Experience of Running Spark on Kubernetes on OpenStack for High Energy Physics Workloads

Prasanth Kothuri, CERN
Piotr Mrowczynski, CERN

#SAISEco11

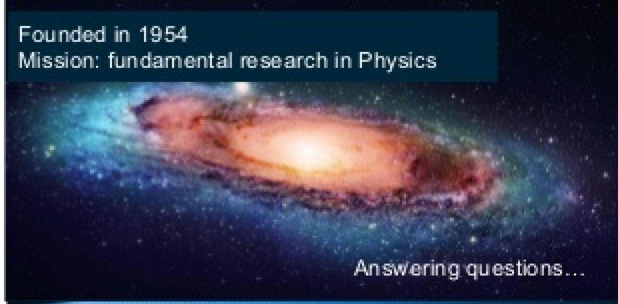
CERN

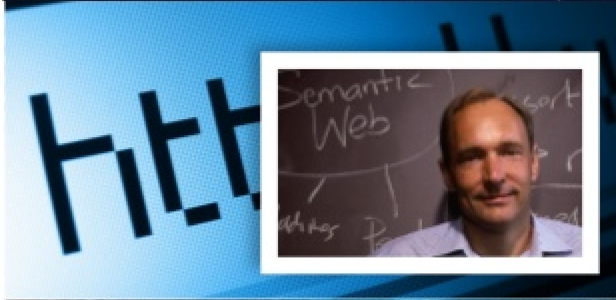
- CERN - European **Laboratory** for Particle Physics
- The place where the **Web** was born
- Home of the **Large Hadron Collider** and 4 big Experiments:
  - ATLAS ~ CMS ~ LHCb ~ ALICE
- 22 member states + 6 associate members + **world-wide** collaborations

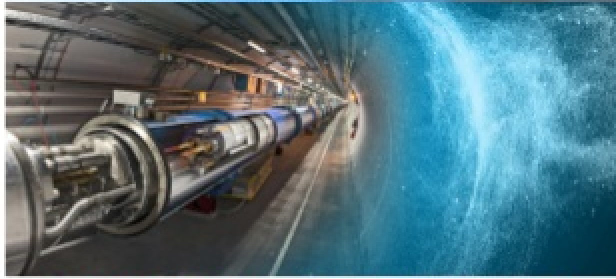
  ~3000 Members of Personnel

  ~12,000 users

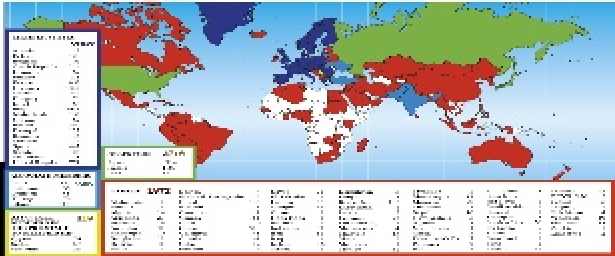
  ~1000 MCHF yearly budget



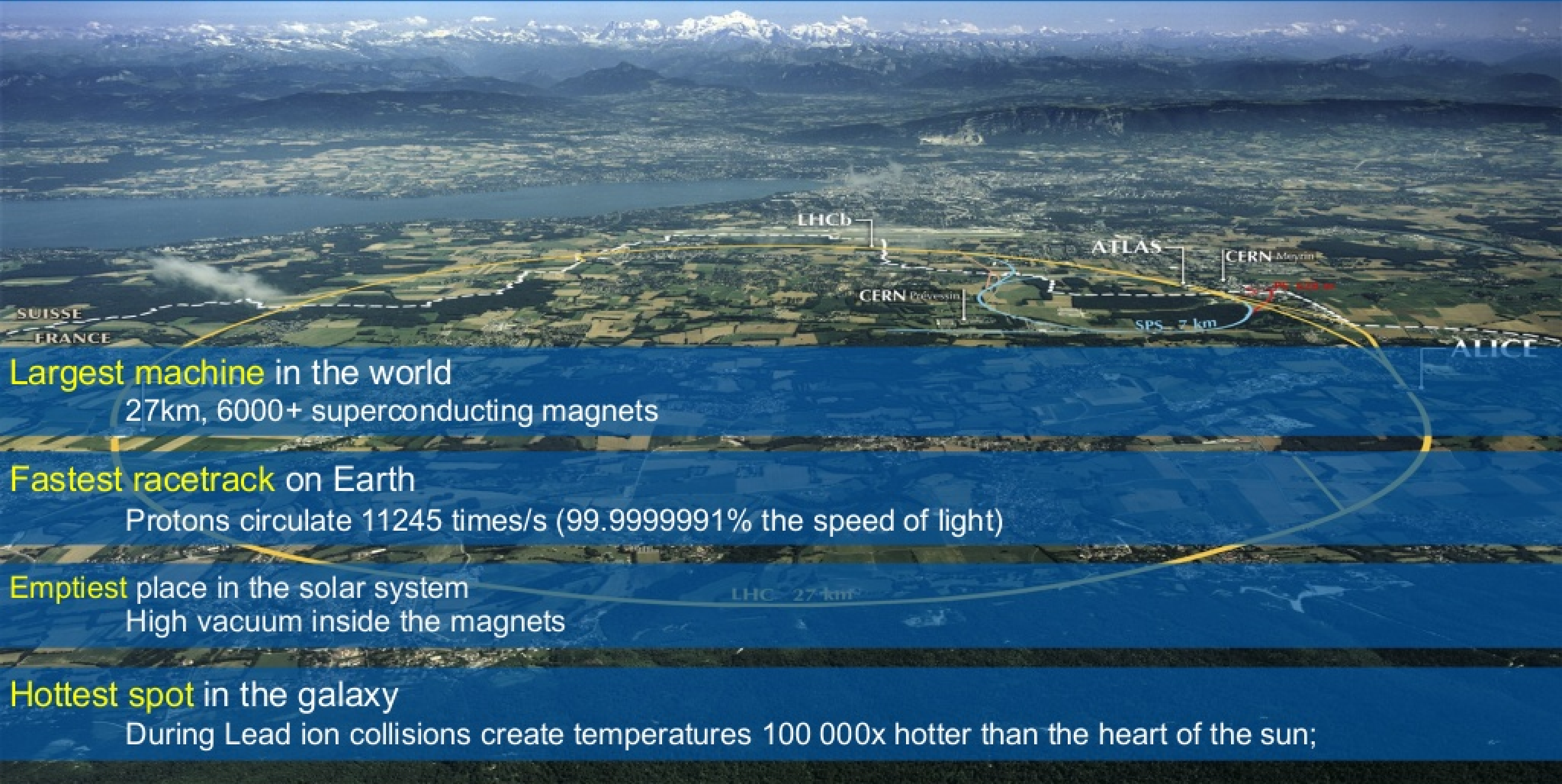




Distribution of All CERN Users by Nationality on 24 January 2018



The Large Hadron Collider (LHC)



Largest machine in the world

27km, 6000+ superconducting magnets

Fastest racetrack on Earth

Protons circulate 11245 times/s (99.9999991% the speed of light)

Emptiest place in the solar system

High vacuum inside the magnets

Hottest spot in the galaxy

During Lead ion collisions create temperatures 100 000x hotter than the heart of the sun;

CERN Data Centre

- Physics data are aggregated in the CERN Data Centre, where initial data reconstruction of physics events is performed
- A remote extension of the CERN data centre is hosted in Budapest, Hungary. It provides the extra computing power required to cover CERN's needs.

