International Journal of Automation and Computing 9(2), April 2012, 142-154
DOI: 10.1007/s11633-012-0627-3

Resource Reconstruction Algorithms for On-demand Allocation in Virtual Computing Resource Pool

Xiao-Jun Chen[1] Jing Zhang[1, 2] Jun-Huai Li[1] Xiang Li[1]

[1]School of Computer Science and Engineering, Xi′an University of Technology, Xi′an 710048, PRC

[2]State Key Laboratory for Manufacturing Systems Engineering, Xi′an Jiaotong University, Xi′an 710048, PRC

Abstract: Resource reconstruction algorithms are studied in this paper to solve the problem of resource on-demand allocation and improve the efficiency of resource utilization in virtual computing resource pool. Based on the idea of resource virtualization and the analysis of the resource status transition, the resource allocation process and the necessity of resource reconstruction are presented. Resource reconstruction algorithms are designed to determine the resource reconstruction types, and it is shown that they can achieve the goal of resource on-demand allocation through three methodologies: resource combination, resource split, and resource random adjustment. The effects that the resource users have on the resource reconstruction results, the deviation between resources and requirements, and the uniformity of resource distribution are studied by three experiments. The experiments show that resource reconstruction has a close relationship with resource requirements, but it is not the same with current distribution of resources. The algorithms can complete the resource adjustment with a lower cost and form the logic resources to match the demands of resource users easily.

Keywords: Virtual computing systems, virtual computing resource pool, resource allocation, resource reconstruction, status transition, resource combination, resource split, resource adjustment.

1 Introduction

A virtual computing resource pool is usually constructed to manage the resources in virtual computing systems of clusters. But resource diversity and imbalance of their capabilities, together with the diversity of users′ resource requirements have produced a contradiction between tight coupling characteristics of resources and users′ multi-granularity resource requirements. It is known that the lack of computing environments for natural features of computing systems is still a bottleneck to restrict the effective use of resources[1]. As a result, a goal to construct a harmonious, secure, transparent, strong resource management in virtual computing systems is still far from achieving. It is a primary challenging issue that how to integrate a variety of heterogeneous physical resources and evaluate the performance of heterogeneous resources. It is a core issue in virtual computing systems that how to aggregate and organize resources for comprehensive utilization to maximize the effectiveness of limited resources[2]. A idea of resource virtualization presented in our study helps us to construct a logical view of resources via resource combination, which solves the contradiction between tight coupling characteristics of resources and multi-granularity resource requirements from users. Resource virtualization refers to underlying resource modules providing an interface to top modules. Thus, the resource utilization in virtualized environment is in conformity with that in non-virtualized platform. As a result, the resource users, the virtual machines in our studies, run in a virtualized resource interface. Resource virtualization, implementing the conversion from physical resources to logic resources through the construction and mapping, forms a view of virtual computing resource pool centered on resource users. It can reduce the complexity of resource utilization to solve the contradictions between the tight coupling characteristics and multiple granularities of resource requirements[3]. Virtualized resource allocation should allocate the appropriate granularities of resources based on resource requirements. Resource reconstruction, a key methodology for optimal allocation, recombines the physical resources that logic resources map to. It would constitute other granularities of logic resources matching with resource users to reduce the waste of resources, improve the resource utilization, satisfy more resource users, and keep workload balance in network and distributed systems[4].

Recent works in resource reconstruction mainly focus on: 1) resource management methodologies in computing systems[5−7]; 2) resource allocation policies[8−14]; and 3) resource reconstruction policies[15−18]. In addition, Wood et al.[19] determined a method implementing the mapping from physical resources to virtualized resources, which performs the dynamic adjustment of resources for virtual machines. Gmach et al.[20] combined the loads in servers and enabled the shared resources utilization with efficiency. These research results present the management and utilization methodology to organize a variety of resources and isolate the close dependence between hardware architecture and software from the view of resource on-demand configuration[21]. It is meaningful for us to create virtual machines on demands and construct a computing environment satisfying a variety of requirements from users. Such researches assumed that the demands of granularities from resource users are fixed, predicted and measured in advance, so the resources satisfying their service-level agreements (SLA) and quality of service (QoS) are allocated before the virtual machines are founded[22, 23], without con-

Manuscript received March 13, 2011; revised August 3, 2011
This work was supported by National High Technology Research and Development Program of China (863 Program) (No. 2007AA010305) and the Excellent Doctor Degree Dissertation Fund of Xi′an University of Technology (No. 102-211007).

sidering the dynamics in resource utilization according to the tasks in virtual machines.

The aim of resource reconstruction is to keep the logic resources in virtual computing resource pool concordant with the resource requirements at a higher service level. Thus, it is necessary for us to study the resource reconstruction algorithms with dynamics.

We present the status transition law in resource utilization in Section 2 and design the resource reconstruction algorithms in Section 3. The experiments are made in Section 4, and Section 5 concludes this paper.

2 Resource allocation in virtual computing resource pool

On-demand resource allocation refers to that virtual computing resource pool which not only satisfies the requirements from users, but also provides simple and convenient services quickly. Meanwhile, it provides an appropriate amount of resources to users to reduce their utilization cost and waiting time, and ultimately improves the efficiency of resources. Resource virtualization is to implement on-demand allocation. We discuss the thought of resource virtualization in this section.

2.1 Resource virtualization for on-demand allocation in virtual computing systems

Based on the characteristics of resources, we divide resources into physical resources and logic resources. Because of the non-tight coupling between logic resources and resource users, the complexity of logic resource is lower than that of physical resources from the view of resource users. The proposition of logic resource is based on following reasons:

1) A variety of physical resources would be allocated to resource users who require the computing services for a short period of time. If resource users use physical resource directly, the systems should compute the amount of resources before they are used. Such computation is much time-consuming, because resource on-demand allocation is not easy to achieve when facing a great number of resource users. Generally, resource requirements belong to a certain distribution law, thus, physical resources are combined into logic resource in accordance with this law. Therefore, the logic resource can be directly and easily used by resource users.

2) Different types of physical resources would lead to different effects in resource combination. We use the least amounts of physical resources to construct a group of logic resources with the maximum size by using on-demand combination.

3) The mapping from physical resources to logic resources can shield the heterogeneity of resources, thus, logic resources in virtual computing resource pool can serve to resource users as a uniform interface.

Resource virtualization can be described as follows: The low resource module provides an interface to high layer module, thus, the expected use environments of users are consistent with original systems. Resource virtualization, as shown in Fig. 1 includes: 1) five entities such as user, logic resources, resource mapping (RM), physical resources and devices, 2) the interface between resource users and logic resources, and 3) the interface between devices and physical resource.

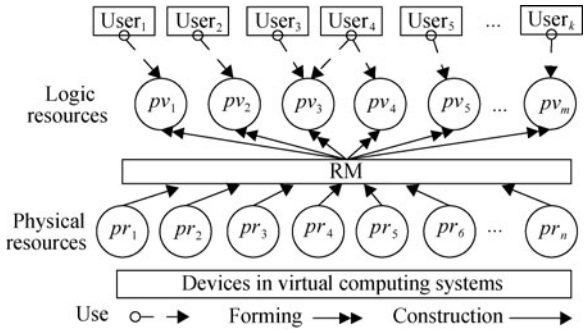


Fig. 1 Resource virtualization

We determine the size of logic resources based on the combination of the sizes from physical resources. The granularity distribution should satisfy the law of SLA distribution of resource users in virtual computing resource pool. When we allocate the logic resources to the queue of resource users, the suitable granularities of logic resources are allocated to match the requirements.

