symmetry

MDPI

Article

DIaaS: Resource Management System for the Intra-Cloud with On-Premise Desktops

Hyun-Woo Kim [1], Jaekyung Han [2], Jong Hyuk Park [3,*] and Young-Sik Jeong [1,*]

1 Department of Multimedia Engineering, Dongguk University, Seoul 04620, Korea; hwkim@dongguk.edu
2 Department of Construction legal Affairs, The Graduate School of Construction Legal Affairs, Kwangwoon University, Seoul 01897, Korea; hjk1014@kw.ac.kr
3 Department of Computer Science and Engineering, Seoul National University of Science and Technology, Seoul 01811, Korea
* Correspondence: jhpark1@seoultech.ac.kr (J.H.P.); ysjeong@dongguk.edu (Y.S.J.)

Academic Editor: Doo-Soon Park
Received: 24 November 2016; Accepted: 4 January 2017; Published: 9 January 2017

Abstract: Infrastructure as a service with desktops (DIaaS) based on the extensible mark-up language (XML) is herein proposed to utilize surplus resources. DIaaS is a traditional surplus-resource integrated management technology. It is designed to provide fast work distribution and computing services based on user service requests as well as storage services through desktop-based distributed computing and storage resource integration. DIaaS includes a nondisruptive resource service and an auto-scalable scheme to enhance the availability and scalability of intra-cloud computing resources. A performance evaluation of the proposed scheme measured the clustering performance time for surplus resource utilization. The results showed improvement in computing and storage services in a connection of at least two computers compared to the traditional method for high-availability measurement of nondisruptive services. Furthermore, an artificial server error environment was used to create a clustering delay for computing and storage services and for nondisruptive services. It was compared to the Hadoop distributed file system (HDFS).

Keywords: resource management; dynamic scheduling; scalability; availability; intra-cloud; infrastructure as a service; cloud computing

1. Introduction

To enable more efficient information technology systems, cloud computing has been introduced to corporations, schools, and public organizations. Although traditional on-premise computing environments continue to be built, the need remains for improving the total cost (TC) of ownership and maintenance costs of cloud systems [1–8]. The only solution to this problem, however, is to discard surplus resources, which are connected to traditional on-premise systems, or to introduce virtualization technology for all surplus resources. Nevertheless, the latter solution drastically increases the cost of cloud computing [9–19].

In this paper, infrastructure as a service with desktops (DIaaS) is therefore proposed as a resource management system for surplus resources. DIaaS is an extensible mark-up language (XML)-based traditional surplus-resource integrated management technology. It is designed to provide fast work distribution and computing services based on user service requests as well as storage services through desktop-based distributed computing and storage resource integration. The three key objectives of DIaaS are achieving high-performance computing (HPC), high-throughput computing (HTC), and high availability.

The computing services relating to the proposed approach include a resource management scheme that shares the resources of on-premise systems while delivering information, dynamic

resource management, and monitoring mechanisms by analyzing the status of surplus resources. The storage services provide a data storage management mechanism using existing storage to prevent organizational data leaks and strengthen security after the introduction of cloud computing. XML-based metadata are defined for computing and storage services using on-premise resources, which are used to provide resource performance information and clustering techniques based on distance for integrated resource management.

Furthermore, DIaaS provides nondisruptive resource services and an auto-scalable scheme to enhance the availability and scalability of intra-cloud computing resources. A nondisruptive resource service mechanism is provided to cope with limitations in the computing service job allocation and storage job allocation. An automatic resource allocation mechanism is provided to request computing and storage logs for the automatic expansion of resources, as well as for computing and storage.

A performance evaluation of these mechanisms was conducted to measure the clustering performance times for surplus resource utilization. The results showed improvements in computing and storage services for the connection of at least two computers compared to the traditional method for high-availability measurements of nondisruptive services. Furthermore, an artificial server error environment was developed to create clustering delays for computing and storage services and for nondisruptive services. It was compared to the Hadoop distributed file system (HDFS).

2. Related Works

Traditional surplus resources are used in DIaaS. In this section, traditional distributed resource integration management and clustering techniques for distributed resources are explained. The most widely used systems are first discussed. These include the network file system (NFS), common Internet file system (CIFS), Google file system (GFS), HDFS, Andrew file system (AFS), and owner-based file system (OwFS) [16–22]. The main characteristics, strengths, and weaknesses of these systems are outlined in Table 1 [16–19].

Table 1. Distributed resource integrated management schemes.

Scheme	Characteristics	Strengths	Weaknesses
NFS/CIFS	File system similar to a local	High-speed save	Less extension, high cost
GFS/HDFS	Distributes large-scale files	Low cost	Manual system recovery when errors occur
AFS	Low server load balancing Independent operations for local file systems	High scalability and strong security	Less extension
OwFS	Saves metadata and data information to "owner" Stable when saving the positions of errors	Low cost	Load-balancing limitation when large-scale files are saved

HDFS: Hadoop distributed file system; NFS: network file system; CIFS: common internet file system; GFS: Google file system; AFS: Andrew file system; OwFS: owner-based file system

Existing studies on distributed resource clustering techniques and clustering visualization are outlined in Table 2 [23–27].

Table 2. Distributed resource clustering and visualization schemes.

Schemes	Setup Method	Visualization	Clustering Organization	Simulation
VCS simulator [24]	Text-based	Text	User-defined	Yes
GridSim [25,26]	Modified source code	Chart	None	Yes
ClusterSim [27]	Clustering-based	Text	User-defined	Yes
CloudSim [28]	Text-based	Text	None	Yes
DIaaS	Graph-based	Graph	User-defined	Yes

VCS: Veritas Cluster Server; DIaaS: Infrastructure as a service with desktops

3. On-Premise Intra-Cloud Resource Management Scheme

In this section, the integrated resource management scheme for resource pool formation is detailed. The unused-resource clustering technique is proposed for job allocating based on computing and storage job requests and nondisruptive resource services. In addition, an auto-scalable scheme for the operation of surplus resources is presented.

3.1. On-Premise Integrated Resource Management

DIaaS requires the stratification of distributed resource integration for job requests, identification, allocation, and performance for desktop-based computing, and storage services. As shown in Figure 1, the distributed integrated resource management scheme consists of desktop resource nodes (DRNs), which are metadata-based, on-premise resources, desktop slave nodes (DSNs), which manage DRNs, and desktop manager nodes (DMNs), which control DSNs. Here, metadata means the minimum unit of the resource information for on-premise resources. It refers to resource specifications, settings, and status information.

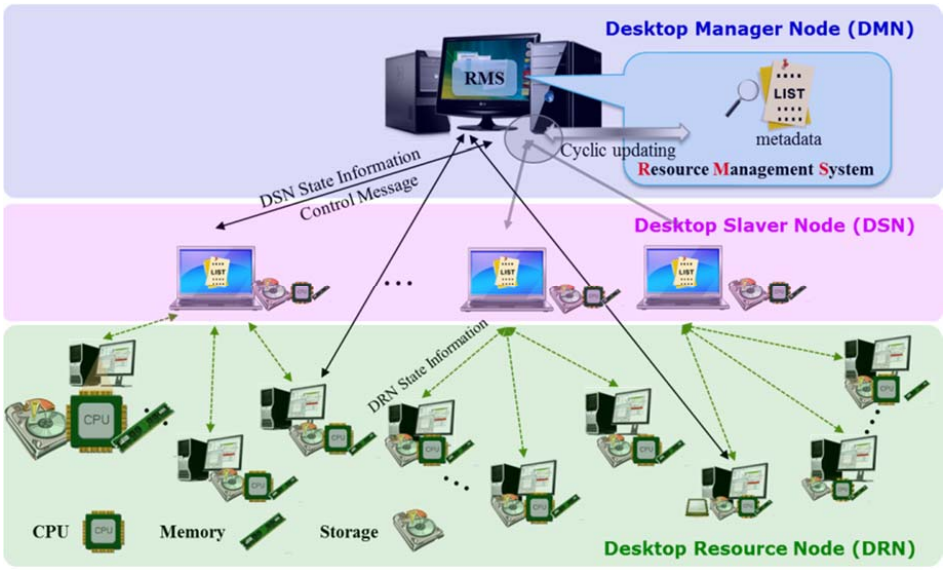


Figure 1. Conceptual scheme for on-premise integrated resource management.

