The INDIGO-DataCloud Data & Computing Platform for Scientific Communities

INDIGO - DataCloud
RIA-653549

Davide Salomoni
INDIGO-DataCloud Project Coordinator
davide.salomoni@cnaf.infn.it

INDIGO-DataCloud is co-founded by the Horizon 2020 Framework Programme
**INDIGO-DataCloud**

- **An H2020 project** approved in January 2015 in the EINFRA-1-2014 call
  - 11.1M€, 30 months *(from April 2015 to September 2017)*

- **Who:** 26 European partners in 11 European countries
  - Coordination by the Italian National Institute for Nuclear Physics (INFN)
  - Including developers of distributed software, industrial partners, research institutes, universities, e-infrastructures

- **What:** develop an open source Cloud platform for computing and data ("DataCloud") tailored to science.

- **For:** multi-disciplinary scientific communities
  - E.g. structural biology, earth science, physics, bioinformatics, cultural heritage, astrophysics, life science, climatology

- **Where:** deployable on hybrid (public or private) Cloud infrastructures
  - INDIGO = INtegrating Distributed data Infrastructures for Global Exploitationation

- **Why:** answer to the technological needs of scientists seeking to easily exploit distributed Cloud/Grid compute and data resources.
To reach the full promises of CLOUD computing, major aspects have not yet been developed and realised and in some cases not even researched. Prominent among these are open interoperation across (proprietary) CLOUD solutions at IaaS, PaaS and SaaS levels. A second issue is managing multitenancy at large scale and in heterogeneous environments. A third is dynamic and seamless elasticity from in- house CLOUD to public CLOUDs for unusual (scale, complexity) and/or infrequent requirements. A fourth is data management in a CLOUD environment: bandwidth may not permit shipping data to the CLOUD environment and there are many associated legal problems concerning security and privacy. All these challenges are opportunities towards a more powerful CLOUD ecosystem.

[...] A major opportunity for Europe involves finding a SaaS interoperable solution across multiple CLOUD platforms.

Another lies in migrating legacy applications without losing the benefits of the CLOUD, i.e. exploiting the main characteristics, such as elasticity etc.
INDIGO Addresses Cloud Gaps

- INDIGO focuses on use cases presented by its scientific communities to address the gaps identified by the previously mentioned EC Report, with regard to:
  - Redundancy/reliability
  - Scalability (elasticity)
  - Resource utilization
  - Multi-tenancy issues
  - Lock-in
  - Moving to the Cloud
  - Data challenges: streaming, multimedia, big data
  - Performance

- Reusing existing open source components wherever possible and contributing to upstream projects (such as OpenStack, OpenNebula, Galaxy, etc.) for sustainability.
INDIGO Work Packages

WP1 – Administrative and Technical Management
Leader Partners: INFN, CSIC

WP2 – Definition of Support to Research Communities
Leader Partners: LifeWatch, EGleu

WP3 – Software Management and Pilot services
Leader Partner: LIP, CEA

WP4 – Resource Virtualization
Leader Partner: DESY, KIT

WP5 – PaaS Platform
Leader Partner: INFN, Cyfronet

WP6 – Portal, Workflows and User Interfaces
Leader Partners: PSCN, INFN

Final product/Deployed App

Feedback/Revision

ESRF, PaNDaaS2 WS, 6-7/7/2016
The INDIGO-DataCloud Platform

5
INDIGO-DataCloud
General Architecture
IaaS Features (1)

• **Improved scheduling for allocation of resources** by popular open source Cloud platforms, i.e. OpenStack and OpenNebula.
  
  • Enhancements will address both better scheduling algorithms and support for spot-instances. The latter are in particular needed to support allocation mechanisms similar to those available on public clouds such as Amazon and Google.
  
  • We will also support dynamic partitioning of resources among “traditional batch systems” and Cloud infrastructures (for some LRMS).

• **Support for standards in IaaS resource orchestration engines** through the use of the TOSCA standard.
  
  • This overcomes the portability and usability problem that ways of orchestrating resources in Cloud computing frameworks widely differ among each other.

• **Improved IaaS orchestration capabilities** for popular open source Cloud platforms, i.e. OpenStack and OpenNebula.
  
