Simulating User Interactions: A Model and Tool for Semi-realistic Load Testing of Social App Backend Web Services

Philipp Brune
University of Applied Sciences Neu-Ulm, Wileystraße 1, D-89231 Neu-Ulm, Germany

Keywords: Model-based Testing, Load Testing, User Simulation, Mobile App Development, Mobile Social Network Interaction, Web Services.

Abstract: Many mobile apps today support interactions between their users and/or the provider within the app. Therefore, these apps commonly call a web service backend system hosted by the app provider. For the implementation of such service backends, load tests are required to ensure their performance and scalability. However, existing tools like JMeter are not able to simulate "out of the box" a load distribution with the complex time evolution of heterogeneous, real and interacting users of a social app, which e.g. would be necessary to detect critical performance bottlenecks. Therefore, in this paper a probabilistic model for simulating interacting users of a social app is proposed and evaluated by implementing it in a prototype load testing tool and using it to test a backend of new real-world social app currently under development.

1 INTRODUCTION

An increasing number of online services nowadays offer features of social networks for their users, which enable their users to interact with each other (like maintaining personal profile, making friends etc.). Such online services range from multiplayer games to business-related communities, just to name a few. In addition, most of these services use mobile apps at least as one of their user frontends, very often also as the primary one. Therefore, the usage of social mobile apps is strongly increasing in recent years (Hsiao et al., 2016).

This requires the implementation and hosting of service backends for such kind of apps, which are typically implemented using RESTful web services. For the quality assurance of such service backends, load tests are required to ensure their performance and scalability. However, existing tools like JMeter[1] using rather static, predefined load profiles or purely random execution of http requests are not able to simulate a load distribution out of the box, which models the complex time evolution of heterogeneous, real and interacting users of an Online Social Network (OSN) app.

In particular, the type, time and sequence of user actions in OSN strongly depend on the current situation of the user, his or her intrinsic motivation to use the network, friends' activity, or external events (Xu et al., 2012), which could be modeled by user agents having an inner state (Bonabeau, 2002) and maintaining a user session (Shams et al., 2006). This might be required e.g. to properly size the runtime hosting environment and to detect critical performance bottlenecks. Therefore, realistic load testing of online applications has been an ongoing topic of research in web and mobile development for many years (Calzarossa and Tucci, 2002; Shams et al., 2006; Terevinto et al., 2016).

However, none of the existing approaches model all these relevant aspects of real users' behaviour. Therefore, in this paper a more holistic probabilistic model for simulating interacting users of a social app is proposed. The approach is evaluated by implementing it in a prototype load testing tool and using it to test a backend of a new real-world social app currently under development.

The rest of this paper is organized as follows: In section 2 the related work is analyzed in detail, section 3 describes the design of the proposed stochastic model. The proof-of-concept implementation of the prototype load testing tool is illustrated in section 4 and its evaluation in section 5. The current limitations of the approach are discussed in section 6 before we conclude with a summary of our findings.

[1] http://jmeter.apache.org

Brune, P.
Simulating User Interactions: A Model and Tool for Semi-realistic Load Testing of Social App Backend Web Services.
DOI: 10.5220/0006248202350242
In *Proceedings of the 13th International Conference on Web Information Systems and Technologies (WEBIST 2017)*, pages 235-242
ISBN: 978-989-758-246-2

2 RELATED WORK

For many years, tools and techniques for the performance evaluation and load testing of internet applications have been a topic of active and still ongoing research (Calzarossa and Tucci, 2002; Terevinto et al., 2016). Therefore, the synthetic generation of realistic load profiles for distributed applications is a crucial prerequisite. Various approaches to achieve this on the network layer of the communications stack have been proposed (Balachandran et al., 2002; Busari and Williamson, 2002; Weigle et al., 2006).

However, for typical distributed and web applications, the generation of realistic load profiles in general requires to model the user interactions with the system (Shams et al., 2006) including the need to maintain a user session, with "a session being a sequence of inter-dependent requests submitted by a single user" (Shams et al., 2006). Since each such session requires a user-specific state, the creation of realistic synthetic workloads on the application level requires the simulation of a large number of independent virtual "users" and their interactions with the system ("'behaviour") (Hlavacs et al., 2000; Shams et al., 2006). In general, agent-based modeling could be used to achieve this (Bonabeau, 2002).

