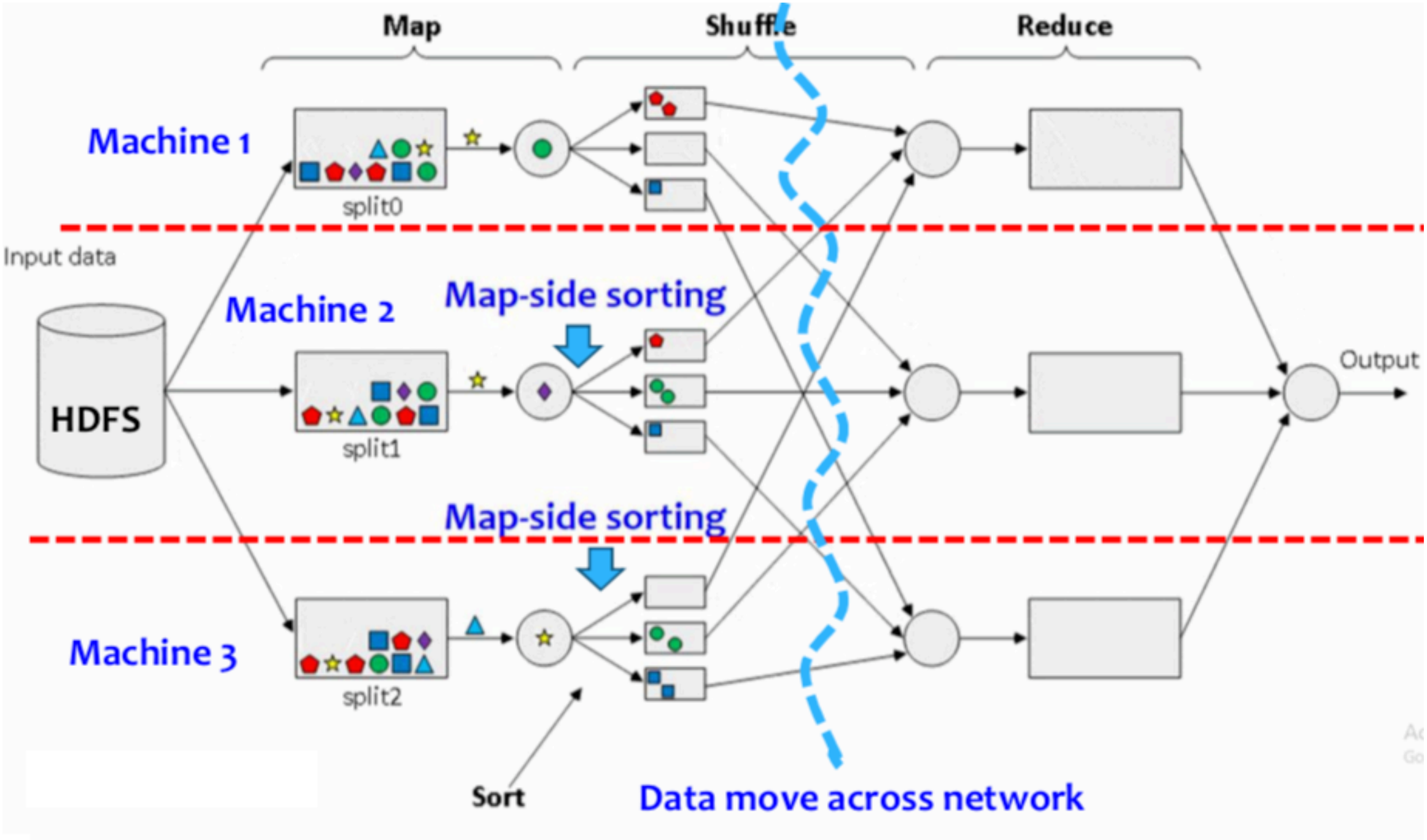
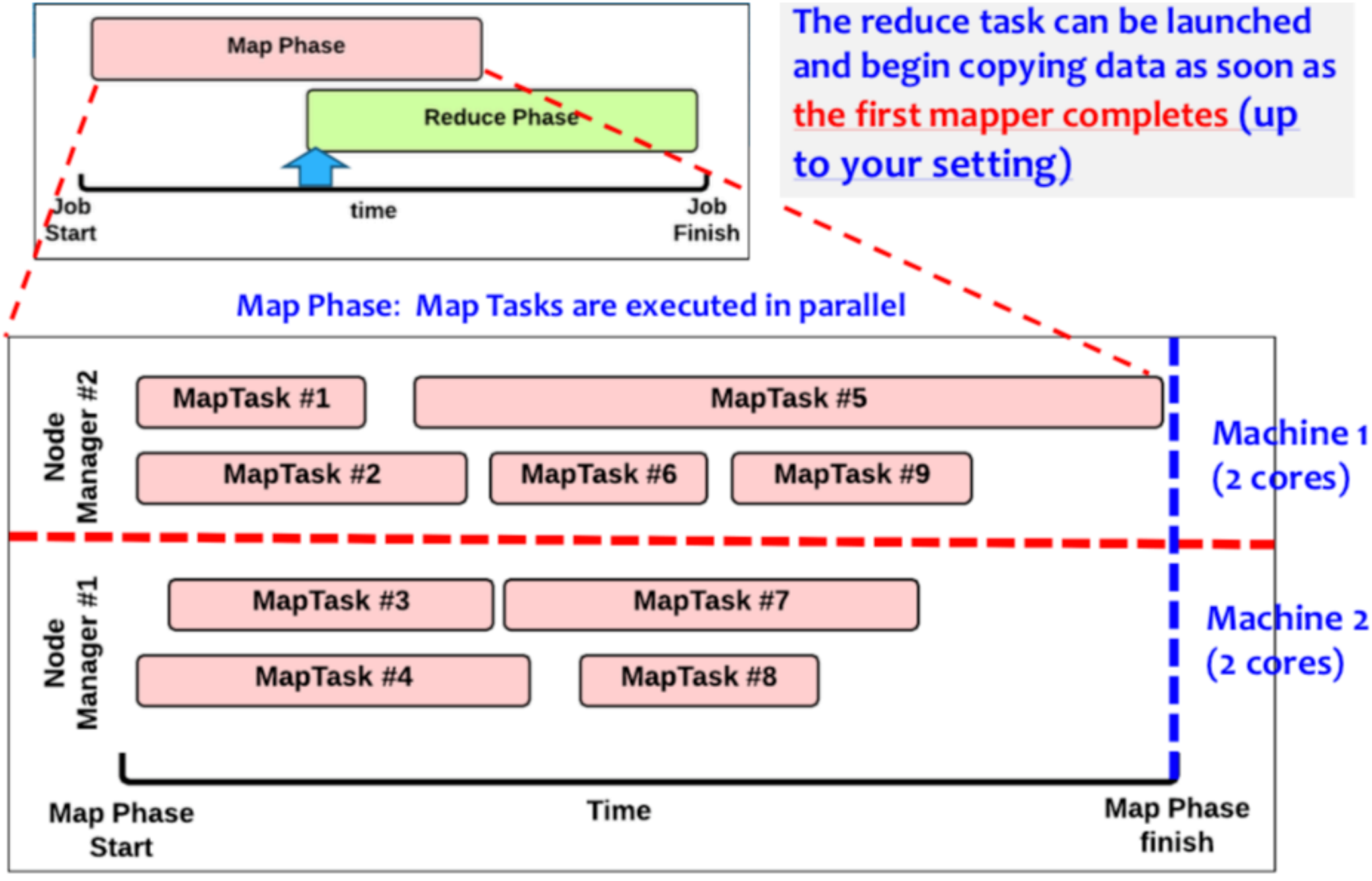

