On Identifying User Session Boundaries in Parallel Workload Logs

Netanel Zakay Dror G. Feitelson

School of Computer Science and Engineering
The Hebrew University of Jerusalem
91904 Jerusalem, Israel

Abstract. The stream of jobs submitted to a parallel supercomputer is actually the interleaving of many streams from different users, each of which is composed of sessions. Identifying and characterizing the sessions is important in the context of workload modeling, especially if a user-based workload model is considered. Traditionally, sessions have been delimited by long think times, that is, by intervals of more than, say, 20 minutes from the termination of one job to the submittal of the next job. We show that such a definition is problematic in this context, because jobs may be extremely long. As a result of including each job's execution in the session, we may get unrealistically long sessions, and indeed, users most probably do not always stay connected and wait for the termination of long jobs. We therefore suggest that sessions be identified based on proven user activity, namely the submittal of new jobs, regardless of how long they run.

1 Introduction

There has recently been increased interest in user-based workload models for parallel supercomputers [16, 11–13]. Such models are generative in nature. This means that instead of modeling the statistical properties of the workload, as was done for example by Jann et al. and Lublin and Feitelson [3, 5], they model the process by which the workload is generated. As jobs are submitted by users, this implies the need to model user behavior.

The motivation for using generative user-based workload models is that such models enable us to include feedback effects in performance evaluations. The stream of jobs submitted to a parallel supercomputer is the result of an interaction between the system and its users. If the system is responsive, users will submit more jobs. If it performs poorly, users may depart in frustration and refrain from submitting more jobs. When we evaluate the performance of a new scheduler design, we need to include such feedback and its effect on user behavior [11, 13].

An important characteristic of user behavior is its temporal pattern. Human users may work for some time, but then they stop and do something else. The periods of continuous work are called *sessions*. There are usually many more user sessions during the day than during the night or weekend, leading to the creation of an overall daily and weekly cycle of activity. Understanding how such patterns are generated is a basic component in defining a generative workload model.

Data about user behavior is contained in accounting logs from existing parallel machines, such as those that are available in the Parallel Workloads Archive [9]. Unfortunately, these logs only include data about individual jobs: when the job was submitted, when it started to run and when it terminated, how many processors it used, etc. Importantly, we usually also know the identity of the user who submitted the job (or at least anonymized identity, in the interest of preserving privacy). But we do not know when the user started or ended each session. If we want to characterize this behavior, we need to glean this data based on the pattern of job submissions.

The common approach to extracting session data is based on the assumption that users typically wait to see the results of their jobs, and then submit additional jobs. Thus the user session extends from the submittal of some job till the termination of that or some later job. Zilber at al. have suggested that breaks of 20 minutes or more between successive jobs indicate a session break [16], and others have followed this definition [12].

The problem with this definition is that parallel jobs may be very long. In some logs we even observe jobs that run for multiple days. Obviously it is unreasonable to expect the user to remain active for such a long time waiting for the job to terminate. And indeed we find that sessions defined according to the above definition may be much longer than is reasonable. We therefore suggest an alternative approach, whereby sessions are defined based on only explicit user activity, namely the submittal of new jobs. The times at which the jobs terminate are ignored.

A basic problem with this line of research is that ground truth is not available for comparison. In other words, we do not really know when users started or ended their sessions. We therefore need to make qualitative judgments. Our main criterion is to look at the distribution of session lengths that is generated by the analysis, and to reject methods that lead to distributions with obvious deficiencies (such as sessions that extend over more than a week).

The next section describes the technical details of how sessions may be defined according to different approaches. Section 3 discusses the selection of threshold values used to identify session breaks. Section 4 shows how we can use the distribution of generated sessions to select among two competing approaches for how to apply the threshold. Section 5 identifies some problems that occur when using inter-arrival times rather than think times. Finally, Section 6 introduces the notion of using the generated session lengths as a criterion for accepting or rejecting different approaches.

2 Definition of Sessions and Batches

Intuitively, a *session* is a period of continuous work by a user. This does not mean that the user was active 100% of the session's time. A user may run a job to completion, think about the result, and then run another job, all within the same session.

The above description seems to imply sequential work, where jobs in a session never overlap. Empirical evidence from parallel supercomputer job logs shows that this is clearly not always the case, and jobs may overlap. Given such an overlap, the later job cannot depend on the earlier one. Following Shmueli, we call a set of such independent, overlapping jobs a *batch* [11]. Thus a session may contain several batches in sequence, and each batch may contain a number of jobs. The interval between batches is called the *think-time*, or TT.

Finding the batches and the sessions of the users is a basic requirement in order to understand and analyze their behavior. However, activity logs do not contain explicit information about neither the sessions nor the batches. Therefore, we need to estimate them based on the data that the logs do contain. The most relevant information is the job arrival times (also called submit times) and the job end times. For job i, we will denote these as $J[i].arr$ and $J[i].end$.

2.1 Definitions Based on Think Times

Assume we scan the jobs in a log one by one. As each job is considered, the question is whether it belongs to the previous session or batch, or starts a new session or batch. The simplest and most commonly used approach makes this decision based on the think time, namely the interval from the termination of one job to the submittal of the next[1]:

1. If the think time is negative, the job overlaps the current batch and therefore belongs to this batch.
2. If the think time is positive but below the *session threshold*, the job starts a new batch in the same session as the previous batch. (We discuss the value of the session threshold in Section 3.)
3. Otherwise, the job starts a new batch in a new session.

