Big Data: Hadoop Ecosystems

Tien-Lam Pham

lam.phamtien@phenikaa-uni.edu.vn

PIAS
VIỆN NGHIÊN CỨU
TIÊN TIẾN PHENIKAA

PHENIKAA
UNIVERSITY

Outline

- Introduction to Parallel computing
- Big-data and Hadoop
- Hadoop Distributed File Systems (HDFS)
- Hadoop Cluster Architecture
- HDFS: Reading and Writing

Computer vs. Human

- **Computer:**
  - ✓ Computing: billion of calculations / s: **+, -, *, /, NOT, AND** ..
  - ✓ Remember: 100s GB
  - ✓ Do what you tell it to do -> Programming -> instructions
- **Human:**
  - ✓ Observe nature
  - ✓ Imagine
  - ✓ Learn
  - ✓ Calculation?
  - ✓ Remember important features?
  - ✓ Interact with environment and action independently

Knowledge

- Declarative knowledge: “chuồn chuồn bay thấp thì mưa” (Human-friendly)
- Imperative knowledge (recipe; how-to -> computer-friendly):
  - ✓ Open terminal
  - ✓ Run “ipython” command
  - ✓ Import “numpy”
  - ✓ Create a random number
  - ✓ Select student corresponding to the random number: Winer

Recipe

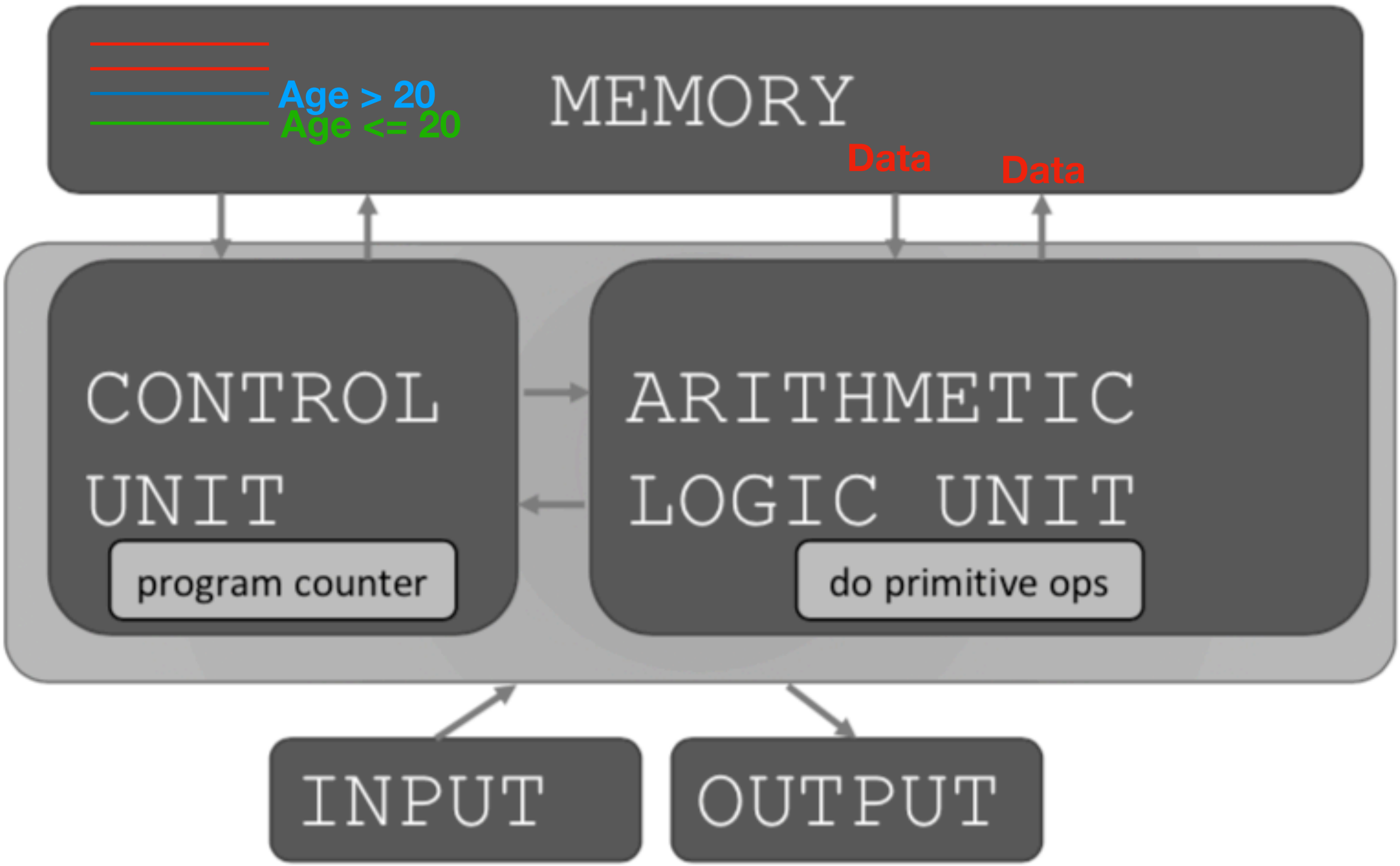
1) sequence of simple **steps**
2) **flow of control** process that specifies when each step is executed
3) a means of determining **when to stop**

1+2+3 = an **algorithm**!

Computer

- Perform calculations following a recipe (algorithm)
- Fixed program computer: calculators
- Stored program computer: storing and performing instructions

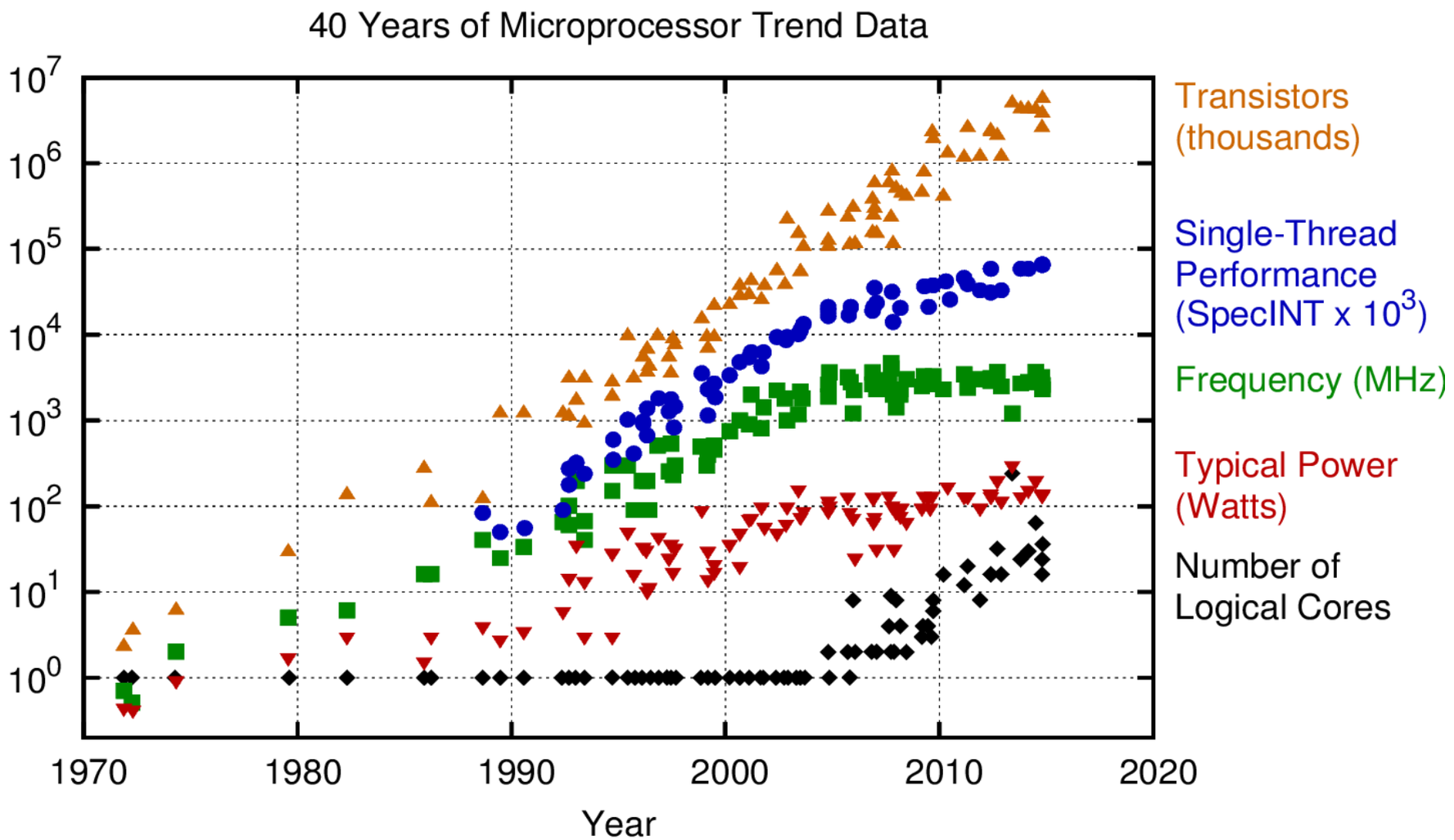
Computer


Age > 20
Age <= 20
MEMORY
Data
Data
CONTROL UNIT
program counter
ARITHMETIC LOGIC UNIT
do primitive ops
INPUT
OUTPUT

Computer program

- sequence of **instructions stored** inside computer
  - built from predefined set of primitive instructions
    1) arithmetic and logic **+, -, *, /, NOT, AND ..**
    2) simple tests
    3) moving data
- special program (interpreter) **executes each instruction in order**
  - use tests to change flow of control through sequence
  - stop when done

How to Speed up Computing?



