Demonstrating a Versatile Model for VoD Buzz Workload in a Large Scale Distributed Network

Jean-Baptiste Delavoix, Shubhabrata Roy, Thomas Begin and Paulo Gonçalves
Inria, ENS Lyon, UCB Lyon 1, CNRS, UMR 5668
46 Allée d'Italie, 69364 Lyon Cedex 07, France
Email: jean-baptiste.delavoix@grenoble-inp.org, {shubhabrata.roy, thomas.begin, paulo.goncalves}@ens-lyon.fr

***Abstract*—In previous works, we proposed a stochastic model able to reproduce buzz dynamics in a Video on Demand (VoD) workload. We also derived an estimation procedure to calibrate all the model's parameter and evaluated the performance of our estimator on synthetic time series. We showed how can this procedure be applied to fit real workload traces. In this work we demonstrate the model on Grid'5000 with an aim of conducting real-life experiments. Grid'5000 is a highly reconfigurable, controllable and monitorable experimental platform for conducting experiments on large scale parallel and distributed systems. Our results show that the implemented model matches the theoretical model in terms of the mean value and the steady state distribution. We believe this demonstration, by emulating a real world VoD system, can provide data that can serve as an input to frame efficient resource management policies.**

***Index Terms*—Workload Generator, Video on Demand, Distributed Network, Grid'5000**

I. Introduction and Motivation

In recent trend of data-intensive applications the providers must handle the challenge of resource management to adapt to volatile workloads. Thus there is a need for realistic workload generators to evaluate the choice of policies prior to full production deployment. In our work we consider a Video on Demand (VoD) systems as a relevant use case of a data-intensive application where bandwidth usage varies rapidly over time. In [1] we propose in a constructive manner, a stochastic model based workload generator for a Video on Demand (VoD) system, that reproduces workload and traffic volatility. We also developed methods to empirically identify and calibrate parameters to assess the goodness of fit of the model against a real VoD workload test [2]. In this work we implement our model in a large scale distributed network with a controlled environment to emulate real world system. This work constitutes an indispensable step towards our ultimate objective of leveraging the statistical properties of the model and data collected from the experiments to frame resource management policies.

A VoD service delivers video contents to consumers on request. According to Internet usage trends, users are increasingly getting more involved in the VoD and this enthusiasm is likely to grow. According to 2010 statistics a popular VoD provider like Netflix accounts for around 30 percent of the peak downstream traffic in the North America and is the "largest source of Internet traffic overall" [3]. Since VoD has stringent streaming rate requirements, each VoD provider needs to reserve a sufficient amount of server outgoing bandwidth to sustain continuous media delivery. We would like to point out that we are not considering IP multicast here. However, resource allocation often fails to accommodate adequate resources during "buzz" periods when a video becomes popular very quickly leading to a *flood* of user requests on the VoD servers. Figure 1 shows a typical pattern of real VoD server workload trace from [4] picturing the buzz dynamics.

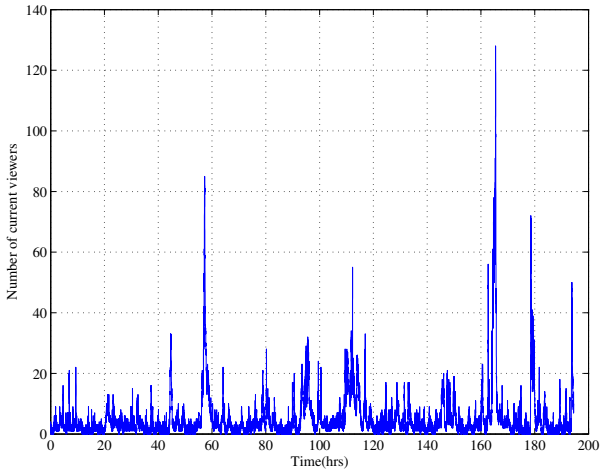

Fig. 1. Real workload time series corresponding to a VoD server demand from [4].

Following epidemic models, we categorize VoD users in three different classes (states). Class S refers to the people who has not watched a video (susceptible viewers), I refers to the people who are currently watching the video and can spread the information about it. I is the workload on the system, but it can also refer to total bandwidth requested at that moment. The class R refers to the past viewers. Posing $(I(t) = i,\ R(t) = r)$ as the current state, Figure 2 depicts the model and the transitions between the states. Here $\beta > 0$ is the rate of information dissemination per unit of time, $l > 0$ fixes the rate of spontaneous viewers, γ^{-1} is the mean watch time of a video. μ^{-1} denotes the mean active period after which an user stops propagating information. We resort to the hidden Markov models to illustrate a buzz like event by considering that β can assume two values depending on its state; $\beta = \beta_1$ in the normal state and $\beta = \beta_2 \gg \beta_1$ in buzz regime. Transition

between these two states occur with rates a_1 and a_2.

