RA^2: Predicting Simulation Execution Time for Cloud-Based Design Space Explorations

Ta Nguyen Binh Duong[*], Jinghui Zhong[*], Wentong Cai[*], Zengxiang Li[†], Suiping Zhou[‡]
[*] School of Computer Science and Engineering
Nanyang Technological University, Singapore
Emails: {donta, jinghuizhong, wtcai}@ntu.edu.sg
[†]Institute of High Performance Computing
Agency for Science, Technology and Research, Singapore
Email: liz@ihpc.a-star.edu.sg
[‡]Department of Computer Science
Middlesex University, United Kingdom
Email: s.zhou@mdx.ac.uk

***Abstract*—Design space exploration refers to the evaluation of implementation alternatives for many engineering and design problems. A popular exploration approach is to run a large number of simulations of the actual system with varying sets of configuration parameters to search for the optimal ones. Due to the potentially huge resource requirements, cloud-based simulation execution strategies should be considered in many cases. In this paper, we look at the issue of running large-scale simulation-based design space exploration problems on commercial Infrastructure-as-a-Service clouds, namely Amazon EC2, Microsoft Azure and Google Compute Engine. To efficiently manage cloud resources used for execution, the key problem would be to accurately predict the running time for each simulation instance in advance. This is not trivial due to the currently wide range of cloud resource types which offer varying levels of performance. In addition, the widespread use of virtualization techniques in most cloud providers often introduces unpredictable performance interference.**

In this paper, we propose a resource and application-aware (RA^2) prediction approach to combat performance variability on clouds. In particular, we employ neural network based techniques coupled with non-intrusive monitoring of resource availability to obtain more accurate predictions. We conducted extensive experiments on commercial cloud platforms using an evacuation planning design problem over a month-long period. The results demonstrate that it is possible to predict simulation execution times in most cases with high accuracy. The experiments also provide some interesting insights on how we should run similar simulation problems on various commercially available clouds.

***Keywords*—neural network, prediction, cloud-based simulations, resource-aware**

I. INTRODUCTION

Design space exploration (DSE) [1] is a process of finding the optimal configurations for a system which minimize or maximize the system's objective function [2]. However, due to the large search spaces present in most real-world systems, finding good configurations by manually setting system parameters' values is a tedious and time consuming task. DSE is usually treated as an automated calibration problem where the ideal configurations are found by running simulations of the actual systems. To achieve the speed needed for practical usage, DSE usually employs heuristics that prioritizes likely optimal configurations instead of doing exhaustive search over the entire design space.

In practice, for many complex systems, we still need to run a large number of simulations to evaluate candidate configurations, which is very time consuming. For instance, Russell et al. [1] noted that evaluating all building configurations for their energy consumption using a single core machine could take more than a year. To cope with such resource requirements, distributed/high performance computing infrastructures are usually employed. Cloud computing, due to the on-demand, elastic nature of its resources and the pay-as-you-go pricing model, might be preferred over other alternatives such as constructing an on-premise data centre. In this work, we consider cloud as the main execution infrastructure for DSE problems. Infrastructure-as-a-Service (IaaS) cloud platforms such as Amazon EC2 or Microsoft Azure provide a seemingly unlimited amount of computing power on an on-demand basis. However, system designers usually have limited budgets, so resource management in cloud-based DSE becomes an important problem. Given a certain budget to explore some design alternatives, the objective in many cases could be to finish the exploration in the shortest time possible.

Efficient resource management for cloud-based DSE is therefore needed to maximize designer's objective given budget or time constraints. Most cloud providers like EC2 charge coarse-grained billing units (e.g., an hour) which may incur resource under-utilisation and high cost if the designer does not plan the DSE execution properly. Sound resource management strategies, including virtual machine provisioning and simulation task scheduling, would benefit significantly from insights into the workload that has to be dealt with. For example, Genaud et al. [3] showed that if the running times of tasks in a workload are available, achieving cost-effective and time-efficient execution for many applications is possible using simple bin packing heuristics.

In the context of DSE problems, accurate prediction of running time for cloud-based DSE tasks (simulations) before they are actually executed is not trivial for several reasons. First, such prediction is application-dependent which means a certain level of knowledge regarding the simulation would be needed. A typical DSE simulation might have many configuration parameters with varying degrees of influence on its execution time. Another challenge is the resource multiplexing nature of most commercial cloud providers. A DSE simulation would be running on a virtual machine (VM) which shares the underlying physical resource with other co-located VMs via a layer called the hypervisor [4]. Despite recent advances regarding resource isolation in contemporary hypervisors, performance interference from noisy neighbors could pose a significant problem for accurate execution time predictions [5]. Last but not least, popular cloud platforms today offer a wide range of resource types which come with varying levels of performance. For example, the recently introduced t2 instances (from EC2) dynamically adjust a VM's share of CPU resource over time [6]. Such performance variability adds another layer of confusion to many existing performance predictors, which have been dealing with only application's parameters [7]–[9].

In this paper, we aim to enable efficient resource management for cloud-based DSE by taking the first step to examine whether it is possible to accurately predict simulation execution times on various production cloud platforms. The specific contributions of this paper are as follows.

