ComplexCloudSim: Towards Understanding Complexity in QoS-Aware Cloud Scheduling

Huankai Chen
School of Computing
University of Kent
Canterbury, UK

Frank Z Wang
School of Computing
University of Kent
Canterbury, UK

***Abstract*—The cloud is generally assumed to be homogeneous in most of the research efforts related to cloud resource management and the performance of cloud resource can be determined as it is predictable. However, a plethora of complexities are associated with cloud resources in the real world: dynamicity, heterogeneity and uncertainty. For heterogeneous cloud resources experiencing vast dynamic changes in performance, a critical role is played by the statistical characteristics of execution times, related to different cloud resources, to facilitate decision making in management. The cloud's performance can be considerably influenced by the differences between the estimated and actual execution times, which may affect the robustness of resource management systems.**

Limitation exists in the study of cloud resource management systems' complexities even though extensive research has been done on complexity issues in various fields from decision making in economics to computational biology. This paper concentrates on managing the research question regarding the complexity's role in QoS-aware cloud resource management systems. We present the ComplexCloudSim. Here, CloudSim, a popular simulation tool-kit, is extended through modelling of complexity factors in the cloud, including dynamic changes of run-time performance, resource heterogeneity, and task execution times' uncertainty. The effects of complexity on performance within cloud environments are examined by comparing four widely used heuristic cloud scheduling algorithms, given that the execution time information is inaccurate. Furthermore, a damage spreading analysis, one amongst the available complex system analysis methods, is applied to the system and simulations are run to reveal the system's sensitivity to initial conditions within specific parameter regions. Finally, how small of a damage can spread throughout the system within the region is discussed as well as research is done for the potential ways to avoid such chaotic behaviours and develop more robust systems.

***Keywords*—Cloud Scheduling; Damage Spreading; QoS; Complexity; Chaotic Behaviour; Cloud Simulation**

I. Introduction

The widely popular pay-as-you-go service has been enabled by Cloud Computing [1], which provides access to a shared pool of physical/ virtual, dynamically heterogeneous and scalable computational resources. Computational resources of any scale can be used in a rented module as per need through commercial cloud providers such as Microsoft Azure, Amazon (AWS), Rackspace Open Cloud and Google Compute Engine, which has been made possible by Infrastructure-as-a-Service (IaaS) model of Cloud Computing. Since availing these services on-demand is convenient, over the last years, the use of Cloud Computing has grown exponentially.

Both industry and academia require tailored cloud applications (customised) to meet their demands and use cloud resources efficiently. The main question here is:

"How should map-reduce alike groups of tasks be scheduled in the complex cloud environment that is reliable and efficient while meeting the application requirement for QoS? [2]"

The Cloud Computing community has been facing a real challenge with the above question. The reported scientific advances in both software platform development and Cloud Computing that enable fast data processing in the cloud is certainly a good news. Successful deployment of analysis engines such as Hive, Dremel, MapReduce, Spar and Impala has helped to run analysis jobs in short time across thousands of cloud resources [11]. However, adaptively scheduling groups of tasks based on dynamic changes in resource performance has been a challenge and remains unsolved. Scheduling is a vital mechanism for many cloud analysis engines. Unfortunately, the performance of rented cloud resource is not familiar with the available cloud scheduling systems, as the characteristic is subject to change dynamically, making it difficult to quantify during run-time. Cloud resources are homogeneous and the performance of resources does not change as assumed by most of the current scheduling solutions. In real-world heterogeneous cloud environment, this results in poor performance.

Scheduling comes under NP-complete problem and its complexity increases significantly in a heterogeneous cloud environment [15]. In the simplest form, scheduling, by just allocating appropriate resources based on availability to the incoming tasks, can be performed in a blind way. Nevertheless, advanced and sophisticated schedulers are significantly more reliable and efficient. Moreover, in general, it is expected that schedulers would react to the cloud resource's dynamic performance, most probably by examining the current CPU load of resources [10]. Also, to deal with the massive scale of the cloud, schedulers have to be easily distributed, have low overhead and lightweight.

First, this paper presents an extension to CloudSim [9], i.e. ComplexCloudSim, which offers many capabilities to model the complexity associated with the heterogeneous cloud environment. Then, four heuristics cloud scheduling algorithms are compared by running simulation (as presented in Section III) to demonstrate how resource complexity can make the scheduling system less robust. Section IV presents a damage spreading analysis model for the complex cloud to reveal the

cloud's sensitivity to initial conditions in specific parameter regions. Thus, such hidden chaotic behaviour present in the cloud scheduling system as a result of complexity is discussed. Finally, Section V provides the conclusion for this paper.

II. ComplexCloudSim: Heterogeneity, Dynamicity and Uncertainty

The cloud can be used to share different types of resources, which are typically accessed through applications running in the cloud. A typical cloud scenario can be an application that can generate several jobs. This application may already have sub-tasks that need to be resolved. Each sub-task is sent to a resource for resolving by the scheduling system. In a simple scenario, adequate resources needed to execute the sub-tasks are decided by the user; however, in general, the application will require schedulers that can efficiently and automatically find the most appropriate resources for completing a group of tasks.

