A CLOUD-FEDERATION-ORIENTED MECHANISM OF COMPUTING RESOURCE MANAGEMENT

Haopeng Chen, Wenyun Dai, Wenting Wang, Xi Chen, Yisheng Wang  
School of Software, Shanghai Jiao Tong University  
chen-hp@sjtu.edu.cn, {wwtvanessa, scorpiodwy}@gmail.com, {april-622, easonyq}@hotmail.com

Abstract

As the cloud of clouds, cloud federation, sometimes called as inter-cloud, provides a feasible and attractive kind of infrastructure to the cloud applications which want to obtain high performance, availability or profit. Meanwhile, the cloud providers also may consider that cloud federation is an ideal form to manage their computing resource since it can improve the utilization of their computing resources and extend the computing power of an individual provider. This paper classifies the cloud federations into different models according to their structure and control modes, including vertical, horizontal and hybrid federation, autonomous and centralized controlled federation. A design of the framework of multi-objective constrained resource management for cloud federation is proposed based on the analysis of the common objectives of such a framework, including dynamic profit-driven provisioning, availability-aware placement and power-saved consolidation. The result of our simulation has demonstrated the feasibility and effectiveness of the proposed framework.

Keywords: cloud federation; resource management; multi-objective constrained; profit-driven provisioning; availability-aware placement; power-saved consolidation

1. INTRODUCTION

Nowadays, the cloud computing paradigm is regarded as a revolution to the conventional information technology. The features of flexible pricing, rapid provisioning and infinite scaling enable make cloud computing appealing for the applications with massive data or large-scale concurrent clients. With cloud computing, the developers can rent the software, platform and infrastructure as services to facilitate rapid application development and reduce the cost of operation and maintenance of their applications. Consequently, more and more applications have been or will be migrated and deployed into clouds.

The cloud providers also can enhance the utilization of their computing resource and obtain extra profit by leasing their idle resource as service in clouds. As a result, almost all of the IT giants build their own public clouds in forms of Software as a Service, Storage as a Service, Infrastructure as a Service and Platform as a Service. For example, Amazon EC2 (2013) and Amazon S3 (2013) respectively “provide resizable computing capability and storage space in order to make web-scale computing easier for developers”; Google (2013) states that “Google App Engine, in form of Platform as a Service, enables enterprises to build web applications on the same scalable systems that power Google applications”; Microsoft (2013) claims that “Azure enables users to quickly build, deploy and manage applications across a global network of Microsoft-managed data centers”; IBM (2013) describes that “SmartCloud is the IBM vision for cloud computing, and it is used to accelerate business transformation with capabilities from IBM cloud offerings”. Besides these enterprises mentioned above, many companies such as Salesforce, AT&T, GoGrid, NetSuite, Rackspace and RightScale also provide cloud computing service in a variety of different manners.

The diversity of public clouds provides the providers of applications with more choices to deploy their own applications. On the one hand, an application can obtain the independence of cloud providers and improve its availability by deployed into an integration of resources from multiple clouds. For example, two instances of an application can be respectively deployed into Amazon EC2 and Windows Azure to improve its global availability. On the other hand, the multi-tier architecture of web applications allows each of the tiers to be deployed into different clouds in order to ensure all the rented services are with best quality. For example, the web tier, application tier and database tier of an application can be respectively deployed into Amazon EC2, Google App Engine and Amazon S3 to ensure each tier is deployed into the cloud with best quality. Both of the cases involve ‘Cloud Federation’ of public clouds which is the cloud of multiple public clouds. As a consequence of cloud collaboration, the cloud federation of public clouds is an inevitable development of cloud computing.

With the advancement of virtualization, it is feasible for an enterprise to make use of virtualization to effectively integrate its heterogeneous computing resource into a private cloud. Thus, more and more enterprises have built or are building their own private clouds. The computing resource of a private cloud is limited, so it is necessary for a private cloud to cooperate with other private or public clouds in order to scale up its computing power when the utilization of its computing resource is saturated. Consequently, the ‘Cloud Federation’ of private clouds...
and public clouds, sometimes called as ‘hybrid cloud’, becomes very important for cloud owners and providers. Actually, for the customers, performance and availability are concerned, while power saving is an important issue for cloud providers. But existing research on cloud federation or hybrid cloud mainly focuses on how to scale up the computing power of a single cloud by building cloud federation. Meanwhile, the research on the cloud federation built for archiving high availability, independence of cloud provider and high quality of services is not adequate yet. Aware of this situation, we analyze the common objectives of resource management in cloud federation and put forward a design of multi-objective constrained framework for it in order to archive the goal of win-win for both cloud customers and providers.

The remainder of the paper is structured as follows. Section 2 classifies the models of cloud federations according to their structure and control modes; Section 3 analyzes the objectives of resource management in cloud federation; Section 4 describes our proposed design of the framework of multi-objective constrained resource management in details; Section 5 discusses the mechanism of multi-objective constrained resource management; Section 6 describes how we perform the simulation and analyzes its result; Section 7 briefly summaries the related works; and conclusion is in Section 8.

