Recap

- Data vs Information
- Evolution of Data Management Technologies
- Big Data and its characteristics
- Application
- How to process Big Data?
- Various computing technologies
- History of Hadoop
- RDBMS vs Hadoop
- Major components of Hadoop cluster

Agenda for today

- The Hadoop Distributed File System
- MapReduce detailed discussion
- Hadoop 1 vs 2
- Various Hadoop installation modes
- Running your first MapReduce program

Design your own HDFS

- Parameters to consider
  - Storing very large files
  - Underlying hardware is commodity hence failures are common
  - Physical hardware can be located anywhere
  - Any missed out?

HDFS

- HDFS is a file system designed for storing very large files with streaming data access patterns, running on clusters of commodity hardware
- Very large files
- Streaming data access
- Commodity hardware

HDFS Blocks

- Single unit of storage
- Size of block will drive the ratio of time to read a block to the seek for a block

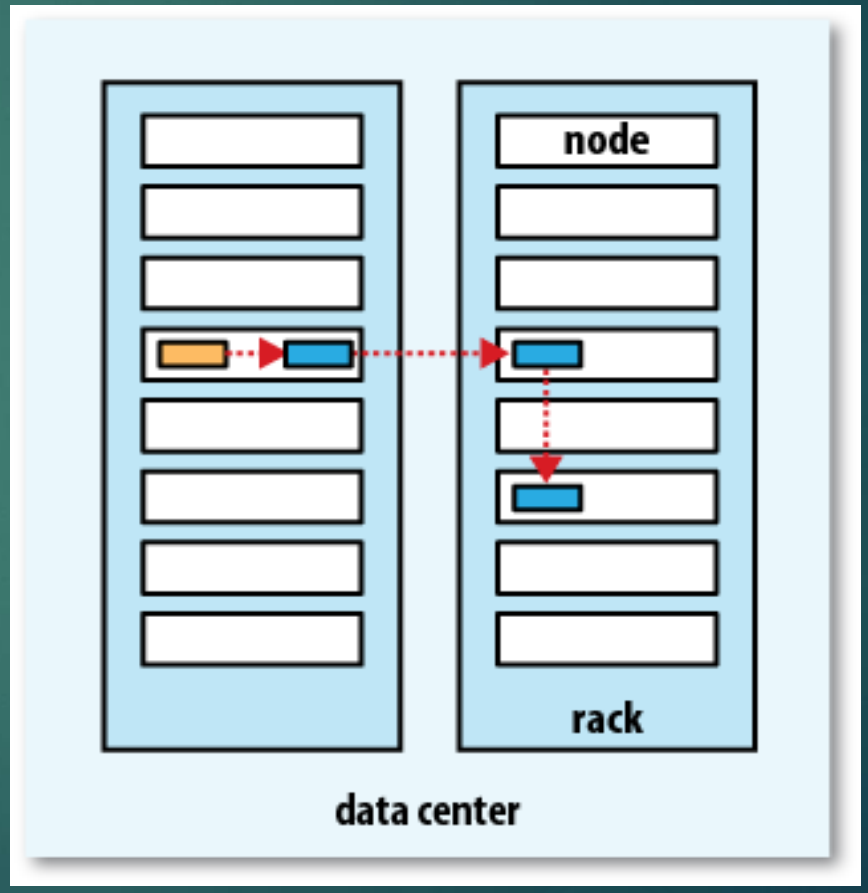


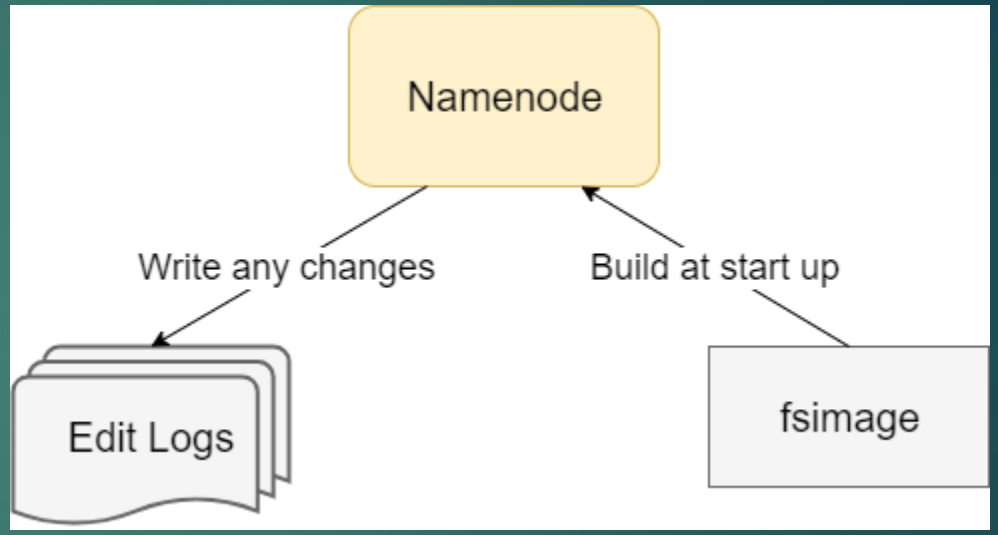
Image Ref: Hadoop definitive guide 4th edition

Benefits of blocks

- Files can be larger than a single disk
- Simplicity at storage level as data node doesn't store any metadata
- Easy to calculate capacity of a node
- Fault tolerance by replicating blocks