- We propose a resource and application-aware (RA^2) approach to predict the execution time of DSE simulation tasks. Most existing work are only application-aware, which might not be adequate in cloud-based environments where performance variability is prevalent.
- We implement the RA^2 approach using an artificial neural network (ANN) based predictor. The predictor is augmented with a lightweight performance monitoring tool named PerfMon to detect if there are variations in the cloud resource's performance.
- We conduct extensive experiments to predict execution times of CPU-bound DSE simulation tasks on three most popular IaaS cloud providers. We use an agent-based crowd evacuation planning problem as a case study [10]. The obtained results have demonstrated the effectiveness of our approach. In particular, the predictor has close to 90% accuracy in most cases, when used in conjunction with data from PerfMon.

The rest of this paper is organized as follows. Section II summarizes key DSE concepts and motivates the simulation time prediction problem. Section III presents the prediction approach and techniques. Section IV describes experiments and results. Section V discusses related work, and Section VI concludes the paper and outlines some future work.

II. Cloud-based Design Space Exploration

In exploring a design space, a system $\mathcal{S}$ can be defined as an array of N configurable parameters ρ_i:

$$\mathcal{S} = \{\rho_1, \rho_2, \dots, \rho_N\} \tag{1}$$

If we assume that each parameter ρ_i has M possible values, then the number of possible configurations θ we might need to explore would be exponential: M^N. Therefore, finding optimal configurations in the search space requires the system designer to evaluate many candidate configurations which could be very time consuming and costly. A commonly used objective is to find the best configurations within a given budget, in a minimum amount of time.

In simulation-based DSE, the exploration process could be improved by employing: 1) a heuristics search algorithm to efficiently search for the optimal or near optimal configurations; 2) a simplified simulation model of the actual system to evaluate the candidate configurations; and 3) an automated framework which enables faster evaluation via scalable, high performance computing infrastructure [1].

In practice, a typical DSE implementation (see Figure 1) could be separated into two parts: *optimization loop* and *execution*. The optimization loop has a *Selector* which incorporates some search heuristics. The execution component consists of a *Resource Manager* which is responsible for executing candidate configurations.

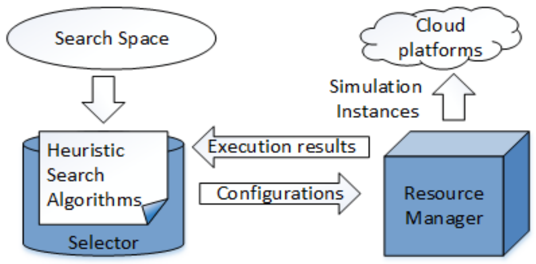

Figure 1. A typical DSE implementation: Selector selects some configurations and sends to Resource Manager to run. Selector waits until all selected configurations have been evaluated. DSE Resource Manager talks to cloud-based infrastructures to enable simulation executions.

Generally, DSE is an iterative process. Selector selects a batch of configurations to be considered in each iteration and sends to the Resource Manager. The latter then executes a simulation instance for each candidate configuration. After the completion of all simulation instances, Resource Manager feeds their outputs back to Selector which will decide whether a satisfying configuration has been found. If not, based on the results of the just evaluated set of configurations, Selector selects another bunch of candidates for evaluation in the next iteration. This process continues until the desired configuration has been found, or all the configurations in the search space have been evaluated, or there is no more budget/time. To speed up the search, Resource Manager needs to employ appropriate resource provisioning and scheduling techniques to run many configurations concurrently. A cloud-based Resource Manager administers VMs from either public or on-premise clouds to facilitate simulation executions. The main objective is to execute all simulations within a given budget in the shortest time possible.

Given the above objective, Resource Manager must effectively deal with the following issues in each DSE iteration: 1) to decide whether it is necessary to send simulation tasks to public clouds; 2) to decide how many VMs are needed and which cloud provider(s) to acquire VMs from; and 3) to schedule tasks onto the acquired resources. It is obvious that efficient provisioning and scheduling would benefit tremendously from accurate insights into task execution times.

III. Simulation Execution Time Prediction

A. Problem Analysis

Predicting execution times for many simulation instances on public clouds is not a trivial task due to several reasons:

- Such prediction could be very much application-dependent which means a thorough understanding regarding the simulation in question might be needed. This is not always possible since the system designer may not be the developer for the DSE's simulations. A typical DSE simulation might have many configuration parameters with varying degrees of influence on total simulation time. The problem here is, without deep domain knowledge, what should be the parameters that we need to choose to maximize prediction accuracy.
- Another challenge is the resource multiplexing nature of most commercial cloud providers. A DSE simulation would typically be running on a VM which shares the underlying physical resource with many other co-located VMs running potentially different applications. Despite recent advances in resource isolation in contemporary VM hypervisors, performance interference from noisy neighbors could still affect the accuracy of many performance predictions [5].
- Newly introduced cloud resource types with different performance guarantees may confuse predictors. For example, the t2 family of instances recently introduced by Amazon EC2 provide a baseline CPU performance with the ability of bursting using CPU credits [6]. When using this kind of instances, the CPU performance could vary over time, depending on the amount of CPU credits available for a VM. Traditionally, EC2 instance types such as m3 provide fixed performance, which makes it easier to carry out predictions.

