Semi-Open Trace Based Simulation for Reliable Evaluation of Job Throughput and User Productivity

Netanel Zakay Dror G. Feitelson
School of Computer Science and Engineering
The Hebrew University of Jerusalem, 91904 Jerusalem, Israel
netanel.zakay@mail.huji.ac.il, feit@cs.huji.ac.il

***Abstract*—New scheduling algorithms are first evaluated using simulation. In these simulations, the workload has a huge influence on the measured performance of the simulated system. Therefore, it is customary to use workload traces recorded previously from real systems. Such open-system simulations preserve all the jobs' properties. However, preserving the jobs' arrival times actually destroys the logic of the user's workflow, especially dependencies and think times between successive jobs. Furthermore, performance in such simulations is measured by the average wait time and slowdown, under the fixed load and throughput conditions dictated by the trace. Therefore, it is impossible to evaluate the system's effect on throughput and productivity.**

As an alternative we propose semi-open trace based simulations that include dynamic user activity and internal feedback from the system to the users. In these simulations, like in a real system, users adjust their job-submittal behavior in response to system performance. As a result, the simulations produce different loads and throughputs for different scheduling algorithms or parametrizations. We implemented such a simulation for evaluating the schedulers of parallel job systems. We also developed a novel user-aware scheduler designed specifically to increase users' productivity. While conventional simulations cannot measure this scheduler's influence reliably, and would suggest it is useless, our simulation evaluates it realistically and shows its beneficial effect on the users' productivity and the system's throughput.

I. Introduction

Scheduling and resource management supporting the execution of jobs on large scale systems — such as supercomputers, clusters, and clouds — is a challenging endeavor. In such systems one cannot just put a new design into production without first extensively evaluating it. This leads to the challenge of faithful evaluations, which reflect all the complexities of the real system. Our focus is on one aspect of this complexity, which is analysing and simulating correctly the users' behavior and and their interaction with the simulated system.

Simulations are widely used in order to evaluate the performance of new system designs. These simulations are usually driven by traces recorded from a real system. Using a real trace is an attempt to create the most realistic simulations possible, which will lead to reliable evaluation results. In particular, using trace data directly retains all the structure that exists in the workload, including locality and correlations between different workload attributes, which might be lost if a statistical model was used instead. For example, traces of real parallel jobs available from the Parallel Workloads Archive [10] include data on job arrival times, parallelism, and runtime.

Note, however, that using the trace data as is implies an open-system model in the simulation. In other words, the arrival process is preordained, and is not affected by the state and performance of the simulated system. As a result the system throughput is dictated by the trace being used, so the simulation cannot be used to measure user productivity (where productivity is assumed to be related to the rate of running additional jobs). Consequently it also cannot be used to evaluate designs intended to promote productivity or satisfy certain service-level agreements, for example user-aware schedulers [21]. Moreover, the preordained arrivals exclude a faithful replication of user feedback effects, which are especially important in cloud system workloads. Finally, each trace can only provide a single load data point to the evaluation.

To overcome these limitations Shmueli suggested a new simulation methodology for evaluating parallel jobs schedulers [19], [21]. This methodology includes a feedback loop, in which the arrival of additional jobs depends on the termination of certain previous jobs submitted by the same user. He also suggested an algorithm to model the think times between jobs (the intervals from the termination of one job to the submittal of the next one) and the possibility of aborting sessions if performance is bad. This methodology solves the problem of predefined load, and facilitates evaluations where the achieved throughput (and hence productivity) in the simulation can change. But it had the drawback of using synthetic users, and sampling the attributes of the jobs from distributions. This is not consistent with our goal to have a workload which is as close to reality as possible.

In previous work we have shown how to extend this and integrate it with trace-based simulations, by extracting dependencies from the trace [27]. Given the dependencies, we can now change the arrival times in the simulations to reflect the users' reactions to system performance, while preserving all other properties of the recorded trace. We can also combine this with resampling, which allows us to generate multiple similar workloads from the same trace, and also to manipulate the load by modifying the number of users [26]. Combining resampling and feedback together leads to Trace-Based Users-Oriented Simulations (TBUOS), which achieves the dual goal of preserving as much detail as possible from the trace and

exhibiting realistic user behavior adjusted to the conditions during the simulation.

To demonstrate the importance and effect of TBUOS we suggest the User Priority Scheduler (UPS), which attempts to prioritize jobs belonging to users who are expected to be active on the system. This departs from the more common approach of prioritizing jobs according to their individual circumstances, e.g. how long they have been waiting in the queue. Evaluation with TBUOS shows that this scheduler can indeed improve system throughput and thereby also user productivity.

The contributions of this paper beyond previous work are:

- A new methodology for performance evaluation. TBUOS is a novel simulation which is the first to achieve a comprehensive realistic workload with controlled load (via resampling) and throughput adjustment (via dependencies and feedback). Previous work had suggested each of these components in isolation.
- Advanced understanding of the interplay between load, performance, and throughput, e.g. how throughput depends on the user population and not only on feedback.
- A new user-aware scheduler called UPS. This is conceptually similar to the CREASY scheduler proposed by Shmueli, but introduces the innovation of prioritization by users instead of treating jobs individually. This increases the level of coordination and reduces the risk of spreading resources too thinly across competing users.

Being the first framework to combine resampling with a simulation of the workflow behavior of users, our suggested approach cannot be expected to be the last word. Our focus is on showing an example of a semi-open simulation, and demonstrating the capabilities of such a simulation. This is especially important for cloud systems, which serve both batch workflows and interactive server workflows.

In the next two sections we review related work and our previous work on resampling and feedback. Section IV then explains how to combine feedback and resampling and shows simulation results. This is followed by Section V, which introduces UPS in detail and evaluates its performance, and by the conclusions.

II. Related Work

This paper touches upon three distinct concerns: the workloads used to evaluate parallel job schedulers, the simulation methodology, and the policy considerations of the schedulers.