One of the most studied research topics in the optimisation community is the scheduling issue related with cloud computing [3] [4] [5]. However, the problem becomes more challenging due to several complexity factors such as:

- **Heterogeneity** : The versatility of the current cloud infrastructures is limited. A crucial feature that needs to be taken into consideration in any cloud system is heterogeneity. Now, a single physical host can run multiple virtual machines (VMs) simultaneously, spurred by the development of virtualisation technology. Nevertheless, virtualisation comes with new challenges that hamper resource scheduling in clouds. This is because of multiple VMs present in the system that share hardware resources (e.g. memory, CPU, network, I/O, etc.) of a physical machine. In such a scenario, accurate measurement of the rented VMs actual performance is difficult. For e.g., in Amazon EC2, instead of fixed performance measures, compute units determine the provisioning of resources to virtual machines. The level of computing power required for provisioning compute unit varies with different host machines, which effectuate heterogeneity amongst VM performance [6]. This suggests that cloud is never homogeneous but always heterogeneous in the real world.
- **Dynamicity** : Another important complexity factor inherent to cloud computing is dynamic changes in resource performance at runtime [7]. In the real world scenario, over- or under-provisioning of resource, hardware/software failures, application misbehaviours and resource CPU overload can lead to such dynamicity of resource performance. The amount of running jobs assigned may also affect the cloud resource and may exhibit local activity. This leads to the creation of complexity. Moreover, the complexity level related to resource dynamicity increased with sharing of common underlying hardware infrastructures with other VMs.
- **Uncertainty** : The availability of complete information about the state of cloud resources is assumed by most of the research efforts related to scheduling. However, in cloud computing, during provisioning, uncertainty can exist between the ready time and the computing capacity of a resource [8]. We argue that the main issues with cloud computing is such uncertainties that bring additional challenges in execution time prediction of tasks, which is vital for many scheduling algorithms. There can be drastic changes in resource states in cloud environments. In most cases, obtaining exact knowledge about a resource is almost impossible. Accurate estimation of runtime tasks, performing prediction correction, undertaking prediction fall-back, improving prediction by historical data, etc. are difficult to execute. Significant uncertainty may rise due to imprecise execution of prediction times in scheduling performance.

A. CloudSim

For Cloud Computing infrastructures, CloudSim is a popular framework to execute simulation of resource scheduling. Mentioning the main entities/concepts regarding CloudSim, in terms of terminology, is vital to introducing it:

- **Datacenter** includes a set of physical hosts that can either be heterogeneous or homogeneous based on hardware configurations (memory, CPUs, storage and bandwidth) and it acts as Cloud Provider. It facilitates resource provision to cloud users.
- **Host**, a physical machine, is defined through the amount of memory present, the list of CPUs (and their types), storage as well as allocated bandwidth. A host allows managing VMs based on a specified VM allocation policy.
- A Cloud Host component manages and hosts the Virtual Machine (**VM**).
- **Cloudlet** is a job assigned by the Cloud User to run on the cloud. A job can be defined by its resource requirement (the number of cores and the amount of memory needed for performing the job), length (millions of instructions), dependencies and type (MapReduce jobs include Map tasks and reduce tasks).
- A **Broker** is the mediator that negotiates between cloud providers and cloud users. it acts on behalf of the cloud user to identify suitable resources that can be obtained from the cloud provider. Broker undertakes online negotiations that are directed towards allocation of resources to meet QoS needs of the user application. Cloudlets are then sent by the broker for scheduling available resources under defined scheduling policies.
- **CloudletScheduler** allows determining the processing power shared amongst Cloudlets based on available resources. Different scheduling policies can be used for implementation of this scheduler.

The hosts and VMs computational capabilities are measured in terms of million instructions per second per core (MIPS) in CloudSim and most of its extensions [12] [14] [13]. Throughout the CloudSim simulation, this measurement plays a crucial role. Provisioned virtual machines are assumed to be stable and predictable, in terms of their performance,