When the on-premise-based integrated resource management scheme requests computing and storage resources from the desktop manager node (DMN), it searches the metadata and provides the resources to which jobs are allocated. This process is illustrated in Figure 2. The step-by-step operations for resource request management shown in Figure 2 are outlined below.

Step 1 The user requests computing or storage resources of the DMN through the client program.

Step 2 Using metadata, the DMN resource management system determines whether to accept the request depending on the requested computing and storage size.

Step 3 When the request is accepted, the location of the resource to process the job is selected. The selected resource information is sent as a control message through the desktop slave node (DSN) for direct connection with the client program.

Step 4 The client information is included in the control message sent to the service resource, and the job performance is prepared.

Step 5 The DMN sends the resource information to perform the job as a response message to the client program.

Step 6 The client program and the job-performing desktop resource node (DRN) execute the job through a direct connection.

Next, integrated resource management, desktop, computing service, and storage service metadata are defined. The resource management metadata structure of the DIaaS is shown in Figure 3.

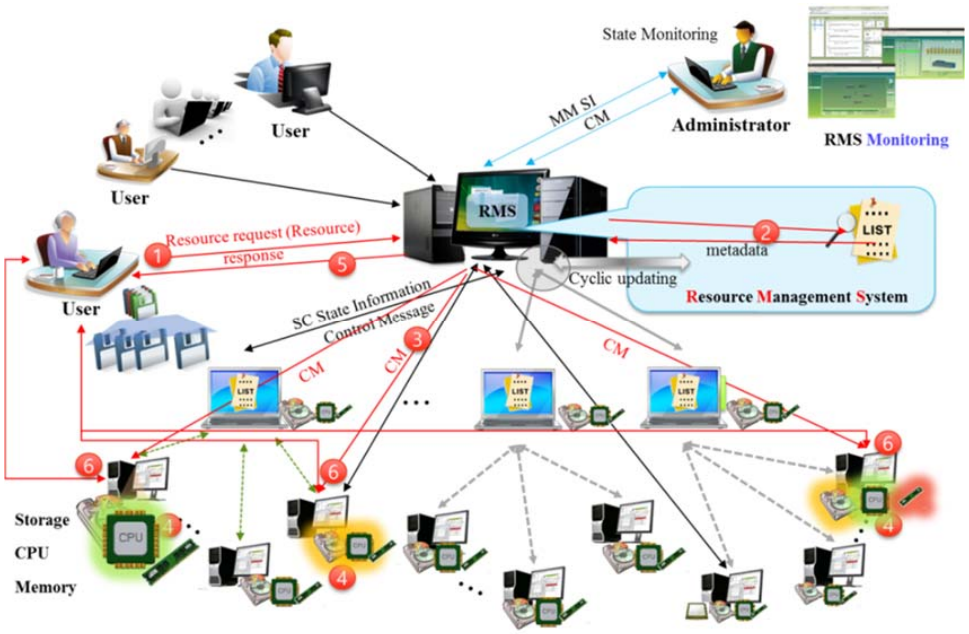
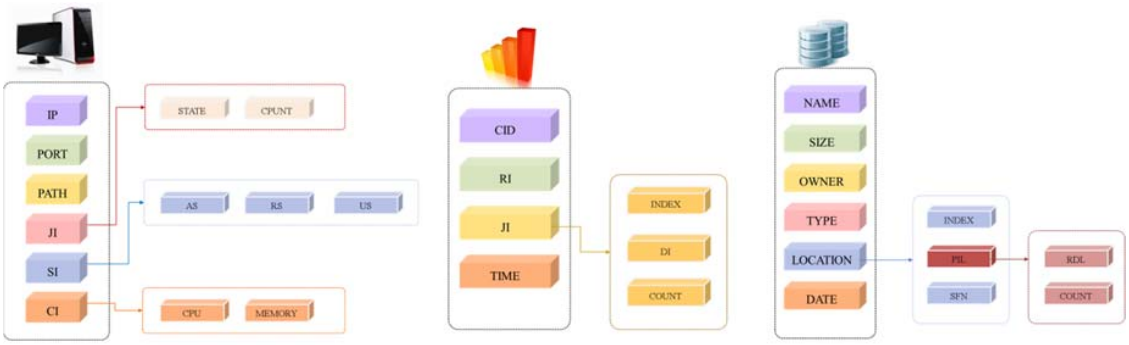


Figure 2. Operational structure for resource requests on the infrastructure as a service with desktops (DIaaS).



(**a**) (**b**) (**c**)

Figure 3. Metadata structure of the DIaaS: (**a**) Desktop metadata; (**b**) Computing service metadata and (**c**) storage service metadata.

Figure 3a includes the performance status metadata of the physical desktops that constitute the resource pool, job information, total number of jobs, and job status information, such as standby, working, and stopped. The attributes of the desktop metadata are the connected desktop (IP); Connected port number of the desktop (PORT); Storage sharing location of the connected desktop (PATH); Job information relating to data storage and duplication, including the job performance status and job progress status (job information (JI)); Permissible total size, remaining volume, and used volume of the connected desktop storage (storage information (SI)); Central processing unit (CPU) and memory status information of the connected desktop (computing information (CI)); Job status indicated by standby, working, or stopped (STATE); Total number of jobs, including working jobs and standby jobs (COUNT); Total allowed storage size of the connected desktop (allowed size (AS)); Remaining storage space available for the AS (residual size (RS)); Used storage size in the AS (usage size (US)); Used CPU size of the connected desktop (CPU); and used memory size of the connected desktop (MEMORY).

The computing service metadata include the job performance, unique identifications for computing jobs, and job requests. The attributes corresponding to these metadata are the unique identity of each computing job (computing identity (CID)); Requesting client's information (requester information (RI)); Parallel performance information, such as the job performing desktop and total number of desktops for the division of jobs (JI); Job requests and performance times (TIME); Index of a divided computing job (INDEX); Information from the desktop performing the computing job (desktop information (DI)); and number of resources participating in the divided job (COUNT).

The storage service metadata include the information on the stored data; basic information, such as the name, size, and type of stored data; JI, the division information for efficient storage in the distributed on-premise storage resources; and duplicated storage information for the integrity of the stored data. The attributes are the original data name (NAME); Original data size (SIZE); Storage service requester (OWNER); Data type (TYPE); Location of the stored data (LOCATION); Date on which the data were saved (DATE); Index of the divided block data (INDEX); Location of the duplicated stored data of the divided block data (path information list (PIL)); Split unique block data name for integration of data stored in a distributed manner in a data-request event (split file name (SFN)); Location of the data divided by the duplicate storage set by the user (redundant data location (RDL)); and number of divided files duplicated in storage for data integrity (COUNT). Among these attributes, INDEX, PIL, and SFN contain the sub-metadata of LOCATION, while RDL and COUNT contain the sub-metadata information of PIL by default.

