A Simplified Model for Simulating the Execution of a Workflow in Cloud

Roland Mathá, Sasko Ristov(✉), and Radu Prodan

Institute for Computer Science, University of Innsbruck,
Technikerstr. 21a, 6020 Innsbruck, Austria
{roland,sashko,radu}@dps.uibk.ac.at

Abstract. Although simulators provide approximate, faster and easier simulation of an application execution in Clouds, still many researchers argue that these results cannot be always generalized for complex application types, which consist of many dependencies among tasks and various scheduling possibilities, such as workflows. DynamicCloudSim, the extension of the well known CloudSim simulator, offers users the capability to simulate the Cloud heterogeneity by introducing noisiness in dozens parameters. Still, it is difficult, or sometimes even impossible to determine appropriate values for all these parameters because they are usually Cloud or application-dependent. In this paper, we propose a new model that simplifies the simulation setup for a workflow and reduces the bias between the behavior of simulated and real Cloud environments based on one parameter only, the *Cloud noisiness*. It represents the noise produced by the Cloud's interference including the application's (in our case a workflow) noisiness too. Another novelty in our model is that it does not use a normal distribution naively to create noised values, but shifts the mean value of the task execution time by the cloud noisiness and uses its deviation as a standard deviation. Besides our model reduces the complexity of DynamicCloudSim's heterogeneity model, evaluation conducted in Amazon EC2 shows that it is also more accurate, with better *trueness* (closeness to the real mean values) of up to 9.2% and *precision* (closeness to the real deviation) of up to 8.39 times.

Keywords: Accuracy · Makespan · Modeling · Precision · Simulator · Trueness

1 Introduction

Sciences of various domains other than computer science use scientific workflows to model their complex computational pipelines, which brings them many benefits such as reusing the results of parts or entire workflows, failure management, or parallelisation. Managing the workflows' execution is a complex task, as each workflow requires different computing, memory or I/O capacity, making the designing of a common, but appropriate, distributed environment for all workflows types very difficult and sometimes almost impossible. Workflows can also be

F.F. Rivera et al. (Eds.): Euro-Par 2017, LNCS 10417, pp. 319–331, 2017.
DOI: 10.1007/978-3-319-64203-1_23

executed in Cloud as similar as in traditional clusters, as many workflow management services that allow the effective utilisation of the Cloud's elastic resources already exist [23]. Still, Cloud produces many additional challenges compared with the traditional clusters [2] caused by its on-demand elastic resource provisioning, dynamic starting of instances [20], and variant performance of virtual machines (VMs) during a time period [22].

Instead of executing in a real Cloud environment to determine the behavior of an application, many researchers resort to simulators for their analysis [3], which allows them to reduce costs for purchasing and maintaining expensive hardware resources, and time for executing time-consuming algorithms [21] and later on to determine that its performance is over- or under estimated. Moreover, simulators can be used to experiment new prototype solutions and identify the "optimal" resource configurations before deployment of production platforms. Most common simulators allow users to create a virtual data center considering their latest computing, networking, energy, or cost requirements. However, although they can simulate an elastic Cloud data center, simulators usually neglect the Cloud performance fluctuation and uncertainty [10], which can lead to wrong estimation. This aspect is especially important for workflow executions, as they consist of a high number of data and control flow dependencies [15] that further affect their overall performance without any correlation [18]. A small disturbance in a task execution can dramatically affect the scheduling of the following workflow tasks, resulting in a completely inefficient schedule that increases the execution cost or execution time (the makespan), or sometimes even both [1].

Although some simulators, such as DynamicCloudSim [4], allow users to configure the simulation considering a certain heterogeneity and instability of Cloud, still, there are several deficiencies for a proper configuration. The configuration itself is a complex process, as users need to configure more than ten parameters, for which they do not know the exact values of parameters in order to configure a specific Cloud and application to be executed. For example, the default values that are intended for Amazon's EC2 are several years old, and cannot be used for other public or private Clouds. Additionally, public Clouds can be seen as a black-box whose internal parameters are unknown to regular users. Even by using results of previous research, some parameters still cannot be generalized as they are valid either for that specific Cloud, a specific application, or even a combination of both.