~ 12.5 PB per month

~ 2 PB accessed every day

~ 4 megawatt of electricity

	Meyrin Data Centre	Wigner Extension	TOTAL
Servers	11 500	3 500	15 000
Processor cores	174 300	56 000	230 300
Disks	61 900	29 700	91 600 (280 PB capacity)
Tape Cartridges			32 200 (~ 400 PB capacity)

Data at Scale @ CERN

- **Physics data** – Today we use WLCG to handle it
  - Optimised for physics analysis and concurrent access
  - ROOT framework - custom software and data format
  - Early stage experimental work ongoing to use Spark for physics analysis

- **Infrastructure data**
  - Accelerators and detector controllers
  - Experiments Data catalogues (collisions, files etc.)
  - Monitoring of the WLCG and CERN data centres
  - Systems logs

Apache Spark @ CERN

- **Current state-of-the-art**
  - Spark running on top of YARN/HDFS. Typically processing happens on the same cluster of machines as storage (data locality)
  - In total **~1850 physical cores** and 15 PB capacity

Cluster Name	Configuration	Software Version
Accelerator logging	20 nodes (Cores 480, Mem - 8 TB, Storage – 5 PB, 96GB in SSD)	Spark 2.2.0 – 2.3.1
General Purpose	48 nodes (Cores – 892,Mem – 7.5TB,Storage – 6 PB)	Spark 2.2.0 – 2.3.1
Development cluster	14 nodes (Cores – 196,Mem – 768GB,Storage – 2.15 PB)	Spark 2.2.0 – 2.3.1
ATLAS Event Index	18 nodes (Cores – 288,Mem – 912GB,Storage – 1.29 PB)	Spark 2.2.0 – 2.3.1

Apache Spark @ CERN

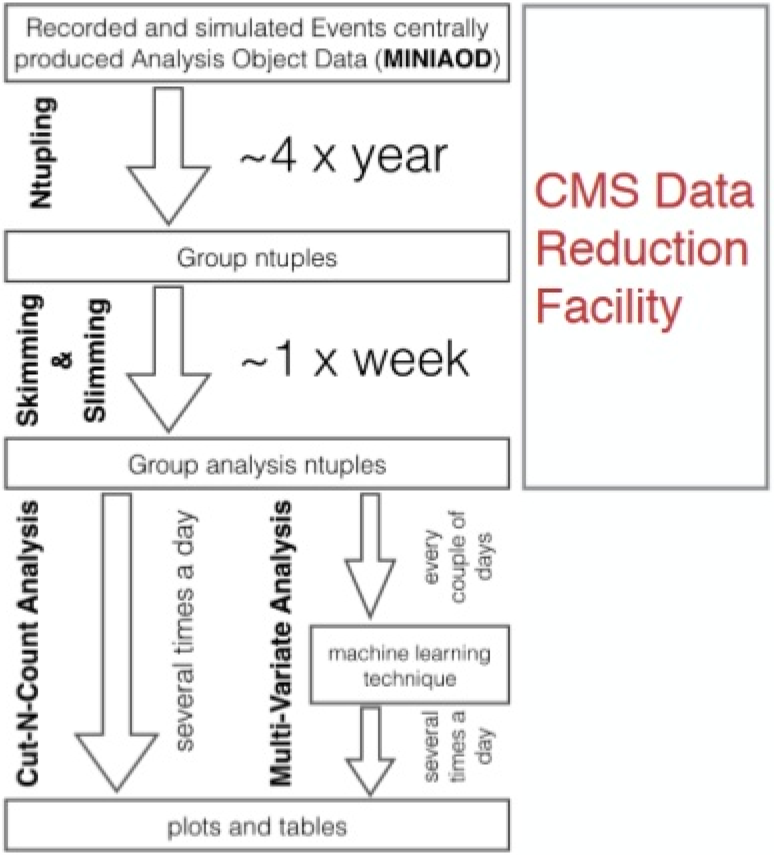
- **Current state-of-the-art**
  - Stable and predictable production workloads from our user communities
  - Physical machines allocated – means no resource elasticity, no isolation of workloads, compute coupled with data storage
  - Ad-hoc workloads from physics users interested analyzing data at scale using external storages (EOS). Physics analysis limited to the capacity and resources of shared Spark/Hadoop clusters.

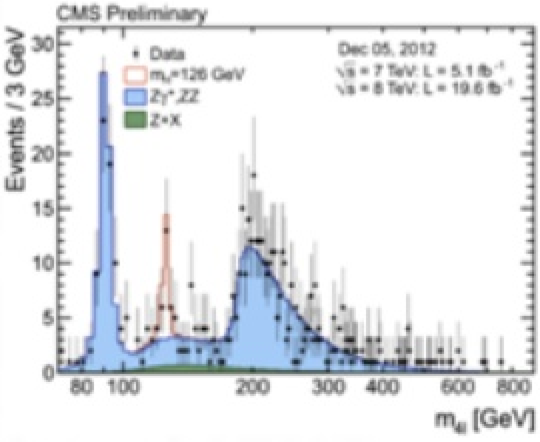
New Programming / Analysis Model for Physics data
- Data and Operational Challenges

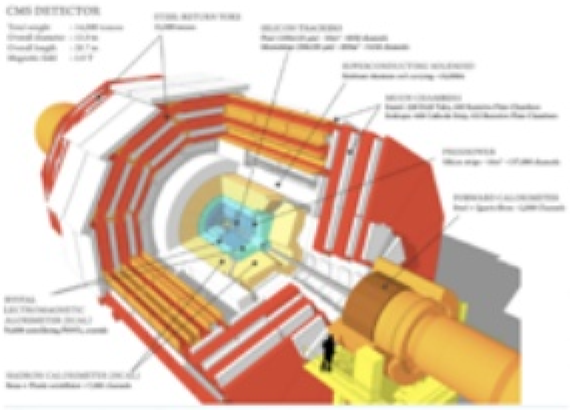
Data and Operational Challenges

- Data volumes expected to grow dramatically with HL-LHC
  - Annual growth at 20-30%
- Decrease time-to-physics
  - On-demand generation of physics n-tuples
- Resource Elasticity
- Isolation of workloads
- Physics data stored externally in EOS – CERN custom built data storage
- Usage of industry big-data solutions

