Optimizing Energy Efficiency in The Cloud Using Service Composition and Runtime Adaptation Techniques

Usman Wajid
School of Computer Science
The University of Manchester
United Kingdom
usman.wajid@manchester.ac.uk

César A. Marín, Anthony Karageorgos
Service Systems, Manchester Business School
The University of Manchester
United Kingdom
{cesar.marin, anthony.karageorgos}@manchester.ac.uk

***Abstract*—This paper describes an approach for service composition optimization and its application in cloud computing to streamline resource usage that in turn contributes towards energy efficiency. The suitability and usefulness of the approach is evaluated by experimentation. In the experiments, physical hosts at various cloud sites represent candidate services that are brought together in a composition to satisfy the requirements of applications. The composition is optimized based on functional and non-functional criteria to determine a set of cloud services representing energy efficient deployment configurations. We also propose a runtime adaptation model that can help in minimizing energy consumption of cloud applications at runtime.**

Keywords - optimization, energy efficiency, cloud computing

I. INTRODUCTION

Service composition involves combining the functionality of multiple services in a unified solution on the basis of several factors, e.g., cost, performance, SLAs, etc. However, an automated approach for service composition may end up with sub-optimal solution (or composition) for a number of reasons e.g. some services may not work properly with other services in a composition and there can be a number of services that offer the same functionality with minute variations. Thus selecting a set of service that when brought together in a composition can achieve a specific goal is a multi-criteria optimization problem. To address such problems, composition optimization approaches are devised to identify services, by applying multi-criteria selection procedures, which can work together while contributing towards the overall goal of the composition.

In this paper we describe an existing service composition optimization approach and analyze its performance in solving a multi-criteria optimization problem in cloud computing. The problem relates to determining a set of cloud services that when brought together (as composition) can ensure optimal energy consumption in the cloud while satisfying the application resource requirements and execution constraints.

In the rest of the paper, we provide an overview of the composition optimization approach that we use for determining energy efficient application deployment configurations (Section II). Further we describe the application domain of the optimization approach and the problem we aim to solve using the approach (Section III). In Section IV we describe the design of the experiments and Section V presents the results of the experimentations that show suitability of the optimization approach in achieving the desired objective. Section VI describes an adaptation model devised to ensure minimal energy consumption by application throughout their deployment lifecycle on the cloud. In the end, Section VII presents an overview of existing work on energy efficiency in cloud computing before drawing direction for our future work in this area.

II. OVERVIEW OF COMPOSITION OPTIMIZATION APPROACH

Here we provide an overview of an existing composition optimization approach that we use in our study. The optimization approach [2] was basically developed for semantically annotated services or semantic web services and uses semantic link-based approach [2] for service composition. In this approach a link is established between the output and input parameter of two services in a way to partially link them based on a matching functionality. This matching functionality is represented by a matching function that enables, at design time, finding some types of compatibilities and incompatibilities among the descriptions of different services. The links between services in a composition are valued based on two functional quality criterion derived from heuristics [7].

a. *Common description of a link between two services* - the first criterion estimates the proportion of service descriptions and how they match with each other.

b. *Matching quality of a link* - the second criterion calculates the matching quality between service links as a function of cost. Where, lower cost represents better matching quality between two services.

In the optimization approach [2] the above functional quality model is extended by considering non-functional properties of services (also known as Quality of Service or QoS parameters) in the composition. In [2] these QoS parameters include Price (i.e. fee requested by the service provider) and

978-1-4799-0652-9/13/$31.00 ©2013 IEEE

Response Time (i.e. expected delay between request and response time of a service). Based on this extended quality model the best quality link (between two services in a composition) is the one with best functional quality (common description, matching quality) and non-functional quality (the cheapest and fastest services).

In this respect, the optimization approach determines the quality of a service composition by considering the overall matching quality (functional and non-functional) of its links. The prototype implementation of the optimization approach (described in [2]) is based on the interplay of three components described below:

i. *Service repository* (implemented as a database) to store candidate web-services.

ii. *Optimizer* or genetic algorithm (implemented in Java) to compute the best set of services in a composition while considering the functional and non-functional quality constraints

iii. *DL Reasoner* (an adaptation of Fact++ [1]) to compute the quality of links between different services.