2.2 Resource status transition

The interfaces that the virtual computing systems provide to resource users include resource virtualization, resource reconstruction, resource allocation, resource utilization, resource monitoring, resource recycle, etc.[24] We use $PetriNET$ to describe the resource status transition: $RSPnet = (S, T, F)$. Where, S, the set of places, includes the status of resources; T includes the set of actions; and F, the relationship of S and T, reflects the path of status transition. In our study, we let S=(Physics, Ready, Requesting, Responding, Executing, Waiting, Completed), T=(Combine, Reconstruct, Distribute, Accept, Reject, Use, Stop, Preemp, Finish, Recycle), thus, the resource status transition can be described in Fig. 2. "Physics" is the physical resources oriented systems. The action "Combine" completes the transition from "Physics" to "Ready", a preparing status of logic resources. Otherwise, the action "Reconstruct" makes resource reconstruction to transfer the status to "Physics". The action "Distribute" sends out the command to allocate resources to resource users, then, the resources are in the status of "Requesting". In this situation, if resource users "Accept" the resources, the status becomes "Responding". Otherwise, the status becomes "Ready" by using the action "Reject". Resource users use the resources through "use" action, then, the status is transferred as "Executing". In "Executing" process, if the resources utilization requires stopping temporarily, resource users need to make "Stop" action, thus, the status becomes "Waiting". If resources are used, the "Use" action should be adopted again, and the status becomes "Executing". If it has long waiting time, the systems would execute the action "Preemp" to release the waiting resources in force, then, the status returns to "Ready". After the

resources are completed, the systems call the "finish" action, then, the status is transferred to "Completed". The resources existing in the status of "Completed" are released by action "Recycle", and then, they return to "Ready" status.

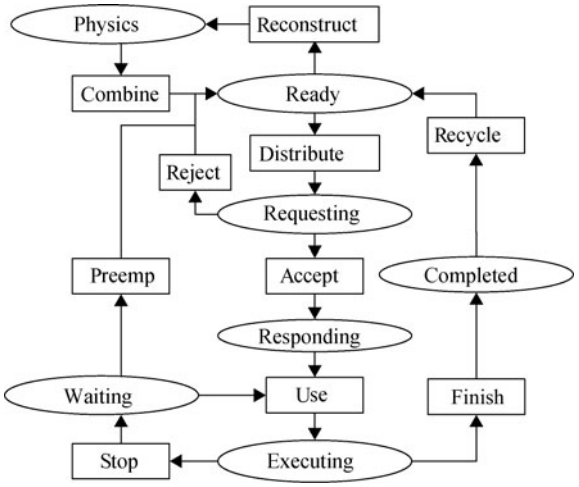


Fig. 2 Resource status transition

The status transition law in resource allocation process can be used to forecast the status of resources in next time[25]. In the systems oriented continuous resource utilization, each logic resource would be in one status of them. In general, "Physics", "Ready", "Executing" have the biggest proportion, and others may be instantaneous since they are temporary statuses. We take the statuses of "Ready", "Physics" as the focus in this paper and mainly study the "Reconstruct" action. If resource requirements change greatly, no reconstruction would lead to low service level and high resource expenditure. The resource reconstruction is from three demands: 1) The granularities of logic resources in virtual computing resource pool cannot satisfy the demands of resource user, so physical resources should be recombined to form the logic resources with greater granularity. 2) The granularities of resources in systems are greater than that of resource users, thus, resource reconstruction can reduce the waste of resource. 3) the distribution of logic resources in systems cannot satisfy the law of requirements from resource users[26], as a result, resource reconstruction can solve this problem.

3 Algorithms design for resource reconstruction

The nature of resource reconstruction is the adjustment of the resources′ granularities. The resource reconstruction algorithms require us to consider at least two factors: the requirements from resource users and the reconfiguration of resources. The requirements are the key factor. Besides requirements, the resource reconstruction also has a close relationship with the features of logic resources, such as the deviations between present resources granularities and the required granularities, and the uniformity level of resource distribution[27].

3.1 Key data structures

We present three key data structures to implement the algorithms:

Description 1 (The queue of resource users). The queue of resource users is a tuple with several elements, $VCRS = (number, VCR_1, VCR_2, VCR_n, currentVCR)$. Where, number is the sequence number of this queue by the order of time, VCR_i is the resource users, and $currentVCR$ is the current requirement from resource users. VCR_i is a tuple with three elements $VCR_i = (user, e, r)$, of which, $user$ is a task, e is the expectation of resource′s granularity, and r is the granularity of VCR_i.

Description 2 (Virtual computing resource pool). Virtual computing resource pool is the set of logic resources. Let the quantity of logic resources be m, then, we take $PVS = (pv_1, pv_2, pv_m)$ as the description of this set, of which, $pv = (no, description, function, r)$, where, no is the number of a logic resource, $description$ is the semantic description, $function$ is the reconfiguration of resource, and r is the size of granularity.

Description 3 (Resource mapping). Resource mapping is a tuple with three elements $RM = (PRS, PVS, MPS)$, of which, PRS is the set of physical resources, $PRS = (pr_1, pr_2, pr_k)$, PVS is the set of logic resources, $PVS = (pv_1, pv_2, pv_m)$, and MPS is the mapping from physical resources to logic resources, $MPS = (mp_1, mp_2, mp_l)$. Where, $mp_i = (pr, pv)$, of which, pr is a physical resource in systems, and pv is logic resource in systems.

3.2 Description of algorithms

We present the resource reconstruction algorithms with dynamics in this section. The dynamics of resource reconstruction algorithms is described from two aspects. For one thing, algorithms perform resource reconstruction periodically after the systems start. The resources in systems and the queue of resource users are scanned in a time interval to determine the type of resource reconstruction at present. Dynamics are embodied in the sensitive response to virtual computing resource pool, the persistence of algorithms scanning, and the timeliness resource adjustment. For another thing, these algorithms not only require dealing with the resources with small granulites which cannot satisfy the requirements, but also need to deal with the resources with great granulites which waste many resources. In addition to those, algorithms also solve the problem that the granulites of resources are too uniform to adapt to the law of requirements from resource users. Thus, some dynamic policies, such as resource combination, resource split and resource random adjustment are adopted to keep the status of overall resources consistent with resource users.

1) Resource reconstruction types.

Resource reconstruction wastes a great deal of resources, so we should satisfy resource users through current virtualized resources[28]. The granularities of logic resource adapting to requirements are judged whether there is a resource satisfying the requirement. Then, we classify the resource reconstruction as three types: resource combination, resource split, and resource random adjustment. When the granularities of logic resources are too small, resource com-

bination satisfies the great granularities in virtual computing resource pool by combining several small resources into a big resource. When the granularities of logic resources are too great, resource split reduces the waste of resources by splitting a big resource into small slices. When the granularities of logic resources are too uniform, resource random adjustment adapts to resource distribution with fluctuations by adjusting the uniform resources into the resources with random distribution. Resource reconstruction types can be determined in Fig. 3.

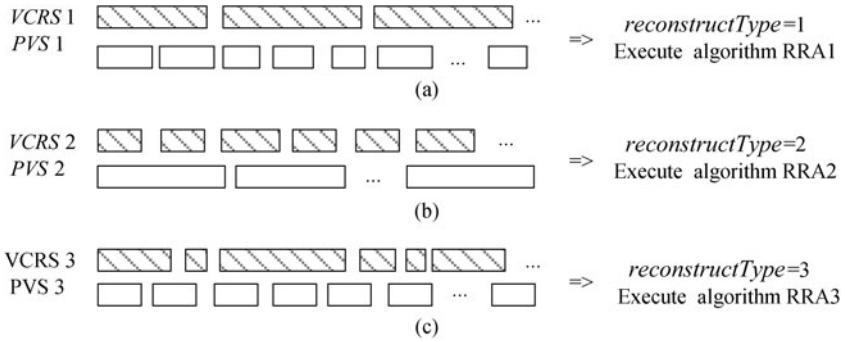


Fig. 3 The determination of resource resconstruction types

The algorithm DRRA completes the determination of resource reconstruction types in below pseudo codes of Fig. 4:

```
Algorithm (DRRA): The determination of resource
 reconstruction types
Input: PVS, VSRS, dis1, dis2
Output: reconstructType
1    isReconstruct←true;
2    Foreach (pv in PVS)
3       If (pv.r>VCRS.currentVCR.r)
4           isReconstruct←false; break;
5       Endif
6    Endfor
7    If(isReconstruct)
8       Return reconstructType←1
     EndIf
10   dis←∞
11   Foreach(pv in PVS)
12      if(pv.r- VCRS.currentVCR.r>0
13            and pv.r-VCRS.currentVCR.r>dis1)
14        dis←pv.r-VCRS.currentVCR.r;
15      Endif
16   Endfor
17   If(dis>dis1)
18      Reture reconstructType←2
19   Endif
20   sumdisPVS←0;
21   For (i←2; i<PVS.count; i++)
22      sumdisPVS+←|PVS[i].r-PVS[i-1].r|;
23   Endfor
24   sumdisVCRS←0;
25   For (i←2; i<=VCRS.count; i++)
26      sumdisVCRS+←|VCRS[i].r-VCRS[i-1].r;
27   Endfor
28   If (|sumdisVCRS- sumdisPVS |>dis2)
29      Return reconstrestTYPE←3
30   Else
31      Return reconstructType←0
32   Endif
```