3.2. Clustering Scheme for Integrated Resource Management

The computing service of DIaaS provides appropriate job allocation through on-premise-based clustering. For the on-premise-based integrated resource clustering elements, mark-up languages (XMLs) are defined, as shown in Figure 4. They are defined based on performance and distance information, such as the CPU, memory, and storage of the desktop metadata shown in Figure 3a. For the basic clustering elements, distance and performance are used. If accomplishing a time measurement by pinging is impossible on account of the network settings, the request time of the operating desktop is used. For performance, the dynamic performance factors, including the idle CPU performance, main memory, and remaining storage size, as well as the static performance factors—including the basic CPU performance, memory, and storage size of the resource—are used.

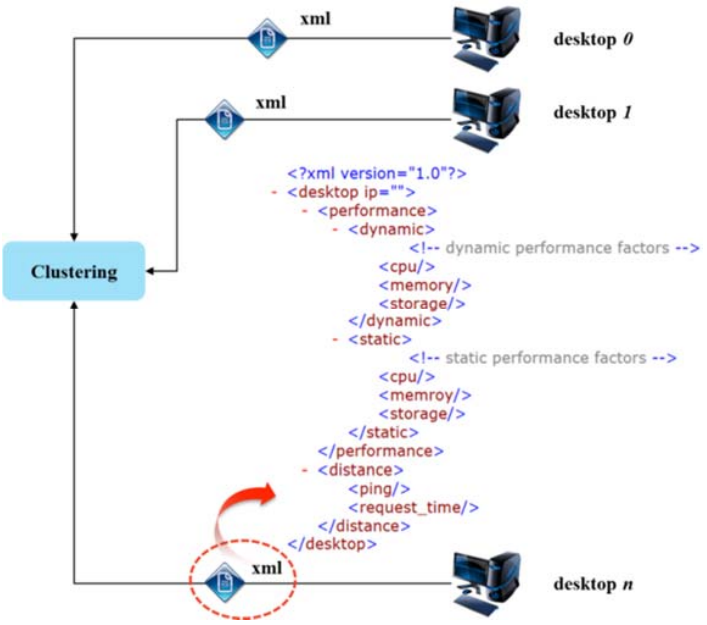


Figure 4. Extensible mark-up language (XML) structure for clustering with on-premise resources.

The detailed clustering process is outlined as follows. ① START; clustering starts when the desktop count is greater than the cluster count. ② CREATE CLUSTER; the user selects the initial cluster and defines the number of separate clusters to select random clusters. The k created clusters form the initial cluster set, C{}. ③ CLUSTERING; the distance to each desktop is measured with the initial cluster set C, which was created in the CREATE CLUSTER step. The closest desktop is included. This is repeated until all desktops connected to the on-premise-based integrated resource are included in the cluster set. ④ CHECK CLUSTER CENTER VALUE; the center of the distance value of the desktops in each cluster of cluster set C is calculated. The location of the closest desktop is selected as a new cluster for the corresponding set. Then, the location coordinates of the new cluster and present cluster are compared. UPDATE CLUSTER is performed if these coordinates do not coincide; otherwise, the process ends (END). ⑤ UPDATE CLUSTER; the new cluster determined in Step ④ is applied to the corresponding cluster set. The CLUSTERING step is performed until the proximities of all desktops to the cluster are again calculated. This is repeated until there is no change in the cluster center value in Step ④. ⑥ END; the metadata obtained after the completion of clustering are applied to each desktop.

3.3. Nondisruptive Computing and Storage Service Scheme

When the computing and storage resource services of DIaaS are requested, the nondisruptive computing and storage service scheme is included to cope with hardware defects or exceptions, such as server downtime. It can also cope with exceptions due to the inoperable state of the DMN managing the on-premise resource pool.

The DMN that generally controls integrated resource management searches the desktop that is performing the job based on metadata. It performs this task when it receives a computing and storage service request. The searched desktop performs the job in accordance with the computing and storage service request, and the DMN updates the job information and performance information of the desktop in the metadata. When an unexpected error occurs and the DMN cannot operate the server, the jobs running in the subordinate DSNs and DRNs, as well as the stored data, are lost. To provide continued service in such instances, the following steps are performed.

Step 1 The first on-premise resource is specified as the DMN. The added resources are limited to DSN and DRN roles. The same server functions of the DMN are included in the DSN and DRN.

Step 2 The DSN and DRN send metadata, including the integrated storage resource settings, as well as CPU and main memory performance information, to the DMN.

Step 3 The DMN creates a substitute server candidate list of metadata on the IP address, performance, stored data amount, and distance criteria. The stored data amount is used as the top priority by default to minimize computing waste and network bandwidth due to duplicated internal data.

Step 4 The substitute server candidate list metadata are synchronized with the connected desktops, and Steps 1–3 are repeated to add resources.

For the DMN substitute selection, priorities are given on the basis of the IP address system, excluding the routing information. The performance of each resource is set so that $\alpha + \beta$ becomes 100% for CPU (α) and main memory (β). The α value is increased if the processing speed improvement is significant, while the β value is increased if the resource utilization count is significant.

For stored data amount, the resource of the smallest size is selected as the DMN substitute after normalization of all the connected resources. It is based on the resource with the minimum size of total stored data. The response time, based on the periodic heartbeat, is applied to the substitute server settings, with the reference distance set as the normalized value based on the shortest response time. After the initial DIaaS resource clustering process, the DSN role settings achieved through the Steps in Figure 5 are required for allocating the computing and storage services.

In Figure 5, Step 1, the distance (x-axis) is normalized by:

$$\text{Desktop}N_{x\text{Value}} = \frac{\text{Desktop}N_{\text{distance}}}{\text{Desktop}_{\text{maxDistance}}} \times 100 \quad (1)$$

the computational capacity (y-axis) is normalized by:

$$\text{Desktop}N_{y\text{Value}} = \frac{\text{Desktop}N_{\text{cpu}}}{\text{Desktop}_{\text{maxCPU}}} \times \alpha + \frac{\text{DesktopN}_{\text{memory}}}{\text{Desktop}_{\text{maxMemory}}} \times \beta \quad (2)$$

and the storage capacity (z-axis) is normalized by:

$$\text{Desktop}N_{z\text{Value}} = \frac{\text{Desktop}N_{\text{storage}}}{\text{Desktop}_{\text{maxStorage}}} \times 100 \quad (3)$$

Steps 2–6 are performed based on the detailed clustering process. For allocating the computing and storage services, the service allocation amount is:

$$\text{DSN}_{\text{workload}_i} = \frac{\text{DSN}_i}{\sum_{k=0}^{n-1} \text{DSN}_{k\text{CPU}}} \times \text{TotalWorkLoadSize},\ i = 0 \sim N-1 \quad (4)$$

when the idle CPU speed for N DSNs is:

$$\{\text{DSN}_{0\text{CPU}},\ \text{DSN}_{1\text{CPU}},\ \text{DSN}_{2\text{CPU}},\ \ldots .,\ \text{DSN}_{N-1\text{CPU}}\}$$

In this case, the idle CPU speed of each J DSN is calculated using:

$$\text{DSN}_{i\text{CPU}} = \sum_{k=0}^{n-1} \text{DRN}_{k\text{CPU}},\ i = 0 \sim J-1 \quad (5)$$

The DRNs of Equation (5) refer to the resources that belong to each DSN. Accordingly, the idle speed of M DRNs becomes:

$$\{\text{DRN}_{0\text{CPU}},\ \text{DRN}_{1\text{CPU}},\ \text{DRN}_{2\text{CPU}},\ \ldots \ldots,\ \text{DRN}_{M-1\text{CPU}}\}$$

