Работа с большими данными

SELEZNEV ARTEM
HEAD OF DATA SCIENCE @ SBER

tg: @SeleznevArtem

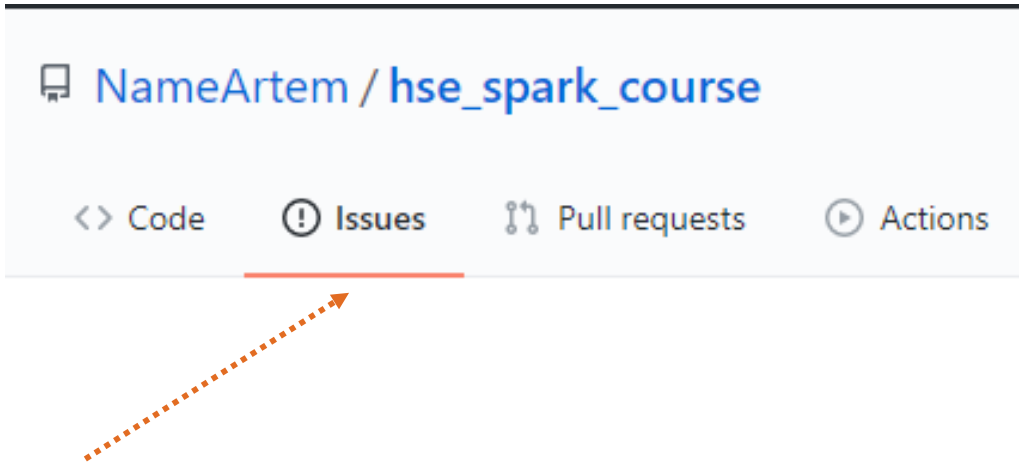
/NameArtem

/seleznev-artem

/seleznev.artem.info

https://github.com/NameArtem/hse_spark_course

https://github.com/NameArtem/hse_spark_course



https://github.com/NameArtem/hse_spark_course

https://cutt.ly/2lSpyZs

О КУРСЕ

№	Тема занятия
1	MapReduce. Введение в распределенные вычисления
2	HDFS. Apache Spark (RDD) (+ FuncProg на Python)
3	Spark SQL. Анализ больших данных
4	Подробнее о модели вычислений Spark. Знакомство со Scala
5	Spark ML
6	Рекомендательные системы на Spark
7	Spark Structure Streaming (+ интеграция со Spark ML)
8	Модели в прод. Управленеи кластеровм

ИНСТРУМЕНТЫ

Python

Linux

Git

Hadoop

Spark

ИНСТРУМЕНТЫ

Python

Linux

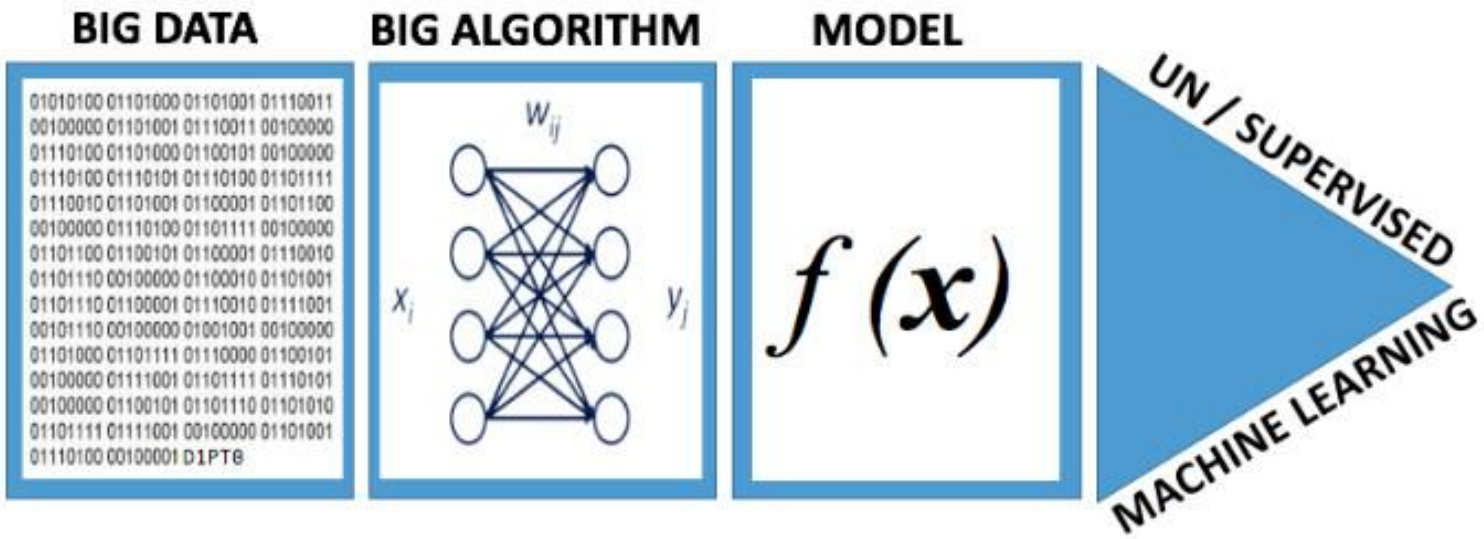
Git

Hadoop

Spark

BIG DATA?

ИЗ БОЛЬШОГО В МАЛОЕ



ИЗ БОЛЬШОГО В МАЛОЕ

Агрегат: data, user, goods_id

Детализация заказа в магазине?

Портфель акций одного инвестора?

Все заказы по всем магазинам?

Все транзакции по всем акциям?

Детализация заказа в магазине?

Портфель акций одного инвестора?

Все заказы по всем магазинам?

Все транзакции по всем акциям?

Детализация заказа в магазине?

Портфель акций одного инвестора?

Все заказы по всем магазинам?

Все транзакции по всем акциям?

Агрегат:
shop_id, cust_id,

Агрегат:
date, stock_id,

DATA БРОСАЕТ ВЫЗОВ

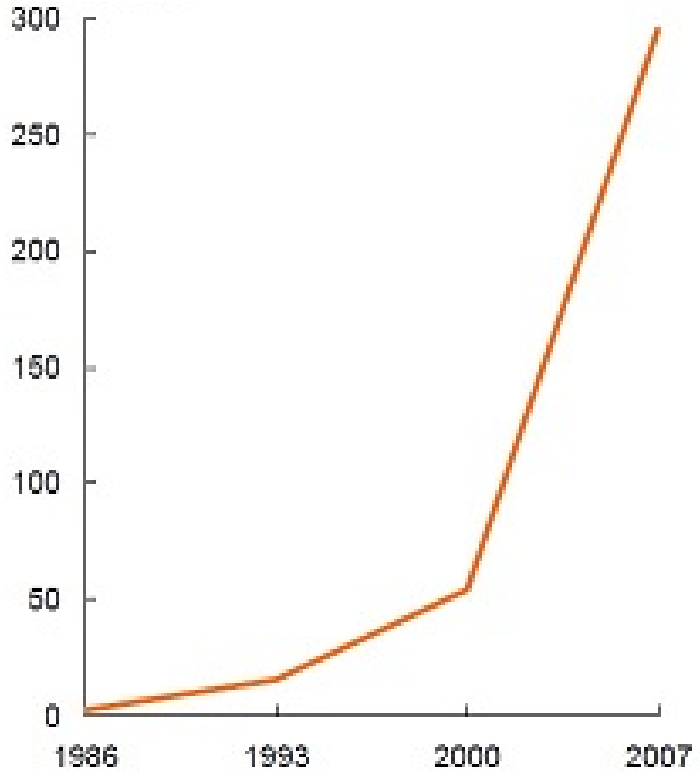
Данные создаются
очень быстро

Данные из разных
источников и в
разных форматах

DATA БРОСАЕТ ВЫЗОВ

Данные создаются очень быстро

Данные из разных источников и в разных форматах



DATA БРОСАЕТ ВЫЗОВ

Данные создаются очень быстро

Данные из разных источников и в разных форматах

3V

VOLUME

VARIETY

VELOCITY

3V

VOLUME

VARIETY

VELOCITY

$

3V

VOLUME

VARIETY

VELOCITY

3V

VOLUME

VARIETY

VELOCITY


```
[mpm_winnt:notice] [pid 5776:tid 740] AH00456: Apache Lounge VC15 Server built:
[core:notice] [pid 5776:tid 740] AH00094: Command line: 'C:\\Server\\bin\\Apach
[mpm_winnt:notice] [pid 5776:tid 740] AH00418: Parent: Created child process 87
[mpm_winnt:notice] [pid 8752:tid 712] AH00354: Child: Starting 64 worker thread
[mpm_winnt:notice] [pid 5776:tid 740] AH00422: Parent: Received shutdown signal
[mpm_winnt:notice] [pid 8752:tid 712] AH00364: Child: All worker threads have e
[mpm_winnt:notice] [pid 5776:tid 740] AH00430: Parent: Child process 8752 exite
[mpm_winnt:notice] [pid 3584:tid 740] AH00455: Apache/2.4.39 (Win64) PHP/7.3.2
[mpm_winnt:notice] [pid 3584:tid 740] AH00456: Apache Lounge VC15 Server built:
[core:notice] [pid 3584:tid 740] AH00094: Command line: 'C:\\Server\\bin\\Apach
[mpm_winnt:notice] [pid 3584:tid 740] AH00418: Parent: Created child process 11
[mpm_winnt:notice] [pid 1140:tid 716] AH00354: Child: Starting 64 worker thread
ing.The 'Apache2.4' service has restarted.winnt:notice] [pid 3584:tid 740] AH00
[ssl:warn] [pid 3584:tid 740] AH01873: Init: Session Cache is not configured [h
```

3V

VOLUME

VARIETY

VELOCITY

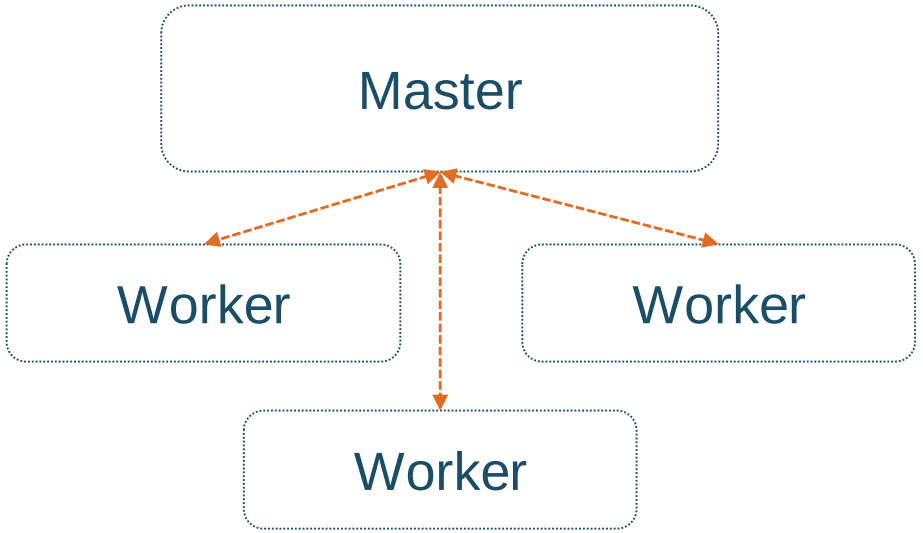
КЛАСТЕР

ПОЯВИЛСЯ КЛАСТЕР

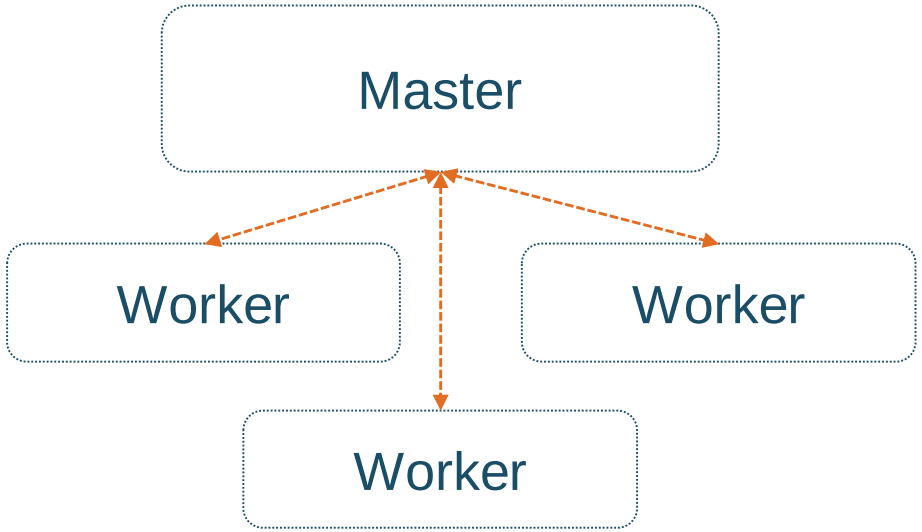
Один сервер !

ПОЯВИЛСЯ КЛАСТЕР

Один сервер !