We therefore propose a simpler model that reduces the configuration of the noisiness to a single parameter the *Cloud noisiness*, instead of dozens. The injected noise causes an instability in *Task Execution Time* (TET), which improves the *accuracy*, represented through the *trueness* (i.e. closeness of the true mean value) and the *precision* (i.e. closeness of corresponding standard deviation) of the simulation, as defined in ISO-5725 standard [12]. We conduct a series of evaluation experiments in Amazon EC2 and the most common state-of-the-art simulator - DynamicCloudSim. The evaluation proves that our simpler model implemented in DynamicCloudSim shows up to 9.2% higher trueness and up to 8.39 times higher precision for workflow execution simulations compared

to DynamicCloudSim. Not only that our proposed model shows better *accuracy* to the real execution, but its configuration is much simpler and easier.

The paper is structured in several sections as follows. Section 2 presents the related works in modeling the workflow execution instability and the features of cloud simulators in this domain. The models for workflow, Cloud, experiments and test cases that are used for our noisiness model are presented in Sect. 3. Our simplified, more accurate model of adding a noise in simulation is described in Sect. 4. Sections 5 and 6 present the testing methodology and results of the evaluation of our model and current state-of-the-art simulation model of DynamicCloudSim. Finally, we conclude the paper and present plans for future work in Sect. 7.

2 Related Work

Many Cloud features and parameters can cause the performance instability: heterogeneity of resources, instance types, number of instances, instance straggling, instance failures, multi-tenancy, networking bottlenecks, resource time-sharing, etc. As a consequence, an instance of the same type provides different performance for the same task over some time period. Dejun et al. [7] reported high performance deviations in Amazon EC2. Jackson et al. [13] determined that different underlying hardware for similar instances caused performance perturbation. Schad et al. [19] detected a long-term performance instability of Amazon EC2, which was correlated also to the CPU model of the same instance type, the hour of the day, and the day of the week. Iosup et al. [11] determined yearly and daily patterns of performance variability, but also periods of constant performance. All these behaviors depend also on the executed application.

CloudSim [5] simulates scheduling algorithms and resource provisioning in elastic Cloud environment, but Cloud performance instability remains unaddressed. Chen and Deelman [6] extended the Cloudsim into WorkflowSim, by introducing several parameters specific to workflows. Still, all these extensions do not introduce the Cloud performance instability. Other works developed scalable simulators covering up to hundreds of thousands of heterogeneous machines [8]. For example, GroudSim [16] is a scalable event-based simulator for Grid and Cloud environments. GloudSim [9] is a simulator that introduces some dynamics in execution by resizing the instances. Still, it does not offer a TET's instability, as the performance of specific VM is constant during a time period.

Bux and Leser [4] went further in this direction by developing the DynamicCloudSim simulator as an extension of CloudSim that introduces several additional characteristics to simulate the Cloud heterogeneity, such as heterogeneous underline hardware, VM stragglers, VM failures, long and short term fluctuations, etc. However, configuring dozens of parameters for heterogeneity is not an easy task, as users are usually not aware of the internal Cloud architecture. We therefore went a step further by treating the Cloud as a black-box and introduced much simpler approach that needs a configuration of the noise into one parameter only. It includes two instabilities in itself: *workflow noisiness* (e.g.

dependencies, structure, TET deviation) and *Cloud noisiness* (e.g. heterogeneity). Nevertheless, although it is a simple method, the results of the evaluation show that our model improves the accuracy compared to the related DynamicCloudSim's instability model. Schad et al. [19] reported that several performance parameters are unstable with a normal distribution. We also use the normal distribution to add a noise in TET, but instead of naively generating the variables distributed with a normal distribution, we inject the noisiness parameter by shifting the TET's mean value by the *Cloud noisiness* parameter.

3 Modeling the Workflow and Cloud

This section formally models the workflow and Cloud environment, which are used for our cloud noisiness model later on.

3.1 Workflow Application Model

We model a *workflow application* W as a precedence constraint graph (T, D) consisting of a set $T = \bigcup_{i=1}^{n} \{T_i\}$ of n *tasks* T_i, which are interconnected through a set of dependencies $D = \{(T_i, T_j, D_{ij}) \,|\, (T_i, T_j) \in T \times T\}$, where (T_i, T_j, D_{ij}) implies that T_i needs to be executed before T_j, and the file size to be transferred from T_i to T_j is D_{ij} bytes. The tasks are assumed to be non-preemptive, so it is not allowed to suspend one and resume it later on.

The function $pred : T \to \mathcal{P}(T)$, where $\mathcal{P}$ denotes the power set, returns the set of immediate *predecessors* of each task $T_i \in T$ (i.e. $T_j \in pred\,(T_i) \iff (T_j, T_i, D_{ji}) \in D$), while the function $succ : T \to \mathcal{P}(T)$ returns the set of immediate *successors* of the task T_i (i.e. $T_j \in succ\,(T_i) \iff (T_i, T_j, D_{ij}) \in D$). Each workflow has an *entry task* T_{entry} with no predecessors (i.e. $T_{entry} \in T : pred\,(T_{entry}) = \emptyset$) and an *exit task* T_{exit} with no successors (i.e. $T_{exit} \in T : succ\,(T_{exit}) = \emptyset$).

