2014 IEEE 6th International Conference on Cloud Computing Technology and Science

Resource Requirement Analysis for Web Applications Running in a Virtualised Environment

Rafidah Pakir Mohamad, Dimitrios S. Kolovos, Richard F. Paige
Department of Computer Science
University of York
York, United Kingdom
{*rpm507, dimitris.kolovos, richard.paige*}*@york.ac.uk*

***Abstract*—Analysis of resource usage and precise correlation with the workload that triggered it is essential in order to conduct capacity planning in computing environments. Virtualisation enables resource optimisation and is widely used in grid, cluster and cloud computing. We present an automated approach for resource usage analysis in a virtualised environment that can support capacity planning for web applications. The approach uses Domain Specific Modelling Languages (DSMLs) and model management techniques, which support tool interoperation and provide precise ways for describing resource and request logs and requirements, and automatically generate different outputs that feed into the capacity planning process. The approach is demonstrated using a proof-of-concept example involving a media streaming web application, and the results of the analysis are presented and discussed.**

***Keywords*-resource monitoring, virtualisation, application workload, capacity planning, domain specific modelling;**

I. Introduction

Capacity planning involves predicting future computing resource requirements by monitoring a system's resource usage patterns and comparing them against different workloads [1]. Since usage patterns and workloads vary with time, automated and continuous comparison (as opposed to discrete comparison at specific points in time) is both valuable and desirable. Capacity planning for modern computing environments (e.g., Amazon Web Services) is even more challenging, in part because such environments are virtualised: applications run on virtual machines on shared hardware. This can reduce costs for application deployers and users, but introduces new challenges in capacity planning. Specifically, in virtualised environments, capacity planning is a multi-phase process that involves *end users* (EU), *virtual infrastructure providers* (ViP) and *physical infrastructure providers* (PiP). Both providers need to conduct capacity planning [2] to ensure that sufficient, but not excessive, resources are allocated to meet their contractual obligations in terms of Quality of Service (QoS) attributes such as availability, throughput and latency.

Current capacity monitoring tools are able to collect, store and display resource metrics over time in order to help operators identify resource usage patterns. However, this is not always sufficient for virtual environments; resource usage patterns need to be associated with the workload that is being processed at the time to enable better-informed capacity planning.

We present a DSML-based approach to support resource requirement analysis activities of the capacity planning process. This approach allows to abstract the respective concrete tools and to automate analysis tasks in a precise, flexible and systematic way, using model management languages and tools. The novelties are: metamodels that allow resource and requests logs as well as workloads to be precisely captured using models, as well as a transparent, automated and repeatable Model-Driven Engineering (MDE) process for generating predictions for resource usage from workload models. The MDE process, which exploits model transformation, comparison and merging, is modularised so that it can be configured for different kinds of capacity planning applications and technical infrastructures. An additional contribution is the ability to derive a set of application-specific resource requirement formulas for *resource metrics* (CPU, memory, network and storage). We demonstrate the modelling approach on a proof-of-concept web application example.

The remainder of the paper is organised as follows. Section II provides an overview of literature related to capacity planning and discusses existing approaches for estimating resource requirements. Section III discusses the proposed framework that aims to automate resource requirements analysis for web applications running in a virtualised environment. Section IV demonstrates the implementation of the proposed framework in a virtualised environment for a media streaming web application. Section V concludes the paper and provides directions for further work.

II. Background

In this section we briefly summarise the key research in the area of automated support for capacity planning (particularly focusing on virtual environments). We also

978-1-4799-4093-6/14 $31.00 © 2014 IEEE
DOI 10.1109/CloudCom.2014.134

outline the key concepts in domain-specific modelling which are used in later sections.

A. Capacity planning in virtual environments

As mentioned earlier, capacity planning (in general) involves predicting future computing resource requirements by monitoring a system's resource usage patterns, and comparing them with known or historical workload patterns. Capacity planning in virtual environments aims at ensuring that allocated virtual computing resources such as CPU, memory, storage and network bandwidth will be sufficient to support future computational needs. In this process, available system resources are observed and performance is measured [3]. Also, resource usage patterns are determined to forecast the resources that need to be allocated to serve future workloads in compliance with the service's QoS requirements [4]. To achieve this, it is necessary to identify incoming workloads, to monitor resource usage, and to associate resource usage with the workloads that triggered it.

We next give an overview of key approaches for estimating and predicting resource requirements. We then consider the key tools used for monitoring resource usage (some of which will be used in later sections in our approach for resource requirement analysis).