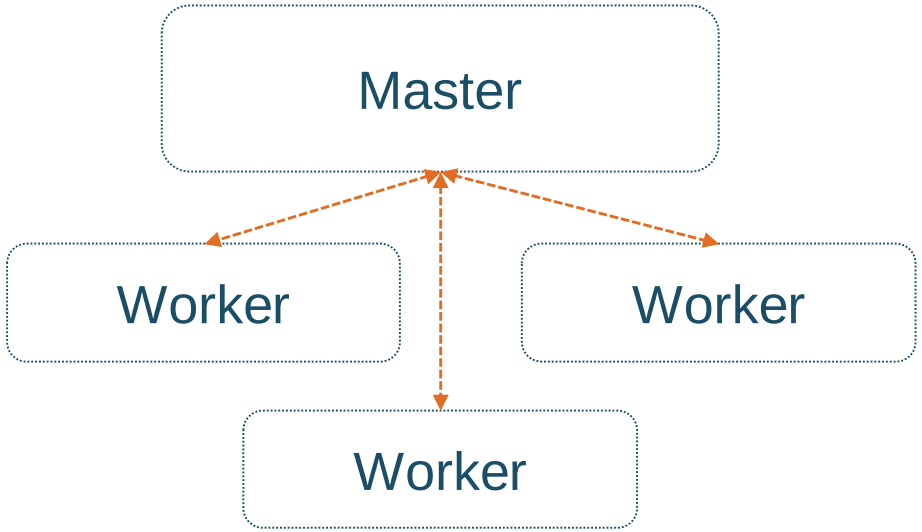


ПОЯВИЛСЯ КЛАСТЕР и добавил проблем



Проблемы координации

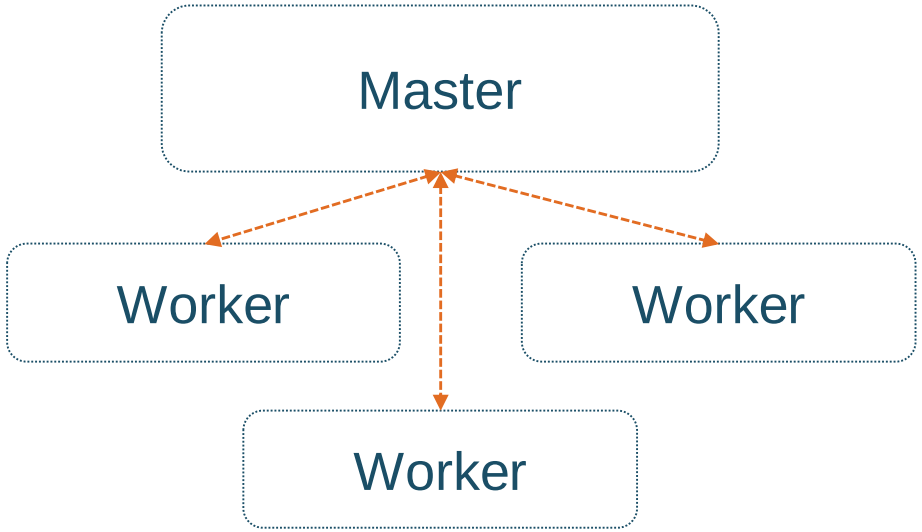
ПОЯВИЛСЯ КЛАСТЕР и добавил проблем



Проблемы координации

Проблемы коммуникации

ПОЯВИЛСЯ КЛАСТЕР и добавил проблем

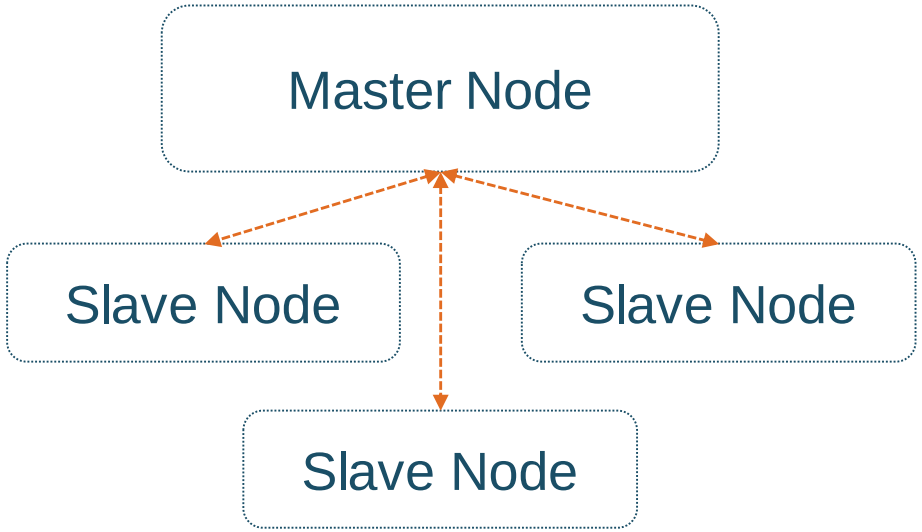


Проблемы координации

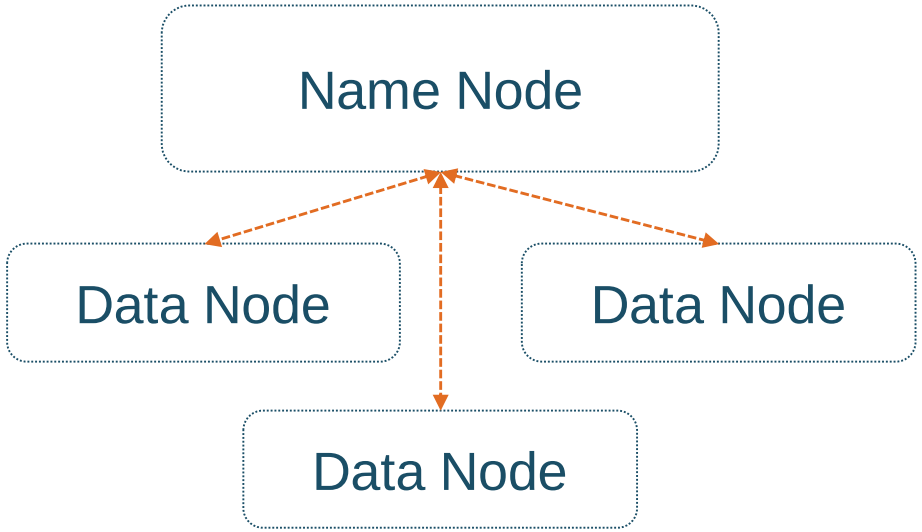
Проблемы коммуникации

Проблемы стабильности

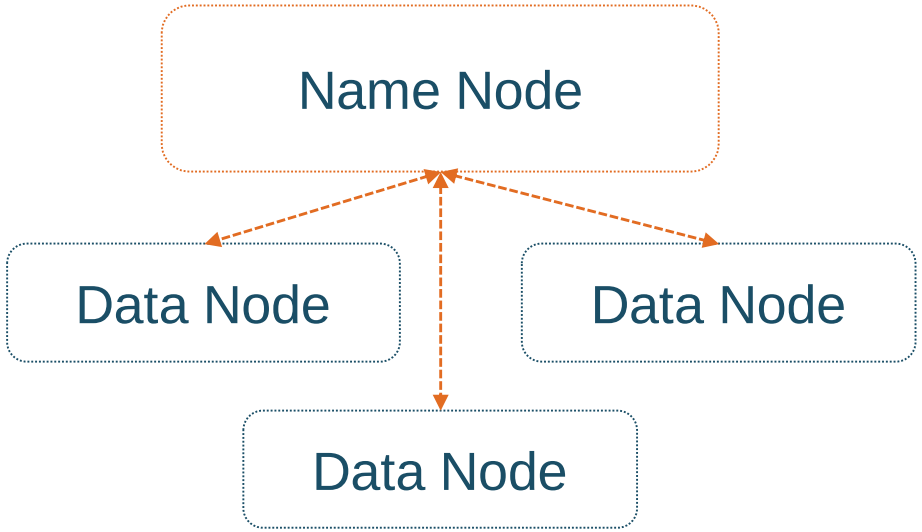
HDFS



HDFS



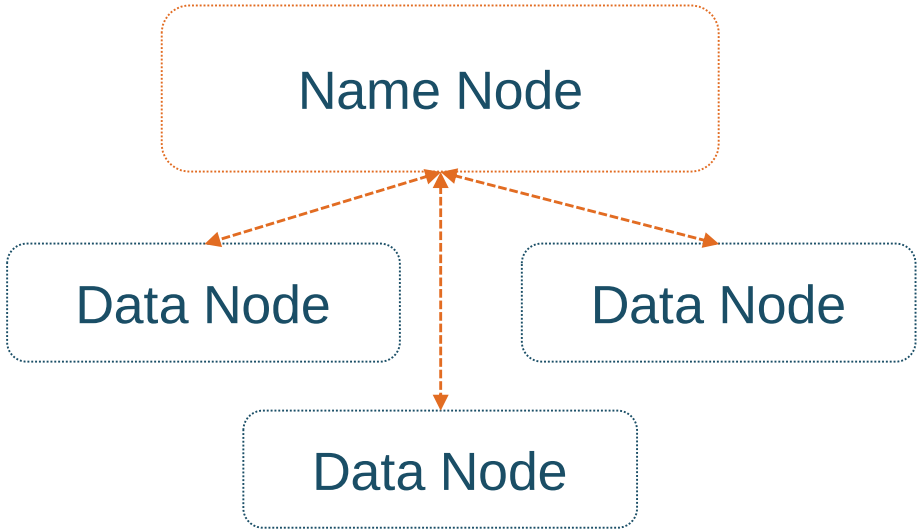
HDFS



Name Node

- Предоставляет и контролирует доступ
- Координирует задачи
- Содержит пространство имен и управляет: (open, close, rename)

