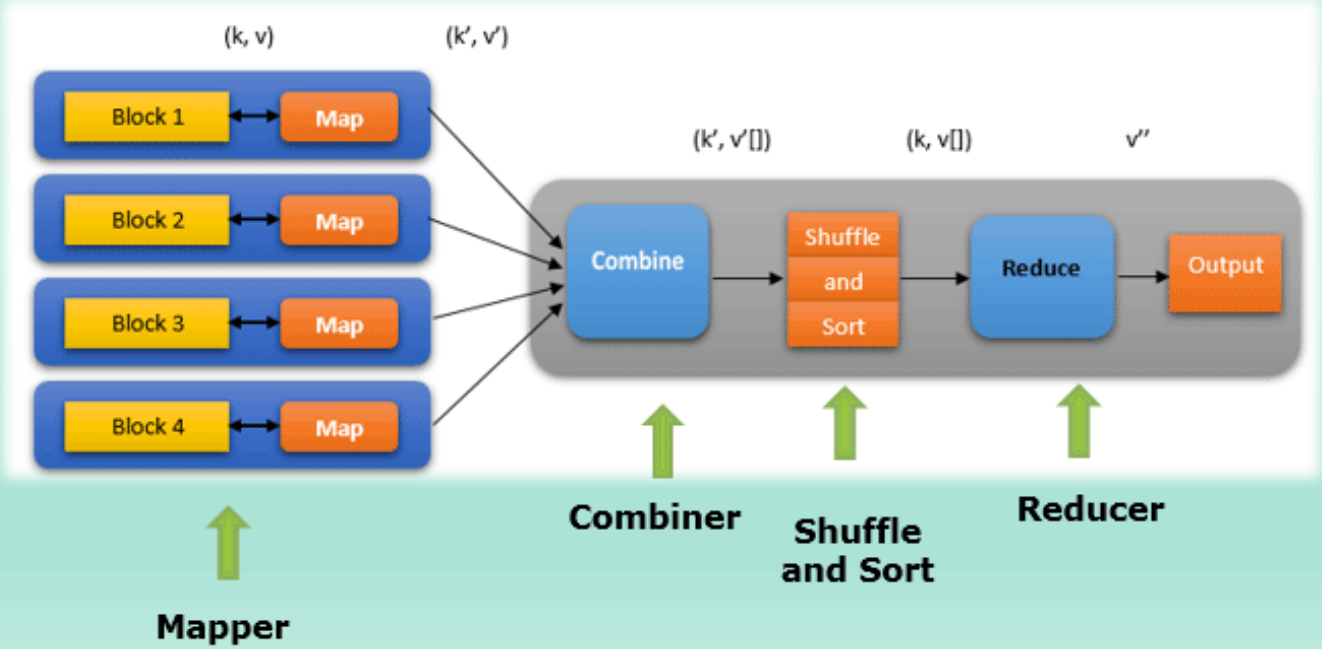
Fundamentals of Big Data Analytics

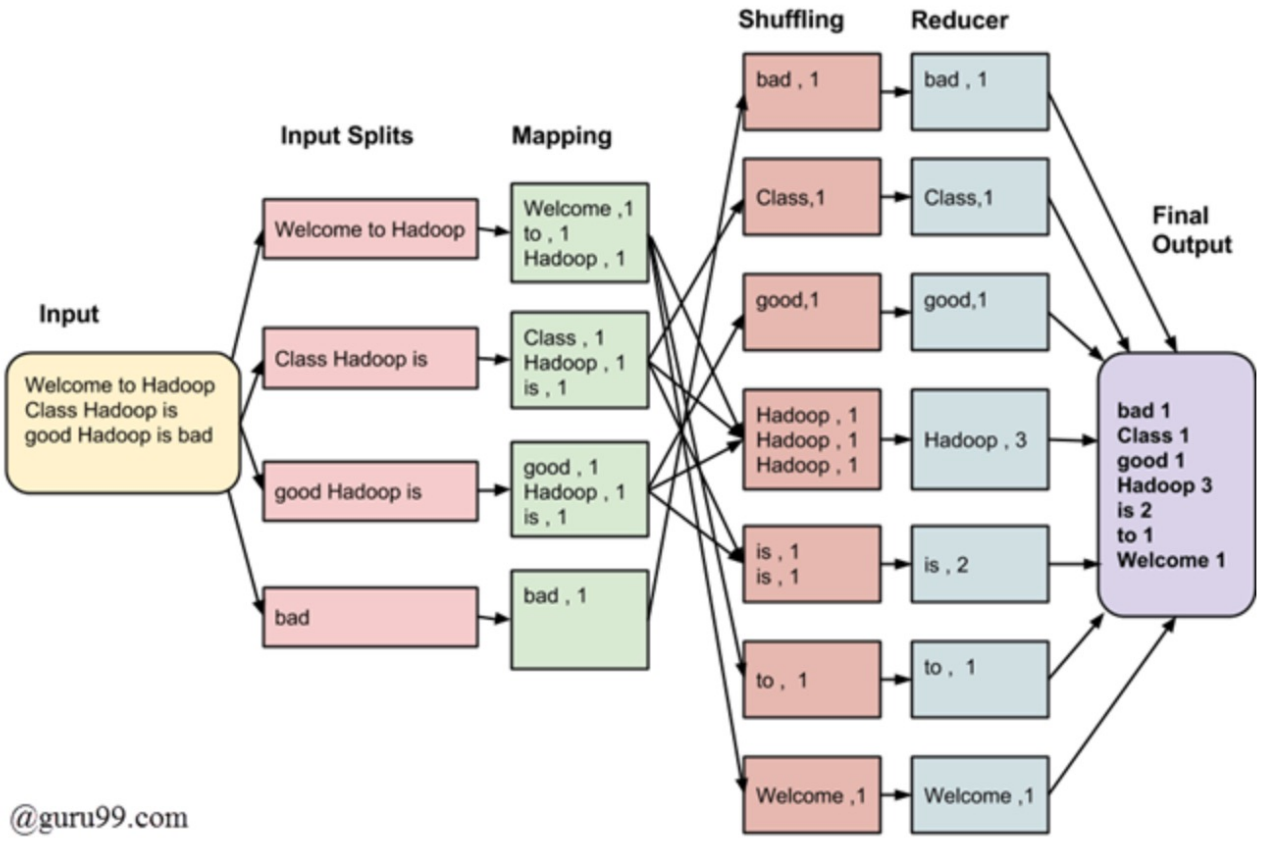
Lecture 17 – Introduction to MapReduce

Dr. Iqra Safder
Assistant Professor
FAST NUCES, Lahore

How MapReduce Works



www.educba.com



MapReduce Architecture

Example:
Vector Multiplication

Vector Multiplication

- Task: multiply 2 arrays of *N* numbers
  - A basic mathematical operation
  - Let's assume *N* is very large

Vector Multiplication

- Task: multiply 2 arrays of *N* numbers

A X	***B***
5	2.7
4	1.9
3.2	1.3
.	.
.	.
.	.
-2	1

=

(5x 2.7) # 1st of *A* & *B*

+ (4x 1.9) # 2nd of *A* & *B*

+ (3.2x 1.3) # 3rd ...