In the next section, we elaborate how the existing optimization approach was adapted for achieving the desired objective in cloud computing domain.

III. COMPOSITION PROBLEM AND CANDIDATE SERVICES

In this section we describe an application and a cloud model. Together these two models form the basis of the problem that we address by applying the composition optimization approach from Section II.

A. Application model and composition problem

We assume that applications are composed of one or more tasks (sample shown in Figure 2) that can be deployed in a single VM or in distributed VMs at different physical hosts in a cloud infrastructure.

Each task has specific resource requirements (e.g. memory, CPU) that need to be fulfilled for successful execution of the task as well as the overall application. Furthermore, tasks interact or exchange information with each other. Based on this model, an application can be represented by a process template that specifies the set of tasks required for successful execution of application, interlinked in a specific order based on any execution-level constraints. The tasks in a process can be annotated with information about specific resource requirements and any execution level constraints. Based on the availability of an annotated process the problem, addressed in this paper, is to find a set of cloud services that, when brought together, can adequately perform the tasks (i.e. satisfy task requirements and constraints) while consuming minimal energy.

B. Cloud model and candidate services

We consider a federated cloud model where the cloud is composed of multiple cloud sites brought together in a federation (cf. [8]). The different layers of federated cloud are shown in Figure 1. Based on this layered representation, a federated cloud is composed of more than one cloud site, where each site can have more than one physical host and each host can run one or more VMs.

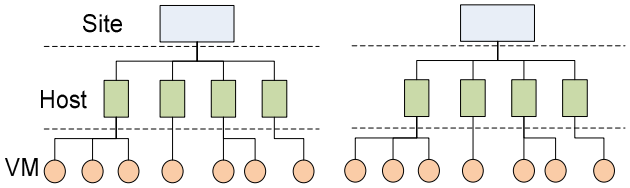

Figure 1: Layered representation of federated cloud infrastructure

Since application deployment in cloud typically involves analyzing the suitability of resources (e.g. available CPU, free memory) offered by physical hosts against the requirements and execution constraints of applications, we consider the host layer as an adequate level to consider in an application deployment decision making model.

Hence, we assume in a federated cloud scenario physical hosts can be represented by cloud services, in a way that each cloud service possesses the following characteristics that contribute towards application deployment decision making:

- *Free memory*
- *Available CPU*
- *Number of active VMs*
- *Power Consumption (in appropriate measures)*

The availability of different cloud services in a federated cloud scenario represents opportunities for the execution of the application tasks on different physical hosts - across distributed cloud sites. In this respect, more cloud services mean more chances of finding suitable cloud services for the execution of tasks, which could mean more complexity in computing the suitability and compatibility of services that can be used in a composition.

IV. DETERMINING APPLICATION DEPLOYMENT CONFIGURATIONS UISNG SERVICE COMPOSITION

Based on the availability of a process template and set of cloud services, we designed some experiments with the purpose to analyze the suitability of the service composition optimization approach (from Section II) in determining energy efficient application deployment configurations. The experiments also allowed us to study the effects of dynamic changes (in the numbers and characteristics of candidate cloud services) on the behavior of the optimization approach.

The design of the experiments involves the following:

Services: the offerings of each physical host in the federated cloud were represented by a cloud service (or web-service in the context of the experiments). Cloud services were annotated with semantic descriptions about the functionality (or resources) they offered and all services were stored in a centralized service repository.

Application: an application (to be deployed on cloud) was represented by a process template composed of several tasks interlinked using a combination of sequential patterns as well as AND/OR branching operators representing execution level

constraints, as shown in Figure 2. Each task was annotated to describe the resource requirements, execution constraints or dependencies. The task annotations enabled searching for candidate cloud services that can offer suitable functionality (or resources) based on specific requirements and constraints.

