Monitoring Checklist for Ceph Object Storage Infrastructure

Pragya Jain, Anita Goel, S. Gupta

To cite this version:

Pragya Jain, Anita Goel, S. Gupta. Monitoring Checklist for Ceph Object Storage Infrastructure. 5th International Conference on Computer Science and Its Applications (CIIA), May 2015, Saida, Algeria. pp.611-623, 10.1007/978-3-319-19578-0_50 . hal-01789979

HAL Id: hal-01789979

https://hal.inria.fr/hal-01789979

Submitted on 11 May 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution| 4.0 International License

Monitoring Checklist for Ceph Object Storage Infrastructure

Pragya Jain[1], Anita Goel[2], S. C. Gupta[3]

[1] Department of Computer Science, University of Delhi, Delhi, India
[2]Department of Computer Science, Dyal Singh College, University of Delhi, India
[3] Department of Computer Science, IIT, Delhi, India
prag_2648@yahoo.co.in,
{goel.anita, gupta.drsc}@gmail.com

Abstract. Object storage cloud is widely used to store unstructured data like photo, emails, video etc. generated from use of digital technologies. The number of object storage services has increased rapidly over the years and so is increased the complexity of the infrastructure behind it. Effective and efficient monitoring is constantly needed to properly operate and manage the complex object storage infrastructure. Ceph is an open source cloud storage platform that provides object storage as a service. Several works have discussed ways to collect the data for monitoring. However, there is little mention of what needs to be monitored. In this paper, we provide an infrastructure monitoring list for Ceph object storage cloud. We analyze the Ceph storage infrastructure and its processes for identifying the proposed lists. The infrastructure monitoring list allows selecting requirements, in contrast to, specifying fresh requirements, for monitoring. The monitoring list helps developer during requirement elicitation of the monitoring functionality when developing a new tool or updating an existing one. The checklist is also useful during monitoring activity for selecting parameters that need to be monitored by the system administrator.

Keywords: Cloud Object Storage, Infrastructure Monitoring, Ceph.

1 Introduction

The mass adoption and increasing popularity of digitization technologies has resulted in generation of data in form of videos, photo, blogs, emails, messages, chat data etc. Object storage cloud is a widely adopted paradigm for storing these voluminous data over the Internet. Ceph 0 is open source cloud storage for storing data as object.

The services provided to the subscribers for object storage solutions have rapidly increased and so has complexity of the underlying infrastructure. Monitoring is necessary in cloud to determine health of system and is beneficial for both service provider and consumer 000. There is a need to scale storage nodes, detect and repair failures, manage load surge and improve performance. Due to the elastic nature of cloud, there is a need to constantly monitor the infrastructure at runtime 0 to optimize the use of storage infrastructure with varying demand for storage. Also, disruption in system performance due to reasons like, node failure, system crash, network error, high memory load etc. requires monitoring during runtime. Moreover, processes for storing activity defined in Ceph require monitoring to detect any erroneous action. Furthermore, since Ceph is integrated with many popular clouds, like, Openstack and Eucalyptus for providing storage as a service, defining of monitoring functionality is essential for determining its proper working.

In Ceph, monitoring functionality is incorporated in different ways – (1) Commands are available to monitor storage cluster, (2) Existing freely available infrastructure monitoring tools, like, Nagios are adapted to suit need of Ceph, or (3) New code is written to include infrastructure monitoring functionality. Generally, freely available infrastructure monitoring tools are used for monitoring Ceph.

Several researchers have discussed different architectures 000000 for monitoring cloud for specific purposes. The focus is mainly on efficient ways of collecting and analyzing data. But, none of them address the issue of what needs to be monitored in object storage cloud. Several tools exist that support monitoring of specific features, like, Calamari monitors Ceph cluster, CollectD and Zabbix monitor system performance, Nagios monitors status of resources, Munin monitors storage capacity. Although the tools specify functionality it supports, there is no mention of requirement specification for the monitoring of Ceph.

In this paper, focus is on creation of the requirement specification for infrastructure monitoring of Ceph from the system administrator perspective. It helps during development of tools for the system administrator, in choosing and specifying requirements for the monitoring functionality.

