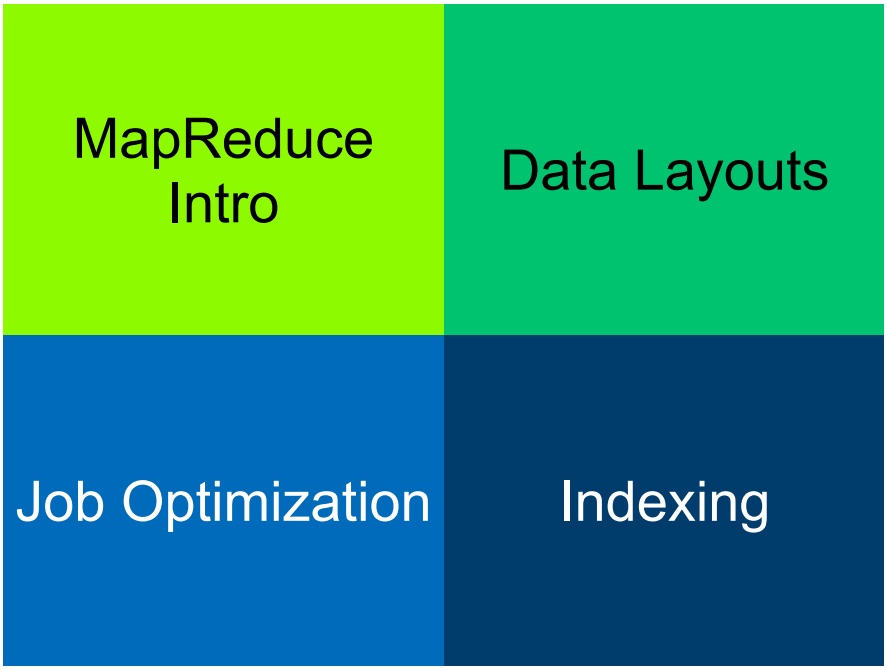
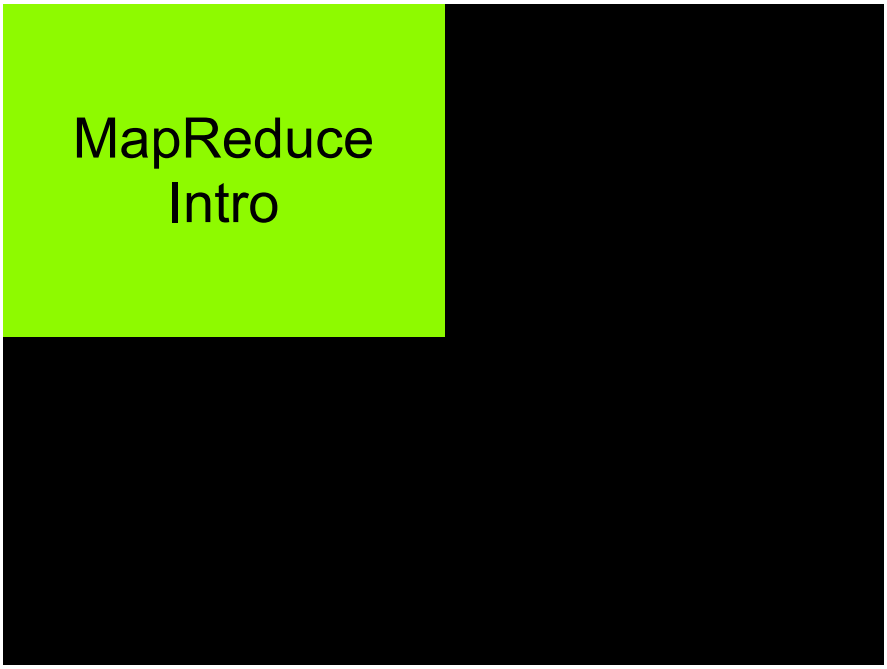
Efficient Big Data Processing in Hadoop MapReduce

Jens Dittrich

Jorge-Arnulfo Quiané-Ruiz

SAARLAND UNIVERSITY
COMPUTER SCIENCE



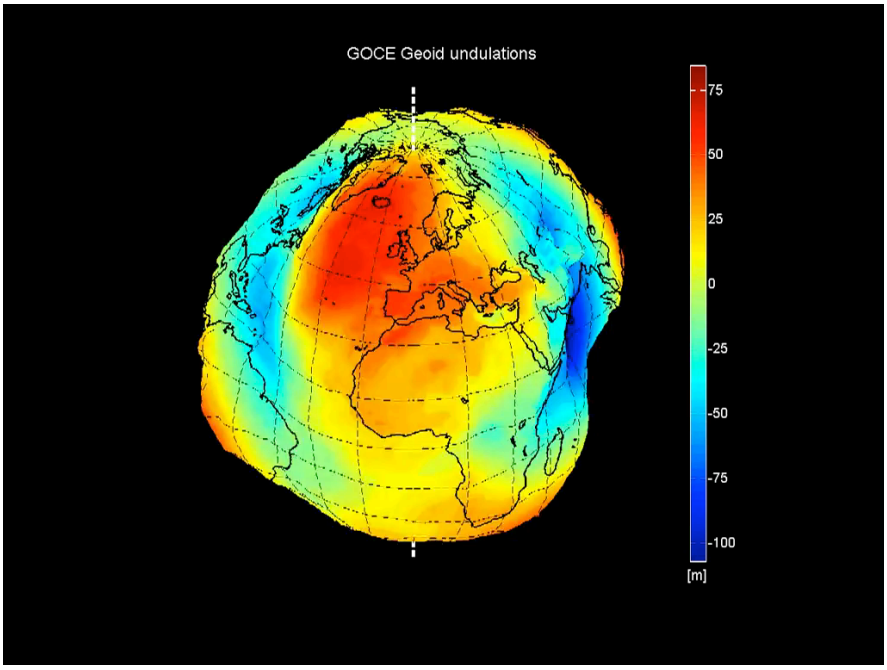


Big Data

http://cdsweb.cern.ch/record/1295244

http://www.flickr.com/photos/14924974@N02/2992963984/

http://it.wikipedia.org/wiki/File:KSC_radio_telescope.jpg



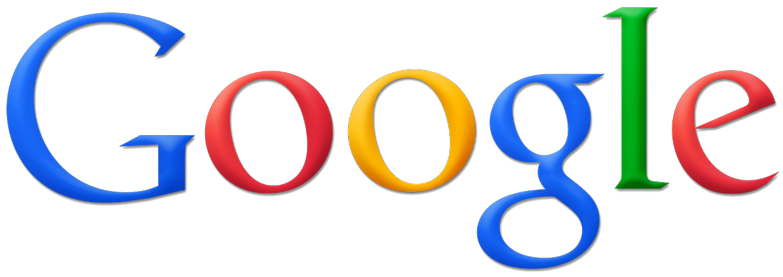
http://www.youtube.com/watch?v=zwRTIUhKfQM

all roads of Germany, from MOVIES-paper SSTD 2009

http://www.istockphoto.com/stock-illustration-16136234-dna-strands.php

YAHOO!®

facebook®



[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

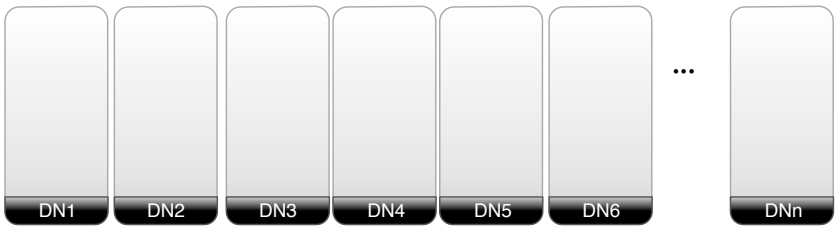
etc.

map() and reduce() with

Big Data ?

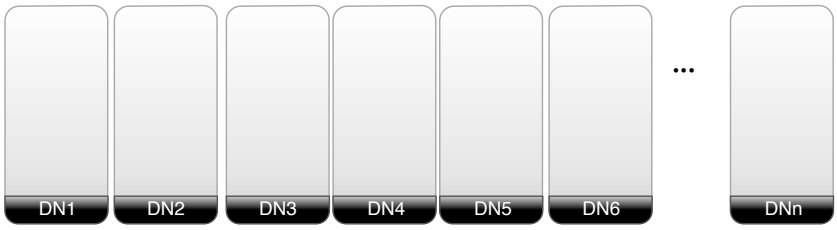
http://www.istockphoto.com/file_closeup.php?id=591134

HDFS



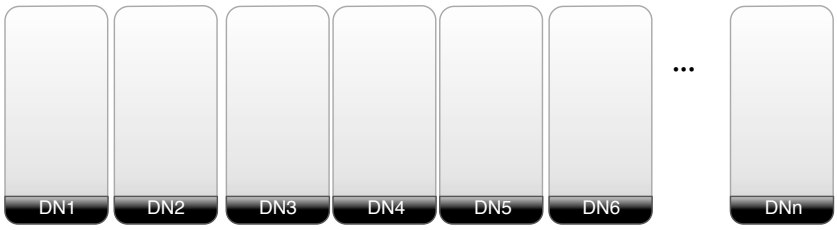
http://www.istockphoto.com/file_closeup.php?id=591134

HDFS



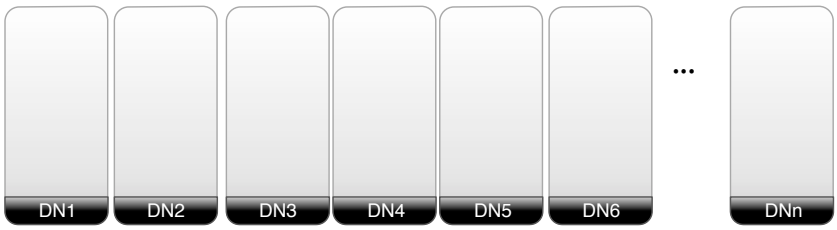
http://www.istockphoto.com/file_closeup.php?id=591134

HDFS



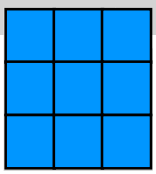
http://www.istockphoto.com/file_closeup.php?id=591134

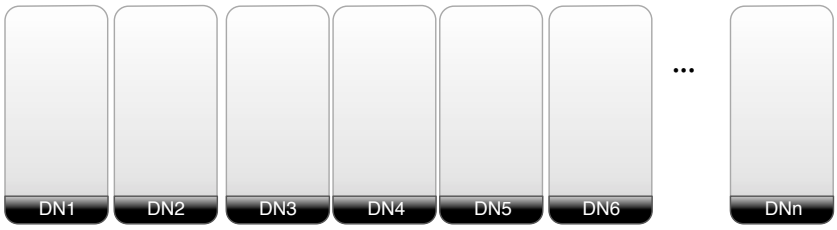
HDFS



HDFS

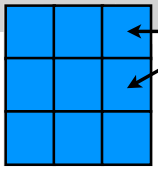
horizontal partitions

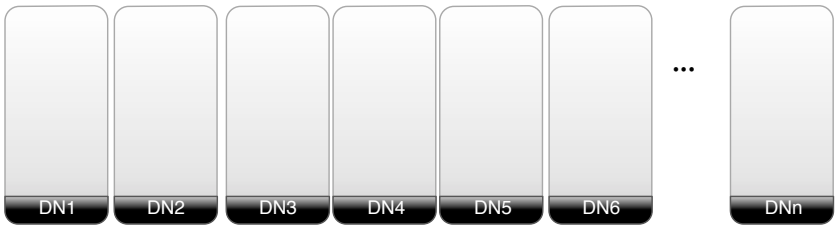




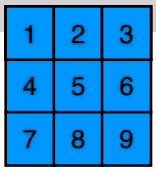
HDFS

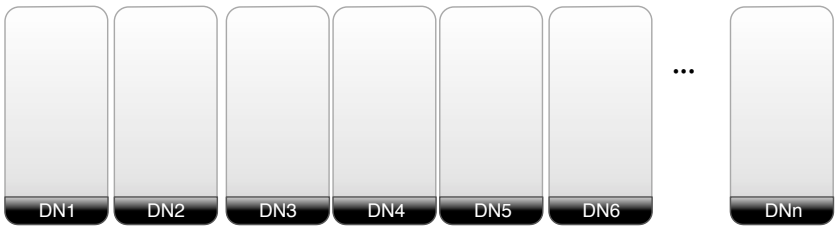
horizontal partitions



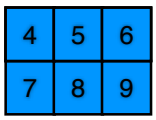


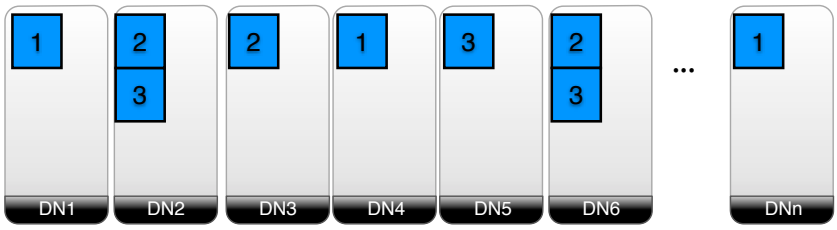
HDFS



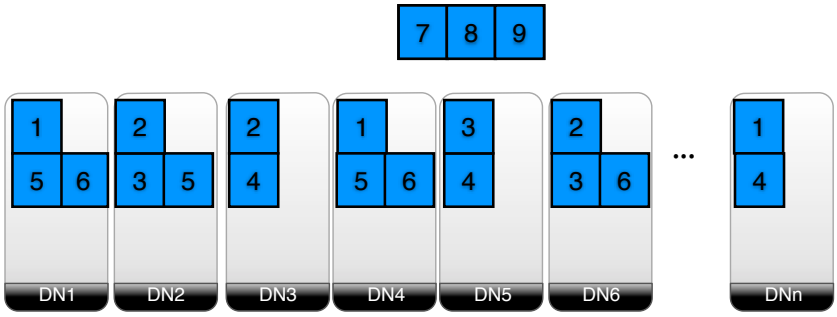


HDFS

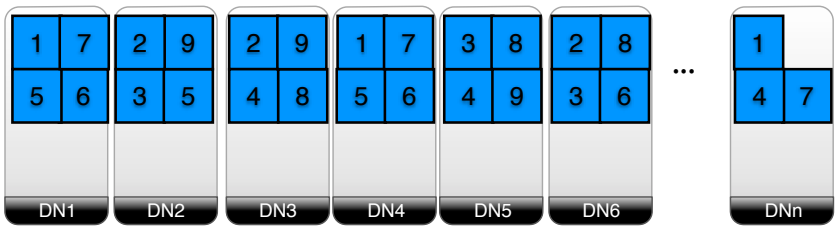




HDFS

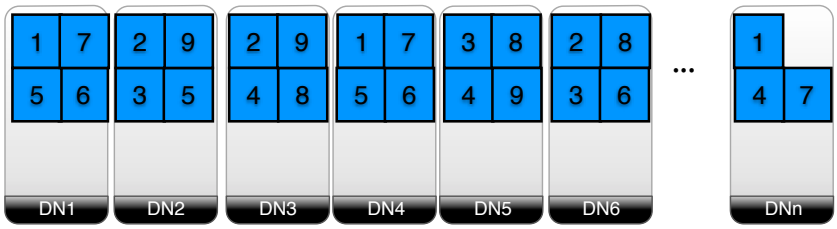


HDFS

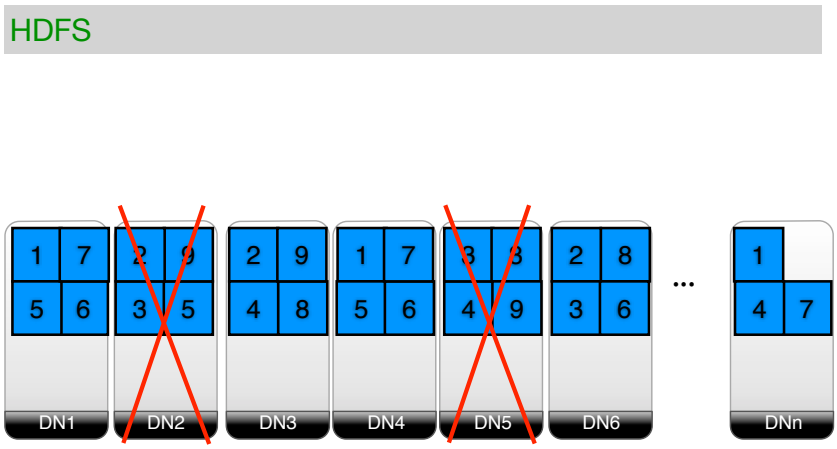


Failover

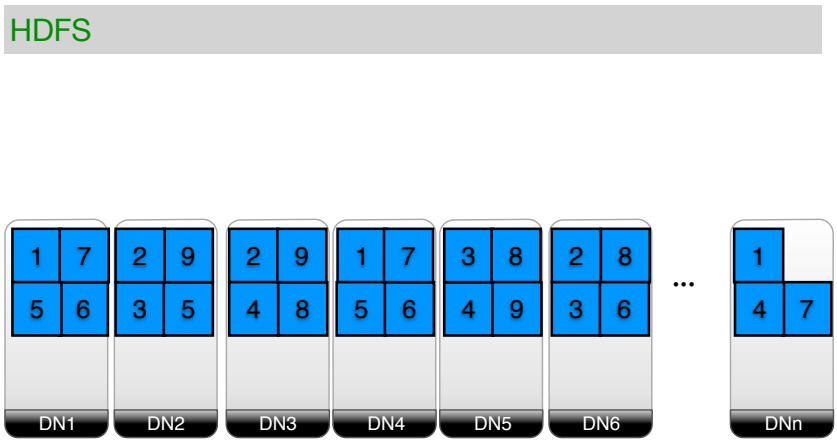
HDFS



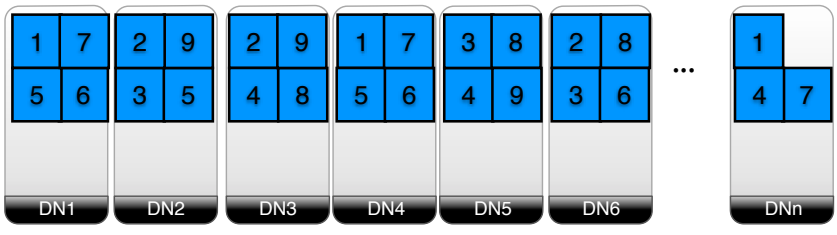
Failover



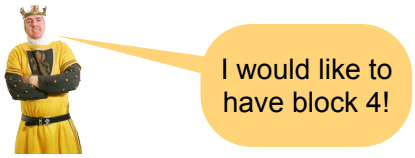
Load Balancing



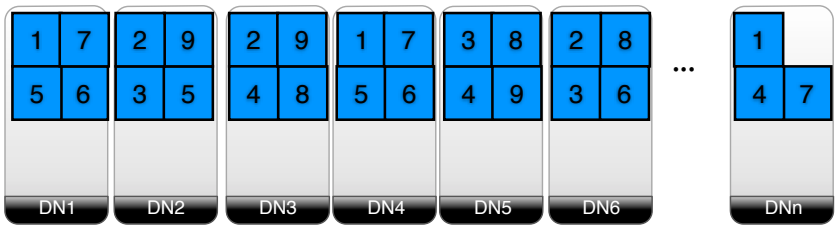
Load Balancing



Load Balancing

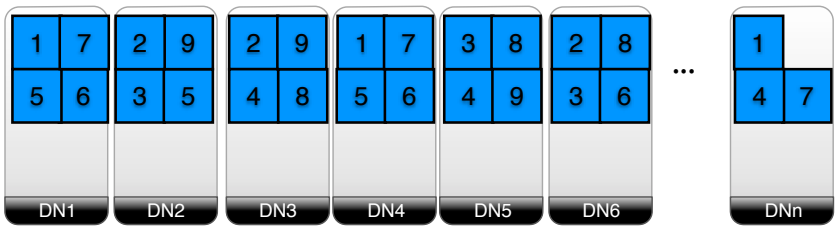


HDFS



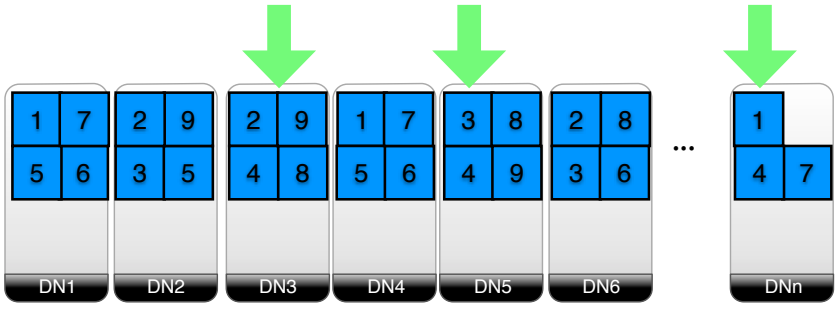
Load Balancing

HDFS

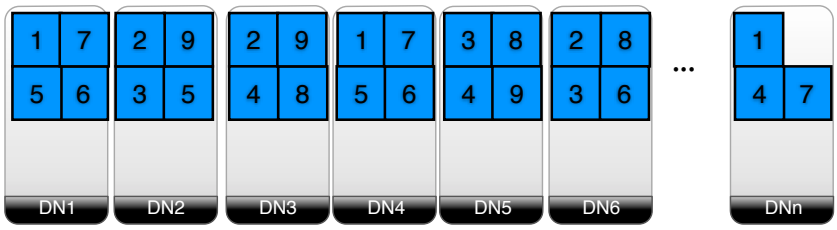


Load Balancing

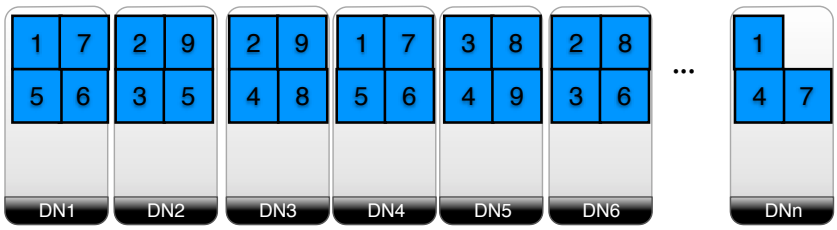
HDFS



HDFS

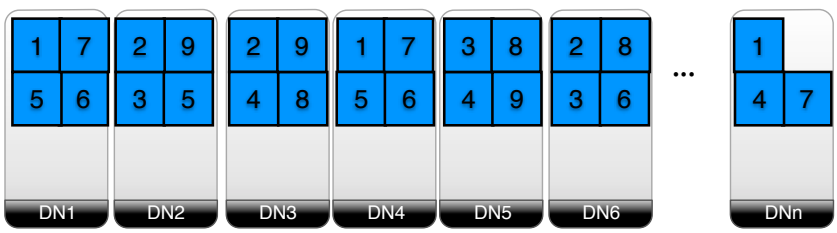


HDFS



MapReduce

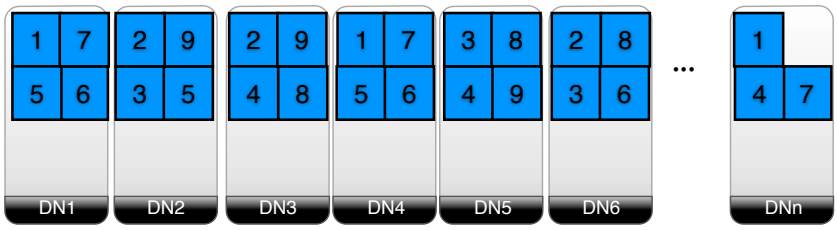
HDFS



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

MapReduce

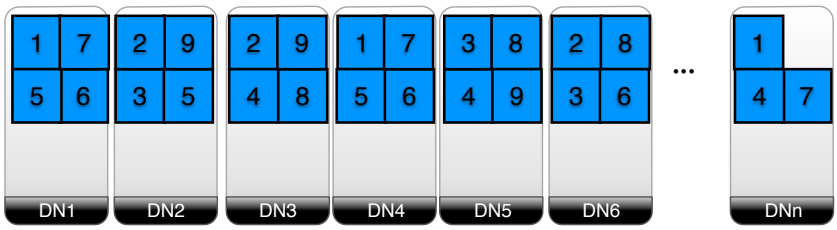
HDFS



Map Phase

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

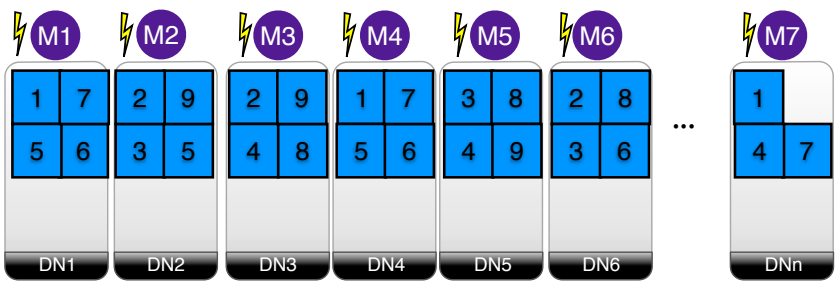
HDFS



Map Phase

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

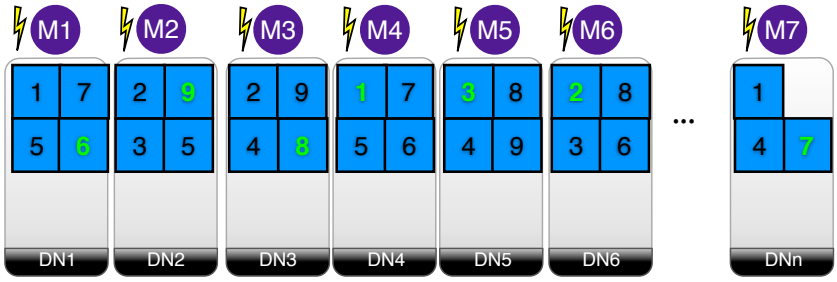
HDFS



Map Phase

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

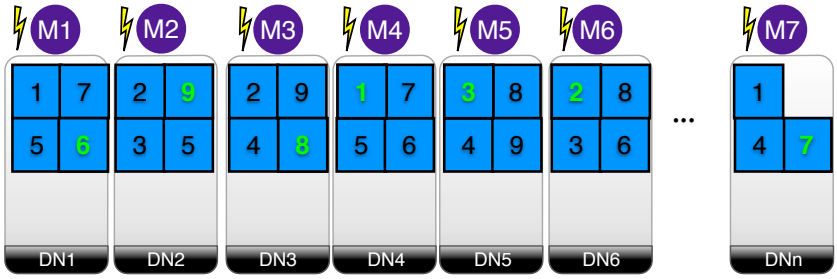
HDFS



Map Phase

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

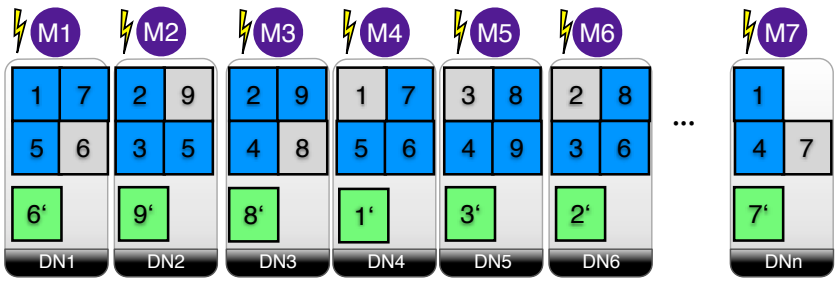
HDFS



Map Phase

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

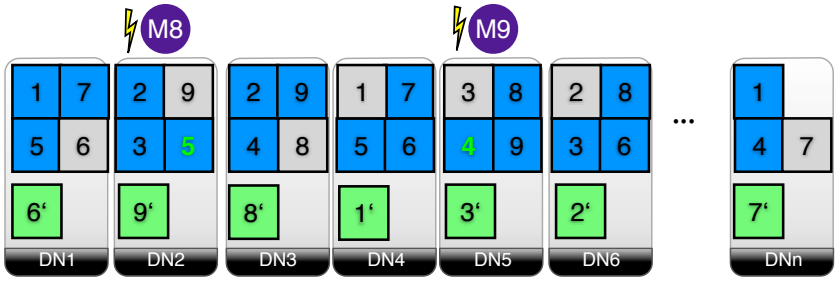
HDFS



Map Phase

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

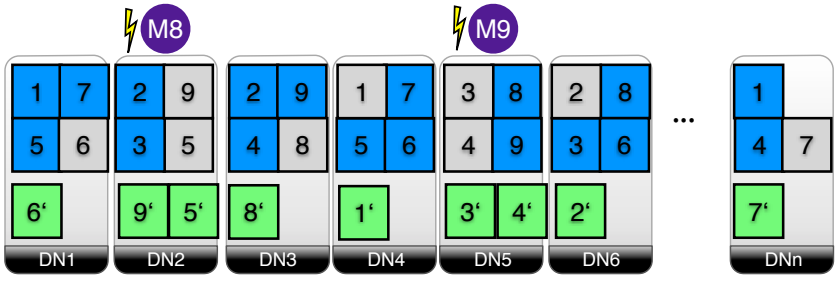
HDFS



Map Phase

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

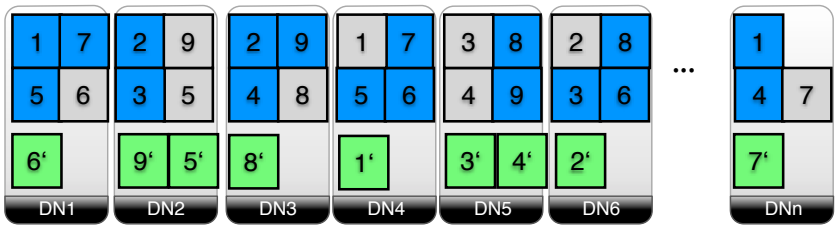
HDFS



Map Phase

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

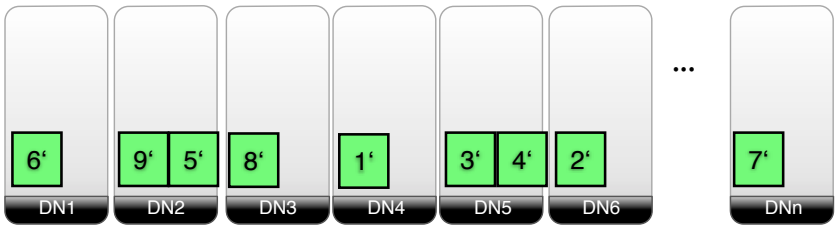
HDFS



Shuffle Phase

MapReduce group by term

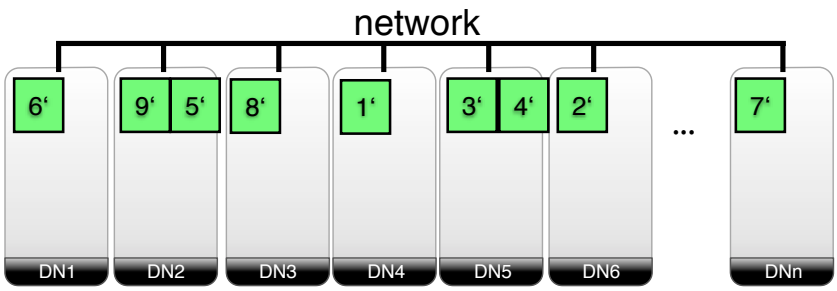
HDFS



Shuffle Phase

MapReduce group by term

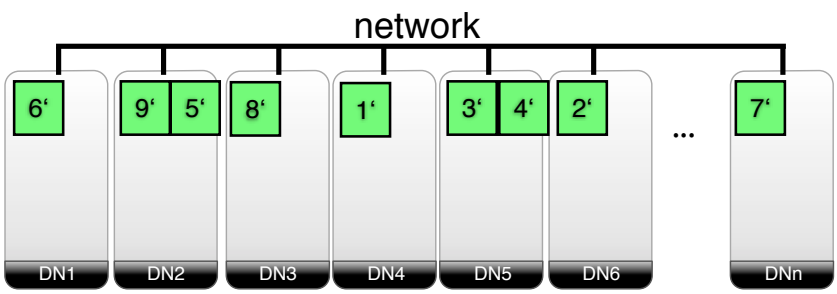
HDFS



Shuffle Phase

MapReduce group by term

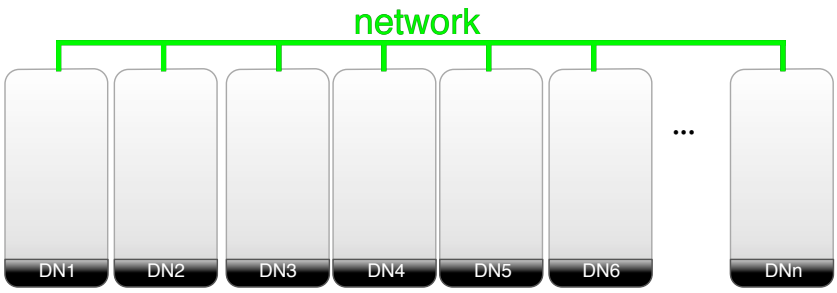
HDFS



Shuffle Phase

MapReduce group by term

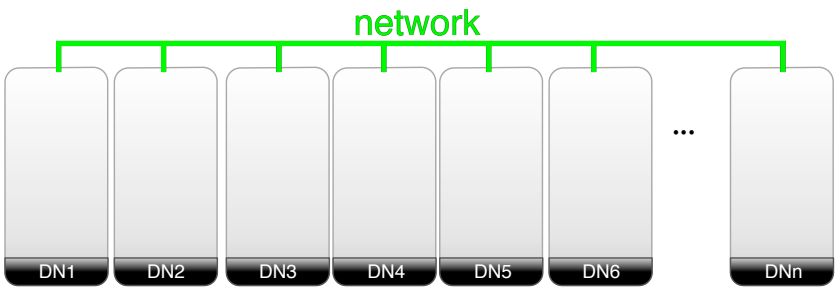
HDFS



Shuffle Phase

MapReduce group by term

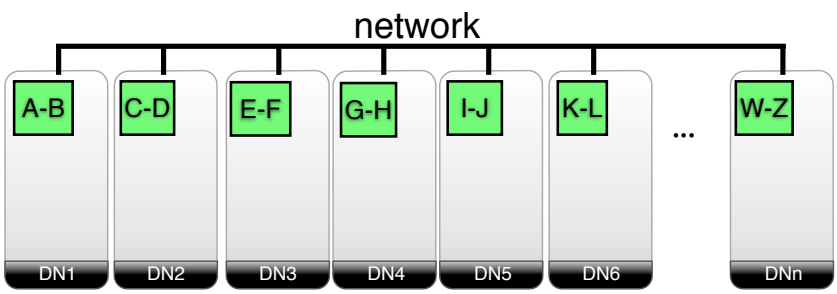
HDFS



Shuffle Phase

MapReduce group by term

HDFS

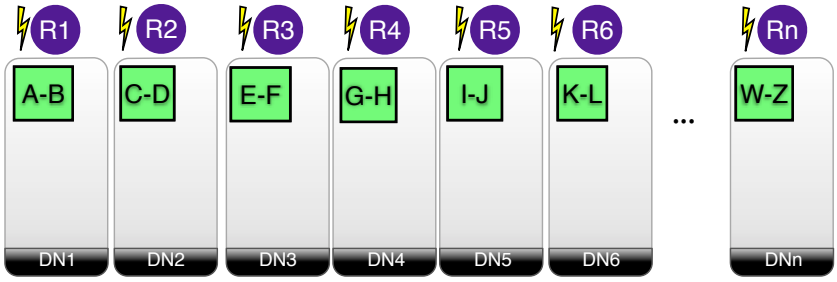


Reduce Phase

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

MapReduce

HDFS

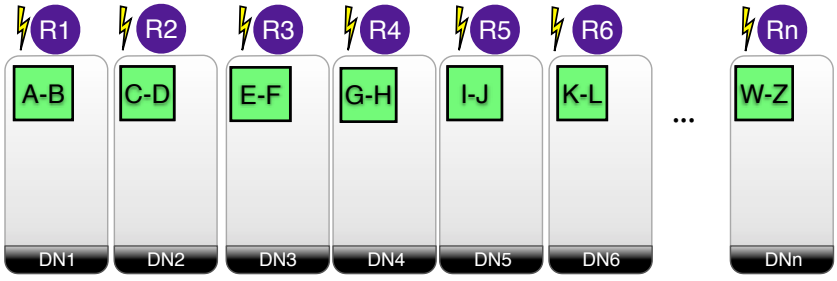


Reduce Phase

MapReduce

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

HDFS

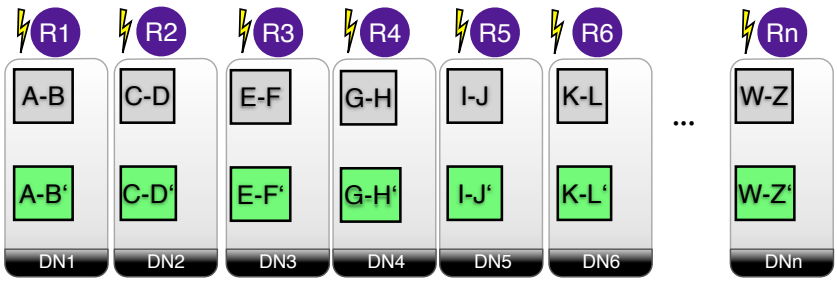


Reduce Phase

MapReduce

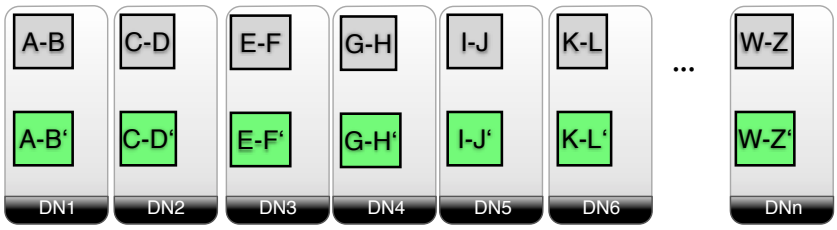
reduce(term, set of docID) -> set of
(term, (posting list of docID, count))

