In ACM/IEEE International Conference on Software Engineering (ICSE 2015), Florence, Italy, 18-23 May 2013 © IEEE

StressCloud: A Tool for Analysing Performance and Energy Consumption of Cloud Applications

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

***Abstract*— Finding the best deployment configuration that maximises energy efficiency while guaranteeing system performance of cloud applications is an extremely challenging task. It requires the evaluation of system performance and energy consumption under a wide variety of realistic workloads and deployment configurations. This paper demonstrates StressCloud, an automatic performance and energy consumption analysis tool for cloud applications in real-world cloud environments. StressCloud supports 1) the modelling of realistic cloud application workloads, 2) the automatic generation and running of load tests, and 3) the profiling of system performance and energy consumption. A demonstration video can be accessed at: https://www.youtube.com/watch?v=0l4_a_CNtVQ**

I. Introduction

Cloud computing is a new paradigm where users lease cloud infrastructures and services rather than buy them outright. 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. Currently, power consumption contributes up to 42% of a data centre's total operational expense [1]. In addition, such a huge amount of power consumption accelerates global warming [2]. On the other hand, cloud service providers must provide their users with satisfactory system performance, measured usually in throughput and response time [3]. Thus, cloud service providers must develop cloud application deployment and management solutions that guarantee system performance and Service Level Agreement (SLAs) at least energy consumption possible.

However, finding such a solution is an extremely challenging task, due to the complexity and heterogeneity of cloud applications and deployment platforms. It 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 simple cloud applications. 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 load tests based on a realistic user behaviour model in a real-world cloud environment.

A lot of research effort has been devoted to building performance evaluation tools for cloud systems [4-6]. 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) [4, 5]. In addition, only a few cloud load test tools have taken into account the energy consumption of data centres [6]. Moreover, most of the existing performance evaluation tools utilise simulated cloud environments, providing only approximations of cloud system performance [4, 6]. The key limitation of simulation-based modelling is that test results may be significantly inaccurate because of the imperfection in the assumptions, input data, work tasks, energy usage and performance in the simulation environment [7]. None of the existing performance evaluation tools can analyse both system performance and energy consumption at the same time in real-word cloud environments.

This paper demonstrates 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 details; (2) the ability to model cloud application deployment configurations at varying levels of details; (3) automatic generation of detailed load test plans; (4) support for automatic running of load tests; and (5) automatic monitoring, profiling and analysis of system performance and energy consumption.

II. Overview of StressCloud

Fig.1 shows how StressCloud is used to perform load tests to profile the performance and energy consumption of a cloud application. First, the performance engineer defines the cloud application workload model (box 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 [8]. A "composite task" is used to represent tasks made up of different kinds of behaviour. This model is then further augmented by the performance engineer with transition probabilities and properties between tasks. 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* [9] is created to specify detailed user requests and required responses from the cloud system. StressCloud can also be used to stress-test existing cloud applications by specifying which deployed cloud services to invoke and the parameter data sent to the deployed application services.

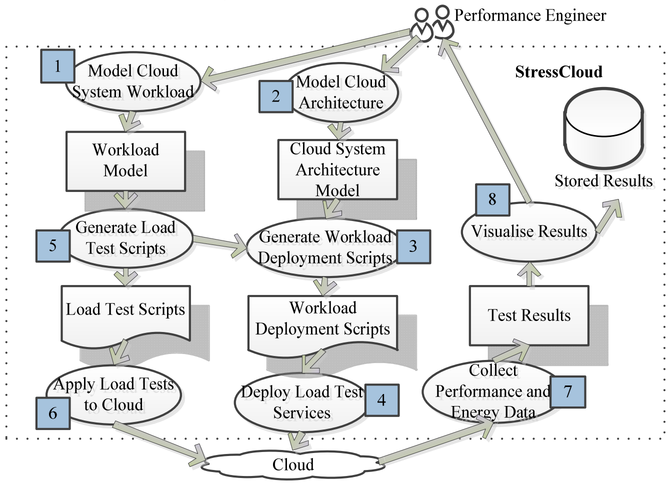

Performance Engineer
StressCloud
1
Model Cloud System Workload
2
Model Cloud Architecture
Workload Model
Cloud System Architecture Model
Stored Results
8
Visualise Results
5
Generate Load Test Scripts
Generate Workload Deployment Scripts
3
Load Test Scripts
Workload Deployment Scripts
Test Results
Apply Load Tests to Cloud
6
Deploy Load Test Services
4
Collect Performance and Energy Data
7
Cloud

