



Работа с большими данными

SELEZNEV ARTEM
HEAD OF CVM ANALYTICS @ MAGNIT



tg: @SeleznevArtem

 /NameArtem

 /seleznev-artem

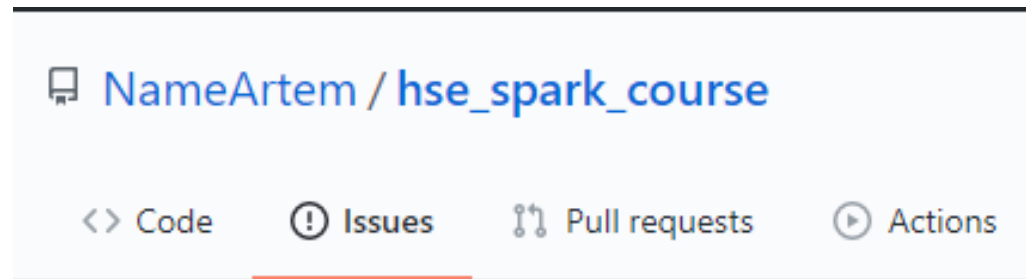
 /seleznev.artem.info



https://github.com/NameArtem/hse_spark_course



https://github.com/NameArtem/hse_spark_course

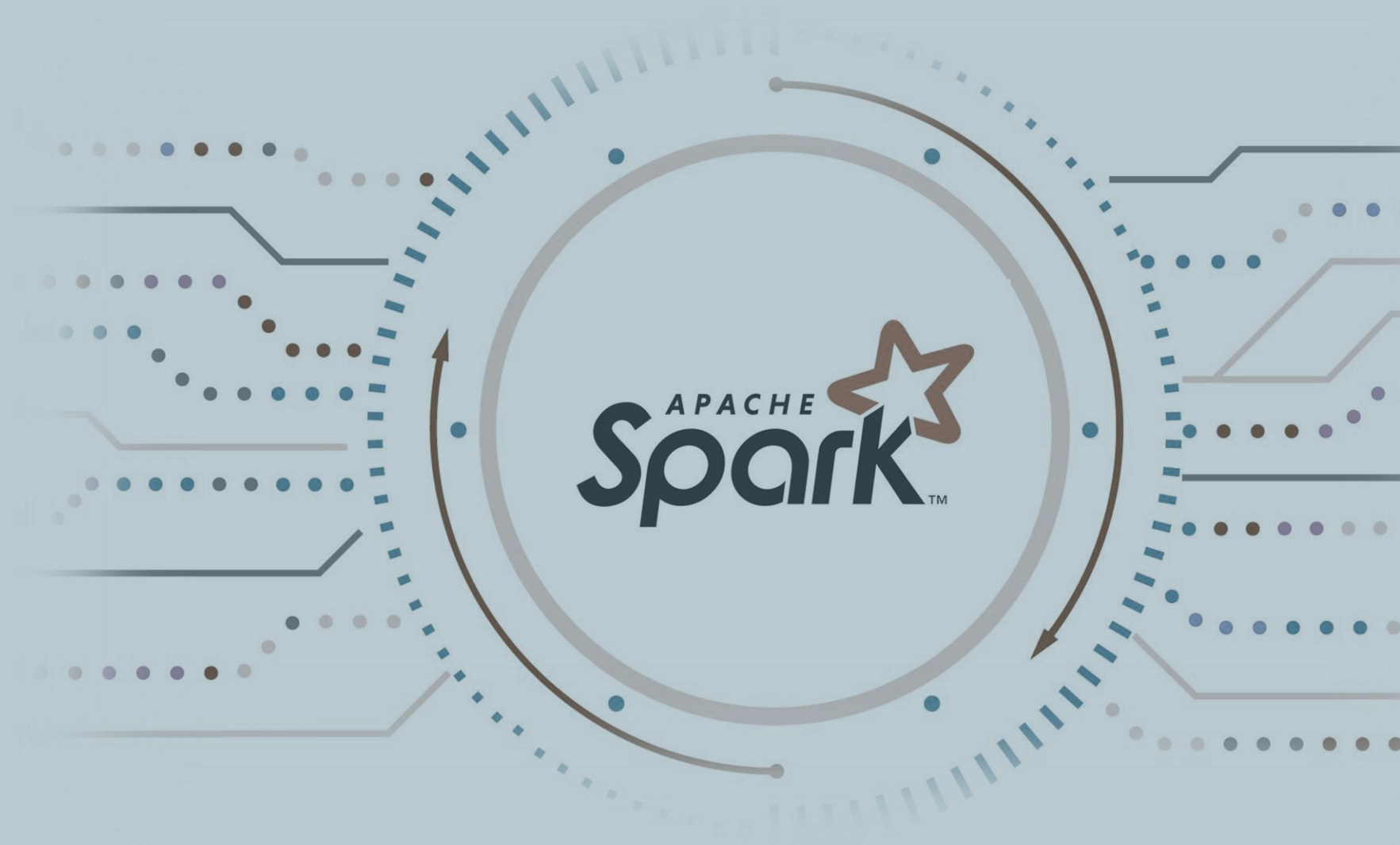




https://github.com/NameArtem/hse_spark_course

<https://cutt.ly/2ISpyZs>

О КУРСЕ



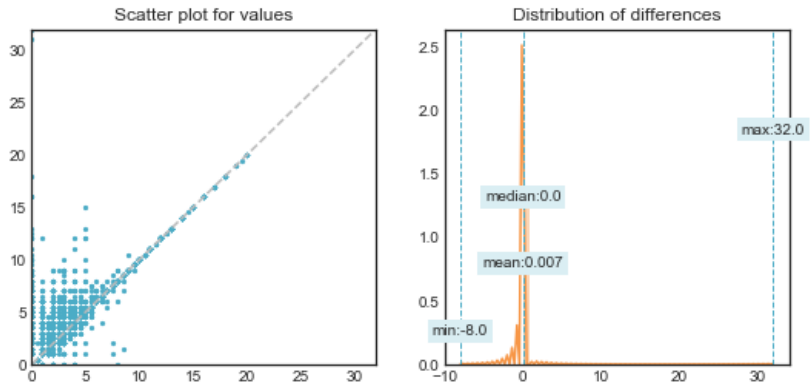
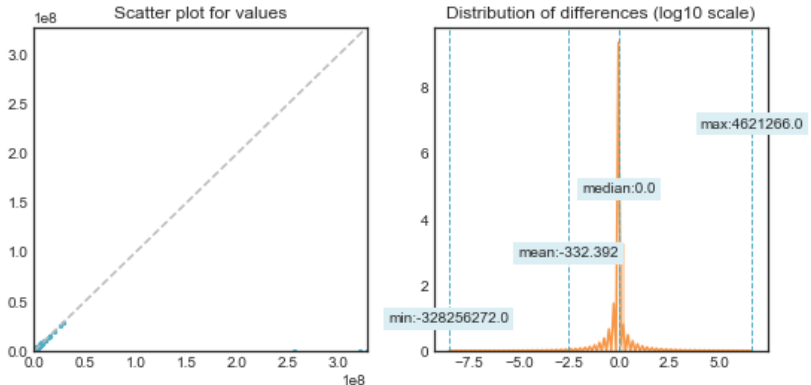
№	Тема занятия
1	Как работают и где живут большие данные
2	Погружение среду Spark. Spark RDD / Spark SQL
3	Spark ML / Spark TimeSeries
4	Advanced ML & проверка результатов качества моделей
5	Spark Ecosystem (AirFlow, H2O AutoML)
6	Spark в архитектуре проекта / Spark CI/CD

№	Тема занятия
1	Как работают и где живут большие данные
2	Погружение среду Spark. Spark RDD / Spark SQL
3	Spark ML / Spark TimeSeries
4	Advanced ML & проверка результатов качества моделей
5	Spark Ecosystem (AirFlow, H2O AutoML)
6	Spark в архитектуре проекта / Spark CI/CD



Курсовой проект

Курсовой проект Data Quality

column	bathroomcnt	consistency check
corr	0,992	 <p>The figure consists of two plots. The left plot, 'Scatter plot for values', shows a strong positive correlation between two variables, with points tightly clustered along a diagonal line from (0,0) to (30,30). The right plot, 'Distribution of differences', is a histogram showing the distribution of differences between the two variables. It features a sharp peak at 0.0, with a mean of 0.007 and a median of 0.0. The x-axis ranges from -10 to 30, and the y-axis ranges from 0.0 to 2.5. Vertical dashed lines indicate the minimum value at -8.0 and the maximum value at 32.0.</p>
min diff	-8	
mean diff	0,007	
median diff	0	
max diff	32	
column	lotsizesquarefeet	consistency check
corr	1	 <p>The figure consists of two plots. The left plot, 'Scatter plot for values', shows a perfect positive correlation between two variables, with points exactly on a diagonal line from (0,0) to (3.0e8, 3.0e8). The right plot, 'Distribution of differences (log10 scale)', is a histogram showing the distribution of differences between the two variables on a logarithmic scale. It features a sharp peak at 0.0, with a mean of -332.392 and a median of 0.0. The x-axis ranges from -7.5 to 5.0, and the y-axis ranges from 0 to 8. Vertical dashed lines indicate the minimum value at -328256272.0 and the maximum value at 4621266.0.</p>
min diff	-328256272	
mean diff	-332,392	
median diff	0	
max diff	4621266	

ИНСТРУМЕНТЫ

Python

Linux

Git

Spark

ИНСТРУМЕНТЫ

Python

Linux

Git

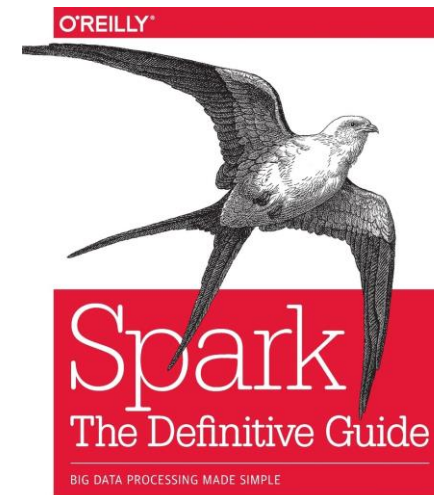
Spark

SPARK ЛИТЕРАТУРА



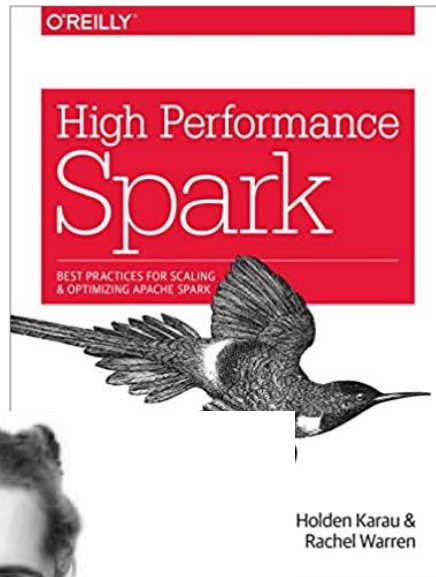
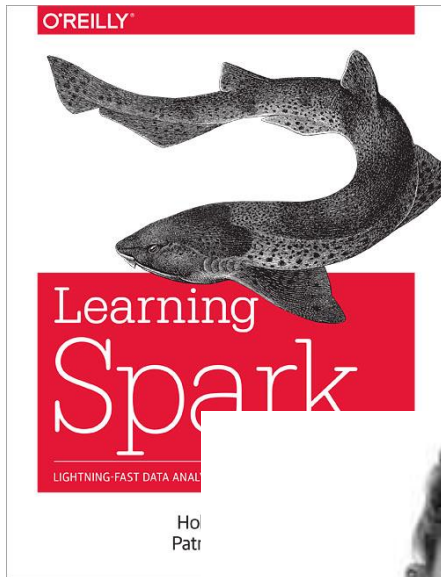
Spark Overview

<https://spark.apache.org/docs/latest/>



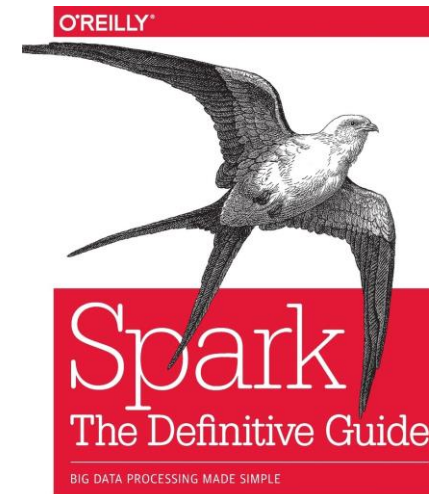
Bill Chambers & Matei Zaharia

SPARK ЛИТЕРАТУРА



Spark Overview

<https://spark.apache.org/docs/latest/>



Bill Chambers & Matei Zaharia

А ГДЕ HADOOP?

Python

Linux

Git

Spark

А ГДЕ HADOOP?



Cluster



Cluster

SPARK vs HADOOP



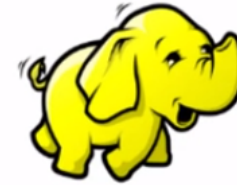
Cluster

MAP – REDUCE +++

CALC IN MEMORY | CACHE

STANDALONE

DATA MINING (!) | ML



Cluster

MAP – REDUCE

РАБОЧАЯ СРЕДА КУРСА

Инфраструктура курса

- [Локальный кластер на Docker](#)
- [DataBricks Community](#)

или установить самостоятельно

CLUSTER
DEEPER....



ЖИЗНЬ НА КЛАСТЕРЕ

- Ansible

ЖИЗНЬ НА КЛАСТЕРЕ

- Ansible
 - Apache Hadoop3

ЖИЗНЬ НА КЛАСТЕРЕ

- Ansible
 - Apache Hadoop3
 - Apache Spark 3

ЖИЗНЬ НА КЛАСТЕРЕ

- Ansible
 - Apache Hadoop3
 - Apache Spark 3
 - Apache Drill

ЖИЗНЬ НА КЛАСТЕРЕ

- Ansible
 - Apache Hadoop3
 - Apache Spark 3
 - Apache Drill
 - JupyterHub + Kernel

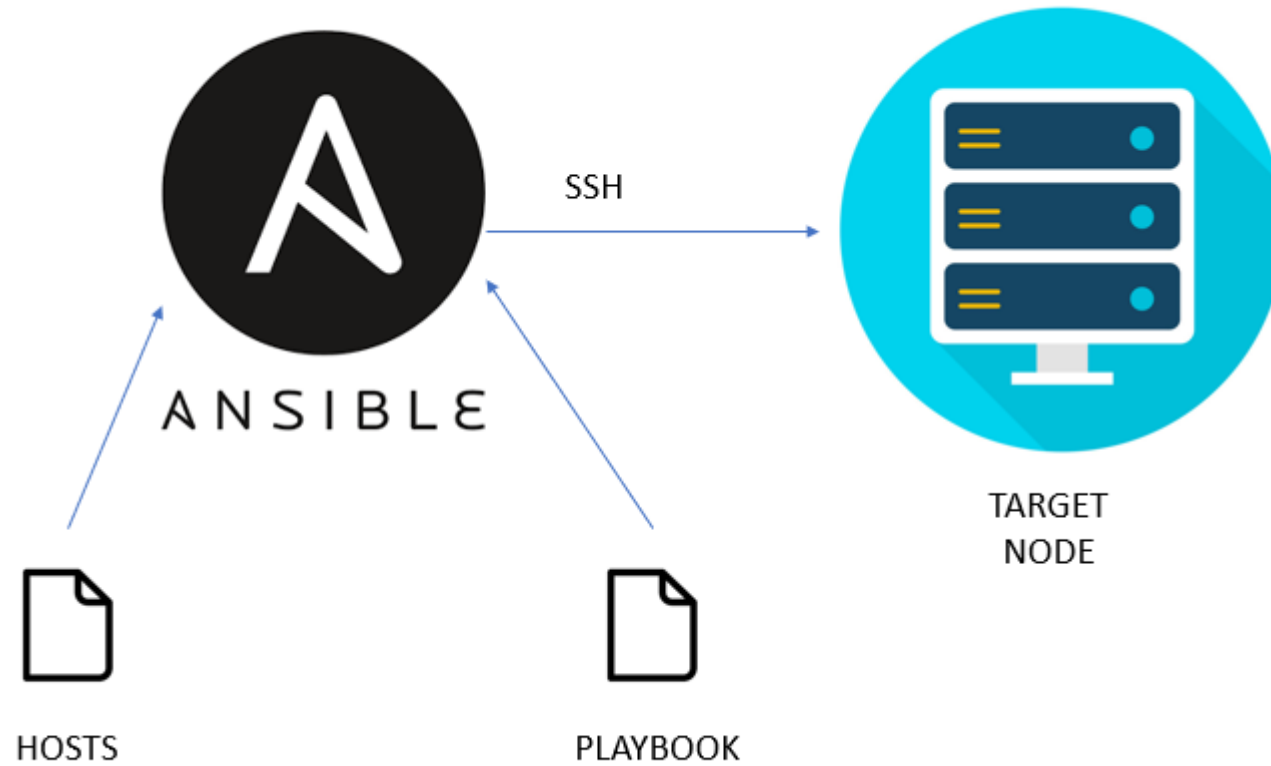
ЖИЗНЬ НА КЛАСТЕРЕ

- Ansible
 - Apache Hadoop3
 - Apache Spark 3
 - Apache Drill
 - JupyterHub + Kernel
 - Feature Store

ЖИЗНЬ НА КЛАСТЕРЕ

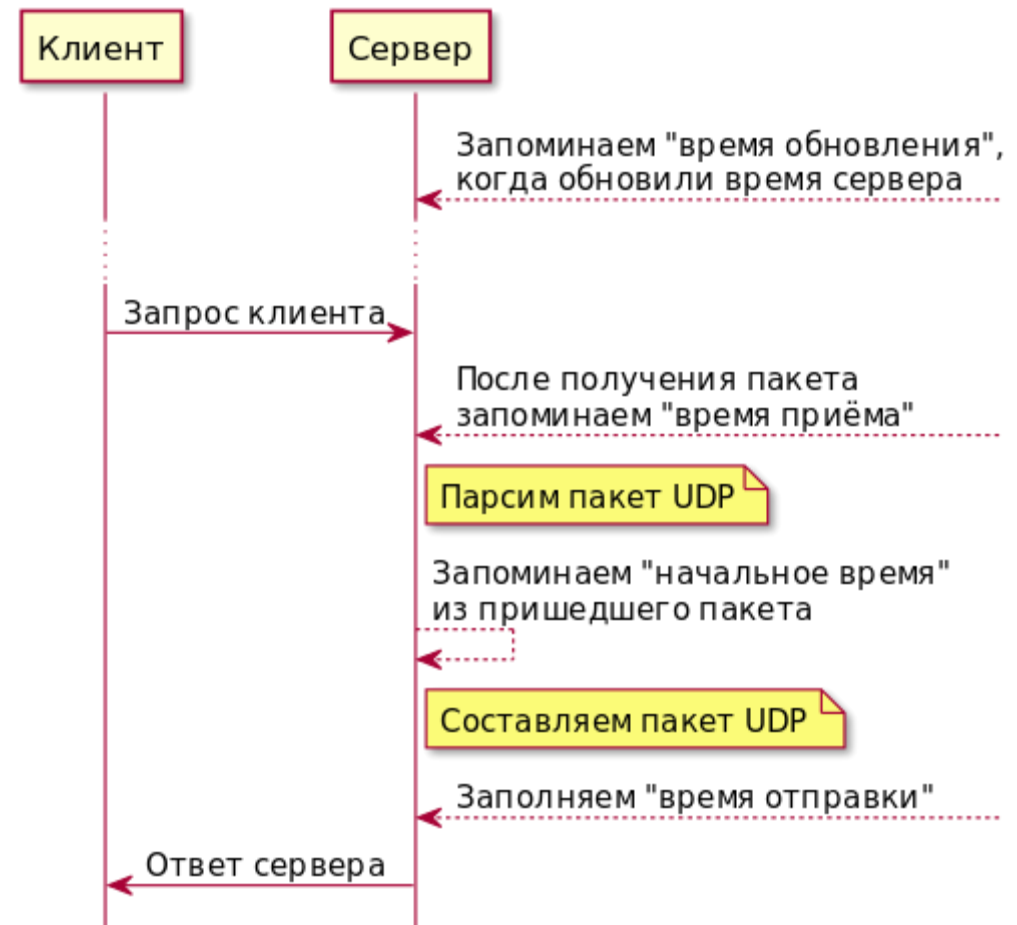
- Ansible
 - Apache Hadoop3
 - Apache Spark 3
 - Apache Drill
 - JupyterHub + Kernel
 - Feature Store

ANSIBLE



КЛАСТЕР

```
yum install -y net-tools openssh-server  
yum install ntp ntpdate ntp-doc -y  
yum install openssl  
yum install -y zookeeper  
yum install -y zookeeper-server
```



HADOOP

Download

Hadoop is released as source code tarballs with corresponding binary tarballs for convenience. The downloads are distributed via mirror sites and should be checked for tampering using GPG or SHA-512.

