Cost Minimization for Computational Applications on Hybrid Cloud Infrastructures

Maciej Malawski [a,b] Kamil Figiela [b] Jarek Nabrzyski [a]

[a] *University of Notre Dame, Center for Research Computing, Notre Dame, IN 46556, USA*

[b] *AGH University of Science and Technology, Department of Computer Science, Mickiewicza 30, 30-059 Kraków, Poland*

Abstract

We address the problem of task planning on multiple clouds formulated as a mixed integer nonlinear programming problem (MINLP). Its specification with AMPL modeling language allows us to apply solvers such as Bonmin and Cbc. Our model assumes multiple heterogeneous compute and storage cloud providers, such as Amazon, Rackspace, GoGrid, ElasticHosts and a private cloud, parameterized by costs and performance, including constraints on maximum number of resources at each cloud. The optimization objective is the total cost, under deadline constraint. We compute the relation between deadline and cost for a sample set of data- and compute-intensive tasks, representing bioinformatics experiments. Our results illustrate typical problems when making decisions on deployment planning on clouds and how they can be addressed using optimization techniques.

Key words: Distributed systems, Cloud computing, Constrained optimization

1. Introduction

In contrast to already well established computing and storage resources (clusters, grids) for the research community, clouds in the form of infrastructure-as-a-service (IaaS) platforms (pioneered by Amazon EC2) provide on-demand resource provisioning with a pay-per-use model. These capabilities together with the benefits introduced by virtualization, make clouds attractive to the scientific community [1]. In addition to public clouds such as Amazon EC2 or Rackspace, private and community cloud installations have been deployed for the purpose of scientific projects, e.g. FutureGrid [1] or campus-based private cloud at Notre Dame [2]. As a result, multiple deployment scenarios differing in costs and performance, coupled together with new provisioning models offered by clouds make the problem of resource allocation and capacity planning for scientific applications a challenge.

The motivation for this research comes from our previous work [2,3], in which we run experiments with

Email addresses: malawski@agh.edu.pl (Maciej Malawski), naber@nd.edu (Jarek Nabrzyski).

[1] http://futuregrid.org

[2] http://www.cse.nd.edu/~ccl/operations/opennebula/

compute-intensive bioinformatics application on a hybrid cloud consisting of Amazon EC2 and a private cloud. The application is composed of a set of components (deployed as virtual machines) that communicate using a queue (Amazon SQS) and process data that is stored on a cloud storage (Amazon S3). The results of these experiments indicate that clouds do not introduce significant delays in terms of virtualization overhead and deployment times. However, multiple options for placement of application components and input/output data, which differ in their performance and costs, lead to non-trivial resource allocation decisions. For example, when data is stored on the public cloud, the data transfer costs between storage and a private cloud may become large enough to make it more economical to pay for compute resources from the public cloud than to transfer the data to a private cloud where computing is cheaper.

In this paper, we address the resource allocation problem by applying the optimization techniques using AMPL modeling language [4], which provides access to a wide range of ready to use solvers. Our model assumes multiple heterogeneous compute and storage cloud providers, such as Amazon, Rackspace, ElasticHosts and a private cloud, parameterized by costs and performance. We also assume that the number of resources of a given type in each cloud may be limited, which is often the case not only for private clouds, but also for larger commercial ones. The optimization objective is the total cost, under deadline constraint. To illustrate how these optimization tools can be useful for planning decisions, we analyze the relations between deadline and cost for different task and data sizes, which are close to our experiments with bioinformatics applications.

The main contributions of the paper are the following:

- We formulate the problem of minimization of cost of running computational application on hybrid cloud infrastructure as a mixed integer nonlinear programming problem and its specification with AMPL modeling language.
- We evaluate the model on scenarios involving limited and unlimited public and private cloud resources, for compute-intensive and data-intensive tasks, and for a wide range of deadline parameters.
- We discuss the results and lessons learned from the model and its evaluation.

The paper is organized as follows: after discussing the related work in Section 2, we introduce the details and assumptions of our application and infrastructure model in Section 3. Then, in Section 4 we formulate the problem using AMPL by specifying the variables, parameters, constraints and optimization goals. Section 5 presents the results we obtained by applying the model to the scenarios involving multiple public and private clouds, overlapping computation and data transfers, and identifying special cases. In section 6 we provide a sensitivity analysis of our model and show how such analysis can be useful for potential users or computing service resellers. In section 7 we estimate how our model behaves if the task sizes are not uniform and change dynamically. The conclusions and future work are given in Section 8.

2. Related work

The problem of resource provisioning in IaaS clouds has been recently addressed in [5] and [6]. They typically consider unpredictable dynamic workloads and optimize the objectives such as cost, runtime or utility function by autoscaling the resource pool at runtime. These approaches, however, do not address the problem of data transfer time and cost, which we consider an important factor.

Integer programming approach has been applied to the optimization of service selection for activities of QoS aware grid workflows [7]. On the other hand, in our model we assume the IaaS cloud infrastructure, while the objective function takes into account costs and delays of data transfers associated with the tasks.

The cost minimization problem on clouds addressed in [8] uses a different model from ours. We impose a deadline constraint and assume that the number of instances available from providers may be limited. To satisfy these constraints, the planner has to choose resources from multiple providers. Our model also assumes that VM instances are billed per hour of usage.

3. Model

3.1. *Application model*

The goal of this research is to minimize the cost of processing a given number of tasks on a hybrid cloud platform, as illustrated in Fig. 1. We assume that tasks are independent from each other, but they have identical computational cost and require a constant amount of data transfer.

