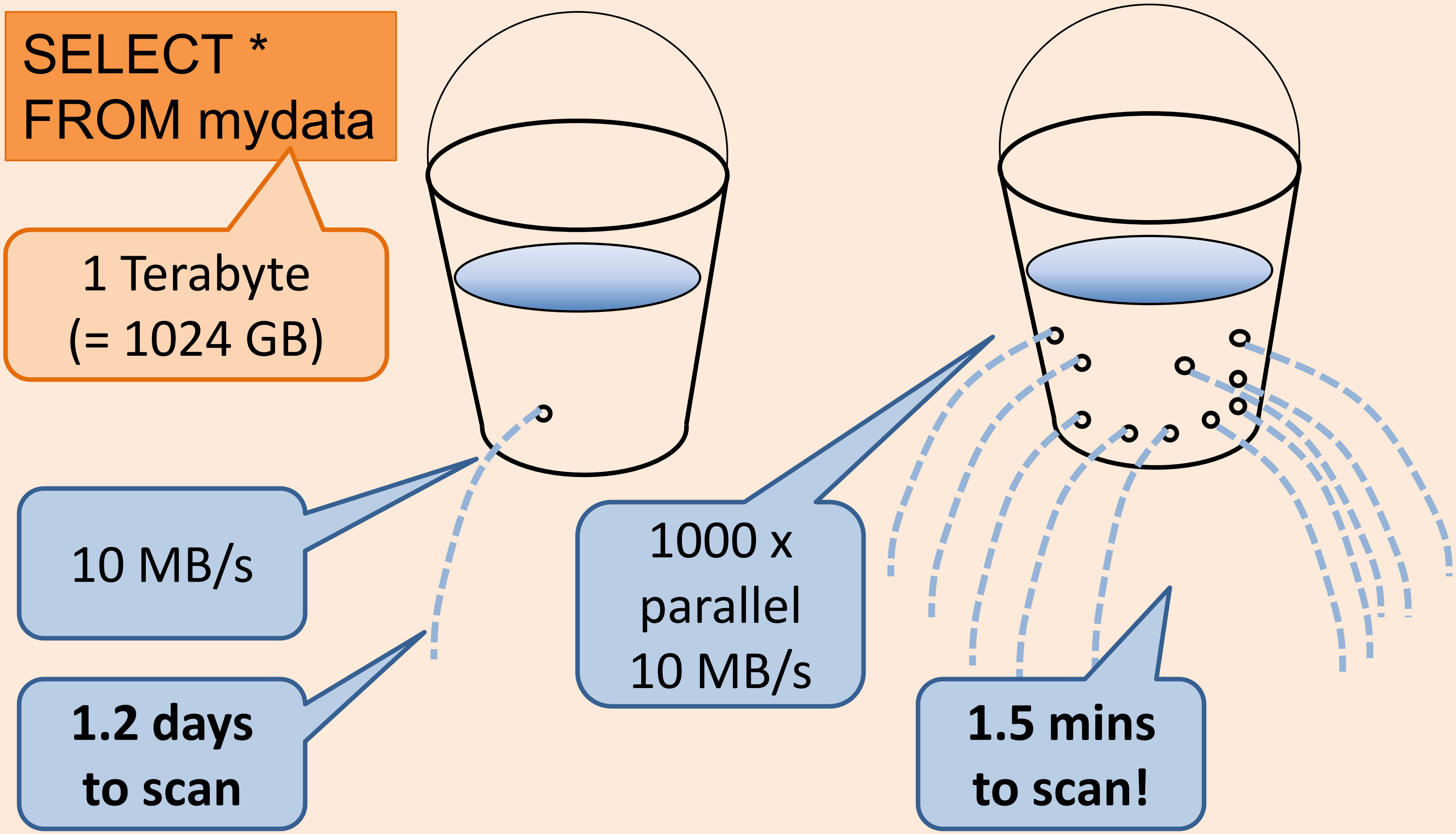
ICS 624 Spring 2013
Parallel Databases & Map-Reduce

Asst. Prof. Lipyeow Lim
Information & Computer Science Department
University of Hawaii at Manoa

Why Parallel Data Access ?



Parallel DBMS

- eBay's main Teradata data warehouse (DW):
  - > 2 petabytes of user data
  - 10s of 1000s of users
  - Millions of queries per day
  - 72 nodes
  - >140 GB/sec of I/O, or 2 GB/node/sec
- eBay's Greenplum DW
  - 6 1/2 petabytes of user data
  - 96 nodes
  - 200 MB/node/sec of I/O
- Walmart – 2.5 petabytes
- Bank of America – 1.5 petabytes

- Some parallel DBMSs besides the usual Oracle-IBM-MS trio:
  - Teradata
  - Netezza
  - Vertica
  - DATAllegro
  - Greenplum
  - Aster Data
  - Infobright
  - Kognitio, Kickfire, Dataupia, ParAccel, Exasol, ...

Parallelism

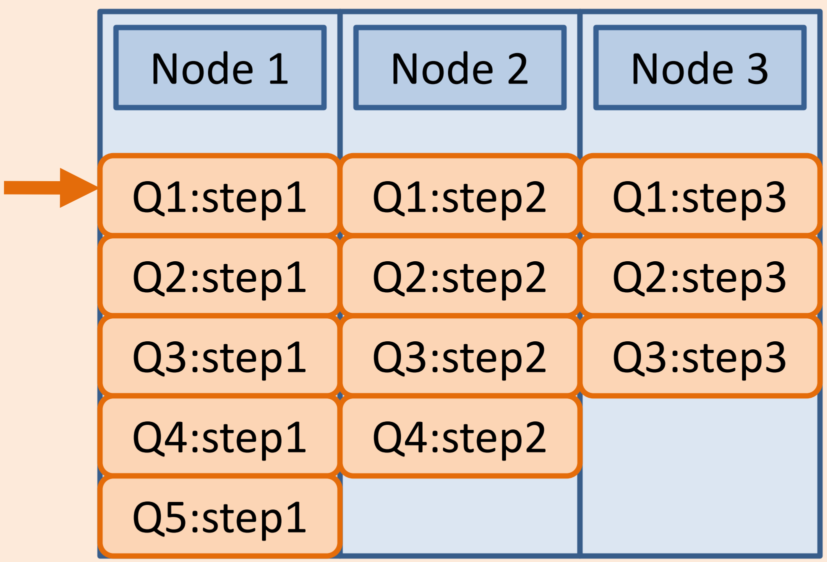
Pipeline parallelism

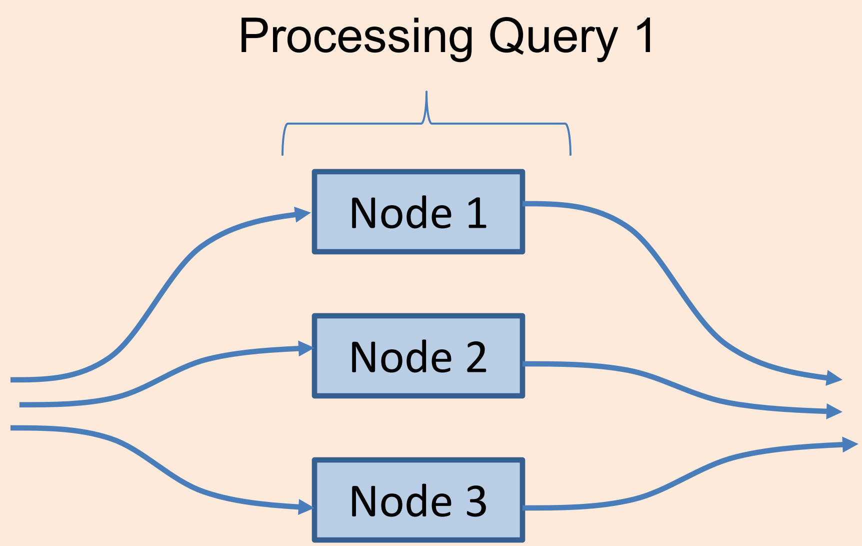
- many machines each doing one step in a multi-step process.

Partition parallelism

- many machines doing the same thing to different pieces of data.

