Harvesting dispersed computational resources with OpenStack: a Cloud infrastructure for the Computational Science community

Mirko Mariotti[*] [ab], **Loriano Storchi** [bcd], **Daniele Spiga** [b], **Giuseppe Vitillaro** [d], **Mirco Tracolli** [b], **Diego Ciangottini** [b], **Manuel Ciangottini** [b], **Valerio Formato** [b], **Matteo Duranti** [b], **Matteo Mergé** [e], **Paolo D'Angeli** [f], **Roberto Primavera** [f], **Antonio Guerra** [f], **Livio Fanó** [ab], **Bruna Bertucci** [abf]

[a]*Dipartimento di Fisica e Geologia, Universitá degli Studi di Perugia*
[b]*INFN sezione di Perugia*
[c]*Dipartimento di Farmacia, Universitá degli Studi G. D'Annunzio, Chieti*
[d]*Istituto di Scienze e Tecnologie Molecolari, CNR*
[e]*INFN sezione Roma 2*
[f]*Space Science Data Center at the Italian Space Agency*
E-mail: mirko.mariotti@unipg.it, loriano@storchi.org, daniele.spiga@pg.infn.it

Harvesting dispersed computational resources is nowadays an important and strategic topic especially in an environment, like the computational science one, where computing needs constantly increase. On the other hand managing dispersed resources might not be neither an easy task not costly effective. We successfully explored the use of OpenStack middleware to achieve this objective, our man goal is not only the resource harvesting but also to provide a modern paradigm of computing and data usage access. In the present work we will illustrate a real example on how to build a geographically distributed cloud to share and manage computing and storage resources, owned by heterogeneous cooperating entities

International Symposium on Grids and Clouds (ISGC) 2018 in conjunction with Frontiers in Computational Drug Discovery
16-23 March 2018
Academia Sinica, Taipei, Taiwan

[*]Speaker.

1. Introduction

Harvesting dispersed computational resources is surely an important, strategic and poorly explored topic. This is especially true in an environment, like the computational science one, where on one hand computing needs constantly increase and, on the other hand, quite frequently is not easy to find the needed know-how to handle continuously updated hardware architecture. However managing dispersed resources might not be neither an easy task not costly effective. We successfully explored the use of OpenStack [1] middleware to achieve this objective, aiming not only at harvesting resource but also at providing a modern paradigm of computing and data usage access.

A cloud is a complex and distributed system made of computing resources, networks and many running applications [2, 3]. Problems and failures will surely occur in this huge set of components and one must constantly ensure their correct operation and co-operation. At the physical layer, a Cloud infrastructure relies on hardware components which provide rough computing resources like networks, routers, switches and servers. Cloud-computing providers offer their services using different models. Services at the PaaS level are more complex because they offer various components to develop and deploy applications. Finally, in the SaaS model [3] the user gets access to a set of applications, instead the cloud provider manages the infrastructure and the platform. The applications are not necessarily owned or managed by the cloud provider itself, but they may be offered by a third party (i.e. the application provider) who benefits of the IaaS or PaaS resources.

Clearly, the deployment of a multi-sites Cloud [3] implies the management of multiple computing services with the goal of sharing resources between more sites, allowing local data centers to scale out with external assets. The cited scenario immediately points out portability, interoperability and compatibility issues, but on the other side this opens really interesting scenarios and use cases especially in load balancing, disaster recovery and advanced features such as cross-site networks, cross-site storage systems, cross-site scheduling and placement of remote VMs. Moreover, a cloud infrastructure allows the exploitation of Platform as a Service, and Software as a Service solutions which in the end it is what really matter for the science.

The main goal of the present work is to illustrate a real example on how to build such a geographically distributed cloud, and how to share and manage computing and storage resources, owned by heterogeneous cooperating entities. We put together four different entities, different both in terms of basic computational resources as well as in terms of scientific interest.

In the following (Section 2), we will describe the context of this work introducing the deployed architecture and we will report all the tests we performed to validate and benchmark our multi-sites Cloud. Afterwards, in Section 3, we are going to dive into more specific test cases showing the full functionalities of our cloud, and the role of its components. Finally, we draw some conclusion and we will discuss the results of our work, pointing out issues and open challenges to be addressed in future developments.

2. Overview of the architecture

The entities involved in the present project, shown in figure 1, are: the Department of Physics and Geology - I.N.F.N. Sez. Perugia, Department of Chemistry (UNIPG) also located in Perugia, the Department of Pharmacy (UdA) located in Chieti and ASI-SSDC (Space Science Data Center at the Italian Space Agency) located in Rome. The involved sites are not only geographically dislocated but also a good representative of the research interest diversity we want to explore within our Cloud infrastructure.

