Parallel Databases/Query Planning

Problem 1

Given the following relations D(A, B) and E(A, C)

Suppose that D and E are partitioned across 3 different machines using random block partitioning, and no indexes are available on any of the machines. If we use a hash-join (aka. shuffle-join) in the relational algebra plan to execute the queries below, determine if the following clauses can be determined before or after the shuffle.

a)
```
SELECT D.A
FROM D, E
WHERE D.A = E.A
AND E.C > 10;
```

Do we need to shuffle before 'E.C > 10' can be determined?
Solution: No, the storage node can do it because no other attributes or values are required during this filter.

b)
```
SELECT D.A
FROM D, E
WHERE D.A = E.A
AND E.C - D.B > 20;
```

Do we need to shuffle before 'E.C - D.B > 20' can be determined?
Solution: Yes, there is no guarantee on the same machine

c)
```
SELECT D.A
FROM D, E
WHERE D.A = E.A
GROUP BY D.A
HAVING MAX(E.C) < 100;
```

Do we need to shuffle before 'GROUP BY D.A' can be determined?
Solution: Yes, we need to shuffle before GROUP BYS because we need all attribute values to create groups

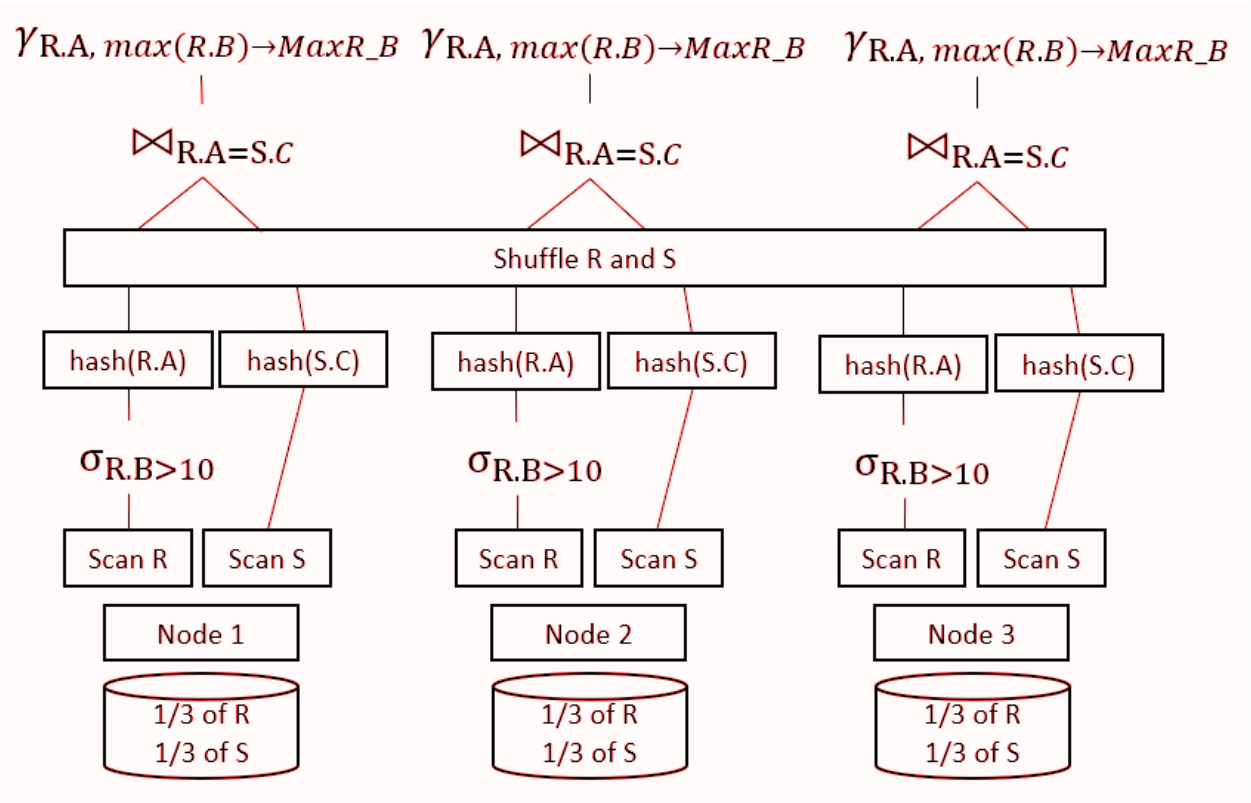
Do we need to shuffle before 'HAVING MAX(E.C) < 100' can be determined?
Solution: Yes, we need to shuffle before any HAVING statement because we need to create groups first

