ELSEVIER

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science ICCS (2010) 1–10

Procedia Computer Science

International Conference on Computational Science, ICCS 2010

Composable cost estimation and monitoring for computational applications in cloud computing environments[☆]

Hong-Linh Truong[a,*], Schahram Dustdar[a]

[a]*Distributed Systems Group, Vienna University of Technology, Austria*

Abstract

With the offer from cloud computing providers, scientists have the opportunity to utilize pay-as-you-go resources together with their own and shared resources. However, scientists need to decide which parts of their applications should be executed in cloud computing systems in order to balance the trade-off between cost, time and resource requirements. In this paper, we present a service for estimating, monitoring and analyzing costs associated with scientific applications in the cloud. Cost models associated with different application execution models are proposed and these cost models can be composed to determine costs of different scenarios. We present techniques to estimate costs for service dependency and to monitor costs associated with typical scientific applications. Experiments with real-world applications are performed to illustrate the usefulness of our techniques. Our service could eventually be integrated into cloud resource management and execution services to support on-the-fly resource scheduling.

Keywords: cloud computing, cost monitoring, cost analysis, cost estimation, scientific applications

1. Introduction

Recently, cloud computing has been considered as an emerging model which aims at allowing customers to utilize computational resources and software hosted by service providers [1, 2, 3], thus shifting the complex and tedious resource and software management tasks typically done by the customers to the service providers. Cloud computing promises to eliminate obstacles due to the management of IT resources and to reduce the cost on infrastructure investments. As a result, besides business customers, cloud computing is also attractive to many scientists from small research groups in the computational science and engineering (CSE) field.

However, still there are many unclear questions about how cloud computing can help CSE scientists [3, 1]. Among them, the need to determine the actual cost when using cloud computing is evident. In general, the cost determination

[☆]This is the revised version of the paper with the same title accepted for publication in *the Proceedings of The International Conference on Computational Science 2010 (ICCS 2010), Tools for Program Development and Analysis in Computational Science Workshop, May 31 - June 2, 2010, Amsterdam, The Netherlands*, (c) Elsevier. Last revised: March 13, 2010. This work was partially supported by the Vienna Science and Technology Fund (WWTF), project ICT08-032. We thank Dieter Pahr from Vienna University of Technology, and David Kreil and Pawel Labaj from University of Natural Resources and Applied Life Sciences, Vienna, for providing applications used in this paper and for many fruitful discussions.

[*]Corresponding author

Email addresses: `truong@infosys.tuwien.ac.at` (Hong-Linh Truong), `dustdar@infosys.tuwien.ac.at` (Schahram Dustdar) is necessary for investigating the return of investment, and, in particular, it is needed to decide when and under which forms cloud computing offers can be used. Let us consider this particular point in the interest of small CSE research groups which have limited resources or limited access to their shared computational resources and storages. These groups cannot entirely rely on that resources and storages as well as on cloud computing offers due to several reasons. Scientists of these groups need a quick assertion on the cost of executing their applications in the cloud. They want to evaluate if it makes sense to run a particular application or parts of the application using cloud computing, if cloud resources should be used in a regular or occasional basis, and if all resources of need are fully or partially based on clouds. Using information provided by the vendor, it is very difficult to calculate the cost of an application because scientists need to map the computation and data transfer requirements associated with their CSE applications to primitive prices of CPUs, storages and network transfer. Scientists expect to have cost models associated with application models, e.g., OpenMP, MPI, and workflows. Furthermore, a scientific experiment might include different parts, each may have a different application model and might or might not be executed in the cloud. Thus, it is interesting to have composable cost models in order to decide which parts of the experiment should be executed by using cloud resources.

In this paper, we present a service for estimating and monitoring costs associated with CSE applications that is, in particular, suitable for scientists from small CSE groups. These scientists have some particular constraints that require them to use different resources in different ways based on fully on-premise, partially cloud, and fully cloud resources. We present composable cost models. Based on these models, we develop a service which supports both cost estimation and real-time monitoring of costs for CSE applications in the cloud.

The rest of this paper is organized as follows. We present requirements for application cost models in Section 2. The composable cost models are described in Section 3. Our cost estimation, monitoring and analysis service is presented in Section 4. Experiments are described in Section 5. Section 6 discusses the related work. We conclude the paper and outline our future work in Section 7.