Figure 1. Structures of cloud federations: (a) architecture of typical multi-tier applications, (b) a vertical cloud federation, (c) a horizontal cloud federation, (d) a hybrid cloud federation

2. Models of Cloud Federations
According to the structure of various cloud federations, we can classify them into three types: vertical, horizontal and hybrid cloud federation. If we classify the cloud federations by the control modes,
they can be divided into two types: autonomous federation and centralized controlled federation.

2.1 STRUCTURES OF CLOUD FEDERATIONS

Most cloud applications inherently have the multi-tier architecture which at least includes web tier, application tier and database tier. Each of the three tiers can be individually deployed in order to meet the customized demand of applications. Each tier can be refined into more fine-grained tiers according to the requirement of applications. For example, the web tier can be divided into presentation tier and controller tier; the application tier can be divided into service tier, domain tier and data access tier. However, the fine-grained tiers of a coarse-grained tier are not deployed separately otherwise the performance will be damaged drastically.

Figure 1(a) is an example of multi-tier application in which the clusters of web tier, application tier and database tier are respectively composed of four, four and three VMs (Virtual Machines). There is a load balancer in front of each cluster to dispatch the requests and balance the workload among the VMs in the same cluster. The multi-tier architecture and the flexible deployment mode of cloud applications enrich the diversity of cloud federations by enabling the vertical cooperation between clouds.

The first type of cloud federation is “Vertical Cloud Federation”, shown as Figure 1(b), in which the clouds vertically collaborate with each other to provide the application with all the necessary services. The “vertical collaboration” means that each cloud provides hosting environment for only one tier and any request dispatch involving multiple tiers needs to be accomplished by the vertical collaboration between clouds. For example, in Figure 1(b), the web tier, application tier and database tier are respectively deployed into cloud A, B and C. The resource from cloud A, B and C allocated to the target application forms a vertical cloud federation. Such a deployment solution is totally determined by quality of services.

The second type of cloud federation is “Horizontal Cloud Federation”, shown as Figure 1(c), by which a cloud application can obtain the independence of cloud providers and improve its availability by deployed multiple instances into an integration of resources from multiple clouds. As the saying — ‘Don’t put all your eggs into one basket’, the instances of an application can be horizontally deployed into different clouds to reduce the failure probability. For example, in Figure 1(c), each of cloud A, B and C has a complete instance of the application. Such a deployment solution is also effective to solve the problem of saturation of computing resource.

The third type of cloud federation is “Hybrid Cloud Federation”, shown as Figure 1(d), which is a combination of vertical and horizontal cloud federations. For example, in Figure 1(d), the collaborations between cloud B and cloud D, cloud C and cloud D, and cloud D and cloud E are vertical while the ones between cloud A and cloud B, cloud B and cloud C, and cloud A and cloud D are horizontal. It is obvious that this type of federations is not so common than the other two types due to its high complexity. However, it is possible that some applications still prefer this type due to their special requirements.

2.2 THE ARCHITECTURE OF CLOUD FEDERATION

Cloud federation is a cloud of clouds, which means the resource of cloud federation is from multiple cloud providers. Since all the resource of cloud providers is encapsulated as services, there should exist a service registry to facilitate the cloud providers to publish their services. Consequently, the service registry is a necessary component of cloud federation. However, the service registry is not enough to provide all the necessary support for constructing the cloud federation. So we consider the architecture of cloud federation should be the one shown in Figure 2, in which the core is the Cloud Federation Center which contains the lightweight kernel, the infrastructure and the extensions. The Cloud Federation Center acts as an agent for cloud customers and cloud providers to facilitate the construction and deconstruction of cloud federation coordinate the services provided by clouds and distribute revenue among clouds.

For cloud providers, the CF(Cloud Federation) Manager is used to communicate with other clouds, which is a new component for most existing clouds. The App Manager and the Cloud Manager are the extension of existing components of clouds, which means that new features should be added into them to support cloud federation. All the cloud applications are deployed into the Virtualized Resources.

Each cloud involved in cloud federation can communicate with Cloud Federation Center and other clouds to provide necessary services to cloud applications and coordinate the cooperation among services.

The details of these components will be given in section 4.
2.3 Control Modes of Cloud Federation

The control modes of cloud federation can be classified into the Autonomous Model and Centralized Controlled Model according to the way of management of cloud federation.

Autonomous Cloud Federation is autonomously managed by cloud providers. The initiators of Autonomous Cloud Federation are cloud providers, especially private cloud providers. They just query available resources through the lightweight core of the Cloud Federation Center and autonomously complete the process of building cloud federation which is transparent to the cloud consumers. This model is used to build horizontal cloud federation in order to extend the computing power of the initiators’ cloud.

Centralized Controlled Cloud Federation is managed by the Cloud Federation Center. The initiators of Centralized Controlled Cloud Federation are usually the cloud consumers. Their request of building cloud federation can be divided into two types. In the first case, the cloud consumers retrieve the information of available resources, lease the resources from cloud providers and deploy their applications. But obviously, it requires the cloud consumers must be very professional. Thus, in the second case, more common than the first one, cloud consumers send their functional requirements of rental resources and SLA (Service-Level Agreement) constraints to the cloud federation center, and then the cloud federation center generate the solution of building cloud federation in form of service composition based on the registered service. This mode is used to build both horizontal and vertical cloud federation.