File system metadata

- Who stores the metadata?
- Backup of metadata
- Role of secondary namenode



Network Topology

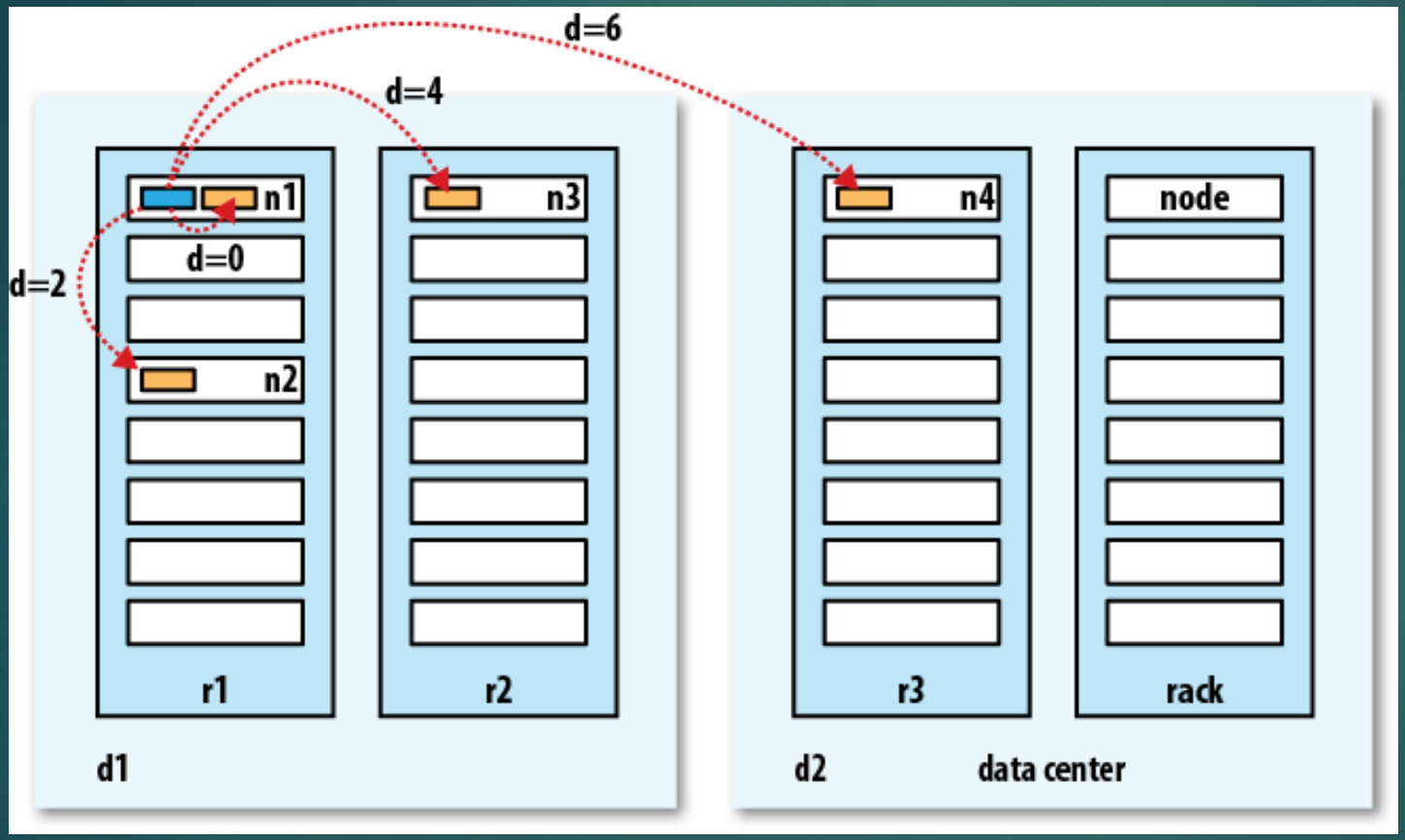


Image Ref: Hadoop definitive guide 4th edition

HDFS Read operation

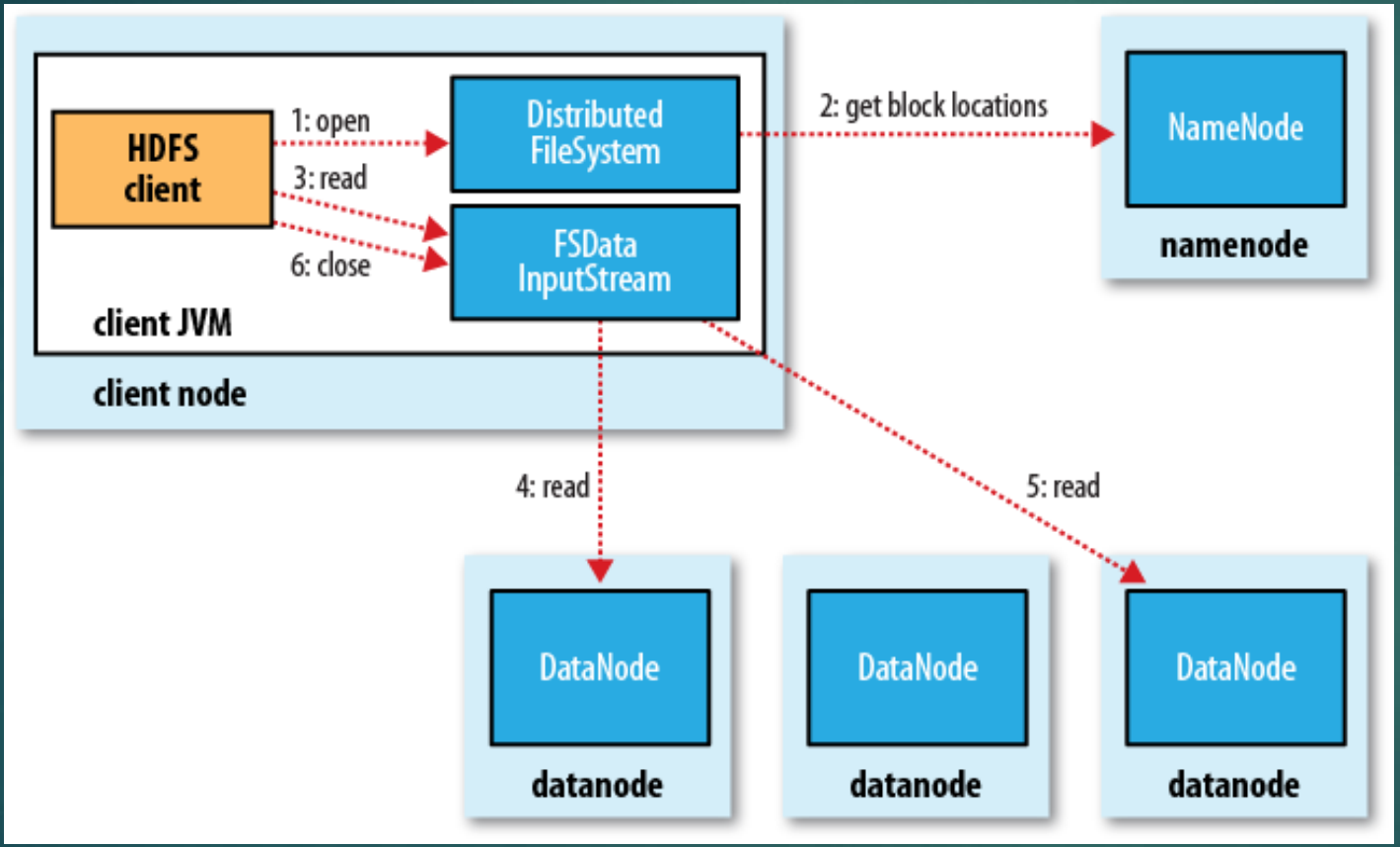


Image Ref: Hadoop definitive guide 4th edition

