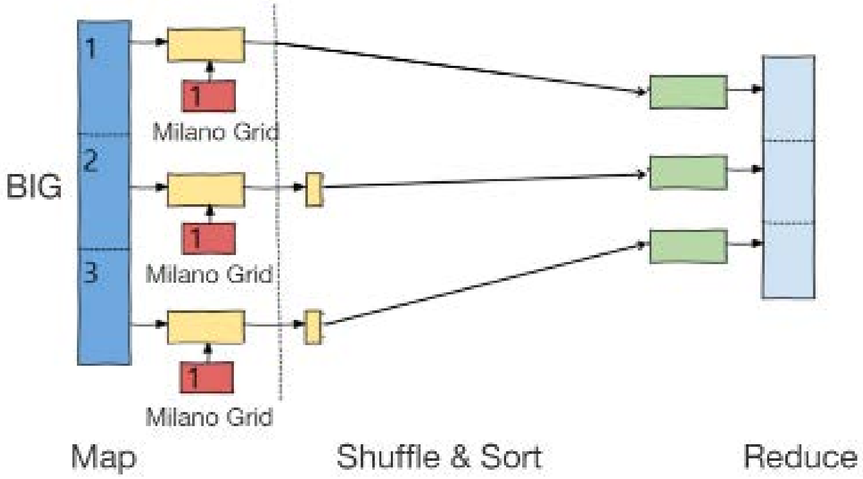
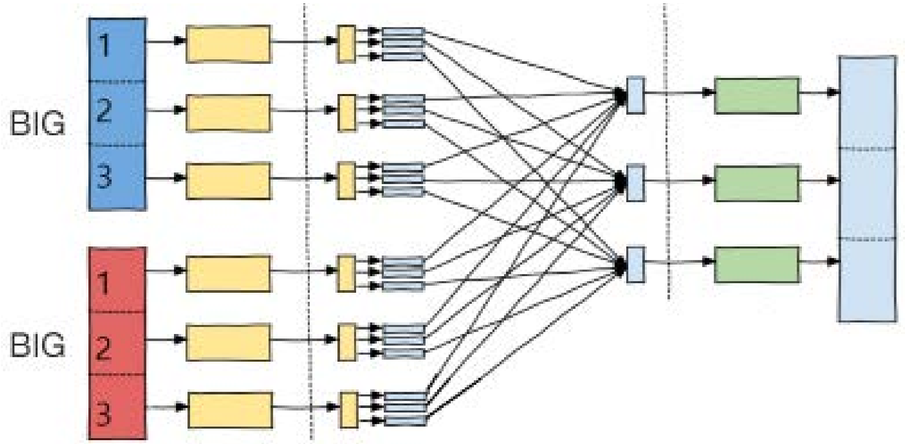
Hive Optimization