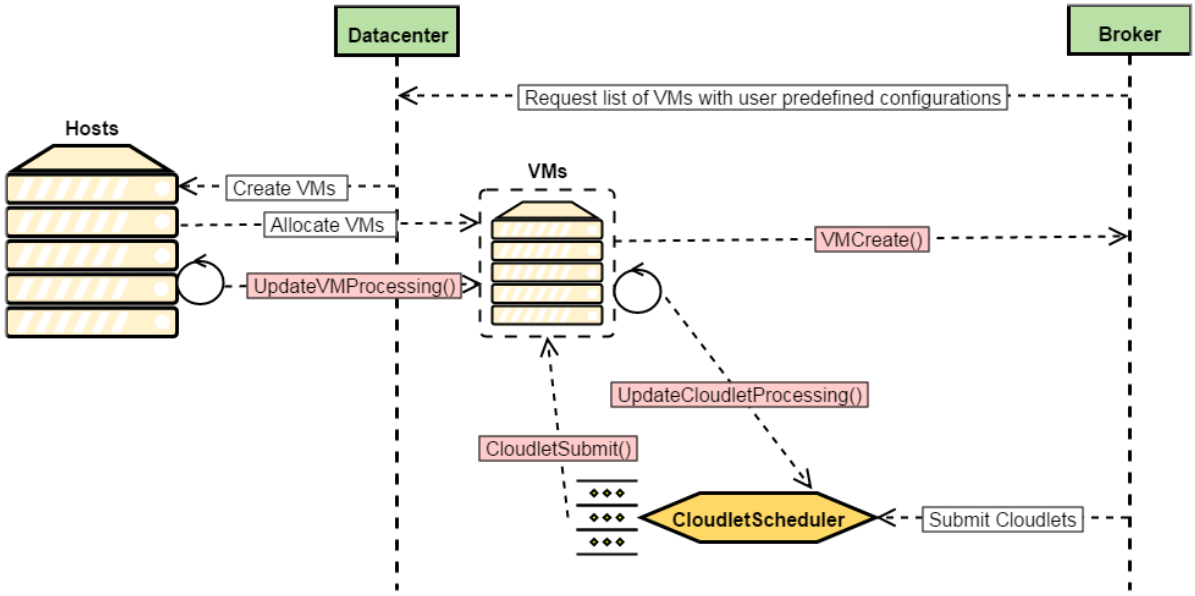

Fig. 1. CloudSim : Simulation Flow Chart

by CloudSim. Guaranteed performance is delivered by VMs, characterised as a fixed amount of MIPS. During a simulation, such a performance does not change, as presented in Figure 1. However, these assumptions do not hold when an actual cloud environment such as Amazon EC2 is used. Even though a certain core speed for each provisioned VM is guaranteed by most cloud providers, the runtime CPU utilisation of the host and the underlying physical hardware is assigned to determine the actual performance of a given VM. Thus, CloudSim may fail to efficiently simulate the cloud environment's complexity due to such incorrect assumptions.

B. ComplexCloudSim

This section explains how cloud simulations can be affected by complexity. This was derived based on a motivational example and a study employing four popular cloud scheduling algorithms. Then, the proposed ComplexCloudSim is presented by including cloud complexity in the original CloudSim.

1) Cloud Scheduling Algorithms: In general, in a cloud scheduler, we integrate a scheduling algorithm that runs on a permanent basis as follows: checking for available resources, receiving new incoming jobs, selecting appropriate resources based on performance (Estimated time to be completed) criteria and feasibility (jobs requirements to resources) as well as generating job plans (to make decision about job priorities and ordering) with selected resources.

Usually, Table I shows a list of terminologies used in relation to scheduling in clouds. For performance evaluation, this paper employs four popularly used heuristic scheduling algorithms related to simulations of cloud-based complexity. The followings are the definitions of these four heuristics.

- **FCFS**: Based on the sequence of submissions, tasks are executed. The task arriving first is prioritised for scheduling based on the available resource, just after submission, following which it is removed from the queue.

TABLE I. TERMINOLOGY FOR SCHEDULING IN CLOUD COMPUTING

Name	Description
QoS	Quality of the service
$MIPS$	Million instructions per second (CPU processing speed)
L_t	Length of task measured in million of instructions
ETC	Estimated time to compute
ERT	Estimated ready time of resource
MCT	Minimum completion time matrix
M_e	Estimated makespan
M_a	Actual makespan

- **Round Robin**: The first task is scheduled on the first resource, and then the second task on the second resource. This goes on through a cycling process for all the available resources.
- **MinMin** : Based on their length (of execution), all tasks in a job are first ordered. Scheduling is first done for the task having the shortest length for which the completion time will be minimum based on the available resource. Then, it is removed from the queue.
- **MaxMin** : Base on their length (of execution time), all tasks in a job are first ordered. Scheduling is done first for the task with the minimum length for which the completion time is maximum based on the available resource. Then, it is removed from the queue.

2) Motivational Example: This section shows how the robustness of a scheduler is affected by the complexity of resources. Let us consider a case in a homogeneous cloud with three VMs where ten independent jobs have to be scheduled (specifications are presented in Tables II and III). To make the complexity of scheduling simple, let us assume the jobs length is fixed and known and also consider that the clouds other performance related features will have no impact on the jobs actual completion timesuch as network bandwidth, memory consumption and disk I/O.

TABLE II. JOBS SPECIFICATIONS

Job Number	Number of Tasks	Task Length (MIs)
1	3	100
2	2	80
3	8	70
4	4	100
5	3	80
6	3	20
7	2	50
8	6	60
9	2	90
10	4	150

TABLE III. VMS SPECIFICATIONS

VMs	Core#	$MIPS_{request}$	$MIPS_{provision}$
VM1 (4 Cores)	1	10	9
	2	10	9
	3	10	9
	4	10	9
VM2 (4 Cores)	5	10	10
	6	10	10
	7	10	10
	8	10	10
VM3 (4 Cores)	9	10	11
	10	10	11
	11	10	11
	12	10	11
Total 3 VMs	12 Cores	120	120

In this example, the Min-Min heuristic is employed to schedule all of these independent jobs. Since this algorithm is efficient and simple, a better schedule (which minimises the jobs' total completion time) is produced when compared with other algorithms in the literature. Also, Algorithm 1 presents the Min-Min algorithm's pseudo code.

