

Efficient Big Data Processing in Hadoop MapReduce

Jens Dittrich

Jorge-Arnulfo Quiané-Ruiz



COMPUTER SCIENCE

MapReduce Intro

Data Layouts

Job Optimization

Indexing

MapReduce

Intro

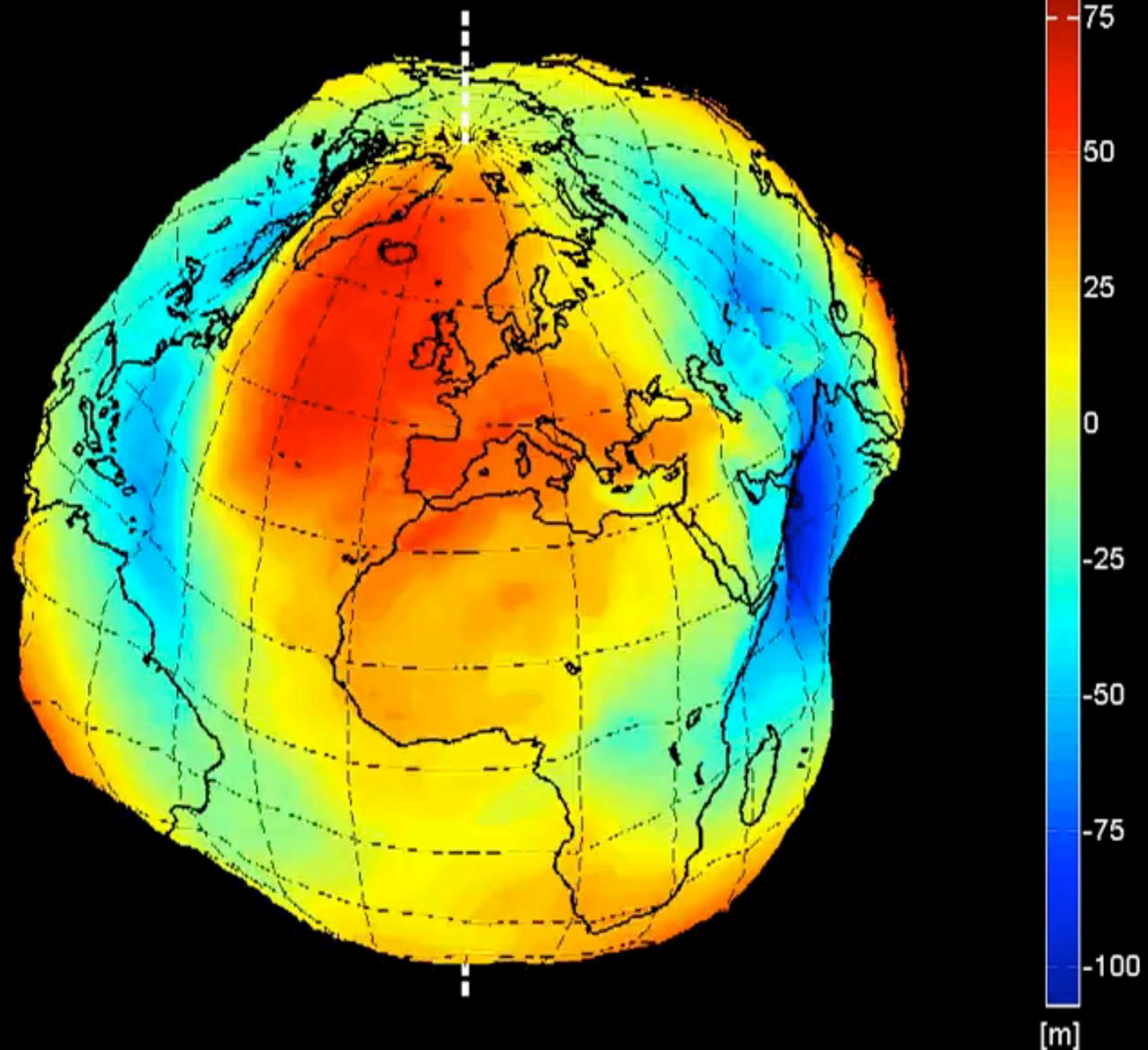
Big Data







GOCE Geoid undulations

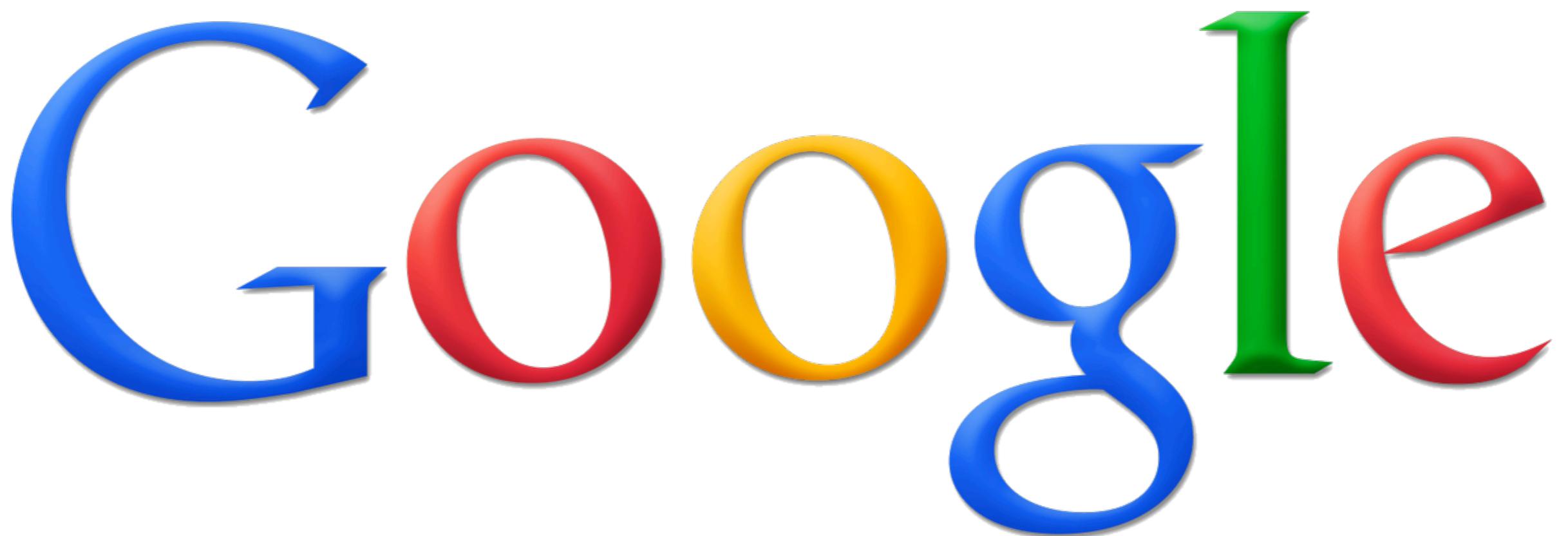






YAHOO!

facebook®

The Google logo consists of the word "Google" in a bold, sans-serif font. Each letter is a different color: the 'G' is blue, the first 'o' is red, the second 'o' is yellow, the 'g' is blue, the 'l' is green, and the 'e' is red. The letters are slightly rounded and have a three-dimensional, shadowed appearance.

[Dean et al, OSDI'04]

MapReduce



Semantics:

map(key, value) -> set of (ikey, ivalue)

reduce(ikey, set of ivalue) -> (fkey, fvalue)

Google-Use Case:

Web-Index

map(key, value)

->

set of (ikey, ivalue)

map(docID, document)

->

set of (term, docID)

```
map(44,  
    "This is text on a website!"  
)  
->  
{  
    ("This", 44),  
    ("is", 44),  
    ("text", 44),  
    ("on", 44),  
    ("a", 44),  
    ("website", 44)  
}
```

```
map(42,  
    "This is just another website!"  
)  
->  
{  
    ("This", 42),  
    ("is", 42),  
    ("just", 42),  
    ("another", 42),  
    ("website", 42)  
}
```

```
map(43,  
    "One more boring website!"  
)  
->  
{  
    ("One", 43),  
    ("more", 43),  
    ("boring", 43),  
    ("website", 43)  
}
```

reduce(ikey, set of ivalue)

->

(fkey, fvalue)

reduce(term, set of docID)

->

(term, (posting list of docID, count))

```
reduce(`This`,  
{42,  
 43}  
)  
->  
(`This`, ([42, 43], 2))
```

```
reduce(`is`,  
{42,  
 43}  
)  
->  
(`is`, ([42, 43], 2))
```

```
reduce(`boring`,
{43}
)
->
(`boring`, ([43], 1))
```

etc.

Other Applications:

Search

rec.a==42 or:

rec.contains(``bla``) or:

rec.contains(0011001)

Search

rec.a==42 or:

rec.contains(``bla``) or:

rec.contains(0011001)

Machine Learning
k-means, mahout library

Search

rec.a==42 or:

rec.contains(``bla``) or:

rec.contains(0011001)

Machine Learning
k-means, mahout library

Web-Analysis

Sum of all accesses to page
Y from user X

Search

rec.a==42 or:

rec.contains(``bla``) or:

rec.contains(0011001)

Machine Learning
k-means, mahout library

Web-Analysis

Sum of all accesses to page

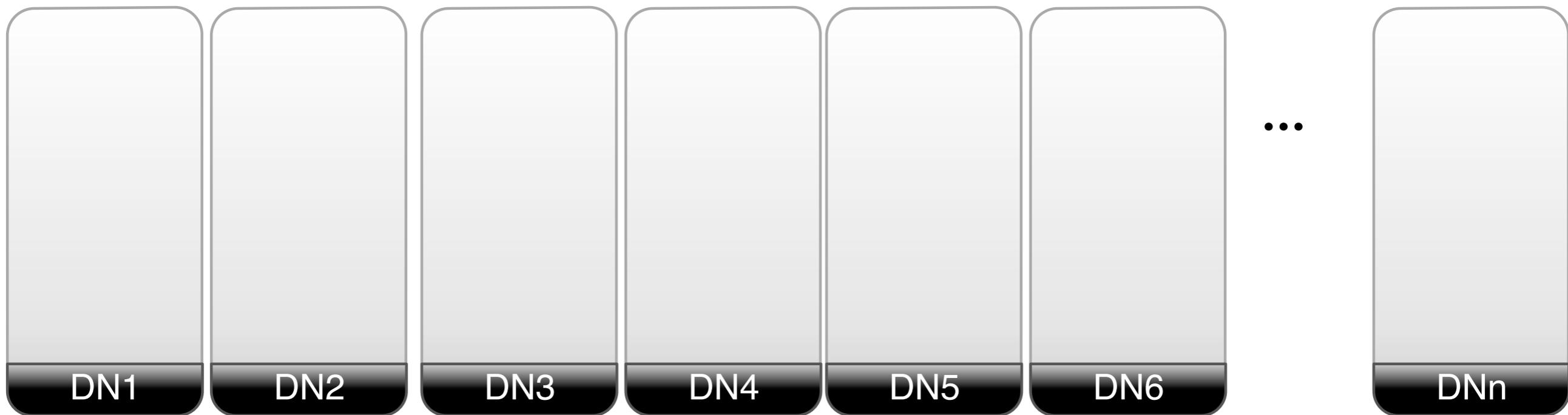
Y from user X

etc.

map() and reduce() with

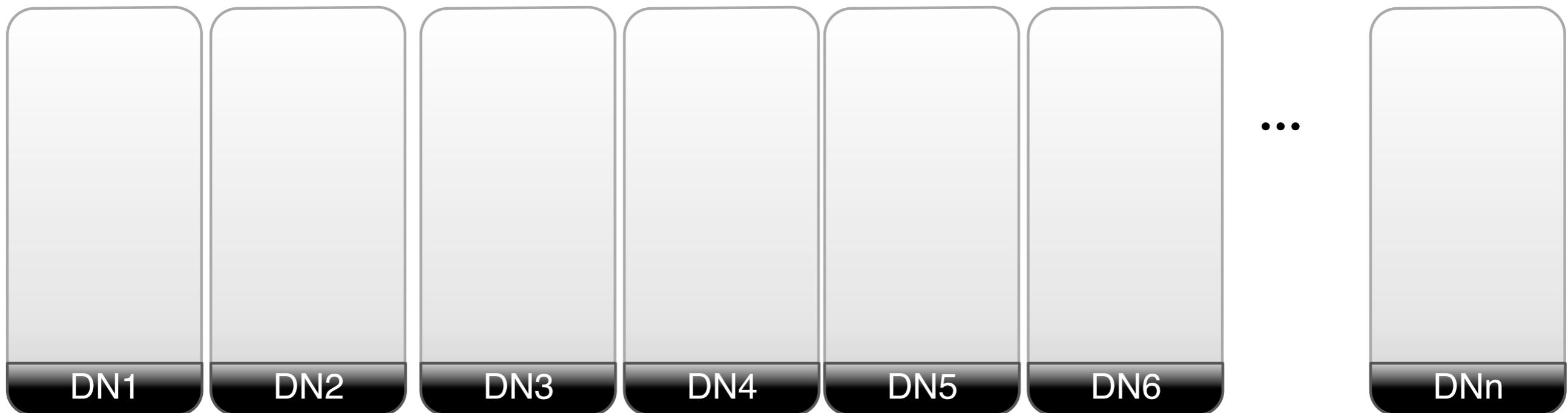
Big Data ?

HDFS



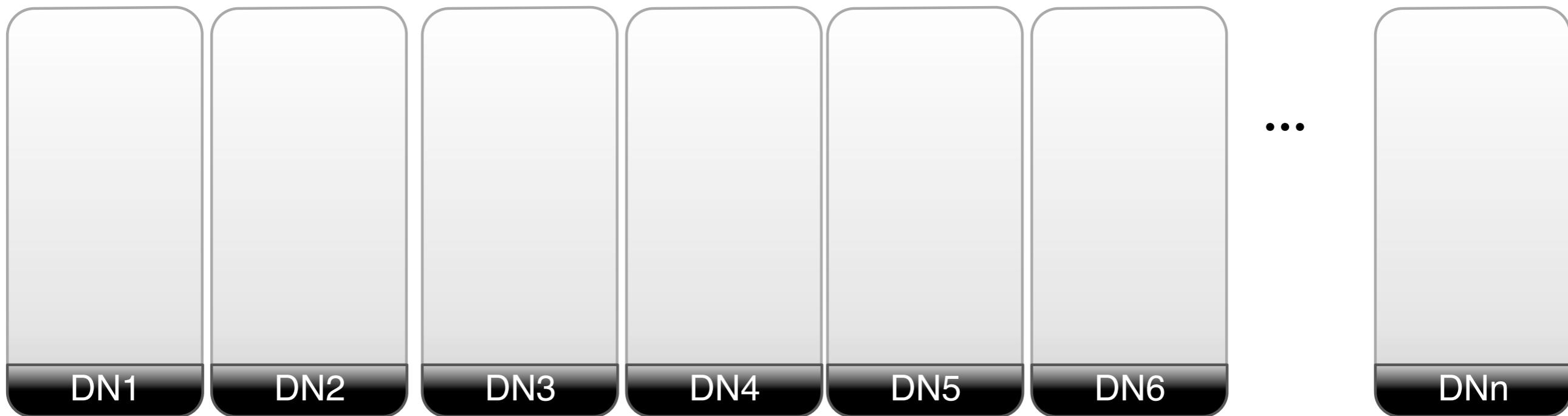


HDFS



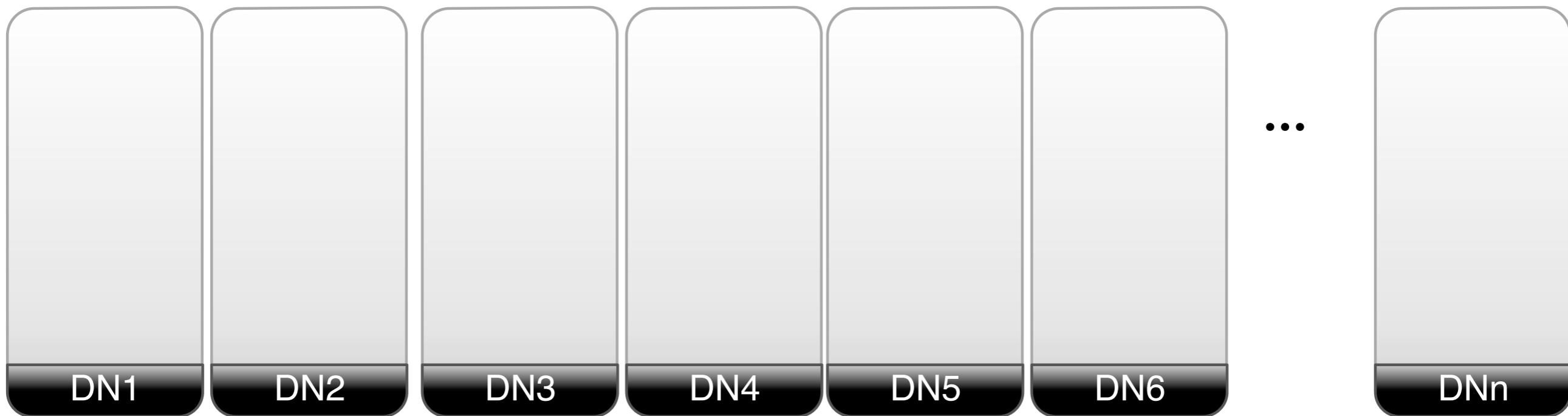


HDFS





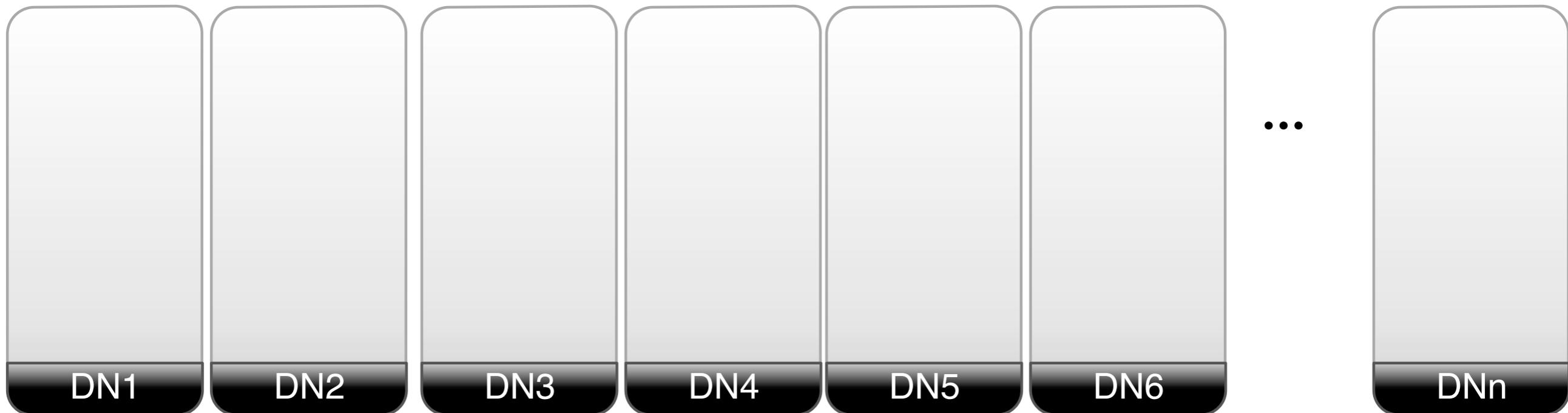
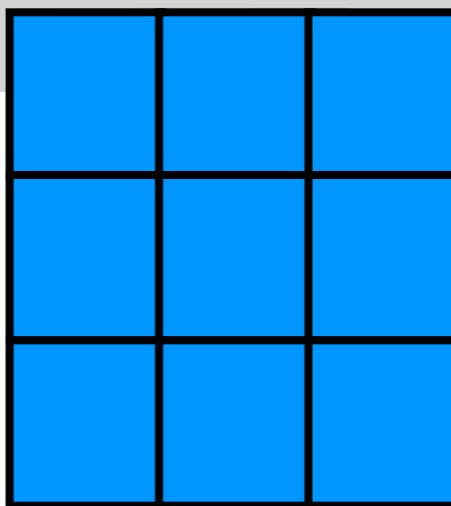
HDFS





HDFS

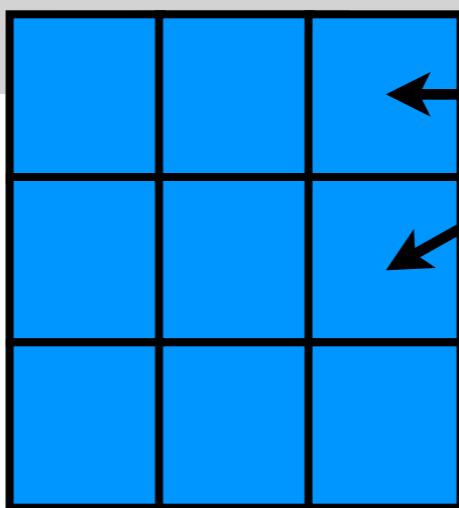
horizontal partitions



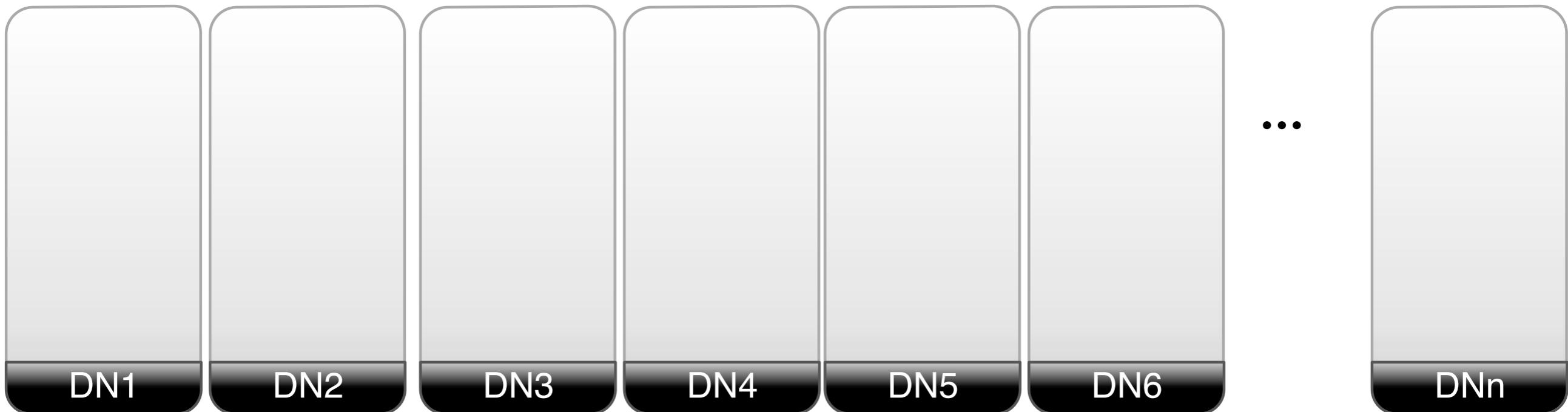


HDFS

horizontal partitions



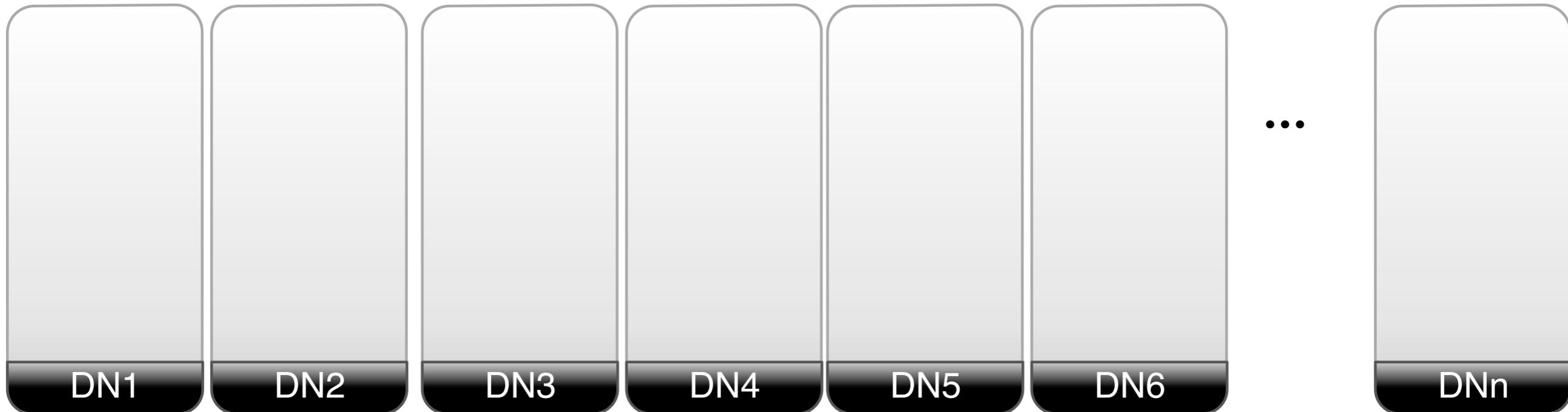
HDFS blocks
64MB (default)





HDFS

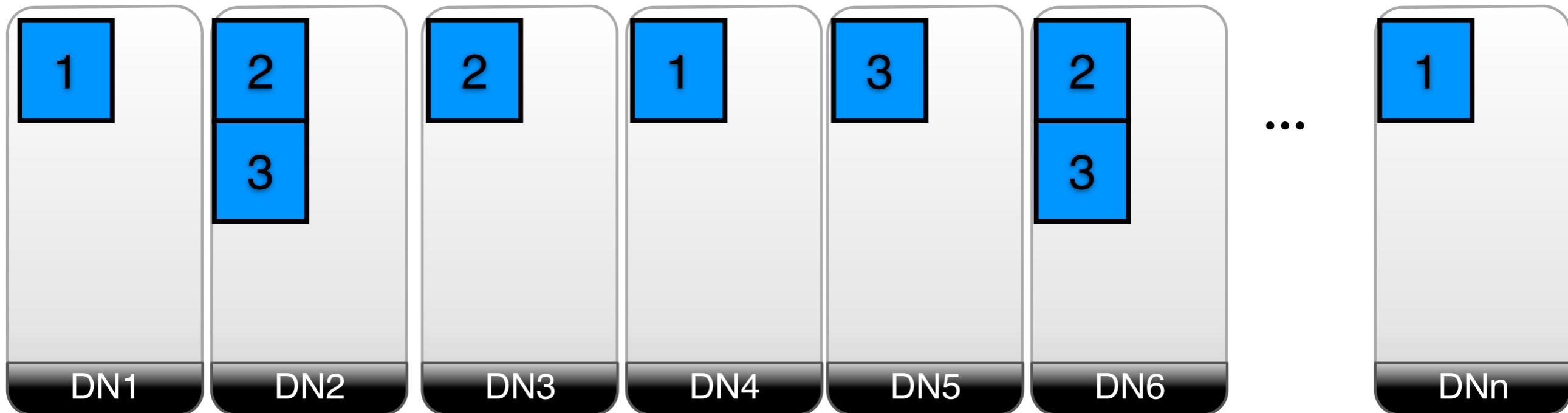
1	2	3
4	5	6
7	8	9





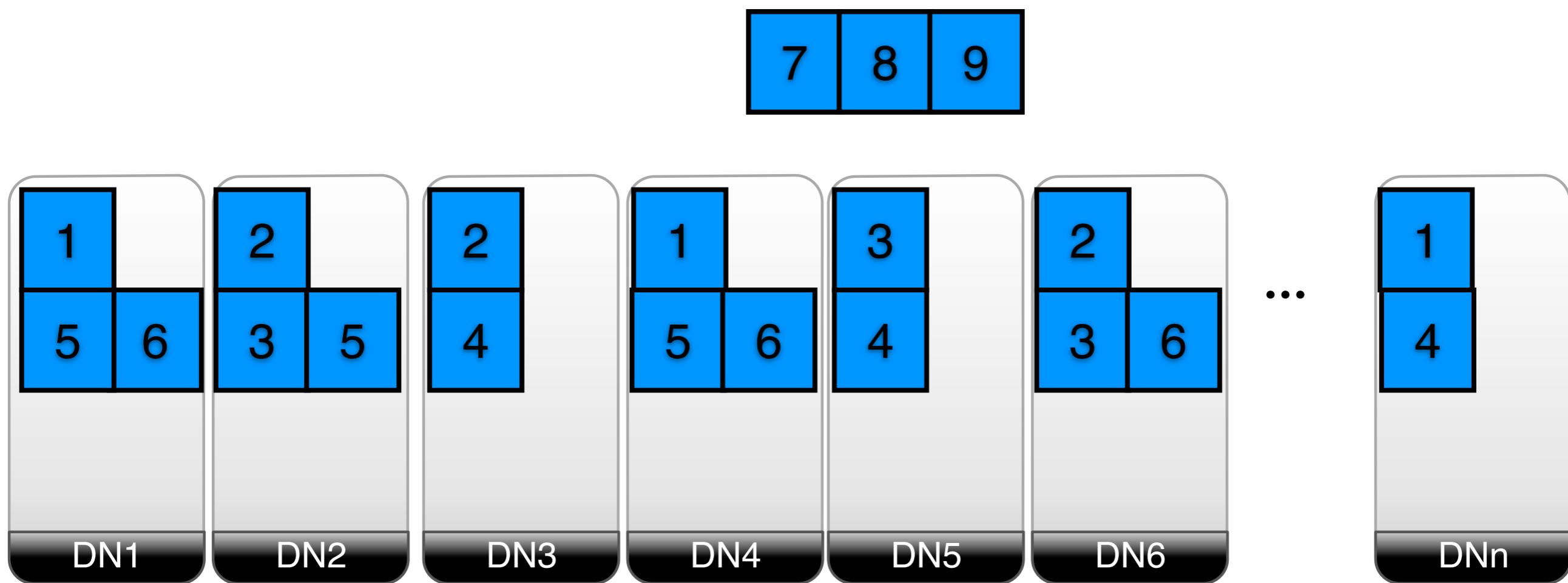
HDFS

4	5	6
7	8	9



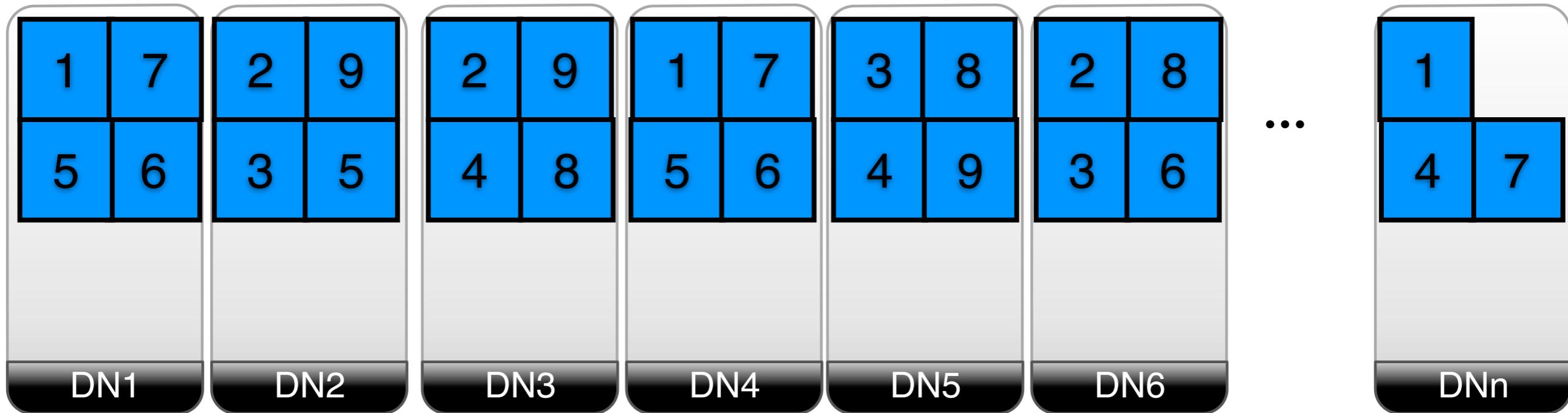


HDFS



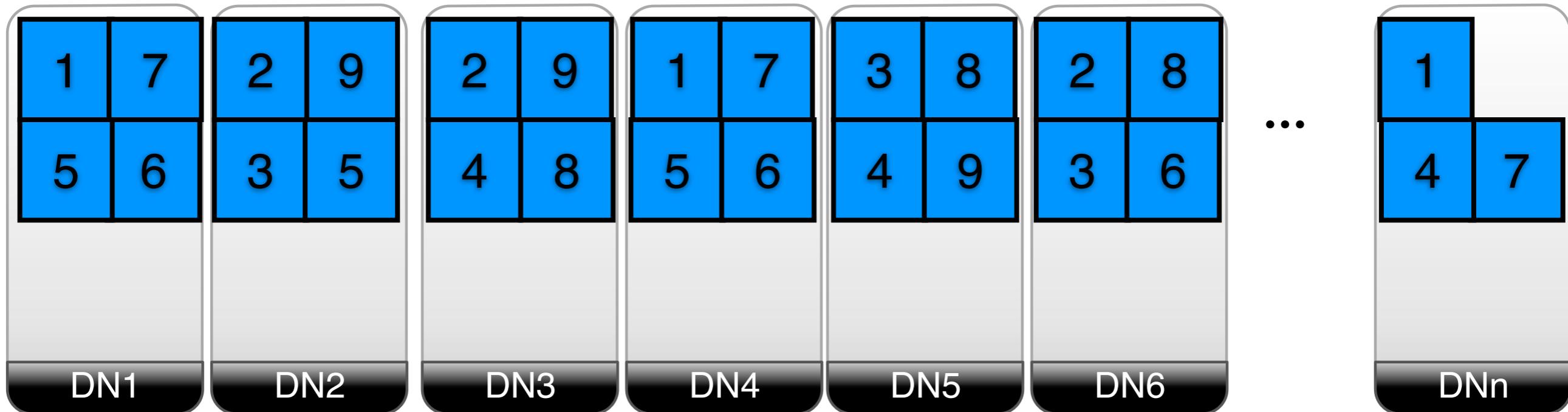


HDFS



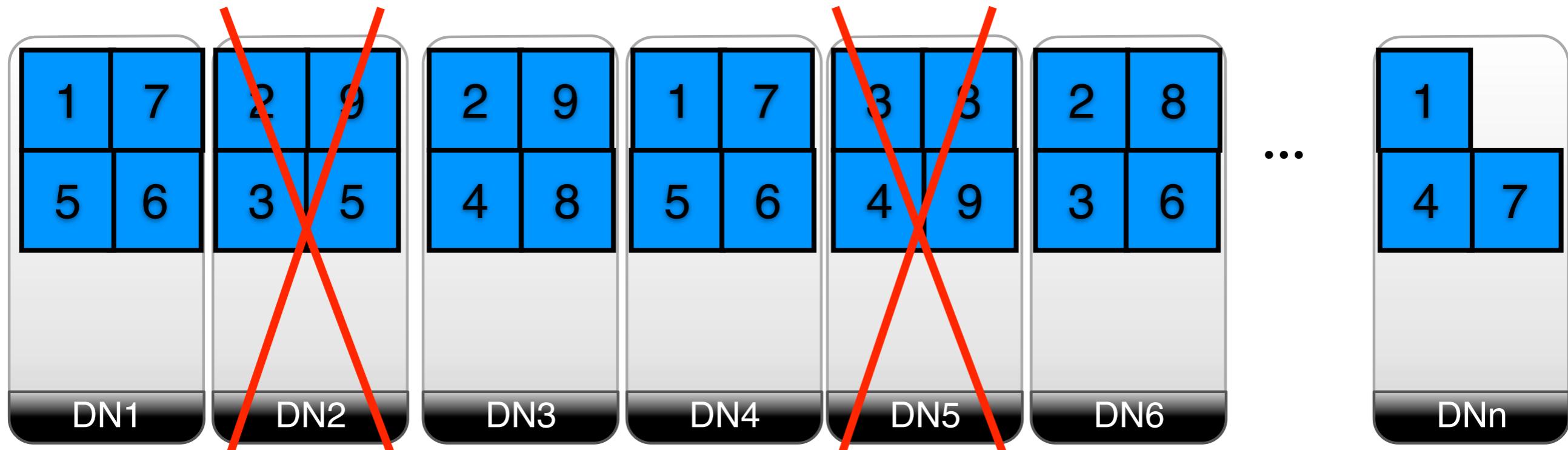
Failover

HDFS



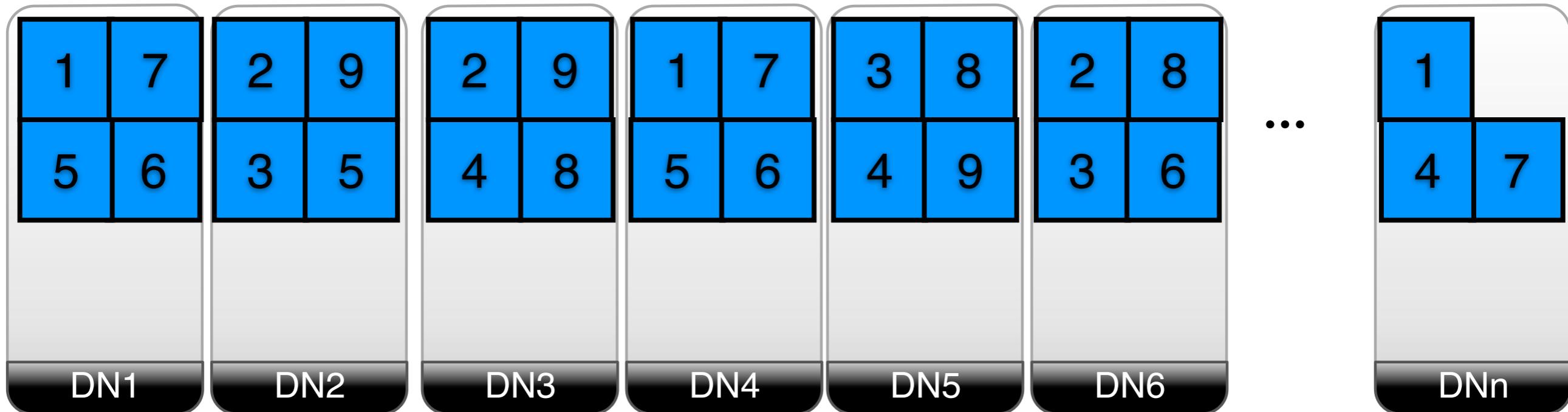
Failover

HDFS



Load Balancing

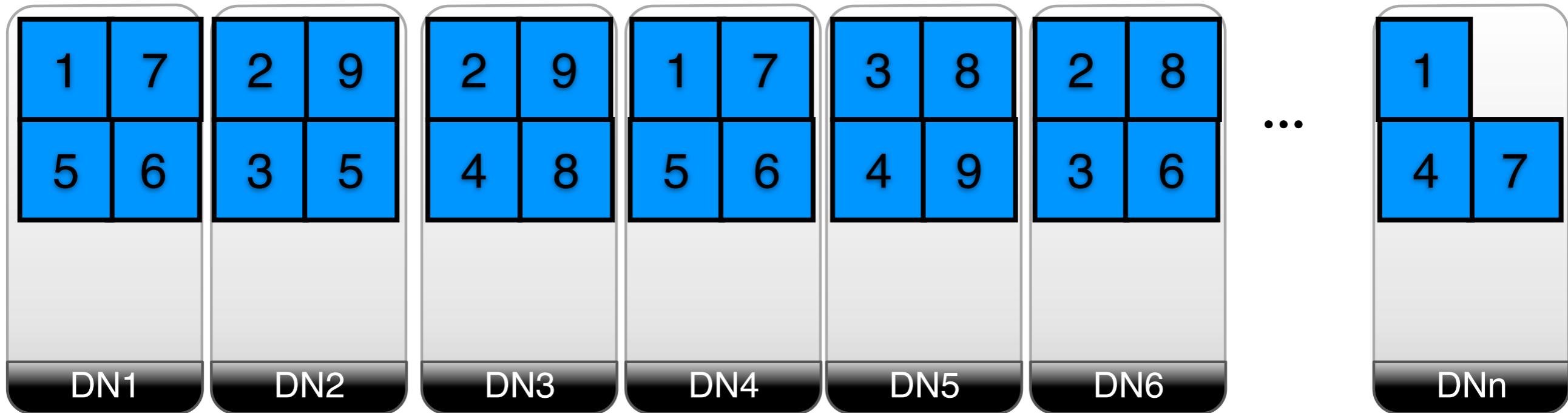
HDFS



Load Balancing



HDFS

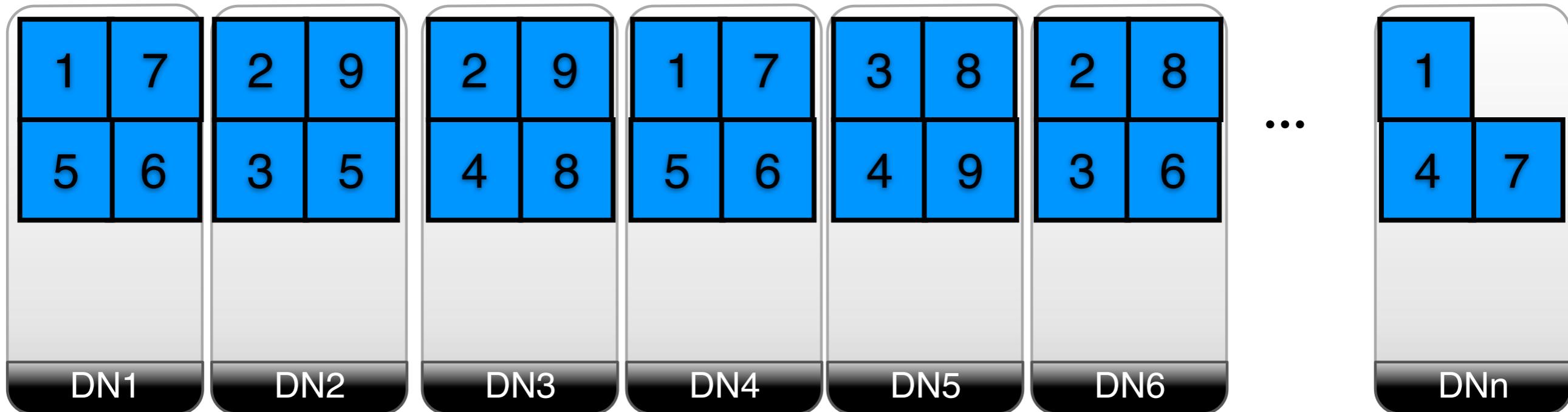


Load Balancing



I would like to have block 4!

HDFS



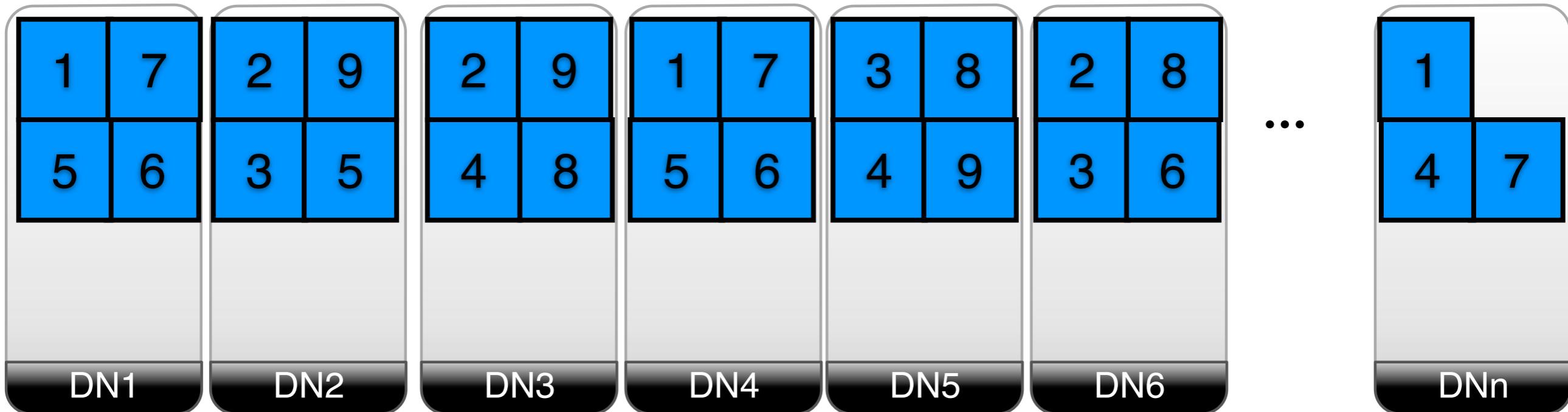
Load Balancing



I would like to have block 4!



HDFS

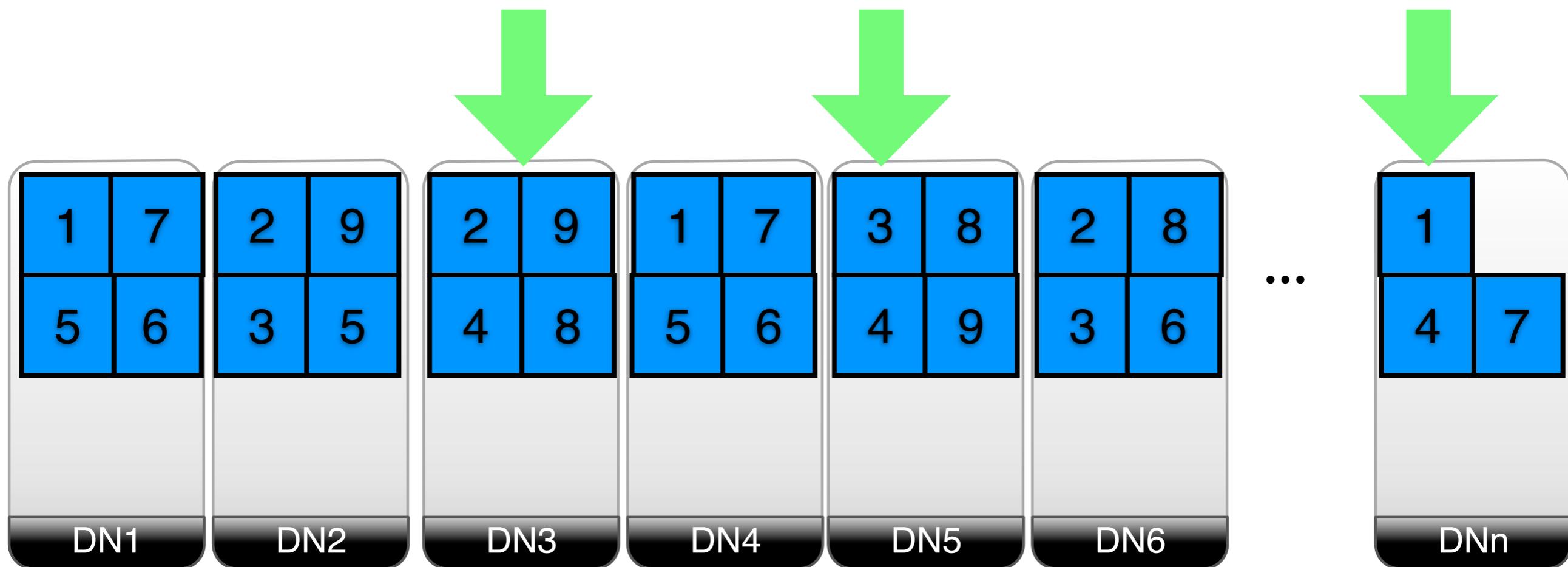


Load Balancing



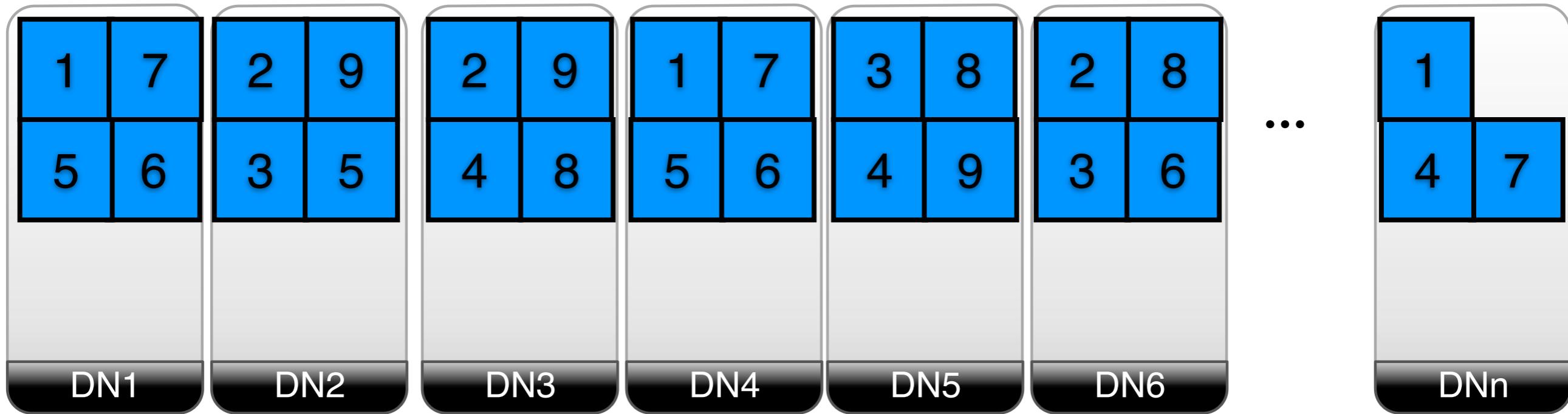
I would like to
have block 4!

HDFS



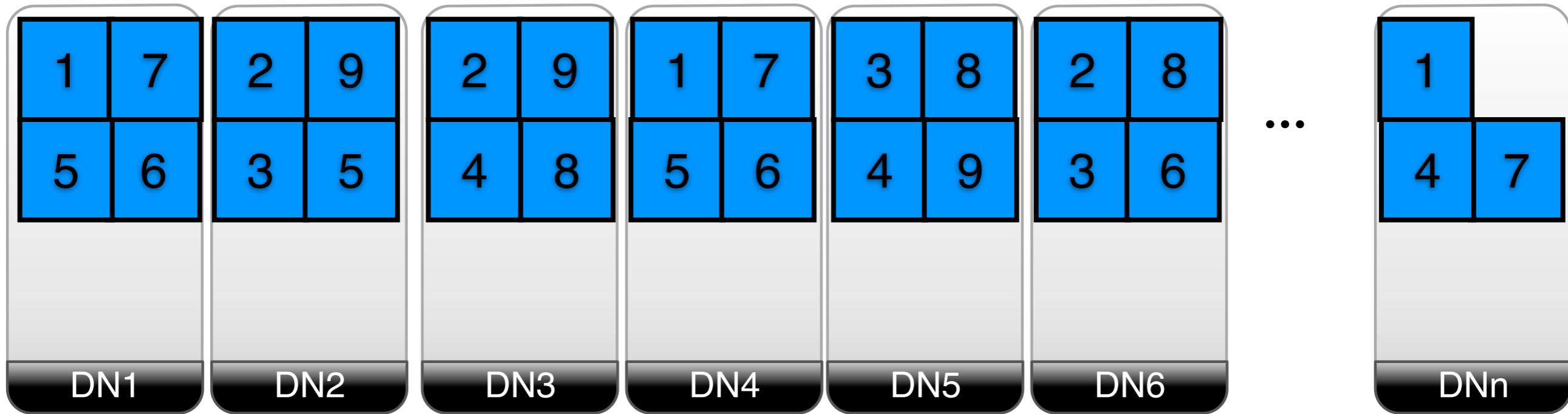


HDFS





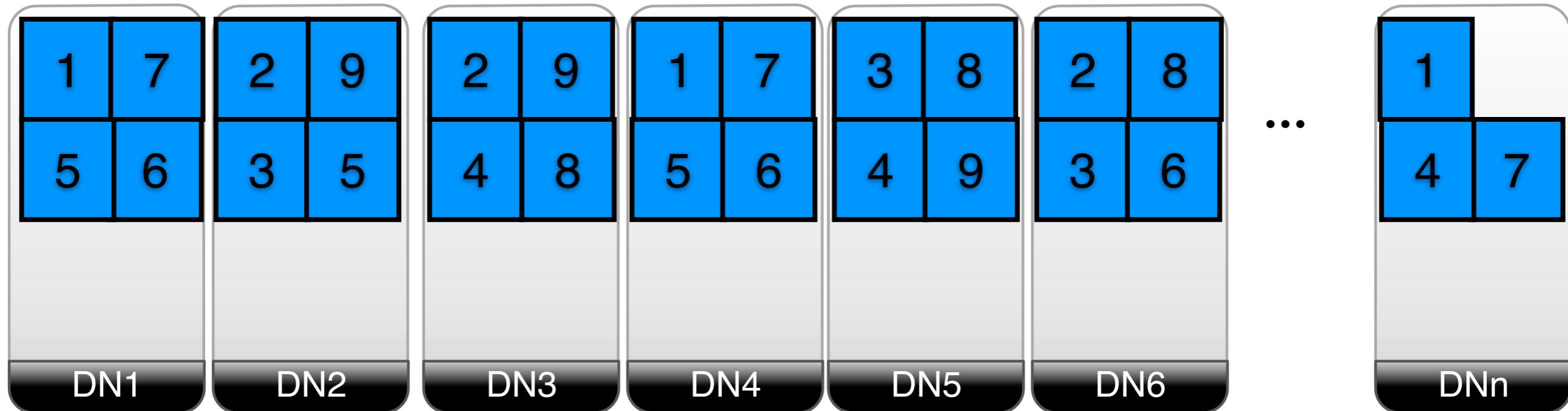
HDFS





MapReduce

HDFS

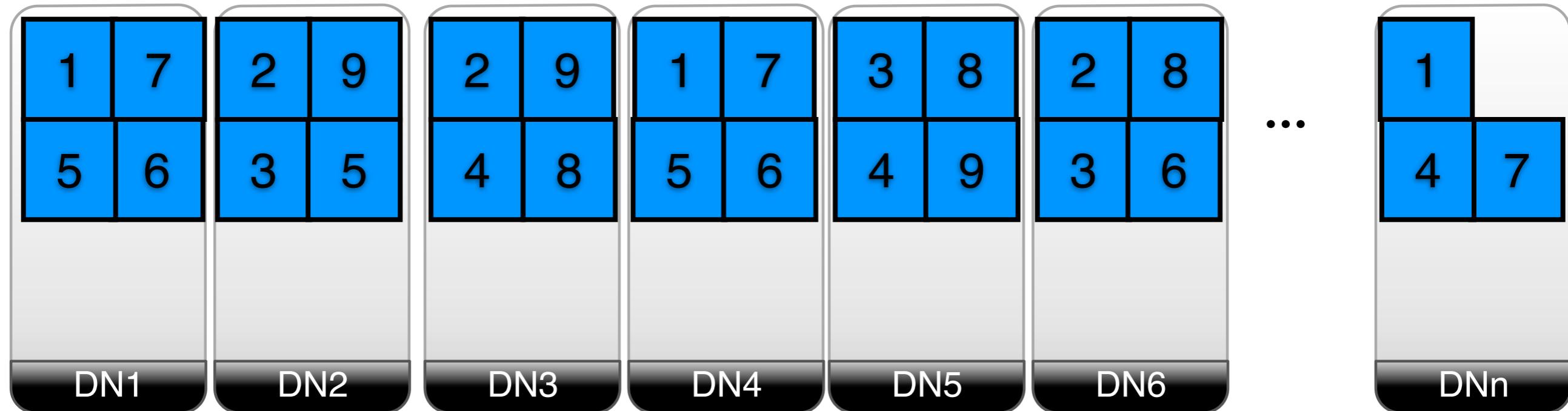




map(docID, document) -> set of (term, docID)

MapReduce

HDFS



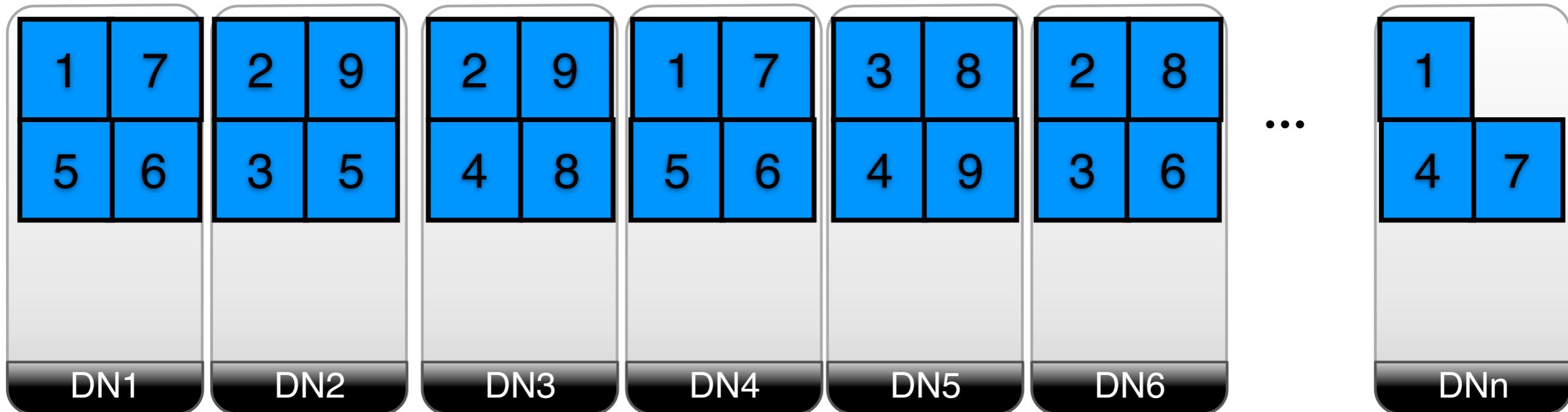
Map Phase



MapReduce

map(docID, document) -> set of (term, docID)

HDFS



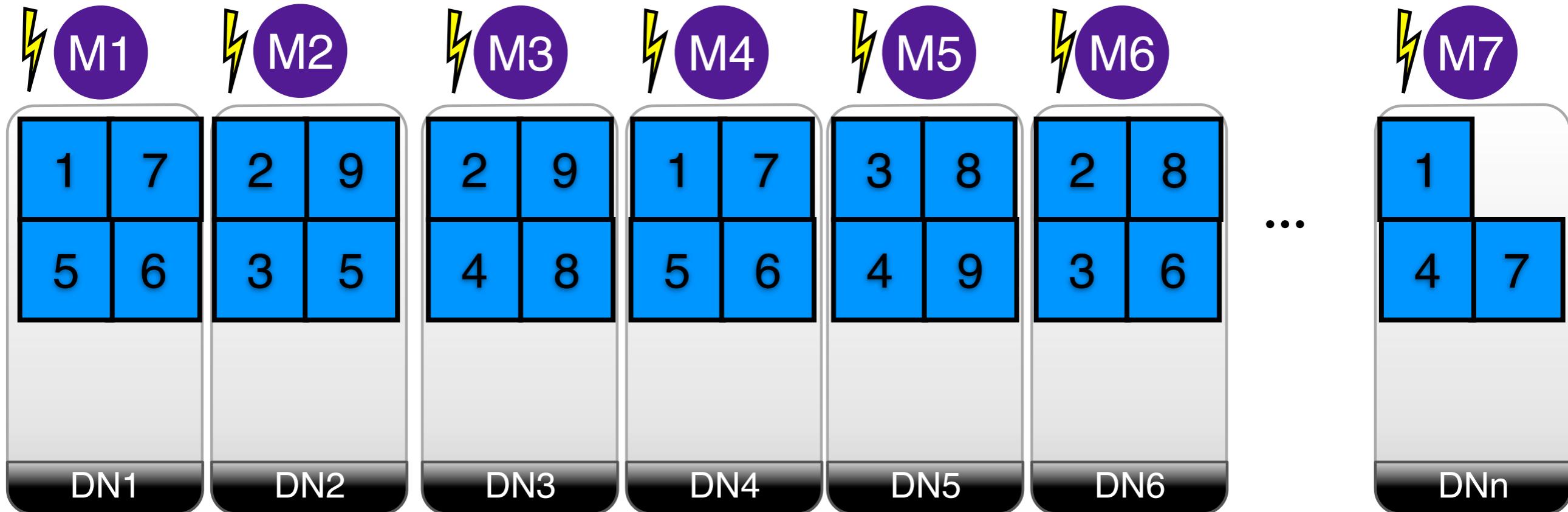
Map Phase



MapReduce

map(docID, document) -> set of (term, docID)

HDFS



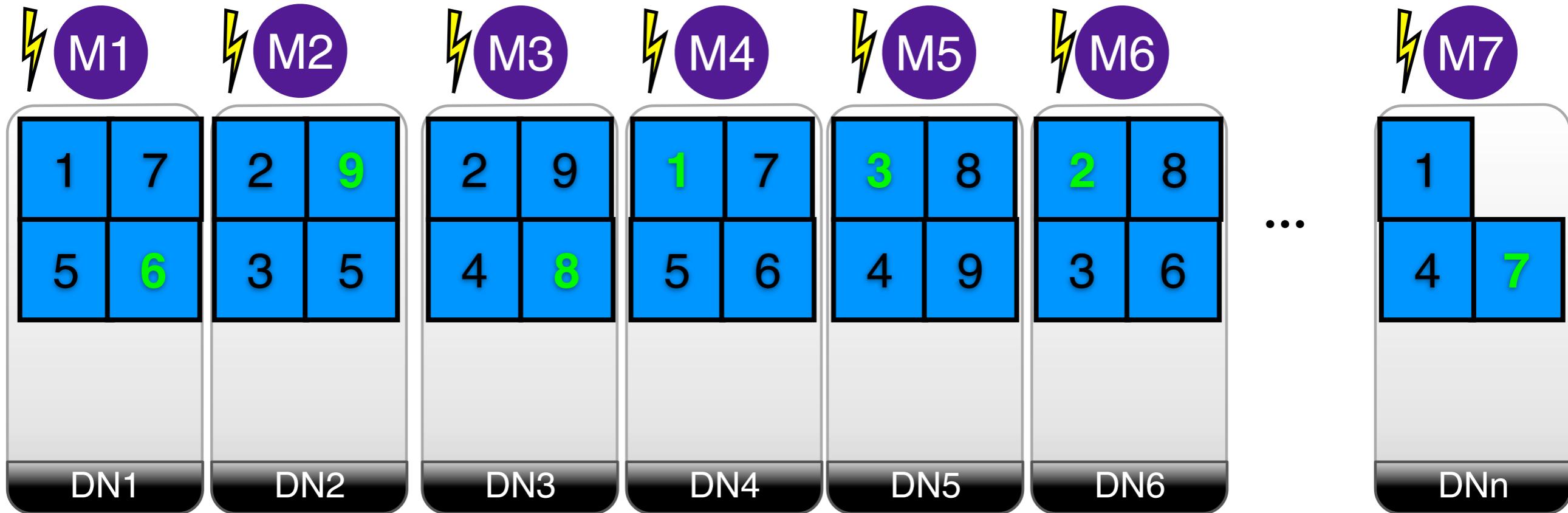
Map Phase



MapReduce

map(docID, document) -> set of (term, docID)

HDFS



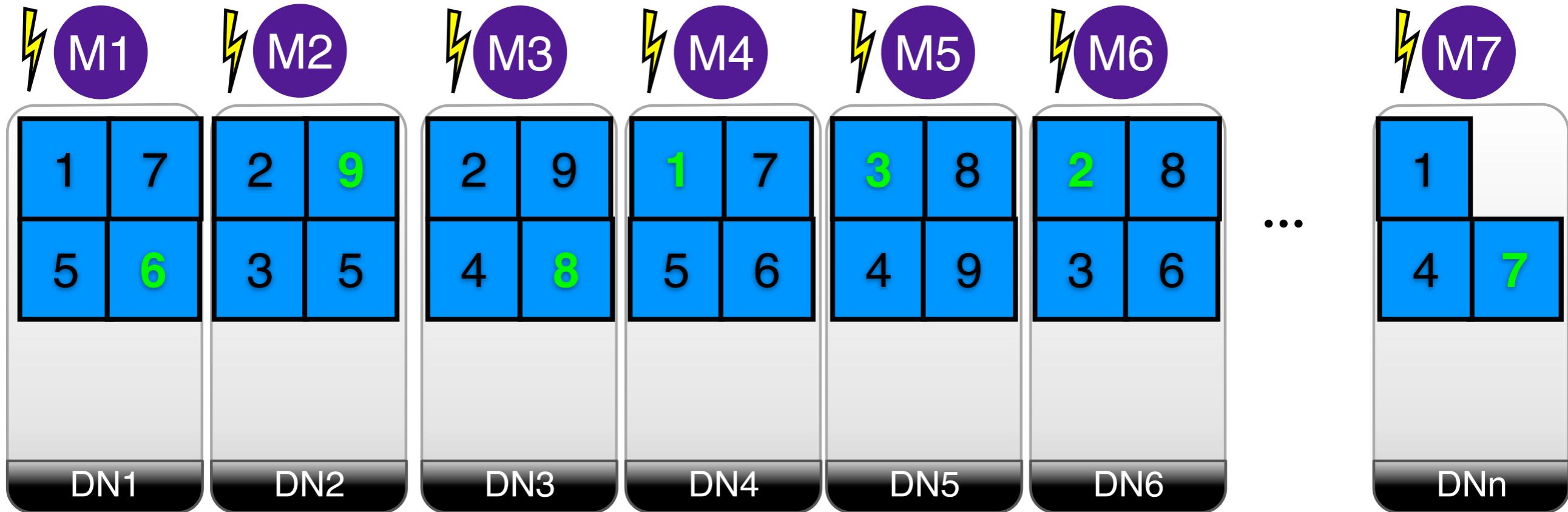
Map Phase



MapReduce

map(docID, document) -> set of (term, docID)

HDFS



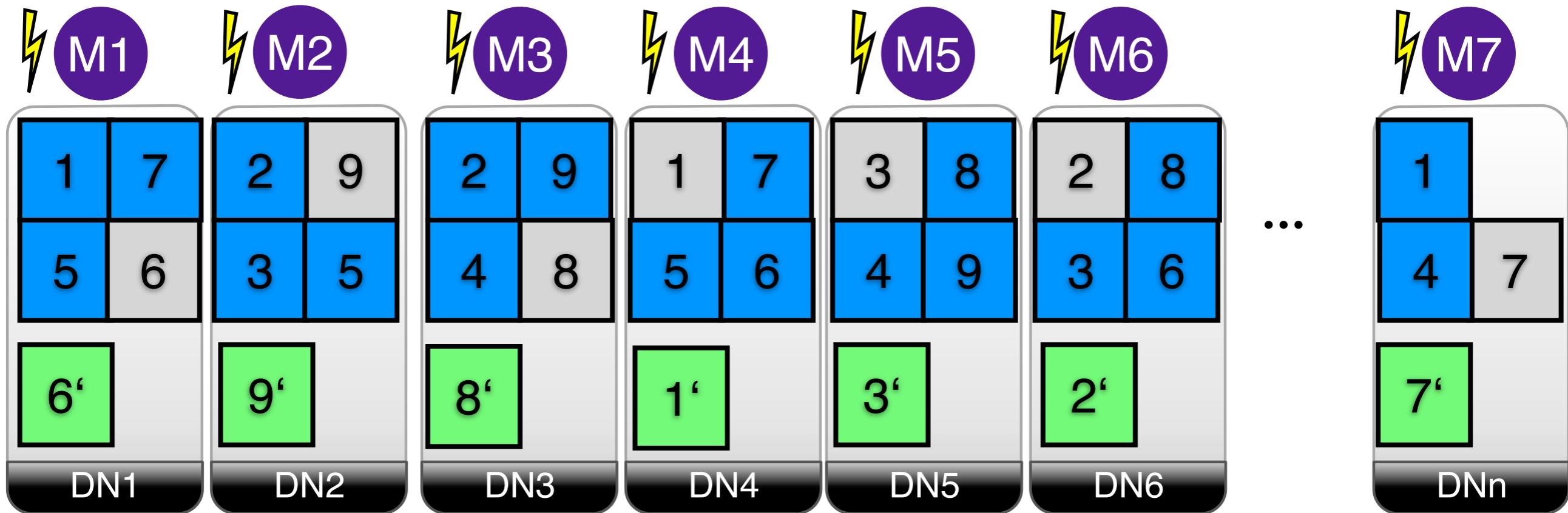
Map Phase



MapReduce

map(docID, document) -> set of (term, docID)

HDFS



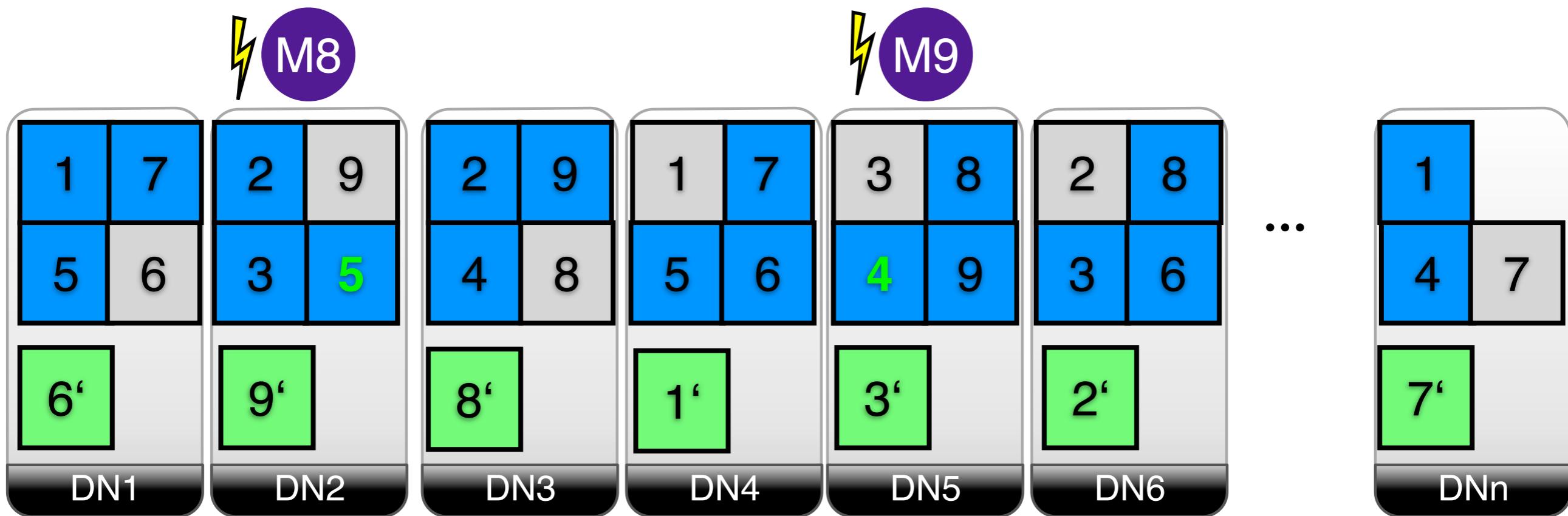
Map Phase



MapReduce

map(docID, document) -> set of (term, docID)

HDFS



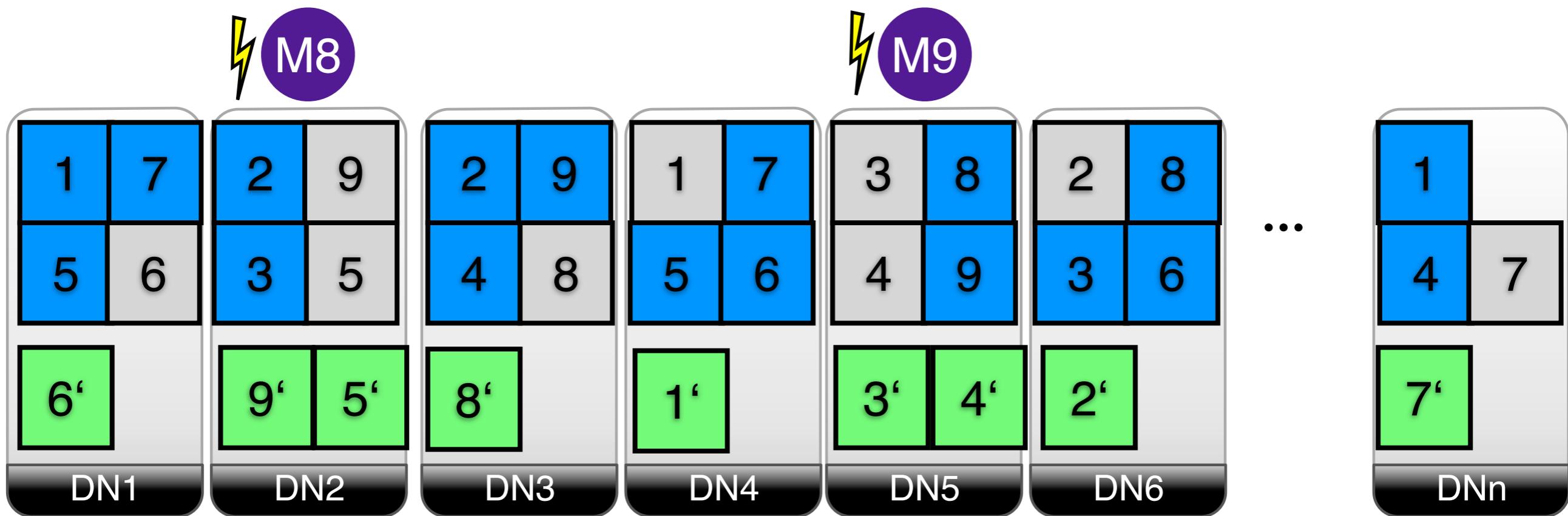
Map Phase



MapReduce

map(docID, document) -> set of (term, docID)

HDFS



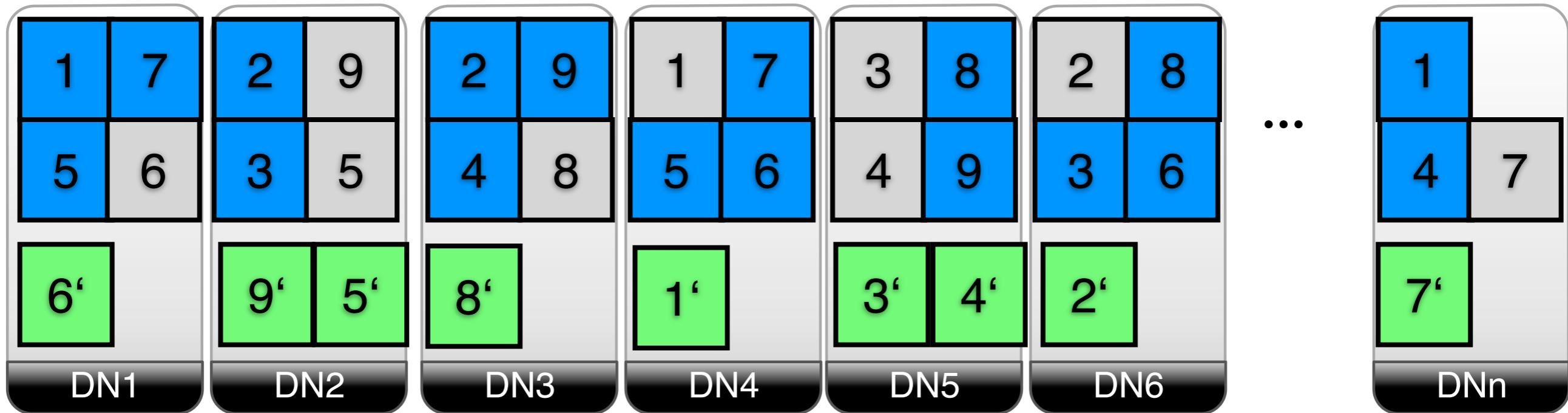
Map Phase



MapReduce

map(docID, document) -> set of (term, docID)

HDFS



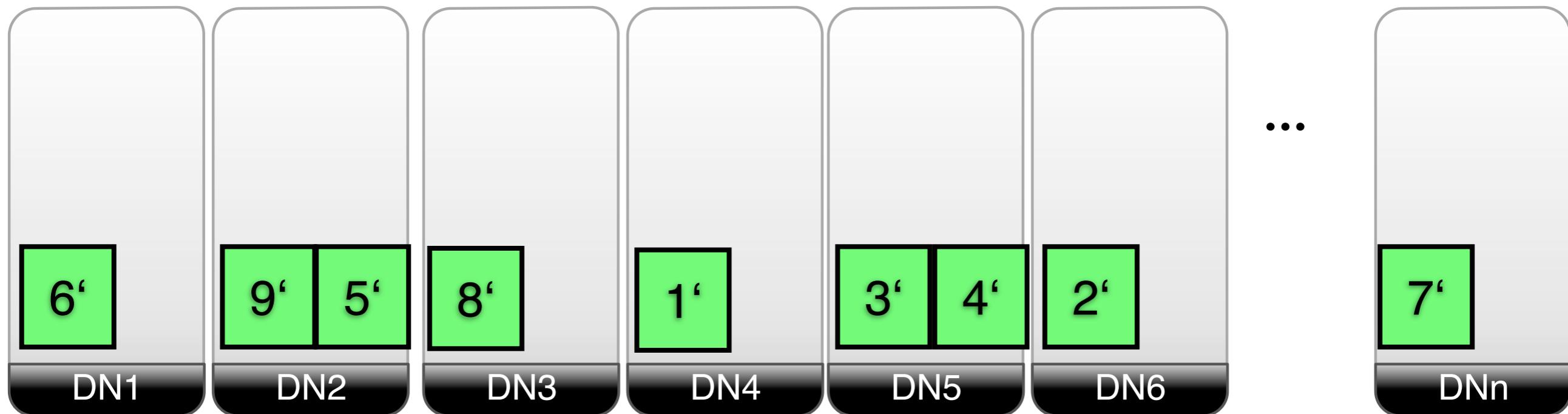
Shuffle Phase



MapReduce

group by term

HDFS



Shuffle Phase

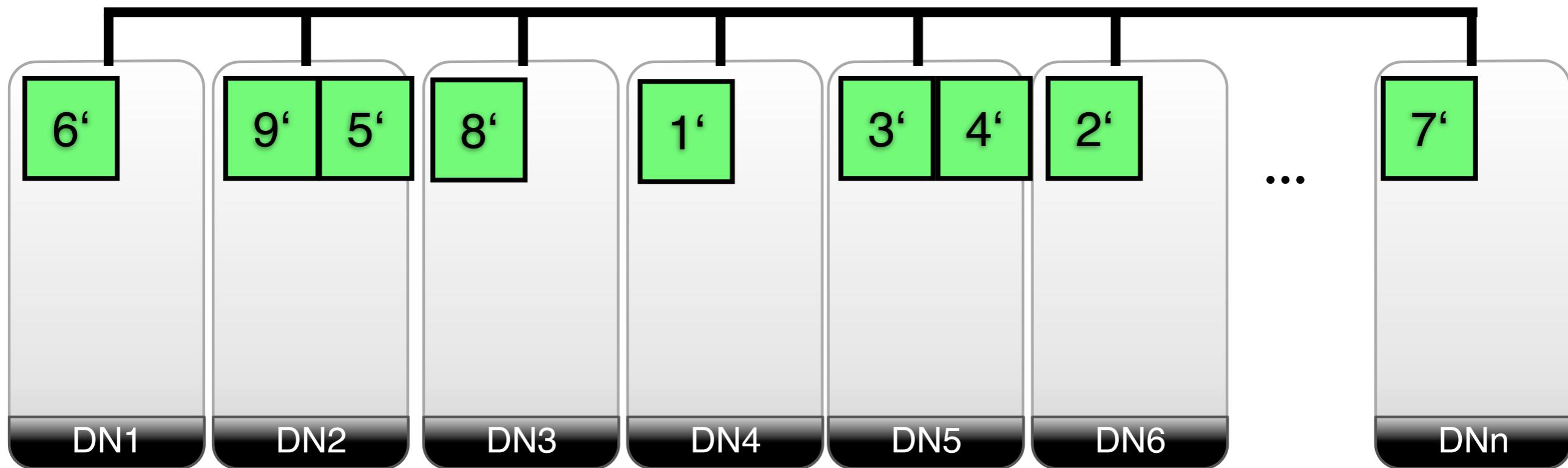


MapReduce

group by term

HDFS

network



Shuffle Phase

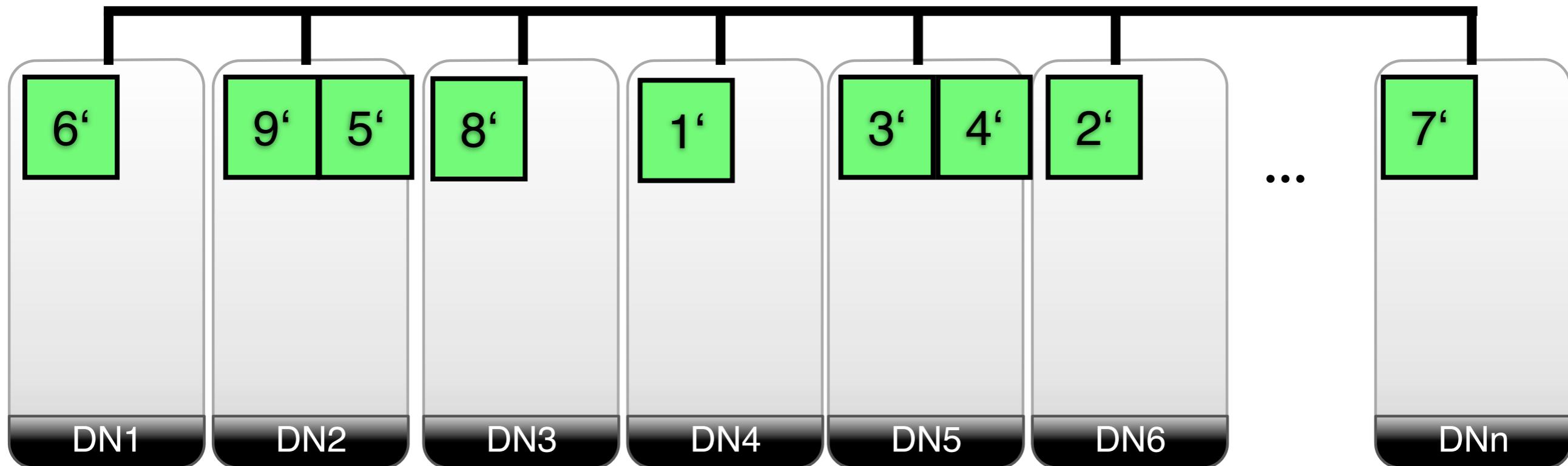


MapReduce

group by term

HDFS

network



Shuffle Phase

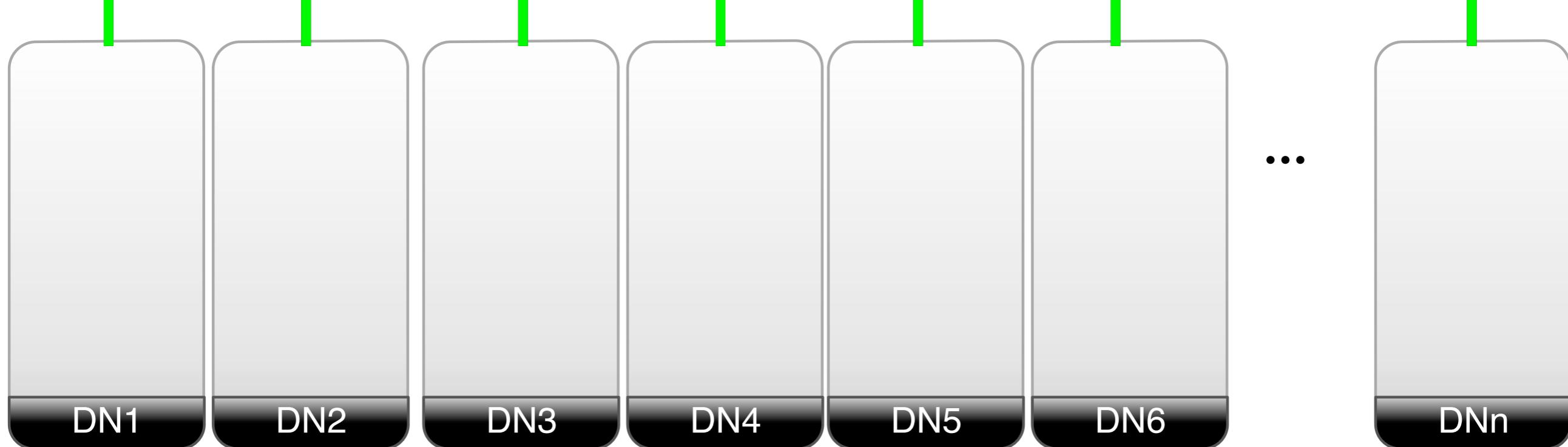


MapReduce

group by term

HDFS

network



Shuffle Phase

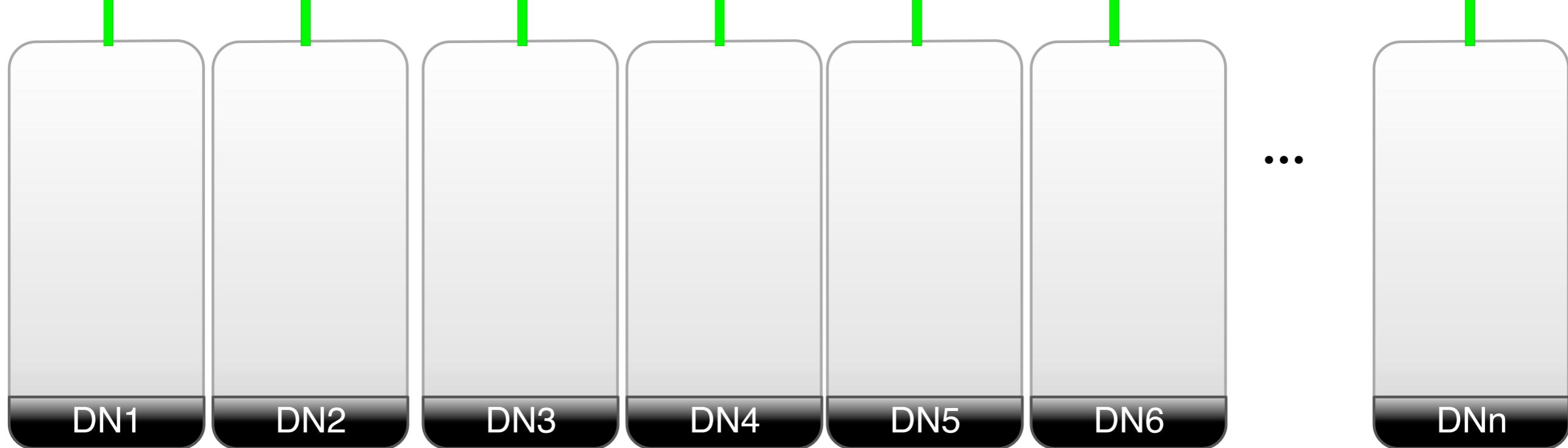


MapReduce

group by term

HDFS

network



Shuffle Phase

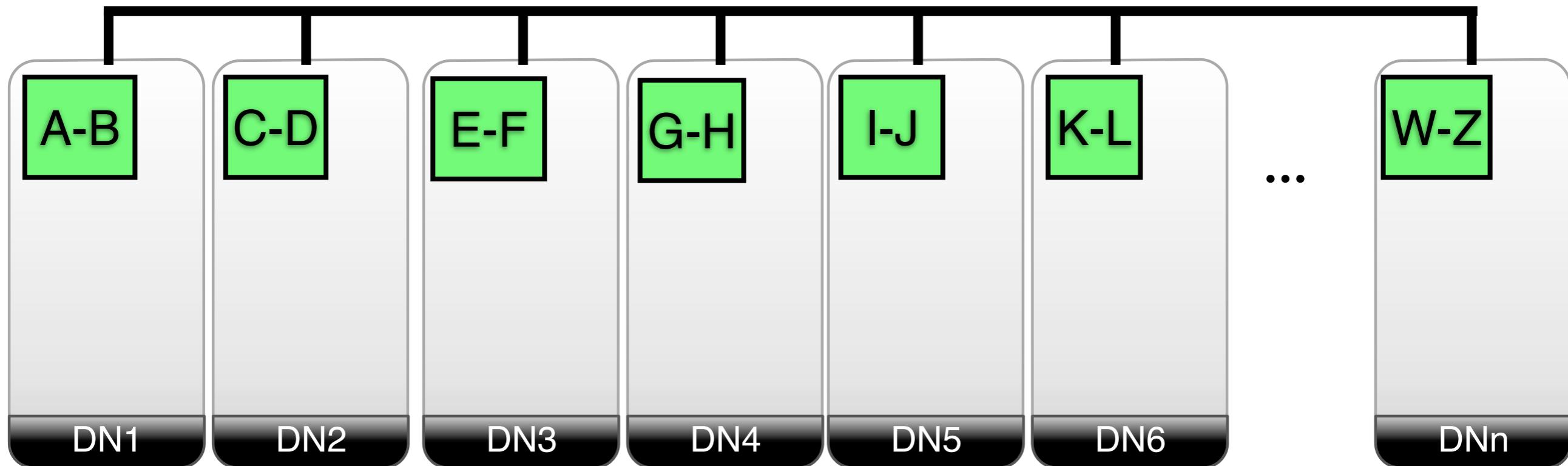


MapReduce

group by term

HDFS

network



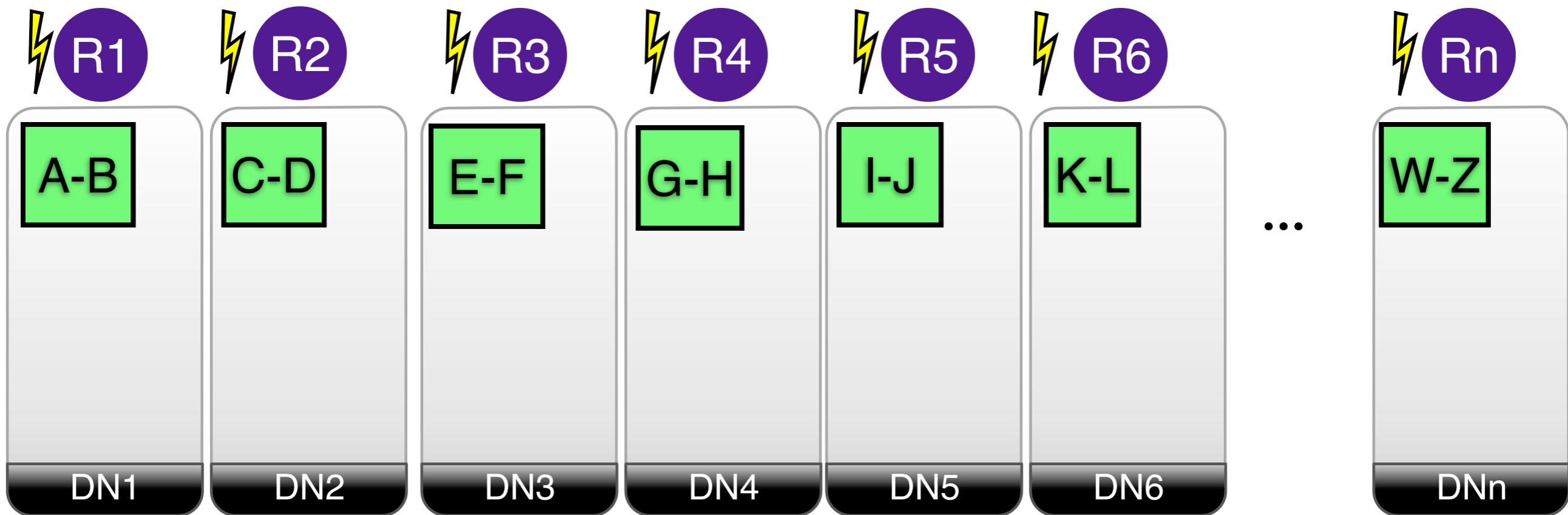
Reduce Phase



reduce(term, set of docID) -> set of (term, (posting list of docID, count))

MapReduce

HDFS



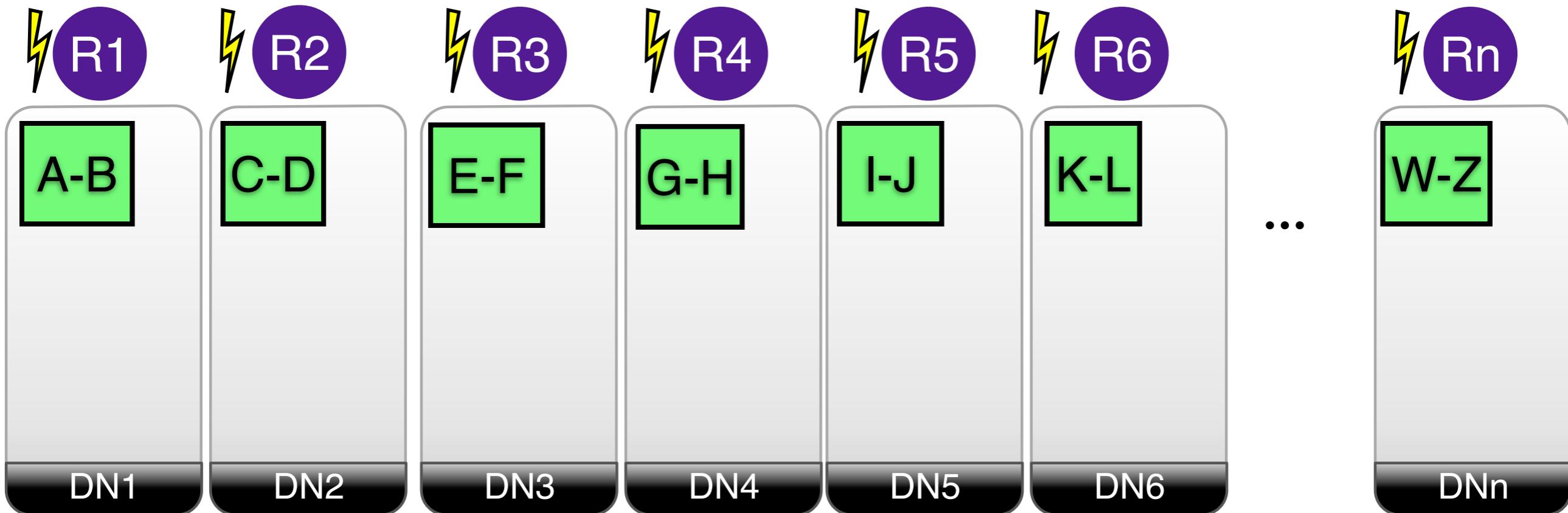
Reduce Phase



MapReduce

reduce(term, set of docID) -> set of
(term, (posting list of docID, count))

HDFS



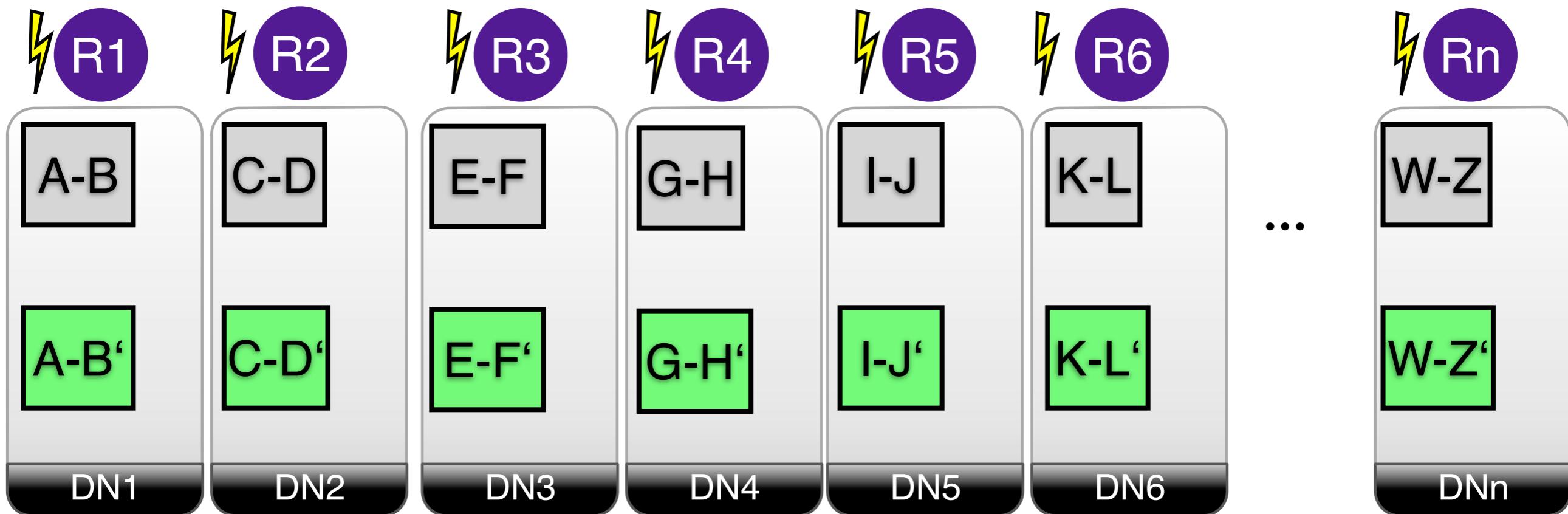
Reduce Phase



MapReduce

reduce(term, set of docID) -> set of
(term, (posting list of docID, count))

HDFS



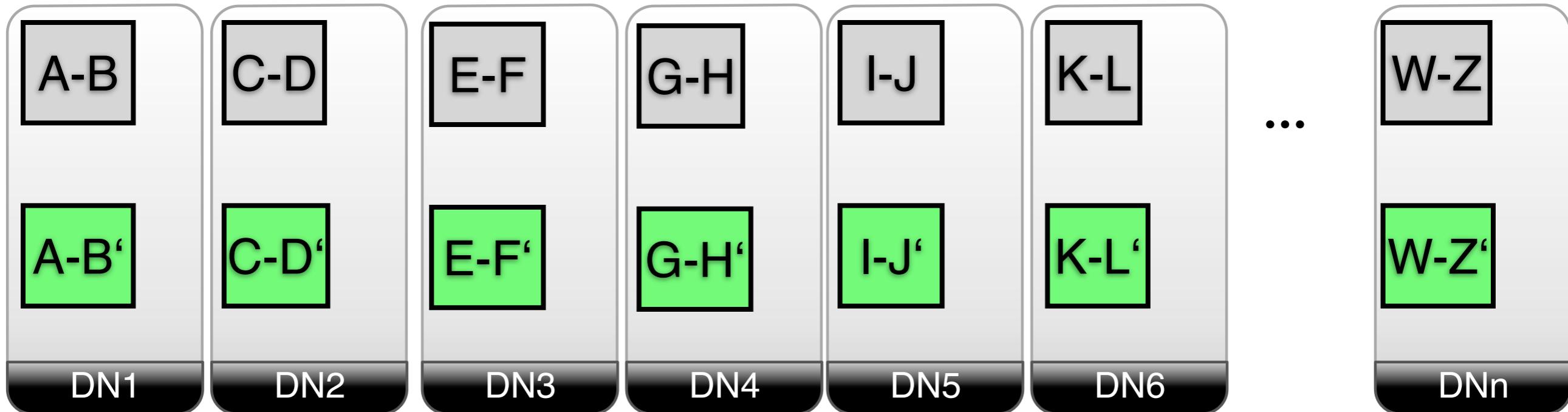
Reduce Phase



MapReduce

reduce(term, set of docID) -> set of
(term, (posting list of docID, count))

