Scalable cloud without dedicated storage

D.V. Batkovich[1], **M.V. Kompaniets, A.K. Zarochentsev**

Department of Theoretical Physics, St. Petersburg State University, Uljanovskaja 1, St. Petersburg, Petrodvorez, 198504 Russia

E-mail: `batya239@gmail.com, mkompan@gmail.com, andrey.zar@gmail.com`

Abstract. We present a prototype of a scalable computing cloud. It is intended to be deployed on the basis of a cluster without the separate dedicated storage. The dedicated storage is replaced by the distributed software storage. In addition, all cluster nodes are used both as computing nodes and as storage nodes. This solution increases utilization of the cluster resources as well as improves fault tolerance and performance of the distributed storage. Another advantage of this solution is high scalability with a relatively low initial and maintenance cost. The solution is built on the basis of the open source components like OpenStack, CEPH, etc.

1. Introduction

The Large Hadron Collider (**LHC**) is preparing for its second three-year run. Cool down of the vast machine has already begun in preparation for research to resume early in 2015 following a first long shutdown to prepare the machine for running at almost double energy in comparison to Run 1 . **ALICE** as the one of **LHC** experiments will need to process a greater amount of data. Even greater energy and hence data flow is planned for Run 3.

ALICE collaboration has plans to add a cloud infrastructure to the **GRID** facility before the start of Run 3. Cloud computing brings many positive aspects because it allows more efficient resource utilization and easy deployment methods. On one hand this allows one to balance the load between the projects, for example, you can easily balance the load between different LHC projects, or during **LHC** shutdowns you can use your cloud for different projects without having to re-install the software. On the other hand it is possible to easily maintain a number of different software versions for a long term data preservation.

At the present time **ALICE** infrastructure contains only **Tier1** and **Tier2** sites, this situation is connected with the fact that installation and configuration of **GRID** infrastructure is non trivial process that requires qualified personnel. At the same time there are a lot of small research groups which would like to contribute into **ALICE** IT infrastructure but do not satisfy even **Tier2** requirements. The main problem for such groups is that they have

- limited computational resources,
- limited finances,
- and lack of qualified IT personnel.

In this paper we offer a solution for **Tier3** sites [1] based on open-source software **OpenStack** [2] and **CEPH** [3]. This solution on one hand is intended to be easily installable,

[1] Speaker

maintainable and requires no highly qualified IT personnel, and on the other hand this solution is assumed to be scalable enough to be able to maintain growing amount of data.

2. Requirements for Tier3 clouds

The main requirement for all **TierX** clouds is to have a uniform interface. In our case **AMAZON EC2 API** [4] is chosen, due to the fact that main deployment method for **ALICE** IT infrastructure is assumed to be using **CernVM 3/CVMFS** [5, 6].

Because of the fact that **Tier3** sites have restrictions mentioned above (limited finances/computational resources), it is obvious that such a solution must be deployed on commodity hardware.

Another requirement is that all installation and maintenance procedures must be as easy as possible to allow non professionals to deploy the cloud.

The last requirement is that system must be at least minimally fault tolerant. **Tier3** sites are not supposed to store any significant data (only cache for data analysis), this means that cloud downtime and data loss must be minimized, but are not critical.

3. Storage configuration

One of the the cloud bottlenecks is a storage. The cheapest cloud solutions assume that there is some node which is used as a storage back-end (for example **OpenStack Cinder**[2] on **LVM**[3]). Such a solution cannot be considered as reliable one, because this is a single point of failure for storage and this configuration is not scalable enough due to concurrent IO from computation nodes.

As alternative one may consider some dedicated storage (hardware or software defined one). Hardware defined storage is very expensive and therefore not suitable for **Tier3** sites. On the other hand software defined storage shows reliable performance and fault tolerance. Another big advantage of the software defined storage is the number of available open-source solutions.

As a storage back-end for our solution we have chosen **CEPH** [3] based on following two considerations: first of all, **CEPH** is easily deployed software defined storage that provides distributed block devices which are extremely suitable for cloud infrastructure, second is that **CEPH** is already used in **CERN** as a storage back-end for **OpenStack**.

