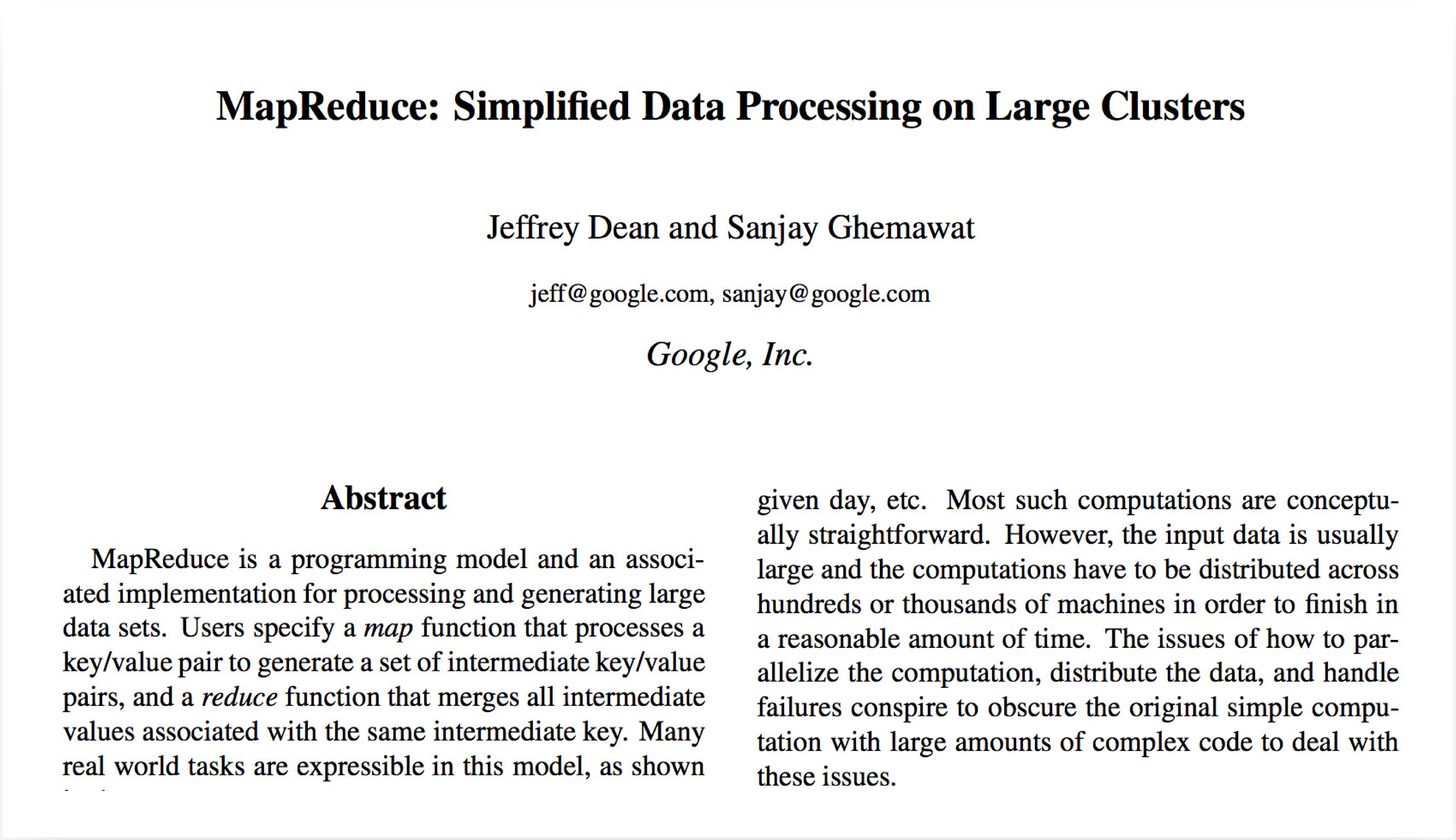
Yandex

MapReduce

MapReduce



MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

jeff@google.com, sanjay@google.com

Google, Inc.

Abstract

MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a *map* function that processes a key/value pair to generate a set of intermediate key/value pairs, and a *reduce* function that merges all intermediate values associated with the same intermediate key. Many real world tasks are expressible in this model, as shown

given day, etc. Most such computations are conceptually straightforward. However, the input data is usually large and the computations have to be distributed across hundreds or thousands of machines in order to finish in a reasonable amount of time. The issues of how to parallelize the computation, distribute the data, and handle failures conspire to obscure the original simple computation with large amounts of complex code to deal with these issues.

MapReduce: Simplified Data Processing on Large Clusters, Symposium on Operating Systems Design and Implementation (OSDI, 2004)

Jeffrey Dean

Jeffrey Dean

- When Jeff Dean designs software, he first codes the binary and then writes the source as documentation.

Jeffrey Dean

- When Jeff Dean designs software, he first codes the binary and then writes the source as documentation.
- Jeff Dean once failed a Turing test when he correctly identified the 203rd Fibonacci number in less than a second.

Jeffrey Dean

- When Jeff Dean designs software, he first codes the binary and then writes the source as documentation.
- Jeff Dean once failed a Turing test when he correctly identified the 203rd Fibonacci number in less than a second.
- The rate at which Jeff Dean produces code jumped by a factor of 40 in late 2000 when he upgraded his keyboard to USB2.0.

Jeffrey Dean

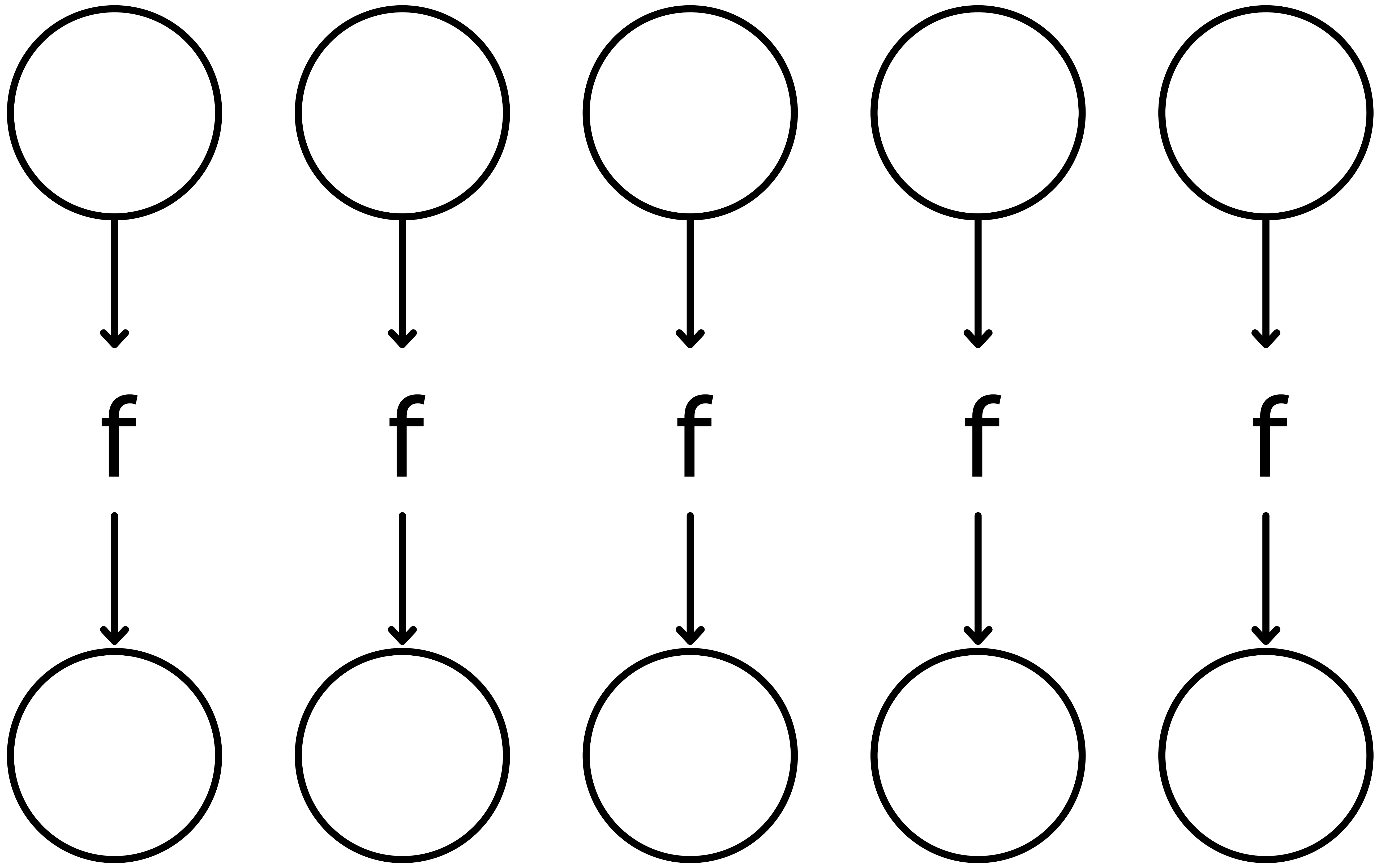
- When Jeff Dean designs software, he first codes the binary and then writes the source as documentation.
- Jeff Dean once failed a Turing test when he correctly identified the 203rd Fibonacci number in less than a second.
- The rate at which Jeff Dean produces code jumped by a factor of 40 in late 2000 when he upgraded his keyboard to USB2.0.
- You use 10% of your brain. The other 90% is running one of Jeff's **mapreduce** jobs.

Jeffrey Dean

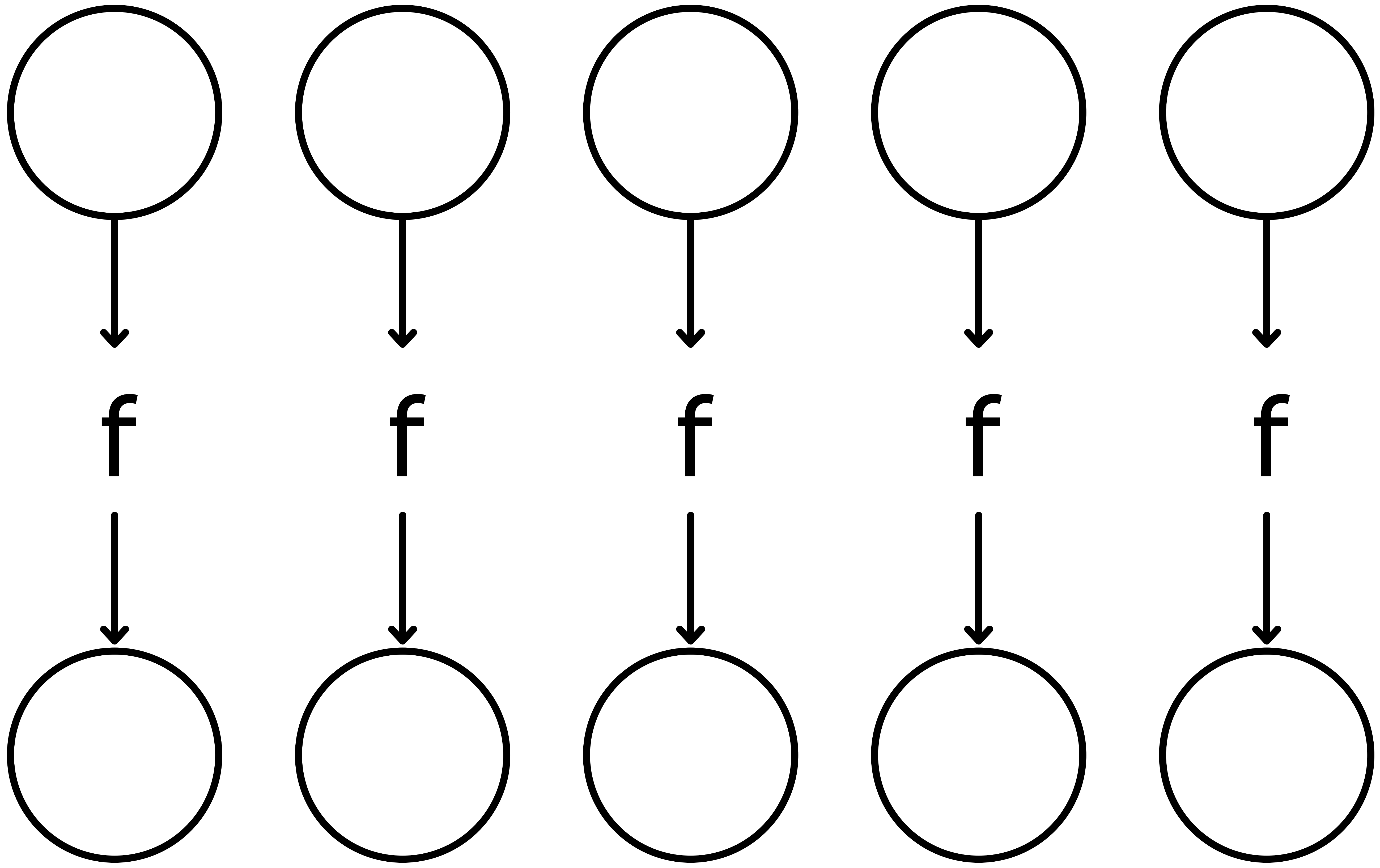
- When Jeff Dean designs software, he first codes the binary and then writes the source as documentation.
- Jeff Dean once failed a Turing test when he correctly identified the 203rd Fibonacci number in less than a second.
- The rate at which Jeff Dean produces code jumped by a factor of 40 in late 2000 when he upgraded his keyboard to USB2.0.
- You use 10% of your brain. The other 90% is running one of Jeff's **mapreduce** jobs.

