Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 823609, 15 pages
http://dx.doi.org/10.1155/2015/823609

Research Article

Multicriteria Resource Brokering in Cloud Computing for Streaming Service

Chih-Lun Chou,[1] Gwo-Jiun Horng,[2] Chieh-Ling Huang,[3] and Wei-Chun Hsieh[4]

[1]*Department of Information Telecommunications Engineering, Ming Chuan University, Taoyuan 33348, Taiwan*
[2]*Department of Computer Science and Information Engineering, Southern Taiwan University of Science and Technology, Tainan 71005, Taiwan*
[3]*Department of Computer Science and Information Engineering, Chang Jung Christian University, Tainan 71101, Taiwan*
[4]*Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan 70101, Taiwan*

Correspondence should be addressed to Chieh-Ling Huang; kaio.hcl@gmail.com

Received 27 October 2014; Accepted 26 March 2015

Academic Editor: Jian Guo Zhou

By leveraging cloud computing such as Infrastructure as a Service (IaaS), the outsourcing of computing resources used to support operations, including servers, storage, and networking components, is quite beneficial for various providers of Internet application. With this increasing trend, resource allocation that both assures QoS via Service Level Agreement (SLA) and avoids overprovisioning in order to reduce cost becomes a crucial priority and challenge in the design and operation of complex service-based platforms such as streaming service. On the other hand, providers of IaaS also concern their profit performance and energy consumption while offering these virtualized resources. In this paper, considering both service-oriented and infrastructure-oriented criteria, we regard this resource allocation problem as Multicriteria Decision Making problem and propose an effective trade-off approach based on goal programming model. To validate its effectiveness, a cloud architecture for streaming application is addressed and extensive analysis is performed for related criteria. The results of numerical simulations show that the proposed approach strikes a balance between these conflicting criteria commendably and achieves high cost efficiency.

1. Introduction

As an emerging technology, cloud computing combines various computing paradigms and provides main service models which are known as SaaS (Software as a Service), PaaS (Platform as a Service), and IaaS (Infrastructure as a Service). Among these three service models, IaaS arouses the outsourcing trend of IT infrastructure by provisioning fundamental computing resources where the consumers are able to deploy and run the desired applications as specific services. Most of the IaaS consumers are providers of Internet application hosting complex service-based platforms. Those service platforms require efficient resource allocation to ensure their particular QoS (Quality of Service) which is usually specified in SLA (Service Level Agreement) and meanwhile prevents overprovisioning in order to reduce operating costs. On the other hand, providers of IaaS also concern their profit performance and energy consumption while providing these virtualized resources. Therefore in this paper, a cloud architecture for streaming service platform is addressed and an efficient resource brokering approach based on the analysis of both objectives is proposed.

Inheriting the essence of distributed and parallel characteristics in grid computing and with the growth of virtualization as well as developed web services technologies, cloud computing becomes the most promising computing and service paradigm. The National Institute of Standards and Technology (NIST) has released the definition of it [1]. Cloud computing is a model for enabling ubiquitous, convenient, and on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. The cloud model defined in [1] also outlines five essential characteristics and three service models. The three service models are SaaS, PaaS,

Table 1: Amazon EC2 pricing table.

	Linux/UNIX usage	Windows usage
Standard on-demand instances		
Small (default)	\$0.085 per hour	\$0.12 per hour
Large	\$0.34 per hour	\$0.48 per hour
Extra large	\$0.68 per hour	\$0.96 per hour
Micro on-demand instances		
Micro	\$0.02 per hour	\$0.03 per hour
Hi-memory on-demand instances		
Extra large	\$0.50 per hour	\$0.62 per hour
Double extra large	\$1.00 per hour	\$1.24 per hour
Quadruple extra large	\$2.00 per hour	\$2.48 per hour
Hi-CPU on-demand instances		
Medium	\$0.17 per hour	\$0.29 per hour
Extra large	\$0.68 per hour	\$1.16 per hour

and IaaS. SaaS lets consumers use the provider's applications running on a cloud infrastructure, for example, Gmail and Google Docs. Consumers generally access those applications via a web-based interface such as a browser from various thin client devices. PaaS and IaaS are similar in a manner. They both provide consumers computing resources like hardware (e.g., servers, networks, and storage) and software (operating systems, and databases). Consumers of PaaS can use programming languages and tools supported by the PaaS providers to create specific applications and deploy them onto the particular cloud platform of PaaS providers, while the consumers of IaaS providers are able not only to deploy and run arbitrary applications but also to setup their desired environment such as operating systems and runtime libraries.

Indubitably, the most significant core in cloud computing technologies is virtualization [2]. By utilizing virtualization in a datacenter, processing, storage, networks, and other computing resources are provided to cloud consumers with ease. This resource pooling characteristic derives the service model so-called Infrastructure as a Service (IaaS). Providers of IaaS such as Amazon EC2 [3] pack fundamental computing resource in the form of virtual machines (VMs) and let the cloud consumers deploy and run arbitrary software, which can include operating systems and applications. Due to the on-demand and rapid elasticity, the outsourcing of computing resources with providers of IaaS is quite beneficial for cloud consumers like various providers of Internet application.

However, this outsourcing trend arises a key issue of resource allocation owing to pay-per-use business model of IaaS like Table 1. Consumers are charged for their usage of these IaaS services.

Consumers pay for compute capacity by the hour with no long-term commitments. This frees consumers from the costs and complexities of planning, purchasing, and maintaining hardware and transforms what are commonly large fixed costs into much smaller variable costs. The pricing in Table 1 includes the cost to run private and public AMIs on the specified operating system. Pricing is per instance-hour consumed for each instance, from the time an instance is launched until it is terminated. Each partial instance-hour consumed will be billed as a full hour.

The concept is from Utility Computing and Services Computing. In fact, after water, electricity, gas, and telephony, there is an increasingly perceived vision that computing will be the 5th utility one day. Thus, a resource provisioning mechanism that both prevents underprovisioning in order to assure QoS via Service Level Agreement (SLA) and avoids overprovisioning so as to reduce cost becomes a crucial priority and challenge in the design and operation of complex service-based platforms such as streaming service. On the other hand, cloud infrastructure providers also concern their profit performance and energy consumption. Profit performance is the profitability. Sustaining profitability and growth is the main target for these companies and corporations. Energy consumption is a critical topic not only in environmental issues but also in reducing operating costs for electricity. Those criteria for both service-based platforms and cloud infrastructure providers are important and may be conflicting.

In this paper, contemplating both service-oriented and infrastructure-oriented criteria, we regard this resource allocation problem as Multicriteria Decision Making problem. Our intention is to design a trade-off based strategy and propose an effective resource provisioning algorithm for an autonomous resource broker in the cloud, as shown in Figure 1. After the streaming service platform evaluates the requirements and defines a mapping between the requests' service level requirements and resource level requirements, the autonomous resource broker will regulate the supply and demand of cloud resources between the streaming service platform and the cloud infrastructure provider based on not only incoming requests of the streaming service but also both the service-oriented criteria which are in the form of SLA and infrastructure-oriented criteria. The solution of the algorithm is obtained by formulating and solving a goal programming model. In particular, a cloud architecture for streaming application is addressed as well as extensive analysis and experiments are performed for related criteria. The results of numerical simulations show that the proposed approach strikes a balance between these conflicting criteria commendably and achieves high cost efficiency.

The rest of this paper is organized as follows. In Section 2, related works are reviewed. The system model and assumption of cloud architecture for streaming service are described in Section 3. Section 4 details the analysis of service-oriented and infrastructure-oriented criteria. In Section 5, the goal programming model and algorithm are depicted. Simulations are presented in Section 6. Finally, Section 7 summarizes our conclusions and outlines our future works.

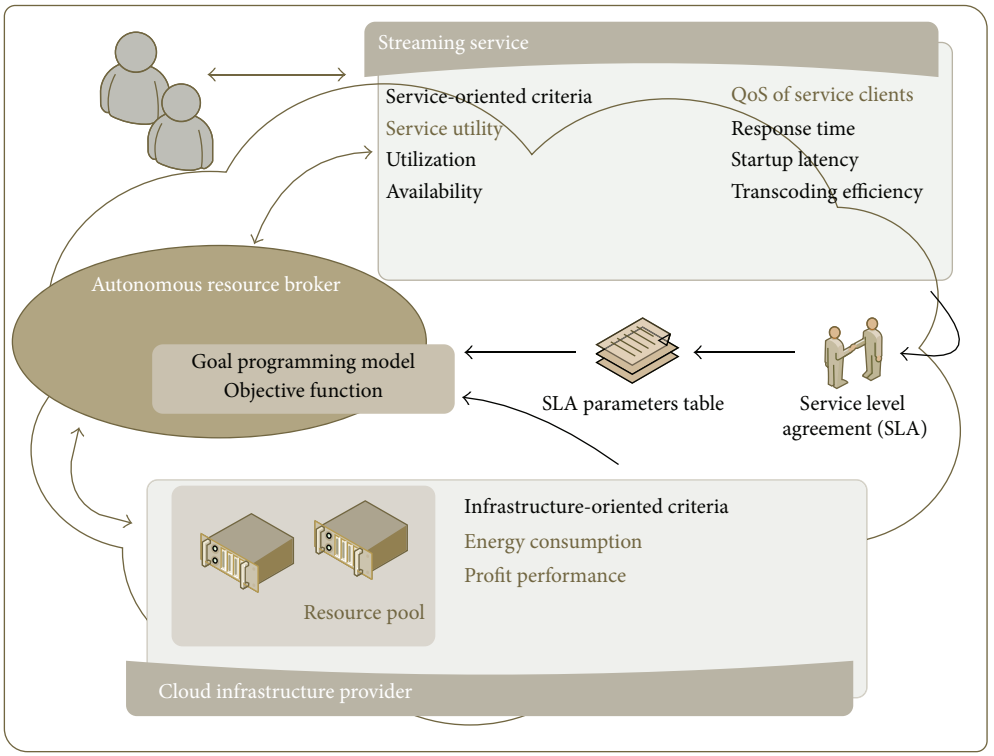

Figure 1: Overview of resource brokering for streaming service in the cloud.

