Spark Scheduler Enhancement

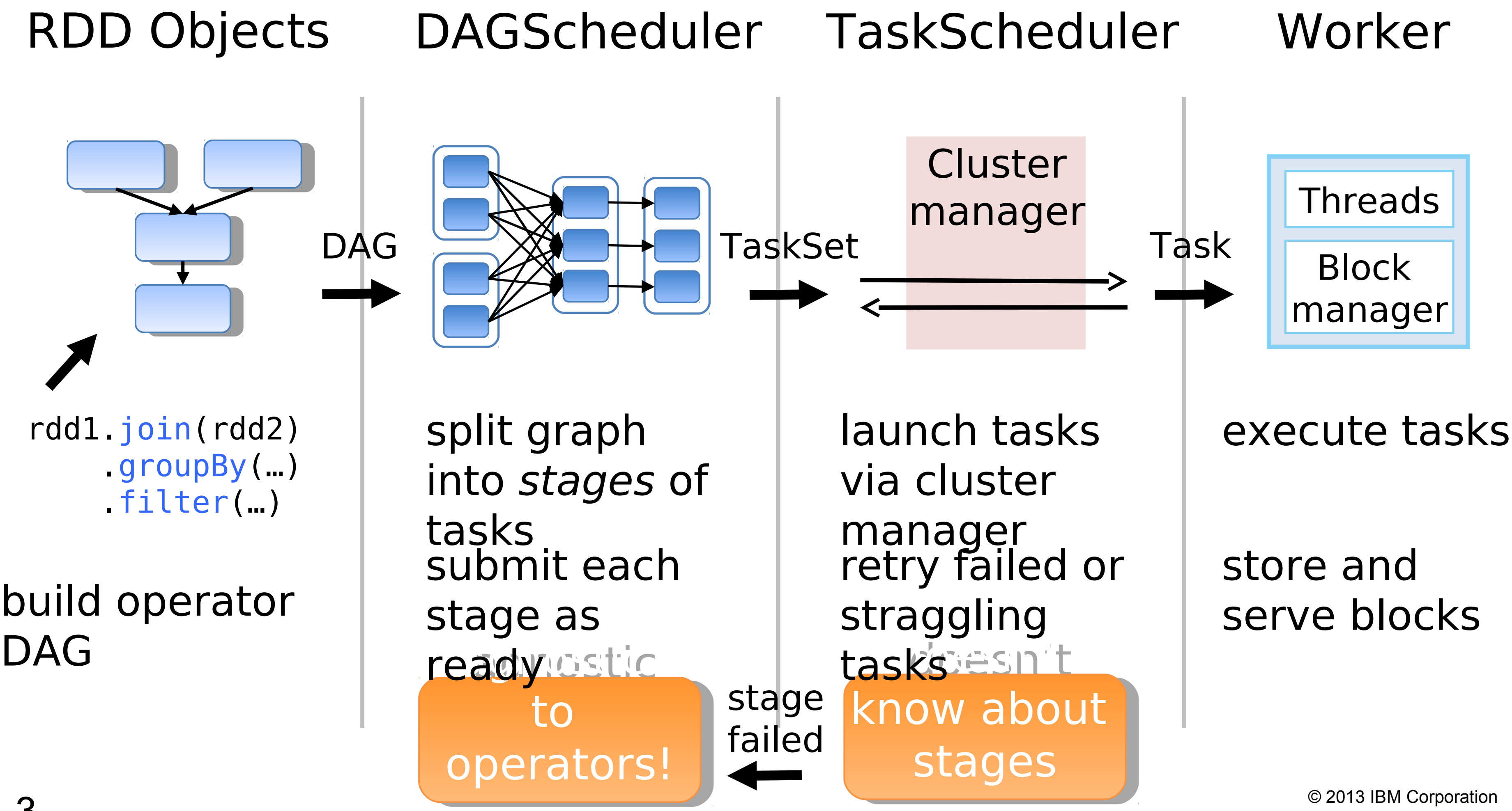
JUN FENG LIU (Software Architect)

liujunf@cn.ibm.com

Agenda

- Background
- Spark on YARN
- Spark on Mesos
- Spark on IBM Platform Symphony
- Spark SQL or Stream

Scheduling Process



RDD Objects

DAGScheduler

TaskScheduler

Worker

```
rdd1.join(rdd2)
    .groupBy(…)
    .filter(…)
```

build operator DAG

DAG

split graph into *stages* of tasks

submit each stage as ready

agnostic to operators!

