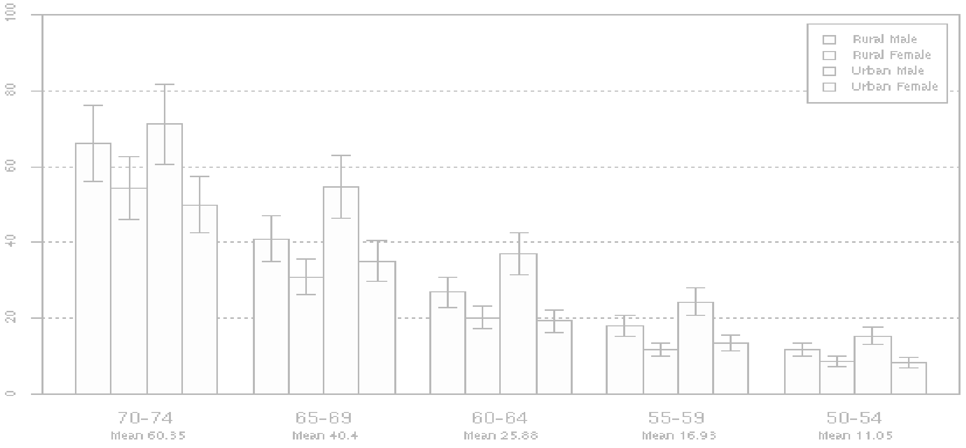
Using rHadoop for collaborative filtering

Binbin Chen

Data Scientist

Revolution Analytics, Singapore



What is ‘Hadoop’?

An open-source software framework that supports data-intensive distributed applications, licensed under the Apache V2 license

"The name my kid gave a stuffed yellow elephant. Short, relatively easy to spell and pronounce, meaningless, and not used elsewhere: those are my naming criteria. Kids are good at generating such. Googol is a kid’s term."

-- Doug Cutting

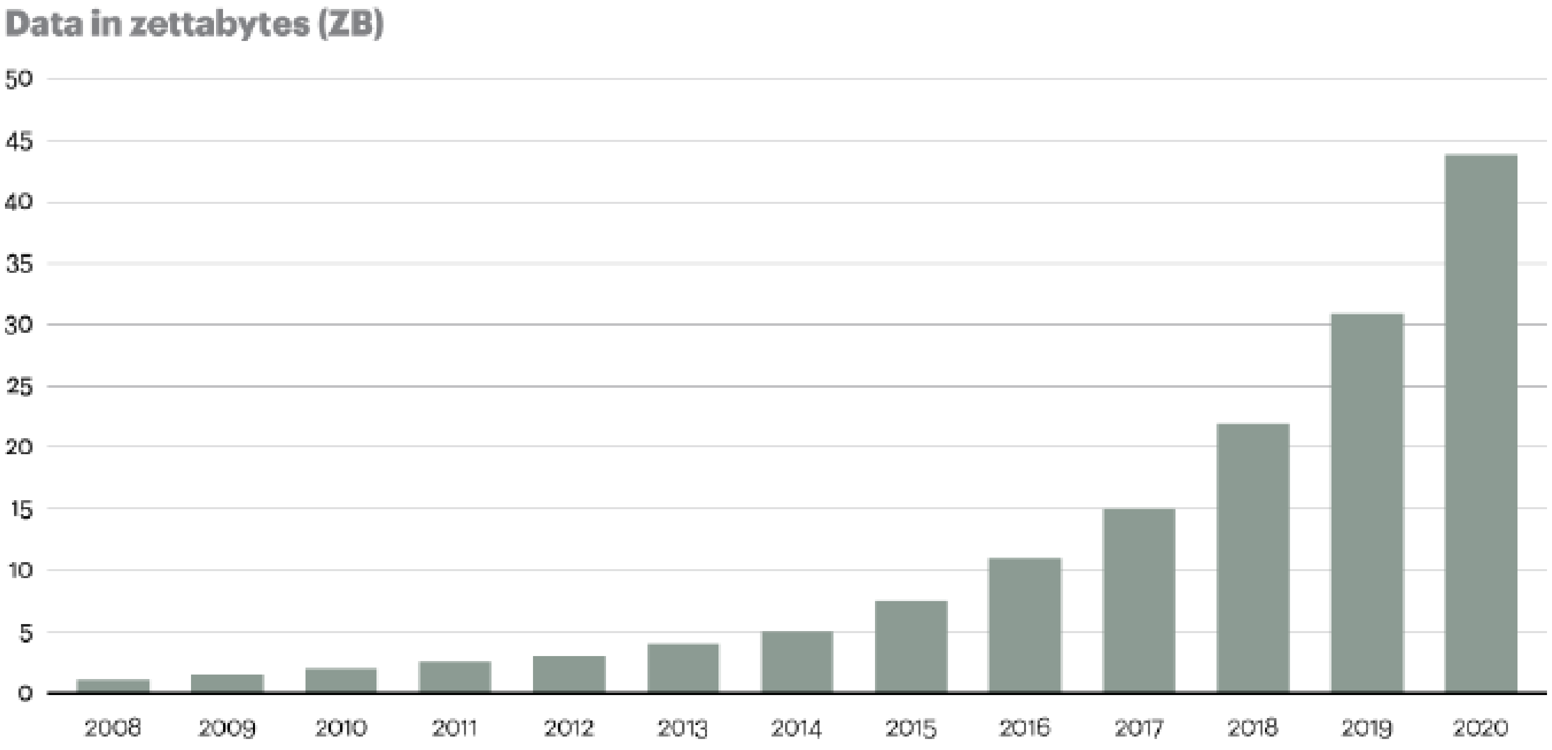
Who are using Hadoop?

PoweredBy

Why Hadoop so popular

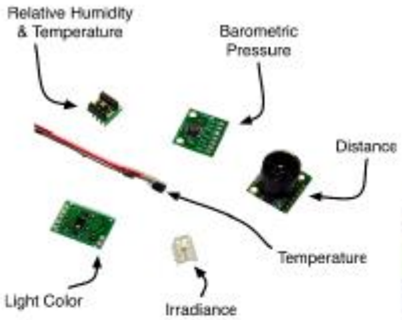
The explosion of data

Data is growing at a 40% compound annual rate, reaching nearly 45 ZB by 2020



Source: Oracle, 2012

The explosion of data



- 35 hours of video uploaded to Youtube each minute
- 200 million tweets in Twitter each day
- 6 billion photos uploaded to Facebook each month

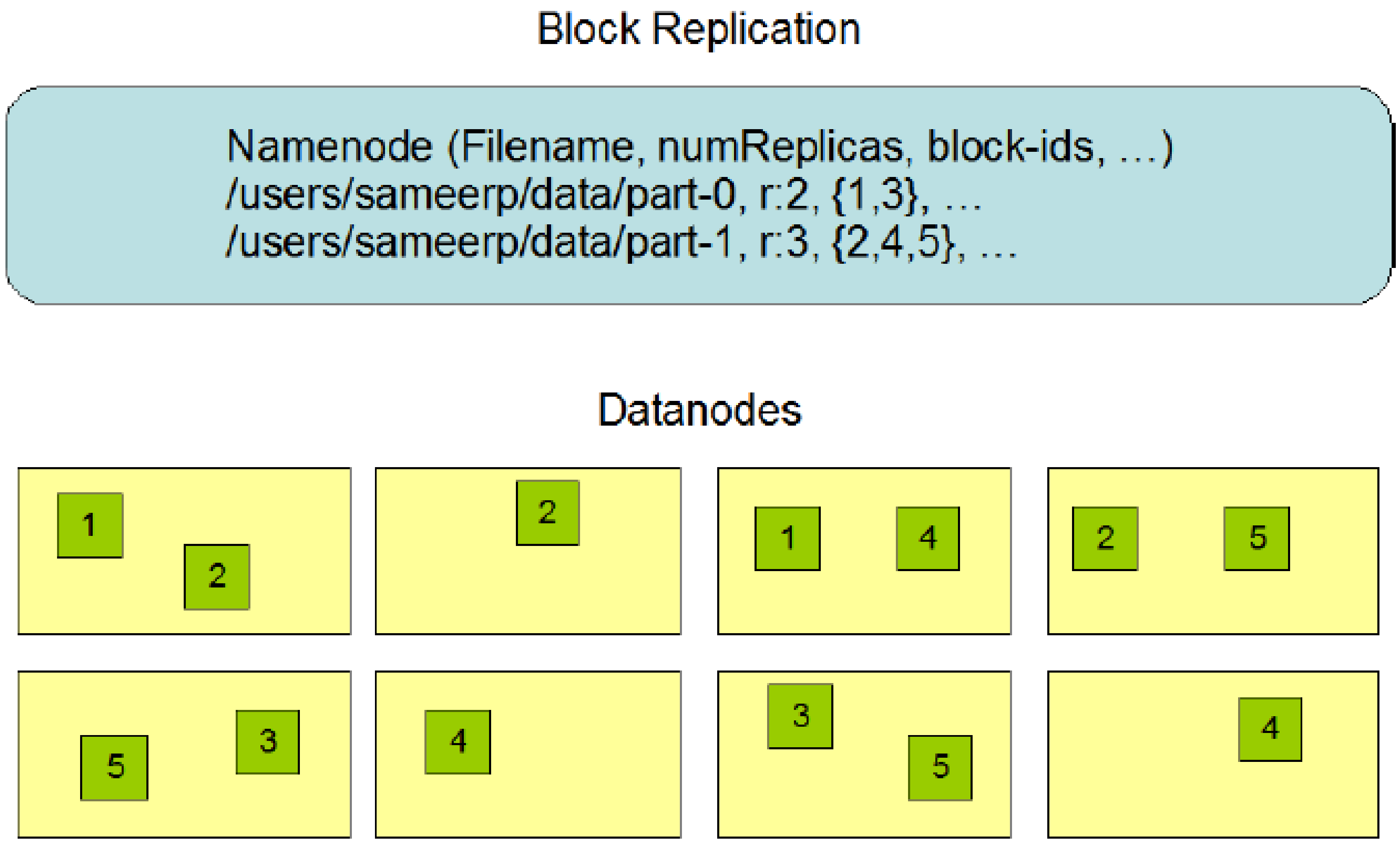
How Hadoop fit to the Big Data

- Hadoop scales massively and predictably
- Hadoop is tolerant to partial failure
- Hadoop simplifies the distributed computing programming
- Hadoop achieves data locality