Problem 2

Given the following query with relations `R(A,B)` and `S(C)`:

```
SELECT R.A,  MAX(R.B) AS MaxR_B
FROM R, S
WHERE R.A = S.C AND R.B > 10
GROUP BY R.A
```

Suppose that `R` and `S` are partitioned across 3 different machines using random block partitioning, and no indexes are available on any of the machines. Draw the relational algebra plan that you would use to execute the query above. Use only shuffle for joins. You do not need to indicate how joins are executed locally on each machine.



(Start the diagram with 3 nodes at the bottom.)

Steps:

1. Scan R and S locally
2. Select on R.b locally
3. Hash on R.a and S.c and shuffle
4. Local join on R.a = S.c
5. Group By R.a, and make the aggregate
6. (Optional) Project and return final results

Problem 3: Parallel Databases (Adapted from 414 18AU Final)

Say you are designing a parallel relational database to store purchase data for manufacturing products. The tables are:

```
Manufacturer(mid, name, category, city, state)
Purchase(pid, mid, date, amount) -- mid is a foreign key to Manufacturer
```

Tuples in the purchase table record individual payments in dollar amounts to a manufacturer for purchases of some product. There is a large amount of data in both tables that would have to be spread between multiple machines. As the database designer, you know that the most common query that will be run on the system is:

```
SELECT m.mid, SUM(p.amount) AS total_revenue
FROM   Purchase p, Manufacturer m
WHERE  p.mid = m.mid
GROUP BY m.mid
```

a) Describe in a few sentences how you would partition the data between machines if your goal is to maximize performance of the above query. (5 points)

The most common query does a join on the mid attribute of both tables and a group by on the same attribute.

We'll get the best performance by hash partitioning both tables on mid to avoid the shuffle required for both the join and group-by.

Range partitioning also works but is less common in practice.

b) Now consider that instead of maximizing performance of any query, your goal is to minimize skew and store the data evenly across the machines. Describe in a few sentences how you would partition the data in that case. (5 points)

While we can't minimize skew with respect to the above query, we can spread both tables as evenly as possible among the machines with horizontal partitioning.

Of the schemes learned in class, we could block partition or hash partition on the primary key of each table. It may even be possible to range partition but we'd need to know the statistics of the attributes to choose the correct range values.

Problem 4

Imagine that you are designing a parallel RDMS to digitize all the letters received by the post office. We assume that "letters" may only contain paper (ie, text) or DVDs (ie, binary blobs).

LetterID	SenderAddr	RecipientAddr	Status	ContentType	Content
12345	1600 Pennsylvania Ave	185 E Stevens Way	Delivered	Text	"Dear Hannah, I would like to request a regrade of the midterm …"
67890	3800 E Stevens Way	185 E Stevens Way	InTransit	DVD	*(lots and lots of 0s and 1s)*

Consider the partitioning strategies we know:

- Block (Horizontal)
- Range (Horizontal) - please specify attribute set
- Hash (Horizontal) - please specify attribute set
- Vertical - please specify attribute set

Which one would you choose under the following circumstances? If you choose Range, Hash, or Vertical partitioning, please specify the attributes that you would partition on.

a) The query load consisted solely of counts of in-transit letters (ie, "SELECT COUNT(*) WHERE Status='InTransit'")
Since the majority of the columns in the table are not used, we could vertically partition the data by Status.

b) All of the attributes have uniformly distributed data except for recipients (eg, the President of the United States gets an unusually large or unusually small amount of letters)
Since no mention is made of expected query load, we should choose a horizontal partitioning scheme (because we don't know which columns are needed). Block partitioning would work well; range partitioning or hash partitioning would also work well if we do not choose an attribute set consisting solely of the recipient column.

c) The query load consists primarily but not solely of randomly-sampled letter contents, grouped by sender (eg, “the President of the United States typically sends letters that start with “Dear Sir or Madam, …”). The database contains every letter sent since the creation of the Postal Service in 1771.
As before, we should choose a horizontal partitioning scheme (although we do know that the letter contents will be used eventually). If we are grouping primarily by sender, this would be a reasonable attribute to partition on.

The founding of the postal service is a red herring in this question (but if you don’t know, now you know, Mr President).