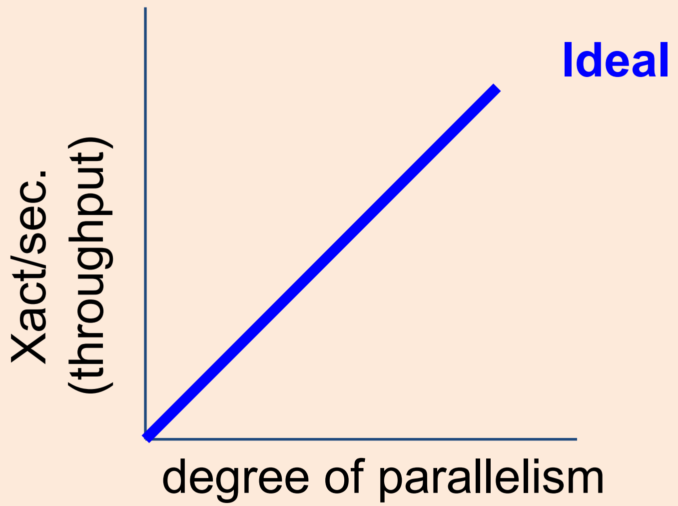
Parallelism is natural to DBMS processing

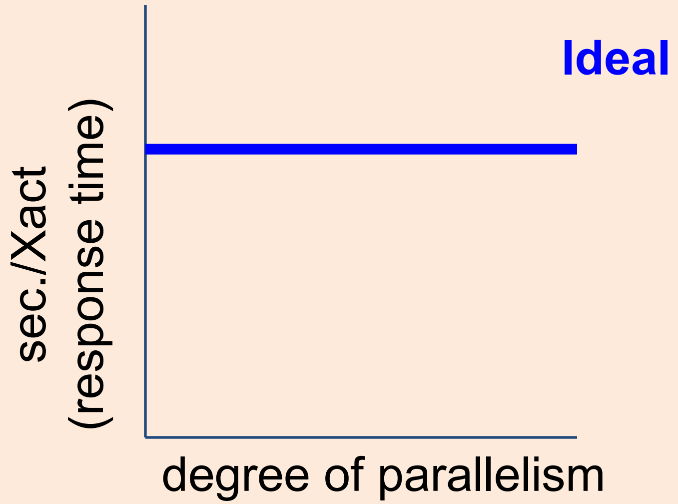




Parallelism Terminology

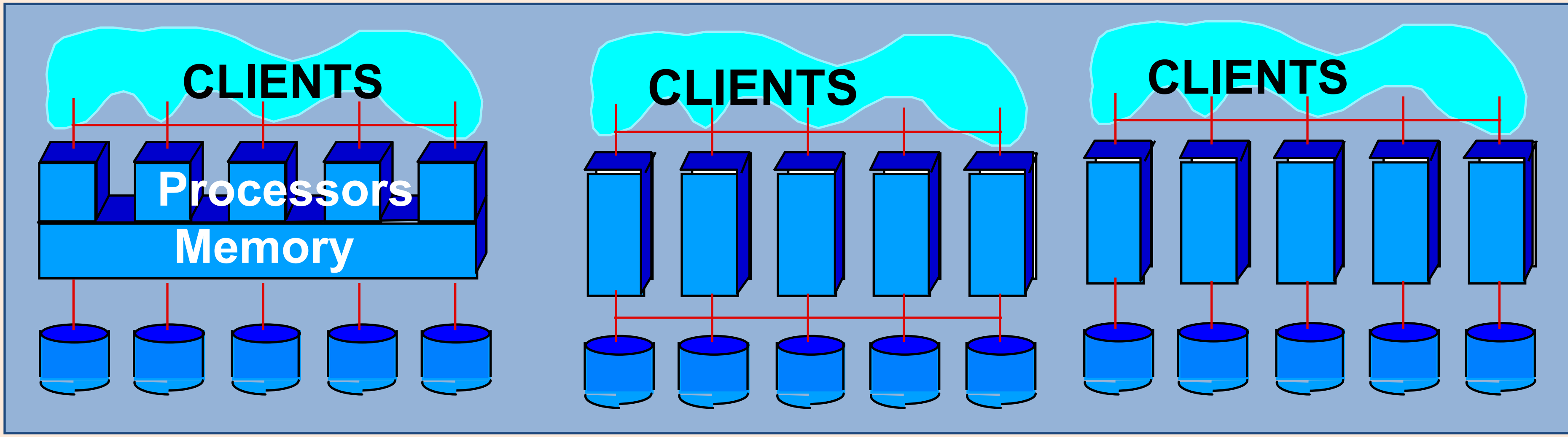
- Speed-up
  - Same job + more resources = less time
- Scale-up
  - Bigger job + more resouces = same time
- Transaction scale-up
  - More clients + more resources = same time





Parallel Architecture : Share What?

Shared Memory (SMP)	Shared Disk	Shared Nothing (network)



Easy to program Expensive to build Difficult to scaleup		Hard to program Cheap to build Easy to scaleup
Sequent, SGI, Sun	VMScluster, Sysplex	Tandem, Teradata, SP2

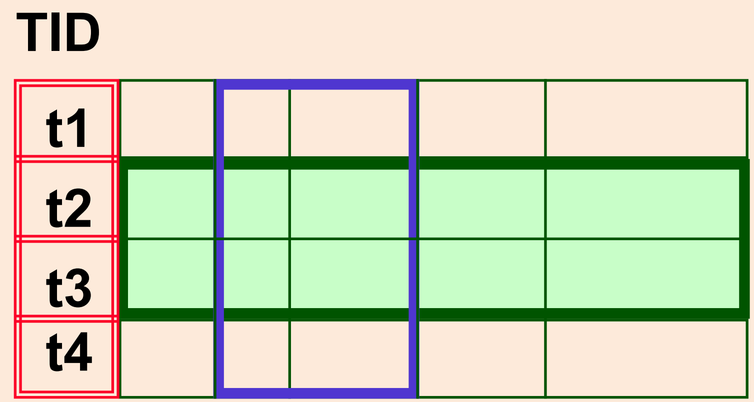
Different Types of DBMS Parallelism

- Intra-operator parallelism
  - get all machines working to compute a given operation (scan, sort, join)
- Inter-operator parallelism
  - each operator may run concurrently on a different site (exploits pipelining)
- Inter-query parallelism
  - different queries run on different sites
- We’ll focus on intra-query parallelism

Parallel vs Distributed DBMS

- A parallel database system
  - Parallelize various operations such as loading data, building indexes, evaluating queries
  - Often homogeneous: Every node runs same type of DBMS.
- A distributed database system
  - Data is physically stored across several (geographical) sites
  - Distribution governed by factors like local ownership & increased availability
  - Often heterogeneous: Different sites run different DBMSs (different RDBMSs or even non-relational DBMSs).
- The boundaries of these traditional definitions are blurring.

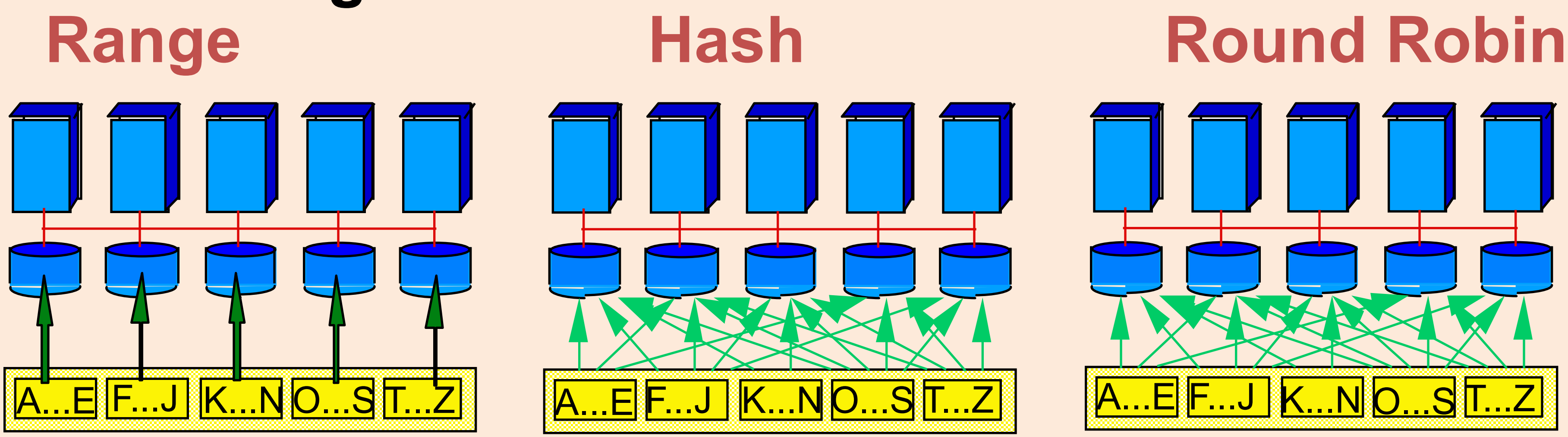
Data Partitioning & Fragmentation



- Parallel DB
  - Data partitioning
- Distributed DB
  - Fragmentation
- Same basic problem : **How do we break up the data (tables) and spread them amongst the "nodes"**
  - Horizontal vs Vertical
  - Range vs Hash
  - Replication
- DB user's view should be one single table.

