Models for Evaluating Throughput

Netanel Zakay Dror G. Feitelson
Department of Computer Science
The Hebrew University, 91904 Jerusalem, Israel

***Abstract*—Analysts are interested in two categories of performance metrics: those concerned with time (response or wait time), and those concerned with rates (throughput or utilization, which reflect productivity and how well resources are used). In principle, these two categories are independent of each other, and both should be evaluated. But a common mistake is to "measure" the throughput in open-system evaluations, where the throughput is actually dictated directly by the workload. In order to evaluate throughput, the system model *must* include a feedback loop which modulates the workload being processed. The common solution is to create a pure closed system with a fixed number of users, who submit jobs in a loop. However, such behavior is often unrealistic. We review and analyze two alternative models that provide the required feedback by combining open and closed components: the *mixed model* which includes two such job classes, and the *re-open model* in which open user arrivals are combined with performance-dependent closed repetition of jobs by these users. These models allow evaluations of the trade-off between response time and throughput, including the throughput as it is observed by each user.**

***Index Terms*—throughput; utilization; mixed open-closed model; re-open model.**

I. Introduction

Evaluating the performance of a system is a major part of system design. Reliable evaluations of a proposed system are expected to lead to better designs and reduced expenses, by considering multiple options, evaluating them, and choosing the best. Therefore, when a new system design is proposed or when we want to improve a current system, it is common to evaluate it before implementing it.

There are two main categories of system performance metrics. The first category includes the response time and the wait time. These capture the delays that a job suffers in the system. The assumption is that users are more satisfied with shorter delays. The second category includes the utilization and the throughput. These capture how much of the system's resources are used and the rate at which the system serves requests. The assumption is that higher utilization and serving more requests are better. In many systems a tradeoff is involved: higher throughputs lead to higher response times.

Importantly, the two categories are not just different manifestations of the same effects. They can change independently of each other, so both should be evaluated. But commonly used performance evaluation approaches such as trace-based simulation and open-system queuing analysis evaluate only the wait time and the response time. Measuring the throughput or utilization using these methodologies can only serve as a sanity check, because the throughput and utilization (which are linearly related in expectation in this case) are completely determined by the workload. In other words, these common evaluation methodologies effectively treat throughput and utilization as an input and not as part of the evaluation.

The classic approach to evaluate throughput is by using a closed model. But a pure closed model with a fixed number of users is unrealistic for many types of system. In many cases a more realistic workload scenario combines the closed behavior with an open behavior. For example, users may arrive randomly as in an open system, but then they may execute a workflow of multiple jobs that depend on each other as in a closed model. Trace-based simulations that preserve all the jobs' properties, including their arrival times, actually destroys the logic of the user's workflow, specifically the dependencies and think times between successive jobs. An better alternative is therefore to preserve the dependencies, and adjust the arrival times [14]. But still, if all the jobs in the workflow are eventually performed, the total amount of work and hence the system throughput are the same as in the original trace. The only thing that may change is the *per-user throughput*, because the workflows of individual users may be spread out differently. This motivates adding per-user throughput to the set of metrics that should be evaluated.

But in real life there do exist situations where the total throughput is indeed affected. To capture this one needs a system model that includes an explicit effect on the number of jobs processed by the system. This can take one of two forms. The first model is systems that use admission controls to throttle their users or just drop superfluous jobs (e.g. [2], [12], [1]). A good metric of performance in this case is the number of jobs that are rejected. The second model is users who change their behavior in response to system performance. For example, it is easy to envision users who become frustrated with poor performance and reduce their activity. This is the type of models we address here.

Importantly, such models allow analysts to assess the impact of system designs on throughput even when the system does not address this explicitly (e.g. it does not employ admission controls). In particular, they facilitate an analysis of the tradeoff between throughput and response time, and the identification of situations where higher response times are actually beneficial because they correlate with higher throughput and utilization. Moreover, such models may capture negative feedback effects as when users back off from an overloaded system and thereby prevent its saturation. Using an oblivious model — as is commonly done in trace-based simulations and queueing analyses — would miss such effects and lead to overly pessimistic results [9].

