MapReduce (M/R)

Part 1

What is MapReduce?

- MapReduce (M/R) is a programming model for data processing. M/R programs are inherently parallel, thus putting very large-scale data analysis into the hands of anyone with enough machines at their disposal.
- M/R works by breaking the processing into two phases: the map phase and the reduce phase. Each phase has key/value pairs as input and output. The programmer also specifies two functions: the map function and the reduce function.

Why Key/Value Data?

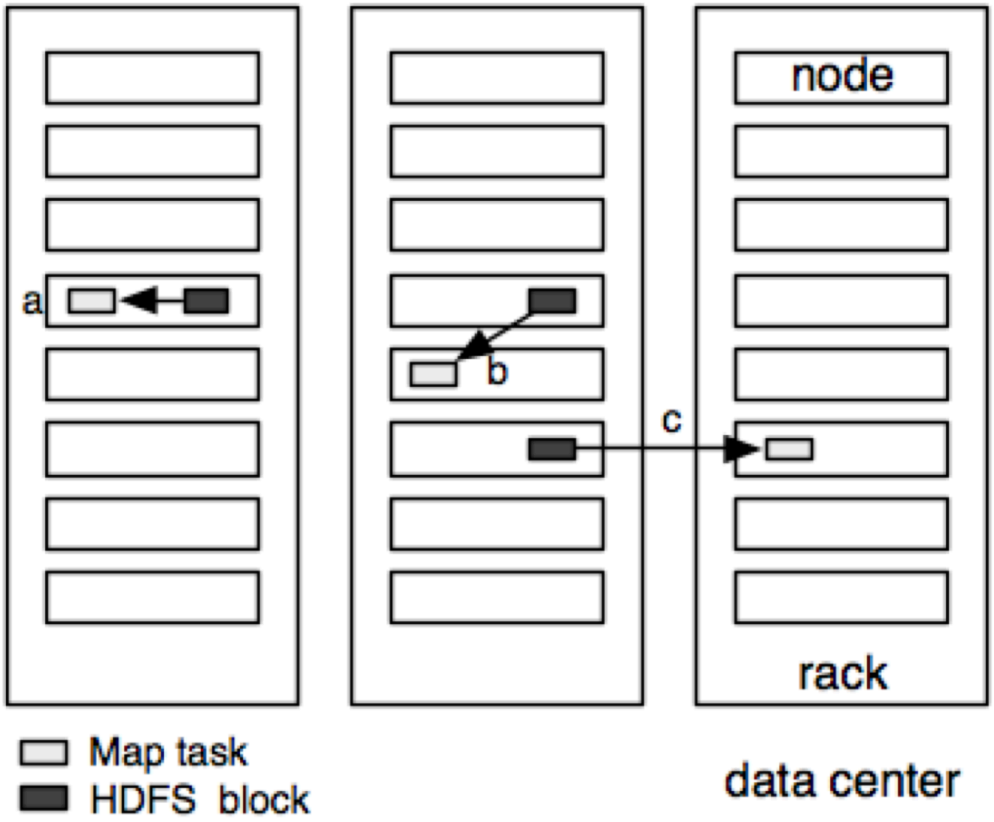
- Using key/value data as the foundation of MapReduce operations allows for a powerful programming model that is surprisingly widely applicable.
- Much data is either intrinsically key/value in nature or can be represented in such a way, e.g., name/contact info, account #/account details, and word/page #. It is a simple model with broad applicability and semantics straightforward enough that programs defined in terms of it can be applied by a framework like Hadoop.
- MapReduce, with its key/value interface, provides such a level of abstraction, whereby the programmer only has to specify these transformations and Hadoop handles the complex process of applying it to arbitrarily large data sets.

Some Terminology

- **An M/R job**: a unit of work that a client wants to be performed, consisting of the input data, the M/R program, and configuration information. Hadoop runs the job by dividing it into tasks: map tasks and reduce tasks.
- **Jobtracker (on name node)**: coordinates all the jobs by scheduling tasks to run on tasktrackers. It keeps a record of the overall progress of each job. If a task fails, the jobtracker can reschedule it on a different tasktracker.
- **Tasktrackers (on data nodes)**: run tasks and send progress reports to the jobtracker.
- **Input splits**: fixed-size pieces (64 or 128 MB) of the input to an M/R job, one map task for each split, which runs the user-defined map function for each record in the split.