2. Related Works

There are many research papers probing into resource allocation and provisioning in cloud computing. Here we can sum up the three main steps of resource allocation in cloud.

2.1. Evaluate the Requirements. Evaluate and define a mapping between service level requirements and resource level requirements. Service level requirements are generally defined in SLA based on specific parameters such as availability, response time. Resource level requirements are often outlined as cores, memory, bandwidth, and so forth. This step also requires performance and capacity modeling.

2.2. Resource Brokering and Provisioning. This is the main step of the whole resource allocation in cloud since the determination of the number of cloud resources such as virtual machines reflects not only the efficiency about the utilization, QoS, and operating cost of cloud infrastructure consumers but also the performance and profit of cloud infrastructure providers.

2.3. Distribute Resources in Physical Machines. Distributing the virtualized resources allocated in data center into the physical machines. In general, this step is defined as a Knapsack Problem or Bin Packing Problem. This step takes data center utilization, the migration overhead, and so forth, into account and needs further studies as well as analysis for virtualization.

Reference [4] proposes an approach for dynamic resource management in cloud which adapts a distributed architecture where resource management is decomposed into independent tasks, each of which is performed by Autonomous Node Agents through Multiple Criteria Decision Analysis using the PROMETHEE method. Thus [4] deals with C. Distribute resources in physical machines. Reference [5] proposes a resource management framework combining a utility-based dynamic VM provisioning manager and a dynamic VM placement manager. Both problems are modeled as Constraint Satisfaction Problems. And [5] deals with B. Resource brokering and provisioning and C. Distribute resources in physical machines. Reference [6] proposes an approach to managing infrastructure resources in PaaS by leveraging two adaptive control loops. The optimization loop improves the resource utilization of a cloud application via management functions provided by the corresponding middleware layers of PaaS. The allocation loop provides appropriate amounts of resources to/from the application system while guaranteeing its performance.

Based on [7] we know in such a Service-Oriented Architecture like cloud, the quality and reliability of the services become important aspects. However the demands of the service consumers vary significantly. From the service provider perspective, a balance needs to be made via a negotiation process since it is not possible to fulfill all consumer expectations. At the end of the negotiation process, provider and consumer commit to an agreement. In SOA terms, this agreement is referred to as a SLA (Service Level Agreement). The SLA serves as the foundation for the expected level of

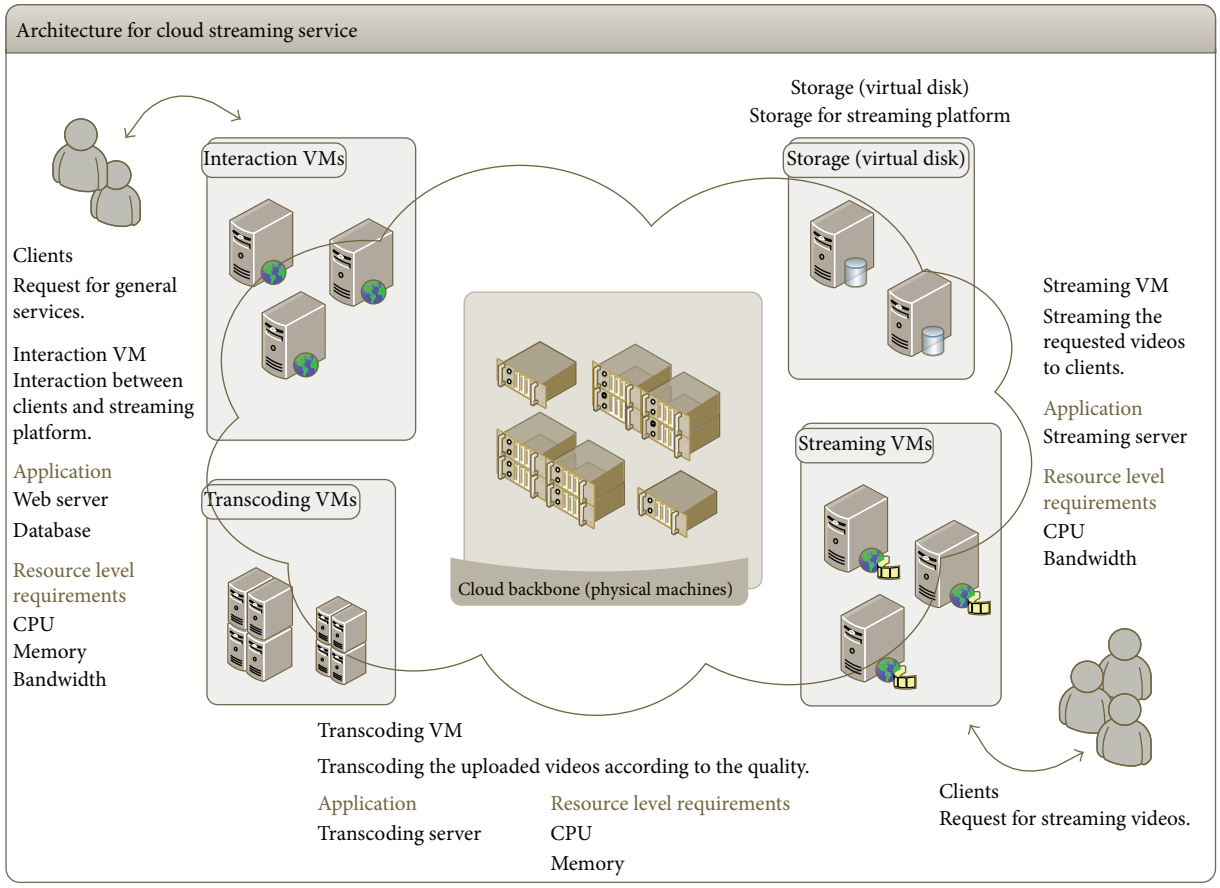

Figure 2: Cloud architecture for streaming service.

service between the consumer and the provider. In general, QoS requirements are part of an SLA.

SLA parameters are specified by metrics and usually included in a SLA parameters table. Metrics define how service parameters can be measured and are typically functions. There are at least two major types of metrics: (1) Resource metrics are retrieved directly from the provider resources and are used as is without further processing. (2) Composite metrics represent a combination of several resource metrics, calculated according to a special algorithm.

3. Cloud Streaming Service

3.1. Cloud Architecture for Streaming Service. As shown in Figure 2, the general system architecture of cloud computing environment for a typical streaming service, including but not limited to Video on Demand (VoD), live broadcasting, IPTV, and so forth, comprises numerous VMs. With the support of virtualization, the VMs can operate on the cloud backbone which consists of thousands of physical machines. These VMs can be categorized into four main types as follows.

(i) Interaction VMs (IVMs). They are responsible for the interaction between clients and streaming service platform. Those VMs host basic applications and services for platform operation, for example, web-based user interfaces and functions and database services. They also coordinate other types of VMs such as redirecting clients' requests for streaming videos to SVMs or queuing clients' requests for uploading videos, allocating enough storage, and assigning videos to TVMs. When considering the factors of resource requirements, bandwidth is the major concern for service capacity. Since many large websites use an in-memory datagrid for caching, applications of IVMs are mostly memory-intensive. Some applications are CPU-intensive due to their architecture.

(ii) Streaming VMs (SVMs). They are responsible for streaming the requested videos to clients. While receiving redirected requests from IVMs, the streaming applications hosted on SVMs process and packetize the source videos for specific format and protocol and then stream them to clients who issue the requests. The factors of resource requirements for SVMs are CPU and bandwidth.

(iii) Transcoding VMs (TVMs). Their duties are transcoding the uploaded videos according to the quality. Those VMs are responsible for transcoding source videos into varied quality videos with custom format settings. Transcoding is both CPU-intensive and memory-intensive. For many advanced encoders nowadays, multicore architectures with

high memory provision speed up the processes dramatically. The factors of resource requirements for TVMs are CPU and memory.

(iv) Storage. It is responsible for storage for streaming service platform. It usually consists of a large number of disk arrays. With the block virtualization and file virtualization, flexible management and optimization of storage utilization and server consolidation are achieved.

Note that those specific VM types are designed for VoD streaming services by the streaming service platform. Thus the mapping to VM instances of the IaaS providers as in Table 1 relies on further performance analysis by the streaming service platform. And those custom VM types may alter depending on the particular requirements of different services as well.

3.2. System Model and Assumption. In our proposed cloud streaming service model, there is a resource pool, P, comprising a set of physical machines (PM), $P = \{\mathrm{PM}_1, \mathrm{PM}_2, \ldots, \mathrm{PM}_n\}$. Assume all the physical machines are homogeneous in machine architecture and possess equivalent size, $\mathrm{PM}_{\mathrm{size}}$, for each considered resource factor. Through server virtualization/consolidation, a number of VMs can share a single PM, which increases utilization and in turn reduces the total number of PMs required. At any given time, we assume that n_{IVM}, n_{TVM}, and n_{SVM} are the amount of IVM, TVM, and SVM.

