Spark on YARN: a deep dive

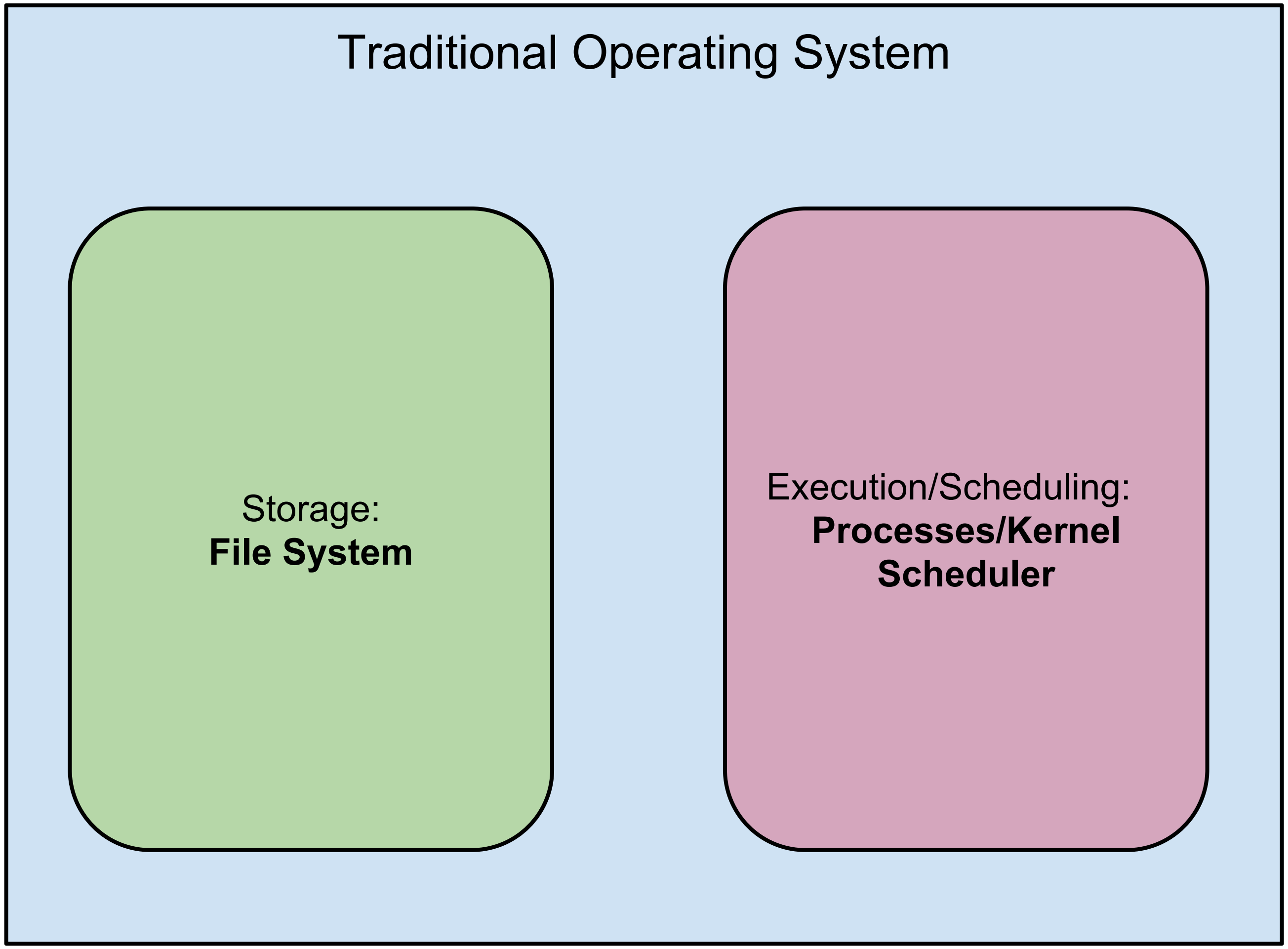
Sandy Ryza

Me

- Data scientist at Cloudera
- Recently lead Apache Spark development at Cloudera
- Before that, committing on Apache YARN and MapReduce
- Hadoop PMC member

What is YARN?