Additional positive aspects come from the fact that we want to maximally utilize resources and make our system as cheap as possible. As it was mentioned above, it is assumed that we run our cloud on commodity hardware (which implies that on each node we have the similar standard configuration including some CPU, RAM and HDD). The idea of the additional improvement comes from the fact that computation nodes are almost completely not using their HDD, while almost all storage nodes do not fully utilize their CPU. This leads us to the idea to combine computation and storage nodes. On one hand this allows us to utilize more disk space and make distributed storage more fast and fault tolerant, on the other hand this allows us to utilize CPU's from the storage nodes. All this allows one to utilize cluster resources more efficiently, but of course some careful tuning is still required to make computation and storage nodes work together on the same host without significant impact on performance. This leads us to the next layout of our configuration:

- Management servers: in most cases all of management responsibilities can be placed on one server. But due to the complex cloud network configuration and security requirements it was decided to setup the network management on a separate server. This server is also configured as a gateway between internal cloudnetwork and internet. As a result the management services are located on two separate hardware nodes.

[2] Cinder is a Block Storage service for OpenStack

[3] Logical Volume Manager. Cinder on LVM is most simple configuration of Block Storage service for OpenStack

- Worker(Computation/Storage) nodes: There must be at least 3 storage nodes (minimal **CEPH** configuration, see [3]). It is assumed, that when we setting up the worker nodes, we install both computation engine and storage engine, so that the number of computation nodes in the cloud infrastructure is the same as the number of storage nodes in the distributed storage. Of course, there is no reasonable limitation that requires to keep exactly the same number of computation and storage nodes. This configuration was considered in order to simplify the deployment procedure. In general case one can setup some nodes which run computation and storage engines simultaneously, and some dedicated nodes which run either computation or storage engine, but such a configuration is much more complicated than a homogeneous one.
- Network facilities: To setup cloud with **CEPH** back-end we need two separate networks. First is standard internal cloud network which is used as management network, network for VM's interconnect and as transport for accessing block devices. The second one is separate **CEPH** network for data replication. Additionally **CernVM3** requires access to **CVMFS** via caching proxy.

Among the existing cloud platforms we have chosen **CloudStack** and **OpenStack**. It should be noted that these solutions not just satisfied our requirements, but **OpenStack** was chosen because it is already used at **CERN** and has big community and support from corporate vendors, and **CloudStack** was chosen as a relatively simple and easily installable cloud solution. Both of them are free and do support **AMAZON EC2 API** and HTTPS (required to secure external connections to cloud infrastructure). Both cloud solutions have built-in **CEPH** support as well.

4. Choice of cloud platform

Historically **CloudStack** [7] was the first cloud platform we started to use with **CEPH**, but we found a lot of problems with setting it up in the required configuration. For example, it is not possible to configure **CloudStack AMAZON EC2 API** bridge in SSL mode. Additionally, automated deployment procedure (from XML-description) is non-functional since over a year now. Apart from that, there should be noted that CloudStack cannot be configured completely from the command line interface (CLI) and requires some manipulations in the web console. The main problem with **CloudStack** is that, in order to create a deployment script, one needs either to fix a number of **CloudStack** internal bugs or to write some nontrivial workarounds which allow one to setup CloudStack in required configuration without user intervention. Due to the facts mentioned above we have decided to continue to create our solution only with OpenStack as the cloud platform.

In contrary to **CloudStack**, **OpenStack** can be easily deployed from CLI, and installation procedure may be fully automated using **Packstack** [8] package based on puppet [9]. Additionally, there is no problem with configuring all external interfaces to use SSL.

OpenStack configuration we use consists from standard components **Glance** (image service), **Nova** (compute service), **Cinder** (block storage service), **Neutron** (networking service)[4] (see [2] for details). Neutron service can operate using different plugins, in our case we use default the Modular Layer 2 (**ML2**) plugin.

To estimate efficiency of proposed approach three different **OpenStack** configurations were created. All configurations has 3 compute nodes the difference is only in **Cinder** configuration:

(Conf. 1) Dedicated storage on **LVM**: **Cinder** is configured with **LVM** back-end on dedicated server

[4] It should be noted that OpenStack developers recommend to switch to Neutron networking service from legacy Nova networking.

(Conf. 2) Dedicated distributed storage: **Cinder** is configured with **CEPH** back-end running on additional 3 nodes

(Conf. 3) Distributed storage on compute nodes: **Cinder** is configured with **CEPH** back-end running on the compute nodes

Packstack includes modules for the most of **OpenStack** components. It means that at least 1st configuration (**Cinder** on **LVM**) could be deployed automatically. This can be easily done with one remark: current version of **ML2 Packstack** module has some bugs which do not allow to setup **ML2** networking correctly. To solve this problem we used custom shell scripts. It is also possible to install **Cinder** on **CEPH** using **Packstack**, but due to the fact that we want to setup computation and storage engine on the same worker node, we need a number of additional scripts which later on will be organized into a **Packstack** module.