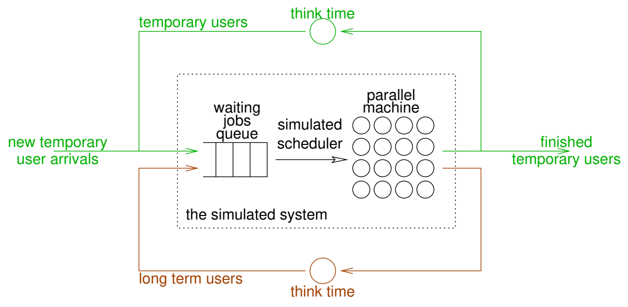

Fig. 1: *Flow of TBOUS.*

II. TBUOS and Semi-Open Simulation

In a previous work we proposed the Trace-Based User-Oriented Simulation (TBUOS), which is a semi-open simulation that includes dynamic user activity and internal feedback from the system to the users [15]. In this simulation, we divide the users into two groups (Figure 1). One is temporary users that arrive to the system at a fixed rate, and are active for only a limited time. The other is long term users who are always active. This is modeled in simulations by sending their first traced job again after the termination of their last traced job.

We found that the overall throughput of a TBOUS simulation can differ from that of a conventional simulation based on the same trace, as a result of subtle interactions between the users. The temporary users each have a fixed number of jobs, so in the long run their contribution is fixed. But the long term users are coupled to them because they contend for the same processing resources. So if the load caused by temporary users affects the rate at which long-term users circulate, the overall throughput is affected. The goal of the present paper is to further investigate this effect, and compare it with another model where all users are temporary but their number of jobs is performance-dependent.

More formally, the two models we will be investigating are as follows:

- **The mixed model**. This model includes an open workload class and a closed workload class. It is an abstraction of the workload used in the TBUOS simulations.
  1) Open users submit a single job to the system and leave. This is like in a conventional open system. Arrivals of these users are not affected by the system state, so the load they impose on the system is also not affected by the system state. In other words, they have a constant contribution to the system throughput.
  2) Closed users who submit a job, wait for it to terminate, think, submit another job, and so on indefinitely. This is just like a conventional closed system, and arrivals are naturally affected by the system state.

  We assume that the properties of the jobs, such as the distribution of run-times, are the same for both classes. The interesting issue is the interaction between the two classes. As in TBOUS, the open class affects the performance of the closed class, and this effect can lead to changes in the system throughput.

  This model represents a system that has both permanent users and temporary users. For example, a cloud service that has users that paid to host a persistent service and other users that try the system for a limited time only and then leave. This is highly relevant today due to the growing popularity of cloud systems.
- **The re-open model**. This is an open model with repeated submittal of jobs: users arrive at a given arrival rate as in a conventional open model, but once they arrive they may submit several jobs one after the other with think times in between. As we show below, if the total number of jobs (or the probability to submit additional jobs) is fixed, the system cannot affect the throughput. But if the probability to submit additional jobs depends on system performance (and specifically on the response time) then throughput is indeed affected.

  This model may be suitable for a web server or similar systems. When users surf to a site they usually send some number of requests. However, if the performance is poor, a user may get discouraged and leave the site. On the other hand, if the server is highly responsive, the users may extend their activity and send more requests.

These models are shown graphically in Figure 2. Note that in the mixed model the two user classes are distinct, and the only interaction between them is that both use the same resources on the server. In the re-open model, on the other hand, there is only one user class, and each user either submits additional jobs (closed behavior) or departs (open behavior) with some probability.

An important issue in both models with whether the workload is assumed to be interactive or batch. In batch workloads closed jobs are submitted one after the other with no intervening think time. As a result the utilization is always 100% and the throughput equals the system's capacity (namely the maximal number of jobs that the system can serve in a unit of time). In this case the only metric that can change is the throughput observed by each user, because delays cause the users to execute the same jobs over longer periods of time.

In interactive workloads the think time throttles the rate of submitting jobs, and therefore changes to the scheduler can lead to changes in throughput. This can happen, for example, by delaying interfering jobs to non-prime time [3]. In our models we focus on interactive workloads.