2. Requirements for Application based Cost Estimation, Monitoring, and Analysis

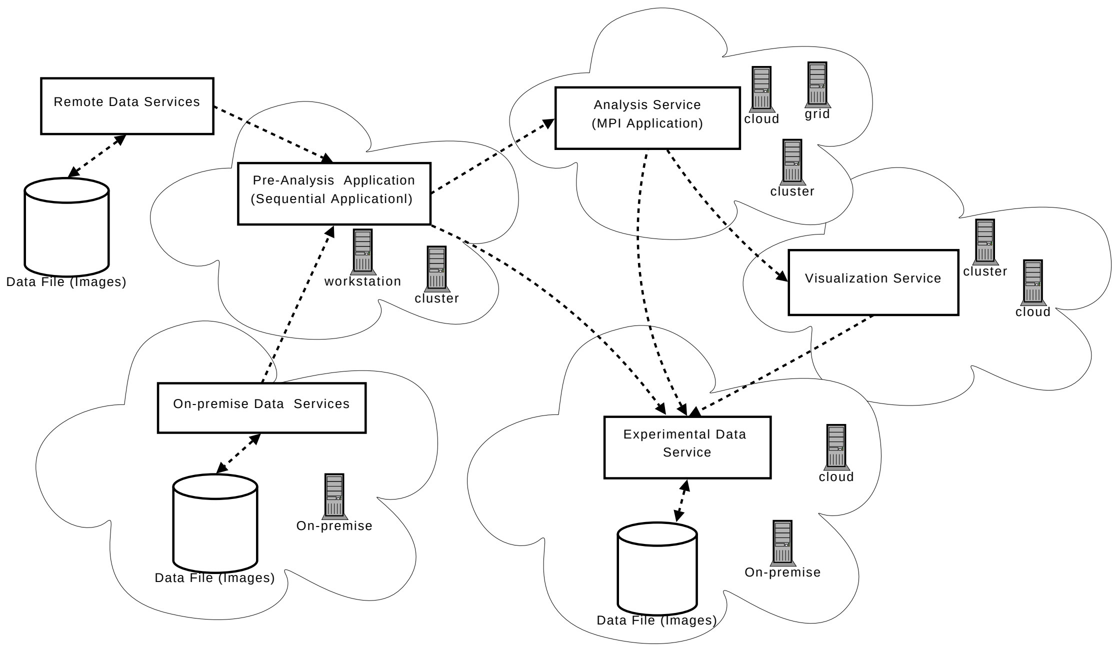


Figure 1: Overview of the bones structure simulation

In this section, we analyze two applications from two small CSE groups in Vienna to motivate the development of our work. Our first application is a *Bones Structure Simulation* (Bones) which is used to simulate the stiffness of human bones. The execution of this application includes three phases and Figure 1 describes the deployment scenario of the Bones application. First, a `Pre-Analysis` is performed by a sequential program (e.g., named `pfeToH5`) that takes CT image and template files to generate input files for micro FE (finite element) solvers (e.g., `ParFE`). Currently, template files and CT images are stored in a storage system. Second, an `Analysis` is performed in which the generated input files will be processed by the FE solver and the analysis results will be added into the input files. The FE solver is an MPI program. Third, a `Visualization` is performed by an interactive, parallel visualization program (e.g., `paraview`) which processes the output files of the `Analysis` part. Currently, many steps in data shipping are performed manually by scientists. Consider the benefit of cloud computing, the scientists of the Bones application would like to investigate whether they should use data storage and computational resources from the cloud or not, which parts of their application should be executed in the cloud, and when. Because of this requirement, the cost determination is important for deciding which parts should rely on cloud resources.

Our second application is a *Next Generation Sequencing Analysis Workflow* (GSA) for mRNA. This workflow is based on the whole shotgun sequencing. Figure 2 presents one example of the workflow. GSA supports the scientist to perform the alignment of sequences, obtained from a high-throughput parallel sequencer (e.g., Illumina, ABI Solid, or 454 Roche), to a reference set of transcript sequences, genes, or genomes. The workflow includes several parts that retrieve data from remote data sources, perform data checking, and compare sequences. Most parts are sequential programs but the comparison applications can be executed as a parallel program in a multi-threaded fashion. The main requirement for this application is online cost monitoring and analysis for different computational and data transfer activities.

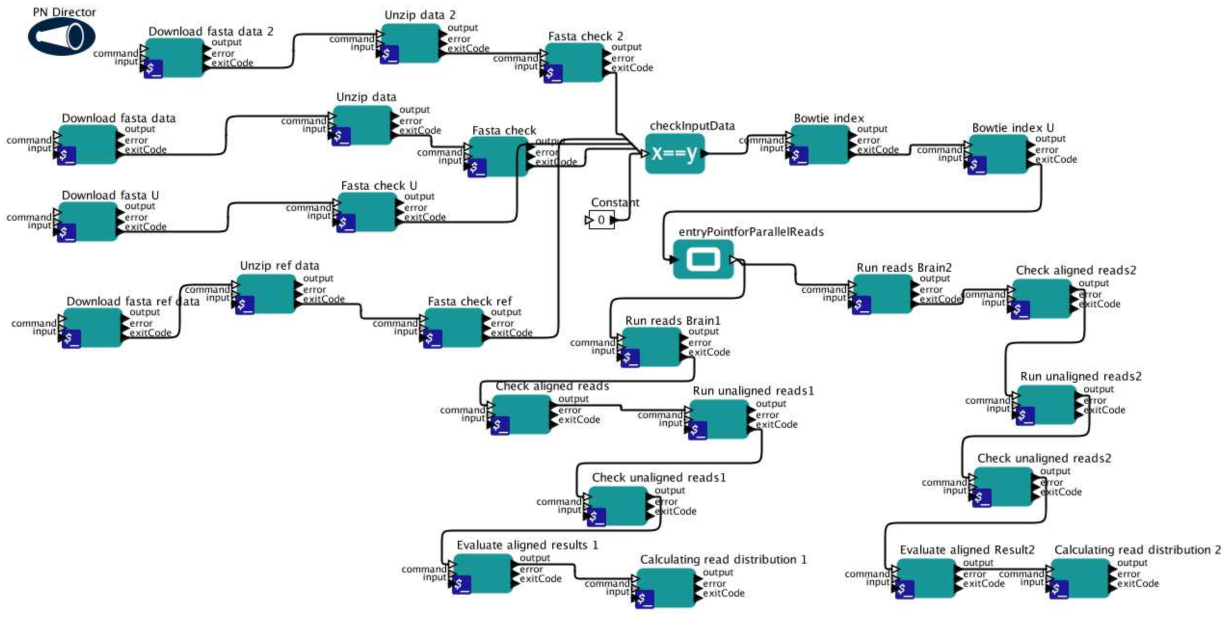


Figure 2: One example of GSA workflows

In general, such applications are popular, and given the offer from the cloud computing, scientists want configure the execution of their applications in different ways: local (e.g., for testing), partial cloud (when dealing with sensitive information) and fully cloud/cluster (e.g., for performance test). The requirement is that depending on the cost, performance and urgency, they could use full resources from the cloud or partial resources from the cloud and their own resources (e.g., clusters from the university). Among many open questions, the cost estimation and monitoring are the key tool for scientists to decide when to run the code in the cloud and under which models, fully or partially clouds.