Here, a infrastructure monitoring checklist is presented that facilitates in selecting requirement when developling tools and techniques for monitoring of Ceph. We have classifed the infrastructure monitoring into four components, namely, (1) Background process functionality, (2) Storage infrastructure attributes, (3) Storage usage data, and (4) OS process utilization data. The monitoring checklist is defined for the four identified components of Ceph. The checklist is for both the administrator and the developer, and facilitates during requirement elicitation in identifying the monitoring functionality to be included in a tool. During requirement phase of tool development, functionality needed for monitoring of Ceph can be selected from the checklist.

For understanding requirements of monitoring Ceph and for formulating the checklist, a study of architecture of Ceph, processes in Ceph for storing and managing data, monitoring commands and configurable parameters of Ceph was performed. A study of associated plug-in of some standard open source monitoring software was also performed. This collectively defines understanding storage architecture and available monitoring provisions for Ceph. Using the use-case based approach; the requirements for monitoring have been identified from the system administrator perspective. The components of infrastructure monitoring, based on interaction of system administrator with Ceph infrastructure are defined. The functionality of each identified component of infrastructure monitoring has been identified.

The monitoring checklist allows selecting requirements from the checklist, in contrast to, specifying fresh requirements, when developing new tool for monitoring. From the checklist, all or part of functionality may be selected. The checklist is for use during requirement elicitation phase, and also for validation and verification of requirements during testing phase of the tool development. The requirement checklist presented here can be easily updated to include any new functionality or feature.

The Ceph infrastructure checklist presented here has been applied to three popular infrastructure monitoring tools of Ceph - Calamari 0, Nagios [18], and CollectD 0 to identify monitoring functionality provided by them. The work is being currently extended to provide generic infrastructure monitoring list for cloud object storage.

In this paper, section 2 gives an overview of Ceph object storage. Section 3 describes Ceph monitoring commands and configurable parameters. Section 4 discusses the components of infrastructure monitoring. Section 5 describes the monitoring checklist in detail. Section 6 illustrates few examples on which the checklist has been applied. Section 7 lists benefits of using the infrastructure monitoring list. Section 8 is a survey of related work. Section 9 states the conclusion.

2 Ceph Object Storage

The Ceph 0 object storage architecture comprises of three main components – Radosgw, Librados and RADOS.

Radosgw is a client interface for object storage that allows end-user to store and retrieve data. It supports Swift and S3 compatible APIs for facilitating end-user to perform various operations, such as, create, read, update and delete data as an object.

Librados is storage cluster protocol that provides native interface to interact with storage cluster and supports different languages, like, C, C++, Java, Ruby and Python. It allows client to interact with Ceph storage cluster, directly, using the defined API.

RADOS 0 is a reliable, autonomous and distributed object store. It consists of two sub-components – Monitor and OSD (Object Storage Device) Daemon. Monitor maintains current status of each component of cluster. Usually, one monitor is sufficient for this purpose, but to ensure high availability, a few monitors are used and a quorum for consensus about current state is established among them. OSD daemon is responsible for reading/writing data to/from storage cluster. OSD daemons communicate with each other to check whether other OSDs are in up and running state and also to replicate data.

3 Ceph Monitoring Commands and Configurable Parameters

In Ceph, several commands 0 exist that provide health of storage cluster and state of individual components, like, their running status and condition. There are also some commands that provide usage statistics of storage cluster.

The component of Ceph storage cluster has some configurable parameters that can be set according to the need of cluster. These parameters can be set at the time of software installation or can be changed dynamically at runtime. Ceph stores its configurable parameters in its configuration file. The configurable parameters 0 are divided into four major sections – global, osd, mon and client.radosgw. Configurable parameters set in ‘global’ section are applied to all instances of all components.

Parameters set in ‘osd’, ‘mon’ and ‘client.radosgw’ is applicable for instances belonging to OSD daemon, monitor and Radosgw, respectively. Configurable parameters define working of different processes running for the component.

