Towards a Model for Cloud Computing Cost Estimation with Reserved Instances

Georg Singer[1], Ilja Livenson[2], Marlon Dumas[1], Satish N. Srirama[1], and Ulrich Norbisrath[1]

[1] University of Tartu, Estonia
{FirstName.LastName}@ut.ee
[2] NICPB, Estonia
ilja@kbfi.ee

Abstract. Cloud computing has been touted as a lower-cost alternative to in-house IT infrastructure recently. However, case studies and anecdotal evidence suggest that it is not always cheaper to use cloud computing resources in lieu of in-house infrastructure. Also, several factors influence the cost of sourcing computing resources from the cloud. For example, cloud computing providers offer virtual machine instances of different types. Each type of virtual machine strikes a different tradeoff between memory capacity, processing power and cost. Also, some providers offer discounts over the hourly price of renting a virtual machine instance if the instance is reserved in advance and an upfront reservation fee is paid. The choice of virtual machine types and the reservation schedule have a direct impact on the running costs. Therefore, IT decision-makers need tools that allow them to determine the potential cost of sourcing their computing resources from the cloud and the optimal sourcing strategy. This paper presents an initial model for estimating the optimal cost of replacing in-house servers with cloud computing resources. The model takes as input the load curve, RAM, storage and network usage observed in the in-house servers over a representative season. Based on this input, the model produces an estimate of the amount of virtual machine instances required across the planning time, in order to replace the in-house infrastructure. As an initial validation of the model, we have applied it to assess the cost of replacing an HPC cluster with virtual machine instances sourced from Amazon EC2.

1 Introduction

p At the infrastructure level – also known as Infrastructure-as-a-Service – cloud computing is generally associated with large datacenter operators (e.g. Amazon, GoGrid, Google, IBM) selling IT resources at a very fine level of granularity (e.g. on an hourly basis). A key claim made in this context is that replacing in-house IT infrastructures with cloud computing resources leads to significant cost savings, which are attributed to the economies of scale that such providers can obtain. In a case study written by Nucleus Research about a local TV channel in the US using Google Apps [1] an ROI of 500% and a paypack of 1 month is proclaimed. A report published by the IBM Cloud Labs [2] supplies various case studies showing annual cost savings of up to 79% and payback periods of 45 days. Akram Assaf has published a case study [3] about "Bayt.com", a recruiting service running on Amazon EC2, and shows monthly savings of 30%. However, while some case studies suggest that cloud computing resources can be cheaper than in-house servers in certain scenarios [2, 1], other case studies and anecdotal evidence suggest the opposite. Ewa Deelman et al. investigate in their paper [4] the cost of running scientific applications on the cloud and could not show any significant cost savings. This is mainly due to the fact that scientific applications are mostly CPU intensive and have limited storage needs.

It appears that whether cloud computing can lead to savings or not greatly depends on the type of usage. Also, there is a wealth of cloud computing offerings with different pricing models. One variable to be considered is the type of virtual machine instances sourced. Each type of virtual machine strikes a different tradeoff between memory capacity, processing power and cost. For example, Amazon Web Service offers a "Small Instance" that has 1.7 GB of memory and 160GB of storage at costs of \$0.0095 per hour. A "High-CPU Instance" on the other hand offers 7GB of memory and 1690 GB of storage at a cost of \$0.76[3] per hour. Another factor which has to be taken into account is related to discounts. Some cloud computing providers offer discounts when a virtual machine instance is reserved in advance (module an upfront reservation fee). This situation calls for a well-grounded model for estimating the cost of replacing in-house IT servers with cloud computing resources, that takes into account the wealth of options available. In this paper we present an initial model for estimating the optimal costs of replacing an existing IT infrastructure with cloud computing resources. The proposed model takes into account the possibility of sourcing different types of instances and the possibility of reserving instances for different lengths of time in order to reduce the hourly cost. Thus, in addition to being useful as a tool for estimating the costs of cloud computing outsourcing, the model can also be used to optimize the costs of already running applications on the cloud. To efficiently explore the large space of options, the model encodes the problem of selecting an optimal schedule and an optimal set of instances by means of linear programming. One of the key design criteria of the proposed model is that it should rely on input data that system administrators can readily obtain. Accordingly, the model takes as an input the load curve, RAM, storage and network usage observed in the in-house servers over a representative period. Such curves can be gathered using monitoring tools such as Ganglia[5] or *collectd* [6]. Based on these curves, the proposed model determines the amount and types of virtual machine instances required at each point of the planning period, in order to handle the previously observed load. For illustration purposes, the paper shows an instantiation of the model in the specific context of Amazon EC2, even though the model has been designed in such a way that it applicable to other cloud computing providers (e.g. GoGrid or ReliaCloud). This instantiation of the model has been implemented in a proof-of-concept tool that takes as

