Find the repartition cost
c. Optimize the aggregate query and find the optimized repartition cost.

- **Optimization:** Can reduce cost of transferring tuples during partitioning by **partial aggregation** before partitioning
  - For distributive aggregate
  - Can be done as part of run generation
  - Consider the **count** aggregation operation:
    - Perform aggregation operation at each node N_i on those tuples stored its local disk
      - results in tuples with partial counts at each node.
    - Result of the local aggregation is partitioned on the grouping attributes, and the aggregation performed again at each node N_i to get the final result.

Given schema person (NID, name, street, city, district, DOB, income)
Person relation has 160 million tuples. Nodes are N1, N2 ……. N64
Partitioning attribute is district. Repartition on age requires to transfer 80% tuples from each node. There are 640 distinct income in person relation
Q31: Select income, count(NID) from person group by income

Question 12-4:
a.Explain how this aggregate query will be processed?
b.Find the repartition cost
c.Optimize the aggregate query and find the optimized repartition cost

Parallel Evaluation of Query Plans

- Intraoperation Parallelism
  - Parallel join
  - Parallel sort

- Interoperation Parallelism
  - Pipeline parallilsm
  - Independent parallelism

- Exchange operator model

Exchange operator model

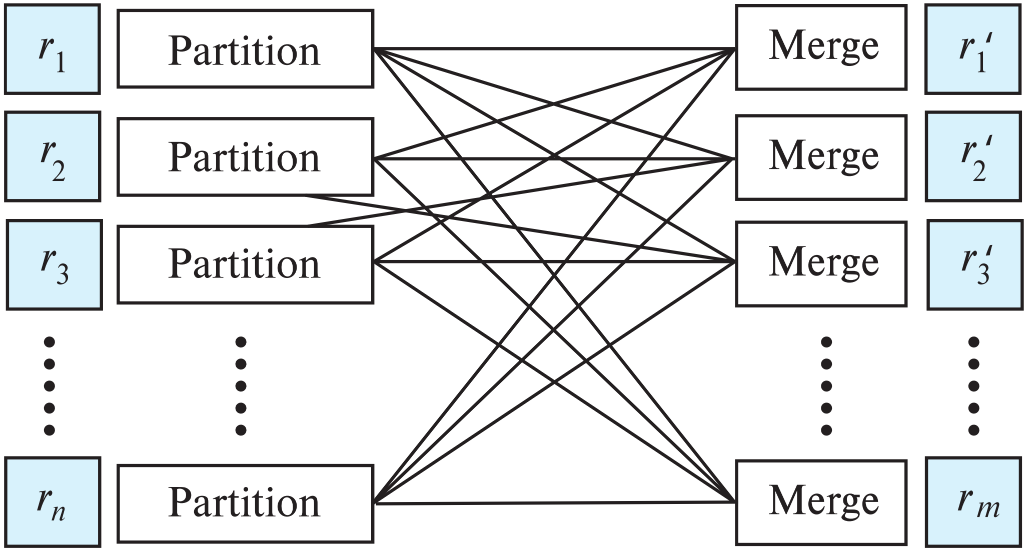
- The Volcano parallel database popularized a model of parallelization called the exchange-operator model.

- The exchange operation repartitions data in a specified way; data interchange between nodes is done only by the exchange operator.

- All other operations work on local data, just as they would in a centralized database system; the data may be available locally either because it is already present, or because of the execution of a preceding exchange operator.