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

A cloud system architecture model is defined to specify the elements in the target cloud system (box 2). Our cloud architecture model includes all available resources in the target cloud system and their detailed configurations. Resources of different types can be specified by different resource locations, such as physical servers and virtual machines. After mapping the tasks defined in the user workload model to corresponding resources in the cloud system architecture model, workload deployment scripts are generated (box 3). Based on the deployment scripts, load test services are uploaded and deployed to the virtual machines in the target cloud system (box 4). These load test services have been developed based on our previous research that incorporated CPU, RAM and data-intensive tasks, and supported service-to-service communication-intensive tasks [8]. Load test scripts are then automatically generated based on the workload model (box 5). Next, the specified load tests are performed automatically on the target cloud system in accordance with the load test scripts (box 6). The performance and energy consumption data of the target cloud system are collected (box 7) and then visualised using a variety of charts (box 8).

III. Example Usage

StressCloud is realised as a set of Eclipse IDE plug-ins. A set of editors are used to support diagrammatic modelling of workloads and deployment platforms. 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. Diagrammatic editors are instantiated for editing these models using the Eclipse Graphical Modelling Framework. We illustrate the usage of key functions of StressCloud using a reference Java application JPetStore [10] as an example.

Workload Modelling. To model the workload of JPetStore, a high-level workload model using an extended stochastic form chart needs to be defined first. An example is shown in Fig.2. In this example, the client (represented by top left icon) is modelled with a set of requests (represented by large icons with sub-request containers) linked to transitions (represented by arrowed lines) with probability annotations. The start and end circle icons define a state-chart-style model for the workload. This example specifies that the the user selects a task (*Signin*, *GetProduct*, *GetIndex*, *GetHelp*, *GetCategory*, *GetCart*, *CreateNewAccount*, *GetProductDetail*, or *CheckOut*) with different probabilities after the workload starts.

Each task in the workload model 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” types of service tasks (i.e., data-intensive, computation-intensive or communication-intensive). These task types are used to model 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.3 shows an example of three JPetStore sub-task models, (a) a communication-intensive sub-task (modelling a client-to-server or service-to-service communication); (b) a computation-intensive sub-task (modelling the information processing on a server node); and (c) a 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.3.

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.4. A cloud platform comprises physical server hosts, virtual machines and networks. Virtual machines have many configuration parameters, e.g., the virtual memory size and the number of CPU cores, deployed application software services such as web servers, database servers, etc. Physical hosts and networks have various characteristics, e.g., the server type, the number of CPU cores, the amount of physical memory, the virtual machine hypervisor type, the bandwidth, etc. Fig.4 (a) shows a high-level model of the cloud platform architecture for running JPetStore in our example performance and energy analysis tests. This is a data centre with one physical host and three virtual machine groups on which different tasks will be hosted. Various configuration parameters are shown. Fig.4 (b) shows performance engineer specifying a particular virtual machine configuration for the physical host machine and its hypervisor, in this case a VMWare hypervisor.