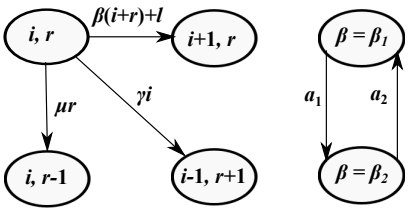

Fig. 2. Markov Chain diagram representing the possible transitions of the number of current (i) and past active (r) viewers.

II. Model implementation

We ran our experimentation on Grid5000, which is a 5000-CPU nationwide grid infrastructure for research in grid computing, providing a scientific tool for computer scientists similar to the large-scale instruments used by physicists, astronomers, and biologists. It is a research tool featuring deep reconfiguration, control, and monitoring capabilities designed for studying large-scale distributed systems and for complementing theoretical models and simulators. As much as 17 French laboratories are involved, and nine sites host one or more clusters of about 500 cores each. A dedicated private optical networking infrastructure interconnects the Grid'5000 sites. In the Grid'5000 platform, the network backbone is composed of private 10-Gb/s Ethernet links. Figure 3 shows the Grid'5000 topology.

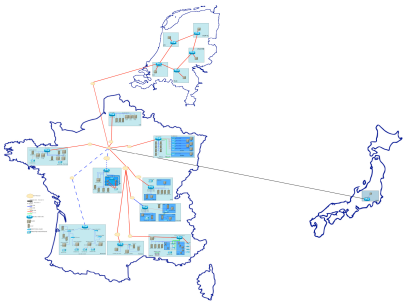

Fig. 3. Topology of Grid'5000

Grid'5000 enables researchers to run successive experiments reproducing the exact experimental conditions several times, a task that is almost impossible with shared and uncontrolled networks. This also ensures large-duration observation windows under stationary conditions, which can not be achieved over the Internet.

A. Global Architecture of the Workload Generating System

In order to generate a certain workload on a VoD server, we consider each node in the Grid'5000 as an user entity. Figure 4 shows how the nodes interact among themselves to emulate user behavior. During implementation, we consider all nodes (users) to be independent. Following a centralized architecture we fix a monitor node, which controls the state of all nodes as-well-as allows and controls communications among them.

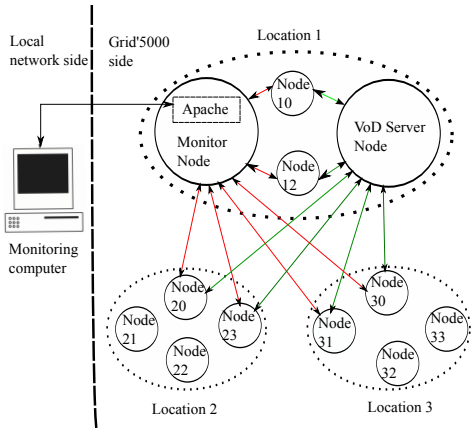

Fig. 4. Architecture and interactions between the nodes to replicate user behavior.

Each node is connected to the monitor node (red links) and to the VoD server (green link) as schematized in Figure 4. Let us remind that an user can be in any three S, I or R states and only solicits the VoD server when it is in infected state. Each time an user changes its state, it sends a message to the monitor to update its status. Moreover, when an user wants to infect another user, it first requests the monitor to choose randomly among susceptible users. Then the monitor node sends back a message to the chosen node to turn it infected. In our implementation we consider that an user goes back to the S state after leaving the R state with the possibility of getting contacted by the server again. We also implement an apache web server on the monitor node to visualize evolution of the workload in real-time (Figure 4).

B. Implementation Issues

The first issue we encounter, is to generate independent random variables, as required by the Markov Model. We know that the classical approach to generate random variables is to define an unique seed to ensure the independence of variables. However, since all nodes are launched at the same time, we cannot use the current time to define the seed. We overcome this by summing up the IP address to the current time for generating the seeds. The last operation is done to facilitate independent realizations in case we want to repeat the same experiments on the same nodes.
The second issue is to implement an efficient server to manage the significant amount of communication among hundreds of nodes. We use a multi-threaded server to handle this. Each time a node wants to communicate with the server, a new thread is created to process the request while the original thread holds ready to listen to any new communication. Use of these threads raise another problem regarding protection of shared variables from multiple accesses. To prevent this, we use mutual exclusion defining specific variables to manage access to these shared variables between threads.

III. Results

Figure 5 shows the workload (N_i) generated by a chosen set of parameters. Here, a buzz period materializes with the abrupt increase of N_i at time = 200 s. Owing to the memory effect (μ) this transitory overload has a more persistent effect on N_r (the number of past viewers) that undergoes a mean shift which lasts beyond the duration of the buzz epoch.