[3] All prices are given for the AWS EU-Ireland region as of 6.06.2010.

input Ganglia collected data and produces the cost of running the same load on the cloud with an optimal reservation schedule. As an initial validation, we have applied this tool to estimate the cost of replacing an existing HPC cluster hosted by the Estonian NICPB institute[4]. The cluster is used for computation-intensive non-interactive tasks. We took the historical data for a one-year period and fed it as input to the tool. The results are consistent with previous findings as in the paper on running scientific applications on the cloud by Deelman et al. [4].

The paper is structured as follows: The following section presents the general cost estimation model and its instantiation for Amazon EC2. Next, Section 3 presents the tool implementation and its validation in the context of the NICPB cluster. Section 4 the discusses related work, while Section 5 discusses our ongoing efforts at extending and further validating the proposed model.

2 Model for Cloud Usage Cost Estimation

This section describes the proposed model for estimating the cost of replacing a set of in-house servers with virtual machines rented from a public cloud such as Amazon EC2. First we describe in details the inputs and the outputs of the model. Then we present the internal details of the model and its implementation.

2.1 Model Input and Pre-Processing Step

The main input of the model are demand curves for different types of resources across the period during which the in-house servers are to be replaced with virtual machine instances (hereon called the *planning period*). We distinguish four types of resources: CPU, RAM, network and storage. Below we outline how the demand curves for each of these four types of resources are obtained and pre-processed before being fed into the cost estimation model.

The RAM demand curve required by the model can be derived from the memory usage observed in each in-house server over a given season (e.g. most recent 3 months or 1 year). The granularity of the season has to be chosen in such a way that it is *representative*, meaning that the consumption is similar from one season to another. The memory usage curve over the observed season is then replicated as many times as required to obtain a RAM demand curve for the entire planning period (e.g. 3 years).

Next, to determine the demand curve for CPU, we first take as input the load curves observed in the in-house servers during a season, and replicating them as required. Here, the term "load" is used in its classic Unix definition, that is, the average number of runnable processes over the period of time. In our experiments, we obtained the load curves by calculating the average load over time intervals of 5 minutes. Load curves are machine specific – they depend on the exact hardware and operating system configuration of the server. To be able to transpose the load curve observed in an in-house server into the load curve

[4] Ganglia charts of this cluster are available at: http://monitor.hep.kbfi.ee/

that would be observed if the same set of processes were run on the cloud, we need to perform a normalization step. This normalization step basically maps the the computational power of the in-house servers into the computational power of the cloud instances that will replace these servers. To perform the scaling step, the model takes as input scaling factors between the in-house servers and the cloud instances that can replace them. One way to obtain these coefficients is to run a standard benchmark, for example commercial SPEC [7] (or a more lightweight benchmark such as *nbench* [8]) both on the in-house servers and on the cloud instances, and to use the ratio between the in-house and the cloud instances measurements as a scaling factor. The precision of this scaling method is not perfect, but it gives a reasonable estimate. In the experiments reported later, we used this method to map between the computational power of a set of cluster nodes and the declared power of the Amazon EC2 instances expressed in terms of Compute Units (CU), which is EC2's unit for expressing the power of their virtual machine instances. In this context, the normalization step leads to CPU demand curves that plot the number of CUs required at each point in the planning period.