Joins

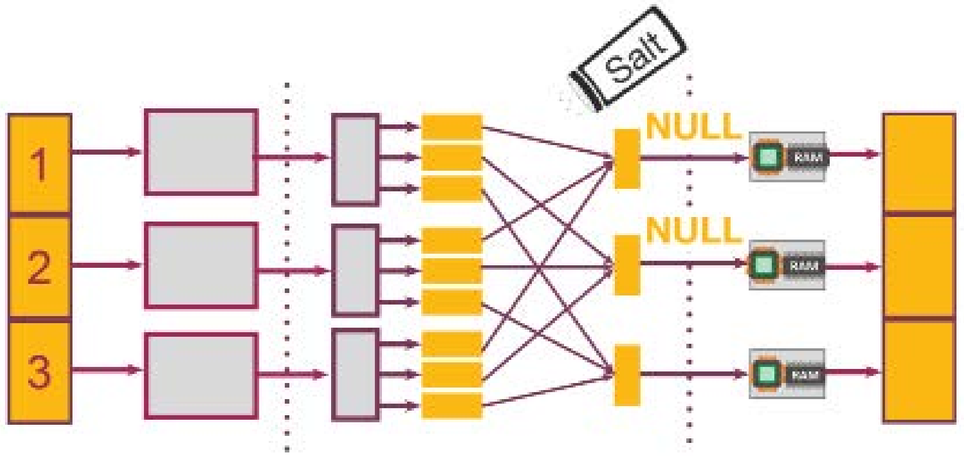
MapSide Join



Reduce side Join

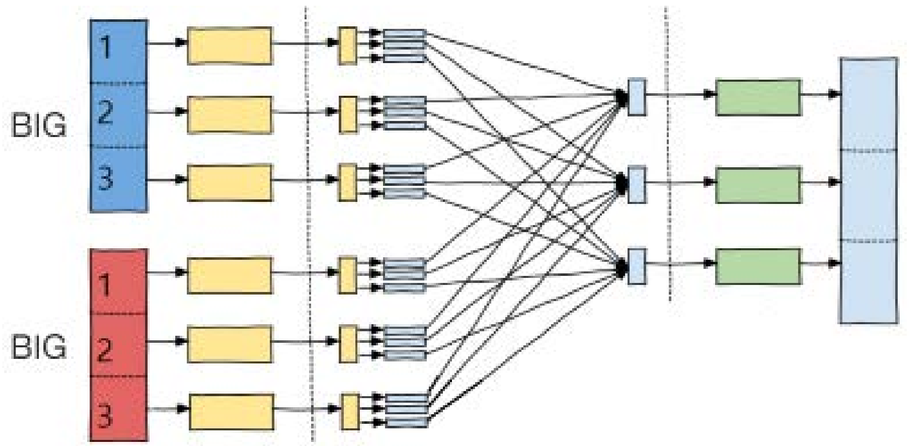


Data Skew

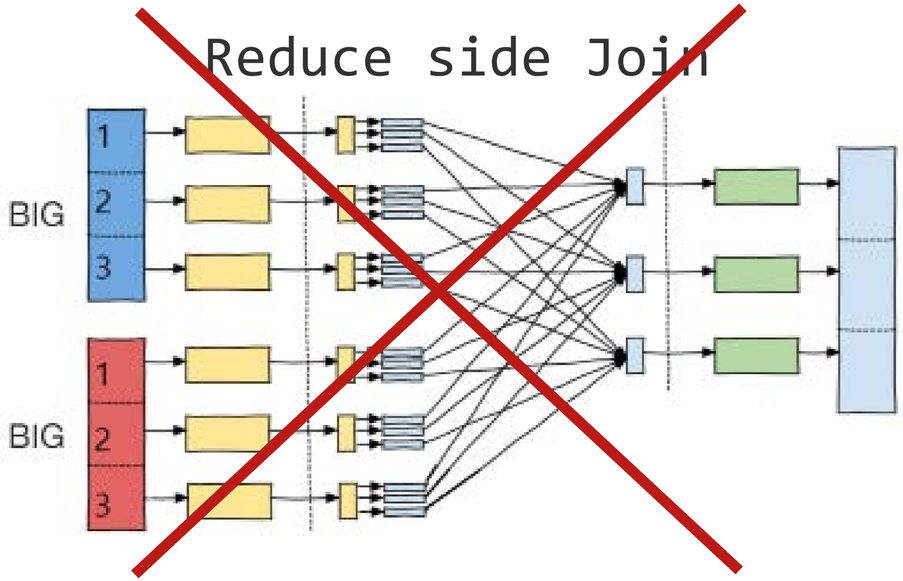


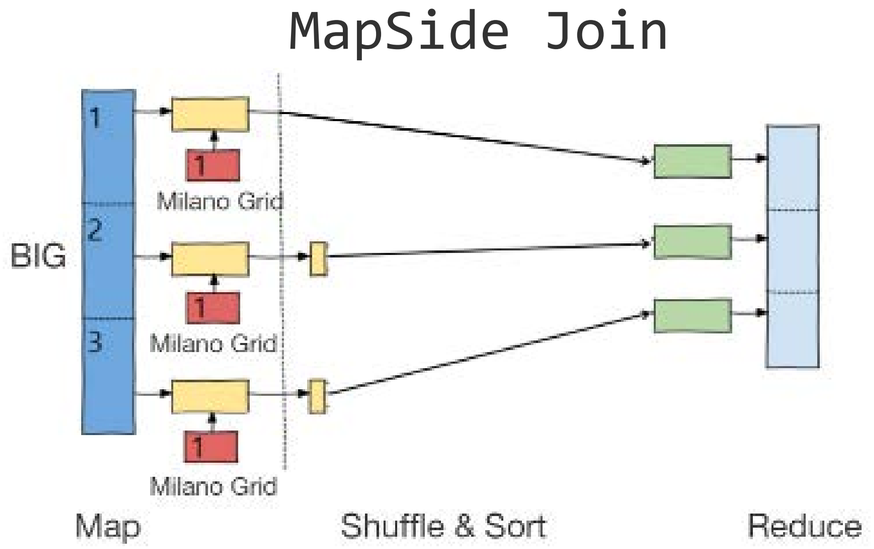
```
SELECT region_city, COUNT(1) AS hit_count
FROM access_log JOIN geo_base
ON (access_log.host = geo_base.host)
GROUP BY region_city ORDER BY hit_count LIMIT 100
```

Reduce side Join



```
SELECT region_city, COUNT(1) AS hit_count
FROM access_log JOIN geo_base
ON (access_log.host = geo_base.host)
GROUP BY region_city ORDER BY hit_count LIMIT 100
```



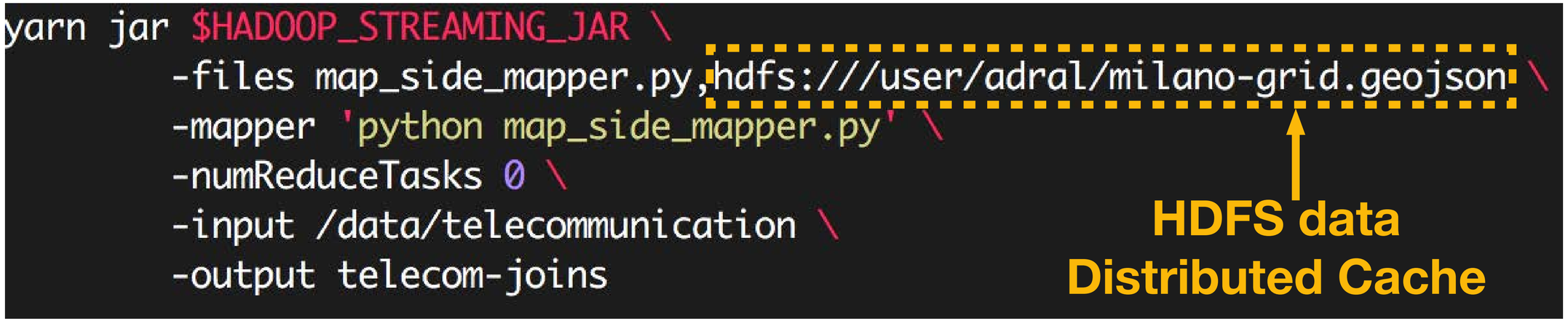



```
yarn jar $HADOOP_STREAMING_JAR \
        -files map_side_mapper.py,hdfs:///user/adral/milano-grid.geojson \
        -mapper 'python map_side_mapper.py' \
        -numReduceTasks 0 \
        -input /data/telecommunication \
        -output telecom-joins
```