Note, however, that we need to be precise regarding how we measure the think time, and in particular, exactly what job end time do we use as a reference point. There are two possibilities:

– The end time of the last job that was submitted. With this approach, the think time of job i will be calculated as

$$TT_{Last} = J[i].arr - J[i-1].end$$

Hereafter we denote this approach by Last.

– The maximal end time among all previous jobs. In this case, the think time is calculated as

$$TT_{Max} = J[i].arr - \max_{j<i} J[j].end$$

This approach will be denoted by Max.

[1] Recall that the conceptual model is that the user submits a job, waits for it to terminate, and then thinks about the result before submitting the next job.

To appreciate the difference, consider a sequence of 3 jobs. Job 1 is very long. Job 2 is short and ends much before job 1 ends. Job 3 arrives after job 2 ended, but still overlaps job 1. In this situation all 3 jobs will be in the same batch based on Max, but job 3 will start a new batch based on Last. This is illustrated in Figure 1.

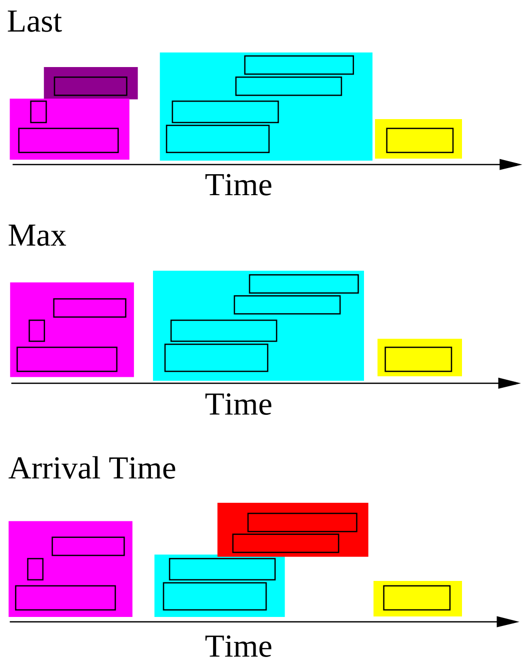


Fig. 1. Batches according to the three approaches: Last, Max, and Arrival.

2.2 Definition Based on Inter-Arrival Times

Another approach to define sessions is according to the arrival times of the jobs, or rather, the inter-arrivals (to be denoted by Arrival). In this approach, the current job would belong to the same session as previous jobs if it arrives up to the session threshold time after the arrival of the previous job in the session. In other words, if the inter-arrival time is longer than the session threshold, we decide that this represents a session break. Once the jobs are partitioned into sessions using this approach, we partition each such session into batches according to the Max approach.

An example showing the effect of this procedure is shown in Figure 1. The four jobs in the middle all overlap, and are considered to be the same batch by both Last and Max. But there is a relatively large gap between the arrival of the first pair and the arrival of the second pair. If this gap is bigger than the session threshold, the two pairs will be in different sessions according to Arrival, and as a result also in different batches.

The reason for using Max to partition a session into batches is as follows. Consider how the end of a batch is defined. If batch A comes a certain TT after batch B according to Max, then it will start only TT time after *all* the jobs in B are finished. But according to Last, it will start TT time after the last job in B has finished, while other jobs from B may still be running. Shmueli indeed used the last job as the critical one [13]. However, this definition is problematic, because it means that the future activity of the user depends only on the last job in each batch, while the other jobs don't effect the future activity at all. This seems very unrealistic. A simple example of the problem is that it is very easy to create a scheduler that reduces both the user's wait-times and the overall system utilization by running the last job of each user last, thereby causing the user to wait a long time before submitting more jobs. Alternatively, one can construct a scheduler that would increase both the wait-times and the utilization by handling the last job of each user first. To avoid such problems, we prefer Max.

We note that Max creates a sequence of batches with no overlaps. In Last they may overlap, but the dependencies between batches are still a linear sequence. In Arrival a batch may depend on multiple earlier batches.

In the area of parallel supercomputer workloads, the common way to define sessions uses the end time (meaning Last or Max). For example Zilber et al. and Shmueli use Last [16, 12]. But in other areas, where job durations are extremely short, it is more common to define sessions based on arrivals. An example is interactive web use (surfing, searching, or e-commerce) [1, 2, 4, 7, 8, 10, 15]. Of course, due to the very short time it typically takes to process a request on the web, requests never overlap. Therefore Last, Max, and Arrival are actually equivalent in this case.

In the next sections we will discuss the session threshold for each approach and the influence of the choice of this unique value. Additionally, we will present a comparison between the Max and Last approaches. Later, we will investigate the session lengths produced by the different approaches, and conclude that Arrival is the best approach to use.

3 Selecting a Session Threshold Value

The dominant methodology to extract session data from activity logs is to postulate a certain threshold value, and assume that breaks in activity which are longer than this threshold represent a division between separate sessions. Such a threshold exists in all three approaches: Last, Max, and Arrival. The main difference between these definitions is the time interval that we compare to the threshold. In Arrival this interval starts at the arrival of the last job, in Last at the end of the last job, and in Max at the maximal end time among previous jobs. The threshold value that is chosen may have a strong effect on the resulting session properties [1]. In this section we will consider how to select the threshold value for each approach, and consider its influence on the sessions.