4 Components of Infrastructure Monitoring

The classification of Ceph infrastructure into different components provides a framework for categorization of functionality of monitoring, from the administrator perspective. To understand requirements of infrastructure monitoring, a study of the components and processes executing on them has been performed. An in-depth study of different monitoring commands and configurable parameters of Ceph object storage has also been done. The infrastructure monitoring has been divided into four broad components as follows:

- *Background Process* - functionality of processes running in background
- *Storage Infrastructure* - attributes of storage infrastructure
- *Storage Usage* - utilization of storage infrastructure
- *OS Process Utilization* - utilization of OS processes

For Ceph, the authors define infrastructure monitoring as, “Monitoring physical infrastructure, logical infrastructure and associated processes”. The components of infrastructure monitoring are briefly described in the following subsections.

4.1 Background Process

During working of the Ceph object storage, several processes run in the background to perform the tasks defined in Ceph. From the different processes present in Ceph software, we identified the processes that are required to be monitored during runtime as shown in Fig. 1. These processes are required to be monitored to check health of system. The background processes that are required to be monitored are - Heartbeat, Authentication, Data scrubbing, Peering, Backfilling, Recovery and Synchronization.

Heartbeat ensures that OSDs responsible for maintaining copies of data are in up and running states. OSDs check heartbeat of other OSDs periodically and report the status to monitor.

Authentication is used to authenticate and authorize the client accessing the storage. Monitor is responsible for authentication process. Client can have different rights for access, like, read-only, write, access to admin commands, etc.

Data scrubbing checks data integrity. The process runs on OSDs and compares objects with their replica stored in another OSD. There are two types of scrubbing– light scrubbing and deep scrubbing. In light scrubbing, metadata of objects is compared to catch bugs. In deep scrubbing, data in objects is compared bit-by-bit. Usually, light and deep scrubbing are performed daily and weekly respectively.

Peering is required for creating an agreement about state of all objects among OSDs that are responsible to keep copy of objects before replication.

Synchronization ensures availability of data in a federated system implemented with multiple regions and multiple zones. A cluster must have a master region and a region must have a master zone. Synchronization process runs on Radosgw. There are two types of synchronization - data synchronization and metadata synchronization. In data synchronization, data of master zone in a region is replicated to a secondary zone of that region. In metadata synchronization, metadata of users and buckets is replicated from master zone in master region to master zone in a secondary region.

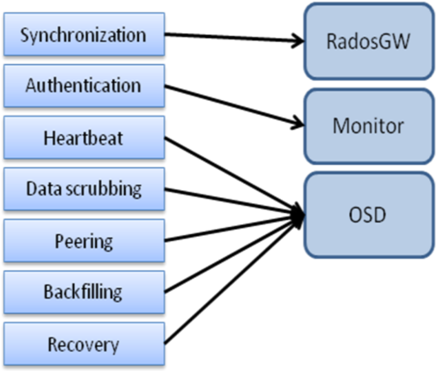

Fig. 1. Background processes for Infrastructure monitoring in Ceph

Backfilling runs when OSD is added or removed to/from Ceph storage cluster. In order to rebalance cluster, objects are moved to or from OSDs. This migration takes place as ‘backfilling’ at lower priority to maintain operational performance of system.

Recovery runs when OSD crashes and comes back online. In such condition, several objects stored in OSD get outdated and goes in recovery mode, when it restarts. To maintain operational performance of system, recovery process takes place with some limitations.

Several other processes like logging and journaling do not require monitoring during runtime.

4.2 Storage Infrastructure

The storage infrastructure of Ceph is logically divided into clusters which contain few monitors and a large number of OSDs. In a typical scenario, an OSD maps to a storage drive or a RAID group. The storage cluster is divided into pools, which are further divided into Placement Group (PG). Each PG maps to some OSDs.

A pool facilitates segregation of data, logically, based on user's requirement. For providing availability, pool is specified as replicated or erasure-coded. Replicated pool maintains multiple copies of data. In erasure-coded pool, data is divided into number of chunks associated with some code chunks. The data is stored in PGs within a pool. For fault tolerance, each copy of PG is stored in separate OSD. A set of OSDs that are responsible for keeping copy of a PG is called Acting set of that PG and a set of OSDs that are ready to handle incoming client request is Up set of that PG. Generally, Acting set and Up set of a PG are identical. If they are not found identical, it implies that Ceph is migrating data or an OSD is recovering or there is any problem. One OSD in Acting set is primary OSD. Client communicates with primary OSD to read/write data. Primary OSD interacts with other OSDs to replicate data.