Fig. 4 The algorithm of resource resconstruction types determination

The input parameters of DRRA are the reconfigurable logic resources PVS, the queue of resource users $VCRS$, and the limit dis_1 and dis_2, where, dis_1 decides whether the big resources need to split, and dis_2 decides whether the uniform logic resources need to adjust. In rows 1–9 of Fig. 4, the algorithm determines that resource combination should be carried out by traversing all elements in PVS, then, it returns $reconstructType = 1$. In rows 10–19, if the deviation between the expected granularity and maximum granularity exceeds the limit, it returns $reconstructType = 2$. In rows 20–31, the algorithm determines that whether the expecting granularities of requirements are too uneven. If the logic resources are more uniform than the limit, it returns $reconstructType = 3$. Besides to those, the system returns $reconstructType = 0$, which means that the resources are not required to reconstruct at present. It is obvious that the time complexity of the DRRA algorithm is O $(PVS.count + VCRS.count)$, because there is only one cycle in the traversal of PVS and $VCRS$.

The system determines the resource reconstruction algorithms based on the result of DRRA. The statuses of logic resources are scanned to select the resources in "Physics" status. And then, RRA1, RRA2, and RRA3 algorithms are executed respectively to implement resource combination, resource split, and resource random adjustment, which make the logic resources satisfy resource users with better granularities.

2) The reconstruction of resource combination

If DRRA returns $reconstructType = 1$, resource combination is completed by using the pseudo codes RRA1 in Fig. 5.

RRA1 algorithm aims to find out a combination of logic resources in PVS, which make their sum of granularities approaches to $VCRS.currentVCR.r$ as much as possible. In row 2, the quantities of logic resources selected are from 1 to $PVS.count$. For any selection i, we let the quantity of selection methods be $number$. In selection k, $getElements$ function is called to obtain their combinations. The function $getElements$ has implemented a recursive algorithm to select i elements from $VCRS.count$ elements in PVS.

We suppose that $n = VCRS.count$ (number as $1, 2, n$), and $m = i$. The path of the combination with m resources selected from n resources is shown in Fig. 6. Each combination approach for nodes in the tree from root to one of the leaves is thought as a selection method. In rows 6–11, the selection methods are pairwisely compared to determine the quantity and the number of the closest combination with present requirement.

Since the time complexity of $CiVCRS.count$ in row 3 is $\mathrm{O}(\min(i, PVS.count - i))$, and the time complexity of rows 4–12 is also $\mathrm{O}(\min(i, PVS.count - i))$, thus, it is obvious that the time complexity of RRA1 algorithm is $\mathrm{O}([\min(i, PVS.count - i)]2.PVS.count)$, because there is only one cycle in the traversal of PVS.

3) The reconstruction of resource split

If DRRA returns $reconstructType = 2$, resource split is completed by using the pseudo codes RRA in Fig. 7.

```
Algorithm (RRA1): Resource combination
Input: VCRS, PVS
Output: PVS
1    minl←0; minK←0; mindisR←+∞;
2    For(i←1, i<=PVS.count; i++)
3      number←C^i_PVS.count
4      For(k←0; k<number; k++)
5        ArrayList choosepvs←PVS.getElements(i, k)
6        SumR←sum r of elements in choosepvs;
7        if(SumR> VCRS.currentVCR.r
8              and SumR-VCRS.currentVCR.r< mindisR)
9            minl←i; minK←k;
10           mindisR← SumR- VCRS.currentVCR.r;
11         Endif
12       Endfor
13     Endfor
14     choosepvs←PVS.getElements (minl, minK);
15     PVS.combine(choosepvs);
```

Fig. 5 Resource combination algorithm

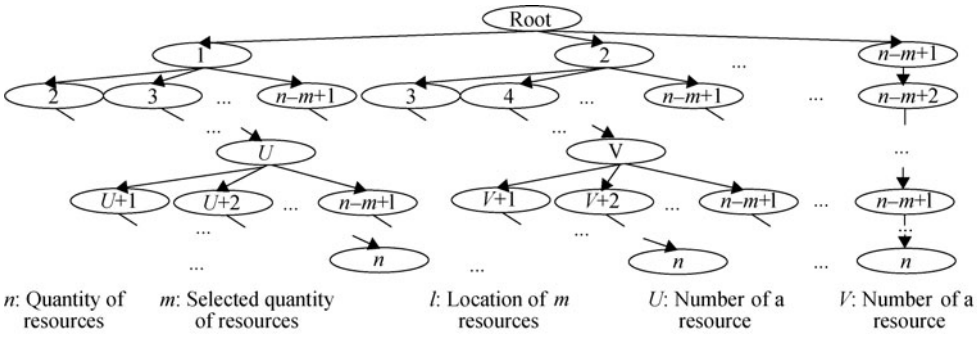


Fig. 6 The tree formed in resources selection

```
Algorithm (RRA2): Resource split
Input: VCRS, PVS
Output: PVS
1    dis←PVS[1].r-VCRS.currentVCR.r
2    dispv←PVS[1];
3    Foreach(pv in PVS)
4      if(pv.r-VCRS.currentVCR.r>0
5            and pv.r-VCRS.currentVCR.r>dis)
6        dis←pv.r-VCRS.currentVCR.r; dispv←pv;
7      Endif
8    Endfor
9    while(dispv.r>0)
10     If(dispv.r> VCRS.currentVCR.r)
11       pvnew←dispv.cut(VCRS.currentVCR);
12     Else
13       pvnew←dispv;
14       Dispv.r←0;
15     Endif
16     PVS.add(newpv);
17   Endwhile
18   PVS.remove(dispv);
```

Fig. 7 Resource split algorithm

In rows 1–8 of RRA2, the system traverses all elements in PVS to find out the $dispv$, which has the maximum deviation. In rows 9–17, the system loops to split $dispv$ into several logic resources, which have the same granularity as $currentVCR$. Thus, the granularities of logic resources in PVS would match that of $currentVCR$. It is obvious that the time complexity of the RRA2 algorithm is $\mathrm{O}(\max[PVS.count, \max(PVS[i].r)/VCRS.currentVCR.r])$.

4) The reconstruction of resource adjustment

If DRRA returns $reconstructType = 3$, resource random adjustment is completed by using the pseudo codes RRA3 in Fig. 8.

```
Algorithm (RRA3): Resource random adjustment
Input: VCRS, PVS
Output: PVS
1    pvs←new pvs ();
2    For (pv in PVS)
3      If(pv.r -VCRS.currentVCR.r>0) and
4        pv.r -VCRS.currentVCR.r< VCRS.currentVCR.r/2)
5          pvs.add(pv);
6      Endif
7    Endfor
8    SumR←sum r of elements in pvs;
9    Produce random data for each pv in pvs;
10   SR←sum random data in pvs;
11   For (pv in pvs)
12       Compute weigh of random data for pv in SR;
13       pv.r←SumR*weigh;
14   Endfor
15   PVS.replace(pvs);
```

Fig. 8 Resource random adjustment

In rows 1–7 of RRA3, based on the requirements from current resource users, the deviations of granularities from present logic resources are computed. And then, pvs, which has uniform granularities, are selected from PVS. In row 8, we compute the sum of logic resources in pvs. In rows 9–10, we produce a random data for every element in pvs and compute a sum of these random data. In row 11–14, the weights of such random data in SR are computed to determine the proportions of resources in pvs. In row 15, the logic resources adjusted are put into PVS to replace the old ones. It is obvious that the time complexity of the RRA1 algorithm is $\mathrm{O}(PVS.count)$, because there is only one cycle in the traversal of PVS.