Version	Release date	Source download	Binary download	Release notes
3.1.4	2020 Aug 3	source (checksum signature)	binary (checksum signature)	Announcement
3.3.0	2020 Jul 14	source (checksum signature)	binary (checksum signature) binary-aarch64 (checksum signature)	Announcement
2.10.0	2019 Oct 29	source (checksum signature)	binary (checksum signature)	Announcement
3.2.1	2019 Sep 22	source (checksum signature)	binary (checksum signature)	Announcement
2.9.2	2018 Nov 19	source (checksum signature)	binary (checksum signature)	Announcement

HADOOP

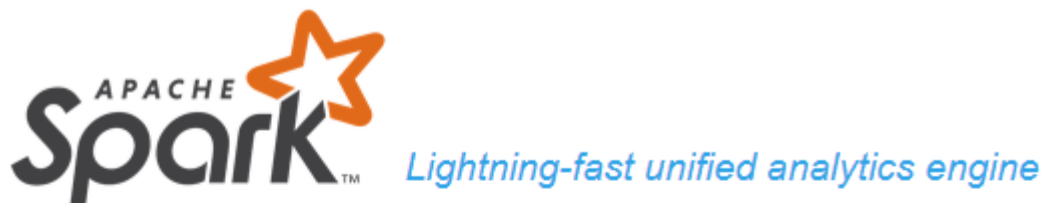
```
Sep  2 15:50 capacity-scheduler.xml
Sep  2 15:49 configuration.xsl
Sep  2 15:50 container-executor.cfg
Sep  2 15:49 core-site.xml
Sep  2 15:50 hadoop-env.cmd
Sep  2 15:50 hadoop-env.sh
Sep  2 15:50 hadoop-metrics2.properties
Sep  2 15:50 hadoop-metrics.properties
Sep  2 15:50 hadoop-policy.xml
Sep  2 15:50 hdfs-site.xml
Sep  2 15:50 httpfs-env.sh
Sep  2 15:50 httpfs-log4j.properties
Sep  2 15:50 httpfs-signature.secret
Sep  2 15:50 httpfs-site.xml
Sep  2 15:50 kms-acls.xml
Sep  2 15:50 kms-env.sh
Sep  2 15:50 kms-log4j.properties
Sep  2 15:50 kms-site.xml
Sep  2 15:50 log4j.properties
Sep  2 15:50 mapred-env.cmd
Sep  2 15:49 mapred-env.sh
Sep  2 15:50 mapred-queues.xml.template
Sep  2 15:49 mapred-site.xml
Sep  2 15:50 mapred-site.xml.template
Sep  2 15:49 masters
Sep  4 20:12 slaves
Sep  2 15:50 ssl-client.xml.example
Sep  2 15:50 ssl-server.xml.example
Sep  2 15:49 yarn-env.cmd
Sep  2 15:50 yarn-env.sh
Sep  2 15:50 yarn-site.xml
```

- core-site.xml
- hdfs-site.xml
- mapred-site.xml
- yarn-site.xml

HADOOP

```
export HADOOP_HOME=/opt/hadoop3
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_ROOT_LOGGER=INFO,console
export HADOOP_SECURITY_LOGGER=INFO,NullAppender
export HADOOP_INSTALL=$HADOOP_HOME
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export HADOOP_PREFIX=$HADOOP_HOME
export HADOOP_LIBEXEC_DIR=$HADOOP_HOME/libexec
export JAVA_LIBRARY_PATH=$HADOOP_HOME/lib/native:$JAVA_LIBRARY_PATH
export HADOOP_YARN_HOME=$HADOOP_HOME
```

SPARK

[Download](#)[Libraries ▾](#)[Documentation ▾](#)[Examples](#)[Community ▾](#)[Developers ▾](#)

Download Apache Spark™

1. Choose a Spark release:

2. Choose a package type:

3. Download Spark: [spark-3.0.0-bin-hadoop3.tgz](#)

4. Verify this release using [SHA-256 checksums](#) or [PGP signatures](#). Please [see KEYS](#).

Note that, Spark 2.x is pre-built with Scala 2.11 except version 2.4.2, which is pre-built with Scala 2.12. Spark 3.0+ is pre-built with Scala 2.12.

```
pip3 install pyspark
pip3 install py4j
```

SPARK

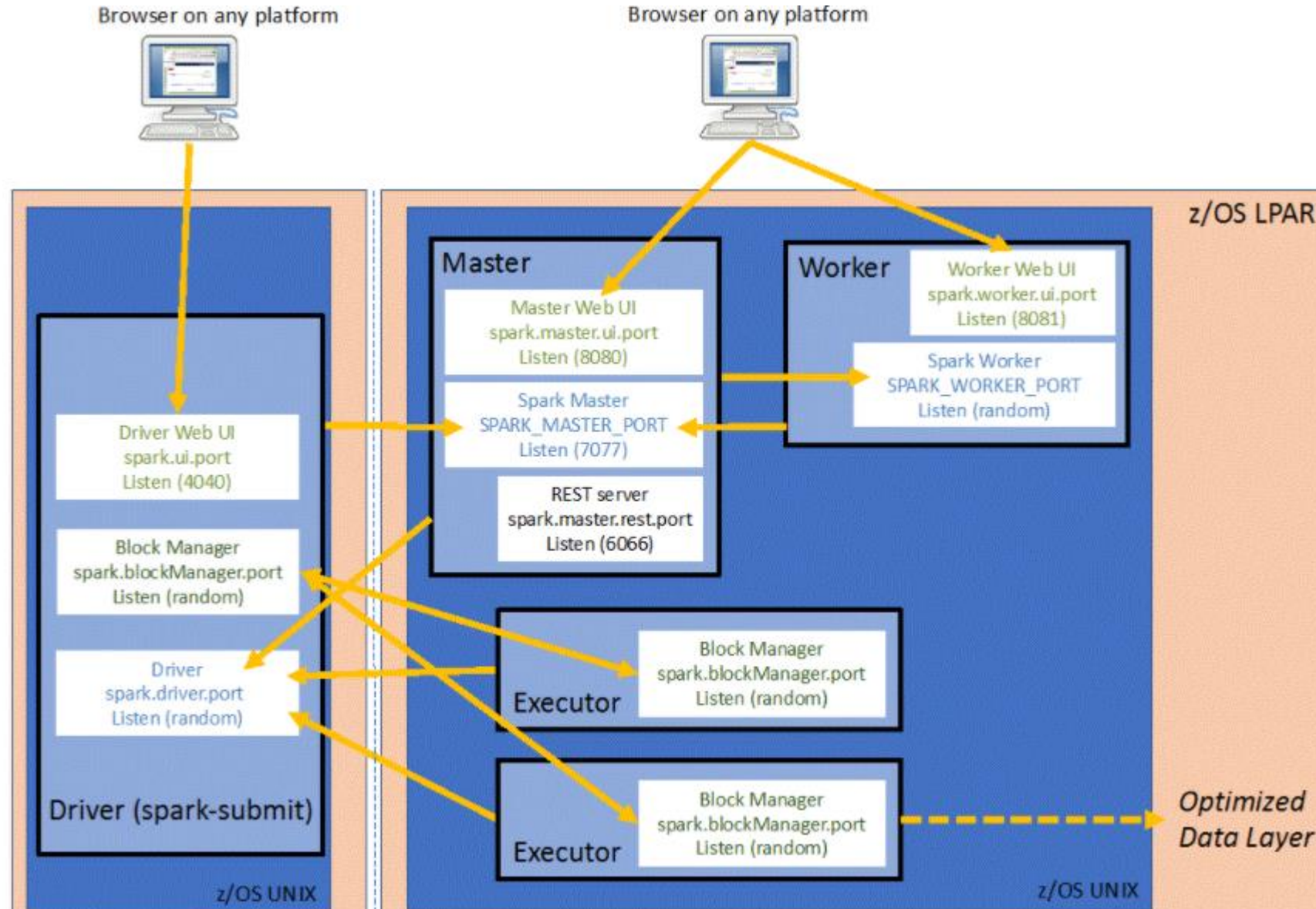
/opt/spark3/conf/spark-env.sh

```
SPARK_LOCAL_IP=cnt-cls-m1
SPARK_MASTER_IP=pub-cnt-cls-m1
SPARK_MASTER_HOST=cnt-cls-m1
SPARK_MASTER_PORT=7070
PYSPARK_PYTHON=/usr/bin/python3
PYSPARK_DRIVER_PYTHON=/usr/bin/python3
```


SPARK

`/opt/spark3/conf/spark-env.sh`

```
SPARK_LOCAL_IP=cnt-cls-m1
SPARK_MASTER_IP=pub-cnt-cls-m1
SPARK_MASTER_HOST=cnt-cls-m1
SPARK_MASTER_PORT=7070
PYSPARK_PYTHON=/usr/bin/python3
PYSPARK_DRIVER_PYTHON=/usr/bin/python3
```

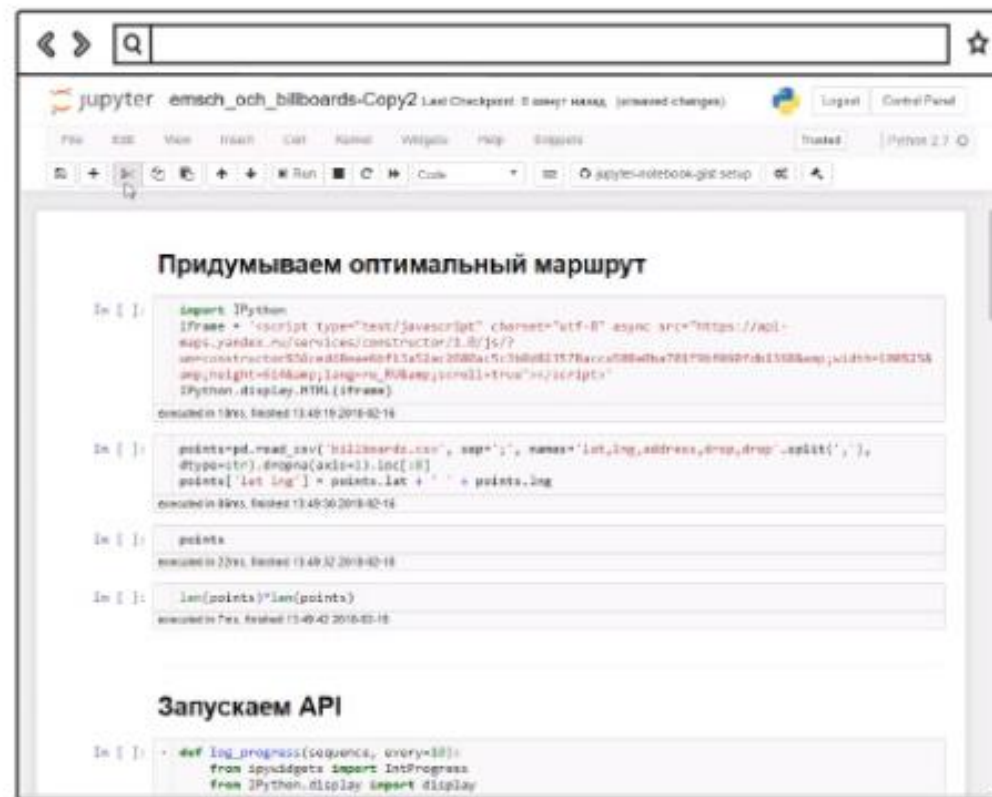


JUPYTERHUB

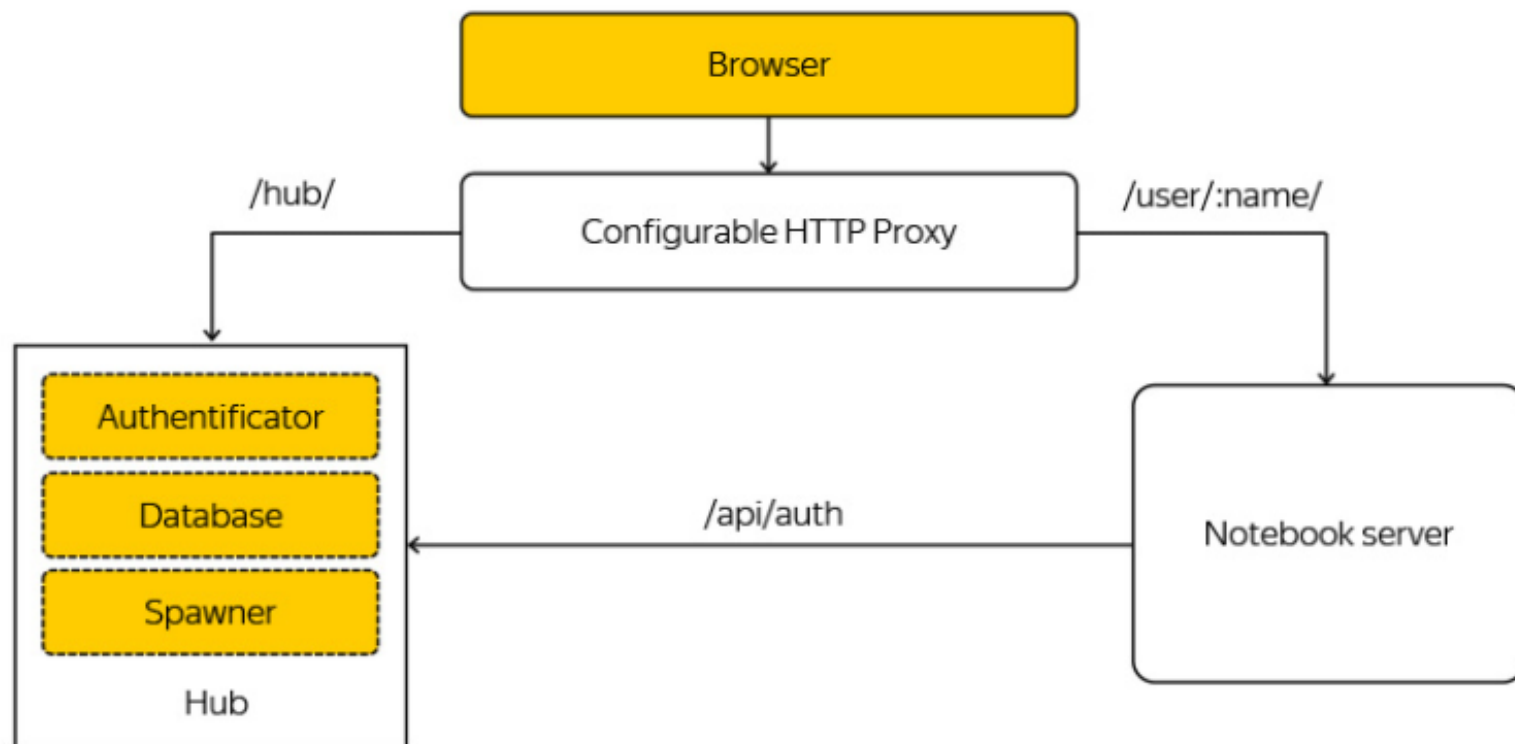


```
yum install install npm nodejs-legacy
pip3 install jupyterhub
npm install -g configurable-http-proxy
```

- › Классический «ноутбук»
- › Различные языки программирования
- › Интерактивный код, легко менять на лету
- › Визуализации, произвольный output



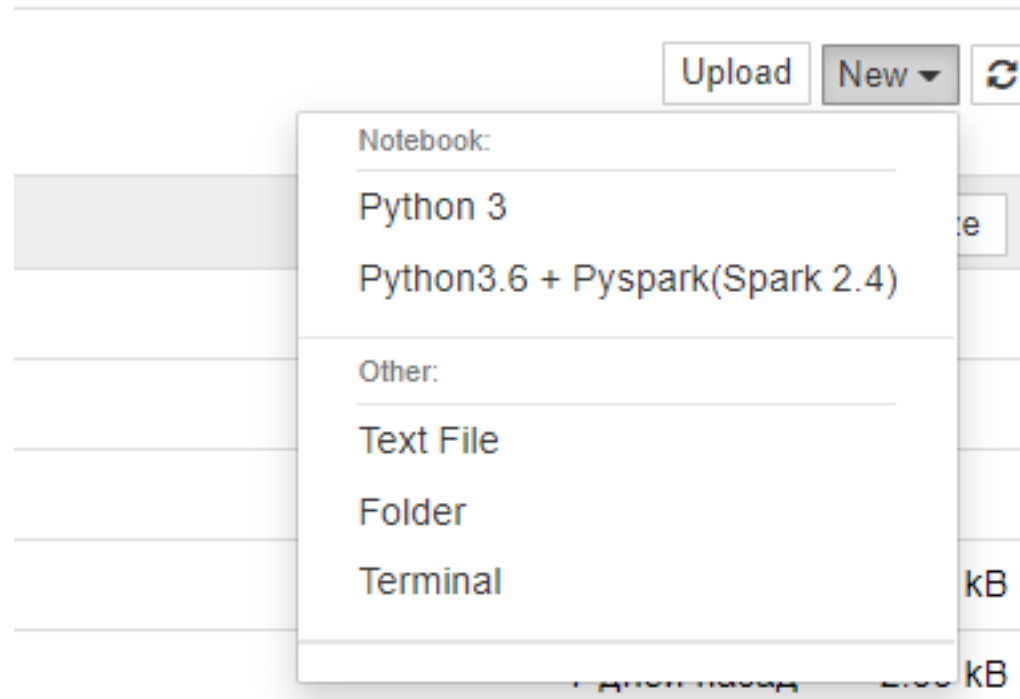
JUPYTERHUB



JUPYTERHUB

```
# базовый путь и публичный IP адрес для хаба
c.JupyterHub.base_url = '/'
c.JupyterHub.bind_url = 'http://pub-cnt-cls-m1:8765'
# если планируется использование более чем для 1 пользователя
c.JupyterHub.spawner_class = 'jupyterhub.spawner.SimpleLocalProcessSpawner'
c.Spawner.args = ['--allow-root', '--debug', '--profile=PHYS131']
# пользователь в linux- это пользователь в jupyterhub
c.Authenticator.admin_users = {'добавляем админов кластера',}
c.Authenticator.whitelist = {'список пользователей Linux, которые будут заходить на jupyterhub'}
# так как у нас кластер на внутренней сети, то добавляем параметр прокси
# localhost (127.0.0.1) меняем на внутреннюю сеть
c.ConfigurableHTTPProxy.api_url='http://10.0.0.2:8108'
c.JupyterHub.proxy_api_ip = '10.0.0.2'
c.JupyterHub.proxy_api_port = 5678
c.JupyterHub.hub_ip = '10.0.0.2'
c.JupyterHub.hub_port = 5678
# переменные среды для spark окружения в jupyterhub
c.YarnSpawner.environment = {
    'PYTHONPATH': 'opt/spark3/python',
    'SPARK_CONF_DIR': '/opt/spark3/conf'
}
```

JUPYTERHUB KERNEL



JUPYTERHUB KERNEL

/usr/share/jupyter/kernels/

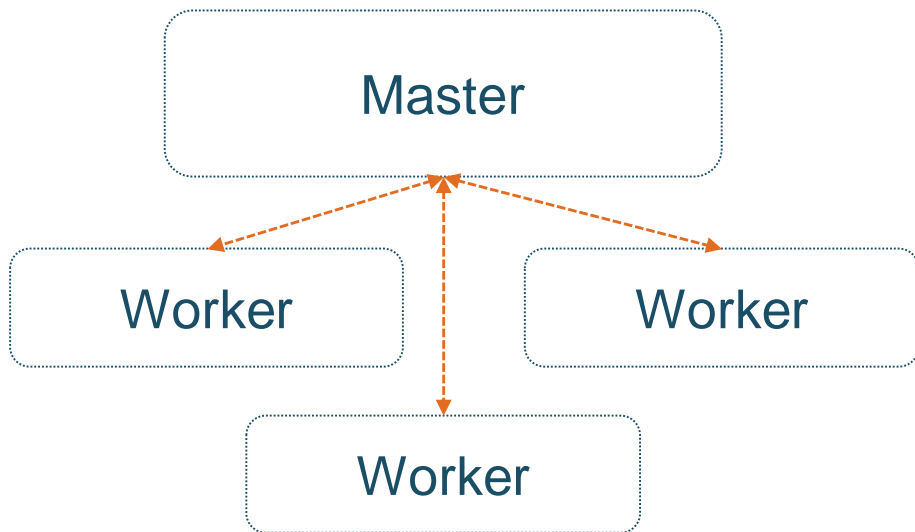
```
{
  "argv": [
    "python3.6",
    "-m",
    "ipykernel_launcher",
    "-f",
    "{connection_file}"
  ],
  "display_name": "Python3.6 + Pyspark(Spark 3.0)",
  "language": "python",
  "env": {
    "PYSPARK_PYTHON": "/usr/bin/python3.6",
    "SPARK_HOME": "/opt/spark3",
    "HADOOP_CONF_DIR": "/etc/spark3/conf/yarn-conf",
    "HADOOP_CLIENT_OPTS": "-Xmx2147483648 -XX:MaxPermSize=512M -Djava.net.preferIPv4Stack=true",
    "PYTHONPATH": "/opt/spark3/python/lib/py4j-0.10.4-src.zip:/opt/spark3/python/",
    "PYTHONSTARTUP": "/opt/spark3/python/pyspark/shell.py",
    "PYSPARK_SUBMIT_ARGS": " --master yarn --deploy-mode client pyspark-shell"
  }
}
```


