www.ijecs.in
International Journal Of Engineering And Computer Science ISSN: 2319-7242
Volume 4 Issue 8 Aug 2015, Page No. 13746-13753

Transformation-based Optimizations Framework (ToF) for Workflows and its Security issues in the Cloud Computing

S.Neelakandan[1], S.Muthukumaran[2]
[1,2] Assistant Professor
Department of Computer Science and Engineering
Jeppiaar Institute of Technology,Sriperumbudur
[1] snksnk17@gmail.com, [2] smkumaran83@gmail.com

***Abstract*—A general transformation-based optimization framework for workflows in the cloud. Specifically, ToF formulates six basic workflow transformation operations. An arbitrary performance and cost optimization process can be represented as a transformation plan (i.e., a sequence of basic transformation operations). All transformations form a huge optimization space. We further develop a cost model guided planner to efficiently find the optimized transformation for a predefined goal and our experimental results demonstrate the effectiveness of ToF in optimizing the performance and cost in comparison with other existing approaches. cloud computing offers its customers an economical and convenient pay-as-you- go service model, known also as usage-based pricing. Cloud customers pay only for the actual use of computing resources, storage, and band-width, according to their changing needs, utilizing the cloud's scalable and elastic computational capabilities. In particular, data transfer costs (i.e., bandwidth) is an important issue when trying to minimize costs**

Index Terms—Cloud computing, monetary cost optimizations, workflows.

1. INTRODUCTION

Cloud Computing is a flexible, cost- effective, and proven delivery platform for providing business or consumer IT services over the Internet. However, cloud Computing presents an added level of

risk because essential services are often outsourced to a third party, which makes it harder to maintain data security and privacy, support data and service availability, and demonstrate compliance. Cloud Computing has become more like an onset trend in the face of enterprises. It sets a milestone in the field of workflow execution in business process management. Workflow Management System is mainly devoted to support the definition as well as execution cum control of business processes. Workflow Scheduling is a key to workflow management. For efficient Scheduling in workflows, cost-time based evaluation of various algorithms has been included in this paper. First, users have different requirements on performance and cost. Some existing studies [5], [6] have focused on minimizing the cost while satisfying the performance requirement, some are aiming to optimize performance for a given budget [11] and others are considering the trade-off between performance and monetary cost [8], [12], [9]. Second, different cloud offerings result in significantly different cost structures for running the workflow. Even from the same cloud provider, there are multiple types of virtual machines (or instances) with different prices and computing capabilities. Third, workflows have very complicated structures and differ-ent computation/IO characteristics, as observed in the previous. We review the existing studies and find that most of them are *ad hoc* in the sense that they fail to capture the optimization opportunities in different user requirements, cloud offerings and workflows. For example, Kllapi et al. [8] consider only a single instance type. However, previous studies [14], [15] have shown that carefully selecting instance types is important forthe overall cost. Mao et al. [5] focus on minimizing the cost while satisfying the performance requirement of individual workflows, and simply use a *fixed* sequence of workflow transformation operations. The fixed sequence can be effective for some of the workflows and cloud offerings, but ineffective for others. All those studies potentially lose optimization opportunities for performance and cost. We have identified three design

principles. First, the framework should have an extensible design on the workflow optimizations, which adapts to different cloud offerings and workflow structures. Second, the framework should have a general optimization process for different requirements on performance and cost constraints. Last, the framework should be light weight for on-line decision making.