HDFS Write operation

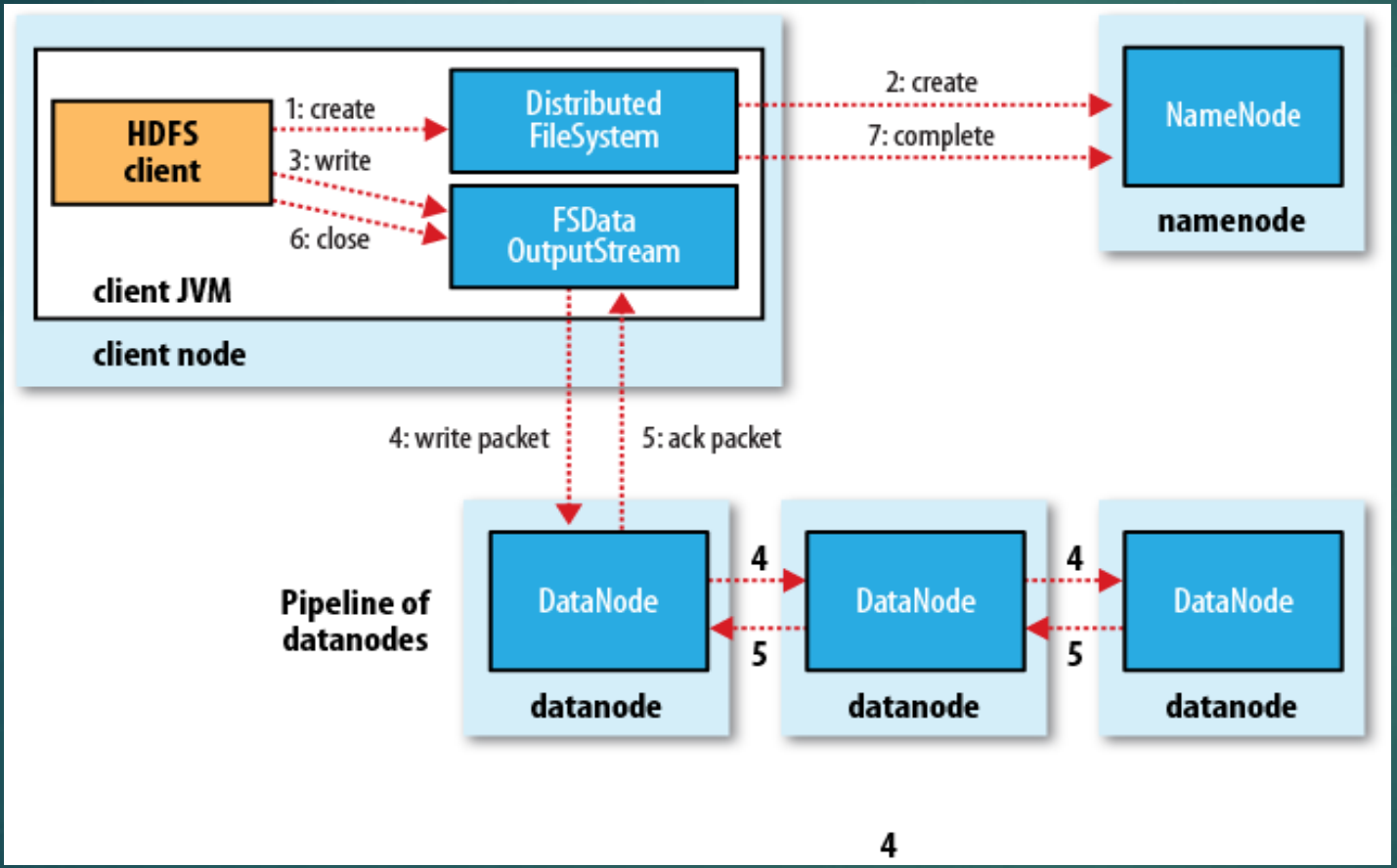


Image Ref: Hadoop definitive guide 4th edition

HDFS not made for

- Low-latency data access
- Lots of small files
- Multiple writers, arbitrary file modifications

Activity

- Data file: http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html
- Goal is to count #requests per day
- Discuss a programming approach for that

MapReduce

- Two major phases: Map and Reduce
- Notion of <Key, Value> pairs
- Divides job into multiple tasks
- Map: extract important information from each record
- Reduce: Aggregate, Summarize, Filter, Transform

