An Incremental Snapshot System using Smart Backup for Persistent Disks in Cloud

Veena S (✉ djohnveena@gmail.com )
S.A.ENGINEERING COLLEGE  https://orcid.org/0000-0003-0126-9119

D.John Aravindhar
Hindustan Institute of Technology and Science: Hindustan University

L. Sudha
S.A.Engineering College

K.B. Aruna
S.A.Engineering College

Research Article

Keywords: Cloud computing, Persistent Storage, Smart Backup, Snapshot, Virtual Machine

DOI: https://doi.org/10.21203/rs.3.rs-397234/v1

License: © This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License
Abstract

Periodical snapshots also called persistent disk snapshots are an essential feature associated with every cloud-hosted virtual instance, which minimizes the risk of unexpected data loss in the server and unavailability issues. The conventional method of creating a snapshot of a production-level server is done by temporarily disabling write access to data during the backup, either by stopping the accessing applications or by using the locking API provided by the operating system to enforce exclusive read access. This is not tolerable for high-availability always-online systems, in which service stoppages are not bearable. In order to solve this downtime issue in high-availability systems, the backup can be performed in a smarter way as incremental snapshots in which a read-only copy of the dataset frozen at a point in time is stored as snapshot by allowing applications to continue processing and writing their data to the instance. Also, incremental snapshots work in a way that only blocks which are different from the former snapshots are processed and stored in the subsequent one. This smart backup will reduce the overall space requirement of the snapshot system by storing only the differences in file storage blocks. Also, when implemented this will save energy and infrastructure requirements of the cloud provider as well as the cost and time of the end-user to create a low-latency server.

Introduction

A Virtual Machine is a computer file, typically called an image, which behaves like an actual computer. In other words, creating a computer within a computer. It runs in a window, much like any other program, giving the end-user the same experience on a Virtual Machine as they would have on the host operating system itself. The Virtual Machine is sandboxed from the rest of the system, meaning that the software inside a Virtual Machine cannot escape or tamper with the computer itself. This produces an ideal environment for testing other operating systems including beta releases, accessing virus-infected data, creating operating system backups and running software or applications on operating systems for which they were not originally intended.

Multiple Virtual Machines can run simultaneously on the same physical computer. For servers, the multiple operating systems run side-by-side with a piece of software called a hypervisor to manage them, while desktop computers typically employ one operating system to run the other operating systems within its program windows. Each Virtual Machine provides its own virtual hardware, including CPUs, memory, hard drives, network interfaces, and other devices. The virtual hardware is then mapped to the real hardware on the physical machine which saves costs by reducing the need for physical hardware systems along with the associated maintenance costs that go with it, plus reduces power and cooling demand.

System Virtual Machines (also termed full virtualization VMs) provide a substitute for a real machine. They provide the functionality needed to execute entire operating systems. A hypervisor uses native execution to share and manage hardware, allowing for multiple environments that are isolated from one another, yet exist on the same physical machine. Modern hypervisors use hardware-assisted
virtualization, virtualization-specific hardware, primarily from the host CPUs. The desire to run multiple operating systems was the initial motive for Virtual Machines, so as to allow time-sharing among several single-tasking operating systems. In some respects, a system Virtual Machine can be considered a generalization of the concept of virtual memory that historically preceded it. IBM’s CP/CMS, the first systems to allow full virtualization, implemented time-sharing by providing each user with a single-user operating system, the Conversational Monitor System (CMS). Unlike virtual memory, a system Virtual Machine entitled the user to write privileged instructions in their code. This approach had certain advantages, such as adding input/output devices not allowed by the standard system.

Process Virtual Machines are designed to execute computer programs in a platform-independent environment. A process VM sometimes called an application Virtual Machine, or Managed Runtime Environment (MRE), runs as a normal application inside a host OS and supports a single process. It is created when that process is started and destroyed when it exits. Its purpose is to provide a platform-independent programming environment that abstracts away details of the underlying hardware or operating system and allows a program to execute in the same way on any platform.

PERSISTENT DISK SNAPSHOTS

A storage snapshot is a set of reference markers for data at a particular point in time. A snapshot acts like a detailed table of contents, providing the user with accessible copies of data that they can roll back to. Each snapshot uses a differencing disk - a virtual hard disk (VHD) - that stores changes made to another virtual disk or the guest operating system. This VHD intercepts all future write operations and leaves the original data in an unaltered state. Snapshots have parent-child relationships and form a tree. Each snapshot taken creates another branch of the tree. Snapshots are generally created for data protection, but they can also be used for testing application software and data mining. A storage snapshot can be used for disaster recovery when information is lost due to human error or data corruption.

Copy-on-write snapshots store metadata about the location of the original data without copying it when the snapshot is created. These snapshots are created almost instantly, with little performance impact on the system taking the snapshot. This enables rapid recovery of data in case of a disk writes error, corrupted file or program malfunction. Data in a copy-on-write snapshot is consistent with the exact time the snapshot was taken, hence the name copy-on-write. However, all previous snapshots must be available if complete archiving or recovery of all the data on a network or storage medium is required. Every copy-on-write process requires one read and two writes; data needs to be read and written to a different location before it is overwritten.

