# Optimized Self-adaptive Fault Tolerance Strategy in Simulation System based on Virtualization Technology

Zhijia Chen, Yuanchang Zhu, Yanqiang Di and Shaochong Feng

***Abstract*—In simulation system, failures of simulation nodes and deficiencies in resources availability affect the effectiveness of simulation system. To improve the reliability of distributed simulation system, an optimized self-adaptive fault tolerance approach is proposed to improve the effect of fault tolerance and reduce the overhead. We construct the self-adaptive fault tolerance architecture based on virtualization technology. We divide the failures according to their locations and realize fault tolerance by self-adaptive selection strategy. Then three main problems of replication fault tolerance strategy are analyzed, including node selection, number of copies and location distribution. Solutions are proposed combining virtualization technology. Besides, virtual machine migration is adopted and optimized with weighted descending multi-attribute matching method to improve the performance and reduce the overhead of simulation system. Finally, we evaluate the performance of the proposed fault tolerance approach using experiment data. The effectiveness of fault tolerance is guaranteed and the overhead is controlled at a low level.**

***Index Terms*—fault tolerance, replication strategy, virtual machine migration, virtualization technology, weighted descending multi-attribute matching method**

## I. Introduction

### A. Background and Related Works

IN distributed simulation system, along with the expansion of simulation scale and system running, the system reliability and service availability decrease while the failure probability increases. Once the key simulation nodes fail, the whole simulation system would be affected and even crash. If the system does not have the function of fault tolerance, then the only way is to restart the whole system, which will result in disastrous consequence and make cause simulation process failure. Thus, to improve the fault tolerance ability in simulation system, the problems caused by node failures and resource deficiency are the key problems supposed to be prevented and treated.

Virtualization [1] is an emerging paradigm that separates operating system and corresponding technology implementations from physical hardware. The development of virtualization technology makes the usage of virtual machine (VM) much convenient. The whole state of the virtual machine can be saved in a small file located in a certain server. The running virtual machine can be easily migrated from one host to another without shutting it down. Once the data transmission of simulation nodes in one host is blocked because of frequent and mass data interaction, specific VMs may be selected to migrate to another host to balance the utilization of network resources. This is important for the simulation system to prevent failures caused by node collapse. Therefore, virtualization technology provided a new reference for the fault tolerance of simulation system.

In the area of fault tolerance, there are different strategies for different systems. To improve the fault tolerance performance, researchers have been carrying out lots of investigations. In [2], a low-overhead and high-performance fault tolerance architecture for application-specific network-on-chip. Link-interface is developed to reduce the hardware overhead. This architecture also improves the average response time of system by 27% comparing to traditional mesh. Pranesh Das [3] adopted virtualization technology and load balance technology to redeploy the failed nodes, normal nodes and the submitted task. The success rate of task processing is improved by this method. LIU Yun-sheng et al. [4] constructed distributed simulation fault tolerance system based on HLA (High Level Architecture). The system was composed of three modules: the simulation resources monitoring module, data storage module and data recovery module. They made research on using grid technology to simplify fault tolerance preliminarily, but still there is space in simplifying and optimization of the system availability. To solve the unreliability of spot instance in cloud environment [5], Daeyong Jung et al. [6] proposed fault tolerance strategy based on virtual machine migration. The algorithm set checkpoint based on SLA (Service Level Agreement). When the service failed, the service virtual machine would be migrated to a new server. The algorithm takes effect in a certain degree, but it is specially designed and cannot be used universally. To satisfy high reliability and efficiency of

Manuscript received March 17, 2015; revised May 06, 2015. This work was supported in part by the Equipment Pre-research Fund of China (Grant No. 9140A04030214JB34058).

Zhijia Chen (IAENG Member) is with the Department of Electronic and optics, Mechanical Engineering College, Shijiazhuang, 050003, China (corresponding author: +86-031187994267; email: youshenshui@163.com).

Yuanchang Zhu is with the Mechanical Engineering College, Shijiazhuang, 050003, China (yuanchang_zhu_oec@163.com).

Yanqiang Di is with the Department of Electronic and optics, Mechanical Engineering College, Shijiazhuang, 050003, China (yanqiang_di@163.com).

Shaochong Feng is with the Department of Electronic and optics, Mechanical Engineering College, Shijiazhuang, 050003, China (fscsat@126.com).

parallel computing system, Xu et al. [7] proposed a multi-layer and multi-angle parallel fault tolerance computer system. In the paper, method of processing the in-transit message and isolated message was researched and validated in DSP development board. However, this method has not been validated in scale-distributed system. To deal with faults and run-time anomalies in the infrastructure in cloud environments, Ganesan Radhakrishnan [8] outlined an approach to decide the mapping relation between task and server nodes based on data throughput and task waiting time. This method improved application fault-tolerance and resource utilization. To achieve high level of cloud serviceability and to meet high level of cloud SLO (Service Level Objectives), Dawei Sun et al. [9] proposed a dynamic adaptive fault tolerance strategy DAFT. It analyzes the mathematical relationship between failure rate and fault tolerance strategy: checkpoint and replication fault tolerance strategy. However, the analysis and derivation are still not consummate and need to be improved. Bo Yang et al. [10] considered recovery on both processing nodes and communication links to improve cloud service reliability. The subtask waiting time, processing time and completion time were modeled in theoretical. In [11], a decision framework for prioritizing business applications for SaaS migration was proposed. In the algorithm, the total system life cycle cost was analyzed for cost estimation. It provides reference for the fault tolerance in virtualization environment.

