Fundamentals of Big Data Analytics

Lecture 16 – Introduction to MapReduce

Dr. Iqra Safder
Assistant Professor
FAST NUCES, Lahore

Quiz 3 – Thursday, 21st April

Lecture 11 - Lecture 15

Quiz 4 – Thursday, 28th April

Lecture 16 - Lecture 18

The problem:

- Big data means …

lots of hard drives

The solution:

- Lots of data means we should…

bring computation to data!

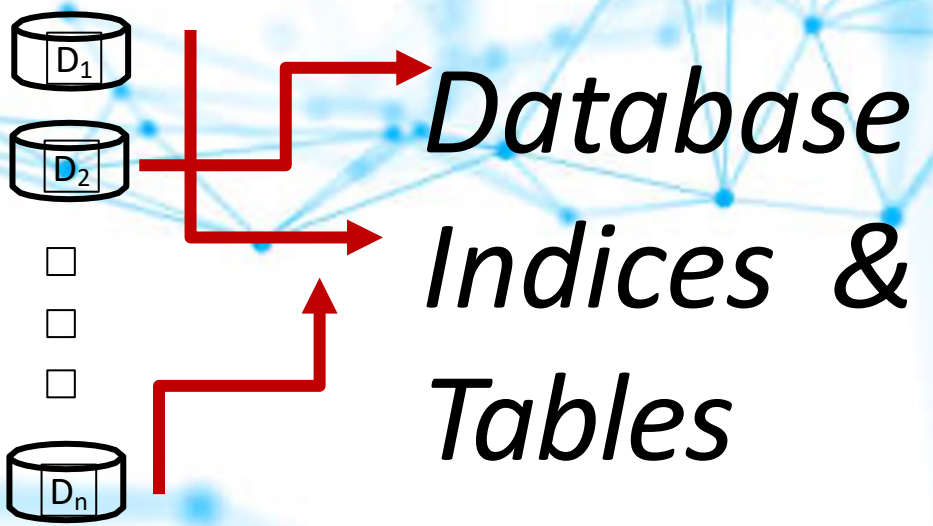
Lots of disks:

Possibilities:

- Case 1: data needs updating

Possibilities:

- Case 1: data needs updating so …

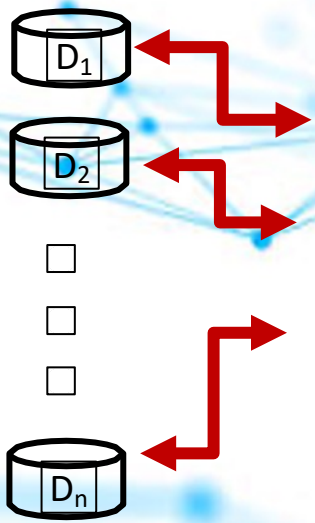


Possibilities:

- Case 2: need to sweep through data

Possibilities:

- Case 2: need to sweep through data so…

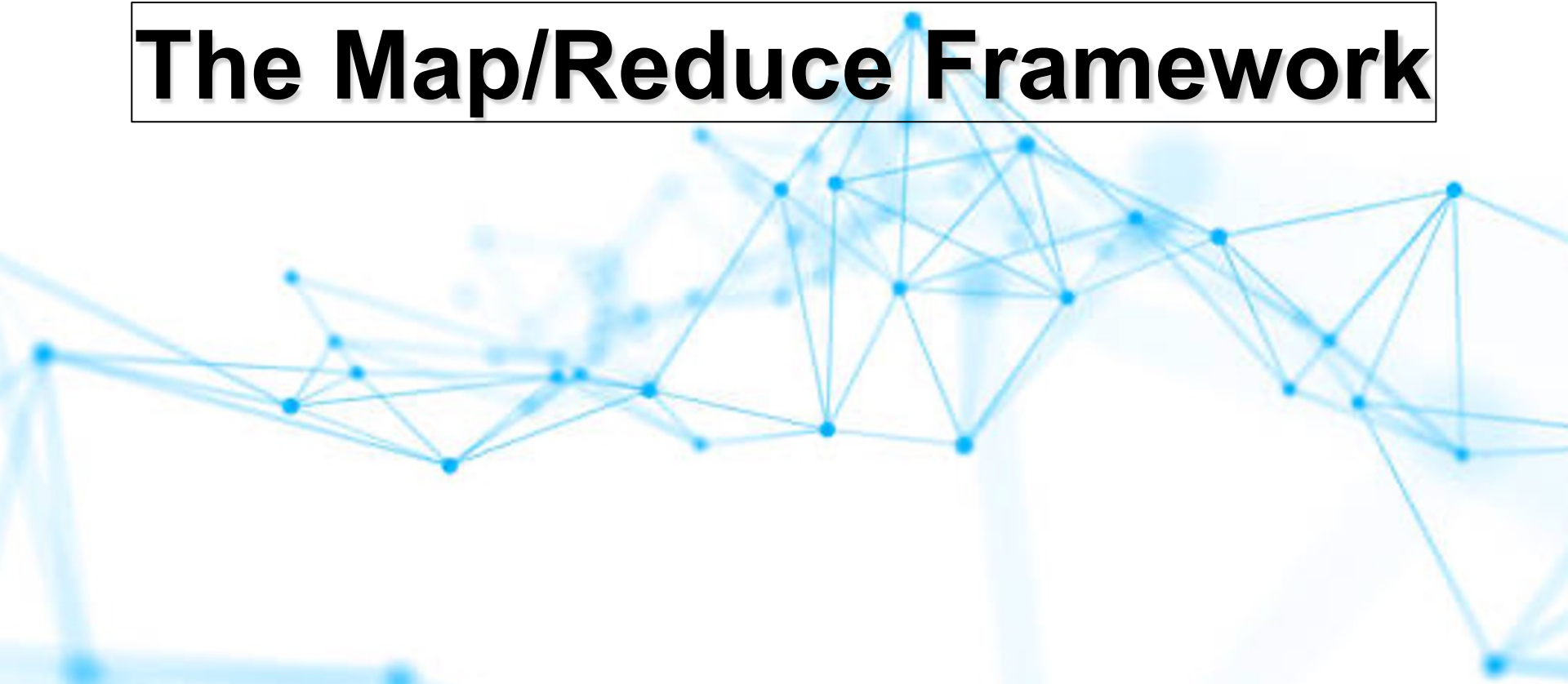


take

Computation

to data

The Map/Reduce Framework



The framework:

- User defines:
  a. <key, value>
  b. mapper & reducer functions
- Hadoop handles the logistics

The logistics:

- Hadoop handles the distribution and execution

Map/Reduce flow

- map() reads data and outputs <key,value>

Map/Reduce flow

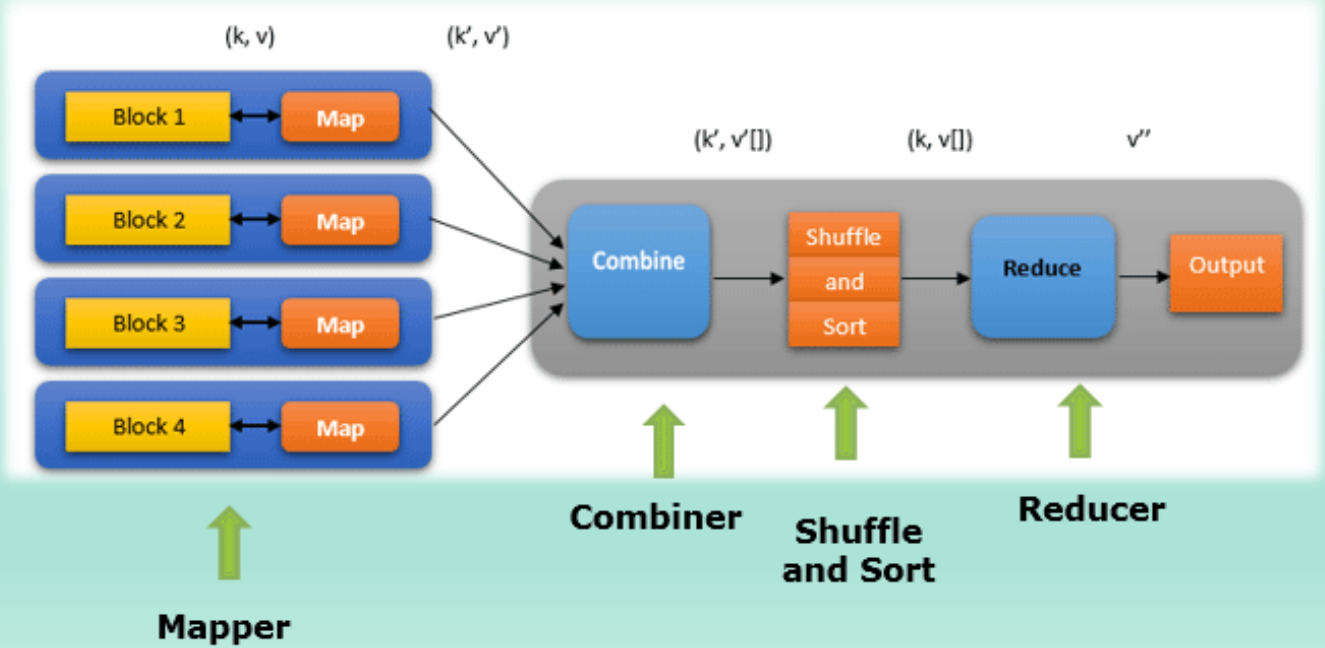
- User defines a reduce function

reduce()