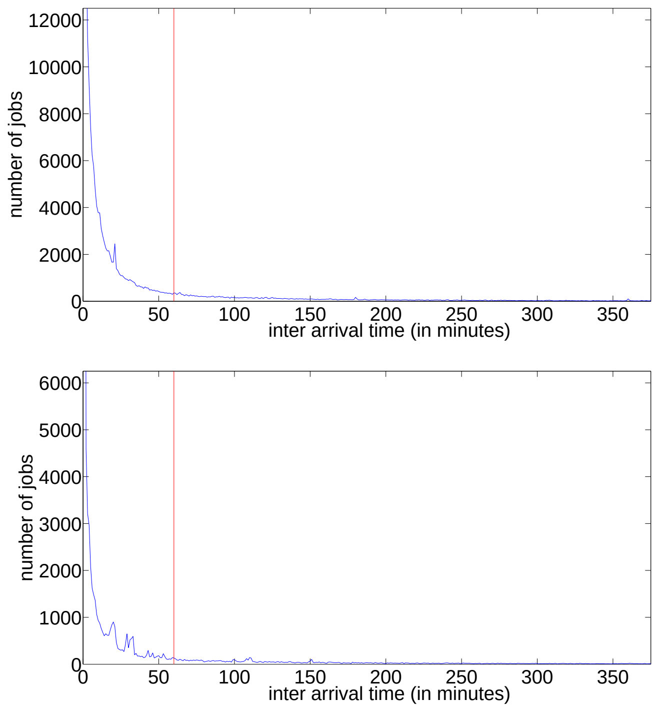


Fig. 2. The distribution of inter-arrival time in the SDSC-BLUE and SDSC-DS logs.

As mentioned above, Last and Max are both popular approaches in this area. Therefore, many previous works have considered the selection of the the threshold value for them. The commonly used value is 20 minutes, because this seems to capture the majority of think times. For example, Zilber et al. and Shmueli [16, 12] used this threshold value.

As far as we know, there has been no previous work concerning the selection of a threshold on inter-arrival times for parallel workloads. Several different values have been used in the context of web workloads, including 30 minutes [2, 7], an hour [14], and even two hours [8]. To find what value would make a suitable threshold for our parallel workloads, we calculated the distribution of inter-arrival times for different logs available from the Parallel Workloads Archive [9]. Thus, for each user we found the difference between the arrival times of each pair of successive jobs. We ignored values that were above a day (1440 minutes), because such long intervals obviously defy the notion of a single session. Examples of the resulting distributions are shown in Figure 2. CDFs[2]

[2] The Cumulative Distribution Function (CDF) is the integral of the probability density function (pdf). For each value x, it gives the probability of observing values that are smaller than or equal to x. In the case of empirical data, it is the fraction of samples that are smaller than or equal to x.

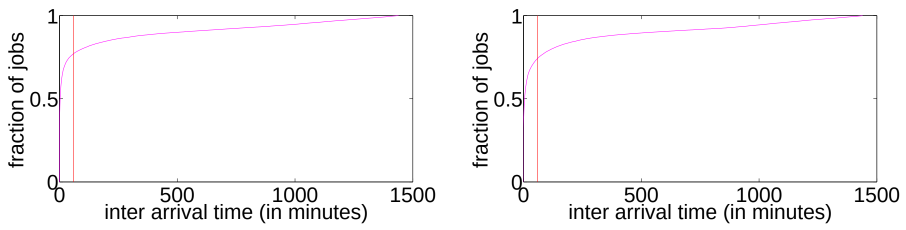


Fig. 3. CDFs of inter-arrival times in the SDSC-SP2 and KTH-SP2 logs.

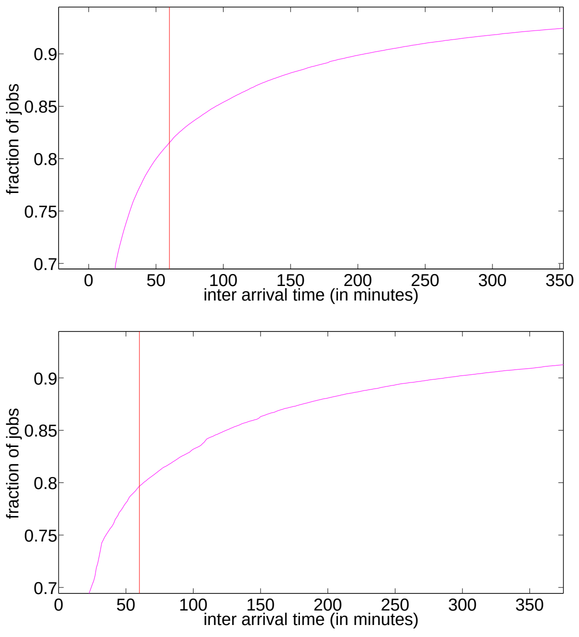


Fig. 4. Zoom in on the CDFs of inter-arrival times for the SDSC-BLUE and SDSC-DS logs.

are shown in Figure 3 and Figure 4. In all these figures we added a vertical line at 60 minutes (1 hour), which is the threshold we eventually chose.

Our goal was to find the point in the distribution where the derivative doesn't changed much any more. From Figure 2 it appears that any value between approximately 25 minutes and 200 minutes will be logical. However, for values below 40 one can still observe an obvious drop in the distribution. This is even clearer in the CDF (and especially in the figures with zoom in). In the range of 100 to 200 minutes the slope is already very low, and therefore we would prefer a lower value for the threshold. We concluded that the value ought to be between 40 minutes to 100 minutes. We chose 60 minutes as it is in the middle of this range and is a round value (one hour). We do not claim this is necessarily the best value, but it seems that there is no other value that is obviously better.

