Sustainability **2015**, *7*, 7568-7580; doi:10.3390/su7067568

OPEN ACCESS

sustainability

ISSN 2071-1050

www.mdpi.com/journal/sustainability

Article

Efficient Sustainable Operation Mechanism of Distributed Desktop Integration Storage Based on Virtualization with Ubiquitous Computing

Hyun-Woo Kim [1], Jong Hyuk Park [2], Duinkhorjav Majigsuren [1] and Young-Sik Jeong [1,*]

[1] Department of Multimedia Engineering, Dongguk University, Seoul 100-715, Korea; E-Mails: hwkim@dongguk.edu (H.-W.K.); majigaa200267@gmail.com (D.M.)

[2] Department of Computer Science and Engineering, Seoul National University of Science and Technology, Seoul 139-743, Korea; E-Mail: jhpark1@seoultech.ac.kr

***** Author to whom correspondence should be addressed; E-Mail: ysjeong@dongguk.edu; Tel.: +82-2-2260-3374; Fax: +82-2-2260-8898.

Academic Editors: Jason C. Hung and Cho-Li Wang

Received: 1 May 2015 / Accepted: 8 June 2015 / Published: 12 June 2015

Abstract: Following the rapid growth of ubiquitous computing, many jobs that were previously manual have now been automated. This automation has increased the amount of time available for leisure; diverse services are now being developed for this leisure time. In addition, the development of small and portable devices like smartphones, diverse Internet services can be used regardless of time and place. Studies regarding diverse virtualization are currently in progress. These studies aim to determine ways to efficiently store and process the big data generated by the multitude of devices and services in use. One topic of such studies is desktop storage virtualization, which integrates distributed desktop resources and provides these resources to users to integrate into distributed legacy desktops via virtualization. In the case of desktop storage virtualization, high availability of virtualization is necessary and important for providing reliability to users. Studies regarding hierarchical structures and resource integration are currently in progress. These studies aim to create efficient data distribution and storage for distributed desktops based on resource integration environments. However, studies regarding efficient responses to server faults occurring in desktop-based resource integration environments have been insufficient. This paper proposes a mechanism for the sustainable operation of desktop storage (SODS) for high operational availability. It allows for the easy addition and removal of desktops in

desktop-based integration environments. It also activates alternative servers when a fault occurs within a system.

Keywords: sustainable operation; virtualization; integration resource; distributed desktop; ubiquitous computing

1. Introduction

A diverse group of smart devices have been developed following the rapid growth of ubiquitous computing processing technologies. This diverse group of devices has created large increases in data usage by devices like digital cameras, smart televisions, smart phones, and tablet PC. Due to increases in atypical data, character data, image data, and position data, the era of big data has begun. To the enhanced performance, miniaturization, and convenient portability of these diverse smart devices, the leisure time of individuals and the work efficiency of businesses have increased [1–8]. The increase in the number of these devices has created a subsequent increase in data creation points. Data are now being rapidly created due to the numerous services provided by smart devices. There are currently many virtualization-based studies in progress regarding the efficient storage and management of big data. This virtualization comes in many forms, including application virtualization, hardware virtualization, desktop virtualization, network virtualization, server virtualization, and storage virtualization. Desktop storage virtualization (DSV) integrates the storage resources of distributed legacy desktops and provides resources in response to other users' requests for storage [9–14].

The reliability of DSV is very important for storage users because this virtualization involves the connection of so many distributed desktops to each other. Therefore, DSV systems attempt to respond to the occurrence of desktop defects or faults in operating servers. These systems also attempt to identify servers that may be down due to incorrect operation by desktop users. Although previous studies of DSV have analyzed hierarchical structures and resource integration in desktop-based integration environments, studies regarding this aspect of efficient operation are currently insufficient.

This research outlines a mechanism for the sustainable operation of desktop storage (SODS). This mechanism enables the use of active responses to desktop server faults occurring in desktop-based storage integration environments. The SODS mechanism allows a separate desktop to operate as an alternative server when the primary desktop server has failed. This mechanism also provides a high quality of service (QoS) to storage users, as it enables them to easily add or remove desktops from their networks.

This paper is composed as follows: In Section 2, previous studies of resource integration schemes, management, and responses to desktop faults are examined. In Section 3, SODS mechanism is explained in depth. In Section 4, the design of SODS mechanism is explained, and in Section 5, the implementation of the SODS mechanism is covered. Section 6 covers evaluation of the maximum operation environment was performed by artificially creating a desktop server. Finally, Section 7 contains a summary of the research's overall conclusions as well as suggestions for future studies.

2. Related Works

Several resource integration schemes have been analyzed in previous studies, including the Google File System (GFS) [15], the desktop Resource Virtualization-Clustering Simulator (DRV-CS) [16], the Resource Integrated System for Big Data (RISBD) [17], the Desktop-Oriented Distributed Public Cloud Storage (CSTORE) [18], and the Hadoop Distributed File System (HDFS) [19].