Exchange Operator

- Repartitioning implemented using the **exchange operator**
  - **Partition** and **merge** steps



Exchange Operator Model

- Movement of data encapsulated in **exchange operator**
- Partitioning of data can be done by
  - Hash partitioning
  - Range partitioning
  - Replicating data to all nodes (called **broadcasting**)
  - Sending all data to a single node
- Destination nodes can receive data from multiple source nodes. Incoming data can be merged by:
  - **Random merge**
  - **Ordered merge**
- Other operators in a plan can be unaware of parallelism
  - **Data parallelism**: each operator works purely on local data
  - Not always best way, but works well in most cases

Parallel Plans Using Exchange Operator

- Range partitioning sort:
  1. Exchange operator using range partitioning, followed by
  2. Local sort
- Parallel external sort merge
  1. Local sort followed by
  2. Exchange operator with ordered merge

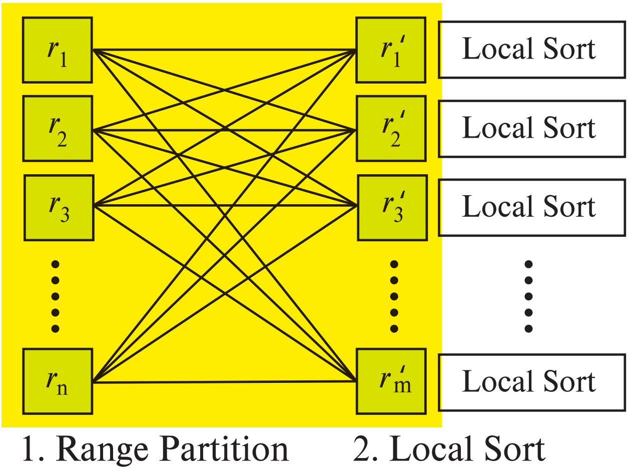
Given schema person (NID, name, street, city, district, DOB, income). Person relation has 160 million tuples. Nodes are N1, N2 ……. N64, Partitioning attribute is district.

Question 13-1:

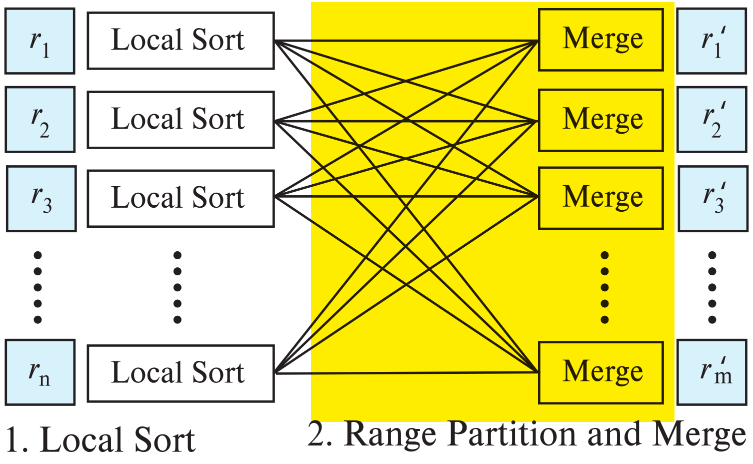
a.Explain how the SQL: select * from person order by income will be executed using exchange operator (parameters: relation name, partition type, partition attribute, number of nodes) with

a. Range partitioning sort:

b. Parallel external sort merge



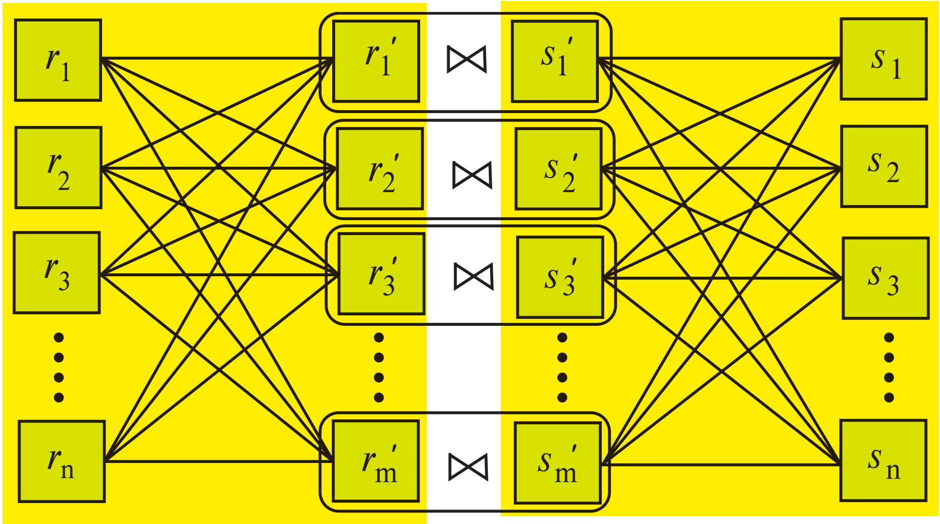
(a) Range Partitioning Sort



(b) Parallel External Sort-Merge

Parallel Plans Using Exchange Operator

- Partitioned join
  1. Exchange operator with hash or range partitioning, followed by
  2. Local join



Step 1: Partition *r*   Step 2: Partition *s*

Step 3: Each node N_i computes $r_i' \bowtie s_i'$

Given schema
person (NID, name, street, city, district, DOB, income)
Employee (e-Id, NID, Organization, position, district)
Person relation has 160 million tuples and employee relation has 10 million tutples. Nodes are N1, N2 ……. N64, Partitioning attribute is district.

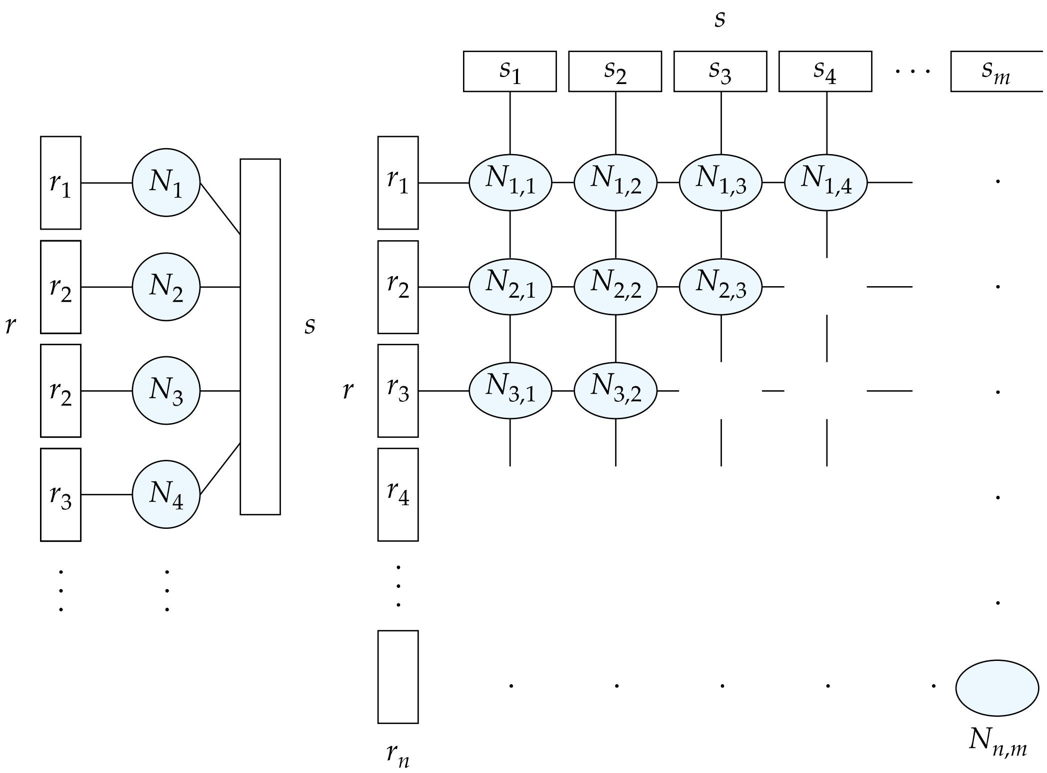
Question 14-1:
Explain how the

SQL: select * from person p, employee e where p.NID = e.NID

will be executed using exchange operator with partitioned join

Parallel Plans Using Exchange Operator

- Asymmetric fragment and replicate
  1. Exchange operator using broadcast replication, followed by
  2. Local join
- Exchange operator can also implement push model, with batching