1) Resource estimation and prediction techniques: Several approaches have been proposed in order to estimate resource requirements for different types of applications, based on workload estimates. A synthetic workload generator tool proposed in [5] evaluates the performance of virtual machines by performing synthetic requests on multi-tier web applications. In [6] a benchmark model is proposed to estimate the number of VMs required for hosting media stream applications based on their memory and disk requirements. In [7], the authors propose a resource allocation algorithm that estimates the number of required VMs based on statistical predictive techniques by considering the challenges of auto-scaling. Microscopic and macroscopic approaches to predict resource consumption for data centres by statistically characterising resource usage patterns are proposed in [8]. The microscopic approach focuses on resource usage prediction for a specific node; it demonstrates that using both CPU and memory usage data can improve the forecasting performance compared to a baseline method of calibrating CPU usage.

2) Resource usage monitoring: Capacity planning requires analysts to monitor resource usage over a number of dimensions: in terms of load (e.g., requests), over time, over outputs, etc. Such analyses rely on the tools available for monitoring resource usage. These include Ganglia[1], VBoxManage[2], CloudWatch[3] and Solarwinds[4]. Such tools are able to collect information related to the usage of resources such as CPU, memory, storage and network by a virtual machine and at predefined intervals (e.g. every 1 second). This information can be stored for further processing and visualisation. As discussed in the sequel, these tools generally provide very similar capabilities.

[1] http://ganglia.sourceforge.net/
[2] https://www.virtualbox.org/
[3] http://aws.amazon.com/cloudwatch/
[4] http://www.solarwinds.com/uk/

B. Domain Specific Modelling

Domain Specific Modelling is a technique for systems and software modelling that advocates the use of Domain Specific Modelling Languages (DSMLs) in particular application domains [9]. In defining a DSML, it is important to have a clear understanding of the key domain concepts that will be expressed in the language, and to understand where information that will be expressed in domain-specific models will come from. In the domain of interest in this paper, the information required to populate models can be gathered from logs recording the *workload* processed by an application running in a virtualised environment, and logs recording the *resource usage* of the VM that hosts the application. The list of key concepts involved in define the DSMLs are listed below:

i ***Resource*** - CPU, memory, storage and bandwidth (incoming and outgoing network);
ii ***Resource usage measurement*** - the amount of CPU, memory, storage and bandwidth used at a given point in time;
iii ***Resource requirement*** - the anticipated need for resources in a future time;
iv ***Workload*** - a structured description of the processing requests issued to the application over a period of time;
v ***Resource usage pattern*** - a structured description of the correlation between the workload and the observed resource usage.

The precise definition of *workload* can vary for different types of software applications; for example, the workload of web applications is often expressed in terms of the number of requests end users make to the application, which does not apply – for instance – to image processing applications, where aspects such as the type, size and format of the processed images are also important.

III. DSML-based Process for Resource Requirement Analysis

In this section, we describe the DSML-based process that we proposed for automating the estimation of resource requirements in capacity planning. DSMLs (and models) are used to precisely specify resource requirements and requests and to facilitate the resource requirement analysis process, which then feeds into a capacity planning process. In this work, DSMLs are used to model:

- *requests* sent to a server
- *resource usage*

- different *groupings* of requests or resources (e.g., requests over a time period, requests against a specific resource)
- *workload patterns*
- *output* from a resource requirement analysis process

The DSMLs provide precise ways to capture these common concepts and structures, used by many different analysis tools. As a result, we can use standardised model management techniques to construct, automatically manipulate and validate such models.

We will shortly present the DSMLs in more detail, but first we describe the overall resource requirement analysis process that can facilitate capacity planning for web applications. A graphical illustration of the process appears in Figure 1. A web application can run on a single or on multiple (virtual or physical) machines. The incoming requests to the application are captured in the web server log. The hosting virtual machine is monitored by a resource monitoring tool which produces a resource usage log. These logs are parsed into *request* and *resource usage* models respectively to shield the rest of the process from hard dependencies to particular web servers and VM monitoring tools. The log models are then compared and transformations are used to produce correlation (grouping) models (step 5) and graphs (step 4 and 8), and to reconcile any measurement errors (step 6). The model management operations are described in more detail in Section III-B.

A. Resource Requirement Analysis DSMLs

As discussed above, we have constructed a set of DSMLs for automating the resource requirement analysis process. Figure 2 illustrates the set of DSMLs and the relations between them, and the following sections provide an overview of their organisation and semantics. The DSMLs are implemented using EMF (Ecore) technology.