In this paper, we propose an approach to predicting execution times for DSE simulations that is both resource and application aware. This approach, termed RA^2, utilises two sets of input data: the set of configuration parameters for a single simulation instance θ, and the current performance level of a particular cloud VM which will be used to run this simulation instance. Here we focus on CPU-bound workload, so the CPU performance of a given VM should be considered carefully. We are interested in prediction models which can capture potentially complex nonlinear relationships in multidimensional input data; and are able to produce a single output which is the predicted execution time.

B. Neural Network: Application-Aware Predictor

Artificial neural network (ANN) [11] is a predictive model that has inspirations from real-world biological neural networks. It performs well on function fitting problems, and is particularly well suited for dealing with non-linear problems which are prevalent in the real world, such as our simulation time prediction problem.

An ANN is organized as layers of interconnected nodes (i.e., neurons). The first layer is referred to as the *input layer* which takes input values (in our case, these are simulation parameters) and communicates to one or more *hidden layers* where the actual computation is carried out via weighted connections. After the computation in hidden layers is done, the output layer finalizes the results and returns the output, which is the predicted execution time in this case. The number of input nodes is equal to the dimension of the input data; whereas the number of nodes in the hidden layers could be varied. Since we need a single output value, the output layer in our ANN-based predictor has only one node, but more than one output is also possible.

The internal computation of each node in a network utilizes a network activation function (e.g., the sigmoid function) which takes results of the previous layer and connections' weights as inputs. After the computation, if the result exceeds the activation threshold value, it is propagated to the next layer. If an output value of the ANN-based predictor differs from the expected output in the training dataset, the difference is propagated back to the hidden layers to adjust their weights accordingly. This is to ensure that next time the predicted output could be closer to the expected output. This is referred to as back-propagational neural networks. In this paper, we consider some of the most commonly employed back-propagational ANN algorithms including Levenberg-Marquardt (LM), Bayesian Regularization (BR), Scaled Conjugate Gradient (SCG), and Resilient Backpropagation (RB) [12]. The Levenberg-Marquardt algorithm is suitable for most problems. Other problems with noisier data may consider the Bayesian Regularization algorithm, which might take a longer time but could obtain a better prediction.

C. PerfMon: Resource-Aware Prediction Augmentation

Up to this point, the ANN-based predictor is just application-aware, for it only looks at application-related information. More specifically, it uses historical data which consist of actual configuration parameter values and their corresponding execution time to build predictive models. It assumes the computing resource is available at a constant level. However, the performance fluctuation inherent in cloud resources and numerous VM instance types offered by cloud providers pose a significant challenge for accurate predictions.

Our resource-aware approach is to augment the ANN-based predictor with appropriate information regarding the cloud resources. Given a set of simulation instances with varying configuration parameters, the first step is to identify the set of cloud resources which will be used to run these simulations. These resources could have been acquired earlier by the DSE Resource Manager in some previous DSE iterations. The next step is to gauge the current level of performance for these resources, e.g., for EC2 t2 instances, a typical measure would be the remaining CPU credits for each VMs. A CPU credit would allow the VM to utilize the full performance of a CPU core in one minute [6].

We develop a lightweight performance monitoring tool named PerfMon to detect potential performance changes in the VMs which will be used to run DSE simulations. PerfMon is deployed onto each of the VMs acquired by DSE Resource Manager. In this paper, we only consider CPU-bound workload. Therefore, PerfMon is designed to provide an estimation of CPU availability for a particular cloud VM. For t2 instances on EC2, PerfMon can make use of the Amazon CloudWatch

API [13] to get the CPU credit balance of a VM. This could be a good indicator of how long a particular VM would run with a full CPU core. For VMs on Azure and Google Compute Engine, PerfMon runs a fixed number of floating point operations for a predefined period. If there are little CPU variations, each run in PerfMon would take about the same amount of time; otherwise, execution times of PerfMon would differ in different runs. PerfMon calculates the average of these execution times as an indicator of the VM's near-term performance.

PerfMon introduces minimal overhead when executing its floating point operations: one run takes about half a second when there is enough CPU credit, and around 5 seconds otherwise on an EC2 t2.micro instance. Nevertheless, running PerfMon on the target VM before every single prediction might not be necessary. For example, performance of t2 instances might not change very frequently. Figure 2 shows the time taken for each PerfMon run over a short period of time on an EC2 t2.micro VM. It is observed that after a period of good performance, execution times increase sharply. This is due to the VM running out of its initial CPU credit [6]. Its CPU performance has been reduced to around 10% of the full performance of a CPU core. By capturing these changes, PerfMon could enable more accurate execution time predictions.

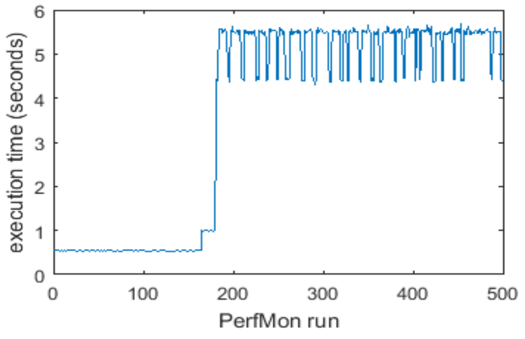

Figure 2. Running PerfMon on an EC2 t2.micro VM. The x-axis shows that there are 500 runs, while the y-axis shows execution time for each run in seconds. Around after the 180th run, each run takes significantly more time. This indicates significant reductions in CPU performance due to insufficient CPU credits.

