applied sciences

MDPI

Article

The Design and Implementation of a Novel Open Source Massive Deployment System

Steven J. H. Shiau [1,2,*], Chen-Kai Sun [2], Yu-Chin Tsai [2], Jer-Nan Juang [1] and Chi-Yo Huang [3,*]

[1] Department of Engineering Science, National Cheng Kung University, No.1, University Road, Tainan City 701, Taiwan; jjuang@mail.ncku.edu.tw
[2] National Center for High-performance Computing, No. 7, R&D Rd. VI, Hsinchu 30076, Taiwan; ceasar@nchc.org.tw (C.-K.S.); thomas@nchc.org.tw (Y.-C.T.)
[3] Department of Industrial Education, National Taiwan Normal University, Taipei 106, Taiwan
* Correspondence: steven@nchc.org.tw (S.J.H.S.); cyhuang66@ntnu.edu.tw (C.-Y.H.); Tel.: +886-3-5776085 (ext. 335) (S.J.H.S.)

Received: 10 May 2018; Accepted: 8 June 2018; Published: 12 June 2018

Abstract: The hypervisor and container are emerging cloud computing and fog computing technologies, which enable rapid system deployment. However, both of the technologies depend on the operating system (OS) and applications that are installed on the host machines. System deployment is the activity to deliver and install OSs and applications onto computers. Such deployment activities are widely required in the infrastructure of cloud computing, fog computing, high-performance computing clusters, and classrooms of computer education. Albeit the concept of system deployment is not new, traditional solutions cannot support the rapid evolution of open source file systems. Furthermore, existing solutions cannot support the massive deployment of disks in a computer as well as the massive deployment in large-scale computers. To resolve the issue, the authors proposed novel system architecture as well as software that is openly available. The experiments are undertaken by deploying a Linux system to 1 to 30 Universal Serial Bus (USB) flash drives in a single machine and to 1 to 32 machines in a network using the software that is being developed in this work. The results have demonstrated the feasibility and efficiency of the proposed work. The relationships between the bus bandwidth, the writing rate of the USB flash drive, and the number of flash drives were also formulated as a govern equation. Performance evaluation and cost savings in comparing to the deployment cases adopting commercial software were also provided for demonstrating the performance enhancement and cost reduction by using the novel deployment system. In general, the proposed architecture and the developed software are highly effective from the aspects of both performance and cost.

Keywords: massive deployment; system deployment; open source; file system imaging; Infrastructure as a Service (IaaS)

1. Introduction

System deployment or system provisioning is the activity of making the operating system (OS) and applications that are available for use in the machine being deployed (i.e., the machine is put into the state of operational readiness) [1,2]. Because system deployment can be used for either physical or virtual machines, the term "bare-metal provisioning" [3] can be used to describe the tasks that are related to the deployment of OSs and applications to physical machines. However, such deployment tasks are time-consuming because many sub-tasks are involved, including OS deployment, application installation, setting configuration, and performance adjustment, as well as the preparation and restoration of images [2]. In addition, various OSs, including Microsoft (MS) Windows, Linux,

Macintosh Operating System (MacOS), and FreeBSD (Berkeley Software Distribution), can be deployed. Hence, system deployment can be very complicated and tedious. However, system deployment is quite dominant due to its nature in ensuring the operational readiness of computer systems. Thus, system deployment can be regarded as the foundation of various computing environments.

The applications of system deployment vary. Infrastructure as a Service (IaaS) [4] in cloud computing is one of the most dominant applications. Other typical cases include computing tasks accessing hardware devices that cannot be virtualized, while dedicated hardware is required for performance or security reasons. In addition, many applications can be found in high-performance computing clusters, content delivery networks, render farms, utility computing, large-scale web services, databases and file systems, appliances and network devices, and computer classrooms. IaaS has three types of architecture [5]: bare metal, hypervisor, and container. Bare metal servers provide services on non-virtualized servers, so there is no overhead, as in the case for hypervisor servers. The overhead that is associated with a container-based virtual server is smaller than that associated with a hypervisor virtual server. Regardless of the IaaS architecture, the OS and basic applications on the host machine must be available. In addition, the emerging fog computing also relies on the hypervisor and container technologies [6]. In a nutshell, without OSs and basic applications on the physical host machines, none of the mentioned scenarios of computing, including cloud computing and fog computing, can provide services. However, the installations of the OSs as well as applications are very time consuming. Therefore, the efficient deployment of OSs and applications to physical machines with all types of architectures is critical.

Open source means that the source code of the software is publicly available, thereby allowing for modification. The copyright owner of open source software allows for others to use, copy, modify, and distribute the source code [7]. Because the open source approach allows for anyone to read and modify the source code(s), the modifications of such software can also be fed back to the developers. Therefore, all of the developers of open source software benefit each other. The model enabling developers to "stand on the shoulders of giants" facilitates the access, use, and improvement of existing software. Thus, a lot of high-quality software has been developed accordingly. Typical cases include the Linux kernel [8] and the OpenStack [9] for cloud computing platforms.

Although system deployment is dominant, to the best of the authors' knowledge, no single open source program for system deployment exists, no matter if being from the aspect of single machine backup and recovery, the massive deployment of disks in a computer, or the massive deployment to large-scale computers. Furthermore, existing commercial proprietary system deployment software lacks a flexible architecture to handle various file systems being generated during the rapid evolution process of open source file systems due to software license or marketing issues.

Therefore, this paper aims to propose an open source massive deployment system that includes system architecture as well as associated software for system deployment. This massive deployment system can handle single machine backup and recovery, the massive deployment of disks in a computer, and massive deployment in large-scale computers. In addition, this system can deploy most dominant OSs, including Linux, MacOS, MS Windows, and FreeBSD. Once verified, the proposed mass deployment system can be promoted and applied in various scenarios of computing, thereby enhancing the efficiency of system deployment. To achieve this goal, the program "Clonezilla" [10] was developed, which was based on the proposed architecture and following the open source developing model.

In order to demonstrate the feasibility of the proposed massive deployment system, the authors validated the system by deploying massive disks in a single machine. Clonezilla live was used to deploy Ubuntu Linux to a large quantity of USB flash drives in parallel on a single x86-64 based personal computer and USB 3.0 hubs. Various factors influence the input/output (I/O) throughputs (e.g., the number of disks, the bandwidth of system I/O bus, and the size of each I/O transfer). A governing equation to determine the optimal performance of the massive deployment system was also proposed and verified. In addition, cost comparisons between the proposed novel massive deployment system versus other widely adopted commercial proprietary solutions are provided. The software was also

validated by deploying a Linux system to massive computers in a computer classroom via the multicast mechanism. The experimental results demonstrate the feasibility of massively deploying disks in a computer, and the use of massive deployment in large-scale computers. Nevertheless, the well-verified mass deployment system can also be used for single machine backup and recovery.

The rest of this paper is organized as follows. Section 2 reviews and discusses related works. Section 3 introduces the system architecture, software implementation, software packaging, massive deployment, and major changes and improvements of the developed software in the past decade. The experimental results are demonstrated in Section 4. The results and comparisons are discussed in Section 5. Conclusions and suggestions for future works are provided in Section 6.

2. Related Works

The OS and applications in a physical machine are the basis for all of the computing scenarios. Therefore, many related works have been conducted concerning system deployment. At least three methods were proposed to deploy a system: (1) installing OS and applications from scratch with an automated installation and configuration program (hereafter referred to as "automated installation") [11]; (2) restoring previously saved files from a disk (hereafter referred to as "differential update") [12]; and, (3) restoring an image of the previously saved file system (hereafter referred to as "file system imaging") [13]. These three methods have specific advantages and disadvantages. The major advantage of the automated installation method is the flexibility to decide which packages will be installed. The system can also be configured in the deployment processes. The major disadvantage of this method is the comparatively longer time that is required. By using the automated installation method, every computer must be installed from scratch. The major advantage of the differential update method is the efficiency in file synchronization and the consumption of network bandwidth. However, the file system must be created before the system deployment starts so that files can be copied. Furthermore, not all of the file systems can be created when the OS is different. Therefore, the differential update method can support a very limited number of OSs. The major advantage of the file system imaging method is the highest efficiency of deployment. However, the method lacks flexibility because all of the systems being deployed are the same. Due to the different pros and cons that are associated with each deployment method, system administrators must choose the most appropriate one when deploying systems.

2.1. Automated Installation Method

Among the automated installation methods, Kickstart Installation [1] and Fully Automatic Installation (FAI) [14,15] are two well-known open source projects. Both use a configuration file to define the steps in the system installation. Once the installation on a given computer is ready to be done, the Kickstart or FAI program takes over and performs the automatic installation without any manual intervention. Cobbler [11] is open source software that is revised based on the Kickstart program. The major enhancement of Cobbler is the provision of a web user interface that allows for the provision of more features, so system administrators can automate many Linux-related tasks. Metal as a Service (MaaS) [16] is another open source solution that is provided by Ubuntu Linux that was designed specifically for deploying Ubuntu Linux. All of the software mentioned here can only be used to deploy the Linux system. The deployment of other OSs (e.g., MS Windows, MacOS, or FreeBSD) requires other solutions. One typical example is open source software called Razor [3], which can deploy both Linux and MS Windows systems. Another open source program is Crowbar [17], which is supported by Dell Inc. and it also provides the ability to deploy both Linux and MS Windows systems.