HDFS data
Distributed Cache


```
yarn jar $HADOOP_STREAMING_JAR \
        -files map_side_mapper.py,hdfs:///user/adral/milano-grid.geojson \
        -mapper 'python map_side_mapper.py' \
        -numReduceTasks 0 \
        -input /data/telecommunication \
        -output telecom-joins
```

HDFS data Distributed Cache



1. client node: download small table <— HDFS


```
yarn jar $HADOOP_STREAMING_JAR \
        -files map_side_mapper.py,hdfs:///user/adral/milano-grid.geojson \
        -mapper 'python map_side_mapper.py' \
        -numReduceTasks 0 \
        -input /data/telecommunication \
        -output telecom-joins
```

HDFS data Distributed Cache

1. client node: download small table <— HDFS
2. build hashtable


```
yarn jar $HADOOP_STREAMING_JAR \
        -files map_side_mapper.py,hdfs:///user/adral/milano-grid.geojson \
        -mapper 'python map_side_mapper.py' \
        -numReduceTasks 0 \
        -input /data/telecommunication \
        -output telecom-joins
```

HDFS data Distributed Cache

1. client node: download small table <— HDFS
2. build hashtable
3. upload hashtable —> Distributed Cache


BIG moderate SMALL

```
CREATE TABLE access_log (
    …
)
CLUSTERED BY (ip)
    INTO 128 BUCKETS
…;
```

```
CREATE TABLE geo_base (
    …
)
CLUSTERED BY (ip)
    INTO 128 BUCKETS
…;
```

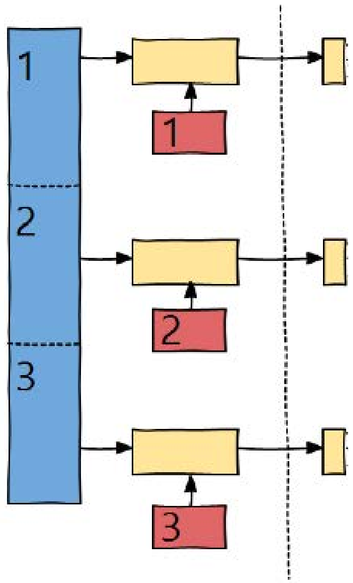
```
CREATE TABLE access_log (
    …
)
CLUSTERED BY (ip)
    INTO 128 BUCKETS
…;
```

```
CREATE TABLE geo_base (
    …
)
CLUSTERED BY (ip)
    INTO 128 BUCKETS
…;
```

```
CREATE TABLE access_log (
    …
)
CLUSTERED BY (ip)
    INTO 128 BUCKETS
…;
```

```
CREATE TABLE geo_base (
    …
)
CLUSTERED BY (ip)
    INTO 128 BUCKETS
…;
```

Bucket Join



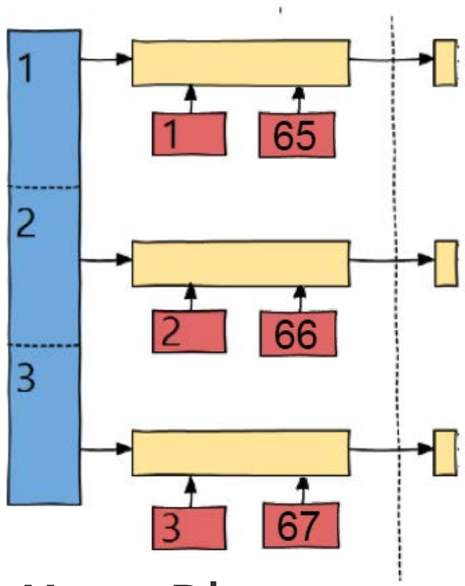
Map Phase

```
CREATE TABLE access_log (
    …
)
CLUSTERED BY (ip)
    INTO 128 BUCKETS
…;
```

```
CREATE TABLE geo_base (
    …
)
CLUSTERED BY (ip)
    INTO 128 BUCKETS
…;
```