https://www.quora.com/What-are-all-the-Jeff-Dean-facts

Map

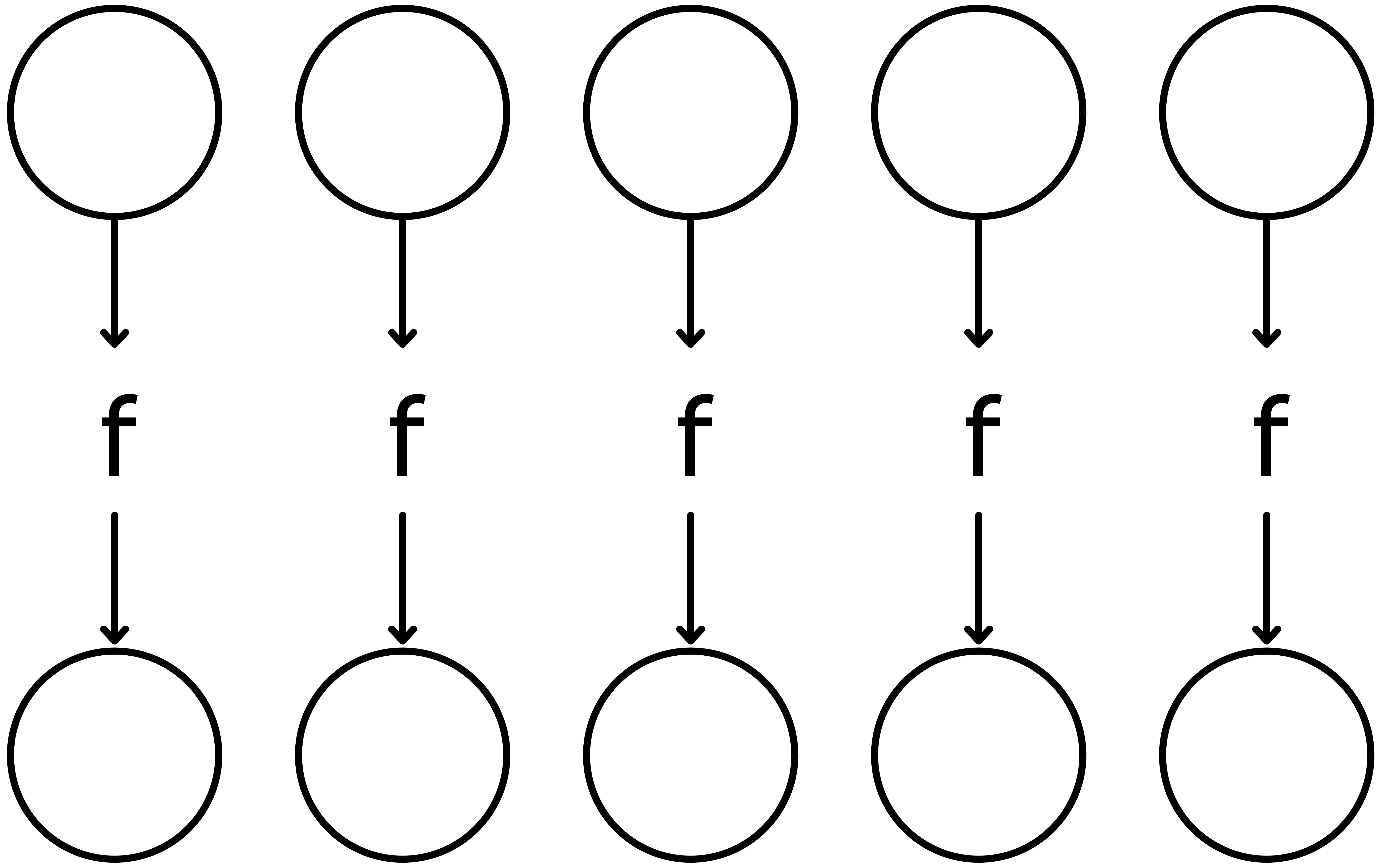


Map



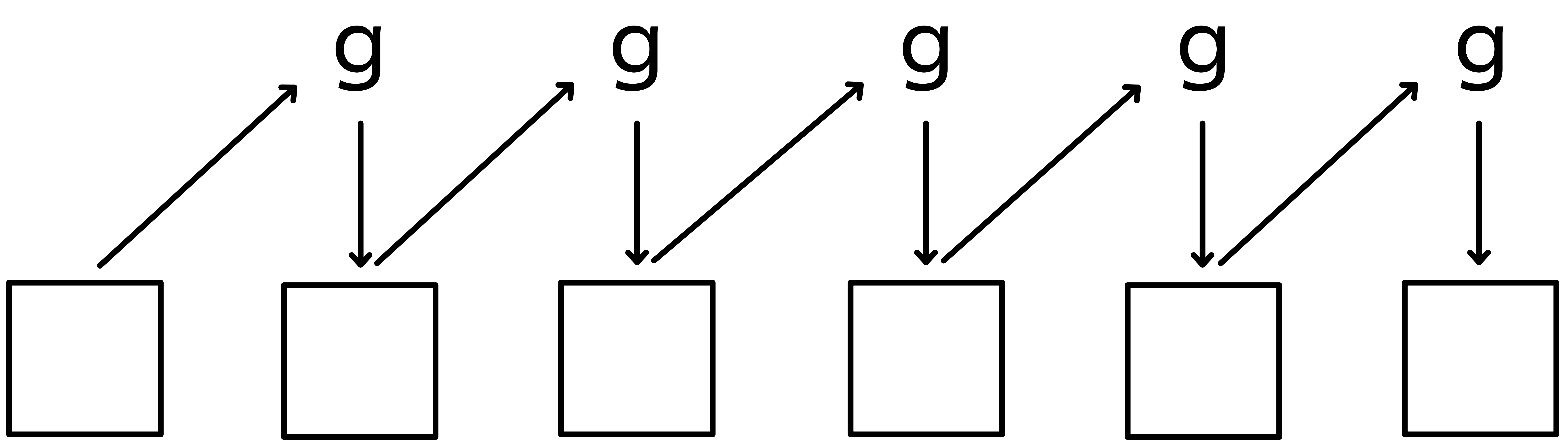
```
>>> map(lambda x: x*x, [1,2,3,4])
???
```

Map

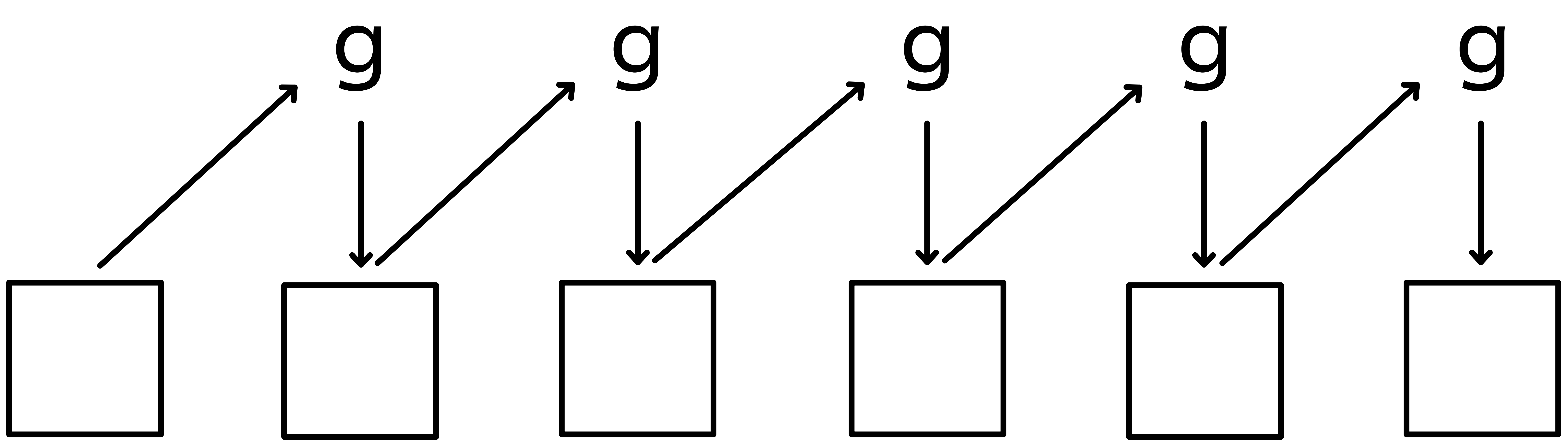


```
>>> map(lambda x: x*x, [1,2,3,4])
[1, 4, 9, 16]
```

Fold / Reduce / Aggregate

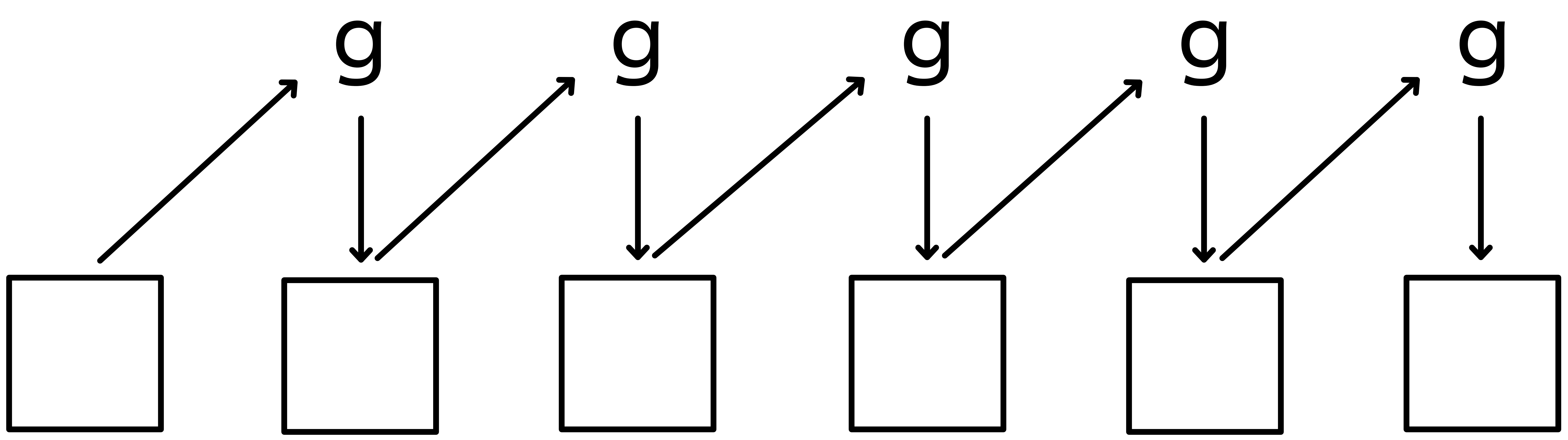


Fold / Reduce / Aggregate



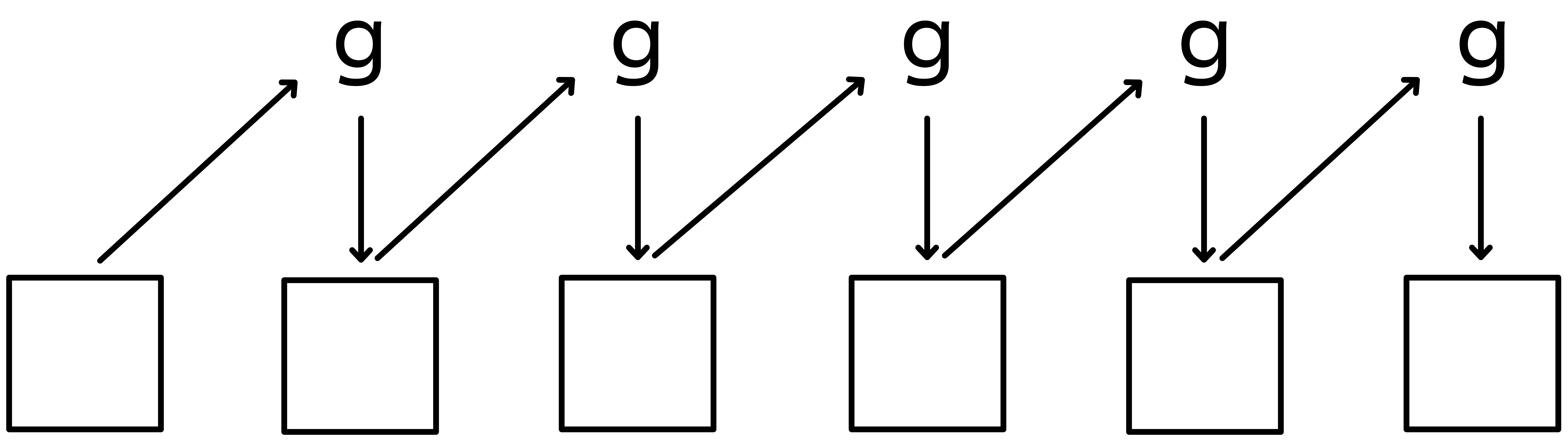
```
>>> reduce(operator.sum, [1, 4, 9, 16])
???
```