Clone or split-mirror snapshots reference all the data on a set of mirrored drives. Each time the utility is run, a snapshot is created of the entire volume, not only of the new or updated data. This makes it possible to access data offline and simplifies the process of recovering, duplicating or archiving all the data on a drive. This is a slower process, and each storage snapshot requires as much storage space as the original data.
Copy-on-write with background copy takes snapshot data from a copy-on-write operation and uses a background process to copy the data to the snapshot storage location. This process creates a mirror of the original data and is considered a hybrid between copy-on-write and cloning.

Redirect-on-write storage snapshots are similar to copy-on-write, but writes are redirected to storage that is provisioned for snapshots, eliminating the need for two writes. Redirect-on-write snapshots write only changed data instead of a copy of the original data. When a snapshot is deleted, that data needs to be copied and made consistent on the original volume. The creation of more storage snapshots complicates original data access along with the snapshot data.

Incremental snapshots create timestamps that allow a user to go back to any point in time. Incremental snapshots can be generated faster and more frequently than other types of storage snapshots. And because they do not use much more storage space than the original data, they can be kept longer. Each time an incremental snapshot is generated, the original snapshot is updated.

VMware snapshots copy a Virtual Machine disk file and can restore a Virtual Machine (VM) to a specific point in time if a failure occurs. VMware snapshot technology is used in VMware virtual environments and is often deleted within an hour. VMware administrators take multiple snapshots of a VM, creating multiple, point-in-time restore points. When a VMware snapshot is taken, any writeable data becomes read-only.

Continuous Data Protection (CDP) uses snapshots to back up a system in a way that allows users to recover the most up-to-date instance of data. While storage snapshots are typically scheduled at predetermined points, CDP can back up data each time a change is made. This allows a user to recover data with the most recent changes included, whereas those updates may be lost if a regular storage snapshot was not taken before the system failed. CDP also keeps a record of every change that occurs, so it is always possible to recover the most recent clean copy of the data.

Snapshot vs Backup

There are several benefits to using storage snapshots as part of a larger backup strategy. Snapshots provide quicker and easier access to data and can be leveraged by backup applications to enable features like instant recovery. But while storage snapshot technology is a helpful supplement to a backup plan, it is not considered a full replacement for a traditional backup. Relying on stored snapshots for backups can take up storage space and seriously impact performance, and a storage snapshot is an instance, not a full copy of the data. Snapshots are dependent on source data, so if that data is lost, the snapshot is gone as well. Because of these vulnerabilities, it is not recommended to use snapshots in lieu of a full backup.

SERVER MONITORING

Server monitoring is the process of reviewing and analyzing a server for availability, operations, performance, security and other operations-related processes. It is performed by server administrators to
ensure that the server is performing as expected and to mitigate problems as they become apparent. Server monitoring's primary objective is the protection of a server from possible failure. Server monitoring can be performed using manual techniques and automated server monitoring software.

Agent-based monitoring consists of a software component, typically a small application, which resides on the client-server and collects data. The data is then returned to the monitoring station based on a policy within the local agent, or as requested by the monitoring station. In best practice situations, the agent responds with information based on requests originating from its monitoring station. This practice makes the agent very lightweight but able to access granular metrics for better monitoring, alerting and reporting, as well as deeper levels of root-cause analysis and troubleshooting.

By implementing an agent-based solution, advanced capabilities can be encapsulated within the agent functionality. The ability to directly interact with the client platform and its services allows the monitoring station to remotely execute automated actions for a more proactive IT delivery. Automated actions can include simple IT service recovery and maintenance tasks or more advanced actions like the spinning up and down of virtual capacity to account for fluctuating demand on IT systems. For example, a service monitor may be watching the log directory on an active Web server. When the directory exceeds a set capacity threshold, the agent can automatically compress and archive the log files, and begin a new set of logs, keeping the volume from filling and potentially crashing the Web server.

Agentless monitoring is deployed in one of two ways: using a remote API exposed by the platform or service being monitored or directly analyzing network packets flowing between service components. Network packet analysis is typically implemented in addition to either an agent-based or agentless monitoring solution. Network analysis will not provide detailed metrics on the servers supporting the application services communicating over the network but will provide data on service performance and availability. End-user experience monitoring typically includes network traffic analysis.

In SNMP Monitoring, a significantly reduced set of data is made available when compared to an agent based or WMI monitoring approach. With SNMP, one is limited to what is exposed by the vendor, which cannot be easily extended in most cases. In agent-based monitoring, one would be able to extend the metric collection to include all the deep metrics, and not just SNMP exposed ones. Gartner strongly recommends an agent-based solution for monitoring mission-critical applications and servers due to the level of metrics required to effectively monitor and manage critical services, and the potential to use agentless monitoring for non-essential servers and applications. As application and service vendors integrate management APIs into their products, this metrics gap is shrinking between the agent and agentless monitoring, but this typically takes several years for the APIs to mature and several more for systems management vendors to fully support the APIs within their products.

Windows Management Instrumentation (WMI) is a good example of how some vendors are exposing their deeper server and platform metrics for agentless monitoring consumption. For many Windows-based servers and applications, agentless monitoring via the WMI gateway provides strong monitoring capabilities. However, there are some cases where an agent-based monitoring solution would be
preferred. For example, a heterogeneous IT environment that includes Windows servers and additional platforms (UNIX, Linux, VMware, etc.) would be best suited for a solution that combines both agent-based and agentless monitoring together, in one dashboard.