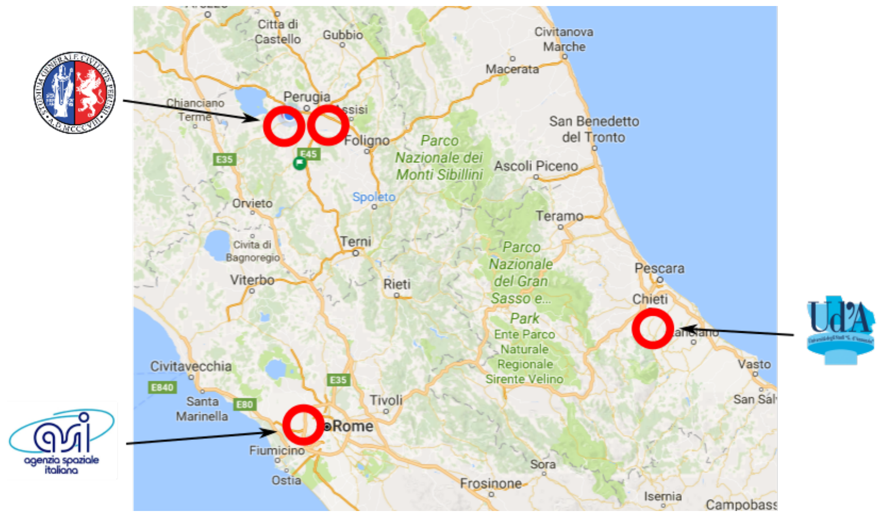


Figure 1: An overview of the location of the involved sites

In the following we are reporting some details about the deployed architecture, as well as the result of the basic benchmark tests performed.

2.1 Connection of sites

The deployed networking structure has been developed for the specific purpose of creating a distributed overlay L2 Ethernet to be used by OpenStack. In such respect two remote sites are connected by a point-to-point link build using the VXLAN [4] protocol. OSI L2 Ethernet frames of both the OpenStack management and projects VLAN networks have been encapsulated within it. The whole project aims to harvest resources so also to build the overlay L2 network infrastructure commodity hardware have been used, specifically, on each site, a Linux server has been installed and configured to create the connection. Each server is running an Ubuntu 16.04 Linux distribution [5].

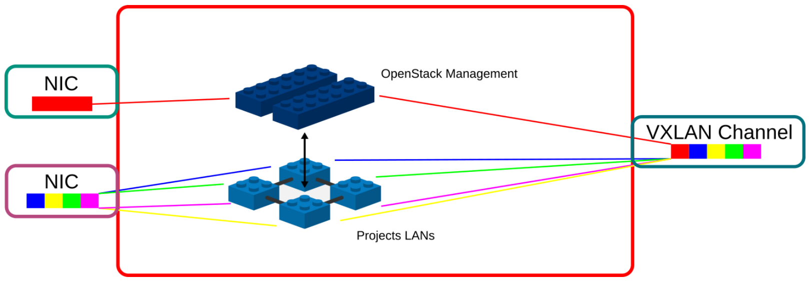
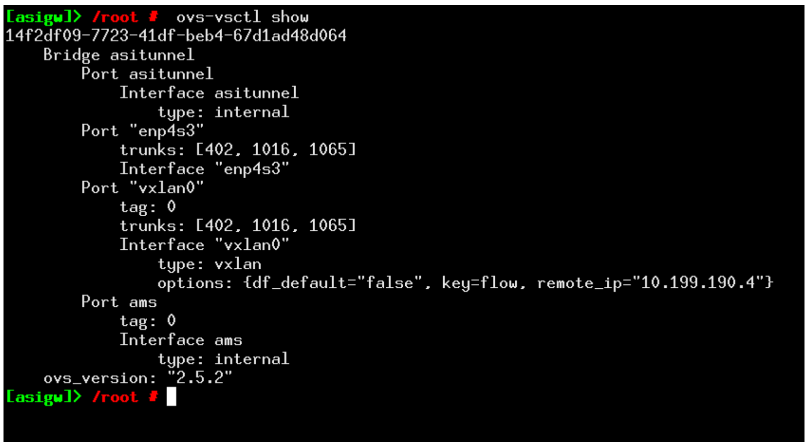


Figure 2: Diagram of a connection server

As shown in figure 2, each server has 3 NICs: the first is connected to the OpenStack management LAN, the second to the OpenStack projects LAN and the last to the Internet via a public

IP Interface. The OpenStack management traffic and the projects traffic, VLAN based in this case, are encapsulated into a VXLAN channel using the Openvswitch software version 2.5.2 [6].



```
[asigw]> /root #  ovs-vsctl show
14f2df09-7723-41df-beb4-67d1ad48d064
    Bridge asitunnel
        Port asitunnel
            Interface asitunnel
                type: internal
        Port "enp4s3"
            trunks: [402, 1016, 1065]
            Interface "enp4s3"
        Port "vxlan0"
            tag: 0
            trunks: [402, 1016, 1065]
            Interface "vxlan0"
                type: vxlan
                options: {df_default="false", key=flow, remote_ip="10.199.190.4"}
        Port ams
            tag: 0
            Interface ams
                type: internal
    ovs_version: "2.5.2"
[asigw]> /root #
```

Figure 3: Examples of configuration of the Openvswitch bridge

An example of the configuration of the Openvswitch bridge is reported in listing 1. Specifically the reported listing creates the configuration shown in figure 3.