2.2. Diffenrential Update Method

Regarding the previously mentioned system deployment software adopting differential update technology, "rsync" is the major open source software that provides fast incremental file transfer and a synchronization mechanism [18]. However, more procedures than just copying files are required in

system deployment. Such procedures include the creation of partition tables and file system(s) and interactions with the boot loader. However, because "rsync" lacks all of the functions that are required to deploy an entire system onto a disk, some other tools have been developed to cross the gap based on the differential update method. Mondo Rescue [19] is open source software that can deploy Linux and FreeBSD systems. Mondo Rescue has been used by thousands of users and companies. Another open source software, called ReaR [20], can also deploy the Linux system and has been expanded as Disaster Recovery Linux Manager (DRLM), which can support massive deployment [21]. Storix System Backup Administrator (SBAdmin) [22] is a commercial proprietary software with good graphical user interface (GUI), Web, and command-line interfaces. Therefore, system administrators can choose the interface that they prefer when conducting system deployment. Because SBAdmin can deploy the Linux system only, the application of SBAdmin is constrained.

2.3. File System Imaging Method

Many solutions are available for the file system imaging method. The Ghost [23], which was commercialized in 1985, is one of the most famous examples of commercial proprietary software focusing on MS DOS (Disk Operating System) and Windows deployment. The enterprise version of Ghost supports the multicast function. Therefore, Ghost is suitable for massive deployment. True Image [24] is another commercial proprietary software provided by Acronis. Two types of deployment methods are available in True Image: online/offline files backup and file system imaging. In addition, in previous years, many open source programs have been developed to fulfill the function of system deployment using the file system imaging method. The Partimage [25], which was developed in 2001, supports the Linux and MS Windows system deployment. The development of Partimage was terminated in 2008 due to the development of another open source system deployment software, the FSArchiver [26], by the same author. FSArchiver provides some new features that Partimage lacks. Redo Backup and Recovery [27] is open source file system imaging software with a GUI; the software can be executed from a bootable Linux CD or a USB flash drive. The open source OpenGnSys (Open Genesis) [13] system provides the functions that are required to deploy, manage, and maintain computers. OpenGnSys has been used in the computer lab in the Department of Electronic Engineering, Computer Systems, and Automatics at the University of Huelva (UHU) in Spain to improve teaching and learning in the classroom. FOG [28] is an open source network computer cloning and management solution that provides a web interface. Microsoft also provides proprietary technology, Windows Deployment Services (WDS) [29], for the automated installation of Windows operating systems. The WDS is especially suitable for Windows Vista, Windows Server 2008, and later versions of Windows. With WDS, system administrators can deploy Windows systems to new computers using network-based installations. Apple Software Restore (ASR) [30] is the system image software that is provided by Apple Computer. ASR can be used for the massive deployment of Macintosh computers using two deployment methods: file-based and block-based modes. Normally, the block-based mode is faster due to the transactions that are not going through the OS file system. Frisbee [31], or Emulab [32,33], is an open source system that was developed at the University of Utah. Frisbee aims to save and deploy entire disk images and has been used in the Utah Emulab testbed. Frisbee manages thousands of disk images for users. OpenStack [9] is open source software that is dedicated for the IaaS in cloud computing. OpenStack supports bare-metal, virtual machine (VM), and container-based hosts [34]. The integrated program in OpenStack for bare-metal machines is OpenStack Ironic [9,35], and it provides bare-metal provisioning for the Linux system.

2.4. Other Related Works

Some tools and mechanisms have been widely adopted in network-based massive deployment, including Preboot Execution Environment (PXE), Internet Protocol (IP), User Datagram Protocol (UDP), Dynamic Host Configuration Protocol (DHCP), and Trivial File Transfer Protocol (TFTP) [36]. The PXE environment actually combines the UDP/IP, DHCP, and TFTP protocols. Because the image of the

PXE firmware is quite small, it can be implemented on a variety of systems, including embedded systems, single-board computers (SBC), PCs and high-end servers. With the PXE environment enabled, bare-metal computers can be booted from network interface cards and initiate system deployment. UDPcast [37] is an open source file transfer software that can transfer data simultaneously to many destinations in either the multicast or broadcast mechanism on a local area network (LAN). The multicast or broadcast method makes the file transfer in UDPcast more efficient than other methods, such as Network File System (NFS) and File Transfer Protocol (FTP). Therefore, UDPcast can be one of the best tools for network-based massive deployment.

Scholars or researchers have also dedicated many efforts to develop deployment software. The Silent Unattended Installation Package Manager (SUIPM) [38] is a method that enables the automatic software installation and uninstallation processes, and generates silent unattended installation/uninstallation packages. Quiet [39] is a framework for automatic installation and uninstallation over a network that is based on the SUIPM. Normally, SUIPM does not cover the OS installation; instead, the SUIPM focuses on software/application packaging and installation. Therefore, SUIPM is different from the open source massive deployment that is proposed in this work due to the lack of deployment for OS. Although the open source programs that are mentioned here aim to deploy systems, the focus is different. For example, FOG focuses on the system deployment using the server–client mechanism. Redo Backup and Recovery targets a single machine backup and restore, while OpenGnSys is dedicated to the management of massive computers in computer laboratories or classrooms.

The open source deployment system that is proposed in this paper is distinguished from the other software systems based on the following features: (1) one deployment system works for system backup and recovery, the massive deployment of disks in a computer, and the massive deployment to large-scale computers; (2) open design, with an easiness to change or add functions; (3) support for deployment of a wide variety of OSs, including Linux, MS Windows, MacOS, and FreeBSD; (4) fully automated, meaning that with proper pre-settings human interaction or intervention is not required; (5) a customization option enabling system administrators to assign different settings by boot parameters; and, (6) the provision of a live system that is ready to use (i.e., no installation required).

The major contribution of this research is that it proposes novel system architecture for system deployment. To accomplish this, a comprehensive program using the open source technique is developed. This architecture, and hence, the associated software can be used for three scenarios: (1) system backup and recovery; (2) massive deployment of disks in a computer; and, (3) massive deployment in large-scale computers. All three scenarios can be processed in one program system: Clonezilla live. Because Clonezilla live is a ready-to-use massive deployment system, no tedious installation or configuration is required before usage. Therefore, significant time can be saved in system deployment. The feasibility of Clonezilla live will be demonstrated in Section 4 by deploying Ubuntu Linux to massive USB flash drives on a single machine and massive machines in a computer classroom.

3. Design and Implementation

This section describes topics that are related to the design and implementation of system deployment, including software architecture, software implementation, software packaging, massive deployment, and major changes and improvements of the software in the past decade.

3.1. Software Architecture for System Deployment

The proposed system architecture consists of computer hardware, live OS, deployment system, and massive deployment mechanism (refer to Figure 1). Four modules are included in the system architecture: (1) partition table processing; (2) partition image processing; (3) pre-processing; and, (4) post-processing modules. Two sub-modules exist in the second partition image processing module: file system process module and partition image file compressing and decompressing module. The massive deployment mechanism can support either local or network connections. The former one enables the connections of massive disks to the system buses, such as the USB or the external Serial

Advanced Technology Attachment (eSATA) bus. The latter one (i.e., network connection mechanism) is for massive machines connected to network switches.

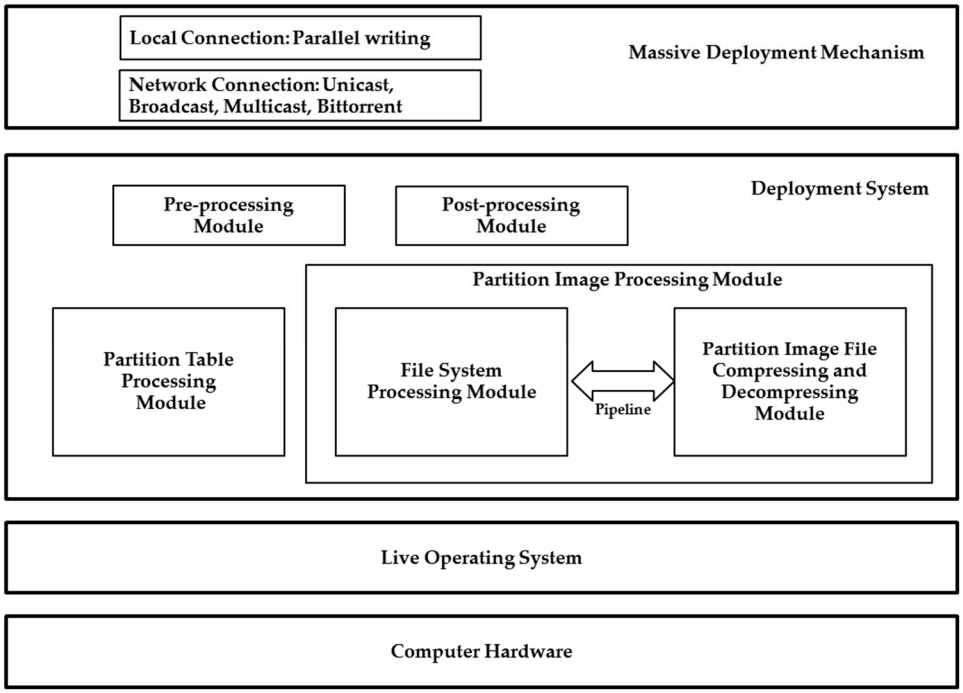

Figure 1. The proposed system architecture for system deployment.