Hadoop Distributed File System (HDFS)

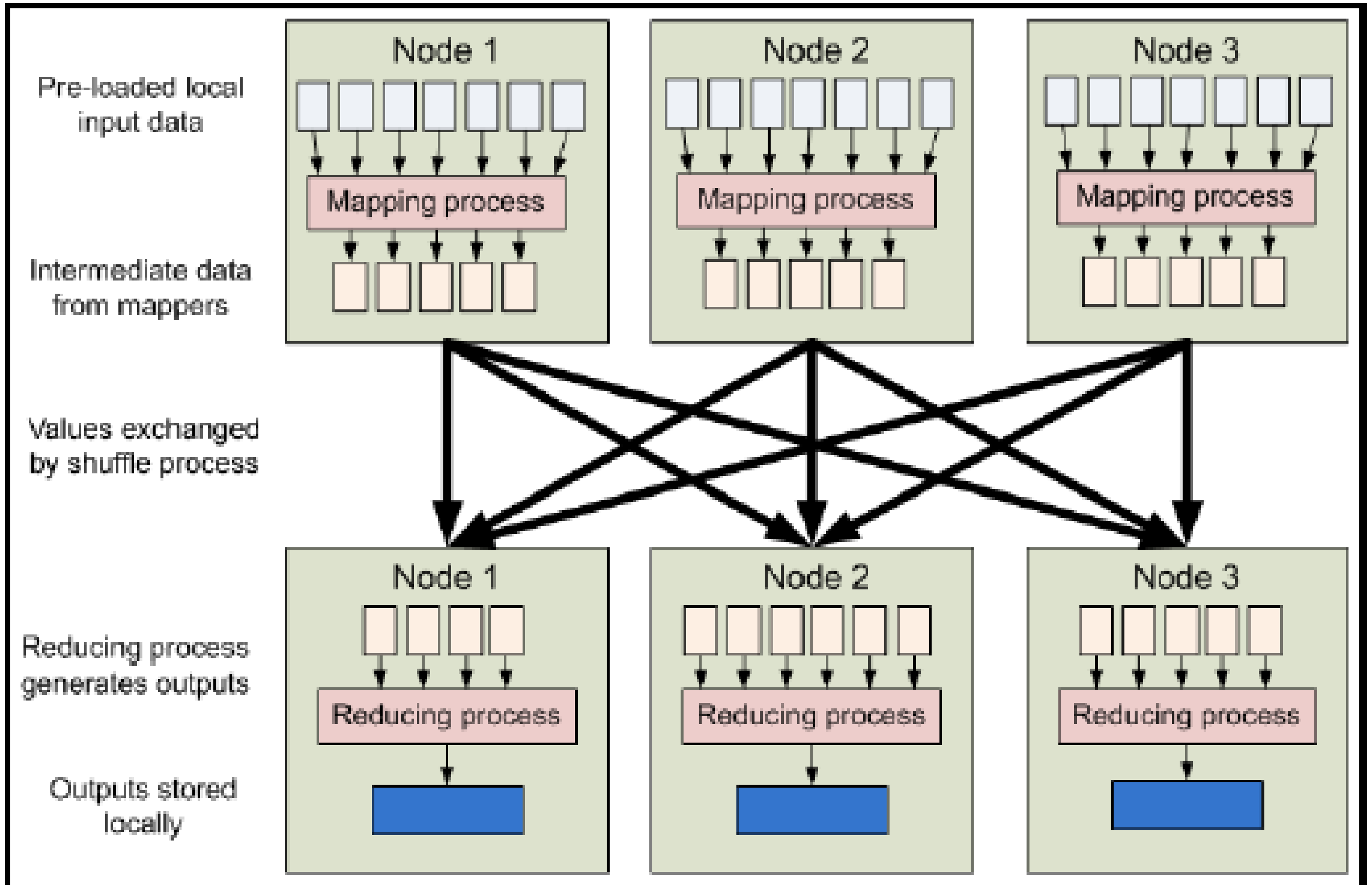
- Master-slave architecture: Namenode and Datanode
- Files are stored in blocks on various nodes
- Blocks are replicated on various nodes
- Low cost and users can put all their data into HDFS

Hadoop Distributed File System (HDFS)



Hadoop MapReduce

- Master-slave architecture: Jobtracker and tasktracker
- Auto-failover of tasks
- Move code to data