Such user behaviour modeling has been used to create synthetic workloads for various types of distributed applications, ranging from multiplayer online games (Lehn et al., 2014) to traditional web applications (Hlavacs et al., 2000; Shams et al., 2006; Shyaamini and Senthilkumar, 2015) and, more recently, social media platforms (Xu et al., 2012; Terevinto et al., 2016). Different approaches exist to create the required large number of independent virtual user instances (Shams et al., 2006; Vögele et al., 2015; Terevinto et al., 2016). As a prerequisite for modeling and predicting workloads, also the real users' behavior of various applications has been analyzed (Yu et al., 2006; Maia et al., 2008; Benevenuto et al., 2009; Radinsky et al., 2012; Awad and Khalil, 2012). In addition, typical challenges regarding this kind of performance evaluation with synthetic workloads have been discussed (Hashemian et al., 2012).

In many cases, server-side components of distributed applications will be deployed in the cloud, typically using the Platform-as-a-Service (PaaS) model (Mell and Grance, 2011). Consequently, synthetic workload generation and performance modeling for cloud environments became a major research focus recently (Chen et al., 2010; Folkerts et al., 2012; Calheiros et al., 2013; Moreno et al., 2013; Chen et al., 2015; Heinrich et al., 2015; Magalhães et al., 2015; Sliwko and Getov, 2016; Gonçalves et al., 2016). However, only few authors consider modeling of PaaS environments specifically (Zhang et al., 2012). But since elasticity is a crucial feature of cloud environments, the simulation of workload variations over time has been considered (Albonico et al., 2016).

While approaches for synthetic workload generation for existing OSN applications have been proposed (Xu et al., 2012; Terevinto et al., 2016), no solution exists yet for performance testing new OSN applications still under development, for which the user roles and behavior are still unknown.

In this case, only stochastic or probabilistic models could be used to model the expected user behavior, based on general observations about OSN users' behavior (Maia et al., 2008; Benevenuto et al., 2009; Xu et al., 2012). In general, the activity of users in OSN has been found to depend on the factors: intrinsic interest, breaking news (i.e. external events) and the activity of their friends (Xu et al., 2012), which could be used to define a generic probabilistic model of users' behavior. In addition, users' activities will in general follow an overall periodic time dependency due to day and night, weekends and weekdays etc. (Radinsky et al., 2012).

Since no approach exists so far combining all these aspects, in this paper a model and software tool for the synthetic creation of workloads for OSN applications are proposed and evaluated, which combine the following features:

- Virtual users modeled by finite state machines with probabilistic transitions between the states, where the transitions refer to different user actions,
- Users' activity depends on their intrinsic interest, external events in their area and their friends' activity,
- Users are spatially distributed,
- External events are localized and have a certain spatial impact range.

3 STOCHASTIC MODEL OF USER INTERACTIONS WITH SOCIAL APPS

The proposed model is based on the simulation of actions of locally dispersed social app users. For this purpose, user objects are generated which are localized at an equally distributed random position in a fictitious quadratic "world" with a coordinate system of longitudinal and latitudinal coordinate values between 0 and 100 each. This allows the simulation of

effects of locality (like e.g. by external events with a limited regional impact).

During test execution, each user object is periodically called to execute actions and send requests to the backend web services of the app. In each of these cycles, every user object might perform such actions with a certain probability, which describes its activity.

In agreement with (Xu et al., 2012), this probability p_A at the time t is defined by

$$p_A(t) = \max(p_F(t) + p_I(t) + p_E(t), 1)\, m(t)\,, \quad (1)$$

where p_I denotes the the user's internal (intrinsic) activity, p_E the probability of using the OSN due to the presence of external events affecting the user's location, and p_F the average activity of the user's direct friends' network (with N being the number of the user's direct friends)

$$p_F(t) = \frac{1}{N} \sum_{i=1}^{N} p_{A,i}(t)\,. \quad (2)$$

In addition, $m(t)$ describes a time-dependent modulation of the resulting probability, depending on the time of the day (a fictive day lasts for a defined time span like e.g. 24 seconds) to simulate sleep, rest or working periods. This modulation has a daily periodicity.