  • Enhancements will include the development of custom TOSCA templates to facilitate resource orchestration for end users, increased scalability of deployed resources and support of orchestration capabilities for OpenNebula.
IaaS Features (2)

- **Improved QoS capabilities of storage resources.**
  - Better support of high-level storage requirements such as flexible allocation of disk or tape storage space and support for data life cycle. This is an enhancement also with respect to what is currently available in public clouds, such as Amazon Glacier and Google Cloud Storage.

- **Improved capabilities for networking support.**
  - Enhancements will include flexible networking support in OpenNebula and handling of network configurations through developments of the OCCI standard for both OpenNebula and OpenStack.

- **Improved and transparent support for Docker containers.**
  - Introduction of native container support in OpenNebula, development of standard interfaces using the OCCI protocol to drive container support in both OpenNebula and OpenStack.
PaaS Features (1)

• **Improved capabilities in the geographical exploitation of Cloud resources.**
  • End users need not know where resources are located, since the INDIGO PaaS layer is hiding the complexity of both scheduling and brokering.

• **Standard interface to access PaaS services.**
  • Currently, each PaaS solution available on the market is using a different set of APIs, languages, etc. INDIGO uses the TOSCA standard to hide these differences.

• **Support for data requirements in Cloud resource allocations.**
  • Resources can be allocated where data is stored.

• **Integrated use of resources coming from both public and private Cloud infrastructures.**
  • The INDIGO resource orchestrator is capable of addressing both types of Cloud infrastructures through TOSCA templates handled at either the PaaS or IaaS level.
PaaS Features (2)

• **Distributed data federations** supporting legacy applications as well as high level capabilities for distributed QoS and Data Lifecycle Management.
  • This includes for example remote Posix access to data.

• **Integrated IaaS and PaaS support in resource allocations.**
  • For example, storage provided at the IaaS layer is automatically made available to higher-level allocation resources performed at the PaaS layer.

• **Transparent client-side import/export of distributed Cloud data.**
  • This supports dropbox-like mechanisms for importing and exporting data from/to the Cloud. That data can then be easily ingested by Cloud applications through the INDIGO unified data tools.

• **Support for distributed data caching mechanisms and integration with existing storage infrastructures.**
  • INDIGO storage solutions are capable of providing efficient access to data and of transparently connecting to Posix filesystems already available in data centers.
PaaS Features (3)

• **Deployment, monitoring and automatic scalability of existing applications.**
  - For example, existing applications such as web front-ends or R-Studio servers can be automatically and dynamically deployed in highly-available and scalable configurations.

• **Integrated support for high-performance Big Data analytics.**
  - This includes custom frameworks such as Ophidia (providing a high performance workflow execution environment for Big Data Analytics on large volumes of scientific data) as well as general purpose engines for large-scale data processing such as Spark, all integrated to make use of the INDIGO PaaS features.

• **Support for dynamic and elastic clusters of resources.**
  - Resources and applications can be clustered through the INDIGO APIs. This includes for example batch systems on-demand (such as HTCondor or Torque) and extensible application platforms (such as Apache Mesos) capable of supporting both application execution and instantiation of long-running services.
AAI Features

• INDIGO provides an advanced **set of AAI features** that includes:
  • User authentication (supporting SAML, OIDC, X.509)
  • Identity harmonization (link heterogeneous AuthN mechanisms to a single VO identity)
  • Management of VO membership (i.e., groups and other attributes)
  • Management of registration and enrolment flows
  • Provisioning of VO structure and membership information to services
  • Management, distribution and enforcement of authorization policies
Storage Quality of Service and the Cloud

<table>
<thead>
<tr>
<th>Service</th>
<th>Quality</th>
<th>Availability</th>
<th>Storage Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon</td>
<td>S3</td>
<td>Durable, Reduces</td>
<td>Glacier</td>
</tr>
<tr>
<td></td>
<td>Standard</td>
<td>Availability</td>
<td>Nearline</td>
</tr>
<tr>
<td>Google</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HPSS/GPFS</td>
<td></td>
<td>Corresponds to the HPSS Classes (customizable)</td>
<td></td>
</tr>
<tr>
<td>dCache</td>
<td></td>
<td>disk+tape</td>
<td>TAPE</td>
</tr>
</tbody>
</table>

