IEEE SERVICES 2015 Visionary Track: The Future of Software Engineering FOR and IN Cloud, New York, USA, July 31 2015 © IEEE.

Automating Performance and Energy Consumption Analysis for Cloud Applications

Feifei Chen, John Grundy, Jean-Guy Schneider, Yun Yang and Qiang He
School of Software & Electrical Engineering
Swinburne University of Technology
Melbourne, Australia 3122
{feifeichen, jgrundy, jschneider, yyang, qhe}@swin.edu.au

***Abstract*— In cloud environments, IT solutions are delivered to users via shared infrastructure, enabling cloud service providers to deploy applications as services according to user QoS (Quality of Service) requirements. One consequence of this cloud model is the huge amount of energy consumption and significant carbon footprints caused by large cloud infrastructures. A key and common objective of cloud service providers is thus to develop cloud application deployment and management solutions with minimum energy consumption while guaranteeing performance and other QoS specified in Service Level Agreements (SLAs). However, finding the best deployment configuration that maximises energy efficiency while guaranteeing system performance is an extremely challenging task, which requires the evaluation of system performance and energy consumption under various workloads and deployment configurations. In order to simplify this process we have developed StressCloud, an automatic performance and energy consumption analysis tool for cloud applications in real-world cloud environments. StressCloud supports the modelling of realistic cloud application workloads, the automatic generation of load tests, and the profiling of system performance and energy consumption. We demonstrate the utility of StressCloud by analysing the performance and energy consumption of a cloud application under a broad range of different deployment configurations.**

***Keywords*—Cloud Computing, Green Computing, Performance Analysis, Form Chart, Load Test**

I. INTRODUCTION

Cloud computing is a relatively new paradigm where users lease cloud infrastructure and services rather than buy them outright [1]. This allows users to: 1) achieve elastic computation and storage, i.e. the ability to dynamically scale those resources up and down according to real-time needs; 2) pay for only what they currently need; 3) avoid high up-front purchase and on-going infrastructure and service maintenance; 4) avoid in-house need for skills sets for specific IT platforms; and 5) for small and medium enterprises (SMEs) in particular - leverage IT security, scalability, reliability and robustness that are difficult to obtain with purely in-house solutions.

However, two major challenges of the cloud model are the huge energy consumption of large-scale cloud data centres and the need to meet ever-increasing system performance and other Quality of Service (QoS) requirements for the cloud applications. High energy consumption directly contributes to data centres' operational costs, especially as the energy unit cost continues to rise significantly. Currently, power consumption contributes up to 42% of a data centre's total expense [2]. In addition, the huge amount of power consumption of data centres potentially accelerates global warming [3]. On the other hand, cloud service providers must provide their users with satisfactory system performance, usually measured in throughput and response time. For instance, even a 100ms extra delay of cloud service response time can cause a 1% drop in sales [4]. Thus, cloud service providers must develop cloud application deployment and management solutions with minimum energy consumption while guaranteeing system performance and Service Level Agreements (SLAs).

However, finding the best deployment configuration that maximises energy efficiency while guaranteeing system performance is an extremely challenging task, due to the complexity and heterogeneity of cloud applications and deployment platforms. It thus requires the evaluation of system performance and energy consumption under various combinations of application workloads and platform configurations. There are however numerous different application workloads and platform configurations, even for a small cloud application like JPetStore[1]. Manual generation of load test plans, changes of system configurations and running of load tests are tedious and error-prone. In addition, the collection of accurate system performance and energy consumption data of cloud applications relies on the use of load tests based on a realistic user behaviour model and running the load tests in a real-world cloud environment.

A lot of research effort has been devoted to building performance evaluation tools for cloud systems [5-10]. However, most existing approaches provide only a fairly basic model for user behaviour: a sequence of user requests on cloud servers arranged into repeating groups with multiple threads (to mimic large number of cloud users) [5, 6, 10]. Thus, a rich and realistic cloud application workload model is required before reliable performance and energy consumption evaluation can be conducted. Moreover, most of the existing performance evaluation tools utilise simulated cloud environments, providing approximations of cloud system performance [5, 7-9]. The key limitation of simulation-based modelling is that test results may be inaccurate because of the imperfection in the assumptions, input data, work tasks, energy usage and performance in the simulation environment [11].

[1] http://java.sun.com/developer/releases/petstore/

In order to address the above issues, we have developed StressCloud, a novel performance and energy consumption analysis tool for cloud applications in real-world cloud environments. StressCloud offers the following features: 1) the ability to model realistic cloud application workloads at varying levels of detail; 2) the ability to model cloud application deployment configurations at varying levels of detail; 3) automatic generation of detailed load test plans; 4) support for automatic load tests; and 5) automatic monitoring, profiling and analysis of system performance and energy consumption.

Key novel contributions of our StressCloud approach are:

- Supporting user-defined high-level architecture and workload models for complex cloud applications;
- Fully automatic generation and deployment of large-scale cloud application workload test services and cloud application model prototype implementations;
- Ability to realistically energy and performance stress test existing cloud applications and potential cloud application models;
- Automatic profiling of system performance and energy consumption of the cloud application;
- Analytical support for pre-test model energy and performance weaknesses and post-test energy and performance metric analysis.

