Databases and Big Data

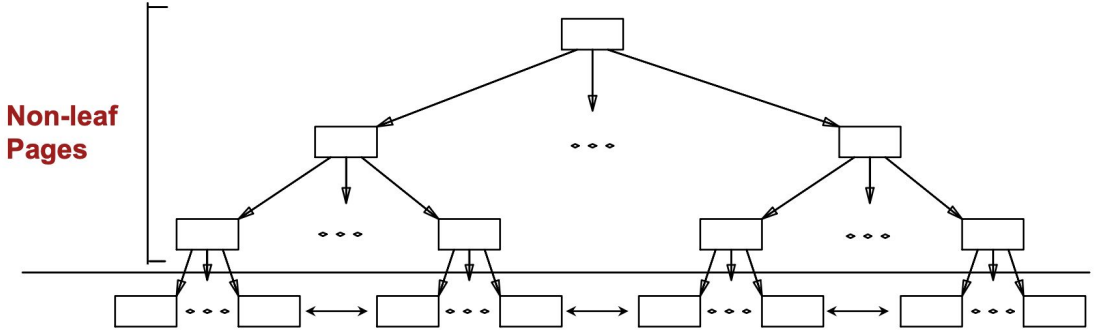
Lab 6

This lab

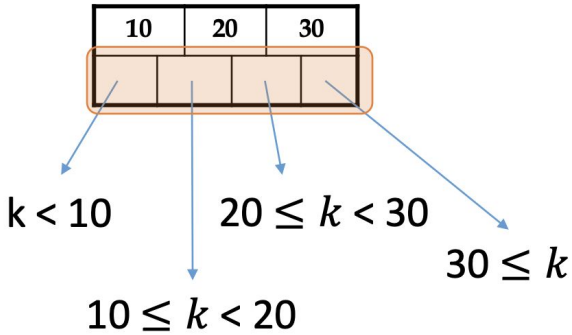
- Database operations and cost
- HDFS installation
- MapReduce Streaming
- MapReduce Examples