3. Objectives of Resource Management

No matter in which cloud federation, the common objectives of resource management are to achieve high performance, high availability and power saving.

3.1 Common Objectives of Resource Management

Either the consumers or the providers of cloud federation have their own objectives of resource management.

For the consumers, namely the providers of cloud applications, it is their primary goal to reduce the leasing cost by optimizing the utilization of leased resource under the condition of guarantee reasonable quality of services they provided. The balance between quality of services and cost will be taken into account in order to maximize the profit of application providers. So the profit-driven resource provisioning is the first objective of resource management.

Availability is also a concerned issue for application providers. Since it is the relative but not absolute locations of the VMs that determine the availability of given resource, the availability-aware resource placement will give the feasible topology of VMs which indicates the network distances among VMs but not determines their physical locations.

The cloud providers will map the relative locations of VMs onto physical nodes according to the runtime status of the latters. The cloud providers need to ensure that the utilization of their resource is limited into a reasonable range in which the power consumption is optimized. Furthermore, the cloud providers need periodically consolidate their resource by VM migration due to the fluctuation of utilization of resource. So the power-saved resource consolidation is another important objective of resource management.

In summary, the common objectives of resource management are profit-driven resource provisioning, availability-aware resource placement, and power-saved resource consolidation. Consequently, the resource management in cloud federation is a multi-objective constrained one.

3.2 Profit-driven Resource Provisioning
From the viewpoint of consumers of cloud federation, the computing power of their cloud federation instances had better be able to be dynamically scaled up and down with the change of real-time workload of their cloud applications, since their profit will be maximized by renting the computing power in an economical way.

The profit of a cloud application is from the difference between its revenue and cost. The revenue is determined by the SLA assigned by the application and its consumers. In general, it is in proportion to the performance level achieved. Monotonic non-increasing utility functions are quite realistic to model the relationship between the revenue and the achieved performance, since the better the achieved performance is, the higher the revenues gained per request are. The cost is determined by the amount and price of computing power in the hosting cloud federation. The more the rented computing power is, the more the cost is. Given the unit price of computing power is fixed, there is a linear dependency between the cost and the amount of rented computing power. It is common that more computing power will result in better performance. But it is possible that the cost increased by the more rented computing power is greater than the revenue increased by improved performance. Thus, the better performance would probably result in less profit. So profit-driven resource management needs to find the point in Figure 3 at which the difference between Revenue(R) and Cost(C) is the maximum.

In Figure 3, the horizontal coordinate presents the number of VMs the cloud application rented, while the vertical coordinate presents the sum of profit. The Cost(C) linearly depends on the number of VMs (Virtual Machines). The Revenue(R) also increases with the increase of the number of VMs, but the relationship is not linear. Given the number of VMs is specified, for example B, we can find point A at which MR equals to MC and the profit is maximal. Given the number of VMs is changed, for example, it is changed to B' or B'', the point A will be changed to A' or A'' correspondingly. The Revenue(R) does not only depend on the number of VMs, it also depends on the real-time workload of the cloud application, since the achieved performance is determined by these two factors. The global maximal profit is the maximum of all of the maximal profit under each number of VMs.

Besides of performance, the availability is also a concerned objective for the owner of a cloud application. The profit-driven resource management only focuses on the appropriate amount of resource to be rented, while the availability-aware resource placement aims to determine the places of the rented resource. When some computing resource is rented by a provider to build the infrastructure of hosting environment of its application, the availability of such an infrastructure can be calculated with the availability and the topological structure of the physical hosts.

For example, Figure 4 shows two infrastructures of an application both of which are composed of two physical hosts. Both of the physical hosts of the blue one are in the cloud A while the two physical hosts of the green one are respectively in cloud A and cloud B. It is obvious that the availability of the green one is higher than that of the blue one, since the blue one is not available when both the clusters of Cloud A the hosts belong to are simultaneously not available while the green one is not available when both the cluster of Cloud A and the cluster of B the hosts belong to are simultaneously not available. In many clouds, for example, in Amazon EC2, the hosts are clustered into availability zones, and the zones are grouped into availability regions. In such a structure, the further the hosts of a cluster are away from each other, the more available the cluster is. But we should be aware that the high availability is obtained at the cost of performance.

3.3 Availability-aware Resource Placement
As a result, the geographic distribution of the hosts of a cluster should be limited into an acceptable range.

Figure 4. Two Deployment Solutions

3.4 POWER-SAVED RESOURCE CONSOLIDATION

From the view of cloud provider, the aim of resource consolidation is to save power and then cut down the operating cost. On the one hand, the cloud providers want to satisfy the resource requirements of cloud applications with minimal number of running physical hosts. On the other hand, they also hope that all the running physical hosts are running at appropriate status which means the utilization of computing resource on each physical host is greater than the lower bound and smaller than the upper bound.

The input of power-saved resource consolidation is the output of availability-aware resource placement in which the places of VMs are logical ones. The physical places of VMs will be located according to the runtime status of physical hosts. Furthermore, they are not fixed since the utilization of computing resource of physical hosts varies with the runtime workload of cloud applications. As a result, the periodical check for overloaded and underloaded nodes is executed and then the dynamical balancing is accomplished by VM migration.

