

CSE 444: Database Internals

Section 9:

Parallel Processing and MapReduce

Review in this section

- Parallel DBMS
- MapReduce

Parallel DBMS

$R(a,b)$ is horizontally partitioned across $N = 3$ machines.

Each machine locally stores approximately $1/N$ of the tuples in R .

The tuples are randomly organized across machines (i.e., R is block partitioned across machines).

Show a RA plan for this query and how it will be executed across the $N = 3$ machines.

Pick an efficient plan that leverages the parallelism as much as possible.

```
SELECT a, max(b) as topb  
FROM R  
WHERE a > 0  
GROUP BY a
```

R(a, b)

```
SELECT a, max(b) as topb  
FROM R  
WHERE a > 0  
GROUP BY a
```

Machine 1

1/3 of R

Machine 2

1/3 of R

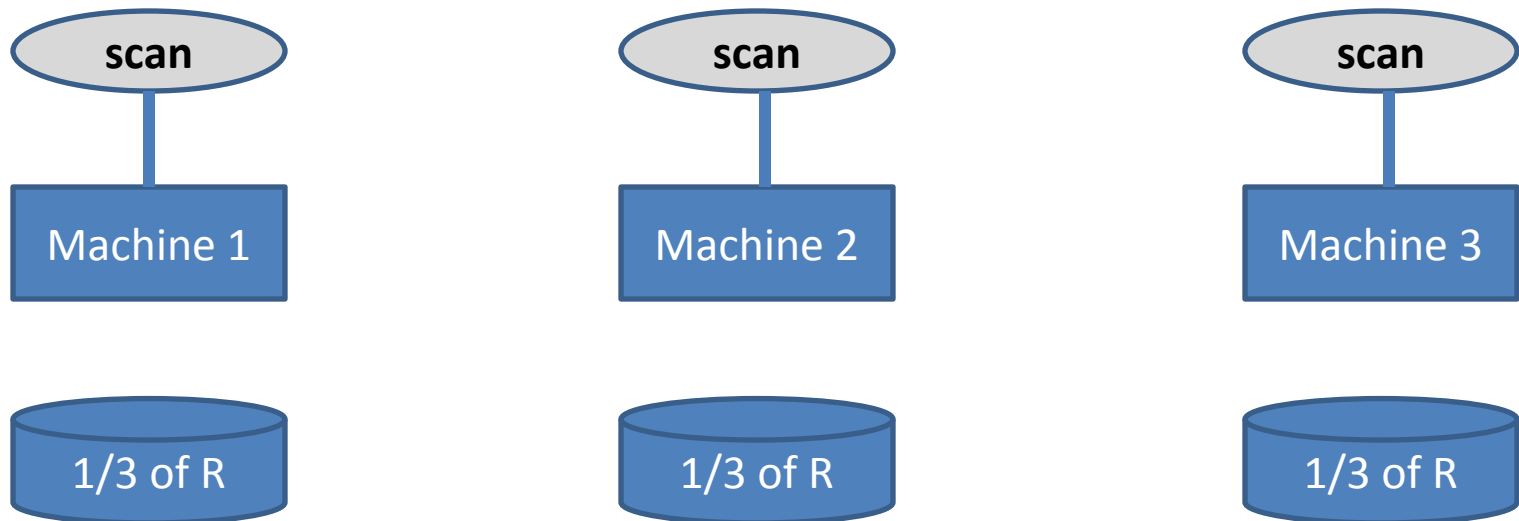
Machine 3

1/3 of R

R(a, b)

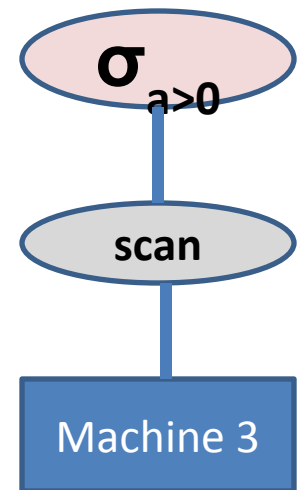
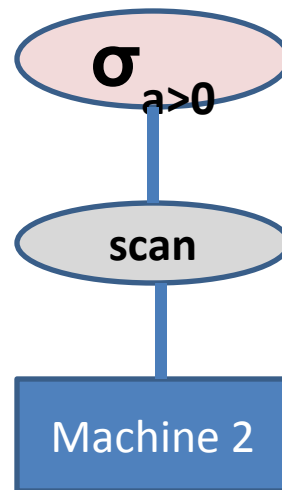
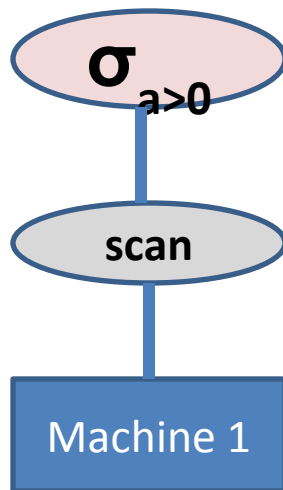
```
SELECT a, max(b) as topb  
FROM R  
WHERE a > 0  
GROUP BY a
```

If more than one relation on a machine, then “scan S”, “scan R” etc



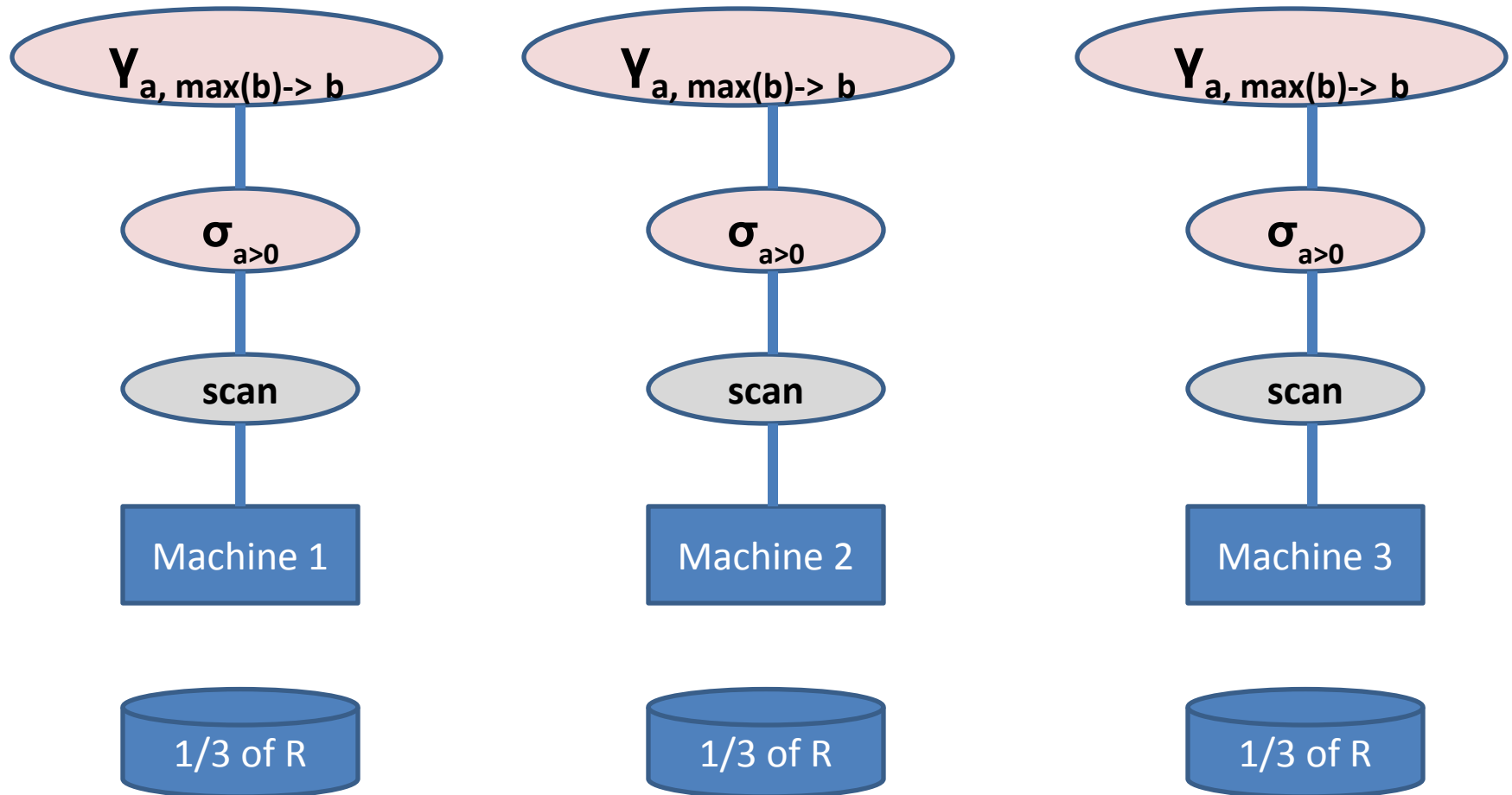
R(a, b)

```
SELECT a, max(b) as topb  
FROM R  
WHERE a > 0  
GROUP BY a
```



