Big Data

Igor Garcia Ballhausen Sampaio

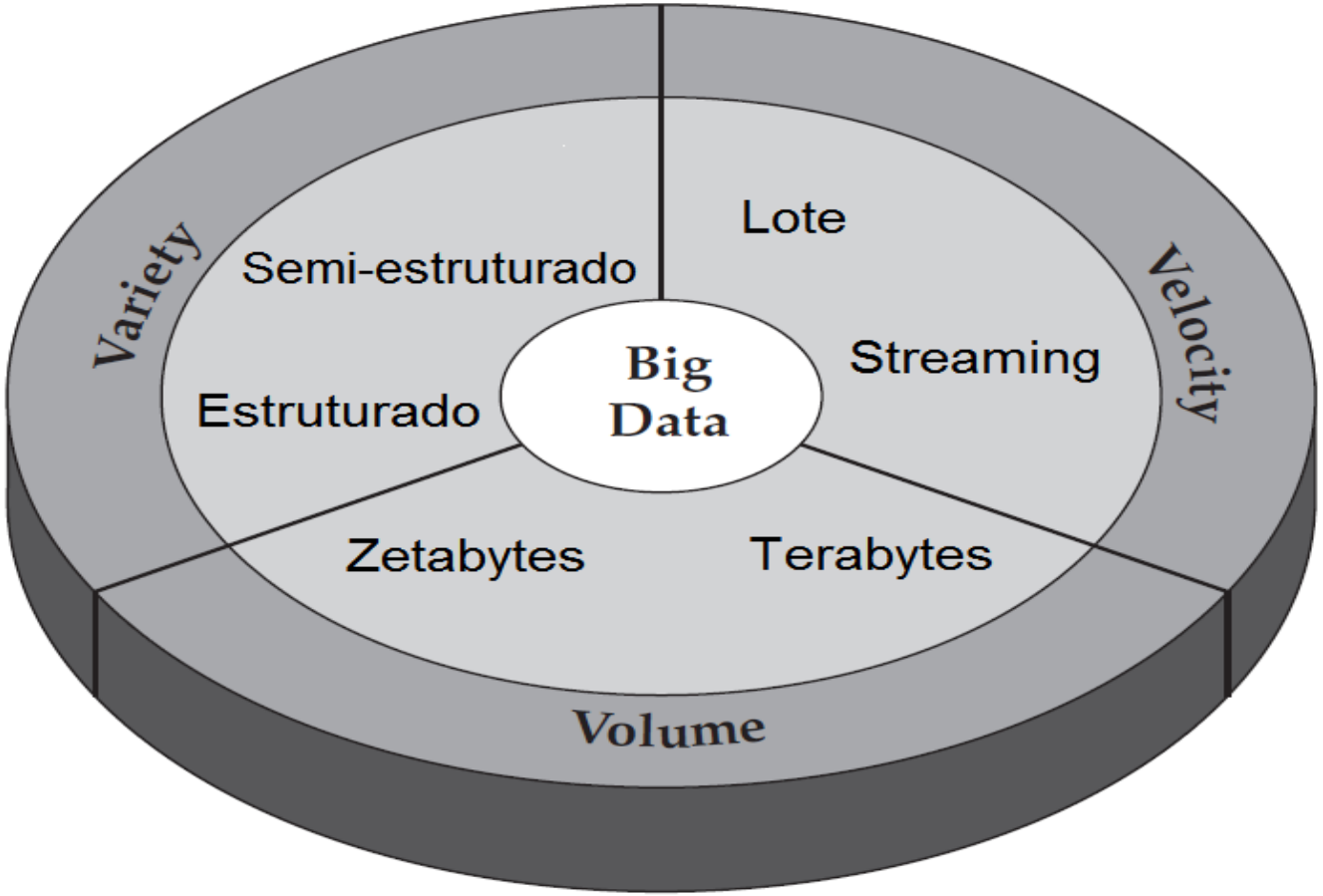
Agenda

- Big Data (Vs)
- Computação Distribuída
- O que é Hadoop (HDFS, MapReduce)

Proposta de Trabalho

- Dado um problema
- Propor uma nova arquitetura
- Implementar essa arquitetura
  - Modelar o BDD novo
  - Projetar um particionamento adequado/efetivo
  - Fazer bateria de testes no ambiente distribuído e centralizado
  - https://dados.gov.br/dados/conjuntos-dados/cadastro-ambiental-rural1