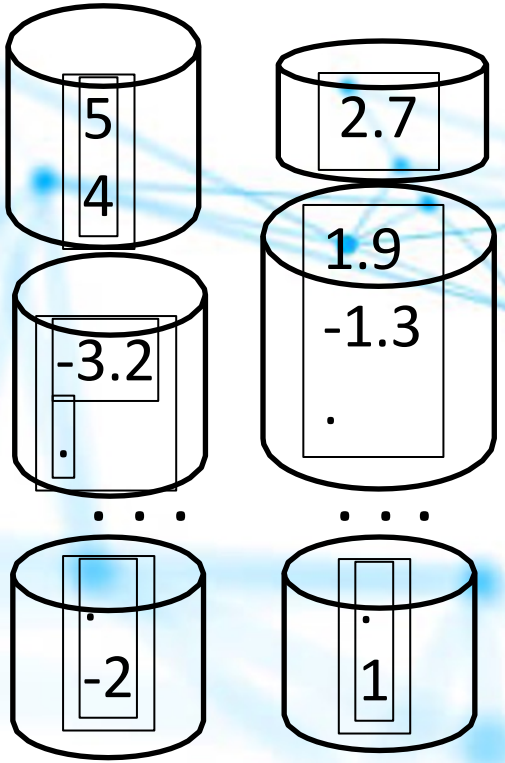
.

.

.

+ (−2x 1) # Nth of *A* & *B*

Vector Multiplication



- Recall:
  - data partitioned in HDFS

Vector Multiplication

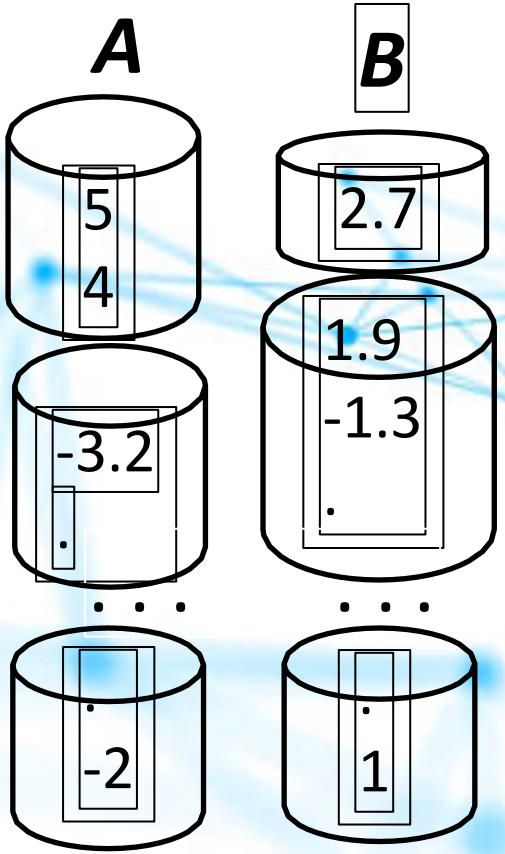
- Main design consideration:

need elements with same index together

Let <key, value> =

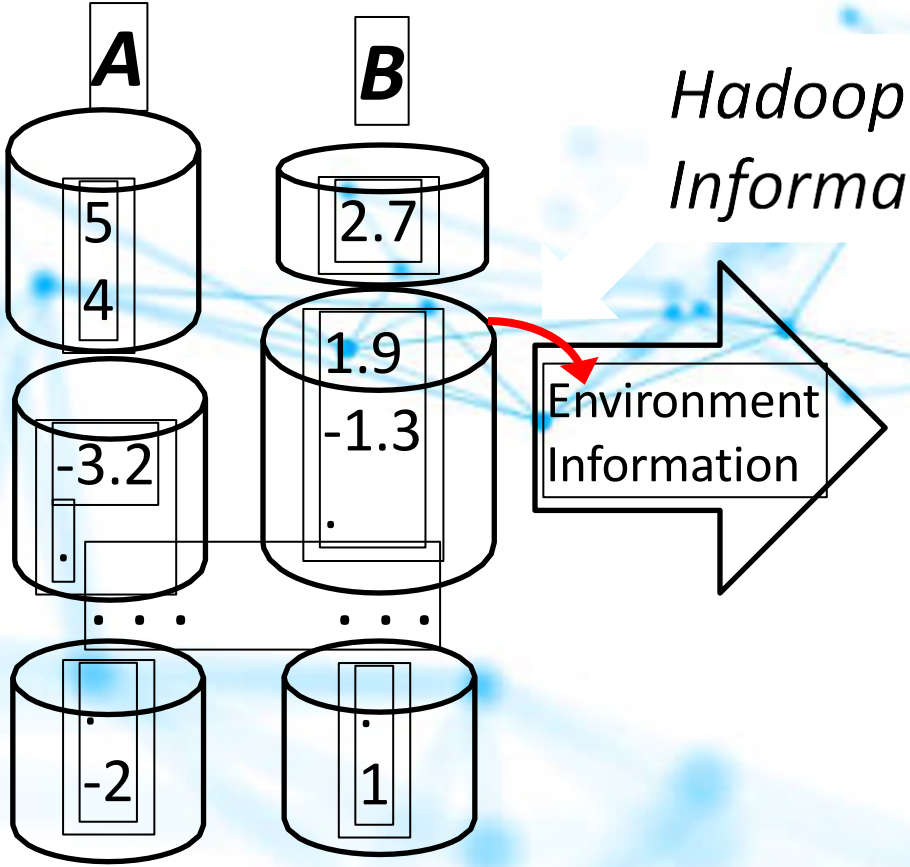
<index, number>

Vector Multiplication



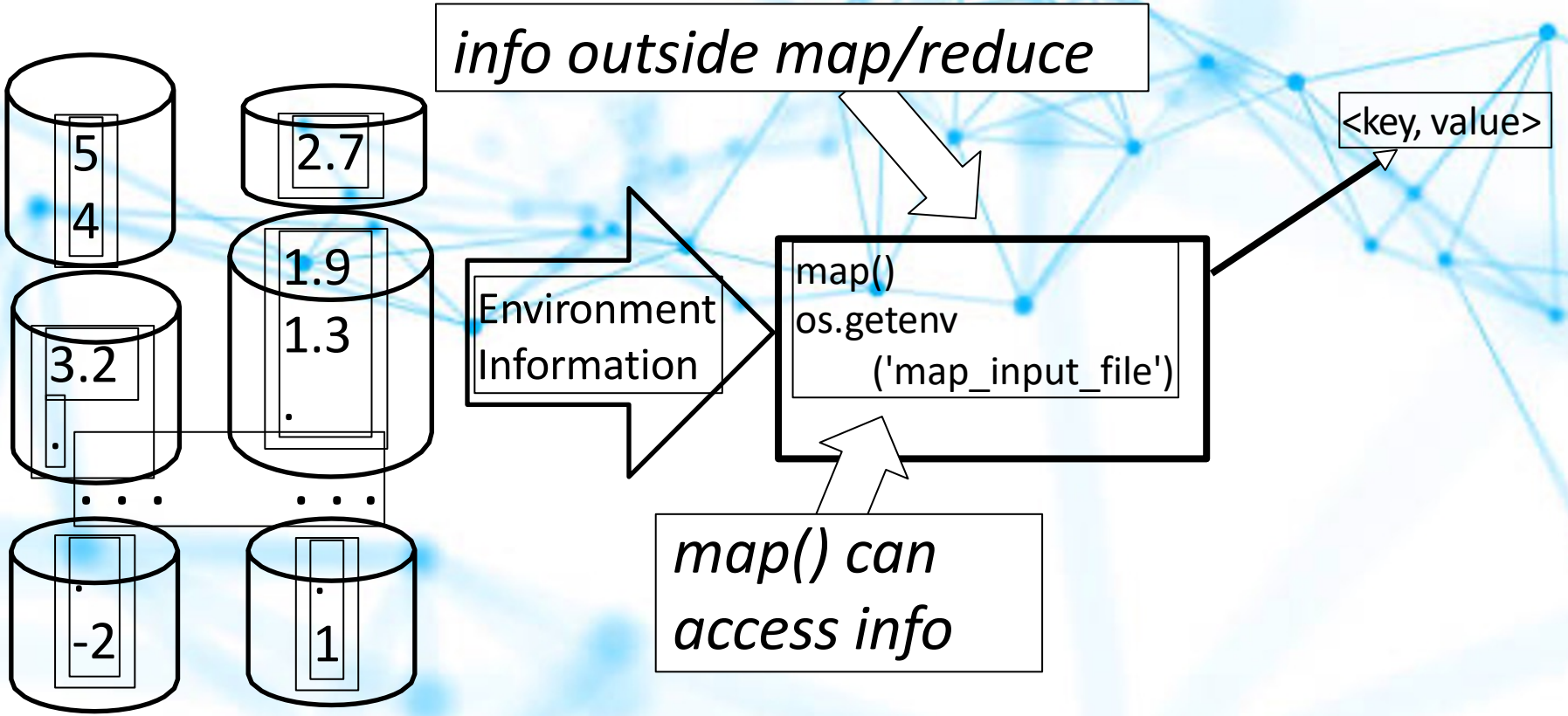
- Problem: array partitions don't have an index