Data Locality Optimization

Hadoop does its best to run the map task on a node where the input data resides in HDFS so that it doesn't use valuable cluster bandwidth.



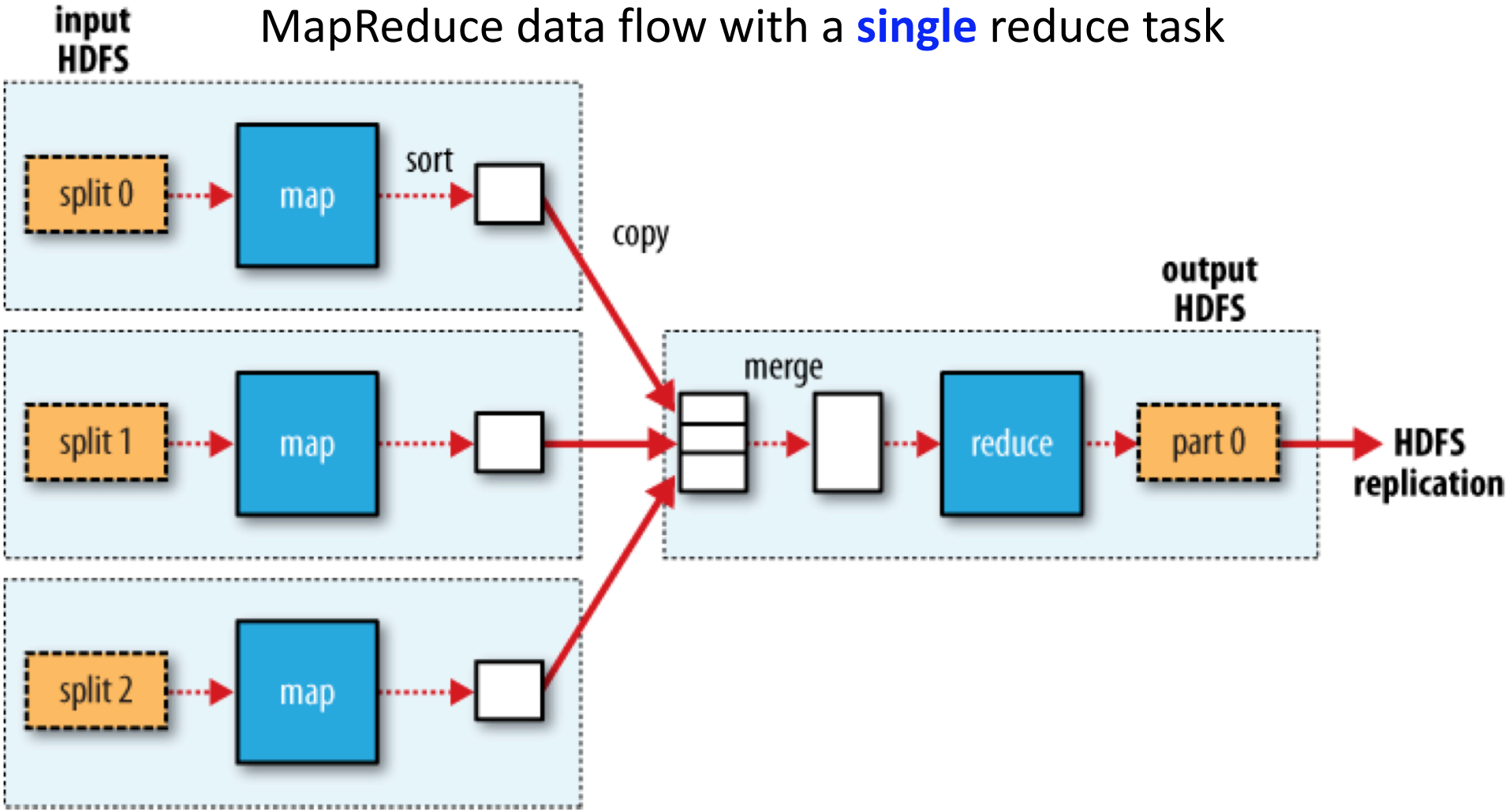
a: Data-local map task
b: rack-local map task
c: off-rack map task

Map Tasks and Reduce Tasks

- Map tasks write their intermediate output to the local disk temporarily, not to HDFS.
- Reduce tasks consumes the map output. Once the job is complete the map output can be thrown away.
- Reduce tasks don't have the advantage of data locality since the input to a single reduce task is normally the output from all mappers from different nodes.
- The map outputs are sorted first and then transferred across the network to the node where the reduce task is running, where they are merged and then passed to the user-defined reduce function.
- The output of the reduce task is normally stored in HDFS (the first replica of the output is stored on the local node and other replicas are on off-rack nodes.