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

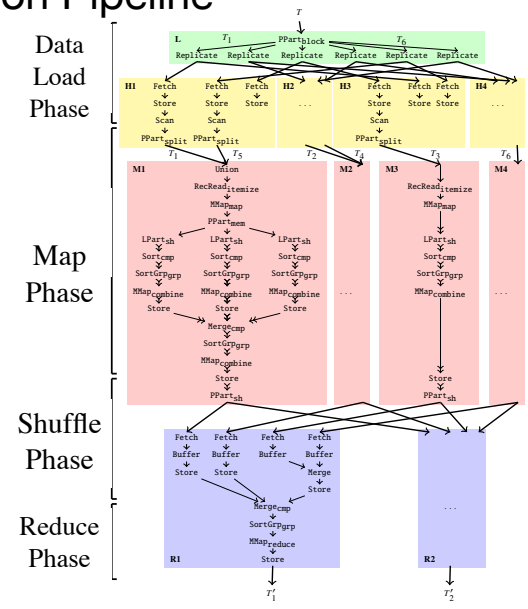
Scalability

schema-later

Hadoop
MapReduce
Disadvantages

Performance

Execution Pipeline



details: see Hadoop++-paper

MapReduce Intro

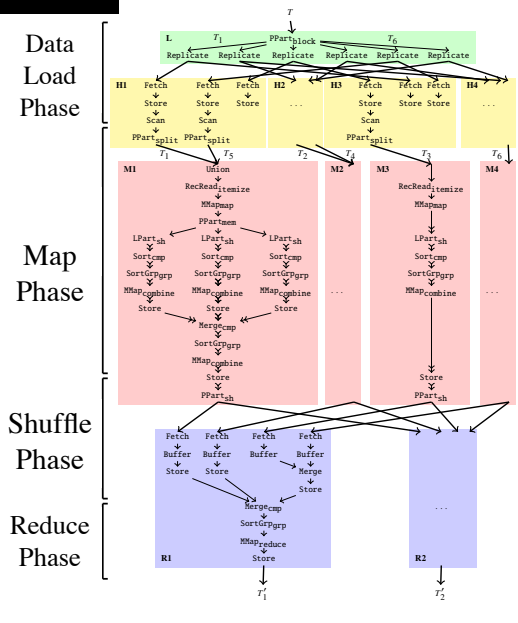
Data Layouts

Job Optimization

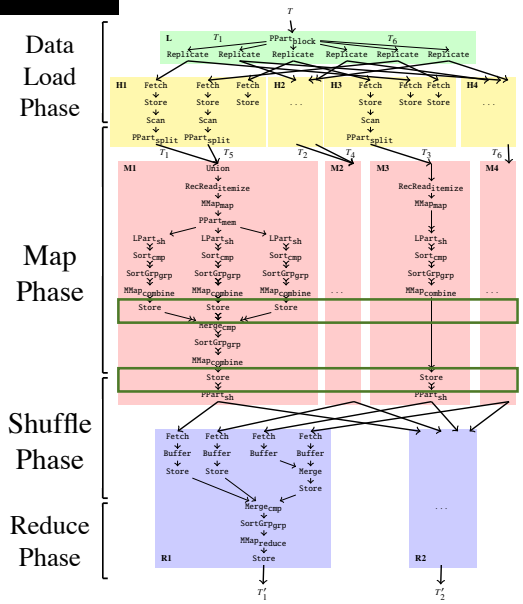
Indexing

Job Optimization

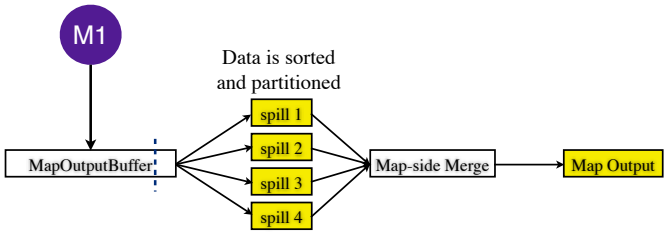
Spill Process



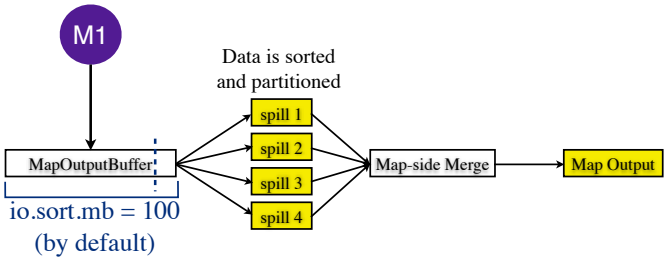
Spill Process



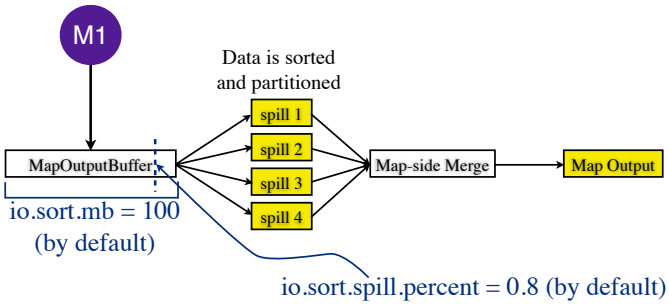
Spill Process Overview



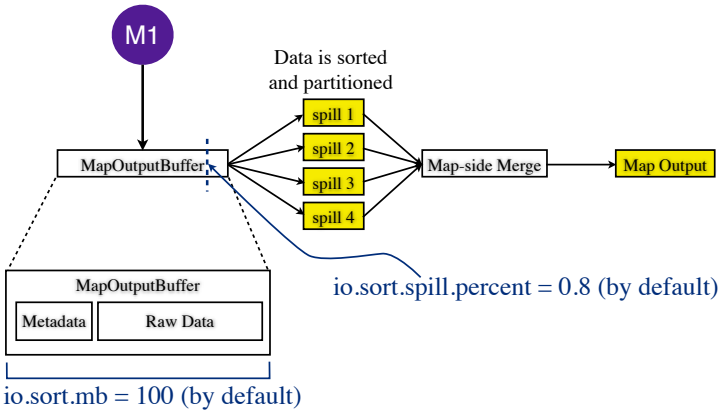
Spill Process Overview



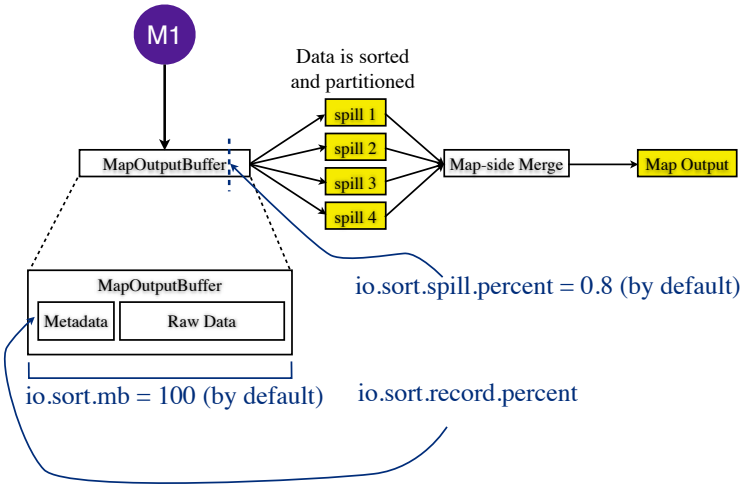
Spill Process Overview



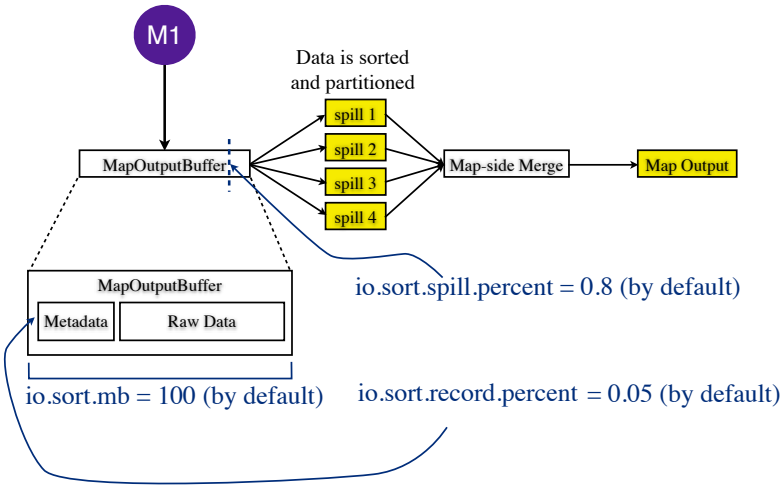
Spill Process Overview



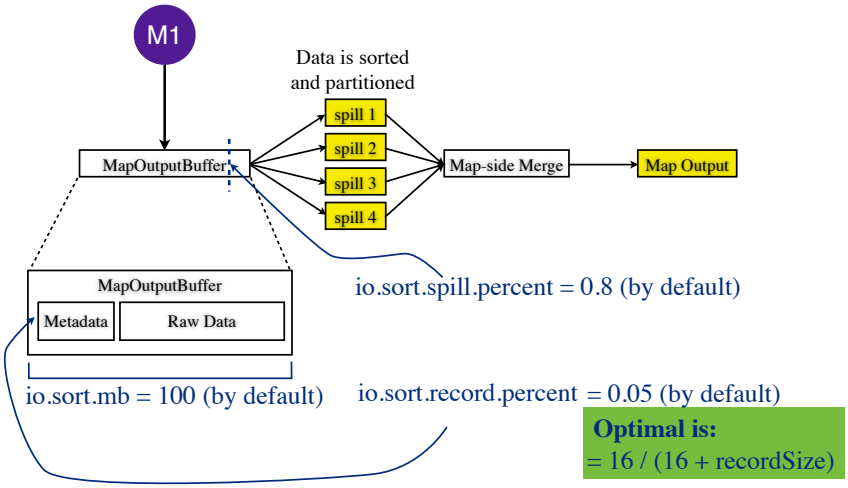
Spill Process Overview



Spill Process Overview

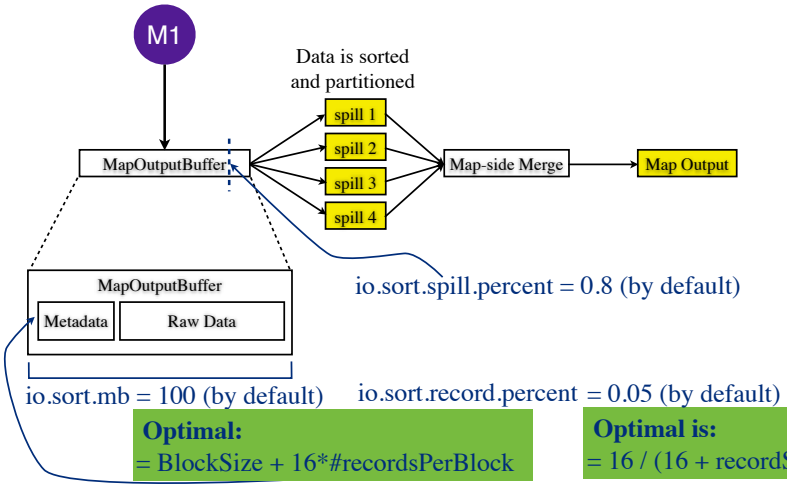


Spill Process Overview



Optimal is:
= 16 / (16 + recordSize)

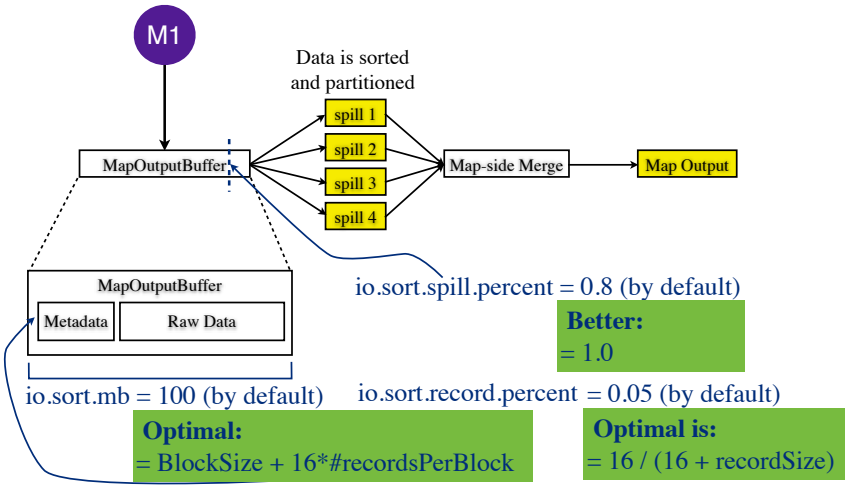
Spill Process Overview



Optimal:
= BlockSize + 16*#recordsPerBlock

Optimal is:
= 16 / (16 + recordSize)

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.

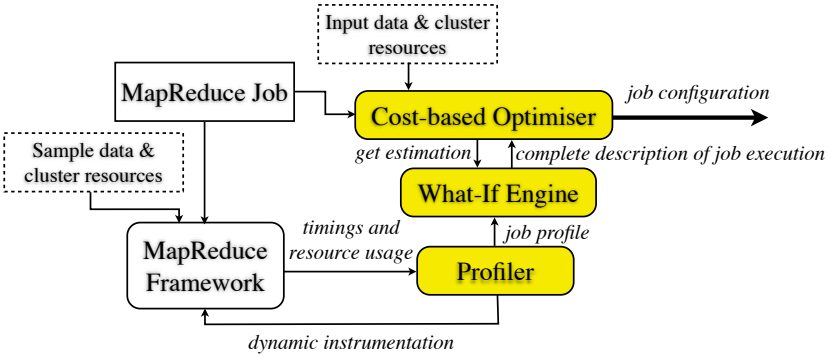
[H. Herodotou and S. Babu: Profiling, What-If, and Cost-based Optimization of MapReduce Programs. PVLDB 2011.] 68

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.



[H. Herodotou and S. Babu: Profiling, What-If, and Cost-based Optimization of MapReduce Programs. PVLDB 2011.]

Automatic Job Optimization

Manimal

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

[E. Jahani et al.: Automatic Optimization for MapReduce Programs. PVLDB 2011.]

Automatic Job Optimization

Manimal

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

Contribution: static code analysis of MapReduce jobs.

[E. Jahani et al.: Automatic Optimization for MapReduce Programs. PVLDB 2011.]

69

Automatic Job Optimization

Manimal

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

Contribution: static code analysis of MapReduce jobs.

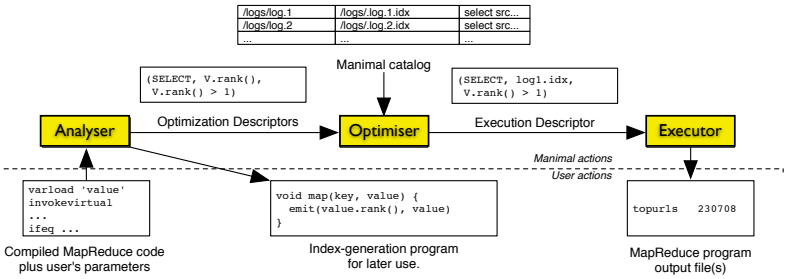
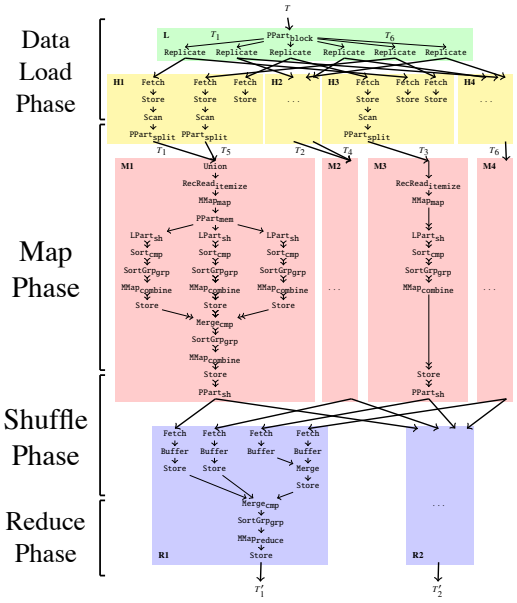
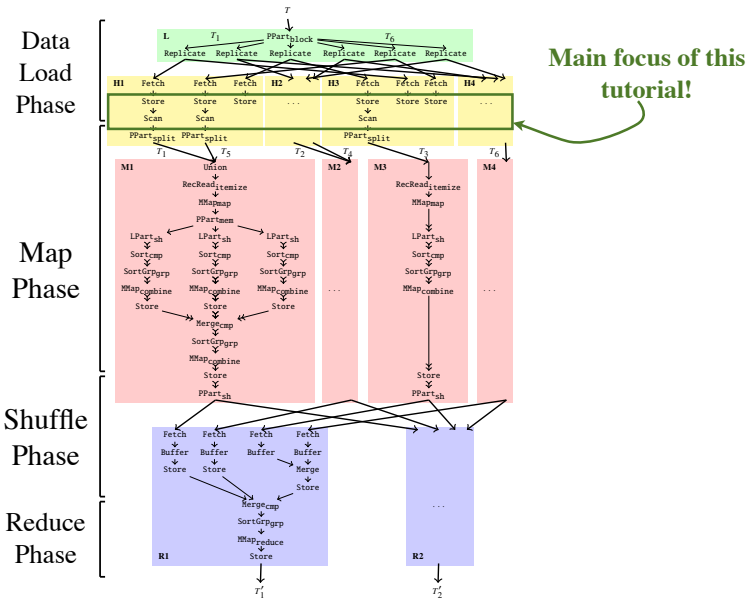


Figure 1: Architecture of the MANIMAL system.

[E. Jahani et al.: Automatic Optimization for MapReduce Programs. PVLDB 2011.] 69





MapReduce Intro

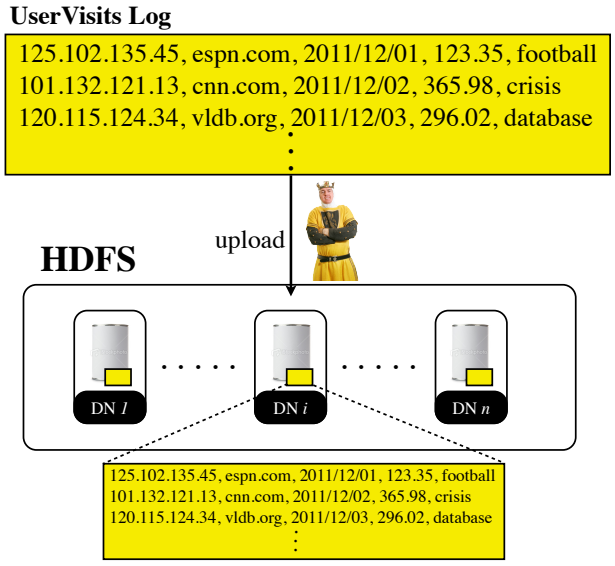
Data Layouts

Job Optimization

Indexing

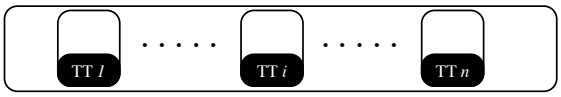
Data Layouts

Default Layout

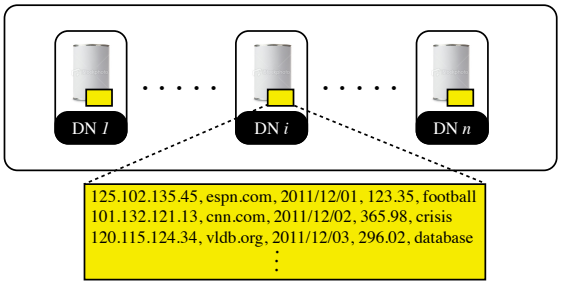


Problem

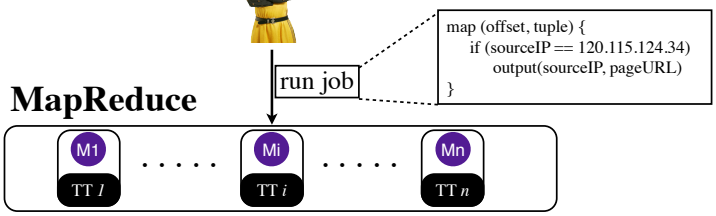
MapReduce



HDFS

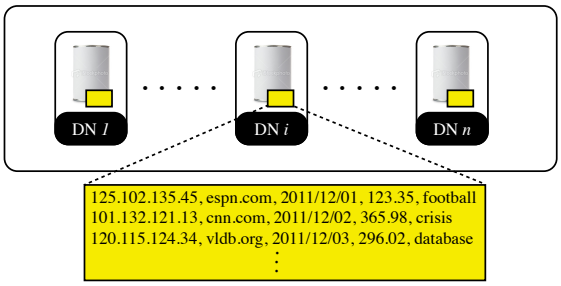


Problem

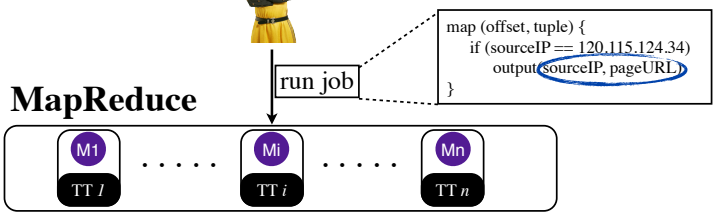


MapReduce

HDFS

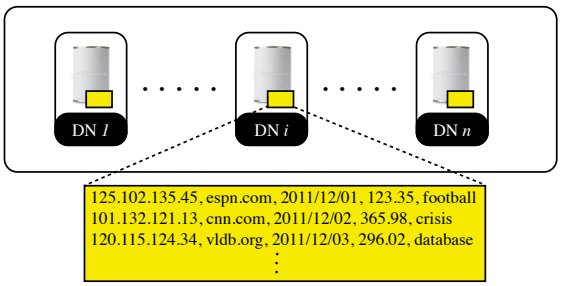


Problem

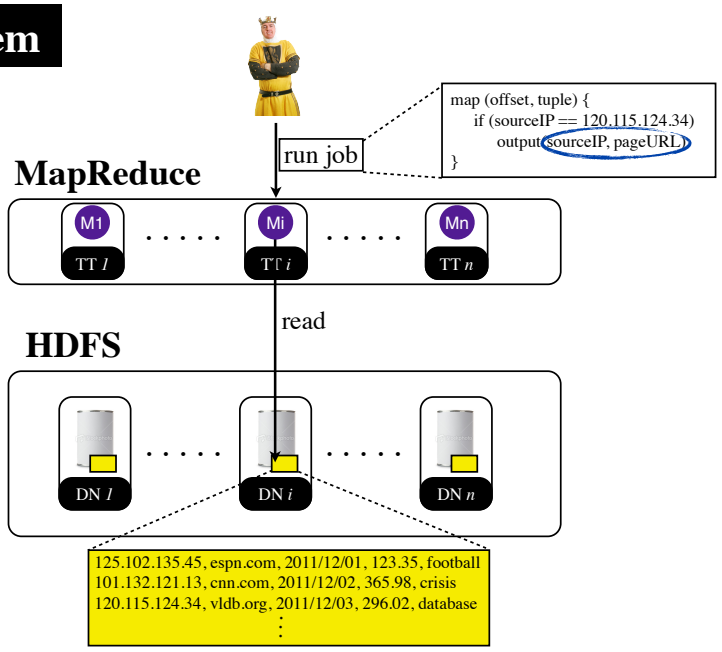


MapReduce

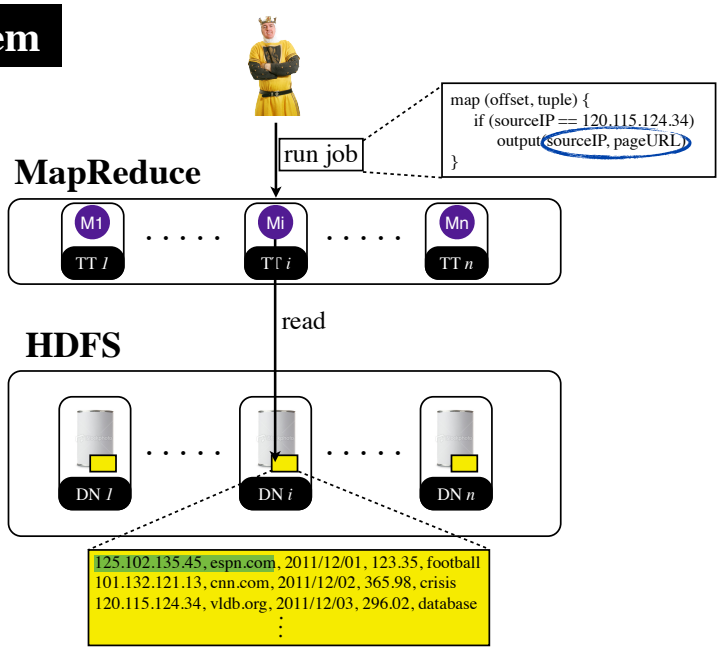
HDFS



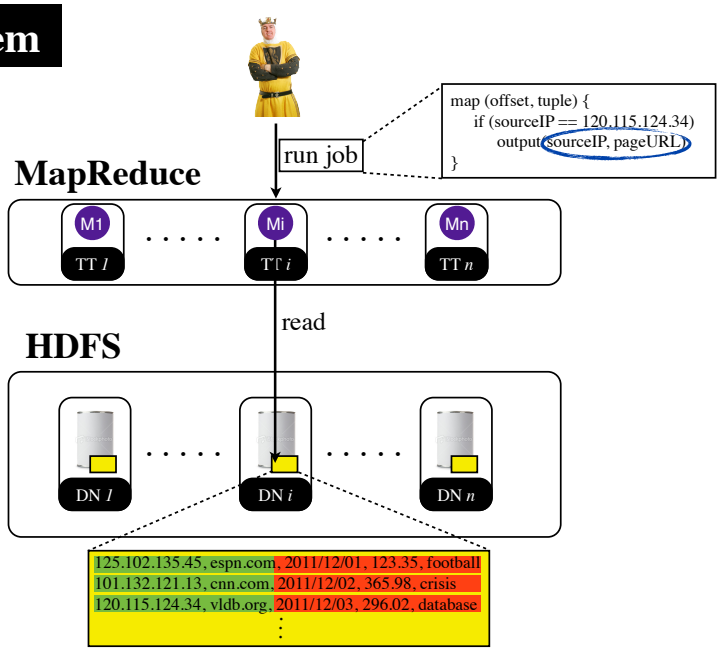
Problem



Problem



Problem



Data Layouts in MapReduce

Data Layouts in MapReduce

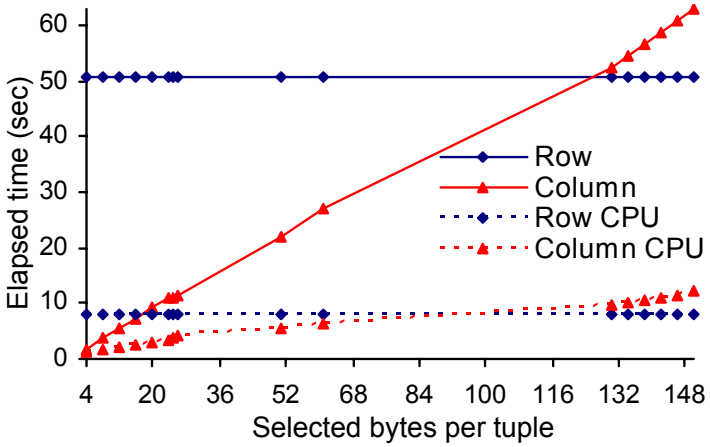
Initial
Row
Read Unnecessary columns

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

[Harizopoulos et al.: Performance Tradeoffs in Read-Optimized Databases. VLDB 2006]

Column Layout in MapReduce?

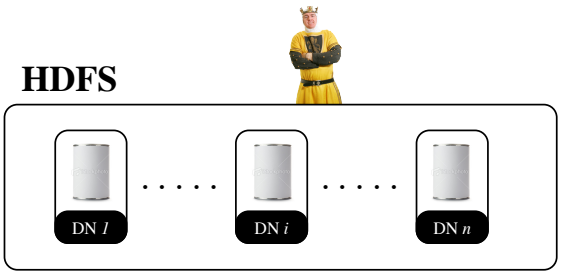
Column-wise File (CFile)

[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

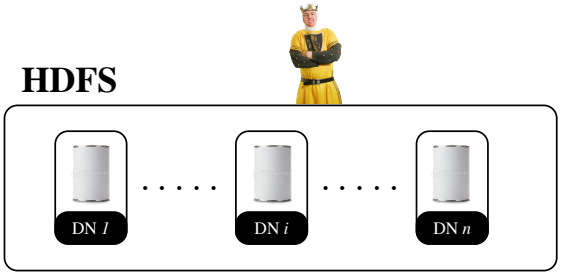


[Y. Lin et al.: Llama: Leveraging Columnar Storage for Scalable Join Processing in the MapReduce Framework. SIGMOD 2011.]

Data Upload

UserVisits Log

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



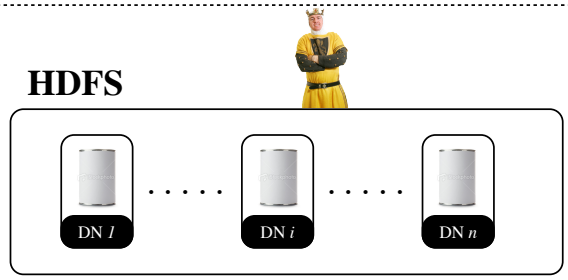
[Y. Lin et al.: Llama: Leveraging Columnar Storage for Scalable Join Processing in the MapReduce Framework. SIGMOD 2011.]

Data Upload

UserVisits Log

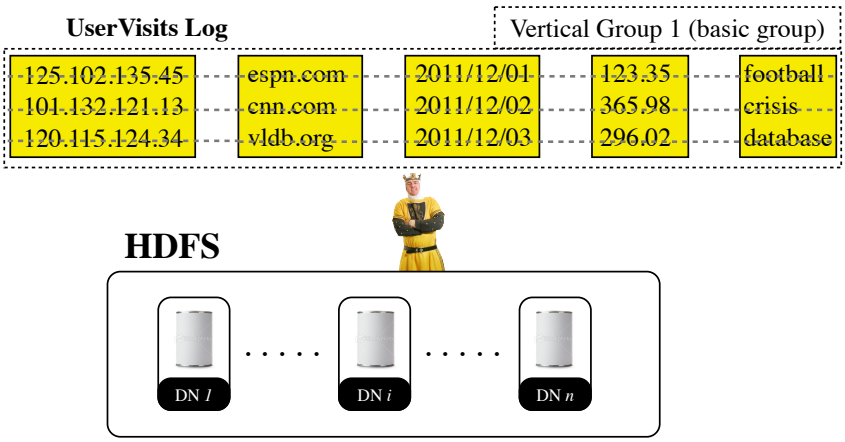
Vertical Group 1 (basic group)

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



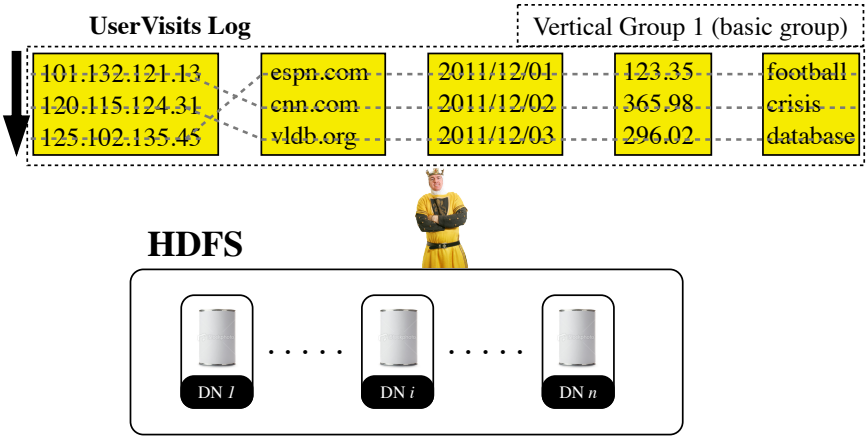
[Y. Lin et al.: Llama: Leveraging Columnar Storage for Scalable Join Processing in the MapReduce Framework. SIGMOD 2011.]