Composition Optimization: We modified the task annotations and service descriptions to match the specific nature of cloud computing domain. Thus enabling functional quality parameter *Common Description* (as described in Section II) to look for task-service pairs based on domain specific information. Furthermore, since we were mainly interested in optimizing the service compositions based on energy consumption parameter the non-functional quality parameter *Price* (from Section II) represented the energy consumption unit for cloud services in the composition.

Thus in our experimentation a composition with lowest overall price represented most energy efficient deployment configuration for the underlying application.

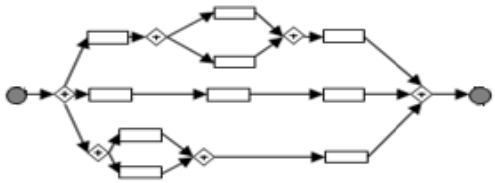

Figure 2: Process template showing various tasks of application

Our experimentation involved a set of simulations where each simulation involved the following three stages:

1. The simulation starts with a process template, where different (annotated) application tasks are organized in a specific order. Once the experiment is initiated the service composition module searches the service repository for candidate services that can perform the different tasks in the process. The result of this step is a *non-optimal* composition where at least one suitable cloud service is associated with each task in the process. The services are selected based on matching task requirements with services descriptions/offerings.
2. Next, the composition is optimized by replacing certain services in the template with better alternatives from the service repository. Here, the optimization approach [2] evaluates the combination of task-service pairs based on function and non-functional optimization criteria.
3. We execute the previous step 63 times during a simulation. At each execution (simulation cycle) we dynamically changed the pool of available services to simulate the dynamism that may be attributed to the changes in application resource requirements, properties of cloud resources and their availability. Consequently, the genetic algorithm optimizes a problem slightly different from one time to the next one.

V. RESULTS FROM EXPERIMETNS

Each simulation is composed of three stages of approximately 21 simulation cycles each: 1) starting from an initial number of 200 services, services are added gradually to the service repository until we reach an empirical maximum of 10,000 services; then 2) gradually those services' parameters are semi-randomly changed to reflect variations in available resources; finally 3) the number of services is reduced gradually until 200 services is reached again. At each simulation cycle, the genetic algorithm was requested to optimize the process template using the available set of services. This process was done to analyze how the optimization results changed over time by making differentiations in the problem the genetic algorithm is optimizing. This emulates the dynamism which the optimization approach does not have control over but have to take into account by consecutive optimizations.

The idea was to analyze the variations of optimization outputs after several consecutive optimizations based on functional and non-functional (such as common description and energy consumption) criteria. Thus the total execution time of the application and physical location where the cloud services were deployed was not taken into account.

We consider that 20 simulations are sufficient to provide any statistical significance to the results. Yet we ran 40 simulations in our experiment and calculated the median after each simulation cycle. The median was chosen because it does not assume any underlying distribution in the analyzed data.

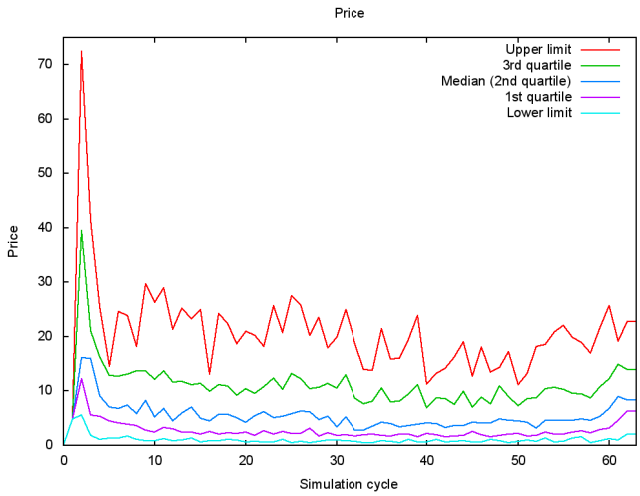

Figure 3: Distribution of Price in 40 simulations (consult colored version of figures for better appreciation)

Figure 3 shows the distribution of Price (or overall energy consumption) over 40 simulations. Here, it can be appreciated that the price variation at the beginning is higher than at any other time during the experiment. This is because the optimization starts with a non-optimal composition. Hence, it is evident in Figure 3 that the optimization approach tries to keep the price at the minimum while yielding consecutive low price or energy efficient compositions despite the variations in the number and properties of services.