III. The Mixed Model

The mixed model in and of itself is not new, and has been used to model the combination of interactive and batch work for example (see [5, sect 7.4.3] and [6, sect 13.7]). The idea there is that the batch jobs constitute a closed system, with a new one starting immediately upon the termination of a previous one. Previous works described the model briefly and how to calculate the performance metrics. To this we add the motivation of affecting throughput, and present graphs that

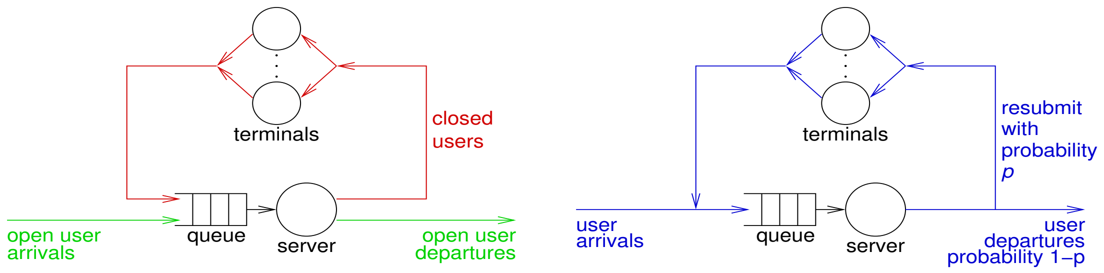

Fig. 2: *The mixed model (left) and the re-open model (right).*

show the relation between the inputs and the performance metrics. Also, we analyze this model using operational analysis, to show both intuitively and mathematically these relations between the performance metrics.

The Mixed Model is an abstraction of the model we used in simulations in the context of suggesting the use of workload resampling from recorded logs and TBOUS [13], [15]. As explained above, the two workload classes were long-term users who are active throughout and behave like a closed model, and temporary users who are active only for a limited duration, thus behaving like an open model (or rather, like a re-open model) in that their arrivals — and consequently also all the load they impose — is uncorrelated with system state.

A. *Model and Dynamics*

Assume a single server system, where arriving jobs queue, receive service, and depart. The parameters of the model are as follows:

open part	λ	arrival rate
closed part	N	number of users
	Z	think time
system params	S	service demand
	μ	service rate ($\frac{1}{E[S]}$)

This model assumes that all the jobs are similar, meaning that they have the same service demand S. Also, the scheduler does not differentiate between jobs submitted by the open users and the closed users.

Given the mixed workload, the system dynamics will evolve as follows. The open component of the workload is oblivious to the system state. It therefore imposes a fixed load of $\lambda E[S]$ per job. This has to be less than the system capacity, implying the common requirement $\lambda < \mu$.

Once the capacity taken up by the open component is acknowledged, the remaining capacity is $(\mu - \lambda)E[S]$. The closed part adjusts to fit in this left over capacity. The mechanism that affects this adjustment is the response time. The lower the system-wide response time, the sooner the closed users submit additional jobs, thereby increasing the load. But if the load is too high the response time will grow, thereby delaying the closed users, and subsequently delaying the submittal of additional jobs and reducing the load. This is a stabilizing negative feedback effect on the throughput X.

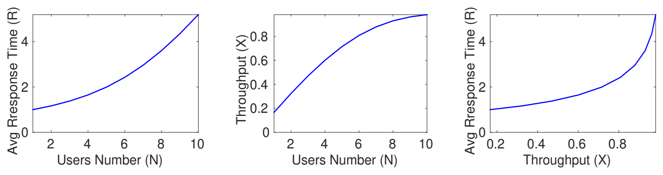

Fig. 3: *The average response time and throughput of the closed submodel (MVA results) with $\mu = 1.0$ and $Z = 5.0$. The first two graphs show how the number of users N affects the average response time and the throughput. The third graph combines these results to show the average response time as a function of the throughput (or equivalently, the utilization).*

B. *Common Evaluation*

As noted above the mixed model is well-known. For completeness, we provide a short review of the common approach to calculating the response times R_c and R_o (of closed and open jobs, respectively; they are not identical because open jobs arrive randomly, whereas closed job arrivals are correlated with system state, but the difference becomes negligible for large N).