ESRF, PaNDaaS2 WS, 6-7/7/2016
The next step: Data Life Cycle

- Data Life Cycle is just the time dependent change of:
  - Storage Quality of Service
  - Ownership and Access Control (PI Owned, no access, Site Owned, Public access)
  - Payment model: Pay as you go ; Pay in advance for rest of lifetime.
  - Maybe other things

6 m | 1 years | 10 years
Data Federation through INDIGO Onedata

- **Amazon S3**: AWS USA
- **INFN Italy**
- **UPV Spain**
- **SAMBA Export**
- **POSIX Volume**
- **NFS Server**
- **VM onezone**
- **VM oneclient**
- **VM oneprovider**
- **boot2docker**
- **DNS: p-aws-useast**

**VM: demo-onedata-upv-provider**
Front-end integration schemas

• We provide graphical user interfaces in the form of **scientific gateways and workflows** through the INDIGO FutureGateway (FG). The FG can directly access the INDIGO PaaS services and software stack and allows to define and set up on-demand infrastructures for the use cases presented by our scientific communities.

  • Setting up whole use case infrastructure: The administrator will be provided with ready to use receipts that he will be able to customize. The final users will be provided with the service end-points and will not be aware of the backend.

  • Use the INDIGO features from their own Portals: User communities, having their **own Scientific Gateway setup**, can exploit the FutureGateway REST API to deal with INDIGO whole software stack.

  • Use of the INDIGO tools and portals, including the FutureGateway, Scientific Workflows Systems, Big Data Analytics Frameworks (such as **Ophidia**), Mobile Applications or Kepler extensions. In this scenario the final users as well as domain administrators will use the GUI tools. The administrator will use it as described in first case. In addition domain specific users will be provided with specific portlets/workflows/apps that will allow graphical interaction with their applications run via INDIGO software stack.
Examples of use cases

The INDIGO-DataCloud Platform

ESRF, PaNDaS2 WS, 6-7/7/2016
UC #1: A web portal that exploits a batch system to run applications

- A user community maintains a “vanilla” version of a portal using Galaxy, a computing image, plus some specific recipes to customize software tools and data
  - Portal and computing are part of the same image that can take different roles.
  - Customization may include creating special users, copying (and registering in the portal) reference data, installing (and again registering) processing tools.
  - Typically web portal image also has a batch queue server installed.
- All the running instances share a common directory.
- Different credentials: end-user and application deployment.
UC#1: Galaxy in the cloud

- Galaxy can be installed on a dedicated machine or as a front/end to a batch queue.
- Galaxy exposes a web interface and executes all the interactions (including data uploading) as jobs in a batch queue.
- It requires a shared directory among the working nodes and the front/end.
- It supports a separate storage area for different users, managing them through the portal.
UC #1: A web portal that exploits a batch system to run applications

1) The web portal is instantiated, installed and configured automatically exploiting Ansible recipes and TOSCA Templates.

2) A remote posix share is automatically mounted on the web portal using Onedata.

3) The same posix share is automatically mounted also on worker nodes using Onedata.

4) End-users can see and access the same files via simple web browsers or similar.

5) A batch system is dynamically and automatically configured via TOSCA Templates.

6) The portal is automatically configured in order to execute job on the batch cluster.

7) The batch cluster is automatically scaled up & down looking at the job load on the batch system.
UC#1: Lifecycle

• Preliminary
  • The use case administrator creates the “vanilla” images of the portal+computing image.
  • The use case administrator, with the support of INDIGO experts, writes the TOSCA specification of the portal, queue, computing configuration.

• Group-specific
  • The use case administrator, with the support of INDIGO experts, writes specific modules for portal-specific configurations.
  • The use case administrator deploys the virtual appliance.