Figure 4 presents the median of the Price and the number of services available. Here the three simulation stages are clearer; it can be appreciated that as the number of services drops, the Price goes up. This is expected because the size of the search

space is reduced (less services) prompting the algorithm to do its best with minimum services.

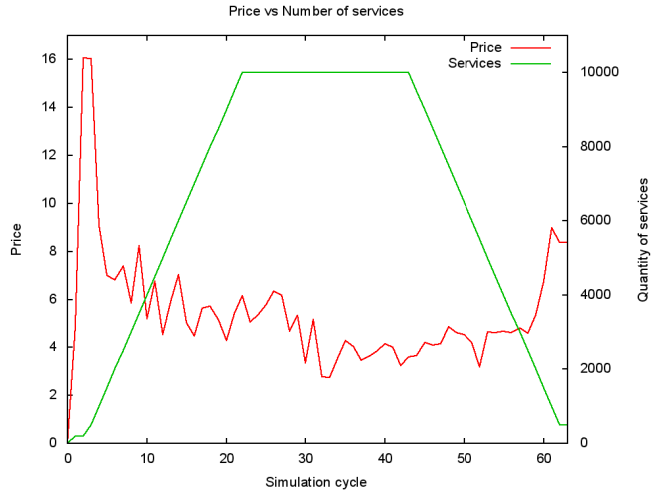

Figure 4: Price (median) vs number of services. The scale of the Y axis has been adjusted for better appreciation

Figure 5 depicts the distribution of composition Quality in the whole experiment. Here quality is measured as cost between 0 and 1, hence lower cost refers to better match or compatibility between services. Similar to the Price, the Quality shows more variation at the beginning of the experiment and at the end. However, when the number of services goes to 10,000, the Quality stabilizes at the same point in all the 40 simulations. Clearly this parameter is affected by the number of services available for optimization.

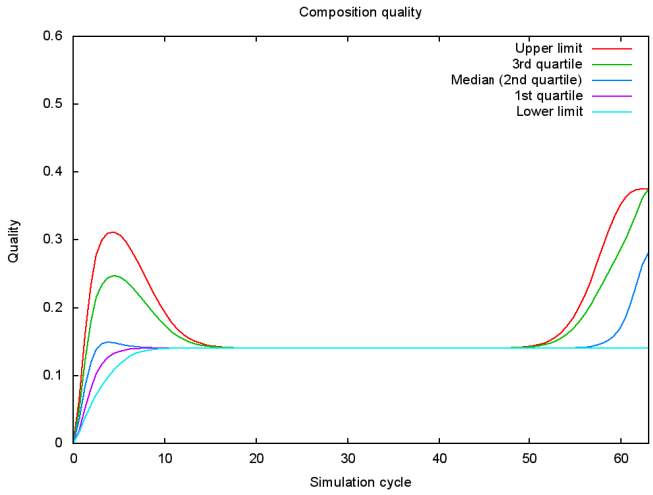

Figure 5: Distribution of Quality in 40 simulations (Smoothing filter has been applied for better appreciation)

Finally, Figure 6 illustrates the medians of Price and Quality. It can be seen that both parameters values are higher when the number of services is reduced, i.e. at the beginning and at the end of the experiment. Yet the Price is more sensitive to slight variations of services because it is directly dependent on the individual services selected for the optimal composition. On the other hand, Quality is mainly affected by the number of services available for composition rather than by those selected for the optimal one.

The above results support the use of optimization approach for streamlining resource usage that can turn improve energy efficiency in cloud computing. The results also show the ability of the optimization approach to perform well under dynamic situations when changes are introduced in the properties and number of available resources. The outcome of our experimentation i.e. an optimized service composition can be easily translated into an energy efficient application deployment configuration allowing different tasks of the application to be deployed or re-deployed on specific cloud services to achieve energy efficiency in the cloud infrastructure.