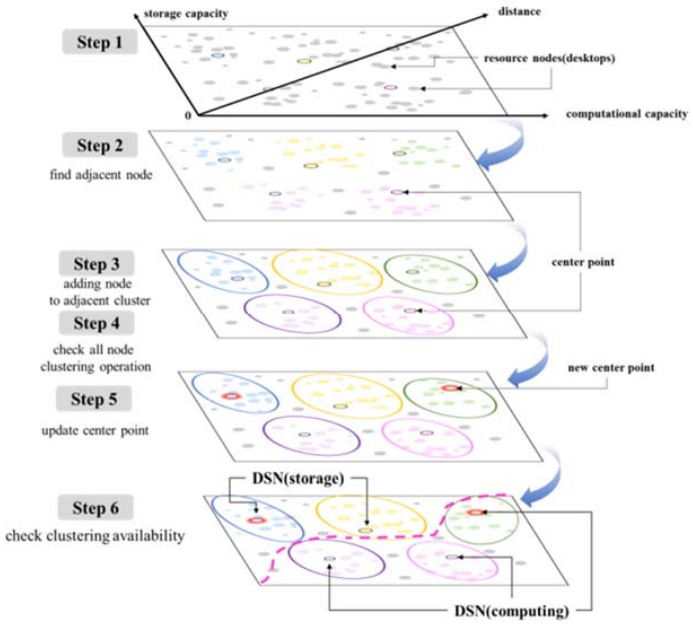


Figure 5. The desktop slave node (DSN) role defined for allocation of computing and storage service.

The DRN CPU speed is calculated using the difference between the CPU occupation value of the used process and the static CPU value. For example, when one DRN has a static CPU value of 3.4 GHz,

and if the used process occupies 10% of the CPU, it is calculated as 3.4 GHz − 0.34 GHz = 3.06 GHz. Because the dynamic idle CPU speed value is fluid, the idle CPU speed value can change in the resource state using the DRN.

In this paper, resources were allocated in a hierarchical manner based on the updated CPU information for each computing service request, as shown in Figure 6. In ① of Figure 6, a new divided computing service is allocated in proportion to the CPU speed of the current DSN according to Equation (4). The DSN is allocated by subdividing the divided computing service in the DRN to which it belongs.

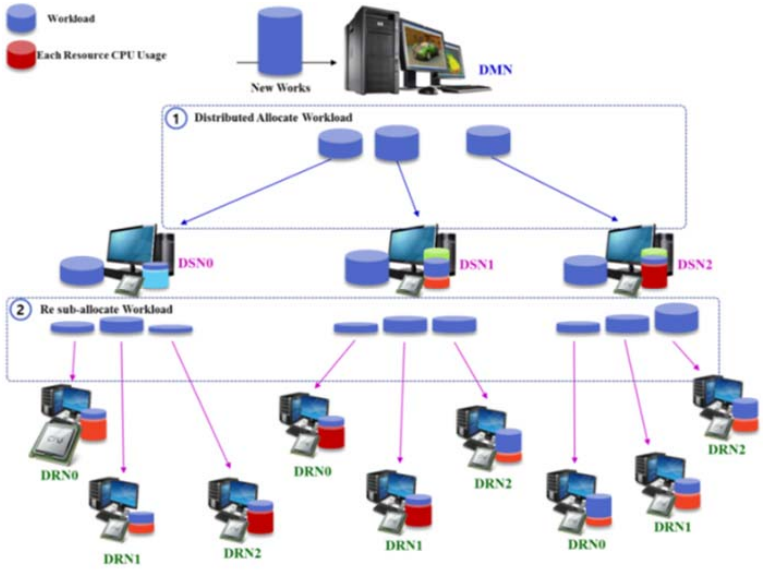


Figure 6. Hierarchical computing service of DIaaS.

When the DSN or DRN has an error in the DIaaS, the fault tolerance for the computing service failover is obtained as illustrated in Figure 7.

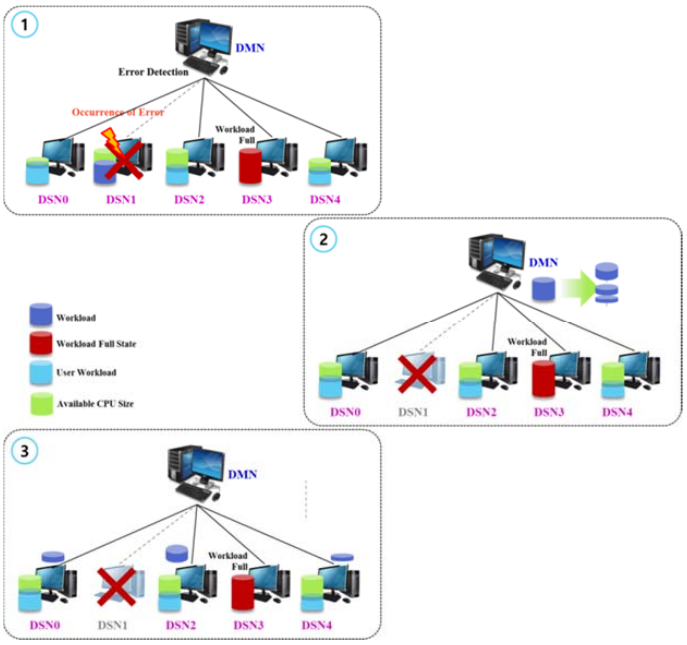


Figure 7. Computing service failover scheme.

In ① of the figure, a fault situation for DSN1 of the DMN is shown. The DMN divides the computing service as shown in ②, except for DSN3, to which no service can be assigned. Moreover, it considers the resource conditions of DSN0, DSN2, DSN3, and DSN4. Finally, continued service is performed by re-allocating resources to DSN0, DSN2, and DSN4, as shown in ③ of Figure 7.

The storage services of DIaaS are divided into storing data and requesting stored data. In the case of storing data, the data to be stored is divided into 64 MB chunks, as shown in the HDFS [20,21] and GFS [20,21]. Furthermore, the chunk size can be changed by a user setting in DIaaS. Figure 8 shows the process of allocating storage resources to the DSN, which plays the storage service role in the DIaaS.

In ① of Figure 8, the storage requester solicits the storage service of the DMN. In ②, the DMN sends the metadata of the DSN that can provide the storage service to the storage requester. It simultaneously sends to the DSN the metadata for the IP address and port number to be connected with the storage requester. The storage requester awaits a "ready" message from the DRN to implement a connection trial to the DRN for data storage.

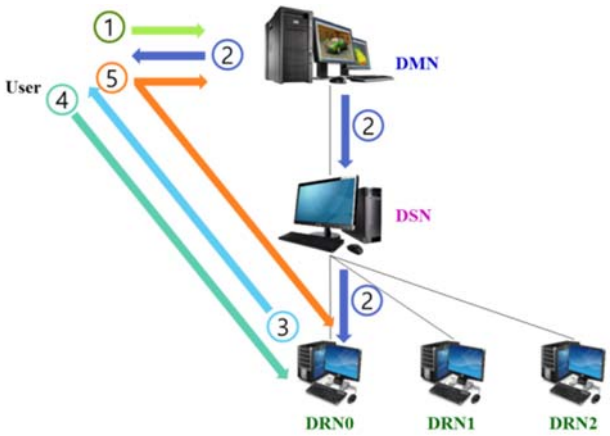


Figure 8. Storage service processing of DIaaS.

In ③, the DRN that is ready for the storage service sends a ready message to the storage requester. In ④, the DRN is connected with the storage requester and sends the data. After all data to be stored are sent, the transmission complete state is sent in ⑤, and the storage job is finished.