With these three design principles in mind, we propose *ToF*, a transformation-based optimization framework for optimiz-ing the performance and cost of workflows in the cloud. A workflow is generally modeled as a Directed Acyclic Graph (DAG) of tasks. ToF guides the scheduling of each task in the workflow, including which instance to assign to and when to start execution. When a workflow is submitted to the cloud, each task is initially assigned to a certain type of instance for execution. Based on the initial instance assignment, we perform transformations on the DAG. We categorize the transformation operations into two kinds, namely *main schemes* and *auxiliary schemes*. The main schemes reduce monetary cost while auxiliary schemes transform workflows into a DAG that is suitable for main schemes to perform cost reduction. Specifically, we have formulated six basic workflow transformation operations (*Merge*, *Demote*, *Split*, *Promote*, *Move* and *Co- scheduling*). The former two are categorized as main schemes and the latter four are auxiliary schemes. Although there are many benefits to adopting Cloud Computing, there are also some significant barriers to adoption. One of the most significant barriers to adoption is security, followed by issues regarding compliance, privacy and legal matters [8]. Because Cloud Computing represents a relatively new computing model, there is a great deal of uncertainty about how security at all levels (e.g., network, host, application, and data levels) can be achieved and how applications security is moved to Cloud Computing [9]. That uncertainty has consistently led information executives to state that security is their number one concern with Cloud Computing [10].

Security concerns relate to risk areas such as externaldata storage, dependency on the ―public‖ internet, lack of control, multi-tenancy and integration with internal security. Compared to traditional technologies, the cloud has many specific features, such as its large scale and the fact that resources belonging to cloud providers are completely distributed, heterogeneous and totally virtualized. Traditional security mechanisms such as identity, authentication, and authorization are no longer enough for clouds in their current form [11]. This paper makes the following two key contributions. First, we propose a transformation-based workflow optimization system to address the performance and monetary cost optimizations in the cloud. Second, we develop and deploy the workflow optimization system in real cloud environments, and demonstrate its effectiveness and efficiency with extensive experiments. To the best of our knowledge, this work is the first of its kind in developing a general optimization engine for minimizing the monetary cost of running workflows in the cloud.

The rest of this paper is organized as follows. We introduce the background on cloud offerings and application scenarios, and review the related work in Section 2. Section 3 gives a system overview. We present our workflow transformation operations in Section 4, followed by the cost model guided planner in Section 5. We present the experimental results in Section 6 and conclude the paper in Section 7.

2. PRELIMINARY AND RELATED WORK

In this section, we first describe cloud computing environments mainly from users' perspective. Next, we present the security issues and review the related work.

2.1 Cloud Environments

Cloud providers offer multiple types of instances with different capabilities such as CPU speed, RAM size, I/O speed and network bandwidth to satisfy different application demands. Different instance types are charged with different prices. Tables 1 and 2 show the prices and capabilities of four on-demand instance types offered by Amazon EC2 and Rack space, respectively. Amazon EC2 mainly charges according to the CPU, whereas Rack space mainly on the RAM size. Both cloud providers adopt the instance hour billing model, whereby partial instance hour usage is rounded up to one hour. Each instance has a non-ignorable instance acquisition time. Users usually have different requirements on performance and monetary cost [8]. One may want to minimize the monetary cost of

2.3 Security Requirements

Confidentiality: outsourced data must be protected from the TTP, the CSP, and users that are not granted access. Integrity: outsourced data are required to remain intact on cloud servers. The data owner and authorized users must be enabled to recognize data corruption over the CSP side. Newness: receiving the most recent version of the outsourced data file is an imperative requirement of cloud-based storage systems. There must be a detection mechanism if the CSP ignores any data-update requests issued by the owner. Access control: only authorized users are allowed to access the outsourced data. Revoked users can read unmodified data, however, they must not be able to read updated/new blocks.

2.4 Cost optimization for workflows

Performance and cost optimizations are a classic research topic for decades. Many scheduling and provisioning algorithms have been proposed leveraging market-based techniques [12], rule-based techniques [9] and models [7] for cost, performance and other optimization objectives. Different applications re-quire different performance and cost optimizations. Many relevant performance and cost optimizations can be found in databases [19], Internet [20], distributed systems [21] and so on. Performance and cost optimizations for workflows have been well studied on grid [22], cloud [12] and heterogeneous computing environments [23]. Specifically, we review the most relevant workflow optimizations in the grid and in the cloud. Performance and cost optimization techniques have been developed for a single cloud provider and multiple cloud providers. There have been much more research studies on a single cloud than on multiple clouds.