The OS analogy

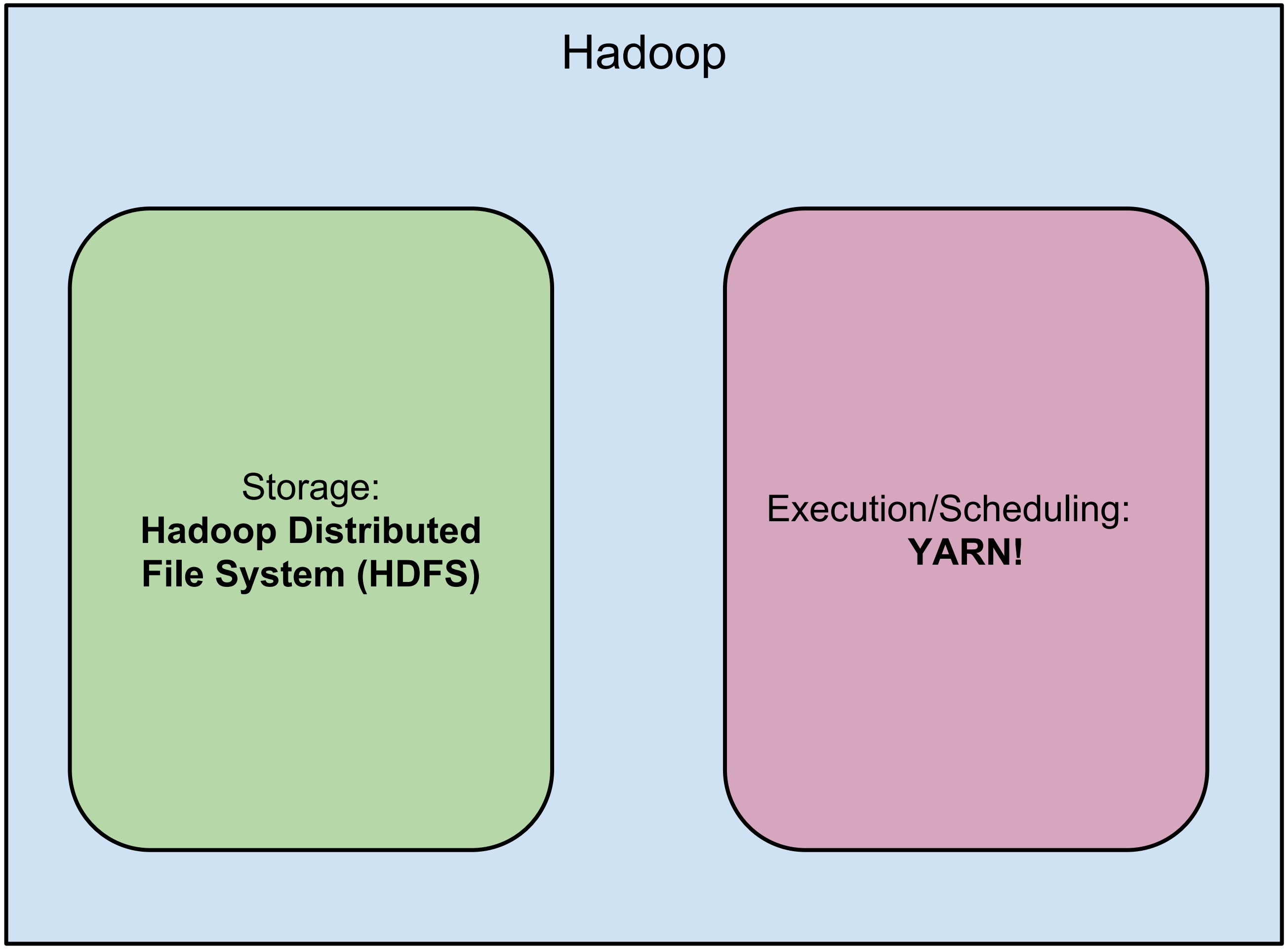


Traditional Operating System

Storage:
File System

Execution/Scheduling:
Processes/Kernel Scheduler

The OS analogy



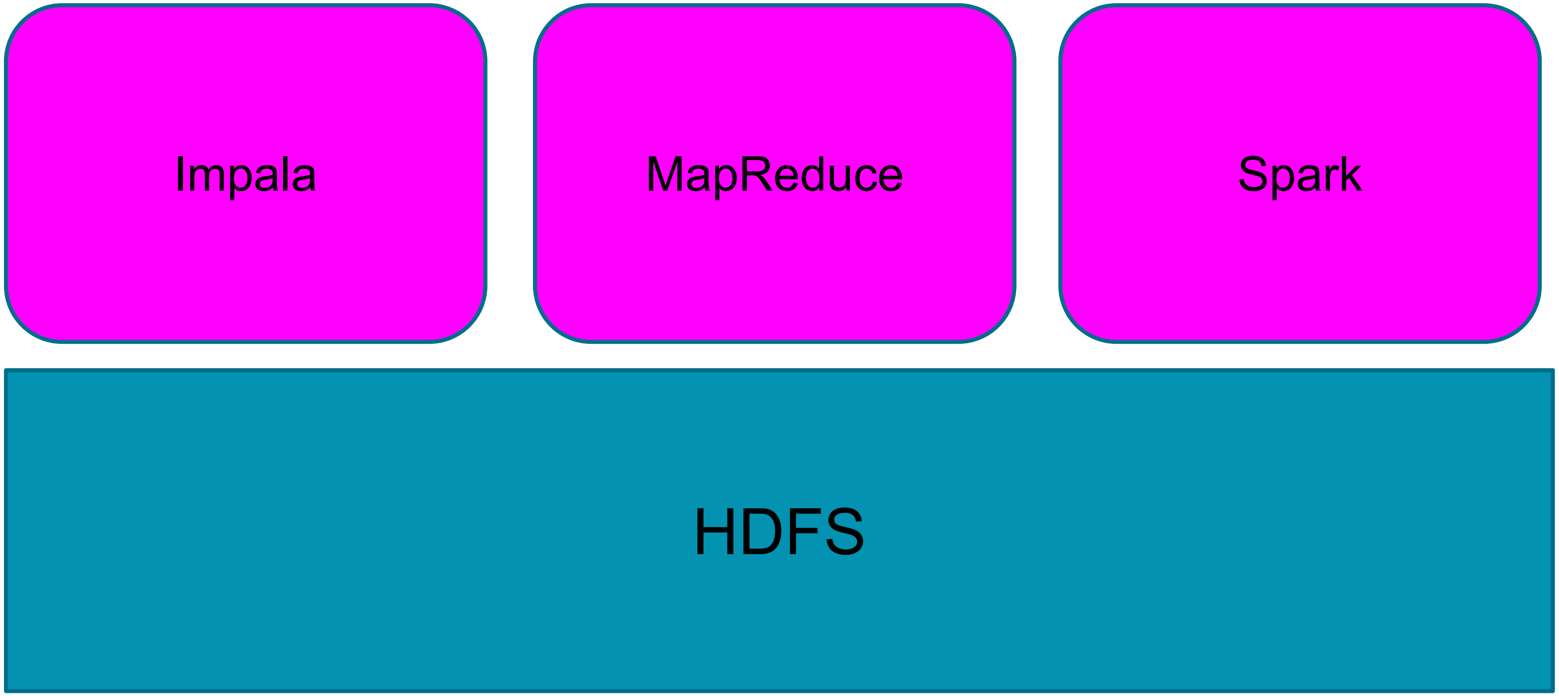
Hadoop

Storage:
Hadoop Distributed File System (HDFS)