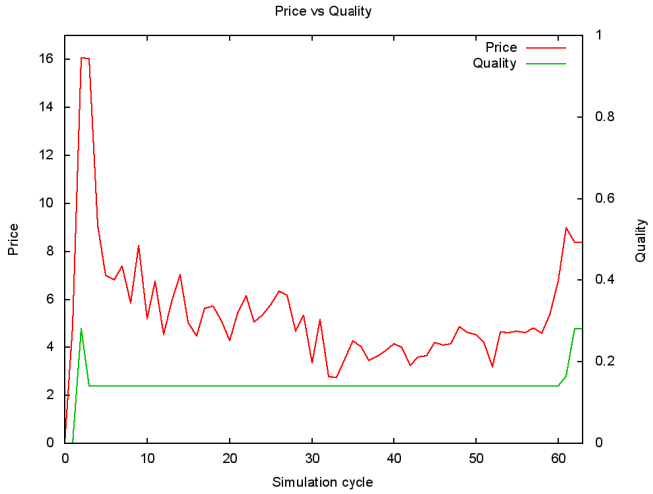

Figure 6: Price (median) vs quality (median)

VI. RUNTIME ADATPATION MODEL

As mentioned in Section III.B application tasks are executed in cloud services which are associated with certain hardware and software characteristics such as energy consumption and number of VMs. Once an application is deployed in the cloud, in many cases the allocation of tasks to cloud services may become inefficient prior to actual task execution. For example when a task has not started execution because it is waiting for input, when new cloud services appear or disappear or when the characteristics of cloud services, namely memory, CPU availability and numbers of VMs change dynamically e.g. when some of these characteristics are modified by previous task allocations.

Furthermore, there may be cases where cloud service availability is dynamically modified such as when hosts are temporarily not available due to hardware and network faults and varying host availability. Therefore it becomes mandatory that allocation of tasks to cloud services adapts on runtime to ensure energy-efficient operation of the cloud infrastructure.

Additionally a centralised coordination authority is not always the best option. In decentralised cloud settings establishing and maintaining a centralised coordinator incurs considerable overheads and is prone to bottleneck delays and network faults. Furthermore, in centralised cloud settings avoiding central reconfiguration results in less CPU usage and

communication traffic between involved parties which in turn contributes towards energy efficiency. Consequently there are valid arguments for run-time re-arrangement of task allocation to cloud services in a decentralised manner aiming to adaptively optimise energy efficiency.

A well-known approach to distributed problem solving is utility-based optimisation, which is widely applied in many application domains [9][10]. In this approach distributed problem solving components are assumed to take rational decisions aiming to maximise their individual utility. Component utility is calculated using a suitable utility function and each component bases its decision on the calculated utility associated with alternative decision options by identifying the option that would result in maximum individual utility. Components are generally not fully interconnected and a key aspect is that the individual component utility depends on the decisions that have been previously taken by its neighbours, namely the components to which it is directly connected. The overall system utility is commonly calculated as a sum of the individual component utilities and therefore the local decisions towards maximising individual utilities contribute towards maximising the overall system utility. Due to the interdependency of component utilities the above optimisation process generally requires more than one step since at each step a component may select to change its decision if the utility resulting from the alternative options has changed due to recent changes to decisions by its neighbours. This generates an evolutionary process which in many cases has been found to produce quite satisfactory results [11] and has been shown to be applicable in adaptive management of cloud infrastructure resources [12].

We propose to use a utility-based decentralised optimisation scheme for run-time adaptive task-to-cloud service allocation aiming to optimise energy efficiency.

Let T a directed acyclic task graph comprising M tasks and S a set of N cloud services. The edges of T define execution precedence relations between tasks, that is to start executing a task requires all its predecessors to have finished execution. Otherwise tasks can be executed at any order.

Each task t_i has specific memory, CPU, and number of VMs requirements, namely m_i^r, cpu_i^r and vm_i^r. Each cloud service $cs_j \in S$ has specific memory, CPU and number of VM capacities denoted by m_j^{max}, cpu_j^{max} and vm_j^{max} respectively. Furthermore, each cs_j consumes p_j amount of energy per unit time when in operation.