The assumption of homegeneous tasks can be justified by the reason that there are many examples of scientific applications (e.g. scientific workflows or large parameter sweeps) that include a stage of a high number of parallel nearly identical tasks. Such examples can be found e.g. in typical scientific workflows executed using Pegasus Workflow Management system, where e.g. CyberShake or LIGO workflows have a parallel stage of nearly homogeneous tasks [9]. Other examples are Wien2K and ASTRO workflows that consist of iteratively executed parallel stages comprising homogeneous tasks[10]. Due to the high number of parallel branches, these stages accumulate the most significant computing time of the whole application, so optimization of the execution of this stage is crucial. Moreover, if the tasks are not ideally homogeneous, it is possible to approximate them using a uniform set of tasks with the mean computational cost and data sizes of the application tasks. Of course, in real execution the actual task performance may vary, so the solution obtained using our optimization method becomes only approximate of the best allocation, and the actual cost may be higher and deadline may be exceeded. In order to estimate the quality of this approximation, we evaluate the impact of dynamic task runtime and non-uniform tasks in section 7.

We assume that for each task a certain amount of input data needs to be downloaded, and after it finishes, the output results need to be stored. In the case of data-intensive tasks, the transfers may contribute a significant amount of total task run time.

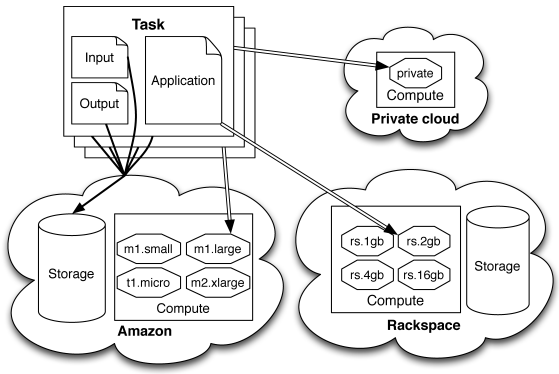


Figure 1. The model of application and infrastructure

3.2. *Infrastructure model*

Two types of cloud services are required to complete tasks: storage and virtual machines. Amazon S3 and Rackspace Cloud Files are considered as examples of storage providers, while Amazon EC2, Rackspace, GoGrid and ElasticHosts represent computational services. In addition, the model includes a private cloud running on own hardware. Each cloud provider offers multiple types of virtual machine instances with different performance and price.

For each provider the number of running virtual machines may be limited. This is mainly the case for private clouds that have a limited capacity, but also the public clouds often impose limits on the number of virtual machines. E.g. Amazon EC2 allows maximum of 20 instances and requires to request a special permission to increase that limit.

Cloud providers charge their users for each running virtual machine on an hourly basis. Additionally, users are charged for remote data transfer while local transfer inside provider's cloud is usually free. These two aspects of pricing policies may have a significant impact on the cost of completing a computational task.

Cloud services are characterized by their pricing and performance. Instance types are described by price per hour, relative performance and data transfer cost. To assess the relative performance of clouds it is possible to run application-specific benchmarks on all of them, or to use publicly available cloud benchmarking services, such as Cloud-

Harmony [3] . CloudHarmony defines performance of cloud instances in the units named CloudHarmony Compute Units (CCU) as similar to Amazon EC2 Compute Unit (ECU), which are approximately equivalent to CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor. Storage platforms include fees for data transfer. Additionally, a constant fee per task may be present, e.g. price per request for a queuing service [4] .

Our model includes all the restrictions that were mentioned above.

4. Problem formulation using AMPL

To perform optimization of the total cost, Mixed Integer Non-Linear Problem (MINLP) is formulated and implemented in A Mathematical Programming Language (AMPL) [4]. AMPL requires to specify input data sets and variables to define the search space, as well as constraints and objective function to be optimized.

4.1. *Input data*

The formulation requires the following input sets, which represent the cloud infrastructure model:

- $S = \{s3, cloudfiles\}$ – defines available cloud storage sites,
- $P = \{amazon, rackspace, \dots\}$ – defines possible computing cloud providers,
- $I = \{m1.small, \dots, gg.1gb, \dots\}$ – defines instance types,
- $PI_p \subset I$ – instances that belong to provider P_p,
- $LS_s \subset P$ – compute cloud providers that are local to storage platform S_s.

Each instance type I_i is described by the following parameters:

- p_i^I – fee in \$ for running instance I_i for one hour,
- ccu_i – performance of instance in CloudHarmony Compute Units (CCU),
- p_i^{Iout} and p_i^{Iin} – price for non-local data transfer to and from the instance, in \$ per MiB.

Storage sites are characterized by:

- p_s^{Sout} and p_s^{Sin} characterize price in \$ per MiB for non local data transfer.

Additionally we need to provide data transfer rates in MiB per second between storage and instances by defining function $r_{i,s} > 0$.

We assume that the computation time of a task is known and constant, this also applies to input and output data size. We also assume that tasks are atomic (non divisible). Computation is characterized by the following parameters:

- A^{tot} – count of tasks,
- t^x – execution time in hours of one task on 1 CCU machine,
- d^{in} and d^{out} – data size for input and output of one task in MiB,
- p^R – price per request for queuing service,
- t^D – total time for completing all tasks in hours (deadline).