In terms of workloads, the two most common approaches have been to replay job traces directly, or else to create statistical models based on job traces. Models facilitate the creation of multiple similar workloads, potentially with controlled variations such as different loads, but they suffer from not necessarily including all the important features of the real workload [9], [2]. Resampling (explained later) improves the representativeness of evaluation workloads by modeling only the parts that need to be manipulated, and using real data to fill in the remaining details [26].

The simulations typically follow an open systems model, where jobs are submitted by some external population of users. Therefore job arrivals are independent of system performance and state. But a closed system model with feedback may be more realistic [11], [19], [17], [16], [14], as poor performance may delay the submittal of additional jobs until previous ones have terminated, and perhaps even discourage users and cause them to submit fewer additional jobs. Such effects have been discussed in several different areas. In a database context, Hsu et al. claim that replaying timestamps from a trace loses feedback [13]. Ganger and Patt recognize the influence of the lack of feedback in simulations of storage subsystems as part of a larger system, and suggested giving higher priority to critical requests even if this degrades the performance of the storage system by itself [12]. In evaluations of networks, the feedback is important in shaping the traffic. For example, TCP congestion control is highly dependent on current conditions, so using traced timestamps for packets in simulations is wrong [23], [11]. Also, several papers dealt with a human user's feedback effects on the performance of applications [15], [22], [16], [24]. For example, Yang and Veciana modeled users that aborted their downloads due to poor performance. Synchronous protocols were shown to naturally throttle load due to feedback [18], [1].

In order to include feedback in evaluations one needs a model of how users react to load. While direct experimental evidence is rare [8], some works have considered user tolerance of delays and bandwidth limitations (e.g. [5], [15], [22], [24]). Shmueli used synthetic workloads with a feedback model and a user-aware scheduler designed to exploit this feedback effect [19], [20]. Our work extends those results in two significant ways. First, we show how to conduct more realistic simulations (TBUOS) using both feedback and resampling. This retains all the structure that exists in workload traces, and avoids any potential over-simplifications that may exist in synthetic workloads. Second, we propose a user-aware scheduler (UPS) that is not based directly on the feedback model being used. This generalizes the results and eliminates the risk that they depend on prior knowledge.

A parallel job contains multiple processes, which need to run in parallel on distinct (possibly virtual) processors. The scheduler allocates processors to such jobs and then they run with no preemptions. A common scheduling approach is EASY, which serves jobs in arrival order and uses backfilling (taking jobs from the back of the queue to fill in holes in the schedule) to pack jobs more tightly and optimize response time and slowdown. The UPS scheduler, based on EASY, has a similar basic structure to Shmueli's CREASY scheduler [21], but introduces user-based prioritization.

Scheduling on cloud platforms is a relatively young field of research. Most papers focus on economic aspects and the matching of requirements to resources and only a few tried to simulate and improve their performance. For both purposes, it is important to take into consideration the feedback and interaction with the users' workflows. The jobs order in a cloud workflow is predefined and preserving it is essential in order to produce the required results. Therefore Di et al. have conducted extensive trace-based analyses of workloads,

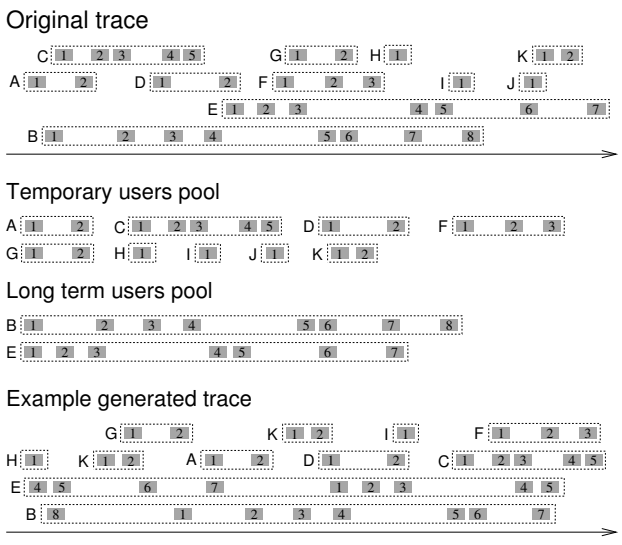

Fig. 1. Illustration of the resampling process.

and created the GloudSim trace-driven simulation environment [7], [6]. While common simulations such as CloudSim and GloudSim ignore dependices, some suggest a new simulation methodology that enforces depedency preservation [4], [3]. This methodology contains a similar mechanism to ours to enforce the preservation of dependencies, but it assumes the dependencies are known; we need to deduce dependencies from a recorded trace.

III. Background

TBUOS is based on a combination of resampling and feedback. For completeness, we review these two mechanisms.

A. Workload Resampling

Replaying a trace provides only a single data point of performance for one workload, while in evaluations several related workloads may be needed, e.g. to compute confidence intervals. Resampling is a way to achieve this [26]. It is done by partitioning workload traces into their basic independent components, and re-grouping them in different ways to create new workloads. Thus resampling combines the realism of real traces with the flexibility of models.

In the context of parallel job scheduling resampling is done at the level of users. We first partition the workload into individual subtraces per user. We then sample from this pool of users to create a new workload trace.

Analysis of real traces suggests the identification of two types of users. Temporary users are those that interact with the system for a limited time, for example while conducting a project. After a short duration they leave the system never to return. Long-term users, in contrast, appear to be active during much of the trace, and may be expected to have been active before logging started, and to send more jobs also after the end of the recording period.

During simulation we handle the dynamics of these user types differently. The simulation is built in units of one week, so as to preserve not only daily but also weekly cycles of activity. In the beginning of each week, a random number of temporary users is chosen such that the accumulated number will be distributed around the original average number of temporary users. The arrival times of the jobs of these users are shifted so that their first recorded job will arrive in this week. A user may be chosen multiple times or zero times.

Long term users are assumed to be active continuously. To create a variation between simulations, we start each such user from an arbitrary week in the trace and shift all subsequent jobs accordingly. If the simulation is supposed to continue after their last job has terminated, we just duplicate their whole sequence of jobs again. Note that the population of long term users is constant throughout the simulation.