Automatic Data Partitioning

Partitioning a table:



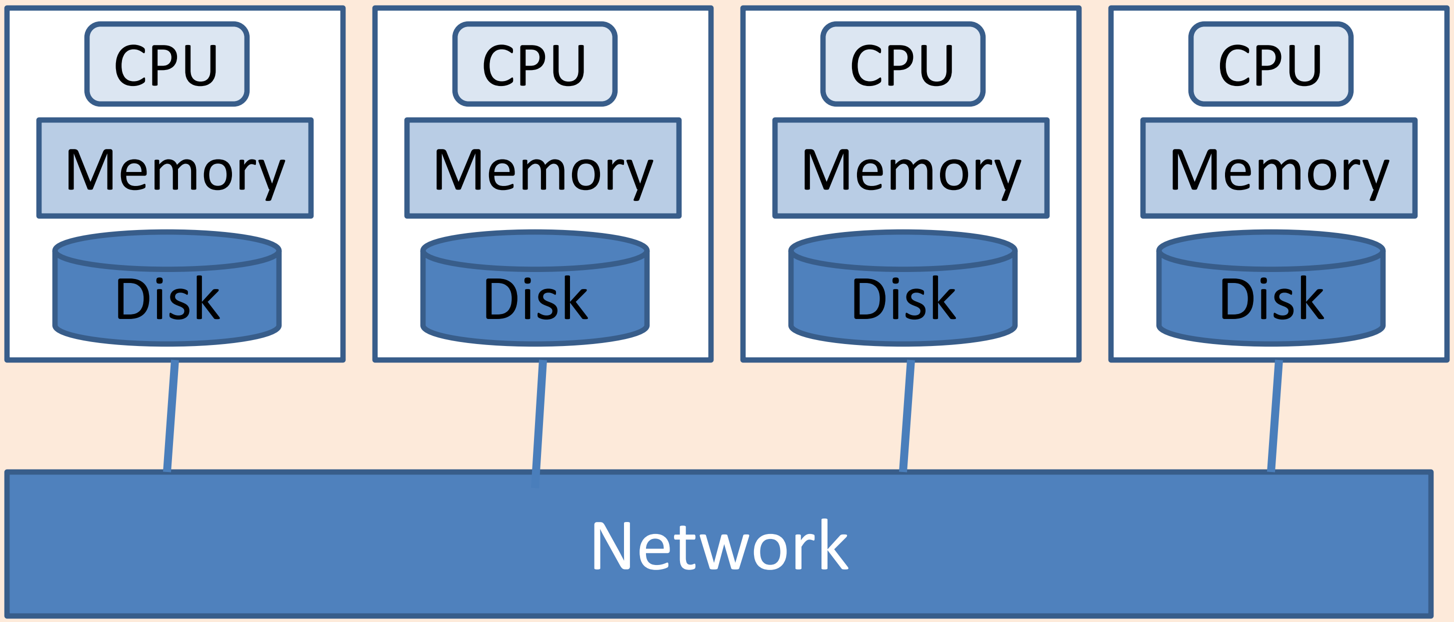
Good for equijoins, range queries group-by

Good for equijoins

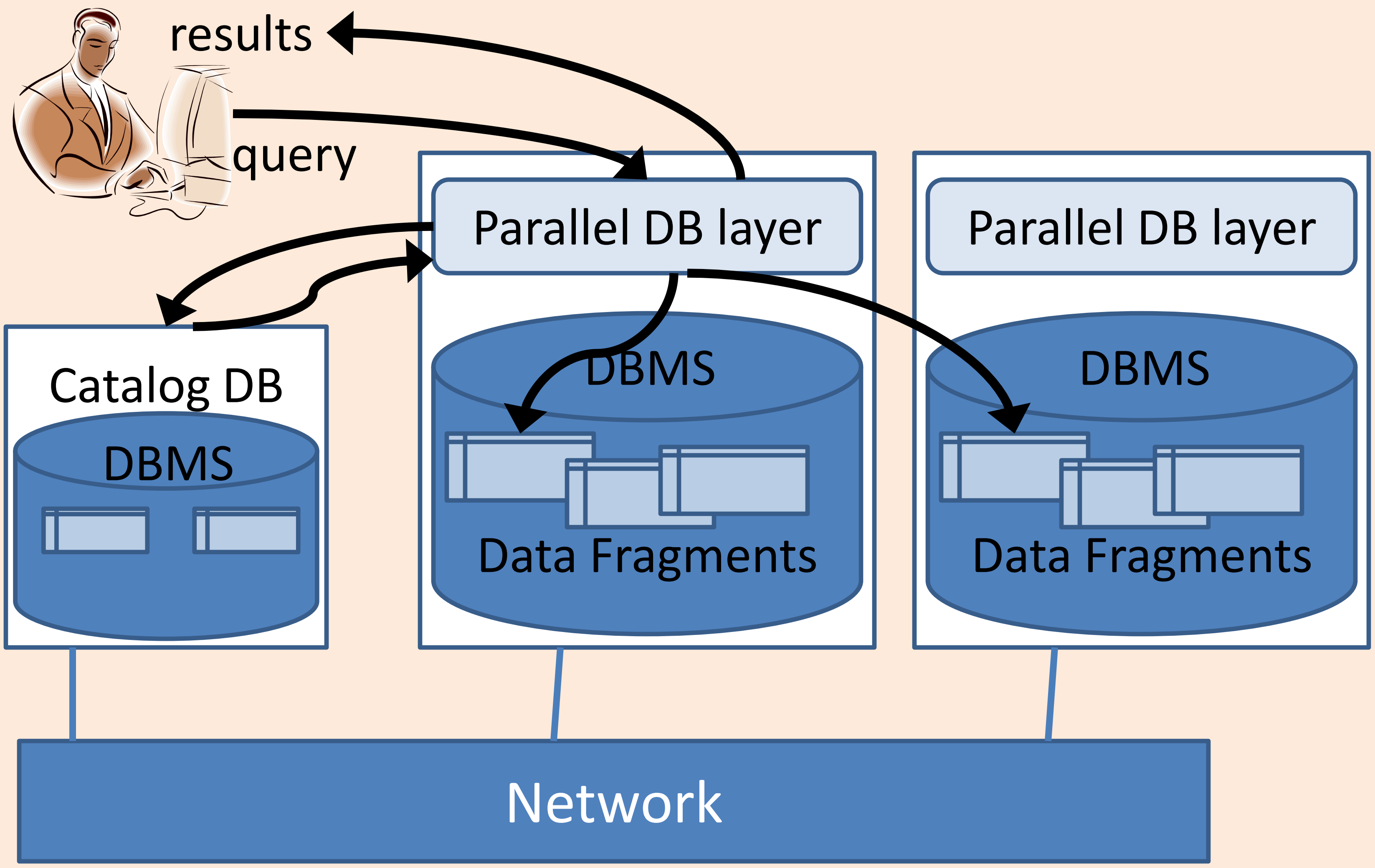
Good to spread load

- **Shared disk and memory less sensitive to partitioning,**
- **Shared nothing benefits from "good" partitioning**

Shared-Nothing Architecture



Logical Parallel DBMS Architecture



Horizontal Fragmentation: Range Partition

sid	sname	rating	age
22	dustin	7	45
29	brutus	1	33
31	lubber	8	55
32	andy	4	23
58	rusty	10	35
64	horatio	7	35

Partition 1

sid	sname	rating	age
29	brutus	1	33
32	andy	4	23

Partition 2

sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35
64	horatio	7	35

Range Partition on rating column

- Partition 1: 0 <= rating < 5
- Partition 2: 5 <= rating <= 10

Range Partition: Query Processing

- Which partitions?
- Better than non-parallel ?

