



# Работа с большими данными

SELEZNEV ARTEM  
HEAD OF DATA SCIENCE @ SBER



tg: @SeleznevArtem

 /NameArtem

 /seleznev-artem

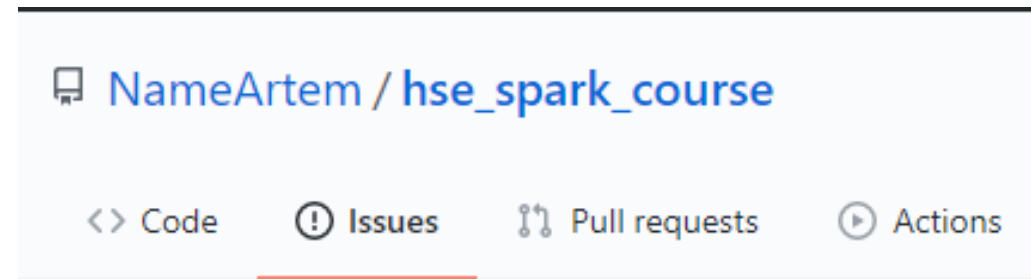
 /seleznev.artem.info



[https://github.com/NameArtem/hse\\_spark\\_course](https://github.com/NameArtem/hse_spark_course)



[https://github.com/NameArtem/hse\\_spark\\_course](https://github.com/NameArtem/hse_spark_course)



O KYPCE

№	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. Управлении кластерами

# ИНСТРУМЕНТЫ

Python

---

Linux

---

Git

---

Hadoop

---

Spark

---

# ИНСТРУМЕНТЫ

Python

---

Linux

---

Git

---

Hadoop

---

Spark

---



# BIG DATA?

# BIG DATA



Детализация заказа  
в магазине?

Все заказы по всем  
магазинам?

Портфель акций  
одного инвестора?

Все транзакции по  
всем акциям?

Детализация заказа  
в магазине?

Портфель акций  
одного инвестора?

Все заказы по всем  
магазинам?

Все транзакции по  
всем акциям?

# DATA БРОСАЕТ ВЫЗОВ

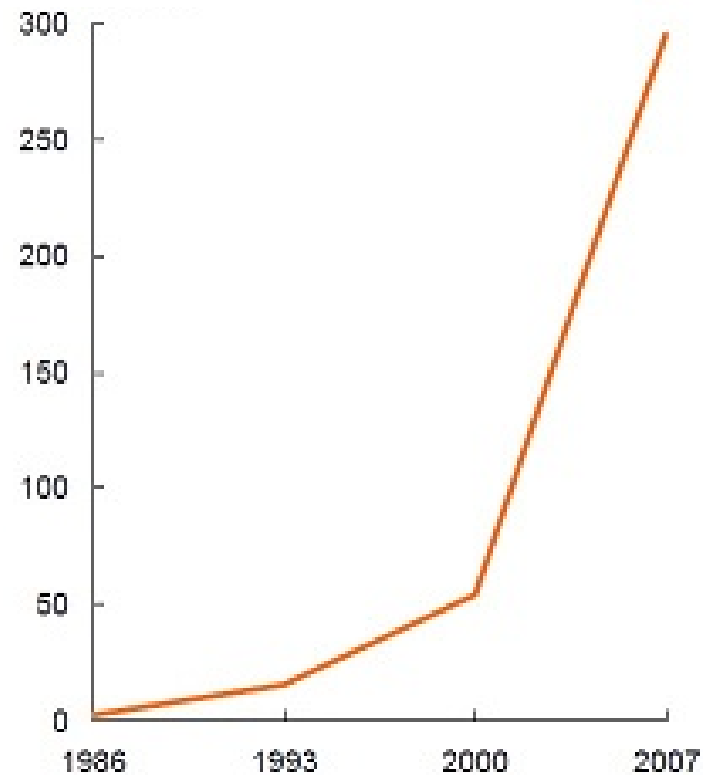
Данные создаются  
очень быстро

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



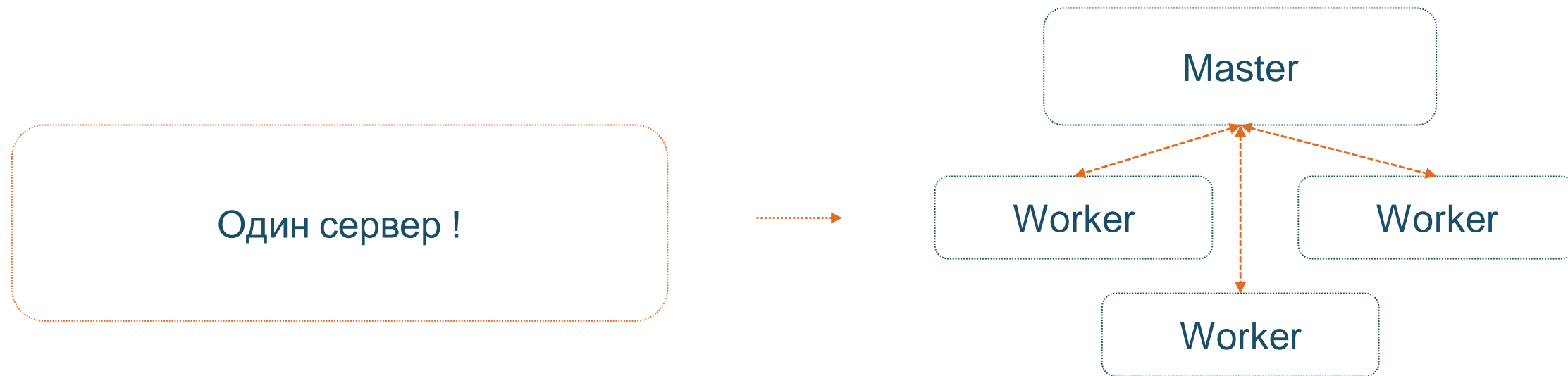
КЛАСТЕР



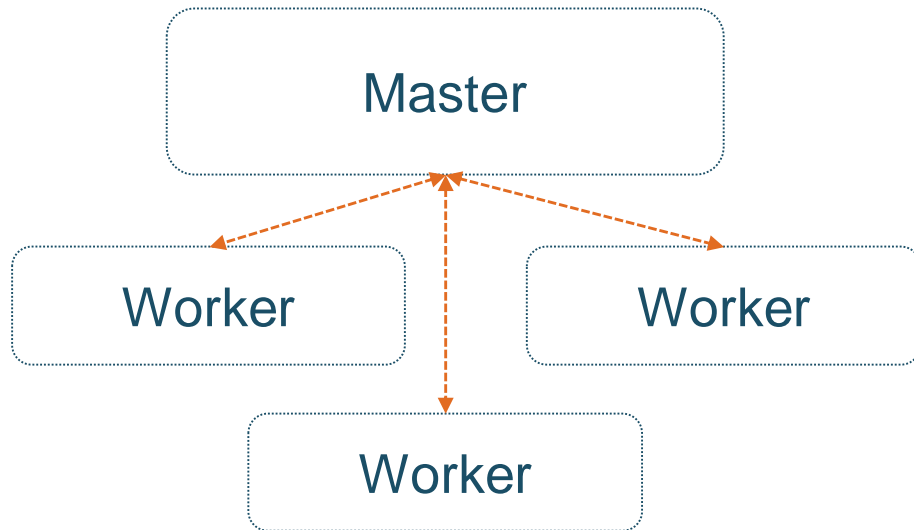
# ПОЯВИЛСЯ КЛАСТЕР

Один сервер !

# ПОЯВИЛСЯ КЛАСТЕР

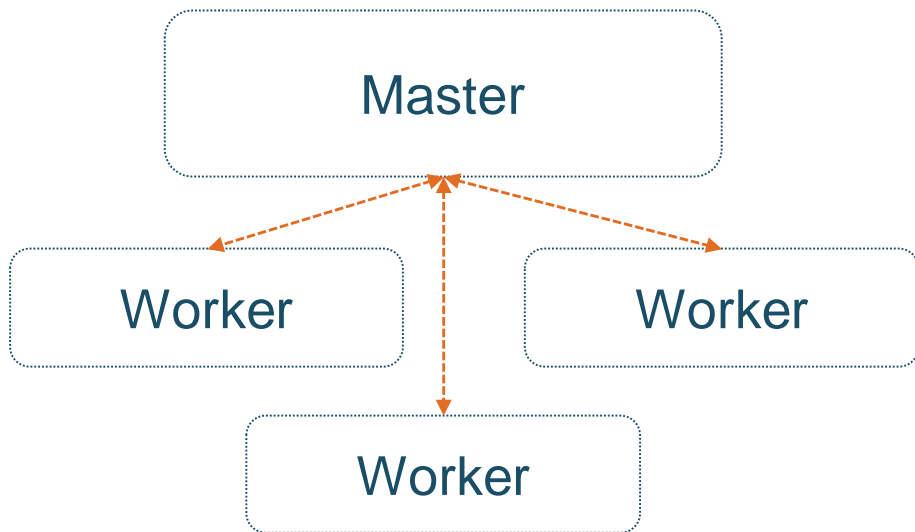


# ПОЯВИЛСЯ КЛАСТЕР и добавил проблем



Проблемы  
координации

# ПОЯВИЛСЯ КЛАСТЕР и добавил проблем

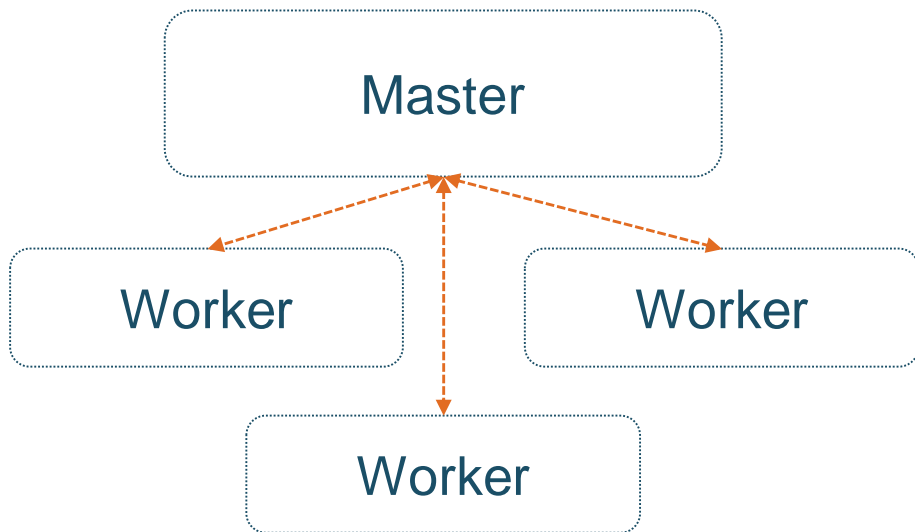


Проблемы  
координации

Проблемы  
коммуникации



# ПОЯВИЛСЯ КЛАСТЕР и добавил проблем

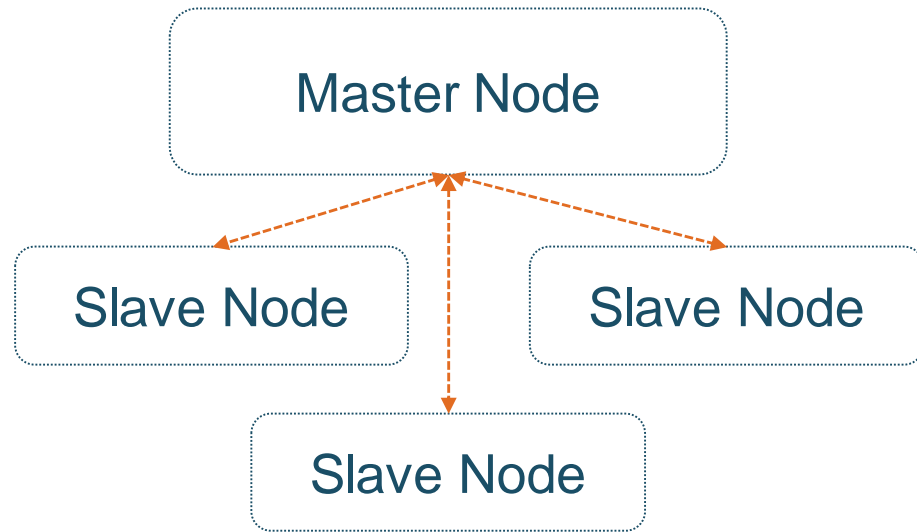


Проблемы  
координации

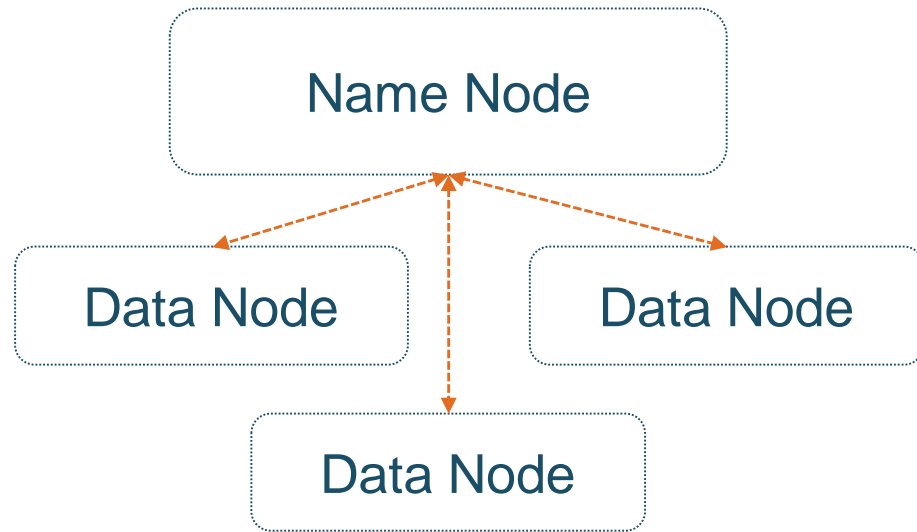
Проблемы  
коммуникации

Проблемы  
стабильности

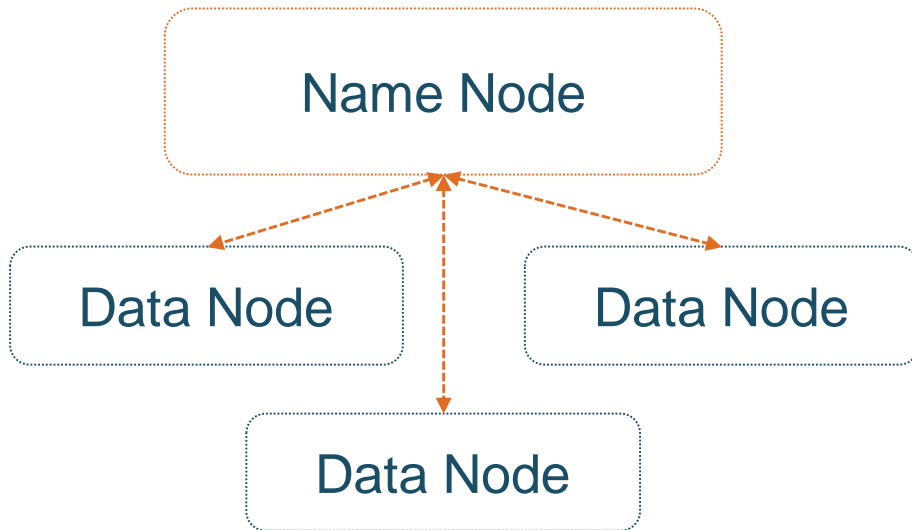
# HDFS



# HDFS



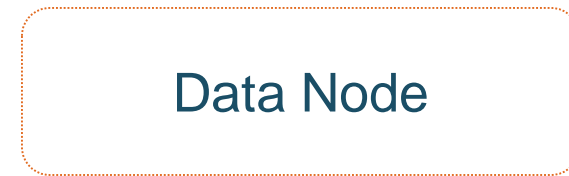
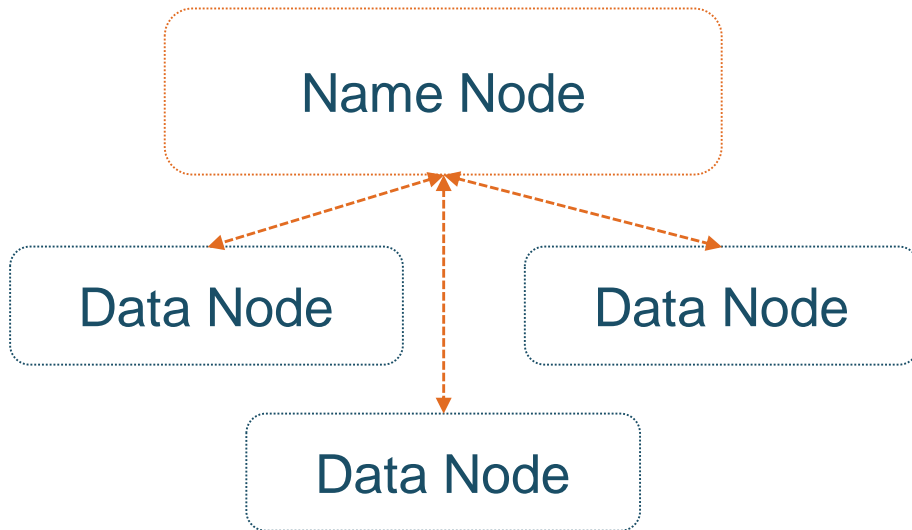
# HDFS