The CPU demand curve can take an arbitrary shape. In order to handle any type of curve while keeping the estimation model computationally practical, we discretize the CPU demand curve using the following procedure. First we identify the local minima in the curve. Each interval between two consecutive local minima becomes one period in the discretization. Each time period is guaranteed to contain one peak. We then construct a stepwise function by mapping each time period to the maximum value observed during that period.[5]

In order to avoid the computational burden of having to deal with a very large number of periods in the discretization, the model allows one to optionally fix a minimum discretization threshold. This threshold can be set for example to one month. In this case, each period in the discretization is assured to be at least one month long.

Once we have computed a discretized demand curve for CPU, we discretize the RAM load curve using the same periods as the CPU curve. The resulting discretized RAM and CPU demand curves for the planning period are then given as input to the estimation model.

Regarding network demand, it is important to distinguish between the amount of data pushed into the cloud (*net_in*) versus the amount of data pulled out of the cloud (*net_out*). This distinction is important because different costs apply to each of these components. The model takes as input the total amount of *net_in*) and *net_out* over the entire planning period. These two numbers can be derived by taking the observed traffic in and out of each in-house server over a season, and replicating this observed traffic as required to fit the length of the planning period. Finally, the storage demand curve can be derived based on the amount of data stored in the hard-drives attached to in-house servers over a season. The storage demand curve needs to be discretized according to the

[5] Alternatively, instead of using the maximum value, we could use the 95% percentile depending on the required level of service level agreement.

time unit that the cloud provider uses for storage billing. In the case of Amazon EC2 (Elastic Block Storage) or S3 (Simple Storage Service), this means that the storage demand curve should be discretized by the month.

2.2 Output

The model produces three outputs: a virtual machine usage function, a reservation schedule, and the total cost. The virtual machine usage function is a function indicating how many virtual machine instances (and of which types) are needed at each point in time in order to satisfy the input demand curves. We call this latter function the *cloud usage plan*. Cloud providers generally offer discounts on the cost of renting a virtual machine instance, if the instance is reserved in advance for a period of 1-3 years. For this, a certain reservation fee is paid upfront. The model takes these discounts into account by computing an optimal schedule for reserving virtual machine instances during the planning period. Accordingly, the second output of the model is a function indicating how many machines need to be reserved and when they should be reserved. This latter time function is called the *reservation schedule*. The reservation schedule is represented as a set of tuples (n, p, t, d) stating that n virtual machines of type p need to be reserved starting at time t for a period of d time units. In the case of Amazon, $d = 1$ year or $d = 3$ years are the only options. However, the model is general enough to accommodate other options. Based on the virtual machine usage function and the optimal reservation schedule, we then calculate the total cost of replacing in-house servers with cloud servers for the given demand curves. This leads to the third output of the model.

2.3 Internal Details of the Model

The key step of the proposed cost estimation model is to transform the CPU demand curve and the RAM demand curve into a cloud usage plan and a reservation schedule that minimize the total cost. To tackle this optimization problem, we formulate it in terms of linear programming.

We recall that these demand curves are discretized into n disjoint time periods t_i such that $T = \bigcup_{i=1}^{n} t_i$. For each time period t_i, the computing demand curve states how many compute units are required during time period t_i while the memory demand curve states how many MB of RAM are required during the time period in question.

For notational convenience, we define $length(t)$ as the function that returns the duration of time period t using the time granularity for which cloud services are billed. In the case of Amazon EC2, the length of a time period is expressed in number of hours, since EC2 instances are billed on an hourly basis. Also, for the sake of simplicity we assume that each time period t_i in the discretization is contained in exactly one year of the planning period.