SPARK | HADOOP
CLUSTER

.... RETURN TO



ПОЯВИЛСЯ КЛАСТЕР и добавил проблем

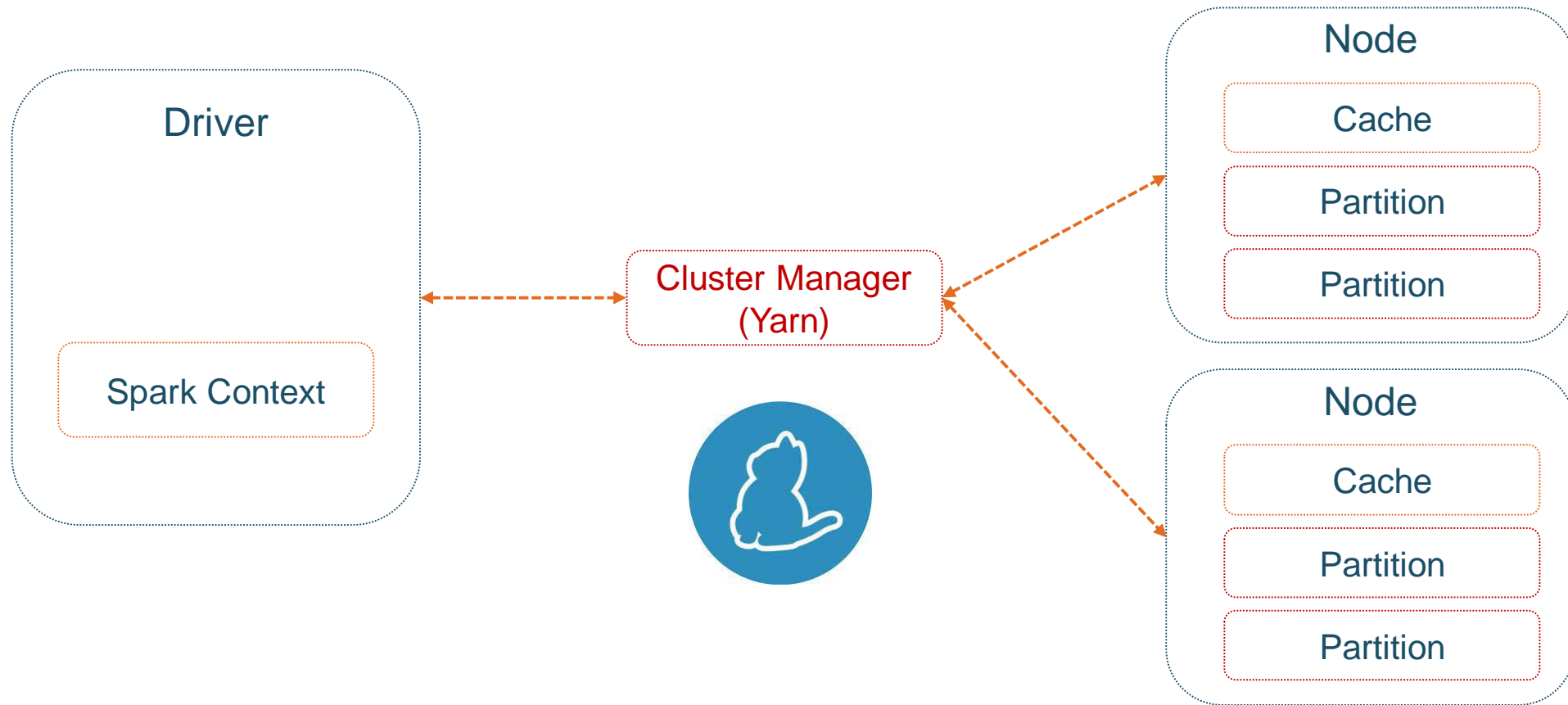


Проблемы
координации

Проблемы
коммуникации

Проблемы
стабильности

НА КЛАСТЕРЕ ДРУГОЕ УПРАВЛЕНИЕ

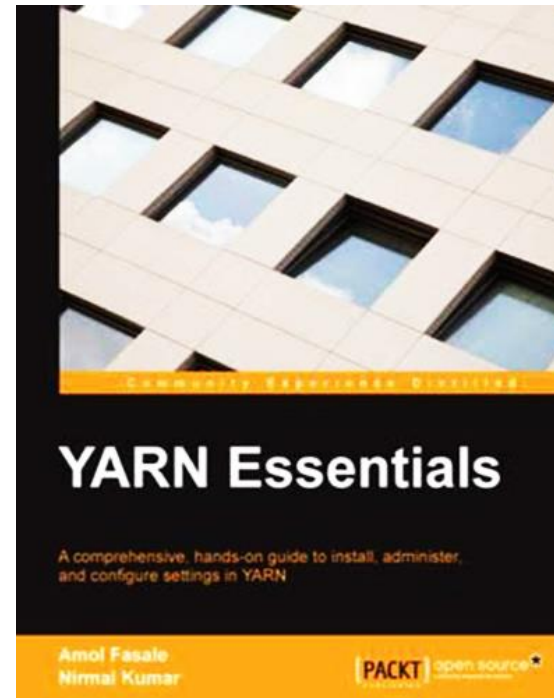


НА КЛАСТЕРЕ ДРУГОЕ УПРАВЛЕНИЕ



- `yarn app -list`
- `yarn app -status`
- `yarn app -appStates`
- `yarn app -destroy appId`
- `yarn app -kill appId`

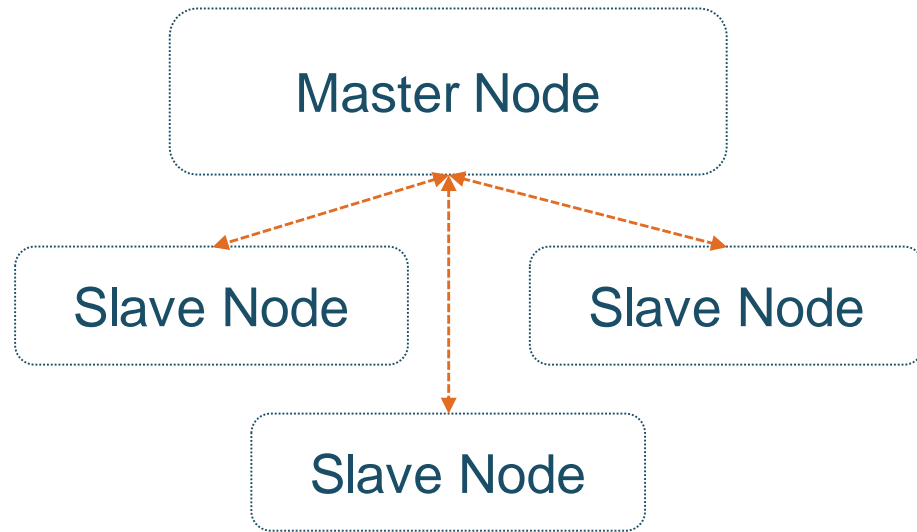
НА КЛАСТЕРЕ ДРУГОЕ УПРАВЛЕНИЕ



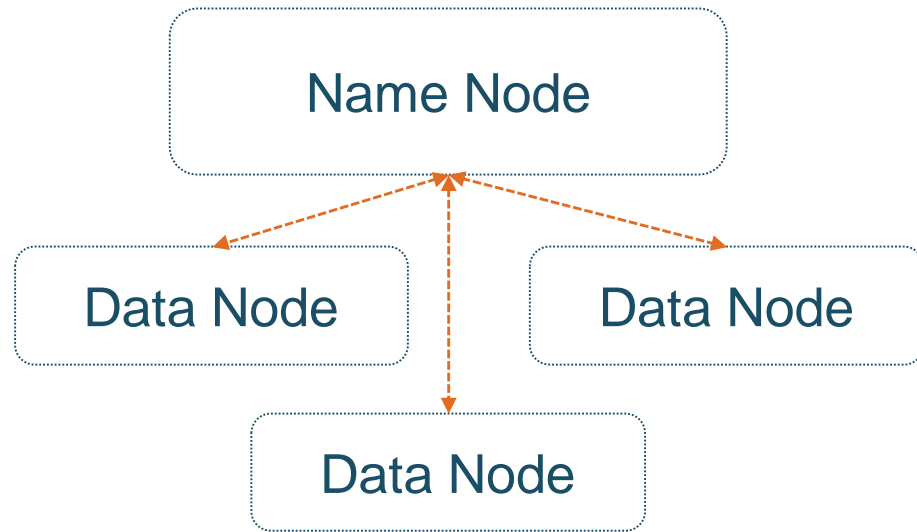
SPARK | HADOOP CLUSTER



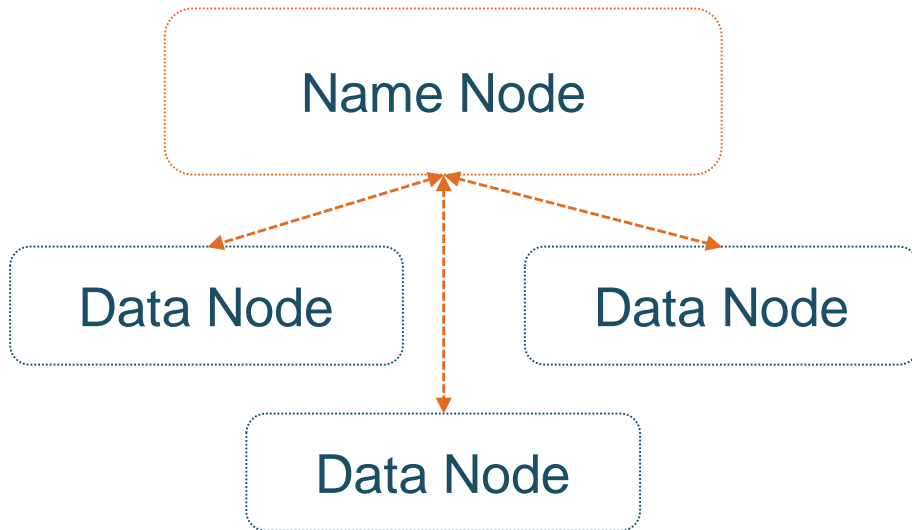
HDFS



HDFS



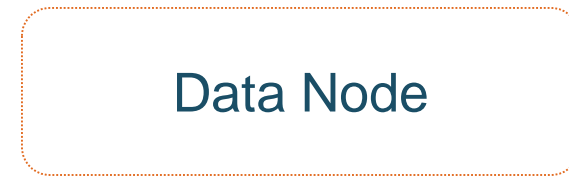
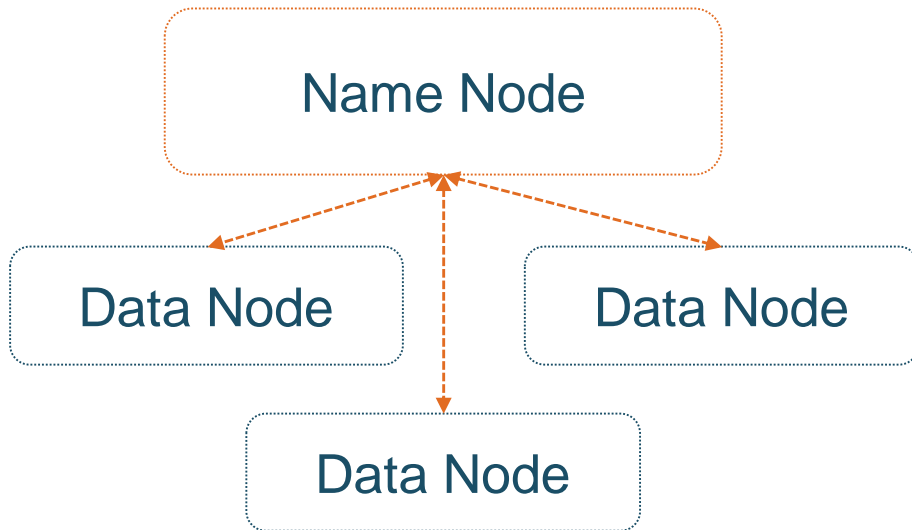
HDFS



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

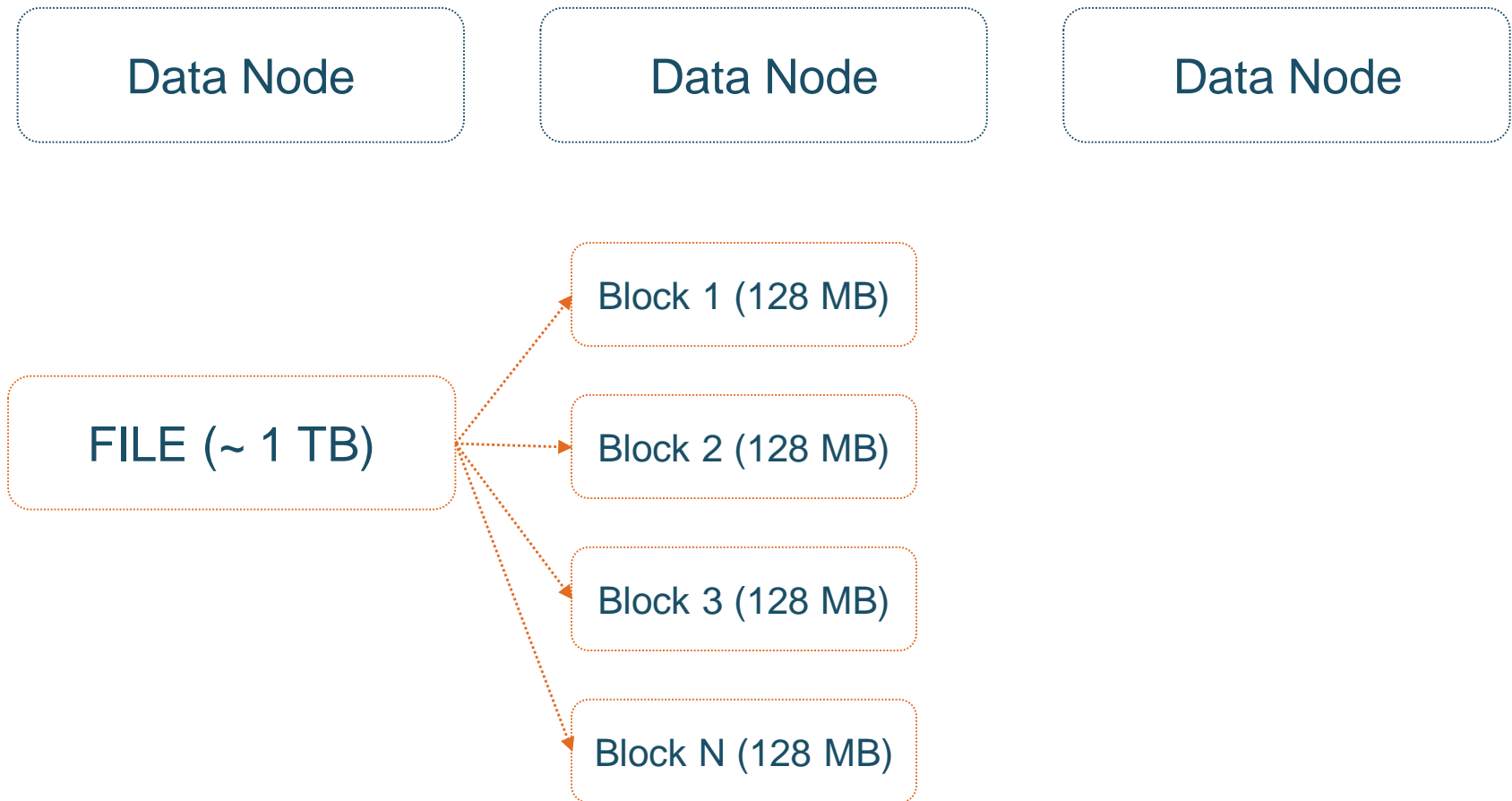
Data Node

Data Node

FILE (~ 1 TB)