The reminder of this paper is organised as follows. Section II briefly summarises the state-of-the-art in cloud performance and energy consumption profiling and analysis approaches, and reviews existing load test tools for cloud systems. Section III gives an overview of StressCloud, our tool for profiling and analysing performance and energy consumption of cloud applications. Section IV describes the system architecture of StressCloud. Section V introduces the StressCloud testing procedure. In Section VI, we demonstrate the effectiveness of StressCloud by presenting the test results of the system performance and energy consumption of JPetStore under different deployment configurations. The key advantages and limitations of StressCloud are discussed in SectionVII. We conclude by summarising the main contributions of this research and key areas of our future work in Section VIII.

II. Related Work

Some research has attempted to leverage the trade-off between the performance and energy consumption of cloud applications. Metri et al. [12] investigated the energy efficiency of data centres by running benchmark applications on cloud servers. However, they focused on a black box in benchmarking the performance and energy consumption of cloud applications without looking into the parameters of the cloud applications. Lee and Zomaya analysed energy-efficient resource utilisation of cloud applications [3]. They concluded that energy consumption can be reduced when two or more tasks are consolidated rather than solely deployed on one resource. However, they did not consider the performance aspect of such applications. Chen et al. [13] profiled and analysed the performance and energy consumption of cloud applications based on individual task types. They aimed to investigate the impact of cloud application workloads and resources allocation strategies on energy consumption and system performance. However, they did not take into account the user behaviours in the workload model.

A wide range of load test tools have been developed for cloud systems. CloudSim [5] is a self-contained cloud platform that allows modelling and simulation of cloud infrastructures containing data centres, users and user workloads. It can be used to simulate a transactional, continuous workload such as a web server. However, it lacks a detailed model of the application and thus its analysis results are limited. iCanCloud [14] is a simulation framework for large storage networks. It provides the ability to evaluate the performance of a particular application on a specific hardware. However, no energy consumption model has been considered.

Some cloud load test tools have taken into account the energy consumption of data centres. GreenCloud [8] is focused on simulating the communications between processes running in a cloud at packet level. It is specially designed to simulate energy consumption of the components of data centre, including servers, switches and links. MDCSim [15] is an event-driven simulation platform which focuses on data centre architecture and cluster configuration, measuring both performance and power metrics. However, both GreenCloud and MDCSim utilise simulation-based modelling. The key limitation of simulation-based modelling is that test results may be inaccurate because of the imperfection of environmental configuration and input data in the simulation [11]. In contrast to simulation-based modelling, model-based test-bed generation [16] provides more accurate test results because a test-bed is a more realistic representation of the real software environments. Compared to simulation, less work has been done in generating a model-based performance and energy test-bed for cloud applications.

A *form chart model* is a technology-independent bipartite state diagram used to simulate users' request/response behaviour [17]. The model describes at a high level what the user sees as system output, and what he or she provides as input to the system. It captures the structure of the target system from the user's perspective and can be augmented with probabilities to capture user interactions with the target system. A stochastic form chart model has been extended from the basic form chart model with stochastic functions that generate performance testing workloads of Web applications [17]. In the stochastic form chart model, the pages of a web site are represented as bubbles, the actions as boxes and the transitions between them as arrows. Cloud users send service requests to the cloud application and get responses from the cloud application with service results. A cloud system can be considered as a request/response system. Therefore, we adopt the stochastic form chart models and extend it to model the cloud application workloads of user-to-service and service-to-service request/response path likelihood.

III. StressCloud Overview

Fig.1 shows how StressCloud is used to perform load test to profile the performance and energy consumption of a cloud

application. First, the performance engineer defines the cloud application workload model (1). The workload model is composed of a set of tasks modelling the target cloud application behaviour. Based on the major type of resource consumed by a task, we categorise runtime tasks into three types: computation-intensive, data-intensive and communication-intensive [18]. In real-world applications, cloud application services are made up of composite tasks that consume multiple types of cloud resources, including CPU, RAM, data storage and network devices. Thus, we introduce a "composite task" in our workload model to represent such composite tasks. This model is then further augmented by the performance engineer with transition probabilities and properties between tasks, forming a detailed, executable workload model.

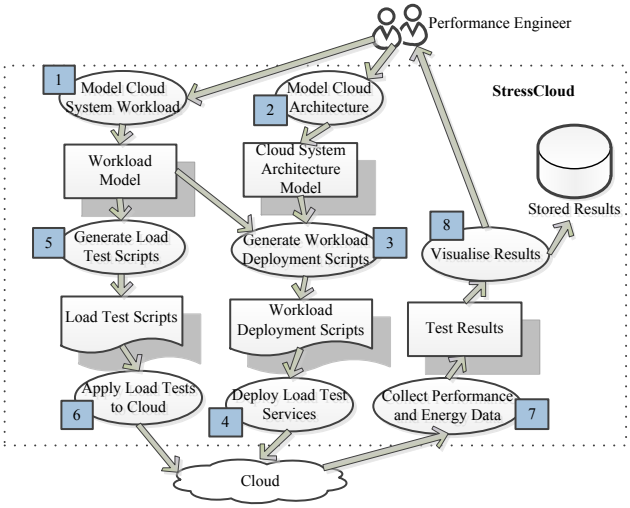

Fig.1. StressCloud Performance and Energy Data Profiling Process.

