Geographically distributed Batch System as a Service: the INDIGO-DataCloud approach exploiting HTCondor

Speaker: Vallero Sara (INFN Torino)

CHEP 2016 - October 10/14 - San Francisco (CA)
INDIGO - DataCloud receives funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement RIA 653549.

INDIGO - DataCloud develops an open source data and computing platform targeted at scientific communities, deployable on multiple hardware and provisioned over hybrid, private or public, e-infrastructures. By filling existing gaps in PaaS and SaaS levels, INDIGO-DataCloud will help developers, resources providers, e-infrastructures and scientific communities to overcome current challenges in the Cloud computing, storage and network areas (https://www.indigo-datacloud.eu/).

See also:
INDIGO-Datacloud: a Cloud-based Platform as a Service oriented to scientific computing for the exploitation of heterogeneous resources (https://indico.cern.ch/event/505613/contributions/2227435/)

Authors:

- Doina Cristina Aiftimiei (INFN CNAF and IFIN-HH)
- Marica Antonacci (University and INFN Bari)
- Stefano Bagnasco (University and INFN Torino)
- Tommaso Boccali (University and INFN Pisa)
- Miguel Caballer (University of Perugia)
- Alessandro Costantini
- Giacinto Donvito (University and INFN Bari)
- Luciano Gaido (University and INFN Torino)
- Alessandro Italiano (INFN Bari)
- Diego Michelotto (INFN CNAF)
- Matteo Panella (INFN CNAF)
- Davide Salomoni (University and INFN Bologna)
- Sara Vallero (University and INFN Torino)
**The scientist’s way**

A typical end-user experience...

The end-user would like to:
- have full control over system and software (know the root password)
- use all resources... now!

or alternatively...
- a system tailored to her needs
- easy to use and modify
- forget about computing and do research!

**The administrator point of view...**

- have full control over system and software (do not reveal the root password)
- prevent users from breaking the system and disrupt other people’s work (isolation)
- do not tailor the system to everybody’s needs (keep it simple)
- a system easy to maintain (IaaS)

ssh to the remote head-node and submit jobs

**Virtualisation is the way to go, but a mindset shift is needed...**
According to GARTNER: Cloud Computing is a style of computing in which scalable and elastic IT-enabled capabilities are delivered as a service using Internet technologies.

From Virtual Machines to Containers:

- **package**, ship and run distributed application components with guaranteed **platform parity** across different environments

- **democratising** virtualisation by providing it to developers in a usable, **application-focused** form

- access to virtual machine virtualisation tends to be provided through, and governed by, gatekeepers in infrastructure and operations

- **Docker** is being adopted from the ground up by developers using a **DevOps** approach
• **Docker**: [https://www.docker.com/](https://www.docker.com/)
  the leading software containerisation platform. Docker containers wrap a piece of software in a complete filesystem that contains everything needed to run: code, runtime, system tools, system libraries - anything that can be installed on a server. This guarantees that the software will always run the same, regardless of its environment.

• **Apache Mesos**: [http://mesos.apache.org/](http://mesos.apache.org/)
  abstracts CPU, memory, storage, and other compute resources away from machines (physical or virtual), enabling fault-tolerant and elastic distributed systems to easily be built and run effectively. Mesos is built using the same principles as the Linux kernel, only at a different level of abstraction. The Mesos kernel runs on every machine and provides applications (e.g., Hadoop, Spark, Kafka, Elasticsearch) with API's for resource management and scheduling across entire datacenter and cloud environments.

• **Calico**: [https://www.projectcalico.org/](https://www.projectcalico.org/)
  a pure L3 approach to data center networking… enable secure IP communication between virtual machines, containers, or bare metal workloads.
  Based on the same scalable IP network principles as the Internet, Calico implements a highly efficient vRouter in each compute node that leverages the existing Linux kernel forwarding engine without the need for vSwitches. Each vRouter propagates workload reachability information (routes) to the rest of the data center using BGP - either directly in small scale deployments or via BGP route reflectors to reach Internet level scales in large deployments.
  Calico peers directly with the data center’s physical fabric (whether L2 or L3) without the need for on/off ramps, NAT, tunnels, or overlays.
  Calico supports rich and flexible network policy which it enforces using bookended ACLs on each compute node to provide tenant isolation, security groups, and external reachability constraints.

• **Gartner**: [http://www.gartner.com/technology/home.jsp](http://www.gartner.com/technology/home.jsp)
Our mission

Key components of the INDIGO PaaS framework.