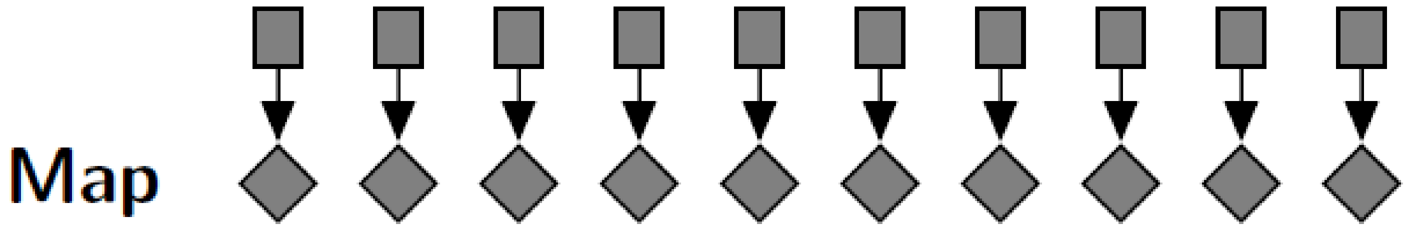
The Combine

Data

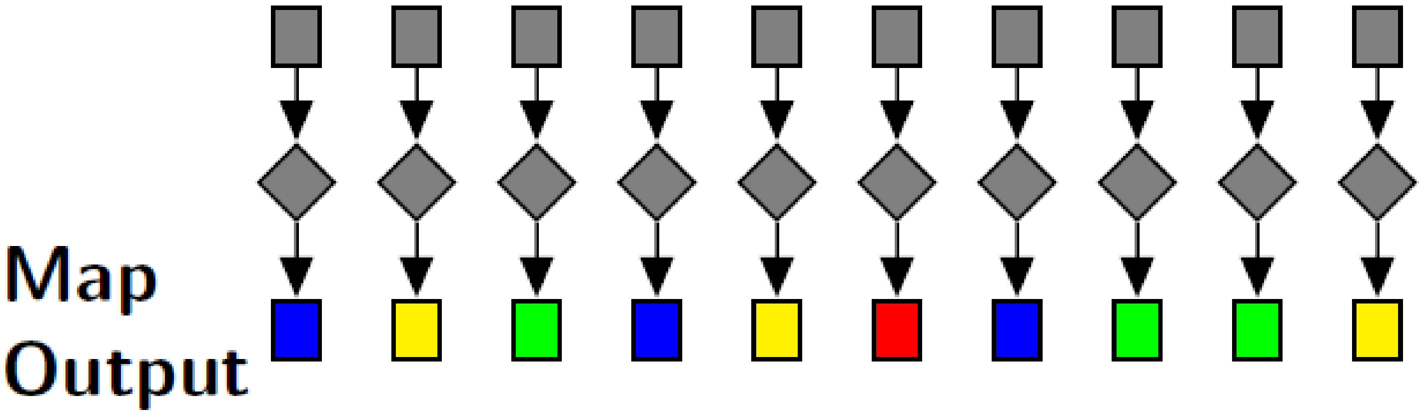
The Combine

Map Input

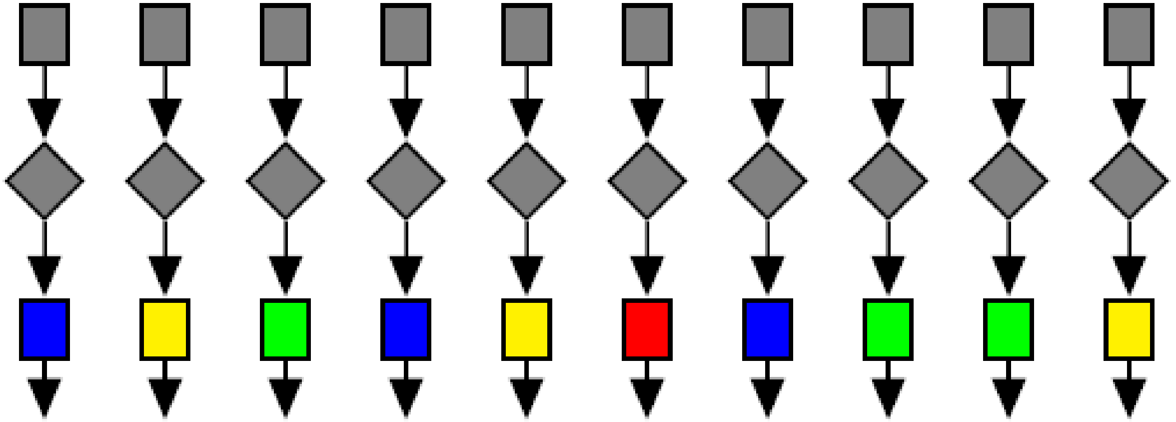
The Combine



The Combine



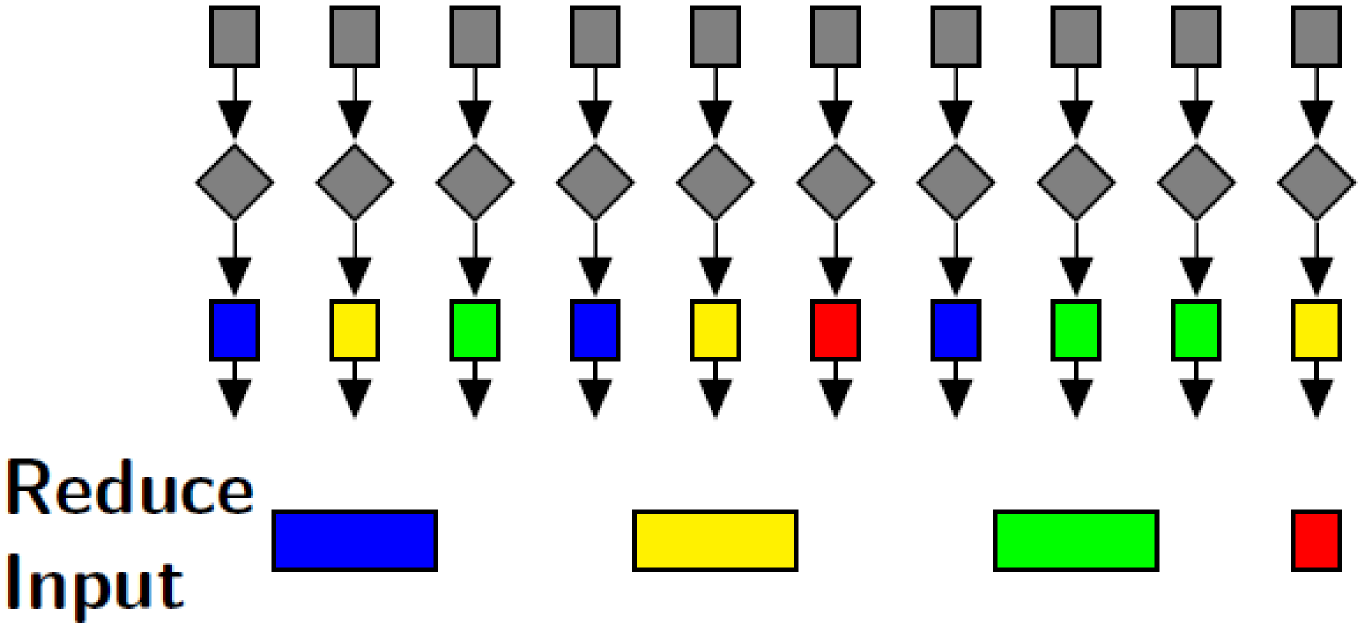
The Combine



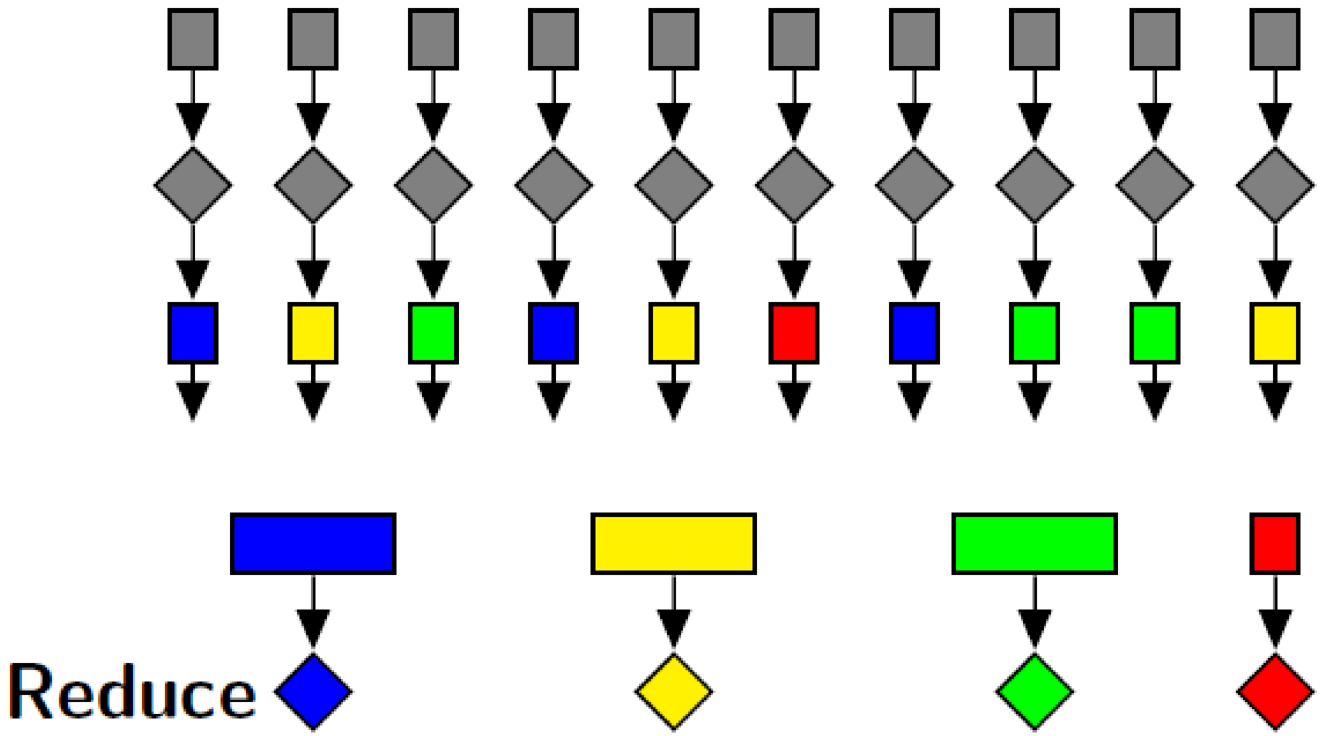
Shuffle-Sort

...