Algorithm 1 MinMin Scheduling algorithm

1: **Require:** A set of jobs with n tasks, m different cores, MCT matrix
2: **procedure** MINMIN SCHEDULING ALGORITHM
3: A list of jobs L_j in queue
4: A list of available cores L_c
5: **while** List L_j is no empty **do**
6: For each job in the list L_j
7: **if** The number of avaliable cores meets the job's requirement **then**
8: find the core that gives the minimum ETC
9: Update MCT matrix
10: From the MCT matrix, find the job with the minimum ETC
11: Remove the job from the job list L_j
12: Schedule the job's tasks to the match cores
13: Update the available cores list L_c

As we can see from the difference between the estimated scheduling plan in Figure 2 and the actual scheduling plan in Figure 3, the complexity of resources have a great impact on the job's QoS. In this simple example, the complexity factor of resources is shown to degrade the robustness of scheduling algorithms, i.e. the average job makespan and the

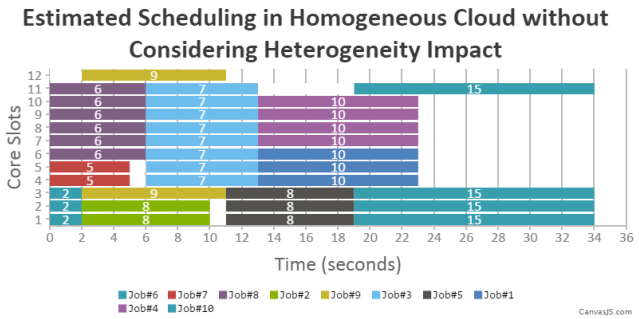

Fig. 2. Motivational Example : Estimated Scheduling Plan

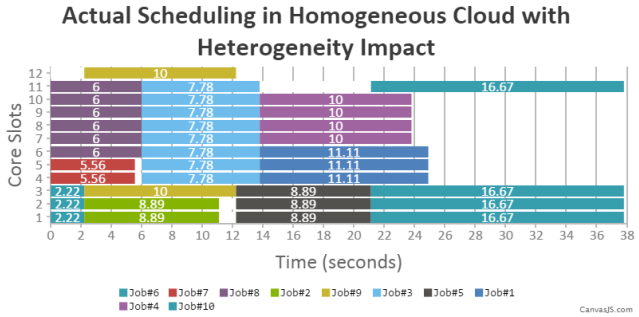

Fig. 3. Motivational Example : Actual Scheduling Plan

total workload runtime in this example, as shown on Table IV. Therefore, in the following section II-B1, we will investigate how different degrees of complexity impact such robustness and how different scheduling heuristics perform under the complex cloud environment.

TABLE IV. JOBS COMPLETION DETAILS

Job Number	M_e	M_a	Makespan Degradation
1	23s	24.89s	1.89s
2	10s	11.11s	1.11s
3	13s	13.78s	0.78s
4	23s	23.78s	0.78s
5	19s	21.11s	2.11s
6	2s	2.22s	0.22s
7	5s	5.56s	0.56s
8	6s	6s	0s
9	11s	12.22s	1.22s
10	34s	37.78s	3.78s (11%)

3) The Implementation for Introducing Complexity: As we have discussed at the beginning of this section, the performance of cloud scheduling is subject to different complexity factors relating to cloud resources: heterogeneity, dynamicity and uncertainty. In the remainder of this section, we will describe, in detail, how ComplexCloudSim attempts to capture these complexity factors.

*a) **Heterogeneity Ratio for VMs Provision**:* In a similar way to the situation with a real-world Cloud Provider, the performance of the provisioning VMs is not guaranteed in ComplexCloudSim. Hence, VMs of equal configuration are likely to have different core performances characterised by the random degradation of request MIPS during provision -

unlike the guaranteed fixed MIPS provision of CloudSim. In ComplexCloudSim, we allocate MIPS to the VMs when they are created, according to the *Heterogeneity Ratio*, as we can see from Algorithm 2.

Algorithm 2 Heterogeneity Ratio for VMs Provision

1: **Require:** VMs MIPS configuration, $MIPS_{request}$
2: **Require:** Heterogeneity Ratio, $0 \leq Ratio_{heterogeneity} \leq 1$
3: **procedure** VMCREATE($MIPS_{request}$,$Ratio_{heterogeneity}$)
4: **if** $Ratio_{heterogeneity} > 0$ **then**
5: $MIPS_{provision} = MIPS_{request} * (1 - Random \in [-Ratio_{heterogeneity}, Ratio_{heterogeneity}])$
6: **else** $MIPS_{provision} = MIPS_{request}$
7: VMProvision($MIPS_{provision}$)

b) ***Dynamicity Ratio for Changes of VM performance at Runtime****:* The idea that there are dynamic changes to performance at runtime, due to the sharing of common resources with other VMs and users, is also an important concept relating to the complexity inherent to Cloud scheduling. In CloudSim, the VM performance is kept to a fixed number of MIPS during simulation. In ComplexCloudSim, we periodically, every second, change the VM's runtime MIPS according to its Dynamicity Ratio and the host's current CPU utilization, as shown in Algorithm 3