TaskSet

Cluster manager

launch tasks via cluster manager

retry failed or straggling tasks

doesn't know about stages

stage failed

Task

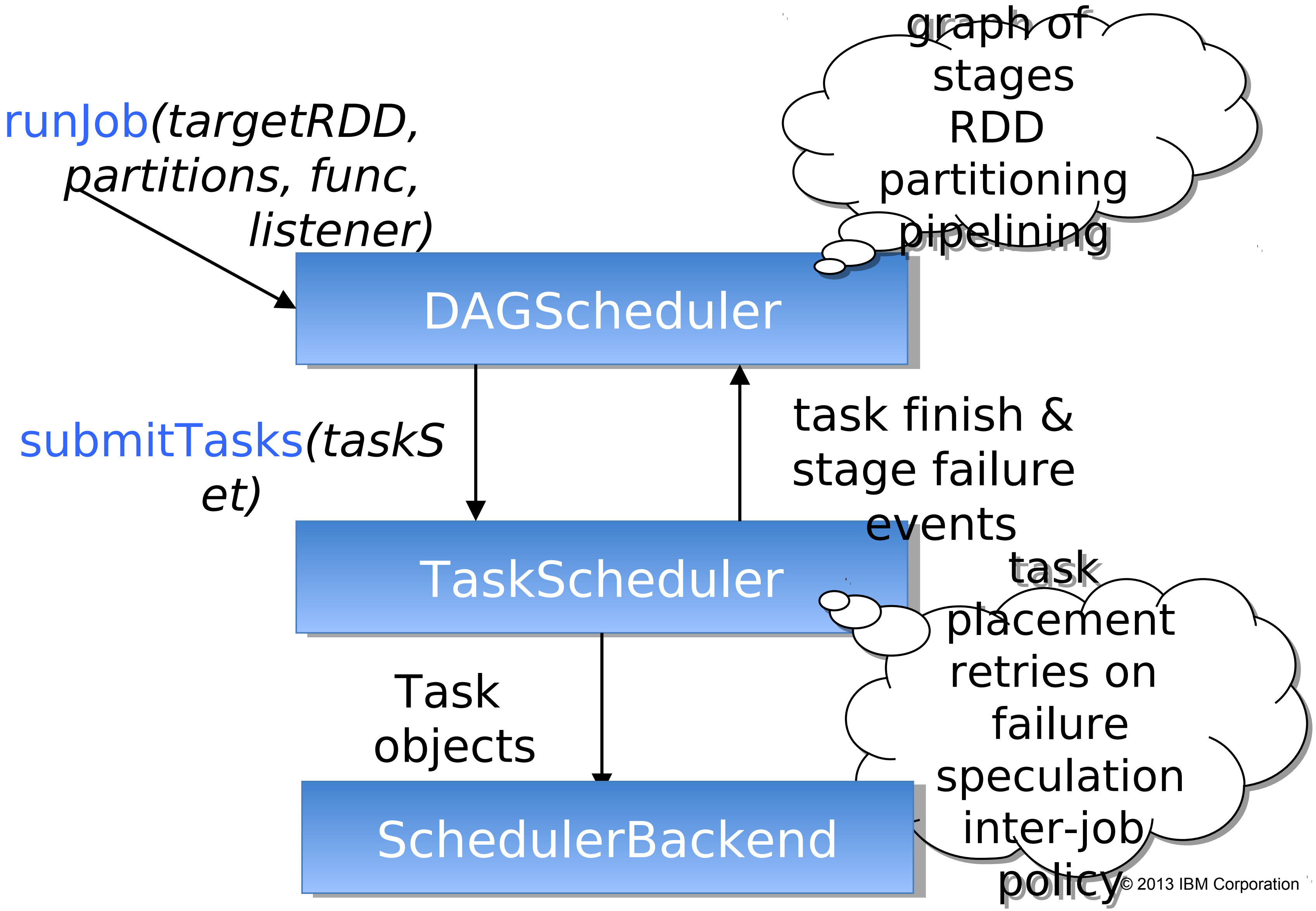
Threads

Block manager

execute tasks

store and serve blocks

Event Flow



runJob*(targetRDD, partitions, func, listener)*

DAGScheduler

graph of stages
RDD partitioning
pipelining

submitTasks*(taskSet)*

task finish & stage failure events

TaskScheduler

task placement
retries on failure
speculation
inter-job policy

Task objects

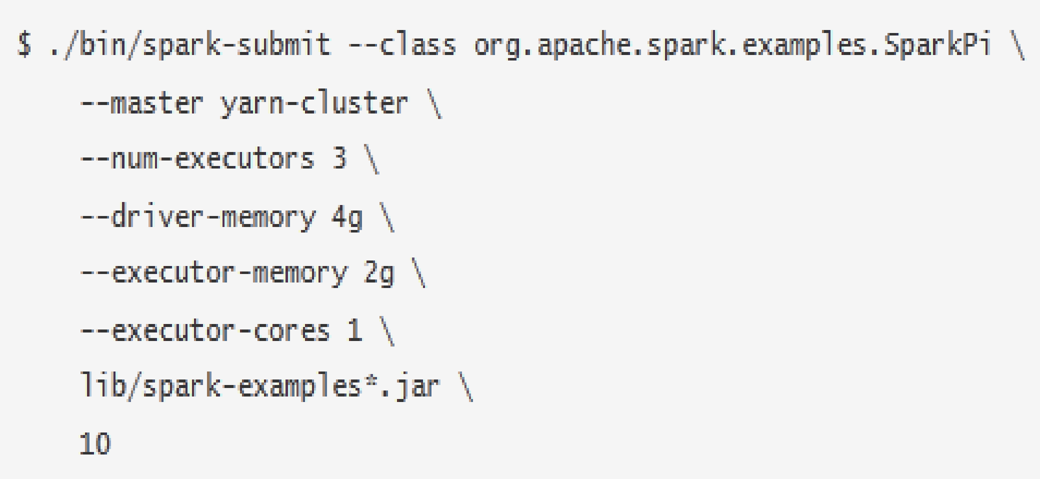
SchedulerBackend

Agenda

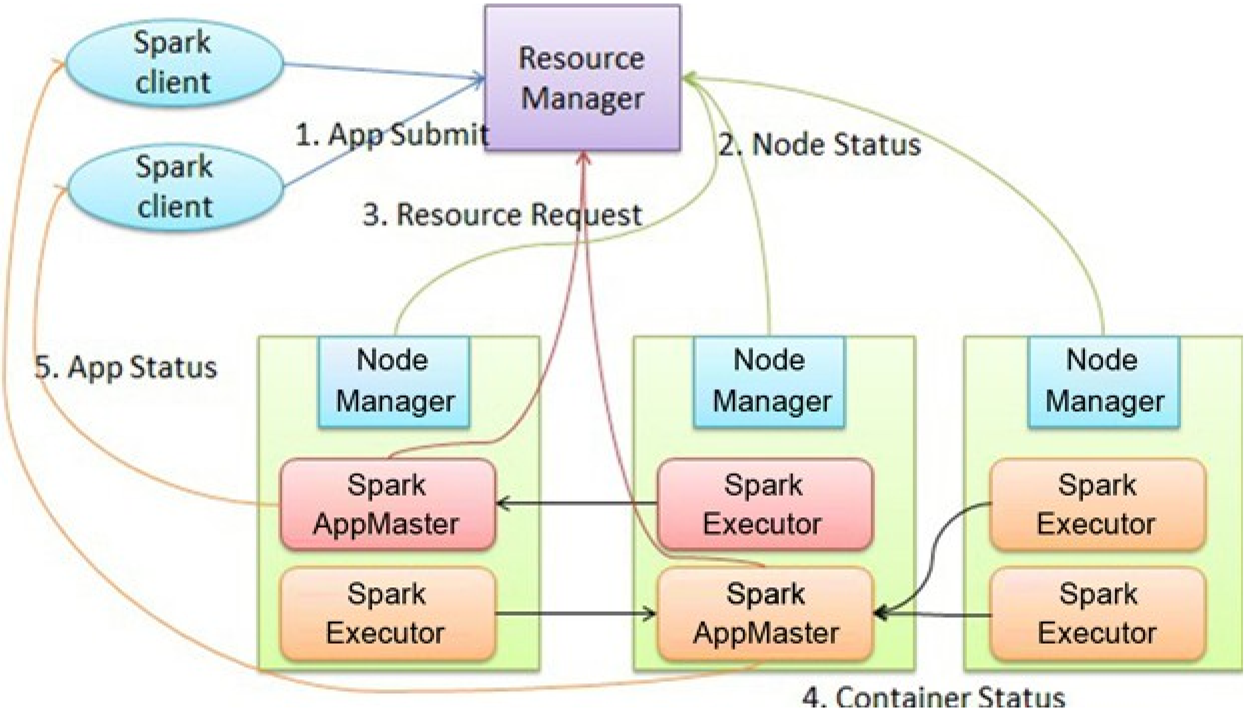
- **Background**
- **Spark on YARN**
- **Spark on Mesos**
- **Spark on IBM Platform Symphony**
- **Spark SQL or Stream**

YARN

- **Coarse Model Scheduler**
  - Fix number Executor