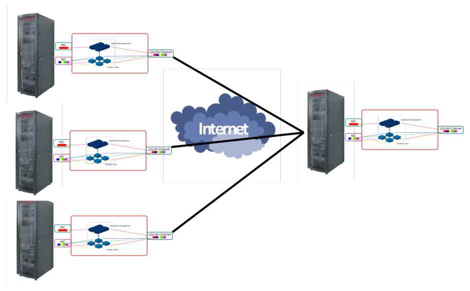
```
        auto asitunnel
        allow-ovs asitunnel
                iface asitunnel inet manual
                ovs_type OVSBridge
                ovs_ports ams enp4s3 vxlan0

        allow-asitunnel enp4s3
        iface enp4s3 inet manual
                ovs_bridge asitunnel
                ovs_type OVSPort
                ovs_options vlan_mode=trunk trunks=402,1016,1065

        allow-asitunnel vxlan0
        iface vxlan0 inet manual
                ovs_bridge asitunnel
                ovs_type OVSTunnel
                ovs_tunnel_type vxlan
                ovs_options tag=0 vlan_mode=native-untagged trunks=402,1016,1065
                ovs_tunnel_options options:remote_ip=10.199.190.4 options:key=flow options:df_default=false
```

Listing 1: Interfaces configuration example

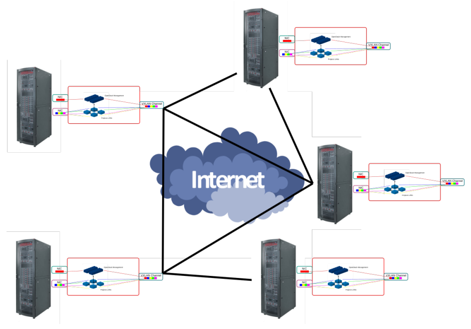
These servers are the "backbone" for the overlay networks, they can be connected in a custom topologies simply creating different vxlan channels among them. Two scenarios have been successfully tested, a simple tree-like connection of the sites (as reported in figure 4a) and a more complex one using a redundant paths as well as the SPT protocol to avoid loops (as instead reported in figure 4b).

2.2 Security

To address the sites network security all the traffic have been encrypted, specifically OpenVPN [7] and IPsec [8] have been used, depending on site to site specific networking constraints. The two solutions have different advantages and disadvantages. Though generally IPsec guarantees better performances, often routers are not properly configured to recognize and to handle it correctly. Instead an indisputable advantage of using OpenVPN is that the VXLAN traffic is encapsulated



(a) Tree like connected nodes



(b) Mesh connected nodes

Figure 4: Examples of connected nodes

into standard TCP or UDP connections. Examples of both configurations are shown in listings 2 and 3.

```
proto tcp-server
port 443
keepalive 10 60
ping-timer-rem
persist-tun
persist-key
dev tun
ifconfig 10.199.190.1 10.199.190.2
secret static.key
```

Listing 2: OpenVPN configuration example

```
#!/usr/sbin/setkey -f
flush;
spdflush;
add 193.205.XXX.YYY 193.204.JJJ.HHH  ah 23024 -A hmac-md5 0x12345678901234567890123456789012;
add 193.204.JJJ.HHH 193.205.XXX.YYY  ah 24023 -A hmac-md5 0xabcdefabcdefabcdefabcdefabcdefabcdef;
add 193.205.XXX.YYY  193.204.JJJ.HHH esp 1023024 -E 3des-cbc 0x1234567890123456789012345 .... ;
add 193.204.JJJ.HHH  193.205.XXX.YYY  esp 1024023 -E 3des-cbc 0xabcdefabcdefabcdefabcdefabcdefabcdef ....;
spdadd 193.205.XXX.YYY  193.204.JJJ.HHH  any -P out ipsec esp/transport//require ah/transport//require;
spdadd 193.204.JJJ.HHH  193.205.XXX.YYY  any -P in ipsec esp/transport//require ah/transport//require;
```

Listing 3: IPSEC configuration example

Finally, it is somehow worth to report that all the interconnected sites have to trust one another since misconfigurations potentially can have consequences on the whole system.

2.3 OpenStack

To fine tune the network setup and avoid cross site unnecessary traffic, different sites have been logically separated using different availability zones, with VMs on every zone having the capability of independently use site-wide resources. Important examples are:

- accessing the Internet through the site where they are physically running in order to avoid the saturation of VXLAN channel.

- accessing site-specific storage

2.4 Automation

All the elements of the described infrastructure have been automatized to minimize the number of manual operations, minimizing also the extra source of errors, in particular:

- The sites are L2 connected and are on the same broadcast segment. Then an automated installation/management system can easily be used on the whole system.
- Openvswitch instances on the connected nodes can be operated via the Openflow protocol, so via a simple script in OpenStack one can trigger changes in the network infrastructure.
- The same could be done with the physical switches within sites, for this to work switches have to be Openflow protocol [9] enabled.

2.5 Basic Tests

In order to validate the described infrastructure two main tests have been conducted, test on the sites connectivity and an OpenStack related test.