Map
Shuffle
Reduce
Machine 1
split0
Input data
Machine 2
Map-side sorting
HDFS
split1
Output
Map-side sorting
Machine 3
split2
Sort
Data move across network

Timeline


Map Phase
Reduce Phase
Job Start
time
Job Finish
The reduce task can be launched and begin copying data as soon as the first mapper completes (up to your setting)
Map Phase: Map Tasks are executed in parallel
Node Manager #2
MapTask #1
MapTask #5
MapTask #2
MapTask #6
MapTask #9
Machine 1 (2 cores)
Node Manager #1
MapTask #3
MapTask #7
MapTask #4
MapTask #8
Machine 2 (2 cores)
Map Phase Start
Time
Map Phase finish

of mappers and reducers

- NOTE 1: Usually no need to set the number of map tasks for the job. Default: 1 mapper for 1 block.
  - E.g., 10TB of input file and blocksize = 128MB -> 81,920 maps.
    - You can change the block size (32MB, 64MB, or 128MB) to get the desired number of mappers
- NOTE 2: the # of Reducers needs to be specified by the user.
- NOTE 3: In Yarn, actual number of map/reduce tasks to be executed concurrently depends on the amount of available memory !! Specifying the container sizes (mapreduce.{map/reduce}.memory.mb) become important !

Rule of thumb

- M map tasks, R reduce tasks:
  - Make M and R much larger than the number of nodes in cluster (e.g., M=200,000; R=4,000; workers=2,000) (M > R > # of workers)
  - One block (64MB) per map is common
  - Other suggestion: set the number of mappers and reducers to the number of cores available minus 1 for each machine.