4.2. *Auxiliary parameters*

The formulation of the problem requires a set of precomputed parameters which are derived from the main input parameters of the model. The relevant parameters include:

$$t_{i,s}^{net} = \frac{d^{in} + d^{out}}{r_{i,s} \cdot 3600} \tag{1}$$

$$t_{i,s}^{u} = \frac{t^x}{ccu_i} + t_{i,s}^{net} \tag{2}$$

$$c_{i,s}^{t} = (d^{out} \cdot (p_i^{Iout} + p_s^{Sin}) + d^{in} \cdot (p_s^{Sout} + p_i^{Iin})) \tag{3}$$

$$a_{i,s}^{d} = \lfloor \frac{t^D}{t_{i,s}^{u}} \rfloor \tag{4}$$

$$t_{i,s}^{q} = \lceil t_{i,s}^{u} \rceil \tag{5}$$

$$a_{i,s}^{q} = \lfloor \frac{t_{i,s}^{q}}{t_{i,s}^{u}} \rfloor \tag{6}$$

$$t_{i,s}^{d} = \lceil \lfloor \frac{t^D}{t_{i,s}^{u}} \rfloor \cdot t_{i,s}^{u} \rceil \tag{7}$$

[3] `http://blog.cloudharmony.com/2010/05/what-is-ecu-cpu-benchmarking-in-cloud.html`

[4] e.g. Amazon SQS

– $t^{net}_{i,s}$ – *transfer time*: time for data transfer between I_i and S_s,
– $t^{u}_{i,s}$ – *unit time*: time for processing a task on instance I_i using storage S_s that includes computing and data transfer time (in hours),
– $c^{t}_{i,s}$ – cost of data transfer between instance I_i and storage S_s,
– $a^{d}_{i,s}$ – number of tasks that can be done on one instance I_i when using storage S_s running for t^D hours,
– $t^{q}_{i,s}$ – *time quantum*: minimal increase of instance running time that is sufficient to increase the number of processed tasks, rounded up to full hour,
– $a^{q}_{i,s}$ – number of tasks that can be done in $t^{q}_{i,s}$ hours,
– $t^{d}_{i,s}$ – *instance deadline*: number of hours that can be effectively used for computation.

4.3. *Variables*

Variables that will be optimized and define the solution space are listed below:

– N_i – number of instances of type I_i to be deployed,
– A_i – number of tasks to be processed on instances of type I_i,
– D_s – 1 iif S_s is used, otherwise 0; only one storage may be used,
– R_i – number of remainder (tail) hours for instance I_i,
– H_i – 1 iif $R_i \neq 0$, otherwise 0.

Fig. 2 illustrates how the tasks are assigned to multiple running instances. The tasks are atomic and there is no checkpointing or preemption. Even though all the tasks have the same computational cost, their total processing time depends on performance of the VM type and data transfer rate. Moreover, the users are charged for the runtime of each VM rounded up to full hours. In our model the tasks are assigned to the instances in the following way. First, in the process of optimization A_i tasks are assigned to the instance I_i. This gives N_i instances of type I_i running for $t^{d}_{i,s}$ hours. Remaining tasks are assigned to an additional instance running for R_i hours. We will refer to them as tail hours.

In order to enforce provider's instance limit, H_i is introduced which indicates if I_i has any tail hours. E.g. in Fig. 2 instances of type 1 have 3 tail hours,

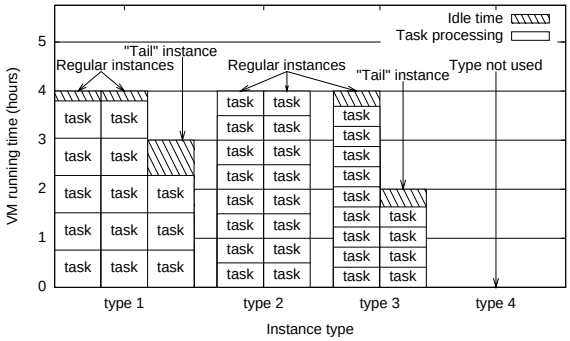


Figure 2. Task scheduling policy

and instances of type 2 have no tail hours.

4.4. *Formulation of objectives*

Cost of running a single task which includes the cost of VM instance time required for data transfer and task computing time, together with data transfer costs and request cost, can be described as:

$$\begin{aligned}\left(t^{net} + t^{u}\right) \cdot p^{I} + \\ d^{in} \cdot \left(p^{Sout} + p^{Iin}\right) + \\ d^{out} \cdot \left(p^{Iout} + p^{Sin}\right) + \\ p^{R}.\end{aligned}$$

The objective function represents the total cost of running multiple tasks of the application on the cloud infrastructure and it is defined as:

$$\begin{aligned}\underset{\text{total cost}}{\text{minimize}} \sum_{i \in I} \Big(\Big(\sum_{s \in S} D_s \cdot t^{d}_{i,s} \cdot N_i + R_i \Big) \cdot p^{I}_{i} \\ + A_i \cdot \Big(p^{R} + \sum_{s \in S} D_s \cdot c^{t}_{i,s} \Big) \Big)\end{aligned} \tag{8}$$

subject to the constraints:

$$\underset{i\in I}{\forall} A_i \in \mathbb{Z} \wedge 0 \leq A_i \leq A^{tot} \tag{9}$$

$$\underset{s\in S}{\forall} D_s \in \{0,1\} \tag{10}$$

$$\underset{i\in I}{\forall} N_i \in \mathbb{Z} \wedge 0 \leq N_i \leq n_i^{Imax} \tag{11}$$

$$\underset{i\in I}{\forall} R_i \in \mathbb{Z} \wedge 0 \leq R_i \leq t^D - 1 \tag{12}$$

$$\underset{i\in I}{\forall} H_i \in \{0,1\} \tag{13}$$

$$\underset{i\in I}{\forall} A_i \geq \sum_{s\in S}(N_i \cdot a_{i,s}^d) \cdot D_s \tag{14}$$

$$\underset{i\in I}{\forall} A_i \leq \sum_{s\in S}(N_i \cdot a_{i,s}^d + \max(a_{i,s}^d - 1, 0)) \cdot D_s \tag{15}$$

$$\underset{i\in I}{\forall} R_i \geq \sum_{s\in S}(A_i - N_i \cdot a_{i,s}^d) \cdot t_{i,s}^u \cdot D_s \tag{16}$$

$$\underset{i\in I}{\forall} R_i \leq \sum_{s\in S}(A_i - N_i \cdot a_{i,s}^d + a_{i,s}^q) \cdot t_{i,s}^u \cdot D_s \tag{17}$$

$$\sum_{i\in I} A_i = A^{tot} \tag{18}$$

$$\sum_{s\in S} D_s = 1 \tag{19}$$

$$\underset{i\in I}{\forall} H_i \leq R_i \leq \max(t^D - 1, 0) \cdot H_i \tag{20}$$

$$\underset{p\in P}{\forall} \sum_{i\in PI_p} (H_i + N_i) \leq n_p^{Pmax} \tag{21}$$