An illustration of the resampling process appears in Figure 1. In this example, user K was chosen twice, and the long term user E was chosen to start from his 4th job, and after his 7th one we start again from the first one.

Due to the regeneration of the long term users' jobs and the addition of new temporary users each week, resampling simulations can continue indefinitely. We usually stop the simulation at a time that corresponds to the length of the original trace. Hence the number of jobs is approximately the same as in the original trace. Resampling produces workloads that are similar to the original workload [26]. However, it is also easy to induce modifications, such as extending a trace or changing its average load. Importantly, while the resampled workloads differ from the original in length, statistical variation, or load, they nevertheless retain important elements of the internal structure such as sessions and the relationship between the sessions and the daily work cycle.

B. Simulations with Internal Feedback

Commonly used simulations are trace driven, and use an open-system model to play back the trace and generate the workload. This means that new requests get issued during simulation exactly according to the timestamps from the trace, irrespective of the logic behind the behavior of the users and of the system state. As a result the throughput of the system being evaluated is also dictated by the timestamps, instead of being affected by the actual performance of the scheduler.

The problem with this approach is that each trace contains a signature of the scheduler that was used on the traced system [19]. In other words, the users actions reflect their reactions to the scheduler's decisions. And real users would react differently to the decisions of another scheduler. Therefore, when we want to evaluate a new scheduling policy using a representative workload, the simulation should reflect user reactions to the evaluated scheduler rather than to the original scheduler. *It is more important to preserve the logic of the users' behavior than to repeat the exact timestamps*.

The way to integrate such considerations into trace-driven simulations is by manipulating the timing of job arrivals. In other words, the *sequence* of jobs submitted by each user stays the same, but the *submittal times* are changed [27].

Specifically, each job's submit time is adjusted to reflect feedback from the system performance to the user's behavior.

The first step in creating such a feedback based simulation is to regenerate the dependency relations between the users' batches of jobs. Session are defined based on inter-arrival times between jobs [25]. Batches are jobs within the same session that run in parallel without waiting for each other, as when a user submits a new job before the previous one had terminated. But a batch can depend on the arrival or termination of previous ones, and these constraints can be tracked and enforced. A batch then arrives to the simulated system only when it has no pending constraints.

However, a batch cannot arrive immediately when all its constraints are removed. Rather, its arrival should reflect reasonable user behavior. One possible model of user behavior is the "fluid" user model. The idea of this model is to retain the original session times of the users, but allow batches of jobs to flow from one session to another according to the feedback. To do that, we keep each session's start and end timestamps from the original workloads. Batches are given think times from the distribution of intra-session think times, but if this leads to an arrival beyond the end of the session, the batch will arrive at the beginning of the next session. If a session is skipped, it is reinserted the following week. This model creates workloads that have very similar distributions to the original [27].

IV. Trace-Based Users-Oriented Simulation

Our goal is to develop a simulation methodology which supports features such as increasing the load and obtaining multiple data points for each setting, uses realistic workloads, and simulates the effect of a system's design reliably, *including its influence on the users' behavior* (which may result in a different throughput). In the previous section we described the two main components, namely resampling and feedback. In this section we will show how to combine them into a trace-based user-oriented simulation (TBUOS). We start with a description of the technicalities of TBUOS. Then, we explain how this leads to more realistic simulations. Finally, we analyze the dynamics of TBUOS, and show how TBUOS supports simulations where system performance affects throughput.

A. Integrating Resampling and Feedback

The description of TBUOS is quite short (Figure 2). First, we preserve all the elements of resampling as they were. This includes:

1) Building the long term users and the temporary users pools.
2) Sampling a few temporary users each week (according to the original rate).
3) Regenerating the long term users when their traced activity is finished.

By not changing anything we retain all the benefits of resampling, such as the ability to extend a trace or increase the load (as was shown in [26]).

The main difference from conventional simulations using resampling is that when a user arrives to the system (whether a temporary user or the regeneration of a long term user), all his

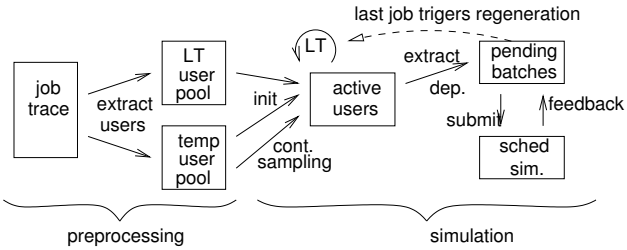

Fig. 2. TBUOS flow.

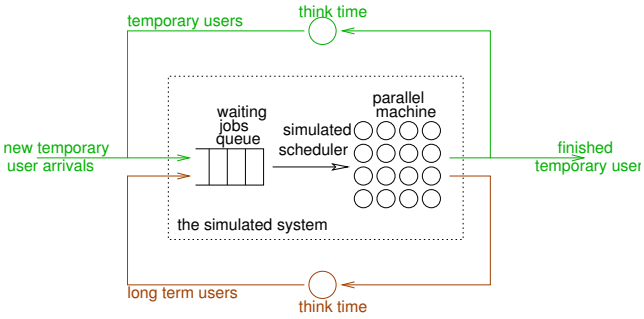

Fig. 3. TBUOS model of long term and temporary users.

jobs are not immediately inserted into the simulation's queue of pending jobs. Rather, we use the feedback mechanism in an integrated manner within the resampling so that job arrivals will reflect the user behavior model. But the Feedback itself remains without any change. This includes:

1) Building the batches structures from the users' sub-traces.
2) Deducing the relations between the batches of each user.
3) Removing constraints as batches arrive and terminate.
4) Releasing to the simulation only the independent batches.
5) Using the user model to choose each batch's arrival time.

The combination of resampling and feedback simply means that jobs that are created by the resampling are then passed through the feedback mechanism. This divides them into batches, tracks each batch's constraints, and sends to the queue of pending jobs only the independent batches at each time step.