MapReduce Stages

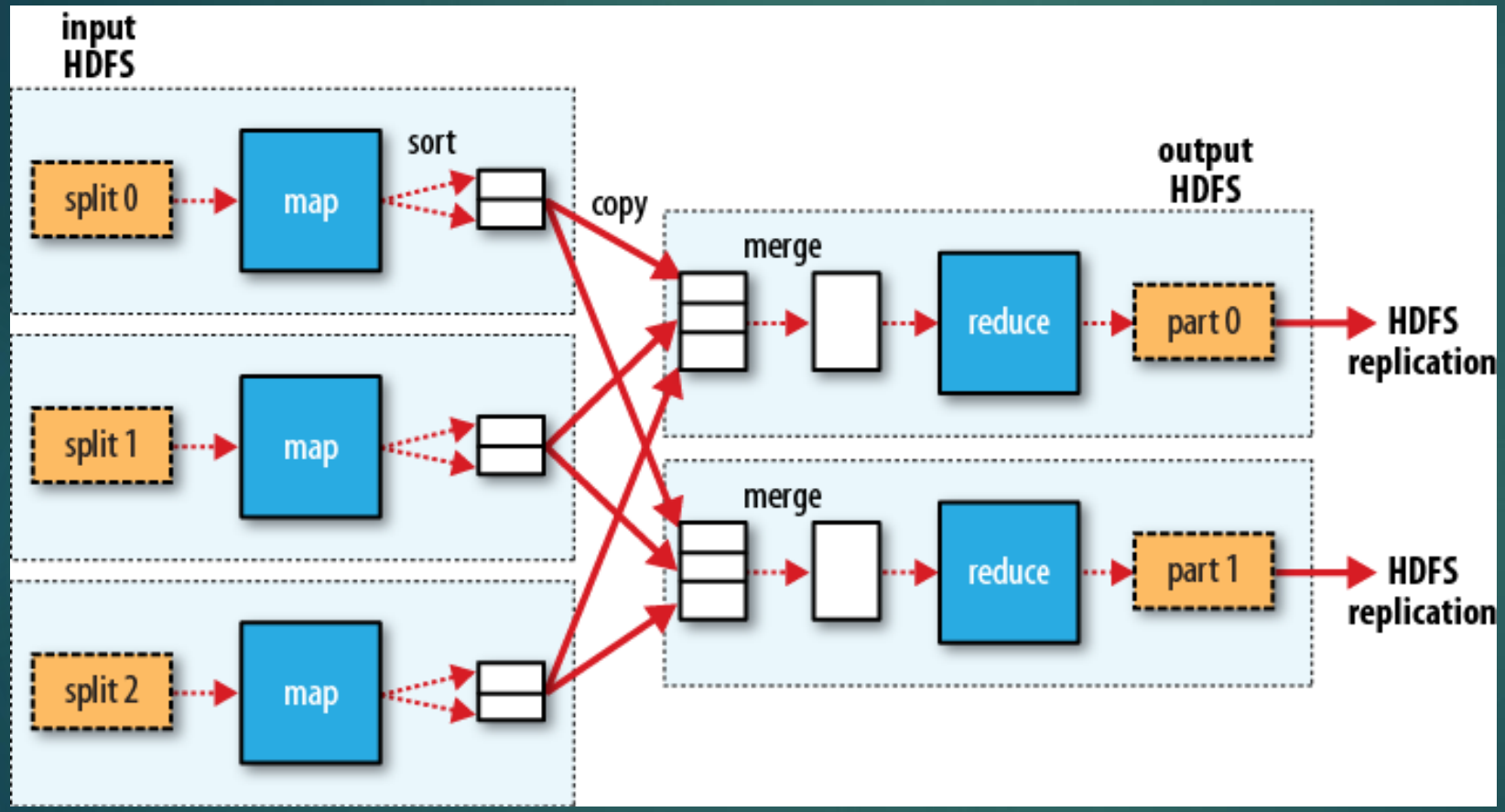


Image Ref: Hadoop definitive guide 4th edition

Map Tasks

- What is a good approach to decide how many map tasks a job should launch?
- Less number of big tasks

  vs

  higher number of small tasks
- Normally same as input data blocks
- Task to node mapping
- Notion of data locality

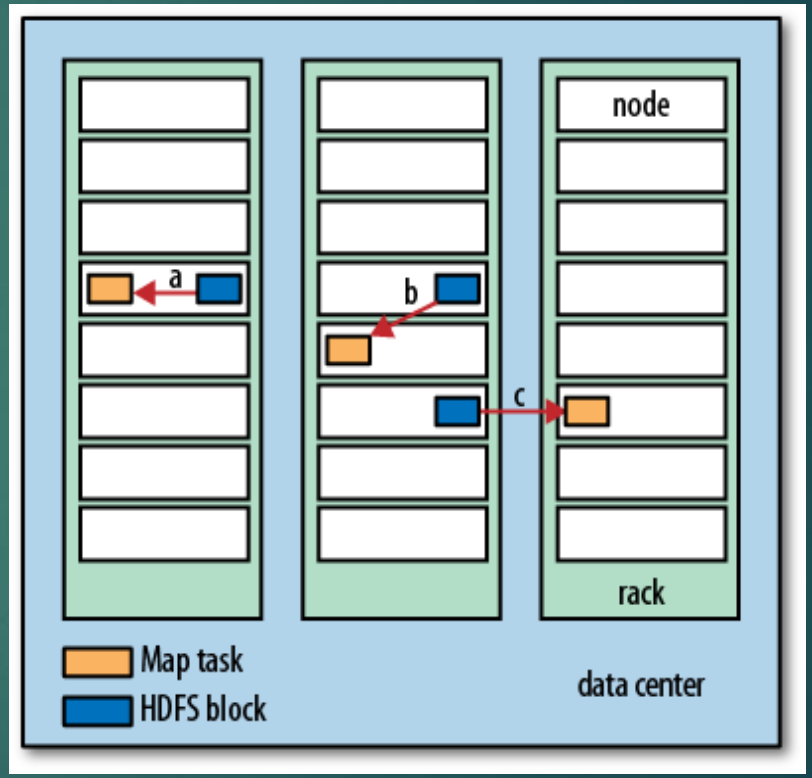


Image Ref: Hadoop definitive guide 4th edition

Input formats

Input format	Description
TextInputFormat	Read Text file line by line. Key is offset and value is record text
KeyValueTextInputFormat	Tab separated key values from a text file
SequenceFileInputFormat<K,V>	Hadoop's file format
NLineInputFormat	Each split is guaranteed of N lines for TextInputFormat.