### *B. Contributions*

To realize high performance and low overhead of fault tolerance in simulation system, we propose an optimized self-adaptive fault tolerance strategy based on virtualization technology. Using the experience of fault tolerance strategies in above researches for reference, we take advantage of virtualization technology to optimize replication fault tolerance strategy and virtual machine migration based fault tolerance strategy. According to the failure location, the fault tolerance strategy is adopted by self-adaptive selecting. Our contributions are summarized in the following:

(1) Introduce virtualization technology to the fault tolerance system to improve the effectiveness and reduce the overhead of fault tolerance.

(2) Construct the fault tolerance architecture of simulation system using virtualization technology. According to the failure locations, we divide the failures as two layers: virtual resource layer and physical resource layer. In different layers, different fault tolerance strategies are adopted by self-adaptive.

(3) Systematically analyze the problems of replication fault tolerance strategy. By reasonable formulation of the mathematical modeling, the simulation node selection, number of copies and location distribution are resolved and optimized.

(4) Propose the weighted descending multi-attribute matching method to realize the virtual machine migration. With the method, the overhead and the effectiveness of the virtual machine migration strategy are improved.

Overall, the logical analysis and experiment results show that the optimized self-adaptive fault tolerance strategy is effective for simulation system fault tolerance. The strategy is also qualified for reducing the overhead.

## II. Fault Tolerance System Overview

To analyze the failures in simulation system, we divide the failures into two types [12]: application layer failure and resource layer failure. Application layer failures mean the failures occur in the upper layer such as simulation middle ware, simulation applications and simulation processes; resource layer failures mean the failures occur in simulation resources that simulation application rely on, especially physical resources or virtual physical resources. Resource layer failures include machine crashes and network block failures, etc. This paper puts emphasis on resource layer failure tolerance based on virtualization technology. When failure occurs in resource layer, replication strategy or virtual machine migration strategy can be adopted. Especially when failures occur in virtual machine, the virtual machine migration will not work, as it cannot clear the virtual machine level failures. Thus, the replication fault tolerance needs to be adopted to guarantee the system reliability and service availability.

### *A. System Architecture*

As depicted in Fig. 1, the fault tolerance system is constructed as four-layer architecture. From bottom layer to top layer, there are physical resource layer, virtualization layer, fault tolerance layer and user layer. In physical resource layer, multiple data centers are created in different geographic locations containing physic resources including CPU, GPU, RAM and storage, etc. One data center is also called one resource pool [13]. The resource pools are connected by network and provide hardware infrastructure for creating virtual machines. Virtualization layer contains virtualization software and virtual resources such as virtual CPU (vCPU), virtual GPU (vGPU) and virtual memory etc. The virtual resources compose virtual machines by customizing. The virtualization software is Hypervisor, which is also known as virtual machine monitor (VMM) [14]. Through virtualization software, the physical resources are virtualized as virtual resources. Then the virtual resources are configured as different kinds of virtual machines. Virtual machines can provide simulation services for users like physical machines. The management of virtual machines is transparent to upper level users and thus users need not concern about the management of VMs because it can be achieved by data center managers. The scalability and flexibility of virtual machines architecture provide excellent support for users' simulation fault tolerance services. In fault tolerance layer, by optimizing replication strategy and virtual machine migration strategy, the overhead is decreased. Here the fault tolerance strategy mainly focuses on fault recovery and decrease of the fault tolerance overhead. The fault monitoring and predicting is not our scope. Corresponding failure monitoring strategy is introduced [15]. Through self-adaptive choosing fault tolerance strategy, the reliability of simulation system is improved. In user layer, users can connect to simulation center and process corresponding simulation task. The simulation service reliability is guaranteed by fault tolerance layer.

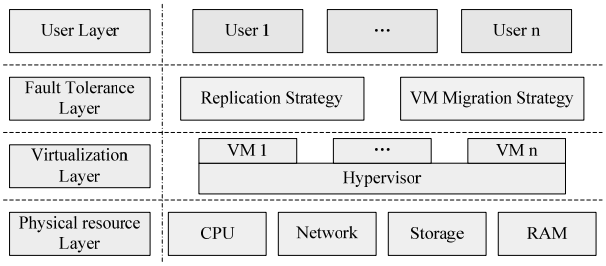


Fig. 1 The architecture of fault tolerance system

The operating principle of the proposed dynamic fault tolerance system is depicted in Fig. 2. As simulation resources include physical resources and virtual resources, thus, when failures happen, we first distinguish the specific position that failures locate. If the failure happens in physical resource layer, then we can adopt the replication strategy to start the simulation in another server to avoid the CPU, network failures etc. Meanwhile, we can adopt the optimized VM migration strategy to reduce the overhead, as the replication strategy needs more physical resources than VM migration does. If the failure happens in virtual resource layer, then the VM migration strategy will not be effective because migration cannot solve the virtual machine level failure. Therefore, we can adopt replication strategy, copy the simulation node to other servers and start the simulation node when original simulation node fails.