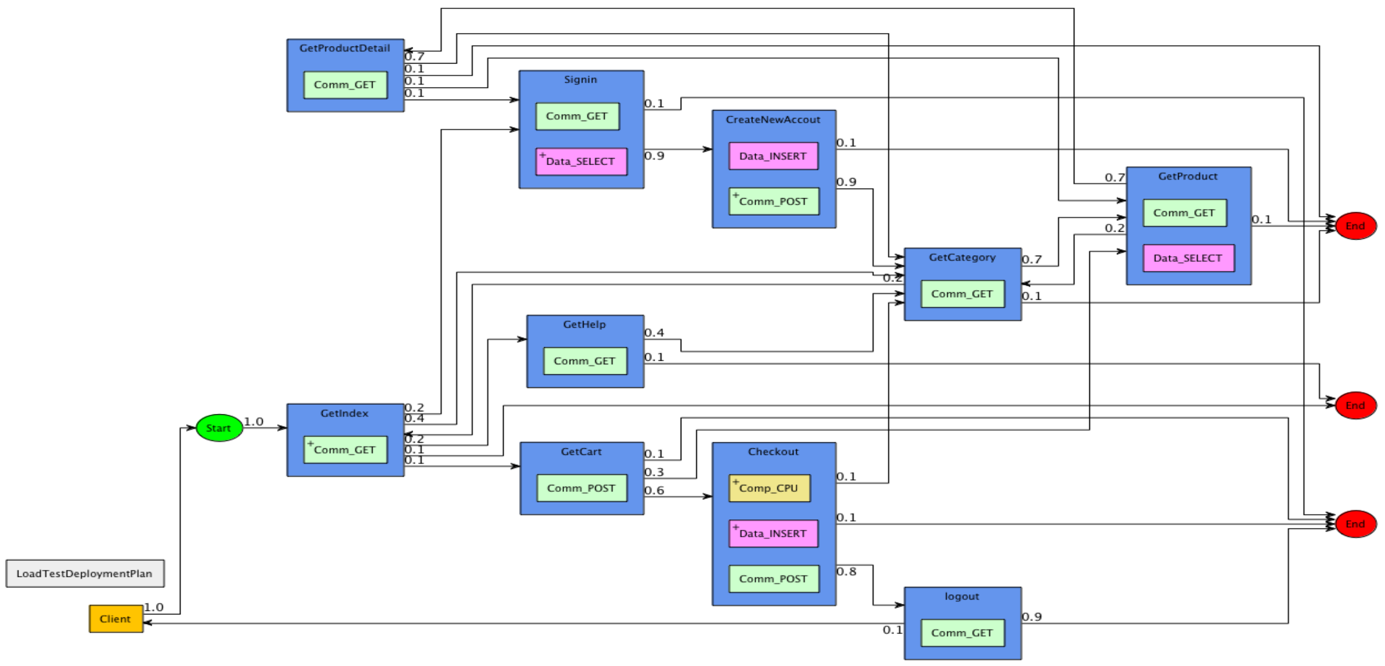
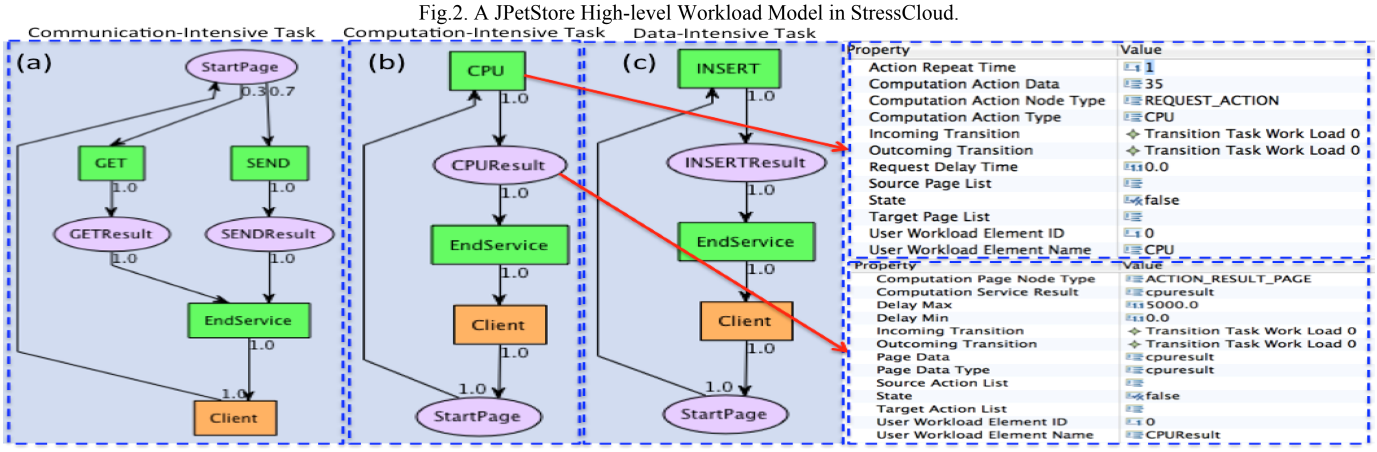


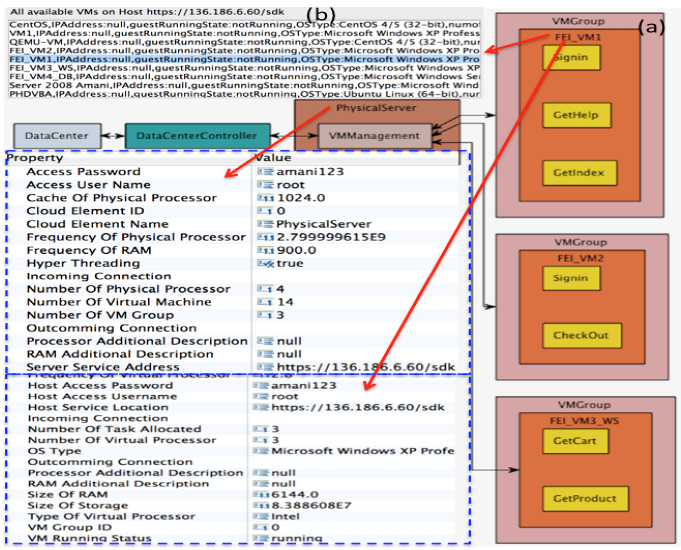
Fig.2. A JPetStore High-level Workload Model in StressCloud.