Name Node

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

# HDFS



- Хранят и обрабатывают данные

# HDFS

ОСОБЕННОСТЬ  
ХРАНЕНИЯ

Data Node

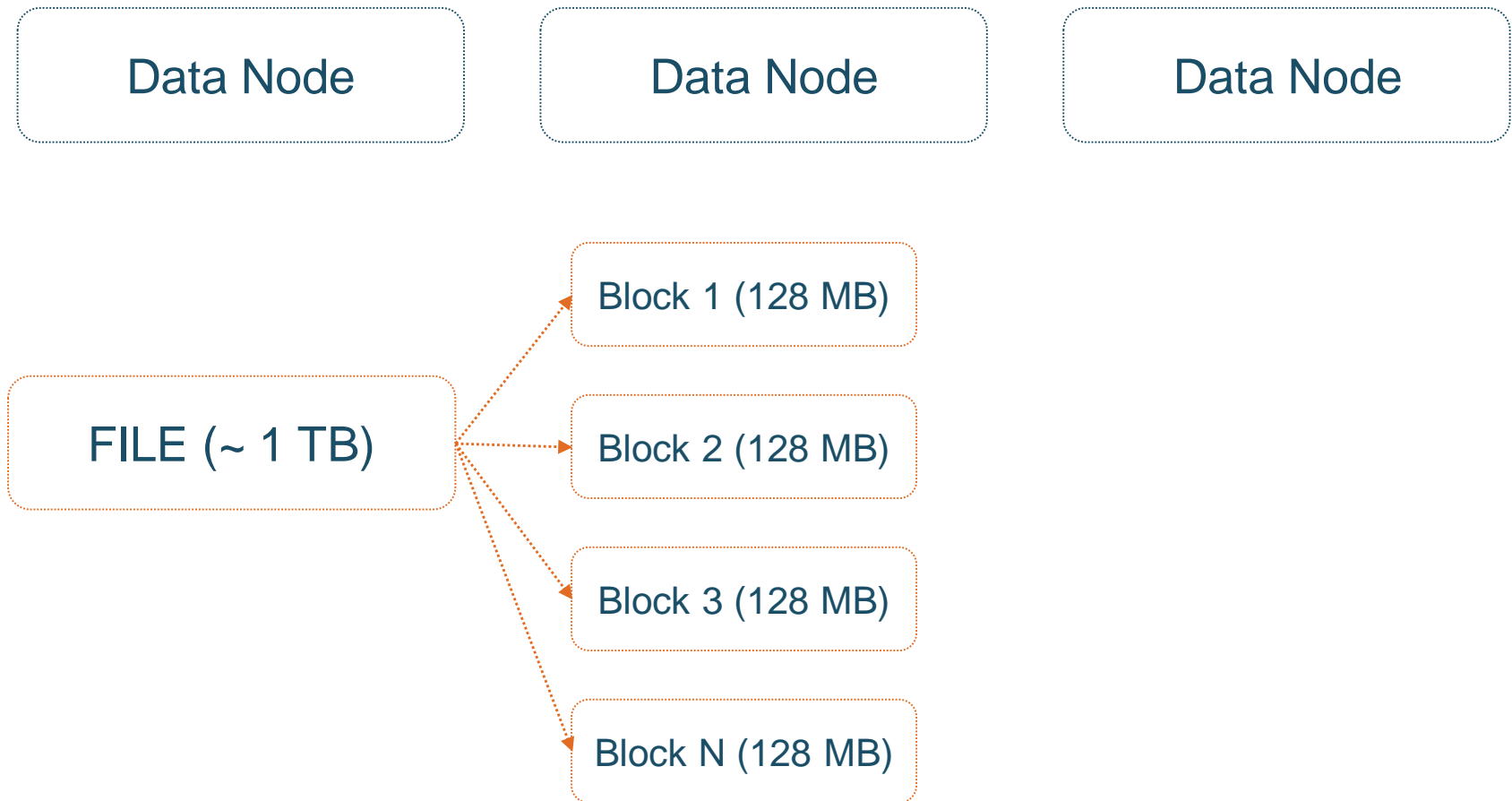
Data Node

Data Node

FILE (~ 1 TB)