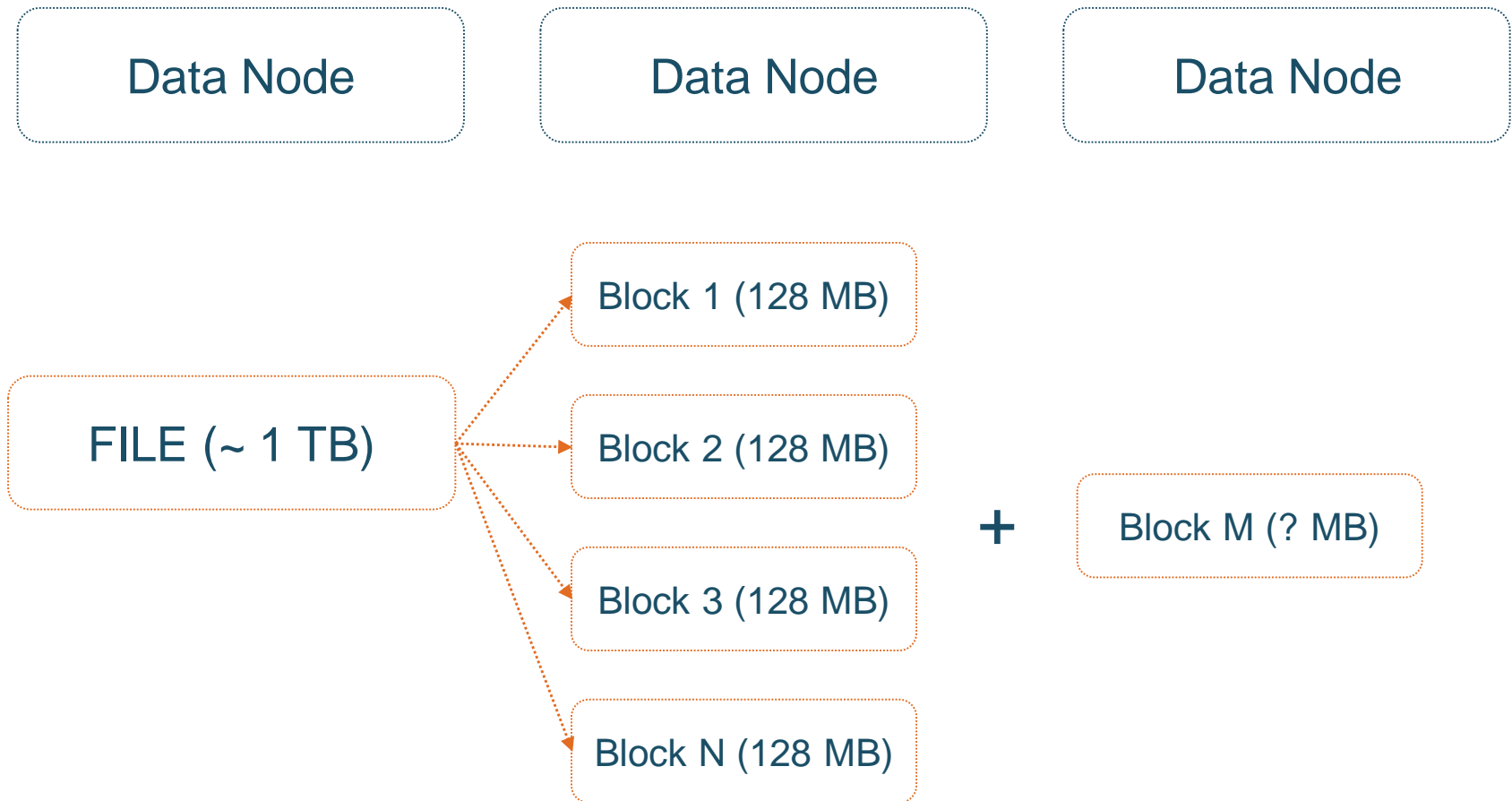
HDFS

ОСОБЕННОСТЬ
ХРАНЕНИЯ



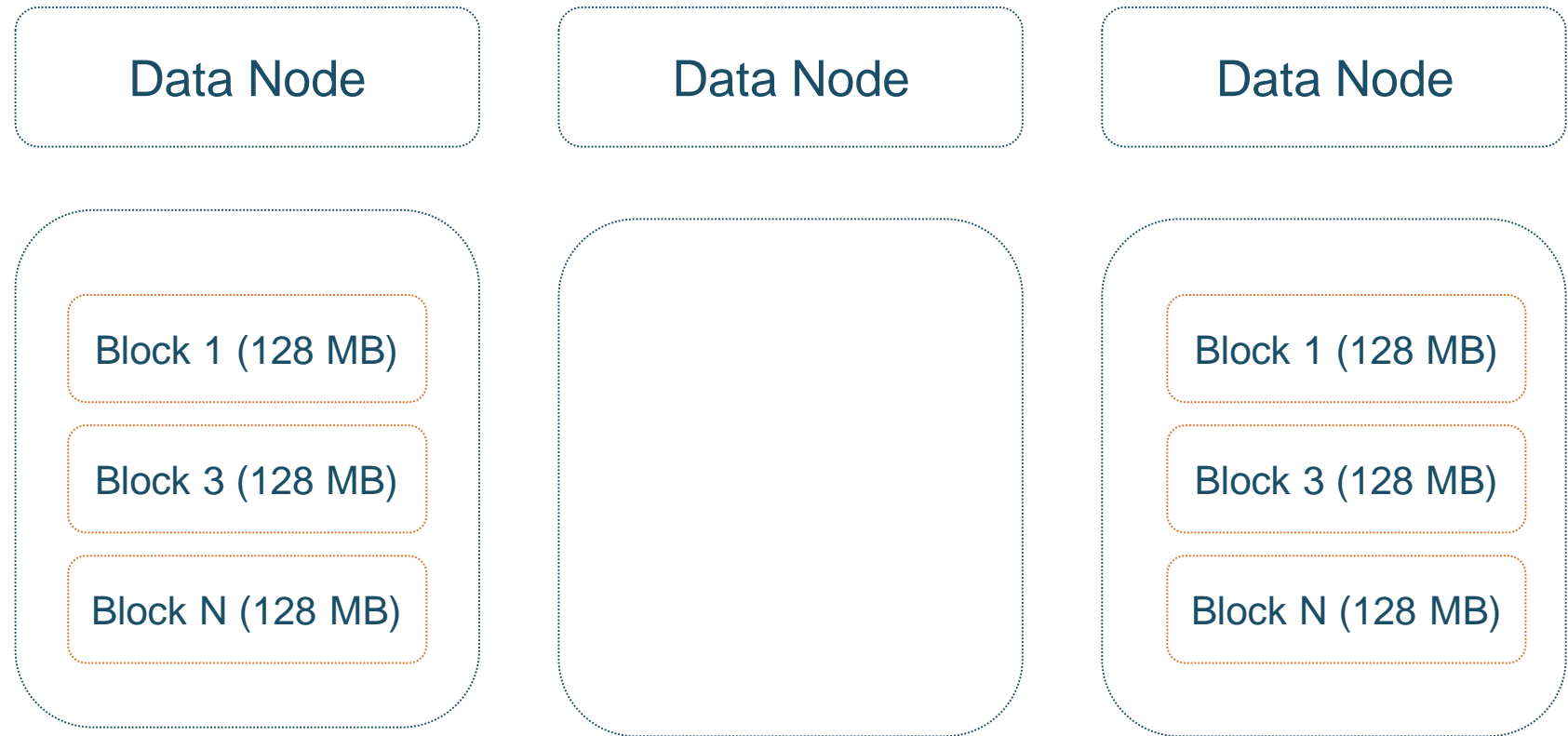
HDFS

ОСОБЕННОСТЬ
ХРАНЕНИЯ



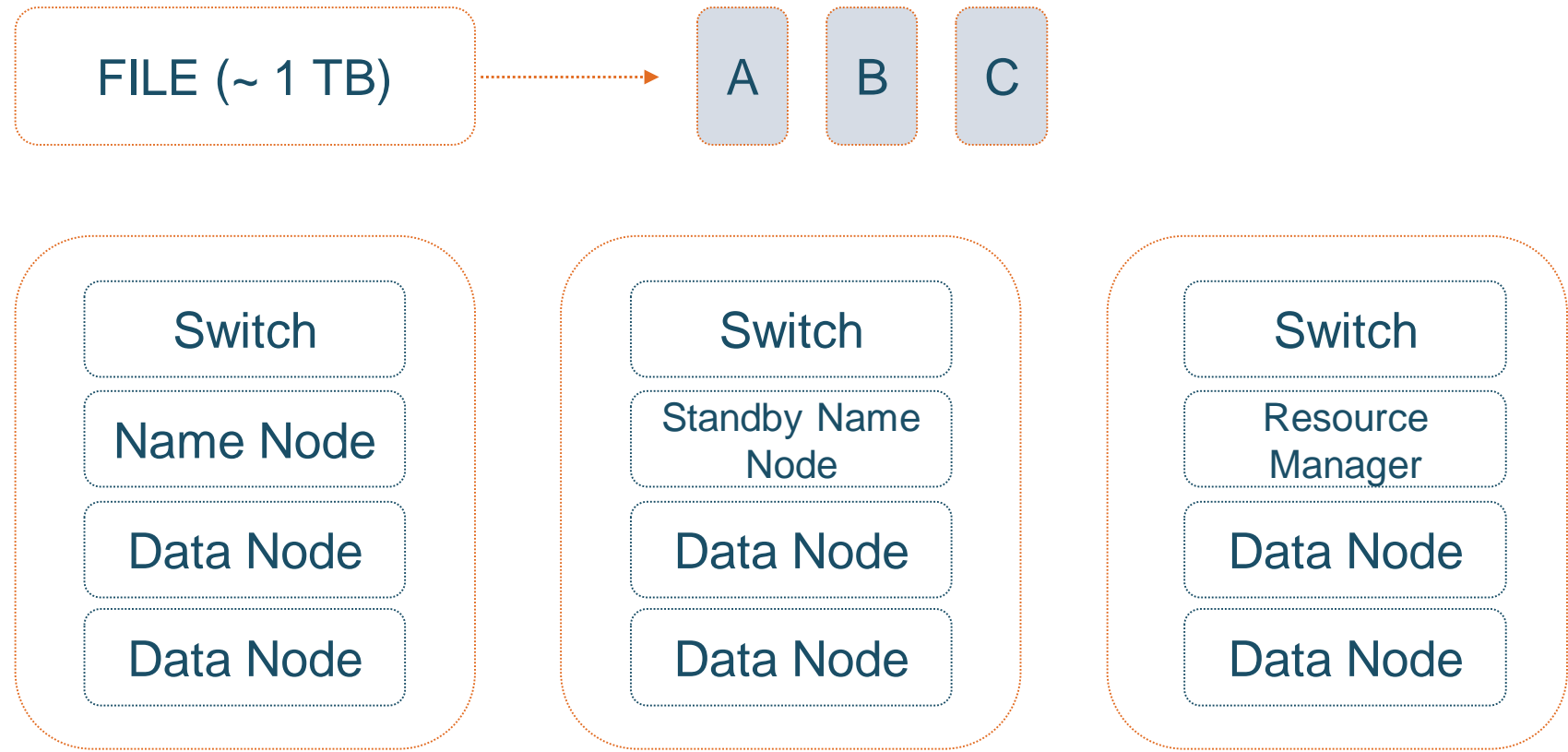
HDFS

ОСОБЕННОСТЬ
ХРАНЕНИЯ



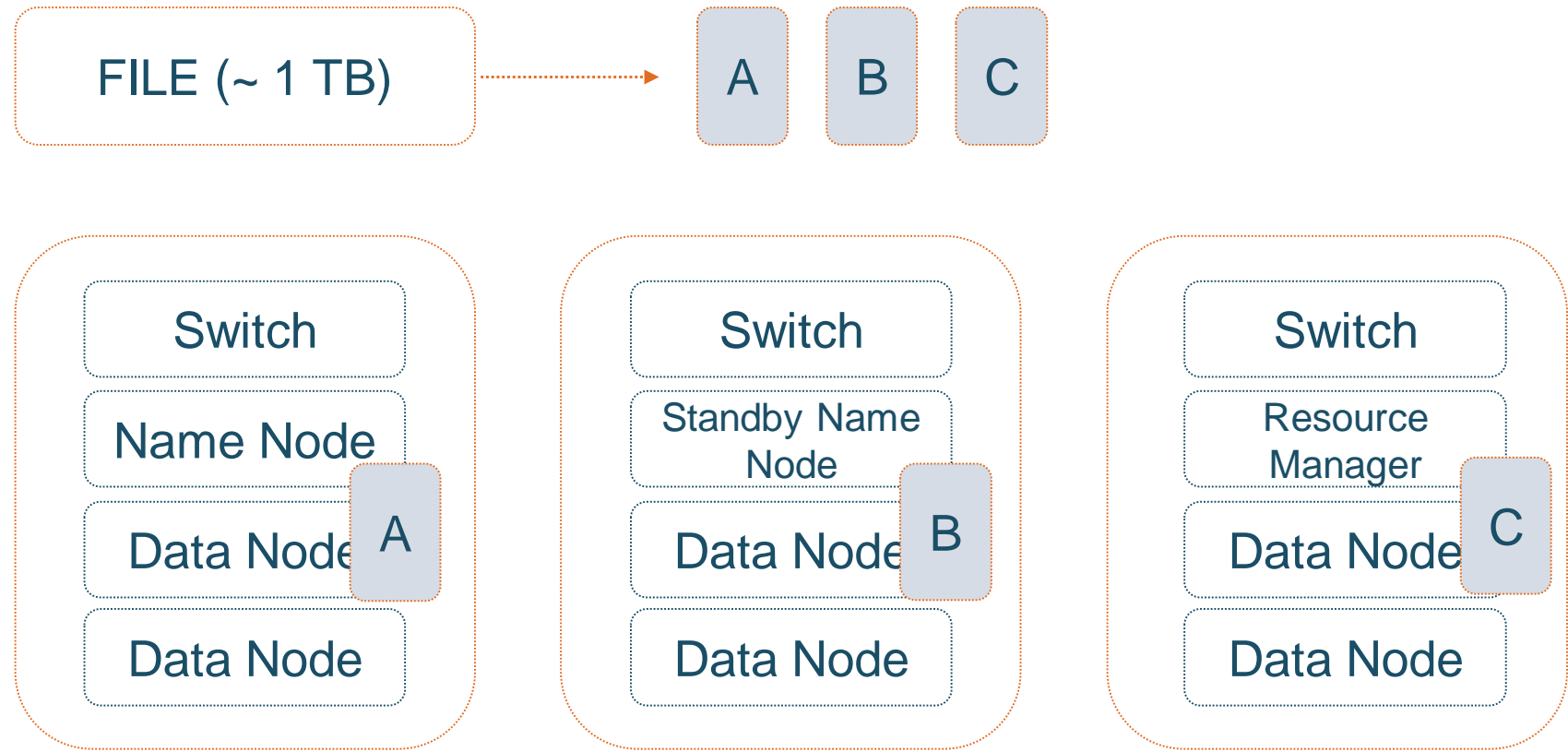
HDFS – RACK AWARENESS

ОСОБЕННОСТЬ
ХРАНЕНИЯ



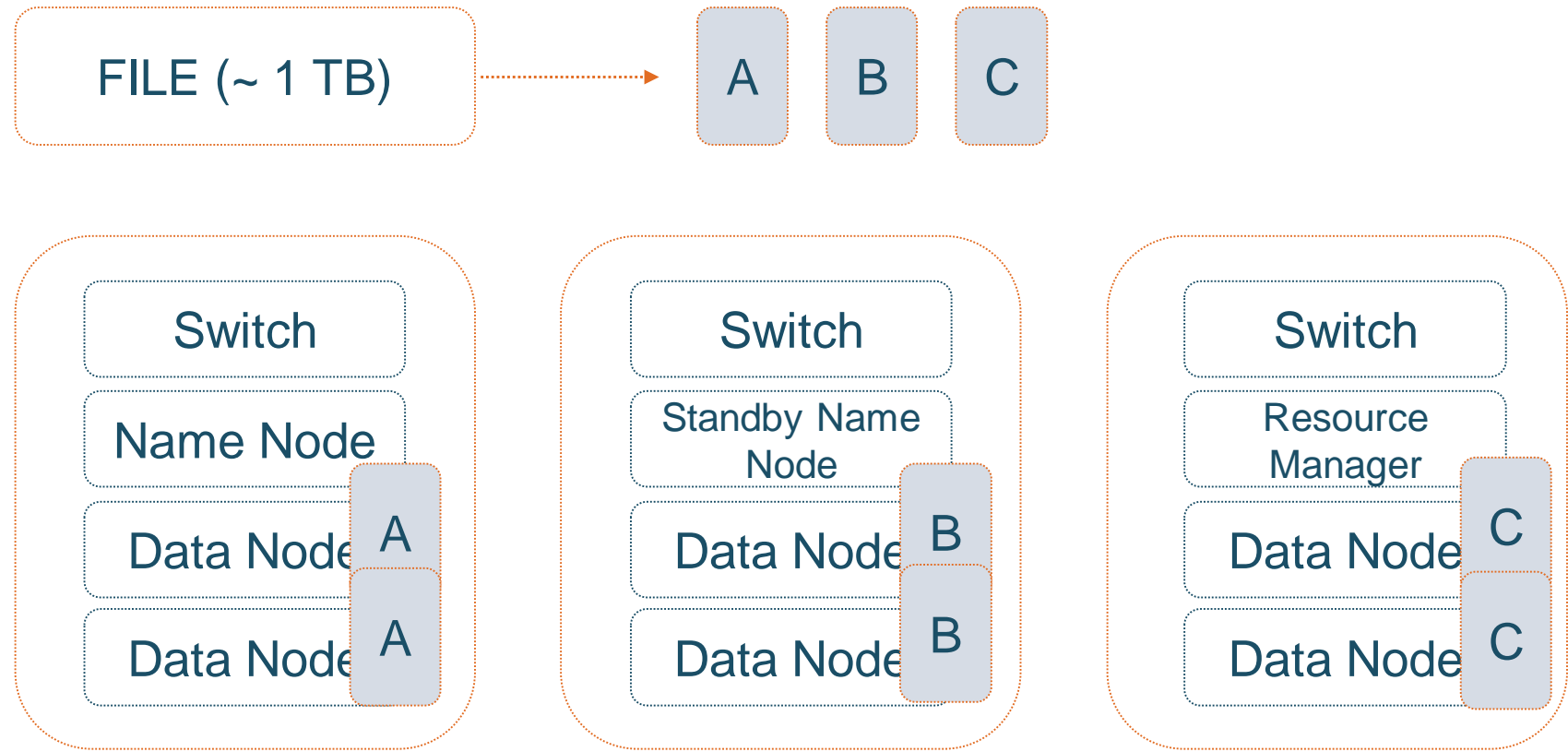
HDFS – RACK AWARENESS

Rep.1
—
на разных узлах



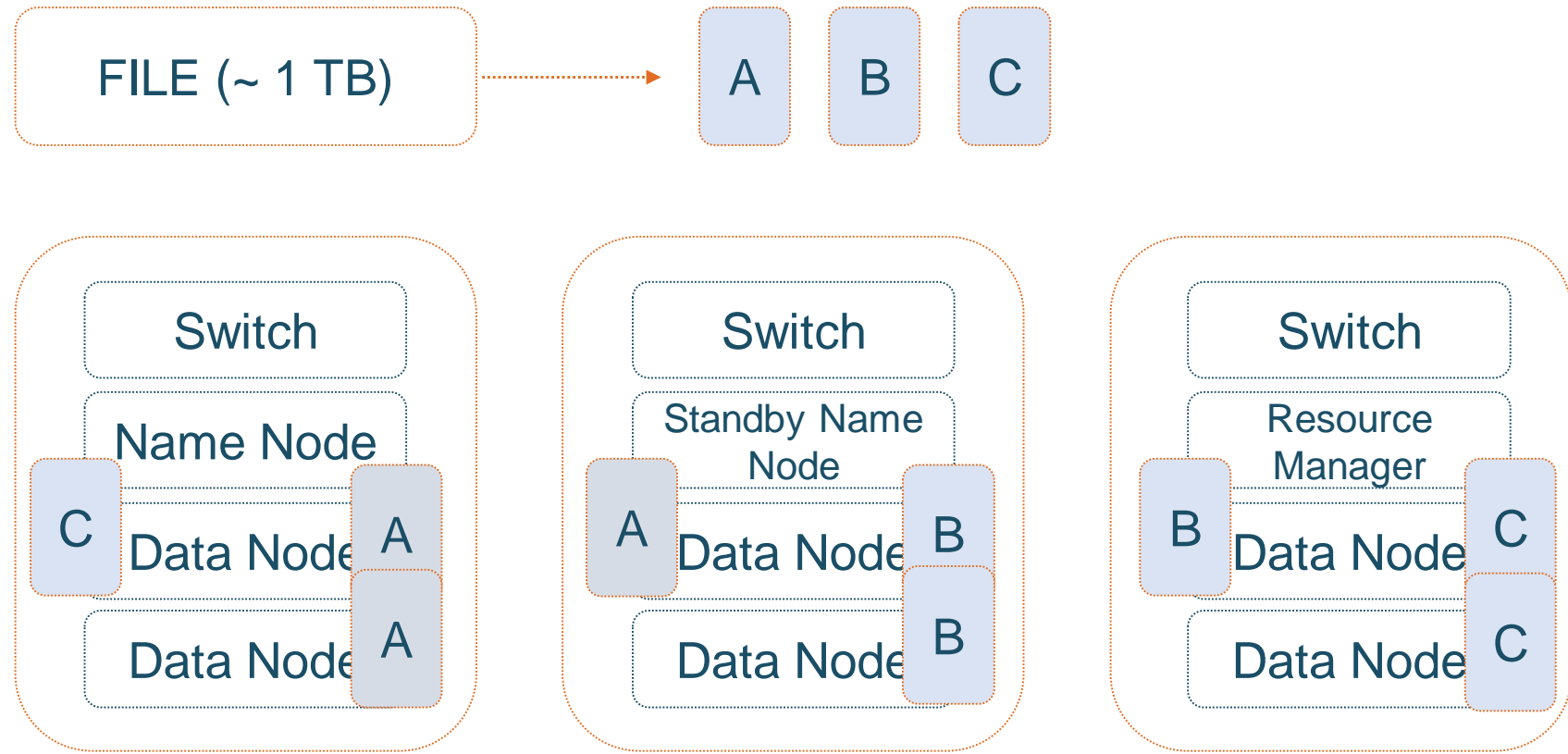
HDFS – RACK AWARENESS

Rep.2
—
на разных нодах внутри



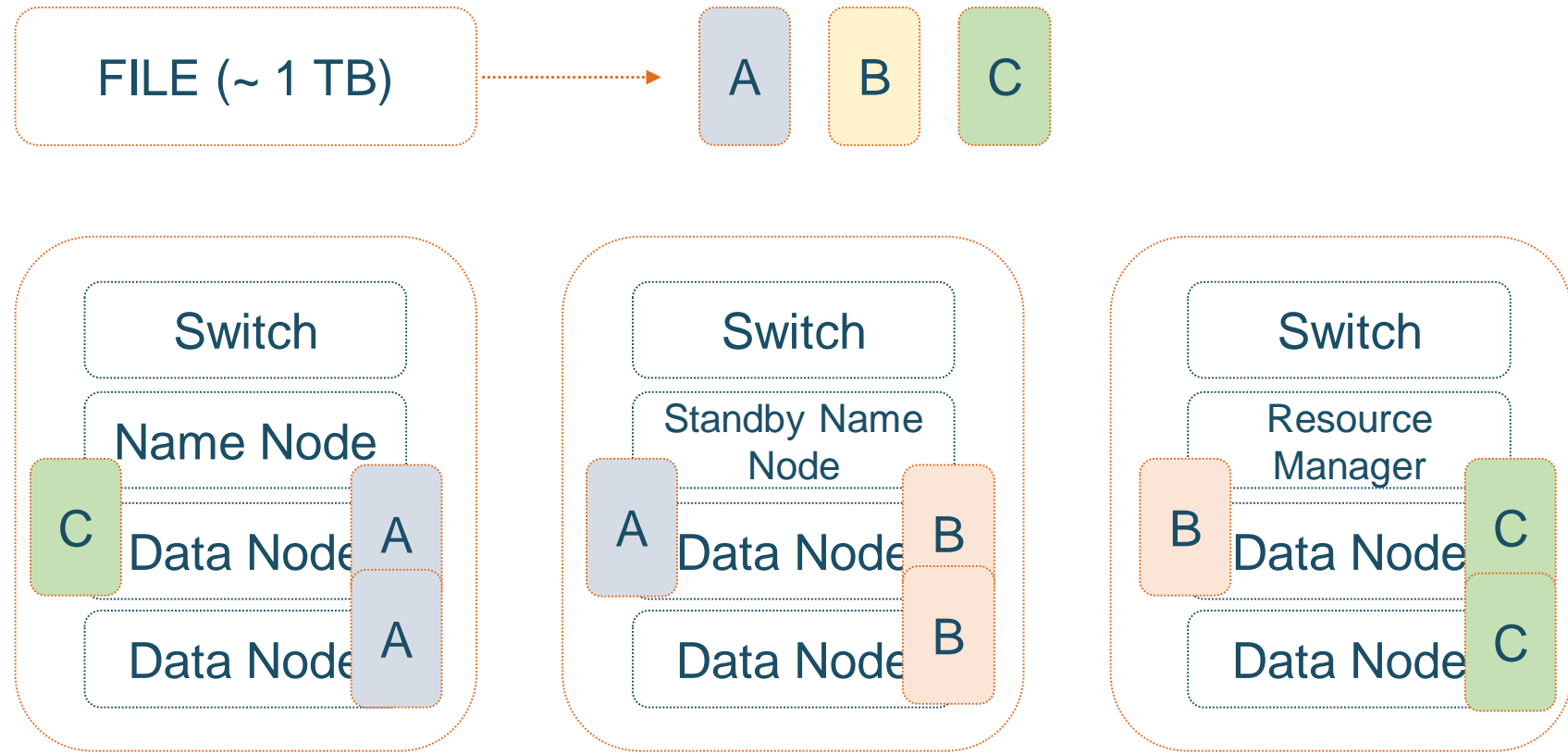
HDFS – RACK AWARENESS

Rep.3
—
на разных стойках



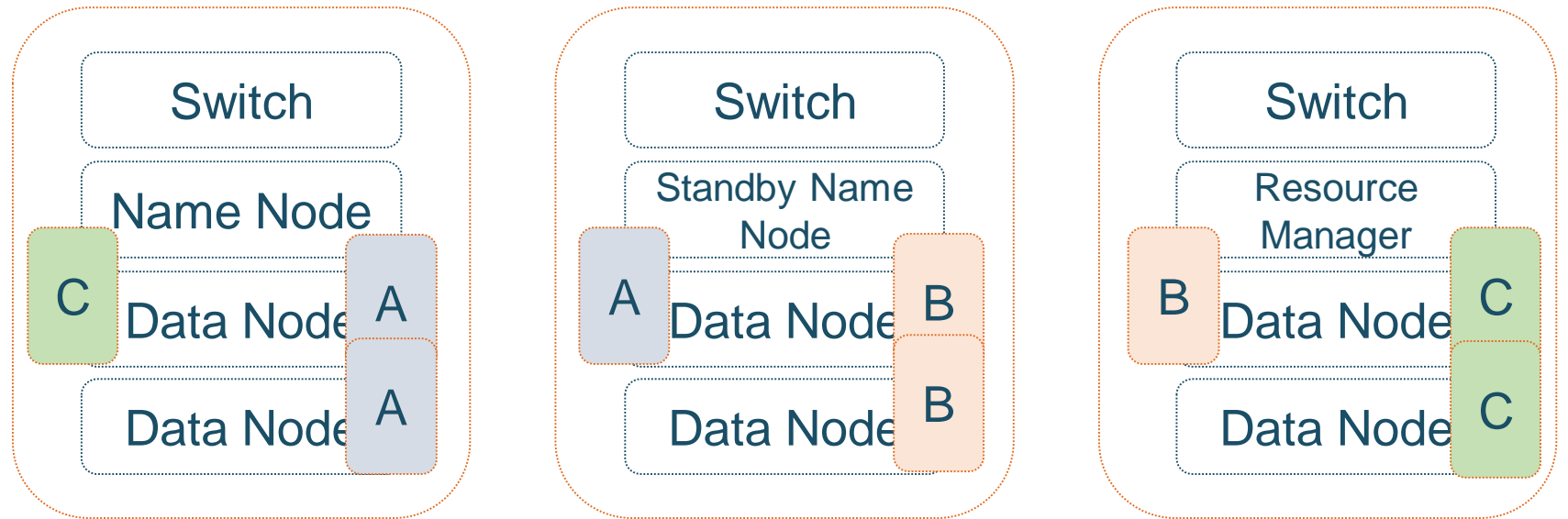
HDFS – RACK AWARENESS

Rep.3
—
на разных стойках



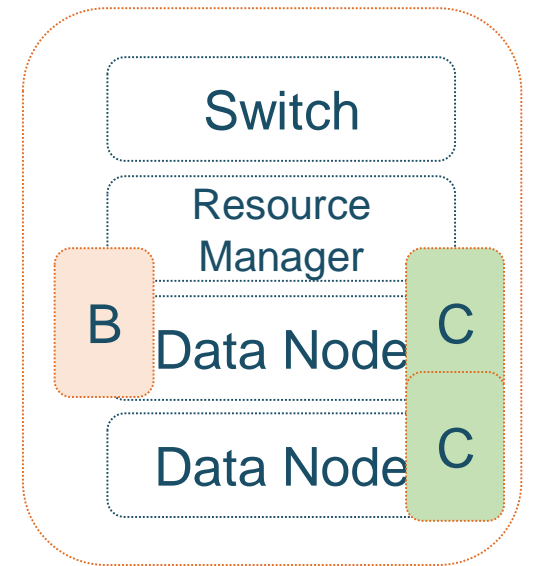
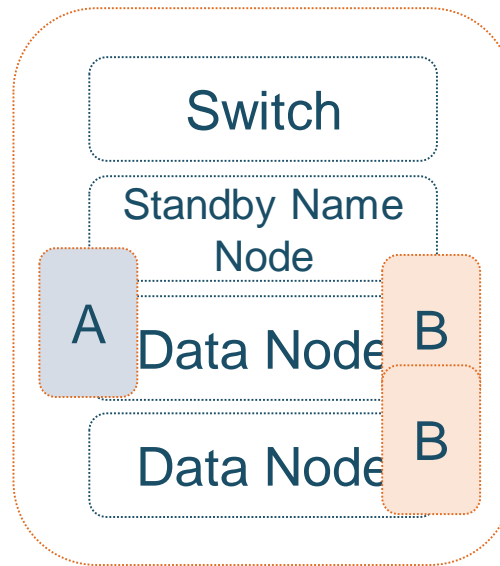
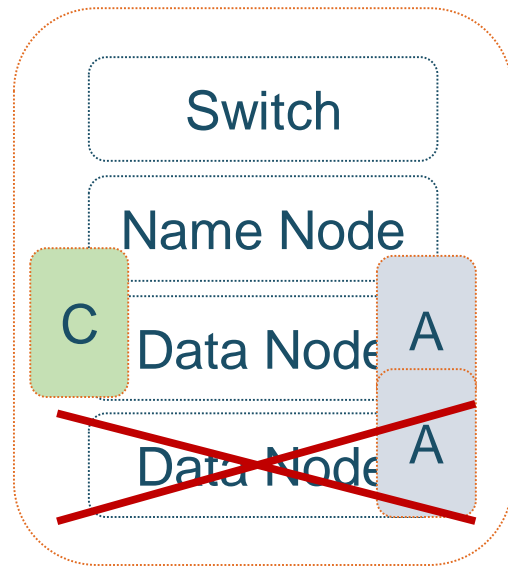
HDFS – FAULT TOLERANCE