The RA^2 prediction algorithm is as follows. Basically, after the ANN-based predictor provides initial execution time predictions for a new batch of simulations, PerfMon will be executed. Depending on the performance estimations, the predicted values given by the ANN model will be adjusted accordingly. For example, assume that the predicted execution time of a simulation s is t minutes initially. Now after running PerfMon, the algorithm could see that there's an available VM v to run s with a full CPU core's performance. In this case, there will be no adjustment to t, and v will be marked to note that it will run s later on. This is to update v's remaining CPU capacity which could be used again for subsequent predictions. However, if the only available VM v has a reduced level of CPU performance, say only 10% of a full CPU core, then t would need to be scaled up by a factor of around 10.

Based on predicted execution times, the DSE Resource Manager would then make a decision on whether to acquire additional VMs, or to release some of its current VMs to save cost. Detailed provisioning and scheduling algorithms are outside the scope of this paper. However, in Section IV, we consider a simple simulation task scheduling algorithm for comparing similar VM types from different cloud providers.

IV. Experiments and Results

In this section, we evaluate the accuracy of the ANN-based predictor, and PerfMon's augmentation mechanism. We first describe the case study used in our experiments. Experiment configurations and setup are elaborated next, followed by results and analysis.

A. *Evacuation Planning Using Agent-Based Crowd Simulation*

Building design for quickly evacuating crowds from an indoor place, e.g., train station, shopping mall, etc., in case of an emergency such as fire or explosion is an important problem in urban planning. In this paper, we consider this evacuation planning case study as a real-world DSE problem. Crowd evacuation simulation (CES) [10] models this problem, and provides a range of parameters for building designers to calibrate their designs.

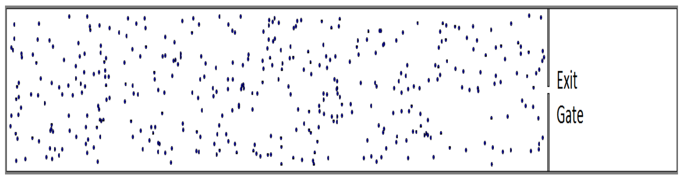

Figure 3. CES models an enclosed space such as an air-conditioned train station or shopping mall. The dots are actually autonomous agents for representing humans. Obstacles and other objects can be inserted into the space.

In CES, a number of autonomous agents are randomly deployed over an enclosed space. An example is shown in Figure 3. In this example, there is an exit gate on the right of the space. All of the agents will try to evacuate from the space via the exit (there could be more than one exits). The size of the space, the exit's width and position, and the number of agents could be varied. The social force model [14] is used to determine the collision avoidance movements of agents. The total evacuation time is defined as the duration from the beginning of the simulation to the time when all agents successfully evacuate the room. In this paper, our objective is to accurately predict the execution time for each instance of this simulation.

The simulation's configurable parameters with their corresponding value ranges are summarized in Table I. Some parameters listed in Table I are model parameters (e.g., *MA, MB, MK1, and MK2*); others are application-specific parameters (e.g., number of gates, gate position, number of agents). DSE can be used for model parameter calibration or for evaluating the effects of application parameters. Some of the important parameters are explained below.

- A: The number of autonomous agents.
- T: The time step to update position and velocity of agents.

Table I. CROWD EVACUATION SIMULATION PARAMETERS

Properties	Parameters									
	Agents (A)	Time Step (T)	CTime Step (CT)	MA	MB	MK1	MK2	V_0	V_{max}	Gates (G)	Gate x-Position (GP)
Min	10	0.0625	0.03125	100	0.01	10	10	0.5	1	1	21
Max	500	0.5	0.5	5000	0.2	300.000	300.000	2	4	10	28

- *CT*: The time step to calculate collision avoidance movements.
- *MA, MB, MK1, MK2*: The parameters for the basic social force model [14].
- V_0: The initial speed of agents.
- V_{max}: The maximum speed of agents.
- *G*: The number of gates.
- *GP*: The x-position of the right wall of the space, i.e., a variable to determine the length of the space.

B. Experiment Setup

1) Workload: Uniform random sampling over the space of configurable parameters is used to select a number of candidate configurations in our experiments. We choose 5100 different sets of CES configurations based on the value range of each parameter given in Table I. Subsequently, 4100 randomly chosen configurations are used for training the ANN-based predictor, while the remaining 1000 configurations are used for testing the prediction accuracy.

2) Cloud resources: We use various VM instance types from three most popular public IaaS cloud providers: Amazon EC2, Windows Azure and Google Compute Engine (GCE). Since CES is a single-threaded application, we only consider VM instance types having a single CPU core. Brief description regarding each instance type used in our experiments are summarized in Table II.

Table II. VM INSTANCE TYPES USED IN OUR EXPERIMENTS. ECU, ACU AND GCEU ARE CPU PERFORMANCE MEASURES FOR EC2, AZURE AND GCE, RESPECTIVELY.