SELECT *
FROM Sailors S

SELECT *
FROM Sailors S
WHERE rating = 2

SELECT *
FROM Sailors S
WHERE age > 30

SELECT *
FROM Sailors S
WHERE rating < 2 and age < 30

Partition 1

sid	sname	rating	age
29	brutus	1	33
32	andy	4	23

Partition 2

sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35
64	horatio	7	35

Horizontal Fragmentation: Hash Partition

sid	sname	rating	age
22	dustin	7	45
29	brutus	1	33
31	lubber	8	55
32	andy	4	23
58	rusty	10	35
64	horatio	7	35

Partition 1

sid	sname	rating	age
31	lubber	8	55
32	andy	4	23
58	rusty	10	35

Partition 2

sid	sname	rating	age
22	dustin	7	45
29	brutus	1	33
64	horatio	7	35

- Hash partitioning using hash function
  - Partition = rating mod 2

Hash Partition: Query Processing

- Which partitions?
- Better than non-parallel ?

```
SELECT *
FROM Sailors S
```

```
SELECT *
FROM Sailors S
WHERE rating = 2
```

```
SELECT *
FROM Sailors S
WHERE age  > 30
```

```
SELECT *
FROM Sailors S
WHERE rating < 2 and age < 30
```

Partition 1

sid	sname	rating	age
31	lubber	8	55
32	andy	4	23
58	rusty	10	35

Partition 2

sid	sname	rating	age
22	dustin	7	45
29	brutus	1	33
64	horatio	7	35

Vertical Fragmentation/Partition

sid	sname	rating	age
22	dustin	7	45
29	brutus	1	33
31	lubber	8	55
32	andy	4	23
58	rusty	10	35
64	horatio	7	35

Partition 1

sid	sname	rating
22	dustin	7
29	brutus	1
31	lubber	8
32	andy	4
58	rusty	10
64	horatio	7

Partition 2

sid	age
22	45
29	33
31	55
32	23
58	35
64	35

- Vertical partitioning
  - Use sid as row identifier

Vertical Partition: Query Processing

- Which partitions?
- Better than non-parallel ?

Partition 1

sid	sname	rating
22	dustin	7
29	brutus	1
31	lubber	8
32	andy	4
58	rusty	10
64	horatio	7

Partition 2

sid	age
22	45
29	33
31	55
32	23
58	35
64	35

```
SELECT *
FROM Sailors S
```

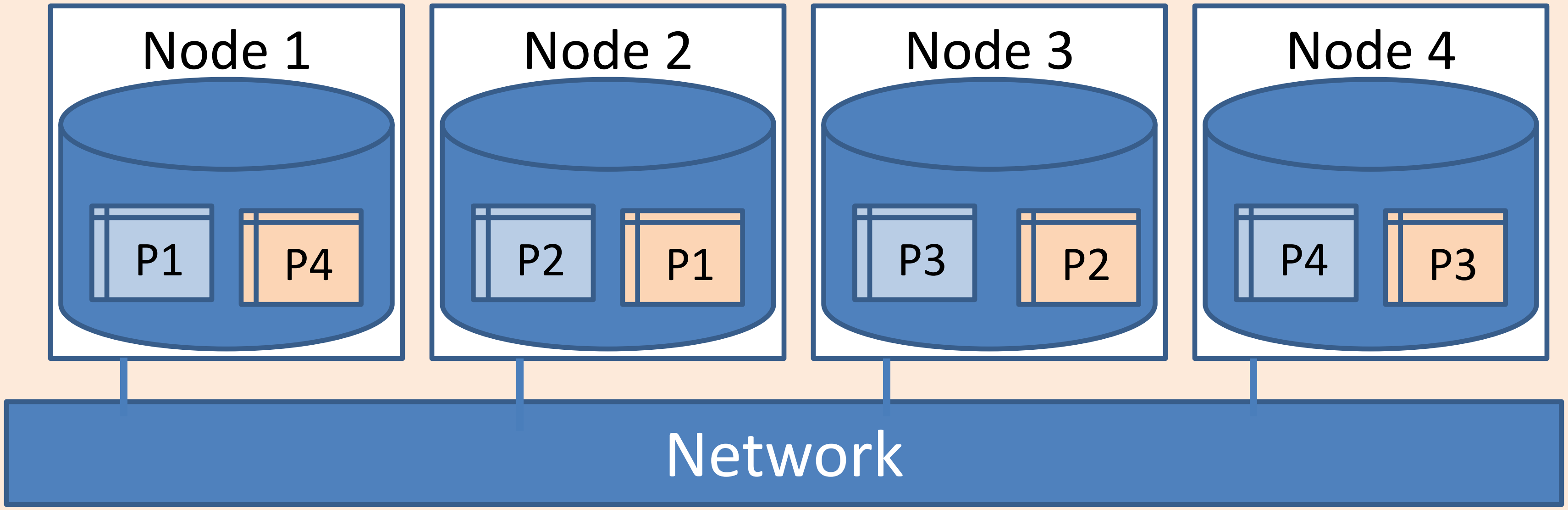
```
SELECT sname
FROM Sailors S
```

```
SELECT *
FROM Sailors S
WHERE rating = 2
```

```
SELECT sid
FROM Sailors S
WHERE age  > 30
```

```
SELECT sid
FROM Sailors S
WHERE rating < 2 and age < 30
```

Fragmentation & Replication



- Suppose table is fragmented into 4 partitions on 4 nodes
- Replication stores another partition on each node
  - What happens when 1 node fails ? 2 nodes ?
  - What happens when a row needs to be updated ?

What about joins ?

SELECT R.sid, R.bid
FROM Sailors S, Reserves R
WHERE S.sid=R.sid AND rating > 8

- Sailors: hash
  - part = rating mod 2
- Reserves: hash
  - part = sid mod 2
- Where to perform join ?
- What data to ship ?