More on B+ tree

B+ tree

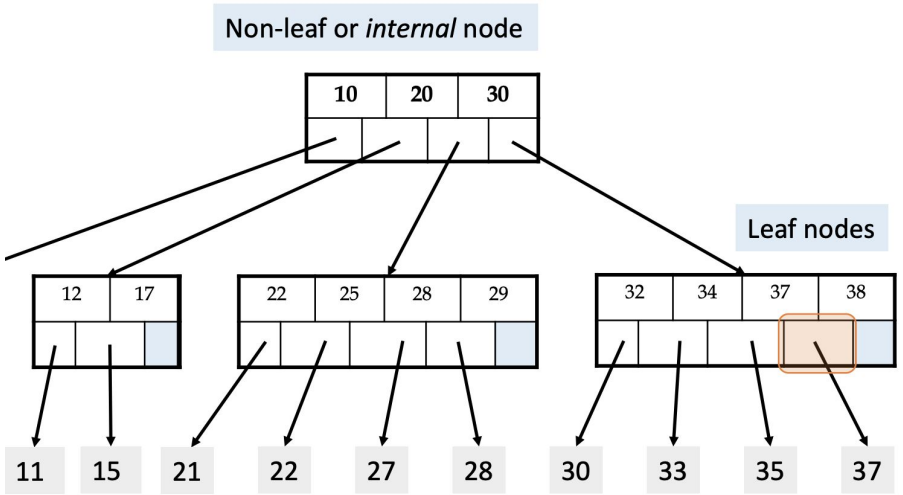


Non-leaf nodes (except the root) are at least half-full



B+ Tree

Each node has 4 slots
- Half-full: 2 slots contains data

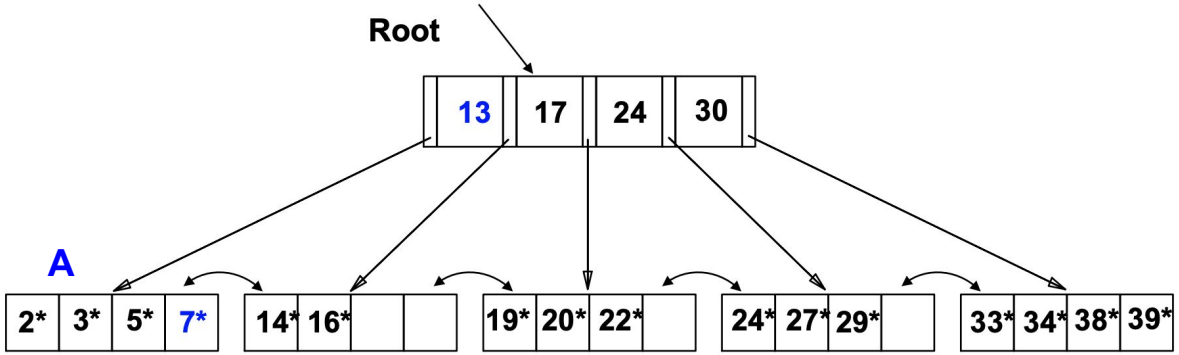


B+ Tree

- Insert:
  - Find correct leaf L
  - Put data to the L
    - If L is not full, DONE!
    - Else
      - Split to two nodes at the middle key
      - If the split is at the leaf node:
        - **Copy** the middle key to the parent
      - **Recursively** for the parent node:
        - Only **pushed up** when splitting internal nodes

B+ Tree

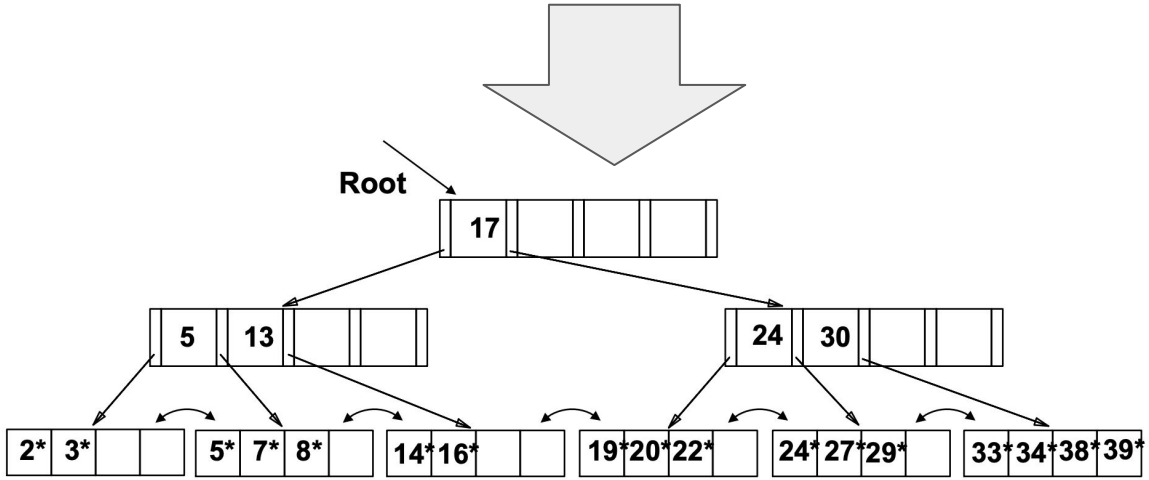
- Insert 8



Split leaf node A

Copy up 5 to the root node:
+ Root node to split
+ **Push** up 17 to a new root

Note the difference when splitting internal node vs. leaf node:
Copy up vs. push up