EC2	t2.nano	t2.micro	t2.small	m3.medium
CPU capacity	Variable	Variable	Variable	3 ECU
Price ($/h)	0.01	0.02	0.04	0.098
Azure	**a1.basic**	**a0.std**	**a1.std**	**d1.v2**
CPU capacity	Variable	approx. 50 ACU	100 ACU	210 ACU
Price ($/h)	0.058	0.02	0.06	0.091
GCE	**f1-micro**	**g1-small**	**n1-standard-1**	
CPU capacity	Variable	1.38 GCEU	2.75 GCEU	
Price ($/h)	0.006	0.019	0.038	

On Amazon EC2, we use four different instance types: t2.nano, t2.micro, t2.small and m3.medium. These are the instances with one CPU core. The t2 instances's performance is governed by their CPU credits [6]. Each t2 instance starts with an initial CPU credit balance, and then continuously receives more CPU credits per hour at a rate depending on instance size. Once a t2 instance has run out of credit, its CPU performance would fall back to a baseline level, which again depends on the instance's size. The m3.medium instance, on the other hand, provides a fixed single-core CPU performance which is equivalent to 3 EC2 Compute Unit (ECU) at a more expensive price per hour than those of the t2 instances.

For Windows Azure, we also use four single-core instance types belonging to different pricing tiers [15]. The a0 and a1 instances in the standard tier (denoted as a0.std and a1.std in this paper) provide approximate performance of around 50 and 210 Azure Compute Units (ACU), respectively. We also consider the a1 instance type from the basic pricing tier (denoted as a1.basic). This instance type does not provide CPU performance guarantees. Lastly, the instance type d1.v2 is from the optimized compute tier which provides excellent CPU performance for higher pricing.

GCE provides three instance types having a single CPU core [16]. f1-micro has the most attractive price, but it is a shared CPU instance whose performance could vary significantly. g1-small and n1-standard-1 have around 1.38 and 2.75 Google Compute Engine Units (GCEU), respectively.

3) Performance measure: In this paper, we use the mean percentage error (mpe) [17] as the performance measure. $1 - mpe$ is used for measuring prediction accuracy (larger is better).

C. Results and Analysis

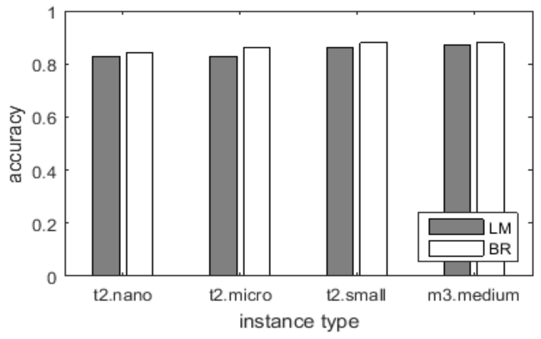

Figure 4. Prediction accuracy on EC2 instances is over 80% in all cases. We can observe that BR algorithm performs better than LM in most cases. The more expensive instances, e.g., m3.medium have better prediction accuracy due to performance stability.

The accuracy in predicting simulation execution times on different EC2 instance types is shown in Figure 4. It is not surprising to see that our ANN-based predictor performs best with m3.medium instances, since these instances have a fixed CPU performance (each has 3 ECU). The accuracies are 87% and 88% for the Levenberg-Marquardt (LM) algorithm and Bayesian Regularization (BR) algorithm, respectively. These results serve as a confirmation that our predictor can model the non-linear relationship between many simulation's parameters and the corresponding execution time. In addition, we note that the BR algorithm performs better than the LM algorithm in most cases. We have also evaluated the Scaled Conjugate Gradient (SCG) and Resilient Backpropagation (RB) algorithms, but their accuracies are not comparable to those of LM and BR. For clarify, we omit SCG's and RB's results in this section.

Although the first three instance types, t2.nano, t2.micro and t2.small, all have variable CPU performance, we can observe that the overall accuracy for all cases is well over 80%. This is mainly because our lightweight monitoring tool PerfMon is able to capture the levels of CPU performance variation as another set of input to augment the initial prediction results from the ANN-based predictor. As a result, even when using the cheapest instance type which is t2.nano, our predictor can achieve an accuracy of around 84%.

To verify the effectiveness of PerfMon, we also run the predictor without PerfMon's input. The difference obtained is very significant. For instance, without PerfMon, when using t2.nano instance, the prediction errors (measured by mpe) are around 115% and 100% for LM and BR algorithm, respectively. Note that an mpe higher than 100% means negative accuracy. Such large errors might mislead cloud resource provisioning algorithms, which may lead to high resource over/under-utilisation. Similar results have been obtained when predicting execution times on t2.micro and t2.small instances without PerfMon.

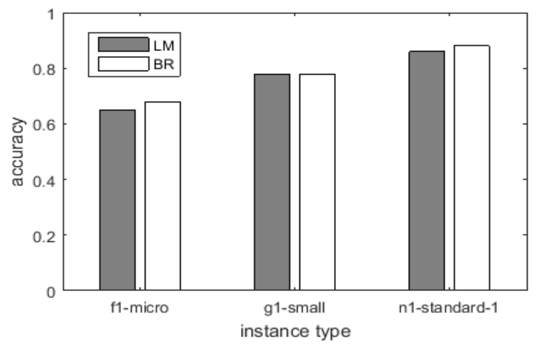

Figure 5. Prediction accuracy on GCE instances. The prediction performance is quite low for f1-micro, which is the cheapest instance type on GCE.

