An Alternative Solution to AMAZON Software Defined Object Storage[1]

CHIRANA-GHEORGHIŢĂ, Eugeniu-Theodor
Spiru Haret University
Faculty of Engineering, Computer Science and Geography
office@adaptcom.ro

IACOB, Nicoleta Magdalena
Spiru Haret University
Scientific Research Center in Mathematics and Computer Science
nicoleta.iacob_2007@yahoo.com

Abstract

Scality Ring is a complete hardware independent solution which can use almost all hardware on the market as the underlying layer to provide access to data in a cloud by different means in a fast and safe manner. The ring solution was designed from the beginning with scalability in mind, also offering high availability and data integrity.

The solution has no single point of failure and it uses 3 major components (Nodes, Connectors and Supervisor) to make it very scalable at any level. Although these components are completely independent they are interconnected through different protocols. They provide data access through RS2, SOFS, SMB or NFS, thus enabling proper data access to different type of servers.

Keywords: *Ring, software defined storage, reliability, cloud storage.*

ACM/AMS Classification: 97P30

1. *Presentation*

Scality is a company that created a solution for Object Cloud Storage named ”Scality Ring” [5] and according to company's data is serving now more than 500 million users worldwide. The solution provide 100 % reliability and high performance and is very cost effective being able to use, as the underlying hardware, a combination of different servers which can be provided by different vendors[2] like Dell, HP, IBM. The Ring can use SAS and SATA disks to store actual data in form of objects and SSD's for it's metadata part.

[1]Citation: Chirana-Gheorghiţă E.-T., Iacob N.M., *An Alternative Solution to AMAZON Software Defined Object Storage*, ”An. Univ. Spiru Haret, Ser. Mat.-Inform.”, 12(2), pp. 5-10, 2016.

[2]All product names, trademarks and registered trademarks are property of their respective owners.

2. *The Ring – the software that controls the hardware*

The software solution proposed by Scality is a SDS – Software Defined Storage that was designed from the beginning to be able to use the new SDDC – Software Defined Datacenter Concepts. The solution is able to provide access to petabyte storage. It is also very scalable and it enables the use of object-based or file-based applications. The Ring can function on a minimum of 6x86 servers (physical nodes) with only 6 node instances each (logical nodes) up to hundreds of physical servers (each with 6 or more logical node instances on it) and so providing a total storage capacity of tens or hundreds of petabytes (figure 1).

The physical servers that serve as the underlying layers of the Ring can be of a vast density, working with servers that have just a few disks up to the large high density servers (HP Apollo, for example) that use hundreds of disks and SSD's. The software solution makes an abstractization of all the hardware that it controls and exposes to the clients/applications the type of storage they require at the speed required ([1],[2],[3]).

The solution was built to have no SPOF – "Single Point of Failure" with no downtime in case of hardware failure or if the client wants to extend the capacity or upgrade the software version to the latest available.

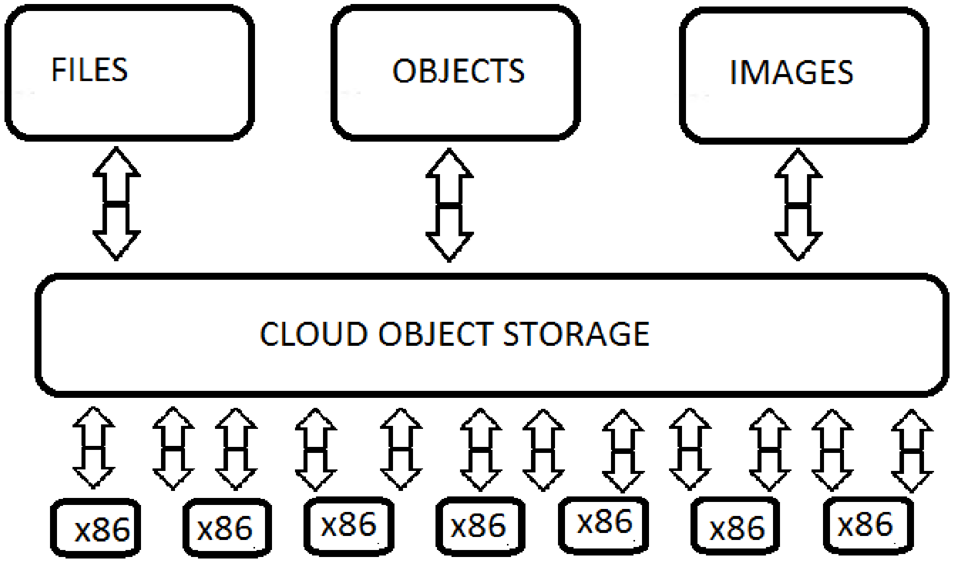


Figure 1. *System architecture*

At the operating system level, the Ring can be installed on most of the popular Linux distributions having no requirements of exotic filesystems. It uses plain ext3 or ext4 filesystems with little or no tuning at all to the kernel side. Because of no HCL's – "Hardware Compatibility Lists" (those are maintained by the software provider), the only compatibility that has to be considered is the compatibility between the Linux distribution chosen and the hardware that is installed onto.