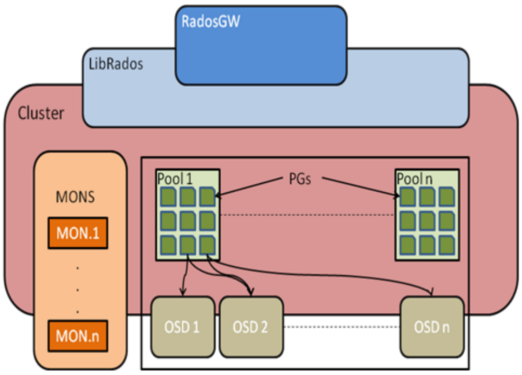

Fig. 2. Ceph Object Storage Structure

Fig. 2 shows object storage infrastructure for Ceph. Monitoring is required at all the different levels of physical and logical infrastructure, for observing existing resources. It helps when there is a need to add or remove resources and to detect failing or failed drives for replacement.

4.3 Storage Usage

The amount of the logical and physical infrastructure that is being consumed is required to be monitored to identify full or near full storage. The logical infrastructure is monitored at different levels - cluster, pool and PG level. This helps in scaling up and scaling down the system resources.

4.4 OS Processes Utilization

The Ceph object storage uses CPU, memory, and network for its own working. Different processes, like, heartbeat, peering etc. run on different components of storage cluster and utilize CPU, memory and network. OS processes utilization is required to be monitored to improve efficiency and performance of system.

5 Infrastructure Monitoring List

For arriving at the monitoring checklist, the Ceph infrastructure was classified under four components - Background process functionality, Storage infrastructure attributes, Storage usage data and OS Process utilization data.

Background process checklist consists of functionality running in background of Ceph. During requirement elicitation, this list helps in deciding process parameters that need to be monitored for Ceph software. The background processes are monitored for three entities – monitor, OSD, and Radosgw.

Heartbeat and peering processes are required to be monitored for finding OSDs in ‘Up’ and ‘Acting’ set of a PG, respectively, to check if number of OSDs in Acting set are same as that defined in pool size and OSDs in Up set of a PG are equivalent to OSDs in Acting set of that PG. The users are monitored for authentication to check access permissions according to defined capabilities. Data scrubbing needs to be monitored for type of scrubbing, its frequency, and number of pending scrubs and errors to identify rate of corrupted files found in system so that any abnormality can be identified. Data and metadata synchronization have parameters, such as, errors during sync, wait time, count of shards that are checked or failed, and error listing metadata to determine correct working of system. Backfilling and recovering processes are monitored for their respective status so that impact on system performance can be decreased. Table 1 lists requirement checklist for background process functionality.

Storage Infrastructure checklist defines parameters for logical and physical infrastrcuture that needs to be monitored. During requirement elicitation this list helps in deciding parameters of infrastructure that need to be monitored for Ceph. The storage infrastructure is divided into five levels – cluster, monitor, OSD, pool and PG.

Table 1. Background Process Functionality

Process	Parameters	Monitor	OSD	Radosgw
Heartbeat	OSDs in Up set of a PG	-	✓	-
Authentication	Users with different capabilities	✓	-	-
Data scrubbing	Type – Light/Deep, frequency, scrub pending, no. of errors	-	✓	-
Peering	OSDs in Acting set of PG	-	✓	-
Data synchronization	Sync error, incremental sync error, retry wait time/until next sync, object sync timeout, no. of shards to check/failed, no. of items synced successfully/ processed	-	-	✓
Metadata synchronization	Time to wait for bucket log consistency, Error listing metadata	-	-	✓
Backfilling	Count, frequency, time to wait for retrying	-	✓	-
Recovering	No. of active recovery request/ recovered chunks, time to delay	-	✓	-

At cluster level, parameters are identified as cluster health status, number and list of monitors, OSDs, pools and PGs in cluster. Detail of each OSD, current epoch, and OSD status can be monitored at OSD level. Pool level parameters are number of PGs in pool, pool is replicated or erasure coded etc. At PG level, parameters define state of PG. Table 2 lists requirement checklist for storage infrastructure attributes.