Data Flow of M/R Tasks

MapReduce data flow with a **single** reduce task



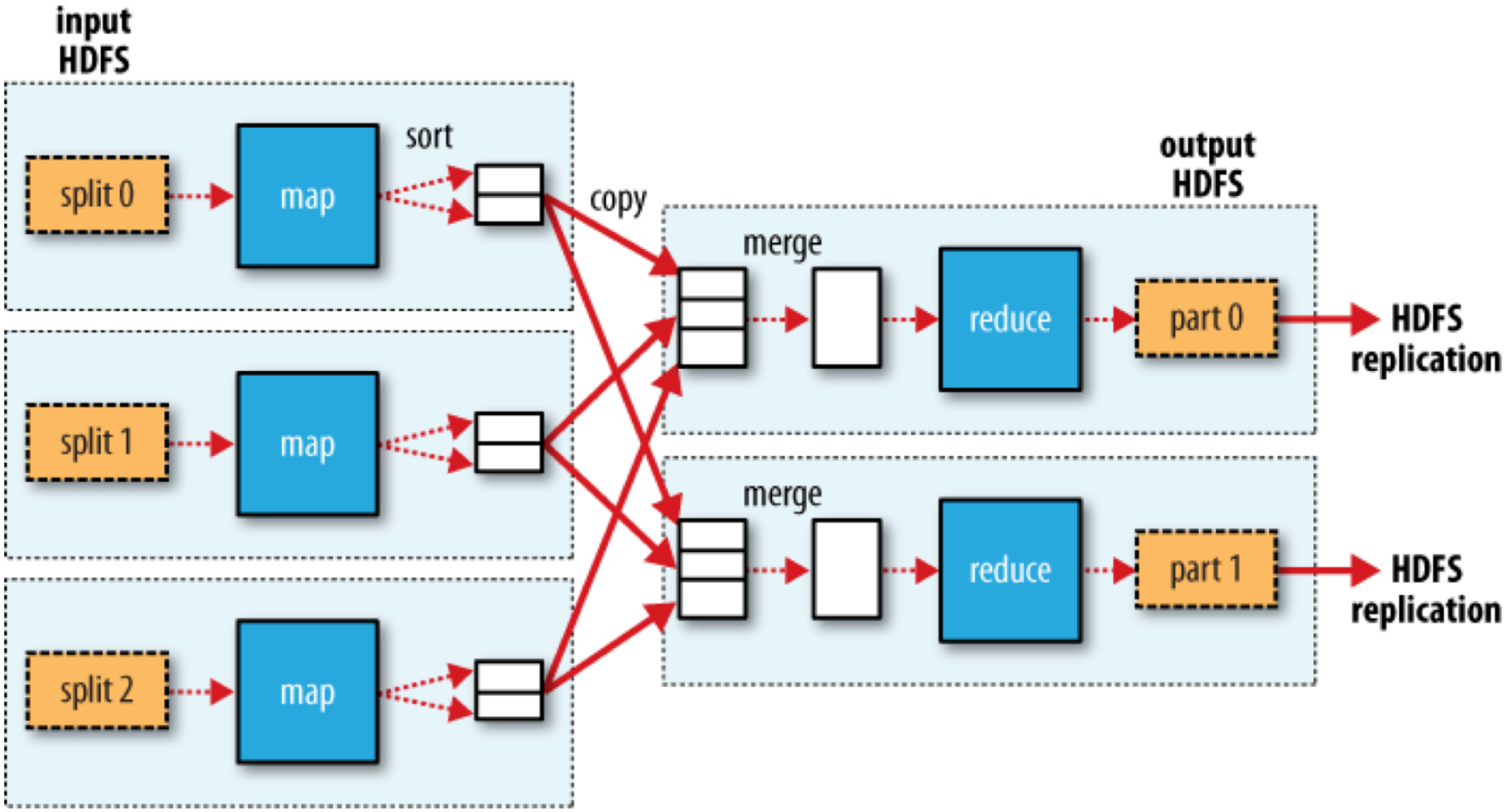
- Dotted boxes: nodes
- Light arrows: data transfers on a node
- Heavy arrows: data transfers between nodes.