External events appear and vanish in the virtual world regularly, being located at random center positions and decreasing in importance for users with increasing distance. Therefore, the impact of external events on using the OSN is modelled by the probability

$$p_E(t) = \sum_{i=1}^{N_E(t)} p_{E,i}(t) = \sum_{i=1}^{N_E(t)} e_i(t) \frac{a_i}{a_i + r_i}\,, \quad (3)$$

where $N_E(t)$ is the current number of external events in the virtual world, $e_i(t)$ the importancy of the event i, a_i its mean impact radius and r_i the distance between the event's center position and the user's location. The importancy $e_i(t)$ of an event is decreased every virtual hour by a defined value Δe_i to limit its impact in time. Upon creation of new events, their center positions and all described parameters are chosen as random values.

Based on this model, the calculation of the current activity of a user and the decision whether to perform an action are implemented by the following Java code fragment (the variables `lat` and `lng` denote latitudinal and longitudinal coordinates of a position in the virtual world):

```
protected boolean isActive(int hour) {
   double friendsActivity = 0;
   synchronized (friends) {
      if (friends.size() > 0) {
         for (GenericUser f : friends) {
            friendsActivity =
              friendsActivity
              + f.getCurrentActivity();
         }
         friendsActivity
           = friendsActivity /
              friends.size();
      }
   }
   double activity = this.activity
      + Event.getTotalImportance(
              this.lat, this.lng)
       + friendsActivity;
   if (activity > 1.0) {
      activity = 1.0;
   }
   if (rand.nextDouble() <
        (activity
         * activityPerHour[hour])) {
      return true;
   }
   return false;
}
```

If this method returns true for a user, one of the following actions is randomly selected and performed:

- Do the next action with respect to the social app (like e.g. login, logout, invite friends, post a comment etc.), depending on the current state of the user object. Usually this involves making a web service call to the app's backend server,
- create a new user at a random position, add it to the network, link with it as a friend, and afterwards perform the next action,
- do nothing, with a small probability for deleting the user from the system to model a user quitting the social network.

4 PROOF-OF-CONCEPT IMPLEMENTATION

The proposed model has been implemented using the Java programming language[2] within a proof-of-concept (PoC) load-testing tool. In figure 1 the class diagram of this load testing tool based on the described stochastic model is illustrated.

There, the class `GenericUser` describes the virtual user objects, which e.g. contain also the met-