Data Upload



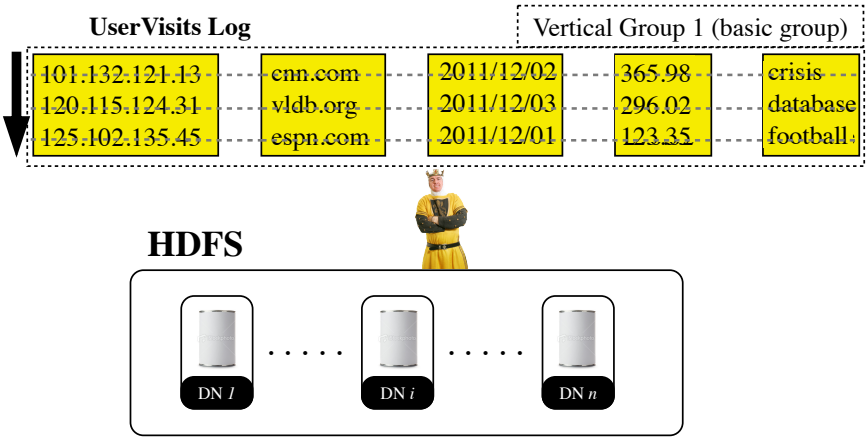
[Y. Lin et al.: Llama: Leveraging Columnar Storage for Scalable Join Processing in the MapReduce Framework. SIGMOD 2011.]

Data Upload



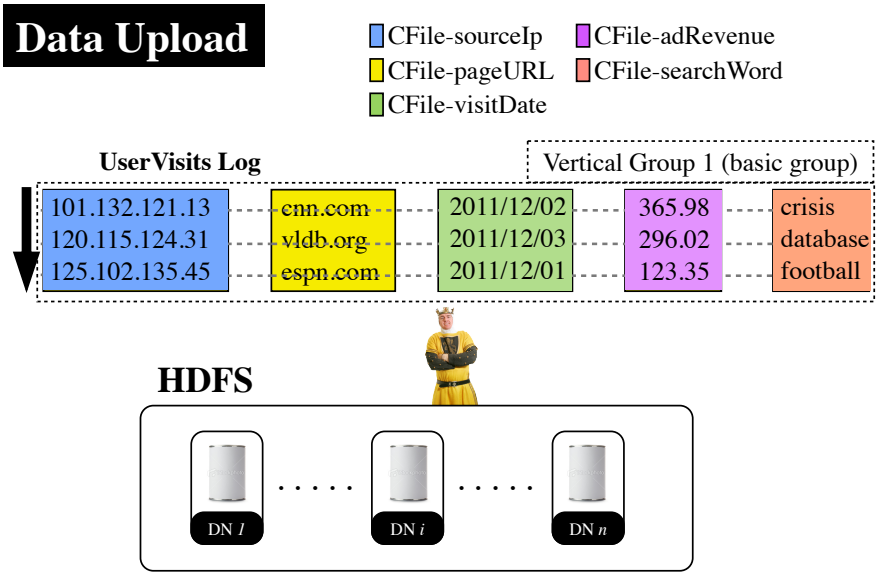
[Y. Lin et al.: Llama: Leveraging Columnar Storage for Scalable Join Processing in the MapReduce Framework. SIGMOD 2011.]

Data Upload



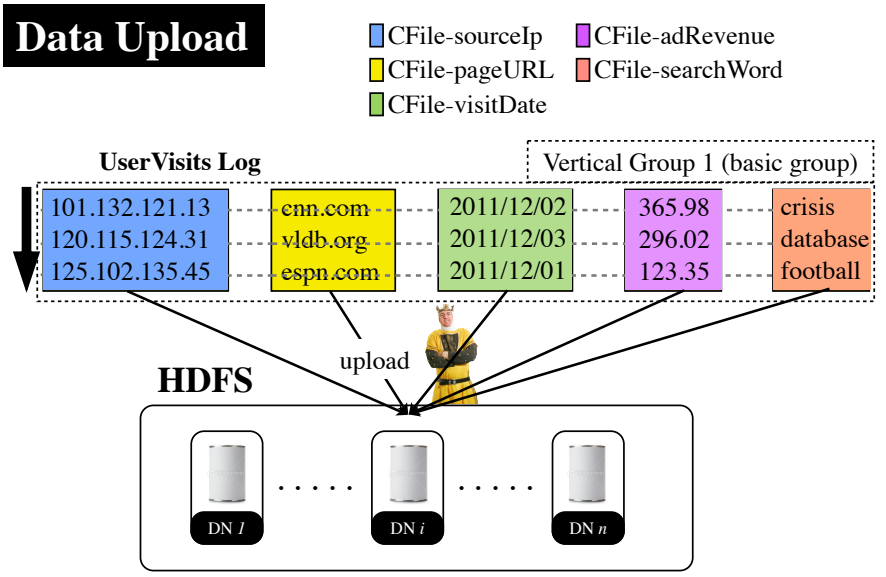
[Y. Lin et al.: Llama: Leveraging Columnar Storage for Scalable Join Processing in the MapReduce Framework. SIGMOD 2011.]

Data Upload



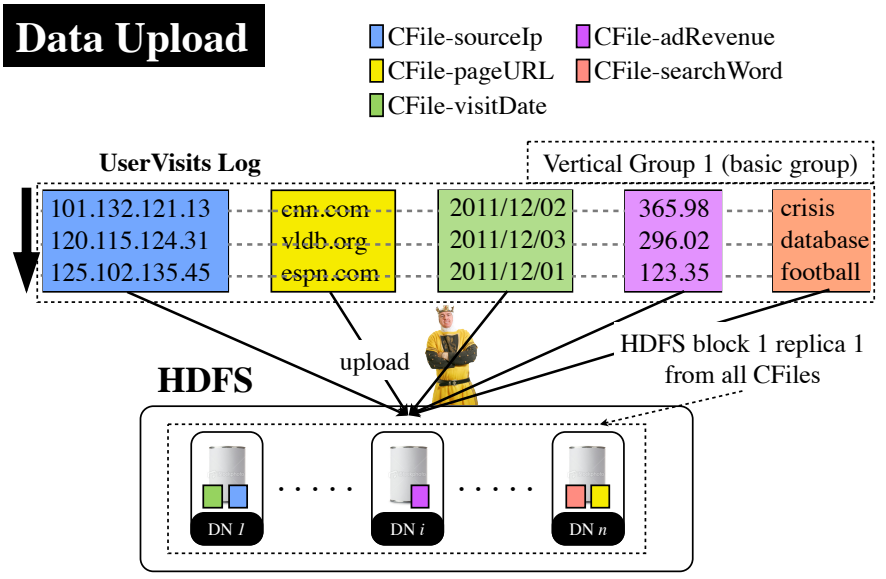
[Y. Lin et al.: Llama: Leveraging Columnar Storage for Scalable Join Processing in the MapReduce Framework. SIGMOD 2011.]

Data Upload



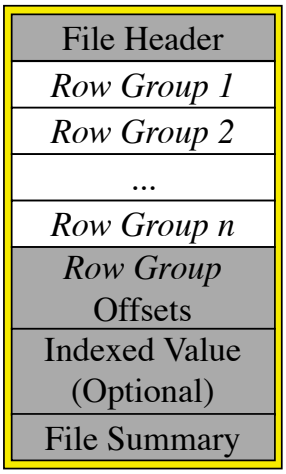
[Y. Lin et al.: Llama: Leveraging Columnar Storage for Scalable Join Processing in the MapReduce Framework. SIGMOD 2011.]

Data Upload



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

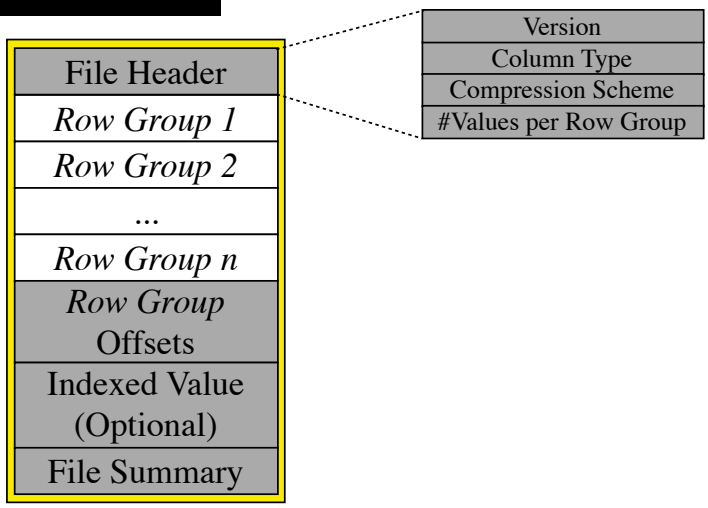
CFile Format



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

Compress picture: http://openclipart.org/detail/68671/compress-by-buggi

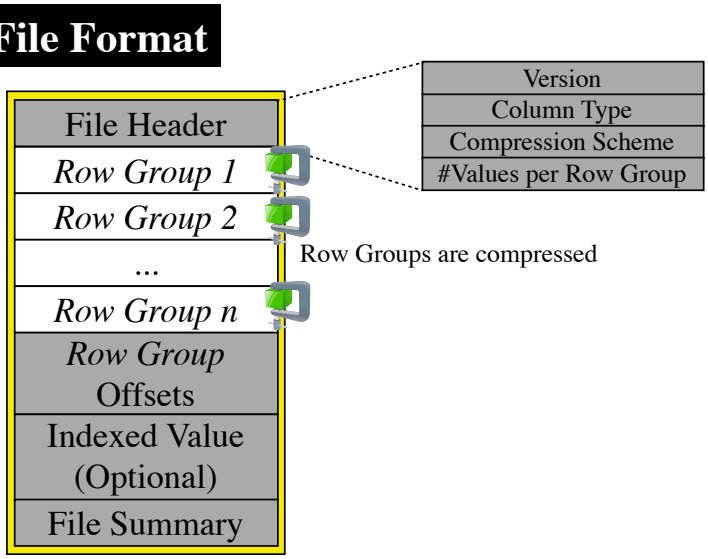
CFile Format



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

Compress picture: http://openclipart.org/detail/68671/compress-by-buggi

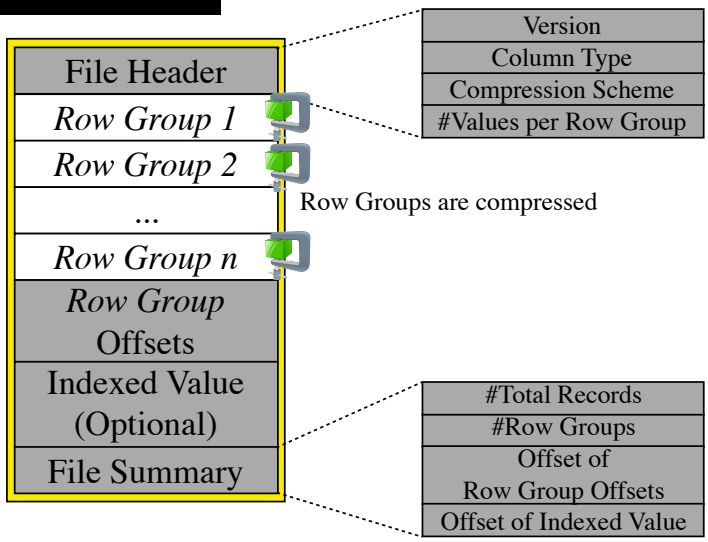
CFile Format



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

Compress picture: http://openclipart.org/detail/68671/compress-by-buggi

CFile Format



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

Compress picture: http://openclipart.org/detail/68671/compress-by-buggi

HDFS Blocks for CFile-adRevenue

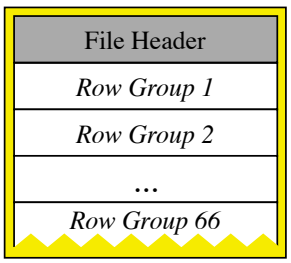
Column Type: float (4 bytes)
#Total Values = 130,000
Row Group = 1,000 values
HDFS Block Size = 64MB

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

HDFS Blocks for CFile-adRevenue

Column Type: float (4 bytes)
#Total Values = 130,000
Row Group = 1,000 values
HDFS Block Size = 64MB

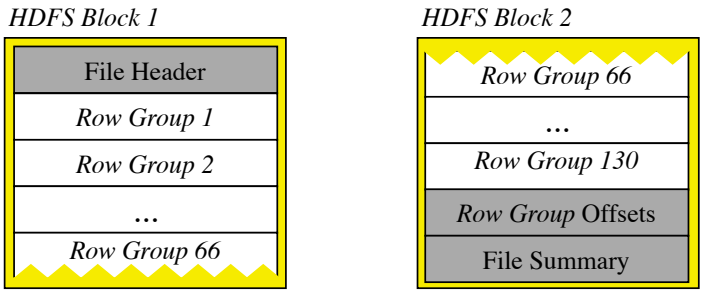
HDFS Block 1



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

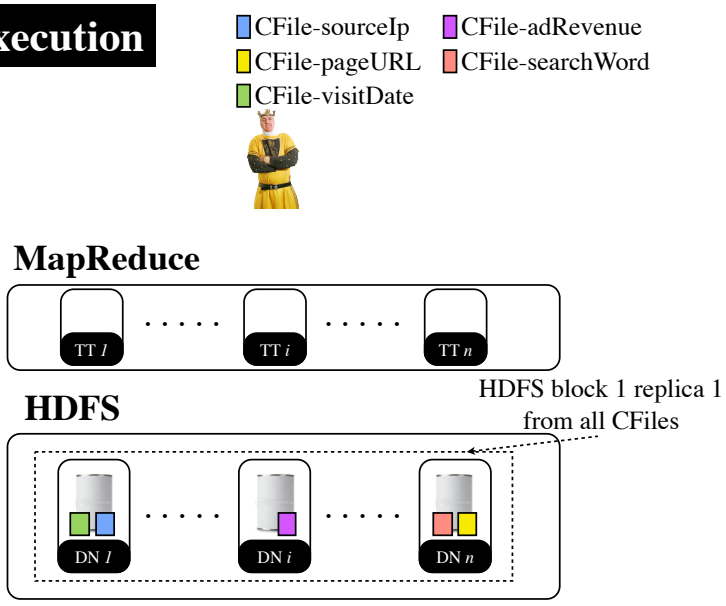
HDFS Blocks for CFile-adRevenue

Column Type: float (4 bytes)
#Total Values = 130,000
Row Group = 1,000 values
HDFS Block Size = 64MB



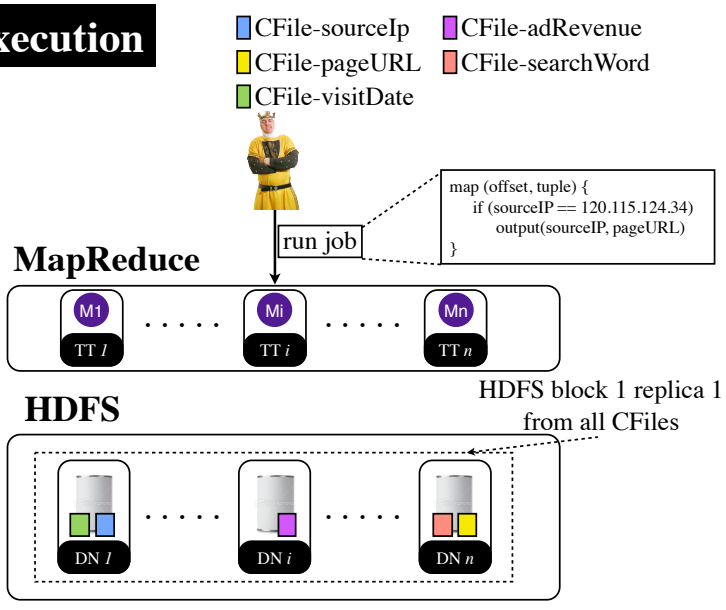
[Y. Lin et al.: Llama: Leveraging Columnar Storage for Scalable Join Processing in the MapReduce Framework. SIGMOD 2011.]

Job Execution



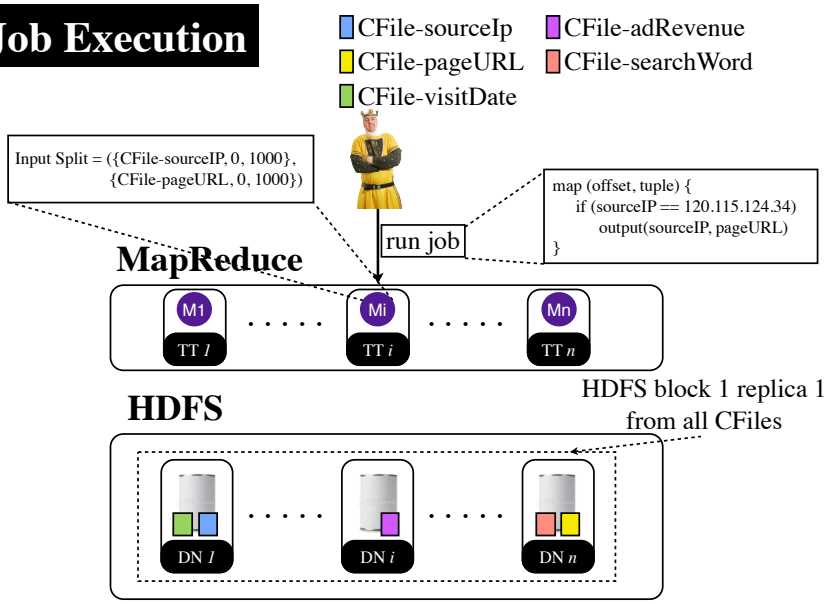
[Y. Lin et al.: Llama: Leveraging Columnar Storage for Scalable Join Processing in the MapReduce Framework. SIGMOD 2011.]

Job Execution

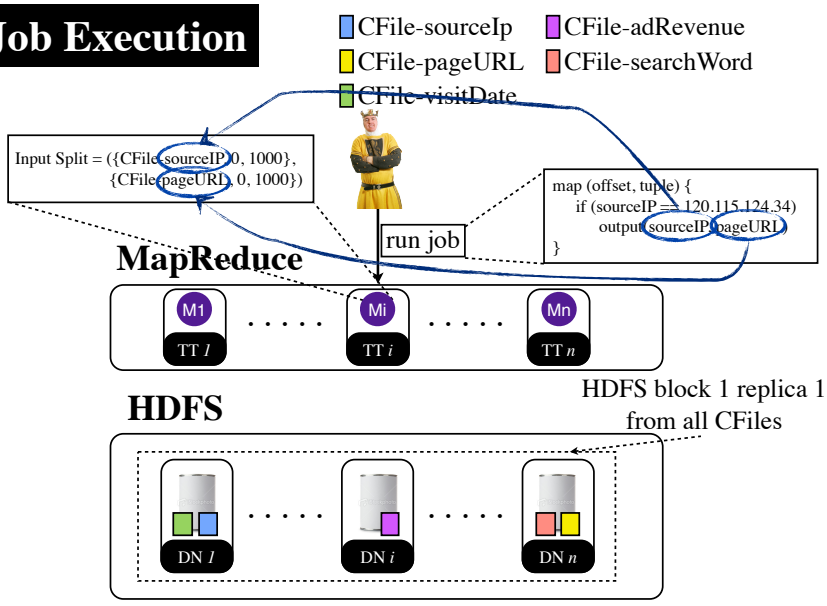


[Y. Lin et al.: Llama: Leveraging Columnar Storage for Scalable Join Processing in the MapReduce Framework. SIGMOD 2011.]

Job Execution

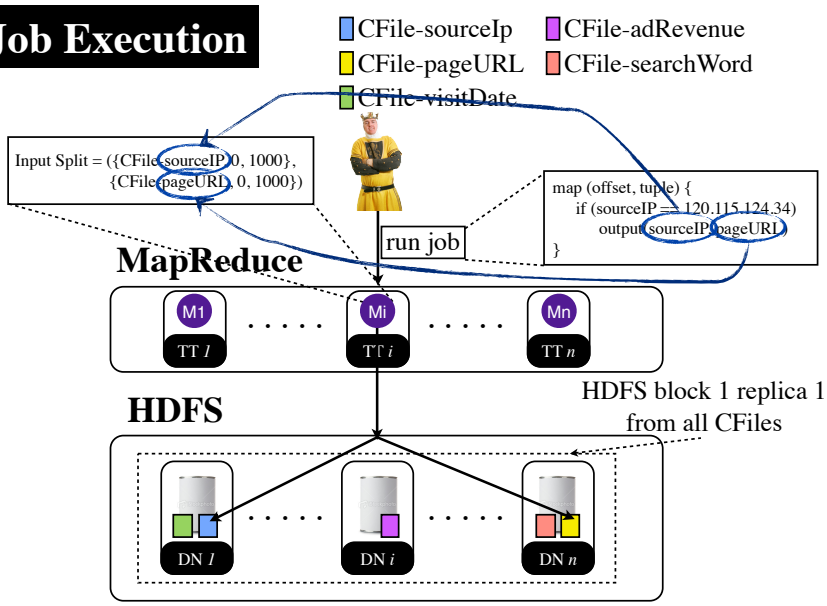


Job Execution



[Y. Lin et al.: Llama: Leveraging Columnar Storage for Scalable Join Processing in the MapReduce Framework. SIGMOD 2011.] 83

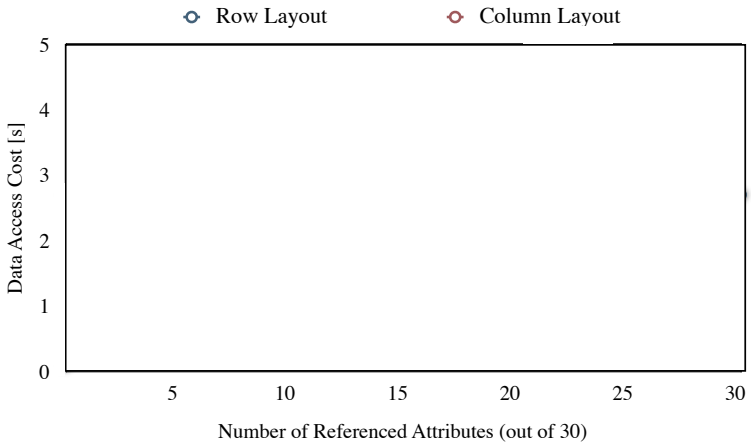
Job Execution



[Y. Lin et al.: Llama: Leveraging Columnar Storage for Scalable Join Processing in the MapReduce Framework. SIGMOD 2011.] 83

Column Layout in MapReduce

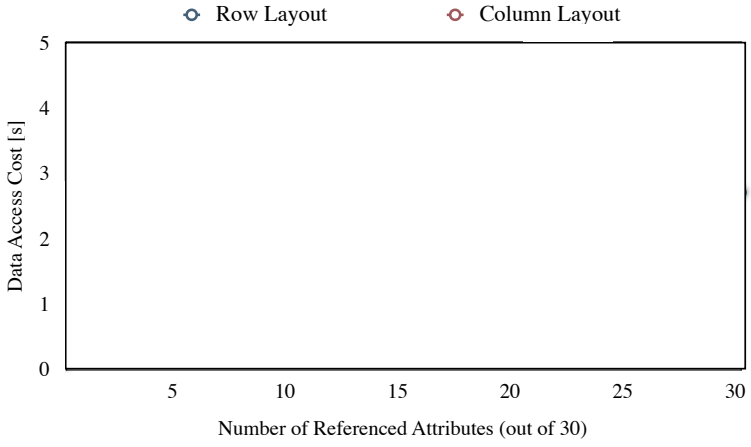
SELECT *a1, a2, ...*
FROM table30Atts



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

Column Layout in MapReduce

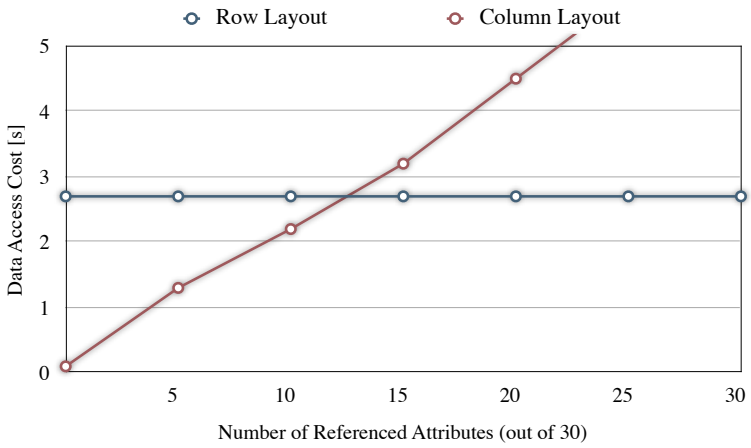
SELECT *a1, a2, ...* ⟵ We vary the number of attributes
FROM table30Atts



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

Column Layout in MapReduce

SELECT *a1, a2, ...* ⟵ We vary the number of attributes
FROM table30Atts



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

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]

Recap

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

[A. Ailamaki et al.: Weaving Relations for Cache Performance. VLDB 2001] 87

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)

[A. Ailamaki et al.: Weaving Relations for Cache Performance. VLDB 2001] 87

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)

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

PAX in MapReduce?

Storage in Cheetah

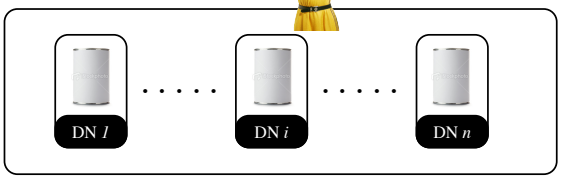
[S. Chen: A High Performance, Custom Data Warehouse on Top of MapReduce. PVLDB 2010]

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

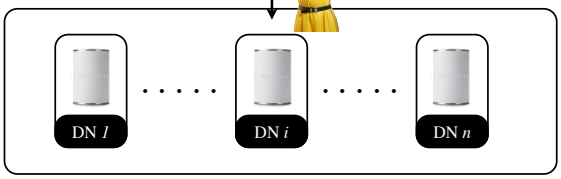


Data Upload

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



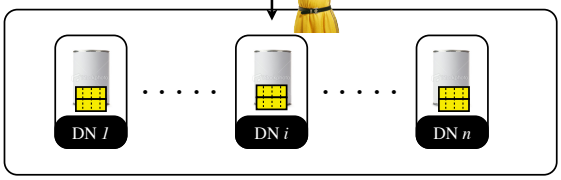
[S. Chen: A High Performance, Custom Data Warehouse on Top of MapReduce. PVLDB 2010] 90

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



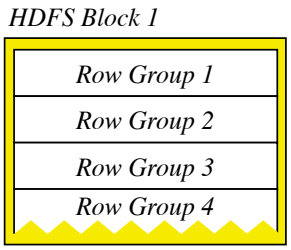
[S. Chen: A High Performance, Custom Data Warehouse on Top of MapReduce. PVLDB 2010] 90

HDFS Block Format

Average Record Size: 100 bytes
#Total Records = 1,000,000
Row Group Size = 200,000 records
HDFS Block Size = 64MB

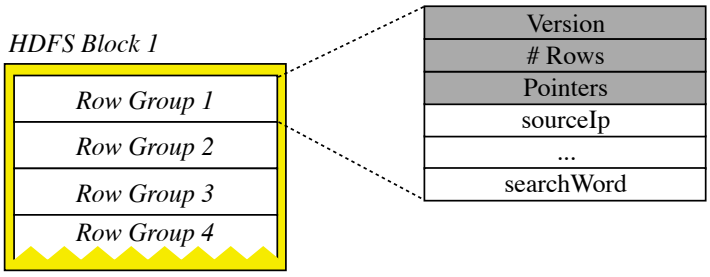
HDFS Block Format

Average Record Size: 100 bytes
#Total Records = 1,000,000
Row Group Size = 200,000 records
HDFS Block Size = 64MB



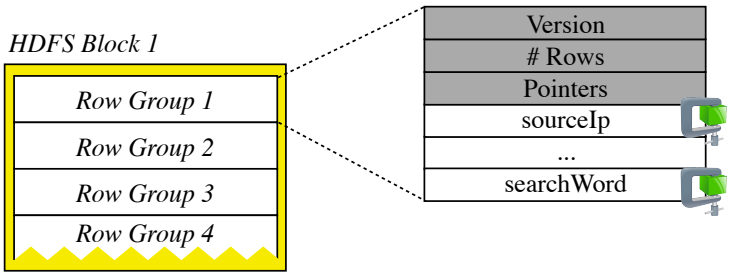
HDFS Block Format

Average Record Size: 100 bytes
#Total Records = 1,000,000
Row Group Size = 200,000 records
HDFS Block Size = 64MB



HDFS Block Format

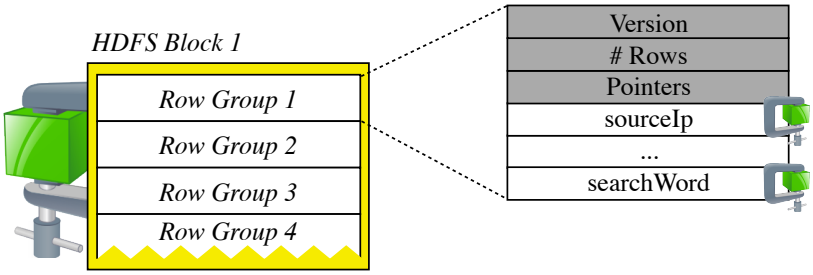
Average Record Size: 100 bytes
#Total Records = 1,000,000
Row Group Size = 200,000 records
HDFS Block Size = 64MB



+ Columns in Row Groups are compressed

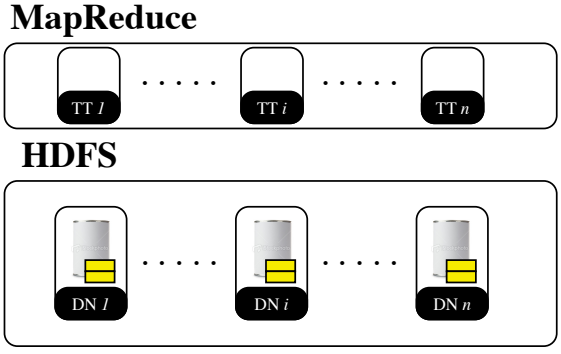
HDFS Block Format

Average Record Size: 100 bytes
#Total Records = 1,000,000
Row Group Size = 200,000 records
HDFS Block Size = 64MB

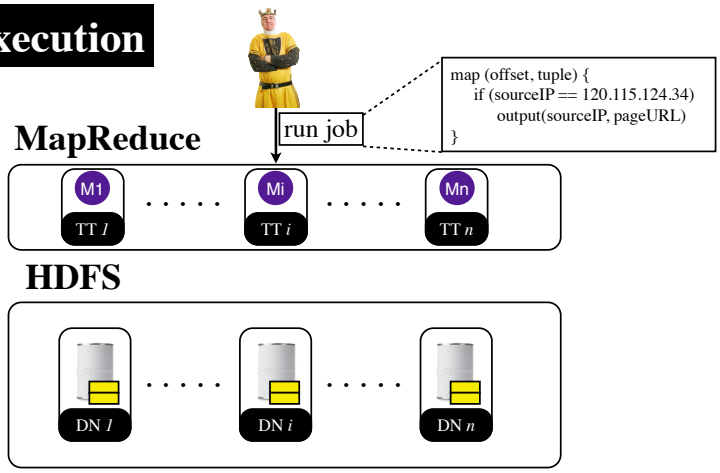


+ Columns in Row Groups are compressed
+ Further compression at the HDFS block level

