International Journal of Power Electronics and Drive Systems (IJPEDS)
Vol. 15, No. 2, June 2024, pp. 1117~1127
ISSN: 2088-8694, DOI: 10.11591/ijpeds.v15.i2.pp1117-1127

High-reliability uninterruptible power supply manager for critical virtualization cluster

Davide Piumatti, Andrea Galletto, Flavio Castagno
Department of Control and Computer Engineering (Dipartimento di Automatica e Informatica (DAUIN)),
Politecnico di Torino, Turin, Italy

Article Info

Article history:

Received May 2, 2023
Revised Nov 23, 2023
Accepted Dec 7, 2023

Keywords:

IT service management
Power grid fault resilience
Safety-critical applications
Server cluster reliability
Uninterruptible power supplies

ABSTRACT

Information technology servers are complex, delicate, and expensive systems; power grid interruption is one of the possible causes that can lead to hardware damage or logical disk structure damage, with severe consequences. Moreover, a power grid drop causes a sudden interruption of the services managed by a server. Uninterruptible power supplies (UPS) provide backup power when the regular power grid source drops. UPSs can only maintain a server for a short time. Therefore, it is necessary to introduce a blackout manager who can correctly shut down the server activities. Moreover, the manager must be able to restore the servers' status when the electrical grid power returns to its normal state. This paper proposes a possible UPS power manager able to manage servers during a prolonged electrical blackout. The UPS power manager identifies the blackout and keeps the servers safe by saving their state and shutting them down properly. Following the power grid restoration, the UPS power manager restarts the servers and restores their state. The proposed approach has been evaluated on a virtualization cluster used for critical activities at the Politecnico di Torino.

Corresponding Author:

Davide Piumatti
Department of Control and Computer Engineering (Dipartimento di Automatica e Informatica (DAUIN))
Politecnico di Torino
Turin 10129, Italy
Email: davide.piumatti@polito.it

1. INTRODUCTION

Almost all information technologies (IT) currently used are hosted and managed by servers devoted to performing one or more activities as services [1]. For example, many servers host and manage web pages, handle e-mails, perform complex data computing operations or managing critical systems [2], [3]. IT servers are safety-critical systems that must work properly and robustly [4]. Today, small and large companies use server clusters to easily manage their IT facilities scalability and performance [5]. A virtualization server cluster is a collection of multiple servers, called nodes, on which to execute numerous virtual machines (VMs) [5]. Each virtual machine is a low-performance server dedicated to performing at least one activity. Every VM has its own virtual-central processing unit (vCPU), a virtual-network interface controller (vNIC), an amount of random-access memory (RAM), and a hard disk drive (HDD) space. Each VM is an independent entity from the others. A VM can host a service (like a traditional physical server performs), or it can be used remotely by a user. The cluster management software, called hypervisor [5], virtualizes and manages the different VMs by assigning RAM, CPU, and HDD portions to each VM, considering the available cluster resources. A virtualization cluster simplifies the management of different servers organized in many independent VMs [5]. However, a virtualization cluster is a very complex system to manage and maintain, but it allows for handling many independent VMs.

A sudden cluster drop can cause the interruption of different services hosted in the VMs. A frequent external cause of a service interruption is a power grid drop. This unpredictable and uncontrollable external event brutally interrupts any operation in execution on the servers. In addition, sudden power failure can cause damage to the logical structure of the disks with possible data corruption. The power grid drop is frequently among the different causes of IT service interruption [6], [7].

Numerous techniques are typically adopted to avoid the power grid drop. Backup batteries, also called uninterruptible power supply (UPS) devices [8], are a common strategy used to maintain a server active during a temporary power blackout [9]. However, UPSs cannot power servers for extended periods; therefore, when the backup battery exhausts, different actions are performed to adequately shut down the servers powered by the UPS [10], [11]. The automatic shutdown of a virtualization cluster is not a trivial operation; in particular, for hibernating the state of the cluster and then restoring the state at the next cluster power-on. Some industrial solutions have been proposed in [10]–[13] by some companies, while a possible open-source solution is proposed in [14]. However, the commercial and open-source solutions are limited to a few typical cases. Preview study [10], [11] two possible commercial software for the automatic shutdown of a cluster are proposed. Cisco [10] proposes a power-down and power-up orchestration strategy based on Cisco HyperFlex Edge environments. HyperFlex is an infrastructure used to manage distributed clusters over many nodes. A DELL UPS Management plug-in for the VMware virtualization infrastructure is proposed in research [11]. The solutions proposed in previous researchers [10], [11] require the installation of software not available for all operating systems (OS) and not available for custom operating systems widely used in some servers (e.g., QTS for QNAP server). In addition, the solution proposed in [11] requires the installation of a plug-in available only for the VMware commercial hypervisor. The APC's PowerChute solution [12] can perform the shutdown of servers directly attached to one of the compatible UPS; moreover, PowerChute can't perform VMs or hypervisor shutdowns. For managing the virtualization cluster shutdown is necessary to use APC software called PowerChute Network Shutdown [12] available only for some compatible UPS and integrated only for VMware, Hyper-V, and NutaniX hypervisor solutions.

DELL UPS Management Software [13] is a DELL software that allows for managing the power off/on of a virtualization cluster; however, it is compatible only with devices manufactured by DELL. It also requires the installation of a plug-in available only for VMware commercial hypervisors. The solutions proposed in [10]–[13] can be applied only for commercial solutions compatible with the UPS Management software of some manufacturers. Considering the open-source solutions, Network UPS Tools (NUT) is compatible only with some UPS, as discussed in [14]. Furthermore, NUT requires installing software on all nodes and VMs of the cluster. NUT is compatible with VMware by installing a devoted plug-in [14] on the servers. In addition to VMware, there are other open-source hypervisors such as Proxmox or oVirt. NUT can be used on Proxmox or oVirt, as described in [15]; however, NUT for Proxmox or oVirt is an inflexible solution difficult to implement, especially for a complex cluster composed of numerous nodes and additional servers. NUT requires the configuration and implementation of numerous scripts; these scripts are necessary to realize the interface between NUT and Proxmox or oVirt hypervisors. In addition, the commercial and open-source solutions do not manage the multilevel VMs application. In a multilevel VMs application, two types of VMs are executed simultaneously in the same virtualization cluster. The first is the user VMs typically used as remote desktops, while the second is the VMs that expose services such as a server (called server VMs). The main difference between the two types of VM is that user VMs do not expose services but use the services exposed by servers, while server VMs expose services. The VMs of the second group can be considered virtual servers. In the multilevel VMs application scenario, the user VMs must properly close all operations performed on server VMs before shutting down; afterwards, when all user VMs are power-off, the server VMs can be shut down. Therefore, some VMs must be powered off in a specific order depending on the service exposed, as for multilevel applications with dependencies; e.g., Microsoft SharePoint Server depends on Structured Query Language (SQL) Server, so it is recommended to maintain SQL Server online until SharePoint Server is shut down. Hence, the VM hosting the SQL service must necessarily be powered off after the VM hosting SharePoint Server. Conversely, when it is time to get everything back online, SQL Server should start before SharePoint Server. Finally, the cluster power-off system must verify that the different VMs are power-down before proceeding with the hypervisor shutdown.