1) ***RequestLog:*** Captures information related to incoming requests including the start time, end time and the name of each request extracted from the web server's request log. This information is common to most web servers. The web server's configuration (maximum users, maximum live users, waiting time, and time out) is also captured as an instance of the *Configuration* class. Multiple models that conform to this DSML can be generated if the application is running on multiple machines.
2) ***ResourceLog:*** Information related to usage of selected resources is extracted from log files generated by tools monitoring the resource usage of the machines involved (e.g. Ganglia, BVoxManage). The time granularity of resource usage measurements should be comparable to that of the request log. The CPU and memory allocation of the machine is captured next to its unique identifier (*name*). The attribute *cpuSpeed* of class *Machine* represents the total processor speed of the machine. This value will be later used to convert the percentage of CPU usage recorded by the resource monitoring tool into an absolute figure. Resource measurements for memory, disk and bandwidth (incoming and outgoing network) are also recorded.
3) ***ResourceRequestVsTime:*** The *ResourceRequestVsTime* DSML provides structures for correlating the occupancy of the system (number of active requests) with the usage of each resource at that time. The number of active requests is obtained by comparing and merging the *RequestLog* and *ResourceLog* models.
4) ***ResourceVsRequest:*** This DSML provides structures for grouping resource usage information by the number of concurrent requests that the application was processing at the time of each measurement (i.e. how much CPU/memory etc. the machine consumed while processing 0, 1, 2 .. n concurrent requests).
5) ***WorkloadPattern:*** The workloads are simulated and the observed arrival sequence is capture in a *WorkloadPattern* model. This model is used to adjust synchronisation errors in the *ResourceVsRequest* model caused by differences in the timestamps reported by the web server and the VM monitoring tool.
6) ***ResourceRequirementAnalysis:*** This DSML is an extension of *ResourceVsRequest* and complements the information stored in models conforming to the *ResourceVsRequest* DSML with additional information related to particular analysis techniques. The domain expert can select different analytical techniques to synthesise *ResourceVsRequest* models and use them for capacity planning.

B. Model Management Techniques

Eight model management operations have been implemented for managing models conforming to the proposed DSMLs. In Figure 1, the abbreviations T2M, M2T and M2M stand for Text-to-Model, Model-to-Text and Model-to-Model transformations respectively; the operations referred to in this section have been implemented with relevant components of the Epsilon framework [10]. The functionality of each operation is explained below in accordance to the numbering used in Figure 1:

1) **Request Logs to Request Models:** The workload (request) information stored in *Request Log Files* is extracted through a T2M transformation into *RequestLog Models*. The number of *Request Log Files* and *RequestLog Models* is equal to the number of machines running the application.
2) **Resource Usage Logs to Resource Models:** Similarly, another T2M transformation extracts *ResourceLog Models* from *Resource Usage Log Files* captured using VM monitoring tools.
3) **Sort, Compare & Merge Request and Resource Models:** Necessary adjustments such as sorting and

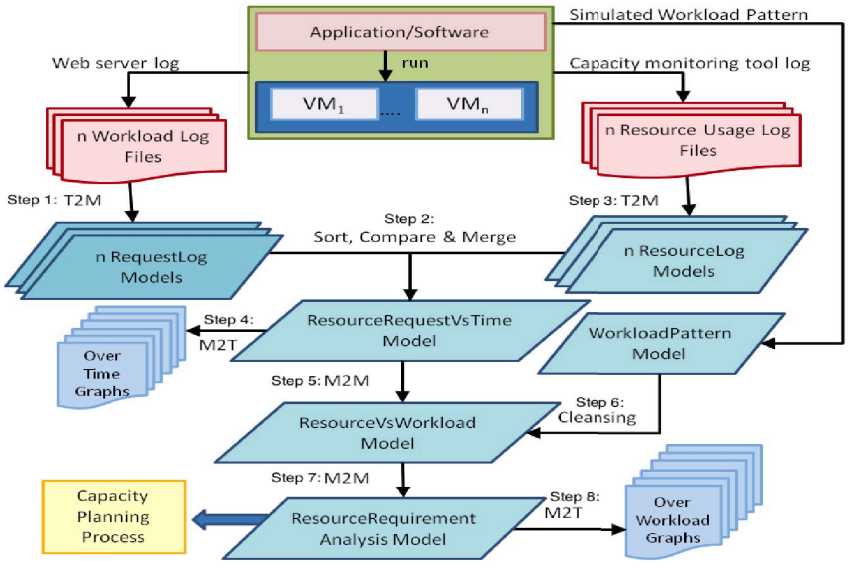

Figure 1: Web Application Resource Requirement Analysis Process.