GFS [15] is a file distribution system consisting of clients who can access control information stored in GFS and who can file and chunk position information appropriately. This system includes a GFS master who manages and names spaces. The GFS has a chunk server that manages chunks and inputs/outputs in response to client requests. Through the GFS master, this system creates chunk copies for data integrity, adjusts the number of chunk copies, and manages unused storage spaces. To prevent chunk server data loss in the event of a fault of the GFS master, this system copies the metadata received from chunk servers and the metadata regarding changes to other local and remote servers. However, when continuous GFS master faults occur, all chunk server data are lost.

In the DRV-CS [16], a clustering scheme allows for efficient integration of distributed desktop resources. The DRV-CS externalizes the performances of distributed desktops and forms clusters of these performances to create efficient hierarchical structures. Although the DRV-CS allows for the selection of optimum desktops based on the required performance, it does have a shortcoming; that is the clustering stops when a server fault occurs. In contrast to this, the mechanism proposed in the present paper enables active responses to sporadic or continuous server faults.

RISBD [17] is a scheme that was proposed as a method for analyzing unused desktop resources being used independently. The RISBD scheme aims to integrate these resources with one another to provide private cloud-based storage. A scheme was also proposed to respond to server faults that may occur due to the integration of too many distributed desktop resources. However, this scheme requires many unnecessary overheads due to its IP-based server selection, which ultimately lowers the performance level of the service. The mechanism proposed in the present paper selects servers more efficiently by considering the performances and locations of desktops and stored resources.

CSTORE [18] is a scheme designed to guarantee independent spaces and data security for users in terms of the operation of distributed desktops. While this system shows better expandability and performance than general public storage systems, it does have one major problem, which is that users cannot access the system when the server has failed.

HDFS [19] consists of DataNodes (a type of individually-distributed desktop resource) and a NameNode that manages the DataNodes. This configuration enables the integration of several thousand desktop resources. Although data integrity is secured by internally-copied data, data stored in the DataNodes will be lost if a NameNode fault occurs. Although a SecondaryNode is implemented if any NameNode fault occurs, data losses cannot be prevented when these faults occur continuously. To overcome this shortcoming, the mechanism proposed in the present paper periodically synchronizes the metadata from data storage locations on distributed desktops.

3. SODS Scheme

SODS scheme based on connected desktops proposed in the present paper has functions for both the server and the client. Desktops are divided into servers and clients according to the composition of operations that exist when the functions are initially implemented. The connected desktops comprise a desktop Master Node (DMN) that plays the role of a server and desktop Storage Nodes (DSN) that are responsible for storing user data. During operation, the DMN adds and deletes desktops and manages stored data and available storage spaces. The DSNs periodically send their state information to the DMN and check whether or not it is operating. The metadata used for selection of an alternative server when a DMN fault has occurred are defined in Table 1.

Table 1. Metadata for the selection of an alternative server when a server fault has occurred in sustainable operation of desktop storage (SODS).

Factor	Description
IP	IP address system-based priorities are given. IP address system-based operation may cause unnecessary overheads because routing information is not included.
Performance	When the central processing unit (CPU) is assumed as α and the Memory is β, the importance of the CPU and the Memory should be set according to the user while ensuring that $\alpha + \beta$ is equal to 100%. Depending on user setting, the CPU can be defined as more important when the enhancement of processing speed is regarded as important, and the Memory can be defined as more important when the number of accommodation times for the connected desktops is regarded as more important. The relevant definition of importance facilitates the selection of the appropriate alternative servers when the number of desktops in the SODS environment is large or small.
Stored Data Amount	In operation-based DSV environments of the SODS, all connected desktops are normalized based on the desktop with the smallest amount of stored data. The stored data of a desktop can cause inefficient additional computing and network bandwidths when the desktop has been selected as an alternative server. For this reason, the SODS gives priority to desktops with small amounts of stored data. If the amount of stored data is not considered, large amounts of resources may be unnecessarily consumed due to overlapping of internal data.
Distance	In operation-based DSV environments of the SODS, response time is determined according to periodic heartbeats and normalization is conducted based on the minimum response time.

The sustainable operation of the desktop implemented as the DMN in DSV environments applying the SODS mechanism is shown in Figure 1. After the initial server start, Accept mode is implemented; this mode makes the mechanism wait for access from the server's desktops. When the desktops have accessed the mechanism, their state information (CPU, Memory, Storage, IP, and Distance) is analyzed to create metadata, which are then reflected on the list. The metadata are then arranged according to the user settings and are transmitted to all connected desktops. This process is repeatedly implemented as desktops are added at the beginning of the configuration of the application of the SODS to the DSV environment.