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

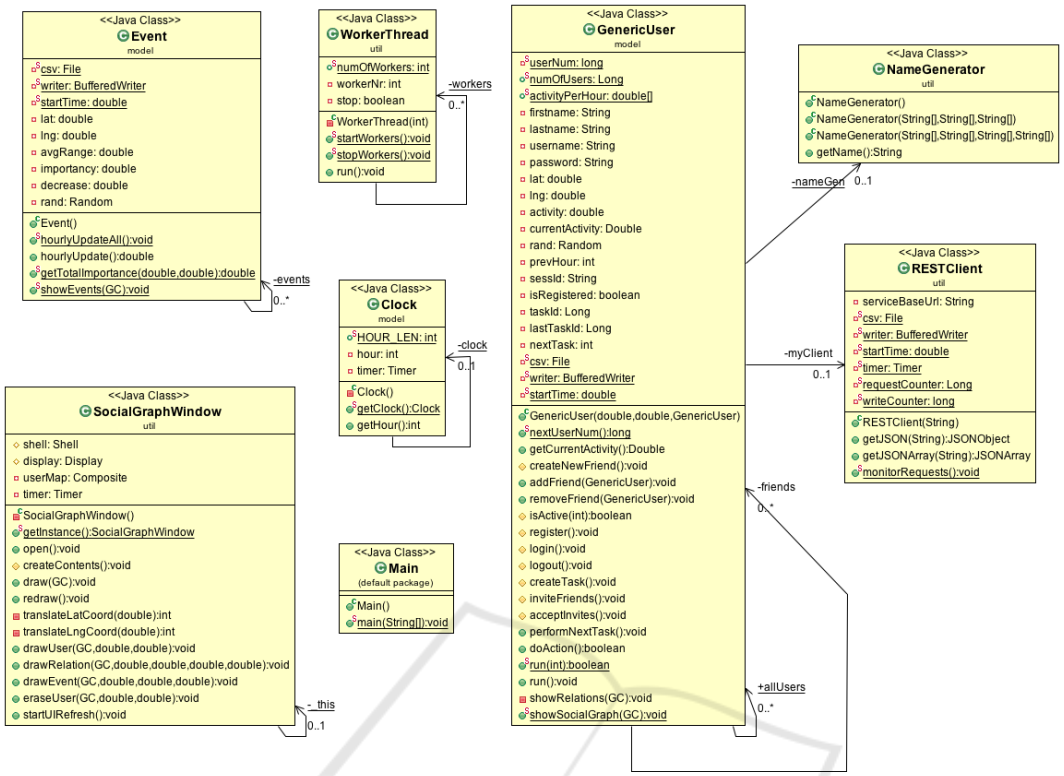

Figure 1: UML class diagram displaying the Java classes of the proposed load testing tool.

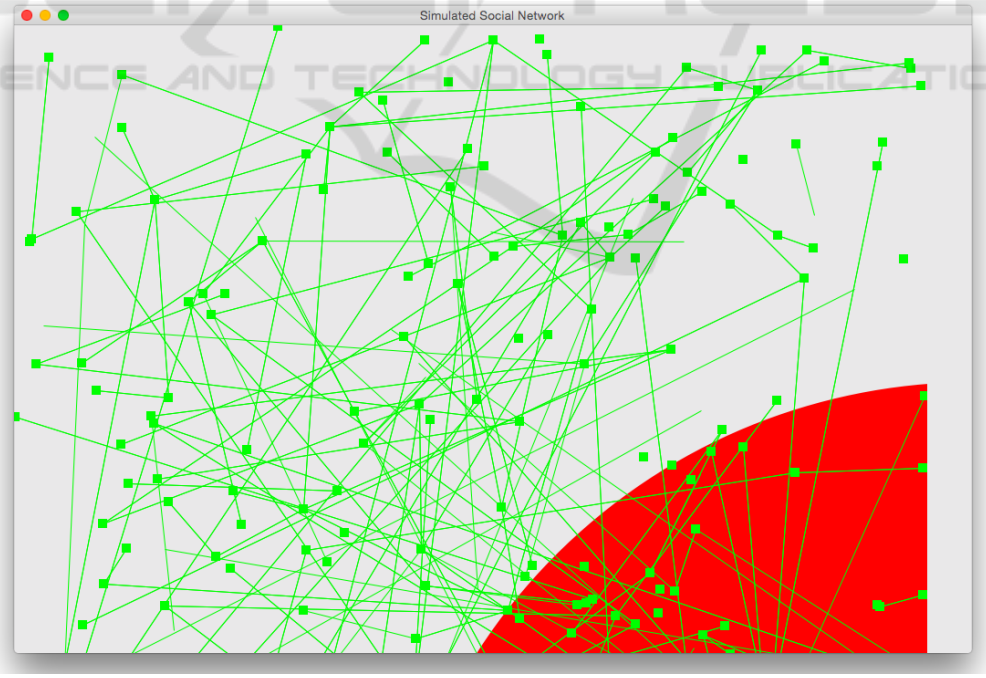

Figure 2: Runtime visualization of the simulated virtual users and their relations by the proposed load testing tool. The tool uses the Standard Widget Toolkit (SWT) to implement the window. Green squares denote the virtual users at their locations, green lines the friendship relations between these users (social graph), and red circles the events with their mean impact radius.

hod `isActive(int hour)` outlined above. The class `Event` models the external events taking place in the virtual world, and the class `Clock` sets the current time in the virtual world. These three classes belong to the package `model`, which contains the classes related to modeling the load.