5. Tests and results

As a performance test we have chosen IO benchmark **Bonnie++** [10]. It allows one to test disk and file system performance (block IO, latency, ...). We ran the tests for three **Cinder** configurations mentioned in the previous section[5]. Results are presented on Fig.1

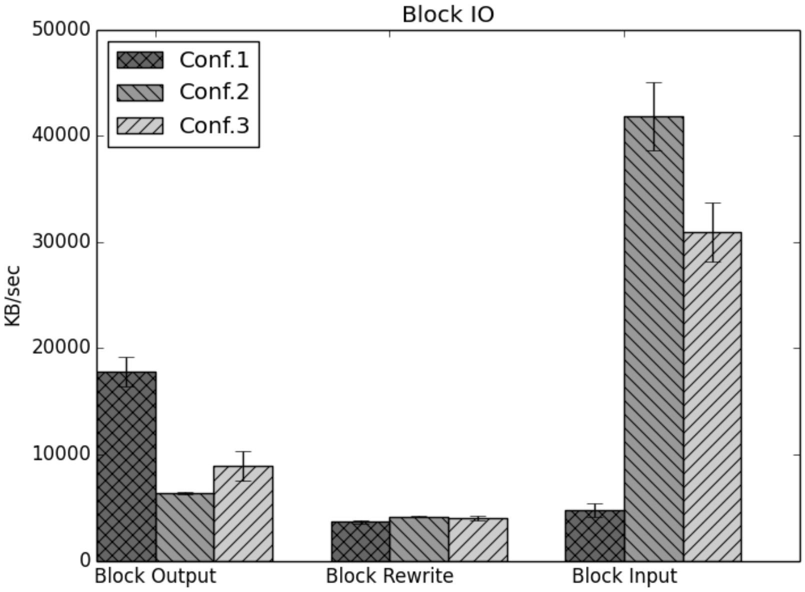

Figure 1. IO performance comparison for different configurations. Conf.1: Dedicated storage on **LVM**, Conf.2: Dedicated distributed storage, Conf.3: Distributed storage on compute nodes

At the y-axis on Fig. 1 one can see the IO speed in KB/sec. On the x-axis we have three groups: block output, block rewrite and block input. Bonnie++ block output is actually system's performance on write operations, block input - on read operations, and block rewrite is symmetric read and write test. Of course this is a synthetic test and it loads only disk IO (more realistic tests must load not only disk IO, but CPU and network also), but this benchmark shows some interesting tendencies.

[5] For tests we used Cisco Catalyst 4948 with 1Gb network and 4 Supermicro Twins. Each twin contains two servers with the following configuration: CPU 2 x Intel Xeon(R) E5420; RAM 16 GB; HDD Seagate Barracuda ES.2 250 Gb SATA-II.

At the "block output" section **Cinder** on **LVM**(conf.1) has best performance, due to the fact that in **CEPH** write operation is completed when data is replicated over whole **CEPH** cluster. While in "block input" we have directly opposite picture: **Cinder** on **CEPH** (conf.2 and 3) shows much better performance. And finally, for symmetric reads and writes (block rewrite) advantages of fast reads for **CEPH** seems to be compensated by slow writes.

As for different configurations with **CEPH**: dedicated distributed storage (conf.2) and distributed storage on compute nodes (conf.3) one can see that difference between performance is close to errors of measurements so we can say that these configurations have approximately the same IO performance.

As the summary we can say that with proper resource allocation between **CEPH** and **OpenStack** that runs on the same nodes we can achieve performance similar to the performance of standalone **CEPH** but with lower cost. This optimal ratio can be different for specific problems. Hence for the next step we have plans to deploy cluster following proposed scheme and to run real problem instead of synthetic **Bonnie++** tests. Moreover we must perform high load runs and test all of 3 configurations on different hardware.

6. Acknowledgments

The authors are grateful to Predrag Buncic for helpfull discussions and advises, to ACAT'14 Organizing Committee for support and hospitality, to Saint-Petersburg State University for a research grant 11.38.197.2014.

References

[1] The Grid: A system of tiers `http://home.web.cern.ch/about/computing/grid-system-tiers`
[2] OpenStack `http://www.openstack.org/`
[3] CEPH `http://ceph.com/`
[4] Amazon EC2 `http://aws.amazon.com/documentation/ec2/`
[5] CernVM File System `http://cernvm.cern.ch/portal/filesystem`
[6] CernVM 3 and mCernVM Beta Release `http://cernvm.cern.ch/portal/ucernvm`
[7] CloudStack `http://cloudstack.apache.org/`
[8] Packstack `https://wiki.openstack.org/wiki/Packstack`
[9] Puppet `http://puppetlabs.com/`
[10] Bonnie++ `http://www.coker.com.au/bonnie++/`