We have developed a collection of cloud services configured by these application workload models in order to provide a realistic target application for energy and performance data collection. These services respond to user requests by performing tasks defined in the workload model. This allows for what-if energy consumption and system performance analysis of planned systems and for modelling the re-engineering of existing systems. For each task, a stochastic form chart is created to specify detailed user requests and required responses from the cloud system. These workload models are deployed and run on real-world cloud hardware platforms to produce a realistic performance and energy consumption behaviours for measurement and comparison.

A cloud system architecture model is defined by the performance engineer to specify the elements in the target cloud system (2). Our cloud architecture model includes all available resources in the target cloud system and their detailed configurations. Resources of different types in the cloud can be specified by different resource locations, such as physical servers and virtual machines (VMs). After mapping the tasks defined in the user workload model to corresponding resources in the cloud system architecture model, workload deployment scripts are generated (3). Based on the deployment scripts, load test services are uploaded and deployed to the VMs in the target cloud system (4). These load test services were developed based on our previous research that incorporated CPU, RAM and data-intensive tasks, and supported service-to-service communication-intensive tasks [13]. Load test scripts are then automatically generated based on the workload model (5). Next, the specified load tests are performed automatically in the target deployed cloud system based on the load test scripts (6). The performance and energy consumption data of the target cloud system are collected (7) and then visualised using a variety of charts (8). The visualised system performance and energy consumption data are updated at a user-specified rate, by default every 20 seconds. Test results are stored for future comparison with new tests running with differing tasks, loads and deployment models.

IV. System Architecture

The high-level architecture of StressCloud is presented in Fig.2. StressCloud is realised as a set of Eclipse IDE plug-ins. A set of editors are used to support diagrammatic modelling of workload and deployment platform (1). Three key diagram types are used, i.e., the high-level workload model of the cloud application, low-level workload model of each task and the cloud platform architecture model (2). Diagrammatic editors are instantiated for editing these models using the Eclipse Graphical Modelling Framework. The "DiagramRe-arrangeManager" component enables automatic layout in all graphical components of the diagrams within the modelling environment, to arrange the graphic model entities into different layouts based on user selection. The workload models are used to synthesize detailed load test scripts by the "LoadTestScriptsTranslator" (3). The deployment model is used to synthesize configuration scripts for deploying, configuring and instantiating VMs, hosted applications and services by the "DeploymentScriptsTranslator" (4). A set of objects are created to traverse the cloud architecture and workload models to generate these scripts.

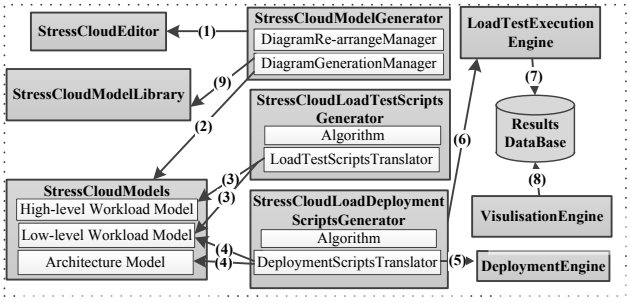

Fig.2. High-level Architecture of StressCloud.

The deployment scripts are sent to a "DeploymentEngine" (5). This component creates and instantiates VMs, runs installation scripts on the VMs, distributes workload configurations to VMs, and configures our workload services hosted on the VMs. It can also retrieve the details of both physical servers and VMs configured on these servers. The engine is implemented as an agent that interacts with the target cloud system manager. StressCloud now supports VMware as the target cloud platform hypervisor. Creating templates of deployment scripts to support more platforms is part of our future work. Similarly, a "LoadTestExecutionEngine" invokes

the load tests on the deployed cloud application (6). This component also manages database connections, captures real time performance and energy consumption data from the target cloud system, transforms the collected data into suitable format, and stores those collected data into files for visualisation (7). The "VisulisationEngine", implemented using Eclipse SWT (8), queries the stored performance and energy data to provide periodic update. All models are saved in the model library as XML files. The "DiagramGenerationManager" component can automatically retrieve the existing models and generate model entities, associations and their visual icons (9).

V. Usage Example

We introduce the key functions of StressCloud using the reference Java application JPetStore as an example in this section.

A. *Workload Modelling*

A performance engineer first needs to define a high-level workload model of cloud application using an extended stochastic form chart. An example workload model of JPetStore is shown in Fig.3. In this example, the client (represented by bottom left icon) is modelled with a set of requests (represented by large icons with sub-request containers) linked together with transitions (represented by grey arrowed lines) with probability annotations. Start (green) and quit (red) circle icons define a state-chart-style model for the workload. This example specifies that the user selects a task (*Signin*, *GetProduct*, *GetIndex*, *GetHelp*, *GetCategory*, *GetCart*, *CreateNewAccount*, *GetProductDetail*, or *CheckOut*) with different probabilities after starting the workload. Multiple workload profiles can be defined for each cloud application.

The performance engineer then defines the low-level workload model of each task in the high-level workload model. Each task is a call to a service in the cloud application. Each such service is modelled as either a call to an existing deployed cloud application service or one or more "basic" service task types (data-intensive, computation-intensive or communication-intensive). These task types are used to build a model of a target cloud application's services. A complex cloud application is thus built up of services comprised of a mix of different types of tasks and different workloads using these tasks.