The algorithm DRRA requires combining with algorithm RRA1, RRA2, and RRA3 closely. Because RRA1, RRA2, and RRA3 are decided by DRRA, they should be implemented as a whole for resource reconstruction in practice. We make some experiments to determine their features in next section, and then discuss their application in a resource management framework of virtual computing systems used in our earlier study.

4 Experimental analysis

We analyze the resource reconstruction process caused by above algorithms in below experiments. To prepare the experimental data, a group of logic resources in virtual computing resource pool are created and taken as the input data: PVS. Let the set of physical resources combined into logic resources be $prs = (pr_1, pr_2, \cdots, pr_N)$, of which, $pr_1, pr_2, \cdots, pr_N$ are the quantities of physical resources, and N is the quantity of types for these resources. We define $pv.r$ as: $r = \eta \sqrt[N]{\prod_{j=1}^{N} pr_j}$, where, η, the proportional coefficient related to the performance of hardware. It reflects the change of granularities after the physical resources are combined into logic resources.

We suppose that there are 100 hosts in virtual computing resource pool. Let the resources′ types in each host be CPU, memory, disk and network, and let η be 1.2, then, the initial virtual computing resource pool with 32 quantities of resources is shown in Table 1 and divided into 9 kinds.

We then obtain the input data: $VCRS$. The experiments suppose that the resource users arrive one by one. Since the randomness of resource users satisfies the normal distribution, we produce six kinds of resource distribution through random data shown in Fig. 9, whose means and variances are $(\mu, \sigma) = (10, 10)$, $(12, 12)$, $(15, 15)$, $(18, 18)$, $(20, 20)$, and $(22, 22)$, respectively. 150 resource users from requirements arrive in turn. We set the resource users as the $currentVCR$ in $VCRS$ one by one, and then input them into DRRA.

We develop the testing programs of DRRA, RRA1, RRA2, and RRA3 in java language.

The goals of our experiments aim to determine the effect that $VCRS$, dis_1, and dis_2 have on resource reconstruction under the condition of fixed logic resources in virtual computing resource pool. We conclude the trends of the distribution of resource reconstruction types, the change of the quantity after resource reconstruction, as well as the change of the resource distribution after resource reconstruction. $VCRS$, six groups of data in Fig. 9, reflects the granularities of resource users. The dis_1, the limit of deviation between logic resources and resource users decides whether there is a need to split the resources. We set dis_1 as 0, 0.1, 0.2, 0.3, 0.4, and 0.5, respectively, in our experiment. The dis_2, the limit of deviation for the distribution of resources, decides whether their resources granularities are required to make adjustment. We also set dis_2 as 0, 0.1, 0.2, 0.3, 0.4, and 0.5, respectively, again in our experiment.

4.1 The distribution of resource users

We set dis_1=0.2 and dis_2=0.3 to determine the effect that the requirements in the queue of resource users have

Table 1 The initial PVS

pv	pv_1	pv_2	pv_3	pv_4	pv_5	pv_6	pv_7	pv_8	pv_9
r	35.68	22.45	27.92	21.35	35.68	21.41	37.76	23.30	22.39
$quantity$	8	4	4	4	3	3	1	2	3

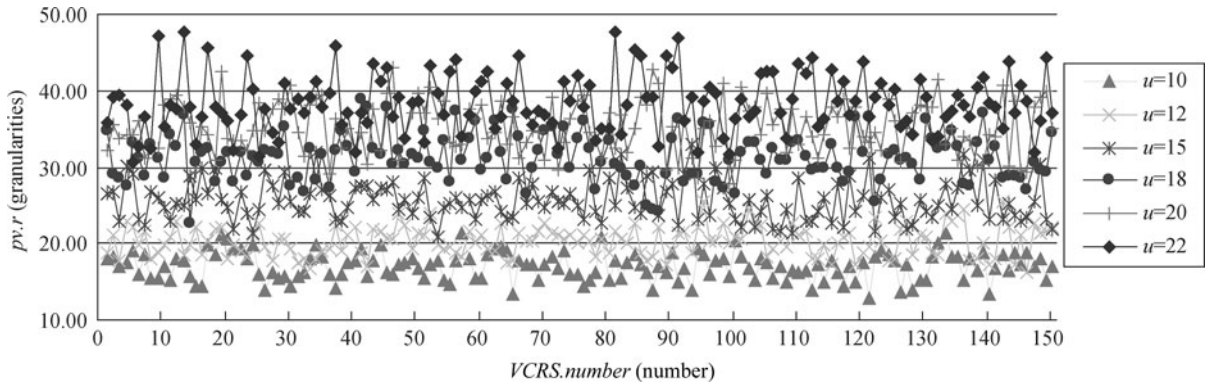


Fig. 9 The distributions of six resource requirements

on resource reconstruction algorithms in this section. Six groups of data in Fig. 9 are inputted into algorithm DRRA to run, then, we collect the results of resource reconstruction types from the output of algorithm DRRA. The occurrences of $reconstructType$=$(0, 1, 2, 3)$ in different u=$(10, 12, 15, 18, 20, 22)$ are shown in Fig. 10.

It is seen from Fig. 10 that, no matter what the scopes of resources granularities are, $reconstructType = 0$ always has the maximum times and $reconstructType = 1$ always has the minimum times in four resource reconstruction types. $ReconstructType = 0$ does not need to reconstruct and the loads of $reconstructType = 1$ need the greatest reconfigurable resources in all types. With the increase of resources granularities, in a certain limit of granularity (u=18 in this experiment), $reconstructType = 3$ presents the trend of significant increase, and $reconstructType = 1$ presents the trend of slight increase. But $reconstructType = 0$ presents the trend of significant increase, and $reconstructType = 3$ presents the trend of fluctuations on a certain scope. Much uniform granularities and the small granularities of resources become the main problems in our virtual computing resource pool with the time going on. We see that $reconstructType = 0$ should be selected as the main type in determining the resource reconstruction. Since the resource reconstruction cause a great deal of cost and RRA1 has the maximum space and time complexity, $reconstructType = 1$ should be avoided as much as possible.

The changing trend of the total quantity of resources after reconstruction from 0 to 150 is shown in Fig. 11.

It is seen from Fig. 11 that, the more the distribution (μ, σ) approaches to $VCRS$, the more the quantity of resources after reconstruction are, and it is unrelated to the scope of different resource requirements. Taking a limit of a certain granularity (μ=12 in this experiment) as an example, the quantity of resources after reconstruction presents the trend of increase. If the resources have more granularities in μ=(18, 20, 22) than that in μ=12, the quantity would present the trend of stability after enough times of reconstruction. Generally, the more the granularities from resource users are, the less are the quantities of resources after reconstruction. The resource distribution after reconstruction in $VCRS.number$=$(0, 30, 30, 90, 120, 150)$ is shown in Fig. 12.

It is seen from Fig. 12 that the granularities of resource distribution present the trend of divergence with the increase of $VCRS.number$, because there would emerge fewer wastes in resource utilization in the scope with more granularities. With the time going on, greater u would lead to more divergent of resources granularities. For each of requirement in $VCRS.number$, the smaller u is, the less the granularities of resources are. It is proved that the resource reconstruction can satisfy the resource requirements, and our algorithms can process resource distribution from different resource requirements.