Data Flow of M/R Tasks: Other Situations

- **Multiple reducers**
  - The map tasks partition their output, each creating one partition for each reduce task.
  - There can be many keys (and their associated values) in each partition, but the records for any given key are all in a single partition.
  - The partitioning can be controlled by a user-defined partitioning function, but normally the default partitioner works very well.
- **No reducers**
  - This can be appropriate when the shuffle is not needed since the processing can be carried out entirely in parallel.

Data Flow of M/R Tasks

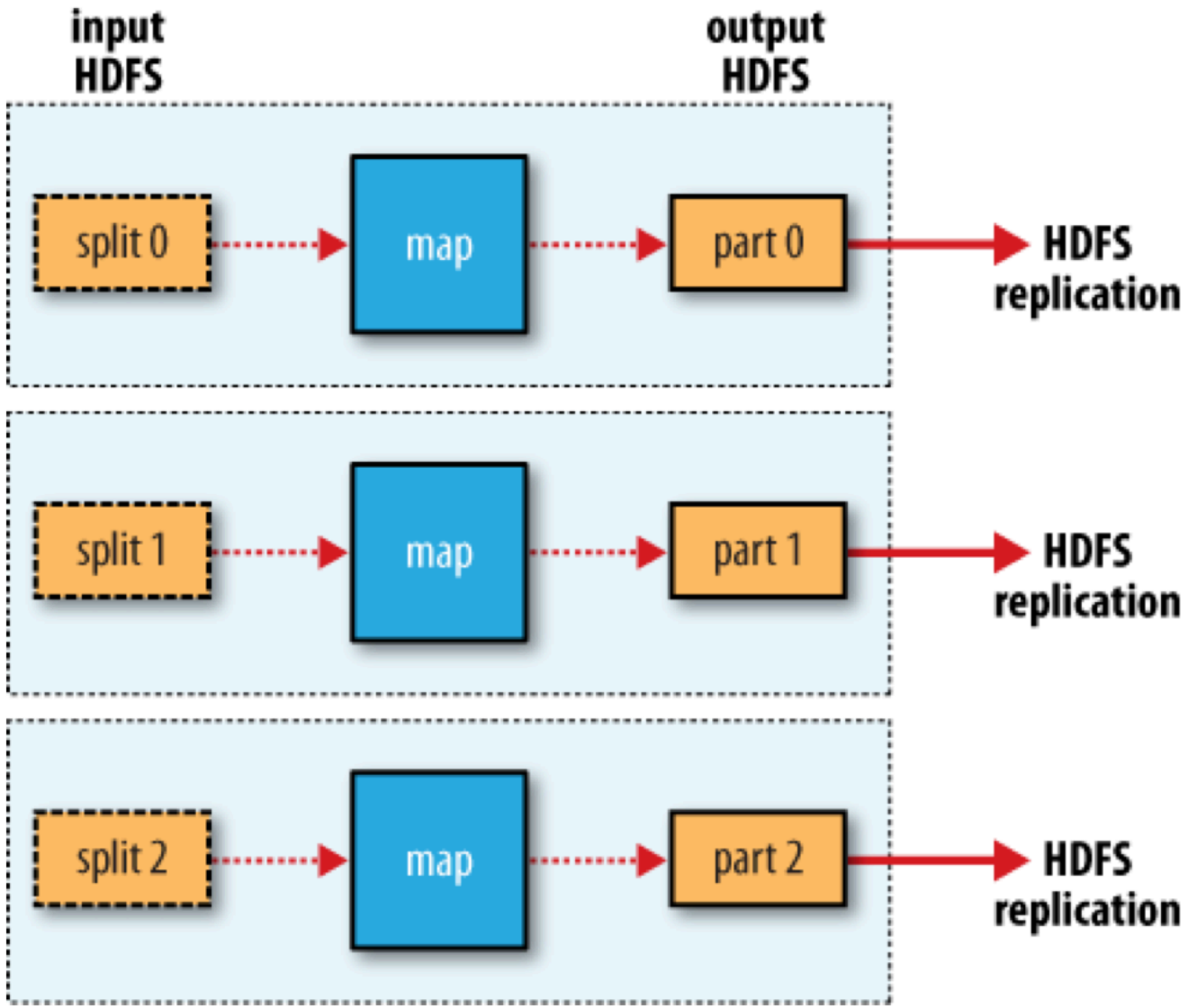
MapReduce data flow with **multiple** reduce tasks



- Dotted boxes: nodes
- Light arrows: data transfers on a node
- Heavy arrows: data transfers between nodes.