1) The open part is oblivious to the system. Its utilization is λS and the throughput is λ.
2) The effect of the open part on the closed part is modeled by extending the service time to $S_c = \frac{S}{1-S\lambda}$. The closed part is then solved using the iterative MVA algorithm.
3) The average queue length of closed customers is added to the resident users of the open model to calculate R_o.

While this method calculates the performance metrics for the system, it doesn't provide the relations between the different metrics nor an intuitive explanation of the system's behavior. To provide this, we present some results in Figure 3. When the number of closed users increases both the response time and the throughput increase, but with different profiles: asymptotically the response time grows linearly, and throughput saturates at system capacity. The relation between them is highly non-linear, and similar to how response time depends on utilization in open systems. To understand this better, we use operational laws.

C. *Operational Analysis*

First, we observe that the system can operate in either of two phases.

- Full utilization phase — in this phase the closed component uses all the remaining capacity.
- Partial utilization phase — in this case the closed component does not use up all the remaining capacity, because the combination of the number of users and the think time does not allow the submittal of sufficient jobs.

Let's start by analyzing the full utilization phase. The fact that the system is fully utilized means that the throughput equals to the system capacity and therefore

$$X = \mu$$

The throughput is the sum of the open throughput λ and the closed throughput $\frac{N}{R_c+Z}$ (from the interactive response time law). Using the first equation we conclude that

$$\mu = \lambda + \frac{N}{R_c + Z}$$

From this we can extract R_c

$$R_c = \frac{N}{\mu - \lambda} - Z$$

Thereby characterizing the system performance using operational laws that do not require assumptions about distributions. This shows the intuitive result that (asymptotically) adding closed users leads to a linear increase in response time because they just pile up in the queue.

The partial utilization phase implies that

$$\mu > X = \lambda + \frac{N}{R_c + Z}$$

and therefore

$$R_c > \frac{N}{\mu - \lambda} - Z$$

In other words, the response time R_c is large enough to throttle the closed users and prevent them from creating additional load. We then have two unknowns: R_c and X, with the relationship

$$R_c = \frac{N}{X - \lambda} - Z$$

While this doesn't allow us to solve for R_c and X, it provides the inverse relationship between them for given N and Z. But if N grows X grows too, giving the result in Figure 3.

IV. The Re-Open Model

The basic elements of the re-open model were introduced by Schroeder et al. under the name "partly-open" [8] and used by others [7], [4]. Specifically, this combined open and closed elements by mandating that users submit additional jobs with a probability p. However, *this alone does not affect throughput*. To affect throughput we add the new condition that p be dependent on the system state.

The parameters of the model are as follows:

user params	λ	arrival rate
	p	probability to submit another job
	Z	think time between submitted jobs
system params	S	service demand
	μ	service rate ($\frac{1}{E[S]}$)

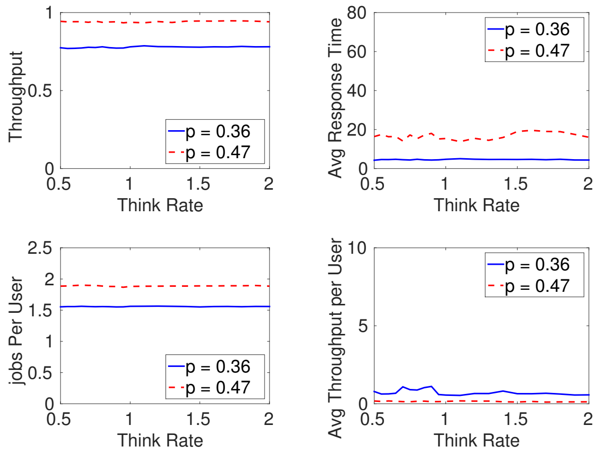

Fig. 4: *Simulating the re-open model for 10,000 users to check the impact of the "think rate"* $\frac{1}{Z}$ *on performance. Simulations use* $\mu = 1.0$ *and* $\lambda = 0.5$*, with* $p = 0.36$ *(solid) and* $p = 0.47$ *(dashed). These* p *values produce high enough load to be interesting, while the system is still stable.*

In the following subsections we analyze this model under different assumptions. To examine this model's characteristics, we created a simulation that simulates the re-open model with a basic FCFS scheduling policy.