In addition, several principles are used when developing the system deployment system based on the existing free and open source software. First, the image is a directory. All of the information about the disk as well as the partitions, boot loaders, and file systems is stored in the directory. As a result, the design provides the flexibility to add more information by putting files in the image directory in the future. Traditionally, a system deployment program normally uses a file to store the file system image. However, when a new mechanism must be added in the future, the image file format has to be redefined. Hence, future maintenance is not easy. Defining an image as a directory can greatly reduce the maintenance efforts in the future. Second, all information in the image directory should be the output or input of existing open source software, unless such a program does not exist or is in short of the required functionalities. If either one of these criteria is fulfilled, a suitable program has to be developed in order to resolve the identified issue(s). Third, the UNIX philosophy, "write programs to handle text streams, because that is a universal interface" [40], should be followed. Therefore, in most programs, pipelines are used to enable these programs to work together. This principle has a very good feature because, for example, adding a new compression format for the image is extremely easy.

3.2. Software Implementation

The key functionality of the proposed system architecture is the file system processing module. Instead of automated installation or differential update methods, the file system imaging method was chosen as the approach for system deployment because the system imaging method is more general, robust, and faster than the automated installation and differential update mechanisms [31]. The system imaging method is more general because users do not need the file system-related knowledge regarding directory structure, file ownership, etc. Instead, users can use the system imaging method-based software for system deployment. Regarding the robustness of the system imaging method-based software, the deployment does not depend on the existing contents of the destination disk because the software just reads or overwrites the whole file system. As for the speed advantage of the system

imaging method that was specified earlier, the deployment does not have to go through the OS file system for every single file. Hence, the file system imaging method that reads or writes an entire disk image is faster than the other two methods that were used to decide the files to be copied or updated. As previously mentioned, the file system imaging mechanism is not perfect due to the shortage of flexibility and the requirement for more network bandwidth. In terms of flexibility shortage, all of the deployed systems should be configured or fine-tuned after deployment. Furthermore, more network bandwidth is required than the differential update mechanism. However, the advantages outweigh the disadvantages when the file system imaging method is compared with automated installation and differential update methods. Therefore, the file system imaging mechanism was chosen in the proposed open source massive deployment system.

Based on the proposed system architecture and principles, the program Partclone, as the engine for the file system processing module, was developed, and it is licensed under GNU General Public License (GPL) [41]. The Partclone is designed to support as many file systems as possible for the major OSs. Existing open source imaging software, like Partimage or FSArchiver, cannot support some of the major file systems (e.g., the imaging Hierarchical File System Plus (HFS+) of MacOS). Partclone aims to solve this problem by leveraging the libraries of file systems to obtain the used blocks of a file system and outputting the data of the used blocks to a file. Nowadays, Partclone can support most file systems, including ext2, ext3, ext4, reiserfs, reiser4, xfs, jfs, btrfs, f2fs, nilfs2, File Allocation Table (FAT), New Technology File System (NTFS), HFS+, Unix File System (UFS), minix, Virtual Machine File System 3 (VMFS3), and VMFS5.

The Clonezilla program consisting of all the modules of the proposed massive deployment system (refer to Figure 1) is implemented in a bash shell script. The components of Clonezilla include: (1) pre-processing module: including read the input parameters, check the disk device, and execute the customized job if assigned by user; (2) partition table processing module: including deal with the partition table and the master boot record (MBR) of a disk; (3) partition image processing system: including checking the file system on a partition and save or restoring the file system on the partition using Partclone; and (4) post-processing module: including process the boot loader on the disk, tune the deployed system (e.g., remove hardware info files from the hard drive), and execute the customized job if assigned by the user.

An example is given here to show how the developed programs and the existing software work together in UNIX pipelines in the Clonezilla program. To save the image of a new technology file system (NTFS) file in the partition, /dev/sda1, the image will be compressed using the parallel gz compression program pigz in a 16-core central processing unit (CPU). Any output files that are larger than 4096 Mega Byte (MB) will be split into smaller 4069 MB pieces. Each piece is output to a file, entitled "sda1.ntfs-ptcl-img.gz" in the /home/partimag/vmware-windows-7 directory. The following command will be executed inside the main Clonezilla program:

```
partclone.ntfs -z 10485760 -N -L /var/log/partclone.log -c -s /dev/sda1 –output - | pigz -c –fast -b 1024
-p 16 –rsyncable | split -a 2 -b 4096MB - /home/partimag/vmware-windows-7/sda1.ntfs-ptcl-img.gz.
```

In this example, partclone.ntfs is the program that was developed in this research, while the programs pigz and split are the existing programs from the Linux system. The procedures for saving an image from a drive and restoring the image to another drive are shown in Figure 2a,b, respectively. Figure 2a shows the imaging engines for a different file system, which includes Partclone, Partimag, ntfsclone, and dd. Among them, Partclone is the default engine for Clonezilla. In addition, the sequential compressing programs from the Linux system, which include gzip, bzip2, lzma, lzip, or xz, could be used to compress the image. By default, gzip is chosen. If the CPU on the system has multiple cores, the corresponding parallel program, like pigz, pbzip2, plzip, or pixz, can be used when such a program is available from the Linux system. However, if the corresponding program is unavailable, then Clonezilla will roll back to use the sequential program. As an image is restored, the image should

be decompressed by the corresponding program and then written to a partition, according to the format of the image, as described in Figure 2b.

A screenshot about the image directory from Clonezilla is presented in Figure 3. One can easily identify the file(s) by reading its name. For example, sda-mbr is the MBR data of disk sda, where sda is the device name under Linux for the first hard drive in the system. Meanwhile, sda1.ext4-ptcl-img.gz.aa is the image of sda1 being saved by the imaging program Partclone. Here, sda1 is the first partition on the first drive. After being compressed by the gzip or pigz program, the image file is split into several volumes. The default limit of each volume is 4 Giga Byte (GB). Because the image size of the example is only 2.5 GB, and there is only one volume, then the suffix is aa. If more volumes exist in the image, the suffixes ab, ac, etc., will be used, respectively. The complete list of the file formats supported is provided in Table 1. The names denoted in brackets are assigned according to the name of the device (disk, partition, or Logical Volume Manager (LVM)), file system, compression format, or suffix. The other names without brackets are fixed styles.