Firstly we performed a connectivity test needed to verify the bare network performances degradation dues to the extra cryptography layer (i.e. OpenVPN and IPsec) and tunneling layer (VXLAN). The results of the test are summarized in Table 1, where average throughput values (i.e. connection performance) are reported for native links of 100Mb/s (that is the link from Perugia to Chieti) and 1Gb/s (that is the link from Perugia to Rome).

Link bandwith	OpenVPN (UDP)	IPsec	VXLAN and OpenVPN
100Mb/s	98Mb/s	97Mb/s	80 Mb/s
1Gb/s	905Mb/s	912Mb/s	797Mb/s

Table 1: Throughput values reported for native links of 100Mb/s and 1Gb/s, see the text from more details.

Secondly, an OpenStack functionality test has been performed as well, that is a cluster of VMs have been deployed on the infrastructure. The deployment of the cluster consists of the creation of principally three actors: a master (one or more), several slaves and some load balancers. To accomplish this task we used the OpenStack Orchestration named Heat [10] and we also rely on Ansible [11] to automate all the process.

We deployed several clusters, each one with a different number of VMs up to a configuration made of a single master node, two load balancers and six slaves. Interestingly, despite the fact we used each time different sites and hardware composition, all the tests were successful. The latter demonstrates surely the robustness of our infrastructure.

3. Porting of concrete use cases

In the present section, we will describe the porting of concrete use cases exploiting the available PaaS solutions provided by INDIGO-DataCloud project and following the upcoming EOSC directives. We identified: ab initio quantum chemistry applications and the data analysis of the AMS-02 (Alpha Magnetic Spectrometer) Cosmic Ray experiment on the International Space Station [12–14].

3.1 The ab initio quantum chemistry: a test case

Over the years quantum chemistry methods have shown an impressive development , and nowadays there are a large number of well known and well established softwares representing a reliable approximation to the well known non-relativistic Schrödinger and relativistic Dirac equations [15–19]. Typically quantum chemistry is strictly related to High Performance Computing, that is most of the quantum chemistry softwares relay on the availability of high performance computing resources. Indeed in most of the ab initio quantum chemistry applications is not always obvious how to efficiently use computational framework as a computational grid [20] or a cloud [21].

Nevertheless there are scenarios where the problem one needs to face can be easily distributed on a Cloud, or Cloud-like, computational infrastructure [22–24], without the strict needs of high throughput and low latency network infrastructures, or different kind of specialized hardware such as GPGPUs. Anyhow is important to underline that in the near future one may try to explore IaaS (Infrastructure as a Service) using Docker containers [25], that provides an easy and efficient way to exploit specialized hardware.

Despite the fact we cited quite different computational schemes, undoubtedly all of them may be faced using essentially a similar workflow. Reason why the approach we will describe in the following may be easily adapted to a really wide range of possible computational chemistry use cases.

In almost any computational chemistry protocol one of the very first step is the geometry optimization [15], that is to find the arrangement in the space of the atoms, constituting the molecule, so that the interatomic forces are close to zero and the position on the PES (Potential Energy Surface) is a stationary point. In the simplest case of a diatomic molecule one can easily determine the optimal geometry performing several energy computation at different interatomic distances. Once the one-dimensional PES is computed one can find the optimal geometry, thus the optimal interatomic distance, easily looking for the minimum value of the energy.

Clearly we can exploit the intrinsic parallelism of the cited PES computation simply distributing the several single point energy calculations on the available resources. Specifically in the present work we used the Dirac-Kohn-Sham module of the BERTHA code [18, 19] to compute the needed energies. Is important to underline that the actual version of the BERTHA software has been fully parallelized, that is there is not a single step still running in serial. In fact, according to Amdahl's law [26], the speed-up of a code is upper limited by the its serial portion, in our case we removed this upper bound being able to archive great results making both CPU-time and memory scalable with the number of processors used. That is, BERTHA virtually overcomes at once both time and memory barriers associated with four-component DKS calculations [18, 27, 28].

The BERTHA executable has been embedded into a Docker image [25] so that it can be easily managed using MESOS/MARATHON [29, 30]. In the present test version a simple file is used to specify the bond length, but obviously in the near future (indeed one of us is already working on a pyBERTHA code, that is a python [31] wrapper to the underlying Fortran code that surely will enormously simplify any further development) an external database may be used to feed any instance with a specific input and to store the final results. Obviously each single BERTHA docker instance (container) can be both a serial or parallel executable depending on the molecular system dimension and on the number of cores available. In Figure 5 a general overview of the deployed architecture is showed.