Property	Value
Action Repeat Time	1
Computation Action Data	35
Computation Action Node Type	REQUEST_ACTION
Computation Action Type	CPU
Incoming Transition	Transition Task Work Load 0
Outcoming Transition	Transition Task Work Load 0
Request Delay Time	0.0
Source Page List	
State	false
Target Page List	
User Workload Element ID	0
User Workload Element Name	CPU

Property	Value
Computation Page Node Type	ACTION_RESULT_PAGE
Computation Service Result	cpuresult
Delay Max	5000.0
Delay Min	0.0
Incoming Transition	Transition Task Work Load 0
Outcoming Transition	Transition Task Work Load 0
Page Data	cpuresult
Page Data Type	cpuresult
Source Action List	
State	false
Target Action List	
User Workload Element ID	0
User Workload Element Name	CPUResult

Fig.3. Stochastic Form Chart Models of Communication-intensive Task (a), Computation-intensive Task (b) and Data-intensive Task (c).

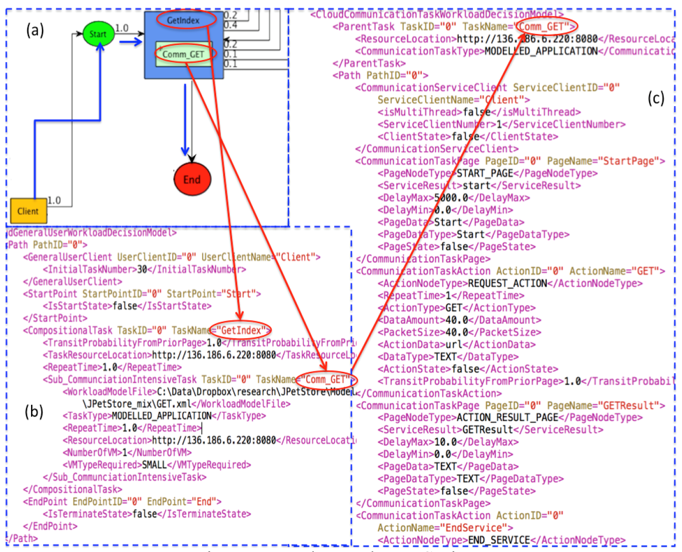


```
All available VMs on Host https://136.186.6.60/sdk
CentOS,IPAddress:null,guestRunningState:notRunning,OSType:CentOS 4/5 (32-bit),numot
VM1,IPAddress:null,guestRunningState:notRunning,OSType:Microsoft Windows XP Profess
QEMU-VM,IPAddress:null,guestRunningState:notRunning,OSType:CentOS 4/5 (32-bit),nur
FEI_VM2,IPAddress:null,guestRunningState:notRunning,OSType:Microsoft Windows XP Pro
FEI_VM1,IPAddress:null,guestRunningState:notRunning,OSType:Microsoft Windows XP Pro
FEI_VM3_WS,IPAddress:null,guestRunningState:notRunning,OSType:Microsoft Windows XP
FEI_VM4_DB,IPAddress:null,guestRunningState:notRunning,OSType:Microsoft Windows Se
Server 2008 Amani,IPAddress:null,guestRunningState:notRunning,OSType:Microsoft Wind
PHDVBA,IPAddress:null,guestRunningState:notRunning,OSType:Ubuntu Linux (64-bit),nur
```

Property	Value
Access Password	amani123
Access User Name	root
Cache Of Physical Processor	1024.0
Cloud Element ID	0
Cloud Element Name	PhysicalServer
Frequency Of Physical Processor	2.799999615E9
Frequency Of RAM	900.0
Hyper Threading	true
Incoming Connection	
Number Of Physical Processor	4
Number Of Virtual Machine	14
Number Of VM Group	3
Outcomming Connection	
Processor Additional Description	null
RAM Additional Description	null
Server Service Address	https://136.186.6.60/sdk