HDFS

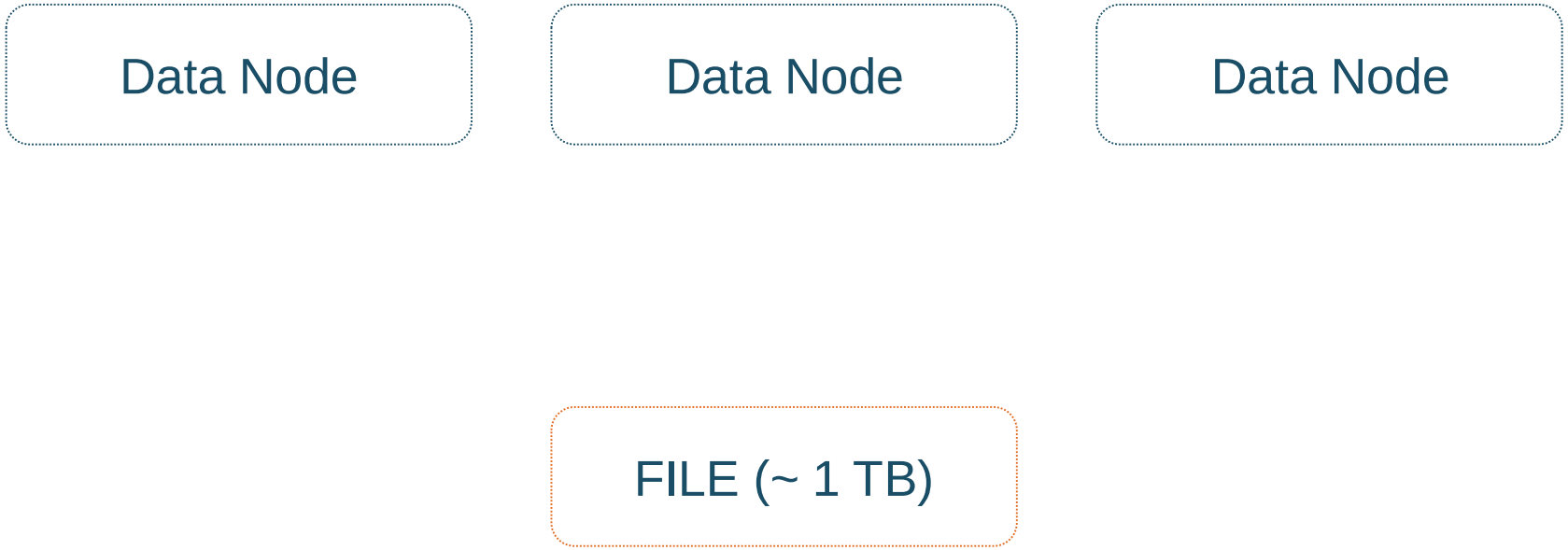


Data Node

- Хранят и обрабатываю данные

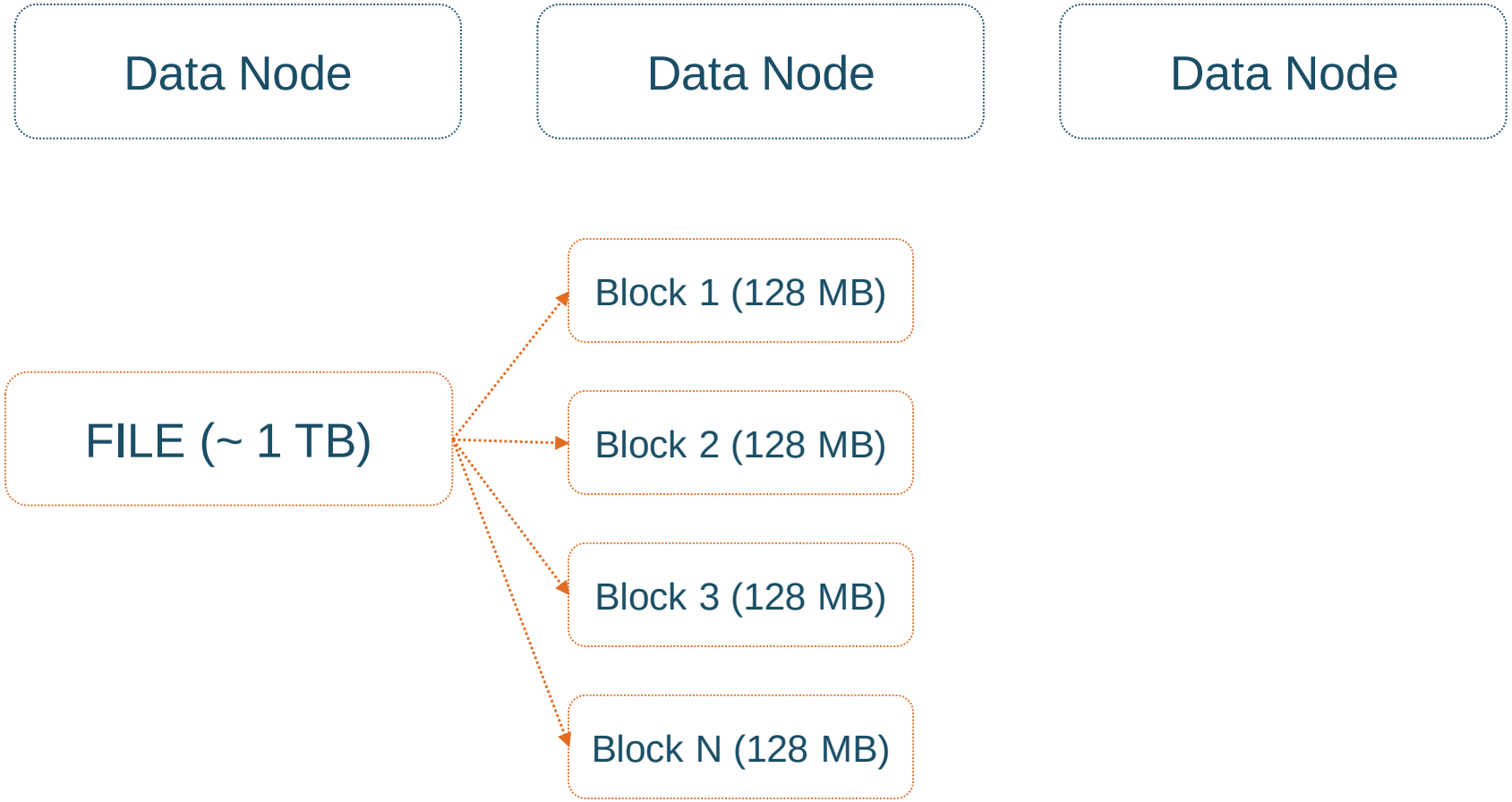
HDFS

ОСОБЕННОСТЬ ХРАНЕНИЯ



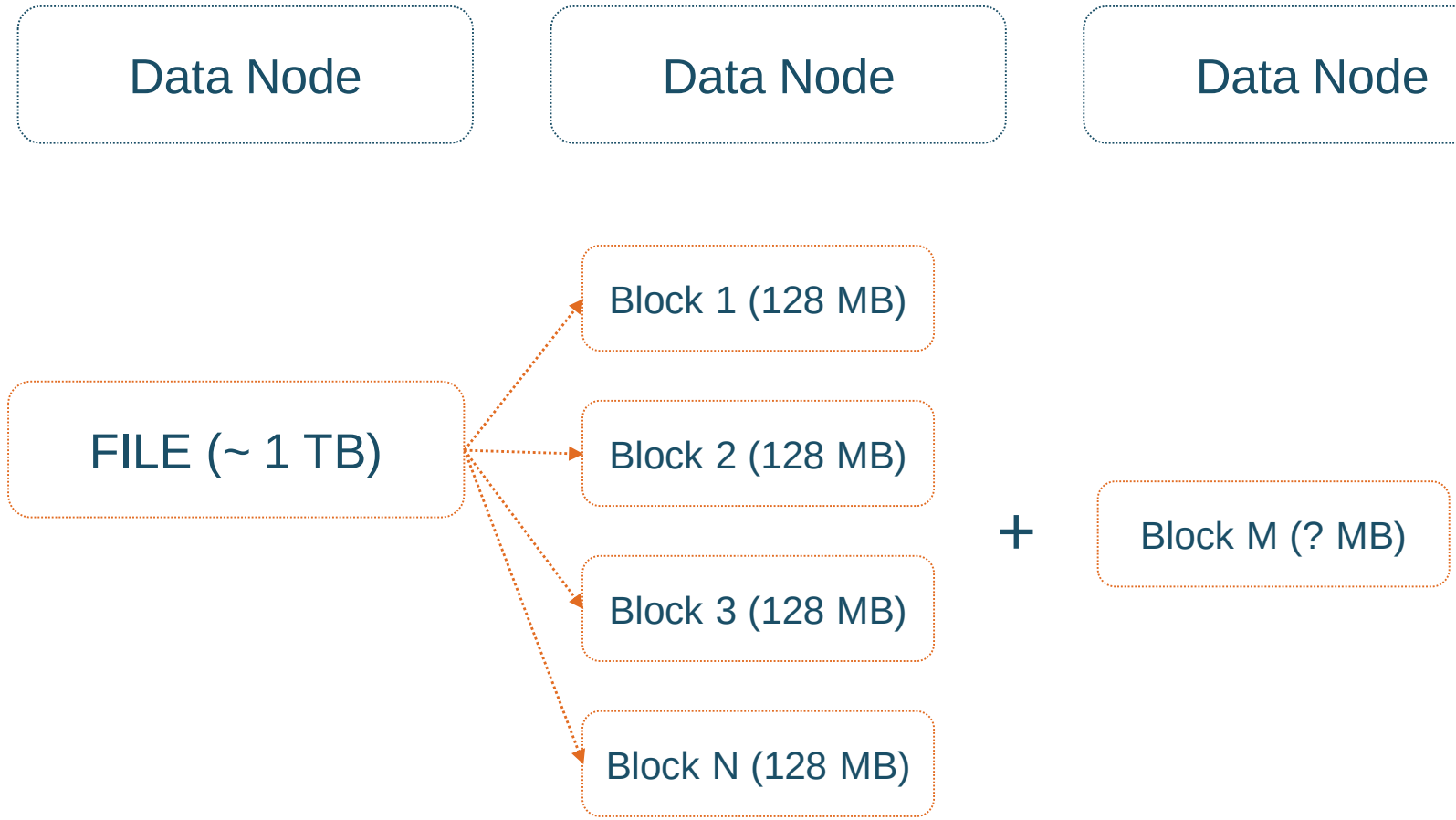
HDFS

ОСОБЕННОСТЬ ХРАНЕНИЯ



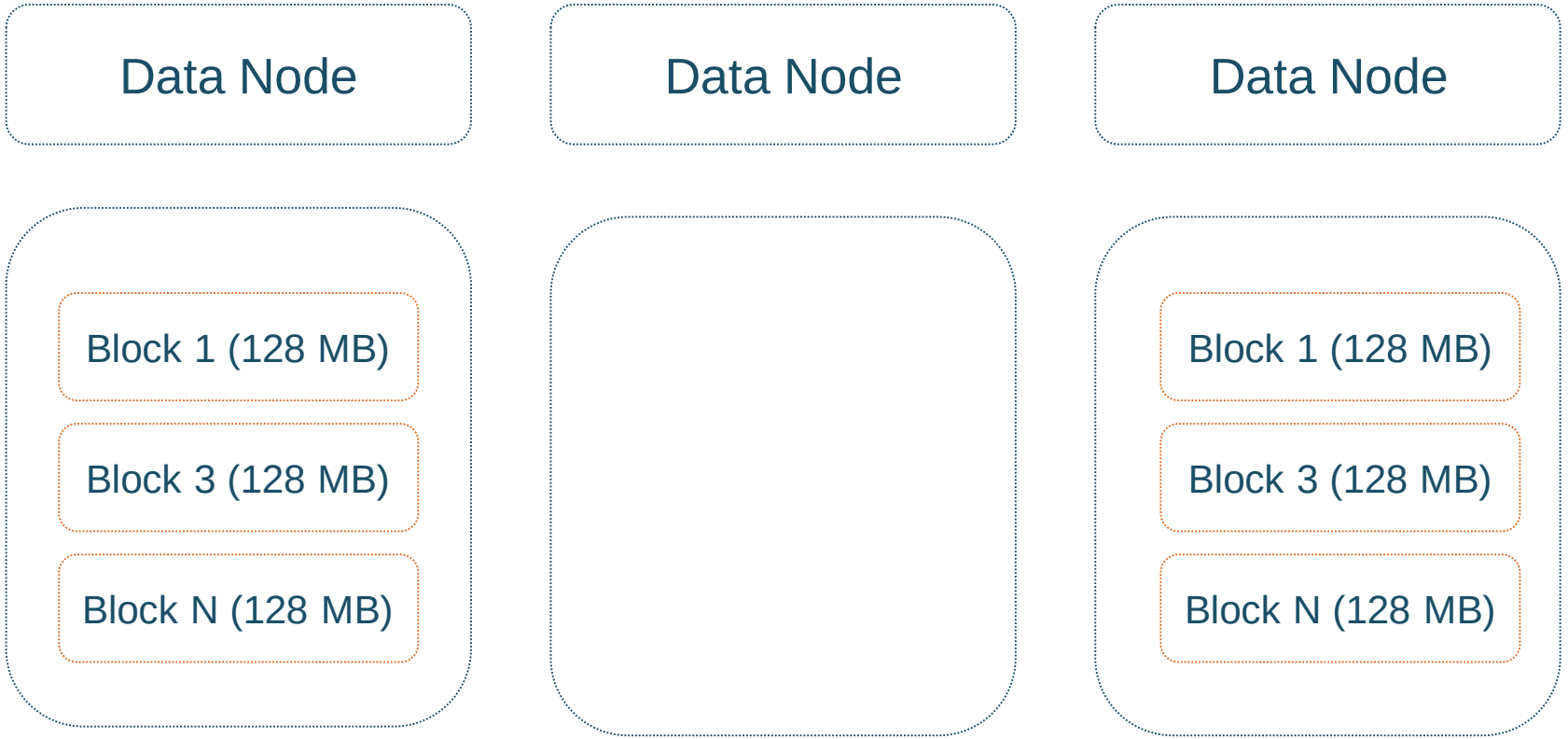
HDFS

ОСОБЕННОСТЬ ХРАНЕНИЯ



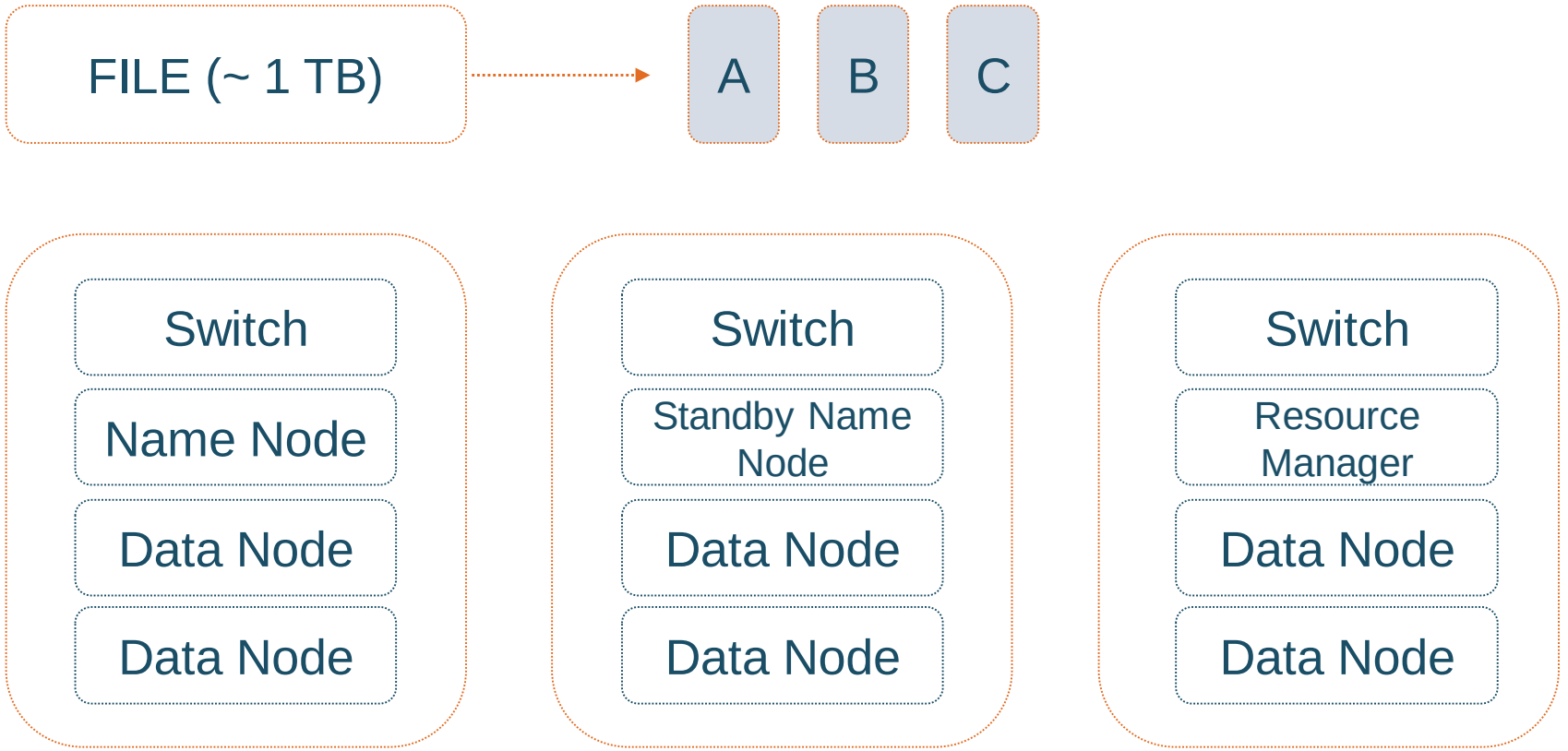
HDFS

ОСОБЕННОСТЬ ХРАНЕНИЯ



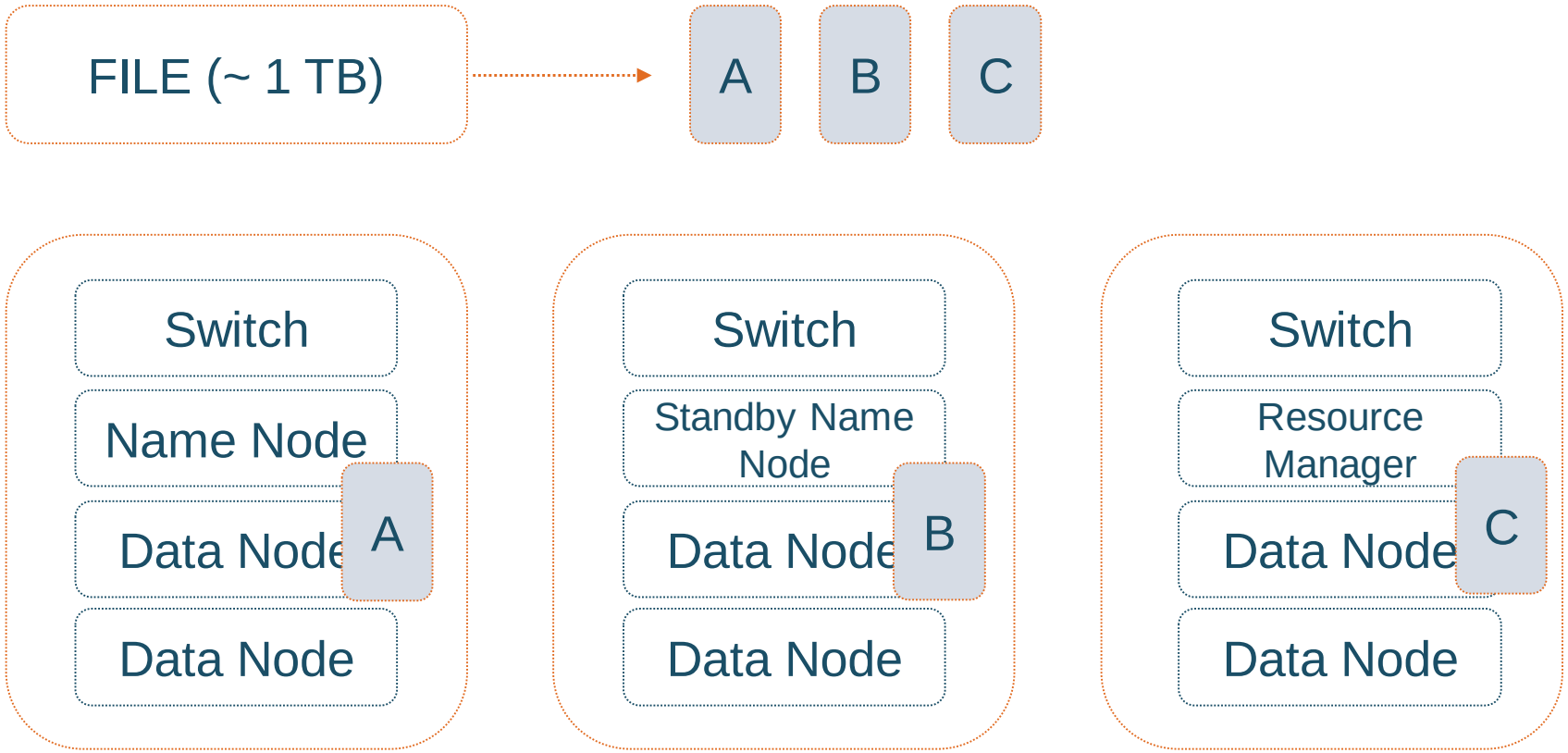
HDFS – RACK AWARENESS

ОСОБЕННОСТЬ ХРАНЕНИЯ



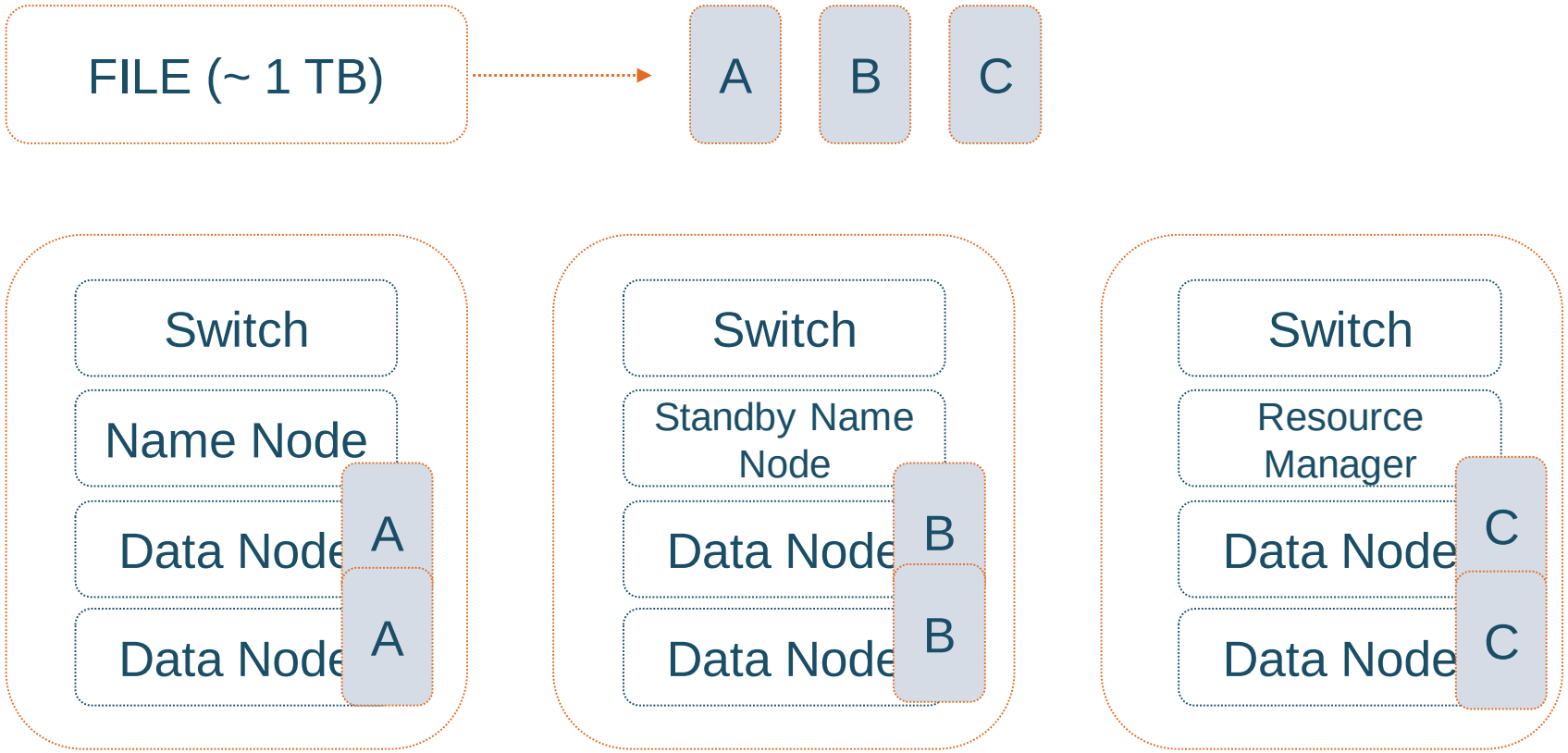
HDFS – RACK AWARENESS

Rep.1
–
на разных узлах



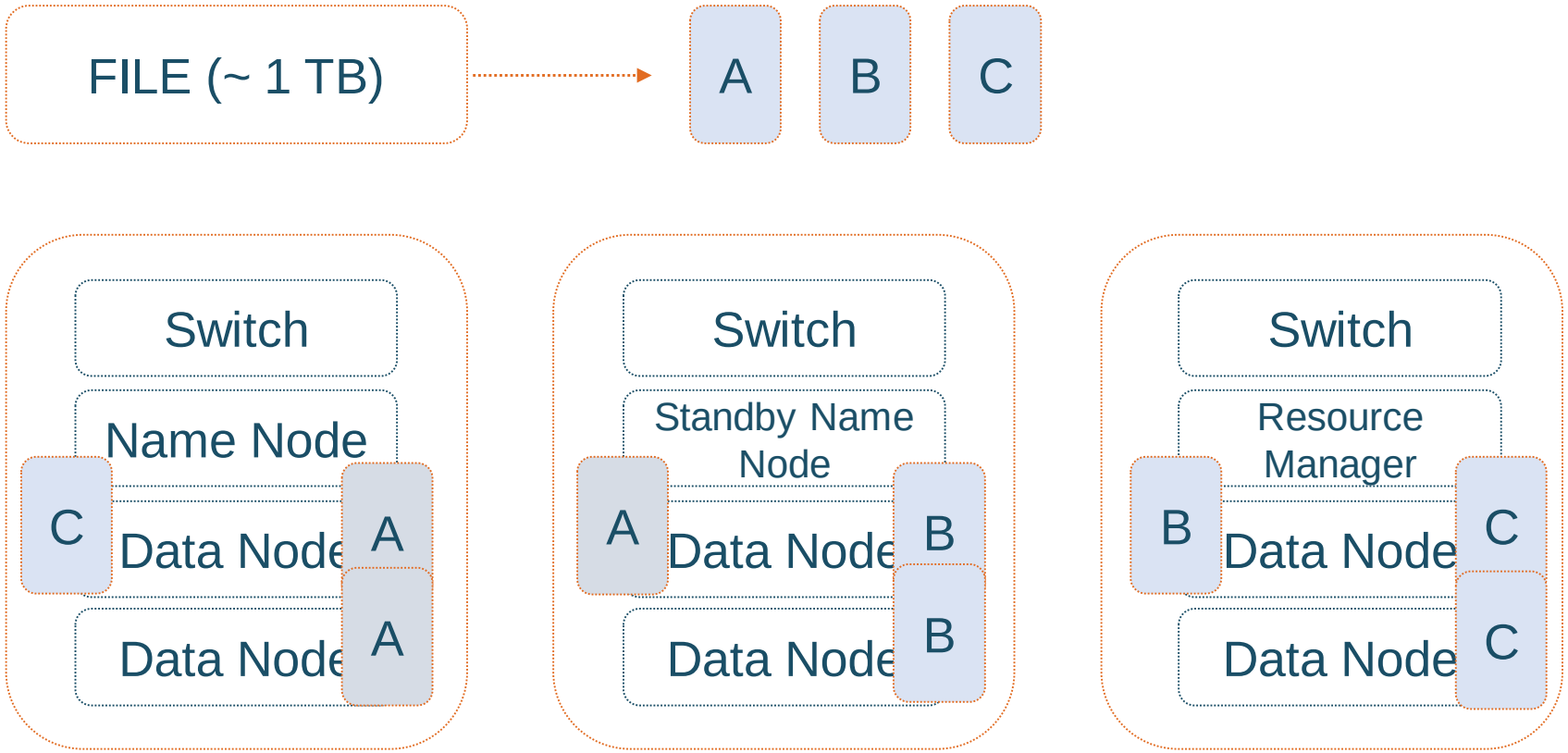
HDFS – RACK AWARENESS

Rep.2
–
на разных нодах внутри



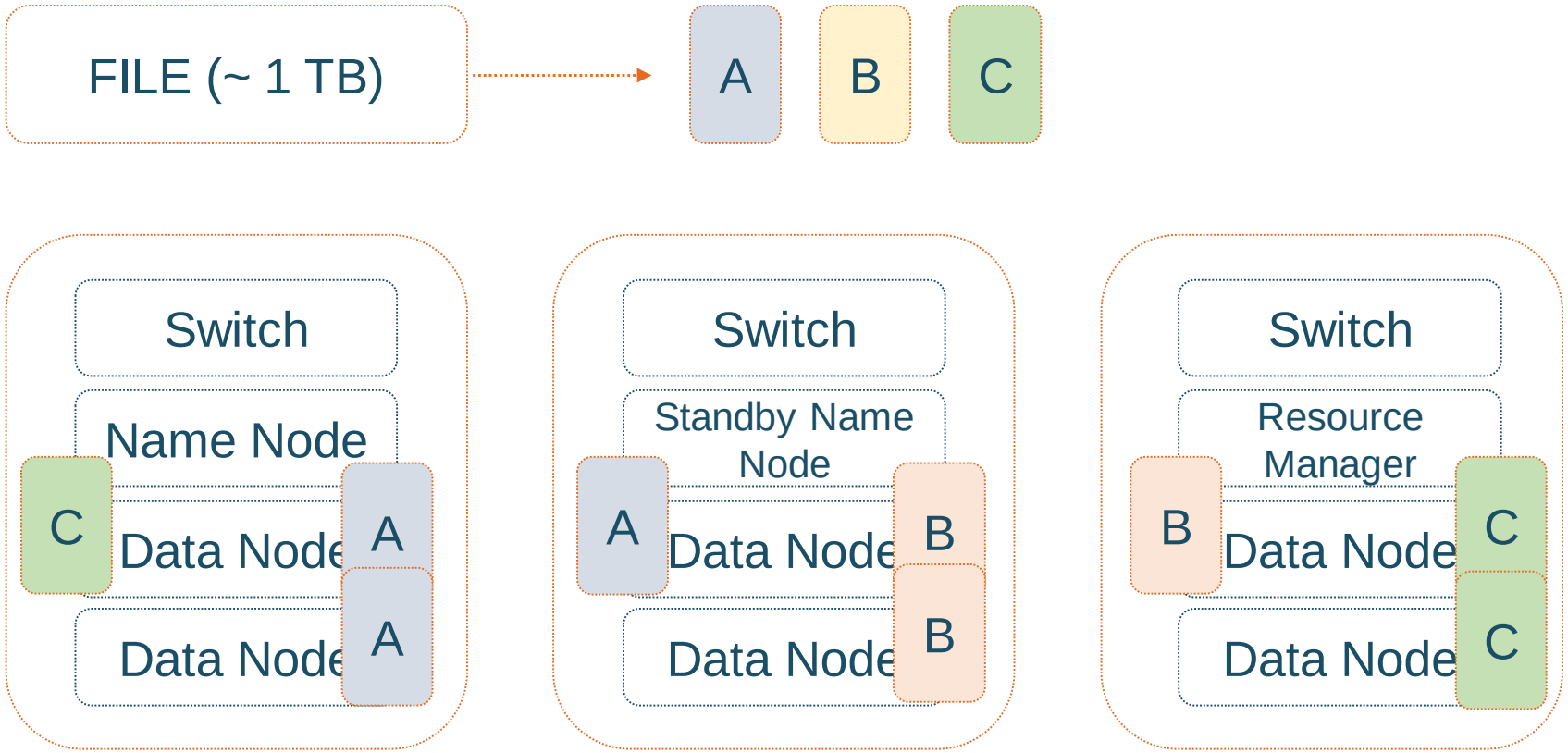
HDFS – RACK AWARENESS

Rep.3
–
на разных стойках



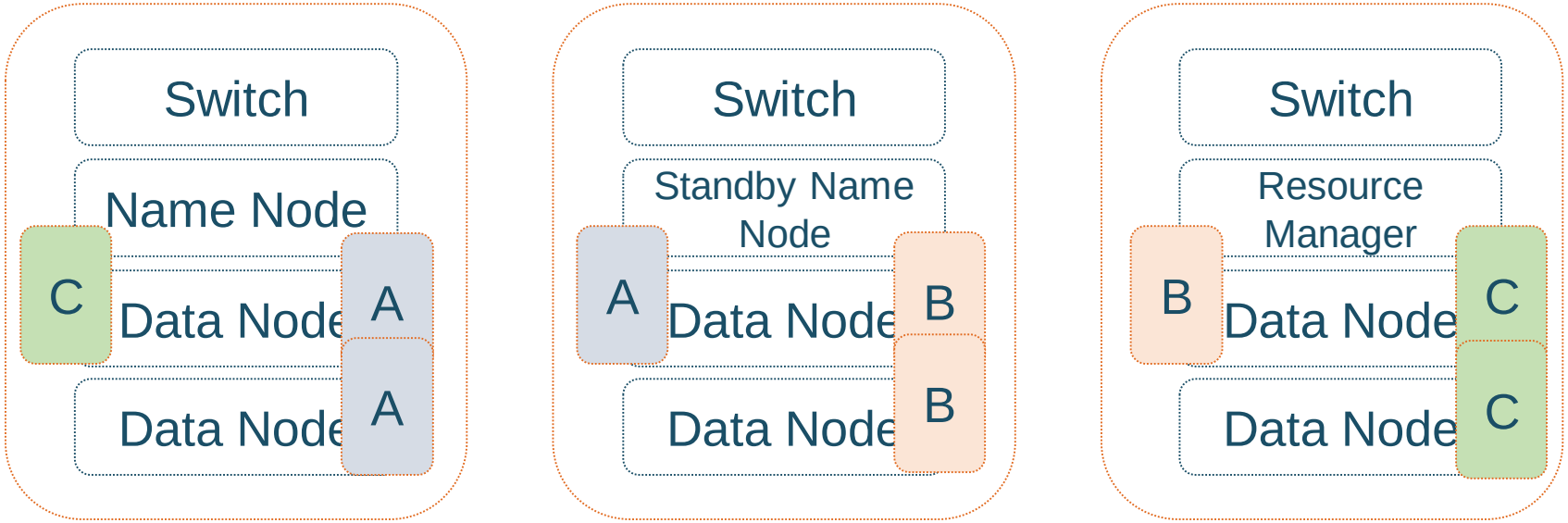
HDFS – RACK AWARENESS

Rep.3
–
на разных стойках



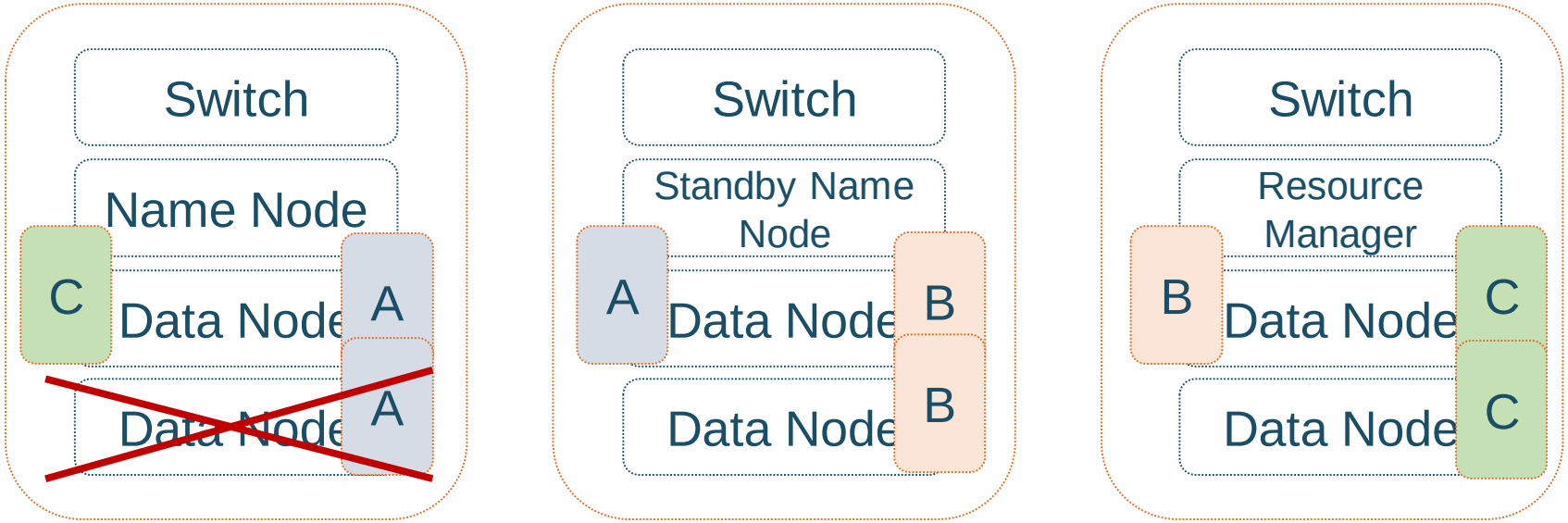
HDFS – FAULT TOLERANCE

Данные доступны всегда



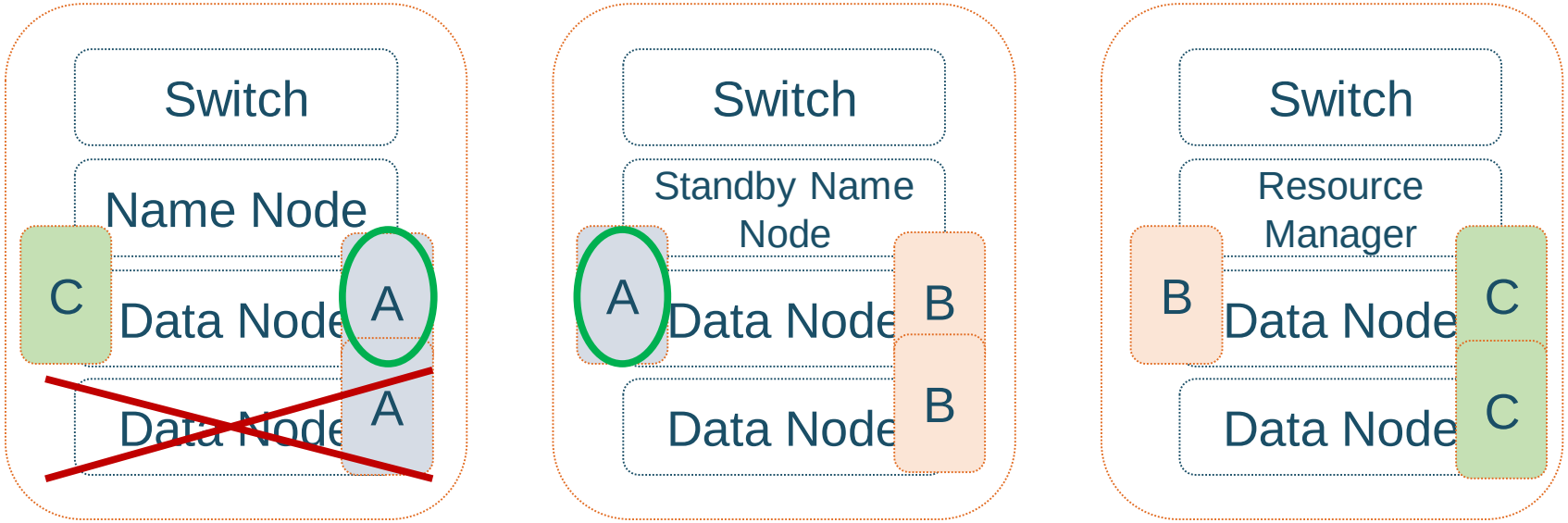
HDFS – FAULT TOLERANCE

Данные доступны всегда



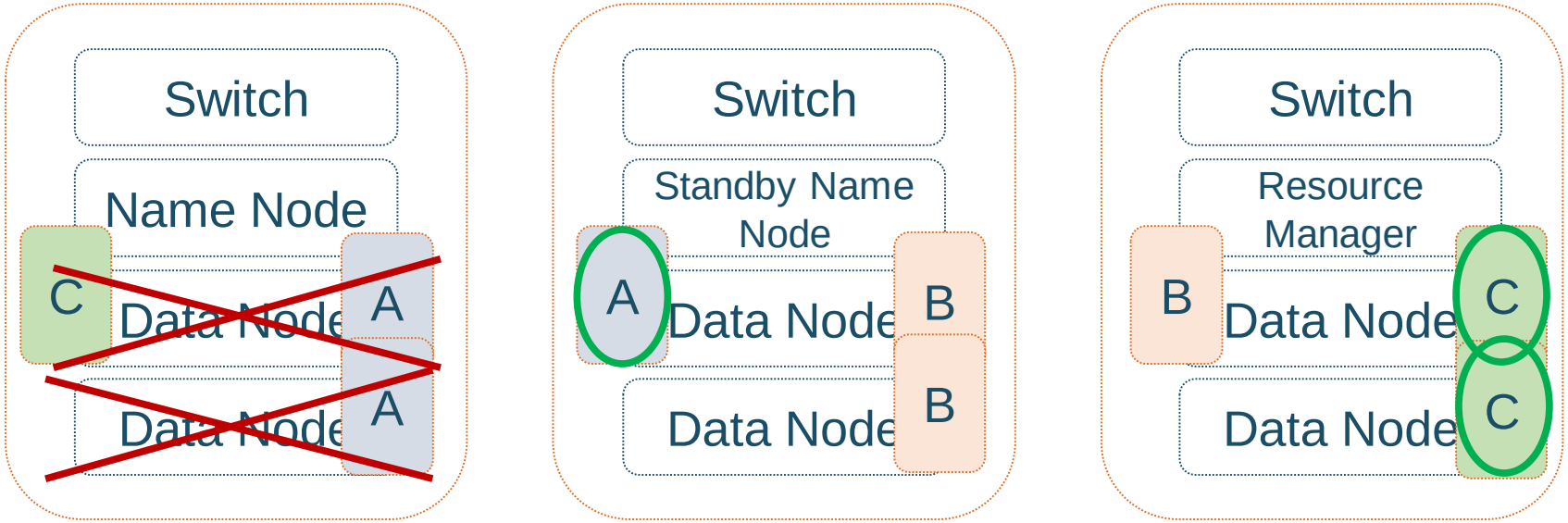
HDFS – FAULT TOLERANCE

Данные доступны всегда



HDFS – FAULT TOLERANCE

Данные доступны всегда



HDFS | БАЛАНСИРОВКА ДИСКОВ

Расчет баланса
и утилизации диска

	Disk1	Disk2	Disk3	Disk4		Total	Ideal