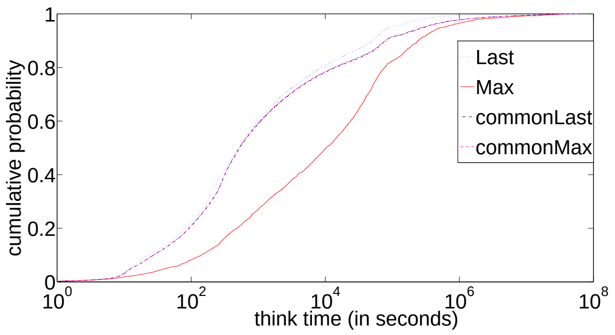
Selecting a session threshold has a strong impact on the resulting analysis. If we were to select a higher threshold, jobs with longer intervals will nevertheless be grouped together. As a result the number of jobs in each session would grow and the number of sessions would decrease. In the following sections we provide an in-depth analysis of the implications of the selected session threshold values, mainly in terms of the distribution of session lengths.

4 Comparing Last and Max Using the Think-Time Distribution

As we mentioned above, the most common definition of sessions is based on think times, using Last or Max. In this section we investigate which definition leads to a more reasonable think time distribution. The problem is that we do not know what the think time distribution should be. We circumvent this problem as follows. First, we identify the batches according to both approaches separately, and calculate the think times. Then we create a list that contains the common batches (batches that are exactly the same according to both approaches). Based on this list, we extract the think times following these common batches. This provides us with two lists of think times: *CommonTTMax* and *CommonTTLast*. Note that the common batch think times may be different because the think times are defined differently in each approach. In Last, the think time is measured from the end of the last job, whereas in Max it is the maximal end time of all jobs. But we expect the distributions to be close, which indeed they are.

Given the agreement on the common batches, and the similarity of their think time distributions, we take this to represent the "real" distribution of think times. The remaining think times, that we didn't put in the common lists, represent the differences between think times of batches that were created according to Last and Max. Therefore, we would prefer the approach for which the distribution of unique think times is similar to the distribution of common think times.

The resulting distributions are shown in Figure 5. The first obvious conclusion from the graphs is that our expectation that the distributions for common batches shared by Max and Last would be very close to one another was correct. It is also quite clear that the distribution for unique batches as identified by Last is much closer to the common distributions than the distribution for unique batches as defined by Max. It is true that the distribution for Max is closer for larger values, but in most of the range, Last is a lot closer. We concluded that Last creates a more realistic distribution of think times. This supports the use of Last by Zilber et al. [16], Shmueli [12], and others.



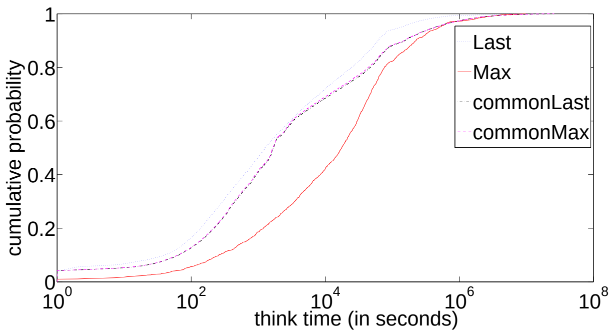


Fig. 5. Comparison between distributions of think times in the SDSC-BLUE and SDSC-DS logs.

5 Artifacts in the Distribution of Session Lengths with Arrival

According to the work of Mehrzadi, using the Arrival approach may lead to artifacts in the distribution of session durations [6]. Specifically, he shows that in web search data the distribution of session lengths exhibits a pronounced drop exactly at the threshold value that was used to define the sessions. In order to check this, we examined the distribution of session durations for each of the three approaches. Due to the large number of very short sessions, we ignore sessions of up to 2 minutes. The results are shown in Figure Figure 6 for Last, Figure 7 for Max, and 8 for Arrival.

Upon examination of the graphs, we found that Last and Max behave very similarly, but Arrival is indeed different. According to the Max and Last approaches, the distribution of session lengths is essentially the same for different threshold values. The difference in heights is due to the fact that larger thresholds lead to a smaller number of sessions, but the behavior of each graph is the same. In addition, for all values, there are no obvious discontinuities.

In contrast, with the Arrival approach it is easy to notice a sharp drop in the distribution at the threshold value, exactly as had occurred in Mehrzadi's

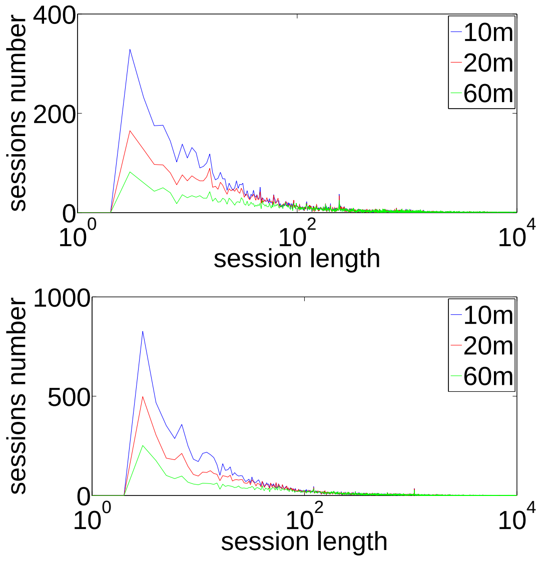


Fig. 6. Distribution of session lengths as created by Last, for the KTH-SP2 and SDSC-SP2 logs.