Agentless SNMP solutions do not provide the same level of expansion and integration that is possible with an agent-based solution; Furthermore, agentless solutions typically do not provide the facilities to interact with the service platform being monitored with the same level of functionality as an agent-based solution. By not having an agent that can act as an arbitrator for commands being executed on the client by the monitoring station, it becomes very difficult to develop proactive and automated actions like service management and recovery scripts. Extending the monitoring capabilities of an agentless solution to include custom application and service monitors is either a very difficult development effort, or simply not possible.

**Literature Review**

Apoorv Saxena [1] have found the right balance between backing up frequently (improving data safety) and reducing resource usage (power consumption and communication cost) in the cloud backup. They have modeled a wide set of exhaustive data backup processes as a general batch service queueing model with multiple vacations and probabilistic restarts. This analysis aided them in computing Quality of Service (QoS) measures of the data backup process such as the fraction of time the backup server is busy, the frequency of new connections and the age of the data at the beginning of a backup period which enabled them to quickly examine the dependence of QoS on the model parameters as well as to compute the optimal parameters in the backup process.

Yasser Aldwyan [2] have presented an approach for achieving availability and performance when deploying web applications in distributed Clouds. A genetic algorithm for data center (DC) selection that factors in proximity to users and inter-DC latencies is presented. They have also focused on the placement issue and improves end-to-end response times even in the presence of failures. It was shown clearly, how latency-aware application deployment can offer higher performance and stability before and after failures. Results were presented based on realistic Cloud-based experiments across the national research Cloud in Australia.

T.H. Nguyen [3] et al. have considered the possibility of turning servers on and off to keep a balance between capacity and energy saving. While turning off servers could save power, it could also delay the response time of requests and therefore reduced the performance. Furthermore, as consistency is one of the most important factors for a system, so they have also analyzed the level of consistency in the form of switching rate and fault occurrence. Several heuristic-based switching policies were introduced by them with a view to balance the cost of power-saving, performance, and consistency. They have presented simulation results and discussed them with requests arriving according to a two-phase Poisson process.
X. Guan [4] et al. have formulated energy-efficient virtual network embedding that incorporates energy costs of operation and migration for nodes and links. From experiments, they have proved that the NP-hardness of the problem and develop a heuristic algorithm to minimize energy consumption. They have considered a practical intra-DC architecture to further improve energy efficiency. Also, they have conducted extensive evaluations and comparisons with existing algorithms and shown that their proposed algorithm substantially saves energy consumption and allows high acceptance ratios.

V. Chang [5] et al. have introduced a new modeling technique in cloud backup called Organizational sustainability modeling (OSM) compares Cloud and non-Cloud storage. They have identified various factors that affect performance and design ways to make fair comparisons. They have also explained how to use OSM including its definitions, input, and output. Experiments were conducted and they have presented two case studies of Big Data storage with 40 runs to support. Results are analyzed and presented with data analysis and visualization and it has been concluded that the improvement in efficiency was higher on the Cloud than the non-Cloud.

P.M. Van de Ven [6] et al. have addressed the tradeoff between frequent backups (increased safety) and reducing the network peak load. They have addressed the problem of shifting backup traffic from peak hours to off-peak hours within the constraints imposed by user connectivity. Backups are scheduled using a distributed protocol characterized by a set of probabilities that indicate the likelihood of a user initiating a backup during a given hour. Given these probabilities, the authors have studied the network capacity by investigating the rate at which users can generate data while retaining stable backlog processes. They have derived explicit expressions for the stationary behavior of the backup process and discussed how to choose the backup probabilities that strike the right balance between a low peak load and data safety. Via simulation experiments, the authors have shown and proved that this approach is highly successful in reducing costs.

R. Xia [7] et al. have investigated the use of Markov Decision Process (MDP) to guide the scheduling data backup operation. The authors have proposed a new framework that can automatically generate an MDP instance given system specifications and data requirements. They have also demonstrated the benefits of the MDP approach. It has been concluded by the authors that their framework allowed the translation of several data and system-related requirements into an MDP instance so that the solution to the instance has provided the optimal schedule with reduced downtime.

D. Claeys [8] et al. have analyzed the threshold-based exhaustive data backup scheduling mechanism by means of a queueing-theoretic approach. Data packets that have not yet been backed up are modeled by customers waiting for service (back-up). The authors have obtained the probability of generating a function of the system content (backlog size) at random slot boundaries in steady-state. The authors have claimed that they have developed first of a kind queueing system has been developed and analyzed to model a threshold-based exhaustive backup policy.
D. Boullery [9] et al. have invented a method for scheduling a backup of digital data includes determining whether a backup has previously been performed within a predetermined period. It was then determined whether a connection to a backup server is available. It was then decided whether to initiate a backup of digital data within a present time slot based at least in part on a randomly generated value when it was determined that a backup has not previously been performed within the predetermined period. The inventors have determined methods to prove that a connection to the backup server is available. Then, the digital data is backed up to the backup server when it is decided that the data backup is to be initiated.

Mingzhong Wang [10] et al. have proposed a method that computes and compares the potential loss with and without data backup to achieve the trade-off between the overhead of intermediate dataset backup and task re-execution after exceptions. The authors have also designed a utility function with the model and applied a genetic algorithm to find the optimized schedule. The results have shown that the robustness of the schedule is increased while the possible risk of failure is minimized, especially when the volume of generated data is not large in comparison with the input.