Partition 1

Sailors

sid	sname	rating	age
31	lubber	8	55
32	andy	4	23
58	rusty	10	35

Reserves

sid	bid	day
31	101	…
29	103	…

Partition 2

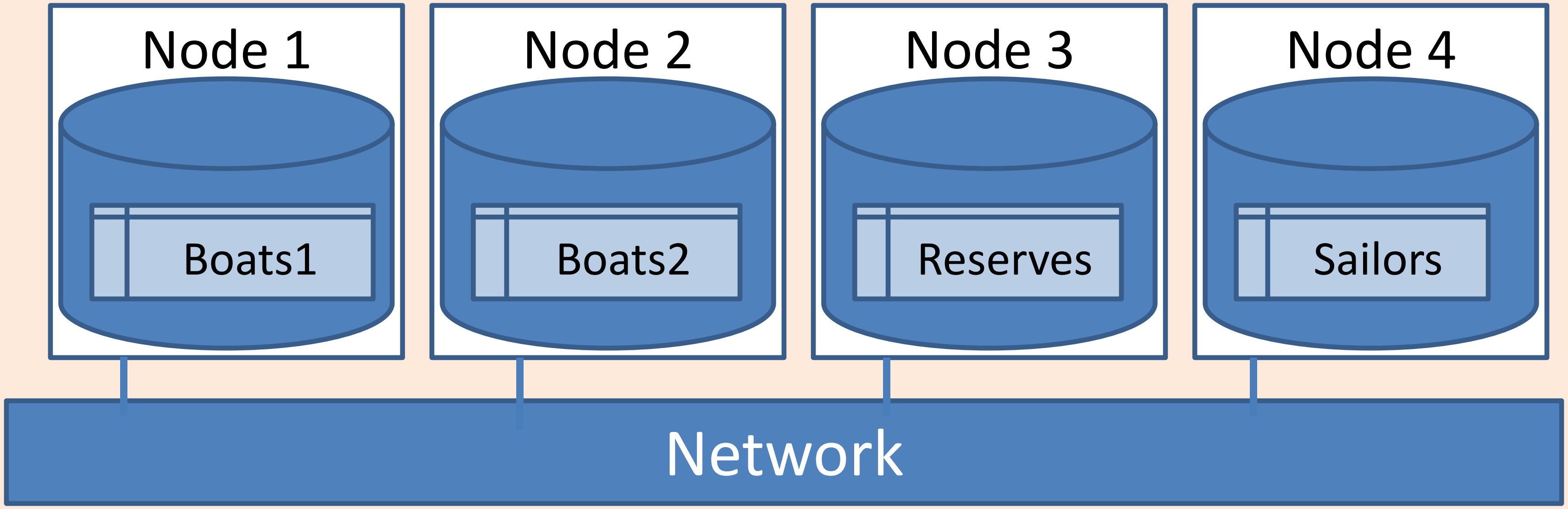
Sailors

sid	sname	rating	age
22	dustin	7	45
29	brutus	1	33
64	horatio	7	35

Reserves

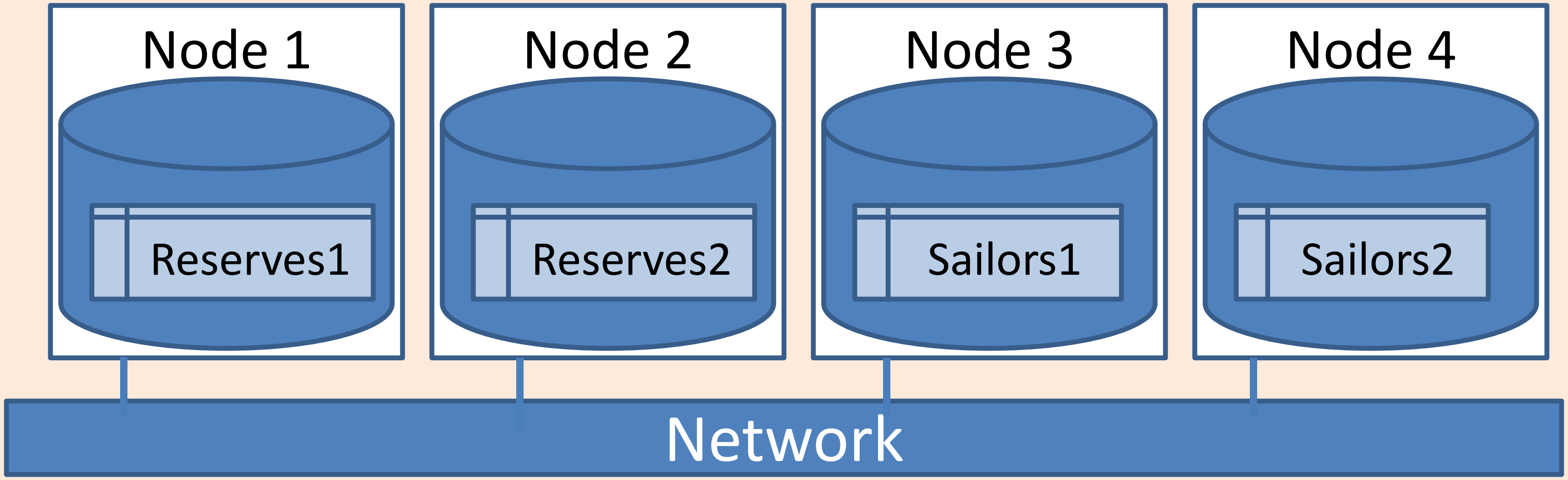
sid	bid	day
64	105	…
58	103	…

Distributed Joins



- Consider:
  - Reserves join Sailors
- Depends on:
  - Which node get the query
  - Whether tables are fragmented/partitioned or not
- Node 1 gets query
  - Perform join at Node 3 (or 4) ship results to Node 1 ?
  - Ship tables to Node 1 ?
- Node 3 gets query
  - Fetch sailors in loop ?
  - Cache sailors locally ?

Distributed Joins over Fragments



R join S

$= \sigma_{R.sid=S.sid} (R \times S)$

$= \sigma_{R.sid=S.sid} ((R1 \cup R2) \times (S1 \cup S2))$

$= \sigma_{R.sid=S.sid} ((R1 \times S1) \cup (R1 \times S2) \cup (R2 \times S1) \cup (R2 \times S2))$

$= \sigma_{R.sid=S.sid} (R1 \times S1) \cup \sigma_{R.sid=S.sid} (R1 \times S2) \cup \sigma_{R.sid=S.sid} (R2 \times S1) \cup \sigma_{R.sid=S.sid} (R2 \times S2)$

= (R1 join S1) ∪ (R1 join S2) ∪ (R2 join S1) ∪ (R2 join S2)

This equivalence applies to splitting a relation into pages in a single server DBMS system too!

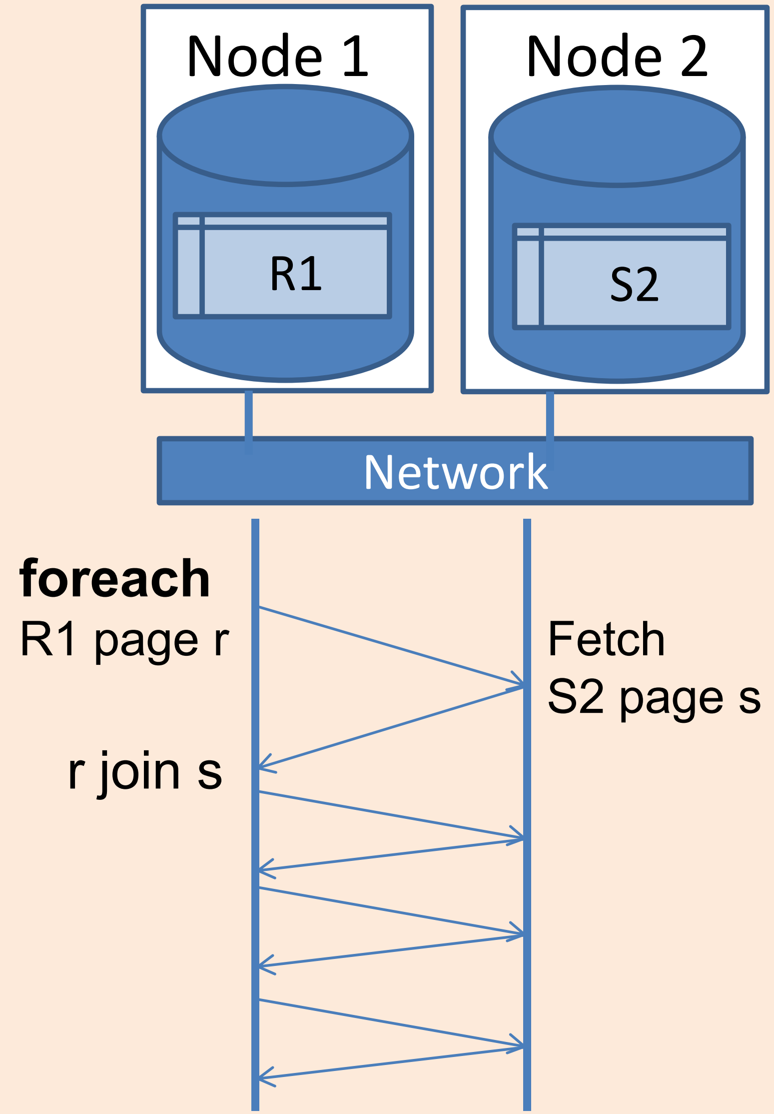
Equivalent to a union of joins over each pair of fragments

Distributed Nested Loop

- Consider performing R1 join S2 on Node 1
- Page-oriented nested loop join:

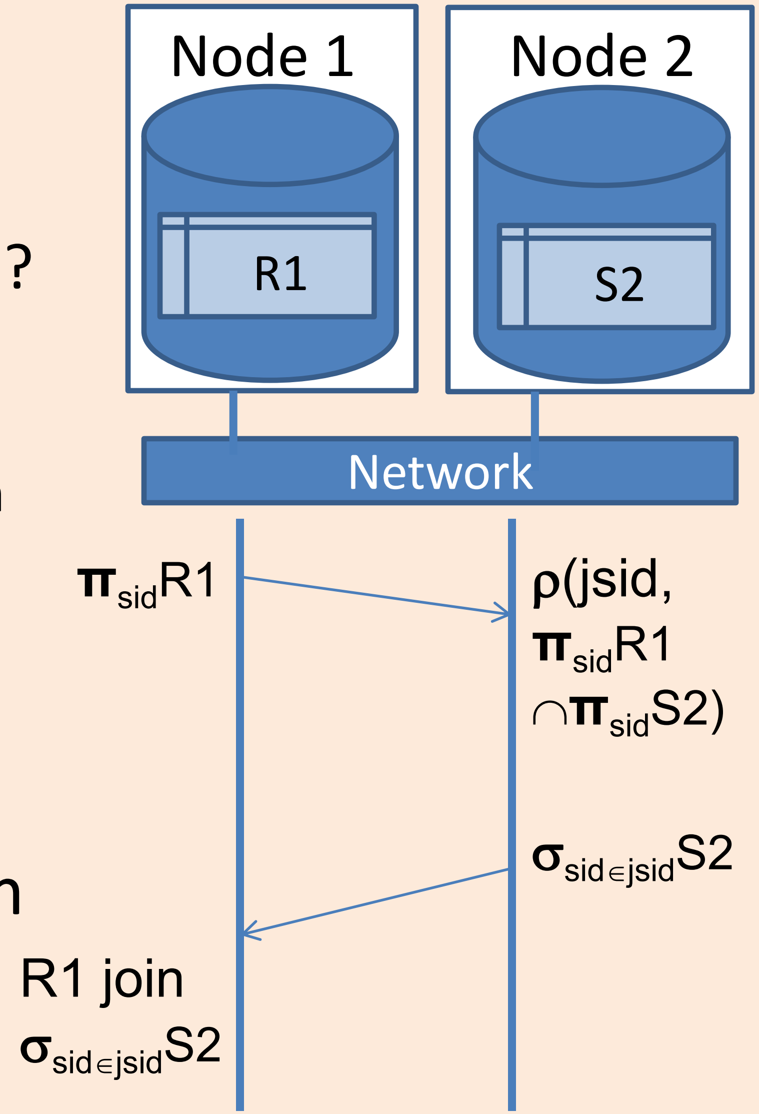
```
For each page r of R1
   Fetch r from local disk
   For each page s of S2
    Fetch s if s∉cache
    Output r join s
```