In a cloud federation, it is possible for cloud providers to lease computing resource from each other which will result in a leasing loop. In such situation, the performance of cloud applications will be impacted since it incurs unnecessary remote communication cost. As a result, the resource consolidation needs to eliminate the leasing loop by VM migration too.

4. A DESIGN OF THE FRAMEWORK OF CLOUD-FEDERATION-ORIENTED RESOURCE MANAGEMENT

This section gives a design of the framework of cloud-federation-oriented resource management, as shown in Figure 5.

Figure 5. Enabling Components in Cloud Federation
4.1 CLOUD FEDERATION CENTER

The Cloud Federation Center is the core of the framework we proposed. As shown in Figure 5, the Cloud Federation Center is composed of three parts.

The Lightweight Kernel is a service registry in which various service descriptions are published by the cloud providers. The cloud customers and providers query the kernel to discover desired services. We have designed and implemented a service registry which can discover the alternative services that meet the demand according to the specified functional and QoS requirements (Xiong & Chen, 2009). Therefore, the lightweight kernel can be implemented by reusing and extending the existing service registry to support the cloud-specific semantic descriptions. This kernel is necessary for either the Autonomous or the Centralized Controlled cloud federations.

The Extensions is used in the Centralized Controlled Cloud Federations but not in the Autonomous ones. The CF(Cloud Federation) Generator generates a solution for deploying Cloud applications according to the SLAs. The generated solution can be a single cloud, or a vertical, a horizontal or a combined cloud federation. We have designed mechanisms for determining the required quantity of computing resource based on the predicted performance (Chen, et al., 2011) and allocating the computing resource dynamically based on the required availability (Wang, et al., 2012). The Provider Manager is used by the administrator of the Cloud Federation Center to ensure that only qualified providers can register their services into the Lightweight Kernel. The Service Composition component is called by CF Generator to obtain a composite service from multiple clouds when the latter fails to find a single cloud as the hosting environment.

The Infrastructure is also used in the Centralized Controlled Cloud Federation but not in the Autonomous ones. The resource involved in cloud federation is monitored by the Resource Monitor at runtime to get their real-time status. The Revenue Distributor can reasonably distribute the revenue of cloud federation to all the involved clouds. The Service Coordinator can do the API and protocol transformation in order to coordinate the service cooperation across clouds.

4.2 DYNAMIC RESOURCE MANAGEMENT

The dynamic resource management in our framework is realized by the collaboration of App Manager, Cloud Manager and Resource Monitor in the Cloud Federation Center. There are three parts of dynamic resource management shown in Figure 5.

The Resource Monitoring Component is located in the Infrastructure part of the Cloud Federation Center. As we mentioned, in this component, the statistics and analysis of real-time monitoring data collected from the clouds in the cloud federation assure the cloud federation center grasping the global real-time state of computing resources.

The App Manager is one module of cloud providers. The Real-time Monitor component monitors the runtime status of Cloud applications deployed in the cloud, including the response time, throughput, failures and so on. It can be implemented in the manners of packet filtering or proxy. The Performance Analyzer will model and predict the performance of cloud applications based on the monitored data. The data obtained from the analysis can be used by the Resource Scheduler to allocate or reclaim computing resource for the Cloud applications. The Resource Arranger periodically rearranges the allocated computing resource by live migration of VMs in order to minimize the resource fragmentation generated at runtime.

The Cloud Manager is an existing module of cloud providers. We need to add some new functions to its existing components. The Resource Manager determines how and when to construct and deconstruct the Autonomous Cloud Federation based on the global utilization of its computing resource. The Payment Manager discriminates the revenue from cloud federation from that completely from its own cloud since the former needs to be distributed among the clouds involved into the cloud federation. The Leasing Manager doesn’t only manage the leasing contracts signed with the Cloud applications, but also manages the ones signed with other clouds in cloud federation. Meanwhile, the User Manager manages all the registered users and trusted cooperative cloud providers.

4.3 SERVICE COOPERATION

As shown in Figure 5, the service cooperation is implemented through the Service Coordinators in Cloud Federation Center and clouds. The Service Coordinator in cloud is a part of CF Manager, and comprises of the following modules.

The Comm(Communication) API should be consistent with the Service Coordinator in the Cloud Federation Center. Meanwhile, various cloud providers should provide adapters for the Comm API, and map it to their proprietary implementation.

The Security Controller realizes the strict control access and encryption of sensitive data which are necessary for all cloud providers.

The Logger realizes the log management. The configured log system is designed to assure the effective log management.

The Semantic Matcher provides a mechanism of semantics extending in which the ontologies and other formal methods are utilized to describe the semantics of collaborative behavior in cloud federation.
4.4 Revenue Distribution

As shown in Figure 5, the revenue distribution of the framework we proposed is implemented through the Revenue Distributor in Cloud Federation Center and the Profit Engines in clouds. The Profit Engine in cloud is a part of CF Manager, and comprises of the following modules.

The Pricing Manager is supported by the dynamic pricing mechanism of existing price management module of cloud providers. Meanwhile, the scheme of constructing cloud federation generated by the cloud federation center is used as an additional factor to determine the dynamic prices of computing resource.