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2015 by K. Rupp

Source: karlrupp.net

Parallel computing

```
C[I] = A[I] + B[I]
```

a1	a2	...	a[p]	a[1*p+1]	...	a[1*p+p]	...	a[i*p+1]	...	a[i*p+p]
b1	b2	...	b[p]	b1*p+1]	...	b[1*p+p]	...	b[i*p+1]	...	b[i*p+p]
c1	c2	...	c[p]	b1*p+1]	...	c[1*p+p]	...	c[i*p+1]	...	c[i*p+p]

Process 1
(thread 1)

Process 2
(thread 2)

Process i
(thread i)

CPU-core 1

CPU-core 2

CPU-core i

Computing Machine

SUPERCOMPUTER

360Node/ 5760 CPU-Core; Total memory: 22.5TB; Performance 119.8TFLPS

COURTESY: CRAY

GP-GPU

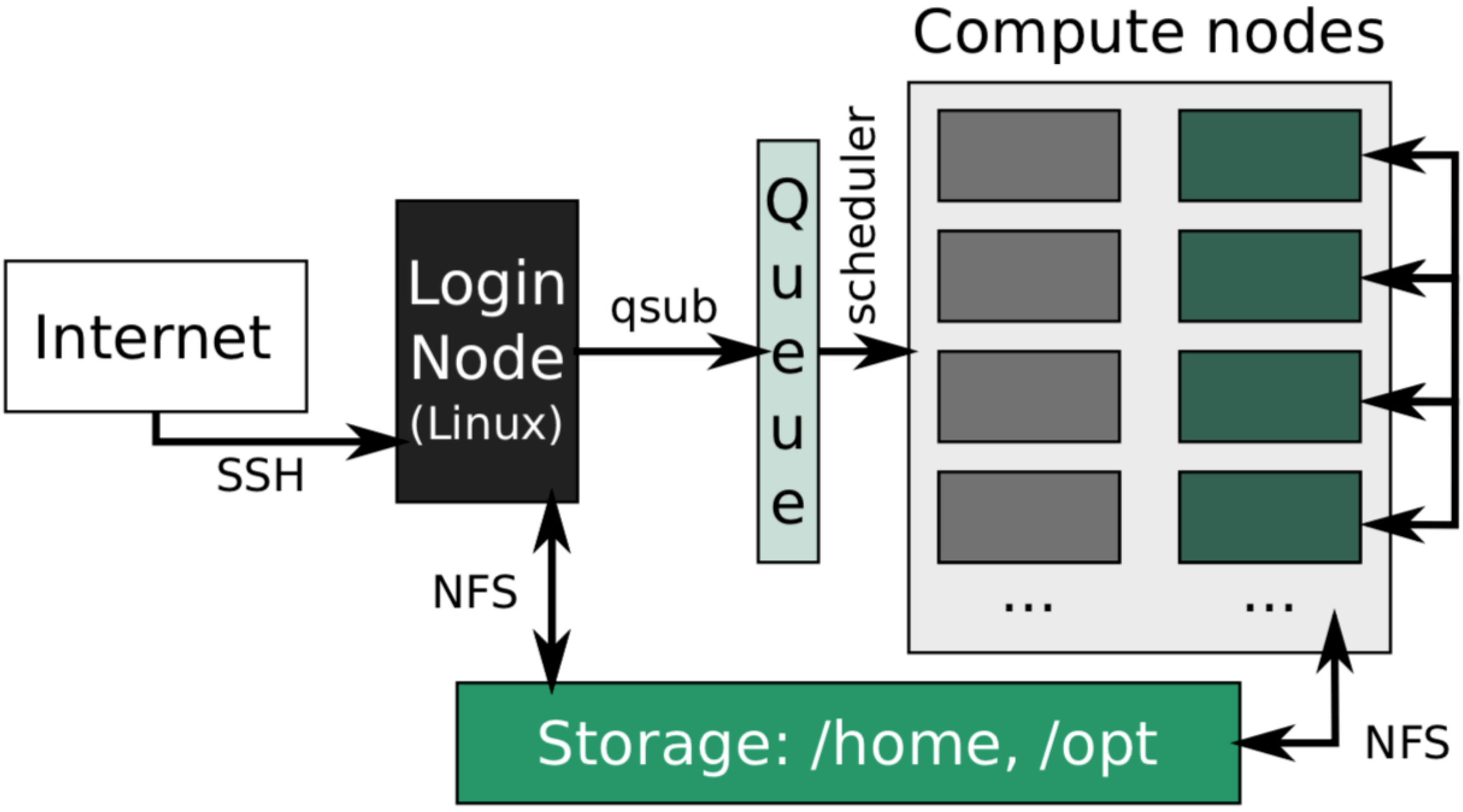
NVIDIA DGX Systems

The World's First Portfolio of Purpose-Built AI Supercomputers

Introducing the world's first portfolio of purpose-built AI supercomputers. Inspired by the demands of deep learning and analytics, NVIDIA® DGX™ solutions are made up of a powerful, fully integrated combination of innovative, GPU-optimized software, groundbreaking performance, and simplified management.

