SESAME

Vi-SEEM

OBERA

Opportunistic Broker for Elastic Resource Allocation

Empowering Pilot-abstractions of Scientific Applications over Data-intensive Clouds and Cyber-infrastructure

By:

Feras Mahmoud Awaysheh

HTCondor
High Throughput Computing

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

CAMPUS VIDA
CAMPUS DE EXCELENCIA INTERNACIONAL

CiTIUS
Centro Singular de Investigación
en Tecnoloxías da
Información

Bio…

Education & Background

- Jordanian Researcher
- BSc BAU, Jordan
- MSc NyIT, USA
- PhD USC, Spain (CiTIUS)

Research interest

- Large-scale Distributed Clusters
- Big Data Architectures
- High-performance Computing
- Data-intensive Applications/ Clouds

Collaborations

- Vi-SEEM
- Prof. Sherif Sakir Germany
- Blesson Varghese United Kingdom

Publications & other stuff

- EME (a MapReduce use case) at CLOSER
- BD deployment architectures at ACM CSUR
- Writing for Elsevier BD & FGCS
- Reviewer and board membership for several Journals and conferences

Good enough isn't good enough if it can be better and better isn't good enough if it can be best!

Feras M. Awaysheh

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

CAMPUS VIDA CAMPUS DE EXCELENCIA INTERNACIONAL

CiTIUS Centro Singular de Investigación en Tecnoloxías da Información

Vi-SEEM

SESAME

CiTIUS

Outlines

- Background
  - Scheduling Large-Scale Clusters (LSC)
  - Resource Management in Big Data (state-of-art)
- Challenges
  - Proposed solutions
- Introducing OBERA
  - What is it and What’s not
- OBERA architecture
- Use case - MapReduce
- Opportunities & Collaboration
- Future work

Scheduling Large-scale Clusters

- Goals:
  - Highest Utilization
  - Maintain ultimate efficiency
  - Scalable (whenever, however we want)
  - High fault tolerance

- Issues:
  - Un-predictable load
  - Increasing workload, clients and cluster size
  - Common delusion
    - Network reliable and homogeneous
    - Transport cost is zero

small data

big data

Resource Management Terminology

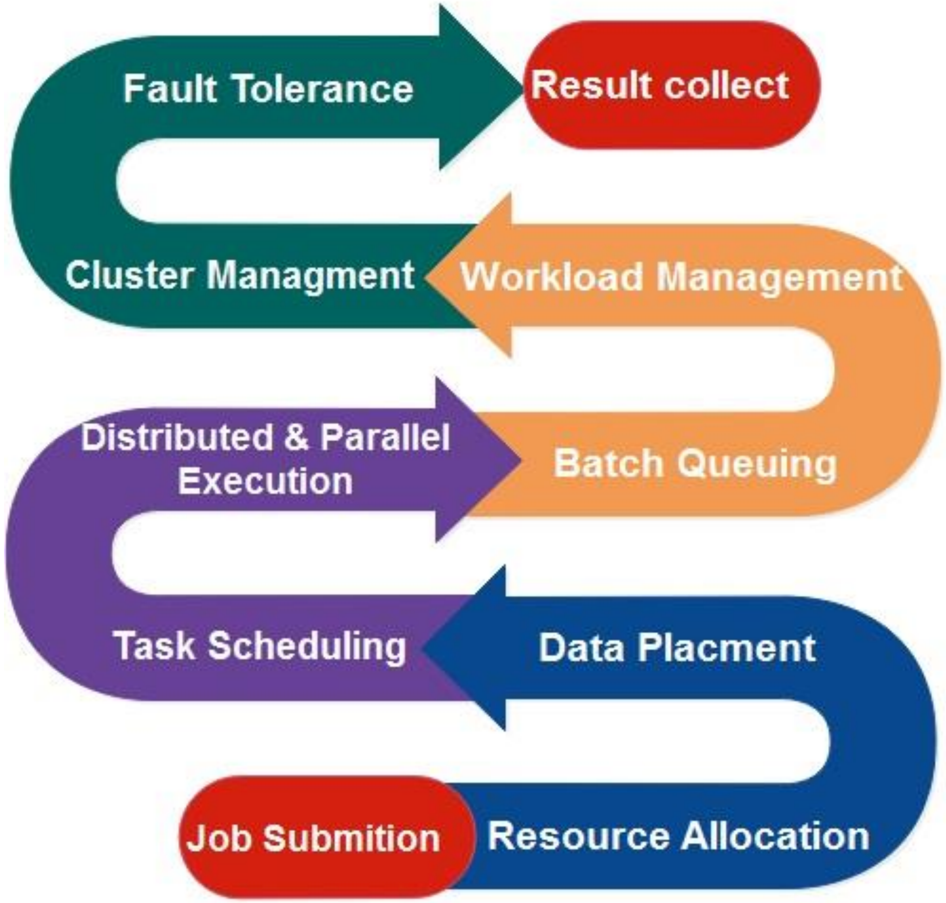
- Different cluster scheduler architectures.
  - Monolithic schedulers
  - Two-level schedulers
  - Shared-state schedulers
  - Hybrid solutions