We recall that a linear programming model is composed of parameters, variables and an objective function. The linear optimizer seeks to minimize or maximize the objective function by assigning values to the variables of the problem. The parameters are set before the optimizer is invoked.

The parameters of the linear programming model are derived from the input of the cost estimation model. They include:

- D_t^{CPU}, D_t^{RAM} – the value of the demand curve for CPU and RAM during a time period t.
- RC_d^p – the cost of reserving a virtual machine instance of type p for a time period of length d. In the case of Amazon EC2, there are three types of instances, namely small (S), large (L) and extra-large (XL), and we would have one parameter RC_d^p for each of these three types and for each possible reservation length (i.e. $p \in \{S, L, XL\}$ and $d \in \{1, 3\}$. We assume these cost parameters do not vary across the planning period.
- RUC^p – the unit cost of using a reserved virtual machine instance of type p.
- ODC^p – the unit cost of using an instance "on demand", i.e. without prior reservation.
- C^p is the capacity of virtual machine instance of type p. In the case of EC2, this parameter is measured in compute units for CPU and in Mega-Bytes for RAM.

The variables correspond to the number of instances of different types to be reserved and used at each period:

- $R_{t,d}^p$ is the amount of virtual machine instances of type p reserved at the beginning of time period t for a duration of d . In the case of EC2, d is 1 or 3 years.
- RU_t^p is the number of reserved instances of type p that are in use during time period t (Reserved and Used).
- ODU_t^p is the number of on demand virtual machine instances used during time period t. (On demand and used)
- U_t^p is the total number of virtual machine instances of type p that are in use during time period t, i.e. $U_t^p = RU_t^p + ODU_t^p$.

The *objective function* consists of three components: (i) the total cost of reserving instances throughout the planning period; (ii) the total cost of using reserved instances; and (iii) the total cost of using on-demand instances. Each component is defined as a sum across all time periods $t \in T$ in the discretized demand curves, all types of instances p, and all allowed reservation lengths d.

$$\begin{aligned} &\sum R_{t,d}^p \times {RC_d}^P \ + \\ &\sum RU_t^p \times RUC^p \times length(t) \ + \\ &\sum ODU_t^p \times ODC^p \times length(t) \end{aligned}$$

Finally, the linear programming model comprises two sets of constraints:

– Needed capacity constraints for the reservable resource (for each period t):

$$\Sigma^p(U_t^p \times C_t^p) \geq D_t$$

– Reservation capacity constraint (for each type of instance p and for each t):

$$RU_t^p \leq \sum R_{td}^p$$

The needed capacity constraint for a given time period t states that the sum of "used reserved instances" and "used on-demand instances" must meet the required capacity during time period t. The reservation capacity constraint states that the number of "used reserved instances" during every time period must be at most equal to the number of reserved instances during that time period. In other words, we cannot use more reserved instances than the number of current reservations.

3 Evaluation

For evaluating our model we analyzed the historical metrics of the NICPB WNB cluster and tried to estimate how reasonable it would be to let this cluster run as cloud resources in Amazon EC2. The NICPB WNB cluster is mainly used for computationally intensive applications in the area of scientific computing. It runs mainly Monte Carlo simulations and data analysis applications on datasets from the high-energy physics, computational linguistics and chemistry. These are usually batch processes. Therefore, the class of applications we are looking at are embarassingly parallel which allows us to assume linear scaling in regard to the processor power available. We implemented the data aggregation and the model in Python and used COIN [9] linear optimizer for finding the best EC2 configuration.

3.1 Dataset Description

We used real load and memory usage data obtained over a 10 month period from March 2009 till December 2009. For the discretization step we used RRDTool [10] and aggregated usage metrics by month. The resulting discretized curve is shown in Figure 1. The objective was to estimate the cost of running the same load on Amazon EC2 during a 10 month period.