HDFS



Hadoop MapReduce Advantages

Failover

Failover

Scalability

Failover

Scalability

schema-later

Hadoop MapReduce **Disadvantages**

Performance

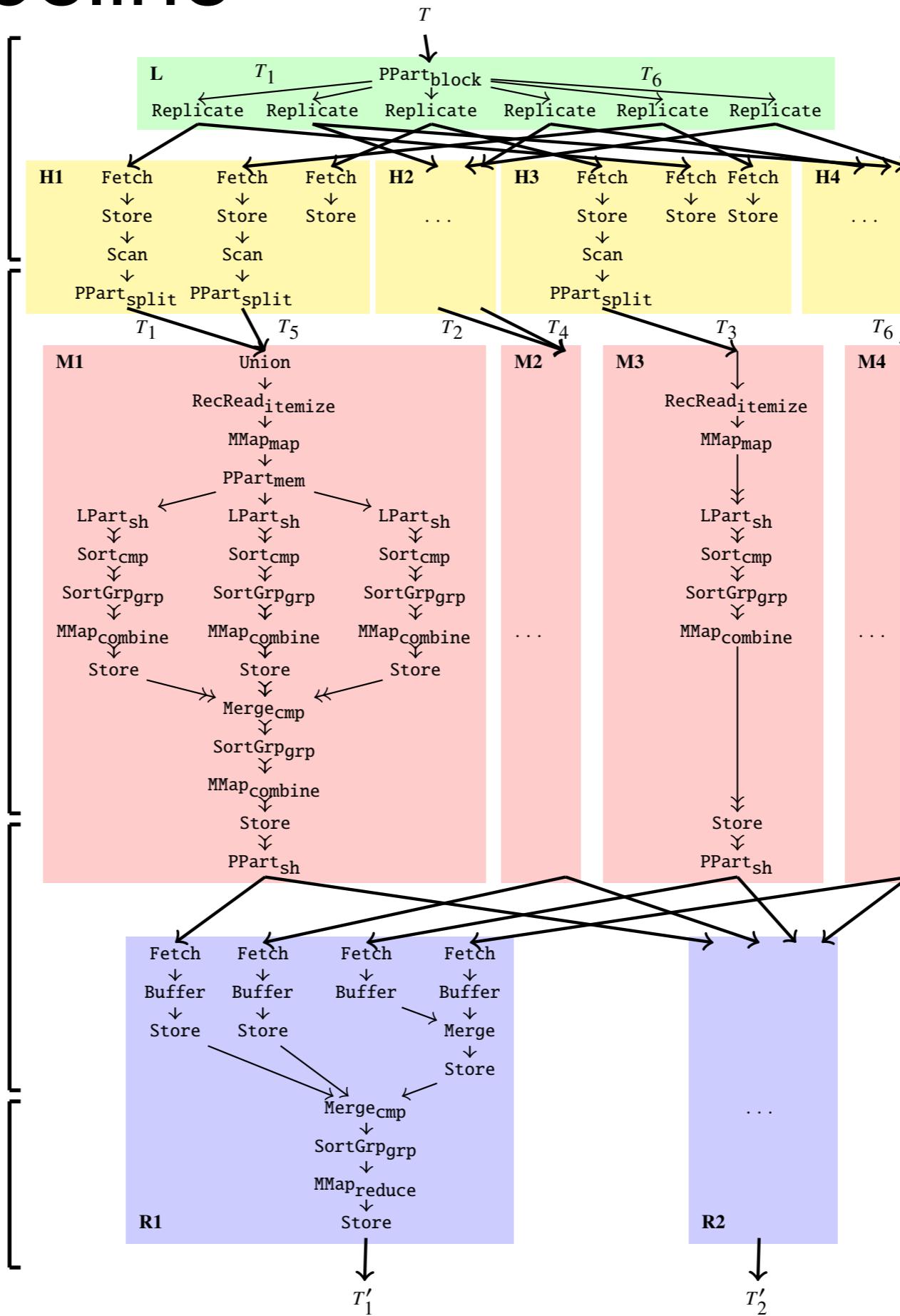
Execution Pipeline

Data
Load
Phase

Map
Phase

Shuffle
Phase

Reduce
Phase



MapReduce Intro

Data Layouts

Job Optimization

Indexing

Job Optimization

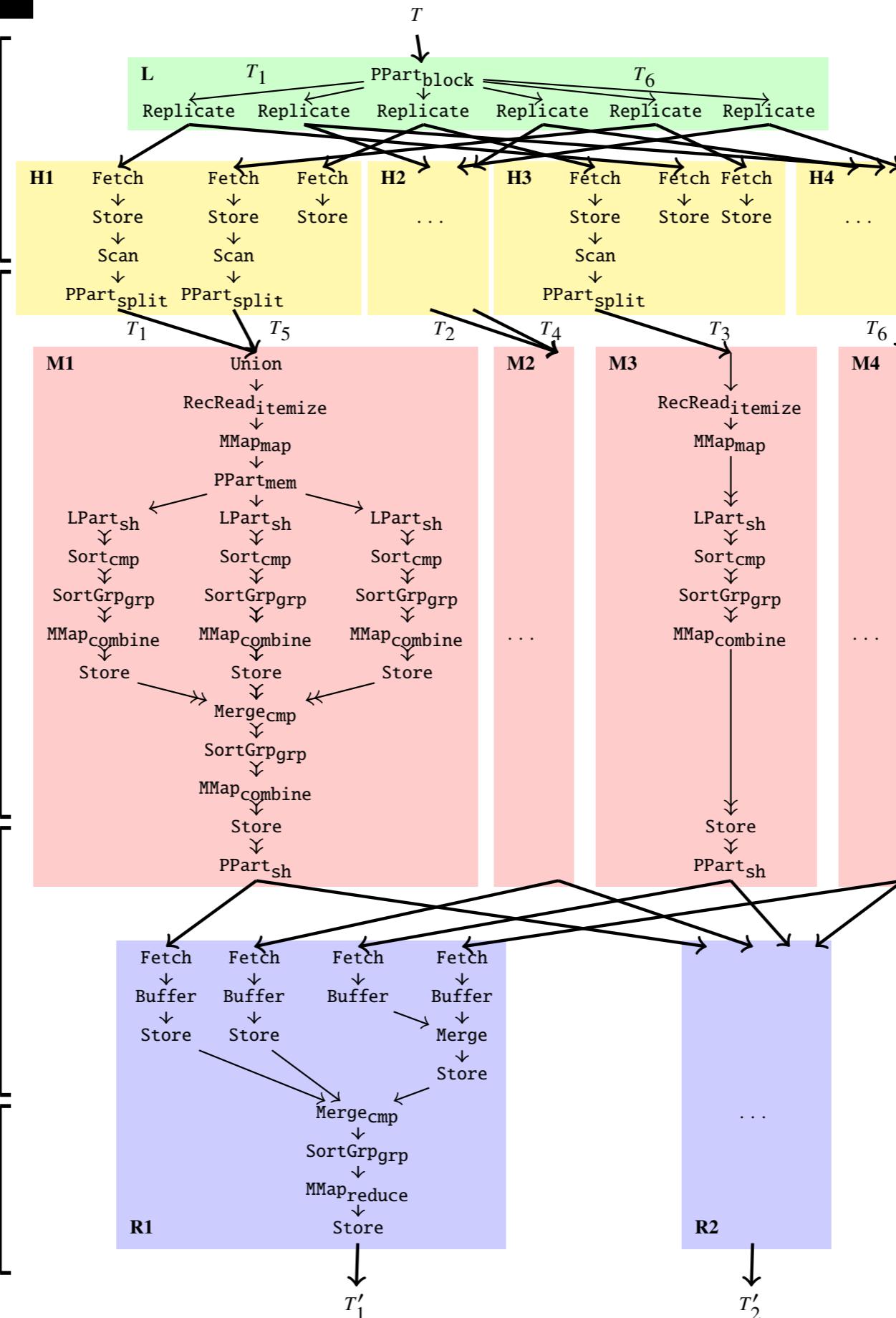
Spill Process

Data Load Phase

Map Phase

Shuffle Phase

Reduce Phase



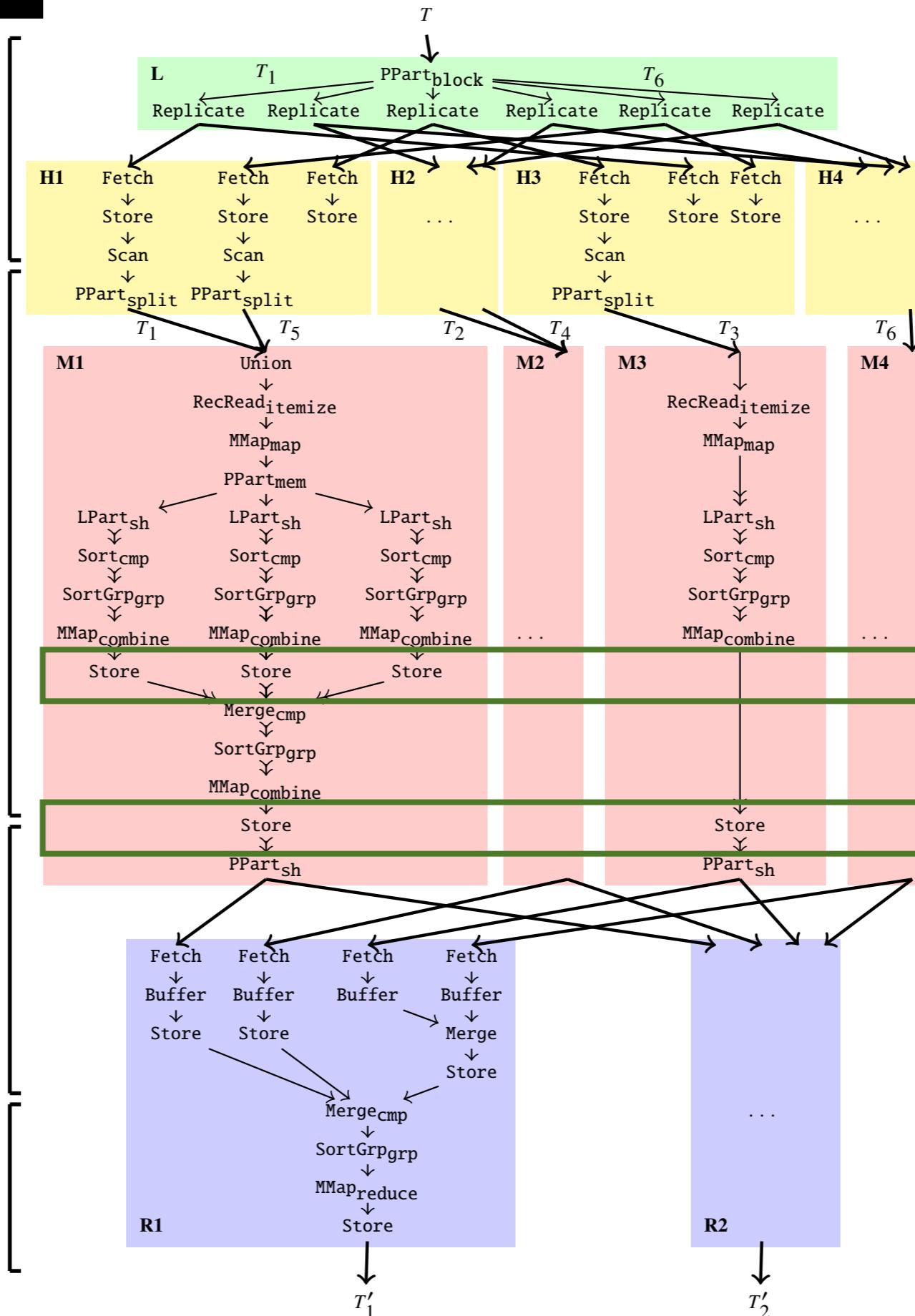
Spill Process

Data Load Phase

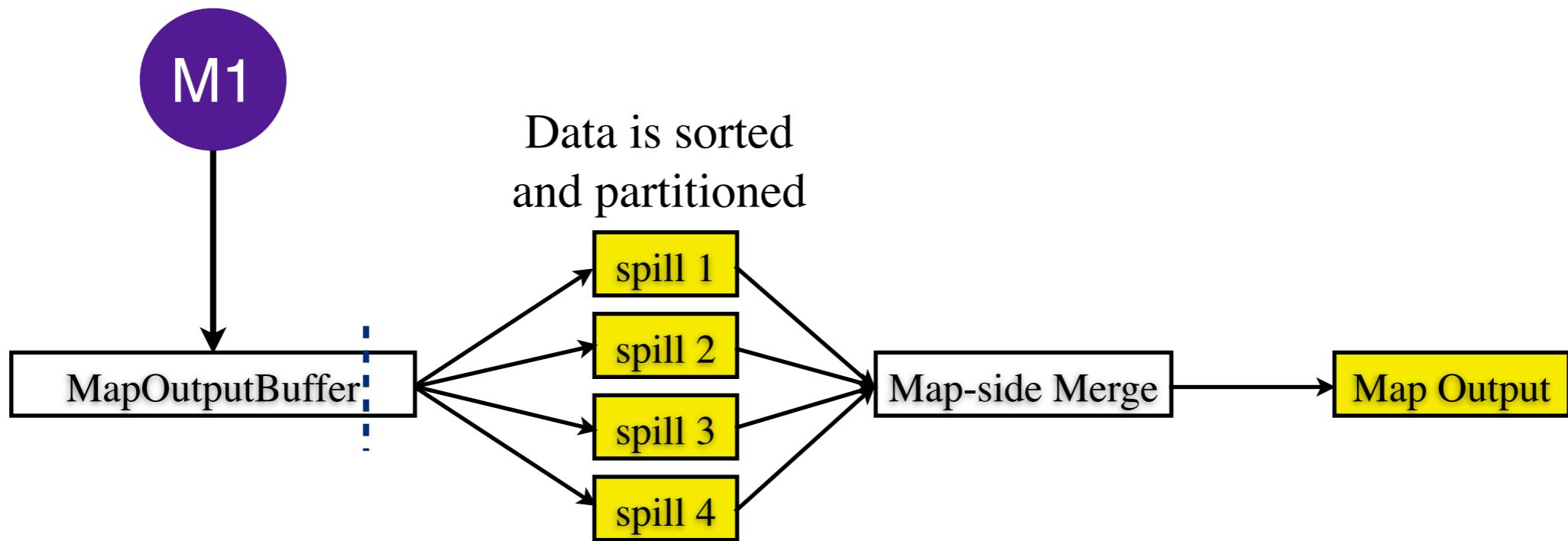
Map Phase

Shuffle Phase

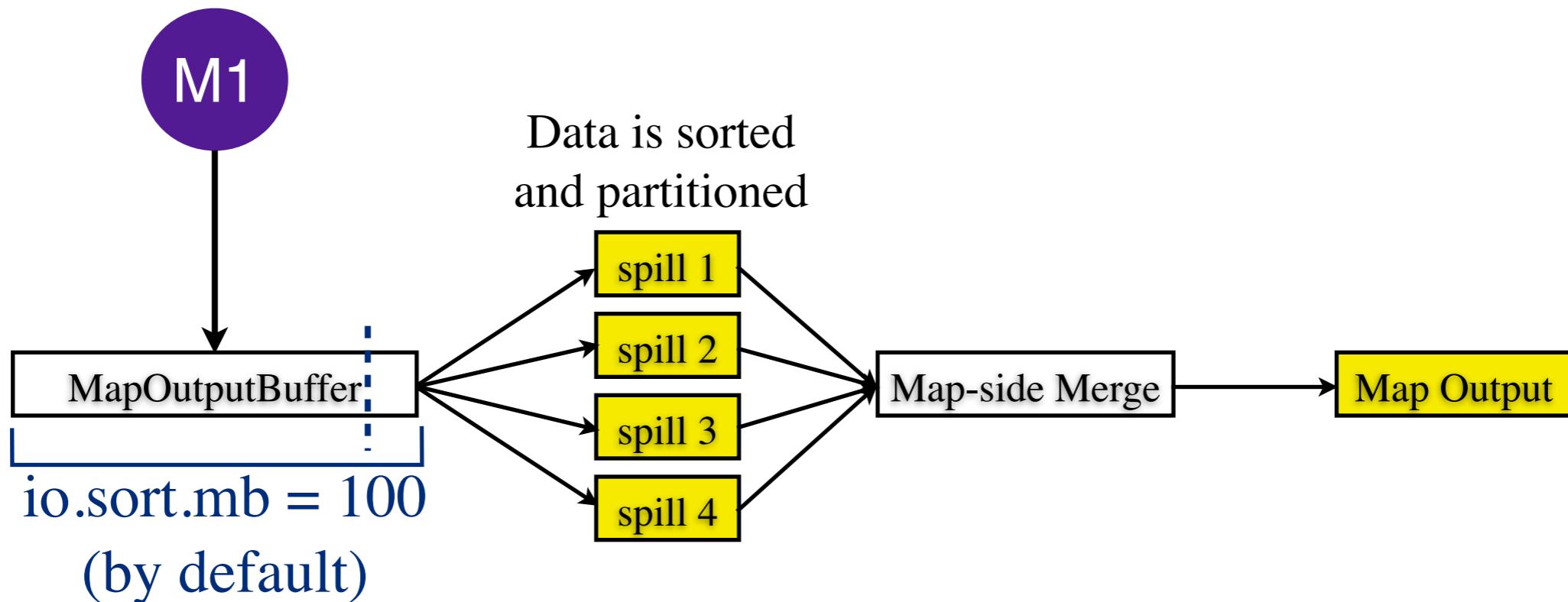
Reduce Phase



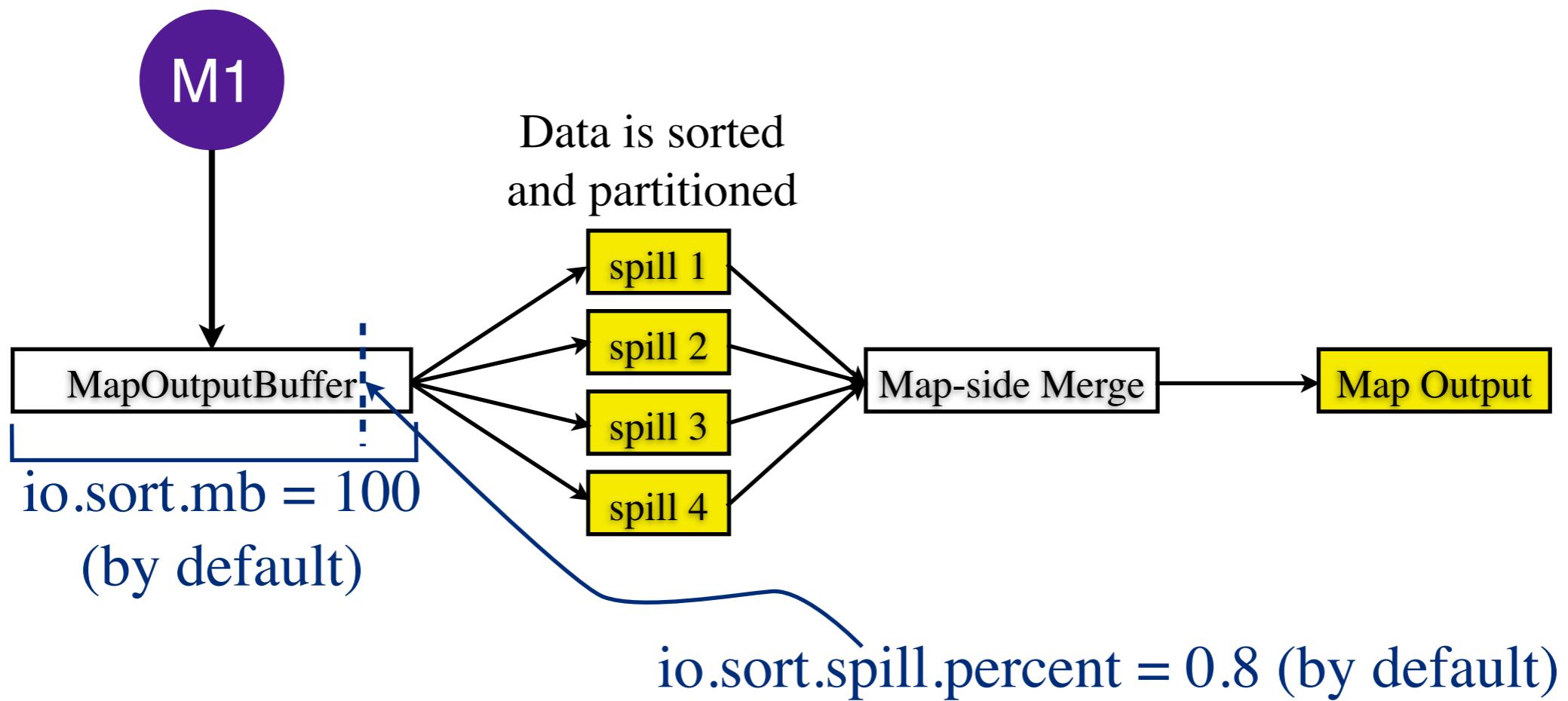
Spill Process Overview



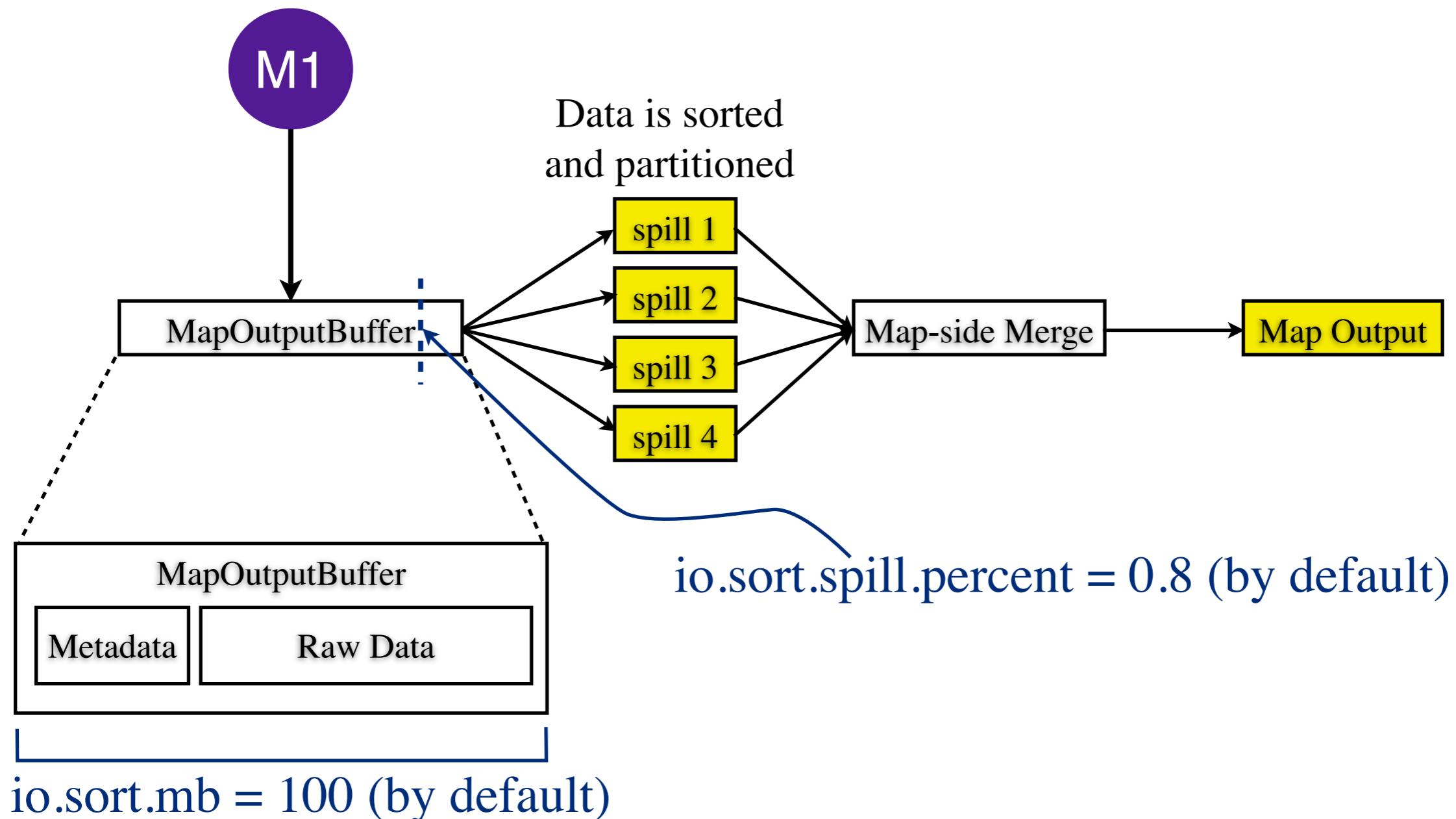
Spill Process Overview



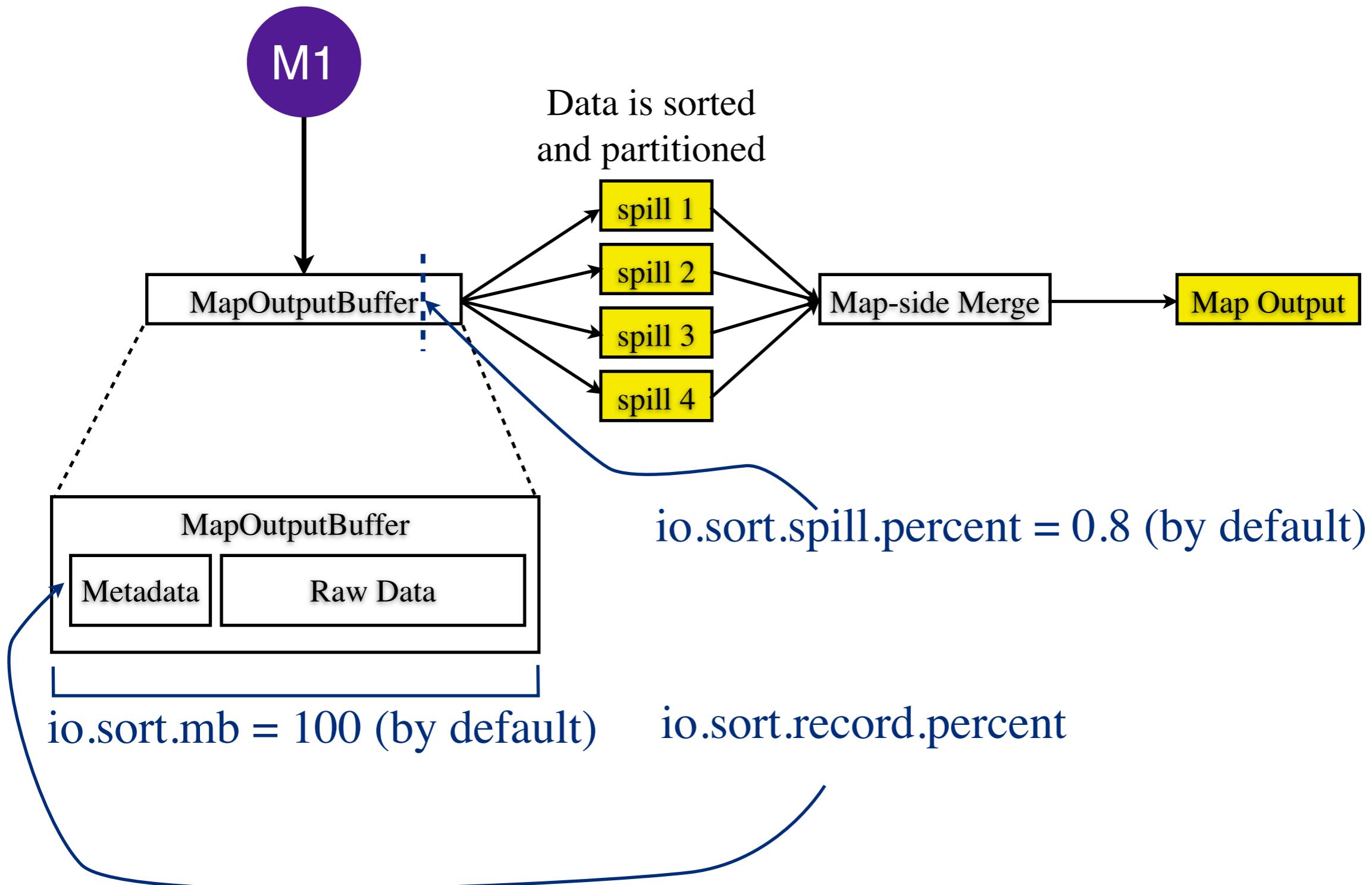
Spill Process Overview



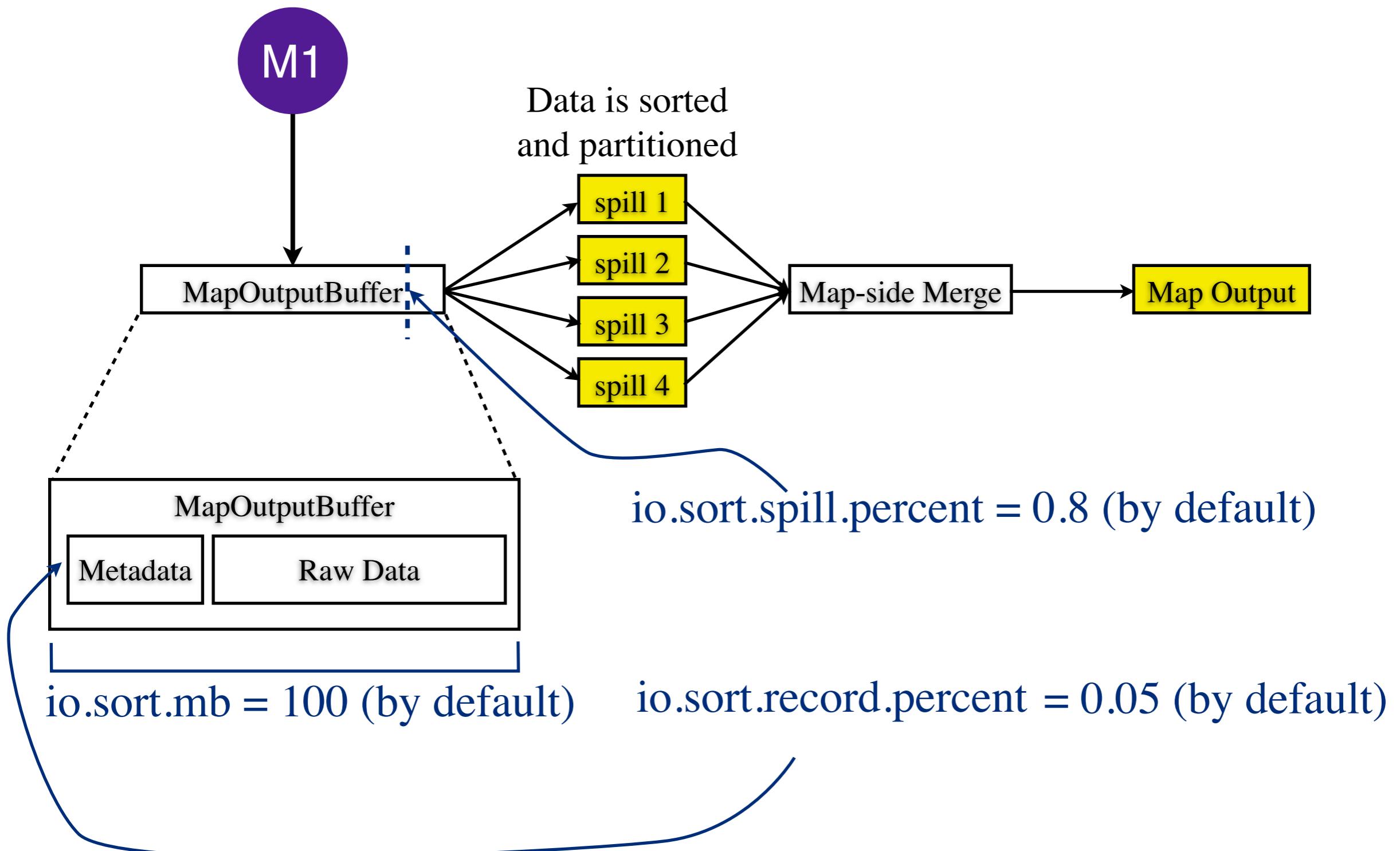
Spill Process Overview



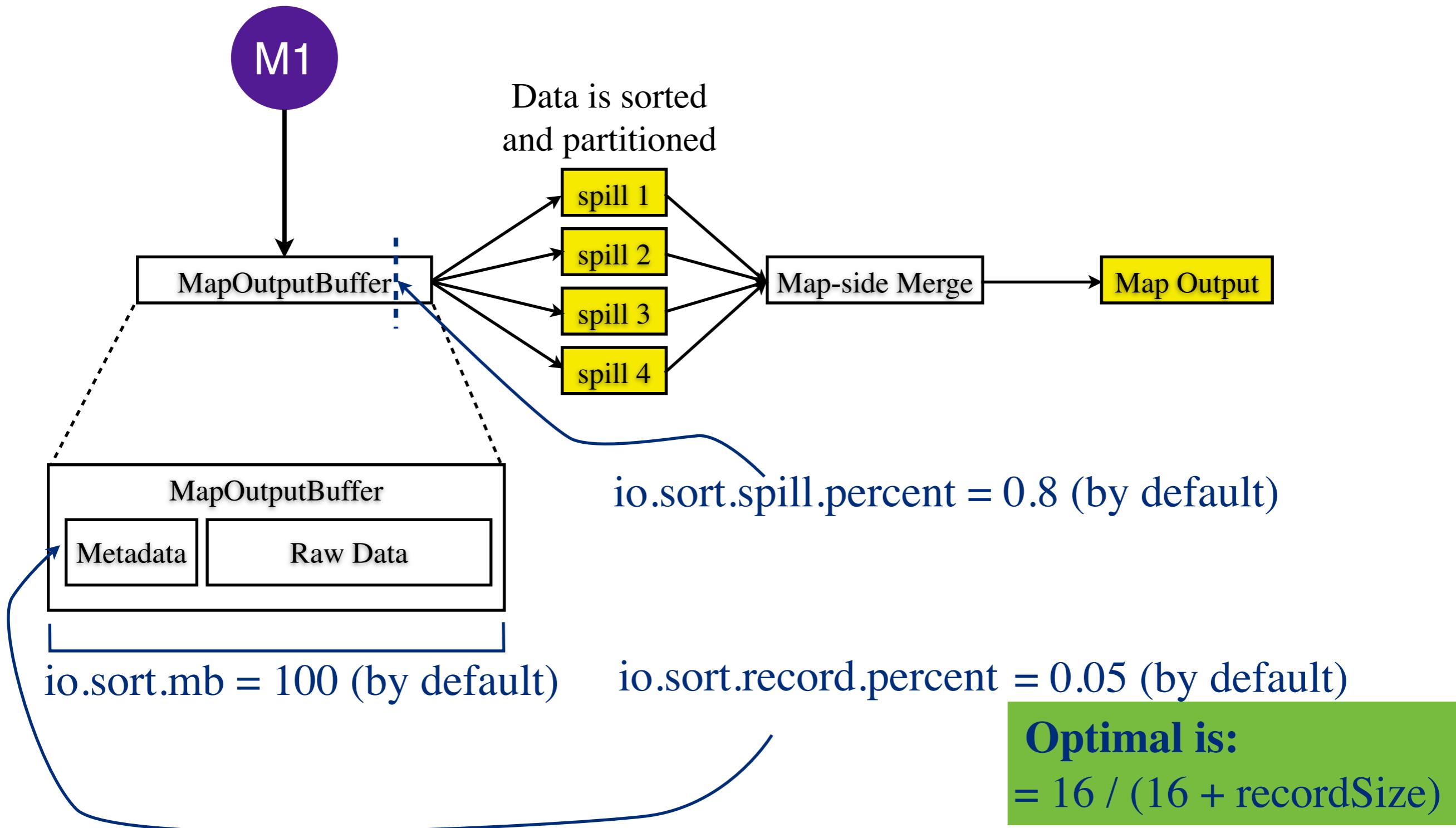
Spill Process Overview



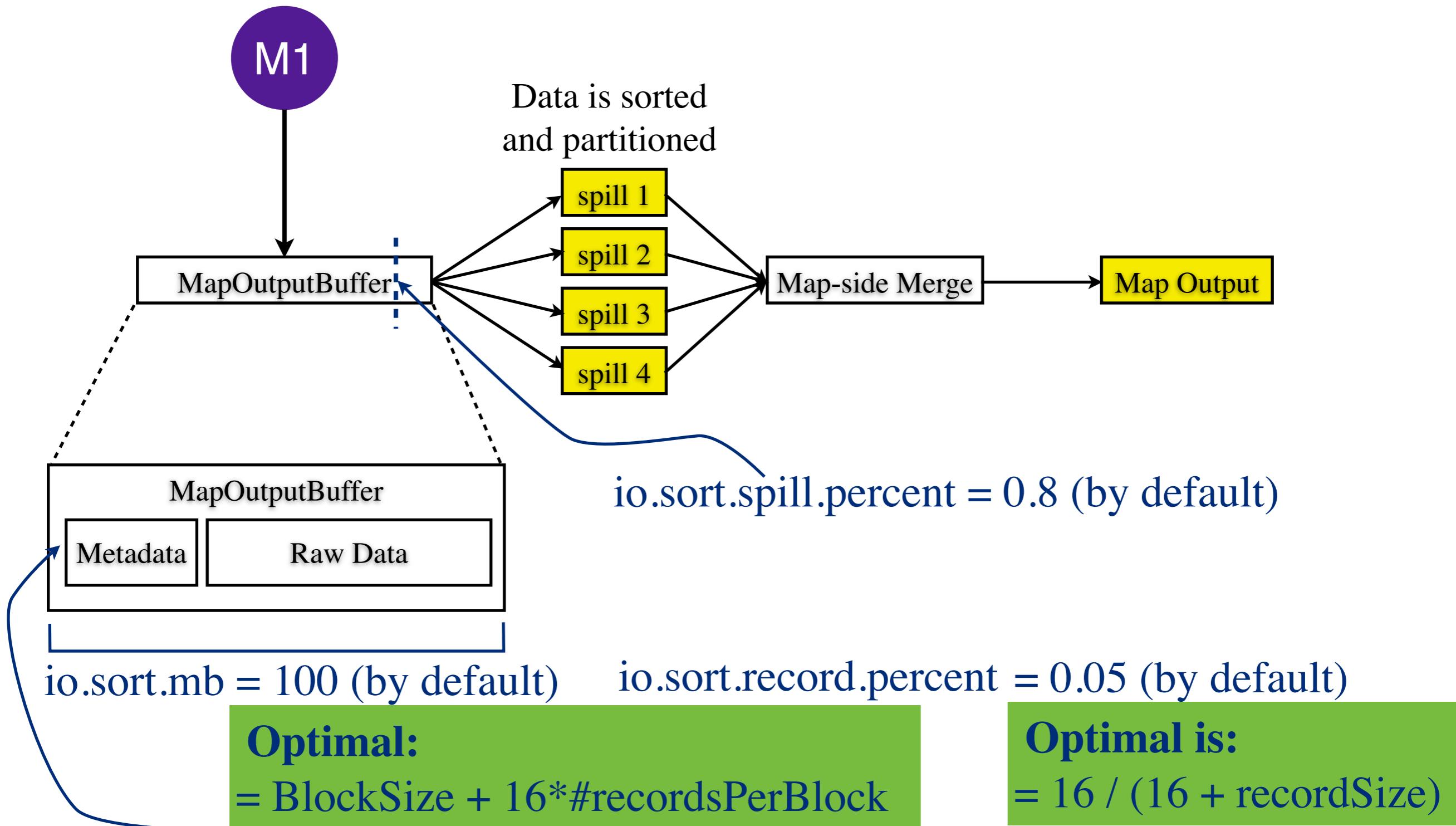
Spill Process Overview



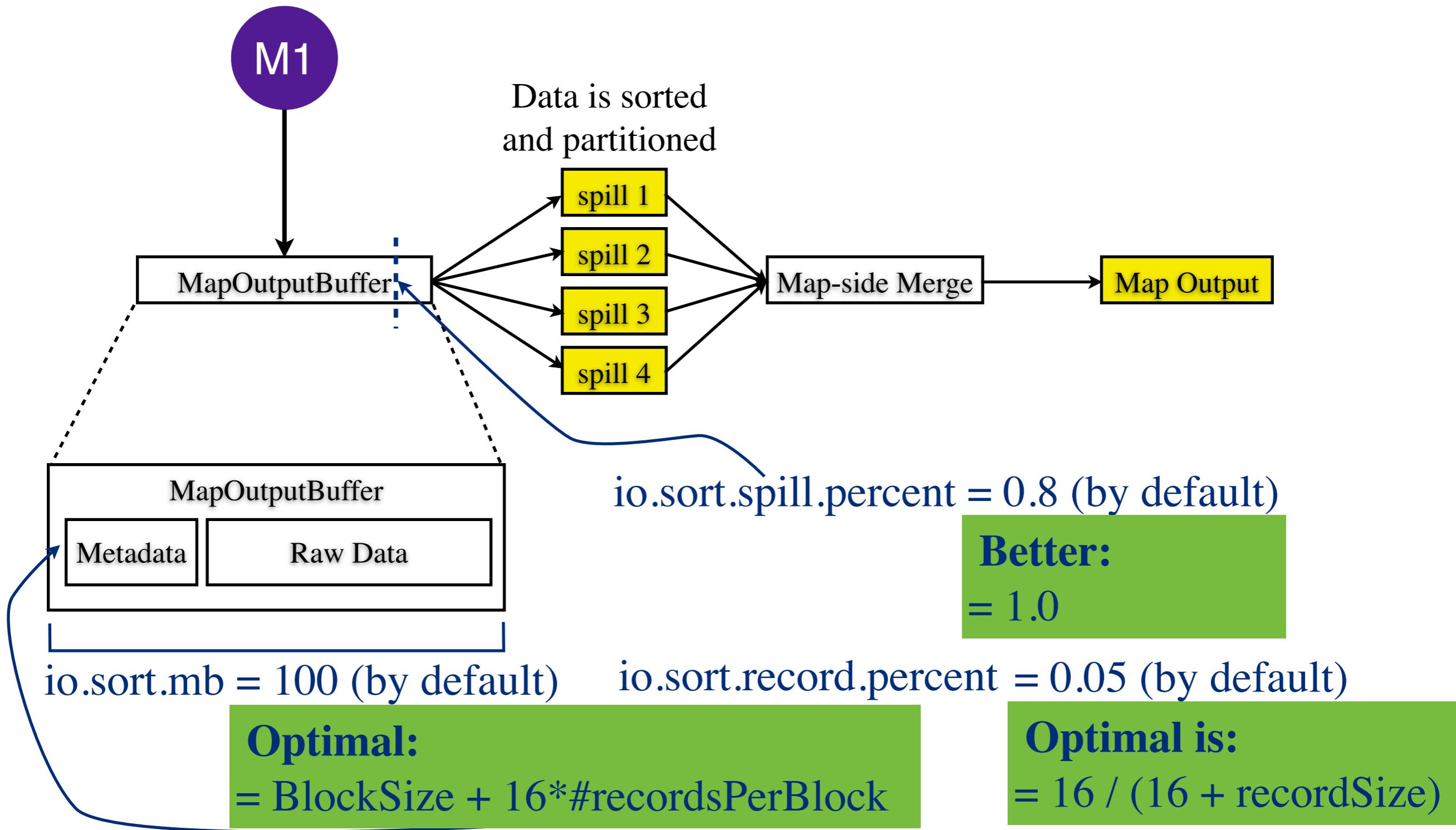
Spill Process Overview



Spill Process Overview



Spill Process Overview



But... there are
many more parameters!

name	value
hadoop.job.history.location	
hadoop.job.history.user.location	
io.sort.factor	10
io.sort.mb	100
io.sort.record.percent	0.05
io.sort.spill.percent	0.80
io.map.index.skip	0
mapred.job.tracker	local
mapred.job.tracker.http.address	0.0.0.0:50030
mapred.job.tracker.handler.count	10
mapred.task.tracker.report.address	127.0.0.1:0
mapred.local.dir	\${hadoop.tmp.dir}/mapred/local
mapred.system.dir	\${hadoop.tmp.dir}/mapred/system
mapred.temp.dir	\${hadoop.tmp.dir}/mapred/temp
mapred.local.dir.minspacestart	0
mapred.local.dir.minspacekill	0
mapred.tasktracker.expiry.interval	600000
mapred.tasktracker.instrumentation	org.apache.hadoop.mapred.TaskTrackerMetricsInst
mapred.tasktracker.memory_calculator_plugin	

name	value
hadoop.job.history.location	
hadoop.job.history.user.location	
io.sort.factor	10
io.sort.mb	100
io.sort.record.percent	0.05
io.sort.spill.percent	0.80
io.map.index.skip	0
mapred.job.tracker	local
mapred.job.tracker.http.address	0.0.0.0:50030
mapred.job.tracker.handler.count	10
mapred.task.tracker.report.address	127.0.0.1:0
mapred.local.dir	\${hadoop.tmp.dir}/mapred/local
mapred.system.dir	\${hadoop.tmp.dir}/mapred/system
mapred.temp.dir	\${hadoop.tmp.dir}/mapred/temp
mapred.local.dir.minspacestart	0
mapred.local.dir.minspacekill	0
mapred.tasktracker.expiry.interval	600000
mapred.tasktracker.instrumentation	org.apache.hadoop.mapred.TaskTrackerMetricsInst
mapred.tasktracker.memory_calculator_plugin	

Still
many more...



Tuning Job Parameters

Starfish

Overall Goal: find out the right parameter settings
for arbitrary MapReduce jobs.

Tuning Job Parameters

Starfish

Overall Goal: find out the right parameter settings
for arbitrary MapReduce jobs.

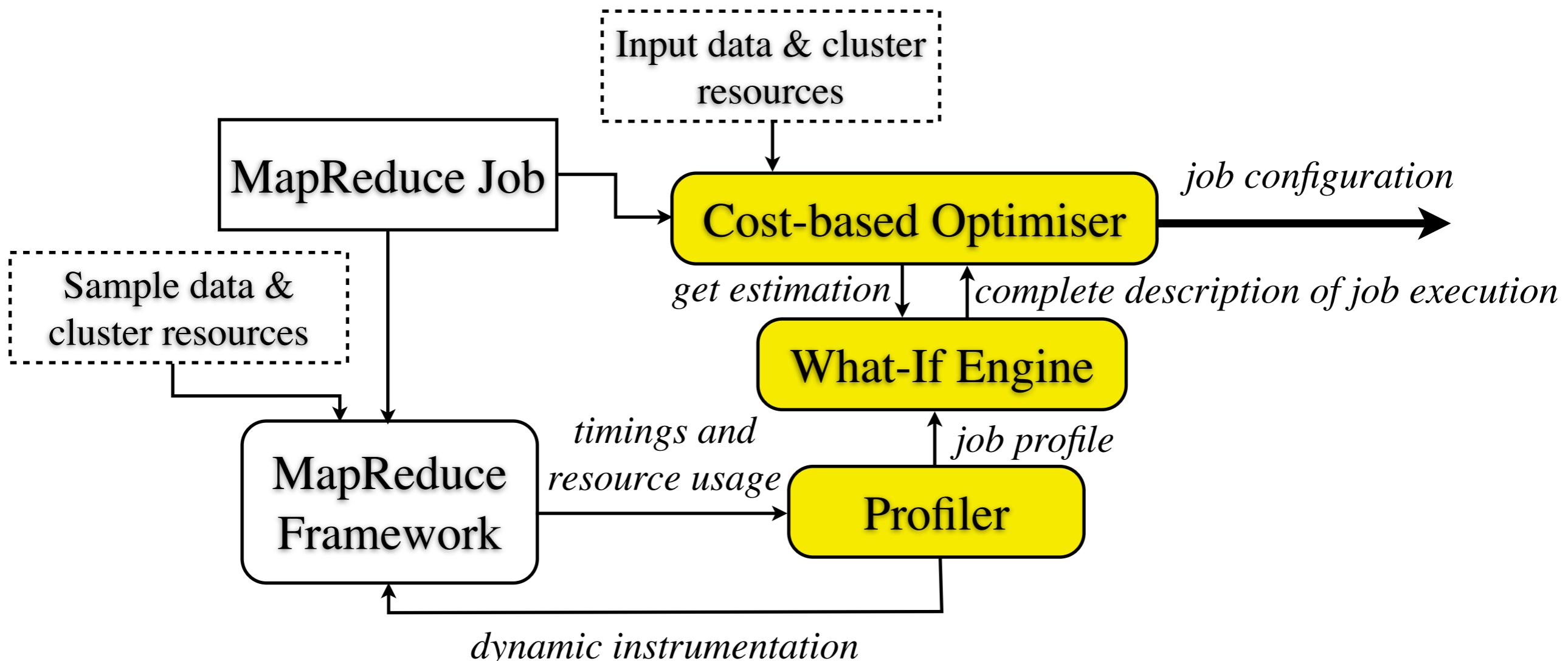
Contribution: Cost-based optimiser based on a
what-if engine.

Tuning Job Parameters

Starfish

Overall Goal: find out the right parameter settings for arbitrary MapReduce jobs.

Contribution: Cost-based optimiser based on a what-if engine.



Automatic Job Optimization

Manimal

Overall Goal: optimise MapReduce jobs by statically analysing their map functions.

Automatic Job Optimization

Manimal

Overall Goal: optimise MapReduce jobs by statically analysing their map functions.

Contribution: static code analysis of MapReduce jobs.

Automatic Job Optimization

Manimal

Overall Goal: optimise MapReduce jobs by statically analysing their map functions.

Contribution: static code analysis of MapReduce jobs.

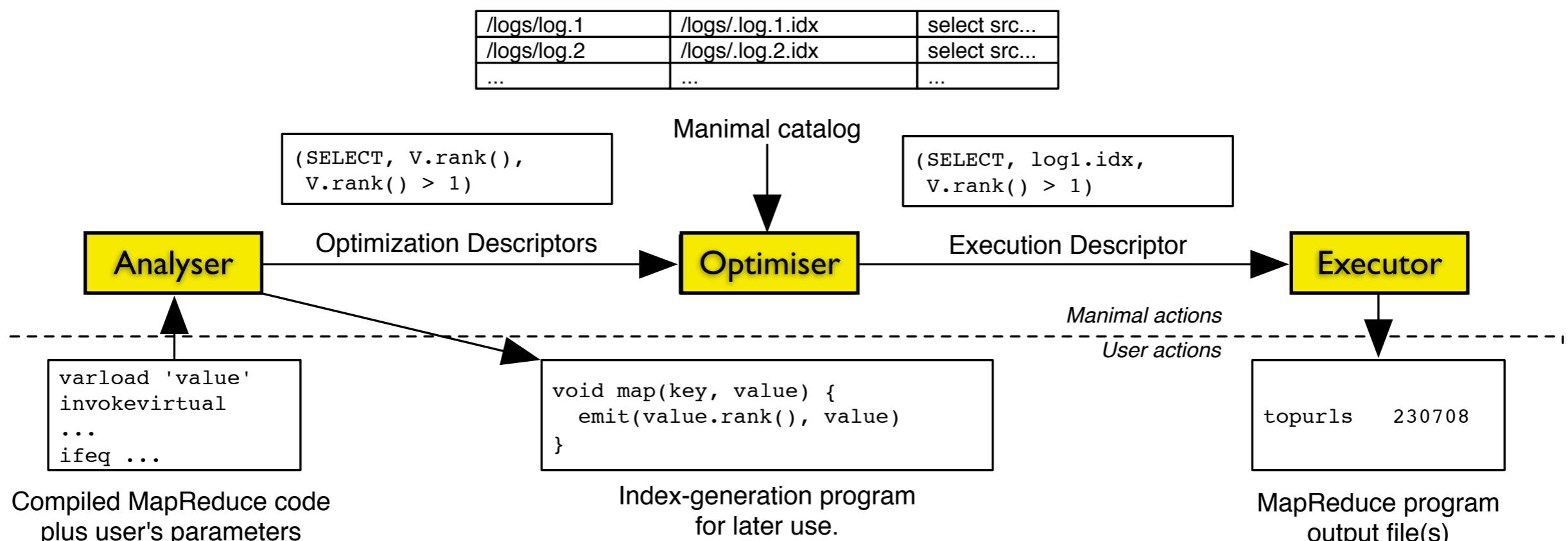
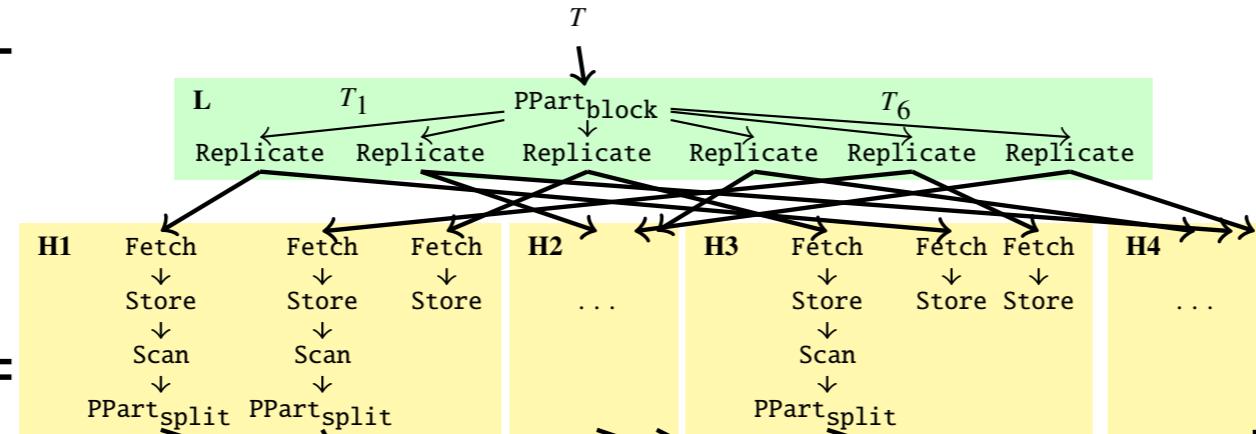
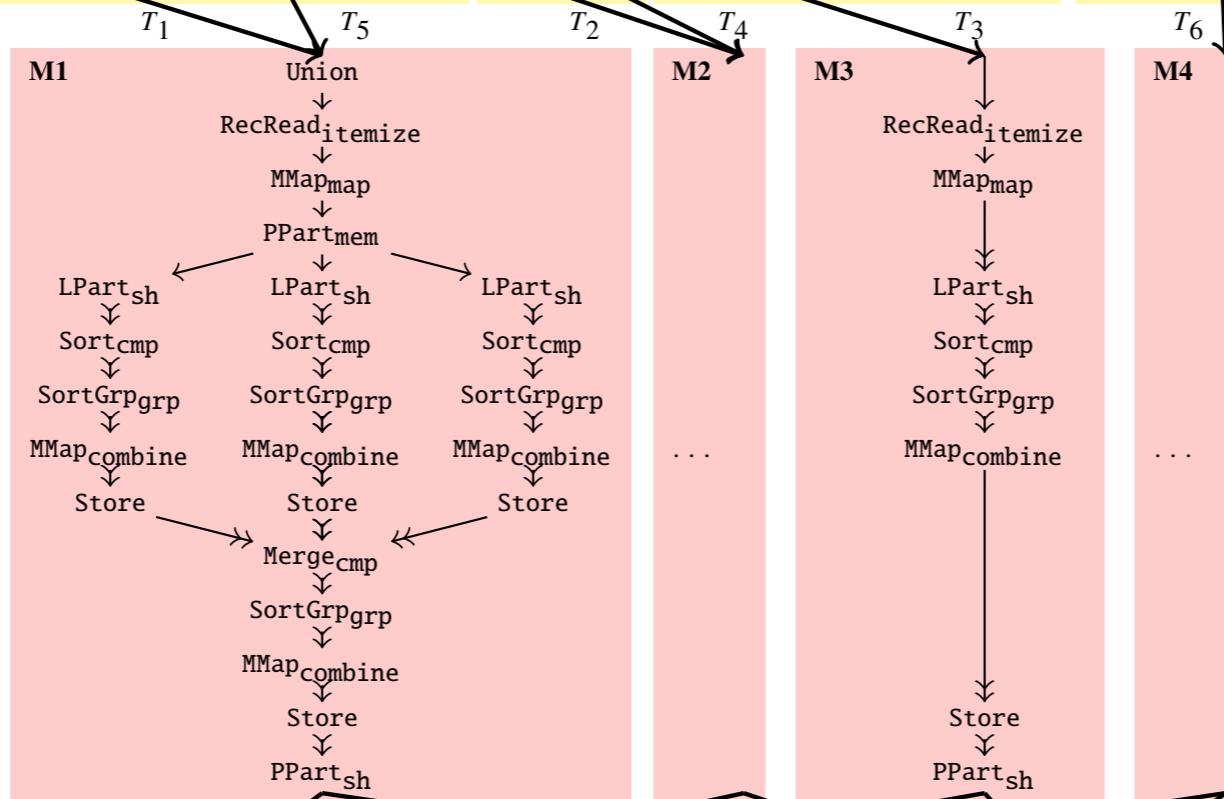


Figure 1: Architecture of the MANIMAL system.

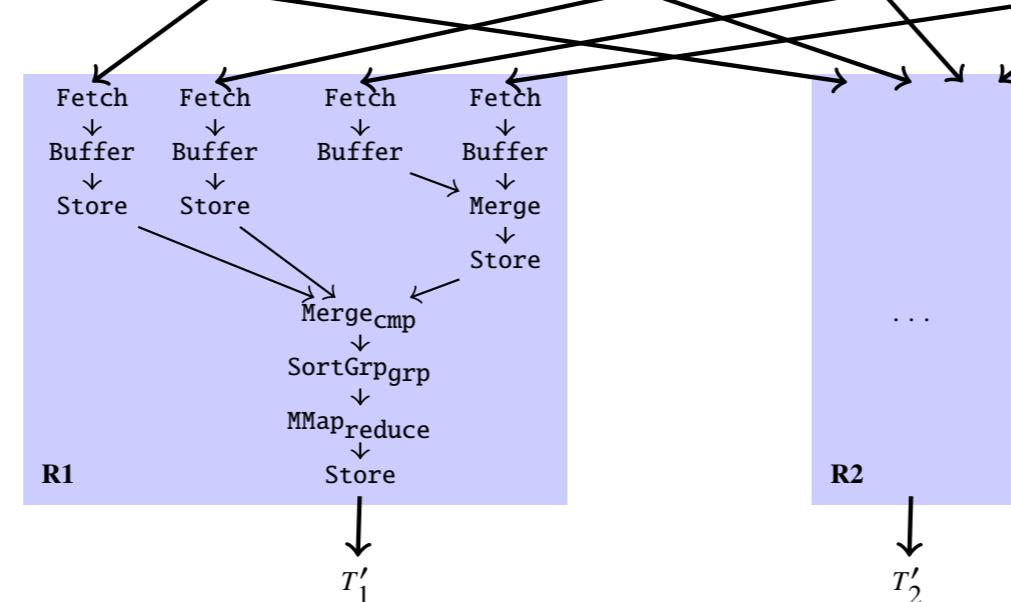
Data Load Phase



Map Phase



Shuffle Phase



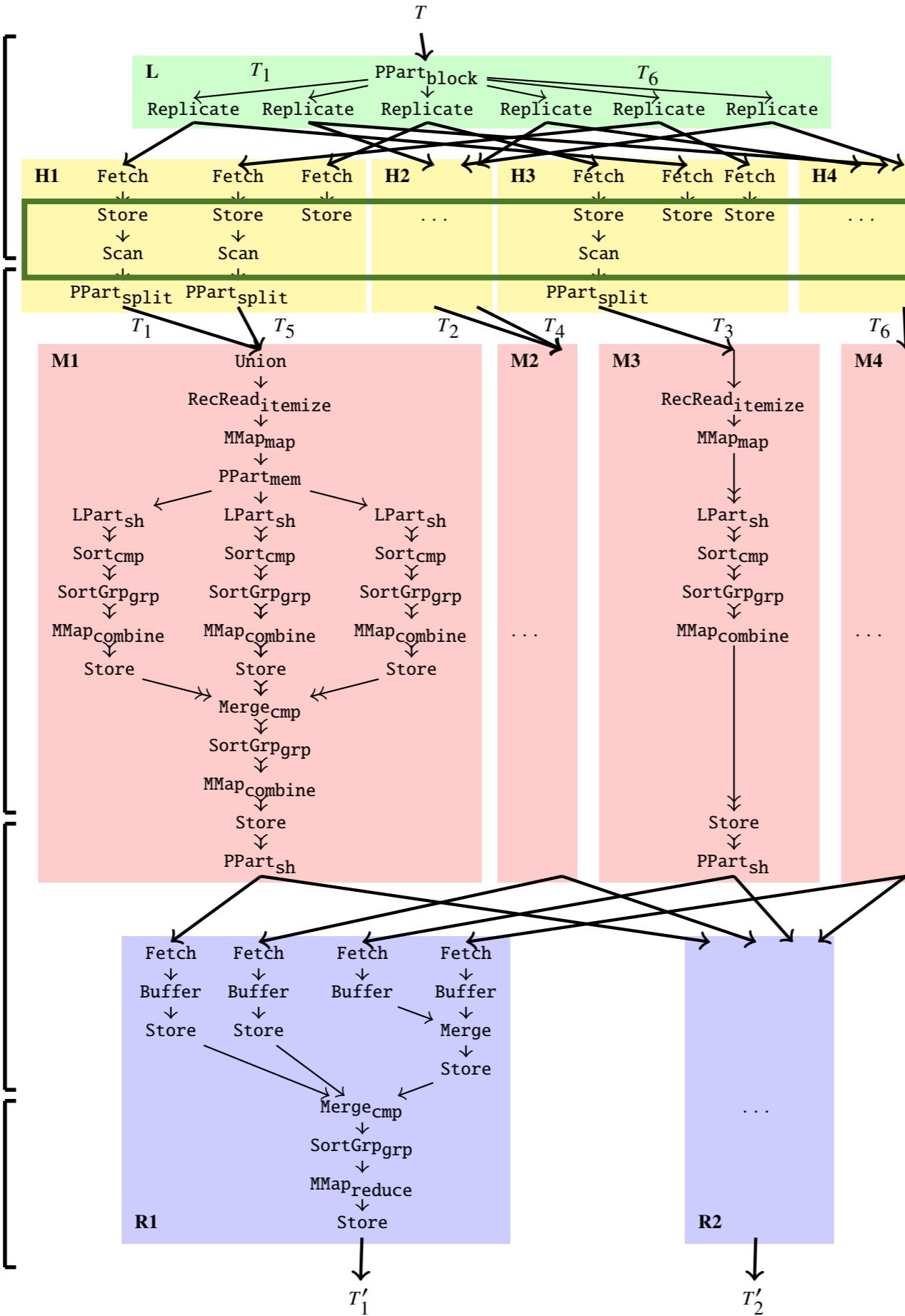
Reduce Phase

Data Load Phase

Map Phase

Shuffle Phase

Reduce Phase



Main focus of this tutorial!

MapReduce Intro

Data Layouts

Job Optimization

Indexing

Data Layouts

• Row-major vs column-major

Default Layout

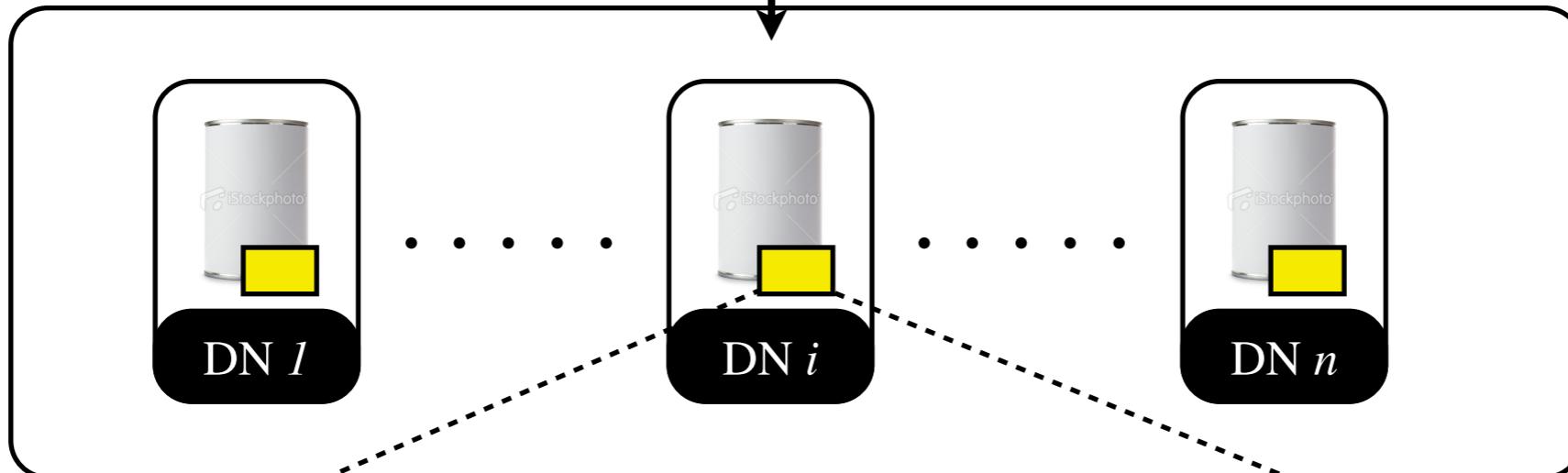
UserVisits Log

```
125.102.135.45, espn.com, 2011/12/01, 123.35, football  
101.132.121.13, cnn.com, 2011/12/02, 365.98, crisis  
120.115.124.34, vldb.org, 2011/12/03, 296.02, database  
.
```

.

HDFS

upload

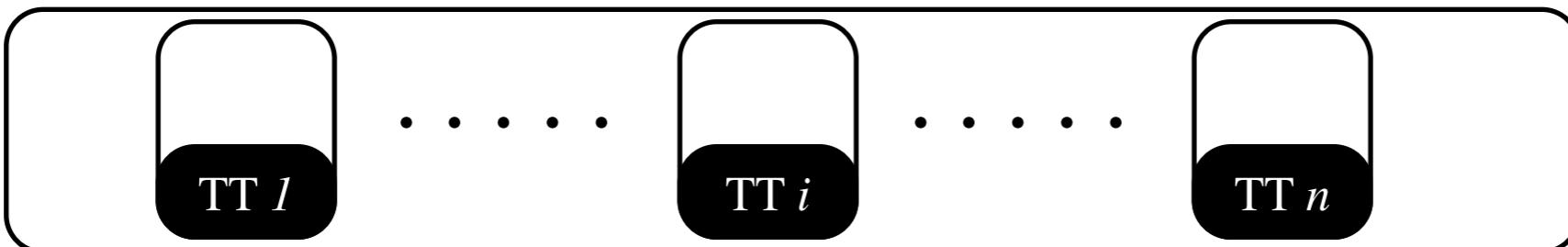


```
125.102.135.45, espn.com, 2011/12/01, 123.35, football  
101.132.121.13, cnn.com, 2011/12/02, 365.98, crisis  
120.115.124.34, vldb.org, 2011/12/03, 296.02, database  
.  
.
```

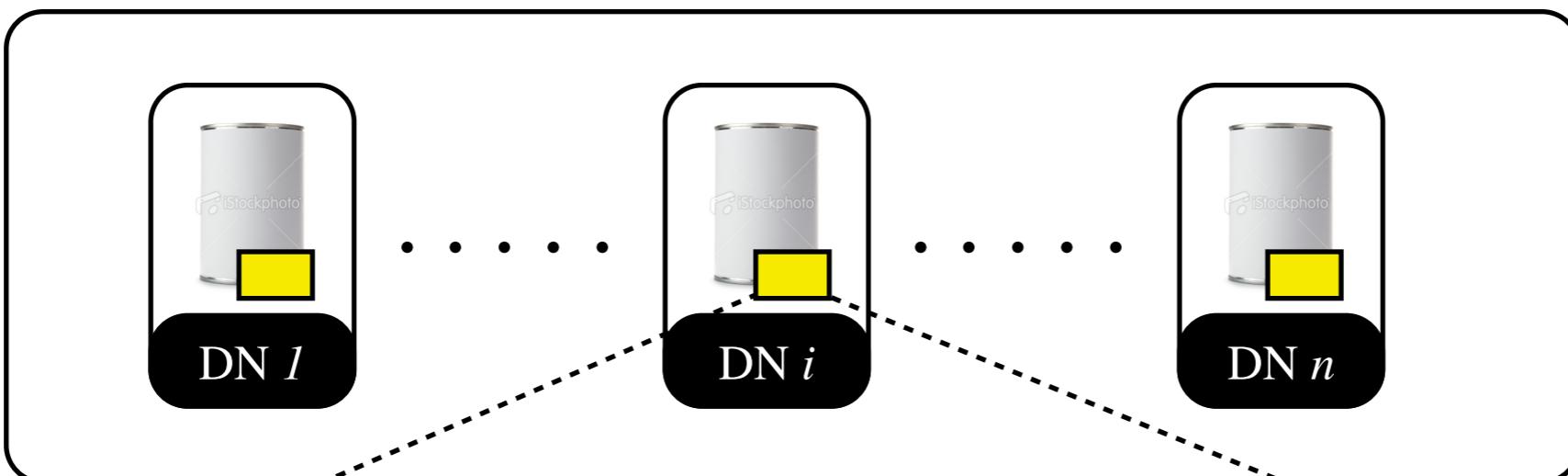
Problem



MapReduce



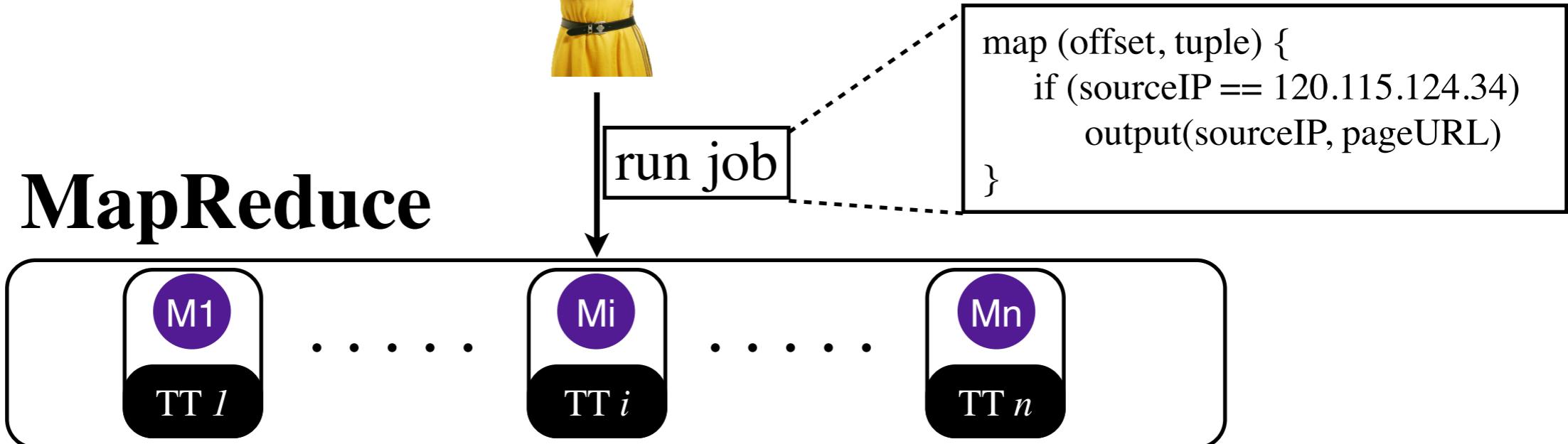
HDFS



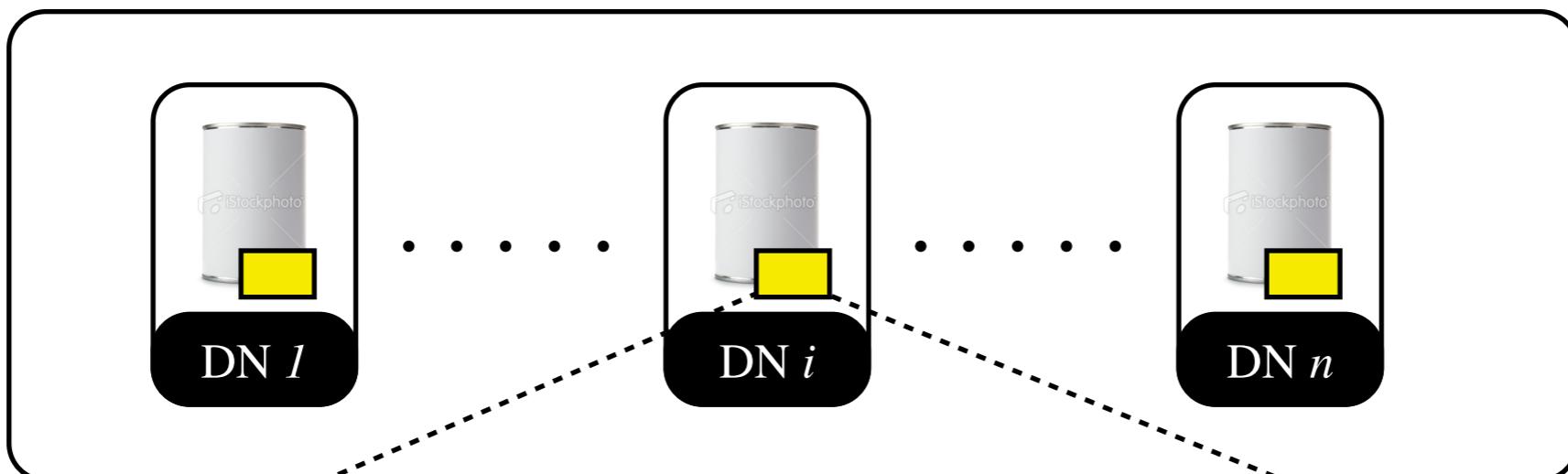
125.102.135.45, espn.com, 2011/12/01, 123.35, football
101.132.121.13, cnn.com, 2011/12/02, 365.98, crisis
120.115.124.34, vldb.org, 2011/12/03, 296.02, database
⋮

Problem

MapReduce



HDFS



```
125.102.135.45, espn.com, 2011/12/01, 123.35, football  
101.132.121.13, cnn.com, 2011/12/02, 365.98, crisis  
120.115.124.34, vldb.org, 2011/12/03, 296.02, database  
:  
:
```

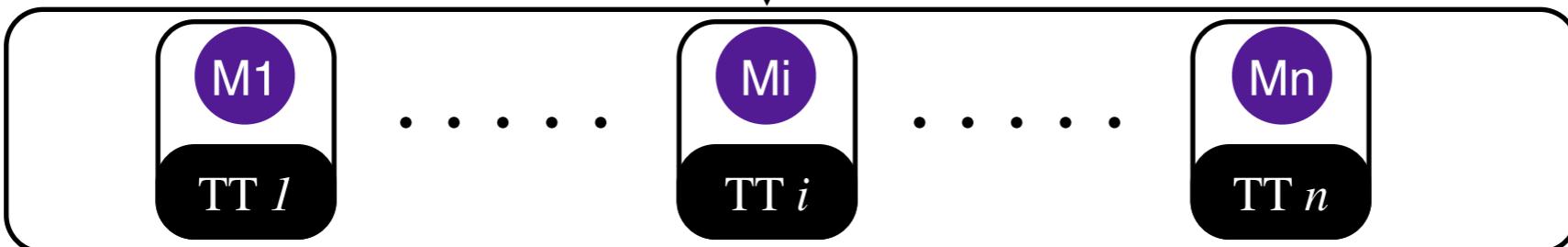
Problem

MapReduce

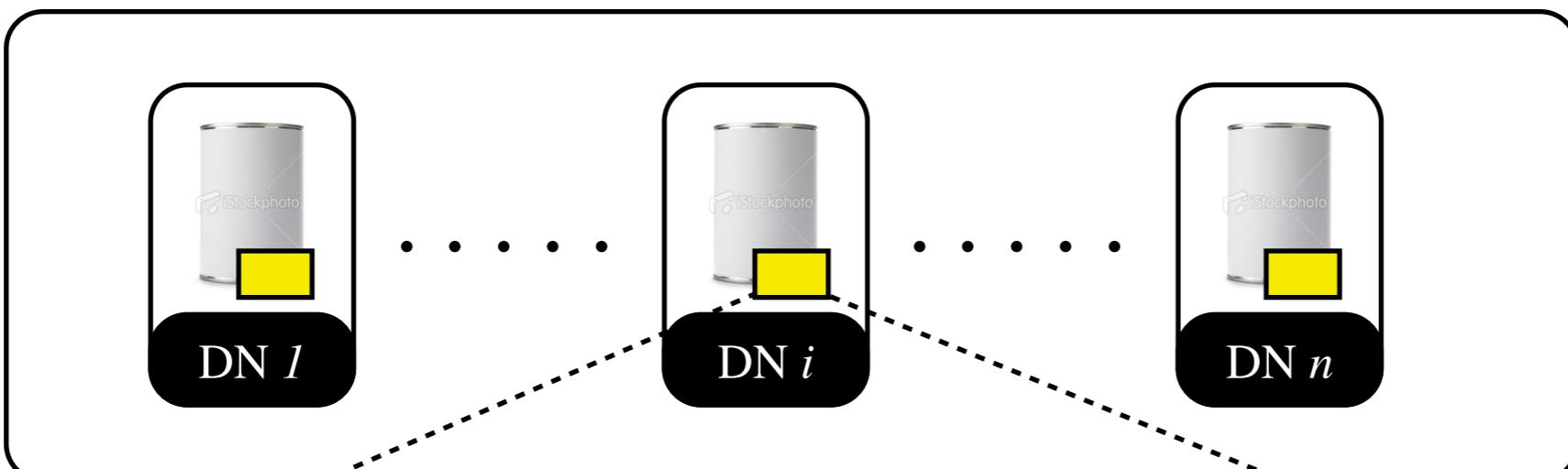


run job

```
map (offset, tuple) {  
    if (sourceIP == 120.115.124.34)  
        output sourceIP, pageURL  
}
```



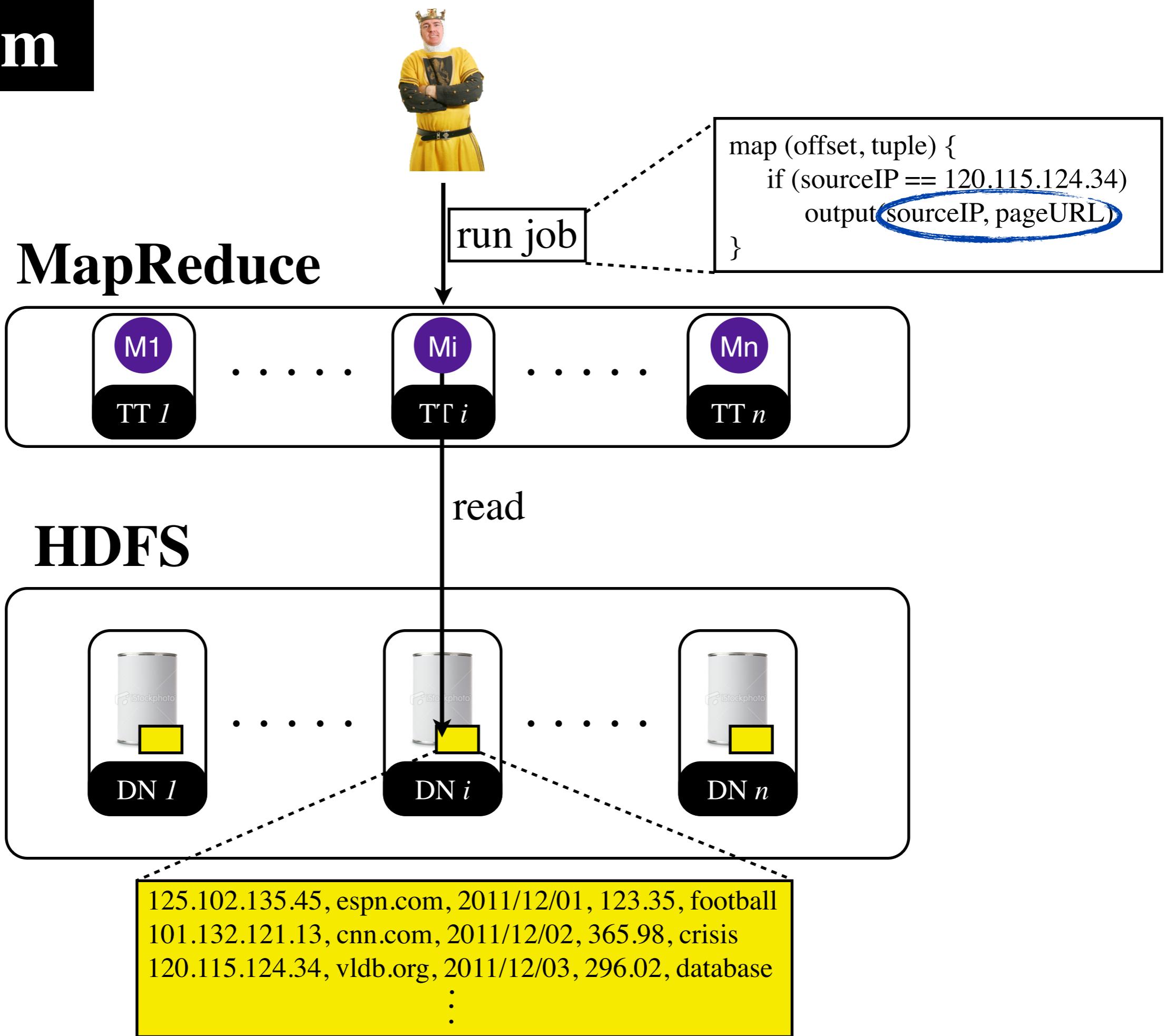
HDFS



```
125.102.135.45, espn.com, 2011/12/01, 123.35, football  
101.132.121.13, cnn.com, 2011/12/02, 365.98, crisis  
120.115.124.34, vldb.org, 2011/12/03, 296.02, database  
:  
:
```

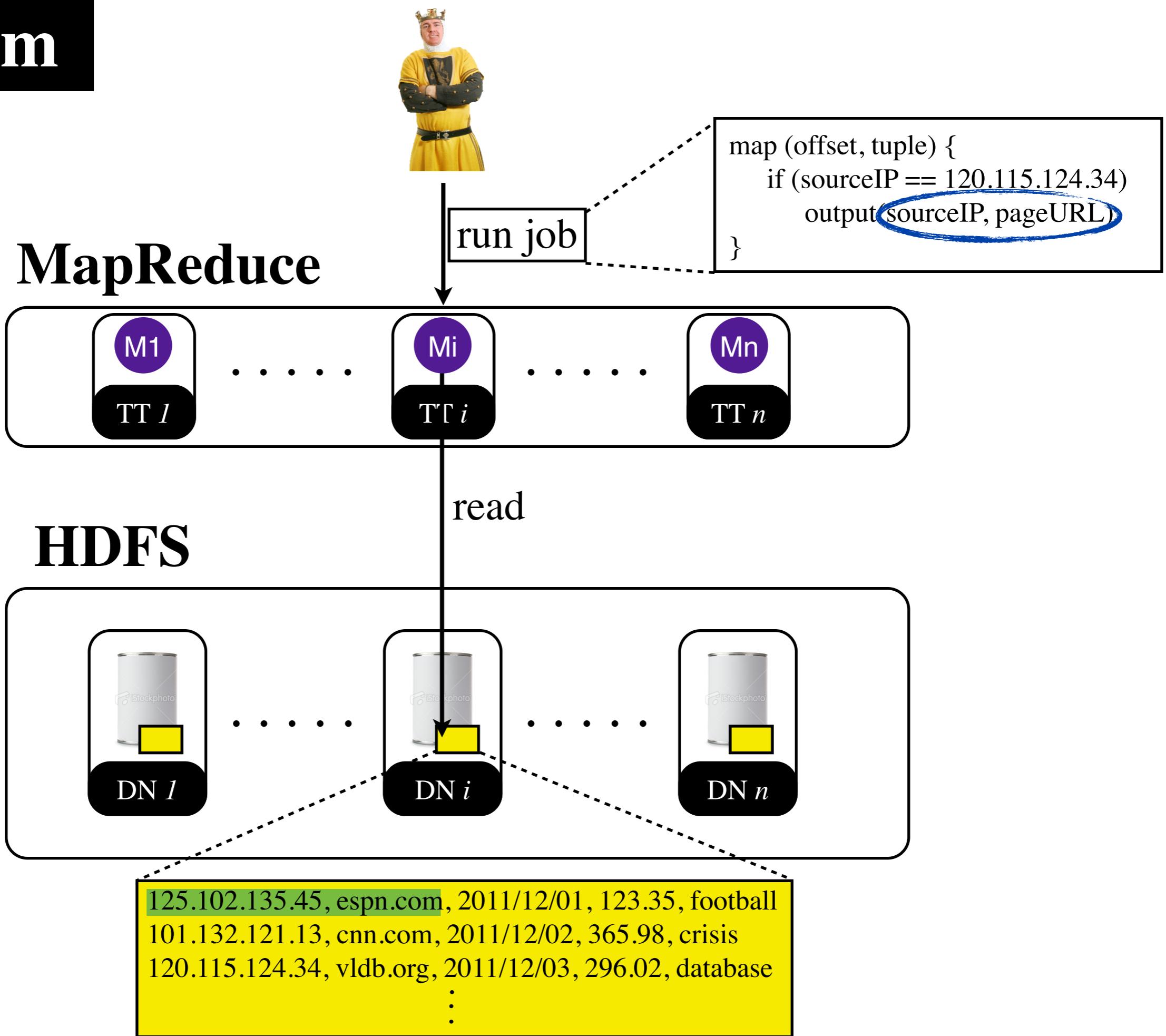
Problem

MapReduce



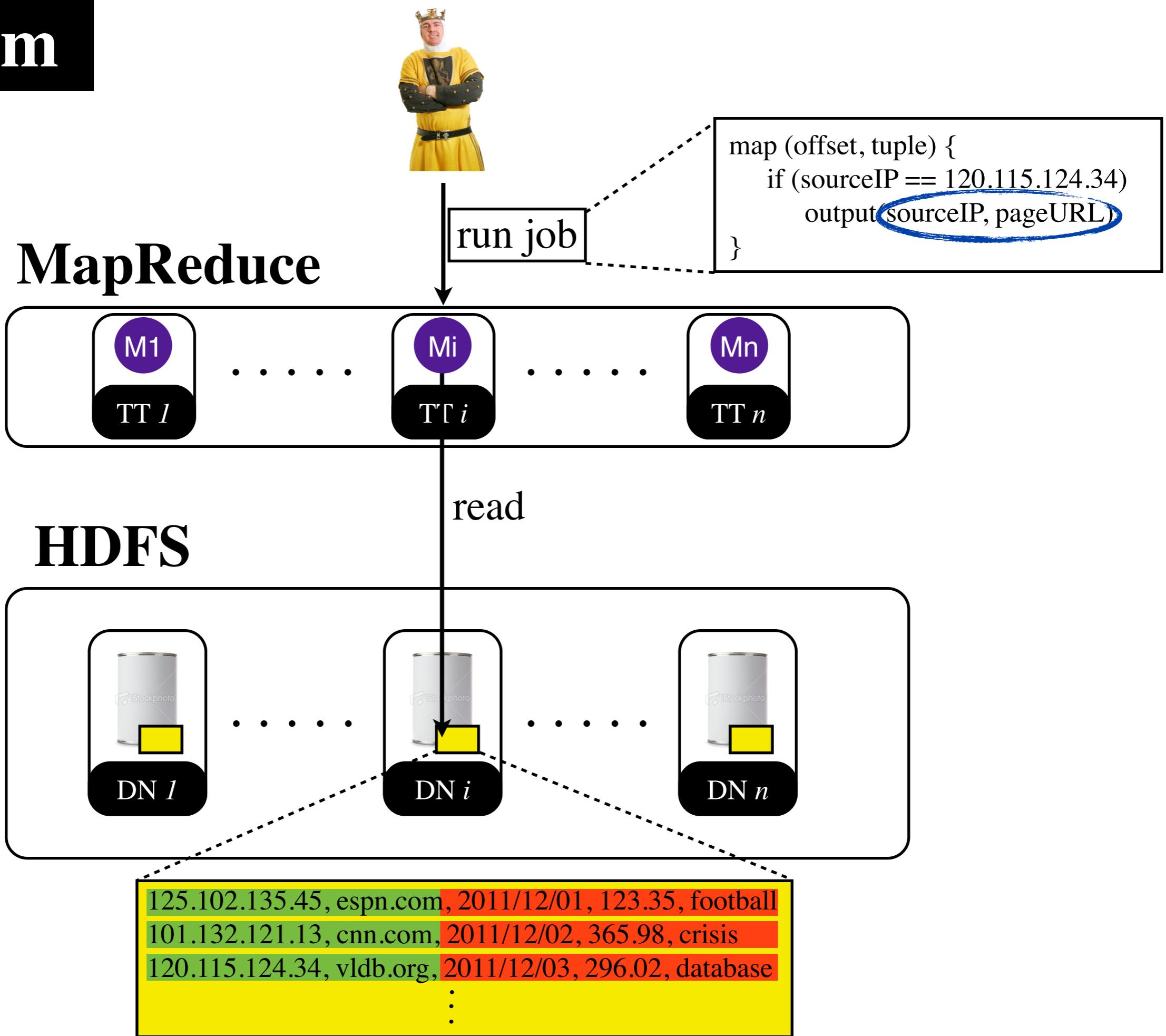
Problem

MapReduce



Problem

MapReduce



Data Layouts in MapReduce

Data Layouts in MapReduce

Initial

Row

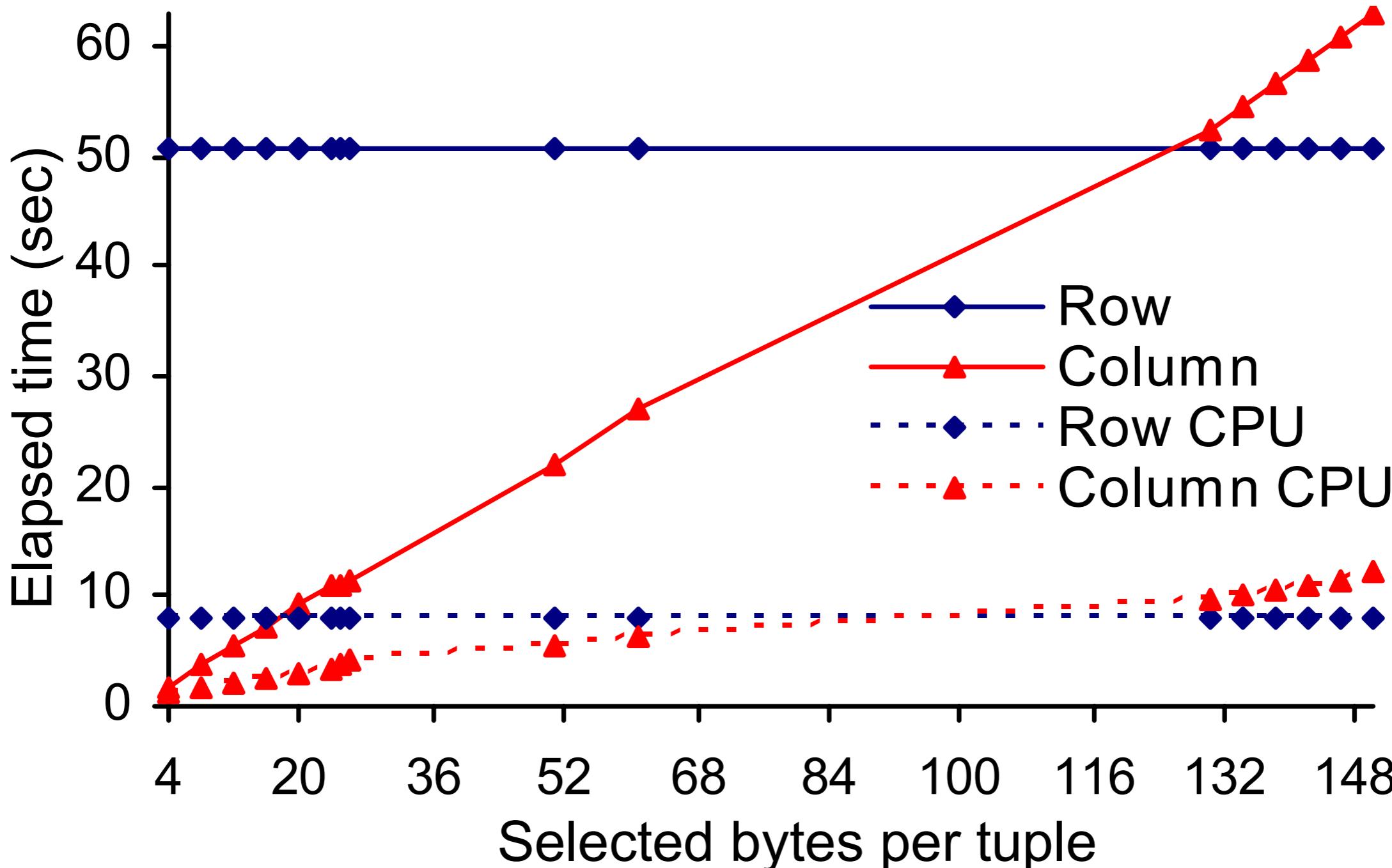
Read Unnecessary
columns

Column Layout

[D. Batory: On Searching Transposed Files. ACM TODS 1979]

[G. Copeland, S. Khoshafian: A Decomposition Storage Model. SIGMOD 1985].

Column vs Row Layout



**SELECT a1, a2, ... FROM Lineitem
WHERE predicate (a1) yields 10% selectivity**

Column Layout in MapReduce?

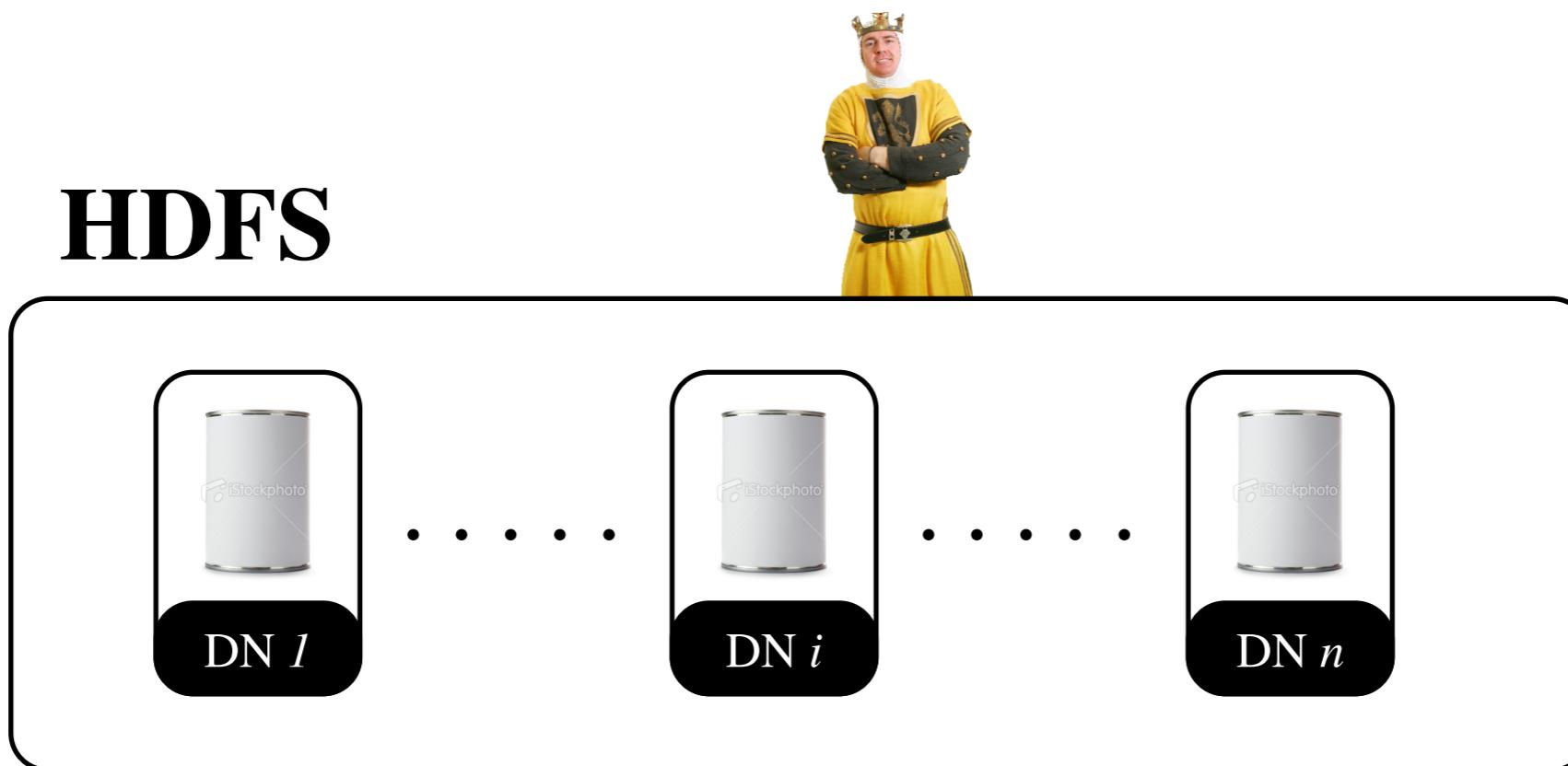
Column-wise File (CFile)

[Y. Lin et al.: Llama: Leveraging Columnar Storage for Scalable Join Processing
in the MapReduce Framework. SIGMOD 2011.]