Fault tolerance

- Case 1: If a task crashes:
  - Retry on another node.
    - OK for a map because it has no dependencies.
    - OK for reduce because map's outputs are saved on disk.
- If the same task fails repeatedly, fail the job or ignore that input block (user-controlled).

Fault tolerance

- Case 2: If a node crashes:
  - OK. Re-launch its current tasks on other nodes.
  - Re-run any maps the node previously ran.
    - Necessary because their output files (saved in disk) were lost along with the crashed node.

Fault tolerance

- Case 3. If a task is going slowly (“straggler”):
  - Launch second copy of task on another node (This is called “speculative execution”).
  - Take the output of whichever copy finishes first, and kill the other.

Fault tolerance

- Speculative execution is surprisingly important in large clusters (according to Google) !
  - Stragglers occur frequently due to failing hardware, software bugs, misconfiguration, etc.
  - Single straggler may noticeably slow down a job.
  - When MapReduce operation is close to finish, the scheduler schedules backup executions of the remaining in-progress tasks.

WordCount.java

```
1.  package org.myorg;
2.
3.  import java.io.IOException;
4.  import java.util.*;
5.
6.  import org.apache.hadoop.fs.Path;
7.  import org.apache.hadoop.conf.*;
8.  import org.apache.hadoop.io.*;
9.  import org.apache.hadoop.mapred.*;
10. import org.apache.hadoop.util.*;
11.
12. public class WordCount {
13.
14.    public static class Map extends MapReduceBase implements
    Mapper<LongWritable, Text, Text, IntWritable> {
15.      private final static IntWritable one = new IntWritable(1);
16.      private Text word = new Text();
17.
18.      public void map(LongWritable key, Text value,
    OutputCollector<Text, IntWritable> output, Reporter reporter) throws
    IOException {
19.        String line = value.toString();
20.        StringTokenizer tokenizer = new StringTokenizer(line);
21.        while (tokenizer.hasMoreTokens()) {
22.          word.set(tokenizer.nextToken());
23.          output.collect(word, one);
24.        }
25.      }
26.    }
```

The **Mapper** implementation (lines **14-26**), via the **map method** (lines **18-25**)

StringTokenizer splits the line into tokens separated by **whitespaces.**

the, 1
quick, 1
brown, 1
fox, 1

Line 23: emits a key-value pair of < <word>, 1>.

```
28.  public static class Reduce extends MapReduceBase implements
     Reducer<Text, IntWritable, Text, IntWritable> {
29.    public void reduce(Text key, Iterator<IntWritable> values,
     OutputCollector<Text, IntWritable> output, Reporter reporter) throws
     IOException {
30.      int sum = 0;
31.      while (values.hasNext()) {
32.       sum += values.next().get();
33.      }
34.      output.collect(key, new IntWritable(sum));
35.    }
36.  }
37.
38.  public static void main(String[] args) throws Exception {
39.    JobConf conf = new JobConf(WordCount.class);
40.    conf.setJobName("wordcount");
41.
42.    conf.setOutputKeyClass(Text.class);
43.    conf.setOutputValueClass(IntWritable.class);
44.
45.    conf.setMapperClass(Map.class);
46.    conf.setCombinerClass(Reduce.class);
47.    conf.setReducerClass(Reduce.class);
48.
49.    conf.setInputFormat(TextInputFormat.class);
50.    conf.setOutputFormat(TextOutputFormat.class);
51.
52.    FileInputFormat.setInputPaths(conf, new Path(args[0]));
53.    FileOutputFormat.setOutputPath(conf, new Path(args[1]));
54.
55.    JobClient.runJob(conf);
57.  }
58.}
```

Reduce()

sums up the values, which are the occurrence counts for each key

Combiner (line 46): local aggregation of the intermediate outputs. (Not explained in lecture)

TextInputFormat: each line in the text file is a *record*.

Input/output paths passed via command line.

Hadoop use cases

- Google: Main tasks: Index construction for Google Search, article clustering for Google News, Statistical machine translation,..

- Yahoo!:
  - Apache Hadoop project was initiated and led by Yahoo!.
  - 2008: the world's largest Hadoop on more than 10,000 core Linux cluster.
  - Main tasks: Yahoo! Search, Spam detection for Yahoo! Mail.

Hadoop use cases

- Facebook: Data mining, Ad optimization, Spam detection. (with Hive)
- LinkedIn:
  - User activity, server metrics, images, transaction logs stored in HDFS are used by data analysts for business analytics like discovering people whom you may know.
- eBay: Search Optimization and Research.
- Others: Walmart, China Mobile, Verizon ...