  - Fix number Core
  - Fix number Memory



```
$ ./bin/spark-submit --class org.apache.spark.examples.SparkPi \
    --master yarn-cluster \
    --num-executors 3 \
    --driver-memory 4g \
    --executor-memory 2g \
    --executor-cores 1 \
    lib/spark-examples*.jar \
    10
```

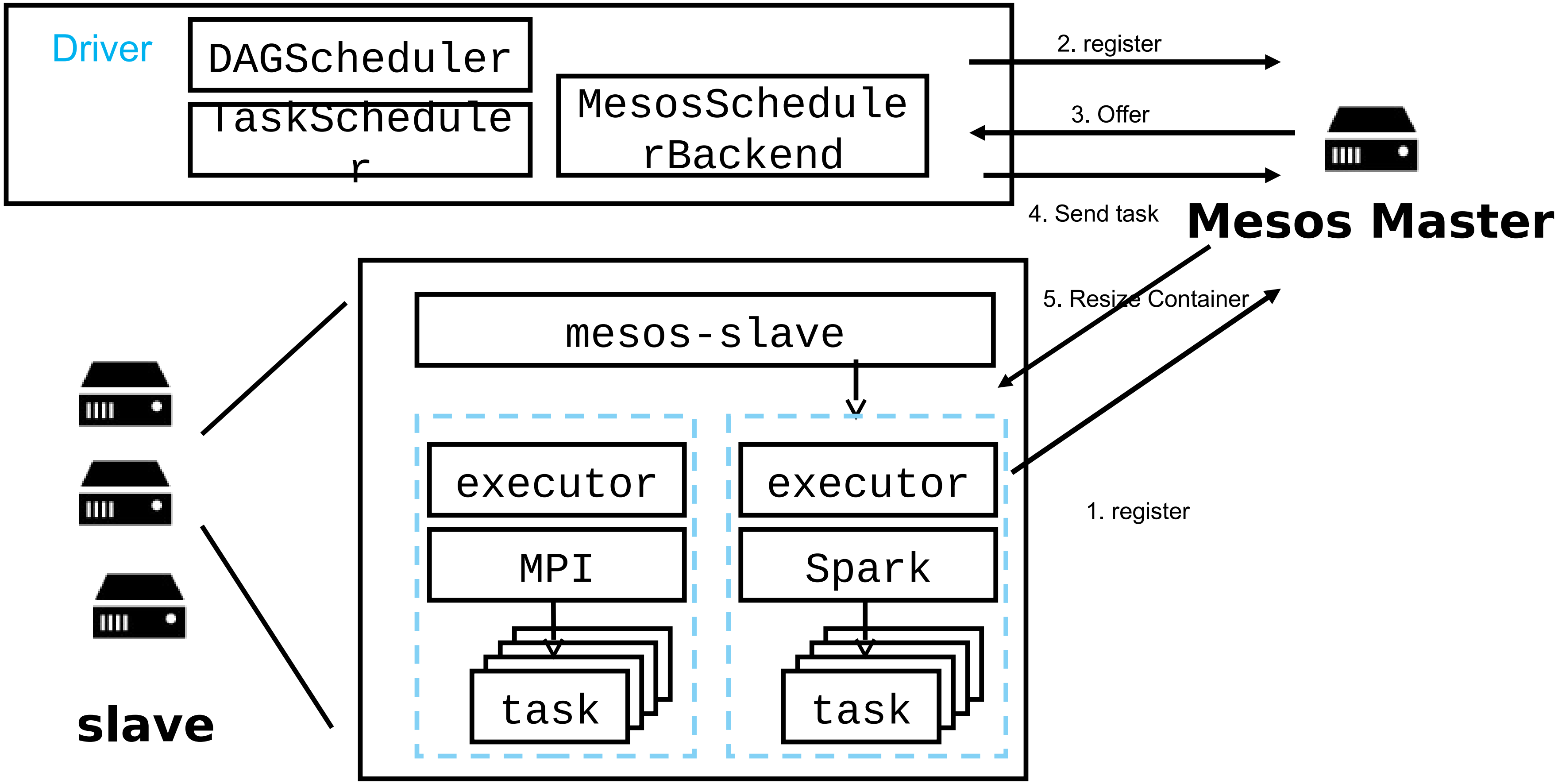


YARN

- **Data locality**
  - Rely on Driver Code (error prone)
  - Query the HDFS to get prefer hosts

```
val sparkConf = new SparkConf().setAppName("SparkHdfsLR")
    val inputPath = args(0)
    val conf = SparkHadoopUtil.get.newConfiguration()
    val sc = new SparkContext (sparkConf,
      InputFormatInfo.computePreferredLocations (
        Seq(new InputFormatInfo(conf,
  classOf[org.apache.hadoop.mapred.TextInputFormat],
  “hdfs://path/to/data.file”)) ))
```