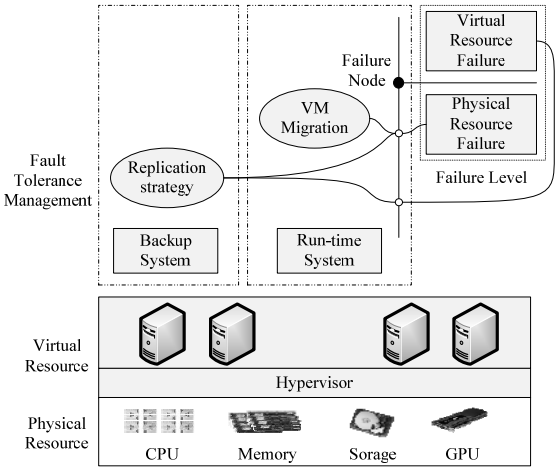


Fig. 2 The sketch map of relationship and principle of dynamic fault tolerance system

### *B. Assumptions and Problem Descriptions*

To make the research more convincing and precise, we make description to the problems as follows:

(1) Failure model: with the time advancing, the system failure probability follows an exponential distribution [16]:

$$F_n(t) = \begin{cases} 0 & t < 0 \\ 1 - e^{-\lambda t} & t >= 0 \end{cases} \quad (1)$$

The failure probability density function is:

$$f_n(t) = \begin{cases} 0 & t < 0 \\ \lambda e^{-\lambda t} & t >= 0 \end{cases} \quad (2)$$

In time interval (a, b), the failure expectation value means the weighted average value of the potential failures in the interval, which can be expressed as:

$$E(t) = \int_a^b \tau \cdot f(\tau) d\tau \quad (3)$$

(2) Failures occur randomly. Failures will be detected once they happen with system running.

(3) All failures follow the same distribution and the recovery time is the same when the same fault tolerance strategy is adopted.

(4) If one node fails or the network of the node fails, the simulation error will not influence the interaction communication of other nodes, and the node can re-join the simulation net after recovery.

(5) We assume the virtual environment is Xen, in which the VM migration time, replication time and recovery time are known.

## III. Replication Fault Tolerance Strategy

In traditional checkpoint strategy [17], it mainly focuses on simulation application failures. Although it can roll back to the checkpoint when simulation application fails, when the physical resource problems arise, checkpoint will not take effect any more. In IaaS (Infrastructure as a Service) cloud simulation system, using virtual machines created by virtualization technology, we can provide replication for simulation system to avoid system crash caused by single node error. This can improve the fault tolerance effect.

The relation of simulation nodes and backup nodes in replication fault tolerance is shown in Fig. 3. In the architecture, the simulation nodes and backup nodes are physical isolated, which can keep the backup nodes from suffering disaster caused by simulation nodes fault. Therefore, the backup of simulation system data will not be influenced and replication fault tolerance strategy can be implemented effectively. When the system is running, we set a heartbeat time, in which the simulation node sends the updating information to the backup node and the backup node will send feedback information to the simulation node, as shown in step ①. If the backup node does not receive the updating information from the simulation node in the stipulated time, then we regard the simulation node as failure. The backup node is set as new simulation node to replace the original failure simulation node. The new simulation node communicates with other nodes to advance the simulation proceeding, as shown in step ②. Meanwhile, system tries to create new backup node in the place of the original simulation node. If the failure cannot recover, the original simulation node will be marked as “dead machine”. The “dead machine” will not interact with other nodes. This may result the number of copy decreasing under a certain threshold. To guarantee the fault tolerance performance, system will create backup node close to the “dead machine” and start a new round replication procedure. The new copy node starts heartbeat interaction with simulation node, as depicted in step ③.

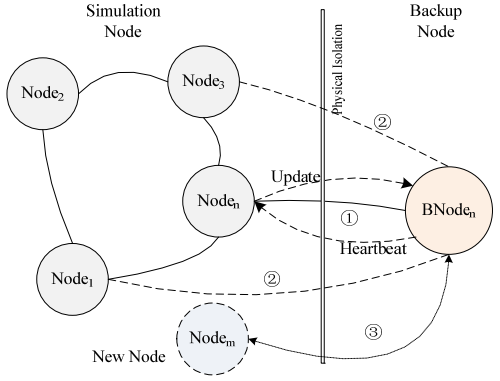


Fig. 3 The relation of simulation nodes and backup nodes in replication strategy

In replication strategy, the optimization mainly considers the following questions. (1) If the original node selection and the copy time selection are improper, the fault tolerance capability cannot be strengthened. Thus, the data type and replication moment selection is important for reducing the fault recovery time and improve the fault tolerance effect. (2) With the growth of the number of the copies, the maintenance of the system will increase. Too many copies will also cause unnecessary overhead. (3) The improper location of copies will cause bandwidth over-occupation, time delay and performance decrease. Thus, the aim of location selection is to process the tasks successfully and minimize the time delay as well.

### *A. Node and Moment Selection of Replication*

In order to improve the efficiency and reduce the overhead, referring to Pareto principle [18] and temporal locality principle [19], we select the dynamic simulation node according to user data interaction frequency and data quantity. When the dynamic degree $d_k$ of node $N_k$ is up to a certain threshold, the replication operation is triggered. To calculate $d_k$, we first introduce forgetting parameter $\xi$. $\xi$ has an exponential function relation with the time interval from start time $t_s$ to current time $t_p$ and data interaction frequency. Smaller the forgetting parameter $\xi$ is, the dynamic degree is higher and vice versa. The forgetting parameter $\xi$ can be calculated using the follow formula:

$$\xi(t_p,t_s)=e^{-\lambda\cdot\frac{k}{t_p-t_s}},\ k\in\{1,2,...\} \tag{4}$$

Set $\Delta t=t_p-t_s$, then formula (4) can be simplified as follows:

$$\xi(t_p,t_s)=e^{-\lambda\cdot\frac{k}{\Delta t}} \tag{5}$$

From formula (5) we can see the range of $\xi$ is: $\xi\in[0,1]$. $k$ represents the number of data interaction times of node $N_k$ in time interval $\Delta t$. If $\frac{k}{\Delta t}\to 0$, then $\xi=\lim_{\frac{k}{\Delta t}\to 0}e^{-\lambda\cdot\frac{k}{\Delta t}}=0$; If $\frac{k}{\Delta t}\to\infty$, then $\xi=\lim_{\frac{k}{\Delta t}\to\infty}e^{-\lambda\cdot\frac{k}{\Delta t}}=0$.