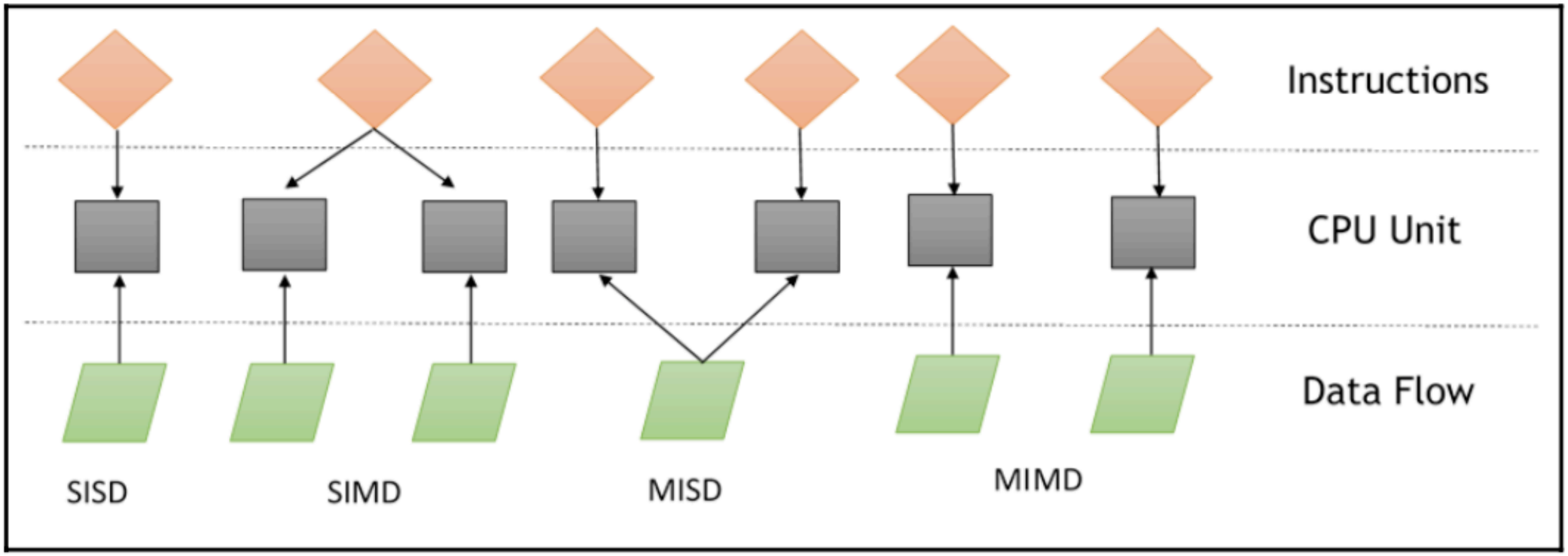
LEARN MORE

High-Performance Computer



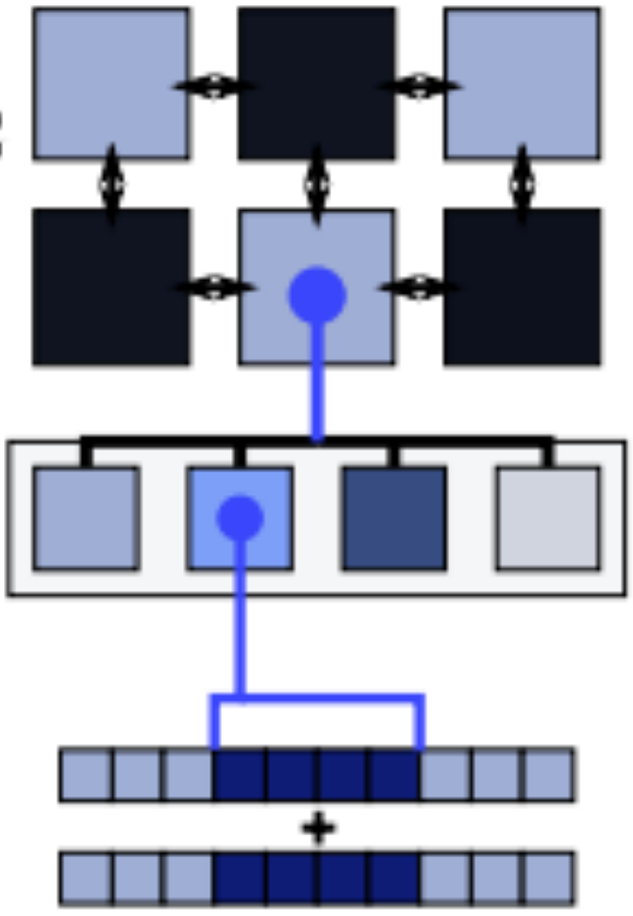
Parallel computing

As the instruction and data flows are independent, there are four categories of parallel machines: **Single Instruction Single Data (SISD)**, **Single Instruction Multiple Data (SIMD)**, **Multiple Instruction Single Data (MISD)**, and **Multiple Instruction Multiple Data (MIMD)**:



Multiprocessing

CLUSTER COMPUTING
in distributed memory



```
MPI_Sendrecv(data, k,
    MPI_DOUBLE, data2,
    ... );
```

MULTITHREADING
in shared memory

```
#pragma omp parallel for
for (j = 0; j < m; j++)
  ComputeSubset(j);
```

VECTORIZATION
of floating-point math

```
#pragma omp simd
for (i = 0; i < n; i++)
  A[i] += B[i];
```

Message Passing Interface

- Specification for message passing
- Multiple implementations exist

- Portable
- Efficient
- Designed for computing

- Distributed-memory computing
- Multiprocessing in shared memory

Open MPI:
Open Source High Performance Computing

MPICH is a high performance and widely portable implementation of the **Message Passing Interface (MPI)** standard.

Intel® MPI Library

Deliver flexible, efficient, and scalable cluster messaging.

MPI

```python
from mpi4py import MPI

comm = MPI.COMM_WORLD
rank = comm.Get_rank()
```

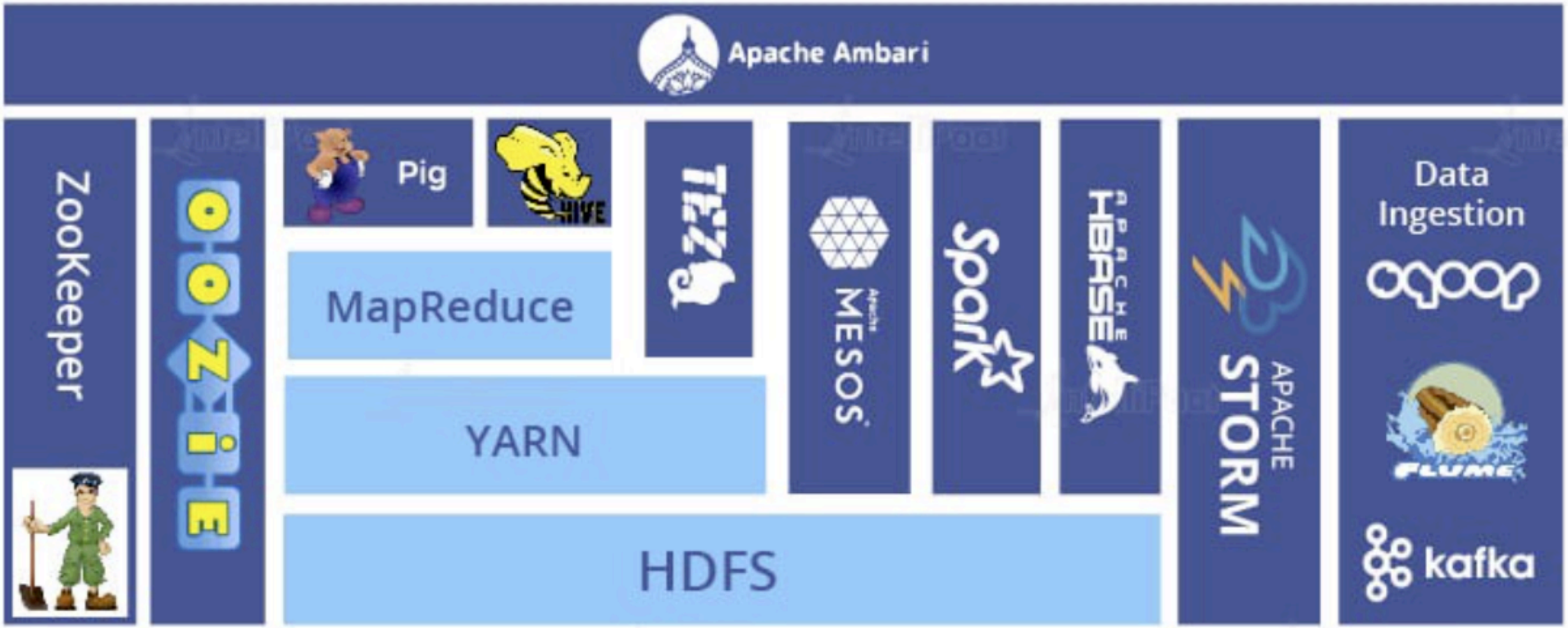
```
mpirun -n 2 python hello_mpi.py
```

MPI point-to-point communication

```python
from mpi4py import MPI

comm = MPI.COMM_WORLD
rank = comm.Get_rank()

if rank == 0:
    data = {'a': 7, 'b': 3.14}
    comm.send(data, dest=1, tag=11)
elif rank == 1:
    data = comm.recv(source=0, tag=11)
```

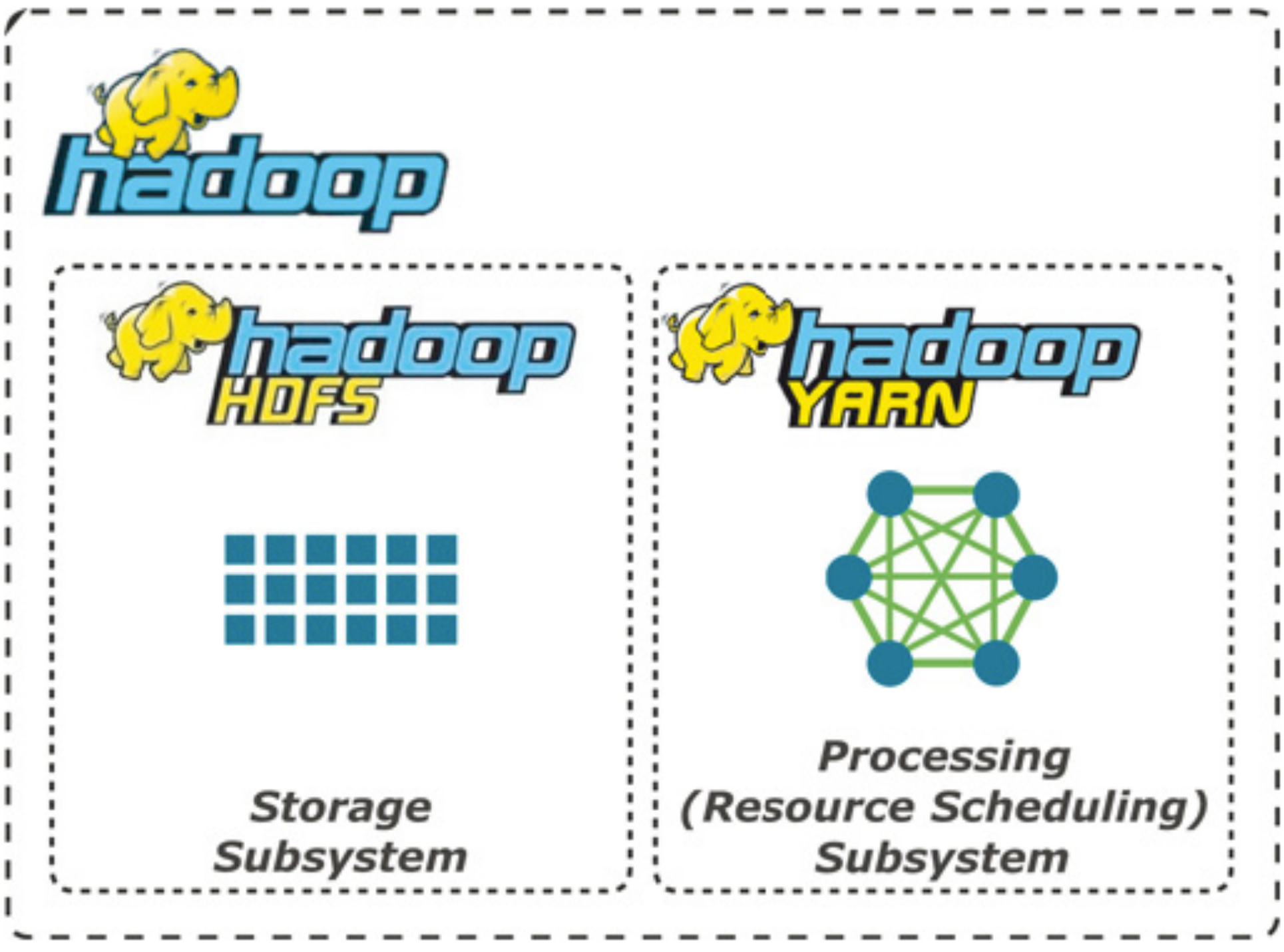
```python
from mpi4py import MPI
import numpy

comm = MPI.COMM_WORLD
rank = comm.Get_rank()

# passing MPI datatypes explicitly
if rank == 0:
    data = numpy.arange(1000, dtype='i')
    comm.Send([data, MPI.INT], dest=1, tag=77)
elif rank == 1:
    data = numpy.empty(1000, dtype='i')
    comm.Recv([data, MPI.INT], source=0, tag=77)

# automatic MPI datatype discovery
if rank == 0:
    data = numpy.arange(100, dtype=numpy.float64)
    comm.Send(data, dest=1, tag=13)
elif rank == 1:
    data = numpy.empty(100, dtype=numpy.float64)
    comm.Recv(data, source=0, tag=13)
```