CMS Data Reduction Facility







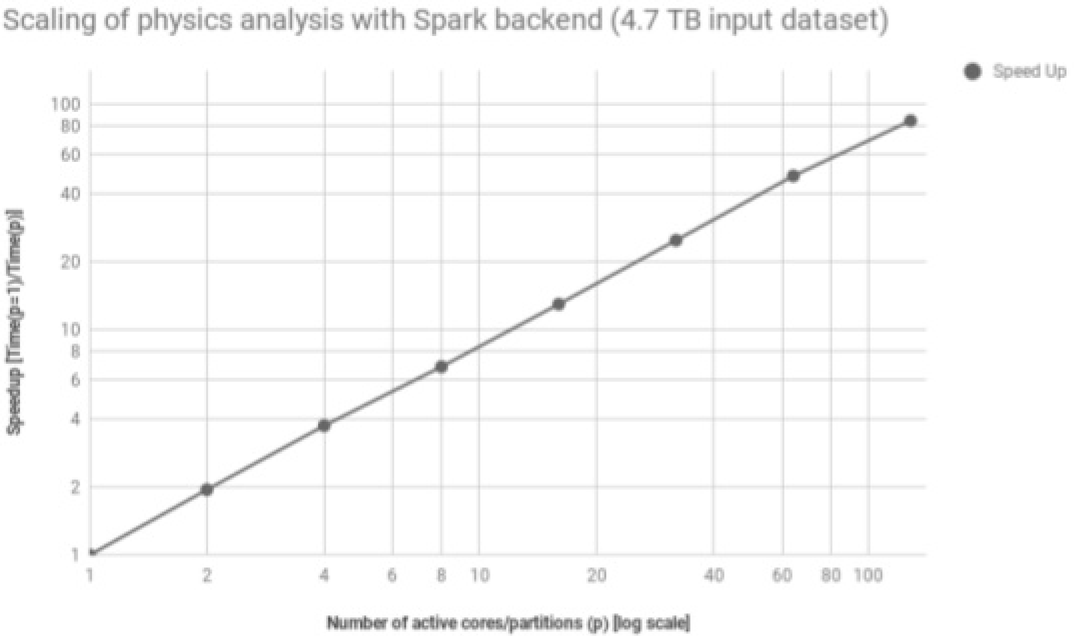
On-demand reduction of large datasets based on complicated user criteria

Input dataset ~ petabytes and requires 1000's cores

Reproducibility is key in the physics research world!

Interactive physics data analysis using notebooks

- To propose a time-efficient way to perform analysis of extensive amounts of data in CERN
- To investigate impact and usefulness of external solutions for HEP computing needs
- To prepare a ready model for future analyses performed in (TOTEM) experiments



Addressing the Challenges

Literature Study

„Cloud-native is an approach to building and running applications that exploits the advantages of the cloud computing model" [1]

Companies attempted cloud-native Spark deployments using Kubernetes Native resource scheduler implementation had experimental release in Apache Spark in March 2018.

Research from Google defends the hypothesis, that in cluster computing disk-locality is becoming irrelevant nowadays [2]

Databricks and Accenture Labs benchmarked Spark with data in external storages (S3/GCS) compared to HDFS. [3][4]

[1] „Kubernetes becomes the first project to graduate from the cloud native computing foundation."
[2] „Disk-Locality in Datacenter Computing Considered Irrelevant"
[3] Databricks - „Top 5 Reasons for Choosing S3 over HDFS"
[4] Accenture Technology Labs - „Cloud-based Hadoop Deployments: Benefits and Considerations"

Separating Compute and Storage for Elastic Big Data

Data Locality

Spark

HDFS

Spark/YARN

Data External (network)

External Storage (EOS, S3, GCS + Kafka, HDFS)

Spark

Spark/YARN

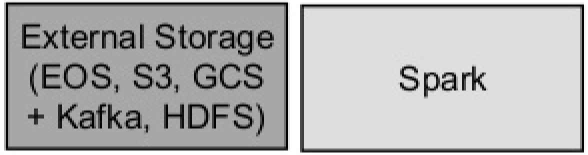
Spark on Kubernetes

Separating Compute and Storage for Elastic Big Data

Data Locality

- assumes **disk bandwidth is higher** than **storage throughput over network**

Data External (network)



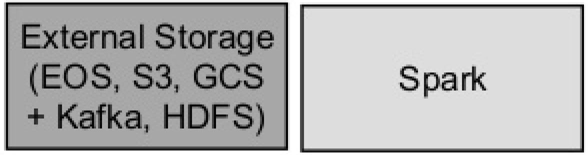
- assumes mass **storage throughput higher than** cluster **disk bandwidth**

Separating Compute and Storage for Elastic Big Data

Data Locality

- assumes **disk bandwidth is higher** than **storage throughput over network**
- **limited elasticity** (both for computation and storage)

Data External (network)



- assumes mass **storage throughput higher than** cluster **disk bandwidth**
- allows **compute not** being **bound** to **storage** (with cloud elasticity)

Separating Compute and Storage for Elastic Big Data

Data Locality

Spark

HDFS

- assumes **disk bandwidth is higher** than **storage throughput over network**
- **limited elasticity** (both for computation and storage)
- **avoids high network bandwidth** for analysis on large datasets

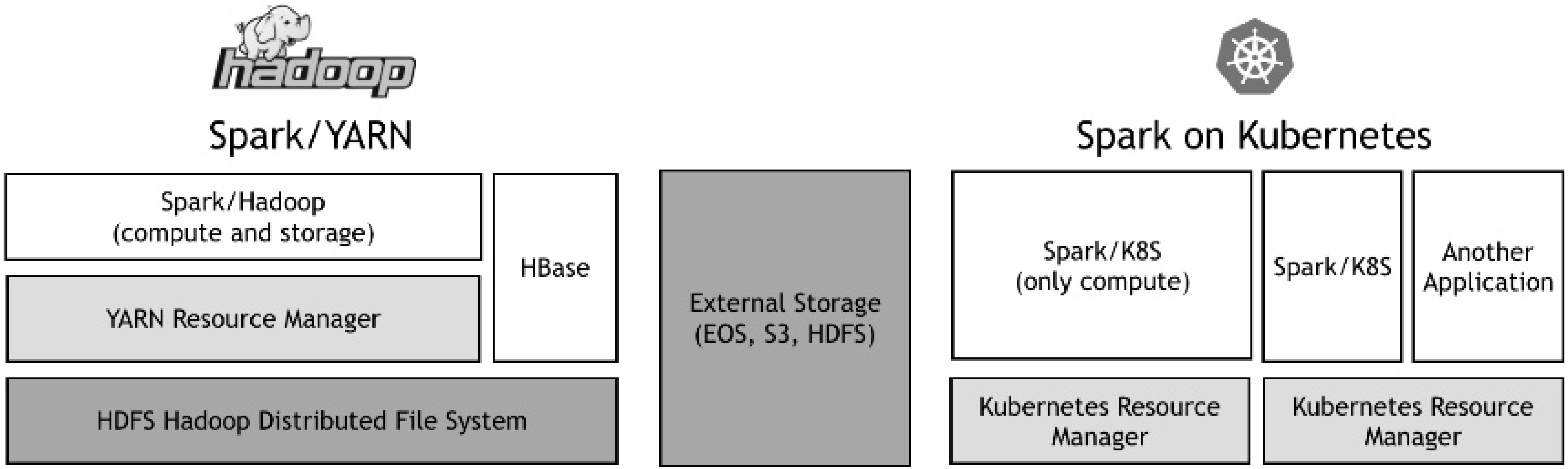
Data External (network)

External Storage (EOS, S3, GCS + Kafka, HDFS)

Spark

- assumes mass **storage throughput higher than** cluster **disk bandwidth**
- allows **compute not** being **bound** to **storage** (with cloud elasticity)
- assumes **network** delivers data **to CPU** faster than **local disks**, might generate **high traffic**