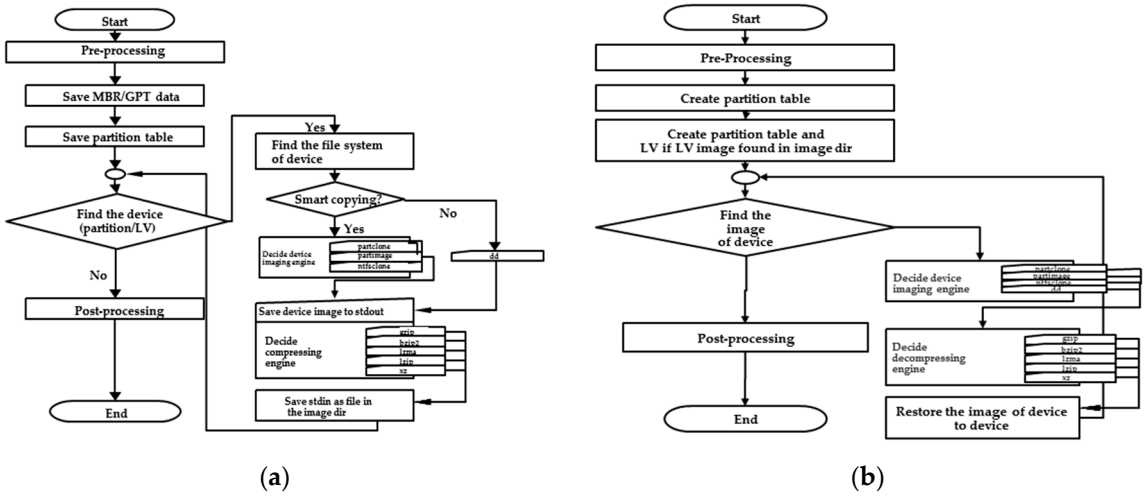

Figure 2. Flowchart for (**a**) saving and (**b**) restoring an image.

```
user@clonezilla:/home/partimag/zesty-x86-20180320$ ls -alFv
total 2.5G
drwxr-xr-x   2 root root 4.0K Mar 30 18:56 ./
drwxr-xr-x 137 root root  12K Apr  5 19:34 ../
-rw-r--r--   1 root root  96K Mar 30 18:55 Info-dmi.txt
-rw-r--r--   1 root root  187 Mar 30 18:56 Info-img-id.txt
-rw-r--r--   1 root root  58K Mar 30 18:55 Info-lshw.txt
-rw-r--r--   1 root root 4.4K Mar 30 18:55 Info-lspci.txt
-rw-r--r--   1 root root  219 Mar 30 18:55 Info-packages.txt
-rw-r--r--   1 root root   96 Mar 30 18:56 Info-saved-by-cmd.txt
-rw-r--r--   1 root root 1.1K Mar 30 18:54 blkdev.list
-rw-r--r--   1 root root  535 Mar 30 18:54 blkid.list
-rw-r--r--   1 root root 4.4K Mar 30 18:56 clonezilla-img
-rw-r--r--   1 root root  144 Mar 30 18:54 dev-fs.list
-rw-r--r--   1 root root    4 Mar 30 18:55 disk
-rw-r--r--   1 root root    5 Mar 30 18:55 parts
-rw-------   1 root root 2.5G Mar 30 18:55 sda1.ext4-ptcl-img.gz.aa
-rw-r--r--   1 root root   35 Mar 30 18:54 sda-chs.sf
-rw-r--r--   1 root root 1.0M Mar 30 18:54 sda-hidden-data-after-mbr
-rw-r--r--   1 root root  512 Mar 30 18:54 sda-mbr
-rw-r--r--   1 root root  273 Mar 30 18:54 sda-pt.parted
-rw-r--r--   1 root root  260 Mar 30 18:54 sda-pt.parted.compact
-rw-r--r--   1 root root  133 Mar 30 18:54 sda-pt.sf
```

Figure 3. Screenshot of the files in an image directory that was saved by Clonezilla.

Table 1. Files in a Clonezilla image directory.

File Name	Attribute	Description	Example
blkdev.list	plain text	Output of command "blkdev"	-
blkid.list	plain text	Output of command "blkid"	-
clonezilla-img	plain text	File to mark the directory is for Clonezilla image; it also contains the screen output when saving the image	-
disk	plain text	Contain the disk device name (e.g., sda, sdb)	-
parts	plain text	Contain the partition device name (e.g., sda1, sdb2, sdb1)	-
lvm_logv.list	plain text	List of the logical volumes and their file systems	-
lvm_vg_dev.list	plain text	List of volume groups, physical volumes, and their UUIDs	-
lvm_[VG].conf	plain text	Output of command "vgcfgbackup"	lvm_vg_f19.conf
Info-dmi.txt	plain text	Output of command dmidecode	-
Info-lspci.txt	plain text	Output of command lspci	-
Info-lshw.txt	plain text	Output of command lshw	-
Info-packages.txt	plain text	List of Clonezilla-related packages and their versions when image is saved	-
Info-saved-by-cmd.txt	plain text	Includes the command and options when image is saved	-
[partition | VG-LV].[fs]-ptcl	binary	File system of a partition, dumped by Partclone, and compressed then split into multiple volumes	sda1.ext4-ptcl-img.gz.aa
[disk]-chs.sf	plain text	The cylinder, head, and sector numbers of disk	sda-chs.sf
[disk]-hidden-data-after-m	binary	The data between the MBR and the 1st partition. Dumped by command "dd"	sda-hidden-data-after-mbr
[disk]-mbr	binary	The MBR data, dumped by command "dd"	sda-mbr
[disk]-pt.parted	plain text	The partition table. Dumped by command "parted -s /dev/[disk] print"	sda-pt.parted
[disk]-pt.parted.compact	plain text	The partition table. Dumped by command "parted -s /dev/[disk] unit compact print"	sda-pt.parted.compact
[disk]-pt.sf	plain text	Output of command "sfdisk -d"	sda-pt.sf
swappt-[partition].info	plain text	The UUID, or labe of a Linux swap partition	swappt-sda6.info
[disk]-gpt.gdisk	binary	The GPT binary data saved by gdisk	sda-gpt.gdisk
[disk]-gpt.sgdisk	plain text	The GPT basic summary data saved by sgdisk in plain text	sda-gpt.sgdisk

3.3. Software Packaging

The Clonezilla program depends on many programs, because, as described in Section 3.2, it uses the UNIX pipelines to make Partclone and the existing open source software work together. Hence, it is tedious for users to collect all of the required programs when they want to use Clonezilla. In addition, most of the system deployment is executed in the so-called bare metal way—that is, the machine may have only one hard drive without any OS or software. An extra OS is required to conduct the system deployment. Thus, the best way for system deployment is to have a live system [42] that includes all of the required programs and has the capability to boot from a machine. A live system is the complete bootable computer software, including OS, which allows for users to boot from removable media and run applications without affecting the contents of the computer. This is why Clonezilla live was designed for system deployment [10] in this research. In order to make use of the open source software, Debian live [42], which is a live system based on Debian Linux, was chosen as the underlying system because its software repository provides thousands of open source software packages. The file size of the developed Clonezilla live system is about 200 to 300 MB. In addition, a mechanism is required to predefine all of the steps required by an unattended mode in Clonezilla live. The kernel boot parameters mechanism [43] was used to predefine the steps in Clonezilla live. Additional parameters that were developed and implemented in Clonezilla live include ocs_live_prerun*, ocs_live_run, ocs_live_postrun*, and ocs_daemonon. The ocs_live_prerun* parameter provides the commands to be executed before the main program, ocs_live_run, can be executed. The ocs_live_postrun* parameter provides the commands to be executed after the main program, ocs_live_run, can be executed. Finally, ocs_daemonon predefines the services that are to be started after the system is booted. The boot parameters mentioned could be used to customize the unattended mode of a deployment system. An example of automatically configuring the network

settings by the DHCP, mounting the NFS file server as an image repository, starting the ssh service, and then running a restoring job using the boot parameters of Clonezilla live is demonstrated, as follows:

ocs_live_prerun1="dhclient -v eth0" ocs_live_prerun2="mount -t nfs 192.168.100.254:/home/partimag/home/partimag" ocs_daemonon="ssh" ocs_live_run="ocs-sr -g auto -e1 auto -e2 -batch -r -j2 -p poweroff restoredisk my-xenial-img sda"

3.4. Massive Deployment

As mentioned in Section 3.1, two types of mechanisms were considered and implemented in the massive deployment system. When dealing with the local connection (i.e., massive disks are connected to a system bus belonging to a computer), writing the data of the system image to massive disks in parallel was developed. The program in Clonezilla that can execute such a massive deployment is called ocs-restore-mdisks. According to the flowchart that is shown in Figure 2b, a system deployment is executed for each drive. The multiple executions of writing images to USB drives in parallel by using the ocs-restore-mdisks program is executed by using the "for loop" on the Linux system.

When dealing with the network connection where massive machines are connected to network switches, at least four types of communication protocols (i.e., unicast, broadcast, multicast [37], and peer-to-peer (P2P) networking [44], such as BitTorrent [45]) can be used to deploy the system image to each machine. These four protocols were implemented in Clonezilla. Basically, the flowchart for deploying each machine is similar to the one in Figure 2b. The only difference is how the client machine to be deployed receives the system image from different protocols. Hence, the steps for such a massive deployment can be defined according to Lee et al. [37]: (1) save a system image from the template machine; (2) start the required services for the client machines, including DHCP, TFTP, and the data-distributing service (unicast, broadcast, multicast, or BitTorrent); and, (3) boot the client machines from PXE, receive the system image data, and then write to the disks.

As mentioned in Section 3.3, Clonezilla live is the software that is used for system backup and recovery, the massive deployment of disks in a computer, and the massive deployment in large-scale computers. Clonezilla live contains the server programs (e.g., DHCP, TFTP, and UDPcast), as well as the programs for system deployment in client machines (e.g., sfdisk, gunzip, Partclone, and Clonezilla). Clonezilla live also provides the mechanism for clients to boot from the PXE. Therefore, Clonezilla live can be used in massive deployment on both the server machine and the client machines by appropriate configurations.

3.5. Major Changes and Improvements of the Software

Since the initial release of Clonezilla live, the software has been revised several times to enhance its functionalities. Table 2 summarizes these major revisions, indicating the version number, revision time (year), and major changes or improvements. As demonstrated in Table 2, in the past decade, the Clonezilla live has been upgraded from the initial single machine version to the latest version, which can support the massive deployment in large-scale computers. The initial single machine version 1.0.0 could support backup and recovery in a computer. The latest version 2.5.2, which can support massive deployment in large-scale computers, was rolled out in 2017. The three types of system deployment (i.e., single machine backup and recovery, massive deployment of disks in a computer, and massive deployment in large-scale computers) were integrated into one program, Clonezilla live. All three modes of system deployment were developed based on the proposed system architecture.

Table 2. Lists of the major changes to and improvements in Clonezilla live's development.