Based on the analysis of operations for streaming service, we outline three VM types as described above and an ideal resource mapping between the host application and their requirement for these types of VMs. Each VM type possesses varied price, PRI, and different capacity, CAP. The former is the cost of the streaming service platform to run a single VM for the duration, for example, \$0.075 per hour for $\mathrm{IVM}_{\mathrm{PRI}}$; the latter is the maximum capacity of one VM, for instance, 500 simultaneous connections for $\mathrm{IVM}_{\mathrm{CAP}}$. Generally speaking, there is a resource mapping table including PRI and CAP, for each type of VMs. This table can be obtained by modifying the pricing table from cloud infrastructure providers, such as Table 1.

At any given time, there is a set of requests R consisting of assorted tasks t which are classified by their VM host, namely, T_{IVM}, T_{TVM}, and T_{SVM}. T_{IVM} is a subset of R that consists of tasks for IVM; T_{SVM} and T_{TVM} are defined in the similar way; therefore $R = \{T_{\mathrm{IVM}}, T_{\mathrm{SVM}}, T_{\mathrm{TVM}}\}$. If there are i tasks of T_{IVM}, j tasks of T_{SVM}, and k tasks of T_{TVM}, then it must satisfy the following:

$$\begin{aligned} T_{\mathrm{IVM}} &= \left\{t_1^{\mathrm{IVM}}, t_2^{\mathrm{IVM}}, \ldots, t_i^{\mathrm{IVM}}\right\}, \\ T_{\mathrm{SVM}} &= \left\{t_1^{\mathrm{SVM}}, t_2^{\mathrm{SVM}}, \ldots, t_j^{\mathrm{SVM}}\right\}, \\ T_{\mathrm{TVM}} &= \left\{t_1^{\mathrm{TVM}}, t_2^{\mathrm{TVM}}, \ldots, t_k^{\mathrm{TVM}}\right\}, \\ &\text{s.t. } i + j + k = |R|. \end{aligned} \tag{1}$$

All request arrival rates follow different distributions. And each type of request possesses diverse execution time based on their characteristics as follows:

(i) Characteristics of T_{IVM} task: short-term execution, noncomputation-intensive, and strict response time.

(ii) Characteristics of T_{TVM} task: long-term execution, computation-intensive, and loose response time.

(iii) Characteristics of T_{SVM} task: execution time depending on media length, noncomputation-intensive, and strict response time.

Further numerical details about arrival rates and execution time distributions of requests will be discussed in Section 6. Also we list the key notations used in this paper in Notations. Note that it includes not only the notations mentioned in this section but also those defined in Section 4.

4. Criteria for Streaming Service

Although streaming is a well-developed application for the Internet, it can still be enhanced by leveraging the cloud technology such as parallel and distributed computing. Also, requests of a streaming service are quite varied in their resource requirements and execution time and this feature definitely fits in with the characteristics of on-demand and rapid elasticity in the cloud. These observations lead us to study streaming service as the service platform model. As illustrated in Figure 1, our resource broker adopts and transforms those criteria into the objective functions of the goal programming model. The criteria concerned by the streaming service and the cloud infrastructure provider, namely, *Service-oriented Criteria* and *Infrastructure-oriented Criteria*, are explored as follows.

4.1. Service-Oriented Criteria. *Service Utility*, including *Utilization* and *Availability*, is a set of criteria which measure the resource utilization and the accessibility and serviceability of the service platform. A resource is said to be critical to performance when it becomes overused or when its utilization is disproportionate to that of other components. On the other hand, availability means the percentage of time that the streaming service is available to process requests of clients.

(i) Utilization. It is assumed that VMs of the same type are load-balanced automatically. Thus utilization of IVM at any given time is $\mathrm{util}_{\mathrm{IVM}}$ as follows. Utilization of TVM and SVM are defined in the same way:

$$\mathrm{util}_{\mathrm{IVM}} = \begin{cases} 1, & \left|T_{\mathrm{IVM}}\right| \geq n_{\mathrm{IVM}} \times \mathrm{IVM}_{\mathrm{CAP}}, \\ \dfrac{\left|T_{\mathrm{IVM}}\right|}{n_{\mathrm{IVM}}}, & \left|T_{\mathrm{IVM}}\right| < n_{\mathrm{IVM}} \times \mathrm{IVM}_{\mathrm{CAP}}. \end{cases} \tag{2}$$

(ii) Availability. Availability of VM at any given time, $\mathrm{avail}_{\mathrm{IVM}}$, is defined in a similar way as *Utilization* and average availability of VM, $\mathrm{Avail}_{\mathrm{IVM}}$, during a time period, time_period, is as follows. Availability of TVM and SVM are defined in the same way:

$$\mathrm{Avail}_{\mathrm{IVM}} = \frac{\sum \mathrm{avail}_{\mathrm{IVM}}}{\mathrm{time\_period}}. \tag{3}$$

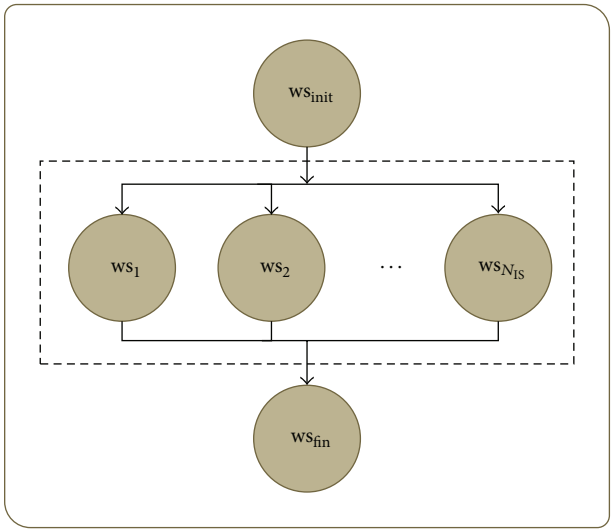

Figure 3: A composite web service.

QoS of Service Clients, which involves *response time*, *startup latency*, and *transcoding efficiency*, is a set of criteria which evaluate the Quality of Service (QoS) of the streaming service platform. Response time is concerned for IVM, while *startup latency* and *transcoding efficiency* are for SVM and TVM, respectively.

(iii) Response Time. Many large scale web sites take advantage of web services to boost their traffic. A single frontend Internet application may invoke many different web services. Such applications are called composite web services [8–10]. For example, browsing the streaming service platform by a registered member may invoke the web services of customization, recommendation, sorting, searching, and so forth.

Figure 3 is a model of a composite web service. After an initialization procedure, $\mathrm{WS}_{\mathrm{init}}$, N_{IS} web services are invoked and executed in parallel. Those parallel invocations of N_{IS} Web services will run on different IVMs and utilize the number of IVMs to speed up. The final procedure can only be carried after all N_{IS} web services have completed. Based on the analysis of the composite web service [8], we derive a function which can be measured in terms of utilization, the number of IVMs, and concurrent requests for IVM. Assume $\mathrm{ws}_{\mathrm{init}}$ and $\mathrm{ws}_{\mathrm{fin}}$ are the time of initial and final procedures in a composite web service and ws_i is the time it takes to execute web service i ($i = 1, \ldots, N_{\mathrm{IS}}$). Then the time it takes to execute N_{IS} web services that must synchronize after all completed is the maximum processing time among these N_{IS} web services. From [11–13], we know that based on queuing theory, as resource utilization increases, the response time per request rises dramatically. Reference [11] also derives a general formula to predict average response time:

$$\text{response } T = \frac{1}{1-p} \times \text{service } T, \tag{4}$$

where response T is average response time, service T is service time, and p is utilization. From Figure 3 and above we can formulate the response time rt of IVM at any given time as follows:

$$\mathrm{rt} = \mathrm{ws}_{\mathrm{init}} + \frac{1}{1-\mathrm{util}_{\mathrm{IVM}}} \left(\frac{N_{\mathrm{IS}}}{n_{\mathrm{IVM}}} \times \max_{1 \le i \le N_{\mathrm{IS}}} \{\mathrm{ws}_i\} \right) + \mathrm{ws}_{\mathrm{fin}}, \quad \text{s.t. } \frac{N_{\mathrm{IS}}}{n_{\mathrm{IVM}}} \ge 1, \ \mathrm{util}_{\mathrm{IVM}} \ne 1. \tag{5}$$

(iv) Startup Latency. As computers and network devices compress, encode, distribute, decompress, and render large amounts of data, buffers play a significant role in assuring the quality of digital media processing. In general, the larger the buffer, the better the end-user experience, but there is one main disadvantage of large buffers in a streaming scenario: Buffers cause delays or latencies. As shown in Figure 4, startup latency is the time a streaming service of SVM receives a request and starts media source processing, packetizing, and transmitting to the time that client fills up its buffer and starts playing. Here we focus on only buffering delay for streaming server. Thus network jitter, client download rate, and so forth, which can also affect startup latency, are not considered. According to the studies in [14–17], single-server streaming systems employ the server-push delivery model while client-pull architecture is more appropriate for multiserver streaming systems. Thus for our streaming service platform, client-pull architecture is assumed. In order to reduce startup latency, the basic idea is that clients need to fill up their buffers as fast as possible. It is vulnerable to network jitters if we tune down the buffer size of clients. Nevertheless, a client can issue multiple requests simultaneously for the video data segments and multiple streaming servers could be used for more throughput than a single server can provide.