Vector Multiplication

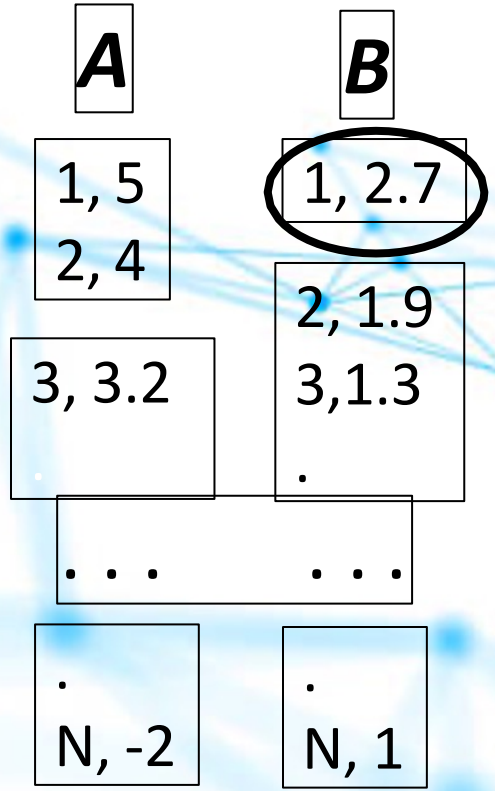


Hadoop makes environmental Information available

Vector Multiplication

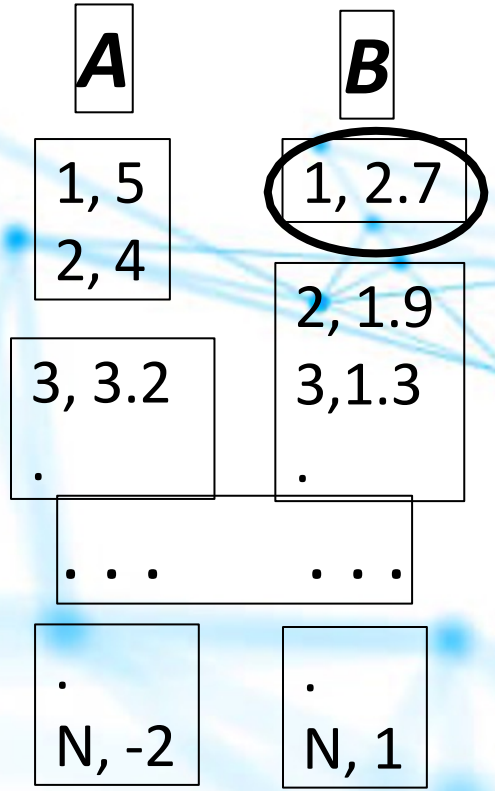


Vector Multiplication



- Let's assume:
  - each line already has <index, number>

Vector Multiplication



- Let's assume:
  - each line already has <index, number>

Note: mapper only needs to pass data (identity function)

Vector Multiplication

A	***B***
<index, num>	<index, num>
1, 5	1, 2.7
2, 4	
3, 3.2	2, 1.9
.	3,1.3
	.
. . .	. . .
.	.
N, -2	N, 1

shuffle &

group indices

<index, num>

1, 5

1, 2.7

3, 1.3

3, 3.2

2, 1.9

2, 4

...