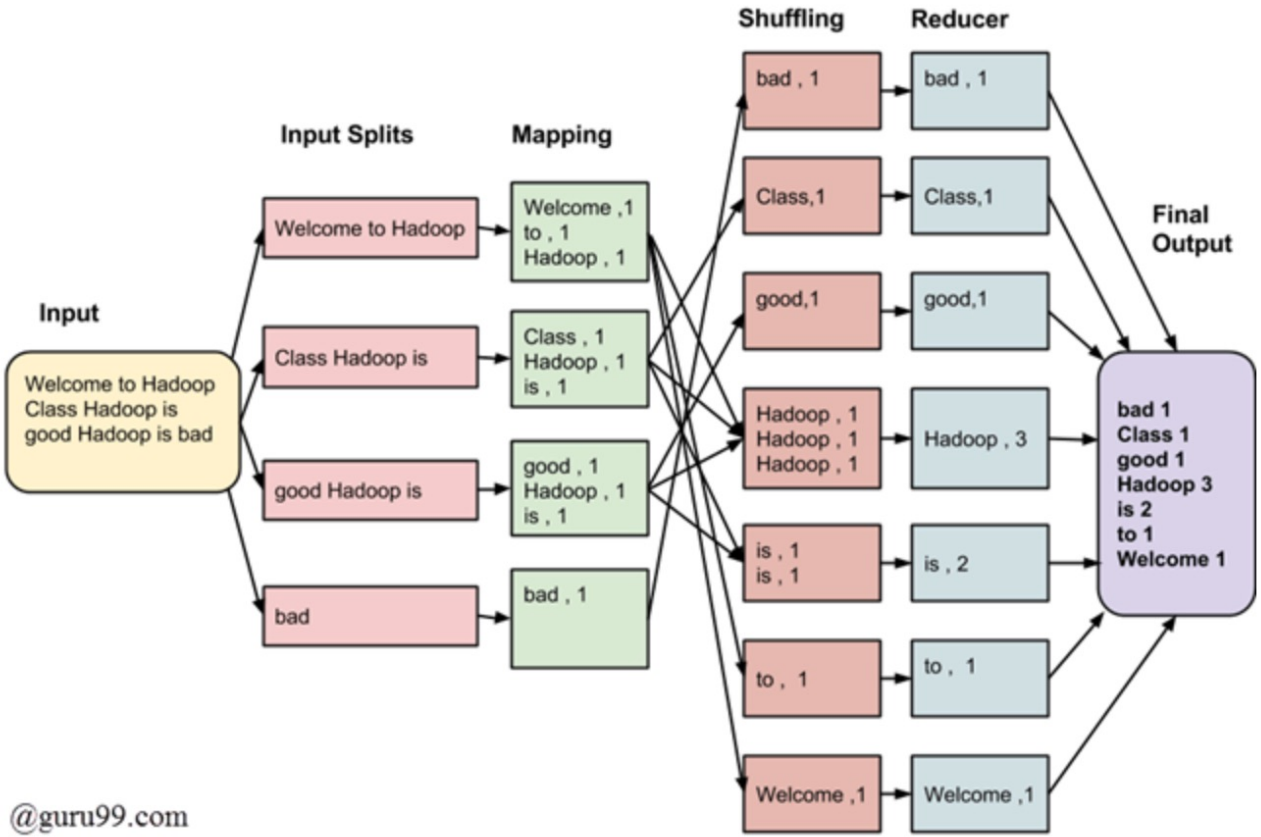
Map/Reduce flow

- reduce() reads <key,value> and outputs your result

How MapReduce Works



www.educba.com



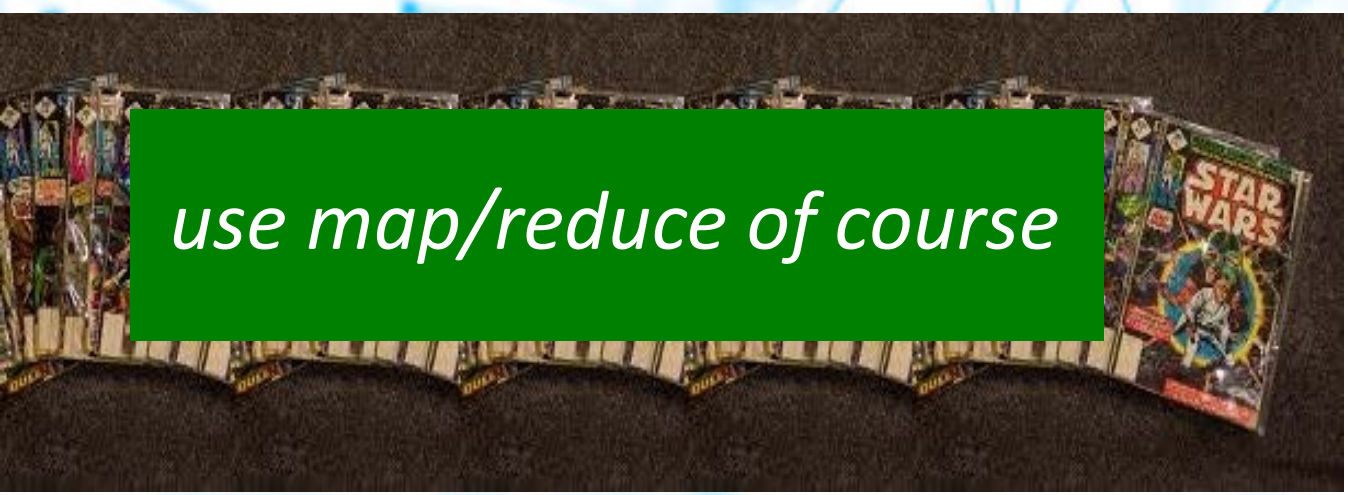
MapReduce Architecture

Wordcount task:

- **… books, blogs, and fan-fiction?**

Wordcount task:

- **… books, blogs, and fan-fiction?**



use map/reduce of course

Map/Reduce Strategy

- Keep it simple!

Wordcount Strategy

- Let <word, 1> be the <key,value>

Wordcount Strategy

- Let Hadoop do the hard work

Wordcount Map/Reduce:

Loop
Until
Done

Get word

Emit <word> < 1>

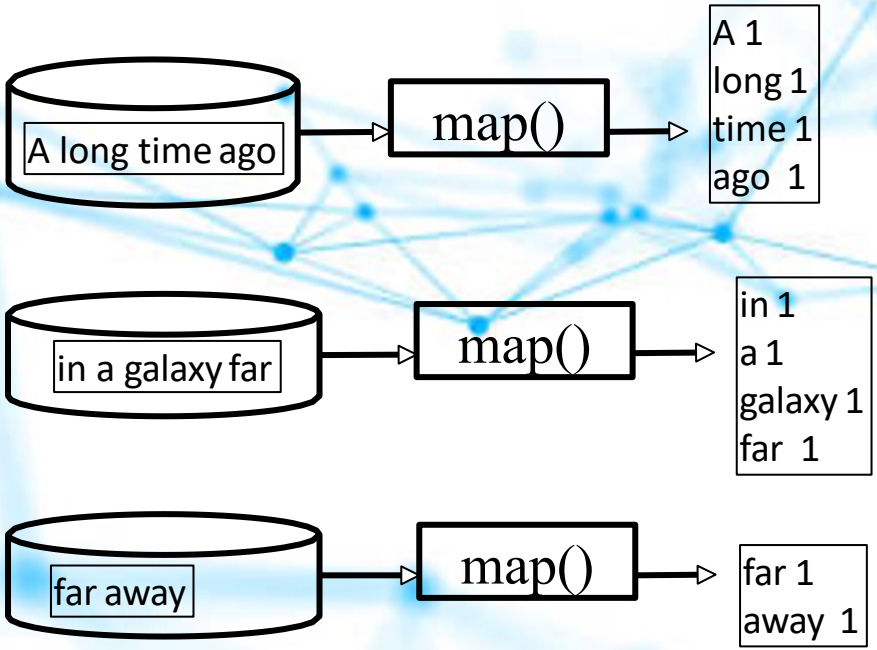
What One Mapper Does

Wordcount Map/Reduce:

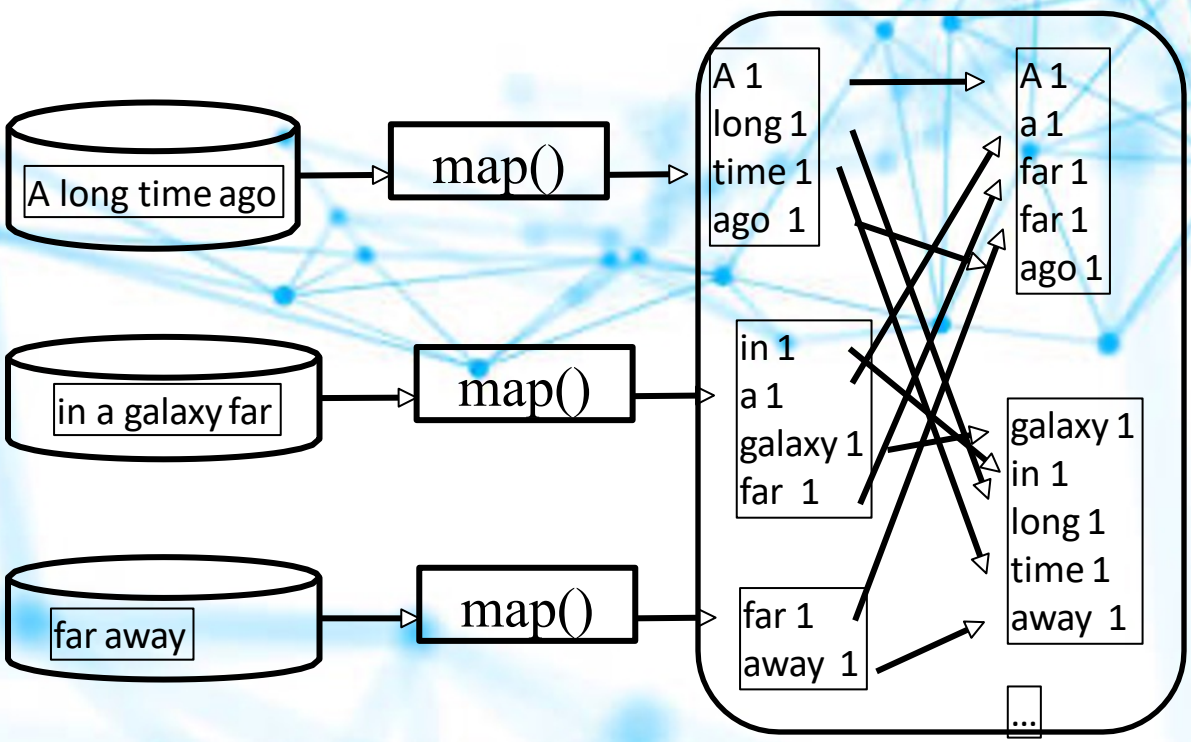
Loop Over key-values

```
Get next <word><value>
If <word> is same as previous word
        add <value> to count
else
        emit <word> < count>
        set count to 0
```