The Revenue Distributor should be consistent with the Revenue Distributor in the Cloud Federation Center. The multi-objective optimization algorithm is utilized to design and implement the strategy of revenue distribution for multi-win.

The Budget Manager determines the leasing policies according to the SLAs of the Cloud application, including the quantity, location and VM types.

The Billing Manager is an existing component of cloud providers to compute the charge of cloud consumers which are either the Cloud applications or the other clouds.

5. The Mechanism of Multi-objective Constrained Resource Management

In this section, we will discuss the mechanism of multi-objective constrained resource management in details, including the management flow, the ways to implement the objectives.

5.1 The Flow of Resource Management

As discussed in section 3, the resource management of cloud federation is a multi-objective constrained one of which the objectives are profit-driven provisioning, availability-aware placement and power-saved consolidation.

In general, there are three ways to implement the multi-objective resource management: parallel process, serial process with feedback and serial process without feedback.

In the parallel process, all the objectives are independently processed to obtain the solution sets for each objectives and then the intersection of these solution sets is calculated. If the intersection has several solutions, one of them will be chosen randomly or by some other rule as the final result. If the intersection is an empty set, the objectives will compromise with each other and the intersection will be iteratively calculated until get a nonempty result. This way is apt to get a global optimal solution since all the objectives are processed independently. But it is obvious that this way is time-consuming one which makes it hard to be applied in practice.

In the serial process with feedback, all the objectives are processed one by one in some order and the output of the process of one objective will be the input of the process of next objective and is fed back to the process of previous objective. For example, the output of profit-driven provisioning is the amount of desired computing resource, which is the input of availability-aware placement. The latter will generate a topological structure of the desired computing resource which possibly violates the SLA on performance due to the overlong distance between VMs. So the topological structure needs to be fed back to the profit-driven provisioning in order to verify its feasibility. If the verification is passed, the topological structure will be sent to the power-saved consolidation to locate the VMs. If the verification is failed, the availability-aware placement will repeatedly generate a new output and feed it back to profit-driven provisioning until one output passes the verification. This way also can get a global optimal solution, however, it is still too time-consuming to be accepted in practice.

Serial process without feedback is designed to reduce the time consumption by removing feedback and adding constraints onto output. For example, the output of profit-driven provisioning is the amount of desired computing resource and the maximal acceptable distance between VMs. The availability-aware placement will generate a topological structure of VMs under such constraint which makes the feedback unnecessary since the performance is guaranteed by the constraint. This way cannot generate a global optimal solution and just can generate an acceptable one which is possible far away from the former. However, the time consumption of this way is dramatically reduced due to the remove of feedback. Consequently, we adopt this way in our proposed mechanism.

In the serial process without feedback, the order of processing of objective embodies the importance of the objectives. The profit-driven provisioning determines the amount of desired computing resource which is the base for other two objectives. As a result, it is the first processed objective. The availability-aware placement determines the topological structure of the desired computing resource which is independent of specific physical nodes. So it is the second processed objective. The power-saved consolidation maps the topological structure onto the physical nodes. It certainly should be the last processed objective.

Summarily, the flow of multi-objective resource management is shown in Figure 6.
In Figure 6, the profit-driven provisioning module reads the monitored runtime status of cloud applications and physical nodes, SLA and resource descriptors as inputs, and calculates the amount of desired computing resource as output for scaling up or down. The availability-aware module receives the output of previous module and reads the static configuration about availability in resource descriptor and requirement in SLA to generate the topological structure of resource. This module needn’t read the runtime status because its output is independent of physical nodes. The power-saved consolidation module reads the monitored data and maps the topological structure onto physical nodes. Furthermore, this module periodically reads its output as input in order to consolidate the resource in background. Meanwhile, the whole flow will also be executed periodically to implement dynamic resource management.

5.2 PROFIT-DRIVEN RESOURCE Provisioning

Since the runtime workload is varying from time to time, the profit-driven resource provisioning needs to dynamically find the global maximal profit point in order to determine the amount of desired computing resource.

We have proposed a performance model for analyzing and forecasting the runtime workload of cloud applications and a method for determining the amount of computing resource need to be added or removed (Chen, et al., 2011). This model and method can be applied into the scenario of cloud federation.

For a cloud application deployed into a cloud federation, its operation cost at time $t$ is:

$$\text{Cost}_t = \sum_{i=1}^{k} (\text{Price}_{ri} \cdot N_{ri} + \min(\text{Price}_{si}, \text{Price}_{ci}) \cdot N_{di})$$

(1)

where $\text{Cost}_t$ is leasing cost, $\text{Price}_{ri}$, $\text{Price}_{si}$ and $\text{Price}_{ci}$ denote the price of long-term lease, auction lease and on-demand lease of the computing resource of cloud $i$ respectively, $N_{ri}$ and $N_{di}$ represent the amount of resource of cloud $i$ for long term and short-term tenancy. This equation means that the cloud application will require a certain amount of long-term leased computing resource to process its regular workload and require the on-demand or on-spot resource to deal with the boom growth of workload. It is obvious that $N_{ri}$ and $N_{di}$ is determined by the runtime workload since the application needs appropriate amount of resource to guarantee the quality of service in SLA with the fluctuation of workload.