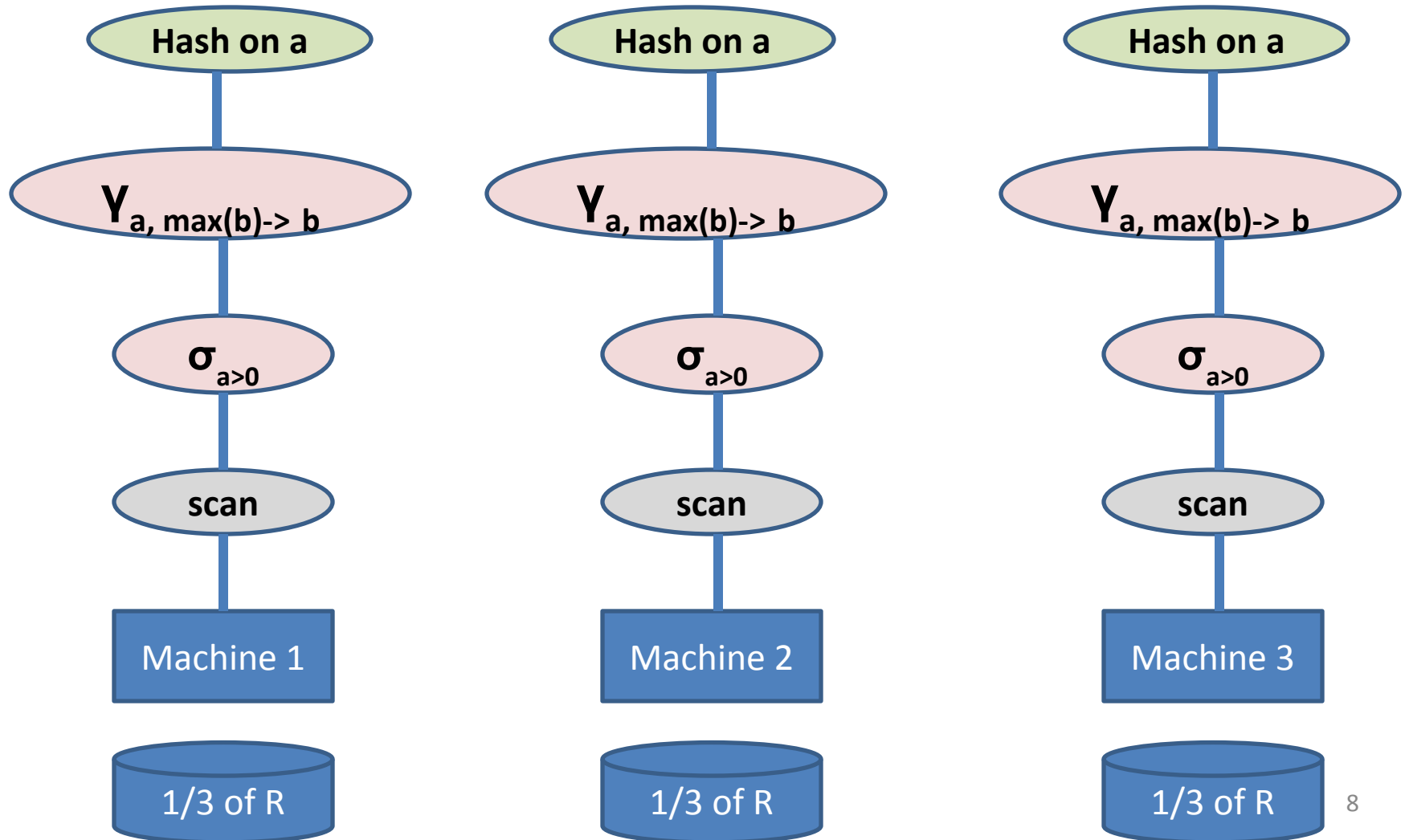
R(a, b)

```
SELECT a, max(b) as topb  
FROM R  
WHERE a > 0  
GROUP BY a
```



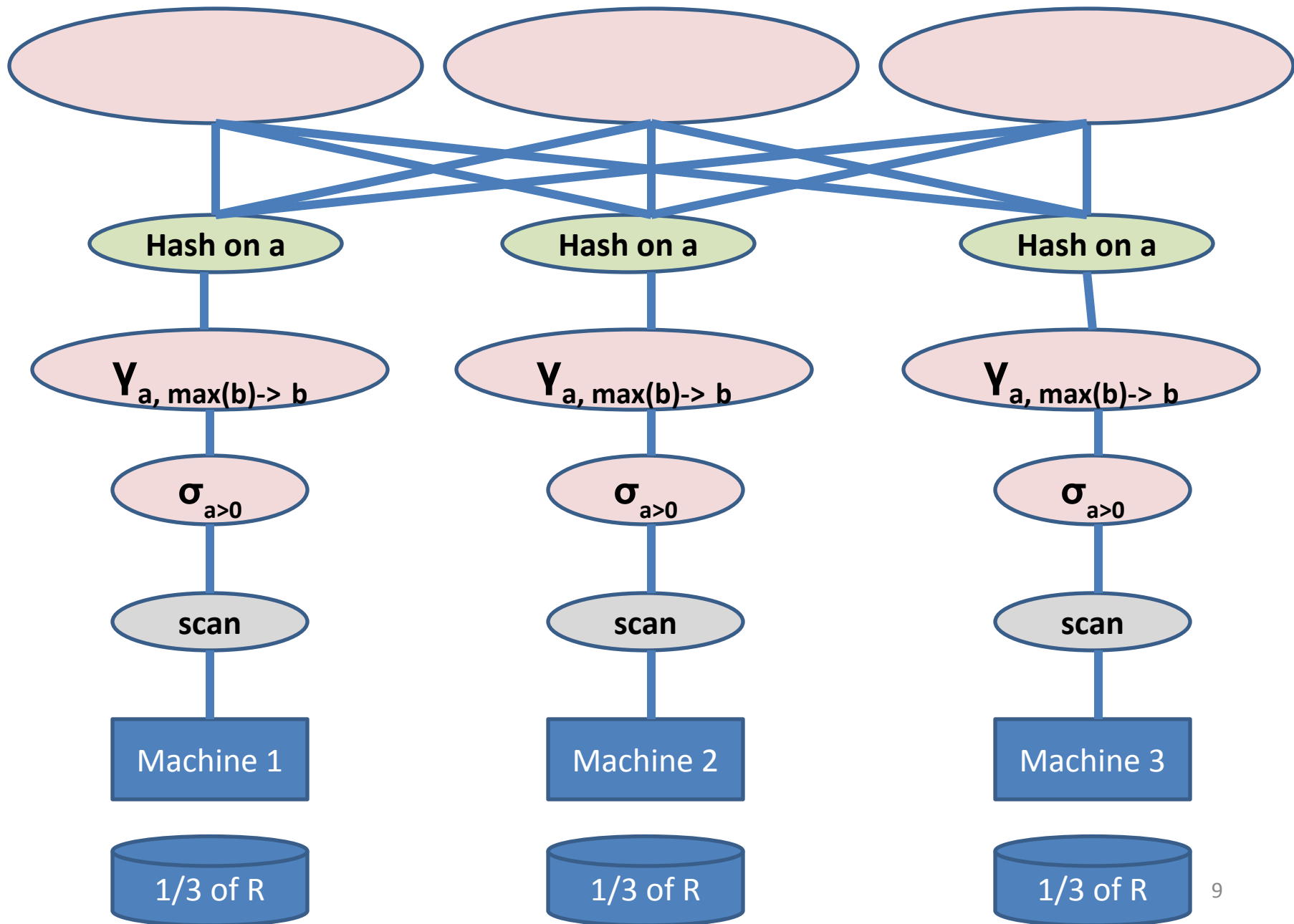
$R(a, b)$

```
SELECT a, max(b) as topb
FROM R
WHERE a > 0
GROUP BY a
```