Fold / Reduce / Aggregate



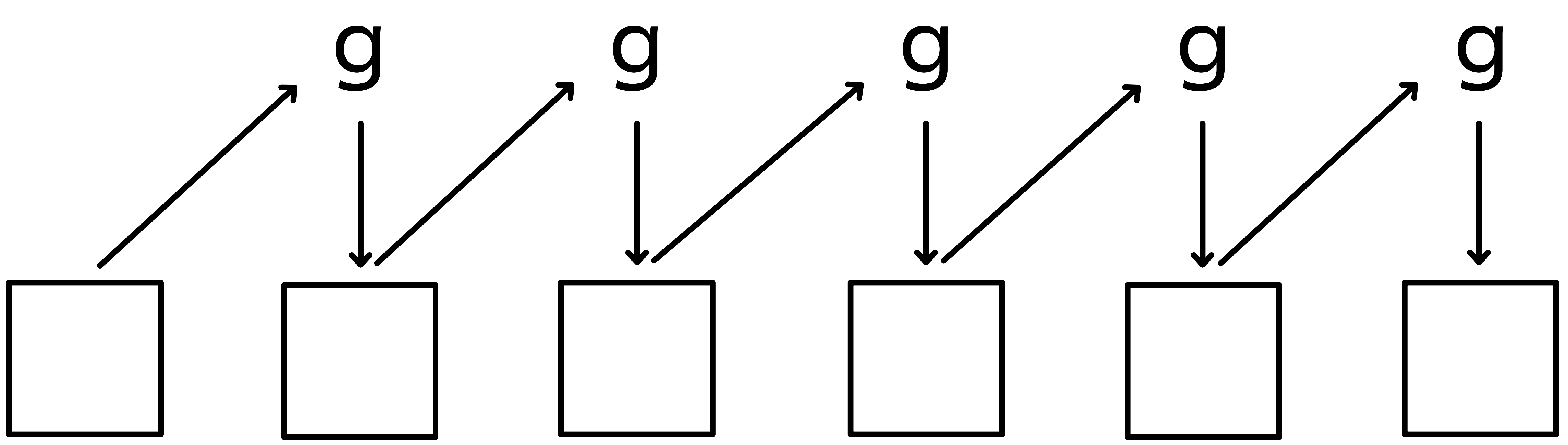
```
>>> reduce(operator.sum, [1, 4, 9, 16])
>>> reduce(operator.sum, [5 = 1 + 4, 9, 16])
…
```

Fold / Reduce / Aggregate



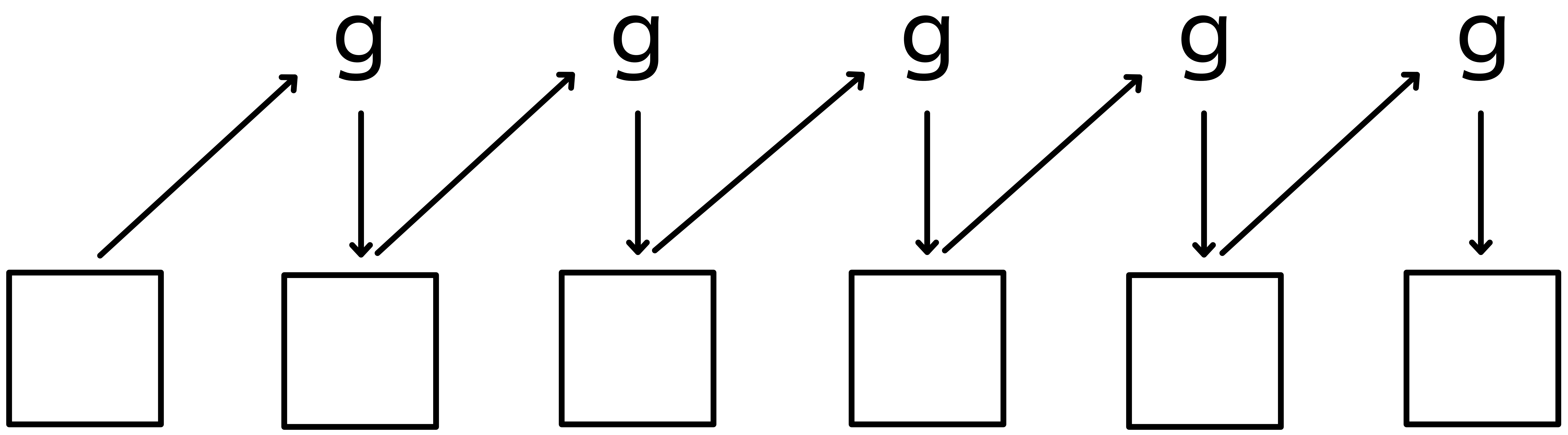
```
>>> reduce(operator.sum, [1, 4, 9, 16])
>>> reduce(operator.sum, [5, 9, 16])
>>> reduce(operator.sum, [14, 16])
30
```

Fold / Reduce / Aggregate



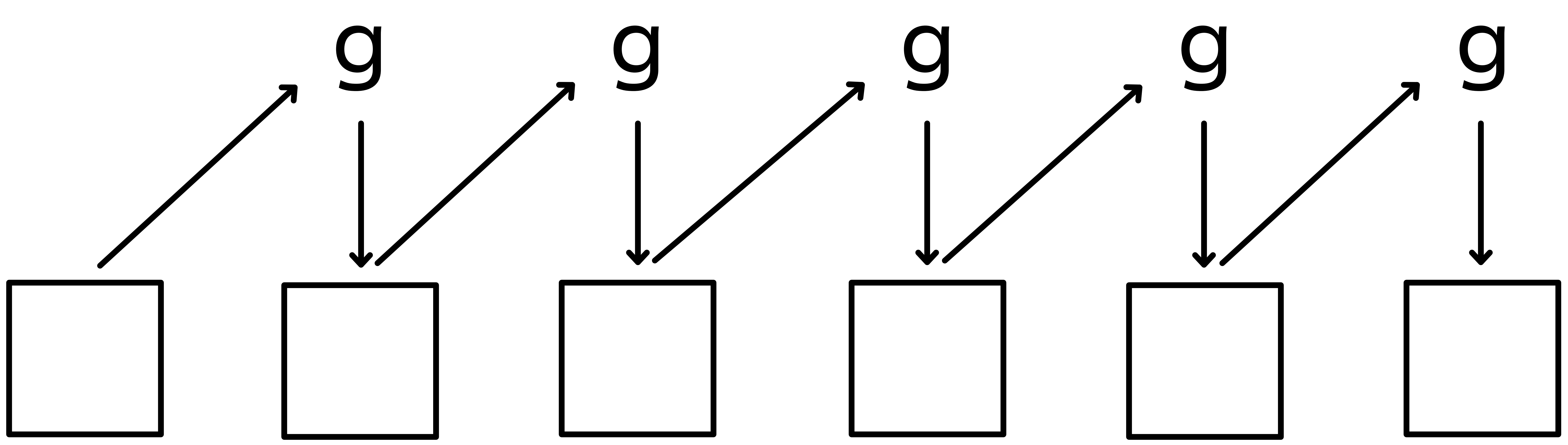
```
>>> average = lambda x, y: (x + y) / 2.
```

Fold / Reduce / Aggregate



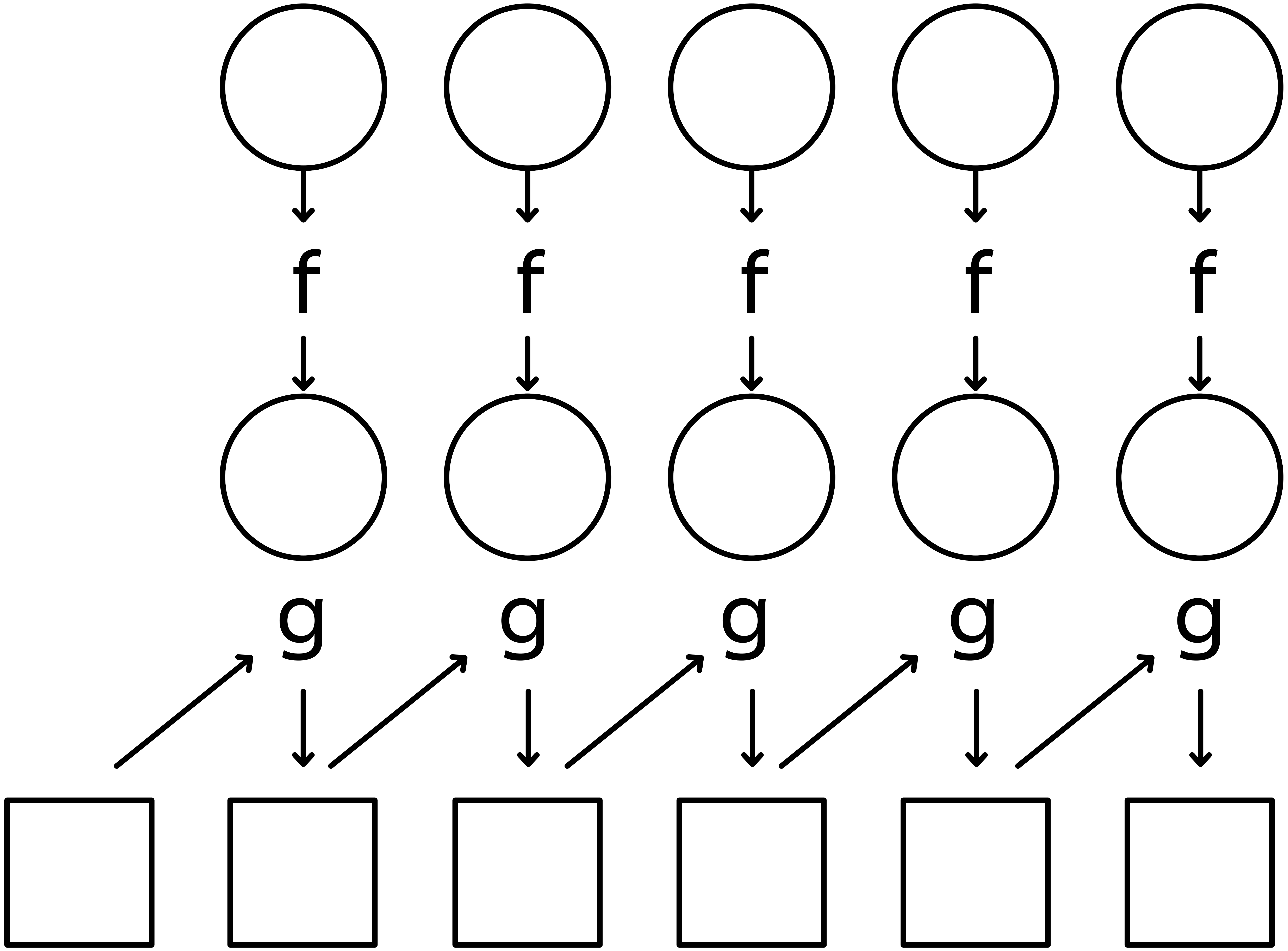
```
>>> average = lambda x, y: (x + y) / 2.
>>> reduce(average, [1, 2, 3])
>>> reduce(average, [1.5, 3])
2.25
```

