The Torino Yoga Cluster

Federica Legger, **Sara Vallero**

Architecture

JupytherHub and HDFS deployed with Helm.

GitHub
OAuth login

Monitoring:Prometheus + Grafana

jupyterhub

jupyter Driver

Executor Executor Executor

jupyter Driver

Executor Executor

HDFS (for data-sets)

Kubernetes Control Plane

Kubernetes Workers

High-class hardware

Lower-class hardware

VMs

A word on multitenancy

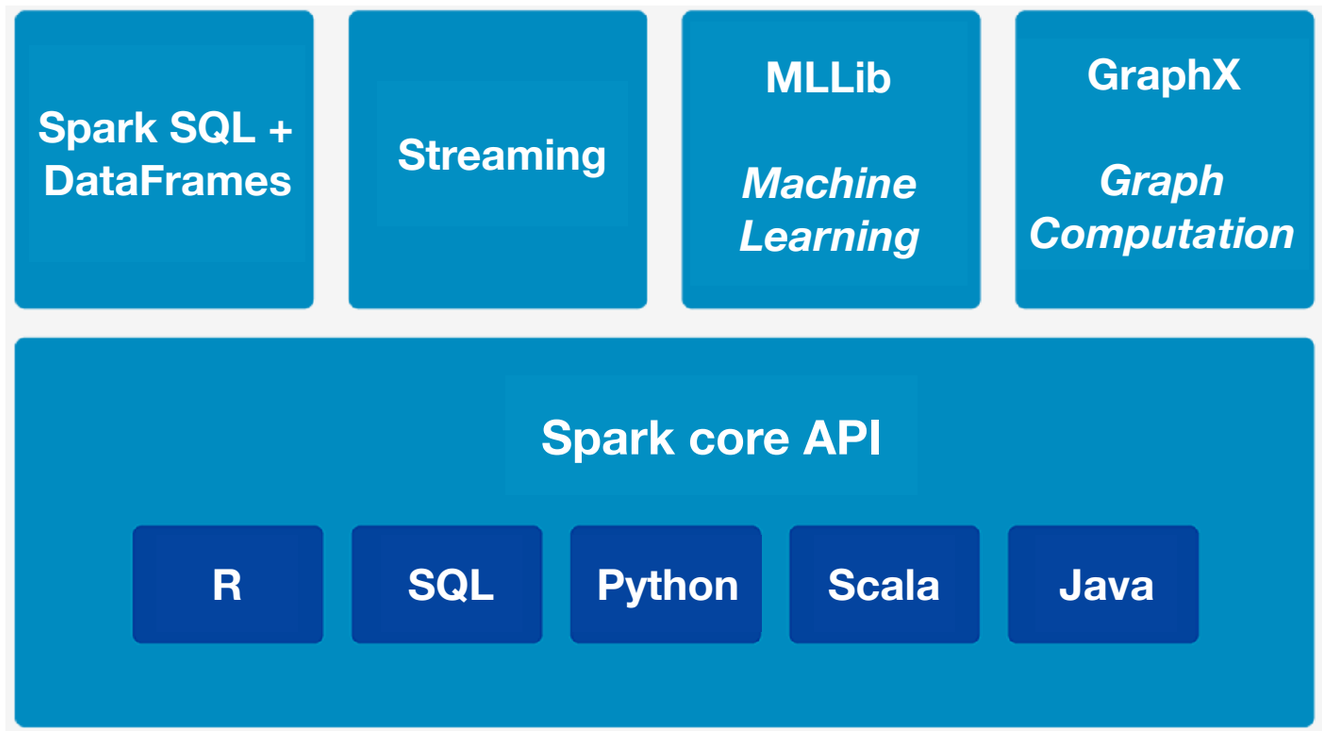
- Whitelist of allowed users
- For each user:
    - Home directory on local storage
    - Headless service for the Spark Driver
    - Home *directory* on HDFS
    - Create a Namespace
    - Create a Service Account (Kubernetes RBAC)
    - Create a Role that can do anything in the given Namespace and watch other users' pods and cluster nodes
    - Write the corresponding Kubernetes config file in the user's home
    - Create a *Farm* resource (more on this later)

All with a trivial bash script.