- Hide the details so that the user focus on application development

- Maintain in high availability, reliability and support frameworks to do so

Open source resource management solution:

- Hortworks, Cloudera, MapR - YARN
- Apache Mesos and Myriad
- Container-based Clusters - Docker Swarm



LSDS Challenges

Low Utilization
= Higher cost

- The utilization problem…
  - Workloads tend to be bursty
- Multi-tenancy problem…
  - Virtualization impacts performance
  - Difficult to do short term borrowing of capacity
- Infrastructure silos…
  - No one size fit all
- Installing new infrastructure
  - Cost, cost and cost
- Fault tolerance, Failure management and security

Schedulers wish list!

- Applications request resources when they need them
  - Automated without user intervention
- Scale-out on demand to a free resources
  - Elastically provisioning
- Multi-tenancy with strong isolation
  - Sandbox with the required libraries etc,
- Minimal configuration
  - Updated/restarted without affecting current running tasks

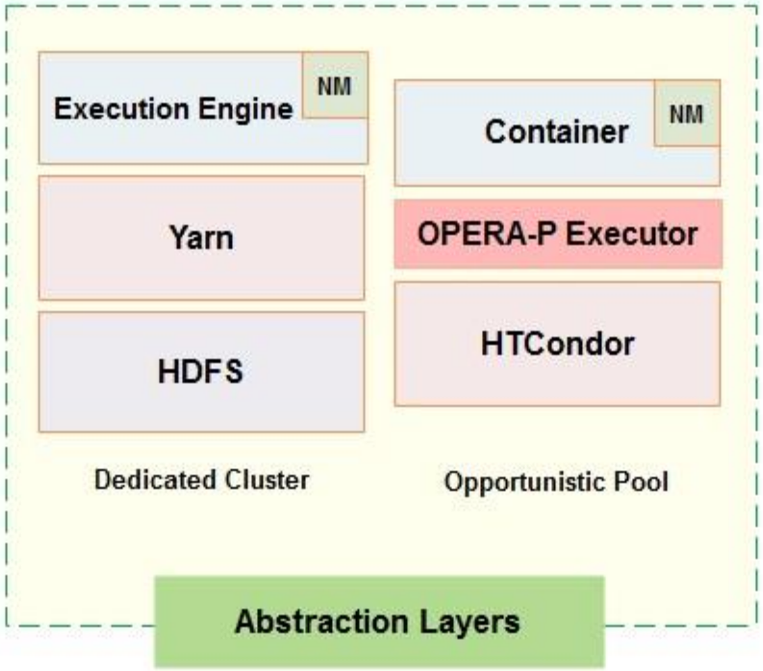
Current solutions

- Statically cluster sizing based on peak utilization
- Installing new infrastructure on-demand
  - Easy with Hadoop (scale-out)
  - Though, it’s not elastic or auto-scale technique
- Virtual Machines
  - High virtualization costs
  - VM licensing
  - Data movement issues
- New development environments
  - Adaptive and untraditional analytical environments