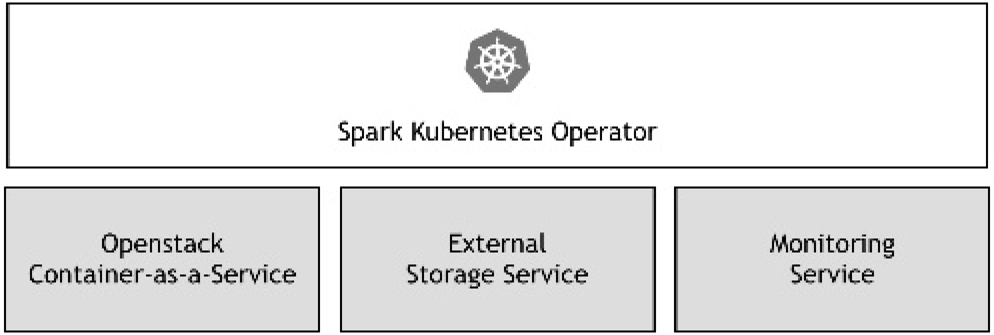
Possible solution for storage elasticity



- **mass storage** services are **easier/cheaper to scale**
- **data stored on disk** can be **large**, and **compute** nodes can be **adjusted to compute needs**

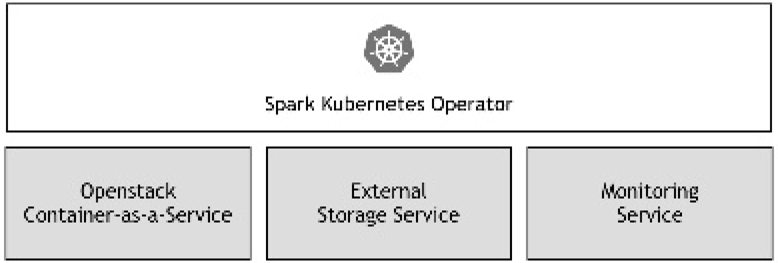
Possible solution for storage and compute elasticity and reproducibility

- Openstack CCE (**Cloud Container Engine)** with **Kubernetes** provides **compute elasticity** and **authentication** mechanism (certificates).



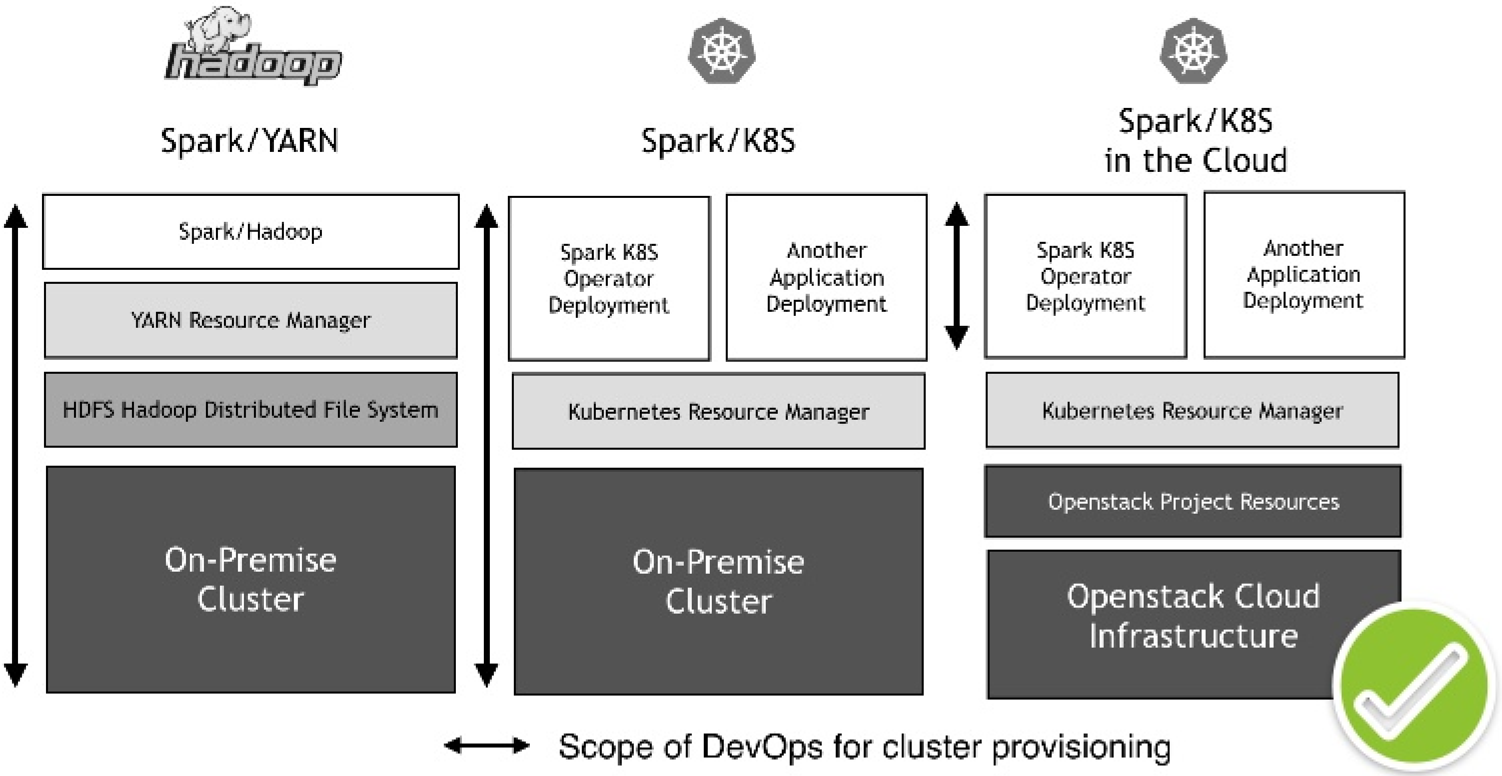
Possible solution for storage and compute elasticity and reproducibility

- Openstack CCE (**Cloud Container Engine)** with **Kubernetes** provides **compute elasticity** and **authentication** mechanism (certificates).
- **Separation** of monolithic **services simplifies operational effort**
- Just **Spark-Submit with Kubernetes** only allows to create drivers and executors (not so friendly to manage).
- **Spark K8S Operator** bring feature parity known from YARN (managing Spark Applications)
- Spark Operator gives **idiomatic submission**, **application restart policies**, **cron support**, **custom volume/config/secret mounts**, **pod affinity/anti-affinity**



[1] https://github.com/GoogleCloudPlatform/spark-on-k8s-operator

Container as a service to automate deployment



Evaluating Spark on Kubernetes properties and demo

Compute Elasticity

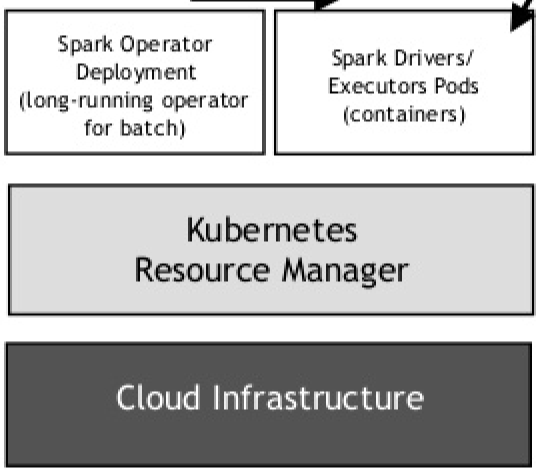
Cloud Container Engine

sparkctl
Submit batch/ stream jobs to K8S via Operator (cluster-mode)

spark-submit
Create driver on behalf of user (customised by operator)