Fold / Reduce / Aggregate



```
>>> average = lambda x, y: (x + y) / 2.
>>> reduce(average, [1, 2, 3])
2.25
>>> reduce(average, [3, 2, 1])
>>> reduce(average, [2.5, 1])
1.75
```

MapReduce



```
>>> reduce(operator.add, map(lambda x: x*x,
[1, 2, 3, 4]))
30
```

IBM®

Google

twitter

Linkedin

Yandex

Microsoft

ORACLE®

facebook®

Distributed Shell

Distributed Shell

```
$ grep <pattern> <file>
```

Distributed Shell

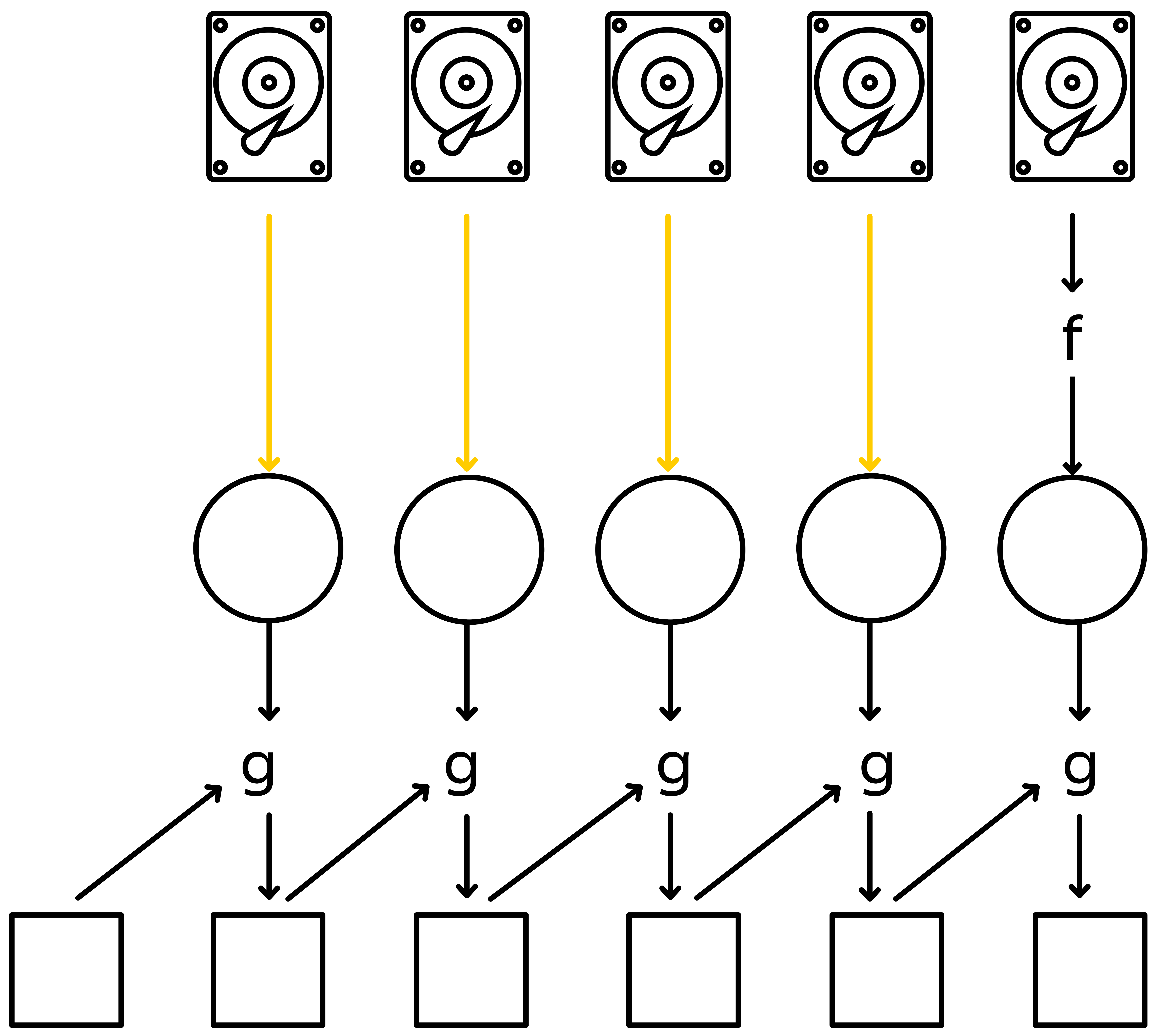
```
$ grep <pattern> <file>
$ grep "hadoop" A.txt
```

Repository git-wip-us.apache.org/repos/asf/**hadoop**.git
Website **hadoop**.apache.org

Distributed Shell

```
$ grep <pattern> <file>
$ grep "hadoop" A.txt
Repository git-wip-us.apache.org/repos/asf/hadoop.git
Website  hadoop.apache.org
$ grep -i "hadoop" A.txt
Apache Hadoop
Apache Hadoop
Hadoop Logo
Repository git-wip-us.apache.org/repos/asf/hadoop.git
Website  hadoop.apache.org
Apache Hadoop ( /həˋdu:p/) is
$ man grep
```

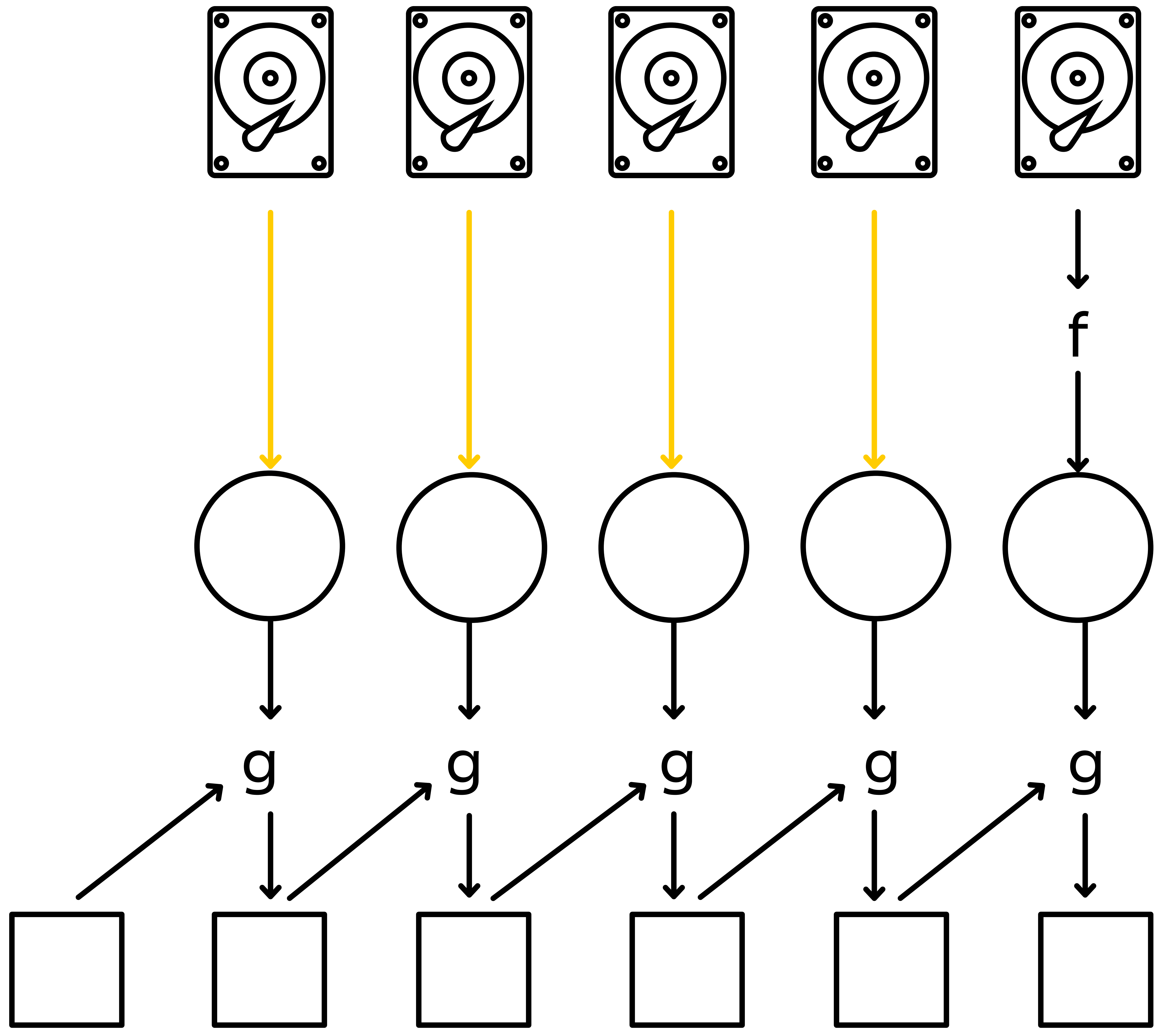
Distributed Shell: **grep**



map = ??

reduce = ??