The utility classes `WorkerThread` and `NameGenerator` describe the parallel worker threads to call to the `GenericUser` objects periodically and a string generator for unique artificial names and email addresses, respectively. `RESTClient` is a helper class for calling RESTful web services of a mobile app backend over the network, and `SocialGraphWindow` implements a Graphical User Interface (GUI) window to visualize at runtime the current state of the simulated social graph between the virtual users on the screen.

This GUI is implemented using the Standard Widget Toolkit (SWT) originating from the Eclipse project[3]. Figure 2 shows a screenshot of this window visualizing the simulated virtual users and their relations (green squares and lines) and external events with their mean impact radius (red circles).

The methods as `inviteFriend()`, `acceptInvites()`, `createNewFriend()`, etc. of `GenericUser` perform the typical actions of participants in a social network by calling the corresponding web services backend functions. Therefore, the helper class `RESTClient` is used. Due to this, it is comparably easy to adapt the described PoC implementation to test other OSN's web service backends by sub-classing `GenericUser` and overriding or extending these methods.

Currently, this PoC implementation is multi-threaded only and not distributable on multiple servers, since it uses two global, static arrays, `GenericUser.allUsers` and `Event.event`, representing the complete sets of all users and events in the system. These need to be shared between all threads. Therefore, it needs to be run within a single Java Virtual Machine (JVM) instance. Regarding the maximum number of simulatable users, it thus is only vertically scalable by using a larger symmetric multiprocessor (SMP) server.

5 EVALUATION

The proposed model and prototype tool were evaluated by testing a web service backend for a new social mobile app currently under development for a prospective internet start-up.

This service backend consists of RESTful web services implemented using Java Enterprise Edition (EE) JAX-RS API, Java Persistence API (JPA) for object-relational mapping, a PostgreSQL database and running on the Heroku Platform-as-a-Service (PaaS) cloud environment[4].

Figure 3 shows the results of a typical test run. Here, both diagrams display measured data for a time period of 350.000 milliseconds (corresponding to about 14 fictive days in the simulated world, with each being 24 seconds long).

The diagram at the top first shows the execution times (black line) of one typical social app web service request (namely: retrieve pending invitations to a group) measured on the server (including database access, without network latency, in milliseconds). Second, the cumulated intensity of the external events is shown (red line, integrated over the whole area of the virtual world). These intensity values are multiplied by 1000 for displaying them together with the execution times in one diagram.

The diagram below first shows the number of simulated users (black line), and second the periodic pattern of the activity filter values (red line). With the latter values the calculated activity of a user is finally multiplied to model periods of activity and inactivity (i.e. sleep) during a day. This corresponds to the statement `activity*activityPerHour[hour]` at the end of the `isActive(int hour)` method shown in section 3. The pattern repeats itself for every fictitious day (about 14 times). The intensity values lie in a range between 0 and 1 and are multiplied here by 1000 for displaying them in one diagram with the number of users.

The data indicates that the measured execution time for processing a service request on average rises with the number of users and positively correlates with the intensity of the external events, as one would expect. However, these execution times are impacted by multiple other factors (like e.g. the general randomness of the model, the internal JVM state, server load), so it could not be expected to observe a simple direct proportionality.

Therefore, the data is in line with the intended behaviour for the proposed load testing tool and supports its applicability in practice. In addition, the tool already helped to detect various design flaws of the service implementation under test (e.g. regarding the concurrent database connection handling and hidden dependencies between different service functions previously overlooked), so it proved to be useful also qualitatively.