The so-called server striping [15, 16] technique divides streaming video into fixed-size segments and distributes those segments over all SVMs. This helps us derive a function which can be estimated in terms of concurrent requests for SVM and the number of SVMs. We assume the policy of segment placement such as round-robin is adopted and the size of a streaming video is much larger than the size of a segment so that load imbalance due to uneven allocation between servers can be ignored. As illustrated in Figure 5, utilizing server striping technique to reduce the time needed to fill up the buffer, we assume the number of frames L which a buffer of client player contains is fixed and the average video processing frame rate for SVM is F. Then startup latency sl of SVM at any given time is defined as follows:

$$\mathrm{sl} = \frac{1}{n_{\mathrm{SVM}}} \times \frac{L}{F}, \quad \text{s.t. } \mathrm{avail}_{\mathrm{SVM}} = 1. \tag{6}$$

(v) Transcoding Efficiency. Based on the analysis of [18, 19], we assume that distributed transcoding architecture is used for the streaming service platform and we can exploit the cloud's elasticity to engage resources dynamically. Split and merge [20] technique enables us to split source video into segments

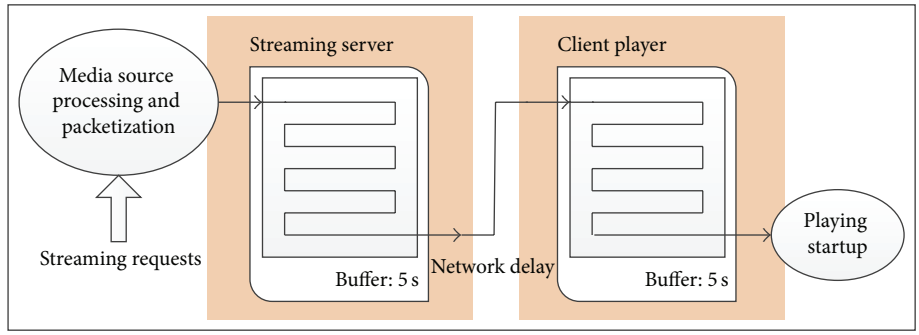

Figure 4: Startup latency in streaming.

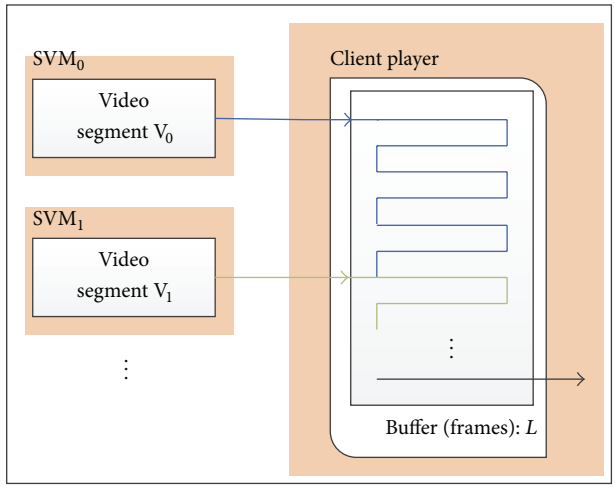

Figure 5: Reduce startup latency by server striping.

and distribute those segments over all TVMs to parallelize the video transcoding processes. Using this architecture, we assume the transcoding time of a single server is T_s and the transcoding time for a length of vt_time video of a cloud-based transcoding system of n_{TVM} TVMs is T_c. On system level, the gain of transcoding time from split and merge technique can be derived straightforwardly:

$$\frac{T_s}{T_c} \cong \frac{\text{vt\_time}}{\text{vt\_time} \times 1/n_{\text{TVM}}} \tag{7}$$

and as resource utilization increases, the transcoding time per request rises dramatically. Assume the degradation coefficient of utilization is α. Then transcoding efficiency te of TVM at any given time is

$$\text{te} = \alpha\left(1 - \text{util}_{\text{TVM}}\right) \times n_{\text{TVM}} \quad \text{s.t. } \text{util}_{\text{TVM}} < 1. \tag{8}$$

As mentioned in Section 1, those criteria with regard to *Service Utility* and *QoS of Service Clients* are defined as a SLA parameters table, which contains a set of QoS parameters and constraints as Table 2.

4.2. Infrastructure-Oriented Criteria. *Energy Consumption* and *profit performance* are criteria which measure the system energy consumption and profit of the cloud infrastructure providers.

Table 2: SLA parameters table.

SLA parameter	Constraint
Utilization	$\text{util}_{\text{IVM}} \geqq \text{constraint\_util}_{\text{IVM}}$ $\text{util}_{\text{SVM}} \geqq \text{constraint\_util}_{\text{SVM}}$ $\text{util}_{\text{TVM}} \geqq \text{constraint\_util}_{\text{TVM}}$
Availability	$\text{Avail}_{\text{IVM}} \geqq \text{constraint\_Avail}_{\text{IVM}}$ $\text{Avail}_{\text{SVM}} \geqq \text{constraint\_Avail}_{\text{SVM}}$ $\text{Avail}_{\text{TVM}} \geqq \text{constraint\_Avail}_{\text{TVM}}$
Response time	$\text{rt} \leqq \text{constraint\_rt}$
Startup latency	$\text{sl} \leqq \text{constraint\_sl}$
Transcoding efficiency	$\text{te} \leqq \text{constraint\_te}$

(i) Energy Consumption. Through server virtualization, a large number of users can share a single physical machine, which increases utilization and in turn reduces the total number of physical machines required. Reasonably cutting down the number of VMs can reduce the number of physical machines needed and achieve energy conservation. Since the services provided by IVM and SVM use a Software as a Service (SaaS) model, we can modify and employ the per-user energy consumption model for SaaS based on (9) in [21]:

$$P_{\text{sf}} = P_{\text{sf,PC}} + \frac{1.5P_{\text{sf,SR}}}{N_{\text{sf,SR}}} + 2B_d\frac{1.5P_{\text{SD}}}{B_{\text{SD}}} + AE_T. \tag{9}$$

The P_{sf} is per-user energy consumption of the SaaS service; $P_{\text{sf,PC}}$ is the power consumption of the user's terminal; $P_{\text{sf,SR}}$ is the power consumption of the server; $N_{\text{sf,SR}}$ is the number of users per server; B_d (bits) is the average size of a accessing file; P_{SD} is the power consumption of the hard disk arrays; B_{SD} is the capacity of the hard disk array; A is the transmission bit rate (bits per second); E_T is the per-bit energy consumption of transport in cloud computing. The multiplication by a factor of 2 in the third term accounts for the power requirements for redundancy in storage and the multiplication by a factor of 1.5 for second and third terms accounts for the energy consumption in cooling as well as other overheads.

Our main intention here is to derive functions to evaluate only the energy consumption of running physical machines for IaaS providers based on (9). Therefore we can eliminate the first term $P_{\text{sf,PC}}$ since the end-user's energy consumption is not taken into account by service platform providers. The third term $2B_d(1.5P_{\text{SD}}/B_{\text{SD}})$ is ignored due to the fact that we do not consider storage issues in this paper. Also the last term AE_T which represents energy consumption of network transmission can be eliminated. From above and with our own notations, we can modify (9). Let EC_{IVM} and EC_{SVM} be the energy consumption of all physical machines which host all the IVMs and SVMs:

$$\begin{aligned}\text{EC}_{\text{IVM}} &= \frac{P_{\text{sf}} \times \text{IVM}_{\text{CAP}}}{1.5} \times \frac{n_{\text{IVM}}}{\text{PM}_{\text{size}}},\\ \text{EC}_{\text{SVM}} &= \frac{P_{\text{sf}} \times \text{SVM}_{\text{CAP}}}{1.5} \times \frac{n_{\text{SVM}}}{\text{PM}_{\text{size}}}.\end{aligned} \tag{10}$$

For services of TVM, per-user energy consumption model for Processing as a Service is adopted on the following basis (11) mentioned in [21]:

$$E_{\text{ps}} = 40P_{\text{ps,PC}} + 1.5NT_{\text{ps,SR}}P_{\text{ps,SR}} + 168AE_T. \tag{11}$$

The above per-user energy consumption (watt hours) E_{ps} is formulated as a function of the number of encodings per week N. $P_{\text{ps,PC}}$ is the power consumption of the user's laptop; $T_{\text{ps,SR}}$ is the average number of hours it takes to perform one encoding and $P_{\text{ps,SR}}$ is the power consumption of the server.

The user's PC is used on average 40 h/week for common office tasks (factor of 40 in first term). A factor of 1.5 is included in the second term to account for the energy consumed to cool the computation servers as well as other over heads. In the third term, A is the per-user data rate (bits per second) between each user and the cloud, E_T is the per-bit energy consumption of transport, and the factor of 168 converts power consumption in transport to energy consumption per week (watt hours).

Here again the first term $40P_{\text{ps,PC}}$ and the third term $168AE_T$ can both be eliminated due to the fact that end-user's energy consumption and energy consumption of network transmission are not taken into consideration.