Hadoop ecosystem: Overview



https://medium.datadriveninvestor.com/hadoop-ecosystem-cebb03315b69

Hadoop Core components



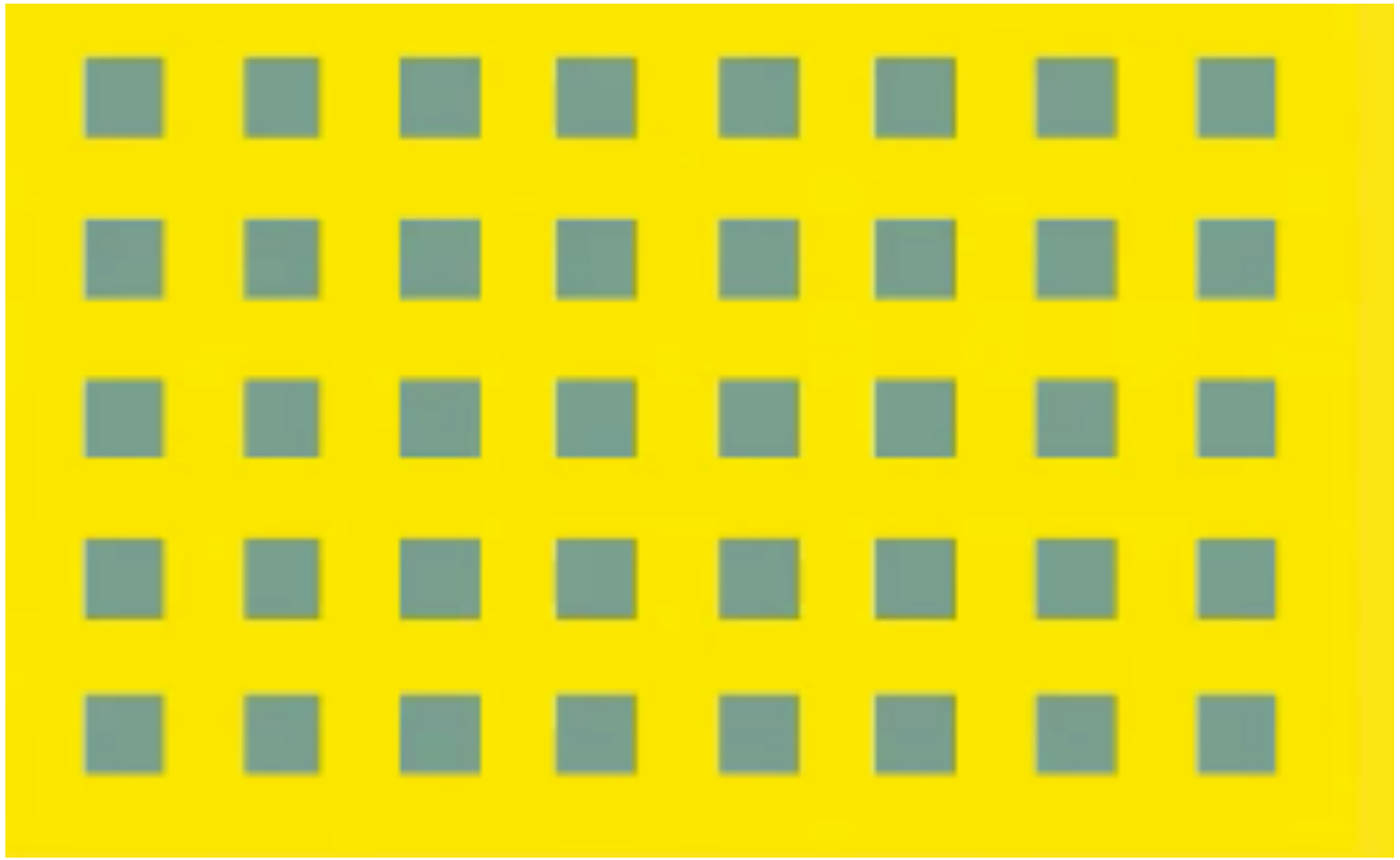
The Commercial Hadoop

Hadoop Usage

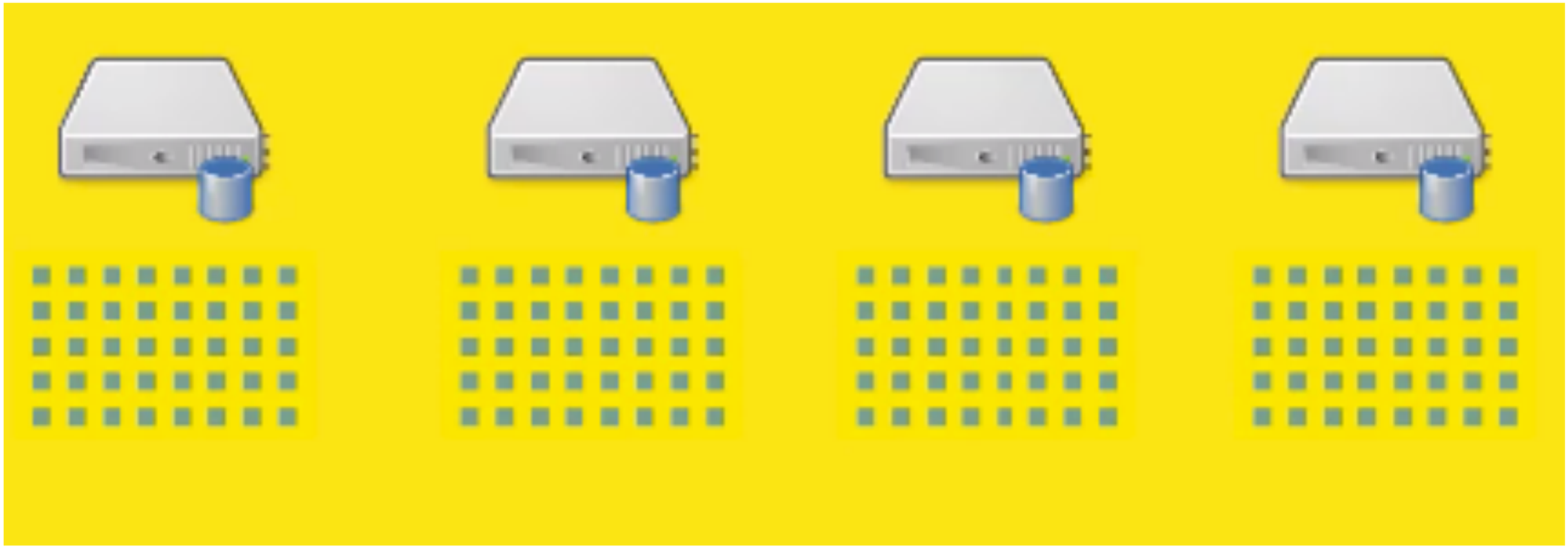
- Data warehouse: Extract-Load-Transform, Extract-Transform-Load
- Event and complex event processing
- Advanced analytics