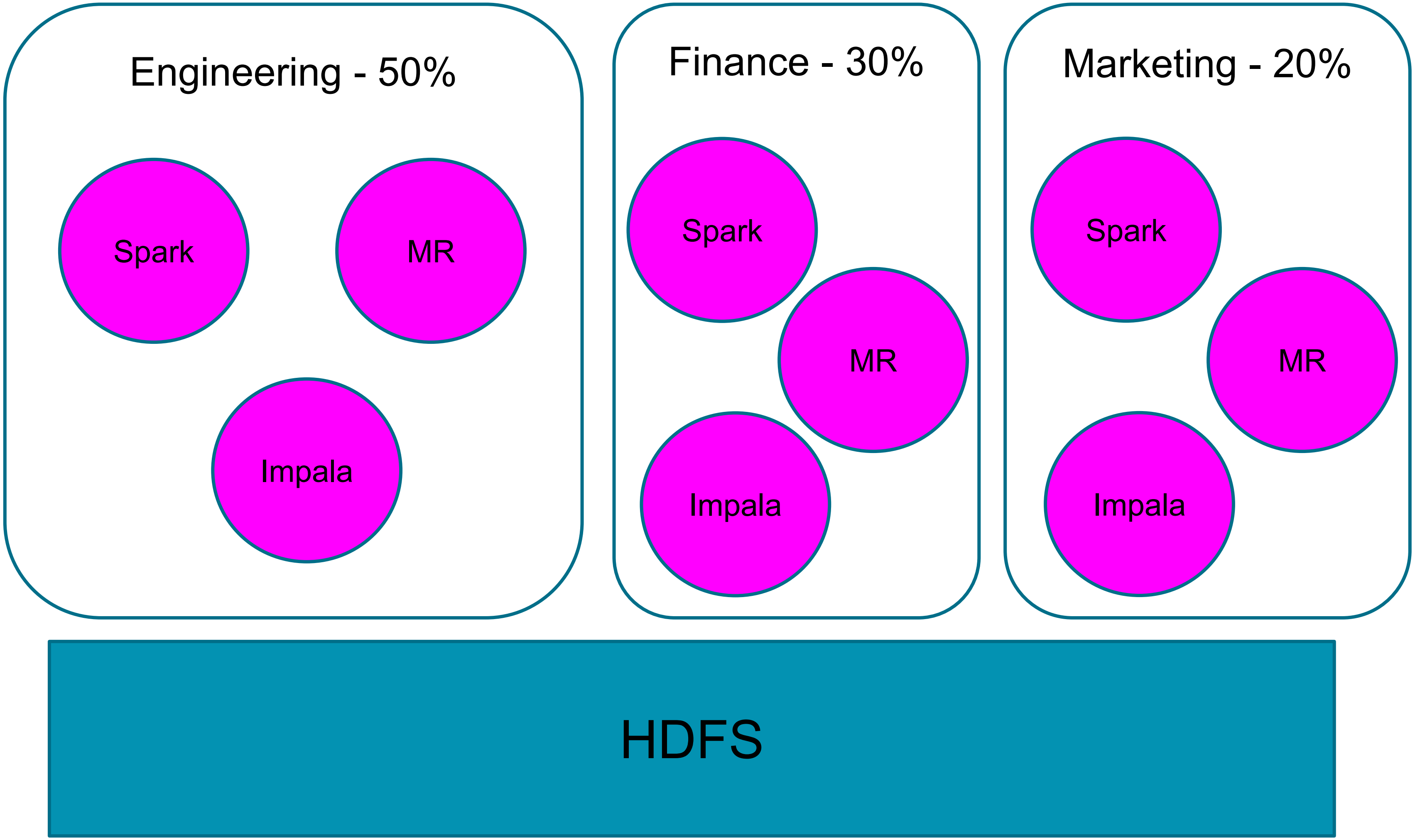
Execution/Scheduling:
YARN!

Why run Spark on YARN?

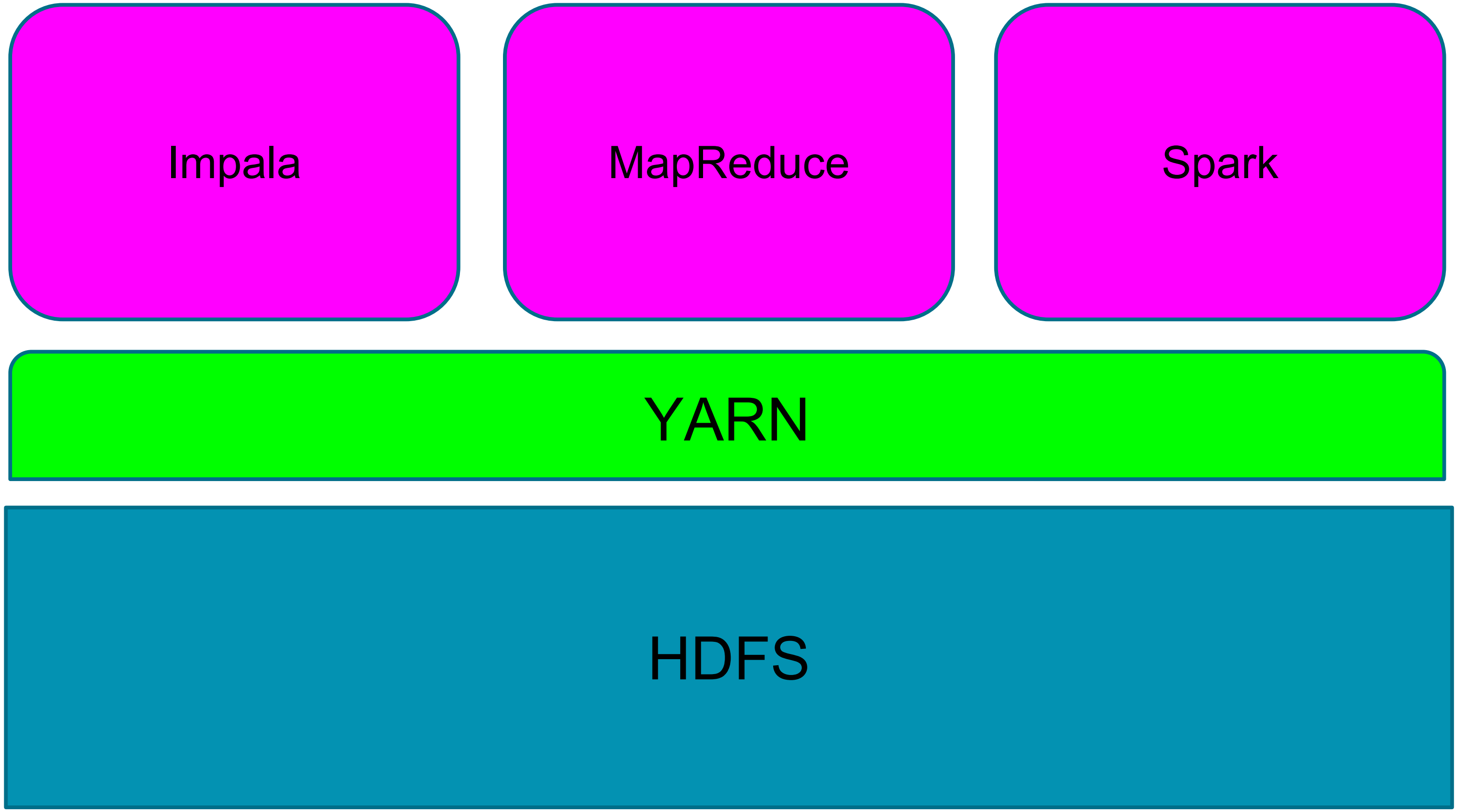
How We See It



How They Want to See It



Central Resource Management



Why run Spark on YARN?