Bucket Join

`hash(ip) % 64 == 1`



`hash(ip) % 128 == 1,65`

Map Phase

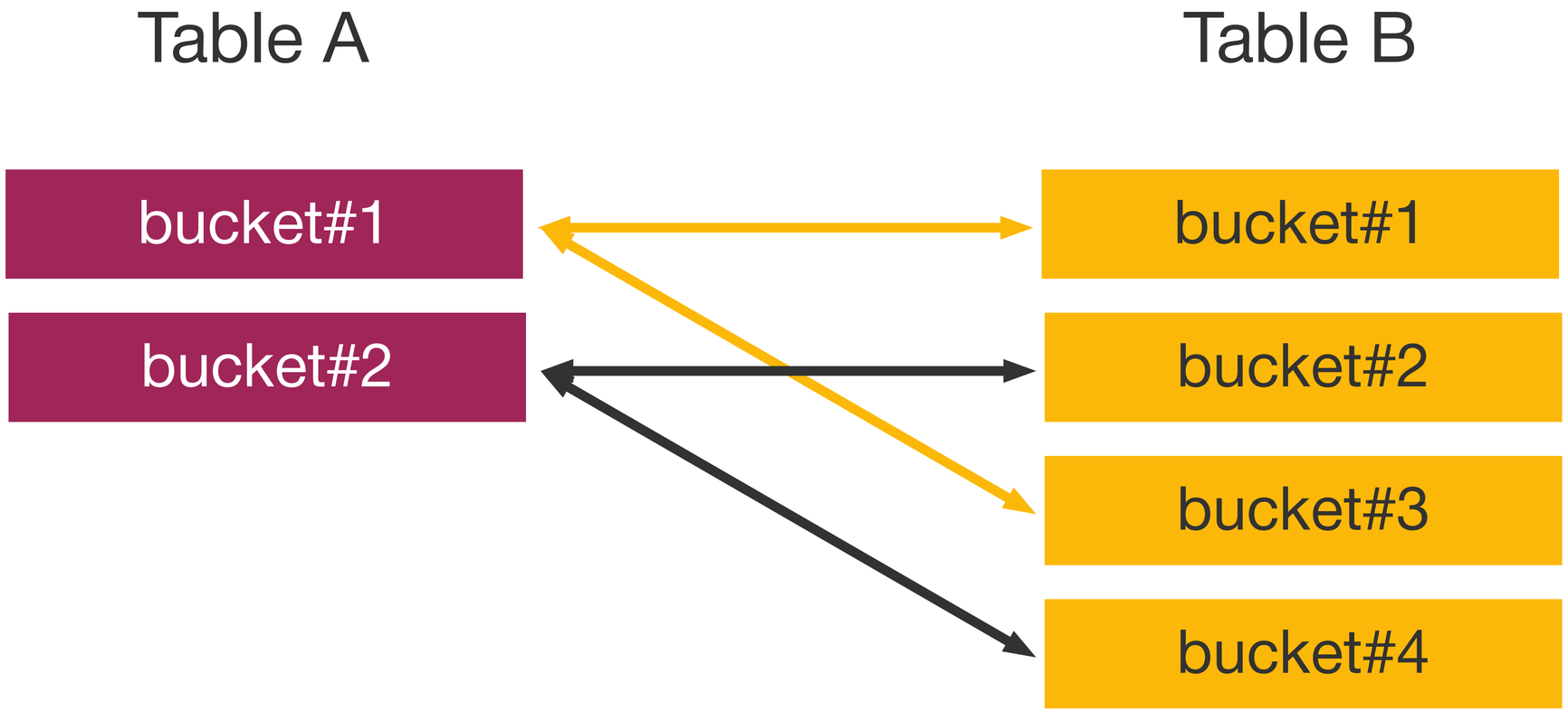


Table A
Table B
bucket#1
bucket#2
bucket#1
bucket#2
bucket#3
bucket#4

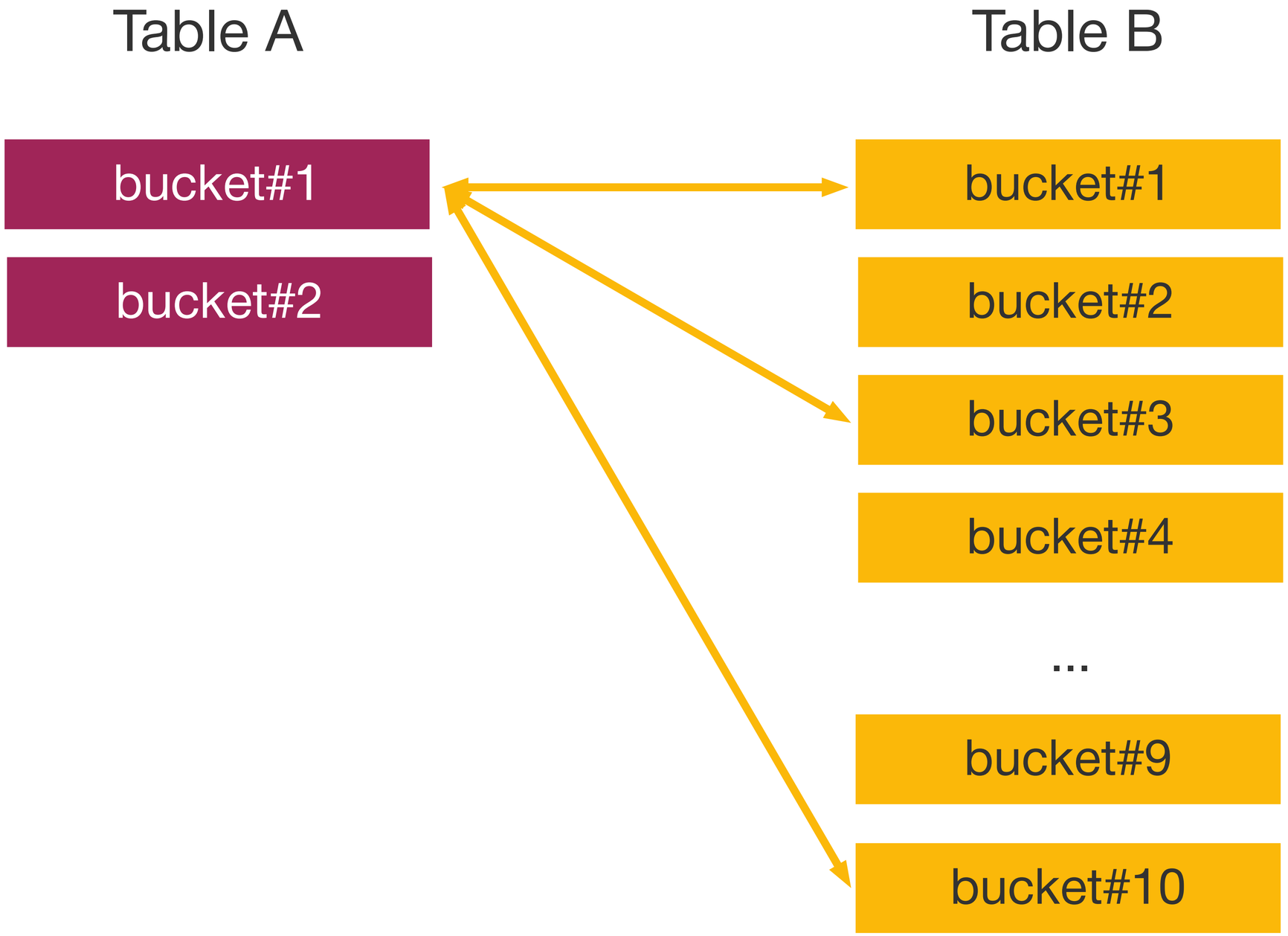


Table A
Table B
bucket#1
bucket#2
bucket#1
bucket#2
bucket#3
bucket#4
...
bucket#9
bucket#10