On a single cloud, the research can be generally divided into three categories. The first category is auto-scaling (e.g., [5], [6]). The studies in this category are usually based on heuristics, applying a fixed sequence of transformations on the workflow. Mao et al. [5] performed the following operations one by one: merge (―task bundling‖ in their paper), deadline assignment and allocating cost-efficient machines, scaling and consolidation. This fixed sequence of workflow transformations is *ad hoc* in the sense that there are many possible sequences of transformations. The authors have not justified or prove the applicability of this fixed sequence. As demonstrated in our experiments, workflows can have very different transformations for minimizing the monetary cost. The second category is dynamic provisioning with prediction models (e.g., [33], [34], [7]). Building prediction models is orthogonal to this study. The last category is workflow scheduling [8], [9] with performance and cost constraints. Kllapi et al. consider the trade-off between completion time and money cost for data processing flows [8]. Malawski et al. [9] proposed task scheduling and provisioning methods for grouped scientific workflows called ensembles, considering both budget and deadline constraints. This paper belongs to this category, but goes beyond the existing studies in two major aspects. First, we formally formulate the performance and cost optimization process with workflow transformation models. Second, we develop a cost model guided optimization framework to optimally utilize the transformation operations.

2.5 Application security

These applications are typically delivered via the Internet through a Web browser [12,22]. However, flaws in web applications may create vulnerabilities for the SaaS applications. Attackers have been using the web to compromise user's computers and perform malicious activities such as steal sensitive data [31]. Security challenges in SaaS applications are not different from any web application technology, but traditional security solutions do not effectively protect it from attacks, so new approaches are necessary [21]. The Open Web Application Security Project (OWASP) has identified the ten most critical web applications security threats [32]. There are more security issues, but it is a good start for securing web applications. Infrastructure-as-a-service (IaaS) security issues

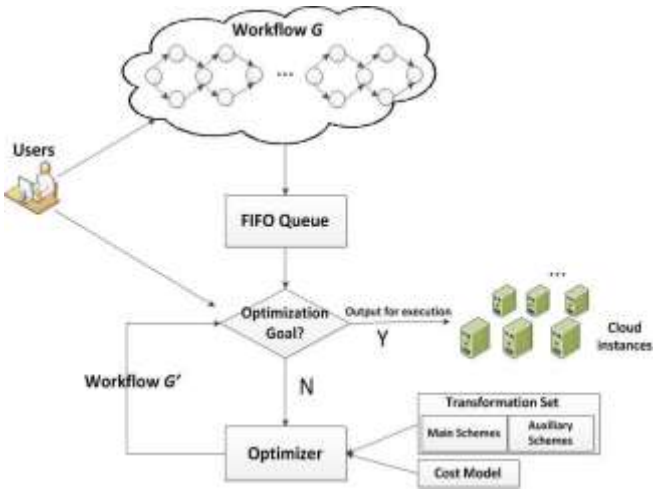
IaaS provides a pool of resources such as servers, storage, networks, and other computing resources in the form of virtualized systems, which are accessed through the Internet [24]. Users are entitled to run any software with full control and management on the resources allocated to them [18]. With IaaS, cloud users have better control over the security compared to the other models as long there is no security hole in the virtual machine monitor [21]. They control the software running in their virtual machines, and they are responsible to configure se-curity policies correctly [41]. However, the underlying compute, network, and storage infrastructure is controlled by cloud providers. IaaS providers must undertake a substantial effort to secure their systems in order to minimize

DOI: 10.18535/ijecs/v4i8.22

these threats that result from creation, communication, monitoring, modification, and mobility [42].

3 Transformation Based Optimization

ToF has two major components for performance and cost optimizations: transformation model and planner. The trans-formation model defines the set of transformation operations for a workflow, and the planner performs thetransformation on the workflow according to the cost model. Figure 1 shows the application model of ToF.