Algorithm 3 Dynamicity Ratio for Changes of VM performance at Runtime

1: **Require:** Host's CPU Utilization, U_{host}
2: **Require:** Dynamicity Ratio, $0 \leq Ratio_{dynamicity} \leq 1$
3: **procedure** UPDATEMIPS(U_{host},$Ratio_{dynamicity}$) EVERY SECOND
4: **if** $Ratio_{dynamicity} > 0$ **then**
5: $MIPS_{runtime} = MIPS_{provision} * (1 - U_{host}) * (1 - Random \in [-Ratio_{dynamicity}, Ratio_{dynamicity}])$
6: **else** $MIPS_{runtime} = MIPS_{provision}$

c) ***Uncertainty Ratio for VM Performance Estimation with Inaccurate Information in Scheduling****:* Accurate resource performance prediction is hard or even impossible to achieve in actual complex cloud environments. CloudSim assumes that full information can be obtained and that such information is always correct for the purposes of performance prediction; this is not feasible in real world scenarios. Thus, we introduce a confidence level, the Uncertainty Ratio, to the resource performance predictions, which is used by several scheduling algorithms when making scheduling decisions (e.g. MinMin, MaxMin). In ComplexcloudSim, we inject the Uncertainty Ratio into all the processes which need to perform performance prediction, according to the algorithm 4.

Algorithm 4 Uncertainty Ratio for VM Performance Estimation with Inaccurate Information in Scheduling

1: **Require:** Estimated VM performance, $MIPS_{estimate}$
2: **Require:** Uncertainty Ratio, $0 \leq Ratio_{Uncertainty} \leq 1$
3: **procedure** PREDICTMIPS($MIPS_{estimate}$,$Ratio_{Uncertainty}$)
4: **if** $Ratio_{Uncertainty} > 0$ **then**
5: $MIPS_{actual} = MIPS_{estimate} * (1 - Random \in [-Ratio_{Uncertainty}, Ratio_{Uncertainty}])$
6: **else** $MIPS_{actual} = MIPS_{estimate}$

III. Complexity Simulation: Comparison of Four Heuristics Cloud Scheduling Algorithms

To showcase a possible application of Complexcloudsim, we simulated the execution of a computationally intensive workload (The Montage workflow) using four different heuristic cloud scheduling algorithms and various degrees of complexity in the Cloud resources. We expected the schedulers to differ in their robustness in relation to complexity, and that this should be reflected in diverging workflow execution times. In this section, we outline the experimental setup and evaluate the impacts of resource complexity on Cloud scheduling systems.

A. Experiment Setup

Simulation of the scheduling system was done to examine the robustness of the degradation created by resource complexity. A Montage workflow was employed for this experiment, which comes with CloudSim. This included 1,000 jobs containing a group of random number sub-tasks. To maintain simplicity, we employ a global variable, a degree of complexity, which allows configuring the ratios of dynamicity, heterogeneity and uncertainty simultaneously. For each configuration, the execution of Montage workflow was repeated 100 times on five VMs, after which the statistical results were generated in terms of workflow runtimes. During the course of the experiments, the degree of complexity caused by ComplexCloudSim was incrementally increased, and the impacts of complexity on cloud scheduling systems QoS performance was measured. To compare ComplexCloudSim with the original CloudSim, a baseline simulation we conducted that ran without considering complexity factors; this was also executed 100 times. As expected, under four scheduling algorithms, we determined the workflow runtime for the same workflow with zero variance maintained in the original CloudSim, as presented in Table V.

TABLE V. BASELINE SIMULATION RESULT WITH ORIGINAL CLOUDSIM

Scheduling Algorithms	FCFS	RR	MinMin	MaxMin
Average Runtime (Minutes)	2862	2865	2864	2862
Variance	0	0	0	0
Standard Deviation	0	0	0	0

B. Experiment Result

Here, the impacts on robustness were compared by employing different degrees of resource complexity and scheduling algorithms. Figures 4 and 5 outlined above present the experiment's results. Figure 4 presents the average runtime of the Montage work-flow between 3,220 and 3,505 minutes for all experiments. This indicates degradation of runtime by around 1323% compared with the performance baseline. Apparently, ComplexCloudSim offers complexity factors that have a considerable impact on the cloud scheduling system's QoS.

The average runtime degradation was also found not to change directly in tandem with increased degree of complexity. However, as observed in Figure 5, the degree of complexity ranging from 20% to 120% was found to be proportional with

the increase in the standard deviation for workflow runtime. It was clear that less reliable scheduling performances were obtained due to increase in the standard deviation. Thus, the complexity of the resources determines the reliability of the cloud scheduling system.

Based on the experimental results, the complexity factor had minimum impact on the MinMin scheduling algorithm in terms of both average and standard deviation of the workflow runtime. This suggests that more robust schedules are generated due to MinMin in a complex cloud environment. So when compared with other three heuristics, the overall performance of MinMin was found to be better, which was in line with the earlier research.