Example: Twitter's Hadoop infrastructure

- Twitter stores 1.5 petabytes of logical time series data, and handles 25K query requests per minute.

- A typical cluster can have more than 100,000 hard drives, translating into 100 petabytes of logical storage.

- HDFS for stored data and Yarn for temporary data.

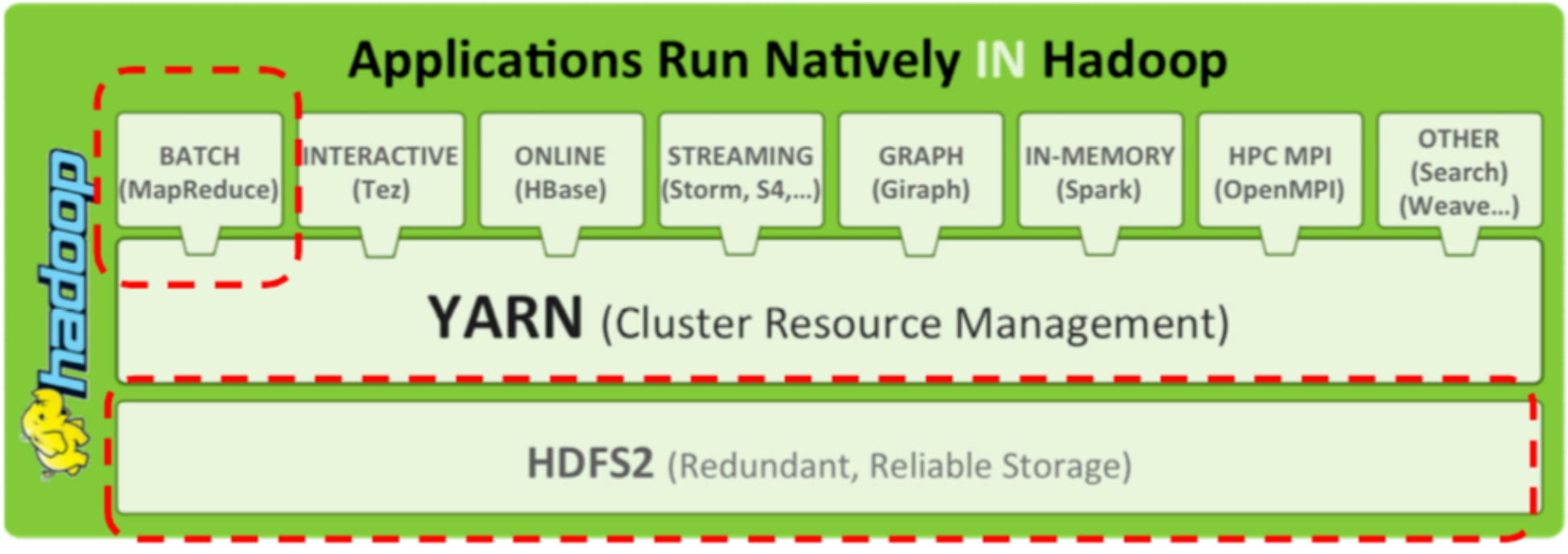
- Caches Yarn-managed temporary data to a fast SSD

- Moving 300PB of cold data storage to Google Cloud (2018)

YARN

Apache Hadoop >= 2.x

- Using YARN (“Yet Another Resource Negotiator”)
- YARN supports non-MapReduce workloads.
- Multiple ways to interact with the data in HDFS: MapReduce, Spark, Storm, Hbase, MPI, Hive and Tez.



In Hadoop 1.x, JobTracker keeps track of all jobs and their tasks. Its performance becomes the bottleneck.

The YARN scheduler

- Each server as a collection of containers
  Container : **fixed CPU + fixed memory** + (disk, network)

- 3 main components:

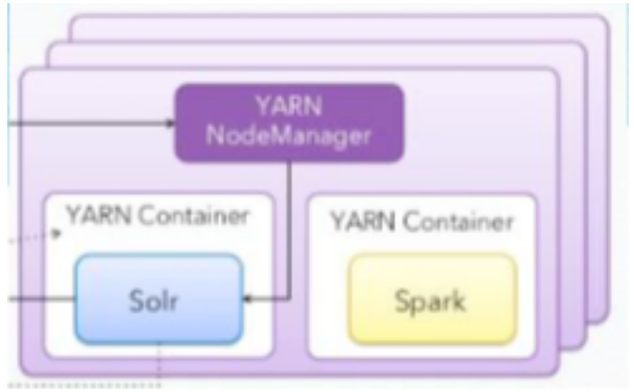
  - **Global Resource Manager (RM)**
    The ultimate authority that arbitrates resources among all the applications in the system

  - **Per-server Node Manager (NM)**
    The “worker” daemon in YARN; launch the applications’ containers, monitor resource usage and report to ResourceManager.