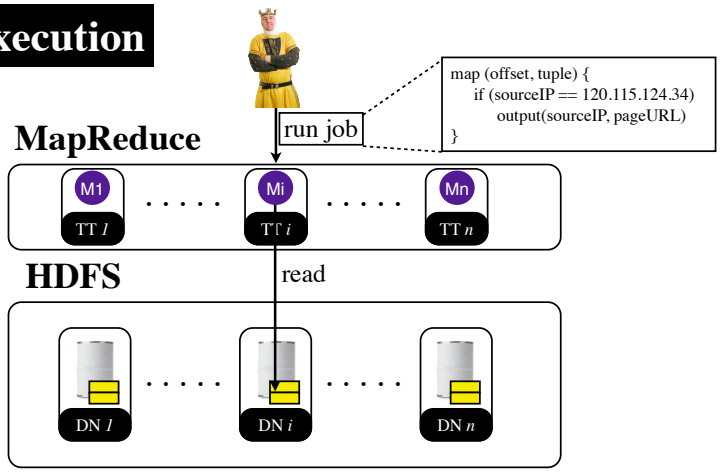
Job Execution



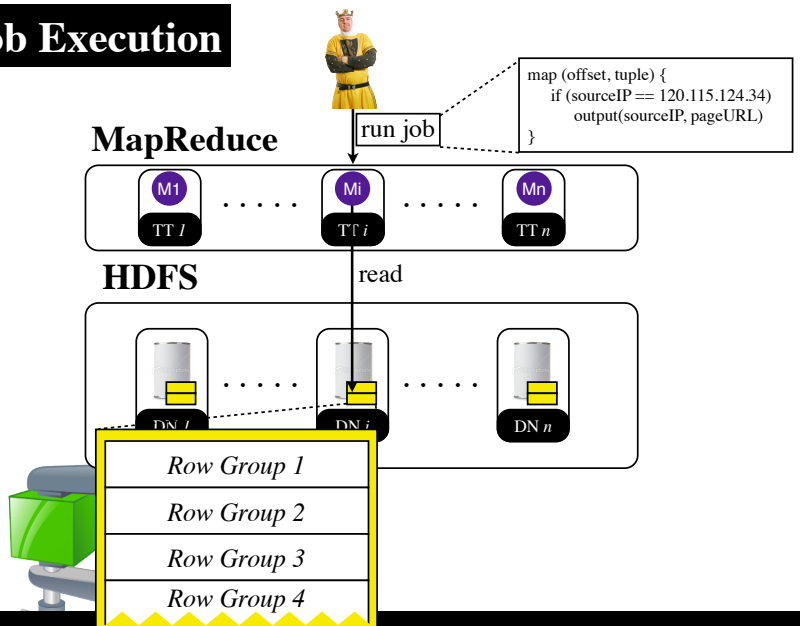
Job Execution



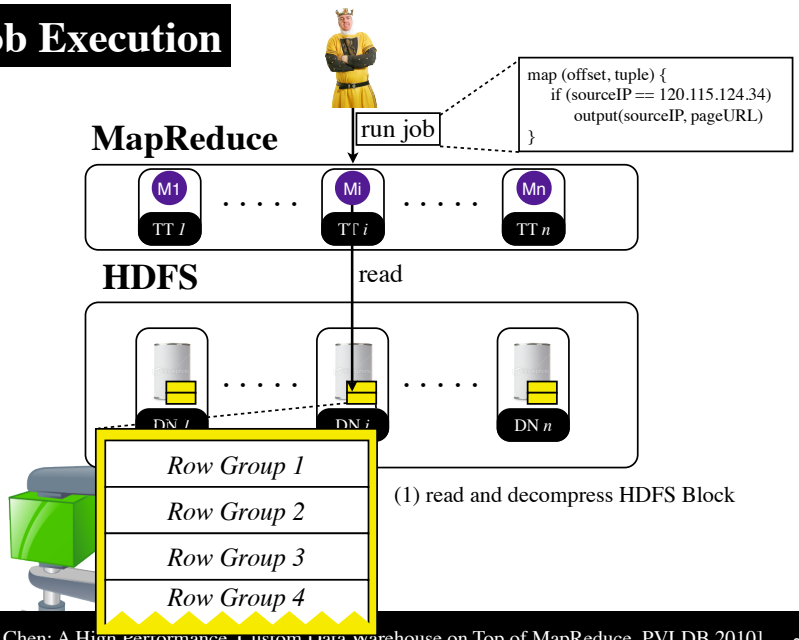
Job Execution



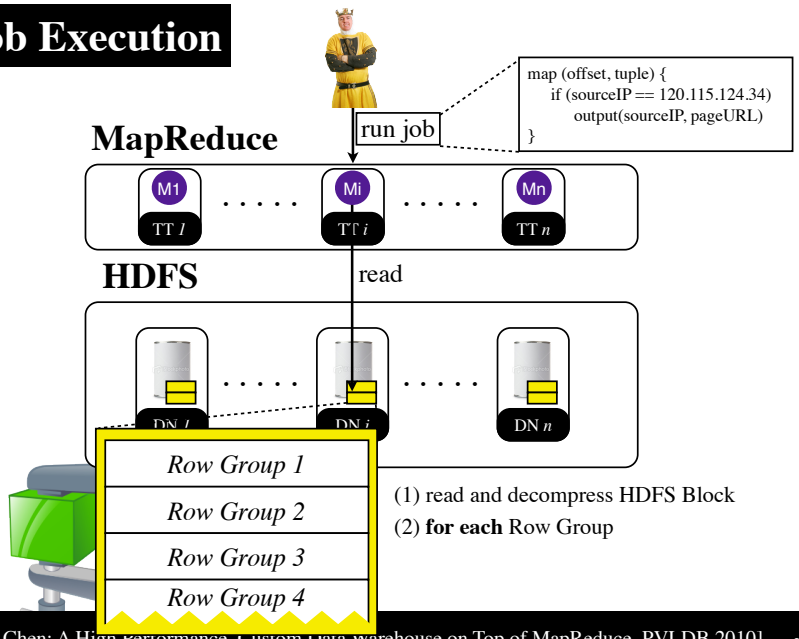
Job Execution



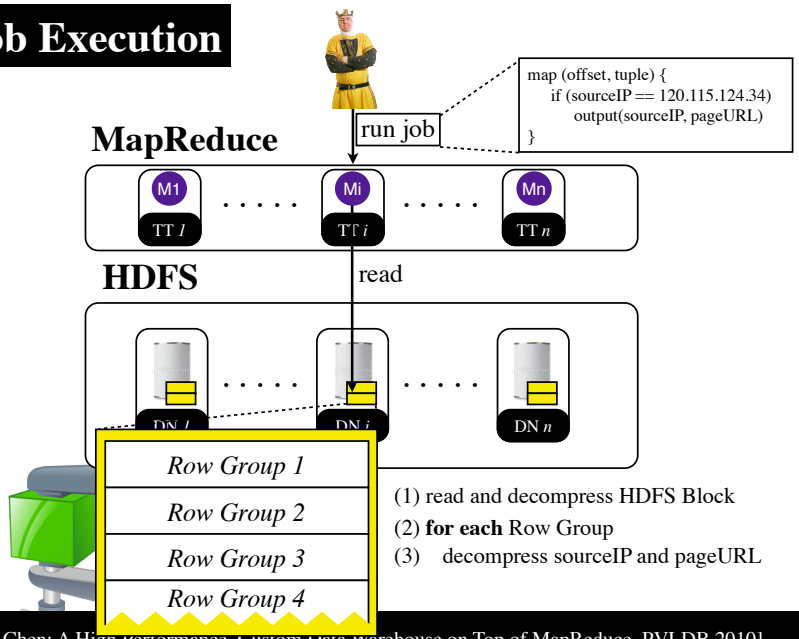
Job Execution



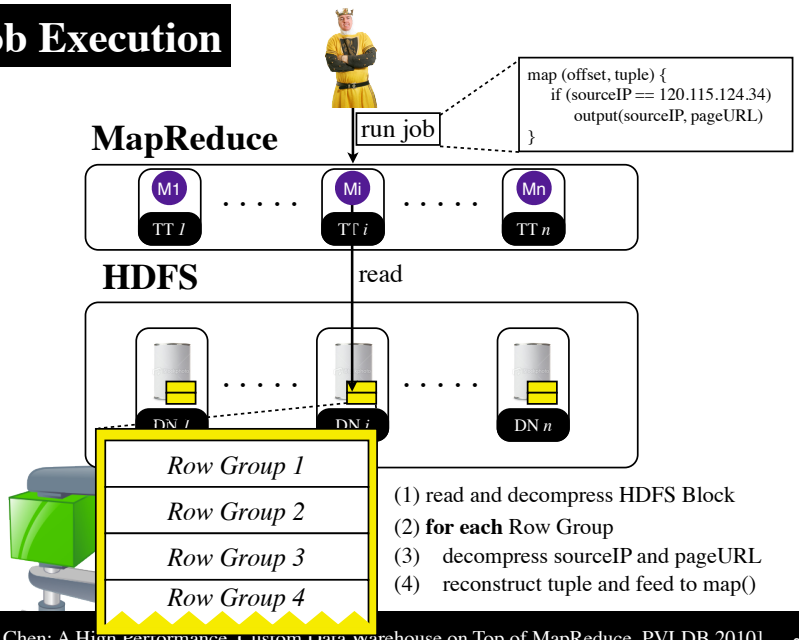
Job Execution



Job Execution



Job Execution



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

Data Layouts in MapReduce

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

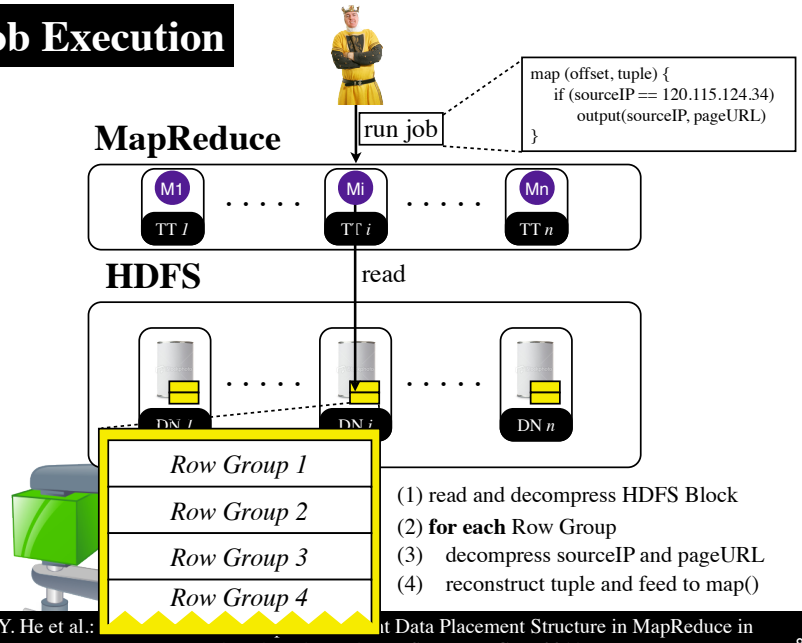
Data Layouts in MapReduce

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]

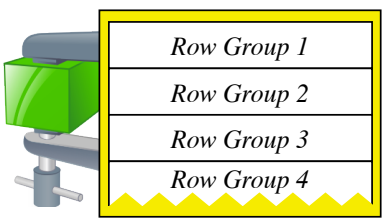
Job Execution



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

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

RecordReader

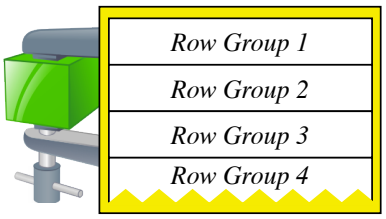


Reading HDFS Blocks with Cheetah

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

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

RecordReader

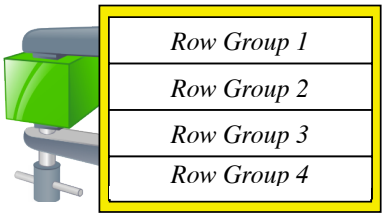


Reading HDFS Blocks with Cheetah

(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



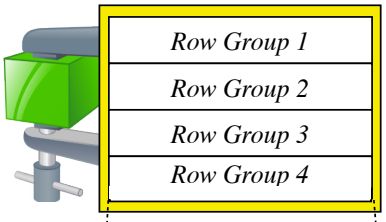
Reading HDFS Blocks with Cheetah

(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

[Y. He et al.: RCFile: A Fast and Space-Efficient Data Placement Structure in MapReduce in MapReduce-based Warehouse Systems. ICDE 2011] 96

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()

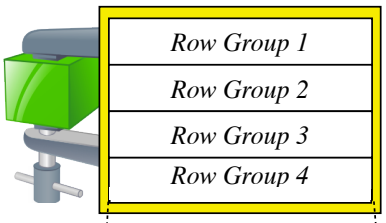
Reading HDFS Blocks with RCFile

RCFile Format

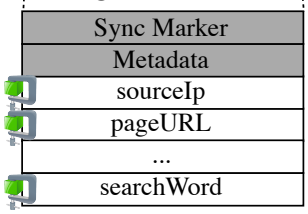
Sync Marker
Metadata
sourceIp
pageURL
...
searchWord

[Y. He et al.: RCFile: A Fast and Space-Efficient Data Placement Structure in MapReduce in MapReduce-based Warehouse Systems. ICDE 2011] 96

RecordReader



RCFile Format



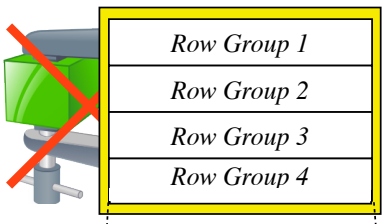
Reading HDFS Blocks with Cheetah

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

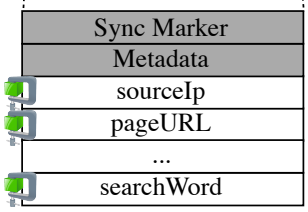
Reading HDFS Blocks with RCFile

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

RecordReader



RCFile Format



Reading HDFS Blocks with Cheetah

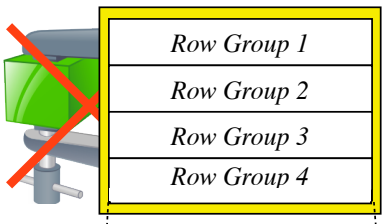
~~(1) read and decompress HDFS Block~~
(2) **for each** Row Group
(3) decompress sourceIP and pageURL
(4) reconstruct tuple and feed to map()

Reading HDFS Blocks with RCFile

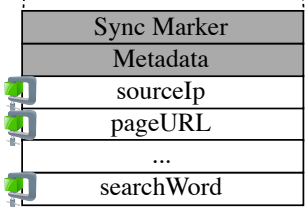
(1) **for each** Row Group
(2) read sourceIP and pageURL

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

RecordReader



RCFile Format



Reading HDFS Blocks with Cheetah

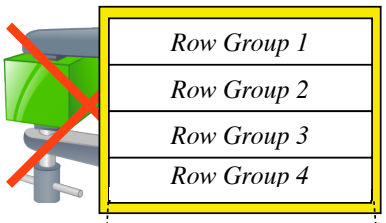
~~(1) read and decompress HDFS Block~~
(2) **for each** Row Group
~~(3) decompress sourceIP and pageURL~~
(4) reconstruct tuple and feed to map()

Reading HDFS Blocks with RCFile

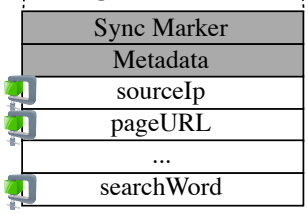
(1) **for each** Row Group
(2) read sourceIP and pageURL
(3) decompress sourceIP

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

RecordReader



RCFile Format



Reading HDFS Blocks with Cheetah

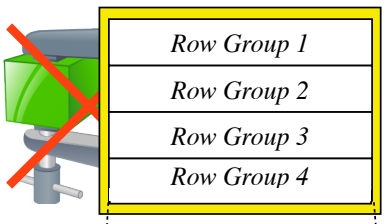
~~(1) read and decompress HDFS Block~~
(2) **for each** Row Group
~~(3) decompress sourceIP and pageURL~~
(4) reconstruct tuple and feed to map()

Reading HDFS Blocks with RCFile

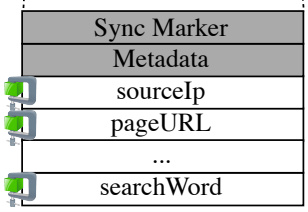
(1) **for each** Row Group
(2) read sourceIP and pageURL
(3) decompress sourceIP
(4) **for each** sourceIP value
(5) **if** sourceIP == 120.115.124.34

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

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
(6) decompress pageURL
(7) reconstruct tuple and feed to map()

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

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

[A. Floratou et al.: Column-Oriented Storage Techniques for MapReduce. PVLDB 2011] 99

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

[A. Floratou et al.: Column-Oriented Storage Techniques for MapReduce. PVLDB 2011] 99

Remarks on Cheetah-Storage and RCFile

(1) I/O elimination becomes difficult

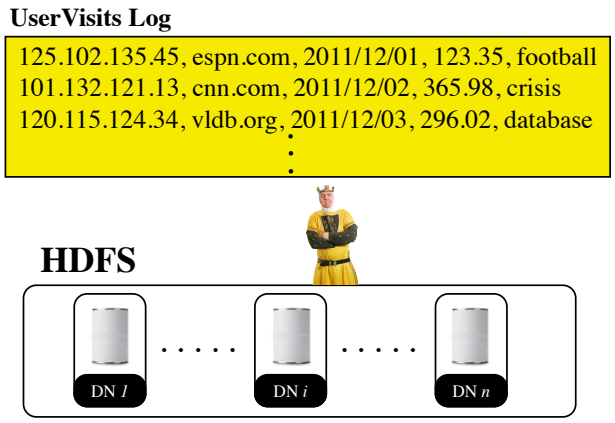
(2) Tuning the row-group size becomes critical

(3) Overhead for per-Row Group metadata

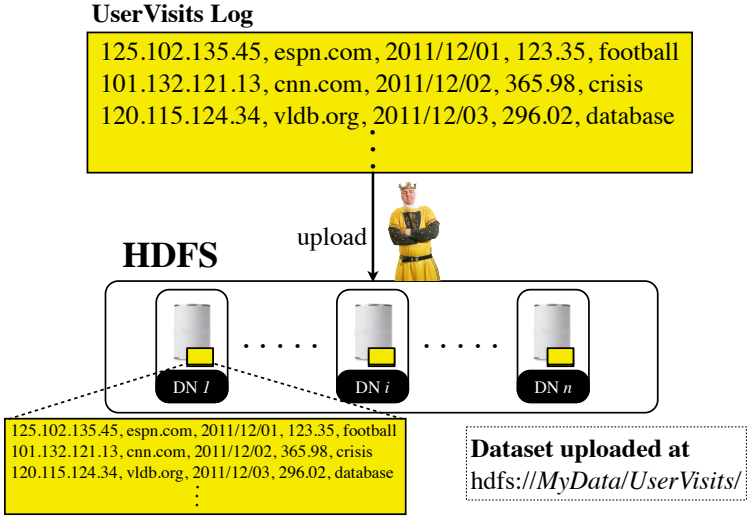
CIF Approach:

CFile + Cheetah Storage (or RCFile)

Data Upload --- Upload UserVisits ---

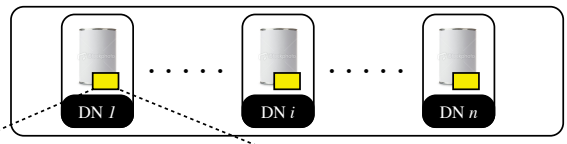


Data Upload --- Upload UserVisits ---



Data Upload --- Run Parallel Loader ---

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 ---

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

HDFS
DN 1
DN i
DN n

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
⋮

[A. Floratou et al.: Column-Oriented Storage Techniques for MapReduce. PVLDB 2011] 101

Data Upload --- Run Parallel Loader ---

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

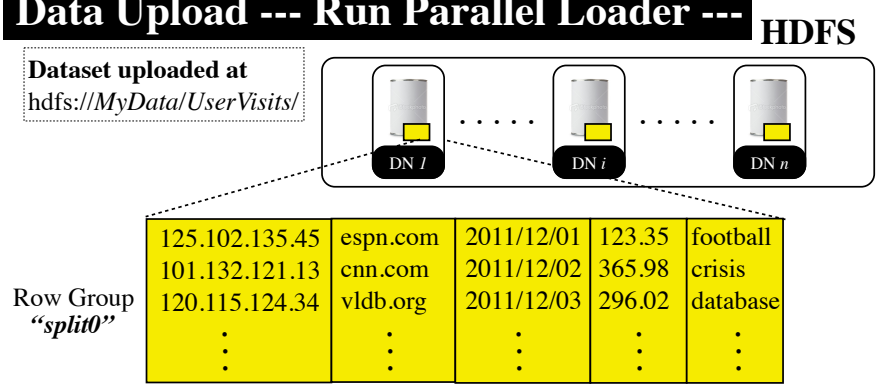
HDFS
DN 1
DN i
DN n

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
⋮

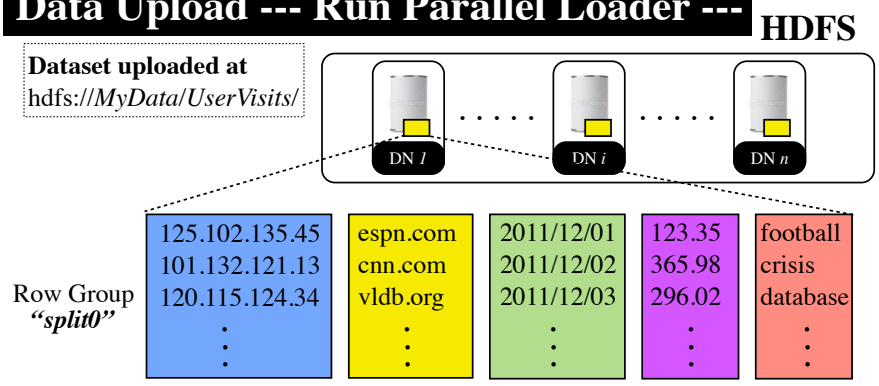
Row Group
"split0"

[A. Floratou et al.: Column-Oriented Storage Techniques for MapReduce. PVLDB 2011] 101

Data Upload --- Run Parallel Loader ---

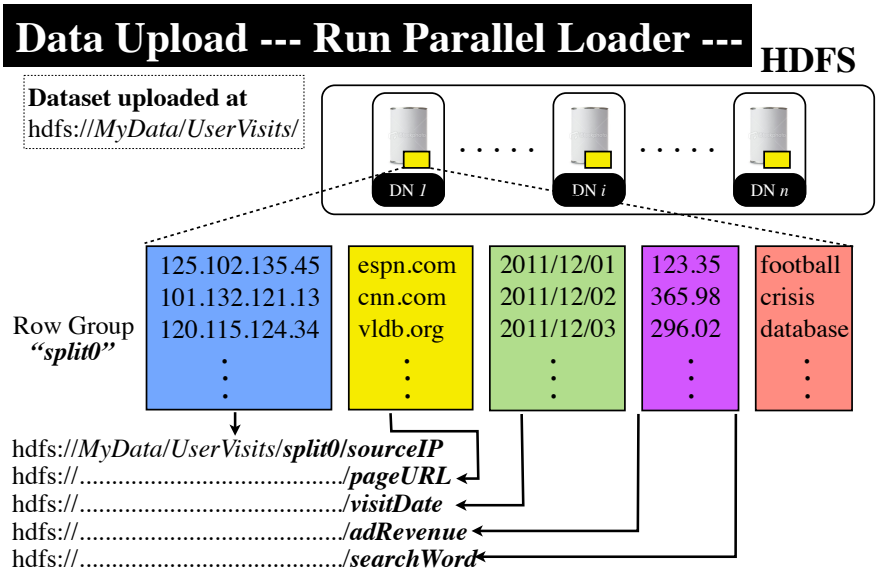


Data Upload --- Run Parallel Loader ---



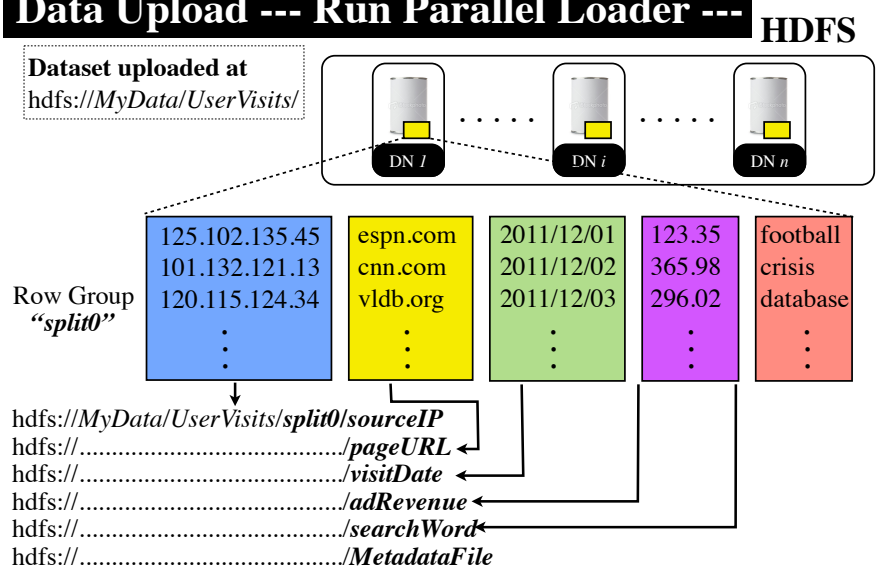
[A. Floratou et al.: Column-Oriented Storage Techniques for MapReduce. PVLDB 2011] 101

Data Upload --- Run Parallel Loader ---

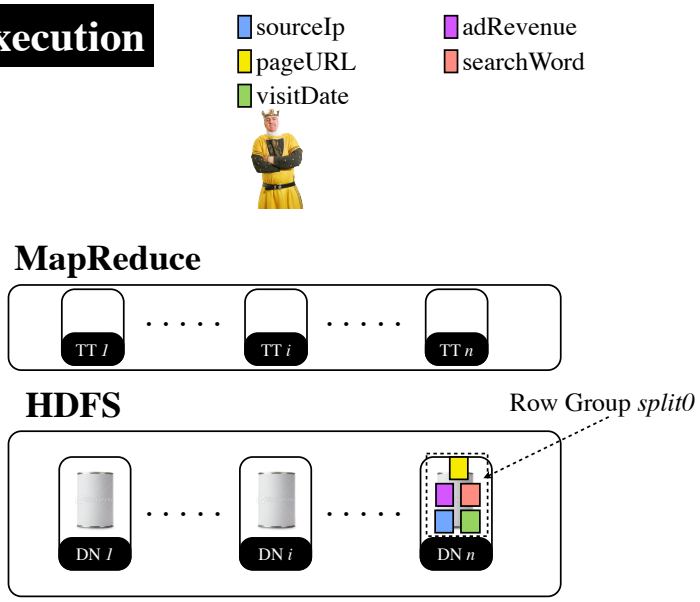


[A. Floratou et al.: Column-Oriented Storage Techniques for MapReduce. PVLDB 2011] 101

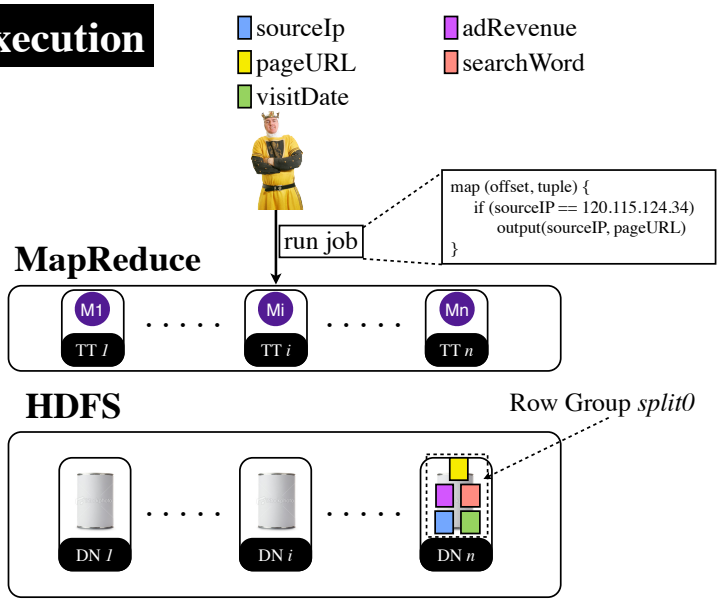
Data Upload --- Run Parallel Loader ---



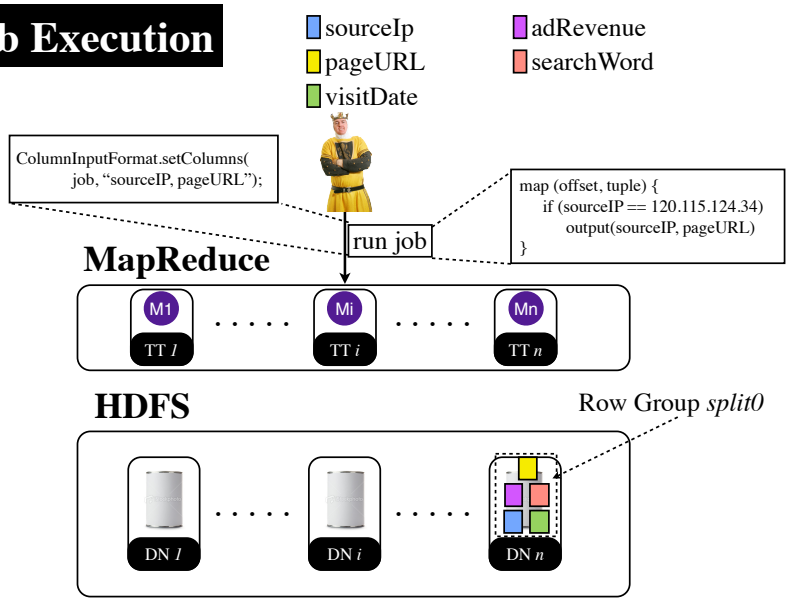
Job Execution



Job Execution

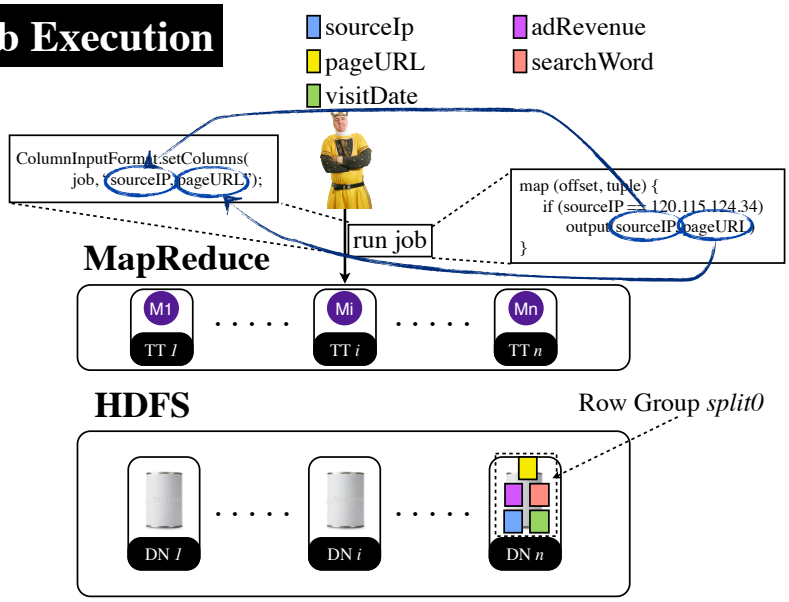


Job Execution



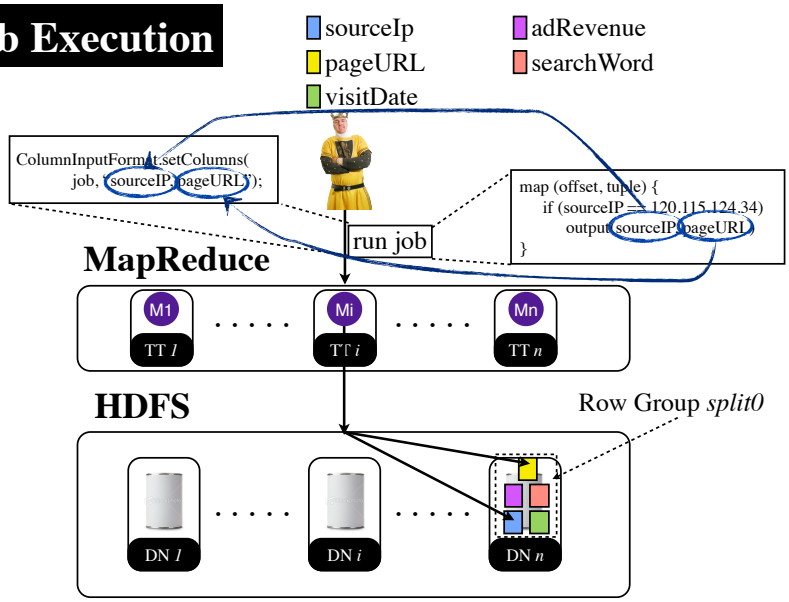
[A. Floratou et al.: Column-Oriented Storage Techniques for MapReduce. PVLDB 2011]

Job Execution



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

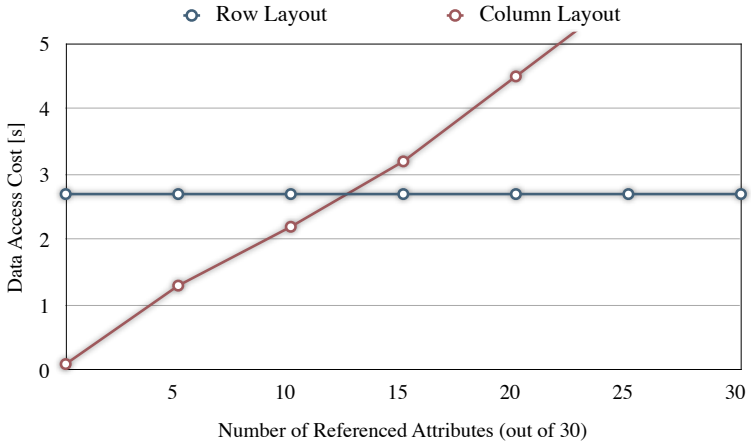
Job Execution



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

PAX Layout in MapReduce

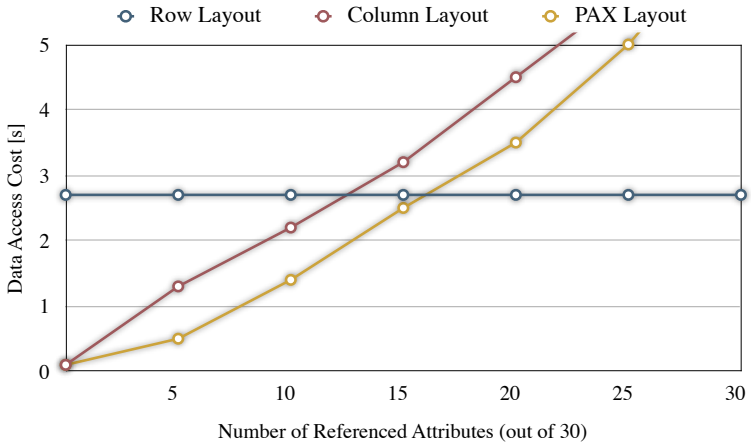
SELECT *a1, a2, ...*
FROM table30Atts



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

PAX Layout in MapReduce

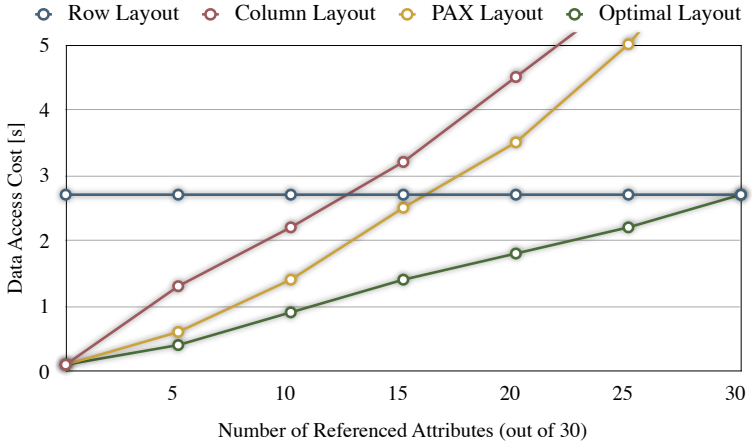
SELECT *a1, a2, ...*
FROM table30Atts



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

Far from Optimal Layout

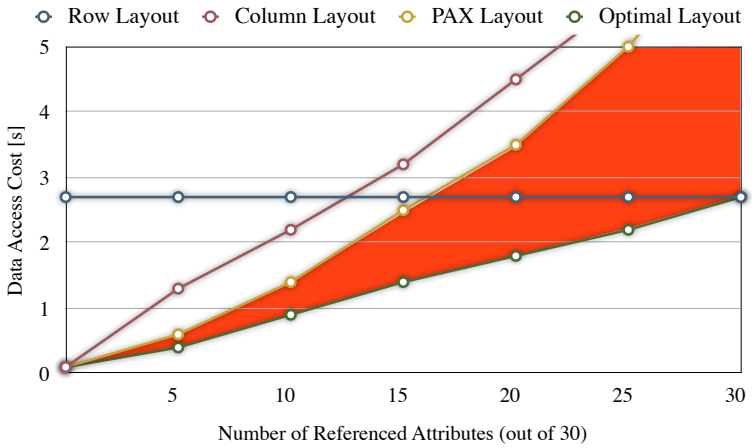
SELECT *a1, a2, ...*
FROM table30Atts



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

Far from Optimal Layout

SELECT *a1, a2, ...*
FROM table30Atts



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

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

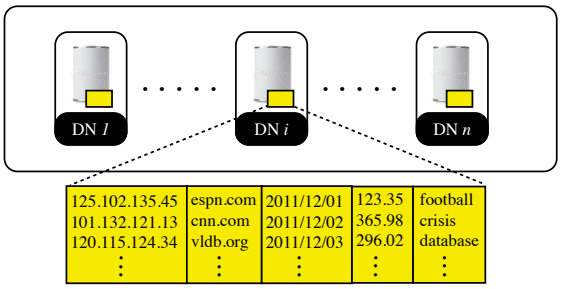
Initial	2009	2010	2011	2011
Row	CFile	Cheetah	RCFile	CIF
Read Unnecessary columns				
	Tuple Reconstruction	Tuple Reconstruction	Tuple Reconstruction	Tuple Reconstruction
	High network costs			
		Block level compression		
		Poor I/O Saving	Poor I/O Saving	