Cloud services belong to service federations, Furthermore, we consider that each cloud service comprises a CPU capacity cpu_j^{cs}. We further consider that the CPU requirement cpu_i^r of task t_i is also measured in instructions per second (ips) and we denote by d_i the required duration of task t_i execution.

The main underlying assumption is that to optimise energy efficiency, tasks are assigned to cloud services in a manner that task requirements are met and energy consumption is minimised. However, task execution does not necessarily begin immediately, for example in the case where tasks are waiting for input that will be produced from preceding tasks in the task graph. The operating system may be utilising this unused CPU capacity for internal tasks, however the application tasks are guaranteed to receive the allocated CPU capacity when they are ready to commence execution, hereby we assume a real-time task execution approach.

Hence, the expected energy consumption c_{ij} of each task t_i assigned to cloud service j depends on its estimated duration d_i and the unit energy consumption requirement of the cloud service to which it is allocated. That is:

$$c_{ij} = d_i \cdot \frac{p_j}{cpu_j^{max}} \tag{1}$$

Following the utility-based optimization approach [9][10] the utility of each task is defined as the inverse of its expected energy consumption, that is:

$$u_i = \frac{1}{c_{ij}} \tag{2}$$

and the global system utility is defined as the sum of all individual task utilities:-

$$U = \sum_i u_i \tag{3}$$

The way utility functions have been defined is unrelated to the task dependency relations and concerns only energy consumption. Then the problem to be solved is defined as:

$$\max \sum_i u_i$$

subject to:

$$\sum_{k=1}^{n_j} vm_k^r \le vm_j^{max} \ , \quad \sum_{k=1}^{n_j} m_k^r \le m_j^{max} \ , \quad \sum_{k=1}^{n_j} cpu_k^r \le cpu_j^{max}$$

$$j = 1..N$$

where n_j is the number of tasks allocated to cloud service j.

We consider that each task is represented by an appropriate software component, such as a software agent. This representation enables tasks to maintain connections with the other tasks t_k in the same task graph T. Furthermore, tasks maintain links to other cloud services in the federation S that are not currently assigned a task of task graph T.

Therefore, a task is aware of the characteristics, such as memory, cpu and number of VMs, of all cloud services in the federation of the cloud service it is currently assigned to, and of all cloud services of the federations that the other tasks of the same task graph T are currently assigned to. Tasks are initially assigned to cloud services by some external authority and then act independently. Finally, the availability of cloud services can also vary resulting in a fully dynamic model.

When tasks select a service to execute them they become associated with that service, they transfer there if located remotely, and they are placed in a queue waiting for execution. Tasks can change their selected service before the start of their

execution according to utility-based local optimization rules described in Algorithm 1. A simple way to trigger adaptation is to have each task periodically check the utility of the current service selection and modify it as needed.

Let S_i be the cloud services assigned with a task from the same task graph T as t_i or belonging to the same federation as the cloud service j currently assigned with task t_i. For every cloud service j we denote with m_j, cpu_j and vm_j the amounts of currently available memory, CPU capacity and VMs respectively. Then for each task t_i currently allocated to cloud service j we can apply the following algorithm:

Algorithm 1:

$d \leftarrow 0$
$k \leftarrow 1$
while $k \leq |S_i|$ do
    if $cpu_i^r < cpu_k$ **and** $vm_i^r \leq vm_k$ **and** $m_i^r \leq m_k$ **and** $c_{ik} < c_{ij}$ **then**
        $cost[d] = c_{ik}$
        $d \leftarrow d + 1$
    end if
    $k \leftarrow k + 1$
end while
$newS \leftarrow argmin(cost[])$
$assign(newS, t_i)$

Successive applications of the above algorithm for all tasks that have not yet started their execution will result in adaptively balancing task allocation so that overall energy consumption is reduced. However further experimentation is required to identify the algorithm behaviour for different types of task graphs and cloud service federations. Furthermore, possible extensions to the algorithm than would reduce the number of iterations need to be examined.