Each workload task may comprise of sub-tasks that allow us to define in detail what the task does. We again use the probability-based stochastic form chart formalism to model these sub-tasks. Fig.4 shows an example of three JPetStore sub-task models, (a) a communication intensive sub-task (modelling client-to-server or service-to-service communications); (b) a computation-intensive sub-task (modelling information processing on a server node); and (c) a

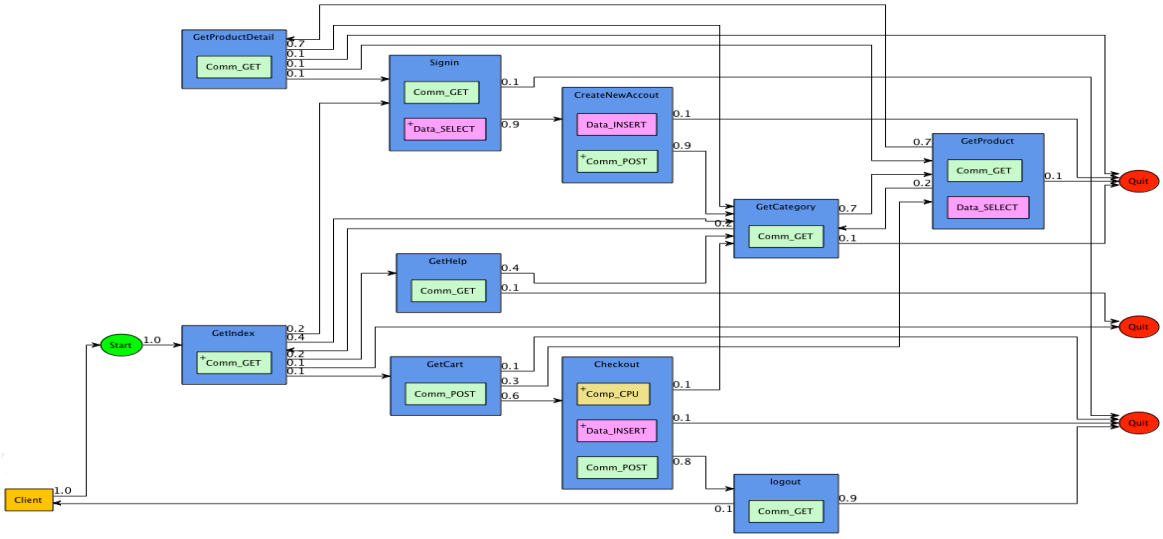

Fig.3. A JPetStore High-level Workload Model in StressCloud.

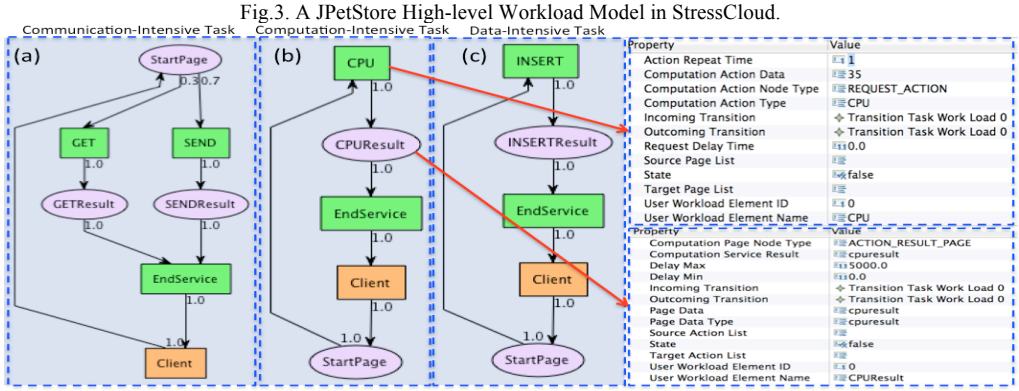

Fig.4. Stochastic Form Chart Models of Communication-intensive Task (a), Computation-intensive Task (b) and Data-intensive Task (c).

data-intensive sub-task (modelling database or file processing on a data storage server node). StressCloud allows the performance engineer to specify a range of information about each sub-task, as shown in the right parts of Fig.4.

B. *Cloud Architecture Modelling*

After defining the detailed workload of a target cloud application, the performance engineer must define the deployment platform for running the application. An example of such a cloud architecture model is shown in Fig.5. A cloud platform comprises of physical server hosts, VMs and networks. Some VMs are optimised for data- or computation-intensive tasks. VMs have configuration parameters, for example, virtual memory size and number of compute cores, a host operating system, and deployed application software services such as web server, database server etc. Physical hosts and networks have various characteristics, for example, server type, number of CPU cores, amount of physical memory, VM hypervisor type, bandwidth, etc. All the details and the platform structure are captured in the cloud architecture model in StressCloud. Different cloud architecture models can be defined to model different physical servers, VM configurations, arrangements of networks and servers, different application software deployments and configurations, etc.

