Introduction to Data Management CSE 344

Lecture 29

Parallel Databases Wrap-up

Announcement

- Homework 8 (last) due on Friday night

- Review session:
  - Saturday, 3/16, 10am, in EEB 037

Review: Map Reduce

- How many map tasks (M) are created?

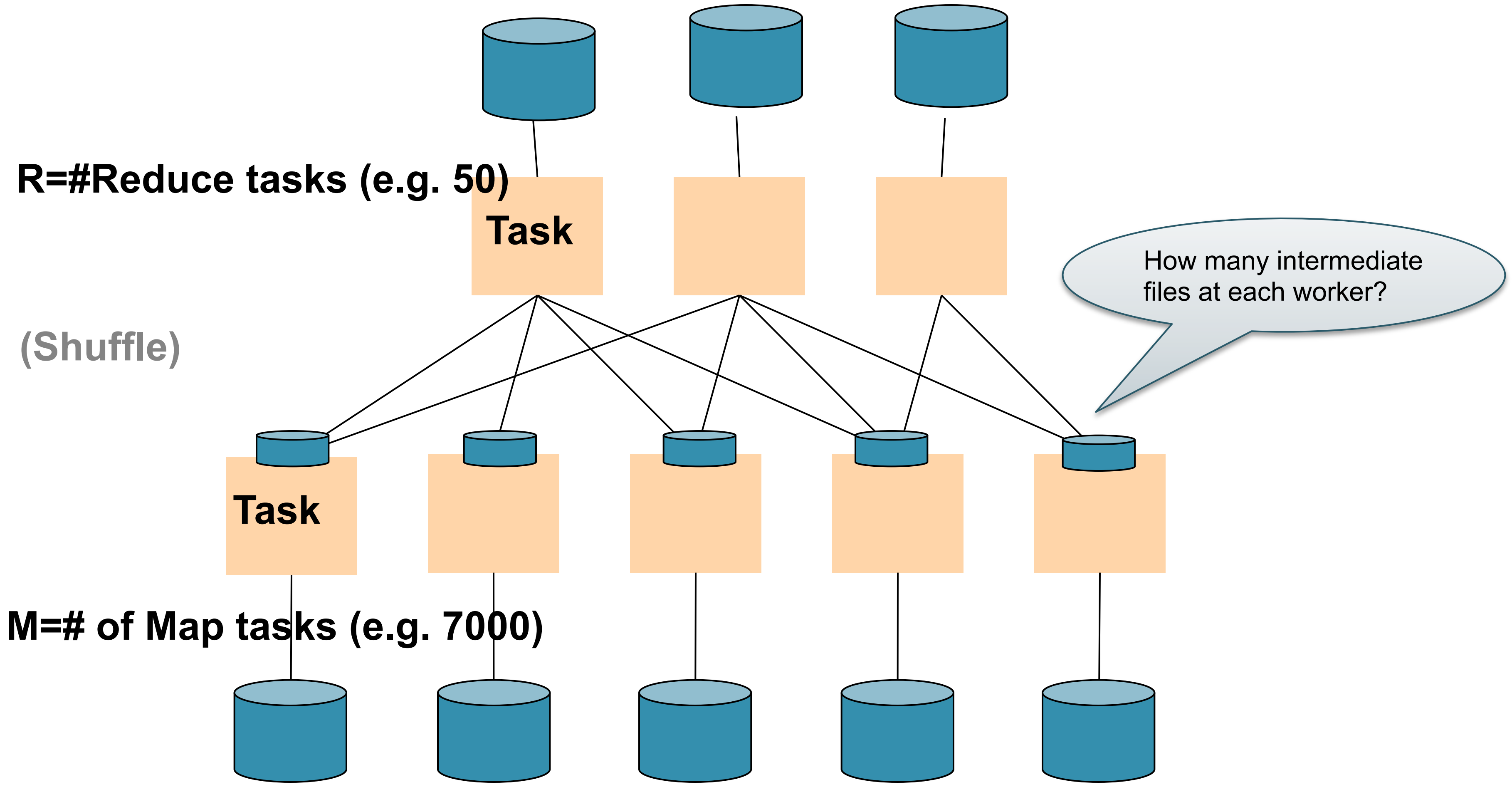
- How many reduce tasks (R) are created?

Review: Map Reduce

- How many map tasks (M) are created?
  - Default: number of chunks of input file
  - Can be configured differently…
- How many reduce tasks (R) are created?
  - Determined manually, e.g. 50
  - Why not R=1 reduce task?