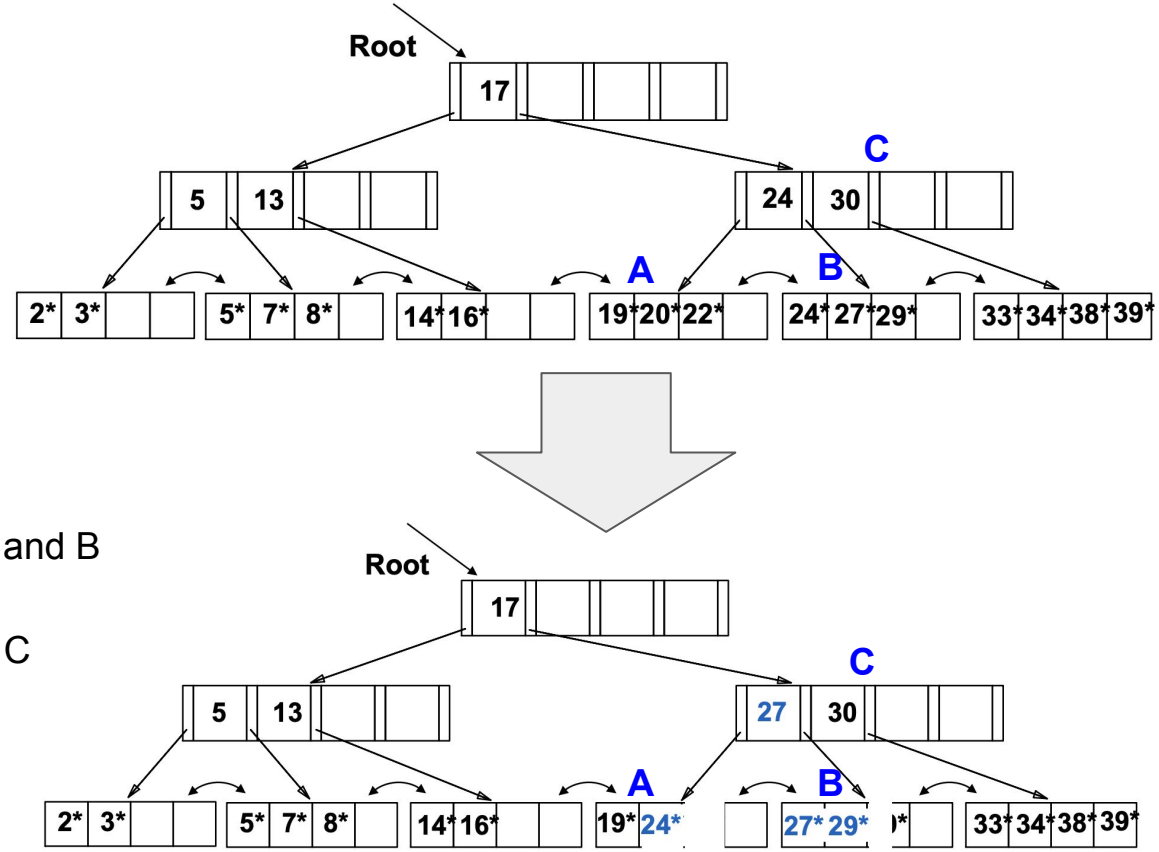


B+ Tree

- Delete:
  - Find leaf L where the entry is
  - Delete the entry:
    - If L is at least half-full. DONE!
    - Else:
      - Redistribution: try to borrow from sibling to make it half-full. If possible, DONE.
      - Else: merge with one of the 2 siblings
        - Must delete one entry from the parent
        - **Recursively** until node is at least half-full.

