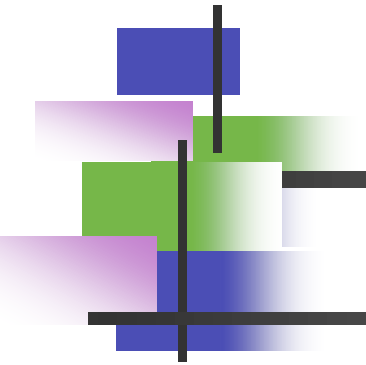


11. MapReduce and Hadoop

Parallel Programming

Dr Hamidreza Khaleghzadeh
School of Computing
University of Portsmouth

Goals

- Review the MapReduce framework for distributed processing of large datasets.
- Introduce Apache Hadoop project – originally conceived as an open-source implementation of MapReduce.

What is MapReduce?

- *MapReduce* is a programming framework for distributed (parallel) processing on very large datasets.
- Claimed benefits include:
  - *Simple "functional" programming model* based on just two user defined functions: map and reduce.
  - *Highly scalable*, allowing to exploit large clusters of nodes to process datasets concurrently, and
  - *Tolerant of failure* of some hosts of cluster – computations that crash due to failure of some nodes will be restarted on other nodes.

Timeline

- Introduced in Google around 2003.
- Paper *MapReduce: Simplified Data Processing on Large Clusters*, Jeffrey Dean, Sanjay Ghemawat published 2004.
  - Used internally by Google around then for many tasks, including a rewrite of the indexing code used by their search engine.
- Open source Java based implementation of MapReduce released as *Apache Hadoop* in 2006
  - (Google implementation based on C++).

Conceptual Model

- User defines map and reduce functions
  - These operate on *key-value pairs*.
- A "functional" view of their specification:

  map (k1, v1) -> list(k2, v2)

  reduce (k2, list(v2)) -> list(k3, v3)

  - Here k1, k2 and k3 are "key types" (e.g. string or integer) and v1, v2, and v3 are "value types" (could be almost anything but commonly strings).
  - Thus, only required matching between types is that key and value types emitted by map corresponded to those input by reduce.

Implementation

- In practice the map function is usually a void function taking k1 and v1 key and value arguments, that internally calls some library method (zero or more times) to *emit* zero or more k2, v2 key-value pairs.
- Likewise reduce is a void function that internally emits zero or more v3 results
  - The list(v2) argument of reduce is likely to be implemented as an *iterator* over v2, so long lists don't have to reside in memory.

Operation of MapReduce

Input documents

to be or not to be

to die to sleep no more and by a sleep to say

to die to sleep to sleep perchance to dream

Key-value pairs from map

(to, 1) (be, 1) (or, 1)
(not, 1) (to, 1) (be, 1)

(to, 1) (die, 1) (to, 1)
(sleep, 1) (no, 1) (more, 1)
(and, 1) (by, 1) (a, 1)
(sleep, 1) (to, 1) (say, 1)

(to, 1) (die, 1) (to, 1)
(sleep, 1) (to, 1) (sleep, 1)
(perchance, 1) (to, 1) (dream, 1)

Outputs from reduce

(a, 1)
(and, 1)
(be, 2)
(by, 1)
(die, 2)
(dream, 1)
(more, 1)
(no, 1)
(not, 1)
(or, 1)
(perchance, 1)
(say, 1)
(sleep, 4)
(to, 9)

Word Count Example

- Adapted from original paper, using Java-like pseudocode:

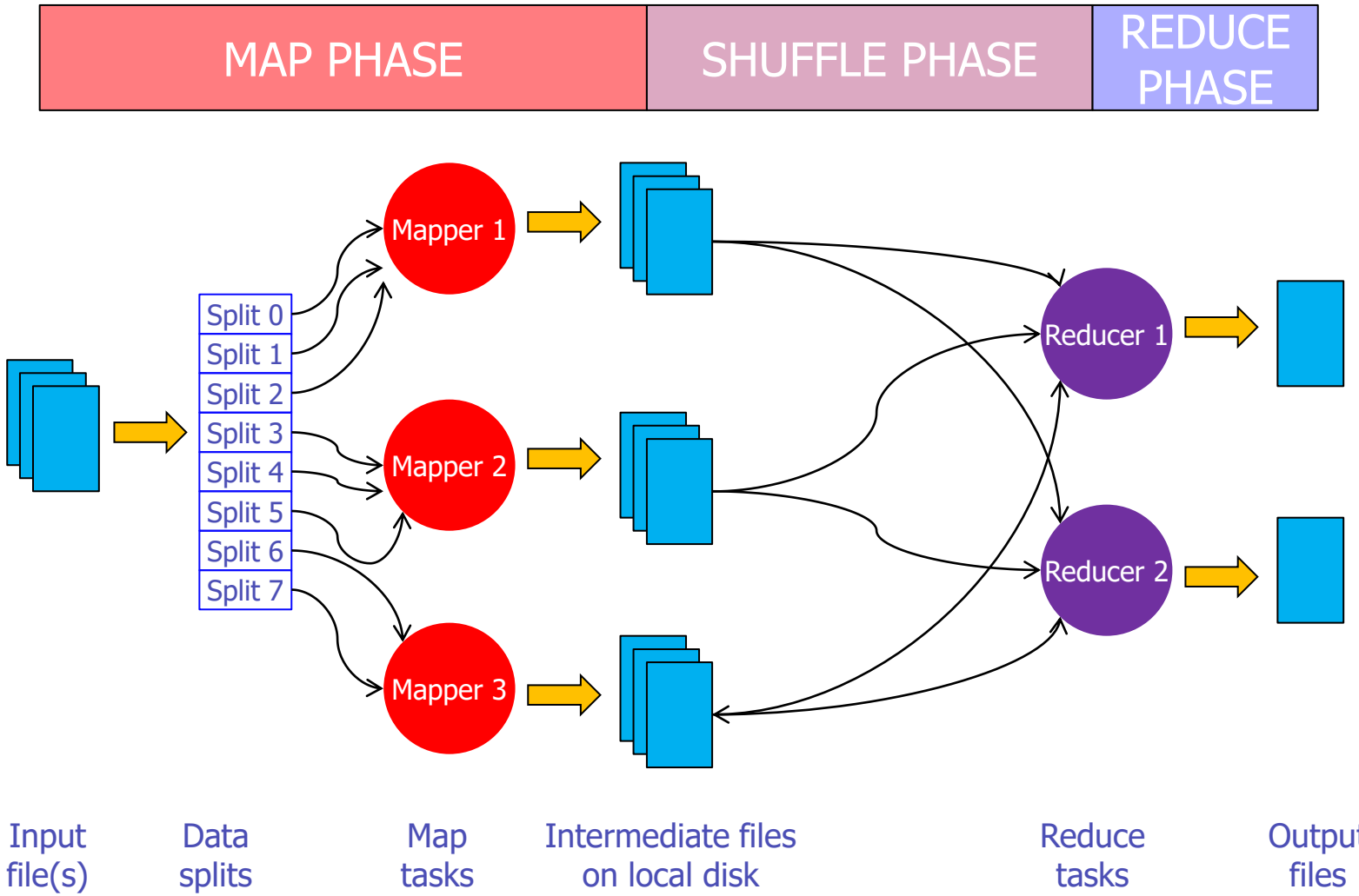
```
void map(String key, String value) {
  // key: document name
  // value: document contents
  for each word w in value
    emit(w, 1) ;
}

void reduce(String key, Iterator<int> values) {
  // key: word
  // values: list of counts emitted above
  int count = 0 ;
  for each v in values
    count += v ;
  emit(key, count) ;
}
```

Execution

- Typically executed as a distributed job in which:
  - Many hosts concurrently process many map functions on parts of some initial dataset (in word count example, some very large number of documents), and
  - Some (often smaller number of) hosts do reduce on outputs from map hosts.
    - Infrastructure ensures all v2 values produced by *all* mappers with the *same value of the k2 key* are passed to the *same reduce invocation*.
    - Implies distributed sorting on keys, called the "shuffle" phase, as data is moved from map hosts to reduce hosts.

MapReduce Pipeline



Notes

- Split sizes generally determined by characteristics of distributed file system that input/output data resides on
  - e.g. in Hadoop likely to be HDFS block size – 128MB by default.
- Shuffle phase implements a distributed sort – output files ordered by intermediate key values.
  - Reducers process distinct subranges of these keys.
- Generally one output file per reducer, partitioned by intermediate keys.