Volume, variedade e velocidade



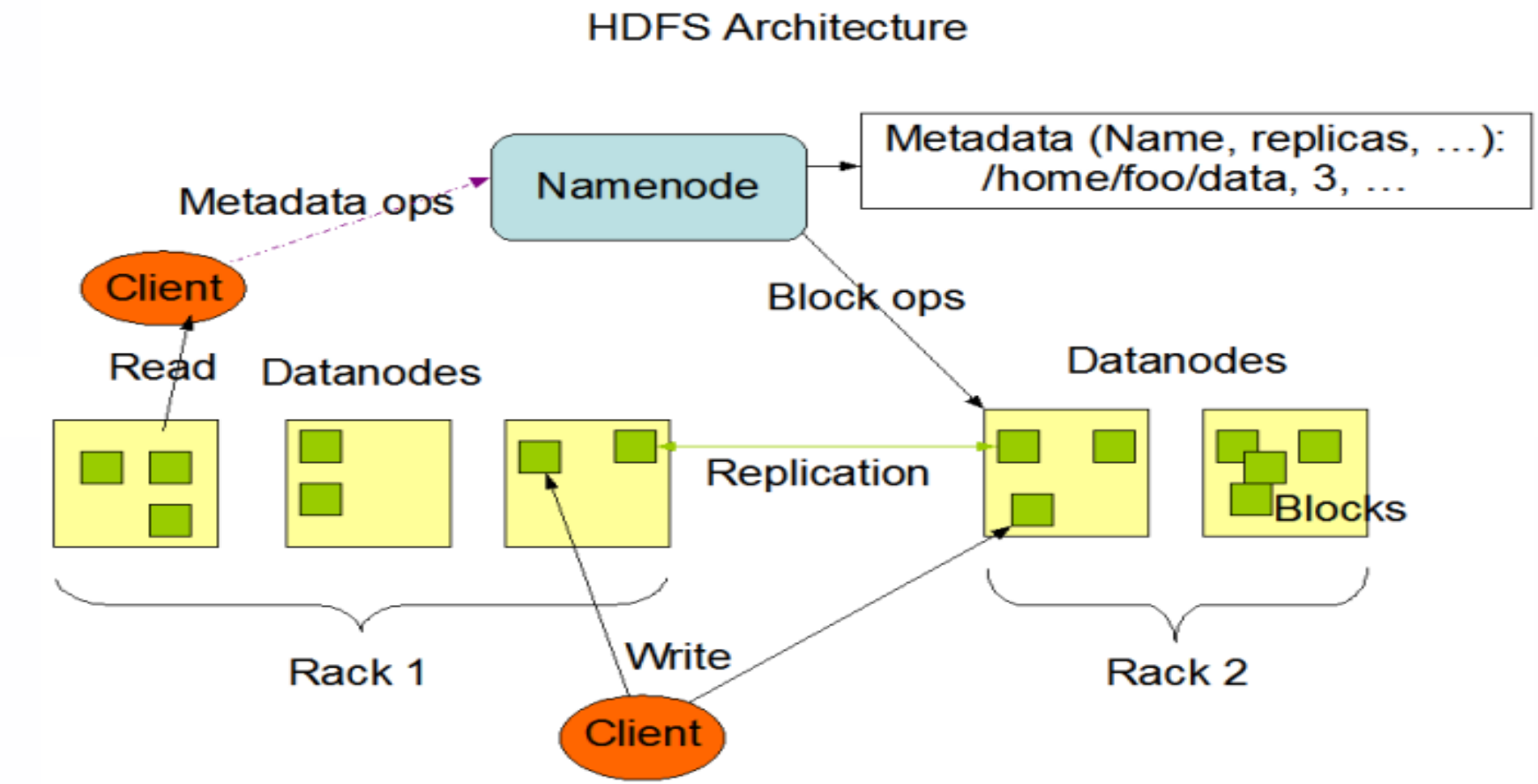
Ecossistema Hadoop

- O Apache Hadoop é um framework open source para o armazenamento e processamento de dados em larga escala
- Algoritmo de MapReduce
- Hadoop Distributed File System (HDFS), para armazenamento de grandes conjuntos de dados, também de forma distribuída.