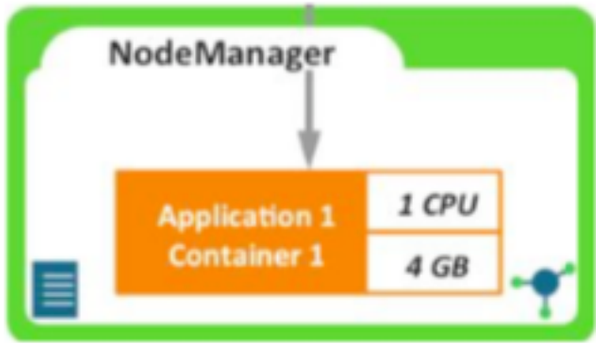
  - **Per-application Application Master (AM)**
    Negotiating resources from the RM and working with the NodeManager(s) to execute and monitor the tasks.

What is a Container in YARN?
(NOTE: Yarn “Container” is different from Docker “Container”)

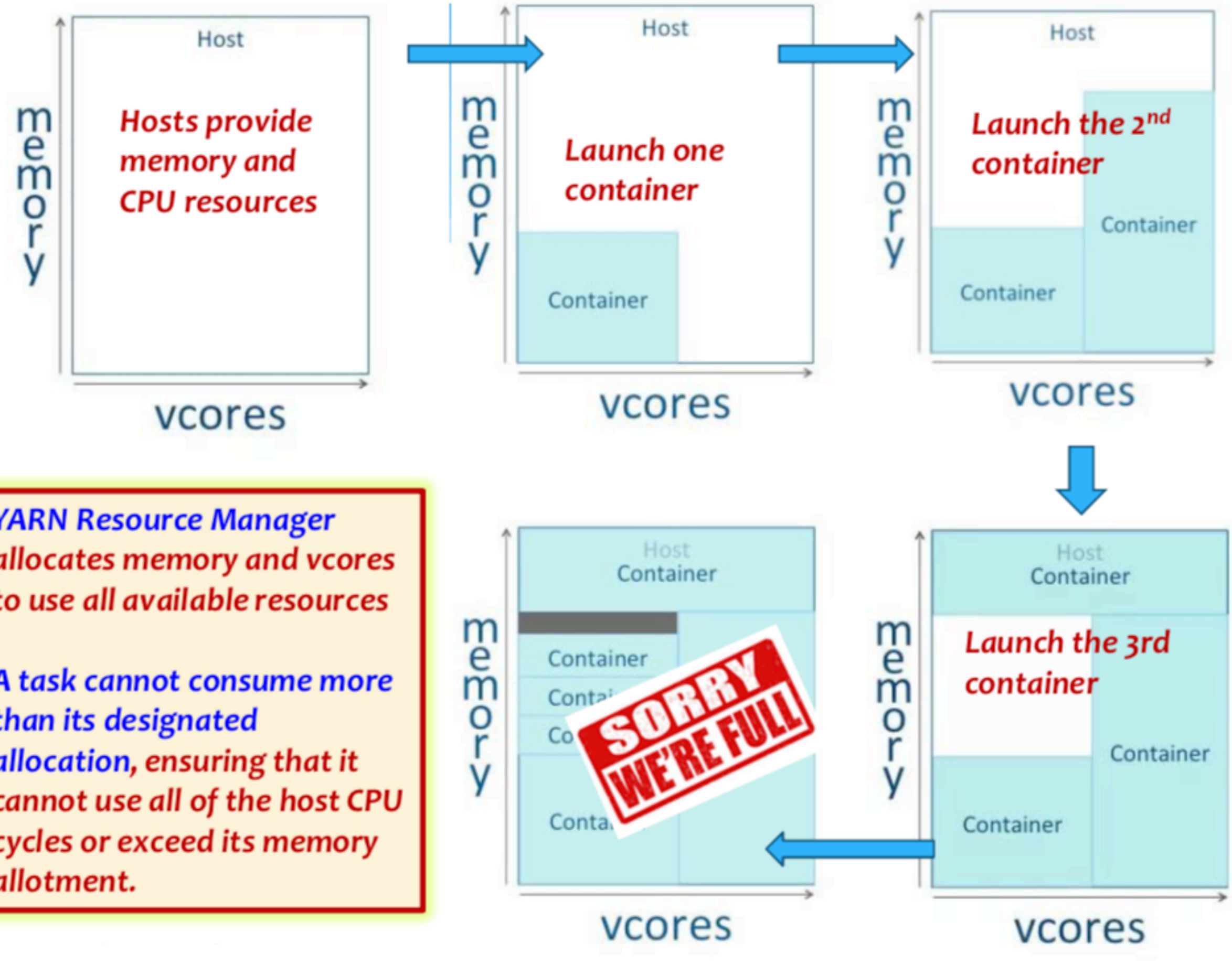
- “A Yarn container is a JVM process”.
  * Launch and monitor by the Node manager
  * Scheduled by the Resource manager