Trojan Data Layouts

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

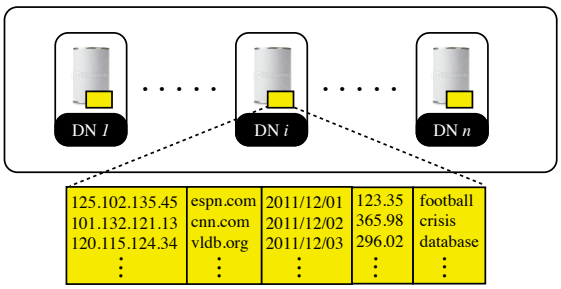
Idea

HDFS



Idea

HDFS

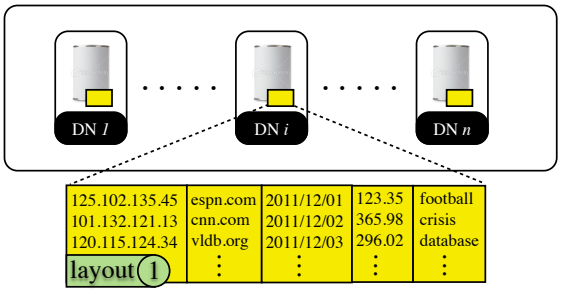


Job 1

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

Idea

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
layout 1	⋮	⋮	⋮	⋮

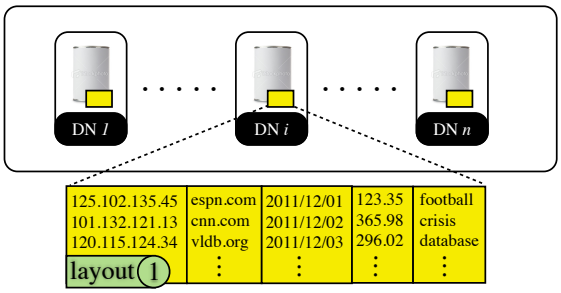
Job 1

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

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

Idea

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
layout 1	⋮	⋮	⋮	⋮

Job 1

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

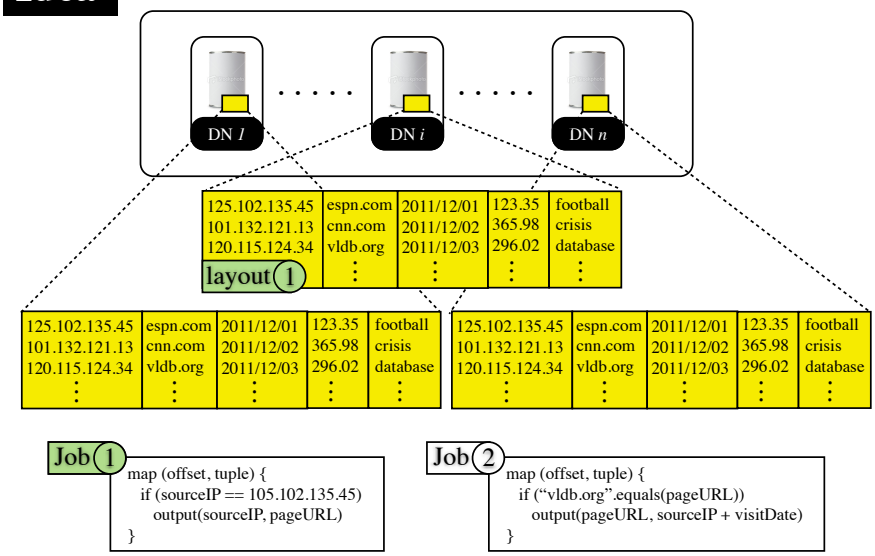
Job 2

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

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

Idea

HDFS



Job 1

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

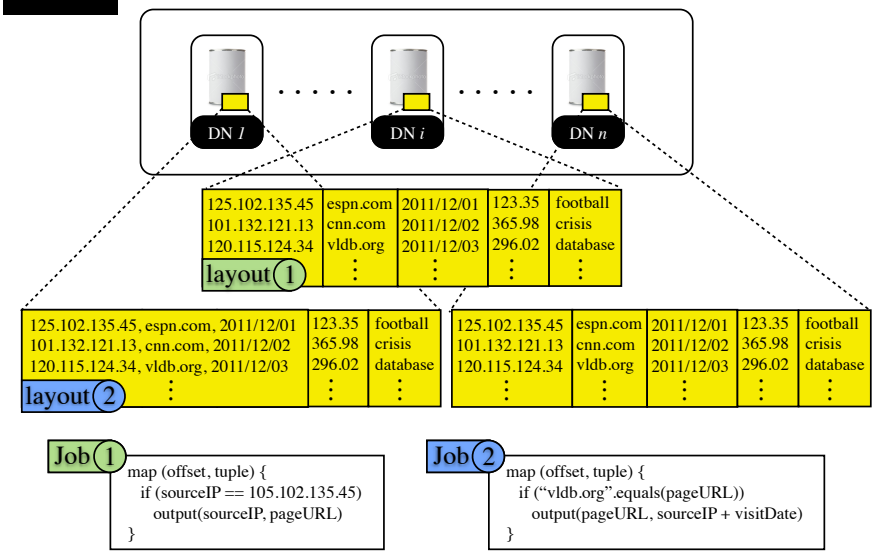
Job 2

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

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

Idea

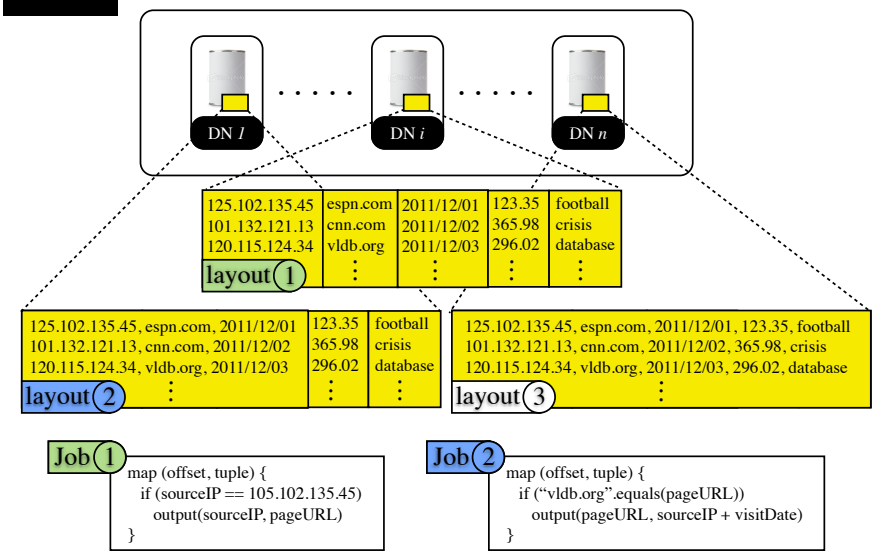
HDFS



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

Idea

HDFS



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

Single HDFS Block Replica

Columns picture: http://www.istockphoto.com/stock-photo-10676885-pile-of-words.php
Filter picture: http://www.istockphoto.com/stock-photo-8235648-kitchen-funnel.php
Packing picture: http://www.istockphoto.com/stock-photo-1373749-c-clamp.php

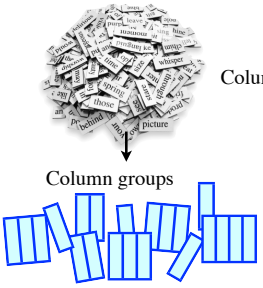
[A. Jindal, J. Quiane, J. Dittrich: Trojan Data Layouts: Right Shoes for a Running Elephant. SoCC 2011]

Single HDFS Block Replica

Columns

Columns picture: http://www.istockphoto.com/stock-photo-10676885-pile-of-words.php
Filter picture: http://www.istockphoto.com/stock-photo-8235648-kitchen-funnel.php
Packing picture: http://www.istockphoto.com/stock-photo-1373749-c-clamp.php

Single HDFS Block Replica

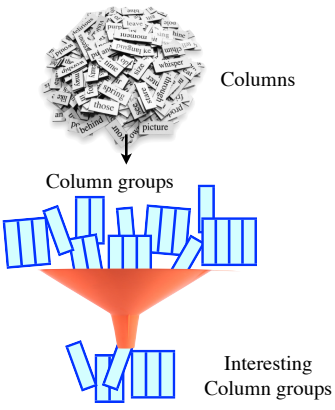


Columns
Column groups

Columns picture: http://www.istockphoto.com/stock-photo-10676885-pile-of-words.php
Filter picture: http://www.istockphoto.com/stock-photo-8235648-kitchen-funnel.php
Packing picture: http://www.istockphoto.com/stock-photo-1373749-c-clamp.php

[A. Jindal, J. Quiane, J. Dittrich: Trojan Data Layouts: Right Shoes for a Running Elephant. SoCC 2011] 109

Single HDFS Block Replica

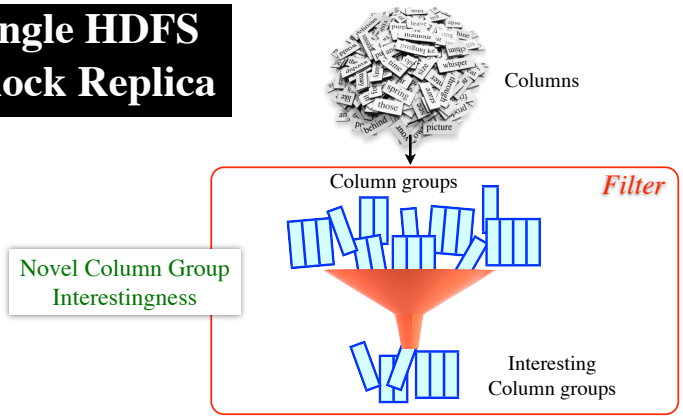


Columns
Column groups
Interesting Column groups

Columns picture: http://www.istockphoto.com/stock-photo-10676885-pile-of-words.php
Filter picture: http://www.istockphoto.com/stock-photo-8235648-kitchen-funnel.php
Packing picture: http://www.istockphoto.com/stock-photo-1373749-c-clamp.php

[A. Jindal, J. Quiane, J. Dittrich: Trojan Data Layouts: Right Shoes for a Running Elephant. SoCC 2011] 109

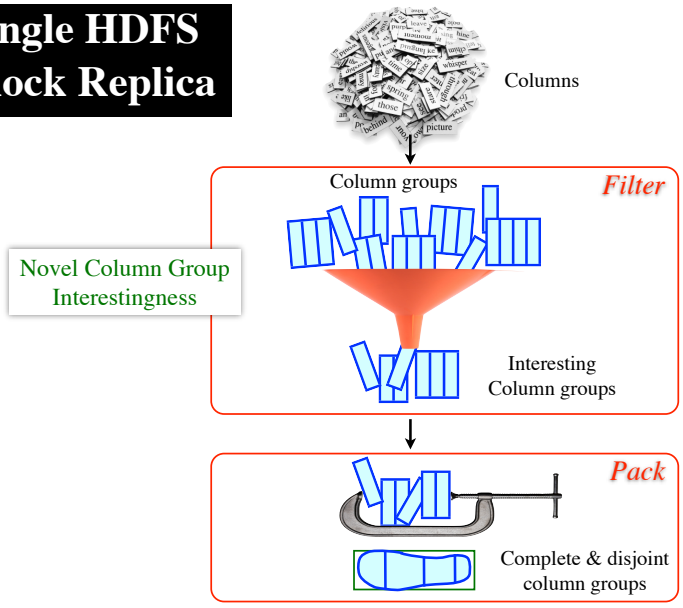
Single HDFS Block Replica



Columns picture: http://www.istockphoto.com/stock-photo-10676885-pile-of-words.php
Filter picture: http://www.istockphoto.com/stock-photo-8235648-kitchen-funnel.php
Packing picture: http://www.istockphoto.com/stock-photo-1373749-c-clamp.php

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

Single HDFS Block Replica

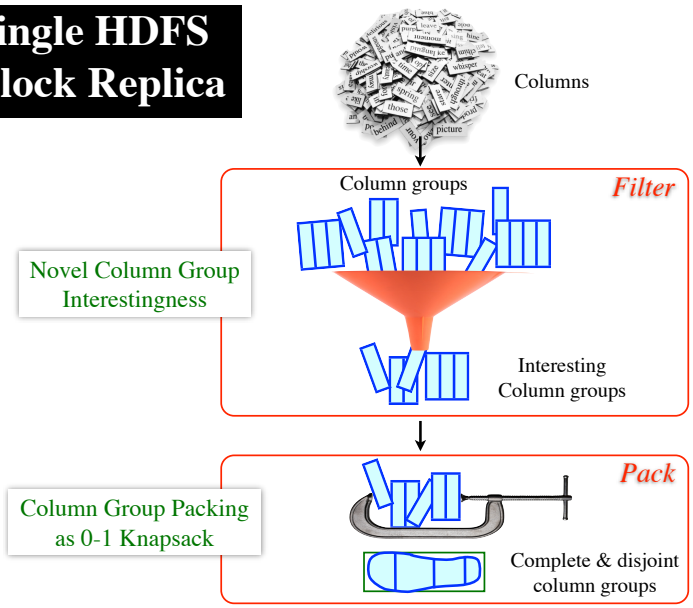


Columns picture: http://www.istockphoto.com/stock-photo-10676885-pile-of-words.php
Filter picture: http://www.istockphoto.com/stock-photo-8235648-kitchen-funnel.php
Packing picture: http://www.istockphoto.com/stock-photo-1373749-c-clamp.php

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

109

Single HDFS Block Replica

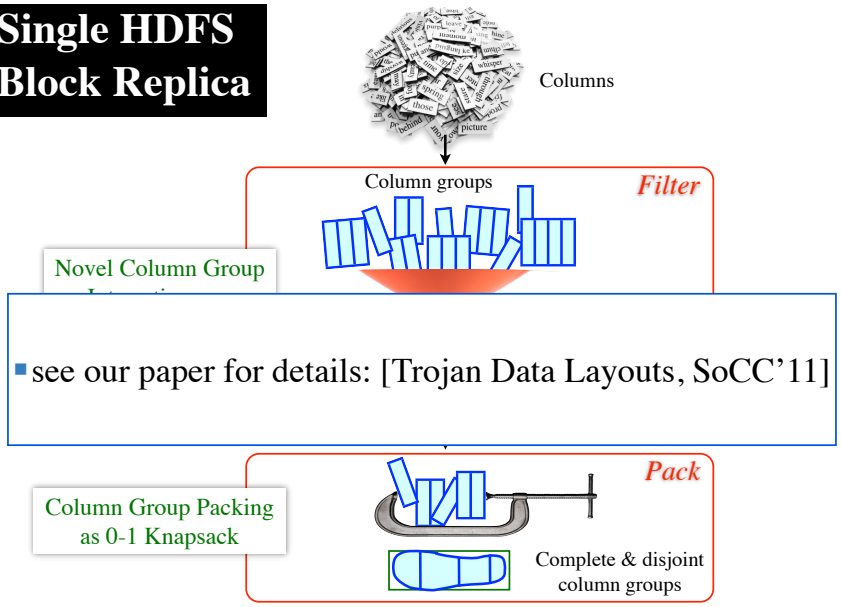


Columns picture: http://www.istockphoto.com/stock-photo-10676885-pile-of-words.php
Filter picture: http://www.istockphoto.com/stock-photo-8235648-kitchen-funnel.php
Packing picture: http://www.istockphoto.com/stock-photo-1373749-c-clamp.php

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

109

Single HDFS Block Replica



[A. Jindal, J. Quiane, J. Dittrich: Trojan Data Layouts: Right Shoes for a Running Elephant. SoCC 2011] 109

Columns picture: http://www.istockphoto.com/stock-photo-10676885-pile-of-words.php
Filter picture: http://www.istockphoto.com/stock-photo-8235648-kitchen-funnel.php
Packing picture: http://www.istockphoto.com/stock-photo-1373749-c-clamp.php

Multiple HDFS Block Replica

[A. Jindal, J. Quiane, J. Dittrich: Trojan Data Layouts: Right Shoes for a Running Elephant. SoCC 2011] 110

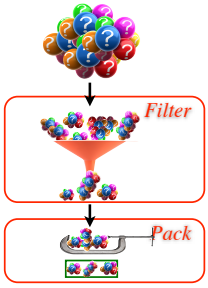
Queries picture: http://www.istockphoto.com/stock-photo-14278066-colorful-balls-with-question-marks.php

Multiple HDFS Block Replica

[A. Jindal, J. Quiane, J. Dittrich: Trojan Data Layouts: Right Shoes for a Running Elephant. SoCC 2011] 110

Queries picture: http://www.istockphoto.com/stock-photo-14278066-colorful-balls-with-question-marks.php

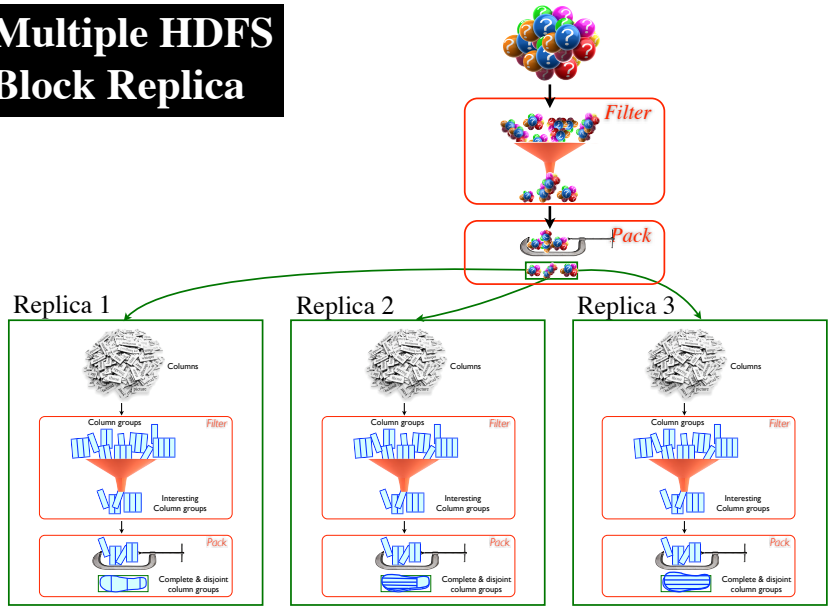
Multiple HDFS Block Replica



Queries picture: http://www.istockphoto.com/stock-photo-14278066-colorful-balls-with-question-marks.php

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

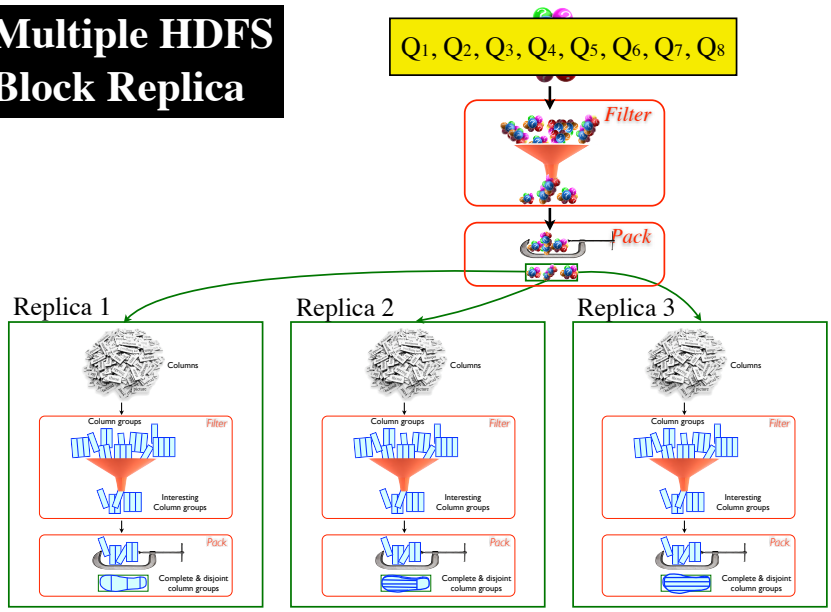
Multiple HDFS Block Replica



Queries picture: http://www.istockphoto.com/stock-photo-14278066-colorful-balls-with-question-marks.php

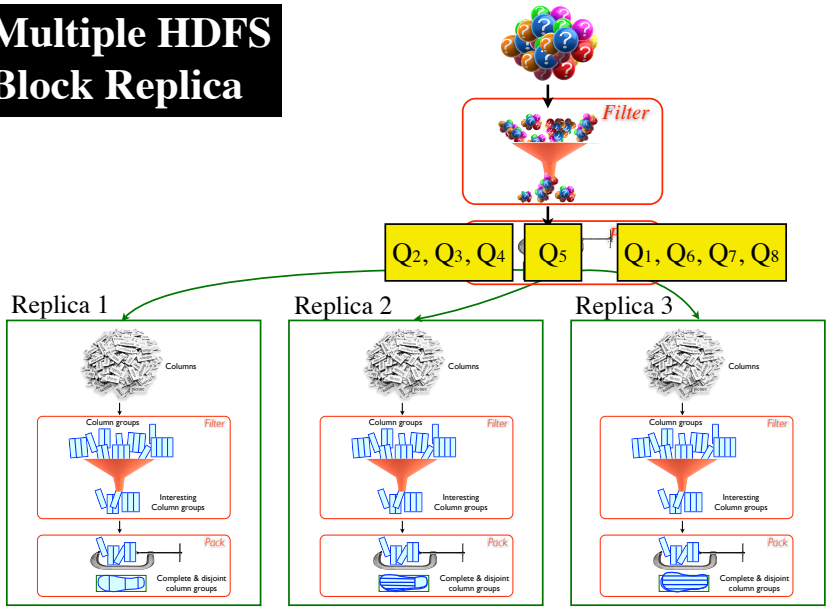
[A. Jindal, J. Quiane, J. Dittrich: Trojan Data Layouts: Right Shoes for a Running Elephant. SoCC 2011]

Multiple HDFS Block Replica



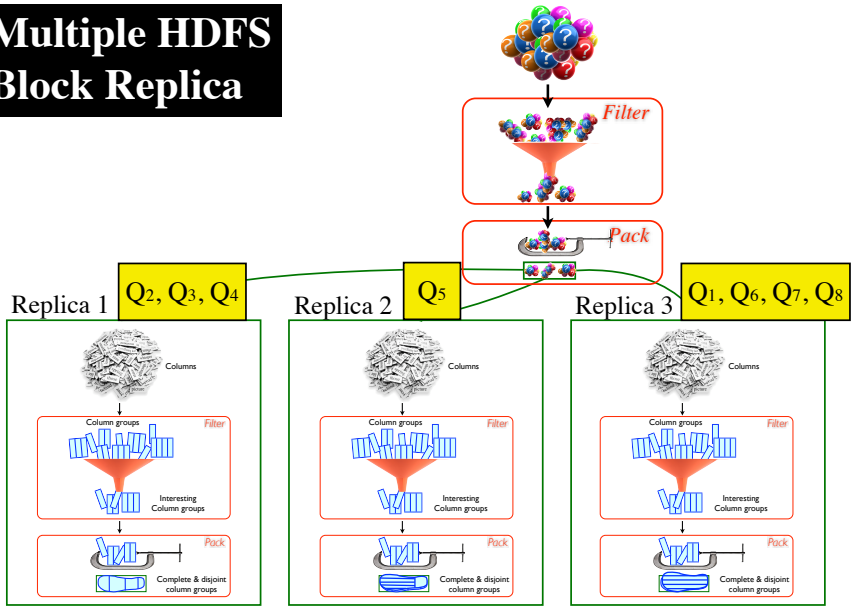
[A. Jindal, J. Quiane, J. Dittrich: Trojan Data Layouts: Right Shoes for a Running Elephant. SoCC 2011]

Multiple HDFS Block Replica



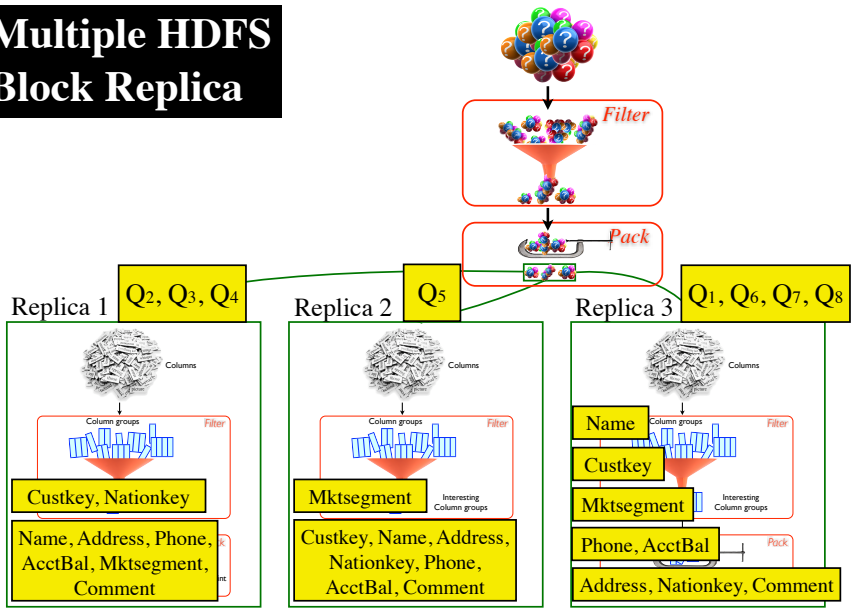
[A. Jindal, J. Quiane, J. Dittrich: Trojan Data Layouts: Right Shoes for a Running Elephant. SoCC 2011]

Multiple HDFS Block Replica



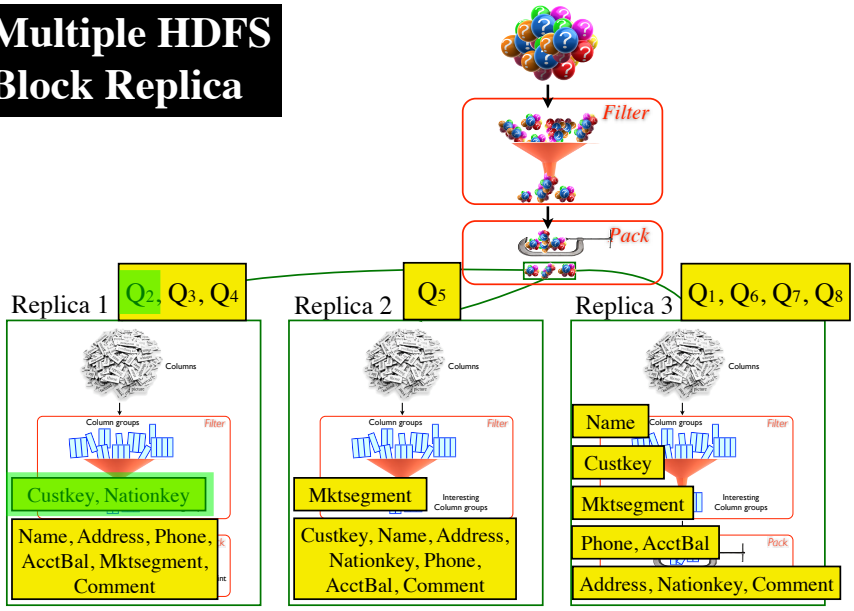
[A. Jindal, J. Quiane, J. Dittrich: Trojan Data Layouts: Right Shoes for a Running Elephant. SoCC 2011]

Multiple HDFS Block Replica



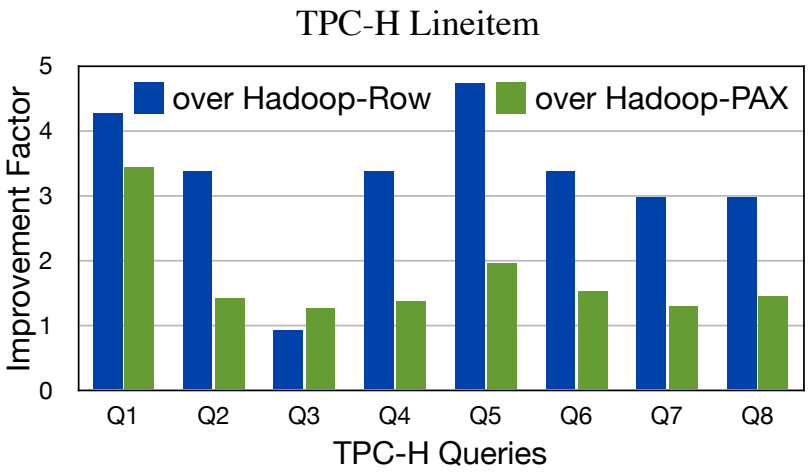
[A. Jindal, J. Quiane, J. Dittrich: Trojan Data Layouts: Right Shoes for a Running Elephant. SoCC 2011]

Multiple HDFS Block Replica



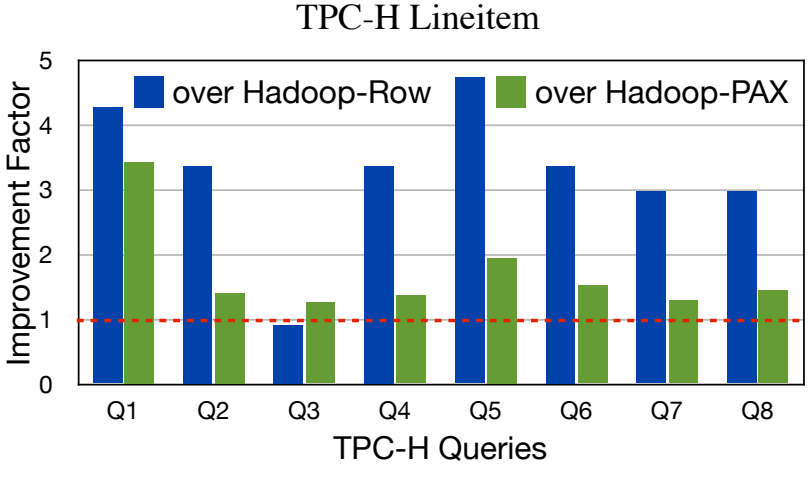
[A. Jindal, J. Quiane, J. Dittrich: Trojan Data Layouts: Right Shoes for a Running Elephant. SoCC 2011]

Trojan Data Layouts Results



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

Trojan Data Layouts Results



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

112

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

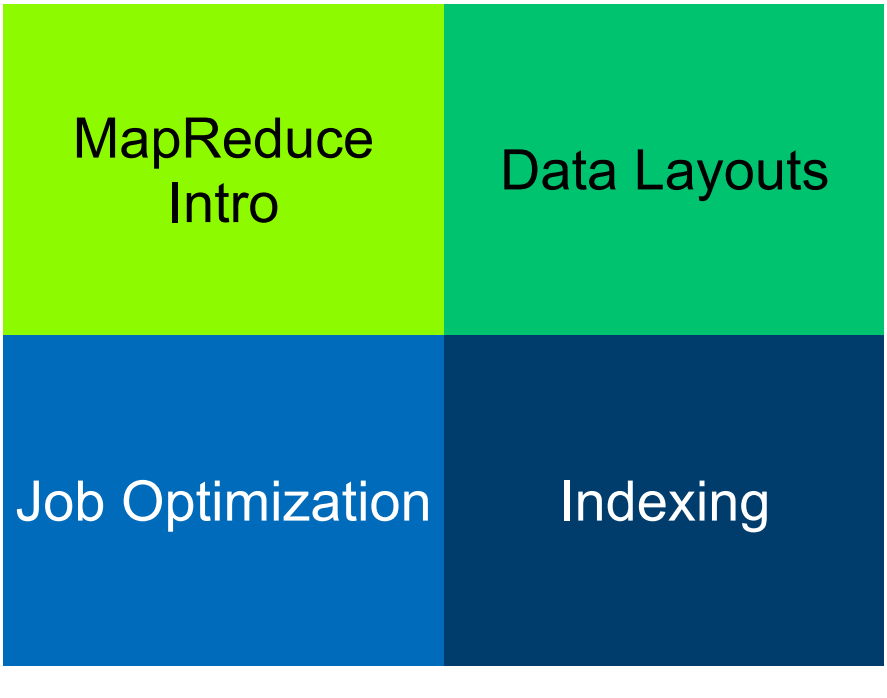
	Low Record Selectivity	High Record Selectivity
Low Attribute Selectivity	Row	Row Groups
Medium Attribute Selectivity	Column Groups	
High Attribute Selectivity	PAX	

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

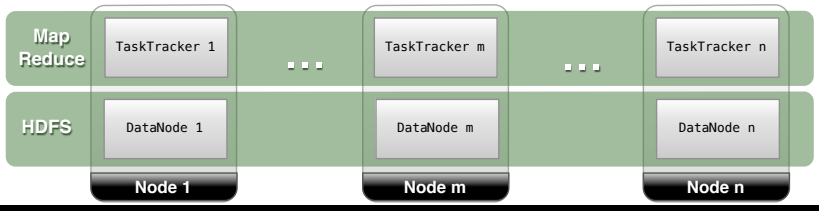


Indexing

DBMS as Data Storage (HadoopDB)

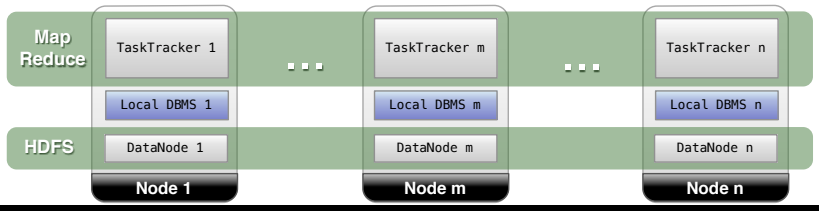
[A. Abouzeid et al.: HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads. PVLDB 2009]

Index Creation



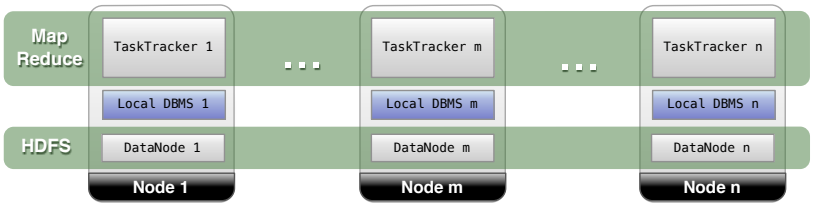
[A. Abouzeid et al.: HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads. PVLDB 2009]

Index Creation



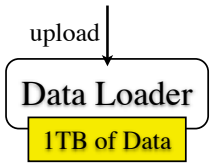
[A. Abouzeid et al.: HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads. PVLDB 2009]