The planner is ran periodically. The period is denoted as the *plan period*. The workflows submitted during the epoch are temporarily stored in a queue, and then the planner performs optimization for all the workflows in the queue at the begin-ning of a plan period. This batch processing has two major benefits: first, maximizing the instance sharing and reuse so that the cost is reduced; second, reducing the planner overhead.

In each plan period, after all the workflows in the queue have been optimized, they are submitted to the cloud with their execution plan for execution. The execution plan is a set of instance requests. Each request includes the instance type, the starting and ending time of the requested instance and the task(s) scheduled to the instance.

To enable instance reuse at runtime, we maintain a pool of running instances, organized in lists according to different instance types. During runtime, instances in the pool which reach hourly running time and are currently idle will be turned off. An instance request is processed at the instance starting time by first looking into the instance pool for an idle instance of the requested type. If such an instance is found, it will be selected for workflow execution. Otherwise, a new instance of the requested type is acquired from the cloud provider. Thus, the instances started during workflows execution can be properly reused and their utilizations are improved. Additionally, if we can reuse the instances, the instance acquisition time is eliminated.

In current ToF, we have developed six basic transformation operations, namely *Merge*, *Split*, *Promote*, *Demote*, *Move* and

4. SCHEDULING

Co-scheduling. These basic transformations are simple and lightweight. Moreover, they can capture the current cloud features considered in this paper. They are the most common operations and widely applicable to workflow structures. For example, the operations of all the comparison algorithms used in the experiments can be represented using those transfor-mations. However, we do not claim they form a complete set. Users can extend more transformation operations into the transformation set. Based on their capabilities in reducing monetary cost, we categorize the transformation operations into two kinds, namely *main* schemes and *auxiliary* schemes. A main scheme can reduce the monetary cost while an auxiliary scheme simply transforms the workflows so that the transformed workflow is suitable for main schemes to reduce cost. By definition, Merge and Demote are main schemes, and the other four operations are auxiliary schemes.

4.1 *Main Schemes*

Merge (*M*). The Merge operation performs on two vertices when they are assigned to the same type of instances and one vertex is assigned to start (shortly) after the completion of the other one. Through merge, the two tasks are assigned to the same instance, and the two instance nodes in the instance DAG are merged to form a super-node, which maintains the hierarchical relationship among the merged nodes and also the structural dependency with other nodes in the DAG. This super-node can be treated the same as the other instance nodes for further transformations. This operation increases the usage of partial hours of instances. Take Figure 2 for example. Task A_1 and A_2 are assigned to the same instance type but different instances. In our implementation, after the Merge operation, the two tasks are running on the same instance. There is no need to adjust other vertices in the instance assignment DAG.

DOI: 10.18535/ijecs/v4i8.22

Demote (*D*). Demote operation performs on a single vertex by demoting the execution to a cheaper instance. That can reduce the cost at the risk of a longer execution time. In our implementation, we not only need to change the demoted vertex, but also need to delay the starting time of all vertices that have dependencies on the demoted vertex.

4.2 ***Auxiliary Schemes***

Move (*V*). The Move operation moves a task afterward to the end of another task in order to make chances for main schemes to further reduce cost. Note, the instance DAG uses the earliest start time and thus there is no―move forward‖ transformation. In our implementation, we not only need to change the moved vertex, but also need to delay the starting time of all vertices that are dependent on the moved vertex.

A key decision for the Move operation is where to move the task. Generally we have two cases: move a task to another task so that both tasks are assigned to the same instance type, or to a task with a different instance type. In the former case, we expect a Merge operation after the Move operation. In the latter case, a Demote and a Merge operation need to be performed to reduce cost. Our cost model is used to guide this decision. Example of the first case, where task B_1 of job B is moved to the end of task A_3. After the Move operation, we can merge A_3 and B_1.