The revenue of a cloud application is:

$$\text{Inc}_t = \sum_{i \in S} \sum_{j=1}^{x_{it}} (c_i \cdot P_{ij} - \eta \cdot (1 - P_{ij}))$$

(2)

where $\text{Inc}_t$ is the revenue, $i$ denotes the $i^{th}$ service the cloud application provided, $x_{it}$ denotes the number of accepted requests to the $i^{th}$ service while $j$ denotes the $j^{th}$ one of the $x_{it}$ requests. $P_{ij}$ is the probability that the measured quality of the $i^{th}$ service for the $j^{th}$ request is superior to the desired one. If a request is processed by the $i^{th}$ service without any violation of SLA, the application provider will obtain the incoming of $c_i$, otherwise, it will suffer the penalty of $r_i$. $P_{ij}$ is dependent on the quality of services, because the heavier the workload is, the poorer the quality of service is, and then the more possible that the provider will suffer penalty.

The difference between the $\text{Cost}_t$ and $\text{Inc}_t$ is the profit of application provider. Both of them are dependent variables of quality of service and runtime.

http://hipore.com/ijcc/
workload. We can find the global maximal profit point by analyzing the runtime status, including average response time and workload. Then, we can dynamically calculate the most appropriate amount of desired computing resource.

The cloud federation center will determine the construction and separation of cloud federation instances according to the provisioning result. The resource provisioning just determines the amount of computing resource needed but not specifies the source of the computing resource. The cloud federation center will determine how to locate the computing power according to the requirements of consumers. For instance, if the consumer wants to maintain the status that multiple instances of the cloud application should be deployed into multiple cloud providers, when the computing resource needs to be scaled down, the cloud federation center will just scale down the computing resource of some instances but not remove any instance. Either the profit of cloud applications or the one of cloud providers will be guaranteed as high as possible in such a dynamic scaling mechanism.

5.3 Availability-aware Resource Placement

We have proposed an availability-aware approach to place VMs for dynamic scaling of cloud applications (Wang, et al., 2012). In this approach, we used Bayesian formula to evaluate the availability of the infrastructure of a cloud application. For instance, the green solution in Figure 4 will be unavailable under the following situations: both the hosts are failure, one host is failure and the other is normal while the master host of its cluster or cloud is failure, both the hosts are normal while both the master hosts of their clusters or clouds are failure. With Bayesian formula, we can calculate the conditional probability that both the hosts of green solution are unavailable and then derive its availability.

In this approach, the maximal acceptable distance between any two VMs was also taken into account to prevent the violation of SLA on performance. However, we just put a single upper limitation on this distance which means all the VMs are equivalent to each other. In fact, if a multiple layered application is deployed as Figure 1 (b), (c) or (d), such a single upper limitation is not suitable any more. For example, the VMs in web layer can be distributed far away from each other since they needn’t to communicate with each other, while a VM of web layer should be close to a VM of application layer in order to reduce the communication cost between layers. So when this approach is applied into cloud federation, a set of the maximal acceptable distances but not a single distance is specified according the relationship between VMs, such as the distance between two VMs in same layer or the one between two VMs in two adjacent layers.

When applying this approach, we will define the availability for each cloud separately because the cloud federation is constructed with the resource from multiple clouds which are highly likely heterogeneous with each other. Thus, the physical nodes of each cloud, including physical machines, network switches and other network equipment, will be assigned with their own availability. The calculation of conditional probability in cloud federation is more complicate than that in a single cloud, so it is a bit time-consuming.

During the availability-aware resource placement, a VM placement plan is generated in which the places of VMs are logic ones but not physical ones. On the one hand, from the view of cloud application, the logic places, such as the relative distances between VMs are more important than the physical places because the physical hosts in a cluster are equivalent to each other. For example, for the blue solution in Figure 4, the cloud application concerns that its two VMs must be deployed into two clusters of a cloud while doesn’t care the VMs are deployed into which physical hosts of the two clusters. On the other hand, the physical places should be determined by cloud provider according to the runtime load of physical hosts.

5.4 Power-saved Resource Consolidation

We proposed an approach to dynamic workload balancing, which periodically checks the overloaded and underloaded nodes and then dynamically balances the workload by VM migration (Zhang, et al, 2011).

In this approach, we define the upper bounds and lower bounds of utilization of computing resource for VM and physical nodes. If the runtime utilization of a physical node is beyond the upper bound, this node will be marked as an overloaded node and its some VM(s) will be migrated to a paired physical node in order to ensure all the physical nodes are operating in normal mode. If the runtime utilization of a physical node is below the lower bound, this node will be marked as an underloaded node and it will be merged with other underloaded nodes or paired with an overloaded node. The virtual nodes are consolidated in the similar way. Taking the performance into account, we execute such consolidation locally but not globally. For example, a VM can only be migrated to the physical nodes in the same region of its host node.

6. Simulation and Analysis

We performed some simulation to show the effect of multi-objective constrained resource management in cloud federation. This section will give the details about the simulation.