A. Constant p Value

When p is constant, this model is a generalization of both the open and closed conventional models. A conventional open model is obtained when $p = 0$, and a conventional closed model is obtained when $p = 1$ and $\lambda = 0$.

If $0 < p < 1$ then each user submits a number of jobs and then leaves. As defined above with constant p the throughput X is fixed by the model, and it is essentially like an open system: each user submits $1+p+p^2+p^3+\ldots = \frac{1}{1-p}$ jobs in expectation, so the throughput is $X = \frac{\lambda}{1-p}$. In such a model the stability constraint is

$$\lambda \leq (1-p)\mu$$

This immediately leads to a bound on p, namely $p \leq 1 - \frac{\lambda}{\mu}$.

To evaluate this behavior, we modeled fixed p in simulations. All the distributions including the think times, interarrival times, and service times are exponential. Figure 4 shows that the think time doesn't have an impact on any of the results, at least when the system is not too close to saturation. Figure 5 shows the impact of p. While throughput, utilization, and jobs per user grow moderately with p, the queue length and wait time grow much more precipitously, but only upon approaching saturation. And note that as the overall throughput increases, the throughput as observed by each user drops.

Finally, Figure 6 shows the relation between the response time, the throughput, and the throughput per user. Note that the throughput per user decreases dramatically for higher response times, because the users are active for much longer, but send the same number of jobs.

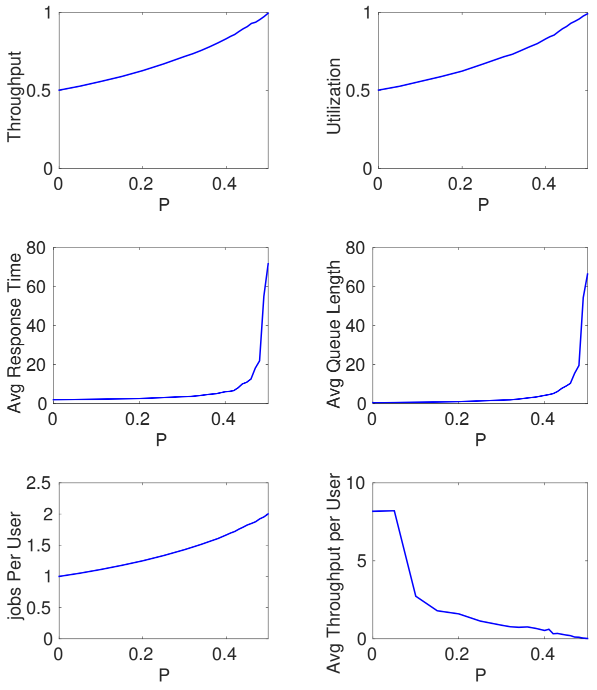

Fig. 5: *Simulating the re-open model for 10,000 users using* $\mu = 1.0$, $\lambda = 0.5$, *and* $Z = 1$ *in order to check the impact of* p *on performance.*

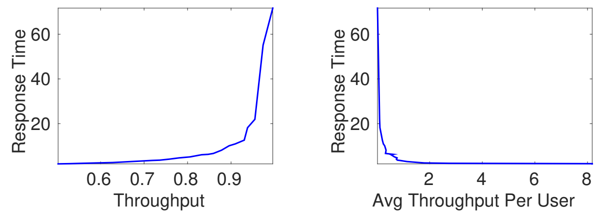

Fig. 6: *tradeoff between* R *and* X *in simulations of the re-open model.*

B. *p as a Function of the Response Time*

A more realistic model is to assume that the probability to send an additional job is not fixed, but rather depends on the performance of the system. High speed response may cause a user to extend his session with the system, while slow responses may cause him to leave the system. Therefore, a few works suggested to take the response time into account when calculating the probability of a user to send an additional job [10], [11].

