# COGROUP: Getting data together

C = COGROUP A BY f1, B BY $0;

A =
<1, 2, 3>
<4, 2, 1>
<8, 3, 4>
<4, 3, 3>
<7, 2, 5>
<8, 4, 3>

B =
<2, 4>
<8, 9>
<1, 3>
<2, 7>
<2, 9>
<4, 6>
<4, 9>

C =
<1, {<1, 2, 3>}, {<1, 3>}>
<2, {}, {<2, 4>, <2, 7>, <2, 9>}>
<4, {<4, 2, 1>, <4, 3, 3>}, {<4, 6>,<4, 9>}>
<7, {<7, 2, 5>}, {}>
<8, {<8, 3, 4>, <8, 4, 3>}, {<8, 9>}>

# JOIN: A special case of COGROUP

C = JOIN A BY $0, B BY $0;

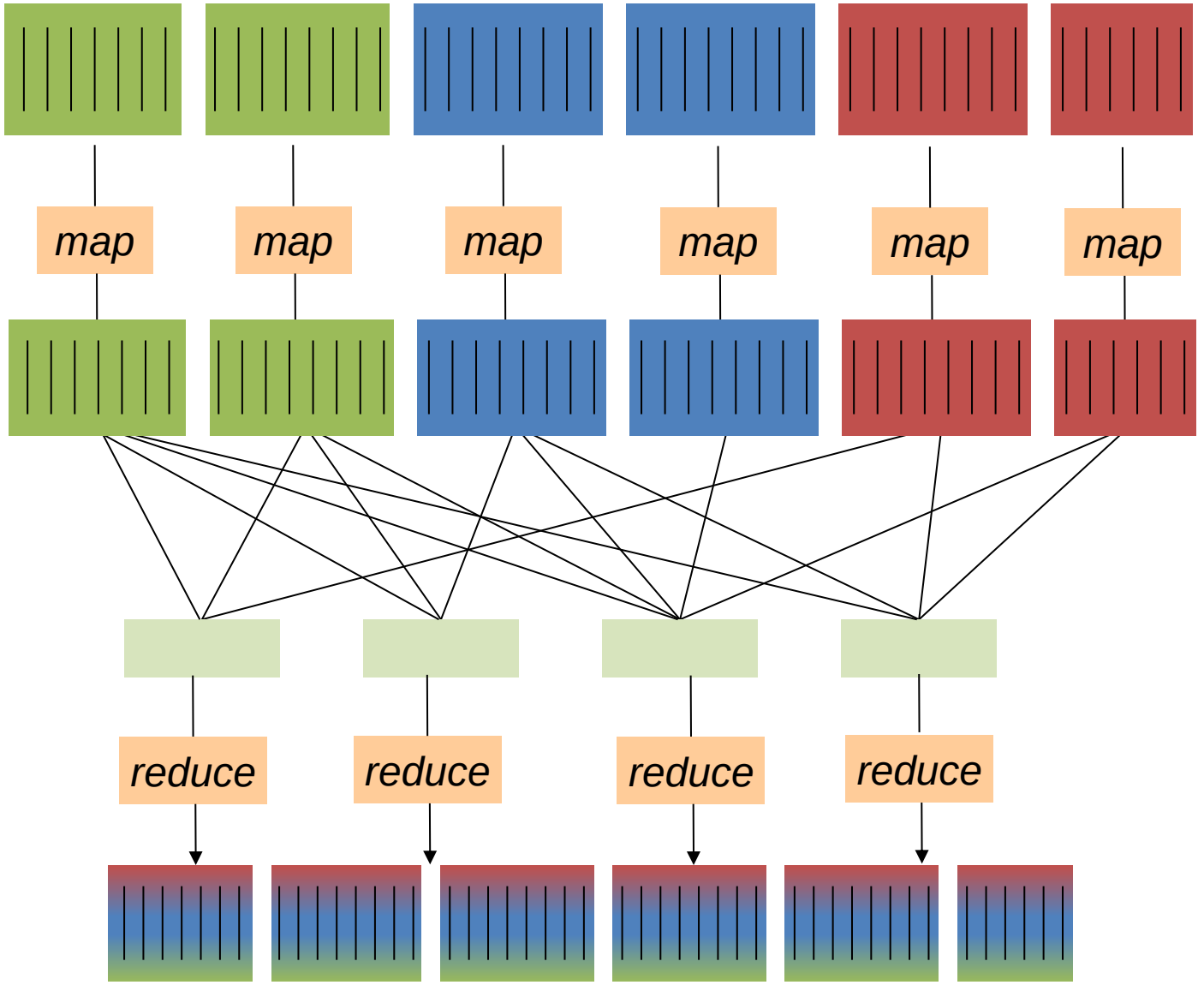
A =
<1, 2, 3>
<4, 2, 1>
<8, 3, 4>
<4, 3, 3>
<7, 2, 5>
<8, 4, 3>