Interpretation of the constraints is the following:

- (9) to (13) define weak constraints for a solution, i.e. they are to ensure that the required variables have the appropriate integer or binary values,
- (14) and (15) ensure that number of instances is adequate to number of assigned tasks; for chosen storage D_s

  $$N_i \cdot a_{i,s}^d \leq A_i \leq N_i \cdot a_{i,s}^d + \max\left(a_{i,s}^d - 1, 0\right)$$

  where the lower bound is given by full allocation of N_i machines and the upper bound includes a fully allocated tail machine,
- (16) and (17) ensure that number of tail hours is adequate to number of remaining tasks, implement $R_i = \left\lceil \left(A_i - N_i \cdot a_{i,s}^d\right) \cdot t_{i_s}^u \right\rceil$,
- (18) ensures that all tasks are processed,
- (19) ensures that only one storage site is selected,
- (20) ensures that R_i has proper value, implements $H_i = \begin{cases} 1 & R_i > 0 \\ 0 & R_i = 0 \end{cases}$.
- (21) enforces providers' instance limits.

Defining the problem in AMPL enables to choose among a wide range of solvers that can be used as backend. The problem itself is MINLP, but can be reduced to Integer Linear Programming (ILP) problem. The nonlinear part of problem comes from storage choice, so by fixing storage provider and running optmization procedure for each storage separately the optimal solution is found.

Initially we used Bonmin [11] solver, but after the model was fully implemented and subject to more tests, it appeared that CBC [12] solver performs better with default options. This results from the fact that Bonmin is a solver designed to solve MINLP problems and uses various heuristics, while CBC uses a branch and bound algorithm tuned for ILP. As the problem is linear and convex, CBC finds global optimum.

The model was optimized so that it should give acceptable results in ∼0.10 seconds [5].

5. Results

To evaluate our model, we first run the optimization process for two scenarios with a private cloud and (1) infinite public clouds (Section 5.1) and (2) finite public clouds (Section 5.2). We also evaluated the effect of possible overlapping of computations and data transfers (Section 5.3). When testing the model under various input parameters, we identified interesting special cases, which are described in Section 5.4. Finally, we performed a sensitivity analysis and discussed its implications and potential usage in Section 6.

We evaluated the model for two types of tasks – data intensive tasks that require relatively large amount of data for short computing time (we assumed 512 MiB of input and 512 MiB of output) and compute intensive tasks that require relatively small amount of data (we assumed 0.25 MiB of input and 0.25 MiB of output). Each case consists of 20,000 tasks, each of them requires 0.1 hour of computing time on a VM with performance of 1 CCU. Two scenarios with infinite and finite public clouds were considered, where as the costs and performance of public clouds we used the data from CloudHarmony bench-

[5] As measured on quad-core Intel i5 machine.

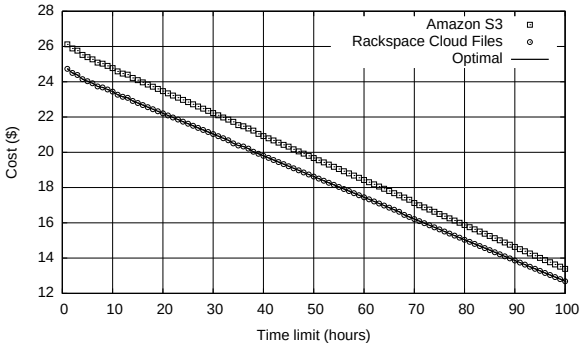


Figure 3. Small data processed on infinite public cloud

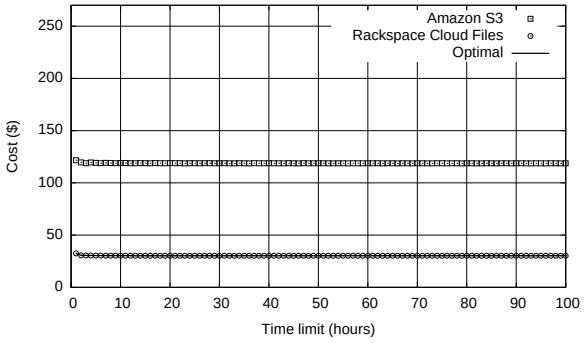


Figure 4. Large data processed on infinite public cloud

marks. This dataset gives the data of 4 compute cloud providers (Amazon EC2, Rackspace, ElasticHosts and GoGrid) and 2 storage providers (Amazon S3 and Rackspace Cloud Files), giving the instance prices between $0.06 and $2.40 per hour and performance between 0.92 and 27.4 CCU. We assumed that the private cloud instances have the performance of 1 CCU and $0 cost. For each scenario we varied the deadline parameter between 5 and 100 hours. Since we considered only two storage providers (S3 and Cloud Files), we run the solver separately for these two parameters.

5.1. *Private + infinite public clouds*

In this scenario we assumed that the private cloud can run a maximum of 10 instances, while the public clouds have unlimited capacity. The results for tasks that require small amount of data are shown in Fig. 3. As the deadline is extended, the total cost linearly drops as expected. As many tasks as possible are run on the private cloud, and for all the remaining tasks the instance type with best price to performance ratio is selected.

This situation changes as data size grows (Fig. 4) – for data intensive tasks the total cost is nearly constant as all the work is done on a public cloud. This results from the fact that data transfer cost to and from the private cloud is higher than the cost of running instances on public cloud. In our case it turns out that the lowest cost can be achieved when using Cloud Files storage from Rackspace, since in our dataset the best instance type in terms of price to performance was available at Rackspace.