Different workload management strategies according to the energy expenditure of the servers have been considered in [16]–[19]. In particular, some strategies are aimed at reducing server energy consumption by switching off unused hardware resources [17] or by considering UPS losses [18], [19]; in UPSs, the power losses are due to the double conversion necessary for the accumulation of electricity in the batteries [17]–[20]. Furthermore, the UPS efficiency depends on the load connected to it, as discussed in [17]. The strategies proposed in [16], [17] aim to distribute the VMs on the cluster nodes to optimize the electrical power consumption considering the UPS efficiency.

The problem faced in this paper is the shutdown of a complex virtualization cluster following a long power grid drop. The dependencies between the different services managed by the VMs (user VMs and server VMs), the state of the VMs, the state of the additional supporting servers and the state of the network are considered during the controlled cluster shutdown. Moreover, the power-on recovery problem of the cluster after the power grid restoration is also faced in this paper.

This paper proposes a systematic methodology for the power-off and power-on orchestration of a virtualization cluster composed of numerous nodes and additional supporting servers. In particular, the methodology proposes a shutdown and power-on order of the whole cluster. The proposed methodology is automatically applied during a prolonged power grid interruption when the UPS battery energy is lower than the critical threshold chosen. The critical battery threshold is chosen according to the time required by the cluster to complete the shutdown of all nodes and additional servers. In order to ensure compatibility with different operating systems and technologies, the proposed approach exploits the main remote server management protocols [21], such as secure shell (SSH) and intelligent platform management interface (IPMI). The proposed approach hibernates the VMs to save their state and restore their execution at the end of the power blackout. This feature is handy for long computational simulation applications [22]; it allows to freeze of the simulation's state by VM hibernation and then resumes it after the blackout. Finally, this paper proposes a hardware and software infrastructure that constantly monitors the power grid and the UPS battery. The proposed infrastructure can autonomously perform the cluster power-on and power-off operations when necessary. The approach proposed is designed to be as general as possible and use widely available standards.

The methodology proposed in this paper differs from the proposed in [10]–[14]. In particular, the methodology proposed does not require installing an agent (i.e., a dedicated software like a client, a service, a plug-in, or a daemon) on each server, node, and VM. The methodology proposed uses services and protocols usually present and managed by standard operating systems (e.g., SSH). As discussed in [23], [24], an agentless solution introduces some advantages; an agent requires to be installed, configured, and maintained for each server and VM in the virtualization cluster. Also, the presence of an agent introduces possible compatibility issues with other software or operating systems. An agent requires resources to be performed (as CPU time, RAM space, and HDD time access and space); a poorly developed or outdated agent can introduce some vulnerability that can be used to attack a server or a VM. Also, if the agent stops working (crash scenario) the cluster shutdown system fails to control a server or a VM. Moreover, the power grid monitoring is entrusted to the UPS, which typically displays its status via a standard remote management protocol, like simple network management protocol (SNMP). In other words, the proposed methodology can be applied to different manufacturers UPS using the SNMP protocol, as described in RFC1628. Standard remote management server protocol avoids using dedicated agents in favor of services natively present in the operating system; these services are well-integrated and implemented directly by the operating systems developers. Furthermore, the standards are fully compatible with different operating systems; for example, for all Linux kernel OS such as Ubuntu, Centos, Debian, RedHat, FreeBSD, Android, MacOS, Solaris, and Windows operating systems. Finally, the solutions proposed in [16]–[19] do not face the problem of identifying an efficient shutdown procedure for a cluster to preserve the cluster status. Still, they identify some possible strategies to reduce server energy consumption and optimize UPS use.

In some critical applications, the UPS is used as a backup battery until a new alternative emergency power source is not started (e.g., a diesel generator) [25]–[27]. The backup generator strategy is not strictly considered in this paper; in this case, a server shutdown procedure is unnecessary. However, if the emergency generator doesn't work, it is necessary to implement an automatic shutdown strategy such as those proposed in this paper.

The remainder of the paper is organized as: i) Section 2 summarizes the UPS working, the virtualization cluster working, and the remote server management protocols typically used; ii) Section 3 discusses the proposed approach to perform the systematic and automatic power-off/on procedures, moreover, the hardware and software infrastructure used for power grid and UPS battery monitoring is discussed; iii) The case study and the experimental results are discussed in section 4; and iv) Section 5 draws some conclusions.

2. TECHNICAL BACKGROUND INFORMATION

This section provides some helpful information for the reader. The UPS functioning and the virtualization cluster functioning are briefly discussed. In addition, remote server management network protocols typically used are also discussed.

2.1. UPS working

A UPS can protect other electronic devices (e.g., servers) from power grid failures; as discussed in [28], a UPS can protect against overvoltage, brown-out, spike, electrical noise, grid frequency variation, parasitic harmonics, and temporary blackout. UPS receives in input the grid voltage and supplies a new stable and protected voltage at its output [29], [30]. Devices to be protected are connected to UPS and powered by it. During regular working, the UPS eliminates defects in the power grid [28] and charges the backup battery. During a power grid blackout, the constant voltage of the backup battery is used by an inverter for generating the UPS sinusoidal output. Modern UPSs are equipped with an Ethernet network interface that can be used to query the batteries and power grid status.