• Daily work
  • Users Access the portal as if it was locally deployed and submit Jobs to the system as they would have been provisioned statically.
UC #1: A Graphic Overview

1) Stage Data

2) Deploy TOSCA with Vanilla VM / Container

1.a.1) build, push
1.a.2) Dockerfile (commit)

3) Install / Configure

4) Install / Configure

5) Mount

6) Access Web Portal

TOSCA Documents and Dockerfiles per Use Case

GitHub

UEC - DataCloud Docker Hub Organization

Virtual Elastic LRMS Cluster

WN ↔ WN ↔ WN

Galaxy

Front-End

Public IP

Orchestrator

IM

OpenNebula

OpenStack

Heat

Other PaaS Core Services

Future Gateway API Server

IM

Cloud Site

VP5

WP6

WP4

Other PaaS Core Services

Esrf, Pandas2 WS, 6-7/7/2016

The INDIGO-DataCloud Platform
**UC#2: A possible Phenomenal-INDIGO integration scenario**

- **Phenomenal** already relies on a very rich set-up exploiting Mesos for data processing and analysis pipelines applied to molecular phenotype data, generated by metabolomics applications.

- INDIGO is able to provide a customizable environment where a complex cluster could be deployed in an automatic way:
  - Using a specific TOSCA Template written with the help of INDIGO PaaS developers

- INDIGO could provide to Phenomenal:
  - (Automatic) Resource provisioning exploiting any kind of cloud environment (private or public)
    - Reacting on the monitoring the status of the services instantiated
  - Advanced and flexible AAI solutions
  - Advanced and flexible data management solutions
  - Advanced scheduling across cloud providers based on:
    - SLA/QoS, Data location, availability monitoring and ranked with highly flexible rules
  - An easy to use web interface for both end users and service admin/developers
UC#2: Phenomenal exploiting INDIGO

TOSCA Documents and Dockerfiles per Use Case

1.b) Automated Build

1.1.b) Dockerfile (commit)

1.a.1) build, push

1.a.2) Dockerfile

TOSCA Documents and Dockerfiles per Use Case

Virtual Elastic Mesos Cluster

Workers ↔ Workers

Public IP

Chronos/Marathon

Mesos Masters

Clues

Access Mesos Services

Deploy TOSCA with Vanilla VM / Container

Future Gateway API Server

Orchestrator

IM

IM

OpenNebula

OpenStack

Other PaaS Core Services

Cloud Site

Heat

ESRF, PaNDaS2 WS, 6-7/7/2016

The INDIGO-DataCloud Platform
**UC#3: enhancing CMS analysis workflows**

- Our initial target is end user analysis@CMS, focusing in the area of the resource usage simplification, seen from multiple perspectives:
  - **Sites:**
    - Easy solution for dynamic/elastic T2 extensions on “opportunistic”/stable resources
    - Friendly procedure to instantiate a spot ‘Tier3 like’
  - **Users:**
    - Allow the generation of a ephemeral T3 on demand, or a personal T3 to serve a group of collaborators or even a single power user - all to be used via standard CRAB
  - **Collaboration:**
    - All this includes (see it as a by-product): an approach to the opportunistic computing
    - This might be extremely useful for small sites/campus, co-funded computing centers/ multidisciplinary centers, etc.
UC#3: how to achieve the goal

Automatically create a dynamic Mesos cluster to instantiate/manage CMS flavored WNs (and needed services)

• The plan is to use VM plus docker in order to
  • Manage credential lifetime/translation/renewal through INDIGO Identity Access Management service
  • Mount posix remote file-system using Oneclient (from Onedata)
    • exploiting the INDIGO data solution for private / site storage
    • We plan to start with Dynafed; Xrootd is automatically supported via AAA
  • Execute condor_startd to connect with a CMS condor Pool (a-la HLT)
    • We’d like to exploit also the possibility to run a own condor_schedd (see next slide)

• The plan foresees to scale resources dynamically through Marathon

• The generation of the Mesos cluster and the described setup will be automated
  • Defining Tosca templates to be managed by INDIGO PaaS Orchestrator
  • Single YAML file describing the setup: Squid, Schedd, WNs in varying numbers, on-demand CMS Site name, ...
TOSCA profile
- Mesos cluster
- SITENAME
- #/type of services
  - SQUIDs
  - Schedd if needed
  - WN (range desired)
- Onedata / Dynafed attached Storage
- TFC rules
  - Fallback strategy
  - Temp storage to be used
- ...