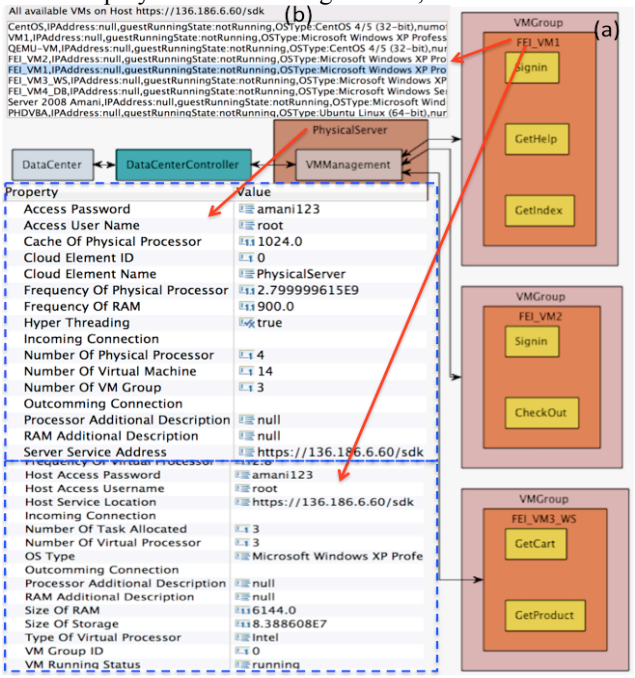

Fig.5. An example Cloud Architecture modelled in StressCloud.

Fig.5 (a) shows a high-level model of the cloud platform architecture for running the JPetStore cloud application in our example loading and energy analysis tests. This is a data centre with one physical host and three VM groups on which different tasks will be hosted. Various configuration parameters are shown at the bottom of the figure.

These cloud platform specifications are used to model and configure actual cloud hardware. Fig.5 (b) shows how the performance engineer specifies a particular VM configuration that has been created for the physical host machine and its hypervisor, in this case a VMWare hypervisor. The performance engineer may configure available parameters for the selected VM type, shown at bottom.

C. *Deployment Plan Generation*

The model of the cloud platform is then used to generate automated configuration scripts in order to configure the platform for load test runs. The performance engineer can assign various JPetStore services to different VMs on the available deployment platform. Some machines may run on the same host, some on different hosts. Those on different hosts will communicate via the physical network, while those on the same host will communicate via the virtual vSwitch provided by the hypervisor. The generated scripts are uploaded to the target servers and VMs by StressCloud and executed to configure them. Within the StressCloud environment, the performance engineer may save their deployment model and define a new one, assign services to different servers/VMs in the new deployment model, and generate and run new deployment scripts.

D. *Load Test Plan Generation*

After generating and deploying the cloud platform and the service configuration scripts, the performance engineer uses StressCloud to generate the load test plans of the modelled JPetStore application. The workload model presented in Fig.3 is used to synthesize a test script that will be run on a client machine. This script models the state machine that describes, the sequence of tasks, transitions between tasks, probabilities that each task will be carried out, iterations of each task, and the workload. Each task in the workload model is modelled either by a call to a deployed real-world cloud application service, or a model of that service. The service model is a set of data-, computation- and communication-intensive sub-tasks. These tasks are themselves organised into a probabilistic model, designed to capture the data, computation and communication intensive aspects in the cloud application service respectively. A service may itself be modelled as a workload model, making use of other services, thereby modelling the service-oriented architecture of many cloud applications.

Each task's sub-model component includes the parameters of the services involved in the task service model. These parameters specify, for example, the number of iterations for storage, compute, send and receive activities in the services, the amount of data to store, process, send and receive in each activity, the type of activity, such as insert, update, delete, select on database, the number of cores to use in processing, the amount of memory to use in processing, the number and size of rows to return for the select activity, and so on.

E. *Test Running and Results Visualisation*

Once load test scripts have been generated they are uploaded to one or more machines acting as "clients" and run. The performance engineer may ask for many hundreds or even thousands of instances of the "client" to be run simultaneously, permitting large-scale stress tests to be performed. Clients running the load tests can be hosted on the same or different physical machines, depending on machine availability, to

enable analysis of network performance and energy consumption under different loads.

The results of load tests are periodically collected and interactive visualisations are provided to the performance engineer Different collection intervals and parameters can be set. Results can be saved for post-hoc analysis and comparative analysis of different workload and deployment configurations.

VI. Evaluation

To evaluate StressCloud's utiltiy, we used JPetStore as a representative cloud application and discuss results of a set of energy usage evaluations and a user evaluation of StressCloud.

A. Experimental Setup

The experiments were conducted on SwinCloud. In order to eliminate the impact of energy consumption and system performance introduced by energy collection applications, a PowerNode[2] power monitor developed by GreenWave Reality is connected directly to SwinCloud servers. It supports measurement of both immediate and average power consumption. The data collected by the PowerNode were reported to a GreenWave Gateway. StressCloud retrieves power consumption readings from the Gateway once every second. The energy consumption of a cloud system includes the energy consumed by the constituent servers and the scheduling overhead across the servers. We focus on the energy consumption of individual servers in our experiments as it is the predominant component in cloud.

The server deployed was a HP Z400, the VM Manager (VMM) used is a VMware ESX 4.1, and the operating systems running on the VMs are Windows XP Professional SP3. The server has 4 physical cores and 8GB of RAM. In our experiments, VMs were assigned with 2GB, 4GB, 6GB or 8GB of RAM. The number of virtual CPUs (vCPUs) of each VM varied from 1 to 4 in steps of 1. The number of vCPUs configured on a VM was equivalent to the number of physical cores assigned to the VM.