The dynamic degree $d_k$ of node $N_k$ in the interval form start time $t_s$ to current time $t_p$ can be calculated as:

$$d_k=\sum_{t_i=t_s}^{t_p}\left[I_k(t_i,t_{i+1})\cdot\left(1-\xi(t_i,t_p)\right)\right] \tag{6}$$

Where $I_k(t_i,t_{i+1})$ means the data interaction number of times in time interval $[t_i,t_{i+1})$. From formula (6), we can see that the more interaction times one node has, smaller the forgetting parameter is, and higher the dynamic degree is.

We define $\varphi_k$ as the multiplication of node dynamic degree and weight $w_k$.

$$\varphi_k=d_k\cdot w_k \tag{7}$$

If $\varphi_k$ is greater than the set threshold, we regard node $N_k$ as a key node. Then the replication operation of node $N_k$ will be triggered. If $\varphi_k$ is smaller than the set threshold, then the dynamic degree is not high enough and node $N_k$ is not a key node. The replication operation of node $N_k$ will not be triggered.

### *B. Number of Copies*

To satisfy the fault tolerance demands, new copies will be created and old copies will be deleted. When fault tolerance has a rational increase, the newly increased copy number $n_k(inc)$ is:

$$n_k(inc)=n_k(new)-n_k(old) \tag{8}$$

Where $n_k(new)$ is the new number of copies, $n_k(old)$ is the old number of copies. $n_k(inc)$ can be positive or negative, means the number of copies is increased or decreased. For node $N_k$, the availability of new copy $\mu_{new}$ has relation with availability of current copy $\mu$ as follows:

$$1-\mu_{new}=(1-\mu)^{n_k(new)} \tag{9}$$

Meanwhile, the availability $\mu_{new}$ of new copy in formula (9) can be calculated using the following formula:

$$\mu_{new}=\mu_{old}+\alpha\left(1-\mu_{old}\right) \tag{10}$$

Where $\mu_{old}$ means the old availability of node $N_k$, $\alpha$ is the adjusting parameter of copy number. From formula (10), it can be known that the availability of new copy is determined by old availability and the adjusting parameter $\alpha$. Here $\alpha$ can be calculated as follows:

$$\alpha=\frac{\varphi_{k(new)}-\varphi_{k(old)}}{\varphi_{k(new)}} \tag{11}$$

Where $\varphi_{k(old)}$ and $\varphi_{k(new)}$ are the old and new replication parameter. We get formula (12) using simultaneous equations (9), (10) and (11):

$$1-(1-\mu)^{n_k(new)}=\mu_{old}+\alpha\cdot(1-\mu_{old}) \tag{12}$$

Changing the formation, we get:

$$n_k(new)=\frac{\ln\left(1-\mu_{old}-\alpha\cdot(1-\mu_{old})\right)}{\ln\left(1-\mu\right)} \tag{13}$$

As:

$$1-\mu_{old}=(1-\mu)^{n_k(old)} \tag{14}$$

We get formula (15) using simultaneous equations (13) and (14):

$$n_k(new) = \frac{\ln\left((1-\mu)^{n_k(old)} \cdot (1-\alpha)\right)}{\ln(1-\mu)} = \frac{n_k(old)\cdot\ln(1-\mu)+\ln(1-\alpha)}{\ln(1-\mu)} \quad (15)$$

So,

$$n_k(inc) = n_k(new) - n_k(old) = \frac{\ln(1-\alpha)}{\ln(1-\mu)} \quad (16)$$

By the above analysis, we can get the number of copies through formula (16).

In Fig. 4 we show the changing curve of number of replications. The positive number represents that the number of replications needs to be increased, while the negative number represents that the number should be decreased. From the figure, we can see that if $\alpha > 0$, the current replication parameter is bigger than the old replication parameter, then the number of replications should be added; vice versa. Meanwhile, if the availability of the node is higher, then the influence is smaller to the current replication number, and the number only needs to be readjusted in a small range. If the node availability is low, the replication number has to be readjusted in a large range to satisfy the fault tolerance requirement.

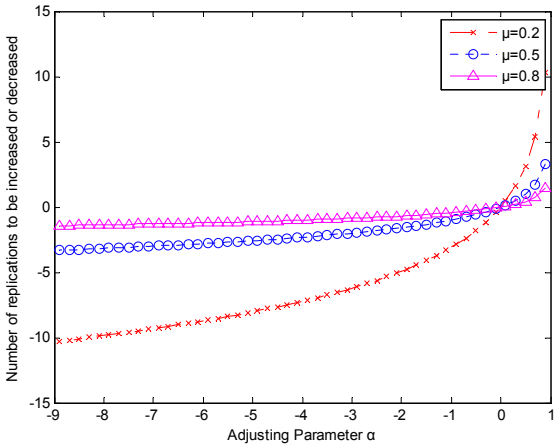


Fig. 4 The number of replications to be increased or decreased

### C. Location Distribution

To reduce data transmission quantity and time delay and satisfy fault tolerance requirements, the location of the copies has to be selected properly. From the angle of decreasing system overhead, first the original node should be selected as replication node. However, the original simulation node may fail to restart and recover from failure status. To reduce fault tolerance overhead, we calculate the distance between original simulation node and other nodes. We sort the nodes following an ascending sequence. First, we consider the nearest node from original simulation node and compare the resource quantity of original node and the target node. If the target node can provide enough simulation resources, then we create the copy on the node. Otherwise, according to distance sequence, we compare the resources of original simulation node and target node in ascending order until the condition is satisfied. The procedure can be concluded as follows:

(1) Calculate the distance between original simulation node and the target nodes;

(2) Compare the distances and sort in an ascending sequence;

(3) Calculate the available resource quantity of each target node;

(4) Compare the resource quantity between original simulation node and the nearest target node to decide whether the target node can effectively process the simulation task;

(5) If the resource quantity of target node is more than the original simulation node, then the target node is selected as replication node;

(6) Otherwise, search target node according to the resource quantity and the ascending distance sequence until condition 4) is satisfied. Then the target node is selected as replication node.