3. Composable Cost Models

As discussed in the requirements, cost estimation, monitoring and analysis for different applications are needed. But we should note that before utilizing cloud resources, scientists already have vast knowledge about their applications. Therefore, our intuition for developing cost models is that, before deciding to move to the cloud, scientists use their known performance characteristics of their applications, e.g., transferred data and execution times, to provide many application-specific parameters that can be used for estimating costs. In our approach, we develop several basic cost models for different application execution models, such as OpenMP, MPI and workflows. This can be achieved by also leveraging previous knowledge in performance estimation, monitoring and analysis, e.g., [4, 5, 6]. Based on basic cost models, cost estimation, monitoring and analysis tools and services can be developed.

Model	**Activities**	**Cost**
M_{ds}	Data storage	$size(total) \times t_{sub} \times cost(storage)$ where t_{sub} is the subscription time
M_{cm}	Computational machine	$cost(machine)$
M_{dfi}	Data transfer into the cloud	$cost(transfer_{in})$
M_{dfo}	Data transfer out to the cloud	$cost(transfer_{out})$
M_{sd}	Single data transfer without the cost for machines performing the transfer	$size(in) \times M_{dfi} + size(out) \times M_{dfo}$
M_{sm}	Sequential/multi-threaded program or single data transfer with the cost for machines performing the transfer (cost monitoring)	$t_e \times M_{cm} + size(out) \times M_{dfo} + size(in) \times M_{dfi}$
M_{se}	Sequential or multi-threaded program (cost estimation)	$f_{pi} \times M_{cm} + size(out) \times M_{dfo} + size(in) \times M_{dfi}$ where f_{pi} is an estimated performance improvement function when n expected threads to be used. f_{pi} can be provided by performance prediction tools or scientists. In our case, currently, we use an ideal parallel performance improvement $f_{pi} = \frac{p}{n} \times t_e(p)$ where p is the number of threads used to obtain $t_e(p)$. p and $t_e(p)$ are known knowledge.
M_{pm}	Parallel/MPI programs on multiple machines (cost monitoring)	$n \times M_{cm} \times t_e + size(out) \times M_{dfo} + size(in) \times M_{dfi}$
M_{pe}	Parallel/MPI programs on multiple machines (cost estimation)	$n \times M_{cm} \times f_{pi} + size(out) \times M_{dfo} + size(in) \times M_{dfi}$ where f_{pi} is an estimated performance improvement function when n processes are used.
M_{wm}	Workflows (cost monitoring)	$\sum_{i=1}^{k} (size(in_i) \times M_{dfi}) + \sum_{i=1}^{l} (size(out_i) \times M_{dfo}) + \sum_{i=1}^{n} (M_{cm} \times t_e(machine_i))$
M_{we}	Workflows (cost estimation)	$\sum_{i=1}^{nwr} cost(wr_i)$. For a workflow region wr_i, $cost(wr_i) = \sum_{j=1}^{q} (cost(activity_j))$ where $cost(activity_j)$ is determined based on M_{mp}, M_{sm}, and M_{sd}, when the activity $activity_j$ is a parallel activity, sequential activity, or a data transfer activity, respectively.

Table 1: Costs. t_e and $t_e(p)$ are the total elapsed time for executing computational task or data transfer and the execution time obtained with p parallel processes/threads, respectively. n is the number of machines used or to be used. $size(in)$ and $size(out)$ are the size of the data transferred into and out to a cloud.

Table 1 describes basic cost models. The first four models, M_{ds}, M_{cm}, M_{dfi} and M_{dfo} are basic models that are provided by cloud providers in their pricing specifications. By using these models and the well-established execution models of sequential, parallel and workflow applications, corresponding cost models are introduced. The execution of sequential, multi-threaded, or OpenMP programs uses a single machine and these programs may involve data transfers from/to the cloud. Therefore, the cost model - M_{sm} - for monitoring costs of these programs is based on the elapsed time, the machine cost, and the data transfer cost. However, with respect to the cost estimation, the cost model for these programs - M_{se} - is also dependent on the number of the threads to be used. When estimating the cost for multi-threaded/OpenMP programs, we assume that an estimated performance improvement function can be obtained, e.g., based on the knowledge of scientists or existing performance prediction tools. Since our work has not been integrated with such tools, by default we assume that such programs can be ideally parallelized (given $t_e(p)$ is the execution time in p threads specified by the user, the estimated execution time in n threads would be $\frac{p}{n} \times t_e(p)$). Similarly, for parallel/MPI programs using n machines, we also have two models, M_{pm} for cost monitoring and M_{pe}

for cost estimation, and the assumption of ideal parallelization. In the case of workflows, the cost monitoring model - M_{wm} - is similar to that of other programming models by determining the cost based on the volume of transferred data and total used time of machines. But the monitoring cost has to be breakdown and associated with activities based on the workflow structure. For the estimating cost model of workflows, M_{we}, we utilize our previous work on online performance analysis of workflows by assuming that a workflow can be structured into set of workflow regions, each region includes a set of activities, and an activity can be a parallel activity or sequential activity [7]. Then, we use cost models for sequential/multi-threaded and parallel programs and data transfer activities for estimating the cost of a workflow based on workflow structures. Note that our workflow models in this paper are not necessary the same as the workflow to be executed by contemporary workflow engines. They can be a workflow of interdependent logical parts of an application, for example, similar to the Bones application in Section 2.