Promote (*P*). The Promote operation is a dual operation for the Demote operation. It promotes a task to a better instance at the benefit of reducing the execution time. The implementation is the same as the Demote operation.There are mainly two incentives to perform the Promote operation. The first is to meet deadline requirement as shown in Figure 2. Second, although promotion itself is not cost-effective, it creates chances for main schemes such as the Merge operation to perform. For example, in Figure 2, we promote task A_2 from instance type *i* to a better instance type *j*. After the Promote operation, A_1, A_2 and A_3 are all assigned to the same instance type and thus can be merged to fully utilize instance partial hours.

Split (*S*). The Split operation splits a task into two, which is equivalent to suspending a running task so that we can make room for a more urgent task which is assigned to the same instance type (with a Merge operation). With the check pointing technique, the suspended task can restart after the completion of the urgent task. The Split operation causes the check pointing overhead. The Split operation splits a node in the instance DAG into two sub-nodes, at the time point when the urgent task starts. We maintain their structural dependency with other nodes in the DAG. The sub-nodes can be treated the same as the other instance nodes for further transformations.Initial instance assignment. It considers multiple heuristics. We experimentally evaluate these heuristics, and pick the one with the best result. In this paper, we present three initialization heuristics for initial instance assignment, namely

Best-fit, *Worst-fit* and *Most-efficient*. The Best-fit heuristic assigns each task with the most ex-pensiveinstance type. This is to maximize performance but at the cost of a high monetary cost. Ideally, it should satisfy the deadline. Otherwise, we raise an error to the user.The Worst-fit heuristic first assigns each task with the cheapest instance type to minimize the cost. Then, we apply the GAIN approach [40] to repeatedly re-assign tasks to a better instance type. GAIN is a greedy approach which picks the task with the largest benefit in execution time until the deadline requirement is met.

The Most-efficient heuristic configures each task according to deadline assignment. It first performs deadline assignment using an existing approach [41] and parallelism. Ligo has more parallelism and Montage has more complicated structure. We generate tasks of different types from a set of predefined task types. The execution time of four different types of tasks on the m1.small instance type are set as 30, 90, 150 and 210 minutes. For each task type, we estimate its execution time on other instance types according to Amdahl's law. Assume the execution time of a task on an instance of type *i* is t_i. Assume the CPU capability of the type-*j* instance is *k* times as powerful as the type-*i* instance, we have the task execution time on the instance of type *j* to be $t_j = Pr \times \tau_i + (1 - Pr) \times t$, where *Pr* is the parallel then assigns each task to the most cost-efficient instance type. This approach is also used in the previous study [5].

5. EVALUATION

DOI: 10.18535/ijecs/v4i8.22

In this section, we present the evaluation results of our pro-posed approach.

5.1 Experimental setup

We conduct our evaluations on real cloud platforms, including Amazon EC2 and Rackspace.

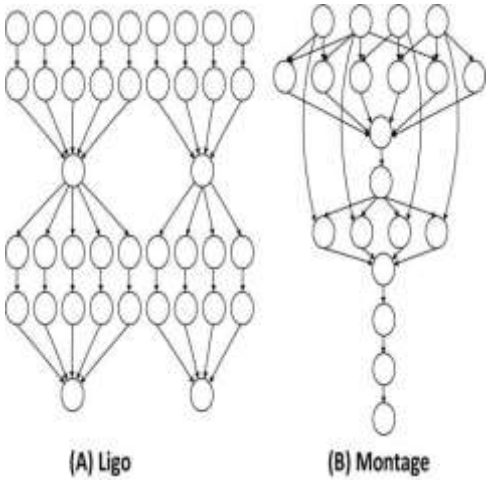


Fig. 4. Workflow structures of Ligo and Montage

We have used synthetic workloads based on two real-world applications, namely Montage and Ligo. Montage is an astronomical application widely used as a Grid and parallel computing benchmark. Ligo (Laser Interferometer Gravitational Wave Observatory) is an application used to detect gravitational-wave. The structures of the two workflows are shown in Figure 4. The two applications have different workflow structuresrate defined in Amdahl's law. The I/O and network time of workloads are included in the sequential part of the total execution time. In Section 6.3.2, we vary *Pr* to study the effectiveness of ToF on workloads with different I/O and network characteristics. The performance degradation rate for Co-scheduling operation is around 1:25 in our study. We define D_{max} as the execution time of a workflow when its tasks are all assigned to the cheapest instance type while D_{min} as the execution time of a workflow when all tasks are assigned to the most expensive instance type.