OBERA: BDaaS Orchestration

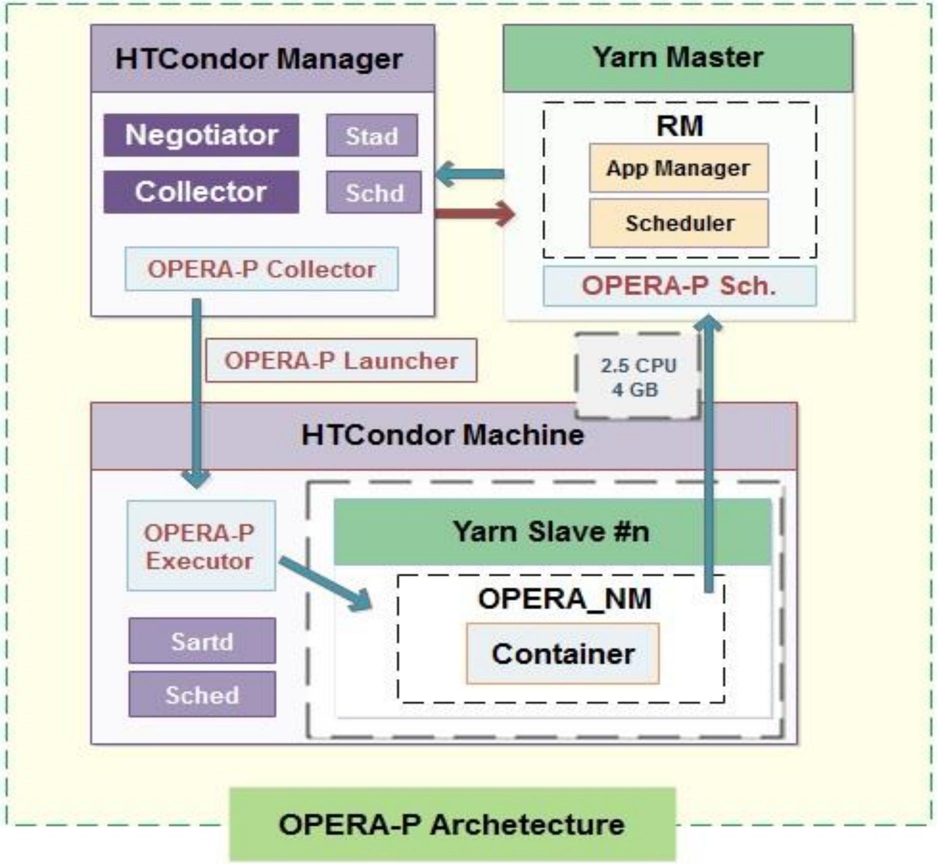
- A POD service that automatically and continuously spawns containers in a HTCondor pool
  - According to the available resources
- An opportunistically analytical environment
  - Runs as a standalone instance on each HTCondor machine
  - Represent a new CaaS service
    - Disposable pilot approach ➝ one job - one container
- This model means that a shared pool of resources can be shared among many frameworks/applications
  - Each capable of allocating additional resources elastically when needed and releasing them when not.

Platforms Provisioner



OBERA Architecture

- Resource management framework:
- second level
  - HTCondor offers resource
  - Framework schedulers accept or reject offered resource
- Lightly used resource allocation:
  - Thanks to HTCondor
  - Elastically provisioning needed frameworks on-demand
- Frameworks Integration
  - Hybrid analytical environment
  - Modify framework scheduler in the container to com- with Yarn master through its API



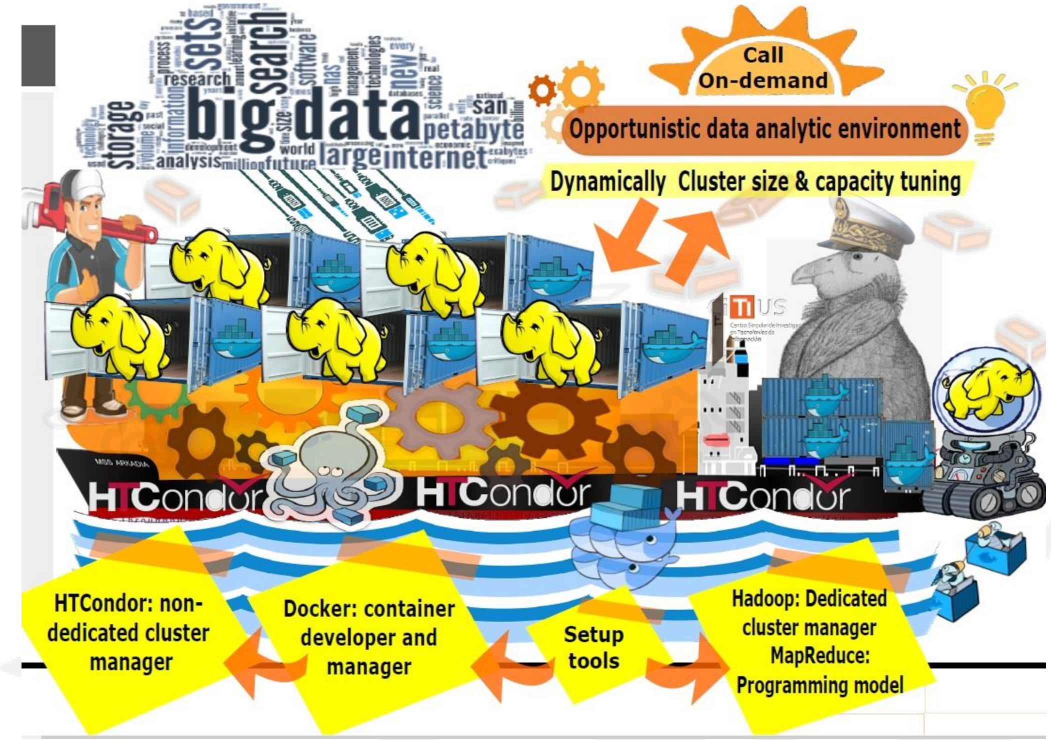
What and What's not OBERA

YES

- ✓ A provisioning service
- ✓ Workflow manager
- ✓ Opportunistic analytical platform