- Run Spark alongside other Hadoop workloads
  - Leverage existing clusters
  - Data locality

Why run Spark on YARN?

- Manage workloads using advanced policies
    - Allocate shares to different teams and users
    - Hierarchical queues
    - Queue placement policies

Why run Spark on YARN?

- Take advantage of Hadoop's security
    - Run on Kerberized clusters

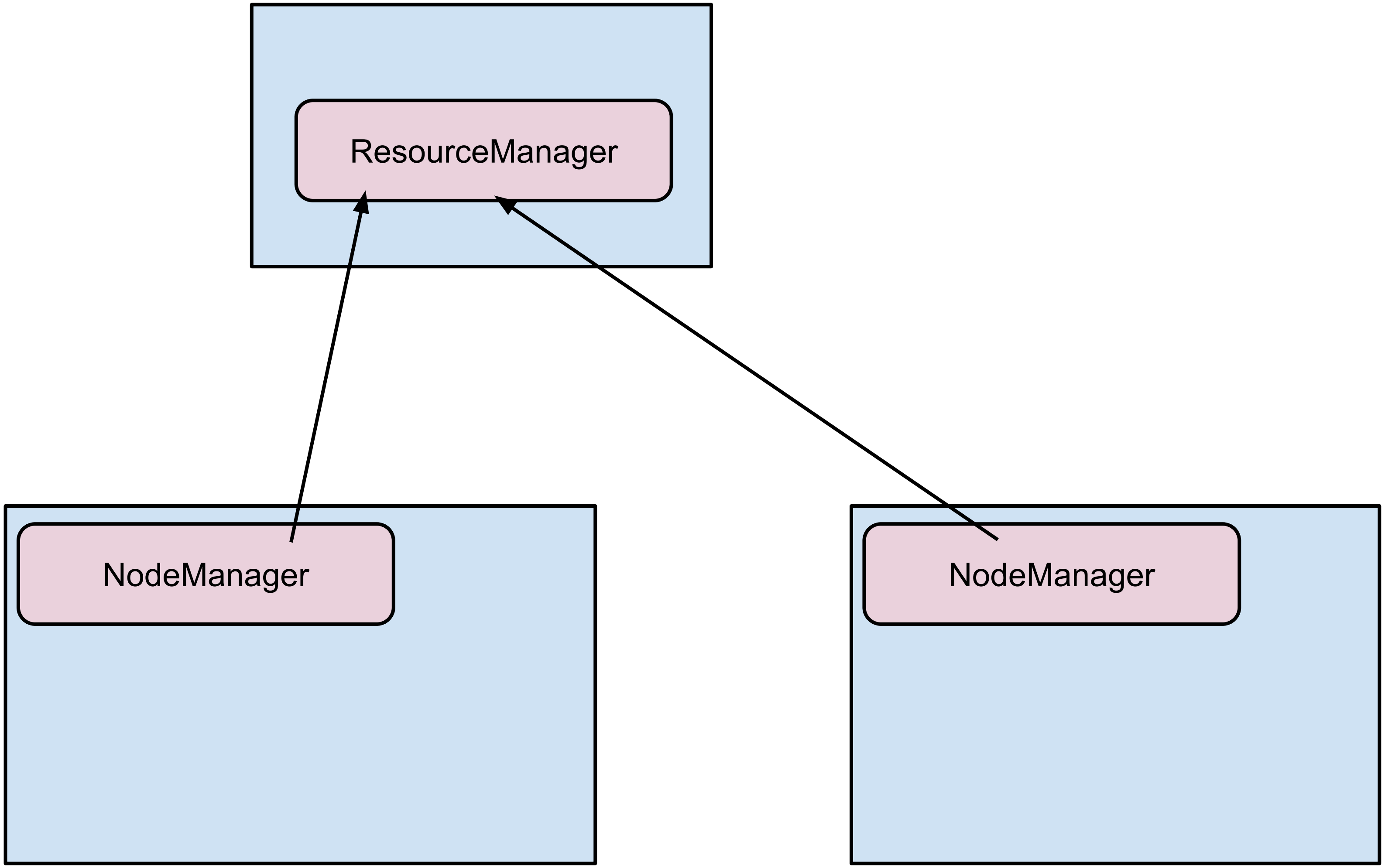
History

History

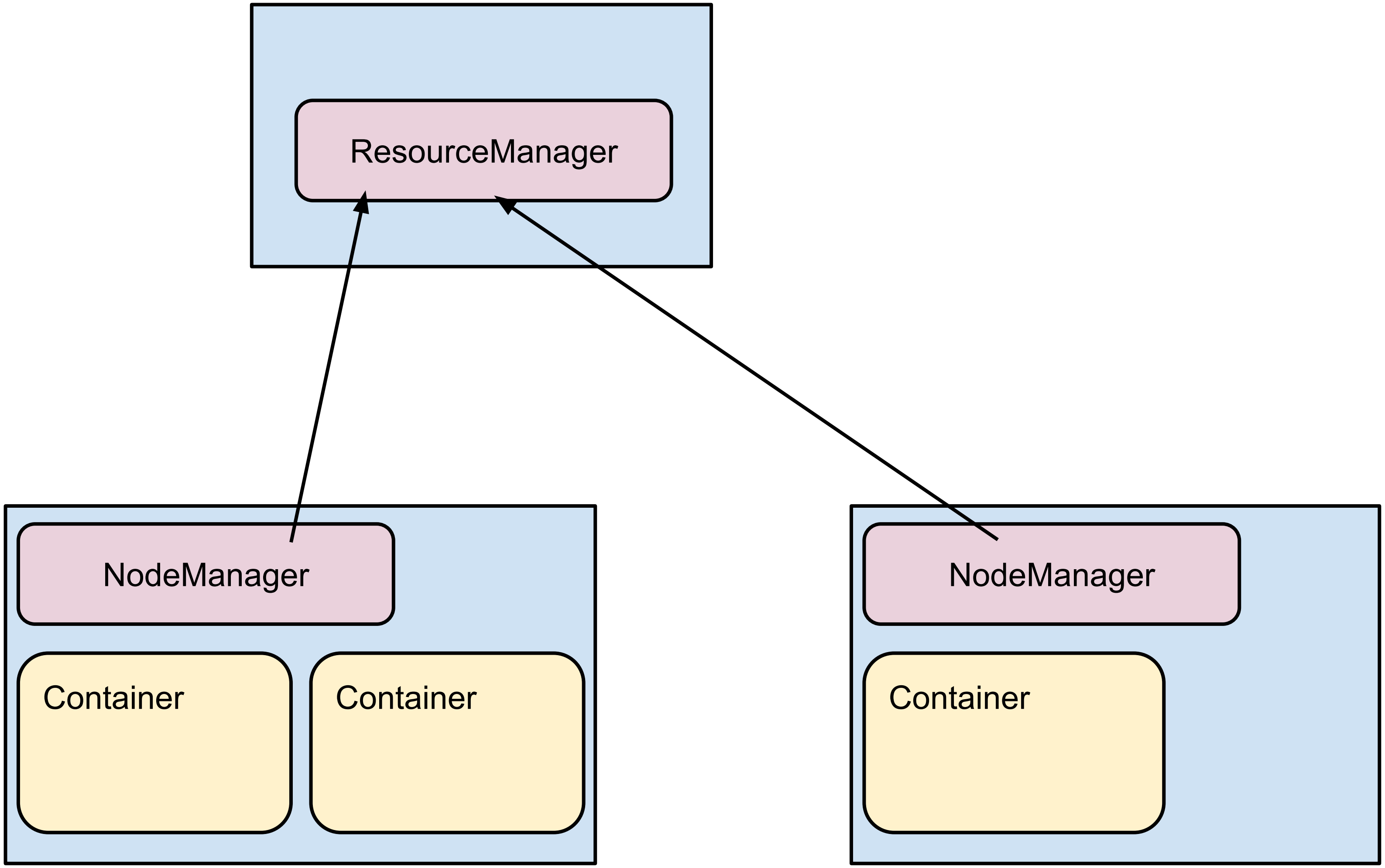
- **Late 2012 / Spark 0.6** - experimental project at Yahoo
- **Late 2013 / Spark 0.8** - pulled into Spark, Hadoop-0.23 only
- **Early 2014 / Spark 0.9** - Hadoop 2.2 line as well, support for `spark-shell`
- **Early 2014 / Spark 0.9.1 / CDH 5.0** - Stable!
- **Mid 2014 / Spark 1.0.0 / CDH 5.1** - Easier app submission with `spark-submit`

How’s it work?