As explained in Section 2, we apply a normalization step to match the source platform with the destination platform on the cloud. For this normalization, we make use of SPECint [11]. The source system consists of 30 IBM System x3550 machines with dual Intel Xeon E5410 Quad Core processors per machine (8 cores per machine). It is the origin of the dataset used for our calculations. SPECint shows a base value for it for one core of 18.7. We selected this value from the nearest measured system: IBM System x3400, Intel Xeon E5410.

The target system consists of EC2 instances of different types. As mentioned earlier, the power of these instances is measured in terms of Elastic Compute

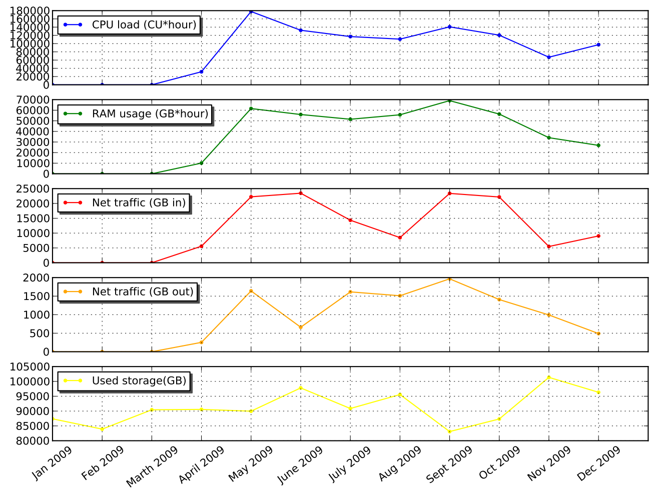

Fig. 1. CPU, RAM, network and storage demand curves used for the experiments (discretized, but non-normalized).

Units (ECU). A standard small instance corresponds to 1 ECU, a standard large instance corresponds to 4 ECU and a high-CPU extra large instance corresponds to 20 ECU. The guaranteed performance of an ECU is defined as the CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor. This maps to a SPECint base value of 5.75 – taking as reference the 1.2 GHz value in SPECint 2006 of an HP BladeSystem bc2000 Blade PC. However, our tests show that the actual processing power of a standard small instance was an Opteron with 2.6 GHz, which maps to a SPECint base value of 12.6.[6] For pragmatic reasons, we chose this latter value. Accordingly, we scaled the CPU load curve by multiplying it by 18.7/12.6 and we used the resulting curve as the CPU demand curve.

Network traffic in Figure 1 is measured in GB of incoming and outgoing traffic. This curves were used to compute the *net_in* and *net_out* cost parameters of the model. Amazon offers two main ways of purchasing storage: Simple Storage Service (S3) and Elastic Block Storage (EBS). In this scenario, we use S3 as it fits the requirements of the in-house applications: downloading the input file, processing it, uploading the results to the storage system.

We gathered the cost data from the EC2 price-list for the European region –“ EU Ireland”.[7] The relevant prices for this scenario are given in table 3.1.

[6] Based on Cpuinfo.
[7] http://aws.amazon.com/ec2/

	standard S	high-CPU M	high-memory XL
On-Demand instance (per hour)	\$0.095	\$0.19	\$0.57
Reserved instance (1 year) fee	\$227.50	\$455	\$1325
Reserved instance (3 year) fee	\$350	\$700	\$2000
Reserved instance (per hour)	\$0.04	\$0.08	\$0.24

Table 1. Price-list for relevant Amazon EC2 Linux instances for EU-Ireland region as of 6.06.2010.

3.2 Results

After running the optimization over the scenario mentioned, we have got the following results. Offering the same kind of service on AWS as the NICPB WNB cluster does, it would cost roughly \$980k. These costs comprise of \$844.61k for CPU usage (86.18%), \$13415.5 (1.37%) for data download, \$1579 (0.16%) for outgoing communications and \$120.40k (12.29%) for data storage. These numbers illustrate the fact that the major operational costs come from the usage of EC2. Note also that the numbers do not include the input data migration to the cloud. This migration is needed as it allows to cut on the network costs: data transfers inside the AWS cloud are free.