StressCloud was installed on a client PC. All workloads were modelled and generated using StressCloud and then sent to SwinCloud servers. A series of web services for load tests were deployed on VMs hosted on the cloud server. It was configured by the StressCloud scripts generated from the cloud application workload models. The system performance and energy consumption data were collected and stored for analysis and visualisation. For the modelled JPetStore, as presented in Section V and named "MJPetStore", we performed a number of load and energy tests with different workload models and platform deployment configurations. We conducted two major sets of experiments. In the first set, we deployed MJPetStore on a single VM and changed the workload. In the second set, we kept the workload of the MJPetStore constant and changed the deployment configurations.

B. Experiment Results

We first deployed MJPetStore on one VM with 3 CPUs and 6GB RAM. The initial number of users was set to 10. We then increased the concurrent requests number of each user from 50 to 200 in steps of 50. The energy consumption and system throughput are presented in Fig.6. As shown in Fig.6 (a), the energy consumption increased with the number of concurrent requests. The throughput, on the other hand, decreased accordingly as shown in Fig.6 (b). Intuitively, more user requests will introduce more scheduling and synchronising overhead in the cloud application, which will result in an increase in VM CPU usage and memory usage. Therefore, the power consumption of the server will increase accordingly. In addition, scheduling and synchronising overhead will also introduce an increase in the processing time for each user request. Thus, the energy consumption of the task increases and throughput decreases.

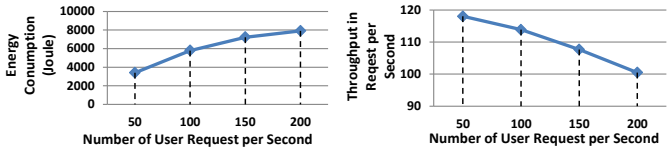

Fig.6. Energy Consumption (a), Throughput (b).

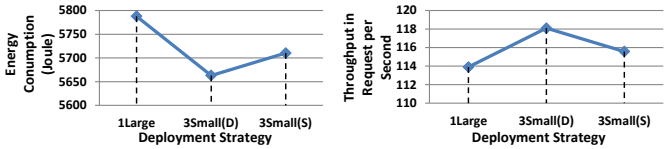

Fig.7. Energy Consumption (a), Throughput (b).

We then kept the workload of the MJPetStore constant and changed the deployment configurations. The initial number of users was set to 10 and the number of concurrent requests for each user was set to 100. Three deployment configurations were evaluated. We first deployed MJPetStore on one VM with 3 CPUs and 6GB RAM. This configuration was named "1Large". Then we deployed MJPetStore on three VMs, each with 1 CPU and 2GB RAM. The data, computation and communication tasks were deployed on different VMs. This configuration was named "3Small(D)". Then we deployed the three types of tasks on the same VM with workloads evenly distributed across all VMs. This configuration was named "3Small(S)". In order to understand the difference between the energy consumption of the target cloud system and the application, we first measured the power consumption of the cloud server in idle state with one Large VM and then three Small VMs. After that, we measured the power consumption of the cloud server with workload under different deployment configurations.

The energy consumption and system throughput under different deployment strategies are presented in Fig.7 (a) and (b) respectively. Although the total resources such as CPU and RAM allocated were the same, the energy consumption decreased when MJPetStore was deployed on multiple VMs compared to a single VM, as shown in Fig.7 (a). The system throughput increased accordingly as displayed in Fig.7 (b). Although the power consumption of the server slightly increased when MJPetstore was deployed on multiple VMs, the service requests sent to MJPetStore could be processed by more concurrent processes compared to one Large VM. This reduced the execution time of the cloud application. As a result, the energy consumption of the system decreased accordingly.

[2] http://www.greenwavereality.com/

In addition, we observed that the energy consumption with deployment configuration "3Small(S)" increased by 0.8% compared to "3Small(D)". The system throughput of "3Small(S)" decreased by 2.1% compared to "3Small(D)". This is because in the client workload we have modelled in this test, the majority of the tasks were communication-intensive. Deploying all communication-intensive tasks on one single VM greatly reduced the overhead of concurrent processes across different VMs. In summary, better energy efficiency and system performance were achieved when MJPetStore was deployed on three VMs and different types of cloud services on different VM.

C. User Evaluation

We carried out a user evaluation of StressCloud with 7 researchers. All selected participants had different backgrounds and experience levels: 3 participants were postdoctoral research fellows and the other 4 participants were PhD candidates; 2 participants had 2~7 years working experience in industry related to software engineering; 2 participants had experiences in performance engineering. All participants had a rough understanding of cloud computing. Each participant was asked to profile the energy consumption and system performance of a simple modelled cloud application on one virtual machine. The selected cloud application in this user evaluation contained one computation-intensive task, one data-intensive task and one communication-intensive task. Before starting the experiment, each participant was given a 20 minutes introduction to StressCloud. Then the participants were asked to complete the following tasks: 1) Create the workload model and augment it by specifying the properties of each workload model component; 2) Create the cloud architecture model and augment it by specifying properties of architecture model components; 3) Deploy workload to target cloud platform; 4) Generate load test scripts; and 5) Run load test and view visualised data.