Hadoop Distributed File Systems (HDFS)

- Handling big file -> blocks
- Distributed processing over blocks
- Store and process across computes

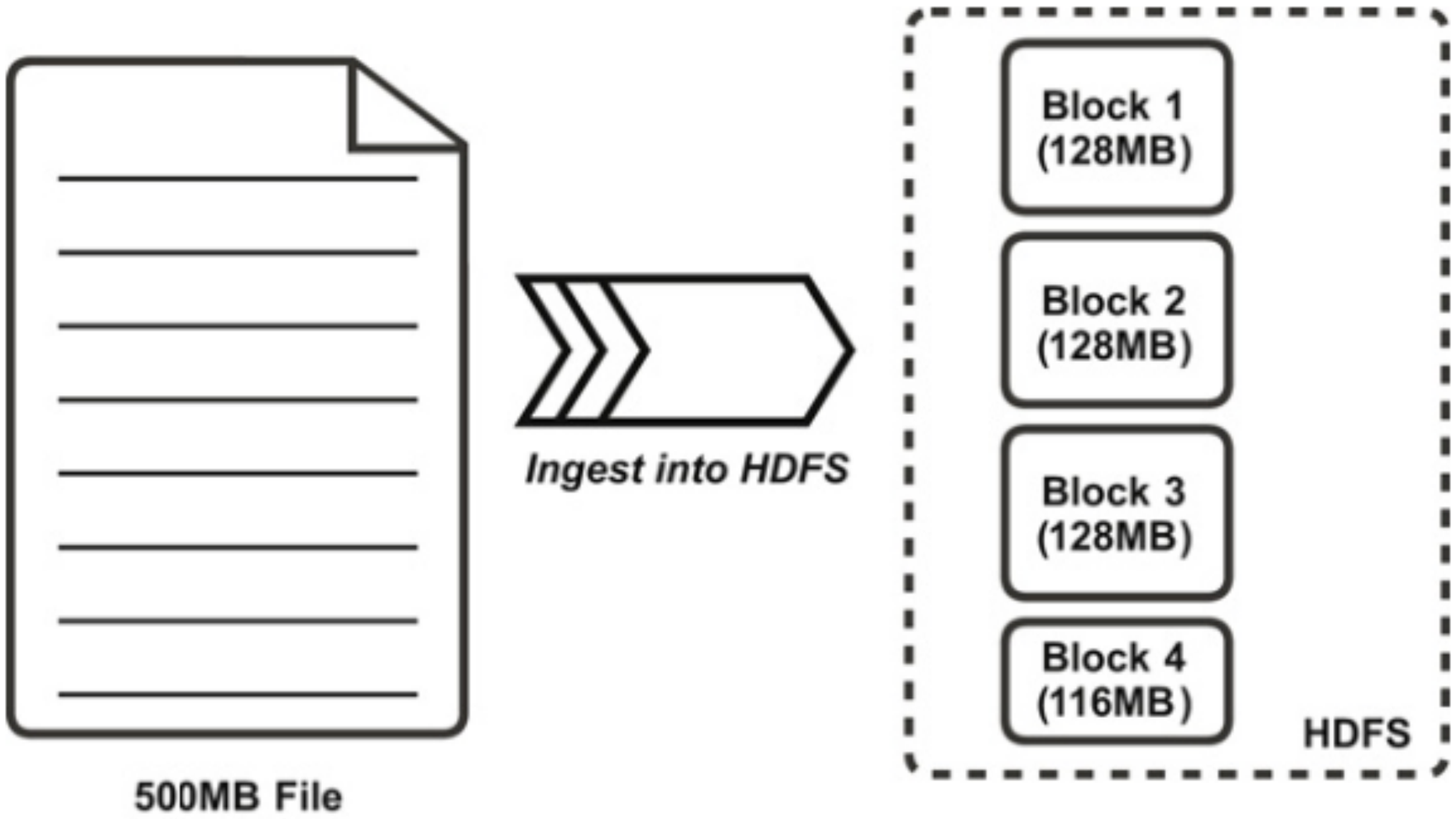


Stored across computers



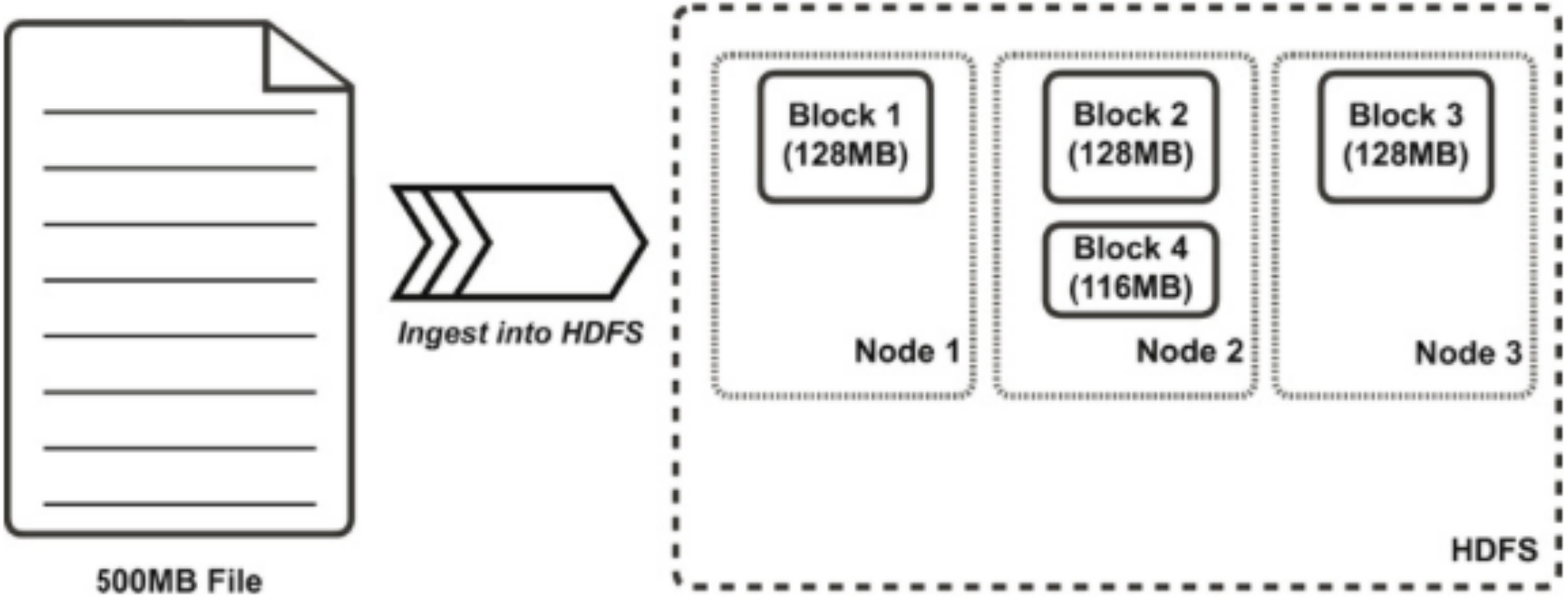
Hadoop Distributed File Systems (HDFS)

- Handling big file -> blocks
- Distributed processing over blocks
- Store and process across computes