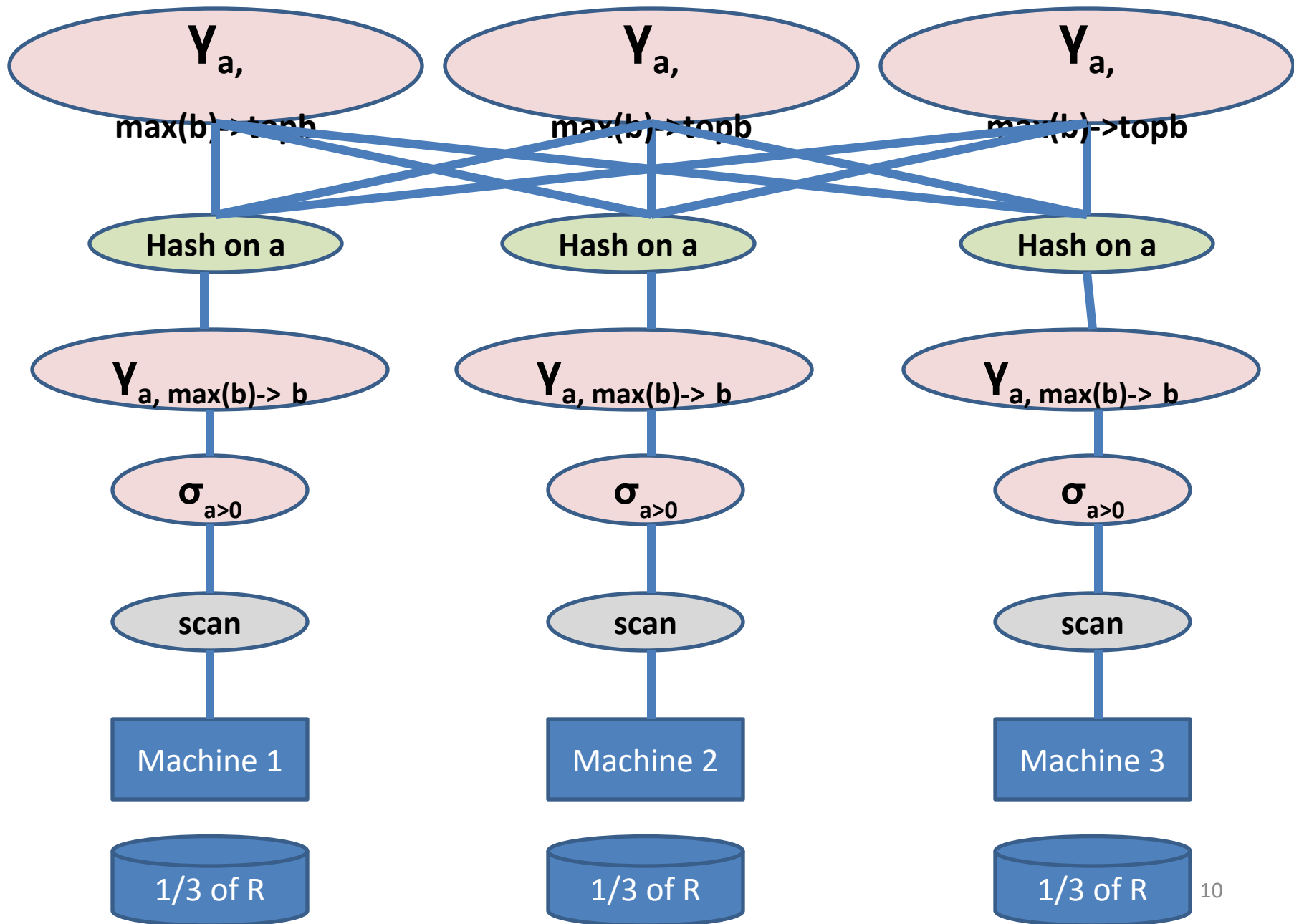
R(a, b)

SELECT a, max(b) as topb FROM R
WHERE a > 0
GROUP BY a



R(a, b)