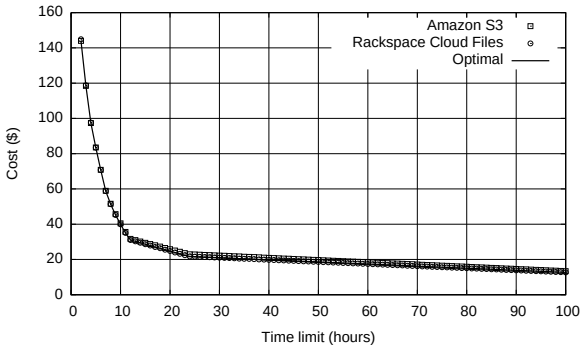


Figure 5. Small data processed on finite public cloud

5.2. *Private + finite public clouds*

If assumption that public clouds are limited is made, situation is not so straightforward (See Fig. 5 and 6). For relatively long deadlines, a single cloud platform for both VMs and storage can be used, which means that the data transfer is free. As the deadline shortens we first observe a linear cost increase. At the point when the selected cloud platform reaches its VM limit, additional clouds need to be used, so we need to begin paying for the data transfer. Therefore the cost begins to increase more rapidly.

This effect is very significant in the case of data intensive tasks (Fig. 6) as the cost growth may become very steep. For example, in our tests the task processing cost in 28 hours was $157.39, in 30 hours it was $131.14 and in 34 hours it was only $30.26. For longer deadlines there was no further decrease. We can also observe that for longer deadlines the Cloud Files storage provider is less expensive for the same reason as it was in Fig. 4. Shorter deadlines, however, require to run more powerful instances from other clouds (Amazon EC2), thus it becomes more

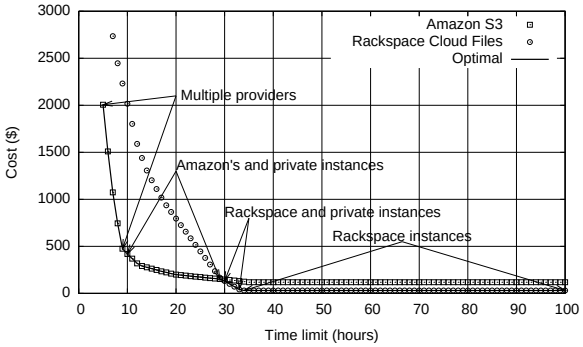


Figure 6. Large data processed on finite public cloud

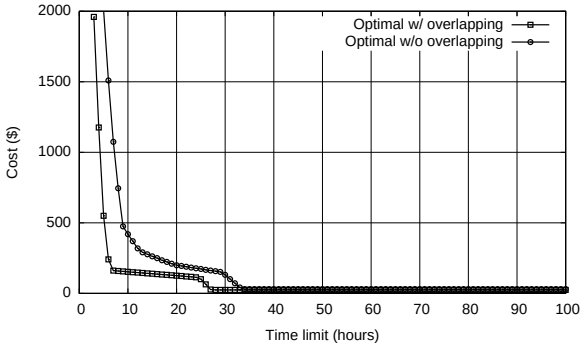


Figure 7. Large data processed on finite public cloud with overlapping computation and data transfer

economical to use its local S3 storage.

5.3. *Overlapping computation and data transfers*

In our model we assume that computation and data transfers are not overlapping. To achieve parallelism of these two processes, the model needs to be modified in the following way. We assumed that the total task computation time is maximum of task execution and data transfer time. Additionally, input for the first task and output of the last task must be transferred sequentially. Equations 2, 4 and 7 are updated for this case as follows:

$$t^u_{i,s} = \max(\frac{t^x}{ccu_i}, t^{net}_{i,s}) \tag{22}$$

$$a^d_{i,s} = \lfloor \frac{(t^D - t^{net}_{i,s})}{t^u_{i,s}} \rfloor \tag{23}$$

$$t^d_{i,s} = \lceil \lfloor \frac{(t^D - t^{net}_{i,s})}{t^u_{i,s}} \rfloor \cdot t^u_{i,s} \rceil \tag{24}$$

Fig. 7 shows results for the same experiment as in Section 5.2 and Fig. 6, but with computing and transfers overlapping. Overlapping reduces total cost as time required for task processing is significantly lower. This is especially true for shorter deadlines when multiple clouds are used, as transfer rates are lower between different clouds comparing to one provider infrastructure.

5.4. *Identifying special cases*

Running a test case with very large tasks which cannot be completed within one hour on largest available instance revealed an interesting model behaviour. Results are shown on Fig. 8. Local minima may occur for certain deadlines thus cost does not increase monotonically with decrease of deadline. This is a consequence of our task scheduling policy, as explained in Section 4.3. In the case of large tasks, the schedule for deadline of 9 hours as seen in Fig. 9a costs less than for deadline of 10 hours as in Fig. 9b. In the first case the total number of VM-hours of the gg-8mb instance is equal to 56, but in the case of deadline = 10 hours the total number is 58, which results in higher cost. This is the result of the policy, which tries to keep VMs running time as close to the deadline as possible. Such policy is based on general observation, that for a given budget it is usually more economical to start less VMs for longer time than to start more VMs for shorter time. However, there are rare cases when such policy may lead to non-optimal solutions. It should be noted, that the solution of the model returned by the solver in this case is optimal, but it is the model itself that does not allow to find the minimum cost.

Moreover, for longer deadlines the cost is a step function – e.g. the cost for deadline = 18 hours is the same as for 14 hours. These two observations suggest that the model could be modified in such a way that the deadline is also a variable with upper bound constraint. Similar result can be achieved in a simple way by solving the current model for multiple deadlines in the neigborhood of the desired deadline and by selecting the best solution.