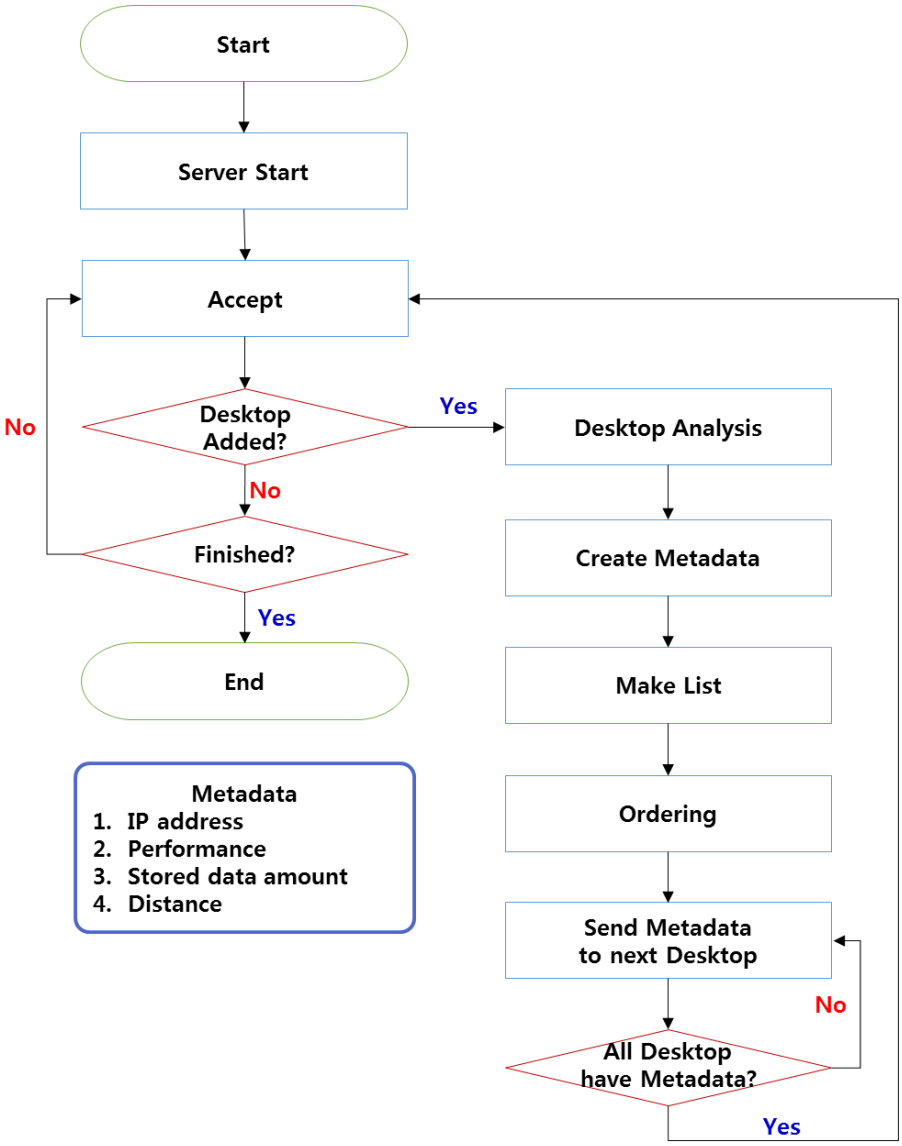

Figure 1. Desktop Master Node (DMN) flow for sustainable operation.

In DSV environments applying the SODS, the desktops implemented as DSNs periodically check whether the server is operating (as shown in Figure 2). The DSNs access the DMN to receive its metadata. The DSNs then transmit heartbeats to check the state of the server and the state of the DMN acting as a server. The DSNs check the state of the server periodically for a predetermined number (N) of seconds set by the user. The operation of the server is usually checked every five seconds. When the server does not operate because a fault has occurred, the Count Value increases. "Count" refers to the number of times a fault occurs; if the count exceeds the number of times set by the user (M), a server failure is considered to have occurred. If a server failure has occurred, each DSN uses metadata to check whether it is supposed to act as the alternative server. The DSN that is supposed to function as the alternative server then performs the function of the DMN. The DSNs that are not the alternative servers will try to access the newly-implemented DMN. Since at least two desktops are required for this operation, continuous occurrences of faults can be responded to in the same manner.