Fine Grained on Mesos



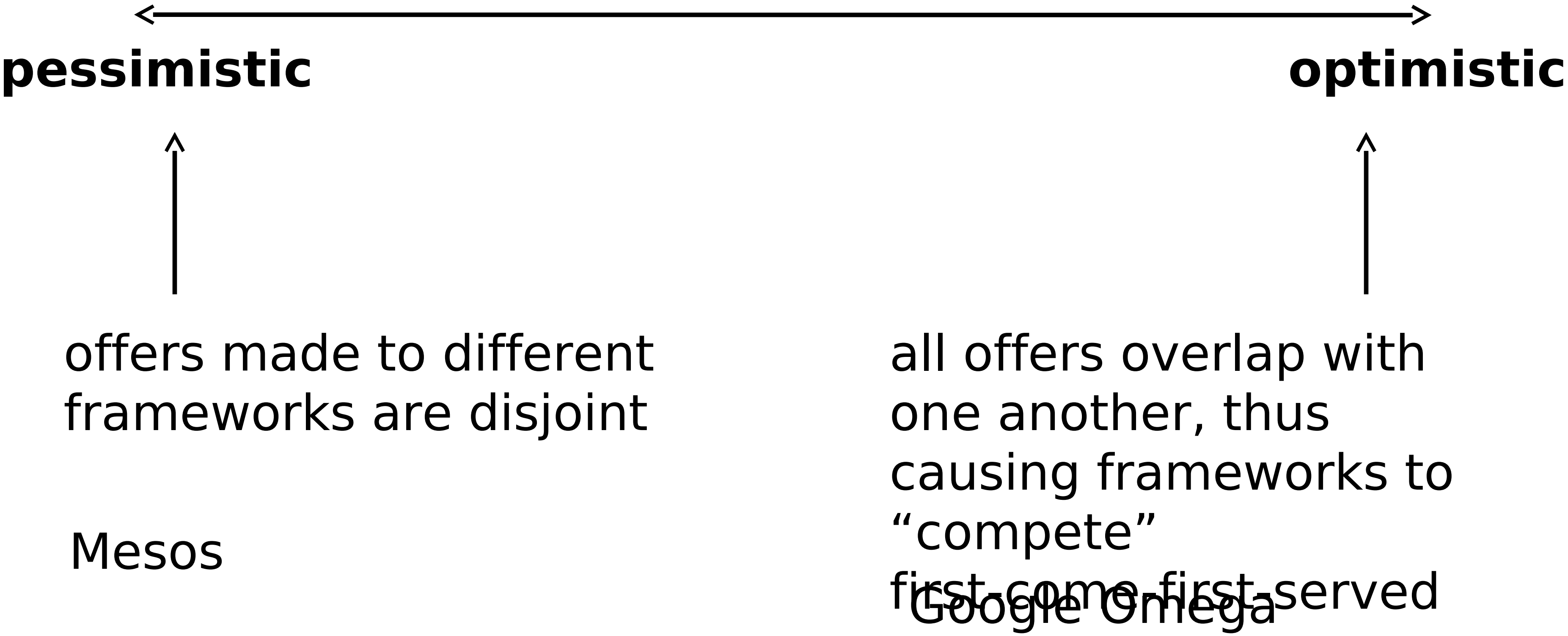
Fine Grained on Mesos

- **Resource Allocation**
  - Decided by Mesos DRF scheduler
  - Offer triggered when
    - Task Finished
    - Extra Slave add in
    - Application launch
  - Pessimistic offer
  - Revocable resources when task complete
  - Resource Share is good when task is short
  - No reclaim when task is long

Fine Grained on Mesos

- **Defer scheduling**
  - Spark can not choice preference host in Mesos model
  - Defer to schedule task if the data locality is not good with current offer
  - Defer scheduling in three level
    - spark.locality.wait.process
    - spark.locality.wait.node
    - spark.locality.wait.rack