Assume EC_{TVM} is the energy consumption of all physical machines which host all the TVMs:

$$\text{EC}_{\text{TVM}} = \frac{E_{\text{ps}} \times \text{TVM}_{\text{CAP}}}{1.5 \times N \times T_{\text{ps,SR}}} \times \frac{n_{\text{TVM}}}{\text{PM}_{\text{size}}} \tag{12}$$

and from above we can derive energy consumption ec at any given time:

$$\text{ec} = \text{EC}_{\text{IVM}} + \text{EC}_{\text{SVM}} + \text{EC}_{\text{TVM}}. \tag{13}$$

(ii) Profit Performance. This criterion is evaluated by a function which can be measured in terms of the number of all types of VMs and their relative cost PRI. Therefore the profit performance pp can be formulated as follows:

$$\text{pp} = n_{\text{IVM}}\text{IVM}_{\text{PRI}} + n_{\text{SVM}}\text{SVM}_{\text{PRI}} + n_{\text{TVM}}\text{TVM}_{\text{PRI}}. \tag{14}$$

5. Goal Programming Model

5.1. MCDM and Goal Programming. Multicriteria Decision Making (MCDM) is a mathematical programming discipline under multiple objectives. It has emerged as a powerful tool to assist in the process of searching for decisions which best satisfy a multitude of conflicting objectives, and there are a number of distinct methodologies for multicriteria decision-making problems that exist. These methodologies can be categorized in a variety of ways, such as form of model (e.g., linear, nonlinear, and stochastic), characteristics of the decision space (e.g., finite or infinite), or solution process (e.g., prior specification of preferences or interactive). There are already many developed MCDM methods in use today and goal programming is one of the most popular and well-known techniques among them.

Goal programming (GP) is a multiobjective optimization technique which can cope with Multicriteria Decision Making problems. The essence of GP consists in the concept of satisfying of objectives. In fact, real-world problems invariably involve nondeterministic systems for which a variety of conflicting, noncommensurable objectives exist [22]. Due to the conflicts of objectives and the incompleteness of available information, it is almost impossible to build a reliable mathematical representation of the decision makers' preferences which has an optimal solution that optimizes all the objective functions. On the contrary, within such decision environment the decision makers try to achieve a set of goals which are represented by objective functions and constraints as closely as possible.

GP models can be classified into two major subsets [23], namely, weighted GP and preemptive/lexicographic GP. The general formulation of a weighted GP is given as follows:

$$\begin{aligned}\min \quad & z = \sum_{i=1}^{k} \left(u_i n_i + v_i p_i\right)\\ \text{s.t.} \quad & f_i(x) + n_i - p_i = b_i,\\ & i = 1 \cdots Q,\end{aligned} \tag{15}$$

where $f_i(x)$ is a linear or nonlinear objective function of x and b_i is the constraint for that objective. The unwanted negative and positive deviations n_i and p_i are assigned weights according to their relative importance to the decision maker and minimized.

In preemptive GP, the deviational variables are assigned into a number of priority levels and minimized in a preemptive way. The formulation of a preemptive GP is given as follows:

$$\begin{aligned}\min \quad & a = \left\{g_1(n,p), g_2(n,p), \ldots, g_L(n,p)\right\}\\ \text{s.t.} \quad & f_i(x) + n_i - p_i = b_i,\\ & i = 1 \cdots Q.\end{aligned} \tag{16}$$

In this model, α is an ordered vector of these L priority levels and g_L is a function of the deviation variables associated with the objectives or constraints at priority level L.

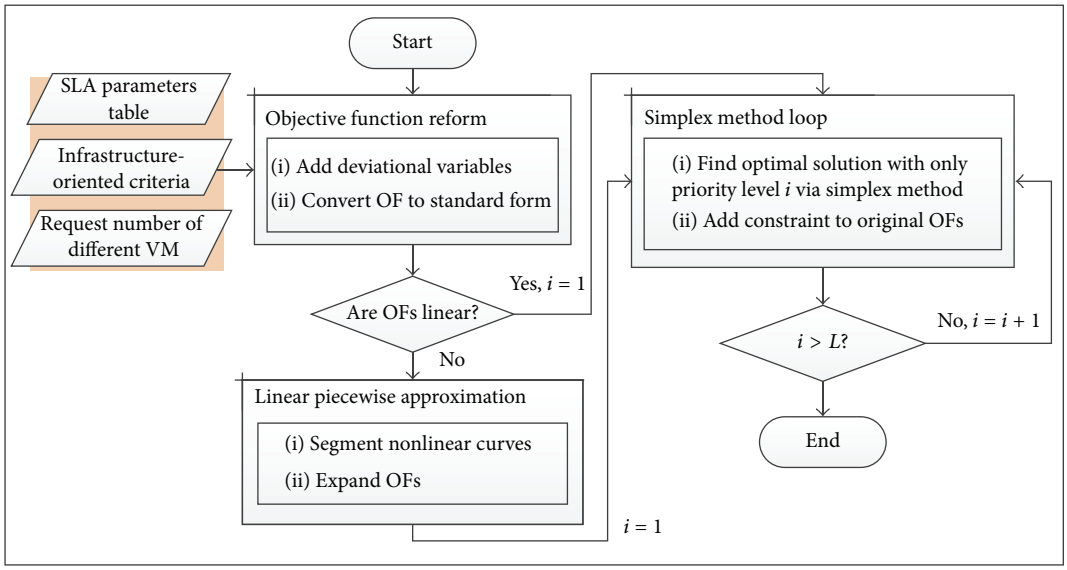

Figure 6: The goal programming procedure.

5.2. Problem Formulation. The resource allocation/provisioning problem in our autonomous resource broker is to judge the quantity of IVMs, TVMs, and SVMs while deliberating upon those conflicting criteria. Let n_{IVM}, n_{TVM}, and n_{SVM} be the number of IVM, TVM, and SVM. The determination of n_{IVM}, n_{TVM}, and n_{SVM} is a Multiple Criteria Decision Making problem for satisfying contrary goals of service-oriented and infrastructure-oriented. This problem can be formulated and solved based on preemptive Goal Programming. Assume p_i is the unwanted positive deviation which numerically exceeds the ith goal and let n_i be the unwanted negative deviation which falls short of the ith goal. We can express this problem as follows:

$$\min \quad \{(p_4 + p_5 + n_6), (n_1 + n_2 + n_3), (p_7 + n_8)\} \tag{17}$$

$$\text{subject to} \quad \text{util}_{\text{IVM}} + n_1 = \text{constraint\_util}_{\text{IVM}} \tag{18}$$

$$\text{util}_{\text{SVM}} + n_2 = \text{constraint\_util}_{\text{SVM}} \tag{19}$$

$$\text{util}_{\text{TVM}} + n_3 = \text{constraint\_util}_{\text{TVM}} \tag{20}$$

$$\text{rt} - p_4 = \text{constraint\_rt} \tag{21}$$

$$\text{sl} - p_5 = \text{constraint\_sl} \tag{22}$$

$$\text{te} + n_6 = \text{constraint\_te} \tag{23}$$

$$\text{ec} - p_7 = \text{constraint\_ec} \tag{24}$$

$$\text{pp} + n_8 = \text{constraint\_pp}. \tag{25}$$

The equations above from (18) to (25) are objective functions of the goal programming model. These equations are formulations of related criteria derived in Section 4 and each of these equations has a specific constraint defined by the streaming service platform. Equations (18), (19), and (20) stand for *Utilization* of IVM, SVM, and TVM, and (21), (22), and (23) are regarded as *response time*, *startup latency*, and *transcoding efficiency*, respectively. And the last two equations, (24) and (25), are viewed as *energy consumption* and *profit performance*.

From the perspective of the streaming service platform, *Service-oriented Criteria* are always more important than *Infrastructure-oriented Criteria*. And criteria for *QoS of Service Clients* are considered first compared to criteria for *Service Utility*. Thus in (17), we first minimize n_4, n_5, and p_6 which represent the unwanted deviations for *response time*, *startup latency*, and *transcoding efficiency*. Then we minimize p_1, p_2, and p_3 that stand for the unwanted deviations for *Utilization* of IVM, SVM, and TVM. Lastly we minimize n_7 and p_8 which count as the unwanted deviations for *energy consumption* and *profit performance*.

5.3. Goal Programming Approach. In general, for each priority level, our goal programming approach identifies basic feasible solutions and refines them iteratively using Simplex algorithm to achieve the best possible compromise solution. The detailed procedure is illustrated in Figure 6. There are three main processes, namely, *Objective Function Reform*, *Linear Piecewise Approximation*, and *Simplex Method Loop*.

(i) Objective Function Reform. In this procedure, first it imports mathematical formulation and constraints from SLA Parameters table (derived from service-oriented criteria) and infrastructure-oriented criteria as the objective functions of goal programming model. Then the deviational variables are defined and added to all the objective functions since it is not known whether a given solution will undersatisfy or oversatisfy the goals or constraints set by criteria.

We seek to minimize the nonachievement of the goals or constraints by minimizing specific deviation variables. Before the simplex method can be used to solve, all the objective functions must be converted into an equivalent functions

Table 3: Object function reform.

Goal or constraint	Objective function	Deviational variables to be minimized
$f_i(x) \leqq b_i$	$f_i(x) + n_i - p_i = b_i$	p_i
$f_i(x) \geqq b_i$	$f_i(x) + n_i - p_i = b_i$	n_i
$f_i(x) = b_i$	$f_i(x) + n_i - p_i = b_i$	$n_i + p_i$

in which all constraints are equations and all variables are nonnegative. The way of adding the deviational variables and converting objective functions into standard form can be summarized in Table 3. The priority of each goal is also adjusted according to the relative importance of service-oriented criteria and infrastructure-oriented criteria, as well as the request number for IVM, SVM, and TVM.

(ii) Linear Piecewise Approximation. After reforming the objective functions, we should verify if these objective functions are linear. As with many real-world problems the functional representations of the objective functions were mostly nonlinear.