Host Access Password	amani123
Host Access Username	root
Host Service Location	https://136.186.6.60/sdk
Incoming Connection	
Number Of Task Allocated	3
Number Of Virtual Processor	3
OS Type	Microsoft Windows XP Profe
Outcomming Connection	
Processor Additional Description	null
RAM Additional Description	null
Size Of RAM	6144.0
Size Of Storage	8.388608E7
Type Of Virtual Processor	Intel
VM Group ID	0
VM Running Status	running

Fig.4. An Example Cloud Architecture Modelled in StressCloud.



```
dGeneralUserWorkloadDecisionModel>
Path PathID="0">
  <GeneralUserClient UserClientID="0" UserClientName="Client">
    <InitialTaskNumber>30</InitialTaskNumber>
  </GeneralUserClient>
  <StartPoint StartPointID="0" StartPoint="Start">
    <IsStartState>false</IsStartState>
  </StartPoint>
  <CompositionalTask TaskID="0" TaskName="GetIndex">
    <TransitProbabilityFromPriorPage>1.0</TransitProbabilityFromPri
    <TaskResourceLocation>http://136.186.6.220:8080</TaskResourceLo
    <RepeatTime>1.0</RepeatTime>
    <Sub_CommunciationIntensiveTask TaskID="0" TaskName="Comm_GET">
      <WorkloadModelFile>C:\Data\Dropbox\research\JPetStore\Model
        \JPetStore_mix\GET.xml</WorkloadModelFile>
      <TaskType>MODELLED_APPLICATION</TaskType>
      <RepeatTime>1.0</RepeatTime>
      <ResourceLocation>http://136.186.6.220:8080</ResourceLocati
      <NumberOfVM>1</NumberOfVM>
      <VMTypeRequired>SMALL</VMTypeRequired>
    </Sub_CommunciationIntensiveTask>
  </CompositionalTask>
  <EndPoint EndPointID="0" EndPoint="End">
    <IsTerminateState>false</IsTerminateState>
  </EndPoint>
</Path>
```

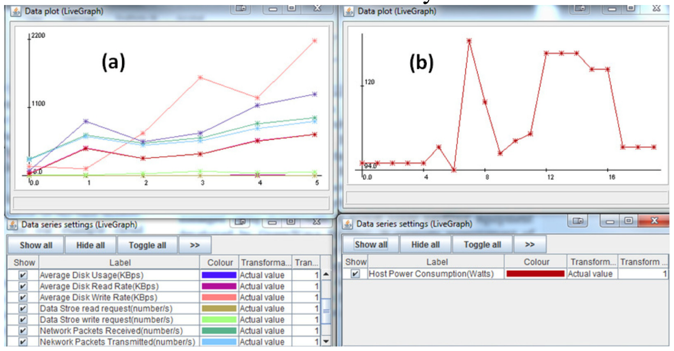
```
<CloudCommunicationTaskWorkloadDecisionModel>
  <ParentTask TaskID="0" TaskName="Comm_GET">
    <ResourceLocation>http://136.186.6.220:8080</ResourceLoca
    <CommunicationTaskType>MODELLED_APPLICATION</Communicati
  </ParentTask>
  <Path PathID="0">
    <CommunicationServiceClient ServiceClientID="0"
      ServiceClientName="Client">
      <isMultiThread>false</isMultiThread>
      <ServiceClientNumber>1</ServiceClientNumber>
      <ClientState>false</ClientState>
    </CommunicationServiceClient>
    <CommunicationTaskPage PageID="0" PageName="StartPage">
      <PageNodeType>START_PAGE</PageNodeType>
      <ServiceResult>start</ServiceResult>
      <DelayMax>5000.0</DelayMax>
      <DelayMin>0.0</DelayMin>
      <PageData>Start</PageData>
      <PageDataType>Start</PageDataType>
      <PageState>false</PageState>
    </CommunicationTaskPage>
    <CommunicationTaskAction ActionID="0" ActionName="GET">
      <ActionNodeType>REQUEST_ACTION</ActionNodeType>
      <RepeatTime>1</RepeatTime>
      <ActionType>GET</ActionType>
      <DataAmount>40.0</DataAmount>
      <PacketSize>40.0</PacketSize>
      <ActionData>url</ActionData>
      <DataType>TEXT</DataType>
      <ActionState>false</ActionState>
      <TransitProbabilityFromPriorPage>1.0</TransitProbabil
    </CommunicationTaskAction>
    <CommunicationTaskPage PageID="0" PageName="GETResult">
      <PageNodeType>ACTION_RESULT_PAGE</PageNodeType>
      <ServiceResult>GETResult</ServiceResult>
      <DelayMax>10.0</DelayMax>
      <DelayMin>0.0</DelayMin>
      <PageData>TEXT</PageData>
      <PageDataType>TEXT</PageDataType>
      <PageState>false</PageState>
    </CommunicationTaskPage>
    <CommunicationTaskAction ActionID="0"
      ActionName="EndService">
      <ActionNodeType>END_SERVICE</ActionNodeType>
```

Fig.5. Example Load Test Scripts.