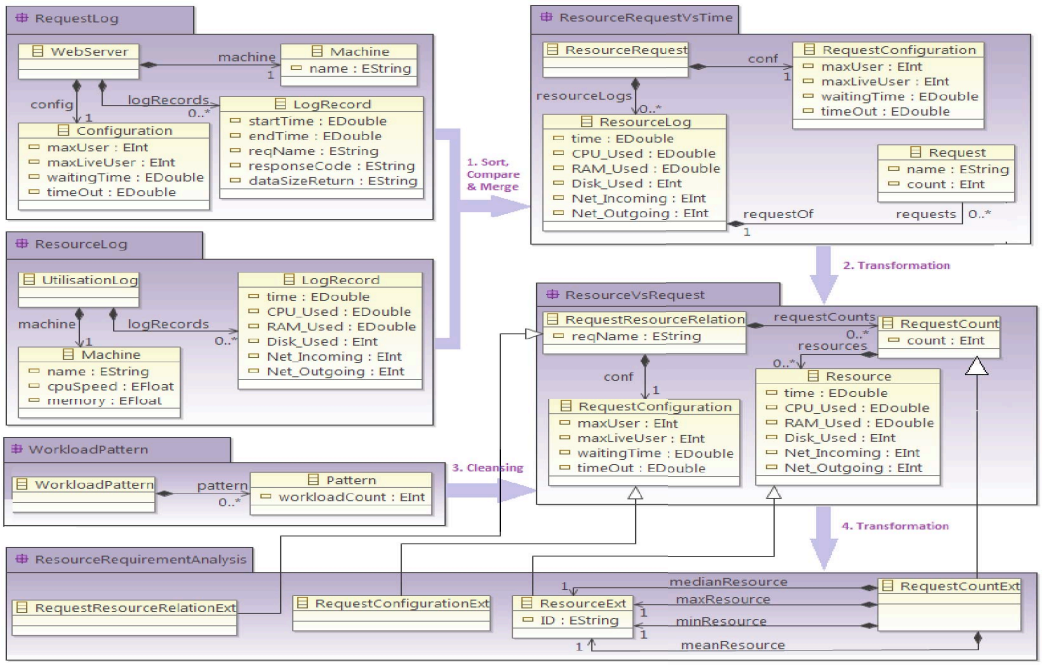

Figure 2: DSMLs for Web Application Resource Requirement Analysis

time conversion are performed on the models before they can be compared and merged. The output of this operation is a single *ResourceRequestVsTime Model* which combines the set of *RequestLog Models* and *ResourceLog Models*.

4) **Capacity Monitoring Graph Generation:** Resource usage and concurrent request count graphs based on time are generated. The outcome of this step is a set of graphs as illustrated in Figure 3 for the case study discussed in Section IV.
5) **Generation of *ResourceVsRequest* Model:** The *ResourceRequestVsTime Model* is analysed and synthesised to produce a *ResourceVsRequest Model*. Multiple resource usage measurements are grouped by the number of concurrent requests that the application was processing at the time they were recorded.
6) **Cleansing:** Performing step 5 can produce noise that can affect correlation due to time synchronisation issues

between the host and the VMs. *WorkloadPattern Model* is used to clean the *ResourceVsRequest Model*. This step is optional.

7) **Produce Analysis Model:** Statistical analysis is performed to compute the minimum, maximum, mean and median value of resource usage measurements with *ResourceVsRequest Model* and the results are stored in a *ResourceRequirementAnalysis Model*.
8) **Resource Requirement Vs Request Count:** A set of graphical presentations of resource and request relationships obtained through the analysis conducted above are generated. Example graphs are illustrated in Figure 4.

IV. Example

In this section we present a proof-of-concept example that demonstrates that the DSMLs and automated process presented in the previous section can be used to support parts of the capacity planning process. In particular, we aim to demonstrate that valuable information in the form of resource utilisation graphs can be automatically produced to aid in capacity planning; we leave scalability and applicability of the approach to other domains as future work.