Marco Gramaglia [11] et al. have presented a micro-payment-based incentive mechanism for long-term peer-to-peer storage systems. The main novelty of the proposed incentive mechanism by the authors is to allow users to be off-line for extended periods of time without updating or renewing their information by themselves. This feature is enabled through a digital cheque, issued by the user, which is later employed by the peers to get gratification for storing the user's information when the user is offline. They have also included a secure and lightweight data verification mechanism, along with improvements in the availability of the stored information and the scalability of the whole system. This paper details the verification and cheque-based incentive mechanisms in the context of P2P backup service and analyzes its scalability and security properties. The authors have also validated the system by means of simulation, proving the effectiveness of the proposed incentive.

**Existing System**

The major drawback of conventional backup systems is they use conventional full backup system, which drastically reduce the runtime performance of the server during backup and snapshot creation. There are also chances of data corruption during backup process, if any read write operations happen. These problems can be possibly rectified by using incremental snapshot system in which the backup process is efficient, quick and cost and power saving.

**INCREMENTAL SNAPSHOT SYSTEM**

The main objective of proposed system is to study, implement and analyze storage snapshot mechanisms for production servers in a way that the snapshot process will not or minimally affect the performance of the server keeping all security policies and aspects associated with the production server.
In order to achieve the goal, a live production server has been selected for experiment conduction. Monitoring agents were installed to monitor the various performance and efficiency aspects of the server.

### 4.1 VM INSTANCE

A scalable, high-performance Virtual Machine has been chosen to test and implement the snapshot backup process associated with the system. A new Virtual Machine also called “VM instance” has been created from Google Cloud Platform (GCP), offered by Google, which is a suite of cloud computing services. Google Compute Engine (GCE) is the Infrastructure as a Service (IaaS) component of Google Cloud Platform which is built on the global infrastructure that runs Google’s search engine, Gmail, YouTube, and other services. Google Compute Engine enables users to launch Virtual Machines (VMs) on demand. VMs can be launched from the standard images or custom images created by users. GCE users must authenticate based on OAuth 2.0 before launching the VMs. Google Compute Engine can be accessed via the Developer Console, RESTful API or command-line interface (CLI).

Google Compute Engine Unit (GCEU), which is pronounced as GQ, is an abstraction of computing resources. According to Google, 2.75 GCEUs represent the minimum power of one logical core (a hardware hyper-thread) based on the Sandy Bridge platform. The GCEU was created by Anthony F. Voellm out of a need to compare the performance of Virtual Machines offered by Google. It is approximated by the Coremark (TM) benchmark run as part of the PerfKitBenchmarker Open Source benchmark created by Google in partnership with many Cloud Providers.

### 4.2 PERSISTENT DISKS

Every Google Compute Engine instance starts with a disk resource called persistent disk. The persistent disk provides the disk space for instances and contains the root filesystem from which the instance boots. Persistent disks can be used as raw block devices. By default, Google Compute Engine uses SCSI for attaching persistent disks. Persistent Disks provide straightforward, consistent and reliable storage at a consistent and reliable price, removing the need for a separate local ephemeral disk. Persistent disks need to be created before launching an instance. Once attached to an instance, they can be formatted with the native filesystem.

A single persistent disk can be attached to multiple instances in read-only mode. Each persistent disk can be up to 10TB in size. Google Compute Engine encrypts the persistent disks with AES-128-CB, and this encryption is applied before the data leaves the Virtual Machine monitor and hits the disk. Encryption is always enabled and is transparent to Google Compute Engine users. The integrity of persistent disks is maintained via an HMAC scheme.

### 4.3 HARDWARE ARCHITECTURE

The architecture of the created VM instance is listed in Table 1.
### Table 1
Instance Architecture

| Key                     | Value                                                                 |
|-------------------------|----------------------------------------------------------------------|
| Instance Id             | 8957763015131814851                                                  |
| Machine type            | n1-standard-1                                                         |
| CPU                     | 1vCPU                                                                |
| Memory                  | 3.75 GB                                                              |
| CPU platform            | Intel Skylake                                                        |
| Zone                    | asia-south1-a (Mumbai)                                               |
| Internal IP             | 10.160.0.28                                                          |
| External IP (Public IP) | 35.200.144.133                                                       |
| Boot disk               | Ubuntu 18.04 LTS                                                     |
| Image Name              | AMD 64-bit Bionic Beaver                                             |
| Disk Type               | SSD Persistent Disk                                                 |
| Disk Volume Size        | 32 GB                                                                |
| Sustained Random IOPS limit | Read 24.00  
Write 48.00                          |
| Sustained throughput limit | Read 3.84  
Write 3.84                          |
| Physical block size     | 4 KB                                                                 |
| Encryption type         | Google managed                                                       |
| Allowed Ingress Firewall Ports | Port 22 (SSH)  
Port 80 (HTTP Traffic) Port 443 (HTTPS Traffic)  
Port 2587 (Amazon SMTP Relay)                          |
| Allowed Egress Firewall Ports | All (0.0.0.0/0)                                      |

### 4.4 SOFTWARE ARCHITECTURE

The VM instance on creation itself is built with Linux Ubuntu 18.04 LTS as the operating system. The boot disk associated with VM comprises complete Ubuntu 18.04 LTS with essential security features, updates, and upgrades. In order to improve the security and usability of the server, the following subsequent actions have been carried out, which helped in completing basic setup associated with the operating system. The persistent disk incremental snapshot architecture is illustrated in Fig. 1.
4.5 SERVER MONITORING

The basic concept of server monitoring is to ensure a server or server infrastructure is functioning as it should be. It is important to understand effective server monitoring allows us to take this a step further to enable effective performance testing to allow us to proactively pick up on any issues or vulnerabilities the server might have.