Данные
доступны
всегда



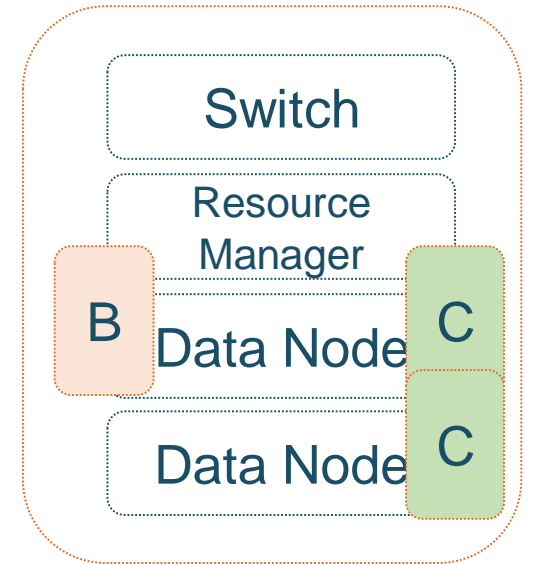
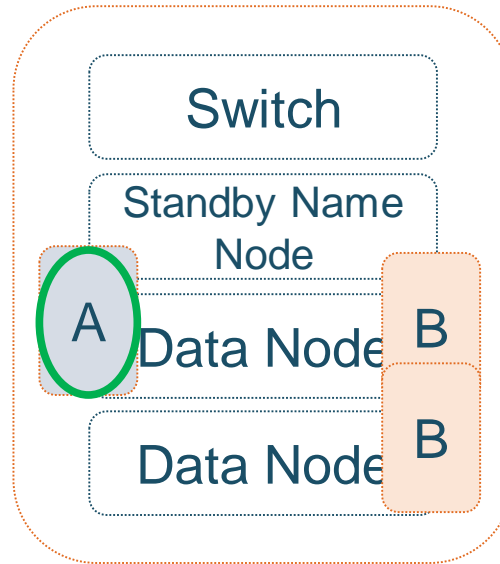
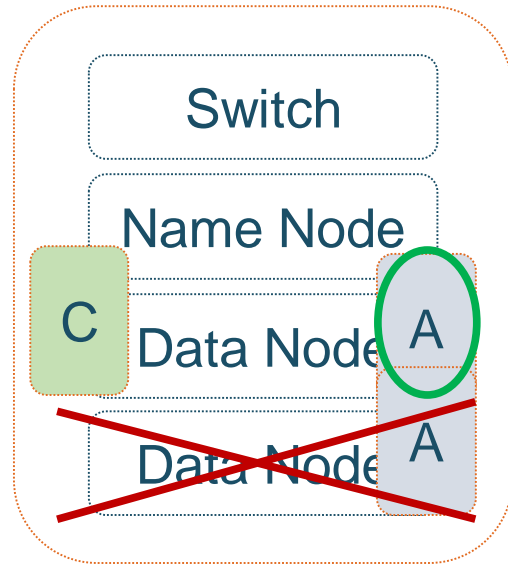
HDFS – FAULT TOLERANCE

Данные
доступны
всегда



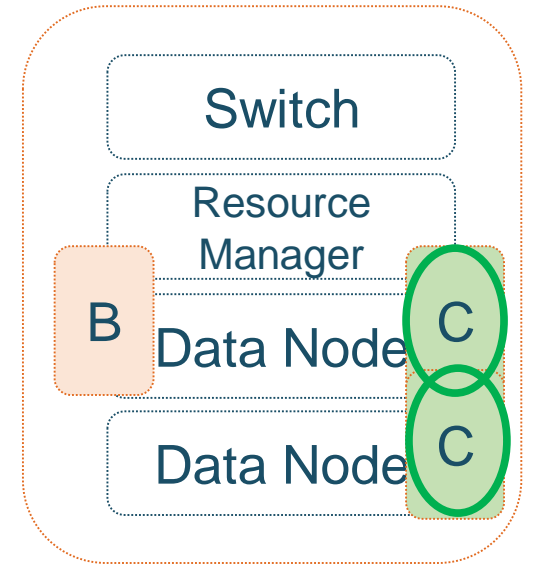
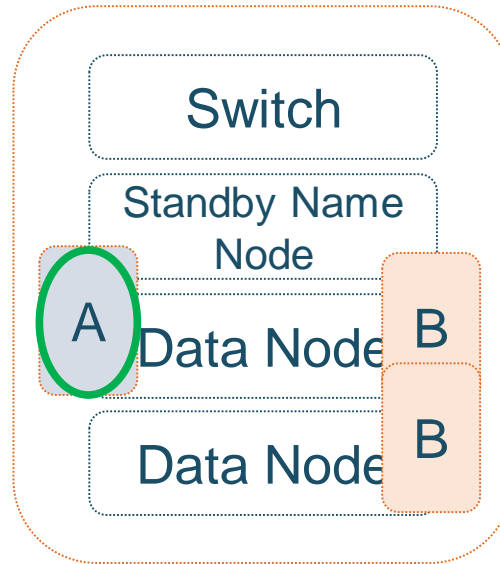
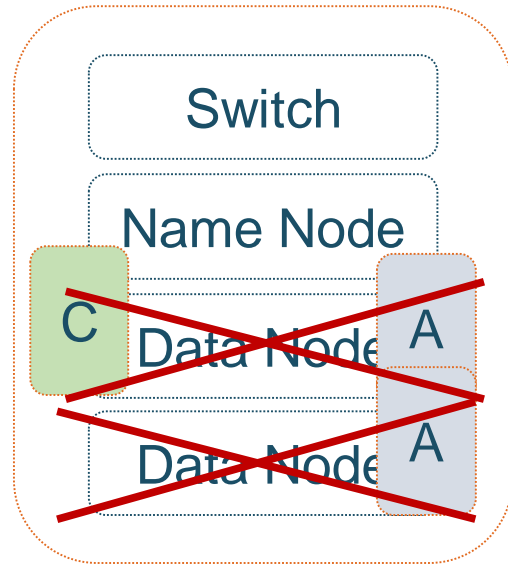
HDFS – FAULT TOLERANCE

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			

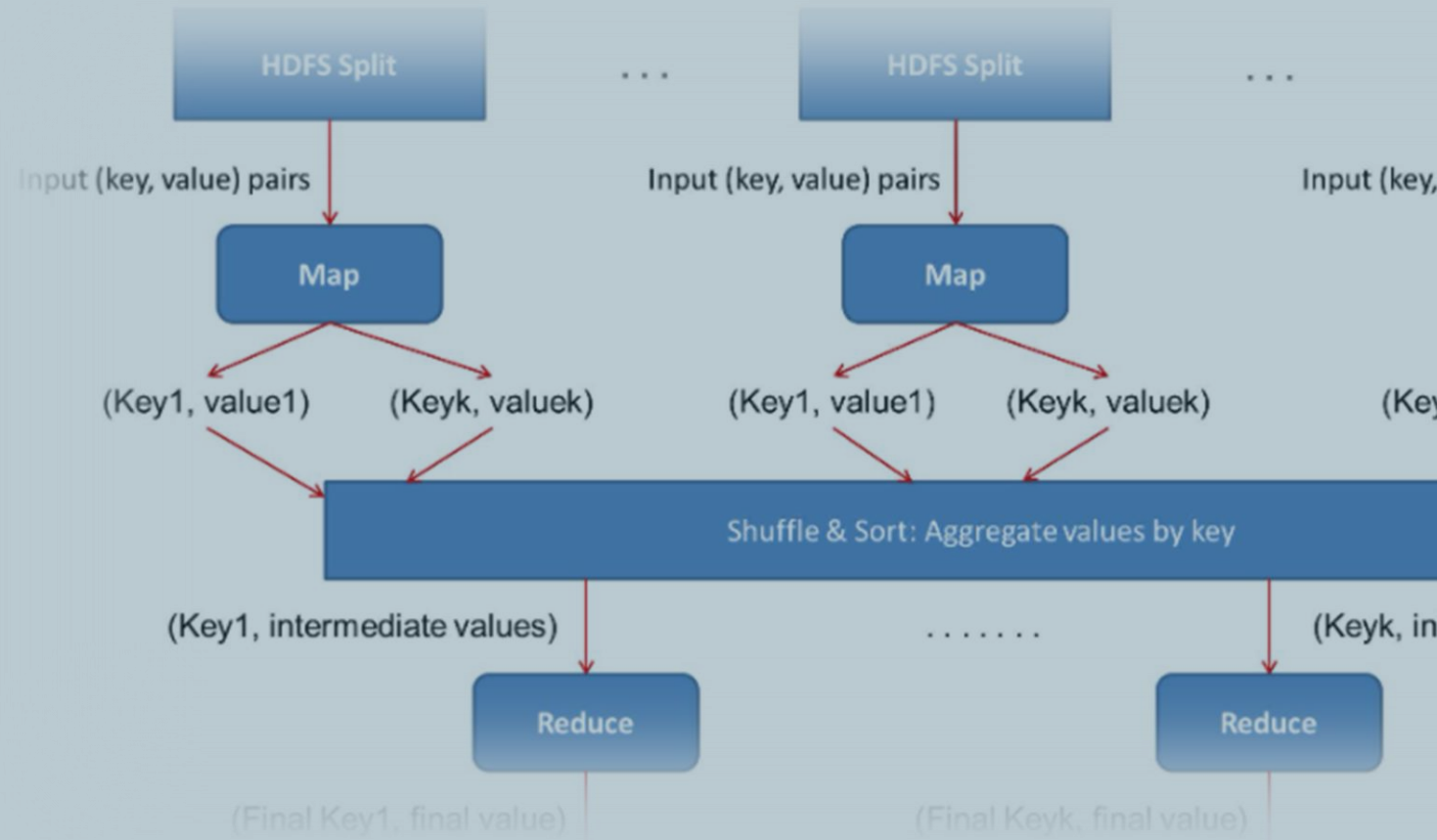
- $\% \text{ плотности} = \text{Ideal} - \% \text{ использования}$