объем	256	512	1024	2048		3840	0,45
использовано	100	176	950	520		1746	
% использования	0,39	0,34	0,93	0,25			
% плотности	0,06	0,11	-0,48	0,2			

- Total объем = сумма объемов всех дисков == sum(Disk (1->N))
- Total использ. = сумма использования всех дисков == sum(Disk (1->N)
- Ideal = Total использ. / Total объем

HDFS | БАЛАНСИРОВКА ДИСКОВ

Расчет баланса
и утилизации диска

	Disk1	Disk2	Disk3	Disk4		Total	Ideal
объем	256	512	1024	2048		3840	0,45
использовано	100	176	950	520		1746	
% использования	0,39	0,34	0,93	0,25			
% плотности	0,06	0,11	-0,48	0,2			

- % плотности = Ideal - % использования

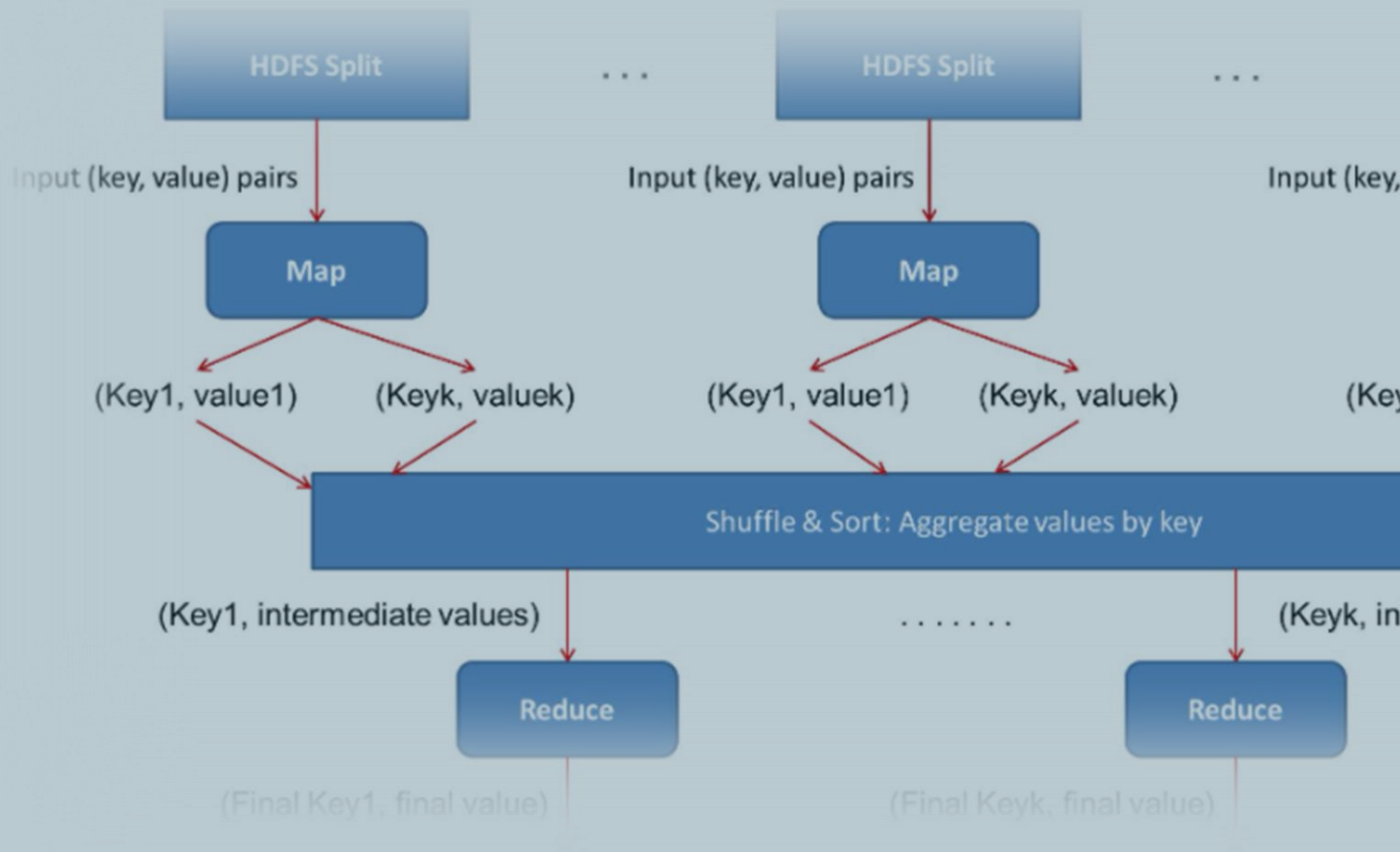
HDFS | БАЛАНСИРОВКА ДИСКОВ

Нормализуем

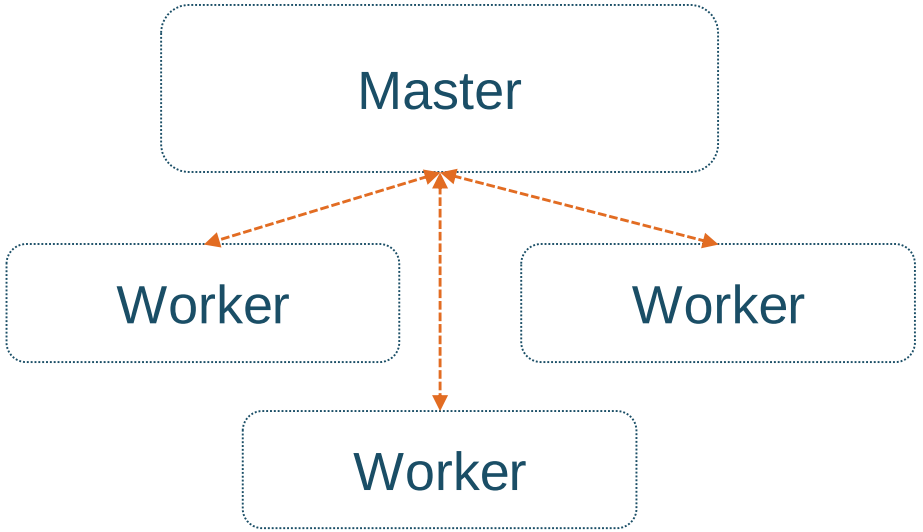
- hdfs diskbalancer – plan <datanode>
- настраиваем ednabled, out, thresholdPercentage, maxerror
- проверяем отчеты:

  hdfs diskbalancer –fs namenode.url –report file_path//

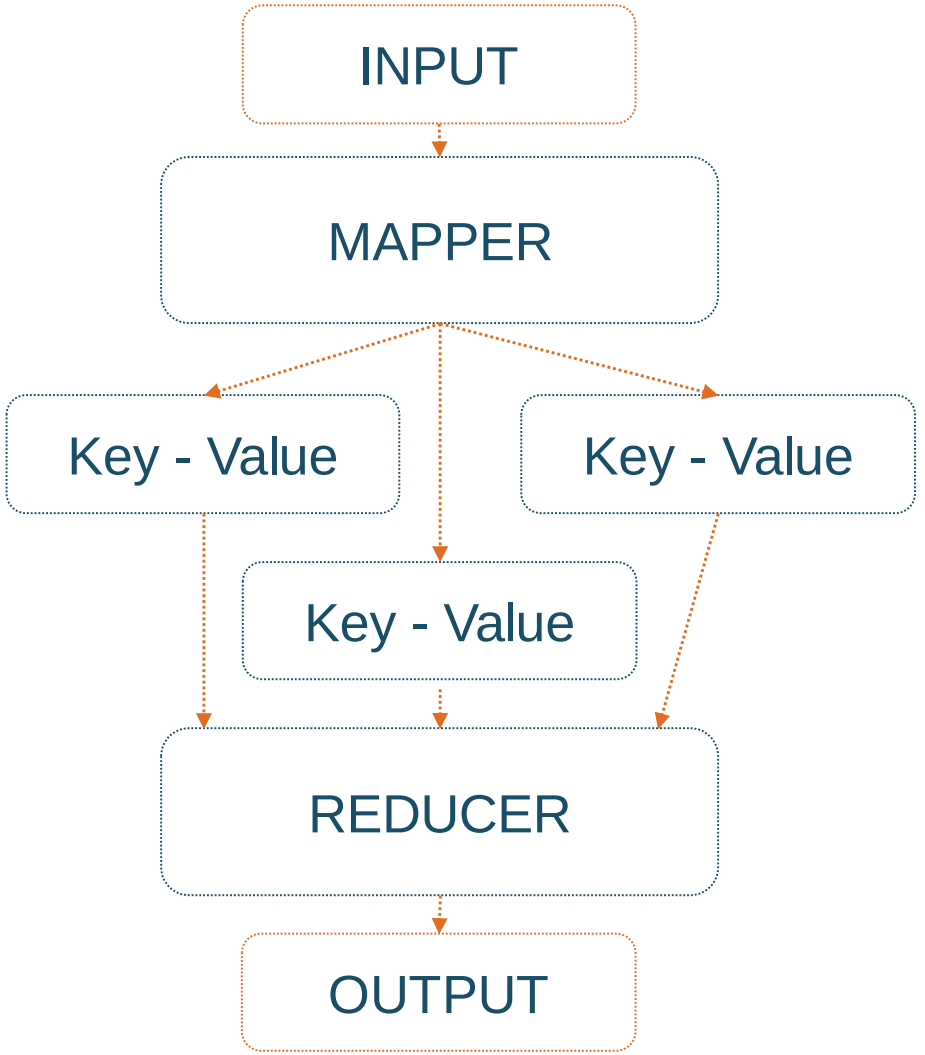
MAP - REDUCE


HDFS Split
HDFS Split
Input (key, value) pairs
Input (key, value) pairs
Map
Map
(Key1, value1)
(Keyk, valuek)
(Key1, value1)
(Keyk, valuek)
Shuffle & Sort: Aggregate values by key
(Key1, intermediate values)
Reduce
Reduce
(Final Key1, final value)
(Final Keyk, final value)