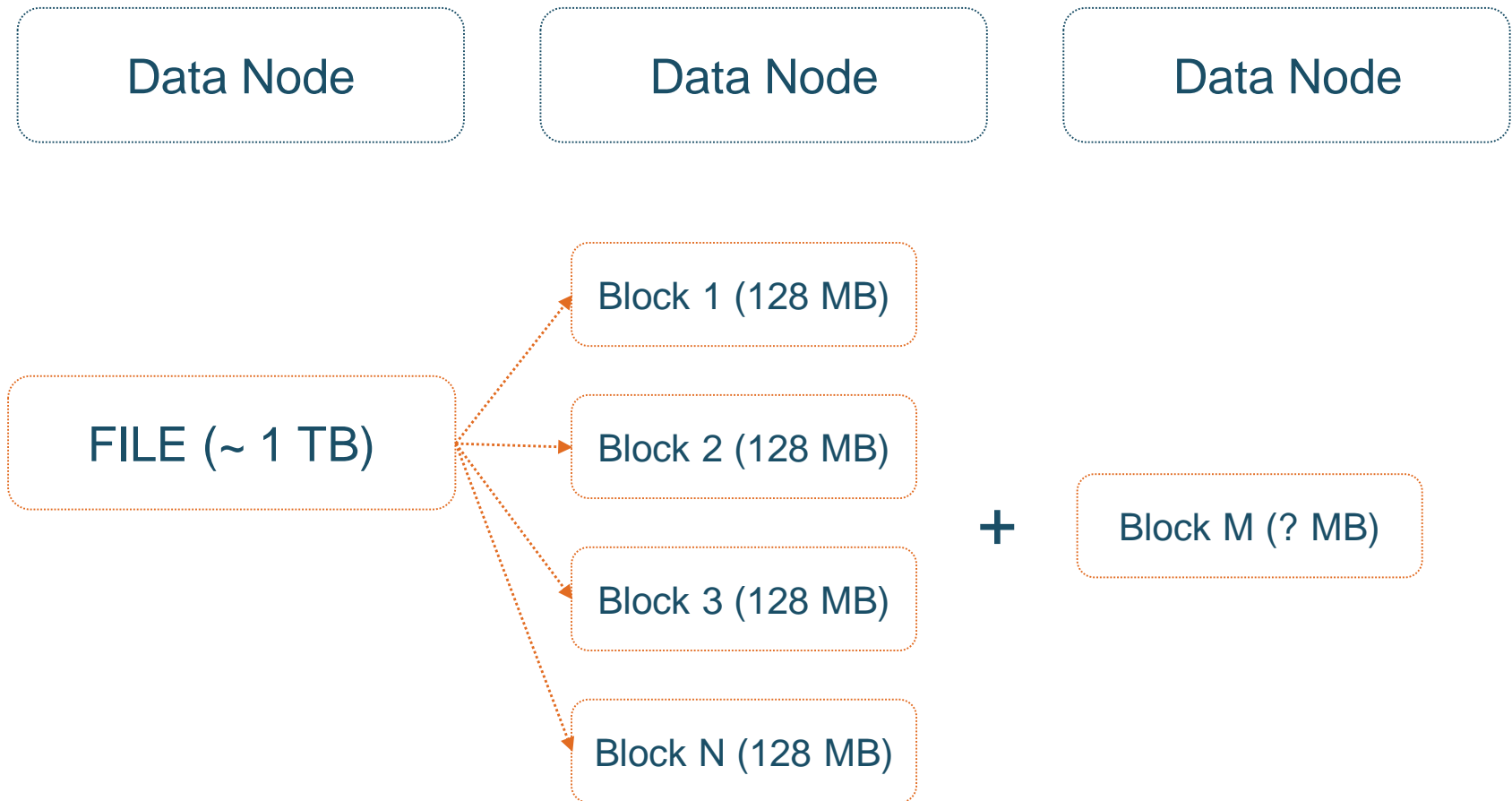
# HDFS

ОСОБЕННОСТЬ  
ХРАНЕНИЯ



# HDFS

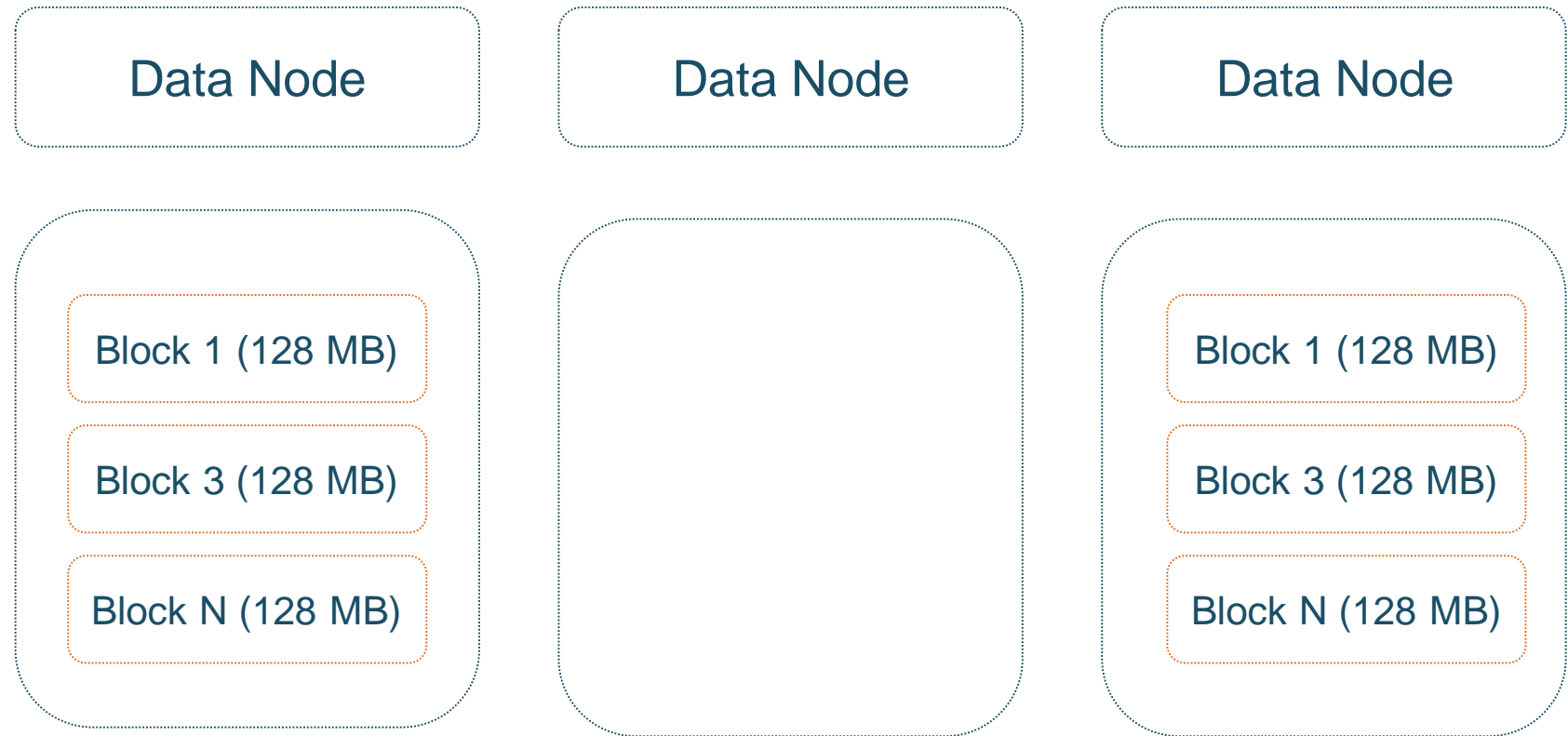
ОСОБЕННОСТЬ  
ХРАНЕНИЯ





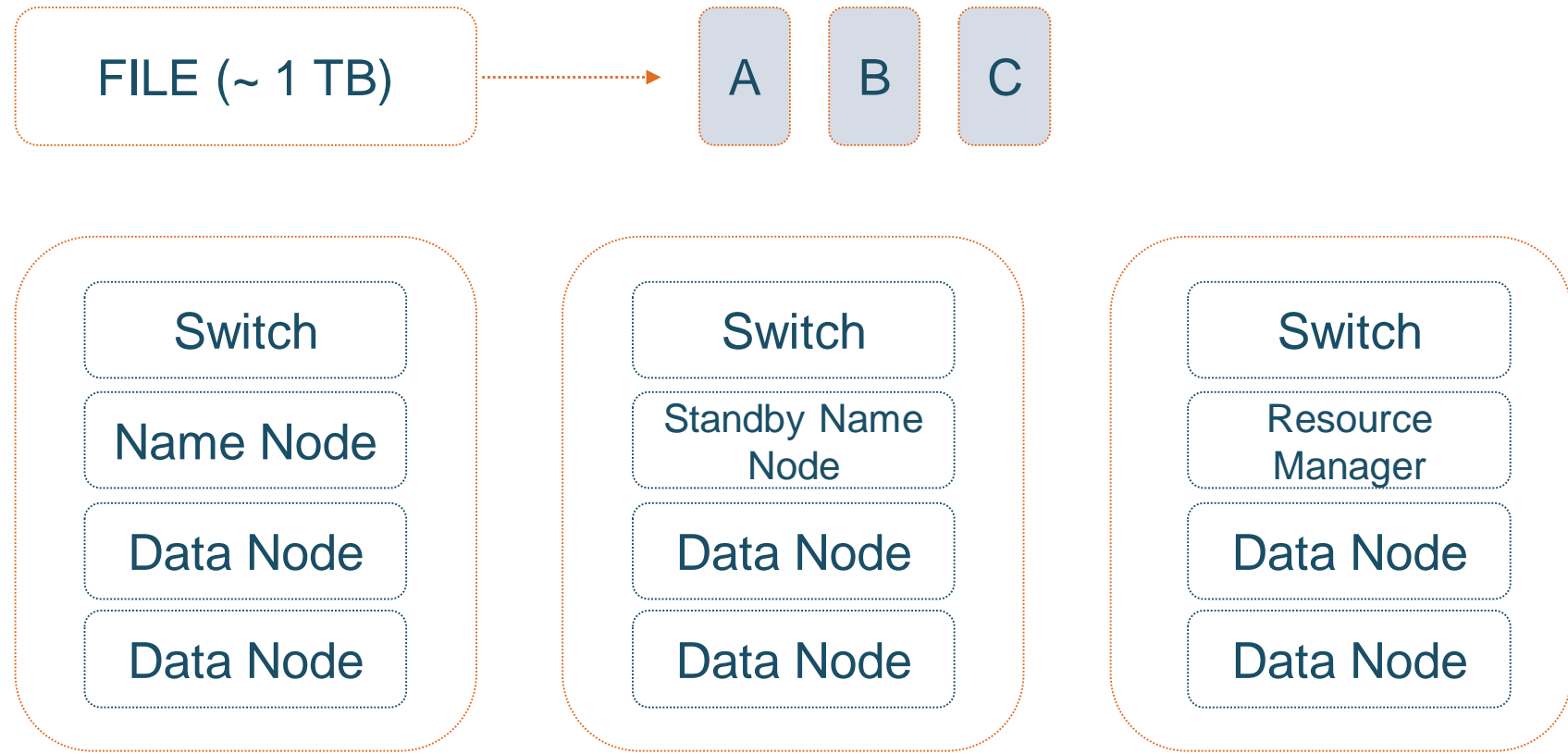
# HDFS

ОСОБЕННОСТЬ  
ХРАНЕНИЯ



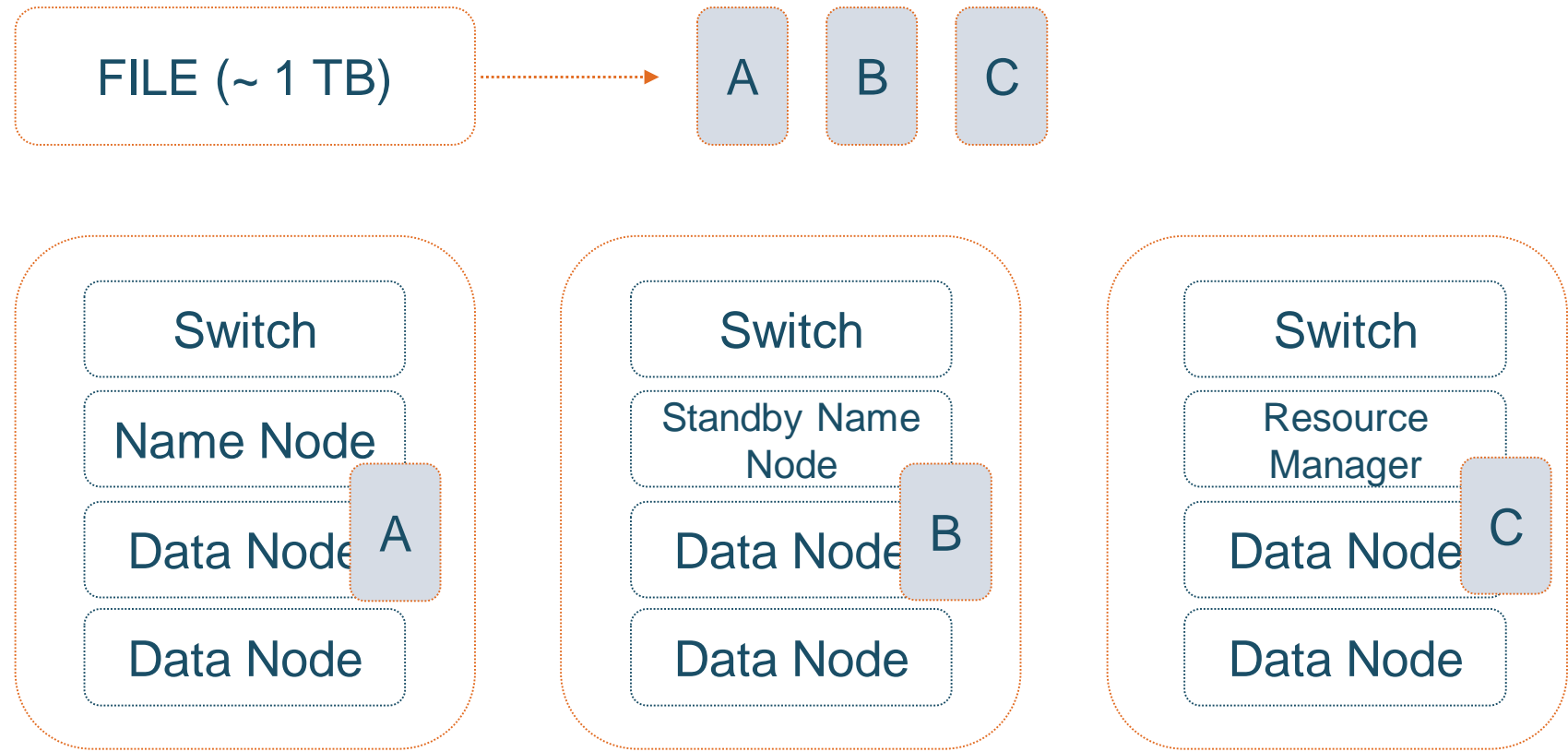
# HDFS – RACK AWARENESS

ОСОБЕННОСТЬ  
ХРАНЕНИЯ



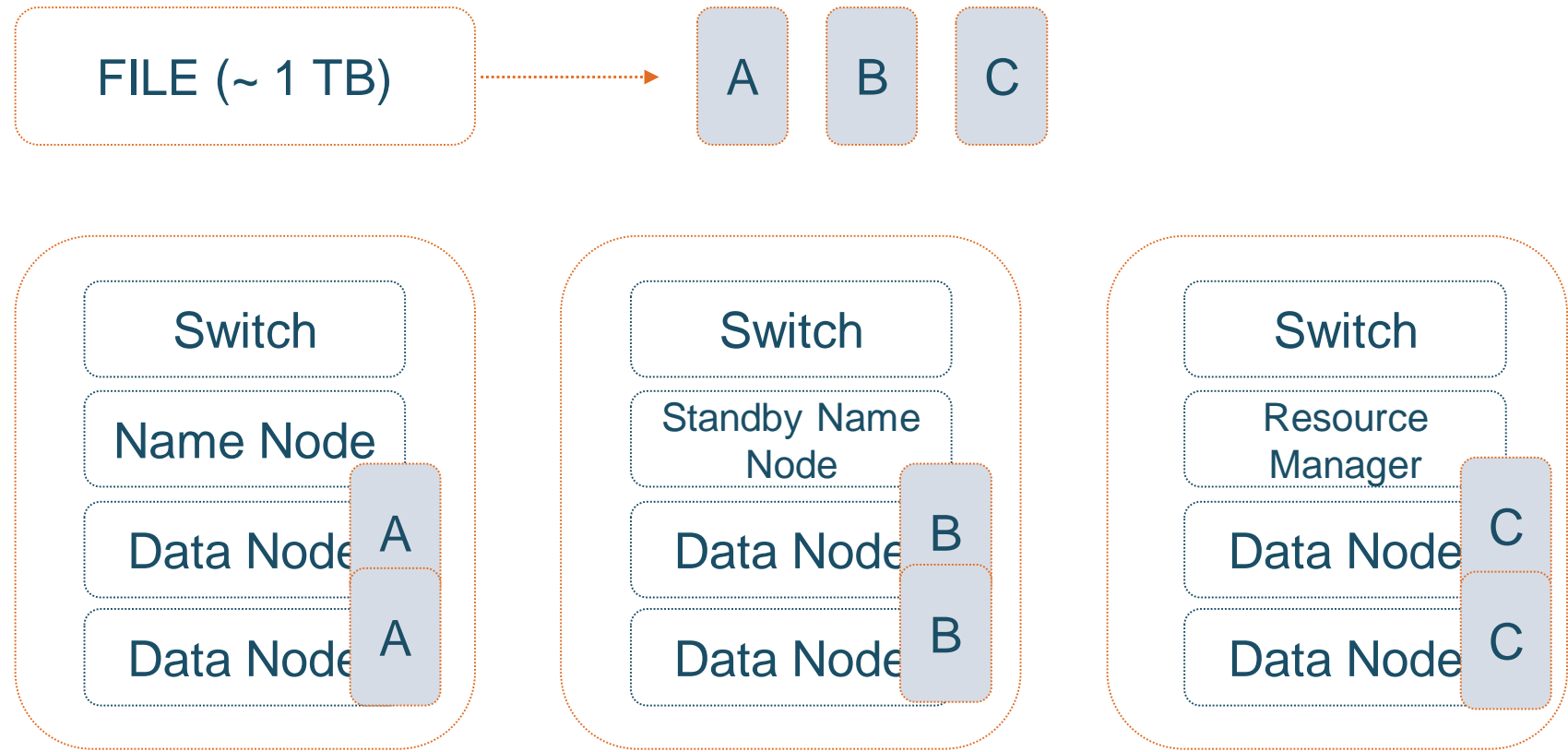
# HDFS – RACK AWARENESS

Rep.1  
—  
на разных узлах



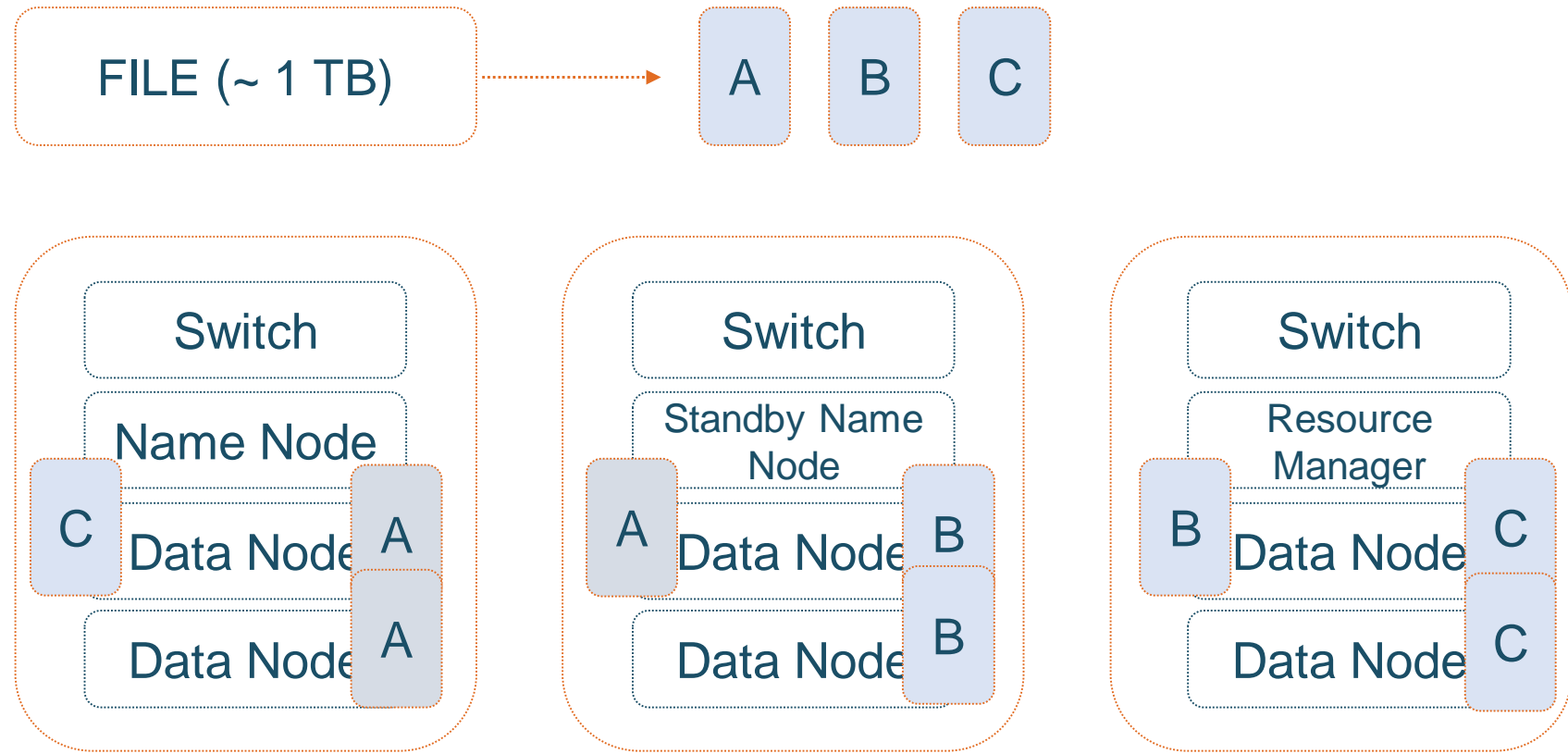
# HDFS – RACK AWARENESS

Rep.2  
—  
на разных нодах внутри



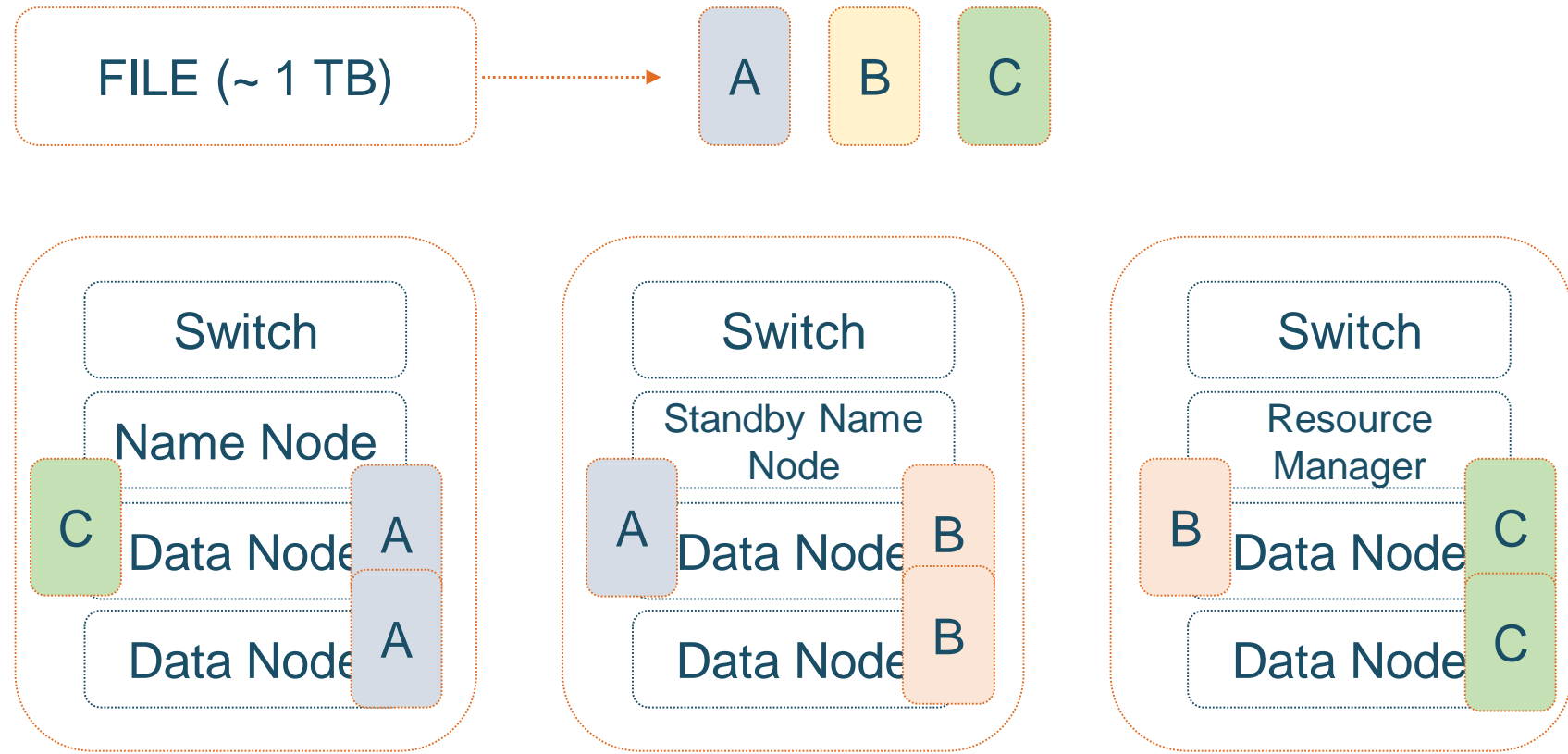
# HDFS – RACK AWARENESS

Rep.3  
—  
на разных стойках



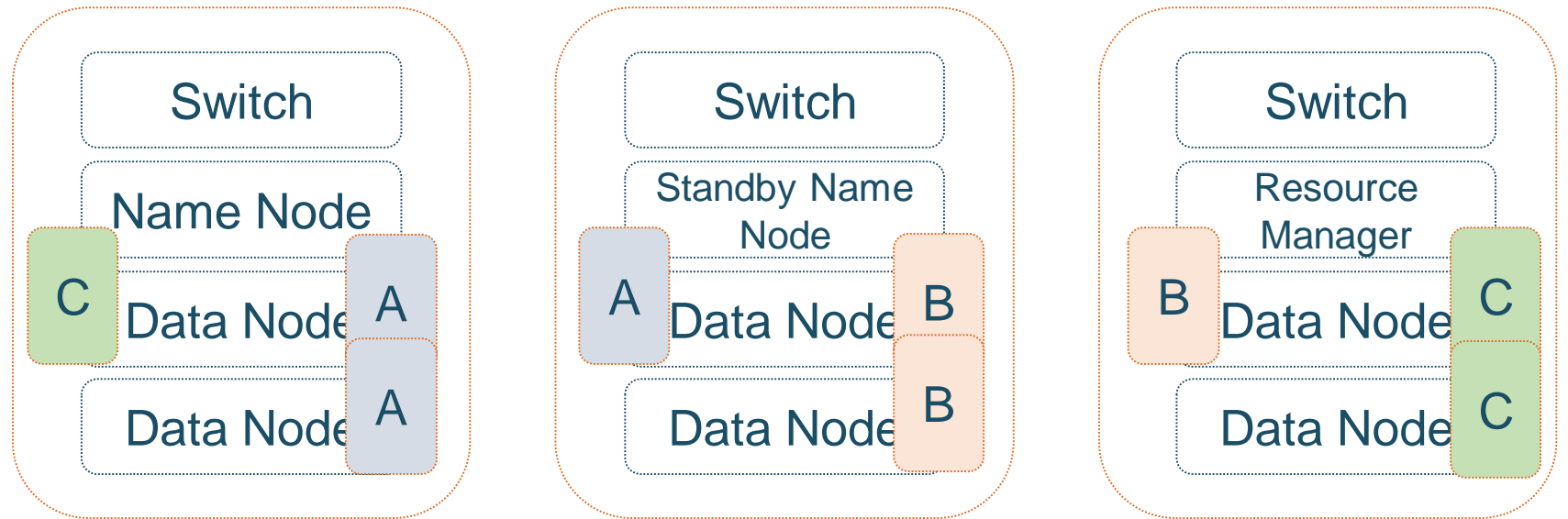
# HDFS – RACK AWARENESS

Rep.3  
—  
на разных стойках



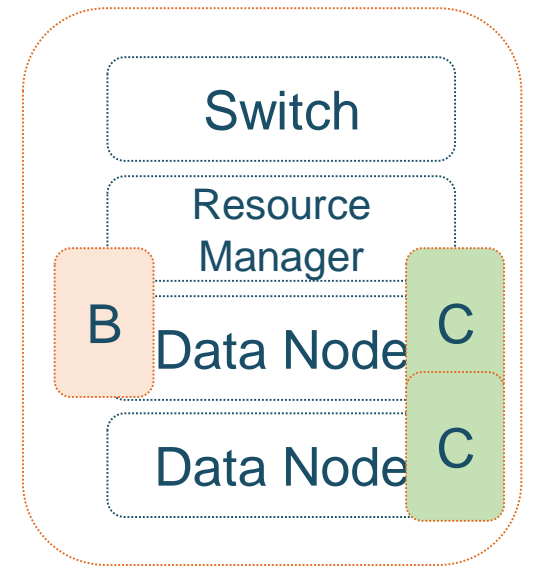
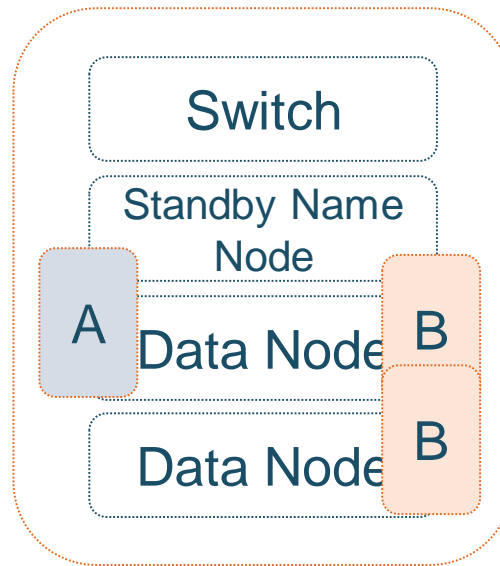
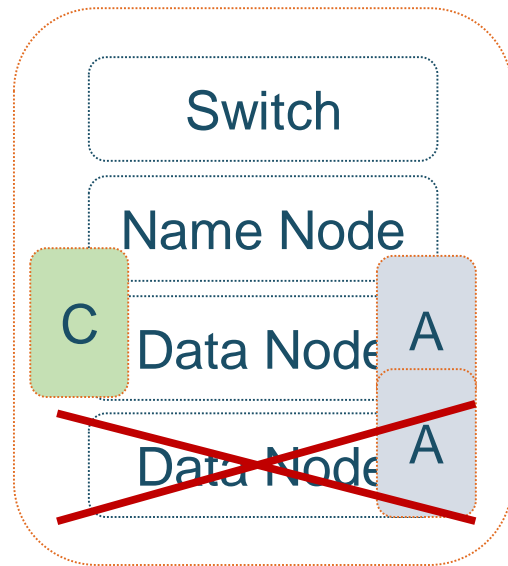
# HDFS – FAULT TOLERANCE