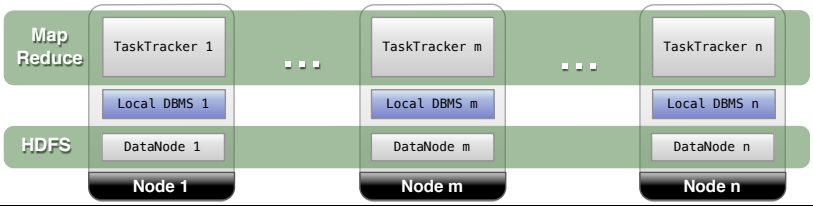
Index Creation



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

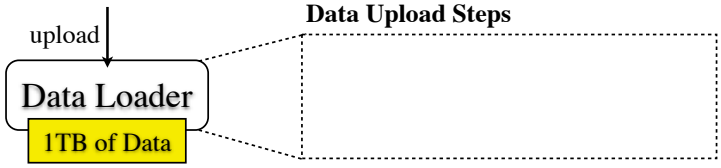
Index Creation

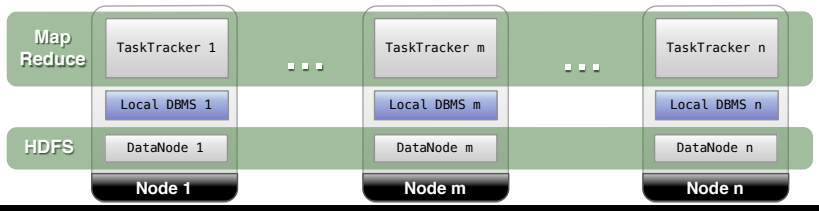




[A. Abouzeid et al.: HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads. PVLDB 2009]

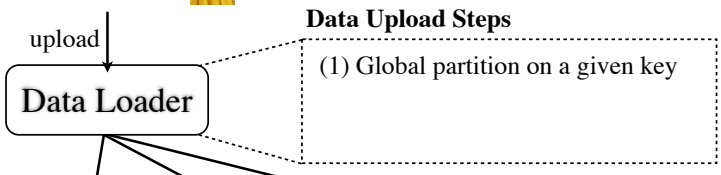
Index Creation

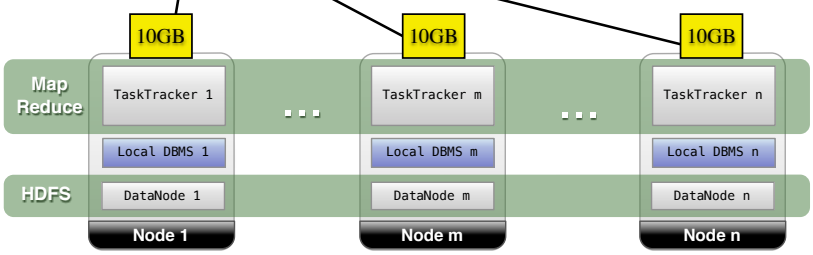




[A. Abouzeid et al.: HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads. PVLDB 2009]

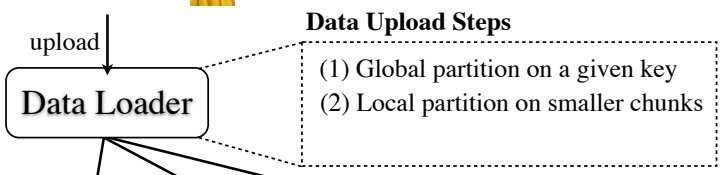
Index Creation

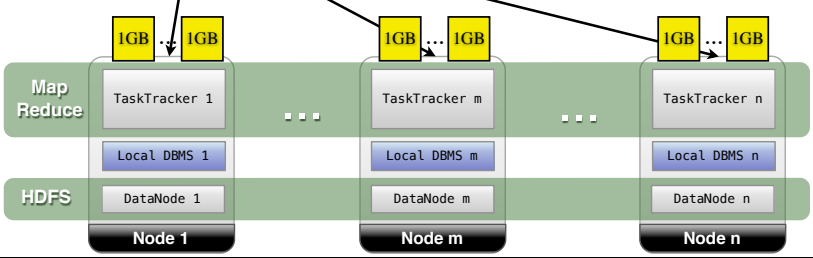




[A. Abouzeid et al.: HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads. PVLDB 2009]

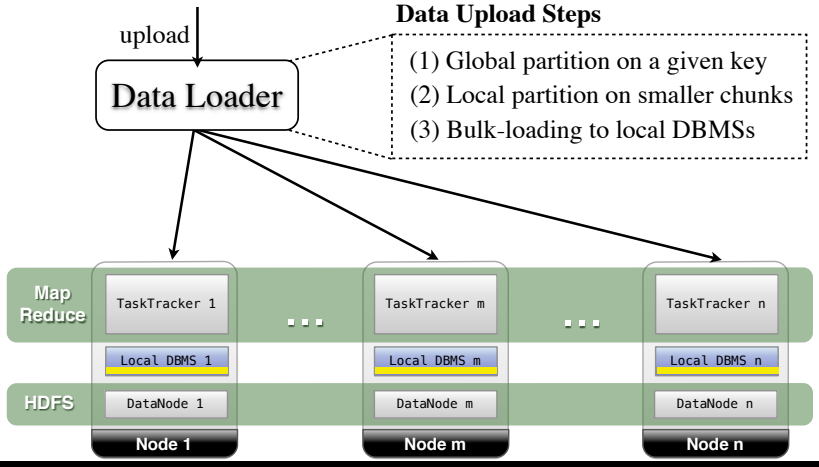
Index Creation





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

Index Creation

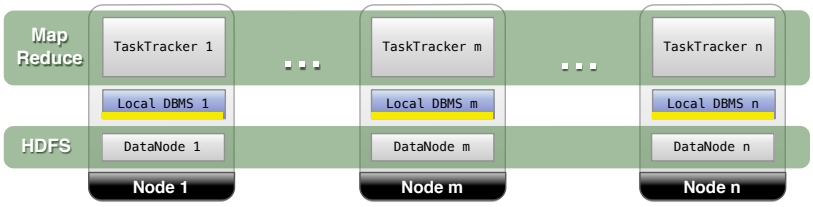


[A. Abouzeid et al.: HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads. PVLDB 2009]

Job Execution

SMS Planner

JobTracker



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

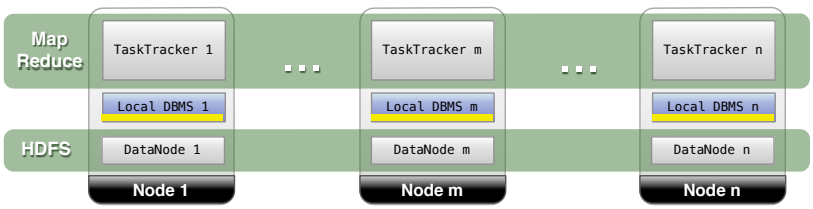
Job Execution

SMS Planner

SQL-like Query

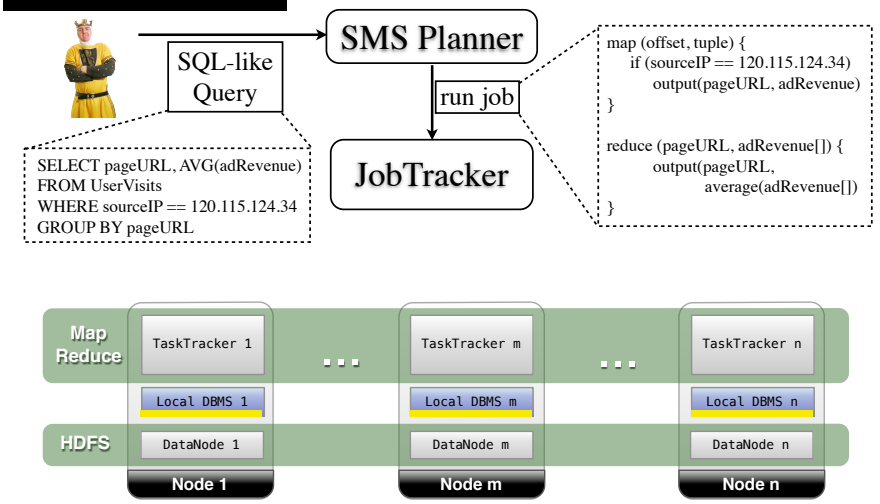
SELECT pageURL, AVG(adRevenue)
FROM UserVisits
WHERE sourceIP == 120.115.124.34
GROUP BY pageURL

JobTracker

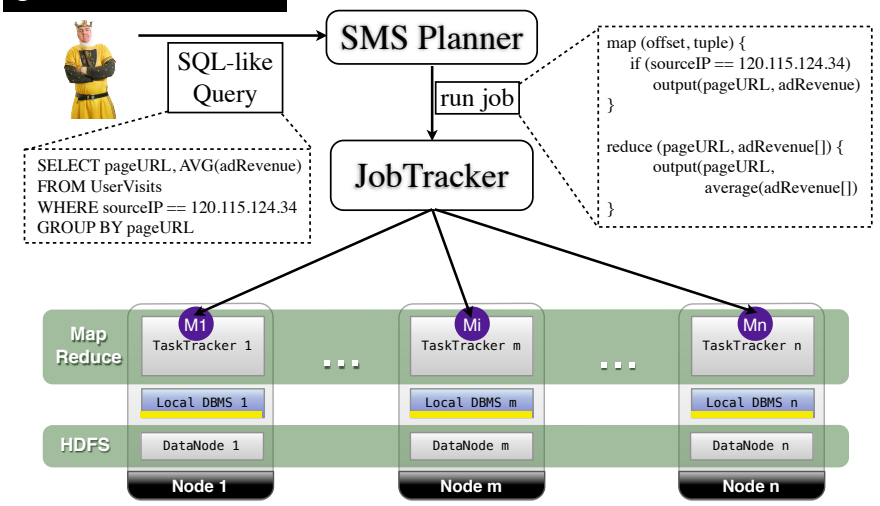


[A. Abouzeid et al.: HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads. PVLDB 2009]

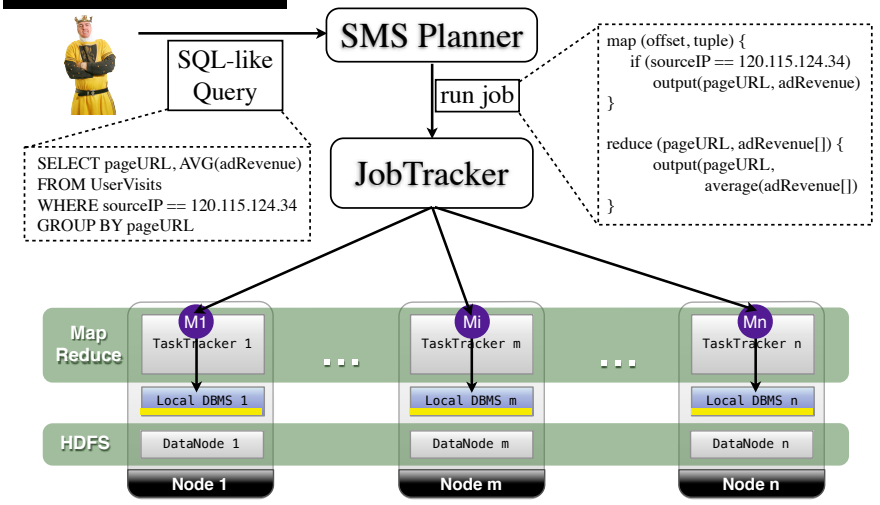
Job Execution



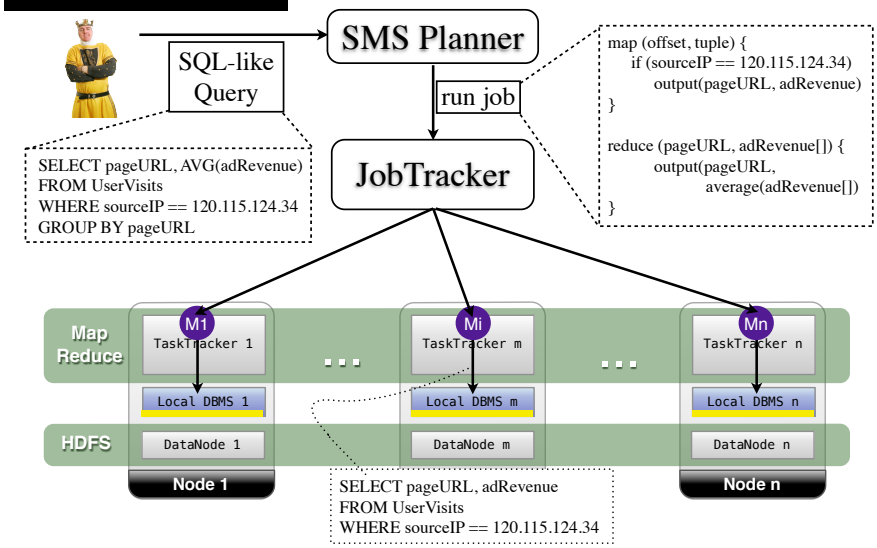
Job Execution



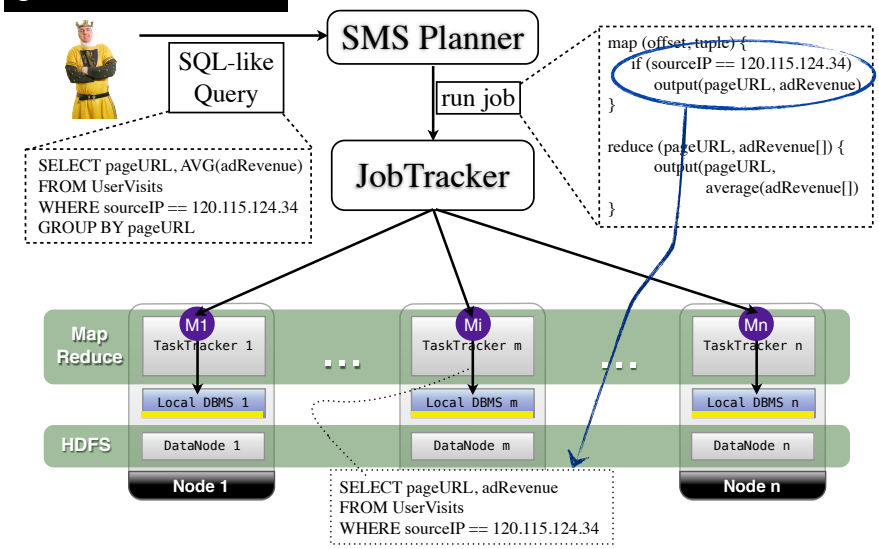
Job Execution



Job Execution



Job Execution



HadoopDB Results (Selection Task)

Rankings Dataset

pageURL
pageRank
avgDuration

HadoopDB Results (Selection Task)

Rankings Dataset

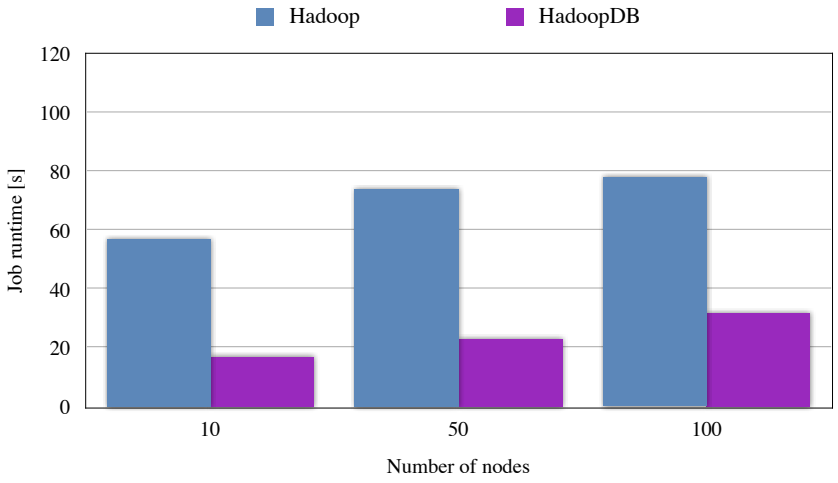
pageURL
pageRank
avgDuration

Query

SELECT pageURL, pageRank
FROM Rankings
WHERE pageRank > 10

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

HadoopDB Results (Selection Task)



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

Indexing in MapReduce

Indexing in MapReduce

2009
HadoopDB
Still a database

But...
inside MapReduce?

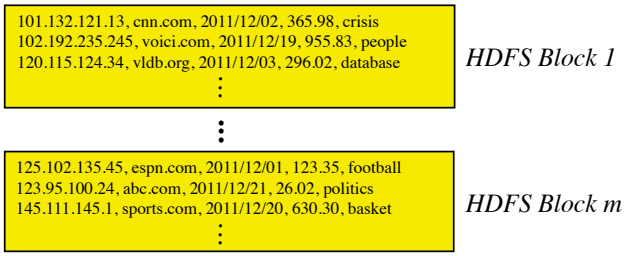
Indexing Levels

Indexing Levels

- File Level: *filters HDFS Blocks*

Indexing Levels

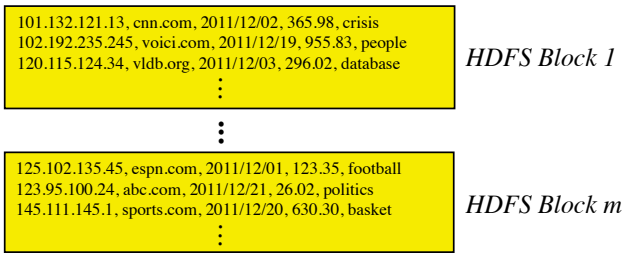
- File Level: *filters HDFS Blocks*



Indexing Levels

Filter Condition:
sourceIP == 120.115.124.34

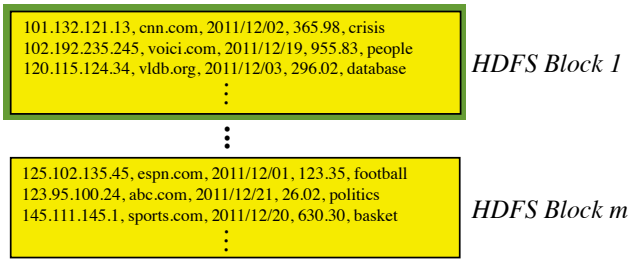
- File Level: *filters HDFS Blocks*



Indexing Levels

Filter Condition:
sourceIP == 120.115.124.34

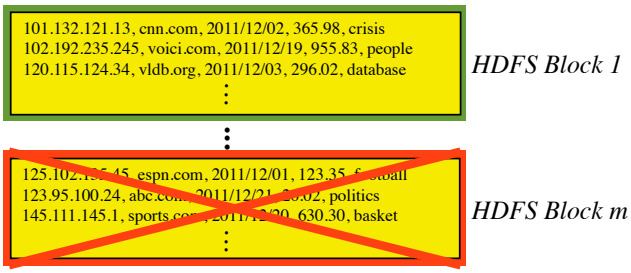
- File Level: *filters HDFS Blocks*



Indexing Levels

Filter Condition:
sourceIP == 120.115.124.34

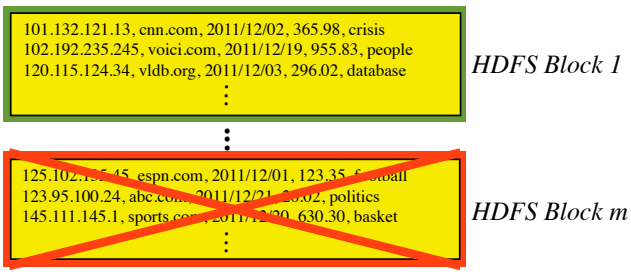
- File Level: *filters HDFS Blocks*



Indexing Levels

Filter Condition:
sourceIP == 120.115.124.34

- File Level: *filters HDFS Blocks*

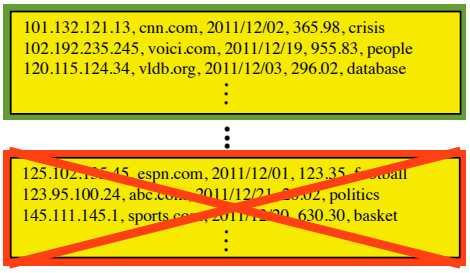


- Block Level: *filters records*

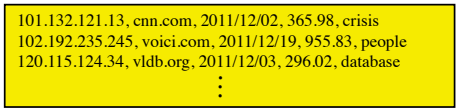
Indexing Levels

Filter Condition:
sourceIP == 120.115.124.34

- File Level: *filters HDFS Blocks*



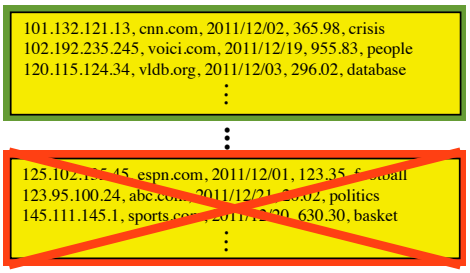
- Block Level: *filters records*



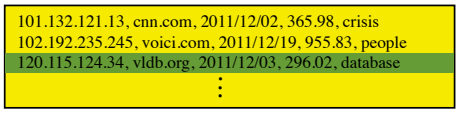
Indexing Levels

Filter Condition:
sourceIP == 120.115.124.34

- File Level: *filters HDFS Blocks*



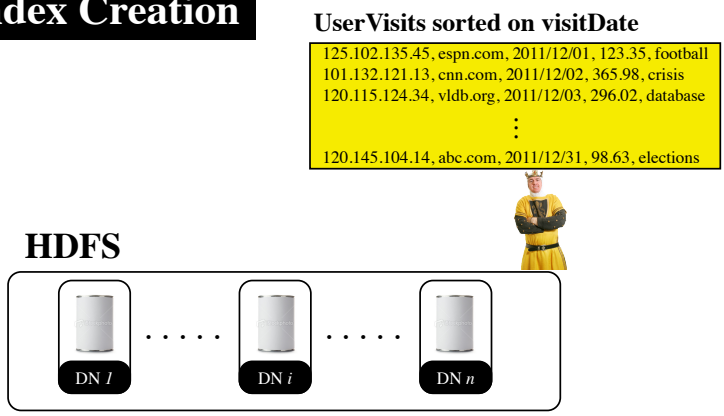
- Block Level: *filters records*



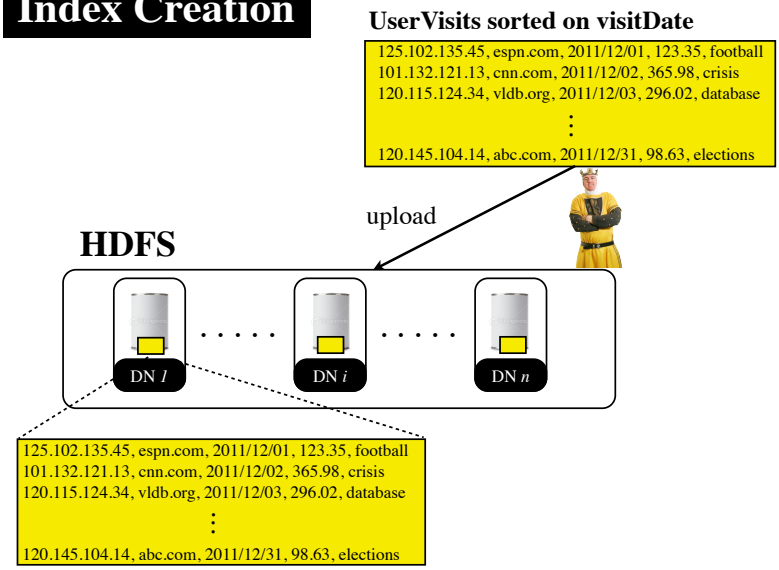
File-Level Indexing (Blocks Directory)

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

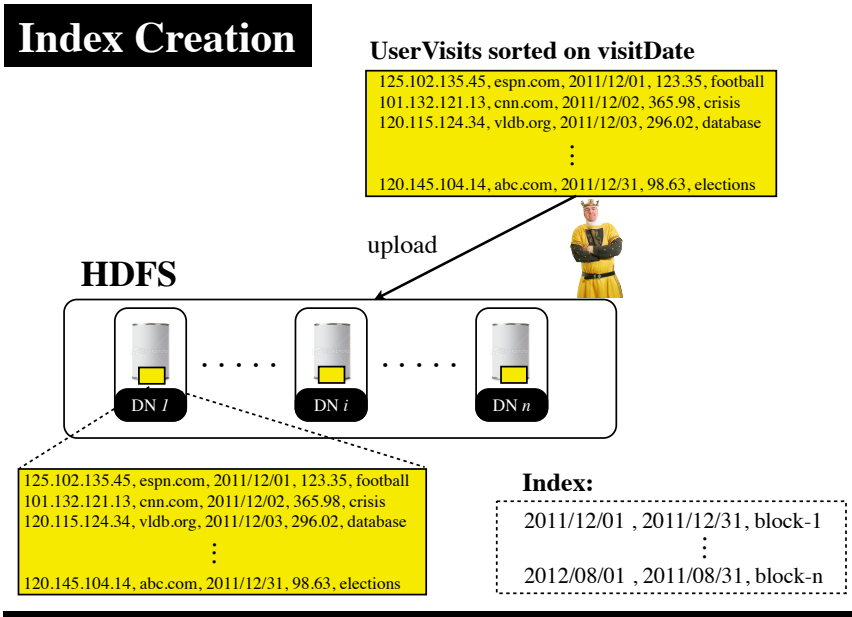
Index Creation



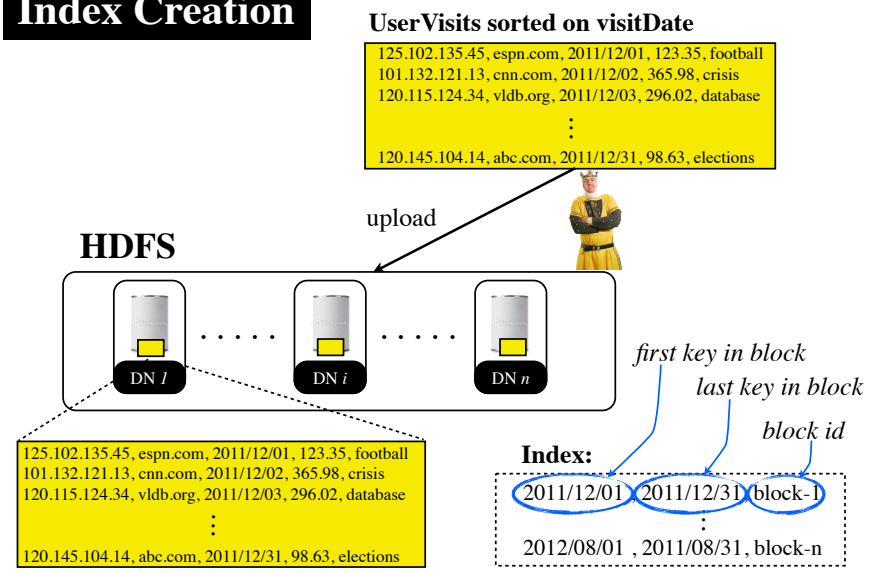
Index Creation



Index Creation

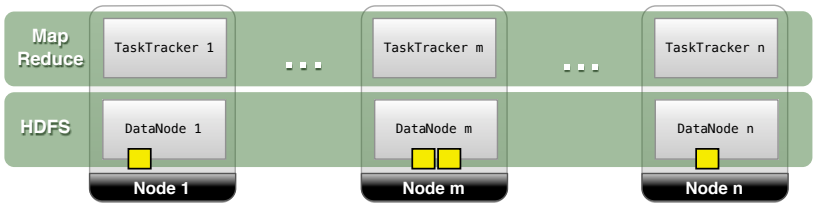


Index Creation



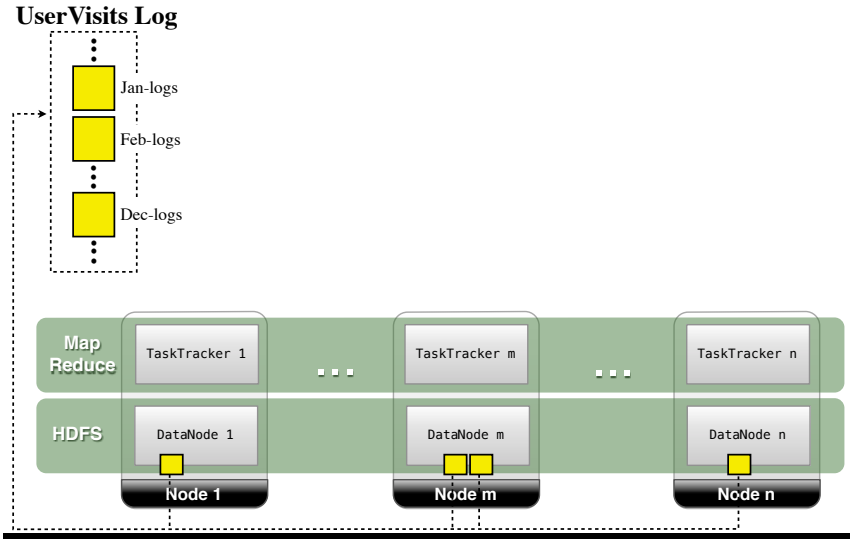
[D. Jiang et al.: The Performance of MapReduce: An In-Depth Study. PVLDB 2010] 128

Job Execution



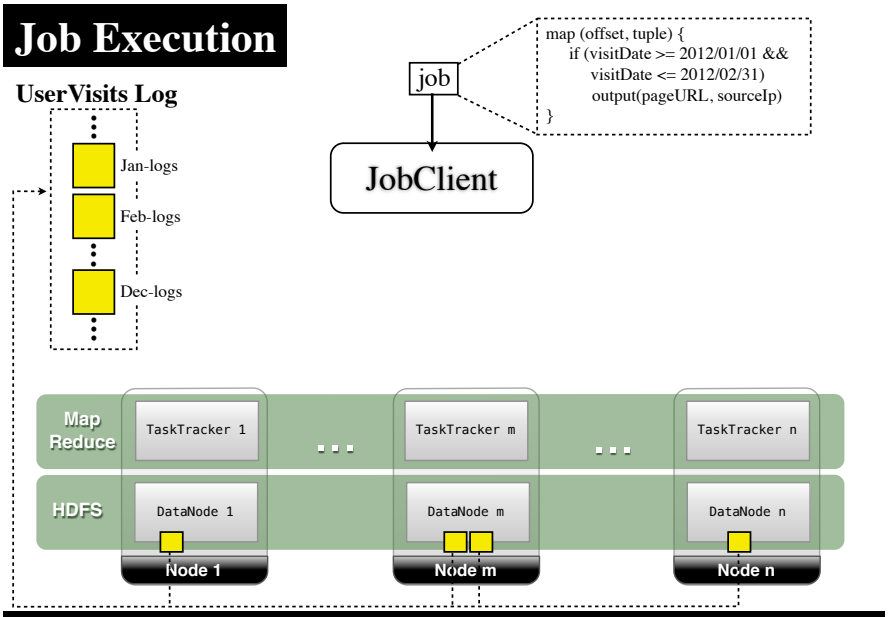
[D. Jiang et al.: The Performance of MapReduce: An In-Depth Study. PVLDB 2010] 129

Job Execution



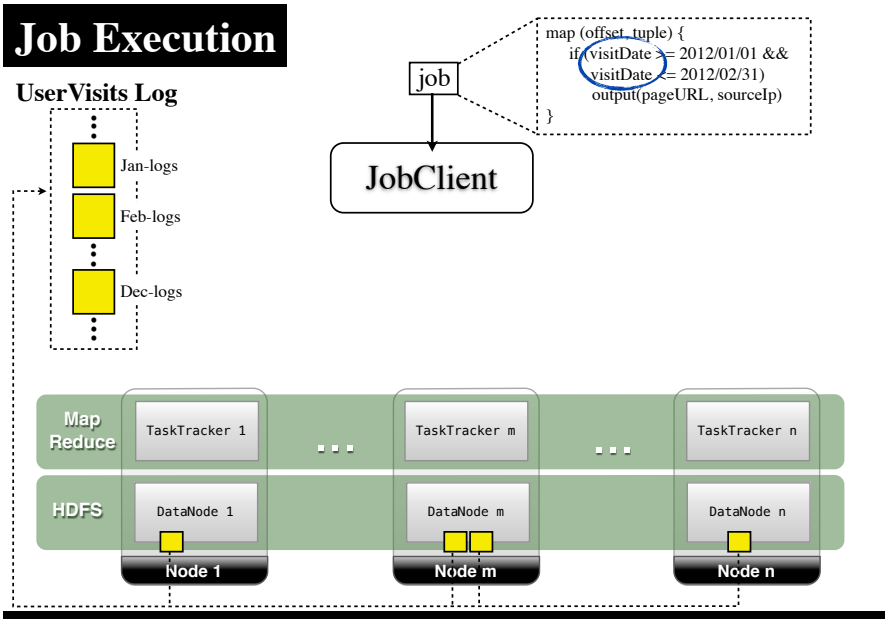
[D. Jiang et al.: The Performance of MapReduce: An In-Depth Study. PVLDB 2010] 129

Job Execution



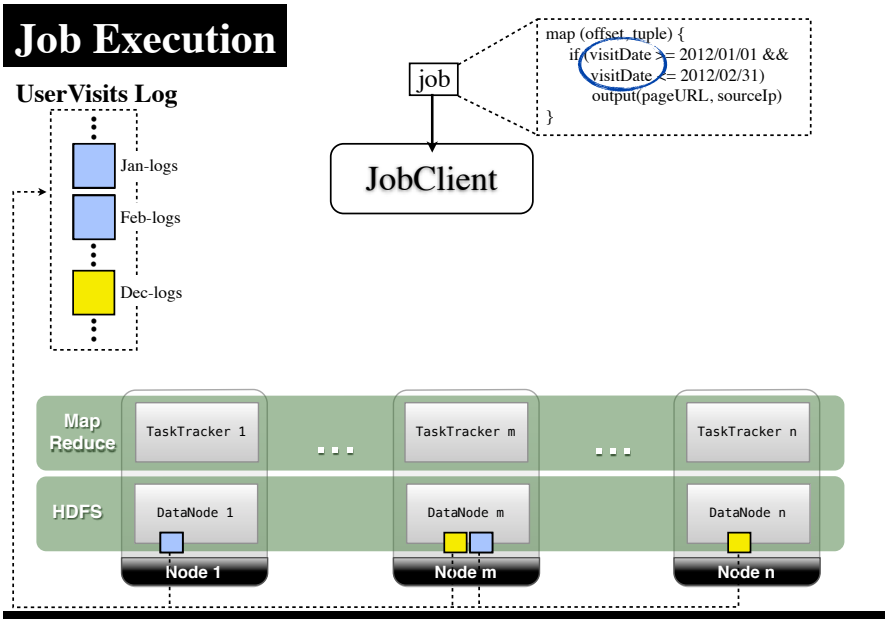
[D. Jiang et al.: The Performance of MapReduce: An In-Depth Study. PVLDB 2010] 129