Deployment Plan Generation. The model of the cloud platform is used to generate automated configuration scripts in order to configure the platform for load test runs. The generated scripts are uploaded to the target physical and virtual machines and executed to configure them.

Load Test Plan Generation. After generating the service configuration scripts and deploying the services, the performance engineer uses StressCloud to generate the load test plans. Fig. 5 shows an example of generated load test scripts and configurations. The load test model (a) is used to synthesise a test script that will be run on a client machine (b). This script models the state machine that describes the sequence of tasks, the transitions between tasks, the probabilities that each task will be carried out, the 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 as in this example, a model of that service (c). The service model is a set of data-, computation- and communication-intensive sub-tasks. Each task's sub-model includes the parameters of the services involved in the task service model. These parameters specify, for example, the number of iterations for storing, computing, sending and receiving 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, etc.

Test Running and Results Visualisation. Once the load test scripts are generated, they are uploaded to one or more machines acting as "clients" and run. The performance engineer may ask to run hundreds or thousands of instances of such clients simultaneously. Clients running the load tests can be hosted on the same or different physical machine, depending on machine availability. Different types and numbers of workload tests can be run simultaneously.



Show	Label	Colour	Transforma...	Tran...
✓	Average Disk Usage(KBps)	🟦	Actual value	1
✓	Average Disk Read Rate(KBps)	🟪	Actual value	1
✓	Average Disk Write Rate(KBps)	🟥	Actual value	1
✓	Data Stroe read request(number/s)	🟫	Actual value	1
✓	Data Stroe write request(number/s)	🟩	Actual value	1
✓	Network Packets Received(number/s)	🟩	Actual value	1
✓	Nekwork Packets Transmitted(number/s)	🟦	Actual value	1

Show	Label	Colour	Transform...	Transform ...
✓	Host Power Consumption(Watts)	🟥	Actual value	1

Fig.6. Visualised Performance (a) and Energy Data (b).

The results of the load tests are periodically collected and visualised in an interactive manner for the performance engineer. Fig.6 shows an example of the JPetStore deployment models and workload models under stress tests. Various physical machine performance metrics can be collected (a). The energy consumption of physical servers and routers are also collected (b) using PowerNode [11]. Different collection intervals and parameters can be set. Results can be saved for post-hoc analysis, and comparative analysis of different workloads and deployment configurations.

IV. Discussion and Summary

StressCloud is a novel tool for analysing the performance and energy consumption of cloud applications in real-world cloud environments. StressCloud supports the modelling of realistic cloud application workloads, automatic deployment of load test services, automatic generation and running of load tests, and automatic profiling of the performance and energy consumption of cloud applications. We evaluated its effectiveness by reproducing previous load test results manually obtained in a real cloud environment [8]. With the same workload and cloud architecture models, we observed consistent energy consumption and system performance variations. In addition, manually completing the whole process took 2 months of full-time work while with StressCloud it only took 1.5 weeks. With the support of StressCloud, performance engineers are able to collect and analyse the performance and energy consumption of cloud applications in realistic cloud environments in an efficient and effective manner.

Acknowledgment

We thank Professor Ryszard Kowalczyk for providing the facilities of Swinburne Energy Research Lab. This research is supported by the Australian Research Council under Discovery Project DP110101340 and NICTA Swinburne Software Innovation Lab.

References

[1] 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*, Asilomar, California, USA, 2009, pp. 1-8.

[2] 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.

[3] 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*, Shanghai, China, 2011, pp. 2372-2380.

[4] 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.

[5] 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,USA, 2012, pp. 57-64.

[6] 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.

[7] 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.

[8] 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*, Prague, Czech Republic, 2013, pp. 295-306.

[9] 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*, Bari, Italy, 2006, pp. 70-81.

[10] http://java.sun.com/developer/releases/petstore/, accessed January 28, 2015

[11] http://www.greenwavereality.com/, accessed January 28, 2015