Data Upload

UserVisits Log

```
125.102.135.45, espn.com, 2011/12/01, 123.35, football  
101.132.121.13, cnn.com, 2011/12/02, 365.98, crisis  
120.115.124.34, vldb.org, 2011/12/03, 296.02, database
```

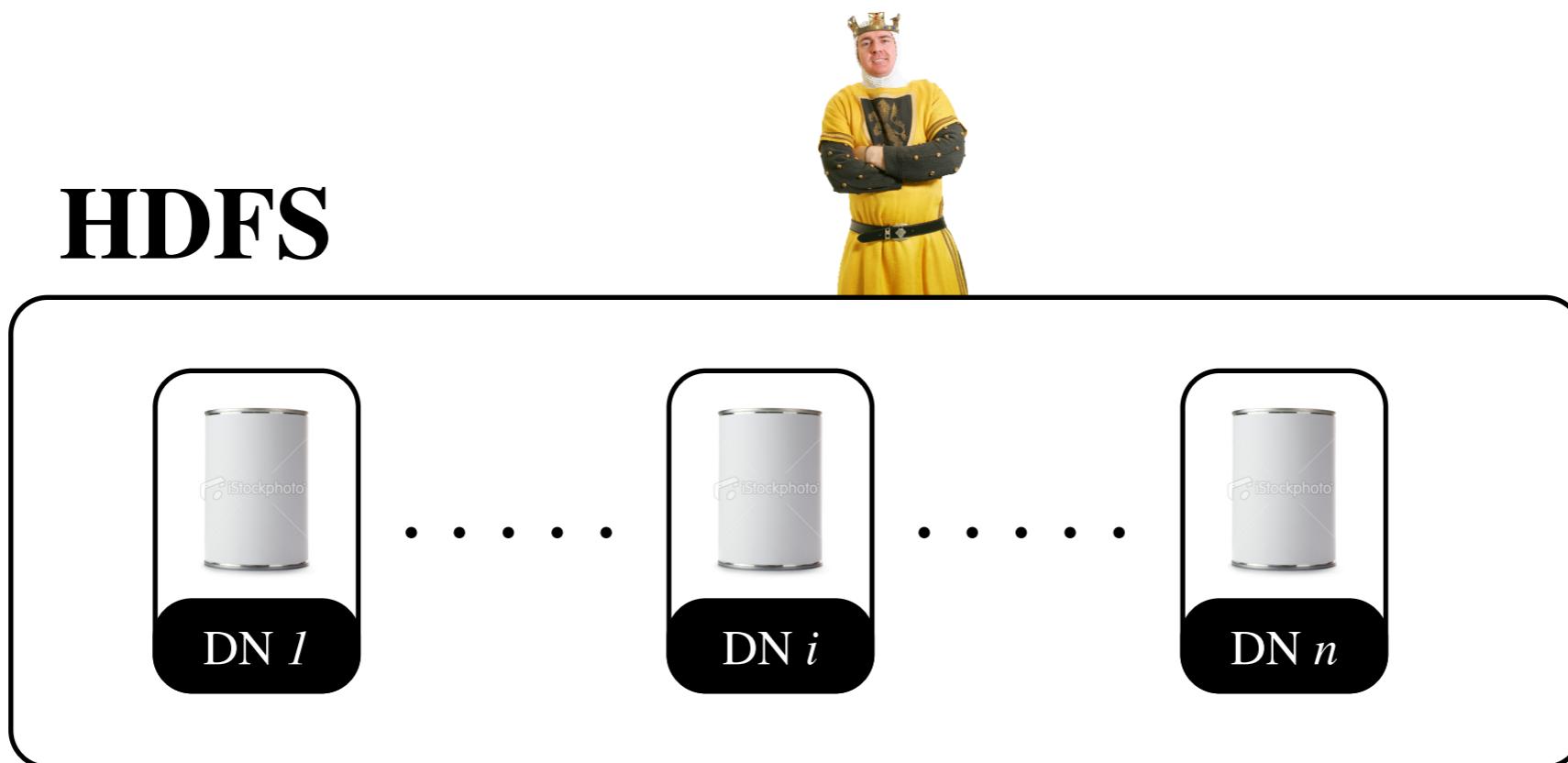


Data Upload

UserVisits Log

125.102.135.45	espn.com	2011/12/01	123.35	football
101.132.121.13	cnn.com	2011/12/02	365.98	crisis
120.115.124.34	vldb.org	2011/12/03	296.02	database

HDFS



Data Upload

UserVisits Log

125.102.135.45
101.132.121.13
120.115.124.34

espn.com
cnn.com
vldb.org

2011/12/01
2011/12/02
2011/12/03

Vertical Group 1 (basic group)

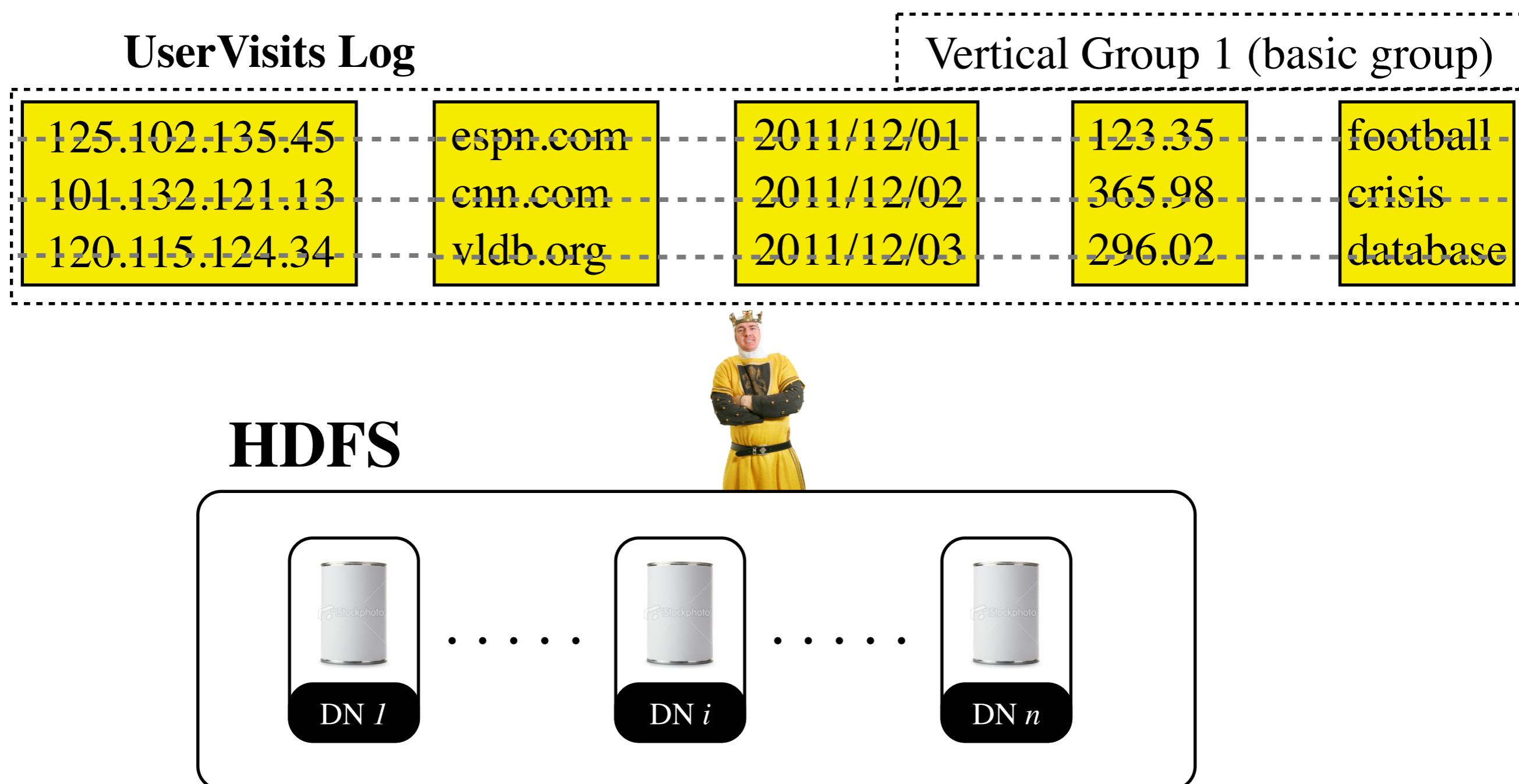
123.35
365.98
296.02

football
crisis
database

HDFS



Data Upload



Data Upload

UserVisits Log

101.132.121.13
120.115.124.31
125.102.135.45

espn.com
cnn.com
vldb.org

2011/12/01
2011/12/02
2011/12/03

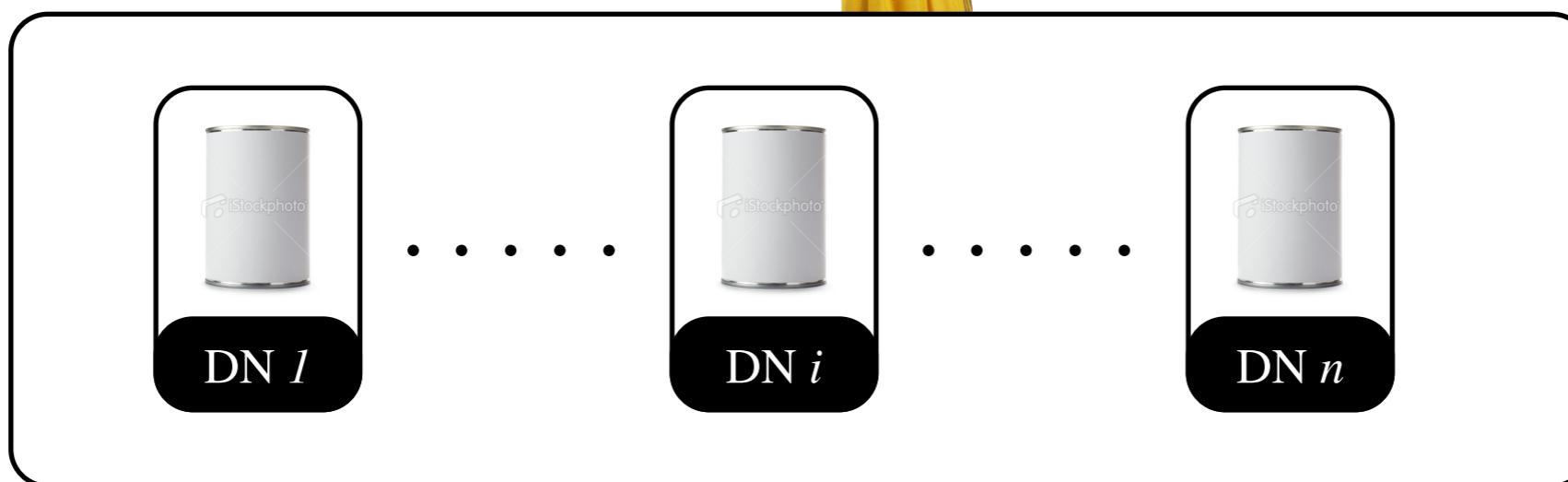
123.35
365.98
296.02

football
crisis
database

Vertical Group 1 (basic group)



HDFS



Data Upload

UserVisits Log

101.132.121.13
120.115.124.31
125.102.135.45

cnn.com
vldb.org
espn.com

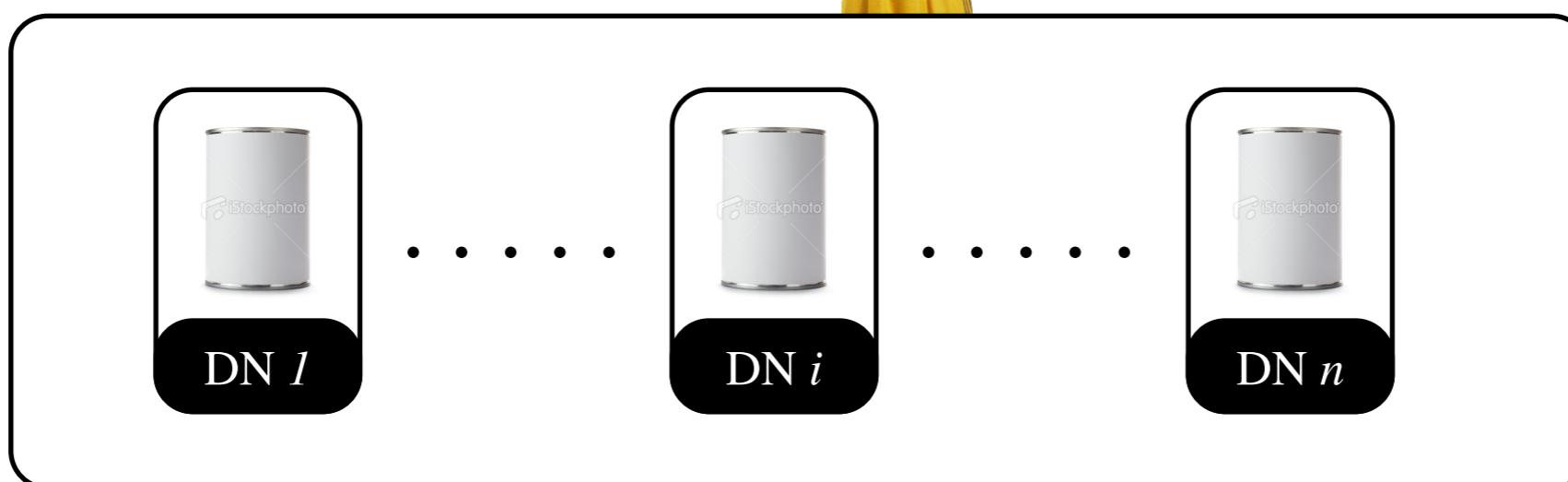
2011/12/02
2011/12/03
2011/12/01

365.98
296.02
123.35

crisis
database
football

Vertical Group 1 (basic group)

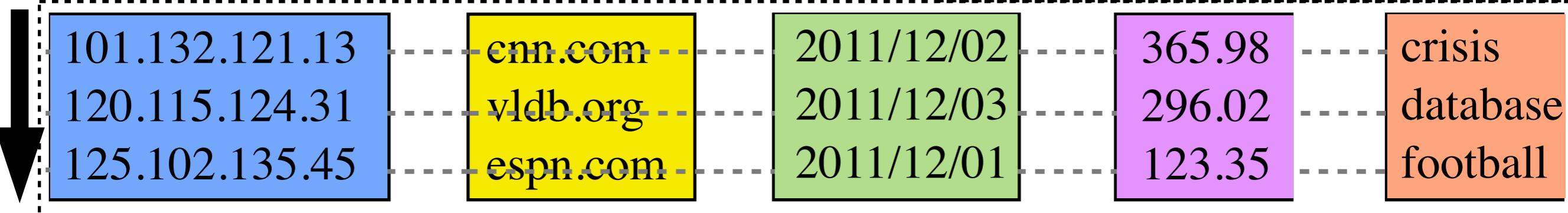
HDFS



Data Upload

- CFile-sourceIp
- CFile-pageURL
- CFile-visitDate
- CFile-adRevenue
- CFile-searchWord

UserVisits Log



HDFS



Data Upload

- CFile-sourceIp
- CFile-pageURL
- CFile-visitDate
- CFile-adRevenue
- CFile-searchWord

UserVisits Log

101.132.121.13
120.115.124.31
125.102.135.45

cnn.com
vldb.org
espn.com

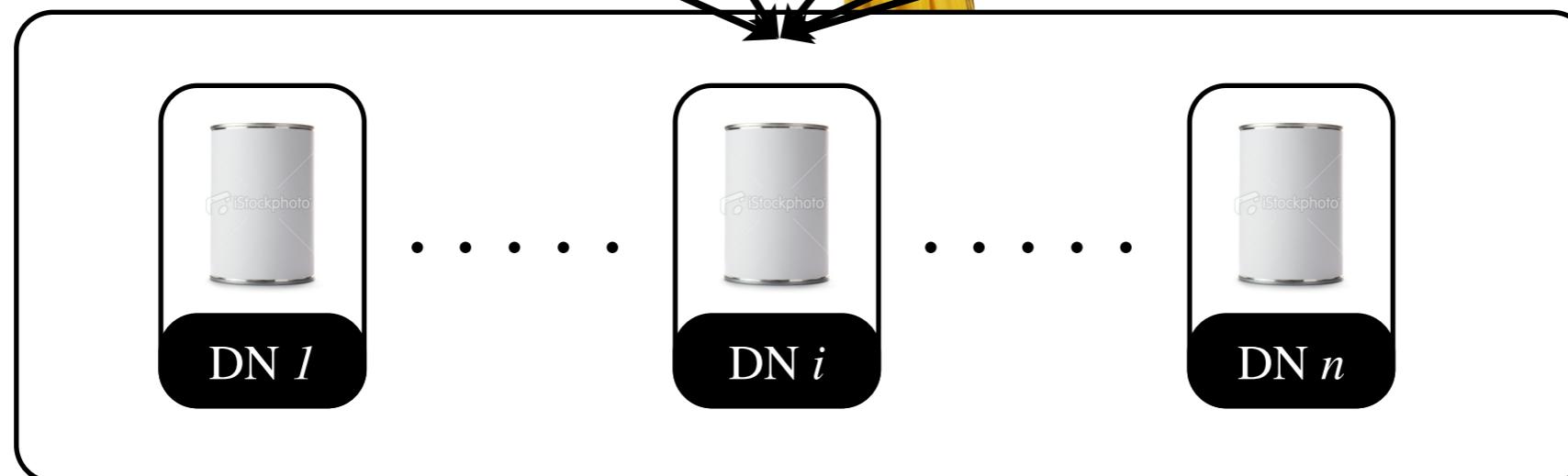
2011/12/02
2011/12/03
2011/12/01

365.98
296.02
123.35

crisis
database
football

HDFS

upload



Data Upload

- CFile-sourceIp
- CFile-pageURL
- CFile-visitDate
- CFile-adRevenue
- CFile-searchWord

UserVisits Log

101.132.121.13
120.115.124.31
125.102.135.45

cnn.com
vldb.org
espn.com

2011/12/02
2011/12/03
2011/12/01

365.98
296.02
123.35

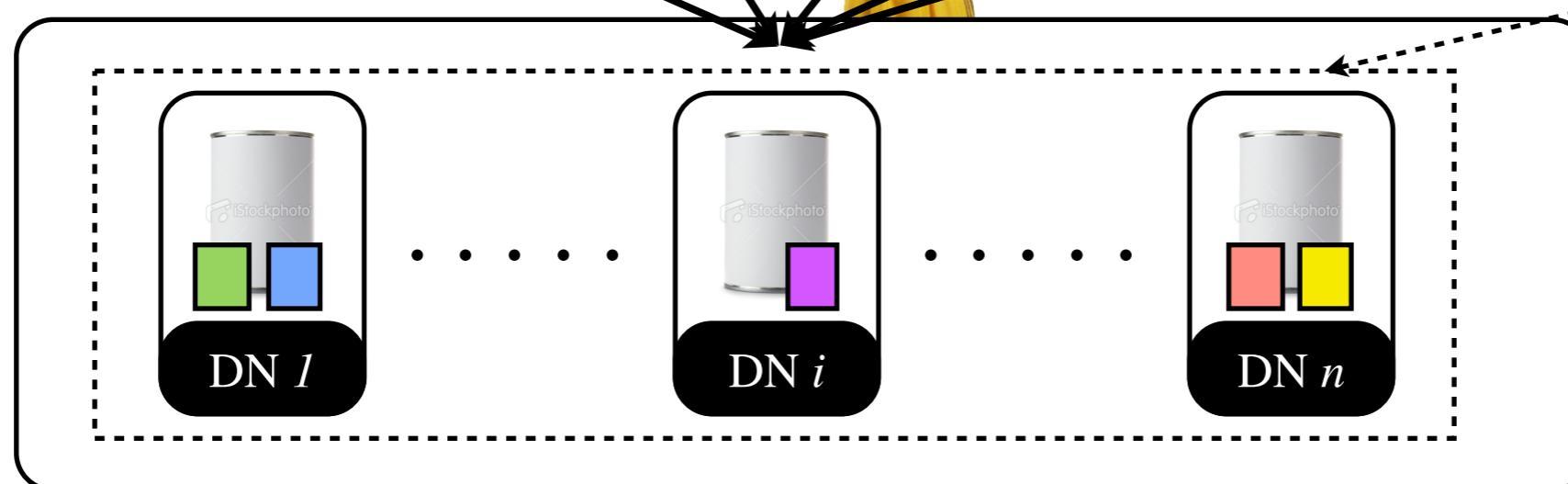
crisis
database
football

HDFS

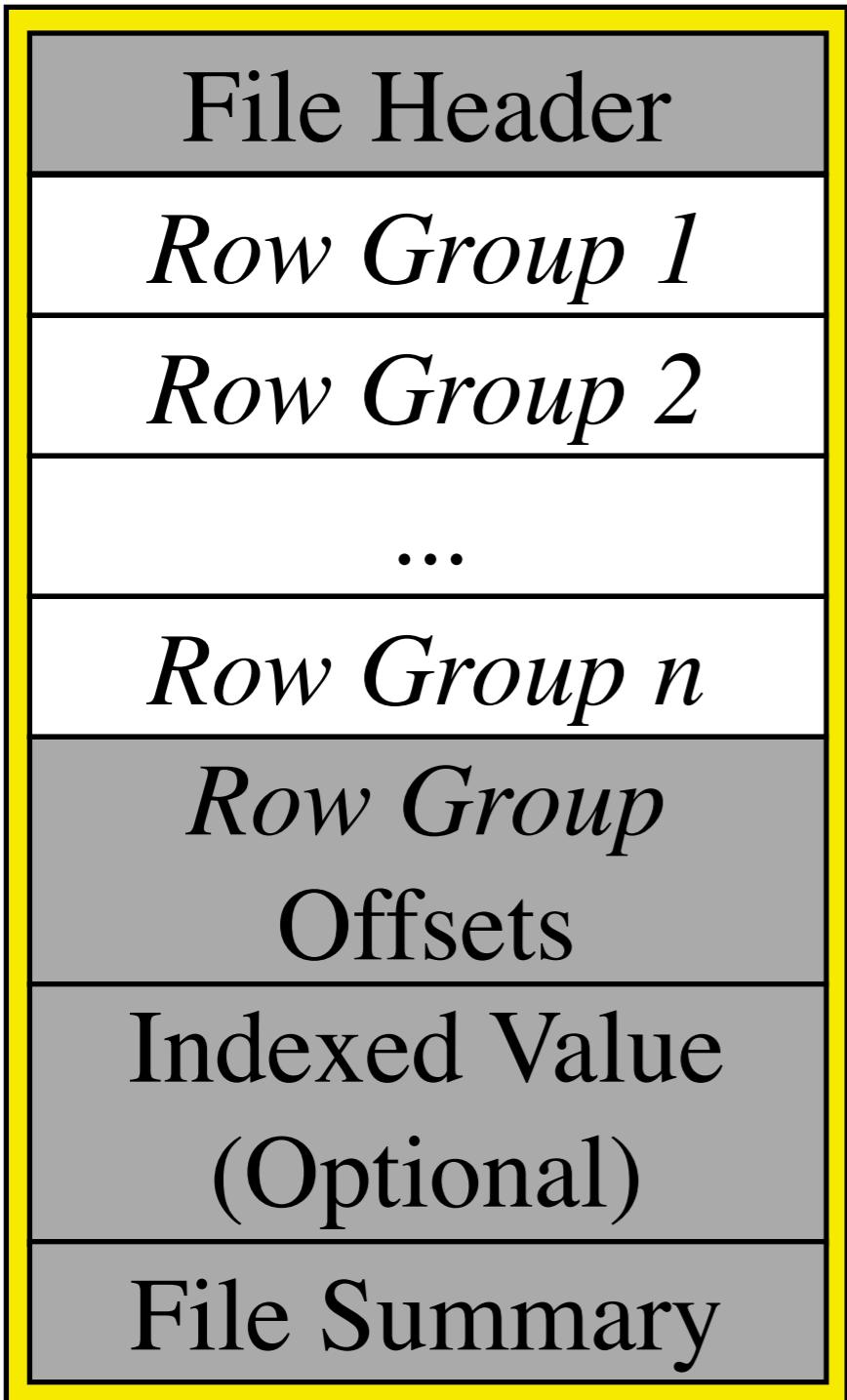
upload



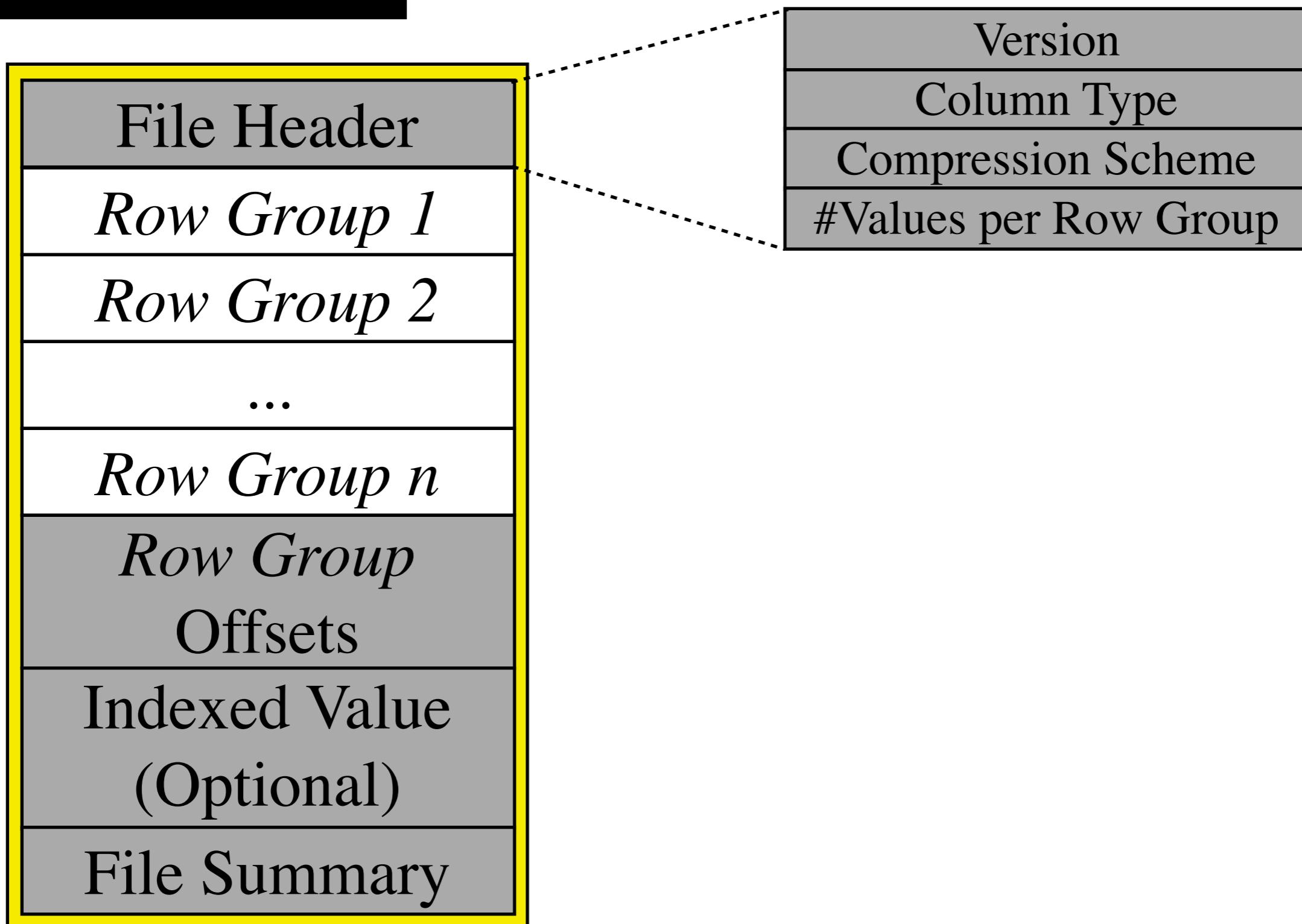
HDFS block 1 replica 1
from all CFiles



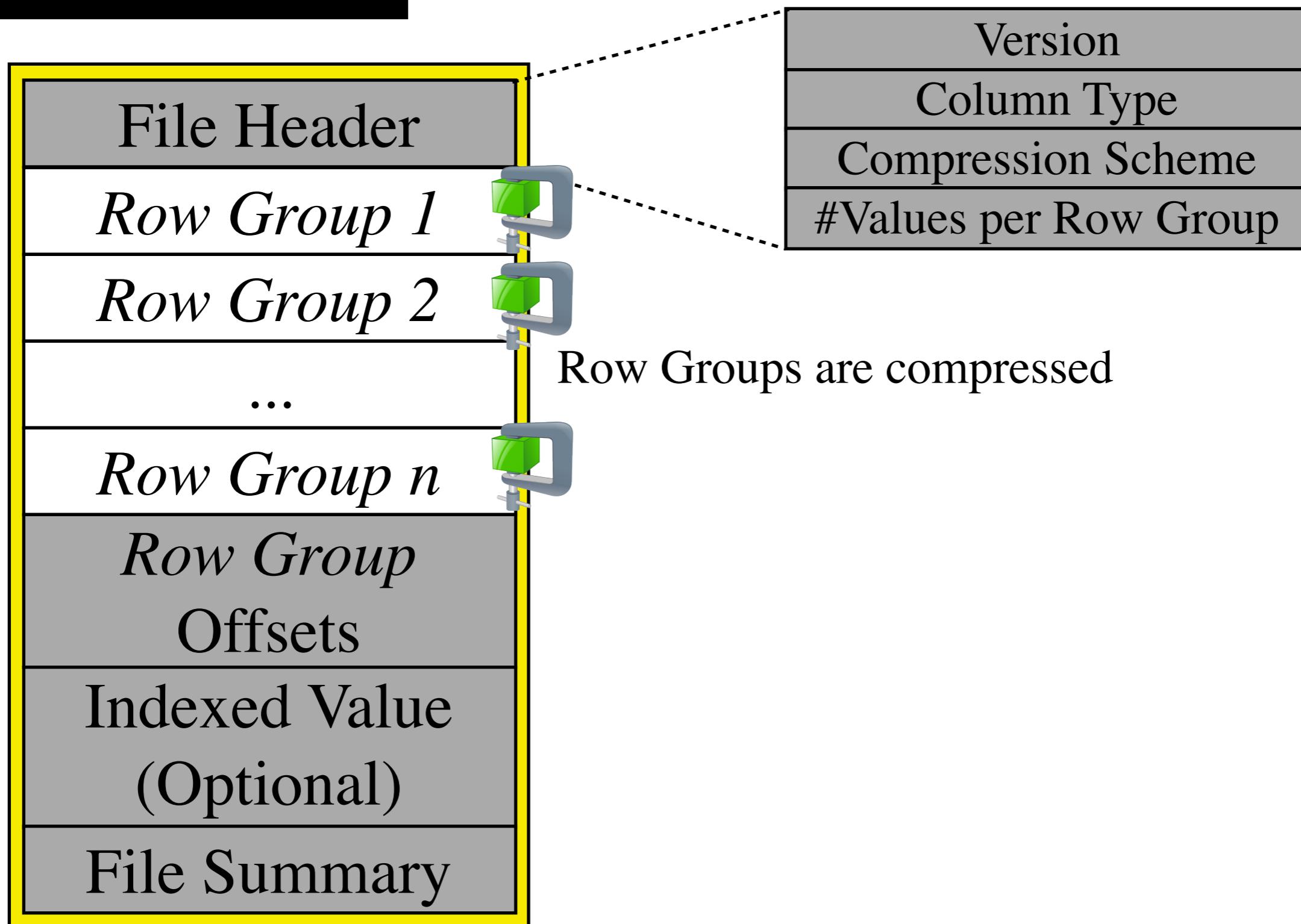
CFile Format



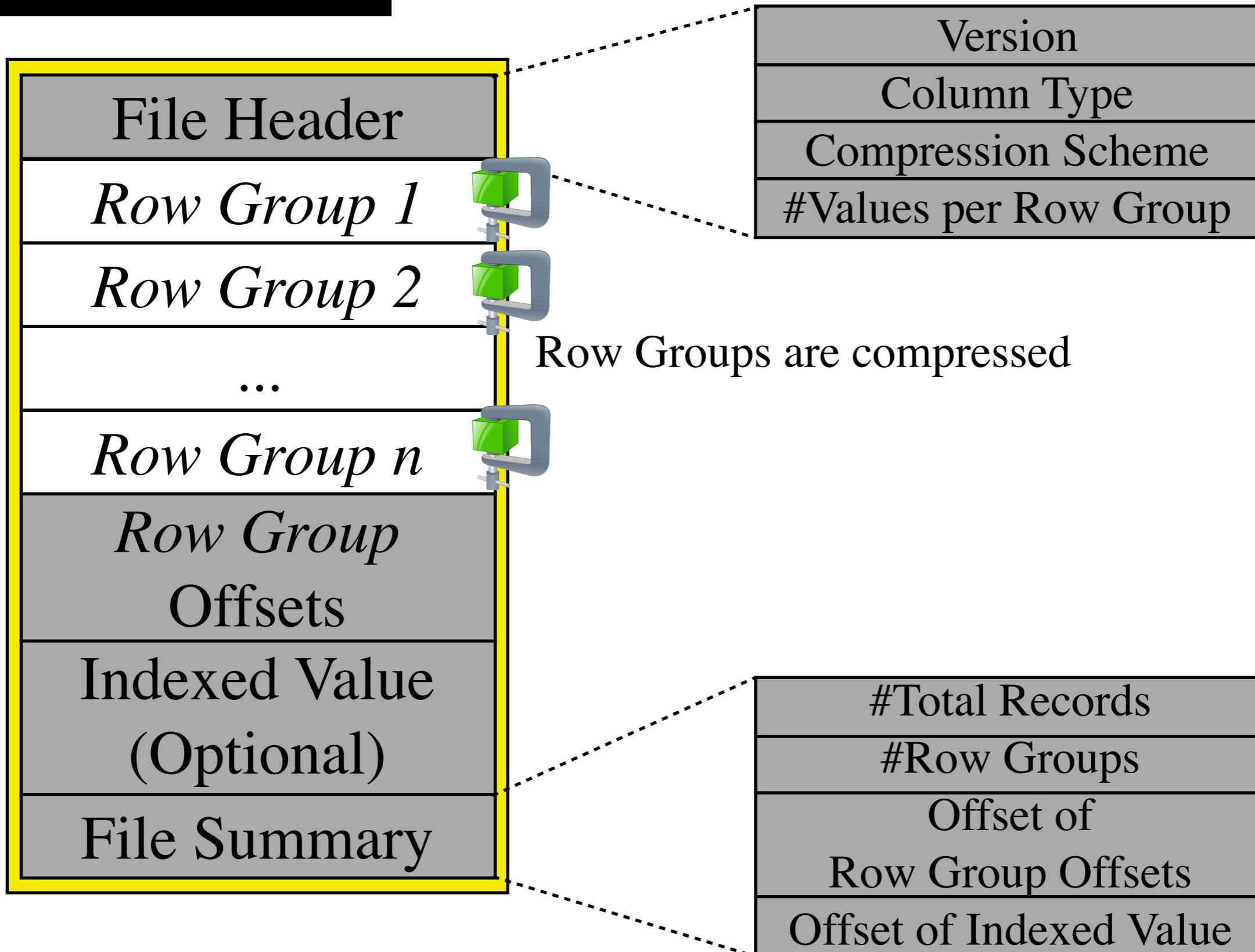
CFile Format



CFile Format



CFile Format



HDFS Blocks for CFile-adRevenue

Column Type: float (4 bytes)

#Total Values = 130,000

Row Group = 1,000 values

HDFS Block Size = 64MB

HDFS Blocks for CFile-adRevenue

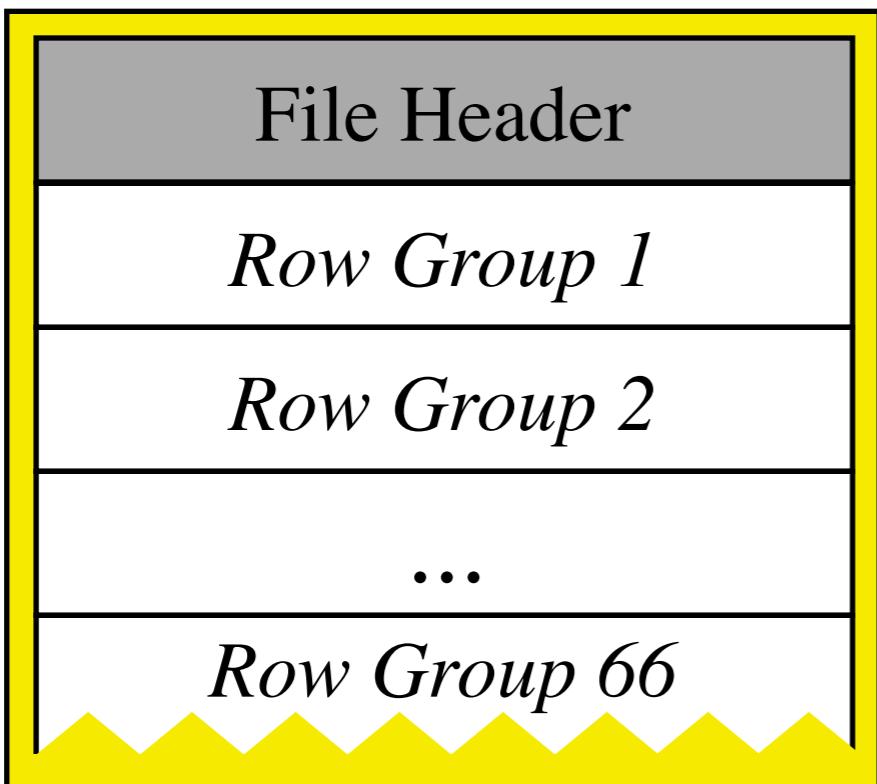
Column Type: float (4 bytes)

#Total Values = 130,000

Row Group = 1,000 values

HDFS Block Size = 64MB

HDFS Block 1



HDFS Blocks for CFile-adRevenue

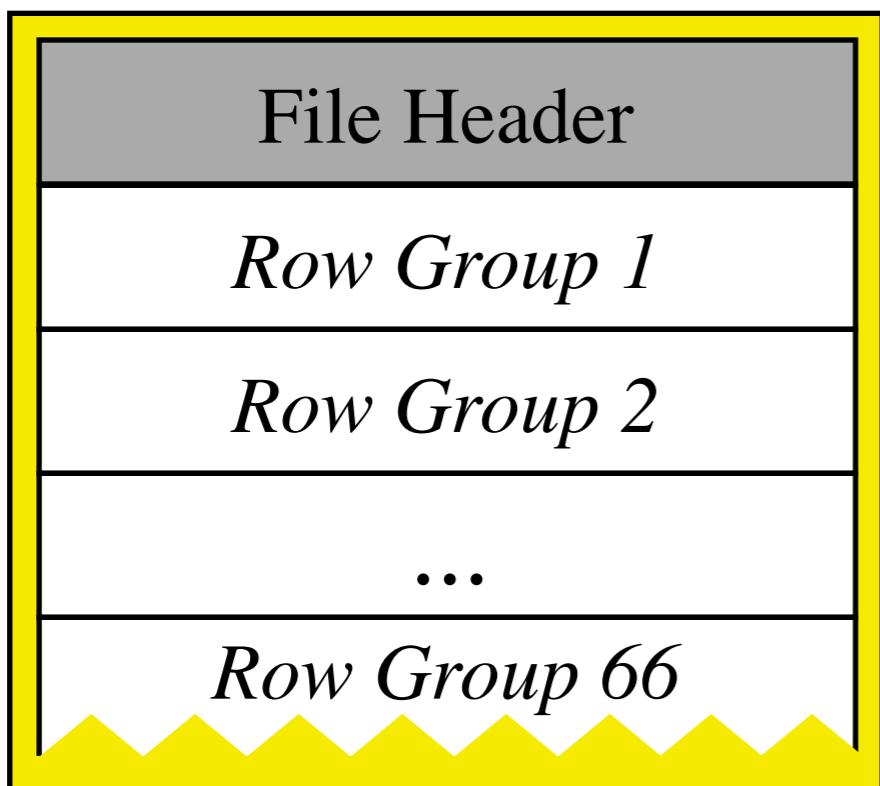
Column Type: float (4 bytes)

#Total Values = 130,000

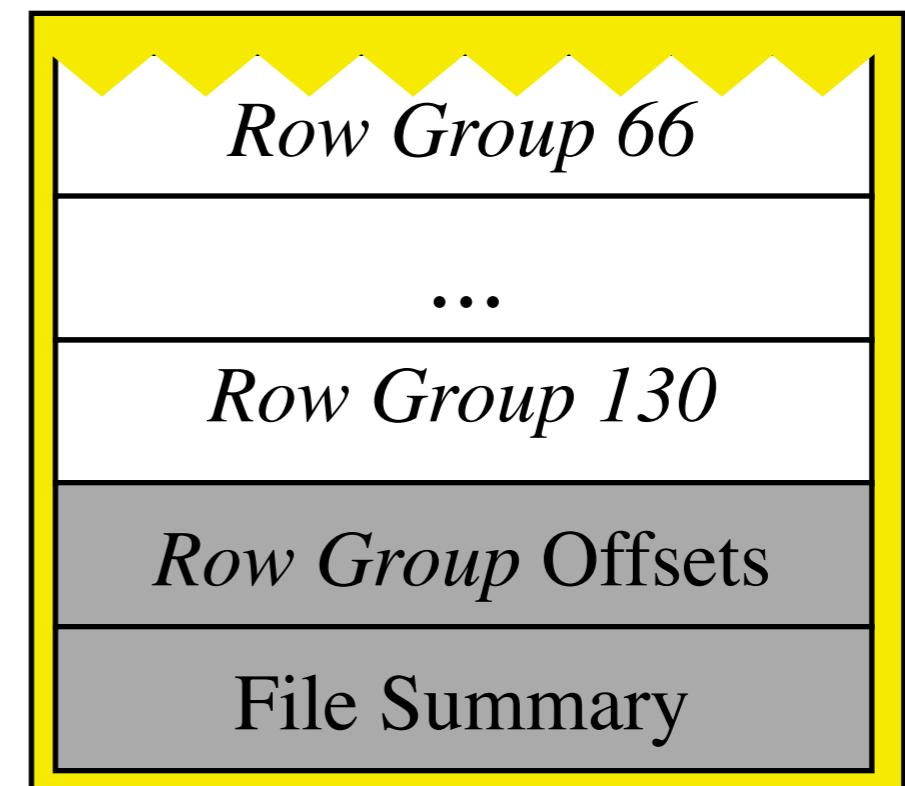
Row Group = 1,000 values

HDFS Block Size = 64MB

HDFS Block 1



HDFS Block 2

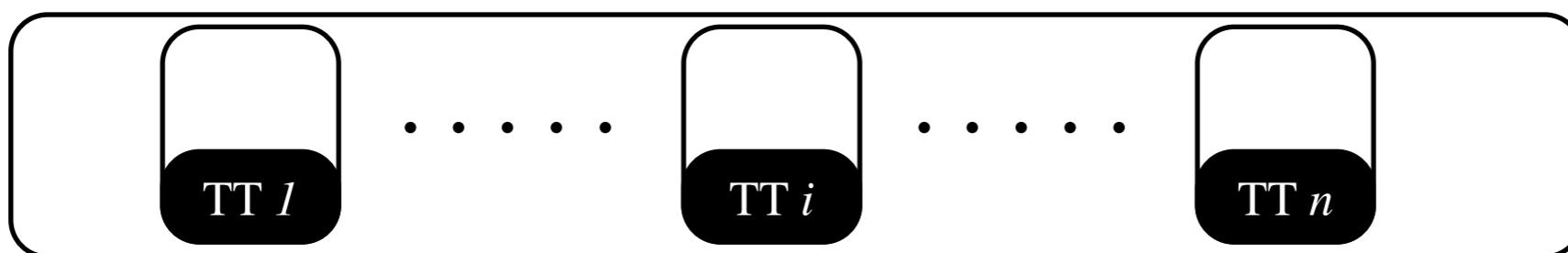


Job Execution

- CFile-sourceIp
- CFile-pageURL
- CFile-visitDate
- CFile-adRevenue
- CFile-searchWord

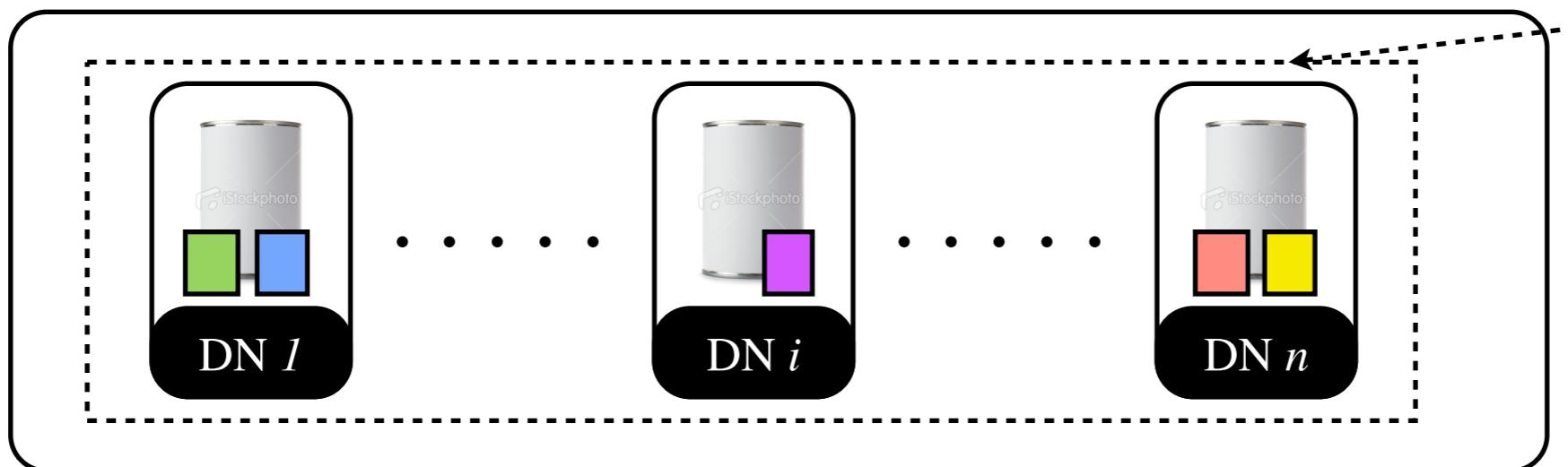


MapReduce



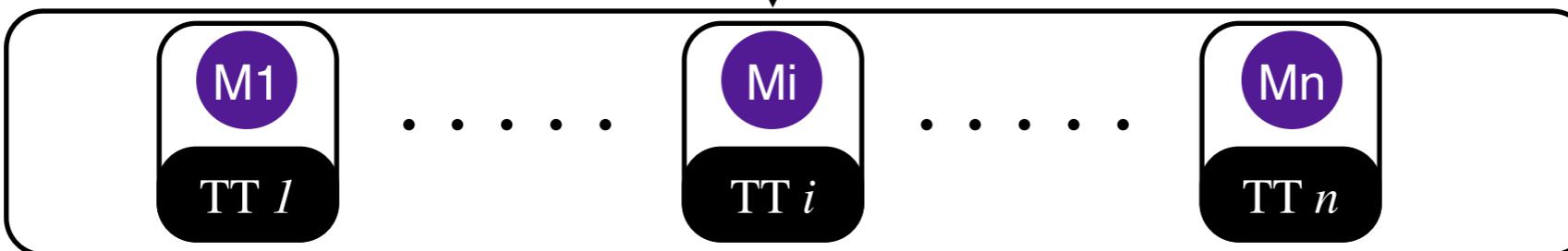
HDFS

HDFS block 1 replica 1
from all CFiles

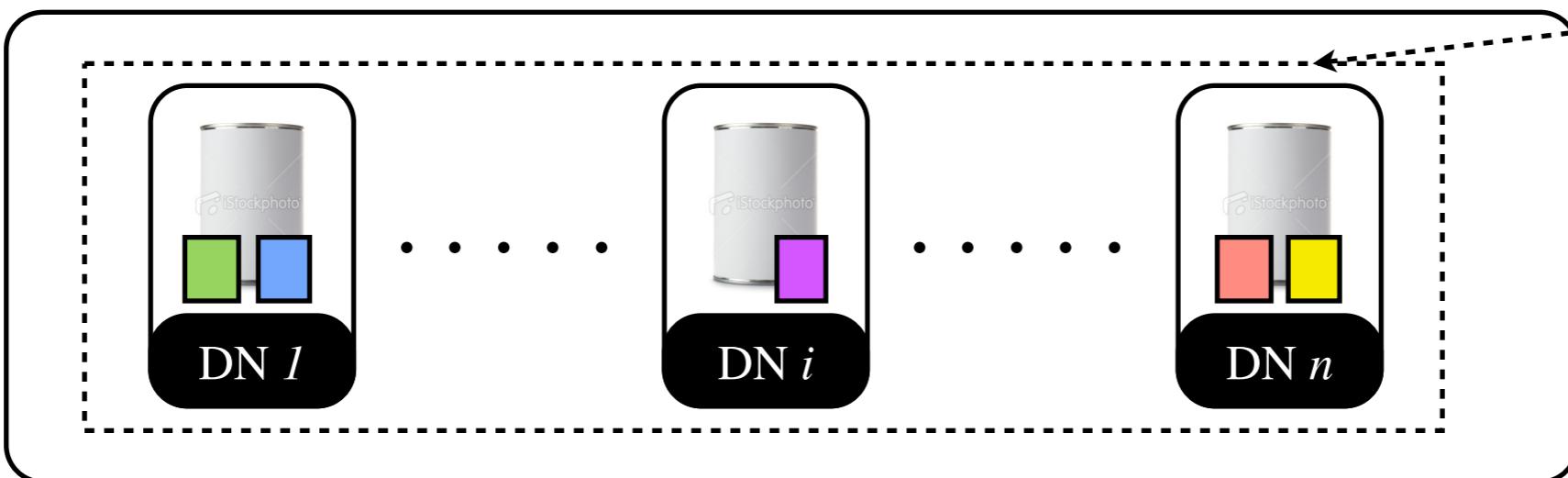


Job Execution

MapReduce



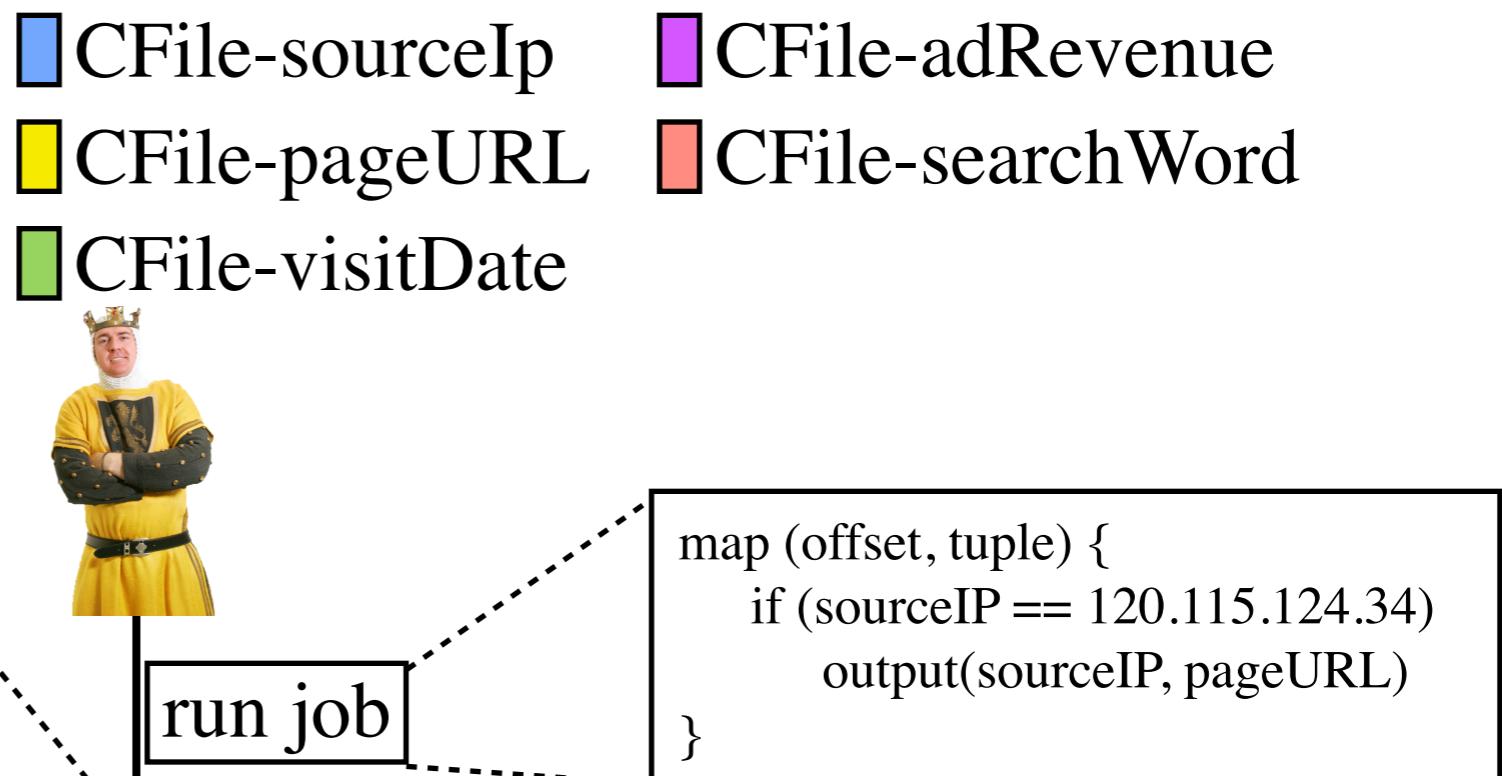
HDFS



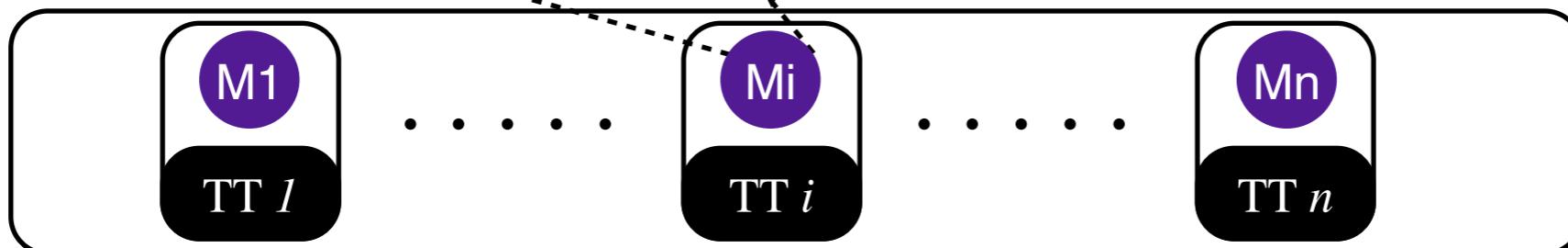
■ CFile-sourceIp
■ CFile-pageURL
■ CFile-visitDate
■ CFile-adRevenue
■ CFile-searchWord

```
map (offset, tuple) {  
    if (sourceIP == 120.115.124.34)  
        output(sourceIP, pageURL)  
}
```

Job Execution

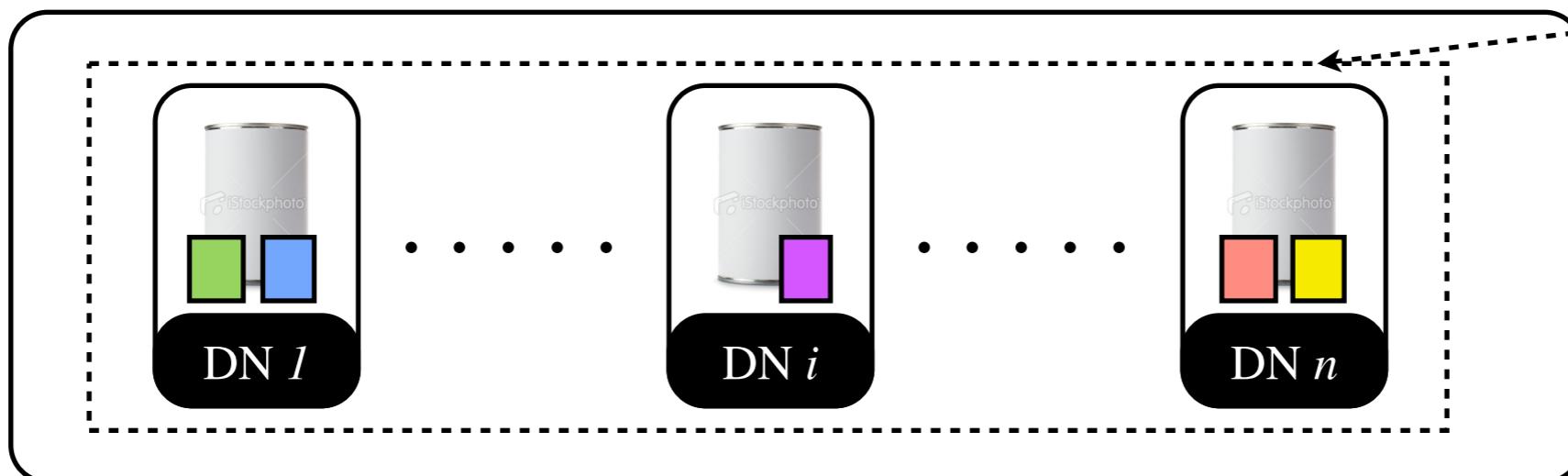


MapReduce

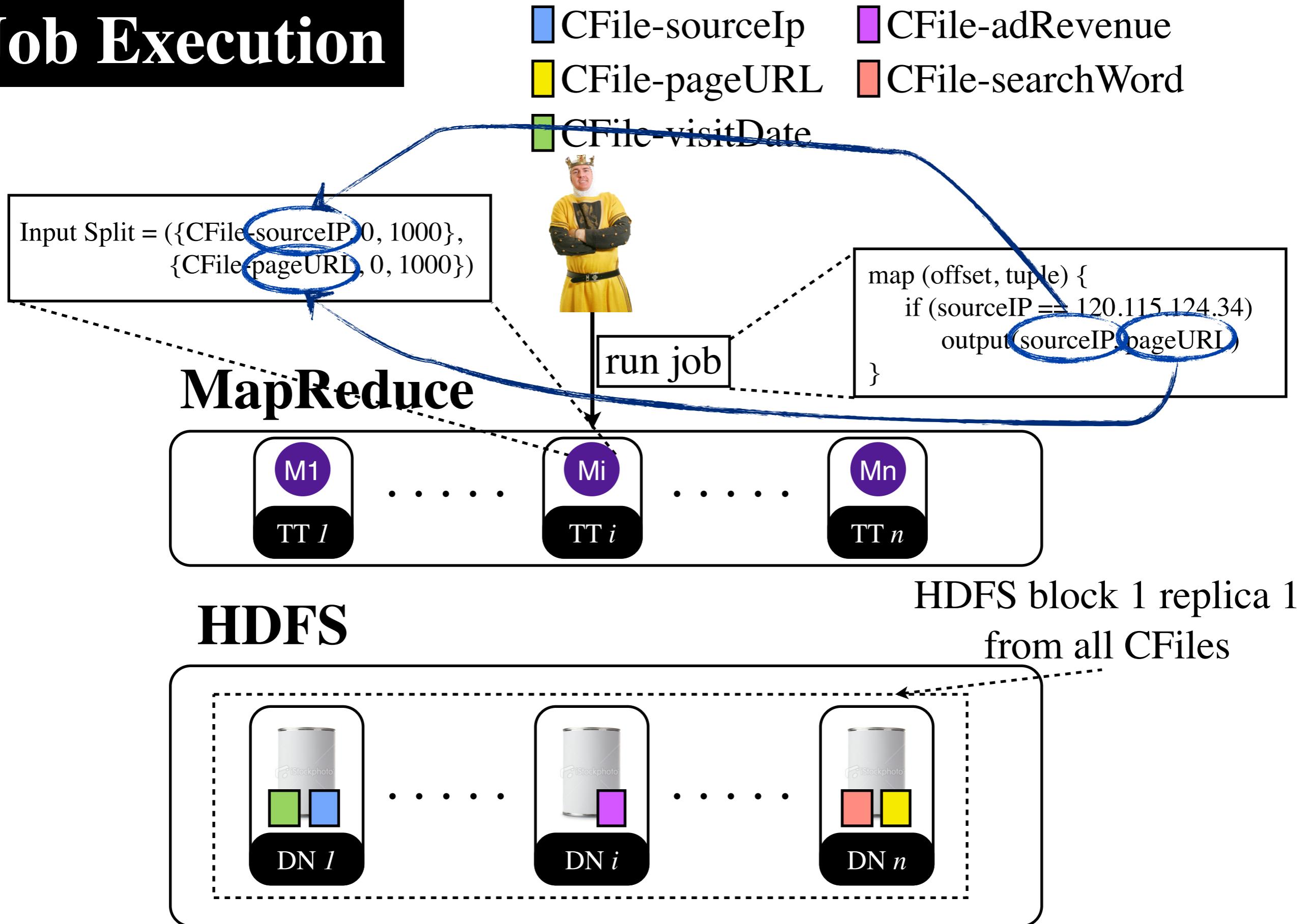


HDFS

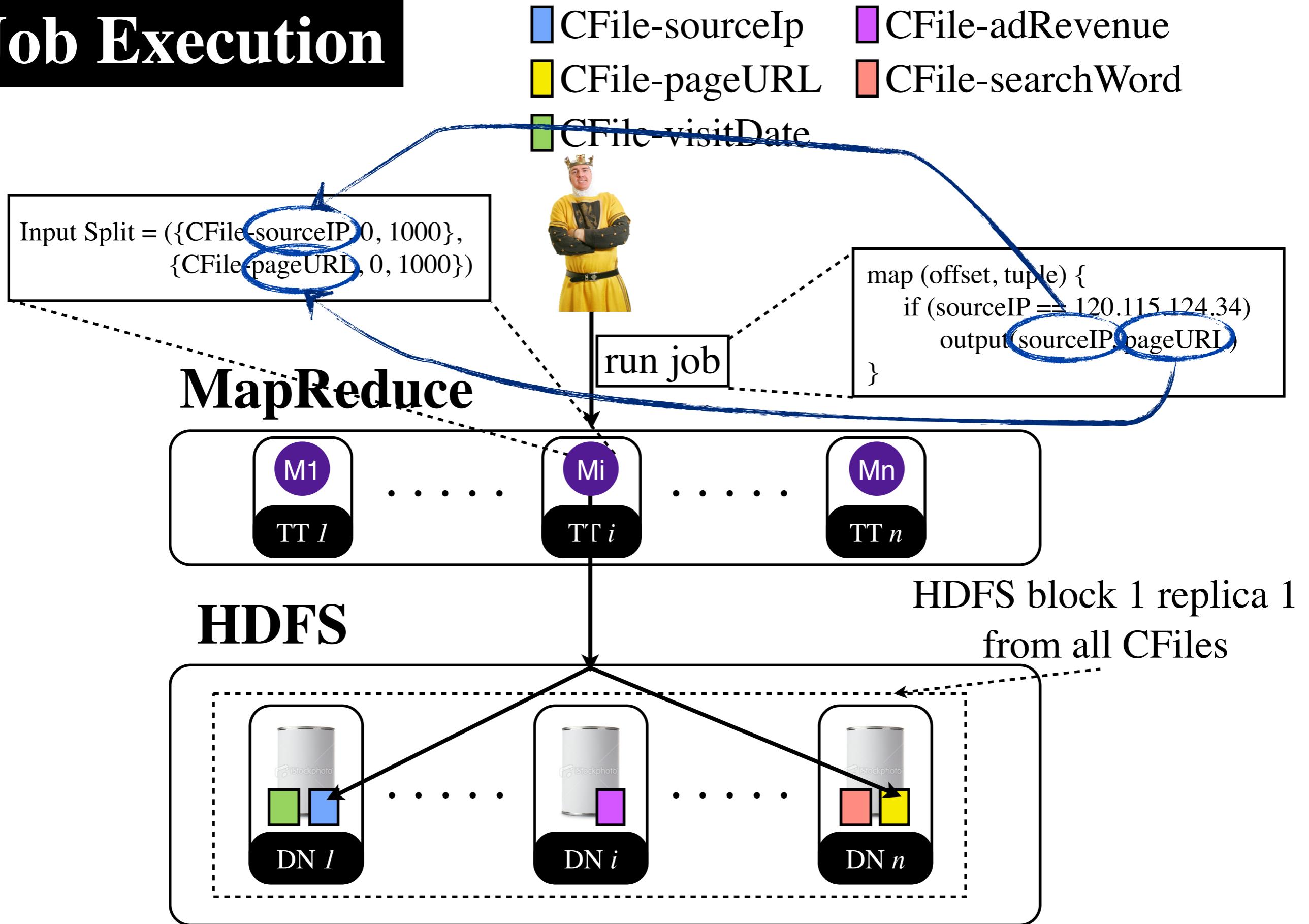
HDFS block 1 replica 1
from all CFiles



Job Execution

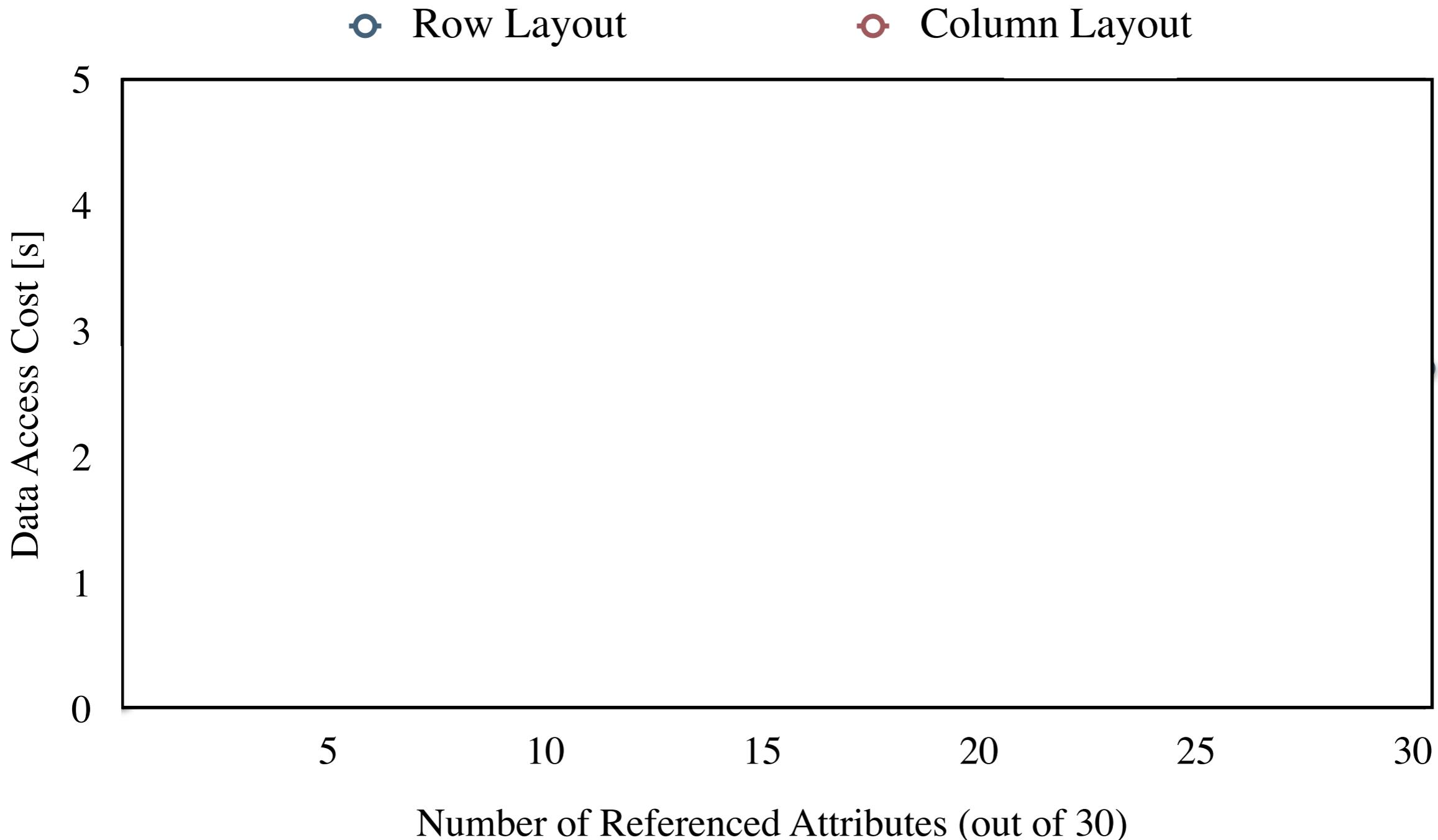


Job Execution



Column Layout in MapReduce

```
SELECT  $a_1, a_2, \dots$   
FROM table30Attrs
```

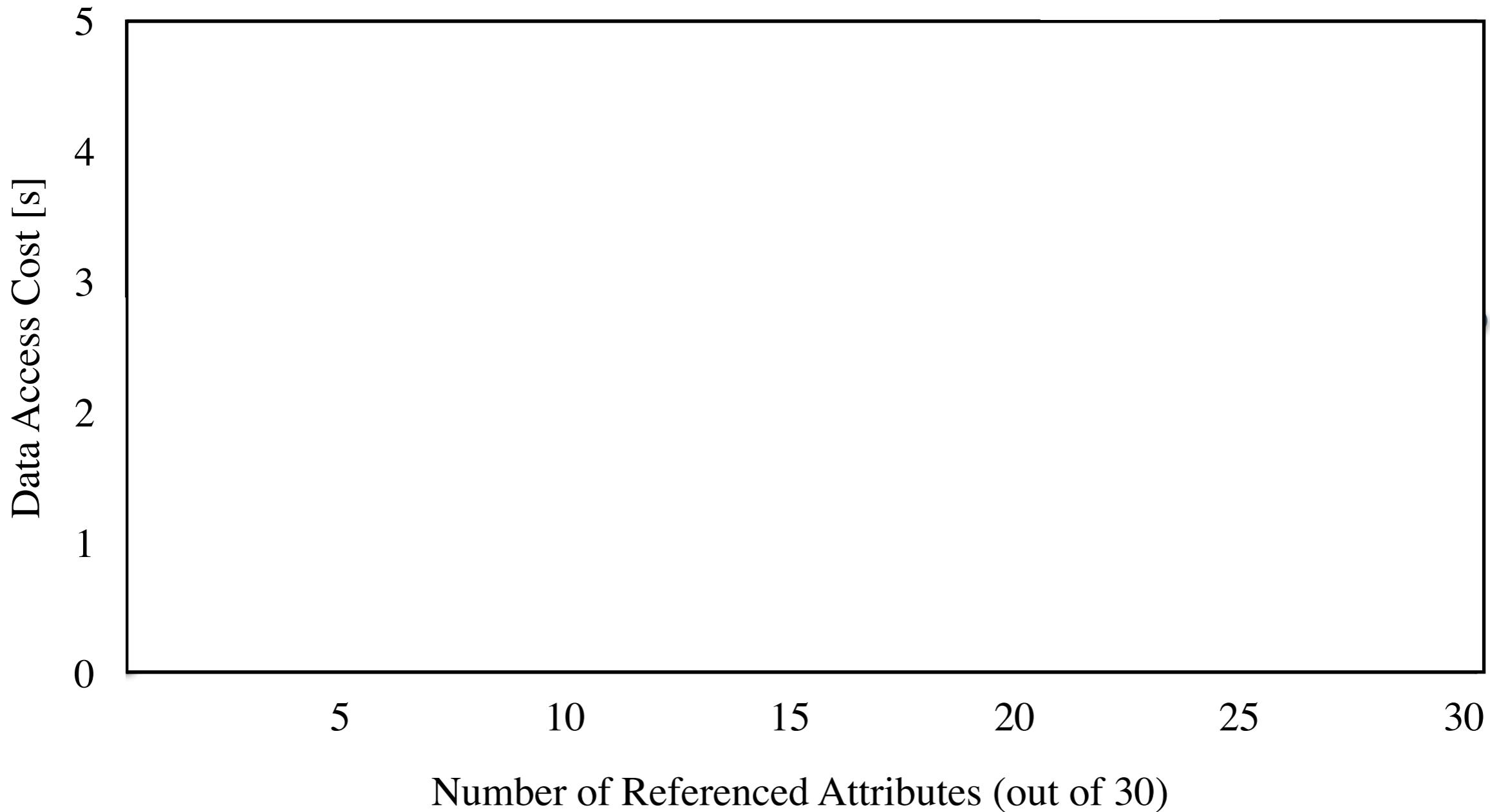


Column Layout in MapReduce

SELECT a_1, a_2, \dots ←----- We vary the number of attributes
FROM table30Attrs

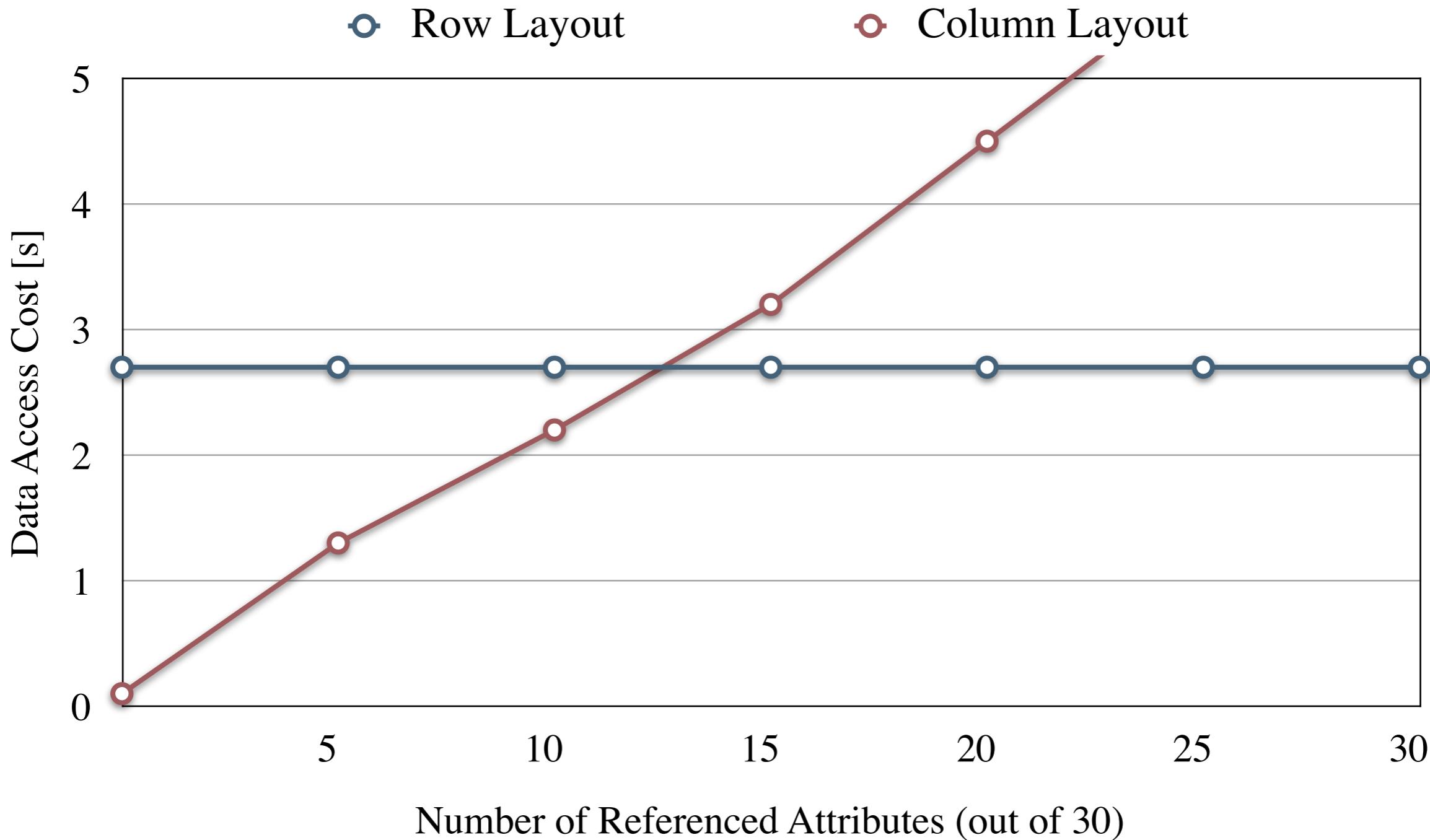
• Row Layout

• Column Layout



Column Layout in MapReduce

SELECT a_1, a_2, \dots ←----- We vary the number of attributes
FROM table30Attrs



Data Layouts in MapReduce

Initial

Row

Read Unnecessary
columns

Data Layouts in MapReduce

Initial	2009
Row	CFile
Read Unnecessary columns	
	Tuple Reconstruction
	High network costs

PAX

[A. Ailamaki et al.: Weaving Relations for Cache Performance. VLDB 2001]

UserVisits Log

125.102.135.45, espn.com, 2011/12/01, 123.35, football

101.132.121.13, cnn.com, 2011/12/02, 365.98, crisis

120.115.124.34, vldb.org, 2011/12/03, 296.02, database

⋮

⋮

⋮

102.192.235.245, voici.com, 2011/12/19, 630.30, queen

145.111.145.1, sports.com, 2011/12/20, 365.98, basket

123.95.100.24, abc.com, 2011/12/21, 26.02, politics

Recap

UserVisits Log

Row Group 1

125.102.135.45, espn.com, 2011/12/01, 123.35, football
101.132.121.13, cnn.com, 2011/12/02, 365.98, crisis
120.115.124.34, vldb.org, 2011/12/03, 296.02, database

•
•
•

Row Group n

102.192.235.245, voici.com, 2011/12/19, 955.83, people
145.111.145.1, sports.com, 2011/12/20, 630.30, basket
123.95.100.24, abc.com, 2011/12/21, 26.02, politics

Recap

UserVisits Log

Row Group 1

125.102.135.45, espn.com, 2011/12/01, 123.35, football
101.132.121.13, cnn.com, 2011/12/02, 365.98, crisis
120.115.124.34, vldb.org, 2011/12/03, 296.02, database

•
•
•

Row Group n

102.192.235.245, voici.com, 2011/12/19, 955.83, people
145.111.145.1, sports.com, 2011/12/20, 630.30, basket
123.95.100.24, abc.com, 2011/12/21, 26.02, politics

Size of a Row Group = Disk Block Size
(but can be any arbitrary size)

Recap

UserVisits Log

Row Group 1

125.102.135.45,	espn.com,	2011/12/01,	123.35,	football
101.132.121.13,	cnn.com,	2011/12/02,	365.98,	crisis
120.115.124.34,	vldb.org,	2011/12/03,	296.02,	database

•
•
•

Row Group n

102.192.235.245,	voici.com,	2011/12/19,	955.83,	people
145.111.145.1,	sports.com,	2011/12/20,	630.30,	basket
123.95.100.24,	abc.com,	2011/12/21,	26.02,	politics

Size of a Row Group = Disk Block Size
(but can be any arbitrary size)

PAX in MapReduce?

Storage in Cheetah

[S. Chen: A High Performance, Custom Data Warehouse on Top of MapReduce.
PVLDB 2010]

Data Upload

UserVisits Log

Row Group 1

125.102.135.45,	espn.com,	2011/12/01,	123.35,	football
101.132.121.13,	cnn.com,	2011/12/02,	365.98,	crisis
120.115.124.34,	vldb.org,	2011/12/03,	296.02,	database

⋮

Row Group n

102.192.235.245,	voici.com,	2011/12/19,	955.83,	people
145.111.145.1,	sports.com,	2011/12/20,	630.30,	basket
123.95.100.24,	abc.com,	2011/12/21,	26.02,	politics

HDFS



Data Upload

UserVisits Log

Row Group 1

125.102.135.45,	espn.com,	2011/12/01,	123.35,	football
101.132.121.13,	cnn.com,	2011/12/02,	365.98,	crisis
120.115.124.34,	vldb.org,	2011/12/03,	296.02,	database

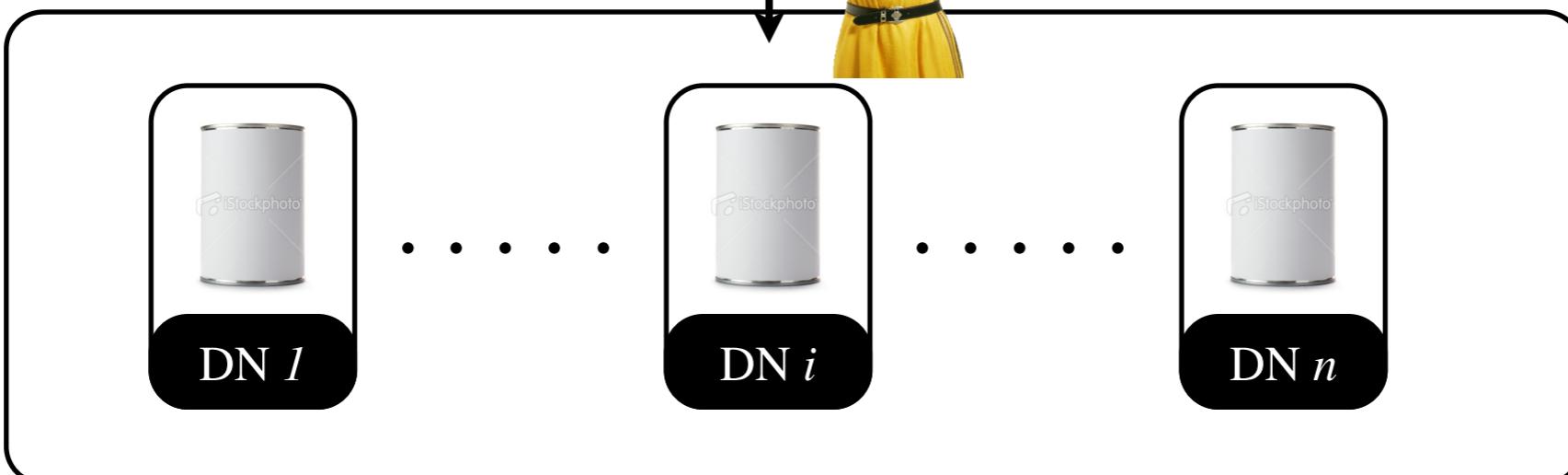
⋮

Row Group n

102.192.235.245,	voici.com,	2011/12/19,	955.83,	people
145.111.145.1,	sports.com,	2011/12/20,	630.30,	basket
123.95.100.24,	abc.com,	2011/12/21,	26.02,	politics

HDFS

upload



Data Upload

UserVisits Log

Row Group 1

125.102.135.45,	espn.com,	2011/12/01,	123.35,	football
101.132.121.13,	cnn.com,	2011/12/02,	365.98,	crisis
120.115.124.34,	vldb.org,	2011/12/03,	296.02,	database

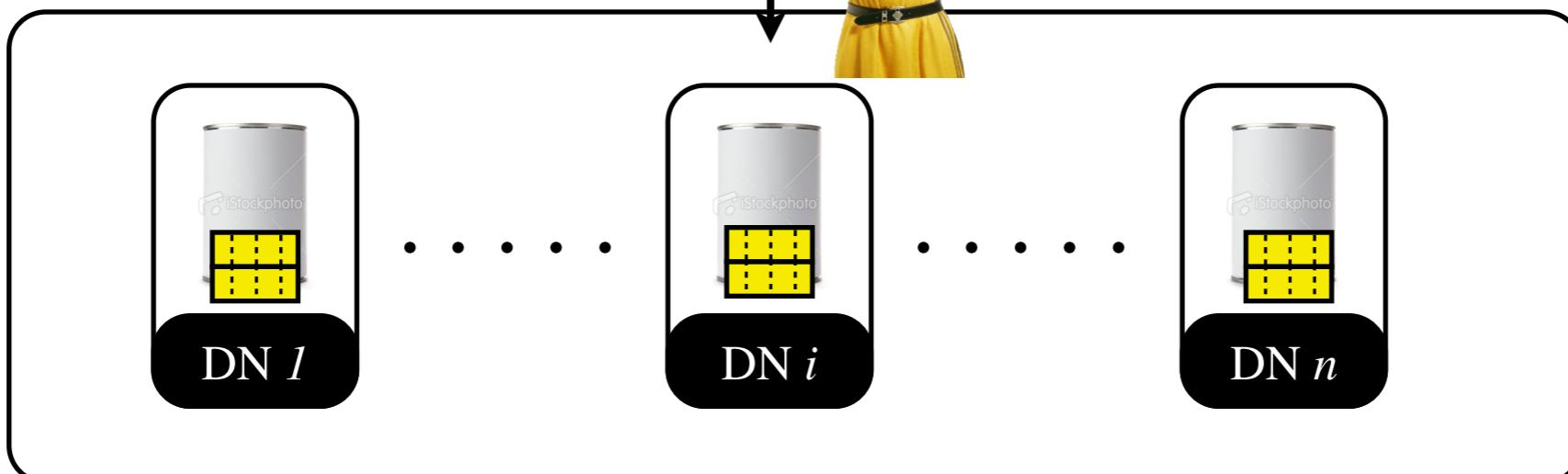
⋮

Row Group n

102.192.235.245,	voici.com,	2011/12/19,	955.83,	people
145.111.145.1,	sports.com,	2011/12/20,	630.30,	basket
123.95.100.24,	abc.com,	2011/12/21,	26.02,	politics

HDFS

upload



HDFS Block Format

Average Record Size: 100 bytes

#Total Records = 1,000,000

Row Group Size = 200,000 records

HDFS Block Size = 64MB

HDFS Block Format

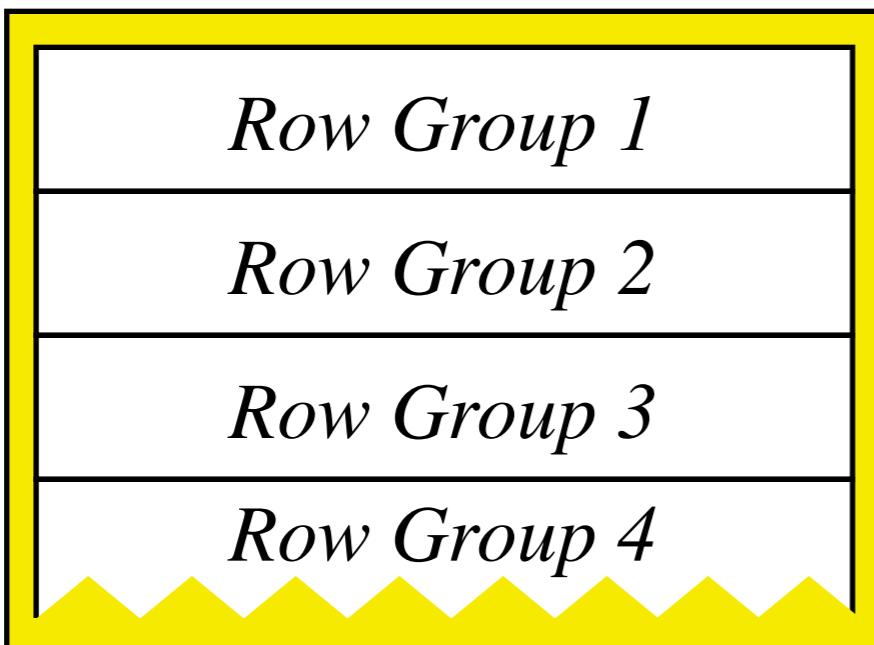
Average Record Size: 100 bytes

#Total Records = 1,000,000

Row Group Size = 200,000 records

HDFS Block Size = 64MB

HDFS Block 1



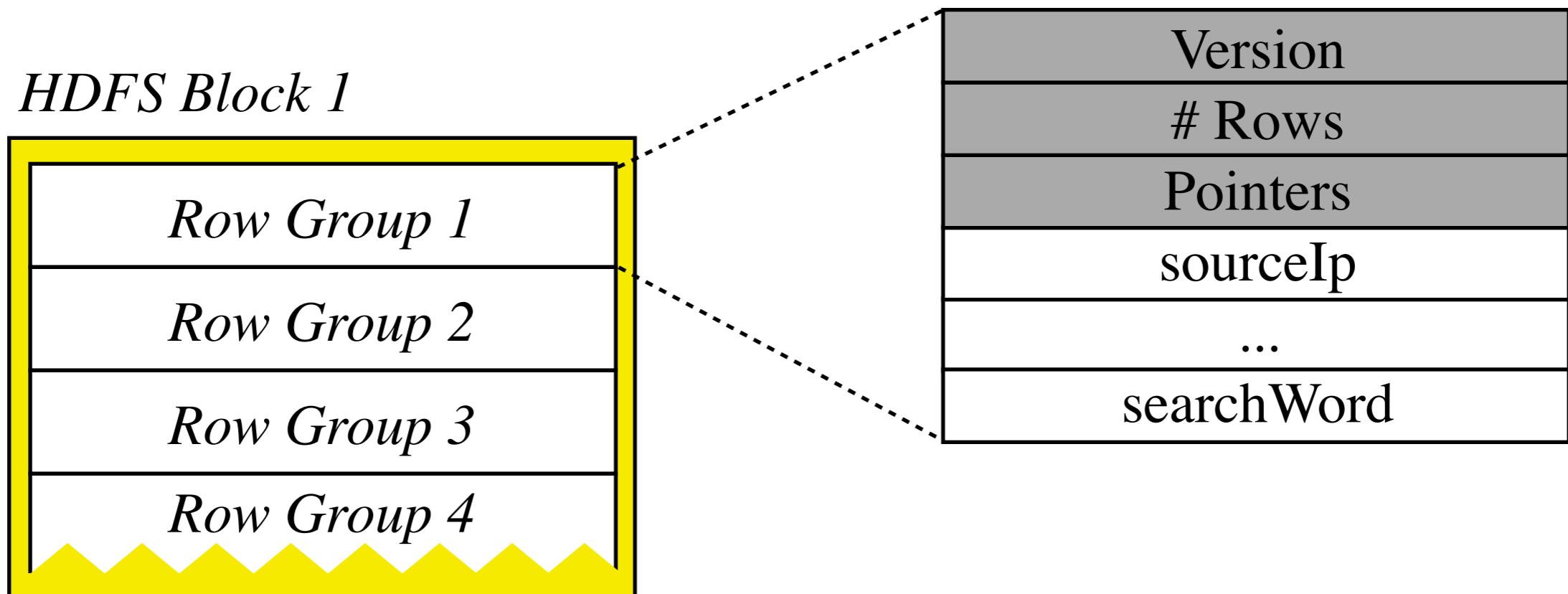
HDFS Block Format

Average Record Size: 100 bytes

#Total Records = 1,000,000

Row Group Size = 200,000 records

HDFS Block Size = 64MB



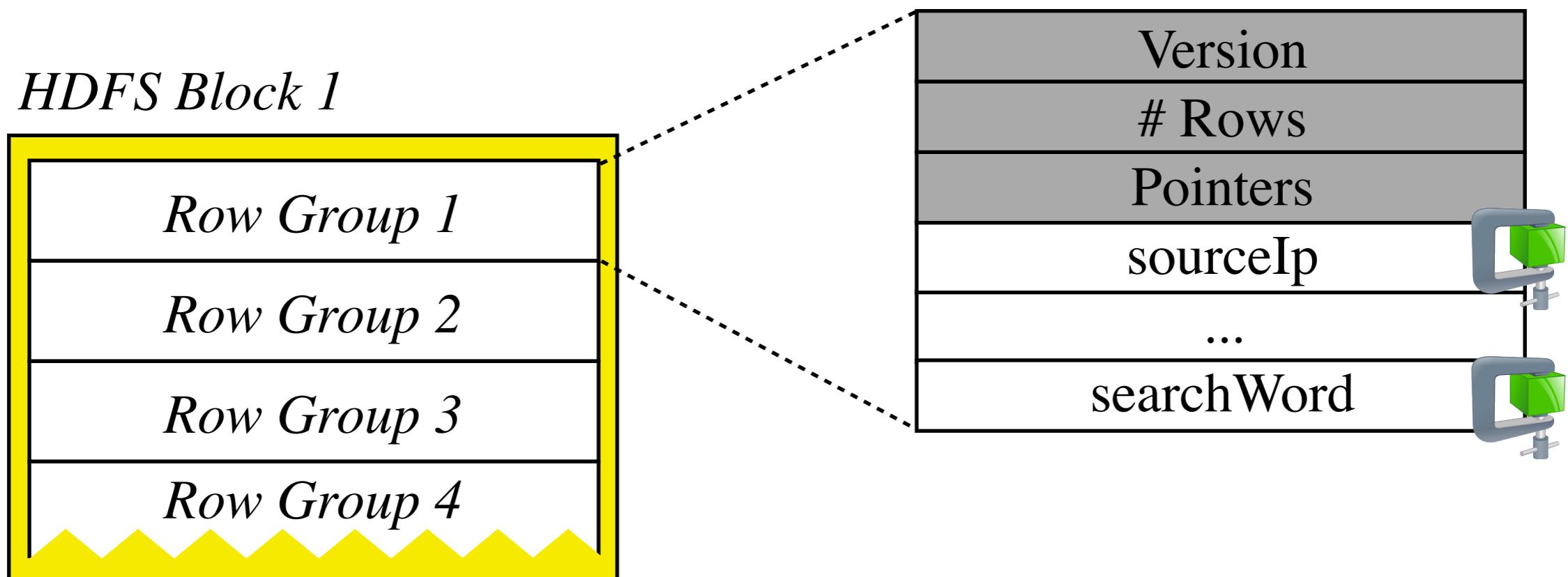
HDFS Block Format

Average Record Size: 100 bytes

#Total Records = 1,000,000

Row Group Size = 200,000 records

HDFS Block Size = 64MB



+ Columns in Row Groups are compressed

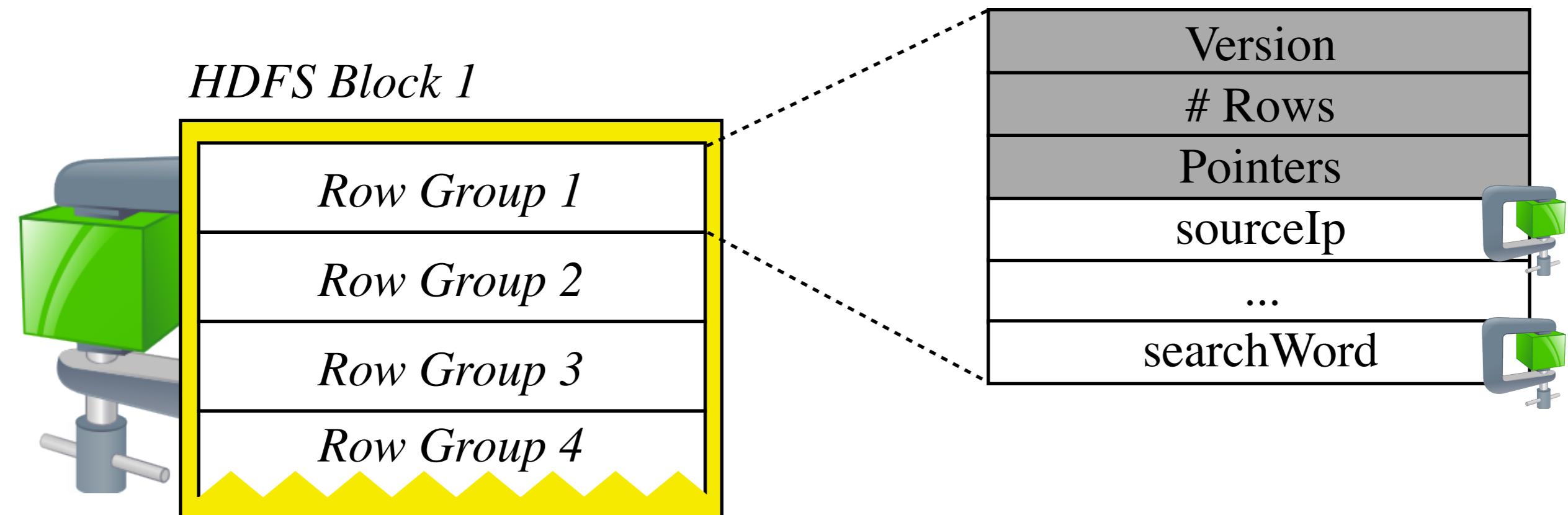
HDFS Block Format

Average Record Size: 100 bytes

#Total Records = 1,000,000

Row Group Size = 200,000 records

HDFS Block Size = 64MB

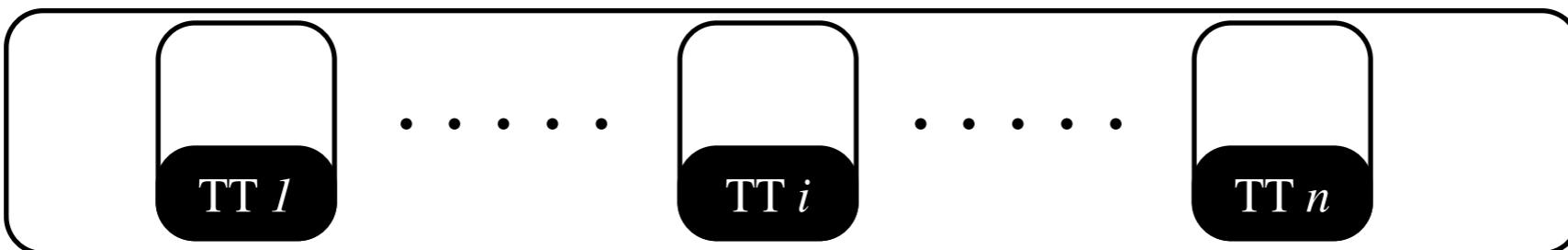


- + Columns in Row Groups are compressed
- + Further compression at the HDFS block level

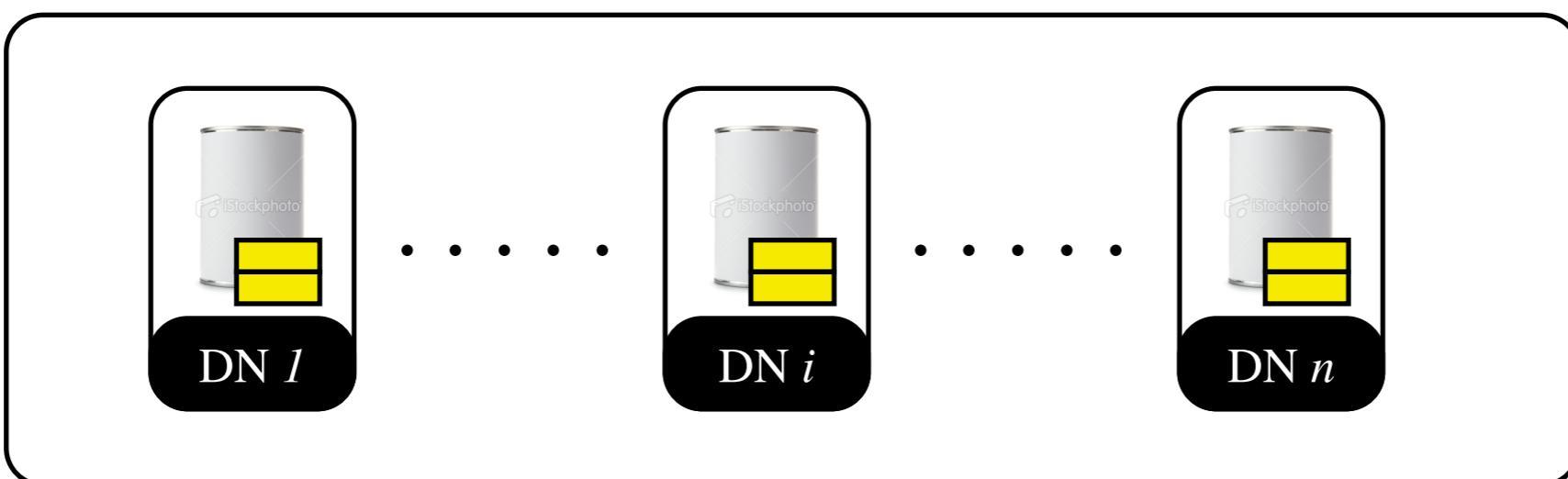
Job Execution



MapReduce



HDFS



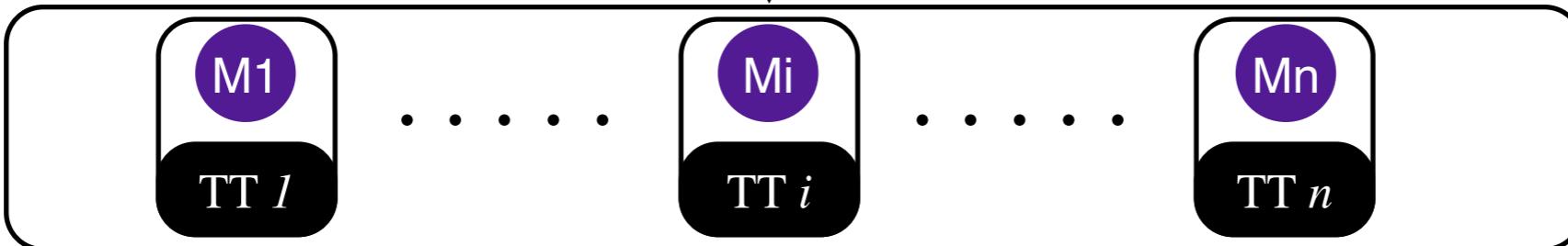
Job Execution

MapReduce

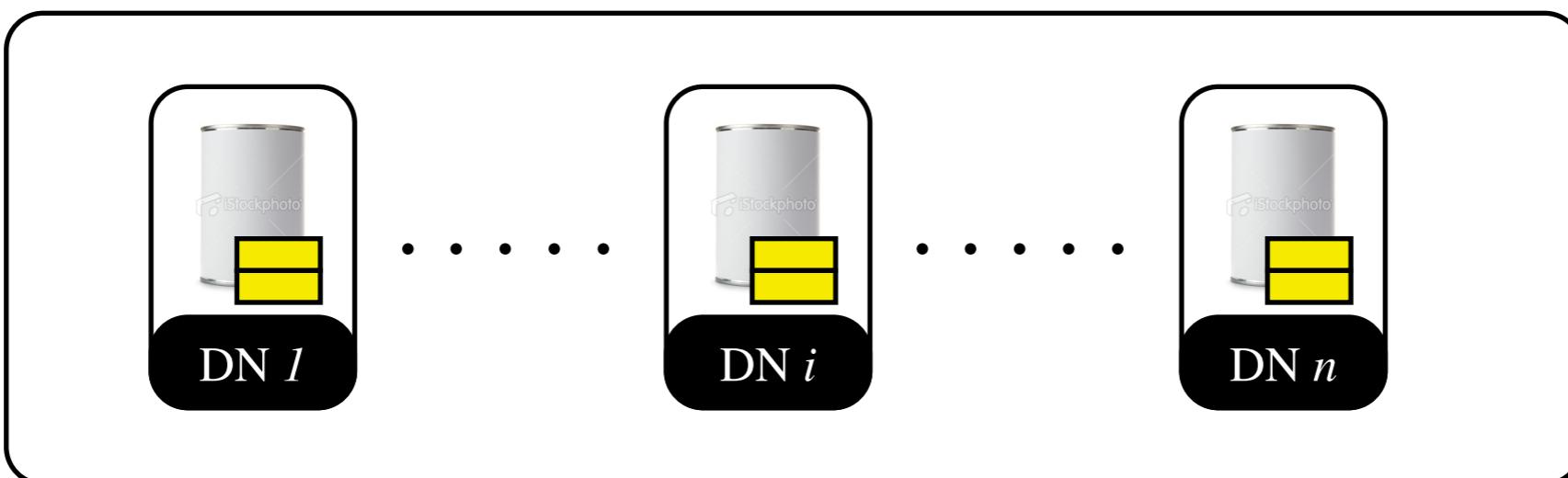


run job

```
map (offset, tuple) {  
    if (sourceIP == 120.115.124.34)  
        output(sourceIP, pageURL)  
}
```