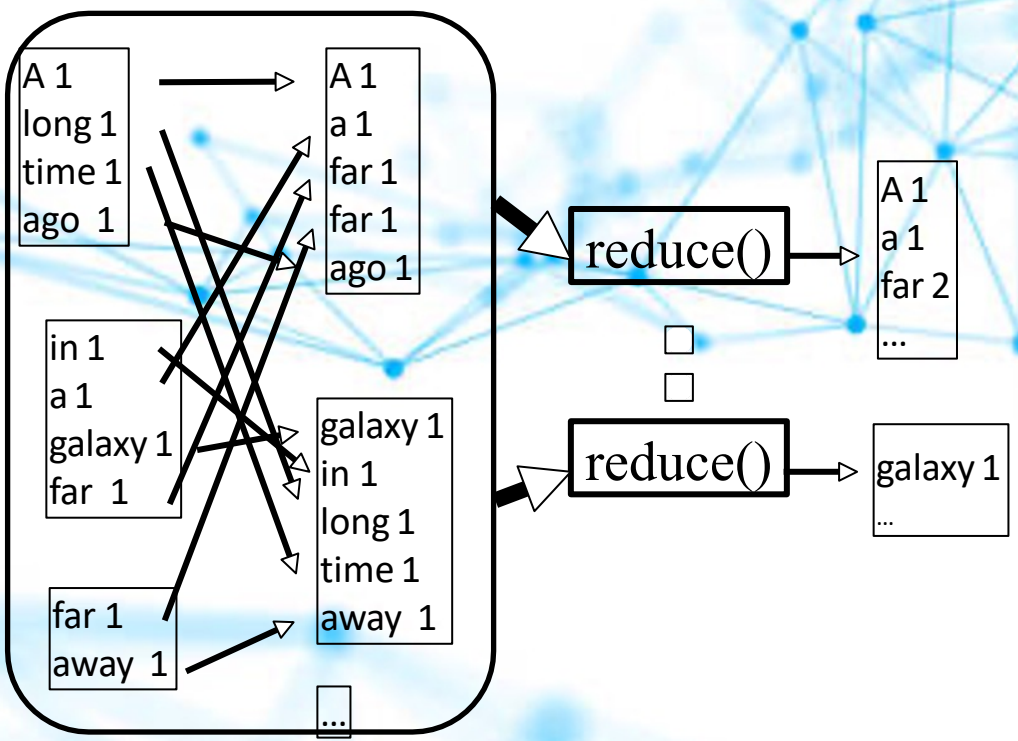
map() output

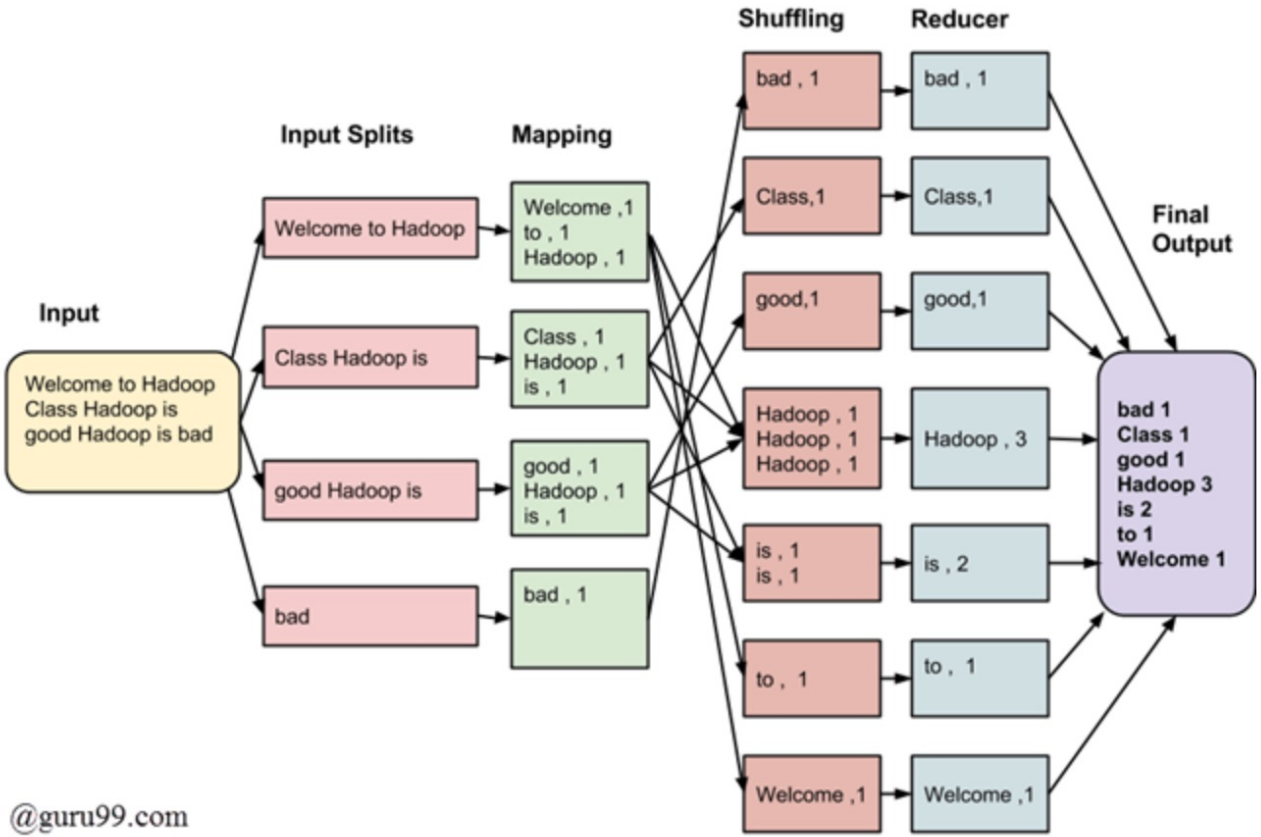

A long time ago
map()
A 1
long 1
time 1
ago 1
in a galaxy far
map()
in 1
a 1
galaxy 1
far 1
far away
map()
far 1
away 1