Reference [24] gives a method of modeling any monotonically increasing or decreasing, nonlinear, and discontinuous function while remaining within the GP format. Generally speaking, nonlinear objective function curves can be segmented by a piecewise linear approximation according to [25] and the resulting straight-line segments can then be modeled using penalty or reverse penalty function methods which expands the original objective functions as described in [24]. Apparently, the more the number of segments used, the greater the accuracy in modeling the objective functions. Nevertheless using more segments also increases the size of the resulting GP model.

(iii) Simplex Method Loop. After objective function reform and linear piecewise approximation, a standard linear GP model is built. Thus we can construct a sequential goal programming code to solve our model in this procedure. The mechanism is described as follows:

(a) Let i be the priority level under consideration and L is the total priority level. Set $i = 1$.

(b) Solve priority level i only. That is to say, minimize $\alpha_i = g_i(n, p)$, subject to only the goals or constraints associated with i. Such a problem is equivalent to a traditional single-objective model which can be solved via simplex method. Let the optimal solution to this problem be given as α_i^*; namely, α_i^* is the optimal solution for $g_i(n, p)$.

(c) Set $i = i + 1$; if $i > L$, go to (d). Add constraint for the optimal solution of previous priority levels; namely, $g_i(n, p) = \alpha_j^*$, $j = 1, \ldots, i - 1$, to original goal programming model for the next priority level. Then go to (b).

(d) The end of the loop and the solution vector, associated with the last single objective model solved, is the optimal vector for the original goal programming model.

The simplex method/algorithm is a well-known algorithm of solving linear programming problems. It is created by George Dantzig in 1947 and used for planning and decision-making in large-scale enterprises. Based on chapter 4 of [26], the general procedure of simplex algorithm is given as follows.

Step 1. Convert the linear programming problem to the standard form.

Step 2. Obtain a BFS (basic feasible solution) from the standard form.

Step 3. Determine whether the current BFS is optimal. If current BFS is not optimal, go to Step 4, else the procedure reaches the end.

Step 4. If the current BFS is not optimal, then find a new BFS with a better objective function value. Then go to Step 3.

The concept of this algorithm is derived from the name "simplex." Simplex is a generalization of the notion of a triangle or tetrahedron to arbitrary dimension and the geometrical interpretation of the behavior of simplex algorithm is that its search procedure follows a simplex and essentially starts from some initial corner point and then follows a path along the edges of the feasible region towards an optimal corner point. Note that all the intermediate corner points visited are improving (more precisely, not worsening) the objective function.

6. Performance Evaluation

In this section, six criteria are considered and evaluated, namely, Utilization of three types of VMs, *response time*, *startup latency*, *transcoding efficiency*, *energy consumption*, and *profit performance*. In order to judge the performance of the proposed goal programming approach, we compared it with a utility-based model which is adopted in [27]. Its concept has been used in microeconomic theory. Here we modify it with our notations and develop it as utility-based model. Let utilization and price be the measured utility function of the utility-based model.

We use MATLAB R2010a as our simulation tool and the settings of parameters are shown in Table 4. According to probability theory and statistics, we simulate the demand for T_{IVM}, T_{SVM}, and T_{TVM} per hour using Poisson distribution with various values of λ. The "off-peak" zones are 5~8. The "normal" zones are 0~2 and 15~18. The "peak" zones are 11~13 and 20~23. For T_{IVM}, there are "off-peak" hours with $\lambda = 200$, "normal" hours with $\lambda = 300$, and "peak" hours with $\lambda = 500$. For T_{SVM}, there are "off-peak" hours with $\lambda = 250$, "normal" hours with $\lambda = 350$, and "peak" hours with $\lambda = 550$. For T_{TVM}, there are "off-peak" hours with $\lambda = 300$, "normal" hours with $\lambda = 350$, and "peak" hours with $\lambda = 400$.

For *response time*, we assume that a composite web service is composed of N_{IS} independent web services and $N_{\text{IS}} = 15$.

Table 4: Simulation parameters.

Parameter	Value
Simulation time	24 hours (mean of 300 days)
IVM_{CAP}	100 requests
SVM_{CAP}	50 requests
TVM_{CAP}	20 requests
IVM_{PRI}	\$0.8 per hour
SVM_{PRI}	\$0.8 per hour
TVM_{PRI}	\$0.9 per hour
ws_{init}	0.8 ms
ws_{fin}	0.7 ms
N_{IS}	15
L	600 frames
F	60 fps
α	1.7
P_{sf}	10 kWh
E_{ps}	12 kWh
PM_{size}	10

Also ws_{init} and ws_{fin} are the time of initial and final procedures in a composite web service. For *Startup Latency*, the number of frames L which a buffer of client player contains is 600 frames and the average video processing frame rate for SVM is $F = 60$ frames per second. For *transcoding efficiency*, the degradation coefficient of utilization $\alpha = 1.7$. For energy consumption, we assume that the per-user energy consumption for IVM and SVM service is $P_{sf} = 10$ kWh and the per-user energy consumption for TVM is $E_{ps} = 12$ kWh. Also all the physical machines possess equivalent size, $PM_{size} = 10$, for IVM, SVM, and TVM.

Based on the utility model of [27], the utility concept has been used in microeconomic theory. Here we modify it with our notations and develop it as utility-based pricing model. Let utilization and price be the measured utility function of the utility-based pricing model:

$$\begin{aligned} \max \quad & U = \sum \left(\text{util}_{IVM}, \text{util}_{SVM}, \text{util}_{TVM}\right) \\ \text{subject to} \quad & \min \text{PRI} \\ & = \sum \left(n_{IVM} IVM_{PRI} + n_{SVM} SVM_{PRI} + n_{TVM} TVM_{PRI}\right). \end{aligned} \tag{26}$$

We use this utility-based pricing model in comparison with our proposed goal programming approach in the simulation.

6.1. Utilization. The platform utilization of IVM, SVM, and TVM during the simulation hours is shown in Figure 7. The horizontal axis represents simulation time and the vertical axis stands for resource utilization of platform. Note that we set the following: constraint_$\text{util}_{IVM} \geqq 60\%$, constraint_$\text{util}_{SVM} \geqq 60\%$, and constraint_$\text{util}_{TVM} \geqq 60\%$.

For utility-based model, the slope of platform utilization of IVM, SVM, and TVM during the simulation hours is more gradual than GP. We can see that util_{IVM} of utility-based model is always greater than 80%, util_{SVM} of utility-based model is always greater than 90%, and util_{TVM} of utility-based model is always greater than 95%. However for GP, most of the time the curves of platform utilization of IVM and SVM are lower than utility-based model and they even drop obviously during the off-peak and the normal period. Note that util_{IVM} of GP and util_{SVM} of GP are lower than 60% during the off-peak period but higher than 60% during the normal and the peak period. Basically, utilization of utility-based model is better than GP, especially during the off-peak period. This is due to the fact that priority of QoS of Service Clients criteria is higher than Service Utility criteria in goal programming approach. While GP is worse than utility-based model in utilization, GP still achieves the goals of constraint_util_{IVM}, constraint_util_{SVM}, and constraint_util_{TVM} mostly.

6.2. Response Time. Figure 8 shows the response Time of IVM. The horizontal axis represents simulation time and the vertical axis stands for response time in millisecond. Note that we set the following constraints for GP: constraint_rt $\leqq$ 3.5 milliseconds.

Utility-based model has a steep slope compared with GP. And the response time of utility-based model rises to almost 6 and 4.5 milliseconds during normal and off-peak period while the response time of GP is around 3.5 milliseconds (slightly higher during off-peak and normal but lower during peak period) most of the time. Obviously the response time of GP is always better than utility-based model and achieves the goal of constraint_rt mostly. Compared with Figure 7, we can derive that, during off-peak and normal period, GP makes a trade-off between utilization of IVM and response time and the latter is prior to the former, whereas utility-based model always consider utilization only.

6.3. Startup Latency. Figure 9 shows the *Startup Latency* of SVM during the simulation hours. The horizontal axis represents simulation time and the vertical axis stands for startup latency in seconds. Note that we set the following constraints for GP: constraint_sl $\leqq$ 1 second.

Similarly, utility-based model has a steep slope compared with GP. The startup latency of utility-based model rises to almost 2 and higher than 1 seconds during normal and off-peak period while the startup latency of GP is around 0.9 milliseconds (again slightly higher during off-peak and normal but lower during peak period) most of the time. We can conclude that the startup latency of GP is always better than utility-based model and achieves the goal of constraint_sl mostly. And compared with Figure 7, we can conclude that, during off-peak and normal period, GP makes a trade-off between utilization of SVM and startup latency and the latter is prior to the former, whereas utility-based model always considers utilization only.

6.4. Transcoding Efficiency. The *transcoding efficiency* of TVM during the simulation hours is shown in Figure 10. The horizontal axis represents simulation time and the vertical axis stands for transcoding efficiency of TVM. Note that we set the following constraints for GP: constraint_te $\geqq 4$.