## IV. Optimization of Fault Tolerance based on Virtual Machine Migration Strategy

Virtual machine migration strategy mainly deals with the following problems. (1) The server has to shut down because of power or maintenance, which will cause the virtual machine unable to process simulation task during the period. (2) Simulation virtual machine resource shortage caused by server resource overutilization can be resolved by virtual machine migration. Migration to a new server is able to provide enough resource for the simulation virtual machine and improve the simulation environment [20]. In this paper, we adopt weighted descending multi-attribute matching method to realize the virtual machine migration.

By real-time monitoring of the remaining resource of the current server, we construct the remaining resource vector $\boldsymbol{s} = \{s_{CPU}, s_{Mem}, s_{Net}, s_{GPU}\}$. Here $s_{CPU}$ means the remaining CPU resource, $s_{Mem}$ means the remaining memory resource, $s_{Net}$ means the remaining network bandwidth resource, $s_{GPU}$ means the remaining GPU resource. Once we find the server is short of certain kind of resource, we will start the virtual machine migration procedure.

According to the maximum utilization of CPU, memory, network and GPU resource of each simulation virtual machine in history, we construct the resource utilization matrix $U$.

$$U = \{u_{vm_1}, u_{vm_2}, \cdots, u_{vm_n}\}^T = \begin{pmatrix} u_{CPU_1} & u_{Mem_1} & u_{Net_1} & u_{GPU_1} \\ u_{CPU_2} & u_{Mem_2} & u_{Net_2} & u_{GPU_2} \\ \cdots & \cdots & \cdots & \cdots \\ u_{CPU_n} & u_{Mem_n} & u_{Net_n} & u_{GPU_n} \end{pmatrix}$$

In accordance with the simulation task requirements for resources, different resource types are assigned with different weights, including CPU weight $w_{CPU}$, memory weight $w_{Mem}$, network bandwidth weight $w_{net}$ and GPU weight $w_{GPU}$. For example, distributed data transmission task is mainly reliable on network performance, so we weight network bandwidth over other resources. While for 3D model simulation task, it is more reliable on GPU performance as well as memory performance, so we weight GPU and memory over other resources. Using the weighted simulation resource requirement, it is more applicable for different simulation nodes and different simulation tasks. The requirements of

simulation tasks are prone to be satisfied as the corresponding weights assigned to the simulation tasks. Different weights compose the weight matrix $W$.

$$W = \left\{ w_{vm_1}, w_{vm_2}, \cdots, w_{vm_n} \right\}^T = \begin{pmatrix} w_{CPU_1} & w_{Mem_1} & w_{Net_1} & w_{GPU_1} \\ w_{CPU_2} & w_{Mem_2} & w_{Net_2} & w_{GPU_2} \\ \cdots & \cdots & \cdots & \cdots \\ w_{CPU_n} & w_{Mem_n} & w_{Net_n} & w_{GPU_n} \end{pmatrix}$$

$\forall i$, s.t. $\sum_j w_{ij} = 1$.

In each row of matrix $W$, different weight represents the degree that the simulation node demands for the resource. We get the weighted requirements matrix by multiplying weight and the corresponding resource requirements as follows:

$$\hat{U} = \begin{pmatrix} w_{CPU} \cdot u_{CPU_1} & w_{Mem} \cdot u_{Mem_1} & w_{St} \cdot u_{Net_1} & w_{GPU} \cdot u_{GPU_1} \\ w_{CPU} \cdot u_{CPU_2} & w_{Mem} \cdot u_{Mem_2} & w_{St} \cdot u_{Net_2} & w_{GPU} \cdot u_{GPU_2} \\ \cdots & \cdots & \cdots & \cdots \\ w_{CPU} \cdot u_{CPU_n} & w_{Mem} \cdot u_{Mem_n} & w_{St} \cdot u_{Net_n} & w_{GPU} \cdot u_{GPU_n} \end{pmatrix} = \begin{pmatrix} \hat{u}_{CPU_1} & \hat{u}_{Mem_1} & \hat{u}_{Net_1} & \hat{u}_{GPU_1} \\ \hat{u}_{CPU_2} & \hat{u}_{Mem_2} & \hat{u}_{Net_2} & \hat{u}_{GPU_2} \\ \cdots & \cdots & \cdots & \cdots \\ \hat{u}_{CPU_n} & \hat{u}_{Mem_n} & \hat{u}_{Net_n} & \hat{u}_{GPU_n} \end{pmatrix}$$

According to the weights, we define the resource types as primary, secondary and so on. For example, in a 3D model simulation node, the GPU resource will be defined as primary resource and the memory resource will be defined as secondary resource. Compare the elements of $S$ and $U_i$. $\forall i, j$, if $\exists\ s_j \le u_{ij}$, then there are virtual machines have to be migrated from the current server. In each column of the weighted requirements matrix $\hat{U}$, compare the elements of the same column. Then arrange the elements in a descending order. We select the virtual machine that has the largest weighted requirement resource as the VM needs to be migrated.