Vector Multiplication

1, 5
1, 2.7
3, 1.3
3, 3.2

2, 1.9
2, 4

. . .

What should reducers do?

Vector Multiplication

A,B grouped

1, 5
1, 2.7
3, 1.3
3, 3.2

2, 1.9
2, 4

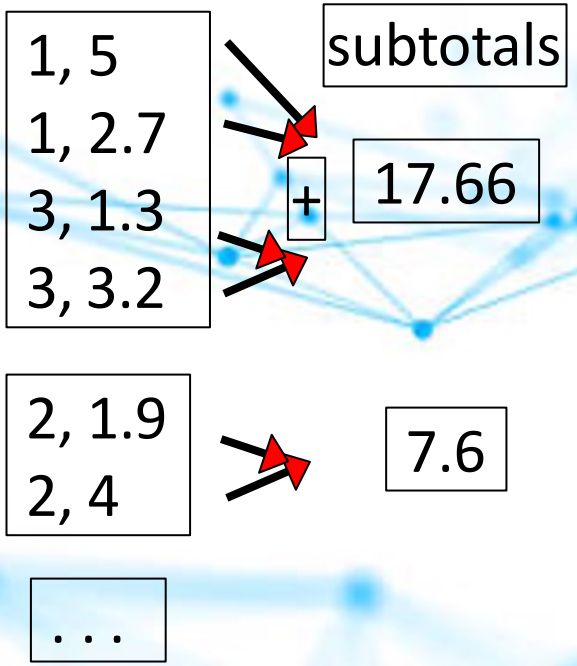
. . .

Reducer:

-get pairs of

<index, number>

Vector Multiplication



Reducer:

-get pairs of <index, number>

-multiply & add

Vector Multiplication

1, 5
1, 2.7
3, 1.3
3, 3.2

subtotals

+ 17.66

2, 1.9
2, 4

7.6

. . .

Reducer:

-get pairs of <index, number>

-multiply & add

(Still need get total sum, but should be largely reduced)

Computational Costs

In Vector Multiplication

Computational Costs

- For Vector Multiplication
  - *How many <index, number> are output from map()?*

Computational Costs

- For Vector Multiplication
  - *How many <index, number> are output from map()?*
  - *How many <index> groups have to be shuffled?*

Computational Costs

- How many <index, number> are output ?

A	***B***
1, 5	1, 2.7
2, 4	2, 1.9
3, -3.2	3, -1.3
.	.
.	.
.	.
N, -2	N, 1

For: 2 Vectors with
N indices each
Then:
2*N* <index, number>
are output from map()

Computational Costs

- How many <index> groups have to be shuffled?

A,B grouped

<index, num>

1, 5
1, 2.7
3, -1.3
3, -3.2

2, 1.9
2, 4
…

For: $2N$ indices and
N pairs
Then:
N groups are shuffled to reducers

Computational Costs

- Can we reduce shuffling?

Computational Costs

- Can we reduce shuffling?

- Try: ‘combine’ map indices in mapper
  (works better for Wordcount)

Computational Costs

- Can we reduce shuffling?
- Or Try: use index ranges of length R

Computational Costs

- Index Ranges: let R=10 & bin the array indices

1 2 3 4 … 10 11 12 …..19 20 21 …. (N-9) …. N

N keys

Computational Costs

- Index Ranges: let R=10 & bin the array indices

$$\underbrace{1\ 2\ 3\ 4 \ldots 10}_{1}\ \underbrace{11\ 12 \ldots\ldots 19\ 20}_{2}\ \underbrace{21 \ldots\ldots}_{3}\ \ldots\ \underbrace{(N\text{-}9) \ldots\ldots N}_{N/R}$$

place in bins

Computational Costs

- For example, let R=10, and bin the array indices

$$\underbrace{1\ 2\ 3\ 4 \ldots 10}_{1}\ \underbrace{11\ 12 \ldots\ldots 19\ 20}_{2}\ \underbrace{21 \ldots\ldots}_{3}\ \ldots\ \underbrace{(N\text{-}9) \ldots\ldots N}_{N/R}$$