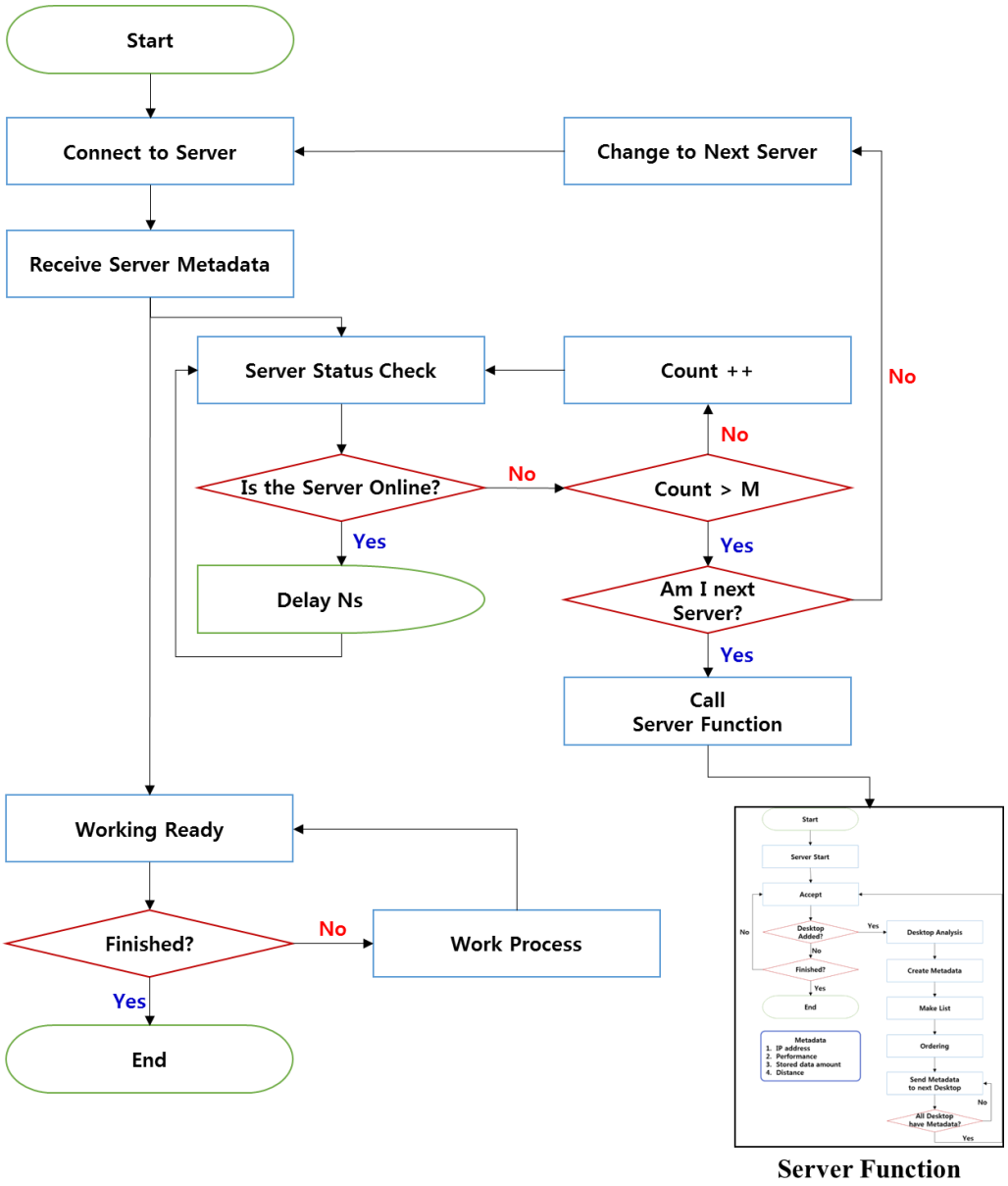

Figure 2. DSN flow for sustainable operation.

4. Design of SOSIDR

The application of an SODS mechanism to distributed desktop integration storage systems can be further defined as a Sustainable Operation System for Integrated desktop Resources (SOSIDR). The components of the SOSIDR are functionally divided into: (1) a User Interface for selecting the DMN and DSN modes, setting desktop information, and determining the number of overlapping storages; (2) an Interaction Broker for transmitting information inputted by the user to the desktop Manager and the Resource Manager to reflect the information; (3) a desktop Manager to analyze and manage connected desktops; (4) a Resource Manager to integrate resources and create and manage the metadata of integrated resources and connected desktops; (5) a Coordinate Converter to process system information for viewing information in the Viewer; and (6), a Viewer that visualizes the states of the desktops and resources. The DSNs are also configured to periodically check the status of the DMN.

As shown in Figure 3, The User Interface is subdivided into six sections: Desktop Information, Duplication Count, Chunk Size, Mode, Connection Information, and Priority Information. Desktop Information is used for setting the allowed storage size of the desktop, Port Number, and data storage locations according to relevant storage provisions. Duplication Count is used for setting the number of times the data storage may overlap as determined in proportion to the number of connected desktops. Chunk Size is used for setting the unit of the size for the stored data (64 MB, 128 MB, 256 MB, or 512 MB). Connection Information is used for setting the IP and Port Number needed by the initial DMN to access the DMN. These settings are applied after it has been determined whether or not the DMN is in operation. Priority Information is used for setting the priorities of the desktops selected as alternatives to the DMN when a DMN failure has occurred. These priorities are applied by taking IPs, Performance, Stored Data Amounts, and Distances into consideration. The IPs are set based on the Internet address system. The Performance is determined considering the CPU and Memory of each desktop. That is, the processing speed and the number of desktops accommodated are applied on the basis of 100%. For instance, if the importance of the CPU is 70% and the importance of the Memory is 30%, the sum of these two is 100%. Stored Data Amount is used for selecting the desktop with the performance closest to the Performance standard set by the user from among the desktops with the smallest sizes of stored data available when storage services are in operation. One reason why a desktop may not be chosen for this purpose is that if the desktop selected as the DMN has a large amount of stored data, the computing and network bandwidth will have to be internally consumed to maintain the Duplication Count. Mode is used for setting each desktop to either operate in the DMN mode or the DSN mode according to the user's setting.