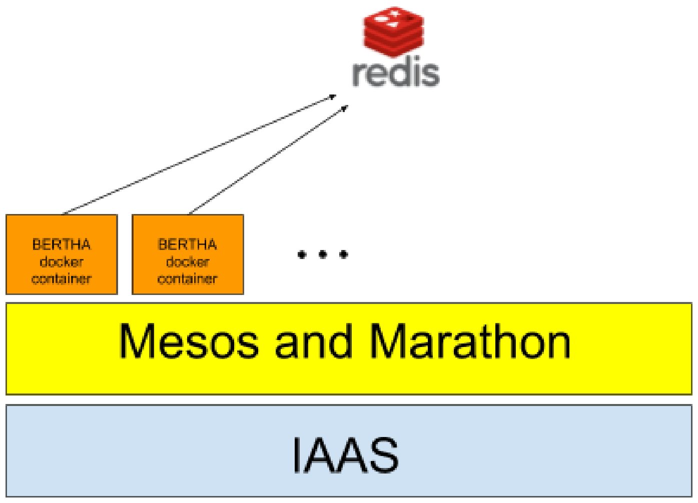


Figure 5: An overview of the deployed architecture, see text for details.

Once all the single point computation have been completed the results, so the energies, can be collected and for example used to compute the equilibrium geometry. In the present test we performed the geometry optimization for the $AuOg^+$ molecular system. By purpose we selected a superheavy elements like the Organesson (Z = 118) that only recently has been added to the periodic table [32]. The reported results were carried out with the cited full-parallel version of BERTHA. The large component of the basis set for Au and Og was generated by uncontracting triple-ζ quality Dyall's basis sets [33–36] augmented with the related polarization and correlating functions. For the gold atom, a previously optimized auxiliary basis set for density fitting denoted as B20 [37] was used. While for the Organesson auxiliary basis sets were automatically generated following the procedure described in Ref. [38]. Finally the BLYP functional made of the Becke 1988 (B88) exchange [39] plus the Lee-Yang-Parr (LYP) correlation [40] was used. An energy convergence criterion of 10^{-7} Hartree on the total energy was adopted.

Once all the energies have been computed and collected to determine the bond length we simply performed a polynomial fitting using NumPy [41], and the bond length is computed calculating the minimum of the energy. In the specific case of the $AuOg^+$ the computed value is 2.862 Å. To determine the reported equilibrium geometry we used 11 single point (as clearly reported in Fig. 6) calculations. Being the proposed approach clearly embarrassingly parallel we were able to easily geographically distribute the full calculations set in our Cloud infrastructure.

Having demonstrated, with a practical application, how to exploit the developed Cloud infrastructure it is i important to stress once again that the described approach can be directly applied to many others computational chemistry scenarios. In this regards is important to name a computational scheme belonging to different area of the computational chemistry, for example the virtual screening of chemical library [42]. Indeed, though in the present section we focused our attention

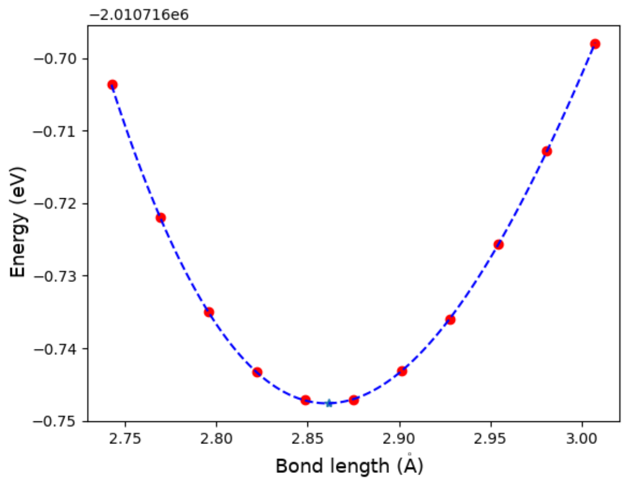


Figure 6: One-dimensional PES for the AuOg+ molecular system. The red dots are the computed total energies, while the blue triangle represent the fitted minimum value and so the equilibrium bond distance.

on quantum chemistry methods, we cannot avoid mentioning how the virtual screening technique fits well with a Cloud infrastructure. Virtual screening (VS) uses computer-based methods to discover new ligands on the basis of biological structure. Thus, a typical VS workflow is designed to run many jobs (i.e. pKa prediction [43], tautomer enumeration [44], docking [45]) to screen very large collections of ligands against a particular biological target [46].

Therefore despite the fact we cited computational schemes belonging to different fields of the computational chemistry, undoubtedly all of them may be faced using essentially a similar approach. Reason why the detailed use case represent undoubtedly a perfect demonstration of the described platform adaptability to computational chemistry use cases.

3.2 DODAS for the AMS-02 data analysis

The described platform has been successfully used through the Dynamic On Demand Analysis Service (DODAS), an automated system that simplifies the process of provisioning, creating, managing and accessing a pool of heterogeneous computing and storage resources, by generating clusters to run batch systems thereby implementing the Batch System as a Service paradigm. In this context it has been used in order to provide a complete batch system to the AMS [12–14] physicist for a early validation of the complete workflow. For the scope of this implementation HTCondor technology [47] is adopted as workload manager building an auto-scaling batch farm. Apache Mesos [29] manages CPU, RAM and storage offered by cloud providers, and Marathon is the application framework.