Job Execution



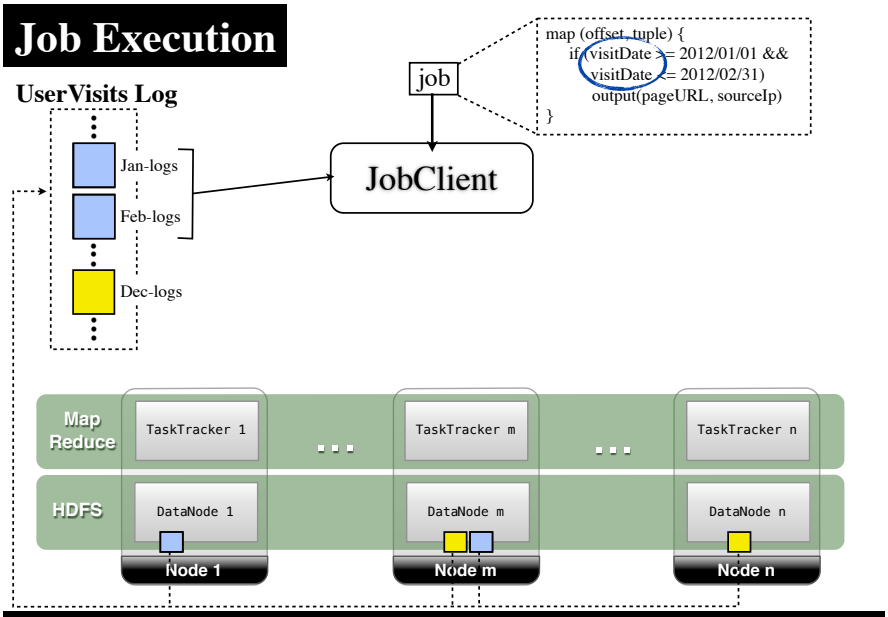
[D. Jiang et al.: The Performance of MapReduce: An In-Depth Study. PVLDB 2010] 129

Job Execution



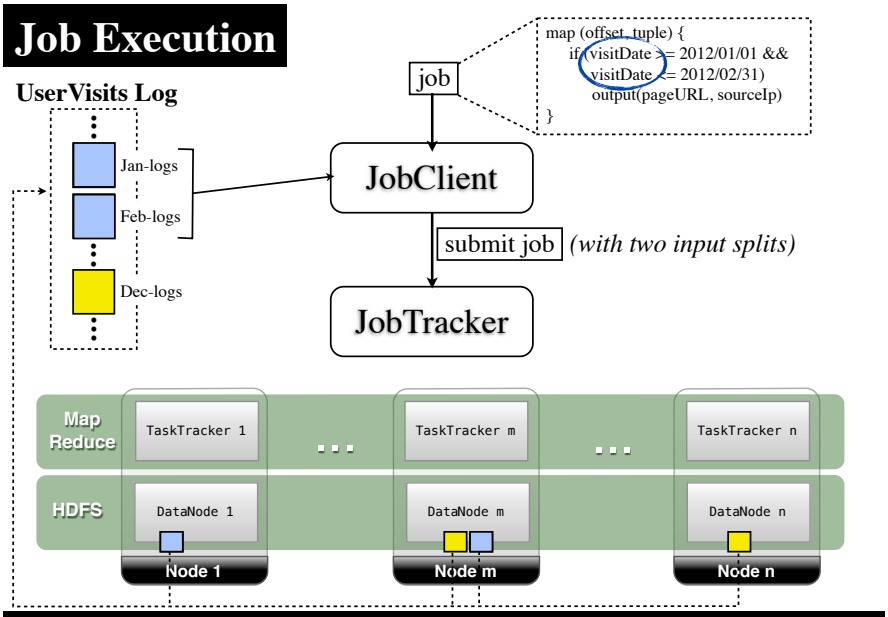
[D. Jiang et al.: The Performance of MapReduce: An In-Depth Study. PVLDB 2010] 129

Job Execution



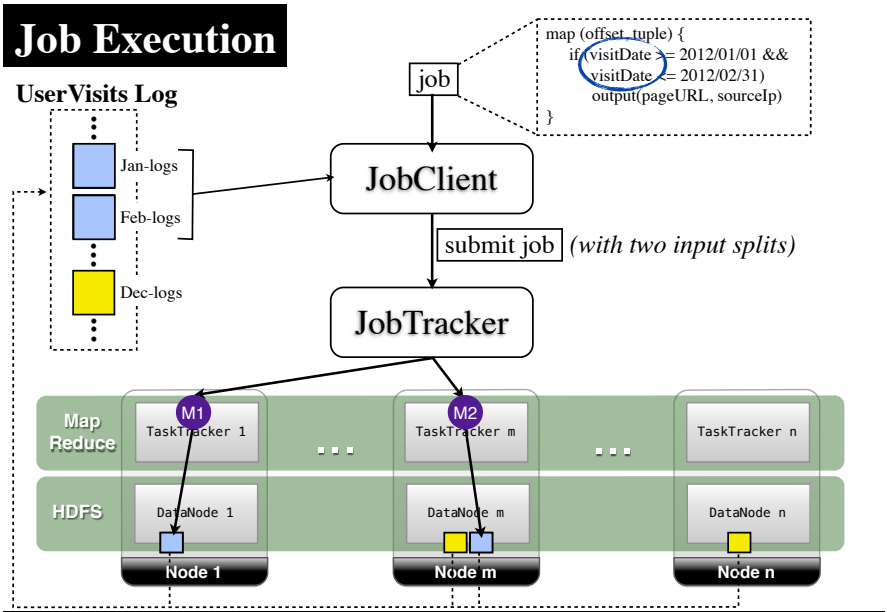
[D. Jiang et al.: The Performance of MapReduce: An In-Depth Study. PVLDB 2010] 129

Job Execution



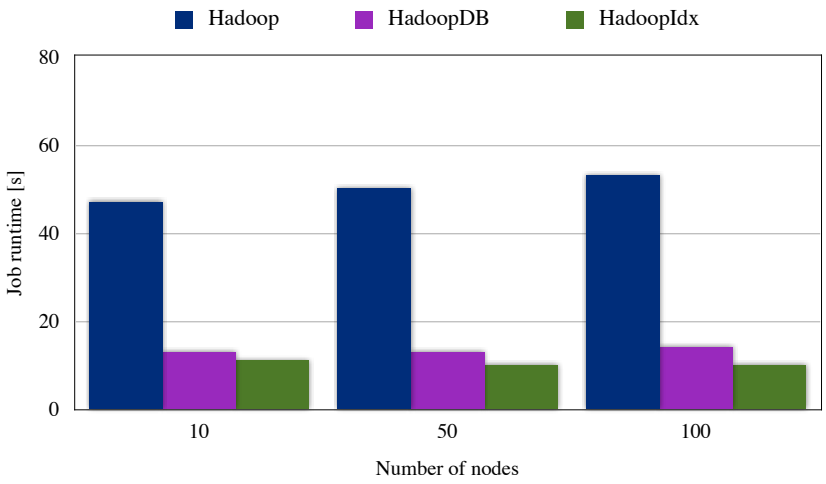
[D. Jiang et al.: The Performance of MapReduce: An In-Depth Study. PVLDB 2010] 129

Job Execution



[D. Jiang et al.: The Performance of MapReduce: An In-Depth Study. PVLDB 2010] 129

File-Level Indexing Results (Selection Task)



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

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

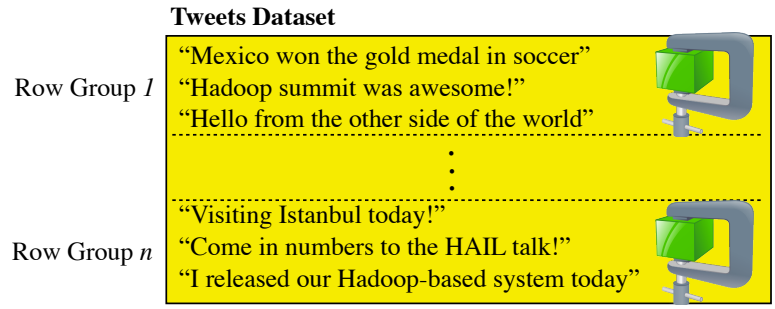
"Mexico won the gold medal in soccer"
"Hadoop summit was awesome!"
"Hello from the other side of the world"
⋮
"Visiting Istanbul today!"
"Come in numbers to the HAIL talk!"
"I released our Hadoop-based system today"

Index Creation

Tweets Dataset

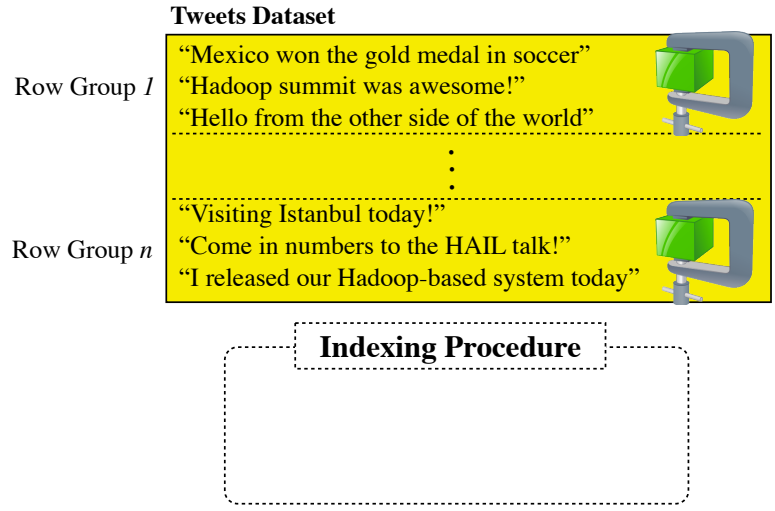
Row Group *1*
"Mexico won the gold medal in soccer"
"Hadoop summit was awesome!"
"Hello from the other side of the world"
⋮
Row Group *n*
"Visiting Istanbul today!"
"Come in numbers to the HAIL talk!"
"I released our Hadoop-based system today"

Index Creation



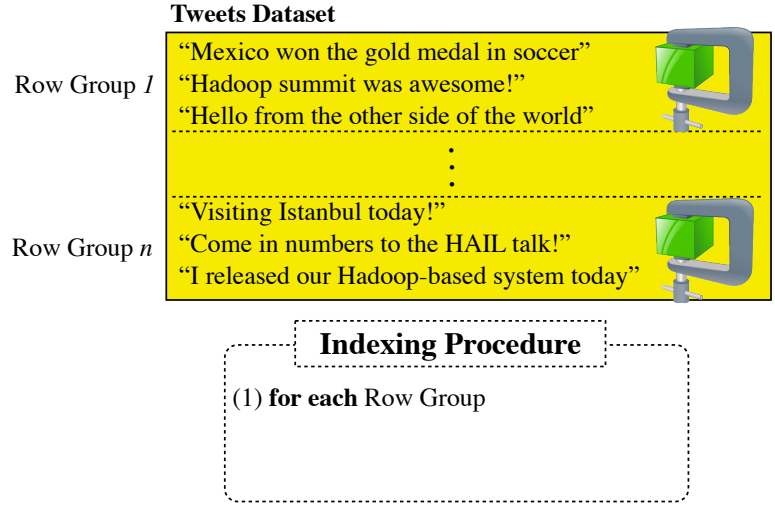
[J. Lin et al.: Full-Text Indexing for Optimizing Selection Operations in Large-Scale Data Analytics. MapReduce Workshop 2011]

Index Creation



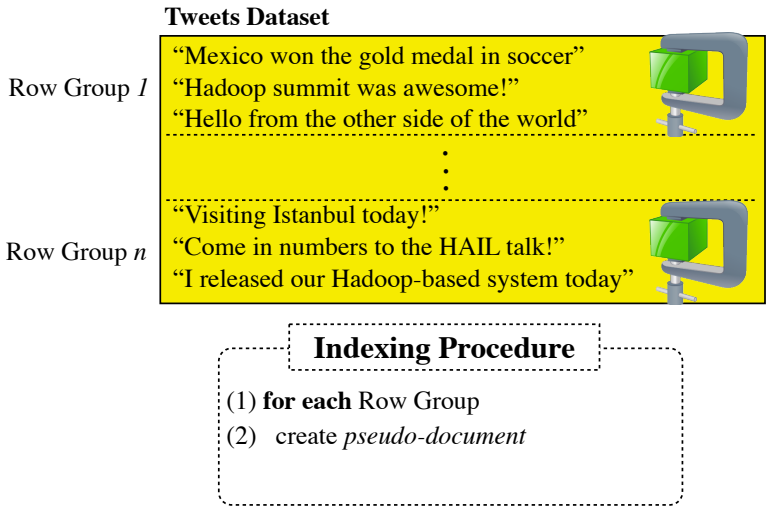
[J. Lin et al.: Full-Text Indexing for Optimizing Selection Operations in Large-Scale Data Analytics. MapReduce Workshop 2011]

Index Creation

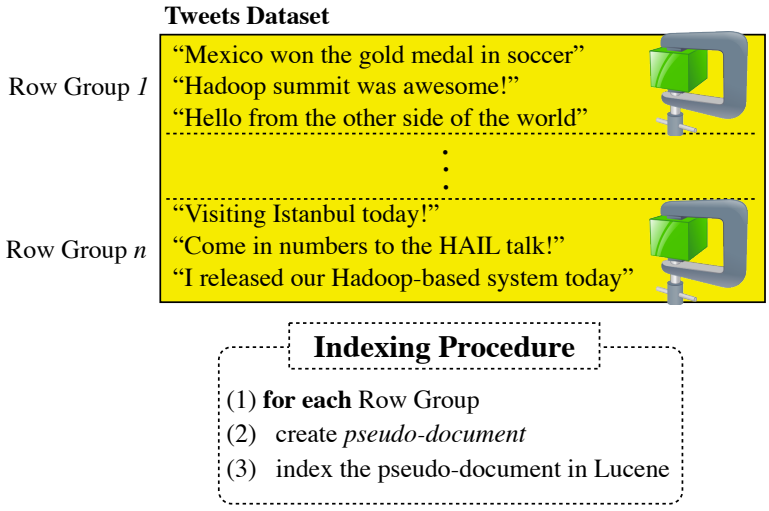


[J. Lin et al.: Full-Text Indexing for Optimizing Selection Operations in Large-Scale Data Analytics. MapReduce Workshop 2011]

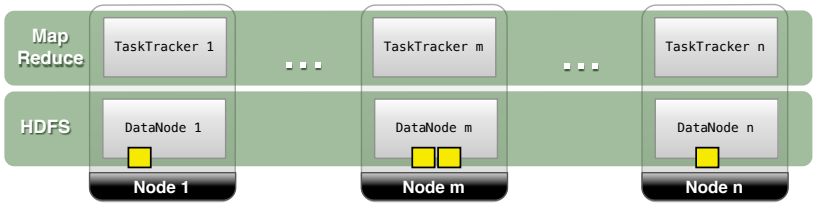
Index Creation



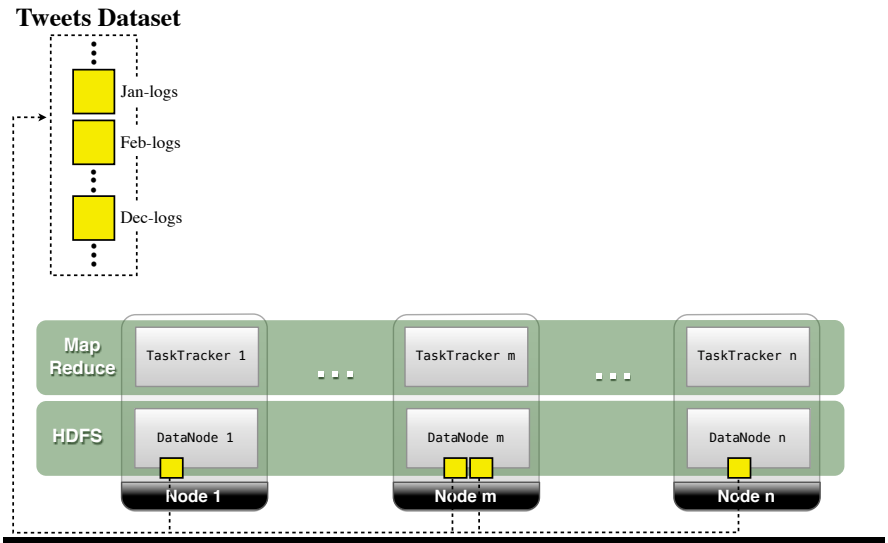
Index Creation



Job Execution

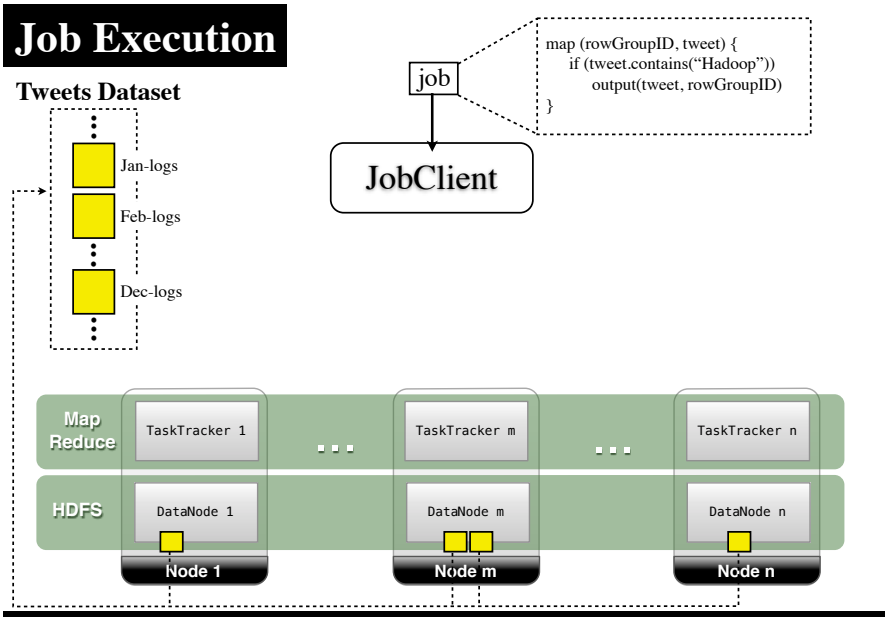


Job Execution



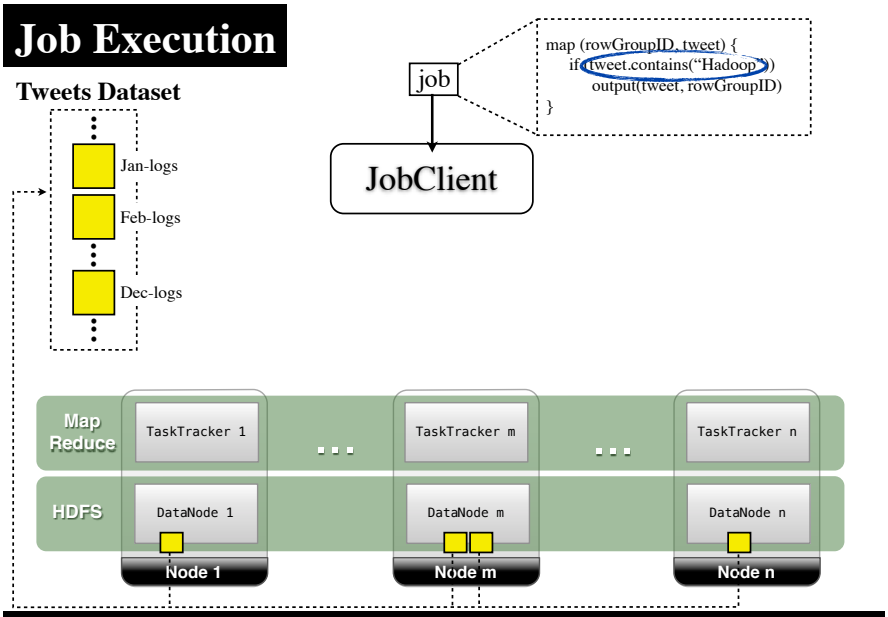
[J. Lin et al.: Full-Text Indexing for Optimizing Selection Operations in Large-Scale Data Analytics. MapReduce Workshop 2011]

Job Execution



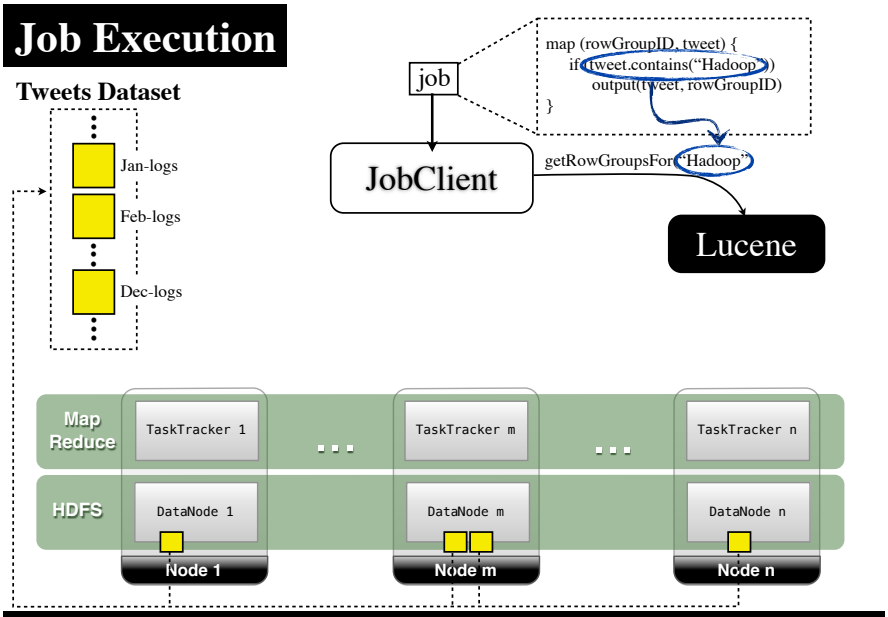
[J. Lin et al.: Full-Text Indexing for Optimizing Selection Operations in Large-Scale Data Analytics. MapReduce Workshop 2011]

Job Execution



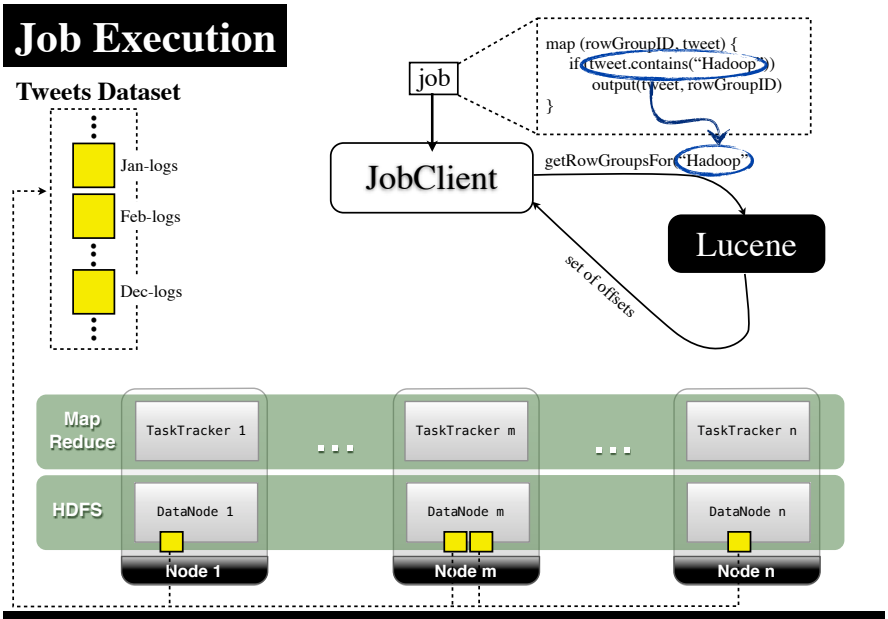
[J. Lin et al.: Full-Text Indexing for Optimizing Selection Operations in Large-Scale Data Analytics. MapReduce Workshop 2011]

Job Execution



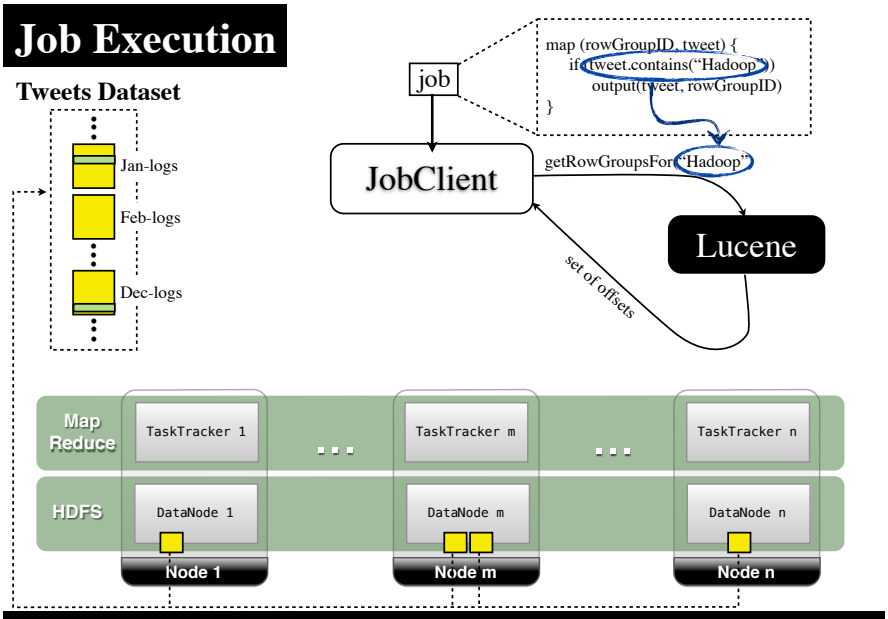
[J. Lin et al.: Full-Text Indexing for Optimizing Selection Operations in Large-Scale Data Analytics. MapReduce Workshop 2011]

Job Execution



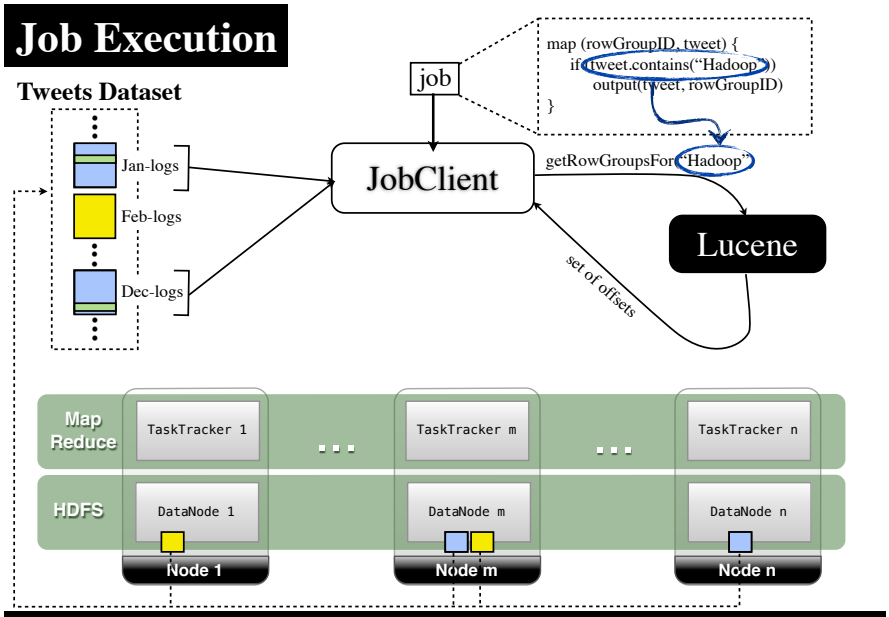
[J. Lin et al.: Full-Text Indexing for Optimizing Selection Operations in Large-Scale Data Analytics. MapReduce Workshop 2011]

Job Execution



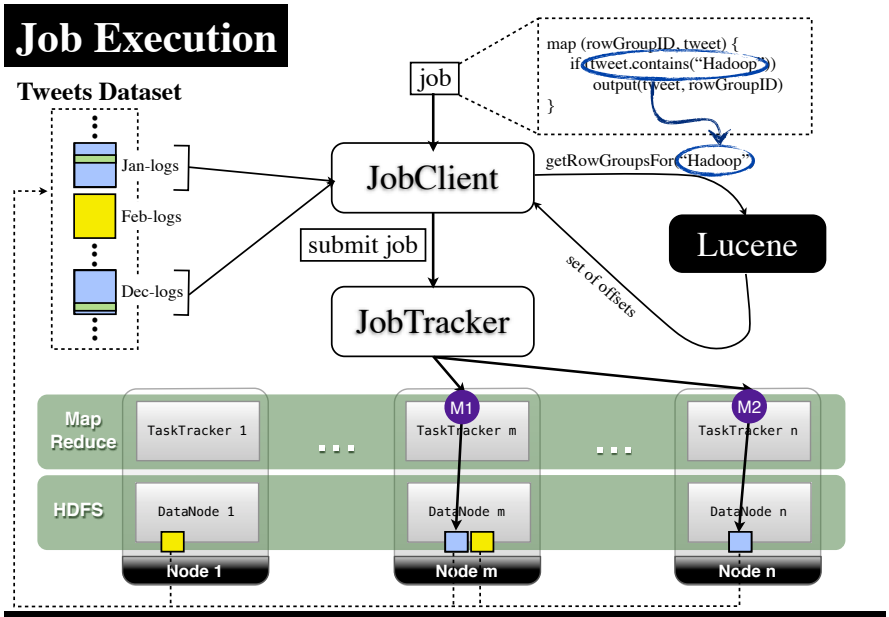
[J. Lin et al.: Full-Text Indexing for Optimizing Selection Operations in Large-Scale Data Analytics. MapReduce Workshop 2011]

Job Execution



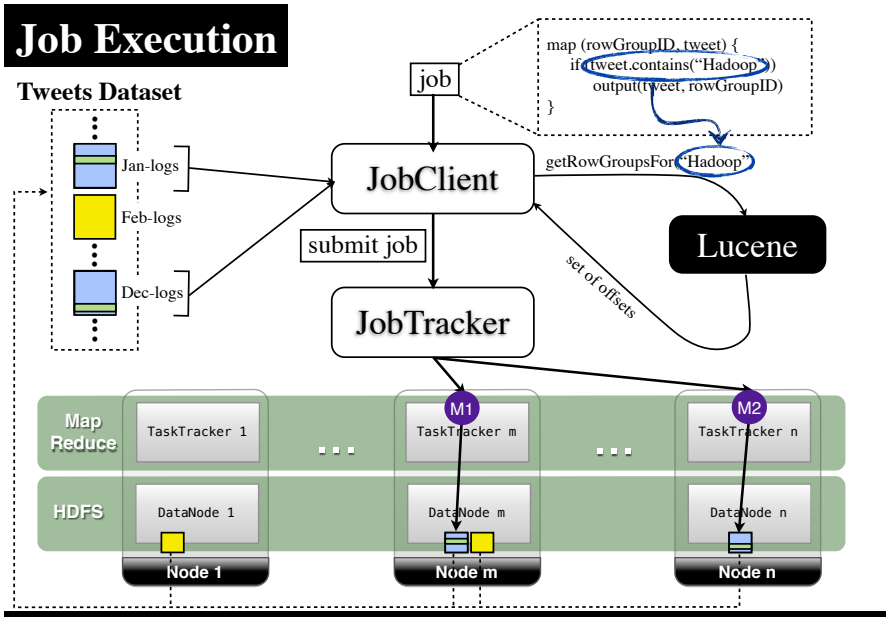
[J. Lin et al.: Full-Text Indexing for Optimizing Selection Operations in Large-Scale Data Analytics. MapReduce Workshop 2011]

Job Execution



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

Job Execution



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

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}

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

Full-Text Indexing Results

Setup

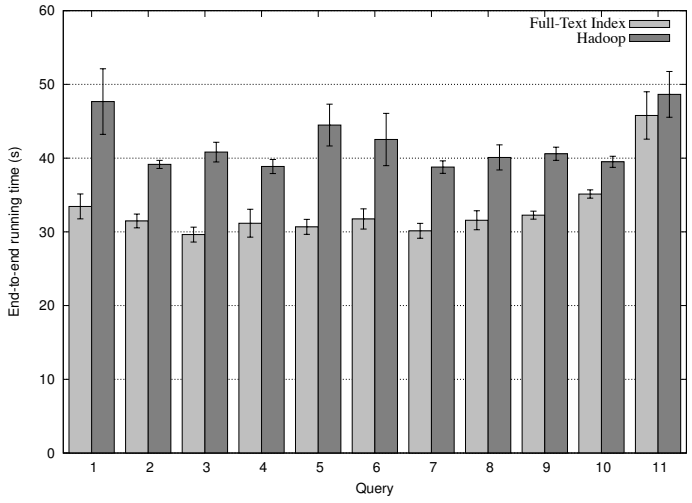
Tweets Dataset: 69.2 million tweets
Dataset Size: 6.07GB
#Row Groups: 39,767
Avg #Records per Row Group: 1,740

>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}

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

Full-Text Indexing Results



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

Indexing in MapReduce

2009	2010
HadoopDB	File Level
Still a database	
	Global Sorting

Indexing in MapReduce

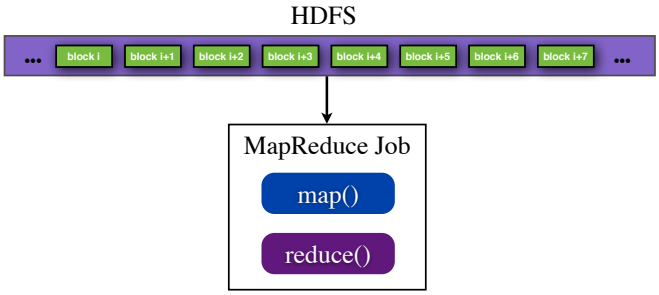
2009	2010	2011
HadoopDB	File Level	Full Text
Still a database		
	Global Sorting	
		Only for high selectivity

Trojan Index

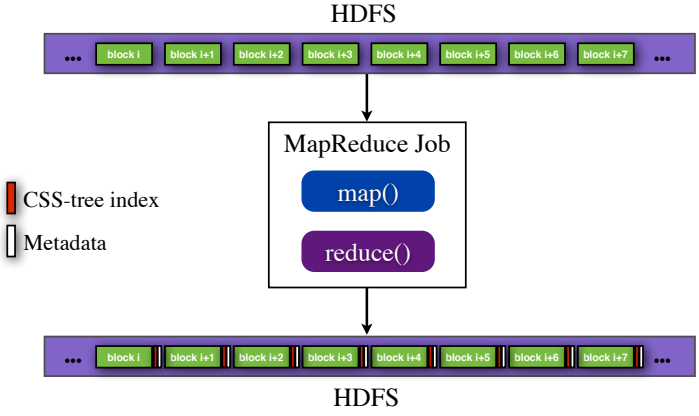
[J. Dittrich, J. Quiané, A. Jindal, Y. Kargin, V. Setty, J. Schad: Hadoop++: Making a Yellow Elephant Run Like a Cheetah (Without It Even Noticing). PVLDB 2010]