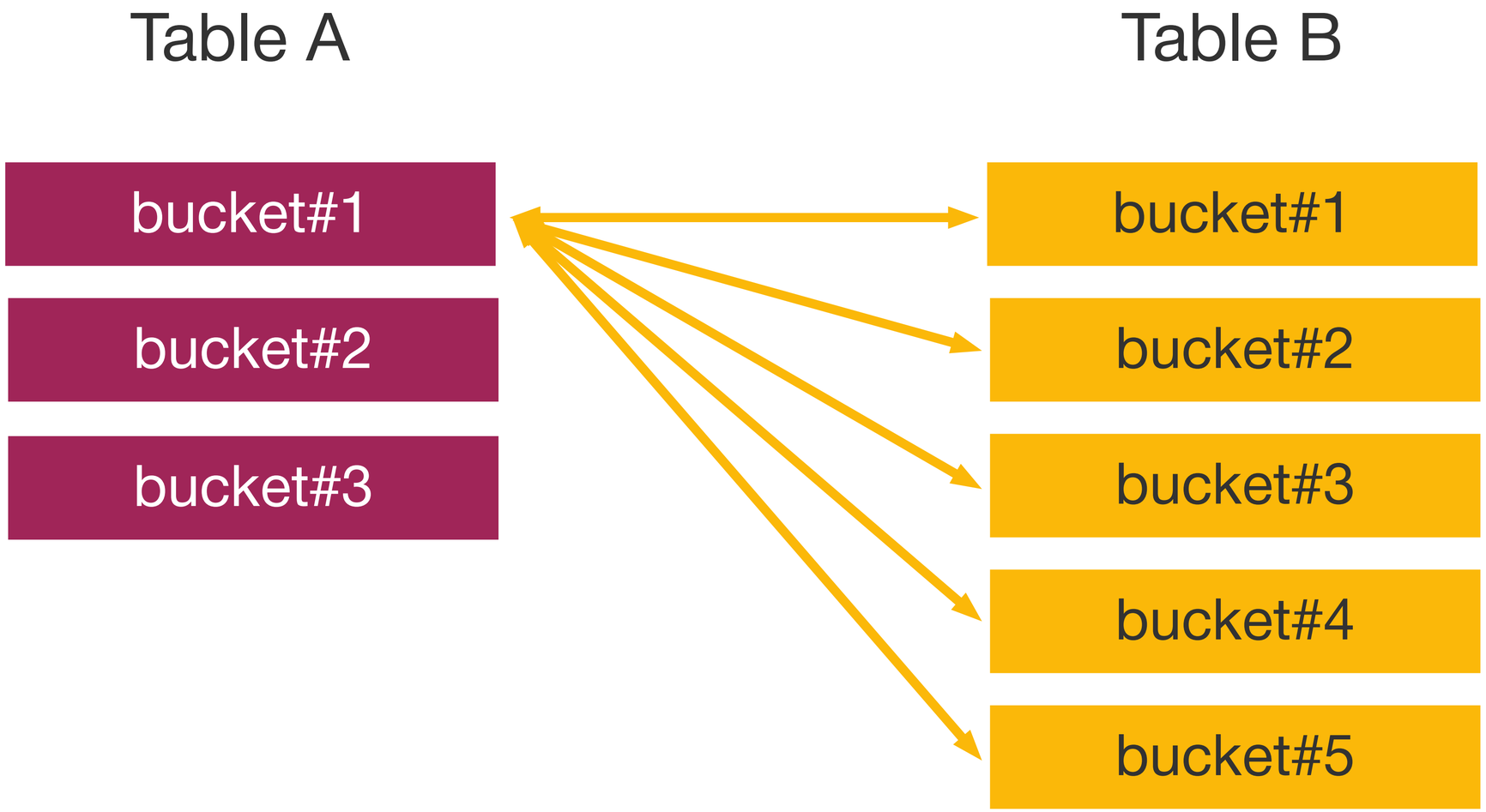
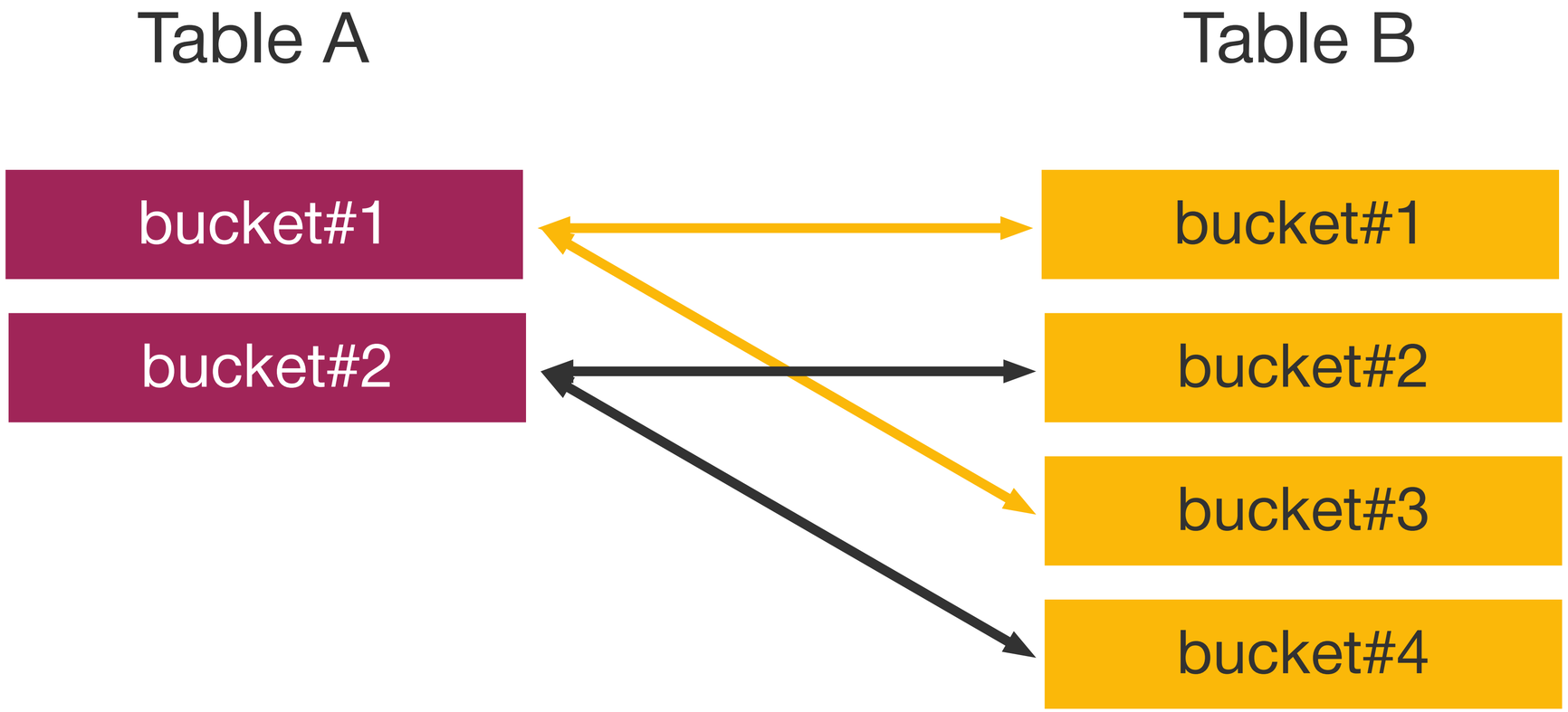
