AUTOMATED CONFIGURATION OF DISK ARRAYS FOR CLUSTER NODES IN LINUX

Rosen Hristev, Magdalena Veselinova, Kristiyan Kolev

Abstract. *Adding new computing power to existing clusters is a time-consuming process for IT departments. In most cluster systems, each node has local disk space on which the operating system is stored, as well as local space for it. It is recommended that the nodes in a cluster system be of the same type, such as hardware and operating system, so the process of configuring local disk space can be automated. This would lead to the optimization of the human resources responsible for the scalability of the cluster. In the research conducted, the differences between the software and hardware RAID controllers, the different types of RAID arrays, as well as the local disk space of a virtualization cluster with high availability – Proxmox VE are considered. A script for a Linux-based operating system is presented to automatically configure software arrays for data storage.*

Key words: RAID arrays, RAID controllers, Virtualization, Linux, Cluster.

Introduction

In recent years organizations continue to add software-based components to their storage infrastructure. This includes RAID (Redundant array of independent disks) to increase data storage capacity and reduce risks of data loss [1]. In most cluster systems, each node has a local disk space on which the operating system is stored, as well as a local disk space for it. It is recommended that the nodes in a cluster system should be of the same type, such as hardware and operating system, the process of configuring local disk space is a process that can be automated. This can improve the implementation of optimization algorithms, such as presented in [2] and [3]. This would lead to the optimization of the human resources responsible for the scalability of the cluster.

Differences Between Software and Hardware RAID Arrays

RAID is a virtual storage resource using multiple storage devices. In a specific sense, RAID uses an array of several disk drives, which creates a fault-tolerant and available system. The array is managed by a controller that connects one or more computing devices to it. RAID presents disks to users as logical storage resources. There are many different RAID storage options available, and costs can vary as widely as the choice of drives and controllers available.

The software in the operating system manages the software RAID [4], while the controllers independent of the operating system manage the hardware RAID, this is the key difference between them. They also differ in price, performance, and access speed.

In hardware RAID, all drives are connected to a hardware RAID controller that is on a separate RAID card, a different server or built into a different motherboard. Hardware RAID controllers physically manage RAID arrays, configurations, and support multiple RAID levels. Since RAID is managed and handled by the controller board, there is no additional load on the server CPU (Central processing unit). Data access is usually faster with this method. The controller manages the drives independently of the connected computer and does not need to use processing power. In case of disk failure, the drive can be easily removed and replaced.

One of the disadvantages of using hardware RAID is that it can be more expensive, although it is more reliable because it does not take processing power away from the drives. Another disadvantage is the possibility of incompatibility with the associated operating system. Performance issues can occur when using different technologies, such as SSDs (Solid state drives).

Software RAID uses the functionality of the RAID software or RAID driver built into the server operating system. No additional equipment is required to connect storage devices with this method. However, this can increase the overall processing load on the server and possibly slow down RAID calculations as well as other processes performed on the drive.

Software RAID is usually less expensive than hardware because a RAID controller is not required. It can be implemented in a single operating system and used by multiple devices. The controller, in the software RAID, manages the drives as part of the connected computer. Data access can be slower compared to hardware RAID. Connected devices must be compatible with the connected operating system.

RAID Levels

There are different RAID levels, each optimized for a specific situation.

RAID 1 (Mirroring)

Data is stored twice by being written to both the storage device and the mirror device [5]. If a drive fails the controller uses either the data drive or the mirror drive. For a RAID 1 array, we need at least 2 disks.

RAID 1 offers excellent read speed and writes speed that is comparable to that of a single drive. The data does not need to be restored, in the event that a device fails, it just needs to be copied to the replacement device. The effective storage capacity is only half of the total capacity of the device because all data is written twice, which is a major drawback of this kind of RAID level.

RAID 5 (Striping with parity)

RAID 5 is a combination of three or more hard drives that function as a single logical drive and thus outperform individual data carriers in terms of reliability and read speed. The combination of drives and that the system distributes user data and parity information to all integrated hard drives is what makes this RAID level special. Unlike other systems, files are not saved in multiple versions thanks to parity blocks. Compared to using individual drives, storage capacity decreases, but RAID 5 networks retain a relatively large portion of their original capacity. RAID 5 is a cost-effective solution to increase read speed. Data partitioning allows parallel access to multiple parts of a concatenated block of data so that requesting devices can complete the read process much faster.