6.1 Toolkit and Dataset of Simulation

http://hipore.com/ijcc/
We extended CloudSim (Calheiros, et al., 2011) with some modules designed by ourselves to support our research. The CloudSim toolkit we used was CloudSim 3.0, which was run at a Mac book with two 2.8GHz CPU and 4GB RAM running OS X 10.8. We used CloudSim to simulate two clouds each of which had tens of VMs.

The dataset of our simulation was NASA Ames iPSC/860 log (Feitelson, 1995). We parsed it and counted the numbers of jobs of each hour in order to simulate the workload. Actually, the iPSC/860 machine located at NASA Ames was a 128-node hypercube. At the time, it was the workhorse of the NAS facility for scientific computations. Up to nine jobs could run on the system at the same time, by using distinct subcubes.

Firstly, we used the autoregressive model to predict the runtime workload at next time node and calculated the amount of VMs needed to meet the performance requirement in SLA. Secondly, we calculated the availability to determine the topological structure of the desire amount of VMs in a cloud federation and then we placed the VMs in such a topological structure. Finally, we periodically consolidated the computing resource by VM migration.

6.2 WORKLOAD FORMATTING AND FORECASTING

The workload file of NASA Ames iPSC/860 log was suffixed with “swf” (Standard Workload Format). We preprocessed the workload and reserved some useful data fields. We calculated the amounts of jobs per hour, shown in Figure 7 (a), as the initial source input of our simulation.

As mentioned in Section 5.2, the real meaningful input is the forecasting workload of next time node but not the workload monitored at present time. So we used autoregressive model, based on the workload of eight previous time nodes, to forecast the dynamic workload of next time node. The result is shown in Figure 7(b).

In the corresponding of the real and forecasting workload, the deviation is about 22%, mostly happened in the peak nodes and some others in the nadir nodes. In most time nodes, the forecasting workloads are acceptable. If more precise forecasting workloads are needed, other complex models can be applied.

Figure 7. (a) Hours of 2-month real workload; (b)The real (Blue line) and forecasting (Black line) workload in one month. The forecasting workload is calculated using AR model.

6.3 RESOURCE PROVISIONING

Based on the forecasting workload, the number of VMs, that is the amount of computing resource, is calculated according to the specific requirements in SLA. In CloudSim, we created two datacenters to simulate two clouds, each datacenter had two hosts. The first host had four CPUs at 10000 MIPS CPU and the second one had two same CPUs, while they both had 2G of memory. The VMs in each host are no more than twenty, and the number of VMs can be changed with the dynamic workload and requirement. Each VM has one CPU at 2000 MIPS and 512 MB of memory.

We chose the first one-hundred hours workload and set the upper bound of response time required of each job was not longer than 200ms. If the response time is beyond the upper bound, scaling up will be executed. Otherwise, we will check whether the computing resource is overprovisioned in order to make a decision to scale down. Figure 8 shows the correspondence of jobs and number of VMs, and we can see the effect of scaling up and down since the number of VMs is changed with the number of jobs.

http://hipore.com/ijcc/
6.4 Resource Placement

We had two clouds simulated in CloudSim with their own average availabilities of regions, zones and hosts, shown as Table 1.

Suppose the communication costs between two VMs on a single host, on different hosts in a single zone, in different zones of a single region, in different regions of a single cloud, and in different clouds are respectively 0, 1, 2, 3 and 6. To satisfy the availability requirement of that being greater than 99.99%, we used availability-aware policy with relocation of VMs (Wang, et al., 2012), and the Figure 9 shows the average availabilities of the system under the two-day workload, which corresponds to the first 48 hours of Figure 8. The reason why we chose the first two-day workload but not the whole five-day workload is just for simplification since the figure will be unclear if we rendered the latter.

<table>
<thead>
<tr>
<th>Cloud</th>
<th>Region</th>
<th>Zone</th>
<th>Host</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>99.99%</td>
<td>99.98%</td>
<td>99.8%</td>
</tr>
<tr>
<td>B</td>
<td>99.98%</td>
<td>99.97%</td>
<td>99.7%</td>
</tr>
</tbody>
</table>

Table 1. Availabilities of clouds

Figure 9. The average availabilities of the system under the 48 hours workload. The blue line shows the VM amount needed per hour. The green line shows the average availabilities with neither placement policy nor VM relocation. The red line with squares shows the average availabilities with availability-aware policy and VM relocation.

From the result shown in Figure 9, the availability-aware resource placement is useful and meaningful for the aim of high-availability since it can guarantee the availability is always higher than the required one.

6.5 Resource Consolidation

We consolidated the deployment of applications in the CloudSim. Our aim is improve the utilization of the whole physical resources. To prove our strategy is useful, we expanded our simulation environment from 4 hosts to 40 hosts, and each host still had 20 virtual machines. Every 20 hosts were in the same zone where the old 2 hosts in. We set the lower bound of utilization is 20% and the upper bound is 70%. We deployed the...
application into 40 hosts with 800 virtual machines initially, and then we did the consolidation in every 300 seconds.

Figure 10. The result of consolidation. The red line shows the number of physical hosts. The black line shows the number of VM migration.

