This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2586048, IEEE Transactions on Cloud Computing

Stochastic Workload Scheduling for Uncoordinated Datacenter Clouds with Multiple QoS Constraints

Yunliang Chen, Lizhe Wang, *Senior Member, IEEE,* Xiaodao Chen, *Member, IEEE,* Rajiv Ranjan, Albert Y. Zomaya, *Fellow, IEEE,* Yuchen Zhou, and Shiyan Hu, *Senior Member, IEEE*

Abstract—Cloud computing becomes a well-adopted computing paradigm. With the unprecedented scalability and flexibility, the computational cloud is able to carry out large scale computing tasks in the parallel fashion. The datacenter cloud is a new cloud computing model that use multi-datacenter architectures for large scale massive data processing or computing.
In datacenter cloud computing, the overall efficiency of the cloud depends largely on the workload scheduler, which allocates clients' tasks to different Cloud datacenters. Developing high performance workload scheduling techniques in Cloud computing imposes a great challenge which has been extensively studied. Most previous works aim only at minimizing the completion time of all tasks. However, timeliness is not the only concern, while reliability and security are also very important. In this work, a comprehensive Quality of Service (QoS) model is proposed to measure the overall performance of datacenter clouds. An advanced Cross-Entropy based stochastic scheduling (CESS) algorithm is developed to optimize the accumulative QoS and sojourn time of all tasks. Experimental results show that our algorithm improves accumulative QoS and sojourn time by up to 56.1% and 25.4% compared to the baseline algorithm, respectively. The runtime of our algorithm grows only linearly with the number of Cloud datacenters and tasks. Given the same arrival rate and service rate ratio, our algorithm steadily generates scheduling solutions with satisfactory QoS without sacrificing sojourn time.

Index Terms—Cloud Computing, DataCenter Clouds, Quality of Service, Workload Scheduling

1 Introduction

Cloud computing [1], which delivers computing as a service, has emerged as a well-adopted computing paradigm which offers vast computing power and flexibility, and an increasing number of commercial cloud computing services are deployed into the market such as Amazon EC2 [2], Google Compute Engine [3], and Rackspace Cloud [4]. The new computing paradigms of "Cloud of Clouds" [5] and "datacenter clouds" [6], [7] are a creation of federated Cloud computing environment that coordinates distributed datacenter computing and achieves high QoS for Cloud applications. Large-scale data-intensive applications across distributed modern datacenter infrastructures is a good implementation and use case of the "Cloud of Clouds" paradigm. A good example for data-intensive analysis is the field of High Energy Physics (HEP). The four main detectors including ALICE, ATLAS, CMS and LHCb at the Large Hadron Collider (LHC) produced about 13 petabyes of data in 2010 [8]. This huge amount of data are stored on the Worldwide LHC Computing Grid that consists of more than 140 computing centers distributed across 34 countries. The central node of the Grid for data storage and first pass reconstruction, referred to as Tier 0, is housed at European Organization for Nuclear Research (CERN). Starting from this Tier, a second copy of the data is distributed to 11 Tier 1 sites for storage, further reconstruction and scheduled analysis.

Since the datacenter cloud computing paradigm offers massive computational resources, it provides enormous opportunities for software designers to architect their software in order to benefit from the massive parallelism. After a customer submits a computational job to a cloud, task scheduling will be performed to decide where, when and how this job can be executed. On the other hand, cloud computing features the high degree of the information heterogeneity which includes different processor speed, different processor location, different processor energy consumption, different job waiting time, different job runtime, different communication cost as well as other uncertainties. Among these, the security and reliability are highly important [9], [10]. These introduce significant technical difficulty in designing a high performance task scheduling framework. Therefore, it is necessary to have a comprehensive Quality of Service (QoS) metric to quantify the performance of a scheduler. Since the scheduler allocates computational tasks to heterogeneous computational resources for optimizing QoS, it is called a QoS aware task scheduler in cloud computing.

Our contributions are summarized as follows.

- *Yunliang Chen, Lizhe Wang and Xiaodao Chen are with the School of Computer Science, China University of Geosciences, Wuhan, 430074, P. R. China.*
- *Rajiv Ranjan is with School of Computing Science, Newcastle University, U.K.*
- *Albert Y. Zomaya is with the School of Information Technologies, The University of Sydney, Australia.*
- *Yuchen Zhou and Shiyan Hu are with the Department of Electrical and Computer Engineering, Michigan Technological University, Houghton, Michigan, 49931.*
- *Corresponding author: Lizhe Wang, Lizhe.Wang@computer.org*

2168-7161 (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2586048, IEEE Transactions on Cloud Computing

- A comprehensive QoS model for evaluating the overall performance of the datacenter Cloud is proposed. Our QoS model provides different metrics, measuring the Cloud computing performance from different angles. It guarantees satisfying performance of the datacenter Cloud in terms of not only timeliness, but also reliability and security.
- A QoS driven Cross-Entropy based stochastic scheduling (CESS) algorithm is developed to optimize the scheduling solution in terms of every metric defined in the QoS model.
- The CESS algorithm improves accumulative QoS and sojourn time by up to 56.1% and 25.4% compared to the baseline algorithm, respectively.
- The CESS algorithm runs efficiently. The runtime scales only linearly with the number of jobs and the number of Cloud datacenters. Therefore, it has the potential to be successfully deployed in the real world.
- Given the same arrival rate and service rate ratio, our CESS algorithm steadily generates scheduling solutions with satisfactory QoS without sacrificing sojourn time.

2 Backgrounds and Related Work

2.1 Cloud of Clouds

A computing Cloud [11], [12] is a set of network enabled services, providing scalable, QoS guaranteed, normally personalized, inexpensive computing infrastructures on demand, which can be accessed in a simple and pervasive way [13], [14], [15].

The paradigm of "Cloud of Clouds", or InterCloud [16], [17], [18], [19], [20], [21], is to leverage the global infrastructure based on multiple Clouds for large scale distributed applications. The multi-datacenter infrastructure [22], [23], a reference implementation of the "Cloud of Clouds" model, implements a global infrastructure across distributed datacenters, storage services or clusters for intercloud applications.

Current research on the "Cloud of Clouds" model includes the programming model & software architecture [24], security & storage service [25], and inter-cloud computing standards [26]. In our previous research we have implemented a programming model for the paradigm of "Cloud of Clouds" by developing the G-Hadoop system [6], [7], a software framework for MapReduce applications across distributed datacenters and clusters.

2.2 Datacenter Clouds

Datacenter Clouds typically refers to the software and hardware infrastructures that provides general-purpose high-performance computing capabilities [27]. Different from conventional distributed systems such as large scale computer clusters, a datacenter Cloud is composed of distributed computer centers or datacenters from multiple geographical locations across the world [28]. In general, datacenters in clouds can communicate with each other via high-speed network interface. With this distributed infrastructure, a single computational task can be carried out on multiple machines in the parallel fashion, with the efficiency significantly improved.

Datacenter clouds promise on-demand access to affordable large-scale resources in computing and storage (such as disks) without substantial upfront investment. Thus it is naturally suitable for processing big data, especially streaming data, via allowing data processing algorithms to run at the scale required for handling uncertain data volume, variety, and velocity.

However, to support a complicated, dynamically configurable big data ecosystem, we need to innovate and implement novel services and techniques for orchestrating cloud resource selection, deployment, monitoring, and QoS control [5], [29].