Now we consider the selection of migration target server. We get the remaining resource quantity of the target servers by real time monitoring and construct target server remaining resource matrix $R$.

$$R = \begin{pmatrix} r_{CPU_1} & r_{Mem_1} & r_{Net_1} & r_{GPU_1} \\ r_{CPU_2} & r_{Mem_2} & r_{Net_2} & r_{GPU_2} \\ \cdots & \cdots & \cdots & \cdots \\ r_{CPU_n} & r_{Mem_n} & r_{Net_n} & r_{GPU_n} \end{pmatrix}$$

Here $r_{CPU}$ means the remaining CPU resource, $r_{Mem}$ means the remaining memory resource, $r_{Net}$ means the remaining network bandwidth resource, $r_{GPU}$ means the remaining GPU resource. To realize high availability and reliability of target server, we need to prevent the overload of target server. First, we compare the remaining resource $r_{kj}$ of target server and the maximum requirements $u_{ij}$ of the virtual machine to be migrated. For all the elements of matrix $R_k$ and matrix $\hat{U}_i$, $\forall i, j$, if $r_{kj} > \hat{u}_{ij}$, then we regard the server can be selected as target server. If there are two or more servers that satisfy the inequality constraints, we need to match the simulation resources according to their weights. We sort the corresponding row of matrix $R$ in a descending sequence according to the weighted resource type of the selected VM. Then we choose the first row of the sorted matrix $R$. The first server will be the target server. If there are two or more servers having the same primary weighted remaining resource, then the secondary weighted remaining resource will be measured, until the target server is selected. If the original server is still short of resource after the virtual machine is migrated, then the virtual machine will be migrated according to the descending order until the server resource provision reaches a reasonable level. The pseudo code of optimized VM migration procedure is presented in Algorithm 1.

**Algorithm 1** Optimized virtual machine migration

**input:** remaining resource vector of original server *S*
resource utilization matrix of VMs *U*
resource remaining matrix of target servers *R*
**output:** {VM$_k$, server *l*}: Selected VM to be migrated, the target server

```
for i=1 to N    // N is the number of the VMs hosted in the original server
  for j=1 to M  // M is the number of resource types of VM
    weight the jth resource of VMi as w_ij according to the simulation tasks
    requirements;
    get weighted resource utilization û_ij by multiplying resource
    utilization u_ij by w_ij ;
  end for
  sort û_i in a descending sequence;
end for
select the VM that has the maximum û_k , marked as VM_k; // select the VM to
be migrated
for i=1 to Q  // Q is the number of the target servers
  sort the servers according to the primary resource type;
  compare û_l with the remaining resources r_ij of server i ;
  mark the row number i ;
end for
if r_ij > û_k , then
  if there is only one server l satisfy the constraint, then
    server l is the target server;
  else
    do
      compare the secondary resource type;
      if the optimal target server l is selected, then
        break;
      end if
    while (1)
  end if
end if
return {VM_k, server l}.  // return the VM that needs to be migrated and the
target server number
```

Above is the optimization of virtual machine migration method. Now we consider the overhead of the virtual machine migration. As the storage of physical resource pool is storage area network (SAN) [21], thus the storage is not migrated. Here we mainly consider the memory migration to be brief and to the point. Suppose the memory is $M$ (Byte), the network bandwidth among servers is $B$ (bps). In research [22,23], dynamic migration technology is introduced which can realize virtual machine migration without shutting down. Here we do not give details due to limited space and we mainly consider the migration time under such a technology.

In memory iterative copy operating procedure, set the memory-modifying rate is $\rho_i$ in the $i$th round, which is called dirty page rate. Then the transmission time of the $i$th round is:

$$t_i = \begin{cases} \dfrac{8 \cdot M}{B}, & i = 1 \\ \dfrac{8 \cdot \rho_{i-1} \cdot t_{i-1}}{B} = \dfrac{8 \cdot M \cdot \prod_{j=1}^{i-1} \rho_j}{\prod_{j=1}^{i-1} B}, & i \geq 2 \end{cases} \quad (17)$$

Set the total number of migration times is $N$. It means after $N$ times of migration, the amount of dirty page is less than a certain threshold or the number of migration times is greater than a certain threshold. The total migration time is:

$$T_{MM} = 8 \cdot \left( \frac{M}{B} + \sum_{i=2}^{N-1} \frac{M \cdot \prod_{j=1}^{i-1} \rho_j}{\prod_{j=1}^{i-1} B} \right) \quad (18)$$

From above analysis, we can see that the actual transmission data is larger than the virtual machine memory. The totality of the transmission data is:

$$D_{MM} = T_{MM} \cdot B = 8 \cdot \left( M + \sum_{i=2}^{N-1} \frac{M \cdot \prod_{j=1}^{i-1} \rho_j}{\prod_{j=1}^{i-2} B} \right) \quad (19)$$

## V. Experimental Evaluation

We evaluated the optimized fault tolerance algorithm on the simulated cloud platform CloudSim [24]. There are 20 virtual machines were created and the configuration parameters are shown in TABLE I. The simulation task number varies from 50 to 3200. The parameters of task are shown in TABLE II.