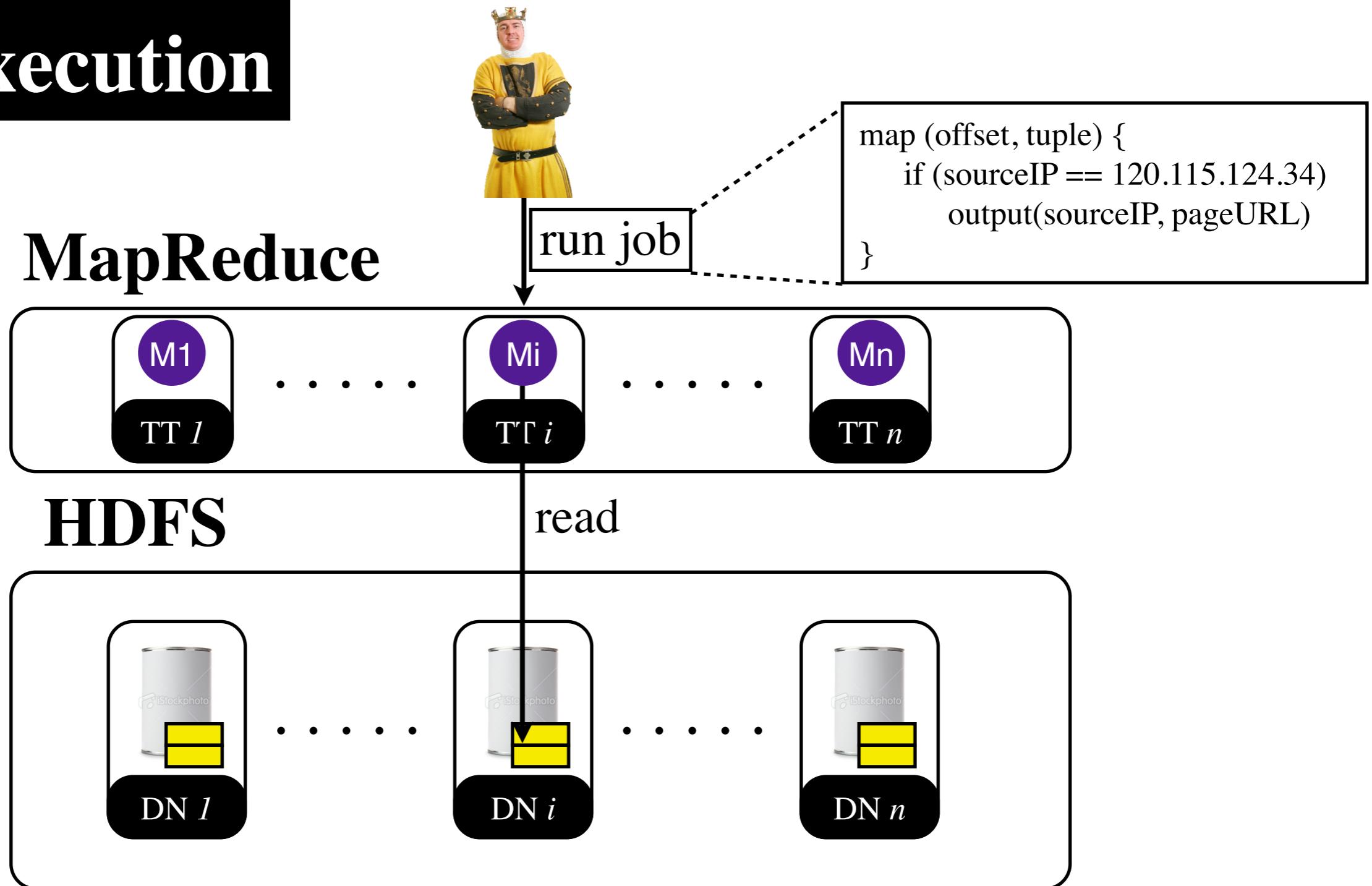


HDFS

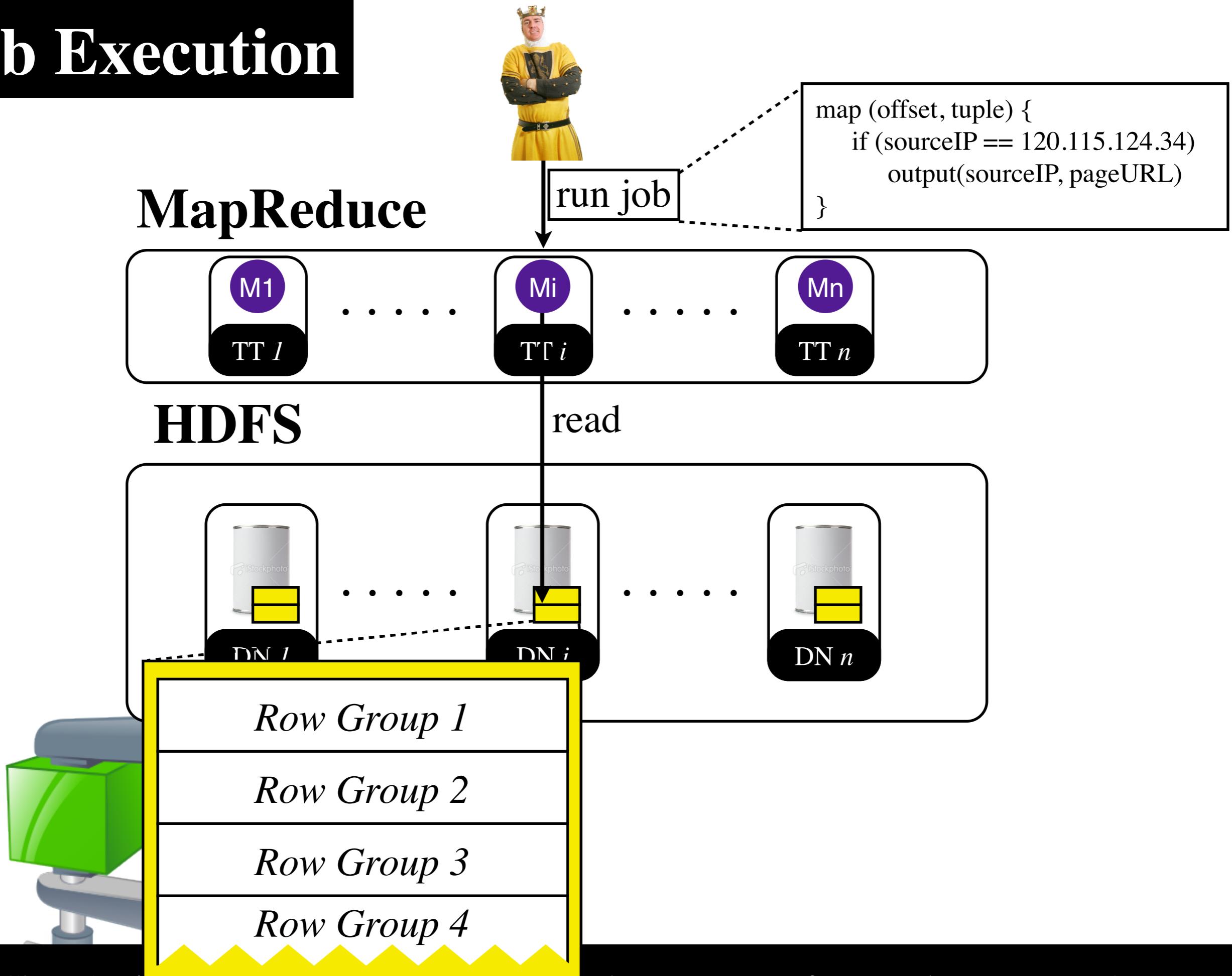


Job Execution

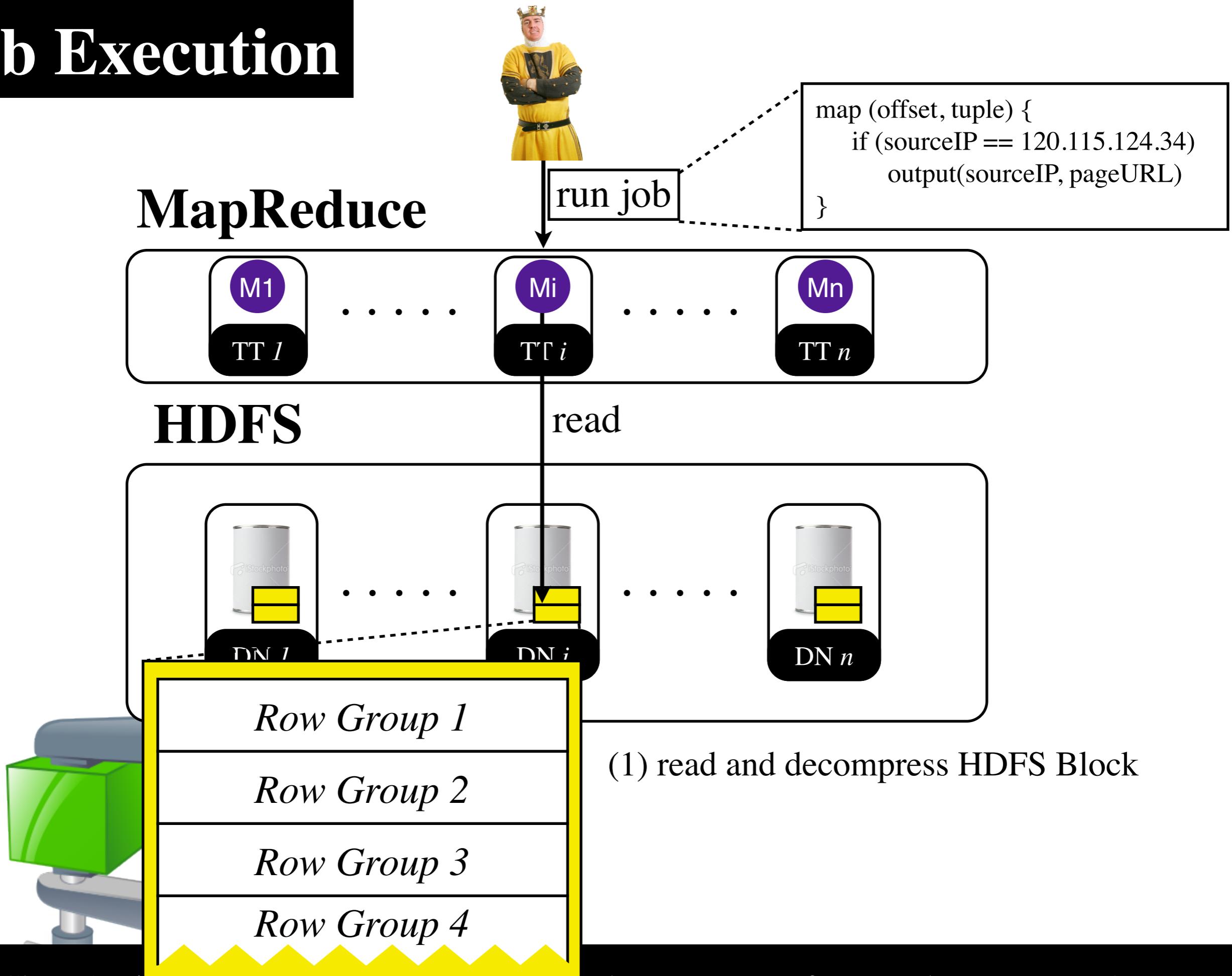
MapReduce



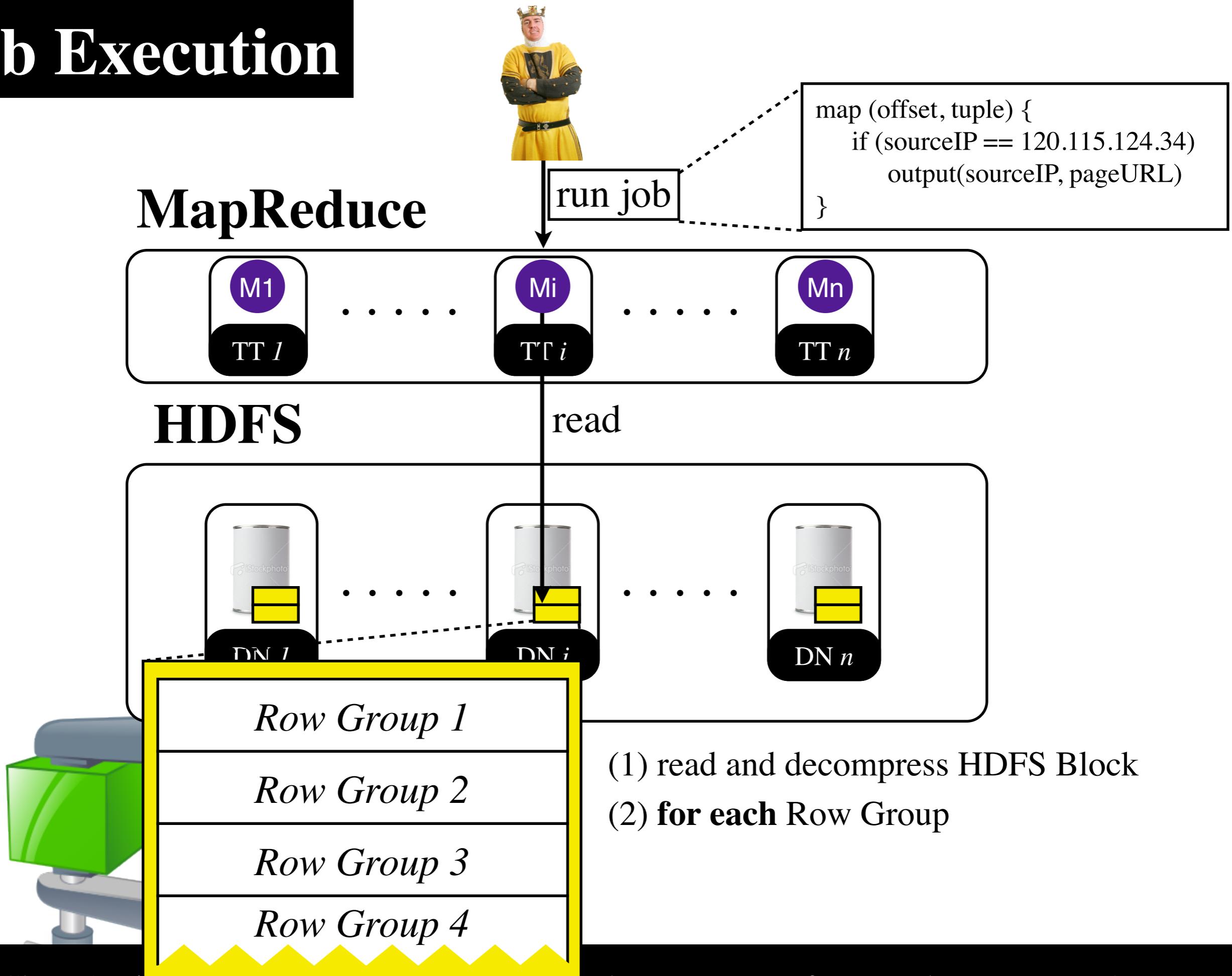
Job Execution



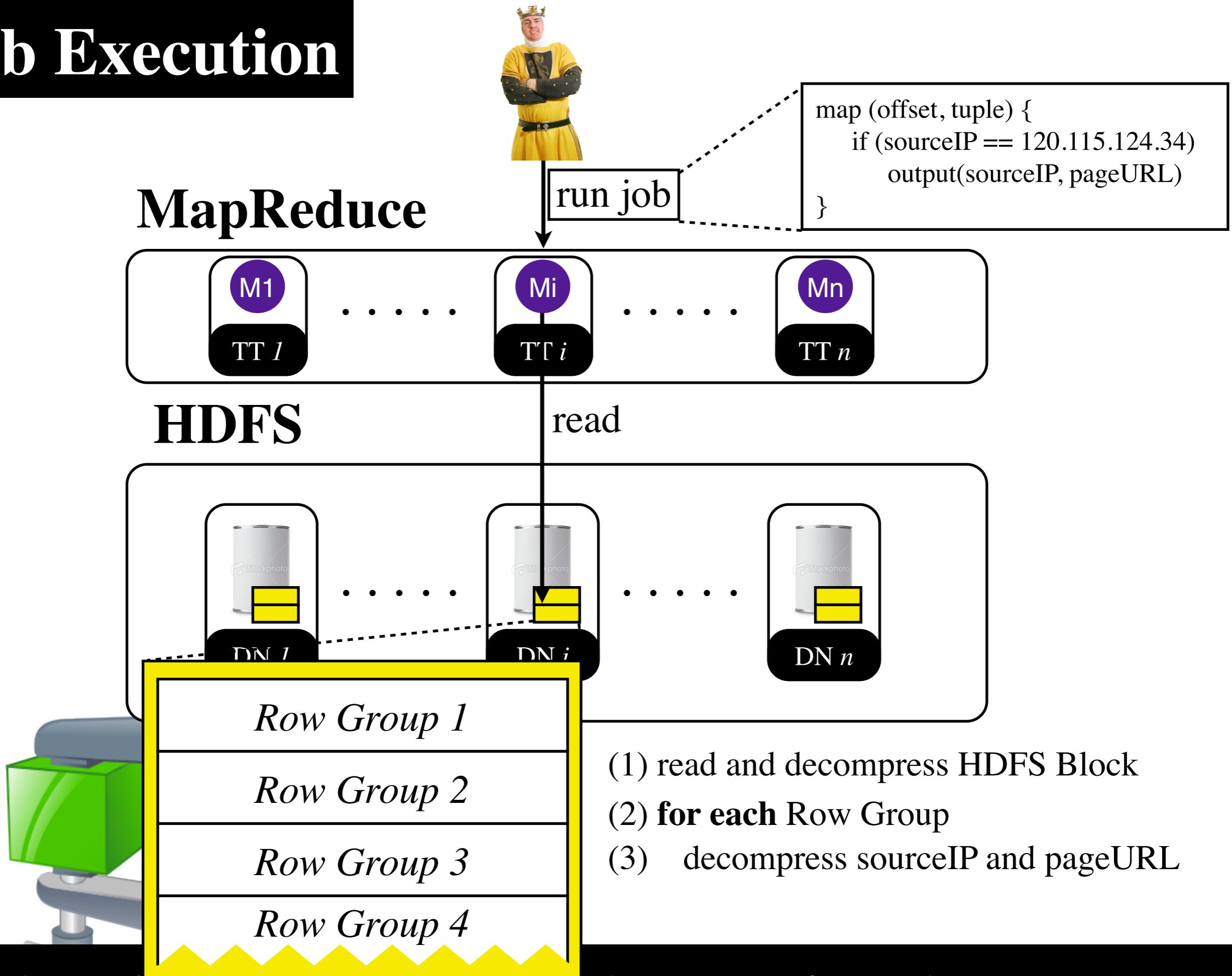
Job Execution



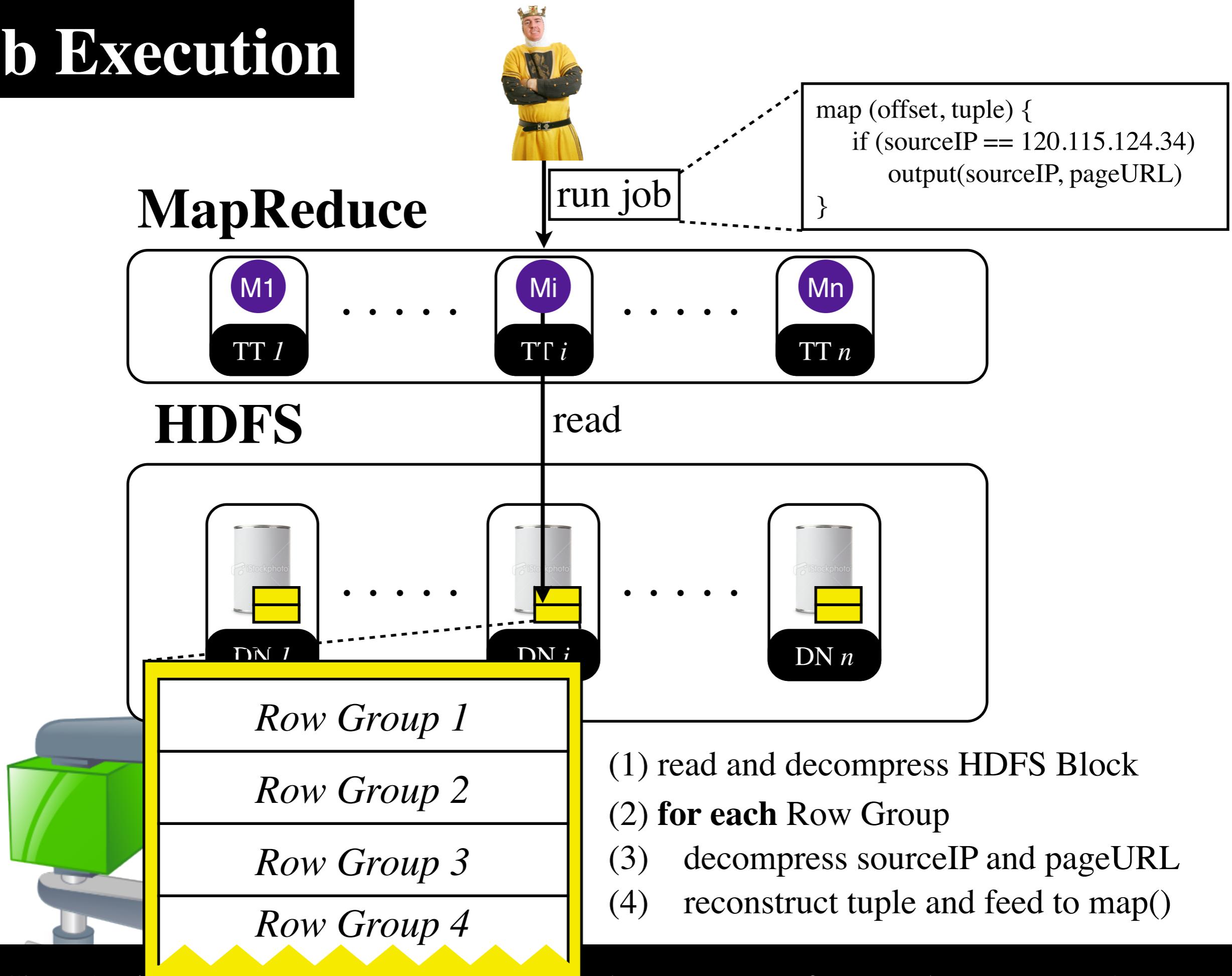
Job Execution



Job Execution



Job Execution



Data Layouts in MapReduce

Initial	2009
Row	CFile
Read Unnecessary columns	
	Tuple Reconstruction
	High network costs

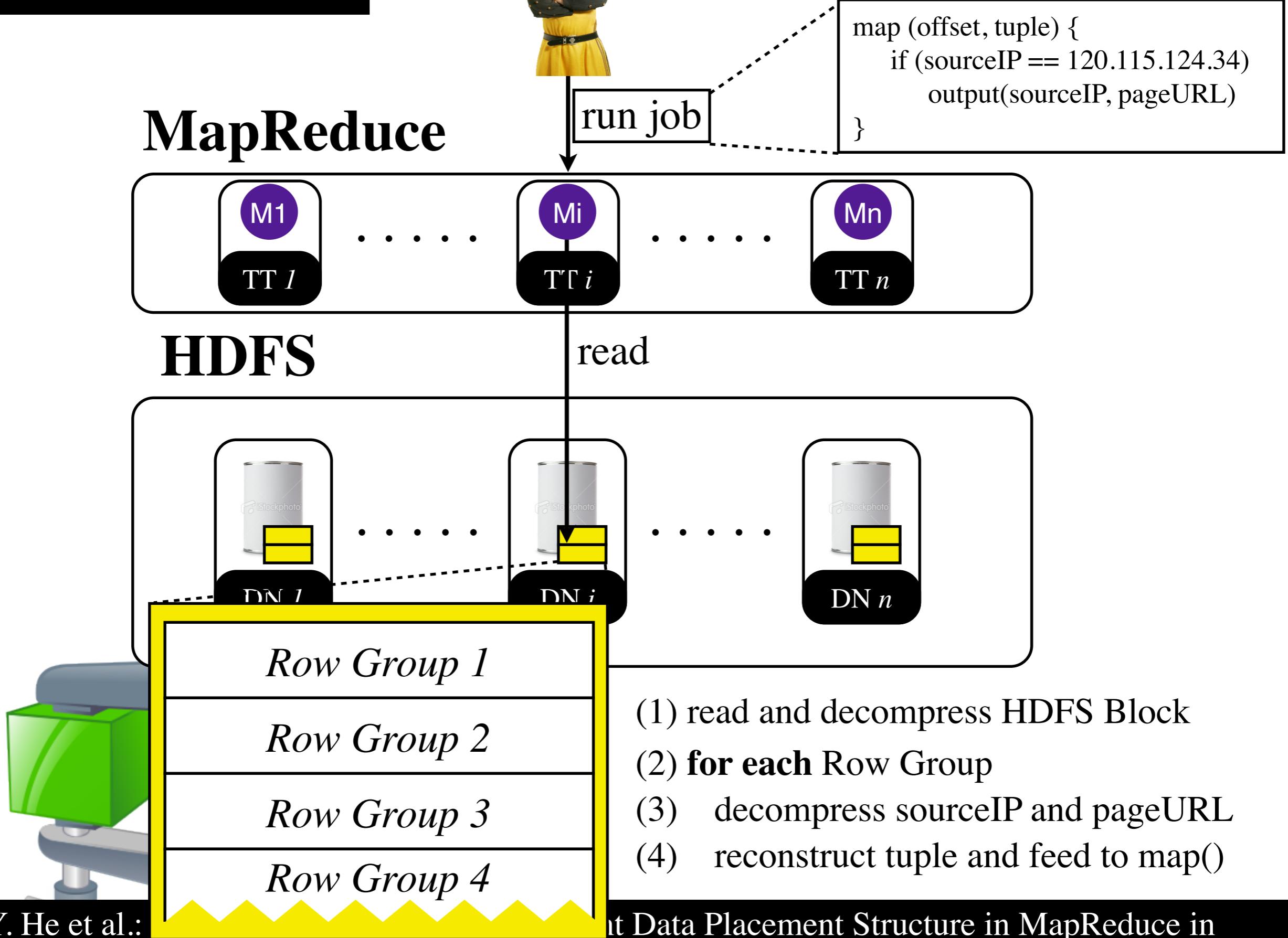
Data Layouts in MapReduce

Initial	2009	2010
Row	CFile	Cheetah
Read Unnecessary columns		
	Tuple Reconstruction	Tuple Reconstruction
	High network costs	
		Block level compression
		Poor I/O Saving

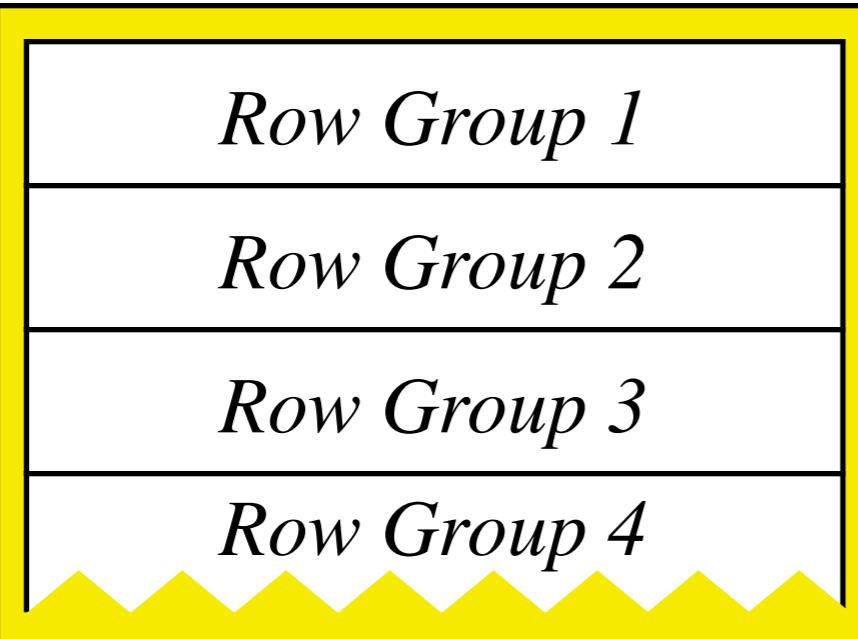
Row Columnar File (RCFile)

[Y. He et al.: RCFile: A Fast and Space-Efficient Data Placement Structure in MapReduce in MapReduce-based Warehouse Systems. ICDE 2011]

Job Execution



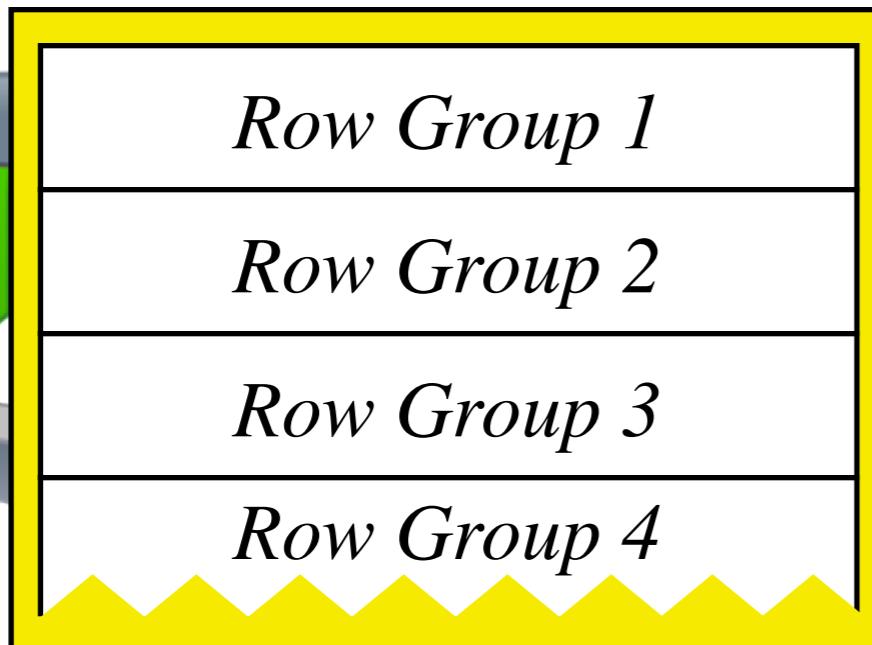
RecordReader



Reading HDFS Blocks with Cheetah

- (1) read and decompress HDFS Block
- (2) **for each** Row Group
- (3) decompress sourceIP and pageURL
- (4) reconstruct tuple and feed to map()

RecordReader

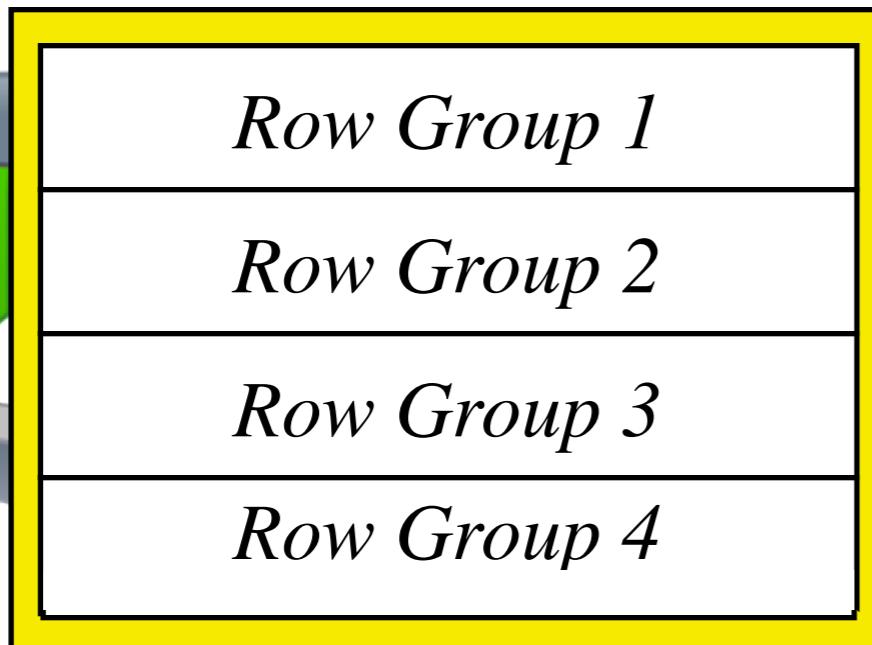


Reading HDFS Blocks with Cheetah

- (1) read and decompress HDFS Block
- (2) **for each** Row Group
- (3) decompress sourceIP and pageURL
- (4) reconstruct tuple and feed to map()

Reading HDFS Blocks with RCFile

RecordReader

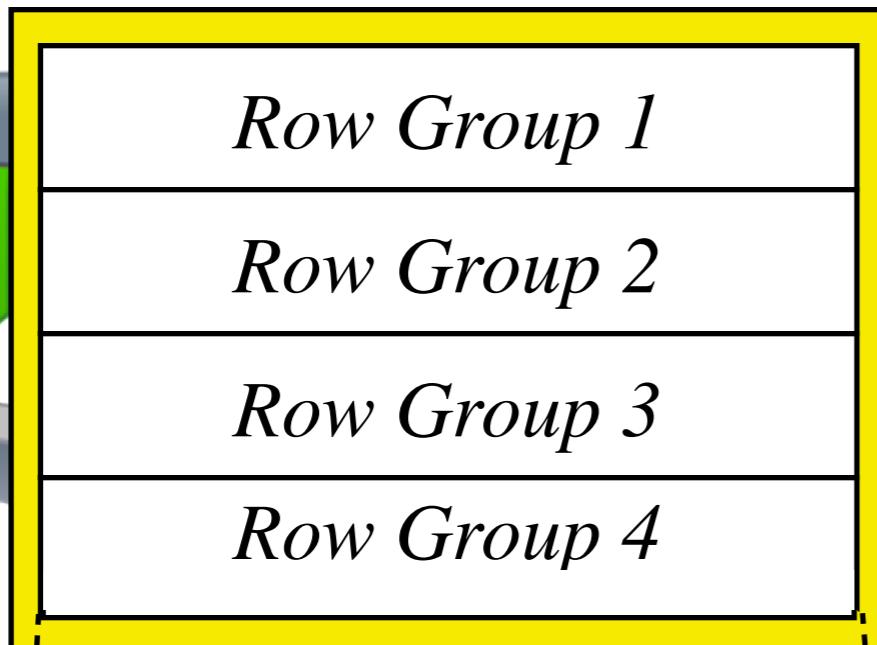


Reading HDFS Blocks with Cheetah

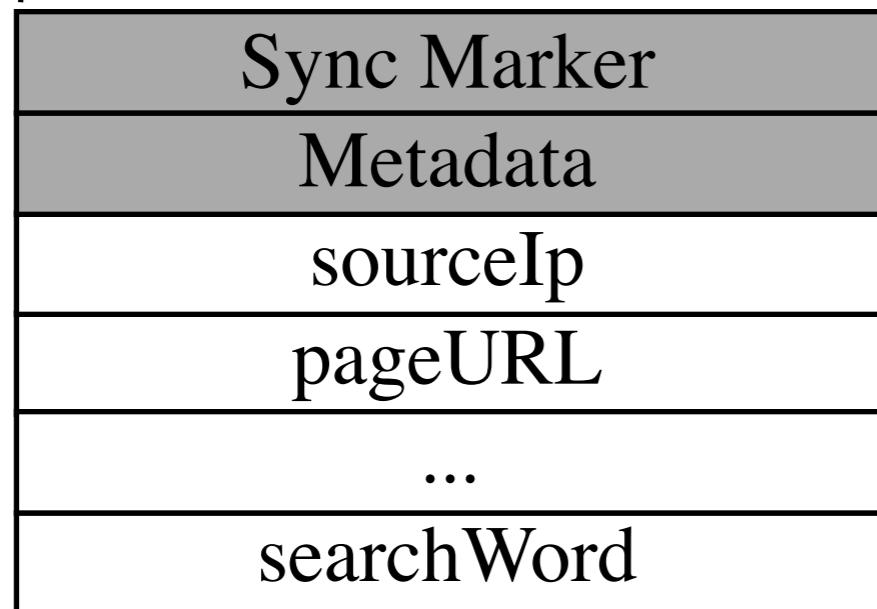
- (1) read and decompress HDFS Block
- (2) **for each** Row Group
- (3) decompress sourceIP and pageURL
- (4) reconstruct tuple and feed to map()

Reading HDFS Blocks with RCFile

RecordReader



RCFile Format

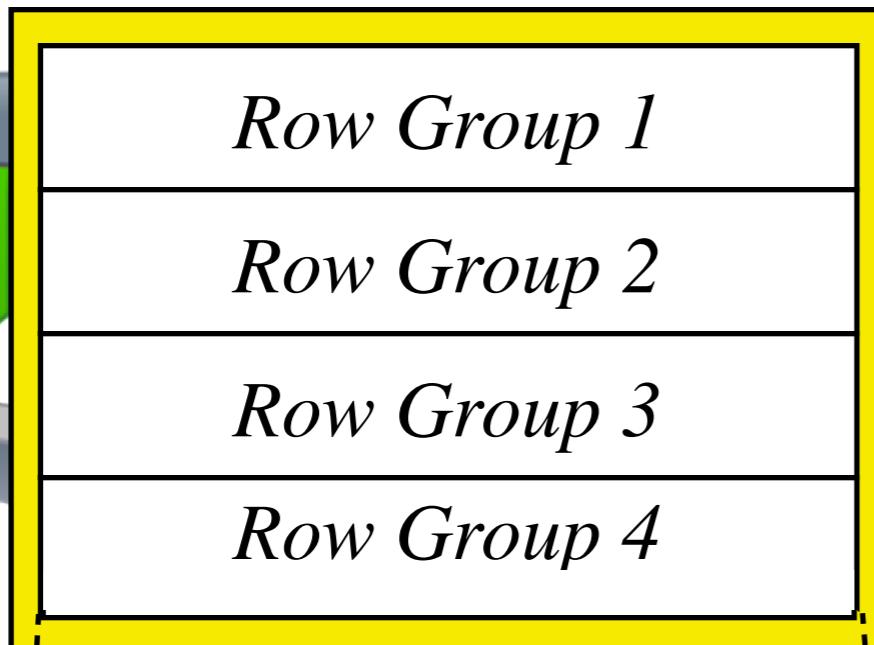


Reading HDFS Blocks with Cheetah

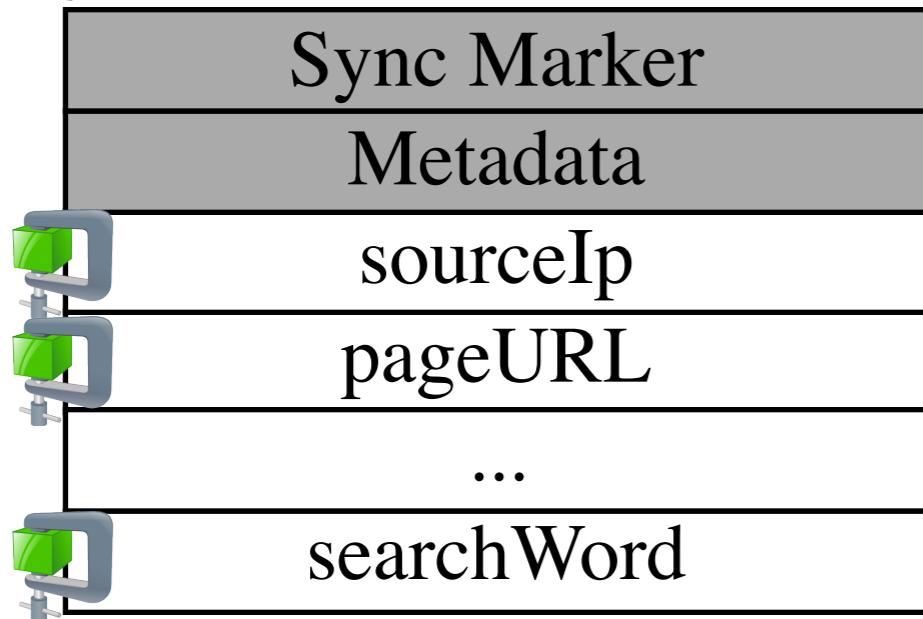
- (1) read and decompress HDFS Block
- (2) **for each** Row Group
- (3) decompress sourceIP and pageURL
- (4) reconstruct tuple and feed to map()

Reading HDFS Blocks with RCFile

RecordReader



RCFile Format

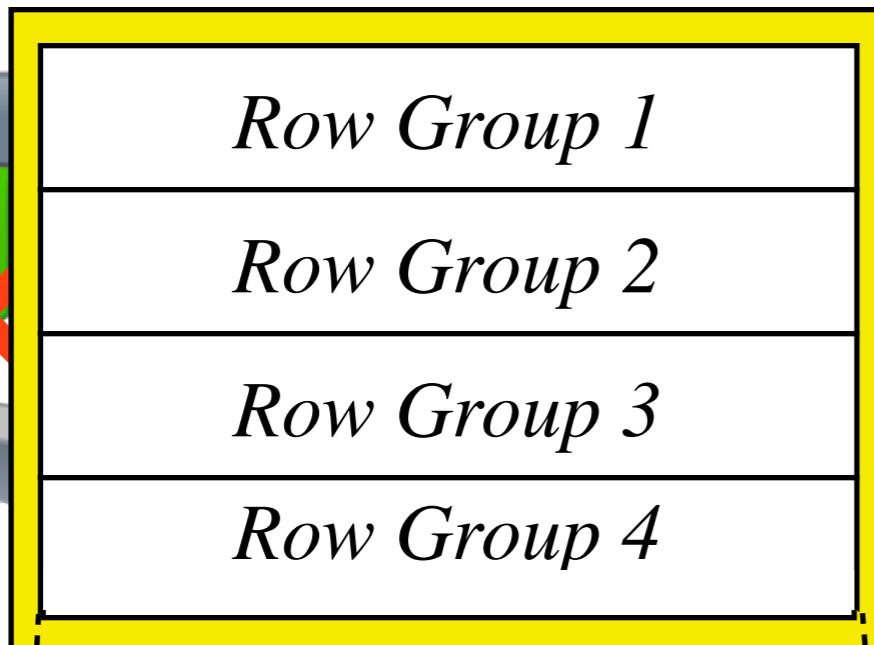


Reading HDFS Blocks with Cheetah

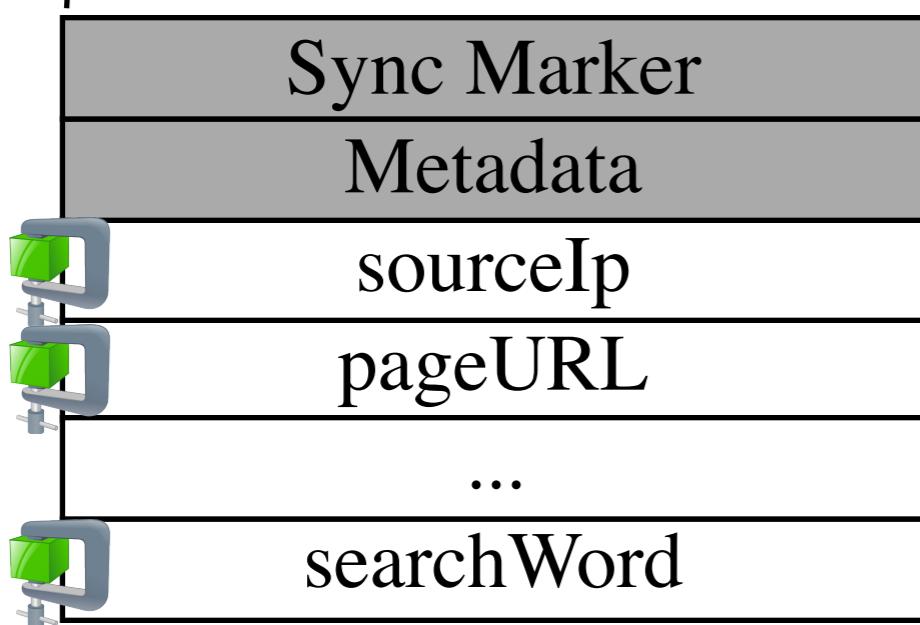
- (1) read and decompress HDFS Block
- (2) **for each** Row Group
- (3) decompress sourceIP and pageURL
- (4) reconstruct tuple and feed to map()

Reading HDFS Blocks with RCFile

RecordReader



RCFile Format



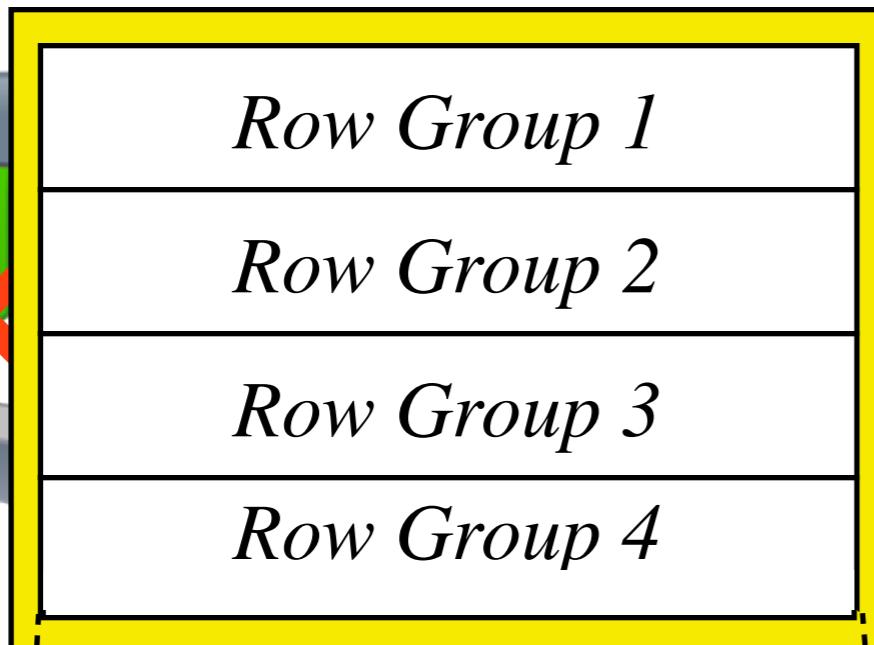
Reading HDFS Blocks with Cheetah

- (1) ~~read and decompress HDFS Block~~
- (2) **for each** Row Group
- (3) decompress sourceIP and pageURL
- (4) reconstruct tuple and feed to map()

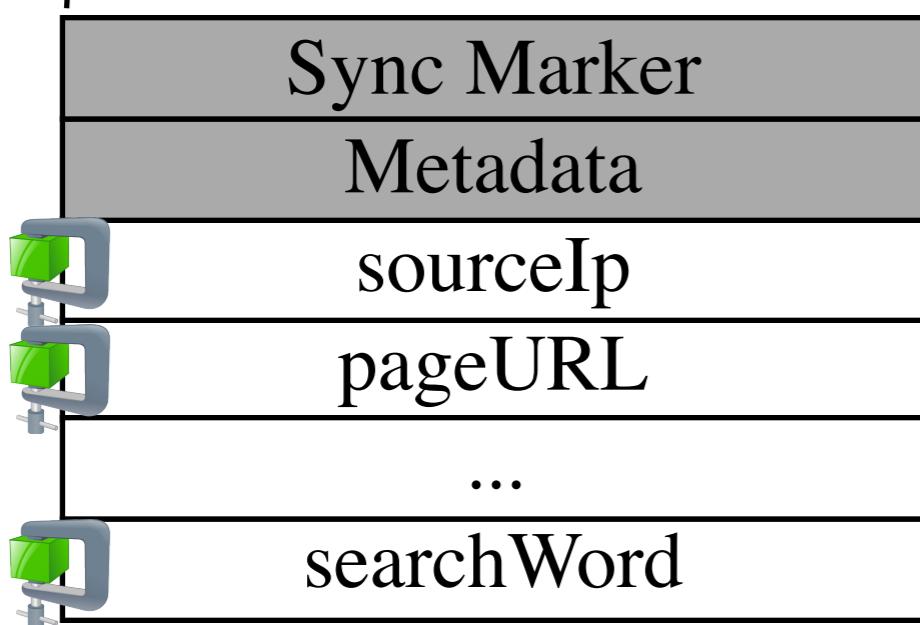
Reading HDFS Blocks with RCFile

- (1) **for each** Row Group
- (2) read sourceIP and pageURL

RecordReader



RCFile Format



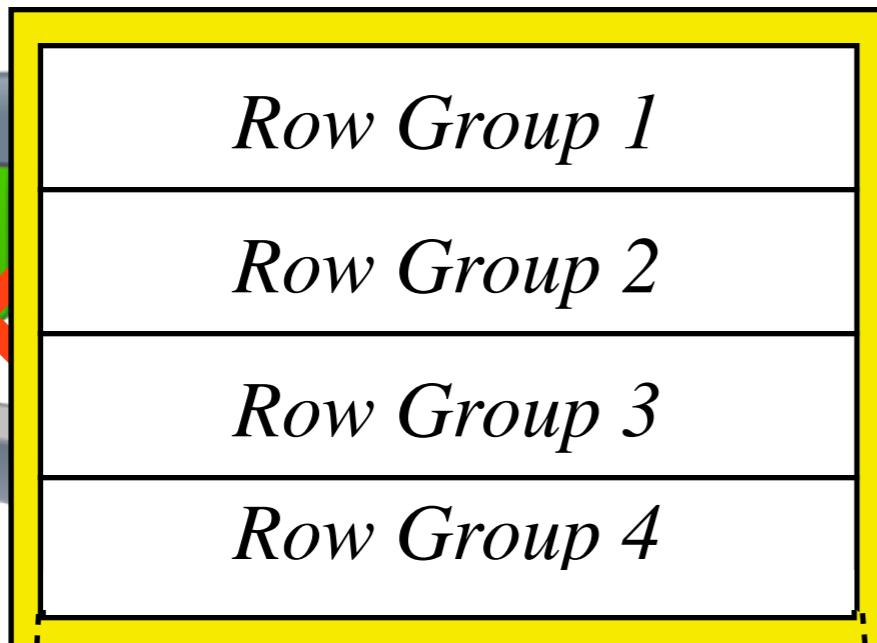
Reading HDFS Blocks with Cheetah

- (1) ~~read and decompress HDFS Block~~
- (2) **for each** Row Group
- (3) ~~decompress sourceIP and pageURL~~
- (4) reconstruct tuple and feed to map()

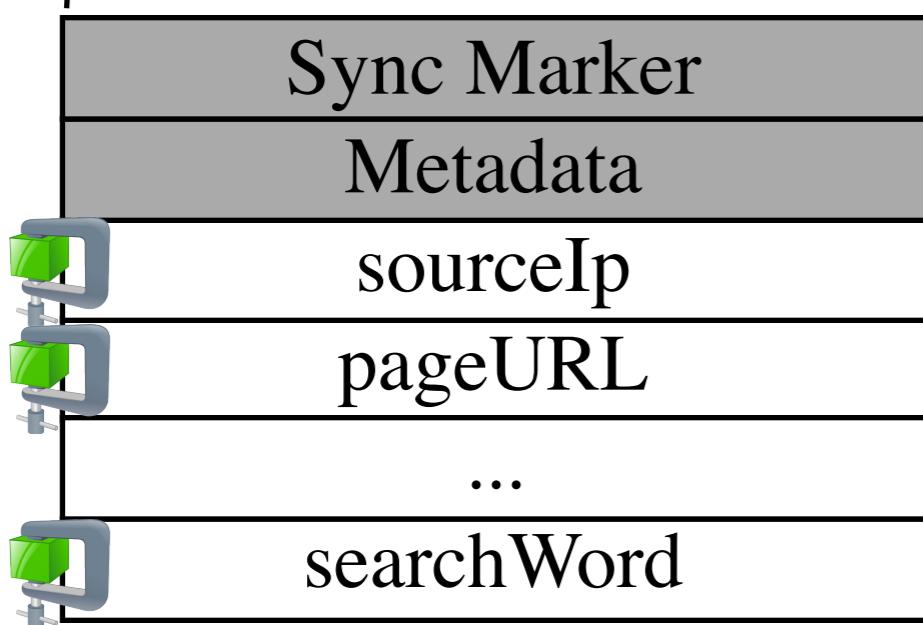
Reading HDFS Blocks with RCFile

- (1) **for each** Row Group
- (2) read sourceIP and pageURL
- (3) decompress sourceIP

RecordReader



RCFile Format



Reading HDFS Blocks with Cheetah

- (1) ~~read and decompress HDFS Block~~
- (2) **for each** Row Group
- (3) ~~decompress sourceIP and pageURL~~
- (4) reconstruct tuple and feed to map()

Reading HDFS Blocks with RCFile

- (1) **for each** Row Group
- (2) read sourceIP and pageURL
- (3) decompress sourceIP
- (4) **for each** sourceIP value
- (5) **if** sourceIP == 120.115.124.34

RecordReader



Row Group 1

Row Group 2

Row Group 3

Row Group 4

RCFile Format



Sync Marker

Metadata

sourceIp

pageURL

...

searchWord

Reading HDFS Blocks with Cheetah

- (1) ~~read and decompress HDFS Block~~
- (2) **for each** Row Group
- (3) ~~decompress sourceIP and pageURL~~
- (4) reconstruct tuple and feed to map()

Reading HDFS Blocks with RCFile

- (1) **for each** Row Group
- (2) read sourceIP and pageURL
- (3) decompress sourceIP
- (4) **for each** sourceIP value
- (5) **if** sourceIP == 120.115.124.34
- (6) decompress pageURL
- (7) reconstruct tuple and feed to map()

Data Layouts in MapReduce

Initial	2009	2010
Row	CFile	Cheetah
Read Unnecessary columns		
	Tuple Reconstruction	Tuple Reconstruction
	High network costs	
		Block level compression
		Poor I/O Saving

Data Layouts in MapReduce

Initial	2009	2010	2011
Row	CFile	Cheetah	RCFile
Read Unnecessary columns			
	Tuple Reconstruction	Tuple Reconstruction	Tuple Reconstruction
	High network costs		
		Block level compression	
		Poor I/O Saving	Poor I/O Saving

Column Input Format (CIF)

[A. Floratou et al.: Column-Oriented Storage Techniques for MapReduce.
PVLDB 2011]

Remarks on Cheetah-Storage and RCFile

Remarks on Cheetah-Storage and RCFile

(1) I/O elimination becomes difficult

Remarks on Cheetah-Storage and RCFile

- (1) I/O elimination becomes difficult
- (2) Tuning the row-group size becomes critical

Remarks on Cheetah-Storage and RCFile

- (1) I/O elimination becomes difficult
- (2) Tuning the row-group size becomes critical
- (3) Overhead for per-Row Group metadata

Remarks on Cheetah-Storage and RCFile

- (1) I/O elimination becomes difficult
- (2) Tuning the row-group size becomes critical
- (3) Overhead for per-Row Group metadata

CIF Approach:

CFile + Cheetah Storage (or RCFile)

Data Upload --- Upload UserVisits ---

UserVisits Log

125.102.135.45, espn.com, 2011/12/01, 123.35, football

101.132.121.13, cnn.com, 2011/12/02, 365.98, crisis

120.115.124.34, vldb.org, 2011/12/03, 296.02, database

⋮

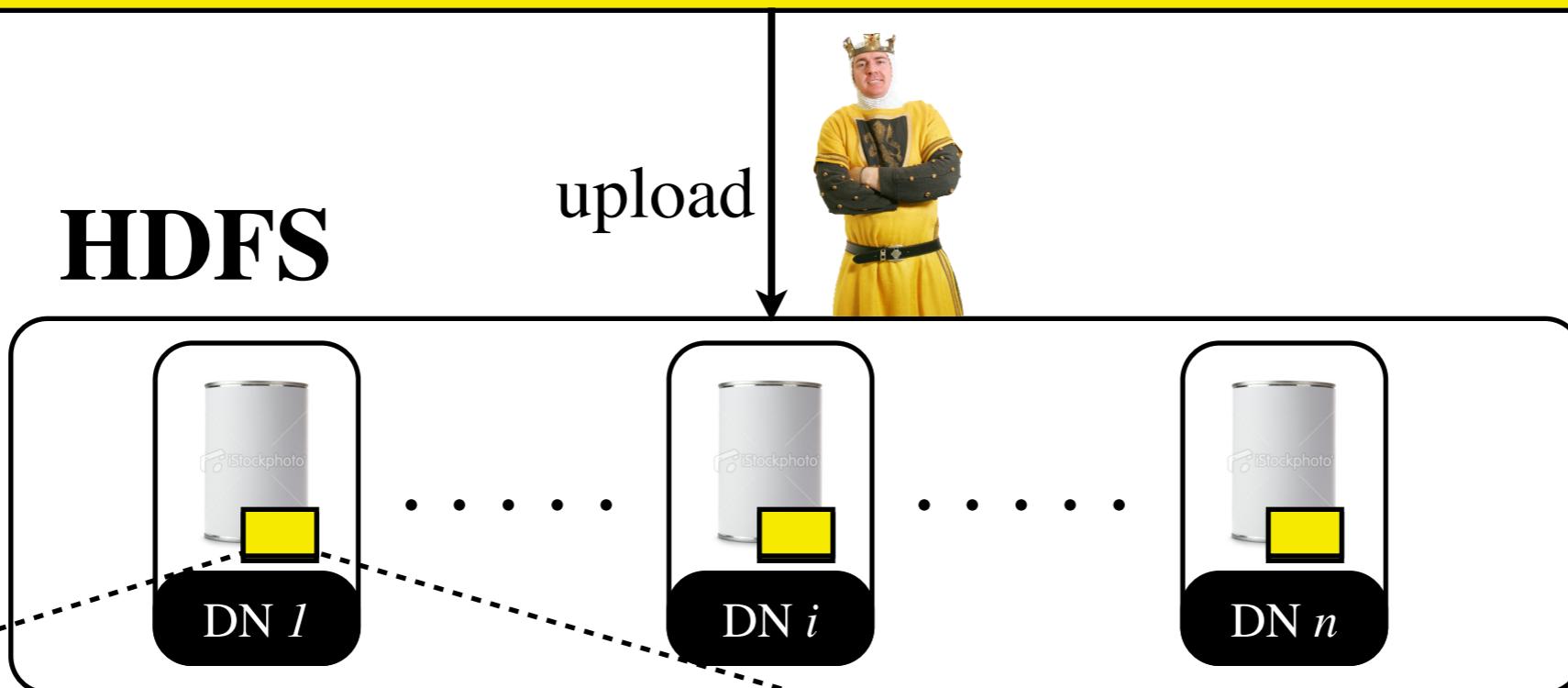
HDFS



Data Upload --- Upload UserVisits ---

UserVisits Log

```
125.102.135.45, espn.com, 2011/12/01, 123.35, football  
101.132.121.13, cnn.com, 2011/12/02, 365.98, crisis  
120.115.124.34, vldb.org, 2011/12/03, 296.02, database  
:  
:
```



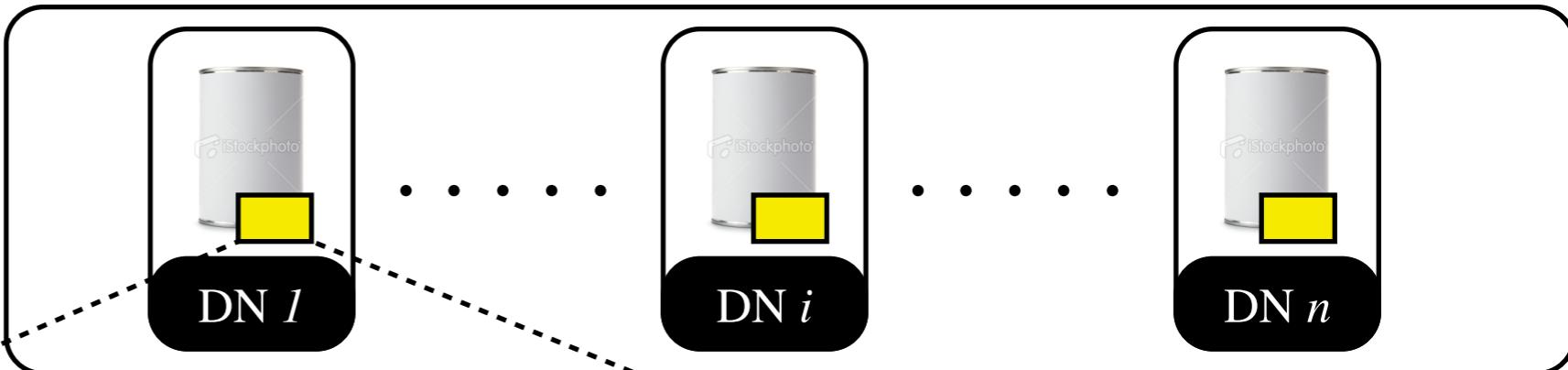
```
125.102.135.45, espn.com, 2011/12/01, 123.35, football  
101.132.121.13, cnn.com, 2011/12/02, 365.98, crisis  
120.115.124.34, vldb.org, 2011/12/03, 296.02, database  
:  
:
```

Dataset uploaded at
`hdfs://MyData/UserVisits/`

Data Upload --- Run Parallel Loader ---

HDFS

Dataset uploaded at
hdfs://MyData/UserVisits/



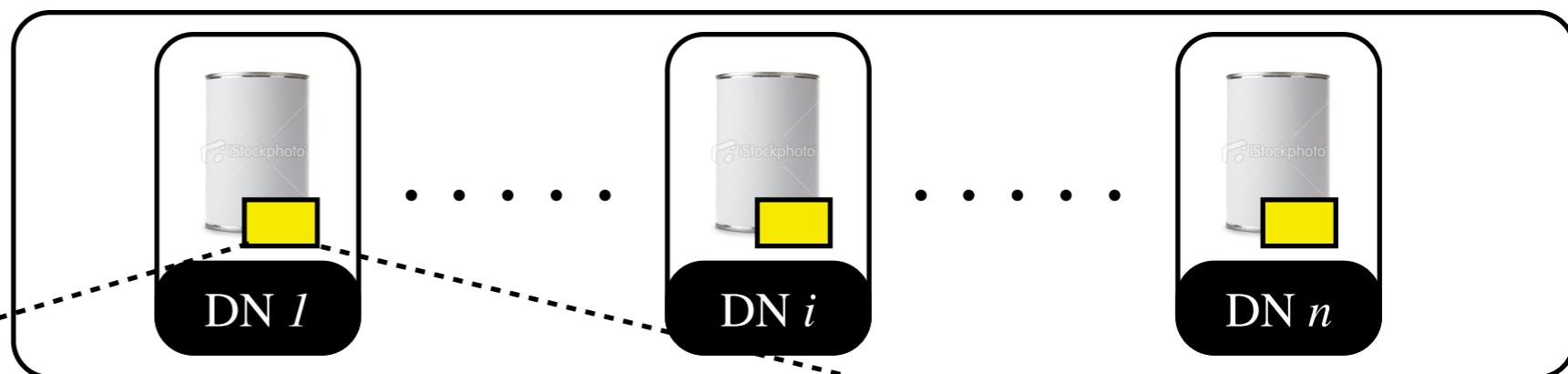
125.102.135.45, espn.com, 2011/12/01, 123.35, football
101.132.121.13, cnn.com, 2011/12/02, 365.98, crisis
120.115.124.34, vldb.org, 2011/12/03, 296.02, database
:
:

Data Upload --- Run Parallel Loader ---

HDFS

Dataset uploaded at

hdfs://MyData/UserVisits/



125.102.135.45, espn.com, 2011/12/01, 123.35, football

101.132.121.13, cnn.com, 2011/12/02, 365.98, crisis

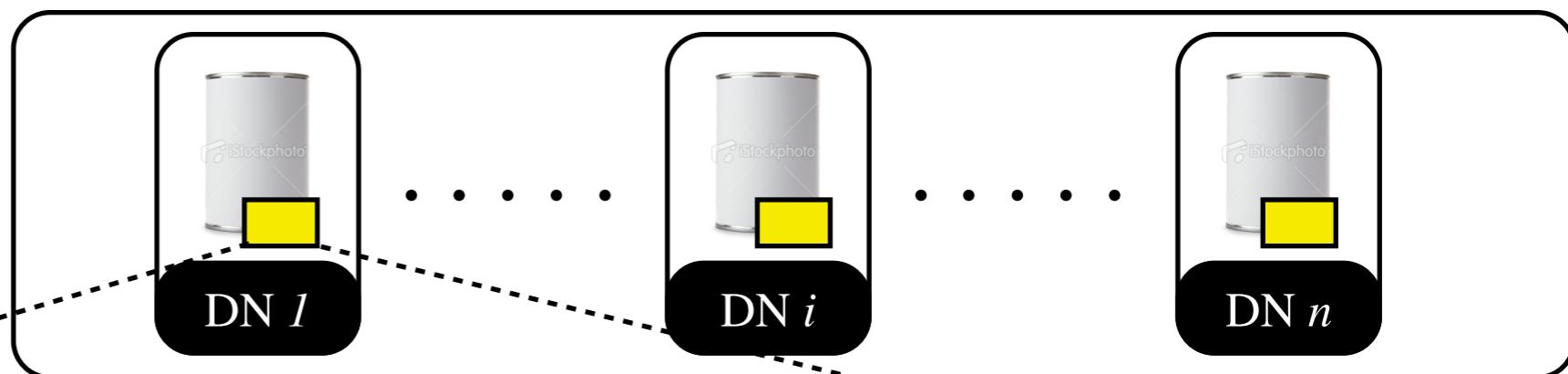
120.115.124.34, vldb.org, 2011/12/03, 296.02, database

•
•
•

Data Upload --- Run Parallel Loader ---

HDFS

Dataset uploaded at
hdfs://MyData/UserVisits/



125.102.135.45, espn.com, 2011/12/01, 123.35, football

101.132.121.13, cnn.com, 2011/12/02, 365.98, crisis

120.115.124.34, vldb.org, 2011/12/03, 296.02, database

•

•

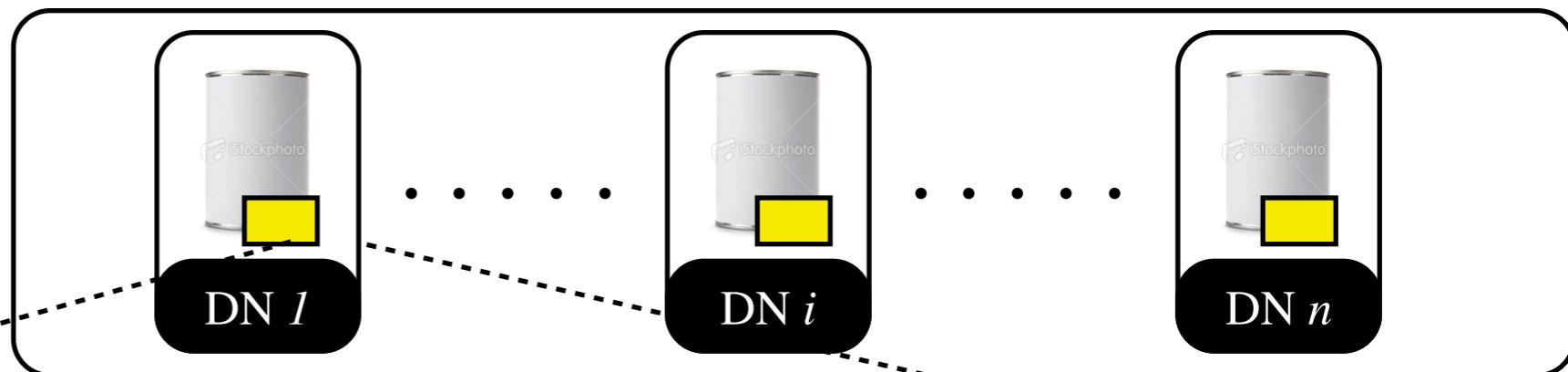
Row Group
“*split0*”

Data Upload --- Run Parallel Loader ---

HDFS

Dataset uploaded at

hdfs://MyData/UserVisits/



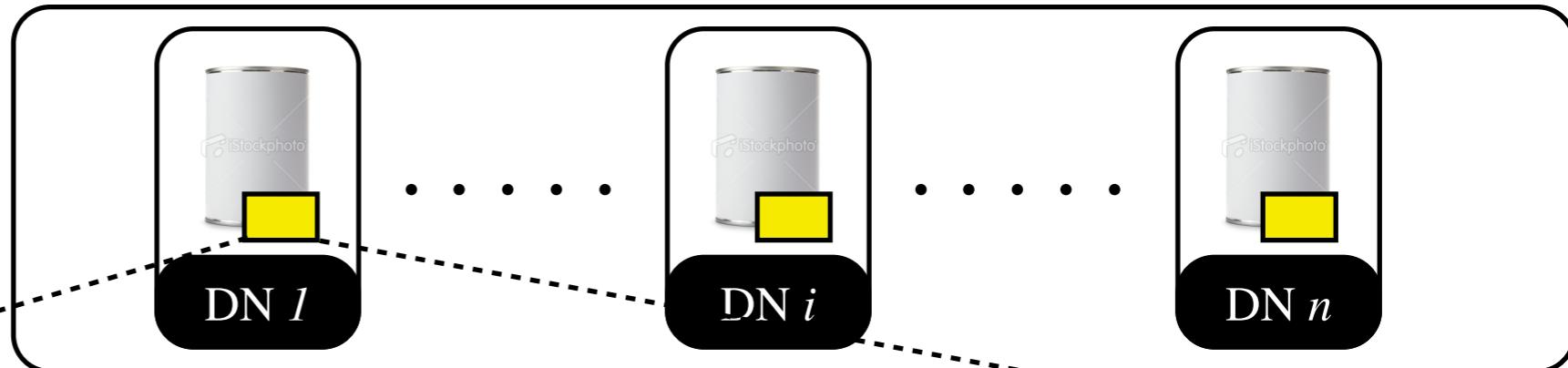
125.102.135.45	espn.com	2011/12/01	123.35	football
101.132.121.13	cnn.com	2011/12/02	365.98	crisis
120.115.124.34	vldb.org	2011/12/03	296.02	database
.
:	:	:	:	:
:	:	:	:	:

Row Group
“*split0*”

Data Upload --- Run Parallel Loader ---

HDFS

Dataset uploaded at
hdfs://MyData/UserVisits/



Row Group
“*split0*”

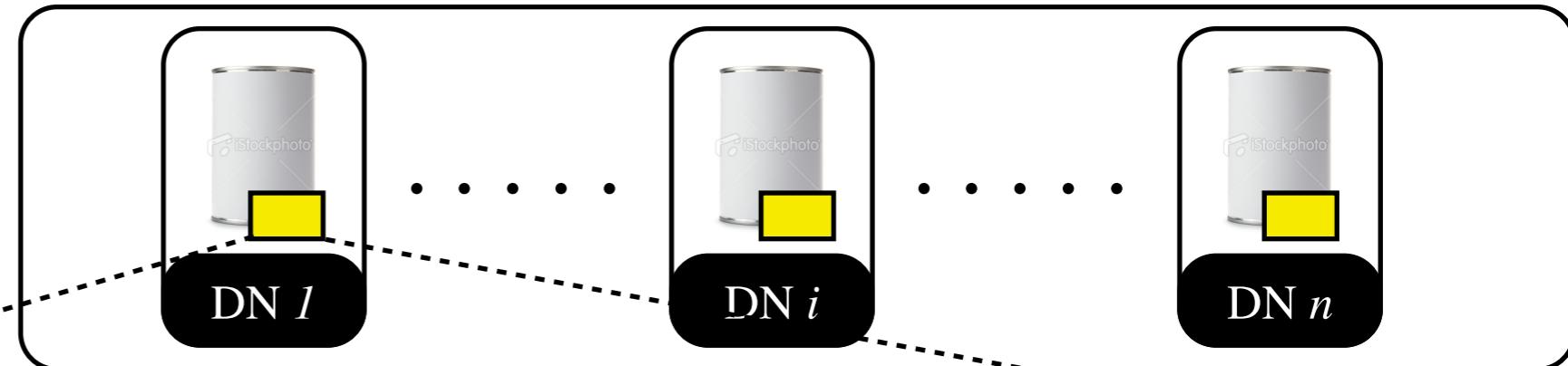
125.102.135.45	espn.com	2011/12/01	123.35	football
101.132.121.13	cnn.com	2011/12/02	365.98	crisis
120.115.124.34	vldb.org	2011/12/03	296.02	database
⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮

Data Upload --- Run Parallel Loader ---

HDFS

Dataset uploaded at

hdfs://MyData/UserVisits/



125.102.135.45
101.132.121.13
120.115.124.34
...

espn.com
cnn.com
vldb.org
...

2011/12/01
2011/12/02
2011/12/03
...

123.35
365.98
296.02
...

football
crisis
database
...

Row Group
“*split0*”

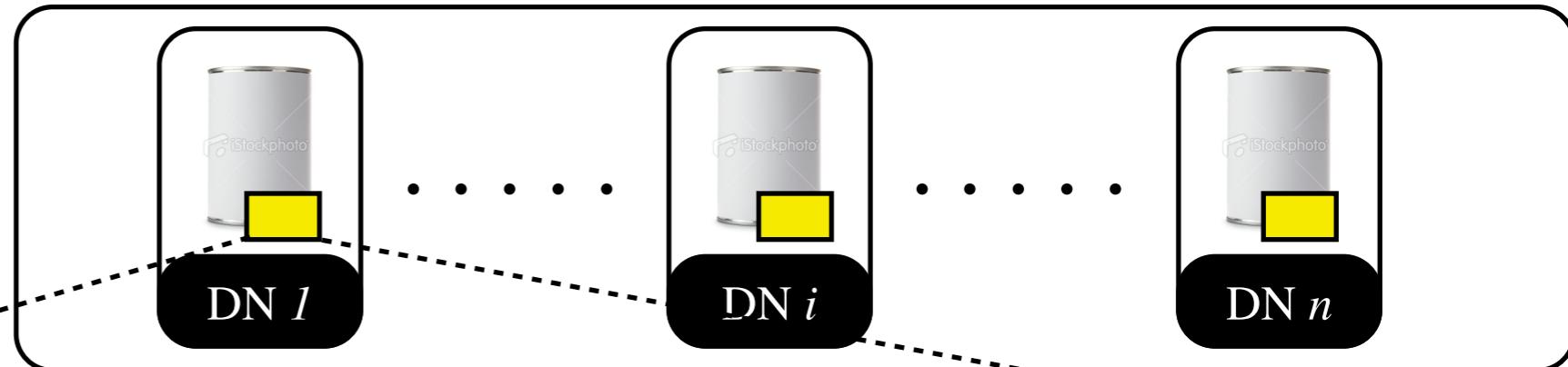
hdfs://MyData/UserVisits/split0/sourceIP
hdfs://...../pageURL ←
hdfs://...../visitDate ←
hdfs://...../adRevenue ←
hdfs://...../searchWord ←

Data Upload --- Run Parallel Loader ---

HDFS

Dataset uploaded at

hdfs://MyData/UserVisits/



Row Group
“*split0*”

125.102.135.45	espn.com	2011/12/01	123.35	football
101.132.121.13	cnn.com	2011/12/02	365.98	crisis
120.115.124.34	vldb.org	2011/12/03	296.02	database
⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮

hdfs://MyData/UserVisits/split0/sourceIP
hdfs://...../pageURL ←
hdfs://...../visitDate ←
hdfs://...../adRevenue ←
hdfs://...../searchWord ←
hdfs://...../MetadataFile

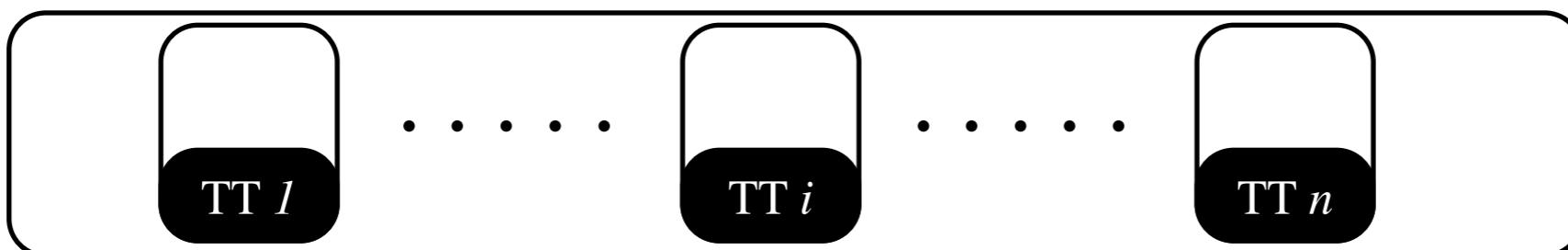
Job Execution

sourceIp
pageURL
visitDate

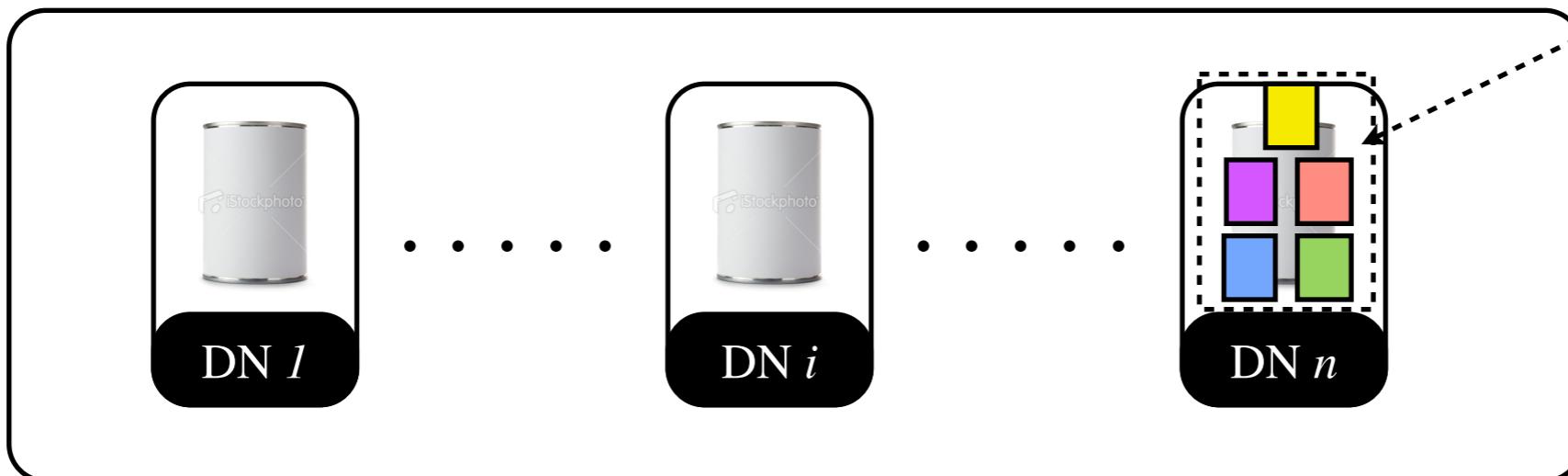
adRevenue
searchWord



MapReduce



HDFS



Job Execution

sourceIp
pageURL
visitDate

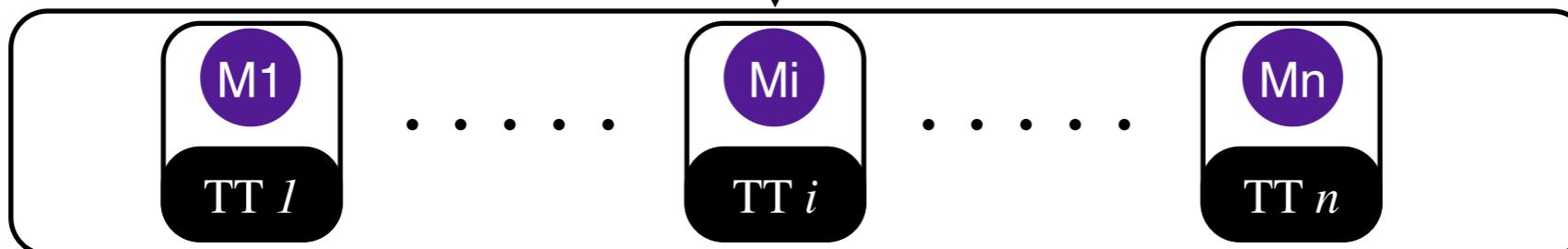
adRevenue
searchWord



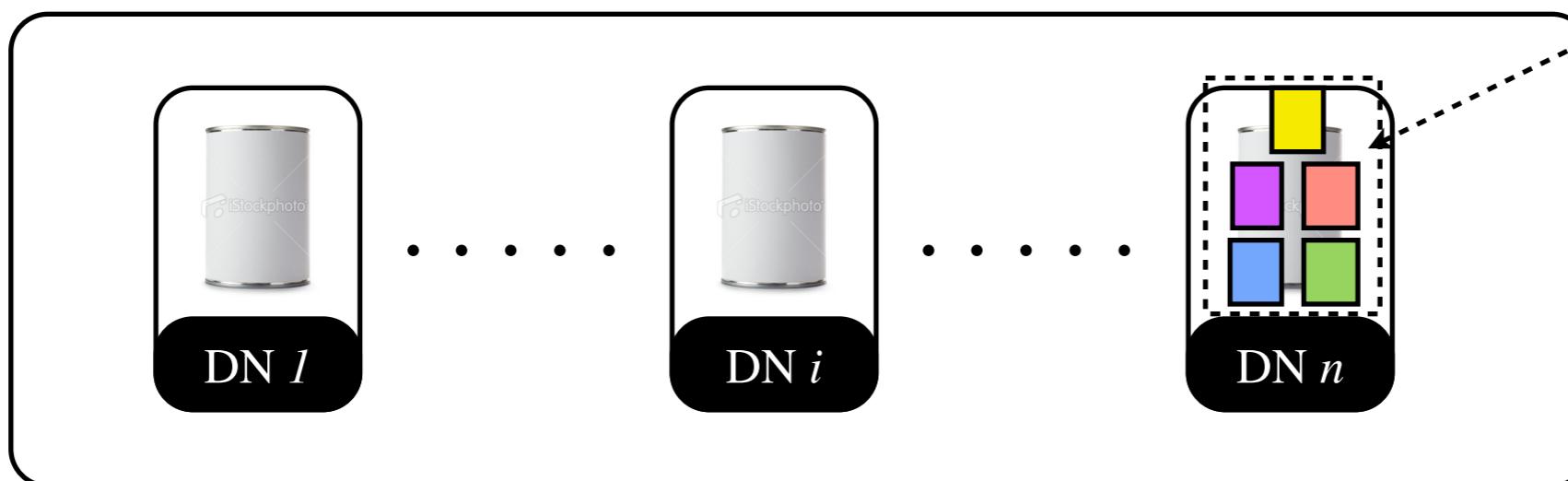
run job

```
map (offset, tuple) {  
    if (sourceIP == 120.115.124.34)  
        output(sourceIP, pageURL)  
}
```

MapReduce



HDFS



Job Execution

```
ColumnInputFormat.setColumns(  
    job, "sourceIP, pageURL");
```

sourceIp
pageURL
visitDate

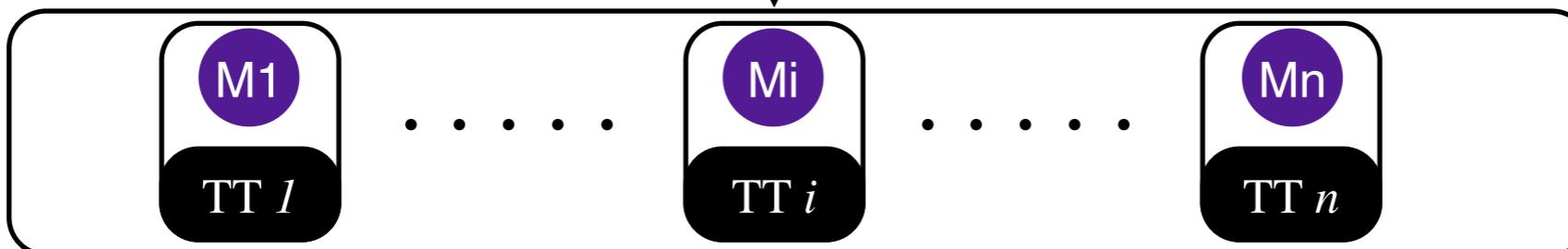
adRevenue
searchWord



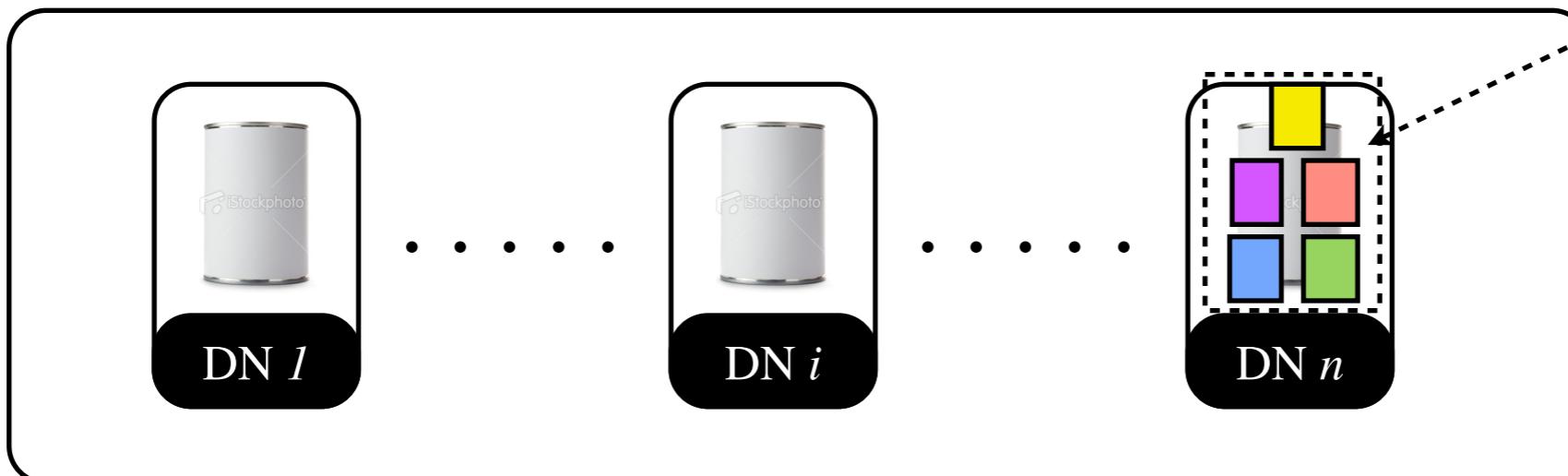
run job

```
map (offset, tuple) {  
    if (sourceIP == 120.115.124.34)  
        output(sourceIP, pageURL)  
}
```

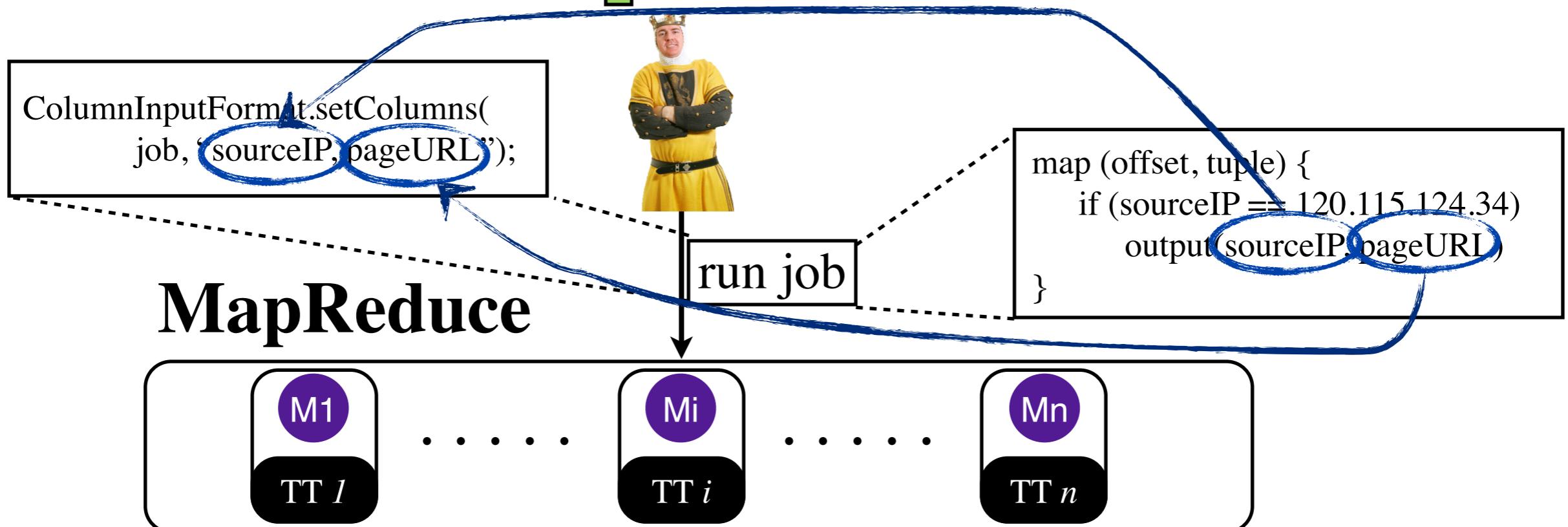
MapReduce



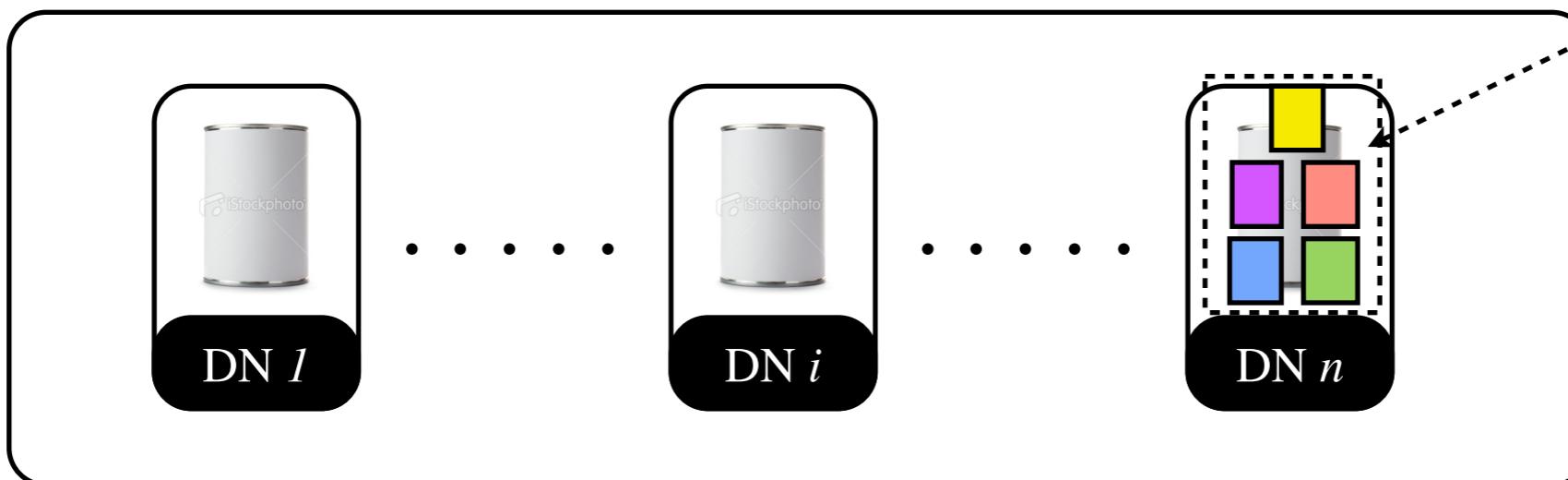
HDFS



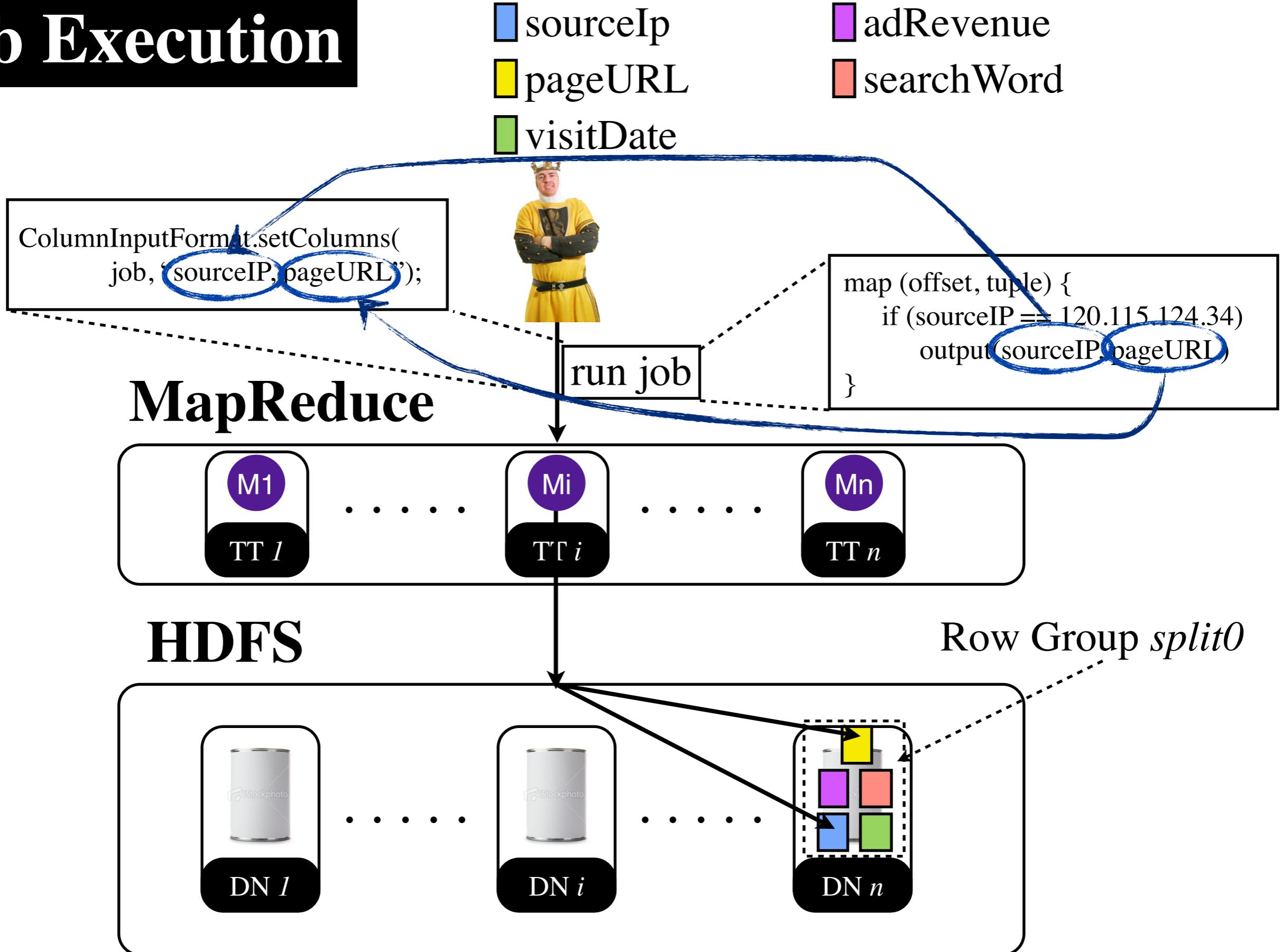
Job Execution



HDFS



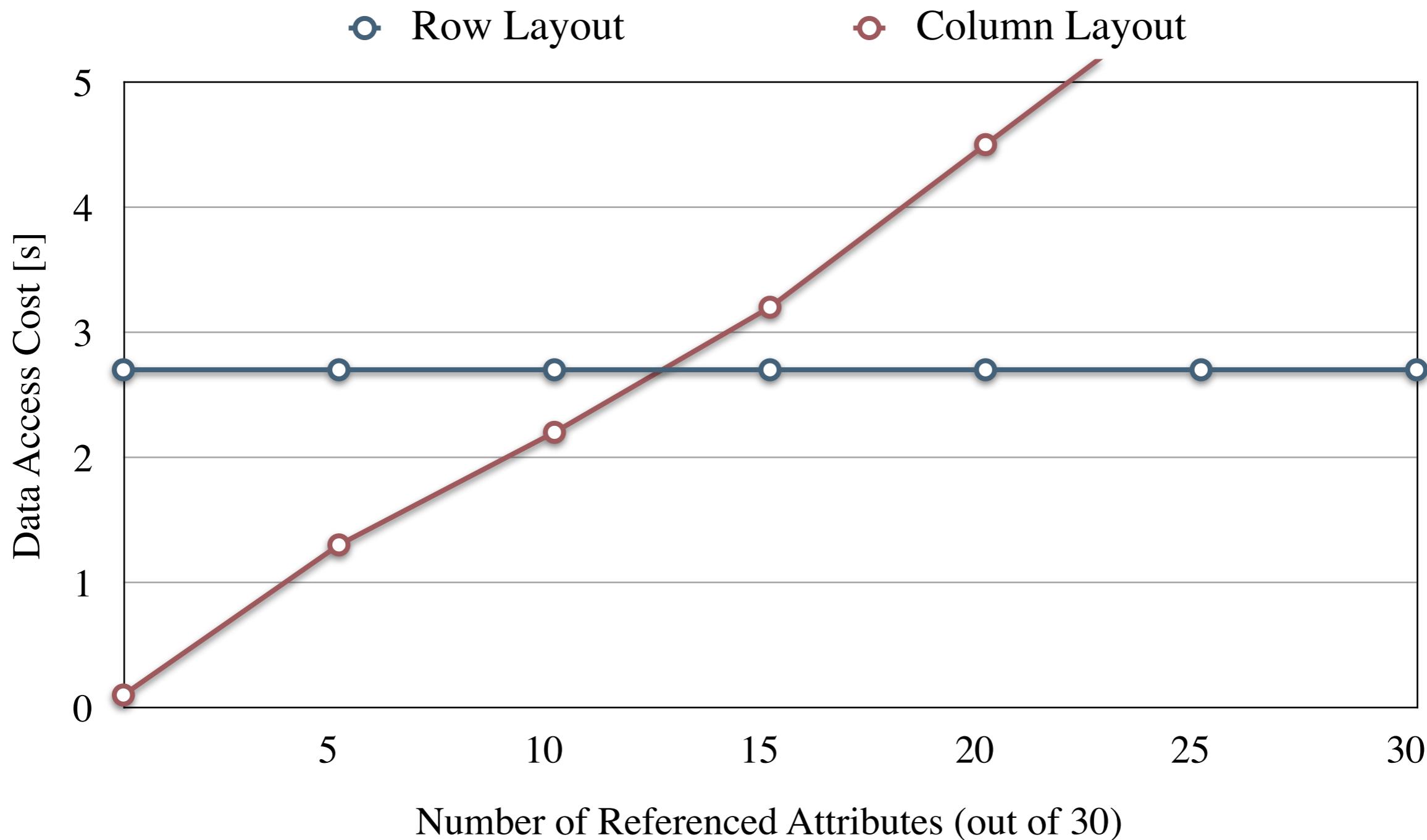
Job Execution



PAX Layout in MapReduce

SELECT a_1, a_2, \dots

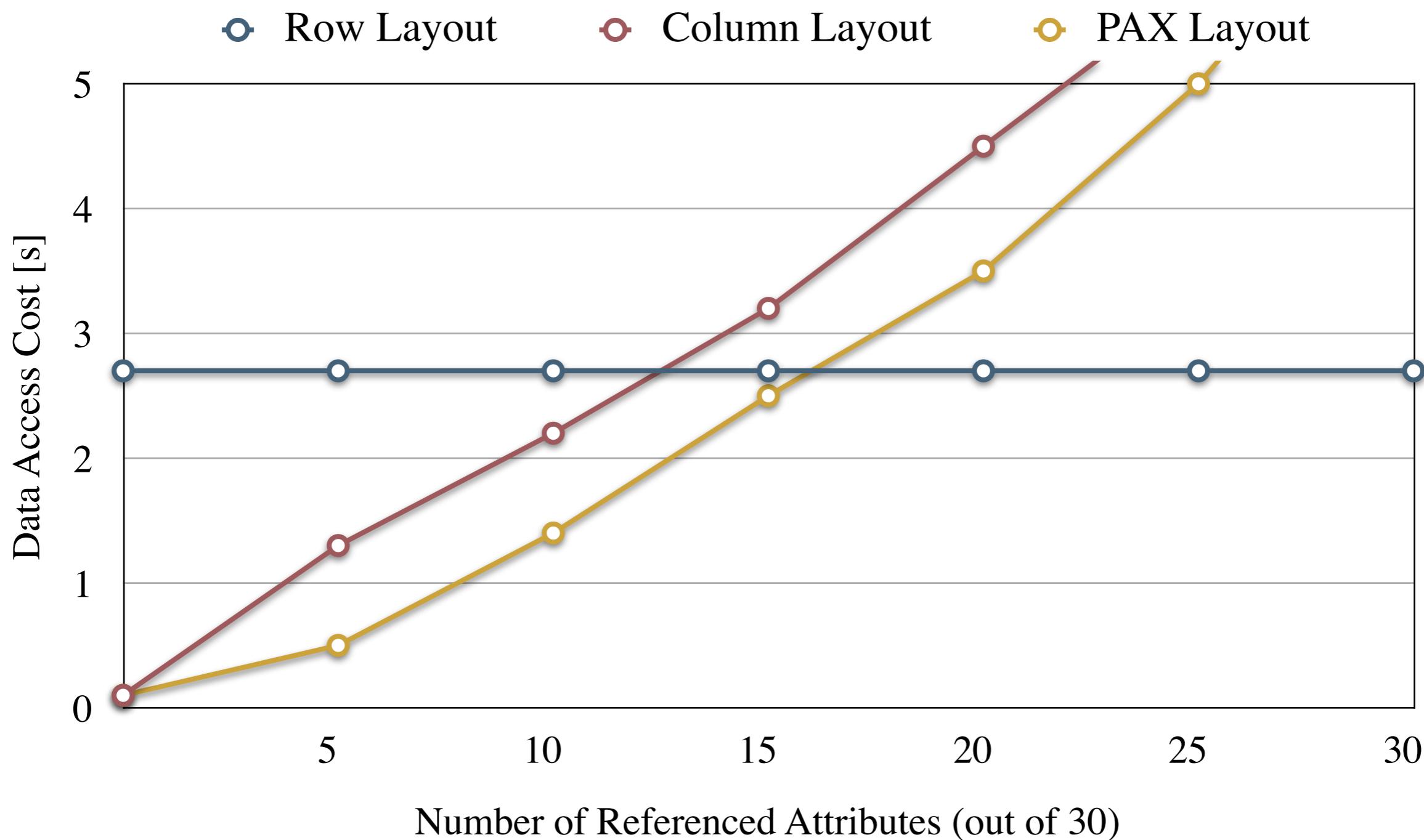
FROM table30Attrs



PAX Layout in MapReduce

SELECT a_1, a_2, \dots

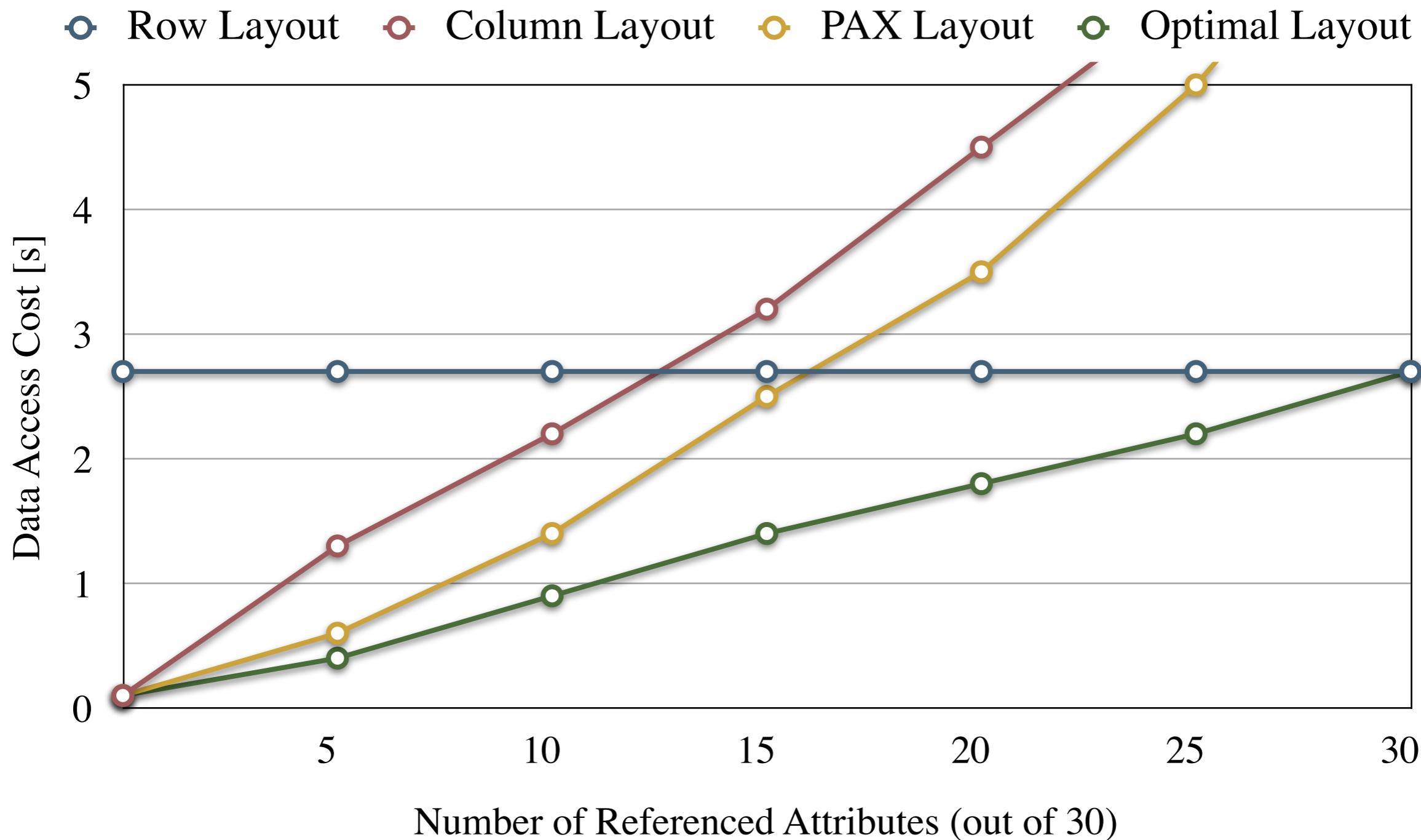
FROM table30Attrs



Far from Optimal Layout

SELECT a_1, a_2, \dots

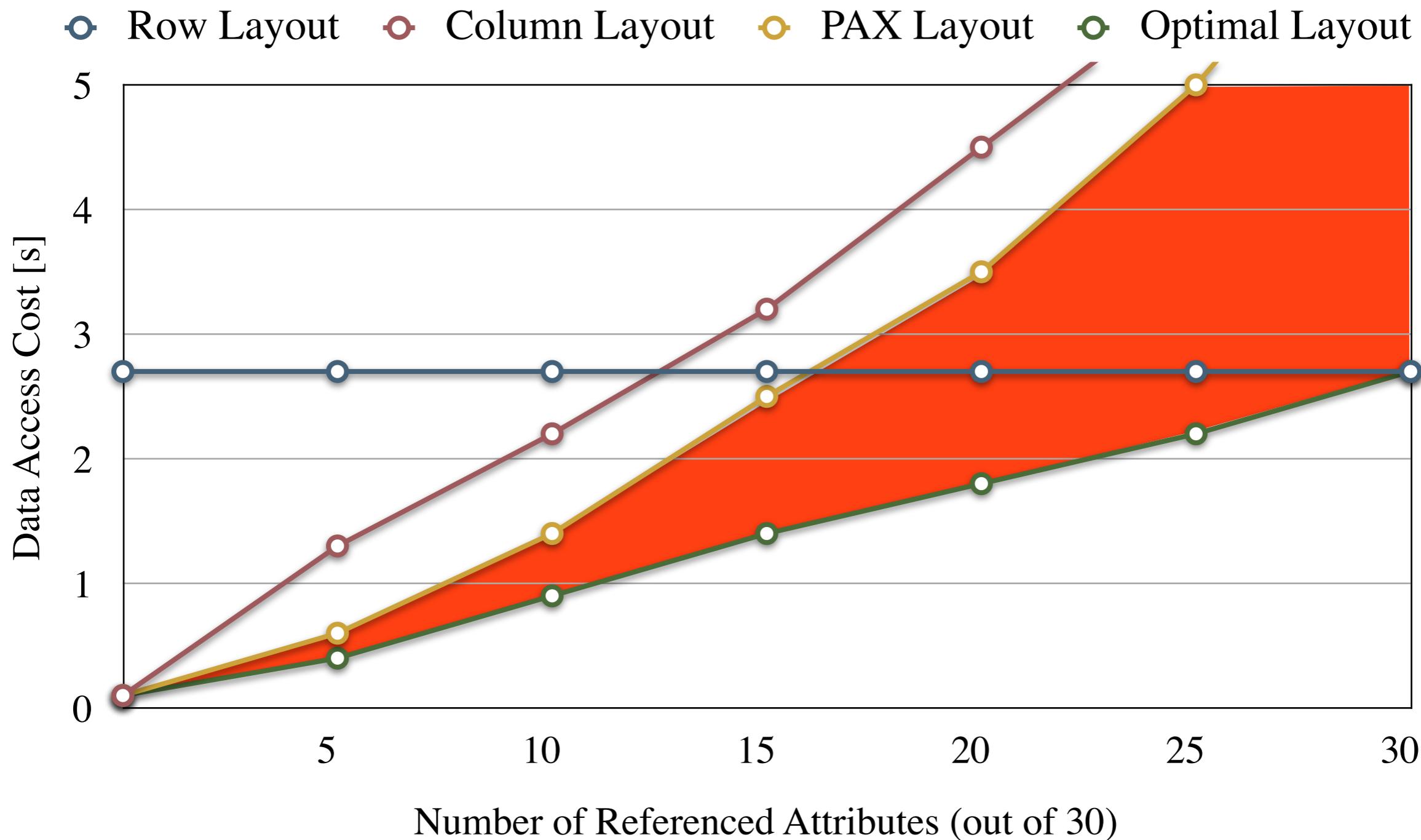
FROM table30Attrs



Far from Optimal Layout

SELECT a_1, a_2, \dots

FROM table30Attrs



Data Layouts in MapReduce

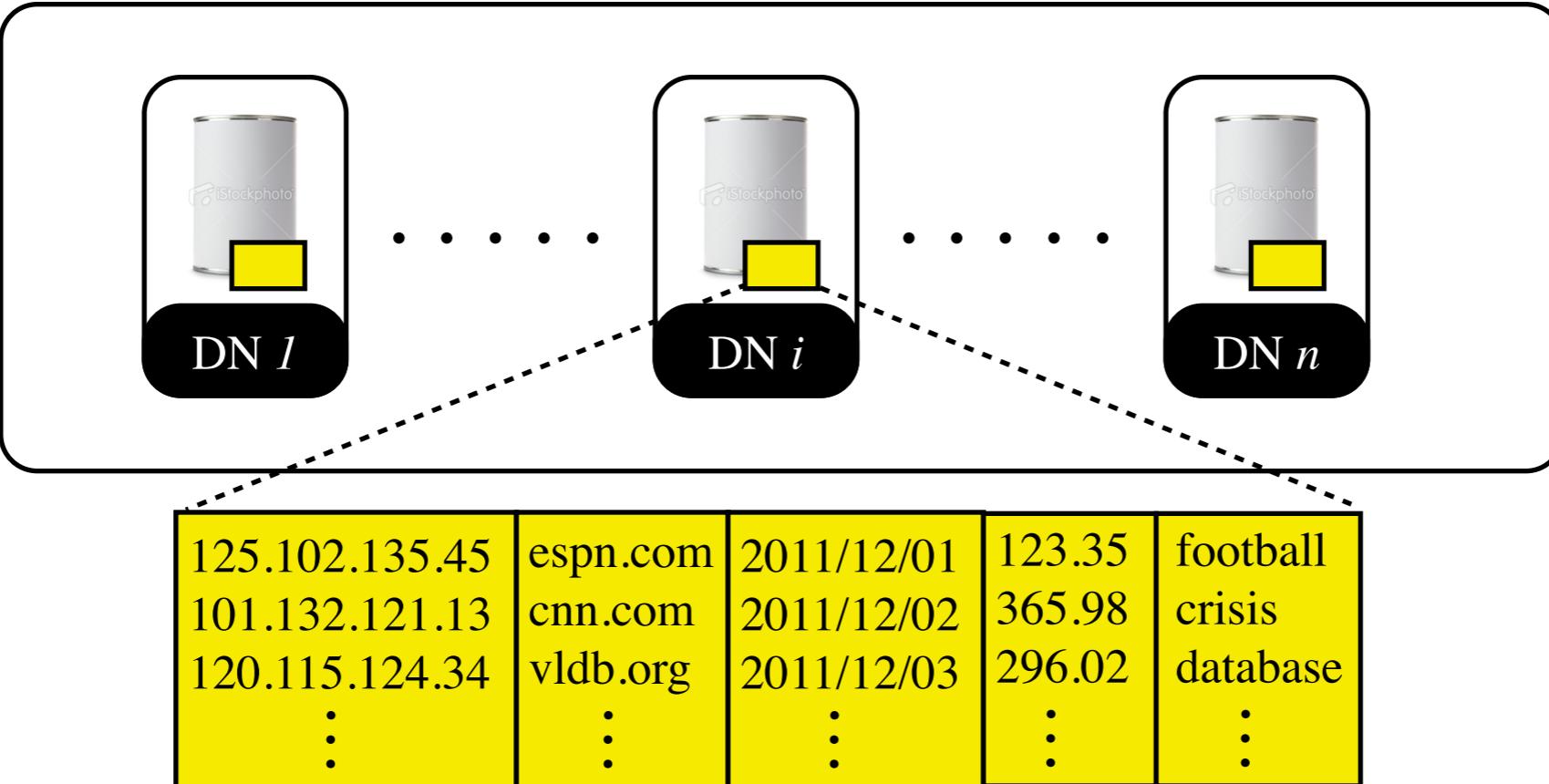
Initial	2009	2010	2011
Row	CFile	Cheetah	RCFile
Read Unnecessary columns			
	Tuple Reconstruction	Tuple Reconstruction	Tuple Reconstruction
	High network costs		
		Block level compression	
		Poor I/O Saving	Poor I/O Saving