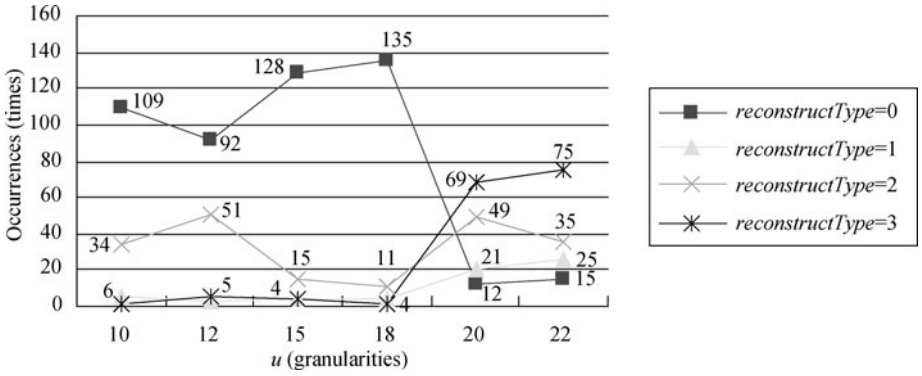


Fig. 10 Occurrences of $reconstructType$=$(0, 1, 2,$ and $3)$ in different resource requirements

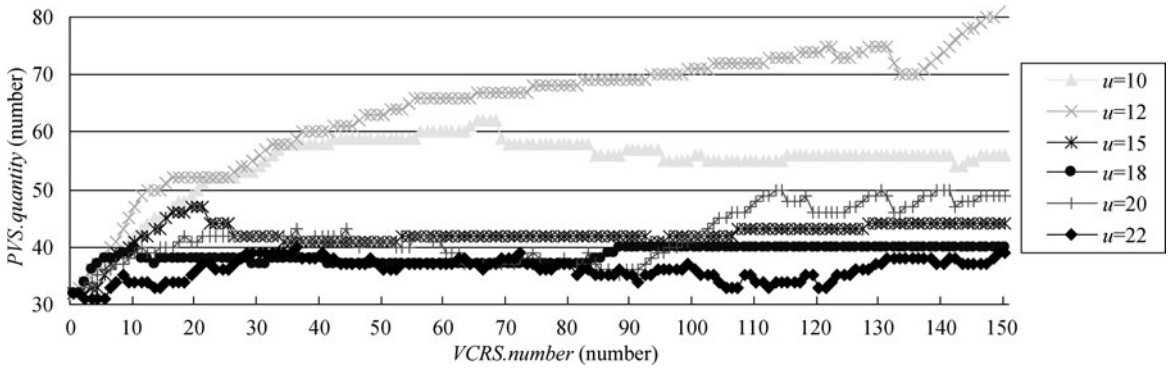


Fig. 11 The trends of quantity of resources after reconstruction in different resource requirements

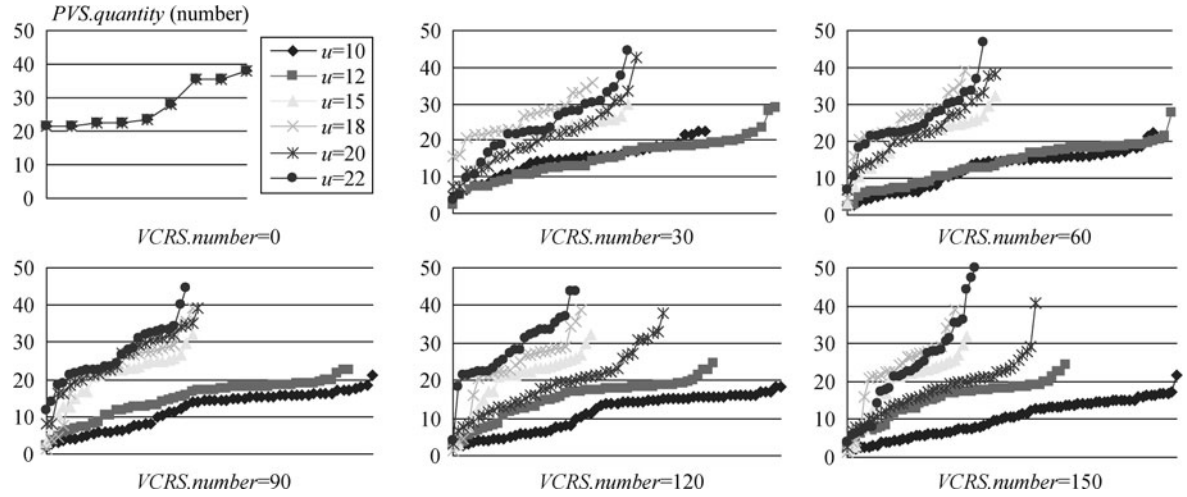


Fig. 12 Resource distributions after reconstruction in different resource requirements

4.2 The deviation between resources and requirements

We set $VCRS = (\mu, \sigma)$=(18, 18) and dis_2 =0.3 to determine the effect that the deviation between resources and requirements has on the resource reconstruction algorithms. We input the data of dis_1=0, 0.1, 0.2, 0.3, 0.4, 0.5, respectively, into algorithm DRRA to run, and collect the results of reconstruction types. The times of $reconstructType$=(0, 1, 2, 3) in different deviations dis_1=(0, 0.1, 0.2, 0.3, 0.4, 0.5) are shown in Fig. 13.

It is seen from Fig. 13 that $reconstructType$ = (0, 1, 2, 3) present the trend of fluctuations in different dis_1=(0, 0.1, 0.2, 0.3, 0.4, 0.5). The $reconstructType$ = 0 has the maximum fluctuations, and $reconstructType$ = 1 has the minimum fluctuations, which are the same as those in different resource requirements. When dis_1 = 0.2 and dis_1 = 0.4, $reconstructType$ = 0 has greater times of reconstruction than others. The fewer times of reconstruction in $reconstructType$ = 1 would produce the lowest resource reconstruction cost. It is seen from Fig. 13 that there is almost no effect that the deviation between resources and requirements has on our resource reconstruction. The goal of resource reconstruction aims to enable the granularities of logic resources close to the requirements from resource users, thus, reduce the waste of resources.

The changing trend of the quantity of resources after reconstruction from dis_1 = 0 to 0.5 is shown in Fig.14.

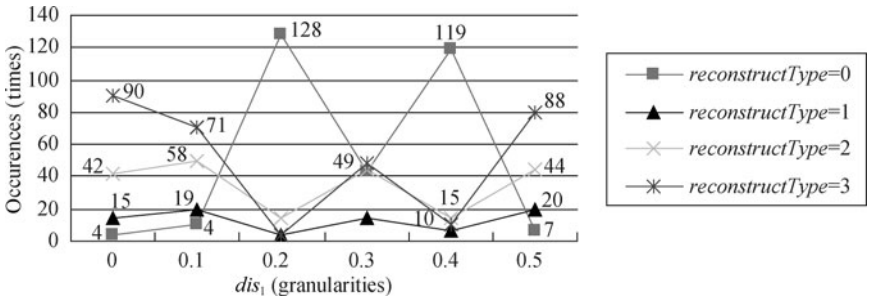


Fig. 13 Times of $reconstructType$=(0, 1, 2, and 3) in different deviations

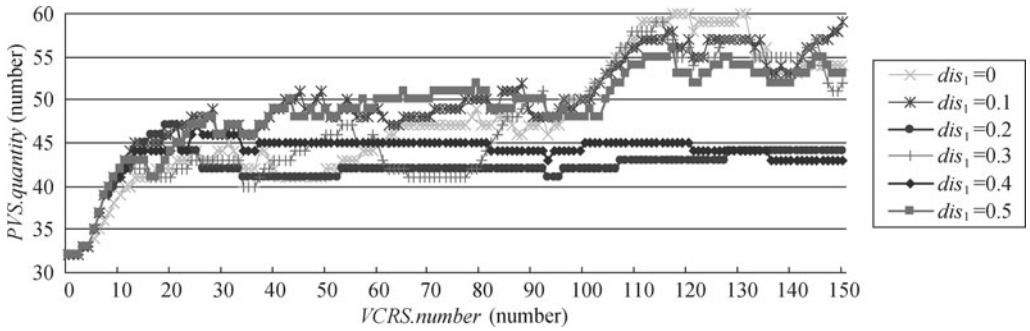


Fig. 14 The trends of quantity of resources after reconstruction in different deviations

It is seen from Fig. 14 that the quantity of resources after reconstruction presents the trend of increase in initial phase. But it presents stability after several times of reconstruction. The deviation between resources and requirements has no effect on resource reconstruction.