TABLE I Virtual Machine Configuration Parameters

| PEs | MIPS | RAM | BW | STORAGE |
|---|---|---|---|---|
| 1 | 1000 | 512 | 1000 | 10000 |

TABLE II The Task Parameters

| Length | File size | Number |
|---|---|---|
| 1000 | 300 | 50-3200 |

### A. Performance of Fault Tolerance Degree

Fault tolerance degree $d_{ft}$ is defined as the ratio of successful processing number of the tasks to the total number of the submitted tasks.

$$d_{ft} = \frac{N_s}{N_t} \quad (20)$$

$N_s$ is the number of the successfully executed tasks, $N_t$ is the number of the submitted tasks. In Fig. 5, the comparison of fault tolerance degree is shown when different strategies are adopted. The X-axis represents the number of simulation tasks, which vary from 50 to 3200. The Y-axis represents the fault tolerance degree as mentioned in the above equation. The fault tolerance degree is greater than 80% when the optimized self-adaptive fault tolerance strategy is adopted. The replication strategy mainly depends on resource redundancy and realizes system reliability by sacrificing the physical resources. The system reliability of this strategy is also high and the fault tolerance degree is greater than 70%. When virtual machine migration strategy is adopted, the fault tolerance degree is lower than when other strategies are adopted. When virtual machine migrates, some time is needed and during this time interval, the submitted tasks will not be processed. Therefore, the fault tolerance degree is lower when the VM migration strategy is adopted.

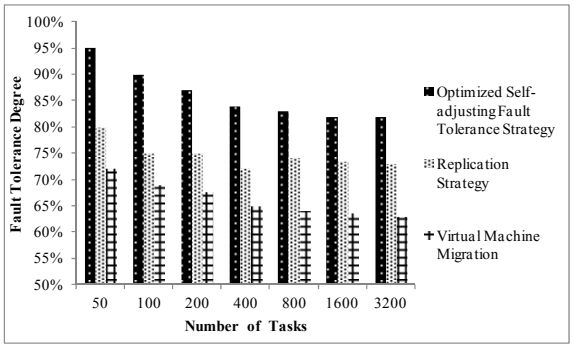


Fig. 5 The comparison of fault tolerance degree

### B. Evaluation of Physical Resource Overhead

Fig. 6 shows the physical resource overhead comparison when different fault tolerance strategies are adopted. The X-axis represents the number of simulation tasks, which vary from 50 to 400. The Y-axis represents the fault tolerance resource overhead. We can see that the resource overhead of replication strategy is the heaviest. This is because the replication strategy creates new copies in different locations. Meanwhile, the transmissions of simulation data among these copies also need to consume network bandwidth. The VM migration strategy only consumes network bandwidth while dirty page refreshing, thus the resource overhead is lighter comparing with replication strategy. When the optimized self-adaptive fault tolerance strategy is adopted, the resource overhead is lighter than VM migration strategy. This is because our proposed method optimizes the two typical fault tolerance strategies and the self-adaptive choosing of the optimized strategies. The proposed self-adaptive fault tolerance strategy realizes minimum overhead on the premise of effective fault tolerance.

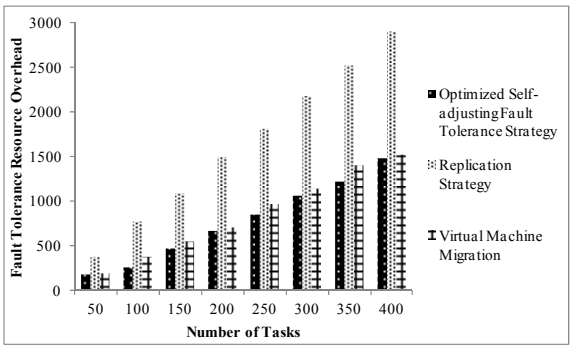


Fig. 6 Comparison of fault tolerance resource overhead

### C. Evaluation of Time Overhead

The time consuming condition is shown in Fig. 7. The X-axis represents the number of simulation tasks, which vary from 50 to 400. The Y-axis represents the fault tolerance time overhead. In the procedure of virtual machine migration, the

memory state has to be copied continuously. So the migration and recovery of virtual machine both need some time. Thus, the time overhead is heavy. While we adopt replication strategy, multi copies are running as the original simulation node does. When original node collapses, the copies can replace the original node and execute the simulation tasks as soon as possible. Therefore, the time overhead is lighter than the virtual machine migration. This properly explains the utilization of physical redundancy to get time efficiency in turn. The proposed optimized self-adaptive fault tolerance strategy reduces the time overhead of fault tolerance. It chooses the most effective method to realize fault tolerance, thus the overhead is smaller than replication strategy and VM migration strategy.

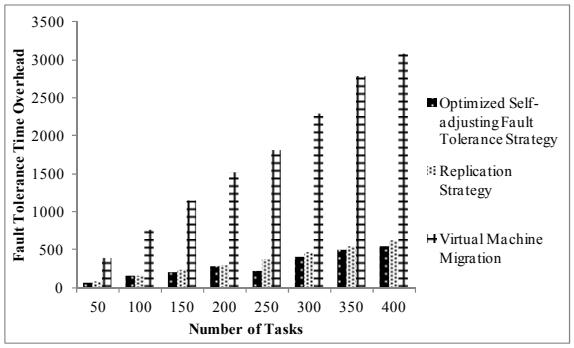


Fig. 7 Comparison of the fault tolerance time overhead

## VI. Concluding Remarks and Discussions

Based on typical fault tolerance, we propose an optimized fault tolerance strategy using virtualization technology. We research the optimization of the number copies and the location of replication strategy and propose an optimization method based on virtualization technology. Meanwhile, we introduce virtual machine migration based fault tolerance strategy to guarantee the system stability. We propose the weighted descending multi-attribute matching method to realize simulation resource sensitive matching of simulation virtual machine to be migrated and target server. By above mentioned methods, effective and light overhead fault tolerance is realized. Our research does an overall consideration of the practical conditions. However, the research still mainly provides a theoretical analysis. The future goal is to test the method in a cloud system for practical use.