B =
<2, 4>
<8, 9>
<1, 3>
<2, 7>
<2, 9>
<4, 6>
<4, 9>

C =
<1, 2, 3, 1, 3>
<4, 2, 1, 4, 6>
<4, 2, 1, 4, 9>
<4, 3, 3, 4, 6>
<4, 3, 3, 4, 9>
<8, 3, 4, 8, 9>
<8, 4, 3, 8, 9>

- COGROUP and JOIN can both on multiple datasets
  - Think about why this works

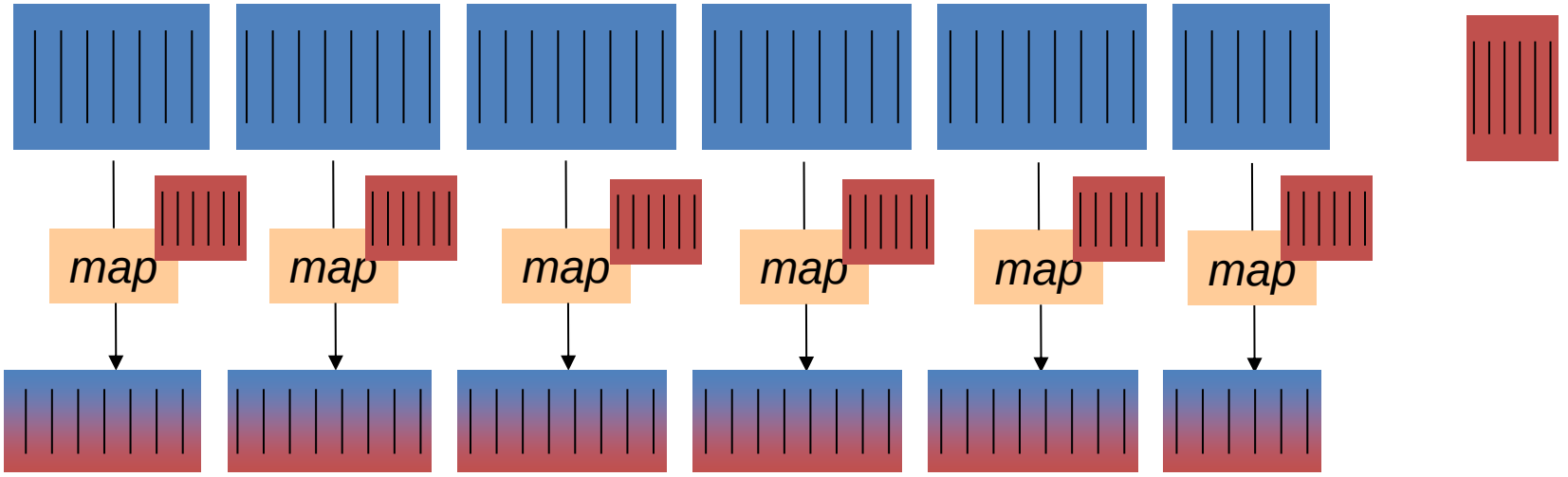
# Join multiple relations



# Three Special Join Algorithms

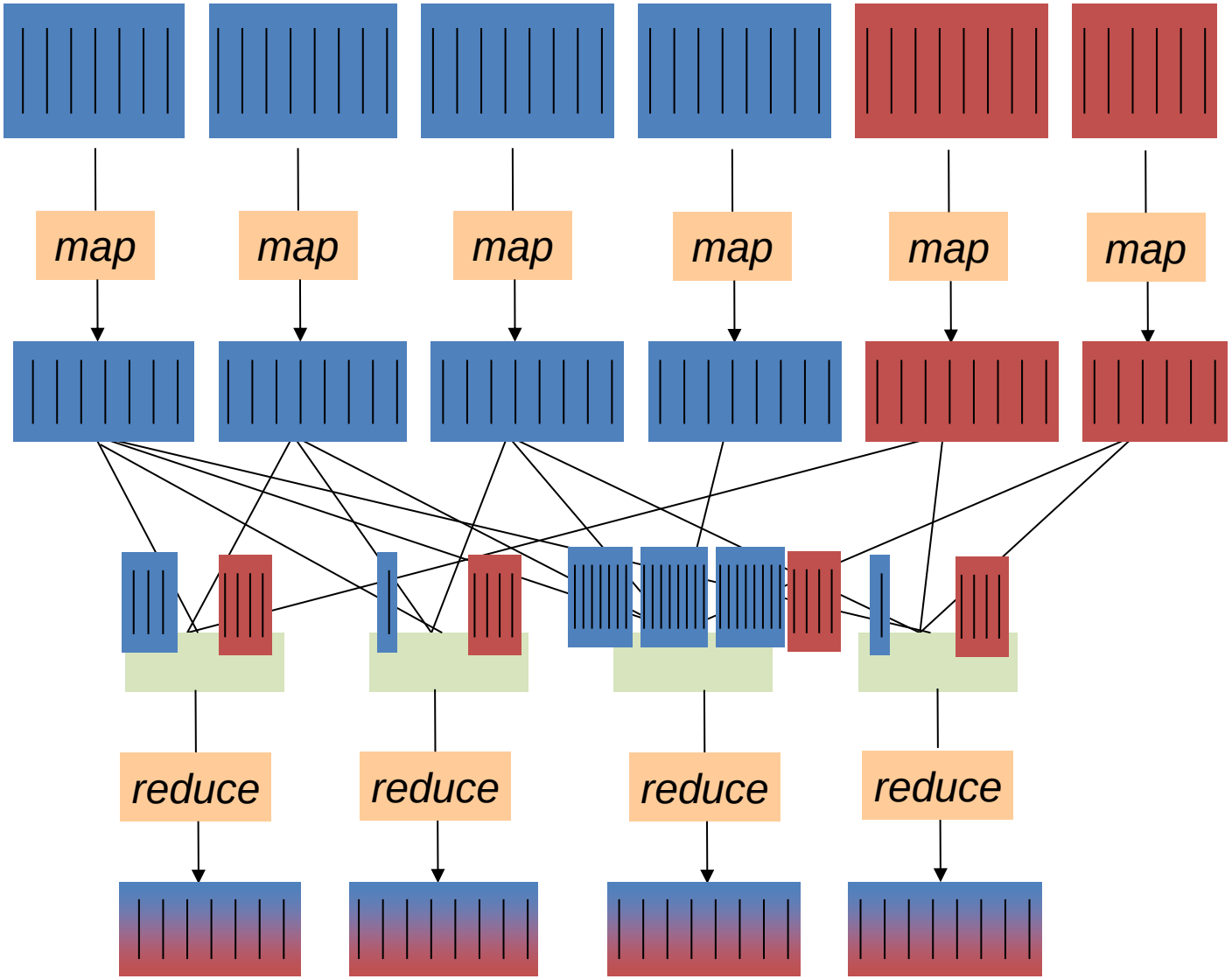
- Replicated
  - One table is very small, one is big
  - Replicate the small one
- Skewed
  - When one join key is associated with most of the data, we're in trouble
  - handle these cases differently
- Merge
  - If the two datasets are already grouped/sorted on the correct attribute, we can compute the join in the Map phase