SELECT a, max(b) as topb FROM R
WHERE a > 0 GROUP BY a



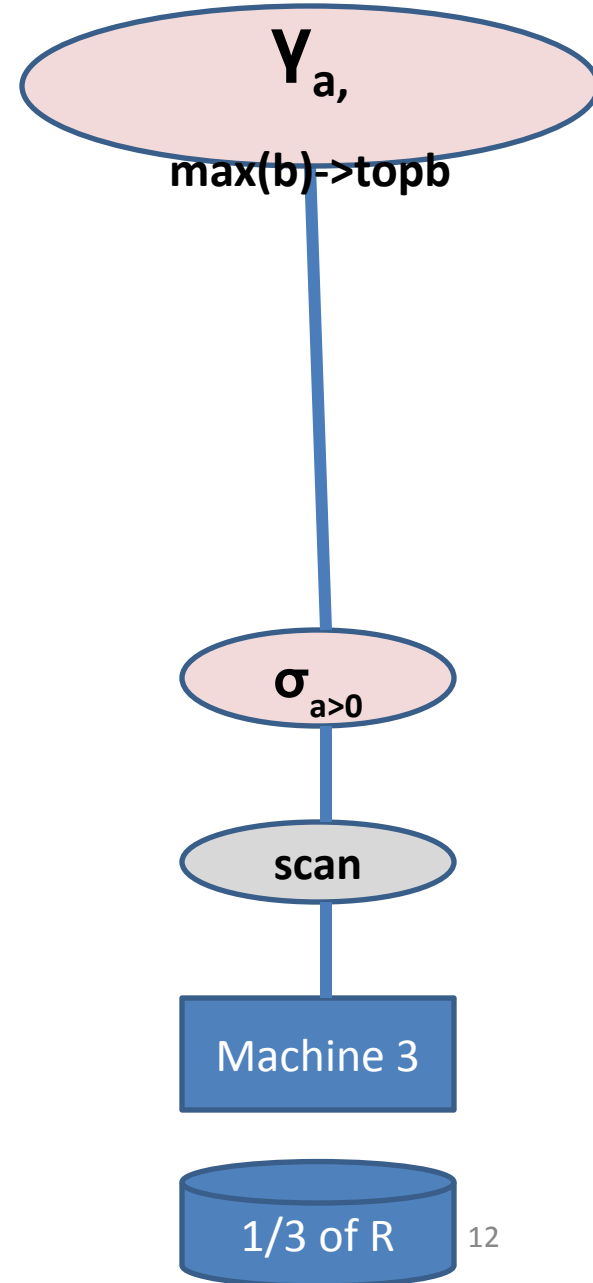
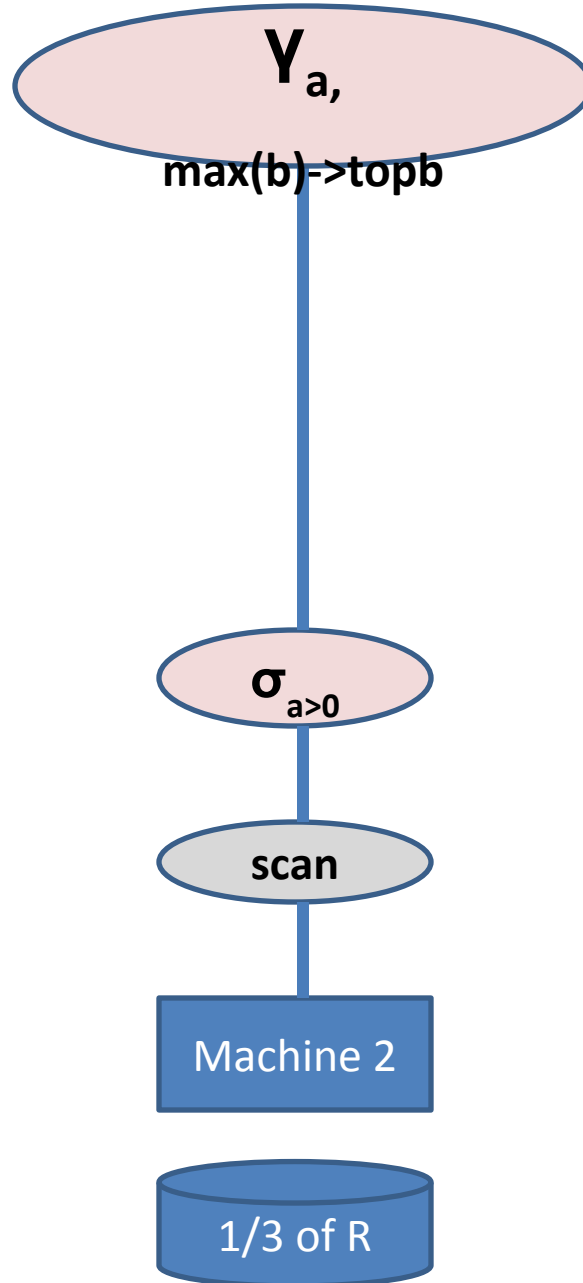
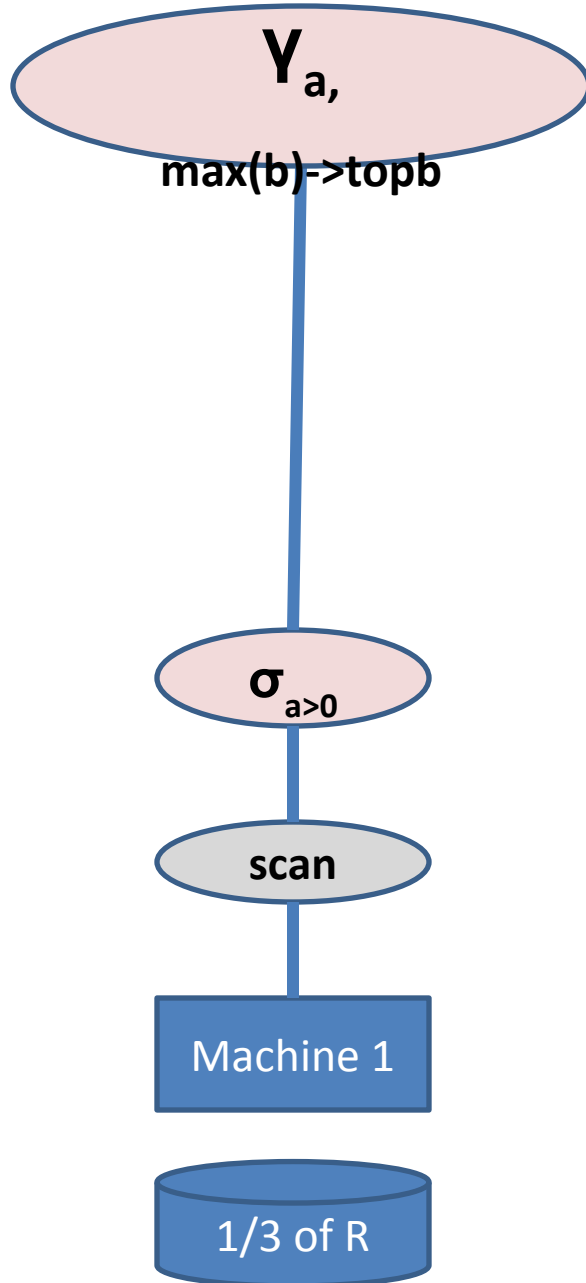
```
SELECT a, max(b) as topb  
FROM R  
WHERE a > 0  
GROUP BY a
```

Benefit of hash-partitioning

- **For parallel DBMS**
 - It would avoid the data re-shuffling phase
 - It would compute the aggregates locally

Hash-partition on a for R(a, b)

SELECT a, max(b) as topb FROM R
WHERE a > 0 GROUP BY a



Problem 1)

Consider relations $R(a,b)$, $S(c,d)$, and $T(e,f)$. All three are horizontally partitioned across $N = 3$ machines.

The tuples are randomly organized across machines.

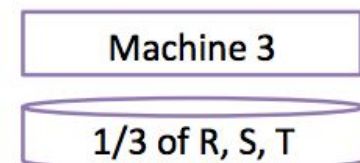
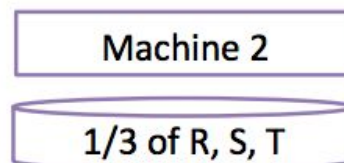
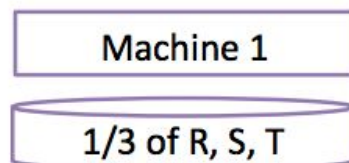
Show a relational algebra plan for the following query and how it will be executed across the $N = 3$ machines:

```
SELECT *  
FROM R, S, T  
WHERE R.b = S.c  
        AND S.d = T.e  
        AND (R.a - T.f) > 100
```

Problem 1)

R(a,b)
S(c,d)
T(e,f)