Evidently, the effect of complex resources can be simulated by ComplexCloudSim. This is a very desirable property as cloud environments always keep facing complexity issues. We expect this to be important going forward as other cloud simulators did not sufficiently support it.

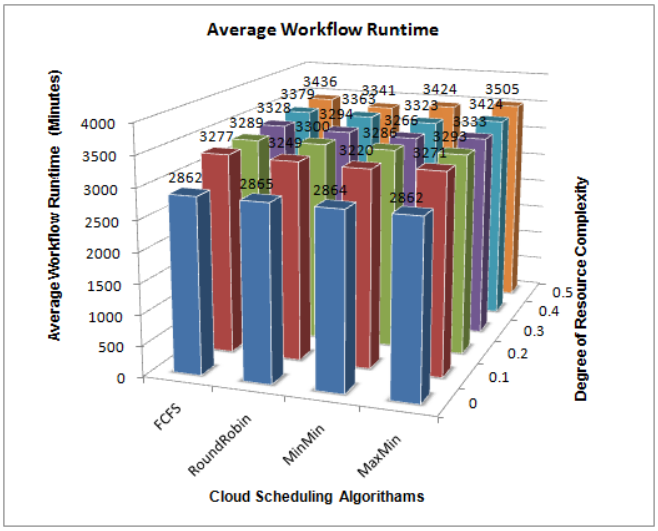

Fig. 4. Complexity Simulation: Average Workflow Runtime

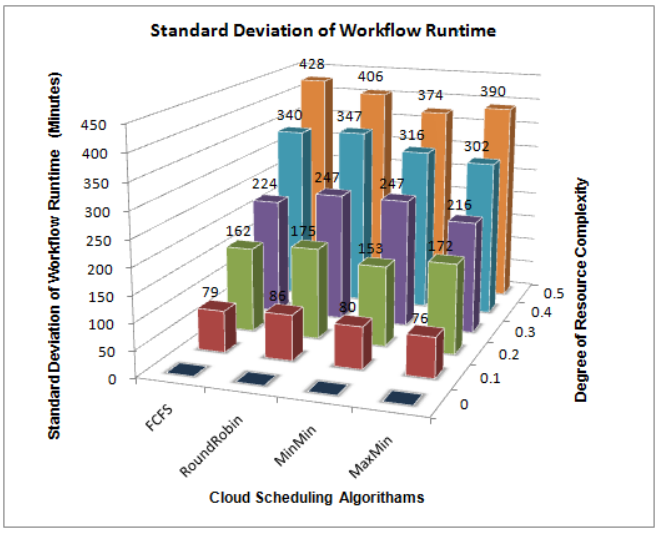

Fig. 5. Complexity Simulation: Standard Deviation of Workflow Runtime

IV. Damage Spreading Evaluation: Chaotic Behaviour in Cloud Scheduling

The original development of the tool Damage Spreading [17] was aimed at studying biologically motivated complex systems. This tool has been commonly referred in the literature for several research areas, including complex network models, for observing systems complex behaviour. In complex systems, the evolution of slightly different configurations of variables can be investigated with this tool, provided they are subjected to the same number sequence. Obtaining information regarding whether or not a small perturbation (damage to the conditions) introduced amongst variables can stay or spread at the same level (even disappears) would assist us in examining a systems robustness in relation to disturbance [16].

Here, "initial damage" is the occurrence of a slight change in the number of VMs C_{vm} and the degree of resource complexity $C_{complexity}$ to run the same workload. We introduced small changes $C_{vm} = 1$ and $C_{complexity} = 0.1$ to a simulation step-wisethe simulation that was executed 100 times with the same workload. Then, we examined if the changes would spread or not by taking into account two important QoS determinants in the scheduling processes - the standard and average deviation of workflow runtime.

To assess the damage spread, the damage was defined as D_{std} (difference in workflow runtime standard deviation R_{std}) and $D_{average}$ (difference in average workflow runtime $R_{average}$) present between two simulation results. As shown through Formulas 1 and 2, these were then calculated, where $j \in [0.1, 0.2, 0.3, 0.4, 0.5]$ represents the degree of complexity and $i \in [5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]$ represents the number of VMs.

$$D_{average}(i,j) = R_{average}(i + C_{vm}, j) - R_{average}(i,j) \quad (1)$$

$$D_{std}(i,j) = R_{std}(i, j + C_{complexity}) - R_{std}(i,j) \quad (2)$$

Figures 6 and 7 show the results of $D_{average}$ and D_{std} respectively.

As observed in Figure 6, for number of VMs $i < 10$ and various degrees of complexity, the changes of $D_{average}$ are relatively small. The damage does not spread in this region and stays low at initial level.

As seen in Figure 7, the changes of D_{std} for various degrees of complexity, for number of VMs $i < 9$, become highly unstable. However, the situation becomes considerably better with an increase in the number of VMs, when $i > 9$.