An advantage of RAID 5 is increased reliability. If a hard drive fails or if data on the drive is lost for other reasons, operations can still be maintained. A major drawback is the write speed of the data carriers in a RAID 5 system. Compared to individual drives and other RAID levels, the write speed is much lower. Each writing process on the hard disk cluster is associated with an additional read step to check and recalculate the available parity information. One more step is required to allocate parity data for the newly stored user data on the disks.

RAID 6 (Striping with double parity)

RAID 6 is one of the most common and useful RAID levels in use today. RAID 6 is based on RAID 5 and has a different parity level. This makes it safer than RAID 5, which is very important, but it affects the data writing because parity data is written to two drives [6]. This means they require at least 4 disks and can withstand up to 2 disks failing at the same time. Each writes operation requires the disks to read the data, read the first parity, read the second parity, write the data, write the first parity, and then finally write the second parity. RAID 6 is like RAID 5, but parity data is written on two drives.

Proxmox VE Data Storage

Data storage with the Proxmox VE high-availability cluster system is flexible. Using a virtual firewall on ProxmoxVE can provide increased security against potential threats to data and networks on cloud servers [7]. Individual nodes in the cluster can use local and shared data stores [8]. Different types of data can be stored depending on the type of storage. Proxmox VE supports all disk space utilization options available to Debian Linux, and the use of the `libpve-storage-perl` library allows disk arrays to be easily managed and configured through the WEB interface for managing the cluster system [9].

Installing a new cluster node is on local disk space for the physical machine, and by default, each node has a “local” disk space that essentially stores its data in `/var/lib/vz`. In it, each node can store virtual machines, containers, ISO (International Organization for Standardization) files, and backups. By default, this space is shared and divided with the system root. This would make it difficult to manage the cluster system if the corresponding node needs to be reinstalled, which would lead to the possibility of partial or complete loss of information.

To minimize the possibility of information loss when reinstalling a node, the available disk space on the machine can be divided into two parts. The first part should be for the root of the virtualization system, where `/var/lib/vz` will again be visualized as local disk space for the node, but not used. And a second part is to be used for local disk space to store ISO files, virtual machines, containers, and backups. This way, if the node needs to be reinstalled, it will avoid the need for the system administrator to first take care of the data stored in `/var/lib/vz`, since it will be on another partition of the disk space, and the administrator can focus on the main you are a task.

The second part of the local disk space, on which data other than the operating system and the high-availability virtualization cluster system will be stored, can be selected between:

- Directory
- ZFS (Zettabyte File System) pool
- BTRFS (B-Tree File System)
- LVM (Logical Volume Management)
- LVM-thin

Like the default local storage, Proxmox VE offers the possibility to map another directory from the node's file structure on which virtual machines, containers, backup files, and ISO images can be stored [10]. There is no restriction on the file system to be used for the directory and it is up to the administrator's preference. Dedicating another partition to store the directory data will avoid the need for the system administrator to first take care of the data stored in the directory before taking action to reinstall the node. The main disadvantage of directories is that they do not allow convenient and important functions such as a snapshot of a virtual machine or migration from one node to another.

ZFS pools are one of the most preferred storages for snapshotting and cloning. A big plus with them is that they can store both raw format and subvol. Pools can be accessed locally and host virtual machines and containers.

The structure of BTRFS pools is similar to directories, the main difference is that raw disks can be used as subvolumes, which expands the capabilities of directories and allows snapshots of virtual machines to be created, as well as offline storage migration [11].

LVM provides a higher level of abstraction where physical partitions from different disks can be merged into a single logical partition. One of the main advantages of LVM is that it allows online partition resizing without having to reboot the operating system. A major advantage of the Proxmox VE high availability cluster system is that the disks of the virtual machines and containers will be stored and used as a block, not as a file.

There are two types of LVM storage, that can be used with Proxmox VE, as it is shown in Table 1. With LVM storage, when a 40 GB disk is created, a 40 GB piece will be allocated to the logical volume, which will be dedicated to this disk. The disk volumes provided to the virtual machine in a real IT infrastructure many times exceed the volumes actually required for its proper functioning. LVM thin handles this thin-provisioning by allocating blocks only when information is written to them. Another important feature is that LVM does not allow the creation of snapshots, while LVM Thin supports them.