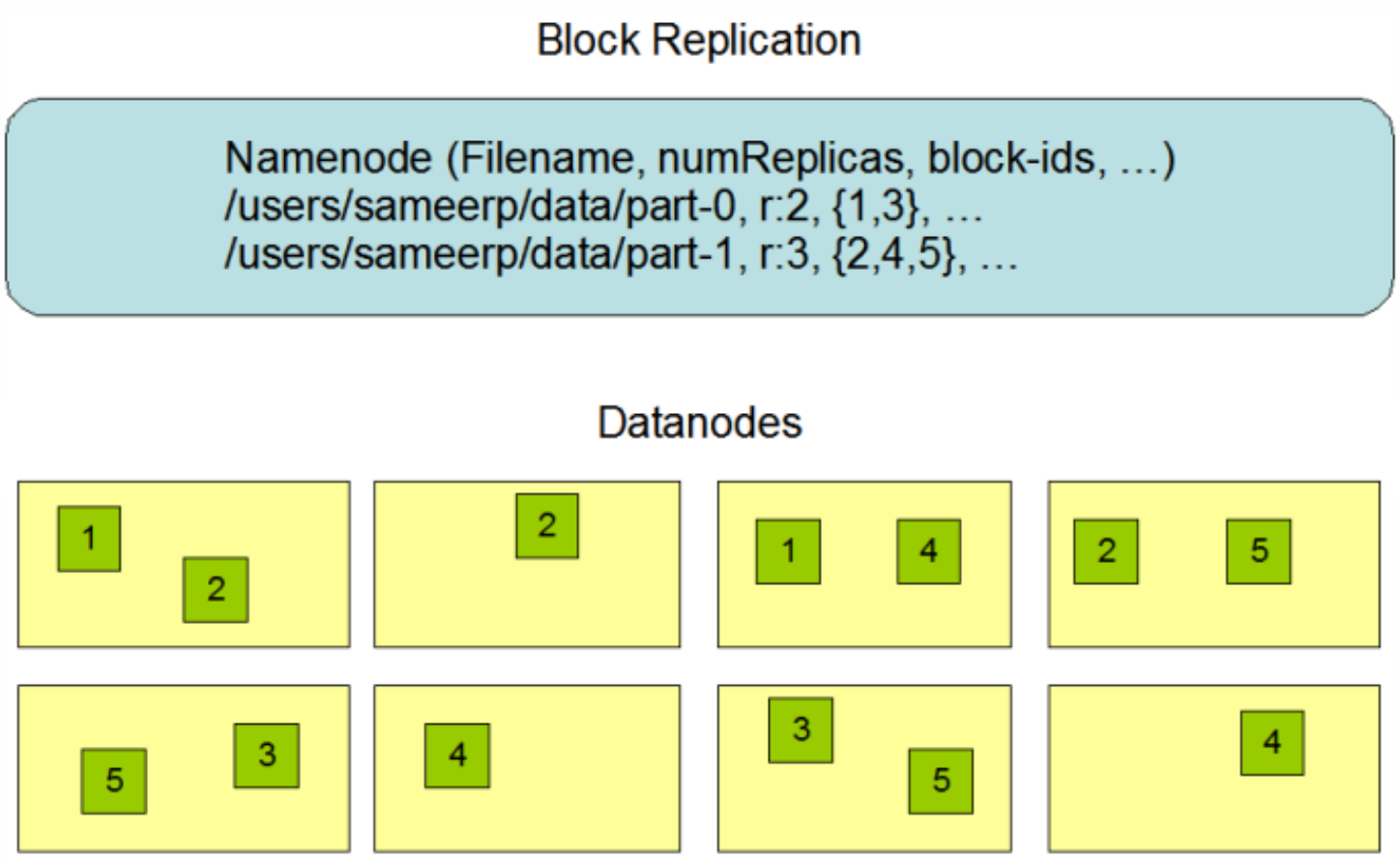
Ecossistema Hadoop

- **Modo Local ou Independente:** Por padrão, o Hadoop foi configurado para executar em modo independente não distribuído. Esse modo é útil para desenvolver e testar um aplicativo;
- **Modo Pseudo distribuído:** Pode executar em um único nó em modo pseudo distribuído. Nesse caso, cada instância de processo Hadoop executa como um processo Java diferente;
- **Modo Totalmente distribuído:** O Hadoop é configurado em cluster com máquinas físicas (ou virtualizadas), cada qual com um endereço IP válido.