Server monitoring involves monitoring many different aspects of a network/server infrastructure. The server hardware, operating system, applications running on the operating system, network traffic, memory, and disk utilization and CPU usage are a few examples of top-level items monitored in common server infrastructure. In more depth monitoring can be performed such as looking as disk queue length, memory pages per second and total network bytes per second. Many other applications or hardware-specific monitoring can be configured and setup depending on which pieces of information are critical to business and support needs. Monitoring can be further enhanced on top of using out of the box OS monitoring by using applications such as Appdynamics which provides a very in-depth application monitoring platform.

4.6 SERVER MONITORING AGENT

NETDATA PHM

Netdata is distributed, real-time, performance and health monitoring for systems and applications. It is a highly optimized monitoring agent that can be installed on almost all systems and containers. Netdata provides unparalleled insights, in real-time, of everything happening on the systems it runs (including web servers, databases, applications), using highly interactive web dashboards. It can run autonomously, without any third-party components, or it can be integrated into existing monitoring toolchains (Prometheus, Graphite, OpenTSDB, Kafka, Grafana, etc).

Netdata has a quite different approach to monitoring. Netdata is a monitoring agent that we install on all our systems. It is a metrics collector - for system and application metrics (including web servers, databases, containers, etc), a time-series database - all stored in memory (does not touch the disks while it runs), a metrics visualizer - super fast, interactive, modern, optimized for anomaly detection, an alarms notification engine - an advanced watchdog for detecting performance and availability issues. All the above, are packaged together in a very flexible, extremely modular, distributed application. Netdata is open-source, free, super-fast, very easy, completely open, extremely efficient, flexible and integrate-able. It has been designed by SysAdmins, DevOps and Developers for troubleshooting performance problems, not just visualize metrics.

Netdata is a Google Visualization API datatable and data source provider, so it can directly be used with Google Charts, which is an added advantage in server metric monitoring and recording process.
| Function | Description |
|----------|-------------|
| Collect | Multiple independent data collection workers are collecting metrics from their sources using the optimal protocol for each application and push the metrics to the database. Each data collection worker has lockless write access to the metrics it collects. |
| Store | Metrics are stored in RAM in a round-robin database (ring buffer), using a custom-made floating-point number for a minimal footprint. |
| Check | A lockless independent watchdog is evaluating health checks on the collected metrics, triggers alarms, maintains a health transaction log and dispatches alarm notifications. |
| Stream | A lockless independent worker is streaming metrics, in full detail and in real-time, to remote Netdata servers, as soon as they are collected. |
| Archive | A lockless independent worker is down-sampling the metrics and pushes them to backend time-series databases. |
| Query | Multiple independent workers are attached to the internal web server, servicing API requests, including data queries. |

Netdata data collection is extensible - we can monitor anything and can get a metric for. Its Plugin API supports all programming languages (anything can be a Netdata plugin, BASH, Python, Perl, node.js, java, Go, Ruby, etc). For better performance, most system-related plugins (CPU, memory, disks, filesystems, networking, etc) have been written in C. For faster development and easier contributions, most application-related plugins (databases, web servers, etc) have been written in python.

Designed to be installed on each system, without interrupting applications running on it. It operates according to the memory requirements specified by the user, using only idle CPU cycles. Once the application begins, it will not perform disk I/O beyond logging. The tool saves to disk at the end of its execution and reloads at startup. By default, it contains certain plugins that collect key system metrics, but its behavior is extensible by using its plugin API.

**Methodologies**

**5.1 Virtual Machine**

The proposed system aims in deploying a new type of snapshot backup process for effective backup of persistent disks associated with public cloud virtual instances. For that purpose, a new VM instance has been created from scratch and installed with Linux Ubuntu 18.04 LTS Bionic Beaver operating system. This server will server dynamic content of various websites and web applications via HTTP and HTTPS.
protocol. Along with the fresh operating system, dependencies such as PHP, MySQL, EngineX and many more were also installed for functioning as a perfect web server.

5.2 Server monitoring

Various server performance and health monitoring tools available were compared based on compatibility and features and netdata PHM tool has been chosen which meets all the requirements of this system. The netdata installer has been customized based on our system requirement and successfully installed on the VM instance. Metrics collected by netdata PHM tool from server will be processed by Google Visualization datatable API and the values are recorded and exported for further processing.

5.3 Conventional Persistent Disk snapshot

A conventional full backup will be done on the running VM instance's persistent disk using bash programming, stored on root of the linux operating system. The process will be automated using crontab installed with the operating system. The server’s performance and health will be continuously monitored and recorded during normal operation as well as during backup process. Deviations in server performance will be amazed and kept ready for comparison with other modes of snapshots.

5.4 Incremental Persistent Disk snapshot

This is the second stage of backup process in which an incremental backup will be performed on VM instance's persistent disk. As followed earlier, using netdata PHM tool, the server’s performance will be continuously monitored during incremental snapshot backup process. The deviations in server performance will be analyzed and the same will be compared with previous set of data in which conventional snapshot has been done.