- Container is a place where a YARN task is run.
  - tasks that can run inside a container: Map/Reduce tasks, Spark task, HBase, Hive, MPI, ...
  - Note: Container location is determined by the resource manager.



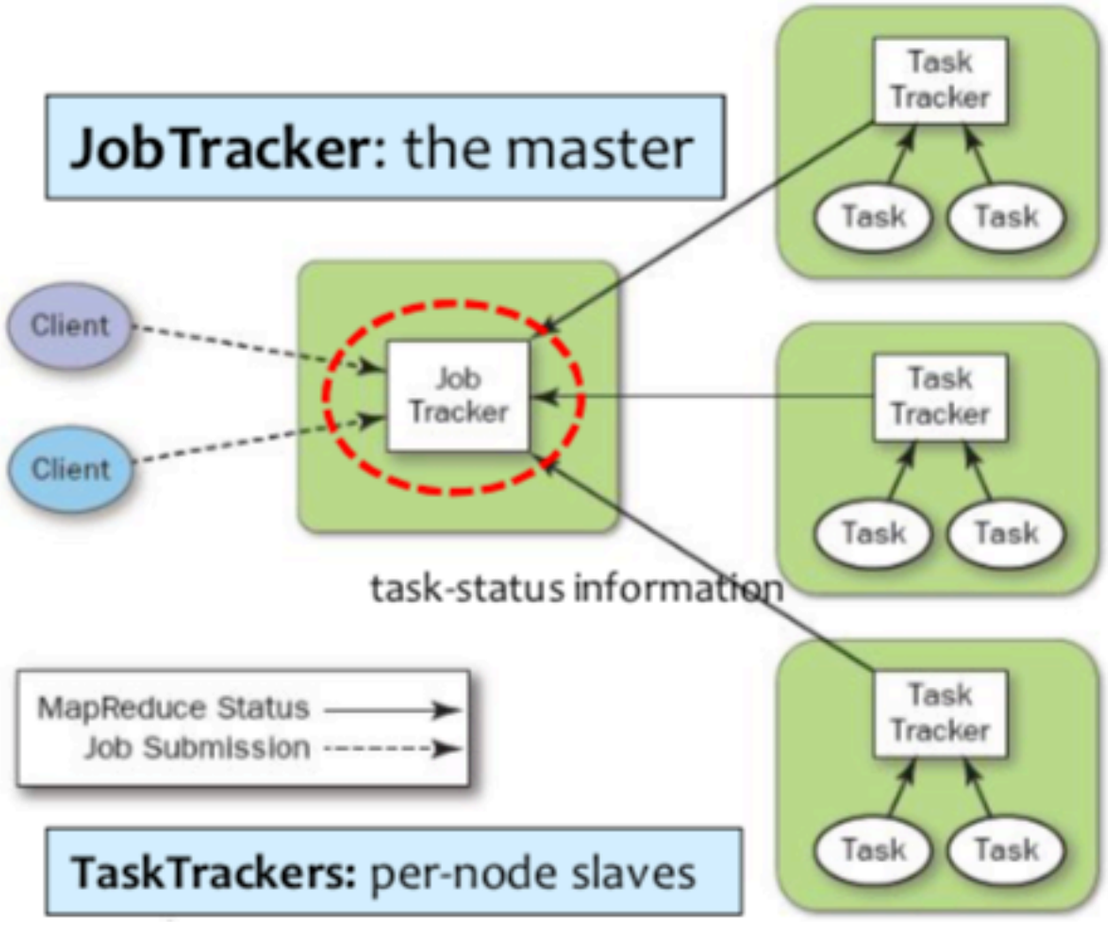
- Each container has a unique Container Id and specific amount of Resource (vcore, RAM) allocated.