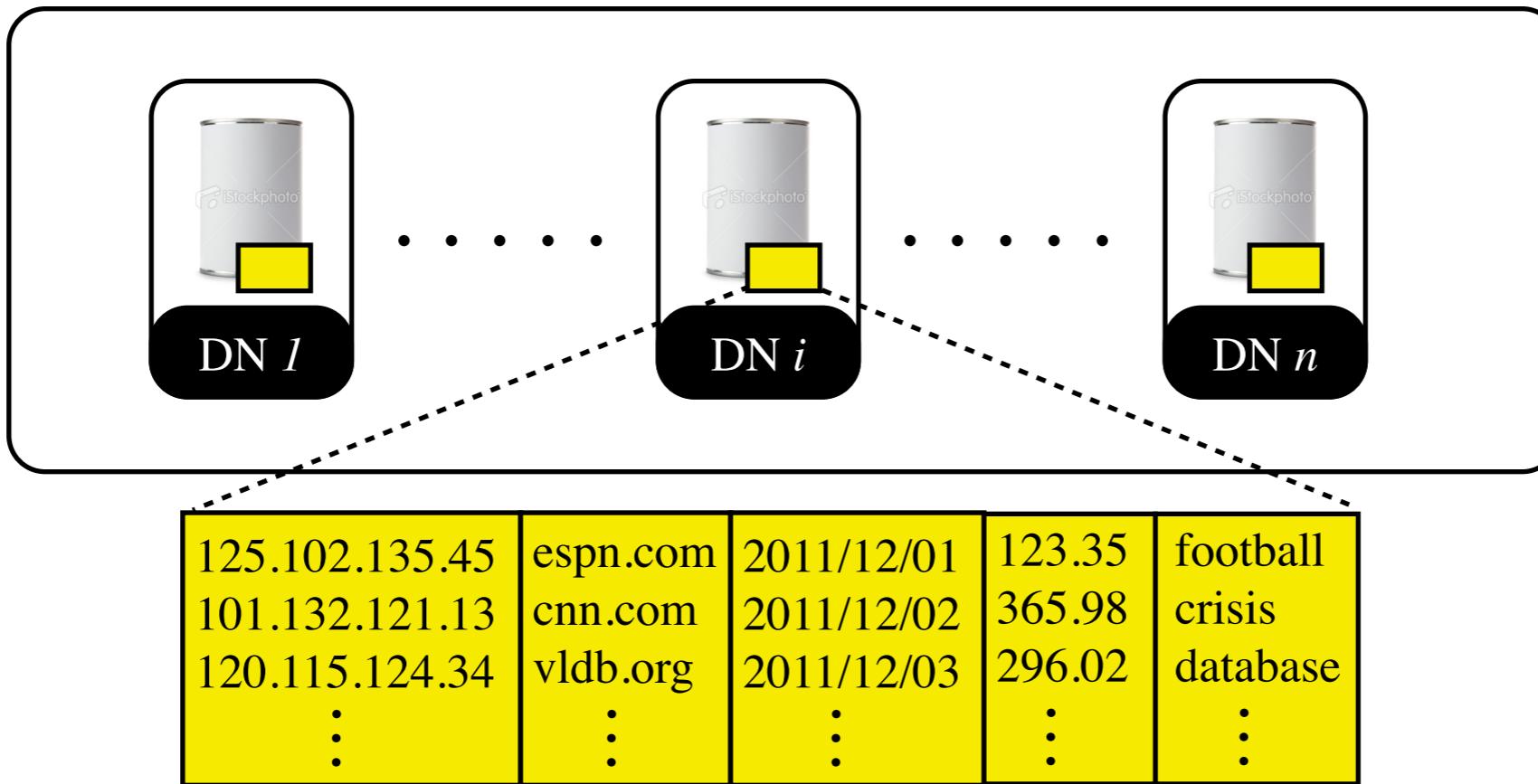
Data Layouts in MapReduce

Initial	2009	2010	2011	2011
Row	CFile	Cheetah	RCFile	CIF
Read Unnecessary columns				
	Tuple Reconstruction	Tuple Reconstruction	Tuple Reconstruction	Tuple Reconstruction
	High network costs			
		Block level compression		
		Poor I/O Saving	Poor I/O Saving	

Trojan Data Layouts

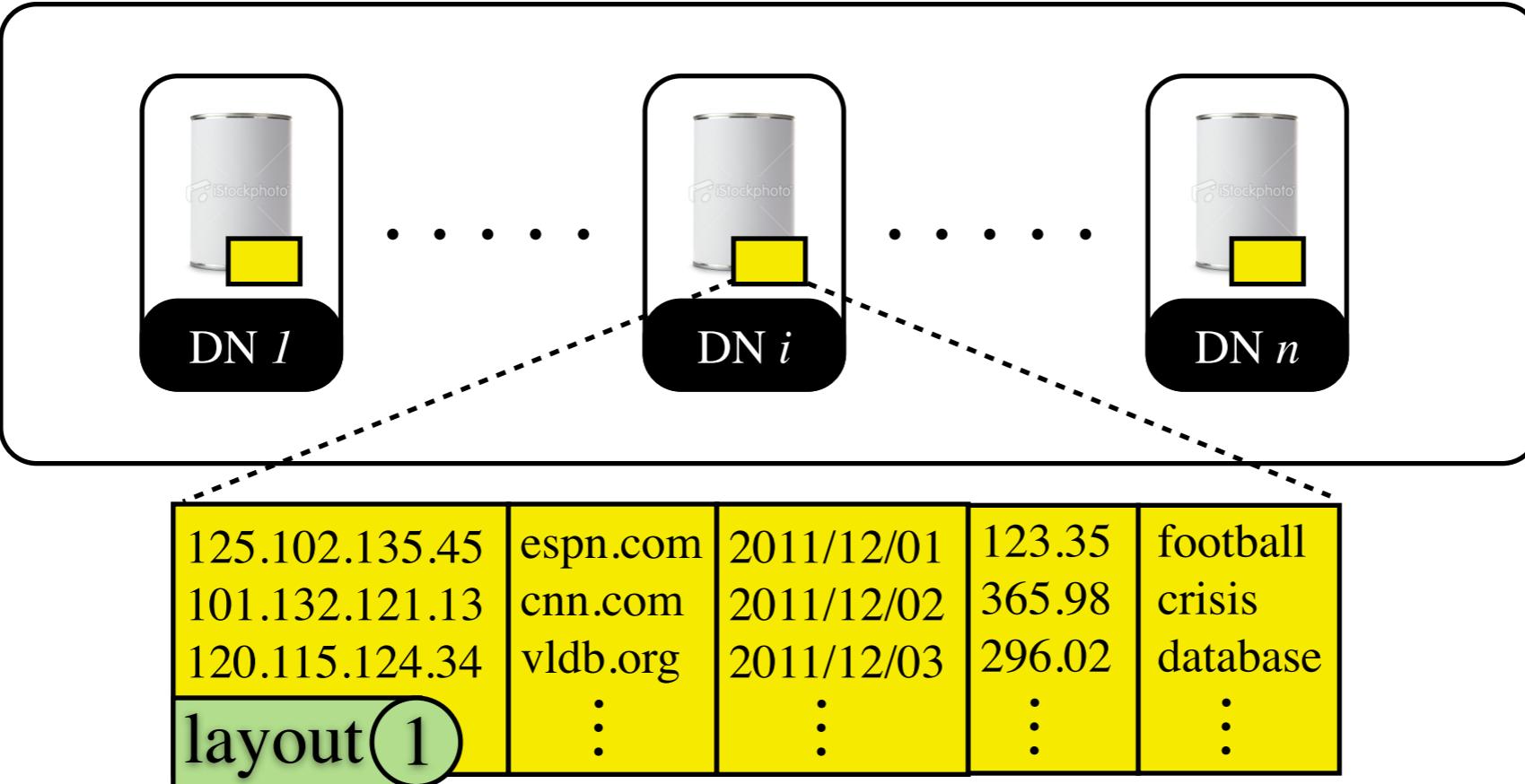
[A. Jindal, J. Quiané, J. Dittrich: Trojan Data Layouts: Right Shoes for a Running Elephant. SoCC 2011]





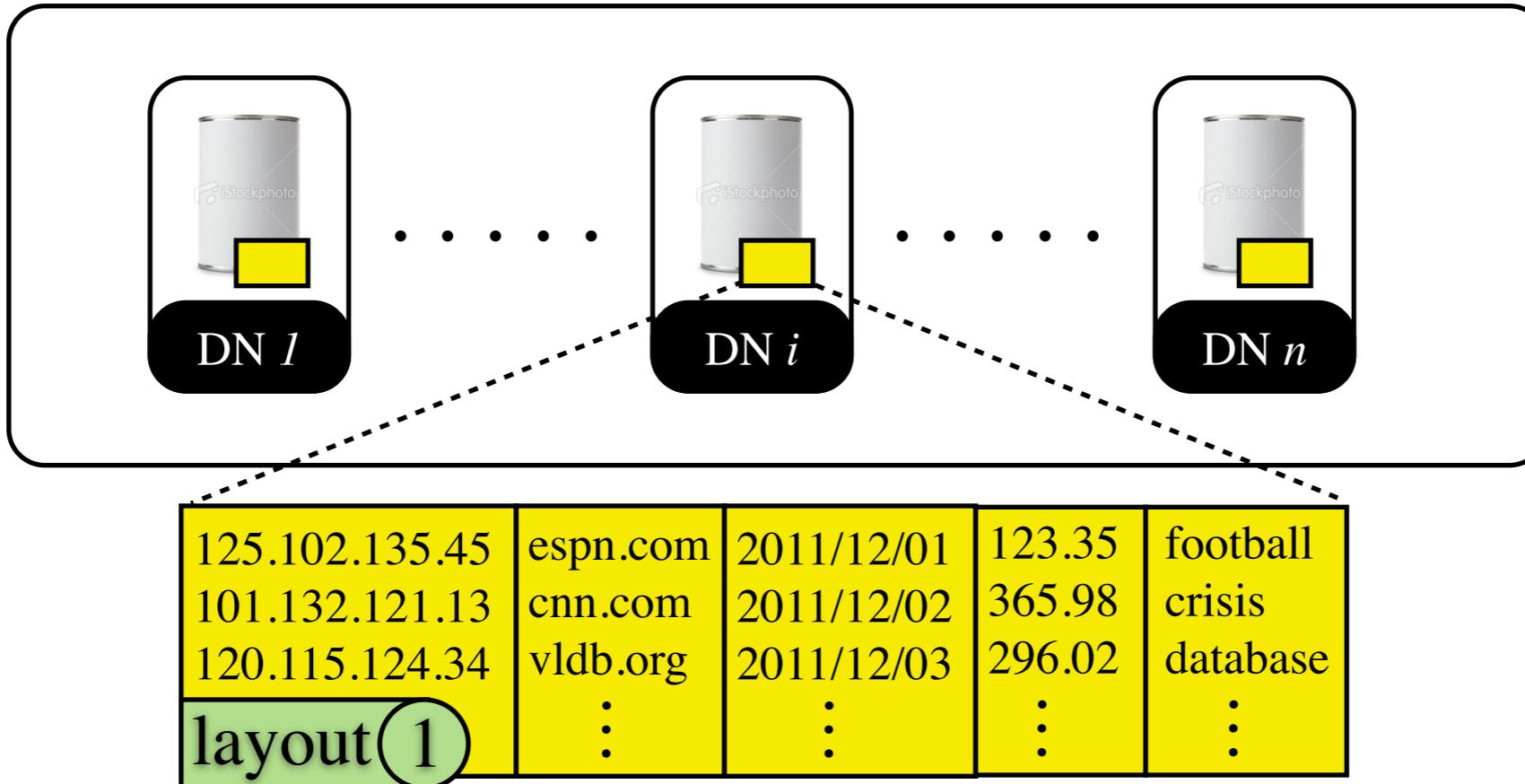
Job 1

```
map (offset, tuple) {  
    if (sourceIP == 105.102.135.45)  
        output(sourceIP, pageURL)  
}
```



Job 1

```
map (offset, tuple) {  
    if (sourceIP == 105.102.135.45)  
        output(sourceIP, pageURL)  
}
```

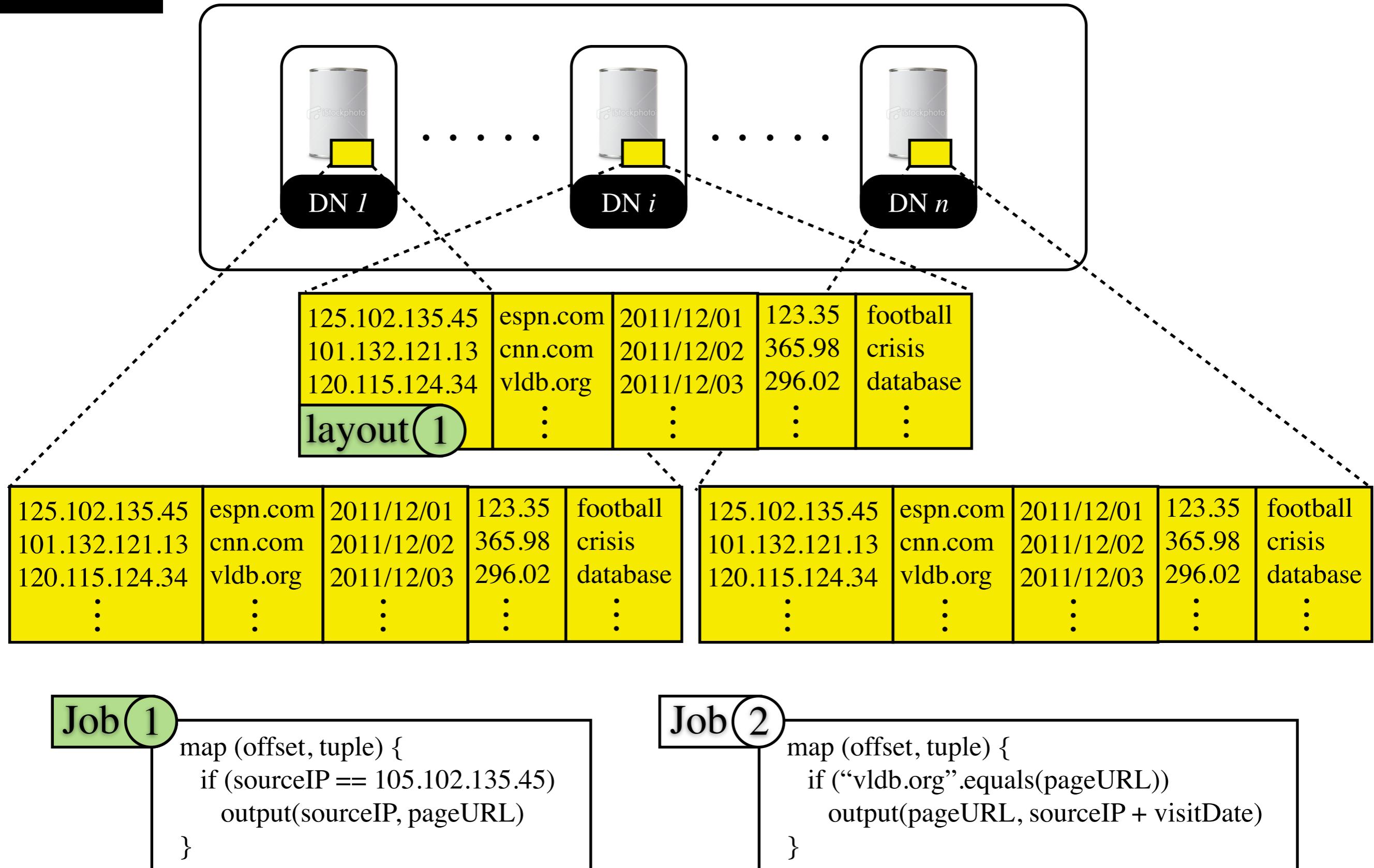


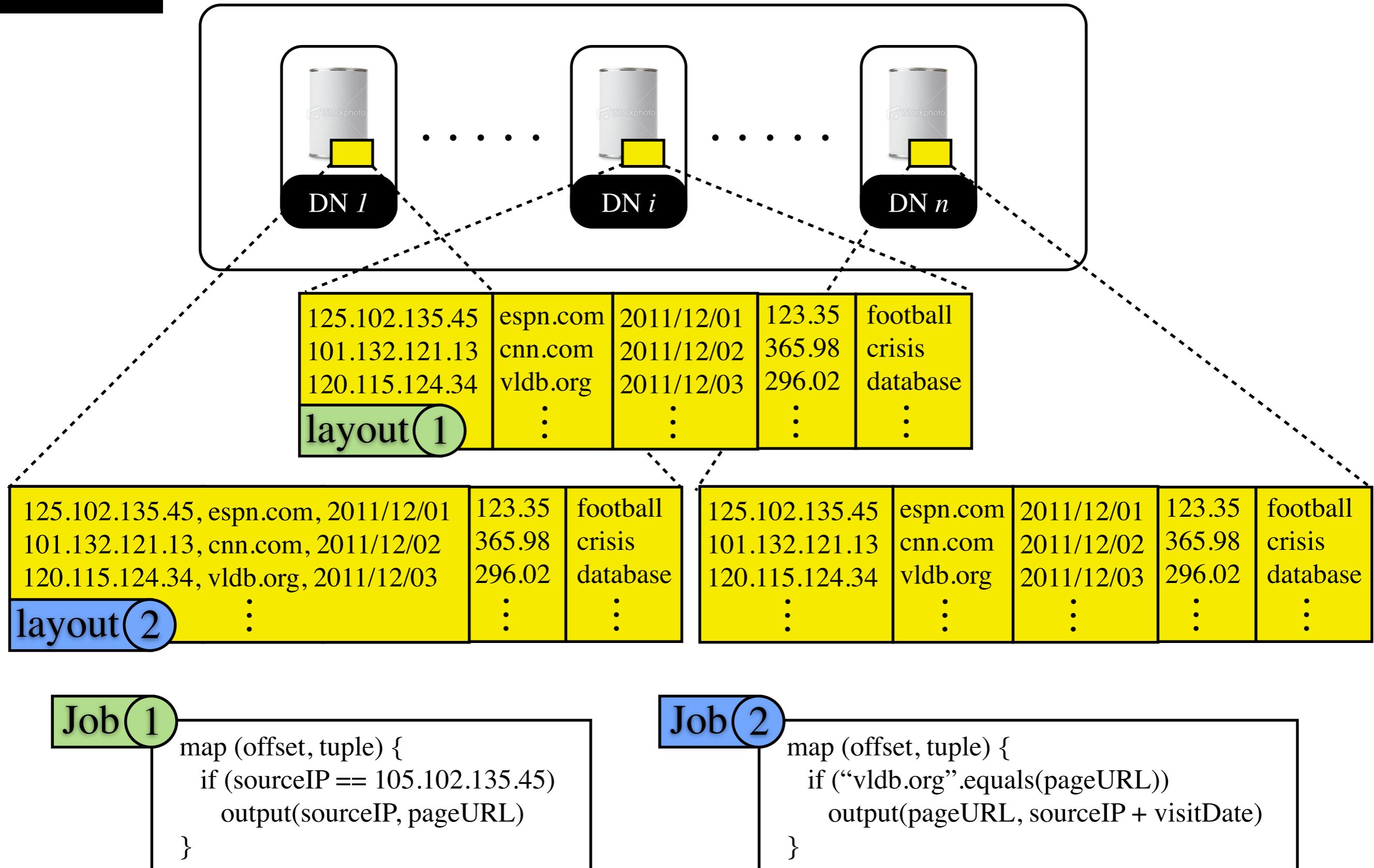
Job 1

```
map (offset, tuple) {  
    if (sourceIP == 105.102.135.45)  
        output(sourceIP, pageURL)  
}
```

Job 2

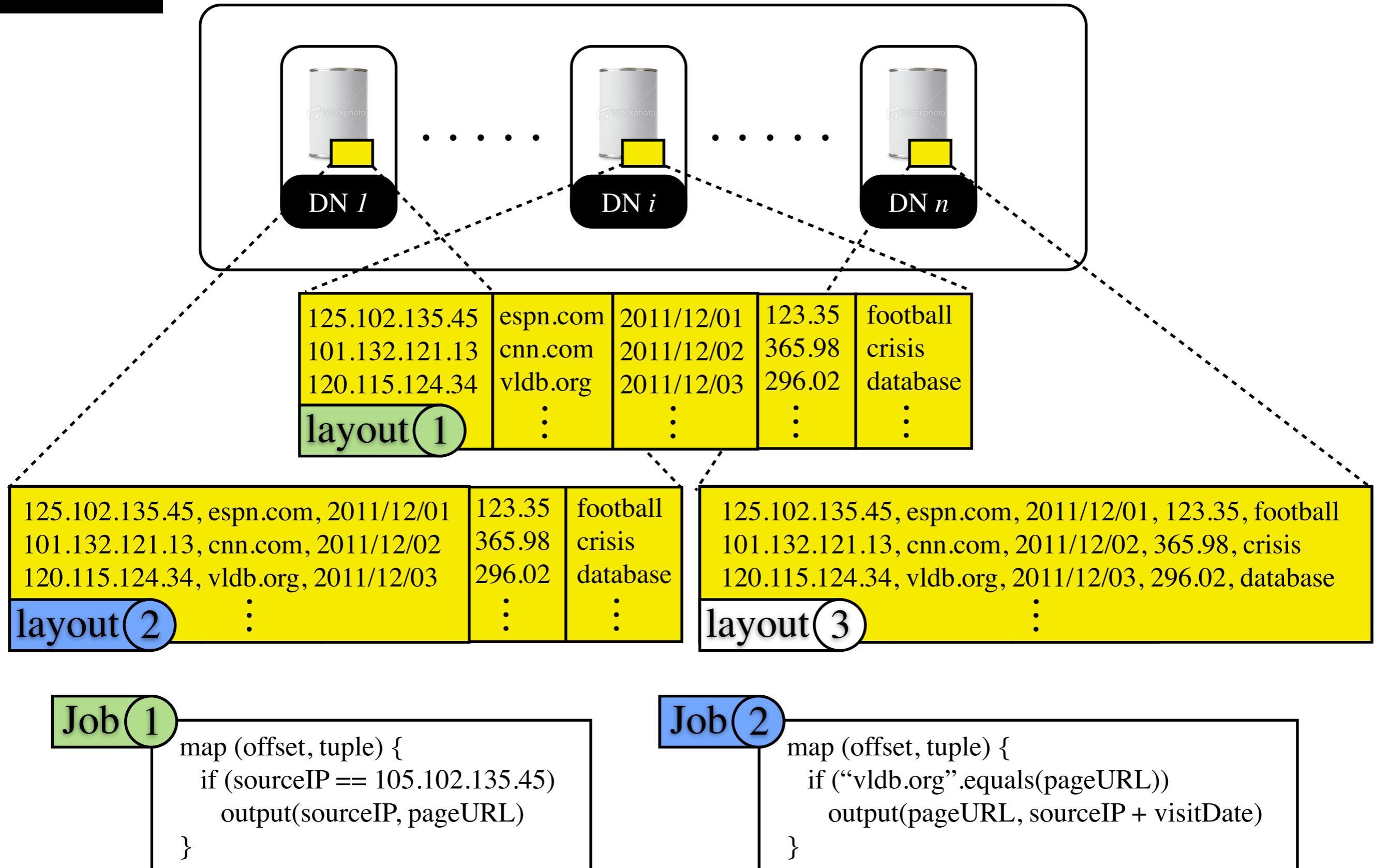
```
map (offset, tuple) {  
    if ("vldb.org".equals(pageURL))  
        output(pageURL, sourceIP + visitDate)  
}
```





Idea

HDFS



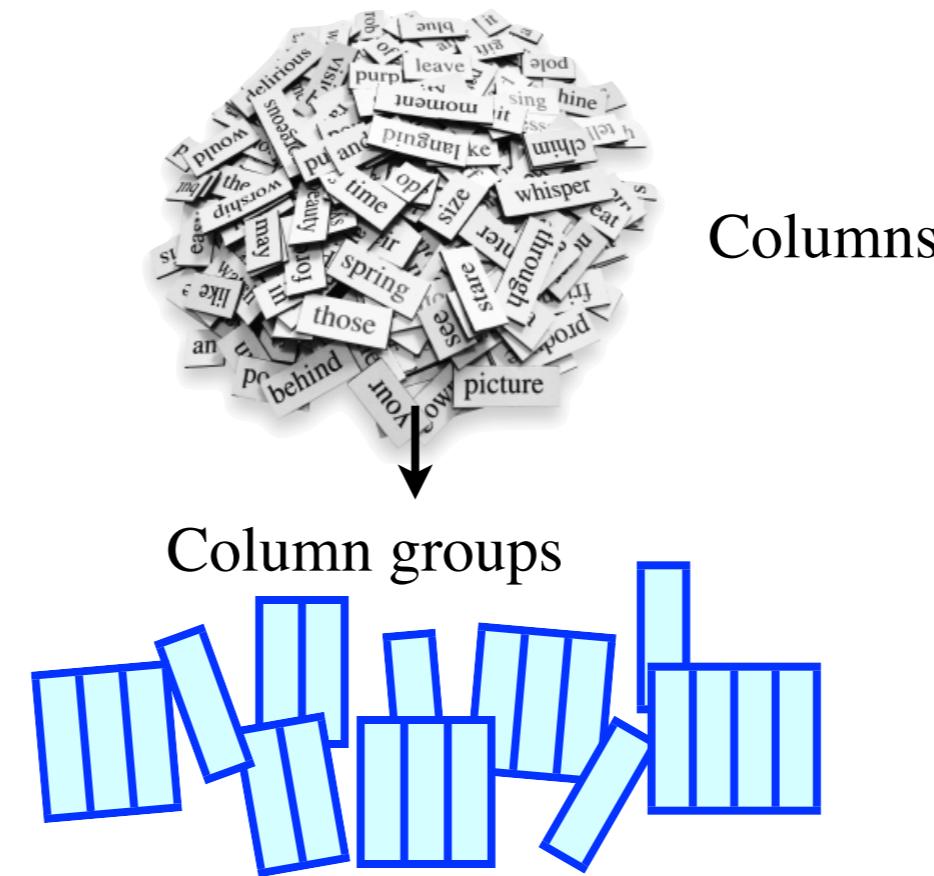
Single HDFS Block Replica

Single HDFS Block Replica

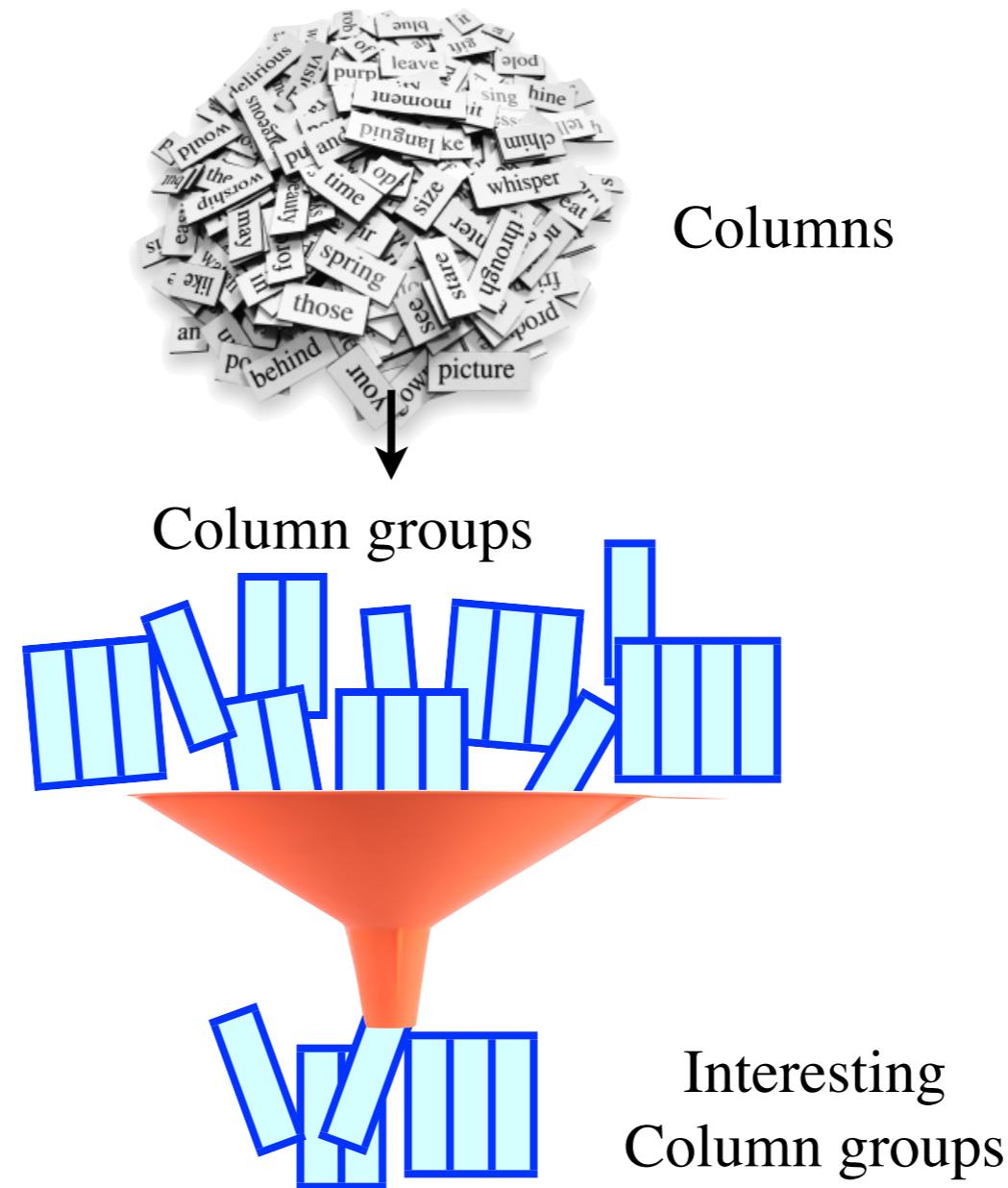


Columns

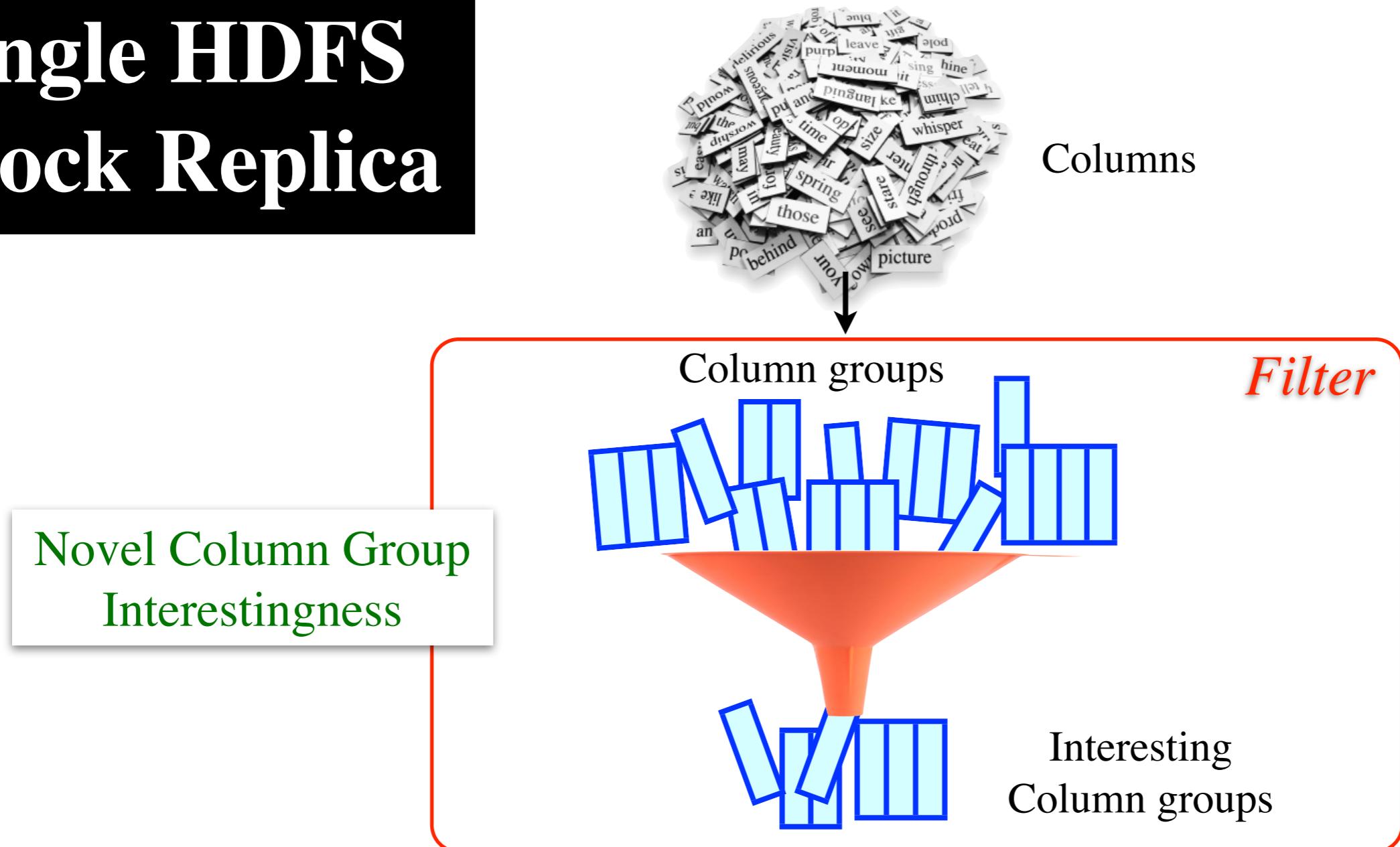
Single HDFS Block Replica



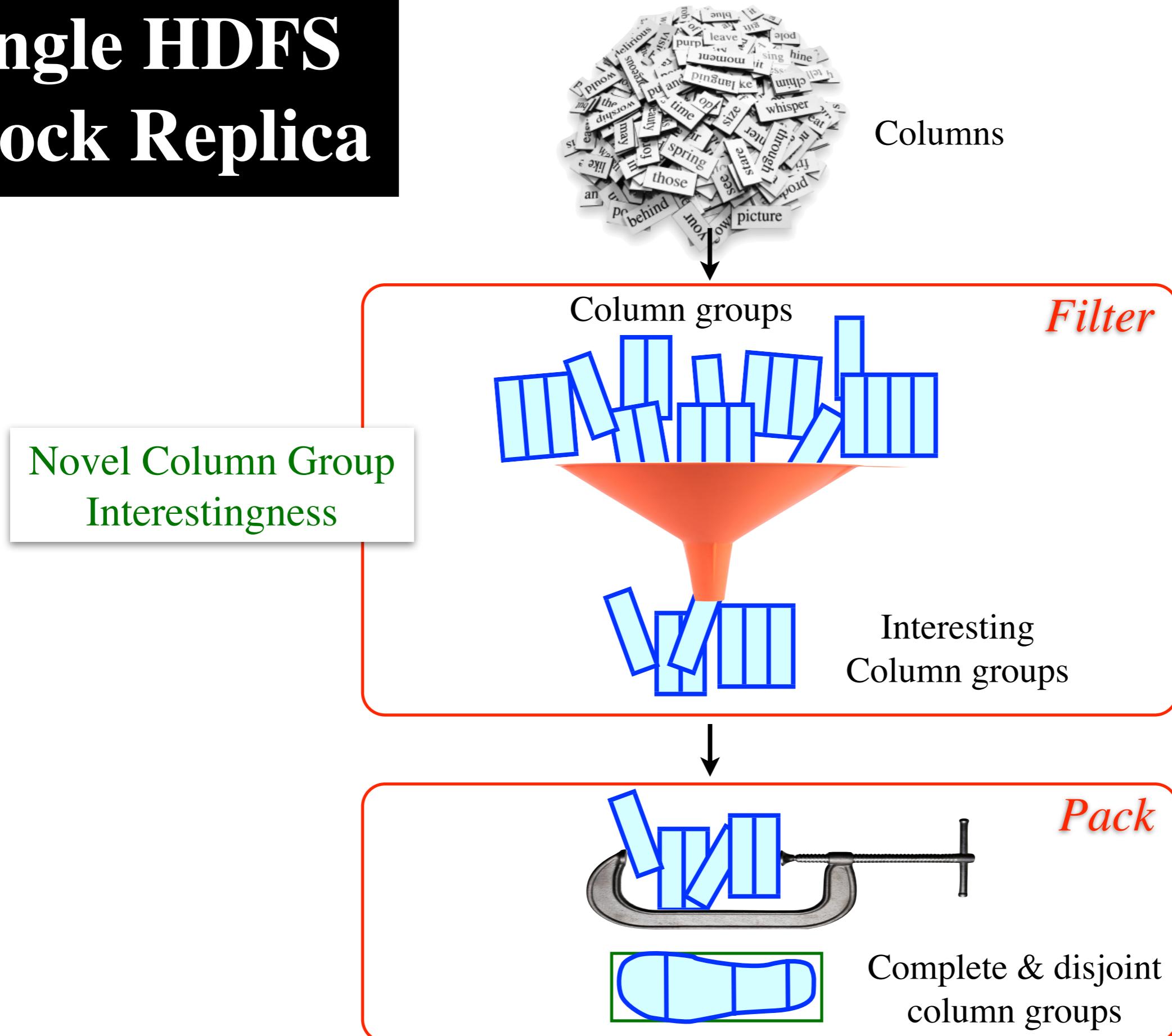
Single HDFS Block Replica



Single HDFS Block Replica



Single HDFS Block Replica



Single HDFS Block Replica

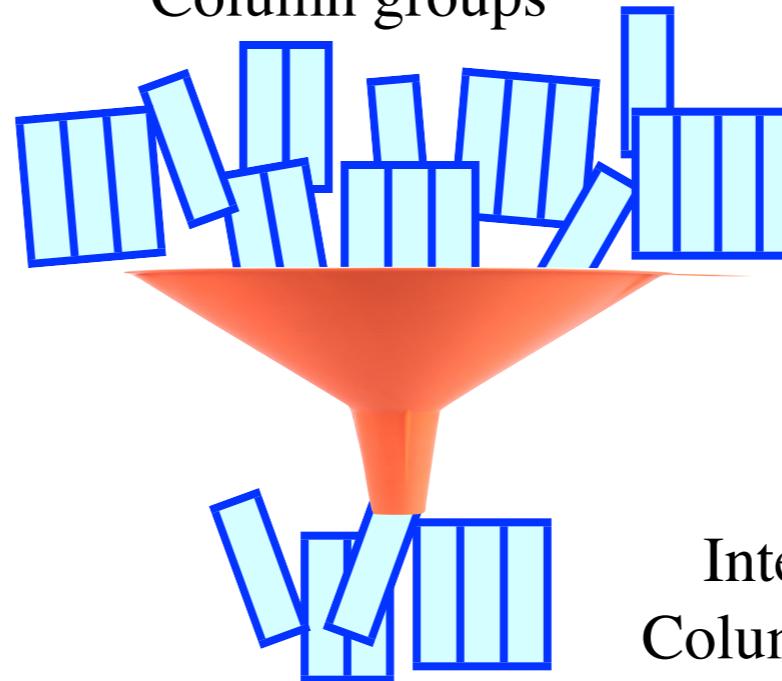


Columns

Novel Column Group
Interestingness

Column groups

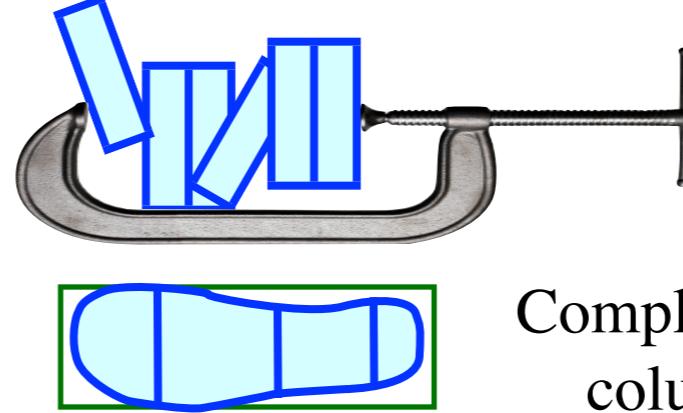
Filter



Interesting
Column groups

Column Group Packing
as 0-1 Knapsack

Pack



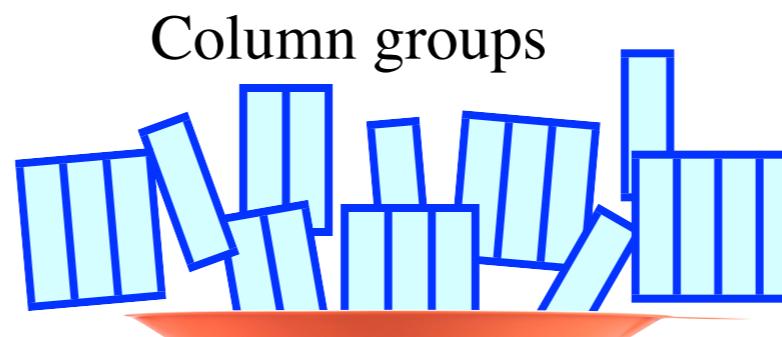
Complete & disjoint
column groups

Single HDFS Block Replica



Columns

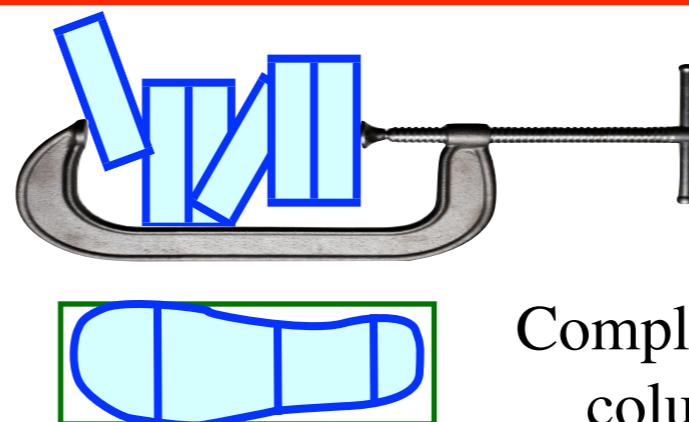
Novel Column Group



Filter

- see our paper for details: [Trojan Data Layouts, SoCC'11]

Column Group Packing as 0-1 Knapsack



Pack

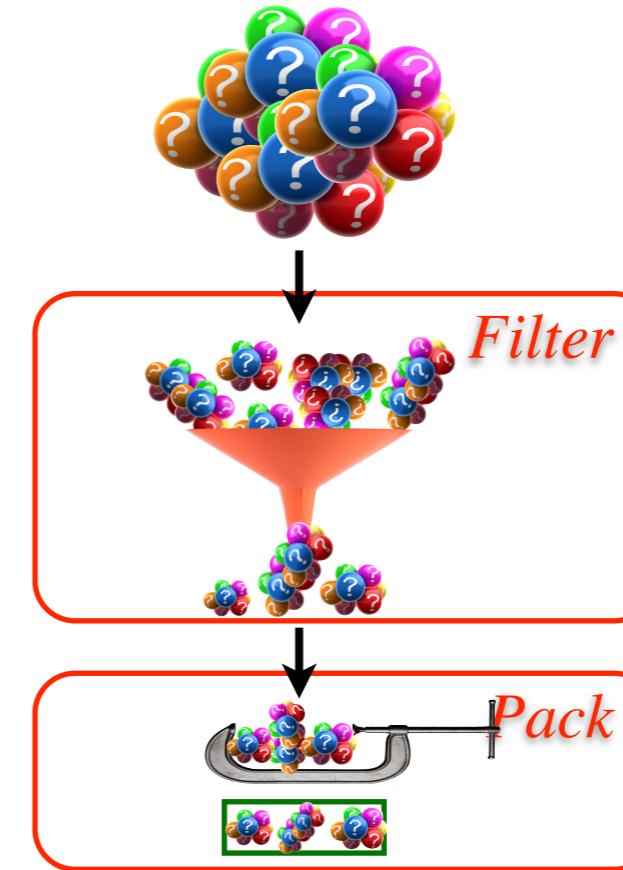
Complete & disjoint column groups

Multiple HDFS Block Replica

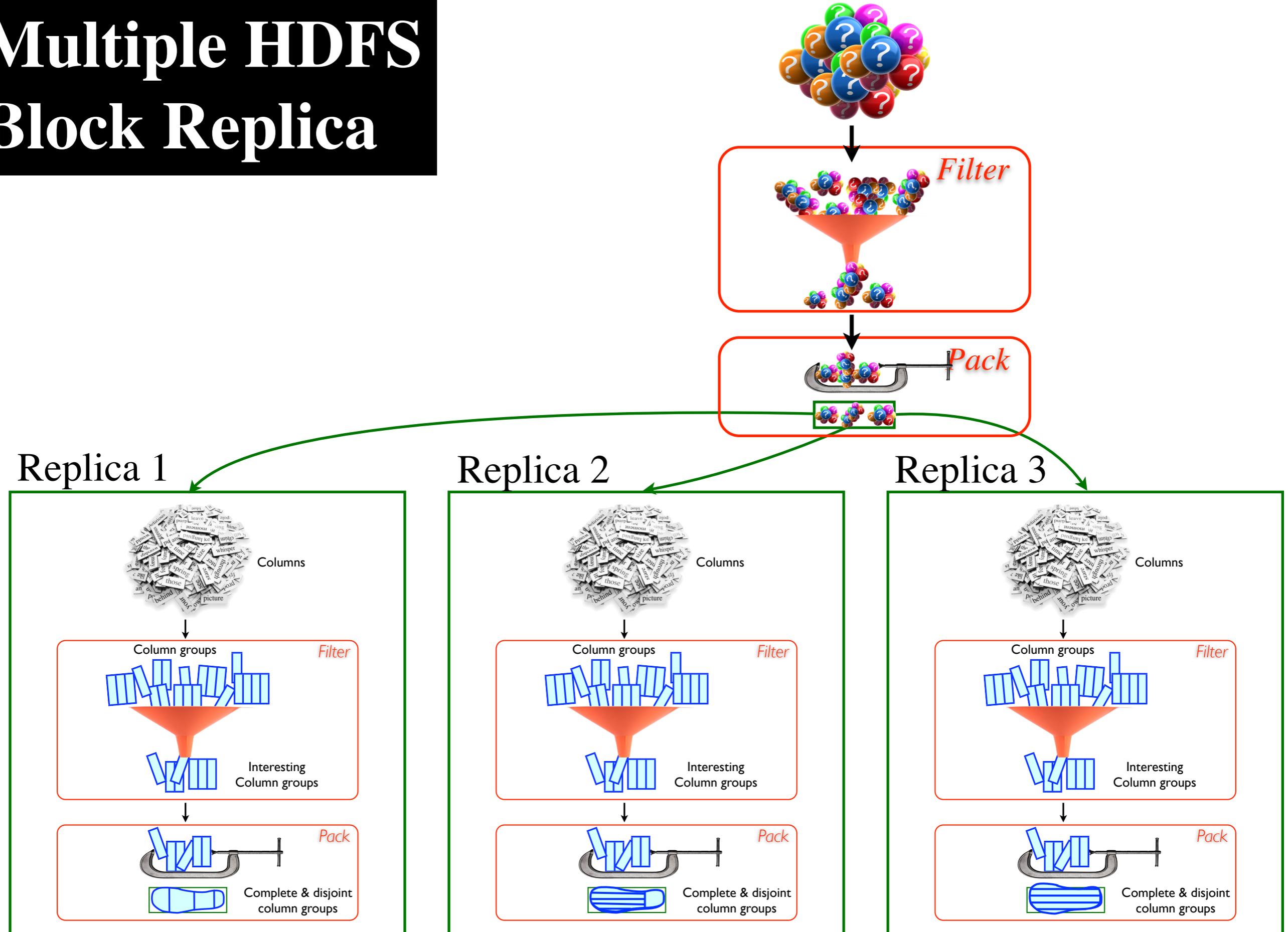
Multiple HDFS Block Replica



Multiple HDFS Block Replica

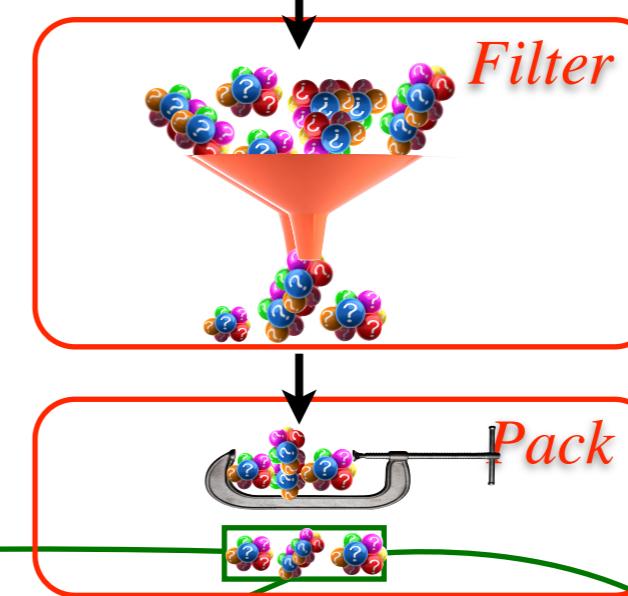


Multiple HDFS Block Replica

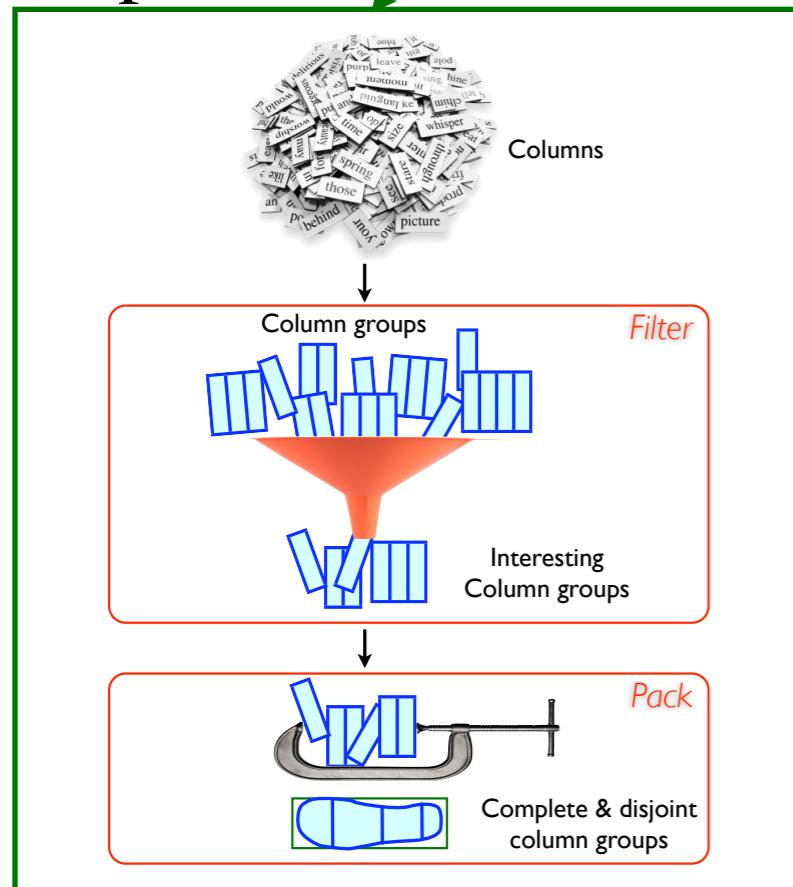


Multiple HDFS Block Replica

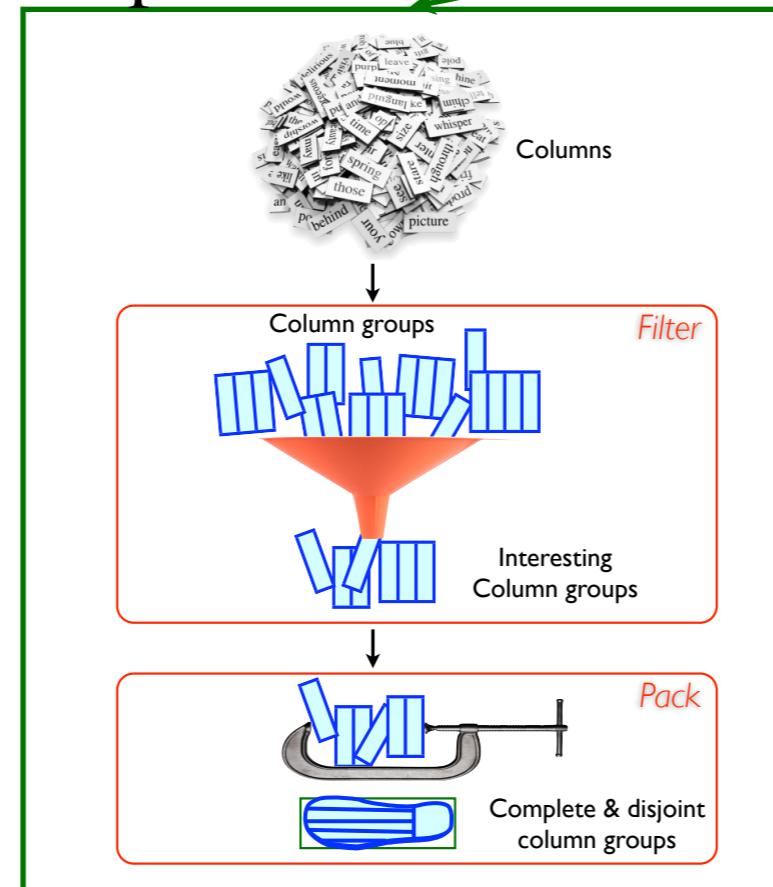
$Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7, Q_8$



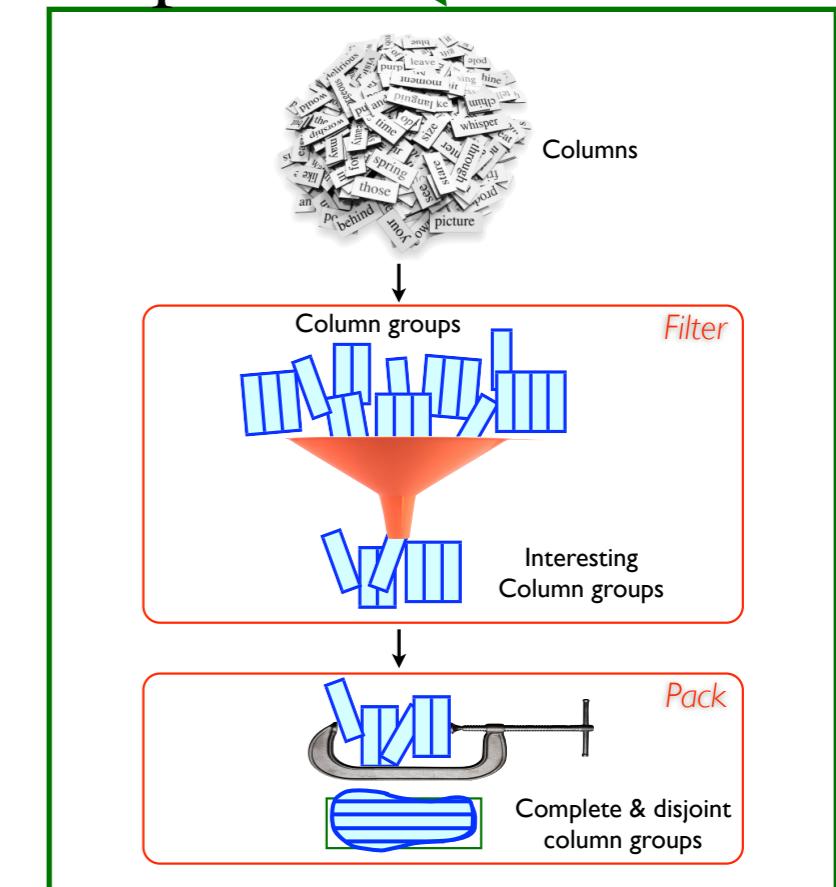
Replica 1



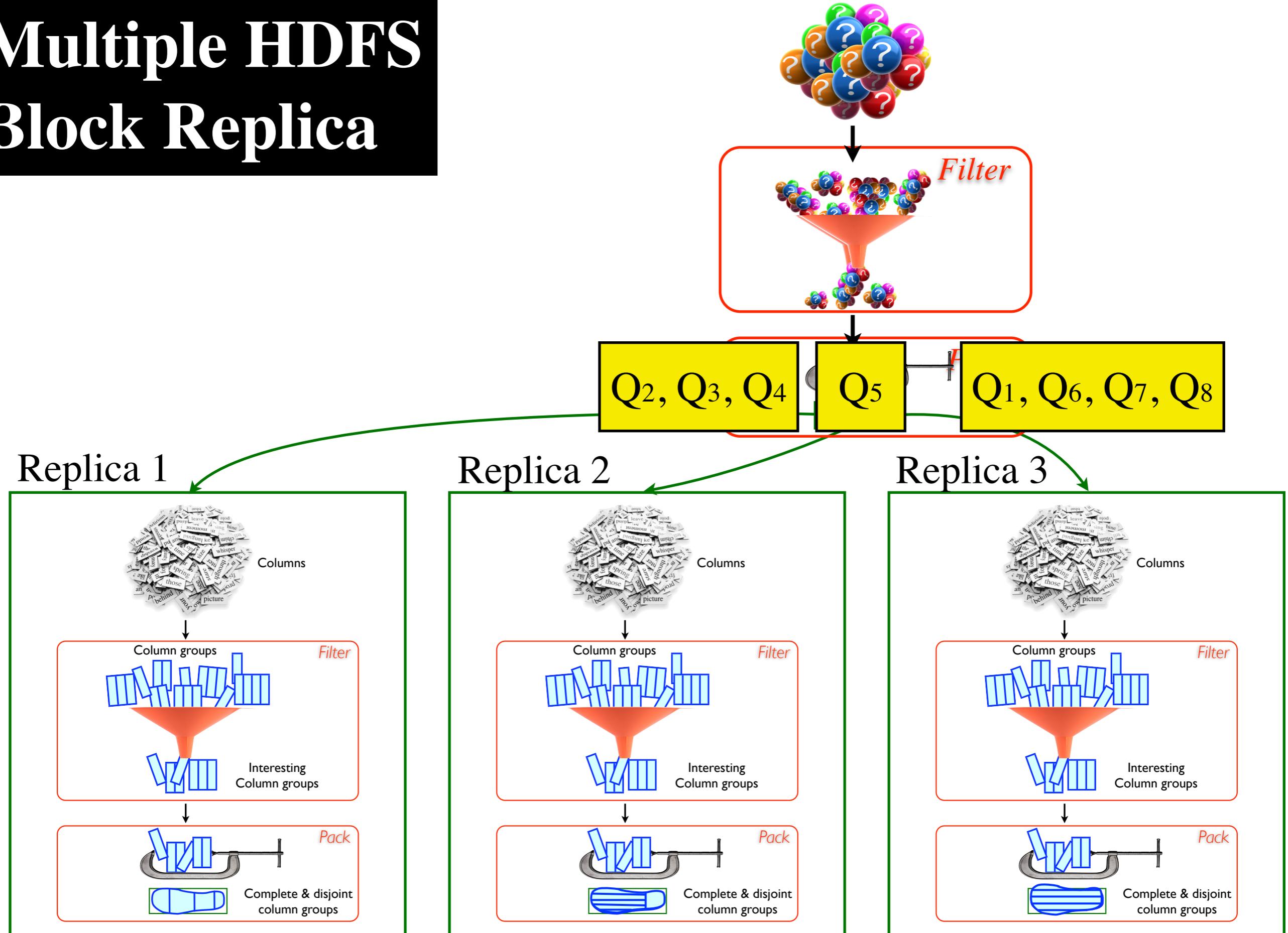
Replica 2



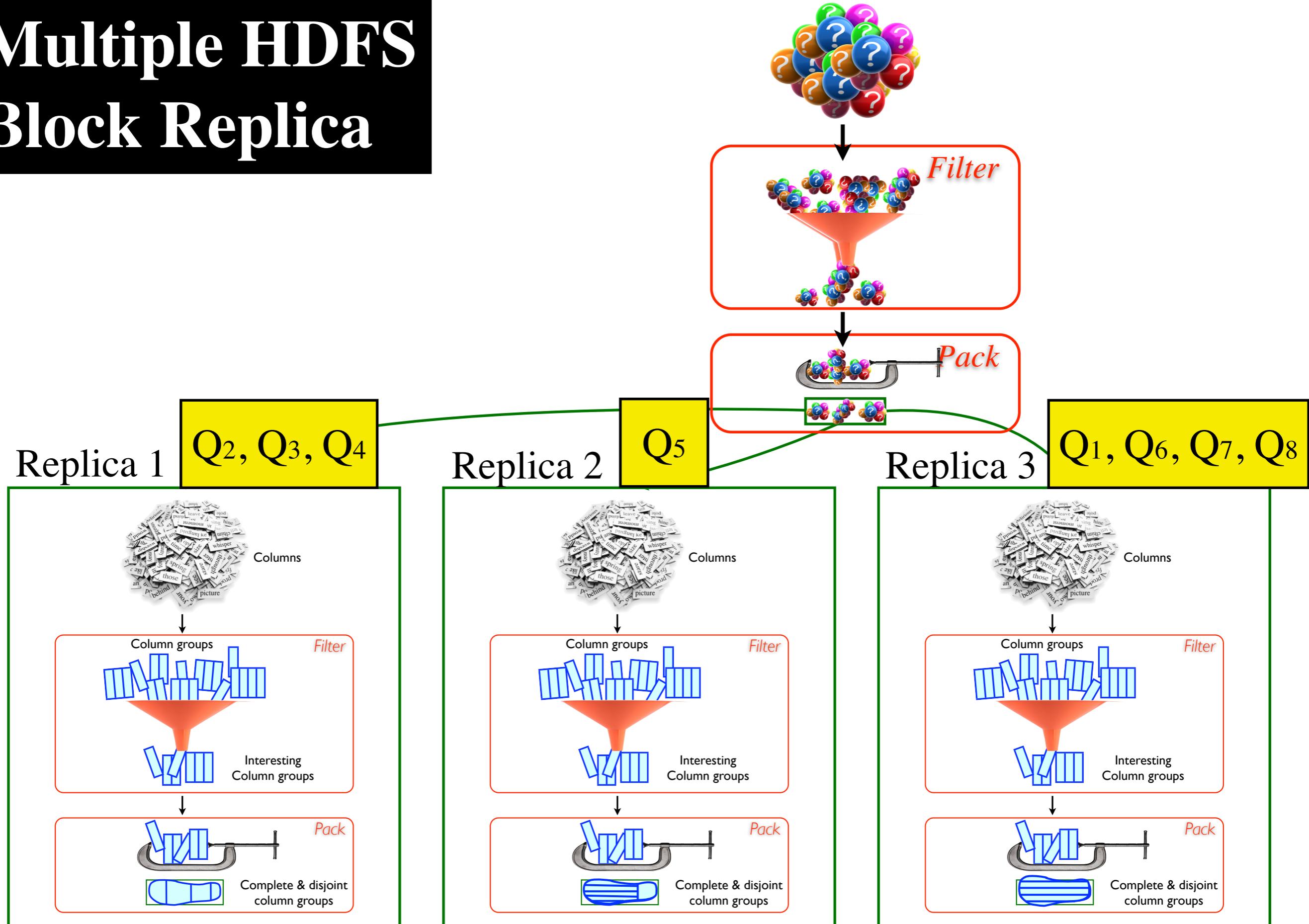
Replica 3



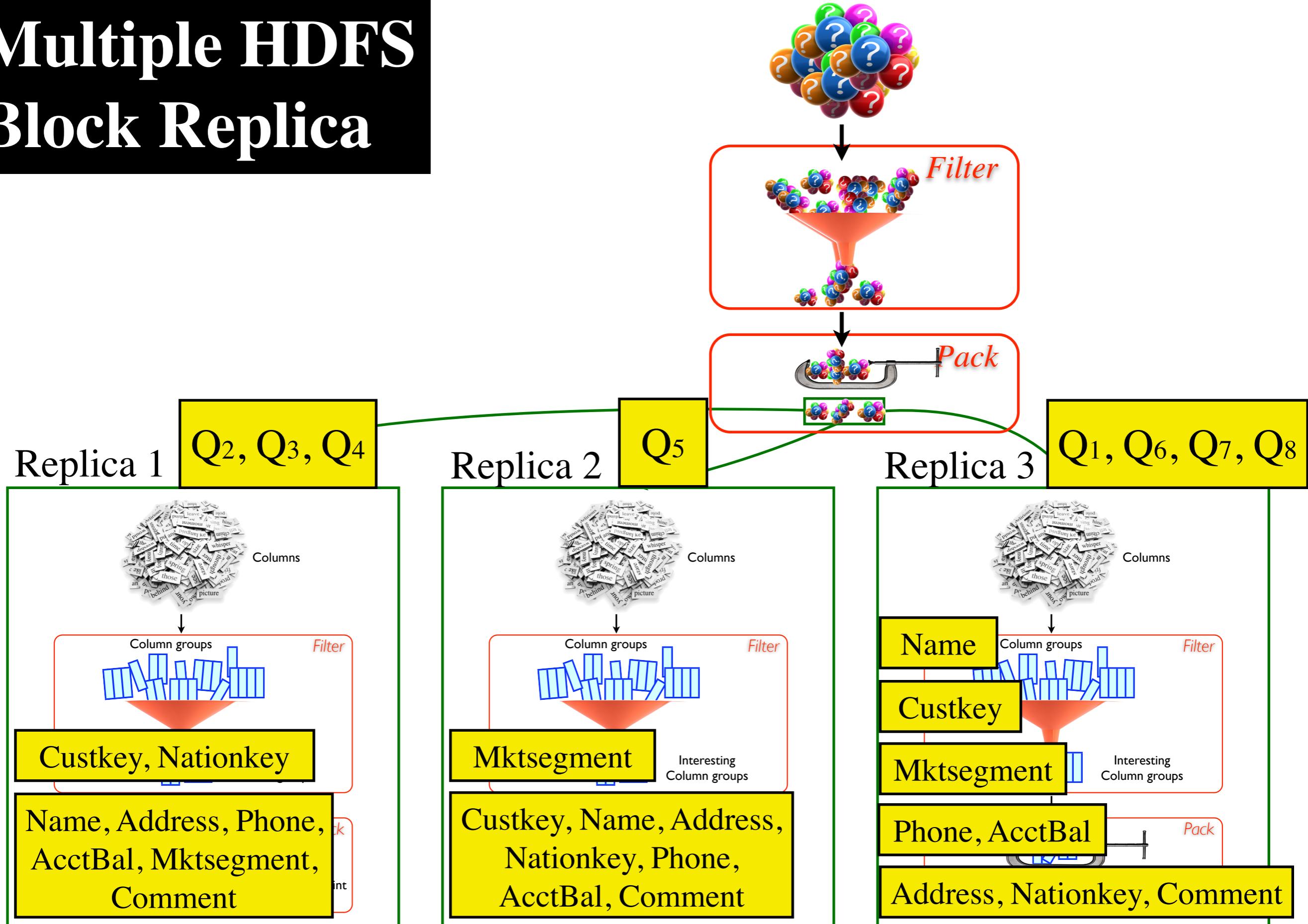
Multiple HDFS Block Replica



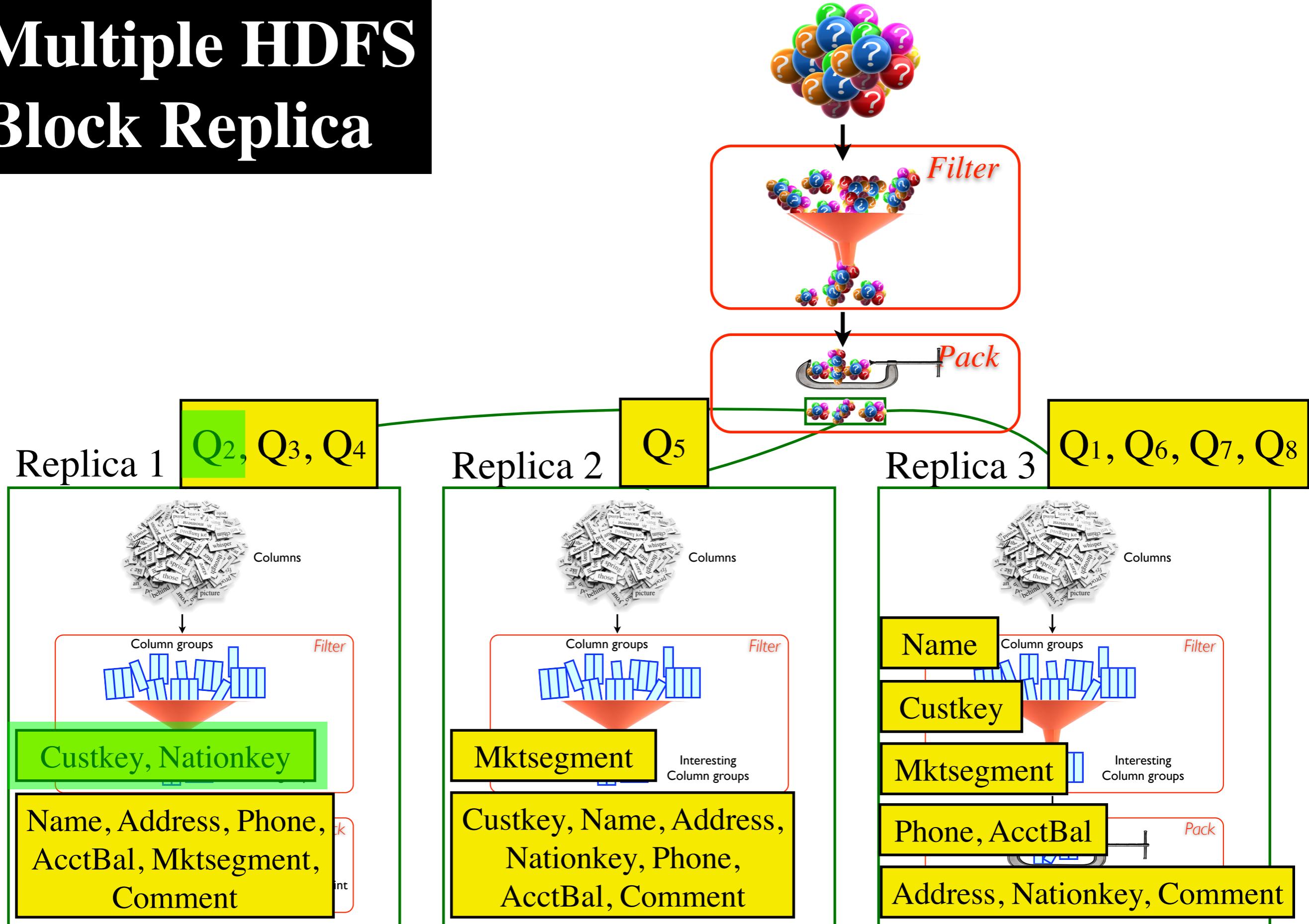
Multiple HDFS Block Replica



Multiple HDFS Block Replica

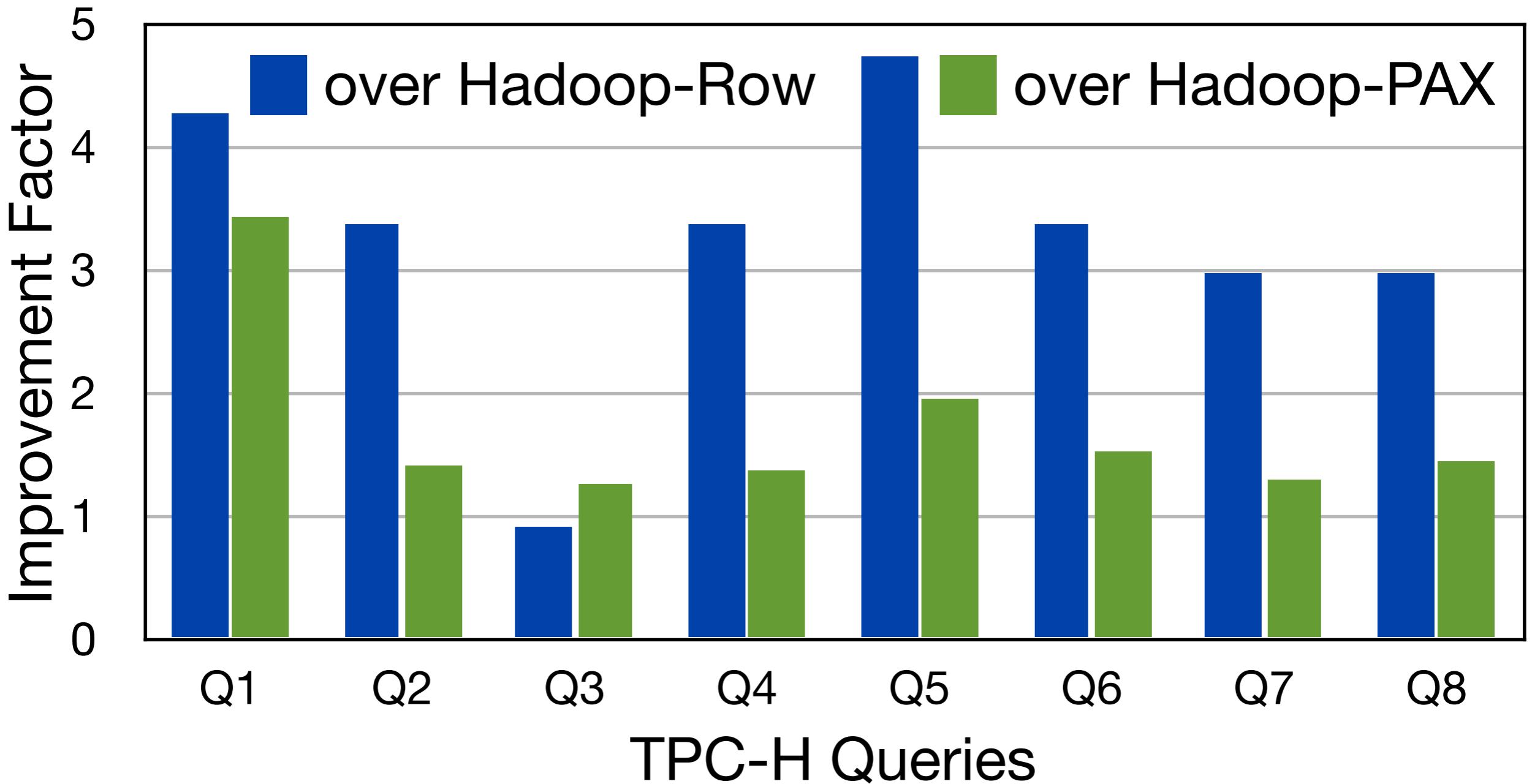


Multiple HDFS Block Replica



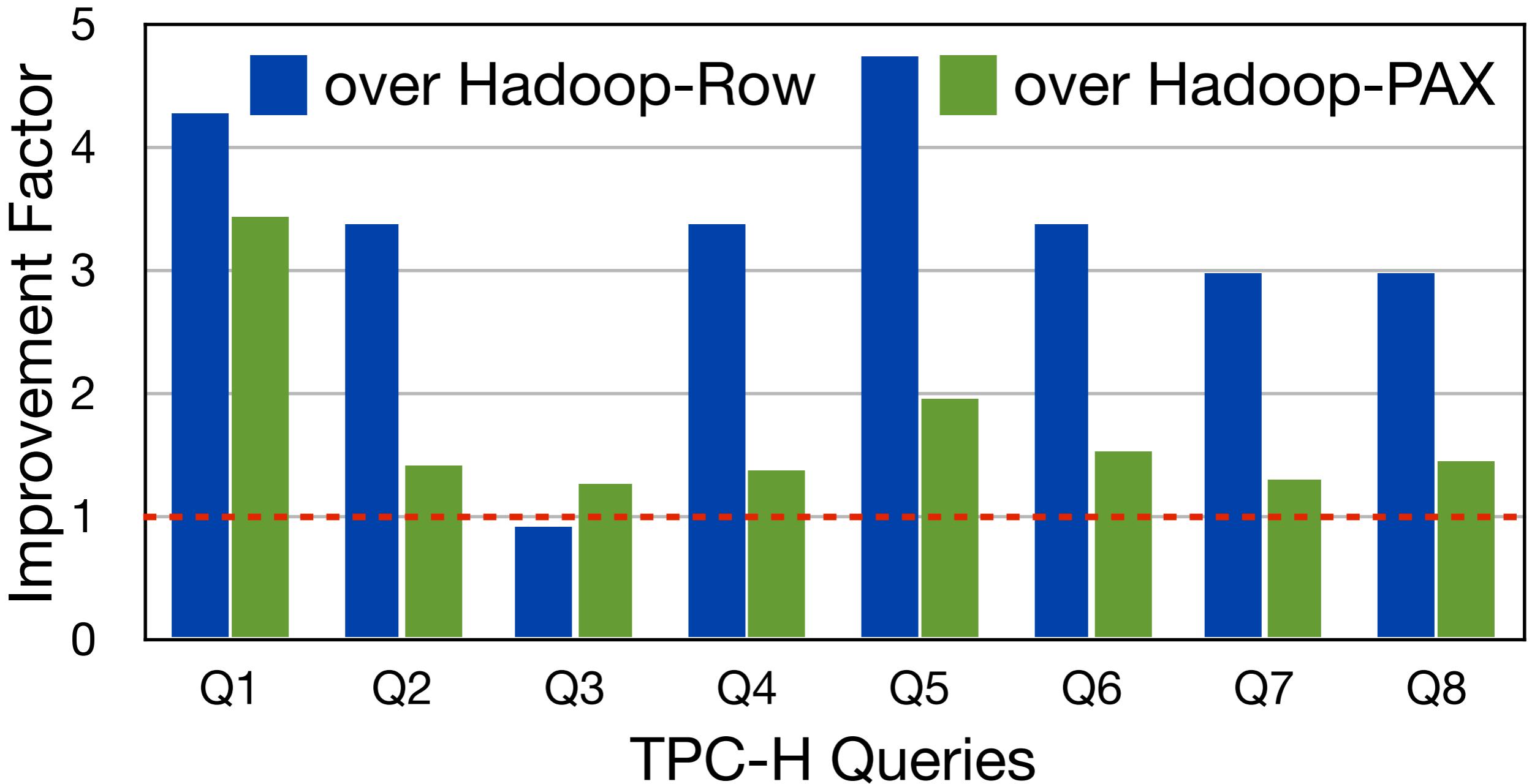
Trojan Data Layouts Results

TPC-H Lineitem



Trojan Data Layouts Results

TPC-H Lineitem



Data Layouts in MapReduce

Initial	2009	2010	2011	2011	2011
Row	CFile	Cheetah	RCFile	CIF	Trojan
Read Unnecessary columns					
	Tuple Reconstruction	Tuple Reconstruction	Tuple Reconstruction	Tuple Reconstruction	Tuple Reconstruction
	High network costs				
		Block level compression			
		Poor I/O Saving	Poor I/O Saving		

Data Layouts in MapReduce

Initial	2009	2010	2011	2011	2011
Row	CFile	Cheetah	RCFile	CIF	Trojan
Read Unnecessary columns					
	Tuple Reconstruction	Tuple Reconstruction	Tuple Reconstruction	Tuple Reconstruction	Tuple Reconstruction
	High network costs				
		Block level compression			
		Poor I/O Saving	Poor I/O Saving		
Single Layout	Single Layout	Single Layout	Single Layout	Single Layout	

Data Layouts in MapReduce

Initial	2009	2010	2011	2011	2011
Row	CFile	Cheetah	RCFile	CIF	Trojan
Read Unnecessary columns					
	Tuple Reconstruction	Tuple Reconstruction	Tuple Reconstruction	Tuple Reconstruction	Tuple Reconstruction
	High network costs				
		Block level compression			
		Poor I/O Saving	Poor I/O Saving		
Single Layout	Single Layout	Single Layout	Single Layout	Single Layout	

Which Layout to Use?

Well...