Other "Simple" Applications

- *Inverted index*: Map parses set of docs and emits (word, doc ID) pairs. Reduce emits (word, list(doc ID)), with docs sorted by some criterion.
  - Think: Google search for term over Web pages.
- *Sort*: Map extracts key from input records and emits (key, record) pairs. Reduce function emits pairs unchanged.
  - e.g in 2013 Hadoop won the Gray Daytona category of sort (100TB data) at sortbenchmark.org

APACHE HADOOP

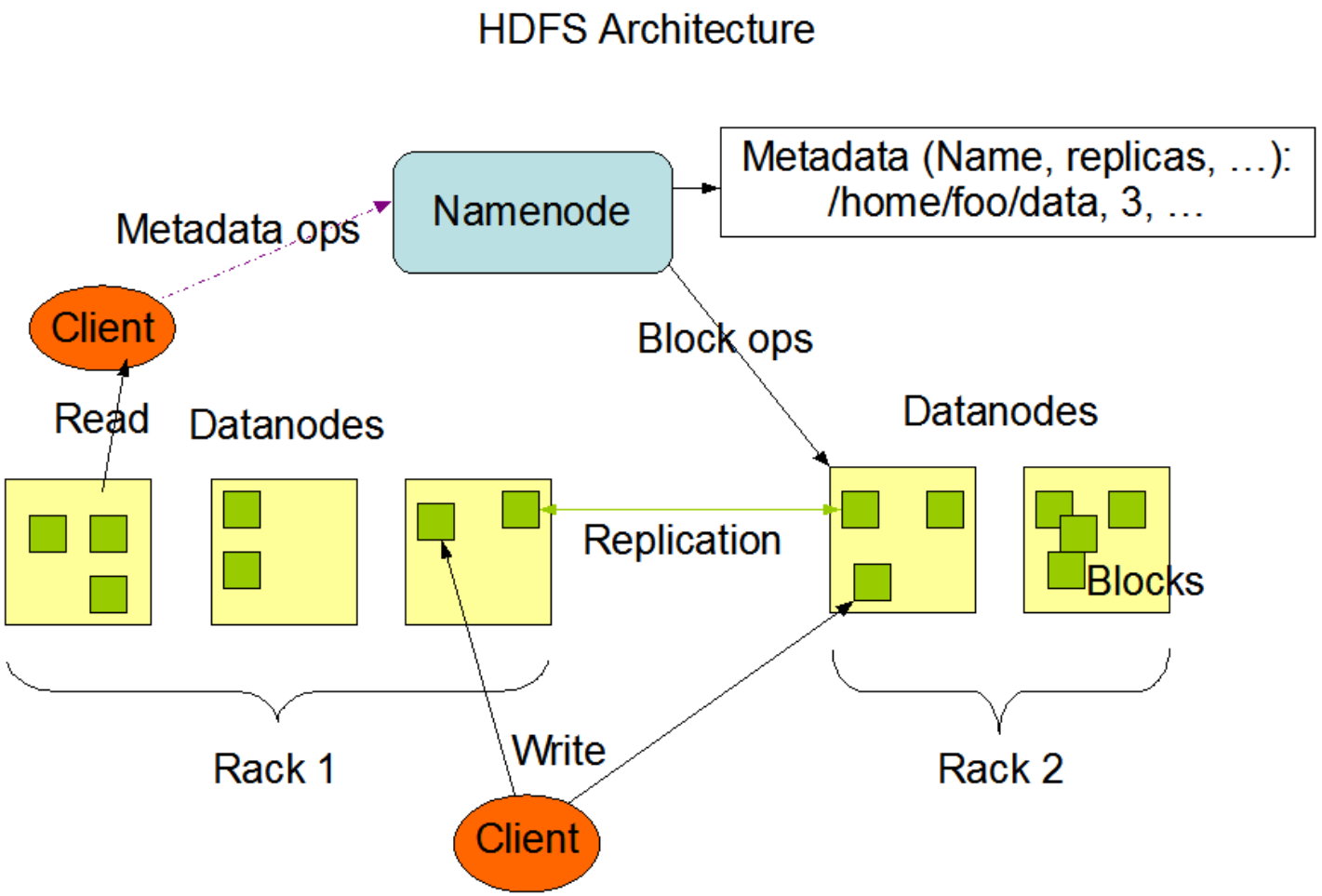
Hadoop

- Open source project inspired by early papers from Google on the *Google File System* and *MapReduce*.
- First public release in 2006.
- Implemented in *Java*.
- Much of its development has been led by programmers from *Yahoo! Inc.*
  - *Hadoop* was reportedly the name of Doug Cutting's son's toy elephant.