Clonezilla Live Version	Year	Changes/Improvements
1.0.0	2007	Initial release of Clonezilla live. Introduced single machine backup and recovery
1.0.9	2008	Introduced Partclone in Clonezilla live
1.2.11	2011	Introduced massive deployment of disks in a computer
2.5.2	2017	Introduced massive deployment in large-scale computers

4. Experimental Results

In order to validate the open source massive deployment system as proposed and implemented, two experiments were performed to demonstrate the feasibility of the proposed novel open source massive deployment system. The first experiment involves deploying a Linux system up to 30 USB flash drives in a single machine. The second experiment verified the feasibility of the second type of massive deployment, the network connection. The image of a Linux system was deployed to a maximum of 32 machines. The details of the first and second experiments will be demonstrated in Sections 4.1 and 4.2, respectively.

4.1. Massive Deployment of Disks in a Computer

The first experiment verifies the feasibility of the first type of massive deployment, the local connection. The massive deployment of USB flash drives on a PC was designed and conducted. A system image was deployed to a maximum of 30 flash drives by using four USB hubs on an x86-64 PC.

The experimental environment consisted of:

- A PC with Intel i7-3770 CPU (3.4 GHz), 32 GB DDR4 Dynamic Random-Access Memory (DRAM), and four USB 2.0, and four USB 3.0 ports
- Two groups of USB flash drives were used: The first group (Type 1) includes 30 32 GB USB 3.0 drives. The second group (Type 2) includes 30 16 GB USB 2.0 flash drives
- USB hubs: 4 Broway USB 3.0 Hubs (BW-U3038B)

The main USB hub was plugged into the USB port of the PC. Three other USB hubs were connected to the main USB hub, as demonstrated in Figure 4.

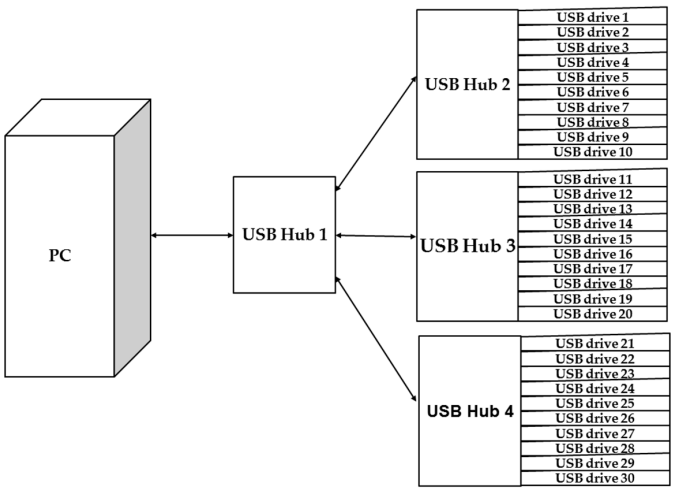

Figure 4. The experiment for massive deployment of USB flash drives: A USB 3.0 hub was used as the main hub; three other USB 3.0 hubs were connected to the main hub; each one of the three USB 3.0 hubs was used to connect 10 USB flash drives.

In the beginning, an image was taken of a template machine being installed with the Ubuntu Linux system. The size of blocks being used on the disk was 3.8 GB, and the Linux system was saved

as an image of 1.2 GB by Clonezilla live using the program pigz. The image and Clonezilla live was put on a master flash drive. The PC was booted with the 'to ram' option in Clonezilla live, which enables Clonezilla live to copy both the OS and the image file to the Random-Access Memory (RAM) when booting. This action facilitated the reading time for the OS and the image file to be minimized when conducting a massive deployment. To eliminate any read and write from the cache memory, the machine was rebooted after each deployment was finished. The command [46] that was being used to deploy the image "my-xenial-img" to 30 USB flash drives was as follows:

ocs-restore-mdisks -b -p -g auto -e1 auto -e2 -nogui -batch -r -j2 -p true my-xenial-img sdf sdg sdh sdi sdj sdk sdl sdm sdn sdo sdp sdq sdr sds sdt sdu sdv sdw sdx sdy sdz sdaa sdab sdac sdad sdae sdaf sdag sdah sdai

Before the mass deployment of USB flash drives was started, an understanding of the writing rates for the two types of flash drives is essential. Therefore, the benchmarking program, the IOzone [47], was used to evaluate the writing rate between the two configurations, Types 1 and 2. Type 1 consists of 32 GB USB 3.0 flash drives. Type 2 consists of 16 GB USB 2.0 flash drives. The evaluation results are demonstrated in Figure 5a. According to the evaluation results, the performance of Type 1 is about two times that of the performance of Type 2. Figure 5b,c further demonstrate the performance of writing an image to various numbers of USB flash drives 1, 2, . . . , 30 USB flash drives, respectively. While the number of USB flash drive(s) is small, e.g., 1, 2, . . . , 10, the deploy time to Type 2 USB flash drive(s) is about two times the deploy time to a single Type 1 USB flash drive (refer to Figure 5b,c). However, the discrepancies between the deploy time to both Types 1 and 2 USB flash drives approaches zero when the number of USB flash drives approaches 30. All of the USB flash drives being deployed have been verified as bootable and can successfully enter the Ubuntu Linux on the test machines. Therefore, the feasibility for the massive deployment of disks in a single computer by the proposed architecture has also been verified.

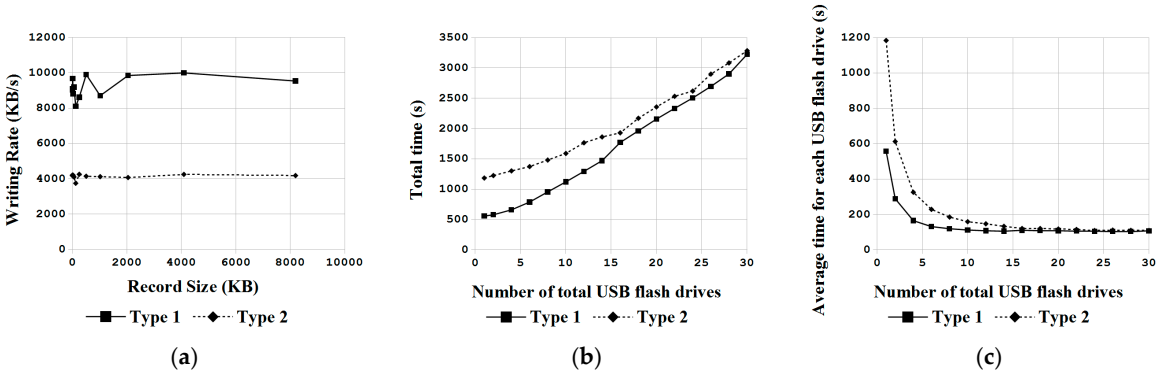

Figure 5. (**a**) Writing rate for two types of USB flash drives; (**b**) total time for deploying various number of USB flash drives; and, (**c**) average time for deploying various number of USB flash drives.

4.2. Massive Deployment in Large-Scale Computers

The second experiment verified the feasibility of the second type of massive deployment, the network connection. The image of a Linux system was deployed to a maximum of 32 machines within a server and a network switch. The experimental environment consisted of:

- A server: a Dell T1700 machine which plays the role of server. The CPU of the Dell T1700 computer is the 3.3 GHz Intel Xeon E3-1226. The size of DRAM is 16 GB. The size of the hard drive is 1 Tera Byte (TB).
- PC clients: the Dell T1700 PCs with the same configuration as the one fulfilling the role as the server were used to act as the PC clients.
- Network switch: the Cisco Catalyst 3560G switch with 48 gigabits ports was used as the network switch.

The PC clients were connected to the Cisco network switch with the Category 5e network cables, as shown in the schematic Figure 6.

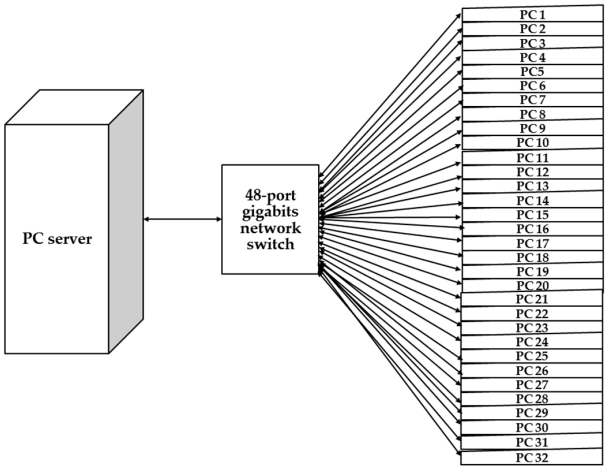

Figure 6. The experiments of massive deployment in large-scale computers. 32 PC clients were connected to the PC server with 48-port Cisco network switch and the Category 5e network cables.