In the first time of deployment without consolidation, almost all the 800 virtual machines were used, and the 40 hosts were of low utilization which is from 2% to 16%. Under our consolidation strategy, the virtual machines in the minimum-utilization host were migrated to another host in the same zone. After doing consolidation for one day (about 245 times), as shown in Figure 10, the amount of physical hosts reduced from 40 to 3. The final three hosts were of the utilization of 57.07%, 55.63% and 62.72% respectively.

7. RELATED WORKS

The concept of cloud federation was first mentioned as Intercloud by Kevin Kelly in 2007, and he said “eventually we’ll have the Intercloud, the cloud of clouds.”(Kelly, 2007). Sam Johnston further expatiated that “the Intercloud is a global cloud of clouds as the Internet is a global network of networks”(Johnston 2009). However, the concept of Intercloud didn’t receive enough concerns, because there was little consensus on how to define the Cloud and many people considered that cloud computing was just a redefinition of the commercial by existing technology.

With the continuous development of cloud computing, more and more people have profoundly understood the essence of cloud computing and realized the importance of cloud federation. During 2009, some researchers used cloud federation to describe the future data center. One of the most important papers was the “Blueprint for the Intercloud” (Bernstein, D., et al., 2009). This blueprint concerned protocols and formats for cloud computing interoperability but didn’t put forth the scheme of many other problems such as when and how to build intercloud and how to distribute profits among all the providers.

In 2009, Global Inter-Cloud Technology Forum (GICTF, 2009) was established in Japan and attempted to promote development of intercloud. In 2010, this forum published a White Book about use cases and functional requirements for intercloud computing (GICTF, 2010).

Research on the architecture of cloud federation is the most fundamental among all researches about cloud federation. One of the two main architectures is using an independent third-party heavyweight cloud federation center as the core which takes charge of the dynamic combination and resolution of cloud federation, such as an architecture proposed by Buyya, et al. in 2010. This architecture is convenient to use and don’t have drastic changes to the existing cloud architectures, but cloud consumers must change the mode of using cloud resources and the center is the single point of failure of this architecture.

Considering of the autonomy of cloud federation, more researcher prefer the other architecture — the lightweight cloud federation center. In this architecture, the center takes charge of the registration and query of resource information from every cloud; meanwhile, all the clouds combine into and split from the cloud federation dynamically autonomously. Two typical representatives of this architecture are a cloud federation mode proposed by Antonio Celesti, et al. (Celesti, et al., 2010) and RESERVIOR mode proposed by IBM (Rochwerger, et al., 2009) In this architecture, the pressure of cloud federation center is largely reduced. This architecture is suitable for active collaboration between cloud providers, but not so helpful for Cloud applications of which all the tiers are not deployed into a single cloud. The two modes mentioned above both are based on performance, not considering other factors, such as availability and power saving, so they can’t fully meet the actual multi-objective demand.

Rodrigo N. Calheiros et al designed Aneka, a platform for developing scalable applications on the Cloud, supports such a vision by provisioning resources

http://hipore.com/ijcc/
from different sources and supporting different application models (Calheiros, et al., 2012). They mentioned that the key concepts and features of Aneka support the integration between Desktop Grids and Clouds. Like almost all existing research on cloud federation, Aneka aims at how to scale up the computing power by integrating the computing resource from multiple providers. They ignored the cloud federation which is built for improving the availability and obtaining independence of providers and best quality of services.

The key of cloud federation is that the clouds can communicate with each other by a unified API and specific adaptors. Apache Deltacloud is right such a project that “gives customers an opportunity to manage cloud instances in the way they want” (Apache, 2013). This project facilitates the cons-truction of cloud federation and makes it feasible. But it doesn’t provide customers any functions to automatically request computing resource according to their constraints.

In conclusion, the most existing research on cloud federation focuses on how to scale up the computing power by cloud federation but not how to improve the quality of services by cloud federation. Both of the two aspects are important for cloud applications. So this paper tried to give a more comprehensive analysis and design of resource management of cloud federation.

8. Conclusions

This paper classifies the cloud federations into different models according to their structure and control modes, including vertical, horizontal and hybrid federation, autonomous and centralized controlled federation. A design of the framework of multi-objective constrained resource management for cloud federation is proposed based on the analysis of the common objectives of such a framework, including dynamic profit-driven provisioning, availability-aware placement and power-saved consolidation. The result of our simulation has demonstrated the feasibility and effectiveness of the proposed framework.

With an implementation of the framework proposed in this paper, the independent cloud federation center would be able to schedule computing power for the providers of cloud applications in a transparent way, which would greatly lower the technical threshold of application of clouds.

9. References


Applications, The Proceedings of 2012 IEEE Conf. on Autonomic and Trusted Computing, 509-516


Authors

Haopeng Chen, Associate Professor of School of Software at Shanghai Jiao Tong University. He received his Ph.D degree from Department of Computer Science and Engineering, Northwestern Polytechnical University in 2001. His research group focuses on Distributed Computing and Software Engineering.

Wenyun Dai, born in Yangzhou, Jiangsu, P.R. China. He graduated from Xiamen University major in Software Engineering in 2010. He is now a third-year master student in School of Software, Shanghai Jiao Tong University. He is interest in resource management in clouds and multi-objective optimization.