The paradigm of datacenter Cloud computing has the following features

- Resource Sharing A Virtual Organization (VO) refers to a dynamic set of individuals and/or organizations bounded by the same set of resource-sharing rules and conditions. Here the resource includes not only data represented in various formats, but also computational power and storage units. They are requested and shared by a wide range of computational tasks from clients in industry, as well as academia. It becomes a technical challenge to coordinate resource sharing among the dynamic virtual organizations [30].
- Site Autonomy Resources shared in datacenter Clouds are commonly owned and controlled by different individuals or organizations in different sites [31], [32], [33]. Administrators of each site decide which resource to share and how to share the resources. Therefore, clients of the datacenter Cloud may experience different scheduling policies and security mechanisms when using datacenter Clouds.
- Hierarchy and Uncoordinated Local Queue Management
  In each geographical site, there may be a local resource management system, e.g., PBS [34], [35] and Sun Grid Engine [36]. Cloud users cannot access the individual resources inside the sites. Cloud users submit tasks to the Global Resource Management System (GRMS). Subsequently, GRMS submits tasks to the Local Resource Management System (LRMS) [37], [38]. The LRMS schedules the tasks to the resources inside local resource system. GRMS and LRMS constitute hierarchical datacenter Cloud environments.
  The uncoordinated LRMS may lead a large variety of queueing policies and queue waiting time, which will make significant impact on Cloud data processing applications. For example, the statistical analysis [39] shows that the queue wait time of Cloud systems, such as World LHC Computing Grid (WLCG), is random and highly complicated to predict.
- Heterogeneity
  Datacenter Clouds a highly heterogeneous environment [40], [41]. Different sites may have different types of resources. Even the resources of the same type, located at different sites, may have different

2168-7161 (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2586048, IEEE Transactions on Cloud Computing

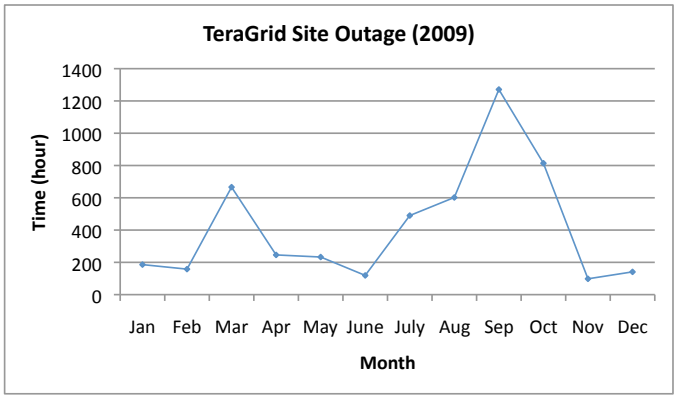


Fig. 1: Outage statistic of TeraGrid in 2009

configurations, capacities and performance.

- Large-Scale Distribution
  As discussed above, datacenter Cloud enables resource sharing among geographically distributed sites. These sites are connected via traditional network interface. Network communication delay may be extremely high when some communication intensive applications are running among these sites. In this scenario, network performance has an important effect on resource management [42], [43].
- Frequent Site Outrage
  The resources in Cloud datacenters could become unavailable during a site outage due to various reasons, such as power supply failure, scheduled maintenance, or hardware failure. As shown in Figure 1, during the year 2009, there were approximately 290 outages with total 5,000 hours on the TeraGrid infrastructures of datacenters and computer centers. On average, a single site experienced the outage time around 4% of the time duration of the year 2009. Apparently, site outages can seriously jeopardize the performance of the datacenter Clouds.

2.3 Multi-Cluster computing

The Multi-Cluster computing paradigm [44] employs multiple distributed clusters to build large computing infrastructure for HPC applications. There have been a considerable portion of research devoted to the multi-cluster scheduling, for example, scheduling of workflow applications [45], [46], [47] and scheduling of independent tasks [48], [49], [50] across the multi-cluster infrastructure. Compared with the aforementioned work, our research in this paper is devoted to scheduling of MapReduce jobs to multiple clusters, where we use a different task model and the data-centric scheduling heuristic.

The Gfarm file system [51] is a distributed file system designed to share vast amounts of data between globally distributed clusters connected via a wide-area network. Similar to HDFS the Gfarm file system leverages the local storage capacity available on compute nodes. In our work, we use Gfarm file system as a global distributed file system that supports the MapReduce framework.

In our G-Hadoop implementation, we use the Torque [52] as a cluster resource manager. The Distributed Resource Management Application API (DRMAA) [53] is a high-level API specification for the submission and control of jobs to one or more cluster resource managers. In G-Hadoop implementation, we use DRMAA as an interface for submitting tasks from G-Hadoop to the Torque Resource Manager.

2.4 Quality of Service (QoS)

Quality of Service (QoS) is a set metrics used to evaluate the overall performance of the datacenter Cloud , given the datacenter Cloud scheduling solution. Each metric evaluates the execution of a given job from one unique perspective. The performance is quantified by the fitness score between 0 and 1. A higher fitness score implies better execution quality. The overall performance of the datacenter Cloud on a specific job is determined by the summation of all fitness scores associated with the job. An effective Cloud computing scheduler optimizes the scheduling solution so that all QoS requirements are satisfied for each job. Our algorithm optimizes the scheduling using the following three QoS metrics.

- Timeliness. This metric defines the severity of missing the deadline of a job. The fitness score when missing the deadline is determined by the priority of the job. For example, for a job with a hard deadline, missing it would generate the fitness score of 0. For a job with no deadline, the fitness score is inversely proportional to the execution time.
- Reliability. Due to the autonomous nature of the datacenter Clouds, each Cloud datacenter has its own policy on its availability in the Cloud computing. In other words, some Cloud datacenters offer the computational power only in the idle state. Others offer all of the remaining resources even when internal tasks are under execution. The reliability of the Cloud datacenter is proportional to its availability. In other words, the Cloud datacenter offering stable computational power receives higher fitness score on reliability than the one disconnecting from the Cloud frequently.
- Security. Since there is no centralized administration over the Cloud, not every machine connected onto the Cloud is trustworthy. It is difficult to prevent malicious participants from compromising the Cloud by providing malfunctioning machines. The malfunctioning machines generate erroneous results and thus jeopardize the integrity of the datacenter Cloud .

2.5 QoS-aware Workload scheduling in datacenter Clouds

The traditional problem of workload scheduling for Cloud computing has been extensively studied. Since the problem is NP-Hard [27], many meta-heuristic algorithms have been proposed to query the optimal solution.

Classical resource management systems were designed for batch scheduling systems that rely on gang-scheduling model [54], which can allocate multiple resource types. The scheduling policies were dependent on job queues and priorities with the goal of keeping all resources busy rather

2168-7161 (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2586048, IEEE Transactions on Cloud Computing

than focusing on optimizing application level QoS. In the gang-scheduling model, jobs are queued and based on their priority they are assigned to cluster resources.

Datacenter management systems such as Amazon EC2, Microsoft Azure, and Eucalyptus, application administrators specify their resource requirements in terms of hardware (e.g. CPU type, CPU speed, number of cores, etc.) and software resource (virtualization format, operating system, etc.) configurations while ignoring specific QoS metrics such as response time, reliability, or security. Other systems such as YARN [55] , Apache Hadoop and Quincy [56] use a system centric fairness (e.g. CPU share or memory share) policy to map jobs to resources. These systems do not allow application administrator to specify and enforce application level QoS metrics and policies. Mesos [57] uses two-level scheduling to manage resources of a cluster that can be hosted within a public or private datacenter. Mesos does not support any scheduling policy, but is a framework that can support multiple policies. The approach proposed in this paper can be implemented as a scheduling policy in Mesos for ensuring application level QoS metrics.

A Chemical Reaction Optimization (CRO) is proposed [27]. The algorithm mimics the chemical reactions during which the potential energy of molecules is minimized. Each candidate scheduling solution is modeled as a molecule with certain potential energy. The potential energy is modeled as the overall quality of the scheduling solution. During each iteration, molecules are selected to perform chemical reactions with each other, generating new molecules with potentially lower potential energy, or solutions with better quality. The solution quality is evaluated by timeliness and reliability of the datacenter Cloud . The simulation results show that CRO based algorithm can generate better solutions than other meta-heuristics like Genetic Algorithm (GA) and Simulated Annealing (SA).

The other meta-heuristic Cloud computing scheduler combines Particle Swarm Optimization (PSO) and Gravitational Emulation Local Search (GELS) [28]. The fitness function of the PSO is inversely proportional to the completion time of the last executed job and the number of jobs missing their deadlines. During each iteration, every candidate scheduling solution is updated towards better fitness values. GELS is used to improve the candidate pool to avoid local optima. The experimental results show that it significantly reduces the completion time of the last job compared to other heuristics.

The common weakness of the previous works is that their scheduling solutions are optimized in terms of only one metric, timeliness. Since other metrics like security and reliability are not used in the optimization, those algorithms may suffer severely from sub-optimality. In this work, the comprehensive model of Quality of Service (QoS) is proposed for evaluating the performance of the datacenter Cloud . The QoS model generates more practical evaluation from various perspectives including timeliness, reliability and security.

3 MODELS

In this section, the system model, the workload model and the QoS model are first introduced. Based on them, the cloud datacenter model is established.