2.2. Virtualisation cluster

This section provides an overview of a virtualization system. The idea of virtualization is to provide a faithful and complete abstraction of the hardware of a physical machine [1], [31]. This abstraction allows generating a VM whose behavior is identical to a physical machine; thanks to this abstraction, the same software's and operating systems designed for physical calculators can also be executed by VMs. The hypervisor software, executed by a physical host server, implements the virtualization of different VMs [31]. Hypervisor isolates the operating system and physical hardware resources from VMs. Moreover, the hypervisor allows for creating and managing VMs. The physical server used by the hypervisor is called the host, while the VMs that use its resources are called guests. The hypervisor handles resources (including CPU, memory, and storage) as a pool of resources that can be easily reallocated between existing guests or on new VMs. For executing the VMs, the hypervisor needs some components of the host operating system, such as a memory manager, process scheduler, input/output stack, device driver, security manager, and network stack. The hypervisor manages and schedules the VMs resources, considering the available physical ones. With the virtualization system, executing several operating systems simultaneously (one for VM) and sharing the same hardware resources is possible.

In contrast to the single server virtualization structure, it is possible of realizing a virtualization system based on a cluster server. In this case, the hypervisor manages the hardware resources available on multiple networked servers. A cluster aims to distribute complex processing across multiple servers, called cluster nodes [31]. The cluster approach increases the computing throughput of the global system; moreover, it also increases the reliability of the virtualization system. In a virtualization cluster, it is necessary to introduce the concept of hyper-convergence, i.e., of a software infrastructure able to manage the hardware resources of multiple servers [31]. Hyperconvergence concentrates the hardware resources of multiple servers into a single unified system managed by the hypervisor. The hyper-convergence paradigm requires fast networks for exchanging data between the different cluster servers; moreover, it requires introducing distributed file systems for synchronizing the data on the storage of the different nodes [31]. Distributed file systems introduce different benefits to virtualization clusters. A VM can be easily migrated from one node to another by exploiting the VM virtual disk duplicated on all cluster nodes. Moreover, in the presence of a disk failure, or more generally, in the presence of a node failure, the hypervisor can exclude the hardware resources of the failed node and maintain the cluster active, and the VMs, with the remaining nodes.

2.2. Remote server management protocols

This last background section discusses the main remote management protocols typically used on servers [21]. In particular, the simple network management protocol (SNMP), intelligent platform management interface (IPMI), and Secure SHell (SSH) protocol are quickly analyzed. In addition, the wake on LAN (WoL) protocol used for starting a server is also discussed.

SNMP protocol allows and simplifies the configuration, management, and supervision of networked devices using a simple client-server architecture [21]. The management of a device is possible by reading or writing specific data, called "managed objects," organized in a hierarchical Management Information Base (MIB). Each object in the MIB is identified by a unique Object ID (OID). The structure of the MIB and the SNMP protocol is defined in RFC1628. Any device managed (e.g., UPS, server, router, and printer) has a standardized MIB, as described in RFC1628; this allows easy management of devices from different manufacturers. The SNMP operation is straightforward; it is necessary to indicate the IP address of the device, the OID of the interesting object, and the access credentials for obtaining the desired data.

SSH protocol allows for establishing an encrypted, remote session via a command-line interface with another host on a computer network [21]. SSH provides a remote shell that allows sending commands to a remote server. The RFC4254 describes the protocol's detailed operation.

IPMI is a standard that provides some interfaces for computer system administration [21]. This standard allows monitoring a server without requiring installing an operating system or dedicated software management. IPMI supports automatic system shutdown and reboot, remote power-up, and resource tracking capabilities. The cornerstone of IPMI is its hardware controller called the baseboard management controller (BMC), the BMC controller can work even with the server in a standby or turn-off state. The BMC controller independently checks the system's health and performs actions such as logging events, generating alerts, and restarting the system. Furthermore, the BMC is equipped with a dedicated network interface through which it is possible to monitor the server.

WoL is an Ethernet standard that allows starting a server from a remote location [21] using the target server's media access control (MAC) address. The Ethernet packet that triggers the wake-up procedure is called magic packet. A server that receives the magic packet starts its power-up process. Nowadays, almost all modern servers support WoL.

3. METHODOLOGY

This section discusses the systematic power-off and power-on procedures proposed. In particular, the first subsection discusses the proposed power-off/on orchestration manager, the algorithm used for triggering the automatic cluster shutdown and the automatic cluster power-on procedures. Finally, the second subsection discusses the orchestration manager architecture proposed; moreover, the architecture of the implemented code is shown and discussed in the last subsection.

3.1. Power-off/on orchestration manager

The proposed approach requires a PC devoted to performing the orchestration manager, called UPS manager. The UPS manager performs the proposed switching off/on procedures. In addition, the UPS manager monitors the UPS status by some SNMP queries, as discussed in section 2.3. The first part of Figure 1 shows the shutdown procedure proposed. In the proposed approach, the VMs are hibernated to restore the VM state at the next power-on. The shutdown procedure comprises 6 steps (from A to F). However, step F is optional and related to the UPS manager shutdown.

In the first step (step A), the active users' VMs are hibernated, i.e., the VMs that do not expose any final services, as discussed in section 1. Hibernation is performed by the hypervisor that creates and writes a virtual memory image of each VM on the storage disk. The VM hibernation command is performed via SSH using the hypervisor's application programming interface (API). Furthermore, a check about the VM state is implemented to verify the complete VM hibernation; after a timeout, if the VM does not reach the hibernate state, the VM shutdown is forced. This check is performed to avoid the shutdown procedure of remaining in attendance of a blocked VM. In step A, the list of the active VMs that have hibernated is saved on file; this list is used during the restore procedure to start only the active VMs before the cluster shutdown. In the next step, active VMs that expose a service (e.g., e-mail server and web server) are hibernated. For this step, the considerations performed about the SSH commands, the timeout, and the list of the active VMs are still valid. Step C turns off the virtualization system by stopping the hypervisor; this step uses the hypervisor-shutdown command via the hypervisor API. Step D turns off the cluster nodes; the power-off command is generally performed via SSH. Alternatively, it is possible to start the shutdown procedure of a node using the IPMI power-off command. Finally, step E shuts down other servers the virtualization cluster uses, such as the storage server, authentication server or firewall. In general, storage and network servers are power-off in the last step to guarantee the storage service for VM hibernation. Step F is about the UPS manager shutting down. This last step is only performed if the UPS battery is entirely exhausted.