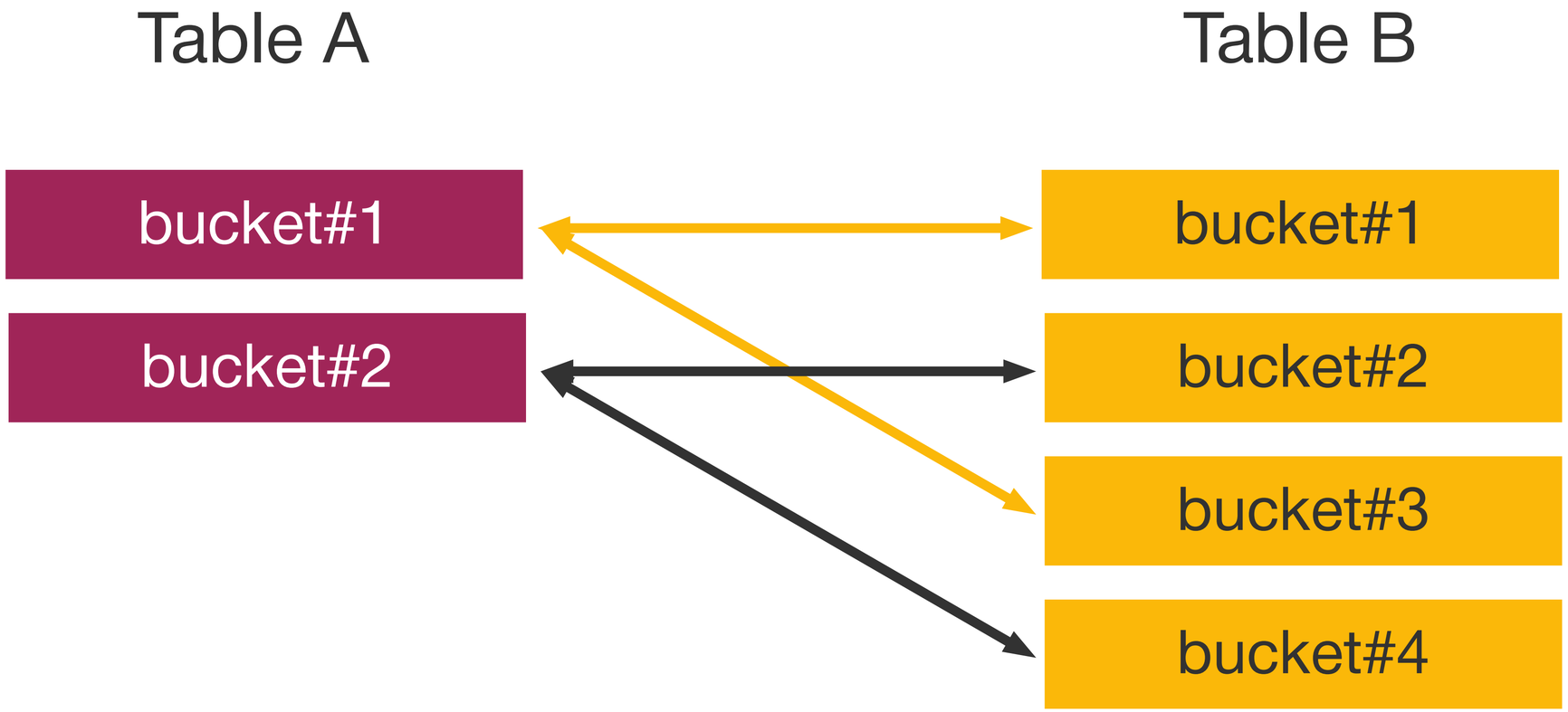