In a similar analysis of running scientific computing application on clouds, [4] observed that the cost of running an application on the public cloud depends heavily on the usage profile of the application. Our model shows that if we use cloud-based storage, the main expense for the HPC applications on Amazon cloud is the computational part.

4 Related Work

Interest for the comparison of in-house computing versus cloud computing is growing in the recent literature [12–14]. For instance the technical report by the University of California at Berkeley [12] clarifies terms, provides simple formulas to quantify the comparison between of cloud and in-house computing. The authors provide a short case study (house computing versus cloud computing) based on the general characteristics such as electricity costs, network bandwidth, operations, software and hardware available. In general this paper provides a lot of information both for the infrastructure of cloud computing and also proposes a number of issues that still remain research questions for the future. This work lacks a systematic view-cut for the targeted cloud versus in-house computing comparison. It does not propose any model that could be used to predict the costs on the cloud.

In [13], a framework for analyzing economic and technical aspects of cloud versus in-house computing is presented. The first part of the framework takes into account the criteria relevant both for the cloud and for the in-house computing. These include (*i*) the business domain, (*ii*) the business objectives, and (*iii*) the demand behaviour. The second part of the framework differentiates between criteria specific for cloud computing (storage capacity, processing power, data transfer number) and for in-house computing (facility, energy, cooling infrastructure, cables, servers, network fees, salaries). In general the assessment of the in-house infrastructure is suggested following the general TCO models [14]. Although conceptually useful, the estimation framework [13] still remains very abstract for the systematic assessment and comparison of the in-house infrastructure and cloud computing.

In [14] Spellmann *et al.* present a case study and a set of criteria that suggest some means to compare cloud computing against in-house computing. The possible criteria include infrastructure scalability, physical resource utilization, energy consumption, network infrastructure, security, software licensing, and facility costs. Some of these criteria are elaborated in the case study provided in the paper. Although the analysis covers different cloud and in-house criteria, this work [14] does not provide any systematic method not framework for the cost assessment.

Similarly for *operational/business costs* and for *risks and their costs*, little is elaborated for the *personnel cost* estimation. The majority of the studies [12] [14] discuss that there are maintenance, business continuity, and other costs, however these are not further detailed. Finally the identified studies discuss little about the *costs changes* during the changing setting (when either the costs of cloud providers and/or SaaS providers are changing) and *migration* (forward and backward in-house and cloud computing) cost. In [12] authors discuss different characteristics of the cloud computing architectures, namely Amazon EC2, Microsoft Azure and Google AppEngine. However the study provides no comparison between these architectural models.

Finally we want to mention another paper that discusses a model that is at the same time broader than ours and still less deep. It is the one by J. Strebel [15]. The paper gives a very broad overview of all different cost factors involved when running applications in a Grid environment. It provides a formula to estimate the costs on the cloud taking many cost factors into consideration. An optimization to get the price minimal instance configuration is not implemented in the model. Further on it does not account for the possibility to reserve instances that is offered by AWS.

The considered publications that investigate different high-level criteria that directly or indirectly describe the *organisations operational* and *business* costs. The major limitation in all works is the lack of the concrete low-level models. The most complete analysis (including two-three level criteria list) is provided in [14], however even here the authors say that the concrete application of the criteria depends on the specific organizational profile.

5 Conclusions and future work

This paper presented our cloud computing economic model that can predict whether or when it is optimal to move an existing application setup to the public cloud. The model is based on the cost optimization function using linear programming principles. We also showed with our model and analysis that it is not necessarily beneficial to move the scientific computing clusters to the clouds for cost reasons only. In the future we will be looking at different load types specific for certain applications. We are considering to look at e-commerce, social networking, and travel sites for this purpose. We will select those examples due to the nature of their yearly traffic pattern.