Input Splits

- Blocks are of fixed size
- Good chances of records being split between two block

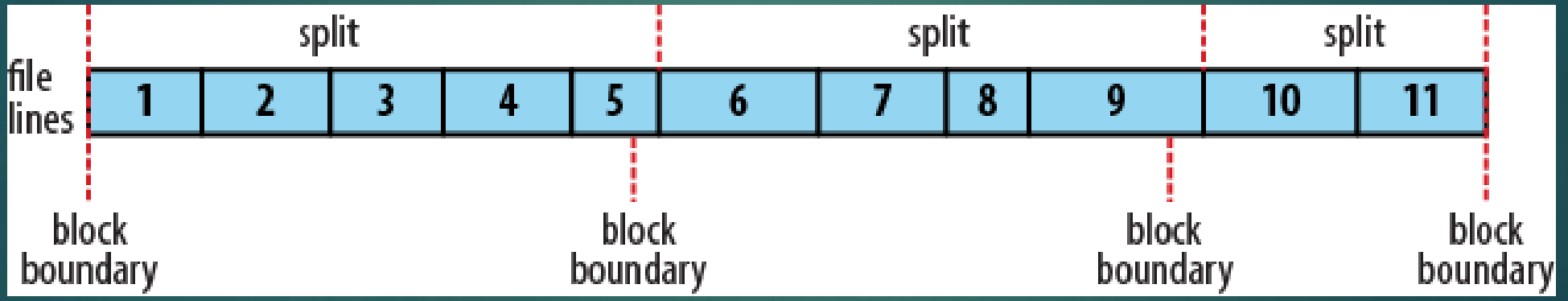


Image Ref: Hadoop definitive guide 4th edition

Reduce Tasks

- Can be configured by programmer
- Normally same as #datanodes participating in execution
- Input Key and Value type should be same as output type of mapper
- One output file per reducer under output directory
- Generates exception if output directory already exists. Why?

Output Formats

Output Format	Description
TextOutputFormat<K,V>	Tab separated key value pairs in plain text format. One record per key value pair
SequenceFileOutputFormat<K, V>	Hadoop's Sequence file format
NullOutputFormat<K,V>	Nothing. Helps in map only job

Intermediate Operations

- Sort
- Partition
- Shuffle
- Merge and Sort

MapReduce: Mapper Code


```
public class WebHitCounterMapper extends Mapper<Input Key, Input Value, Output Key, Output Value> {

    public void map(Input Key, Input Value, Context context) throws IOException, InterruptedException {




            context.write(Output Key, Output Value)
    }
}
```

MapReduce: Reducer Code


```
public class WebHitCounterReducer extends Reducer<Input Key, Input Value, Output Key, Output Value> {

    public void reduce(Input Key, Iterable<Value Data type> values,Context context) throws IOException, InterruptedException {









        context.write(Output Key, Output Value);
    }
}
```