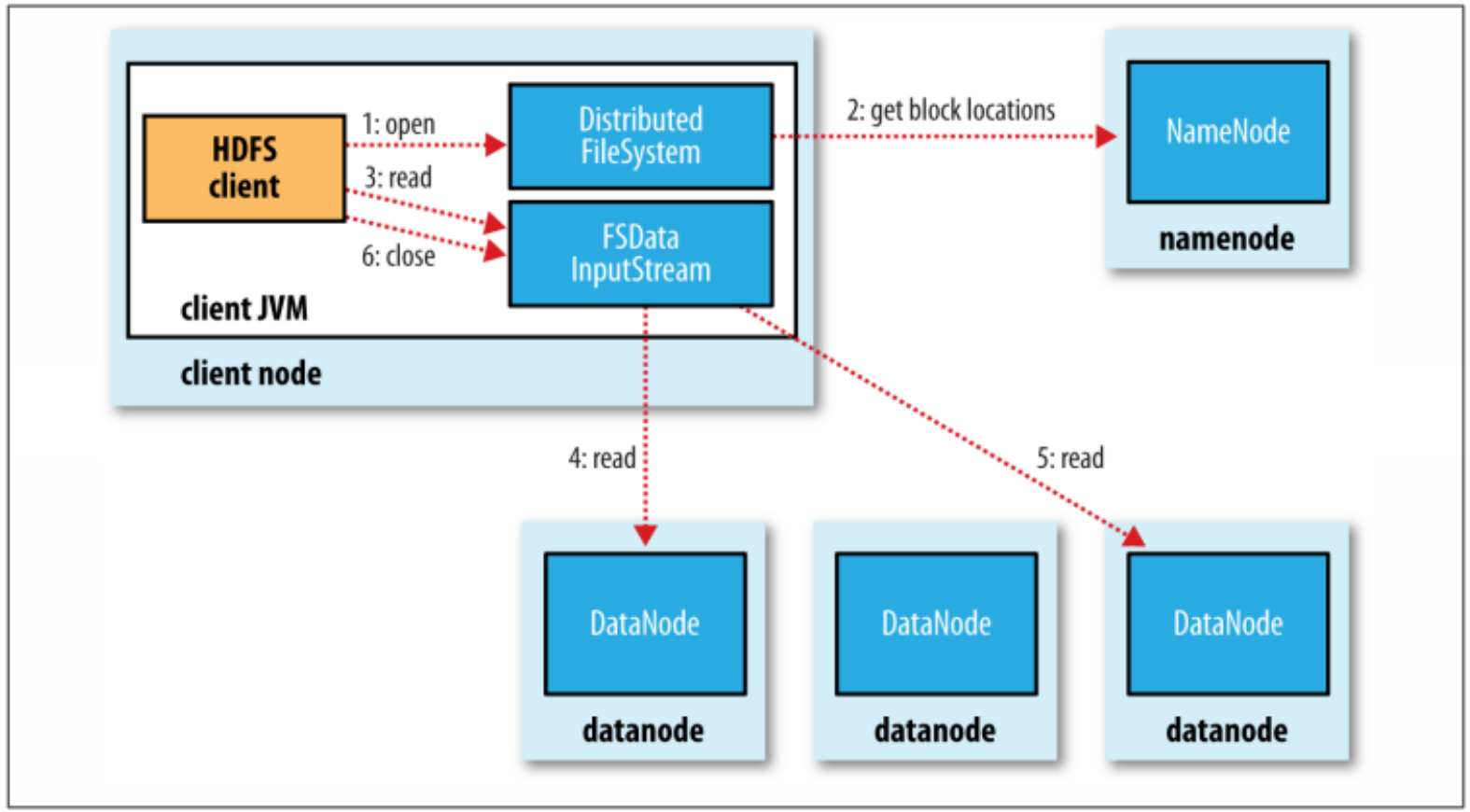
Arquitetura



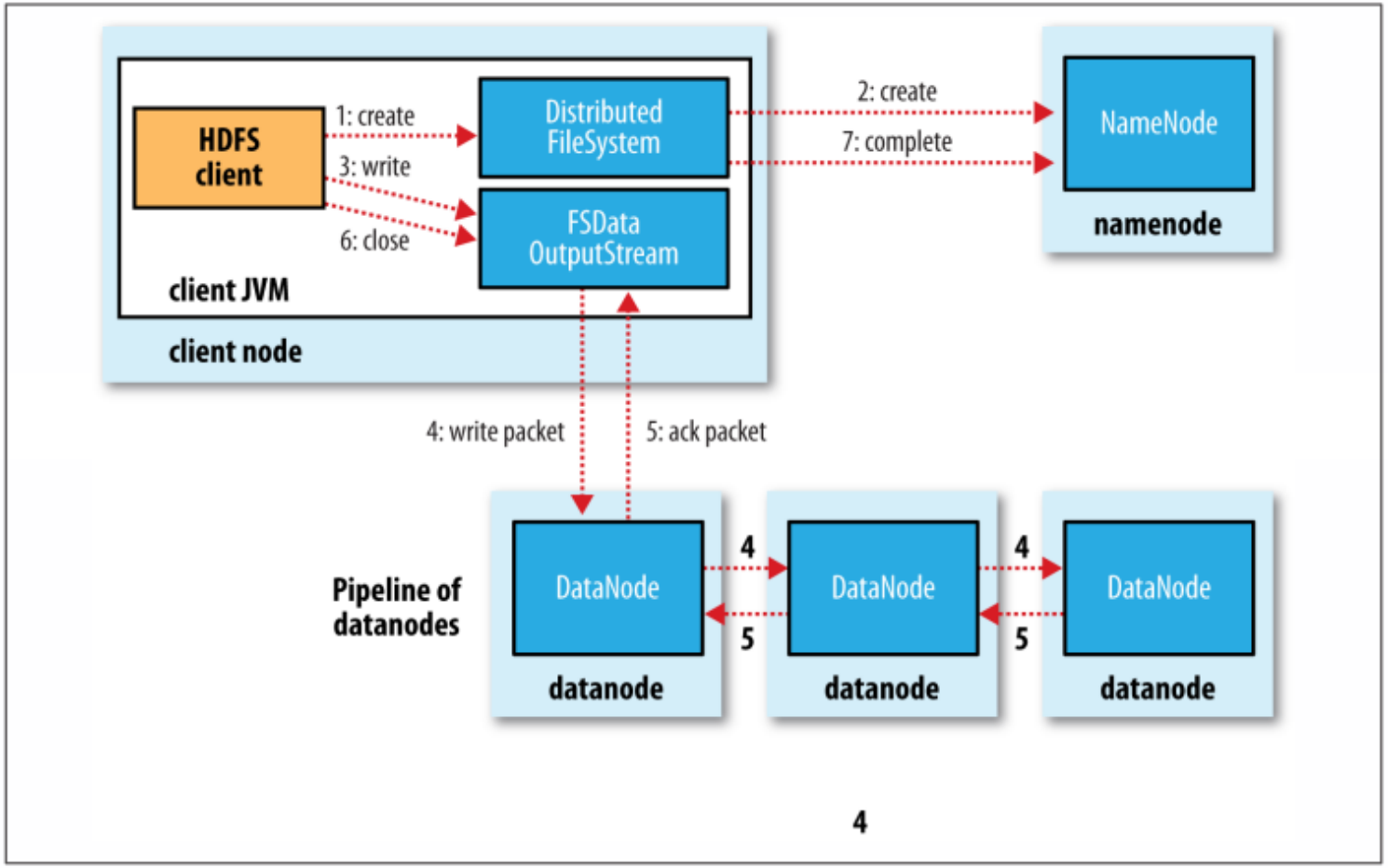
Block Replication