Both utility-based model and GP have gradual slopes. The transcoding efficiency of utility-based model is at almost 2.5

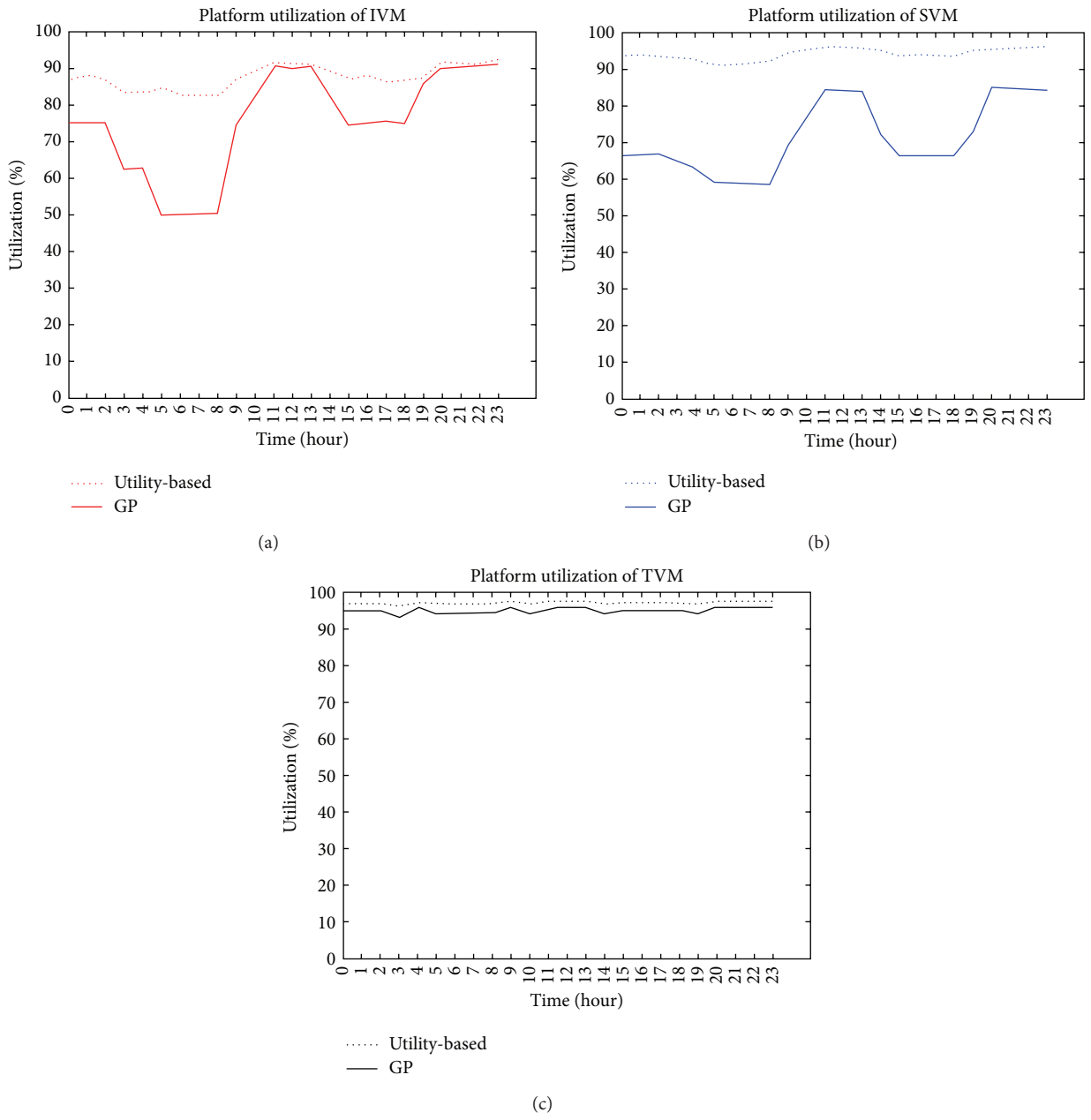

Figure 7: (a) Resource utilization of streaming service platform for IVM. (b) Resource utilization of streaming service platform for SVM. (c) Resource utilization of streaming service platform for TVM.

while the transcoding efficiency of GP is above 4. Clearly the transcoding efficiency of GP is always better than utility-based model and can achieve the goal of constraint_te all the time.

6.5. Energy Consumption and Profit Performance. Here we show the results of energy consumption and profit performance in Figures 11 and 12. The horizontal axis represents simulation time and the vertical axis stands for energy consumption in kWh and profit performance in $.

The energy consumption of utility-based model is always lower than GP which indicates that the amount of IVM, SVM, and TVM of utility-based model is less than GP and so is the number of the host physical machines. That is to say, GP always tends to allocate more resources in order to maintain *QoS of Service Clients*. On the other hand, more resources allocated of the service platform means more profit for the cloud infrastructure provider. Utility-based model is not considered what QoS level should be achieved or SLA should be followed. Thus, utility-based model just tries to reach the needs of physical machine. On the contrary, GP in order to conform Quality of Service and SLA for clients, it is necessary to acquire more resources to achieve the goal. Naturally, for the service provider, more resources required means more profit; for the client, GP provides a better and stable service experience.

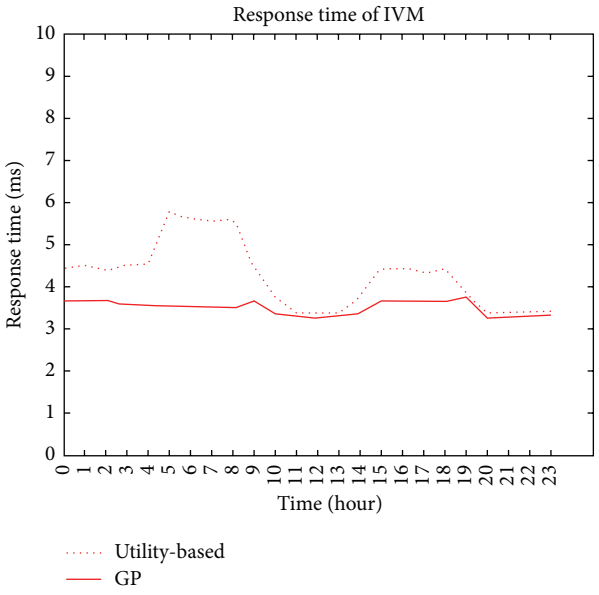

Figure 8: Response time of IVM.

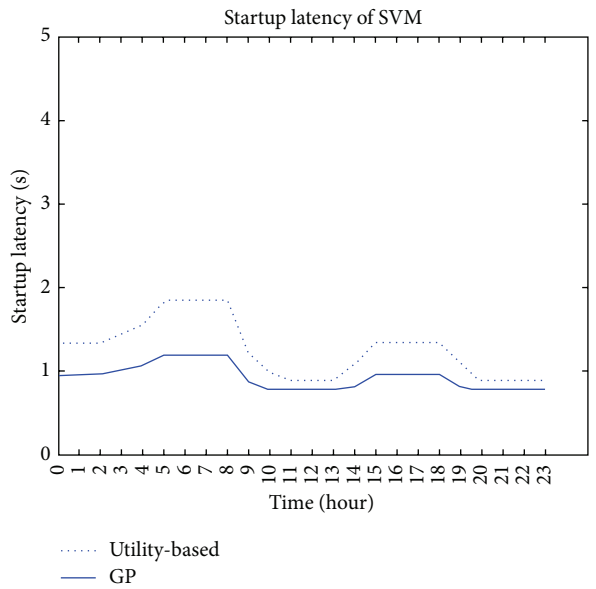

Figure 9: Startup latency of SVM.

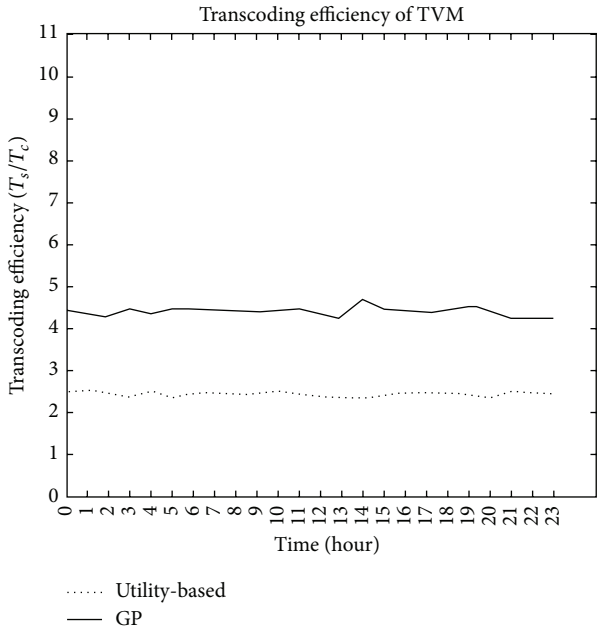

Figure 10: Transcoding efficiency of TVM. Energy consumption and profit performance.

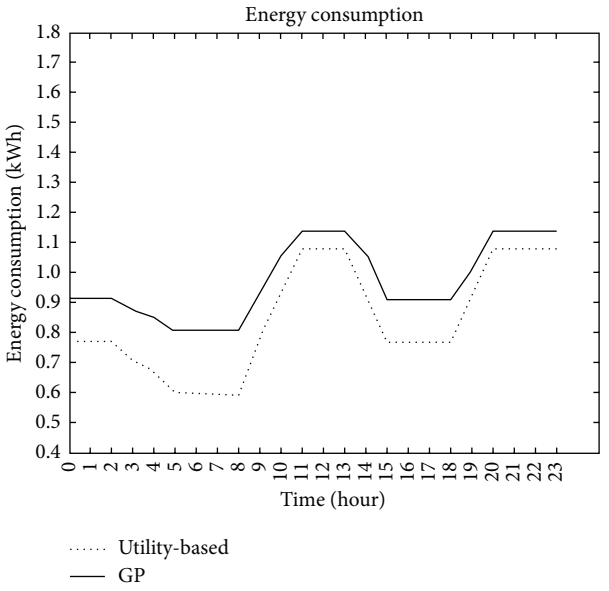

Figure 11: Energy consumption.