Data Flow

MapReduce data flow with **no** reduce tasks



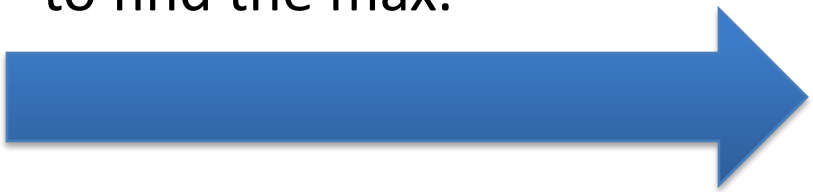
The Combiner Functions

- A combiner function is for optimizing/minimizing the number of key/value pairs that will be shuffled across from the mappers to reducers to save as most bandwidth as possible.
- The function runs on the map output and its output forms the input to the reduce function.
- There is no guarantee of how many times the function is called for a particular map output record, if at all.
- No matter how many times the function is called, the reducer produces the same output.

Combiner Function Examples

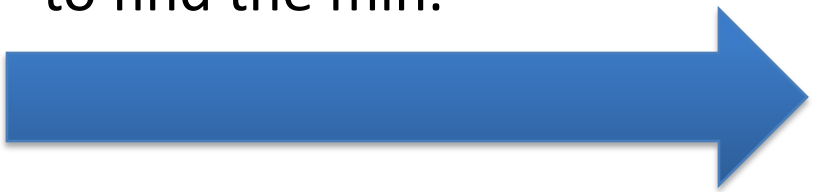
IBM	2000
IBM	2556
IBM	2459
IBM	2466
IBM	2744
IBM	2008
IBM	2014
IBM	2020
IBM	2345
IBM	2122
IBM	2009
IBM	2567
IBM	2876
IBM	2349
IBM	2001
IBM	2378

A combiner function to find the max.



IBM	2876

A combiner function to find the min.



IBM	2000

Analyze Data with MapReduce

- Use an NCDC (National Climatic Data Center) dataset, the global temperature dataset.
- The input to the map phase is the raw NCDC data. We choose a text input format that gives us each line in the dataset as a text value. The key is the offset of the beginning of the line from the beginning of the file.
- The map function pulls out the year and the air temperature for the reduce function.
- The reduce function finds the maximum temperature for each year.

A Portion of the Dataset

```
0029029070999991901010106 … N0000001N9-00781+99999102001 …
0029029070999991901010113 … N0000001N9-00721+99999102001 …
0029029070999991901010120 … N0000001N9-00941+99999102001 …
0029029070999991901010206 … N0000001N9-00611+99999101831 …
0029029070999991901010213 … N0000001N9-00561+99999101761 …
0029029070999991901010220 … N0000001N9-00281+99999101751 …
0029029070999991901010306 … N0000001N9-00671+99999101701 …
```

```
(0,   0029029070999991901010106 … N0000001N9-00781+99999102001 …)
(134, 0029029070999991901010113 … N0000001N9-00721+99999102001 …)
(268, 0029029070999991901010120 … N0000001N9-00941+99999102001 …)
(402, 0029029070999991901010206 … N0000001N9-00611+99999101831 …)
(536, 0029029070999991901010213 … N0000001N9-00561+99999101761 …)
(670, 0029029070999991901010220 … N0000001N9-00281+99999101751 …)
(804, 0029029070999991901010306 … N0000001N9-00671+99999101701 …)
```

These lines are presented to the map function as the key-value pairs

What the Map Function Emits

1901 -78

1901 -72

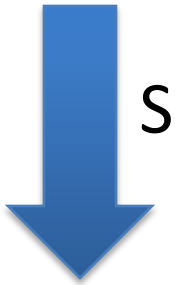
1901 -94

1901 -61

1901 -56

1901 -28

1901 -67



Sort by key (shuffle)

1901 [-78, -72, -94, -61, -56, -28, -67]

The Action of the Reduce Function

The reduce function iterates through the list and pick up the maximum reading. For the sample data, the final output should be:

1901 -28

The Map and Reduce Functions

- Specific map and reduce functions must be written to process the data and to get the desired results.
- We will write a map function called **temp_map.py** and a reduce function called **temp_reduce.py**.
- These two functions must at least be given the owner execution permission, which can be achieved by the following commands:

chmod 744 temp_map.py

chmod 744 temp_reduce.py

```
-rwxr--r--  1 root root      241 Feb 14 08:20 temp_map.py
-rwxr--r--  1 root root      398 Feb 14 08:20 temp_reduce.py
```

The Map Function in Python

The following is a map function written in Python with vi, which is applied to the first 7 rows of the global temperature dataset. It successfully extracted the information wanted and emitted the results.

temp_map.py

```
#!/usr/bin/env python

import re
import sys

for line in sys.stdin:
    val = line.strip()
    (year, temp, q) = (val[15:19], val[87:92], val[92:93])
    if (temp != "+9999" and re.match("[01459]", q)):
        print "%s\t%s" % (year, temp)
~
```

```
1901    -0078
1901    -0072
1901    -0094
1901    -0061
1901    -0056
1901    -0028
1901    -0067
```

```
[root@sandbox stsci5065]# cat /stsci5065/1901-1 | ./temp_map.py
```

The Reduce Function in Python

The following is a reduce function written in Python, which is applied to the output of the map function. It successfully processed output emitted by the map function and found the maximum temperature of the sample dataset.

temp_reduce.py

```
#!/usr/bin/env python

import sys

(last_key, max_val) = (None, None)
for line in sys.stdin:
    (key, val) = line.strip().split("\t")
    if last_key and last_key != key:
        print "%s\t%s" % (last_key, max_val)
        (last_key, max_val) = (key, int(val))
    else:
        (last_key, max_val) = (key, max(max_val, int(val)))
if last_key:
    print "%s\t%s" % (last_key, max_val)
~
```

```
# cat /stsci5065/1901-1 | ./temp_map.py | ./temp_reduce.py
```

```
1901    -28
```

The Map Function Applied to the Whole Dataset

The same map function is applied to the whole dataset. It successfully extracted all the information wanted and emitted the results.

temp_map.py

```
#!/usr/bin/env python

import re
import sys

for line in sys.stdin:
    val = line.strip()
    (year, temp, q) = (val[15:19], val[87:92], val[92:93])
    if (temp != "+9999" and re.match("[01459]", q)):
        print "%s\t%s" % (year, temp)
~
```

```
[root@sandbox stsci5065]# cat /stsci5065/1901 | ./temp_map.py |less
```

```
1901    -0094
1901    -0061
1901    -0056
1901    -0028
1901    -0067
1901    -0033
1901    -0028
1901    -0033
1901    -0044
1901    -0039
1901    +0000
1901    +0006
1901    +0000
1901    +0006
1901    +0006
1901    -0011
1901    -0033
1901    -0050
1901    -0044
1901    -0028
1901    -0033
1901    -0033
1901    -0050
1901    -0033
1901    -0028
...     ...
```

The M/R Functions Applied to the Whole Dataset

The same map/reduce functions are applied to the whole 1901 dataset. The map function produces intermediate results, which are fed into the reduce function. The maximum temperature of 1901 is found successfully.

temp_reduce.py

```
#!/usr/bin/env python

import sys

(last_key, max_val) = (None, None)
for line in sys.stdin:
    (key, val) = line.strip().split("\t")
    if last_key and last_key != key:
        print "%s\t%s" % (last_key, max_val)
        (last_key, max_val) = (key, int(val))
    else:
        (last_key, max_val) = (key, max(max_val, int(val)))
if last_key:
    print "%s\t%s" % (last_key, max_val)
```

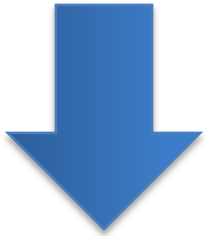
```
# cat /stsci5065/1901 | ./temp_map.py | ./temp_reduce.py
```

```
1901    317
```

The M/R Functions Applied to a Bigger Dataset

The same map/reduce functions are applied to a bigger dataset, the global temperature record of 1901 and 1902. The maximum temperatures of these two years are found successfully.

```
]# cat /stsci5065/1901-1902 | ./temp_map.py | ./temp_reduce.py
```



```
1901    317
1902    244
```