Container Registry
Fetch containers with required software

Helm/Kubectl
Create/Delete/Update Spark Operator to control batch/stream jobs

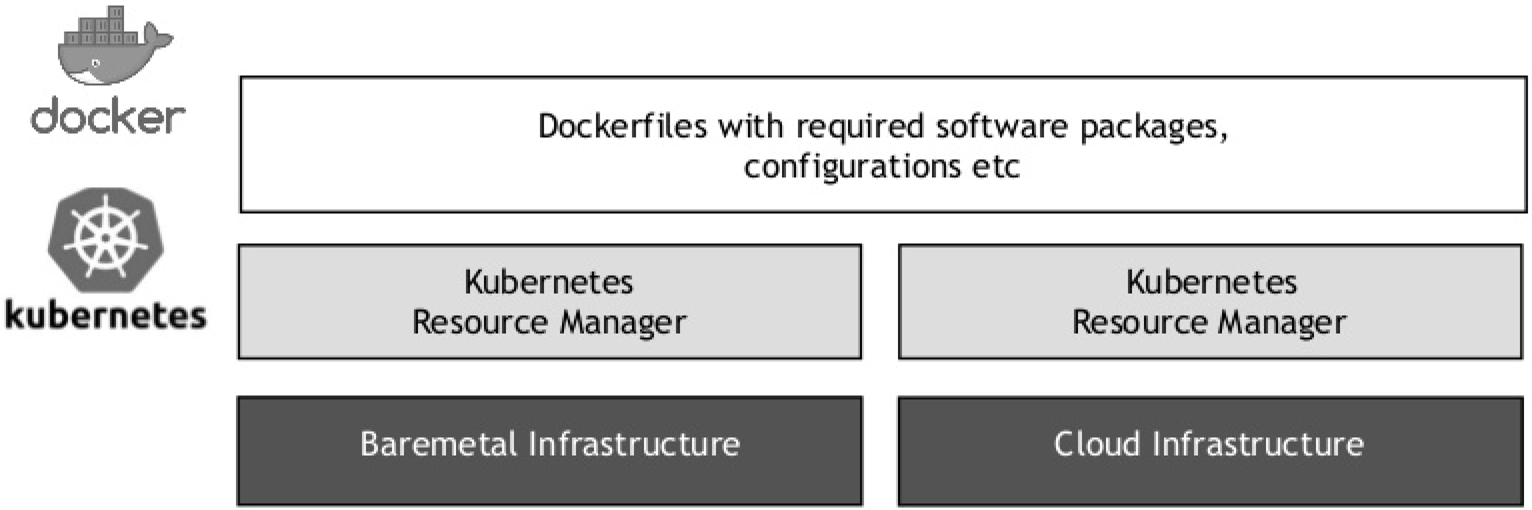


openstack coe cluster
Create/Resize/Delete/GetConfiguration for Kubernetes clusters

Workflow Reproducibility

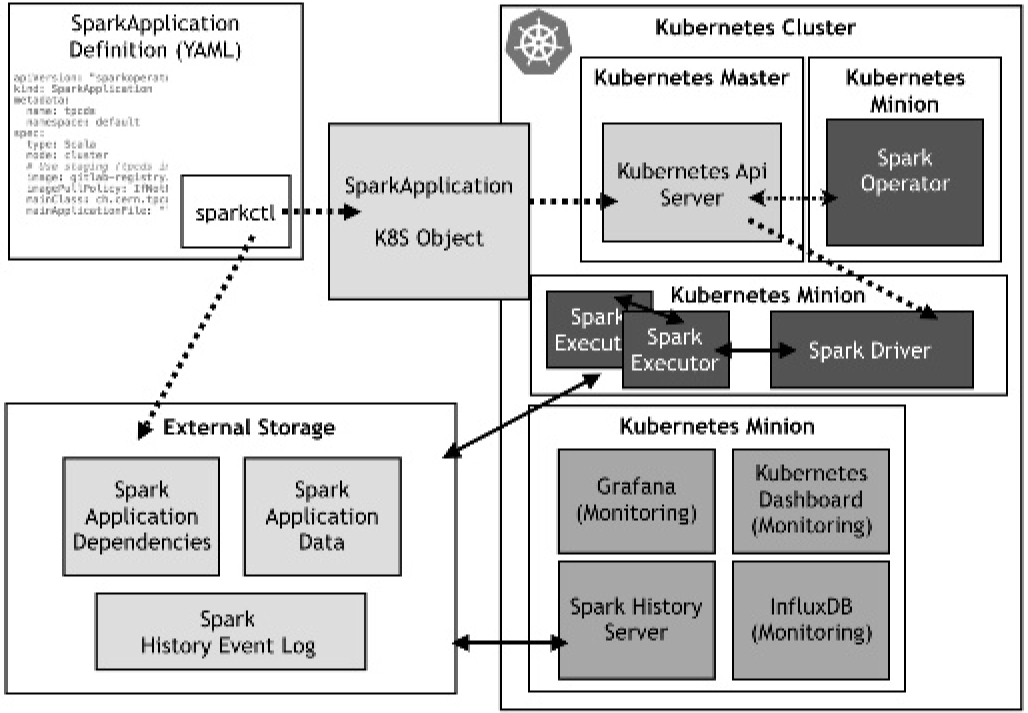
Docker containers

- Kubernetes and use of containers allows Spark developers to build their **isolated** and **reproducible environments**
- Encapsulate software packages, libraries, configurations