[3]http://www.eclipse.org/swt

[4]http://www.heroku.com

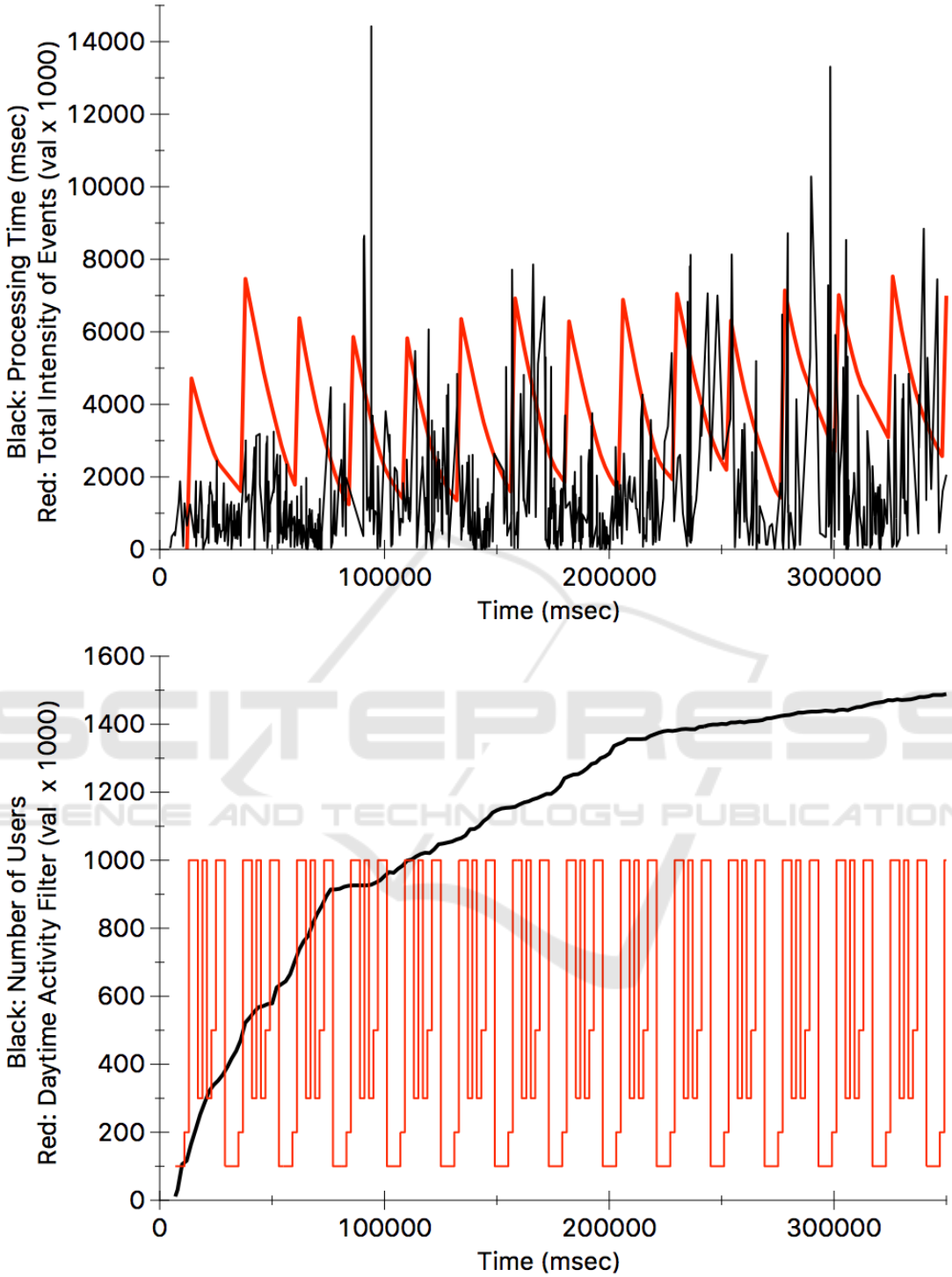

Figure 3: Measured results from running the prototype load test tool for the backend services of a new social mobile app currently under development. For this test, the backend services were running in the Heroku PaaS cloud and the test tool was run locally on a PC. The diagram at the top shows the cumulated intensity of the external events (red line, intgrated over the whole area and multiplied by 1000 for displaying purposes) and the execution times of a specific frequent service request measured inside the server (black line, in milliseconds) over time (in milliseconds). The diagram below shows the increasing number of simulated users (black line) and the overall activity filter factor for the time of the day (red line, values between 0 and 1, multiplied by 1000 for displaying purposes) over time (in milliseconds).