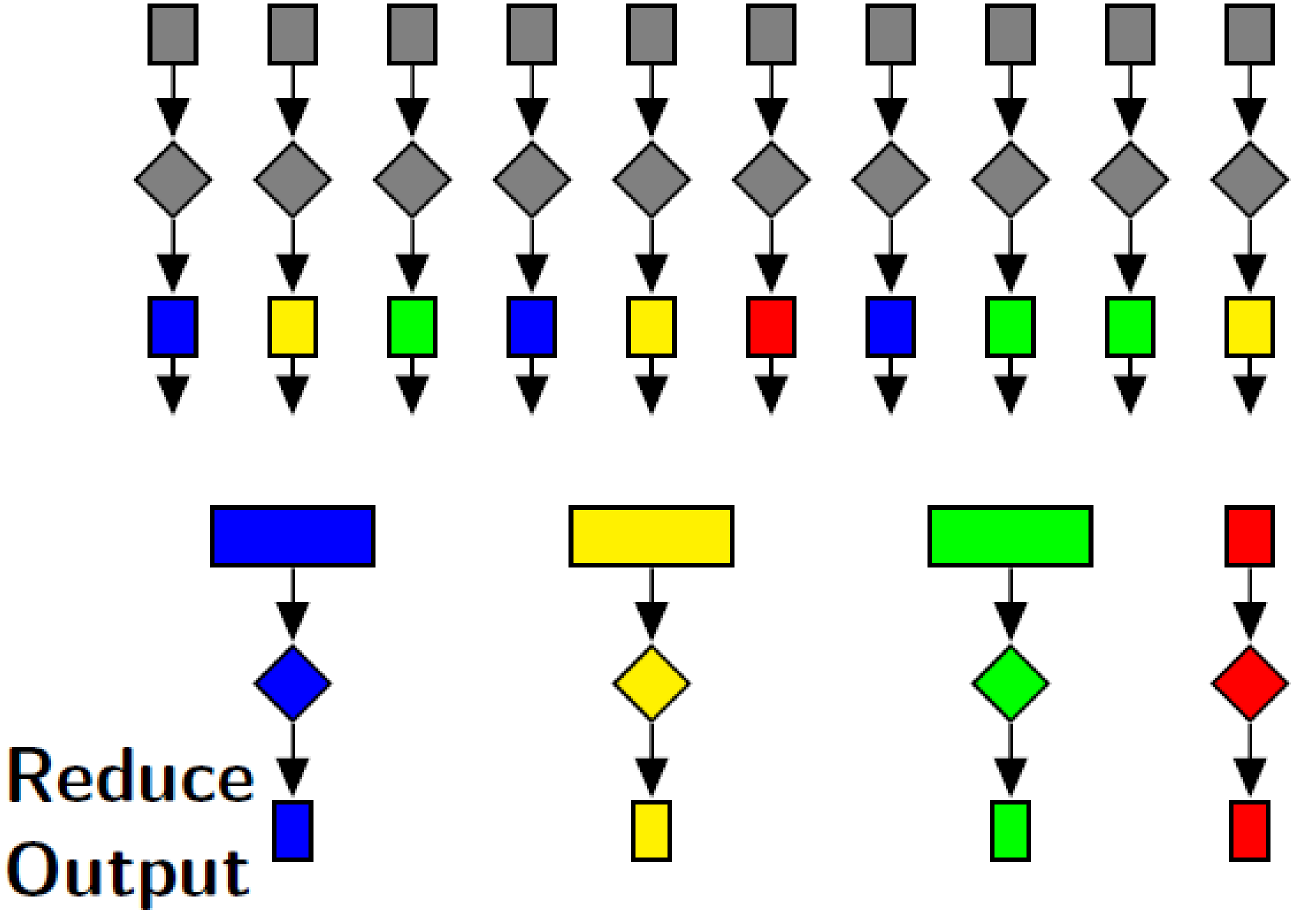
The Combine



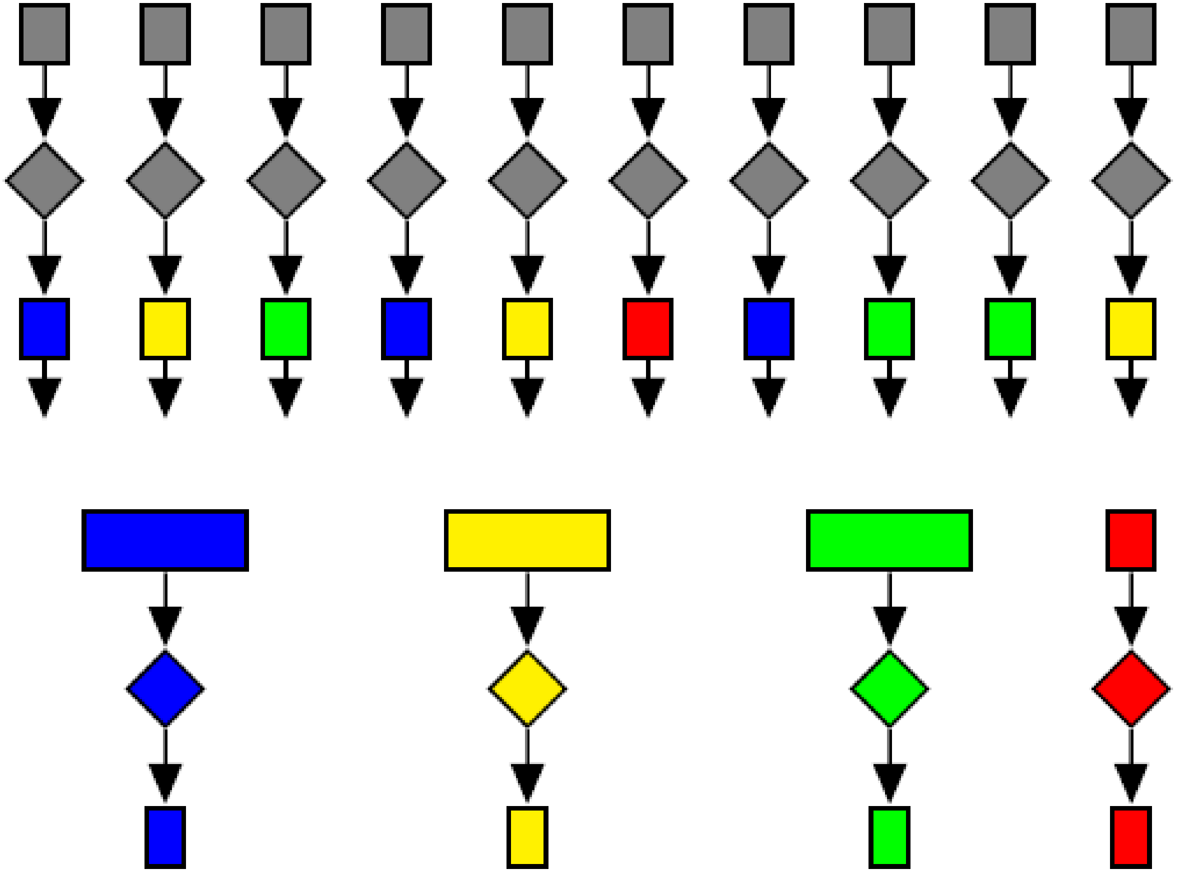
The Combine



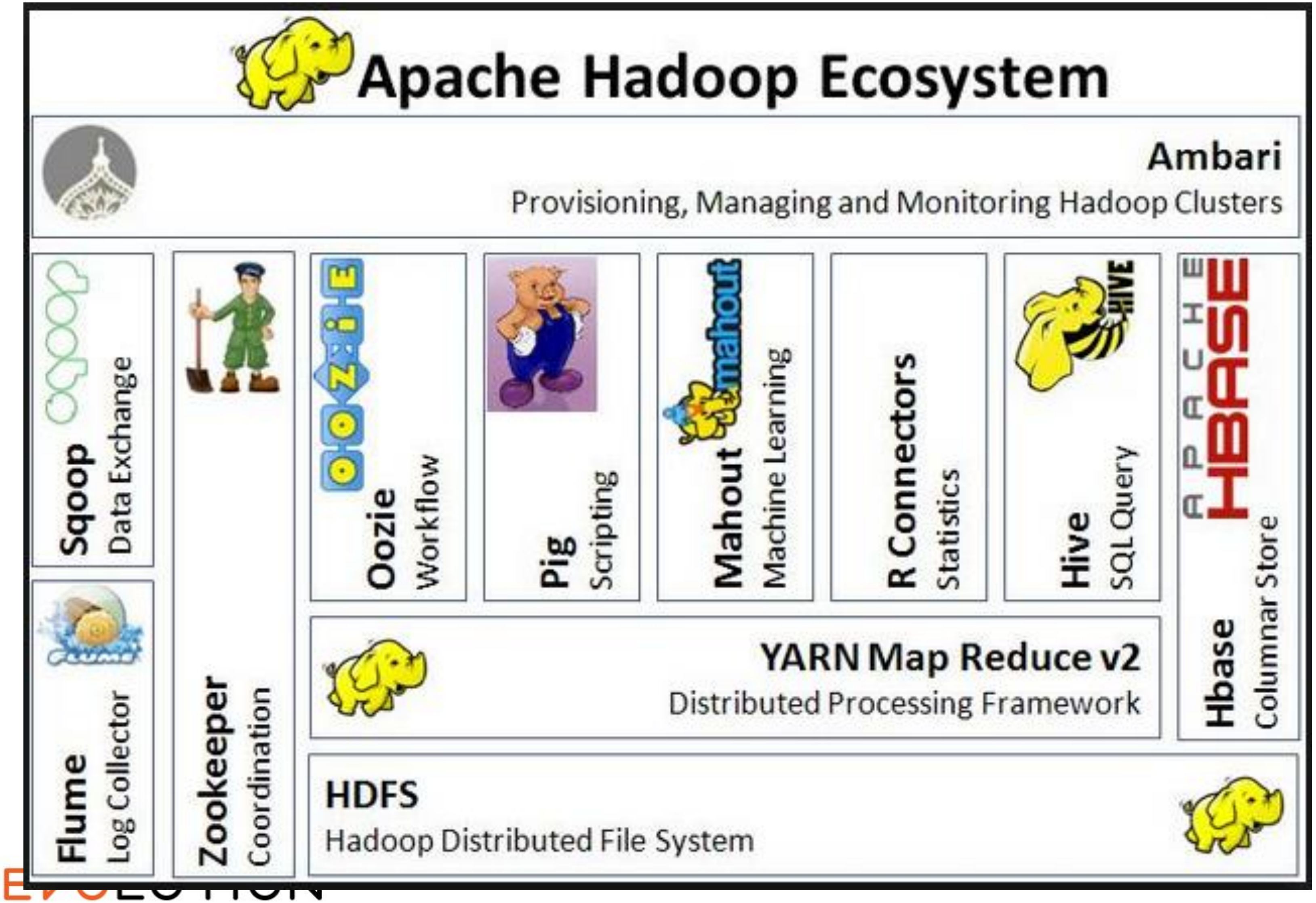
The Combine



The Combine



Hadoop Ecosystem



amazon webservices™

cloudera Ask Bigger Questions

Hortonworks®

IBM

Greenplum

intel

MAPR

Hadoop use cases

- Risk Modelling

  Banks put all sources of data in Hadoop, including customer profiles, call-center recorders, chat sessions, and emails to get a clear view of each customer’s financial status

- Credit card fraud detection

  Visa use Hadoop to process all transaction records and build predictive models to detect fraudulent transactions