Each task T_i has a *requirement vector* R_i, which defines its hardware or software requirements such as the minimum value of memory or storage needed for execution. We express the *computational complexity* w_i (i.e. work) of each task T_i in million of instructions (MI).

Note that this is a simplified view of a workflow. Still, workflows with multiple entry/exit tasks are covered by adding a single "dummy" entry task with the computational complexity $w = 0$ (i.e. without any complexity) before all entry tasks or a single "dummy" exit task, also with the computational complexity $w = 0$, after all exit tasks, respectively.

3.2 Cloud Infrastructure Model

A Cloud offers a set of r *VM types* $IT = \bigcup_{k=1}^{r} \{IT_k\}$. Each instance type IT_k is characterized by two parameters: *computational speed* s_k in million instructions per second (MIPS) and *number of CPUs* c_k. We denote the set of available *VM instances* as: $I = \bigcup_{j=1}^{m} \{I_j\}$, whose number m is constant during the workflow

execution. Each instance I_j has an associated instance type IT_k defined as a function: $type : I \rightarrow IT$.

We model the *expected TET* t_i^j of a task T_i as the ratio between its computational complexity and the speed of the instance I_j on which it is executed ($t_i^j = w_i/s_k$, $I_j \in I \wedge IT_k = type\,(I_j)$).

The *completion time* of a task T_i executed on an instance I_j is the latest completion time of all its predecessors plus its expected TET:

$$end\,(T_i, k) = \begin{cases} t_i^j, & T_i = T_{entry}, IT_k = type(I_j); \\ \max\limits_{T_p \in pred(T_i)} \left\{ end\,(T_p, k) + t_i^j \right\}, & T_i \neq T_{entry}, IT_k = type(I_j). \end{cases}$$

In a single experiment, we are using a constant number of the same type VMs, which keep running. Thus, for an experiment that uses VM instances of type k, we define the workflow *makespan* $M = end(T_{exit}, k)$.

3.3 Experiment and Test Case Model

In order to model the environment-independent model, we define a set of *experiments* $EXP = \bigcup_{x=1}^{q} \{EXP_x\}$. Each element EXP_x is modeled as a triple $EXP_x(W, IT_k, v)$, which means that a workflow W is executed on a specific number of VMs v, all of the same type IT_k.

As we want to simulate Cloud's behavior, we repeat each experiment N times and we refer to each execution as a *test case*. Therefore, a *test case* ${}^xTC^c$ represents the c-th repetition of an experiment EXP_x. This means that an experiment can be considered as a matrix of N columns (workflow execution repetitions) and n rows (tasks within a workflow).

Let ${}^xt_i^c$ denotes the *measured TET* of a task T_i in a test case ${}^xTC^c$ of an experiment EXP_x. Analogue, the makespan of test case ${}^xTC^c$ will be denoted as ${}^xM^c$. As we want to analyze the distribution of makespans per experiment, we define the *TET's mean value* ${}^x\bar{t}_i = \frac{1}{N} \cdot \sum_{c=1}^{N} {}^xt_i^c$ of a task T_i and the *mean makespan* ${}^x\overline{M} = \frac{1}{N} \cdot \sum_{c=1}^{N} {}^xM^c$. Both mean values are defined for an experiment EXP_x.

4 Noise Simulation Model

In this section we present our new model of simulating a workflow execution in Cloud, which improves the accuracy through a much simpler approach that uses only one parameter for noising for all the tasks of a workflow - *Cloud noisiness*, instead of DynamicCloudSim's 13 parameters for heterogeneity and instability.