Temporary users in TBUOS arrive at the same average rate as in the trace, and each one has the same jobs as in the trace. When the last job of the user terminates, the user leaves the system. Therefore, each temporary user contribute the same load in TBUOS as in conventional simulations. However, due to the feedback effects, the jobs of each individual user may be spread out differently in simulated time.

The number of long term users in TBUOS is constant and equal to their number in the trace. However, unlike temporary users, they are all generated during the initialization, and they never leave the system. Instead, they are regenerated after the termination of their last job. Due to the feedback effect, the end time of the last job is not predefined but rather depends on the terminations of previous jobs. This means that the number of jobs submitted by simulated long term users will be different from the number in the original trace, depending on the performance of the system in the simulation. For example,

if the the new design leads to better service, the user will send his next batches earlier. Therefore, the users' traced activity will be finished earlier, and the user will be regenerated and send more jobs, contributing to increased throughput.

Figure 3 illustrates the behavior of the two types of users, and shows why TBUOS is a semi-open system. The temporary users are the open part. They arrive independently of system state, send their jobs (using internal feedback between the jobs), and eventually leave the system. The long term users are the closed part. They remain in the system for the duration of the simulation. It is worth mentioning that only up to about 20% of the jobs in our workloads belong to temporary users, and in average much less than that [26]. Therefore, a very big part of the workload comes from the closed part.

B. Improved Realism of Simulations

As stated our goal is to develop a simulation methodology which supports features such as increasing the load and obtaining multiple data points for each setting, but at the same time maintains realistic workloads with all the features that exist in real traces. TBUOS achieves this by inheriting the characteristics of resampling and feedback. For example, each user submits the same jobs in the same sequence with the same dependencies between them, as in the original workload [26], [27].

However, resampling by its very nature does mix up the users from the original trace. Thus user activity that was originally performed when the system was highly loaded may be matched up with activity of another user that was originally performed when the system was lightly loaded [19]. This is an unnatural combination. When the system is highly loaded jobs take longer to execute and therefore come at longer intervals. When the system is lightly loaded jobs run immediately, leading to a higher rate of submitting new jobs. These different behaviors are not expected to coexist at the same time.

TBUOS solves this problem by using feedback in addition to resampling. The feedback preserves all the properties of the workload (including internal dependencies and think-times between the jobs) except the arrival times of the jobs, which are adapted to the system's performance. In particular, it adjusts the job arrival rate to match the momentary conditions in the simulated system. As a result, the simulated workload in TBUOS is a more representative workload than the original traced workload *in the context of the simulated system*.

To demonstrate this effect we compare an evaluation of the EASY scheduler using conventional simulations with resampling and using TBUOS, based on traces from the Parallel Workloads Archive[1] (Figure 4). The first observation is that

[1]The traces used in this paper are from the following systems: the San Diego Supercomputer Center Blue Horizon (BLUE), the Seth cluster from the High-Performance Computing Center North in Sweden (HPC2N), the Los Alamos National Lab Connection Machine CM-5 (CM5), the Cornell Theory Center IBM SP2 (CTC), the Argonne National Laboratory BlueGene/P system Intrepid (Intrepid), the Swedish Royal Institute of Technology IBM SP2 (KTH), the San Diego Supercomputer Center DataStar (SDSC-DS), and the San Diego Supercomputer Center IBM SP2 (SDSC-SP2). Full details about these logs are available at http://www.cs.huji.ac.il/labs/parallel/workload/.

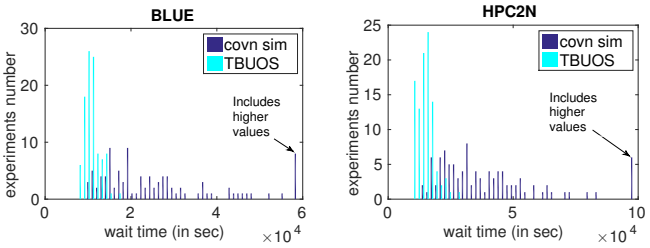

Fig. 4. Histograms of the average wait time (in seconds) in a hundred simulations of the EASY scheduler. Using TBUOS leads to lower wait times and a less skewed distribution.

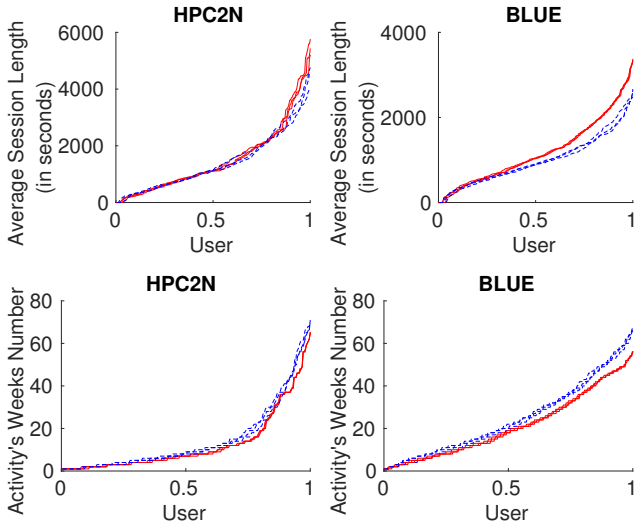

Fig. 5. Comparing user behavior in resampling (red solid lines) and TBUOS (blue dashed lines) workloads. The X axis represents users sorted by the metric being shown; user IDs are normalized to [0..1] to enable easy comparison between logs with different numbers of users.

the average and maximum wait times for each log are usually much lower in TBUOS. Indeed, in some simulations the average wait time was more than 100,000 seconds, which is 27.8 hours, meaning that the average wait time came out to be more than a day. Under TBUOS there was no simulation with an average wait time of more than 8 hours, and the common values are smaller than 20,000 seconds (5.6 hours), which is much more realistic.