NOPE

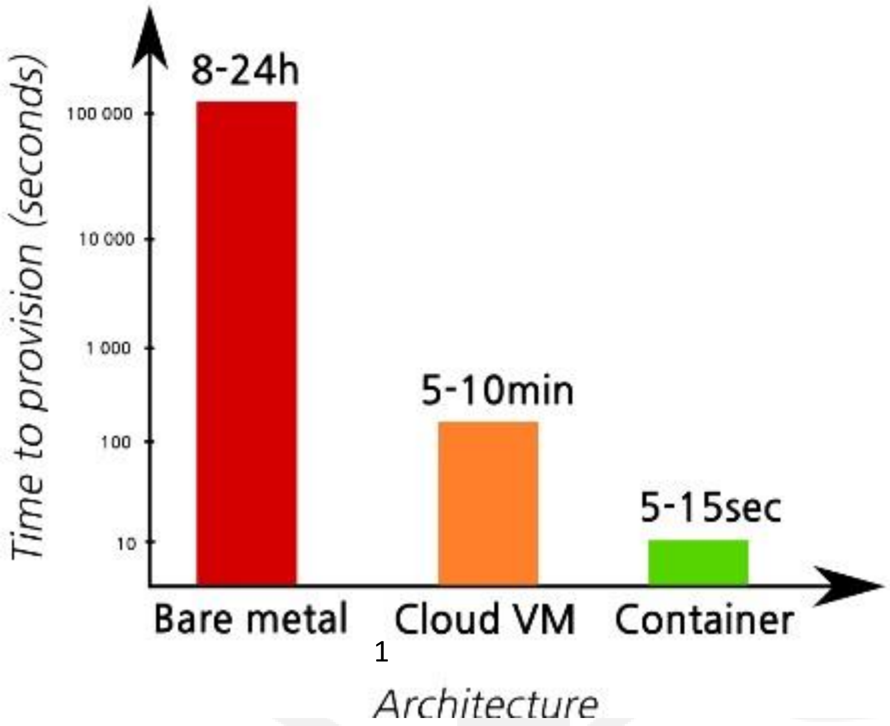
- ✗ Container scheduler
- ✗ Hadoop distribution
- ✗ MR implementation



big data
research
sets
search
storage
analysis
petabyte
internet
large
Call On-demand
Opportunistic data analytic environment
Dynamically Cluster size & capacity tuning
HTCondor
HTCondor
HTCondor
HTCondor: non-dedicated cluster manager
Docker: container developer and manager
Setup tools
Hadoop: Dedicated cluster manager MapReduce: Programming model

Design considerations

- Pilot abstraction
  - Beyond a traditional remote distributed processing
  - From static and dedicated resource to dynamic resource
- Light weight virtualization
  - Near Bare-metal
  - Thanks to Docker containers
- Resource capping and isolation
  - Workloads don't interfere with operational applications



1

[1] 1- Introduction to Apache Mesos: https://www.slideshare.net/tomasbart/introduction-to-apache-mesos

Fault tolerance

"design your system for failure"

- Every component must have redundancy
  - No single point of failure!
- Tuning the heartbeat
- Majority voting (result checking):
  - Leaving work-done flag until collect two out of three results
  - Directly enhance fault tolerance as well