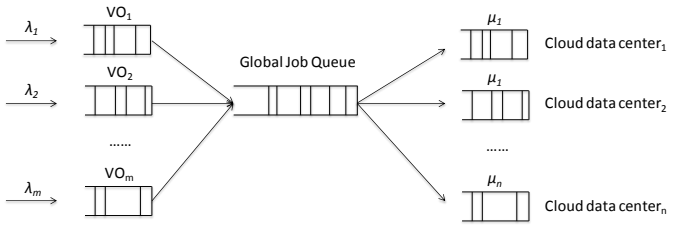


Fig. 2: System model

3.1 System model for Datacenter Cloud computing

The whole system is modeled as a M/M/1 system. The following assumptions about the system model.

- the incoming jobs are modeled as exponential distribution,
- the service rate of Cloud datacenters are exponential distributed,
- each Cloud datacenter is modeled as one server,
- the Cloud datacenter's service discipline is non-preemptive and First Come First Serve (FCFS).

3.2 Workload model

It is assumed that there are m Virtual Organizations (VOs) that share the datacenter Cloud system defined above. Each VO is modeled as a VO Job Queue (VOJQ). All jobs from VOJQs are submitted a Global Job Queue (GJQ). We define that:

- λ^i is the arrival rate of the VO_i, $1 \leq i \leq m$
- λ_j^i is the arrival rate of jobs of VO_i on the Cloud datacenter $Site_j$, $1 \leq i \leq m$, $1 \leq j \leq n$
- λ_j is the arrival rate of jobs from all VOs on the Cloud datacenter $Site_j$, $1 \leq j \leq n$, $\lambda_j = \sum_{i=1}^{m} \lambda_j^i$
- λ is the arrival rate of all VOs, $\lambda = \sum_{i=1}^{m} \lambda^i$

The job distribution possibility matrix is defined as follows:

$$P = [p_{ij}] = \begin{bmatrix} p_{11} & p_{12} & p_{1n} \\ p_{21} & p_{22} & p_{2n} \\ \vdots & \vdots & \vdots \\ p_{m1} & p_{m2} & p_{mn} \end{bmatrix} \quad (1)$$

p_{ij} is the possibility that a job from VO_i is scheduled to the Cloud datacenter $Site_j$, $1 \leq i \leq m, 1 \leq j \leq n$.

Therefore, the following is obtained.

$$\lambda_j^i = p_{ij} \times \lambda^i \quad (2)$$

$$\lambda_j = \sum_{i=1}^{m} (p_{ij} \times \lambda^i) \quad (3)$$

2168-7161 (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

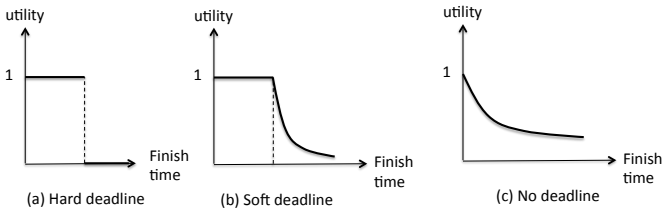


Fig. 3: Sample utility function: timeliness

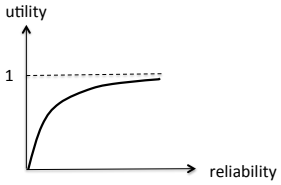


Fig. 4: Sample utility function: reliability

3.3 QoS model

Assume there are totally s QoS requirements, $Q = [Q_1, Q_2, \ldots, Q_s]$. Examples of QoS include availability, timeliness, security, and reliability, Therefore the QoS requirement matrix for all VOs is as follows:

$$Q = [q_{ik}] = \begin{bmatrix} q_{11} & q_{12} & q_{1s} \\ q_{21} & q_{22} & q_{2s} \\ \vdots & \vdots & \vdots \\ q_{m1} & q_{m2} & q_{ms} \end{bmatrix} \quad (4)$$

where, q_{ik} is a requirement of tasks in VO_i for QoS Q_k, $1 \leq i \leq m$, $1 \leq k \leq s$.

As QoS definition is associated a utility function [58], [59], which defines the benefit received by a VO. The utility function associated with q_{ik} is defined as:

$$q_{ik} : q_{ik} \rightarrow \mathbf{R} \quad (5)$$

where, $1 \leq i \leq m$, $1 \leq k \leq s$ and $\mathbf{R}$ is the set of positive real numbers. Examples of normalized utility functions are shown in Figure 3, 4 and 5. Figure 3 shows a task is associated with (a) a hard deadline, (b) a soft deadline and (c) no deadline. A task with the reliability QoS requirement is shown in Figure 4 and a task with the security QoS requirement is shown in Figure 5.

The above models are limited to express an individual QoS for a task, it is thus required to develop a further model that can express multiple QoS requirements for different VOs. The concept of weighted QoS achievement is proposed to denote the concept of QoS interests obtained for a job

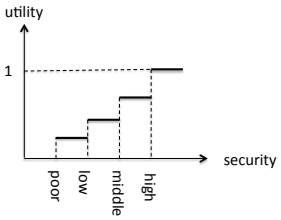


Fig. 5: Sample utility function: security

distribution of a VO on a Cloud datacenter. The weighted QoS achievement for VO_i is defined as following:

$$a_i = \sum_{k=1}^{s} w_{ik}(q_{ik}^{req} - q_{ik}^{get}) \quad (6)$$

where,

- $1 \leq i \leq m, 1 \leq k \leq s$
- w_{ik} denotes the weight of QoS Q_k for VO_i, $\sum_{k=1}^{s} w_{ik} = 1$
- q_{ik}^{get} denotes the QoS Q_k allocation for VO_i, and
- q_{ik}^{req} denotes the VO_i's requirement for QoS Q_k.

Given the job distribution possibility matrix defined in Equation 1, the q_{ik}^{get} can be calculated as follows:

$$q_{ik}^{get} = \sum_{j=1}^{n}(p_{ij} \times \tilde{q}_{jk}) \quad (7)$$

where $\tilde{q}_{jk}$ is the QoS allocation of Q_k from Site S_j.

The overall QoS achievement for all VOs is defined as follows:

$$A(P) = \sum_{i=1}^{m} a_i \quad (8)$$

Where is P is a job distribution possibility matrix and defined in Equation 1. It is thus an objective for a job distribution to maximize the overall QoS achievement for all VOs from a system perspective.

3.4 Cloud datacenter model

The service rate of a Cloud datacenter site is modeled as follows:

- μ_j is the service rate of the Cloud datacenter $Site_j$, $1 \leq j \leq n$,
- μ_j^i is the service rate of jobs of VO_i on the Cloud datacenter $Site_j$, $1 \leq j \leq n, 1 \leq i \leq m$.

The following part of this section models the unreliability of Cloud datacenters. The unreliable production Cloud datacenter is modeled with a set of successive periods of "up" and "down" as follows.

- η_j: the rate of up state of Cloud datacenter $Site_j$, and
- θ_j: the rate of Cloud datacenter down state of Cloud datacenter $Site_j$.

We define

- $E(S_j^i)$ is the sojourn time of a job from VO_i at a Cloud datacenter $Site_j$ and
- $E(L_j^i)$ is the queue length when the job from VO_i arrives at Cloud datacenter $Site_j$.