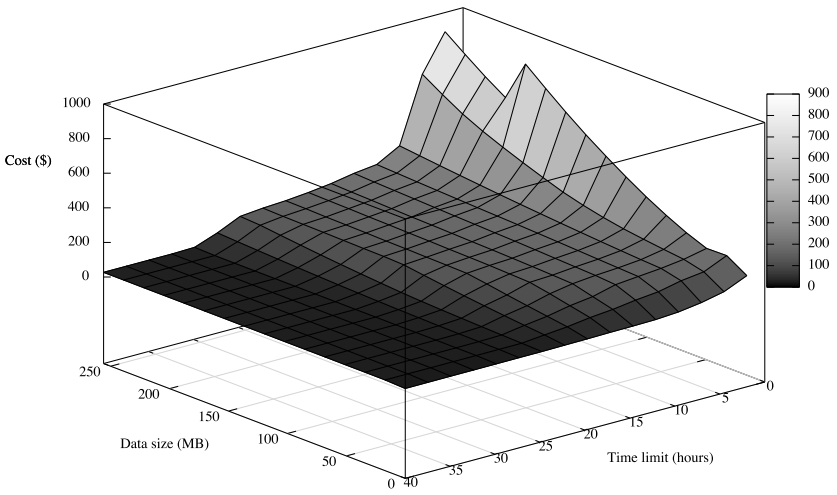


Figure 10. Optimal cost for a wide range of deadline constraints and data sizes.

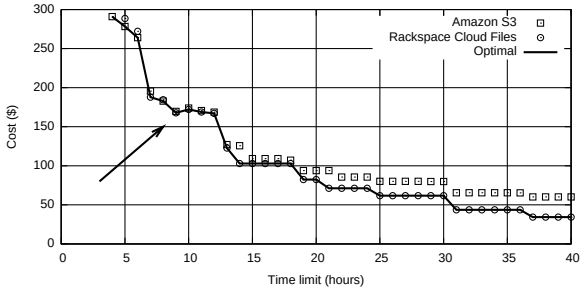
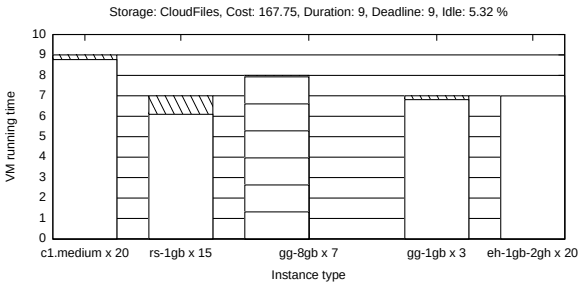
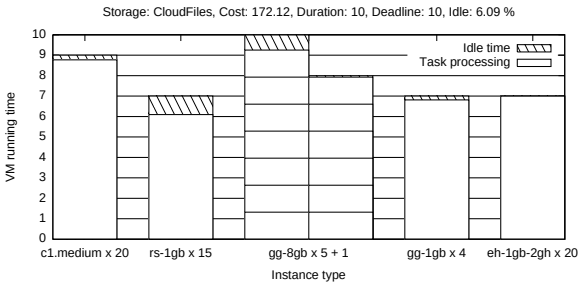


Figure 8. Special case with local minimum for deadline 9



(a) Deadline = 9 hours



(b) Deadline = 10 hours

Figure 9. Task schedule for the identified special case

6. Sensitivity analysis

In order to better assess how our model behaves in response to the changing constraints and for varying input parameters, we performed a sensitivity analysis by sampling a large parameter space. Fig.10 shows the solutions obtained for deadlines ranging from 2 to 40 hours and for input and output data sizes ranging from 0 to 256 MiB. As it can be seen in the plot, for all data sizes the cost increases monotonically with the decrease of the deadline, which confirms that no anomalies are observed. The same data can be also observed as animated plot available as on-line supplement [6] .

Fig. 11 presents these results from a perspective where each curve shows how the cost depends on data size for varying deadlines.

One of the questions that our model can help answer is how the changes in cost are sensitive to the changes

[6] See also `http://youtu.be/FWjgMwLdZW4`

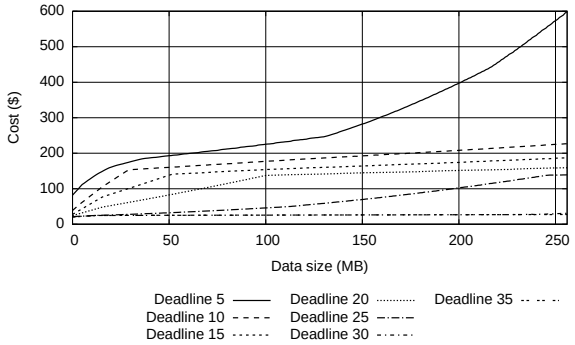
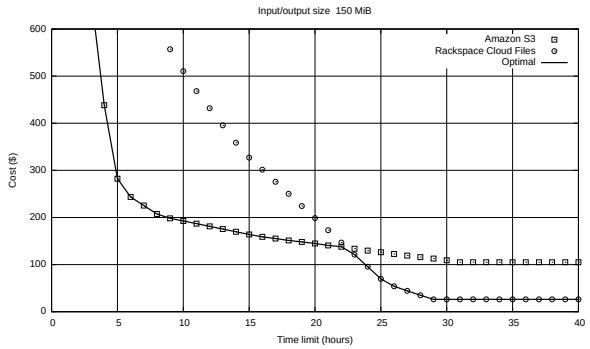
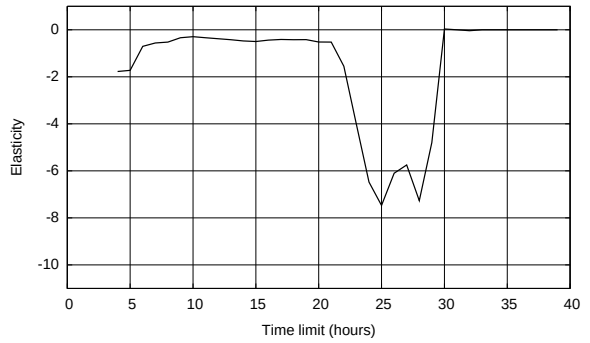


Figure 11. Cost as a function of data size for different deadlines.