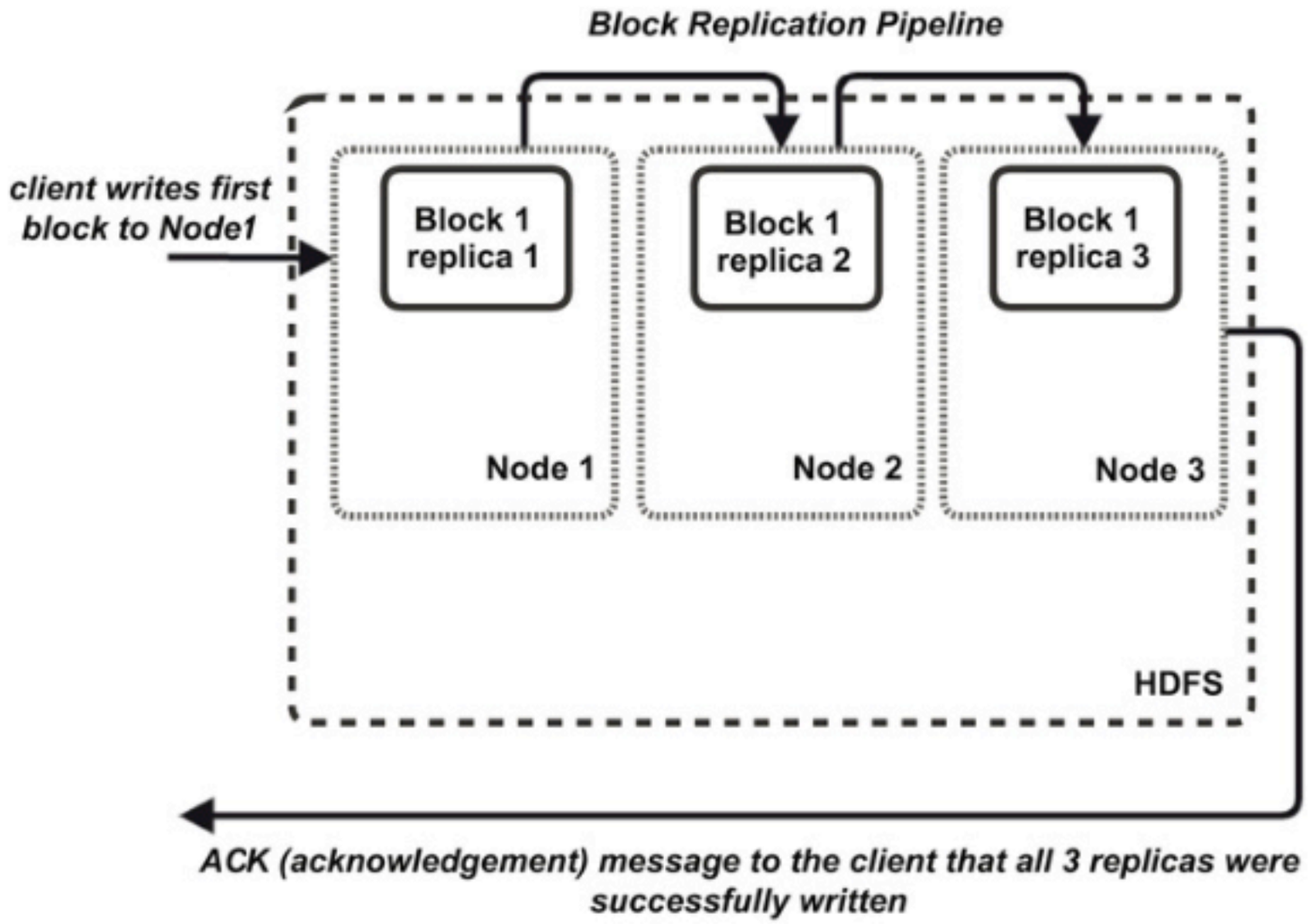


Hadoop Distributed File Systems (HDFS)

- Handling big file -> blocks
- Distributed processing over blocks
- Store and process across computes

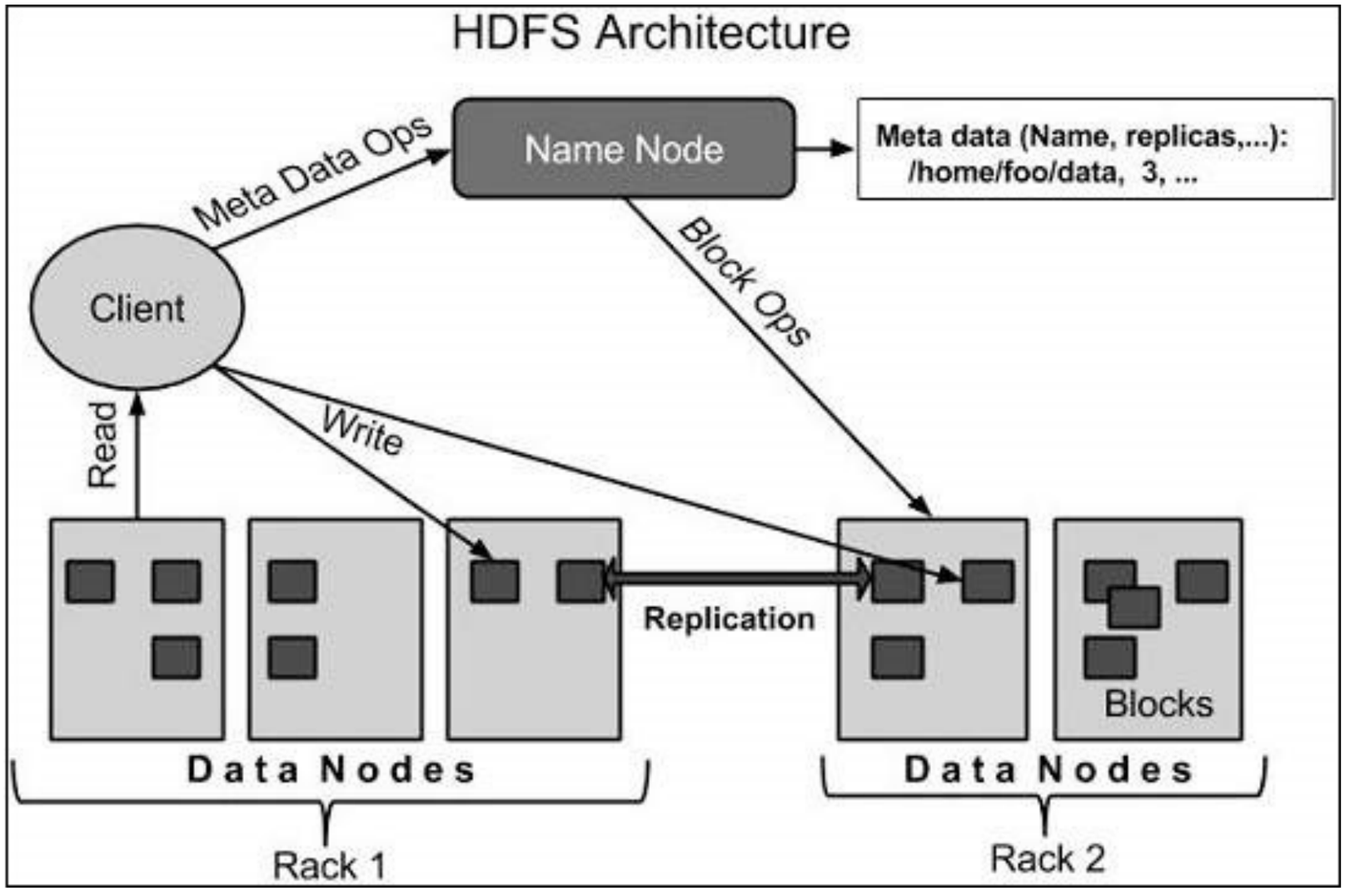


Block Replication pipeline



HDFS Architecture

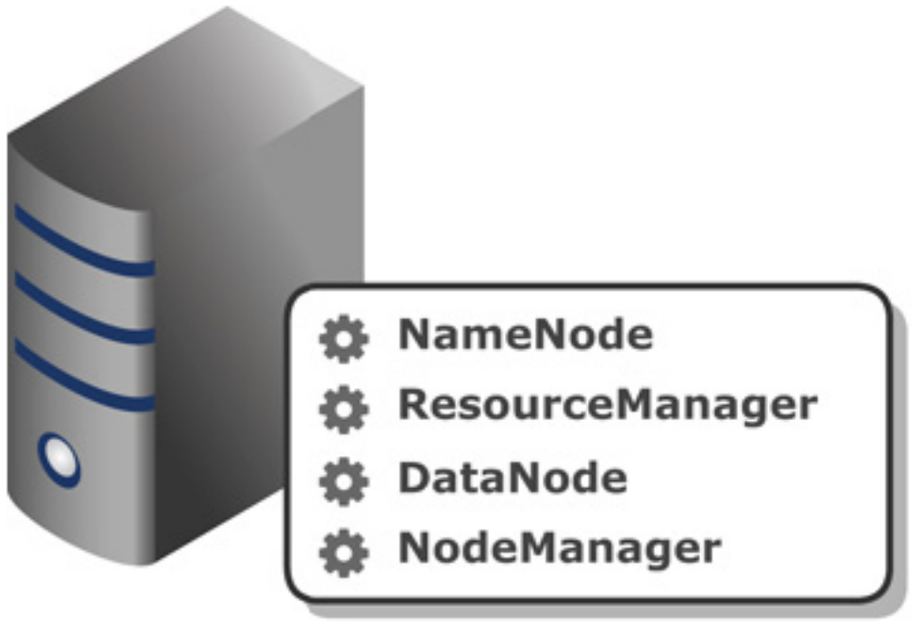
Meta data: dữ liệu về các bocks of data, index, địa chỉ lưu trên các datanode