The representative example that we use is a video streaming application developed with the WordPress Stream Video Player plugin[5], operating in a virtualised environment using VirtualBox; Apache[6] is used as an application server. Workload (request) information is retrieved from Apache's log file and a resource usage log file is generated using the VBManage monitoring tool which comes with VirtualBox. To fully execute the process of Figure 1 we need additional configuration data, which can only be gathered from the specific application under test, and the experimental set-up (e.g., web server used, maximum number of users, number of CPUs, etc). For this proof-of-concept, we used the default Apache server configuration which defines a request timeout of 300 seconds, a queuing time of 5 seconds, and a upper limit of 150 concurrent requests and 100 live requests. A virtual machine with 3 CPUs and 2GB of memory was used and VBoxManage was configured to capture and store resource usage measurements in a log. VBoxManage operates on the host server and therefore its operation does not interfere with that of the virtual machine.

This parameterised data is then transformed into models using the T2M transformations indicated in Figure 1. Artificial workloads involving concurrent requests (R={1, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150}) were generated with JMeter[7] following the request pattern illustrated in Figure 3(F).

The simulation of workloads created two log files; i) a *Request Log* file produced by the web server and ii) a *Resource Usage Log* produced by the VM monitoring tool. Both log files were transformed into *RequestLog* and *ResourceLog* models using text-to-model transformations (steps 1 & 2 in Figure 1). These two models were then sorted, compared and merged (step 3 in Figure 1) based on time, to produce a single model (*ResourceRequestVsTime*). Resource usage and occupancy graphs (Figure 3) were produced using a model-to-text transformation (step 4 in Figure 1). Subsequently, the *ResourceRequestVsTime* model is transformed to a *ResourceVsRequest* model by correlating the calculated occupancy and resource usage. Noise introduced due to technical constraints in the implementation environments (time synchronisation between host and VMs) was then removed by examining the *WorkloadPattern* model.

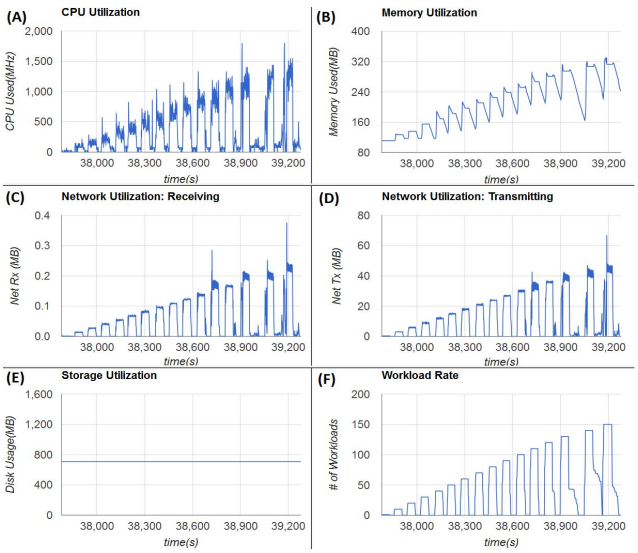

Figure 3: Resource utilisation graph for CPU(A), memory(B), incoming network(C), outgoing network(D) and storage(E). Graph (F) is simulated request pattern.

Then, statistical analysis was performed on the *ResourceVsRequest* model in order to calculate formulas that can predict resource usage for arbitrary workloads. In this case study we used linear regression which produced the following formulas: $CPU = 2.8 * \#requests + 15$, $Memory = 1.438 * \#requests + 120$, $IncomingNetwork = 1458.3 * \#requests$, $OutgoingNetwork = 30590 * \#requests$ and $Storage = 706$. The formulas can be used to both inform the current capacity planning process and to support prognostics, though of course many iterations of experiments must be carried out to increase confidence in the validity of the formulas; what we show here is proof-of-concept.

V. Conclusions

In this paper, we have presented a systematic and largely automated approach to support resource requirement analysis for web applications based on use of DSMLs and auto-

[5]http://wordpress.org/plugins/stream-video-player/
[6]http://httpd.apache.org/
[7]http://jmeter.apache.org/