In the beginning, an image was taken from a template machine that was installed with an Ubuntu Linux system. The size of the blocks that were being used on the disk was 50 GB, and the Linux system was saved as an image of 34 GB by Clonezilla live using pigz. The image was put in the local hard drive belonging to the server. The server was booted with the 'to ram' option in the Clonezilla live boot menu, and the local hard drive was mounted so that the image that was saved from the template machine can be read. After that, the server entered the lite-server mode, which relays the DHCP requests from the PC clients to the existing DHCP service in the LAN. After choosing the image, the multicast mode is then assigned. The client machines were booted from the PXE. Six tests were conducted, i.e., deploying the image to 1, 2, 4, 8, 16 and 32 clients. Figure 7 demonstrates the results of the massive deployment. Figure 7a,b illustrate the total time and the average time to massively deploy the image, respectively. Based on the illustrations, the average time to deploy a machine is mainly reduced by using the multicast mechanism in the massive deployment. All of the client machines were verified as bootable and could enter the Ubuntu Linux. Hence, the second experiment verified the feasibility of massive deployment in large-scale computers.

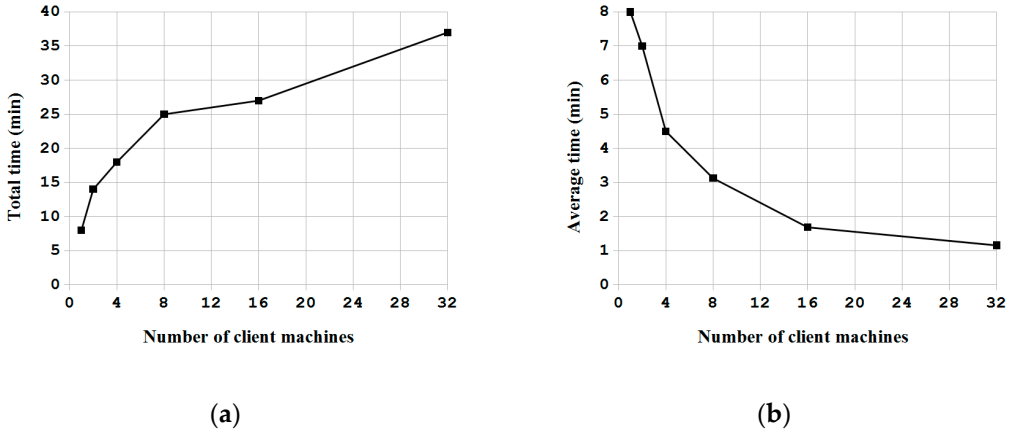

Figure 7. (**a**) Total time and (**b**) average time being required for deploying various numbers of computers using the multicast mechanism.

5. Discussion

5.1. Performance of Massive Deployment by Using Local Connection

The speedup, S, for the first experiment is defined as:

$$S = \frac{T_s}{T_p} \tag{1}$$

where T_s is the time to deploy multiple USB flash drives in sequence and T_p is the time to deploy multiple USB flash drives in parallel.

When an image of size D is deployed to n USB flash drives in parallel on a machine, the bandwidth of the USB bus is B_w, and the writing rate for each flash drive is H_w. The total deployment time T_n can be formulated as:

$$T_n = Max\left[\frac{D \times n}{B_w}, \frac{D \times n}{H_w \times n}\right] = D \times n \frac{1}{Min[B_w, H_w \times n]} \tag{2}$$

where Max and Min are the maximum and minimum functions, respectively.

When $n \leq B_w/H_w$, $T_n = D/H_w$, but when $n > B_w/H_w$, $T_n = D \times n/B_w$. Thus, the speedup, S, can be formulated as:

$$S = \begin{cases} \frac{\frac{D}{H_w}}{\frac{D}{H_w} \times \frac{1}{n}} = n, n \leq \frac{B_w}{H_w} \\ \frac{\frac{D}{H_w}}{\frac{D \times n}{B_w} \times \frac{1}{n}} = \frac{B_w}{H_w}, n > \frac{B_w}{H_w} \end{cases} \tag{3}$$

Therefore, from Equation (3), Bw/Hw is the key factor to judge the speedup for massive deployment by using the local connection. In theory, the maximum transfer rate of the USB 2.0 is 480 Mb/s. From Figure 5a, the average writing rates for Types 1 and 2 flash drives were 9.2 MB/s and 4.1 MB/s, respectively. For the two types of USB flash drives, the ratio of the bandwidth to the writing rate is demonstrated in Table 3. For the Type 1 flash drives, when the number of drives was less than 4, the speedup increased almost linearly. When the number of drives was more than 6, the speedup increased more slowly. When the number of drives was more than 10, the speedups were almost the same, which was about 5. For Type 2 flash drives, when the number of drives was less than 10, the speedup increased almost linearly. When the number of drives was more than 16, the speedup is almost the same, which was about 10 to 11. The number of the total USB flash drives versus the speedup for each configuration is illustrated in Figure 8. Due to the overhead during the system deployment, including the USB bus constraints, the time to create the partition table, and the time to install the boot loader on the flash drives, the theoretic ratio of bandwidth to the writing rate Bw/Hw being demonstrated in Table 3 cannot match the experimental values in Figure 8 exactly when checking Equation (3). Nevertheless, the trend matches well between the theory and experiments, since for both Types 1 and 2, when the number of drives being deployed "n" is smaller than Bw/Hw, the speedup values that were obtained in the experiments increased almost linearly. Conversely, the speedup values obtained in the experiments became almost the same, which are roughly the value of Bw/Hw when the number of drives being deployed "n" is larger than Bw/Hw.

Table 3. The ratio of bandwidth to the writing rate.

USB Type	B_w (MB/s)	H_w (MB/s)	B_w/H_w
Type 1	60.0 (in theory)	9.2	6.5
Type 2	60.0 (in theory)	4.1	14.3

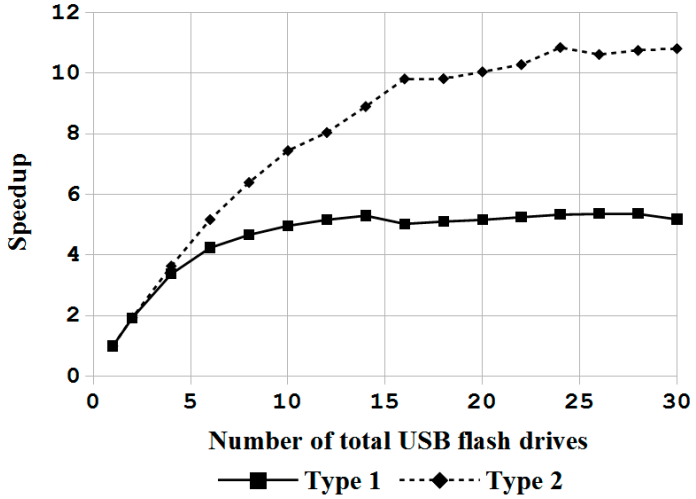

Figure 8. Speedup for deploying USB flash drives.

5.2. *Performance of Massive Deployment by Using Network Connections*

Similar to Equation (1) being defined in Section 5.1, the speedup for massive deployment in large-scale computers is defined as the ratio of time being required to deploy multiple computers in sequence to the time that is required to deploy multiple computers in the multicast mechanism. Figure 9 illustrates the speedup factor versus various configurations of machines, 1, 2, 4, 8, 16 and 32 in the test environment. The ideal speedup factor assumes no overhead in massive deployment. So, the speedup increases linearly as the number of client machines increases. However, for the real-world number, derived from the experiment in Section 4.2, the speedup is getting saturated as more computers are deployed using the multicast mechanism. From Figure 7b, the efficiency became better when deploying more than eight machines, because the average time differences are decreasing when compared with the cases for deploying one, two, and four machines. This finding is consistent with previous research result in the similar experiments by Lee et al. [37], which concluded that the total network bandwidth consumption of the deployment is significantly reduced if the machines are more than eight machines in general.

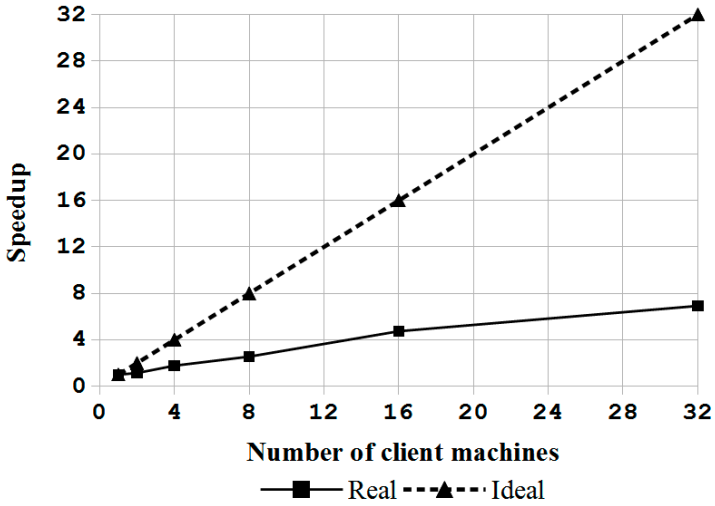

Figure 9. Speedup for deploying massive computers using multicast mechanism.