CiTIUS

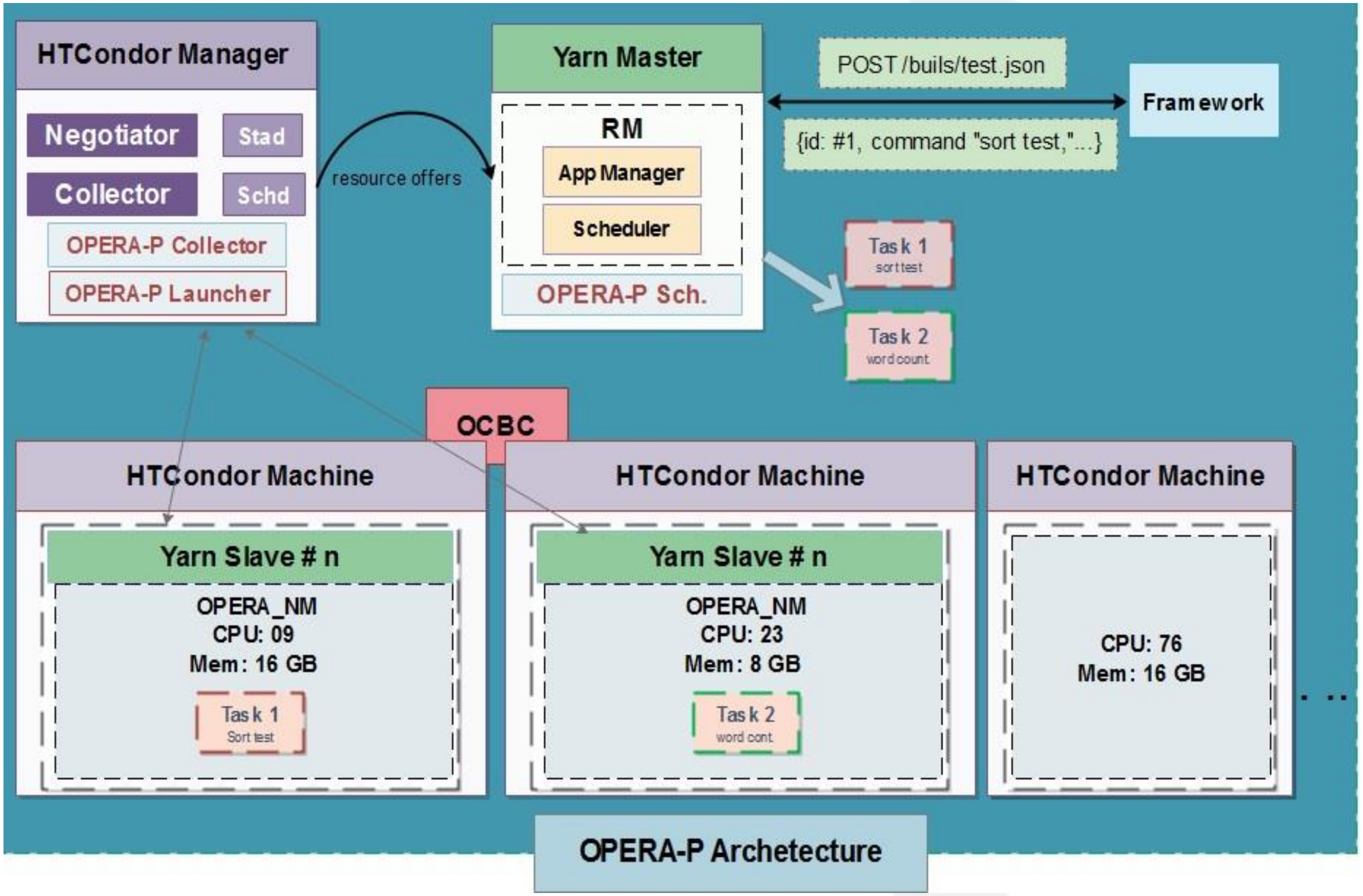
SESAME

Use case example

EME: An Enhanced Mapreduce Environment

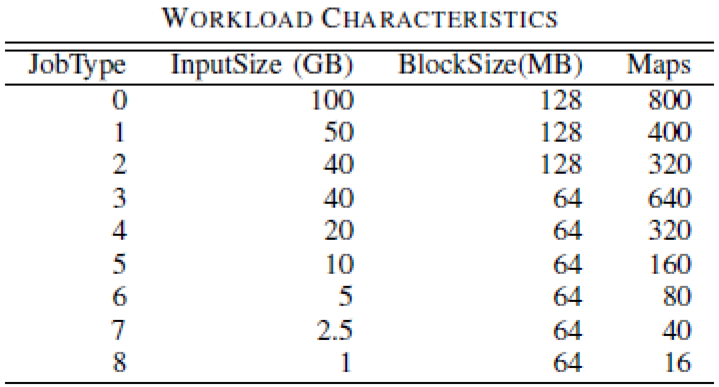
How it works

- Very similar to how multiple apps run concurrently on a laptop or smartphone
- New threads are spawned, and more resources are joining the Hadoop cluster as they are needed
- OBERA will match the request to incoming HTCondor resource offers and can then consume the resources as it sees fit
- HTCondor, in turn, will pass it on to its worker machines, and launches pilot containers among the underutilized nodes (idle workstations)