... it depends on your
query workload

Lessons Learned

Lessons Learned

	Low Record Selectivity	
Low Attribute Selectivity		

Lessons Learned

	Low Record Selectivity	
Low Attribute Selectivity	Row	

Lessons Learned

	Low Record Selectivity	
Low Attribute Selectivity	Row	
Medium Attribute Selectivity		

Lessons Learned

	Low Record Selectivity	
Low Attribute Selectivity	Row	
Medium Attribute Selectivity	Column Groups	

Lessons Learned

	Low Record Selectivity	
Low Attribute Selectivity	Row	
Medium Attribute Selectivity	Column Groups	
High Attribute Selectivity		

Lessons Learned

	Low Record Selectivity	
Low Attribute Selectivity	Row	
Medium Attribute Selectivity	Column Groups	
High Attribute Selectivity	PAX	

Lessons Learned

	Low Record Selectivity	High Record Selectivity
Low Attribute Selectivity	Row	
Medium Attribute Selectivity	Column Groups	
High Attribute Selectivity	PAX	

Lessons Learned

	Low Record Selectivity	High Record Selectivity
Low Attribute Selectivity	Row	Row Groups
Medium Attribute Selectivity	Column Groups	
High Attribute Selectivity	PAX	

Lessons Learned

	Low Record Selectivity	High Record Selectivity
Low Attribute Selectivity	Row	Row Groups
Medium Attribute Selectivity	Column Groups	Row Groups + Column Groups
High Attribute Selectivity	PAX	

Lessons Learned

	Low Record Selectivity	High Record Selectivity
Low Attribute Selectivity	Row	Row Groups
Medium Attribute Selectivity	Column Groups	Row Groups + Column Groups
High Attribute Selectivity	PAX	Row Groups + PAX

MapReduce Intro

Data Layouts

Job Optimization

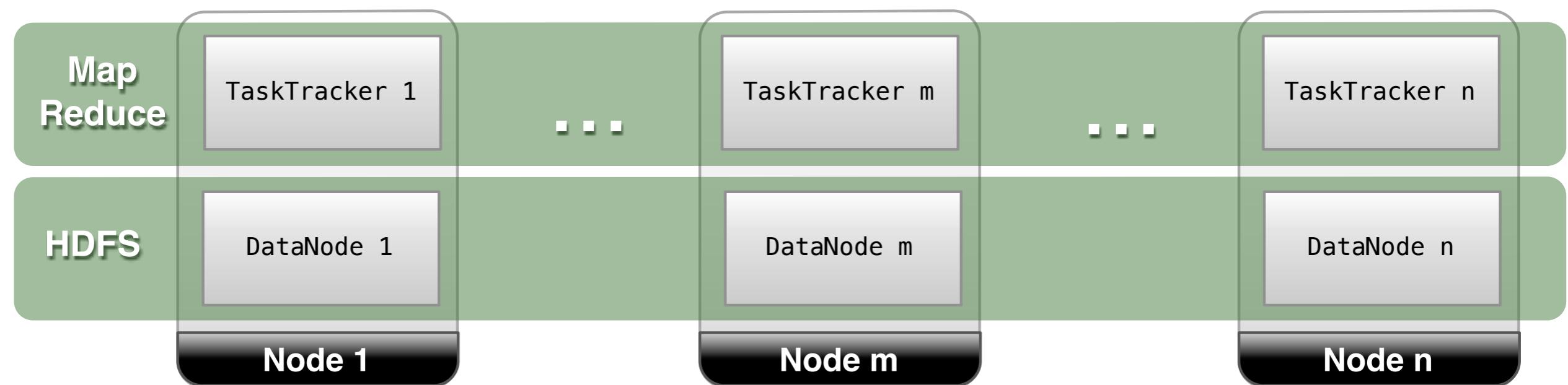
Indexing

Indexing

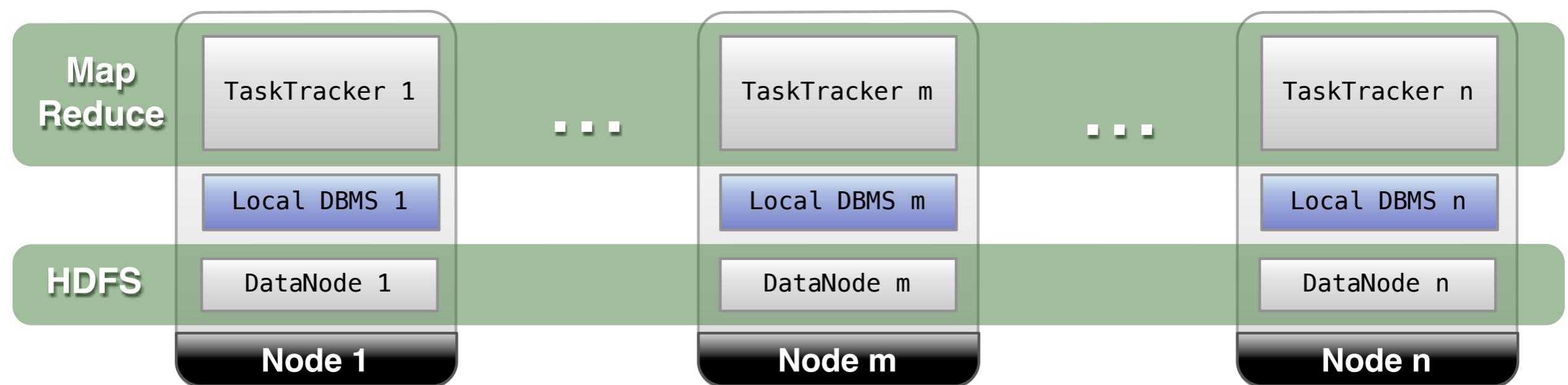
DBMS as Data Storage (HadoopDB)

[A. Abouzeid et al.: HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads. PVLDB 2009]

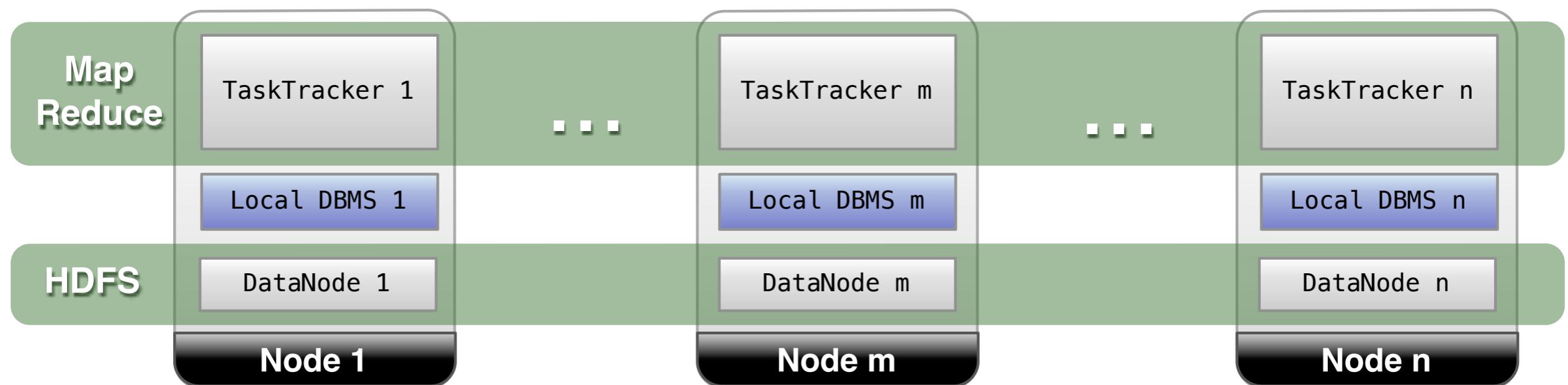
Index Creation



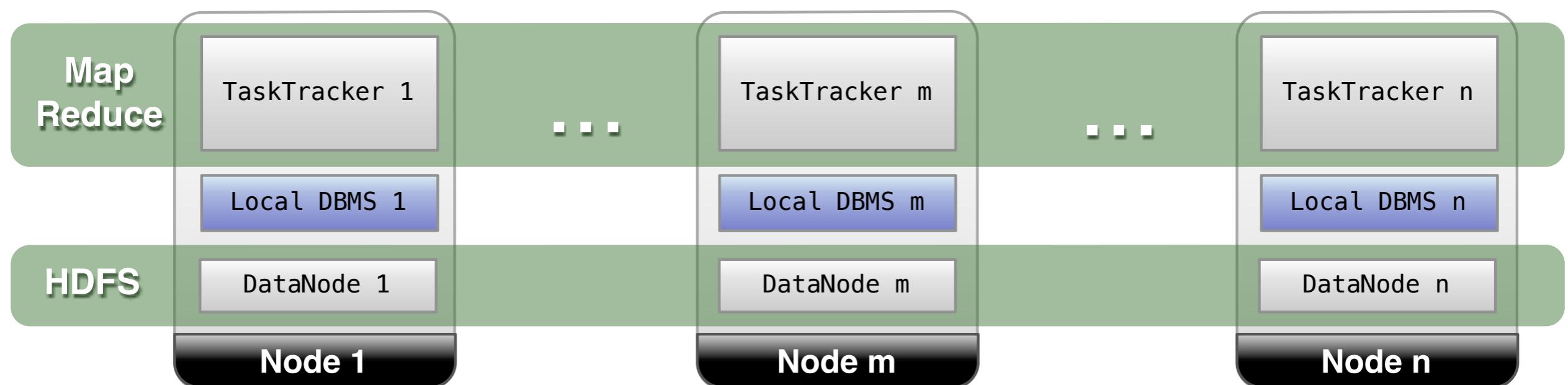
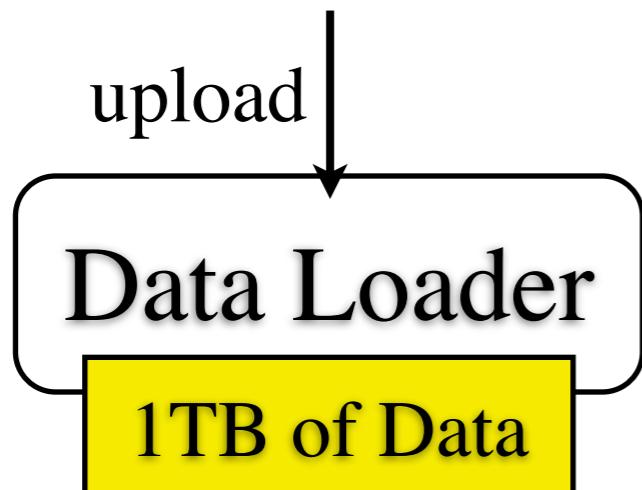
Index Creation



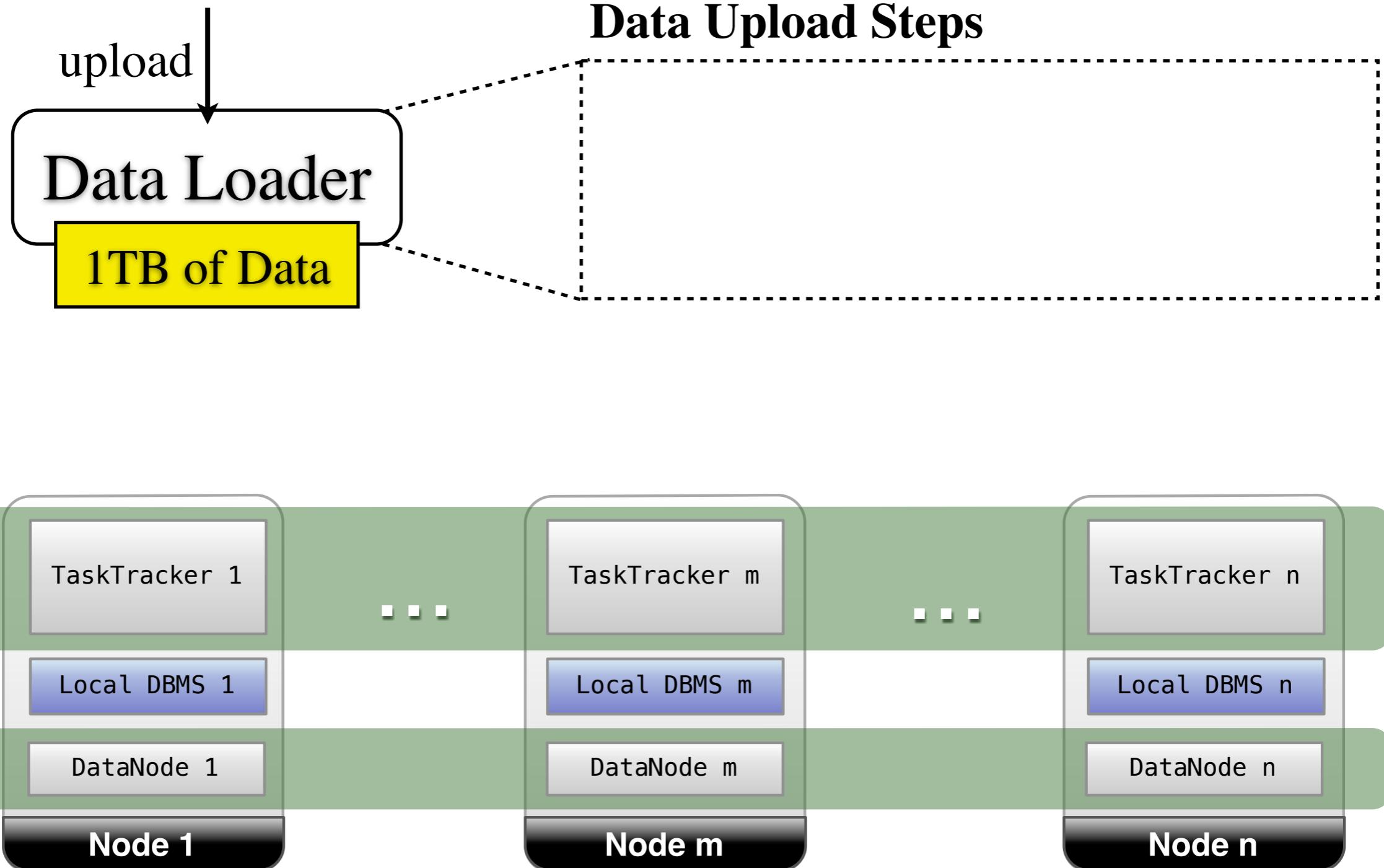
Index Creation



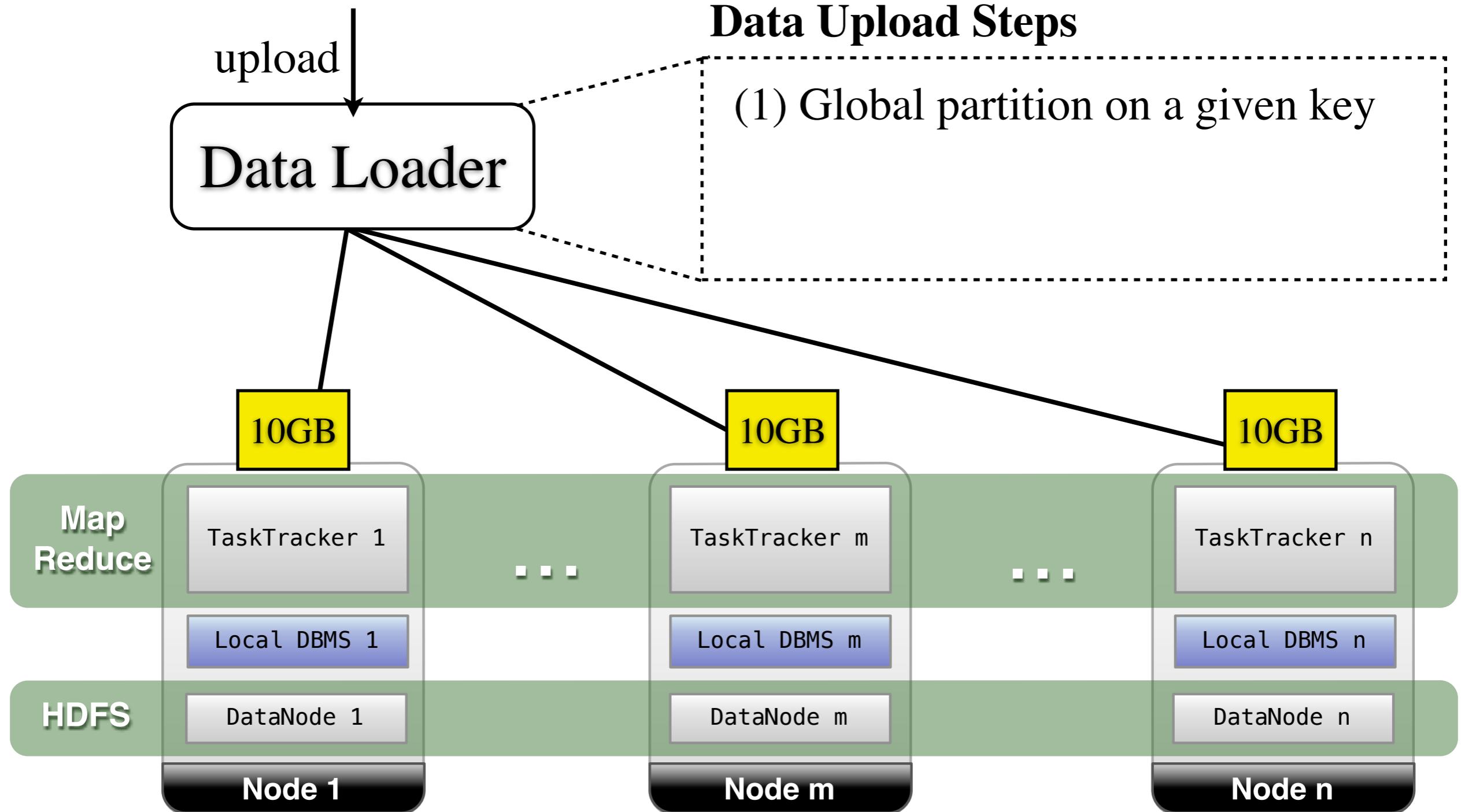
Index Creation



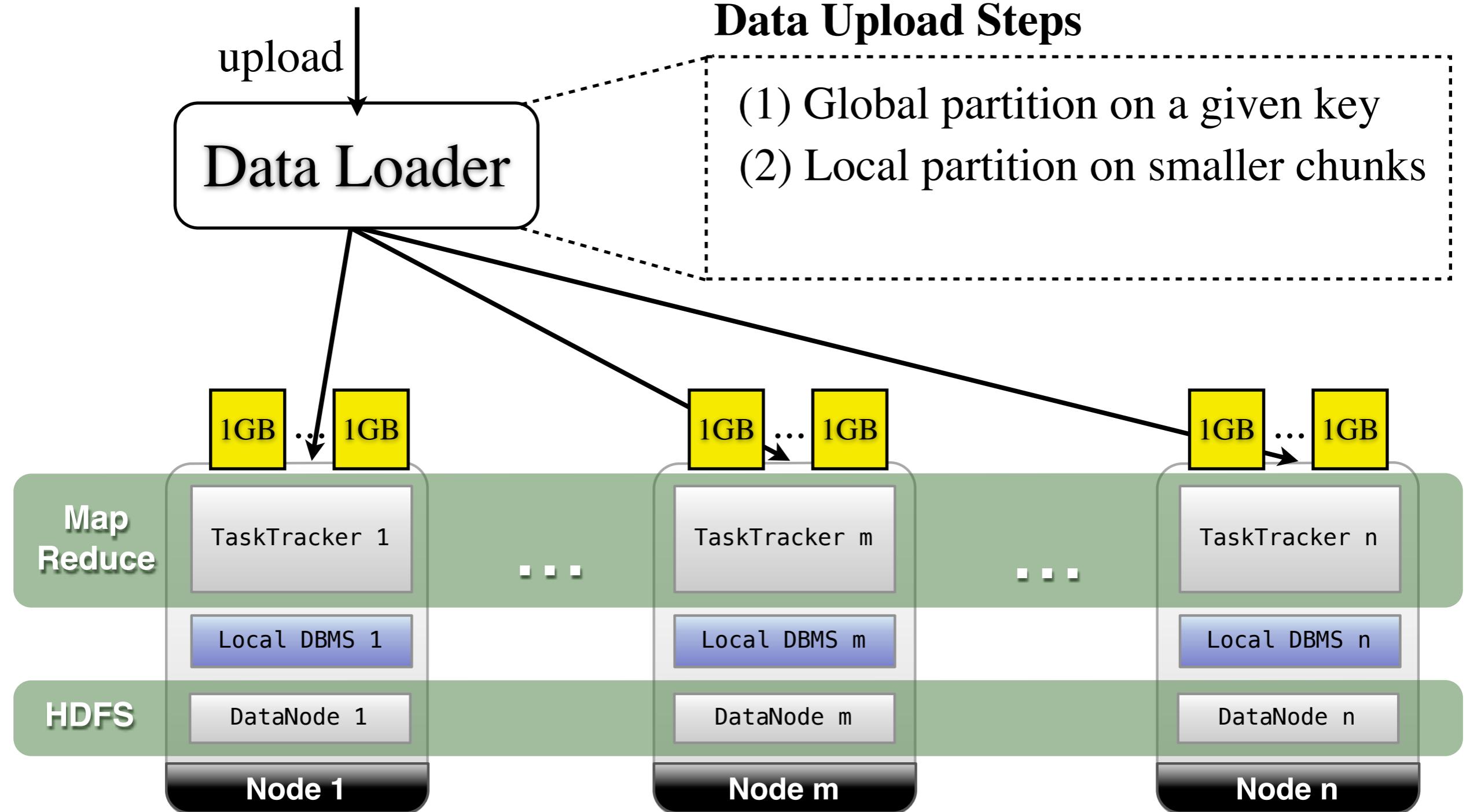
Index Creation



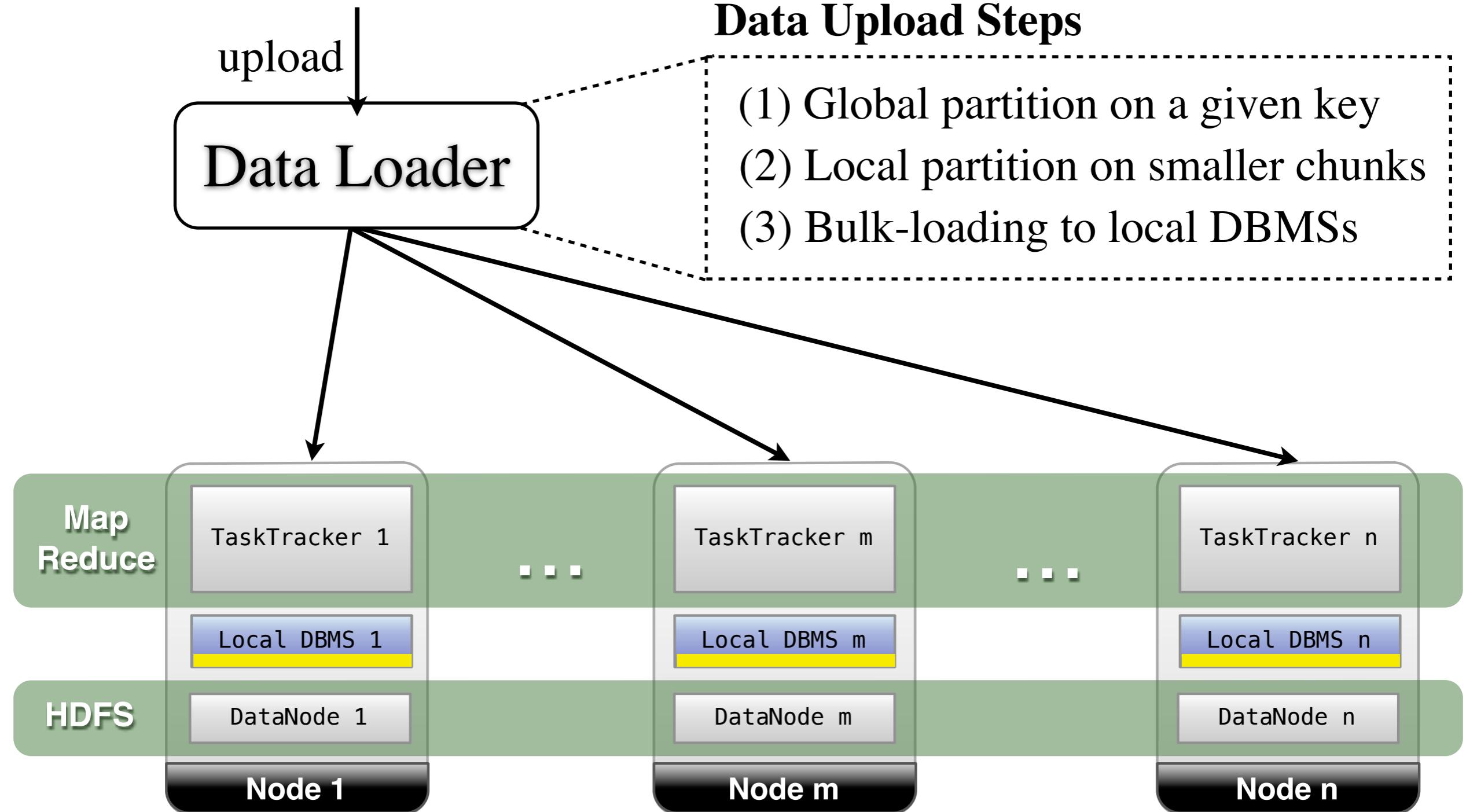
Index Creation



Index Creation



Index Creation

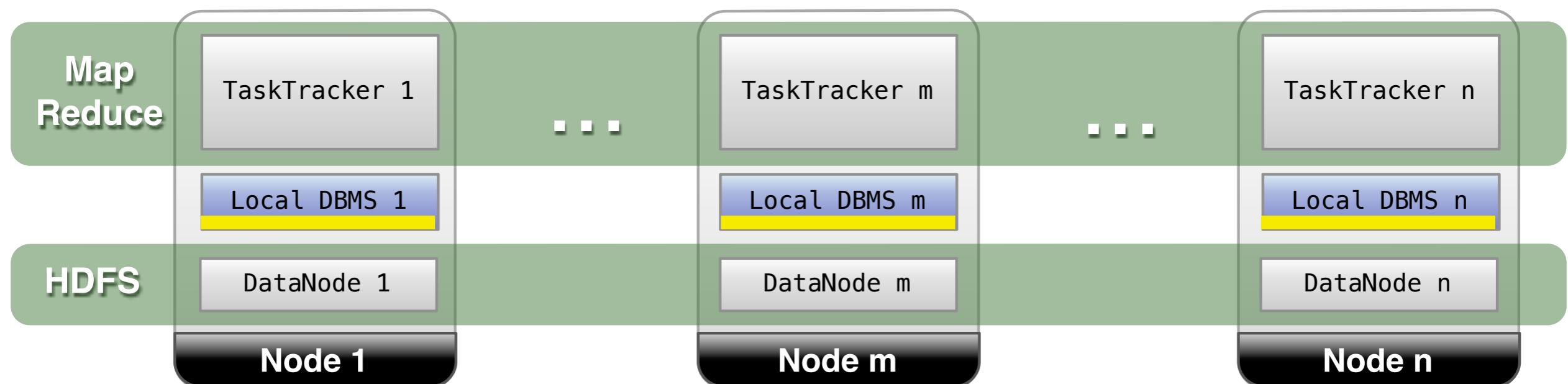


Job Execution



SMS Planner

JobTracker



Job Execution

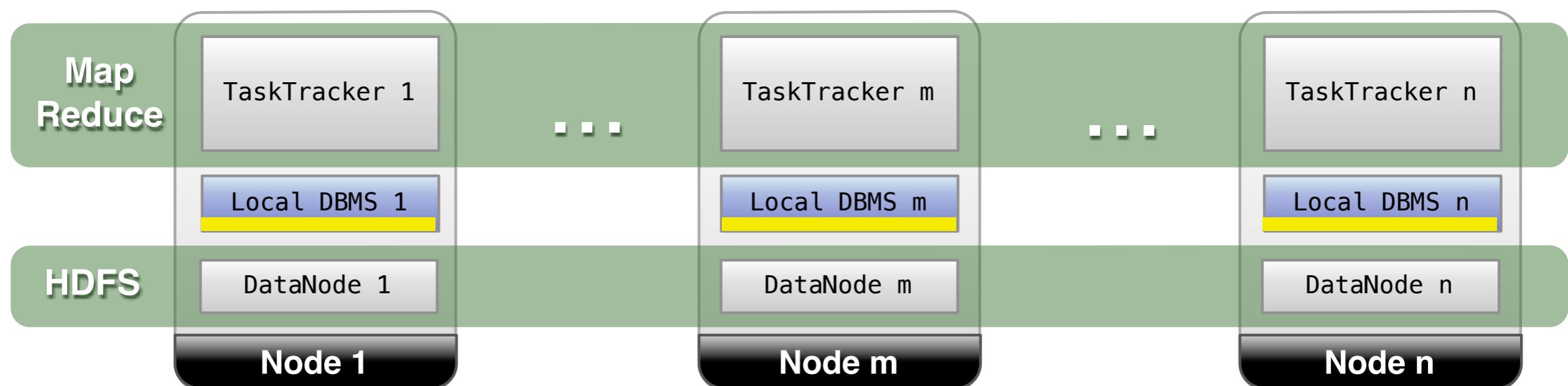


SQL-like
Query

SMS Planner

JobTracker

```
SELECT pageURL, AVG(adRevenue)  
FROM UserVisits  
WHERE sourceIP == 120.115.124.34  
GROUP BY pageURL
```



Job Execution



SQL-like
Query

SMS Planner

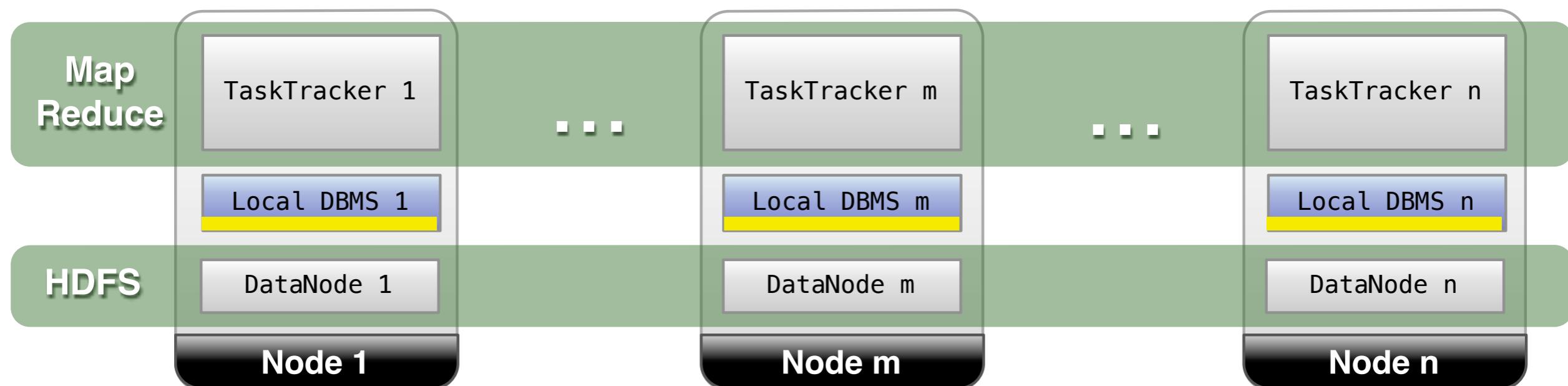
run job

JobTracker

```
SELECT pageURL, AVG(adRevenue)
FROM UserVisits
WHERE sourceIP == 120.115.124.34
GROUP BY pageURL
```

```
map (offset, tuple) {
    if (sourceIP == 120.115.124.34)
        output(pageURL, adRevenue)
}

reduce (pageURL, adRevenue[]) {
    output(pageURL,
          average(adRevenue[])
}
```



Job Execution



SQL-like
Query

SMS Planner

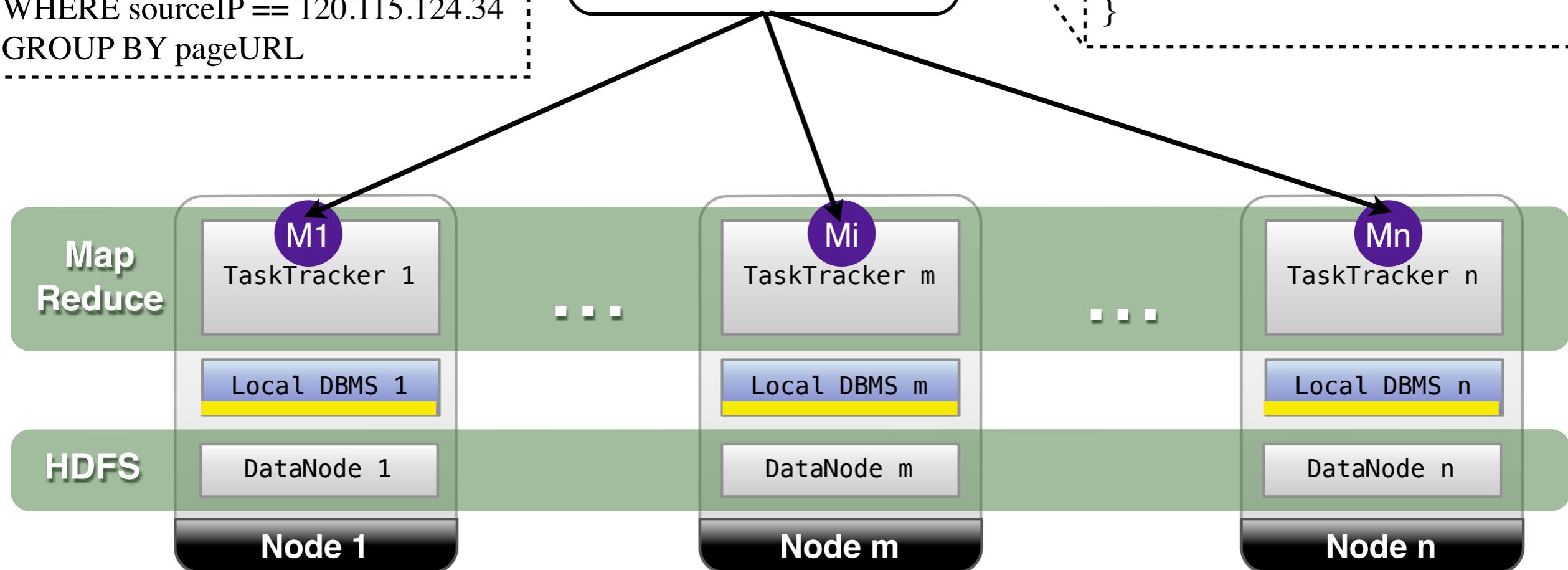
run job

JobTracker

```
SELECT pageURL, AVG(adRevenue)
FROM UserVisits
WHERE sourceIP == 120.115.124.34
GROUP BY pageURL
```

```
map (offset, tuple) {
    if (sourceIP == 120.115.124.34)
        output(pageURL, adRevenue)
}

reduce (pageURL, adRevenue[]) {
    output(pageURL,
          average(adRevenue[])
}
```



Job Execution



SQL-like
Query

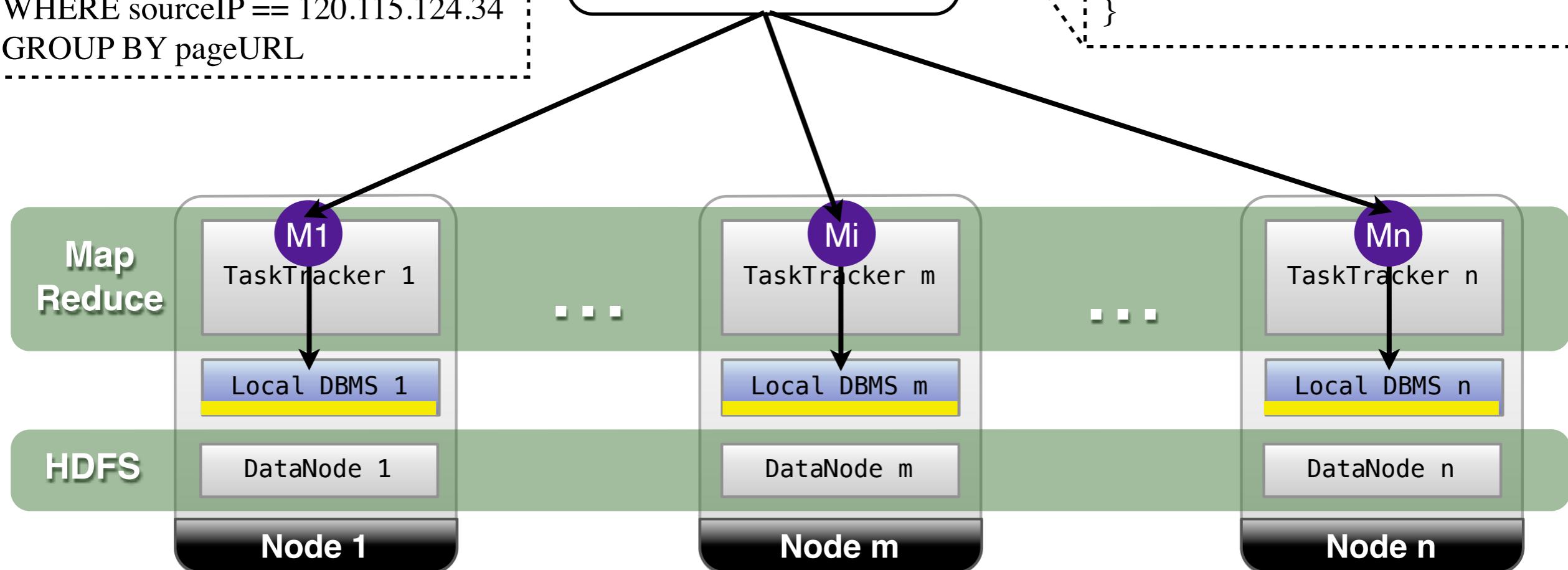
SMS Planner

run job

JobTracker

```
SELECT pageURL, AVG(adRevenue)  
FROM UserVisits  
WHERE sourceIP == 120.115.124.34  
GROUP BY pageURL
```

```
map (offset, tuple) {  
    if (sourceIP == 120.115.124.34)  
        output(pageURL, adRevenue)  
}  
  
reduce (pageURL, adRevenue[]) {  
    output(pageURL,  
          average(adRevenue[]))  
}
```



Job Execution



SQL-like
Query

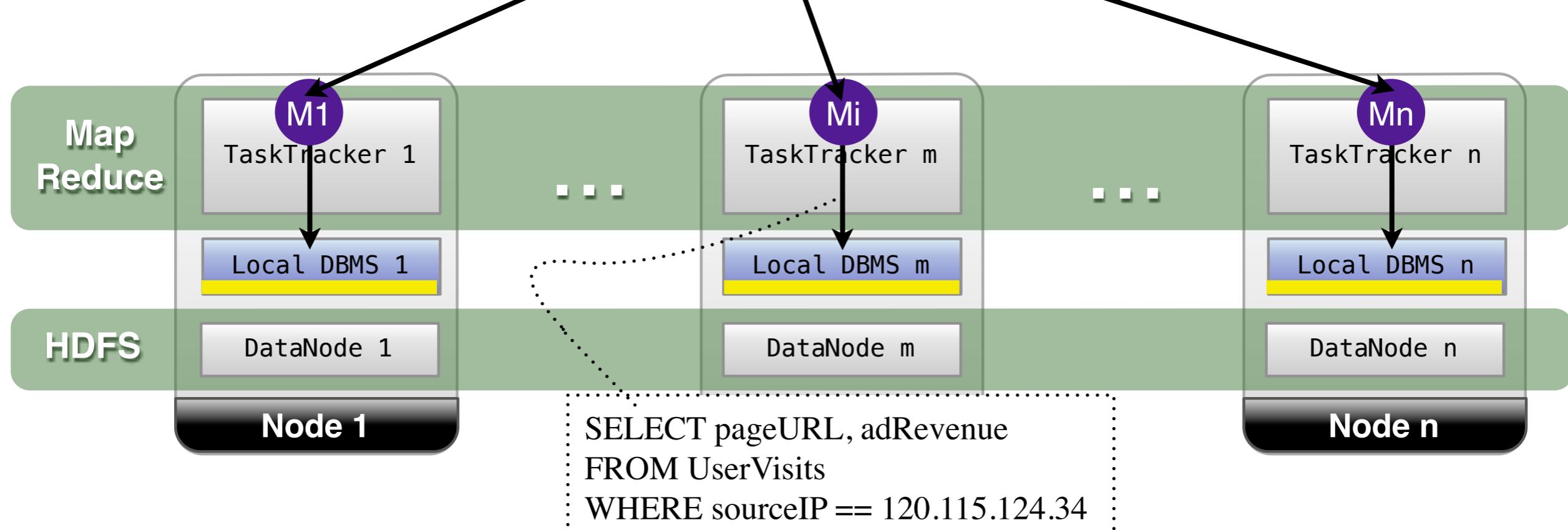
SMS Planner

run job

JobTracker

```
SELECT pageURL, AVG(adRevenue)  
FROM UserVisits  
WHERE sourceIP == 120.115.124.34  
GROUP BY pageURL
```

```
map (offset, tuple) {  
    if (sourceIP == 120.115.124.34)  
        output(pageURL, adRevenue)  
}  
  
reduce (pageURL, adRevenue[]) {  
    output(pageURL,  
          average(adRevenue[]))  
}
```



Job Execution



SQL-like
Query

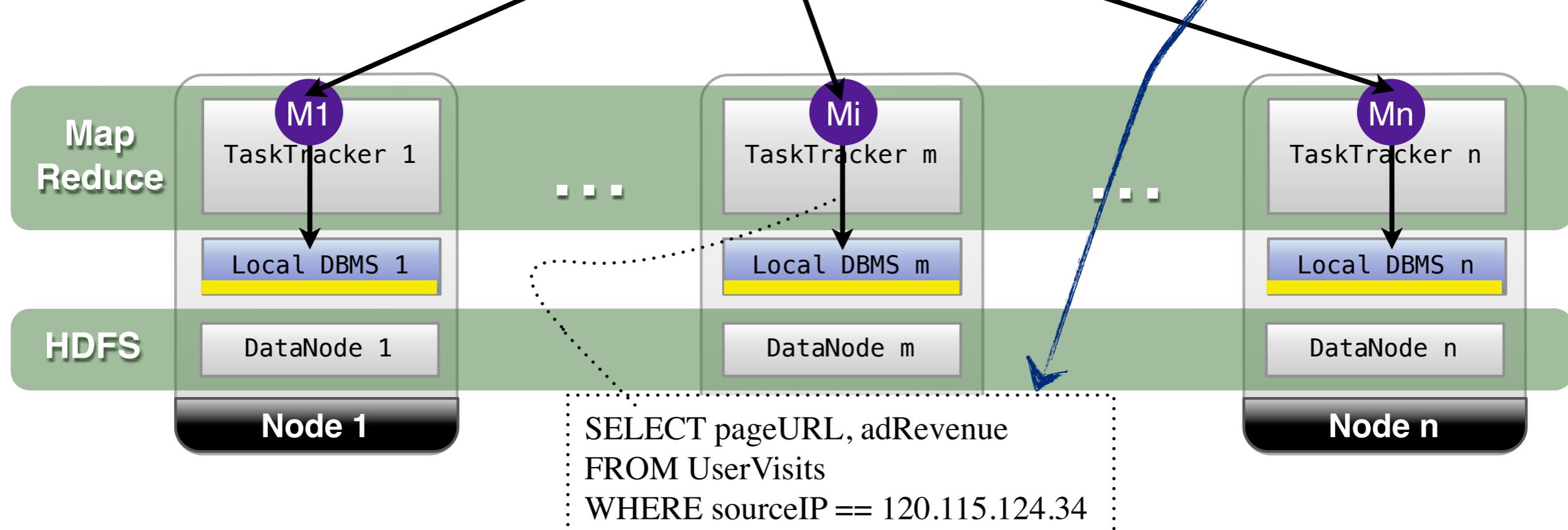
SMS Planner

run job

JobTracker

```
SELECT pageURL, AVG(adRevenue)  
FROM UserVisits  
WHERE sourceIP == 120.115.124.34  
GROUP BY pageURL
```

```
map (offset, tuple) {  
    if (sourceIP == 120.115.124.34)  
        output(pageURL, adRevenue)  
}  
  
reduce (pageURL, adRevenue[]) {  
    output(pageURL,  
          average(adRevenue[]))  
}
```



HadoopDB Results (Selection Task)

Rankings Dataset

pageURL

pageRank

avgDuration

HadoopDB Results (Selection Task)

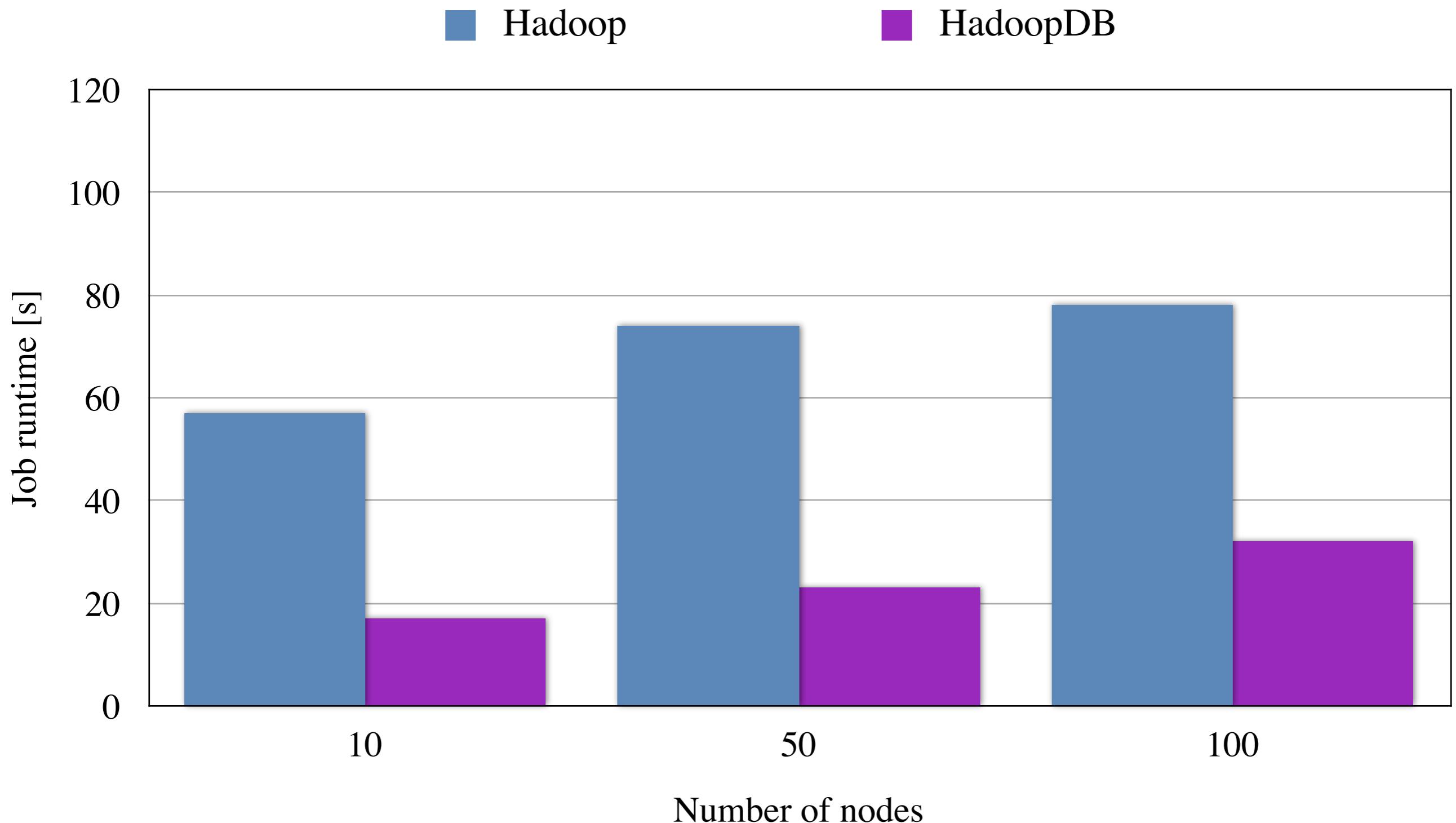
Rankings Dataset

pageURL
pageRank
avgDuration

Query

```
SELECT pageURL, pageRank  
FROM Rankings  
WHERE pageRank > 10
```

HadoopDB Results (Selection Task)



Indexing in MapReduce

Indexing in MapReduce

2009

HadoopDB

Still a database

But...

inside MapReduce?

Indexing Levels

Indexing Levels

- File Level: *filters HDFS Blocks*

Indexing Levels

- File Level: *filters HDFS Blocks*

101.132.121.13, cnn.com, 2011/12/02, 365.98, crisis
102.192.235.245, voici.com, 2011/12/19, 955.83, people
120.115.124.34, vldb.org, 2011/12/03, 296.02, database

⋮

HDFS Block 1

⋮

125.102.135.45, espn.com, 2011/12/01, 123.35, football
123.95.100.24, abc.com, 2011/12/21, 26.02, politics
145.111.145.1, sports.com, 2011/12/20, 630.30, basket

⋮

HDFS Block m

Indexing Levels

Filter Condition:

sourceIP == 120.115.124.34

- File Level: *filters HDFS Blocks*

101.132.121.13, cnn.com, 2011/12/02, 365.98, crisis
102.192.235.245, voici.com, 2011/12/19, 955.83, people
120.115.124.34, vldb.org, 2011/12/03, 296.02, database

⋮

HDFS Block 1

⋮

125.102.135.45, espn.com, 2011/12/01, 123.35, football
123.95.100.24, abc.com, 2011/12/21, 26.02, politics
145.111.145.1, sports.com, 2011/12/20, 630.30, basket

⋮

HDFS Block m

Indexing Levels

Filter Condition:

sourceIP == 120.115.124.34

- File Level: *filters HDFS Blocks*

101.132.121.13, cnn.com, 2011/12/02, 365.98, crisis
102.192.235.245, voici.com, 2011/12/19, 955.83, people
120.115.124.34, vldb.org, 2011/12/03, 296.02, database

⋮

HDFS Block 1

⋮

125.102.135.45, espn.com, 2011/12/01, 123.35, football
123.95.100.24, abc.com, 2011/12/21, 26.02, politics
145.111.145.1, sports.com, 2011/12/20, 630.30, basket

⋮

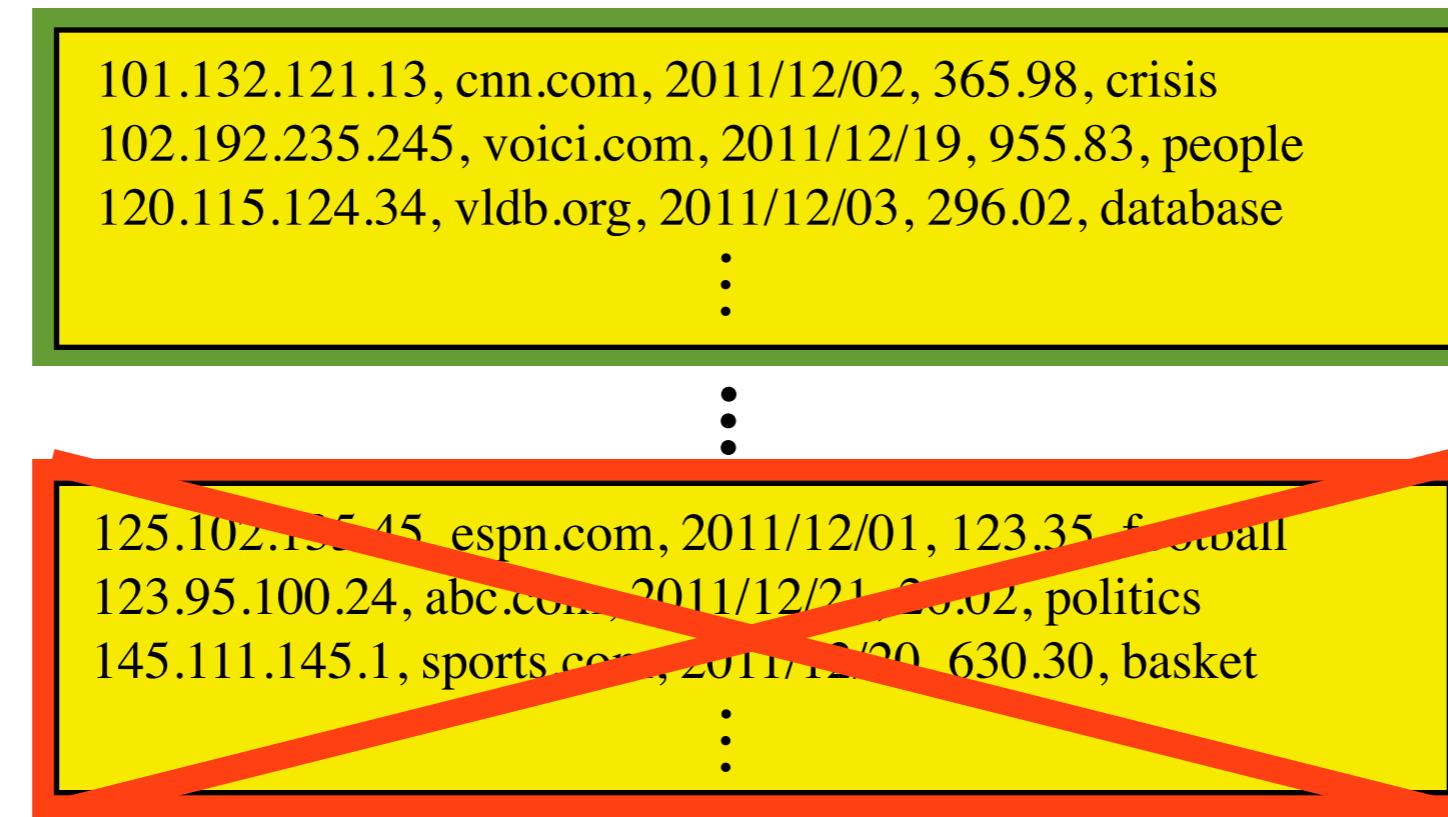
HDFS Block m

Indexing Levels

Filter Condition:

sourceIP == 120.115.124.34

- File Level: *filters HDFS Blocks*

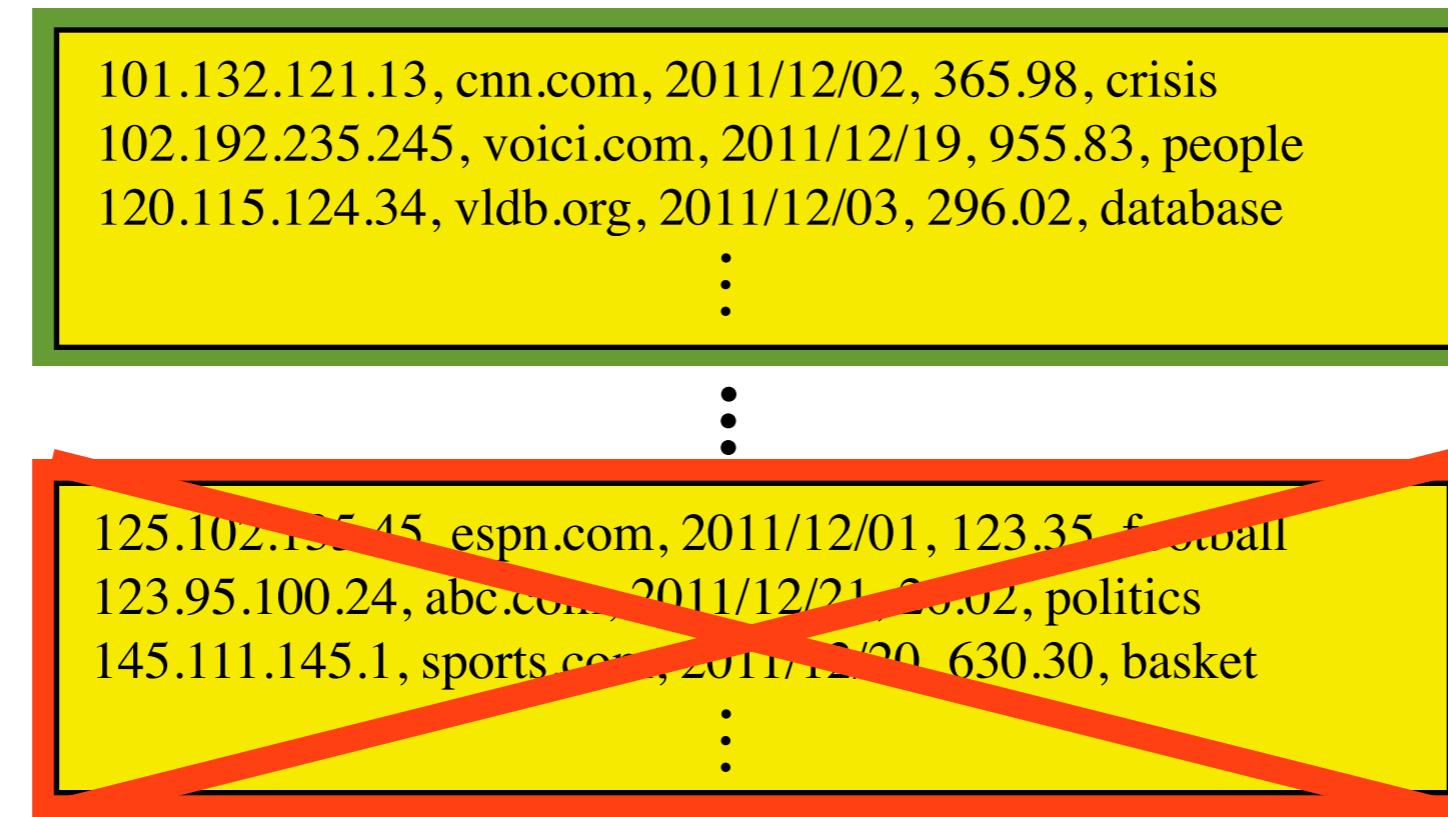


Indexing Levels

Filter Condition:

sourceIP == 120.115.124.34

- File Level: *filters HDFS Blocks*



- Block Level: *filters records*

Indexing Levels

Filter Condition:

sourceIP == 120.115.124.34

- File Level: *filters HDFS Blocks*



HDFS Block 1



HDFS Block m

- Block Level: *filters records*



HDFS Block 1

Indexing Levels

Filter Condition:

sourceIP == 120.115.124.34

- File Level: *filters HDFS Blocks*

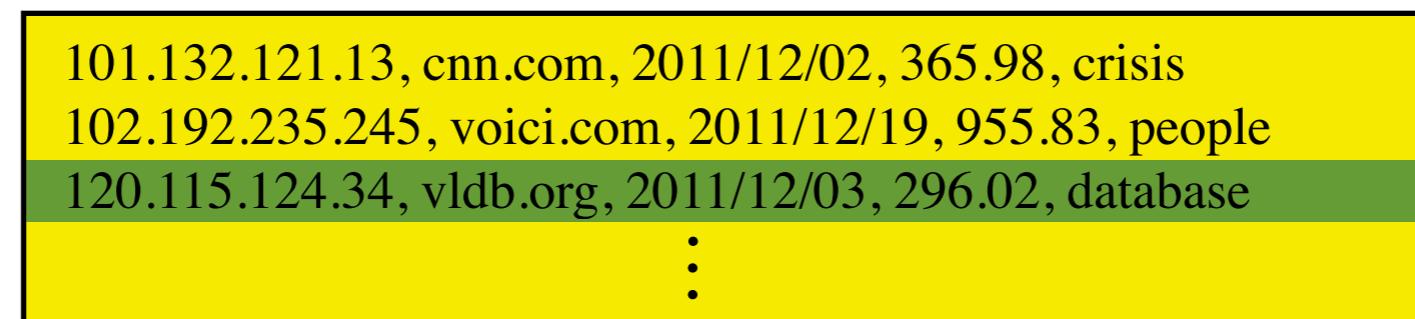


HDFS Block 1



HDFS Block m

- Block Level: *filters records*



HDFS Block 1

File-Level Indexing (Blocks Directory)

[D. Jiang et al.: The Performance of MapReduce: An In-Depth Study. PVLDB
2010]

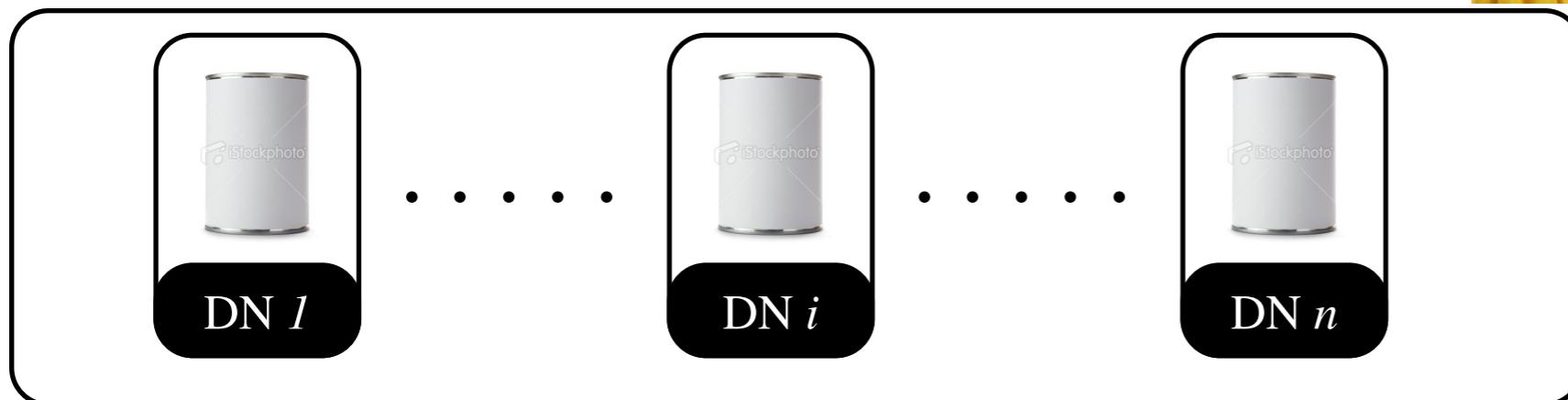
Index Creation

UserVisits sorted on visitDate

```
125.102.135.45, espn.com, 2011/12/01, 123.35, football  
101.132.121.13, cnn.com, 2011/12/02, 365.98, crisis  
120.115.124.34, vldb.org, 2011/12/03, 296.02, database  
:  
120.145.104.14, abc.com, 2011/12/31, 98.63, elections
```



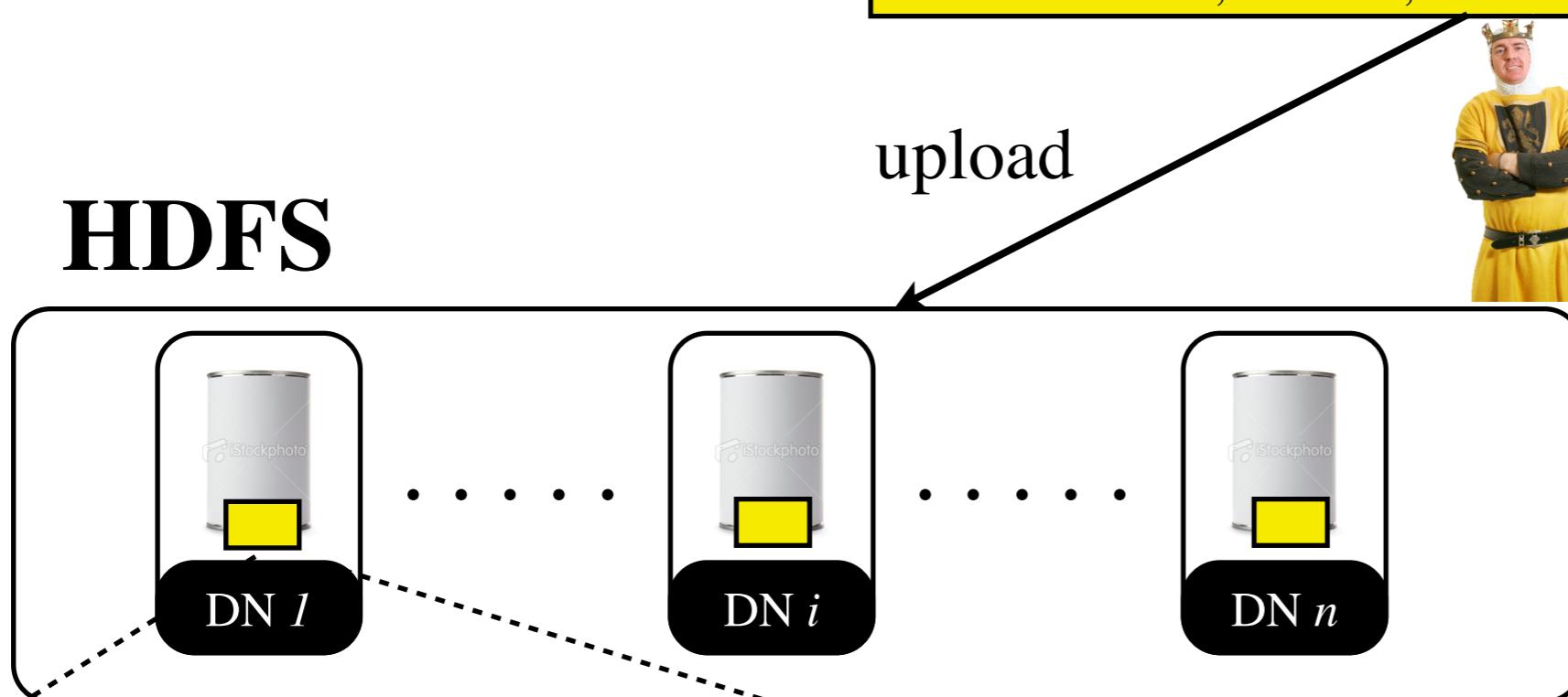
HDFS



Index Creation

UserVisits sorted on visitDate

125.102.135.45, espn.com, 2011/12/01, 123.35, football
101.132.121.13, cnn.com, 2011/12/02, 365.98, crisis
120.115.124.34, vldb.org, 2011/12/03, 296.02, database
⋮
120.145.104.14, abc.com, 2011/12/31, 98.63, elections

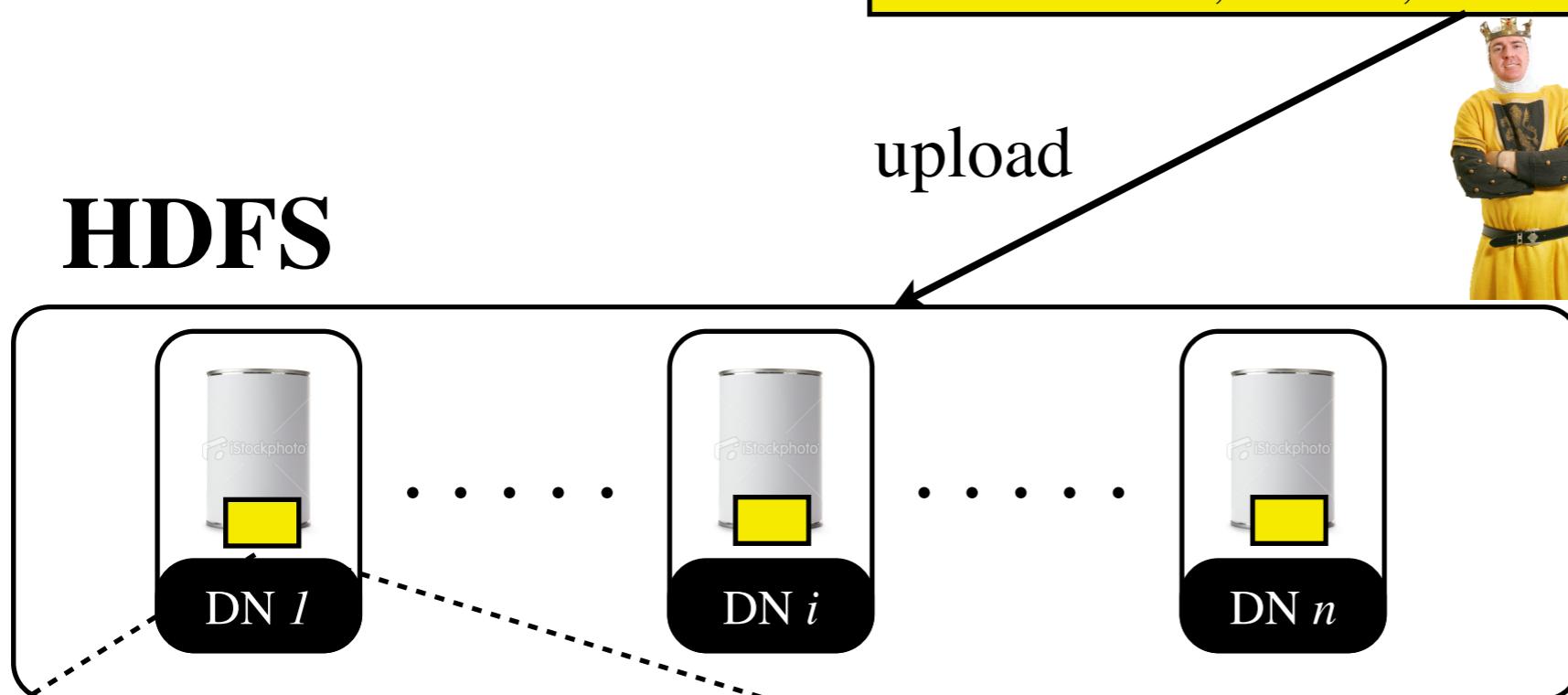


125.102.135.45, espn.com, 2011/12/01, 123.35, football
101.132.121.13, cnn.com, 2011/12/02, 365.98, crisis
120.115.124.34, vldb.org, 2011/12/03, 296.02, database
⋮
120.145.104.14, abc.com, 2011/12/31, 98.63, elections

Index Creation

UserVisits sorted on visitDate

125.102.135.45, espn.com, 2011/12/01, 123.35, football
101.132.121.13, cnn.com, 2011/12/02, 365.98, crisis
120.115.124.34, vldb.org, 2011/12/03, 296.02, database
⋮
120.145.104.14, abc.com, 2011/12/31, 98.63, elections



125.102.135.45, espn.com, 2011/12/01, 123.35, football
101.132.121.13, cnn.com, 2011/12/02, 365.98, crisis
120.115.124.34, vldb.org, 2011/12/03, 296.02, database
⋮
120.145.104.14, abc.com, 2011/12/31, 98.63, elections

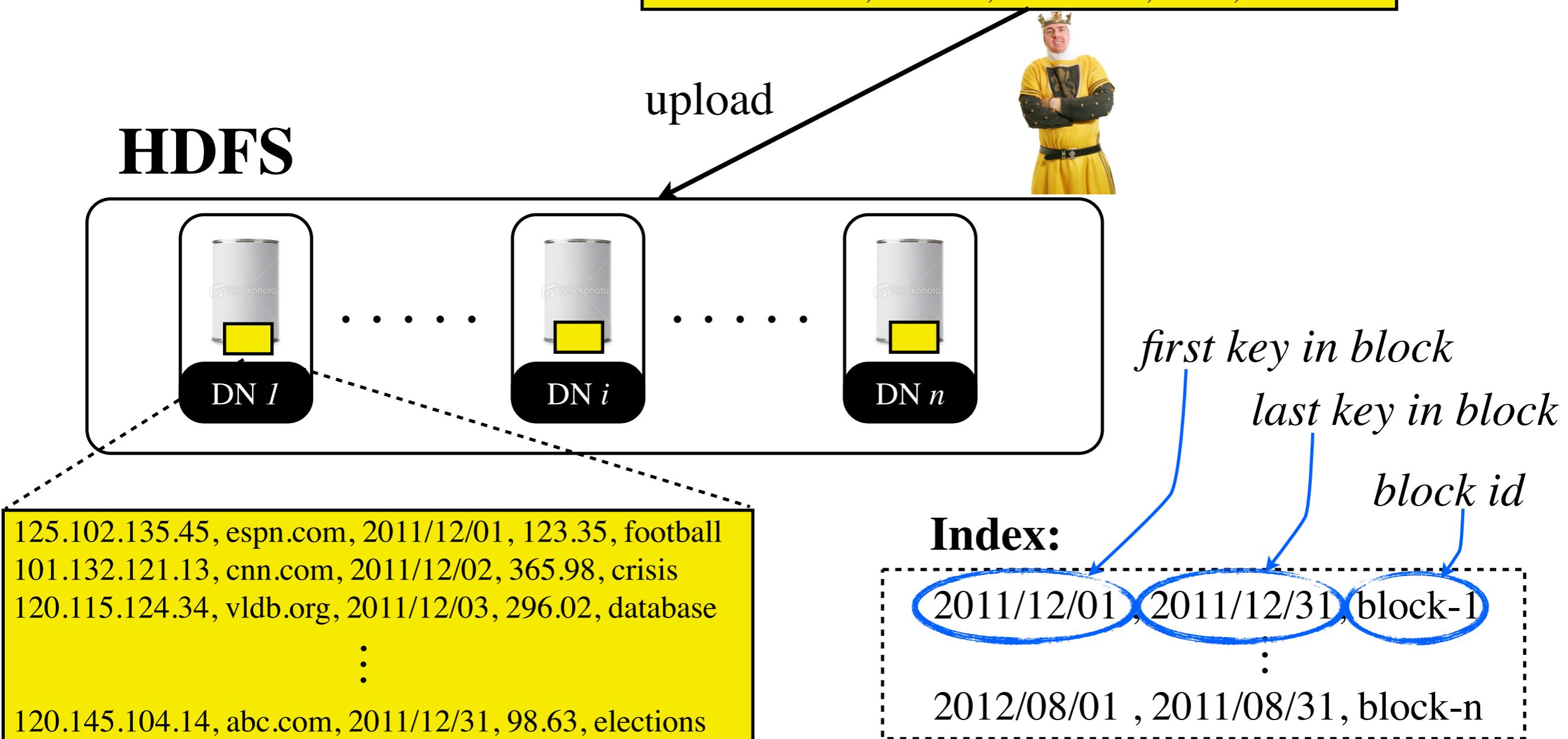
Index:

2011/12/01 , 2011/12/31, block-1
⋮
2012/08/01 , 2011/08/31, block-n

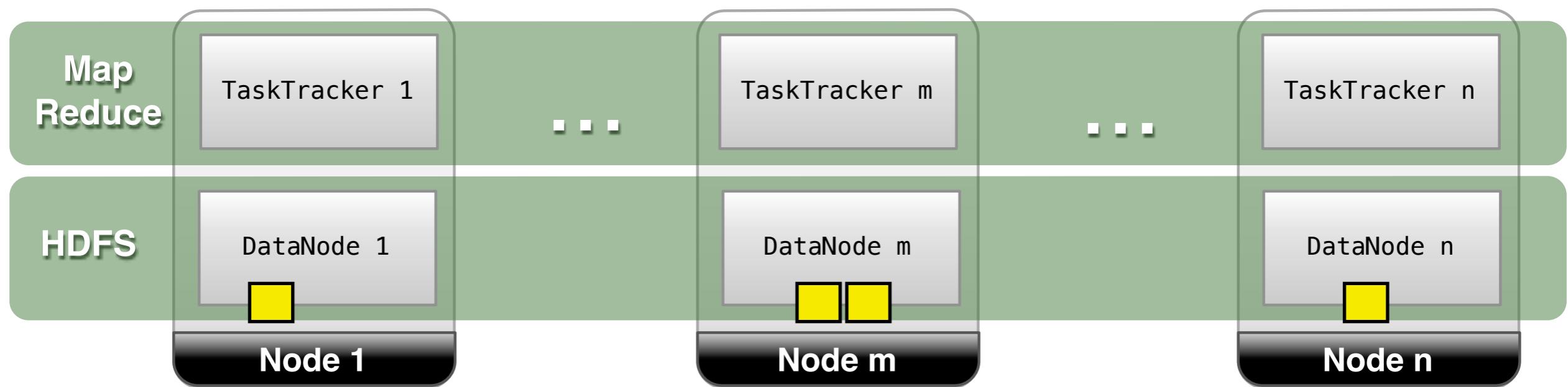
Index Creation

UserVisits sorted on visitDate

125.102.135.45, espn.com, 2011/12/01, 123.35, football
101.132.121.13, cnn.com, 2011/12/02, 365.98, crisis
120.115.124.34, vldb.org, 2011/12/03, 296.02, database
⋮
120.145.104.14, abc.com, 2011/12/31, 98.63, elections

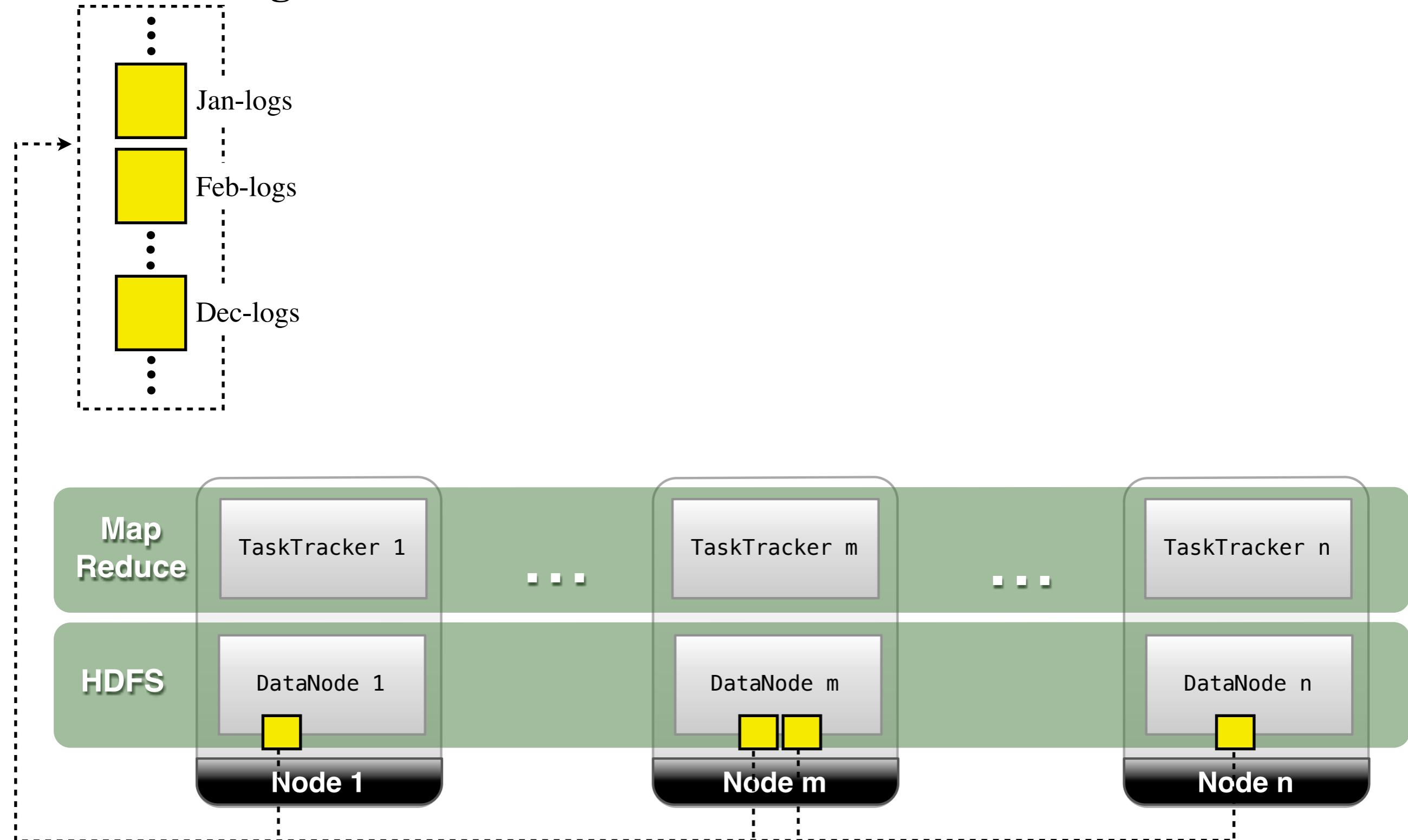


Job Execution



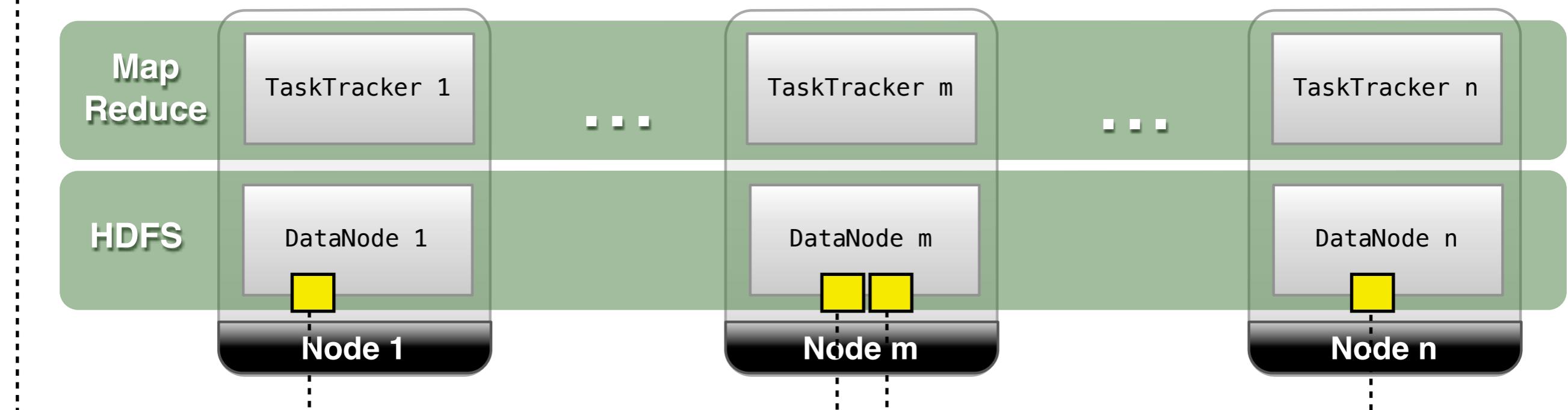
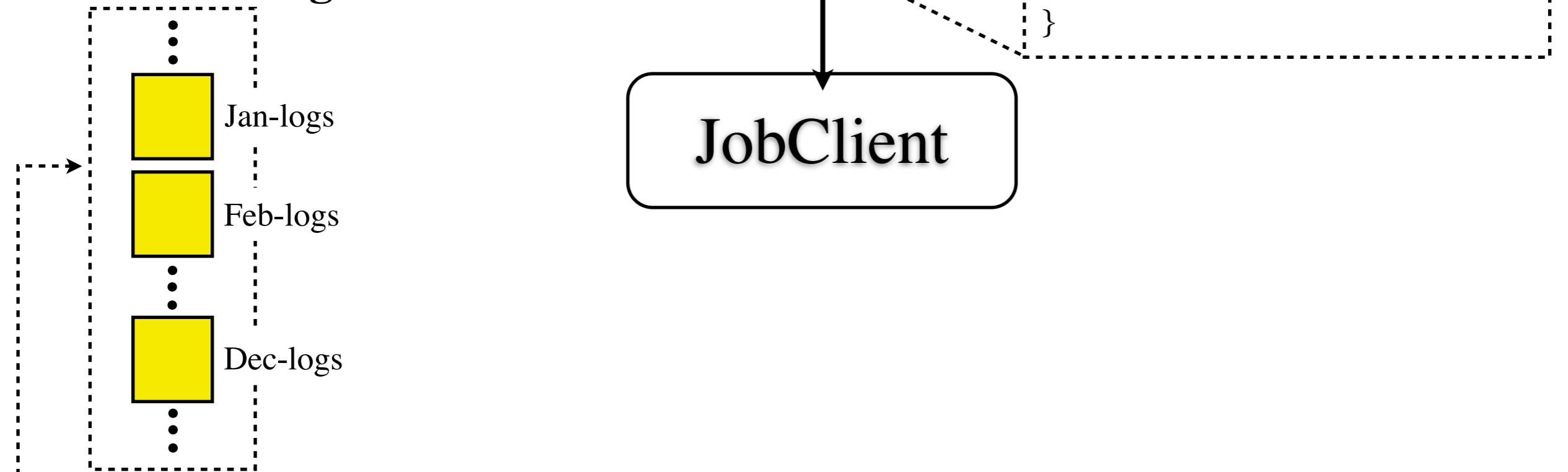
Job Execution

UserVisits Log



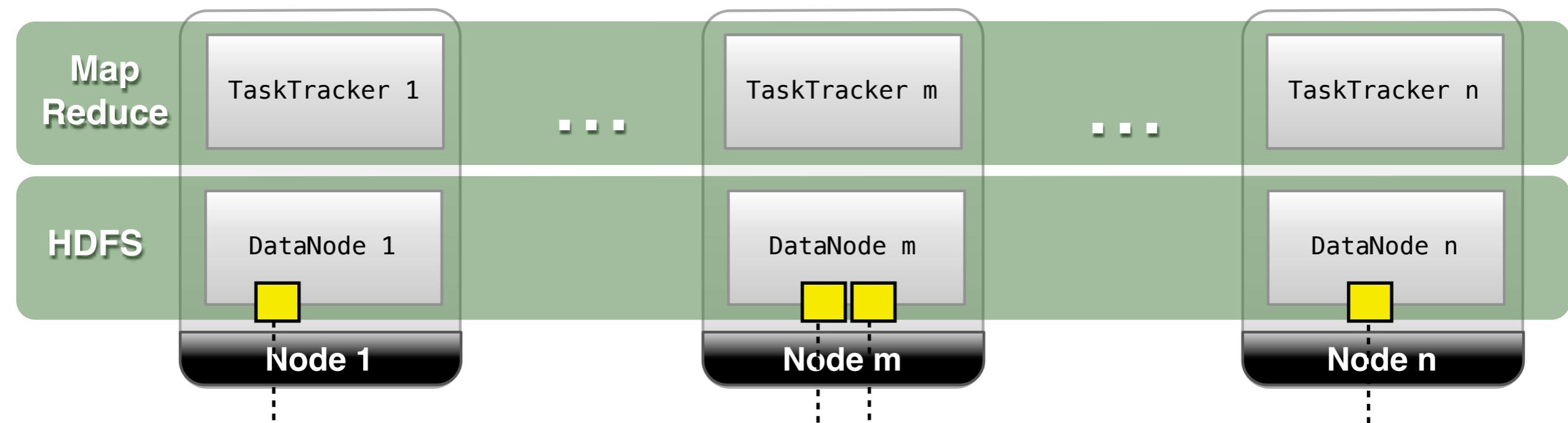
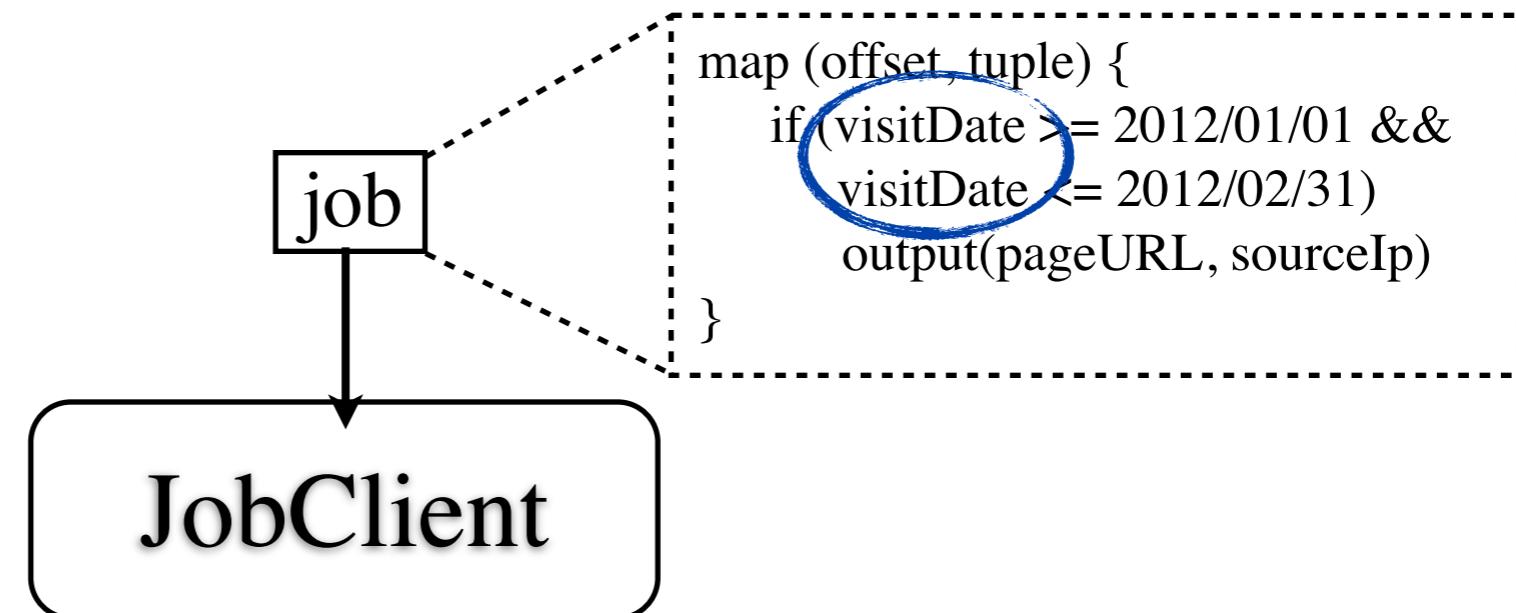
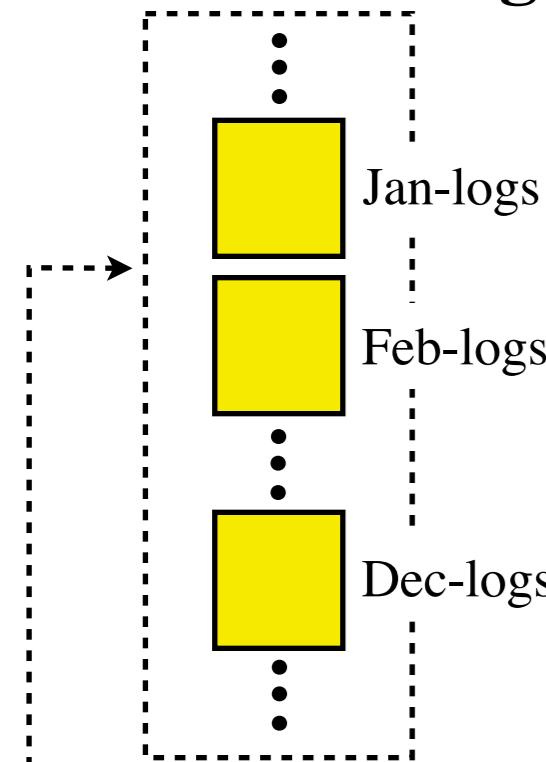
Job Execution

UserVisits Log



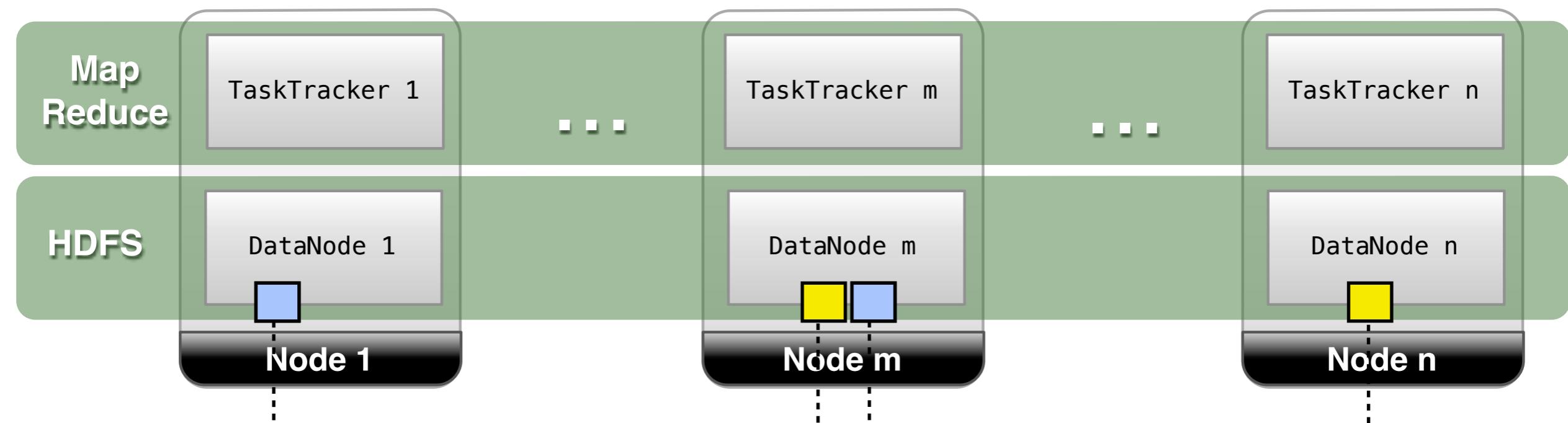
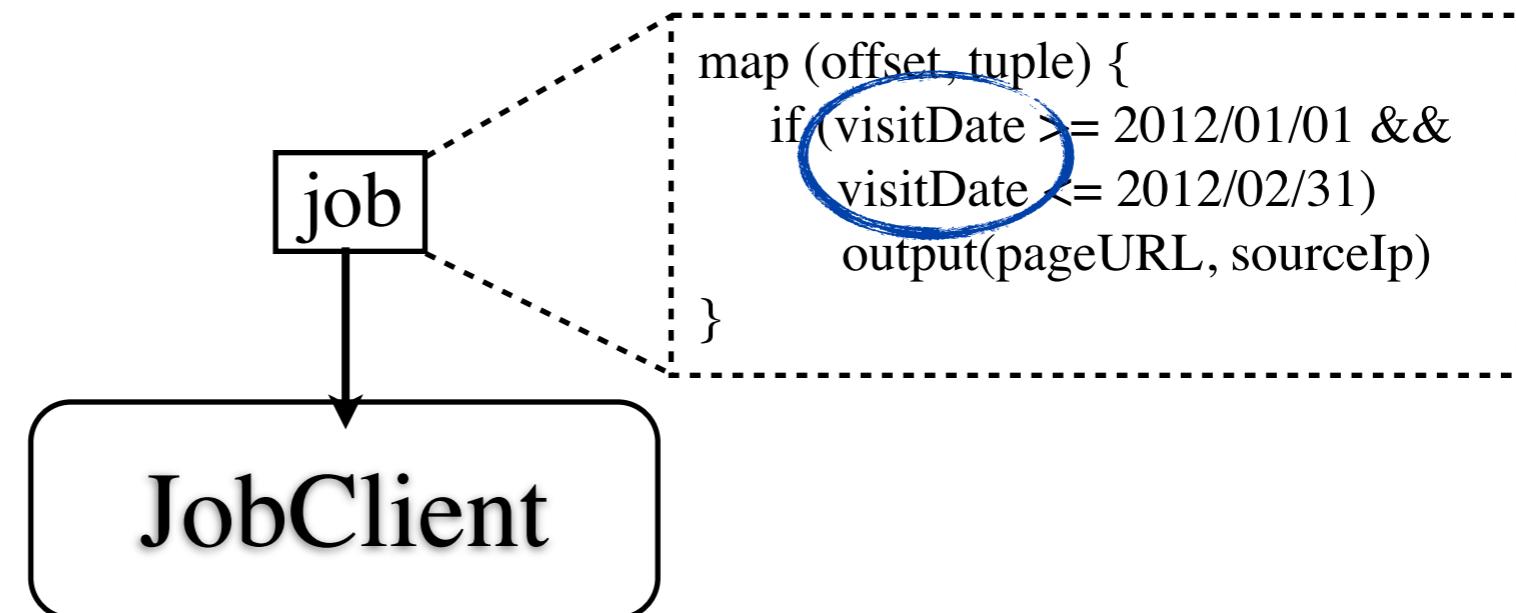
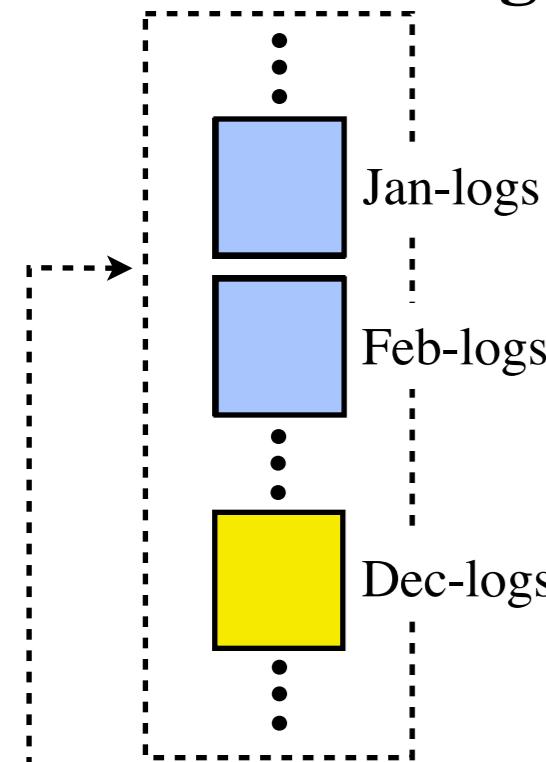
Job Execution

UserVisits Log



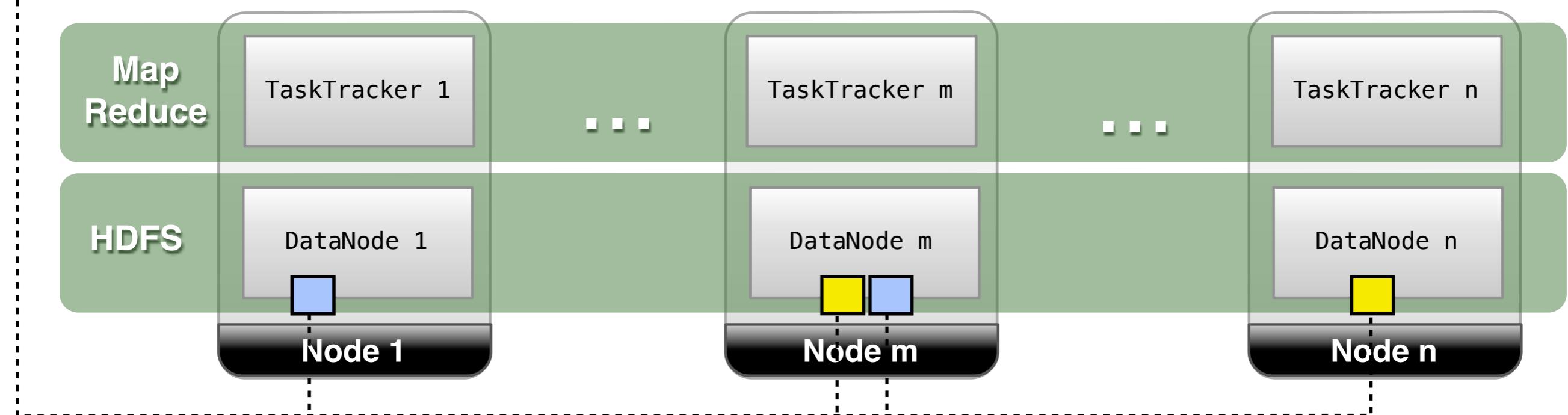
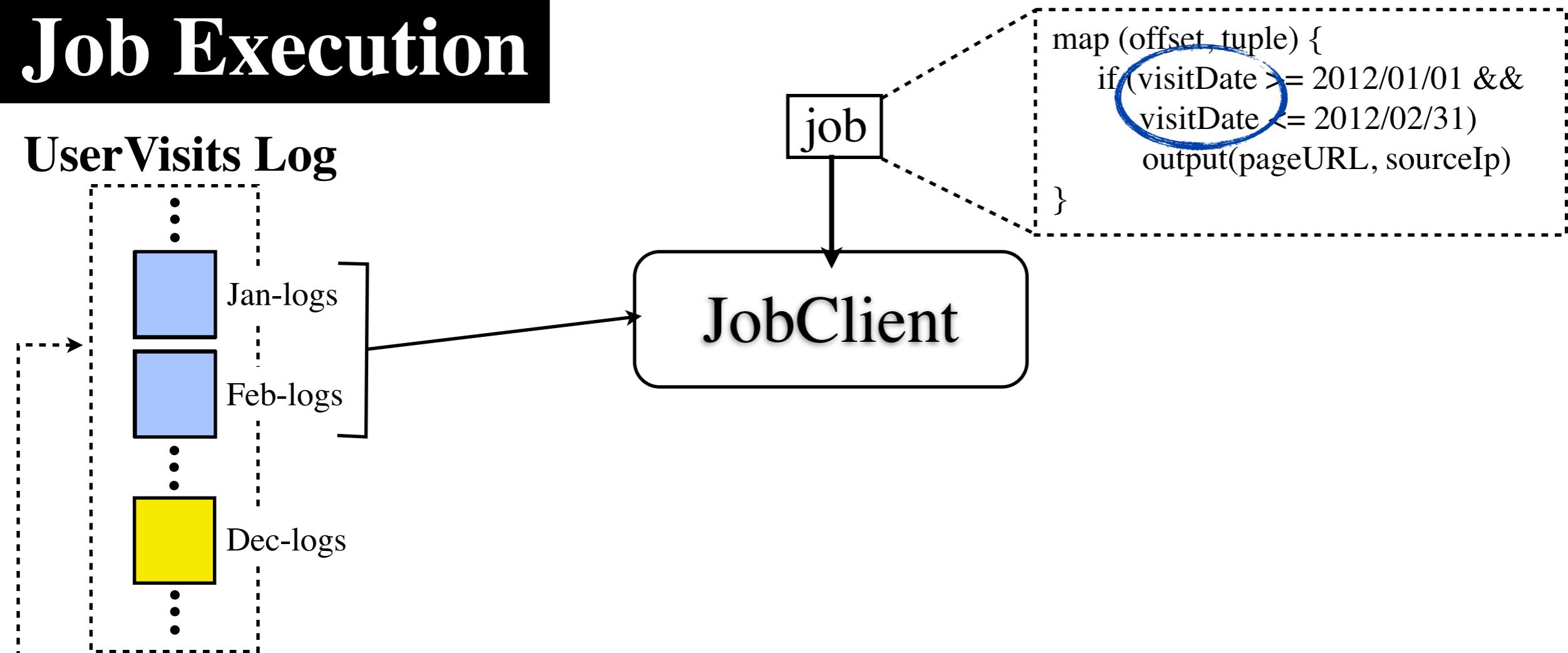
Job Execution

UserVisits Log



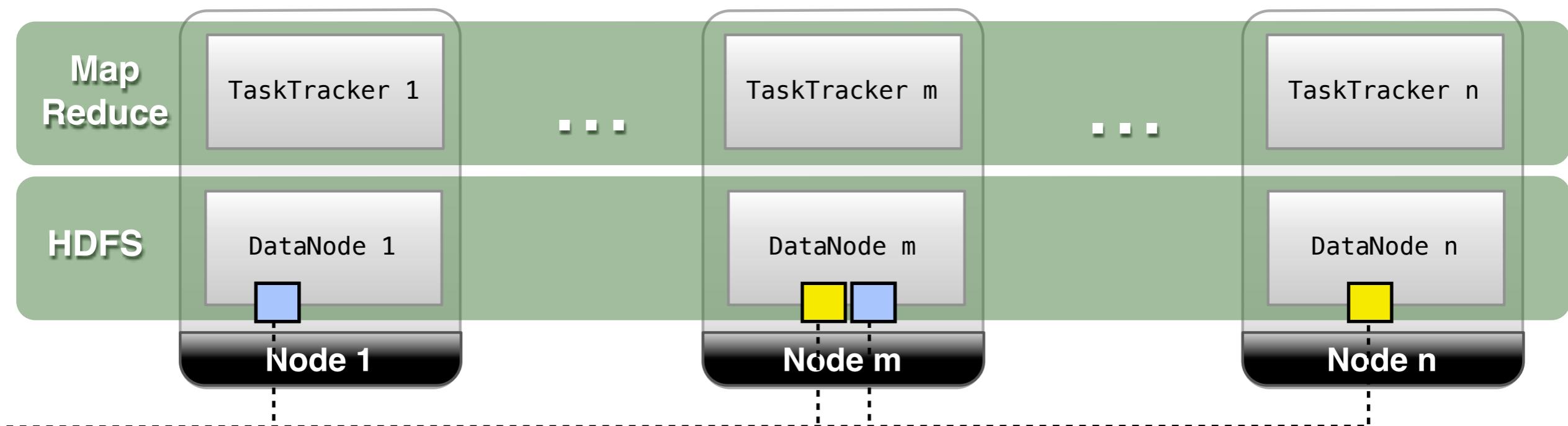
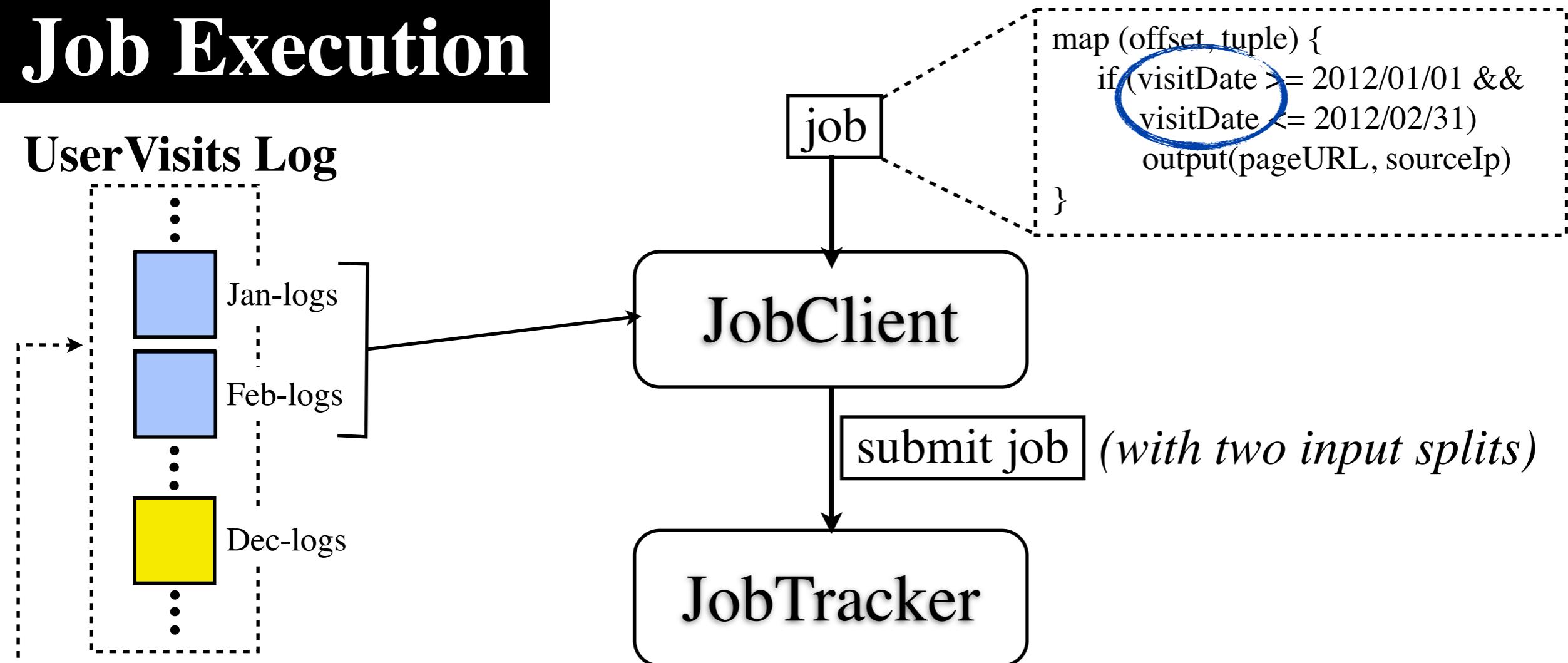
Job Execution

UserVisits Log



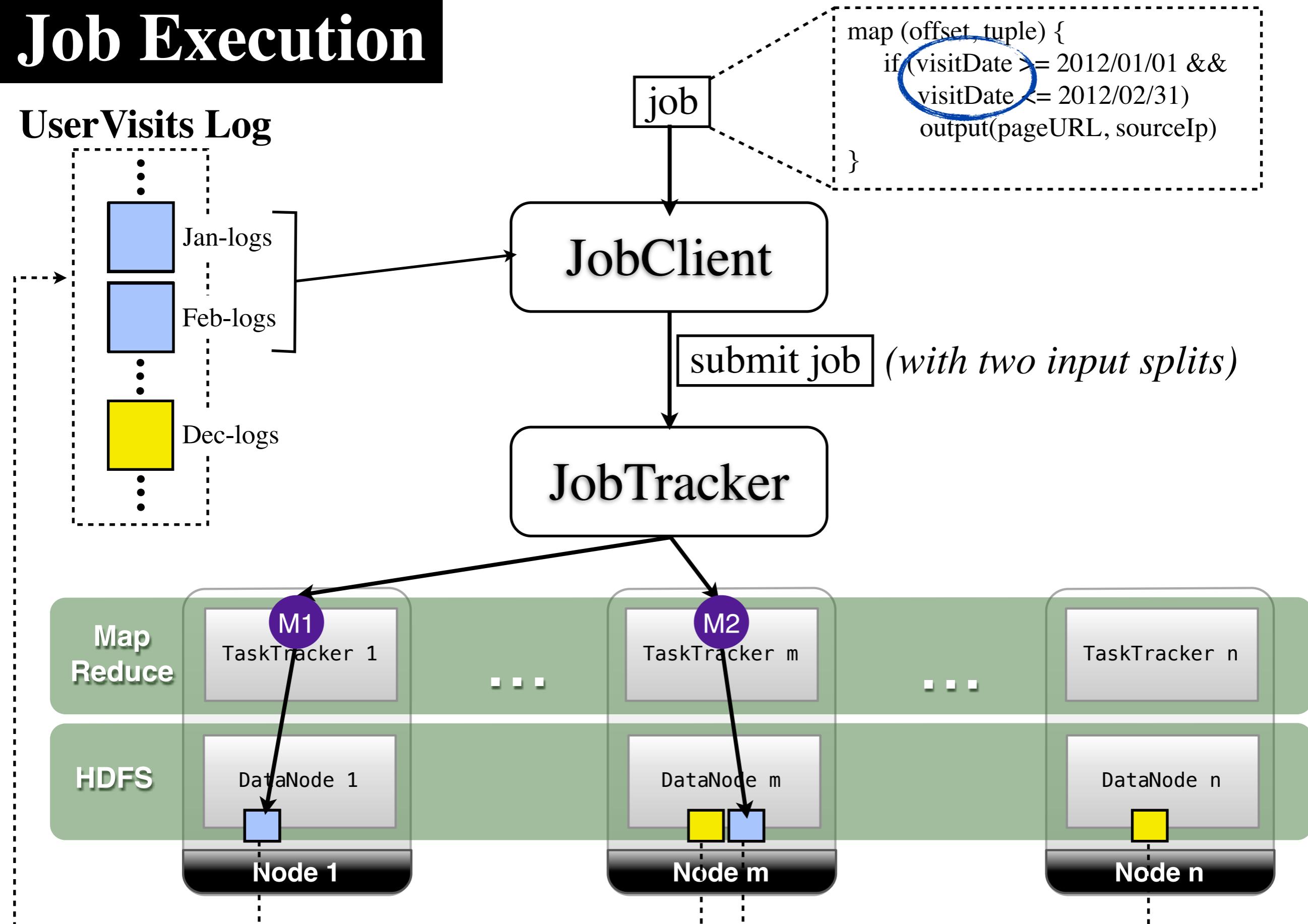
Job Execution

UserVisits Log

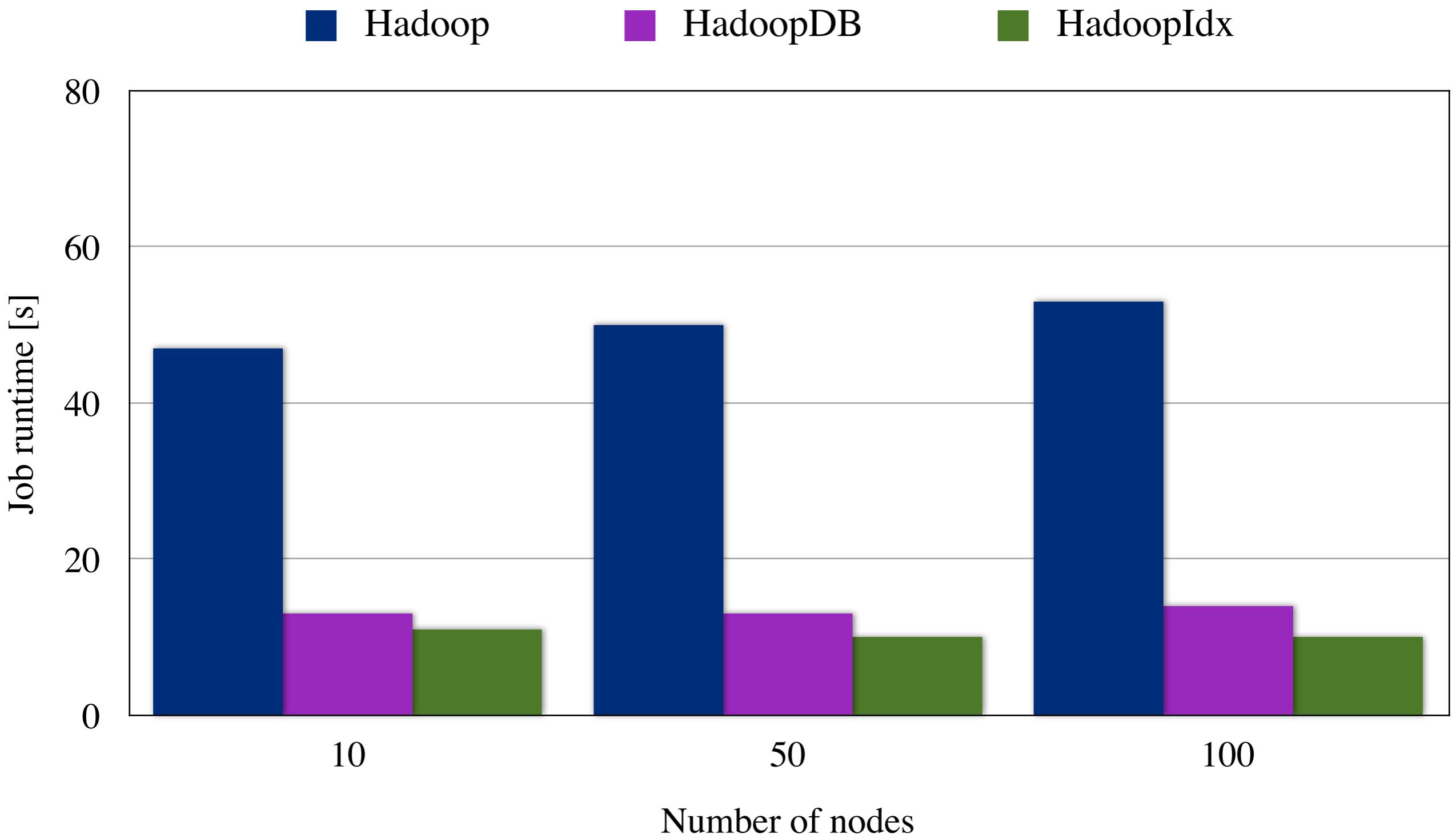


Job Execution

UserVisits Log



File-Level Indexing Results (Selection Task)



Indexing in MapReduce

2009

HadoopDB

Still a database

Indexing in MapReduce

2009	2010
HadoopDB	File Level
Still a database	
	Global Sorting

Full-Text Indexing

[J. Lin et al.: Full-Text Indexing for Optimizing Selection Operations in Large-Scale Data Analytics. MapReduce Workshop 2011]

Index Creation

Tweets Dataset

“Mexico won the gold medal in soccer”

“Hadoop summit was awesome!”