Fine Grained on Mesos



pessimistic

optimistic

offers made to different frameworks are disjoint

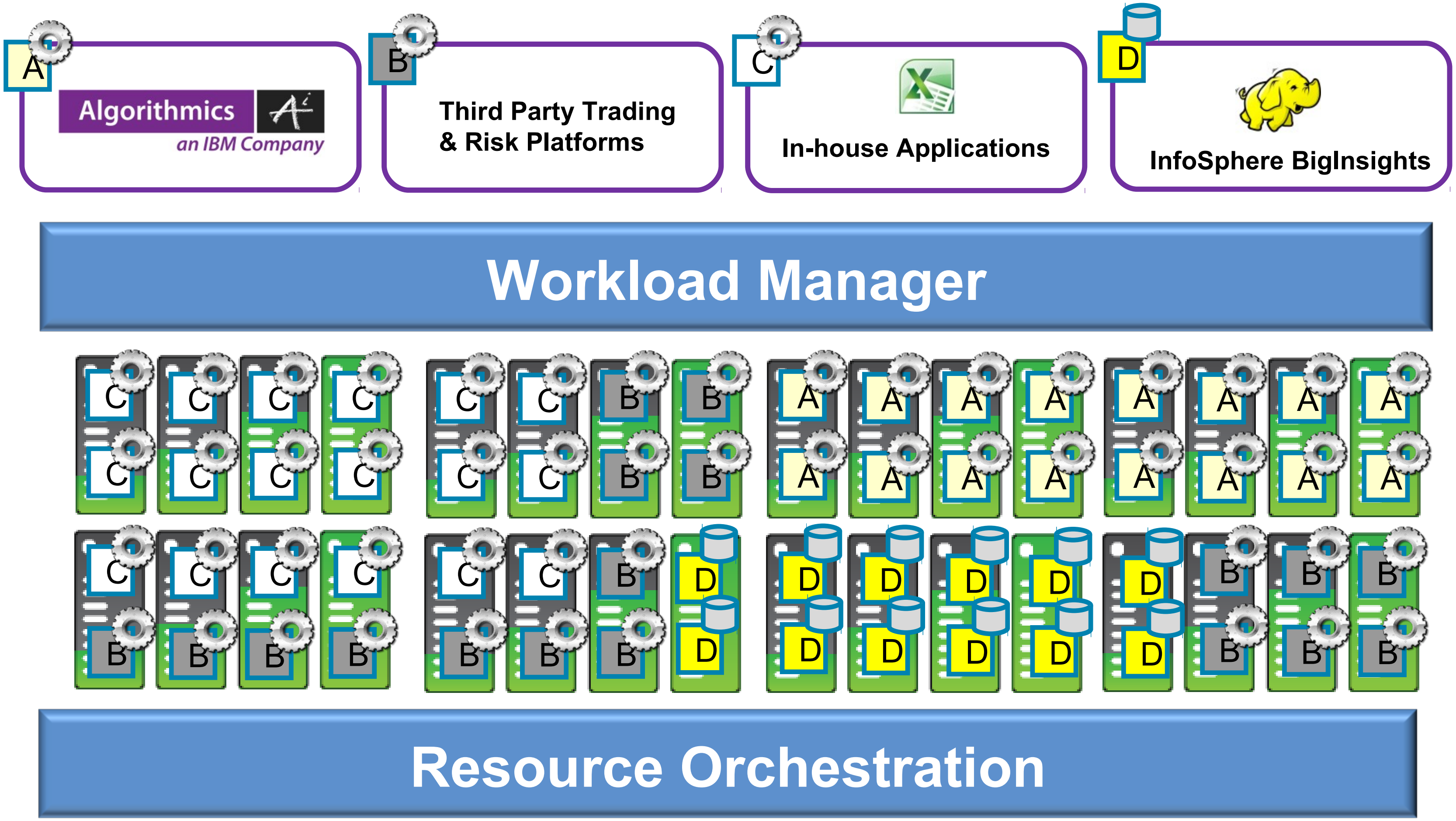
Mesos

all offers overlap with one another, thus causing frameworks to “compete” first-come-first-served

Google Omega

IBM Platform Symphony

Multiple users, applications and lines of business on a shared, heterogeneous, multi-tenant grid