Default Components of Hadoop

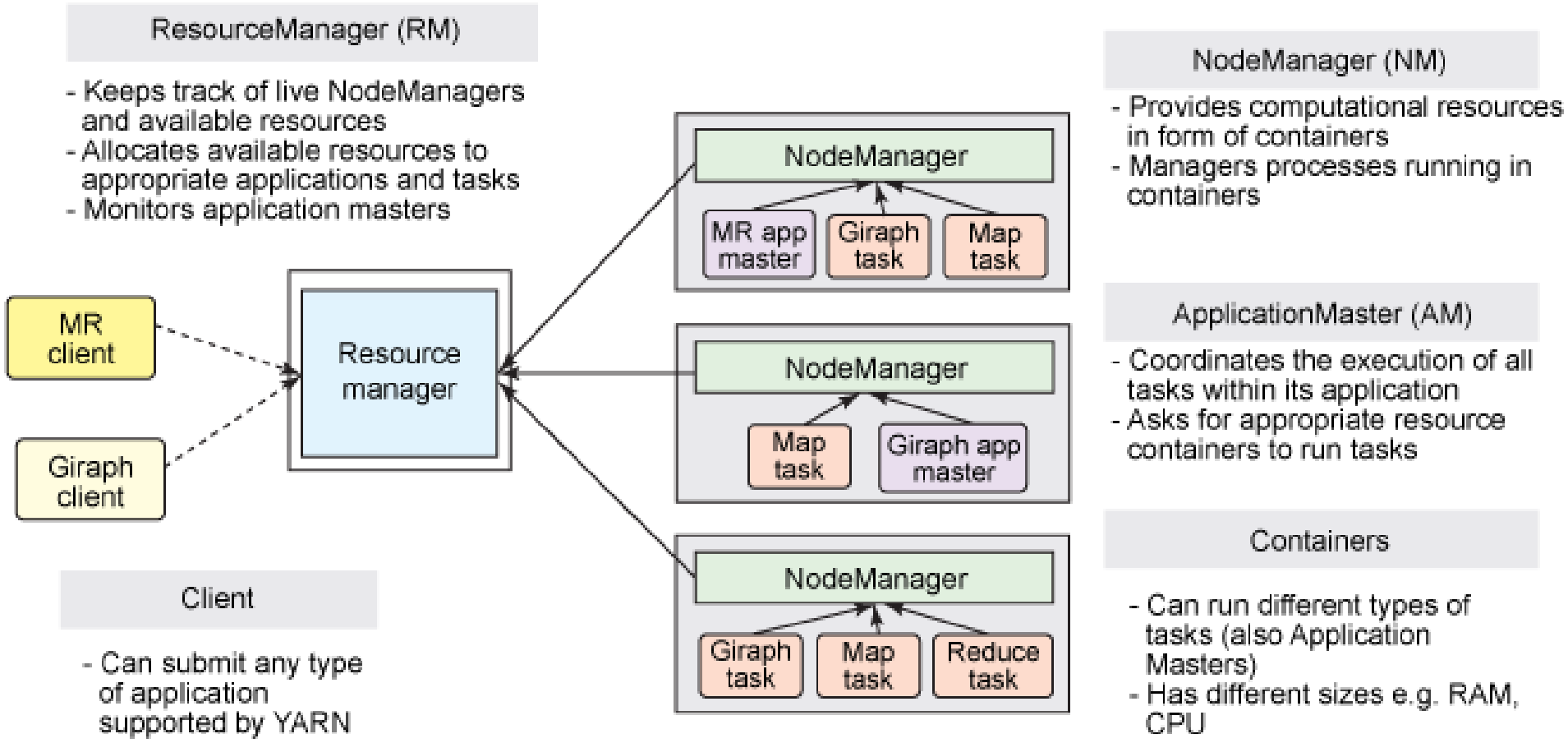
- The *Hadoop Distributed File System (HDFS)* is a reliable file system designed to span large clusters of commodity servers.
  - Files – usually very large - are broken into blocks stored in replicated fashion across many servers.
- The *YARN* resource manager, responsible for arbitrating requests for cluster resources from various applications (introduced in *Hadoop 2.0*).
- The *MapReduce* computational engine.
  - Since Hadoop 2.0 many other computational frameworks can be deployed in the same Hadoop cluster.

HDFS Architecture†



†Figure taken from hadoop.apache.org.

Architecture of YARN†



†Figure taken from www.ibm.com/developerworks/library/bd-yarn-intro.

Summary

- *MapReduce* was introduced in early 2000s as a distributed, parallel computation framework for processing Big Data.
- *Hadoop* open source implementation has been very widely adopted.
- MapReduce itself appears to have declined in popularity over the last few years, but, in the same time frame, Hadoop has come to support more general processing models (e.g. Apache Spark, Apache Giraph, etc).

Further Reading

- Ghemawat et al, *The Google File System*, 19th ACM Symposium on Operating Systems Principles, 2003.
- Dean and Ghemawat, *MapReduce: Simplified Data Processing on Large Clusters*, OSDI '04, 2004.
- White, *Hadoop: The Definitive Guide*, O'Reilly.
- Murthy et al, *Apache Hadoop YARN: Moving beyond MapReduce and Batch Processing with Apache Hadoop 2*, Addison Wesley.