The following rules are applied to the storage job to ensure data integrity and permanence. First, the data stored in chunk units cannot be edited for data integrity. In case the file requires modification, it must be saved as a new file. Second, data stored in chunk units are stored in duplication to maintain three internal data chunks. Duplicate storage offers data permanence when one or two data chunks are maintained on account of an error. This storage process maintains duplicate storage in the original data chunk in a DRN, specifically inside the same DSN of the original data chunk, and in a DRN, specifically inside a DSN that is different.

Errors that occur during the storage service are handled by a four-step failover, as shown in Figure 9. In ① of Figure 9, the storage service normally operates. In ②, an error is generated and detected. In ③, the storage service requester receives new DRN information from the DSN. In ④, data storage is completed through the normal storage service.

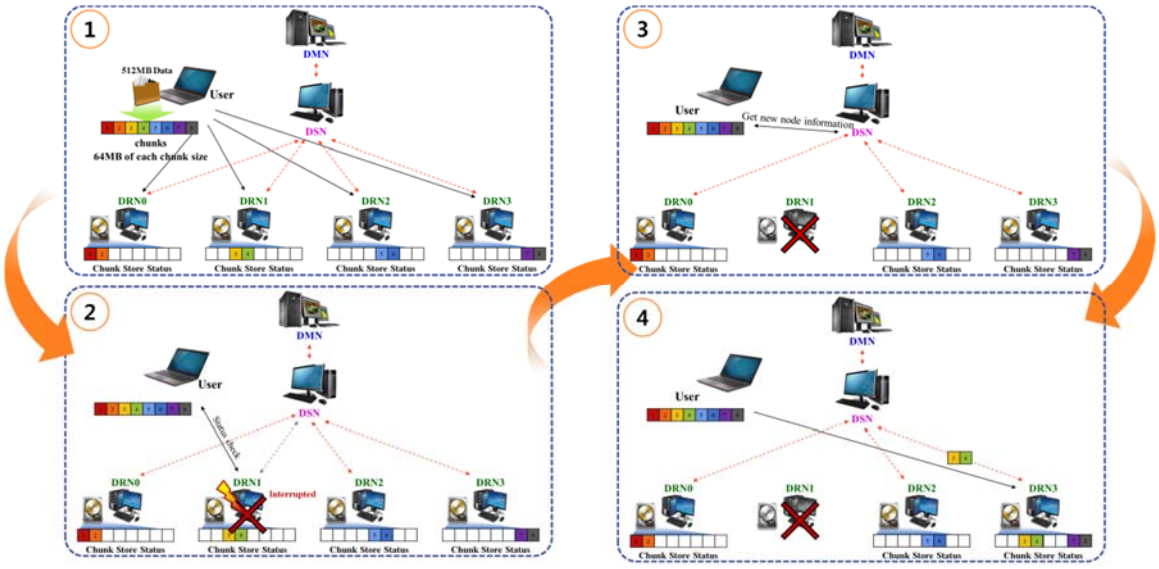


Figure 9. Storage service failover scheme.

3.4. Auto-Scalable Scheme

The DMN performs computing and storage services depending on the job. Moreover, it defines an auto-scalable scheme to provide resource services, such as flexible computing and storage, according to the same service request jobs. The goal is to minimize resource waste and improve service reliability in the intra-cloud, which provides services with limited resources.

First, the computing service auto-scheme defines the automatic expansion method according to a computing log. It is automatically expanded, as shown in Figure 10a. Accordingly, idle computing resources are held in the time slot in which the same computing service is requested. In ① of Figure 10a, the requested time for the computing service and the specific requested computing service are shown as a computing service log.

In ② of Figure 10a, automatic expansion is performed to ensure a smooth computing service in ③ at the requested time for providing the computing service to the computing service requester. The latter requests the computing service, C4. The storage service also logs the storage use the starting time with log-based automatic expansion. It analyzes the continuously used storage space, as shown in Figure 10b. The required size of the storage space is determined by analyzing the storage size required by the user based on the day or month.

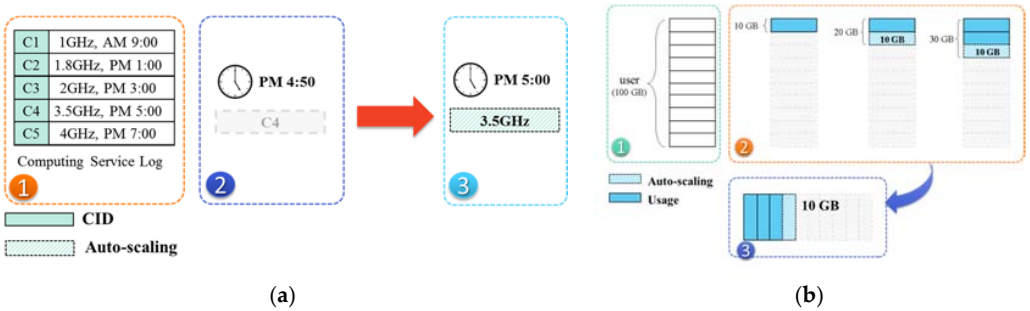


Figure 10. Automatic scalability scheme of DIaaS: (**a**) Computing service and (**b**) storage service.

4. Infrastructure as a Service with Desktops (DIaaS) Design

The DIaaS is composed of a user interface for user controls, including the on-premise resource setting, desktop information, and control information; The resource-integrated manager for

distributed on-premise resource integration; Clustering manager for hierarchical resource management; Nondisruptive manager for continuous computing and storage support services; Auto-scalable manager for minimization of resource waste in the intra-cloud; and viewer for visualization of the on-premise resource management operations (Figure 11).

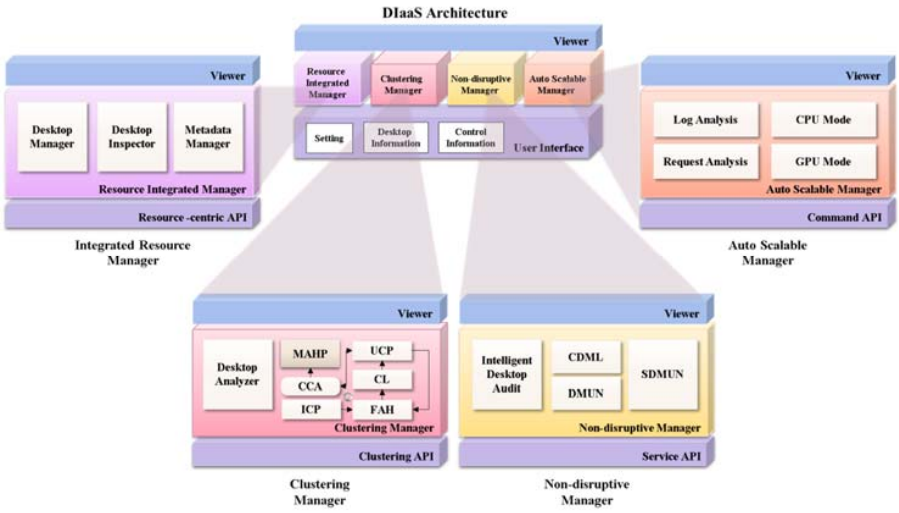


Figure 11. Overall architecture of DIaaS.

The integrated resource manager is subdivided into the desktop manager, desktop inspector, and metadata manager. The desktop manager connects a desktop through the resource-centric application programming interface (API) and periodically monitors the connection status of the desktop through heartbeats. Furthermore, it manages the on-premise resource availability statuses, such as the CPU, memory, and storage, as well as the currently operating desktop and other desktops. This management information is sent to the metadata manager after the information for the desktop connected to the desktop manager is analyzed through the desktop inspector. The metadata manager creates desktop metadata through the desktop information received from the desktop inspector. It delivers or updates information according to the request from the Desktop Manager.