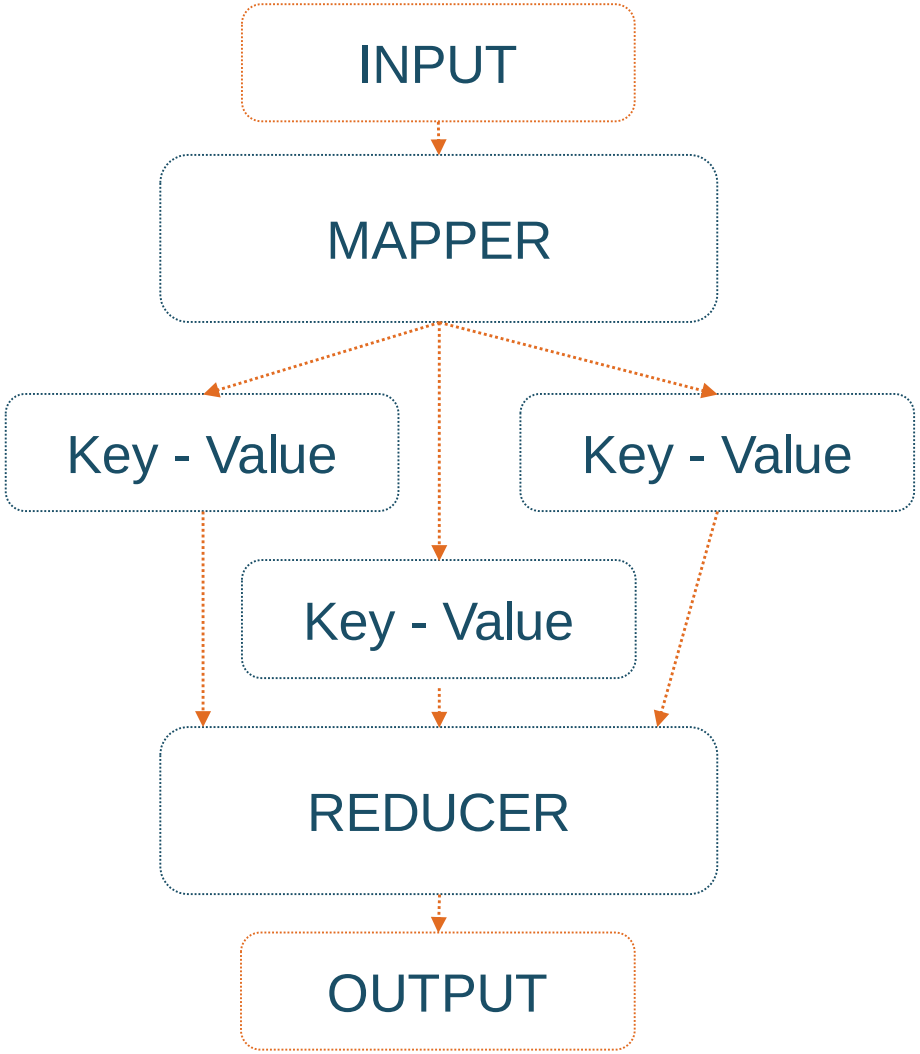
MAP – REDUCE



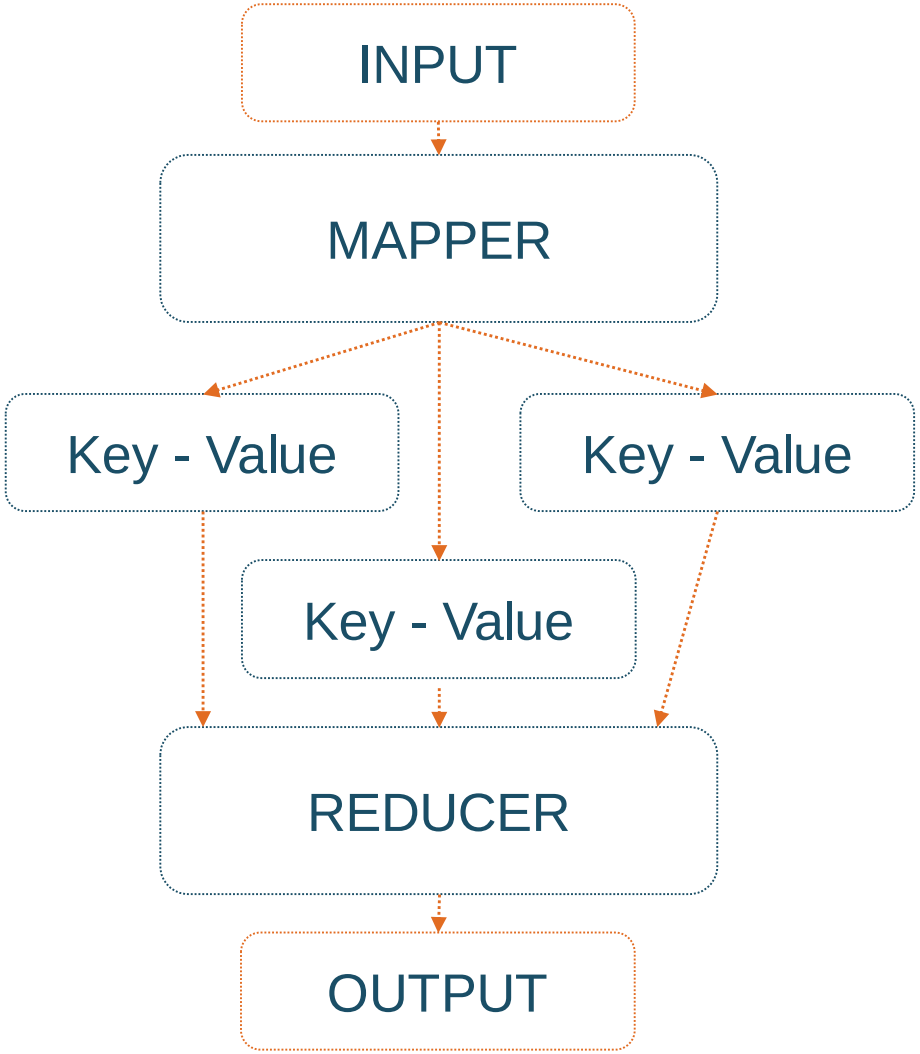
MAP – REDUCE



MAP – REDUCE



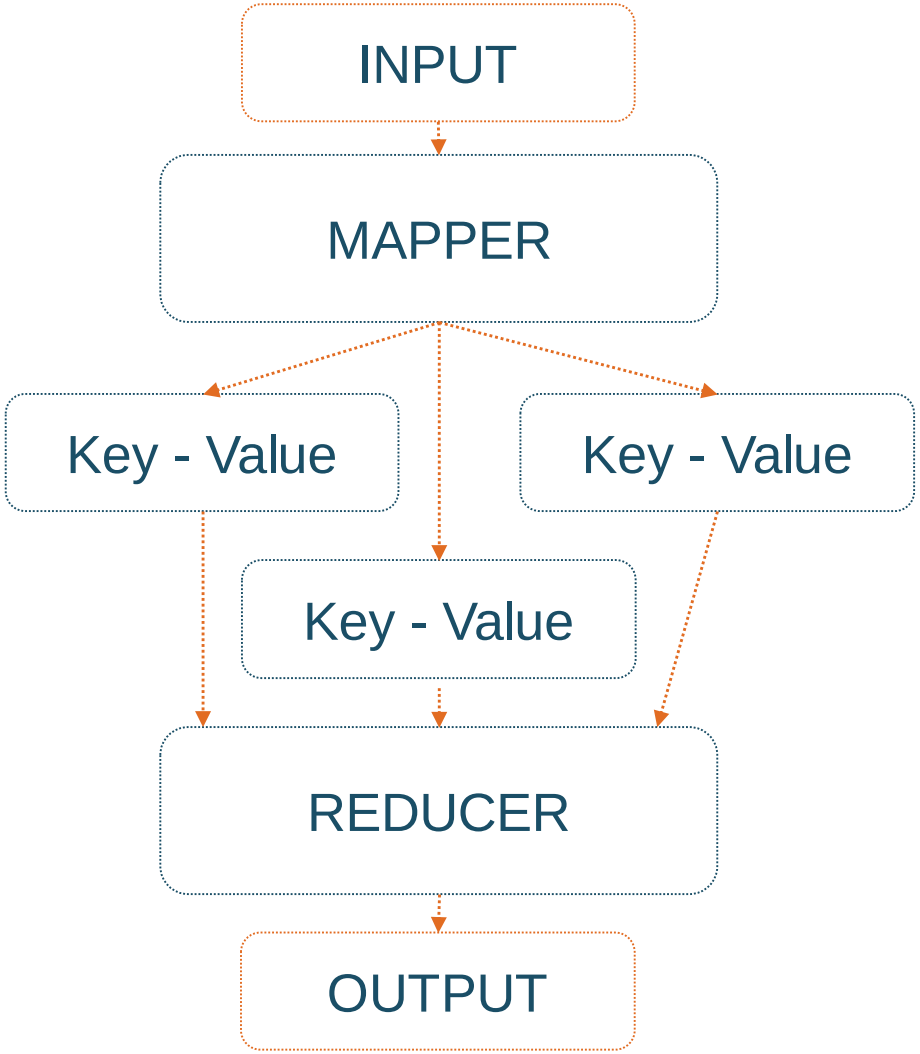
MAP – REDUCE



INPUT

обычный файл\tc вашими\tданными

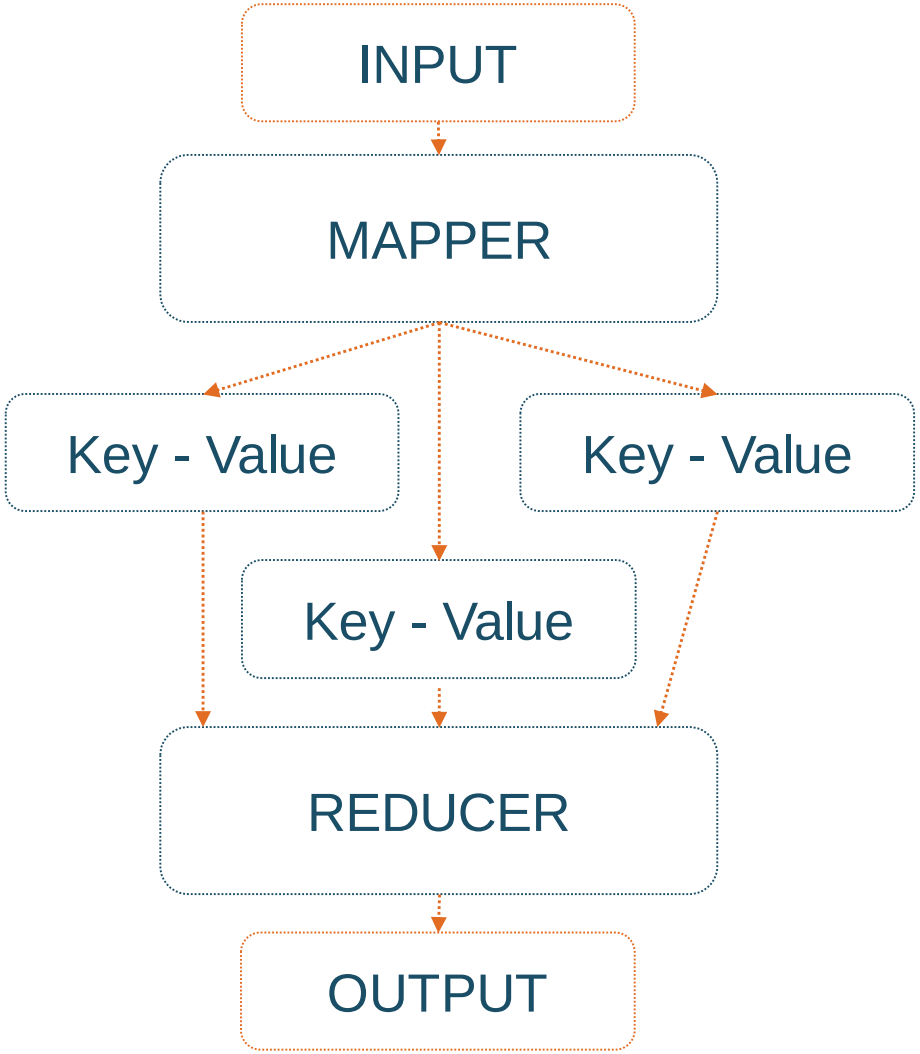
MAP – REDUCE



INPUT

обычный файл\tс вашими\tданными

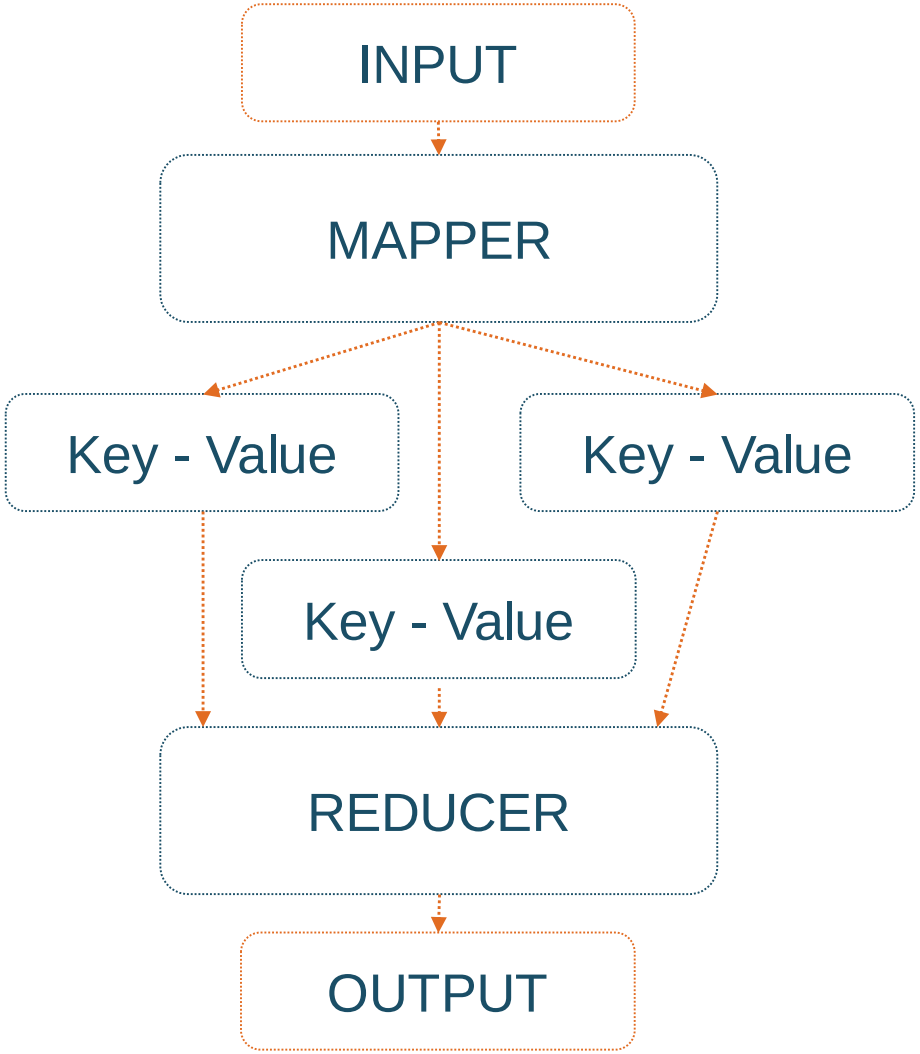
MAP – REDUCE



MAPPER

- str(обычный файл\tс вашими\tданными)
- list(list(str(обычный файл),
  str(с вашими),
  str(данными)
  ))
- function(object) <- list(str)
- return: key – value

MAP – REDUCE



MAPPER

- опция процесса:

  mapred.max.split.size

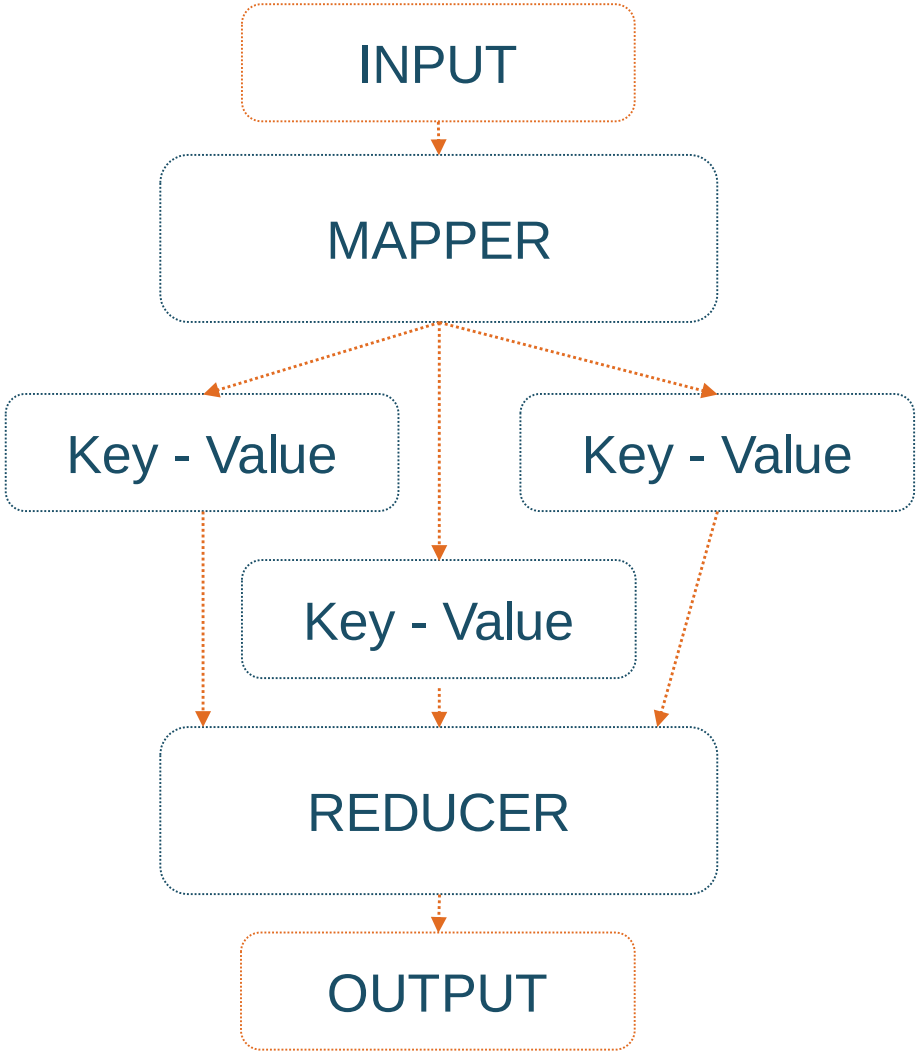
- формула расчета мапперов:

  общий размер данных / mapred.max.split.size

  Пример: 1TB данных, 100MB split size:

  (1000*1000) / 100 = 10000 мапперов

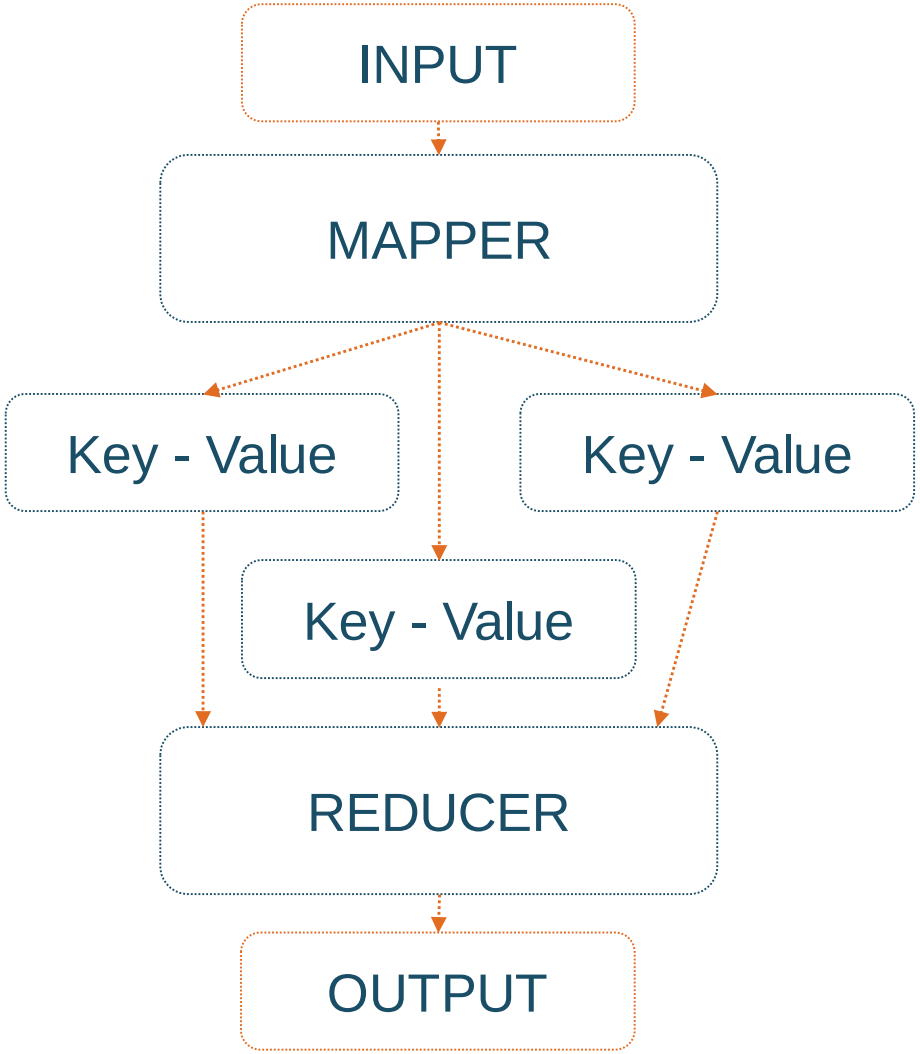
MAP – REDUCE



REDUCER

- list(key - value)
- function(object) <- key - value
- return: result

MAP – REDUCE



REDUCER

- опция процесса:

  job.setNumreduceTasks(int)