6 LIMITATIONS AND FURTHER RESEARCH

While these evaluation results in principle demonstrate its feasibility, some open issues remain to be addressed for evaluating the general applicability and benefits of the proposed approach in practice:

First, its adaptability to test different OSN backends needs to be demonstrated by evaluating the tool in various scenarios, in lab tests as well as in practice. As indicated in section 4, sub-classing of `GenericUser` provides a mean to adapt the present PoC implementation with reasonable effort to new OSN backends.

Second, as described in section 4, the current PoC implementation is limited to a single JVM instance and therefore is only vertically scalable. While this is sufficient to evaluate the proposed model and the basic feasibility of the approach, the creation of more realistic, spatially distributed load tests requires horizontal scalability using an implementation distributable over multiple servers. The presented implementation in principle could be extended to run on multiple server nodes in a network by implementing a synchronisation and replication mechanism between the nodes for the two global arrays described. E.g., a message passing communication model could be used for this. However, further research is needed to implement and evaluate this extension.

7 CONCLUSION

In conclusion, in this paper a probabilistic model for simulating interacting users of a social app has been proposed and evaluated by implementing it in a prototype load testing tool. The proposed approach is capable not only of simulating the users' activity depending on their interest, external events in their area and their friends' activity, but also takes into account the spatial distribution of users and external events, which are localized and have a certain spatial impact range within a virtual world.

While the evaluation is still preliminary, the results obtained so far are promising. Already with the prototype tool some serious design flaws of a service backend of a new, real-world social app currently under development were detected. However, further research is needed to continue the evaluation of the proposed approach in lab and field tests with respect to its possible applications, ease of use, performance and adaptability.

REFERENCES

Albonico, M., Mottu, J.-M., and Sunyé, G. (2016). Automated workload generation for testing elastic web applications.

Awad, M. A. and Khalil, I. (2012). Prediction of user's web-browsing behavior: Application of markov model. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(4):1131–1142.

Balachandran, A., Voelker, G. M., Bahl, P., and Rangan, P. V. (2002). Characterizing user behavior and network performance in a public wireless lan. In *Proceedings of the 2002 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '02, pages 195–205, New York, NY, USA. ACM.

Benevenuto, F., Rodrigues, T., Cha, M., and Almeida, V. (2009). Characterizing user behavior in online social networks. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '09, pages 49–62, New York, NY, USA. ACM.

Bonabeau, E. (2002). Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences*, 99(suppl 3):7280–7287.

Busari, M. and Williamson, C. (2002). Prowgen: a synthetic workload generation tool for simulation evaluation of web proxy caches. *Computer Networks*, 38(6):779–794.

Calheiros, R. N., Netto, M. A., De Rose, C. A., and Buyya, R. (2013). Emusim: an integrated emulation and simulation environment for modeling, evaluation, and validation of performance of cloud computing applications. *Software: Practice and Experience*, 43(5):595–612.

Calzarossa, M. C. and Tucci, S. (2002). *Performance Evaluation of Complex Systems: Techniques and Tools: Performance 2002. Tutorial Lectures*, volume 2459. Springer Science & Business Media.

Chen, F., Grundy, J., Schneider, J.-G., Yang, Y., and He, Q. (2015). Stresscloud: a tool for analysing performance and energy consumption of cloud applications. In *Proceedings of the 37th International Conference on Software Engineering-Volume 2*, pages 721–724. IEEE Press.

Chen, Y., Ganapathi, A. S., Griffith, R., and Katz, R. H. (2010). Towards understanding cloud performance tradeoffs using statistical workload analysis and replay. *University of California at Berkeley, Technical Report No. UCB/EECS-2010-81*.

Folkerts, E., Alexandrov, A., Sachs, K., Iosup, A., Markl, V., and Tosun, C. (2012). Benchmarking in the cloud: What it should, can, and cannot be. In *Technology Conference on Performance Evaluation and Benchmarking*, pages 173–188. Springer.