The clustering manager is composed of the desktop analyzer and clustering function. The desktop analyzer analyzes desktop information for clustering, which it then indicates in the two-dimensional x- and y-axes, and the three-dimensional x-, y-, and z-axes. For two-dimensional clustering, the x-axis indicates the performance, such as the CPU, memory, and storage; and the y-axis indicates the distance, which is the response time of each desktop. For three-dimensional clustering, the x-axis is defined as the computing CPU and memory, the y-axis is the storage, and the z-axis is the distance, which has the same definition as that in two-dimensional clustering.

Normalization is performed based on the highest performing desktop information. Clustering performs an initialization center point (ICP); It creates random clusters according to the cluster in line with the number of clusters set by the user through the clustering API. In addition, it finds the adjacent host (FAH), that is, it finds adjacent desktops based on the random cluster that has been created. The desktops identified through the FAH process are added to the cluster list (CL).

After all desktops perform this process, they are moved to a new center point based on the desktop added to each cluster through the update center point (UCP) step. Then, the FAH process is repeated to move to the optimum center point. Additional performance or lack of it is determined through check clustering availability (CCA). When additional performance through CCA becomes unnecessary, the desktop that is closest to the center is selected as a cluster based on the move adjacent host point (MAHP).

The nondisruptive manager includes detailed functions, such as intelligent desktop audit, create desktop metadata list (CDML), desktop metadata update notify (DMUN), and stored data metadata

update notify (SDMUN). The intelligent desktop audit receives the list of connected desktops from the desktop manager of the resource integrated manager through the service API. It creates metadata for the selected DMN substitutes. The created DMN metadata are conveyed to the DMUN function. The DMUN function performs synchronization by sending the DMN metadata to all connected desktops. In addition, the SDMUN function sends the storage service metadata, including the storage location of data received from the user, to all desktops for synchronization. This method can be continuously operated using the DMN metadata and data stored in the DSN, while selecting the optimum DSN in the event of a DMN error. Furthermore, the intelligent desktop audit function sets the DSN role for the allocation of on-premise-based computing and storage services.

The auto-scalable manager is subdivided into functions that include log analysis, request analysis, CPU mode, and graphic processing unit (GPU) mode. The log analysis logs the provided resources, times, and completion times of the computing and storage services. The request analysis evaluates the average resource request and time of the daily or monthly resource usage for the computing and storage services. The CPU mode is the basic computation mode, which is performed using the CPU. The automatic expansion performance time is minimized by operating in GPU mode, specifically if it can be performed by assessing the DMN GPU function activation. Furthermore, the automatic expansion criteria can be changed to day, month, or year through the command API.

5. DIaaS Implementation

This section explains the integrated management scheme for detailed resource pool creation, the resource clustering technique for computing and storage services, and the implementation of the nondisruptive service and auto-scalable scheme for surplus resource operation. The access screen of the DRN for the new resource connection to the DMN running in the DIaaS is configured as shown in Figure 12a.

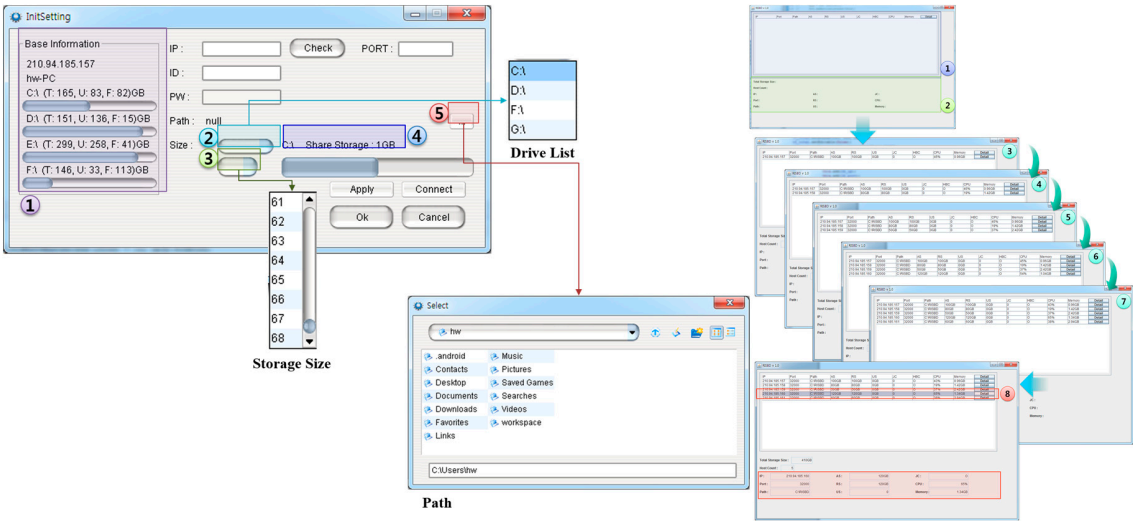


Figure 12. Resource connection of the desktop manager node (DMN) in DIaaS: (**a**) New resource connection to DMN and (**b**) sequence connection of DMN.

In ① of Figure 12a, the basic information of the resource attempting to connect to the DMN is shown. It includes the IP address, host name, and storage drive setting. Here, T denotes the total storage size, U represents the usage storage size, and F is the free storage size. In ② of the same figure, the storage drive allocated to the DMN is set. In ③, the storage allocated for the DMN is set. In ④, visualization information is provided for the drive set by the user and the storage size. In ⑤, the path for sharing the storage in the DMN is set through the activated path dialogue.

Figure 12b shows the sequential connection process among the five resources in the DMN. In ① of the figure, the connected resources list is shown. In ②, visualization of the integrated computing and storage information of the connected resources is provided. In ③, ④, ⑤, ⑥, and ⑦, the basic information of each resource is shown, such as the IP address, port number, path, AS, RS, US, JC, heartbeat check (HBC), CPU, and memory. In ⑧ of the same figure, information is provided that details when a specific resource is selected, thereby indicating a complete connection to the DMN. The clustering of desktops connected to the DIaaS DMN is internally performed.

In this paper, the actual clustering was verified through the development of a simulator. Figure 13 shows 1000 random desktops, the settings of 10 clusters, and the clustered resource information, which are the results of the simulation. In ① of Figure 13, the information from the resources selected by the user for a DSN cluster is presented. In ②, cluster information is provided to the user for IP-based identification when a cluster DSN is selected. In ③, the information of the DRN, a desktop included in the DSN, is provided.