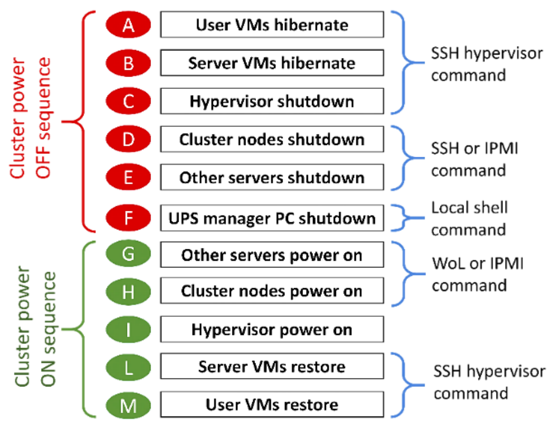


Figure 1. Shutdown and restore procedures proposed

The second part of Figure 1 shows the virtualization cluster restore procedure. This procedure is composed of 5 steps complementary to the cluster shutdown procedure. In particular, step G turns on all the servers previously turned off in step E of the shutdown procedure. A remote switch-on can be performed using WoL or the IPMI protocol, as discussed in section 2.3. Between two consecutive power-on commands, one second of delay is introduced to avoid excessive inrush current [32] caused by simultaneous turning on many servers. Inrush current is the maximal instantaneous input current drawn by an electrical device during the turned-on phase. An excessive inrush current can cause the activation of the magneto thermic switch located downstream of the UPS. In step H, the virtualization cluster nodes are turned on; also, for step H, it is possible to use different protocols for performing the server's power-on, as previously discussed in step G. Step I is typically automatically performed; in general, the hypervisor is automatically started as a service when the cluster nodes are starting. In addition, the step I can be used for performing some integrity tests. Finally, the last two steps (L and M) restore the VM image complementary to steps B and A of the shutdown procedure. The VMs list to restore is read from a file, as described in steps A and B of the shutdown procedure. VMs restore is performed using the hypervisor APIs via SSH commands.

The UPS manager performs switching off/on procedures proposed in Figure 1 using the management algorithm proposed in Figure 2. The proposed algorithm monitors the UPS status and battery charge level. The proposed approach is based on identifying three battery charge thresholds (T1, T2, and T3). In Figure 2, each flow chart element is labelled (from 1 to 12) for easy identification. It is possible to assume that the UPS is in the normal state no 1, i.e., the power grid working and the backup battery is fully charged. The UPS manager periodically checks the UPS power source (states 2 and 3) via the SNMP protocol. UPS manager performs automatic polling of the UPS status every second to not generate excessive network traffic. Moreover, the polling helps verify that the UPS manager reaches the UPS periodically. For reliability reasons, this choice is preferable to using an SNMP trap. SNMP trap sends a packet to the UPS manager during an event, such as a blackout. However, the periodic polling strategy performs a continuous network check about the ability to reach the UPS to the UPS manager via the data network. Furthermore, at each polling, the status of the UPS is checked, and any UPS error status codes are read (such as sensors or battery errors, about the battery temperature, or about maintenance or by-pass UPS's status).

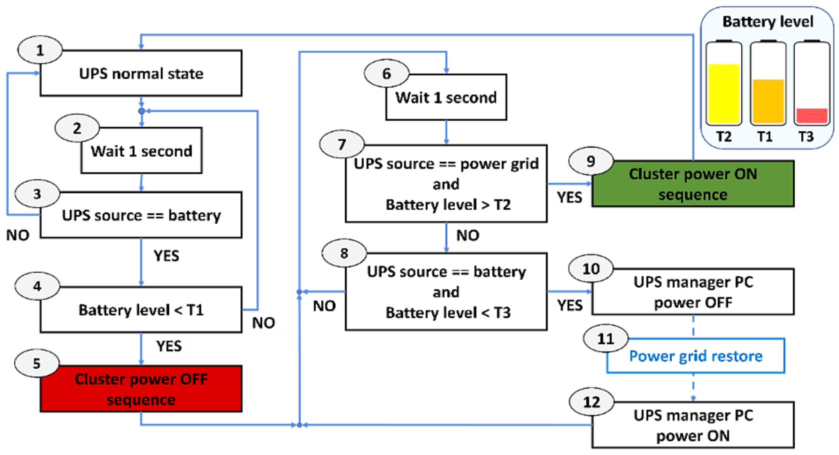


Figure 2. Management algorithm proposed

During a blackout, the UPS intervenes automatically, selecting the battery as the power source, as described in section 2.1. Until the power grid is not restored, the battery's charge is used to maintain the cluster active. The UPS manager checks the battery charge level no 4; if the battery charge is higher than T1, the UPS manager waits for the power grid to be restored; instead, if the battery charge is lower than T1, the cluster shutdown procedure described in Figure 1 (steps A to E) is executed no 5. The T1 threshold is chosen considering the time necessary to shut down the whole cluster; this data can be easily estimated by timing a cluster shutdown. About 10-15 minutes are needed to complete the virtualization cluster power-off. Once the shutdown procedure has been completed, the UPS manager waits for the power grid to be restored (states 6 and 7). When the power grid is restored, the UPS manager starts the cluster switch-on procedure no 9 only if the UPS battery level exceeds the T2 threshold. Figure 1 (steps G to M) discusses the cluster switch-on procedure. The UPS automatically recharges its battery when the power grid is restored, as described in section 2.1. The T2 threshold must be greater than the T1 threshold; generally, the T2 value is double that of

T1 (e.g., 60% - 80%). The T2 threshold is identified to guarantee sufficient UPS autonomy for performing the restore and shutdown procedures in the event of a new subsequent blackout. If the power grid is not restored, the UPS continues to power only the UPS manager. In this configuration, the UPS powers a significantly lower load. Finally, if the battery level reaches the T3 threshold no 8, it is necessary to turn off the UPS manager no 10. UPS manager power-off is performed with step F of the cluster shutdown procedure. The T3 threshold can be chosen by measuring the average shutdown time of a PC with Linux operating system. The T3 threshold must be less than T1. A typical T3 value is around 10% of the battery charge. Following power grid restoration, no 11, the UPS powers the UPS manager again. UPS manager automatically starts no 12 by exploiting the "AC auto power recovery" option enabled from the BIOS. UPS state is again monitored by the UPS manager, that performs the switch-on procedure when the battery charge reaches the T2 threshold. Interestingly, the UPS does not supply power to the UPS manager until the battery reaches at least 10% of the charge (this value can be set from the UPS control panel). This configuration allows a UPS manager shutdown in the event of an immediate new blackout.