- Cost = Npages(R1)* t_d + Npages(R1)*Npages(S2)*(t_d + t_s)
- If cache can hold entire S2, cost is Npages(R1)* t_d +Npages(S2)* t_s +Npages(R1)*Npages(S2)*t_d



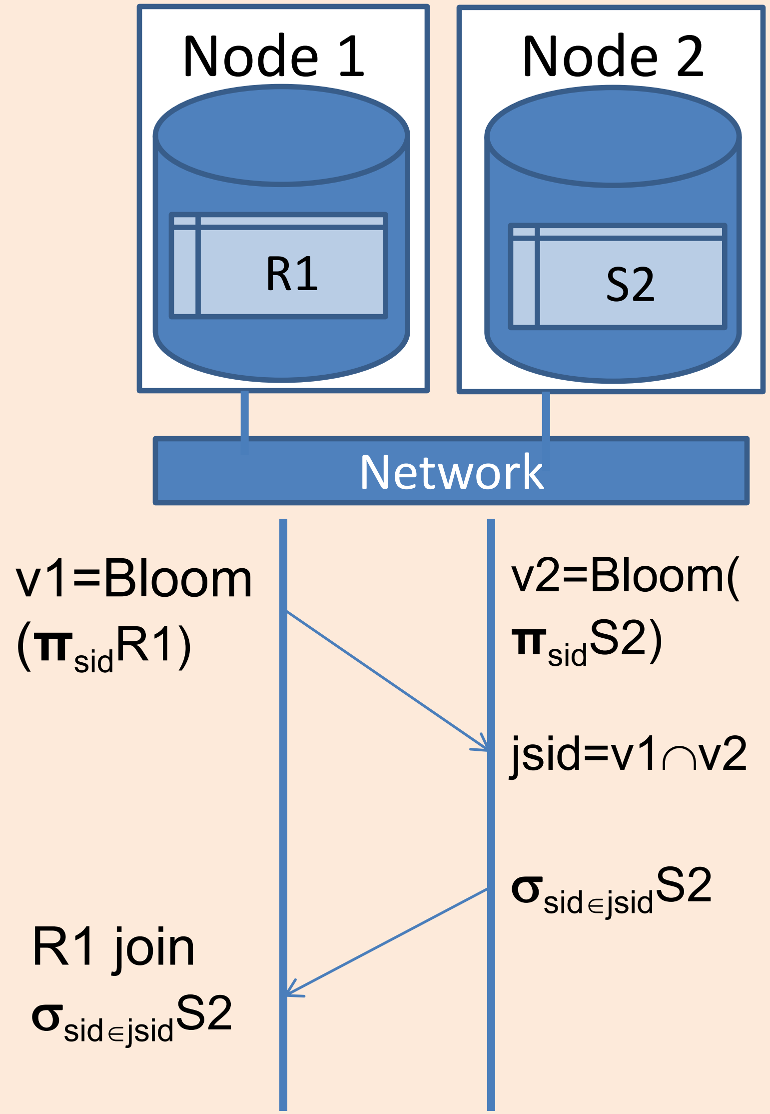
Semijoins

- Consider performing R1 join S2 on Node 1
- S2 needs to be shipped to R1
- Does every tuple in S2 join with R1 ?
- Semijoin:
  - Don't ship all of S2
  - Ship only those S2 rows that will join with R1
  - Assumes that the join causes a reduction in S2!
- Cost = $Npages(R1)*t_d + Npages(\pi_{sid}R1)*t_s + Cost(\cap) + Npages(\sigma_{sid \in jsid}S2)*t_s + Cost(R1 \text{ join } \sigma_{sid \in jsid}S2)$



Bloomjoins

- Consider performing R1 join S2 on Node 1
- Can we do better than semijoin ?
- Bloomjoin:
  - Don't ship all of ($\boldsymbol{\pi}_{sid}$R1)
  - Node 1: Ship a "bloom filter" (like a signature) of ($\boldsymbol{\pi}_{sid}$R1)
    - Hash each sid
    - Set the bit for hash value in a bit vector
    - Send the bit vector v1
  - Node 2:
    - Hash each ($\boldsymbol{\pi}_{sid}$S2) to bit vector v2
    - Computer (v1 $\cap$ v2)
    - Send rows of S2 in the intersection
- False positives