This is an interesting model, because it means that the throughput is dynamic and depends on the performance of the system. Short response times will lead to higher throughput. But higher throughput will lead to more contention and thus to higher response times, and hence to reduced throughput. Therefore, the throughput and the response times balance each other, and again we have a stabilizing negative feedback effect.

To formalize the model we need to decide how p depends on the response time of previous jobs. A reasonable assumption

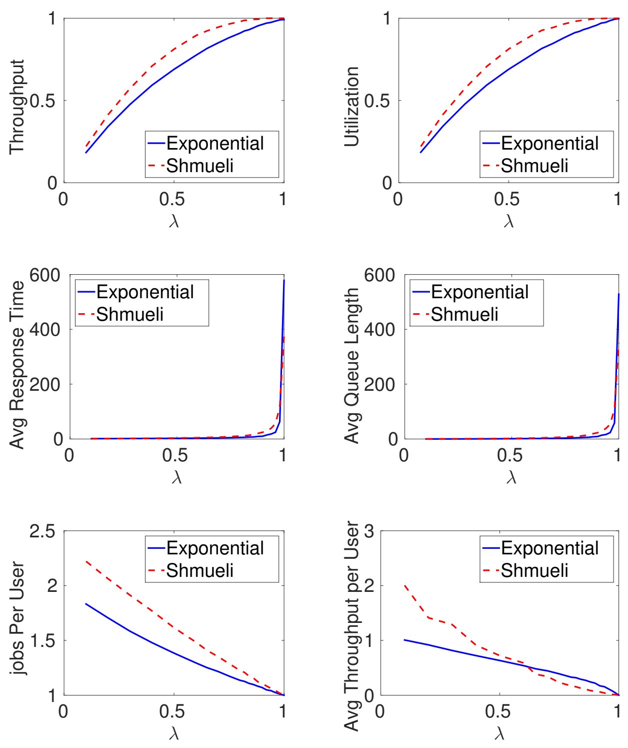

Fig. 7: *Simulating the re-open model for 10,000 users with* $\mu = 1.0$ *and* $Z = 1$ *where* p *is exponentially decreasing with* r *(solid) and where* p *is set according to Shmueli's formula (dashed).*

is that p is a monotonically decreasing function of r, where r is the response time of the last job of this user. We modeled two different functions for p:

- Exponentially decreasing function. This means that the probability to submit another job drops off exponentially with the response time of the previous job. In other words, $p = e^{-r}$. Therefore p starts from one for zero response time and decreases exponentially to 0 with longer response times.
- Shmueli's model. Shmueli and Feitelson analyzed the probability of a user to continue a session for several logs in the Parallel Workloads Archive [10], [11]. They discovered that the relationship is a hyperbola

$$p = \frac{0.8}{0.05 \cdot r + 1}$$

where r is in minutes. The average running time of jobs was from a couple of minutes to perhaps 10 minutes depending on the log. In our model, we use $S = 1$ as the unit of time. Therefore we used the same formula, but used $10r$ instead of r.

We simulated these two approaches. Figure 7 shows the resulting metrics for different λ. Note that the throughput and utilization are larger than λ due to the job repetitions. As λ increases, the throughput and utilization converge to 1. Thus when more users arrive, there are less resources available for