Third Party Trading & Risk Platforms

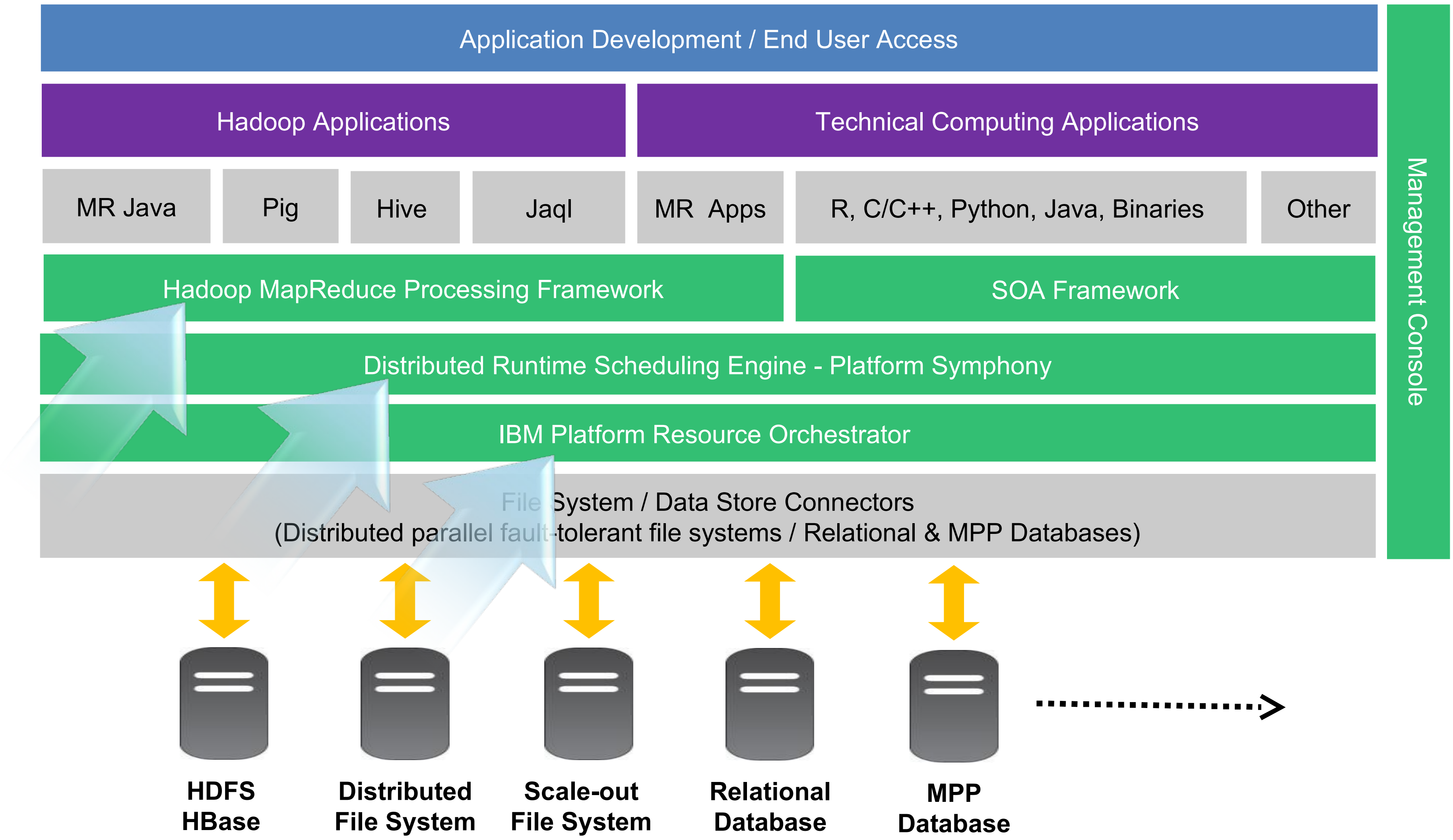
In-house Applications

InfoSphere BigInsights

Workload Manager

Resource Orchestration

Application & Data Integration Architecture



Application Development / End User Access

Hadoop Applications

Technical Computing Applications

MR Java

Pig

Hive

Jaql

MR Apps

R, C/C++, Python, Java, Binaries

Other

Hadoop MapReduce Processing Framework

SOA Framework

Distributed Runtime Scheduling Engine - Platform Symphony

IBM Platform Resource Orchestrator

File System / Data Store Connectors
(Distributed parallel fault-tolerant file systems / Relational & MPP Databases)