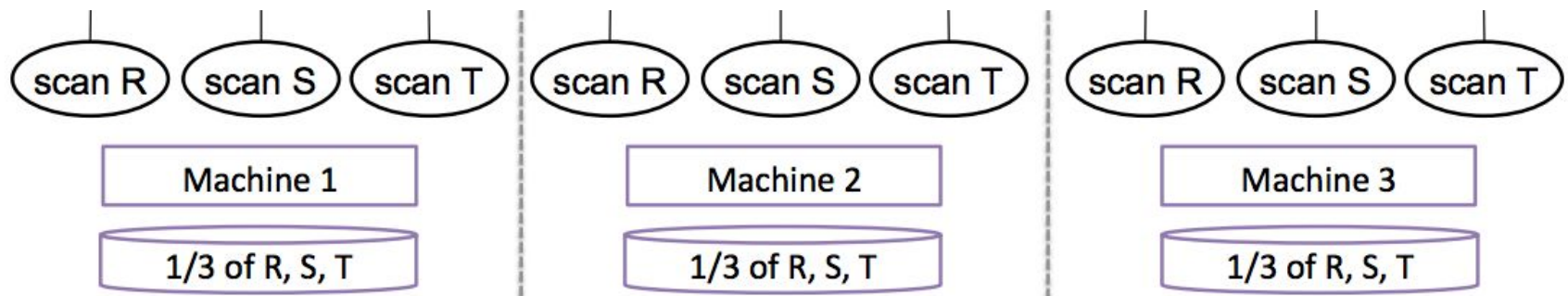
```
SELECT *  
FROM R, S, T  
WHERE R.b = S.c  
      AND S.d = T.e  
      AND (R.a - T.f) > 100
```



Problem 1)

R(a,b)
S(c,d)
T(e,f)

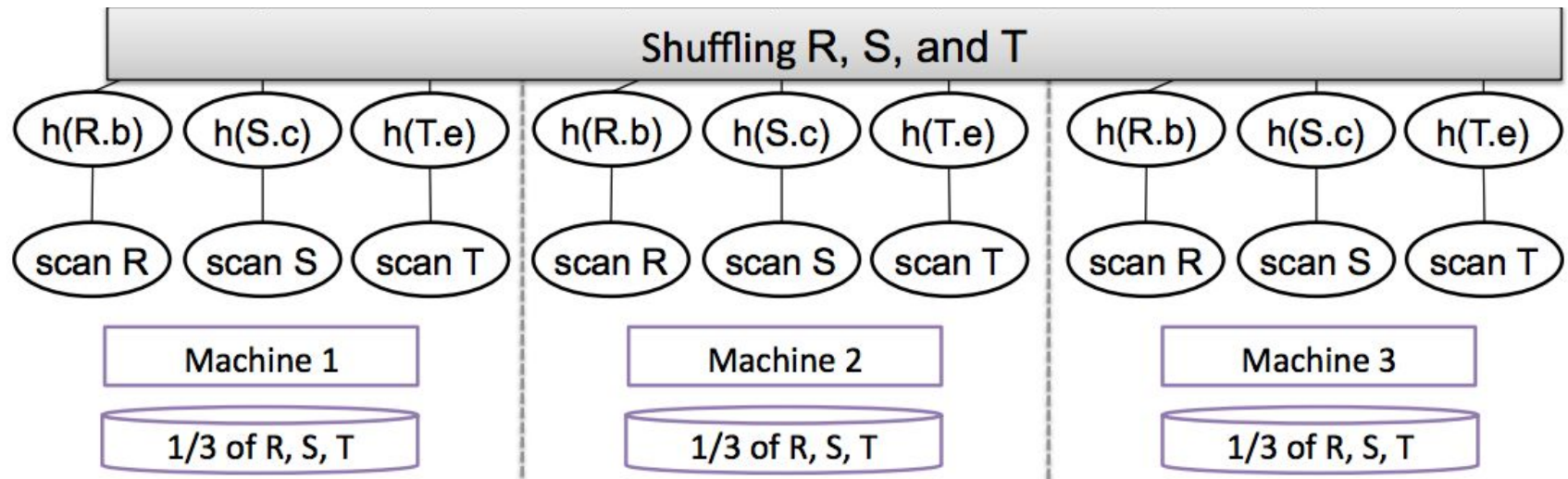
```
SELECT *  
FROM R, S, T  
WHERE R.b = S.c  
      AND S.d = T.e  
      AND (R.a - T.f) > 100
```



Problem 1)

R(a,b)
S(c,d)
T(e,f)

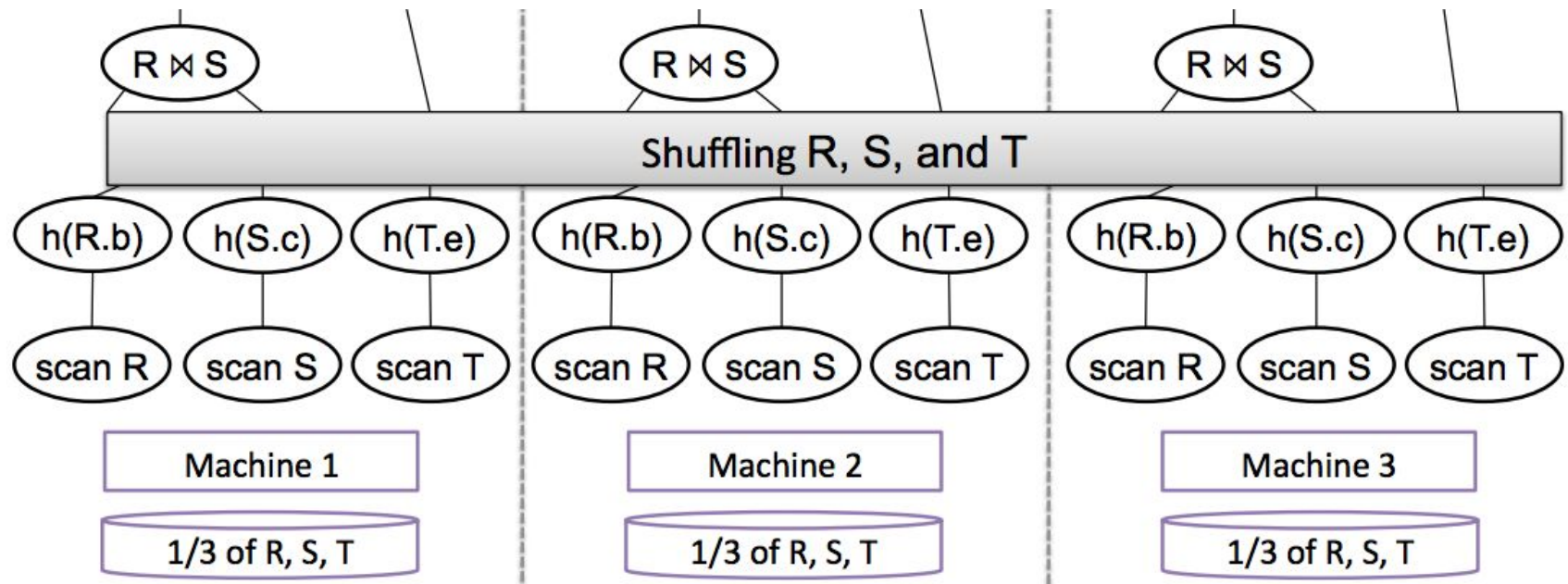
```
SELECT *  
FROM R, S, T  
WHERE R.b = S.c  
      AND S.d = T.e  
      AND (R.a - T.f) > 100
```



Problem 1)

R(a,b)
S(c,d)
T(e,f)