Given schema
person (NID, name, street, city, district, DOB, income)
Employee (e-Id, NID, Organization, position, district, salary)
Person relation has 160 million tuples and employee relation has 10 million tutples. Nodes are N1, N2 ……. N64, Partitioning attribute is district.

Question 14-2:

Explain how the

SQL: select * from person p, employee e where p.income > e.salary
will be executed using exchange operator with AFR join

Fragment-and-Replicate Join

- Partitioning not possible for some join conditions
  - E.g., non-equijoin conditions, such as r.A > s.B.

- For joins were partitioning is not applicable, parallelization can be accomplished by **fragment and replicate technique**

- Special case – **asymmetric fragment-and-replicate**:
  - One of the relations, say r, is partitioned; any partitioning technique can be used.
  - The other relation, s, is replicated across all the processors.
  - Node Ni then locally computes the join of ri with all of s using any join technique.
  - Also referred to as **broadcast join**

Question 14-3: “Partitioning not possible for some join conditions” Explain with an example

Exchange Operator Model

- Movement of data encapsulated in **exchange operator**
- Partitioning of data can be done by
  - Hash partitioning

    Xchg-HP (N, A:M), N: set of input pipelines, M: set of output pipelines, A is the set of attributes

    Xchg-HP (r, A:r′, M), r: input relation, r′ is output relation, M: set of output pipelines, A is the set of attributes

  - Range partitioning

    Xchg-RP (N, A:M)

    Xchg-RP (r, A:r′, M), r: input relation, r′ is output relation, M: set of output pipelines

Exchange Operator Model

- Movement of data encapsulated in **exchange operator**
- Partitioning of data can be done by
  - Replicating data to all nodes (called **broadcasting**)

  Xchg-Broadcast(N:M), N: set of input pipelines, M: set of output pipelines

  Xchg-Broadcast(r:M), r: input relation, M: set of output pipelines

  - Sending all data to a single node

  Xchg-Union(N:M_i), M_i is the output node

  Xchg-Union(r:M_i)

Exchange Operator Model

- Movement of data encapsulated in **exchange operator**

- Destination nodes can receive data from multiple source nodes. Incoming data can be merged by:
  - **Random merge**

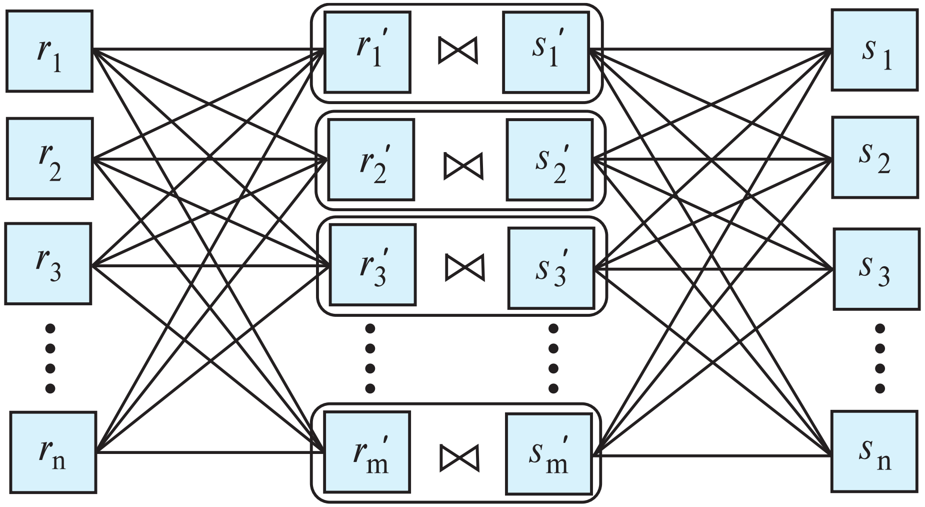
    Xcg-RM (N:M)