4. Cost Estimation, Monitoring and Analysis Service

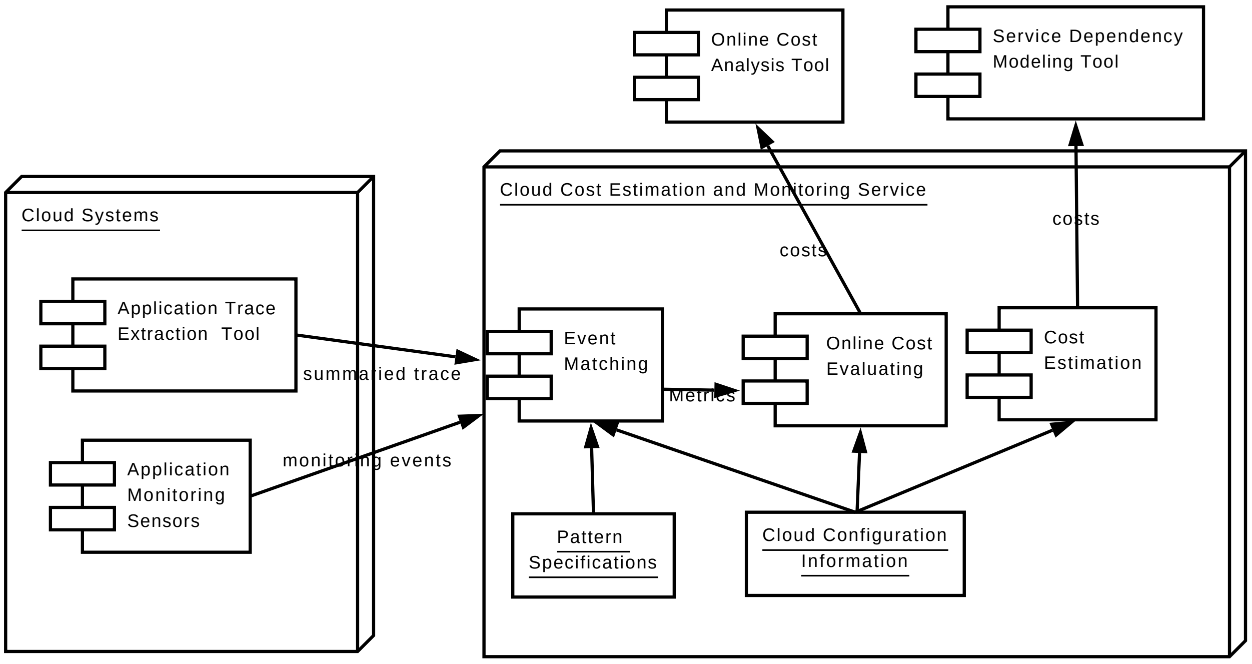

Online Cost Analysis Tool
Service Dependency Modeling Tool
Cloud Systems
Cloud Cost Estimation and Monitoring Service
costs
costs
Application Trace Extraction Tool
summaried trace
Event Matching
Metrics
Online Cost Evaluating
Cost Estimation
Application Monitoring Sensors
monitoring events
Pattern Specifications
Cloud Configuration Information

Figure 3: Cloud Cost Estimation and Monitoring Service

Utilizing cost models and monitoring data, we develop a cost estimation, monitoring and analysis service for the cloud. Figure 3 presents the architecture of our service named CCEMS. The core of the service includes several components for (i) processing monitoring events based on complex event processing techniques (*Event Matching* and *Pattern Specification*), (ii) evaluating costs based events (*Online Cost Evaluating*), and estimating costs based on service/application dependency models (*Cost Estimation*). CCEMS provides the *Service Dependency Modeling Tool* for specifying service/application dependency models and the *Online Cost Analysis Tool* for visualizing the cost. Furthermore, it relies on pricing knowledge about data transfers and resources costs of cloud systems specified in the *Cloud Configuration Information*.

To estimate and monitor costs associated with applications, we need to capture information about execution time, data transfer, machines and storages. Main types of data to be collected are execution time, machine name, storage, data transfer size, and data transfer source and destination. We use events to capture such data. An event is described in Table 2. All events belonging to the same application will be correlated via a `resourceID`. Events can be generated/provided by the *Service Dependency Modeling Tool*, *Application Monitoring Sensors*, or *Application Trace Extraction Tool*.

Name	Description
resourceID	identifier of the application
eventName	event name, whether it is a start or end of a transfer or computation
eventTime	timestamp of the event
activityName	name of the activity (e.g., code region, workflow activity, data transfer)
activityInstanceId	instance id of an activity
activityType	whether it is a data transfer or computation task
computationalNode	the machine where the activity is executed
dataSize	the size of transferred data
sourceId	identifier of a source from which data is transferred
destinationId	identifier of a destination to which the data is transferred

Table 2: Event description used for cost estimation, monitoring and analysis

To support the cost estimation, we develop a tool to describe the dependency among parts of applications, computational resources and storages. We model dependencies in a directed graph $G(N, E)$ where N is a set of nodes and E is a set of edges. A node $n \in N$ represents a part of applications, computation resources and storages whereas an edge $e(n_i, n_j) \in E$ means that n_j is dependent on n_i. Three types of dependencies are supported: data, control, and resource dependencies. Resources (computation service/data service) can be specified as a cloud resource or an on-premise resource. Based on the knowledge specified by scientists for nodes and edges, simulated events specified in Table 2 are generated and sent to the *Cost Estimation* component. This component will utilize the *Cloud Configuration Information* to estimate the cost.