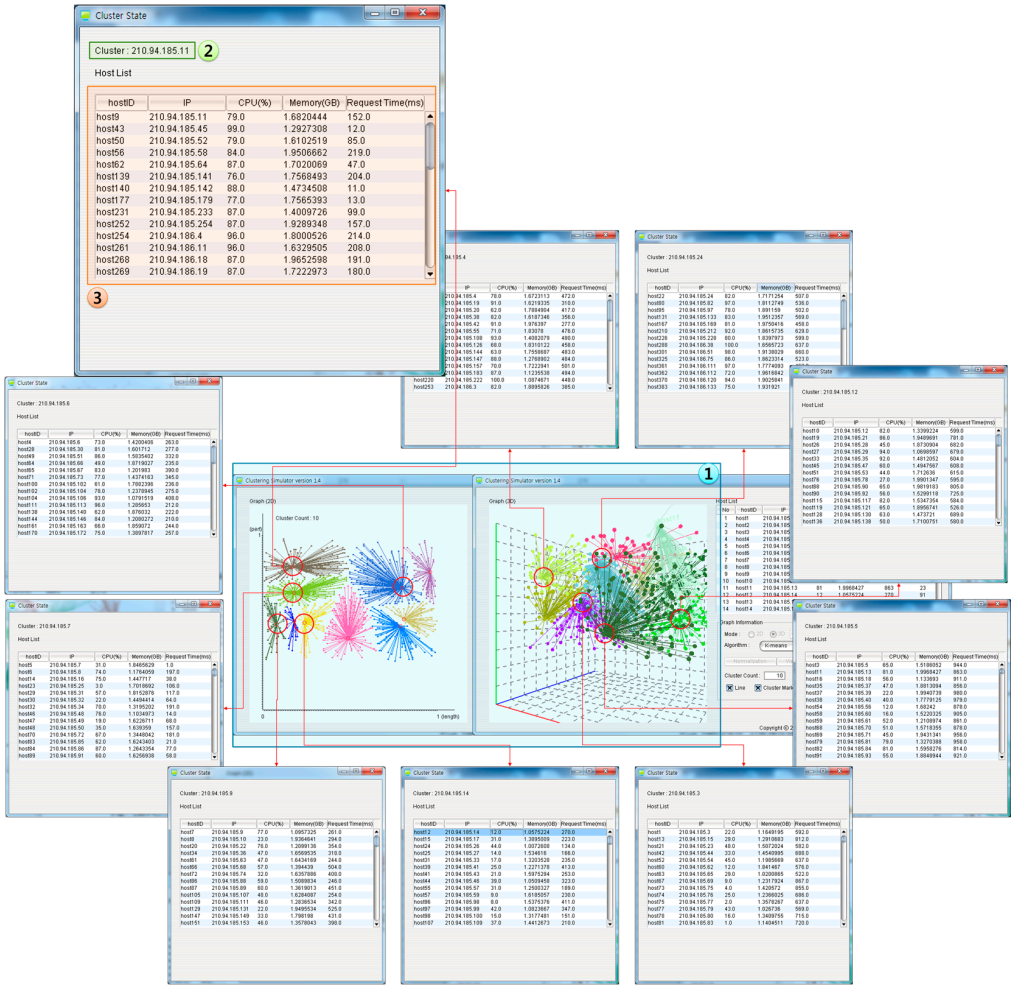


Figure 13. Verification of resource clustering in DIaaS.

For the nondisruptive service following a DMN error, the DMN and the four DSNs beneath it are connected, as shown in Figure 14. When a DMN error occurs, the first DSN in the candidate DMN list is replaced with the DMN. In ① of Figure 14, a case in which the DMN normally operates is shown. In ② of the same figure, the connections from DSN0, DSN1, DSN2, and DSN3 to the DMN are released as a result of a DMN error. In ③, the substitute server candidate of the first priority is determined based on the metadata from the candidate DMN list. In ④, the DSNs, excluding the DSN selected as

the DMN, are connected to the new DMN. After the connection, the new candidate DMN list is sent to every DSN through the new DMN for synchronization.

Figure 14. Nondisruptive service scheme of the DMN in DIaaS.

The auto-scalable scheme is implemented by the log, and requests computing and storage services. The basic operation process is shown in Figure 15. The required resources are automatically expanded according to the resource requested by the desktop. When 30 GB of storage are provided, 10 GB are used during the initial stage, and 10 GB are provided during the next stage through automatic expansion.

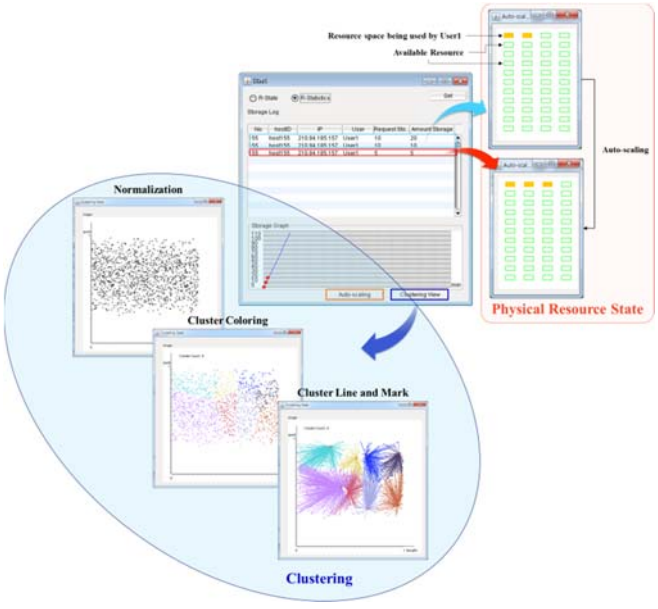


Figure 15. Auto-scalable scheme of DIaaS.

6. Performance Evaluation

For the performance evaluation, the DIaaS created an artificial server error environment for testing the clustering delay time and nondisruptive service for computing and storage services. First, to measure the clustering delay time, the number of desktops was increased to 100, 200, 300, 400, 500, 1000, 2000, 3000, 4000, and 5000, as shown in Figure 16. Clustering was performed after the number of clusters was increased to 5, 10, 15, 20, 25, and 30 at each increasing point. The clustering speed at each increasing point was determined by averaging the performance speeds obtained from 50 cycles. The definition of cycle is the average value of execution number for each clustering. It required more than 10 s to accomplish this task for 1000 desktops, and 21 s when clustering was performed using 30 clusters and 5000 desktops. As shown in Figure 15, this process was more efficient and cost effective than the clustering test using 5000 actual desktops and the related cost of labor, electricity, and time.

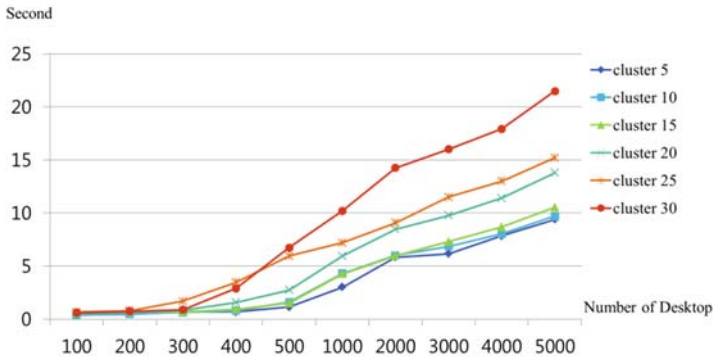


Figure 16. Number of workloads with clusters and desktops.

Figure 17 shows the connection of 10 desktops and the operation status when the master node has an error with the DIaaS and HDFS. In this case, the SecondaryNode function is performed when NameNode contains an error. As a result, eight DataNodes stably operate. However, continuous error occurrences result from the situation in which DataNodes are not operating. In the case of the DIaaS, every connected desktop has embedded server functions and synchronizes the data storage information and server priority list. It thus enables a continuous server operation, even when errors occur. This feature is supported by the connection of a minimum of two desktops implementing the DMN and DSN. This sustainable operation can provide high availability for server errors in an environment in which several thousand desktops are connected. DIaaS can service continuously the operation as server better than HDFS.