$E(S_j^i)$ is derived as follows. In case the job meets a reliable Cloud datacenter, the job's sojourn time is:

$$E_1 = \frac{E(L_j^i) + 1}{\mu_j^i} \quad (9)$$

However, the Cloud datacenter is unreliable, there exist extra waiting time due to down states of a Cloud datacenter.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2586048, IEEE Transactions on Cloud Computing

The mean number of down states experienced by the job is equal to $\eta_j \times \frac{E(L_j^i)+1}{\mu_j^i}$, and the mean duration of each down state is $\frac{1}{\theta_j}$. There for the extra waiting time due to down states of a Cloud datacenter is

$$E_2 = \eta_j \times \frac{E(L_j^i)+1}{\mu_j^i} \times \frac{1}{\theta_j} \tag{10}$$

Furthermore, the Cloud datacenter $Site_j$ is already down when the job comes, then there is another extra delay:

$$E_3 = \frac{1}{\theta_j} \times \frac{\eta_j}{\theta_j + \eta_j} \tag{11}$$

Therefore the sojourn time of a job from the i^{th} VO VO_i at a Cloud datacenter $Site_j$ is as follows:

$$\begin{aligned} E(S_j^i) &= E_1 + E_2 + E_3 \\ &= \frac{E(L_j^i)+1}{\mu_j} + \eta_j \times \frac{E(L_j^i)+1}{\mu_j} \times \frac{1}{\theta_j} \\ &\quad + \frac{1}{\theta_j} \times \frac{\eta_j}{\theta_j + \eta_j} \end{aligned} \tag{12}$$

Then, with Little's law:

$$E(L_j^i) = \lambda_j^i \times E(S_j^i) \tag{13}$$

the following is obtained:

$$E(S_j^i) = \frac{\frac{1}{\mu_j^i \times \rho_U} + \frac{\rho_D}{\theta_j}}{1 - \frac{\lambda_j^i}{\mu_j^i \times \rho_U}} \tag{14}$$

where,

- $\rho_D = \frac{\eta_j}{\eta_j + \theta_j}$: the fraction of down state of a Cloud datacenter.
- $\rho_U = \frac{\theta_j}{\eta_j + \theta_j}$: the fraction of up state of a Cloud datacenter.

Consequently,

$$E(S^i) = \sum_{j=1}^{n} (p_{ij} \times E(S_j^i)) \tag{15}$$

Finally, the mean sojourn time for all VOs is as follows:

$$E(S) = \sum_{i=1}^{m} (\frac{\lambda_i}{\lambda} \times E(S^i)) \tag{16}$$

It is thus an objective for a job distribution to minimize the mean sojourn time for all VOs from a system perspective.

4 Scheduling Algorithms

4.1 Research issue definition

It is assumed that there is a global scheduler that schedules workloads in the Cloud Job Queue to multiple Cloud datacenters (see also Figure 2). A global scheduler distributed incoming workloads from various VOs to multiple Cloud datacenters with the following objectives:

- minimize the mean sojourn time for all VOs $E(S)$, and
- minimize the QoS advantage $A(P)$.

Formally based on the job model and Cloud system model, the schedule function is defined as follows:

$$f : (VO, Site) \rightarrow P, \qquad f \in \mathbf{F} \tag{17}$$

where F is a set of all feasible schedule functions.

The research issue is defined as follows:

To find a schedule function $f \in \mathbf{F}$, which gives the $\min E(S)$ and $\min A(P)$.

4.2 Cross Entropy Theoretical Foundations

Cross entropy optimization, originally proposed in [60], is a stochastic optimization technique based on the theory of importance sampling. It casts a deterministic optimization problem into a stochastic optimization problem which can be solved to approximate the optimal solutions. This powerful optimization framework has been successfully applied to various different combinatorial optimizations problems such as those in [61], [62], [63], [64], [65], [66], [67]. For completeness, some details of this technique provided in [60], [68] are elaborated as follows.

To minimize a function $\min_{x \in \mathcal{D}} f(x)$ with variables x defined in the solution space $\mathcal{D}$, cross entropy firsts converts it into a stochastic optimization problem. That is, it uses a set of probability density functions (PDF) $g(x, p)$ defined in the space $\mathcal{D}$ to model the possible distributions on the solutions of the minimization problem. Given a set of random samples $X = \{X_1, X_2, \ldots, X_n\}$ generated according to $g(x, p)$, one can define $\delta(a)$ as

$$\delta(a) = P[f(X) \leq a], \tag{18}$$

where a is a parameter. Define an indicator function $I(\cdot)$ such that $f(x) \leq a$ if and only if $I_{f(x) \leq a} = 1$. Therefore, $P[f(X) \leq a] = E[I_{f(X) \leq a}]$ where E denotes the expectation. It is clear that if one can computes the largest a which makes $\delta(a)$ approach zero, this a gives a near optimal solution for the minimization problem $\min_{x \in \mathcal{D}} f(x)$. This is the basic idea of the conversion of a deterministic minimization problem to a stochastic minimization problem.

However, when $\delta(a)$ approaches zero, it is difficult to evaluate its value. If one uses the straightforward Monte Carlo simulations based technique, a large number of samples will be needed which is computationally expensive. That is, one can generate a set of samples according to $g(x, u)$ and an unbiased estimator is

$$\delta(a) = \frac{1}{n} \sum_{i=1}^{n} I_{f(X_i) \leq a} \tag{19}$$

As the solution approaches the optimal solution, $\delta(a)$ will approach zero, which means that a large number of samples are needed. In other words, $f(X) \leq a$ becomes a *rare event*. This is why cross entropy technique uses the importance sampling to tackle this technical difficulty.

In contrast to using $g(x, p)$, the importance sampling in the cross entropy technique uses a variant probability density function $k(x, p)$ also defined on $\mathcal{D}$. $\delta(a)$ can then be approximated by

$$\widehat{\delta}(a) = \frac{1}{n} \sum_{i=1}^{n} I_{f(X_i) \leq a} \frac{g(X_i)}{k(X_i)} \tag{20}$$

2168-7161 (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2586048, IEEE Transactions on Cloud Computing

Suppose that one can compute $k^*(x)$ such that $k^*(x) = \frac{I_{f(X_i)\leq a} g(x,u)}{\delta(a)}$. One has

$$\widehat{\delta}(a) = \frac{1}{n}\sum_{i=1}^{n} I_{f(X_i)\leq a} \frac{g(X_i)}{k^*(X_i)} = \delta(a) \quad (21)$$

The technical difficulty is that k^* cannot be computed explicitly. Therefore, the cross entropy technique uses a PDF which well approximates $k^*(x)$. This PDF has the property that it minimizes the so-called cross entropy between the two PDFs $k(x)$ and $g(x,v)$, which is

$$d(k,g) = E_g \ln \frac{k(X)}{g(X)} = \int k(x)\ln k(x)dx - \int k(x)\ln g(x)dx \quad (22)$$

Plugging this into the original functions, one has

$$argmax_v \int k^*(x)\ln g(x,v)dx \quad (23)$$

which is

$$argmax_v E_u I_{f(X)\leq a} \ln g(X,v) \quad (24)$$

The cross entropy technique uses importance sampling technique again with the new parameter w such that

$$argmax_v E_u I_{f(X)\leq a} \frac{f(x,u)}{f(x,w)} \ln g(X,v) \quad (25)$$

Subsequently, the solution to the original minimization problem can be written as

$$\widehat{v}^* = argmax_v \frac{1}{n}\sum_{i=1}^{n} I_{f(X_i)\leq a} \frac{f(X_i,u)}{f(X_i,w)} \ln g(X_i,v) \quad (26)$$

where the samples X are generated using $g(x,w)$. Refer to [60], [68] for the further details.

4.3 Cross Entropy Based Scheduling Algorithm

This work proposes a Cross Entropy based Scheduling Scheme (CESS) Algorithm to optimize the QoS and the waiting time. The CESS algorithm is iteratively proceeded to the solution by updating the probability density function (PDF) throughout the whole optimization procedure. This PDF is used to depict the candidate job assignments and employed to generate samples during each iteration. In this work, the Gaussian distribution is employed as the PDF function to solve the scheduling problem. Note that, the sample here denotes a scenario of the job assignment. On the other hand, the PDF are updated by elite samples in each iteration, where elite samples are job assignments which are high quality solutions in terms of the QoS and the waiting time.

Figure 6 shows the details of the CESS algorithm and Figure 7 shows an example of one iteration of the algorithm. The proposed algorithm first initialize the PDF array for all Cloud datacenters. Each Cloud datacenter is associated with a PDF over the its selection index (SI), an variable indicating the preference of our selection. A higher SI implies higher probability for the corresponding Cloud datacenter to be selected. If it is not the first iteration, the PDF array is inherited from the last iteration. Otherwise, each PDF is initialized with the same mean and variance, as indicated in Figure 7.