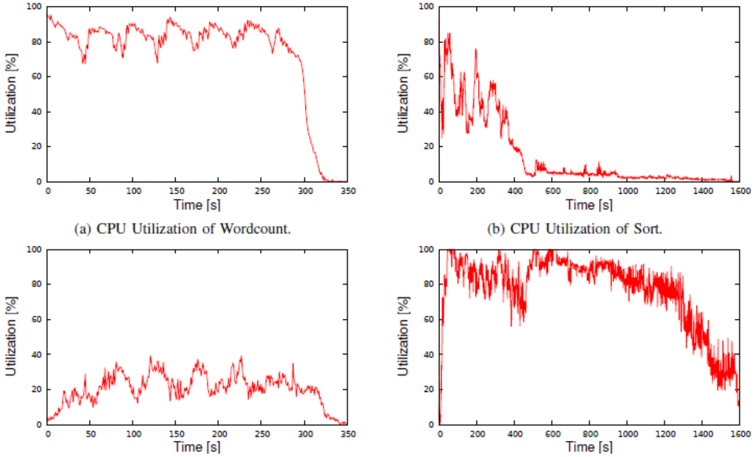
Validation and Results

Performance evaluation



WORKLOAD CHARACTERISTICS

JobType	InputSize (GB)	BlockSize(MB)	Maps
0	100	128	800
1	50	128	400
2	40	128	320
3	40	64	640
4	20	64	320
5	10	64	160
6	5	64	80
7	2.5	64	40
8	1	64	16

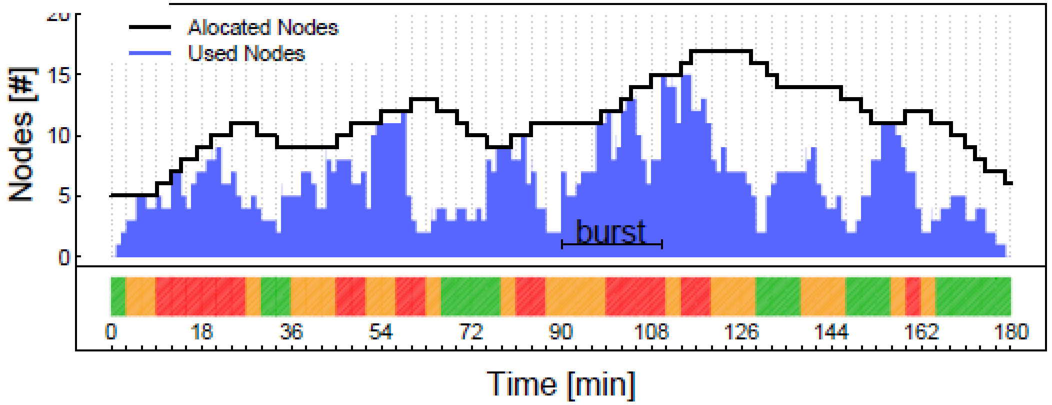


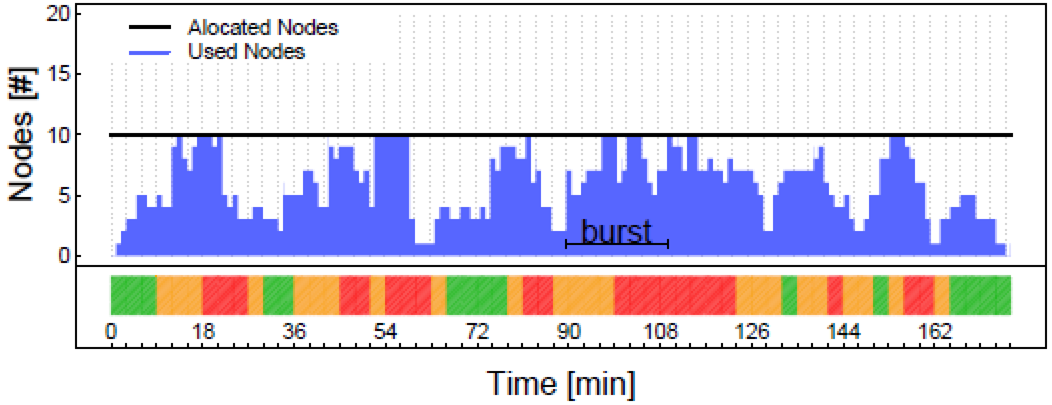
(a) CPU Utilization of Wordcount.

(b) CPU Utilization of Sort.

(c) Disk Utilization of Wordcount.