The prediction result for GCE is shown in Figure 5. In general, the accuracy is around 80% and above for g1-small and n1-standard-1 instance types, with or without PerfMon, mainly due to their relatively stable performance. However, predictions on the cheapest instance type, f1-micro, do not seem to be very accurate. A more detailed look into the performance of this particular instance type with PerfMon is shown in Figure 6. We notice that f1-micro's CPU performance fluctuates significantly and very frequently. This makes it very hard for any prediction algorithms. Even if we run PerfMon before every single prediction, the instance's performance could go up or down considerably right in the next moment. Therefore, our recommendation is that, for this kind of CPU-bound workload, it might not be advisable to use GCE's f1-micro instance type due to unpredictable performance fluctuations.

The prediction accuracy for simulations running on Microsoft Azure is shown in Figure 7. We can observe that even for one of the cheapest instance types, a0.std, the accuracy level is also very high (around 87-88%). This is mainly due to the narrow range of performance fluctuations, as demonstrated in Figure 8. The question is therefore: are Azure instances more suitable to run CPU-bound workloads such as the crowd simulations used in this paper?

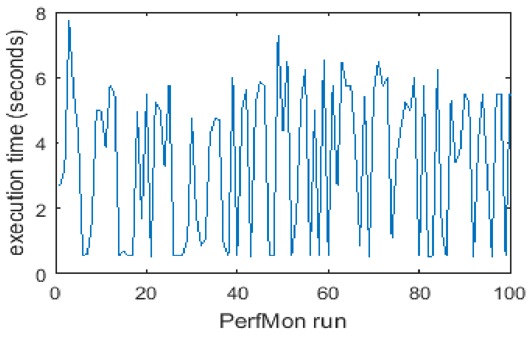

Figure 6. Running PerfMon on a GCE f1-micro instance before each simulation execution. The x-axis shows that there are 100 runs, while the y-axis shows execution time for each run in seconds. We observe significant and frequent fluctuations in CPU performance for this instance type.

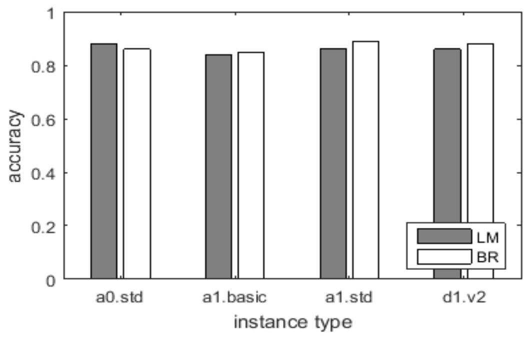

Figure 7. Prediction accuracy on Azure instances is very high in all cases. This demonstrates the stability in CPU performance of Azure instances including the cheapest instance type.

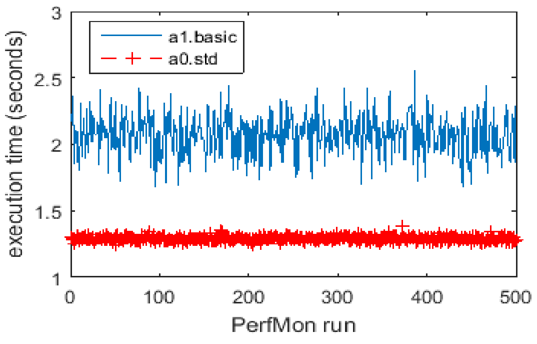

Figure 8. Running PerfMon on Azure's a0.std and a1.basic instances. The x-axis shows that there are 500 runs, while the y-axis shows execution time for each run in seconds. The CPU performance fluctuation is more pronounced in a1.basic compared to that in a0.std.

We conduct more experiments using a simple scheduling algorithm to compare the actual overall simulation execution times on Azure's a0.std and EC2's t2.micro VMs. These two are chosen since they have the same cost per hour. The scheduling algorithm's aim is to minimize overall execution time for a batch of simulations by balancing the workload across number of VMs, using each simulation's predicted execution time. Given a budget of $0.4, we can acquire either

20 t2.micro or 20 a0.std VMs for one hour (for EC2, you can only run up to 20 t2 instances simultaneously). The overall execution time is obtained using the actual execution time of each simulation. The result is quite interesting: 20 t2.micro VMs completed the execution of 1000 CES simulations in about 10 minutes; while it took 20 a0.std VMs around 52 minutes.

However, if the t2.micro VMs are not newly acquired from EC2, i.e., their CPU credits might have run out during previous DSE iterations, the a0.std VMs would have the advantage. In another experiment, we execute the same 1000 simulations on a set of 20 t2.micro VMs whose initial CPU credits have been fully depleted. This time round, t2.micro VMs took around 97 minutes to complete, which is 87% slower compared to the time given by a0.std VMs. The cost of t2.micro VMs is now also over the budget, as with $\$0.4$ we can only get 20 t2.micro VMs for 60 minutes. From these experiments, we can see that the accuracy of our RA^2 approach enable the DSE Resource Manager to make more informed scheduling and VM provisioning decisions, e.g., whether to acquire new VMs, if yes then from which cloud providers, or to continue with existing resources. Such decisions are the key factors in determining if a DSE execution could finish on time or within the given budget, i.e., meeting designers' objectives.