We conjecture that the reason for this effect is that the resampling loses the synchronization between the users as exists in reality. An interesting question is how TBUOS avoids the unrealistic long wait times. To analyze this, we compared the relevant user behavior properties, including the number of weeks in which the users were active and the average session lengths (Figure 5). In TBUOS users are active during more weeks and their session lengths tend to be shorter. This may be interpreted as evidence that in TBUOS users react to the system state, and if the system is overloaded, they send less jobs during the session and delay the next session to another time.

Another interesting difference between the distributions in Figure 4 is their shapes. In most cases the conventional

TABLE I
AVERAGE WAIT TIME MAX/MIN RATIO WITH AND WITHOUT FEEDBACK.

Sim type	BLUE	CM5	CTC	HPC2N	Intrepid	KTH	SDSC-DS	SDSC-SP2
conv + res	17.31	7.08	5.48	12.52	2.68	6.59	9.29	13.47
TBUOS	2.25	1.42	1.58	2.99	2.37	2.04	2.27	2.05

simulations with resampling create skewed distributions with a tail to high values, whereas TBUOS creates a much narrower and less skewed distribution. This is quantified using the ratio of maximal to minimal average wait time results in Table I. This reflects the same effect as above: resampling can create unrealistic load conditions, but TBUOS avoids them. To appreciate the significance of this finding, recall that these figures show the distribution of results from 100 independent simulations. If you run only one simulation, you can get any of these results. In particular, without TBUOS your simulation may happen to be from the tail of the distribution, and therefore non-representative in general. with TBUOS the danger is much reduced, because all simulations are reasonably similar.

C. Enabling an Effect on Throughput

Throughput is probably the best indicator for user productivity, and testifies to the scheduler's capacity for keeping its users satisfied and motivating them to submit more jobs. In conventional simulations, the throughput of the system being evaluated is dictated by the job arrival timestamps, instead of being affected by the actual performance of the scheduler. Even when feedback is used, this alone does not change the throughput.

It is worth mentioning that feedback-based simulations do change the throughput *per user*. In other words, the throughput of each user does depend on the simulated system's performance, which is an important step forward. But this is an uncommon performance metric, and more research about its impact is needed. The more common metric of *global* throughput is not changed by feedback alone, because the number of users and the number of jobs that each one of them submits are dictated by the trace.

TBUOS, which combines resampling and feedback, is unique in facilitating dynamically changing throughputs. The large number of long term users operate as a closed-system model, and lead to throughput that depends on the system's performance. If the scheduler allows a user to send jobs earlier, this user will actually send *more* jobs during the simulation. Therefore, in TBUOS the scheduler is able to influence the users' productivity, and the simulation will actually produce different throughputs for different system designs.

To demonstrate the ability of the scheduler to change the throughput in TBUOS we consider the simulation of a poor first-come, first-serve scheduler (FCFS) and a more effective EASY scheduler. For each simulation type (conventional+resampling or TBUOS) and scheduler (FCFS or EASY) we run a hundred simulations and tabulate the throughputs achieved. Figure 6 shows the results. Using conventional simulation with resampling, the end time and the intervals between the jobs are preserved. As a result, the distributions of throughput under FCFS and EASY are similar to each other and to the original value.

On the other hand, it is easy to see that in TBUOS the EASY scheduler led to increased throughput relative to the FCFS scheduler for all the traces. The difference testifies to the more realistic simulation of the users in TBUOS, including a logical response to a poor system performance. This results in reduced throughput with the FCFS scheduler, somtimes less than in the original trace.

An important observation regarding Figure 6's results is that the throughput with TBUOS can be quite different from the throughput of the original trace. This indicates that each trace includes a signature of the original scheduler from the traced system, and may not be suitable for the direct evaluation of other schedulers. But TBUOS compensates for this mismatch, and can even compensate for gaps in the evaluation, as actually happened to us in the following example. The CTC trace was initially simulated using a system size of 430 nodes. But later it was discovered that the correct size was 336 nodes [10], implying that the simulation had an artificially low load. With such low load good performance is easy to achieve, and when using TBUOS the throughput was increased considerably to "fill the system". Once we identified the source of the problem and used the correct size, the change in load was reduced significantly (Figure 7).

Note that TBUOS is not specifically designed to handle workloads with unrealistic low load and create a realistic simulation. For example, if the recorded trace has only one session per week, TBUOS will also have one session per week (and therefore extremely low load). However, when we use a recorded workload with realistic users' properties, if the simulated system has good performance, the users may send more jobs and use the system's resources better. This enables the evaluation to overcome the potential deficiencies of the recorded trace in the context of this specific simulation.

V. Simulating A User-Aware Scheduler

The performance metrics used in conventional trace-based simulations are the average response time and slowdown. This led researchers to focus on the packing of jobs in the schedule as a means to improve the average values in simulation. The suggested justification is that decreasing the average wait time will improve user satisfaction and productivity. However, this reasoning is debatable, and it may even be that increased wait times correspond to higher user productivity. This can be demonstrated by a simple hypothetical example. Assume that the system is highly loaded and all the processors are being used. In this situation, using a FCFS scheduler ensures fairness, but also guarantees that all users have to wait long times for their jobs to run. This risks users becoming frustrated and aborting their sessions, thereby reducing their productivity. An alternative LIFO scheduler can perhaps reduce this effect by prioritizing the most recently active users at the expense of the users who had submitted jobs longer ago.