HDFS | БАЛАНСИРОВКА ДИСКОВ

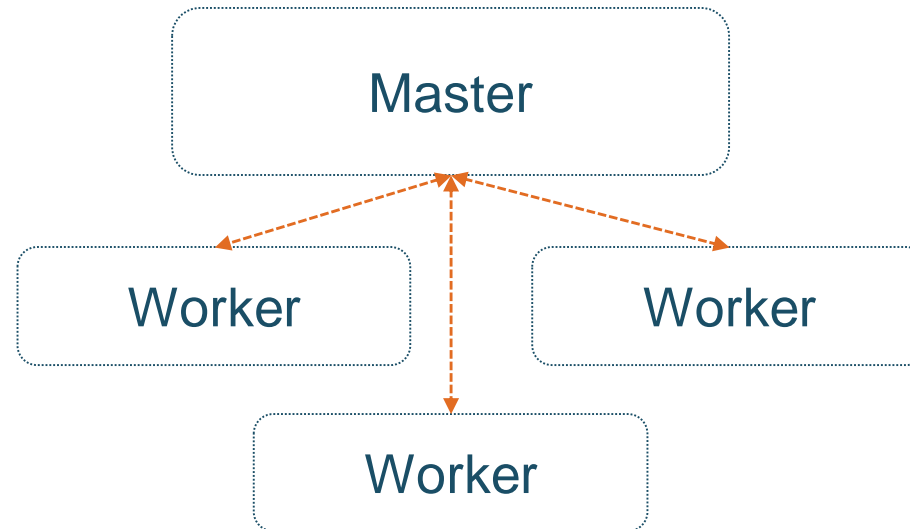
Нормализуем

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

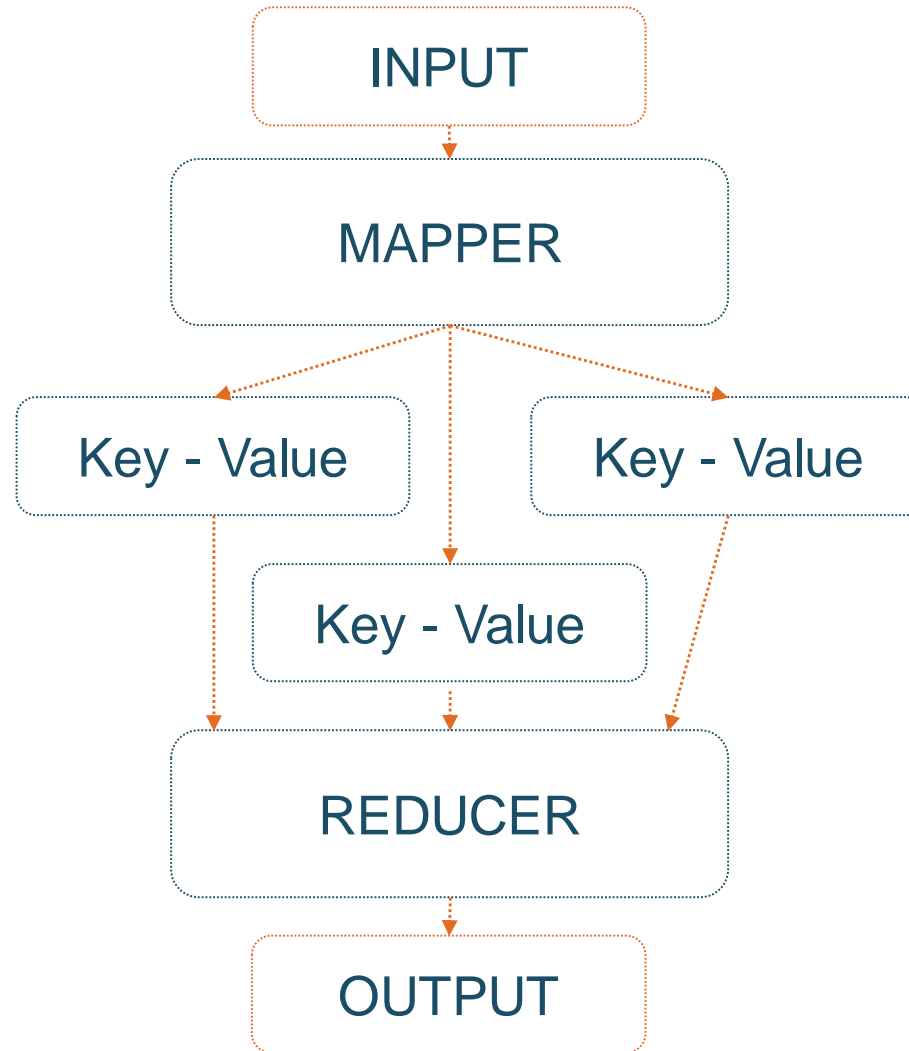
MAP - REDUCE



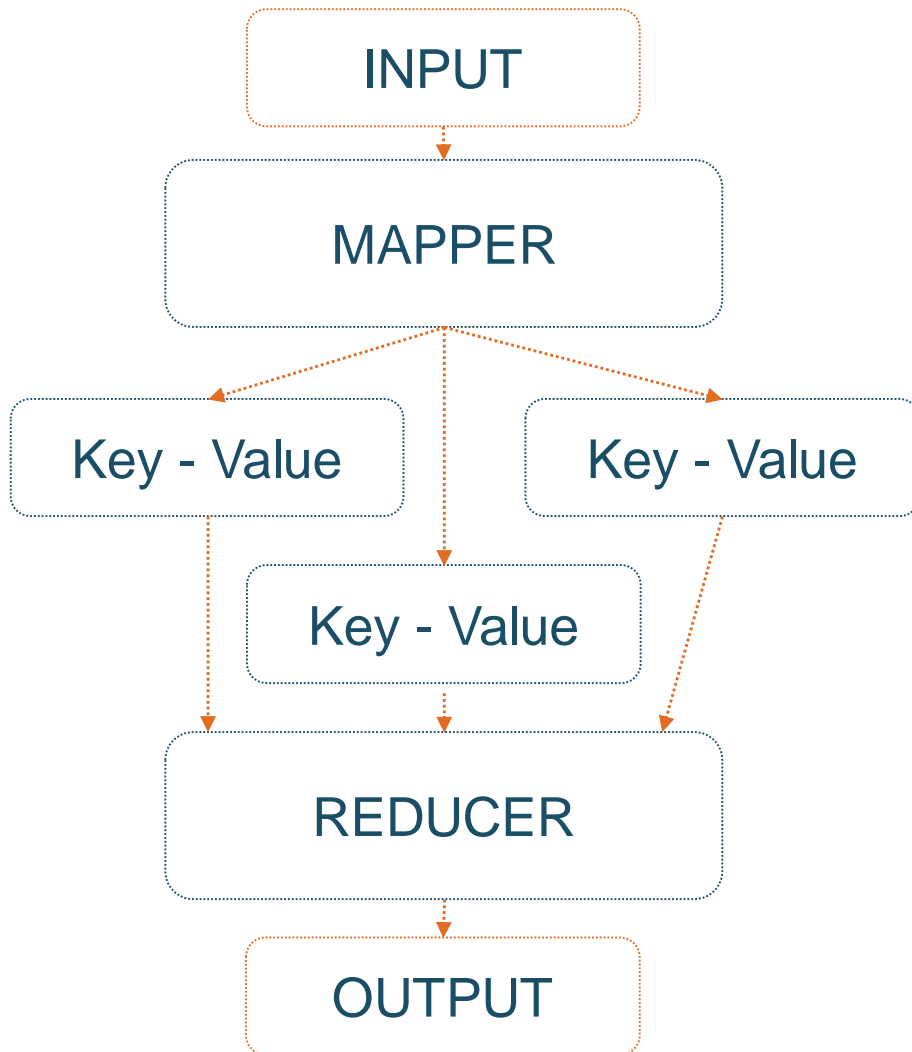
MAP – REDUCE



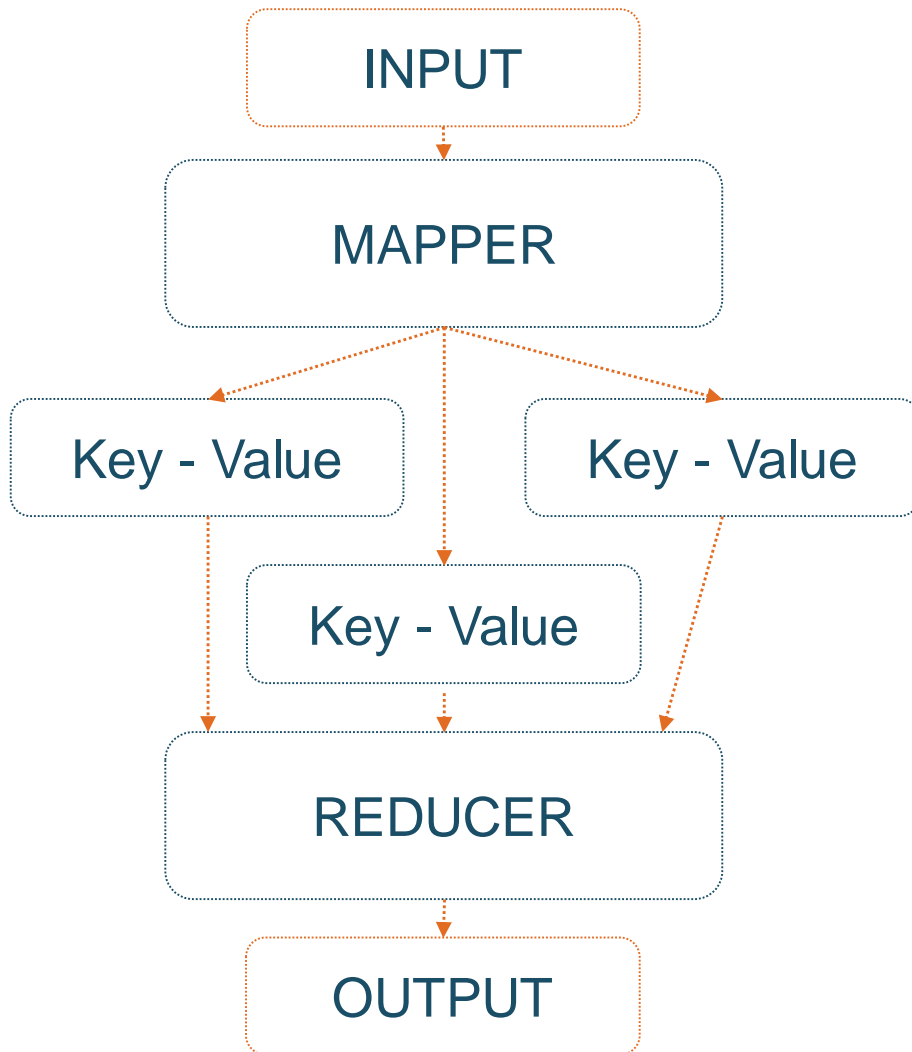
MAP – REDUCE



MAP – REDUCE



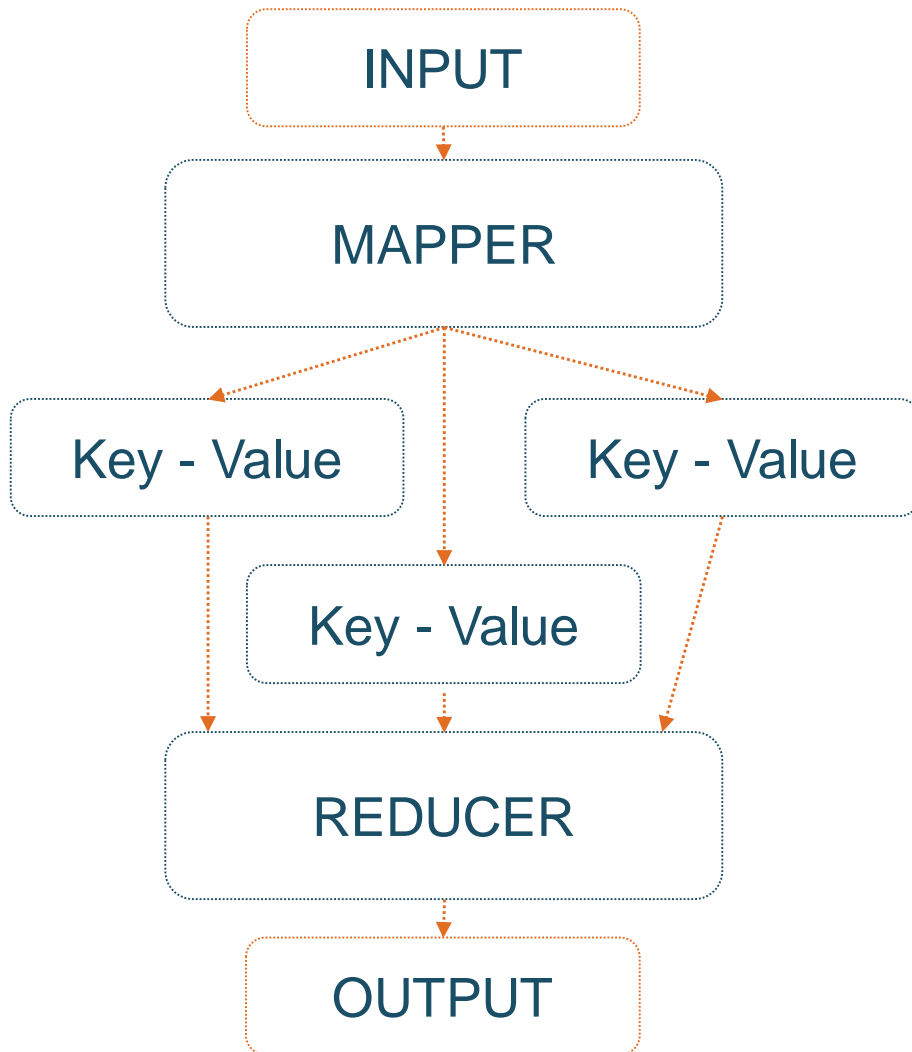
MAP – REDUCE



INPUT

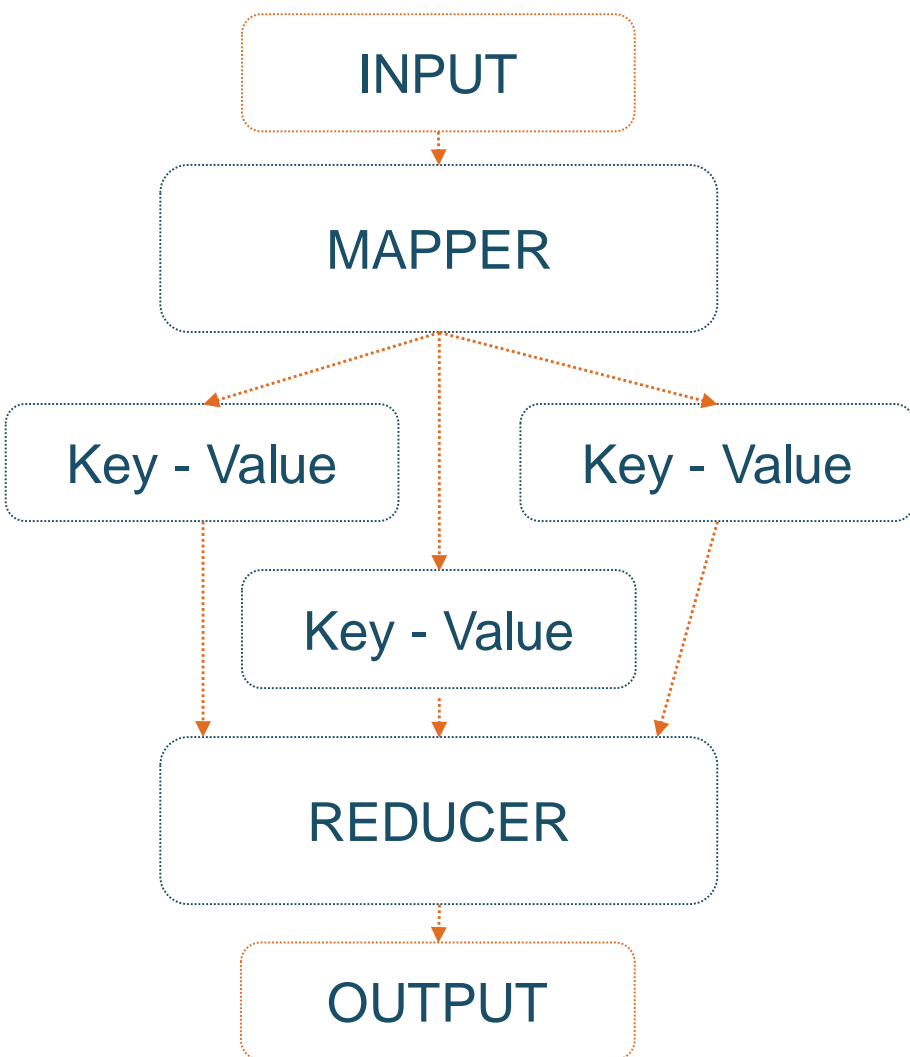
обычный файл\тс вашими\тданными

MAP – REDUCE



обычный файл\тс вашими\тданными

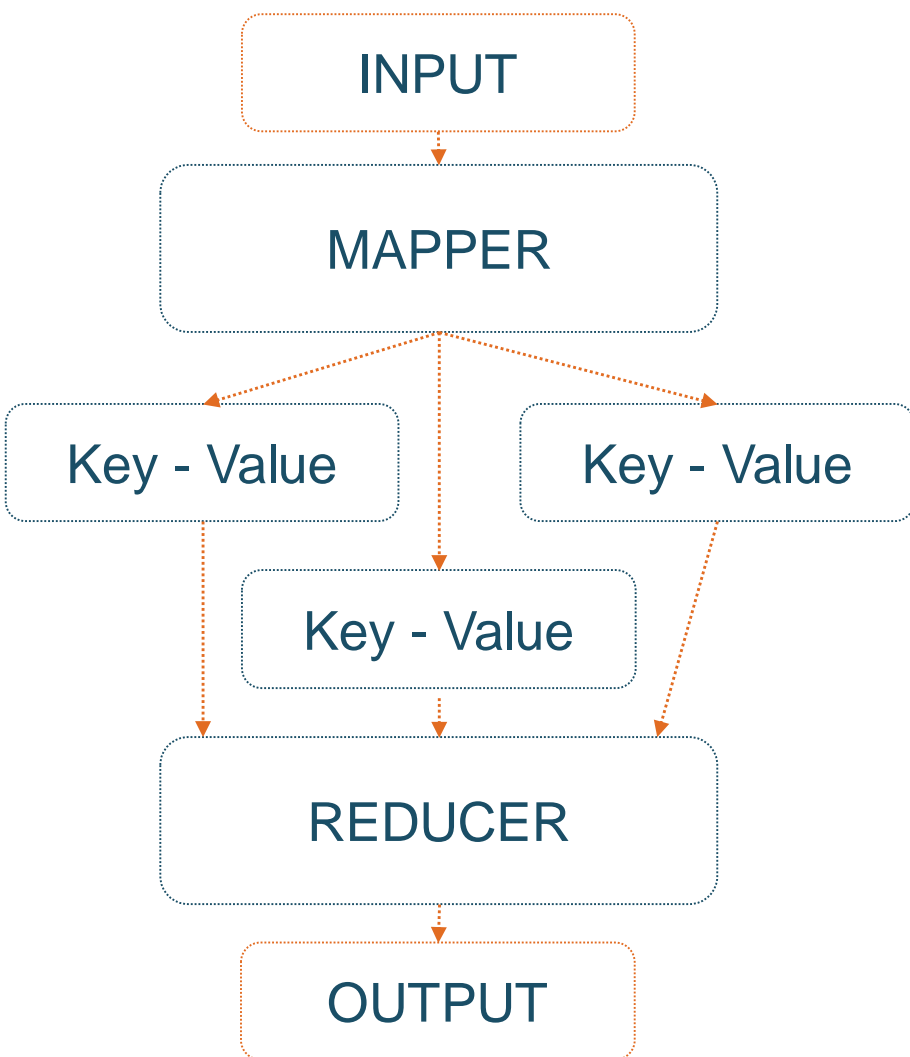
MAP – REDUCE



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

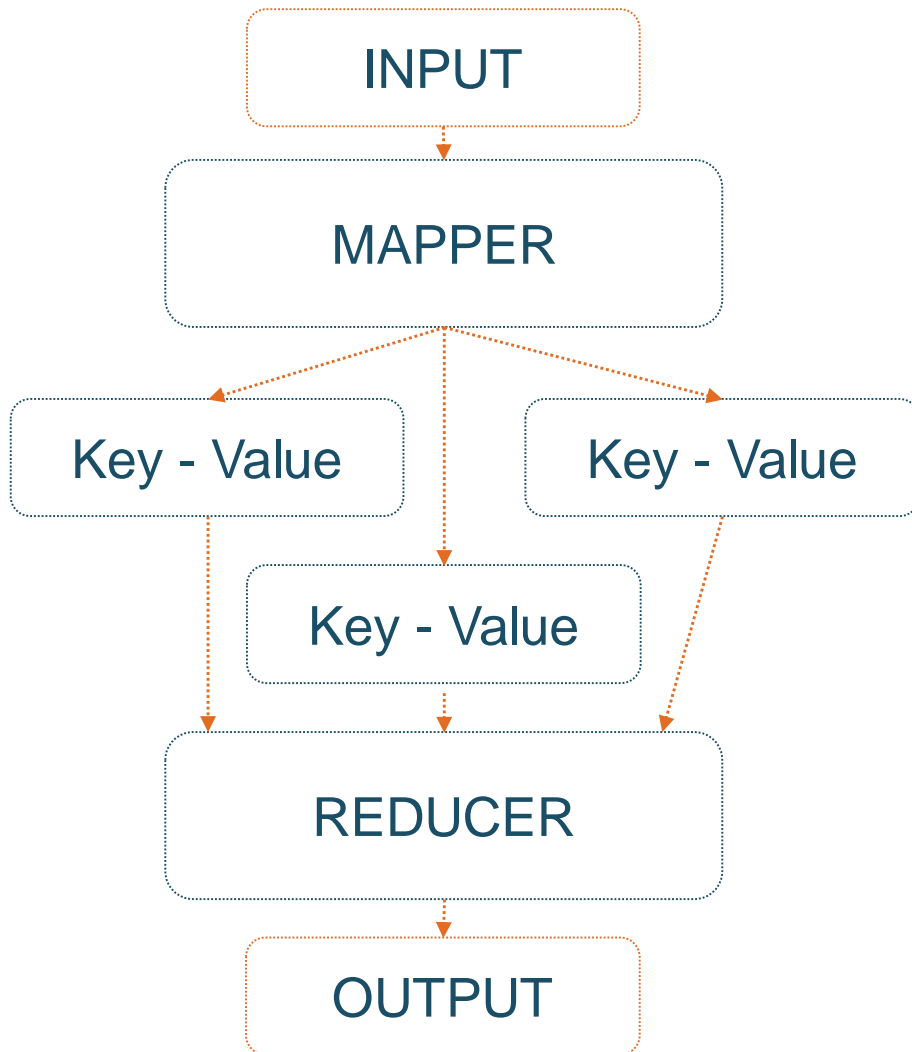


MAPPER

- 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

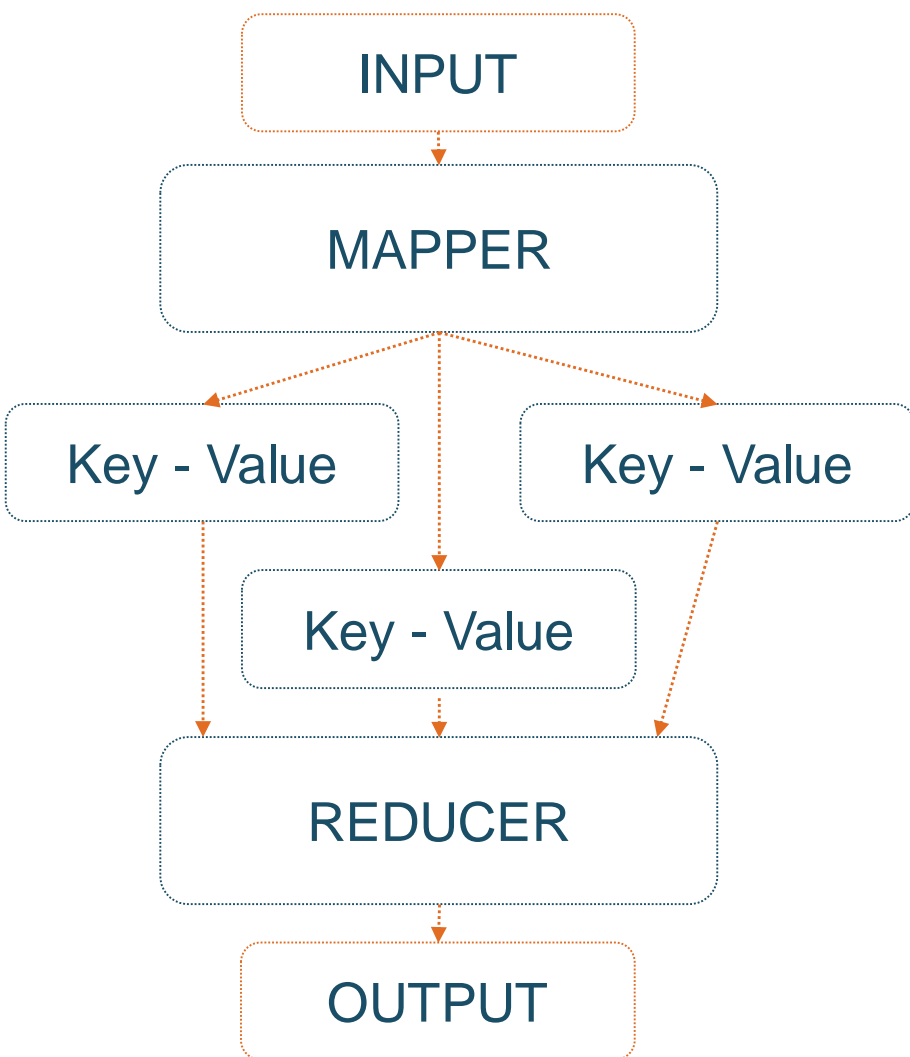
MAP – REDUCE



REDUCER

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

MAP – REDUCE



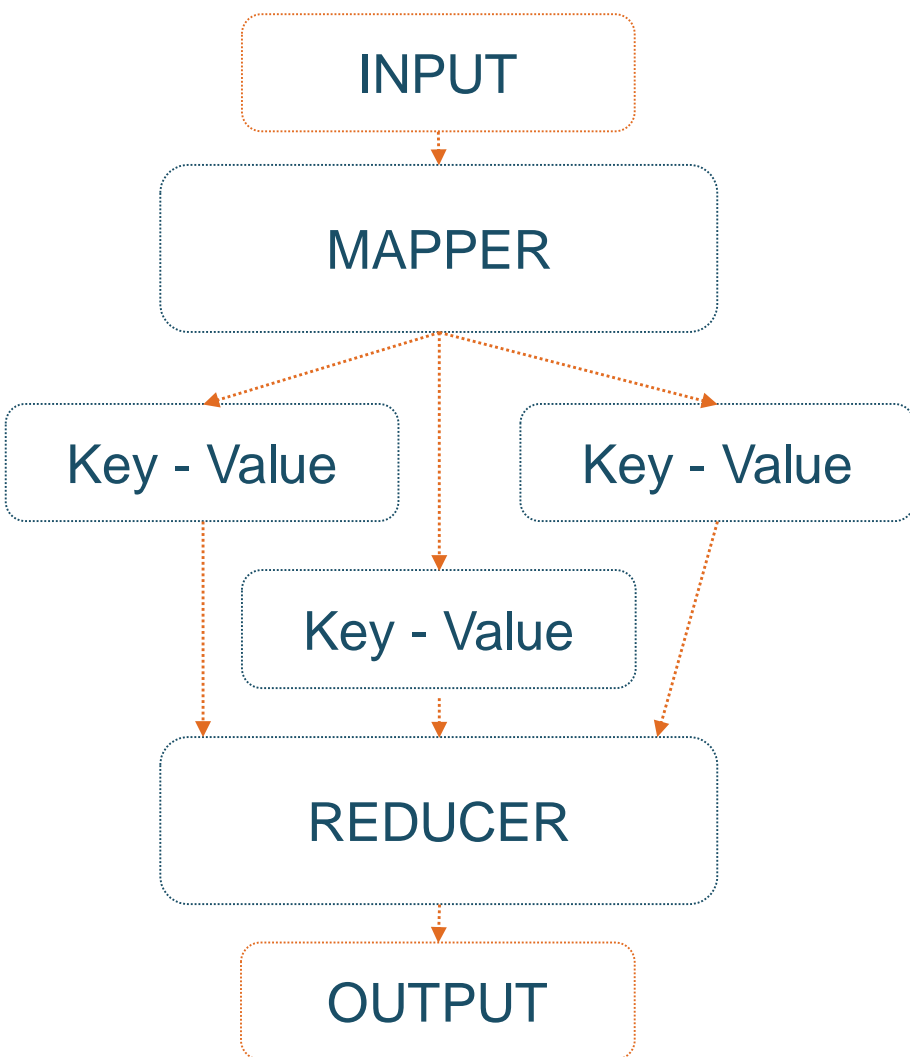
REDUCER

- 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



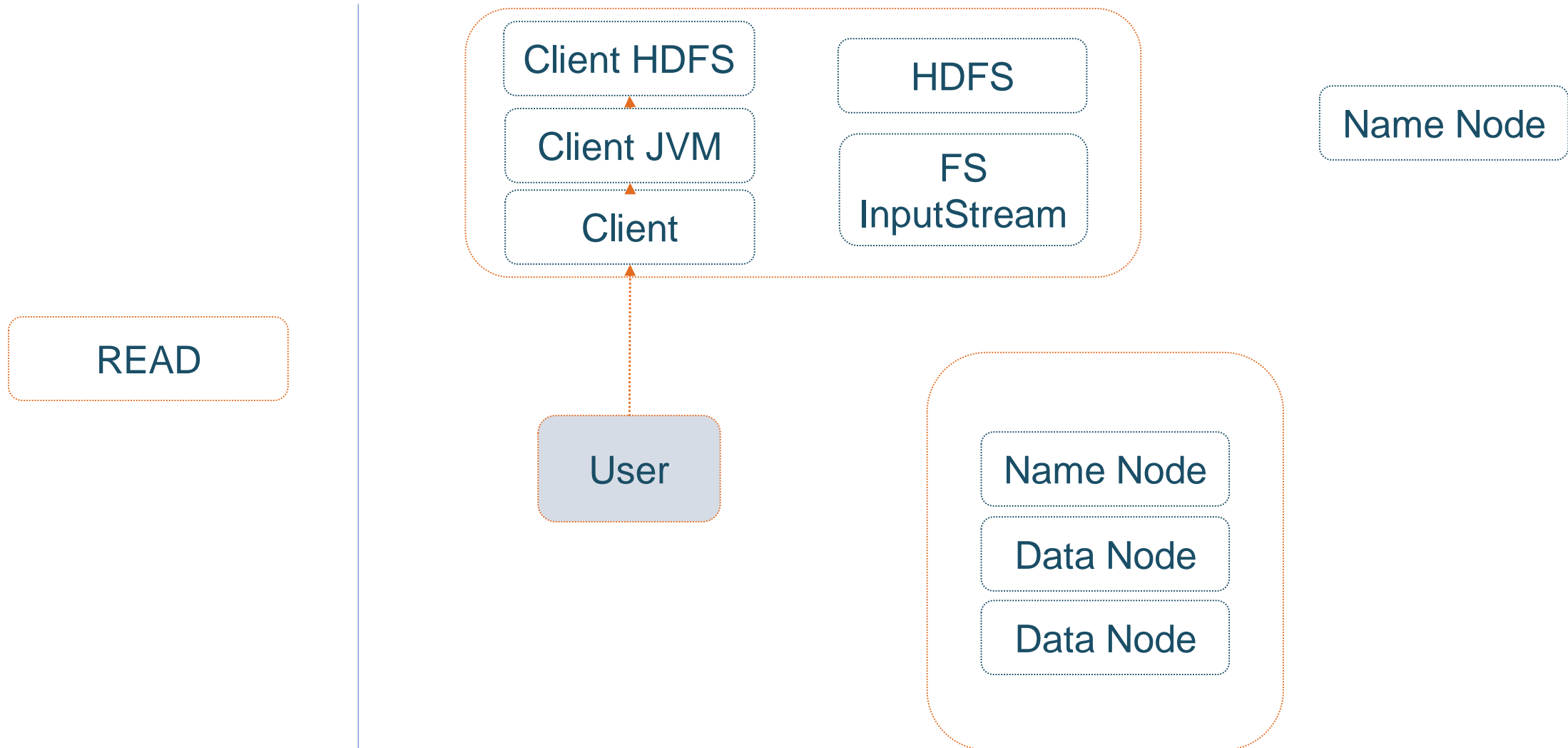
REDUCER

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

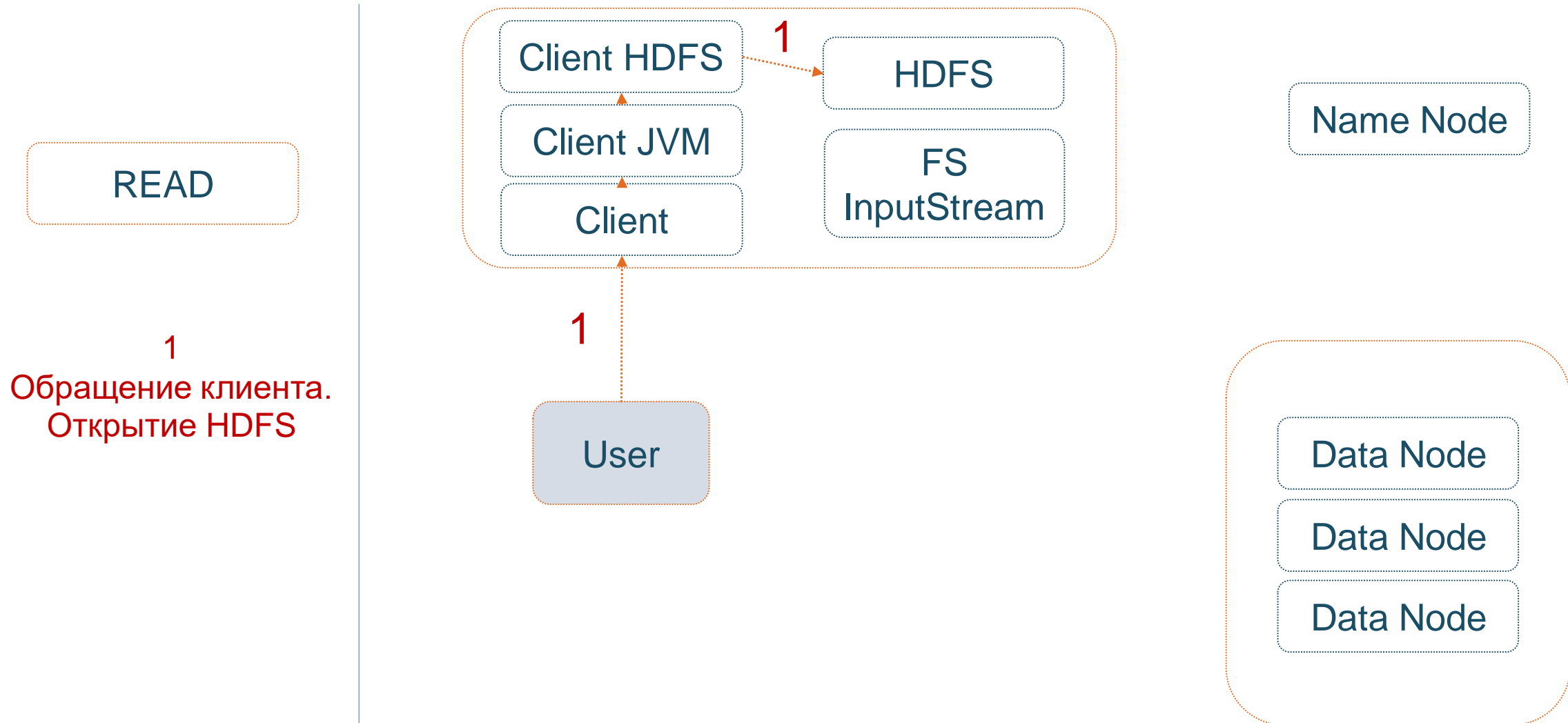
Пример: `const = 0.95` или `1.75` всегд, 3 ноды,
8 контейнеров на ноде