Name	**Storage Type**	**Content Types**	**Image Format**	**Shared**	**Snapshots**	**Clones**
Directory	file	images, rootdir, vztmpl, iso, backup, snippets	raw, qcom2, vmdk, subvol	no	qcow2	qcow2
BTRFS	file	images rootdir vztmpl iso backup snippets	raw, qcom2, vmdk, subvol	no	yes	qcow2
ZFS pool	file	images rootdir	raw subvol	no	yes	yes
LVM	block	images, rootdir	raw	possible	no	no
LVM Thin	block	images, rootdir	raw	no	yes	yes

Table 1. Local storage types in Proxmox VE

A script for automating the configuration and creation of a software raid

The process of creating and configuring disk arrays is universal and can be automated using scripts. Since there is a wide variety of hardware RAID controller manufacturers on the market, creating a universal script to automate the creation and configuration of the disk array is practically a difficult task. Different programs are responsible for creating an array of disks, depending on the operating system in software drives. For Linux-based operating systems, the packages responsible for creating a RAID array are `mdadm`. With the same type of packages and programs responsible for implementing the software RAID, the task can be automated using a bash script [12]. For the research, a script that scans the available disk drives on the server hardware was implemented, and from them creates two partitions on 2 uninitialized disks, one 100 GB for the system root and the rest of the disk to be used for local storage of the node.

All available devices on the system can be listed using `lsblk`. This process examines the Linux `sysfs` file system and the `udev` database to determine the existence of any block devices. Using command options, we can filter the output and direct it to the next part of the bash script.

The holder device and `subdrives` (such as sda1 and sda2) can be removed with the `-d` option. By selecting the `-e` 11 option, all devices below block 11 are removed from the results output. These are devices such as SCSI (Small Computer System Interface) CD-ROMs that are not necessary to achieve the intended purpose. The last two options are `-n` to skip the main output information line and `-o` with the `NAME` and `SIZE` parameters to output only them. This will allow us to sort the disks by size from large to small when we run the sort command with the options `-r` for reverse order and `-n` to compare a numeric value sorted by a key located in the second column that is selected by parameter - `k2`. By combining these commands we can get all the currently available drives on the system and store them in the `AllDrives` variable just with their names by using the awk command with parameters `"BEGINFS=" " $2>100 print $1,"` which also removes all drives for storage with a capacity of less than 100 GB.

By using the results of `lsblk` as a parameter to the `blkid` command, we can obtain file system and UUID (Universally Unique Identifiers) information for already assigned storage devices. In this way, it can be determined whether a disk is already in use, if a partition table is formed on it, the answer will indicate whether it is a DOS (Disk Operating System) or GTP table. If the disk has no partition table, it can be considered free and added to a list of free devices that are currently available.

```
COUNT=0
EXPECTED_DRIVES=2
PARTION=1
AllDrives=$(lsblk -l -d -e 11 -n -o NAME,SIZE |
 sort -rn -k2 | awk 'BEGIN{FS=" "} $2>100 {print $1}')
for i in $AllDrives
   do
   blkid "/dev/$i" &> /dev/null
   if [ $? != 0 ] && [ $COUNT -lt $EXPECTED_DRIVES ];
   then
   AvailableDrives+=(/dev/$i)
   (( COUNT++ ))
 fi
```

```
done
```

Once the available free drives are identified, the partitioning process can begin, which involves splitting each disk into two partitions that will be used to create two mirror images. Using a series of instructions, the disk can be divided into two main partitions: the first partition is with size 100 GB, to store the root of the operating system, and the second partition with the rest of the disk space. Change the partition type to "Linux raid autodetect" and save the changes. The same method is used for the second disc.

```
for Device in "${AvailableDrives[@]}"
do
  sudo echo -e "n\np\n1\n\n+100G\nn\np\n2\n\n\nt
             \n1\nfd\nt\n2\nfd\nw" | sudo fdisk ${Device}
done
```

Before proceeding with the creation of the RAID disks, the following correction should be considered regarding "NVME" disks. In addition to the number at the end of the drive, as for a SCSI drive (eg /dev/sda1 and /dev/sda2), the letter "p" followed by the partition number must be included. Once these specifics are taken into account, one can proceed with the creation of mirrored raid partitions.

```
for ((i=0;i<${#AvailableDrives[@]};i++));
do
    if [[ " ${AvailableDrives[i]} " == *"nvme"* ]]; then
            AvailableDrives[i]="${AvailableDrives[i]}p"
    fi
done
```

A software-based Linux RAID can be created using the mdadm command. The first RAID must be labeled md0 with type RAID1 (mirroring) and must be used to select each first partition of the two disks, partitions that are with size 100 GB. The second mirror partition is created in a similar way with the label md1.

```
for MD_DEVICE in {0..1}
do
   recipe="yes | sudo mdadm
   --create /dev/md$MD_DEVICE
   --level 1 --force --raid-devices
   $EXPECTED_DRIVES ${AvailableDrives[0]}$PARTION
   ${AvailableDrives[1]}$PARTION"
   eval $recipe
   (( PARTION++ ))
done
```