VII. Existing Work and Future Directions

Energy efficiency gets relatively less attention as relevant context information for application deployment on clouds and current research while focusing on security and other SLA related issues overlooks the energy consumption and resulting environmental implications as a critical factor of application deployment on cloud.

Among the first efforts on energy efficiency, [4] use game theoretical methodologies to optimize energy consumption with system performance in a grid environment. Further, in [5] Kolodziej *et al* describe genetic-based scheduling techniques that apply dynamic voltage and frequency scaling models to reduce the cumulative energy consumed by computational resources. The dynamic voltage and frequency scaling allow the processing units to operate at various energy consumption levels, thus making it feasible for multi-criteria, multi-objective genetic-base scheduling techniques to workout energy efficient task allocations (to appropriate processing units) based on user requirements. However, their approach requires support from specialized hardware components (e.g. CPU) to dynamically manipulate energy or performance parameters and such support cannot be promised in a multi-site cloud infrastructure. Further, in [6], Lingberg *et al* provide an overview of eight scheduling algorithms that can be used in scheduling tasks on a distributed system with the aim to minimize energy consumption subject to task resource requirements and constraints. The algorithms use a variety of techniques such as iterative, greedy, constructive and genetic. The results from simulating the algorithms in a simulation testbed are used in the performance comparison, allowing the selection of different algorithms based on specific requirements. The work presented in this paper can be seen as an addition to the set of techniques discussed in [6].

In our future work, we intend to introduce more cloud related parameters in the optimization approach (such as network traffic and latency) and study the behavior of optimization approach when considering cross-site demographics and application execution time constraints. Further, we intend to evaluate the run-time adaptation model by means of experimentation or simulation before investigating the integration of two models (for initial deployment and runtime adaptation) in a comprehensive energy efficiency application deployment solution for federated clouds.

References

[1] Horrocks, I. (1998) Using an expressive description logic: FaCT or fiction? In: KR. 636–649

[2] Lecue. F and Mehandjiev. N., Seeking Quality of Web Service Composition in a Semantic Dimension. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*(2010)

[3] Lecue, F., Leger, A. (2006) *A formal model for semantic web service composition*. In proceedings of International Semantic Web Conference. 385–398

[4] Khan. S. Ahmed. I.,*A cooperrative game theoretical tehcnique for joint optimization of energy consumption and response time in computational grids*. IEEE Transactions on Parallel and Distributed Systems, pp 246-360. 2009

[5] Kolodziej. J., Khan. S. U., Wang. L., Zomaya. A. Y. (2012) Energy efficient genetic-based schedulers in computational Grids. *Concurrency and Computation: Practice and Experience*

[6] Lindberg. P., *et al*. (2012). Comparison and analysis of eight scheduling heuristics for the optimization of energy consumption and makespan in large-scale distributed systems. In *Journal of Supercomputing*, Vol 59, Issue 1- 323-360.

[7] Lecue, F., Delteil, A., Leger, A.: *Optimizing causal link based web service composition*. In: ECAI. (2008) 45–49

[8] BonFIRE Cloud Federation: http://www.bonfire-project.eu/infrastructure/testbeds

[9] N. Houede, et al., "Utility-Based Optimization of Combination Therapy Using Ordinal Toxicity and Efficacy in Phase I/II Trials," *Biometrics*, vol. 66, pp. 532-540, June 2010 2010.

[10] L. Skorin-Kapov, et al., "Approaches for Utility-Based QoE-Driven Optimization of Network Resource Allocation for Multimedia Services," in *Data Traffic Monitoring and Analysis. vol. 7754, E. Biersack, et al., Eds., ed: Springer Berlin Heidelberg,* 2013, pp. 337-358.

[11] A. Goel and H. Nazerzadeh, "Price based protocols for fair resource allocation: convergence time analysis and extension to Leontief utilities," presented at the Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms, San Francisco, California, 2008.

[12] M. Koehler and S. Benkner, "Design of an Adaptive Framework for *Utility-Based Optimization of Scientific Applications in the Cloud," in Utility and Cloud Computing (UCC), 2012 IEEE Fifth International Conference.* 2012.