5.5 NETDATA INSTALLATION

Netdata is a monitoring agent. It is designed to be installed and run on all our systems: physical and virtual servers, containers, even IoT. The best way to install Netdata is directly from the source. There are two major steps involved in netdata installation namely Prepare the system, Install the required packages on remote VM instance and finally Install Netdata.

In the preparation phase, few dependencies must be installed prior to netdata which has been completed by running the following Linux bash commands as root user in the remote server. Once the dependencies are successfully installed, shell script has been prepared for the installation of netdata. In our case of monitoring server for snapshot creation, netdata installation has been customized to meet the requirements of the system. The shell script “installer.sh” has been modified and presented below.

Upon successful installation, netdata dashboard can be accessed via TCP port “19999”. In our case, the netdata dashboard can be directly accessed by visiting the following URL:

http://35.200.144.133:19999
Few snapshots of netdata performance and monitoring agent is displayed in Figs. 2,3,4,5 and 6 respectively.

### 5.6 NETDATA REST API

One of the extended features of netdata is its integrated support for REST API. This API helps in integrating netdata with third-party applications to export the metrics collected for performance and health monitoring of the remote VM instance. Netdata tool, when linked with Google Visualization API Reference, can collect and export various metrics and information related to the server.

Netdata is a Google Visualization API datatable and data source provider, so it can directly be used with Google Charts. Using the REST API the server performance can be monitored as well as recorded in real-time. Also, the data recorded can be exported in the form of charts, graphs and CSV documents. The below illustrations shows single and multi-chart data collected using the REST API.

Netdata combined with Google visualization datatable API is now ready and can now be used for perfectly monitoring and recording the server performance while performing various modes of snapshot backup process which will be done in the next phase system.

### Results And Discussions

Using the Netdata PHM tool, various performance and health factors of the server were measured with a conventional and incremental backup which were discussed in detail as follows. The following charts indicate the performance and health of the virtual instance between the timer interval of 10.00 to 11.00 IST. The first 30 minutes (10.00 to 10.30) indicates the server performance and health with an ongoing conventional backup process and the next 30 minutes (10.30 to 11.00) indicates the same server performance and health with an ongoing incremental backup process.

#### CPU PERFORMANCE

Figure 9 shows the combined total CPU utilization of the VM instance with a conventional and incremental snapshot. It can be seen from the chart that the total CPU utilization was high during the conventional snapshot process.

A maximum peak utilization of 80% was attained by the CPU during a conventional snapshot backup process. In the case of an incremental snapshot process, it can be noted that the total CPU utilization is almost linear and capped at 40%. Thus, the implementation of the incremental snapshot has reduced the total CPU utilization by 50% compared to conventional backup and improved the CPU health and performance allotted for other essential processes like steal, softriq and users.

CPU pressure is a state wherein the CPU is fully occupied with currently assigned tasks and there are more tasks in the queue that have not yet started. The CPU pressure parameter of the VM instance is
illustrated in Fig. 10. During the conventional backup process between 10.00 to 10.30, it can be seen that CPU pressure was high and reached a peak value of 32%.

But in the case of the incremental backup process, the CPU pressure has been found to be low with an average peak of 9%. A drastic reduction in CPU pressure was observed while using incremental backup for the VM instance.

Figure 11 shows the uptime of apps running in the VM instance during the snapshot backup process. From the chart, it can be clearly understood that the uptime of apps has drastically increased in the second half of the timeline, i.e., incremental backup process. Due to low CPU utilization and CPU pressure, the uptime of apps such as python.d, httpd, PHP and SQL have increased and resulted in better performance of the VM instance for end-users requests.

An interrupt is a way to get the CPU to do something else for a while. Typically an interrupt is caused by an external event, like a timer expiring or an I/O operation finishing, or something like a memory error being detected. When an interrupt occurs, the hardware of the CPU will save the state of the running program (register contents, program counter, processor status word) and jump to an unrelated (to the running program) bit of code. What code runs for an interrupt is determined by the operating system.

Figure 12 shows the number of interrupts per second of the VM instance for both conventional and incremental backup process. From the chart, the CPU interrupts were high during the conventional backup process due to the full replication of the persistent disk. This has affected the real-time performance.

The implementation of the incremental snapshot backup process has reduced the system interrupts as seen in Fig. 5.5. This is due to the differential block backup process associated with the VM instance.

**DISK PERFORMANCE**

Figures 13 and 14 show the persistent disk I/O usage and I/O pressure during conventional and incremental snapshot backup process. Since the conventional backup process makes an exact copy of the disk attached to VM, the amount of data transferred and the data transfer rate was higher compared to the incremental backup process, in which only new data blocks are replicated.

A peak of 1.95 MB/s data transfer rate was found in conventional backup, whereas in the case of the incremental backup process, the peak data transfer rate is 0.6 MB/s which is three times lesser. The impact was observed in both ingress as well as egress data transfers and I/O pressure of the persistent disk.

**NETWORK PERFORMANCE**

Network performance is an integral consideration when it comes to running a successful enterprise. As networks become ever more complex, the challenges, dangers, and potential complications likewise
increase. As such, the standard network performance metrics used in the past are not up to the task of accurately measuring today’s complex, high-speed networks.