- формула расчета редьюсеров:

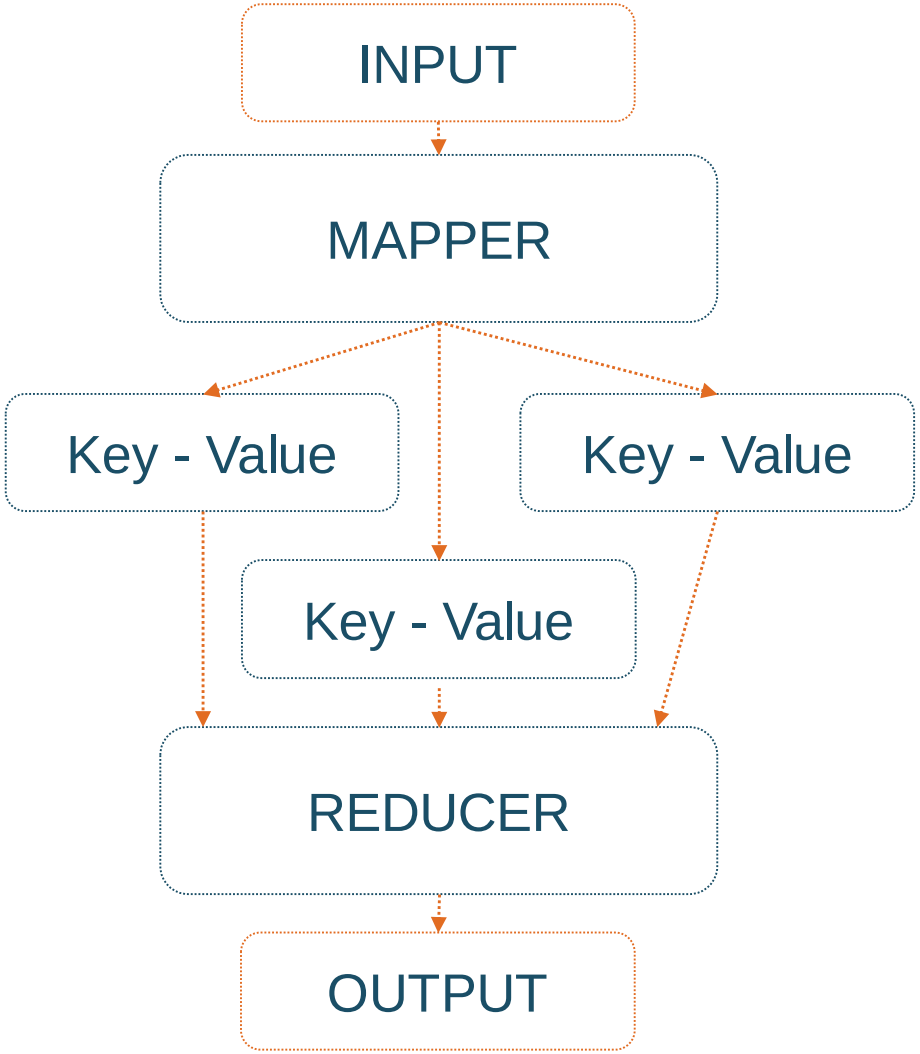
  const * (кол-во нод * макс. контейнеров на ноде)

  Пример: const = 0.95 или 1.75 всегд, 3 ноды,

  8 контейнеров на ноде

  math.ceil(0.95 * (3 * 8))

MAP – REDUCE



REDUCER

- опция процесса:

  job.setNumreduceTasks(int)

  если int = 0, то reducers не выполнятся
- формула расчета редьюсеров:

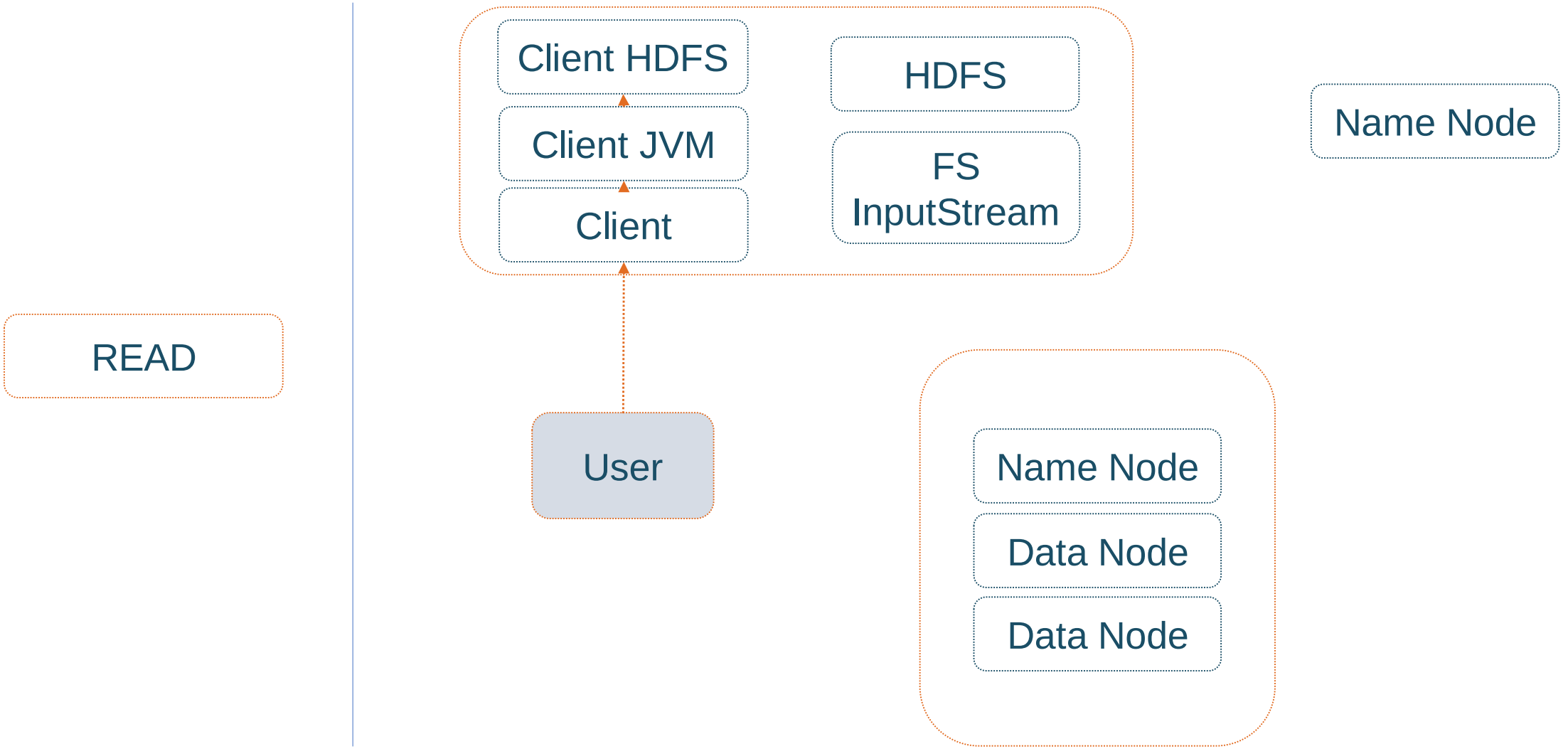
  const * (кол-во нод * макс. контейнеров на ноде)

  Пример: const = 0.95 или 1.75 всегд, 3 ноды,

  8 контейнеров на ноде

  math.ceil(0.95 * (3 * 8))