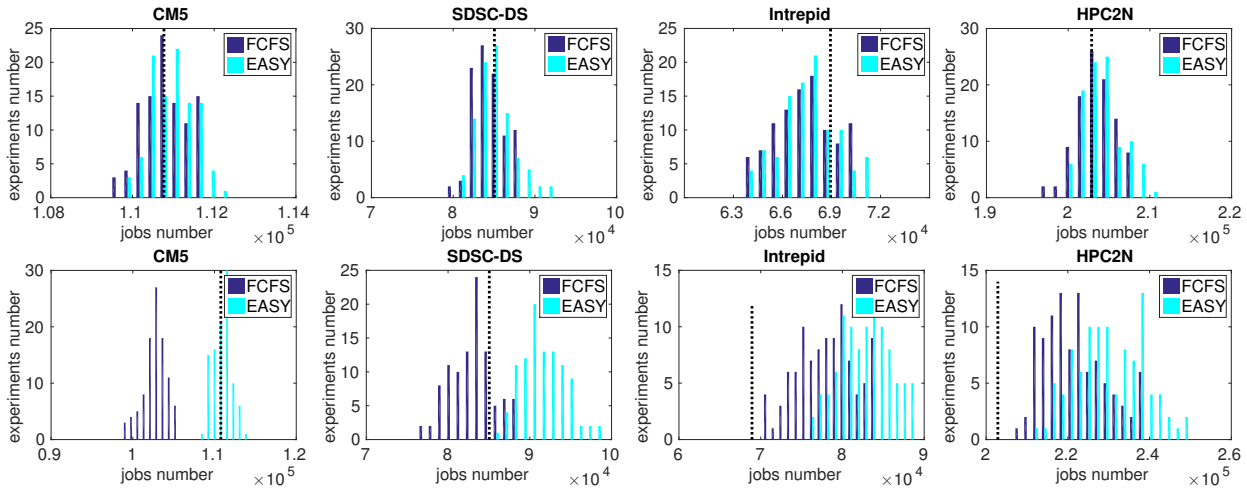

Fig. 6. Histograms of the throughput in one hundred simulations of EASY and FCFS schedulers. Top: conventional simulation (with resampling but no feedback). Bottom: TBUOS. The vertical dashed black line represents the throughput in the original workload. Clearly in conventional simulations the throughput does not depend on the system, in contrast to TBUOS in which EASY leads to higher throughputs.

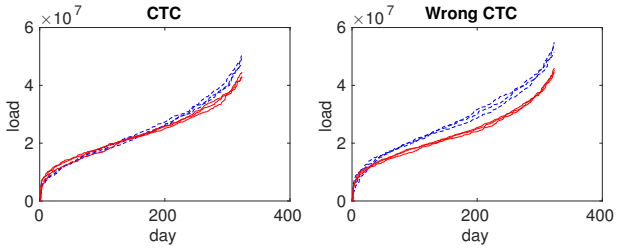

Fig. 7. Comparing the load per day (number of processor-seconds) in two versions of the CTC trace which differ only in the specified system size. TBUOS (blue dashed lines) compensates for the artificially lower load when too many nodes were specified.

More generally, user-aware schedulers try to anticipate user behavior. The scheduler's goal is then to improve productivity by facilitating more effective use of the system, allowing users to submit more jobs. This is done by trying to assess whether users are still active, and prioritizing those who have the highest probability of submitting more jobs. This may increase the average wait time, because jobs belonging to users who are thought to be inactive are delayed, but will hopefully also increase the system throughput and the productivity of active users. Evaluating such schedulers requires methodologies like TBUOS, because under conventional simulations the throughput is fixed in advance.

A. The User Priority Scheduler

In this section we describe a simple new scheduler named the User Priority Scheduler (UPS). This is a user aware scheduler intended to motivate users to submit more jobs. It's general structure is similar to the CREASY scheduler suggested by Shmueli [21]. Jobs submitted to UPS are kept in a priority queue. When a scheduling decision has to be made (e.g. when some job terminates and processors become available) the highest priority job is selected for execution. The priorities are based on a weighted sum of two terms, one of which reflects the probability that a user will submit more jobs, and the other reflecting the waiting time of the job.

In more detail, the operation of UPS is based on EASY. Selecting jobs for execution is done as follows. First the queue is scanned in job priority order, and all jobs that can run (because enough processors are available) are dispatched. Then a reservation is made for the first queued job in order to ensure that it will not be starved. Finally the remainder of the queue is scanned in user priority order and jobs are backfilled provided they do not violate the reservation for the first job.

The differences from EASY are as follows. In EASY jobs are prioritized by their arrival time, and both the original scheduling and the backfilling are done in this order. In UPS the original scheduling is done according to job priorities, which reflect a weighted sum of wait time and user priority. The backfilling is done in a slightly different order: first according to user priority, and then according to waiting time for all the jobs of each user.

Given a job J submitted by a user u its job priority is calculated by the following expression:

$$Jpri(J) = \alpha_{user} \cdot Upri(u) + \frac{\alpha_{arr} \cdot J.wait}{4 \cdot (\text{SECONDS\_IN\_HOUR})}$$

$Jpri$ and $Upri$ are the job and user priorities, respectively. $Upri$ represents the scheduler's "user awareness" as explained below. α_{user} and α_{arr} are the weights of the two terms, where $\alpha_{user} + \alpha_{arr} = 1$. They determine the balance between the user awareness and the waiting time. If $\alpha_{arr} = 1$ then we only consider the wait time and ignore the user, so this is similar to EASY (but with backfilling order based on the user priority). If $\alpha_{user} = 1$ all the weight is placed on user awareness, at the risk of starving jobs submitted by low-priority users.

The user priority $Upri(u)$ reflects the scheduler's goal to give higher priority to users who are assumed to be active and therefore have the potential to submit additional jobs. As an initial suggestion, we consider two metrics for user activity:

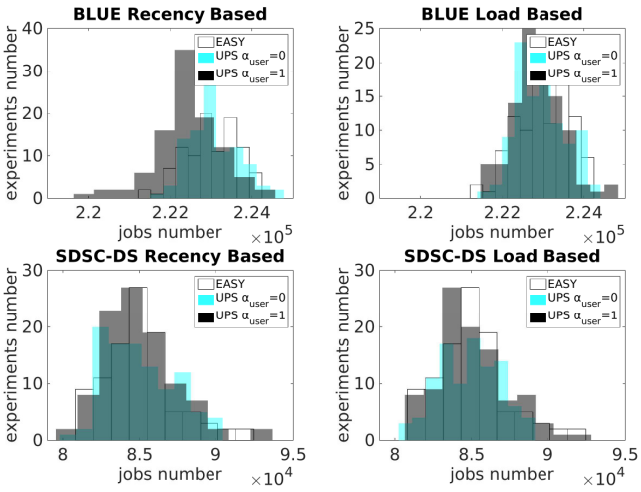

Fig. 8. Distribution of throughput results in simulations without feedback of EASY and UPS (recency based on left and load based on right) with $\alpha_{user} = 0$ or 1. The scheduler doesn't have a significant effect on the throughput.