“Hello from the other side of the world”

⋮
⋮
⋮

“Visiting Istanbul today!”

“Come in numbers to the HAIL talk!”

“I released our Hadoop-based system today”

Index Creation

Tweets Dataset

“Mexico won the gold medal in soccer”

“Hadoop summit was awesome!”

“Hello from the other side of the world”

⋮
⋮
⋮

“Visiting Istanbul today!”

“Come in numbers to the HAIL talk!”

“I released our Hadoop-based system today”

Row Group 1

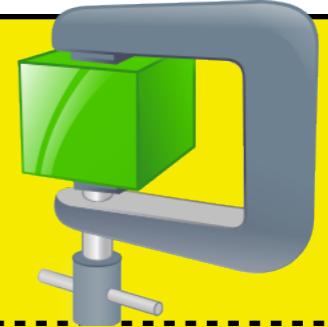
Row Group n

Index Creation

Tweets Dataset

Row Group 1

“Mexico won the gold medal in soccer”

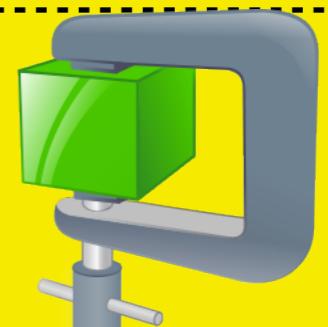


“Hadoop summit was awesome!”

“Hello from the other side of the world”

⋮
⋮

“Visiting Istanbul today!”



“Come in numbers to the HAIL talk!”

“I released our Hadoop-based system today”

Row Group n

Index Creation

Tweets Dataset

Row Group 1

“Mexico won the gold medal in soccer”



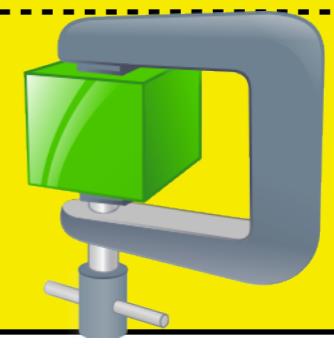
“Hadoop summit was awesome!”

“Hello from the other side of the world”

⋮
⋮
⋮

Row Group n

“Visiting Istanbul today!”



“Come in numbers to the HAIL talk!”

“I released our Hadoop-based system today”

Indexing Procedure

Index Creation

Tweets Dataset

Row Group 1

“Mexico won the gold medal in soccer”



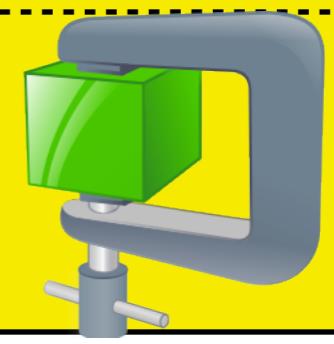
“Hadoop summit was awesome!”

“Hello from the other side of the world”

⋮
⋮

Row Group n

“Visiting Istanbul today!”



“Come in numbers to the HAIL talk!”

“I released our Hadoop-based system today”

Indexing Procedure

(1) for each Row Group

Index Creation

Tweets Dataset

Row Group 1

“Mexico won the gold medal in soccer”



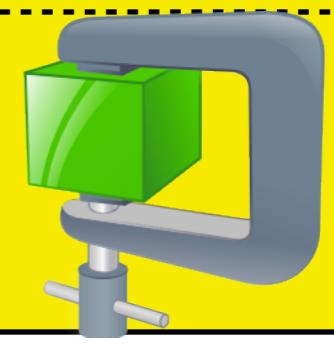
“Hadoop summit was awesome!”

“Hello from the other side of the world”

⋮
⋮

Row Group n

“Visiting Istanbul today!”



“Come in numbers to the HAIL talk!”

“I released our Hadoop-based system today”

Indexing Procedure

- (1) **for each** Row Group
- (2) create *pseudo-document*

Index Creation

Tweets Dataset

Row Group 1

“Mexico won the gold medal in soccer”



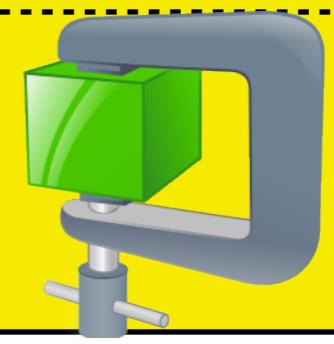
“Hadoop summit was awesome!”

“Hello from the other side of the world”

⋮
⋮

Row Group n

“Visiting Istanbul today!”



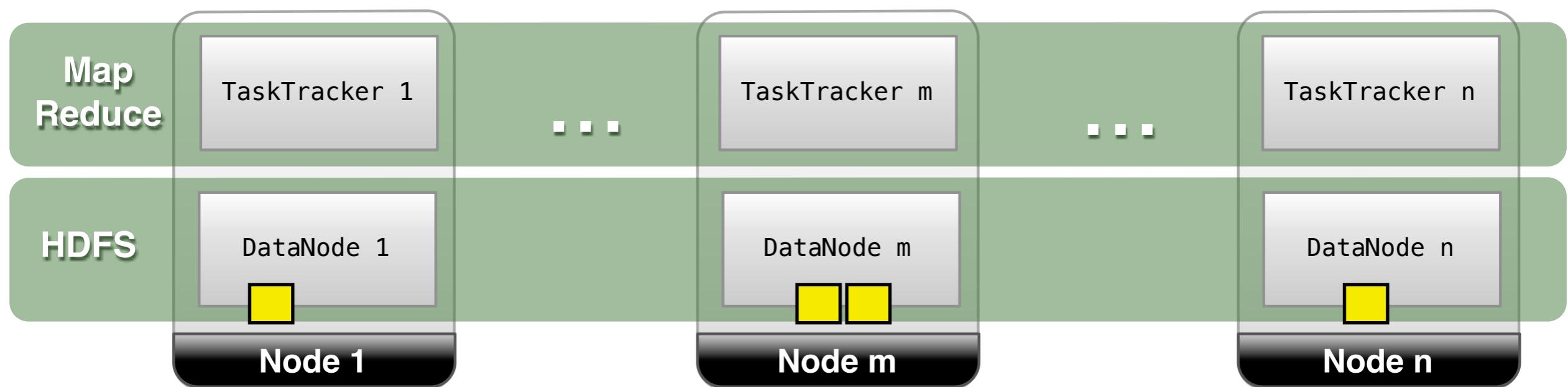
“Come in numbers to the HAIL talk!”

“I released our Hadoop-based system today”

Indexing Procedure

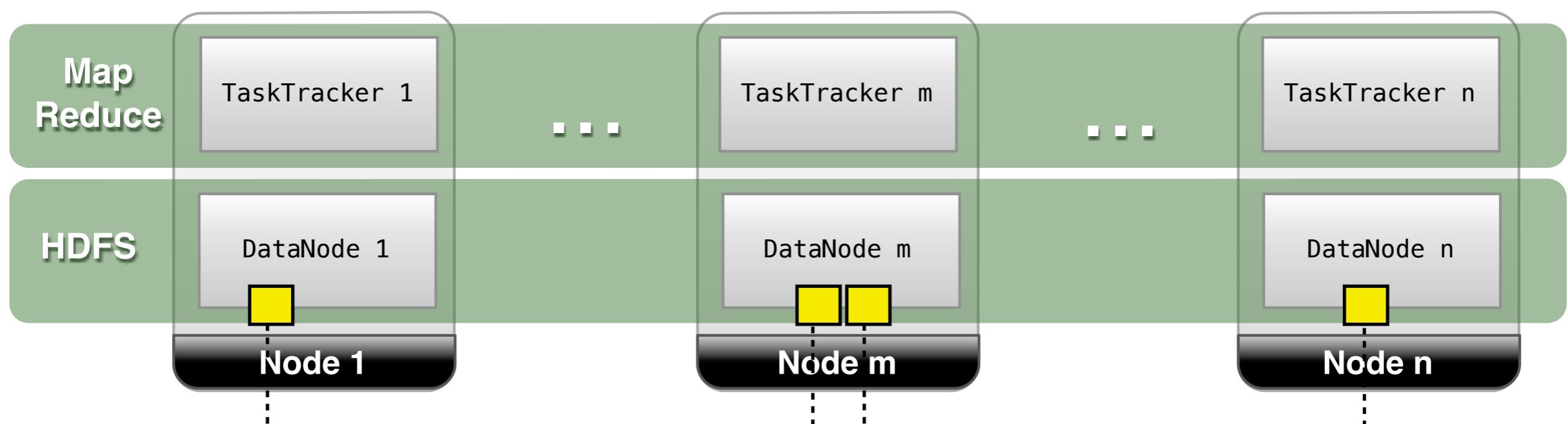
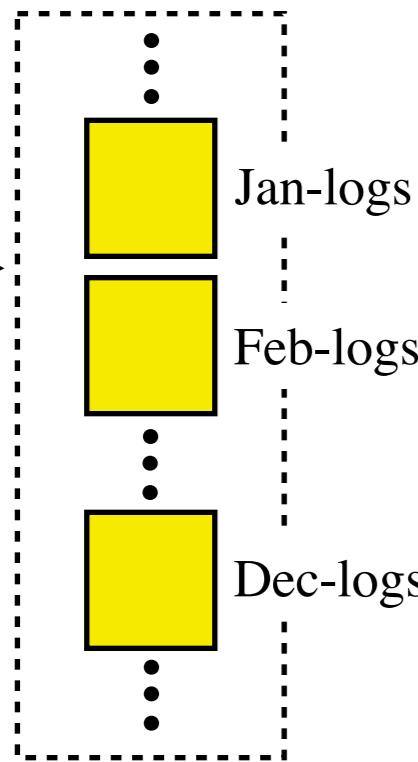
- (1) **for each** Row Group
- (2) create *pseudo-document*
- (3) index the pseudo-document in Lucene

Job Execution



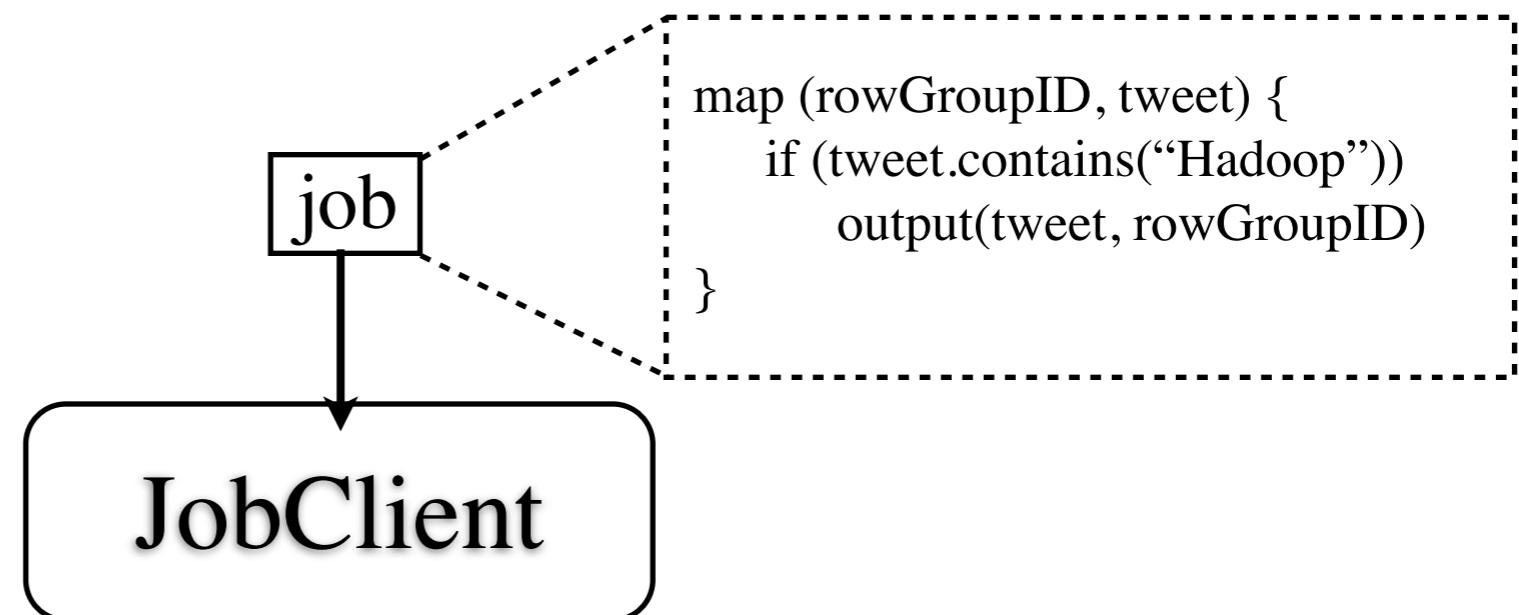
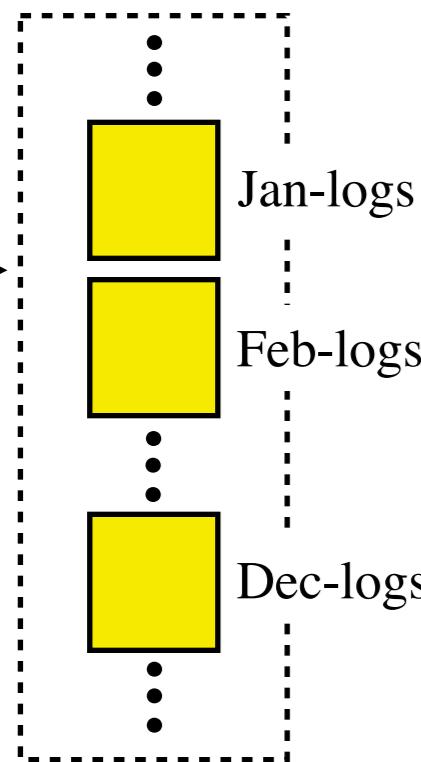
Job Execution

Tweets Dataset

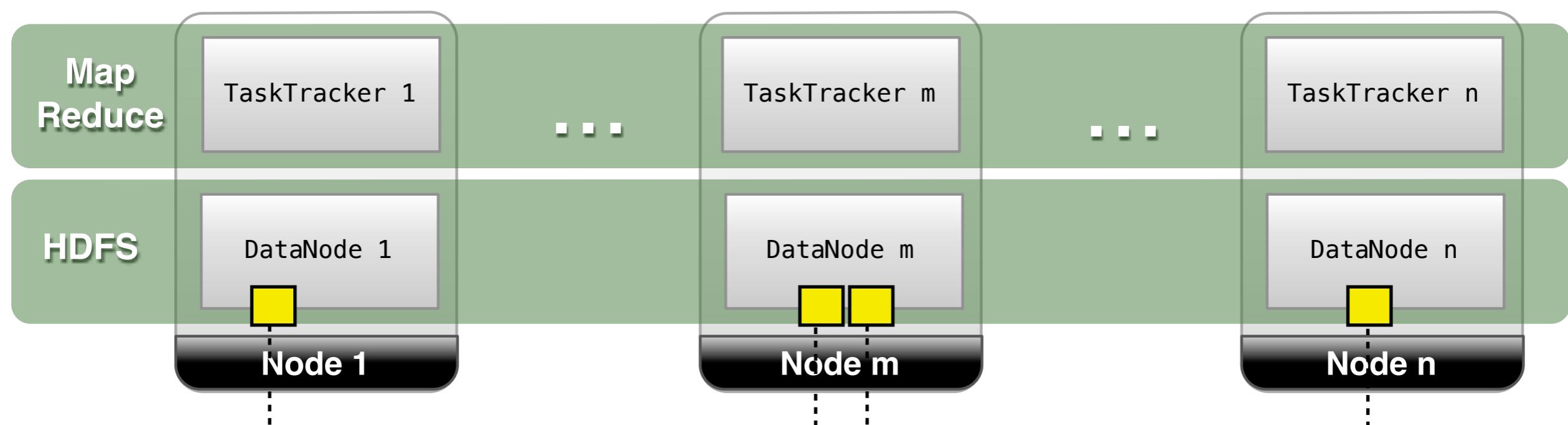


Job Execution

Tweets Dataset

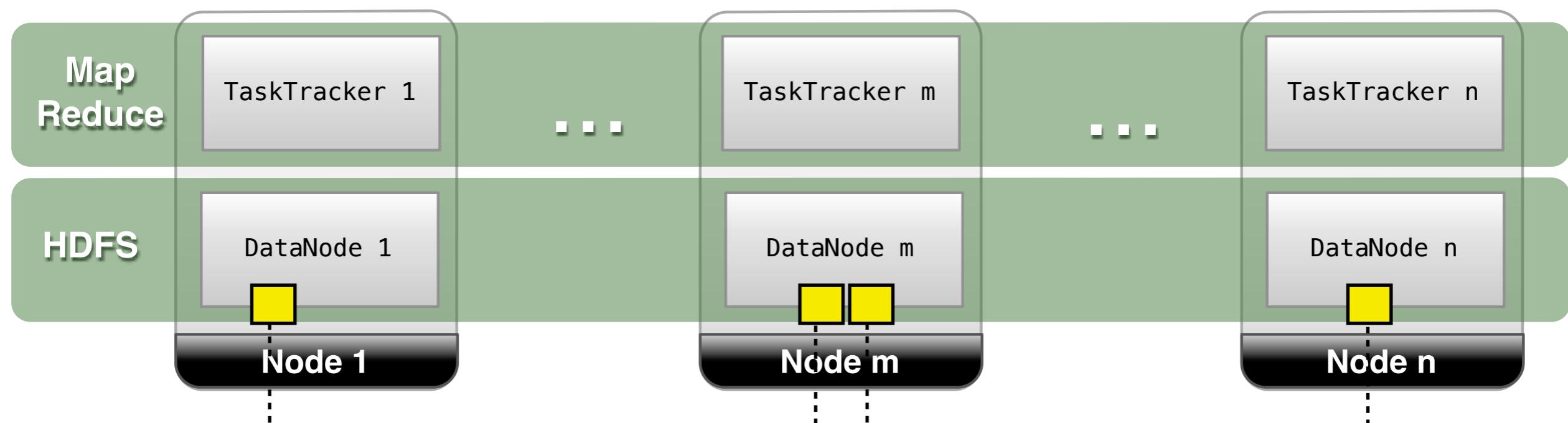
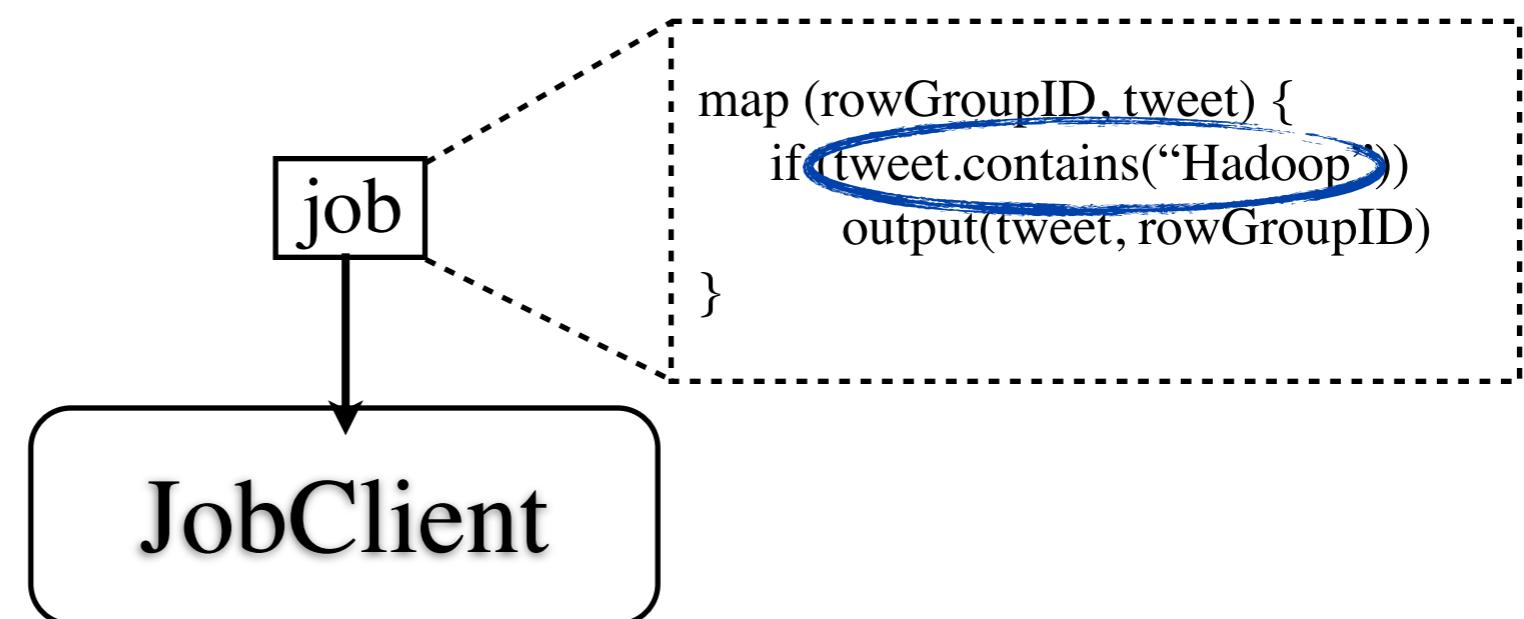
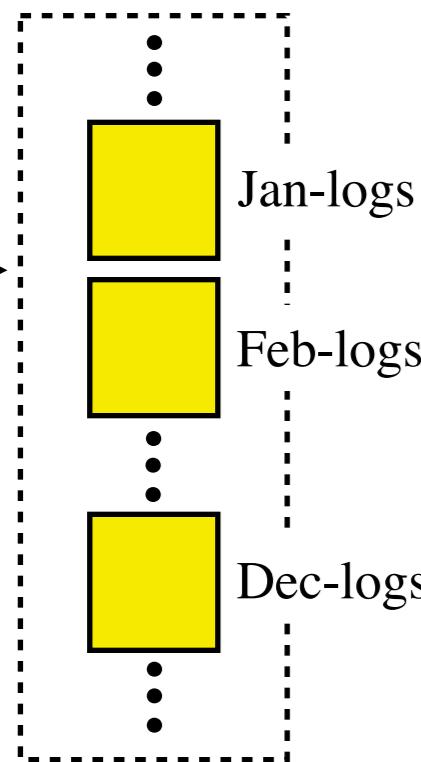


```
map (rowGroupID, tweet) {  
    if (tweet.contains("Hadoop"))  
        output(tweet, rowGroupID)  
}
```



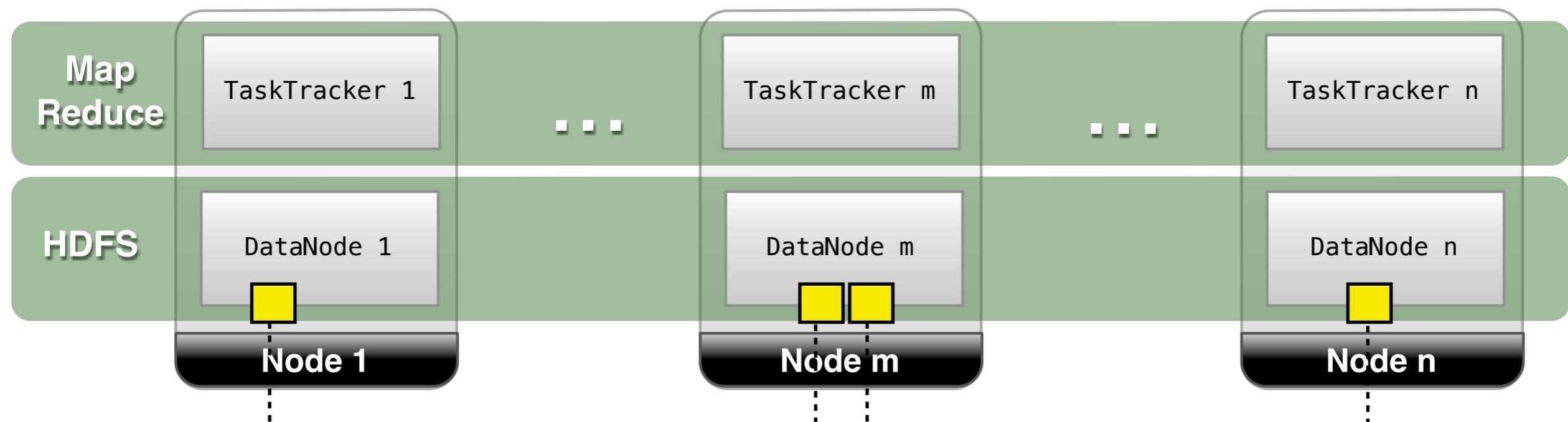
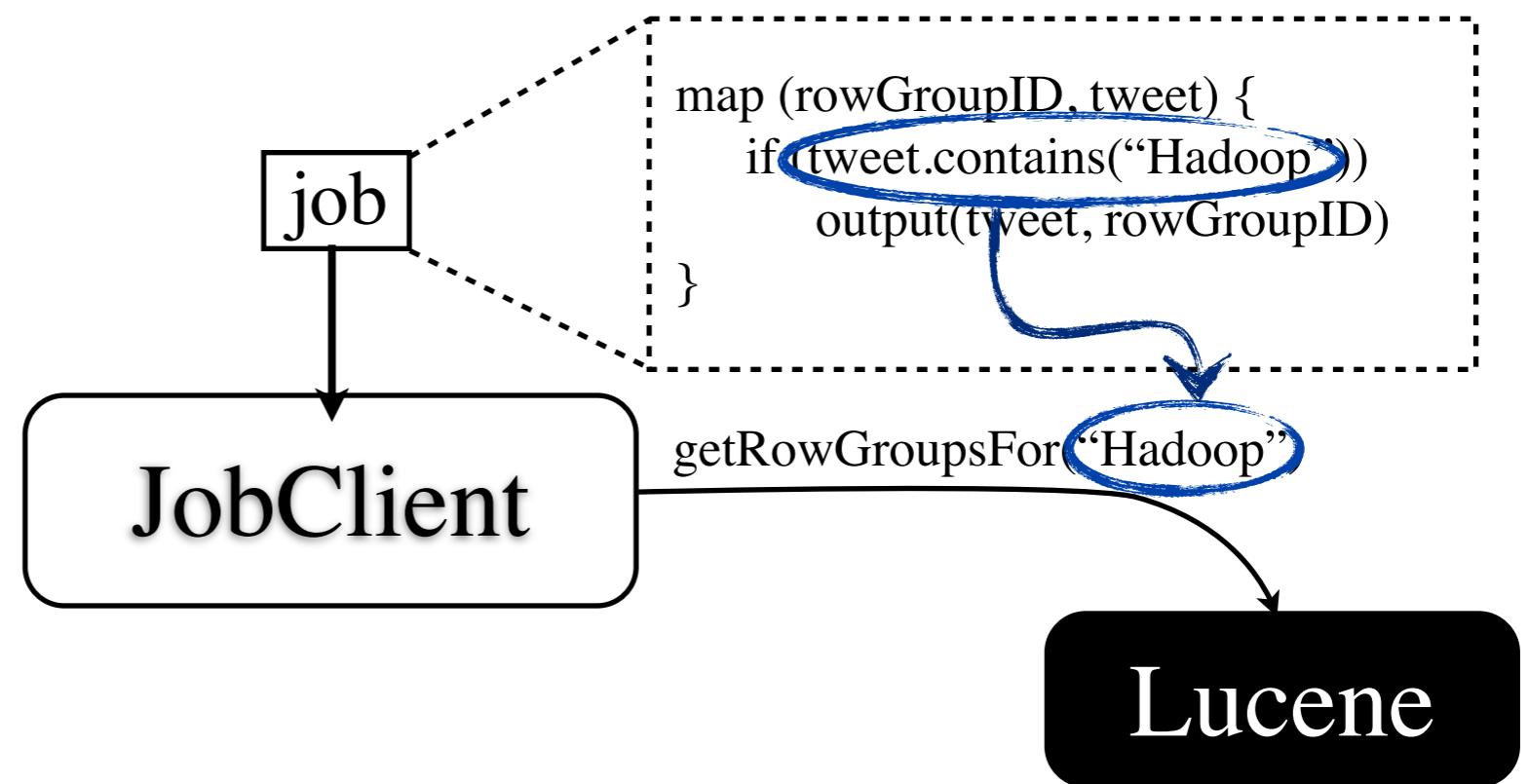
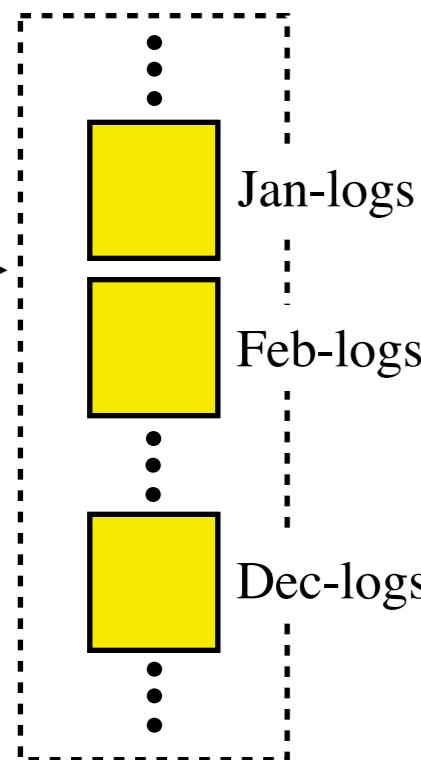
Job Execution

Tweets Dataset



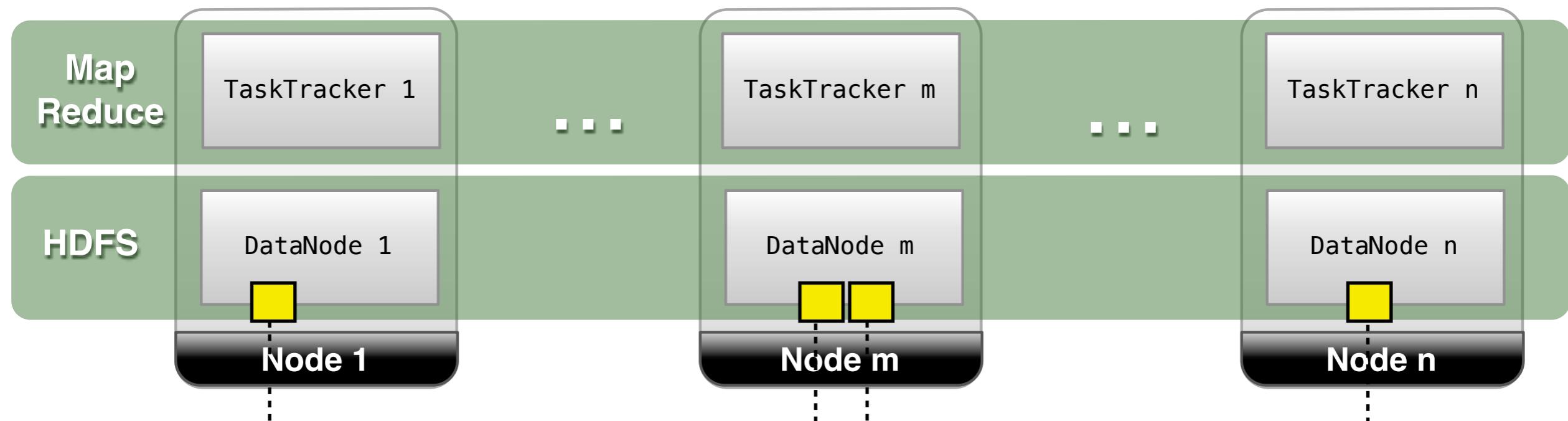
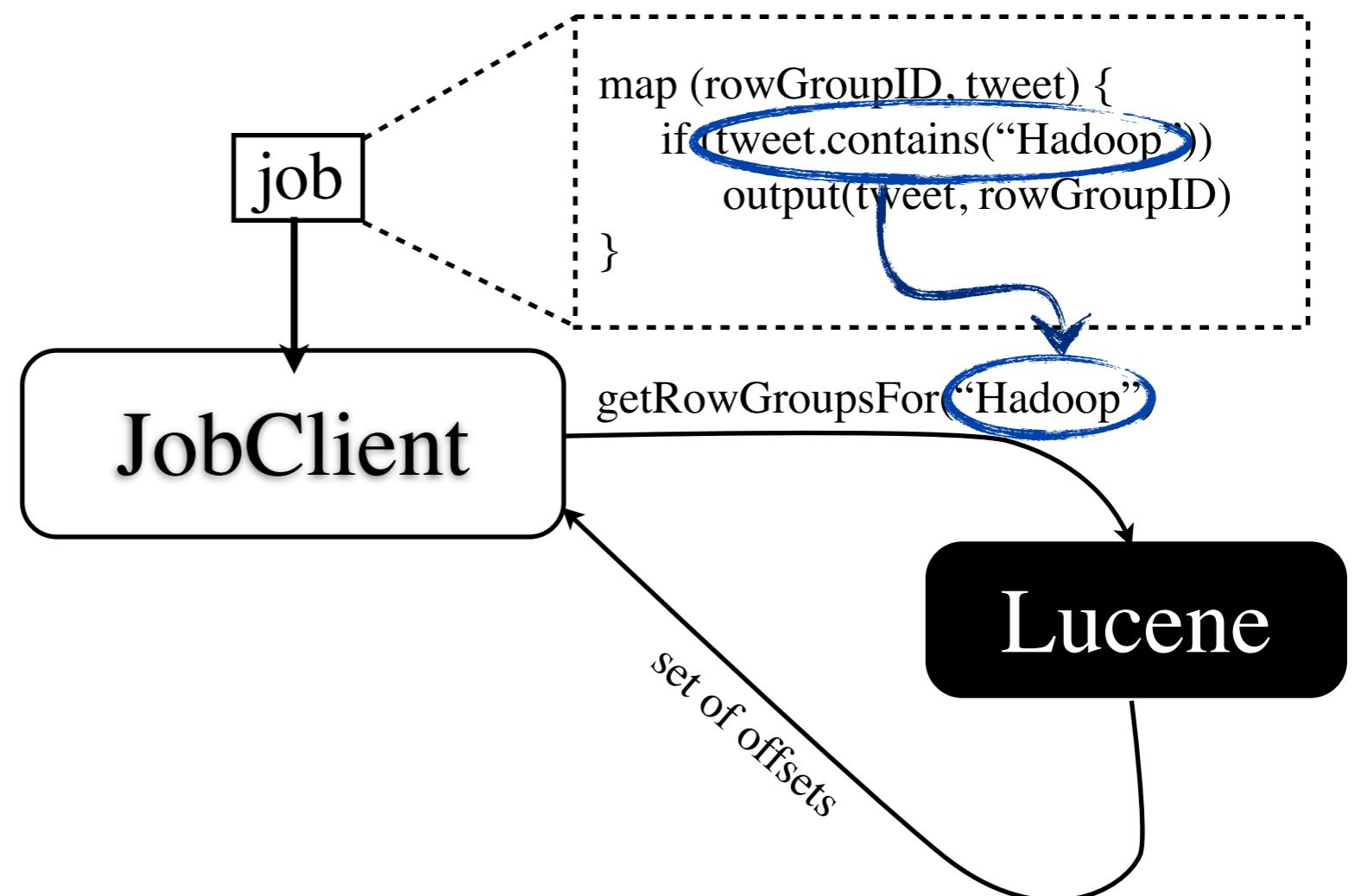
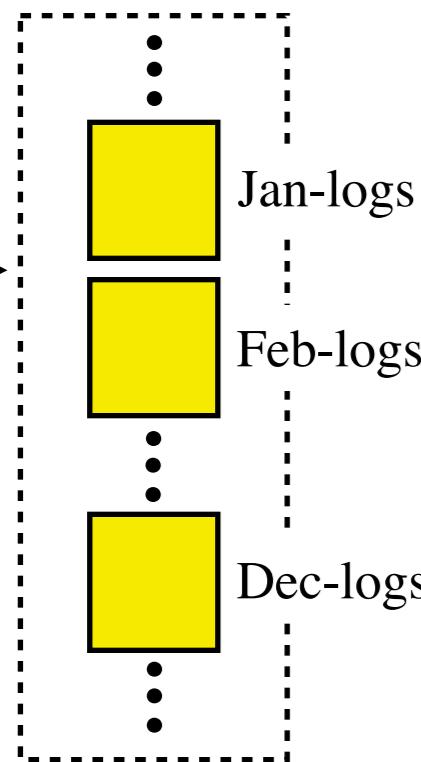
Job Execution

Tweets Dataset



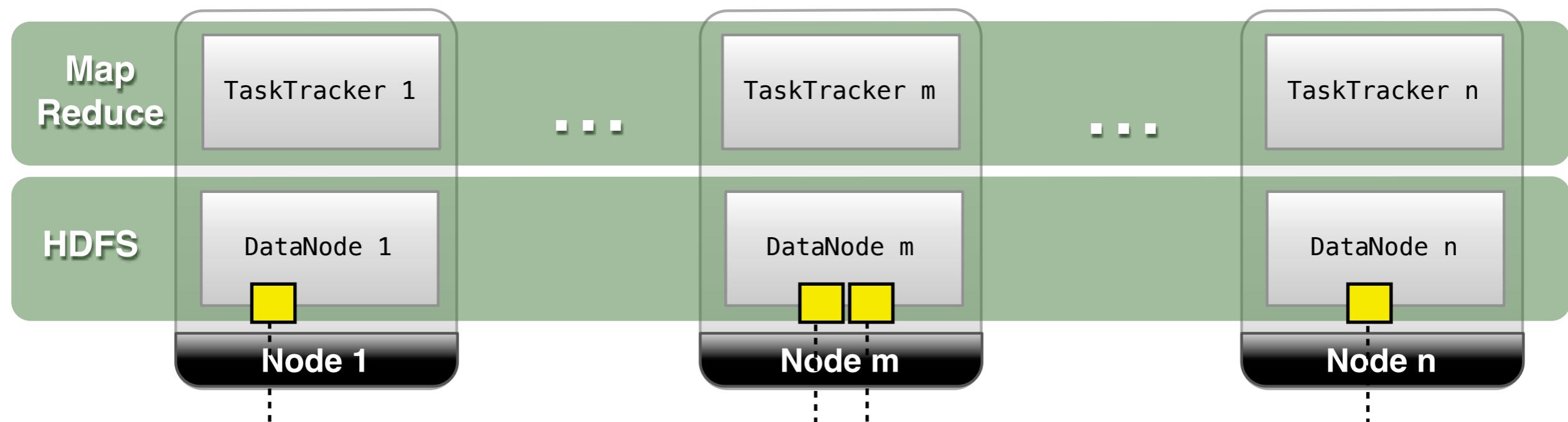
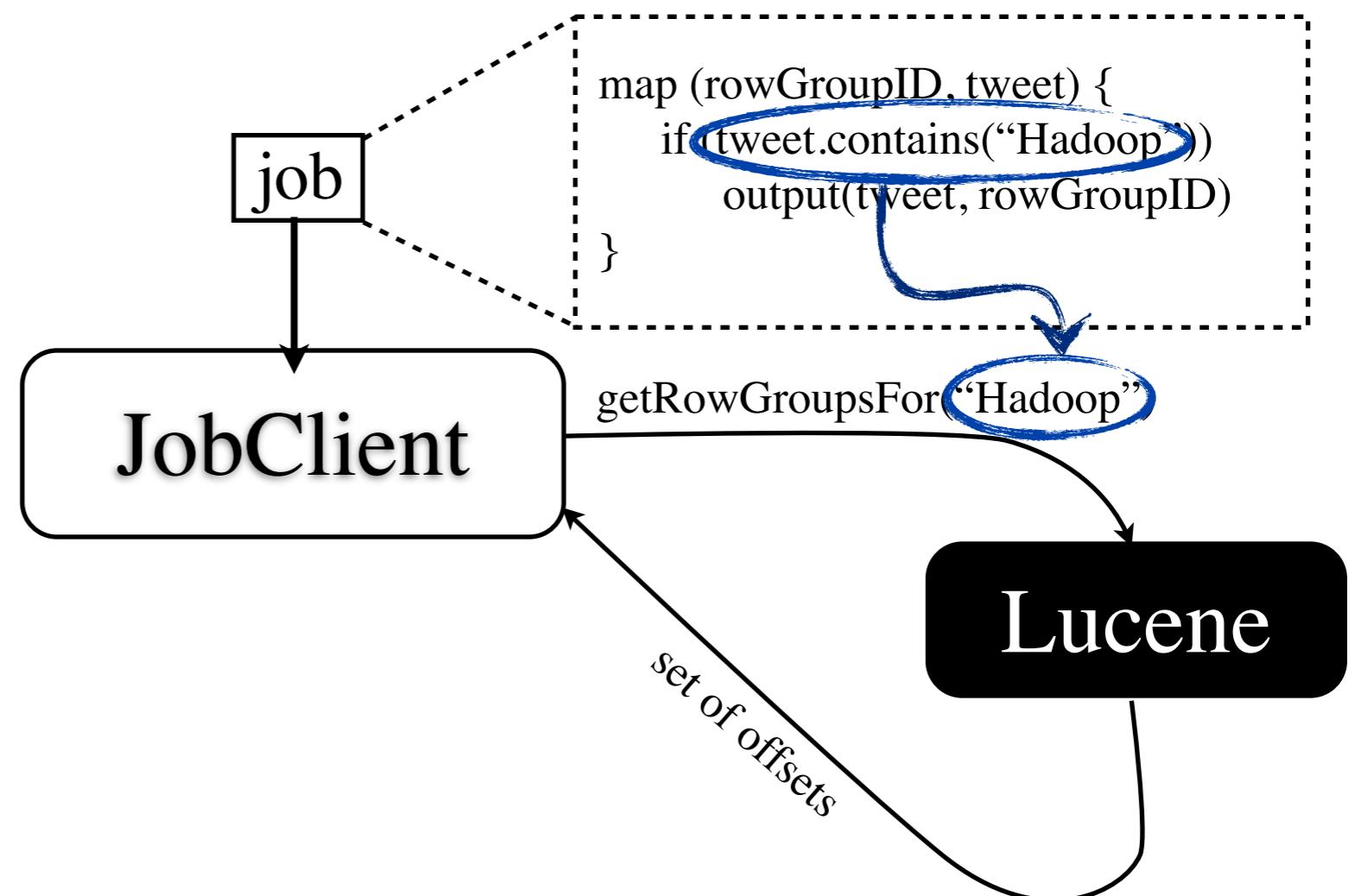
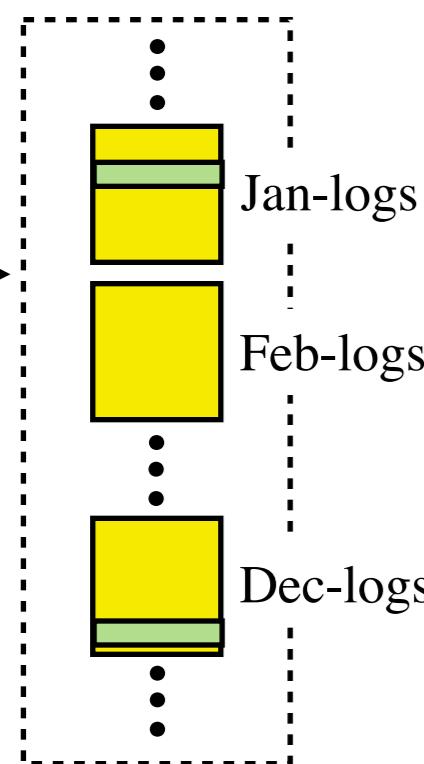
Job Execution

Tweets Dataset



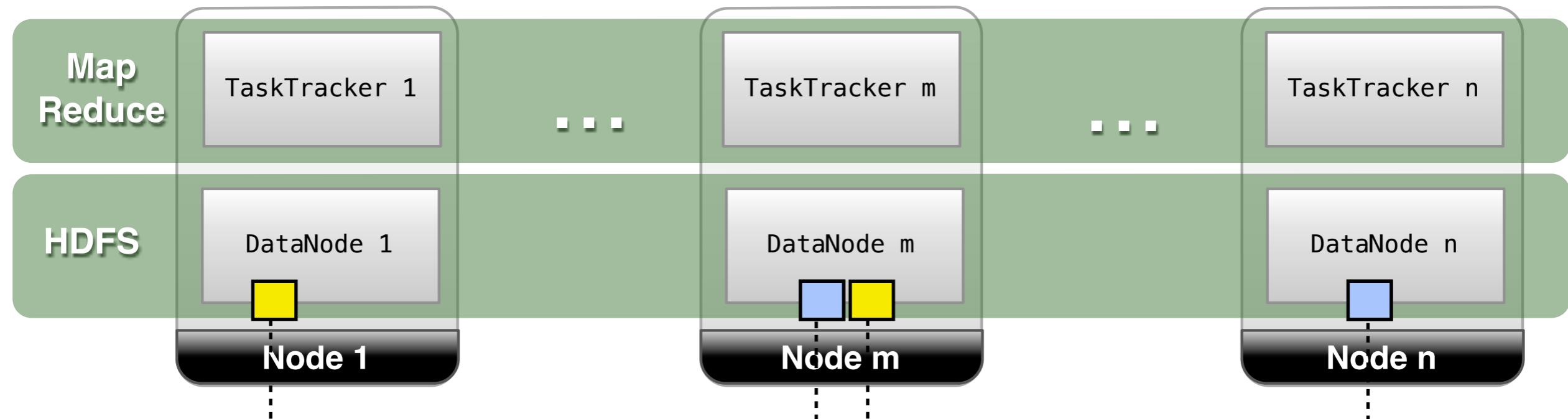
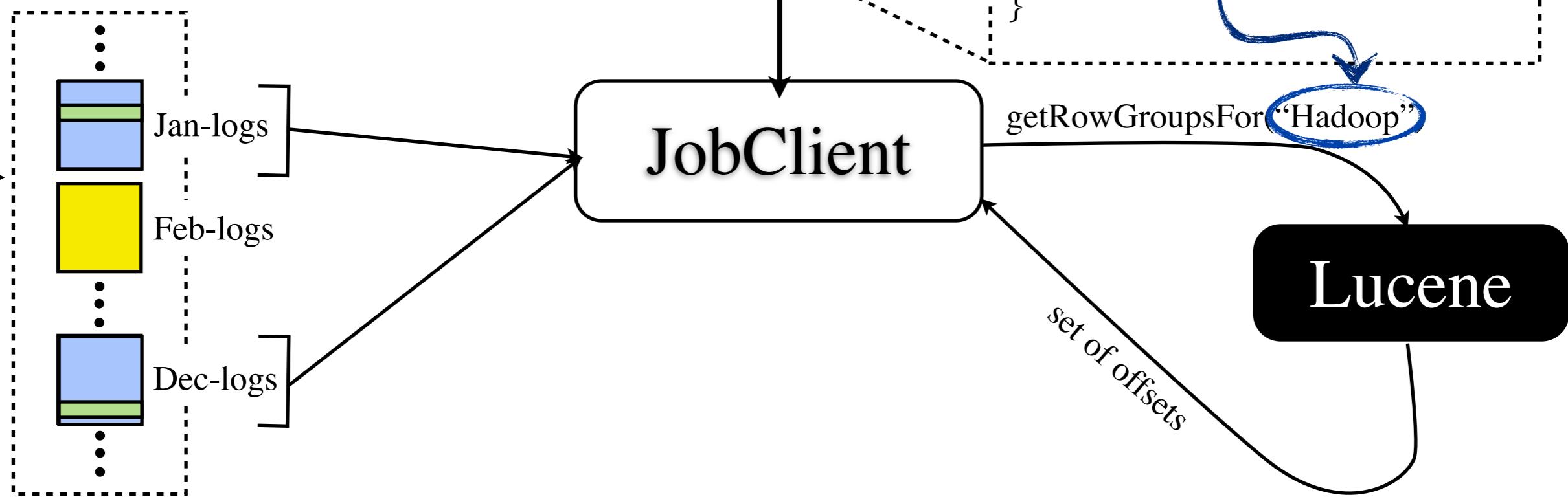
Job Execution

Tweets Dataset



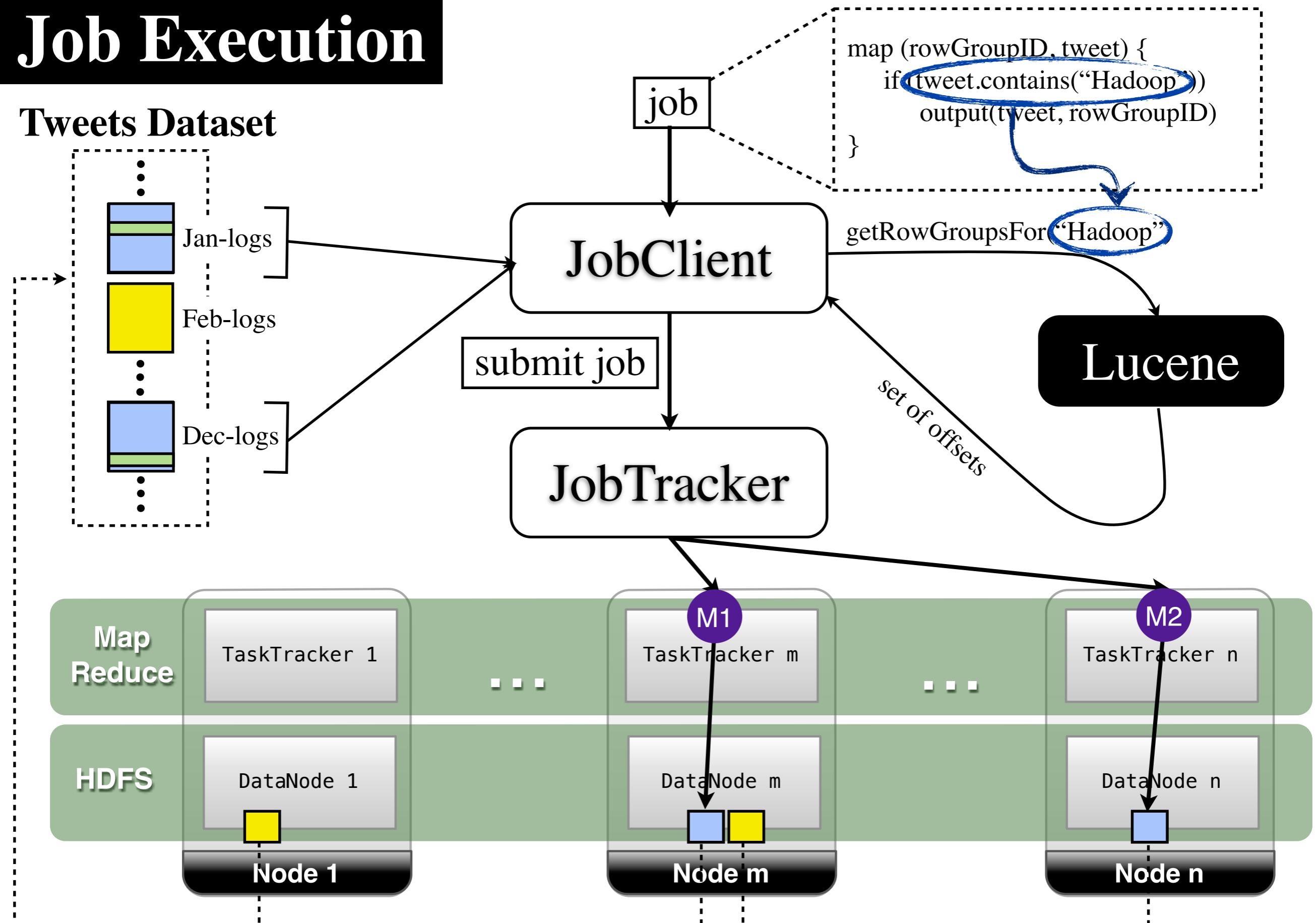
Job Execution

Tweets Dataset



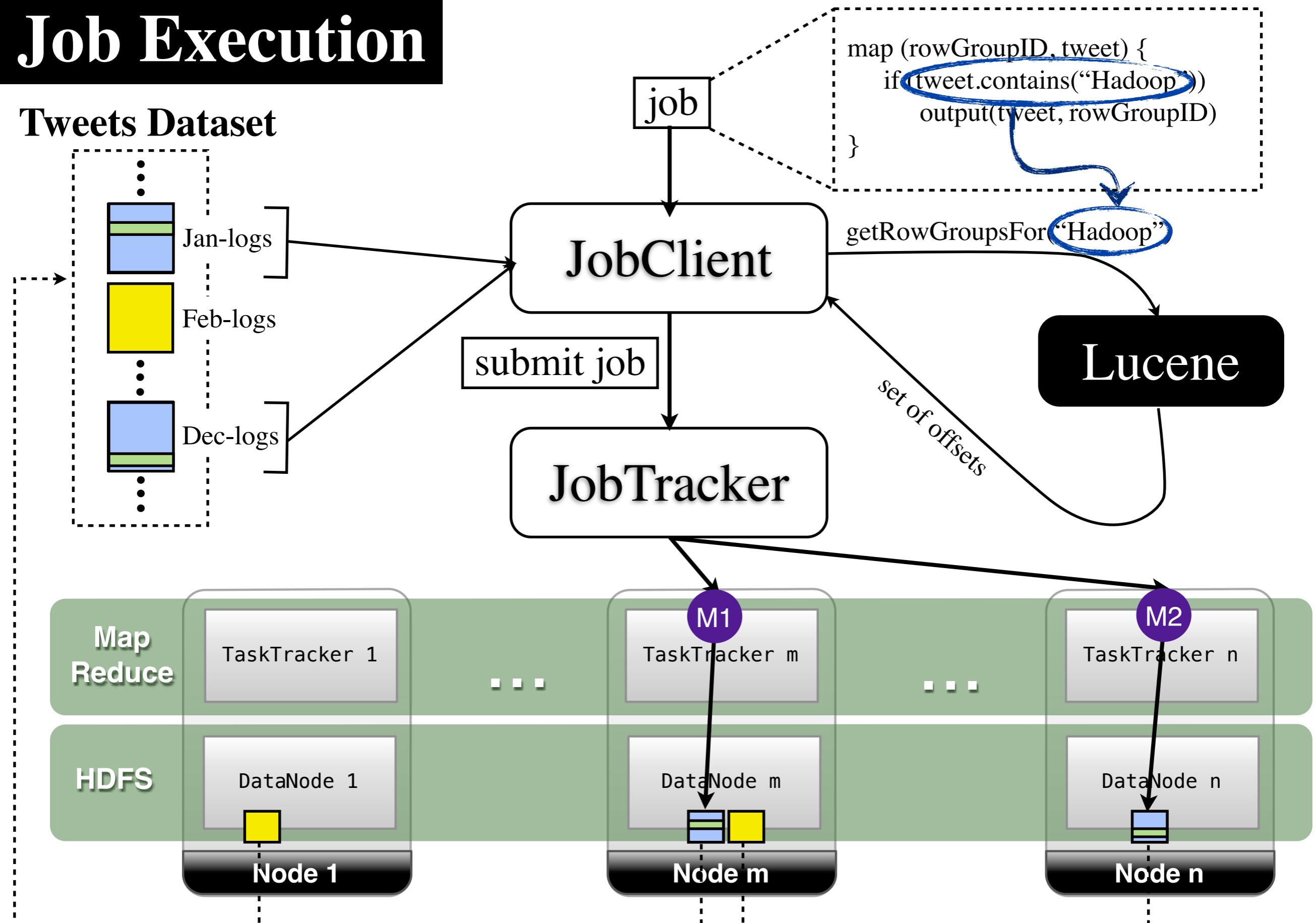
Job Execution

Tweets Dataset



Job Execution

Tweets Dataset



Full-Text Indexing Results

Setup

Tweets Dataset: 69.2 million tweets

Dataset Size: 6.07GB

#Row Groups: 39,767

Avg #Records per Row Group: 1,740

Full-Text Indexing Results

Setup

Tweets Dataset: 69.2 million tweets

Dataset Size: 6.07GB

#Row Groups: 39,767

Avg #Records per Row Group: 1,740

Query	Row Groups	Records	Selectivity
1 hadoop	97	105	1.517×10^{-6}
2 replication	140	151	2.182×10^{-6}
3 buffer	500	559	8.076×10^{-6}
4 transactions	819	867	1.253×10^{-5}
5 parallel	999	1159	1.674×10^{-5}
6 ibm	1437	1569	2.267×10^{-5}
7 mysql	1511	1664	2.404×10^{-5}
8 oracle	1822	1911	2.761×10^{-5}
9 database	3759	3981	5.752×10^{-5}
10 microsoft	13089	17408	2.515×10^{-4}
11 data	20087	30145	4.355×10^{-4}

Full-Text Indexing Results

Setup

Tweets Dataset: 69.2 million tweets

Dataset Size: 6.07GB

#Row Groups: 39,767

Avg #Records per Row Group: 1,740

Highly selective queries

	Query	Row Groups	Records	Selectivity
1	hadoop	97	105	1.517×10^{-6}
2	replication	140	151	2.182×10^{-6}
3	buffer	500	559	8.076×10^{-6}
4	transactions	819	867	1.253×10^{-5}
5	parallel	999	1159	1.674×10^{-5}
6	ibm	1437	1569	2.267×10^{-5}
7	mysql	1511	1664	2.404×10^{-5}
8	oracle	1822	1911	2.761×10^{-5}
9	database	3759	3981	5.752×10^{-5}
10	microsoft	13089	17408	2.515×10^{-4}
11	data	20087	30145	4.355×10^{-4}

Full-Text Indexing Results

Setup

Tweets Dataset: 69.2 million tweets

Dataset Size: 6.07GB

#Row Groups: 39,767

Avg #Records per Row Group: 1,740

168,675 additional records

	Query	Row Groups	Records	Selectivity
1	hadoop	97	105	1.517×10^{-6}
2	replication	140	151	2.182×10^{-6}
3	buffer	500	559	8.076×10^{-6}
4	transactions	819	867	1.253×10^{-5}
5	parallel	999	1159	1.674×10^{-5}
6	ibm	1437	1569	2.267×10^{-5}
7	mysql	1511	1664	2.404×10^{-5}
8	oracle	1822	1911	2.761×10^{-5}
9	database	3759	3981	5.752×10^{-5}
10	microsoft	13089	17408	2.515×10^{-4}
11	data	20087	30145	4.355×10^{-4}

Full-Text Indexing Results

Setup

Tweets Dataset: 69.2 million tweets

Dataset Size: 6.07GB

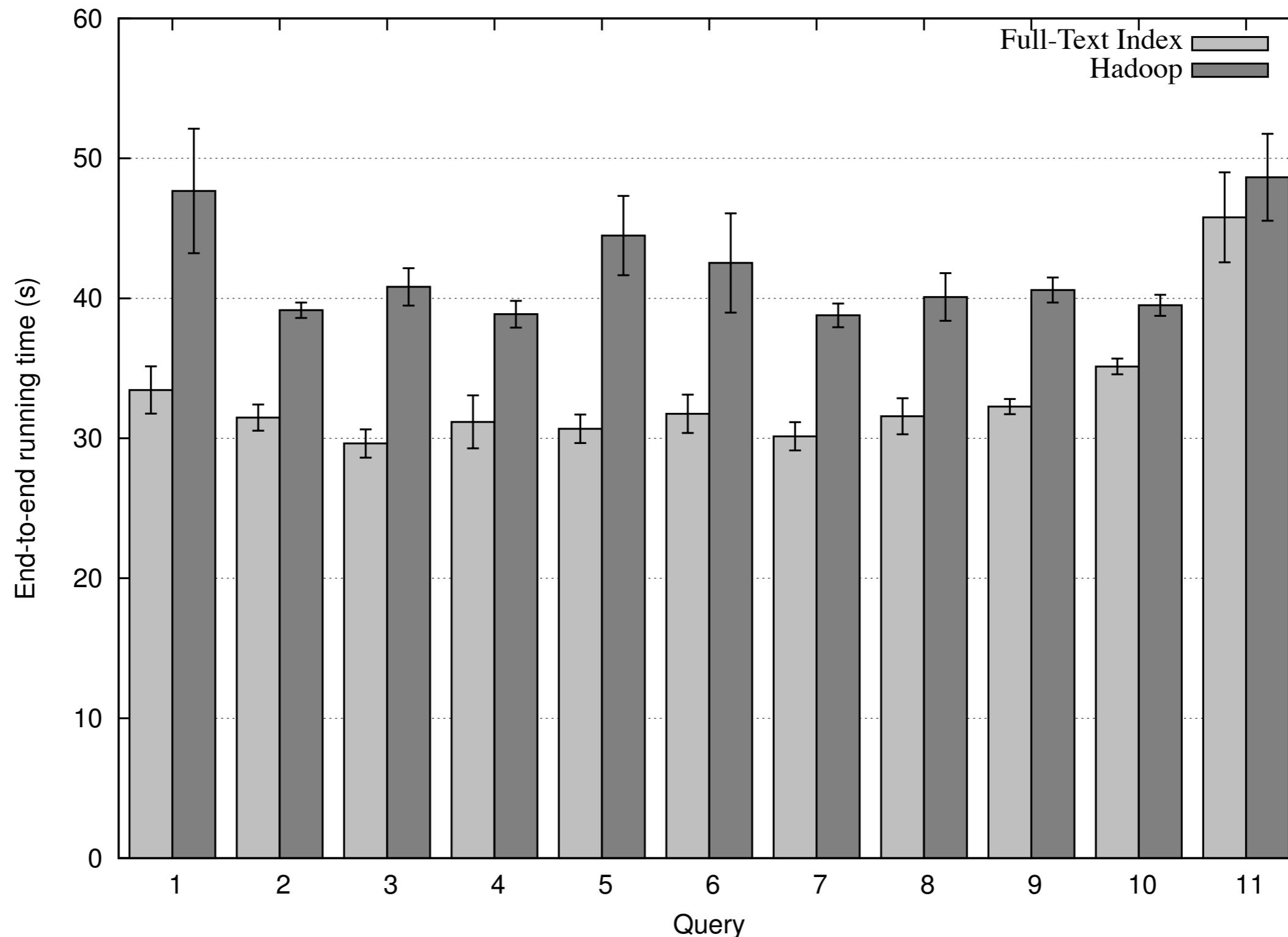
#Row Groups: 39,767

Avg #Records per Row Group: 1,740

>30% of Row Groups are read!

Query	Row Groups	Records	Selectivity
1 hadoop	97	105	1.517×10^{-6}
2 replication	140	151	2.182×10^{-6}
3 buffer	500	559	8.076×10^{-6}
4 transactions	819	867	1.253×10^{-5}
5 parallel	999	1159	1.674×10^{-5}
6 ibm	1437	1569	2.267×10^{-5}
7 mysql	1511	1664	2.404×10^{-5}
8 oracle	1822	1911	2.761×10^{-5}
9 database	3759	3981	5.752×10^{-5}
10 microsoft	13089	17408	2.515×10^{-4}
11 data	20087	30145	4.355×10^{-4}

Full-Text Indexing Results



Indexing in MapReduce

2009	2010
HadoopDB	File Level
Still a database	
	Global Sorting

Indexing in MapReduce

2009	2010	2011
HadoopDB	File Level	Full Text
Still a database		
	Global Sorting	
		Only for high selectivity

Trojan Index

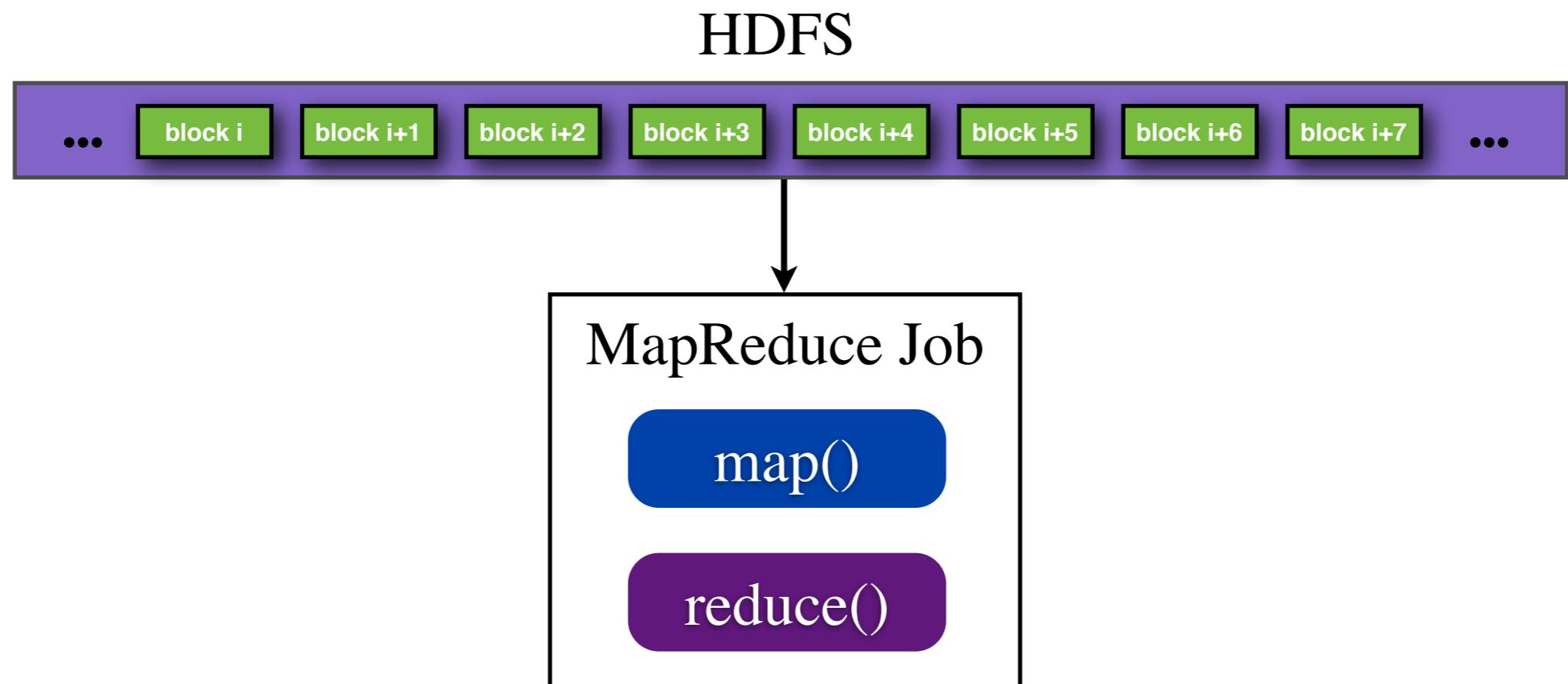
[J. Dittrich, J. Quiané, A. Jindal, Y. Kargin, V. Setty, J. Schad: Hadoop++:
Making a Yellow Elephant Run Like a Cheetah (Without It Even Noticing).
PVLDB 2010]

Index Creation

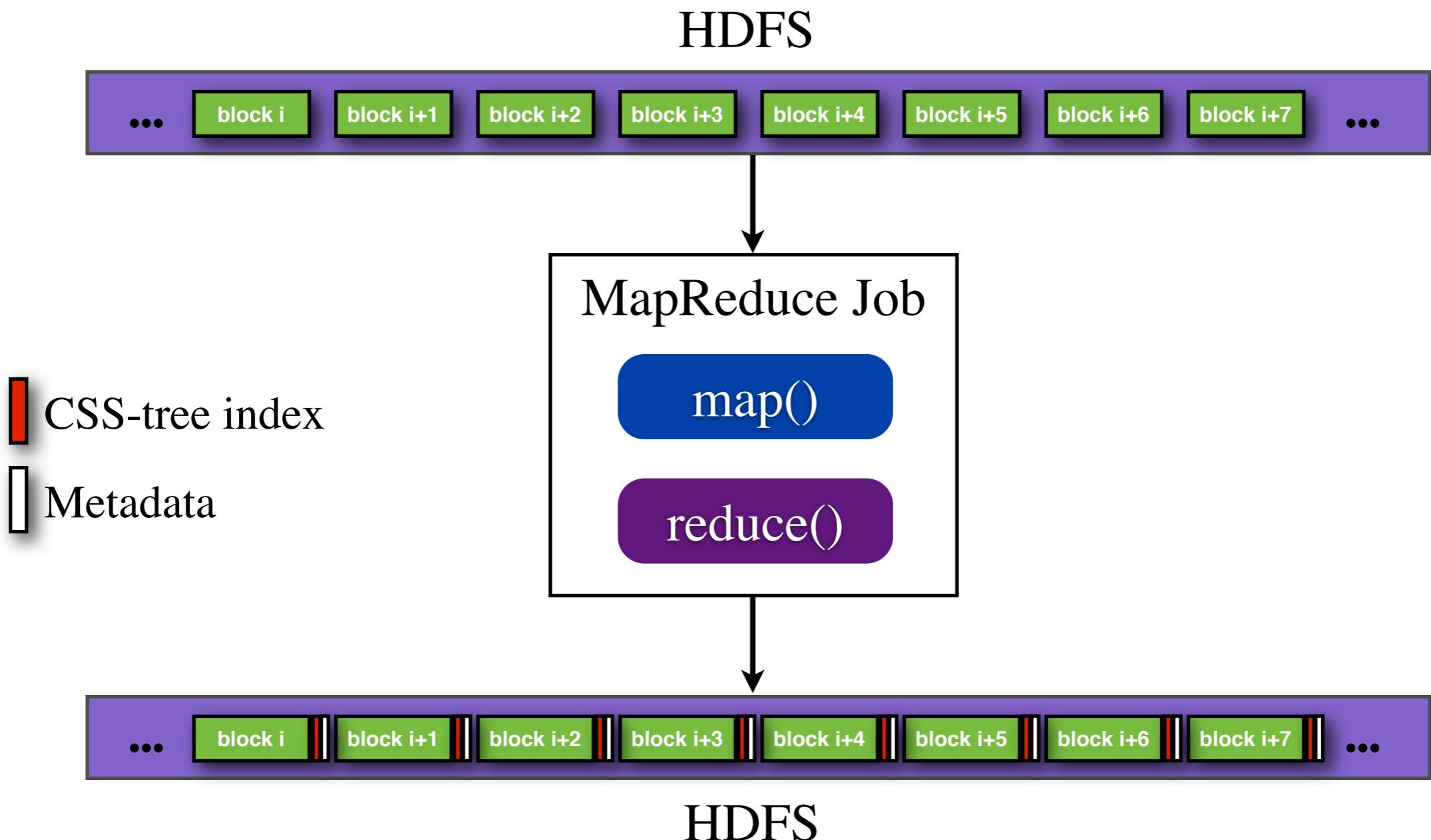
HDFS



Index Creation



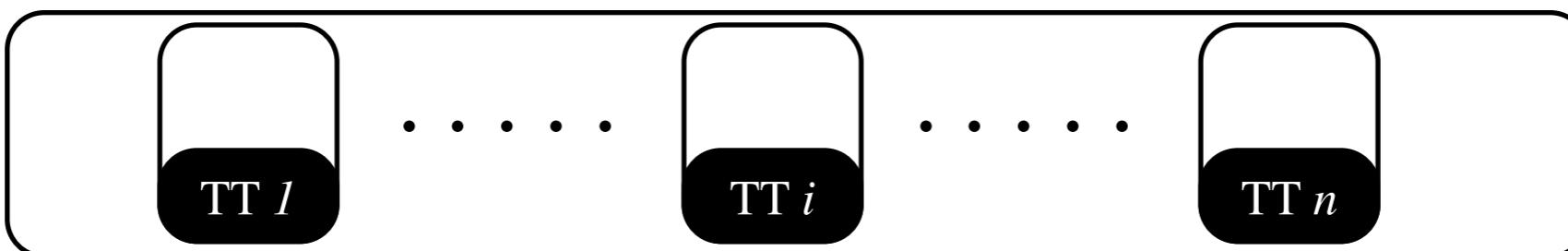
Index Creation



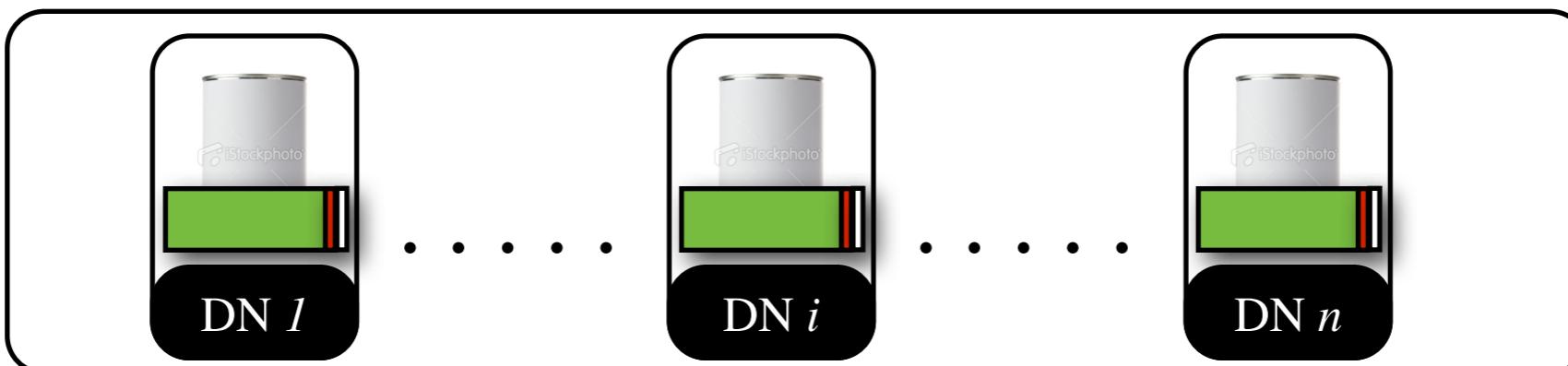
Job Execution



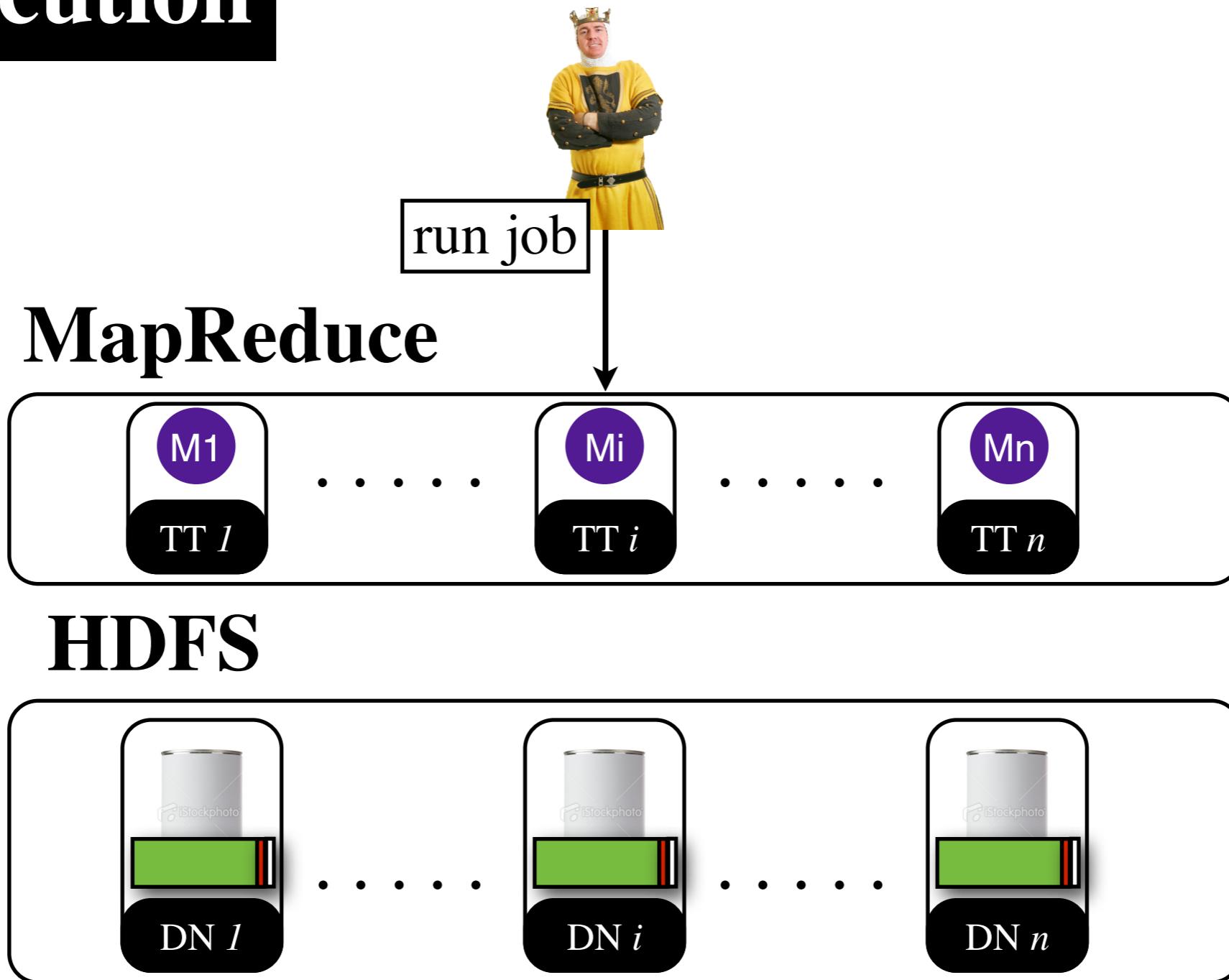
MapReduce



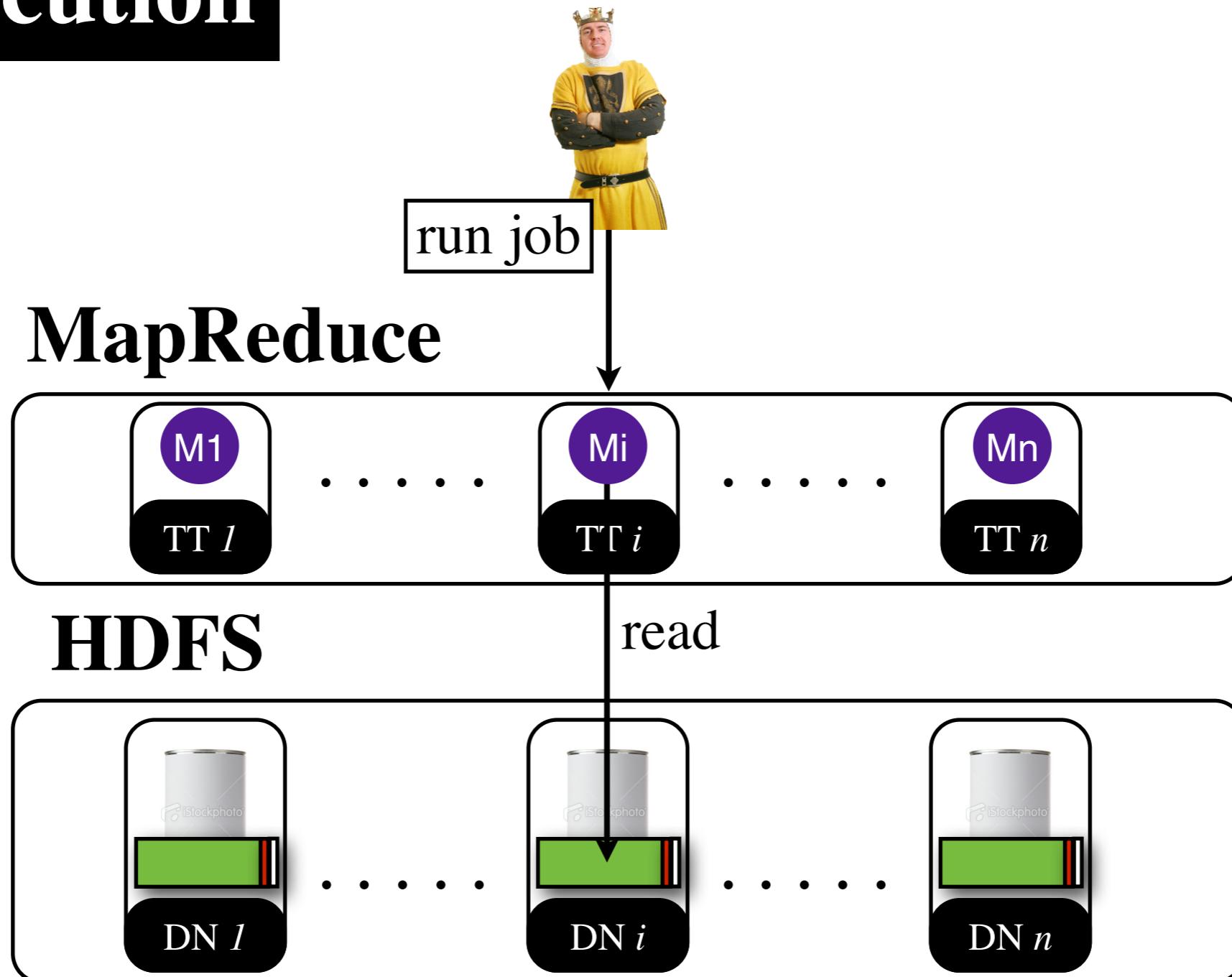
HDFS



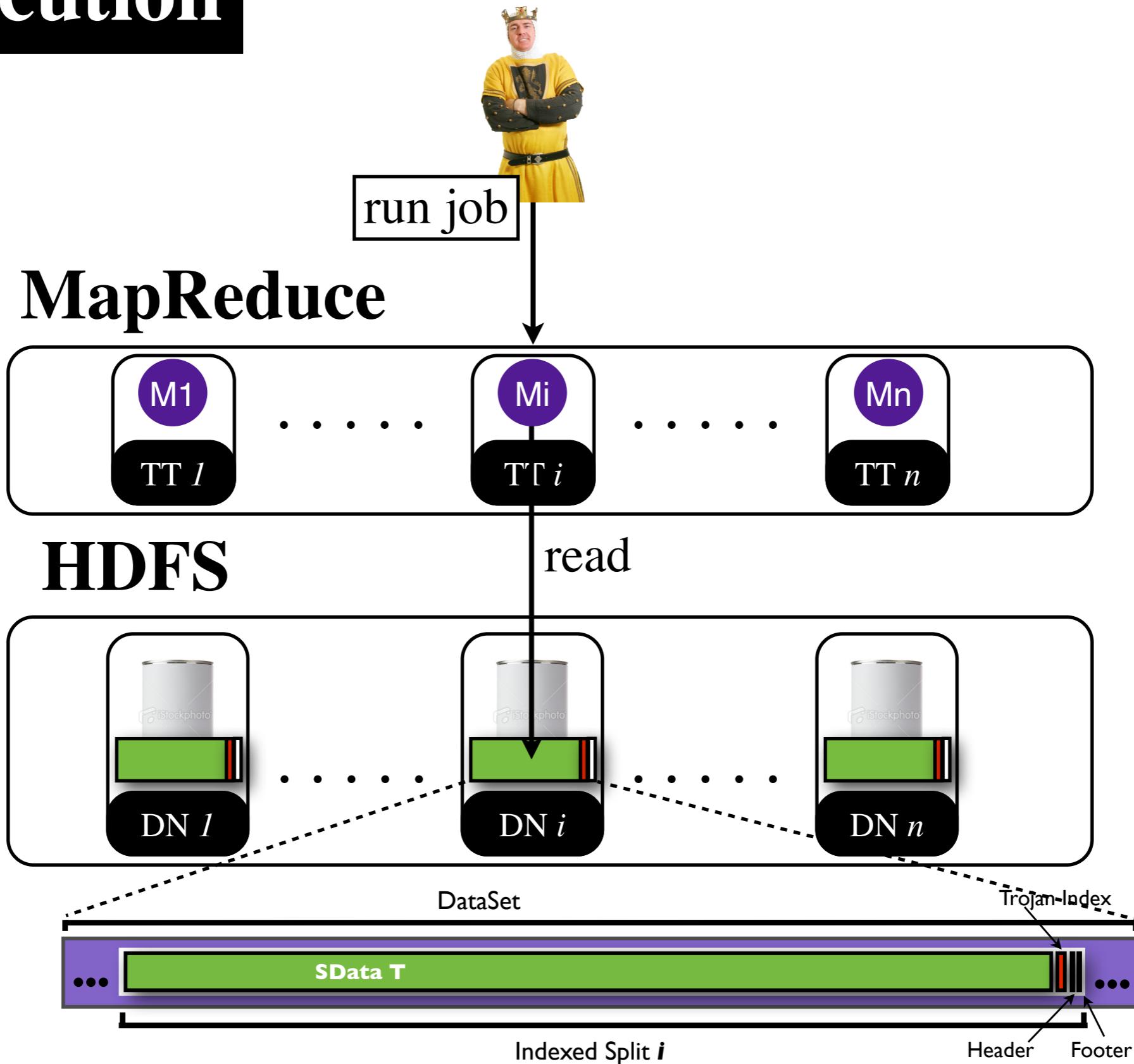
Job Execution



Job Execution

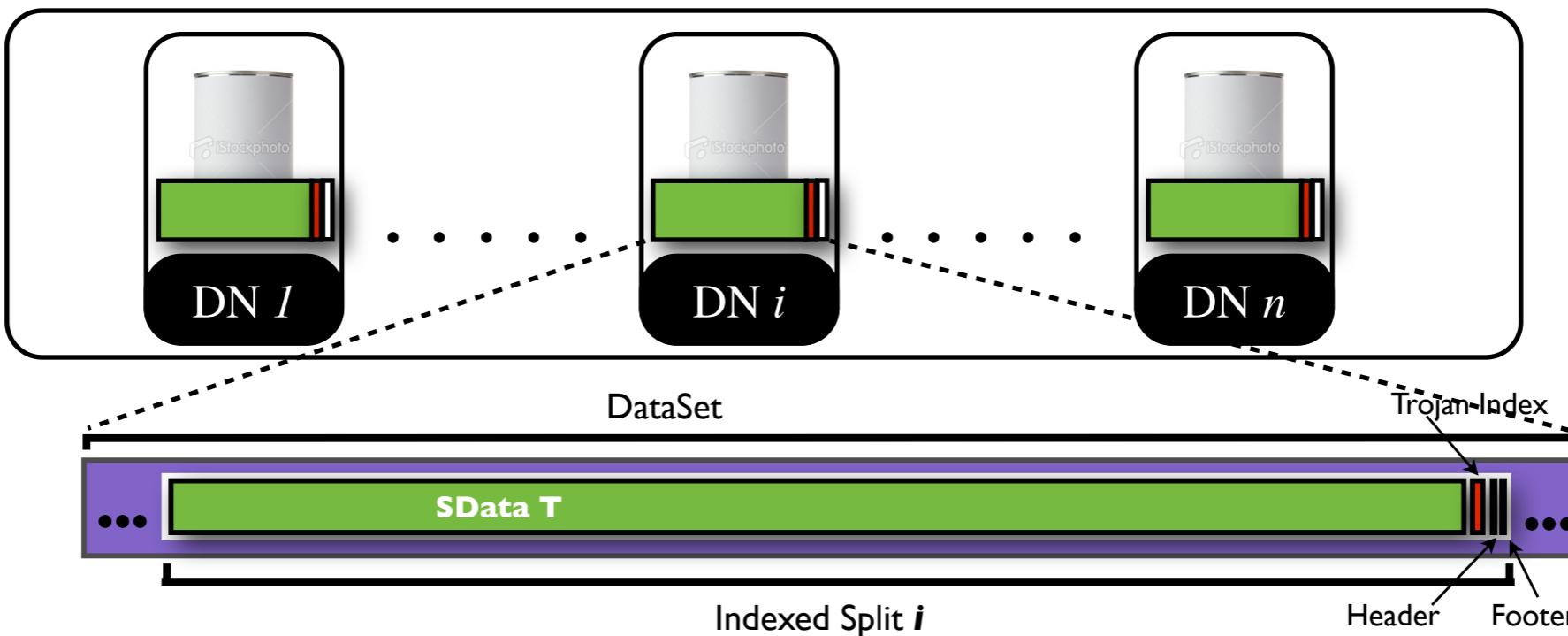


Job Execution



Job Execution

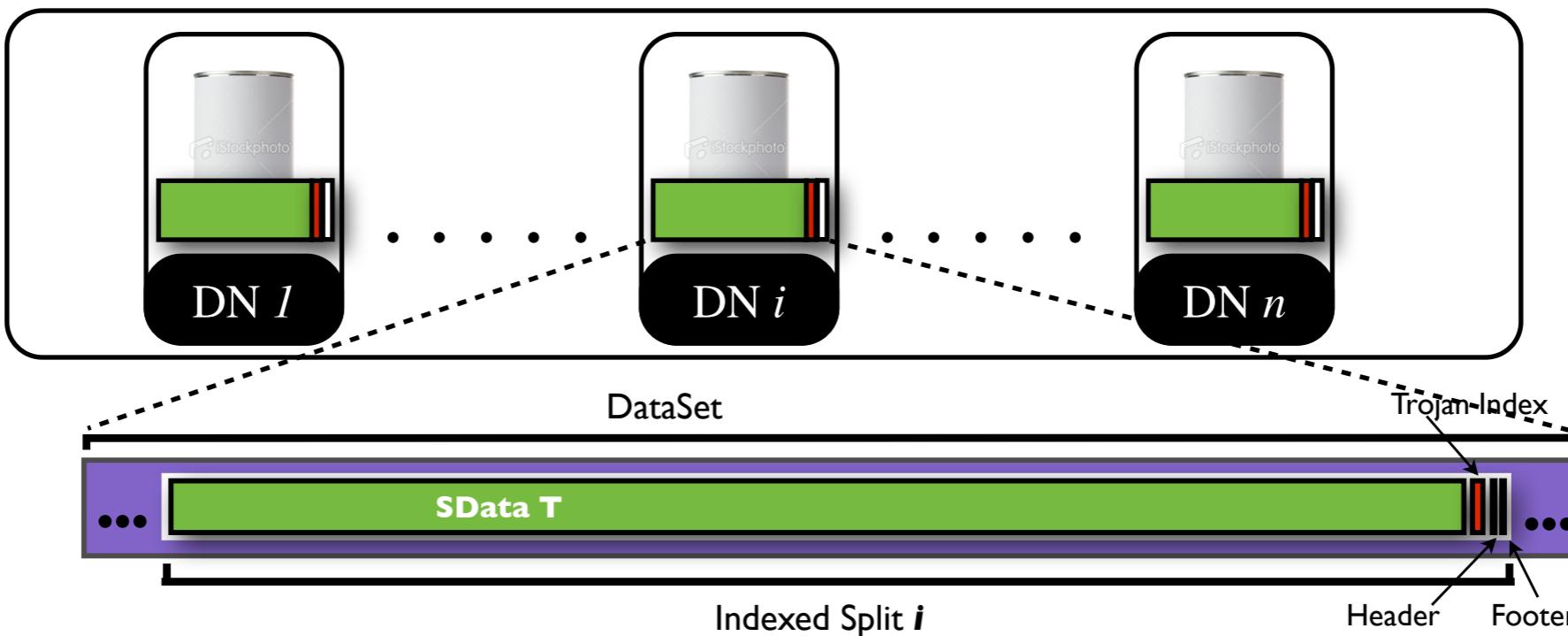
HDFS



RecordReader

Job Execution

HDFS

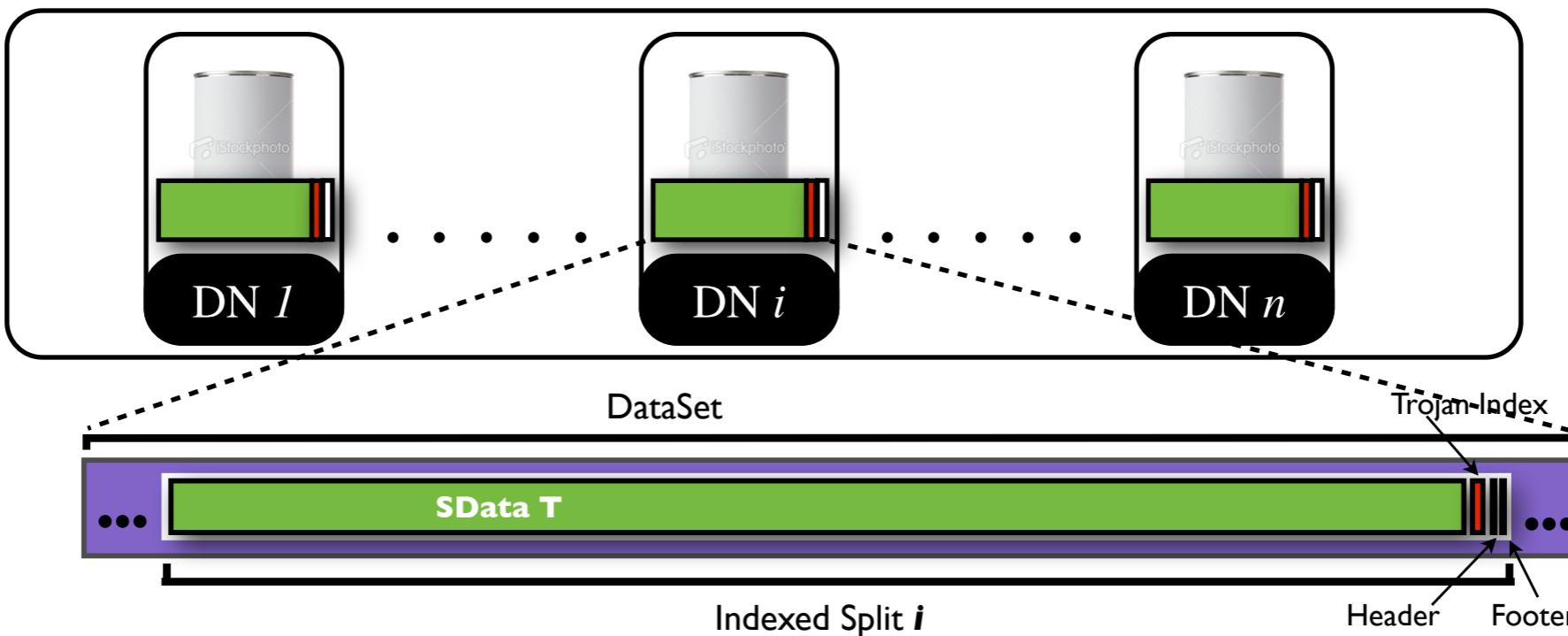


RecordReader

- (1) Read header to obtain $[\text{key}_{\min}, \text{key}_{\max}]$ -range of index