Some useful tools

Apache Spark

- a large-scale distributed **general-purpose** cluster-computing framework
- an interface for programming entire clusters with **implicit data parallelism** and **fault tolerance**
- core data processing engine + specific libraries
- these libraries can be combined in modern data **pipelines** (i.e. analytics and machine learning workloads)
- jobs perform multiple operations consecutively, **in memory**, only spilling to disk when required

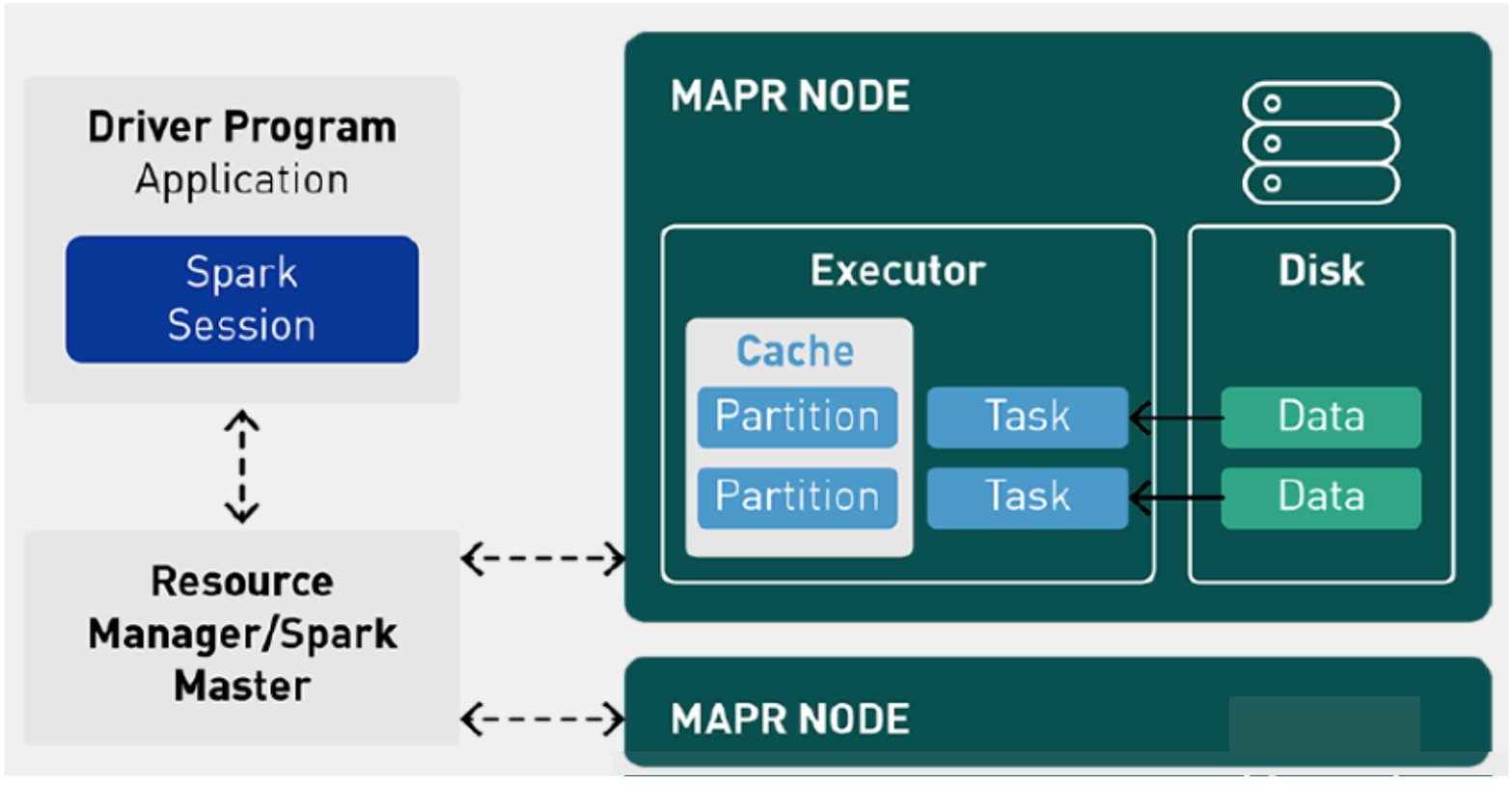


Spark Architecture

The Spark project was born (Berkley 2009) as an improvement of the Hadoop MapReduce framework.

Spark makes it better because:

- runs multi-threaded lightweight tasks inside of JVM processes, providing fast job startup and **parallel multi-core CPU utilization**
- **caches data in memory** across multiple parallel operations, especially fast for parallel processing of distributed data with **iterative algorithms**



hadoop Distributed FileSystem (HDFS)

- for managing pools of **big data** and supporting related analytics applications
- rapid transfer of data between compute nodes
- closely coupled with MapReduce
- breaks the information down into separate blocks and distributes them to different nodes in a cluster: highly **efficient parallel processing**
- highly **fault-tolerant** (data-wise). The file system replicates each piece of data multiple times and distributes the copies to individual nodes. In case of a node crash, processing can continue while data is recovered
- echoes POSIX design style in some aspects
- very large-scale implementations
- support for **low-cost commodity hardware**