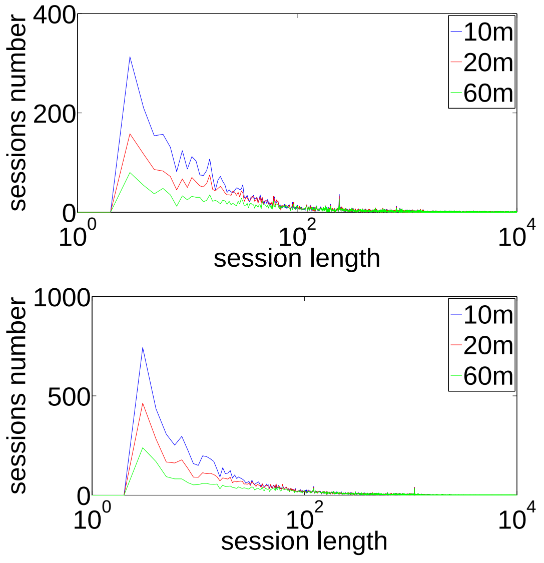


Fig. 7. Distribution of session lengths as created by Max, for the KTH-SP2 and SDSC-SP2 logs.

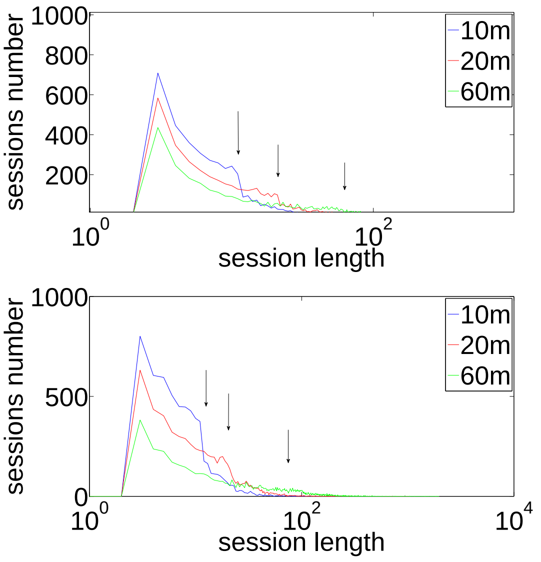


Fig. 8. Distribution of session lengths as created by Arrival, for the KTH-SP2 and SDSC-SP2 logs. Arrows denote threshold values.

data. In particular, each specific threshold value changes the distribution of session lengths in a different way (see arrows). However, this effect is reduced when we increase the value of the threshold. This is more evident in Figure 9. In this figure we ignored all session lengths below 9 minutes, and present the histogram without any connecting lines for clarity. With a 10 minute threshold the discontinuity is very significant. For 20 minutes the discontinuity is smaller (but still noticeable). With 60 minutes the drop becomes a step. It is worth mentioning that in some logs (although not in most) there is a clearer drop for 60 minutes, yet rather less dramatic than for 20.

In conclusion, we find that the Arrival approach is sensitive to the threshold value, although with large values (like the one we chose) the effect is rather small.

6 The Problem of Very Long Sessions

The graphs in Figures 6 and 7 show that with Last and Max most sessions are short, and few sessions are very long, possibly unrealistically so. However, it is impossible to see the details. In order to emphasize the long sessions we calculated the survival function[3], and present the results in Figure 10. This shows that when using the Last approach, approximately 13.5% of the session

[3] The survival function is the complement to the CDF: for each value x, it gives the probability of observing values that are larger than x.

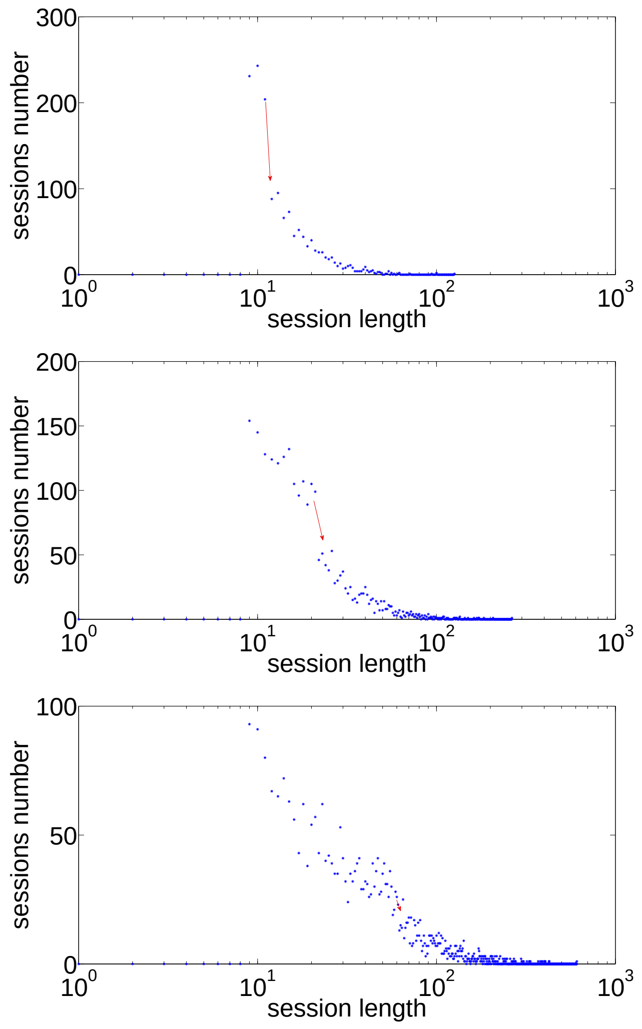
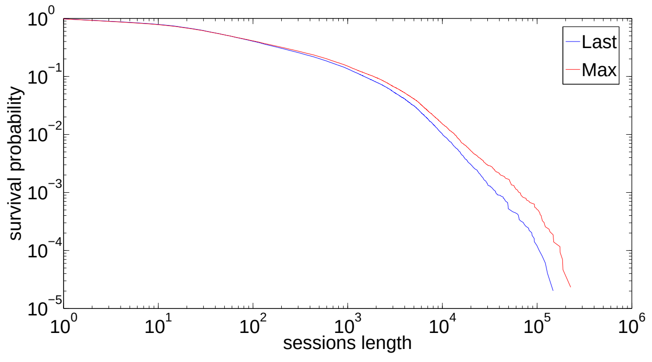