Last but not least, we consider the overhead of PerfMon with respect to the overall resource consumption for simulation time prediction and execution. We measure such overhead by first obtaining the total time that PerfMon takes to gauge CPU performance before predictions, denoted as t_p. We then compute the overhead as $\frac{t_p}{t_p + t_s}$, where t_s is the total time for actual simulation executions.

Table III. OVERHEAD OF PERFMON ON VARIOUS INSTANCE TYPES

t2.nano	**t2.micro**	**t2.small**	**f1-micro**	**a0.std**
3.9%	4.2%	4.2%	4.3%	3.2%

The average overhead for the entire experiment is reported in Table III. It can be observed that even if we run PerfMon before every single prediction, its average overhead is quite low, which is around 3-4% for most of the cases. We should also note that this is considered the worst case overhead, since we might not need to run PerfMon all the time for the more stable instance types such as t2.small or Azure standard instances like a0.std. We believe that our RA^2 approach could be a practical solution towards more accurate predictions of execution time on cloud resources.

V. RELATED WORKS

Predicting task execution time has been well considered in previous research, for example [7]–[9]. Chtepen et al. [7] noted that prior knowledge of the execution times is essential for efficient application scheduling, and proposed an online prediction method based on extrapolation. The proposed method improved the scheduling performance up to 15% for three different workflow systems. Marco and Vecchiola [8] proposed an execution time prediction algorithm for iterative parallel applications and were able to reduce up to 35% of runtime over-estimations with a 7% prediction error. Wang et al. [9] made use of the linear relationship between task size and execution time to obtain estimations for MapReduce jobs in order to improve scheduling performance. These approaches have been focusing more on accurately predicting execution times without much consideration for the underlying computing resources. We note that performance prediction for cloud-based applications is a more challenging problem due to performance variability and diverse cloud resource configurations.

Recently, there have been a growing interest in the area of performance prediction for applications running on public cloud infrastructures. CloudProphet [18] considered the problem of selecting the best-performing cloud providers for a given application. Hence, its focus is on predicting an applications performance when running on a chosen cloud platform, without actual deployments due to cost or security concern. This could be done by emulating the application's behavior in the cloud. Similar approaches to [18] include [19], [20]. On the other hand, empirical approaches including [21] and [22] evaluate the application's performance on actual cloud infrastructures. Their focus is more on providing automated methods to deploy and test applications using synthetic workloads. In this work, we collect actual, historical runtime data for a limited set of our application configurations running on commercial cloud providers. Such data will then be used to build predictive models. Finally, we incorporate online monitoring to obtain more accurate predictions before actual application deployment.

Mozafari et al. [23] builds models to predict performance and resource utilisation for cloud-based database applications. Such applications have many different workload characteristics from simulation-based DSE problems. More recently, Gonalves et al. [24] considered a new approach called performance inference to predict an application's performance when running on public clouds. The authors argued that it should be possible to establish a relationship between different resource configurations offered by an IaaS provider and the actual resource capacity. Such relationship would enable more accurate predictions for the expected performance of an application, given a certain resource configuration and workload. The prediction still needs to use historical data regarding the application's actual performance under other but related resource configurations/workloads. Our approach is different in that we use frequent performance monitoring to adjust initial predictions based on historical data when needed.

In [25], the authors proposed an application-agnostic performance modeling for cloud-based applications. Such a model if successfully implemented would potentially be applicable to a wide range of applications. This is an interesting direction that might need more consideration in our future work; as it could address public cloud's inherent performance variability to enable better prediction accuracy. However, the idea presented in [25] is still preliminary and not much results have been reported.

There have been not much work in the area of predicting execution time for simulations. [26] and [27] investigated load prediction techniques in HLA-based simulations. The aim is to enable dynamic load balancing of distributed simulations to reduce the overall execution time. The approach presented in [27] is specific to HLA-based distributed simulations. Our focus in this paper is on predicting the execution times for many single-core simulations so that more efficient resource provisioning algorithms could be implemented.

VI. Conclusion & Future Work

In this paper, we have looked at the issue of executing simulation-based design space exploration problems over commercially available cloud infrastructures. Such computing platforms offer cost-efficient execution for various application domains via a pay-per-use model; however the performance variability makes it hard to carry out cost optimization in many cases. We have proposed a hybrid approach to enable more accurate predictions of simulation running time on cloud resources. The approach combines a neural network based predictor, and a lightweight monitoring tool to capture performance fluctuations with negligible overhead. Data obtained from the monitoring tool is then used to augment the initial prediction results if necessary.

We have conducted extensive experiments on three of the most popular cloud platforms, namely Amazon EC2, Microsoft Azure and Google Compute Engine, using an actual crowd evacuation simulation as a case study. The results obtained demonstrated the effectiveness of our approach, with prediction accuracy of over 85% in most cases. Our next step is to investigate some application-agnostic prediction approaches. This would make it possible to accurately predict the performance for a wide range of applications. Another direction would be developing more effective cloud resource provisioning and scheduling strategies, leveraging the predicted execution times, for some real-world DSE problems such as building design and evacuation planning.

Acknowledgment

The authors would like to thank Mr. Murat Karakus for his initial contributions and help in this paper.