We then interviewed them to gain their perceptions of StressCloud's utility and usability. All participants found that StressCloud was very useful for automating the energy and performance profiling process of cloud applications. They also found StressCloud was easy to use and understand. Qualitatively, they stated that it was easy to use StressCloud to model the application workload and cloud architecture, and apply load tests. Particularly, 2 participants who had experiences in performance engineering and used JMeter[3] stated that StressCloud was much easier to use and more efficient than JMeter in terms of test scripts generation. StressCloud provides a visualised user interface for the performance engineer to automatically generate load test scripts while in JMeter all test scripts are manually specified. However, most of them would like a better approach to automate augmenting the workload model and architecture model instead of manually specifying the properties of each model element.

We compared the time taken to evaluate the system performance and energy consumption of JPetStore manually to using StressCloud. This work efficiency comparison compares the total effort to build workload and energy test-bed models, conduct all the load tests, and profile the system performance and energy data. All of the load tests were undertaken by an experienced performance engineer who fully understood the target cloud application and the evaluation process of system performance and energy consumption before StressCloud development. Manually completing the whole process took 2 months of full-time work while repeating this with StressCloud only took 1.5 weeks. An additional one-off overhead of approximately 2 hours was incurred for the engineer to become familiar with StressCloud. However, extensive further analysis activities with StressCloud can be performed within hours or even minutes, i.e. changing deployment models, workload probabilities and tasks, re-running tests. Conducting these analysis activities manually would take extensive application and test redevelopment running into weeks. We are currently undertaking a more extensive user evaluation to achieve more reliable results.

VII. DISCUSSION

Most current cloud application load test tools require significant programming effort of performance engineers in building prototypical cloud-side service components [7, 19]. This requires considerable knowledge of the cloud application under test and a lot of effort to build and maintain the prototypes, especially for large cloud applications or cloud applications which the performance engineer is unfamiliar with. In contrast, StressCloud provides the ability to build a model of the cloud application under test by comprising a mix of data, computation and communication tasks under different workloads. In addition, StressCloud can also be used to stress a deployed real-world cloud application. In this case, the performance engineer needs to specify which deployed cloud services to invoke, including parameter data to send to the application services.

The key advantages of StressCloud include: 1) its use of a formal model for user behaviour modelling; 2) the ability to automatically generate the load tests scripts and cloud service deployment scripts; and 3) the ability to automatically run load test and profile system performance and energy data. Unlike simulation-based modelling for such analysis [7], StressCloud allows cloud services to be deployed and run in a real-world cloud environment, resulting in more accurate and realistic estimation of cloud application performance and energy consumption. The stochastic form chart model used by StressCloud allows the performance engineer to model user behaviours as probabilistic interactions between user request and cloud server response. The accuracy of using stochastic form chart to model user behaviour has been presented and discussed in [17]. Performance engineers can build different workload models for all or part of a cloud application to compare and contrast performance and energy consumption under different user behaviours and deployment configurations.

The key limitation of StressCloud is the adequacy of the stochastic form chart model to capture the "realistic" user behaviours of cloud applications. StressCloud's stochastic form

[3] https://jmeter.apache.org/

charts rely on the performance engineer specifying user behaviours and probabilities of different user interactions with cloud applications and parameters of invoked cloud services. These are thus as sensitive to erroneous data as in any other performance testing tool. One mitigating approach we have adopted for StressCloud is to allow multiple form chart models to be defined for the same target cloud application. This allows the performance engineer to experiment with a variety of different workload modes and compare cloud application's performance and energy under different conditions.

Currently, StressCloud only supports VMware as the cloud platform hypervisor. A key piece of future work is creating a set of templates to traverse the workload and cloud architecture model in order to generate load test scripts and deployment scripts for the selected cloud platform. This will allow a performance engineer to add more templates to StressCloud to support more cloud platforms.

VIII. Conclusion and Future Work

Finding the best deployment configuration for cloud applications to maximise energy efficiency and performance is an extremely challenging task, due to the complexity and heterogeneity of cloud applications and their deployment platforms. In this paper, we presented StressCloud, a novel performance and energy consumption analysis tool for cloud applications in real-world cloud environments. StressCloud supports: 1) modelling realistic cloud application workloads; 2) automatic deployment of load test services; 3) automatic generation of load tests; and 4) automatic profiling of the performance and energy consumption of cloud applications. We demonstrated the effectiveness of StressCloud by analysing the performance and energy consumption of an example cloud application with a range of different deployment configurations. The experimental results demonstrated that, with the support of StressCloud, the performance and energy consumption of cloud applications in realistic cloud environments can be collected and analysed in an effective and efficient manner. Our key future work is creating templates for traversing the workload and cloud architecture model to support more cloud platforms.

Acknowledgment

We thank Professor Ryszard Kowalczyk for providing the facilities of Swinburne Energy Research Lab. The authors wish to acknowledge substantial support provided by NICTA, and partial support from Australian Research Council under Discovery Project DP110101340. NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program.

References

[1] Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R.H., et al.: "Above the clouds: a Berkeley view of cloud computing". UC Berkeley Reliable Adaptive Distributed Systems Laboratory, USA, Technical Report, UCB/EECS-2009-28, Feb 10, 2009.