https://www.tutorialspoint.com/hadoop/hadoop_hdfs_overview.htm

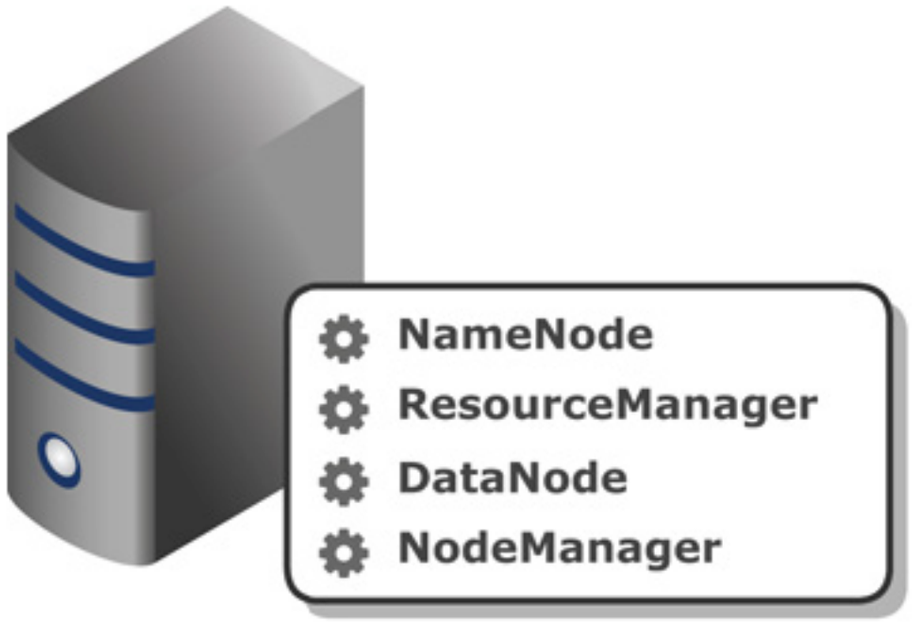
NameNode

- Governs the distributed filesystem
- Manage the filesystem's metadata
- Metadata is on memory
- Metadata: locations of blocks in HDFS
- NameNode services queried by clients: MapReduce, Spark, and others

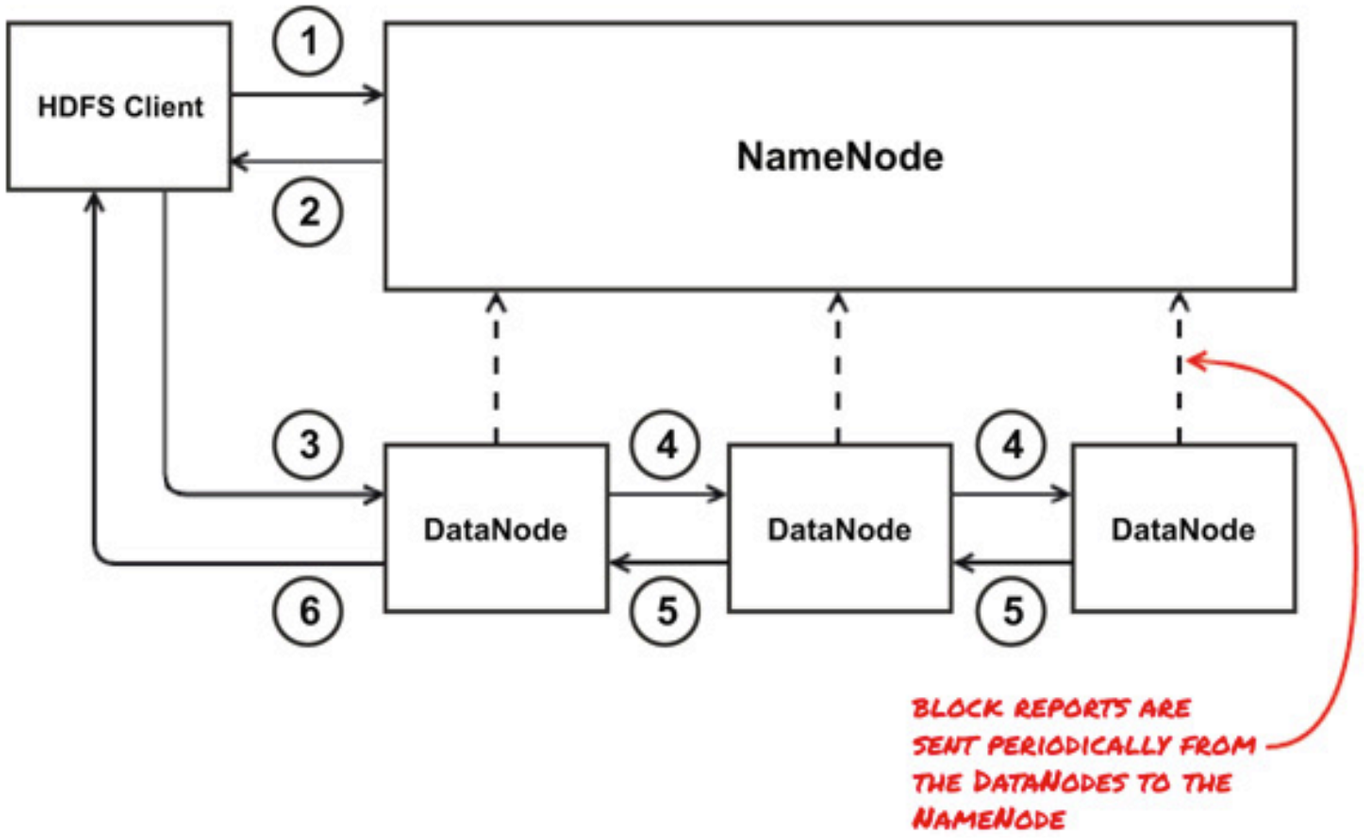


DataNode

- Participating in the block replication pipeline
- Managing local volumes and storage
- Providing block reports to the NameNode