Storage usage checklist consists of parameters that provide data about the usage of storage infrastructure. During requirement elicitation the list helps in deciding parameters for usage of storage infrastructure that need to be monitored. Storage usage data for monitoring is defined at three levels - cluster, pool and PG.

IOPS (Input Output Per Second) measure input/output load to avoid I/O bottleneck in system. Latency provides time taken in data transfer so that in cases of interruption the cause can be found.

Table 2. Storage Infrastructure Attributes

Level	Parameters
Cluster	Cluster ID Cluster health status Number and list of monitors Number and list of OSDs Number and list of Pools Number and list of PGs
Monitor	Detail – position, name, address, port of monitor Current epoch – when map created, last modified Status - Running/ not running Status of monitor quorum
OSD	Details – id, weight, type, name Current epoch – when map created, last modified Status - In/out, up/down
Pool	Details - Name, Pool ID Number and list of PGs Replicated/erasure coded Cache tiering status
Placement Group	Detail – PG ID, PG version, timestamp PG state (Creating, Peering, Active, Clean, Degraded, Recovering, Backfilling, Remapped, Stale, Unclean, Inactive)

Total storage capacity and free space availble are inspected so that alerts can be raised before system reaches near-full capacity. Amount of data stored and number of objects stored provide estimate of storage capacity. IOPS and latency are monitored at cluster and pool level. Notional value monitored at pool level determines utilized space excluding space used by its replicas. Table 3 lists storage infrastructure usage parameters at different levels.

Table 3. Storage Usage Data

Level/Parameters	Cluster	Pool	PG
IOPS – read, write	✓	✓	-
Latency – max., avg., min.	✓	✓	-
Overall storage capacity	✓	-	✓
Amount of data stored	Total & Notional	Notional	Total
Number of objects stored	Total	Notional	-
Amount of free space available/ used	Total	Notional	Total

OS processes checklist contains parameters to determine utilization of operating system processes. During requirement elicitation this list helps in deciding parameters for utilization of OS processes that need to be monitored.

CPU, memory and network utilization data is monitored to determine consumption of OS resources during execution. It helps to identify processes and components that are under utilizing or highly utilizing OS resources so that extra resources can be provisioned based on demand. Table 4 lists parameters for OS process utilization.

Table 4. OS Process Utilization data

Parameters	Reason
CPU Utilization	Find CPU consumption by processes and system to identify processes and components which have high CPU load and which are under utilized
Memory utilization	Track available memory to determine processes and components that are consuming more memory so that memory can be upgraded
Network utilization	Track network traffic and identify network interfaces that have excessive use

6 Case Study

The infrastructure monitoring functionality lists have been applied for case study to three monitoring software - Calamari 0, Nagios 0, and CollectD 0.

Calamari is management and monitoring service specifically for Ceph. It exposes high level REST APIs and a user interface built on these APIs for monitoring Ceph infrastructure. *Nagios* is open source infrastructure monitoring software that enables organizations to identify and resolve IT infrastructure problems before they have drastic effect on system. Nagios provides some built in plug-ins for monitoring health of cluster and individual components of Ceph object storage, like, check_ceph_health and check_ceph_mon. *CollectD* is daemon that collects system information and helps system administrators to maintain an overview of resources to avoid bottlenecks.

Table 5 displays comparative checklist of the three monitoring software for storage infrastructure, usage and OS process utilization of infrastructure monitoring. In the table, '√' denotes parameter is supported by tool; 'x' not supported. 1st, 2nd and 3rd column for each is for storage infrastructure, usage and OS process, respectively.

Fig. 3 displays percentage of parameters of each monitoring checklist functionality supported by tools in our case study. Some of our key observations are as follows-

- *Calamari* monitors 10% of background processes; 95.65 % of storage infrastructure; 87.5% storage usage; and 33.33% OS processes.
- *Nagios* monitors mainly the status of storage infrastructure (69.56%). It does not monitor background processes, storage usage and OS processes.
- *CollectD* monitors 100% storage usage; 100% OS processes utilization; 38.46% of storage infrastructure. It does not monitor background processes.