Fig. 9. Detailed view of discontinuities in the histogram of session lengths, using the Arrival approach, with thresholds of 10m (top), 20m (middle), and 60m (bottom), for the KTH-SP2 log.

lengths are longer than 10^3 minutes (16 hours) in the SDSC-BLUE log, and 17.7% are longer than this value in the SDSC-DS log. With the Max approach, the percentages are a little higher: 15% in SDSC-BLUE and 19.5% in SDSC-SP2. In addition, one may notice that the maximum session length is above 10^5 minutes (approximately 70 days) in both logs.

Recall that a session is supposed to represent the time period when the user is active at the computer (the interval from when the user begins to work and until



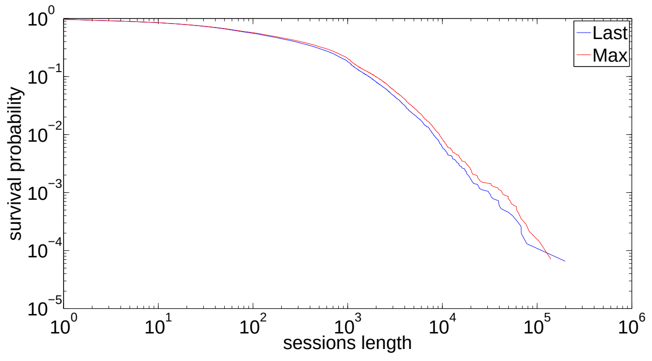


Fig. 10. The survival function of session lengths in the SDSC-BLUE and SDSC-SP2 logs, using log-log axes.

he is done working). Therefore, session lengths above 10^4 minutes are impossible. In addition, even sessions of 16 hours are not reasonable. It should be very rare that a user would work this long continuously, yet still the results show that more than 13% of the sessions are that long. This is very unlikely.

The reason both approaches create sessions that are far too long so many times is that both are based on a wrong assumption. This is the assumption that all work is interactive. With interactive work, it is reasonable to assume that the users wait for the termination of each job, think for a while, and then send the next jobs. But on parallel supercomputers at least some of the work is not interactive. In particular, this is the case for very long jobs that run for many hours or even days. Including these very long jobs within the session, as is done by both Last and Max, then leads to unrealistically long sessions. For example, if a user sends out a job that takes 5 days, and after 3 days sends another job to the system, both approaches will put these two jobs into the same session, although the user most probably wasn't active in the system this whole time. The same problem may also occur on a smaller scale of a hew hours. If there are jobs that run during a break in the user's activity in the middle of the day (for example, during meetings or lunch), these jobs may overlap new work done after the user returns. Therefore, instead of a few short sessions of a couple of hours scattered along the day, we would get one long session — from the first job the user submits in the morning until after he goes home at night.

In order to avoid such problems, we suggest alternative versions of Last and Max which we call Last+Cut and Max+Cut. In these versions we define a new threshold value, called the *Cut*. Then, we use each job's arrival time plus Cut as its effective end time, instead of using the real end time, provided it is shorter. This means that if a job ends within Cut time from its arrival, we measure the think time from its end time without change. Otherwise, we use its arrival time + Cut as the start of the think time. Assuming a session threshold of T minutes we then have:

– Last+Cut: A job will belong to the current batch if it arrives before the arrival time of the last job + Cut + session-threshold (or the end time + session-threshold):

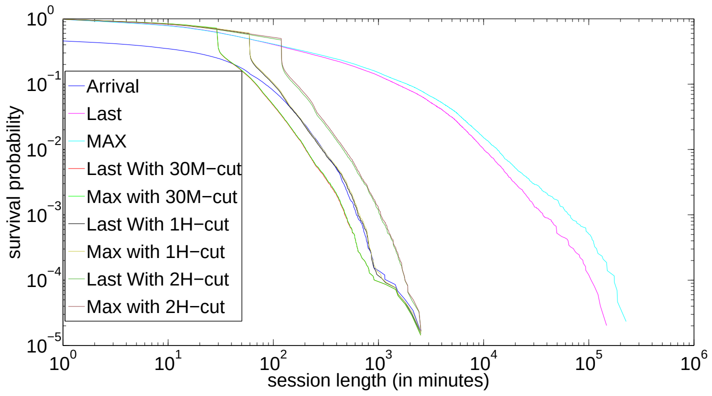
$$J[i].arr \leq \min\{\ J[i-1].end,\ J[i-1].arr + Cut\ \} + T$$

– Max+Cut: A job will belong to the current batch if it arrives before the maximum of the arrival times of all the jobs in the batch + Cut + session-threshold (or with end times):

$$J[i].arr \leq \max_{j<i}[\ \min\{\ J[j].end,\ J[j].arr + Cut\ \}\ ] + T$$

The results of using these approaches are shown in Figure 11. We checked three different values for Cut: 30 minutes, 1 hour, and 2 hours. (Last, Max, and Arrival are also included for comparison.) As expected, in all 3 cases the problem of overly long sessions is largely eliminated. Also, the difference between Max+Cut and Last+Cut with the same threshold is very marginal. Therefore we will distinguish between the Cut approaches only according to the threshold. In the SDSC-BLUE log, the fraction of sessions longer than 10^3 is a little more than 10^{-3} with a large Cut value of 2 hours, but with 1 hour or 30 minutes this fraction is only a little higher than 10^{-4} (approximately $10^{-3.9}$). In the SDSC-SP2 log, this fraction is approximately $10^{-3.4}$ with a 2 hours Cut, $10^{-3.7}$ with 1 hour, and $10^{-3.9}$ with 30 minutes. The value of the maximum session length is also dramatically decreased in the Cut approaches: down to 2600 minutes (43 hours) in the SDSC-BLUE log and less than 2000 minutes (33 hours) in the SDSC-SP2 log.