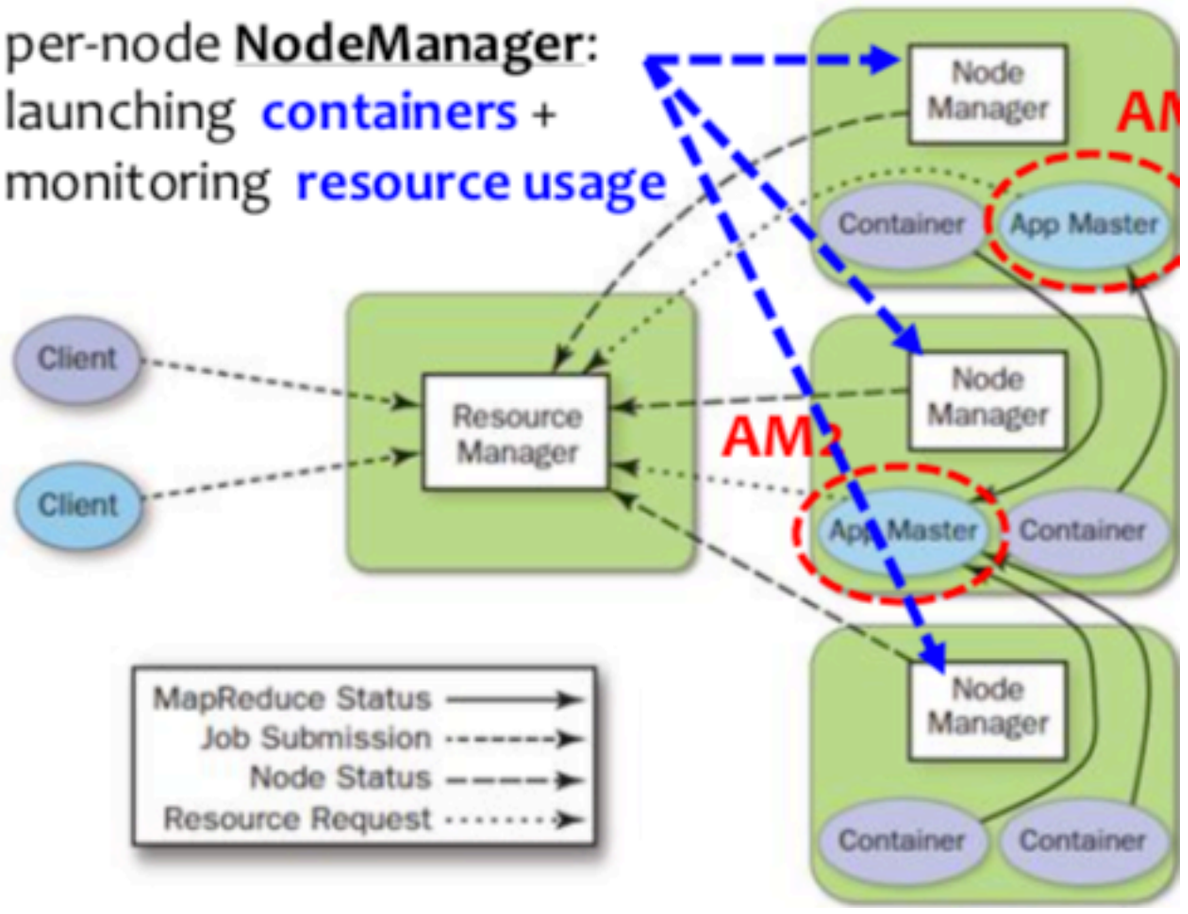

Host
memory
Hosts provide memory and CPU resources
vcores
Host
memory
Launch one container
Container
vcores
Host
memory
Launch the 2nd container
Container
Container
vcores
YARN Resource Manager allocates memory and vcores to use all available resources
A task cannot consume more than its designated allocation, ensuring that it cannot use all of the host CPU cycles or exceed its memory allotment.
Host
Container
memory
Container
SORRY WE'RE FULL
vcores
Host
Container
memory
Launch the 3rd container
Container
Container
vcores

Hadoop 1.x (Old) : JobTracker does
(1) Resource management
(2) Job scheduling/monitoring.

Hadoop 2.x YARN

JobTracker is split up into :
(1) a global **ResourceManager** (**RM**) and
(2) per-application **ApplicationMaster** (AM).





Application Master (AM)

- Application Master is responsible for running ONE single application (e.g., a WordCount).
  - Note: AM itself runs on a specific container
- Application Master is aware of execution logic
  - MapReduce, MPI, Hive, Spark
- AM creates one map task per split and a number of reduce tasks determined by the mapreduce.job.reduces configuration property
- AM tells NodeManagers to start containers on its behalf.