Hadoop shuffles, groups, and distributes

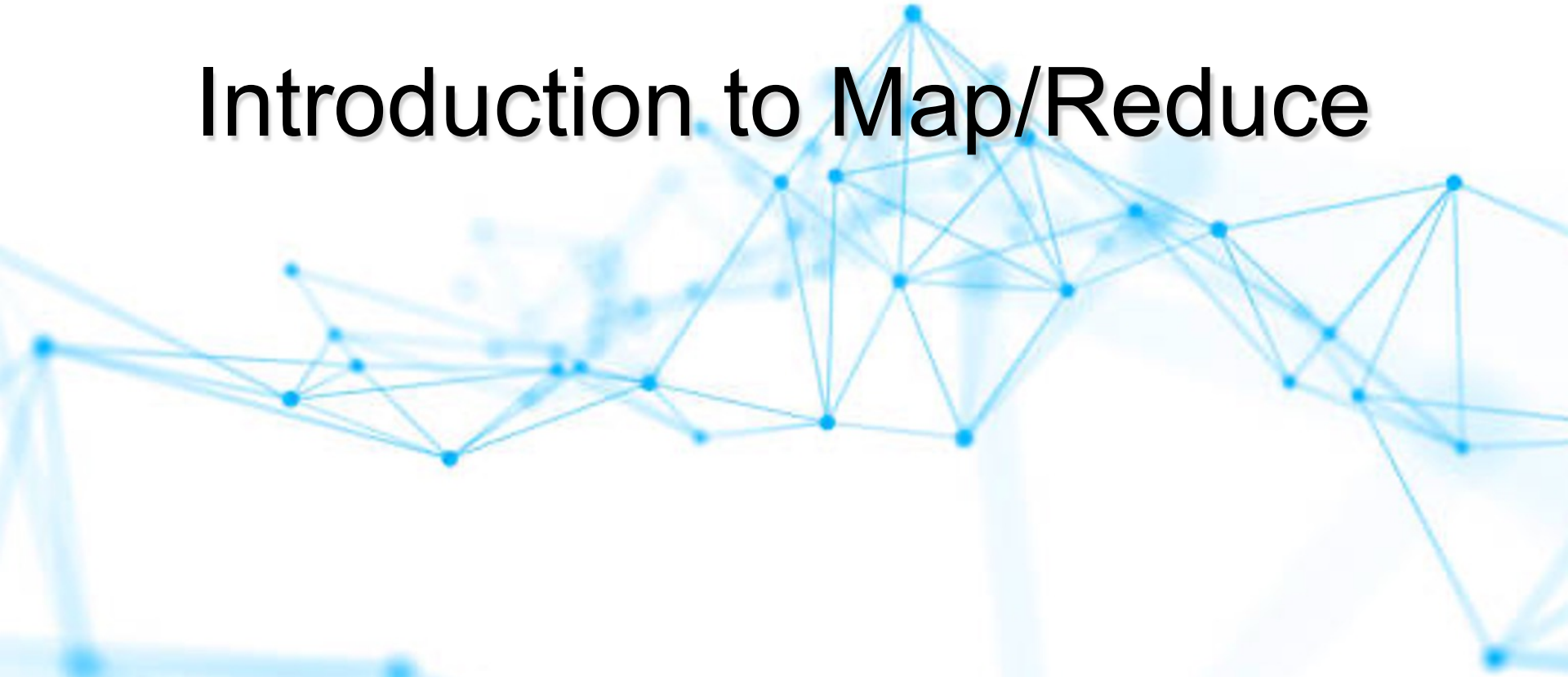


reduce() aggregates





MapReduce Architecture

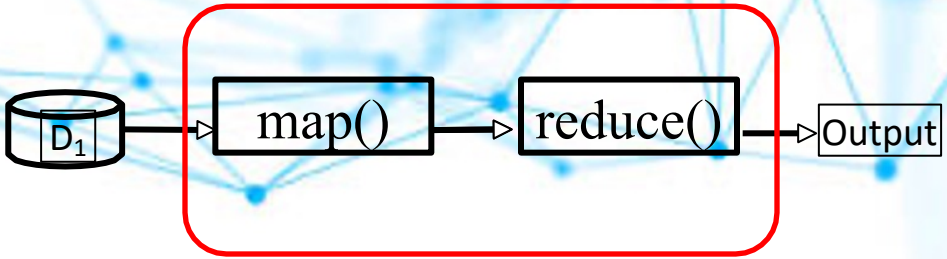


Introduction to Map/Reduce

Examples and Principles

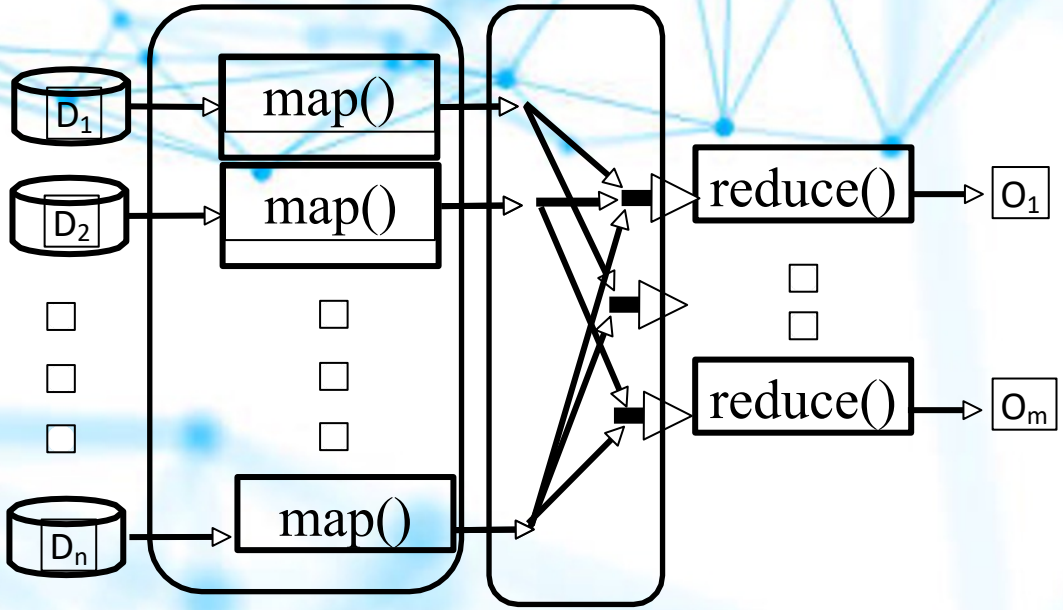
Recall the framework:

- User defines <key,value>, mapper, and reducer



Recall the framework:

- Hadoop handles the logistics



Hadoop Rule of Thumb

- 1 mapper per data split (typically)

Hadoop Rule of Thumb

- 1 mapper per data split (typically)
- 1 reducer per computer core (best parallelism)

Hadoop Rule of Thumb

- 1 mapper per data split (typically)
- 1 reducer per computer core (best parallelism)

Number Output Files

Processing Time

Wordcount Strategy

- Let <word, 1> be the <key,value>
- Simple mapper & reducer
- Hadoop did the hard work of shuffling & grouping

Good key-value properties

- Simple
- Enables reducers to get correct output

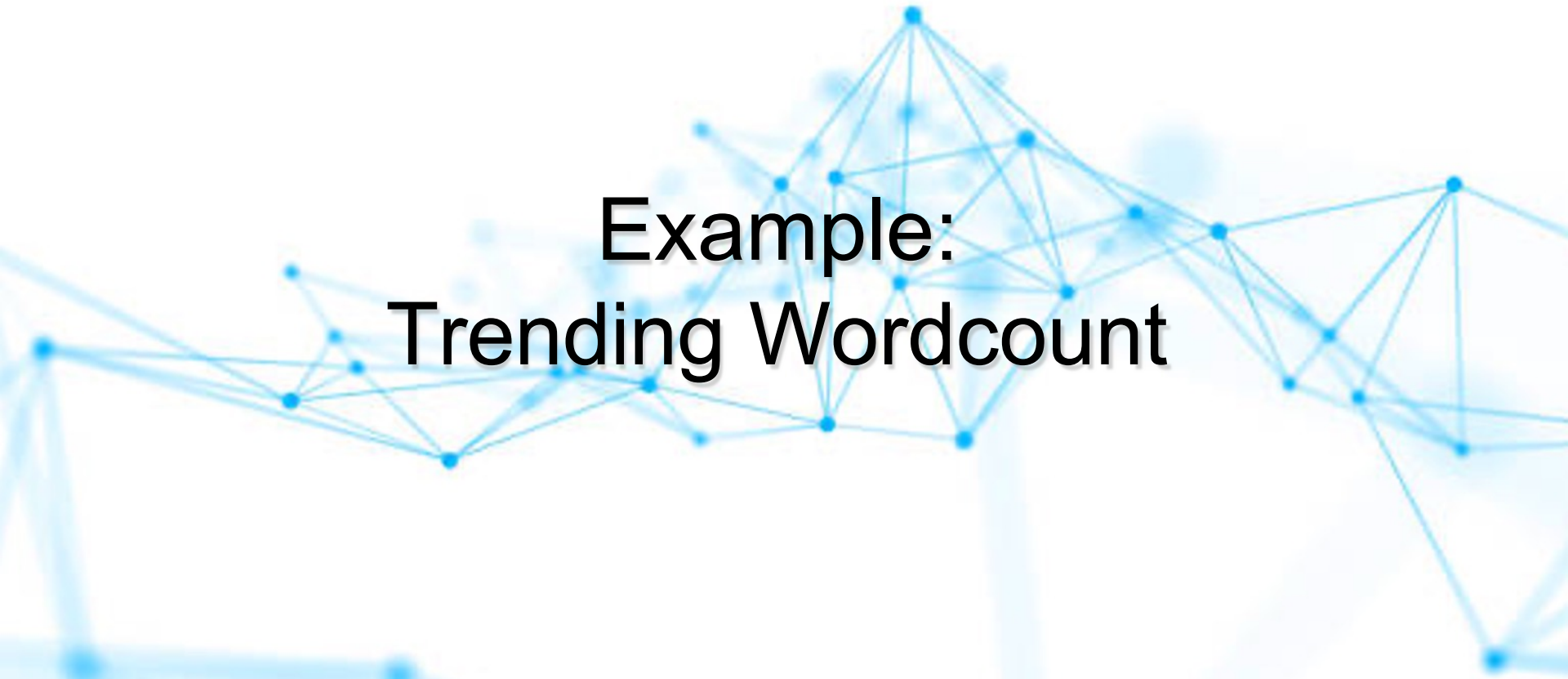
Key-Value simplicity

Shuffling & Grouping

Good Task Decomposition:

Mappers: simple and separable

R*educers:* easy consolidation


Example:
Trending Wordcount

Trending Wordcount

- Twitter Data: date, message, location, … [other metadata]

Trending Wordcount

- Twitter Data: date, message, location, … [other metadata]