Management of Spark Applications

Kubernetes Operator for Apache Spark

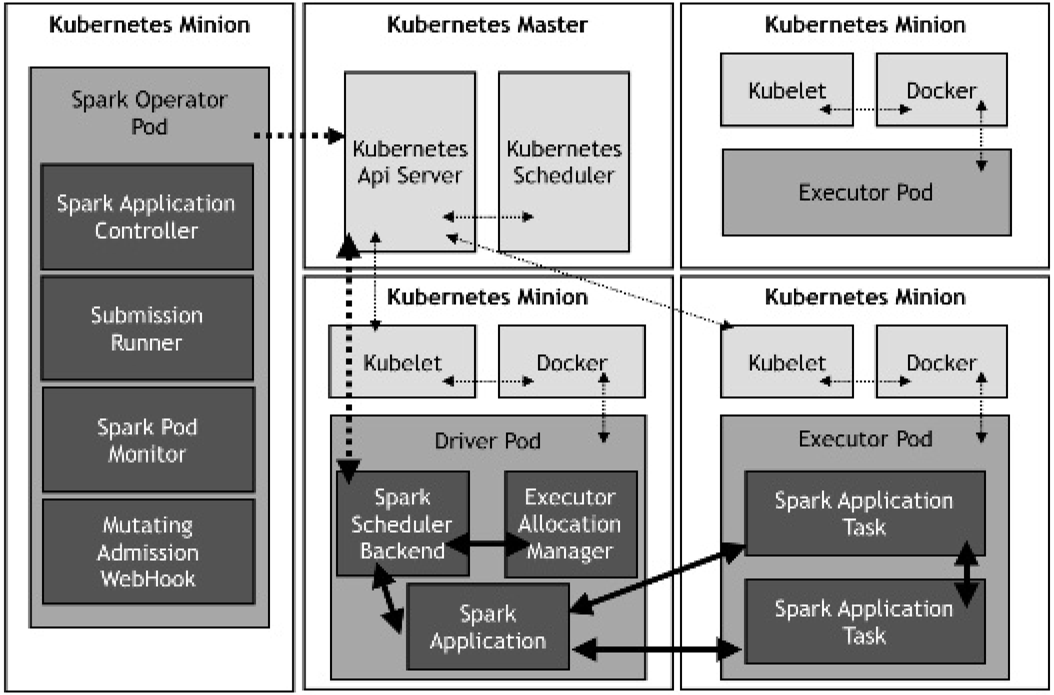


- Operator watches for create/delete/update events of SparkApplication
- Spark Operator executes spark-submit with required configurations
- Data is uploaded/read from external storage service, and monitoring is realised with internal/external monitoring tools

https://github.com/GoogleCloudPlatform/spark-on-k8s-operator

Management of Spark Applications

Kubernetes Operator for Apache Spark



- **Spark Application Controller** handles restarts and resubmissions.
- **Submission Runner** executes spark-submit
- **Spark Pod Monitor** reports updates of pods to controller
- **Mutating Admission WebHook** handles customisation of Docker containers and their affinities

https://github.com/GoogleCloudPlatform/spark-on-k8s-operator

Persistent Storage for cloud-native

Data over network - available solutions with CERN Openstack Cloud

Persistent Storage Feature	Network Volume Mounts	Filesystem Connectors	Object Storage Connectors	Monitoring Storage
Example	**CephFS [1], CVMFS [2],** other [3]	**EOS/XRootD [4],** HDFS external, no data locality	**Ceph S3**, GCS [5]	**InfluxDB/Grafana**, Stackdriver, Prometheus
Use-case	**Software Packages, Checkpoints, Events (History Server)**	**Data processing**, Events, Checkpoints	**Data processing, Events (History Server), Checkpoints**	**Statistics**
Spark Transactional Writes	-	Yes	Requires committer (Directory Committer Hadoop 3.1) [6]	-

[1] https://clouddocs.web.cern.ch/clouddocs/containers/tutorials/cephfs.html

[2] https://clouddocs.web.cern.ch/clouddocs/containers/tutorials/cvmfs.html

[3] https://kubernetes.io/docs/concepts/storage/storage-classes

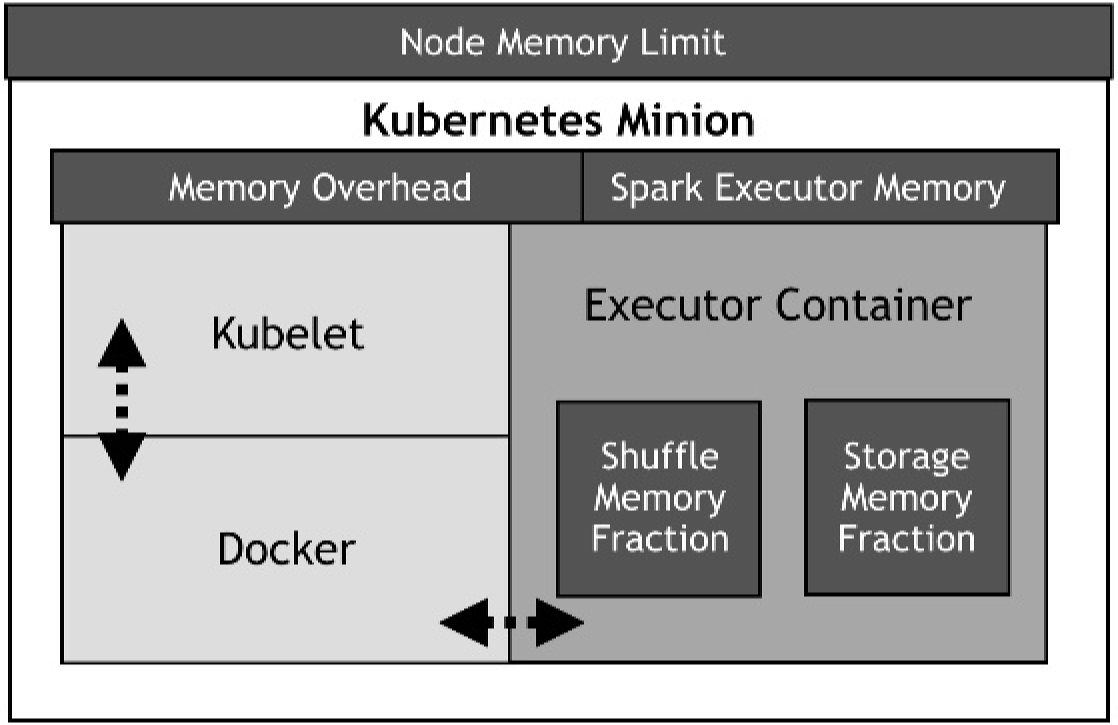
[4] https://github.com/cern-eos/eos, https://github.com/cerndb/hadoop-xrootd

[5] https://cloud.google.com/dataproc/docs/concepts/connectors/cloud-storage

[6] http://hadoop.apache.org/docs/r3.1.1/hadoop-aws/tools/hadoop-aws/committers.html