Subsequently, n samples are generated according to the PDF array. For each sample, a selection score is generated for each Cloud datacenter according to the corresponding PDF. The Cloud datacenter with the largest selection score

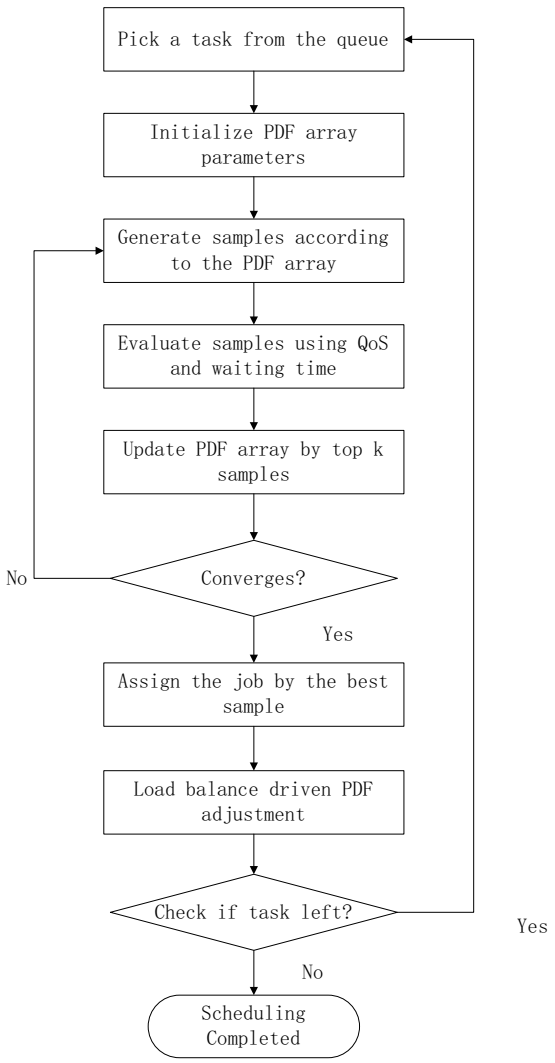


Fig. 6: Cross Entropy Based Scheduling Algorithm Flow

is the one selected for that sample. For example, for sample 1 in Figure 7, Cloud datacenter 1 has the largest score of 0.6, so it is selected in that sample. Similarly, Cloud datacenter 2 is selected in sample 2. For the case in which several Cloud datacenters share the same largest selection scores, the one with the largest mean on its PDF is selected. The reason is that, statistically, the one with the largest mean on the PDF performs the best.

After n samples are generated, each one is evaluated by QoS and sojourn time. k samples with the best QoS and Sojourn time form the set of elite samples. For the Cloud datacenter selected for elite samples, the mean is increased for the corresponding PDF. For each PDF, the variance is decreased. For example, as shown in Figure 7, suppose sample 1 and sample 2 are elite samples. Since Cloud datacenter 1 and 2 are the selected for the two samples, the PDFs are updated with larger mean. In this way, the Cloud datacenters with better QoS and sojourn time become more likely to be generated while the algorithm approaches convergence. If the algorithm goes through λ iterations or all samples selects the same Cloud datacenter, the convergence

2168-7161 (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2586048, IEEE Transactions on Cloud Computing

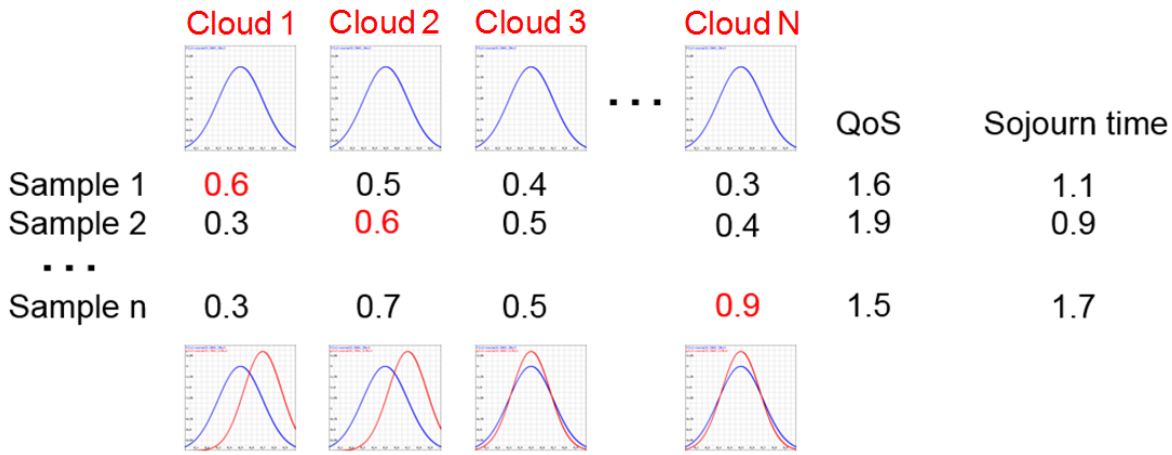


Fig. 7: Example of one iteration of CESS algorithm. The updated PDF for each is indicated by red curves.

criterion is met. After convergence, the job is assigned to the Cloud datacenter for the sample with the best QoS and sojourn time.

Straight forward implementation of the CESS algorithm would suffer from Cloud datacenter overloading issue. The reason is that the algorithm tends to assign every job to the Cloud datacenter with the best QoS. Consequently, the best Cloud datacenter becomes overloaded. To alleviate the issue, the load-balance driven PDF adjustment is proposed. That is, the mean value of the PDF of the Cloud datacenters selected is intentionally increased. As a result, the chance for a Cloud datacenter to be repeated selected is decreased and the loads are more evenly distributed over the Cloud datacenters.

5 Experimental Results

The proposed cross entropy based QoS-aware Workload scheduling with the Stochastic Modeling technique is implemented in C++ and tested on a machine with 2.8 GHz Intel® Core™ i5 CPU, 4 GB memory and 64 bit operating system. Due to inaccessibility to real world distributed datacenters, we construct a set of 500 synthetic test cases with up to 1000 VOs and 50 Cloud datacenters.

To demonstrate the superiority of our Cross Entropy based Scheduling Scheme (CESS) algorithm, we compare it with the baseline greedy algorithm. The baseline algorithm always greedily assigns the incoming jobs to the Cloud datacenter with the best Quality of Service (QoS) and least sojourn time. Note that, the QoS value includes the reliability and security values with weighted factors. The solutions to both algorithms on each test case is evaluated using the following metrics.

- Accumulative sojourn time of all jobs. Since our target is to minimize the average queuing time for all jobs, the scheduling quality is inversely proportional to this metric.
- Accumulative QoS fitness score of all jobs provided by all Cloud datacenters. Each QoS fitness score is the weighted sum of timeliness score, reliability score and security score. A higher QoS fitness score suggests faster execution time, higher reliability and better security. Apparently, the scheduling solution quality is proportional to this metric.

The comparison between the baseline greedy algorithm and our CESS algorithm is shown in Table 1. We have the following observations.

- In contrast to the baseline algorithm, our CESS algorithm generates better accumulative QoS on every test case. Statistically, the QoS is improved by 56.1% on average from the baseline algorithm. The reason is that our algorithm optimizes the scheduling of every job in terms of the QoS and the sojourn time.
- Comparing with the greedy algorithm, the proposed CESS algorithm can save up to 25.4% waiting time. On average, the waiting time is saved by 9.2%. The greedy algorithm tends to assign all jobs to the site with the best QoS. Apparently it overloads the Cloud datacenter, resulting in larger sojourn time. In contrast, the PDF performance-tuning in our algorithm mitigates the overburden of Cloud datacenters. It limits the probability that a particular Cloud datacenter is frequently selected, so that jobs are evenly distributed over the Cloud datacenters. Consequently, the accumulative sojourn time is decreased.
- The proposed algorithm are performed very efficiently. The results over all test cases can be within 864.55 seconds on average. Apparently, the runtime scales only linearly with test cases of different sizes.
- Although the waiting time of the greedy algorithm are quite close to the proposed algorithm, the QoS of the proposed algorithm dominates the greedy one.

To assess the performance of our algorithm in real world, we evaluate our CESS algorithm with different job arrival rate and Cloud service rate. The resulting sojourn time and QoS are shown in Figure 8. We have the following observations.

- Within the same period of time, the accumulative QoS and waiting time is proportional to the job arrival rate. As the job arrival rate increases, more jobs are handled by the Cloud datacenters. Since each job, handled by a Cloud datacenter, produces

2168-7161 (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2586048, IEEE Transactions on Cloud Computing

Fig. 8: QoS and waiting comparison with different job arrival rate and Cloud service rate

TABLE 1: Comparisions of QoS, waiting time and runtime among the greedy algorithm and the proposed CESS algorithm with varying sizes when tasks are assigned to a cluster system.