5.3. Cost Comparisons

The feasibility of Clonezilla live to deploy one source image to multiple drives in a single machine has been verified. When using other deployment software, users must deploy the source image one by one. Otherwise, users must write their own script(s) or software for this kind of massive deployment. However, users' writing of their own scripts makes the deployment more complicated. Hardware solutions are also available for deploying one source image to multiple drives. However, such hardware solutions are much more expensive than using Clonezilla live. The overall cost to assemble the USB drive duplicator by using four USB hubs and Clonezilla live in this experiment was about $200. (The cost of the PCs and the time cost for learning how to use the Clonezilla live are excluded.) Furthermore, the PC is easily available and Clonezilla live is ready to use; no installation and configuration are required. Hence, the cost of the PC and the software learning can be ignored when comparing Clonezilla live versus other solutions. In Table 4, the prices of the solutions that are similar to Clonezilla live are summarized. Meanwhile, the ratios of Clonezilla live to the costs of each solution are provided. In general, the average price of other similar solutions was $2561, which is about 13 times the cost of the Clonezilla live solution.

In addition, Clonezilla live is an open source program. All of the source codes are easily available on the GitHub [48,49] platform. The developer can modify and add new features based on this paper easily. Thus, this USB drive duplicator based on Clonezilla live has higher values than other commercial proprietary products from the aspects of cost, upgradability, and modification.

Table 4. Commercial USB flash drive duplicator cost and cost saving percent for using Clonezilla live.

Commercial Proprietary Product	Price (US$)	Cost Savings in Using Clonezilla Live (%)
SySTOR 1 to 31 Multiple USB Thumb Drive Duplicator/USB Flash Card Copier (USBD-31) [50]	2360	91.5
BestDuplicator Premium-M Series-31 Target (1 to 31) USB Duplicator/Multiple Flash USB Card Copier [51]	2750	92.7
1-To-31 USB Duplicator [52]	2789	92.8
1 to 31 Econ Tower Series USB Drive Duplicator/USB Flash Card Wiper (UB632DG) [53]	2345	91.5
Average	2561	92.2

5.4. Comparisions with Other System Deployment Solutions

The proposed flexible architecture and the Clonezilla implementation can provide better features than other proprietary or open source software. The comparisons between the Clonezilla and other nine software packages (which include six open source and three proprietary solutions) are summarized in Table 5. All of the open source programs are released under the GPL license, except for OpenStack Ironic, which is released under the Apache License. Two of them, i.e., FOG and OpenGnSys, also use the Partclone as the file system imaging engine. This offers the advantage of open source software, which allows the development and contribution of the software to be used and recognized by others. As shown in Table 5, Clonezilla is superior to the proprietary software in the supporting of more file systems. In addition, Clonezilla is also superior to other open source software. Clonezilla can be started from a live system, supports the massive deployment of disks in a computer, and provides a client-server mechanism which can be used for massive deployment in large-scale computers. Based on the comparison results, the proposed novel architecture and the Clonezilla implementation in this research have more features. These aspects show that the developed massive deployment system in this study can be a better choice for system administrators.

Table 5. Comparisons of system deployment software.

Program	License	Case *				File System Supported
		1	2	3	4	
Clonezilla	GPL	Y	Y	Y	Y	ext2, ext3, ext4, reiserfs, reiser4, xfs, jfs, btrfs, f2fs, nilfs2, FAT, NTFS, HFS+, UFS, minix, VMFS3, VMFS5
FOG	GPL	N	Y	N	Y	ext2, ext3, ext4, reiserfs, reiser4, xfs, jfs, btrfs, f2fs, nilfs2, FAT, NTFS, HFS+, UFS, minix, VMFS3, VMFS5
Fsarchiver	GPL	Y	N	N	N	ext2, ext3, ext4, reiserfs, reiser4, xfs, jfs, btrfs, HPFS, FAT, NTFS
Mondo Rescue	GPL	Y	Y	N	Y	ext2, ext3, ext4, jfs, xfs, ReiserFS, FAT
OpenGnSys	GPL	N	Y	N	Y	ext2, ext3, ext4, reiserfs, reiser4, xfs, jfs, btrfs, f2fs, nilfs2, FAT, NTFS, HFS+, UFS, minix, VMFS3, VMFS5
OpenStack Ironic	Apache License	N	N	N	Y	ext2, ext3, ext4, xfs, jfs, btrfs, FAT, NTFS
Acronis True Image	Proprietary	Y	Y	N	Y	ext2/ext3/ext4, ReiserFS, HFS+, APFS, FAT, NTFS
WDS (Windows Deployment Services)	Proprietary	Y	Y	N	Y	FAT, NTFS
ASR (Apple Software Restore)	Proprietary	Y	Y	N	Y	HFS+

* 1. From a live system (run without installation); 2. Raw copying (sector by sector); 3. Massive deployment of disks in a computer; 4. Server/client.

6. Conclusions and Future Work

In this paper, a novel system architecture was proposed and developed in an open source manner to support single machine backup and recovery, massive deployment of disks in a computer, and massive deployment in large-scale computers. To the best of the authors' knowledge, this is the first open source system deployment program that can provide these three types of functions. In addition, this system deployment software can support various OSs, including Linux, MacOS, MS Windows and FreeBSD. A comparison between Clonezilla live with other commercial proprietary or open source software was provided. Regardless, whether considered through the aspect of the file system being support, the capability of starting from a live OS, the support of massive deployment of disks in a computer, and the provision of the client-server mechanism for massive deployment in large-scale computers, the proposed mass deployment system is a better choice. The experiments that were performed by deploying a Linux system to 1 to 30 USB flash drives in a single machine and to 1 to 32 machines in a network using the software being developed in this work have demonstrated the feasibility and efficiency of the proposed work. A comparison between the cost of the USB drive duplicator that was developed in this research, which is $200, is 92% lower than the cost of commercial proprietary products with similar functionalities. In addition, due to the open source nature, this USB drive duplicator can be upgraded easily. The experiments also validated the capabilities of Clonezilla live to conduct massive deployment in large-scale computers. The result is consistent with previous works. In general, the comparisons with other utilities and the experiment results have proved and demonstrated that the proposed architecture and the software being developed in this study are highly efficient and cost-effective for system deployment.

For future research possibilities, more file systems can be supported. Meanwhile, the multicast mechanism and the support of BitTorrent to distribute images to large-scale computers can be further improved. Also, systems can be upgraded to support non-x86 CPU (e.g., the ARM processor) based systems and support for the massive deployment in these non-x86 computer systems, as well as the Open Network Install Environment (ONIE) network switch systems [54], because this will allow for all of the facilities in a cabinet inside a data center to be ready for use after they are deployed. In addition, the proposed novel open source massive deployment system can further be expanded to support the fog computing or even the fog of everything [6] in the future.

Author Contributions: S.J.H.S. and C.-K.S. developed the program Clonezilla, packed the Linux distribution Clonezilla live, conceived and designed the experiments and analyzed the data; Y.-C.T. developed the file system processing module engine Partclone; S.J.H.S. wrote and edited the paper; and J.-N.J. finalized the paper. C.-Y.H. rewrote the whole article and revised the work according to the review opinions.