3.2. Orchestration manager architecture proposed

This last subsection discusses the orchestration manager architecture proposed. The manager algorithm discussed in Figure 2 is developed in Python with the software architecture shown in Figure 3. The orchestration manager algorithm interfaces to UPS through puresnmp Python library; this library allows performing queries about the UPS status, as described in section 2.3. The proposed power-off/on procedures are implemented in Ansible and WoL. Ansible is open-source software that automates configuration and management procedures on Unix-like and Windows systems. Ansible scripts are organized in playbooks that collect different tasks. Each task identifies an operation performed on the target server using the SSH protocol. Ansible allows running the same task in parallel on multiple target servers simultaneously.

The orchestration manager executes the playbooks using the ansible_runner Python library. Additionally, a local log is implemented using the Python logging library. This library allows tracking of the exciting events that occur during the orchestration manager execution in a text file. Furthermore, it is possible to send the most relevant events via e-mail by exploiting the smtplib Python library to inform systems engineers quickly. Finally, an SQLite database periodically stores the main electrical parameters of the UPS; in particular, the power consumption, the output voltage, the input voltage, the backup battery voltage, and the battery temperature are stored in the database. This information helps analyze the cluster power consumption and grid quality.

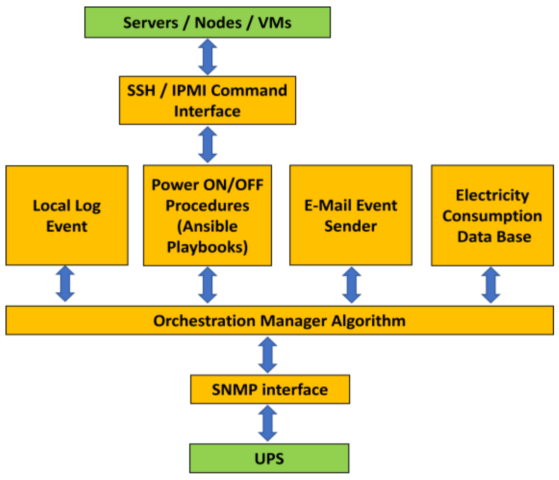

Servers / Nodes / VMs
SSH / IPMI Command Interface
Local Log Event
Power ON/OFF Procedures (Ansible Playbooks)
E-Mail Event Sender
Electricity Consumption Data Base
Orchestration Manager Algorithm
SNMP interface
UPS

Figure 3. Orchestration manager architecture proposed

4. RESULTS AND DISCUSSION

This last section discusses the results obtained. In particular, the first subsection shows the case study considered. Besides, the second subsection discusses the experimental results obtained.

4.1. Case study

This section briefly describes the virtualization cluster architecture. The cluster comprises six nodes; each node is equipped with two 28 cores Intel Xeon CPUs and 256 GB of RAM. In addition, each node is equipped with 20 HDDs of 2 TB, each configured in redundant array of independent disks - 6 (RAID-6). In addition, there are two storage servers equipped with 12 HDDs of 4 TB, each configured in RAID-6. The network comprises a gigabit switch and a 10-gigabit switch; this last one creates a fast network dedicated to synchronizing the storage between the different nodes of the cluster. Finally, there is a network firewall and an authentication server. All servers have a dual redundant power supply unit (PSU). The 6 cluster nodes data synchronization is entrusted to GlusterFS. In general, the virtualization cluster hosts 50 VMs dedicated to different tasks (remote desktop PC with Windows or Linux, Apache server, mail server, git server, computation of physical simulations and training of neural networks, and other tasks); however, during a few clusters' intensive sessions, up to 140 VMs were used at the same time. The cluster has a power consumption of about 3kW with non-intensive use and about 5kW during intensive use.

The cluster is equipped with two UPSs configured in parallel [33]; each UPS has a two lead-acid battery of 10 KVA with a battery capacity of 150 Ah [33]. The two UPSs guarantee an autonomy of about 60 minutes to the whole cluster during a blackout. In addition, an auxiliary UPS is used to power the PC devoted to running the UPS manager. This latest PC is a Zotac ZBOX AD02 equipped with AMD Dual-Core Processor E-350, 8GB of RAM, and a 128GB solid state drive (SSD). The auxiliary UPS has a lead-acid battery of 1200 VA that guarantees about 60 minutes of autonomy exclusively to the UPS manager PC. In the case study, oVirt is the cluster hypervisor used; oVirt has different APIs accessible via Ansible collection library, Python-SDK library, or SSH commands.

4.2. Experimental results

Aiming to assess the effectiveness of what has been presented, the proposed method was validated on the case study discussed in section 4.1. As discussed in section 3, the thresholds were identified empirically by measuring the cluster shutdown time in the worst case (i.e., with 140 VMs running concurrently). In particular, an average of 10 minutes is needed to perform a complete virtualization cluster shutdown. Moreover, at most 1 minute is needed to turn off the UPS manager's PC. The thresholds T1, T2, and T3 are chosen, considering an additional margin of 50%. In particular, the T1 threshold equals 25% of the battery charge level, guaranteeing 40 minutes of autonomy before starting the proposed shutdown procedure. The T2 threshold is 60% of the UPS battery level, as described in section 3.1. Finally, T3 is set at 10% of the UPS battery level, as discussed in section 3.1. Overall, the two UPS batteries guarantee an autonomy of about 60 minutes to the cluster at intensive use. Therefore, considering about 15 minutes necessary for the complete cluster shutdown, the UPS manager waits for the power grid to be restored for about 40 minutes before starting the proposed shutdown procedure. Moreover, the UPS manager has a reaction time of approximately 1200 ms from the blackout to its notification by e-mail. Finally, about 14 minutes are needed for the proposed cluster power-on procedure.