## References

[1] Qi Z, Yao J, Zhang C, et al., "VGRIS: virtualized GPU resource isolation and scheduling in cloud gaming," *ACM Transactions on Architecture & Code Optimization*, vol. 11, no. 2, pp61-85, 2014.

[2] Fathollah Karimi Koupaei, Ahmad Khademzadeh, Majid Janidarmian, "Low-Overhead and High-Performance Fault-Tolerant Architecture for Application-Specific Network-on-Chip," *IAENG International Journal of Computer Science*, vol. 39, no. 1, pp96-101, 2012.

[3] Pranesh Das, "*Virtualization and Fault Tolerance in Cloud Computing*," Rourkela: National Institute of Technology Rourkela. 2013.

[4] LIU Yun-sheng, ZHA Ya-bing, ZHANG Chuan-fu, et al., "Research of Fault-Tolerance Mechanism in Distributed Simulation System," *Journal of System Simulation*, vol. 17, no. 2, pp355-357, 2005.

[5] Ling Qian, Zhiguo Luo, Yujian Du, et al., "Cloud Computing: An Overview," *Lecture Notes in Computer Science*, vol. 5931, no. 1, pp626-631, 2009.

[6] Daeyong Jung, SungHo Chin, Kwang Sik Chung, et al., "VM Migration for Fault Tolerance in Spot Instance Based Cloud Computing," *Grid and Pervasive Computing*, vol. 7861, no. 1, pp142-151, 2013.

[7] Xu Xiaodong, Zhao Jianting, Xu Chunlei, "Fault Tolerance in Real-Time and Multitask Parallel Computing System," *Computer Engineering and Applications*, vol. 49, no. 9, pp33-37, 2013.

[8] Ganesan Radhakrishnan, "Adaptive Application Scaling for Improving Fault-Tolerance and Availability in the Cloud," *Bell Labs Technical Journal*, vol. 17, no. 2, pp5-14, 2012.

[9] Dawei Sun, Guiran Chang, Changsheng Miao, "Analyzing, Modeling and Evaluating Dynamic Adaptive Fault Tolerance Strategies in Cloud Computing Environments," *Journal of Supercomputing*, vol. 66, no. 1, pp193-228, 2013.

[10] Bo Yang, Feng Tan, Yuan-Shun Dai, "Performance Evaluation of Cloud Service Considering Fault Recovery," *Journal of Supercomputing*, vol. 65, no. 1, pp426-444, 2013.

[11] Eugene Rex L. Jalao, Dan L. Shunk, Teresa Wu, "Life Cycle Costs and the Analytic Network Process for Software-as-a-Service Migration," *IAENG International Journal of Computer Science*, vol. 39, no. 3, pp269-275, 2012.

[12] Liu Yunsheng, "*Research on Key Technologies of Fault Tolerance of Large Scale Distributed Simulation System*," Changsha: National University of Defense Technology, 2006.

[13] Lu Huang, Haishan Chen, Tingting Hu, "Survey on Resource Allocation Policy and Job Scheduling Algorithms of Cloud Computing," *Journal of Software*, vol. 8, no. 2, pp480-487, 2013.

[14] Zhen Xiao, Weijia Song, Qi Chen, "Dynamic Resource Allocation using Virtual Machines for Cloud Computing Environment," *IEEE Transaction on Parallel and Distributed Systems*, vol. 24, no. 6, pp1107-1117, 2013.

[15] Kyoungho An, "Resource Management and Fault Tolerance Principles for Supporting Distributed Real-time and Embedded Systems in the Cloud," *Middleware 2012 Doctoral Symposium*, pp4, 2012.

[16] Wang Jing, Wang Gaocai, Huang Yihai, "Study on Fault Tolerance of Mesh Networks based on Node Stochastic Failure Probability," *Journal of Chinese Computer Systems*, vol. 31, no. 5, pp888-891, 2010.

[17] Zhang Haojia, "*A VM-level Fault-Tolerant System for Virtual Clusters with Coordinated Checkpointing*," Wuhan: Huazhong University of Science and Technology, 2010.

[18] Susumu Cato, "Pareto Principles, Positive Responsiveness, and Majority Decisions," *Theory and Decision*, vol. 71, no. 4, pp503-518, 2011.

[19] Ruay-Shiung Chang, Hui-Ping Chang, "A Dynamic Data Replication Strategy Using Access-weights in Data Grids," *Journal of Supercomputing*, vol. 45, no. 3, pp277-295, 2008.

[20] Anju Bala, Inderveer Chana, "VM Migration Approach for Autonomic Fault Tolerance in Cloud Computing," *International Conference on Grid and Cloud Computing and Applications*, pp3-9, 2013.

[21] Jun Yao, Ji-Wu Shu, Wei-Min Zheng, "Distributed Storage Cluster Design for Remote Mirroring Based on Storage Area Network," *Journal of Computer Science & Technology*, vol. 22, no. 4, pp521-526, 2007.

[22] Ma Fei, "*Research on Virtual Machine Placement and Live Migration in Cloud Data Center*," Beijing: Beijing Jiaotong University, 2013.

[23] Du Yuyang, "*Research on Virtual Machine State Migration and Phase-Change Memory Wear-Leveling Method*," Beijing: Tsinghua University, 2011.

[24] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, et al., "CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms," *Jounal of Software: Practice and Experience*, vol. 41, no. 1, pp23-50, 2010.