Orchestrator

Cloud#1
VM#1 Squid1 WN#1

Cloud#2
VM#3 WN#2
VM#4 WN#3

Data as defined in TFC (Onedata, Dynafed, Xrootd Fed)

USER
Crab.py pointing to SITENAME

Schedd (CMS or private)
UC#4: running Docker containers without Docker 😊

- Adoption of docker is being very slow in HPC centers
- Thus the typical situation is that docker is not installed and one cannot run containers without some support from the system software.
- In general Docker adoption will be slow in any computing farm or interactive linux system shared by many users.
  - It will take time for sysadmins to overcome the concerns of their security teams.
  - It is yet another service to maintain...
  - .... you name it.
UC#4: INDIGO udocker

• A tool to execute content of docker containers in user space when docker is not available
  • enables download of docker containers from dockerhub
  • enables execution of docker containers by non-privileged users

• It can be used to execute the content of docker containers in Linux batch systems and interactive clusters managed by others

• A wrapper around other tools to mimic docker capabilities
  • current version uses proot to provide a chroot like environment without privileges (it runs on CentOS 6, CentOS 7, Fedora, Ubuntu)

• More info and downloads at:
  • https://indigo-dc.gitbooks.io/udocker/content/doc/user_manual.html
UC#4: INDIGO udocker

• Examples:

  # download, but could also import or load a container exported/save by docker
  $ udocker.py pull ubuntu:latest
  $ udocker.py create --name=myubuntu ubuntu:latest

  # make the host homedir visible inside the container and execute something
  $ udocker.py run -v $HOME myubuntu /bin/bash <<EOF
  cat /etc/lsb-release
  ls -l $HOME
  EOF

udocker is NOT an alternative to docker: we need the container image built by docker.

It is a tool to handle and run containers with regular user privileges and/or when docker is not available for some reason: it is very convenient to access clusters and Grid resources
UC#4: INDIGO udocker

- Everything is stored in the user home dir or some other location
- Container layers are download to the user home
- Directory trees can be created/extracted from these container layers
- proot uses the debugger ptrace mechanism to change pathnames and execute transparently inside a directory tree
- No impact on read/write or execution, only impact on system calls using pathnames (ex. open, chdir, etc)

- Does not require installation of software in the host system:
  - udocker is a python script
  - proot is statically compiled
Conclusions

• The first official INDIGO release will be out at the end of July 2016.
• The first prototype is already available for internal evaluation
• Several concrete use cases are currently being implemented by the INDIGO scientific communities
• A lot of important developments are being carried on with the original developers community so that code maintenance is not (only) in our hands
Thank you

https://www.indigo-datacloud.eu
Better Software for Better Science.
From the INDIGO FAQ

• How can INDIGO achieve resource redundancy and high availability?
  • This is achieved at multiple levels:
    • at the data level, redundancy can be implemented exploiting the capability of INDIGO's Onedata of replicating data across different data centers.
    • at the site level, it is possible to ask for copies of data to be for example on both disk and tape using the INDIGO QoS storage features.
    • for services, the INDIGO architecture uses Mesos and Marathon to provide automatic service high-availability and load balancing. This automation is easily obtainable for stateless services; for stateful services this is application-dependent but it can normally be integrated into Mesos through, for example, a custom framework (examples of which are provided by INDIGO).

• How can INDIGO achieve resource scalability?
  • First of all, we can distinguish between vertical (scale up) and horizontal (scale out) scalability. INDIGO provides both:
    • Mesos and Marathon handle vertical scalability by deploying Docker containers with an increasing amount of resources.
    • The INDIGO PaaS Orchestrator handles horizontal scalability through requests made at the IaaS level to add resources when needed.
From the INDIGO FAQ

• How can INDIGO achieve resource scalability?
  • The INDIGO software does this in a smart way, i.e. for example it does not look at CPU load only:
    • In the case of a dynamically instantiated LRMS, it checks the status of jobs and queues and accordingly adds or removes computing nodes.
    • In the case of a Mesos cluster, in case there are applications to start and there no free resources, INDIGO starts up more nodes. This happens within the limits of the submitted TOSCA templates. In other words, any given user stays within the limits of the TOSCA template he has submitted; this is true also for what regards accounting purposes.

• How do you know when and where resources are available?
  • We are extending the Information System available in the European Grid Infrastructure (EGI) to inform the INDIGO PaaS orchestrator about the available IaaS infrastructures and about the services they provide. It is therefore possible for the INDIGO orchestrator to optimally choose a certain IaaS infrastructure given, for example, the location of a certain dataset.