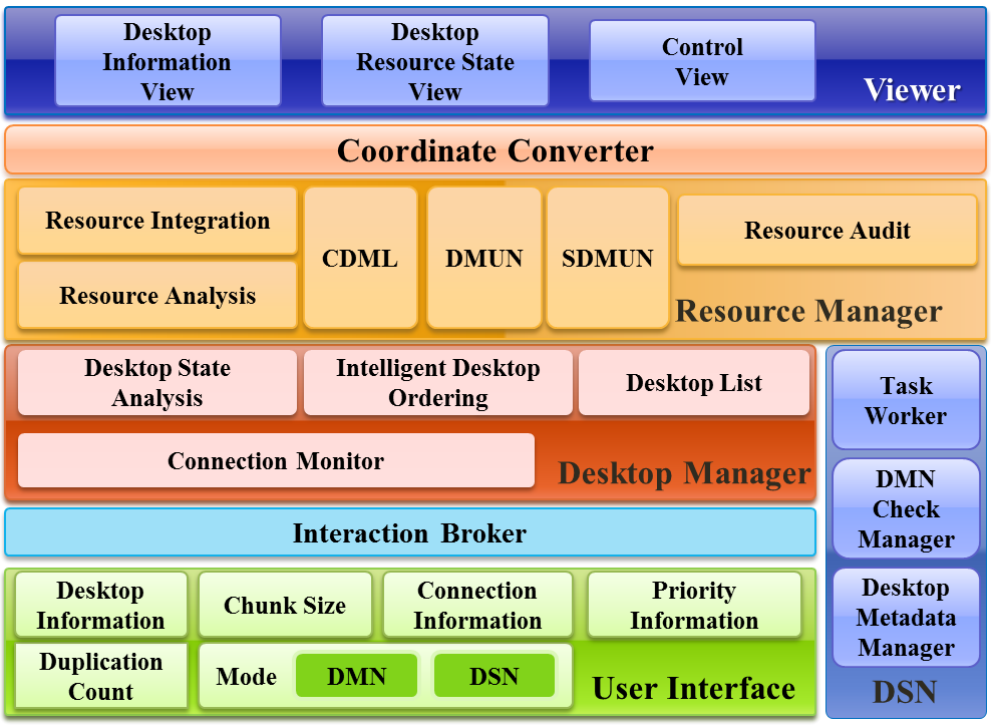

Figure 3. Sustainable Operation System for Integrated desktop Resources (SOSIDR) Architecture for Sustainable Operation.

Interaction Broker delivers the basic desktop information received from the user through the User Interface. It is also responsible for the priority information set for responses to faults that may occur to the desktop Manager and the Resource Manager.

Desktop Manager is subdivided into four stages: Desktop State Analysis, Intelligent Desktop Ordering, the Desktop List, and the Connection Monitor. The Desktop State Analysis analyzes the sizes of data stored in the desktops and the static and dynamic performances of the desktops. The analyzed desktop state information is sent to the Intelligent Desktop Ordering. The Intelligent Desktop Ordering selects the alternative DMN using the desktop state information it receives from the Desktop State Analysis. After this ordering, the list of alternative DMNs is sent to the Desktop List and stored there. Then, the Desktop List sends the list of alternative DMNs to the Resource Manager.

Resource Manager is subdivided into Resource Analysis, Resource Integration, Create Desktop Metadata List (CDML), Desktop Metadata Update Notify (DMUN), Stored Data Metadata Update Notify (SDMUN), and Resource Audit. The Resource Analysis analyzes the storage information from the desktops. Through this analysis, the sizes of storage are determined according to the DMN's storage and distribution. This stage also determines whether data have been properly stored in the relevant DSN and whether the permissible storage set for each DSN is available. The Resource Integration includes the DSN storages based on the Resource Analysis' judgment of the available storage of the DSNs. The CDML creates DMN metadata to synchronize the DMN list sent from the Desktop List of the Desktop Manager to all desktops. The created DMN metadata are transmitted to the DMUN. The DMUN sends the DMN metadata to all connected desktops to conduct synchronization. The SDMUN sends metadata, including the storage locations of the data received from the user, to all desktops to conduct synchronization. Continuous operation can be enabled by selecting an optimum DSN in the case of DMN failure using the DMN metadata.