In about 12 months, the proposed approach has intervened 7 times following a power grid blackout. In 4 cases, the power grid was restored after a few minutes, and the UPS manager did not perform the power-off/on procedures proposed. The other 3 times, during extended blackouts, the proposed approach successfully shut down the cluster safely; moreover, the UPS manager resumed the cluster preserving the VMs and storage disks' health and integrity.

Figure 4 shows the cluster power consumption considering the number of active VMs. In the absence of active VMs, the cluster has a consumption of about 1kW; this consumption is associated with the cluster's hypervisor activities, the additional server activities (storage server and authentication server), the tasks performed by the operating system executed on each server, and the network devices. The cluster power consumption increases with the number of active VM, as shown in Figure 4.

However, once the cluster resources are fully used, there is no power consumption increase. When cluster resources are fully used, the VMs in execution suffer from latency; this latency is due to the contest switch of the cluster resources managed by the hypervisor. When the cluster resources are fully used, they are exploited at total capacity and consumed at the maximum of their possibilities. The graph of Figure 3 was obtained by observing the power consumption recorded by the orchestration manager in 2 months, as discussed in section 3.2.

Finally, Figure 5 shows the power consumption during the shutdown in Figure 5(a) and restoration in Figure 5(b) of the virtualization cluster. The graphs show consumption over time for each step of the shutdown and restore procedures proposed in section 3.1. (from A to M); the time interval of each step is also reported in Figure 5(a) and Figure 5(b). User VMs are hibernated in step A, and server VMs are hibernated in step B. At the end of step B, all the VMs are turned off, with a drastic reduction in cluster power consumption. However, turning off the hypervisor (step C) and the nodes (step D) further increases the CPU workload with a new temporary increase in the cluster power consumption. Step E concludes the cluster shutdown procedure by also shutting down the additional servers. Similar considerations can be performed during the cluster restore sequence proposed. Each switch-on step introduces a local peak of consumption due to the CPU and HDD node increase activity; about 600 W after one minute in step G (power on of the auxiliary servers), about 1 kW during step H (power on of the nodes), about 1.2 kW at the minute 8 during step I (hypervisor power up), 2450 W and 3200 W during steps L and M related to user VMs and server VMs power up. Step F, relating to UPS manager shutdown, is not considered in our implementation because the UPS manager is powered by an auxiliary UPS, as described in section 4.1.

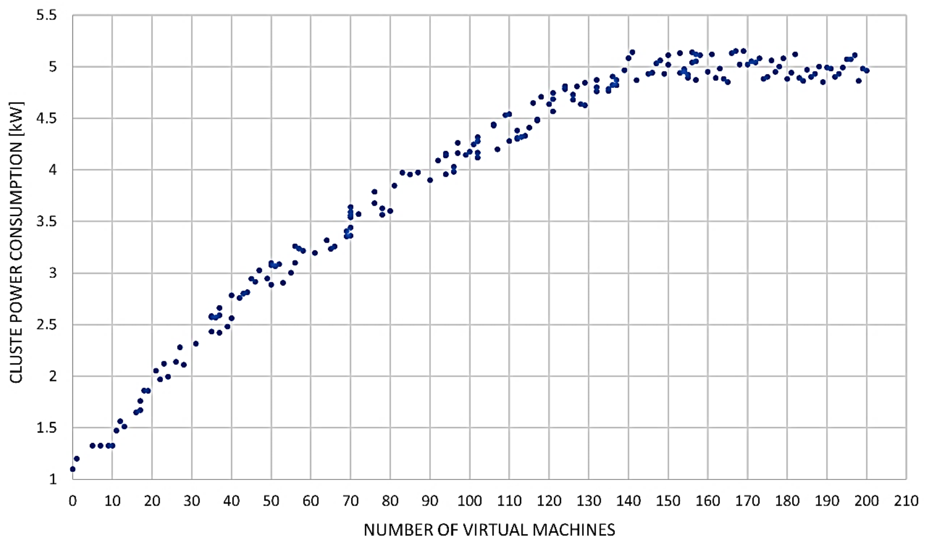
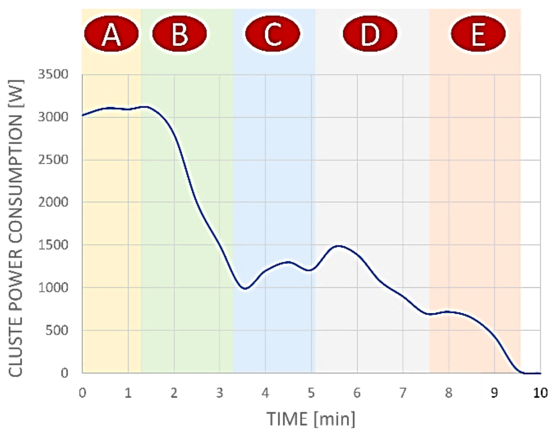
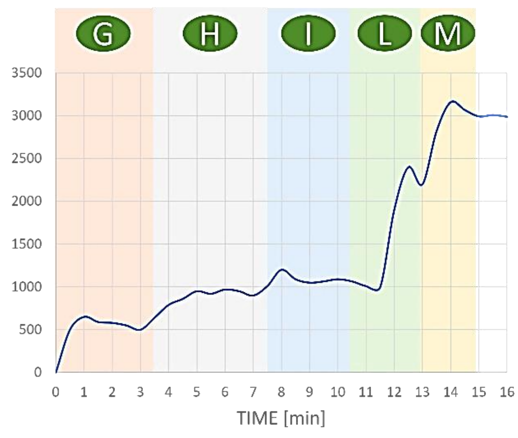


Figure 4. Cluster power consumption



(a)



(b)

Figure 5. Cluster power off/on consumption: (a) cluster power off sequence and (b) cluster power on sequence

5. CONCLUSION

This paper proposes a systematic methodology for a virtualization cluster's power-off and power-on orchestration. The proposed procedure aims to avoid servers breaking due to a sudden power blackout. The approach proposed uses a PC dedicated to executing a UPS management algorithm. In a blackout, the proposed algorithm performs a controlled shutdown procedure of the virtualization cluster. Then, when the power grid is restored, the proposed algorithm performs a controlled power-on procedure for the whole cluster. The proposed approach aims to save the state of VMs executed by the virtualization cluster and restore them after an electrical blackout. This feature is handy for long computational simulation applications; it allows to freeze of the simulation's state by VM hibernation and then resumes it after the blackout. This feature is helpful for all VMs executed on a virtualization cluster, regardless of the VM service exposed. The proposed approach does not require commercial software or vendors' proprietary software; moreover, the proposed approach exploits the main remote management protocols to turn off/on servers. Furthermore, UPS monitoring is performed using the networked device's remote management protocols. However, the proposed approach requires a low-power consumption PC to perform the proposed management algorithm. The UPS manager cannot be managed and hosted by a VM or a general-purpose server; the UPS manager must be up when the cluster is completely off. The proposed approach uses widely adopted standards and open-source code and it is intended to be as general as possible.