Данные  
доступны  
всегда



# HDFS – FAULT TOLERANCE

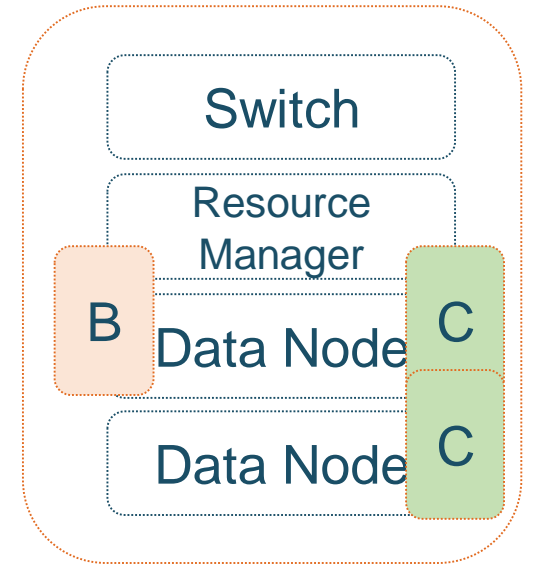
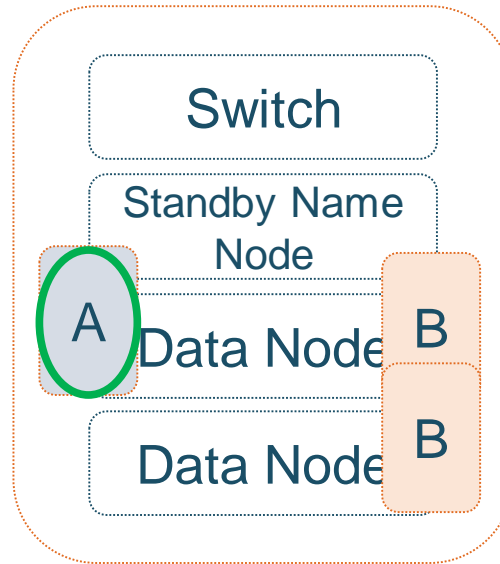
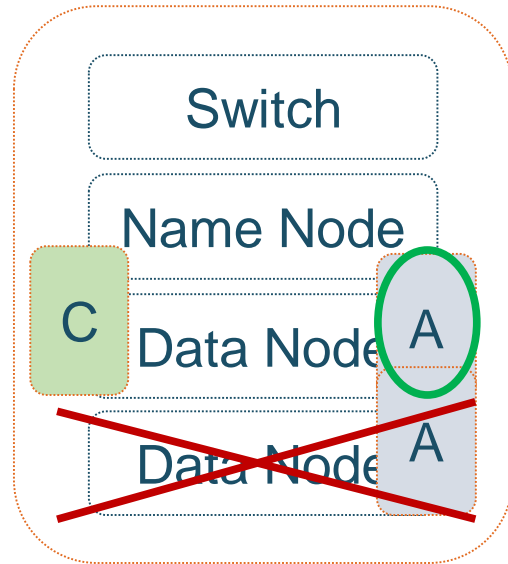
Данные  
доступны  
всегда





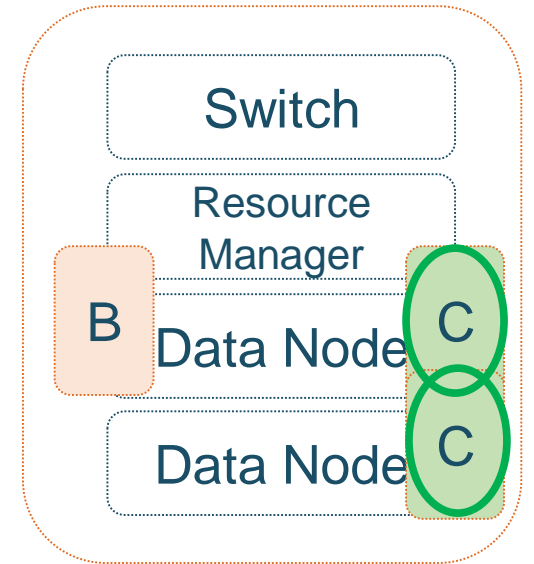
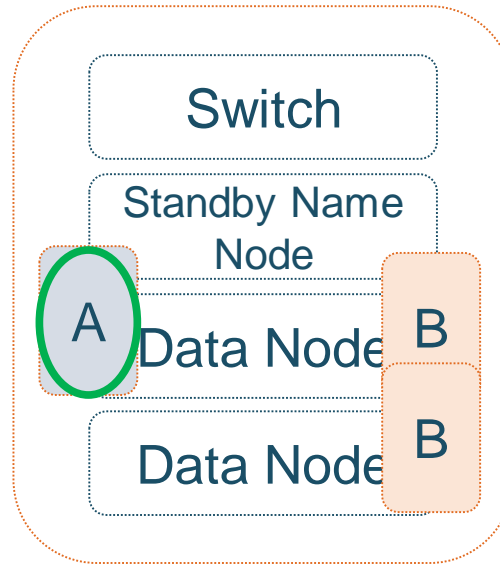
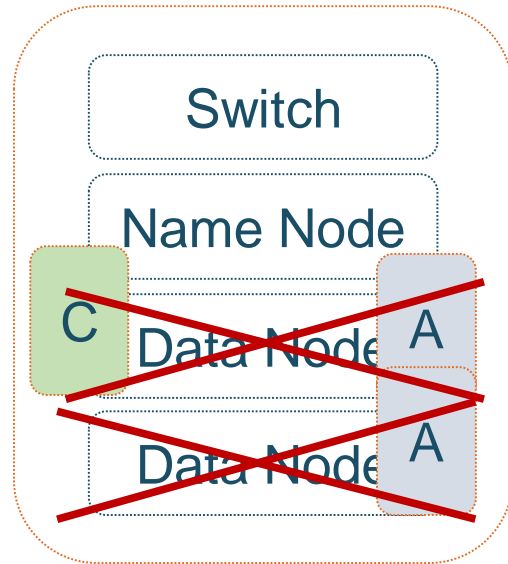
# HDFS – FAULT TOLERANCE

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 ==  $\text{sum}(\text{Disk } (1 \rightarrow N))$
- Total использ. = сумма использования всех дисков ==  $\text{sum}(\text{Disk } (1 \rightarrow N))$
- 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			

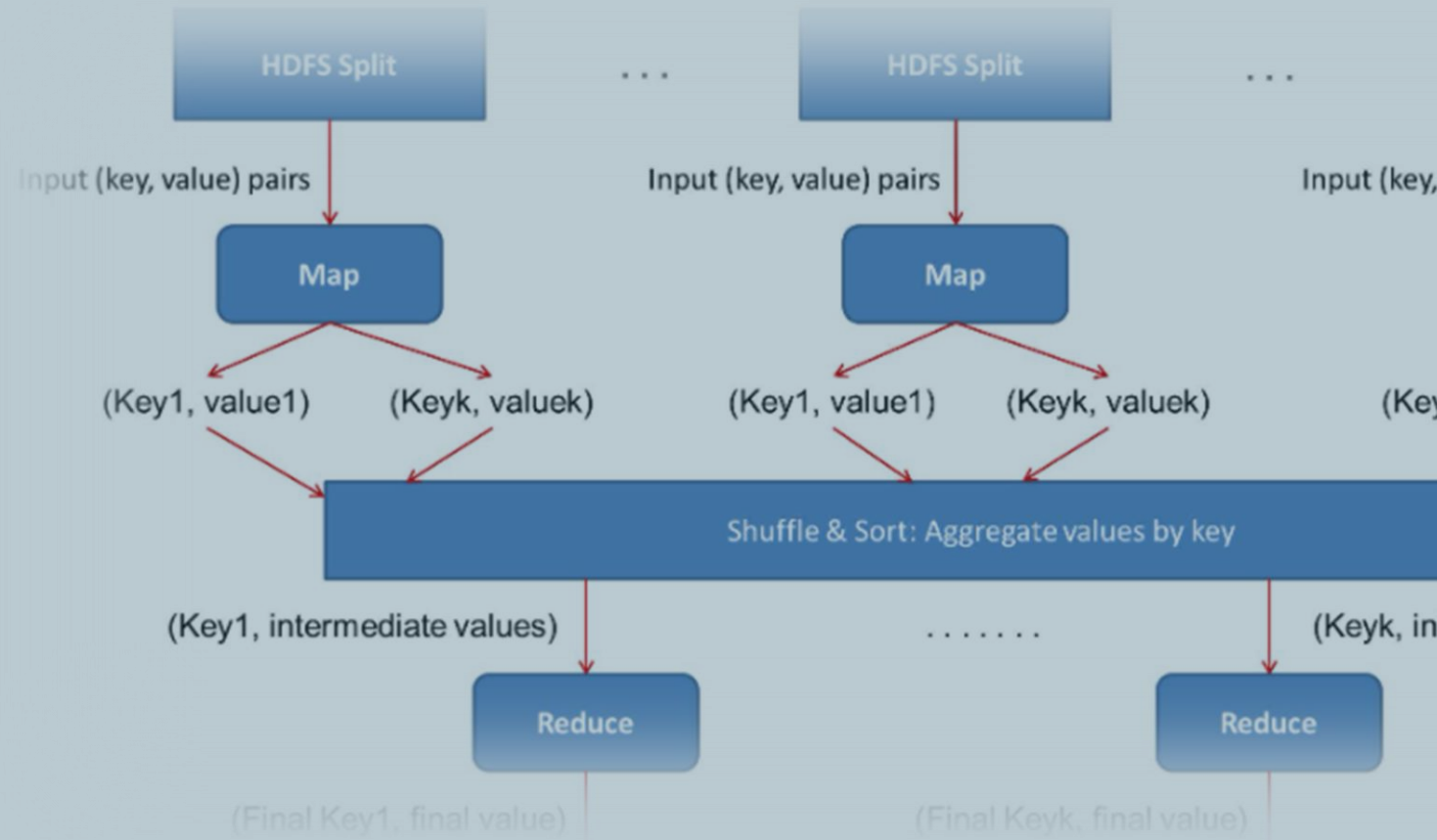
- $\% \text{ плотности} = \text{Ideal} - \% \text{ использования}$

# HDFS | БАЛАНСИРОВКА ДИСКОВ

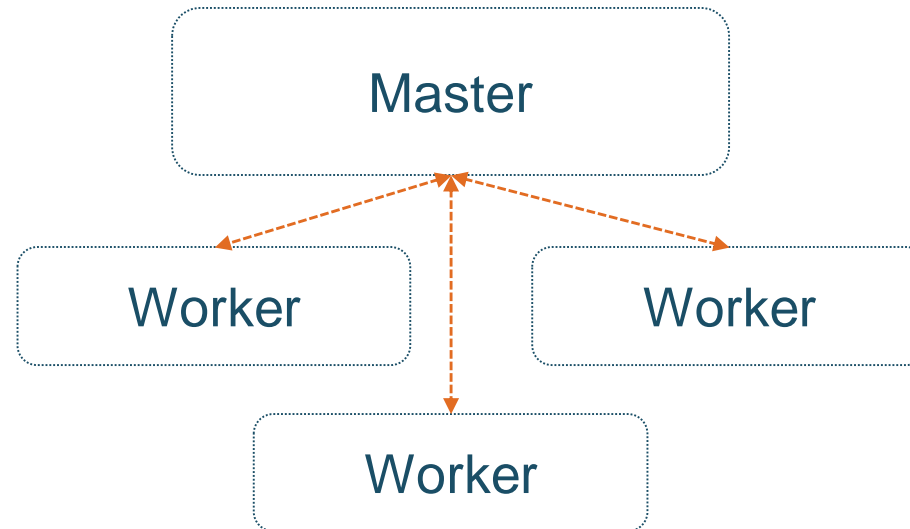
Нормализуем

- `hdfs diskbalancer – plan <datanode>`
- настраиваем `ednabled`, `out`, `thresholdPercentage`, `maxerror`
- проверяем отчеты:  
`hdfs diskbalancer –fs namenode.url –report file_path//`