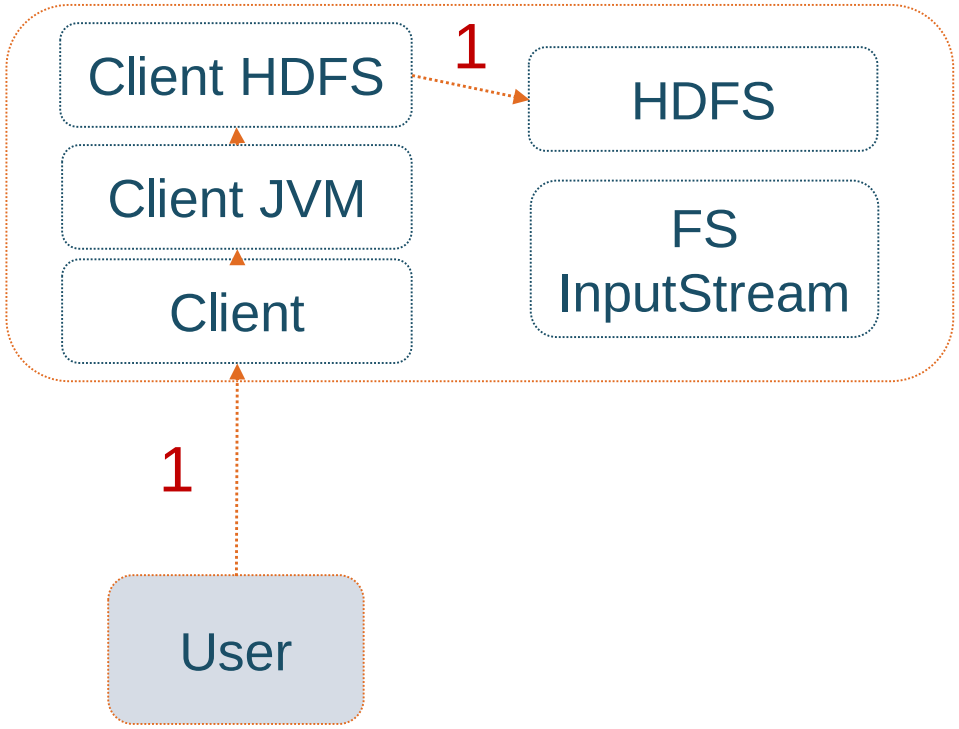
MAP – REDUCE | READ HDFS



MAP – REDUCE | READ HDFS

READ

1
Обращение клиента.
Открытие HDFS



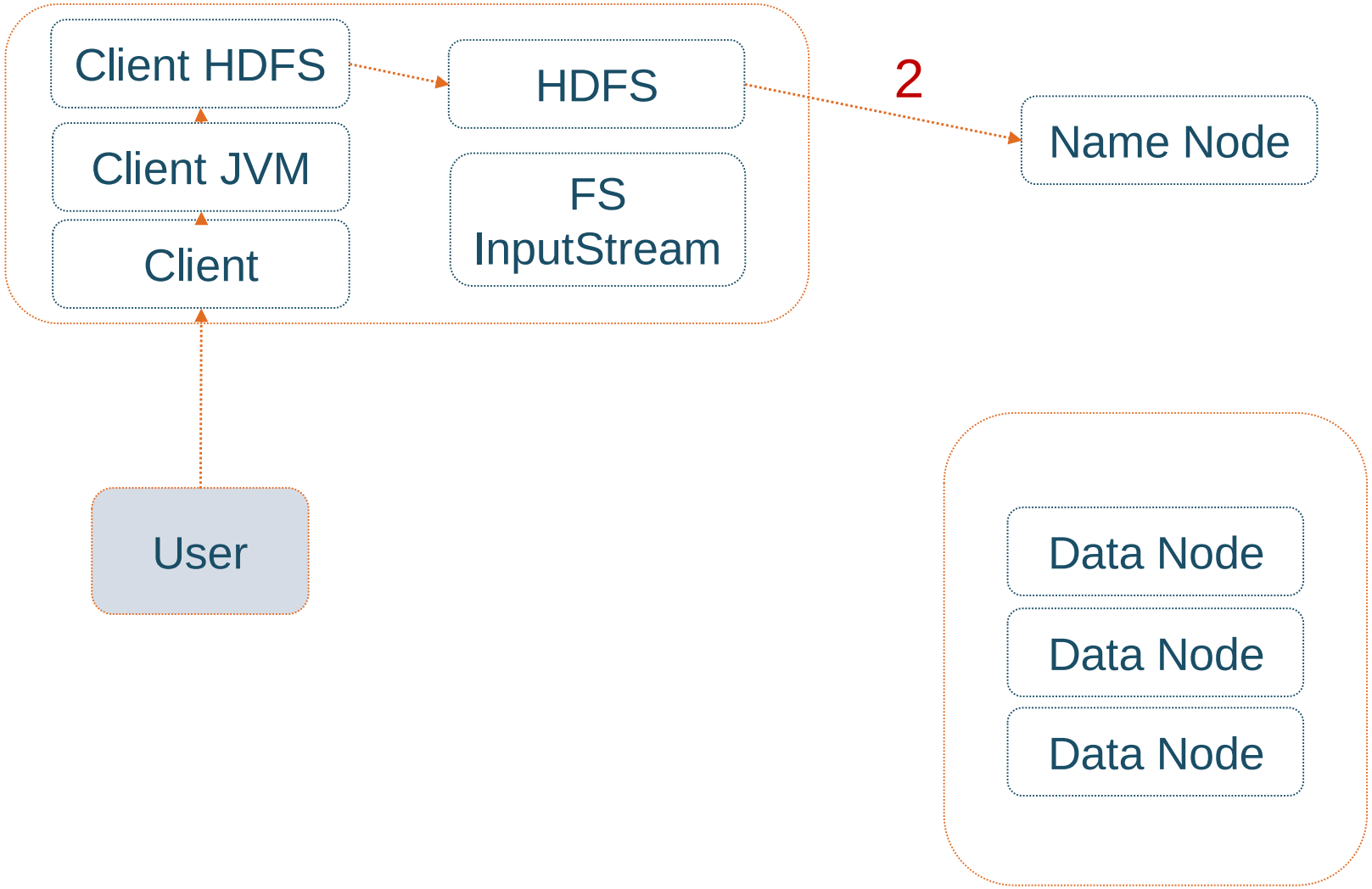
Name Node

Data Node
Data Node
Data Node

MAP – REDUCE | READ HDFS

READ

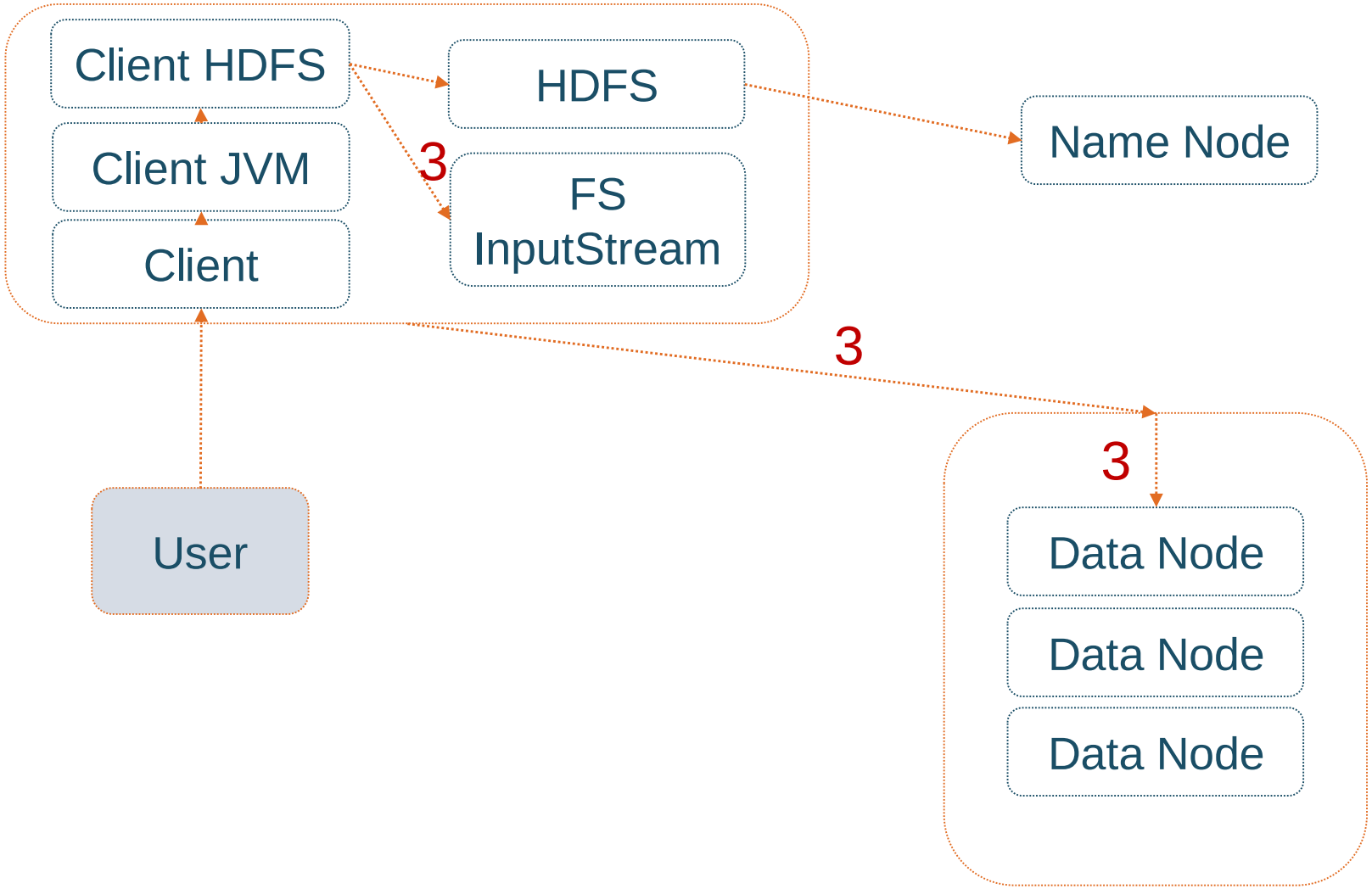
2
Получение мест расположения объектов



MAP – REDUCE | READ HDFS

READ

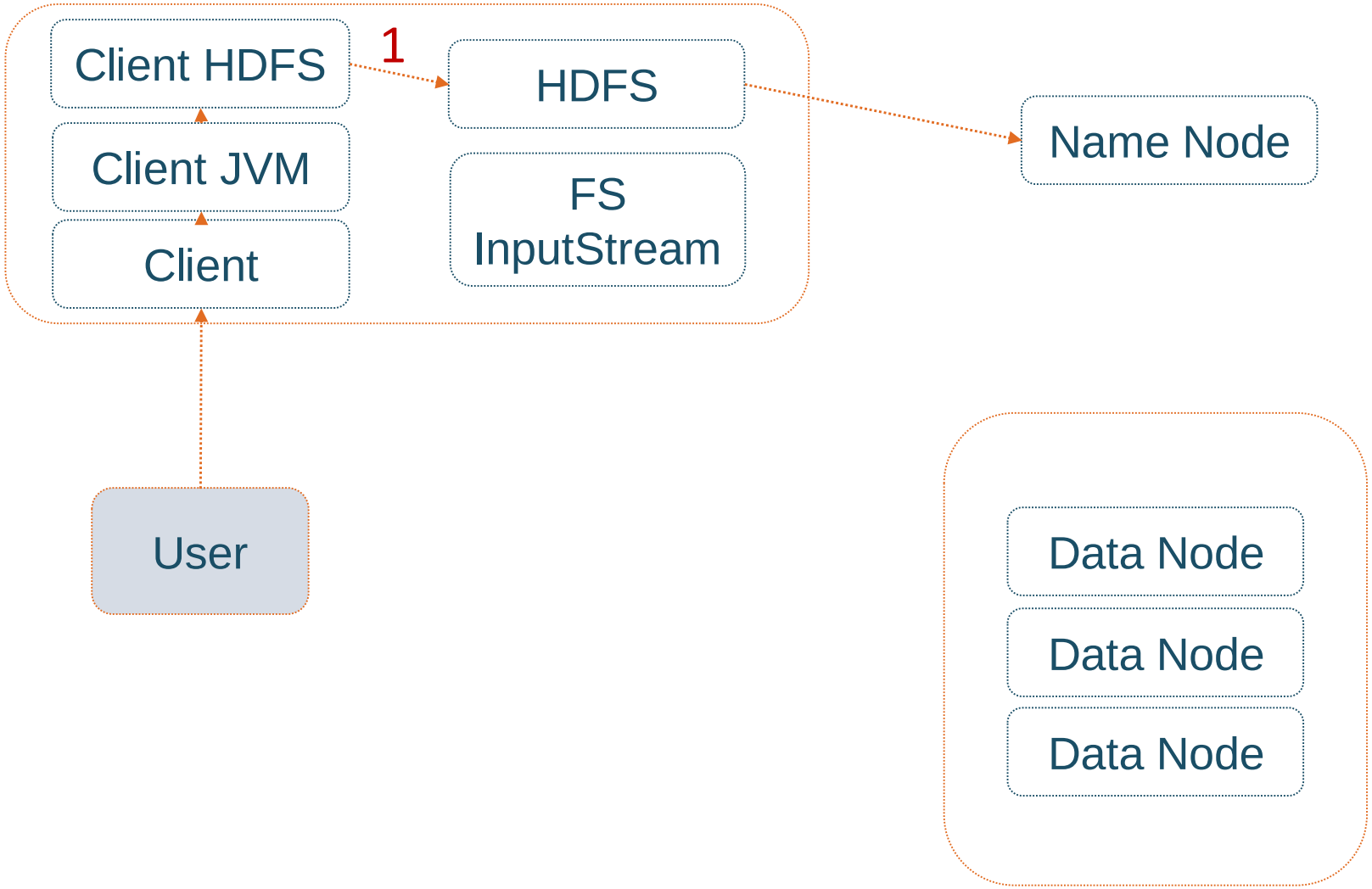
3
Чтение блоков



MAP – REDUCE | WRITE HDFS

READ

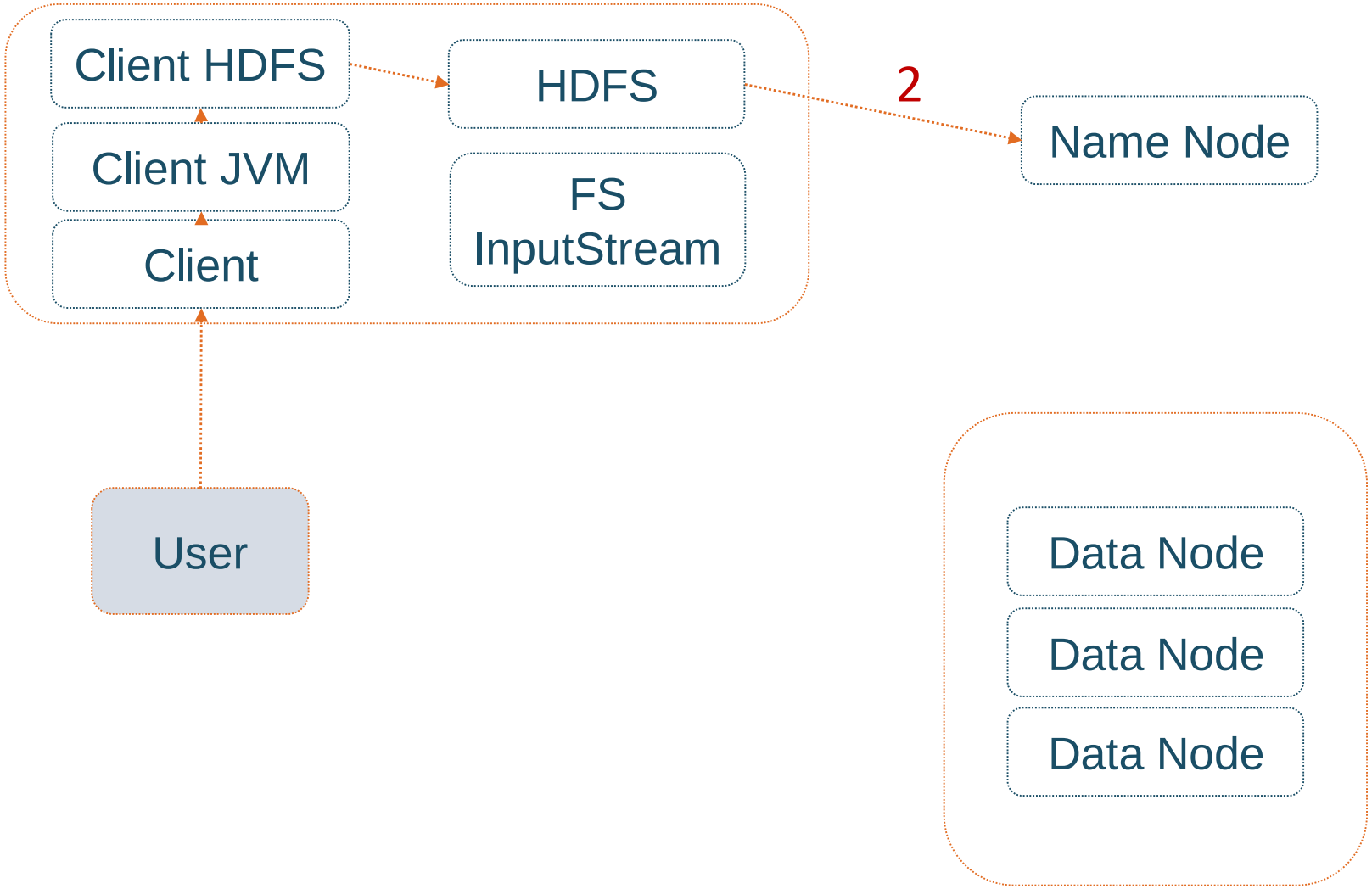
1
Создание файла



MAP – REDUCE | WRITE HDFS

READ

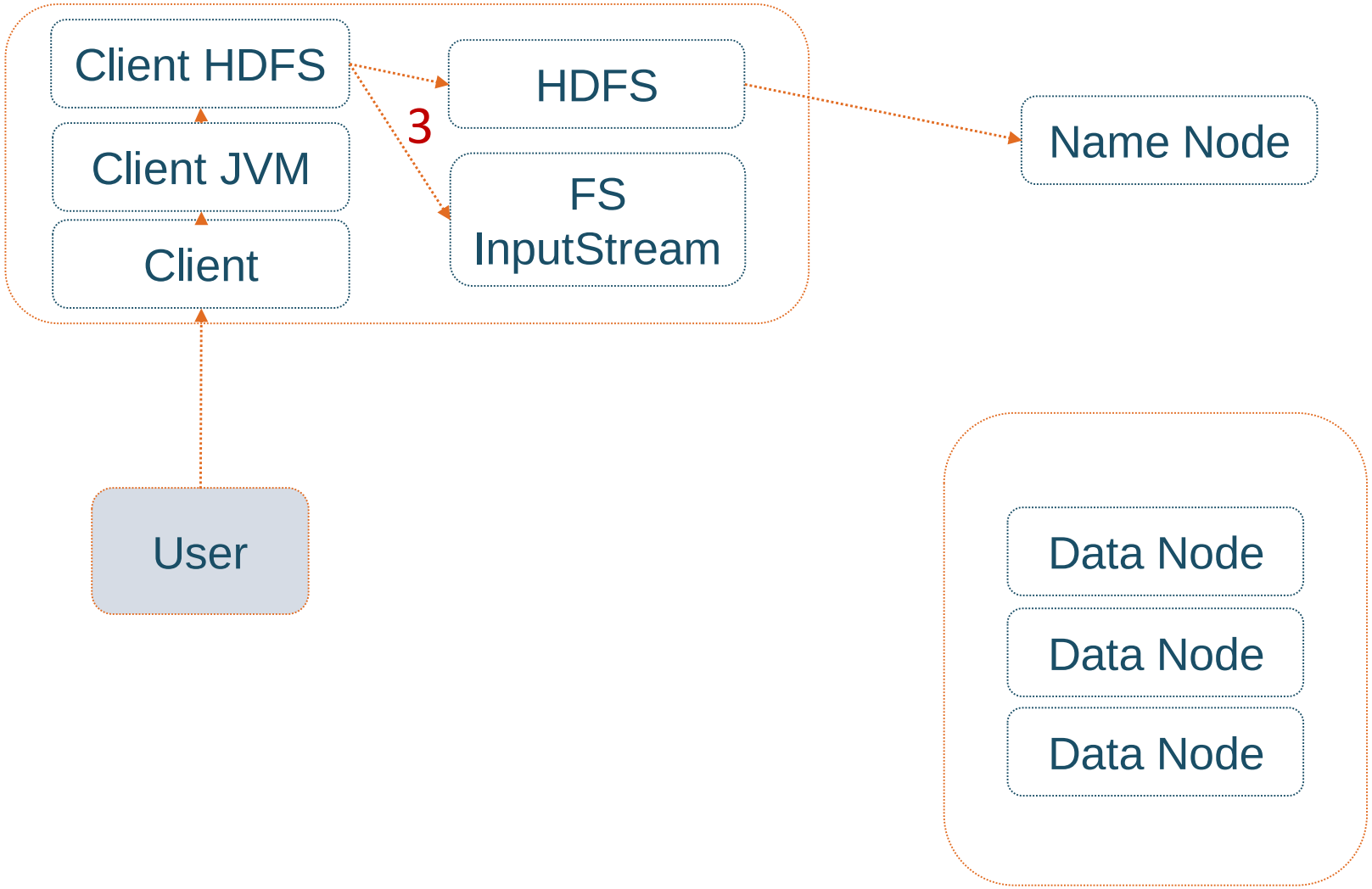
2
Получение файла
в файловой системе
(без блоков)