Coordinate Converter processes information to visualize the desktop state information and resource states on the Viewer. The processed information is delivered to the Viewer.

Viewer takes the desktop and resource information received from the Coordinate Converter and, along with the control views, delivers it to the user for controlling.

DSN consists of a Task Worker (which responds to data storage requests from the DMN), a DMN Check Manager (which checks the operating state of the DMN), and a Desktop Metadata Manager (which manages the metadata received from the DMN). Sustained operation is possible in this system because the Desktop Metadata Manager selects an alternative server when the DMN Check Manager recognizes the DMN's server fault, which initiates the operation of the alternative server.

5. Implementation of SOSIDR

The initial execution screen of SOSIDR is shown in Figure 4. Figure 4-① shows the mode selection from the initial execution. The selection at this time is between the DMN and the DSN. Figure 4-② shows the screen that is activated when the DMN has been selected. This screen displays the information from the current desktop; that is, it displays the desktop's IP and Port Number. Figure 4-② also shows the list of connected desktops and the user's settings, which can be changed using the Set State. Figure 4-③ shows the Data List, which includes information about the data stored in the DSNs and information about the state of the DMN. Information regarding faults that have occurred in

the DMN is checked in real time. The Candidate DMN List shows how the metadata is synchronized according to the priorities of the alternative servers set by the user in the DMN. Figure 4-③ also shows how the data is stored and which storage sizes are integrated through the Set.

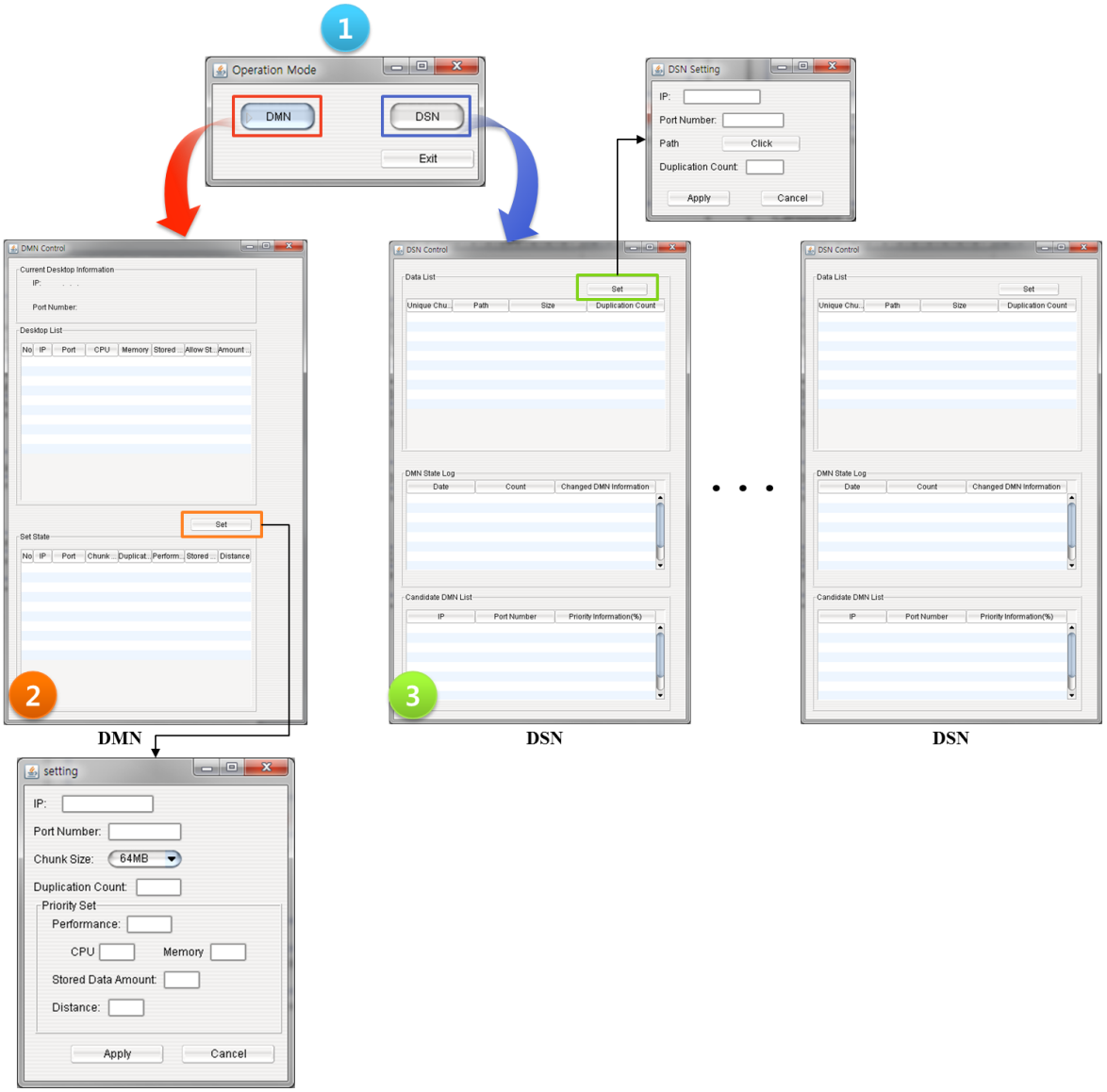

Figure 4. DMN and DSN executions according to SOSIDR Mode setting.