The goal:
- keep delivering a well consolidated computational framework, while complying to modern computing paradigms
- ease system administration to all levels (hardware/applications)
- provide a smooth end-user experience

Batch System as a Service:
- automatically and dynamically deploy a complete batch system cluster (with appropriate user interfaces) in highly-available and scalable configurations
- use HTCondor:
  - widely used within the scientific community
  - cloud aware
- networking:
  - HTCondor shared port mode
  - overlay networks
  - span the cluster over multiple sites (Connection Broker)
- storage:
  - scalability, performance and reliability
  - CVMFS (using Parrot)
  - integrate with the INDIGO Data Services (Onedata, Dynafed, FTS)
Two approaches

**Mesos**

- **Framework**
  - fine grained control on the application’s tasks
  - ad hoc authorisation and scaling rules
  - isolation and packaging achieved with the Docker Containerizer module of Mesos
  - components:
    - master
    - scheduler (implements policies on the resource offers)
    - executors (to launch tasks on the slave nodes)
  - can be used in standalone or as a component in the INDIGO PaaS system

**Marathon**

- using Marathon framework (health checks, failover capabilities)
- master, scheduler and executors packaged in separate pre-configured Docker containers
- containers are deployed as Long Running services
GEOGRAPHICAL DEPLOYMENT

Torino  Bologna

GARR-X network (10 Gbps)

Bari
Networking

**Calico functionalities:**

- **Orchestrator Plugin:** to manage the Calico network as with the orchestrator (Docker) tools
- **Felix agent:** programs routing rules and ACLs into the Linux kernel (FIB and iptables)
  - **etcd:** communication between components and datastore
  - **BGP client:** distributes the routing tables around the datacenter

**Network topology:**

- now full-mesh peering among all hosts
- consider a more scalable configuration as sketched in AS3:
  - iBGP peering within single AS
  - one node per AS acts as Route Reflector (RR)
  - eBGP peering among RR
Ansible provides: workflow orchestration with configuration management, provisioning and application deployment.

A single playbook to deploy the cluster nodes:

Some configuration variables:

```bash
- hosts: localhost
  connection: local
  roles:
    - { role: indigo-dc.docker }
    - { role: indigo-dc.etcd }
    - { role: indigo-dc.calico, etcd_peers: "{{ etcd_peers_list }}" }
    - { role: indigo-dc.zookeeper }
    - { role: indigo-dc.mesos, mesos_install_mode: "master" }
    - { role: indigo-dc.mesos, mesos_install_mode: "slave" }
    - { role: indigo-dc.marathon, marathon_hostname: "{{ ansible_fqdn }}" }

-  docker_bridge_ip: "172.17.0.1"
  zookeeper_host_list: "[mesos.cloud.ba.infn.it, mesos.cloud.cnaf.infn.it, cloud-test-01.to.infn.it]"
  mesos_masters_list: "{{ zookeeper_host_list }}"
  zookeeper_peers: "{{ zookeeper_host_list }}"
  etcd_peers_list: "{{ zookeeper_host_list }}"
  marathon_password: "XXX"
  etcd_force_new_cluster: false
  service_discovery: "none"
  etcd_initial_cluster_state: "existing"
```

We support CentOS 7 and Ubuntu 14.
Cluster status

Monitoring:
- data are collected from Mesos with a custom Python script that does http GET using the `requests` module
- metrics are injected in Graphite and visualised through Grafana
First results

Application Monitoring:

- Linux stress utility to simulate workload
- data collected with Grafana Docker plugin, relying on HTCondor Python API (https://github.com/fifemon/probes)
- ElasticSearch backend
- more custom metrics to be defined

DISCLAIMER: results are not meaningful per se, they show that we have a couple of clusters up and running and we are able to monitor the application performance. Benchmarking tests are ongoing.

<table>
<thead>
<tr>
<th>FROM</th>
<th>Torino</th>
<th>CNAF</th>
<th>Bari</th>
</tr>
</thead>
<tbody>
<tr>
<td>Torino</td>
<td>0.065</td>
<td>10.760</td>
<td>18.409</td>
</tr>
<tr>
<td>CNAF</td>
<td>32.424</td>
<td>0.092</td>
<td>13.913</td>
</tr>
<tr>
<td>Bari</td>
<td>18.351</td>
<td>13.879</td>
<td>0.175</td>
</tr>
</tbody>
</table>
Summary:

• consolidated computational framework, modern paradigms
• automatically deploy a complete batch system cluster
• software stack: HTCondor, Mesos, Marathon, Docker, Calico
• first steps towards geographical deployment:
  • VPN between 3 hosts at Torino, CNAF and Bari INFN sites
  • L3 network isolation with Calico
  • service clusters distributed over the 3 sites (etcd, Zookeeper, Calico, Mesos and Marathon)
• Docker images for HTCondor daemons
• automatic cluster deployment with Ansible
• cluster and application monitoring with Grafana

Coming soon:

• HTCondor benchmarking tests
• scalable network configuration
• definition of a model for user login to the batch-farm on-demand
• get ready for the INDIGO-DataCloud Service Catalogue (https://www.indigo-datacloud.eu/service-component)