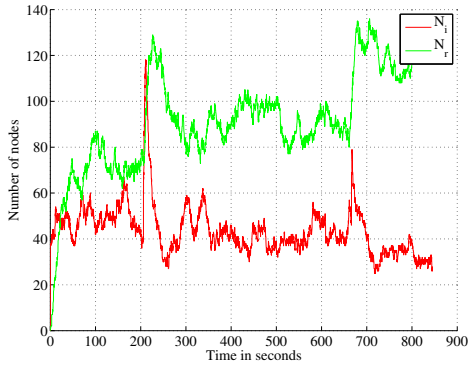

Fig. 5. Workload generated from the following parameters: $\beta_1 = 0.01$, $\beta_2 = 1$, $\gamma = 0.05$, $\mu = 0.033$, $l = 0.02$, $a_1 = 0.002$, $a_2 = 0.2$, total number of nodes = 240. N_i and N_r are number of current viewers and past viewers, respectively.

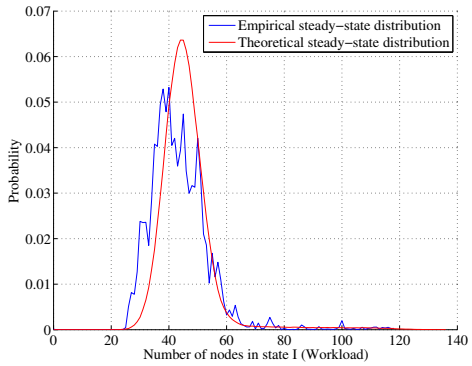

Fig. 6. Steady-state probability of the workload

In Figure 6 we compare the empirical steady-state probabilities computed from the generated trace to the corresponding theoretical values as given in [2]. In particular the empirical value of the mean workload is 44.08 compared to its theoretical value 49.35 (10.2% error). Given the limited duration of the experiment and possible uncertainties on networks, the theoretical and experimental results match satisfactorily, validating our implementation.

IV. Objective of the demonstration

In the demonstration we aim to show the effectiveness of our workload generator to emulate several realistic VoD traffic traces (having different workload profile) with different sets of parameters. The main asset of our approach lies in the combination of a versatile, plausible theoretical model with a fully controllable large-scale test-bed involving heterogeneous equipment and an advanced networking infrastructure. Figure 7 shows a snapshot from the monitoring computer displaying a typical real-time server workload.

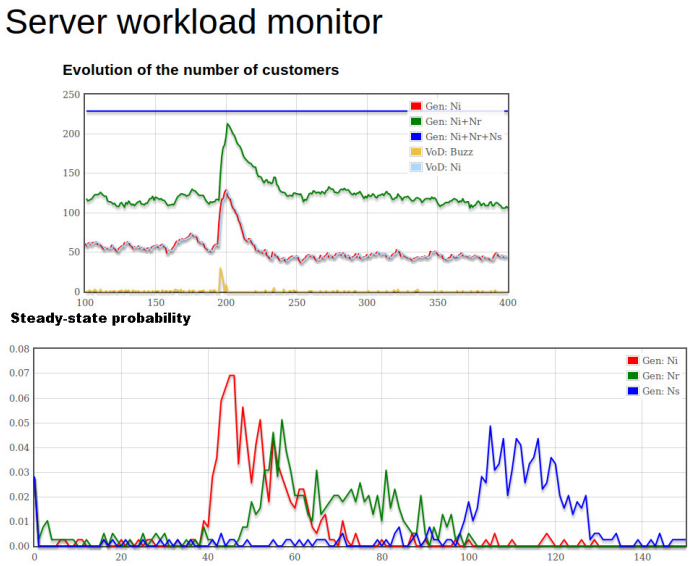

Fig. 7. Snap shot of the real-time server workload from the monitoring computer.

V. Conclusion

Objective of this demonstration is to emulate and validate a previously defined theoretical model to harness probabilistic methods for resource provisioning in the Clouds. We illustrate our purpose with a Video on Demand scenario, a characteristic service whose demand relies on information spreading. We implemented, on a real test-bed, a constructive model to capture the users' behavior, that satisfy some statistical properties, needed to devise resource management policies. We also demonstrated that the model, upon implementation in a real test-bed, satisfies the theoretical properties described in the model. Our future objective is to exploit the traces generated out of this model for framing resource management policies in a cloud network.

Acknowledgments

This work has been supported by the EU FP7 project SAIL.

References

[1] P. Gonçalves, S. Roy, T. Begin, and P. Loiseau, "Dynamic resource management in clouds: A probabilistic approach.," *IEICE Transactions on Communications, special session on Networking Technologies for Cloud Services*, vol. E95-B(08), August 2012.

[2] S. Roy, T. Begin, P. Loiseau, and P. Gonçalves, "Un modèle de trafic adapté à la volatilité de charge d'un service de vidéo à la demande: Identification, validation et application à la gestion dynamique de ressources," Inria research report, number 8072, INRIA, Lyon, France, September 2012, http://fr.arxiv.org/abs/1209.5158.

[3] Sandvine, "Sandvine's spring 2011 global internet phenomena report reveals new internet trends," May 2011, http://www.sandvine.com/news/pr_detail.asp?ID=312/.

[4] GRNET, "Video traces obtained from grnet," 2011, http://vod.grnet.gr/ accessed on July 2012.