Figure 15 shows the IPv4 TCP socket allotment of the VM with conventional and incremental snapshot backup processes. During the incremental snapshot backup process, it can be seen that more TCP sockets were made to be allotted by the VM due to the high volume of data transfer involved. But during the incremental snapshot process, the TCP socket allotment has reduced at an average of 60 sockets compared to 140 sockets allotted for the conventional snapshot process. This improves the network performance of them by allotting more packets to users and less for the backup processes associated with the VM instance.

The telnet command establishes a TCP connection with the host on the port corresponding to the discard service. This is exactly the type of service we need to see what happens when a connection is established and terminated, without having the server initiate any data exchange. Figure 16 clearly illustrates the number of connection aborts per second. The connection abort rate was higher during the conventional backup process with a peak value of 2.4 connections per second. The greater number of aborts are since the VM instance network was flooded with backup process packet requests rather than user requests. This has been solved by the implementation of the incremental backup process, in which the number of aborts has reduced to 1.7 per second and the average aborts lie below 0.5 aborts per second, thereby increasing the reliability of network connection from the user end.

**REVERSE PROXY PERFORMANCE**

A proxy server is a gateway for users to the Web at large. Users configure the proxy in their browser settings, and all HTTP requests are routed via the proxy. A reverse proxy is a gateway for servers and enables one web server to provide content from another transparently. As with a standard proxy, a reverse proxy may serve to improve the performance of the web by caching; this is a simple way to mirror a website. Load balancing a heavy-duty application, or protecting a vulnerable one, are other common usages. But the most common reason to run a reverse proxy is to enable controlled access from the Web at large to servers behind a firewall. The proxied server may be a web server itself, or it may be an application server using a different protocol, or an application server with just rudimentary HTTP that needs to be shielded from the web at large.

Figure 17 shows the number of reverse proxy requests sent to apache daemon per second. There is an increase in the number of requests during the conventional backup process, but it was found to reduce when the VM instance snapshot process was changed to incremental type. Reduction in apache requests enables good performance for the VM to serve web pages to end-users with a minimum of 500 bad gateway errors.

Figure 18 shows the lifetime average of apache requests created and served by the VM instance network with conventional and incremental backup processes. During the conventional backup process, the average request received was high and the volume of data served to users was low in the VM instance.
network. This has resulted in poor network performance, increased latency, and increased time to the first-byte performance of domains hosted in the webserver. This issue has been solved by switching to an incremental backup process in which, more bandwidth has been allotted to domains instead of backup process daemon and thereby improving lifetime and bandwidth of hosted domains. It was found that the number of bad requests and 500 internal server errors were minimum during the incremental backup process compared to the conventional backup process.

**Conclusion And Future Enhancement**

In the proposed system, in order to study and analyze the snapshot creation performance of a remote cloud server, the following works have been conducted and are successfully completed. Virtual Machine has been identified and created with requirement hardware infrastructure. The operating system has been installed on the Virtual Machine and various dependencies required for the smooth operation of the server have been installed alongside. Methods to access remote VM instance has been identified and the required setup has been completed for root-level access. LEMP stack package has been installed on the server to change it into a live production server and the same has been tested successfully. Net data has been selected as the PHM tool to monitor the server continuously and the same has been installed successfully on the server and the same is available for public access. Integration of net data and Google visualization data table API has been completed to export and record metrics collected from the net data PHM tool for post-processing.

Then, the VM instance along with the PHM tool has been used to conduct various experiments. The experiments on VM instance with a conventional snapshot and the performance of the server has been monitored and recorded using netdata and Google visualization data table API. Also, the same set of experiments has been conducted with incremental and differential snapshot backup procedures and the performance of the server has been monitored and recorded using netdata and Google visualization data table API. The recorded metrics data of conventional snapshot, incremental snapshot and differential snapshot associated with the VM instance has been compared and the performance improvement was determined and analyzed. From the performance and health observations of Virtual Machine instance operated with conventional and incremental snapshot backups, the following conclusions have been drawn. The cloud-hosted Virtual Machine's performance has improved while using an incremental snapshot backup process. The CPU utilization has drastically reduced, and the core availability has increased for serving essential process requests. The performance of I/O process has improved and resulted in better data transfer and bandwidth allotments to ingress and egress packets. The disk usage also showed significant improvement in packet size and virtual memory allotments for incremental snapshot backups. The network performance has improved in quality and transfer rates. The number of bad requests and 500 internal server errors have reduced. The average lifetime of apache reverse proxy requests also improved. This concludes the system work and the successful completion of system has resulted in time and energy saving of the virtual instances without minimal downtime by the effective implementation of incremental snapshot backup system.
In future, the incremental snapshot backup technology can be applied to a versatile range of cloud servers and the performance of these servers can be recorded for further analysis. The metrics obtained from these experiments can be compared with other snapshot technologies such as differential snapshots, continuous snapshots etc., and this present snapshot technology system can be improved further.

Declarations

ACKNOWLEDGMENT

S.Veena, is a member of ACM and Life member of ISTE. She has published and presented a total of 40 papers in Conferences and Journals and she has also published 1 Book. Her areas of interest are Data Mining, Network Security and Cloud Computing. I would like to thank the Management I for their motivation and support given to carry out research activities in our college.

D.John Aravindhar, is a member of ACM Professional Society. He has published and presented a total of 40 papers in Conferences and Journals and he has also published 1 Book. His areas of interest are Cloud Computing, Data Mining and Data Science. I would like to thank the Management for their motivation and support given to carry out research activities in our college.