The resource distribution after reconstruction in $VCRS.number$=(0, 30, 30, 90, 120, 150) is shown in Fig. 15.

It is also seen from Fig. 15 that there is no much change in the distribution of logic resources with the time going on and the increase of $VCRS.number$ in different deviations.

4.3 The uniformity of resource distribution

We set $VCRS = (\mu, \sigma)$=(18, 18) and dis_1 =0.2 to determine the effect that the uniformity of resource distribution has on the resource reconstruction algorithms. We input the data of dis_2 =0, 0.1, 0.2, 0.3, 0.4, 0.5, respectively, into algorithm DRRA to run, and collect the results of reconstruction types. The times of $reconstructType$=(0, 1, 2, 3) in different deviations dis_2=(0, 0.1, 0.2, 0.3, 0.4, 0.5) are shown in Fig. 16.

It is seen from Fig. 16 that $reconstructType$=(0, 1, 2, 3) presents the trend of fluctuations in different dis_2=(0, 0.1, 0.2, 0.3, 0.4, 0.5). The $reconstructType$ = 0 has the maximum fluctuations and $reconstructType$ = 1 has the minimum fluctuations, which is the same as those in different resource requirements. When dis_2 = 0.1 and dis_2 = 0.3, $reconstructType$ = 0 has greater times of reconstruction than others. The fewer times of reconstruction in $reconstructType$ = 1 would produce the lowest resource reconstruction cost. It is seen from Fig.16 that there is almost no effect that the uniformity level of resource distribution has on resource reconstruction. Resource reconstruction aims to enable the granularities of logic resources close to the requirements of resource users, thus, reduce the waste of resources.

The changing trend of the quantity of resources after reconstruction from dis_2 = 0 to 0.5 is shown in Fig. 17.

It is seen from Fig. 17 that the quantity of resources after reconstruction presents the trend of increase in initial phase, but it presents stability after several times of reconstruction. The uniformity level of resource distribution has no effect on resource reconstruction.

The resource distribution after reconstruction in $VCRS.number$=(0, 30, 30, 90, 120, 150) is shown in Fig. 18.

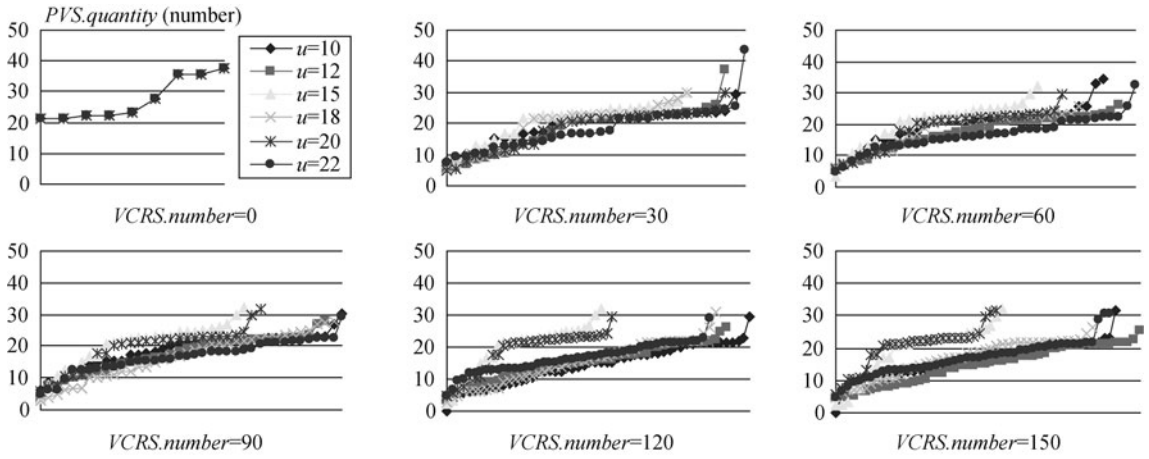


Fig. 15 Resource distributions after reconstruction in different deviations

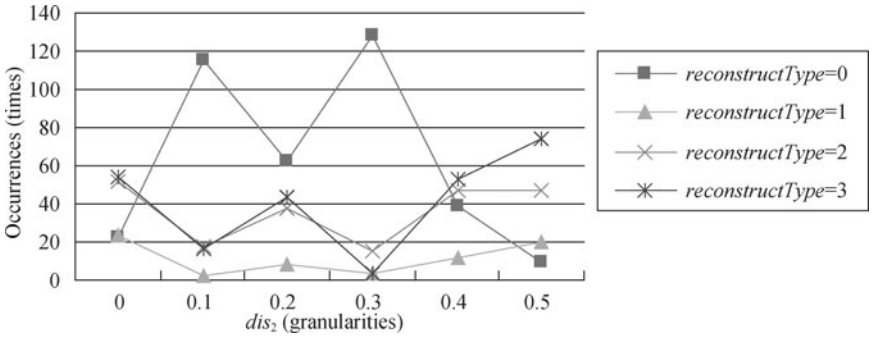


Fig. 16 Time of $constructType$=(0, 1, 2 and 3) in different uniformity levels

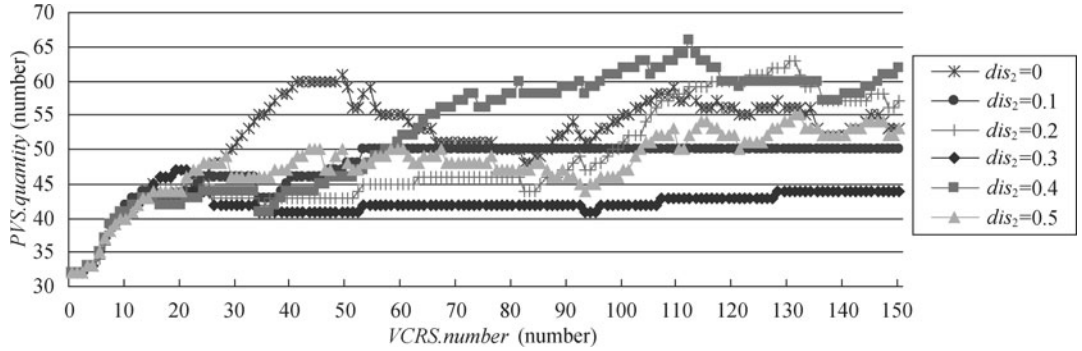


Fig. 17 The trend of quantity of resources after reconstruction in different uniformity levels

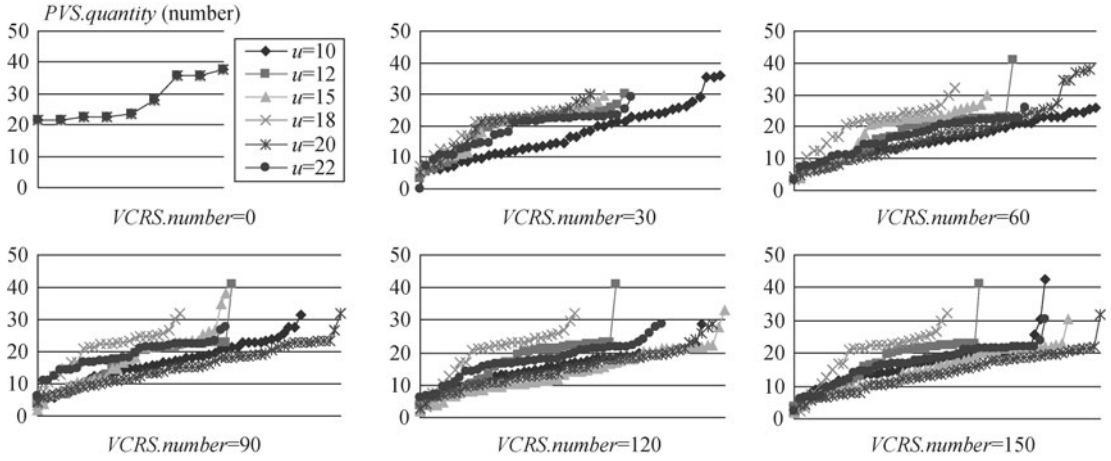


Fig. 18 Resource distributions after reconstruction in different uniformity levels

It is also seen from Fig. 18 that there is no much change in the distribution of logic resources with the time going on and the increase of $VCRS.number$ in different uniformity levels.