References

[1] R. Russell, Z. Hayes, and F. Stuart, "Large-scale building simulation using cloud computing for estimating lifecycle energy consumption," *Canadian Journal of Civil Engineering*, vol. 41, no. 3, pp. 252–262, 2014.

[2] A. D. Pimentel, M. Thompson, S. Polstra, and C. Erbas, "Calibration of abstract performance models for system-level design space exploration," *J. Signal Process. Syst.*, vol. 50, no. 2, pp. 99–114, 2008.

[3] S. Genaud and J. Gossa, "Cost-wait trade-offs in client-side resource provisioning with elastic clouds," in *IEEE International Conference on Cloud Computing*, 2011, pp. 1–8.

[4] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," *SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 164–177, 2003.

[5] X. Chen, L. Rupprecht, R. Osman, P. Pietzuch, F. Franciosi, and W. Knottenbelt, "Cloudscope: Diagnosing and managing performance interference in multi-tenant clouds," in *23rd IEEE MASCOTS*, 2015, pp. 164–173.

[6] P. Leitner and J. Scheuner, "Bursting with possibilities - an empirical study of credit-based bursting cloud instance types," in *8th IEEE/ACM International Conference on Utility and Cloud Computing,*, 2015, pp. 227–236.

[7] M. Chtepen, F. H. Claeys, B. Dhoedt, F. Turck, J. Fostier, P. Demeester, and P. A. Vanrolleghem, "Online execution time prediction for computationally intensive applications with periodic progress updates," *J. Supercomput.*, vol. 62, no. 2, pp. 768–786, Nov. 2012.

[8] M. A. S. Netto, C. Vecchiola, M. Kirley, C. A. Varela, and R. Buyya, "Use of run time predictions for automatic co-allocation of multi-cluster resources for iterative parallel applications," *Journal of Parallel and Distributed Computing*, 2011.

[9] G. Wang, A. Khasymski, K. R. Krish, and A. R. Butt, "Towards improving mapreduce task scheduling using online simulation based predictions," in *Parallel and Distributed Systems (ICPADS), 2013 International Conference on*, 2013, pp. 299–306.

[10] J. Zhong, W. Cai, L. Luo, and M. Lees, "Ea-based evacuation planning using agent-based crowd simulation," in *Proceedings of the 2014 Winter Simulation Conference*. IEEE Press, 2014, pp. 395–406.

[11] B. Yegnanarayana, *Artificial neural networks*. PHI Learning Pvt. Ltd., 2009.

[12] *The Neural Network Toolbox*. The MathWorks Inc., 2011.

[13] Amazon, "Amazon cloudwatch documentation," https://aws.amazon.com/documentation/cloudwatch, 2016.

[14] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical review E*, vol. 51, no. 5, p. 4282, 1995.

[15] Microsoft, "Virtual machines pricing," https://azure.microsoft.com/en-us/pricing/details/virtual-machines, 2016.

[16] Google, "Google compute engine pricing," https://cloud.google.com/compute/pricing, 2016.

[17] C. Tofallis, "A better measure of relative prediction accuracy," *J Oper Res Soc*, vol. 66, no. 8, pp. 1352–1362, 2015.

[18] A. Li, X. Zong, S. Kandula, X. Yang, and M. Zhang, "Cloudprophet: towards application performance prediction in cloud," in *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, 2011, pp. 426–427.

[19] F. Fittkau, S. Frey, and W. Hasselbring, "Cdosim: Simulating cloud deployment options for software migration support," in *6th IEEE International Workshop on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA)*, 2012, pp. 37–46.

[20] G. Jung, T. Mukherjee, S. Kunde, H. Kim, N. Sharma, and F. Goetz, "Cloudadvisor: A recommendation-as-a-service platform for cloud configuration and pricing," in *9th IEEE World Congress on Services*, 2013, pp. 456–463.

[21] M. Cunha, N. Mendonca, and A. Sampaio, "A declarative environment for automatic performance evaluation in iaas clouds," in *6th IEEE International Conference on Cloud Computing*, 2013, pp. 285–292.

[22] D. Jayasinghe, G. Swint, S. Malkowski, J. Li, Q. Wang, J. Park, and C. Pu, "Expertus: A generator approach to automate performance testing in iaas clouds," in *5th IEEE International Conference on Cloud Computing*, 2012, pp. 115–122.

[23] B. Mozafari, C. Curino, and S. Madden, "Dbseer: Resource and performance prediction for building a next generation database cloud," in *CIDR*, 2013.

[24] M. Gonalves, M. Cunha, N. C. Mendona, and A. Sampaio, "Performance inference: A novel approach for planning the capacity of iaas cloud applications," in *8th International Conference on Cloud Computing*, 2015, pp. 813–820.

[25] S. Imai, S. Patterson, and C. A. Varela, "Cost-efficient elastic stream processing using application-agnostic performance prediction," in *16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (to appear)*, 2016.

[26] R. Alkharboush, R. E. De Grande, and A. Boukerche, "Load prediction in hla-based distributed simulation using holt's variants," in *IEEE/ACM 17th International Symposium on Distributed Simulation and Real Time Applications*, 2013, pp. 161–168.

[27] R. E. De Grande, A. Boukerche, and R. Alkharboush, "Time series-oriented load prediction model and migration policies for distributed simulation systems," *IEEE Transactions on Parallel and Distributed Systems*, no. 99, 2016.