Memory management in Kubernetes

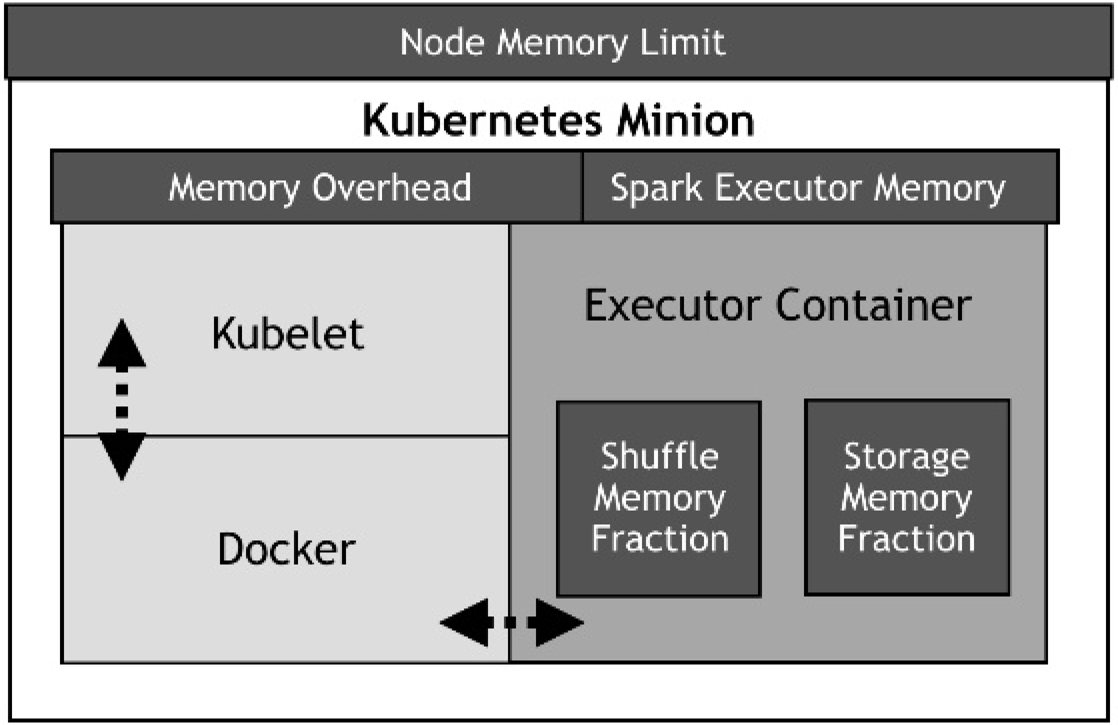
Spark and Kubernetes Nodes



- **Kubelet** monitors **memory** and **disk** available to the Node.
- Detecting **MemoryPressure** or **System Out Of Memory** and **reclaiming resources** from running containers **(OOMKilled errors)**

Memory management in Kubernetes

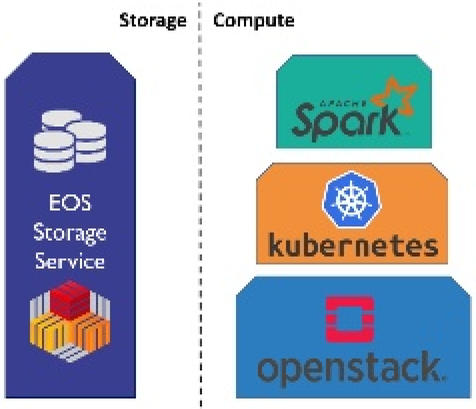
Spark and Kubernetes Nodes

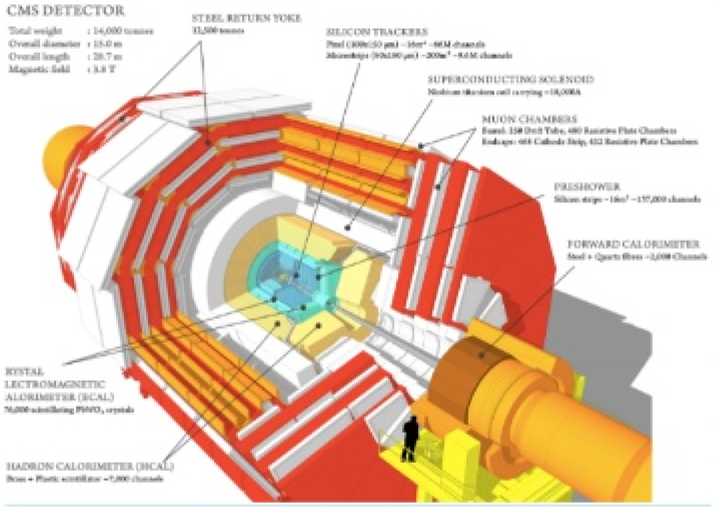


- **Kubelet** monitors **memory** and **disk** available to the Node.
- Detecting **MemoryPressure** or **System Out Of Memory** and **reclaiming resources** from running containers **(OOMKilled errors)**
- **MemoryOverheadFactor** - memory for **off-heap memory, non-JVM processes** (e.g. in case of Python higher limit) and processes required for **operation of container**.

Scaling Spark on Kubernetes

Data Reduction and Dimuon Mass Calculation on 20TB (target is 1 PB dataset)

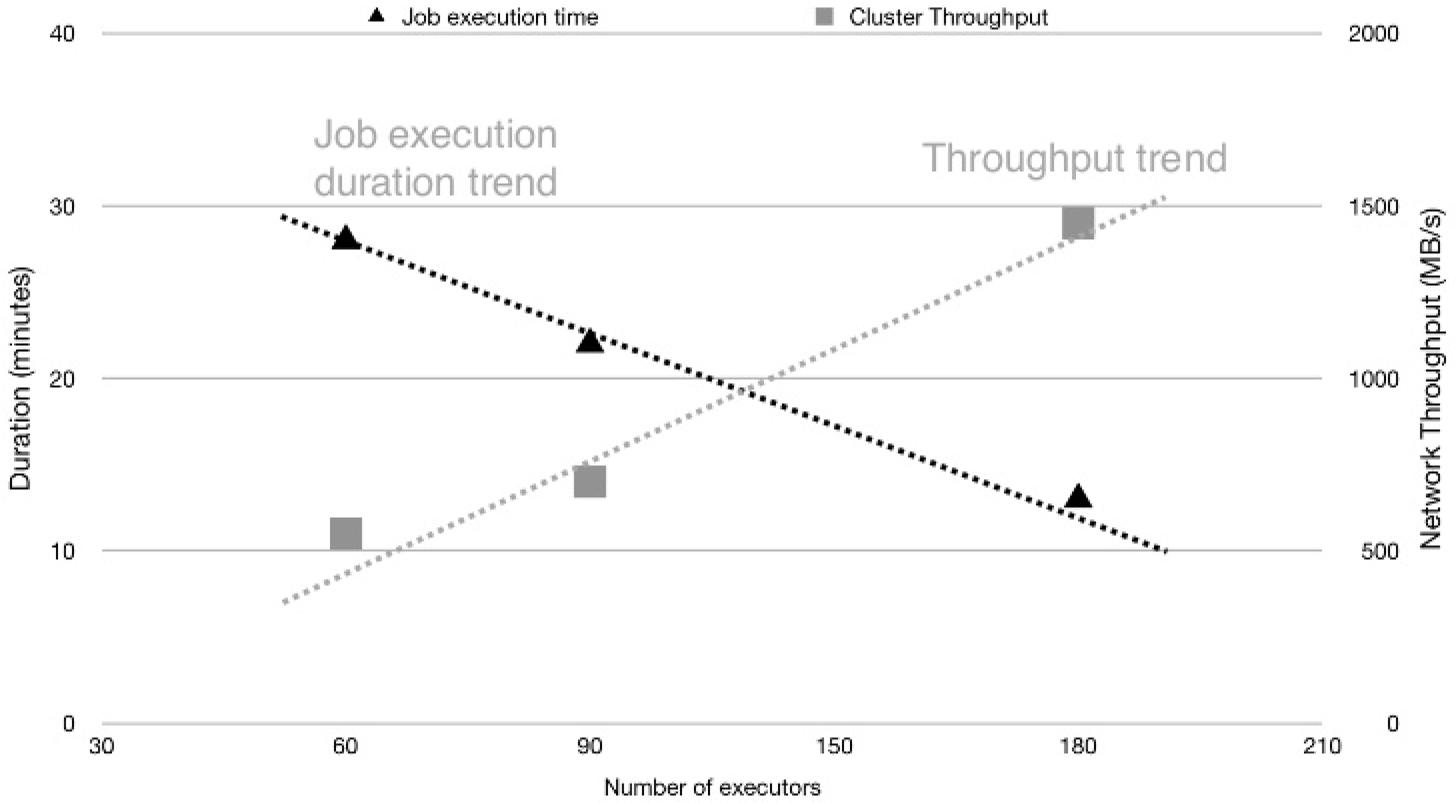




- **1 VM** is **2CPU** and **12GB RAM**
- **Scaling Test** with 60 to 180 VMs
- **Load Test** with 500 VMs (200 hypervisors)