Distributed Shell: **grep**



map = grep

reduce = None

Distributed Shell

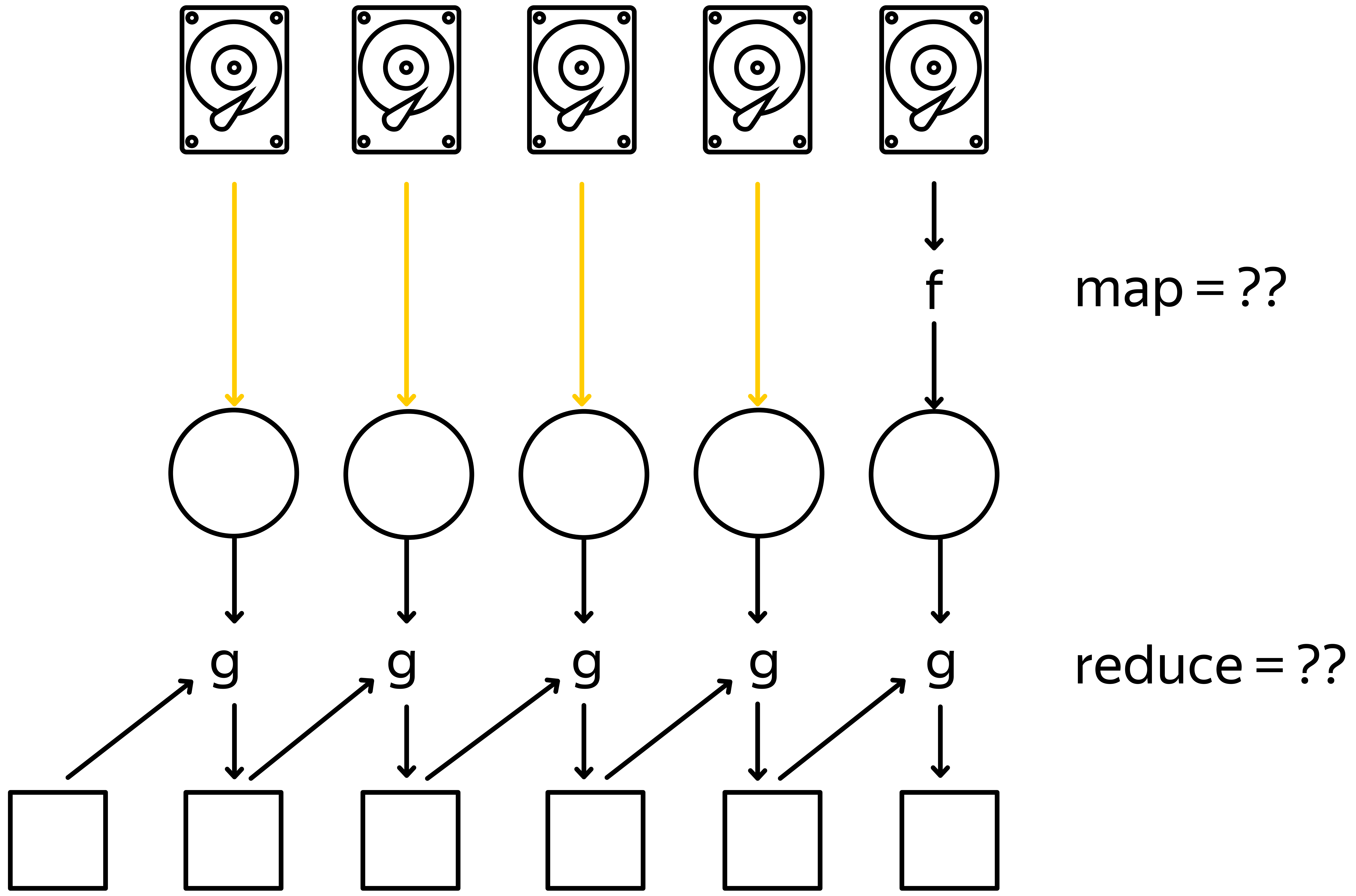
```
$ head <file>
```

Distributed Shell

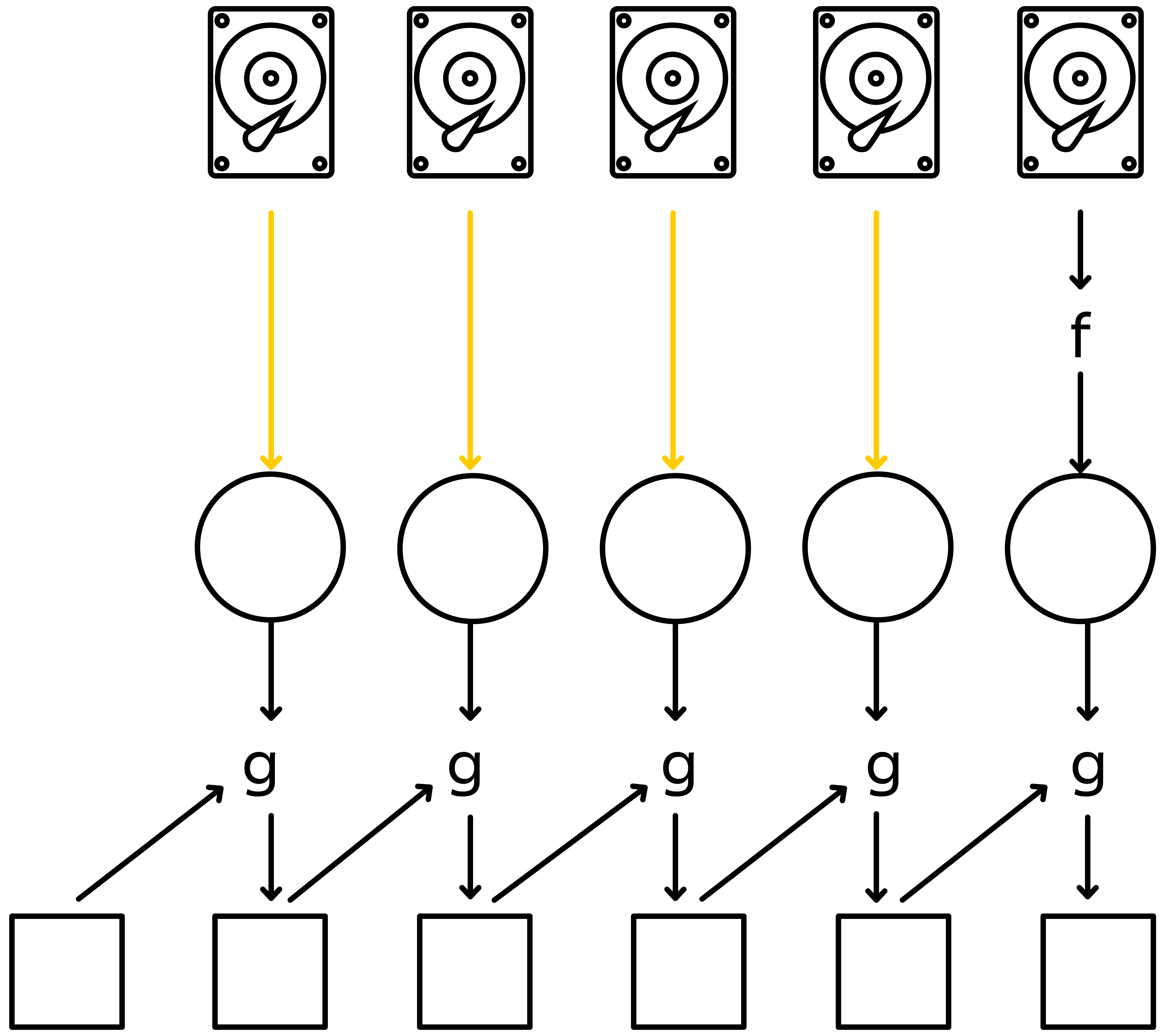
```
$ head <file>
$ head A.txt
```

```
Apache Hadoop
From Wikipedia, the free encyclopedia
[hide]This article has multiple issues. Please help improve it or discuss these is-
sues on the talk page. (Learn how and when to remove these template messages)
This article contains content that is written like an advertisement. (October 2013)
This article appears to contain a large number of buzzwords. (October 2013)
This article may be too technical for most readers to understand. (May 2017)
Apache Hadoop
Hadoop Logo
Developer(s)Apache Software Foundation
```

Distributed Shell: **head**



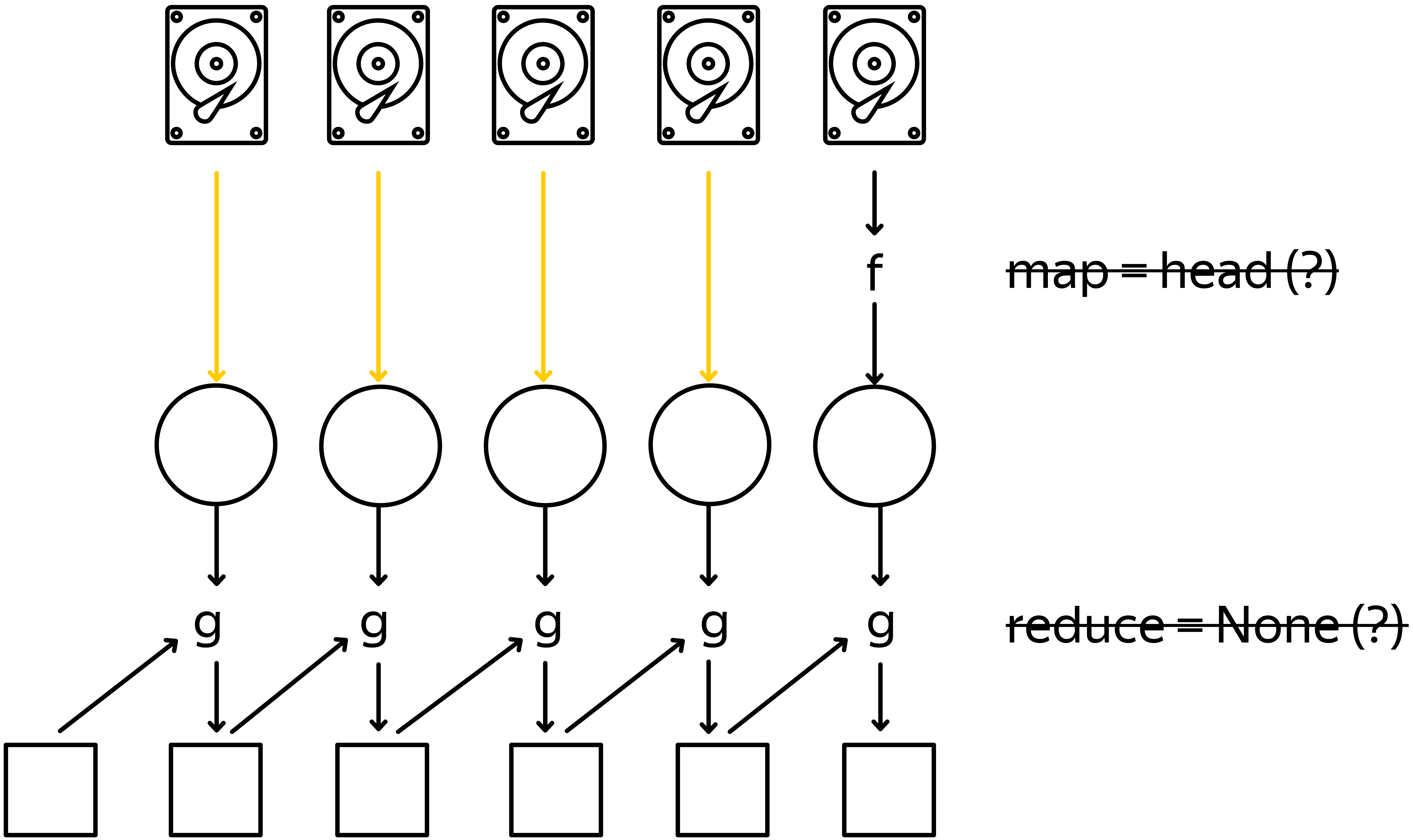
Distributed Shell: **head**



map = head (?)

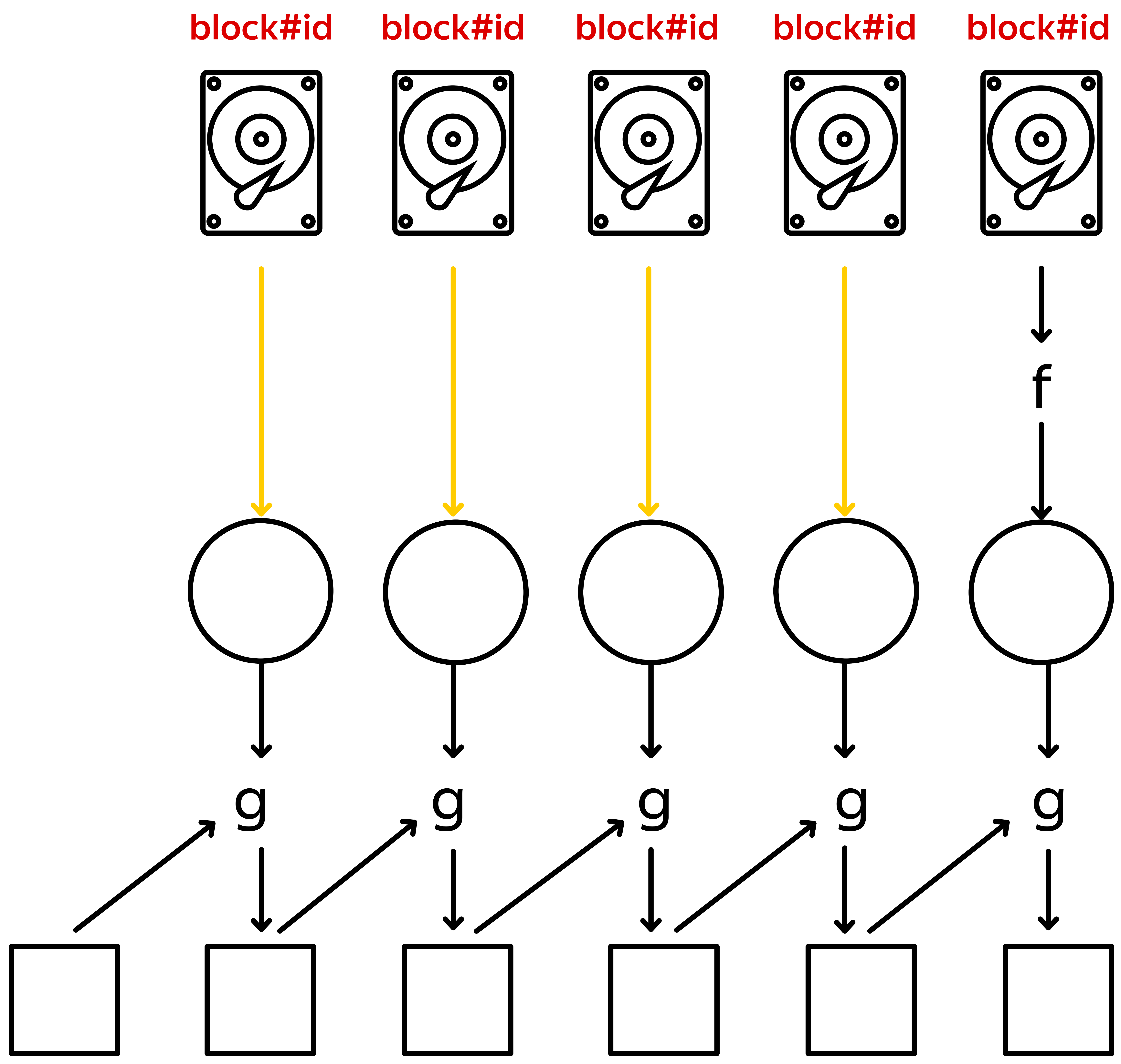
reduce = None (?)