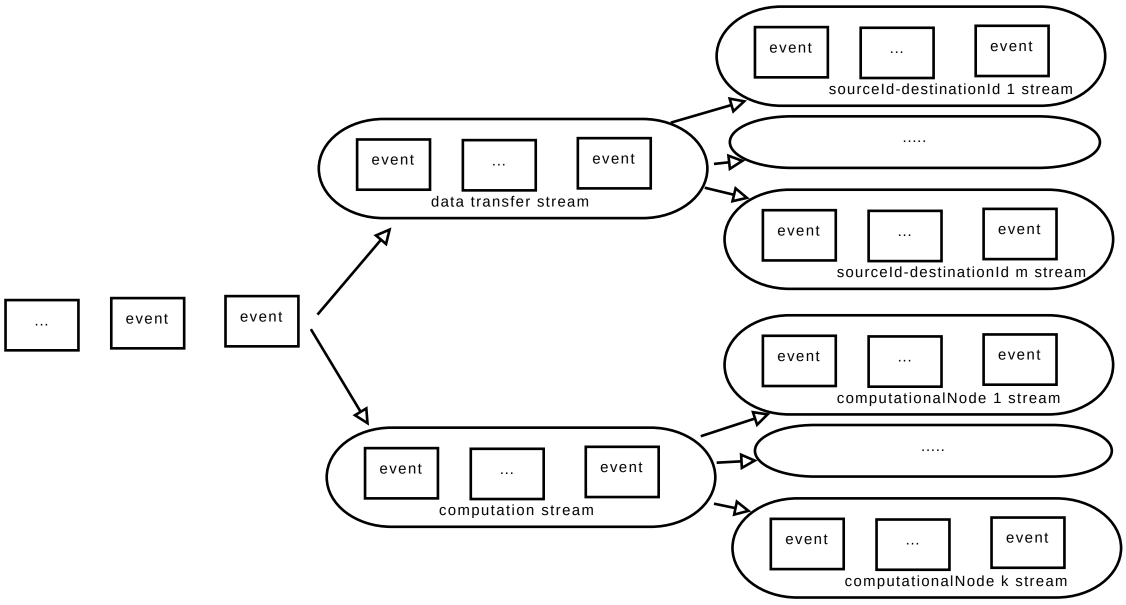


Figure 4: Complex event processing for events

For monitoring and analyzing costs, main types of data need to be collected, such as, execution time, machine name, data transfer size, and data transfer source and destination. To this end, we utilize well-known instrumentation techniques [4, 5, 6]. Given the fact that instrumentation techniques are well-developed, it is straightforward to collect these types of data, using existing instrumentation techniques, such as manually or automatically inserting probes at the beginning and at the end of the data transfer, application processes, MPI IO calls, and workflow activities. The required data for cost analysis may also be obtained by using existing performance tracing and monitoring tools, such as [4, 5, 6], and by performing data extraction from these tools. Hence we assume that such data is provided based on events specified in Table 2. Our service analyzes these data on the fly and costs are determined based on the models given in Section 3. Although from the monitoring perspective, costs are determined based on machine usages and data transfers, the determined cost has to be divided into cost elements associated with particular parts of the application. Figure 4 presents how application monitoring events are processed to provide detailed costs associated with different application models. First, events are filtered into two different event data streams, st_{df} for data transfer activities and st_{comp} for computation activities. Then, events in the data streams st_{df} and st_{comp} are filtered into streams $st_{df}(sourceId, destinationId)$ and $st_{comp}(computationalNode)$, respectively. Based on that, costs are determined for activities based on event information and pricing knowledge. A cost breakdown tree is built by mapping costs determined from different streams to functions, activities and programs. Costs are updated when a new event triggers the cost evaluation.

We have implemented both tools for estimating costs and for monitoring and analyzing costs and the Web service. Our tool for monitoring and analyzing costs provides only application execution costs. In our implementation, we use Web services technologies to ensure that our service will be suitable for the cloud environment in which access APIs are mostly offered via Web services. Monitoring events are described in XML or JSON while the CCEMS is implemented as a RESTful service.

5. Experiments

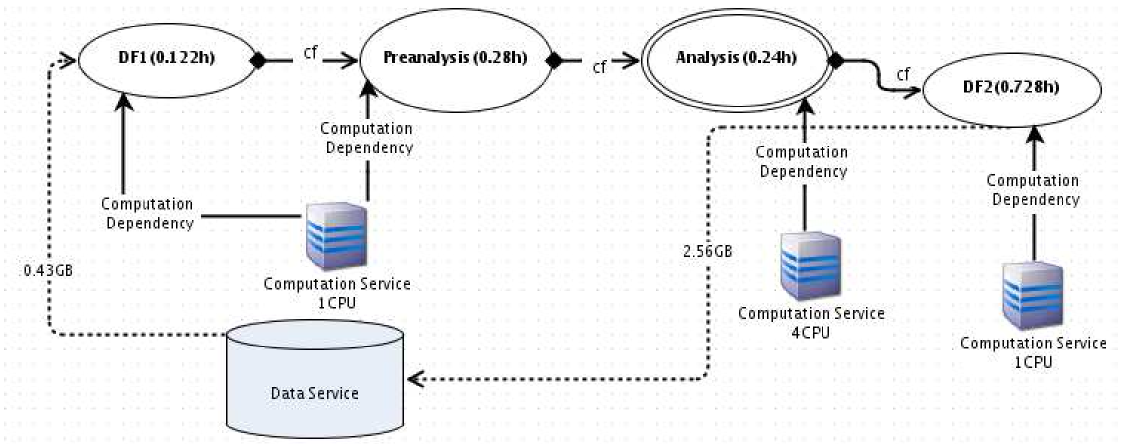


Figure 5: Dependency model with on-premise data service. `DF1` represents a data transfer of source images from an on-premise data service and `DF2` represents a data transfer of analysis results to an on-premise data service.