N keys are now *N/R = N/10* keys

Computational Costs

- For example, let R=10, and bin the array indices

$$\underbrace{1\ 2\ 3\ 4 \ldots 10}_{1}\ \underbrace{11\ 12 \ldots\ldots 19\ 20}_{2}\ \underbrace{21 \ldots\ldots}_{3}\ \ldots\ \underbrace{(N\text{-}9) \ldots\ldots N}_{N/R}$$

N keys are now *N/R=N/10* keys

<key,value> is now

<index bin, original-index number>

Computational Costs

- Now shuffling costs depend on N/R groups

If: *R=1*
Then: *N/R=N groups (same as before)*

If: *R>1*
Then: *N/R<N (less shuffling to do)*

Computational Costs

If:

size of (N/R) ↑

Then:

shuffle costs ↑

Computational Costs

If:

size of (N/R) ↑

Then:

shuffle costs ↑

But:

reducer complexity ↓

Computational Costs

If:

size of (N/R) ↑

Then:

shuffle costs ↑

But:

reducer complexity ↓

-you control R (specific tradeoffs depend on data and hardware)

Vector to Matrices

- Matrix multiplication needs row-index and col-index in the keys
- Matrix multiplication more pertinent to data analytic topics

Summary

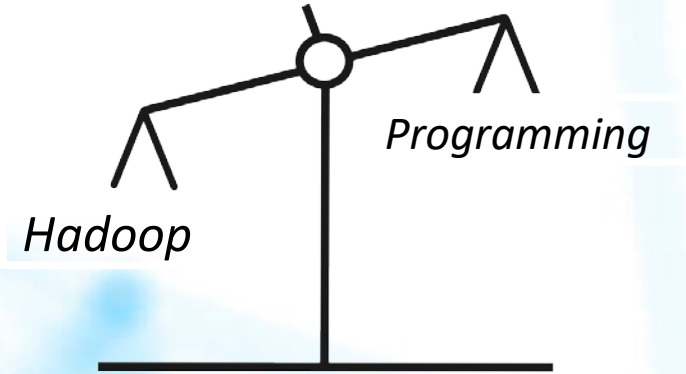
And Looking Beyond

Task Decomposition

- mappers are separate and independent
- mappers work on data parts

Summary of Map/Reduce Design Considerations

- <key, value> must enable correct output
- Let Hadoop do the hard work
- Trade-offs



Summary of Map/Reduce Design Considerations

- Common mappers:
  - Filter (subset data)
  - Identity (just pass data)
  - Splitter (as for counting)

Summary of Map/Reduce Design Considerations

- Composite <keys>

Summary of Map/Reduce Design Considerations

- Composite <keys>
- Extra info in <values>

Summary of Map/Reduce Design Considerations

- Composite <keys>
- Extra info in <values>
- Cascade Map/Reduce jobs

Summary of Map/Reduce Design Considerations

- Composite <keys>
- Extra info in <values>
- Cascade Map/Reduce jobs
- Bin keys into ranges

Summary of Map/Reduce Design Considerations

- Composite <keys>
- Extra info in <values>
- Cascade Map/Reduce jobs
- Bin keys into ranges
- Aggregate map output when possible (combiner option)

Potential Limitations Map/Reduce

- Must fit <key, value> paradigm
- Map/Reduce data not persistent
- Requires programming/debugging
- Not interactive

Beyond Map/Reduce

- Data access tools (Pig, HIVE)
  - SQL like syntax
- Interactivity & Persistency (Spark)