It is concluded from three experiments that resource reconstruction algorithms are very necessary to adjust the granularities of resources for virtual computing resource pool. Resource reconstruction not only can satisfy the requirements of resource users, but also can reduce the waste of resources and improve the resource utilization.

5 The application of algorithms

Resource management, the essential function of virtual computing systems, is usually used in production scheduling systems[29], network supported collaborative design systems[30], etc. Combined with results in earlier study, we present a resource management framework in this section. Resource reconstruction algorithms, as an indivisible part of systems, are implemented in this framework. Resource management framework shown in Fig. 19 composes of computing nodes, management node, and client in cluster. All nodes in cluster are connected by high-speed network. The components are expressed as follows: In computing nodes, $Xenhypervisor$ is taken as the virtual machines monitor (VMM), in which, several virtual machines (named as domain in $Xenhypervisor$) are created in each computing node. The domain $VM0$, a virtual machine used for the management of $Xenhypervisor$. Other domains are used for providing services to users. A daemon named as $Xend$ runs in $VM0$ is a service of virtualized platform. $Nodectled$, a node controller developed by us, communicates with Xend by Xen API to manage the host. $Nodectled$ transmits the resource data to resource manager running in management node. Meanwhile, $Nodectled$ receives the requests from resource manger to execute the commands. Resource manager, the core of framework, composes of the components, such as resource collection, resource virtualization, resource database, etc. Resource manager also implements a dynamic resource management by using the technology of timer, multithreads and service-thread[31]. The client calls web server or $XMLRPCServer$ to access resource manager to make the query and operation based on resource status.

Resource reconstruction algorithms, implemented as an important group of subcomponents in resource manager, have the same life cycle as resource manager. Resource reconstruction algorithms, together with resource virtualization, play a key role to implement the optimal allocation for virtual computing systems. Our earlier study proved that these components can allocate the resources in virtual computing resource pool efficiently.

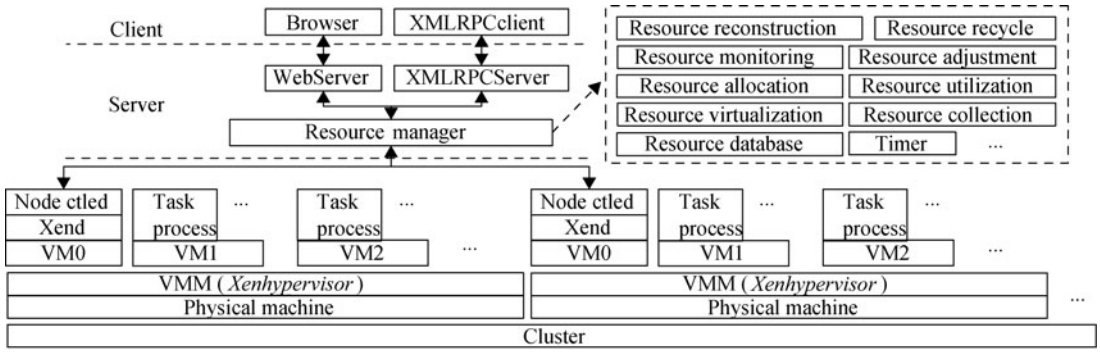


Fig. 19 Resource management framework

6 Conclusions

The resource reconstruction for on-demand allocation in virtual computing resource pool was studied in this paper to enable the distribution of logic resources more satisfy the requirements of resource users with a high satisfaction and resource utilization. We designed the resource reconstruction algorithms to implement the goal of on-demand allocation through resource combination, resource split and resource random adjustment. We verified the algorithms by making three experiments. The theory and the results of experiments conclude that: 1) Resource reconstruction, such as the distribution of resources after reconstruction, has close relationships with the queue of resource users. 2) The deviation between resources and requirements and the uniformity level of resource distribution almost have no effect on the resource reconstruction algorithms. We can change the algorithms to remove the input parameters dis_1 and dis_2, or set the constant values of them in the algorithms. 3) Resource reconstruction aims to achieve the goal of on-demand allocation with rationality. We must avoid the emergency of great and small granularities of resources, and then increase the fluctuations of the resource distribution. 4) We should ensure $reconstructType = 0$ and avoid $reconstructType = 1$ to reduce the reconstruction cost. The results show that our algorithms can solve the problem of resource reconstruction for on-demand allocation with high efficiency in virtual computing resource pool.

Acknowledgement

We thank for the editors of IJAC and the anonymous reviewers for their helpful suggestions on the quality improvement of our present paper.

References

[1] H. Jin, X. F. Liao. Virtualization technology for computing system. *China Basic Science*, vol. 10, no. 6, pp. 12–18, 2008. (in Chinese)

[2] A. Tang, Z. Liu, C. H. Xia, Z. Li. Distributed resource allocation for stream data processing. In *Proceedings of International Conference on High Performance Computing and Communications*, Munich, Germany, pp. 91–100, 2006.

[3] X. Zhang, M. F. Zhu, L. M. Xiao. Research on virtualization technology of distributed I/O resource. *Microelectronics and Computer*, vol. 25, no. 10, pp. 178–181, 2008. (in Chinese)

[4] X. M. Tang, J. S Yu. Feedback scheduling of model-based networked control systems with flexible workload. *International Journal of Automation and Computing*, vol. 5, no. 4, pp. 389–394, 2008.

[5] A. R. Molina, A. Ponniah, J. Simcock, M. S. Irwin, C. M. Malata. Resource implications of bilateral autologous breast reconstruction — A single centre′s seven year experience. *Journal of Plastic Reconstructive and Aesthetic Surgery*, vol. 63, no. 10, pp. 1588–1591, 2010.

[6] R. R. Huang, W. Xue, J. W. Shu, W. M. Zheng. Storage performance virtualization under out-of-band structure. *Journal of Chinese Computer Systems*, vol. 28, no. 6, pp. 1139–1143, 2007.

[7] G. Y. Zhang, J. W. Shu, W. Xue, W. M. Zheng. A persistent out-of-band virtualization system. *Journal of Computer Research and Development*, vol. 43, no. 10, pp. 1842–1848, 2006. (in Chinese)

[8] W. Yu, J. Wang. Scalable network resource management for large scale virtual private networks. *Simulation Modelling Practice and Theory*, vol. 12, no. 3–4, pp. 263–285, 2004.

[9] A. A. Chien, N. Taesombut. Integrated resource management for lambda-grids: The distributed virtual computer (DVC). *Future Generation Computer Systems*, vol. 25, no. 2, pp. 147–152, 2009.

[10] K. Zhou, X. J. Tong, W. B. Liu. Sensitivity analysis of source management. *Journal of Huazhong University of Sci-*

ence and Technology, vol. 34, no. 8, pp. 122–124, 2006. (in Chinese)

[11] K. Zhou, X. J. Tong, Z. H. Gao, Q. S. Gao. Analysis and implementation of mathematica-based algorithm for source management. *Journal of Huazhong University of Science and Technology (Nature Science)*, vol. 34, no. 7, pp. 57–59, 2006. (in Chinese)

[12] M. Ghobadi, S. Ganti, C. S. Gholamali. Resource optimization algorithms for virtual private networks using the hose model. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, vol. 52, no. 16, pp. 3130–3147, 2008.

[13] Y. T. Lu, N. Xiao, X. J. Yang. Scalable resource management system for high productive computing. In *Proceedings of the 3rd China Grid Annual Conference*, IEEE, Dunhuang, PRC, vol. 3, pp. 331–337, 2008.

[14] Z. K. Wang, Y. Chen, D. Gmach, S. Singhal, B. J. Watson, W. Rivera, X. Zhu, C. D. Hyser. AppRAISE: Application-level performance management in virtualized server environments. *IEEE Transactions on Network and Service Management*, vol. 6, no. 4, pp. 240–254, 2009.