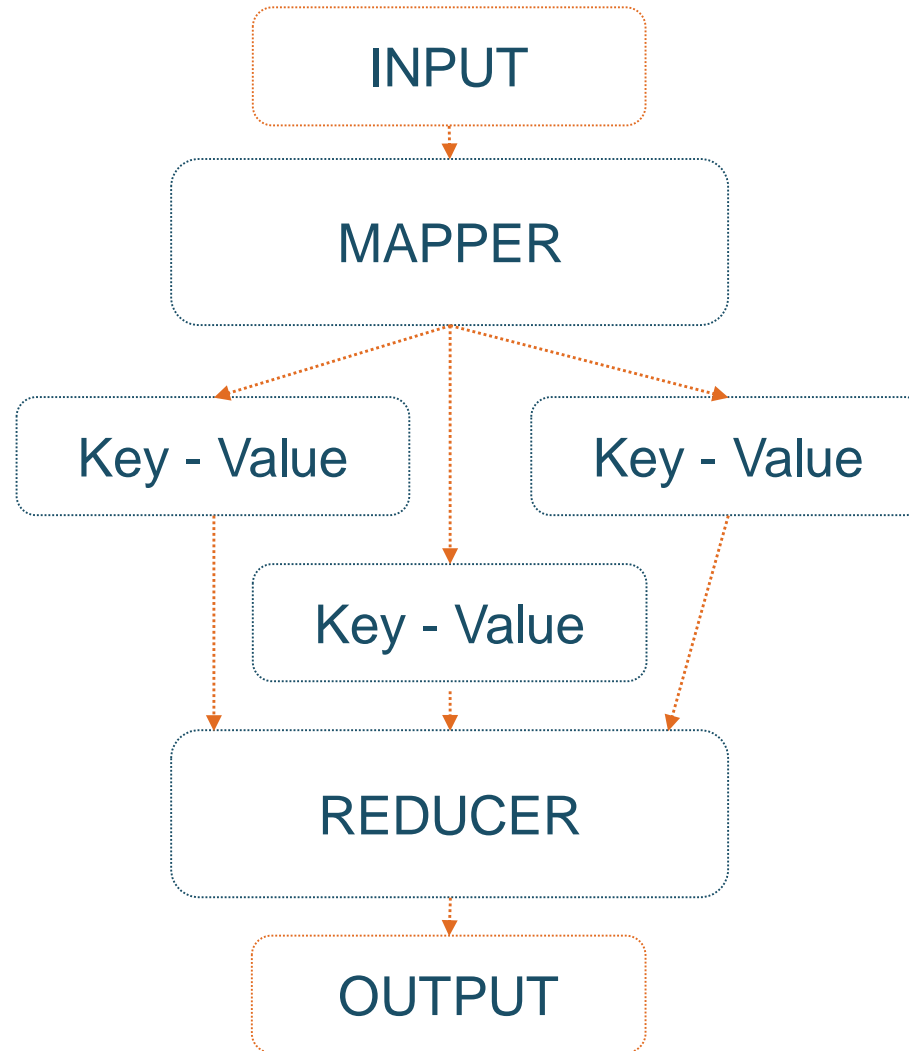
# MAP - REDUCE



# MAP – REDUCE

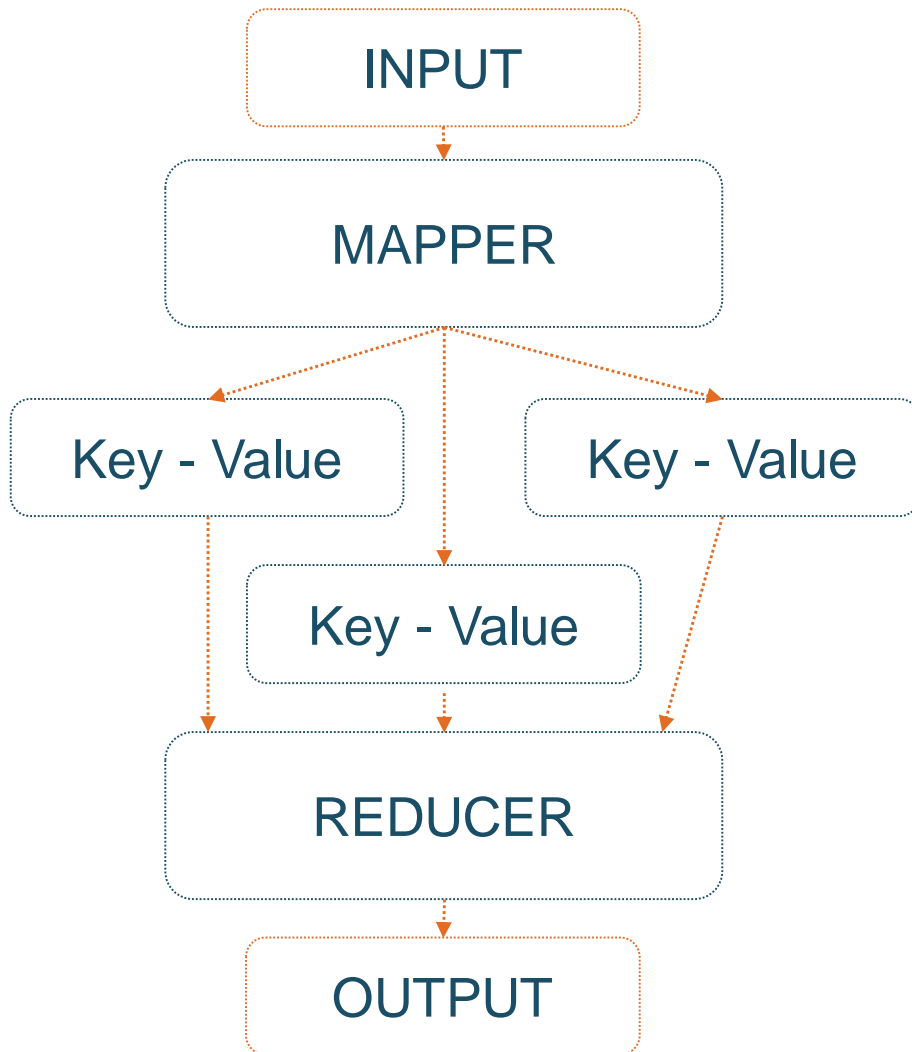


# MAP – REDUCE

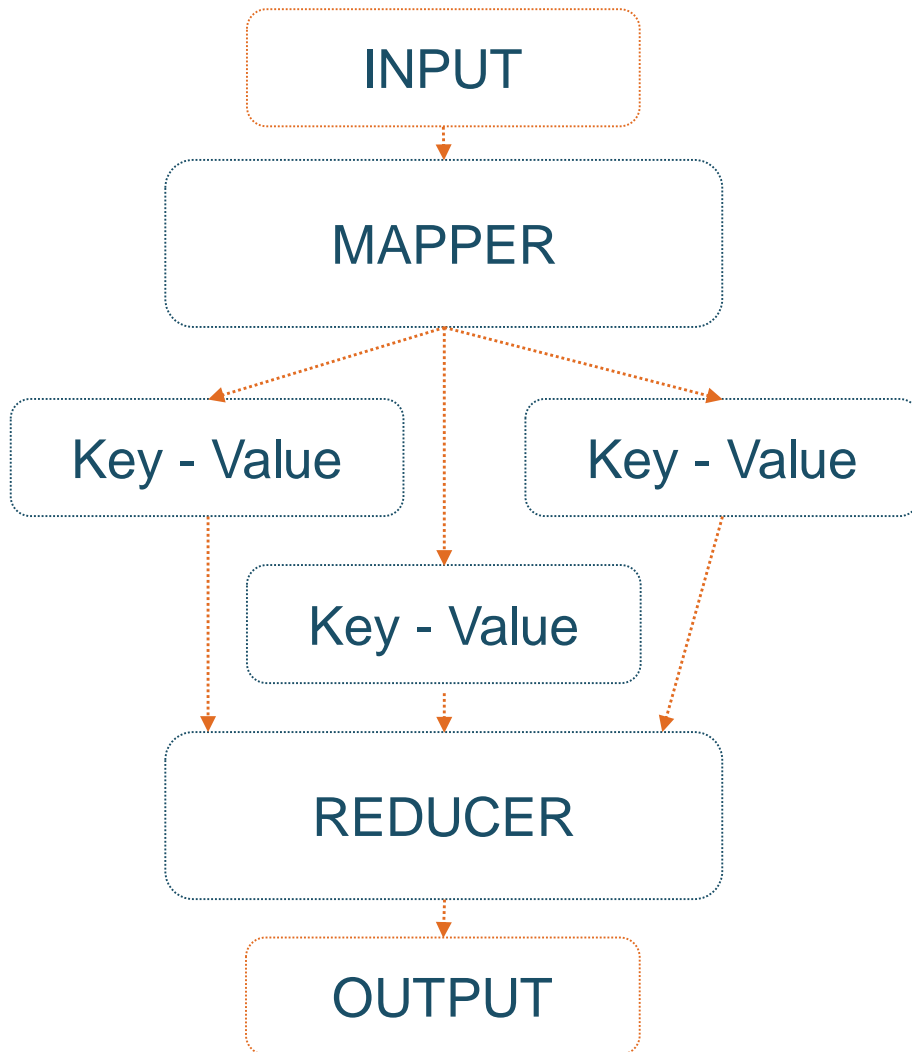




# MAP – REDUCE

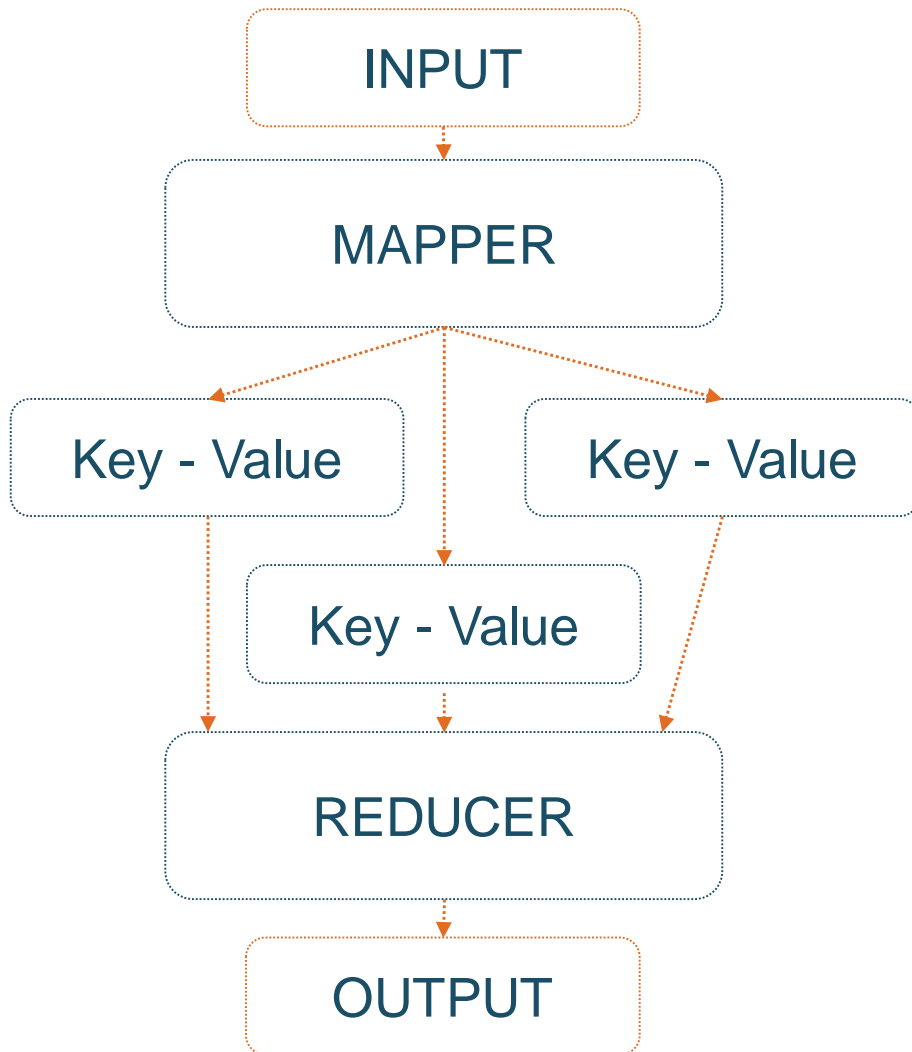


# MAP – REDUCE



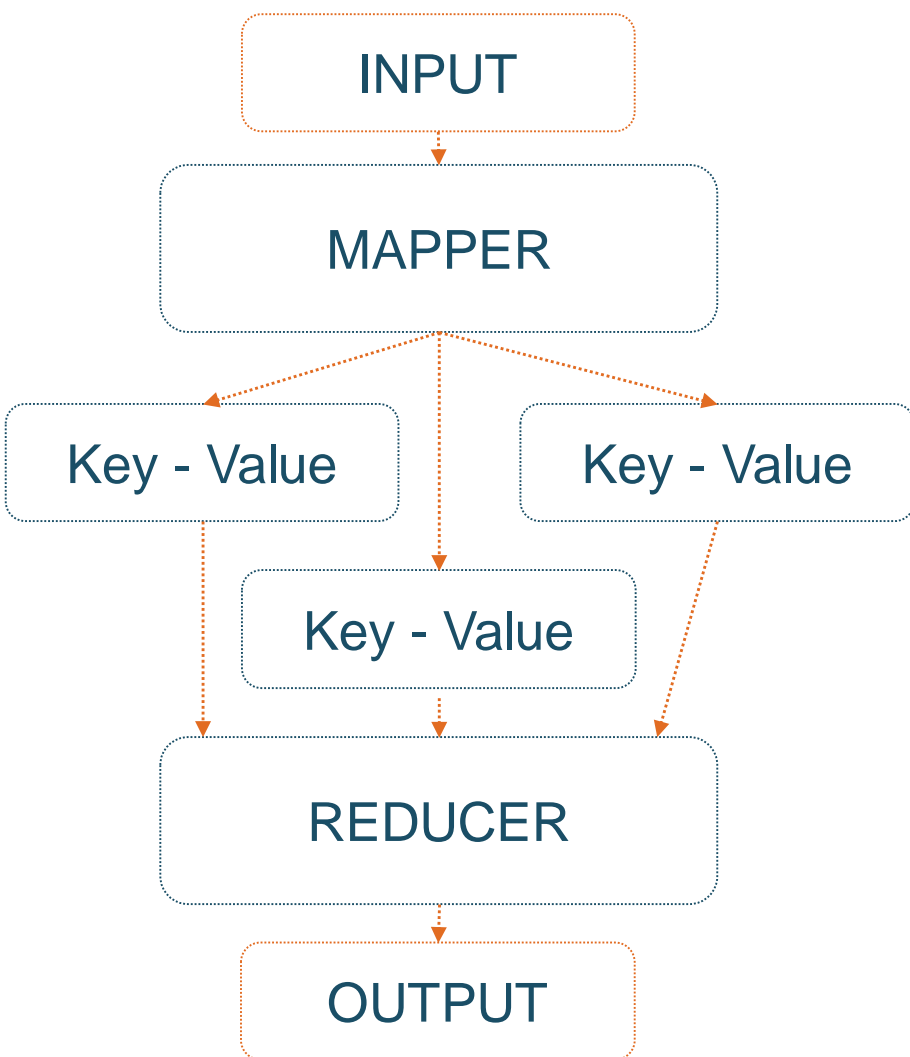
обычный файл\тс вашими\тданными

# MAP – REDUCE



обычный файл\тс вашими\тданными

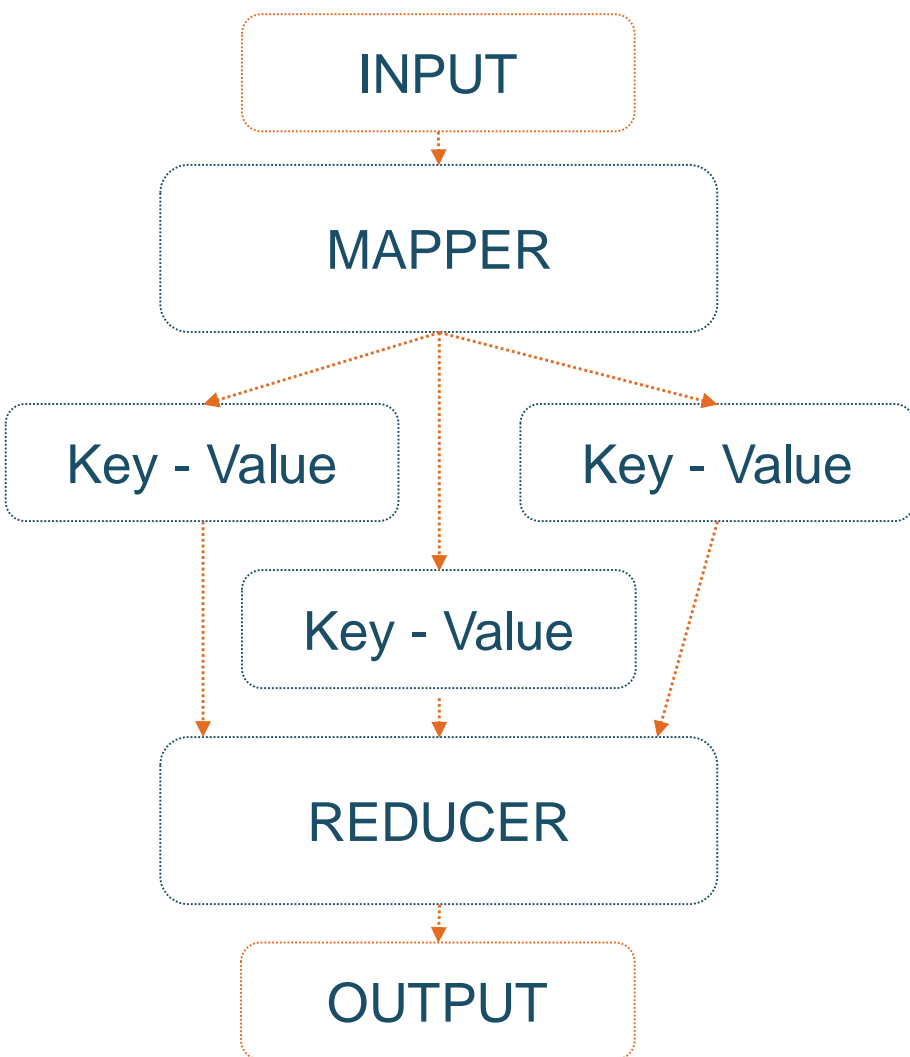
# MAP – REDUCE



## 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

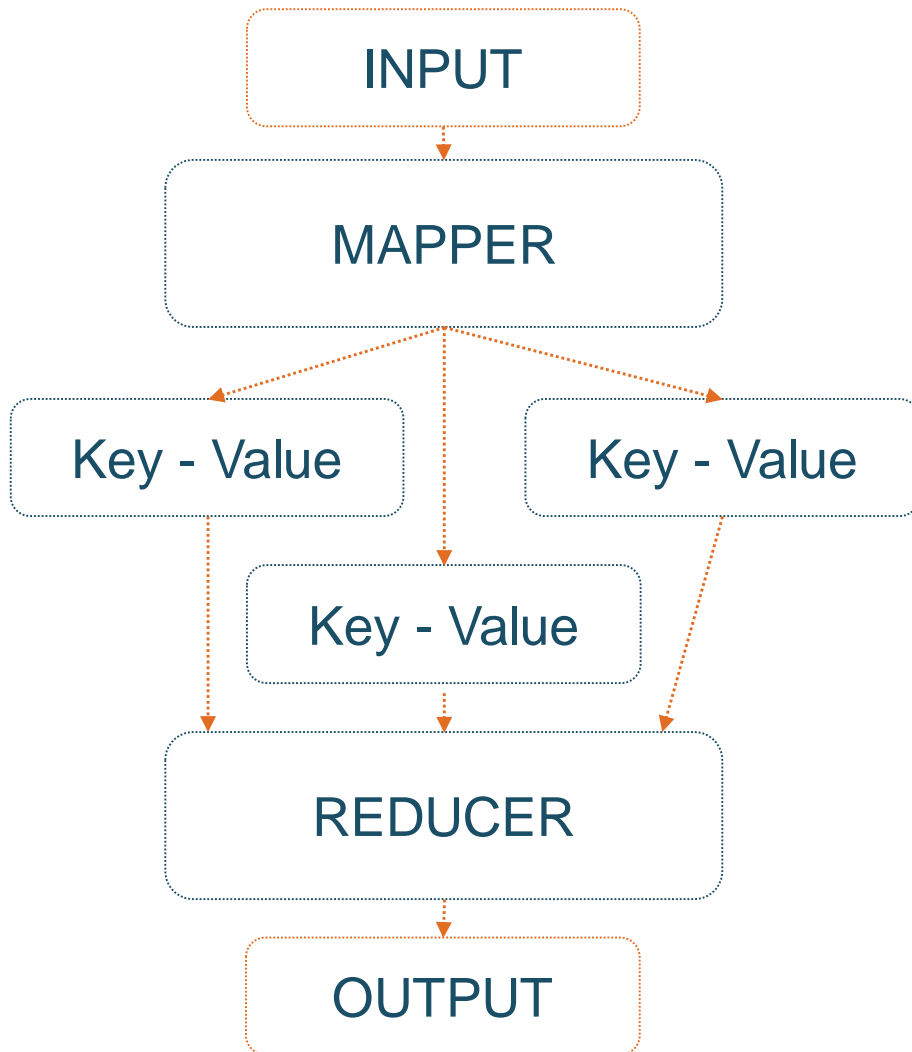


## MAPPER

- опция процесса:  
`mapred.max.split.size`
- формула расчета мапперов:  
общий размер данных / `mapred.max.split.size`

Пример: 1ТВ данных, 100MB split size:  
 $(1000 \times 1000) / 100 = 10000$  мапперов

