Orchestration on Cloud + HPC

In a nutshell: Many applications which run on High Performance Computers (HPC) are part of bigger workflows that run in the cloud, such as those with tightly-coupled data or extensive big data analytics. As well as it is done for microservices applications in the Cloud, an automatic hybrid deployment and management of a modularized app in HPC and Cloud optimizes its performance and enables new development architectures.

Why Cloud + HPC?

Coming from Grid Computing, both Cloud and High-Performance Computing – HPC – architectures optimize complex application runtime by parallelizing as many processes as possible. The use of one or the other is usually driven by the nature of the parallelization itself: Are the processes in need of fast and constant communication between them? If the answer is No, the application can run on a cluster that works as a "cloud of resources": Cloud Computing – while if the answer is Yes, the cluster needs to work as a "unique supercomputer": An HPC. In that way, cloud clusters focus on providing resources as flexible and fast as possible, while HPC ones provide fast and fine-grain parallelization between processes.

The fact that the vast majority of applications that runs nowadays in the internet fall in the first group, and that cloud clusters are much more cheaper to create and maintain, have led cloud computing ecosystem to flourish with uncountable architectures, design patterns, and tools in almost any computer science domain.

Applications that fall on the second group are usually those which involve some sort of modelling and simulation algorithms, such as those in the fields of astronomy, cosmology, chemistry, engineering, fluid dynamics, earth and climate science, condensed matter physics, life science, particle physics and plasma physics. Before running any application on an HPC, an expert user must learn about the particularities on the system, how to access and compile the required software, how to launch jobs, etc. This process is usually painful and requires the help of the HPC administrators. On top of that, it is common that these models and simulations need additional work to get the desired results, like data aggregation from different sources in tightly-coupled simulations, or big data analysis. These operations, however, clearly fall within the first group! Moreover, the complete workflow of these applications is often no different from any other Cloud architecture!

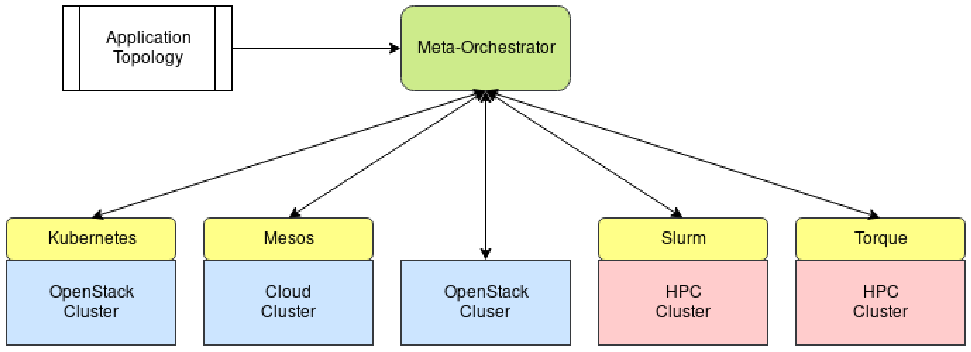
What if we could run applications that benefit from the cloud ecosystem (IaC, SaaS, DevOps, Interoperability, etc.) while keeping some processes running in HPC?

How? Meta-orchestration

One of the key concepts that have enabled cloud computing is "Orchestration". It oversees deploying, running and monitoring all the components of an application in the cluster. Additionally, it can perform other tasks like healing (manage errors), scaling and logging. Cloud cluster resources can be accessed rather directly by virtualization, or through orchestrators like the well-known Kubernetes or Mesos. However, HPC resources are expensive and highly contended, and as a result resources are managed by queues systems (workload managers) like Slurm or Torque. These can be seen as simple orchestrators, that only manage the access to the resources.

To use applications using both cloud and HPC as we do in cloud computing, the same concept of orchestration is needed, but working on both clusters. As cloud orchestrators work by directly managing the resources, one way to go would be to remove the workload manager in the HPC and install the same orchestrator to manage both clusters. Unfortunately, this option is most of the time not possible as changing a running HPC from one workload manager to another suppose huge changes in the cluster and in the way the users use it. Moreover, even for HPC clusters that are not yet in production, using a cloud orchestrator to manage the bare metal resources is often a worse option than using a workload manager like Slurm or Torque, as the latter are much more prepared to deal with the kind of workloads that an HPC must face.

In this post the proposed solution is to use a meta-orchestrator, that is, an orchestrator that works like any other cloud orchestrator, but instead of managing resources, it orchestrates the application through other low-level orchestrators. The following image illustrates that idea, in which a meta-orchestrator could orchestrate application(s) running on a heterogeneous architecture, where each computing system is at the same time managed by a lower-level orchestrator. In a heterogeneous infrastructure, it could occur that, on the cloud part, not all of the clusters have an orchestrator already in place (the OpenStack cluster in the image below). Because of this fact, it also desirable that the meta-orchestrator can work as a regular cloud orchestrator as, dealing directly with the cluster without intermediaries.