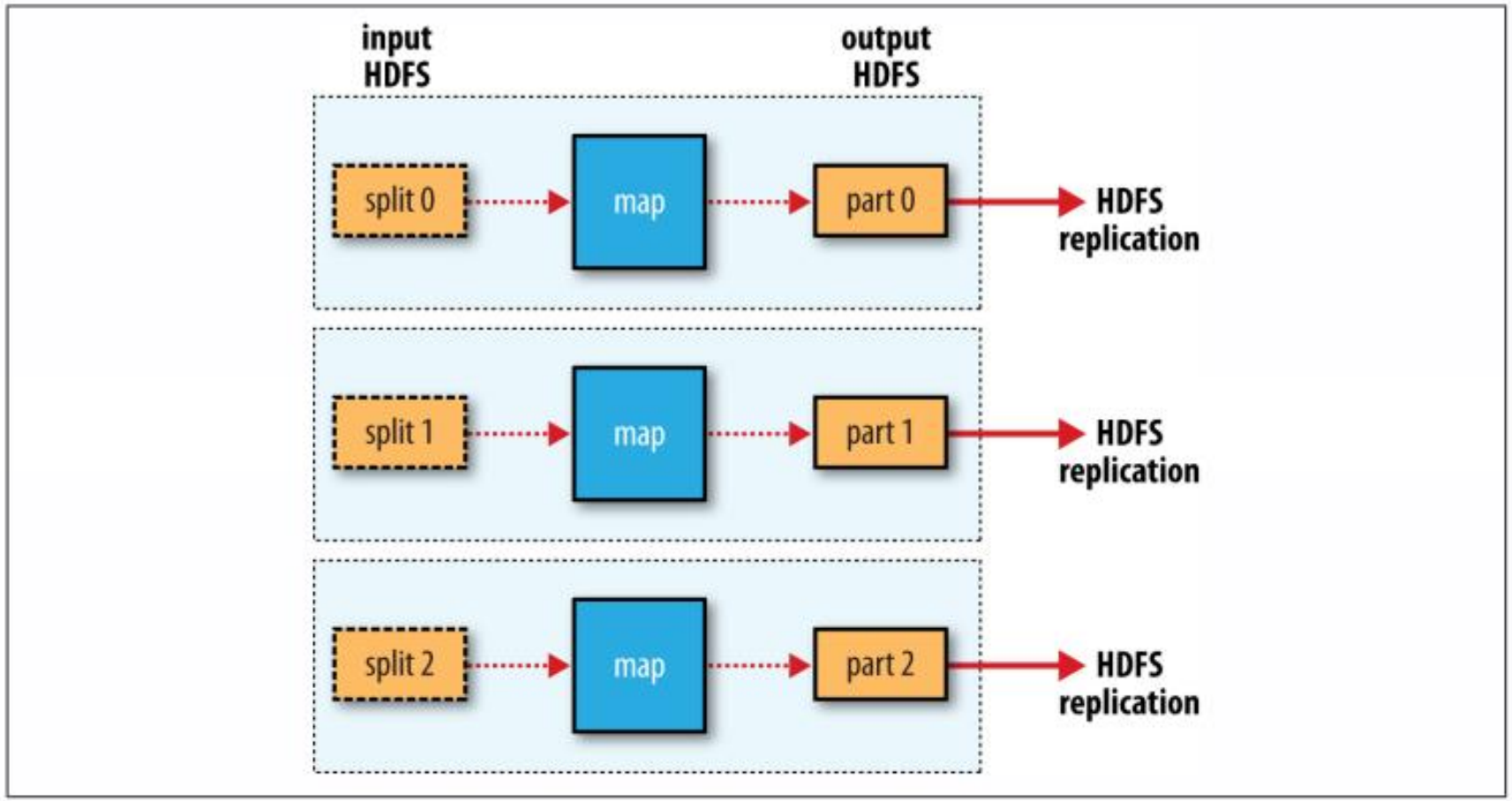
Leitura


HDFS client
1: open
3: read
6: close
Distributed FileSystem
FSData InputStream
client JVM
client node
2: get block locations
NameNode
namenode
4: read
5: read
DataNode
datanode
DataNode
datanode
DataNode
datanode