Acknowledgments: This project was supported by the National Center for High-Performance Computing in Taiwan. The authors thank those Clonezilla users who have provided ideas and reported problems with the program.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Papadopoulos, P.M.; Katz, M.J.; Bruno, G. Npaci rocks: Tools and techniques for easily deploying manageable linux clusters. *Concurr. Comput. Pract. Exp.* **2003**, *15*, 707–725. [CrossRef]
2. Mirielli, E.; Webster, L.; Lynn, M. Developing a multi-boot computer environment and preparing for deployment to multiple workstations using symantec ghost: A cookbook approach. *J. Comput. Sci. Coll.* **2005**, *20*, 29–36.
3. Chandrasekar, A.; Gibson, G. *A Comparative Study of Baremetal Provisioning Frameworks*; Parallel Data Laboratory, Carnegie Mellon University, Technical Report CMU-PDL-14-109; Carnegie Mellon University: Pittsburgh, PA, USA, 2014.
4. Sotomayor, B.; Montero, R.S.; Llorente, I.M.; Foster, I. Virtual infrastructure management in private and hybrid clouds. *IEEE Internet Comput.* **2009**, *13*. [CrossRef]
5. Yamato, Y. Openstack hypervisor, container and baremetal servers performance comparison. *IEICE Commun. Express* **2015**, *4*, 228–232. [CrossRef]
6. Baccarelli, E.; Naranjo, P.G.V.; Scarpiniti, M.; Shojafar, M.; Abawajy, J.H. Fog of everything: Energy-efficient networked computing architectures, research challenges, and a case study. *IEEE Access* **2017**, *5*, 9882–9910. [CrossRef]
7. Feller, J.; Fitzgerald, B. *Understanding Open Source Software Development*; Addison-Wesley: London, UK, 2002.
8. Love, R. *Linux Kernel Development*; Novell Press: Old Tappan, NJ, USA, 2005.
9. Kumar, R.; Gupta, N.; Charu, S.; Jain, K.; Jangir, S.K. Open source solution for cloud computing platform using openstack. *Int. J. Comput. Sci. Mob. Comput.* **2014**, *3*, 89–98.
10. Sun, C.; Shiau, S.J.H.; Wang, J.; Tsai, T. Clonezilla: A Next Generation Clone Solution for Cloud. In Proceedings of the Oral presented at Open Source Conference Tokyo/Fall, Tokyo, Japan, 7–8 September 2012.
11. Aswani, K.; Anala, M.; Rathinam, S. Bare metal cloud builder. *Imp. J. Interdiscip. Res.* **2016**, *2*, 1844–1851.
12. Sampaio, D.; Bernardino, J. Open source backup systems for SMES. In *New Contributions in Information Systems and Technologies*; Springer: Berlin, Germany, 2015; pp. 823–832.
13. Sanguino, T.M.; de Viana, I.F.; García, D.L.; Ancos, E.C. Opengnsys: A novel system toward centralized deployment and management of computer laboratories. *Comput. Educ.* **2014**, *75*, 30–43. [CrossRef]
14. Doelitzscher, F.; Sulistio, A.; Reich, C.; Kuijs, H.; Wolf, D. Private cloud for collaboration and e-learning services: From IAAS to SAAS. *Computing* **2011**, *91*, 23–42. [CrossRef]
15. Konrad, M.; Bonnes, U.; Burandt, C.; Eichhorn, R.; Nonn, P.; Enders, J.; Pietralla, N. Digital base-band RF control system for the superconducting darmstadt electron linear accelerator. *Phys. Rev. Spec. Top. Accel. Beams* **2012**, *15*, 052802. [CrossRef]
16. Petersen, R. *Ubuntu 16.04 lts Server: Administration and Reference*; Surfing Turtle Press: Alameda, CA, USA, 2016.
17. Crago, S.; Dunn, K.; Eads, P.; Hochstein, L.; Kang, D.-I.; Kang, M.; Modium, D.; Singh, K.; Suh, J.; Walters, J.P. Heterogeneous Cloud Computing. In Proceedings of the 2011 IEEE International Conference on Cluster Computing (CLUSTER), Austin, TX, USA, 26–30 September 2011; pp. 378–385.
18. Rsync Program. Available online: https://rsync.samba.org (accessed on 3 September 2017).
19. Cornec, B. Mondo rescue: A GPL disaster recovery solution. *Proc. Linux Symp.* **2008**, *1*, 77–84.
20. Relax-and-Recover (ReaR) Project. Available online: http://relax-and-recover.org (accessed on 3 September 2017).
21. DRLM (Disaster Recovery Linux Manager). Available online: http://drlm.org (accessed on 3 September 2017).
22. Storix System Backup Administrator. Storix Inc.: San Diego, CA, USA. Available online: https://www.storix.com (accessed on 3 September 2017).
23. Cougias, D.J.; Heiberger, E.L.; Koop, K. *The Backup Book: Disaster Recovery from Desktop to Data Center*; Network Frontiers: Lecanto, FL, USA, 2003.
24. Acronis True Image. Acronis International GmbH, Germany. Available online: https://www.acronis.com (accessed on 3 September 2017).

25. Partimage Software. Available online: http://www.partimage.org (accessed on 3 September 2017).
26. FSArchiver-File System Archiver for Linux. Available online: http://www.fsarchiver.org (accessed on 3 September 2017).
27. Redo Backup and Recovery. Available online: http://redobackup.org (accessed on 3 September 2017).
28. FOG Project. Available online: https://fogproject.org (accessed on 3 September 2017).
29. Windows Deployment Services. Microsoft Corporation, Redmond, WA, USA. Available online: https://msdn.microsoft.com/en-us/library/aa967394.aspx (accessed on 3 September 2017).
30. Edge, C.S.; Smith, W. Mass deployment. In *Enterprise Mac Administrator's Guide*; Springer: Berlin, Germany, 2015; pp. 213–281.
31. Hibler, M.; Stoller, L.; Lepreau, J.; Ricci, R.; Barb, C. Fast, Scalable Disk Imaging with Frisbee. In Proceedings of the USENIX Annual Technical Conference, General Track, San Antonio, TX, USA, 9–14 June 2003; pp. 283–296.
32. Pullakandam, R. Emustore: Large Scale Disk Image Storage and Deployment in the Emulab Network Testbed. Master's Thesis, University of Utah, Salt Lake City, UT, USA, 2014.
33. Lin, X.; Hibler, M.; Eide, E.; Ricci, R. Using deduplicating storage for efficient disk image deployment. *EAI Endorsed Trans. Scalable Inf. Syst.* **2015**, *2*, e1.
34. Kominos, C.G.; Seyvet, N.; Vandikas, K. Bare-Metal, Virtual Machines and Containers IN Openstack. In Proceedings of the 2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN), Paris, France, 7–9 March 2017; pp. 36–43.
35. Lima, S.; Rocha, Á.; Roque, L. An overview of openstack architecture: A message queuing services node. *Clust. Comput.* **2017**, 1–12. [CrossRef]
36. Books, L. *Network Booting: Preboot Execution Environment, Bootstrap Protocol, Netboot, GPXE, Remote Initial Program Load*; General Books LLC: Memphis, TN, USA, 2010.
37. Lee, K.-M.; Teng, W.-G.; Wu, J.-N.; Huang, K.-M.; Ko, Y.-H.; Hou, T.-W. Multicast and customized deployment of large-scale operating systems. *Autom. Softw. Eng.* **2014**, *21*, 443–460. [CrossRef]
38. Manzoor, U.; Nefti, S. Silent unattended installation package manager—SUIPM. In Proceedings of the 2008 International Conference on Computational Intelligence for Modelling Control & Automation, Vienna, Austria, 10–12 December 2008; pp. 55–60.
39. Manzoor, U.; Nefti, S. Quiet: A methodology for autonomous software deployment using mobile agents. *J. Netw. Comput. Appl.* **2010**, *33*, 696–706. [CrossRef]
40. Raymond, E.S. *The Art of UNIX Programming*; Addison-Wesley Professional: Boston, MA, USA, 2003.
41. Stallman, R. *Free Software, Free Society: Selected Essays of Richard M. Stallman*; Lulu.com: Morrisville, NC, USA, 2002.
42. Abreu, R.M.; Froufe, H.J.; Queiroz, M.J.R.; Ferreira, I.C. Mola: A bootable, self-configuring system for virtual screening using AutoDock4/Vina on computer clusters. *J. Cheminform.* **2010**, *2*, 10. [CrossRef] [PubMed]
43. Kroah-Hartman, G. *Linux Kernel in a Nutshell*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2006.
44. Shojafar, M.; Abawajy, J.H.; Delkhah, Z.; Ahmadi, A.; Pooranian, Z.; Abraham, A. An efficient and distributed file search in unstructured peer-to-peer networks. *Peer-to-Peer Netw. Appl.* **2015**, *8*, 120–136. [CrossRef]
45. Qiu, D.; Srikant, R. Modeling and performance analysis of bittorrent-like peer-to-peer networks. In *ACM SIGCOMM Computer Communication Review*; ACM: New York, NY, USA, 2004; pp. 367–378.
46. Clonezilla Project. Available online: http://clonezilla.org (accessed on 3 September 2017).
47. Xavier, M.G.; Neves, M.V.; Rossi, F.D.; Ferreto, T.C.; Lange, T.; De Rose, C.A. Performance evaluation of container-based virtualization for high performance computing environments. In Proceedings of the 2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, Belfast, UK, 27 February–1 March 2013; pp. 233–240.
48. Clonezilla Source Codes Repository. Available online: https://github.com/stevenshiau/clonezilla (accessed on 3 September 2017).
49. Partclone Project. Available online: https://github.com/Thomas-Tsai/partclone (accessed on 3 September 2017).
50. Systor 1 to 31 Multiple USB Thumb Drive Duplicator/USB Flash Card Copier (USBD-31). Available online: https://www.amazon.com/SySTOR-Multiple-Duplicator-Copier-USBD-31/dp/B00DV39MN4 (accessed on 18 April 2018).

51. Bestduplicator Premium-M Series-31 Target (1 to 31) USB Duplicator/Multiple Flash USB Card Copier. Available online: https://www.amazon.com/BestDuplicator-Premium-M-Target-Duplicator-Multiple/dp/B00CFXZ7H6 (accessed on 18 April 2018).
52. 1-to-31 USB Duplicator. Available online: https://www.amazon.com/Kanguru-Solutions-U2D2-31-1-To-31-Duplicator/dp/B00BO0MGSE (accessed on 18 April 2018).
53. 1 to 31 Econ Tower Series USB Drive Duplicator/USB Flash Card Wiper (ub632dg). Available online: https://www.amazon.com/UB632DG-Tower-Drive-Duplicator-Flash/dp/B0189S5I2E (accessed on 18 April 2018).
54. Sherwood, R. *Tutorial: White Box/Bare Metal Switches*; Open Networking User Group: New York, NY, USA, 2014.