4.1 Workflow Noisiness

In order to define a model for the noisiness provided by the workflow, first we model the noisiness of a task $T_i \in T$ itself with the parameter *TET's deviation*

${}^{x}\rho_i^{cd}$, which is defined in (1) as a relative TET difference of task T_i in two test cases ${}^{x}TC^{c}$ and ${}^{x}TC^{d}$ of an experiment EXP_x with $c, d \in [1, N] \wedge c \neq d$.

$$ {}^{x}\rho_i^{cd} = \frac{|{}^{x}t_i^{c} - {}^{x}t_i^{d}|}{max({}^{x}t_i^{c}, {}^{x}t_i^{d})} \tag{1} $$

The TET's deviation ${}^{x}\rho_i^{cd}$ is used to introduce the *workflow noisiness* ${}^{x}\overline{\Delta}^{cd}$, which describes the total noisiness of all TETs within a workflow in two test cases ${}^{x}TC^{c}$ and ${}^{x}TC^{d}$ of an experiment EXP_x, as defined in (2). Formally, it represents a normalized mean TET's difference of all corresponding tasks T_i of the same workflow, in two test cases ${}^{x}TC^{c}$ and ${}^{x}TC^{d}$ of an experiment EXP_x.

$$ {}^{x}\overline{\Delta}^{cd} = \frac{1}{n} \cdot \sum_{i=1}^{n} {}^{x}\rho_i^{cd} \tag{2} $$

The *workflow noisiness* ${}^{x}\overline{\Delta}^{cd}$ can be used to extract noisiness of Cloud environment and includes vertical average of TETs of all tasks within a workflow W. ${}^{x}\overline{\Delta}^{cd}$ is an intermediate metric and serves as input for the *Cloud noisiness*, which is explained in the following subsection.

4.2 Cloud Noisiness

The workflow noisiness shows the instability of a workflow in two executions only. As Cloud environment is unstable, we want to take some average of a set of test cases for a single experiment EXP_x. Therefore, we introduce the *Cloud noisiness* ${}^{x}\overline{\Delta}$ of an experiment EXP_x. As defined in (3), ${}^{x}\overline{\Delta}$ represents the average experiments' makespan instability of all N repetitions (test cases) of an experiment EXP_x.

$$ {}^{x}\overline{\Delta} = \frac{1}{\frac{N \cdot (N-1)}{2}} \cdot \sum_{\forall c,d | 1 \leq c < d \leq N} {}^{x}\overline{\Delta}^{cd} \tag{3} $$

We measure the workflow noisiness ${}^{x}\overline{\Delta}^{cd}$ of each unique pair of test cases ${}^{x}TC^{c}$ and ${}^{x}TC^{d}$ for all N executions (test cases) of the same experiment EXP_x. Accordingly, the total number of unique pairs that can be generated from a set of N elements is $\binom{N}{2} = \frac{N \cdot (N-1)}{2}$. The *Cloud noisiness* ${}^{x}\overline{\Delta}$ includes horizontal average of all workflow executions (test cases) within a single experiment EXP_x.

4.3 Modeling the Noising

Now, after formally definition of cloud noisiness, we define how to model and add a noise in a simulation. The *noised TET* ${}^{x}\widetilde{\tau}_i$ of a task T_i is defined in (4), where $Gaussian(Mean, STDEV)$ represents a random function with Gaussian (normal) distribution.

$$ {}^{x}\widetilde{\tau}_i = (1 + {}^{x}\overline{\Delta} + Gaussian(0, \sigma_{{}^{x}\overline{\Delta}})) \cdot {}^{x}\bar{t}_i \tag{4} $$

Accordingly, the noise in (4) is modeled as a Gaussian distribution, where the mean value is the TET's mean value ${}^{x}\bar{t}_i$ of a task T_i, shifted with the Cloud noisiness ${}^{x}\overline{\Delta}$ in order to cover uncertain overheads, measured in the experiments. For noisiness, we use the standard deviation $\sigma_{{}^{x}\overline{\Delta}}$ of all workflow noisiness of a single cloud noisiness, which is determined by each pair of test cases TC^c and TC^d $(\forall c, d \mid 1 \leq c < d \leq N)$ of a single experiment.

5 Testing Methodology

In this section we present the testing methodology in order to evaluate our noising model in Sect. 6.

5.1 Synthetic Workflow

The synthetic workflow that is used in our experiments consists of two parallel sections (Second and Fourth) of same size, with three synchronisation tasks (First, Third and Fifth) in between, as depicted in Fig. 1. The chosen workflow structure is the result of workflow characteristics analysis of several well known workflows, such as *EPIGENOMICS* and *SIPHT* [14]. The workflow size is related to the number of tasks in the parallel sections. In the experiments we use a parallel section size of 13 (SYNWF/13) and 44 (SYNWF/44). We use two different workflow sizes in order to cover balanced and unbalanced executions, such that one is a prime number and the other is dividable by the numbers 2 and 4, corresponding to the number of instances. Additionally, we used instance number of 3, which is not a divisor of neither workflow sizes. With this workflow structure and selected parameters, we also want to investigate if there is a correlation between the workflow parameters and the execution environment (chosen resources and the inefficient workflow execution).