Some interesting observations emerging from the case study are as follows-

- Background processes is only monitored 10% by Calamari
- Storage infrastructure is monitored by all three - Calamari, Nagios, CollectD
- Storage usage is monitored by Calamari and CollectD
- OS processes is monitored by Calamari and CollectD

It can be seen that tools offer different coverage for components being monitored and there is no consistency for same. Also background processes are hardly monitored because Ceph does not have commands to provide status of running processes.

Table 5. Storage infrastructure, Storage usage, and OS processes case study

Level	Parameters–Storage Infrastructure	Parameters–Storage Usage	Parameters–OS Processes	Calamari			Nagios			Collect D	
Cluster	ID	IOPS – read, write	CPU Util.	√	√	√	√	x	x	x	√
	Health status	Latency-max/avg/min	Memory Util.	√	x	x	√	x	x	x	√
	No. & list of monitors	Notional data stored	Network Util.	√	√	x	√	x	x	√	√
	No. & list of OSD	No. of objects stored	-	√	√	-	√	x	-	√	√
	No. & list of Pools	Total data stored	-	√	√	-	√	x	-	√	√
	No. &List of PGs	Free space available	-	√	√	-	√	x	-	√	√
	-	Used raw storage	-	-	√	-	-	x	-	-	√
	-	% of raw storage used	-	-	√	-	-	x	-	-	√
	-	Overall storage capacity	-	-	√	-	-	√	-	-	√
Monitor	Detail	-	-	√	-	-	√	-	-	x	-
	Current epoch	-	-	√	-	-	√	-	-	√	-
	Status	-	-	√	-	-	√	-	-	√	-
	Monitor quorum status	-	-	√	-	-	√	-	-	√	-
OSD	Detail	-	-	√	-	-	√	-	-	x	-
	Current epoch	-	-	√	-	-	√	-	-	x	-
	Status- in/out, up/down	-	-	√	-	-	√	-	-	√	-
Pool	Name, ID	IOPS – read, write	CPU Util.	√	√	√	x	x	x	x	√
	No. & list of PGs	Latency – max/ avg/min	Memory Util.	√	x	x	√	x	x	√	√
	Replicate/Erasure	Notional data stored	Network Util.	√	√	x	x	x	x	x	√
	Cache tiering status	Notional objects stored	-	√	√	-	x	x	-	x	√
PG	Detail	Amount of data used	-	√	√	-	x	x	-	x	√
	-	Free storage capacity	-	-	√	-	-	x	-	-	√
	PG state	Total storage capacity	-	√	√	-	√	x	-	√	√

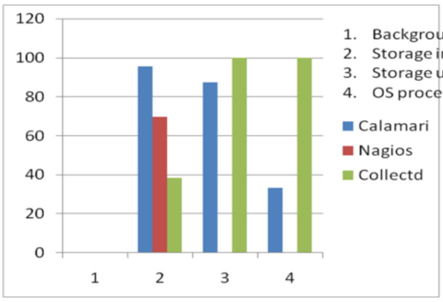

Fig. 3. Graph showing percentage of checklist used by monitoring software

7 Benefits of Checklist

The requirement checklist presented here has been derived after performing a detailed study of Ceph storage architecture, processes running in it and exhaustive study of basic monitoring commands and configurable parameters over Ceph.

Usually, infrastructure monitoring software is developed for a specific purpose without planning or preparation of list of possible functionality that can be included. The checklist helps system administrator to choose functionality required to monitor infrastructure of Ceph with minimum effort. The developers use checklist to check functionality required for infrastructure monitoring during development of tool. The checklist allows developer and administrator to include more functionality in monitoring software rather than just basic functionality.

8 Related Work

Cloud monitoring has gathered focus in research. Several researchers discuss about research motivation, approaches used for monitoring cloud and different methodologies applied to monitor a cloud for different purposes. Alhamazani et al. 0 discusses methodology to monitor cloud for facilitating automated QoS management; Adinarayan 0 discuss challenges in monitoring private cloud and describe capabilities of IBM SmartCloud monitoring to tackle these challenges.

Several frameworks are proposed by researchers for different purposes on monitoring the cloud infrastructure 000. Gogouvitis et al. 0 propose an architectural design and implementation of monitoring solution in context of VISION cloud project. Mdhaffar et al. 0 propose dynamic Complex Event Processing architecture for cloud monitoring and analysis; Uriate and Westphall 0 propose monitoring architecture 'Panoptes' for autonomic clouds; Chaves, et al. 0 discuss design and implementation of private cloud monitoring system (PCMONS).