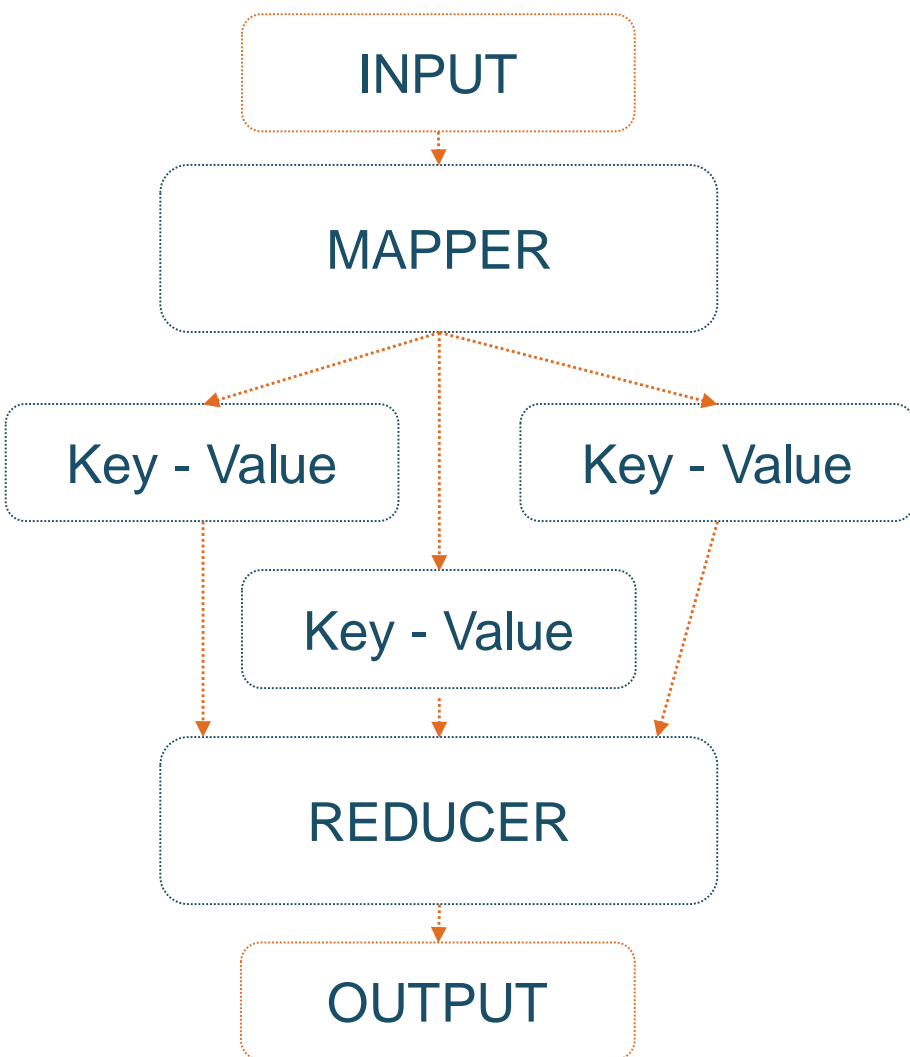
# MAP – REDUCE



REDUCER

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

# MAP – REDUCE



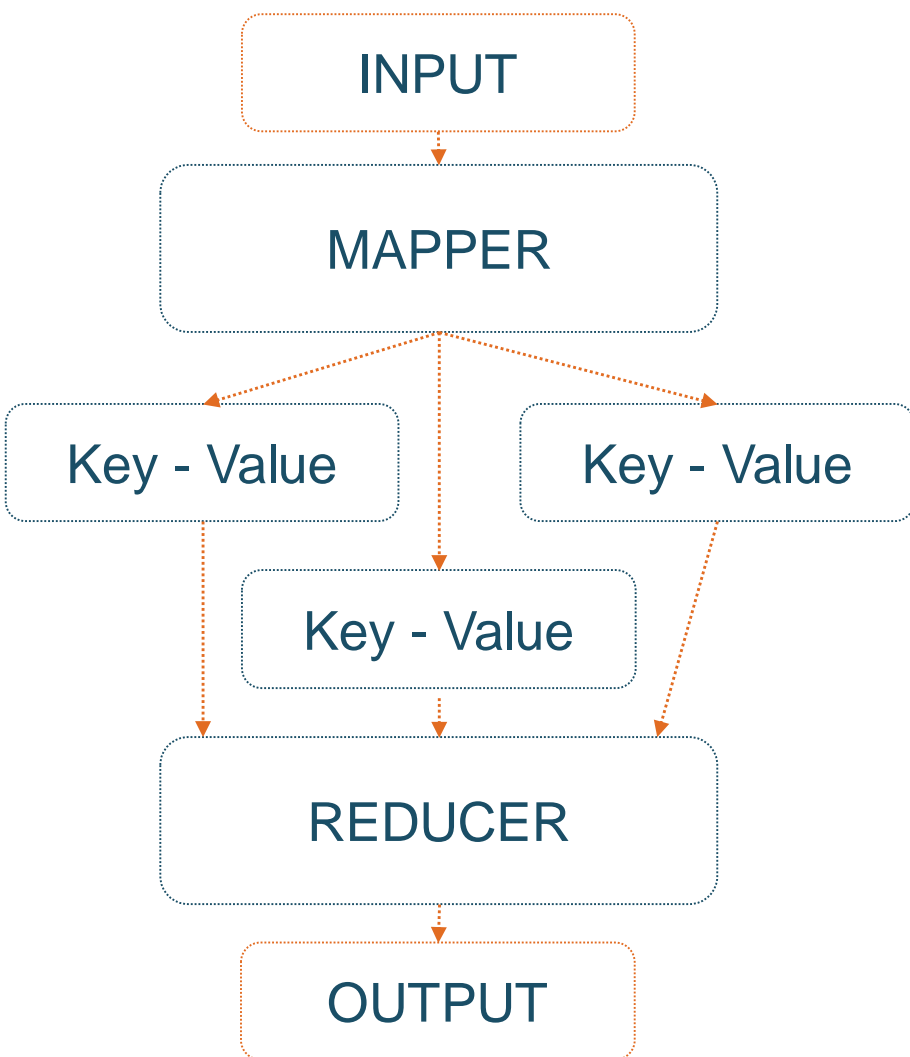
## REDUCER

- 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

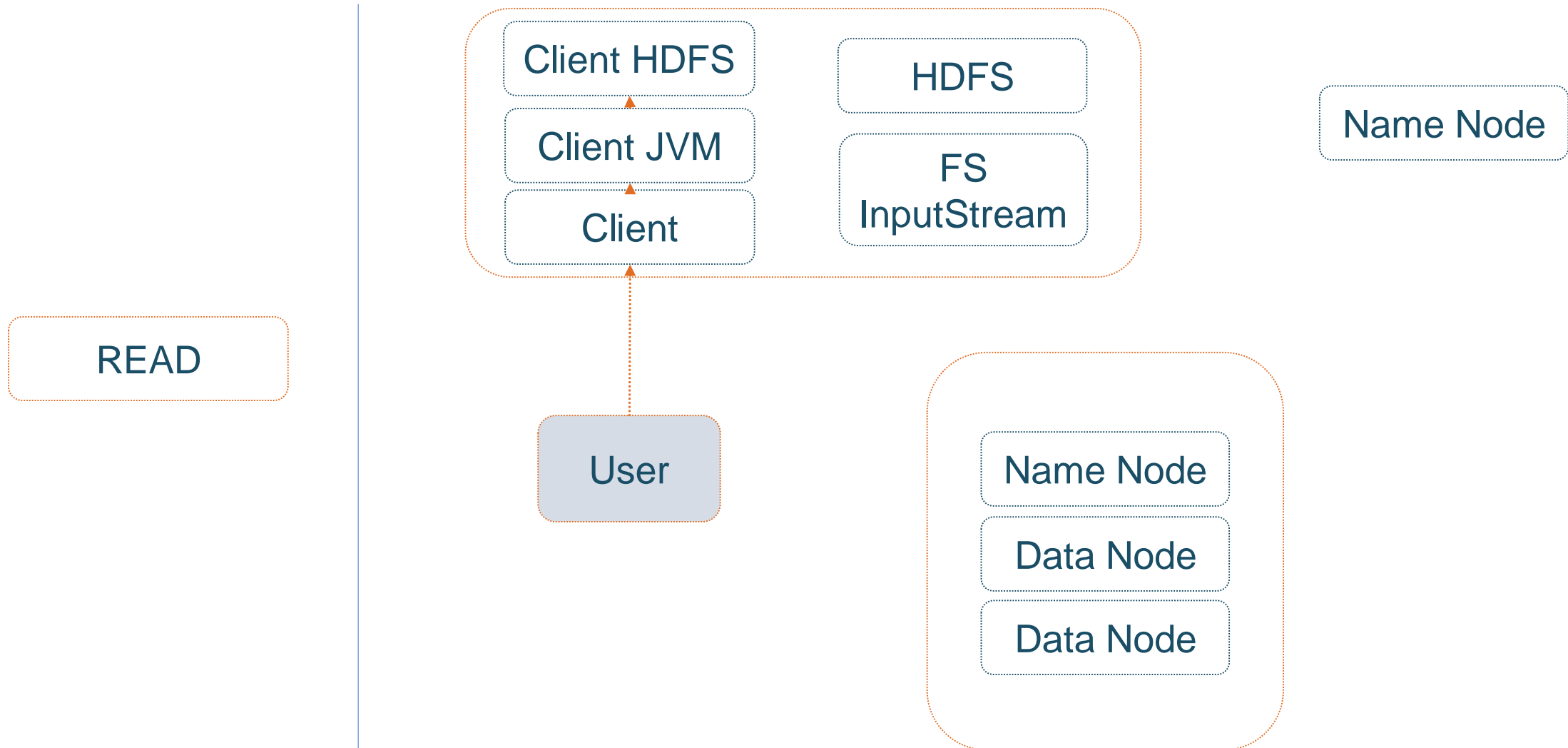
- опция процесса:  
`job.setNumreduceTasks(int)`  
**если `int = 0`, то reducers не выполнятся**
- формула расчета редьюсеров:  
`const * (кол-во нод * макс. контейнеров на ноде)`