Management Console

HDFS
HBase

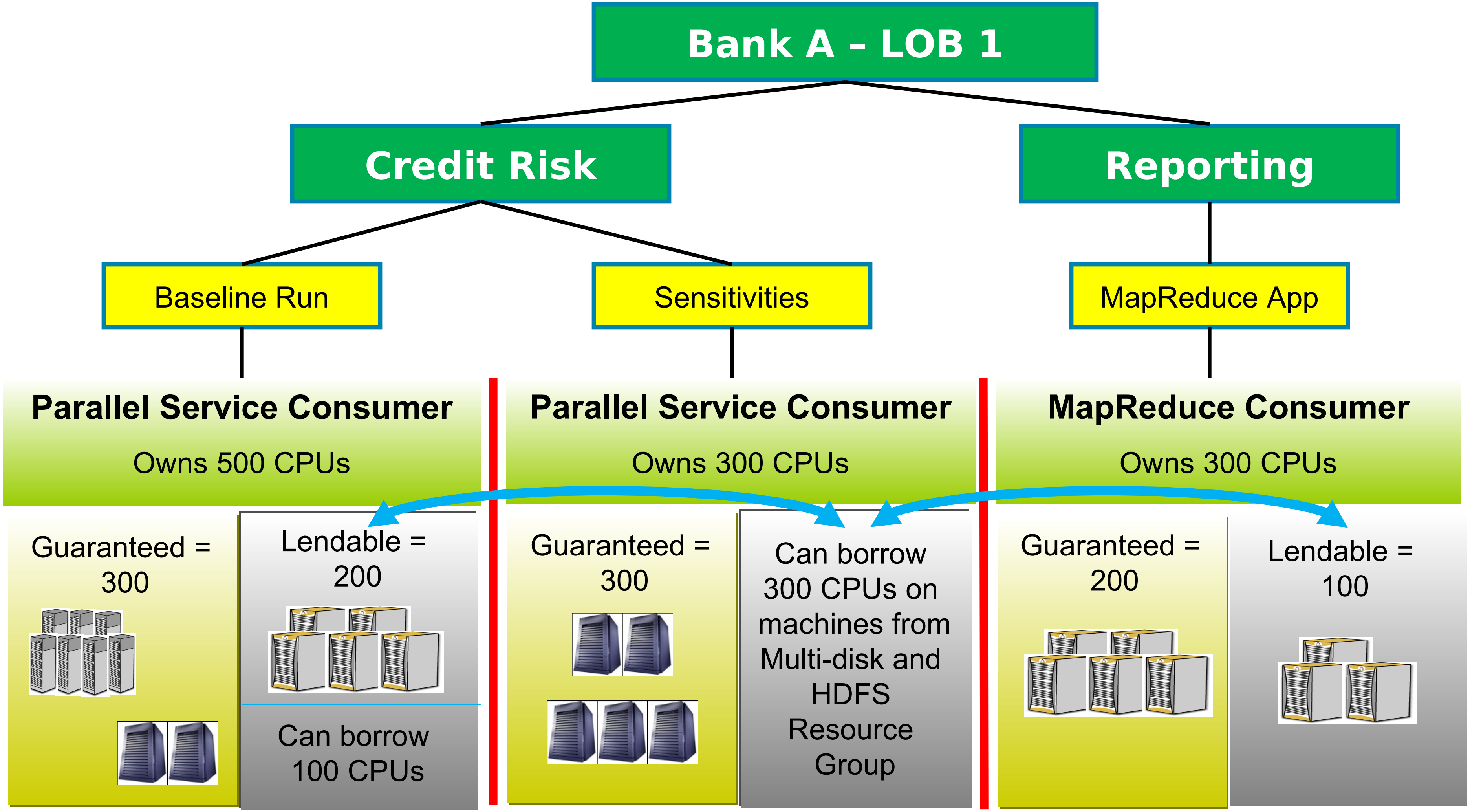
Distributed
File System

Scale-out
File System

Relational
Database

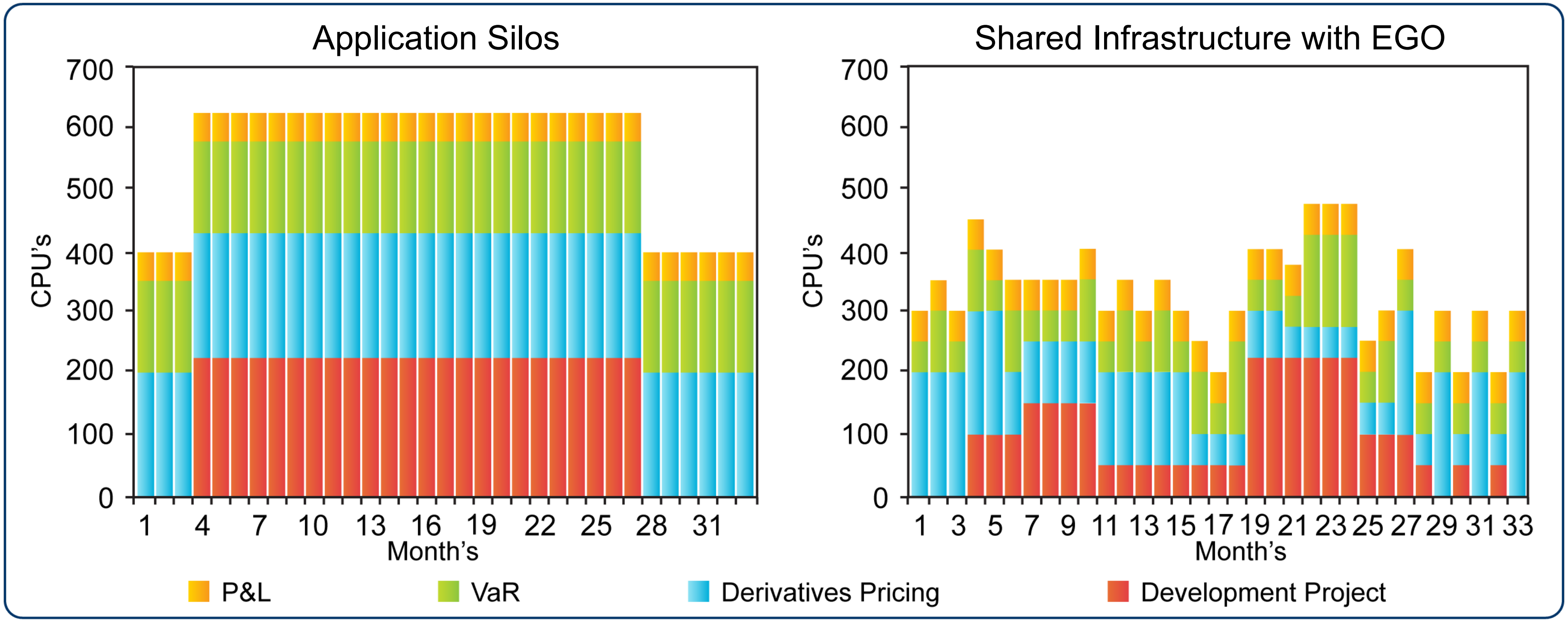
MPP
Database

Parallel Services and MapReduce sharing an EGO Cluster

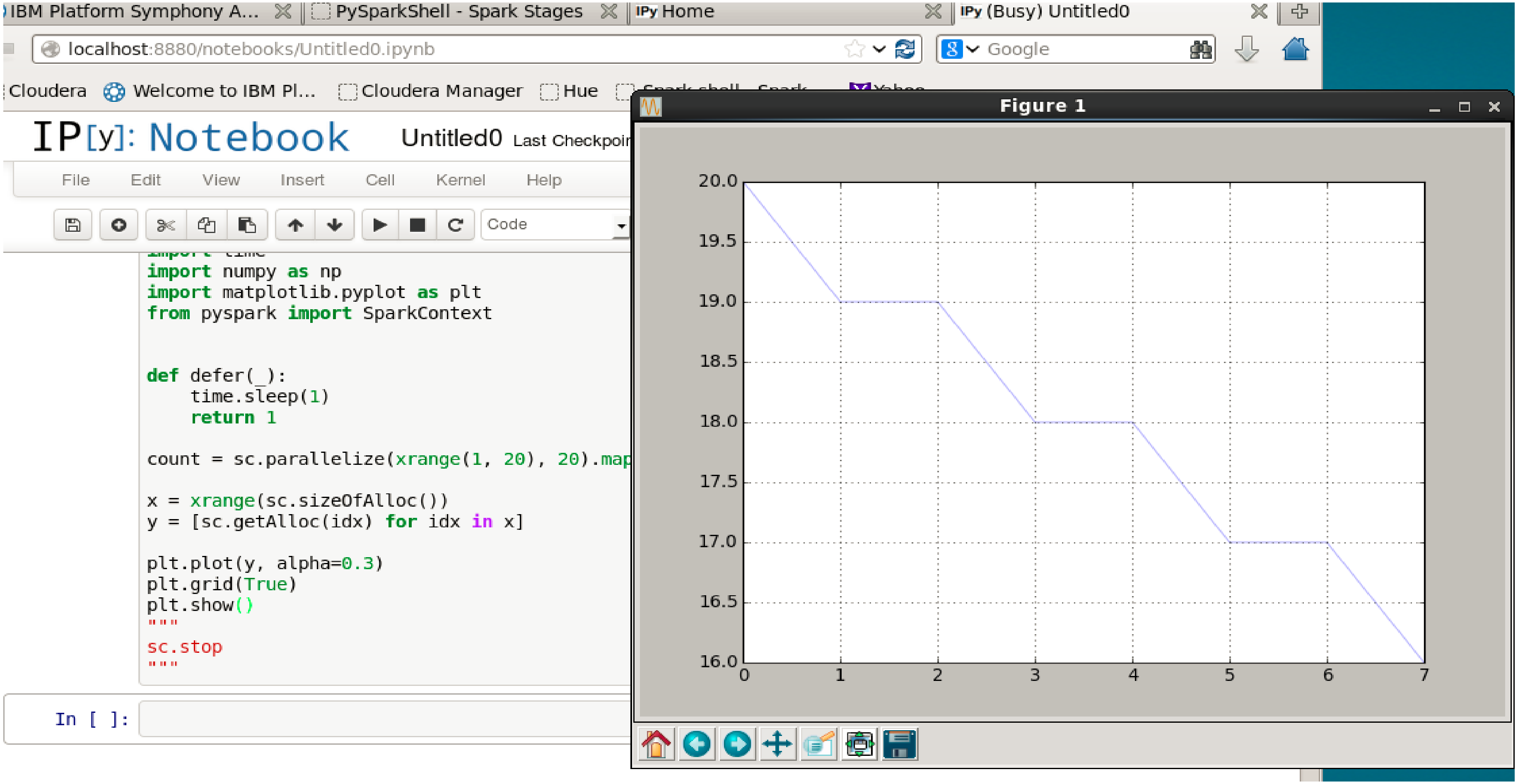


Policy-driven sharing of resources: Lending / Borrowing / Reclaiming

On-Demand & Reservation Allocations



Screen Shot

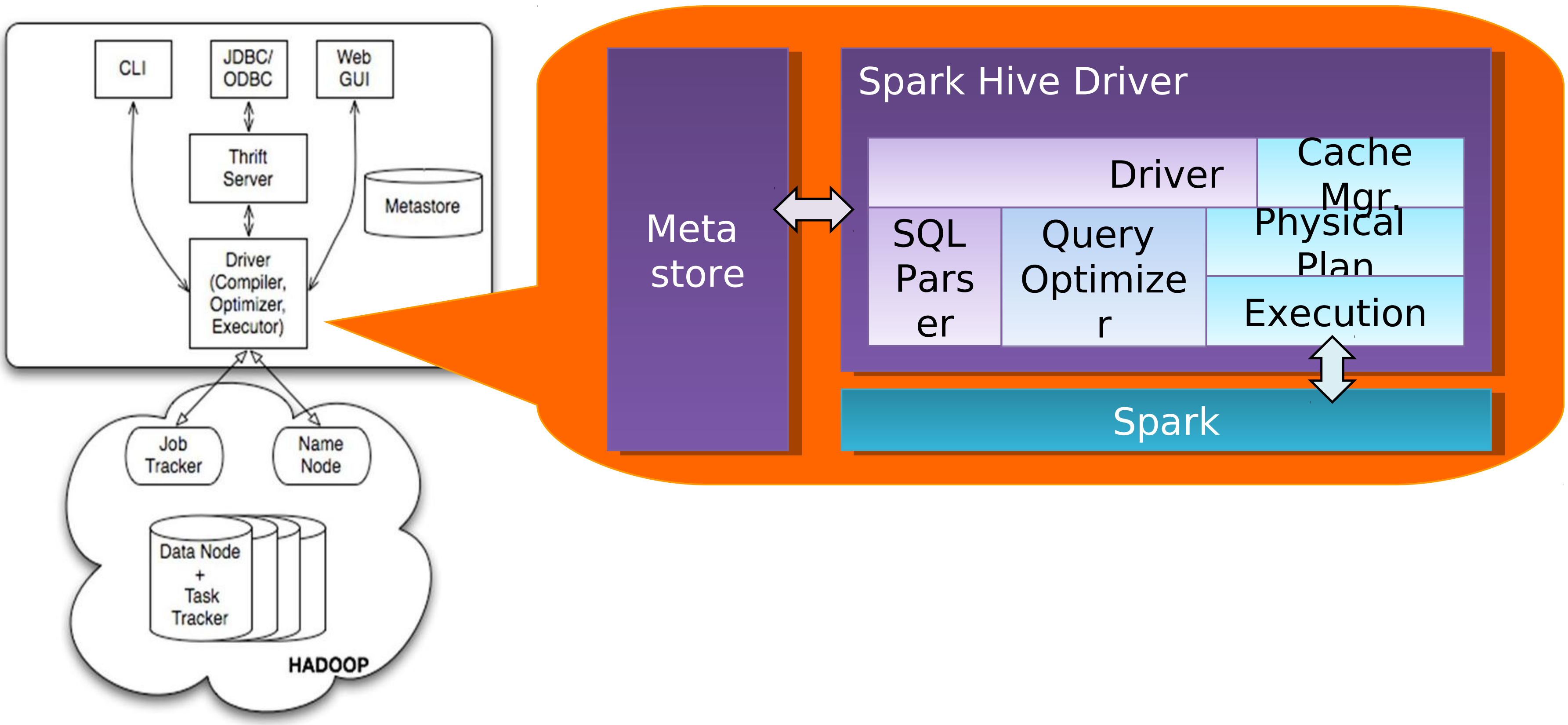


```
import time
import numpy as np
import matplotlib.pyplot as plt
from pyspark import SparkContext


def defer(_):
    time.sleep(1)
    return 1

count = sc.parallelize(xrange(1, 20), 20).map

x = xrange(sc.sizeOfAlloc())
y = [sc.getAlloc(idx) for idx in x]

plt.plot(y, alpha=0.3)
plt.grid(True)
plt.show()
"""
sc.stop
"""
```

Agenda

- **Background**
- **Spark on YARN**
- **Spark on Mesos**
- **Spark on IBM Platform Symphony**
- **Spark SQL or Stream**

Spark SQL


CLI
JDBC/ ODBC
Web GUI
Thrift Server
Metastore
Driver (Compiler, Optimizer, Executor)
Job Tracker
Name Node
Data Node + Task Tracker
HADOOP
Meta store
Spark Hive Driver
Driver
Cache Mgr.
SQL Parser
Query Optimizer
Physical Plan
Execution
Spark

Benefits for SQL style workload

- **Centralized Scheduling**
  - Shared spark context among SQL query
  - Consolidated requests from clients
- **Resource negotiation**
  - Spark Context Scheduler
  - Request resource from RM as batch style
  - Reduce overhead to RM
- **Resource Share**
  - Better resource utilization based on workload driven
  - Reclaim happen based on priority
- **Task priority**
  - Based on existing Spark Context style