(a) Cost as a function of deadline



(b) Elasticity of cost versus deadline

Figure 12. Sensitivity of costs in response to changes in deadline

of deadline. Fig.12 shows the cost function as well as corresponding elasticity, which is computed as: $Ef(x) = \frac{x}{f(x)} f'(x)$, where in this case f is cost and x is deadline. Elasticity gives the information on what is the percentage change of cost in response to the percentage change of deadline. The function has negative values, since the cost decreases with the increase of deadline. It is interesting to see in which ranges the elasticity has larger absolute value, which corresponds to more steep cost function. Here we can see that the elasticity grows for short deadlines and close to the deadline of 25 hours, which is the point where the solution cannot use only the VM instances from most cost-effective clouds and requires more expensive ones to meet the deadline.

Identifying such ranges with high absolute value of elasticity is important for potential users of the cloud system, including researchers (end users) or resellers (brokers). The end user can for example observe that changing the deadline from 40 to 30 hours or even from 20 to 10 hours will not incur much additional cost. However, changing the deadline from 30 to 20 hours is very costly, so it should be avoided. On the other hand, the situation looks different from the perspective of a reseller, who buys cloud resources from providers and offers them to end-users in order to make profit. The reseller can for example offer to compute the given set of tasks in 20 hours for $150 with minimal profit, but it can also offer to complete the same set of tasks in 30 hours for $50. Such price can seem attractive to the end user who pays 1/3 of the price for increasing the deadline by 50%, but it is in fact very beneficial for the reseller, whose profit reaches 100%. Such cost analysis is an important aspect of planning large computing experiments on cloud infrastructures.

7. Impact of dynamic environment

As stated in section 3, we assume that execution time, transfer rate and data access time for all the tasks are constant. However, in real environments the actual runtime of tasks will vary. The goal of the following simulation experiment was to estimate the impact of this runtime variation on the quality of results obtained using our optimization model.

For all the task assignment solutions presented in Fig. 10 we add runtime variations to the task runtimes in the following way. For each task we generate a random error in the range from $-v$ to v using uniform distribution, where v is the runtime variation range. We tested values of $v = 10\%, 20\%, 30\%, 40\%, 50\%$. Such modified task runtimes are then used to calculate the actual runtime and cost of VM. Due to variation of task runtimes, it is possible that some computations may not finish before the deadline (time overrun) and that the

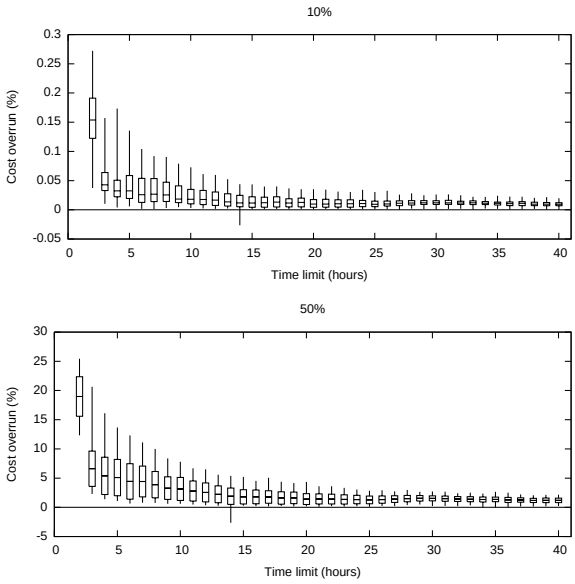


Figure 13. Impact of runtime variation on cost overrun as a function of deadline. Results obtained for random variation of task runtime in the range from −10% to +10% (top) and from −50% to +50% (bottom).

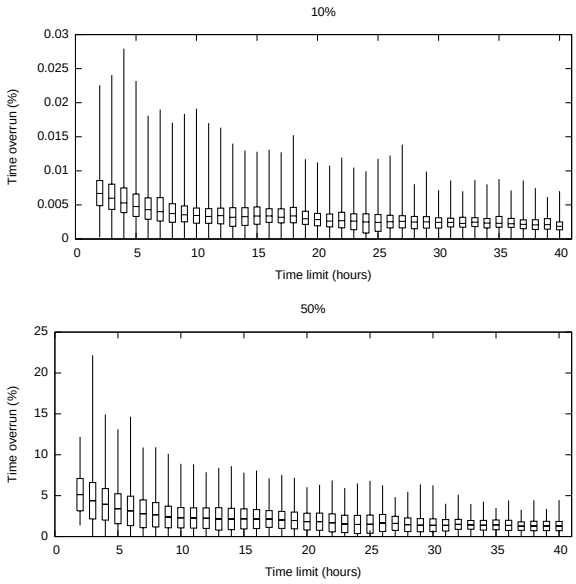


Figure 14. Impact of runtime variation on deadline overrun as a function of deadline. Results obtained for random variation of task runtime in the range from −10% to 10% (top) and from −50% to +50% (bottom).

cost of additional VM-hours may need to be paid (cost overrun).

We assume that task size variations include also variations of data transfer time. We don't take into account variations of data transfer costs, since the transfer costs for each task depend linearly on data size, so the aggregated impact of positive and negative variations cancels to nearly 0.