The conclusion is that the Cut approach creates more realistic session lengths. The longest sessions still seem to be too long, lasting nearly 2 days, but still this is much better than the sessions that last for more than 2 months we had before. While unreasonable for humans, such long sessions may be due to a short script or a number of people who might have replaced each other on the computer, sending the jobs through the same user name. In addition, the percentage of long sessions has dropped. Only a very small percentage of the sessions were more than 1000 minutes (16 hours) long, in comparison to 13% or more with the original Last and Max.



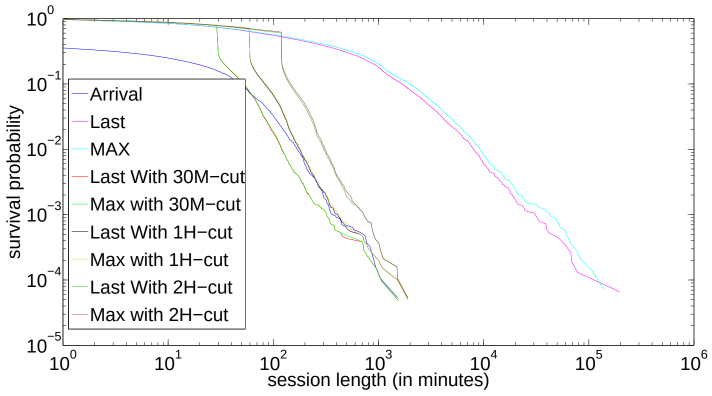


Fig. 11. The survival function of session lengths according to all the different approaches, for the SDSC-BLUE log and the SDSC-SP2 log.

However, although the length of the sessions in the Cut approaches are more realistic, the effect of the Cut value on the distribution is enormous: There is a very sharp drop in the graph at the point of the Cut value. In order to examine this effect, we created histograms of the session lengths generated by Last+Cut and Max+Cut. These are presented in Figure 12. It is easy to see that the Cut values produce a very significant mode in the distributions. The reason for these modes is as follows. For all the sessions with one job, if the job ends before the Cut value, the length will be the end time minus the arrival time. This part of the distribution will be continuous. But if the job ends after the Cut Value, the length will be the equal to the Cut value. Therefore, many sessions will receive the Cut value length.

The bottom line is that Last and Max remain problematic. In the original version, they create sessions that are way too long. Introducing the Cut heuristic leads to a strong artifact in the distributions of session lengths. Hence, the only logical approach is to use the Arrival approach. It is equivalent to the Cut ap-

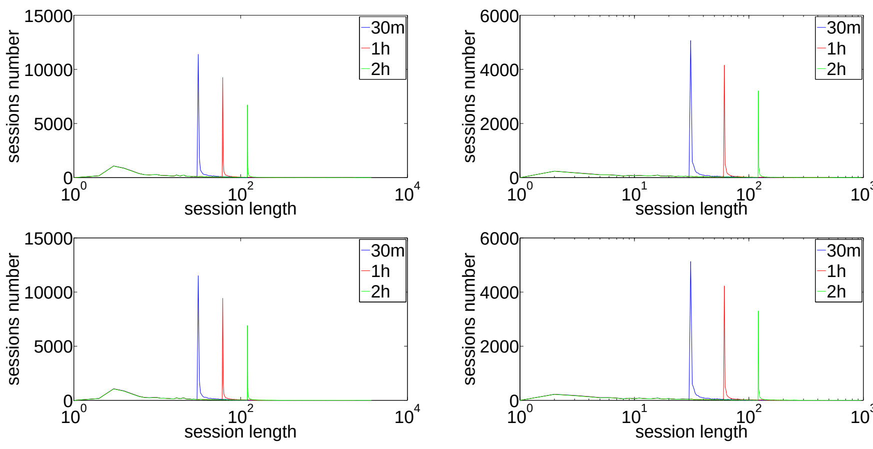


Fig. 12. Histograms of session lengths generated by Last+Cut (top) and Max+Cut (bottom) using the SDSC-DS and KTH-SP2 logs (left and right).

proach, where Cut=0, and with a larger session-threshold (60m instead of 20m). (Note that if the Cut value is 0, then Max+Cut is equivalent to Last+Cut.)

The Arrival approach produces realistic session lengths similar to the Cut approaches, But in addition, the distribution is smooth with no modes that depend on parameter values. Therefore, it seems that this approach creates the most sensible distribution of session lengths. We conclude that the Arrival approach, especially with a relatively long session threshold of 1 hour, is the most promising approach to delimit sessions.

7 Results with the Arrival Approach

Due to the fact that it is innovative and uncommon to use the Arrival approach to define sessions in parallel workloads, we present a few details and distributions of sessions and batches.