As workflow makespan consists of computations and file transfers (including both the network and I/O), we have chosen different file transfer to computation

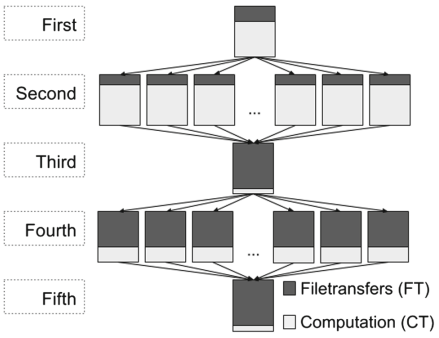


Fig. 1. The structure of the synthetic workflow with the file transfer to computation time ratios

Table 1. File transfer (FT) to computation time (CT) ratio in % for SYNWF/13 and SYNWF/44

Task type	SYNWF/13		SYNWF/44	
	FT[%]	CT[%]	FT[%]	CT[%]
First	30	70	30	70
Second	15	85	15	85
Third	90	10	95	5
Fourth	75	25	75	25
Fifth	90	10	95	5

time ratios for all five task types, as depicted in Fig. 1 and presented in Table 1. A file transfer describes the copying process of a set of files from one task to another and combines network bandwidth with I/O. The ratios of the First, Second and Fourth task types are assumed to be constant for different workflow sizes because the number and size of file inputs is not changing. This is different for the Third and Fifth task types, which are synchronization tasks and collect all the output files produced by the Second and Fourth tasks (parallel sections), correspondingly. Thus, in order to compensate this, we slightly increase the file transfer time for the workflow size 44. Moreover, the ratios of both parallel sections are inverse to each other.

5.2 Cloud Testing Environment

All Cloud experiments were executed in Amazon EC2 with the VM image *Amazon Linux AMI (ami-1ecae776)* in the availability zone *US East (N. Virginia)*. For the workflow execution and measurements of the TETs and makespan, we used the workflow execution engine Askalon [17]. We use two Amazon instance types $IT = \{t2.small, t2.medium\}$ as well as the number of instances $m = \{2, 3, 4\}$.

According the definition of experiments in Sect. 3.3, EXP_1 is defined as $EXP_1(SYNWF/13, t2.small, 2)$, which means that it executes the synthetic workflow with parallel section size 13, by using two small VMs. As we used two different workflows that are specified in the previous subsection, we define $q = 12$ experiments, and since we execute 20 repetitions (test cases) of each experiment, we execute a total of 240 test cases in Cloud. In order to cover different behavior of Amazon's EC2, we run all test cases of each experiments in the period of two weeks.

5.3 Simulation Testing Environment

In all our simulations, we used DynamicCloudSim, which extends CloudSim by adding features that allow a user to simulate the heterogeneity in Cloud described as the performance deviation of a resource, including *VM heterogeneity*, *host heterogeneity*, *File I/O heterogeneity*, and *Cloud instability*, *VM Stragglers* and *VM Failures*. DynamicCloudSim introduces 13 parameters that cover Cloud heterogeneity and instability, which are described in Table 2.

The same two workflows, two VM types and two, three, and four number of VMs are used to reproduce the same 12 experiments as in real Cloud. We execute $N = 20$ test cases per experiment in order to have equal number of experiments in real and simulated executions. Note that we consider the instance startup as warm up period and thus it is discarded in the experiments.

In order to evaluate our model, we compare it with the DynamicCloudSim's model (denoted as S_{sDCS}) with default values for Amazon EC2 (Table 2), which are based on other performance-based researches, experience, and assumptions.

Table 2. DynamicCloudSim heterogeneity parameter setup for S_{sDCS} and S_{noise}

Heterogeneity parameter	Description	S_{sDCS}	S_{noise}
cpuHeterogeneityCV	Randomize the power of the host	0.4	0
ioHeterogeneityCV	(CPU, I/O and bandwidth)	0.15	0
bwHeterogeneityCV		0.2	0
cpuNoiseCV	Randomize the performance characteristics	0.028	0
ioNoiseCV	of a VM (CPU, I/O and bandwidth)	0.007	0
bwNoiseCV		0.010	0
cpu/io/bw BaselineChangesPerHour	Randomize the dynamic changes of Cloud's performance	0	0
likelihoodOfStraggler	Probability of a VM being a straggler	0	0
stragglerPerformanceCoefficient	Diminished performance of a straggler	1	1
likelihoodOfFailure	Average rate of failure	0	0
runtimeFactorInCaseOfFailure	TET factor of a failed task	1	1