  - Why not R=100000 reduce tasks?

MapReduce Execution Details


R=#Reduce tasks (e.g. 50)
Task
(Shuffle)
How many intermediate files at each worker?
Task
M=# of Map tasks (e.g. 7000)

Review: Map Reduce

- How many map tasks (M) are created?
  - – Default: number of chunks of input file
  - – Can be configured differently…
- How many reduce tasks (R) are created?
  - – Determined manually, e.g. 50
  - – Why not R=1 reduce task?
    - Will not use all workers
  - – Why not R=100000 reduce tasks?
    - Too many intermediate files to manage

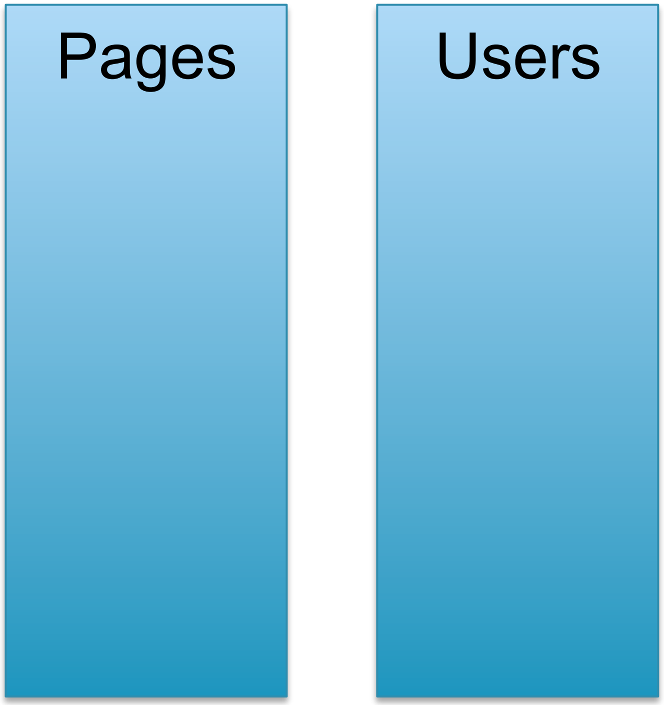
Parallel Joins in MapReduce

Reading assignment:

- Chapter 2 (Sections 1,2,3 only) of Mining of Massive Datasets, by Rajaraman and Ullman http://i.stanford.edu/~ullman/mmds.html

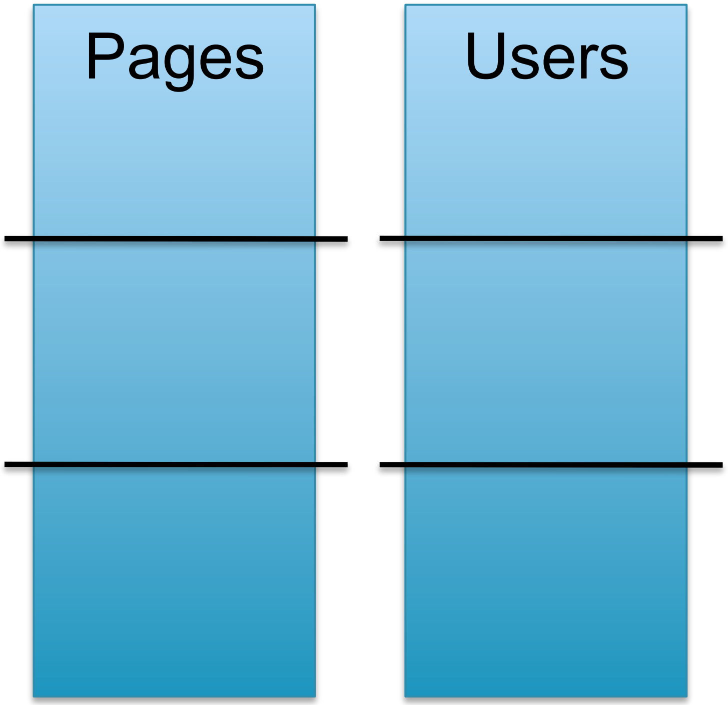
Hash Join in Pig

```
Users = load ‘users’ as (name, age);
Pages = load ‘pages’ as (user, url);
Jnd = join Users by name, Pages by user;
```