Job Execution

HDFS

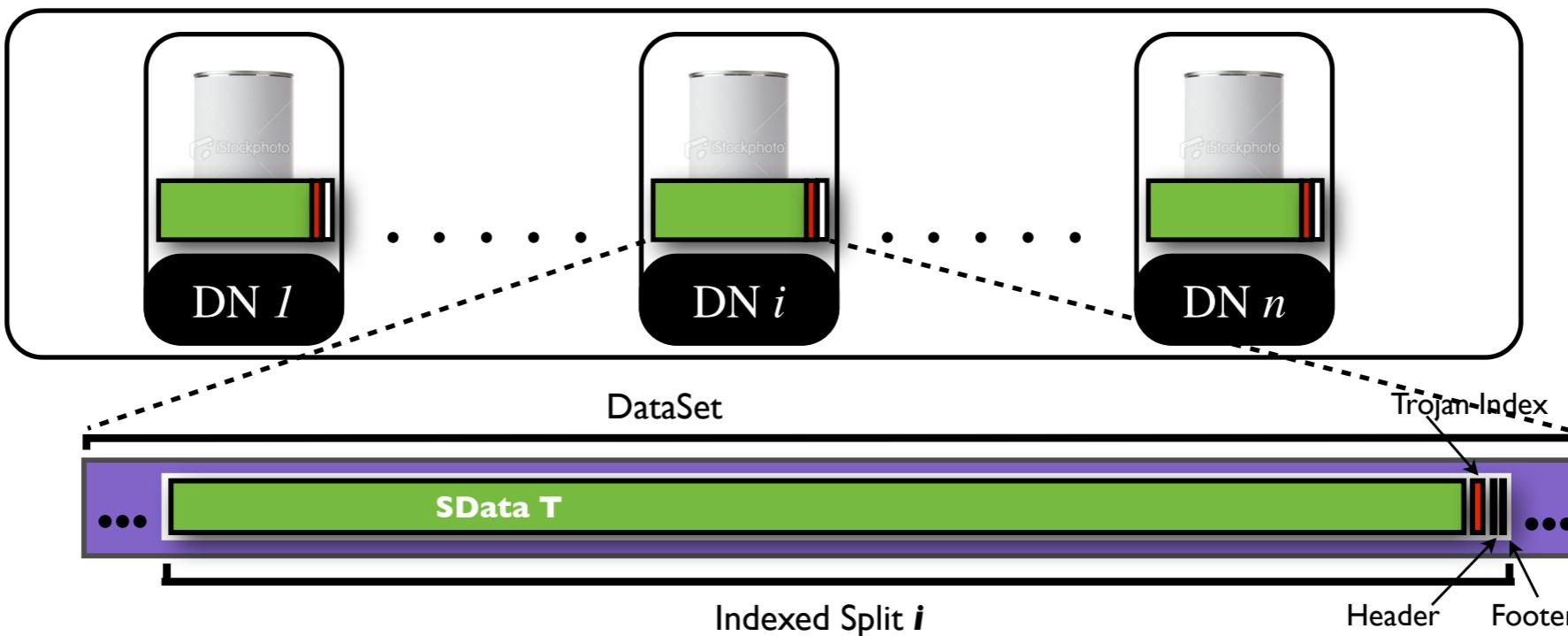


RecordReader

- (1) Read header to obtain $[key_{min}, key_{max}]$ -range of index
- (2) if search key overlaps $[key_{min}, key_{max}]$ -range:

Job Execution

HDFS

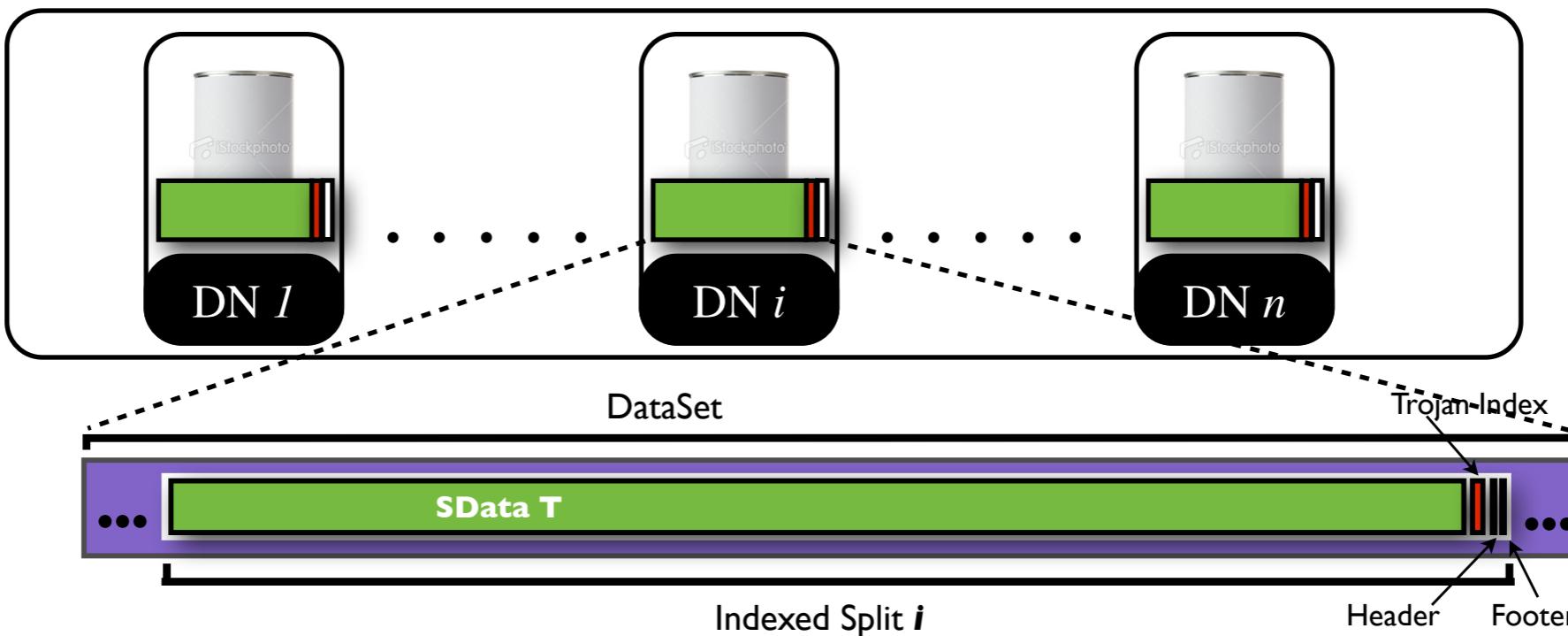


RecordReader

- (1) Read header to obtain $[\text{key}_{\min}, \text{key}_{\max}]$ -range of index
- (2) if search key overlaps $[\text{key}_{\min}, \text{key}_{\max}]$ -range:
- (3) read CSS-tree into main memory

Job Execution

HDFS

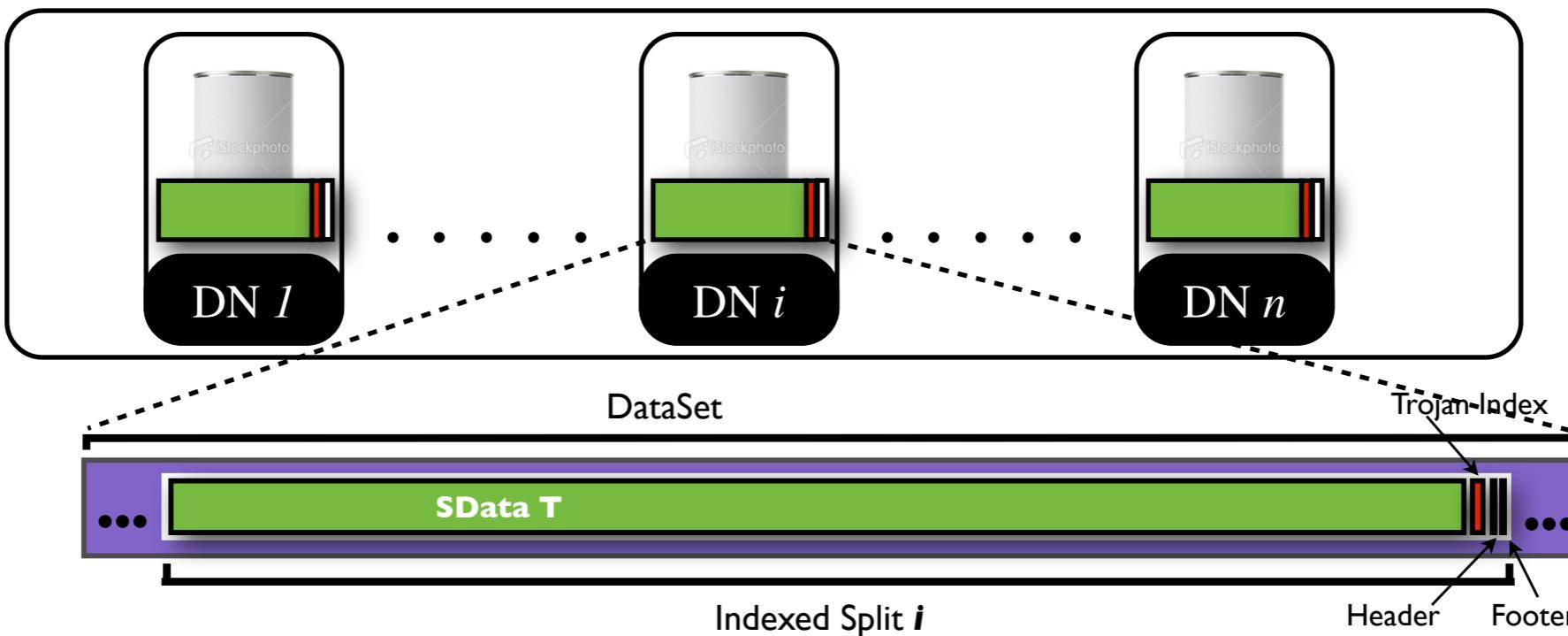


RecordReader

- (1) Read header to obtain $[\text{key}_{\min}, \text{key}_{\max}]$ -range of index
- (2) if search key overlaps $[\text{key}_{\min}, \text{key}_{\max}]$ -range:
 - (3) read CSS-tree into main memory
 - (4) read and pass only qualifying records to map()

Job Execution

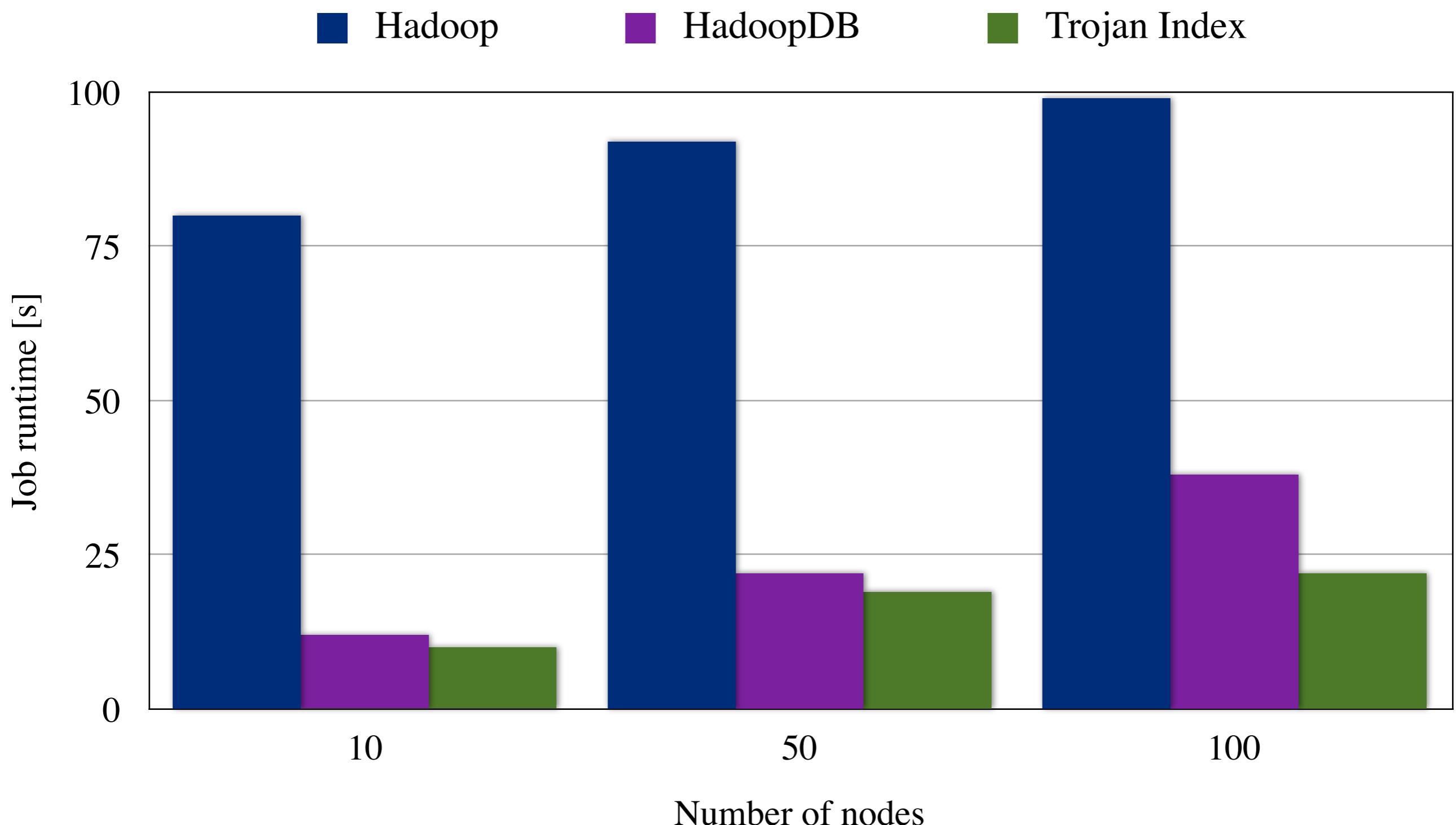
HDFS



RecordReader

- (1) Read header to obtain $[key_{min}, key_{max}]$ -range of index
- (2) **if** search key overlaps $[key_{min}, key_{max}]$ -range:
 - (3) read CSS-tree into main memory
 - (4) read and pass only qualifying records to map()
- (5) **else:** skip this split

Block-Level Indexing Results (Selection Task)



Indexing in MapReduce

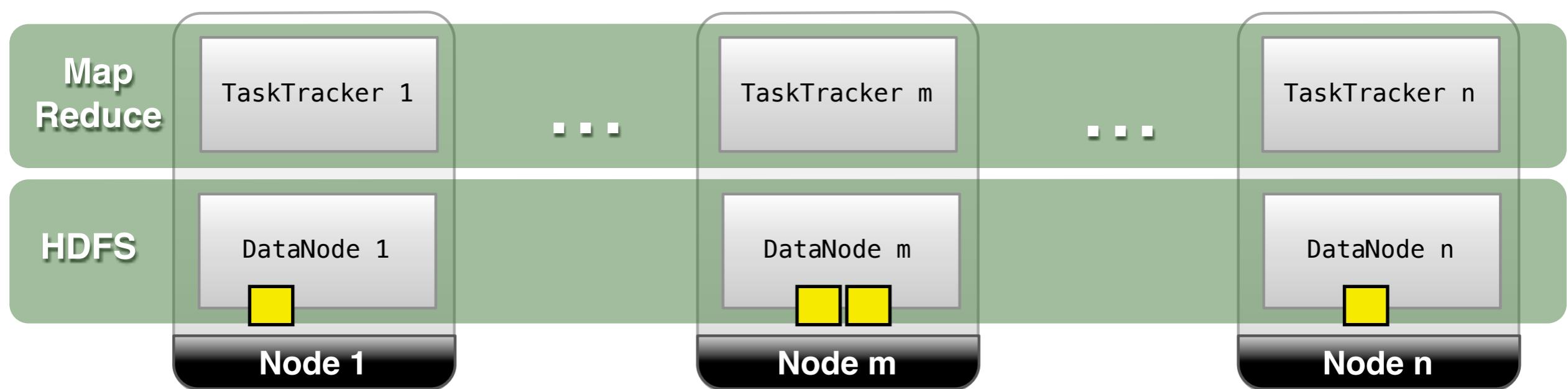
2009	2010	2011
HadoopDB	File Level	Full Text
Still a database		
	Global Sorting	
		Only for high selectivity

Indexing in MapReduce

2009	2010	2011	2010
HadoopDB	File Level	Full Text	Trojan
Still a database			
	Global Sorting		
		Only for high selectivity	

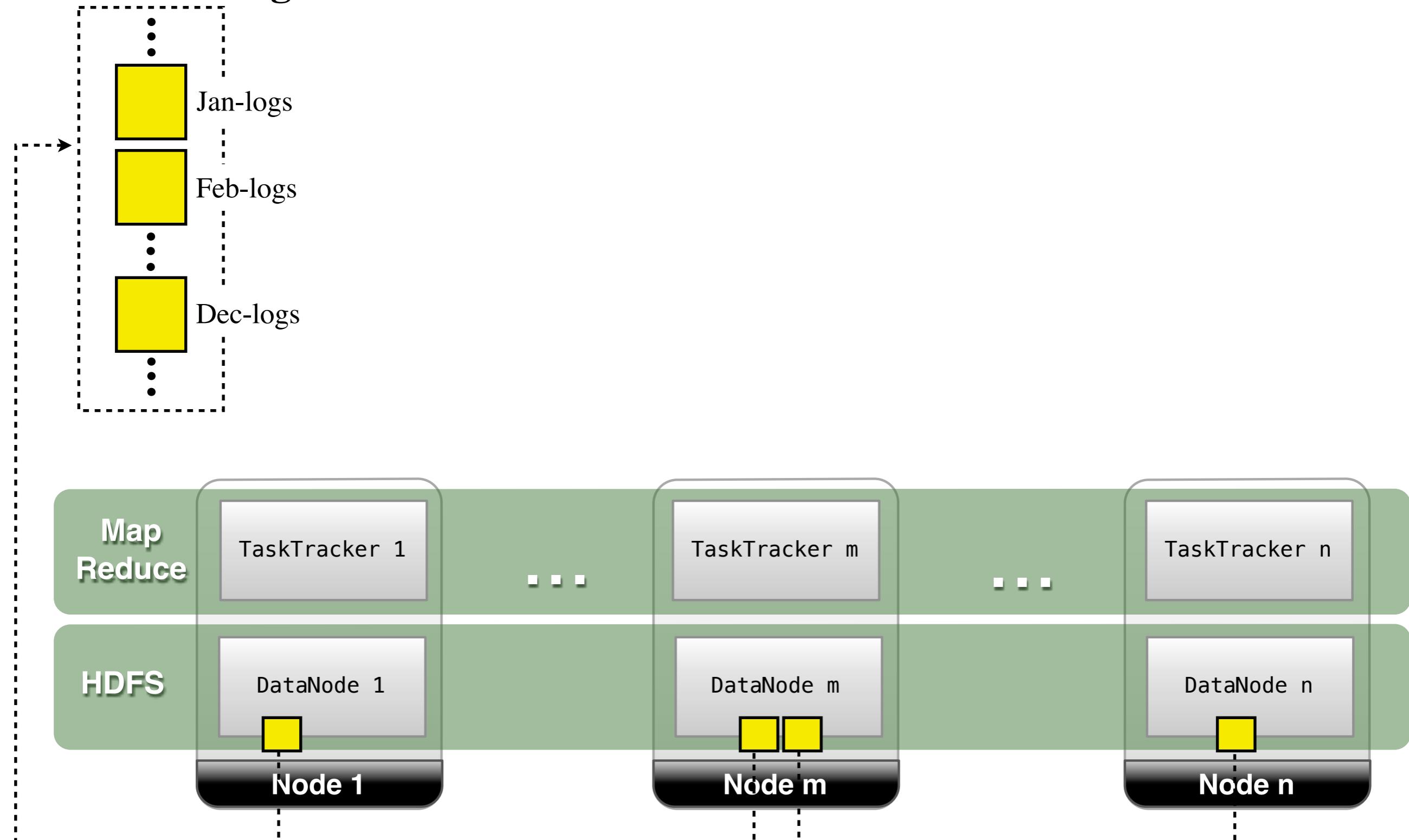
Can We Exploit them
All Together?

Putting All Together



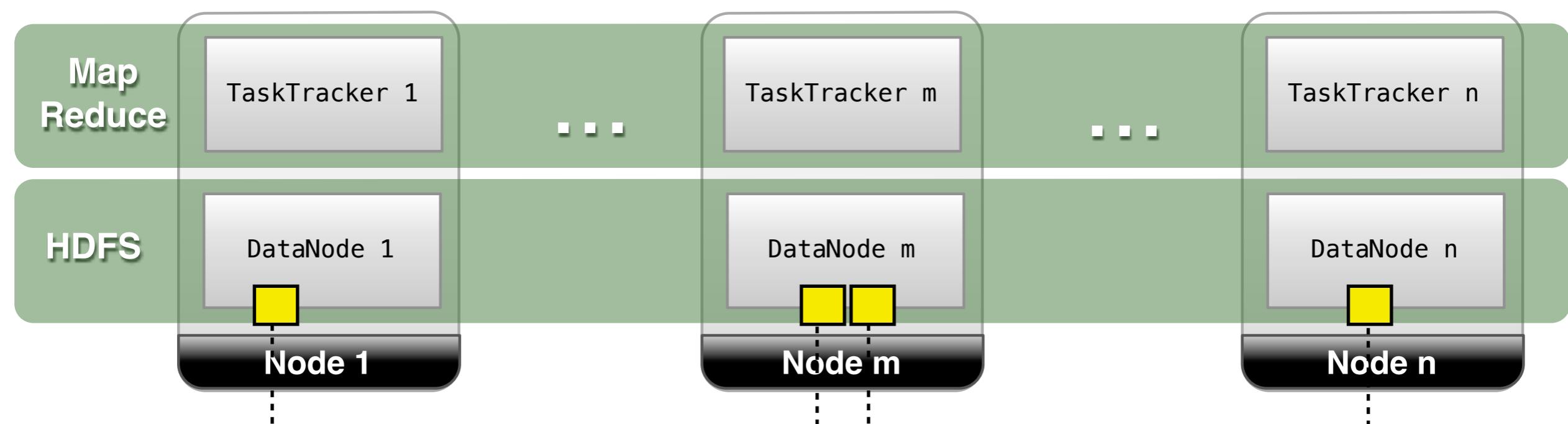
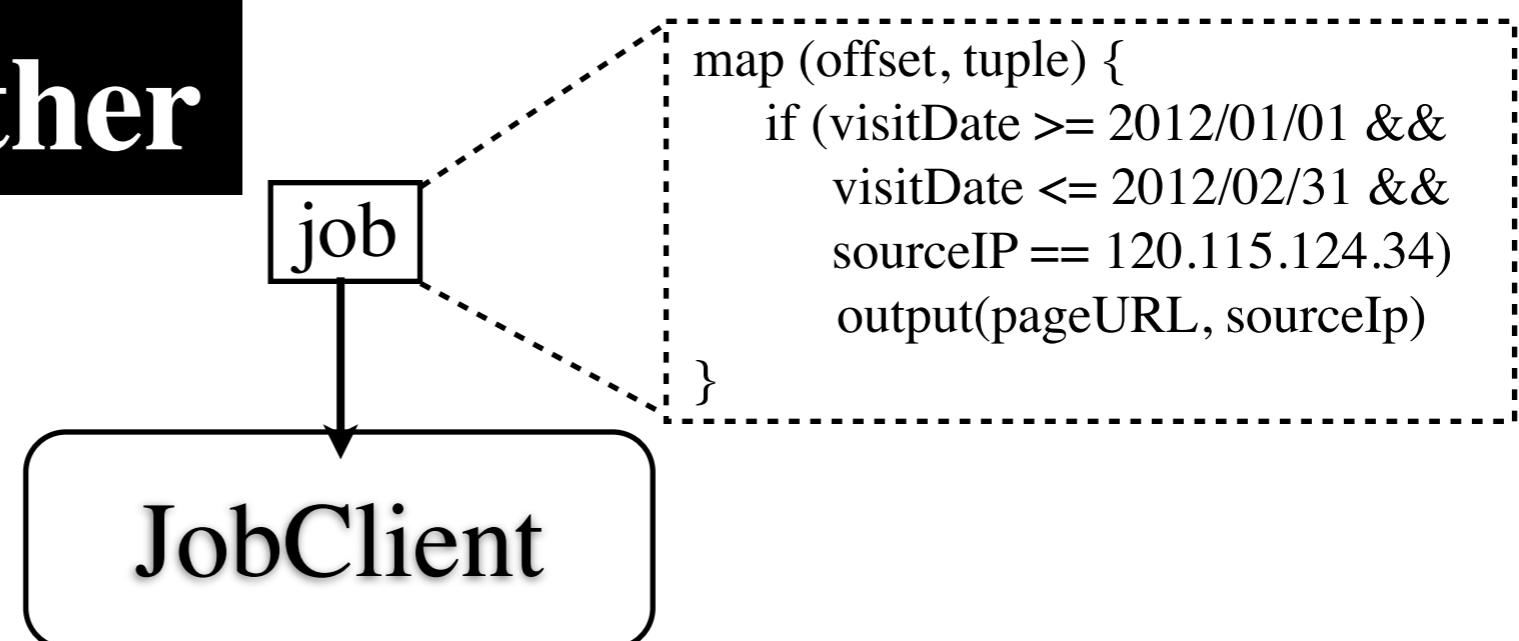
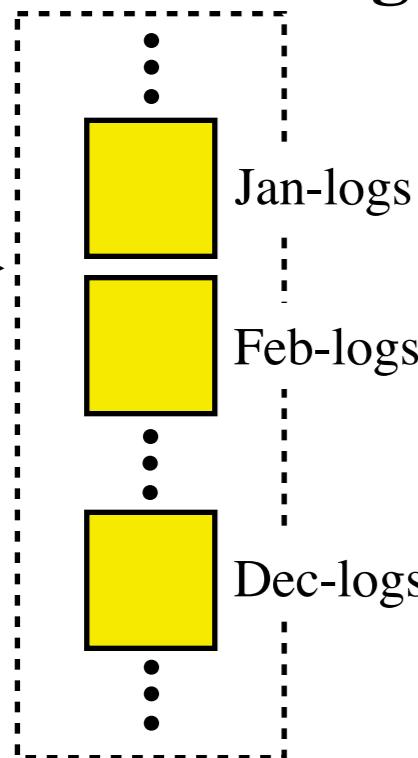
Putting All Together

UserVisits Log



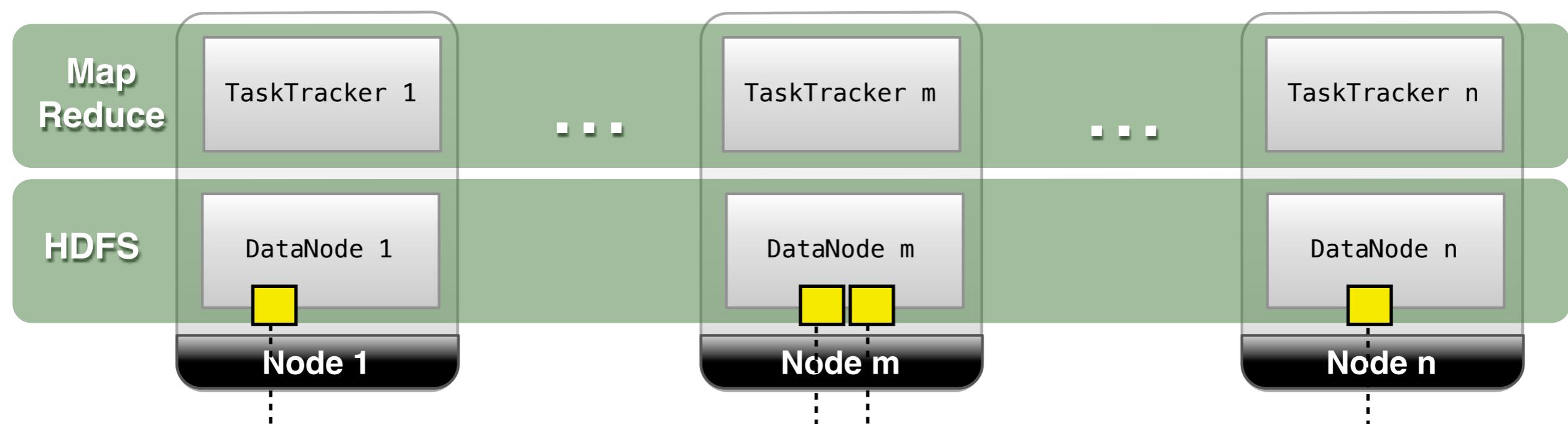
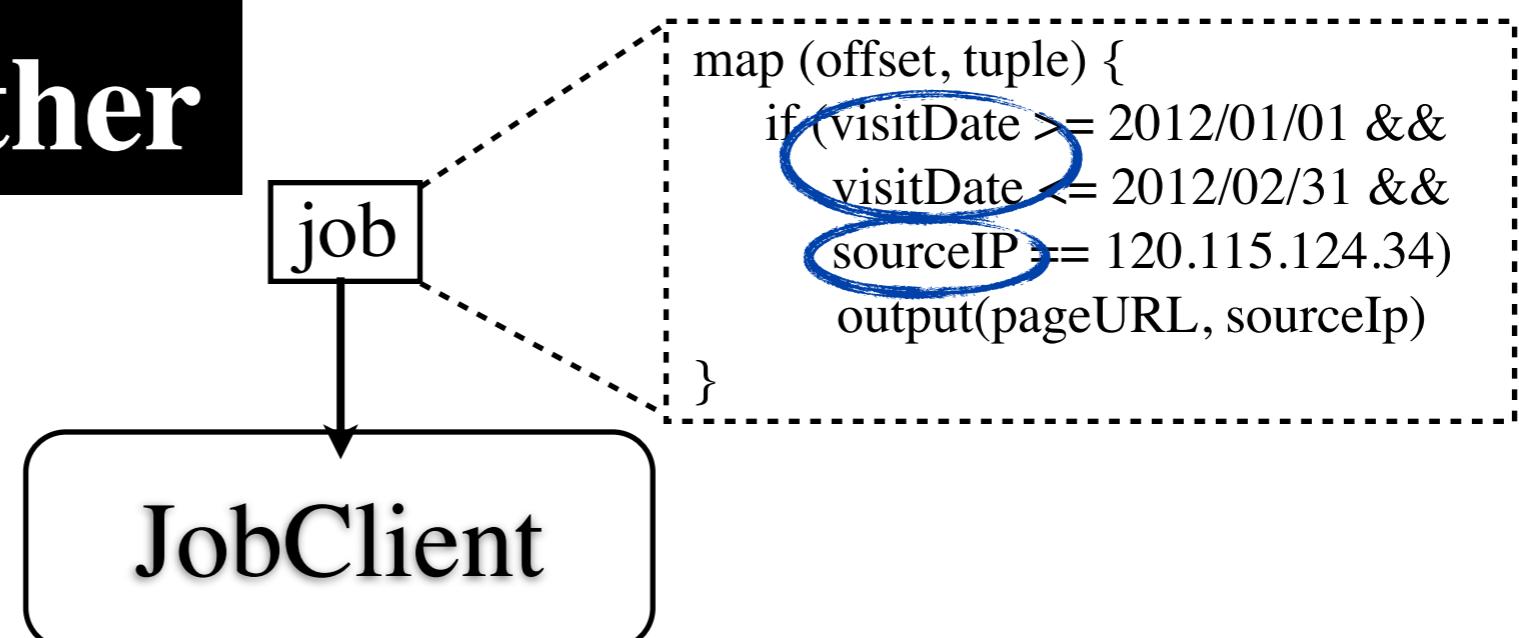
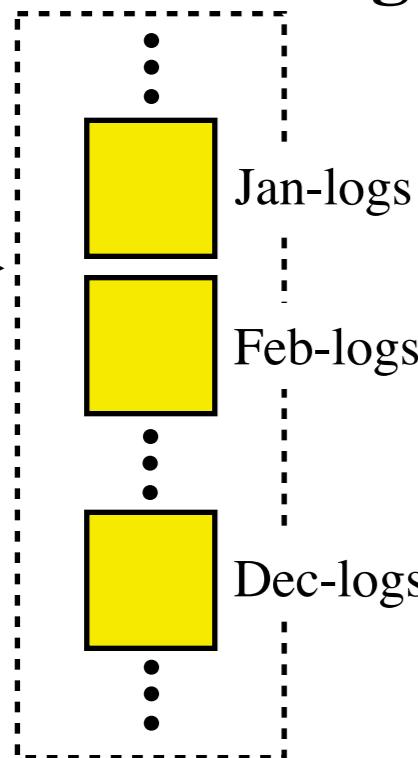
Putting All Together

UserVisits Log



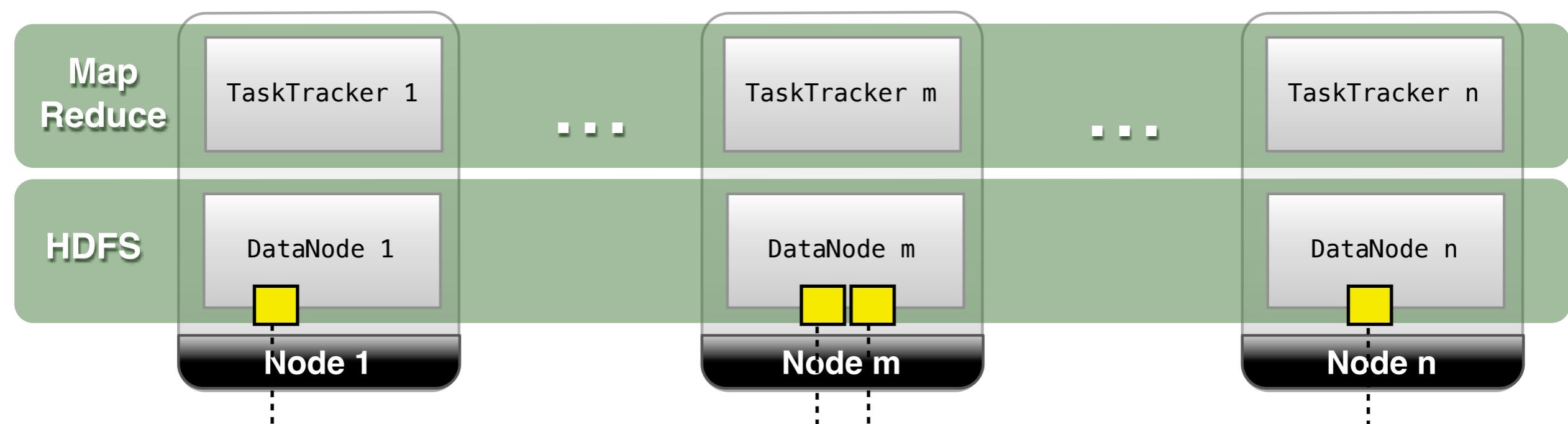
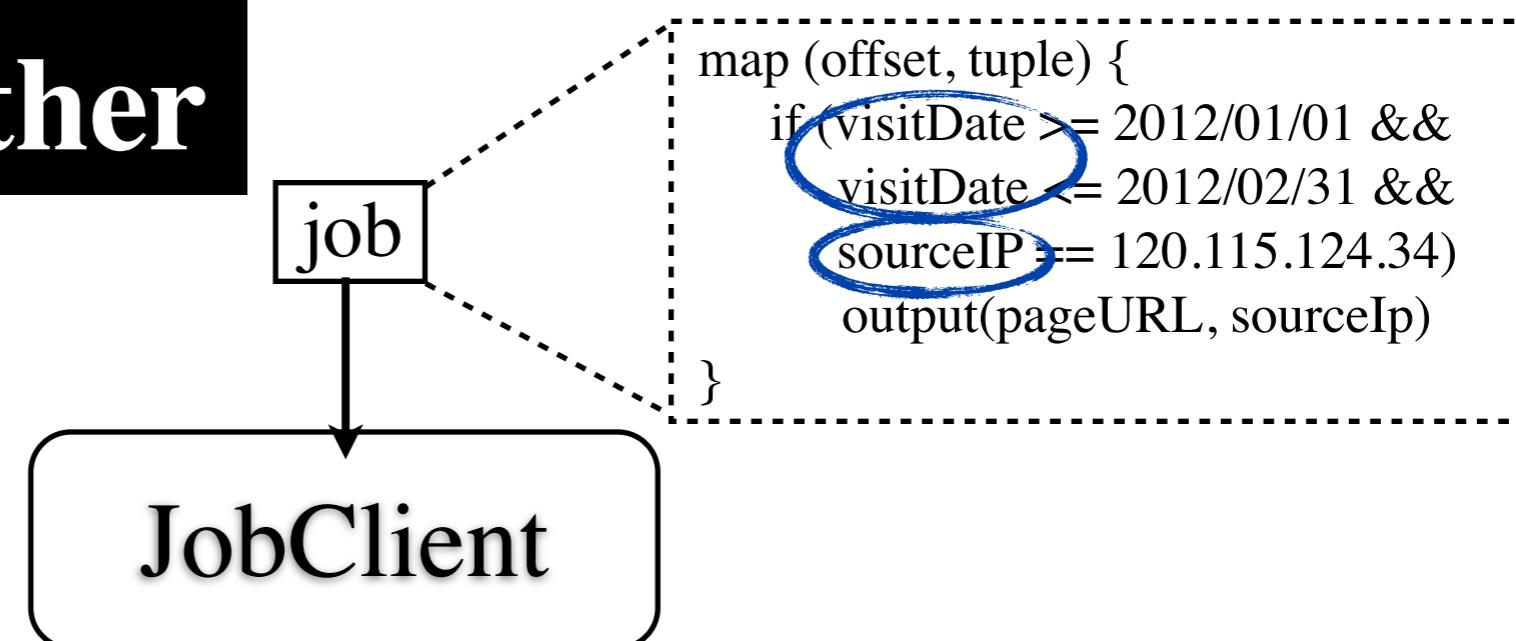
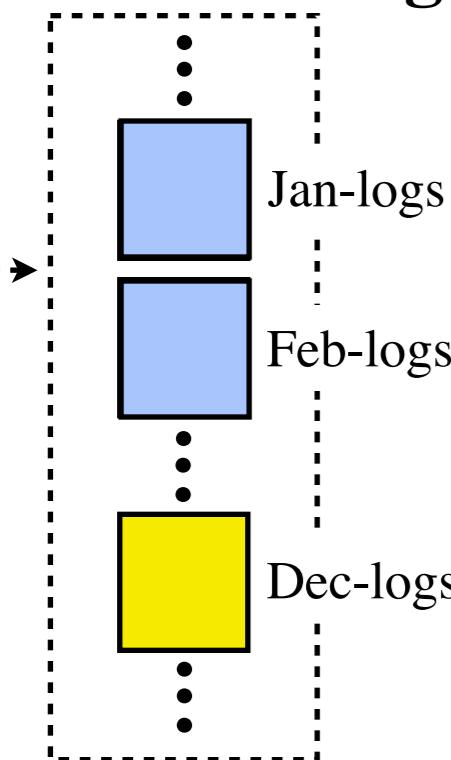
Putting All Together

UserVisits Log



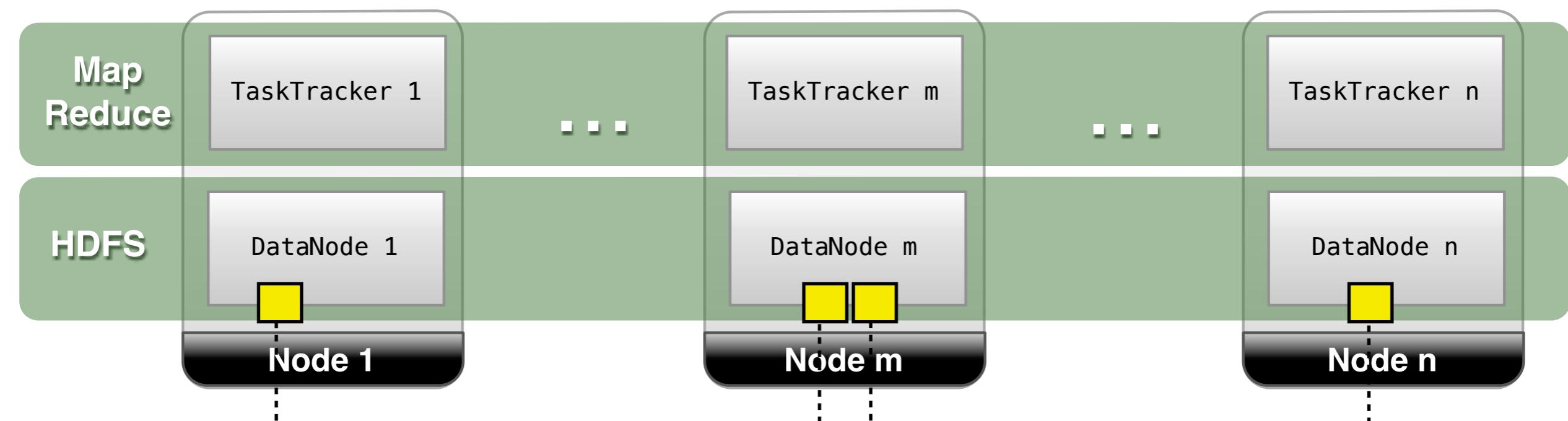
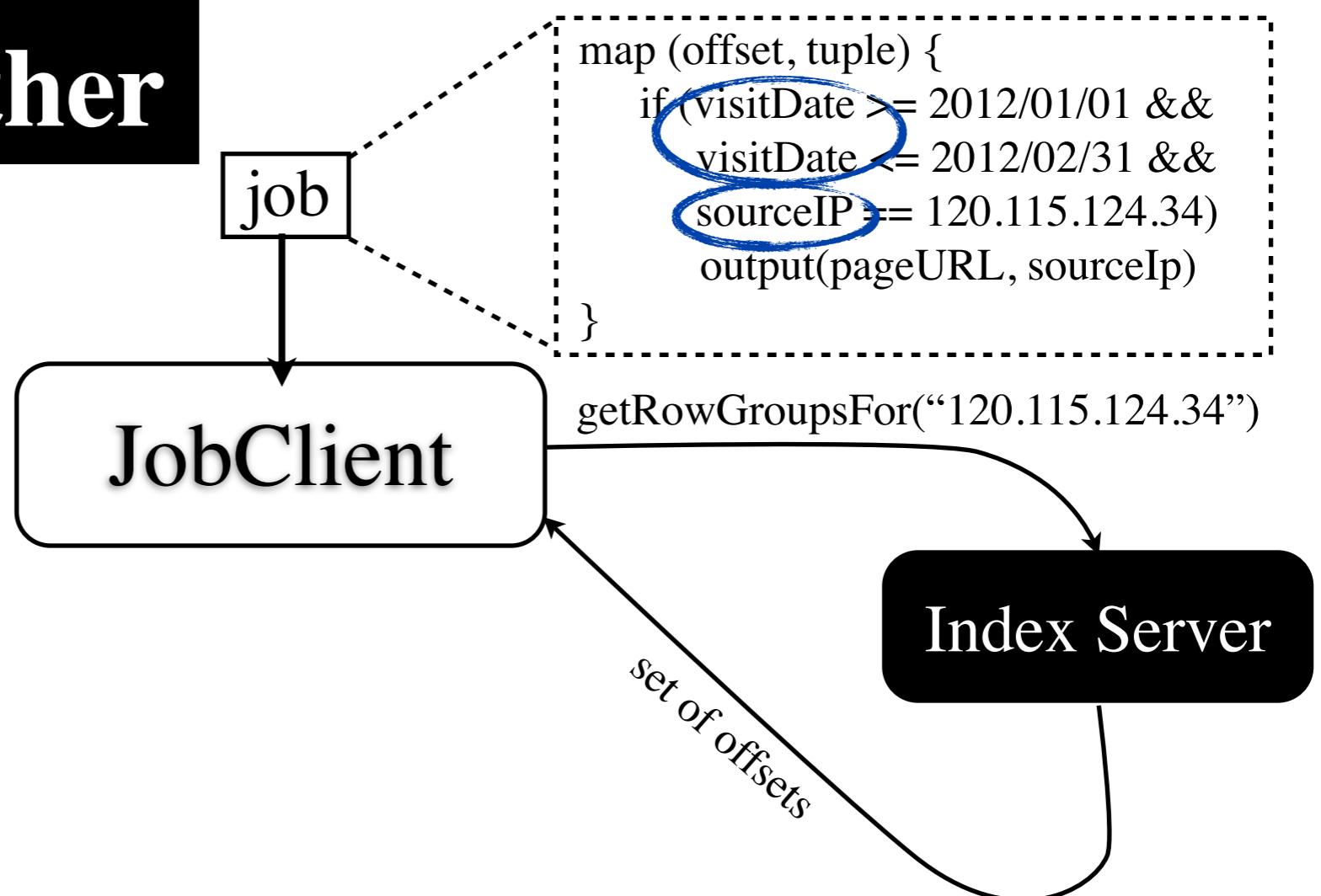
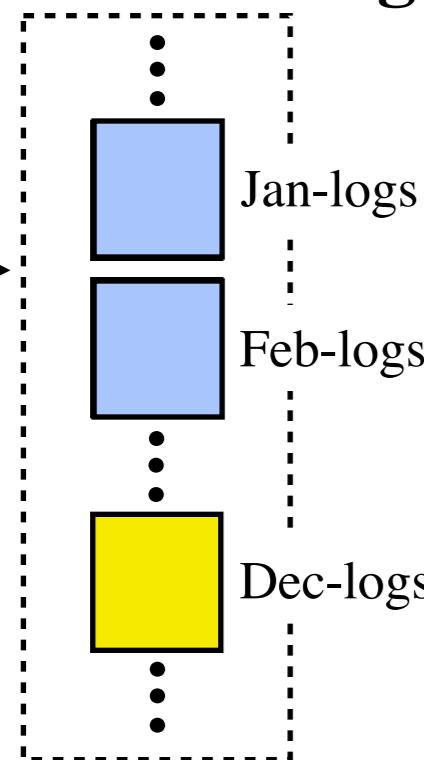
Putting All Together

UserVisits Log



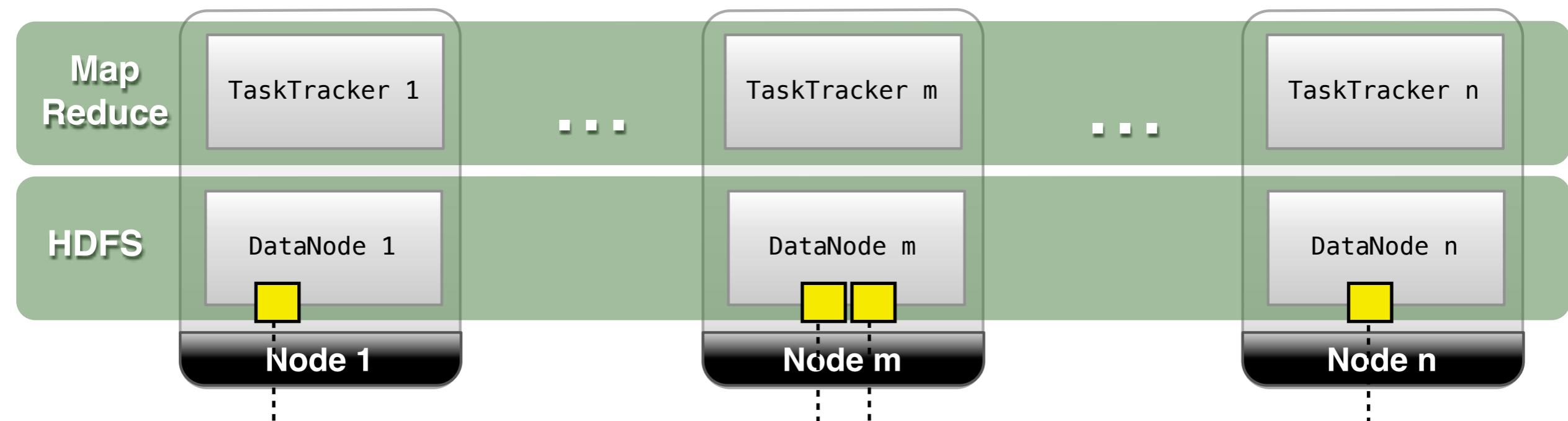
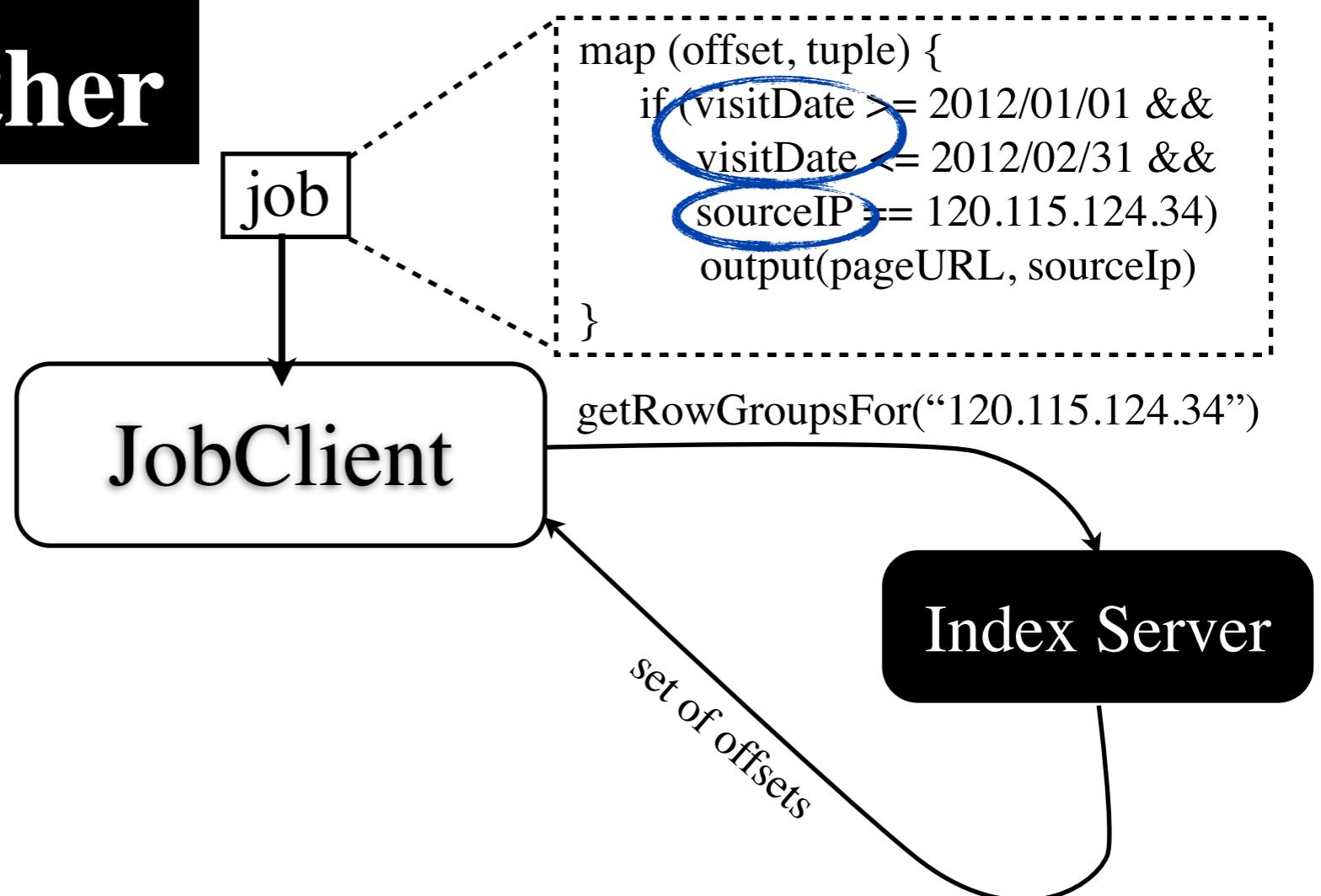
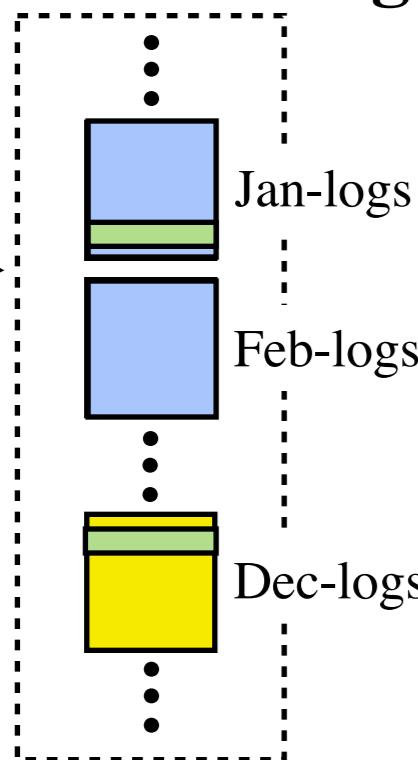
Putting All Together

UserVisits Log



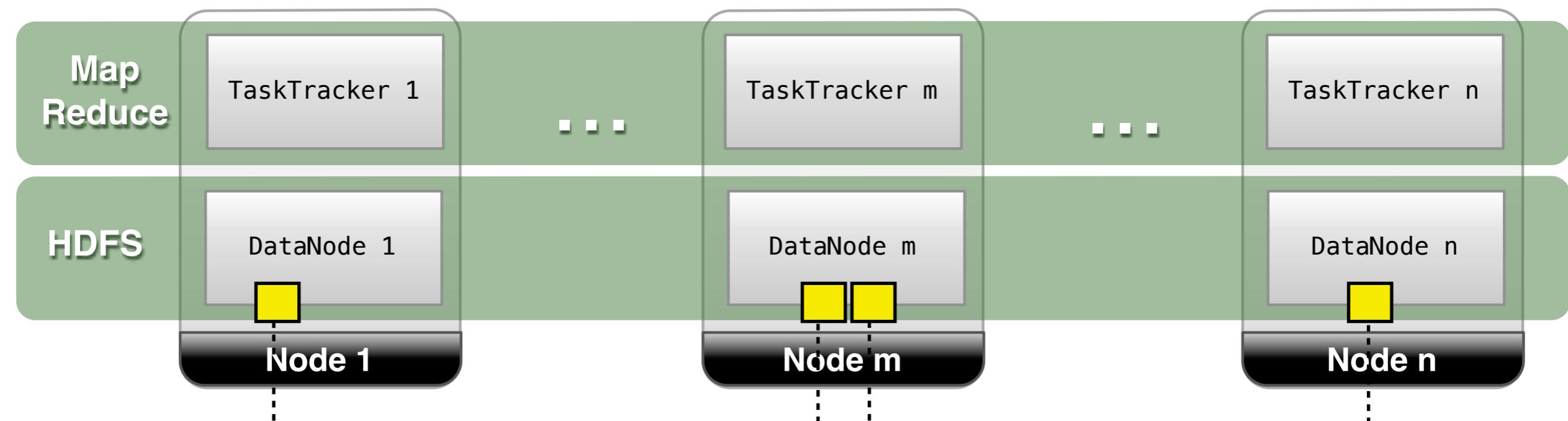
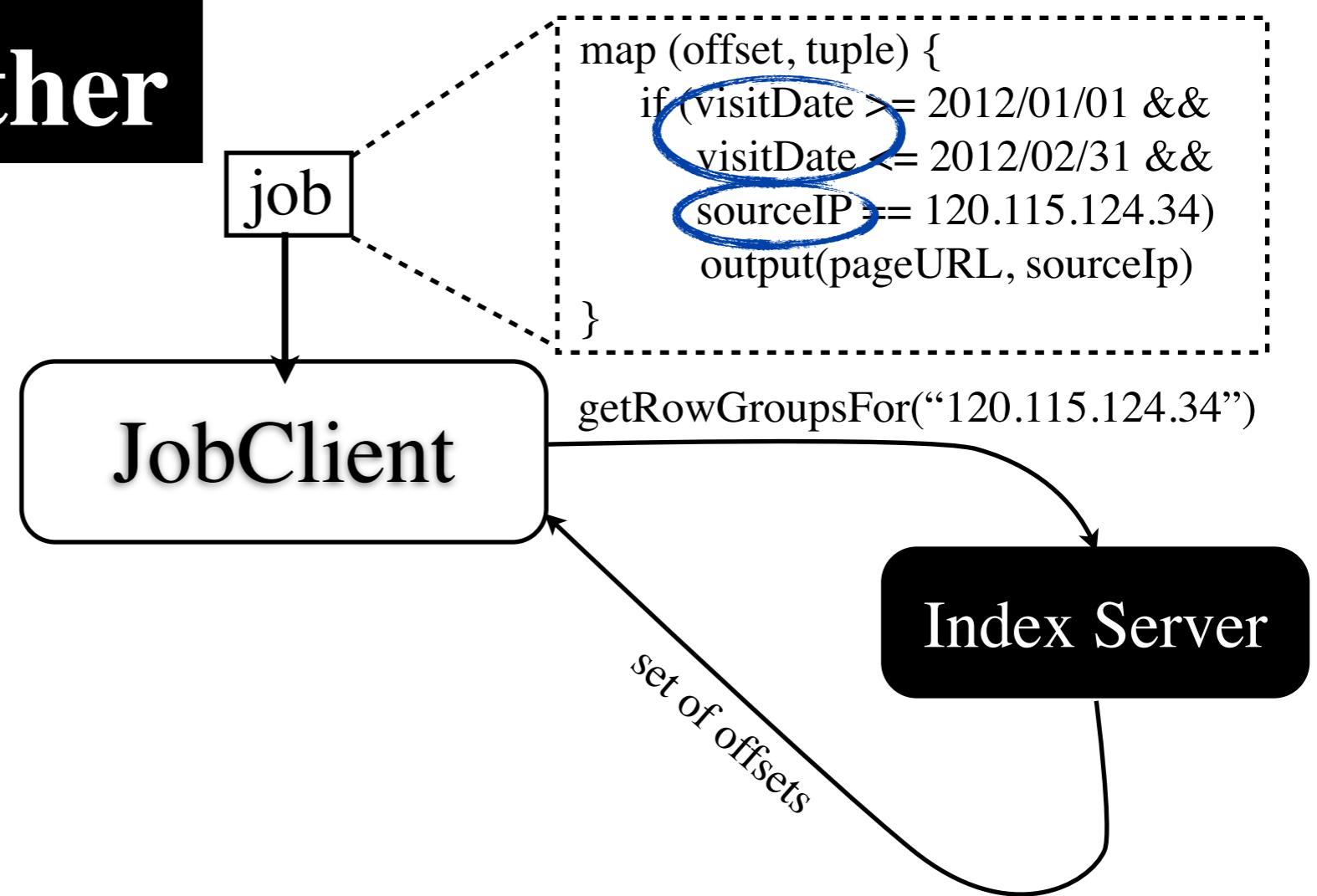
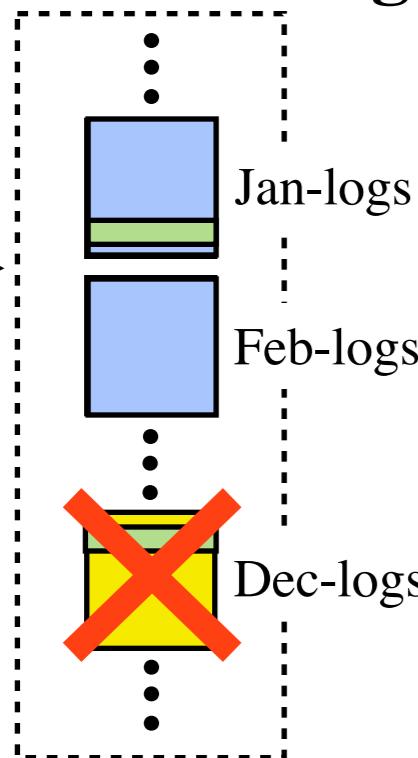
Putting All Together

UserVisits Log



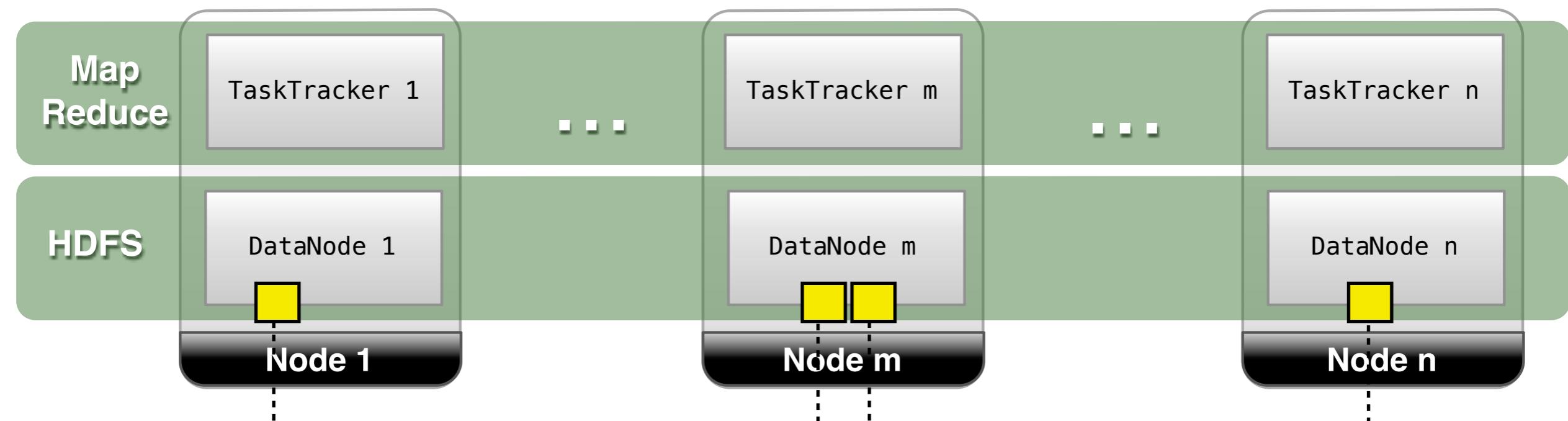
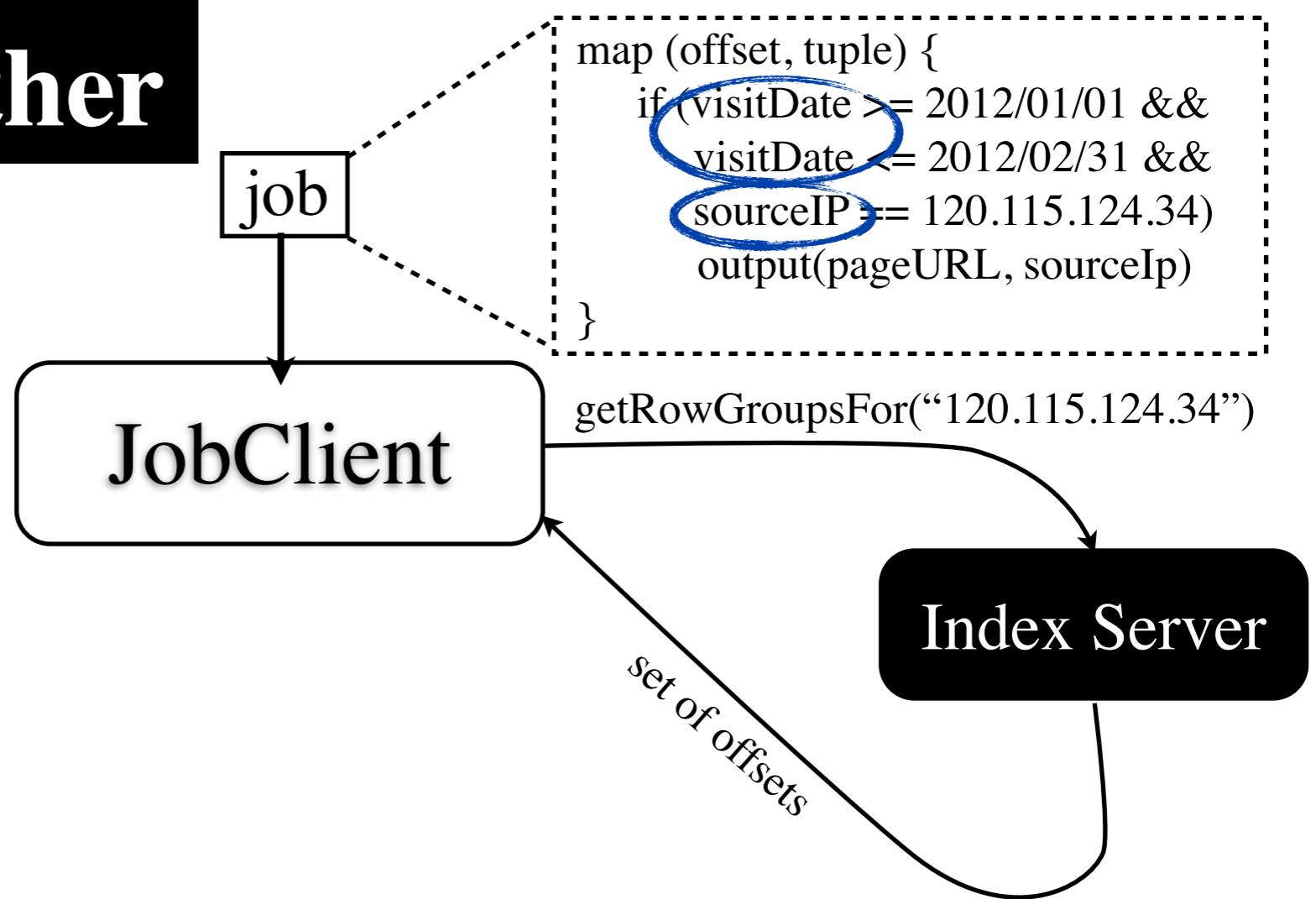
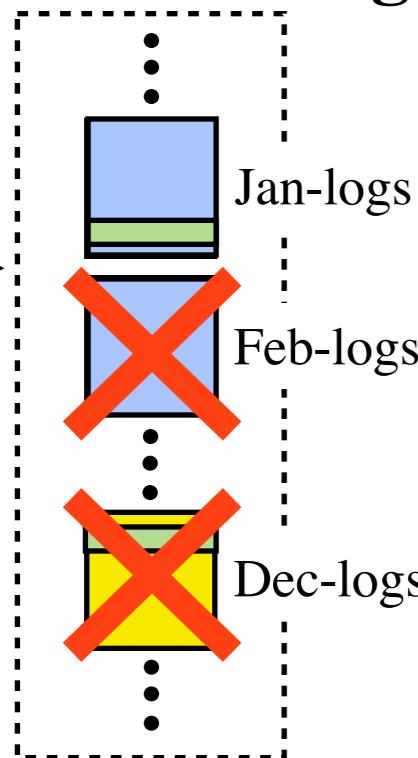
Putting All Together

UserVisits Log

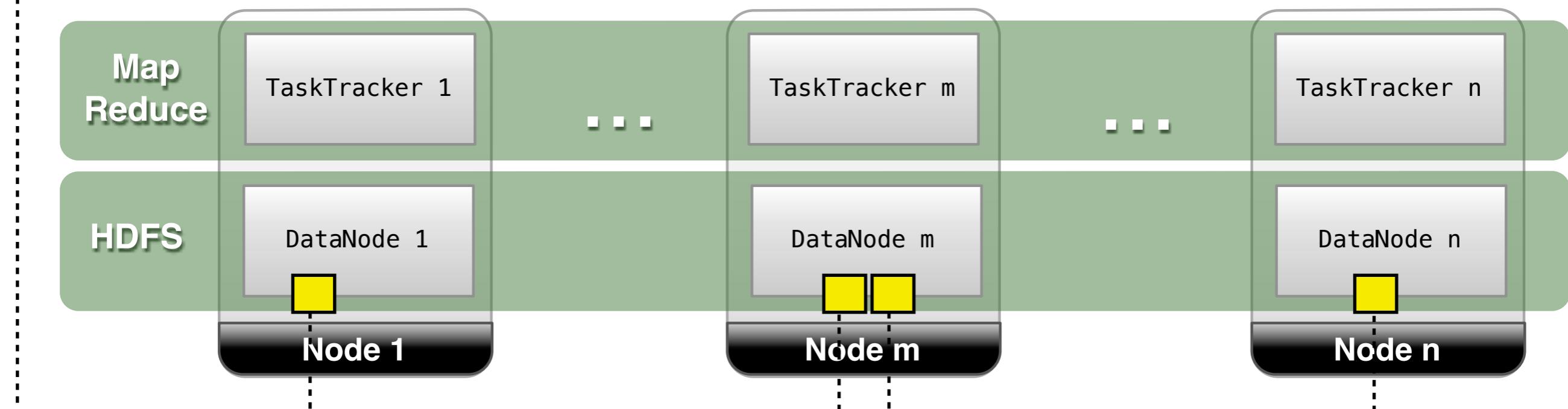
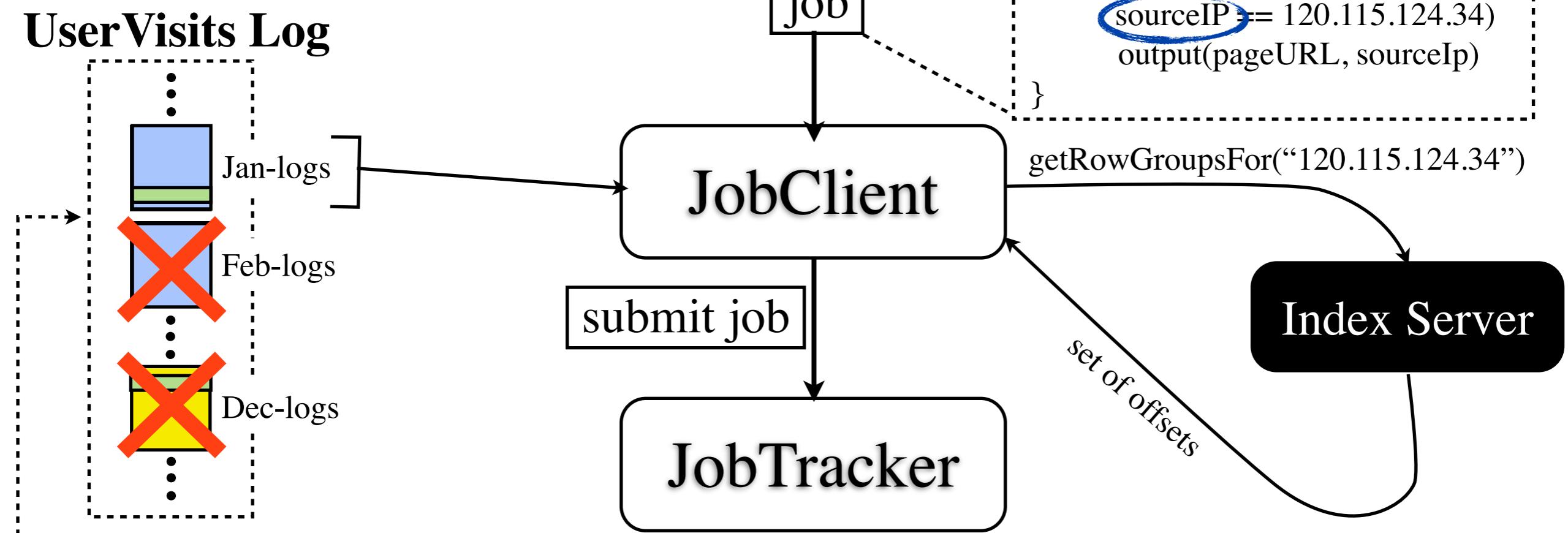


Putting All Together

UserVisits Log

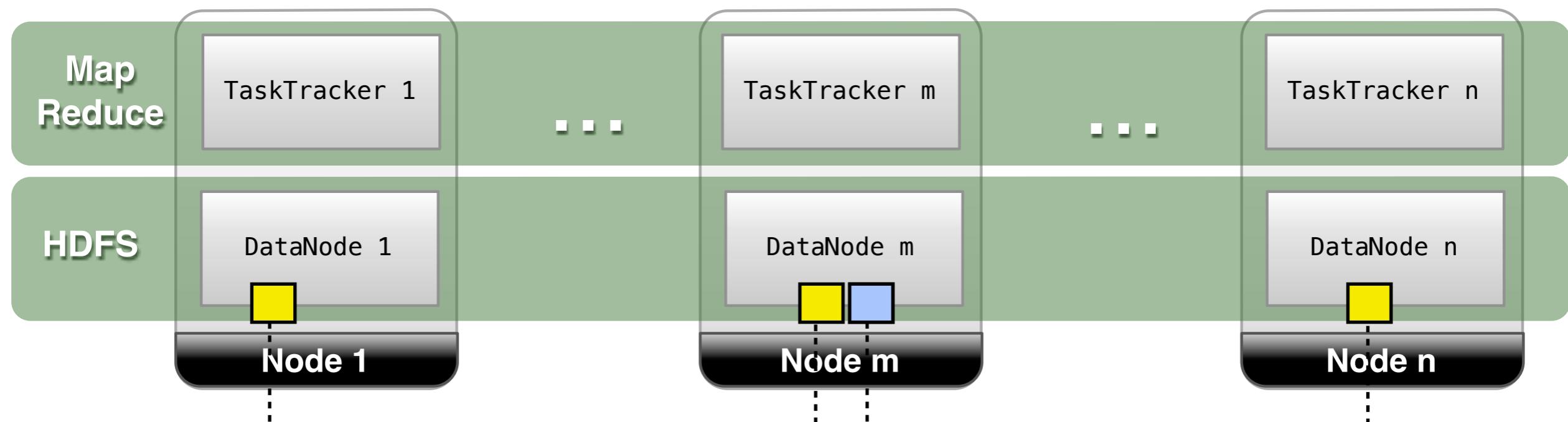
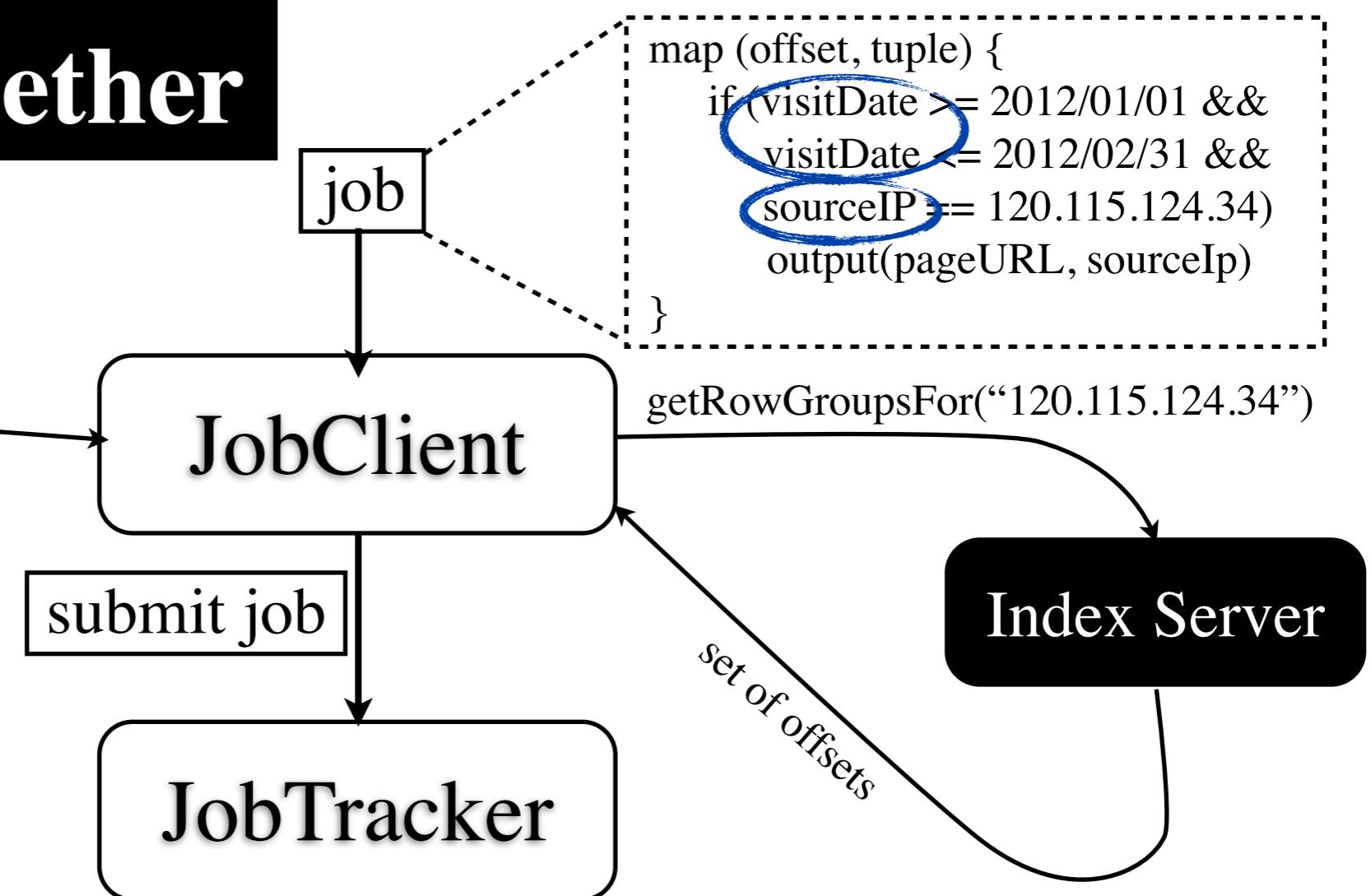
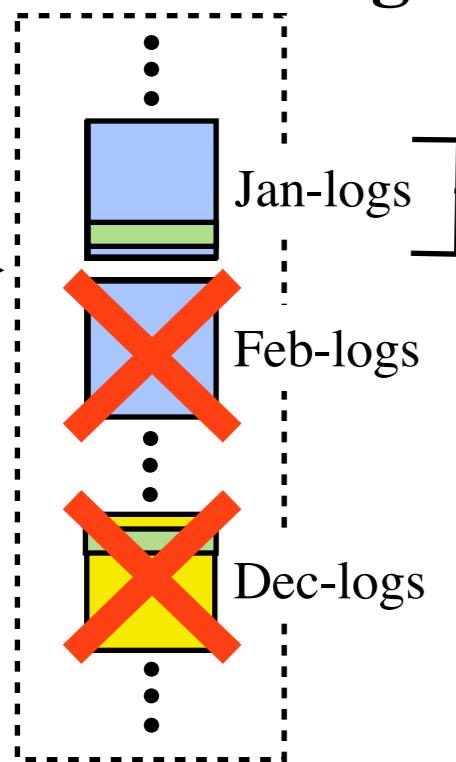


Putting All Together



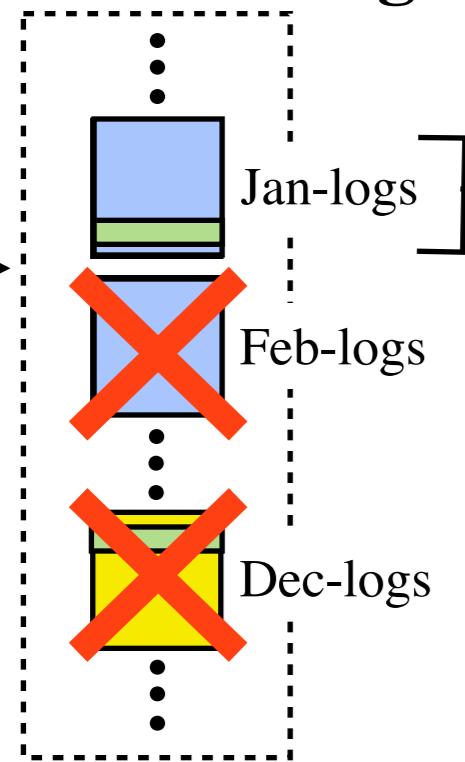
Putting All Together

UserVisits Log



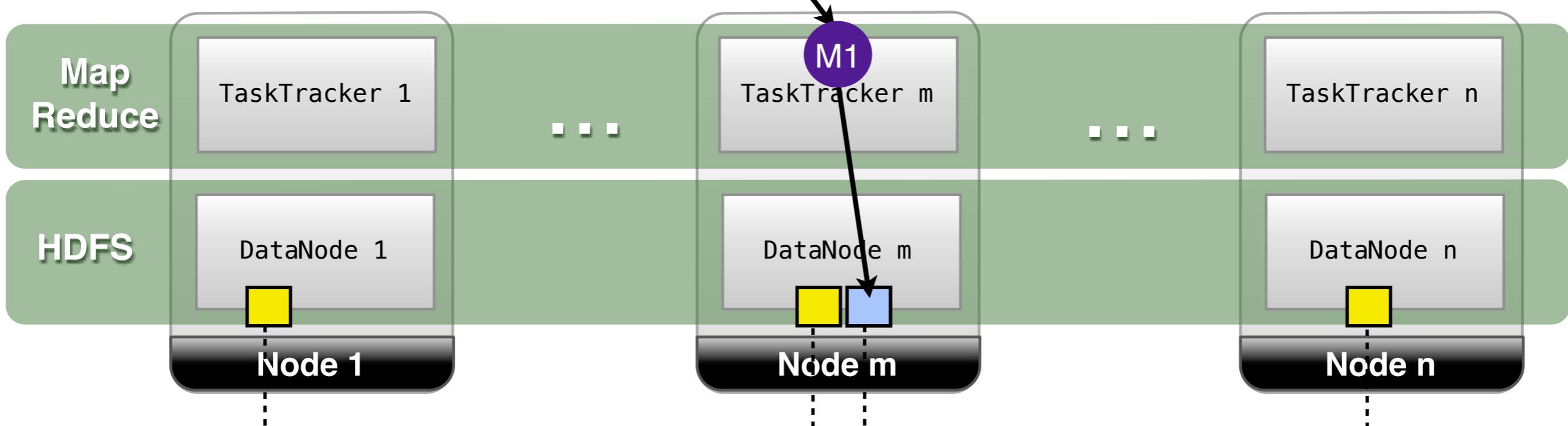
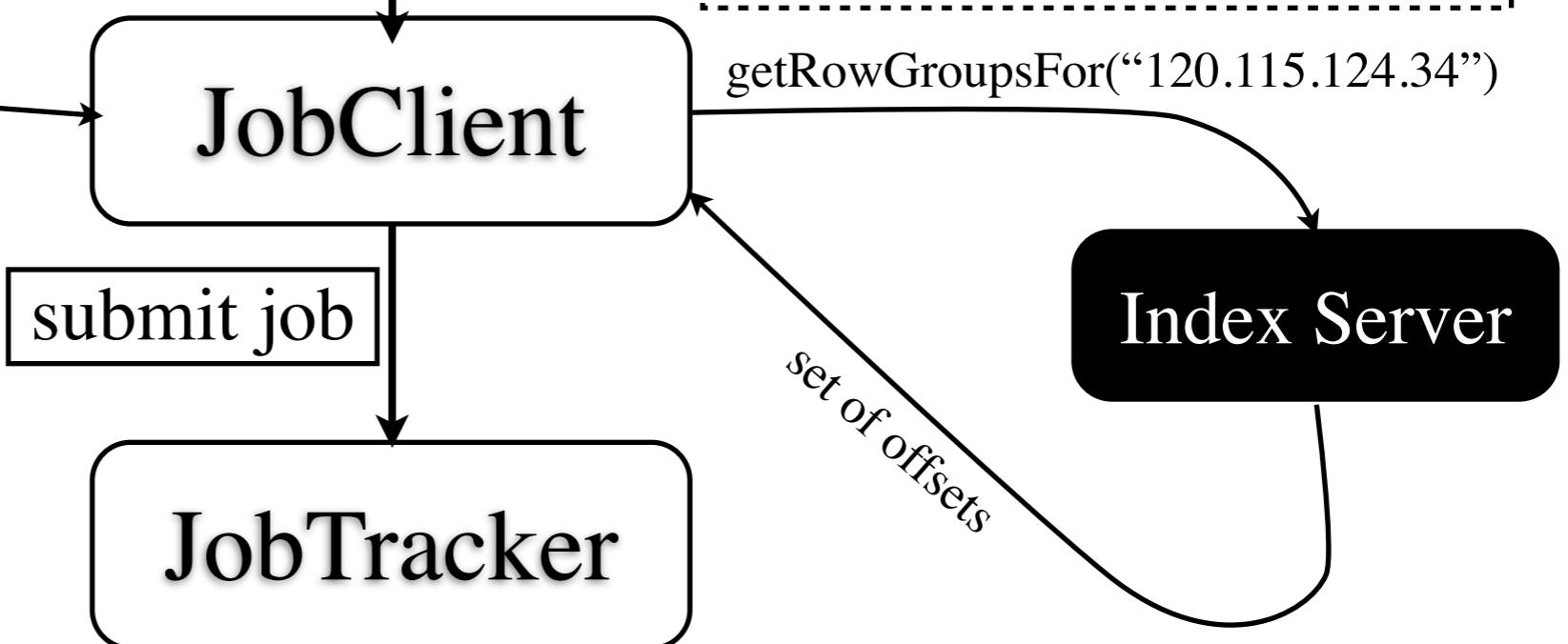
Putting All Together

UserVisits Log

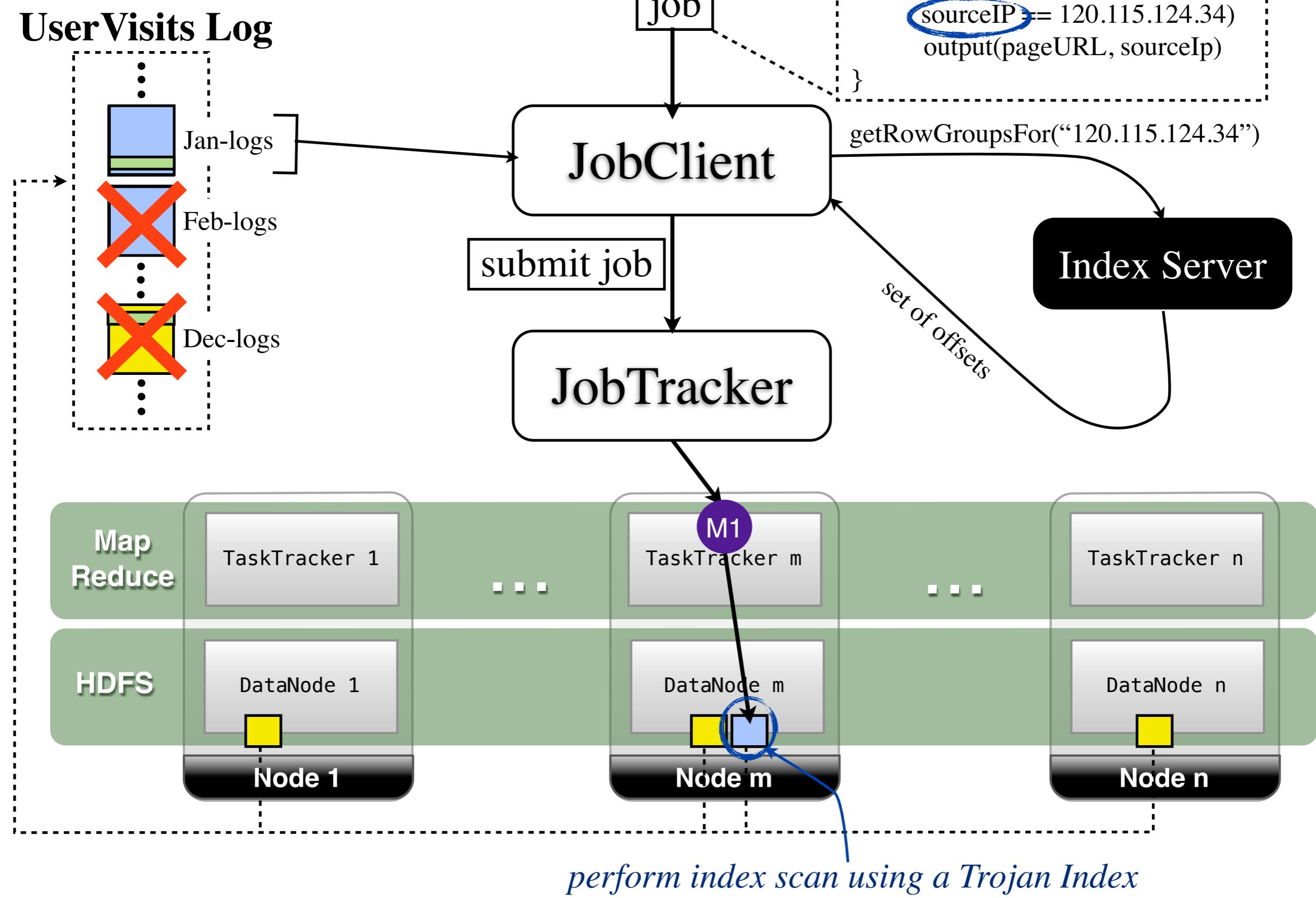


```
map (offset, tuple) {  
    if(visitDate >= 2012/01/01 &&  
        visitDate <= 2012/02/31 &&  
        sourceIP == 120.115.124.34)  
    output(pageURL, sourceIp)  
}
```

job



Putting All Together



Still,

Still,

Long index creation times

Still,

Long index creation times

&

**One clustered index per
dataset**

Hadoop Aggressive Indexing Library (HAIL)

[J. Dittrich, J. Quiané, S. Richter, S. Schuh, A. Jindal, J. Schad: Only Aggressive Elephants are Fast Elephants. PVLDB 2012]

Inspired by Trojan Data Layouts¹

¹[A. Jindal, J. Quiané, J. Dittrich: Trojan Data Layouts: Right Shoes for a Running Elephant. SoCC 2011]

Indexing in MapReduce

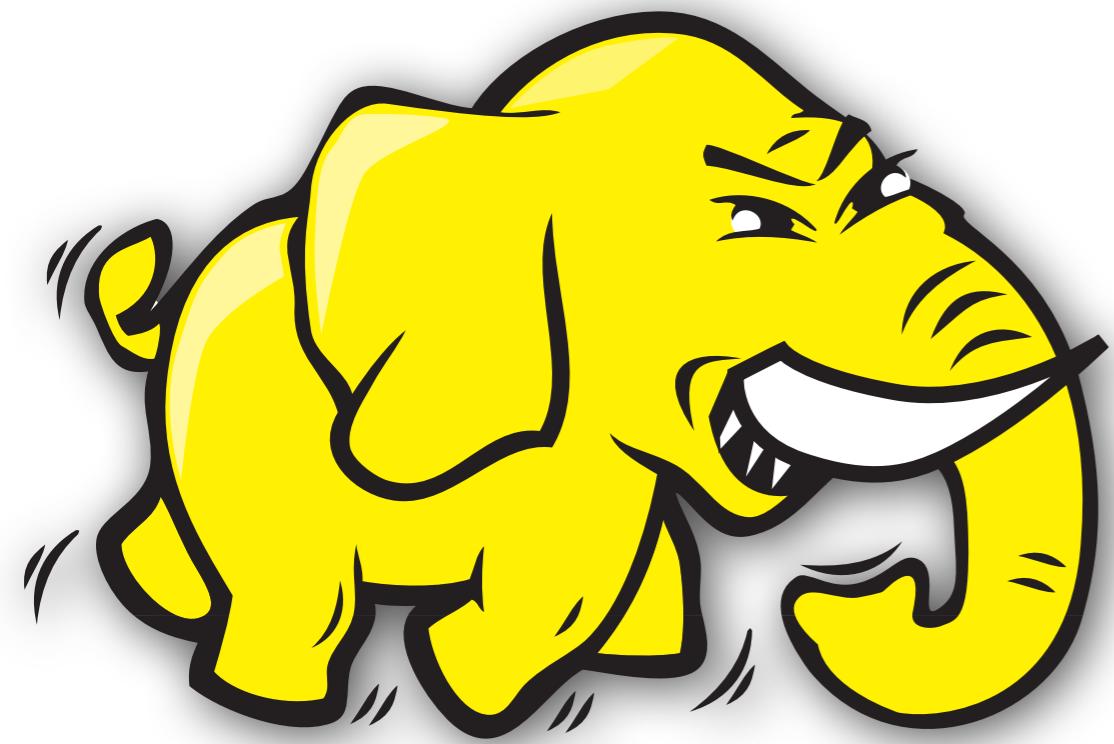
2009	2010	2011	2010	2012
HadoopDB	File Level	Full Text	Trojan	HAIL
Still a database				
	Global Sorting			
		Only for high selectivity		

Indexing in MapReduce

2009	2010	2011	2010	2012
HadoopDB	File Level	Full Text	Trojan	HAIL
Still a database				
	Global Sorting			
		Only for high selectivity		
High upload time	High upload time	High upload time	High upload time	
Single Index	Single Index	Single Index	Single Index	

Indexing in MapReduce

2009	2010	2011	2010	2012
HadoopDB	File Level	Full Text	Trojan	HAIL
Still a database				
	Global Sorting			
		Only for high selectivity		
High upload time	High upload time	High upload time	High upload time	
Single Index	Single Index	Single Index	Single Index	



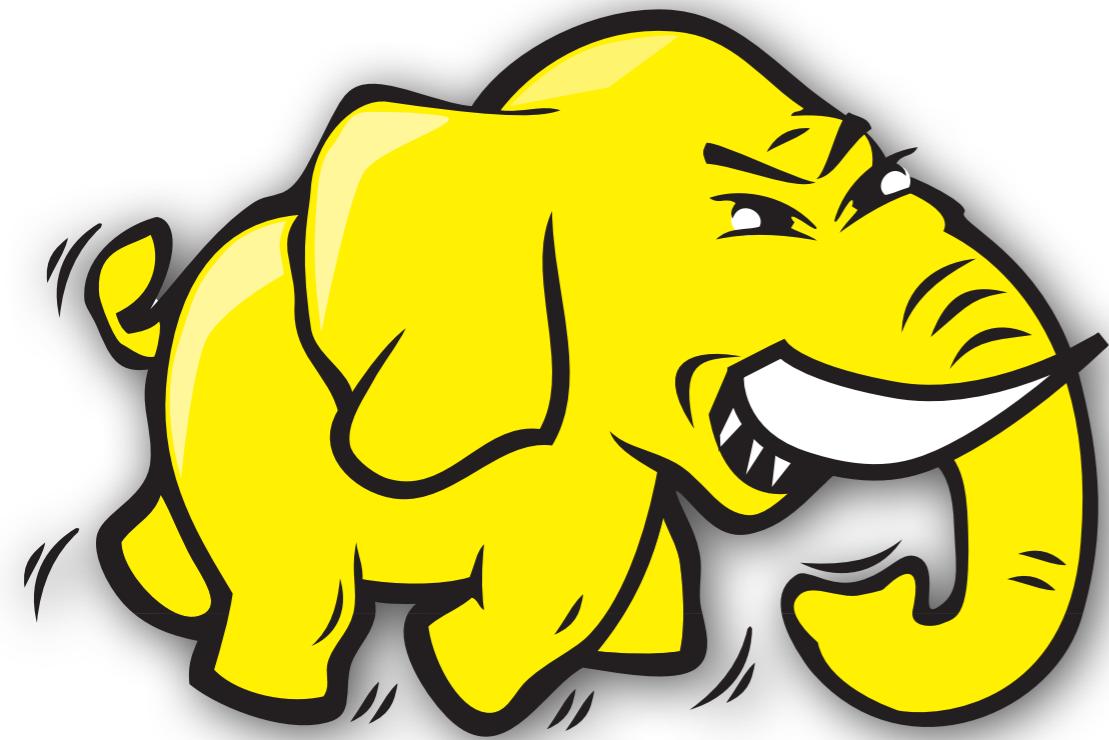
TALK:

Only Aggressive Elephants are Fast Elephants

Wednesday August 29th

11:30 a.m. at the Convention Lower Hall 2

(Research Session 13: MapReduce II)



TALK:

Only Aggressive Elephants are Fast Elephants

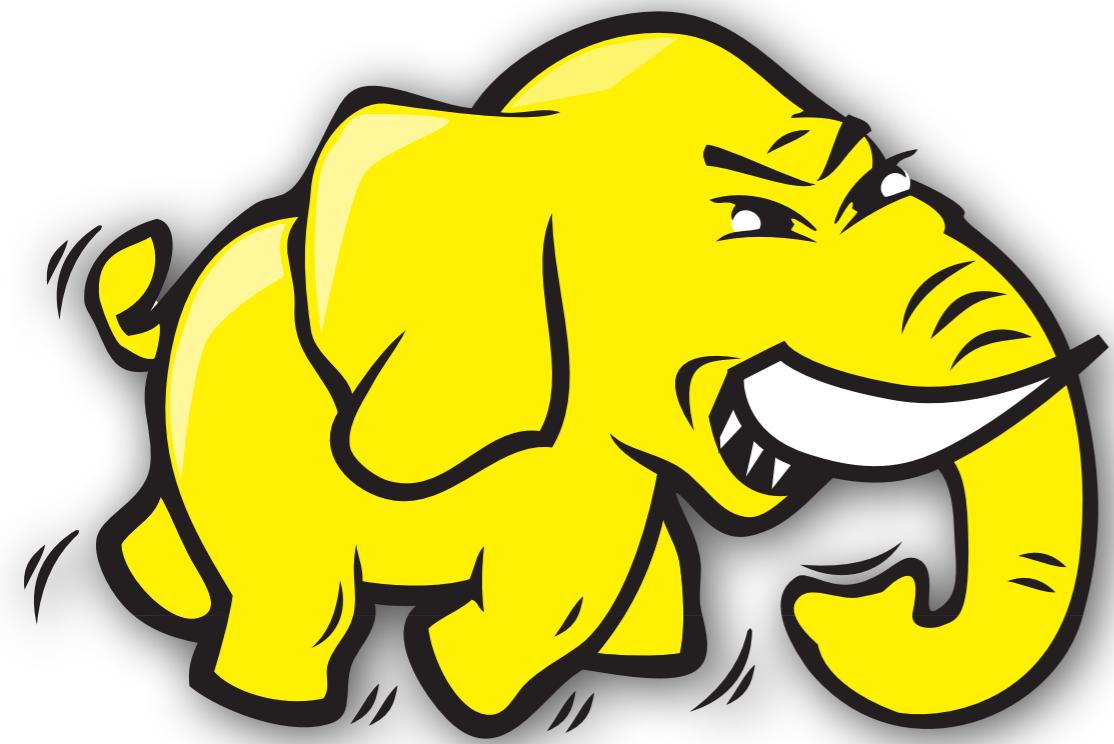
Wednesday August 29th

11:30 a.m. at the Convention Lower Hall 2

(Research Session 13: MapReduce II)

Invisible Index Creation Times:

up to 7.3 times faster than Hadoop++



TALK:

Only Aggressive Elephants are Fast Elephants

Wednesday August 29th

11:30 a.m. at the Convention Lower Hall 2

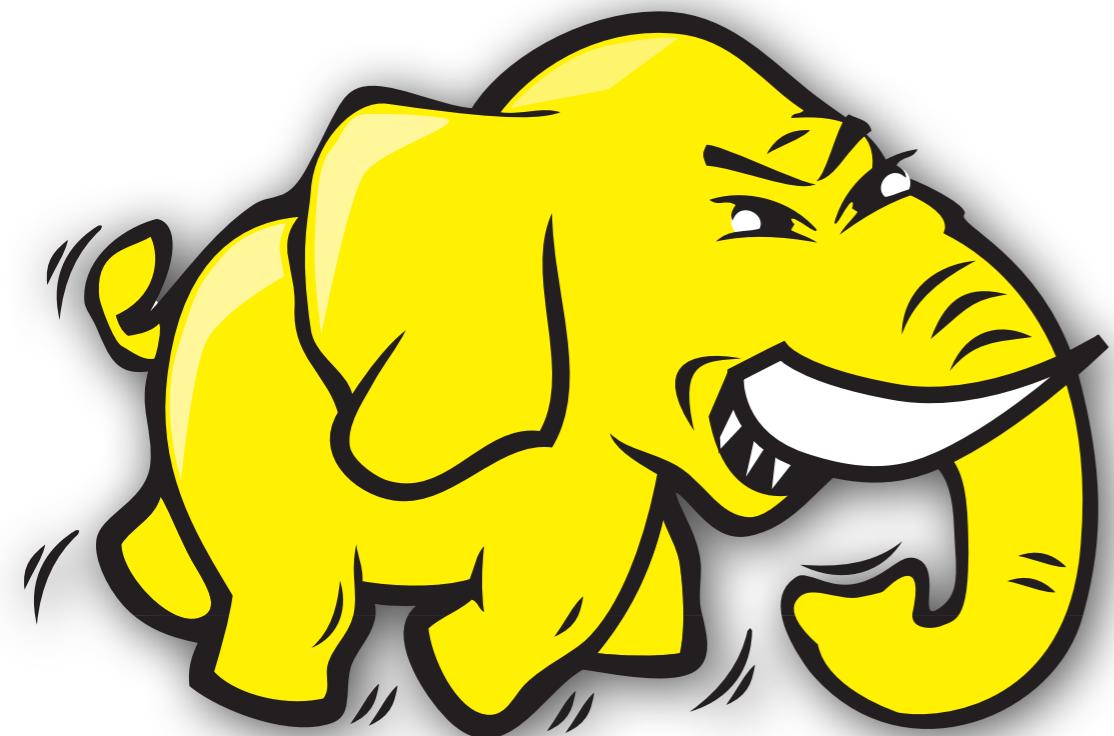
(Research Session 13: MapReduce II)

Invisible Index Creation Times:

up to 7.3 times faster than Hadoop++

Fast Data Upload:

up to 1.6 times faster than Hadoop



TALK:

Only Aggressive Elephants are Fast Elephants

Wednesday August 29th

11:30 a.m. at the Convention Lower Hall 2

(Research Session 13: MapReduce II)

Invisible Index Creation Times:

up to 7.3 times faster than Hadoop++

Fast Data Upload:

up to 1.6 times faster than Hadoop

Fast Job Runtimes:

up to ~70 times faster than Hadoop and Hadoop++

MapReduce Intro

Data Layouts

Job Optimization

Indexing

Efficient Big Data Processing in Hadoop MapReduce

Jens Dittrich

Jorge-Arnulfo Quiané-Ruiz



COMPUTER SCIENCE