Our noisining model S_{noise}, adds noise to one parameter (TET) only. Thus, all other heterogeneity and noise related parameters are set to 0 in DynamicCloudSim. Note that also the *cpuNoiseCV* parameter is also set to 0, because we insert our noise through the Cloud noisiness parameter. Table 3 shows the measured Cloud noisiness ${}^{x}\overline{\Delta}$ and the corresponding standard deviations for S_{noise} in each experiment executed on Amazon EC2. It shows that EC2 provides computation instabilities from 8.9% up to 17.6% for various workflows and using different number of various instance types. The deviation of all test cases per a single experiment is in the range of 2.8% up to 11.2%.

Table 3. ${}^{x}\overline{\Delta}$ and $\sigma_{x\overline{\Delta}}$ values for S_{noise}

	W	2*S	3*S	4*S	2*M	3*M	4*M	W	2*S	3*S	4*S	2*M	3*M	4*M
${}^{x}\overline{\Delta}$	13	0.110	0.089	0.120	0.144	0.162	0.165	44	0.093	0.112	0.166	0.129	0.176	0.158
$\sigma_{x\overline{\Delta}}$		0.077	0.029	0.062	0.028	0.041	0.048		0.039	0.058	0.112	0.039	0.042	0.031

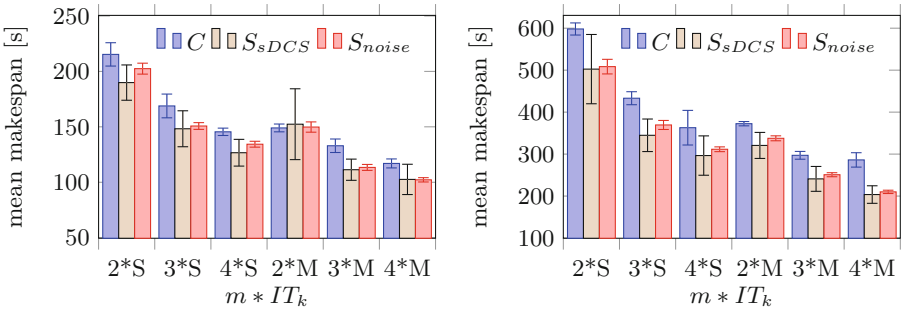
For a fair comparison, both simulations have equal base network, storage and computation speed. Moreover during the experiments in real Cloud, we did

not detect any VM failures, and therefore the straggler, failure and cpu/io/bw BaselineChangesPerHour parameters are set to 0 for both simulations.

6 Evaluation

In this section we present the results of a series of experiments to evaluate our noising model. The summary of the evaluation shows not only that our model is simpler, but it is more accurate than DynamicCloudSim.

Figure 2 shows the mean makespans of both workflows SYNWF/13 and SYNWF/44. The experiments are denoted as the product of number of instances (m) and the abbreviation of VM type (M for t2.medium and S for t2.small VM type). For example, $2 * S$ denotes the experiment that uses two small instances, while $4 * M$ is used for the experiment with four medium instances.



(a) SYNWF/13 (b) SYNWF/44

Fig. 2. Mean makespans with variations $\pm\sigma(mean\ makespan)$ of SYNWF/13 and SYNWF/44 using two, three or four instances of type t2.small (S) or t2.medium (M)

The mean values of makespans of all experiments with SYNWF/13 are depicted in Fig. 2a, along with the standard deviation. We observe that our S_{noise} model shows higher accuracy, that is, both higher trueness and precision, than S_{sDCS}. Increasing the number of small instances, S_{noise} improves its precision with 20.16% up to 71.61%, while S_{sDCS} has minimum 51.21% of precision offset compared to C. For all experiments, our noise model shows higher makespan trueness for real Cloud C, compared to S_{sDCS}. In detail, the trueness' offset of S_{noise} is between 6.0% and 10.7% for small instances, while S_{sDCS}'s is between 11.8% and 13.0%. Thus, S_{noise} is up to 7.0% better than S_{sDCS} for small instances. We observe a similar behavior with increasing number of medium instances, where the makespan trueness offset of S_{noise} is between 0.5% and 14.7%, while S_{sDCS}'s is between 2.3% and 16.3%. In the experiments with two medium instances, both simulations show higher makespan than C, but S_{noise} has still higher trueness with 0.5% offset.