B+ Tree

- Delete 20, 22



Leaf node redistribution

Delete 20: simple

Delete 22: redistribution between A and B

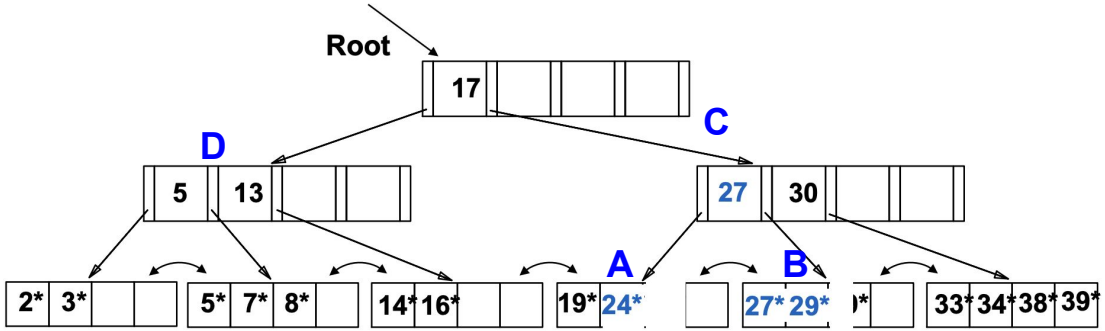
+ 24 moved from B to A
+ 27 moved up to replace 24 in C

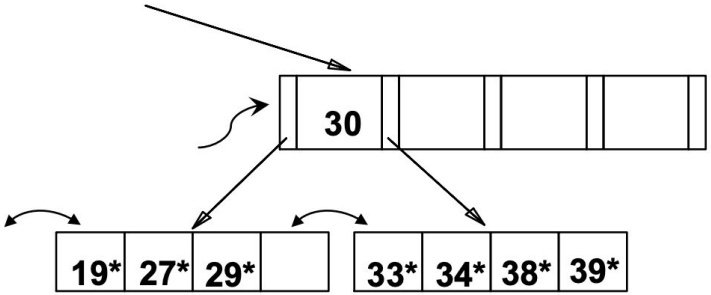
B+ Tree

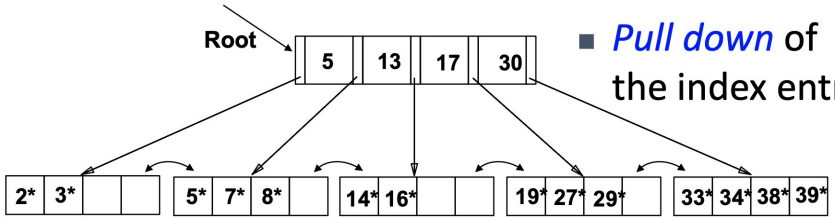
- Delete 24

Must merge A and B

+ Become 1 node
+ Delete the parent entry in C (27)
+ Node C not half full
+ If C cannot redistribute with D:
  + Pull down the parent
  + Merge with D