Task 1 Get word count by day

Task 2 Get total word count

Trending Wordcount

Task 1: get word count by day

Trending Wordcount

Task 1: get word count by day

Design: *Use composite key*

Map/Reduce: <date word,count>

Trending Wordcount

Task 2: get total word count

Trending Wordcount

Task 2: get total word count

Easy way:

re-use previous wordcount

Trending Wordcount

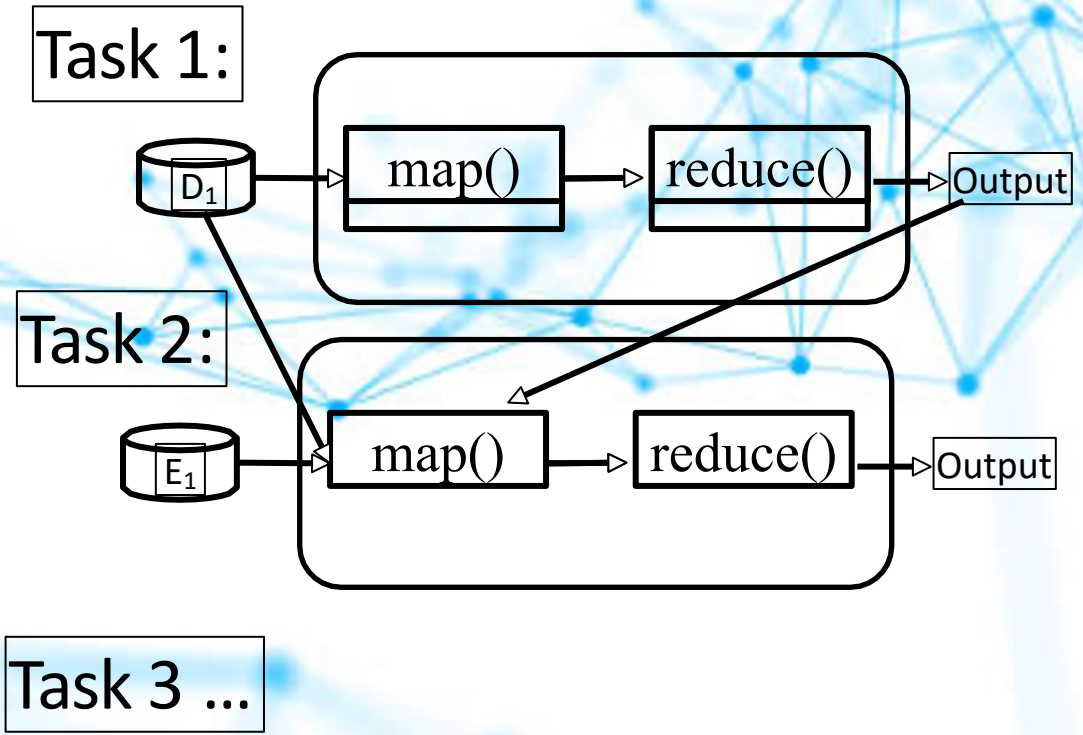
Task 2: get total word count

Alternatively:

use Task 1 output

(it's partially aggregated)

Cascading Map/Reduce



Example:
Joining Data

Joining Data

- Task: combine datasets by key
  - A standard data management function

Joining Data

- Task: combine datasets by key
  - A standard data management function
  - In pseudo SQL

    Select * from table A, table B, where A.key=B.key

Joining Data

- Task: combine datasets by key
  - A standard data management function
  - In pseudo SQL

    Select * from table A, table B, where A.key=B.key
  - Joins can be inner, left or right outer

Joining Data

- Task: given two wordcount datasets …

Joining Data

- Task: given two wordcount datasets …

File A: <word, total-count>

```
able ,   5
actor ,   18
burger ,  25
.
.
.
```

Joining Data

- Task: given two wordcount datasets …

File A: <word, total-count>

```
able ,   5
actor ,   18
burger ,  25
.
.
.
```

File B: <date word, day-count>

```
Jan-16  able  ,   2
Feb-22  actor ,   15
May-03 actor ,   3
Jul-4      burger,  20
.
.
.
```

Joining Data

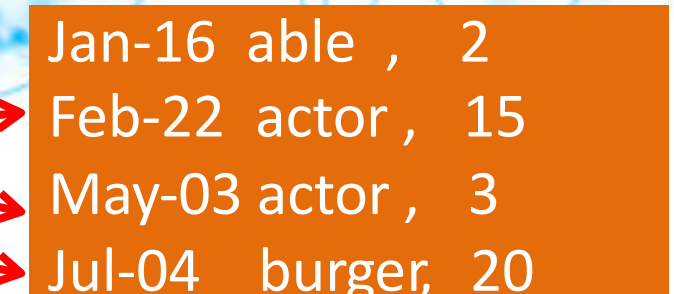
- Task: combine by word

File A: <word, total-count>

able , 5
actor , 18
burger , 25
.
.
.

File B: <date word, day-count>

Jan-16 able , 2
Feb-22 actor , 15
May-03 actor , 3
Jul-04 burger, 20
.
.
.



Joining Data

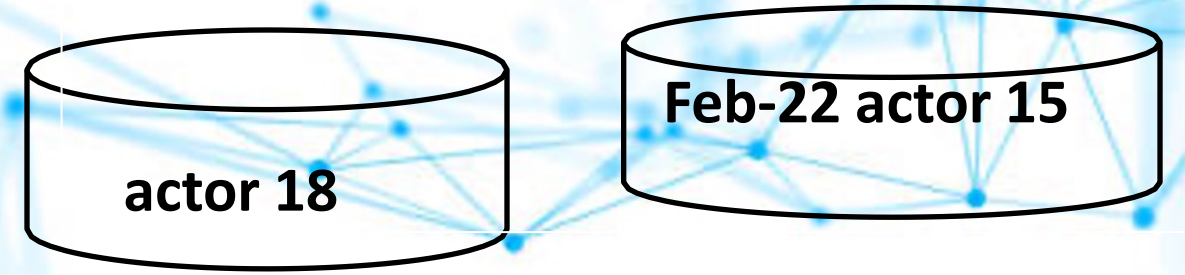
- Result wanted:

File AjoinB: <word date, day-count total-count >

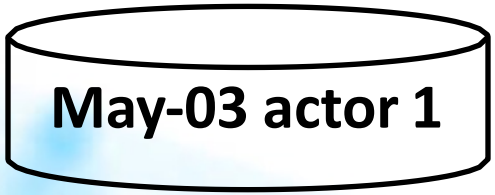
```
able    Jan-16,    2   5
actor   Feb-22,    15  18
actor   May-03,    1   18
burger  Jul-04,    20  25
.
.
.
```

Joining Data

- Recall that data is split in parts



How to gather the right pieces?