Control flow

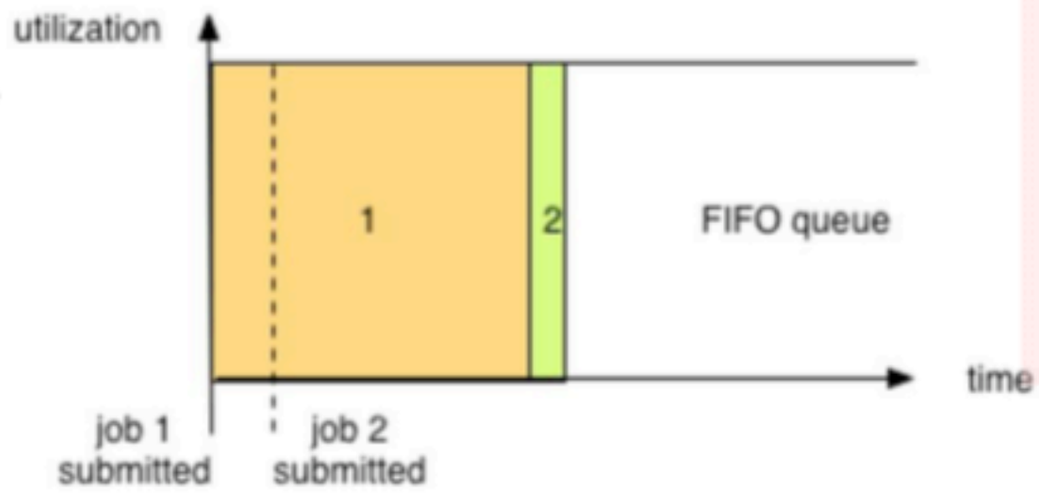
- Resource Request: AM, RM
  - applications specify the resource needed. This lets the RM know about their resource requirements.
- Granting a container : RM, AM
  - The Scheduler responds to a resource request by granting a container, which satisfies the requirements laid out by the AM in the initial ResourceRequest.
- Container-launch request: AM, NM
- NM launches the container: NM, Container
  - To execute the actual map or reduce task, YARN will run a JVM within the container.

YARN schedulers

- (1) FIFO Scheduler: first in first out
- (2) Capacity Scheduler (Default)
  - Queues are allocated a fraction of the capacity.
    - All applications submitted to a queue will have access to the capacity allocated to the queue.
- (3) Fair Scheduler
  - Assigning resources to applications such that all applications (jobs) get, on average, an equal share of resources over time.
  - Advantage: short applications finish in reasonable time while not starving long-lived applications

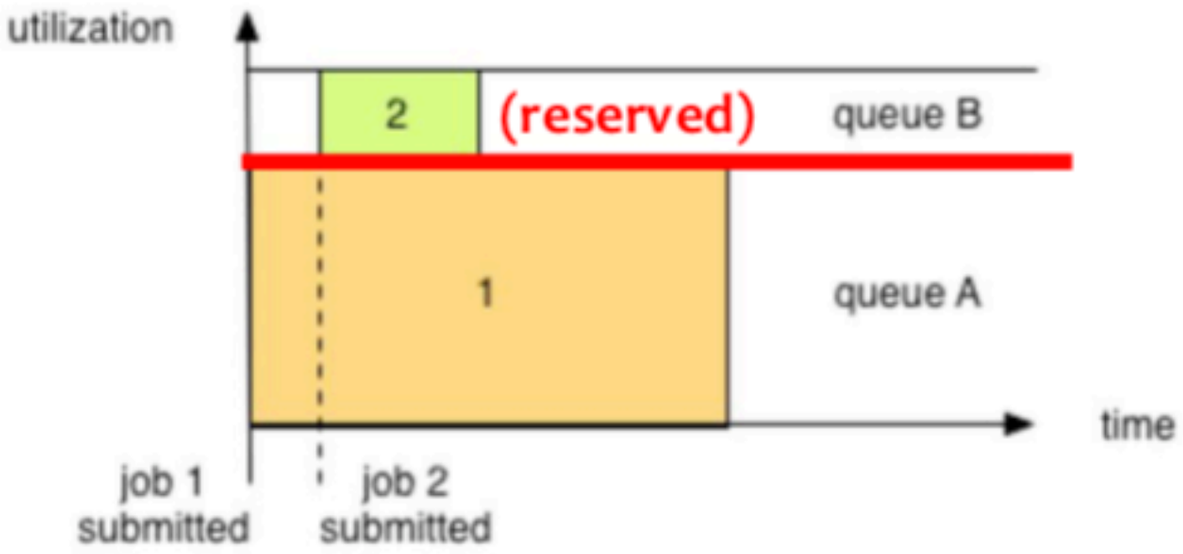
***FIFO**: places applications in a queue and runs them in the order of submission.*

FIFO Scheduler:



BAD: the small job is blocked until the large job (e.g., Job 1) completes.

Capacity Scheduler (Default)



Fair Scheduler

Next: Spark