MAP – REDUCE | WRITE HDFS

READ

3
Запись файла

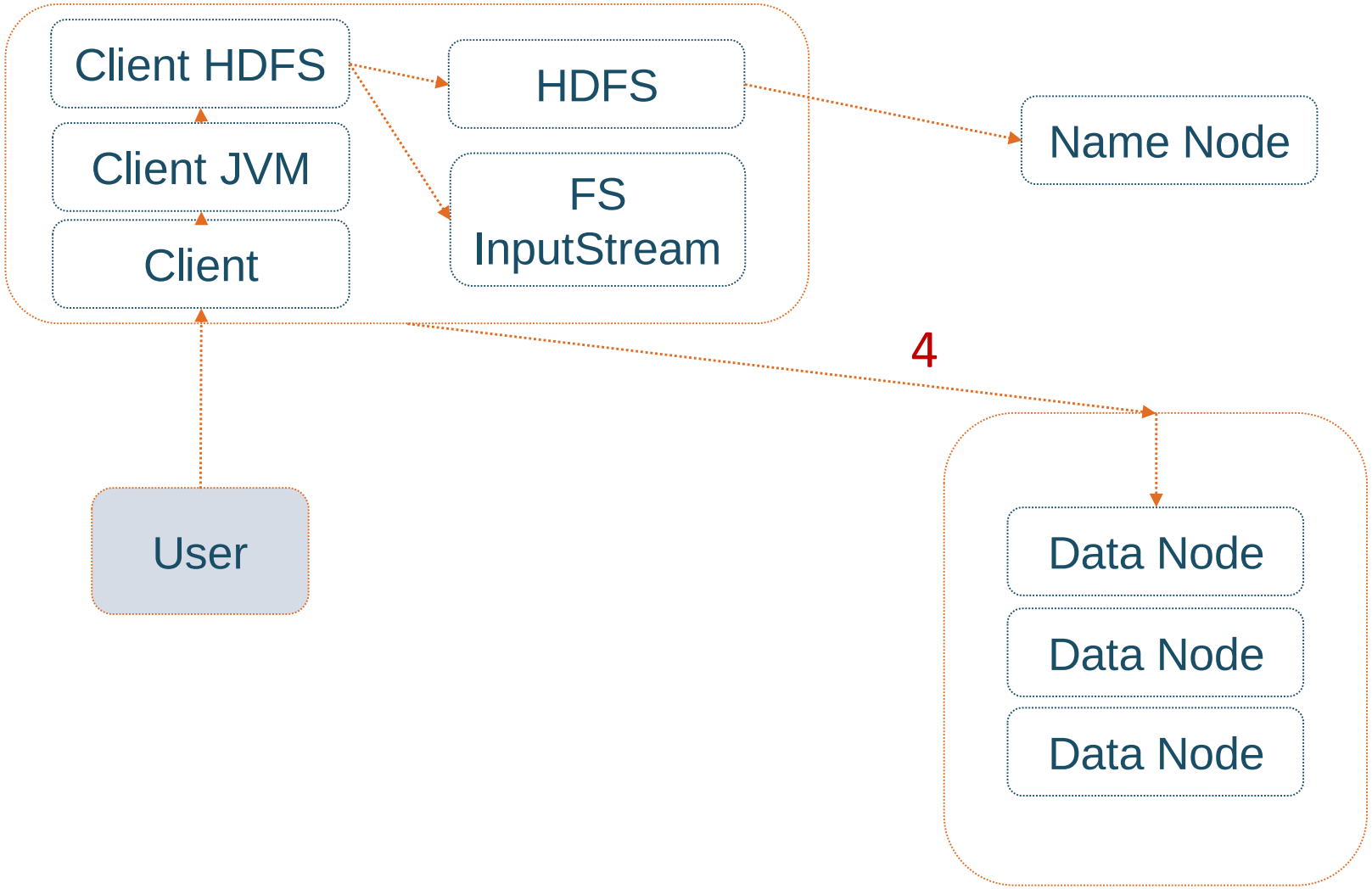


MAP – REDUCE | WRITE HDFS

READ

4
Запись файла:
- разделение на блоки
- разделениена ноды

DataNode Pipeline

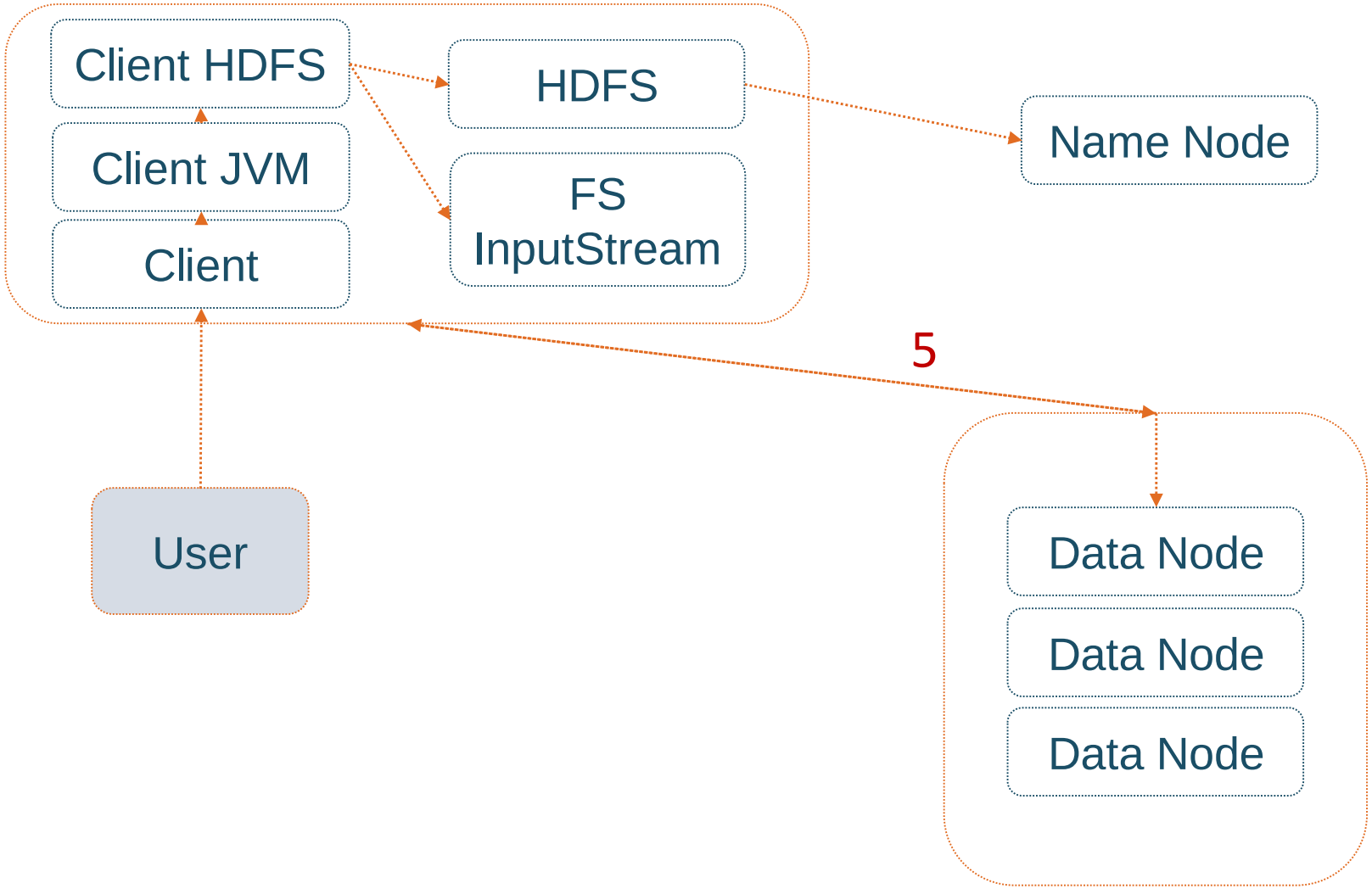


MAP – REDUCE | WRITE HDFS

READ

5
Знания о успехе

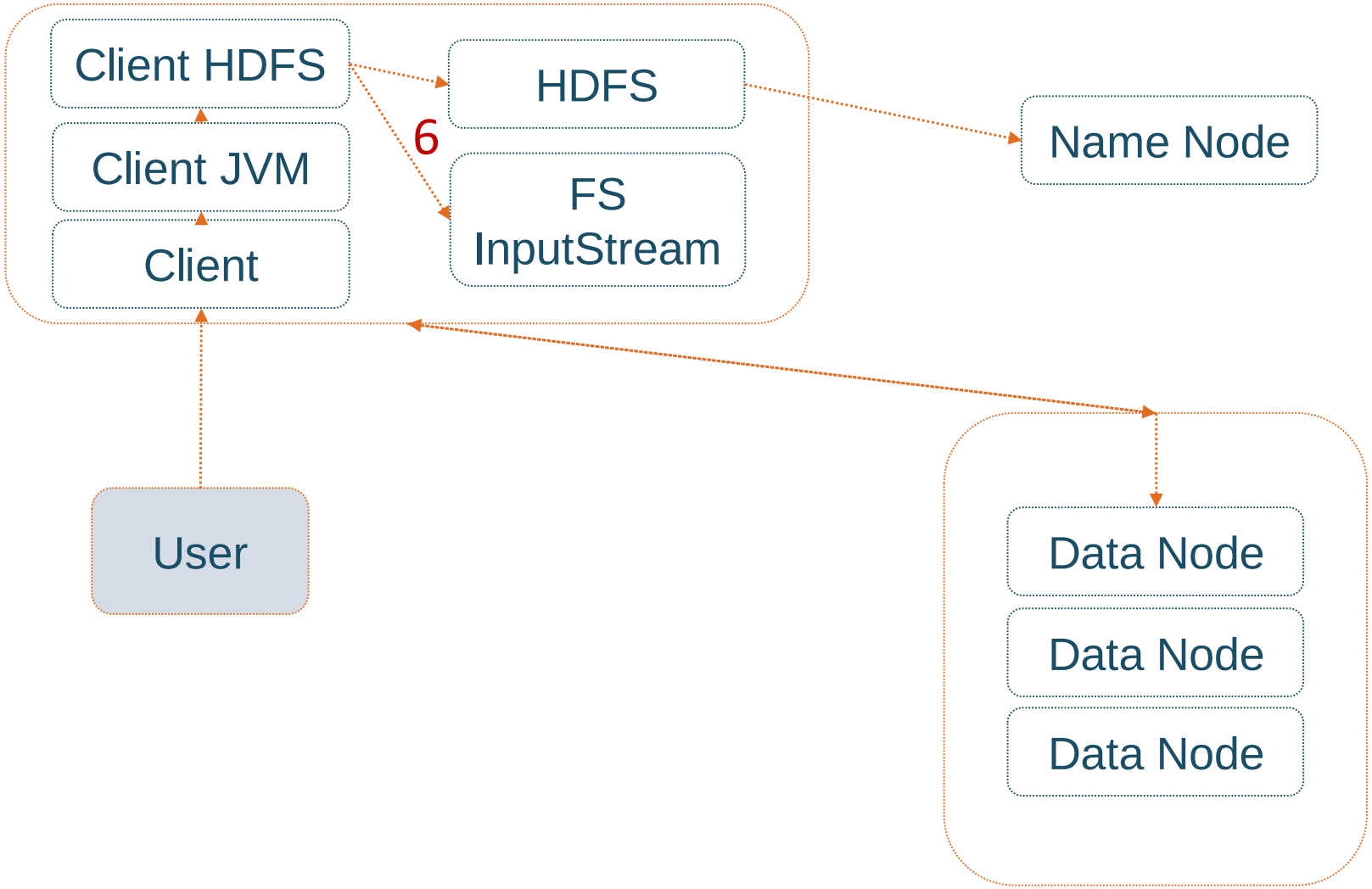
__success



MAP – REDUCE | WRITE HDFS

READ

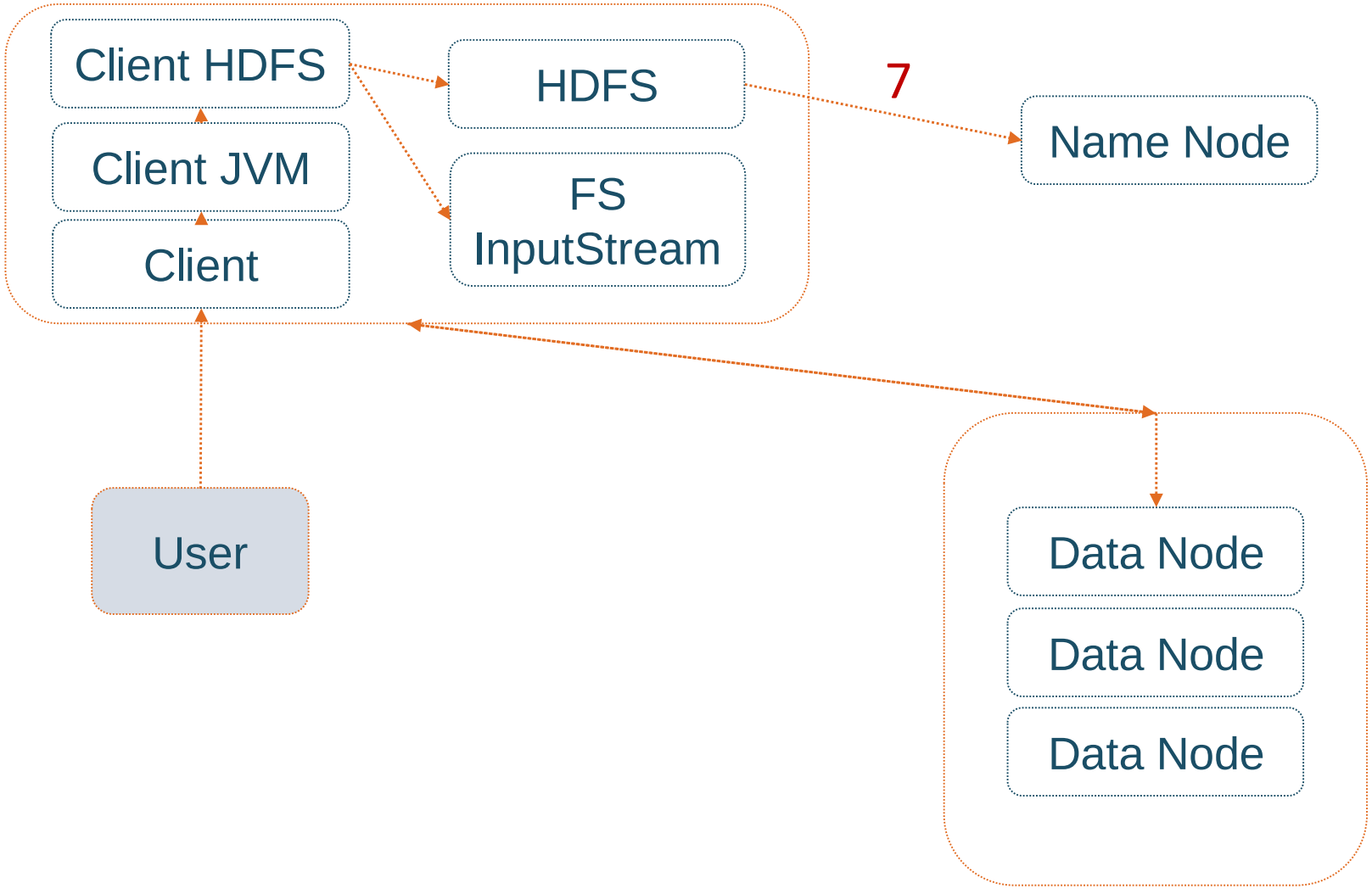
6
Закрытие операций



MAP – REDUCE | WRITE HDFS

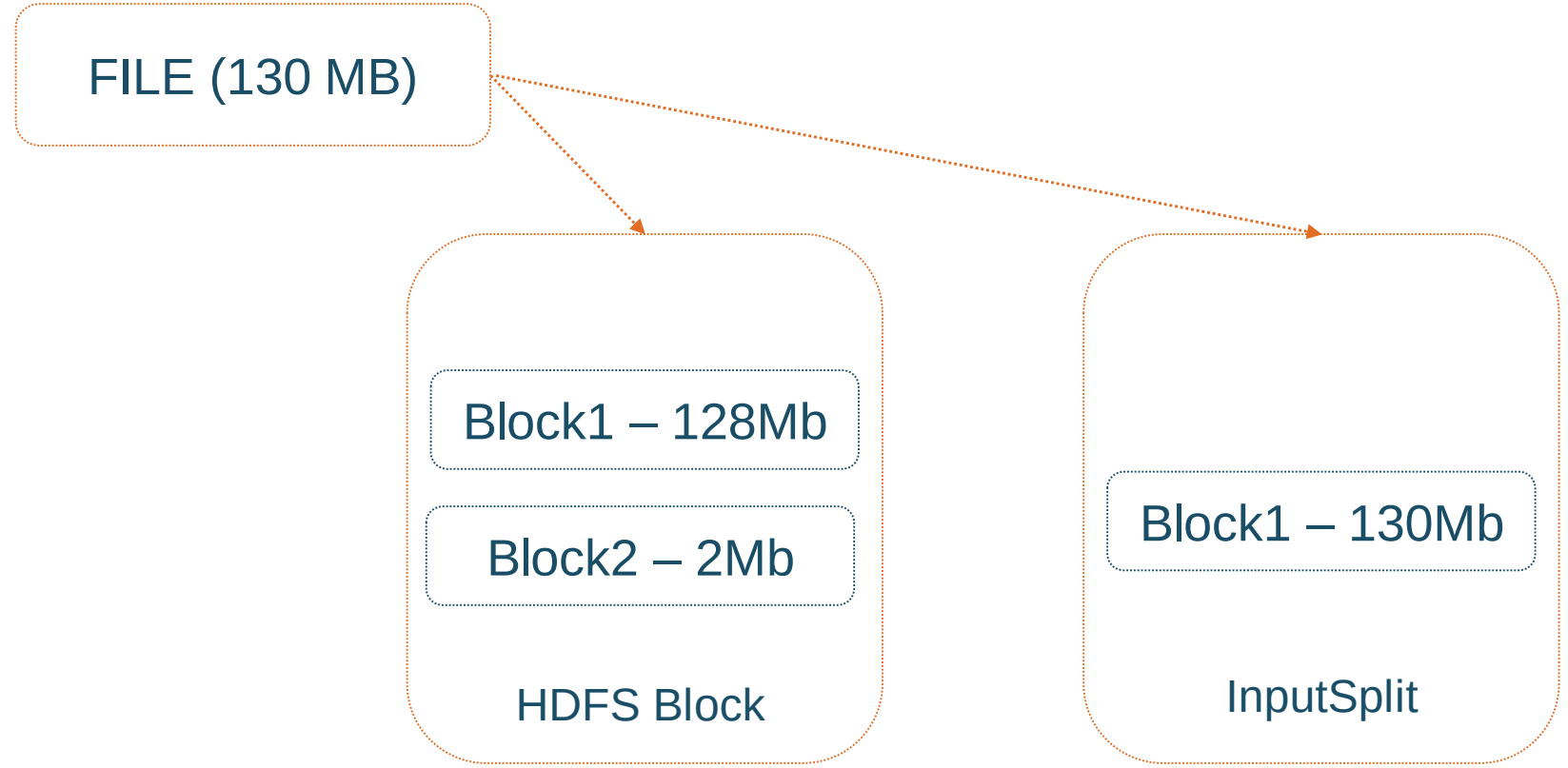
READ

7
Запись знаний
о новых данных



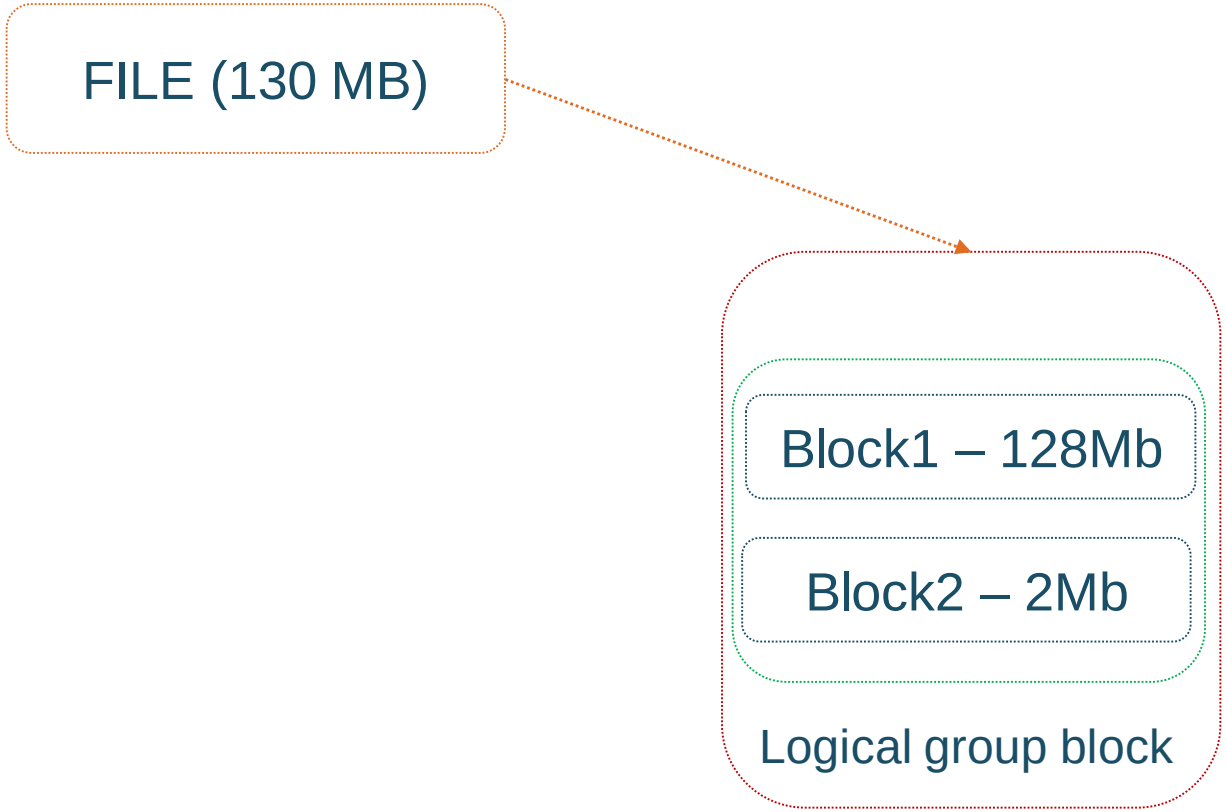
MAP – REDUCE | HDFS BLOCKS

InputSplit != HDFS Block



MAP – REDUCE | HDFS BLOCKS

InputSplit != HDFS Block



ПОПРОБУЕМ САМОСТОЯТЕЛЬНО

ДОМАШНЯЯ РАБОТА

MAR-REDUCE

- **t1_mapr. Используя библиотеку MRJOB:**

  - Исследуйте данные и определите ключ для join двух наборов данных

  - Сделайте join двух наборов

  - Определите самый популярный почтовый домен у пользователей

  - Определите куда больше всего транзачат

  - Определите популярность (топ 3):

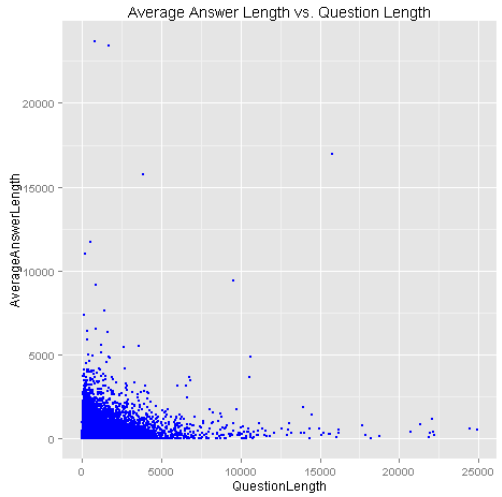
  по стране отправителя,

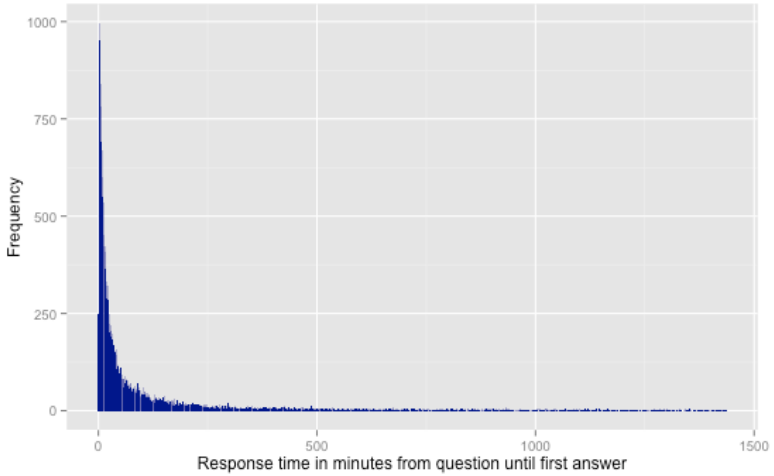
  по связке страна-домен,

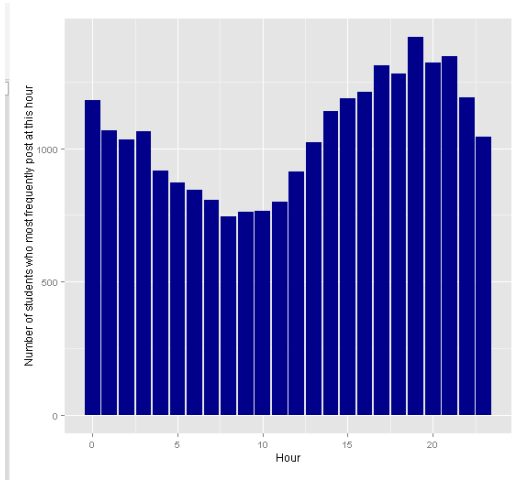
  по связке страна-транзакция

- **t2_mapreduce_viz. Используя Hadoop map-reduce извлеките данные и визуализируйте результат вычислений (пример далее)**

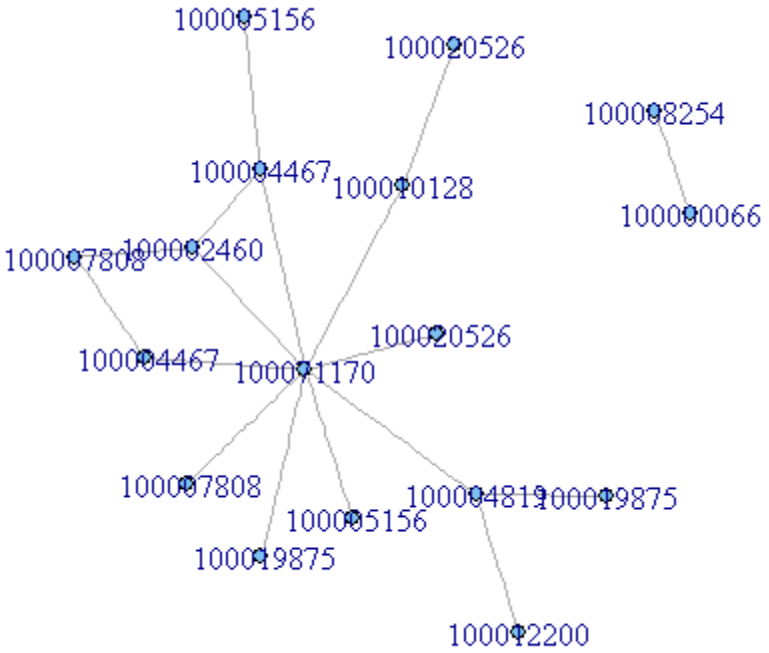
MAR-REDUCE


Average Answer Length vs. Question Length
AverageAnswerLength
QuestionLength
0
5000
10000
15000
20000
25000


Frequency
Response time in minutes from question until first answer
0
250
500
750
1000
1500


Number of students who most frequently post at this hour
Hour
0
500
1000
5
10
15
20


cs101
cs253
st101
bug
cs212
homework
meta
cs262
cs373
discussion


100005156
100020526
100008254
100004467
100010128
100000066
100007808
100002460
100020526
100004467
100001170
100007808
100005156
100019875
100012200