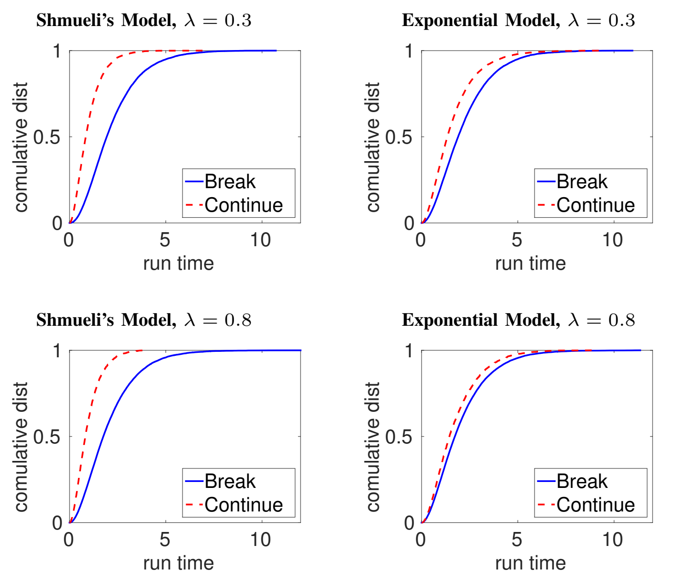

Fig. 8: *Runtime distributions of jobs after which the decision was to submit another job (continue), and jobs where the decision was to quit (break). Results for simulation of 10,000 users with* $\mu = 1.0$ *and* $Z = 1$.

running repeated jobs. Therefore response times grow and p drops. Consequently users submit fewer jobs, and the system does not saturate. Also, the throughput per user drops, but much more moderately than in the constant p model. And again, queue length and wait time shoot up only when the system is close to saturation.

Both approaches have different characteristics than the constant p model due to the fact that a higher number of users (higher λ) leads each user to send fewer jobs. Comparing between the two approaches, Shmueli's approach starts with a higher number of jobs per user, and drops nearly linearly to 1, while in the exponential approach the slope become smaller for higher λ. As a result, the throughput when using Shmueli's approach converges to 1 for lower λ and the average wait time is longer.

An interesting new metric we introduce is the throughput per user. This is the quotient of the number of jobs that a user submits divided by the total residence time in the system. In Shmueli's approach it starts higher, which means that users take advantage of the empty system to send more jobs, but they keep sending a high number of jobs when the system is loaded, and this leads to delays and subsequently to low throughput per user when λ is high.

C. Runtime Distribution Bias

An interesting artifact of the re-open model is the possible creation of bias in the distribution of job runtimes for each user. When p depends on the response time of the previous job, this reflects the confluence of two factors: the length of the job itself, and how long it had to wait in the queue. So if the job was short there is an increased probability to continue with additional jobs, and if the job was long this probability is reduced — regardless of the performance of the system. As a result the sequence of jobs executed by a certain user may tend to include several short jobs and only one long job, the last one. This may affect the throughput per user metric.

Evidence for this effect is shown in Figure 8. The distribution of runtimes of jobs that were the last one for a user (meaning that the probabilistic decision was not to submit additional jobs) tends to have longer times than the distribution of runtimes of jobs that were not the last. The effect is stronger with the Shmueli model, and weaker when λ is high, because then each user submits fewer jobs. More work is needed to decide if this is a problem or perhaps it actually reflects an effect that exists in reality.

V. Conclusions and Future Work

In conventional trace-based simulations and open-system analyses the throughput is given, and only the response time can be evaluated. In addition there is no feedback and the system may saturate as users continue to submit jobs.

We considered two models that include realistic negative feedback effects and allow the tradeoff between response time and throughput to be explored. The mixed model includes closed users whose throughput is affected by contention from open users. In the re-open model users submit additional jobs depending on their response time. In either case, the total throughput converges to the system capacity as more users are added. Therefore a more interesting metric is the throughput per user.

Use of such models is mandatory if one wishes to assess the effect of a scheduling scheme on throughput. This is an important performance metric for schedulers that attempt to prioritize different users or affect their productivity and behavior [11], [15]. Preferring one model over the other depends on the type of evaluated system and the users' behavior in the environment.

Our next goal is to continue developing these models and make them more realistic, similarly to the TBOUS simulation [15]. One useful expansion is to combine them by adding repeated jobs based on performance to the mixed model. In effect, this creates a mixture of closed and re-open instead of closed and open.

Another interesting issue is the think time (Z). In both the mixed model and the re-open model results above, the think time was sampled from an exponential distribution with parameters given as an input of the model. However, in reality, the think time might depend on various factors. For example, if the response time was short, the user might still be engaged with his session and send the next job with a short delay. For longer response times, the user might break his session and send the next job later in a new session (e.g. after taking a coffee break) or even the next day (if the job finished late at night). This suggests that more elaborate user behavior models are needed [14].