The main target of this project is to create an HTCondor cluster accessible by different users but managed in a cloud environment. Practically we want to use opportunistic resources and scale them in the base of the demand. HTCondor is composed principally of three actors: the central manager, the submitter node and the worker node. The central manager controls the HTCondor pool and coordinates the job requests. The submitter is the node where users can access and prepare their tasks, which will then be sent to the pool. The worker nodes take care of performing the proper task and to send back the results to the user. All nodes are easy to maintain due to Docker containers, that guarantees also the scalability and versatility of the cluster. The deployment of the described cluster is obviously customizable and the cluster maintainers may adjust also the resources used by each container.

We want to stress that the AMS (Alpha Magnetic Spectrometer) is an international collaboration that includes more than 250 physicists, from almost 50 institutions and about 15 countries. The data produced by the AMS-02 Cosmic Ray detector, installed since May 19$^{\text{th}}$ 2011 on the International Space Station, amounts to 35 TiB of raw data that become about 100 TiB once processed and converted in ROOT [48] format. The full processing of the almost 2 PiB of reconstructed data and MonteCarlo simulations typically requires several days with thousand of concurrent jobs for each user. This analysis work is typically done by tens of users at the same time. The typical analysis workflow requires a remote reading the data stored at CERN (European Organization for Nuclear Research [49]) or in the other regional centers of the collaboration and the remote writing of the ROOT files with the analysis results.

DODAS is built using several INDIGO-DataCloud [50] technological solutions and recipes, the overall cluster topology as well as the orchestration of all services such as HTCondor, Mesos, Marathon [51], Squid proxy [52], including any possible software dependency, are orchestrated with TOSCA [53] templates and Ansible roles. The key outcome is that a single YAML file allows the description of the complex setup required leaving end-user just with the management of a trivial configuration file. A high level of automation is obtained, spanning from the services deployment and configuration up to the self-healing and auto-scaling features. In the context of the presented project a prototype of the OpenStack HEAT [10] template has been developed, always using Ansible as a solution for the underlying system configuration. The TOSCA migration will be done as soon as the integration of the AMS services and components will be finished.

Soon after the TOSCA based orchestration will be completed the plan is to start the integration and the testing of the PaaS Orchestrator, the Infrastructure Manager, and the Identity and Access Manager. The first two represent the resource abstraction layer, a key to the open interoperation across CLOUD solutions; the latter is the pillar of authentication, authorization and delegation mechanism, adopted to securely tie modern federations and well established computing models, in particular for the AMS workflow this will be a key for the remote data access.

4. Conclusions

In the present paper, we described the deployed Cloud infrastructure, a multi-sites Cloud implemented with the specific purpose of harvesting dispersed computational resources We have shown how to use the OpenStack middleware to manage dispersed resources and to provide a modern paradigm of computing and data usage access. We illustrated, via real test cases, how to build a geographically distributed cloud with the specific purpose of sharing and managing computing and storage resources, owned by heterogeneous cooperating entities

We were able to link together four different entities, different both in terms of basic computational resources as well as in terms of scientific interest, overcoming portability, interoperability and compatibility issues. Our work opens interesting scenarios and use cases especially in load balancing, disaster recovery and advanced features such as cross-site networks, cross-site storage systems, scheduling and placement of remote VMs. Moreover, the deployed Cloud infrastructure allows the exploitation of new way of cooperating and resources sharing among very different scientific communities.

References

[1] A. Shrivastwa, S. Sarat, K. Jackson, C. Bunch, E. Sigler, and T. Campbell, *OpenStack: Building a Cloud Environment*. Packt Publishing Ltd, 2016.

[2] G. Aceto, A. Botta, W. De Donato, and A. Pescapè, “Cloud monitoring: A survey,” *Computer Networks*, vol. 57, no. 9, pp. 2093–2115, 2013.

[3] P. Mell, T. Grance, *et al.*, “The nist definition of cloud computing,” 2011.

[4] M. Mahalingam, D. Dutt, K. Duda, P. Agarwal, L. Kreeger, T. Sridhar, M. Bursell, and C. Wright, “Virtual extensible local area network (vxlan): A framework for overlaying virtualized layer 2 networks over layer 3 networks,” tech. rep., 2014.

[5] “Ubuntu linux.” http://www.ubuntu.com. Accessed: 2018-04-01.

[6] “Openvswitch.” https://www.openvswitch.org/. Accessed: 2018-04-01.

[7] “Openvpn.” https://openvpn.net/. Accessed: 2018-04-01.

[8] N. Doraswamy and D. Harkins, *IPSec: the new security standard for the Internet, intranets, and virtual private networks*. Prentice Hall Professional, 2003.

[9] “Openflow.” https://en.wikipedia.org/wiki/OpenFlow. Accessed: 2018-04-01.

[10] “Openstack heat.” https://wiki.openstack.org/wiki/Heat. Accessed: 2018-04-01.

[11] L. Hochstein and R. Moser, *Ansible: Up and Running: Automating Configuration Management and Deployment the Easy Way*. " O’Reilly Media, Inc.", 2017.

[12] “Ams.” http://www.ams02.org/. Accessed: 2018-04-01.