Hadoop and R: Simplified MapReduce

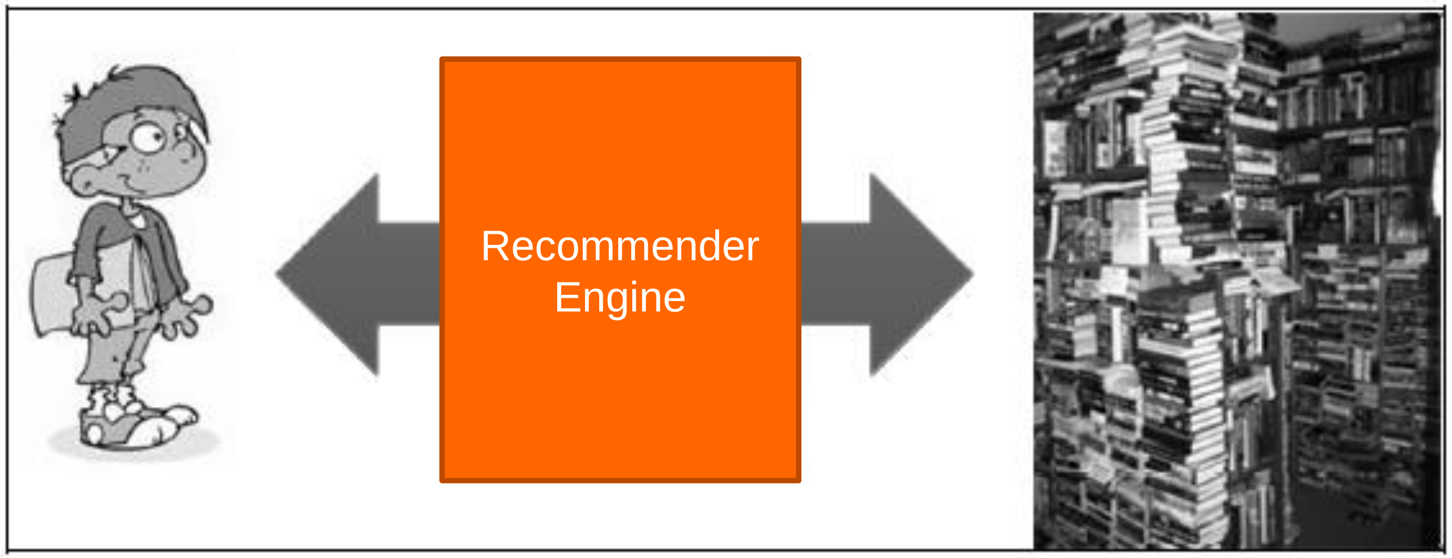
- Rhadoop
  - rmr2, rhdfs, rhbase, plyrmr
  - Utilizing Hadoop the streaming interface
  - Coding R functions following MapReduce concepts
- RevoScaleR Hadoop Features
  - Read data from HDFS and apply big-data statistical models from Revolution R Enterprise to excute Hadoop Jobs

Collaborative Filtering

Outline

- Introduction to the collaborative filtering
- Implement CF algorithm in R RAM
- Implement CF algorithm using rmr2

What's Recommender Engine



In the information overload time, recommender engine:

1. Help the users to discover the information that are valuable to them
2. Recommend the information that are interested to the users

Reference: 推荐系统实践

Users’ behavior

- What’s users’ behavior
  - Explicit feedback: like/not, buy/not, rating,…
  - Implicit feedback: page view, click, time for a page and so on
- For this example, we using the basis data:

user	item	pref
1	101	5
1	102	3
…	…	…

Collaborative filtering algorithms

- Neighborhood-based
  - User-based CF: recommend the items that the similar user like
  - Item-based CF: recommend the similar items to what he likes before
- Latent factor model
- Random walk on graph

The most popularly used in industry: Amazon, Alibaba, Hulu,...

Netflix competing

REVOLUTION ANALYTICS

Item-based CF

- Two steps:
  - Calculate the similarity of the items
  - Recommend items to the user based on the similarity of the items and user's historical preference.

- Reference:
  - **Programming Collective Intelligence**
  - Mahout in Action
  - 推荐系统实践
  - Zhang Dan's Blog: http://blog.fens.me/

REVOLUTION ANALYTICS

Step 1: Calculate the similarity of two items

List all the users

	u1	u2	u3	u4	u5	u6
Book 1	3	0	0	5	2	0
Book 2	2	5	0	0	4	0

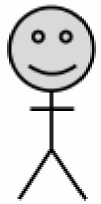
cosine:	sim(Book 1,Book 2) = cos(Book 1, Book 2) = 0.34
Pearson correlation:	sim(Book 1,Book 2) = cor(Book 1, Book 2) = -0.19
cooccurence:	sim(Book 1,Book 2) = #{intersection of the users} / #{union of the user} = 2/4

Step 2: score the preference of each book

Books I gave the rating

	R in a Nut Shell Score: 5	Machine learning with R Score: 4	
R in action	0.8	0.6	0.8*5+0.6*4=6.4
Programming Collective Intelligence	0.4	0.5	0.4*5+0.5*4=4

similarity



recommend

Two steps for recommendation:

1. ***Calculate the similarity matrix***
2. ***Give score for each book recommended***

REVOLUTION ANALYTICS

Item-based algorithm in RAM

```
> dataCF
   user item pref
1     1  101  5.0
2     1  102  3.0
3     1  103  2.5
4     2  101  2.0
5     2  102  2.5
6     2  103  5.0
7     2  104  2.0
8     3  101  2.0
9     3  104  4.0
10    3  105  4.5
11    3  107  5.0
12    4  101  5.0
```

reshape

```
> dataCF2
  101 102 103 104 105 106 107
1   5 3.0 2.5 0.0 0.0   0   0
2   2 2.5 5.0 2.0 0.0   0   0
3   2 0.0 0.0 4.0 4.5   0   5
4   5 0.0 3.0 4.5 0.0   4   0
5   4 3.0 2.0 4.0 3.5   4   0
```

```
> prefUser1

104 6.855
```

Calculate cosine similarity

```
> simMat
     101  102  103  104  105  106  107
101 1.00 0.76 0.80 0.78 0.47 0.74 0.23
102 0.76 1.00 0.79 0.46 0.37 0.43 0.00
103 0.80 0.79 1.00 0.63 0.18 0.53 0.00
104 0.78 0.46 0.63 1.00 0.75 0.80 0.53
105 0.47 0.37 0.18 0.75 1.00 0.43 0.79
106 0.74 0.43 0.53 0.80 0.43 1.00 0.00
107 0.23 0.00 0.00 0.53 0.79 0.00 1.00
```

Item-based algorithm in RAM

```
> dataCF
   user item pref
1     1  101  5.0
2     1  102  3.0
3     1  103  2.5
4     2  101  2.0
5     2  102  2.5
6     2  103  5.0
7     2  104  2.0
8     3  101  2.0
9     3  104  4.0
10    3  105  4.5
11    3  107  5.0
12    4  101  5.0
```

reshape

```
> dataCF2
  101 102 103 104 105 106 107
1   5 3.0 2.5 0.0 0.0   0   0
2   2 2.5 5.0 2.0 0.0   0   0
3   2 0.0 0.0 4.0 4.5   0   5
4   5 0.0 3.0 4.5 0.0   4   0
5   4 3.0 2.0 4.0 3.5   4   0
```

> prefUser1

	user1
~~101~~	~~9.280~~
~~102~~	~~8.775~~
~~103~~	~~8.870~~
104	6.855
105	3.910
106	6.315
107	1.150

```r
### reshape the data into rating matrix
dataCF2 <- as.matrix(simple_triplet_matrix(i
      = dataCF$user, j = dataCF$item-100, v =
      dataCF$pref))
rownames(dataCF2) <- 1:5
colnames(dataCF2) <- 101:107

### Calculate the similarity matrix
tmp1 <- t(dataCF2)%*%dataCF2
tmp2 <- diag(tmp1)
tmp2 <- tmp2%*%t(tmp2)
simMat <- round(tmp1/sqrt(tmp2),digits = 2)

### calculate the preference
prefMat <- simMat %*% t(dataCF2)
```

Calculate cosine similarity

```
> simMat
     101  102  103  104  105  106  107
101 1.00 0.76 0.80 0.78 0.47 0.74 0.23
102 0.76 1.00 0.79 0.46 0.37 0.43 0.00
103 0.80 0.79 1.00 0.63 0.18 0.53 0.00
104 0.78 0.46 0.63 1.00 0.75 0.80 0.53
105 0.47 0.37 0.18 0.75 1.00 0.43 0.79
106 0.74 0.43 0.53 0.80 0.43 1.00 0.00
107 0.23 0.00 0.00 0.53 0.79 0.00 1.00
```

REVOLUTION ANALYTICS

Item-based algorithm using Mapreduce

- We using 6 mapreduce jobs
  - Calculate the item **similarity matrix**: mapreduce1-4.
  - Calculate the user’s **preference over each item**: mapreduce 5-6.
- ZhangDan use cooccurence matrix as the similarity matrix. Code can be found here.
- Outline:
  - Introduce matrix multiplying

How to multiply two matrix

	u1	u2	u3	u4	u5
B101	5	2	2	5	4
B102	3	2.5	0	0	3
B103	2.5	5	0	3	2
B104	0	2	4	4.5	4
B105	0	0	4.5	0	3.5
B106	0	0	0	4	4
B107	0	0	5	0	0