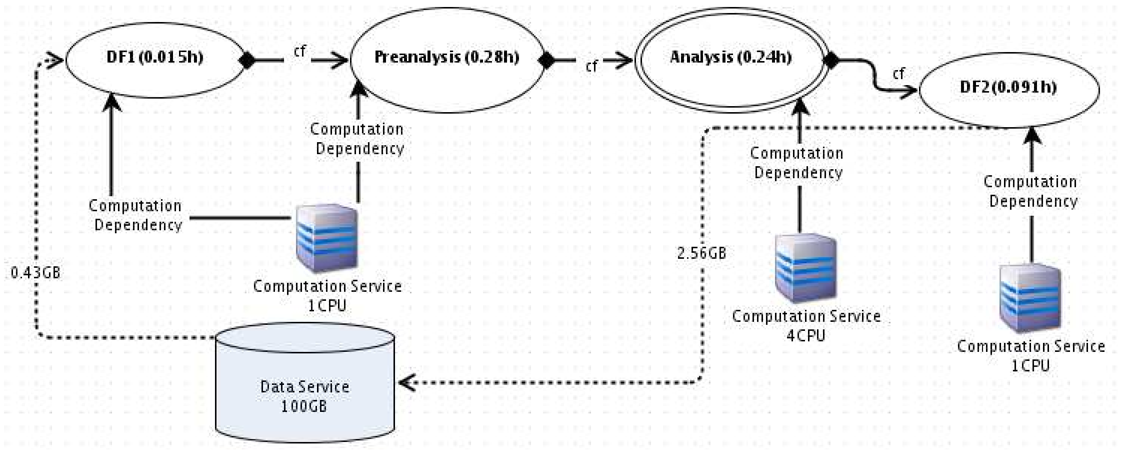


Figure 6: Dependency model with cloud data service. `DF1` and `DF2` rely on a cloud-based data service.

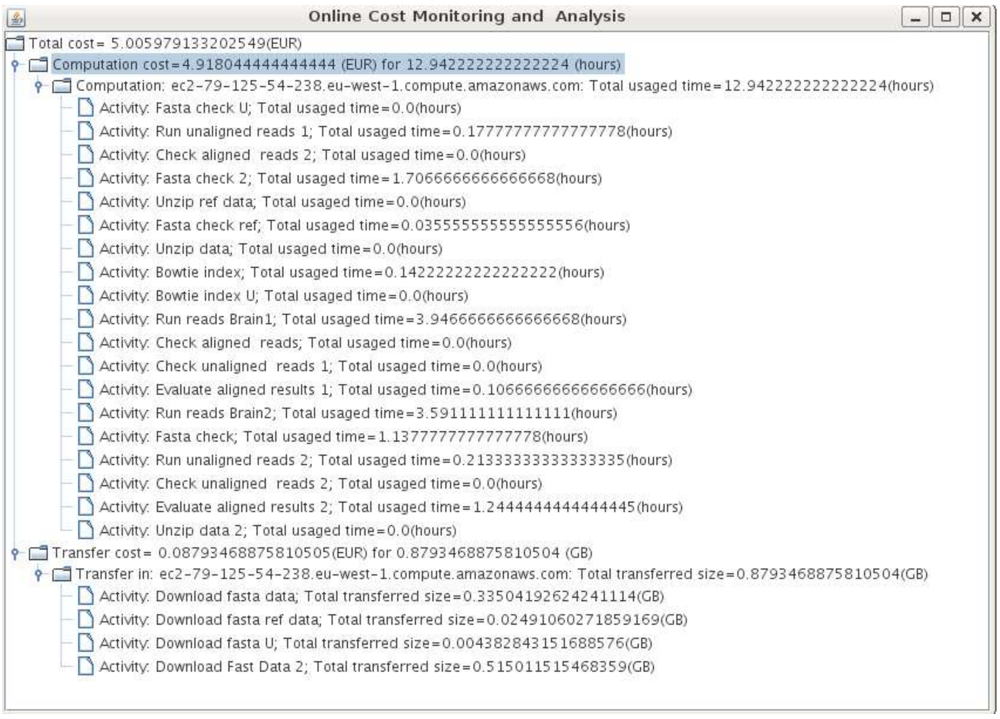
To illustrate the cost estimation, we examine the case of the Bones application. Figures 5 and 6 present two scenarios that the Bones scientists want to play with. In Figure 5 the data service used to store CT image files and analysis results is on premise. Resources in Figure 6 are fully in the cloud. Many known and estimated performance factors are specified in the graph, such as the amount of data transfer, execution time of activities, number of machines and volume of storages. Table 3 gives examples of the estimated costs based on performance knowledge obtained in our private cloud but using Amazon EC2 prices[1]. Table 3 shows that given the specified performance metrics and configuration, the tool helps decide different strategies to use resources. For example, if using partial cloud resources it is cheaper with the illustrated setting, but it will lead to a double amount of execution time, compared with using all cloud resources. Depending on different cloud prices and configuration, many variations of costs can be estimated by our service. The above-mentioned example just illustrates our ideas of composing specific cost models for estimating relevant costs associated with different steps in scientific experiments.

Activities	**Nr Machine**	**Transfer In (GB)**	**Transfer Out(GB)**	**Time (hours)**	**Storage (GB)**	**1 Run Cost (EUR)**	**30 Runs Cost (EUR)**
Partial cloud-based resources							
DF1	1	0.43	0.43	0.122	0	0.139	4.18
Preanalysis	1	0	0	0.28		0.053	1.596
Analysis	4	0	0	0.24		0.182	5.472
DF2	1	2.56	2.56	0.728		0.83	24.887
Total				1.37		1.204	36.135
Full cloud-based resources							
DF1	1	0.43	0.43	0.015	3	0.569	17.07
Preanalysis	1	0	0	0.28		0.053	1.596
Analysis	4	0	0	0.24		0.182	5.472
DF2	1	2.56	2.56	0.091		0.708	21.255
Total				0.626		1.513	45.393

Table 3: Example of estimated costs for running the Bones application with different possibilities. Costs for input data transfer/GB, output data transfer/GB, CPU instance/hour and storage/month are 0.1 EUR , 0.17 EUR, 0.19 EUR and 0.15 EUR. Data transfer within our cloud is 8MB/s and between our cloud and user's on-premise storage is 1MB/s.