Google Map Reduce

Word Count over a Given Set of Web Pages

see bob throw

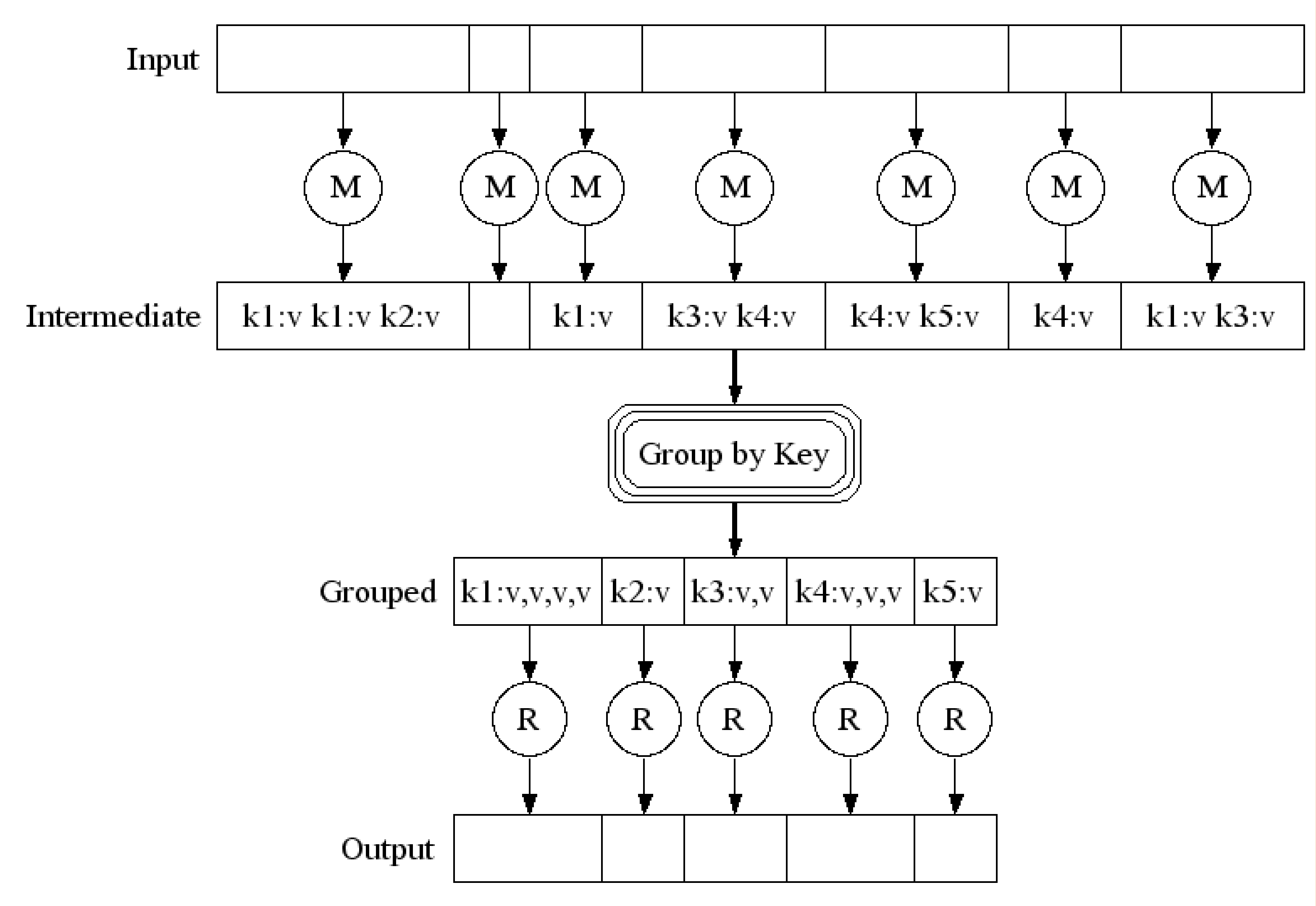
see spot run

see	1
bob	1
throw	1
see	1
spot	1
run	1

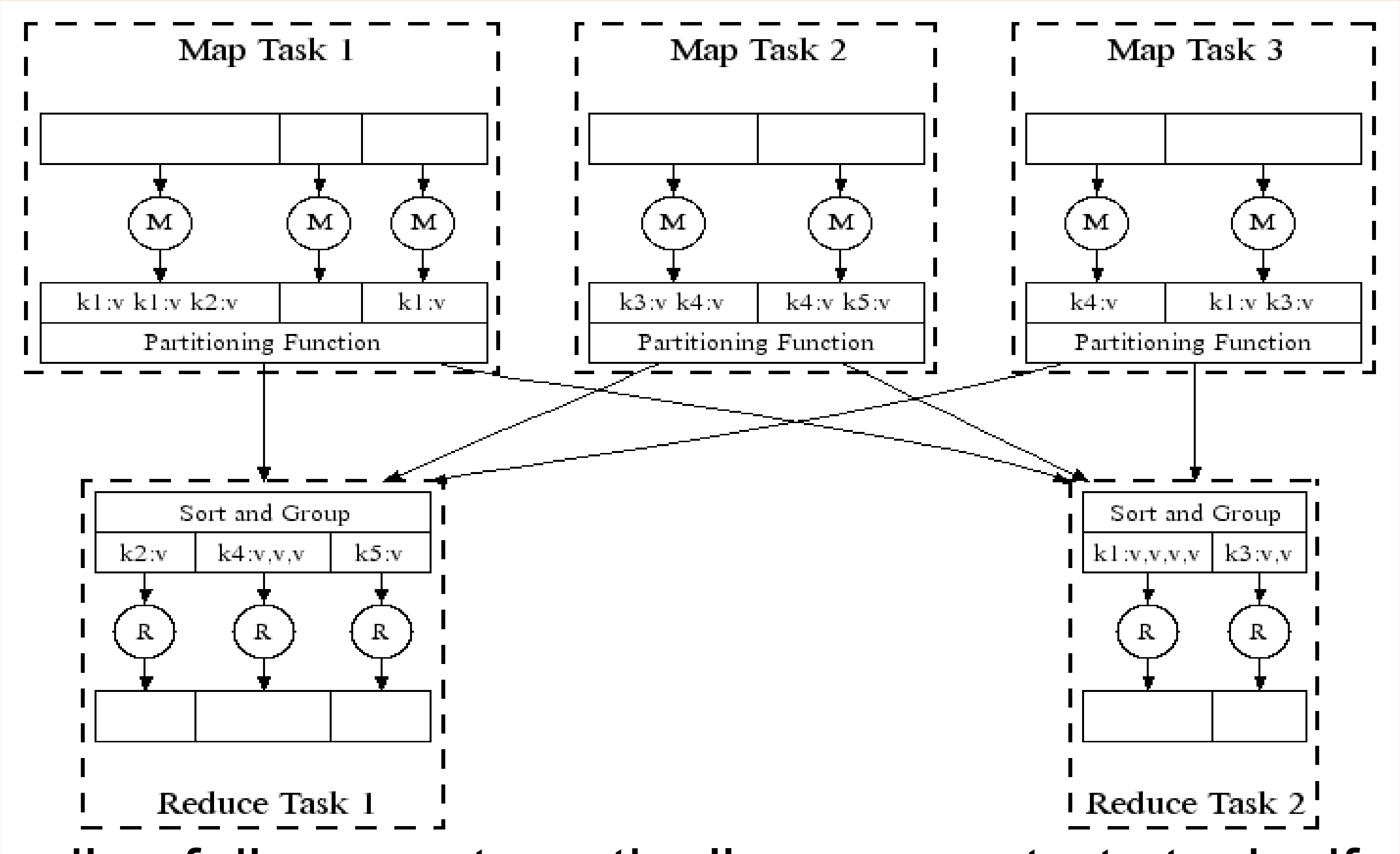
bob	1
run	1
see	2
spot	1
throw	1

Can we do word count in parallel?

The MapReduce Framework (pioneered by Google)



Automatic Parallel Execution in MapReduce (Google)



Handles failures automatically, e.g., restarts tasks if a node fails; runs multiples copies of the same task to avoid a slow task slowing down the whole job

MapReduce in Hadoop (1)

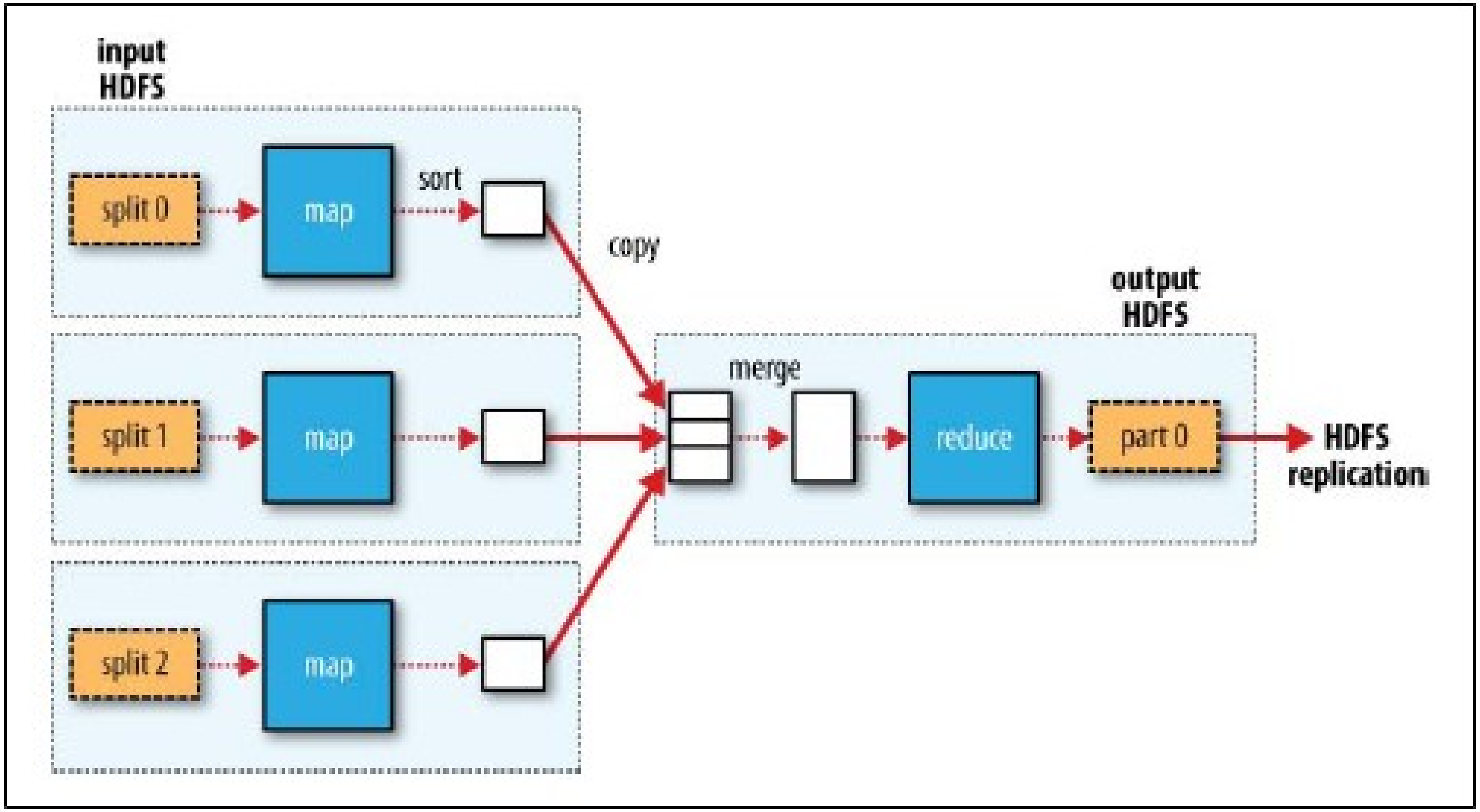


Figure 2-2. MapReduce data flow with a single reduce task

MapReduce in Hadoop (2)