Table A
Table B
bucket#1
bucket#2
bucket#3
bucket#1
bucket#2
bucket#3
bucket#4
bucket#5

Tip: use powers of 2

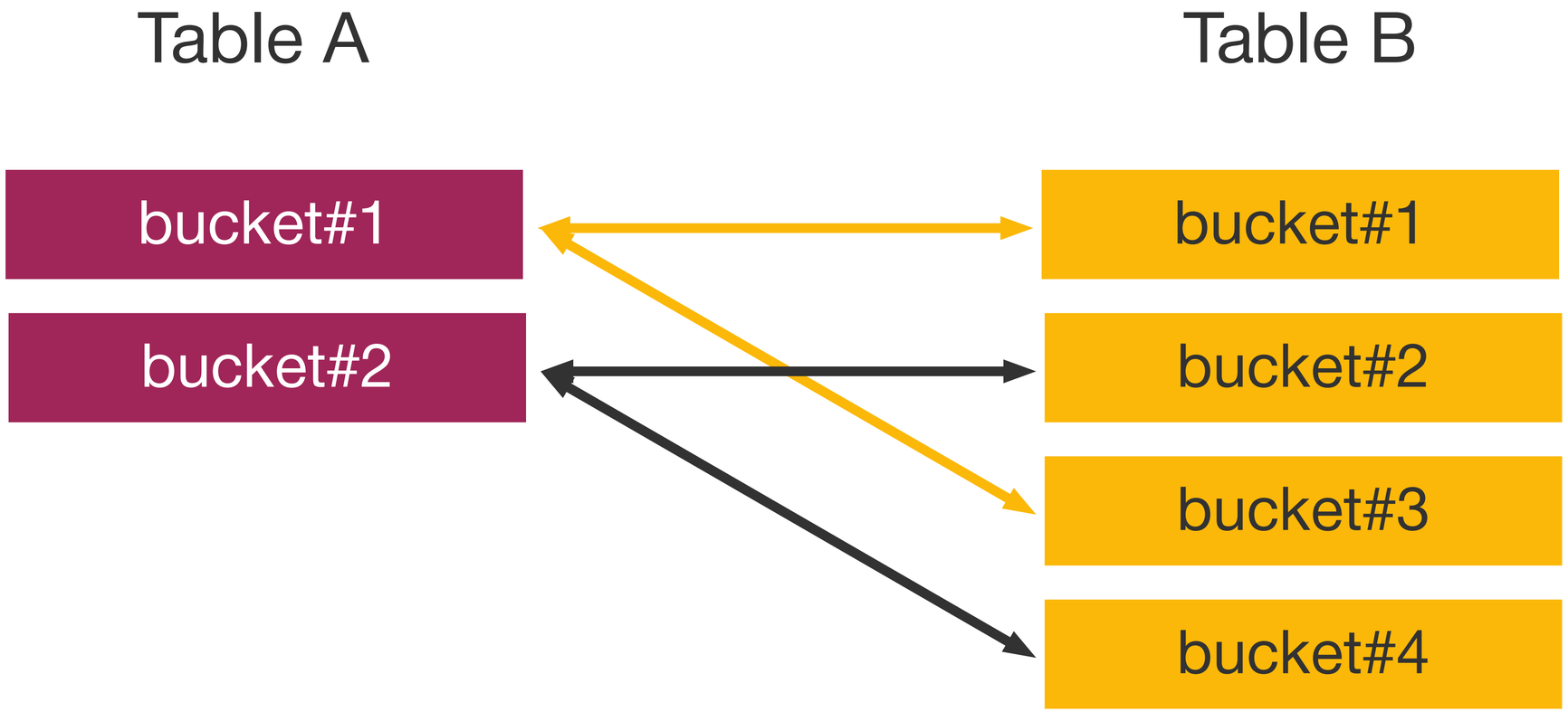


Tip: use powers of 2



1. MapSide: bucket join

Tip: use powers of 2

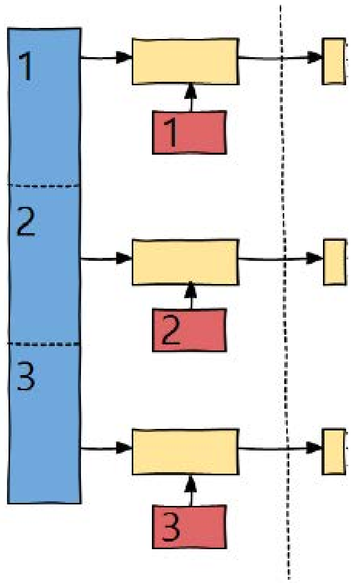


1. MapSide: bucket join
2. The Most Cost-Effective Namenode RAM usage

```
CREATE TABLE access_log (
    …
)
CLUSTERED BY (ip)
    SORTED BY (ip)
    INTO 128 BUCKETS
…;
```

```
CREATE TABLE access_log (
    …
)
CLUSTERED BY (ip)
    SORTED BY (ip)
    INTO 128 BUCKETS
…;
```

Bucket Join limitations?