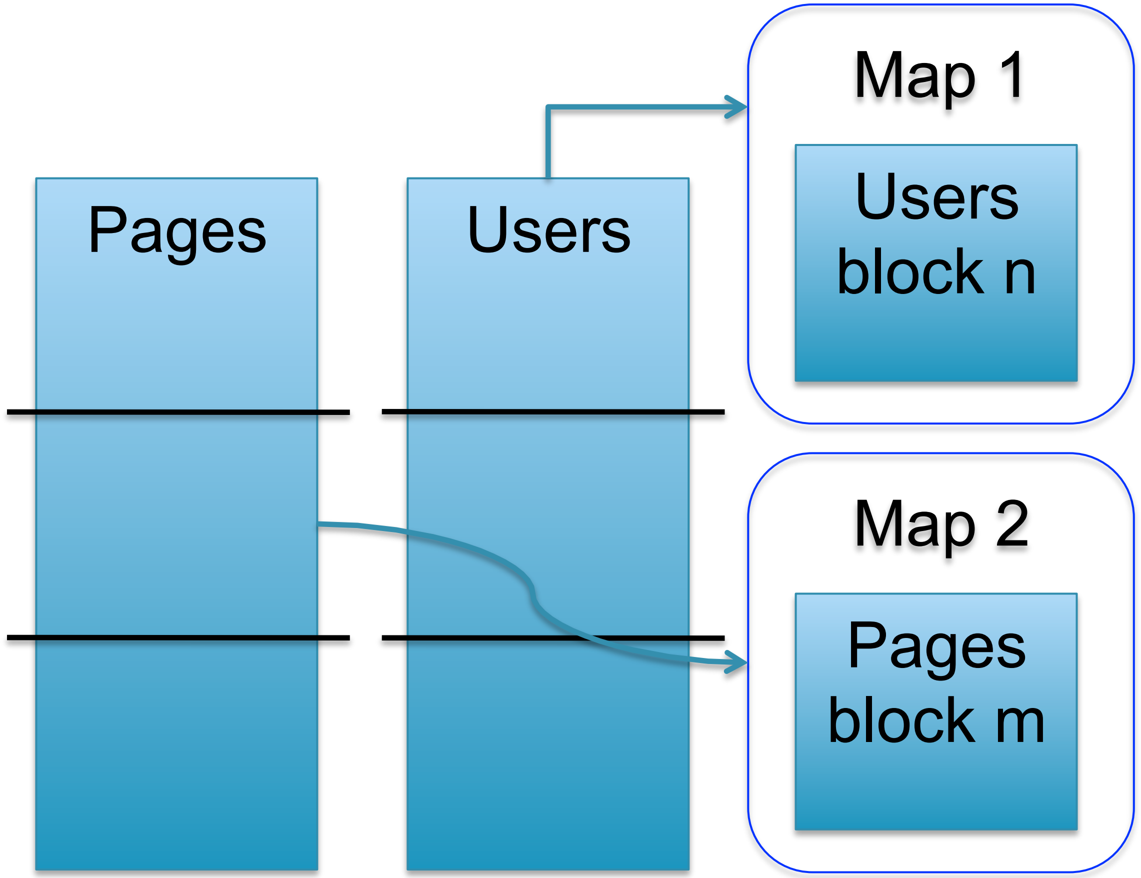
Hash Join

```
Users = load 'users' as (name, age);
Pages = load 'pages' as (user, url);
Jnd = join Users by name, Pages by user;
```