Index Creation

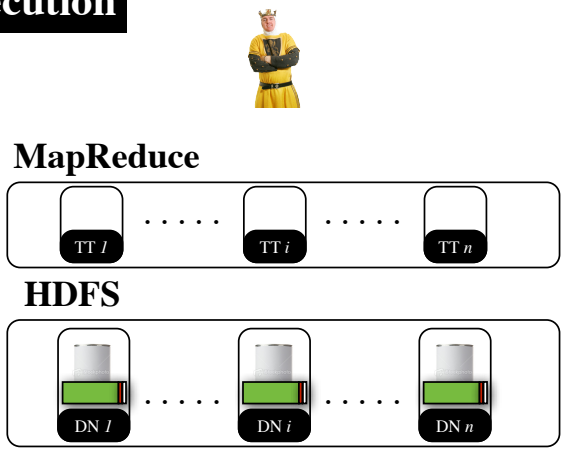
Index Creation



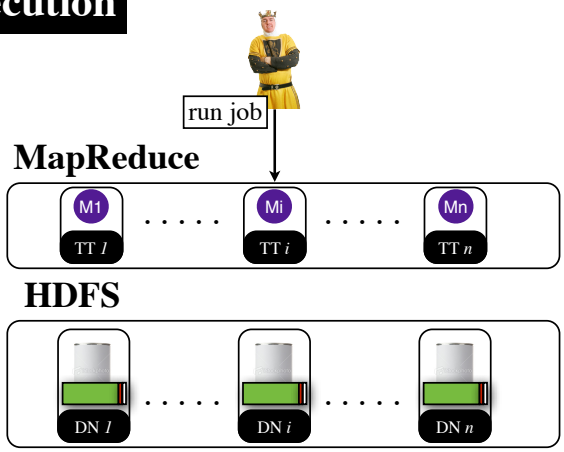
Index Creation



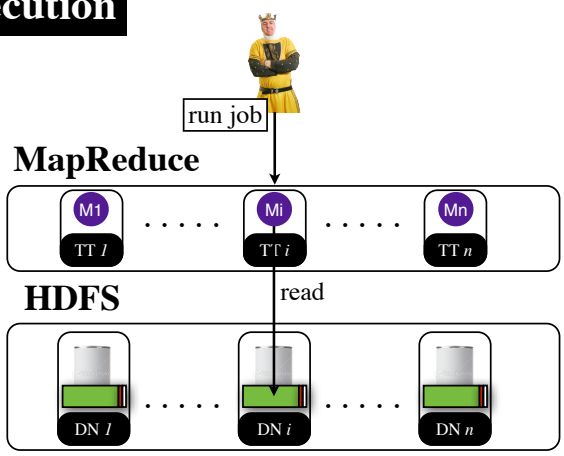
Job Execution



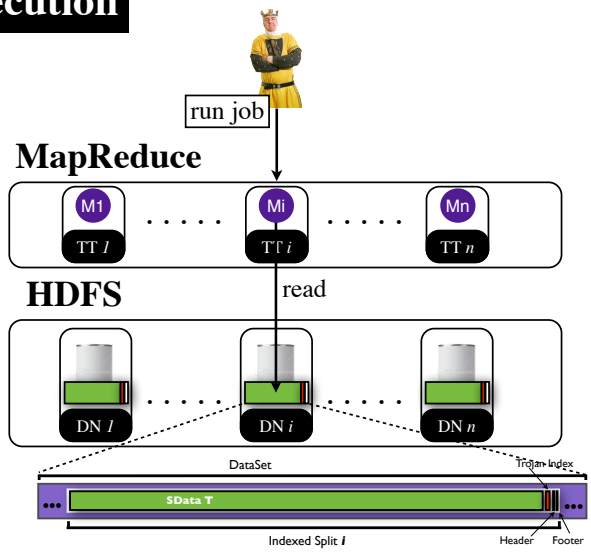
Job Execution



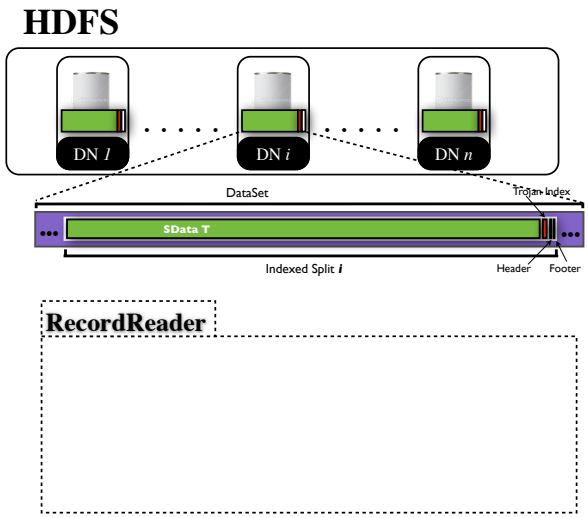
Job Execution



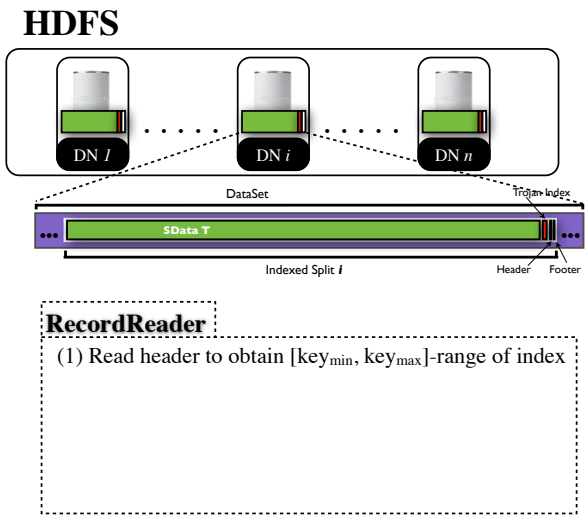
Job Execution



Job Execution

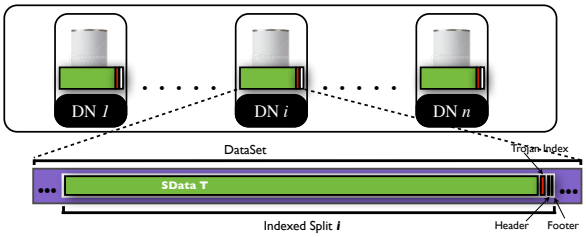


Job Execution



Job Execution

HDFS

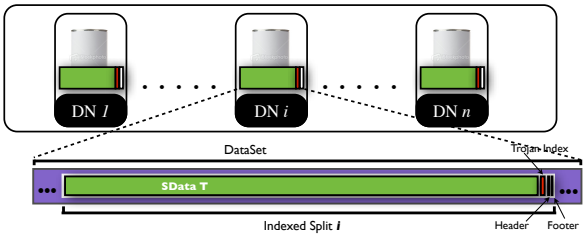


RecordReader

(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

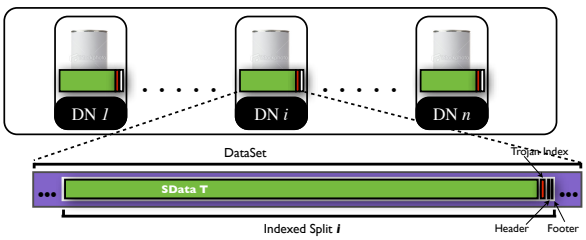


RecordReader

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

Job Execution

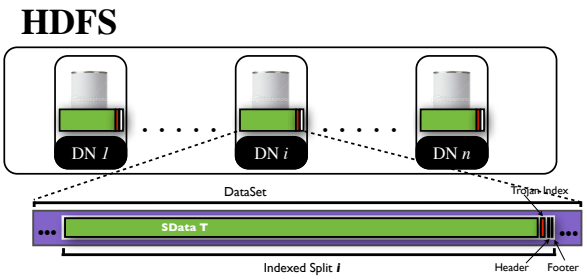
HDFS



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()

Job Execution

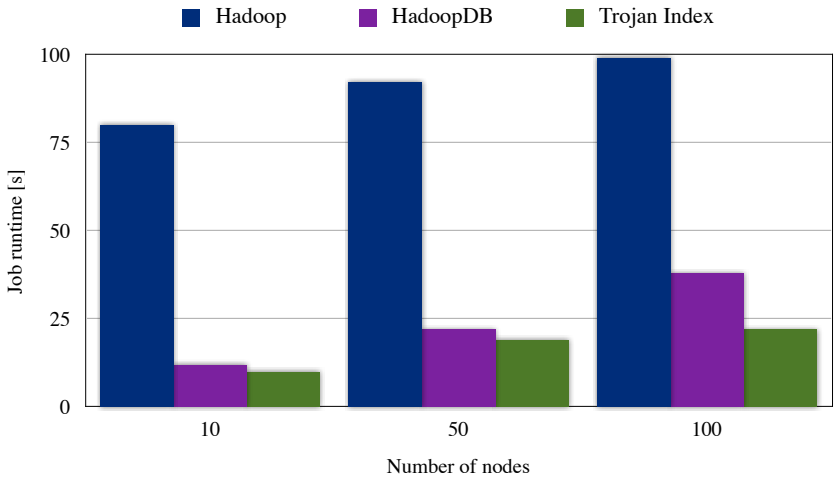


RecordReader

(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

[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]

Block-Level Indexing Results (Selection Task)



[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]

Indexing in MapReduce

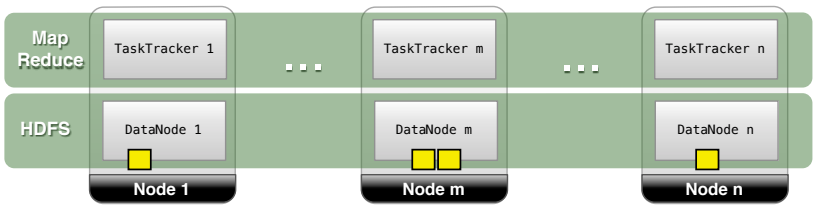
2009	2010	2011
HadoopDB	File Level	Full Text
Still a database		
	Global Sorting	
		Only for high selectivity

Indexing in MapReduce

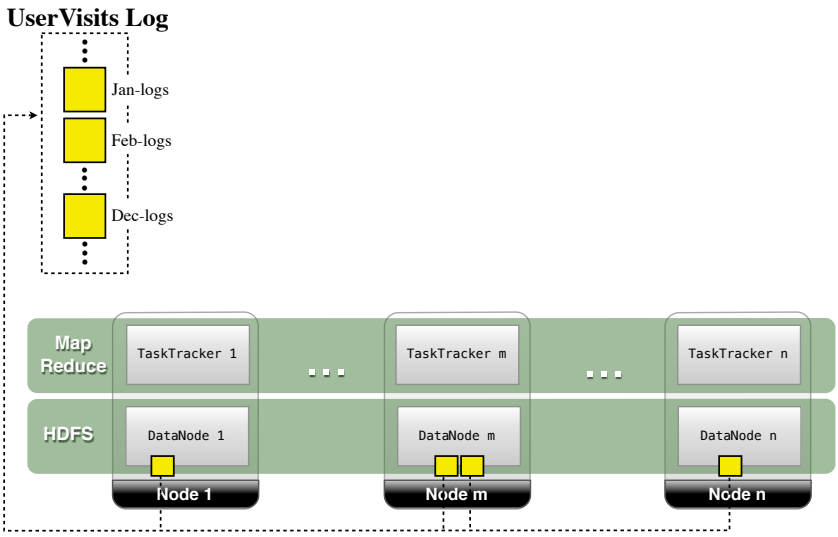
2009	2010	2011	2010
HadoopDB	File Level	Full Text	Trojan
Still a database			
	Global Sorting		
		Only for high selectivity	

Can We Exploit them All Together?

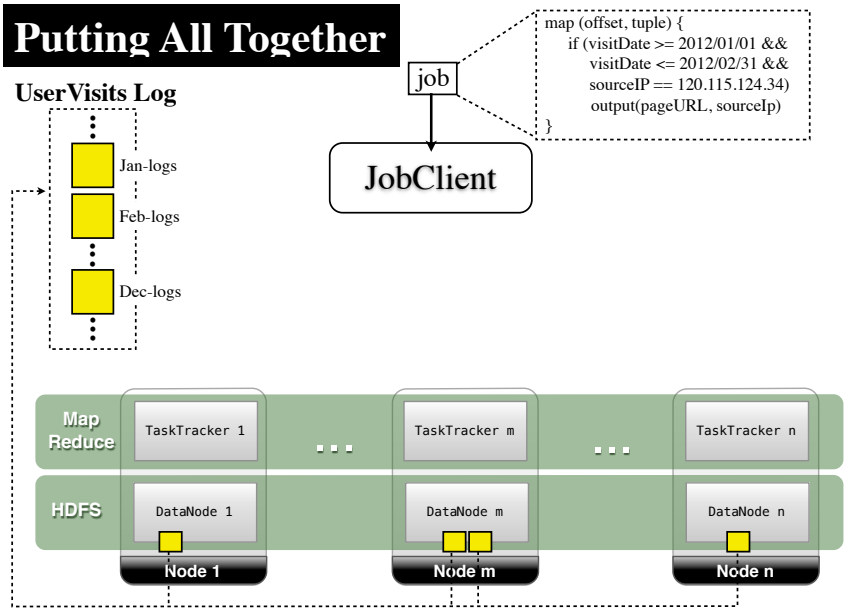
Putting All Together



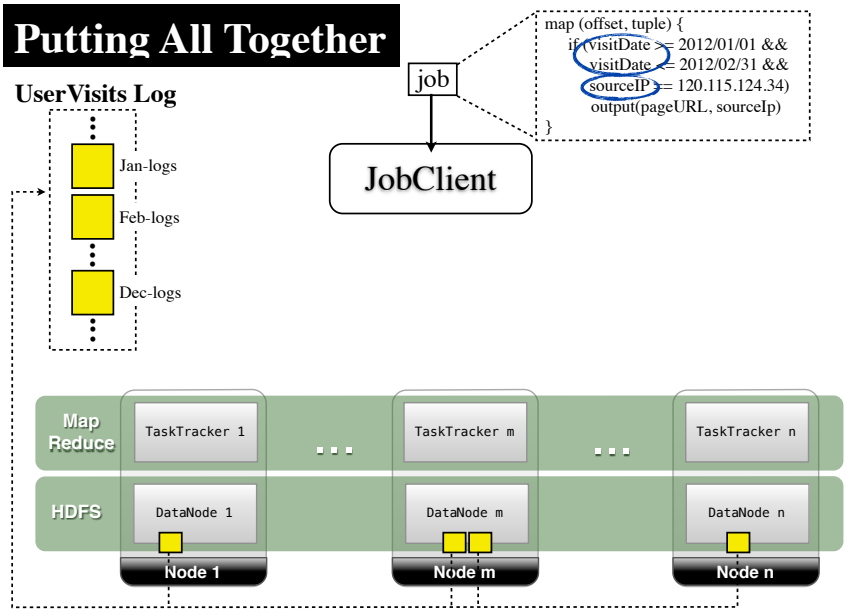
Putting All Together



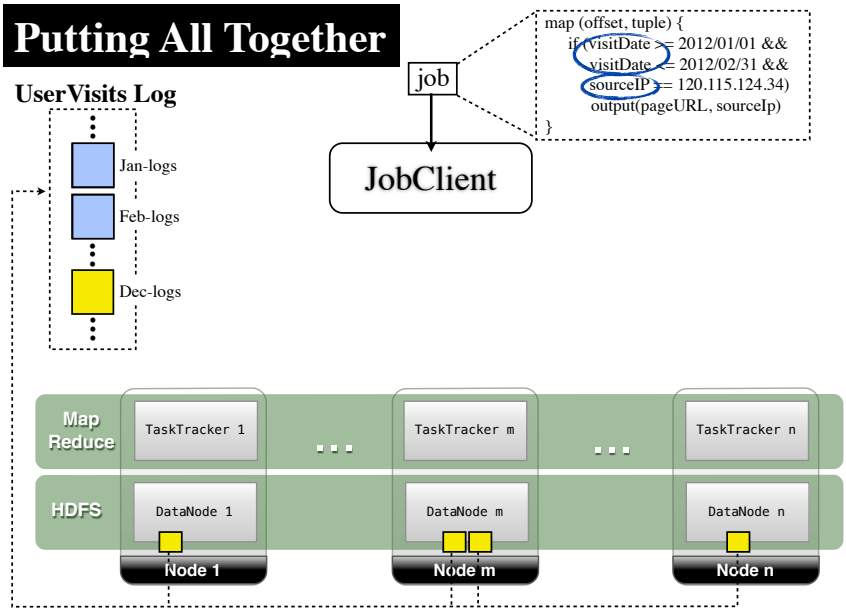
Putting All Together



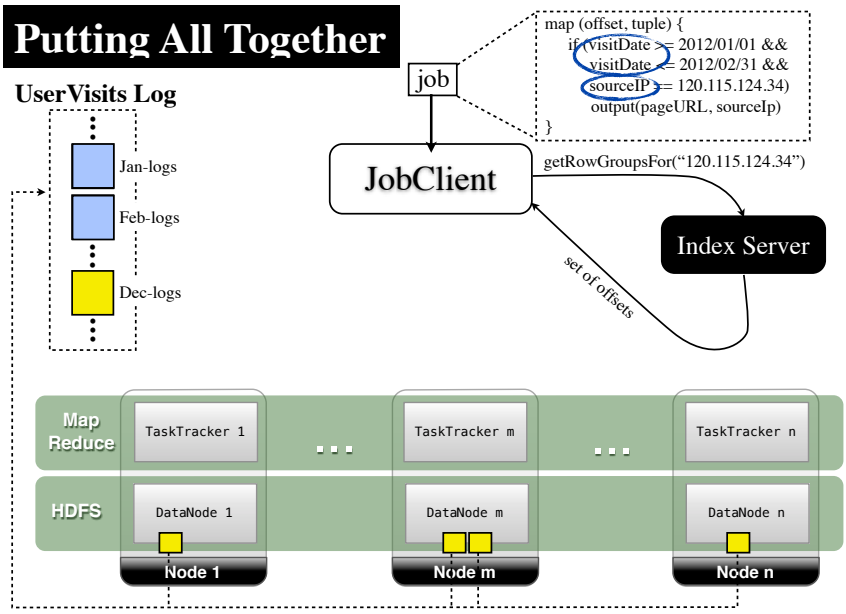
Putting All Together



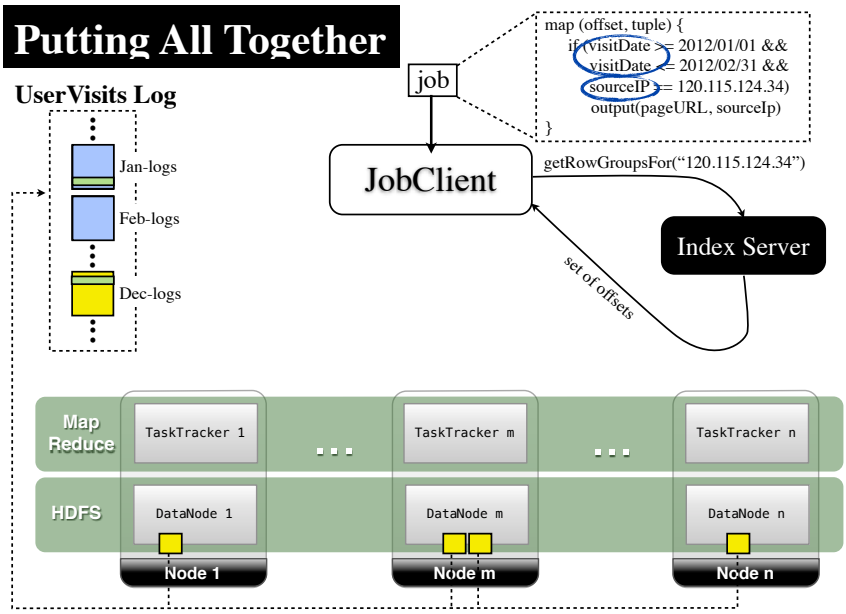
Putting All Together



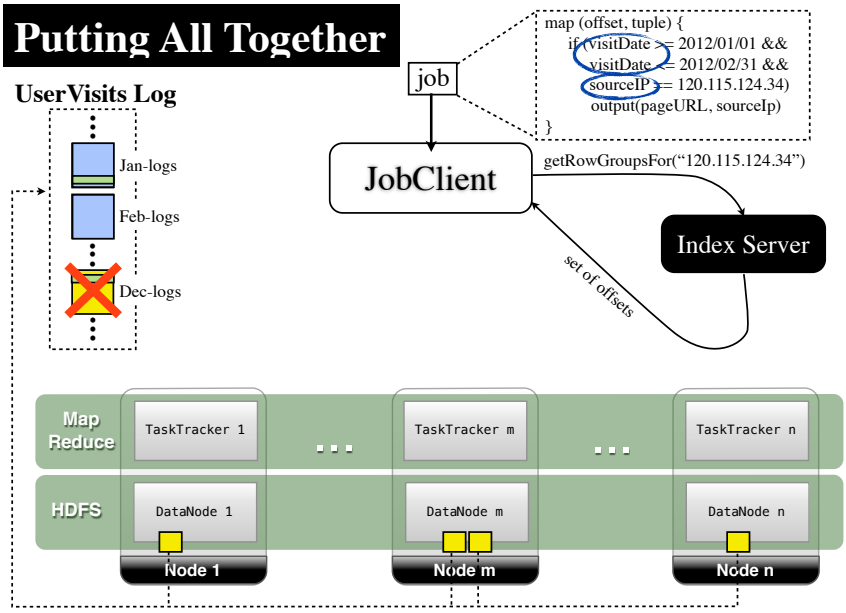
Putting All Together



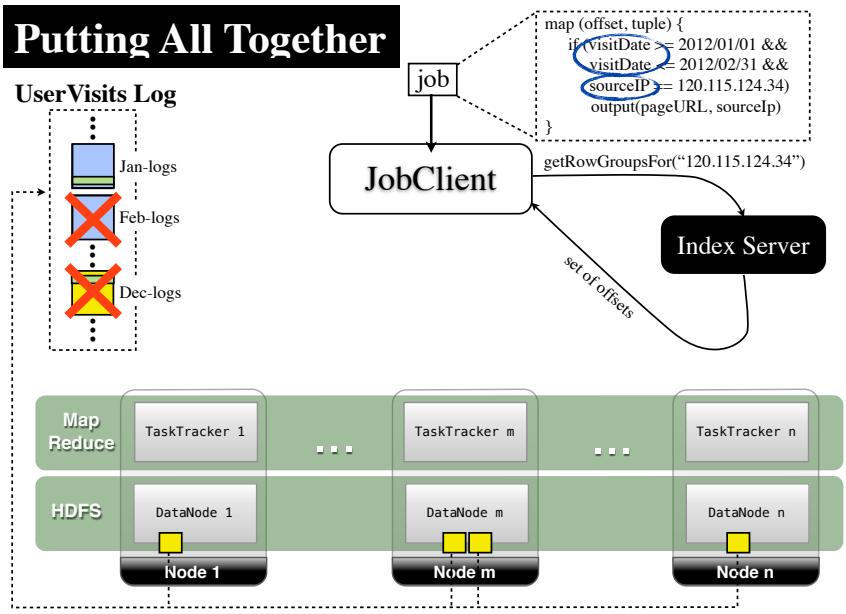
Putting All Together



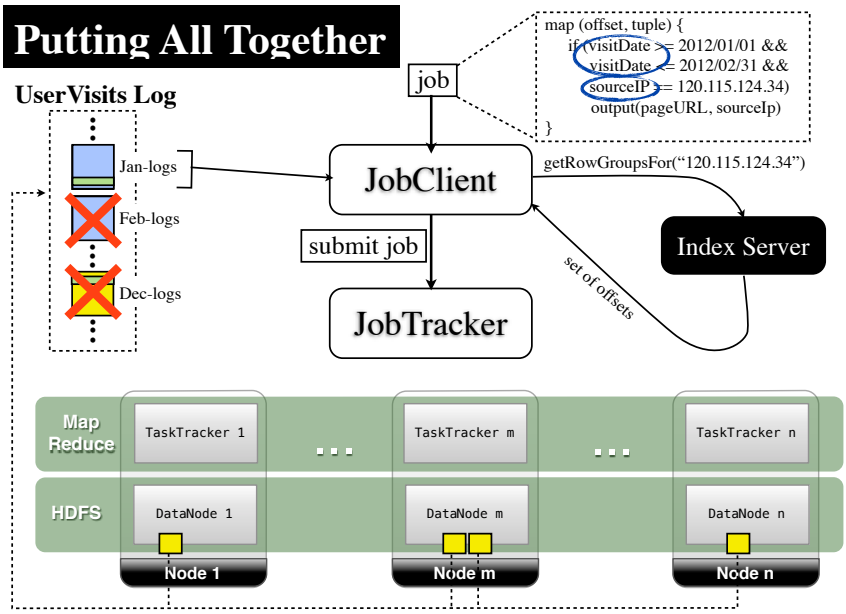
Putting All Together



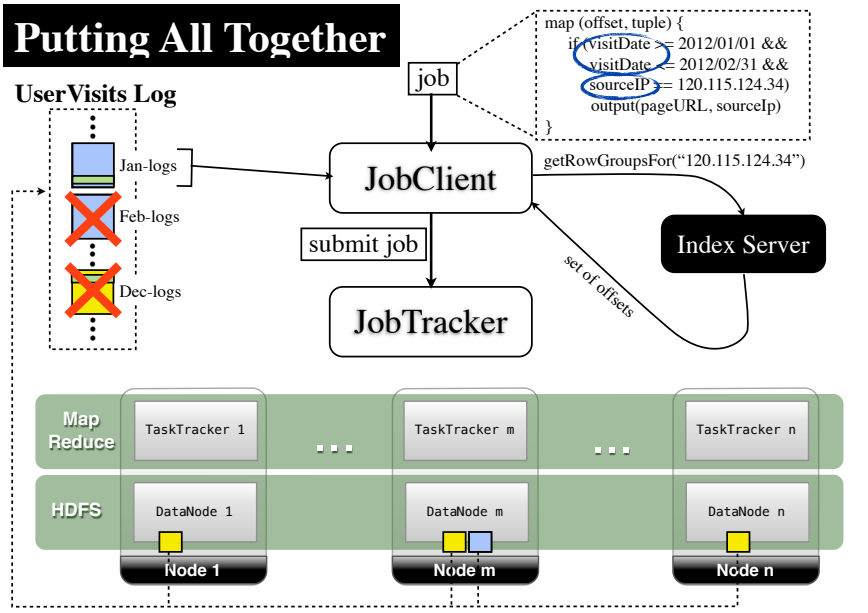
Putting All Together



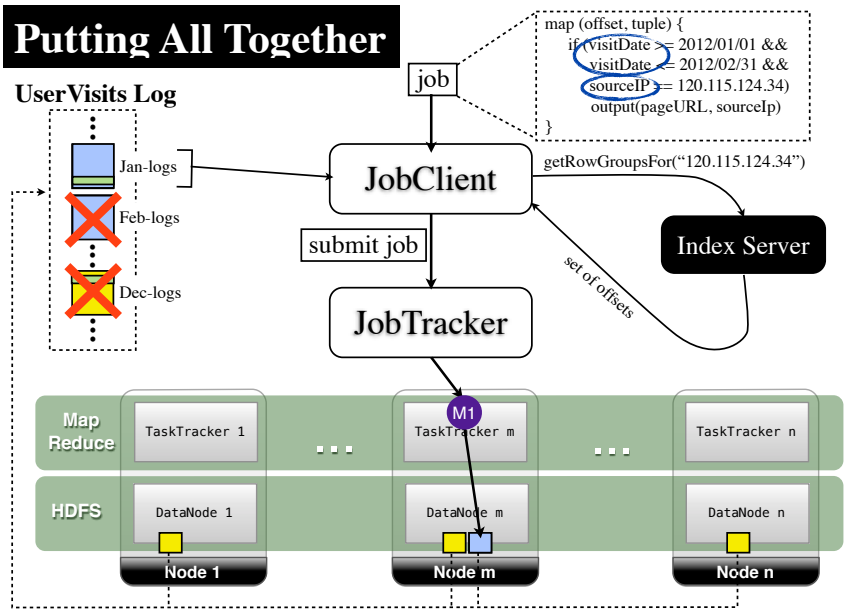
Putting All Together



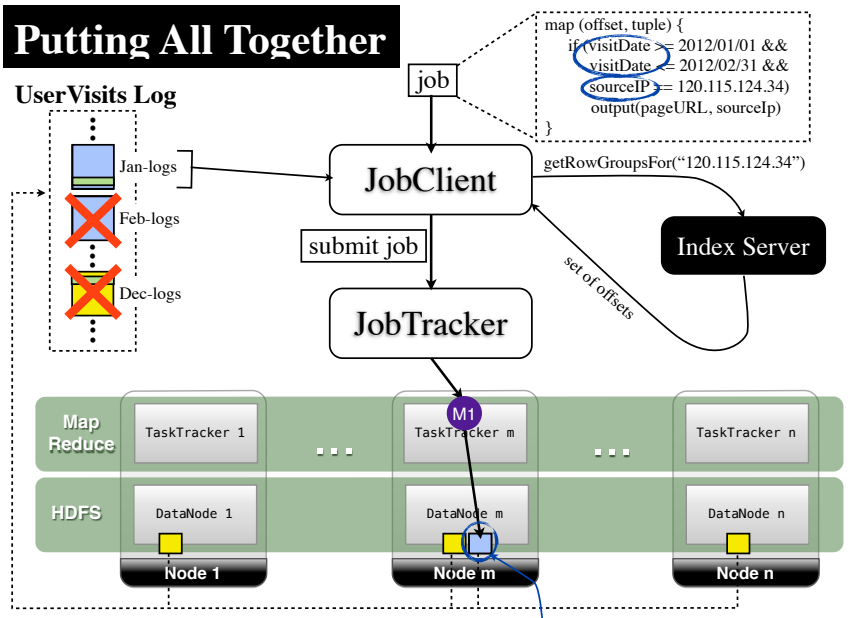
Putting All Together



Putting All Together



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 Layouts[1]

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

Indexing in MapReduce

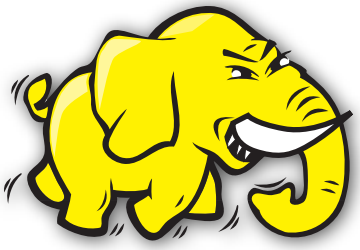
2009	2010	2011	2010	2012
HadoopDB	File Level	Full Text	Trojan	HAIL
Still a database				
	Global Sorting			
		Only for high selectivity		

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	

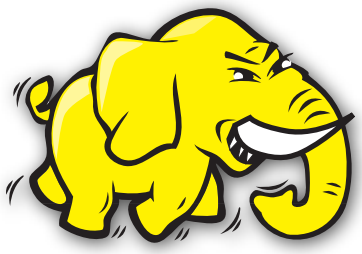
Indexing in MapReduce

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)

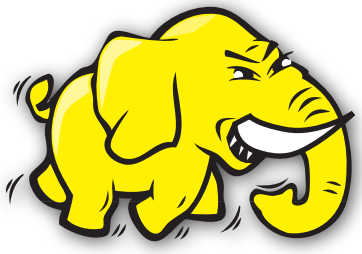
[J. Dittrich, J. Quiané, S. Richter, S. Schuh, A. Jindal, J. Schad: Only Aggressive Elephants are Fast Elephants. PVLDB 2012]



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++

[J. Dittrich, J. Quiané, S. Richter, S. Schuh, A. Jindal, J. Schad: Only Aggressive Elephants are Fast Elephants. PVLDB 2012] 150

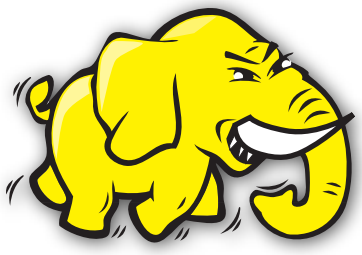


TALK:
Only Aggressive Elephants are Fast Elephants
Wednesday August 29th
11:30 a.m. at the Convention Lower Hall 2
(Research Session 13: MapReduce II)

Invisible Index Creation Times:
up to 7.3 times faster than Hadoop++

Fast Data Upload:
up to 1.6 times faster than Hadoop

[J. Dittrich, J. Quiané, S. Richter, S. Schuh, A. Jindal, J. Schad: Only Aggressive Elephants are Fast Elephants. PVLDB 2012] 150



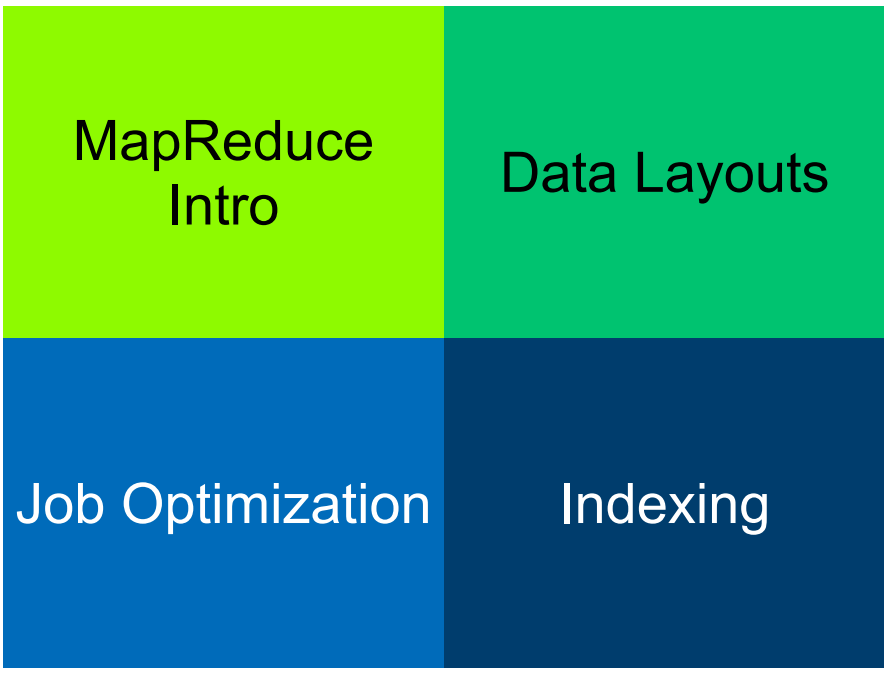
TALK:
Only Aggressive Elephants are Fast Elephants
Wednesday August 29th
11:30 a.m. at the Convention Lower Hall 2
(Research Session 13: MapReduce II)

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++

[J. Dittrich, J. Quiané, S. Richter, S. Schuh, A. Jindal, J. Schad: Only Aggressive Elephants are Fast Elephants. PVLDB 2012] 150



Efficient Big Data Processing in Hadoop MapReduce

Jens Dittrich

Jorge-Arnulfo Quiané-Ruiz

SAARLAND UNIVERSITY
COMPUTER SCIENCE