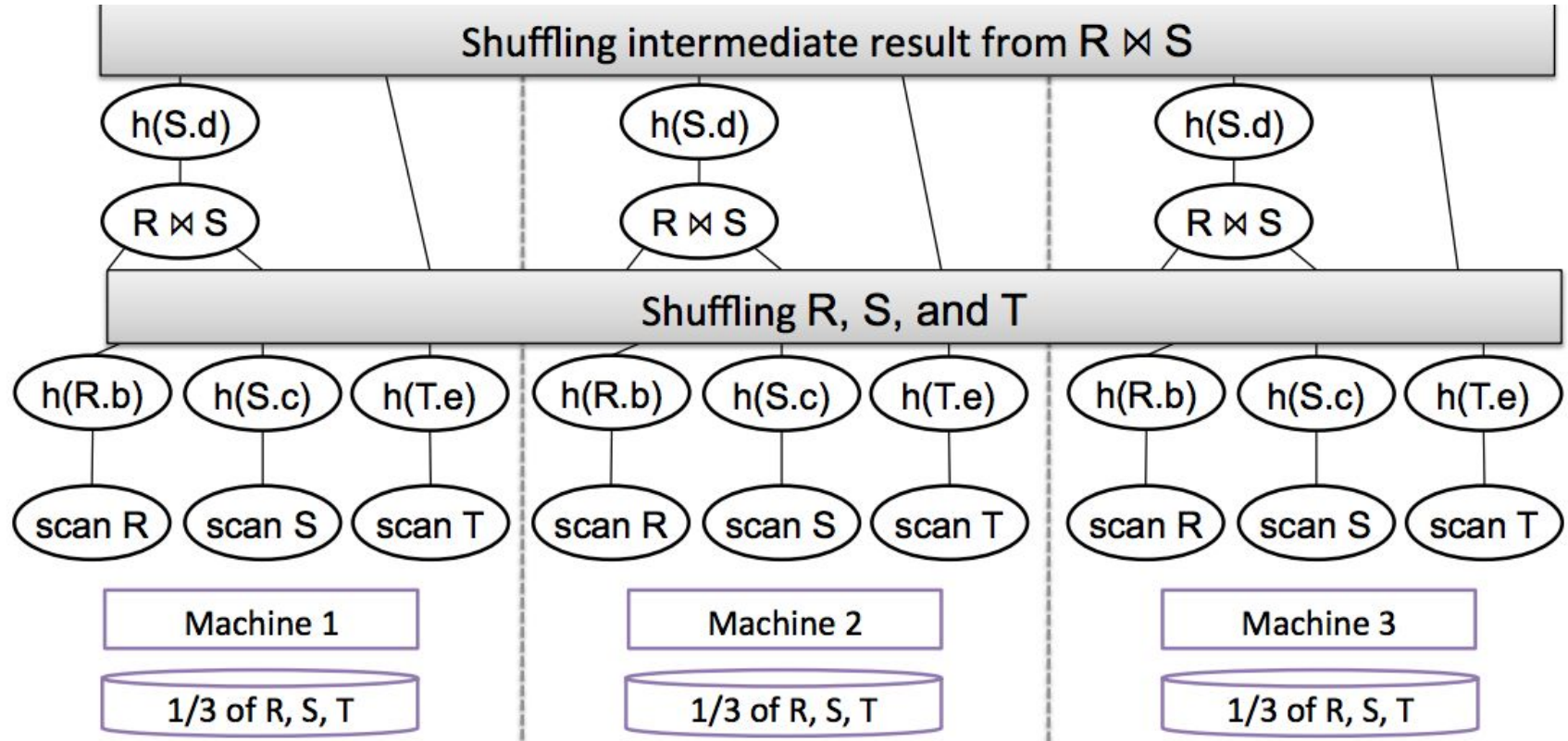
```
SELECT *  
FROM R, S, T  
WHERE R.b = S.c  
      AND S.d = T.e  
      AND (R.a - T.f) > 100
```



Problem 1)

R(a,b)
S(c,d)
T(e,f)

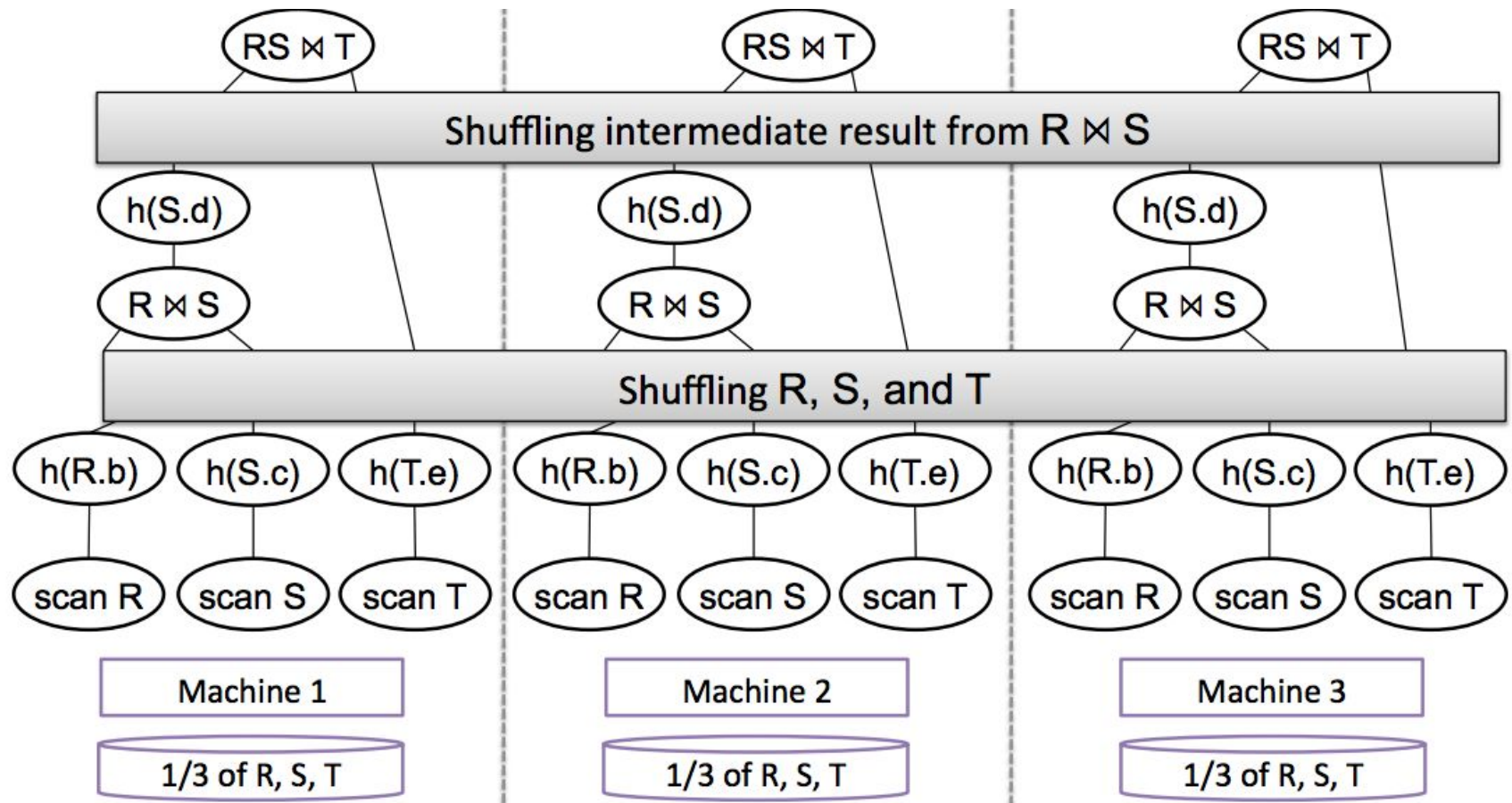
```
SELECT *
FROM R, S, T
WHERE R.b = S.c
      AND S.d = T.e
      AND (R.a - T.f) > 100
```



Problem 1)

R(a,b)
S(c,d)
T(e,f)