(d) Disk Utilization of Sort.





Data sets and benchmarking

Opportunities

- Exploit more than 2TB of RAM & 65PB HDD available resources at the CiTIUS
- Opportunistic Container-based Cluster (OCBC)
  - A new CaaS service
- A 3D models
  - Running dedicated only
  - Running Opportunistic only
  - Provisioning BD platforms on-demand
- Organizations can deploy, manage, and monitor their BD system, on both dedicated Hadoop cluster and opportunistic HTCondor pool as a single machine

Conclusion & Future

- OBERA is an enabling technology to take advantage of leveraging all of available resources within an enterprise or cloud as a single pool of resources
- OBERA provides a seamless bridge from the pool of resources available in HTCondor to the YARN tasks that want those resources.
- OBERA prototype can easily be adapted to other resource managers, e.g., Apache Mesos and Docker Swarm
- OBERA is an ongoing project, we start prototyping in a virtualized cluster and, when proving its usefulness, test it in a bare-metal environment.

Many ideas grow better when transplanted into another mind than the one where they sprang up. “Oliver Wendell Holmes”

Thank you for your attention

Unidad de Innovación:

https://citius.usc.es/equipo/investigadores-en-formacion/feras-awaysheh-mahmoud

feras.awaysheh@usc.es

HTCondor
High Throughput Computing

Sofía, Bulgaria
Mayó 16 2018

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Vi-SEEM

CiTIUS Centro Singular de Investigación en Tecnoloxías da Información