3. ***The design of the solution***

The Scality Ring is created to support uses similar to Amazon or Google storage clouds enabling applications of users to seamlessly migrate from or towards. The key features are:

- Data and Metadata distributed design, enabling the scaling of the capacity and performance having no SPOF and no service interruptions;
- Supports multiple protocols enabling applications to use a great range of access methods to the actual storage capacity like object or file based;
- Multiple data protection mechanisms adapted to the specific data that has to be protected;
- In case of a failure, the Ring self-heals and resolves the issues until they are corrected.

4. ***The architecture***

For being able to support both scalability and performance, the solution is designed as a distributed and scale-out architecture. The software also provides services for data access and presentation, data protection and the management and monitoring of the systems.

The ring has multiple components, each providing and serving a different scope. Each Scality Ring is composed of 3 basic elements that are the same regardless of the size, geographic layout and physical placement: storage nodes, connectors and one or more supervisors.

4.1. **Storage Nodes and I/O Daemon Processes**

These components are the core of the solution. They are hardware X86 servers, usually having a Linux operation system installed, many hard drives and SSD's, proper amount of RAM and CPU capacity. The speed of the network cards is crucial for obtaining proper performance, usually there are 2 or more 10G fiber cards aggregated with LACP network protocol.

Each server runs 6 or more logical storeNode processes that manage a defined and balanced amount of keys which aggregate in the total KeySpace range defined at installation. The key is the basic addressing unit (similar to the common inode in classical filesystems) and each node manages a predefined number of these keys. At node level run processes concerned of key integrity, key deduplication and also processes that communicate with other nodes for the Ring to be able to know, at every given time, where a specific piece of data is located.

Each node receives data in form of objects or files from the Connectors which provide the top-level access to these different protocols that are supported by the different client-side applications.

The node also monitors the health and integrity of the keyspace assigned to it and, in case of failure, sends distress messages and other nodes in the ring can take the decision to change the ownership of specific data parts so that the end-client receives the data requested. This mechanism is possible because every piece of data is present on multiple parts of the ring and can be reconstructed immediately. When the issue with the node that has been affected by any event is solved, the data is also internally reconstructed to assure the number of copies is the same as it was before the event. This is the self-healing characteristic of the ring.

Node processes communicate directly with the physical disks and write data directly on them. The process that writes the data is responsible for maintaining the metadata table for each object that is stored into the ring, so that when a request to retrieve that data is received the ring knows exactly where it is and delivers it fast.

4.2. **Connectors**

The Connectors are also software daemons that run on dedicated physical servers and provide the interface between applications and data by providing access points through different protocols to the applications. These servers do not need the density of the hard drives as Nodes do, but they require fast network access, a considerable amount of RAM and CPU. On these servers specific daemons run, typically more that on the node servers. On each server of this type we can have 40 or more connector instances, each binded to a specific port.

The Connectors receive requests of put, get and delete from applications and pass them to the nodes. The nodes process the data and return the result to the connectors which in turn deliver the response to the application.

There are multiple types of connectors each of them serving data to the applications in the requested form:

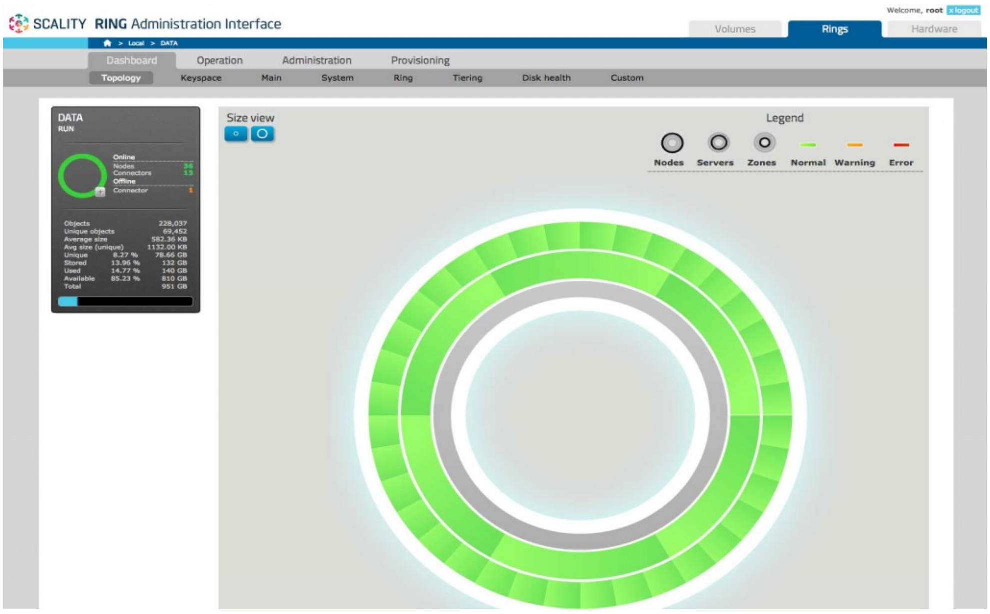
- **Rest or RS2 connectors**
  This type of connector works on HTTP/REST protocol and provide access to data via simple PUT/GET/DELETE HTTP messages. There are many implementations of this type of connectors, the most common being the S3 protocol and the Openstack Swift driver. The S3 type of connectors uses similar Amazon AWS S3 technology and data is stored in bucket containers. The S3 connector provides advanced data storage and management. The Openstack Swift driver provides a interface for the Openstack suite of software and enables a Openstack cloud to use this storage as it's underlying storage layer protected by the ring's recovery and self-healing mechanisms.
- **SOFS connectors**
  This type of connector provide Scale-Out-File-System support which is a POSIX based filesystem so it can provide file storage support without the need for external file server gateway. A good example of an application that uses such a connector is the s3fs which can provide a normal mount point to the operation system. The application will use the same file storage techniques as in a normal filesystem. The SOFS connectors provide full support for protocols such as NFS or SMB.
- **FUSE connectors**
  FUSE filesystems are supported in all major Linux and Windows distributions. This type of connector support full quotas, as well as parallel IO operations.
- **NFS connectors**
  NFS is supported through this type of connector as a standard client interface on almost all filesystems and operation systems [4]. TCP or UDP transport protocols are fully supported and so data is accessed in the pure NFS way by

the clients with no need for them to be aware of what lies behind this type of connector.

- **SMB connectors**
  Server Message Block (SMB) applications can use this type of connector via servers running Samba or CIFS connection methods.

4.3. **The Supervisor**

The supervisor is a standalone server which provides a monitoring and management GUI to the ring. It is an independent server and the ring can function without a Supervisor, but with limited administration and configuration capabilities, although the performance and data availability is not affected. In geographic distributed systems there can be more than one Supervisor, each of them managing a separate datacenter. The Supervisor services can be accessed by a Graphical User Interface (GUI) or using a Command Line Interface (CLI) named ringsh. The GUI is the one below (figure 2):



DATA
RUN

Online
Nodes 36
Connectors 13
Offline
Connector 1

```
Objects                        228,037
Unique objects                  69,452
Average size                 582.36 KB
Avg size (unique)           1132.00 KB
Unique          8.27 %        78.66 GB
Stored         13.96 %          132 GB
Used           14.77 %          140 GB
Available      85.23 %          810 GB
Total                           951 GB
```

Figure 2. *Graphical User Interface*

5. ***Conclusion***

This platform provides high reliability and performance and is designed as a distributed and scale-out architecture. The concept of Software Defined Storage was implemented in this software solution in order to be able to provide access to petabyte storage with the lowest hardware costs.

References

1. C.L. (Ciobanu) Defta, *Wireless LAN Security - WPA2-PSK Case Study*, ”Global Journal on Technology since”, Vol. 1 / 2012 / pp. 62-67.
2. M. Pirnau, *The analysis of the. NET architecture security system*, ”Computers and Artificial Intelligence (ECAI)”, 2013.
3. C. Pirnau, M.A. Botezatu, I.S. Grigorescu, *Databases role correlated with knowledge transfer between entities of a cluster,* ”Mircea cel Batran Naval Academy Scientific Bulletin”, Volume XVIII 2016 Issue 2, Constanta, Romania, pp. 111.
4. https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6 /html/Storage_Administration_Guide/ch-nfs.html, accesed 2016.
5. www.scality.com, accesed 2016.