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

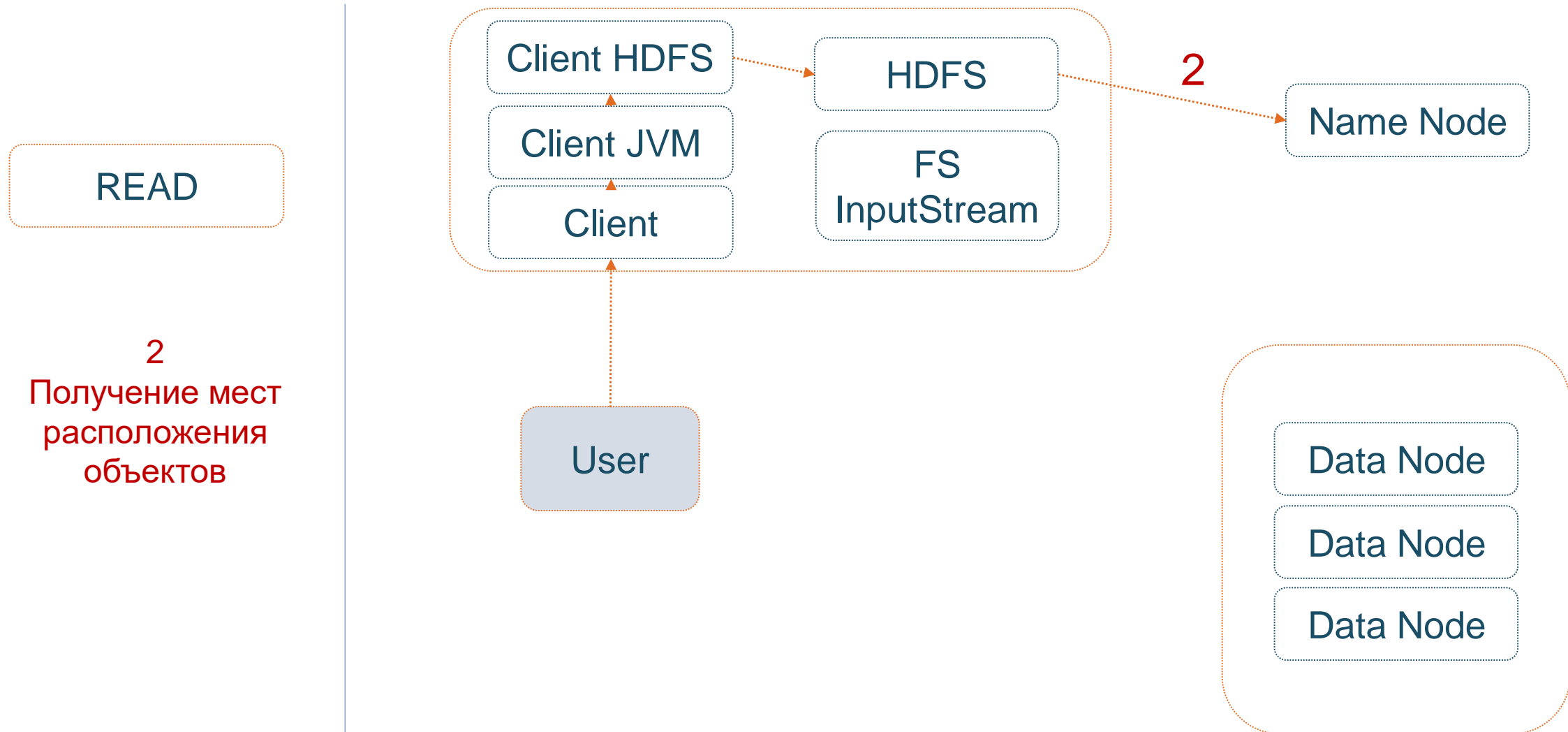
MAP – REDUCE | READ HDFS



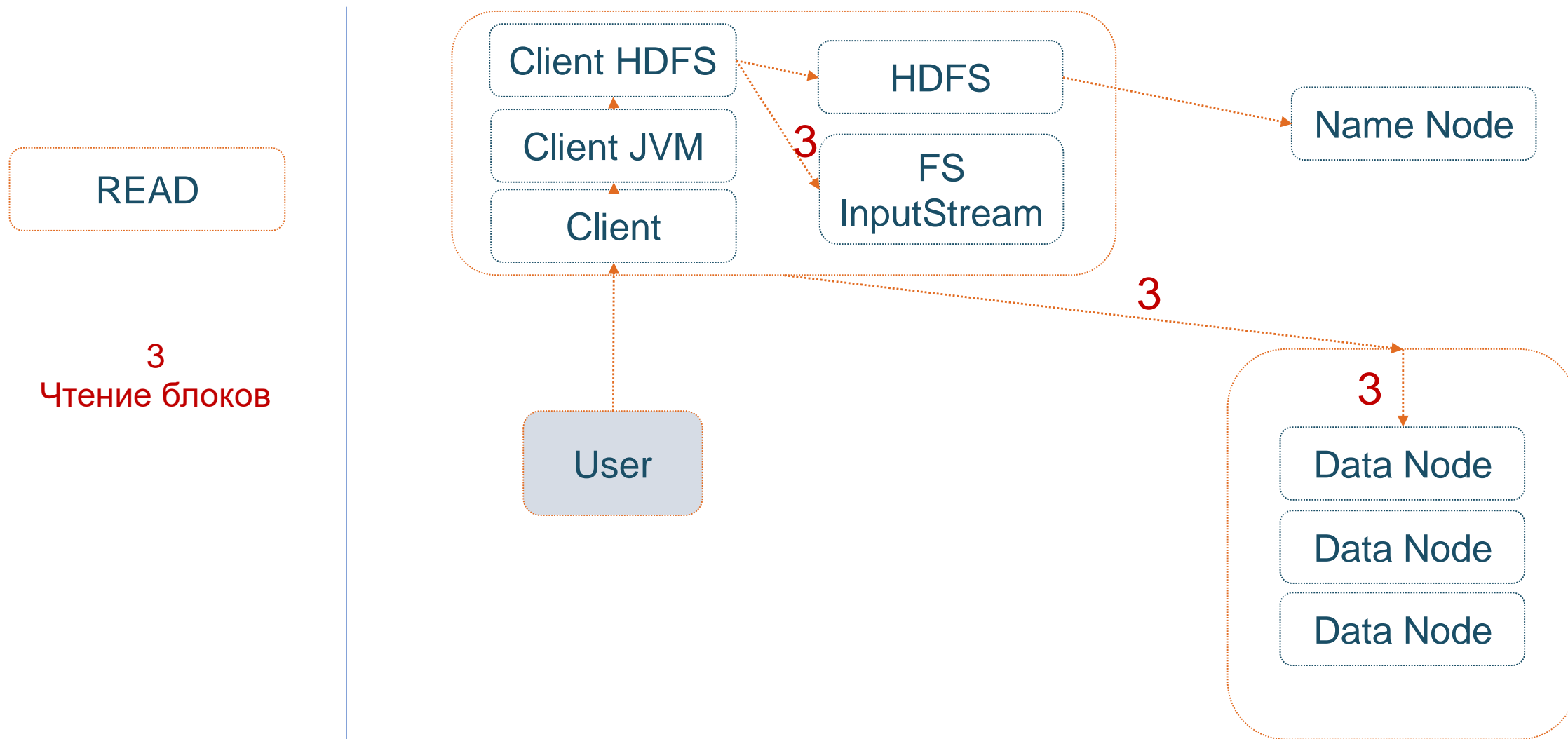
MAP – REDUCE | READ HDFS



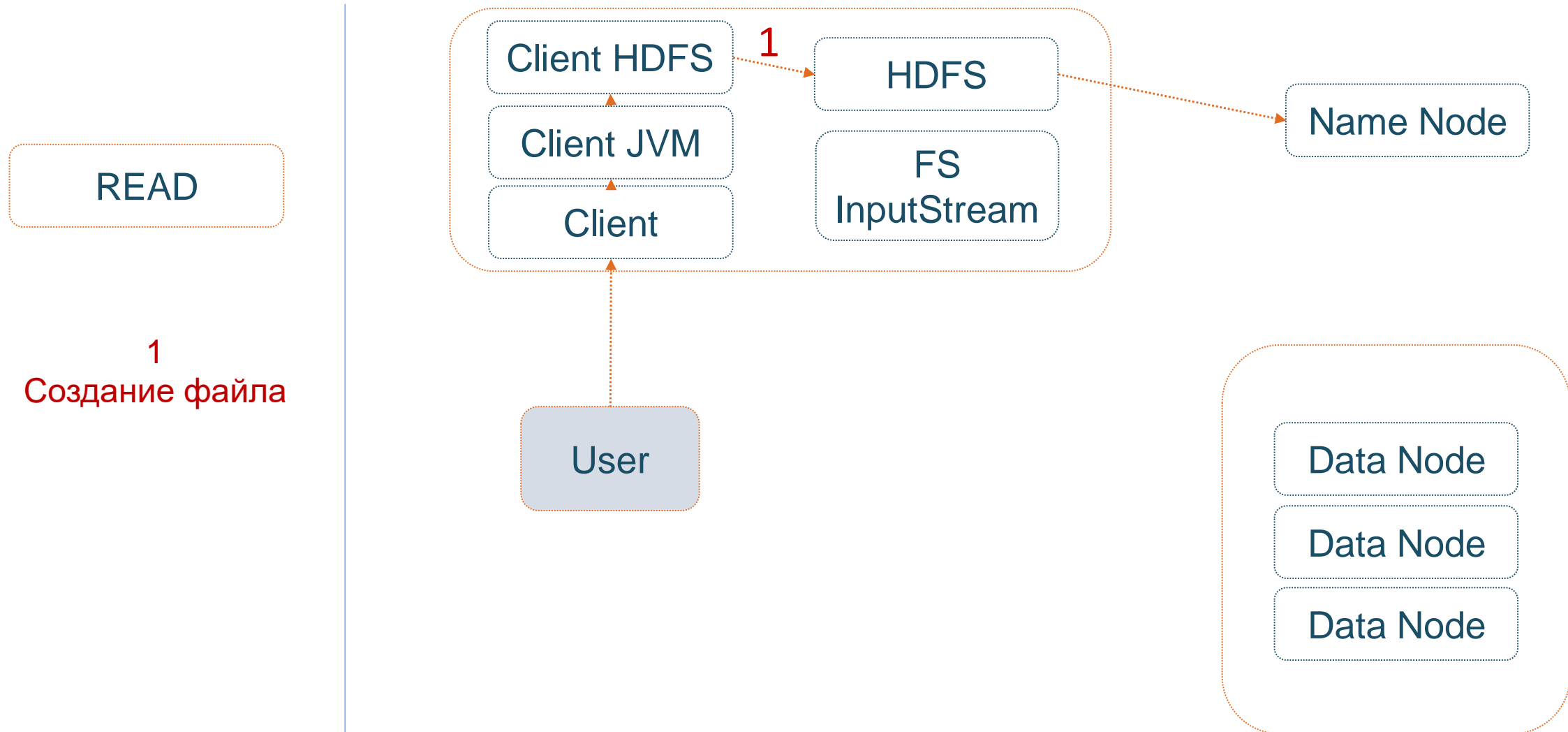
MAP – REDUCE | READ HDFS



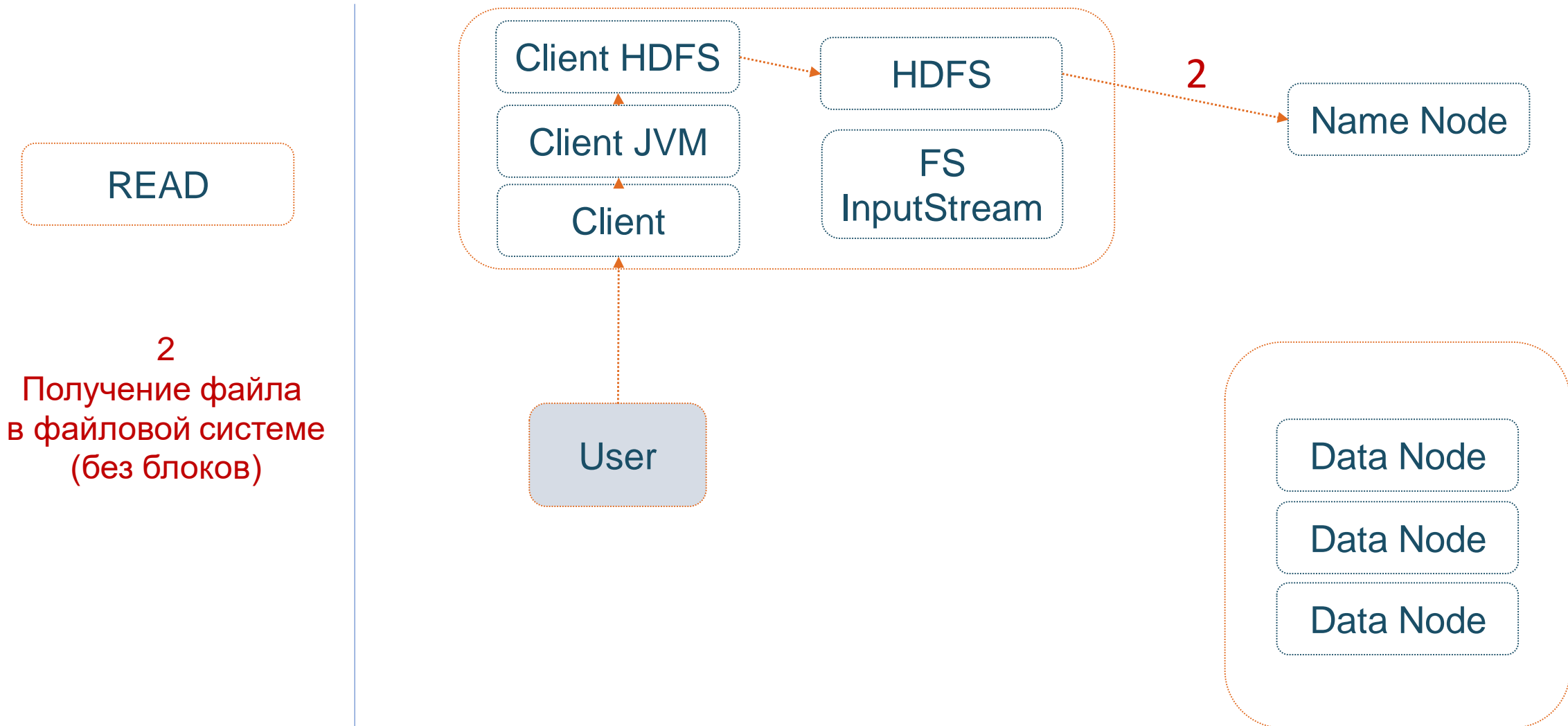
MAP – REDUCE | READ HDFS



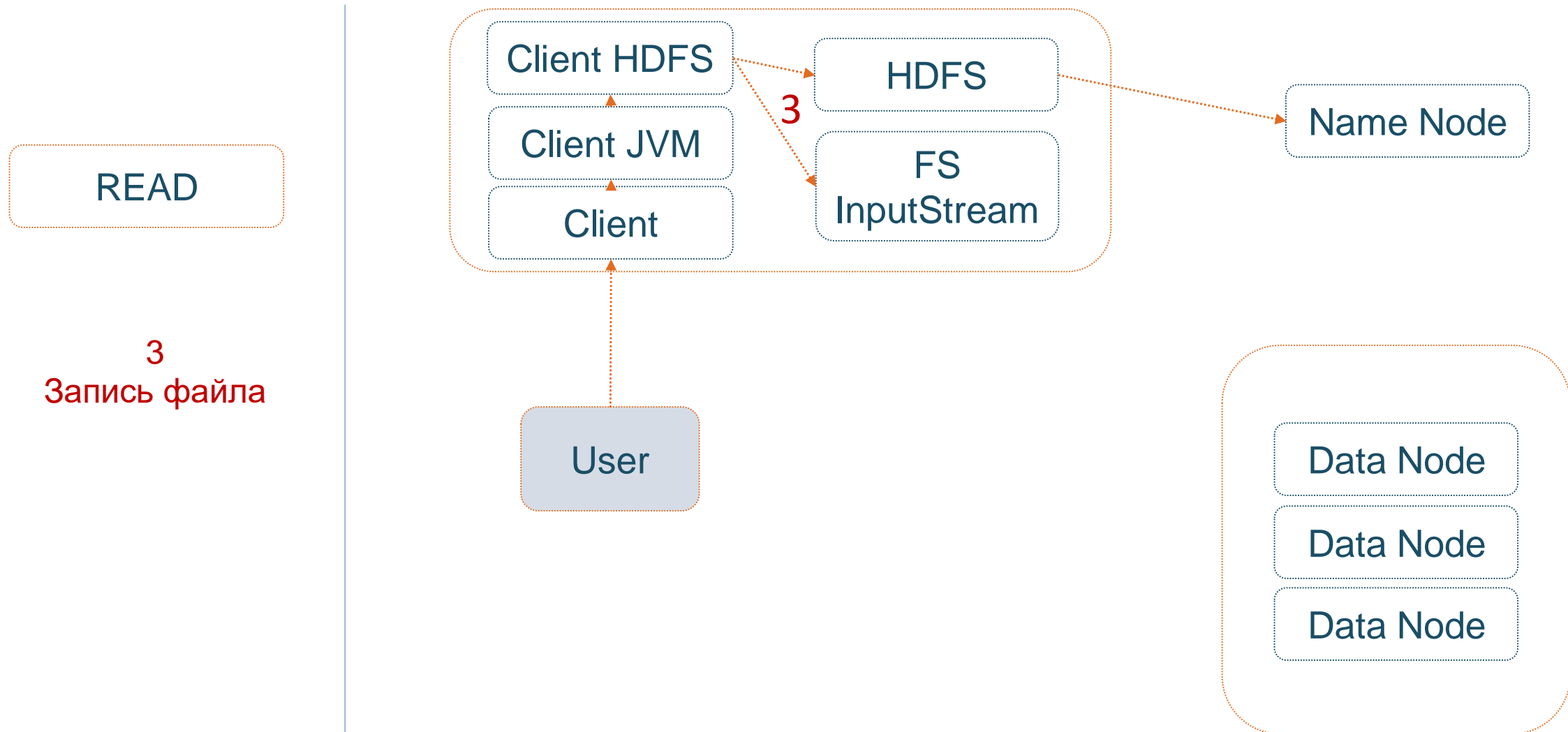
MAP – REDUCE | WRITE HDFS



MAP – REDUCE | WRITE HDFS



MAP – REDUCE | WRITE HDFS



MAP – REDUCE | WRITE HDFS

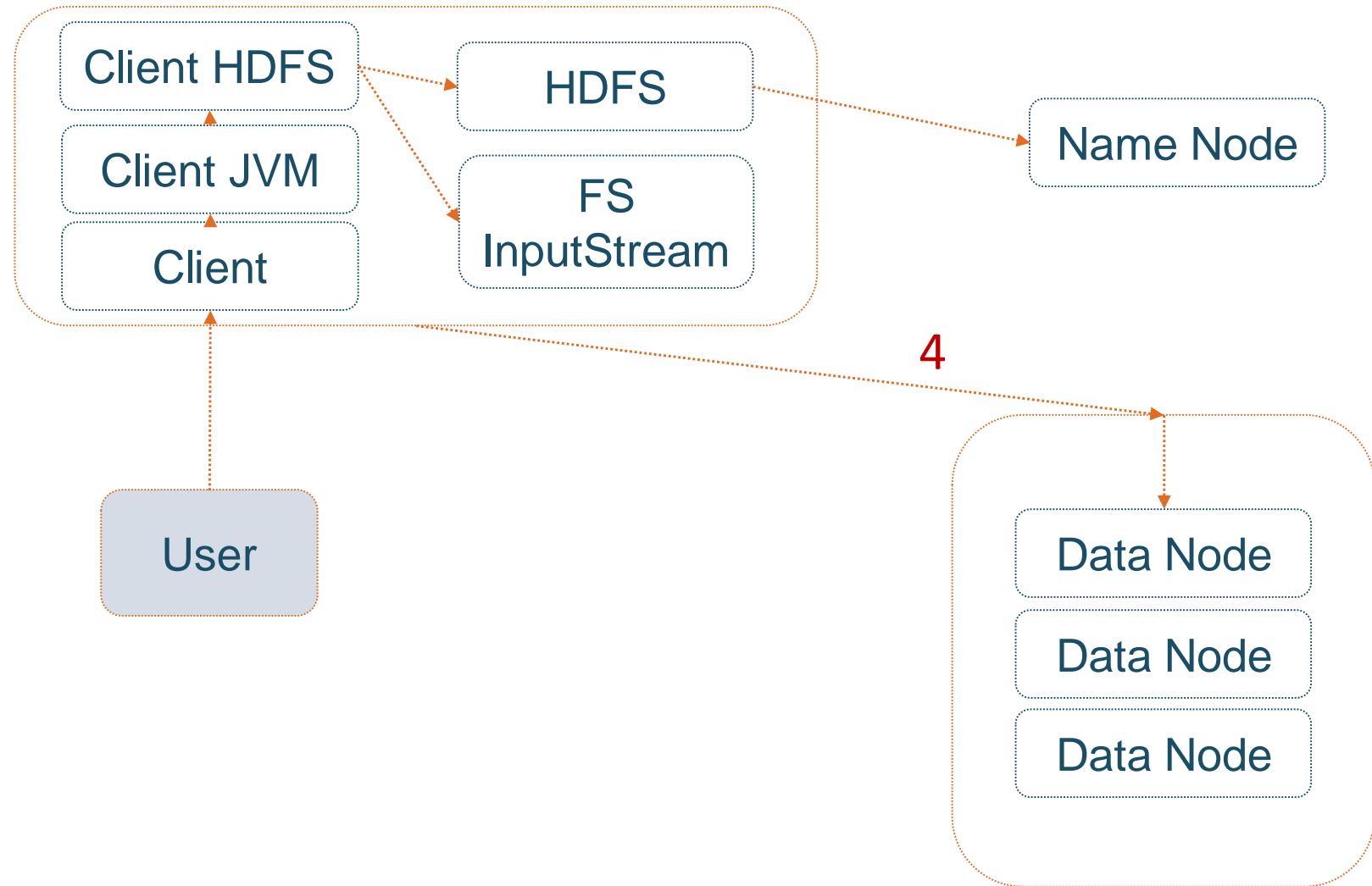
READ

4

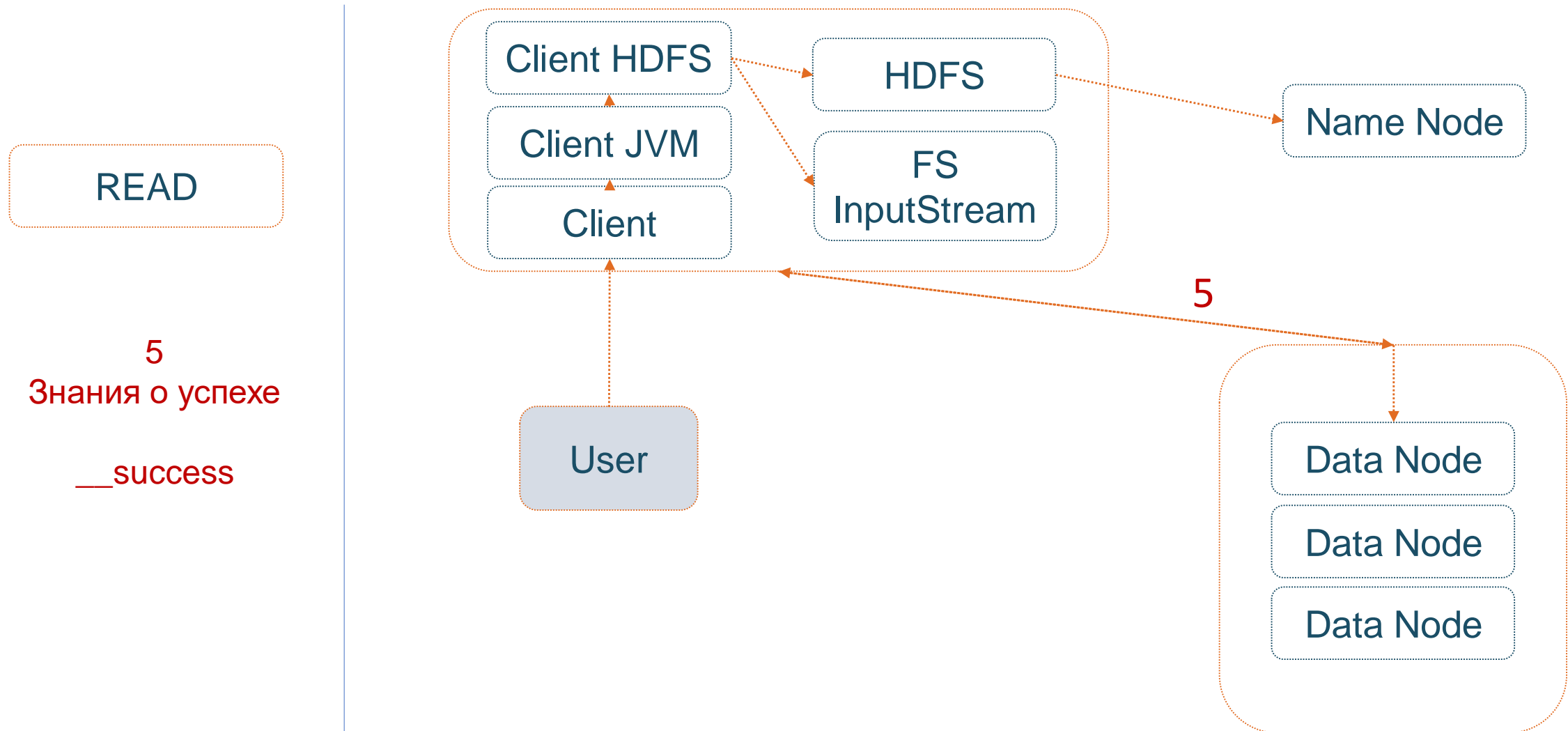
Запись файла:

- разделение на блоки
- разделение на ноды