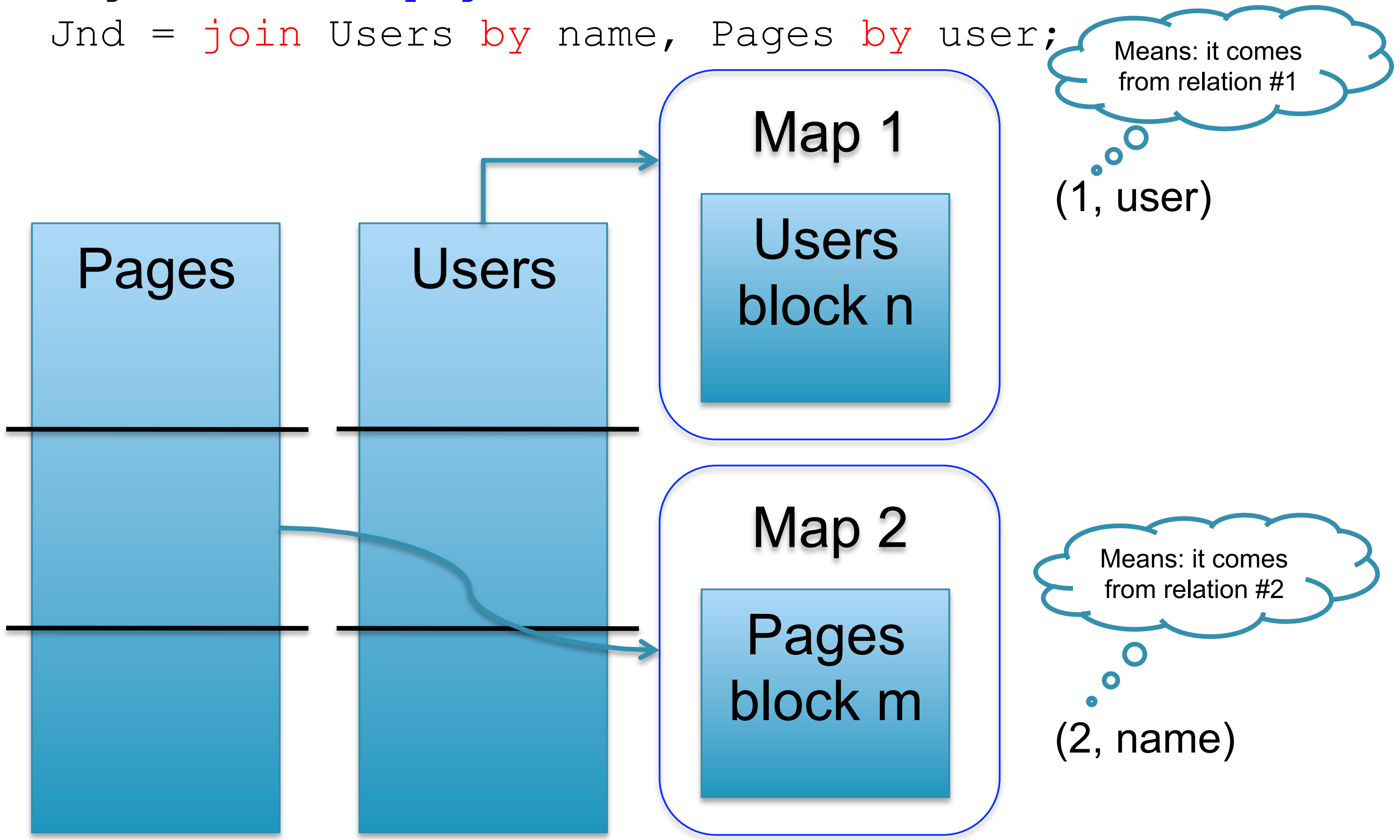
Hash Join

```
Users = load 'users' as (name, age);
Pages = load 'pages' as (user, url);
Jnd = join Users by name, Pages by user;
```