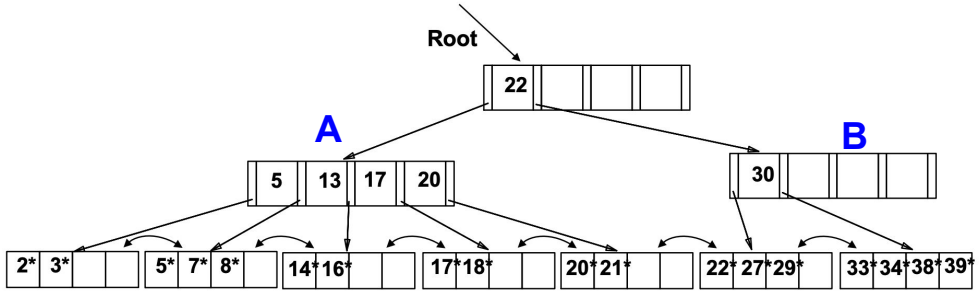


- *Pull down* of the index entry

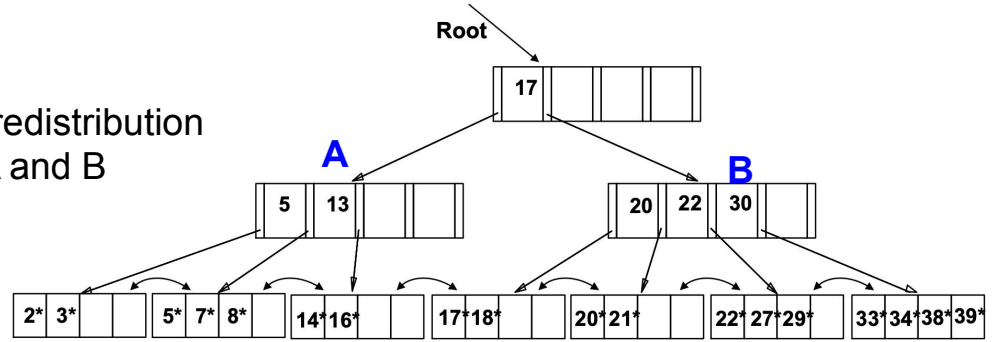
B+ tree

- **Internal nodes** redistribution during delete
  - **Push through** parent node

During deletion of 24



Final tree, after redistribution between A and B



Recaps

- HDFS
  - Most popular distributed file systems
  - NameNode vs. DataNode
  - Access pattern: append-only
  - Replication
- MapReduce:
  - On top of HDFS
  - Map phase
  - Reduce phase
  - Shuffle phase

HDFS Installation

- We’re using **hadoop-3.1.2**
- Google
- A good source:

https://www.linode.com/docs/databases/hadoop/how-to-install-and-set-up-hadoop-cluster/

HDFS Setup

- Step 1: make sure Namenode can SSH to other nodes without passwords
- Step 2: edit configuration files at all nodes
- Step 3: start it

HDFS Setup

- All nodes have same set of configuration files
  - .xml files
  - Workers

Same at every machine
Use PublicDNS, not IP

Config files

- **core-site.xml**
- **hdfs-site.xml**
- yarn-site.xml
- mapred-site.xml
- **workers**