[2] Hamilto, J.: "Cooperative expendable micro-slice servers (CEMS): low cost, low power servers for internet-scale services". in *the 4th Biennial Conference on Innovative Data Systems Research(CIDR2009)*, Asilomar, California, USA, 2009, pp. 1-8.

[3] Lee, Y.C., and Zomaya, A.Y.: "Energy efficient utilization of resources in cloud computing systems", The Journal of Supercomputing, 2012, 60(2), pp. 268-280.

[4] Wang, Y.A., Huang, C., Li, J., and Ross, K.W.: "Estimating the Performance of Hypothetical Cloud Service Deployments: A Measurement-Based Approach". in *the 30th IEEE International Conference on Computer Communications(INFOCOM2011)*, Shanghai, China, 2011, pp. 2372-2380.

[5] Calheiros, R.N., Ranjan, R., Beloglazov, A., Rose, C.A.F.D., and Buyya, R.: "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms", Software - Practice and Experience, 2011, 41(1), pp. 23-50.

[6] Kamra, M., and Manna, R.: "Performance of Cloud-Based Scalability and Load with an Automation Testing Tool in Virtual World". in *IEEE the 8th World Congress on Services*, Honolulu, HI, USA, 2012, pp. 57-64.

[7] Wickremasinghe, B., Calheiros, R.N., and Buyya, R.: "CloudAnalyst: A CloudSim-based Visual Modeller for Analysing Cloud Computing Environments and Applications". in *the 24th IEEE International Conference on Advanced Information Networking and Applications*, Perth, Australia, 2010, pp. 446-452.

[8] Kliazovich, D., Bouvry, P., and Khan, S.U.: "GreenCloud: a packet-level simulator of energy-aware cloud computing data centers", The Journal of Supercomputing, 2012, 62(3), pp. 1263-1283.

[9] Gupta, S.K.S., Gilbert, R.R., Banerjee, A., Abbasi, Z., Mukherjeey1, T., et al.: "GDCSim: A Tool for Analyzing Green Data Center Design and Resource Management Techniques". in *the 2011 International Green Computing Conference and Workshops(IGCC2011)*, Orlando, FL, USA, 2011, pp. 1-8.

[10] Jayasinghe, D., Swint, G., Malkowski, S., Li, J., Wang, Q., et al.: "Expertus: A Generator Approach to Automate Performance Testing in IaaS Clouds". in *5th IEEE International Conference on Cloud Computing (CLOUD2012)*, Honolulu, Hawaii, USA, 2012, pp. 115-122.

[11] Grundy, J., Cai, Y., and Liu, A.: "SoftArch/MTE: Generating Distributed System Test-beds from High-level Software Architecture Descriptions", Automated Software Engineering(ASE), 2005, 12(1), pp. 5-39.

[12] Metri, G., Srinivasaraghavan, S., Shi, W., and Brockmeyer, M.: "Experimental Analysis of Application Specific Energy Efficiency of Data Centers with Heterogeneous Servers". in *the 5th IEEE International Conference on Cloud Computing*, Honolulu, Hawaii, USA, 2012, pp. 786-793.

[13] Chen, F., Grundy, J., Yang, Y., Schneider, J.-G., and He, Q.: "Experimental Analysis of Task-based Energy Consumption in Cloud Computing Systems". in *the 4th ACM/SPEC International Conference on Performance Engineering(ICPE2013)*, Prague, Czech Republic, 2013, pp. 295-306.

[14] Núñez, A., Vázquez-Poletti, J.L., Caminero, A.C., ·, G.G.C., Carretero, J., et al.: "iCanCloud: A Flexible and Scalable Cloud Infrastructure Simulator", Journal of Grid Computing, 2012, 10(1), pp. 185-209.

[15] Seung-Hwan, L., Sharma, B., Nam, G., Kim, E.K., and Das, C.R.: "MDCSim: A Multi-tier Data Center Simulation Platform". in *the 2009 IEEE International Conference on Cluster Computing(CLUSTER2009)*, New Orleans, Louisiana, USA, 2009, pp. 1-9.

[16] Grundy, J., Cai, Y., and Liu, A.: "Generation of Distributed System Test-beds from High-level Software Architecture Descriptions". in *the 16th Annual International Conference on Automated Software Engineering (ASE 2001)*, San Diego, CA, USA, 2001, pp. 193-200.

[17] Draheim, D., Grundy, J., Hosking, J., Lutteroth, C., and Weber, G.: "Realistic Load Testing of Web Applications". in *the 10th European Conference on Software Maintenance and Reengineering (CSMR'06)*, Bari, Italy, 2006, pp. 70-81.

[18] Chen, F., Schneider, J.-G., Yang, Y., Grundy, J., and He, Q.: "An energy consumption model and analysis tool for Cloud computing environments". in *the 1st International Workshop on Green and Sustainable Software(GREENS2012)*, Zurich, Switzerland, 2012, pp. 45-50.

[19] Sobel, W., Subramanyam, S., Sucharitakul, A., Nguyen, J., Hubert Wong, et al.: "Cloudstone: Multi-Platform, Multi-Language Benchmark and Measurement Tools for Web 2.0". in *the 1st Workshop on Cloud Computing and Its Applications(CCA)*, 2008, pp. 1-7.