# Replicated Join



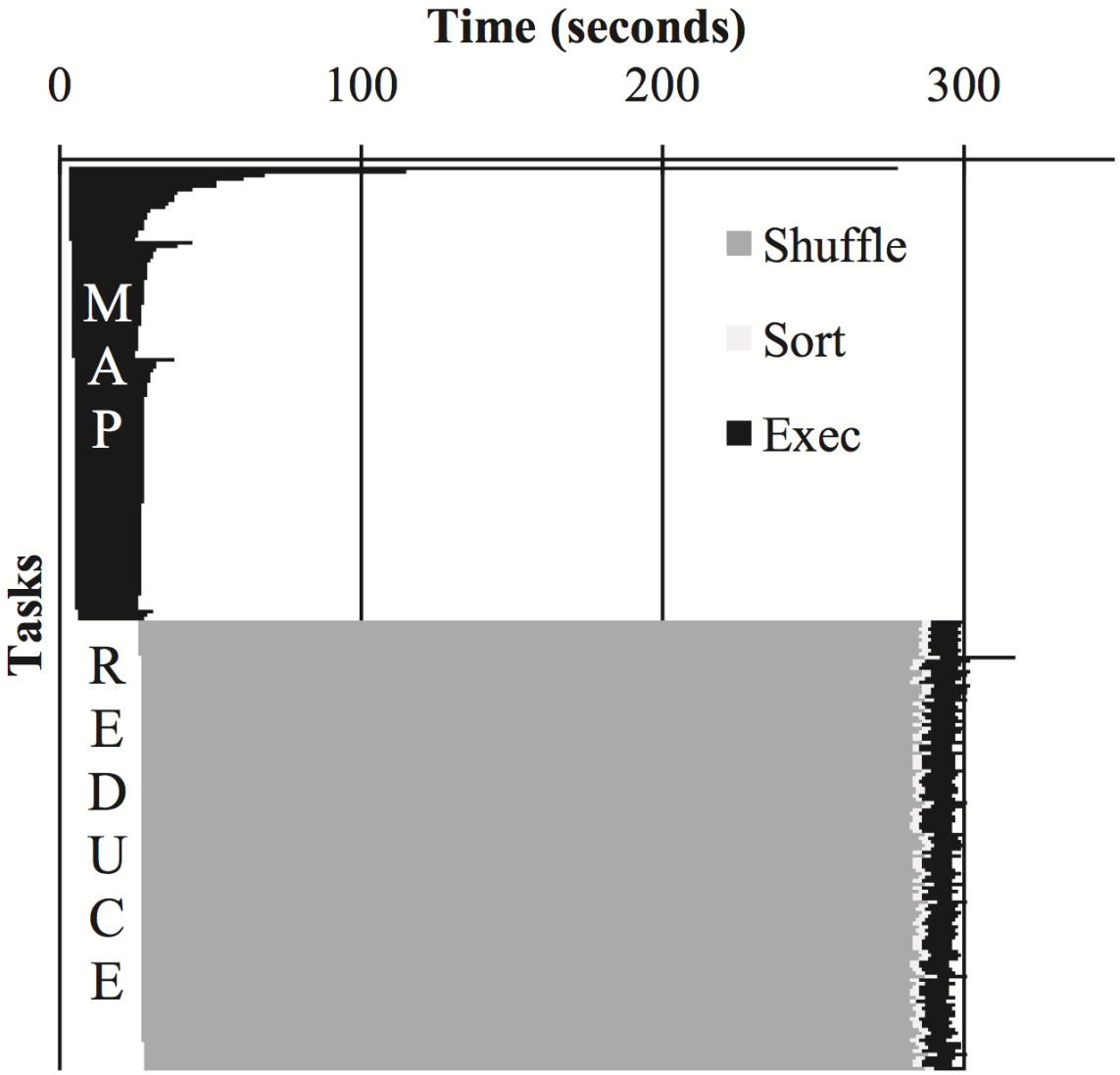

- Each mapper pulls a copy of the small relation from HDFS
- Small relation must fit in main memory
- You'll see this called "Broadcast Join" as well