As a next step, we are working on a simulation environment for business transaction oriented websites. The problem in this field is to determine how many compute units (or VM instances) are needed in order to cope with a given workload. Preliminary empirical studies have suggested that this problem is non-trivial because it depends on the type of the application and the configuration employed (e.g. where the load balancer is placed). Accordingly we plan to use well-known benchmarks (eg. TPC-W, rubis) that are representatives of specific domains (e.g. E-commerce) under different configurations to derive a function that maps the workloads to required number of instances. This is also in line with the work undertaken in cloudXplorer.

We are also aware of the studies like [16] that propose alternative architectures and/or dynamically switching among several configurations depending on workload and traffic pattern, as they argue that none of the configurations of cloud components can excel in all traffic scenarios. However we would presume that a proper cost optimization model can be drawn for any of the configurations. We will base our tests on TPC-W, the rubis auction shop simulation system, and a special configuration of the open source shop software opencart[8] to exactly measure scalability of the different system components under different kinds of loads and traffic.

It will be possible to extend our predictions even forward, when we relate the traffic of a web site with the expected load corresponding to the type of the web site. With this we can extend our prediction to E-commerce sites like Amazon.com or travel sites like Lastminute.com or social networking sites like Facebook.com. All cites have a unique traffic pattern. We want to investigate if it is possible to establish a set of rules that can make a reliable suggestion whether is makes financial sense (or not) to move an in house IT infrastructure to the cloud purely based on the usage pattern of the application.

Acknowledgments This research is part of the EUREKA Project 4989 (SITIO) co-funded by the European Regional Development Funds via Enterprise Estonia.

References

1. NucleusResearch: ROI case study Google apps TVR communications. http://www.google.com/apps/intl/en/business/case_studies/tvr.pdf
2. Jinzy, Z.: IBM cloud computing case studies (2010)
3. Akram, A.: Making the case for the economic benefits of cloud computing: Case study: Bayt.com

[8] `http://www.opencart.com`

4. Deelman, E., Singh, G., Livny, M., Berriman, B., Good, J.: The cost of doing science on the cloud: the montage example. In: Proceedings of the 2008 ACM/IEEE conference on Supercomputing. (2008) 1–12
5. Massie, M.L., Chun, B.N., Culler, D.E.: The ganglia distributed monitoring system: design, implementation, and experience. Parallel Computing **30**(7) (July 2004) 817–840
6. Collectd: The system statistics collection daemon. http://collectd.org
7. SPEC: The standard performance evaluation corporation. http://www.spec.org
8. nbench: Linux/unix port of the bytemark benchmark. http://www.tux.org/ mayer/linux/bmark.html
9. Lougee-Heimer, R.: The Common Optimization INterface for Operations Research: Promoting open-source software in the operations research community. IBM Journal of Research and Development **47**(1) (2003) 57–66
10. Oetiker, T.: RRDtool. http://people.ee.ethz.ch/oetiker/webtool/rrdtool
11. Standard Performance Evaluation Corporation (SPEC): Specint
12. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R.H., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I., Zaharia, M.: Above the Clouds: A Berkeley View of Cloud Computing. Technical report UCB/EEC-2009-28, Electrical Engineering and Computer Sciences, University of California at Berkeley (2009)
13. Klems, M., Nimis, J., Tai, S.: Do Clouds Compute? A Framework for Estimating the Value of Cloud Computing. In: Proceeding of the 7th Workshop on E-Business, WEB 2008, Springer Berlin Heidelberg (2009) 110–123
14. Spellmann, A., Gimarc, R., Preston, M.: Leveraging the Cloud for Green IT: Predicting the Energy, Cost and Performance of Cloud Computing. In: CMG'09 International Conference. (2009)
15. Strebel, J.: Cost optimization model for business applications in virtualized grid environments. In: Grid Economics and Business Models. (2009) 74–87
16. Liu, H., Wee, S.: Web server farm in the cloud: Performance evaluation and dynamic architecture. In: Cloud Computing. (2009) 369–380