Then, the relation between the spreading damage and the number of increased VMs i is examined by employing the standard deviation of $D_{average}$ and D_{std}. The standard deviation of $D_{average}(i)$ as $\sigma_{average}(i)$, and the standard deviation of $D_{std}(i)$ as $\sigma_{std}(i)$ are defined. Hence, the mean values: $Mean(\sigma_{std})$) and $Mean(\sigma_{average})$ of all $\sigma_{average}$ and σ_{std} are calculated, as presented in Tables VI and VII.

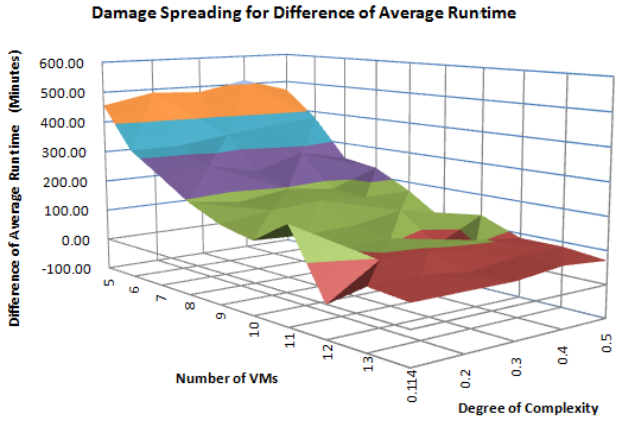

Fig. 6. Damage Spreading Evaluation: $D_{average}$

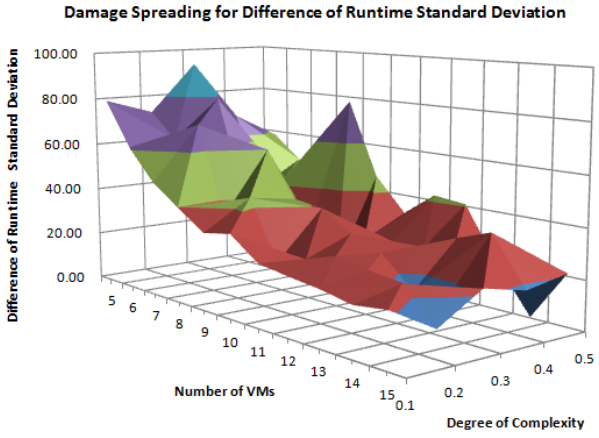

Fig. 7. Damage Spreading Evaluation: : D_{std}

Now, the state of the region is categorised loosely by employing such mean values. We now assign region $\sigma_{average}(i) \leq Mean(\sigma_{average})$ or $\sigma_{std} \leq Mean(\sigma_{std})$ as "Stable Regions". In this region, spreading and initial damages maintain a stable correlation. Reliable improvements in QoS occur with increased number of VMs, which signify smooth and robust running of the scheduling system against degree of complexity changes. We also categorise $\sigma_{average}(i) > Mean(\sigma_{average})$ or $\sigma_{std} > Mean(\sigma_{std})$ as "Chaotic Regions" [18], as highlighted in Tables VI and VII by the red colour. In this region, throughout the scheduling system, small disturbances may spread, which result in significant changes in performance due to the degree of complexity experienced. This suggests that it is difficult to guarantee QoS to an increase in the number of VMs.

Knowing when the scheduling system is in a chaotic region or stable region helps in providing important guidelines to quickly make decisions regarding achieving of a more robust scheduling. For e.g. in a real world situation, we might run a similar workload with more than 9 VMs based on the results from simulation of ComplexCloudSim, but we could also avoid choosing 11 or 12 VMs in a bid to satisfy the requirement of QoS.

TABLE VI. RELATION BETWEEN NUMBER OF VMS AND $D_{average}$

	$D_{average}(i)$					
	Degree of Complexity					Mean($\sigma_{average}$)=23
(i) VMs	0.1	0.2	0.3	0.4	0.5	$\sigma_{average}(i)$
5	456	489	481	514	469	22
6	320	322	344	363	377	25
7	258	271	237	282	248	18
8	193	174	196	178	231	23
9	148	168	180	169	171	12
10	124	117	122	149	94	19
11	198	101	108	64	135	50
12	-1	96	98	104	86	44
13	80	81	65	83	86	8
14	69	68	67	83	71	7

TABLE VII. RELATION BETWEEN NUMBER OF VMS AND D_{std}

	$D_{std}(i)$					
	Degree of Complexity					Mean(σ_{std})=24
(i) VMs	0.1	0.2	0.3	0.4	0.5	$\sigma_{std}(i)$
5	58	69	94	73	80	49
6	48	37	79	63	61	38
7	42	43	39	71	48	31
8	78	23	60	34	40	30
9	46	9	41	44	32	21
10	32	23	39	20	34	18
11	42	25	31	24	26	18
12	41	26	26	28	24	17
13	19	32	15	26	22	13
14	0	37	15	24	20	11
14	21	18	22	11	22	11

V. CONCLUSION AND FUTURE WORK

This paper presents an extension to the CloudSim, which is ComplexCloudSim, to analyse scheduling under a complex cloud environment. The design of a resource complexity module (dynamicity, heterogeneity, and uncertainty) is based on implementation with the primary goal to offer a useful tool for testing and validating the cloud scheduling algorithms robustness. Section III presents the examination results of four cloud scheduling algorithms to showcase the capability of ComplexCloudSim to simulate different complexity factors for the cloud scheduling system as well as replicate the shortcoming and known strengths of these algorithms. Then, based on simulation in Section IV, we found two regions: Stable Region, the region with converged small damage and Chaotic Region, the region where damage spread, in the complex cloud scheduling system.