- Start

bin/hdfs namenode -format

sbin/start-dfs.sh

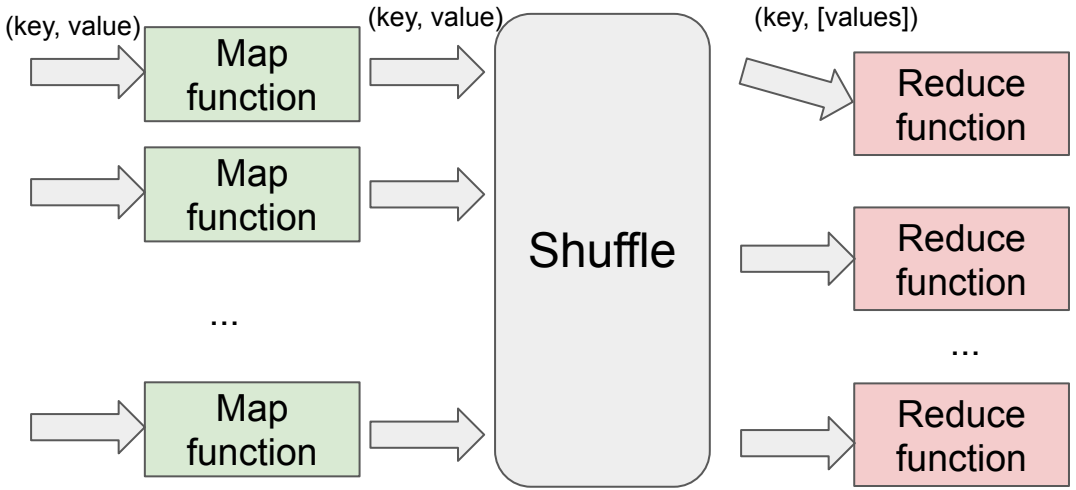
bin/hadoop dfsadmin -report

Check HDFS cluster status

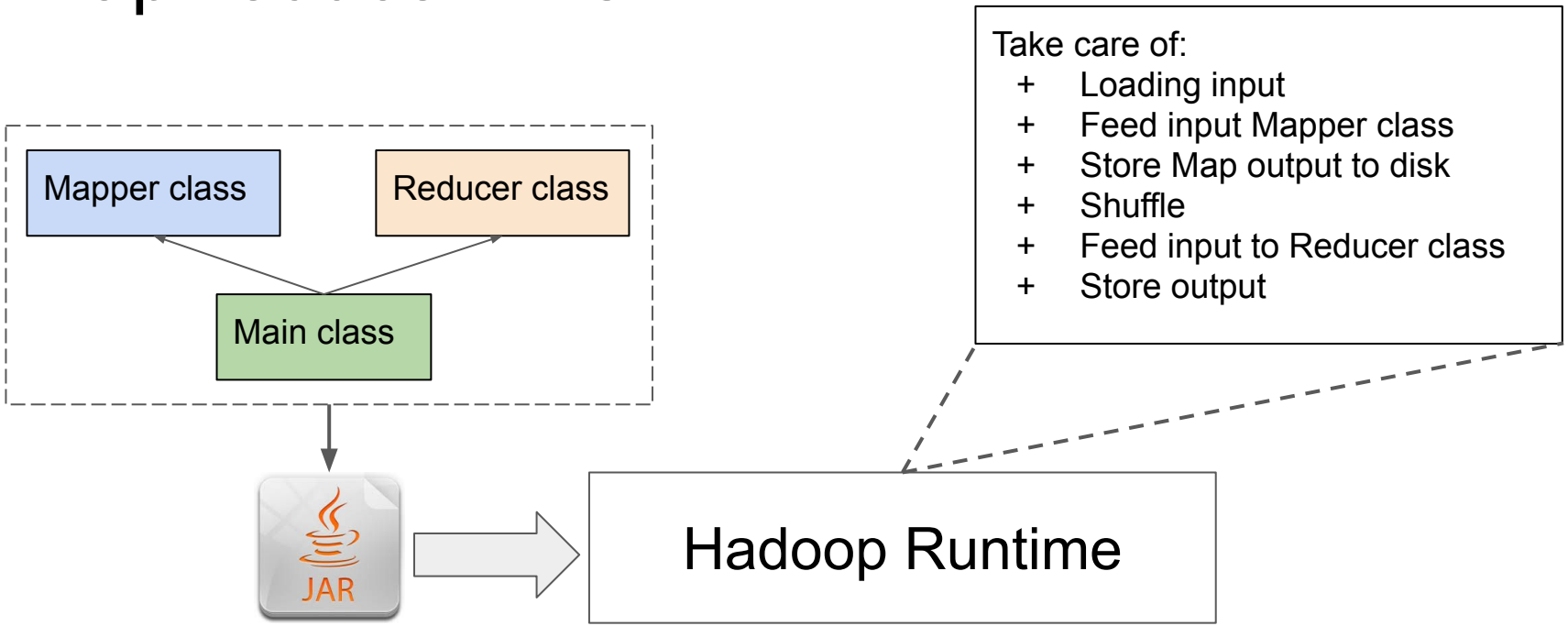
Hadoop Streaming

Hadoop Streaming

- Warning!
  - This is **not for serious MapReduce programming**
  - Quick and dirty way to write a simple MapReduce job
- Recall
  - MapReduce execution