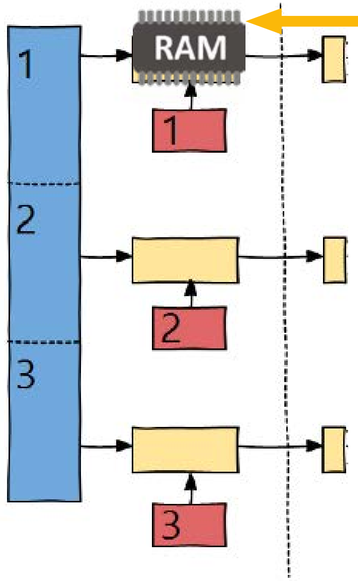


Map Phase

```
CREATE TABLE access_log (
    …
)
CLUSTERED BY (ip)
    SORTED BY (ip)
    INTO 128 BUCKETS
…;
```

Bucket Join limitations?

RAM limitations

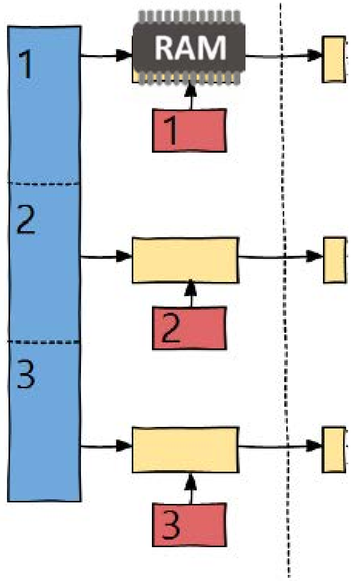


Map Phase

```
CREATE TABLE access_log (
    …
)
CLUSTERED BY (ip)
    SORTED BY (ip)
    INTO 128 BUCKETS
…;
```

see: https://en.wikipedia.org/wiki/Merge_sort
see: https://en.wikipedia.org/wiki/External_sorting

Bucket Join limitations?

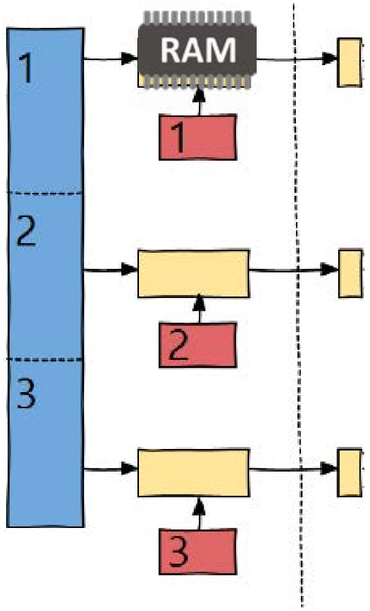


RAM
limitations

Map Phase

```
CREATE TABLE access_log (
    …
)
CLUSTERED BY (ip)
    SORTED BY (ip)
    INTO 128 BUCKETS
…;
```

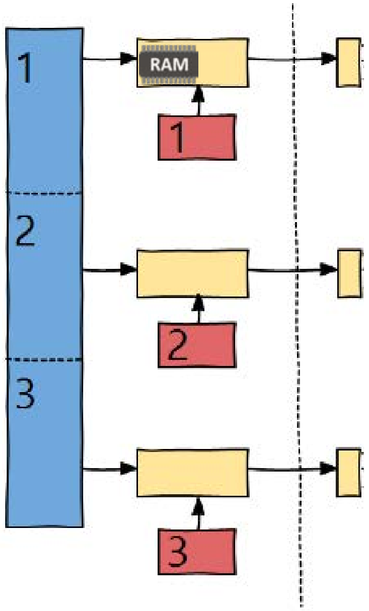
Bucket Join



Map Phase

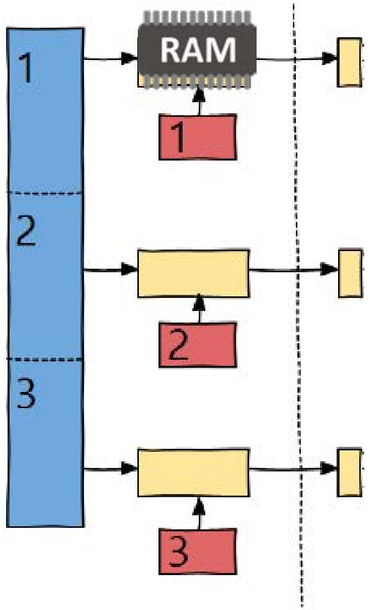
use: merge-sort

Sort-Merge-Bucket Join



Map Phase

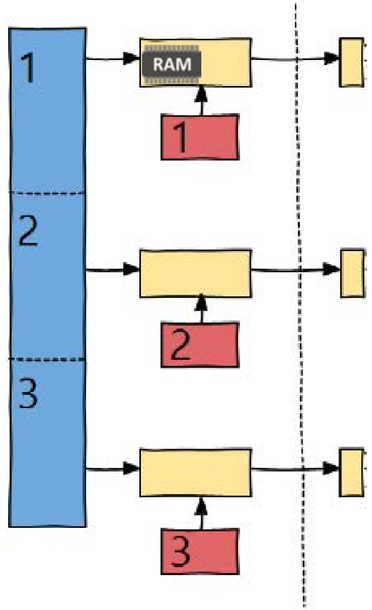
Bucket Join



Map Phase

use: merge-sort

Sort-Merge-Bucket Join



Map Phase

```
hive> SET hive.auto.convert.sortmerge.join=true;
…
```

see: https://cwiki.apache.org/confluence/display/Hive/LanguageManual+JoinOptimization

Summary

Summary

- You can **optimise** Hive warehouse and **execute** HiveQL Map-Side Joins (including: **Bucket** Join, **Sort-Merge-Bucket** Join)

see: https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Joins
see: https://cwiki.apache.org/confluence/display/Hive/LanguageManual+JoinOptimization
see: https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties
see: https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL#LanguageManualDDL-BucketedSortedTables