%*%

	B101	B102	B103	B104	B105	B106	B107
u1	5	3	2.5	0	0	0	0
u2	2	2.5	5	2	0	0	0
u3	2	0	0	4	4.5	0	5
u4	5	0	3	4.5	0	4	0
u5	4	3	2	4	3.5	4	0

(B101,u1) (u1,B101)
(B101,u2) (u2,B101)
(B101,u3) (u3,B101)
(B101,u4) (u4,B101)
(B101,u5) (u5,B101)

(B101,u1) (u1,B102)
(B101,u2) (u2,B102)
(B101,u3) (u3,B102)
(B101,u4) (u4,B102)
(B101,u5) (u5,B102)

Matrix multiplying: The **col** of left matrix is **identical** to the **rows** of right matrix

(??, u1) * (u1, ??)

left ??: B101 - B105

right ??: B101 - B105

REVOLUTION ANALYTICS

How to multiply two matrix

	u1	u2	u3	u4	u5
B101	5	2	2	5	4
B102	3	2.5	0	0	3
B103	2.5	5	0	3	2
B104	0	2	4	4.5	4
B105	0	0	4.5	0	3.5
B106	0	0	0	4	4
B107	0	0	5	0	0

%*%

	B101	B102	B103	B104	B105	B106	B107
u1	5	3	2.5	0	0	0	0
u2	2	2.5	5	2	0	0	0
u3	2	0	0	4	4.5	0	5
u4	5	0	3	4.5	0	4	0
u5	4	3	2	4	3.5	4	0

(B101,u1)	(u1,B101)
(B101,u1)	(u1,B102)
(B101,u1)	(u1,B103)
(B102,u1)	(u1,B101)
(…,u1)	(u1,…)
(B101,u2)	(u2,B101)
(B101,u2)	(u2,B102)
(B101,u2)	(u2,B103)
(B101,u2)	(u2,B104)
(B102,u2)	(u2,B101)

Matrix multiplying: The **col** of left matrix is **identical** to the **rows** of right matrix

(??, u1) * (u1, ??)

left ??: B101 - B105

right ??: B101 - B105

rHadoop parallel computing for CF: Step 1

Step 1-4 is to calculate the similarity matrix.
Step 1: For each user, generate the item pairs. The return is the item pairs for each user:

- **item.x**: you can consider it as the ***rownames*** of the item similarity matrix
- **item.y**: can be considered as the ***colnames*** of the item similarity matrix

Result of step 1

```
> step1RAM$val
   user item.x pref.x item.y pref.y
1     1    101    5.0    101    5.0
2     1    101    5.0    102    3.0
3     1    101    5.0    103    2.5
4     1    102    3.0    101    5.0
5     1    102    3.0    102    3.0
6     1    102    3.0    103    2.5
7     1    103    2.5    101    5.0
8     1    103    2.5    102    3.0
9     1    103    2.5    103    2.5
10    2    101    2.0    101    2.0
11    2    101    2.0    102    2.5
12    2    101    2.0    103    5.0
13    2    101    2.0    104    2.0
14    2    102    2.5    101    2.0
15    2    102    2.5    102    2.5
16    2    102    2.5    103    5.0
17    2    102    2.5    104    2.0
18    2    103    5.0    101    2.0
19    2    103    5.0    102    2.5
20    2    103    5.0    103    5.0
```

Step 1: What Data looks like in RAM and HDFS

Data in RAM

user	item	pref
1	101	5
1	102	3
1	103	2.5
2	101	2
2	102	2.5
2	103	5
2	104	2
3	101	2
3	104	4
…	…	…

Data in HDFS

user	item	pref
1	101	5
1	102	3
2	101	2

Node 1

user	item	pref
2	103	5
2	104	2
3	101	2

Node 2

user	item	pref
1	103	2.5
3	104	4
…	…	…

Node 3

Revolution Analytics

Step 1: Matrix multiplying: MapReduce Job 1

map: set the user as the key

Data in HDFS

key
1
1
2

user	item	pref
1	101	5
1	102	3
2	101	2

key
2
2
3

user	item	pref
2	103	5
2	104	2
3	101	2

key
1
3
…

user	item	pref
1	103	2.5
3	104	4
…	…	…

shuffer

…

…

…

…

reduce input

user	item	pref
1	101	5
1	102	3
1	103	2.5

user	item	pref
2	101	2
2	102	2.5
2	103	5
2	104	2

user	item	pref
3	101	2
3	104	4
…	…	…

Merge by user

Merge by user

Merge by user

reduce output

user.x	Item.x	Pref.x	User.y	Item.y	Pref.y
1	101	5	1	101	5
1	101	5	1	102	3
1	101	5	1	102	3
1	102	3	1	101	5
1	102	3	1	102	3
1	102	3	1	102	3
1	103	2.5	1	101	5
1	103	2.5	1	102	3
1	103	2.5	1	102	3

……..

……..

Step 1: Matrix multiplying: MapReduce Job 1

map: set the user as the key

Data in HDFS

key
1
1
2

user	item	pref
1	101	5
1	102	3
2	101	2

key
2
2
3

user	item	pref
2	103	5
2	104	2
3	101	2

key
1
3
…

user	item	pref
1	103	2.5
3	104	4
…	…	…

shuffer

…

…

…

…

reduce input

user	item	pref
1	101	5
1	102	3
1	103	2.5

Merge by user

user	item	pref
2	101	2
2	102	2.5
2	103	5
2	104	2

Merge by user

user	item	pref
3	101	2
3	104	4
…	…	…

Merge by user

reduce output

Set as Keys and write the result to hdfs

user .x	Ite m.x	Pref .x	Use .y	Ite m.y	.y
1	101	5	1	101	5
1	101	5	1	102	3
1	101	5	1	102	3
1	102	3	1	101	5
1	102	3	1	102	3
1	102	3	1	102	3
1	103	2.5	1	101	5
1	103	2.5	1	102	3
1	103	2.5	1	102	3

……..

……..

Step 1: R Code

Read the data and put the data into hdfs

```r
### read/generate the data
#dataCF <- generateDataFun()
dataCF <- read.table("/home/binbin.chen/rHadoop/data/
  small.txt",header=F,sep=",")
names(dataCF)<-c("user","item","pref")

#### use the hadoop backend
rmr.options(backend = 'hadoop')
## The data path in hdfs
dataPathHdfs <- "/user/binbin.chen/MovieLens/data"
#### put the data into hdfs
step0hdfs <- file.path(dataPathHdfs, "dataCF")
##hdfs.del(step0hdfs)
to.dfs(dataCF,output=step0hdfs)
```

```
> dataCF
   user item pref
1     1  101  5.0
2     1  102  3.0
3     1  103  2.5
4     2  101  2.0
5     2  102  2.5
6     2  103  5.0
7     2  104  2.0
8     3  101  2.0
9     3  104  4.0
10    3  105  4.5
11    3  107  5.0
12    4  101  5.0
13    4  103  3.0
14    4  104  4.5
```

Mapreduce job 1:

```r
step1hdfs <- file.path(dataPathHdfs,"step1")
mapreduce(input=step0hdfs, output=step1hdfs,
  map= function(k,v) keyval(v$user,v),
  reduce=function(k,v){
  m = merge(v,v,by="user")
  keyval(m$item.x,m)
})
step1RAM <- from.dfs(step1hdfs)
```

- Read the data and put the data into hdfs
- Map: set the “user” as the keys
- Reduce: merge the items

```
> step1RAM$val
   user item.x pref.x item.y pref.y
1     1    101    5.0    101    5.0
2     1    101    5.0    102    3.0
3     1    101    5.0    103    2.5
4     1    102    3.0    101    5.0
5     1    102    3.0    102    3.0
6     1    102    3.0    103    2.5
7     1    103    2.5    101    5.0
8     1    103    2.5    102    3.0
9     1    103    2.5    103    2.5
10    2    101    2.0    101    2.0
11    2    101    2.0    102    2.5
12    2    101    2.0    103    5.0
13    2    101    2.0    104    2.0
14    2    102    2.5    101    2.0
15    2    102    2.5    102    2.5
16    2    102    2.5    103    5.0
17    2    102    2.5    104    2.0
18    2    103    5.0    101    2.0
19    2    103    5.0    102    2.5
20    2    103    5.0    103    5.0
```

Key

rHadoop parallel computing for CF: Step 2

Step 2: calculate the product the each :

- **item.x**: you can consider it as the ***rownames*** of the item similarity matrix
- **item.y**: can be considered as the ***colnames*** of the item similarity matrix
- Key: item.y

```
> t(dataCF2)%*%dataCF2
     101   102   103   104  105 106  107
101 74.0 32.00 45.50 50.50 23.0  36 10.0
102 32.0 24.25 26.00 17.00 10.5  12  0.0
103 45.5 26.00 44.25 31.50  7.0  20  0.0
104 50.5 17.00 31.50 56.25 32.0  34 20.0
105 23.0 10.50  7.00 32.00 32.5  14 22.5
106 36.0 12.00 20.00 34.00 14.0  32  0.0
107 10.0  0.00  0.00 20.00 22.5   0 25.0
```

compare

Return of step 2

```
> step2RAM$val
   item.x item.y inProd module.x
1     101    101  74.00 8.602325
2     101    102  32.00 8.602325
3     101    103  45.50 8.602325
4     101    104  50.50 8.602325
5     101    105  23.00 8.602325
6     101    106  36.00 8.602325
7     101    107  10.00 8.602325
8     102    101  32.00 4.924429
9     102    102  24.25 4.924429
10    102    103  26.00 4.924429
11    102    104  17.00 4.924429
12    102    105  10.50 4.924429
13    102    106  12.00 4.924429
14    103    101  45.50 6.652067
15    103    102  26.00 6.652067
16    103    103  44.25 6.652067
17    103    104  31.50 6.652067
18    103    105   7.00 6.652067
19    103    106  20.00 6.652067
20    104    101  50.50 7.500000
```