This meta-orchestrator must comply with the following set of requirements for the above diagram to work:

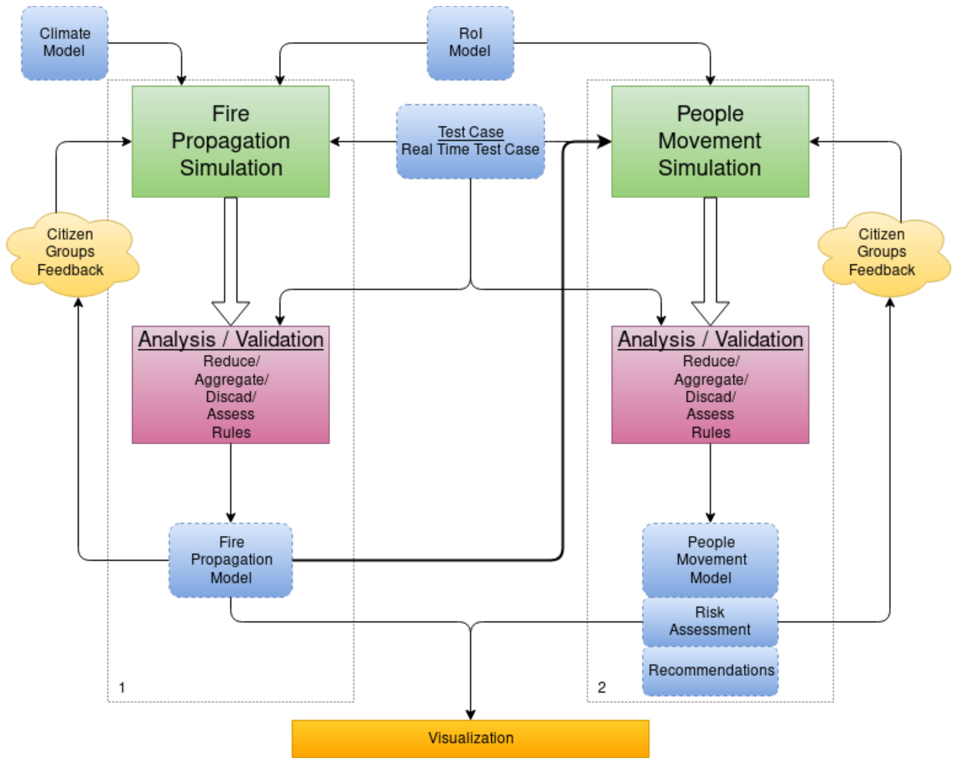
- Agentless architecture: The use of agents it is not possible / recommended, as those agents should have to run in the lower-level orchestrators. Therefore, the higher-level orchestrator needs to stablish communication with the layer below without the use of agents.
- Event driven workflows: As direct manage of resources is not possible, it must orchestrate driven by the events occurring in the infrastructure.
- Batch jobs vs long-time running applications: HPC tasks must always have a maximum time of execution, as they are always "batch" jobs. On the contrary, cloud tasks usually stay running indefinitely listening on events. Both types of tasks must be orchestrated.
- DSL: The meta-orchestrator needs to define and understand a Domain Specific Language – DSL- that allows the app developer to describe the topology of its application agnostic to the infrastructure.
- Containers: Containerization enables interoperability between multiple computing hosts. Hence, a meta-orchestrator should as well support container orchestration.

Croupier

Atos R&D – ARI - [1], with the help of University of Stuttgart [2, 3] and CESGA [4] has implemented such piece of technology, named Croupier [5, 6, 7]. It is based on Cloudify [8, 9], a cloud orchestrator that instead of Kubernetes or Mesos, it is based on event driven workflows, it already supports many cloud platforms and orchestrators like OpenStack or Kubernetes, and it is extensible through plugins.

Croupier is therefore implemented as a plugin that installs on a Cloudify server, adding to it two main characteristics: (1) Connection with the two most important HPC workload managers, Slurm and Torque; and (2) the possibility to run workflows of batch jobs. It also supports another cloud key component in HPC, containerization, through Singularity [10]. In that sense Croupier is defined as a **Cloudify plugin to orchestrate batch applications in HPC and Cloud environments**.

Croupier, as well as Cloudify, uses TOSCA [11], an open standard based to define topology and orchestration for cloud applications. It allows extensions using Object Oriented Design, and Croupier has extended it to cover not only cloud applications but also HPC ones. Below is the graphic representation of a TOSCA file (usually called "blueprint") that describe a real application that simulates and analyses citizen evacuation under an ongoing fire. Each rectangle is a high-level view of a sub-workflow that can run on different computing infrastructure (green for HPC, purple and yellow for cloud). Rectangles with rounded corners (light blue) represent the data flow.



A blueprint attached to a YAML file of inputs is called a "deployment". Apart from setting specific arguments for the application itself, in the inputs file there is defined the computing infrastructures that are going to be used (the concrete clouds and HPCs). Croupier uses the deployment to execute the visualization, analysis and validation processes on the Cloud in a long-time running way, while executing the simulations (batch tasks) on the HPC triggered by certain events.

As for the interfaces available to interact with Croupier, out of the box there are the ones provided by Cloudify: Web based, API Rest, and CLI. Moreover, Atos is developing a new web interface that aims to improve how the user is queried for enter the inputs of the applications: They can be displayed in categories and in a concrete order, rendered in different manners (as a drop box, as dates, etc.), and they can be automatically filled based on the value on other inputs.

Croupier provides documentation [5] on how to install and use it, as well as it provides multiple examples [6] to get started. It is 100% open source Apache2.0 licensed. **Try it!**

More information

[1] Atos Research & Innovation: http://booklet.atosresearch.eu/content/ari-profile

[2] University of Stuttgart: https://www.uni-stuttgart.de/en/

[3] HPC Center Stuttgart: https://www.hlrs.de/home/"

[4] CESGA: https://www.cesga.es/

[5] Croupier docs: https://croupier.readthedocs.io/en/latest/

[6] Croupier examples: https://github.com/ari-apc-lab/croupier-resources

[7] Croupier source code: https://github.com/ari-apc-lab/croupier

[8] Cloudify: https://cloudify.co/

[9] Cloudify Docs: https://docs.cloudify.co/4.6/

[10] Singularity: https://singularity.lbl.gov/

[11] TOSCA: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca

Javier Carnero
Organic Analyst – Advanced Parallel Computing Lab @ ARI
T: +34 955 25 41 03
Av. Kansas City 9, Sevilla – Spain
http://booklet.atosresearch.eu