Fig.13 shows the cost overrun with 10% and 50% of runtime variation range. The boxplots for each deadline represent averaged results for the data sizes from 0 to 256 MiB as in Fig. 10, with box boundaries at quartiles and whiskers at maximum and minimum values. We observe that the highest overruns are for the shortest deadlines, which results from the fact that when each VM instance runs only for a few hours, then the additional hour will incur relatively high additional cost. On the other hand, for longer deadlines the cost of additional VM-hour becomes less significant. We also observe that the aggregated impact of positive and negative variations in task execution time may cancel to nearly 0 and in some cases the cost overrun may be negative.

In a similar way Fig. 14 shows the deadline overrun in the presence of runtime variations. We can observe that the deadline overrun is much smaller than cost overrun. This means that the actual finish time of all tasks is not significantly affected by the runtime variation, due to the cancellation effect. We observe that even for high variations in the range from −50% to 50% the actual runtime rarely exceeds the deadline by more than 10%. This overrun can be thus easily compensated by solving the optimization problem with a respectively shorter deadline giving a safety margin in the case of expected high variations of actual task runtimes. Similar estimation is also possible in the case when the application consists of many tasks of similar size which distribution in known in advance.

8. Conclusions and Future Work

The results presented in this paper illustrate typical problems when making decisions on deployment planning on clouds and how they can be addressed using optimization techniques. We have shown how the mixed integer nonlinear programming can be applied to model and solve the problem of resource allocation on multiple heterogeneous clouds, including private and public ones, and taking into account the cost of compute instances and data transfers.

Our results show that the total cost grows slowly for long deadlines, since it is possible to use free resources from a private cloud. However, for shor deadlines it is necessary to use the instances from public clouds, starting from the ones with best price/performance ratio. The shorter the deadlines, the more costly instance types have to be added, thus the cost grows more rapidly. Moreover, our results can be also useful for multi-objective optimization. In such a case, it would be possible to run the optimization algorithm in a certain neighbourhood of the desired deadline and select the best solution using a specified cost/time trade-off. Alternatively, multiple solutions as in Fig. 6, 10 or 11 may be presented to the users allowing them to select the most acceptable solution. Our model can be also used as an approximate method to solve the problems where tasks sizes are not ideally uniform, but can differ within a limited range.

Optimal task allocation in hybrid cloud environment is not a trivial problem as one needs to know the estimates of computational cost of tasks in advance. If such data are available, it is possible to use tools such as AMPL. This approach may be successful as long as one is able to formulate the optimization model and select a suitable solver. These tasks are not straightforward though, since small change in model may move problem from one class to another (e.g. from mixed integer to MINLP) requiring to find another solver. Optimal specification of the model is also important as the same problem may be formulated in various ways, each of which which may differ considerably in performance.

In future work we plan to experiment with variations of the model to represent other classes of applications, such as scientific workflows [1] that often consist of multiple stages, each characterized by different data and compute requirements.

References

[1] E. Deelman, "Grids and clouds: Making workflow applications work in heterogeneous distributed environments," *International Journal of High Performance Computing Applications*, vol. 24, no. 3, pp. 284–298, August 2010.

[2] M. Malawski, J. Meizner, M. Bubak, and P. Gepner, "Component approach to computational applications on clouds," *Procedia Computer Science*, vol. 4, pp. 432–441, May 2011.

[3] M. Malawski, T. Gubała, and M. Bubak, "Component-based approach for programming and running scientific applications on grids and clouds," *International Journal of High Performance Computing Applications*, vol. 26, no. 3, pp. 275–295, August 2012.

[4] R. Fourer, D. M. Gay, and B. W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming.* Duxbury Press, 2002.

[5] J. Chen, C. Wang, B. B. Zhou, L. Sun, Y. C. Lee, and A. Y. Zomaya, "Tradeoffs between profit and customer satisfaction for service provisioning in the cloud," in *Proceedings of the 20th international symposium on High performance distributed computing*, ser. HPDC '11. New York, NY, USA: ACM, 2011, pp. 229–238.

[6] H. Kim, Y. el-Khamra, I. Rodero, S. Jha, and M. Parashar, "Autonomic management of application workflows on hybrid computing infrastructure," *Sci. Program.*, vol. 19, pp. 75–89, April 2011.

[7] I. Brandic, S. Pllana, and S. Benkner, "Specification, planning, and execution of qos-aware grid workflows within the amadeus environment," *Concurrency and Computation: Practice and Experience*, vol. 20, no. 4, pp. 331–345, 2008.

[8] S. Pandey, A. Barker, K. K. Gupta, and R. Buyya, "Minimizing Execution Costs when Using Globally Distributed Cloud Services," in *24th IEEE International Conference on Advanced Information Networking and Applications.* IEEE Computer Society, 2010, Conference proceedings (article), pp. 222–229.

[9] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M.-H. Su, and K. Vahi, "Characterization of scientific workflows," in *Workflows in Support of Large-Scale Science, 2008. WORKS 2008. Third Workshop on.* IEEE, Nov. 2008, pp. 1–10. [Online]. Available: http://dx.doi.org/10.1109/WORKS.2008.4723958

[10] R. Duan, R. Prodan, and X. Li, "A sequential cooperative game theoretic approach to Storage-Aware scheduling of multiple Large-Scale workflow applications in grids," in *Grid Computing (GRID), 2012 ACM/IEEE 13th International Conference on.* IEEE, 2012, pp. 31–39. [Online]. Available: http://dx.doi.org/10.1109/Grid.2012.14

[11] P. Bonami, L. T. Biegler, A. R. Conn, G. Cornuéjols, I. E. Grossmann, C. D. Laird, J. Lee, A. Lodi, F. Margot, and N. Sawaya, "An algorithmic framework for convex mixed integer nonlinear programs," *Discrete Optimization*, vol. 5, no. 2, pp. 186–204, May 2008.

[12] J. Forrest, "Cbc (coin-or branch and cut) open-source mixed integer programming solver," 2012. [Online]. Available: https://projects.coin-or.org/Cbc