In the second experiment, we executed the GSA workflow in a Linux-based large instance in Amazon EC2 in Europe. We used a default AMI (Amazon Machine Image). The workflow is executed through an SSH connection which is maintained entirely during the execution. Figure 7 presents the cost and performance of the GSA application. Costs are monitored and analyzed on the fly and associated with individual activities. Table 5 presents the extracted usage values for this experiment that are reported by the Amazon billing system, which does not consider application-specific costs in this case. Compared with our monitoring tool, the cost for using the machine instance is similar. However, there are strong differences with respect to data transfer in, transfer out, and I/O calls reported by our tool and Amazon. With respect to the data transfer in, the difference is due to the transfer of necessary libraries and codes for the execution of the workflow. In the execution of workflow, there is no data transfer out, however, during the execution of the workflow, some activities produce several application warning messages, (using the `print` function), which are redirected to the user machine. This produces a large volume of data transfer out. With respect to the I/O calls, the cloud AMI we used automatically attaches an Amazon EBS (Elastic Block Store) volume and the workflow activities perform I/O calls in the file system of the cloud instance. Thus, such calls are also charged by Amazon, while this has not been counted by our monitoring tool. The differences between costs reported by the monitoring tool and the Amazon billing system show that there are cost elements that the user may not be fully aware of in advance or may pay due to the lack of attention (e.g., the output of `print` commands). Furthermore, it shows that it is not trivial to determine some types of costs due to the complex dependency among services in the cloud (e.g., the case of cloud instances and storages mounted in the local file system in the instances) or due to the overhead for determining the costs (e.g., monitoring all I/O calls in an application could introduce severe performance overhead). While we

[1]`http://aws.amazon.com/ec2/`



```
Total cost= 5.005979133202549(EUR)
  Computation cost=4.918044444444444 (EUR) for 12.942222222222224 (hours)
    Computation: ec2-79-125-54-238.eu-west-1.compute.amazonaws.com: Total usaged time=12.942222222222224(hours)
      Activity: Fasta check U; Total usaged time=0.0(hours)
      Activity: Run unaligned reads 1; Total usaged time=0.17777777777777778(hours)
      Activity: Check aligned  reads 2; Total usaged time=0.0(hours)
      Activity: Fasta check 2; Total usaged time=1.7066666666666668(hours)
      Activity: Unzip ref data; Total usaged time=0.0(hours)
      Activity: Fasta check ref; Total usaged time=0.035555555555555556(hours)
      Activity: Unzip data; Total usaged time=0.0(hours)
      Activity: Bowtie index; Total usaged time=0.14222222222222222(hours)
      Activity: Bowtie index U; Total usaged time=0.0(hours)
      Activity: Run reads Brain1; Total usaged time=3.9466666666666668(hours)
      Activity: Check aligned  reads; Total usaged time=0.0(hours)
      Activity: Check unaligned  reads 1; Total usaged time=0.0(hours)
      Activity: Evaluate aligned results 1; Total usaged time=0.10666666666666666(hours)
      Activity: Run reads Brain2; Total usaged time=3.591111111111111(hours)
      Activity: Fasta check; Total usaged time=1.1377777777777778(hours)
      Activity: Run unaligned reads 2; Total usaged time=0.21333333333333335(hours)
      Activity: Check unaligned  reads 2; Total usaged time=0.0(hours)
      Activity: Evaluate aligned results 2; Total usaged time=1.2444444444444445(hours)
      Activity: Unzip data 2; Total usaged time=0.0(hours)
  Transfer cost= 0.08793468875810505(EUR) for 0.8793468875810504 (GB)
    Transfer in: ec2-79-125-54-238.eu-west-1.compute.amazonaws.com: Total transferred size=0.8793468875810504(GB)
      Activity: Download fasta data; Total transferred size=0.33504192624241114(GB)
      Activity: Download fasta ref data; Total transferred size=0.02491060271859169(GB)
      Activity: Download fasta U; Total transferred size=0.004382843151688576(GB)
      Activity: Download Fast Data 2; Total transferred size=0.5150115154683559(GB)
```

Figure 7: Cost analysis of an execution of GSA workflow using an Amazon EC2 Large Instance (0.38 EUR per instance-hour in Europe)

do not show the cost monitoring in a large scale setting, this experiment demonstrated the usefulness of an online cost monitoring tool to provide realtime cost information that can be used to steer the use of resources with respect to performance and financial requirements.

Operation Name	Usage Type	Usage Value	Cost (EUR)
RunInstances	EU-BoxUsage:m1.large	13 hours	4,94
RunInstances	EU-DataTransfer-In-Bytes	1,043 GB	0,104
RunInstances	EU-DataTransfer-Out-Bytes	4,48 GB	0,672
EBS:IO-Write	EU-EBS:VolumeIOUsage	2083828 I/O calls	0,229

Table 4: Operation name, usage types and usage values extracted from the Amazon billing report during the execution of the GSA workflow. We determined costs by using the following prices: 0.38 EUR per large instance (m1.large) instance-hour, 0.1 EUR per GB data transfer in, 0.150 EUR per GB data transfer out, 0.11 EUR per 1 million I/O requests.

When the cost estimation and monitoring tools are integrated with existing application execution management/deployment systems in the cluster and cloud systems, then depending on times and budgets, the scientists can use the tools to decide which cloud systems and which parts of their application should be executed in the cloud.