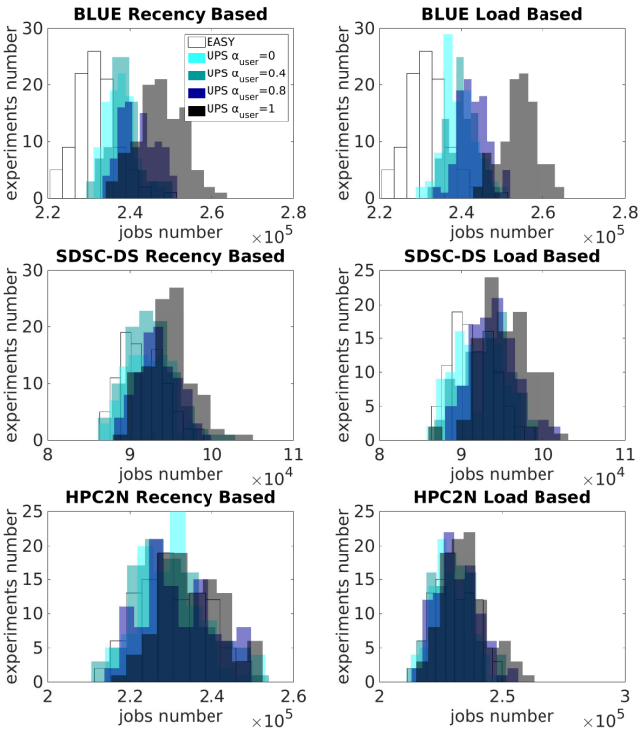

Fig. 9. Distribution of throughput results in TBOUS Simulations comparing EASY with UPS with different values of α_{user} (left: recency based; right: load based). Different logs lead to different effects of the user awareness, from significant differences at top, through noticeable differences in the middle, to minor differences at bottom.

1) Load based: priority is determined by the estimated work (requested runtime×processors) in waiting jobs, where more work implies lower priority. The idea is that if users have dependencies between their jobs, we will be able to run quickly the jobs of the user that needs less resources, and then he will be able to send the following jobs sooner, maybe even in the same session.
2) Recency based: priority is determined by the last job arrival from this user, where more recent activity leads to higher priority. The hypothesis of this approach is that maybe the user is still within an active session, and if he will see the results rapidly, he may decide to continue the session.

Either of these metrics can be used to define an order on users, and then we assign user i in this order the priority $Upri(u_i) = 1/i$. Consequently the effective range of $Upri(u)$ is $[0, 1]$. Users with no waiting jobs are assigned $Upri(u) = 0$.

$J.wait$ is the waiting time of job J in seconds, so a job's priority grows linearly with the time it waits in the queue. The dividing factor of four hours normalizes this with respect to the user priorities, such that a value of 1 is reached after 4 hours. Thus if $\alpha_{user} = \alpha_{arr} = 0.5$ a newly arrived job belonging to the highest priority user will have $Jpri = 0.5$, and any job by any other user that is waiting for 4 hours or more will have a higher priority than it. But if $\alpha_{user} = 0.2$ and $\alpha_{arr} = 0.8$ then any other job that is waiting more than 1 hour will already have a higher priority.

The UPS avoids starvation when $\alpha_{arr} > 0$ because $Upri \leq 1$, so eventually every job can become the highest priority job. It then gets a reservation due to the EASY algorithm, and subsequently gets an allocation of processors.

B. Simulation Results

First we analyze the performance of UPS using a simulation without feedback. We compare the UPS scheduler with $\alpha_{user} = 0$ and $\alpha_{user} = 1$ for both approaches for prioritizing the users (Figure 8). The throughput does not change noticeably between the different schedulers. In particular, $\alpha_{user} = 1$ does not lead to higher throughput, which means that user awareness does not seem to provide any benefits.

However, we claim that the user aware scheduler is actually better. To demonstrate this, we show in Figure 9 the results obtained using TBUOS for different values of α_{user}, and both approaches for prioritizing the users. We can see that with feedback UPS facilitates higher throughputs in average than EASY, and the gap grows with higher emphasis on user prioritization (larger α_{user}). Moreover, load-based prioritization appears to be more effective than recency based prioritization. However, the improvement of using user aware schedulers actually depends on the workload. In some logs, such as BLUE, there is a very significant effect. In others, such as HPC2N, there is only a minor effect. The logs shown in the figure were selected to demonstrate the range of effects we saw with the 8 logs used.

VI. Conclusions and Future Work

Conventional open-model trace-based simulations, which are commonly used to evaluate the performance of a new system designs, do no adjust the simulated workload to the system state. As a result they are unable to measure the true throughput with the new system design.

We suggest an alternative methodology named TBUOS. TBUOS also uses a recorded trace, and retains all the attributes

of the recorded workload except one: instead of keeping job arrival times it keeps job dependencies and think-times, and adjusts the arrival times to reflect dependencies and performance. The simulation doesn't simulate only the scheduler, but also the users' behavior, namely how users would respond to the new system. As a consequence, simulations may produce different throughputs depending on the simulated system.

Moreover, scheduling algorithms which are based on affecting the input workload can't be evaluated with conventional simulations. The UPS is such a scheduler, and requires an evaluation with TBUOS to demonstrate its ability to increase the throughput.

Creating a user-oriented simulation is a new world, and there is no single correct approach for doing so. While TBUOS provides a proof-of-concept for user-based semi-open simulation, other approaches are possible. Specifically, several assumptions may be changed depending on the modeler's point of view. For example, maybe the temporary users should also have a dynamic number of jobs? Maybe some of the long term users may leave if the performance is too poor? Maybe even the jobs properties (such as the requested number of processors) should be adapted to the system?