HDFS directories

Hadoop Overview Datanodes Snapshot Startup Progress Utilities

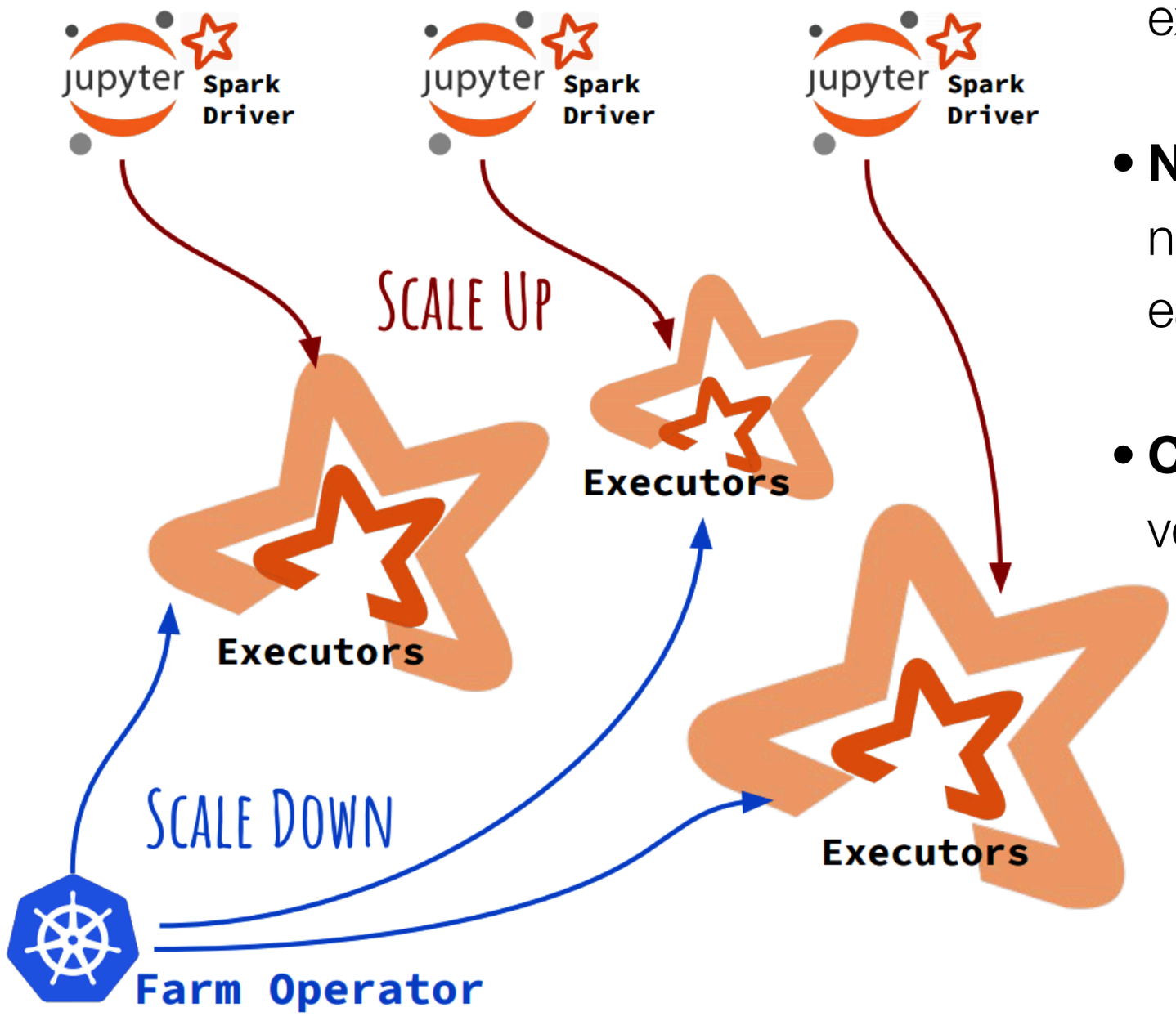
Browse Directory

/user Hierarchical directory structure Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	jovyan	jovyan	0 B	11/15/2019, 12:12:08 PM	0	0 B	aliber92
drwxr-xr-x	jovyan	jovyan	0 B	11/14/2019, 6:48:25 PM	0	0 B	bagnasco
drwxr-xr-x	jovyan	jovyan	0 B	11/14/2019, 6:40:00 PM	0	0 B	gabrielefronze
drwxr-xr-x	jovyan	jovyan	0 B	11/14/2019, 6:53:12 PM	0	0 B	giorgiobar
drwxr-xr-x	jovyan	jovyan	0 B	11/14/2019, 6:36:21 PM	0	0 B	leggerf
drwxr-xr-x	jovyan	jovyan	0 B	11/15/2019, 12:13:42 PM	0	0 B	marco-ph
drwxr-xr-x	jovyan	jovyan	0 B	11/15/2019, 12:15:11 PM	0	0 B	obertino
drwxr-xr-x	jovyan	jovyan	0 B	11/14/2019, 6:49:07 PM	0	0 B	slusso
drwxr-xr-x	jovyan	jovyan	0 B	9/30/2019, 12:53:34 PM	0	0 B	svallero
drwxr-xr-x	jovyan	jovyan	0 B	10/2/2019, 11:37:29 AM	0	0 B	testuser
drwxr-xr-x	jovyan	jovyan	0 B	10/7/2019, 7:16:50 PM	0	0 B	testuser2

Elasticity

Elasticity



- **Spark driver** continuously **scales up** to reach the requested number of executors

- **No static quotas** enforced, but a Min number of executors to be granted to each tenant

- **Custom Kubernetes Operator** (alpha version):
  - lets tenants occupy all available resources in a FIFO manner
  - undeploys exceeding executors only to grant the Min number of resources to all registered tenants

Kubernetes Operators

- define and encapsulate custom units of business logic
- define, monitor and recover custom components (CRD)
- the things that Kubernetes monitors can be referred to as Perceived State, and comparing this against reality, referred to as Actual State
- RECONCILIATION: when Kubernetes detects a discrepancy between perceived and actual states it then calculates which action to take to make these two states match

At its heart Kubernetes is nothing more than a monitoring system; essentially an infinite loop, constantly monitoring everything it knows about.

https://itnext.io/kubernetes-operators-getting-down-to-business-logic-93d6f5303813