The frameworks and architectures highlight ways of collecting data from system required for monitoring and how to monitor. However, there is no mention of parameters required to be monitored in cloud. Usually, freely available monitoring software, like, Zenoss, Nagios are adapted for incorporating monitoring functionality for Ceph object storage. Our extensive search for work carried out for finding list of parameters required for monitoring Ceph object storage yielded no result.

9 Conclusion and Future Work

In this paper, we have presented infrastructure monitoring list for Ceph object storage. The list eases the task of administrator and developers by providing them a list from where the functionality can be selected. Designers and developers of new monitoring software for Ceph can also use the list as a reference for identifying possible functionality that can be incorporated in monitoring software. The list is extendible and can be updated to add new functionality and features.

Since our functionality checklist is specific for Ceph object storage, other cloud object storage may have some more functionality which does not lie in scope of this paper. In future, the authors aim to develop a generic functionality checklist for cloud object storage system. We also propose to prioritize the proposed list.

References

1. Adinarayan G.: Monitoring and Capacity Planning of Private Clouds: The Challenges and the Solutions. IEEE Int. Conf. on Cloud Computing in Emerging Markets (CCEM), India (2012) 1-3
2. Alhamazani K., et al.: Cloud monitoring for optimizing the QoS of hosted applications. IEEE 4th Int. Conf. on Cloud Computing Technology and Science (CloudCom), Taipei (2012) 765-770
3. Barbosa de Carvahlo M., et al.: A cloud monitoring framework for self-configured monitoring slices based on multiple tools. 9th Int. Conf. on Network and Service Management (CNSM), Zurich (2013) 180-184
4. Chaves S., et al.: Towards an architecture for monitoring private clouds. IEEE Communications Magazine Vol. 49 (2011) 130-137
5. Gogouvitis S., et al.: A Monitoring Mechanism for Storage Clouds. 2nd Int. Conf. on Cloud and Green Computing, Xiangtan (2012) 153-159
6. Grobauer B., Walloschek T., Stocker E.: Understanding cloud-computing vulnerabilities. IEEE Security and Privacy Vol. 9 (2010) 50-57
7. Mdhaffar A., et al.: A Dynamic Complex Event Processing Architecture for Cloud Monitoring and Analysis. IEEE 5th Int. Conf. on Cloud Computing Technology and Science (CloudCom) (Vol. 2) Bristol (2013) 270-275
8. Moses J., Iyer R., Illikkal R., Srinivasan S., Aisopos K.: Shared Resource Monitoring and Throughput Optimization in Cloud-Computing Datacenters. IEEE Int. Parallel & Distributed Processing Symposium (IPDPS), Anchorage AK (2011) 1024-1033
9. Rehman Z., et al.: A Framework for User Feedback based Cloud Service Monitoring. 6th Int. Conf. on Complex, Intelligent and Software Intensive Systems (CISIS), Palermo (2012) 257-262
10. Shao J., Wei H., Wang Q., Mei H.: A Runtime Model Based Monitoring Approach for Cloud. IEEE 3rd Int. Conf. on Cloud Computing (CLOUD), Miami, FL (2010) 313-320
11. Uriarte R., Westphall C.: Panoptes A monitoring architecture and framework for supporting autonomic Clouds. IEEE Network Operations and Management Symposium (NOMS), Krakow (2014) 5-9
12. Weil S., et al.: RADOS A Scalable, Reliable Storage Service for Petabyte-scale Storage Clusters. 2nd Int. workshop on Petascale data storage, ACM, New York (2007) 35-44
13. Yongdnog H., et al.: A Scalable And Integrated Cloud Monitoring Framework Based On Distributed Storage. 10th Web Information System and Application Conference, Yangzhou (2013) 318-323
14. https://github.com/ceph/calamari
15. http://ceph.com/ceph-storage/object-storage/
16. https://collectd.org/
17. http://ceph.com/docs/v0.78/rados/operations/monitoring/
18. http://www.nagios.org/