Distributed Shell: **head**



```
hdfs dfs -text distributed_A.txt | head
```

Distributed Shell: **head**



map = head
+ tweaks*

reduce = None

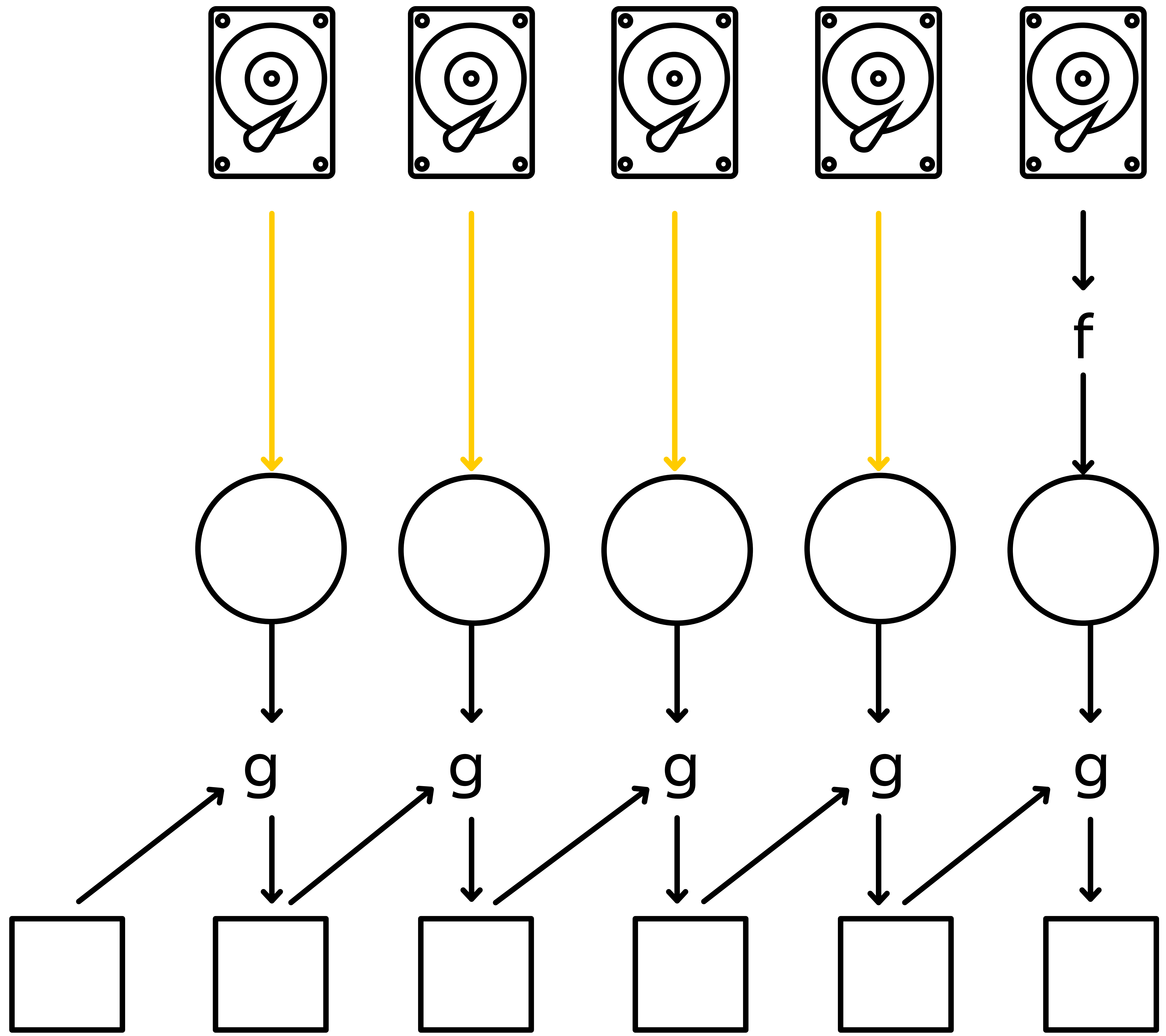
Distributed Shell

```
$ wc <file>
```

Distributed Shell

```
$ wc <file>
$ wc A.txt
269  4319  28001  A.txt
```

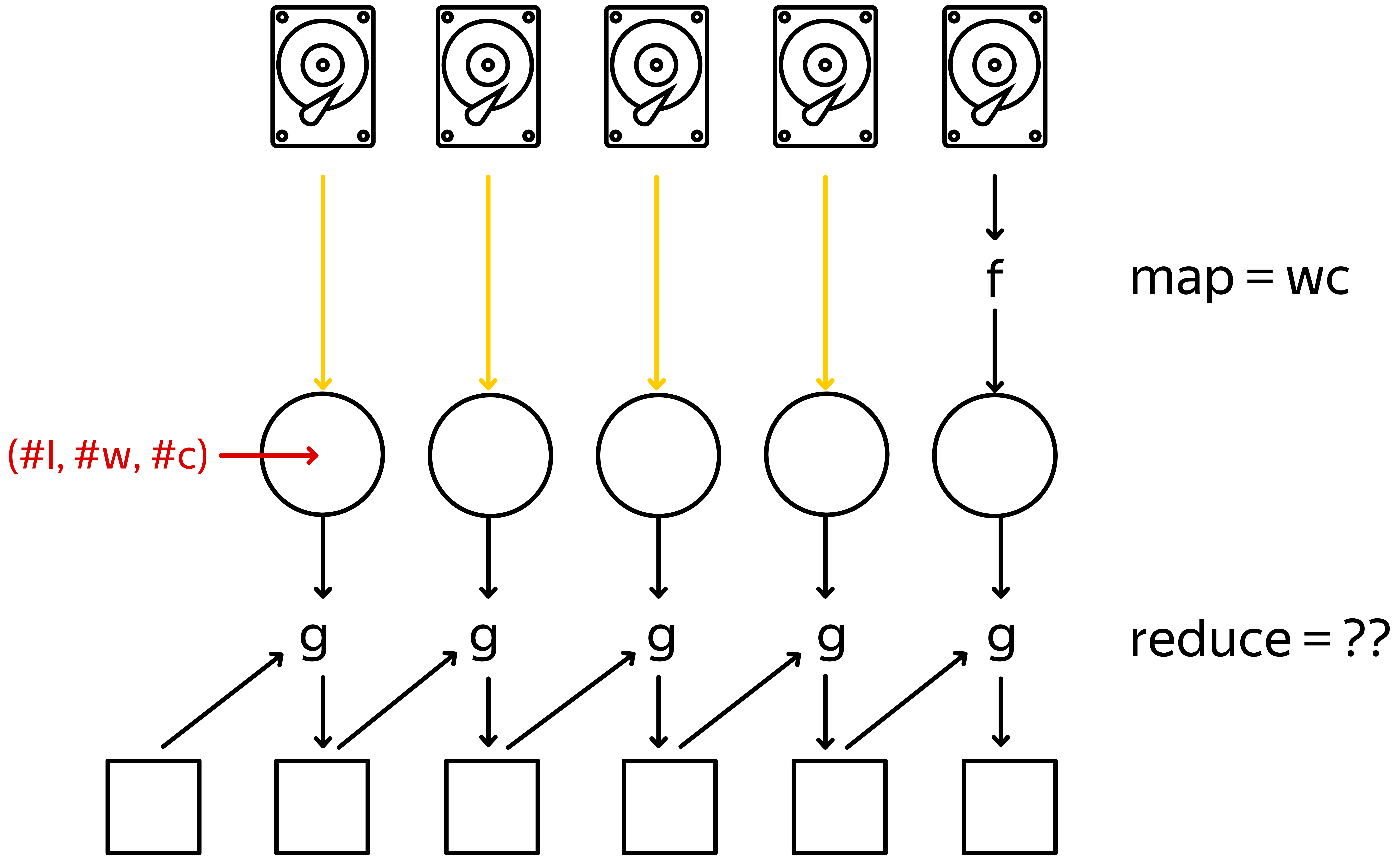
Distributed Shell: **wc**



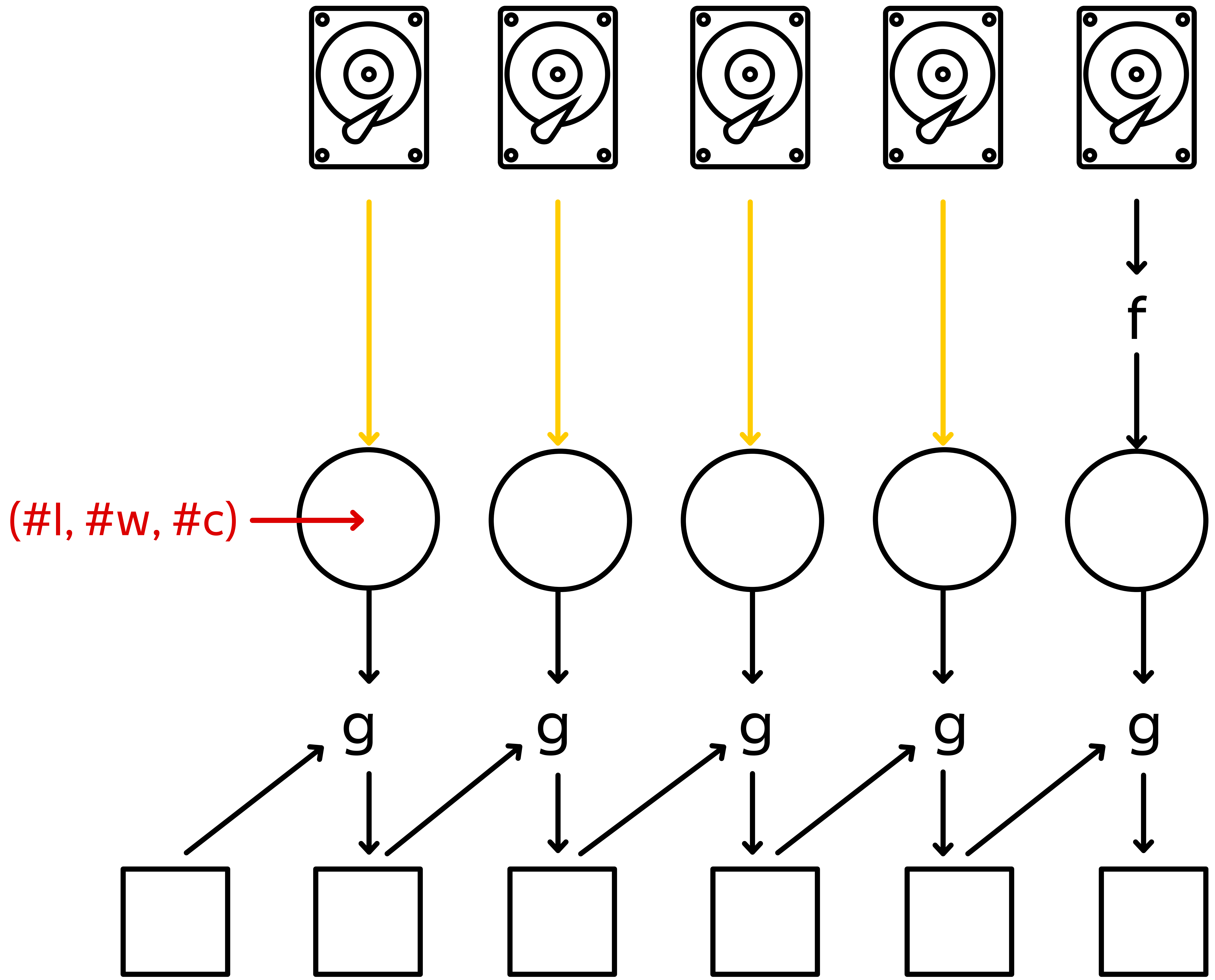
map = wc

reduce = ??