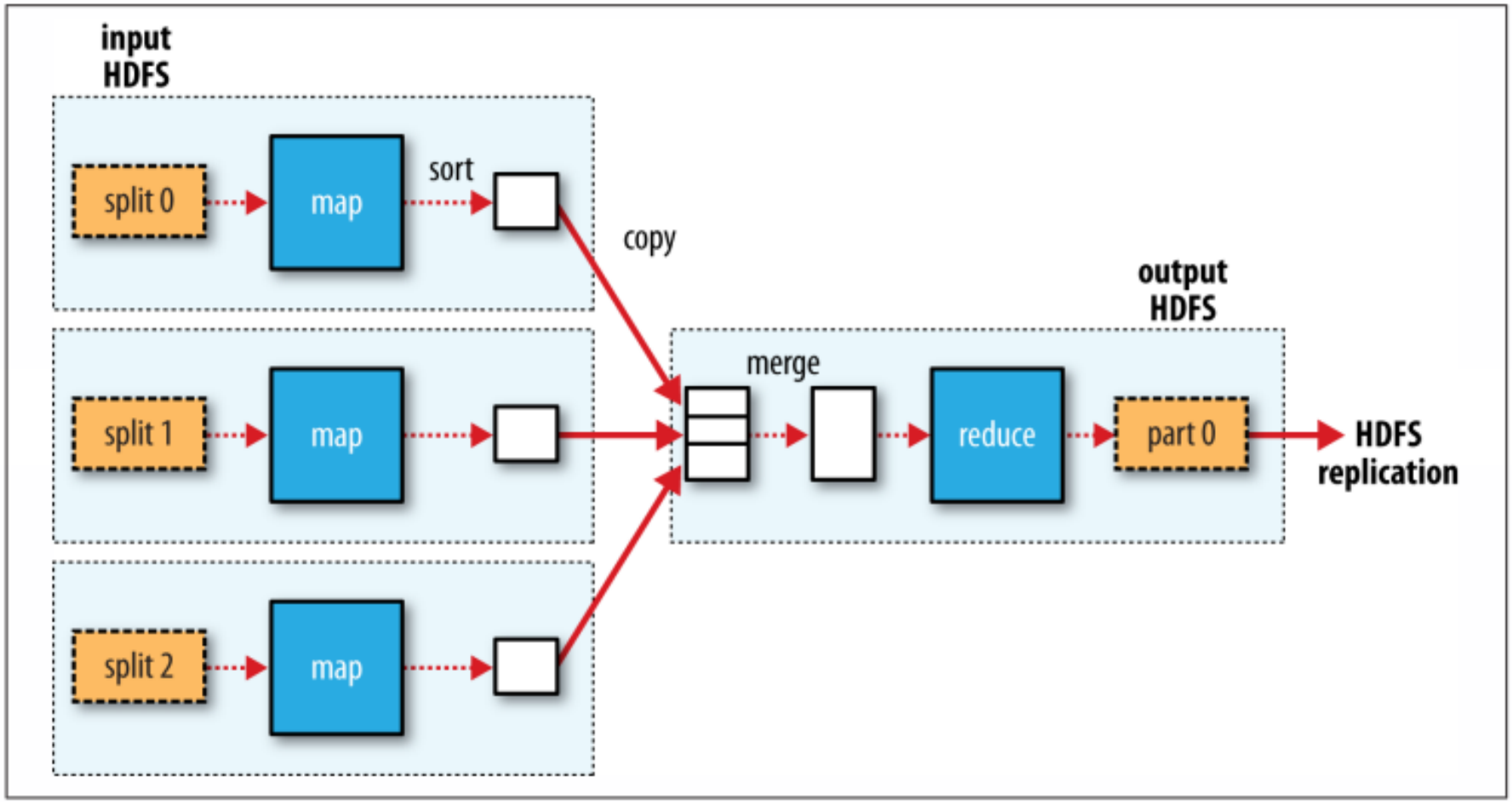
Escrita


HDFS client
1: create
3: write
6: close
Distributed FileSystem
FSData OutputStream
client JVM
client node
2: create
7: complete
NameNode
namenode
4: write packet
5: ack packet
Pipeline of datanodes
DataNode
DataNode
DataNode
4
5
4
5
datanode
datanode
datanode
4

Map Reduce


input
HDFS
output
HDFS
split 0
map
part 0
HDFS
replication
split 1
map
part 1
HDFS
replication
split 2
map
part 2
HDFS
replication

Map Reduce


input
HDFS
split 0
map
sort
split 1
map
split 2
map
copy
output
HDFS
merge
reduce
part 0
HDFS
replication

Algoritmo

“O MapReduce é um algoritmo para processamento paralelo das aplicações. Ele abstrai as dificuldades do trabalho com dados distribuídos, eliminando quaisquer problemas que o compartilhamento de informações pode trazer em um sistema dessa natureza.”

Algoritmo

- **Map:** Responsável por receber os dados de entrada, estruturados em uma coleção de pares chave/valor. Tal função map deve ser codificada pelo desenvolvedor, através de programas escritos em Java ou em linguagens suportadas pelo Hadoop;
- **Shuffle:** A etapa de shuffle é responsável por organizar o retorno da função Map, atribuindo para a entrada de cada Reduce todos os valores associados a uma mesma chave. Esta etapa é realizada pela biblioteca do MapReduce;
- **Reduce:** Por fim, ao receber os dados de entrada, a função Reduce retorna uma lista de chave/valor contendo zero ou mais registros, semelhante ao Map, que também deve ser codificada pelo desenvolvedor.