Testcase Size	Number of Clouds	Greedy Algorithm			CESS Algorithm			Improvement	
		QoS	Waiting Time	Runtime(ms)	QoS	Waiting Time	Runtime(s)	QoS	Waiting Time
50-100	10	47995.7	6748.6	1.38	64071.5	5037.7	30.11	33.5%	25.4%
101-200	20	125640.2	15984.0	4.32	180122.1	14214.51	178.37	43.4%	11.1%
201-400	30	252917.5	55384.2	9.10	378107.5	52555.5	567.75	49.5%	5.1%
401-600	40	409185.2	119858.9	14.86	628853.1	115267.9	1195.19	53.7%	3.8%
601-1000	50	630207.1	256373.7	25.81	983728.9	255296.8	2351.33	56.1%	0.0%
Average	-	293189.2	90869.88	11.09	446976.64	88474.49	864.55	47.2%	9.08%

a QoS value, the accumulative QoS increases. Larger waiting time can be explained by the gird site overloading effect. With the same Cloud service rate, increasing arrival rate requires Cloud datacenters to handle more jobs. Consequently, the accumulative waiting time increases.

- For the same amount of jobs with the same job arrival rate, both the accumulative QoS and the waiting time can be improved by increasing the Cloud service rate. Apparently the waiting time is inversely proportional to Cloud service rate. The PDF performance-tuning in our algorithm contributes to the improved QoS. With higher Cloud service rate, the Cloud datacenter overloading issue is mitigated. Since each Cloud datacenter is able to handle more jobs, the PDF performance-tuning intelligently assigns more jobs to sites with better QoS. As a result, the accumulative QoS is improved.
- By increasing both the job arrival rate and the Cloud service rate, both QoS and waiting time increase. Again, increment in accumulative QoS is due to the additional jobs, each contributing a QoS value to the accumulative QoS. In contrast to the case with higher job arrival rate and the same service rate, the waiting time does not increase dramatically with the job arrival rate. It suggests that the severity of Cloud datacenter overloading is significantly alleviated. Therefore, our algorithm has the capacity to be deployed in the real world, given steady service rate/job arrival rate ratio.

6 Conclusion

Cloud computing, which delivers computing as a service, has emerged as a promising computing paradigm which offers vast computing power and flexibility. However, it faces many challenges such as system modeling with variations and optimization scheduling issues. This work proposes a stochastic modeling of workload scheduling for the cloud computing environment considering timeliness, security and reliability. A cross entropy based QoS-aware workload scheduling technique is developed to compute scheduling solutions optimizing the QoS metric. Our experiments on 500 testcases demonstrate that the proposed approach significantly outperforms the greedy algorithm with up to 56.1% QoS improvement with largest size of testcases and 25.4% waiting time improvement with the testcases which has the size of $50 - 100$.

References

[1] C. Yang, C. Liu, X. Zhang, S. Nepal, and J. Chen, "A time efficient approach for detecting errors in big sensor data on cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 2, pp. 329–339, 2015. [Online]. Available: http://dx.doi.org/10.1109/TPDS.2013.2295810

[2] Amazon, "Amazon ec2," http://aws.amazon.com/ec2/, 2014, accessed: 2014-11-06.

2168-7161 (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2586048, IEEE Transactions on Cloud Computing

[3] Google, "Google compute engine," https://cloud.google.com/compute/, 2014, accessed: 2014-11-06.

[4] Rackspace, "Rackspace cloud," http://www.rackspace.com/cloud/, 2014, accessed: 2014-11-06.

[5] R. Ranjan, "Streaming big data processing in datacenter clouds," *IEEE Cloud Computing*, vol. 1, no. 1, pp. 78–83, 2014. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/MCC.2014.22

[6] J. Zhao, L. Wang, J. Tao, J. Chen, W. Sun, R. Ranjan, J. Kolodziej, A. Streit, and D. Georgakopoulos, "A security framework in g-hadoop for big data computing across distributed cloud data centres," *J. Comput. Syst. Sci.*, vol. 80, no. 5, pp. 994–1007, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.jcss.2014.02.006

[7] L. Wang, J. Tao, R. Ranjan, H. Marten, A. Streit, J. Chen, and D. Chen, "G-hadoop: Mapreduce across distributed data centers for data-intensive computing," *Future Generation Comp. Syst.*, vol. 29, no. 3, pp. 739–750, 2013. [Online]. Available: http://dx.doi.org/10.1016/j.future.2012.09.001

[8] G. Brumfiel, "High-energy physics: Down the petabyte highway," *Nature*, no. 7330, pp. 282–283, January 2011.

[9] C. Wang, Q. Wang, K. Ren, N. Cao, and W. Lou, "Toward secure and dependable storage services in cloud computing," *Services Computing, IEEE Transactions on*, vol. 5, no. 2, pp. 220–232, April 2012.

[10] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[11] P. Mell and T. Grance, "The NIST definition of cloud computing," *National Institute of Standards and Technology*, vol. 53, no. 6, p. 50, 2009. [Online]. Available: http://csrc.nist.gov/groups/SNS/cloud-computing/cloud-def-v15.doc

[12] X. Zhang, L. T. Yang, C. Liu, and J. Chen, "A scalable two-phase top-down specialization approach for data anonymization using mapreduce on cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 2, pp. 363–373, 2014. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/TPDS.2013.48

[13] X. Yao, H. Liu, H. Ning, L. T. Yang, and Y. Xiang, "Anonymous credential-based access control scheme for clouds," *IEEE Cloud Computing*, vol. 2, no. 4, pp. 34–43, 2015. [Online]. Available: http://dx.doi.org/10.1109/MCC.2015.79

[14] W. Chen, L. Xu, G. Li, and Y. Xiang, "A lightweight virtualization solution for android devices," *IEEE Trans. Computers*, vol. 64, no. 10, pp. 2741–2751, 2015. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/TC.2015.2389791

[15] X. Zhang, C. Liu, S. Nepal, S. Pandey, and J. Chen, "A privacy leakage upper bound constraint-based approach for cost-effective privacy preserving of intermediate data sets in cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1192–1202, 2013. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/TPDS.2012.238

[16] D. Bernstein, E. Ludvigson, K. Sankar, S. Diamond, and M. Morrow, "Blueprint for the intercloud - protocols and formats for cloud computing interoperability," *Internet and Web Applications and Services, International Conference on*, vol. 0, pp. 328–336, 2009.

[17] R. Buyya, R. Ranjan, and R. Calheiros, "Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services," in *Algorithms and Architectures for Parallel Processing*, ser. Lecture Notes in Computer Science, C.-H. Hsu, L. Yang, J. Park, and S.-S. Yeo, Eds. Springer Berlin / Heidelberg, 2010, vol. 6081, pp. 13–31, 10.1007/978-3-642-13119-6_2.

[18] T. Aoyama and H. Sakai, "Inter-cloud-computing," *WIRTSCHAFTSINFORMATIK*, vol. 53, pp. 171–175, 2011, 10.1007/s11576-011-0272-4.

[19] S. Sotiriadis, N. Bessis, F. Xhafa, and N. Antonopoulos, "From meta-computing to interoperable infrastructures: A review of meta-schedulers for hpc, grid and cloud," in *IEEE 26th International Conference on Advanced Information Networking and Applications*, 2012, pp. 874–883.

[20] N. Loutas, E. Kamateri, F. Bosi, and K. A. Tarabanis, "Cloud computing interoperability: The state of play," in *IEEE 3rd International Conference on Cloud Computing Technology and Science*, 2011, pp. 752–757.

[21] X. Wang, J. Cao, and Y. Xiang, "Dynamic cloud service selection using an adaptive learning mechanism in multi-cloud computing," *Journal of Systems and Software*, vol. 100, pp. 195–210, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.jss.2014.10.047

[22] M. Devarakonda, V. K. Naik, and N. Rajamanim, "Policy-based multi-datacenter resource management," in *6th IEEE International Workshop on Policies for Distributed Systems and Networks*, Jun. 2005.

[23] W. Song, S. Yue, L. Wang, W. Zhang, and D. Liu, "Task scheduling of massive spatial data processing across distributed data centers: What's new?" in *IEEE 17th International Conference on Parallel and Distributed Systems*, 2011, pp. 976–981.