Пример: `const = 0.95` или `1.75` всегд, 3 ноды,  
8 контейнеров на ноде

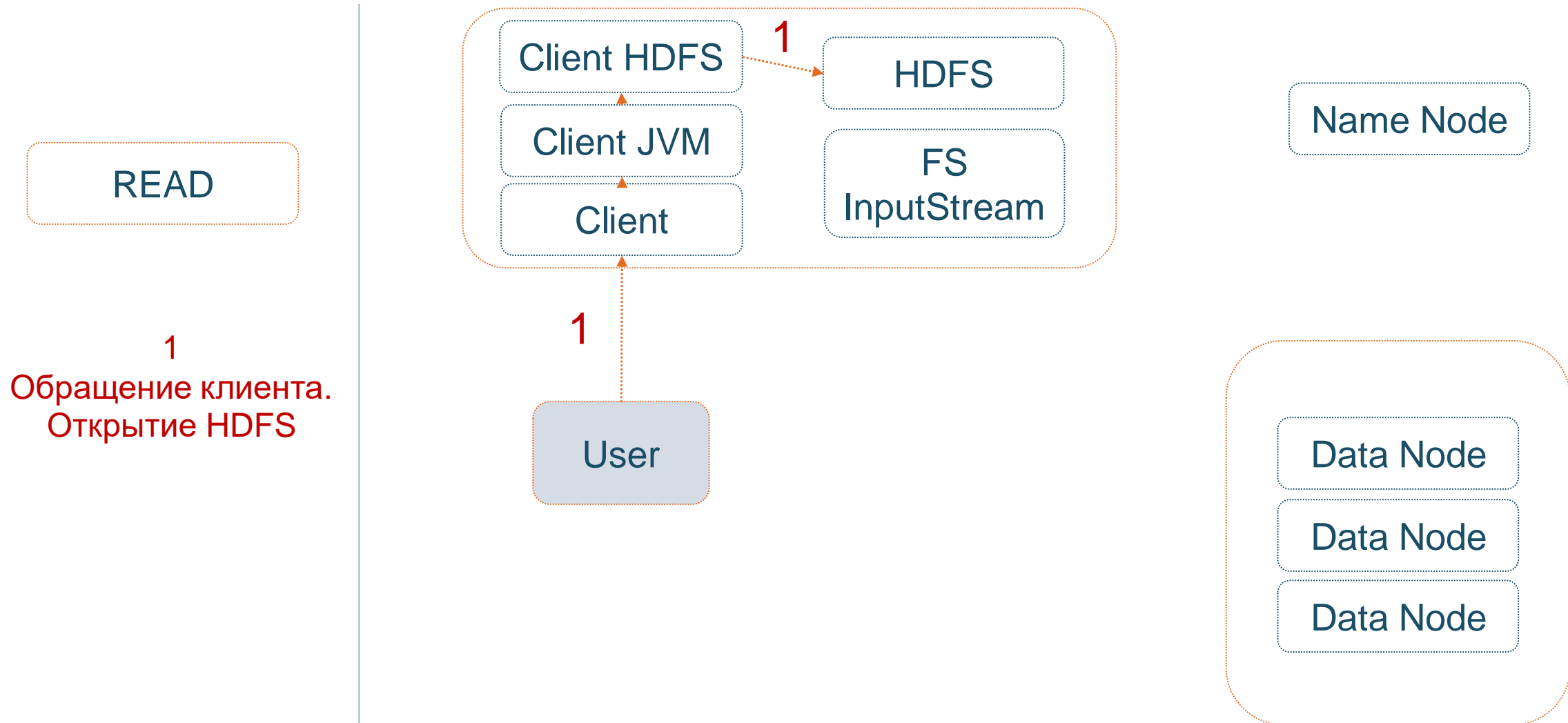
`math.ceil(0.95 * (3 * 8))`



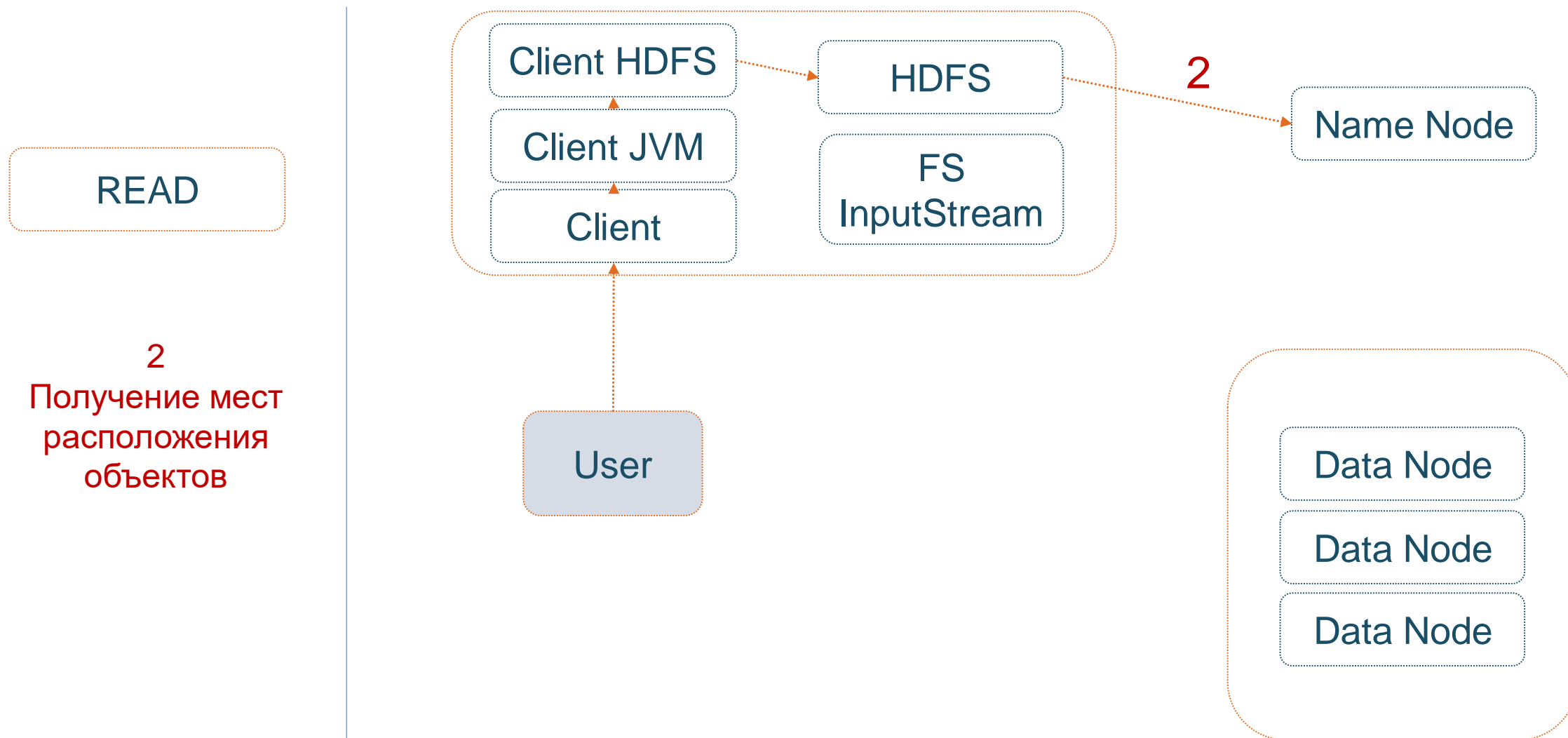
# MAP – REDUCE | READ HDFS



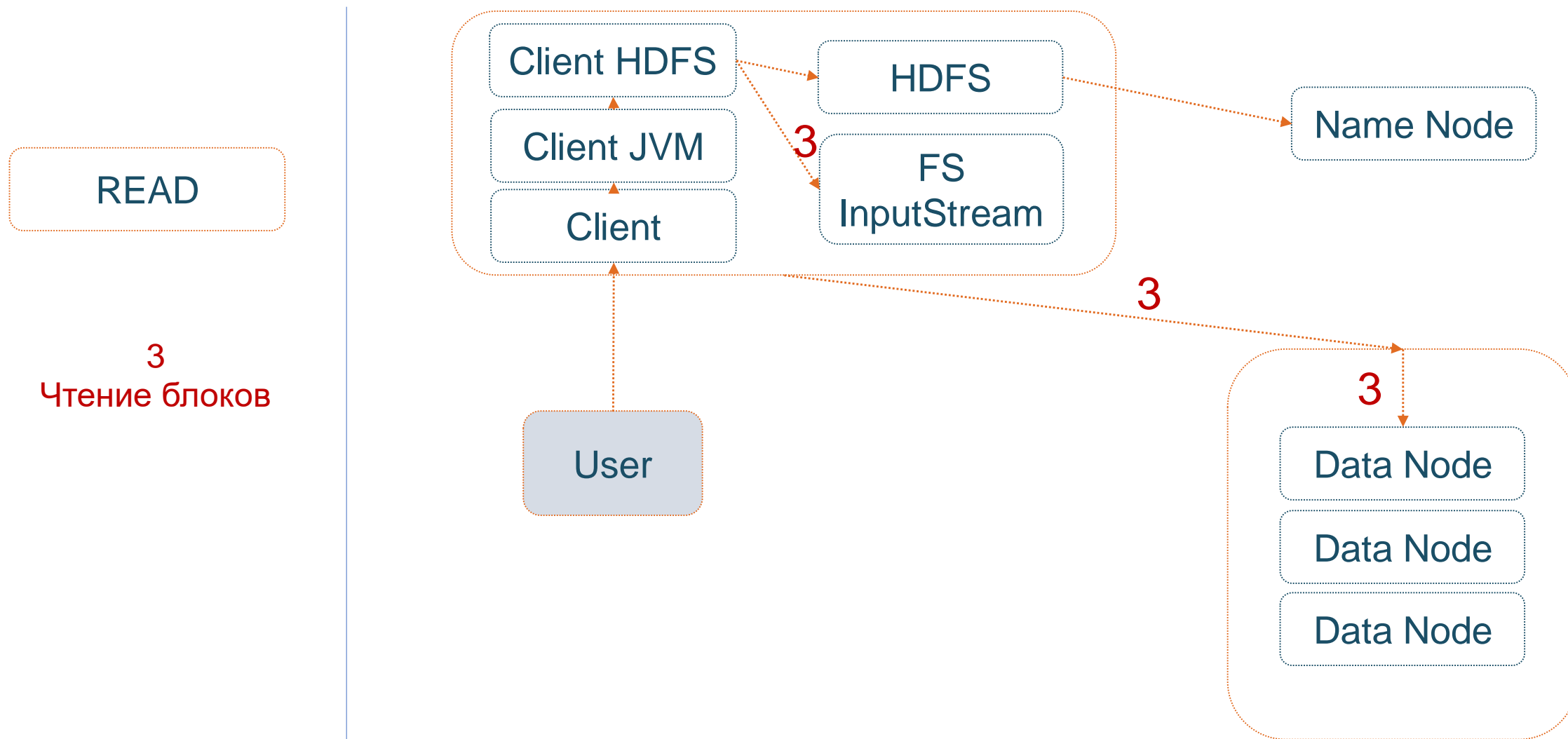
# MAP – REDUCE | READ HDFS



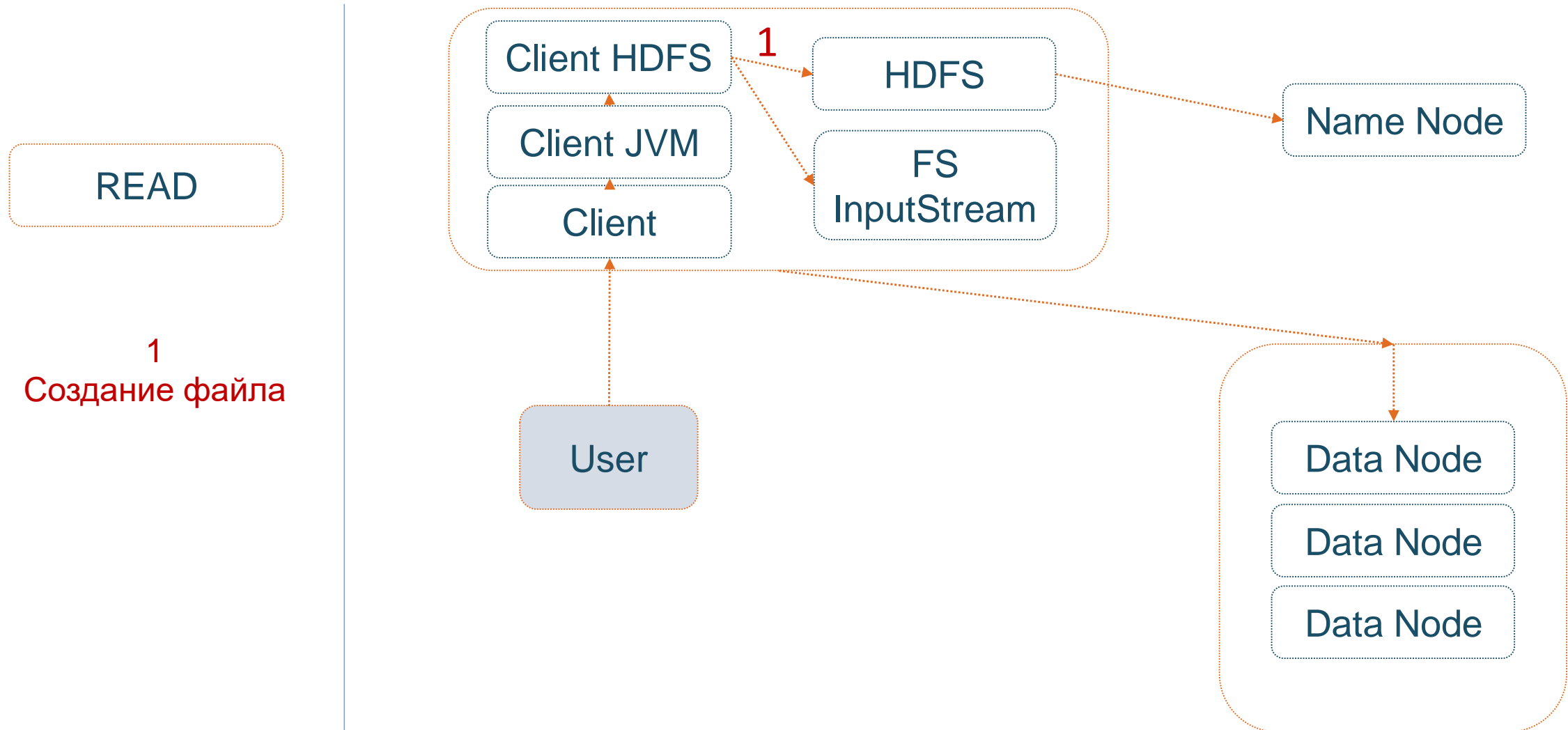
# MAP – REDUCE | READ HDFS



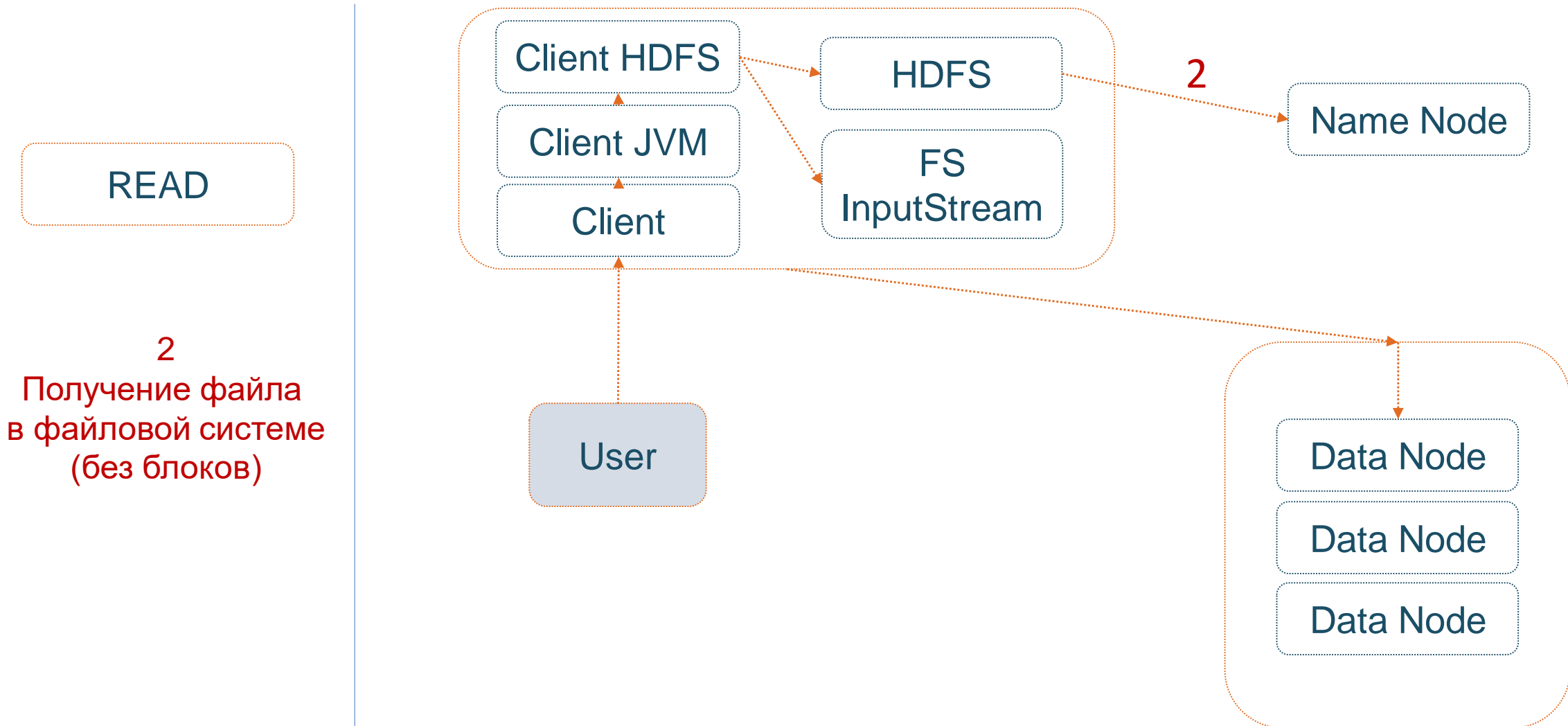
# MAP – REDUCE | READ HDFS



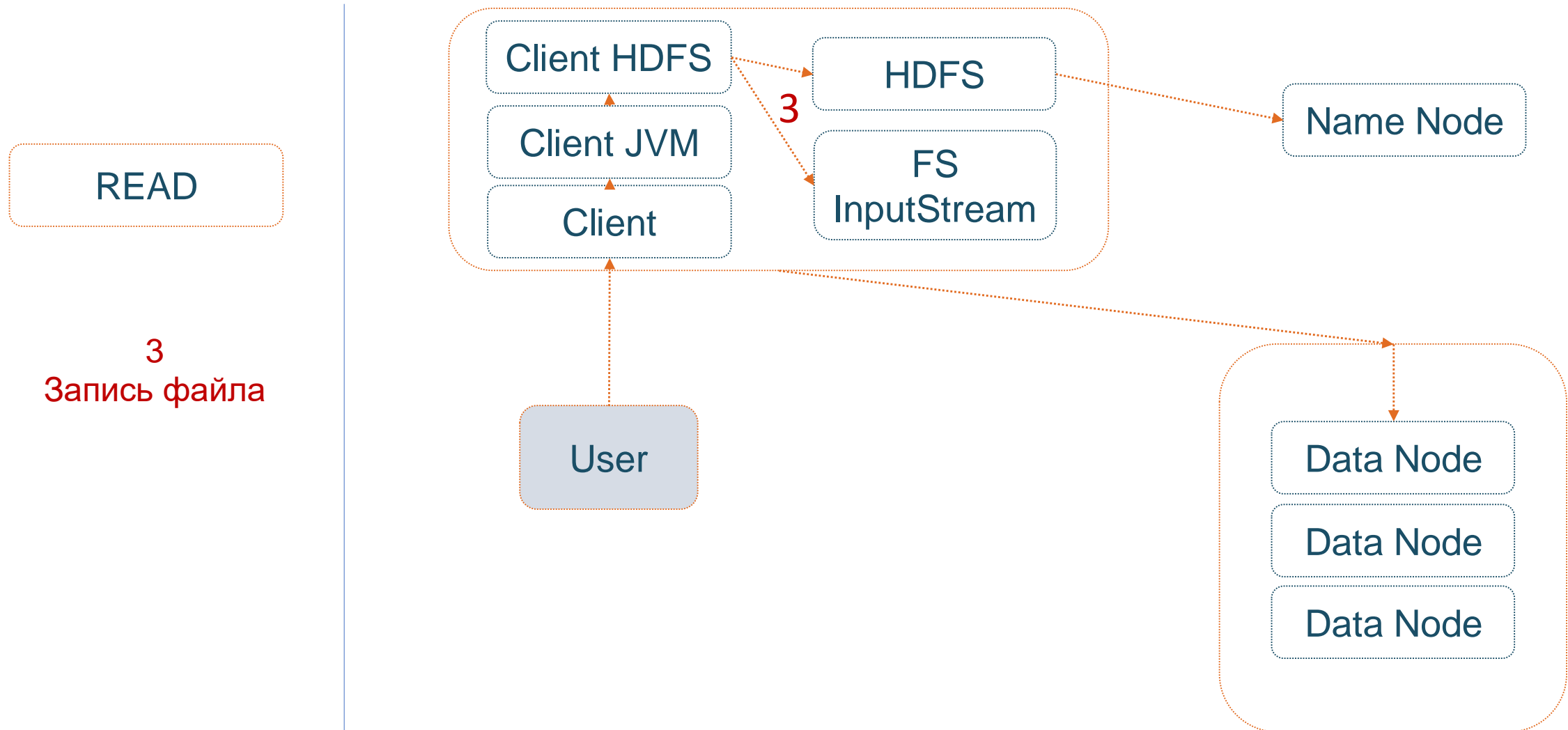
# MAP – REDUCE | WRITE HDFS



# MAP – REDUCE | WRITE HDFS



# MAP – REDUCE | WRITE HDFS



# MAP – REDUCE | WRITE HDFS

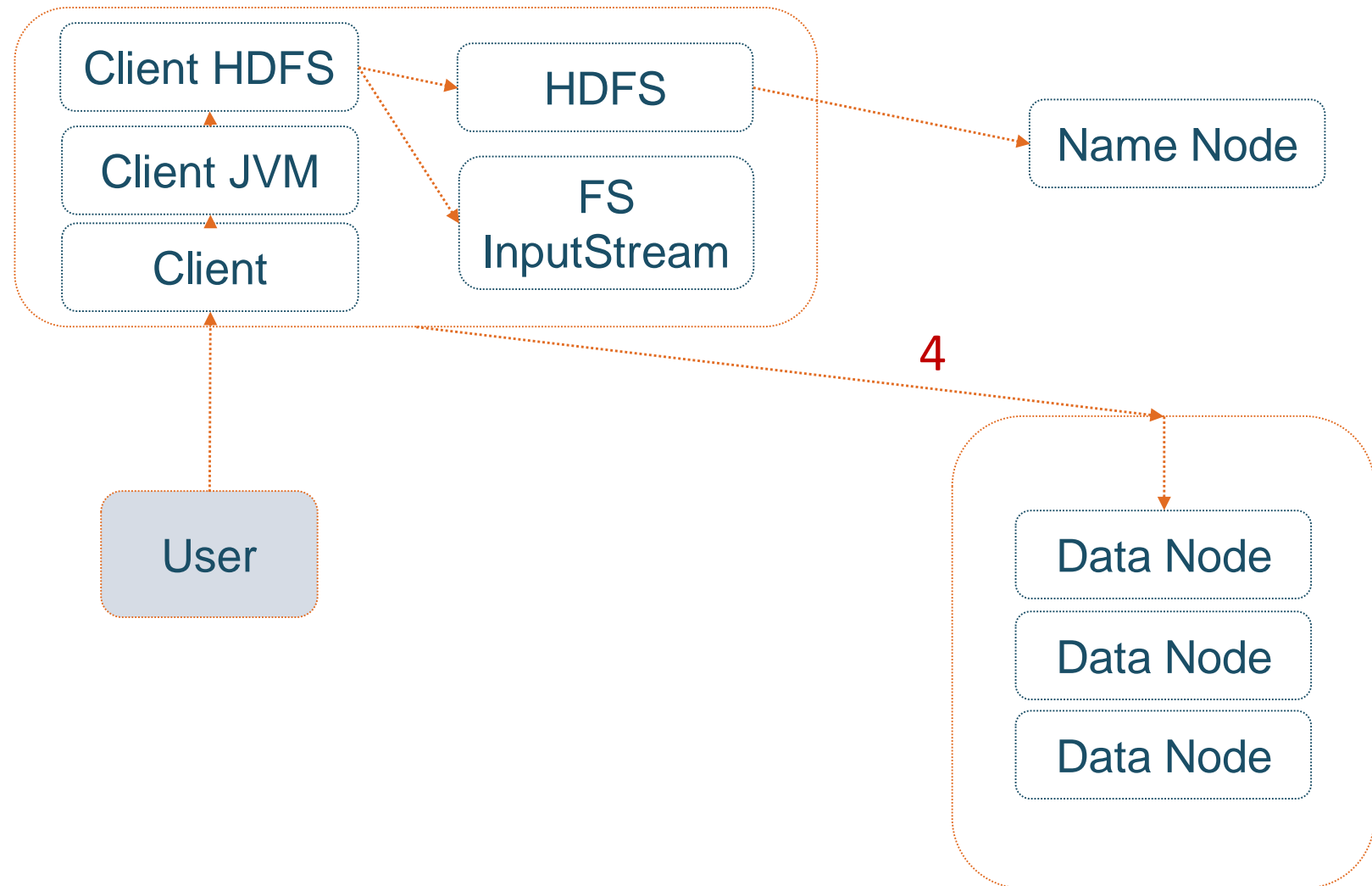
READ

4

Запись файла:

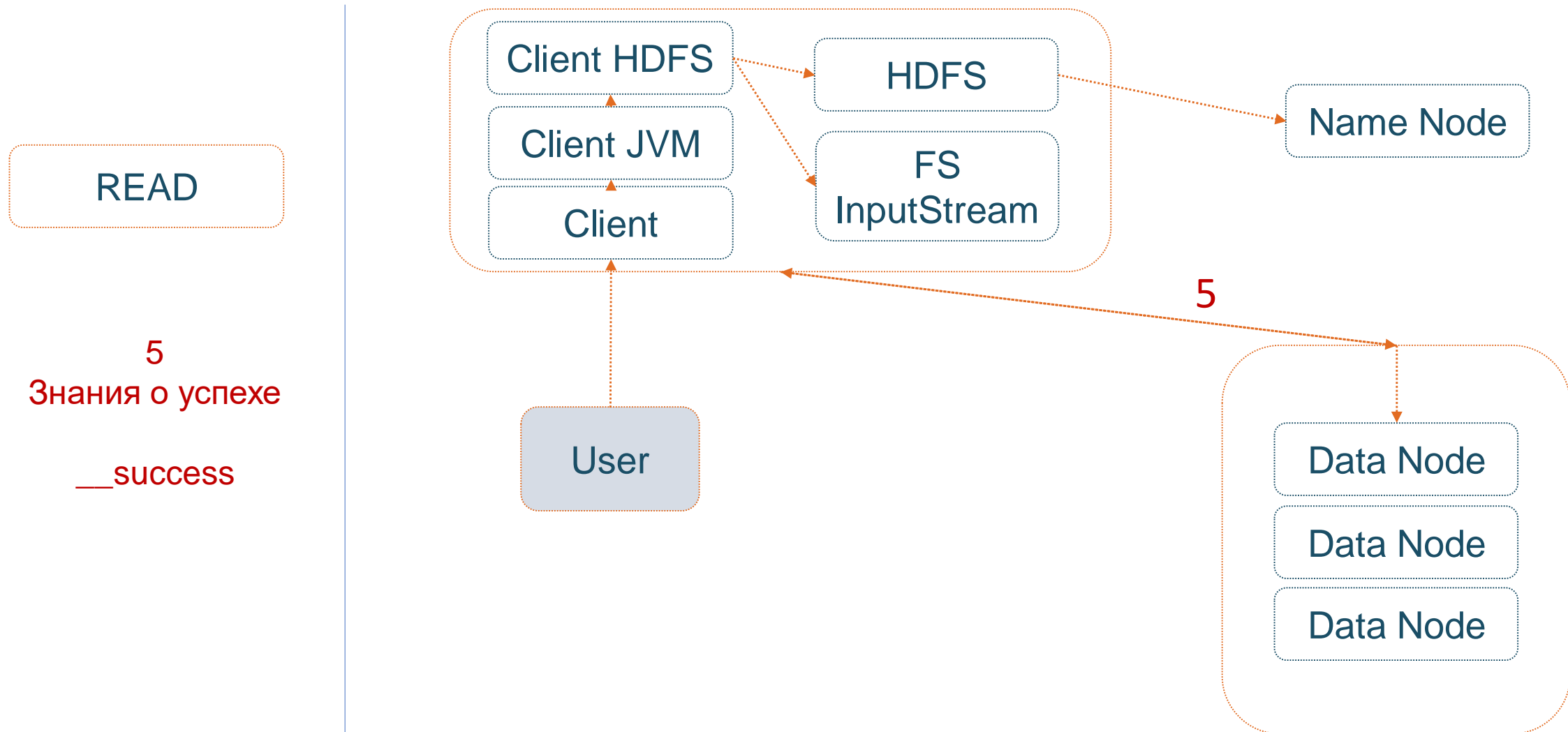
- разделение на блоки
- разделение на ноды