In future work we intend to continue the development of TBUOS and UPS. More work is needed to better characterize user behavior and create better user feedback models. We also intend to consider more advanced models of the user population dynamics, including session aborts and system abandonment. Furthermore, we need to evaluate UPS more deeply, and compare it to other approaches, such as Shmueli's CREASY scheduler. Finally, A long-term goal is to implement this work also in additional domains, demonstrating this methodology to be effective in general in performance evaluation, and not only in the context of parallel jobs scheduling.

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

REFERENCES

[1] V. S. Adve and M. K. Vernon, "*Performance analysis of mesh interconnection networks with deterministic routing*". *IEEE Trans. Parallel & Distributed Syst.* **5(3)**, pp. 225–246, Mar 1994, doi: 10.1109/71.277793.

[2] J. Aikat, S. Hasan, K. Jeffay, and F. D. Smith, "*Towards traffic benchmarks for empirical networking research: The role of connection structure in traffic workload modeling*". In 20th *Modeling, Anal. & Simulation of Comput. & Telecomm. Syst.*, pp. 78–86, Aug 2012, doi: 10.1109/MASCOTS.2012.19.

[3] M. A. Amer, A. Chervenak, and W. Chen, "*Improving scientific workflow performance using policy based data placement*". In 22nd *Policies for Distrib. Syst. & Netw.*, pp. 86–93, Jul 2012.

[4] W. Chen and E. Deelman, "*WorkflowSim: A toolkit for simulating scientific workflows in distributed environments*". In 8th *IEEE Intl. Conf. E-Science*, pp. 1–8, Oct 2012.

[5] P. Cremonesi and G. Serazzi, "*End-to-end performance of web services*". In *Performance Evaluation of Complex Systems: Techniques and Tools*, M. C. Calzarossa and S. Tucci (eds.), pp. 158–178, Springer-Verlag, 2002, doi:10.1007/3-540-45798-4_8. Lect. Notes Comput. Sci. vol. 2459.

[6] S. Di and F. Cappello, "*GloudSim: Google trace based cloud simulator with virtual machines*". *Software — Pract. & Exp.* 2015, doi: 10.1002/spe.2303.

[7] S. Di, D. Kondo, and F. Cappello, "*Characterizing and modeling cloud applications/jobs on a Google data center*". *J. Supercomput.* 2014, doi: 10.1007/s11227-014-1131-z.

[8] P. A. Dinda, G. Memik, R. P. Dick, B. Lin, A. Mallik, A. Gupta, and S. Rossoff, "*The user in experimental computer systems research*". In *Workshop Experimental Comput. Sci.*, art. no. 10, Jun 2007, doi: 10.1145/1281700.1281710.

[9] 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.

[10] D. G. Feitelson, D. Tsafrir, and D. Krakov, "*Experience with using the Parallel Workloads Archive*". *J. Parallel & Distributed Comput.* **74(10)**, pp. 2967–2982, Oct 2014, doi:10.1016/j.jpdc.2014.06.013.

[11] S. Floyd and V. Paxson, "*Difficulties in simulating the Internet*". *IEEE/ACM Trans. Networking* **9(4)**, pp. 392–403, Aug 2001, doi: 10.1109/90.944338.

[12] G. R. Ganger and Y. N. Patt, "*Using system-level models to evaluate I/O subsystem designs*". *IEEE Trans. Comput.* **47(6)**, pp. 667–678, Jun 1998, doi:10.1109/12.689646.

[13] W. W. Hsu, A. J. Smith, and H. C. Young, "*The automatic improvement of locality in storage systems*". *ACM Trans. Comput. Syst.* **23(4)**, pp. 424–473, Nov 2005, doi:10.1145/1113574.1113577.

[14] D. Krishnamurthy, J. A. Rolia, and S. Majumdar, "*A synthetic workload generation technique for stress testing session-based systems*". *IEEE Trans. Softw. Eng.* **32(11)**, pp. 868–882, Nov 2006, doi: 10.1109/TSE.2006.106.

[15] R. Morris and Y. C. Tay, *A Model for Analyzing the Roles of Network and User Behavior in Congestion Control*. Tech. Rep. MIT-LCS-TR898, MIT Lab. Computer Science, May 2003.

[16] R. S. Prasad and C. Dovrolis, "*Measuring the congesion responsiveness of Internet traffic*". In 8th *Passive & Active Measurement Conf.*, pp. 176–185, Apr 2007, doi:10.1007/978-3-540-71617-4_18.

[17] 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.

[18] S. L. Scott and G. S. Sohi, "*The use of feedback in multiprocessors and its application to tree saturation control*". *IEEE Trans. Parallel & Distributed Syst.* **1(4)**, pp. 385–398, Oct 1990, doi:10.1109/71.80178.

[19] 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.

[20] 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.

[21] 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.

[22] D. N. Tran, W. T. Ooi, and Y. C. Tay, "*SAX: A tool for studying congestion-induced surfer behavior*". In 7th *Passive & Active Measurement Conf.*, Mar 2006.

[23] W. Willinger, V. Paxson, and M. S. Taqqu, "*Self-similarity and heavy tails: Structural modeling of network traffic*". In *A Practical Guide to Heavy Tails*, R. J. Adler, R. E. Feldman, and M. S. Taqqu (eds.), pp. 27–53, Birkhäuser, 1998.

[24] S. Yang and G. de Veciana, "*Bandwidth sharing: The role of user impatience*". In *IEEE Globecom*, vol. 4, pp. 2258–2262, Nov 2001, doi:10.1109/GLOCOM.2001.966181.

[25] N. Zakay and D. G. Feitelson, "*On identifying user session boundaries in parallel workload logs*". In *Job Scheduling Strategies for Parallel Processing*, W. Cirne et al. (eds.), pp. 216–234, Springer-Verlag, 2012, doi:10.1007/978-3-642-35867-8_12. Lect. Notes Comput. Sci. vol. 7698.

[26] 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.

[27] 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.