Gonçalves, G. D., Drago, I., Vieira, A. B., da Silva, A. P. C., Almeida, J. M., and Mellia, M. (2016). Workload models and performance evaluation of cloud storage services. *Computer Networks*.

Hashemian, R., Krishnamurthy, D., and Arlitt, M. (2012). Web workload generation challenges–an empirical investigation. *Software: Practice and Experience*, 42(5):629–647.

Heinrich, R., Jung, R., Schmieders, E., Metzger, A., Hasselbring, W., Reussner, R., and Pohl, K. (2015). Architectural run-time models for operator-in-the-loop adaptation of cloud applications.

Hlavacs, H., Hotop, E., and Kotsis, G. (2000). Workload generation by modeling user behavior. In *In Proceedings OPNETWORKS 2000*.

Hsiao, C.-H., Chang, J.-J., and Tang, K.-Y. (2016). Exploring the influential factors in continuance usage of mobile social apps: Satisfaction, habit, and customer value perspectives. *Telematics and Informatics*, 33(2):342 – 355.

Lehn, M., Triebel, T., Rehner, R., Guthier, B., Kopf, S., Buchmann, A., and Effelsberg, W. (2014). On synthetic workloads for multiplayer online games: a methodology for generating representative shooter game workloads. *Multimedia Systems*, 20(5):609–620.

Magalhães, D., Calheiros, R. N., Buyya, R., and Gomes, D. G. (2015). Workload modeling for resource usage analysis and simulation in cloud computing. *Computers & Electrical Engineering*, 47:69–81.

Maia, M., Almeida, J., and Almeida, V. (2008). Identifying user behavior in online social networks. In *Proceedings of the 1st workshop on Social network systems*, pages 1–6. ACM.

Mell, P. and Grance, T. (2011). The nist definition of cloud computing.

Moreno, I. S., Garraghan, P., Townend, P., and Xu, J. (2013). An approach for characterizing workloads in google cloud to derive realistic resource utilization models. In *Service Oriented System Engineering (SOSE), 2013 IEEE 7th International Symposium on*, pages 49–60. IEEE.

Radinsky, K., Svore, K., Dumais, S., Teevan, J., Bocharov, A., and Horvitz, E. (2012). Modeling and predicting behavioral dynamics on the web. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 599–608, New York, NY, USA. ACM.

Shams, M., Krishnamurthy, D., and Far, B. (2006). A model-based approach for testing the performance of web applications. In *Proceedings of the 3rd international workshop on Software quality assurance*, pages 54–61. ACM.

Shyaamini, M. B. and Senthilkumar, M. (2015). A novel approach for performance testing on web application services. *International Journal of Applied Engineering Research*, 10(18):38679–38683.

Sliwko, L. and Getov, V. (2016). Agocs–accurate google cloud simulator framework.

Terevinto, P. N., Pont, A., Gil, J. A., and Domenech, J. (2016). A flexible workload model based on roles of interactive users in social networks. In *2016 IFIP Networking Conference (IFIP Networking) and Workshops*, pages 524–529.

Vögele, C., Robert, H., Robert, H., van Hoorn, A., and Krcmar, H. (2015). Modeling complex user behavior with the palladio component model.

Weigle, M. C., Adurthi, P., Hernández-Campos, F., Jeffay, K., and Smith, F. D. (2006). Tmix: a tool for generating realistic tcp application workloads in ns-2. *ACM SIGCOMM Computer Communication Review*, 36(3):65–76.

Xu, Z., Zhang, Y., Wu, Y., and Yang, Q. (2012). Modeling user posting behavior on social media. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 545–554, New York, NY, USA. ACM.

Yu, H., Zheng, D., Zhao, B. Y., and Zheng, W. (2006). Understanding user behavior in large-scale video-on-demand systems. In *ACM SIGOPS Operating Systems Review*, volume 40, pages 333–344. ACM.

Zhang, W., Huang, X., Chen, N., Wang, W., and Zhong, H. (2012). Paas-oriented performance modeling for cloud computing. In *2012 IEEE 36th Annual Computer Software and Applications Conference*, pages 395–404. IEEE.