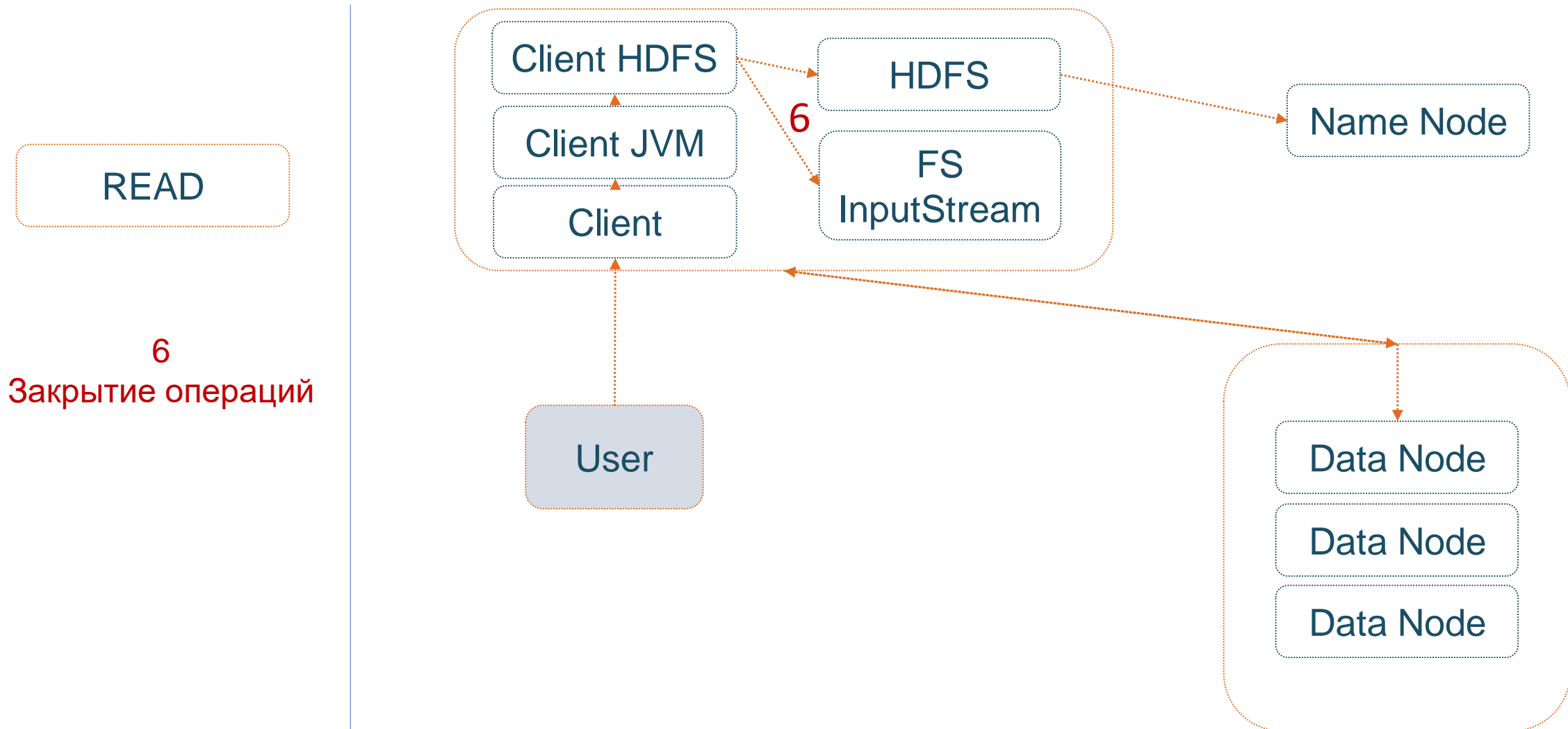
DataNode Pipeline



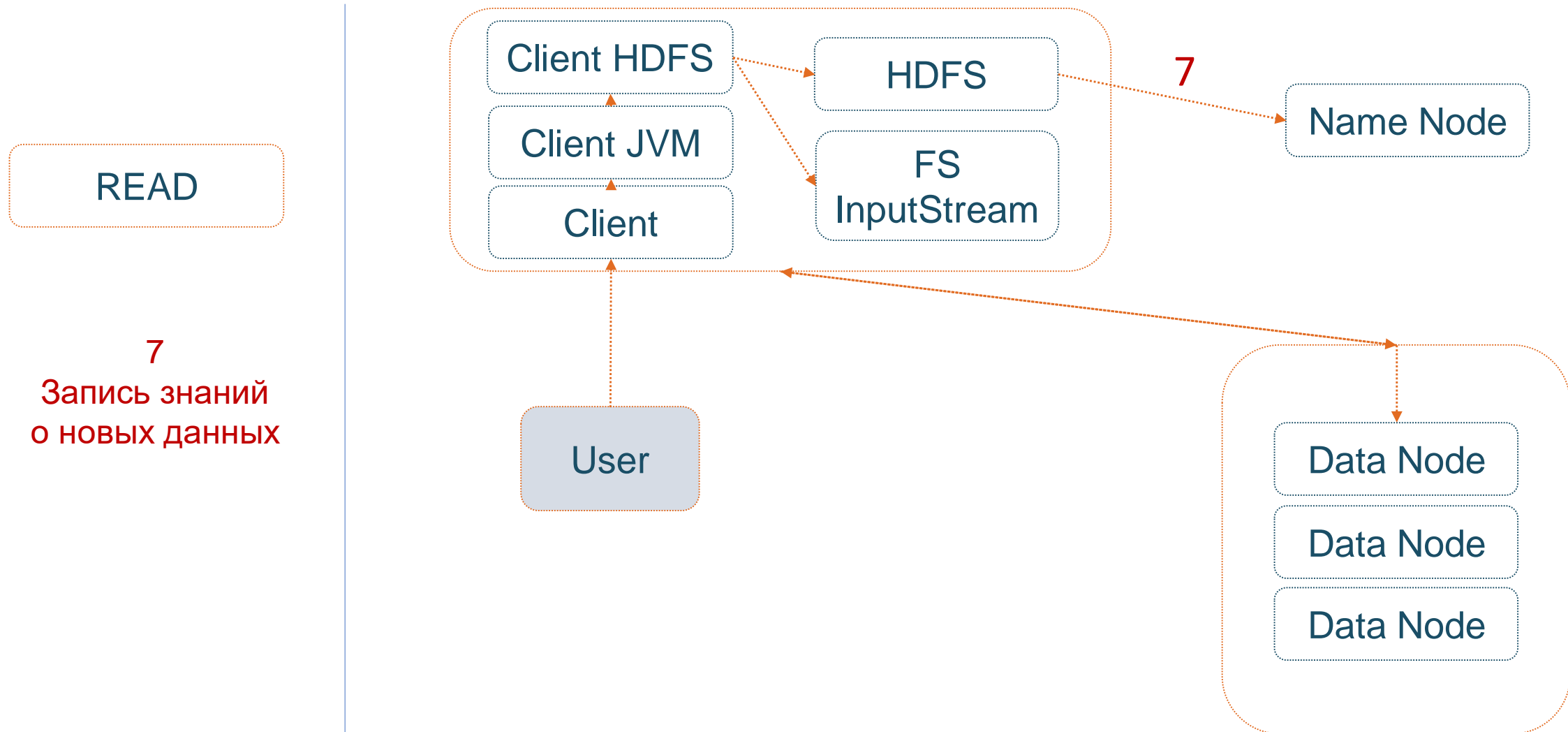
MAP – REDUCE | WRITE HDFS



MAP – REDUCE | WRITE HDFS

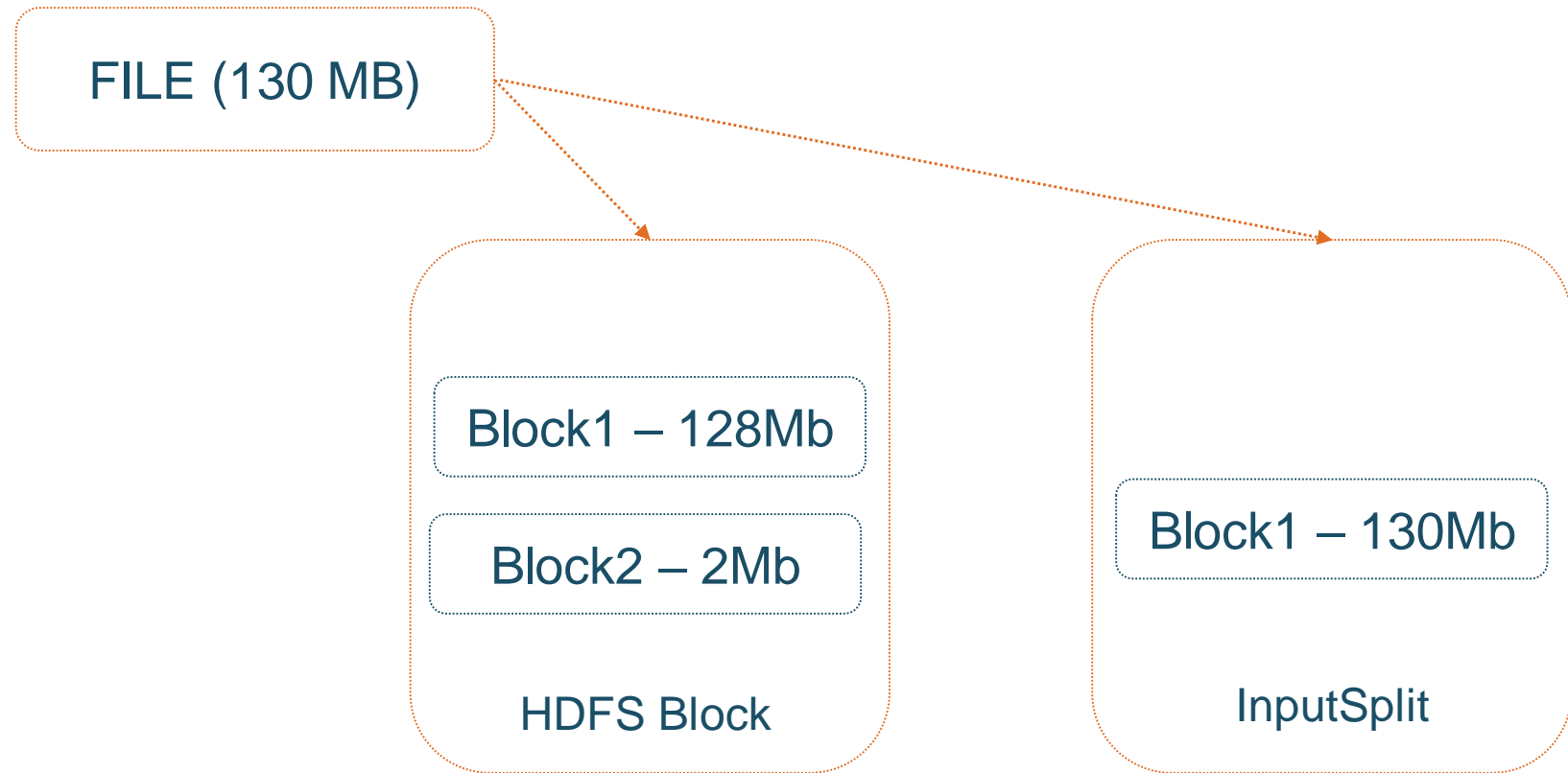


MAP – REDUCE | WRITE HDFS



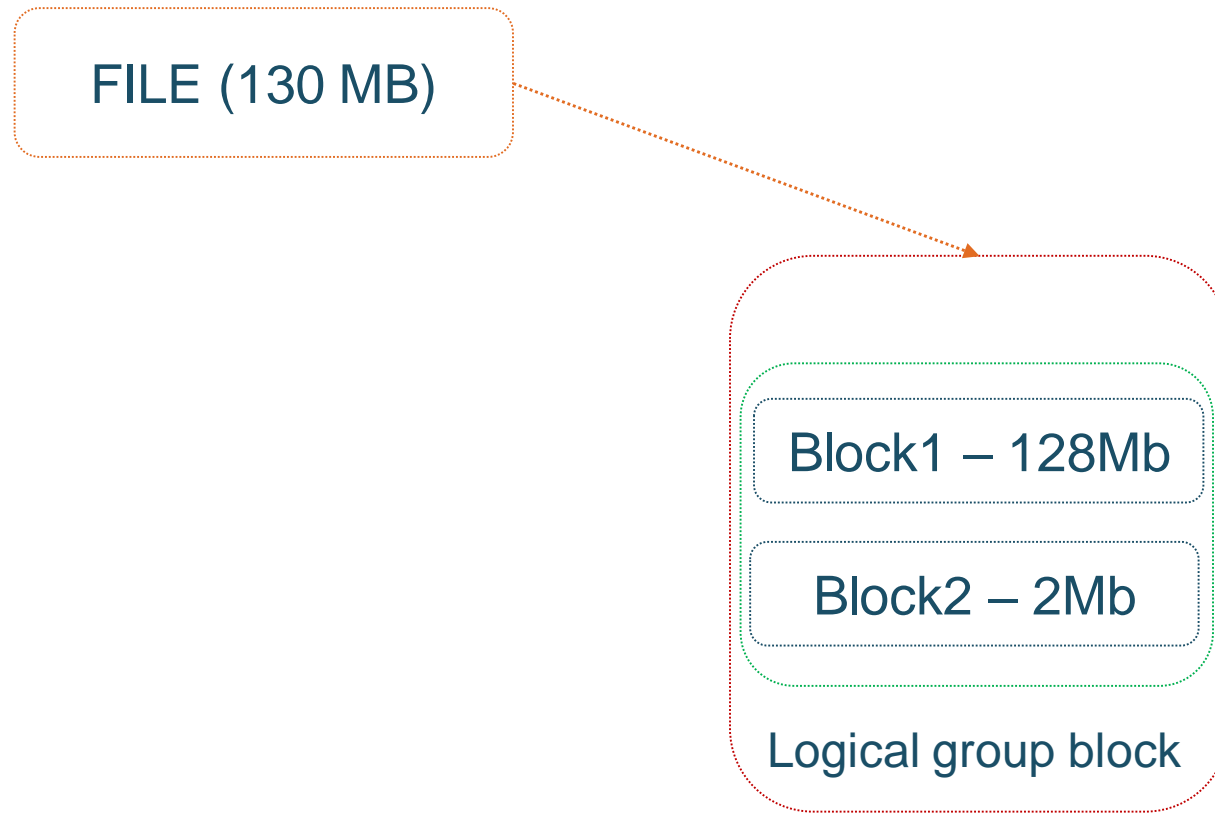
MAP – REDUCE | HDFS BLOCKS

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**

ЗАДАНИЕ ПО ПРОЕКТУ



TEST ON HADOOP TASK

- Используя библиотеку MRJOB:
 - напишите класс, который будет:
 - соединять данные их двух наборов
 - определять самый популярный почтовый домен
 - напишите 3 теста на данный класс
 - напишите Python функцию, которая будет запускать код класса, если все 3 теста прошли успешно