Distributed Shell: **wc**



Distributed Shell: **wc**



map = wc

reduce = operator.add for tuples

Distributed Shell

distributed grep: (map=grep) + (reduce=None)
distributed head: (map=head*) + (reduce=None)
distributed wc: (map=wc) + (reduce=operator.add*)

World Count

Apache Hadoop (/hə`du:p/) is an open-source software framework used for distributed storage and processing of dataset of big data using the MapReduce programming model. It consists of computer clusters built from commodity hardware.

All the modules in Hadoop are designed with a fundamental assumption that hardware failures are common occurrences and should be automatically handled by the framework…

↓

```
'the': 3, 'of': 3, 'hadoop': 2, …
```

World Count

```
one computer: cat * | tr ' ' '\n' | sort | uniq -c
```

World Count

```
distributed: cat * | tr ' ' '\n'  | sort | uniq -c
```

World Count

```
distributed: cat * | tr ' ' '\n'  | sort | uniq -c
```

~~map=sort~~

World Count

```
distributed: cat * | tr ' ' '\n'  | sort | uniq -c
```

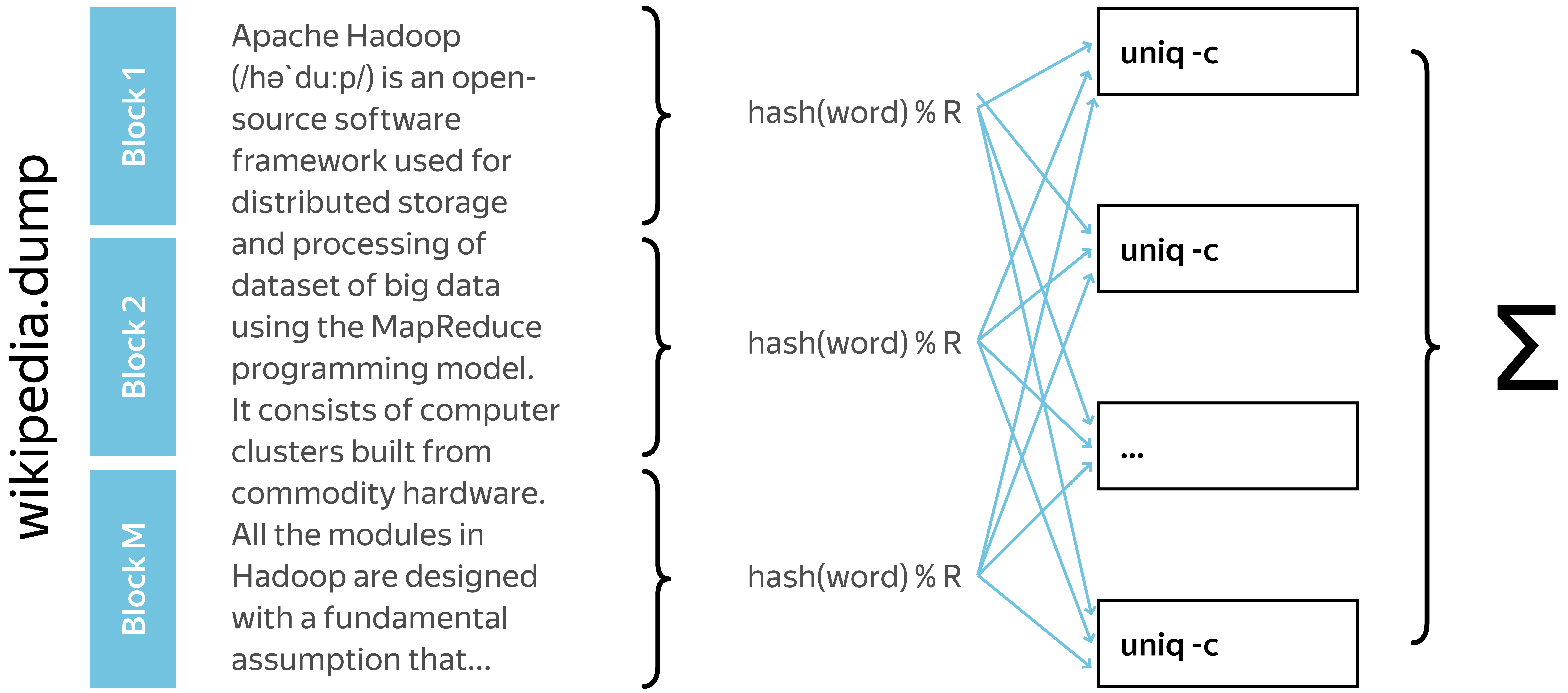
~~map=sort~~

~~reduce=sort~~ (doesn’t feat in Memory / Disk)

Map → Shuffle & Sort → Reduce

MapReduce (example)

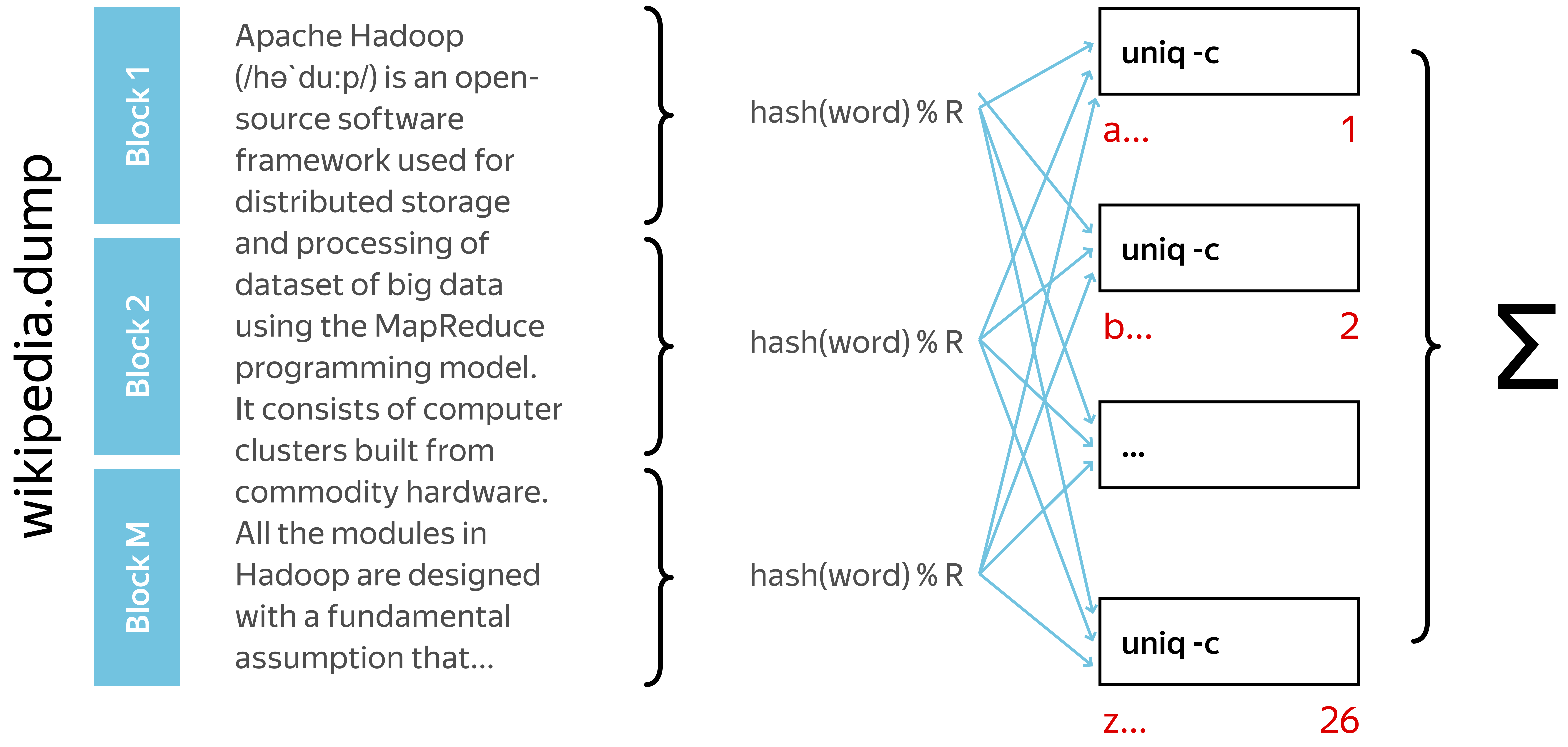
```
cat wikipedia.dump | tr ' ' '\n'  |    sort      |        uniq -c
```


wikipedia.dump
Block 1
Block 2
Block M
Apache Hadoop (/hə`du:p/) is an open-source software framework used for distributed storage and processing of dataset of big data using the MapReduce programming model. It consists of computer clusters built from commodity hardware. All the modules in Hadoop are designed with a fundamental assumption that...
hash(word) % R
hash(word) % R
hash(word) % R
uniq -c
uniq -c
...
uniq -c
Σ

wikipedia.dump -> map () -> word   shuffle & sort   reduce()

MapReduce (example → WordCount)

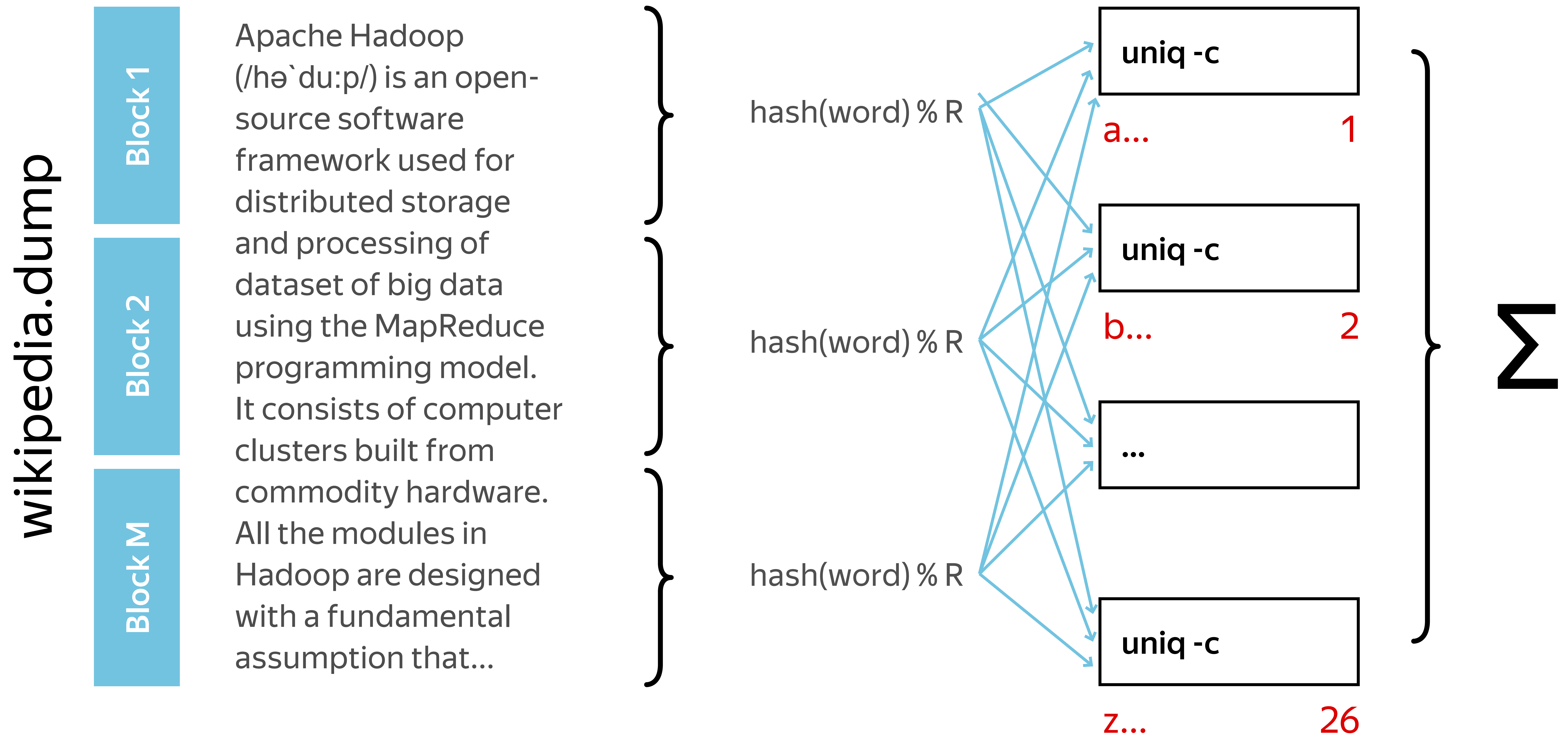
wikipedia.dump | tr ' ' '\n' | sort | uniq -c


wikipedia.dump
Block 1
Block 2
Block M
Apache Hadoop (/hə`du:p/) is an open-source software framework used for distributed storage and processing of dataset of big data using the MapReduce programming model. It consists of computer clusters built from commodity hardware. All the modules in Hadoop are designed with a fundamental assumption that...
hash(word) % R
hash(word) % R
hash(word) % R
uniq -c
a... 1
uniq -c
b... 2
...
uniq -c
z... 26
Σ

wikipedia.dump -> map () -> word   shuffle & sort   reduce()

MapReduce (example → WordCount)

wikipedia.dump | tr ' ' '\n' | sort | uniq -c



wikipedia.dump
Block 1
Block 2
Block M
Apache Hadoop (/hə`du:p/) is an open-source software framework used for distributed storage and processing of dataset of big data using the MapReduce programming model. It consists of computer clusters built from commodity hardware. All the modules in Hadoop are designed with a fundamental assumption that...
hash(word) % R
hash(word) % R
hash(word) % R
uniq -c
a... 1
uniq -c
b... 2
...
uniq -c
z... 26
Σ

wikipedia.dump -> map () -> word   shuffle & **sort**   reduce()

MapReduce Formal Model

MapReduce Formal Model

map: (key, value) → (key, value)
reduce: (key, value) → (key, value)

WordCount example

```
$ cat -n wikipedia.dump | tr ' ' '\n'|
sort | uniq -c
```

- cat **-n** wikipedia.dump: [(line_no, line), ...]