[13] K. Lübelsmeyer, A. S. V. Dratzig, M. Wlochal, G. Ambrosi, P. Azzarello, R. Battiston, R. Becker, U. Becker, B. Bertucci, K. Bollweg, J. Burger, F. Cadoux, X. Cai, M. Capell, V. Choutko, M. Duranti, C. Gargiulo, C. Guandalini, S. Haino, M. Ionica, A. Koulemzine, A. Kounine, V. Koutsenko, G. Laurenti, A. Lebedev, T. Martin, A. Oliva, M. Paniccia, E. Perrin, D. Rapin, A. Rozhkov, S. Schael, H. Tholen, S. Ting, and P. Zuccon, “Upgrade of the alpha magnetic spectrometer (ams-02) for long term operation on the international space station (iss),” *Nucl. Instr. Meth. Phys. Res. A*, vol. 654, pp. 639–648, 2011.

[14] “The alpha magnetic spectrometer (ams) on the international space station: Part i âĂŞ results from the test flight on the space shuttle,” *Physics Reports*, vol. 366, no. 6, pp. 331 – 405, 2002.

[15] A. Szabo and N. S. Ostlund, *Modern quantum chemistry: introduction to advanced electronic structure theory*. Courier Corporation, 2012.

[16] G. t. Te Velde, F. M. Bickelhaupt, E. J. Baerends, C. Fonseca Guerra, S. J. van Gisbergen, J. G. Snijders, and T. Ziegler, “Chemistry with adf,” *Journal of Computational Chemistry*, vol. 22, no. 9, pp. 931–967, 2001.

[17] M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, G. Scalmani, V. Barone, G. A. Petersson, H. Nakatsuji, X. Li, M. Caricato, A. V. Marenich, J. Bloino, B. G. Janesko, R. Gomperts, B. Mennucci, H. P. Hratchian, J. V. Ortiz, A. F. Izmaylov, J. L. Sonnenberg, D. Williams-Young, F. Ding, F. Lipparini, F. Egidi, J. Goings, B. Peng, A. Petrone, T. Henderson, D. Ranasinghe, V. G. Zakrzewski, J. Gao, N. Rega, G. Zheng, W. Liang, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, T. Vreven, K. Throssell, J. A. Montgomery, Jr., J. E. Peralta, F. Ogliaro, M. J. Bearpark, J. J. Heyd, E. N. Brothers, K. N. Kudin, V. N. Staroverov, T. A. Keith, R. Kobayashi, J. Normand, K. Raghavachari, A. P. Rendell, J. C. Burant, S. S. Iyengar, J. Tomasi, M. Cossi, J. M. Millam, M. Klene, C. Adamo, R. Cammi, J. W. Ochterski, R. L. Martin, K. Morokuma, O. Farkas, J. B. Foresman, and D. J. Fox, "GaussianËIJ16 Revision B.01," 2016. Gaussian Inc. Wallingford CT.

[18] L. Storchi, S. Rampino, L. Belpassi, F. Tarantelli, and H. M. Quiney, "Efficient parallel all-electron four-component dirac–kohn–sham program using a distributed matrix approach ii," *Journal of chemical theory and computation*, vol. 9, no. 12, pp. 5356–5364, 2013.

[19] L. Belpassi, L. Storchi, H. M. Quiney, and F. Tarantelli, "Recent advances and perspectives in four-component dirac–kohn–sham calculations," *Physical Chemistry Chemical Physics*, vol. 13, no. 27, pp. 12368–12394, 2011.

[20] I. Foster, C. Kesselman, *et al.*, "The grid 2: Blueprint for a future computing infrastructure," *Waltham: Morgan Kaufmann Publishers*, 2004.

[21] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[22] L. Storchi, C. Manuali, O. Gervasi, G. Vitillaro, A. Laganà, and F. Tarantelli, "Linear algebra computation benchmarks on a model grid platform," in *International Conference on Computational Science*, pp. 297–306, Springer, 2003.

[23] L. Storchi, F. Tarantelli, and A. Laganà, "Computing molecular energy surfaces on a grid," in *International Conference on Computational Science and Its Applications*, pp. 675–683, Springer, 2006.

[24] S. Rampino, L. Storchi, and A. Laganà, "Automated simulation of gas-phase reactions on distributed and cloud computing infrastructures," in *International Conference on Computational Science and Its Applications*, pp. 60–73, Springer, 2017.

[25] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," *Linux Journal*, vol. 2014, no. 239, p. 2, 2014.

[26] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *Proceedings of the April 18-20, 1967, spring joint computer conference*, pp. 483–485, ACM, 1967.

[27] S. Rampino, L. Belpassi, F. Tarantelli, and L. Storchi, "Full parallel implementation of an all-electron four-component dirac–kohn–sham program," *Journal of chemical theory and computation*, vol. 10, no. 9, pp. 3766–3776, 2014.

[28] L. Storchi, L. Belpassi, F. Tarantelli, A. Sgamellotti, and H. M. Quiney, "An efficient parallel all-electron four-component dirac- kohn- sham program using a distributed matrix approach," *Journal of chemical theory and computation*, vol. 6, no. 2, pp. 384–394, 2010.