7. Conclusion

In this paper, we have addressed the problem of resource brokering between the streaming service platform and the cloud infrastructure provider considering both service-oriented and infrastructure-oriented criteria. An effective resource provisioning algorithm for an autonomous resource broker in the cloud is proposed and the core mechanism of this broker is obtained by formulating and solving a goal programming model. This resource broker manages the provisioning of cloud resources between the streaming service platform and the cloud infrastructure provider based on not only the number of incoming requests for the various services of the streaming platform but also both the service-oriented criteria which are in the form of SLA and infrastructure-oriented criteria. Also, a cloud architecture for the streaming service is proposed and extensive analysis is performed for related criteria. The simulation results show that the proposed approach makes an effective trade-off between these conflicting criteria commendably and achieves our goals of QoS.

For the future work, we hope that the approaches proposed in our work can further be practiced and applied to a real cloud environment, for example, being implemented as a

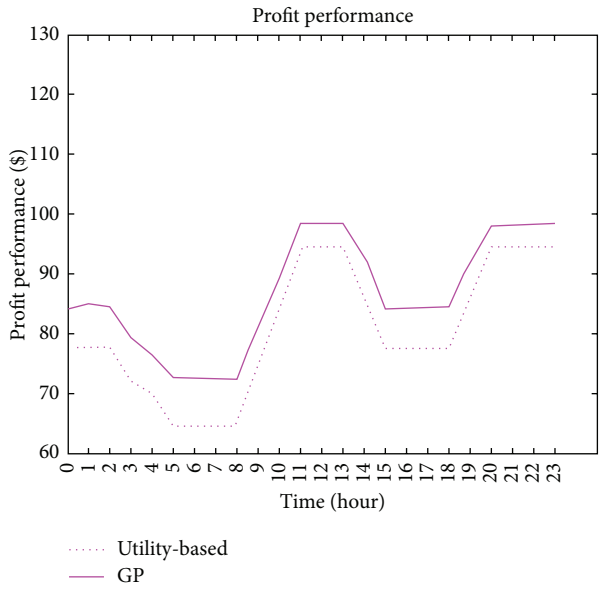

Figure 12: Profit performance.

decision making system used to make decision plans for efficiently provisioning resources considering multiple criteria for both cloud infrastructure providers and their customers. Also for resource brokering, there are still many performance issues that need further analysis in more complex deployment models like hybrid cloud.

Notations

P: Resource pool comprising a set of physical machines
PM_{size}: The number of VMs that a single PM can host
IVM: Interaction virtual machine
SVM: Streaming virtual machine
TVM: Transcoding virtual machine
n_{IVM}: The amount of IVM
n_{SVM}: The amount of SVM
n_{TVM}: The amount of TVM
IVM_{PRI}: The cost to run a single IVM for a duration
SVM_{PRI}: The cost to run a single SVM for a duration
TVM_{PRI}: The cost to run a single TVM for a duration
IVM_{CAP}: Maximum capacity of a single IVM for a duration
SVM_{CAP}: Maximum capacity of a single SVM for a duration
TVM_{CAP}: Maximum capacity of a single TVM for a duration
R: A set of platform clients' requests
t: The tasks which are ran by their specific VM hosts
T_{IVM}: A subset of R that consists of tasks for IVM
T_{SVM}: A subset of R that consists of tasks for SVM
T_{TVM}: A subset of R that consists of tasks for TVM
$util_{IVM}$: Average resource utilization of all IVMs
$util_{SVM}$: Average resource utilization of all SVMs
$util_{TVM}$: Average resource utilization of all TVMs
$Avail_{IVM}$: Average availability of IVM during a time period
$Avail_{SVM}$: Average availability of SVM during a time period
$Avail_{TVM}$: Average availability of TVM during a time period
rt: Response time of IVM services
sl: Startup latency of SVM services
te: Transcoding efficiency of TVM services
EC_{IVM}: Energy consumption of all IVMs
EC_{SVM}: Energy consumption of all SVMs
EC_{TVM}: Energy consumption of all TVMs
ec: Overall energy consumption of these VMs
pp: Profit performance evaluated by the allocated VMs.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This work was supported by the Ministry of Science and Technology under Grant MOST 103-2221-E-309-004.

References

[1] P. Mell and T. Grance, *The NIST Definition of Cloud Computing (Draft)*, NIST Special Publication 800-145, National Institute of Standards and Technology, 2011.

[2] P. Barham, B. Dragovic, K. Fraser et al., "Xen and the art of virtualization," in *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP '03)*, pp. 164–177, October 2003.

[3] Amazon EC2, http://aws.amazon.com/ec2/.

[4] Y. O. Yazir, C. Matthews, R. Farahbod et al., "Dynamic resource allocation in computing clouds using distributed Multiple Criteria Decision Analysis," in *Proceedings of the 3rd IEEE International Conference on Cloud Computing (CLOUD '10)*, pp. 91–98, July 2010.

[5] H. N. Van, F. D. Tran, and J.-M. Menaud, "Performance and power management for cloud infrastructures," in *Proceedings of the 3rd IEEE International Conference on Cloud Computing*, pp. 329–336, July 2010.

[6] Y. Zhang, G. Huang, X. Liu, and H. Mei, "Integrating resource consumption and allocation for infrastructure resources on-demand," in *Proceedings of the 3rd IEEE International Conference on Cloud Computing (CLOUD '10)*, pp. 75–82, July 2010.

[7] P. Patel, A. Ranabahu, and A. Sheth, "Service level agreement in cloud computing," in *Proceedings of the Workshop on Best Practices in Cloud Computing: Implementation and Operational*

Implications for the Cloud at ACM International Conference on Object-Oriented Programming, Systems, Languages, and Applications, October 2009.

[8] D. A. Menascé, "QoS issues in web services," *IEEE Internet Computing*, vol. 6, no. 6, pp. 72–75, 2002.

[9] D. A. Menascé, "Response-time analysis of composite web services," *IEEE Internet Computing*, vol. 8, no. 1, pp. 90–92, 2004.

[10] D. A. Menascé, "Composing web services: a QoS view," *IEEE Internet Computing*, vol. 8, no. 6, pp. 88–90, 2004.

[11] J. Gray and A. Reuter, *Transaction Processing: Concepts and Techniques*, Morgan Kaufmann Publishers, San Francisco, Calif, USA, 1993.

[12] C. Devlin, "SaaS Capacity Planning: Transaction Cost Analysis Revisited," 2008, http://msdn.microsoft.com/en-us/library/cc261632.aspx.

[13] H. Al-Hilali, D. Guimbellot, and M. Oslake, *Capacity Model for Internet Transactions*, 1999, http://research.microsoft.com/pubs/69700/tr-99-18.doc.

[14] S. S. Rao, H. M. Vin, and A. Tarafdar, "Comparative evaluation of server-push and client-pull architectures for multimedia servers," in *Proceedings of the 6th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '96)*, Zushi, Japan, April 1996.

[15] J. Y. B. Lee, "Parallel video servers: A tutorial," *IEEE Multimedia*, vol. 5, no. 2, pp. 20–28, 1998.

[16] J. Y. B. Lee and P. C. Wong, "Performance analysis of a pull-based parallel video server," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 12, pp. 1217–1231, 2000.

[17] J. Y. B. Lee, "Buffer management and dimensioning for a pull-based parallel video server," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 4, pp. 485–496, 2001.

[18] Z. Tian, J. Xue, W. Hu, T. Xu, and N. Zheng, "High performance cluster-based transcoder," in *Proceedings of the International Conference on Computer Application and System Modeling (ICCASM '10)*, pp. V248–V252, October 2010.

[19] Y. Sambe, S. Watanabe, D. Yu, T. Nakamura, and N. Wakamiya, "Distributed video transcoding and its application to grid delivery," in *Proceedings of the 9th Asia-Pacific Conference on Communications (APCC '03)*, vol. 1, pp. 98–102, Penang, Malaysia, 2003.

[20] K. Breitman, M. Endler, R. Pereira, and M. Azambuja, "When TV dies, will it go to the cloud?" *Computer*, vol. 43, no. 4, Article ID 5445174, pp. 81–83, 2010.

[21] J. Baliga, R. W. A. Ayre, K. Hinton, and R. S. Tucker, "Green cloud computing: Balancing energy in processing, storage, and transport," *Proceedings of the IEEE*, vol. 99, no. 1, pp. 149–167, 2010.

[22] J. P. Ignizio, "A review of goal programming: a tool for multiobjective analysis," *Journal of the Operational Research Society*, vol. 29, no. 11, pp. 1109–1119, 1978.

[23] M. Tamiz, D. Jones, and C. Romero, "Goal programming for decision making: an overview of the current state-of-the-art," *European Journal of Operational Research*, vol. 111, no. 3, pp. 569–581, 1998.

[24] D. F. Jones and M. Tamiz, "Expanding the flexibility of goal programming via preference modelling techniques," *Omega*, vol. 23, no. 1, pp. 41–48, 1995.

[25] H. P. Williams, *Model Building in Mathematical Programming*, John Wiley & Sons, New York, NY, USA, 1978.

[26] W. L. Winston, *Operations Research: Applications and Algorithms*, Duxbury Press, 3rd edition, 1997.

[27] J.-T. Park, J.-W. Baek, and J. W.-K. Hong, "Management of service level agreements for multimedia internet service using a utility model," *IEEE Communications Magazine*, vol. 39, no. 5, pp. 100–106, 2001.