[24] Y. Demchenko, "Defining intercloud architecture and cloud security infrastructure." Salt Lake City, USA: Presented at Cloud Federation Workshop, OGF32, July 2012. [Online]. Available: http://www.ogf.org/OGF32/materials/2314/ogf32-cloudfed-intercloud-security-v01.pdf

[25] A. Bessani, M. Correia, B. Quaresma, F. André, and P. Sousa, "Depsky: dependable and secure storage in a cloud-of-clouds," in *Proceedings of the sixth conference on Computer systems*, ser. EuroSys '11. New York, NY, USA: ACM, 2011, pp. 31–46. [Online]. Available: http://doi.acm.org/10.1145/1966445.1966449

[26] "P2302 - standard for intercloud interoperability and federation (siif)," [Online], http://standards.ieee.org/develop/project/2302.html.

[27] J. Xu, A. Y. Lam, and V. O. Li, "Chemical reaction optimization for task scheduling in grid computing," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 22, no. 10, pp. 1624–1631, 2011.

[28] Z. Pooranian, M. Shojafar, J. H. Abawajy, and A. Abraham, "An efficient meta-heuristic algorithm for grid computing," *Journal of Combinatorial Optimization*, pp. 1–22, 2013.

[29] R. Ranjan, "The cloud interoperability challenge," *IEEE Cloud Computing*, vol. 1, no. 2, pp. 20–24, 2014. [Online]. Available: http://dx.doi.org/10.1109/MCC.2014.41

[30] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations," *Int. J. High Perform. Comput. Appl.*, vol. 15, pp. 200–222, August 2001. [Online]. Available: http://portal.acm.org/citation.cfm?id=1080644.1080667

[31] M. J. Lewis, A. J. Ferrari, M. A. Humphrey, J. F. Karpovich, M. M. Morgan, A. Natrajan, A. Nguyen-Tuong, G. S. Wasson, and A. S. Grimshaw, "Support for extensibility and site autonomy in the legion grid system object model," *J. Parallel Distrib. Comput.*, vol. 63, pp. 525–538, May 2003. [Online]. Available: http://portal.acm.org/citation.cfm?id=876705.876708

[32] L. Wang, J. Chen, and W. Jie, *Quantitative Quality of Service for Grid Computing: Applications for Heterogeneity, Large-scale Distribution, and Dynamic Environments*. Hershey, PA: Information Science Reference – Imprint of: IGI Publishing, 2009.

[33] L. Wang, W. Jie, and J. Chen, *Grid Computing: Infrastructure, Service, and Applications*, 1st ed. Boca Raton, FL, USA: CRC Press, Inc., 2009.

[34] B. Nitzberg, J. M. Schopf, and J. P. Jones, "Grid resource management," J. Nabrzyski, J. M. Schopf, and J. Weglarz, Eds. Norwell, MA, USA: Kluwer Academic Publishers, 2004, ch. PBS Pro: Grid computing and scheduling attributes, pp. 183–190. [Online]. Available: http://portal.acm.org/citation.cfm?id=976113.976127

[35] J. P. Jones, "Beowulf cluster computing with linux." Cambridge, MA, USA: MIT Press, 2002, ch. PBS: portable batch system, pp. 369–390. [Online]. Available: http://portal.acm.org/citation.cfm?id=509876.509895

[36] J. Stosser, P. Bodenbenner, S. See, and D. Neumann, "A discriminatory pay-as-bid mechanism for efficient scheduling in the sun n1 grid engine," in *Proceedings of the Proceedings of the 41st Annual Hawaii International Conference on System Sciences*, ser. HICSS'08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 382–. [Online]. Available: http://dx.doi.org/10.1109/HICSS.2008.17

[37] P. Kokkinos and E. A. Varvarigos, "Resource information aggregation in hierarchical grid networks," in *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, ser. CCGRID'09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 268–275. [Online]. Available: http://dx.doi.org/10.1109/CCGRID.2009.63

[38] P. Cremonesi and R. Turrin, "Performance models for hierarchical grid architectures," in *Proceedings of the 7th IEEE/ACM International Conference on Grid Computing*, ser. GRID'06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 278–285. [Online]. Available: http://dx.doi.org/10.1109/ICGRID.2006.311026

2168-7161 (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2586048, IEEE Transactions on Cloud Computing

[39] A. Iosup, H. Li, M. Jan, S. Anoep, C. Dumitrescu, L. Wolters, and D. H. J. Epema, "The grid workloads archive," *Future Gener. Comput. Syst.*, vol. 24, pp. 672–686, July 2008. [Online]. Available: http://portal.acm.org/citation.cfm?id=1377055.1377376

[40] K. Lu, R. Subrata, and A. Y. Zomaya, "On the performance-driven load distribution for heterogeneous computational grids," *J. Comput. Syst. Sci.*, vol. 73, pp. 1191–1206, December 2007. [Online]. Available: http://portal.acm.org/citation.cfm?id=1296332.1296457

[41] P. Lindner, E. Gabriel, and M. M. Resch, "Gcm: a grid configuration manager for heterogeneous grid environments," *Int. J. Grid Util. Comput.*, vol. 1, pp. 4–12, May 2005. [Online]. Available: http://portal.acm.org/citation.cfm?id=1359318.1359319

[42] W. Jie, W. Cai, L. Wang, and R. Procter, "A secure information service for monitoring large scale grids," *Parallel Comput.*, vol. 33, pp. 572–591, August 2007. [Online]. Available: http://portal.acm.org/citation.cfm?id=1279016.1279273

[43] S. Viswanathan, B. Veeravalli, D. Yu, and T. G. Robertazzi, "Design and analysis of a dynamic scheduling strategy with resource estimation for large-scale grid systems," in *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, ser. GRID'04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 163–170. [Online]. Available: http://dx.doi.org/10.1109/GRID.2004.19

[44] M. E. Barreto, R. B. Ávila, and P. O. A. Navaux, "The multicluster model to the integrated use of multiple workstation clusters," in *Parallel and Distributed Processing*, ser. Lecture Notes in Computer Science, J. Rolim, Ed. Springer Berlin / Heidelberg, 2000, vol. 1800, pp. 71–80. [Online]. Available: http://dx.doi.org/10.1007/3-540-45591-4_8

[45] Z.-F. Yu and W.-S. Shi, "Queue waiting time aware dynamic workflow scheduling in multicluster environments," *J. Comput. Sci. Technol.*, vol. 25, no. 4, pp. 864–873, 2010.

[46] O. O. Sonmez, N. Yigitbasi, S. Abrishami, A. Iosup, and D. H. J. Epema, "Performance analysis of dynamic workflow scheduling in multicluster grids," in *HPDC*, 2010, pp. 49–60.

[47] L. He, S. A. Jarvis, D. P. Spooner, and G. R. Nudd, "Performance evaluation of scheduling applications with dag topologies on multiclusters with independent local schedulers," in *IPDPS*, 2006.

[48] H. Blanco, J. L. Lérida, F. Cores, and F. Guirado, "Multiple job co-allocation strategy for heterogeneous multi-cluster systems based on linear programming," *The Journal of Supercomputing*, vol. 58, no. 3, pp. 394–402, 2011.

[49] L. He, S. A. Jarvis, D. P. Spooner, H. Jiang, D. N. Dillenberger, and G. R. Nudd, "Allocating non-real-time and soft real-time jobs in multiclusters," *IEEE Trans. Parallel Distrib. Syst.*, vol. 17, no. 2, pp. 99–112, 2006.

[50] L. He, S. A. Jarvis, D. P. Spooner, and G. R. Nudd, "Optimising static workload allocation in multiclusters," in *IPDPS*, 2004.

[51] O. Tatebe, K. Hiraga, and N. Soda, "Gfarm grid file system," *New Generation Comput.*, vol. 28, no. 3, pp. 257–275, 2010.

[52] "Torque resource manager," Website, http://www.clusterresources.com/products/torque-resource-manager.php.

[53] "Distributed Resource Management Application API (DRMAA)," Website, http://drmaa.org/.

[54] D. G. Feitelson and L. Rudolph, "Gang scheduling performance benefits for fine-grain synchronization," *Journal of Parallel and Distributed Computing*, vol. 16, pp. 306–318, 1992.