References

1. Saxena, A. (2018). Modeling data backups as a batch-service queue with vacations and exhaustive policy. Computer Communications – Elsevier, 128, 46–59.
2. Aldwyan, Y. (2019). Latency-aware failover strategies for containerized web applications in distributed clouds. Future Generation Computer Systems – Elsevier, 101, 1081–1095.
3. Nguyen, T. H., Forshaw, M., & Thomas, N. (2015). Operating policies for energy efficient dynamic server allocation. Electronic Notes in Theoretical Computer Science, 318(C), 159–177.
4. Guan, X., Choi, B. Y., & Song, S. Energy efficient virtual network embedding for green data centers using data center topology and future migration, Computer Communications 69 (2015) 50–59.
5. Chang, V., & Wills, G. (2016). A model to compare cloud and non-cloud storage of big data, Future Gener. Comput. Syst. Int. J. Esci., 57, 56–76.
6. Van de Ven, P. M., Zhang, B., Schörgendorfer, A., Distributed backup scheduling: modeling and optimization, 2014 Proceedings of the IEEE INFOCOM, (2014), pp. 1644–1652.
7. Xia, R., Machida, F., Trivedi, K. S., A Markov decision process approach for optimal data backup scheduling, Proceedings of the Forty-Fourth Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2014, Atlanta, GA, USA, (2014), pp. 660–665.
8. Claeyts, D., Dorsman, J.-P., Saxena, A., Walraevens, J., Bruneel, H., A queueing-theoretic analysis of the threshold-based exhaustive data backup scheduling policy, Proceedings of the International Conference on Numerical Analysis and Applied Mathematics, (2016).
9. Boullery, D., Schörgendorfer, A., Van de Ven, P., & Zhang, B. (2016). Balanced distributed backup scheduling. *US Patent, 9, 244,777.

10. Wang, M., Zhu, L., & Zhang, Z. (February 2016). Risk-aware intermediate dataset backup strategy in cloud-based data intensive workflows. *Future Generation Computer Systems, 55*, 524–533.

11. Marco, Gramaglia, Manuel, & Urueña, Isaias Martinez-Yelmoc, Off-line incentive mechanism for long-term P2P backup storage, Computer Communications, Volume 35, Issue 12, 1 July 2012, Pages 1516–1526.

12. Google Cloud Compute Engine Documentation, Virtual Machine instances, https://cloud.google.com/compute/docs/instances.

13. Google Cloud Compute Engine Documentation, REST Resource: firewalls, https://cloud.google.com/compute/docs/reference/rest/v1/firewalls.

14. Linux Ubuntu 18.04 LTS Bionic Beaver, Documentation, I. https://tutorials.ubuntu.com/tutorial/tutorial-install-ubuntu-desktop.

15. Initial Server Setup with Ubuntu 18.04 LTS, Ocean, D. https://www.digitalocean.com/community/tutorials/initial-server-setup-with-ubuntu-18-04.

16. LEMP stack Installation Documentation on Ubuntu 18.04 LTS, Ocean, D. https://www.digitalocean.com/community/tutorials/how-to-install-linux-nginx-mysql-php-lemp-stack-ubuntu-18-04.

17. Netdata, G., & Repository, https://github.com/netdata/netdata.

18. Netdata, I., & Guide, https://www.netdata.cloud.

19. Google Visualization datatable API Reference Documentation, https://developers.google.com/chart/interactive.

**Figures**
Figure 1

Incremental Persistent Disk snapshot architecture
**System Overview**

Overview of the key system metrics.

**cpu**

Total CPU utilization (all cores). 100% here means there is no CPU idle time at all. You can get per core usage at the CPUs section and per application usage at the Applications Monitoring section.

Keep an eye on iowait (3.0%). If it is constantly high, your disks are a bottleneck and they slow your system down.

An important metric worth monitoring, is softirq (0.00%). A constantly high percentage of softirq may indicate network driver issues.

**Figure 2**

Netdata dashboard view – CPU
Figure 3

Netdata dashboard view – Load & Disk usage
Figure 4

Netdata dashboard view – RAM & Network usage
Figure 5

Netdata dashboard view – Network sockets & packets
systemd Services

Resources utilization of systemd services. netdata monitors all systemd services via CGROUPS (the resources accounting used by containers).

**cpu**

Systemd Services CPU utilization (100% = 1 core) (services.cpu)

**mem**

Systemd Services Used Memory without Cache (services.mem_usage)

---

**Figure 6**

Netdata dashboard view – Systemd performance
Figure 7

Single chart Google visualization datatable API
Figure 8

Multi chart Google visualization datatable API

Total CPU utilization (system.cpu)

Figure 9

Total CPU utilization
Figure 10

CPU Pressure

Figure 11

Apps Average Uptime

CPU Interrupts
Figure 12

CPU Interrupts

Figure 13

Disk I/O usage

Figure 14

I/O Pressure
Figure 15

IPv4 TCP Sockets (ipv4.sockstat_tcp_sockets)

Figure 16

TCP Connection Aborts (ip.tcpconnaborts)

TCP Connection Aborts

Lifetime Avg. Requests/s (apache_local.reqpersec)

Lifetime Avg. Bandwidth/s (apache_local.bytespersec)
Figure 17

Apache Requests

Figure 18

Lifetime average of requests