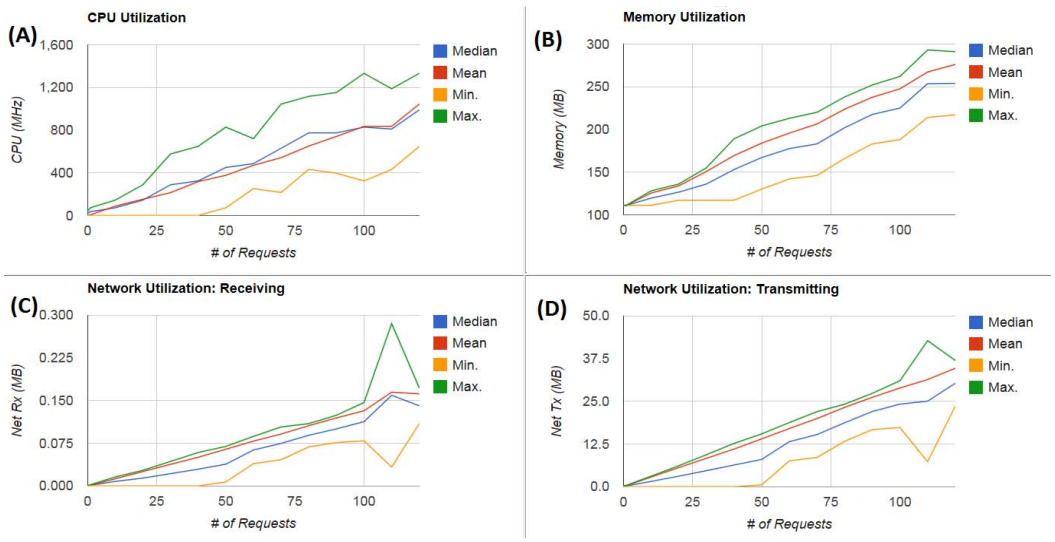

Figure 4: Statistical results of resource usage of CPU(A), memory(B), incoming network(C) and outgoing network(D) for the number of request to view a video file.

mated model management. The practicality and potential of the proposed approach was demonstrated through a representative example involving a media stream web application. The use of DSMLs allows us to abstract from concrete tools (for implementing web applications, for monitoring, for analysing logs), and to automate analysis tasks in a precise, flexible and systematic way. It also opens up new opportunities: we implemented the DSMLs using EMF/Ecore, and the use of these standardised technologies thus makes the DSMLs potentially usable or reusable by others. This in turn can make it easier for developers to define *reusable connections* between tools to support analysis (and hence capacity planning), thus enriching the automated capability for the capacity planning community in general.

The process that we described in the paper is not yet fully automated: configuration data (e.g., number of CPUs being used by the application, number of users) must be provided at the application level. At the moment this is done manually, by editing the models, but automating this further by providing a bespoke editor (e.g., implemented using update-in-place model transformations) may be useful. In future iterations of this work we also plan to carry out further experiments that will target different classes of applications, in particular those with heterogeneous incoming requests (the example used in this paper had only one type of incoming request). This may entail developing a new DSML for modelling predicted request arrivals over time.

REFERENCES

[1] John Allspaw. *The Art of Capacity Planning*. O'Reilly, 2008.

[2] Daniel Gmach, Jerry Rolia, Ludmila Cherkasova, and Alfons Kemper. Capacity Management and Demand Prediction for Next Generation Data Centers. In *Web Services, 2007. ICWS 2007. IEEE International Conference on*, pages 43–50, July 2007.

[3] Barrie Sosinsky. *Cloud Computing Bible*. Wiley Publishing, 2011.

[4] Diego Perez-Palacin, Radu Calinescu, and José Merseguer. log2cloud: Log-based Prediction of Cost-Performance Trade-offs for Cloud Deployments. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC '13, pages 397–404, New York, NY, USA, 2013. ACM.

[5] Arshdeep Bahga and Vijay Krishna Madisetti. Synthetic Workload Generation for Cloud Computing Applications. *Journal of Software Engineering and Applications*, 4(7):396–410, July 2011.

[6] Ludmila Cherkasova, Wenting Tang, and Sharad Singhal. An SLA-Oriented Capacity Planning Tool for Streaming Media Services. In *Proceedings of the 2004 International Conference on Dependable Systems and Networks*, pages 743–752, Washington, DC, USA, 2004. IEEE Computer Society.

[7] Nilabja Roy, Abhishek Dubey, and Aniruddha Gokhale. Efficient Autoscaling in the Cloud Using Predictive Models for Workload Forecasting. In *IEEE International Conference on Cloud Computing (CLOUD)*, pages 500–507, July 2011.

[8] Jian Tan, P. Dube, Xiaoqiao Meng, and Li Zhang. Exploiting Resource Usage Patterns for Better Utilization Prediction. In *31st International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pages 14–19, June 2011.

[9] Thomas Stahl, Markus Völter, Jorn Bettin, Arno Haase, and Simon Helsen. *Model-Driven Software Development: Technology, Engineering, Management*. Wiley, 2006.

[10] Dimitrios Kolovos, Louis Rose, and Richard Paige. *The Epsilon Book*. 2011.