[29] "Apache mesos." http://mesos.apache.org/. Accessed: 2018-04-01.

[30] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. H. Katz, S. Shenker, and I. Stoica, “Mesos: A platform for fine-grained resource sharing in the data center.,” in *NSDI*, vol. 11, pp. 22–22, 2011.

[31] “Python.” https://www.python.org/. Accessed: 2018-04-01.

[32] P. J. Karol, R. C. Barber, B. M. Sherrill, E. Vardaci, and T. Yamazaki, “Discovery of the element with atomic number z= 118 completing the 7th row of the periodic table (iupac technical report),” *Pure and Applied Chemistry*, vol. 88, no. 1-2, pp. 155–160, 2016.

[33] K. G. Dyall, “Relativistic double-zeta, triple-zeta, and quadruple-zeta basis sets for the 5d elements hf–hg,” *Theoretical Chemistry Accounts*, vol. 112, no. 5-6, pp. 403–409, 2004.

[34] K. G. Dyall and A. S. Gomes, “Revised relativistic basis sets for the 5d elements hf–hg,” *Theoretical Chemistry Accounts*, vol. 125, no. 1-2, p. 97, 2010.

[35] K. G. Dyall, “Core correlating basis functions for elements 31–118,” *Theoretical Chemistry Accounts*, vol. 131, no. 5, p. 1217, 2012.

[36] K. G. Dyall, “Relativistic double-zeta, triple-zeta, and quadruple-zeta basis sets for the 4d elements y–cd,” *Theoretical Chemistry Accounts*, vol. 117, no. 4, pp. 483–489, 2007.

[37] L. Belpassi, F. Tarantelli, A. Sgamellotti, and H. M. Quiney, “Electron density fitting for the coulomb problem in relativistic density-functional theory,” *The Journal of chemical physics*, vol. 124, no. 12, p. 124104, 2006.

[38] S. Rampino, L. Storchi, and L. Belpassi, “Gold–superheavy-element interaction in diatomics and cluster adducts: A combined four-component dirac-kohn-sham/charge-displacement study,” *The Journal of chemical physics*, vol. 143, no. 2, p. 024307, 2015.

[39] A. D. Becke, “Density-functional exchange-energy approximation with correct asymptotic behavior,” *Physical review A*, vol. 38, no. 6, p. 3098, 1988.

[40] C. Lee, W. Yang, and R. G. Parr, “Development of the colle-salvetti correlation-energy formula into a functional of the electron density,” *Physical review B*, vol. 37, no. 2, p. 785, 1988.

[41] T. E. Oliphant, *A guide to NumPy*, vol. 1. Trelgol Publishing USA, 2006.

[42] B. K. Shoichet, “Virtual screening of chemical libraries,” *Nature*, vol. 432, no. 7019, p. 862, 2004.

[43] F. Milletti, L. Storchi, G. Sforna, and G. Cruciani, “New and original p k a prediction method using grid molecular interaction fields,” *Journal of chemical information and modeling*, vol. 47, no. 6, pp. 2172–2181, 2007.

[44] F. Milletti, L. Storchi, G. Sforna, S. Cross, and G. Cruciani, “Tautomer enumeration and stability prediction for virtual screening on large chemical databases,” *Journal of chemical information and modeling*, vol. 49, no. 1, pp. 68–75, 2009.

[45] O. Trott and A. J. Olson, “Autodock vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading,” *Journal of computational chemistry*, vol. 31, no. 2, pp. 455–461, 2010.

[46] N. Triballeau, F. Acher, I. Brabet, J.-P. Pin, and H.-O. Bertrand, “Virtual screening workflow development guided by the âĂIJreceiver operating characteristicâĂİ curve approach. application to high-throughput docking on metabotropic glutamate receptor subtype 4,” *Journal of medicinal chemistry*, vol. 48, no. 7, pp. 2534–2547, 2005.

[47] E. M. Fajardo, J. M. Dost, B. Holzman, T. Tannenbaum, J. Letts, A. Tiradani, B. Bockelman, J. Frey, and D. Mason, “How much higher can htcondor fly?,” *Journal of Physics: Conference Series*, vol. 664, no. 6, p. 062014, 2015.

[48] R. Brun and F. Rademakers, “RootâĂŤan object oriented data analysis framework,” *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 389, no. 1-2, pp. 81–86, 1997.

[49] “Cern the european organization for nuclear research.” https://home.cern/. Accessed: 2018-04-01.

[50] D. Salomoni, I. Campos, L. Gaido, G. Donvito, M. Antonacci, P. Fuhrman, J. Marco, A. Lopez-Garcia, P. Orviz, I. Blanquer, *et al.*, “Indigo-datacloud: foundations and architectural description of a platform as a service oriented to scientific computing,” *arXiv preprint arXiv:1603.09536*, 2016.

[51] “Marathon.” https://mesosphere.github.io/marathon/. Accessed: 2018-04-01.

[52] “squid.” http://www.squid-cache.org/. Accessed: 2018-04-01.

[53] “Tosca.” http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.html. Accessed: 2018-04-01.