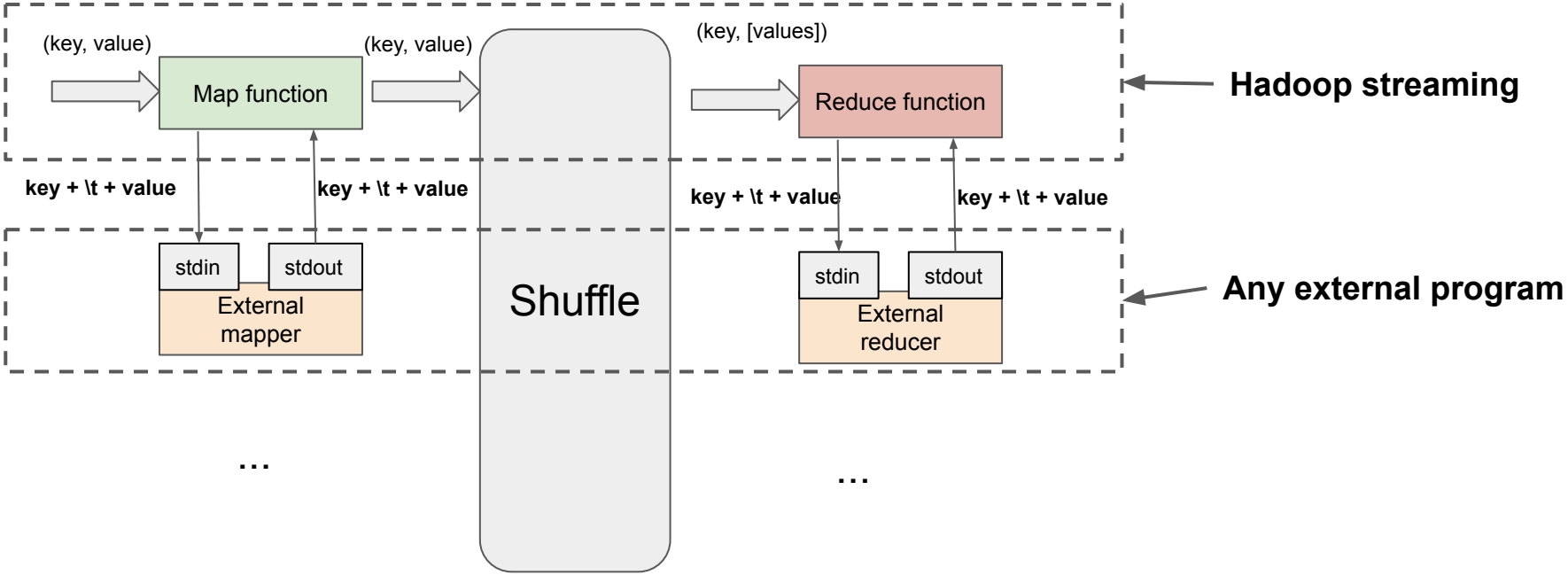


MapReduce APIs




```
hadoop jar <job>.jar <class name> <input> <output> [options]
```

Hadoop Streaming



Hadoop Streaming

- What you need to write
  - A mapper program:
    - Read lines from stdin
    - Output key-value pair, separated by “\t” to stdout

```
import sys

for line in sys.stdin:
    keys = line.strip().split()
    for key in keys:
        print( "%s\t%d" % (key, 1) )
```

Hadoop Streaming

- What you need to write
  - A reducer program:
    - Read lines from stdin, of format <key + “\t” + value>
      - These lines are sorted by keys
    - Output reducer result

```
import sys

last_key = None
running_total = 0

for input_line in sys.stdin:
   this_key, value = input_line.strip().split("\t", 1)
   value = int(value)

   if last_key == this_key:
       running_total += value
   else:
       if last_key:
           print( "%s\t%d" % (last_key, running_total) )
       running_total = value
       last_key = this_key

if last_key == this_key:
   print( "%s\t%d" % (last_key, running_total) )
```

Hadoop Streaming

- Running it:

  bin/mapred streaming -D mapred.child.java.opts=-Xmx4096m -input /input -output /out -mapper "python <path>/mapper.py" -reducer "python <path>/reducer.py"

- It’s the same as the following (single machine, Linux):

  cat <input> | python <path>/mapper.py | sort | python <path>/reducer.py

- See it scales with 3,5,7 nodes