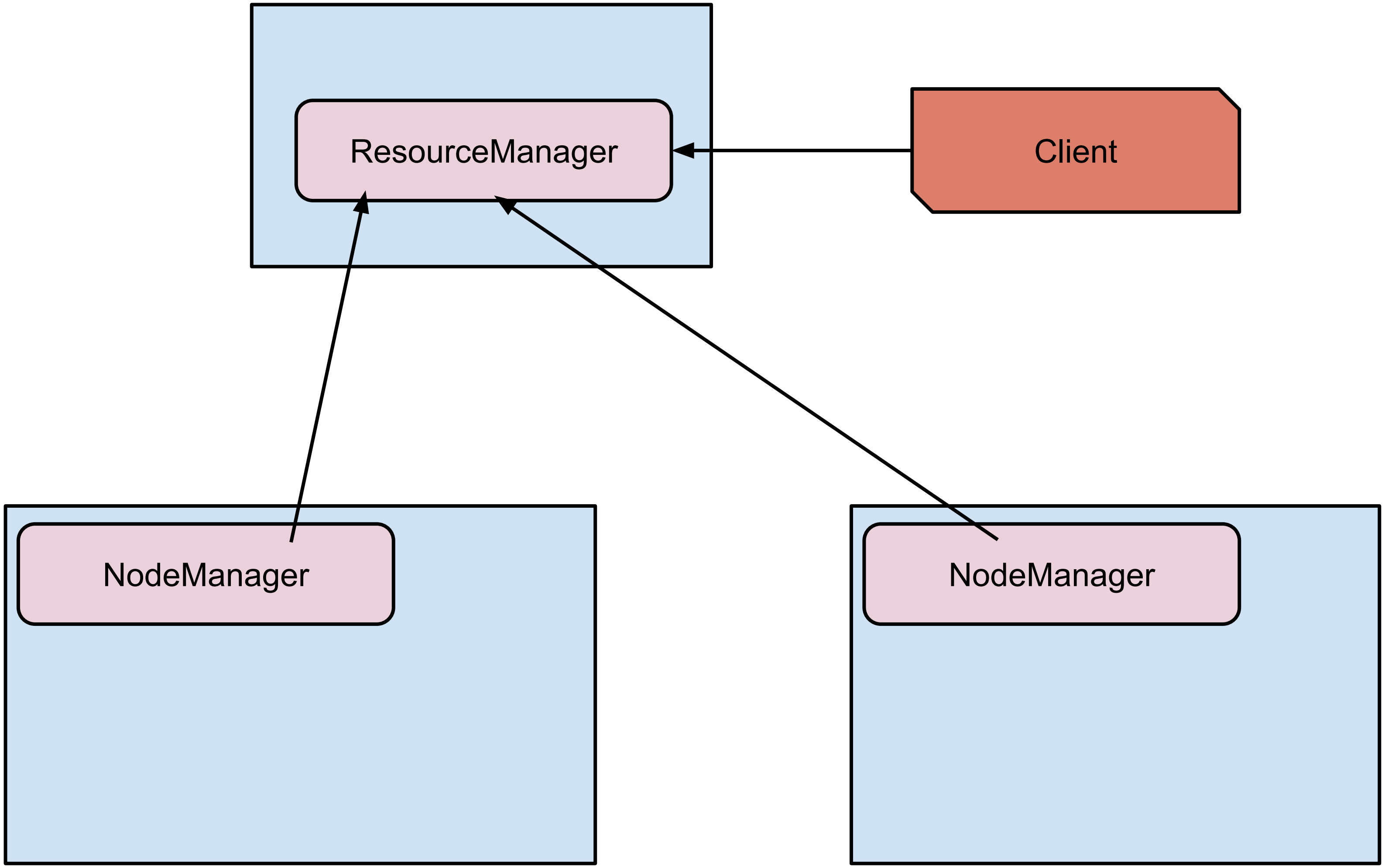
YARN architecture



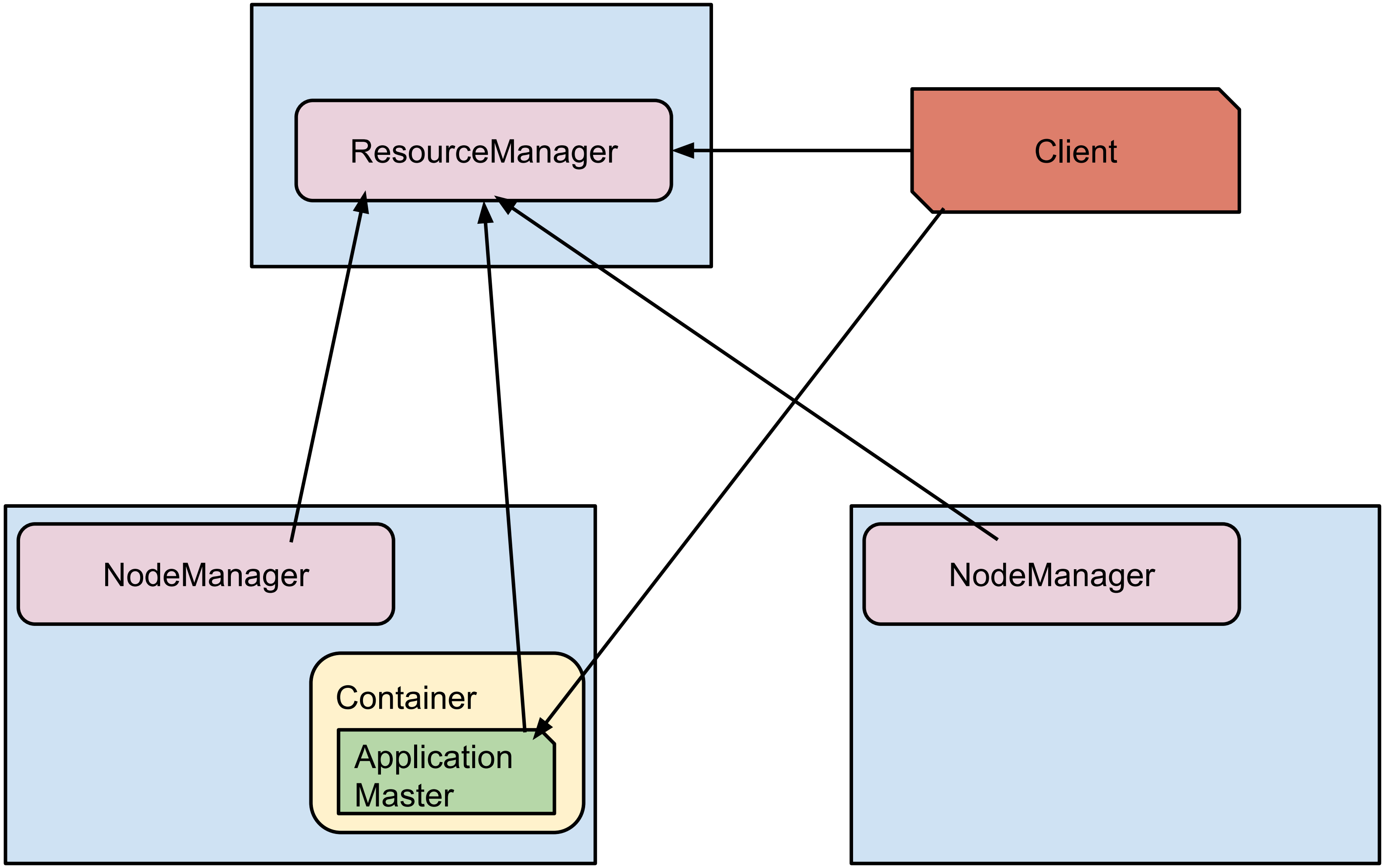
YARN architecture



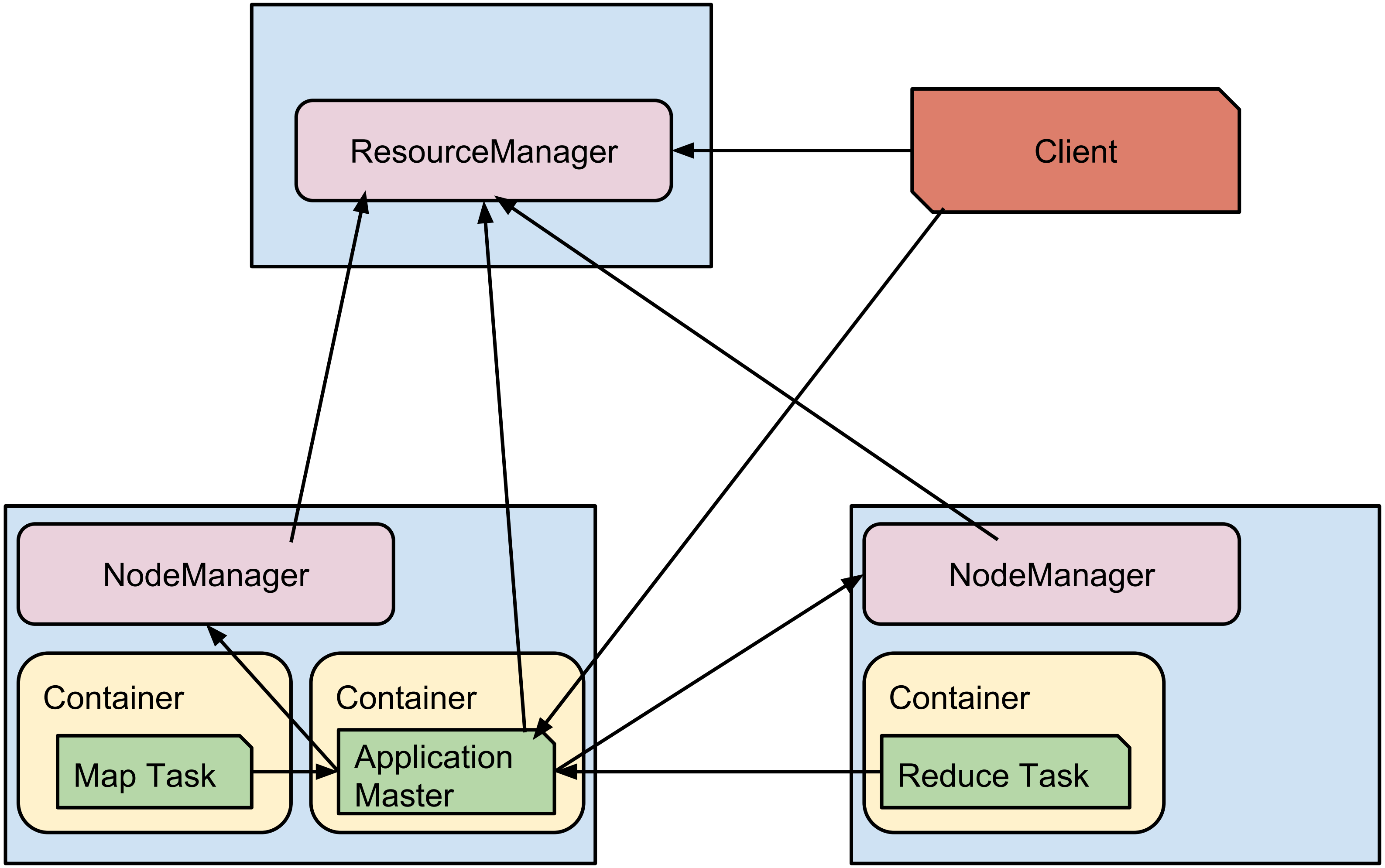
YARN architecture



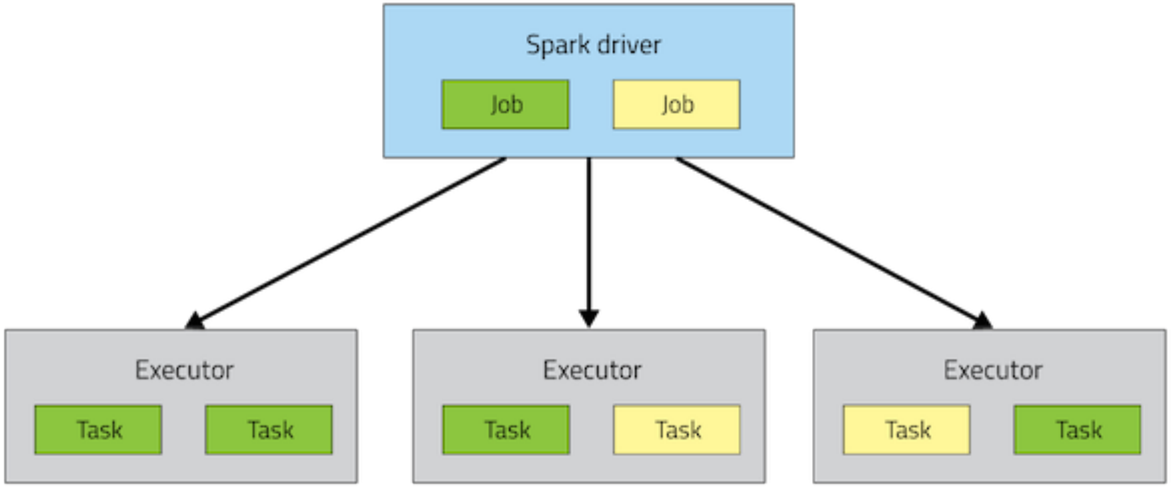
YARN architecture



YARN architecture

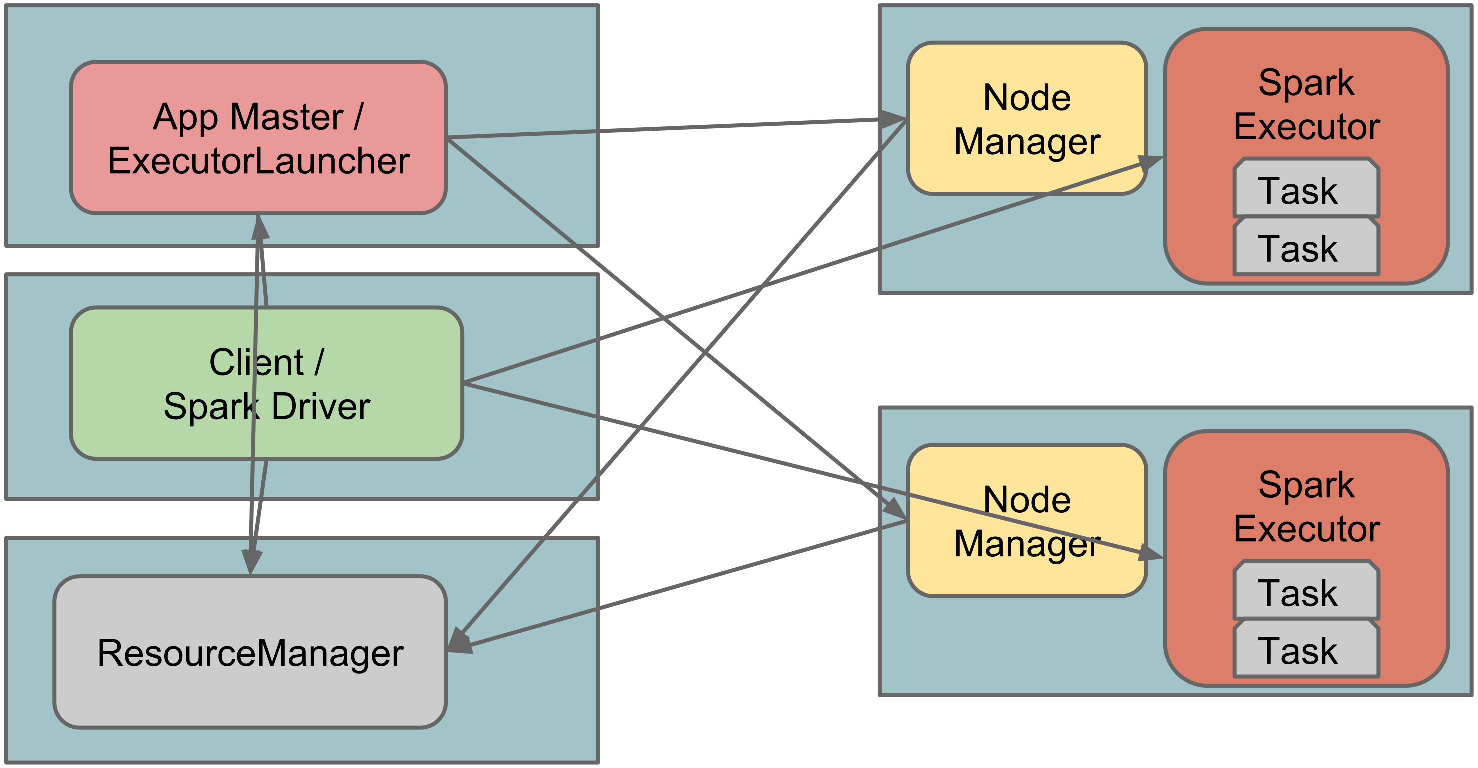


Spark architecture

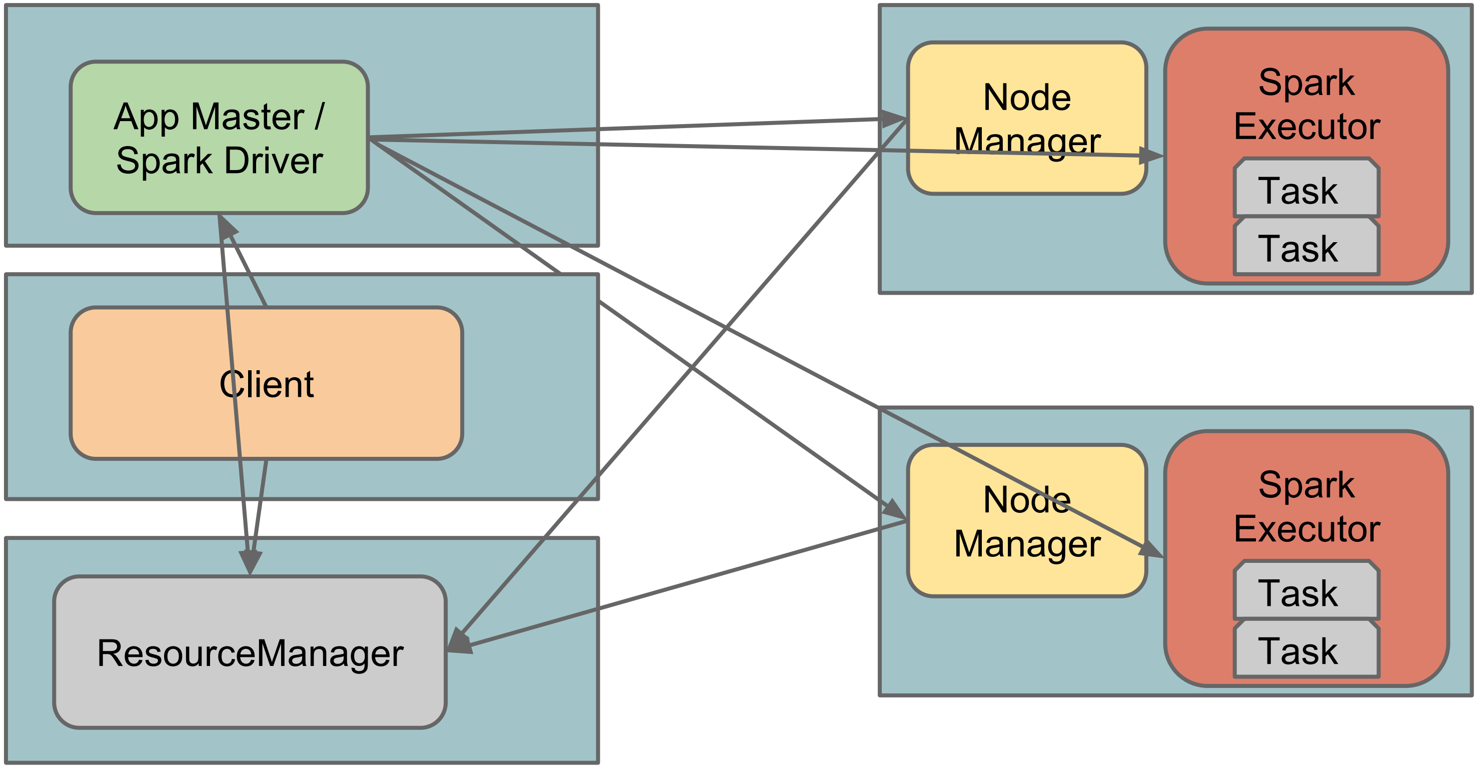


Where does the driver go?

yarn-client


App Master /
ExecutorLauncher
Client /
Spark Driver
ResourceManager
Node
Manager
Spark
Executor
Task
Task
Node
Manager
Spark
Executor
Task
Task

yarn-cluster (née yarn-standalone)



Data locality

- When running a job, Spark tries to place tasks alongside HDFS blocks

- Problem: Spark needs to ask YARN for executors before it runs its jobs

preferredNodeLocationData

- Solution: Tell Spark what files you’re going to touch when creating SparkContext

```
val locData = InputFormatInfo.computePreferredLocations
  (Seq(new InputFormatInfo(conf, classOf
  [TextInputFormat], new Path(“myfile.txt”)))

val sc = new SparkContext(conf, locData)
```

The Future

Don’t be greedy

- Spark holds on to full resources even when app is idle
- Give back to cluster
- Requires container-resizing (YARN-1197)

Integrate with Application TimelineStore

- Spark History Server
- Generic YARN history (YARN-321)

Smooth out rough edges

- Looking at logs should be easier
- Better documentation on data locality
- Remove unnecessary sleeps
- Long-running apps on secure clusters (YARN-941)

Thanks!