[15] S. S. Thamarai, R. A. Balachandar, R. Kumar, P. Balakrishnan, K. Rajendar, R. Rajiv, G. Kannan, G. R. Britto, E. Mahendran, B. Madusudhanan. CARE resource broker: A framework for scheduling and supporting virtual resource management. *Future Generation Computer Systems*, vol. 26, no. 3, pp. 337–347, 2010.

[16] T. Li, Y. L. Yang. Algorithms of reconfigurable resource management and hardware task placement. *Journal of Computer Research and Development*, vol. 45, no. 2, pp. 375–382, 2008. (in Chinese)

[17] F. Song. Failure-aware resource management for high-availability computing clusters with distributed virtual machines. *Journal of Parallel and Distributed Computing*, vol. 70, no. 4, pp. 384–393, 2010.

[18] Y. Liao, X. D. Chen, N. Sang, L. H. Hu, G. Z. Xiong, Q. X. Zhu. Adaptive resource management middleware in distributed real-time systems. *Journal of University of Electronic Science and Technology of China*, vol. 37, no. 1, pp. 101–104, 2008. (in Chinese)

[19] T. Wood, P. Shenoy, A. Venkataramani, M. Yousif. Sandpiper: Black-box and gray-box resource management for virtual machines. *Computer Networks*, vol. 53, no. 17, pp. 2923–2938, 2009.

[20] D. Gmach, J. Rolia, L. Cherkasova, A. Kemper. Resource pool management: Reactive versus proactive or let′s be friends. *Computer Networks*, vol. 53, no. 17, pp. 2905–2922, 2009.

[21] N. V. Hien, F. D. Tran, J. M. Menaud. SLA-aware virtual resource management for cloud infrastructures. In *Proceedings of the 9th IEEE International Conference on Computer and Information Technology*, IEEE, Xiamen, PRC, vol. 1, pp. 357–362, 2009.

[22] H. N. Van, F. D. Tran, J. M. Menaud. Autonomic virtual resource management for service hosting platforms. In *Proceedings of ICSE Workshop on Software Engineering Challenges of Cloud Computing*, IEEE, Vancouver, Canada, vol. 1, pp. 1–8, 2009.

[23] J. Qi, X. Li, N. Hu, X. H. Zhou, Y. C. Gong, F. Wang. Algorithms of resource management for reconfigurable systems based on hardware task vertexes. *Acta Electronica Sinica*, vol. 34, no. 11, pp. 2094–2098, 2006. (in Chinese)

[24] Y. J. Joung. On quorum systems for group resources allocation. *Distributed Computing*, vol. 22, no. 3, pp. 197–214, 2010.

[25] G. Y. Wei, A. V. Vasilakos, Y. Zheng, N. X. Xiong. A game-theoretic method of fair resource allocation for cloud computing services. *The Journal of Supercomputing*, vol. 54, no. 2, pp. 252–269, 2009.

[26] K. Jansen, L. Porkolab. On preemptive resource constrained scheduling: Polynomial-time approximation schemes. *Integer Programming and Combinatorial Optimization*, vol. 2337, pp. 329–349, 2006.

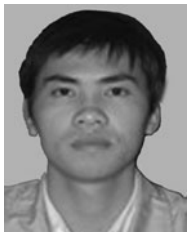
[27] Y. Hirozumi, E. F. Khaled, V. B. Gregor, H. Teruo. Protocol synthesis and re-synthesis with optimal allocation of resources based on extended Petri nets. *Distributed Computing*, vol. 16, no. 1, pp. 21–35, 2003.

[28] A. Panconesi, M. Sozio. Fast primal-dual distributed algorithms for scheduling and matching problems. *Distributed Computing*, vol. 22, no. 4, pp. 269–283, 2010.

[29] Z. Q. Sheng, C. P. Tang, C. X. Lv. Modeling of agile intelligent manufacturing-oriented production scheduling system. *International Journal of Automation and Computing*, vol. 7, no. 4, pp. 596–602, 2010.

[30] Q. Wang, W. R. Zhong, F. G. Zhong. XML-based data processing in network supported collaborative design. *International Journal of Automation and Computing*, vol. 7, no. 3, pp. 330–335, 2010.

[31] Y. Chang, S. Wilkinson, R. Potangaroa, E. Seville. Donor-driven resource procurement for post-disaster reconstruction: Constraints and actions. *Habitat International*, vol. 35, no. 2, pp. 199–205, 2011.



Xiao-Jun Chen received the bachelor degree from Department of Industrial Engineering of Xi′an University of Technology, PRC in 2004 and the master degree from School of Economics and Management of Xi′an University of Technology in 2009. He entered School of Computer Science and Engineering of Xi′an University of Technology to learn about distributed computing technology in 2008. He engaged in software development job from September, 2004 to September, 2006 in Delta. Software Company. He is now a member of IEEE Computer Society. He has published 8 papers since 2006.

His research interests include virtual technology and cloud computing.

E-mail: army.net@163.com (Corresponding author)

Jing Zhang received the bachelor degree from Department of Automatic Control of Xi′an University of Technology, PRC in 1981, the master degree from Department of Software and Theory of Xi′an Jiaotong University, PRC in 1986, and the doctor degree from Department of Systems Engineering of Xi′an Jiaotong University in 1994. He has worked in Department of Computer of Xi′an University of Technology since 1977, and now is a professor and the Ph. D. supervisor of School of Computer Science and Engineering, Xi′an University of Technology. Of which, he worked in Computer Training Center of Ministry of Education, PRC in 1982 for a short time. He has published 60 papers and hosted 20 research projects in recent 10 years, of which, completed 863 Programs. He has published 4 books, of which, representatives are *Artificial Intelligence Basis* (Electronic Industry Press, Beijing, 2000), *Practical Course on Computer Network* (Electronic Industry Press, Beijing, 2007) and *Computer Network* (Xidian University Press, Xi′an, Shaanxi, 2007). He is now one of the national key new product projects consultants and a Xi′an information technology consultant. He is also a Shaanxi manufacturing informatization expert, the member of Services Computing Professional Committee in China Computer Federation, member of E-government and Office Automation Committee in China Computer Federation, the evaluation expert of National Natural Science Foundation and the member of IBM Software Innovation Center Expert Committee.

His research interests include distributed system, virtualization, grid computing and cloud computing.

E-mail: zhangjing@xaut.edu.cn

Jun-Huai Li received the bachelor degree from Department of Computer Science and technology of Xi′an University of Technology, PRC in 1994, the master degree in Department of Computer Application of Xi′an University of Technology in 1997, and the doctoral degree from Department of Software and Theory of Northwestern Polytechnic University (NPU), PRC in 2002. He made a cooperation research in university of Tsukuba, Japan in 2004. He has worked in Department of Computer Science and Engineering of Xi′an University of Technology since 1997, and now is a professor and a master supervisor of School of Computer Science and Engineering, Xi′an University of Technology. He is also the dean of Department of Computer Science and Technology and the dean of Network and Information Management Center. He has published 40 papers and hosted 12 research projects in recent 10 years, of which, completed four 863 Programs. He has published 3 books, of which, representatives are *Practical Course on Computer Network* (Electronic Industry Press, Beijing, 2007), *Computer Network* (Xidian University Press, Xi′an, Shaanxi, 2007), and *Network Security Technology* (Xidian University Press, Xi′an, Shaanxi, 2010). He is a member of Chinese Computer Society and a member of IEEE.

His research interests include network computing, distributed computing internet technology, RFID technology, and web data mining.

E-mail: lijunhuai@xaut.edu.cn

Xiang Li received the bachelor degree from Department of Computer Application of Xi′an Jiaotong University, PRC in 2000 and the master degree from Department of Software and Theory of Xi′an Jiaotong University, PRC in 2003. She has entered into School of Computer Science and Engineering of Xi′an University of Technology to learn about distributed computing technology since 2008. After graduation from Xi′an Jiaotong university, she has been working in Shannxi University of Science and Technology, and now she is a lecturer of School of Electronic Engineering, Shannxi University of Science and Technology. She has published 10 papers since 2000.

Her research interests include distributed system and parallel computing with multi-cores.

E-mail: lixiang@163.com