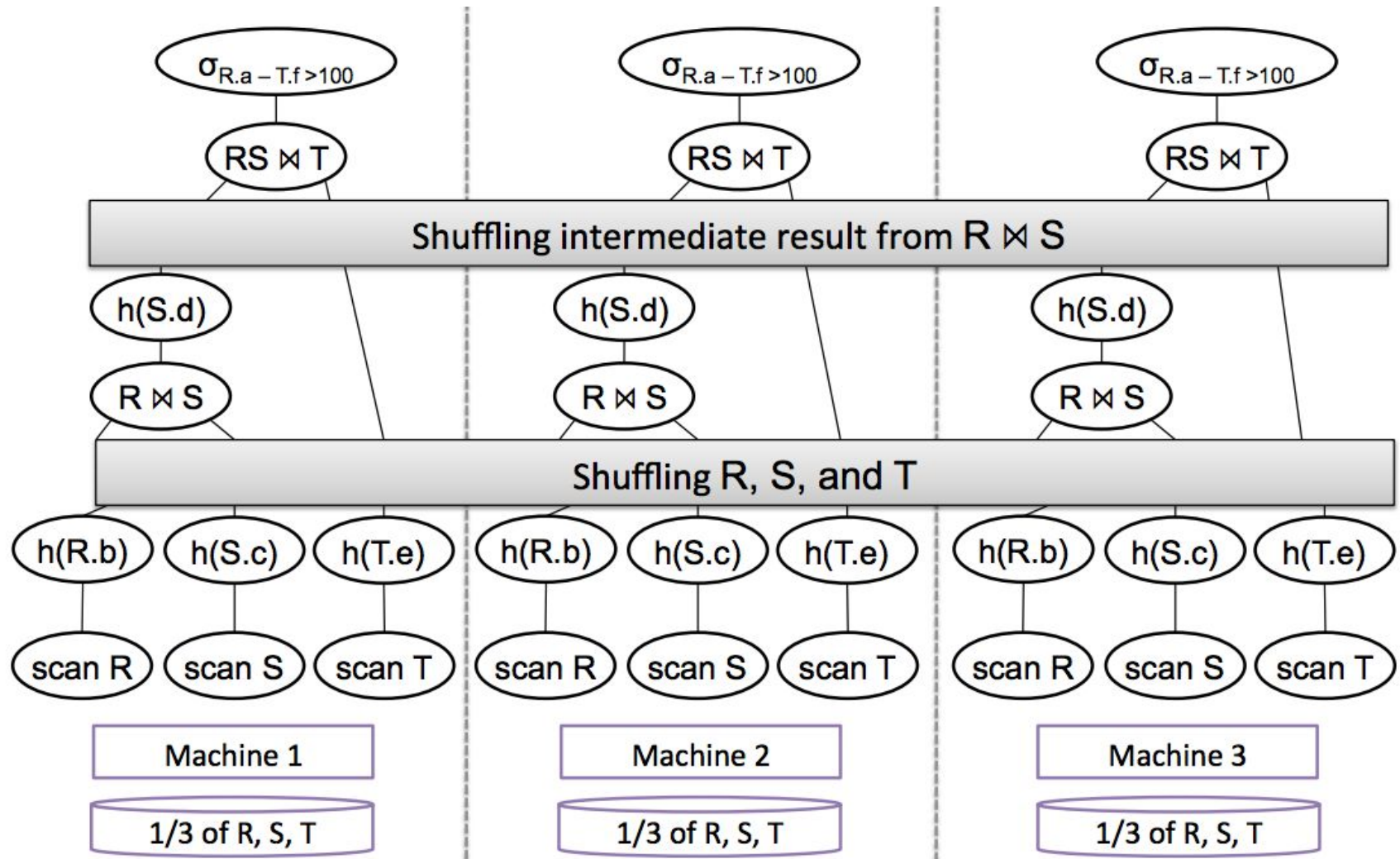
SELECT *
FROM R, S, T
WHERE R.b = S.c
 AND S.d = T.e
 AND (R.a - T.f) > 100



Problem 1)

R(a,b)
S(c,d)
T(e,f)

SELECT *
FROM R, S, T
WHERE R.b = S.c
AND S.d = T.e
AND (R.a - T.f) > 100



Map Reduce

Explain how the query will be executed in MapReduce

```
SELECT a, max(b) as topb  
FROM R  
WHERE a > 0  
GROUP BY a
```

Specify the computation performed in the map and the reduce functions

R(a, b)

```
SELECT a, max(b) as topb  
FROM R  
WHERE a > 0  
GROUP BY a
```

Map

- Each **map** task
 - Scans a block of R
 - Calls the map function for each tuple
 - The map function applies the selection predicate to the tuple
 - For each tuple satisfying the selection, it outputs a record with key = a and value = b

Note: When each map task scans multiple relations, it needs to output something like **key = a and value = ('R', b)** which has the relation name 'R'

R(a, b)

```
SELECT a, max(b) as topb  
FROM R  
WHERE a > 0  
GROUP BY a
```

Shuffle

- The MapReduce engine reshuffles the output of the map phase and groups it on the intermediate key, i.e. the attribute a

Note: the programmer has to write only the map and reduce functions, the shuffle phase is done by the MapReduce engine (although the programmer can rewrite the partition function).