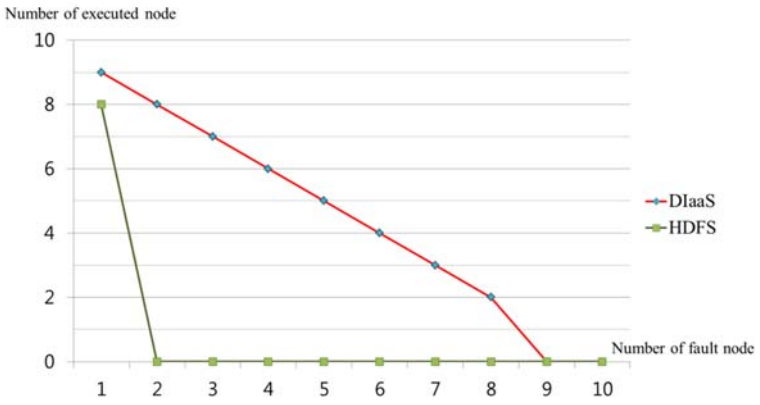


Figure 17. Fault tolerance for the master node error.

7. Conclusions

In this paper, a DIaaS was proposed that includes an integrated resource management scheme for computing and storage services using surplus resources by default. XML-based metadata were defined for computing and storage services using on-premise resources. Moreover, resource performance information and distance-based clustering techniques were proposed for integrated resource management using the metadata. Furthermore, a nondisruptive resource service and auto-scalable scheme were developed to enhance the availability and scalability of the intra-cloud computing resources.

In a performance evaluation, the clustering performance time for surplus resources was utilized. The high-availability measurement of the nondisruptive service showed improved results for computing and storage services with a connection of at least two desktops compared to the traditional method. In addition, an artificial server error environment was created to test the clustering delay time for computing and storage services, and the nondisruptive service was compared to the HDFS.

Acknowledgments: This research was supported by BK21 Plus project of the National Research Foundation of Korea Grant. And also this research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2016-H8501-16-1014) supervised by the IITP (Institute for Information & communications Technology Promotion).

Author Contributions: All the authors contributed equally to this work. All authors 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. [CrossRef]
2. Jararweh, Y.; Al-Ayyoub, M.; Darabseh, A.; Benkhelifa, E.; Voukc, M.; Rindos, A. Software defined cloud: Survey, system and evaluation. *Future Gener. Comput. Syst.* **2016**, *58*, 56–74. [CrossRef]
3. Vasoya, S.; Gadhavi, L.; Bhatia, J.; Bhavsar, M. Resource provisioning strategies in cloud: A Survey. *Int. J. Comput. Sci. Commun.* **2016**, *7*, 12–15.
4. Alam, M.I.; Pandey, M.; Rautaray, S.S. A comprehensive survey on cloud computing. *Int. J. Inf. Technol. Comput. Sci.* **2015**, *7*, 68–79. [CrossRef]
5. Guzek, M.; Bouvry, P.; Talbi, E.G. A survey of evolutionary computation for resource management of processing in cloud computing. *IEEE Comput. Intell. Mag.* **2015**, *10*, 53–67. [CrossRef]
6. Suralkar, S.; Mujumdar, A.; Masiwal, G.; Kulkarni, M. Review of distributed file systems: Case studies. *Int. J. Eng. Res. Appl.* **2013**, *3*, 1293–1298.
7. Asadianfam, S.; Shamsi, M.; Kashany, S. A review: Distributed file system. *Int. J. Comput. Netw. Commun. Secur.* **2015**, *3*, 229–234.
8. Yadav, J.S.; Yadav, M.; Jain, A. Distributed file system. *Int. J. Sci. Res. Educ.* **2013**, *1*, 126–134.
9. Bera, S.; Misra, S.; Rodrigues, J.J.P.C. Cloud computing applications for smart grid: A survey. *IEEE Trans. Parallel Distrib. Syst.* **2015**, *26*, 1477–1494. [CrossRef]
10. Patel, I.; Shah, B. Survey on resource allocation technique in cloud. *Int. J. Sci. Res.* **2016**, *5*, 232–235.
11. Jackson, J.C.; Vijayakumar, V.; Quadir, M.A.; Bharathi, C. Survey on programming models and environments for cluster, cloud, and grid computing that defends big data. *Procedia Comput. Sci.* **2015**, *50*, 517–523. [CrossRef]
12. Ranjan, R.; Benatallah, B.; Dustdar, S.; Papazoglou, M.P. Cloud resource orchestration programming. *IEEE Internet Comput.* **2015**, *19*, 46–56. [CrossRef]
13. Zhan, Z.-H.; Liu, X.-F.; Gong, Y.-J.; Zhang, J.; Chung, H.S.-H.; Li, Y. Cloud computing resource scheduling and a survey of its evolutionary approaches. *ACM Comput. Surv.* **2015**, *47*. [CrossRef]
14. Jennings, B.; Stadler, R. Resource management in clouds: Survey and research challenges. *J. Netw. Syst. Manag.* **2015**, *23*, 567–619. [CrossRef]
15. Kukade, P.P.; Kale, G. Survey of load balancing and scaling approaches in cloud. *Int. J. Emerg. Trends Technol. Comput. Sci.* **2015**, *4*, 189–192.

16. Motavaselalhagh, F.; Esfahani, F.S.; Arabnia, H.R. Knowledge-based adaptable scheduler for SaaS providers in cloud computing. *Hum. Centric Comput. Inf. Sci.* **2015**, *5*. [CrossRef]
17. Kim, S.; Lee, H.; Kwon, H.; Lee, S. Evaluation model of defense information systems use. *J. Converg.* **2015**, *6*, 18–26.
18. Kar, J.; Mishra, M.R. Mitigating threats and security metrics in cloud computing. *J. Inf. Process. Syst.* **2016**, *12*, 226–233.
19. Kashyap, D.; Viradiya, J. A survey of various load balancing algorithms in cloud computing. *Int. J. Sci. Technol. Res.* **2014**, *3*, 115–119.
20. 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. [CrossRef]
21. Kim, H.-W.; Park, J.H.; Jeong, Y.-S. Human-centric storage resource mechanism for big data on cloud service architecture. *J. Supercomput.* **2015**, *72*, 2437–2452. [CrossRef]
22. Kumbhare, A.G.; Simmhan, Y.; Frincu, M.; Prasanna, V.K. Reactive resource provisioning heuristics for dynamic dataflows on cloud infrastructure. *IEEE Trans. Cloud Comput.* **2015**, *3*, 105–118. [CrossRef]
23. Lingawar, R.P.; Srode, M.V.; Ghonge, M.M. Survey on load-balancing techniques in cloud computing. *Int. J. Advent Res. Comput. Electron.* **2014**, *1*, 18–21.
24. Magalhães, D.; Calheiros, R.N.; Buyya, R.; Gomes, D.G. Workload modeling for resource usage analysis and simulation in cloud computing. *Comput. Electr. Eng.* **2015**, *47*, 69–81. [CrossRef]
25. De Assunçãoa, M.D.; Cardonha, C.H.; Netto, M.A.S.; Cunha, R.L.F. Impact of user patience on auto-scaling resource capacity for cloud services. *Future Gener. Comput. Syst.* **2016**, *55*, 41–50. [CrossRef]
26. Al-Ayyoub, M.; Jararweh, Y.; Daraghmeh, M.; Althebyan, Q. Multi-agent based dynamic resource provisioning and monitoring for cloud computing systems infrastructure. *Cluster Comput.* **2015**, *18*, 919–932. [CrossRef]
27. Tan, Y.; Xia, C.H. An adaptive learning approach for efficient resource provisioning in cloud Services. *ACM Sigmetrics Perform. Eval. Rev.* **2015**, *42*, 3–11. [CrossRef]
28. Yue, T.; Xia, C.H. An Adaptive Learning Approach for Efficient Resource Provisioning in Cloud Services. *ACM Sigmetrics Perform. Eval. Rev.* **2015**, *42*, 3–11.