Acknowledgments

This work was supported by the Ministry of Science and Technology, Israel.

References

[1] T. Brecht, D. Pariag, and L. Gammo, "*accept()able strategies for improving web server performance*". In *USENIX Ann. Tech. Conf.*, pp. 227–240, Jun 2004.

[2] L. Cherkasova and P. Phaal, "*Session-based admission control: A mechanism for peak load management of commercial web sites*". *IEEE Trans. Comput.* **51**(**6**), pp. 669–685, Jun 2002, DOI: 10.1109/TC.2002.1009151.

[3] D. G. Feitelson and E. Shmueli, "*A case for conservative workload modeling: Parallel job scheduling with daily cycles of activity*". In 17th *Modeling, Anal. & Simulation of Comput. & Telecomm. Syst.*, Sep 2009, DOI: 10.1109/MASCOT.2009.5366139.

[4] R. Hashemian, D. Krishnamurthy, and M. Arlitt, "*Web workload generation challenges – an empirical investigation*". *Softw. — Pract. & Exp.* **42**(**5**), pp. 629–647, May 2012, DOI: 10.1002/spe.1093.

[5] E. D. Lazowska, J. Zahorjan, G. S. Graham, and K. C. Sevcik, *Quantitative System Performance: Computer System Analysis Using Queueing Network Models*. Prentice-Hall, Inc., 1984.

[6] D. A. Menascé, V. A. F. Almeida, and L. W. Dowdy, *Performance by Design: Computer Capacity Planning by Example*. Prentice Hall, 2004.

[7] D. Pariag, T. Brecht, A. Harji, P. Buhr, and A. Shukla, "*Comparing the performance of web server architectures*". In *EuroSys*, pp. 231–243, Mar 2007, DOI: 10.1145/1272998.1273021.

[8] B. Schroeder, A. Wierman, and M. Harchol-Balter, "*Open versus closed: A cautionary tale*". In 3rd *Networked Systems Design & Implementation*, pp. 239–252, May 2006.

[9] E. Shmueli and D. G. Feitelson, "*Using site-level modeling to evaluate the performance of parallel system schedulers*". In 14th *Modeling, Anal. & Simulation of Comput. & Telecomm. Syst.*, pp. 167–176, Sep 2006, DOI: 10.1109/MASCOTS.2006.50.

[10] E. Shmueli and D. G. Feitelson, "*Uncovering the effect of system performance on user behavior from traces of parallel systems*". In 15th *Modeling, Anal. & Simulation of Comput. & Telecomm. Syst.*, pp. 274–280, Oct 2007, DOI: 10.1109/MASCOTS.2007.67.

[11] E. Shmueli and D. G. Feitelson, "*On simulation and design of parallel-systems schedulers: Are we doing the right thing?*" *IEEE Trans. Parallel & Distributed Syst.* **20**(**7**), pp. 983–996, Jul 2009, DOI: 10.1109/TPDS.2008.152.

[12] M. welsh and D. Culler, "*Adaptive overload control for busy Internet servers*". In 4th *Conf. Internet Technology & Syst.*, USENIX, Mar 2003.

[13] N. Zakay and D. G. Feitelson, "*Workload resampling for performance evaluation of parallel job schedulers*". *Concurrency & Computation — Pract. & Exp.* **26**(**12**), pp. 2079–2105, Aug 2014, DOI: 10.1002/cpe.3240.

[14] N. Zakay and D. G. Feitelson, "*Preserving user behavior characteristics in trace-based simulation of parallel job scheduling*". In 22nd *Modeling, Anal. & Simulation of Comput. & Telecomm. Syst.*, pp. 51–60, Sep 2014, DOI: 10.1109/MASCOTS.2014.15.

[15] N. Zakay and D. G. Feitelson, "*Semi-open trace based simulation for reliable evaluation of job throughput and user productivity*". In 7th *IEEE Intl. Conf. Cloud Comput. Tech. & Sci.*, pp. 413–421, Nov 2015, DOI: 10.1109/CloudCom.2015.35.