Finally, the proposed approach differs from the main commercial solutions available. In particular, the main commercial solutions only work for some virtualization hypervisors using agents available only for some commercial solutions. The proposed approach is agentless and compatible with UPS from various manufacturers equipped with network interfaces. Furthermore, the main commercial solutions do not specify a precise shutdown sequence for VMs, nodes, and other servers; more generally, commercial solutions do not allow specifying a complex orchestration of different physical servers and VMs. Finally, main commercial solutions do not consider a multilevel VMs application cluster scenario composed of user VMs and server VMs. The proposed approach has been evaluated and used on a real virtualization cluster for the Politecnico di Torino's teaching, research, and secretarial activities. The proposed approach is currently used on the virtualization cluster discussed in this paper.

REFERENCES

[1] F. Abidi and V. Singh, “Cloud servers vs. dedicated servers — A survey,” in *2013 IEEE International Conference in MOOC, Innovation and Technology in Education (MITE)*, Dec. 2013, pp. 1–5. doi: 10.1109/MITE.2013.6756294.

[2] V. Mach, M. Adamek, J. Sevcik, J. Valouch, and K. Barcova, “Design of an internet of things based real-time monitoring system for retired patients,” *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 3, pp. 1648–1657, Jun. 2021, doi: 10.11591/eei.v10i3.2699.

[3] S. Narasimha and S. R. Salkuti, “Design and development of smart emergency light,” *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 18, no. 1, pp. 358–364, Feb. 2020, doi: 10.12928/telkomnika.v18i1.13934.

[4] J. Hill and R. Baggs, “Software safety risk in legacy safety-critical computer systems,” in *IEEE Southeast Conference 2007 - IEEE Region 3 Technical Professional and Student Conference*, 2007. [Online]. Available: https://ntrs.nasa.gov/citations/20130011354

[5] M. F. Bari *et al.*, “Data center network virtualization: a survey,” *IEEE Communications Surveys & Tutorials*, vol. 15, no. 2, pp. 909–928, 2013, doi: 10.1109/SURV.2012.090512.00043.

[6] M. Rahmat, S. Jovanovic, and K. Lo, “Reliability estimation of uninterruptible power supply systems: Boolean truth table method,” in *INTELEC 06 - Twenty-Eighth International Telecommunications Energy Conference*, Sep. 2006, pp. 1–6. doi: 10.1109/INTLEC.2006.251636.

[7] Y. Fang, H. Wang, H. Li, C. Hu, and D. Xu, “Fault reconfiguration of super uninterruptible power supply,” in *2016 IEEE 8th International Power Electronics and Motion Control Conference (IPEMC-ECCE Asia)*, May 2016, pp. 3586–3590. doi: 10.1109/IPEMC.2016.7512870.

[8] M. Aamir, K. A. Kalwar, and S. Mekhilef, “Review: uninterruptible power supply (UPS) system,” *Renewable and Sustainable Energy Reviews*, vol. 58, pp. 1395–1410, May 2016, doi: 10.1016/j.rser.2015.12.335.

[9] Y. Kuroda, A. Akai, T. Kato, and Y. Kudo, “High-efficiency power supply system for server machines in data center,” in *2013 International Conference on High Performance Computing & Simulation (HPCS)*, Jul. 2013, pp. 172–177. doi: 10.1109/HPCSim.2013.6641410.

[10] Cisco, “Validate APC PowerChute network shutdown integration with Cisco HyperFlex edge cluster,” *Cisco Systems, Inc.*, 2020. [Online]. Available: https://www.cisco.com/c/en/us/products/collateral/hyperconverged-infrastructure/hyperflex-hx-series/deploy-powerchute-hyperflex-edge-wp.html

[11] S. Tavitas, A. McDonald, and B. Gruetzmacher, “VMware vCenter Shutdown Scenarios using Dell UPS Management Software,” 2012. [Online]. Available: https://docplayer.net/47445341-Vmware-vcenter-shutdown-scenarios-using-dell-ups-management-software.html

[12] Schneider, “PowerChute^{TM} Business Edition v10.0.5 Agent SFPCBE1005.” 2024. [Online]. Available: https://www.se.com/us/en/download/document/SPD_CCON-AT5KYC_EN/

[13] S. Tavitas, A. McDonald, and B. Gruetzmacher, “Dell UPS management software: common software installation scenarios,” 2012. [Online]. Available: https://dl.dell.com/manuals/all-products/esuprt_ser_stor_net/esuprt_rack_infrastructure/dell-online-rack-ups-3750r_reference%20guide4_en-us.pdf

[14] R. Kroll, A. Quette, and A. de Korte, “Network UPS tools user manual,” 2023. [Online]. Available: https://networkupstools.org/historic/v2.8.0/docs/user-manual.pdf

[15] R. Kroll, A. Quette, C. Lepple, P. Selinger, J. Klimov and NUT project community contributors, “Network UPS Tools Developer Guide,” 2024. [Online]. Available: https://networkupstools.org/docs/developer-guide.pdf

[16] G. Ye, F. Gao, and J. Fang, “A mission-driven two-step virtual machine commitment for energy saving of modern data centers through UPS and server coordinated optimizations,” *Applied Energy*, vol. 322, p. 119467, Sep. 2022, doi: 10.1016/j.apenergy.2022.119467.