Matrix multiplying: MapReduce Job 2

map: Do nothing

Data in HDFS

Key	user.x	Item.x	Pref.x	User.y	Item.y	Pref.y
101	1	101	5	1	102	3
101	1	101	5	1	102	3
102	1	102	3	1	102	3

Item.x	user.x	Item.x	Pref.x	User.y	Item.y	Pref.y
101	2	101	2	2	101	2
101	2	101	2	2	102	2.5
101	2	101	2	2	103	5
103	1	103	2.5	1	101	5

Key	user.x	Item.x	Pref.x	User.y	Item.y	Pref.y
101	2	101	2	2	103	5
101	2	101	2	2	104	2
101	1	101	5	1	101	5

Item.x	user.x	Item.x	Pref.x	User.y	Item.y	Pref.y
…	…	…	…	…	…	…

shuffer

…

…

…

…

reduce input

user	item.x	pref.x	item.y	pref.y
1	101	5	101	5
1	101	5	102	3
1	101	5	103	2.5
2	101	2	101	2
2	101	2	102	2.5
2	101	2	103	5
2	101	2	104	2
3	101	2	101	2
3	101	2	104	4
3	101	2	105	4.5
3	101	2	107	5
4	101	5	101	5
4	101	5	103	3
4	101	5	104	4.5
4	101	5	106	4
5	101	4	101	4
5	101	4	102	3
5	101	4	103	2
5	101	4	104	4
5	101	4	105	3.5
5	101	4	106	4

user	item.x	pref.x	item.y	pref.y
1	102	5	101	5
…	…	…	…	…

ddply

item.x	item.y	inProd
101	101	74

Matrix multiplying: MapReduce Job 2

map: Do nothing

Data in HDFS

Key	user.x	Item.x	Pref.x	User.y	Item.y	Pref.y
101	1	101	5	1	102	3
101	1	101	5	1	102	3
102	1	102	3	1	102	3

Item.x	user.x	Item.x	Pref.x	User.y	Item.y	Pref.y
101	2	101	2	2	101	2
101	2	101	2	2	102	2.5
101	2	101	2	2	103	5
103	1	103	2.5	1	101	5

Key	user.x	Item.x	Pref.x	User.y	Item.y	Pref.y
101	2	101	2	2	103	5
101	2	101	2	2	104	2
101	1	101	5	1	101	5

Item.x	user.x	Item.x	Pref.x	User.y	Item.y	Pref.y
…	…	…	…	…	…	…

shuffer

…

…

…

…

reduce input

user	item.x	pref.x	item.y	pref.y
1	101	5	101	5
1	101	5	102	3
1	101	5	103	2.5
2	101	2	101	2
2	101	2	102	2.5
2	101	2	103	5
2	101	2	104	2
3	101	2	101	2
3	101	2	104	4
3	101	2	105	4.5
3	101	2	107	5
4	101	5	101	5
4	101	5	103	3
4	101	5	104	4.5
4	101	5	106	4
5	101	4	101	4
5	101	4	102	3
5	101	4	103	2
5	101	4	104	4
5	101	4	105	3.5
5	101	4	106	4

user	item.x	pref.x	item.y	pref.y
1	102	5	101	5
…	…	…	…	…

ddply

item.x	item.y	inProd
101	101	74
101	102	32

Matrix multiplying: MapReduce Job 2

map: Do nothing

Data in HDFS

Key
101
101
102

user.x	Item.x	Pref.x	User.y	Item.y	Pref.y
1	101	5	1	102	3
1	101	5	1	102	3
1	102	3	1	102	3

Item.x
101
101
101
103

user.x	Item.x	Pref.x	User.y	Item.y	Pref.y
2	101	2	2	101	2
2	101	2	2	102	2.5
2	101	2	2	103	5
1	103	2.5	1	101	5

Key
101
101
101

user.x	Item.x	Pref.x	User.y	Item.y	Pref.y
2	101	2	2	103	5
2	101	2	2	104	2
1	101	5	1	101	5

Item.x
…

user.x	Item.x	Pref.x	User.y	Item.y	Pref.y
…	…	…	…	…	…

shuffer

…

…

…

…

reduce input

user	item.x	pref.x	item.y	pref.y
1	101	5	101	5
1	101	5	102	3
1	101	5	103	2.5
2	101	2	101	2
2	101	2	102	2.5
2	101	2	103	5
2	101	2	104	2
3	101	2	101	2
3	101	2	104	4
3	101	2	105	4.5
3	101	2	107	5
4	101	5	101	5
4	101	5	103	3
4	101	5	104	4.5
4	101	5	106	4
5	101	4	101	4
5	101	4	102	3
5	101	4	103	2
5	101	4	104	4
5	101	4	105	3.5
5	101	4	106	4

user	item.x	pref.x	item.y	pref.y
1	102	5	101	5
…	…	…	…	…

ddply

item.x	item.y	inProd
101	101	74
101	102	32
101	103	45.5
101	104	50.5
101	105	23
101	106	36
101	107	10

REVOLUTION ANALYTICS

Matrix multiplying: MapReduce Job 2

map: Do nothing

Data in HDFS

Key
101
101
102

use r.x	Ite m.x	Pref .x	Use r.y	Ite m.y	Pref .y
1	101	5	1	102	3
1	101	5	1	102	3
1	102	3	1	102	3

Ite m.x
101
101
101
103

use r.x	Ite m.x	Pref .x	Use r.y	Ite m.y	Pref .y
2	101	2	2	101	2
2	101	2	2	102	2.5
2	101	2	2	103	5
1	103	2.5	1	101	5

Key
101
101
101

user .x	Ite m.x	Pref .x	Use r.y	Ite m.y	Pref .y
2	101	2	2	103	5
2	101	2	2	104	2
1	101	5	1	101	5

Ite m.x
…

user .x	Ite m.x	Pref .x	Use r.y	Ite m.y	Pref .y
…	…	…	…	…	…

shuffer

…

…

…

…

reduce input

user	item.x	pref.x	item.y	pref.y
1	101	5	101	5
1	101	5	102	3
1	101	5	103	2.5
2	101	2	101	2
2	101	2	102	2.5
2	101	2	103	5
2	101	2	104	2
3	101	2	101	2
3	101	2	104	4
3	101	2	105	4.5
3	101	2	107	5
4	101	5	101	5
4	101	5	103	3
4	101	5	104	4.5
4	101	5	106	4
5	101	4	101	4
5	101	4	102	3
5	101	4	103	2
5	101	4	104	4
5	101	4	105	3.5
5	101	4	106	4

user	item.x	pref.x	item.y	pref.y
1	102	5	101	5
…	…	…	…	…

ddply

item.x	item.y	inProd	module.x
101	101	74	74
101	102	32	74
101	103	45.5	74
101	104	50.5	74
101	105	23	74
101	106	36	74
101	107	10	74

REVOLUTION ANALYTICS

Matrix multiplying: MapReduce Job 2

map: Do nothing

Data in HDFS

Key	user.x	item.x	pref.x	user.y	item.y	pref.y
101	1	101	5	1	102	3
101	1	101	5	1	102	3
102	1	102	3	1	102	3

item.x	user.x	item.x	pref.x	user.y	item.y	pref.y
101	2	101	2	2	101	2
101	2	101	2	2	102	2.5
101	2	101	2	2	103	5
103	1	103	2.5	1	101	5

Key	user.x	item.x	pref.x	user.y	item.y	pref.y
101	2	101	2	2	103	5
101	2	101	2	2	104	2
101	1	101	5	1	101	5

item.x	user.x	item.x	pref.x	user.y	item.y	pref.y
…	…	…	…	…	…	…

shuffer

…

…

…

…

reduce input

user	item.x	pref.x	item.y	pref.y
1	101	5	101	5
1	101	5	102	3
1	101	5	103	2.5
2	101	2	101	2
2	101	2	102	2.5
2	101	2	103	5
2	101	2	104	2
3	101	2	101	2
3	101	2	104	4
3	101	2	105	4.5
3	101	2	107	5
4	101	5	101	5
4	101	5	103	3
4	101	5	104	4.5
4	101	5	106	4
5	101	4	101	4
5	101	4	102	3
5	101	4	103	2
5	101	4	104	4
5	101	4	105	3.5
5	101	4	106	4

user	item.x	pref.x	item.y	pref.y
1	102	5	101	5
…	…	…	…	…

Set as the Key

ddply

item.x	item.y	inProd	module.x
101	101	74	74
101	102	32	74
101	103	45.5	74
101	104	50.5	74
101	105	23	74
101	106	36	74
101	107	10	74