Exemplo - Dataset

Nome do produto	Quantidade	Preço Unitário
Produto A	10	5.00
Produto B	5	7.50
Produto A	8	5.00
Produto C	3	12.00
Produto B	2	7.50
Produto A	12	5.00

Exemplo - Map

- Venda 1: ("Produto A", $50.00)
- Venda 2: ("Produto B", $37.50)
- Venda 3: ("Produto A", $40.00)
- Venda 4: ("Produto C", $36.00)
- Venda 5: ("Produto B", $15.00)
- Venda 6: ("Produto A", $60.00)
- Shuffle and Sort...

Exemplo – Reduce

- Para "Produto A": ("Produto A", [$50.00, $40.00, $60.00])
- Soma as receitas: ("Produto A", $150.00)

- Para "Produto B": ("Produto B", [$37.50, $15.00])
- Soma as receitas: ("Produto B", $52.50)

- Para "Produto C": ("Produto C", [$36.00])
- Soma as receitas: ("Produto C", $36.00)

Exemplo – Final

- ("Produto A", $150.00)
- ("Produto B", $52.50)
- ("Produto C", $36.00)

Exemplo – Algoritmo em Python

```
# Dados de exemplo
data = [
    ("Produto A", 10, 5.00),
    ("Produto B", 5, 7.50),
    ("Produto A", 8, 5.00),
    ("Produto C", 3, 12.00),
    ("Produto B", 2, 7.50),
    ("Produto A", 12, 5.00)
]

# Função de Mapeamento
def mapper(record):
    produto, quantidade, preco_unitario = record
    receita = quantidade * preco_unitario
    return (produto, receita)
```

```
# Função de Redução
def reducer(intermediate_data):
    grouped_data = {}
    for key, value in intermediate_data:
        if key not in grouped_data:
            grouped_data[key] = []
        grouped_data[key].append(value)

    result = []
    for key, values in grouped_data.items():
        total_receita = sum(values)
        result.append((key, total_receita))

    return result
```

```
# Executar a Fase de Mapeamento
intermediate_data = [mapper(record) for record in data]

# Executar a Fase de Redução
final_result = reducer(intermediate_data)

# Saída Final
for key, value in final_result:
    print(f"{key}: ${value:.2f}")
```

Vantagens

- **Escalabilidade Horizontal:** O Hadoop foi projetado para escalabilidade horizontal, o que significa que é capaz de lidar com grandes volumes de dados distribuídos em clusters de computadores. À medida que a quantidade de dados cresce, é possível adicionar mais nós ao cluster.

- **Processamento Distribuído:** O Hadoop distribui o processamento de dados em paralelo em vários nós do cluster. Isso permite o processamento eficiente de tarefas em grandes conjuntos de dados.

- **Tolerância a Falhas:** O Hadoop é projetado para ser tolerante a falhas. Se um nó falhar, as tarefas são redistribuídas para outros nós, garantindo que o processamento continue.

- **Acessível:** O Hadoop é de código aberto e gratuito. Isso o torna uma escolha acessível para organizações que desejam lidar com big data sem altos custos de licença.

- **Ecossistema Rico:** Além do Hadoop Distributed File System (HDFS) e do MapReduce, o ecossistema Hadoop inclui várias outras ferramentas e frameworks, como Hive, Pig, Spark e HBase, que permitem uma variedade de tarefas de análise de dados.

Desvantagens

- **Complexidade:** O Hadoop pode ser complexo de configurar e gerenciar. É necessário conhecimento técnico e experiente para operar efetivamente o ecossistema.
- **Latência:** O Hadoop não é a melhor escolha para tarefas que exigem baixa latência, pois é otimizado para processamento em lote. Tarefas em tempo real podem ser mais desafiadoras de implementar.
- **Uso Ineficiente de Recursos:** Em algumas situações, o Hadoop pode usar recursos de hardware de forma ineficiente, especialmente para tarefas de processamento de dados menores.
- **Curva de Aprendizado:** Aprender a usar efetivamente todas as ferramentas e componentes do ecossistema Hadoop pode ser uma curva de aprendizado íngreme, especialmente para iniciantes.
- **Armazenamento Redundante:** O armazenamento em HDFS envolve duplicação de dados (redundância) para fins de tolerância a falhas. Isso pode consumir mais espaço de armazenamento do que sistemas de armazenamento convencionais.