[55] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, B. Saha, C. Curino, O. O'Malley, S. Radia, B. Reed, and E. Baldeschwieler, "Apache hadoop yarn: Yet another resource negotiator," in *Proceedings of the 4th Annual Symposium on Cloud Computing*, ser. SOCC '13. New York, NY, USA: ACM, 2013, pp. 5:1–5:16. [Online]. Available: http://doi.acm.org/10.1145/2523616.2523633

[56] M. Isard, V. Prabhakaran, J. Currey, U. Wieder, K. Talwar, and A. Goldberg, "Quincy: Fair scheduling for distributed computing clusters," in *Proceedings of the ACM SIGOPS 22Nd Symposium on Operating Systems Principles*, ser. SOSP '09. New York, NY, USA: ACM, 2009, pp. 261–276. [Online]. Available: http://doi.acm.org/10.1145/1629575.1629601

[57] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. Katz, S. Shenker, and I. Stoica, "Mesos: A platform for fine-grained resource sharing in the data center," in *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'11. Berkeley, CA, USA: USENIX Association, 2011, pp. 295–308. [Online]. Available: http://dl.acm.org/citation.cfm?id=1972457.1972488

[58] A. Dogan and F. Özgüner, "On qos-based scheduling of a meta-task with multiple qos demands in heterogeneous computing," in *Proceedings of the 16th International Parallel and Distributed Processing Symposium*, ser. IPDPS'02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 227–. [Online]. Available: http://portal.acm.org/citation.cfm?id=645610.661882

[59] C. Lee, J. Lehoczky, D. Siewiorek, R. Rajkumar, and J. Hansen, "A scalable solution to the multi-resource qos problem," in *Proceedings of the 20th IEEE Real-Time Systems Symposium*, ser. RTSS'99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 315–. [Online]. Available: http://portal.acm.org/citation.cfm?id=827271.829073

[60] R. Y. Rubinstein and D. P. Kroese, *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer, 2004.

[61] G. Alon, D. P. Kroese, T. Raviv, and R. Y. Rubinstein, "Application of the cross-entropy method to the buffer allocation problem in a simulation-based environment," *Annals of Operations Research*, vol. 134, no. 1, pp. 137–151, 2005.

[62] S. Asmussen, D. P. Kroese, and R. Y. Rubinstein, "Heavy tails, importance sampling and cross–entropy," *Stochastic Models*, vol. 21, no. 1, pp. 57–76, 2005.

[63] K. Chepuri and T. Homem-de Mello, "Solving the vehicle routing problem with stochastic demands using the cross-entropy method," *Annals of Operations Research*, vol. 134, no. 1, pp. 153–181, 2005.

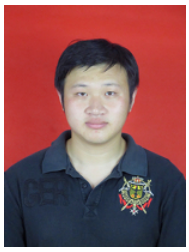
[64] I. Cohen, B. Golany, and A. Shtub, "Managing stochastic, finite capacity, multi-project systems through the cross-entropy methodology," *Annals of Operations Research*, vol. 134, no. 1, pp. 183–199, 2005.

[65] K.-P. Hui, N. Bean, M. Kraetzl, and D. P. Kroese, "The cross-entropy method for network reliability estimation," *Annals of Operations Research*, vol. 134, no. 1, pp. 101–118, 2005.

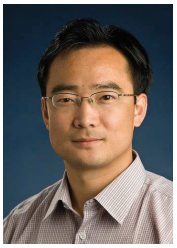
[66] A. Ridder, "Importance sampling simulations of markovian reliability systems using cross-entropy," *Annals of Operations Research*, vol. 134, no. 1, pp. 119–136, 2005.

[67] M. Yi-de, L. Qing, and Q. Zhi-bai, "Automated image segmentation using improved pcnn model based on cross-entropy," in *Intelligent Multimedia, Video and Speech Processing, 2004. Proceedings of 2004 International Symposium on*. IEEE, 2004, pp. 743–746.

[68] P.-T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, "A tutorial on the cross-entropy method," *Annals of operations research*, vol. 134, no. 1, pp. 19–67, 2005.



Yunliang Chen received the B. Sc. and M. Eng. degree from China University of Geosciences, and the Ph.D. degree from Huazhong University of Science and Technology, China. He currently is an Associate Professor with the School of Computer Science, China University of Geosciences, Wuhan, China. His research interests included computer network engineering, Cloud Computing, and IoT.



Lizhe Wang (SM' 2009) received the B.Eng. degree (with honors) and the M.Eng. degree both from Tsinghua University, Beijing, China, and the Doctor of Engineering in applied computer science (magna cum laude) from University Karlsruhe (now Karlsruhe Institute of Technology), Karlsruhe, Germany.

He is a Professor at Institute of Remote Sensing & Digital Earth, Chinese Academy of Sciences (CAS), Beijing, China and a "ChuTian" Chair Professor at School of Computer Science, China University of Geosciences, Wuhan, China.

Prof. Wang is a Fellow of IET and Fellow of BCS.

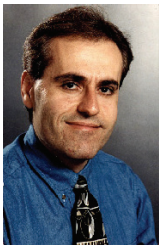
2168-7161 (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2586048, IEEE Transactions on Cloud Computing

Xiaodao Chen received the B.Eng. degree in telecommunication from Wuhan University of Technology, Wuhan, China, in 2006, the M.Sc. degree in electrical engineering from Michigan Technological University, Houghton, USA, in 2009, and the Ph.D. in computer engineering from Michigan Technological University, Houghton, USA, in 2012.

He is currently an Associate Professor with School of Computer Science, China University of Geosciences, Wuhan, China.

Rajiv Ranjan is an associated professor at School of Computing, Newcastle University. Prior this position, Rajiv Ranjan was a Research Scientist and a Julius Fellow in CSIRO Computational Informatics Division (formerly known as CSIRO ICT Centre). His expertise is in datacenter cloud computing, application provisioning, and performance optimization. He has a PhD (2009) in Engineering from the University of Melbourne. He has published 62 scientific, peer-reviewed papers (7 books, 25 journals, 25 conferences, and 5 book chapters). His hindex is 20, with a lifetime citation count of 1660+ (Google Scholar). His papers have also received 140+ ISI citations. 70% of his journal papers and 60% of conference papers have been A*/A ranked ERA publication. Dr. Ranjan has been invited to serve as the Guest Editor for leading distributed systems journals including IEEE Transactions on Cloud Computing, Future Generation Computing Systems, and Software Practice and Experience. One of his papers was in 2011's top computer science journal, IEEE Communication Surveys and Tutorials.



Albert Zomaya (F 2004)

Albert Y. Zomaya is currently the Chair Professor of High Performance Computing & Networking and Australian Research Council Professorial Fellow in the School of Information Technologies, The University of Sydney. He is also the Director of the Centre for Distributed and High Performance Computing which was established in late 2009. Professor Zomaya held the CISCO Systems Chair Professor of Internetworking during the period 2002-2007 and also was Head of school for 2006-2007 in the same school. Prior to his current appointment he was a Full Professor in the School of Electrical, Electronic and Computer Engineering at the University of Western Australia, where he also led the Parallel Computing Research Laboratory during the period 1990-2002. He served as Associate-, Deputy-, and Acting-Head in the same department, and held numerous visiting positions and has extensive industry involvement. Professor Zomaya received his PhD from the Department of Automatic Control and Systems Engineering, Sheffield University in the United Kingdom.

Yuchen Zhou received the B.S. degree in microelectronics from Hefei University of Technology, Hefei, China, in 2010. He is currently pursuing the Ph.D. in computer engineering at the Department of Electrical and Computer Engineering, Michigan Technological University, Houghton, MI, USA.

His research interests are in the area of grid computing and computer-aided design of VLSI circuits.

Shiyan Hu (SM' 2010) received the Ph.D. degree in computer engineering from Texas A&M University, College Station, in 2008.

He is currently an Associate Professor with the Department of Electrical and Computer Engineering, Michigan Technological University, Houghton, where he serves as the Director of the Michigan Tech VLSI CAD Research Laboratory. He was a Visiting Professor with the IBM Austin Research Laboratory, Austin, TX, in 2010. He has over 50 journal and conference publications. His current research interests include computer-aided design for very large-scale integrated circuits on nanoscale interconnect optimization, low power optimization, and design for manufacturability.

Dr. Hu has served as a technical program committee member for a few conferences such as ICCAD, ISPD, ISQED, ISVLSI, and ISCAS. He received the Best Paper Award Nomination from ICCAD 2009.

2168-7161 (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.