Scaling Spark on Kubernetes

Data Reduction and Dimuon Mass Calculation on 20TB (target is 1 PB dataset)



Spark Operator Demo

Create cluster

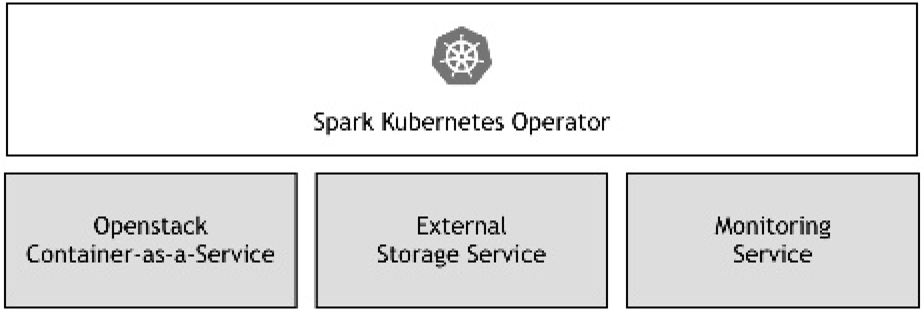
https://youtu.be/vuSLS7-JqQI

Conclusions and observations

- **Openstack provisions Kubernetes cluster** of 10s of nodes in automated fashion (**Cloud Container Engine,** Cloud Horizontal Autoscalers optional).

- **Kubernetes** allows **isolated** and **reproducible Spark, limited operational effort** with Docker containers.

- **Spark K8S Operator** provides management of Spark Applications similar to YARN ecosystem

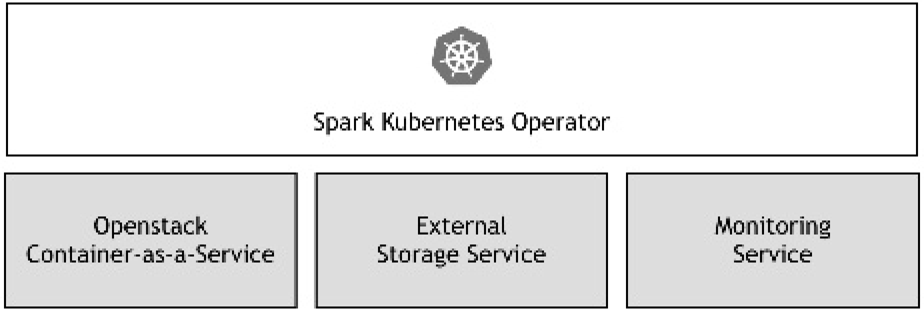


Conclusions and observations

- **Openstack provisions Kubernetes cluster** of 10s of nodes in automated fashion (**Cloud Container Engine,** Cloud Horizontal Autoscalers optional).

- **Kubernetes** allows **isolated** and **reproducible Spark, limited operational effort** with Docker containers.

- **Spark K8S Operator** provides management of Spark Applications similar to YARN ecosystem



- **Data** is **external,** this has advantages but also drawbacks - **do you need data locality and why?**

- **Compute** cluster is **managed service**, storage can **scale separately** from **compute** or in the cloud.

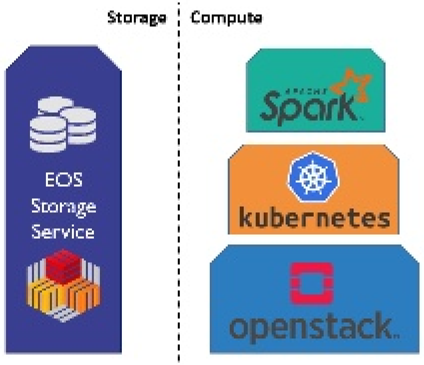
- **Storage interoperability** (CephFS, EOS, S3 and GCS can serve more use-cases than just Hadoop)

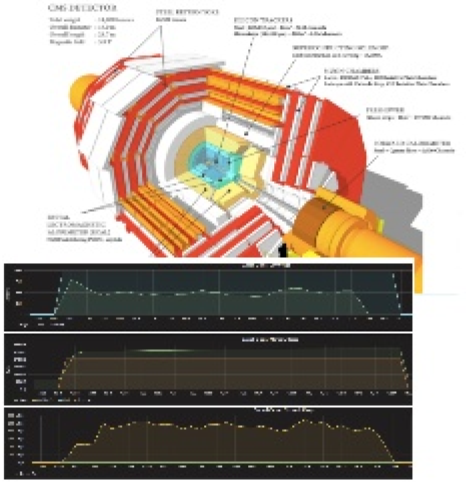
Conclusions and observations

- **Without data locality**, **network** can be a **serious problem/bottleneck** (specifically in case of over-tuning or bugs). Monitoring at hand.

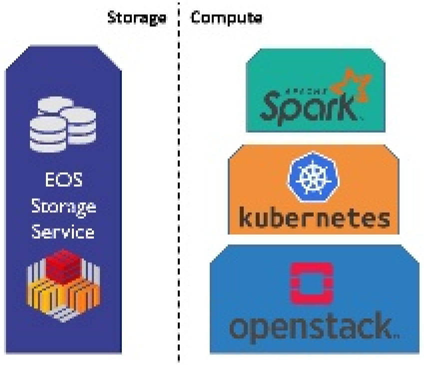
- Large shuffle writes problematic, **assumed compute VMs** and **large storage** space **is not available**. On other hand **SSD can be used cheaply**.

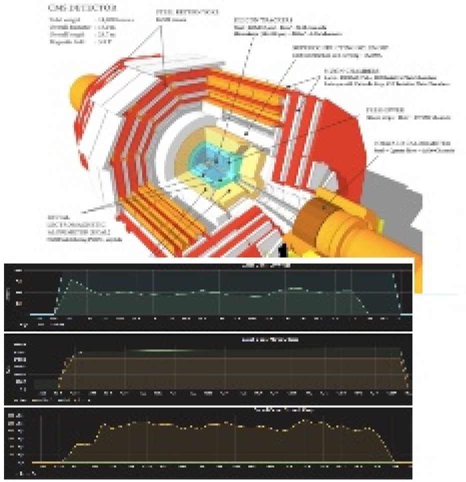
- **Multi-tenancy** still a question.





Conclusions and observations





- **Without data locality**, **network** can be a **serious problem/bottleneck** (specifically in case of over-tuning or bugs). Monitoring at hand.
- Large shuffle writes problematic, **assumed compute VMs** and **large storage** space **is not available**. On other hand **SSD can be used cheaply**.
- **Multi-tenancy** still a question.

- No particular overhead between **K8S VMs** and **bare-metal YARN** with TPCDS Benchmark on similar hardware. Production YARN had more spikes than Cloud K8S due to shared environment.
- **Spark/YARN** had ~**40 nodes fixed**, **Spark/K8S** 500 VMs (**~200 nodes**) provisioned for **short time** period

Thank you

Questions?