We find Chaotic Behaviour in the cloud scheduling system to be interesting because it signifies that the future schedules in principle cannot be predicted. Such findings may explain why in the real-world production environment, it is difficult to put most of the scheduling algorithms in research, which rely on prediction and the complexity exists everywhere. Even if we know the precise processing time in advance, it does not guarantee the precise completion time of tasks for complex product systems such as the cloud. Therefore, if the scheduling system decides to plan for a more robust production schedule, it has to first predict if it is in Chaotic Region or Stable Region. Then, suppose the system is under the Chaotic Region, it has

to look out for VMs with a suitable number to meet the QoS requirement of the application.

Even through the ComplexCloudSim can model complexity factors to an extent, still it cannot cover all the situations occurring in the real-world cloud. However, the findings related to chaotic behaviour in cloud scheduling system have inspired new ideas to develop a more robust QoS-aware scheduling algorithm. More detailed analysis is required in further work to understand the cloud scheduling systems chaotic behaviour as well as the damage spreading mechanisms. Such chaotic behaviour needs to be studied for applying in real-world applications. We deem our research work to be one of the many steps towards multiple fruitful research topics.

REFERENCES

[1] Mell, Peter, and Tim Grance. "The NIST definition of cloud computing." (2011): 20-23.

[2] Plestys, Rimantas, et al. "The measurement of grid QoS parameters." Information Technology Interfaces, 2007. ITI 2007. 29th International Conference on. IEEE, 2007.

[3] Braun, Tracy D., et al. "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems." Journal of Parallel and Distributed computing 61.6 (2001): 810-837.

[4] Gutierrez-Garcia, J. Octavio, and Kwang Mong Sim. "A family of heuristics for agent-based elastic cloud bag-of-tasks concurrent scheduling." Future Generation Computer Systems 29.7 (2013): 1682-1699.

[5] Bala, Anju, and Inderveer Chana. "A survey of various workflow scheduling algorithms in cloud environment." 2nd National Conference on Information and Communication Technology (NCICT). 2011.

[6] Iosup, Alexandru, Nezih Yigitbasi, and Dick Epema. "On the performance variability of production cloud services." Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on. IEEE, 2011.

[7] Schad, Jrg, Jens Dittrich, and Jorge-Arnulfo Quian-Ruiz. "Runtime measurements in the cloud: observing, analyzing, and reducing variance." Proceedings of the VLDB Endowment 3.1-2 (2010): 460-471.

[8] Herroelen, Willy, and Roel Leus. "Project scheduling under uncertainty: Survey and research potentials." European journal of operational research 165.2 (2005): 289-306.

[9] Calheiros, Rodrigo N., et al. "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms." Software: Practice and Experience 41.1 (2011): 23-50.

[10] Chen, Huankai, et al. "Complexity Reduction: Local Activity Ranking By Resource Entropy For QoS-aware Cloud Scheduling." Services Computing (SCC), 2016 IEEE International Conference on. IEEE, 2016.

[11] Chen, Huankai, and Frank Z. Wang. "Spark on entropy: A reliable & efficient scheduler for low-latency parallel jobs in heterogeneous cloud." Local Computer Networks Conference Workshops (LCN Workshops), 2015 IEEE 40th. IEEE, 2015.

[12] Chen, Weiwei, and Ewa Deelman. "Workflowsim: A toolkit for simulating scientific workflows in distributed environments." E-Science (e-Science), 2012 IEEE 8th International Conference on. IEEE, 2012.

[13] Garg, Saurabh Kumar, and Rajkumar Buyya. "Networkcloudsim: Modelling parallel applications in cloud simulations." Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on. IEEE, 2011.

[14] Bux, Marc, and Ulf Leser. "Dynamiccloudsim: Simulating heterogeneity in computational clouds." Future Generation Computer Systems 46 (2015): 85-99.

[15] Chen, Huankai, Frank Wang, and Na Helian. "A Cost-Efficient and Reliable Resource Allocation Model Based on Cellular Automaton Entropy for Cloud Project Scheduling." system 4.4 (2013).

[16] Ikeda, Hinata. "Chaotic behavior in complex shop scheduling." Soft Computing and Intelligent Systems (SCIS) and 13th International Symposium on Advanced Intelligent Systems (ISIS), 2012 Joint 6th International Conference on. IEEE, 2012.

[17] Kauffman, Stuart A. "Metabolic stability and epigenesis in randomly constructed genetic nets." Journal of theoretical biology 22.3 (1969): 437-467.

[18] Boccaletti, Stefano, et al. "The control of chaos: theory and applications." Physics reports 329.3 (2000): 103-197.