CAMPUS VIDA CAMPUS DE EXCELENCIA INTERNACIONAL

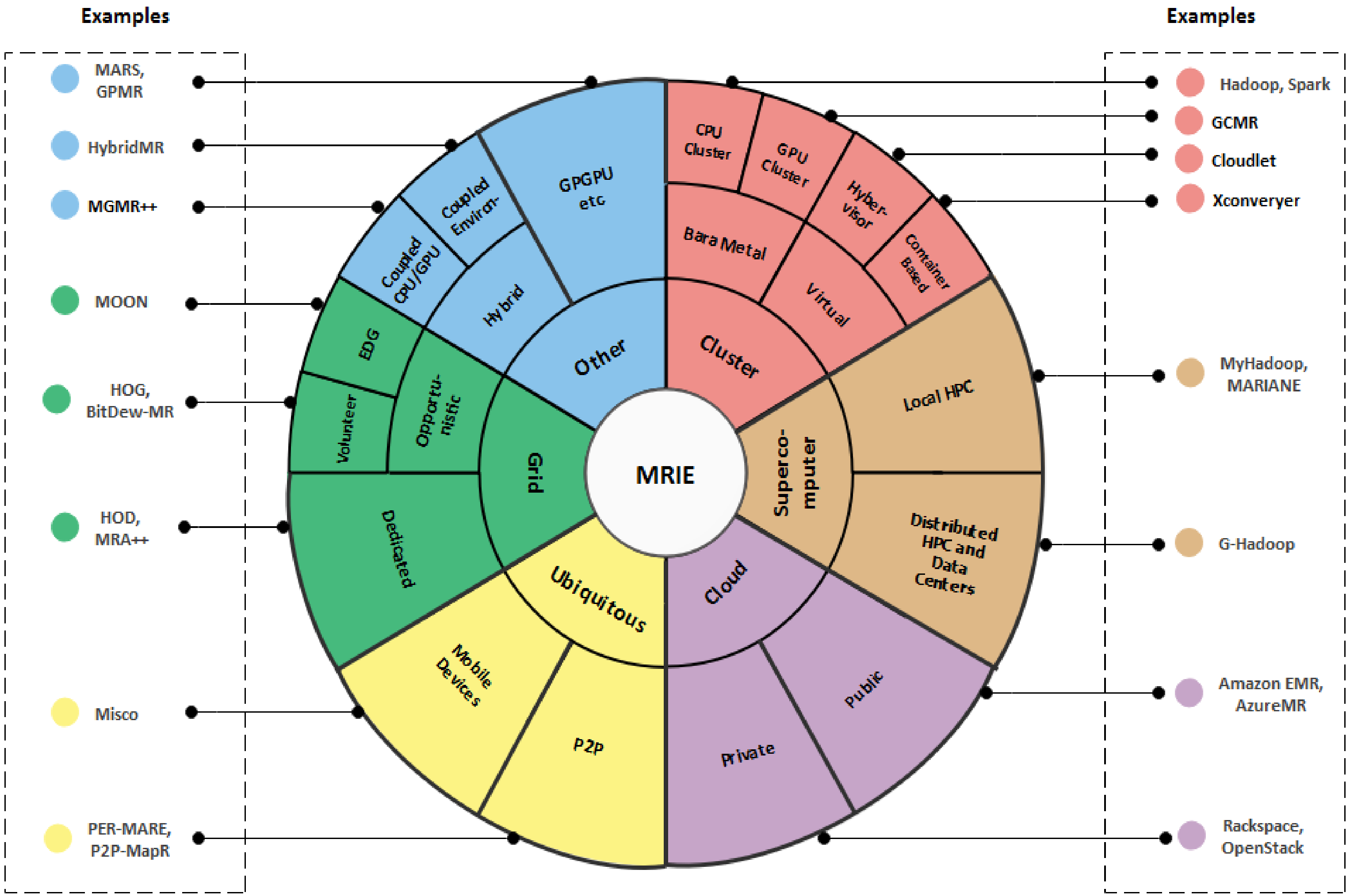
SESAME

UNIÓN EUROPEA

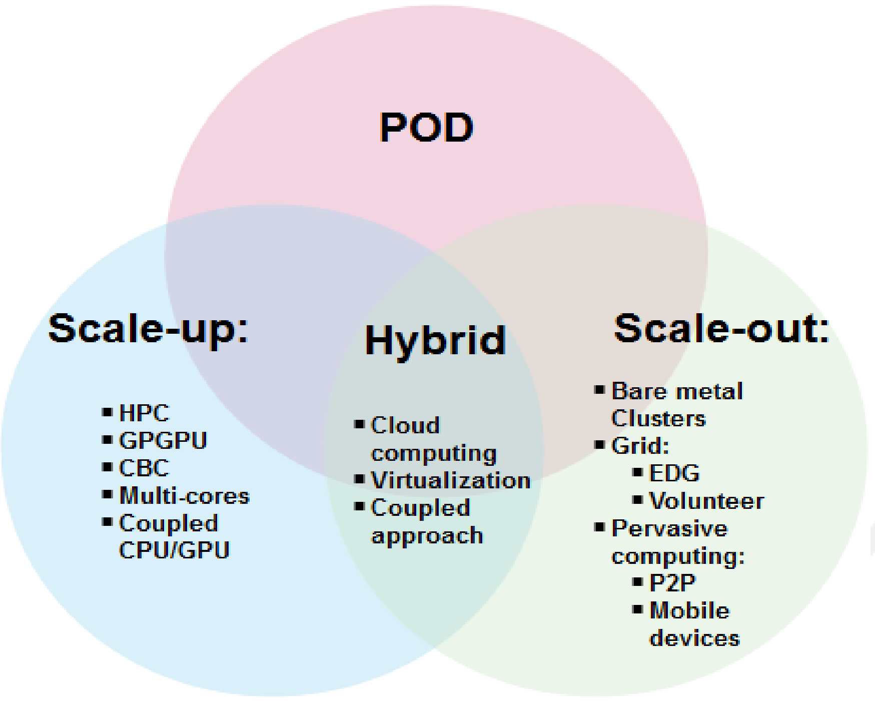
Fondo Europeo de Desenvolvemento Rexional *“Unha maneira de facer Europa”*

Appendix

Appendix #1



Big Data job execution Environments


POD

Scale-up:

- HPC
- GPGPU
- CBC
- Multi-cores
- Coupled CPU/GPU

Hybrid

- Cloud computing
- Virtualization
- Coupled approach

Scale-out:

- Bare metal Clusters
- Grid:
  - EDG
  - Volunteer
- Pervasive computing:
  - P2P
  - Mobile devices

Back…

- **Dynamically allocating available resources as YARN slaves on-demand:**
  - OBERA environment is established by running an initial application for the resource manager.
  - This initial application name is specified in the yarn-site.xml file with the the yarn.resourcemanager.hostname property and the value, <app-ID>.<framework>.HTCondor Using API to: yarn-daemon.sh start/stop resourcemanager.
  - The available data node send a JOSN file to pass a boolean flag, either of the values: true or false to identify new instance to launch a container using OPERA_DNS (OPERA_ launcher daemon)
- Example:

```
http://<IP address>:8192/api/cluster/Add_service
<resource_manager_host>:8192/api/cluster/Add_instance // For example: http://<IP
address>:8192/ (http://10.141.141.20:8192/)
 instances=<integer>
 constraints=<["JSON array of int"]>
Container_ launcher =<TRUE>
//Then
-d instances=2
-d Cluster_ID= rm01
hadoop jar HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-
<version>.jar wordcount
```