Hash Join

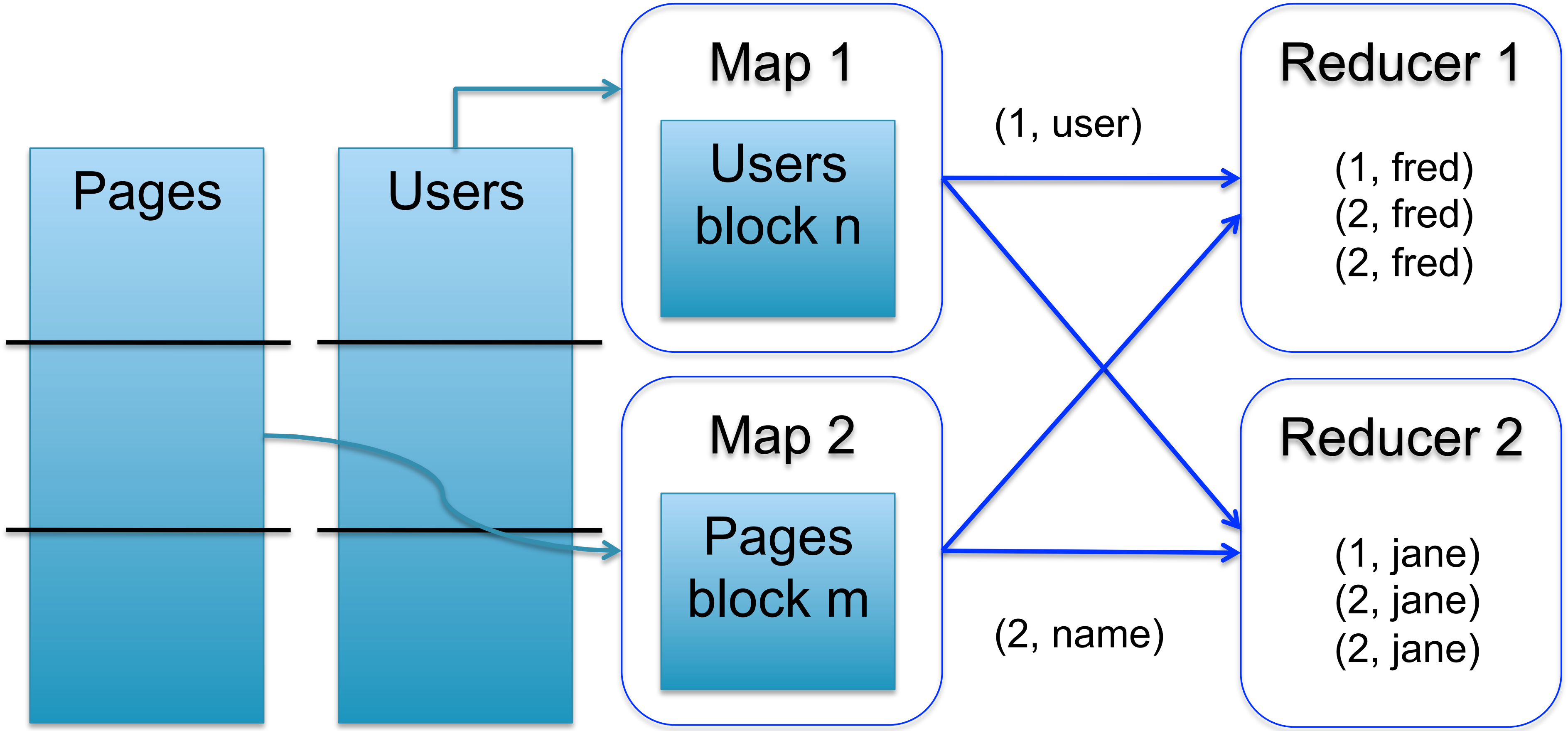
```
Users = load ‘users’ as (name, age);
Pages = load ‘pages’ as (user, url);
Jnd = join Users by name, Pages by user;
```



Means: it comes from relation #1
(1, user)
Map 1
Users block n
Pages
Users
Map 2
Pages block m
Means: it comes from relation #2
(2, name)

Hash Join

```
Users = load 'users' as (name, age);
Pages = load 'pages' as (user, url);
Jnd = join Users by name, Pages by user;
```



Hash Join

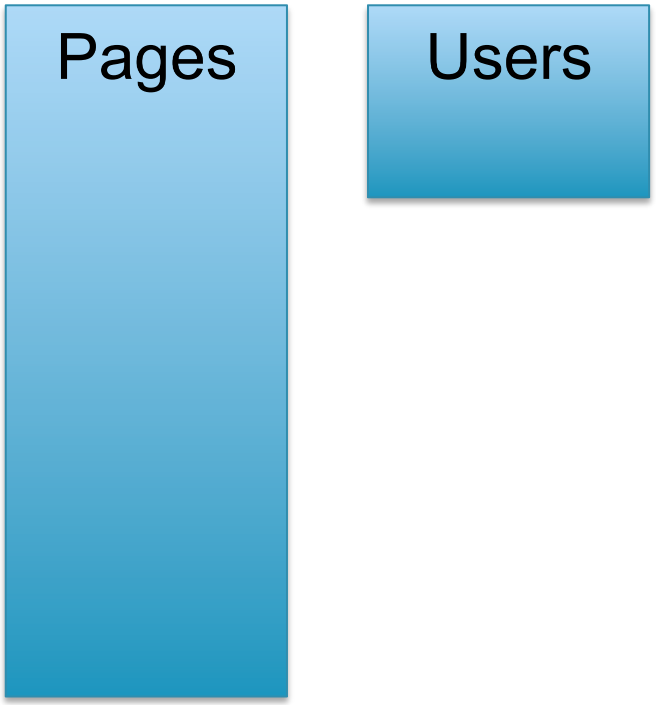
```
Users = load ‘users’ as (name, age);
Pages = load ‘pages’ as (user, url);
Jnd = join Users by name, Pages by user;
```

```
map(String usr, String value):
    // usr: either Users.name or Pages.user
    // value.relation is either ‘Users’ or ‘Pages’
    if value.relation=‘Users’:
      EmitIntermediate(usr, (1, value));
    else
      EmitIntermediate(usr, (2, value));
```

```
reduce(String usr, Iterator values):
    Users = empty;  Pages = empty;
    for each v in values:
      if v.type = 1: Users.insert(v)
      else Pages.insert(v);
    for v1 in Users, for v2 in Pages
      Emit(usr, v1,v2);
```