[17] F. AL-Hazemi *et al.*, “Dynamic allocation of power delivery paths in consolidated data centers based on adaptive UPS switching,” *Computer Networks*, vol. 144, pp. 254–270, Oct. 2018, doi: 10.1016/j.comnet.2018.08.004.

[18] Q. Zhang and W. Shi, “Energy-efficient workload placement in enterprise datacenters,” *Computer*, vol. 49, no. 2, pp. 46–52, Feb. 2016, doi: 10.1109/MC.2016.58.

[19] G. Ye and F. Gao, “Joint workload scheduling method in geo-distributed data centers considering UPS loss,” in *2021 IEEE/IAS Industrial and Commercial Power System Asia (I&CPS Asia)*, Jul. 2021, pp. 50–57. doi: 10.1109/ICPSAsia52756.2021.9621553.

[20] K. S. Muhammad, R. Baharom, S. Z. M. N, and W. N. W. A. Munim, “Comparative performance analysis of bridgeless boost and bridgeless buck converter for UPS application,” *International Journal of Power Electronics and Drive Systems (IJPEDS)*, vol. 11, no. 2, pp. 801–809, Jun. 2020, doi: 10.11591/ijpeds.v11.i2.pp801-809.

[21] D. Jielin, *Network protocol handbook*. 4th Edition. Javvin Press. Morangis, France. ISBN: 1602670021. 2007.

[22] D. Piumatti, E. Sanchez, P. Bernardi, R. Martorana, and M. A. Pernice, “An efficient strategy for the development of software test

libraries for an automotive microcontroller family," *Microelectronics Reliability*, vol. 115, p. 113962, Dec. 2020, doi: 10.1016/j.microrel.2020.113962.
[23] Red Hat Ansible, "The benefits of agentless architecture," *Red Hat, Inc*, 2018. [Online]. Available: https://www.ansible.com/hubfs/pdfs/Benefits-of-Agentless-WhitePaper.pdf
[24] J. Geerling, *Ansible for DevOps: Server and configuration management for humans*. Midwestern Mac. Dallas, TX, U.S.A. ISBN: 0986393428. 2020.
[25] X. Peng and X. Qin, "Energy efficient data centers powered by on-site renewable energy and UPS devices," in *2020 11th International Green and Sustainable Computing Workshops (IGSC)*, Oct. 2020, pp. 1–3. doi: 10.1109/IGSC51522.2020.9291205.
[26] F. Wang, Q. Wang, Z. Fan, H. Wang, W. Cao, and M. Han, "A coordinated control scheme to realize uninterruptible power supply for AC-side system in AC/DC hybrid distribution grid," in *2019 IEEE 8th International Conference on Advanced Power System Automation and Protection (APAP)*, Oct. 2019, pp. 517–520. doi: 10.1109/APAP47170.2019.9225088.
[27] C. Li, Y. Hu, J. Gu, J. Yuan, and T. Li, "Oasis: scaling out datacenter sustainably and economically," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 7, pp. 1960–1973, Jul. 2017, doi: 10.1109/TPDS.2016.2615625.
[28] Legrand, "Uninterruptible power supply technical guide," 2012. [Online]. Available: https://ecataleg.be/documize/2013/5/20130507_82870_1.pdf
[29] M. S. M. Rawi, R. Baharom, and N. H. Abdullah, "Simulation model of unity power factor uninterruptible power supply topology using single-phase matrix converter," *International Journal of Power Electronics and Drive Systems (IJPEDS)*, vol. 13, no. 2, pp. 969–979, Jun. 2022, doi: 10.11591/ijpeds.v13.i2.pp969-979.
[30] M. Zadehbagheri, M. J. Kiani, T. Sutikno, and R. A. Moghadam, "Design of a new backstepping controller for control of microgrid sources inverter," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 12, no. 4, pp. 4469–4482, Aug. 2022, doi: 10.11591/ijece.v12i4.pp4469-4482.
[31] S. D. Lowe and D. M. Davis, *Hyperconverged infrastucture implementation strategies*. 2015. [Online]. Available: https://www.actualtechmedia.com/wp-content/uploads/2017/12/Hyperconverged-Infrastructure-Implementation-Strategies-eBook.pdf
[32] Y. Wang, S. G. Abdulsalam, and W. Xu, "Analytical formula to estimate the maximum inrush current," *IEEE Transactions on Power Delivery*, vol. 23, no. 2, pp. 1266–1268, Apr. 2008, doi: 10.1109/TPWRD.2008.919153.
[33] Braga Moro, "Orion plus 6/10KVA - User manual," 2014. [Online]. Available: https://www.bragamoro.com/uploads/products/orion_plus_manuale_utente.pdf

BIOGRAPHIES OF AUTHORS

Davide Piumatti received the M.Sc. degree in computer engineering from Politecnico di Torino in 2015 and the Ph.D. in computer and systems engineering from the Politecnico di Torino in 2021 at the Department of Control and Computer Engineering. From 2015, he worked as a research fellow on the test of highly reliable microcontroller systems at Politecnico di Torino. His research interests include the test of analog, power, and digital systems for safety-critical applications and managing servers and network equipment. Currently, he works in the technical staff of the research and teaching laboratory of the Control and Computer Engineering department at Politecnico di Torino, since 2021. He can be contacted at email: davide.piumatti@polito.it.

Andrea Galletto was born in 1977 and received the bachelor's degree in computer science engineering in 2018. Since 2001, he has worked as a system and network administrator at the Department of Control and Computer Engineering of Politecnico di Torino. His main activities include configuring and administrating server and client systems in heterogeneous Linux/Windows environments in the Department's Computer Laboratories, especially for didactic purposes. In recent years, he has been particularly interested in open-source solutions in Linux environments in the areas of network security, virtualization clusters, parallel storage, and HPC clusters for research. He can be contacted at email: andrea.galletto@polito.it.

Flavio Castagno was born in 1963. He received the bachelor's degree in computer engineering in 2016; since 1994, he is employed at Politecnico di Torino, and since 2000 he has worked as a computer technician at the Department of Control and Computer Engineering. In these years, his main activities were database administration, technical support to administrative employees, configuration and administration of Windows/Linux computers and servers, virtual machines and virtualization clusters management, and technical support for hardware and software purchases (tender specifications). He can be contacted at email: flavio.castagno@polito.it.