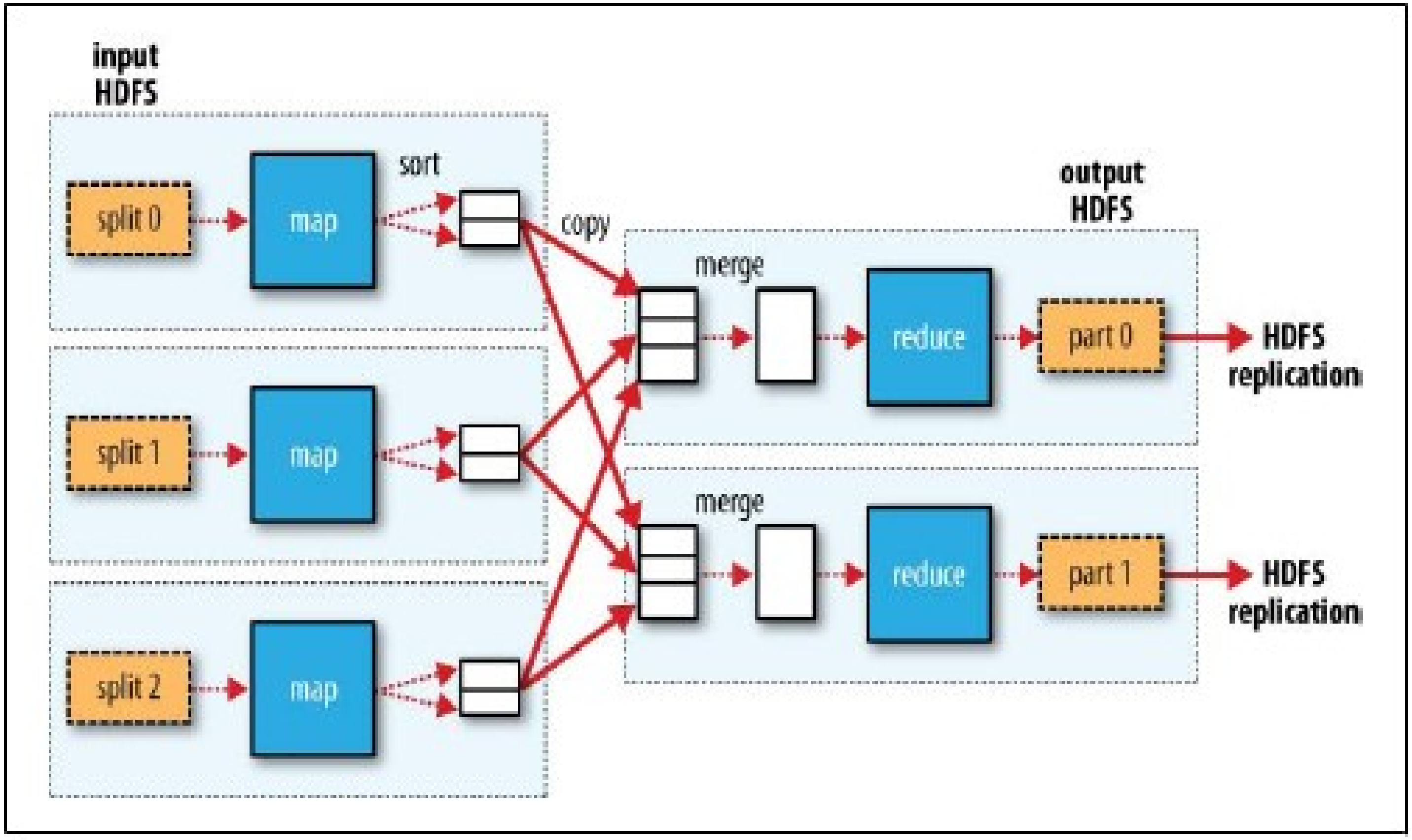


Figure 2-3. MapReduce data flow with multiple reduce tasks

MapReduce in Hadoop (3)

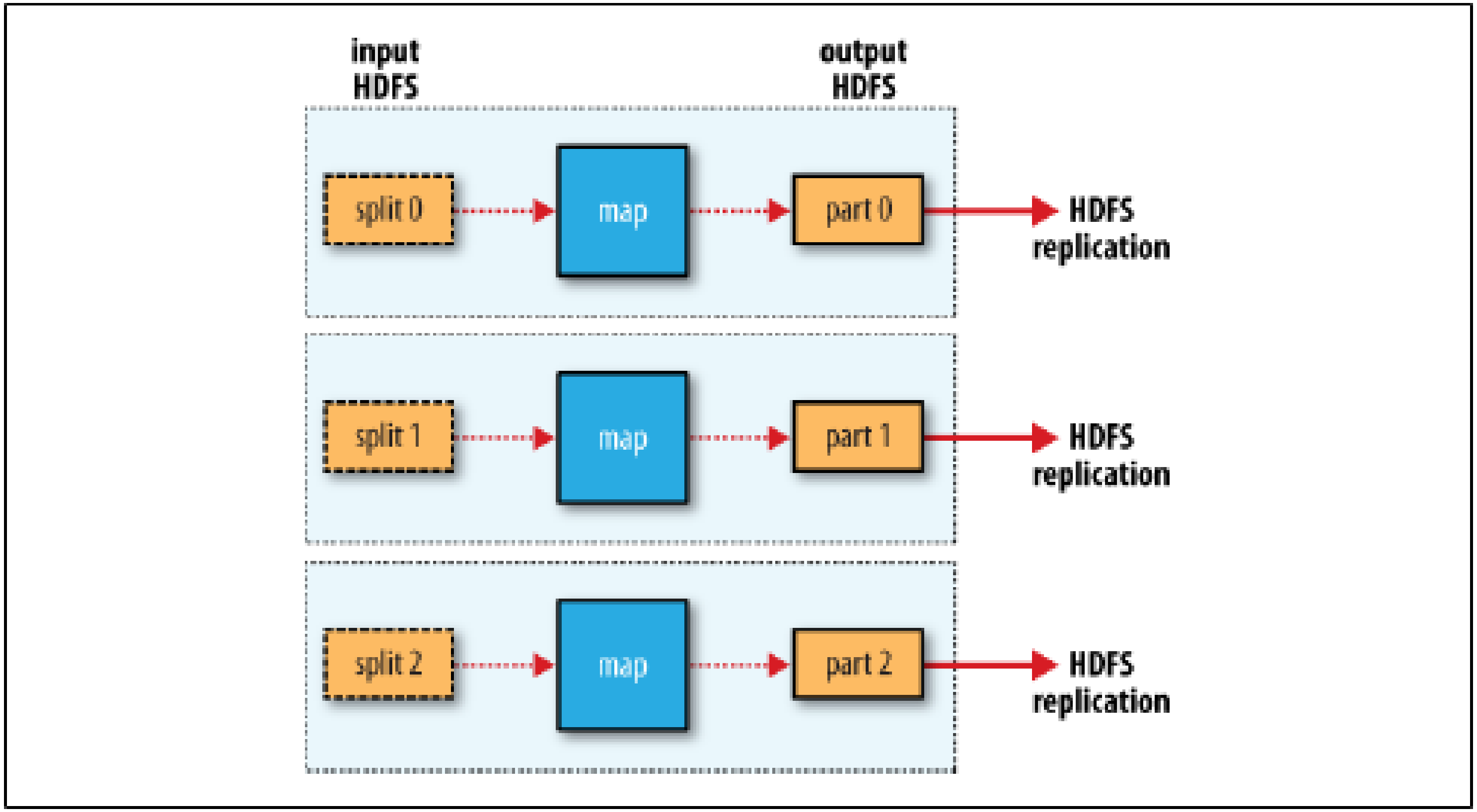


Figure 2-4. MapReduce data flow with no reduce tasks