Figure 2b shows the average makespan results for all experiments with the SYNWF/44 workflow. Similar to the experiments with the SYNWF/13 workflow, our model S_{noise} shows again higher accuracy than S_{sDCS}. The trueness of our model S_{noise} is better than S_{sDCS} for all experiments, while the precision is comparable, but slightly worse, only for experiments with 4 instances, and for all others our model S_{noise} is still better. In detail, S_{noise} shows between 15.29% and 20.59% higher standard deviation than C for 2 and 3 instances, while S_{sDCS} has up to 5 times higher offset. On experiments with 4 instances, S_{noise} shows between 76.94% and 86.21% precision offset, while S_{sDCS} has only up to 21.93%. Regarding the simulated makespans' trueness (closeness to the mean value), S_{noise} shows a better trueness than S_{sDCS} in all experiments. In detail, comparing the trueness offset of the simulated results and the real Cloud results, S_{noise} is still better with trueness offset between 9.4% and 26.6%, while S_{sDCS} has between 16.0% and 28.8%.

Comparing all experiments conducted with both workflows, S_{noise} shows higher and more closer precision than S_{sDCS} for all experiments with SYNWF/13. We observe similar behavior with SYNWF/44, except for experiments with 4 small and medium instances. Additionally, S_{noise} shows up to 9.2% higher makespans' trueness compared to S_{sDCS} for all experiments. We also observe that the precision of both models does not depend if the number of instances is a divisor or not of the workflow parallel section size, which can significantly reduce the number of experiments to determine the cloud noisiness.

7 Conclusion and Future Work

This paper presents a new simplified *Cloud noisiness* model for noising the workflow execution while it is simulated in order to behave as the real Cloud unstable environment. Instead of configuring dozens parameters in order to achieve a noised simulation, as it is required in DynamicCloudSim, our model configures only one - the Cloud noisiness. A series of experiments in Amazon EC2 that were reproduced in simulated environment show that our Cloud noisiness model simplifies the simulation configuration and improves the simulation trueness, and especially precision.

The main novelty in our model is the calculation of noisiness. Instead of using the normal distribution naively for the tasks' runtime, we shift the mean TET by the Cloud noisiness and then add a noise (deviation of the Cloud noisiness) of each task. The workflow noisiness smooths the impact of the workflow structure by calculating the average instability of all tasks in a workflow, while the Cloud noisiness estimates the environmental noise by calculating the horizontal average instability of the same task. With these two parameters, we inject not only the noise of a task itself, when being executed in Cloud, but we inject the impact of common tasks' noises within the workflow and environmental noise provided when a whole workflow is being repeatedly executed in that environment.

Although our Cloud noisiness approach requires several executions of a workflow in order to calculate the Cloud noisiness parameter, the results show that

the instability is not correlated to the parallel section size, that is, if we execute a workflow efficiently or inefficiently. However, the experiments show that the instability is highly correlated with the instance type and the number of instances. We will try to extend our model with other parameters in order to reduce this dependency and therefore the cost of learning the Cloud noisiness. However, our analysis shows that the performance instability of up to 17% between two experiments is comparable with the standard deviation of up to 11% between test cases within a single experiment.

Our noisiness model improved the simulator's trueness for all and the precision for most instance types and number of instances. As the results show that the instability is instance type dependent, we will extend our model towards modeling network, as well as I/O, and including them in order to improve the trueness and precision even more.

Acknowledgments. This work is being accomplished as a part of project *ENTICE: "dEcentralised repositories for traNsparent and efficienT vIrtual maChine opErations"*, funded by the European Unions Horizon 2020 research and innovation programme under grant agreement No. 644179.

References

1. Alkhanak, E.N., Lee, S.P., Rezaei, R., Parizi, R.M.: Cost optimization approaches for scientific workflow scheduling in cloud and grid computing: A review, classifications, and open issues. J. Syst. Softw. **113**, 1–26 (2016)
2. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M.: A view of cloud computing. Commun. ACM **53**(4), 50–58 (2010)
3. Basu, A., Fleming, S., Stanier, J., Naicken, S., Wakeman, I., Gurbani, V.K.: The state of peer-to-peer network simulators. ACM Comput. Surv. **45**(4), 46:1–46:25 (2013)
4. Bux, M., Leser, U.: DynamicCloudSim: Simulating heterogeneity in computational clouds. Future Gen. Comput. Syst. **46**, 85–99 (2015)
5. Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A., Buyya, R.: CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw.: Practice Exp. **41**(1), 23–50 (2011)
6. Chen, W., Deelman, E.: WorkflowSim: A toolkit for simulating scientific workflows in distributed environments. In: 2012 IEEE 8th International Conference on E-Science (e-Science), pp. 1–8, October 2012
7. Dejun, J., Pierre, G., Chi, C.-H.: EC2 performance analysis for resource provisioning of service-oriented applications. In: Dan, A., Gittler, F., Toumani, F. (eds.) ICSOC/ServiceWave -2009. LNCS, vol. 6275, pp. 197–207. Springer, Heidelberg (2010). doi:10.1007/978-3-642-16132-2_19
8. Depoorter, W., Moor, N., Vanmechelen, K., Broeckhove, J.: Scalability of grid simulators: an evaluation. In: Luque, E., Margalef, T., Benítez, D. (eds.) Euro-Par 2008. LNCS, vol. 5168, pp. 544–553. Springer, Heidelberg (2008). doi:10.1007/978-3-540-85451-7_58
9. Di, S., Cappello, F.: GloudSim: Google trace based cloud simulator with virtual machines. Softw. Pract. Exper. **45**(11), 1571–1590 (2015)