To assess our framework under the context of different workflow types, we study the workload with continuous sub-missions of one workflow type (either Montage or Ligo), as well as a mixed workload of the two applications. In the mixed workload, the workflow type is randomly selected between Montage and Ligo. We assume the workflow submission rate follows Poisson distribution. By default, the arrival rate of workflows is 0:1 per minute. For each workload, we submit around 100 jobs which is sufficiently large for measuring the stable performance.

5.2 Evaluations on Cost Estimation

In this section, we evaluate the accuracy of our cost estimation model. The real and estimated monetary cost saving of specific operations on Ligo and Montage work-flows in one plan period. The estimated monetary cost saving is returned by our cost estimation model and the real cost saving is calculated by differentiating the total monetary cost before and after performing an operation. The estimated cost is almost the same as the real cost on both workflows except for some outliers. The further cost change caused by other tasks dependent on the operated task is ignored in estimation. The optimization overhead of ToF is 0.4 seconds and 0.2 seconds for one Montage job and one Ligo job, respectively.

5.3 Results on Cost Optimizations

In this section, we first present the overall comparison re-sults of ToF with other optimization methods on minimizing monetary cost and then conduct sensitivity studies of ToF on different parameters. The optimization goal is to minimize the monetary cost given a deadline to each workflow the overall monetary cost results of ToF, Auto-scaling and Greedy methods on the Montage, Ligo and mixed workload using the pricing scheme of Amazon EC2. The standard errors of the overall monetary cost of ToF, Auto-scaling and Greedy are 0.01–0.05, 0.01–0.06 and 0.03–0.06 respectively on the tested workloads. On average, ToF obtains the smallest monetary cost in all three workloads, meaning the designed transformation set is suitable for different structures of workflows. ToF saves more monetary cost than Baseline, Auto-scaling and Greedy by 21–27%, 21–30% and 15–17%, respectively. Auto-scaling has similar monetary cost result as Baseline. This means Auto-scaling has missed a great number of optimization opportunities that can be discovered by our transformation operations. Figure 5(b) shows the average execution

time results. The standard errors of the average execution time of ToF, Auto-scaling and Greedy are 0.02–0.08, 0.02–0.07 and 0.05–0.1 respectively on the tested workloads. Although the average execution time of ToF is longer than the other algorithms, it guarantees the deadline requirement in all cases.

5.4 Results on Performance Optimizations

In this section, we evaluate ToF with the goal of minimizing workflow execution time given a budget on each workflow. The extension is simple. First, in the initialization step, the three initialization heuristics are extended to satisfy budget requirement. The Best-fit and Most-efficient heuristics adopt the GAIN [40] approach to make sure of budget while the Worst-fit heuristic simply assigns each task to the cheapest instance type. The planner guides the optimization process with a time- based rule. Since Auto-scaling was not proposed to optimize execution time, we also modified it in the following way: starting from a tight deadline, each task is configured with the most cost-efficient instance type according to deadline assignment. According to the current task configuration, if the total cost is larger than budget, we loose the deadline a bit and re-run the task configuration until the total cost is less than budget. We resume the rest of the auto-scaling optimizations from this preprocessed state.