6. Related Work

To an application execution, the cost of using cloud computing would be computing resources, storages and data transfer, but the total cost is related to many factors and activities, such as compute resources, storage, data transfers, security, and data cleaning [1, 8]. In this paper we discuss only costs associated with application executions. While cloud service providers give some basic tools for determining computation and data transfer costs, they are trivial.

Scientists have started using cloud computing for CSE applications, such as [9, 10] and they have provided various feedback on experimenting cloud computing for scientific applications. Some works are focused on the evaluation of particular cloud systems, such as for Amazon [11]. With respect to cost estimations for scientific applications in cloud computing, [1] gives generic cost comparison discussion and [8] provides experimental work on costs for workflows in cloud computing. Both raise important questions on the complexity of cloud cost models but do not develop cost estimation and monitoring tools. We share a similar view but we focus on cost models for typical application models and provide a cost estimation and monitoring service.

Clearly, many SaaS metering tools, such as eVapt Instant SaaS Metering [12], for cloud computing are useful for billing. Our cost estimation models are based on various previous work on performance analysis of parallel and distributed applications. We utilize the execution model and time evaluation models for our cost models. Many performance prediction models have been developed for CSE applications, such as based on application inputs and outputs or historical performance data [13, 14]. In our work, we do not perform the performance prediction but provide a tool for scientists to define the dependency and known/estimated metrics for their applications based on that the cost is estimated. Thus, the accuracy of the cost estimation is dependent on scientist knowledge, and this accuracy quality can be improved if our service can also be well integrated with existing performance prediction tools.

7. Conclusion and Future Work

In this paper, we have analyzed requirements for cost estimation and monitoring tools from scientists of small CSE groups. By exploiting past knowledge in performance monitoring and analysis techniques for parallel and workflow applications, we have developed a service to support real cost estimation and monitoring and analysis. Our contribution is a service that is able to perform the cost determination for scientific applications in cloud computing environments. We have demonstrated our solution with different experiments.

Our work is just a starting point and it has not been integrated into any real application execution management systems. Furthermore, our experiments are conducted in a small scale so that experiments should be extended to examine the quality of cost estimation and monitoring. The estimation service should be also improved with existing research in performance prediction to demonstrate the trade-off between real costs and performance.

References

[1] M. Armbrust, A. Fox, R. Grifth, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia, Above the clouds: A berkeley view of cloud computing, Tech. rep., University of California at Berkeley (February 2009).
URL `http://berkeleyclouds.blogspot.com/2009/02/above-clouds-released.html`

[2] I. T. Foster, Y. Zhao, I. Raicu, S. Lu, Cloud computing and grid computing 360-degree compared, CoRR abs/0901.0131.

[3] R. Buyya, C. S. Yeo, S. Venugopal, Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities, High Performance Computing and Communications, 10th IEEE International Conference on 0 (2008) 5–13. doi:http://doi.ieeecomputersociety.org/10.1109/HPCC.2008.172.

[4] H. L. Truong, P. Brunner, V. Nae, T. Fahringer, Dipas: A distributed performance analysis service for grid service-based workflows, Future Generation Comp. Syst. 25 (4) (2009) 385–398.

[5] M. Geimer, S. Shende, A. D. Malony, F. Wolf, A generic and configurable source-code instrumentation component, in: G. Allen, J. Nabrzyski, E. Seidel, G. D. van Albada, J. Dongarra, P. M. A. Sloot (Eds.), ICCS (2), Vol. 5545 of Lecture Notes in Computer Science, Springer, 2009, pp. 696–705.

[6] A. Knüpfer, D. Kranzlmüller, B. Mohr, W. E. Nagel, M09 - program analysis tools for massively parallel applications: how to achieve highest performance, in: SC, ACM Press, 2006, p. 223.

[7] H. L. Truong, S. Dustdar, T. Fahringer, Performance metrics and ontologies for grid workflows, Future Generation Comp. Syst. 23 (6) (2007) 760–772.

[8] E. Deelman, G. Singh, M. Livny, B. Berriman, J. Good, The cost of doing science on the cloud: the montage example, in: SC '08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing, IEEE Press, Piscataway, NJ, USA, 2008, pp. 1–12. doi:http://doi.acm.org/10.1145/1413370.1413421.

[9] C. Hoffa, G. Mehta, T. Freeman, E. Deelman, K. Keahey, B. Berriman, J. Good, On the use of cloud computing for scientific workflows, eScience, IEEE International Conference on 0 (2008) 640–645. doi:http://doi.ieeecomputersociety.org/10.1109/eScience.2008.167.

[10] C. Evangelinos, C. Hill, Cloud computing for parallel scientific hpc applications: Feasibility of running coupled atmosphere-ocean climate models on amazon's ec2, in: The first workshop on Cloud Computing and its Applications (CCA'08), 2008, `http://www.cca08.org/papers/Paper34-Chris-Hill.pdf`.

[11] M. R. Palankar, A. Iamnitchi, M. Ripeanu, S. Garfinkel, Amazon s3 for science grids: a viable solution?, in: DADC '08: Proceedings of the 2008 international workshop on Data-aware distributed computing, ACM, New York, NY, USA, 2008, pp. 55–64. doi:http://doi.acm.org/10.1145/1383519.1383526.

[12] evapt instant saas metering, `http://www.evapt.com/`. Last access: 16 June 2009.

[13] S. Pllana, T. Fahringer, Performance prophet: A performance modeling and prediction tool for parallel and distributed programs, in: ICPP Workshops, IEEE Computer Society, 2005, pp. 509–516.

[14] M. Casas, R. M. Badia, J. Labarta, Prediction of behavior of mpi applications, in: CLUSTER, IEEE, 2008, pp. 242–251.