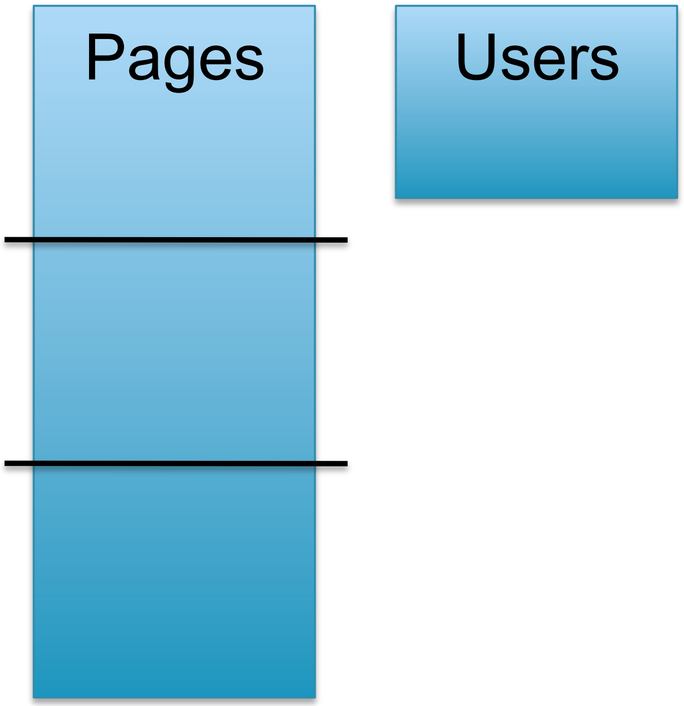
Broadcast Join

```
Users = load 'users' as (name, age);
Pages = load 'pages' as (user, url);
Jnd = join Pages by user, Users by name using "replicated";
```



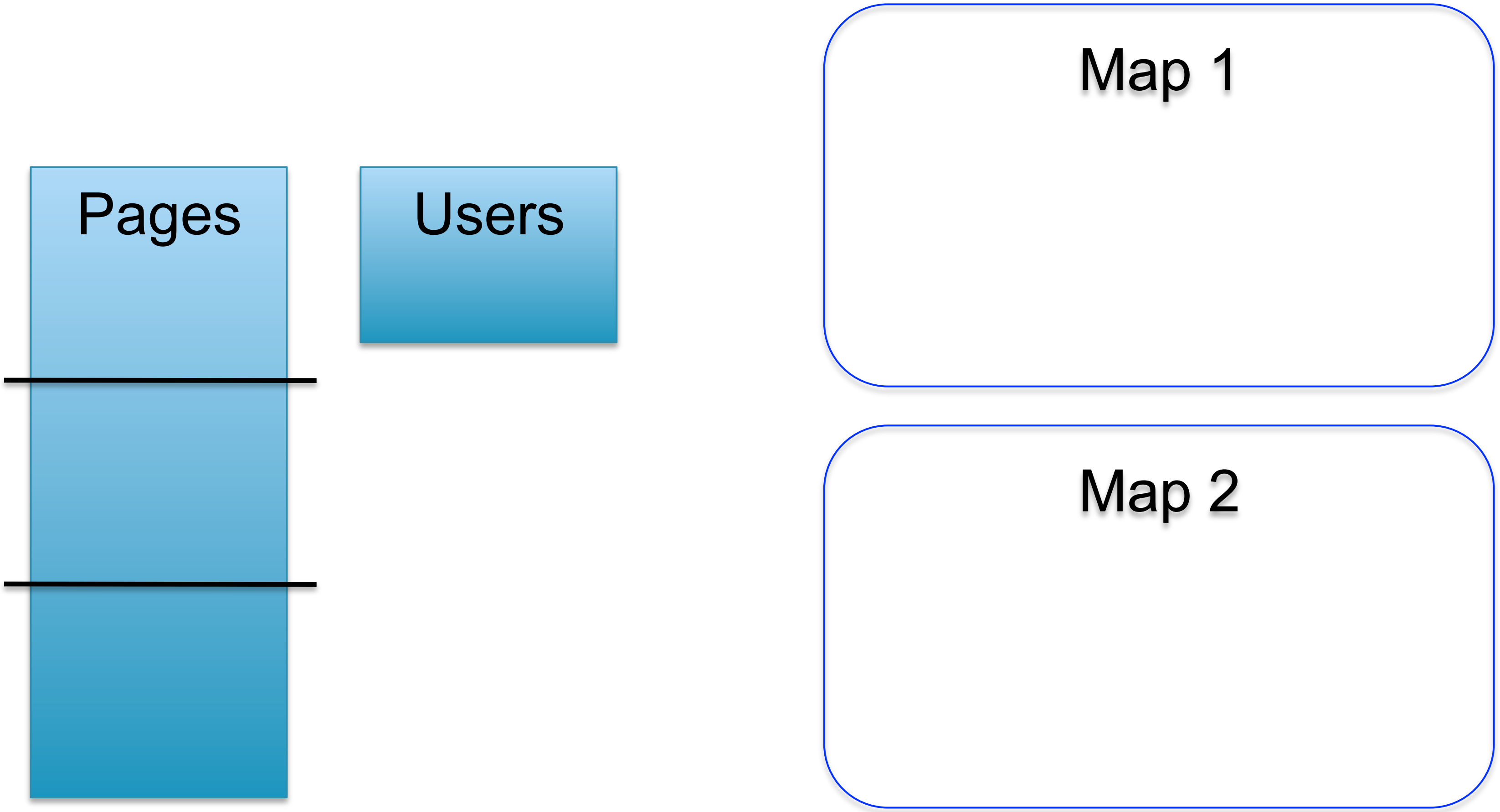
Broadcast Join

```
Users = load 'users' as (name, age);
Pages = load 'pages' as (user, url);
Jnd = join Pages by user, Users by name using "replicated";
```