# Skew Join



Kwon 2012

# The Problem of Skew

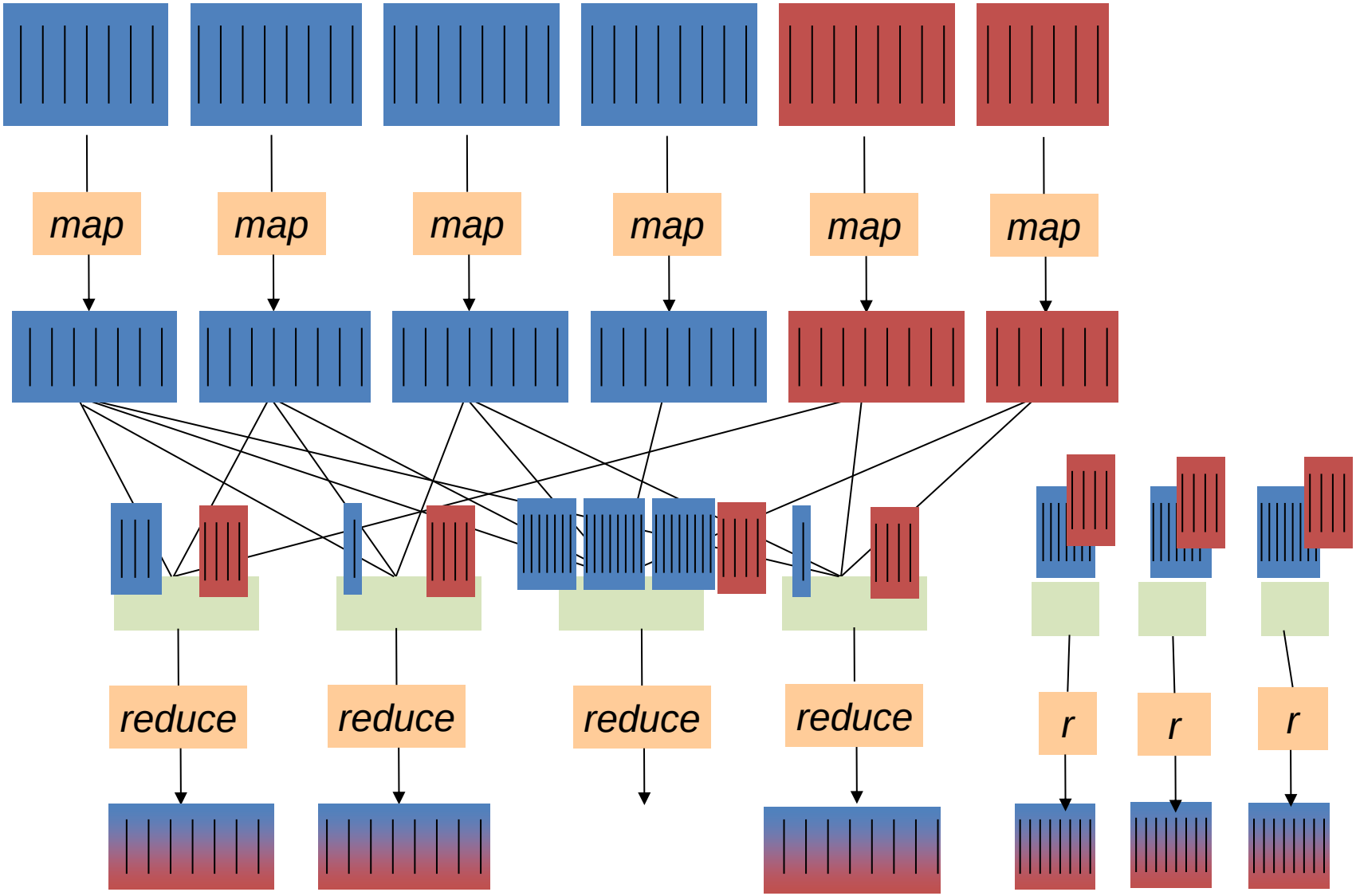



One task takes 5X longer than the average! Very little benefit to parallelism

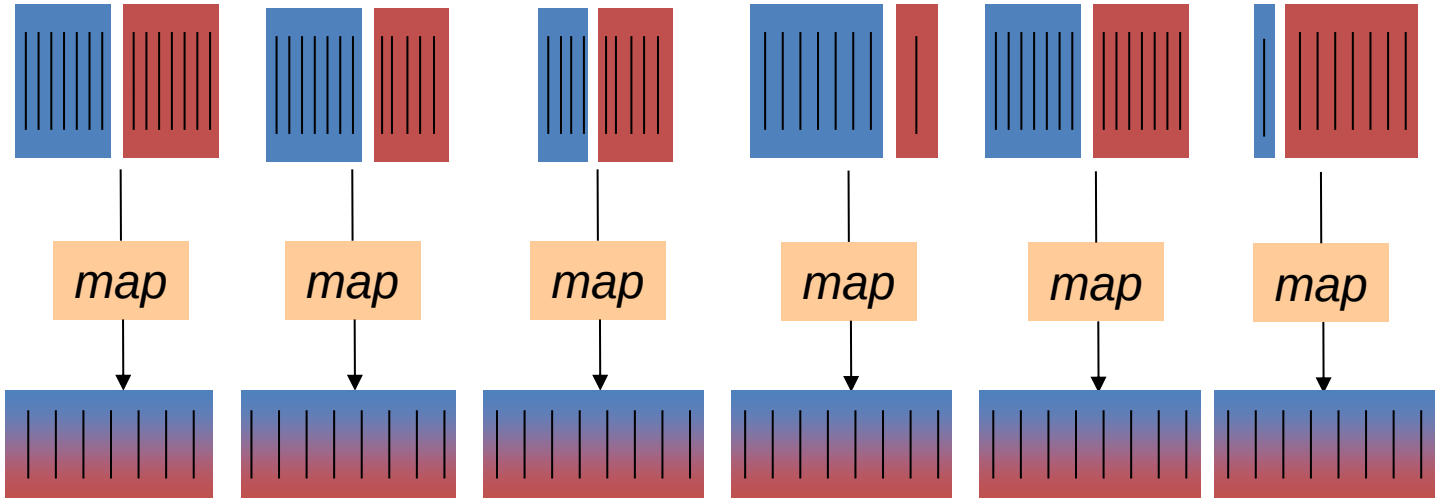
If someone asks you "What is the biggest performance problem with MapReduce in practice?"

Say: "Stragglers!"

# Skew Join



# Merge Join




- Each mapper already has local access to the records from both relations, grouped and sorted by the join key.
- Just read in both relations from disk, in order.

# Why COGROUP and not JOIN?

- JOIN is a two-step process
  - Create groups with shared keys
  - Produce joined tuples
- COGROUP only performs the first step
  - You might do different processing on the groups
  - e.g., count the tuples

```
join_result = JOIN A BY $0, B BY $0;

temp = COGROUP A BY $0, B BY $0;

join_result = FOREACH temp GENERATE
                    FLATTEN(results), FLATTEN(revenue);
```

# Other Commands

- STORE
- UNION
- CROSS
- DUMP
- ORDER

```
STORE bagName INTO 'myoutput.txt'
    USING some_func();
```