WordCount example

```
$ cat -n wikipedia.dump | tr ' ' '\n'|
sort | uniq -c
```

› cat **-n** wikipedia.dump: [(line_no, line), ...]

› tr ' ' '\n': (-, line) —> [(word, 1), ...]

WordCount example

```
$ cat -n wikipedia.dump | tr ' ' '\n'|
sort | uniq -c
```

- cat **-n** wikipedia.dump: [(line_no, line), ...]
- tr ' ' '\n': (-, line) —> [(word, 1), ...]
- sort: Shuffle & Sort

WordCount example

```
$ cat -n wikipedia.dump | tr ' ' '\n'|
sort | uniq -c
```

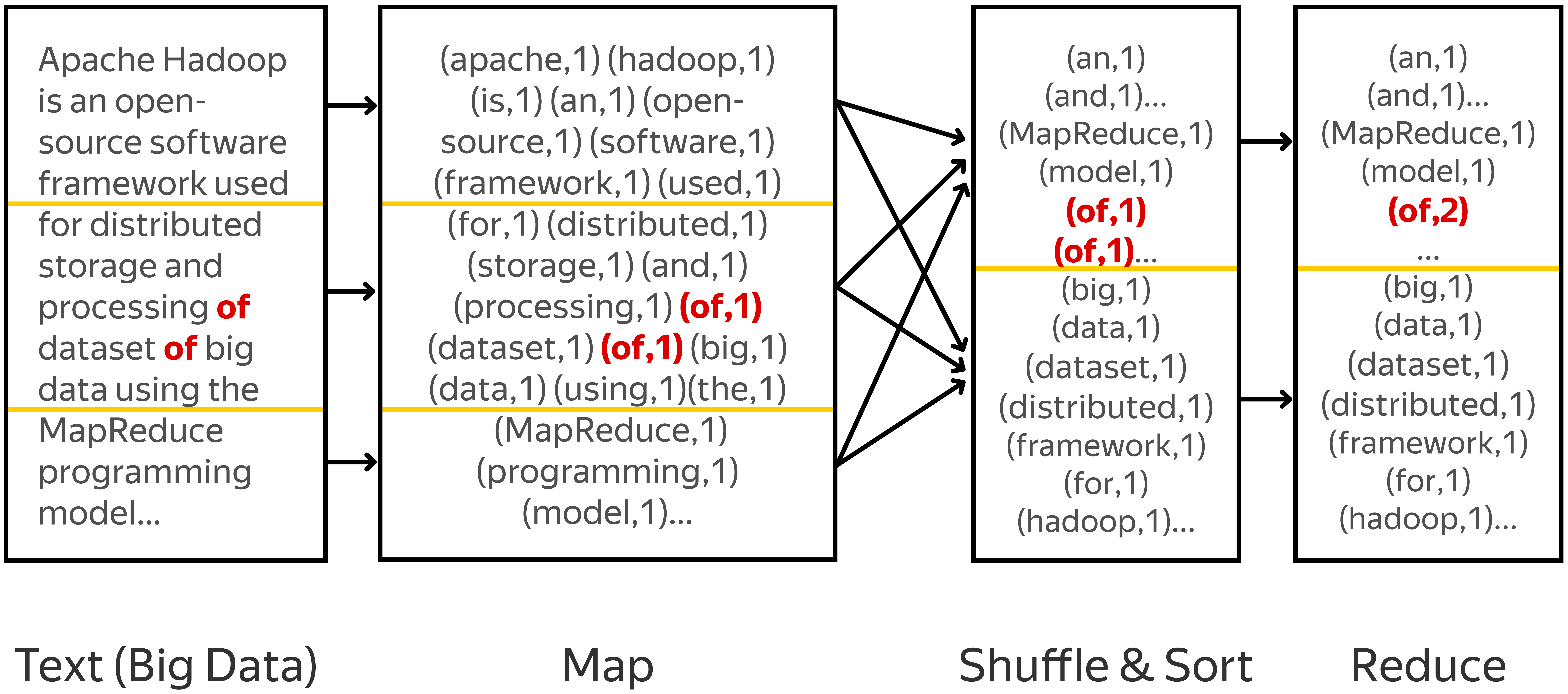
› cat **-n** wikipedia.dump: [(line_no, line), ...]

› tr ' ' '\n': (-, line) —> [(word, 1), ...]

› sort: Shuffle & Sort

› uniq -c: (word, [1, ...]) —> (word, count)

Word Count



Apache Hadoop is an open-source software framework used
for distributed storage and processing of dataset of big data using the
MapReduce programming model...
(apache,1) (hadoop,1) (is,1) (an,1) (open-source,1) (software,1) (framework,1) (used,1)
(for,1) (distributed,1) (storage,1) (and,1) (processing,1) (of,1) (dataset,1) (of,1) (big,1) (data,1) (using,1)(the,1)
(MapReduce,1) (programming,1) (model,1)...
(an,1) (and,1)... (MapReduce,1) (model,1) (of,1) (of,1)...
(big,1) (data,1) (dataset,1) (distributed,1) (framework,1) (for,1) (hadoop,1)...
(an,1) (and,1)... (MapReduce,1) (model,1) (of,2) ...
(big,1) (data,1) (dataset,1) (distributed,1) (framework,1) (for,1) (hadoop,1)...
Text (Big Data)
Map
Shuffle & Sort
Reduce

WordCount example

```
$ cat -n wikipedia.dump | tr ' ' '\n'|
sort | uniq -c
```

- cat **-n** wikipedia.dump: [(line_no, line), ...]
- **read: [(k_in, v_in), ...]**

WordCount example

```
$ cat -n wikipedia.dump | tr ' ' '\n'|
sort | uniq -c
```

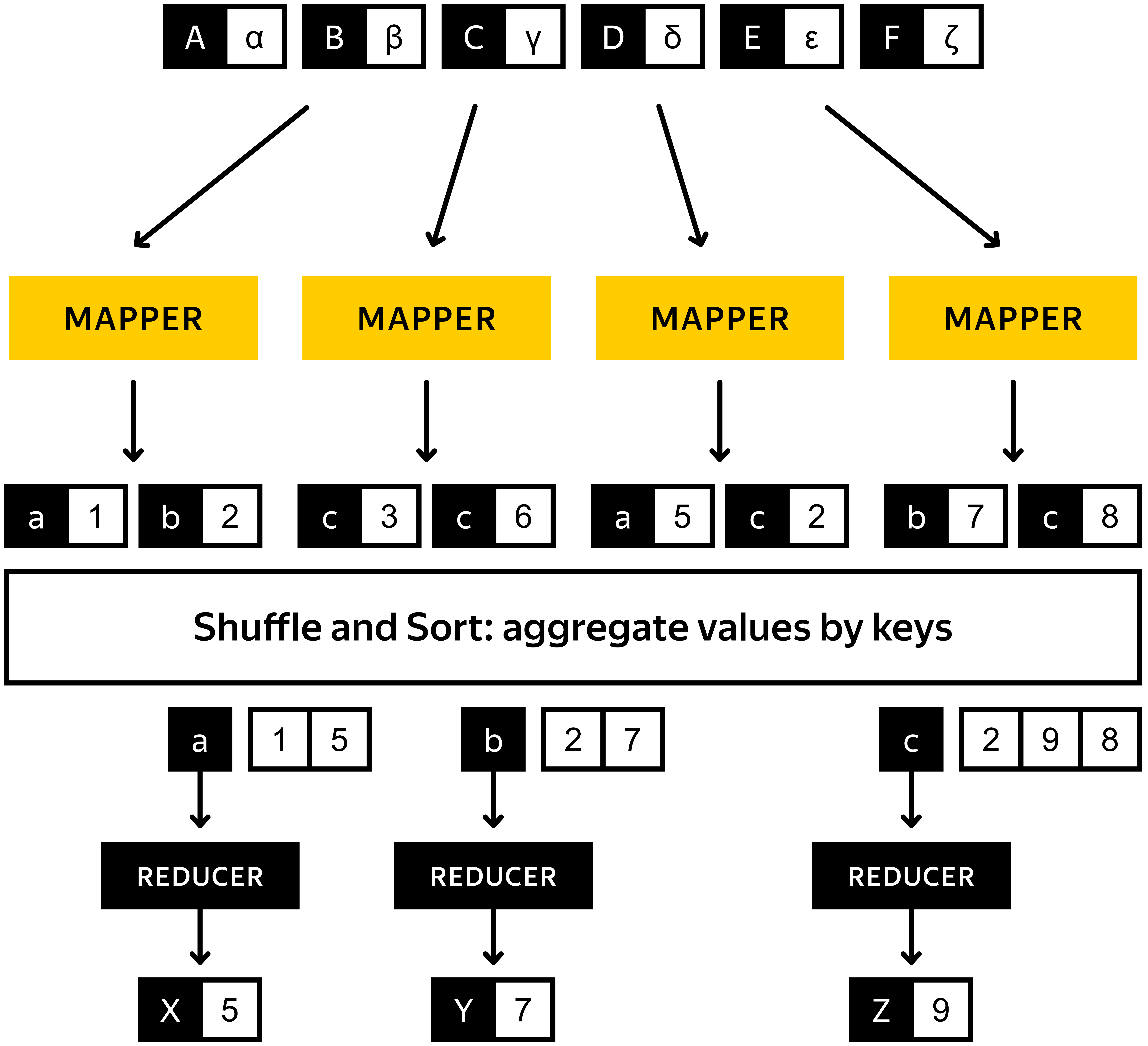
- cat **-n** wikipedia.dump: [(line_no, line), ...]
- **read: [(k_in, v_in), ...]**
- tr ‘ ‘ ‘\n’: (-, line) —> [(word, 1), ...]
- **map: (k_in, v_in) —> [(k_interm, v_interm), ...]**

WordCount example

```
$ cat -n wikipedia.dump | tr ' ' '\n'|
sort | uniq -c
```

- cat **-n** wikipedia.dump: [(line_no, line), ...]
- **read: [(k_in, v_in), ...]**
- tr ' ' '\n': (-, line) —> [(word, 1), ...]
- **map: (k_in, v_in) —> [(k_interm, v_interm), ...]**
- **Shuffle & Sort: sort and group by k_interm**
- uniq -c: (word, [1, ...]) —> (word, count)
- **reduce: (k_interm, [(v_interm, ...)]) —> [(k_out, v_out), ...]**

MapReduce



- read: [(k_in, v_in), ...]
- map: (k_in, v_in) —> [(k_interm, v_interm), ...]
- Shuffle & Sort: sort and group by k_interm
- reduce: (k_interm, [(v_interm, ...)]) —> [(k_out, v_out), ...]

MapReduce

- You **know** the phases of MapReduce: Map, Shuffle & Sort, Reduce;
- You **know** how to solve simple tasks such as distributed “grep”, “head”, “wc” and “Word Count” with MapReduce.

BigDATAteam