Figure 5 shows the process through which the first DSN in the Candidate DMN List operates as the DMN following the occurrence of a DMN fault in an environment with 5 integrated desktops. Figure 5-① shows a case where the DMN operates normally. Figure 5-② shows disconnections of DSN1, DSN2, DSN3, and DSN4 from the DMN following the occurrence of the fault. Figure 5-③ shows the selection of the first priority through the metadata information in the DSN Candidate List. The first-priority DSN activates the server so that it can begin operating as the DMN. Figure 5-④ shows the DSNs that were not chosen to act as the DMN following a fault. After the new connection is made, the new DMN Candidate List is transmitted to all connected DSNs through the new DMN for synchronization. The DMN also compares the Duplication Count set by the user with the number of overlapping storages to judge whether the Duplication Count satisfies this condition. If the condition is

not satisfied, the DMN calls the DSNs with the correct data and orders them to copy their data to other DSNs to satisfy the Duplication Count. Faults can be repeatedly responded to using this process.

Figure 5. Response process for DMN faults in SOSIDR.

6. Performance Evaluation

Figure 6 shows the operation states that occur when a master fault has occurred in an SOSIDR with 10 connected desktops and an HDFS with 10 connected desktops. In the case of the HDFS, when a NameNode fault has occurred, the SecondaryNode takes over the role of the NameNode and the eight DataNodes operate stable on their own. However, if faults occur continuously, no DataNode will be able to operate. In the case of the SOSIDR, the server operation can be maintained even when faults continuously occur because all the connected desktops have server functions and data storage information and the server priority lists are synchronized. This function can only operate when there are at least two desktops that can serve as DMNs and DSNs. This sustainable operation provides protection against server faults even in desktop environments with several thousand connected desktops.

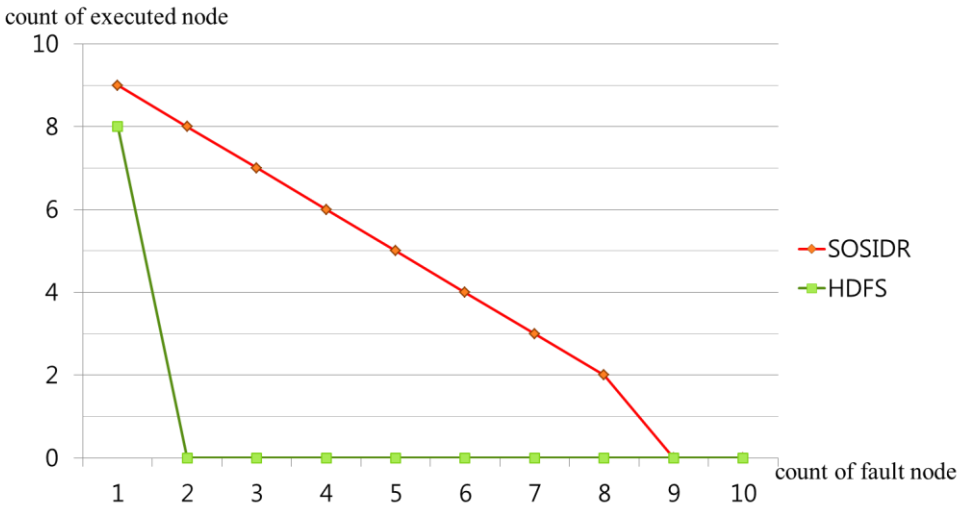

Figure 6. Comparison of the states of desktop operation that occur when a master node fault has occurred.

7. Conclusions

Due to ubiquitous computing, storage servers are now responsible for storing large amounts of diverse data. When continuous faults occur in these systems, huge data losses and costs can result. The present paper proposed an SODS mechanism to respond to conventional server faults. In the SODS, all desktops are connected to each other within the DSV environment. This system allows for the creation of alternative servers based on desktop performances, desktop distances, and the amounts of stored data in each desktop. This method reduces delay times for server re-operation by selecting optimized servers from the connected desktops of a network.

In the future, algorithms should be studied regarding not only the high availability of servers, but also the efficient storage and extraction of data. In addition, studies for the application of these algorithms to network switch configurations should be conducted.

