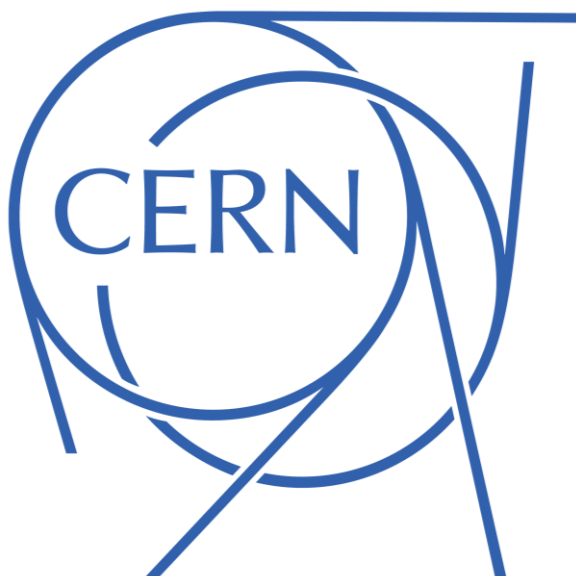


Summer Student Program 2024: Compress Backup Data from CBack

LANTIGNY Valentin



18/09/2024

—

Contact : v.lantigny@gmail.com

—

Supervisor : VALVERDE
CAMESELLE Roberto



PROCESSUS

Abstract

The primary motivation behind my project at CERN was to address the growing need for efficient data storage management. User data at CERN is stored in Restic repositories, and we utilize a custom-developed software named CBack to perform automatic daily backups of this data. With the release of Restic version 0.14.0, a new feature was introduced that allows for the compression of repository data, including backup data. This feature presents a significant opportunity to optimize storage space. Given that CERN's user data amounts to a staggering 8 petabytes, leveraging this compression capability is crucial for enhancing storage efficiency and managing resources effectively. This project aims to implement and utilize this new feature to compress the vast amount of user data, thereby achieving substantial space savings.

Process of Data Compression

With the release of Restic version 0.14.0, a significant enhancement was introduced in the form of file format versions 1 and 2 for repositories. Historically, repositories were created in version 1 by default. However, version 2 repositories now support data compression, offering a new avenue for optimizing storage. While it is possible to convert an existing repository from version 1 to version 2, this conversion alone does not automatically compress the data. To leverage this new feature, we can use the Linux command:

```
$ restic backup --compression="auto"
```

This command allows us to perform backups while compressing all data added since the previous backup. However, it does not compress data that was already present before the latest backup. To address this, we use another command:

```
$ restic prune --repack-uncompressed
```

This command compresses the older, uncompressed data, ensuring that all data within the repository benefits from the new compression capabilities. By systematically applying these commands, we can significantly reduce the storage footprint of user data, making the most of the 8 petabytes of storage available at CERN.

Development of the Compression Script

After conducting several tests to understand the functionality of Restic and to evaluate the effectiveness of its compression feature, I found that the initial tests suggested a potential compression rate of up to 50%. However, this rate is highly dependent on the file formats used within a repository. The core of my project involved developing a Bash script to automate the compression of repositories sequentially on a virtual machine (VM). The script, named `compression_script.sh`, includes several key functionalities and can be executed with the following command:

```
$ ./compression_script.sh -f REPO_FILE -m MIN_SIZE -M MAX_SIZE -l LINE --no-debug
```

Here are the main features of the script:

- **-f**: Allows the user to specify a file containing the URLs of the repositories to be compressed. This enables batch processing of selected repositories rather than compressing the entire list.
- **-m** and **-M**: These options allow the user to set the minimum and maximum sizes of the repositories to be compressed. If not specified, the script processes all repositories.
- **-I**: This option is used to ignore the first few URLs in the file specified by **-f**. This is particularly useful if the script crashes and the user does not want to restart the entire list.
- **--no-debug**: By default, the script does not execute commands that affect user data, allowing for a dry run to ensure everything is functioning correctly. Enabling this option triggers the actual execution of the commands.

By automating the compression process with this script, we can efficiently manage and reduce the storage footprint of user data at CERN, ensuring optimal use of the available 8 petabytes of storage.

Initial Testing and Implementation

The initial compression tests were conducted on replicated data rather than the actual scientific data from CERN. Using the ***rclone*** command, I created copies of real data to perform these tests. Once the process was functioning correctly, I proceeded to compress the repositories of the IT department members. With the successful implementation of these tests, we have now initiated the compression of repositories for other users.

To monitor the progress, I developed a second script to count the number of repositories currently in version 1 on CBack. Initially, there were over 46,000 repositories in version 1. I integrated this counter into the compression script to provide a real-time tally as repositories are compressed. This integration ensures that we can track our progress effectively and verify the success of the compression process.

Monitoring and Performance Tracking

Once we had a script that worked well in test conditions, we needed to ensure it performed effectively in a live environment. To achieve this, I deployed two key tools: node exporter and Prometheus.

Node exporter is a tool that collects hardware and operating system metrics from the VM where our script runs. Each time the script compresses a repository, it sends specific information to the node exporter, including:

- The size of the repository before compression
- The size of the repository after compression
- The time taken to compress the repository

These metrics are crucial for understanding the efficiency and impact of the compression process.

Prometheus is a powerful monitoring system that collects and stores these metrics from the node exporter. It allows us to query and analyze the data in real-time. Prometheus then forwards this data to **Grafana**, a visualization tool used by the IT department. Grafana enables us to create detailed dashboards that display various metrics and trends over time.

Scaling Up with Multiple Virtual Machines

By using this setup, we can continuously monitor the script's performance, ensuring it compresses the repositories effectively. This monitoring system provides valuable insights, such as the total space saved and the time efficiency of the compression process, which are essential for optimizing our storage management at CERN.

Based on the initial real-world tests, I calculated the compression speed and extrapolated it to the total 8 petabytes of data. The estimated compression time was approximately 1300 days, which is far too long. This extended duration is primarily due to the limited computational resources of the single VM I was using. To address this, deploying multiple VMs to work in parallel would be a more efficient solution.

To facilitate this, I explored CERN's Puppet service. **Puppet** is a configuration management tool that allows us to define a template for VMs with pre-installed services. This is extremely useful when deploying multiple VMs that need to operate in the same manner. By using Puppet, we can quickly and consistently set up several VMs, each equipped with the necessary tools and configurations to perform the compression tasks concurrently. This parallel processing approach significantly reduces the overall compression time, making the project more feasible and efficient.

Coordinating Multiple Virtual Machines

Once the possibility of deploying multiple VMs was established, it became clear that while each VM could work independently, they were all part of a common project. To perform their tasks correctly, the VMs needed to share certain information, such as:

- the number of remaining uncompressed repositories
- the list of already compressed repositories to avoid redundant work

To facilitate this, I utilized CERN's **OpenStack** service, which allows for the allocation of virtual resources, including a shared storage space. This shared storage, enabled by CERN's **Ceph** storage system, allows multiple devices to access and modify the same data. I set up this shared space and made it accessible to all VMs using my Puppet template. Now, each script can read and modify the necessary files, ensuring seamless collaboration and efficient compression across all VMs.

Conclusion

After running the project for several weeks with two VMs, we have processed nearly 400 terabytes of data using the compression script. Currently, we are achieving a compression rate of over 16%, which, if maintained, could potentially save up to 1.3 petabytes of storage by the end of the operation. This significant reduction in data size underscores the efficiency and effectiveness of our approach, highlighting the potential for substantial storage savings at CERN.