10. Fard, H.M., Ristov, S., Prodan, R.: Handling the uncertainty in resource performance for executing workflow applications in clouds. In: 9th IEEE/ACM International Conference on Utility and Cloud Computing, UCC 2016, pp. 89–98. ACM (2016). http://doi.acm.org/10.1145/2996890.2996902
11. Iosup, A., Yigitbasi, N., Epema, D.: On the performance variability of production cloud services. In: 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), pp. 104–113, May 2011
12. ISO: ISO 5725: 1994. http://www.iso.org/obp/ui/#iso:std:11833:en
13. Jackson, K.R., Ramakrishnan, L., Muriki, K., Canon, S., Cholia, S., Shalf, J., Wasserman, H.J., Wright, N.J.: Performance analysis of high performance computing applications on the Amazon web services cloud. In: 2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom), pp. 159–168, November 2010
14. Juve, G., Chervenak, A., Deelman, E., Bharathi, S., Mehta, G., Vahi, K.: Characterizing and profiling scientific workflows. Future Gen. Comput. Syst. **29**(3), 682–692 (2013). http://www.sciencedirect.com/science/article/pii/S0167739X12001732. Special Section: Recent Developments in High Performance Computing and Security
15. Malawski, M., Juve, G., Deelman, E., Nabrzyski, J.: Cost- and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds. In: Proceedings of the International Conference on HPC, Networking, Storage and Analysis, SC 2012, pp. 1–11 (2012)
16. Ostermann, S., Plankensteiner, K., Prodan, R., Fahringer, T.: GroudSim: An event-based simulation framework for computational grids and clouds. In: Guarracino, M.R., et al. (eds.) Euro-Par 2010. LNCS, vol. 6586, pp. 305–313. Springer, Heidelberg (2011). doi:10.1007/978-3-642-21878-1_38
17. Ostermann, S., Prodan, R., Fahringer, T.: Extending grids with cloud resource management for scientific computing. In: 2009 10th IEEE/ACM International Conference on Grid Computing, pp. 42–49. IEEE (2009)
18. Ristov, S., Mathá, R., Prodan, R.: Analysing the performance instability correlation with various workflow and cloud parameters. In: 2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP), pp. 446–453, March 2017
19. Schad, J., Dittrich, J., Quiané-Ruiz, J.A.: Runtime measurements in the cloud: Observing, analyzing, and reducing variance. Proc. VLDB Endow. **3**(1–2), 460–471 (2010)
20. Tchernykh, A., Schwiegelsohn, U., Alexandrov, V., ghazali Talbi, E.: Towards understanding uncertainty in cloud computing resource provisioning. Procedia Comput. Sci. **51**, 1772–1781 (2015). International Conference on Computational Science (2015)
21. Tian, W., Xu, M., Chen, A., Li, G., Wang, X., Chen, Y.: Open-source simulators for cloud computing: comparative study and challenging issues. Simul. Modell. Practice Theory **58**(Part 2), 239–254 (2015). http://www.sciencedirect.com/science/article/pii/S1569190X15000970. Special issue on Cloud Simulation
22. Wu, F., Wu, Q., Tan, Y.: Workflow scheduling in cloud: A survey. J. Supercomput. **71**(9), 3373–3418. http://dx.doi.org/10.1007/s11227-015-1438-4
23. Zhao, Y., Li, Y., Raicu, I., Lu, S., Lin, C., Zhang, Y., Tian, W., Xue, R.: A service framework for scientific workflow management in the cloud. IEEE Trans. Serv. Comput. **8**(6), 930–944 (2015)