R(a, b)

```
SELECT a, max(b) as topb  
FROM R  
WHERE a > 0  
GROUP BY a
```

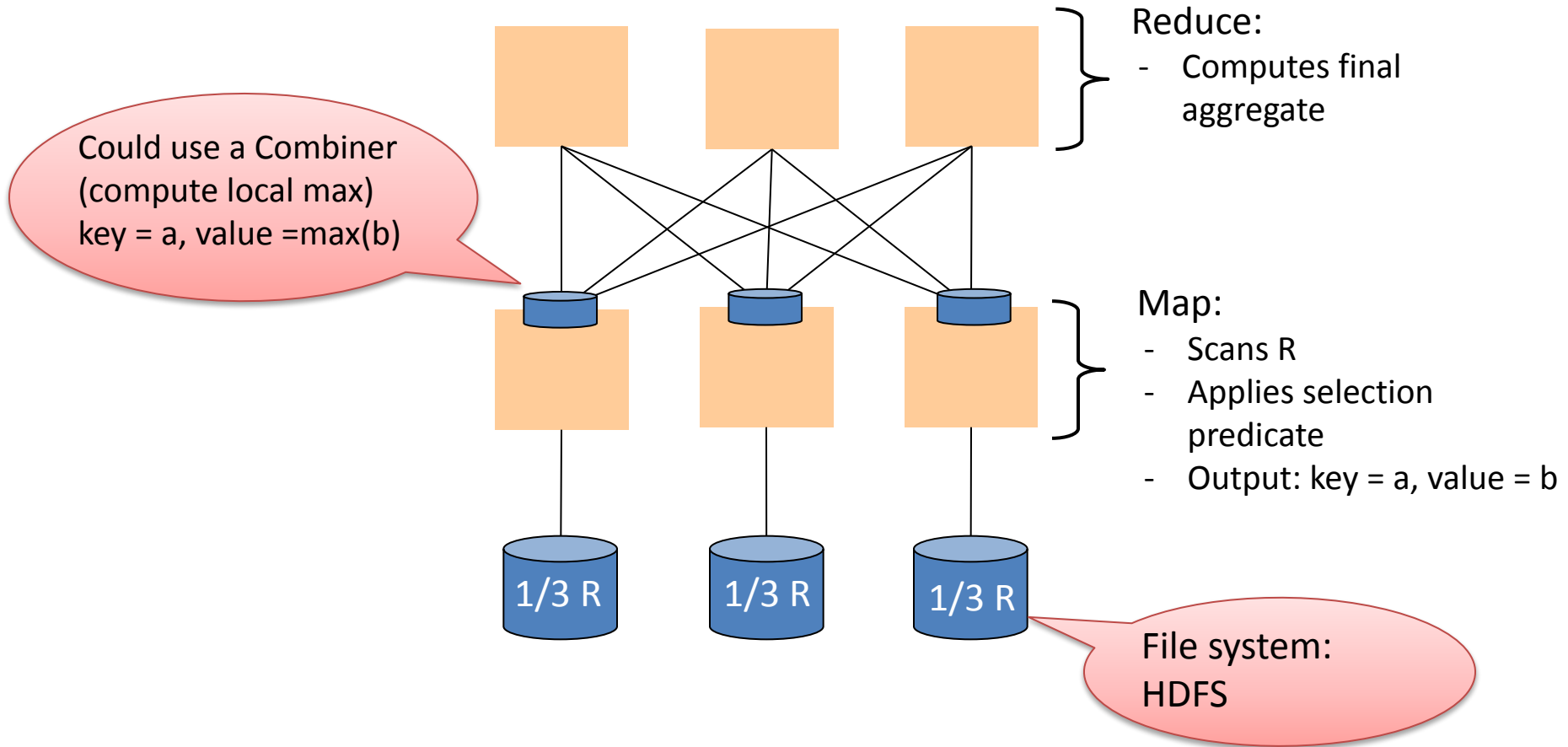
Reduce

- Each reduce task
 - computes the aggregate value **max(b) = topb** for each group (i.e. **a**) assigned to it (by calling the reduce function)
 - outputs the final results: **(a, topb)**

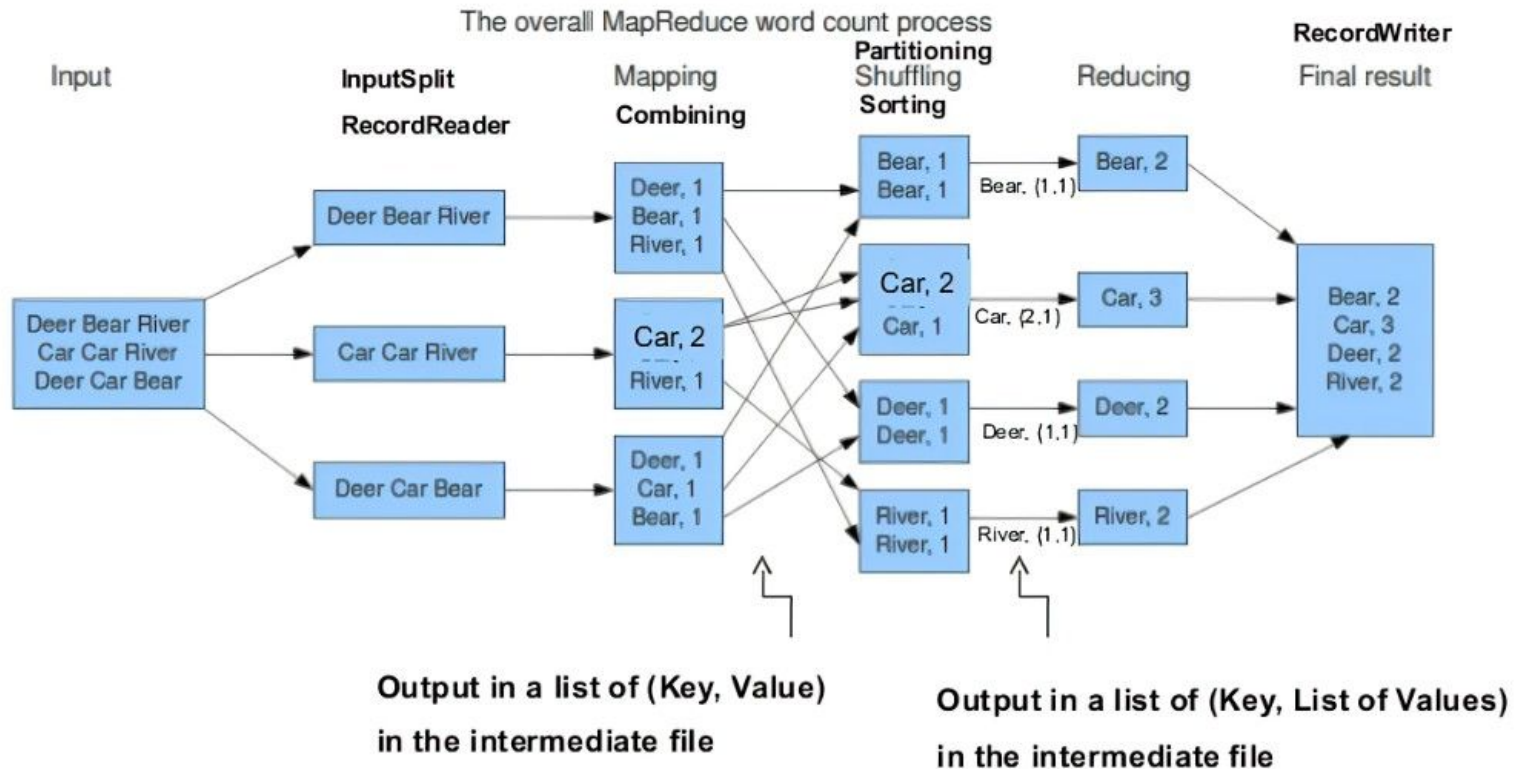
Note: A local combiner can be used to compute local max before data gets reshuffled (in the map tasks)

$R(a, b)$

```
SELECT a, max(b) as topb
FROM R
WHERE a > 0
GROUP BY a
```



How does the MapReduce work?



```
SELECT a, max(b) as topb  
FROM R  
WHERE a > 0  
GROUP BY a
```

Benefit of hash-partitioning

- **For MapReduce**

- Logically, MR won't know that the data is hash-partitioned
- MR treats map and reduce functions as black-boxes and does not perform any optimizations on them

- But, if a local combiner is used

- Saves communication cost:
 - fewer tuples will be emitted by the map tasks
- Saves computation cost in the reducers:
 - the reducers would not have to do as much work

Problem 2)

Consider two relations $R(a, b)$ and $S(b, c)$.

```
SELECT R.b, max(S.c) as cmax  
FROM R, S  
WHERE R.b = S.b  
        AND R.a <= 100  
GROUP BY R.b
```

For the **Map** function, what are the computations performed, and what will be its outputs? Assume that the Map function reads a block of R or S relation as input.

For the **Reduce** function, what will be its inputs, what are the computations performed, and what will be its outputs?

Problem 2)

```
SELECT R.b, max(S.c) as cmax
FROM R, S
WHERE R.b = S.b
      AND R.a <= 100
GROUP BY R.b
```

R(a, b)
S(b, c)

Note: In some cases, you may need to perform more than one MapReduce job to get the final result

Map Function:

- If the map function processes a block of the **R** relation, it applies the selection predicate to each R tuple in that block ($R.a \leq 100$), and if the tuple passes the selection, it outputs a record with key= R.b and value= ('R' , R.a).
- If the map function processes a block of the **S** relation, it outputs a record with key = S.b and value = ('S', S.c).

Reduce Function:

- Input to the reducer: The same b as the key and a list of R or S tuples ('R' , R.a) or ('S', S.c). In other words, we have ... (b, (value from R, value from S, value from S, etc ...))
- Computation: The reducer performs the local join of R and S and finds the max(S.c) value.

Comparing between Parallel DBMSs and MapReduce Systems

Parallel DBMS:

- Offers updates, transactions, indexing
- Pipelined parallelism

MapReduce:

- Fault-tolerance
- Can handle stragglers
- Easy to scale