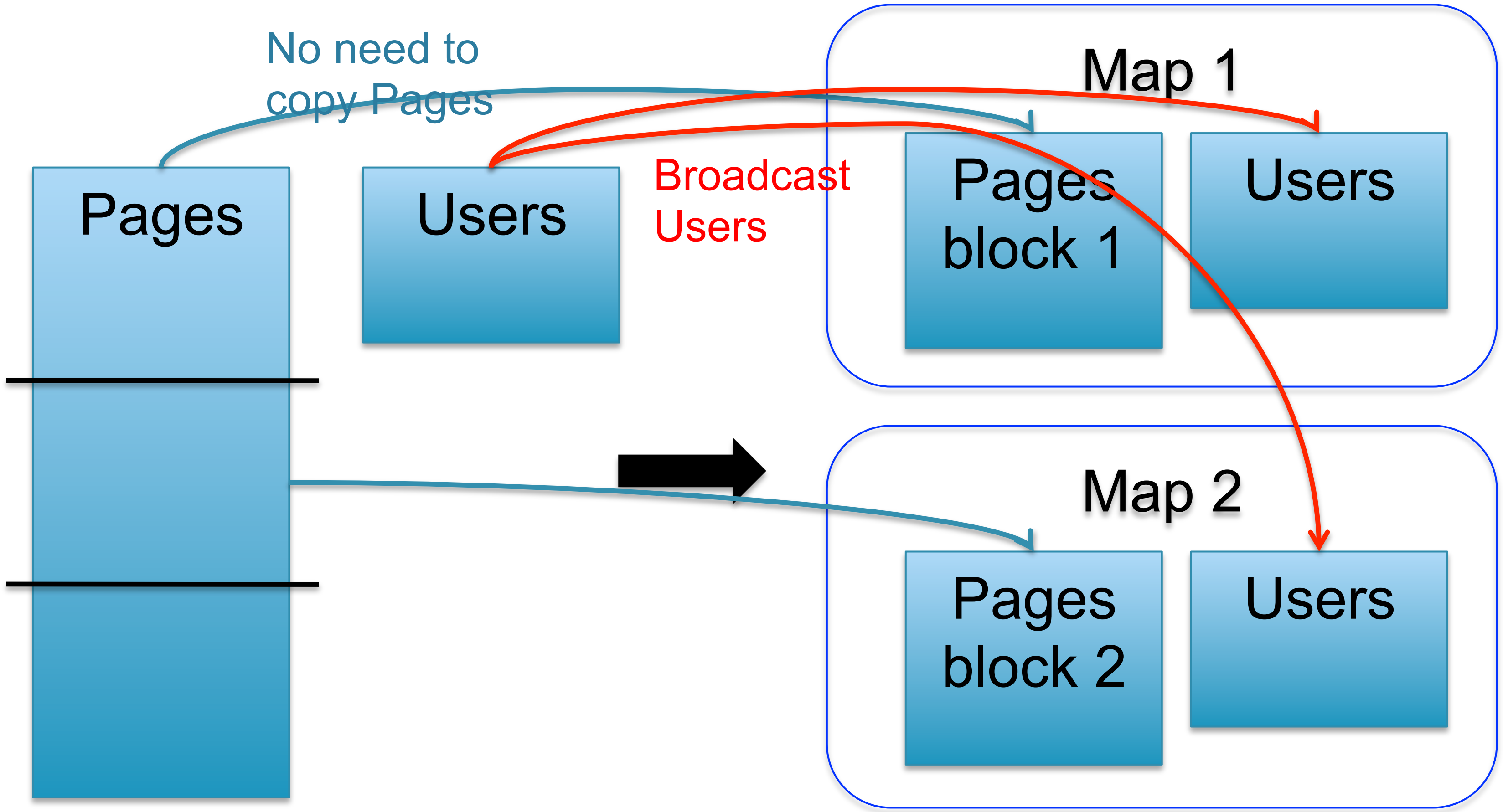
Broadcast Join

```
Users = load 'users' as (name, age);
Pages = load 'pages' as (user, url);
Jnd = join Pages by user, Users by name using "replicated";
```