  - **Ordered merge**

    Xch-OM (N:M)

Parallel Plans Using Exchange Operator

- Partitioned join
  1. Exchange operator with hash or range partitioning, followed by
  2. Local join



Step 1: Partition *r* Step 2: Partition *s*

Step 3: Each node N_i computes $r_i' \bowtie s_i'$

Question 14-1:
Explain how the

SQL: select * from person p, employee e where p.NID = e.NID

will be executed using exchange operator with partitioned join

1.
Xchg-HP (p, NID:p′, 64)
Xchg-HP (e, NID:e′, 64)

2.

p′ ⋈ e′

Query Optimization for Parallel Query Execution

Query Optimization For Parallel Execution

- Query optimization in parallel databases is significantly more complex than query optimization in sequential databases.

  - Different options for partitioning inputs and intermediate results

  - Cost models are more complicated, since we must take into account partitioning costs and issues such as skew and resource contention.

Parallel Query Plan Space

A parallel query plan must specify

- How to parallelize each operation, including which algorithm to use, and how to partition inputs and intermediate results
- How the plan is to be **scheduled**
  - How many nodes to use for each operation
  - What operations to pipeline within same node or different nodes
  - What operations to execute independently in parallel, and
  - What operations to execute sequentially, one after the other.
- E.g., In query ${}_{r.A}\gamma_{\text{sum}(s.C)}(r \bowtie_{r.A=s.A \;\; r.B=s.B} s)$
  - Partitioning *r* and *s* on (*A*,*B*) for join will require repartitioning for aggregation
  - But partitioning *r* and *s* on (*A*) for join will allow aggregation with no further repartitioning
- Query optimizer has to choose best plan taking above issues into account

Cost of Parallel Query Execution

- **Resource consumption cost model**
  - used for centralized databases
- **Response time cost model**
  - attempts to better estimate the time to completion of a query
  - E.g., If an operation performs I/O operations in parallel with CPU execution, the response time
    $T = max$(CPU cost, I/O cost)
    - Resource consumption cost model uses (CPU cost + I/O cost).
  - E.g., if two operations o_1 and o_2 are in a pipeline, with CPU and I/O costs $c_1; io_1$ and $c_2; io_2$ respectively, then response time
    $T = max(c_1 + c_2, io_1 + io_2)$.
  - Operators in parallel: $T = max\ (T_1, T_1, \ldots, T_n)$
    - Skew is an issue

Question 15-1: Explain the impact of skewing using the parallel cost model: $T = max\ (T_1, T_1, \ldots, T_n)$ for range partition join of r ⋈ s.

Cost of Parallel Query Execution (Cont.)

- Response time cost model would have to take into account
  - **Start-up costs** for initiating an operation on multiple nodes
  - **Skew** in distribution of work
- Response time cost model better suited for parallel databases
  - But not used much since it increases cost of query optimization