Conclusion

One of the activities when installing an operating system is configuring disk space on the machine. Nodes in a cluster system are no exception and the operation is repeated for them as well. In cluster systems, it is recommended that the nodes have the same configuration and settings, and adding a new compute node is a standard operation that takes time. The article discusses the different types of RAID arrays, as well as the most commonly used RAID levels. The ways in which the Proxmox VE high availability cluster system stores its data locally are described in detail. An automated script was created that creates software RAID 1 to be used for subsequent node setups and configurations. The script optimizes human resources responsible for cluster scalability.

Acknowledgments

This research was funded by the Bulgarian National Science Fund under Project KP-06-NP62/1.

References

[1] J. Hsieh, C. Stanton, R. Ali, Performance evaluation of software RAID vs. hardware RAID for Parallel Virtual File System, Proc. of *Ninth International Conference on Parallel and Distributed Systems*, 2002, 307–313, ISSN: 1521-9097, DOI:10.1109/ICPADS.2002.1183417.

[2] S. Monov, N. Pavlov, A. Rahnev, Improving Efficiency of Routing Transportation Equipment using Genetic Algorithms, *International Journal of Differential Equations and Applications*, Vol. 21, No. 1, 2022, 105–116, ISSN (Print): 1311-2872; ISSN (Online): 1314-6084, doi:10.12732/ijdea.v21i1.9, https://www.ijpam.eu/en/index.php/ijdea/article/view/5994/278.

[3] S. Monov, N. Pavlov, M. Dobreva, Custom WCF serialization, Proc. of National Scientific Conference “Education, Science, Society”, November 3–4 2022, Smolyan, Bulgaria, 928–937, ISBN: 978-619-7663-43-3 (online), https:

//uni-plovdiv.bg/uploads/site/filiali/smolian/NID/NNK%20-%202022/Sbornik_Conf_Smolian_2022-1.pdf.

[4] Jong-Hoon Kim, S. Noh, Yoo-Hun Won, An efficient caching scheme for software RAID file systems in workstation clusters, *Proceedings High Performance Computing on the Information Superhighway*, HPC Asia'97, 1997, 331–336, ISBN: 0-8186-7901-8, DOI:10.1109/HPC.1997.592169.

[5] D. Vadala, *Managing RAID on Linux: Fast, Scalable, Reliable Data Storage*, O'Reilly & Associates, Inc., 2003, ISBN: 978-1-565-92730-8.

[6] M. Gilroy, J. Irvine, RAID 6 Hardware Acceleration, *2006 International Conference on Field Programmable Logic and Applications*, 2006, 1–6, ISNN: 1-4244-0312-X, DOI:10.1109/FPL.2006.311196.

[7] Y. Ariyanto, B. Harijanto, V. Firdaus, S. Arief, Performance analysis of Proxmox VE firewall for network security in cloud computing server implementation, *IOP Conf. Ser.: Mater. Sci. Eng.*, 732 (2020) 012081, ISSN: 17578981, DOI:10.1088/1757-899X/732/1/012081.

[8] M. Simon, C. Cheng, *Proxmox High Availability*, Packt Publishing Ltd., 2014, ISBN: 978-1-78398-088-8.

[9] W. Ahmed, *Mastering Proxmox: Build virtualized environments using the Proxmox VE Hypervisor Third Edition*, Packt Publishing Ltd, 2017, ISBN: 978-1-78839-760-5.

[10] R. Goldman, *Learning Proxmox VE*, Packt Publishing Ltd, 2016, ISBN: 978-1-78398-178-6.

[11] Proxmox VE Storage, Available on https://pve.proxmox.com/pve-docs/chapter-pvesm.html#storage_btrfs (visited on 20.11.2022).

[12] R. Hristev, A. Golev, Cluster Systems. Algorithm for automated installation and addition of nodes in existing clusters, Proc. of *Anniversary International Scientific Conference "Computer Technologies and Applications"* (CTA'2021), 2021, ISBN: 978-619-202-702-5.

Rosen Hristev[1,*], Magdalena Veselinova[2], Kristiyan Kolev[3]
[1,2,3] "Paisii Hilendarski" University of Plovdiv,
Faculty of Mathematics and Informatics,
236 Bulgaria Blvd., 4003 Plovdiv, Bulgaria
Corresponding author: hristev@uni-plovdiv.bg