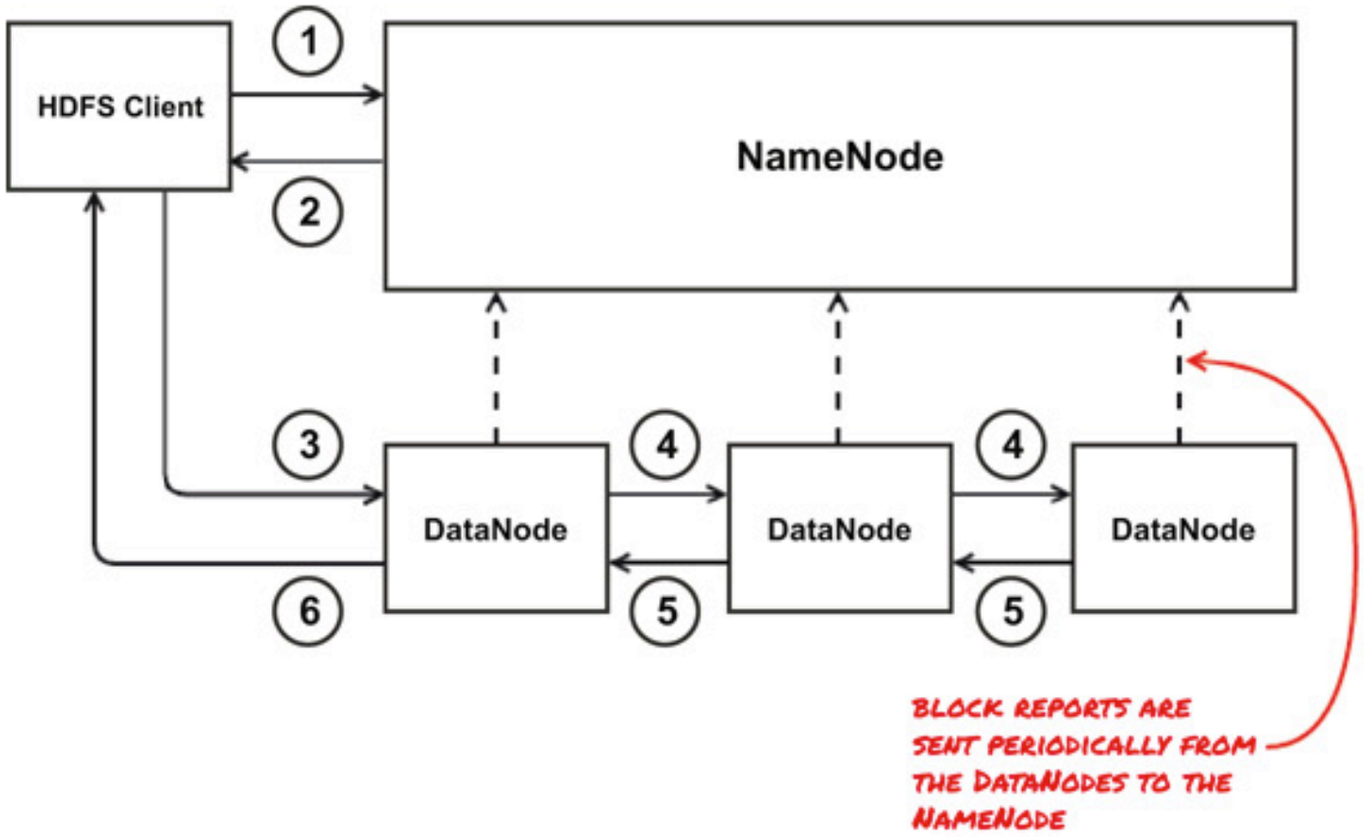


HDFS Writing process



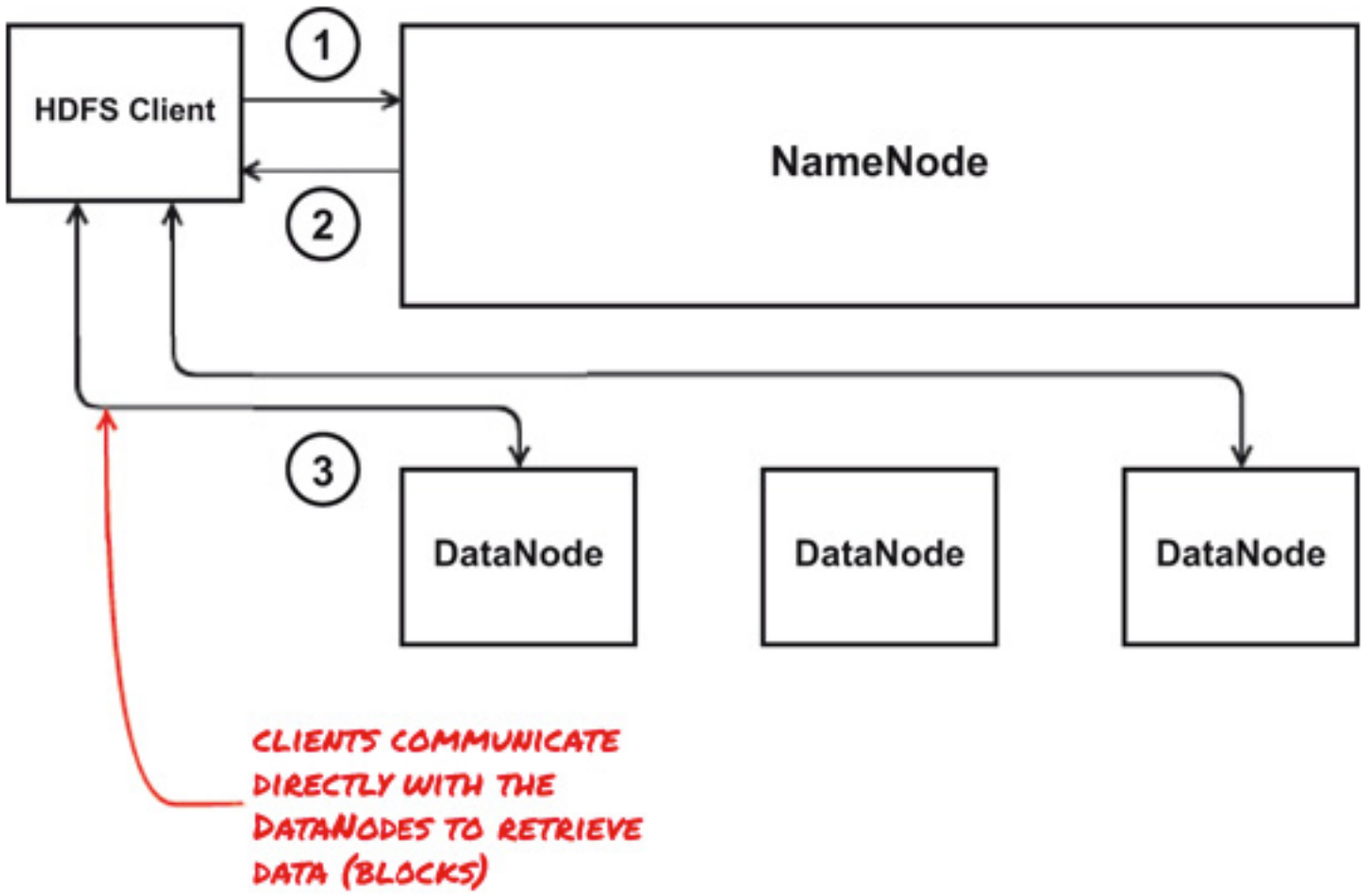
1. The HDFS Client requests to write a file block.
2. The NameNode responds to the Client with the DataNode to which to write the block.
3. The Client requests to write the block to the specified DataNode.
4. The DataNode opens a block replication pipeline with another DataNode in the cluster, and this process continues until all configured replicas are written.
5. A write acknowledgment is sent back through the replication pipeline.
6. The Client is informed that the write operation was successful.

HDFS Writing process



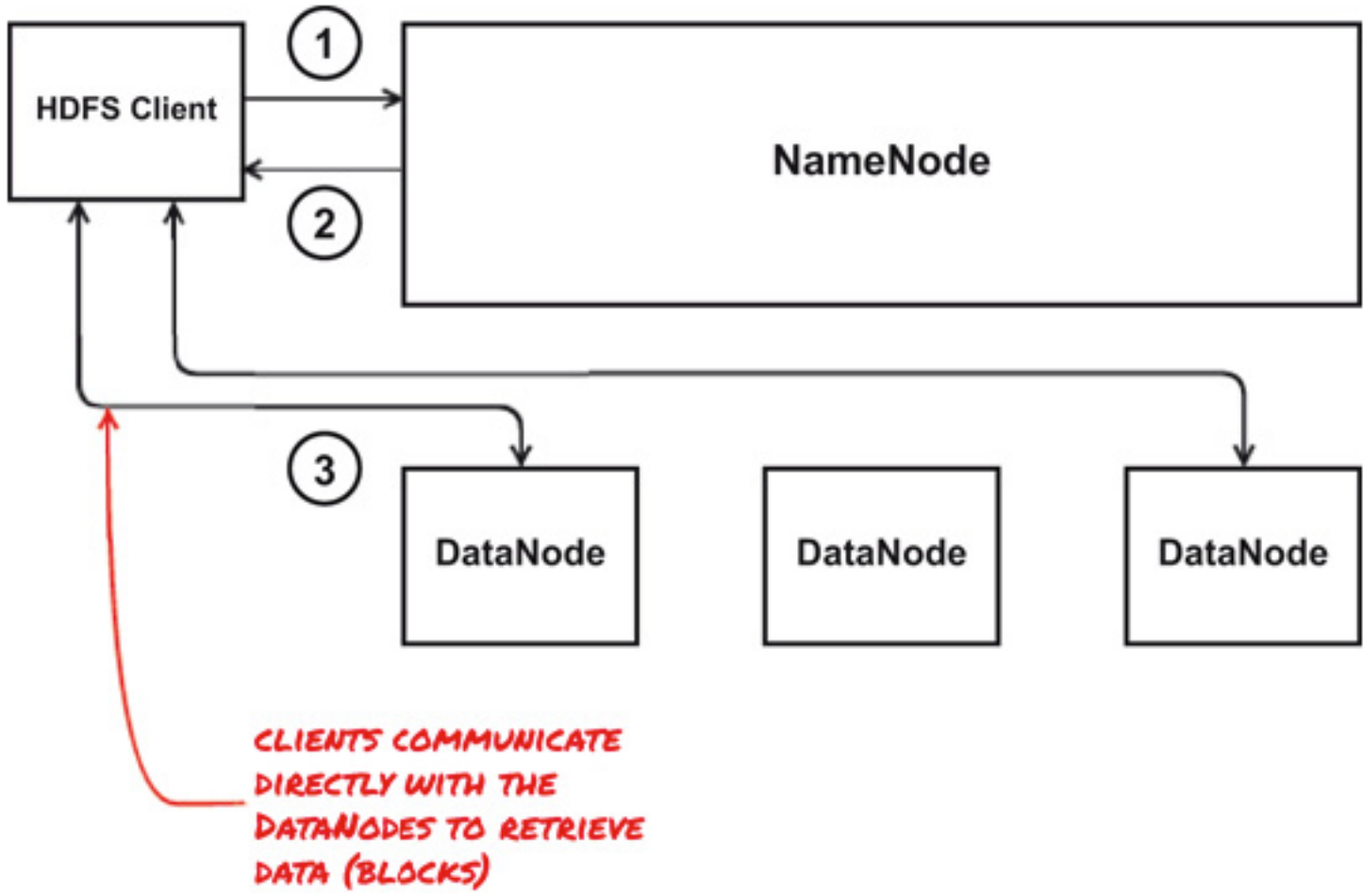

```
hadoop fs -copyFromLocal warandpeace.txt /data/books

hdfs dfs -put warandpeace.txt /data/books
```

HDFS Reading process



1. The HDFS Client requests to read a file.
2. The NameNode responds to the request with a list of DataNodes containing the blocks that comprise the file.
3. The Client communicates directly with the DataNodes to retrieve blocks for the file.

HDFS Reading process




```
hadoop fs -get /data/reports/report.csv
hadoop fs -ls /data/reports
hadoop fs -rm -r /data/reports
```