6. CONCLUSION

Performance and monetary cost optimizations for running workflows from different applications in the cloud have be-come a hot and important research topic. However, most existing studies fail to offer general optimizations to cap-ture optimization opportunities in different user requirements, cloud offerings and workflows. To bridge this gap, we pro-pose a workflow transformation-based optimization frame-work namely ToF. We formulate the performance and cost optimizations of workflows in the cloud as transformation and optimization. We further develop a cost model guided planner to efficiently and effectively find the suitable transformation sequence for the given performance and cost goal. We evaluate our framework using real-world scientific workflow applications and compare with other state-of-the-art scheduling algorithms. Results show our frame-work outperforms the state-of-the-art Auto-scaling algorithm by 30% for monetary cost optimization, and by 21% for the execution time optimization. Moreover, the planner is lightweight for online optimization in the cloud environments. As for future work, we consider ToF on multiple clouds. Still, there are many practical and challenging issues for current multi-cloud environments [42]. Those issues include relatively limited cross-cloud network bandwidth and lacking of cloud standards among cloud providers.

REFERENCES

[1] Amazon Case Studies, –http://aws.amazon.com/solutions/case-studies/.‖

[2] Windows Azure Case Studies, –http://www.microsoft.com/azure/ casestudies.mspx.‖

[3] S. Ibrahim, B. He, and H. Jin, –Towards pay-as-you-consume cloud computing,‖ in *SCC*, 2011, pp. 370–377.

[4] C. Chen and B. He, –A framework for analyzing monetary cost of database systems in the cloud,‖ in *WAIM'13*, 2013, pp. 118–129.

[5] M. Mao and M. Humphrey, –Auto-scaling to minimize cost and meet application deadlines in cloud workflows,‖ in *SC*, 2011, pp. 49:1–49:12.

[6] M. Mao, J. Li, and M. Humphrey, –Cloud auto-scaling with deadline and budget constraints,‖ in *GRID*, 2010, pp. 41–48.

[7] E.-K. Byun, Y.-S. Kee, J.-S. Kim, and S. Maeng, –Cost optimized provisioning of elastic resources for application workflows,‖ *Future Gener. Comput. Syst.*, vol. 27, no. 8, pp. 1011 – 1026, 2011.

[8] H. Kllapi, E. Sitaridi, M. M. Tsangaris, and Y. Ioannidis, –Schedule optimization for data processing flows on the cloud,‖ in *SIGMOD*, 2011.

[9] M. Malawski, G. Juve, E. Deelman, and J. Nabrzyski, –Cost- and deadline-constrained provisioning for scientific workflow ensembles in iaas clouds,‖ in *SC*, 2012, pp. 22:1–22:11.

[10] Y. Gong, B. He, and J. Zhong, –Network performance awarempi collective communication operations in the cloud,‖ *IEEE Trans. Parallel Distrib. Syst*, vol. 99, no. PrePrints, 2013.

[11] M. Mao and M. Humphrey, –Scaling and scheduling to maximize application performance within budget constraints in cloud workflows,‖ in *IPDPS*, 2013.

[12] H. M. Fard, R. Prodan, and T. Fahringer, –A truthful dynamic workflow scheduling mechanism for commercial multicloud environments,‖ *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1203–1212, 2013.

[13] Q. Zhu, J. Zhu, and G. Agrawal, –Power-aware consolidation of scientific workflows in virtualized environments,‖ in *SC*, 2010, pp. 1–12.

[14] H. Wang, Q. Jing, R. Chen, B. He, Z. Qian, and L. Zhou, –Distributed systems meet economics: pricing in the cloud,‖ in *HotCloud*, 2010, pp. 6–6.

[15] H. Herodotou and S. Babu, –Profiling, what-if analysis, and cost-based optimization of mapreduce programs,‖ *PVLDB*, vol. 4, no. 11, pp. 1111– 1122, 2011.

[16] J. C. Jacob, D. S. Katz, G. B. Berriman, J. C. Good, A. C. Laity, E. Deelman, C. Kesselman, G. Singh, M. Su, T. A. Prince, and R. Williams, –Montage: a grid portal and software toolkit for science,

grade astronomical image mosaicking,‖ *Int. J. Comput. Sci. Eng.*, vol. 4, no. 2, pp. 73–87, Jul. 2009.