Choosing Query Plans

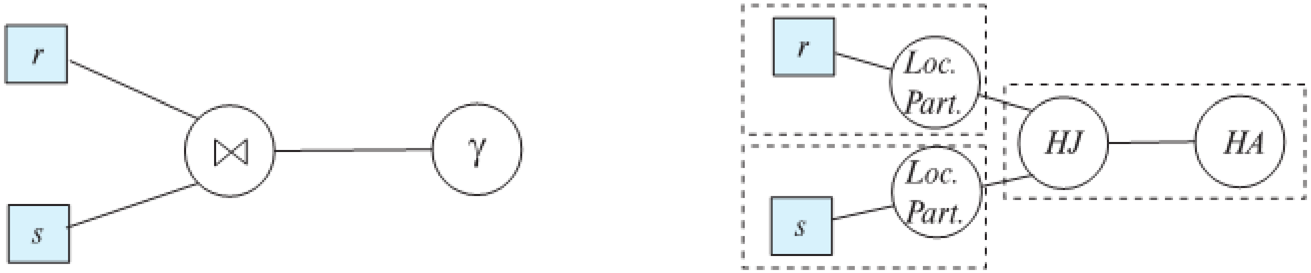
- The number of parallel evaluation plans from which to choose from is much larger than the number of sequential evaluation plans
  - Many alternative partitioning options
  - Choosing a good physical organization (partitioning technique) is important to speed up queries.
- Two alternatives often used for choosing parallel plans:
  - First choose most efficient sequential plan and then choose how best to parallelize the operations in that plan
    - Heuristic, since best sequential plan may not lead to best parallel plan
  - Parallelize every operation across all nodes
    - Use exchange operator to perform (re)partitioning
    - Use standard query optimizer with extended cost model

Parallel Plans

Query

Given r(A, C), s(G, B, D, E)

$_{r.C,s.D}\gamma_{\text{sum}(s.E)}(r \bowtie_{r.A=s.B} s)$.



(a) Logical Query

(b) Sequential Plan

Dashed boxes denote pipelined segment

The sequential plan uses a hash join (denoted as “HJ” in the figure), which executes in three separate stages.

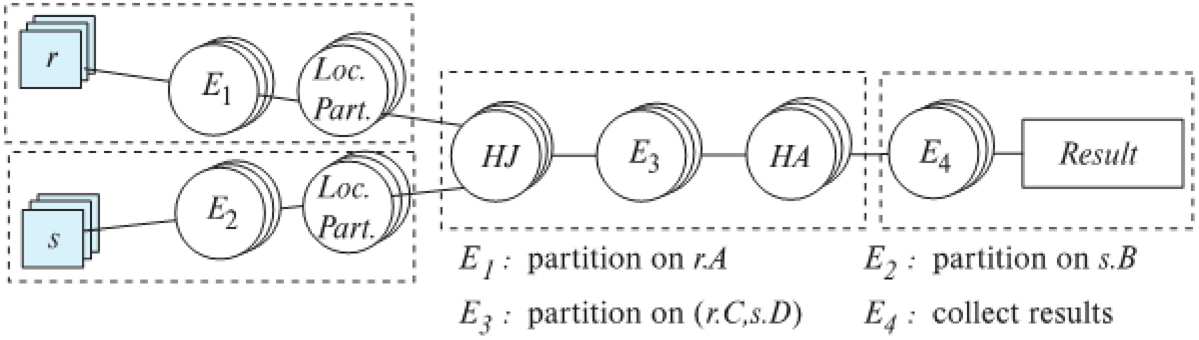
- The first stage partitions the first input (r) locally on r.A;
- the second stage partitions the second input (s) locally on s.B;
- and the third stage computes the join of each of the corresponding partitions of r and s.

The aggregate is computed using in-memory hash-aggregation, denoted by the operator HA

Parallel Plans

Query
Given r(A, C), s(G, B, D, E)

$r.C,s.D\ \gamma\ \text{sum}(s.E)(r \bowtie_{r.A=s.B} s)$.



(c) Parallel Plan

- The parallel query evaluation plan starts with r and s already partitioned, but not on the required join attributes.
- The plan, therefore, uses the exchange operation E1 to repartition r using attribute r.A; similarly, exchange operator E2 repartitions s using s.B.
- Each node then uses hash join locally to compute the join of its partition of r and s.
- Note that exchange of tuples across nodes is done only by the exchange operator, and all other edges denote tuple flows within each node.