DataNode Pipeline

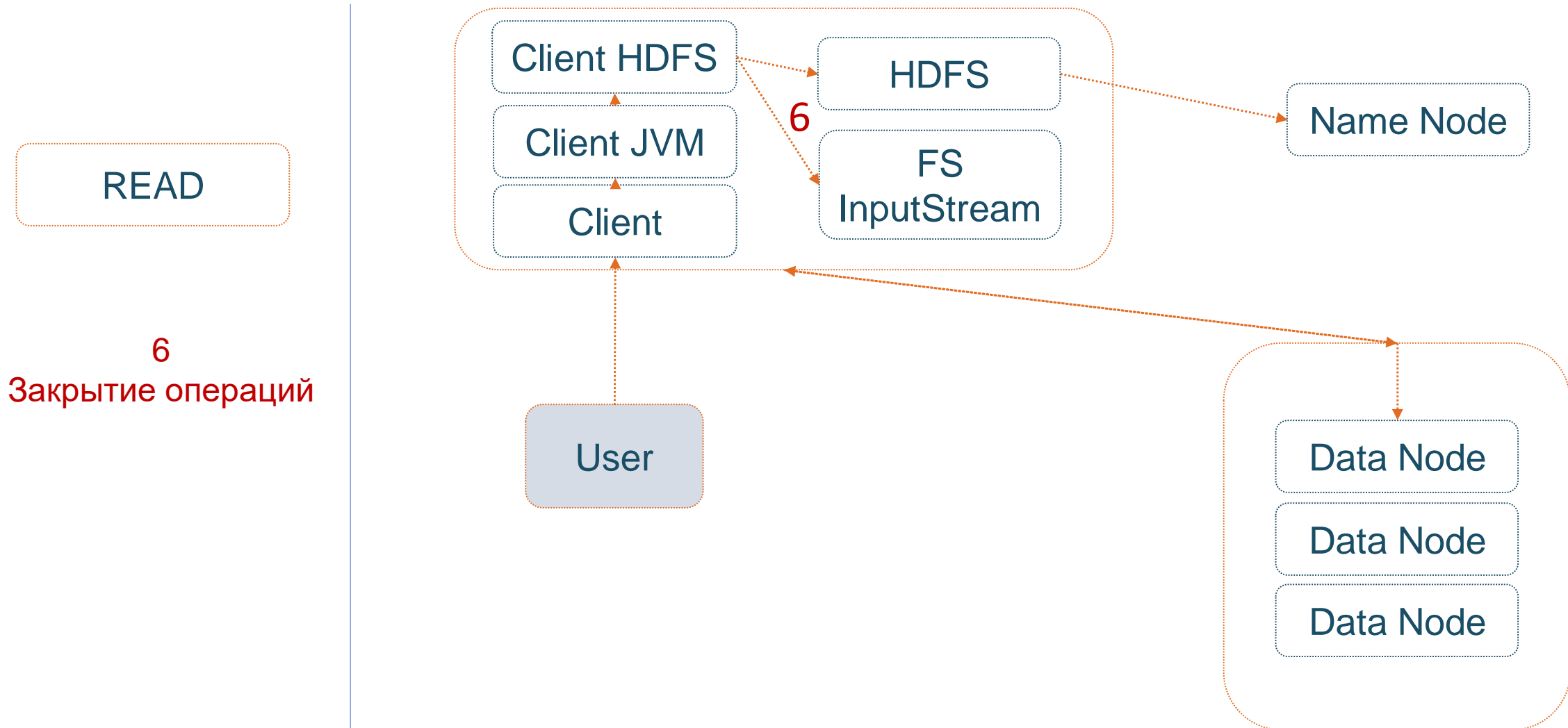




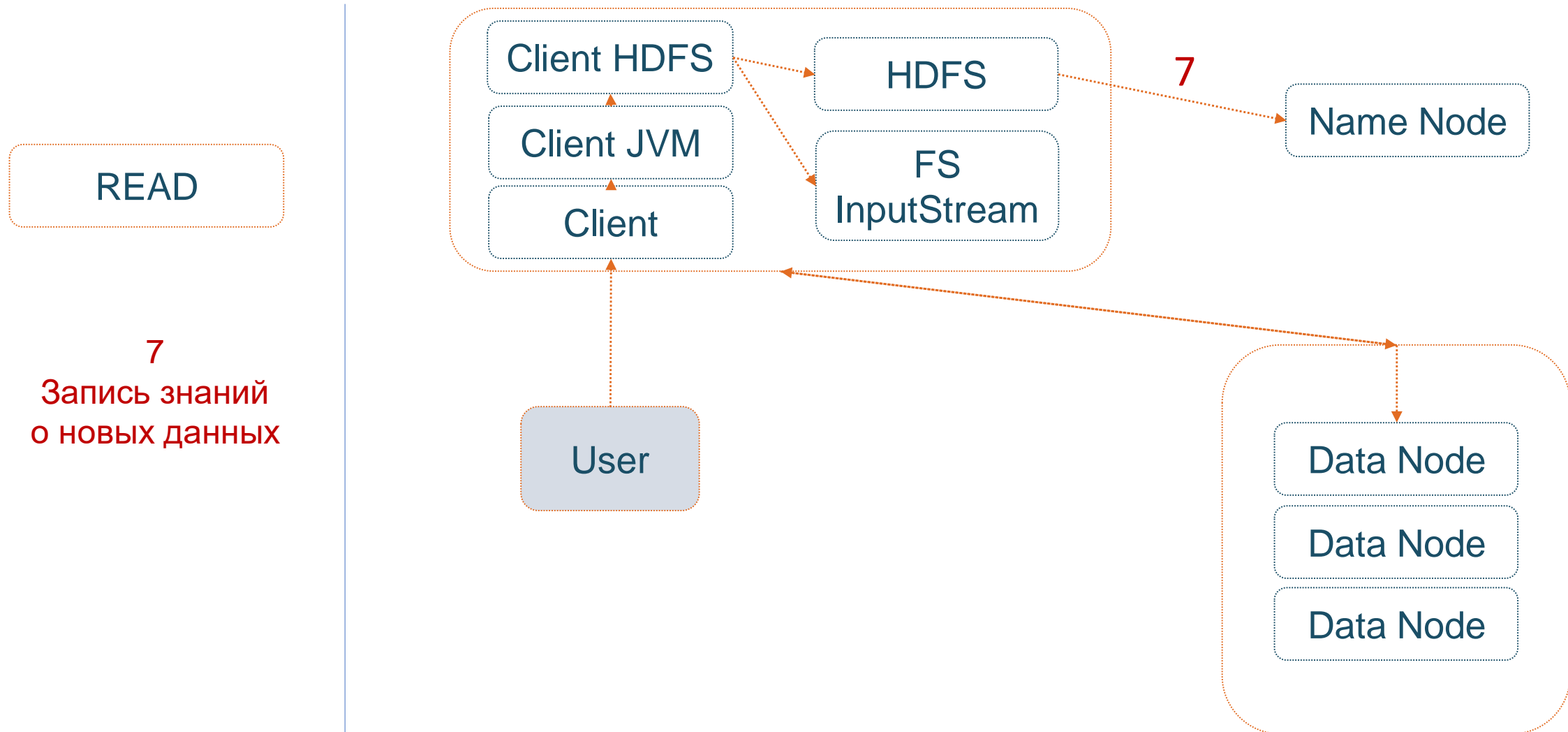
# MAP – REDUCE | WRITE HDFS



# MAP – REDUCE | WRITE HDFS

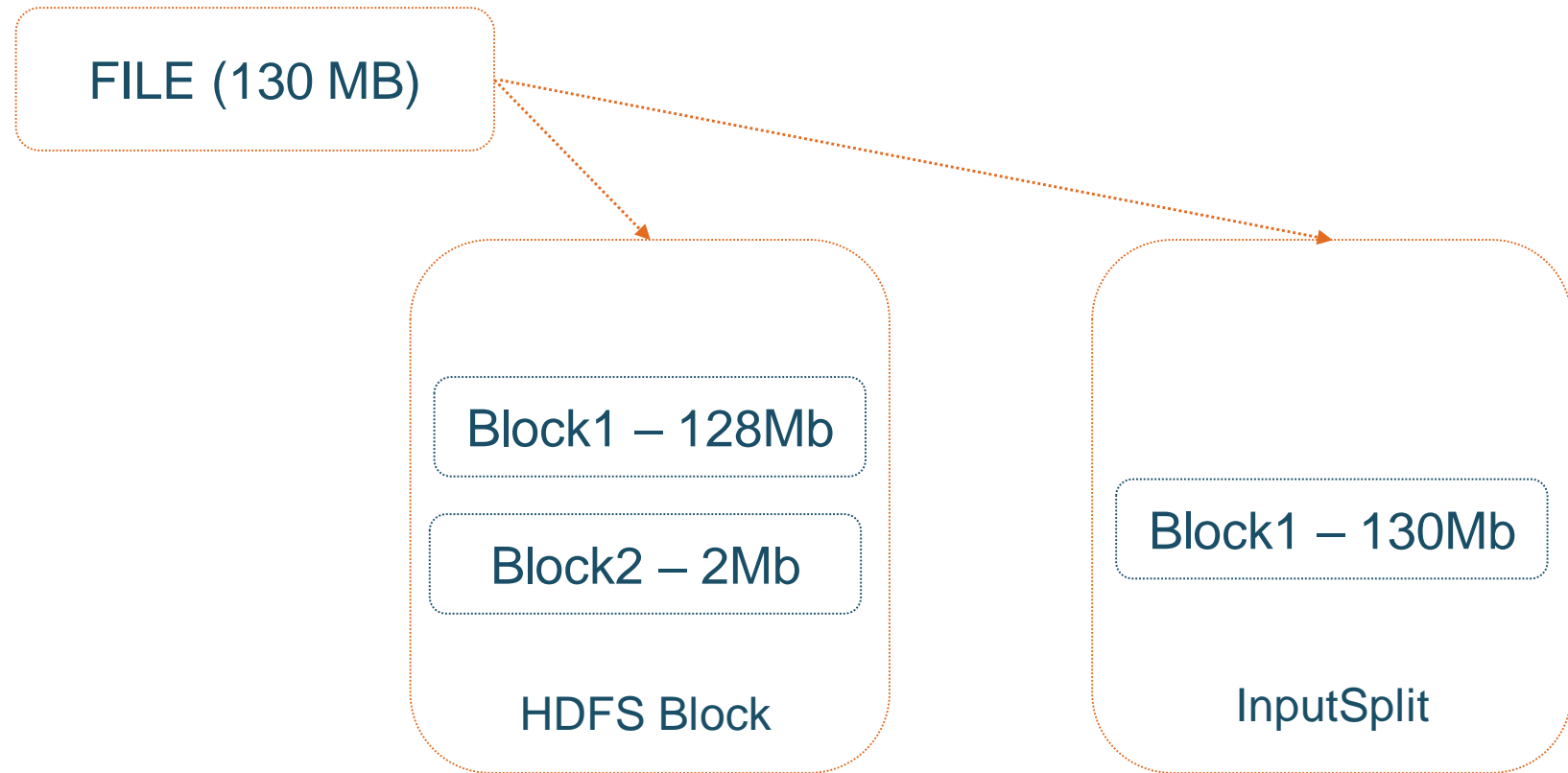


# MAP – REDUCE | WRITE HDFS



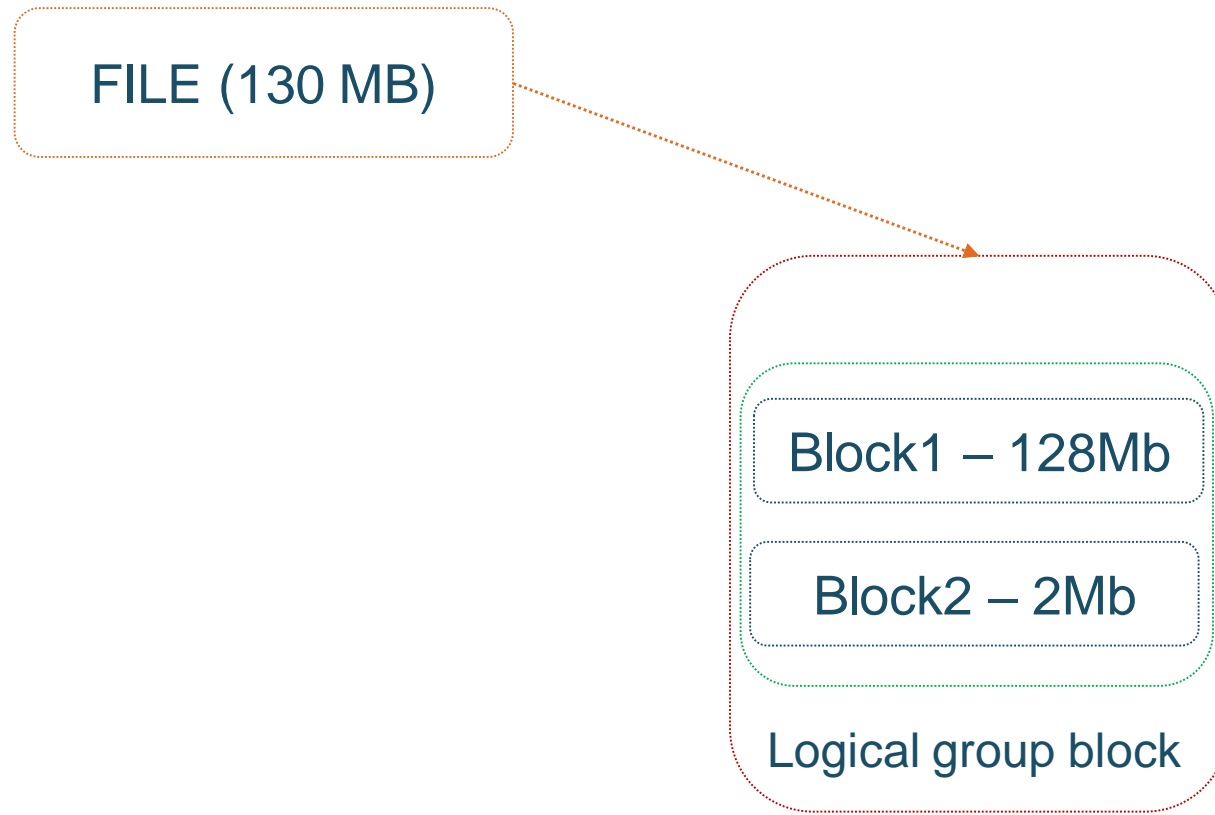
# MAP – REDUCE | HDFS BLOCKS

InputSplit != HDFS Block



# MAP – REDUCE | HDFS BLOCKS

InputSplit != HDFS Block



ПОПРОБУЕМ  
САМОСТОЯТЕЛЬНО

**ТЫ НЕ ДЕЛАЕШЬ ЭТО НЕПРАВИЛЬНО**



**ЕСЛИ НИКТО НЕ ЗНАЕТ, ЧТО  
КОНКРЕТНО ТЫ ДЕЛАЕШЬ**

ДОМАШНЯЯ РАБОТА

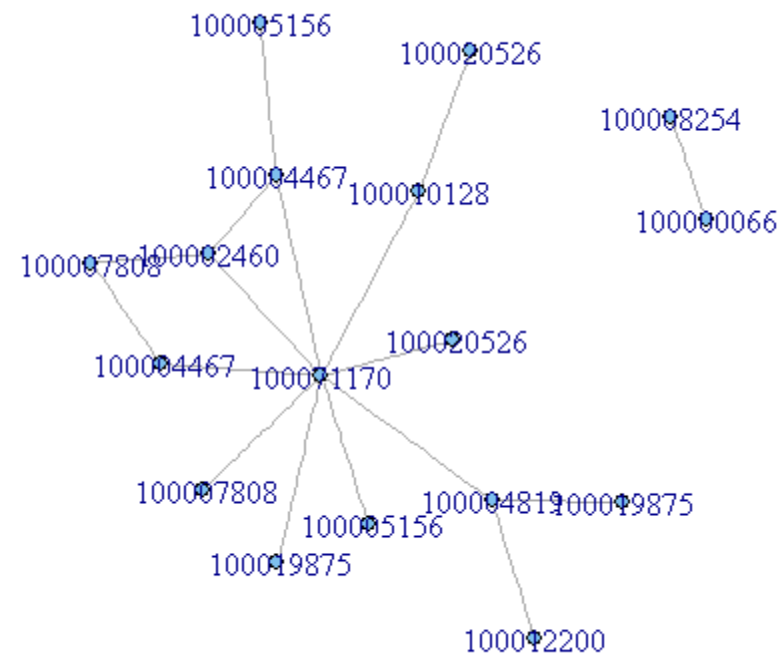
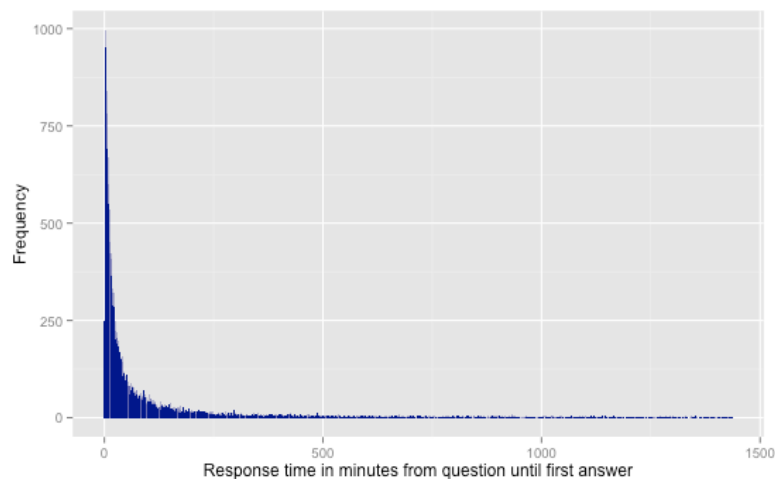
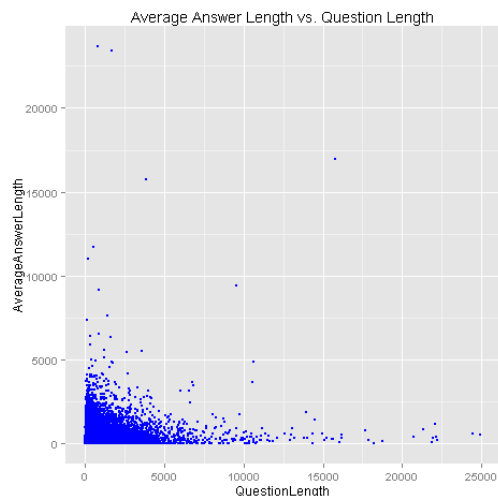


# MAR-REDUCE

- **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



cs101  
cs253  
st101  
bug  
homework  
meta  
cs212  
discussion  
cs373  
cs262