Farm Kube Operator

https://github.com/svallero/farmcontroller

- Spark Driver deploys executor Pods with given namespace/label/name (let's call this triad a selector)
- But a Pod is not a scalable Kubernetes Resource (i.e. a Deployment is)
- The farm Operator **implements two Custom Resource Definitions** (CRDs) with their own Controller:
  - Farm Resource
  - FarmManager Resource
- The Farm Operator **can be applied to any other app** (farm type) with similar features
- CAVEAT:
  - The Farm app should be resilient to the live removal of executors (i.e. Spark, HTCondor)

The Farm CRD

- Collects Pods with given selector
- Has a configurable desired number of Replicas (used for scaledown only)
- Defines a Min number of executors (quota)
- Reconciles on selected Pod events
- One Farm for each user

The Farm CRD

```
Name:         jupyter-svallero
Namespace:    svallero
Labels:       spark-role=executor
Annotations:  kubectl.kubernetes.io/last-applied-configuration:
                {"apiVersion":"farmcontroller.toinfn.it/v1alpha1","kind":"Farm","metadata":{"anno
tations":{},"labels":{"spark-role":"executor"},"name":"ju...
API Version:  farmcontroller.toinfn.it/v1alpha1
Kind:         Farm
Metadata:
  Creation Timestamp:  2019-11-04T17:36:23Z
  Generation:          4780
  Resource Version:    115516235
  Self Link:           /apis/farmcontroller.toinfn.it/v1alpha1/namespaces/svallero/farms/jupyter-
  UID:                 c2c043ba-d4e5-4916-87e0-f113272d4141
Spec:
  Label Key:                    spark-role
  Label Value:                  executor
  Max Executors:                0
  Min Executors:                25
  Scaledown After N Triggers:   5
Status:
  All Executors:         0
  Error Executors:       0
  Overquota:             0
  Pending Executors:     0
  Running Executors:     0
  Scaledown Triggered:   0
Events:
  Type    Reason   Age                    From  Message
  ----    ------   ----                   ----  -------
  Normal  Updated  11m (x1098 over 12d)  Farm  Farm status updated
```

Desired number of Replicas (0 = not set)

Cleanup executors in Error state

The FarmManager CRD

- Reconciles on Farm events
- Scales down Farms over quota (after a configurable number of *triggers*) only if some other Farm requests resources and it's below its quota
- **Simple algorithm:** number of killed pods per Farm is proportional to the number of Pods over the quota (could be improved/modified)
- One FarmManager per Farm type (i.e. Spark,HTCondor)

The FarmManager CRD

```
Kind:         FarmManager
Metadata:
  Creation Timestamp:  2019-10-29T16:38:18Z
  Generation:          225
  Resource Version:    8392250
  Self Link:           /apis/farmcontroller.toinfn.it/v1alpha1/farmmanagers/farmmanag
er-spark
  UID:                 ad53b81f-c7bd-49f4-8eb4-2a0b42dc459f
Spec:
  Label Key:    spark-role
  Label Value:  executor
Status:
  Overquota:
    svallero/jupyter-svallero:    0
    testuser/jupyter-testuser:    0
    testuser2/jupyter-testuser2:  0
  Pending:
    svallero/jupyter-svallero:    5
    testuser/jupyter-testuser:    5
    testuser2/jupyter-testuser2:  5
  Tobescaleddown:
    svallero/jupyter-svallero:    0
    testuser/jupyter-testuser:    0
    testuser2/jupyter-testuser2:  0
  Underquota:
    svallero/jupyter-svallero:    0
    testuser/jupyter-testuser:    0
    testuser2/jupyter-testuser2:  0
```

Could be any other cloud-aware application

What about HPCs?

- HPC = high performance processors + low latency interconnect
- HPC clusters are typically managed with a batch system
- The OCCAM HPC @University of Torino employs a Cloud like management strategy coupled to lightweight virtualization

(https://c3s.unito.it/index.php/super-computer)

For the time being we use Mesos…
but Kubernetes is coming.