MapReduce: Driver Code


```
public class WebHitCounterMain {
    public static void main(String[] args) throws Exception {

        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "Daily Web Hit Counter");

        job.setJarByClass(main.WebHitCounterMain.class);
        job.setMapperClass(mapper.WebHitCounterMapper.class);
        job.setReducerClass(reducer.WebHitCounterReducer.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

Challenges with Hadoop 1

- Applications were limited to MapReduce implementations only
- Namenode machine crash or maintenance activity
- Namespace scaling
- Backup and Recovery
- Batch oriented architecture
- Support for various file formats
- Dual responsibilities of Job tracker

Image Ref: https://www.greycampus.com/blog/big-data/top-differences-between-hadoop-1-0-and-hadoop-2-2

Hadoop 2

- Support for other data processing engines
- High Availability
- HDFS Federation
- HDFS Snapshot
- Introduced Streaming and Interactive analysis
- Support for various file formats
- Yarn

YARN

- Yet Another Resource Negotiator

MapReduce 1	YARN
Job Tracker	Resource Manager, Application Master and Timeline server
Task Tracker	Node Manager
Slot	Containers

YARN model

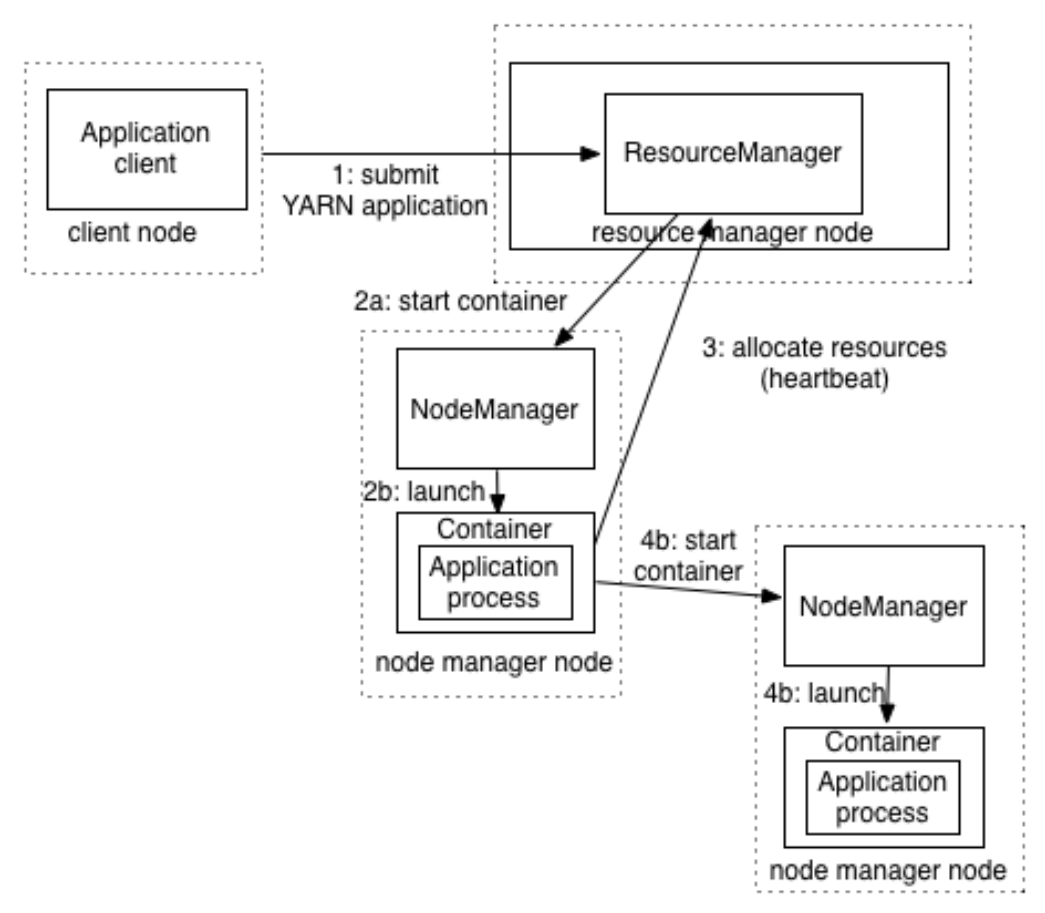


Image Ref: Hadoop definitive guide 4th edition

Pros of YARN

- Scalability
- Availability
- Utilization
- Multitenancy

Hadoop installation modes

- Standalone – Single node cluster
- Pseudo distributed mode - Single node cluster
- Distributed mode – Multi node cluster

Programming Exercise