Broadcast Join

```
Users = load 'users' as (name, age);
Pages = load 'pages' as (user, url);
Jnd = join Pages by user, Users by name using "replicated";
```



Matrix Multiplication v.s. Join

Dense matrices:

$$\begin{bmatrix} 6 & 6 & 0 \\ 1 & 0 & 0 \\ 2 & 0 & 6 \end{bmatrix} = \begin{bmatrix} 0 & 3 & 3 \\ 1 & 0 & 0 \\ 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 3 \\ 0 & 2 & 0 \\ 2 & 0 & 0 \end{bmatrix}$$

```
forall i,k do
    $C[i,k] = \Sigma_j A[i,j] * B[j,k]$
```

Matrix Multiplication v.s. Join

Dense matrices:

$$\begin{bmatrix} 6 & 6 & 0 \\ 1 & 0 & 0 \\ 2 & 0 & 6 \end{bmatrix} = \begin{bmatrix} 0 & 3 & 3 \\ 1 & 0 & 0 \\ 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 3 \\ 0 & 2 & 0 \\ 2 & 0 & 0 \end{bmatrix}$$

```
forall i,k do
    C[i,k] = Σ_j A[i,j] * B[j,k]
```

Sparse matrices as relations:

B(j,k,v)

j	k	v
1	1	1
1	3	3
2	2	1
3	1	2

A(i,j,v)

i	j	v
1	2	3
1	3	3
2	1	1
3	1	2

```
SELECT A.i, B.k, sum(A.v*B.v)
FROM A, B
WHERE A.j=B.j
GROUP BY A.i,B.i
```

Matrix Multiplication v.s. Join

Dense matrices:

$$\begin{bmatrix} 6 & 6 & 0 \\ 1 & 0 & 0 \\ 2 & 0 & 6 \end{bmatrix} = \begin{bmatrix} 0 & 3 & 3 \\ 1 & 0 & 0 \\ 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 3 \\ 0 & 2 & 0 \\ 2 & 0 & 0 \end{bmatrix}$$

```
forall i,k do
    C[i,k] = Σ_j A[i,j] * B[j,k]
```

Sparse matrices as relations:

B(j,k,v)

j	k	v
1	1	1
1	3	3
2	2	1
3	1	2

A(i,j,v)

i	j	v
1	2	3
1	3	3
2	1	1
3	1	2

```
SELECT A.i, B.k, sum(A.v*B.v)
FROM A, B
WHERE A.j=B.j
GROUP BY A.i,B.i
```

Matrix multiplication = a join + a group by

Parallel DBs v.s. MapReduce

Parallel DB
- Plusses

- Minuses

MapReduce
- Minuses

- Plusses

Parallel DBs v.s. MapReduce

Parallel DB

- Plusses
  - Efficient binary format
  - Indexes, physical tuning
  - Cost-based optimization
- Minuses
  - Difficult to import data
  - Lots of baggage: logging, transactions

MapReduce

- Minuses
  - Lots of time spent parsing!
  - Text files
  - "Optimizers is between your eyes and your keyboard"
- Plusses
  - Any data
  - Lightweight, easy to speedup
  - Arguably more scalable