Key-Value & Task Decomposition

- Main design consideration:

Join depends on word

*(e.g. Select * where A.word=B.word)*

Key-Value & Task Decomposition

- For the join:
  - Let <key> = word
  - Let <value> = other info

<*word,* … >

Key-Value & Task Decomposition

File A: <word, total-count>

```
able ,   5
actor ,  18
. . .
```

File B: <date word, day-count>

```
Jan-16  able ,   2
Feb-22  actor ,  15
. . .
```

Key-Value & Task Decomposition

- Note:

File A: <word, total-count>

```
able ,   5
actor ,  18
. . .
```

File B: <date word, day-count>

```
Jan-16  able ,   2
Feb-22  actor ,  15
. . .
```

word already the key

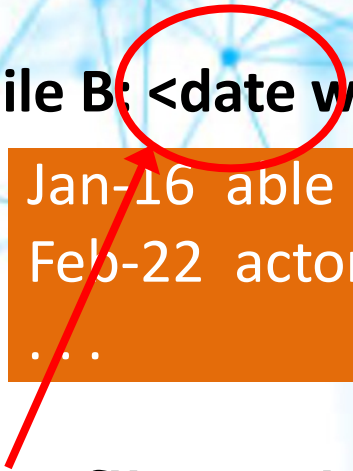
Key-Value & Task Decomposition

File A: <word, total-count>

```
able ,   5
actor ,  18
. . .
```

File B: <date word, day-count>

```
Jan-16  able ,   2
Feb-22  actor ,  15
. . .
```



date needs to be filtered out

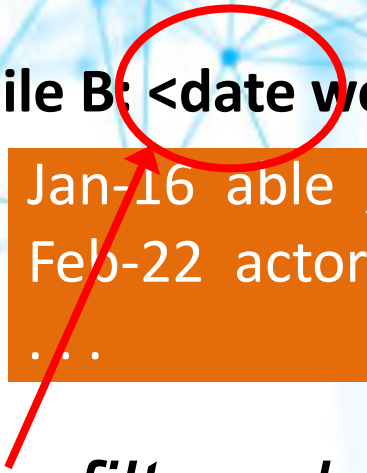
Key-Value & Task Decomposition

File A: <word, total-count>

```
able ,   5
actor ,  18
. . .
```

File B: <date word, day-count>

```
Jan-16  able ,   2
Feb-22  actor ,  15
. . .
```



date needs to be filtered out

Where should date info go?

Key-Value & Task Decomposition

<word, date day-count total-count >

Task Decomposition

- Now data sets are:

File A: <word, total-count>

```
able ,     5
actor ,   18
burger ,  25
.
.
.
```

File B_new: <word, date count>

```
able   ,  Jan -16  2
actor ,   Feb-22 15
actor ,   May-03 3
burger , Jul-04 20
.
.
.
```

Task Decomposition

- How will Hadoop shuffle & group these?

File A: <word, total-count>

```
able ,     5
actor ,    18
burger ,   25
.
.
.
```

File B_new: <word, date day-count>

```
able   ,  Jan-16   2
actor ,   Feb-22 15
actor ,   May-03 3
burger , Jul-04 20
.
.
.
```

Task Decomposition

- How will Hadoop shuffle & group these?

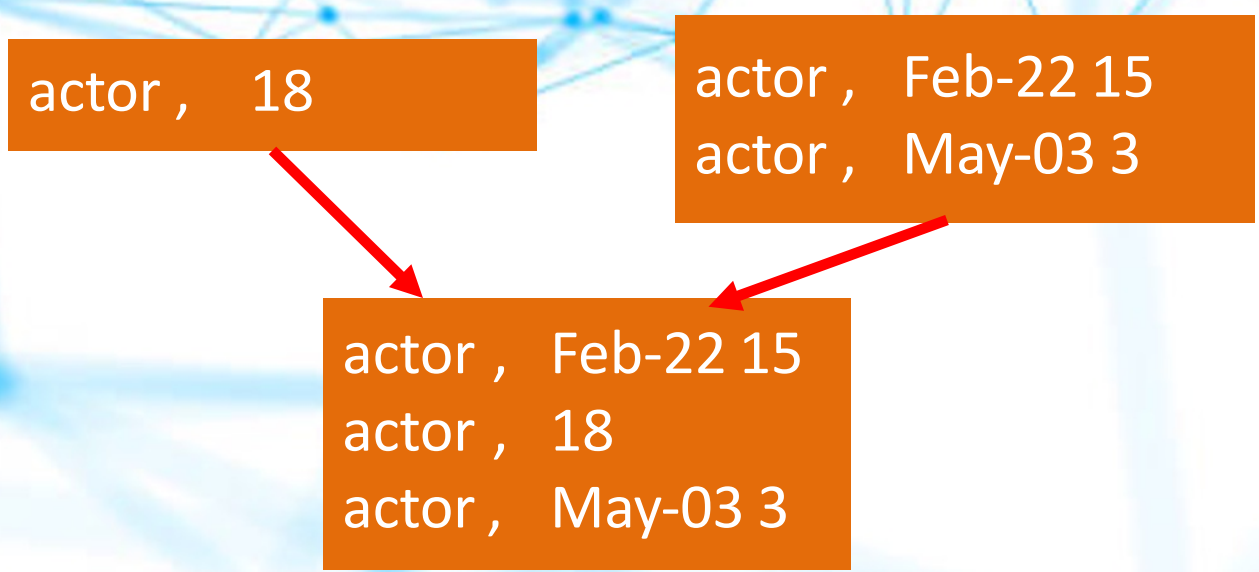
Let's focus on 1 key:

actor , 18

actor , Feb-22 15
actor , May-03 3

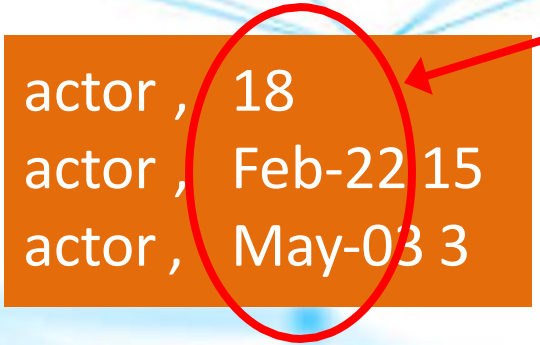
Task Decomposition

- Hadoop gathers the data for a join



Task Decomposition

- Reducer now has all the data for same word grouped together



Task Decomposition

- Reducer can now join the data and put date back into key

Example:
Vector Multiplication