Acknowledgments

This work (Grants No. C0249205) was supported by Business for Cooperative R&D between Industry, Academy, and Research Institute funded Korea Small and Medium Business Administration in 2014. And also this research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2014R1A1A2053564).

Author Contributions

Hyun-Woo Kim and Jong Hyuk Park designed this research; Hyun-Woo Kim and Duinkhorjav Majigsuren retrieved and analyzed the data, and wrote the paper basically; Young-Sik Jeong reviewed the research and initiate the idea and totally process the idea, design and implementation. Jong Hyuk Park and Young-Sik Jeong revised the paper. All authors have read and approved the final manuscript.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. Jeong, Y.-S.; Kim, H.-W.; Jang, H.J. Adaptive resource management scheme for monitoring of CPS. *J. Supercomput*. **2013**, *66*, 57–69.
2. Gil, J.-M.; Park, J.H.; Jeong, Y.-S. Data center selection based on neuro-fuzzy inference systems in cloud computing environments. *J. Supercomput*. **2013**, *66*, 1194–1214.
3. Degefa, F.B.; Won, D. Extended Key Management Scheme for Dynamic Group in Multi-cast Communication. *J. Converg*. **2013**, *4*, 7–13.
4. Malkawi, M.I. The art of software systems development: Reliability, Availability, Maintainability, Performance (RAMP). *Hum.-Centr. Comput. Inf. Sci*. **2013**, *3*, 1–17.
5. Song, E.-H.; Kim, H.-W.; Jeong, Y.-S. Visual Monitoring System of Multi-Hosts Behavior for Trustworthiness with Mobile Cloud. *J. Inf. Process. Syst*. **2012**, *8*, 347–358.
6. Lee, S.-H.; Lee, I.-Y. A Secure Index Management Scheme for Providing Data Sharing in Cloud Storage. *J. Inf. Process. Syst*. **2013**, *9*, 287–300.
7. Shrivastava, N.; Kumar, G. A survey on cost effective multi-cloud storage in cloud computing. *Int. J. Adv. Res. Comput. Eng. Technol*. **2013**, *2*, 1291–1997.
8. Kim, S.-Y.; Roh, H.-C.; Park, C.-H.; Park, S.-H. Analysis of Metadata Server on Clustered File Systems. In Proceedings of the Korea Computer Congress 2009 (KCC), Seoul, Korea, 1–3 July 2009; Volume 36, pp. 64–69.
9. Gaonkar, P.E.; Bojewar, S.; Das, J.A. A Survey: Data Storage Technologies. *Int. J. Eng. Sci. Innov. Technol.* **2013**, *2*, 547–554.
10. Gibson, G.A.; van Meter, R. Network attached storage architecture. *Commun. ACM* **2000**, *43*, 37–45.
11. Dong, B.; Zheng, Q.; Tian, F.; Chao, K.; Ma, R.; Anane, R. An optimized approach for storing and accessing small files on cloud storage. *J. Netw. Comput. Appl*. **2012**, *35*, 1847–1862.
12. Chattopadhyay, M.; Dan, P.K.; Mazumdar, S. Comparison of visualization of optimal clustering using self-organizing map and growing hierarchical self-organizing map in cellular manufacturing system. *Appl. Soft Comput*. **2014**, *22*, 528–543.
13. Silva, J.L.; Campos, J.C.; Harrison, M.D. Prototyping and analysing ubiquitous computing environments using multiple layers. *Int. J. Hum.-Comput. Stud*. **2014**, *72*, 488–506.
14. Mohammed, J. Evolution of the Next Generation of Technologies: Mobile and Ubiquitous Computing. *Int. J. Adv. Res. Sci. Eng. Technol*. **2014**, *1*, 247–253.
15. Wang, M.; Li, B.; Zhao, Y.; Pu, G. Formalizing Google File System. In Proceedings of the 2014 IEEE 20th Pacific Rim International Symposium on Dependable Computing, Singapore, 18–21 November 2014; pp. 190–191.
16. Park, J.H.; Kim, H.-W.; Jeong, Y.-S. Efficiency Sustainability Resource Visual Simulator for Clustered Desktop Virtualization Based on Cloud Infrastructure. *Sustainability* **2014**, *6*, 8079–8091.

17. Kim, H.-W.; Park, J.H.; Jeong, Y.-S. Human-centric storage resource mechanism for big data on cloud service architecture. *J. Supercomput*. **2015**, doi:10.1007/s11227-015-1390-3.
18. Duan, H.; Yu, S.; Mei, M.; Zhan, W.; Li, L. CSTORE: A desktop-oriented distributed public cloud storage system. *Comput. Electr. Eng*. **2015**, *42*, 60–73.
19. Shafer, J.; Rixner, S.; Cox, A.L. The Hadoop Distributed Filesystem: Balancing Portability and Performance. In Proceedings of the IEEE International Symposium on Performance Analysis of System and Software (ISPASS 2010), White Plains, NY, USA, 28–30 March 2010; pp. 122–133.