REVOLUTION ANALYTICS

R Code: MapReduce Job 2

```r
### mapreduce job2: pref.x*pref.y and create the module of item.x
step2hdfs <- file.path(dataPathHdfs,"step2")
mapreduce(input=step2hdfs, output=step2hdfs,
  map=function(k,v) keyval(k,v),
  reduce=function(k,v) {
  val<-ddply(v,c("item.x","item.y"), function(x) sum(x$pref.x*x$pref.y))
  val$module.x <- sqrt(as.numeric(val[val$item.x==val$item.y,3]))
  names(val)[3] <- "inProd"
  keyval(val$item.y,val)})
step2RAM <- from.dfs(step2hdfs)
```

```
> step2RAM$val
   item.x item.y inProd module.x
1     101    101  74.00 8.602325
2     101    102  32.00 8.602325
3     101    103  45.50 8.602325
4     101    104  50.50 8.602325
5     101    105  23.00 8.602325
6     101    106  36.00 8.602325
7     101    107  10.00 8.602325
8     102    101  32.00 4.924429
9     102    102  24.25 4.924429
10    102    103  26.00 4.924429
11    102    104  17.00 4.924429
12    102    105  10.50 4.924429
13    102    106  12.00 4.924429
14    103    101  45.50 6.652067
15    103    102  26.00 6.652067
16    103    103  44.25 6.652067
17    103    104  31.50 6.652067
18    103    105   7.00 6.652067
19    103    106  20.00 6.652067
20    104    101  50.50 7.500000
```

Key

rHadoop parallel computing for CF: Step 3

Step 3: calculate module of the item.y:

- Using “item.y” as the key, in the reduce step, find the “inProd” with the same “item.x” and “item.y”

	item.x	item.y	inProd	module.x
1	101	101	74.00	8.602325
8	102	101	32.00	4.924429
14	103	101	45.50	6.652067
20	104	101	50.50	7.500000
27	105	101	23.00	5.700877
34	106	101	36.00	5.656854
40	107	101	10.00	5.000000
2	101	102	32.00	8.602325
9	102	102	24.25	4.924429
15	103	102	26.00	6.652067
21	104	102	17.00	7.500000
28	105	102	10.50	5.700877
35	106	102	12.00	5.656854
3	101	103	45.50	8.602325
10	102	103	26.00	4.924429
16	103	103	44.25	6.652067
22	104	103	31.50	7.500000
29	105	103	7.00	5.700877

In the reduce with key (item.y = 102)

```
### mapreduce job 3:
step3hdfs <- file.path(dataPathHdfs,"step3")
step3hdfs <- mapreduce(input=step2hdfs, output=step3hdfs,
  map=function(k,v) keyval(k,v),
  reduce=function(k,v){
    val <- v
    val$module.y <- sqrt(as.numeric(val[val$item.x==val$item.
    keyval(k,val)
    })
step3RAM <- from.dfs(step3hdfs)
```

Return of step 3

```
> step3RAM$val
```

	item.x	item.y	inProd	module.x	module.y
1	105	101	23.00	5.700877	8.602325
2	106	101	36.00	5.656854	8.602325
3	107	101	10.00	5.000000	8.602325
4	101	101	74.00	8.602325	8.602325
5	102	101	32.00	4.924429	8.602325
6	103	101	45.50	6.652067	8.602325
7	104	101	50.50	7.500000	8.602325
8	101	102	32.00	8.602325	4.924429
9	102	102	24.25	4.924429	4.924429
10	103	102	26.00	6.652067	4.924429
11	104	102	17.00	7.500000	4.924429
12	105	102	10.50	5.700877	4.924429
13	106	102	12.00	5.656854	4.924429
14	105	103	7.00	5.700877	6.652067
15	106	103	20.00	5.656854	6.652067
16	101	103	45.50	8.602325	6.652067
17	102	103	26.00	4.924429	6.652067
18	103	103	44.25	6.652067	6.652067
19	104	103	31.50	7.500000	6.652067
20	101	104	50.50	8.602325	7.500000
21	102	104	17.00	4.924429	7.500000
22	103	104	31.50	6.652067	7.500000
23	104	104	56.25	7.500000	7.500000
24	105	104	32.00	5.700877	7.500000
25	106	104	34.00	5.656854	7.500000

rHadoop parallel computing for CF: Step 4

Step 4: calculate the similarity

- Only map function, no reduce

```
### mapreduce job 4:
step4hdfs <- file.path(dataPathHdfs,"step4")
mapreduce(input=step3hdfs, output=step4hdfs,
  map=function(k,v){
    val <- v
    val$sim <- val$inProd/val$module.x/val$module.y
    keyval(val$item.y,val[,c("item.x","item.y","sim")])
    })
step4RAM <- from.dfs(step4hdfs)
```

Return of step 4

```
> step4RAM$val
   item.x item.y       sim
1     104    105 0.7484228
2     105    105 1.0000000
3     106    105 0.4341216
4     107    105 0.7893522
5     101    105 0.4689972
6     102    105 0.3740173
7     103    105 0.1845864
8     101    106 0.7397954
9     102    106 0.4307749
10    103    106 0.5314940
11    104    106 0.8013877
12    105    106 0.4341216
13    106    106 1.0000000
14    104    107 0.5333333
15    105    107 0.7893522
16    107    107 1.0000000
17    101    107 0.2324953
18    105    101 0.4689972
```

REVOLUTION ANALYTICS

rHadoop parallel computing for CF: Step 5

Step 5-6: Calculate the user's preference for each item.

- It's a matrix multiplying again. We did it when calculating the item similarity matrix.
- Item is the same, but different matrices.
- Using the function equijoin, and join by the "item.y"

Merge the two dataset by the item

> simMat

	101	102	103	104	105	106	107
101	1.00	0.76	0.80	0.78	0.47	0.74	0.23
102	0.76	1.00	0.79	0.46	0.37	0.43	0.00
103	0.80	0.79	1.00	0.63	0.18	0.53	0.00
104	0.78	0.46	0.63	1.00	0.75	0.80	0.53
105	0.47	0.37	0.18	0.75	1.00	0.43	0.79
106	0.74	0.43	0.53	0.80	0.43	1.00	0.00
107	0.23	0.00	0.00	0.53	0.79	0.00	1.00

%*%

> dataCF2

	101	102	103	104	105	106	107
1	5	3.0	2.5	0.0	0.0	0	0
2	2	2.5	5.0	2.0	0.0	0	0
3	2	0.0	0.0	4.0	4.5	0	5
4	5	0.0	3.0	4.5	0.0	4	0
5	4	3.0	2.0	4.0	3.5	4	0

(item.x, item.y)

(item.y, user)

Return of step 4

> step5RAM$val

	item.x.l	item.y.l	sim.l	user.r	item.r	pref.r
1	104	107	0.5333333	3	107	5.0
2	105	107	0.7893522	3	107	5.0
3	107	107	1.0000000	3	107	5.0
4	101	107	0.2324953	3	107	5.0
5	105	103	0.1845864	1	103	2.5
6	106	103	0.5314940	1	103	2.5
7	101	103	0.7951314	1	103	2.5
8	102	103	0.7937081	1	103	2.5
9	103	103	1.0000000	1	103	2.5
10	104	103	0.6313827	1	103	2.5
11	105	103	0.1845864	2	103	5.0
12	106	103	0.5314940	2	103	5.0
13	101	103	0.7951314	2	103	5.0
14	102	103	0.7937081	2	103	5.0
15	103	103	1.0000000	2	103	5.0
16	104	103	0.6313827	2	103	5.0
17	105	103	0.1845864	4	103	3.0
18	106	103	0.5314940	4	103	3.0
19	101	103	0.7951314	4	103	3.0
20	102	103	0.7937081	4	103	3.0
21	103	103	1.0000000	4	103	3.0
22	104	103	0.6313827	4	103	3.0
23	105	103	0.1845864	5	103	2.0
24	106	103	0.5314940	5	103	2.0
25	101	103	0.7951314	5	103	2.0

rHadoop parallel computing for CF: Step 6

Step 6: Calculate the user's preference for each item.

- Take the weighted sum of the user's preference with weight are the similarity of the items.
- Keys are the "item.x"

item.x.l	item.y.l	sim.l	user.r	item.r	pref.r
101	103	0.795131	1	103	2.5
101	101	1	1	101	5
101	102	0.755402	1	102	3
102	103	0.793708	1	103	2.5
102	101	0.755402	1	101	5
102	102	1	1	102	3
103	103	1	1	103	2.5
103	101	0.795131	1	101	5
103	102	0.793708	1	102	3
104	103	0.631383	1	103	2.5
104	101	0.782734	1	101	5
104	102	0.46029	1	102	3
105	103	0.184586	1	103	2.5
105	101	0.468997	1	101	5
105	102	0.374017	1	102	3
106	103	0.531494	1	103	2.5
106	101	0.739795	1	101	5
106	102	0.430775	1	102	3
107	101	0.232495	1	101	5
106	103	0.531494	2	103	5

Map: set the "item.x.l" as the keys

*Reduce: take the sum of sim.l*pref.r*

```
> step6RAM$val
  item user     pref
1  101    1 9.254035
2  102    1 8.761281
3  103    1 8.856781
4  104    1 6.872998
5  105    1 3.928504
6  106    1 6.320037
7  107    1 1.162476
8  101    2 9.429631
```

REVOLUTION ANALYTICS

Thanks