[17] E. Deelman, C. Kesselman, G. Mehta, L. Meshkat, L. Pearlman, and S. Koranda, ―Griphyn and ligo, building a virtual data grid for gravitational wave scientists,‖ in *HPDC*, 2002.

[18] J.-S. Vockler,¨ G. Juve, E. Deelman, M. Rynge, and B. Berriman, ―Ex-periences using cloud computing for a scientific workflow application,‖ in *ScienceCloud*, 2011, pp. 15–24.

[19] G. Graefe and J. Gray, ―The five-minute rule ten years later, and other computer storage rules of thumb,‖ Microsoft Research, Tech. Rep. MSR-TR-97-33, 1997.

[20] R. T. B. Ma, D.-M. Chiu, J. C. S. Lui, V. Misra, and D. Rubenstein, ―Internet economics: the use of shapley value for isp settlement,‖ in *CoNEXT*, 2007.

[21] J. Gray, ―Distributed computing economics,‖ Microsoft, Tech. Rep. tr-2003-24, 2003.

[22] S. Abrishami, M. Naghibzadeh, and D. H. J. Epema, ―Cost-driven scheduling of grid workflows using partial critical paths,‖ *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 8, pp. 1400–1414, Aug. 2012.

[23] H. M. Fard, R. Prodan, J. J. D. Barrionuevo, and T. Fahringer, ―A multi-objective approach for workflow scheduling in heterogeneous environments,‖ in *CCGRID*, 2012, pp. 300–309.

[24] J. Yu and R. Buyya, ―A taxonomy of workflow management systems for grid computing,‖ JGC, Tech. Rep., 2005.

[25] R. Sakellariou and H. Zhao, ―A hybrid heuristic for dag scheduling on heterogeneous systems,‖ in *IPDPS*, 2004.

[26] E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, A. Laity, J. C. Jacob, and D. S. Katz, ―Pegasus: A framework for mapping complex scientific workflows onto distributed systems,‖ *Sci. Program.*, vol. 13, no. 3, pp. 219–237, 2005.

[27] Y. C. Lee, R. Subrata, and A. Y. Zomaya, ―On the performance of a dual-objective optimization model for workflow applications on grid platforms,‖ *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, no. 9, pp. 1273– 1284, sep 2009.

[28] J. Yu and R. Buyya, ―Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms,‖ *Sci. Program.*, vol. 14, no. 3,4, pp. 217–230, 2006.

[29] G. Mateescu and G. Mateescu, ―Quality of service on the grid via metascheduling with resource co-scheduling and co-reservation,‖ *Int. J. High Perform. Comput. Appl.*, vol. 17, pp. 209–218, 2003.

[30] K. Yoshimoto, P. Kovatch, and P. Andrews, ―Co-scheduling with user-settable reservations,‖ in *JSSPP*, 2005, pp. 146–156.

[31] R. Buyya, C. S. Yeo, and S. Venugopal, ―Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities,‖ in *HPCC*, 2008, pp. 5–13.

[32] I. Foster, Y. Zhao, I. Raicu, and S. Lu, ―Cloud computing and grid computing 360-degree compared,‖ in *GCE*, 2008, pp. 1–10.

[33] M. D. de Assuncao, A. di Costanzo, and R. Buyya, ―Evaluating the cost-benefit of using cloud computing to extend the capacity of clusters,‖ in *HPDC*, 2009, pp. 141–150.

[34] H. C. Lim, S. Babu, J. S. Chase, and S. S. Parekh, ―Automated control in cloud computing: challenges and opportunities,‖ in *ACDC*, 2009, pp. 13–18.

[35] J. Lucas Simarro, R. Moreno-Vozmediano, R. Montero, and I. Llorente, ―Dynamic placement of virtual machines for cost optimization in multi-cloud environments,‖ in *HPCS*, 2011, pp. 1–7.