First, we present the number of jobs, batches, and sessions in Table 1. In all of the logs the ratios are very similar. On average, the number of jobs is a little less than twice the number of batches, and the number of batches is a little less

Log	Jobs	Batches	Sessions
SDSC-SP2	54,051	32,614	18,730
SDSC-DS	85,003	41,679	24,294
KTH-SP2	28,489	16,488	10,303
SDSC-BLUE	223,407	136,460	58,311

Table 1. Number of jobs, batches, and sessions in the main logs.

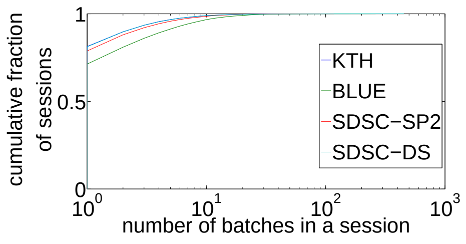


Fig. 13. CDF of the number of batches in a session.

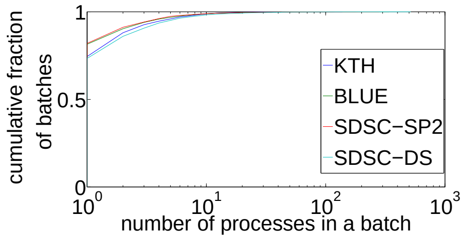


Fig. 14. CDF of the number of jobs in a batch.

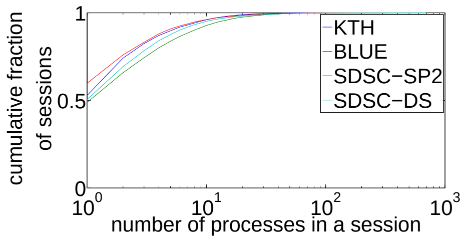


Fig. 15. CDF of the number of jobs in a session.

than twice the number of sessions. Additional data on batches and sessions are presented in Figure 13, Figure 14, and Figure 15. A very important observation is that generally more than 50% of the sessions and 75% of the batches contain only one job. This means that when users work with supercomputers, most of the time they send out a single job and then stop their interaction with the computer for a while. However, it is important to note that some sessions have very many jobs, so the distribution is skewed, and most jobs do not constitute single-job sessions.

8 Conclusions

A summary of the methods that can be used to identify sessions when analyzing parallel workloads is given in Table 2.

Approach	*Issues*
Last, Max	excessively long sessions
Last+Cut, Max+Cut	strong peak at cut value
Arrival	peak at threshold value many zero-length sessions

Table 2. *Summary of approaches and their effect on the session length distribution.*

The most common approach is to use the Last and Max approaches. These approaches are based on setting a threshold on think times: if the think time is long, this is assumed to be a break between sessions. However, these approaches occasionally cause extremely long sessions, due to the fact that some of the jobs running on such systems are extremely long.

A possible improvement is to use Last+Cut or Max+Cut. This eliminates the very long sessions, at the price of producing a strong peak in the distribution of session length at the value of the cut threshold being used. This is also undesirable.

The alternative is to use the Arrival approach, as in commonly done in other domains, such as the analysis of web workloads. In this approach, inter-arrival times are used. If the inter-arrival is longer than some threshold, a session break is assumed. The main problem with this approach is that long sessions may not be identified correctly, and again a peak in the distribution is created at the value of the threshold being used. However, the size of this peak decreases with increasing threshold values. We suggest to use a threshold of 1 hour. With such a threshold the peak in the distribution of session lengths is very small.

The obvious deficiency with the above is that it is based on common sense, not on data. A desirable avenue for future work is therefore to conduct a user study in which the actual activity patterns of users are followed, and this is correlated with their job submittal patterns.

Acknowledgments

Many thanks to all those who have made their workload data available through the Parallel Workloads Archive.

References

1. M. Arlitt, "*Characterizing web user sessions*". *Performance Evaluation Rev.* **28(2)**, pp. 50–56, Sep 2000.
2. D. Downey, S. Dumais, and E. Horvitz, "*Models of searching and browsing: Languages, studies, and applications*". In 20th *Intl. Joint Conf. Artificial Intelligence*, pp. 1465–1472, Jan 2007.

3. J. Jann, P. Pattnaik, H. Franke, F. Wang, J. Skovira, and J. Riodan, "*Modeling of workload in MPPs*". In *Job Scheduling Strategies for Parallel Processing*, pp. 95–116, Springer Verlag, 1997. Lect. Notes Comput. Sci. vol. 1291.
4. B. J. Jansen, A. Spink, C. Blakely, and S. Koshman, "*Defining a session on web search engines*". *J. Am. Soc. Inf. Sci. & Tech.* **58(6)**, pp. 862–871, Apr 2007.
5. U. Lublin and D. G. Feitelson, "*The workload on parallel supercomputers: Modeling the characteristics of rigid jobs*". *J. Parallel & Distributed Comput.* **63(11)**, pp. 1105–1122, Nov 2003.
6. D. Mehrzadi and D. G. Feitelson, "*On extracting session data from activity logs*". In 5th *Intl. Syst. & Storage Conf.*, Jun 2012.
7. D. A. Menascé, V. A. F. Almeida, R. Riedi, F. Ribeiro, R. Fonseca, and W. Meira Jr., "*A hierarchical and multiscale approach to analyze E-business workloads*". *Performance Evaluation* **54(1)**, pp. 33–57, Sep 2003.
8. A. L. Montgomery and C. Faloutsos, "*Identifying web browsing trends and patterns*". *Computer* **34(7)**, pp. 94–95, Jul 2001.
9. "*Parallel workloads archive*". URL http://www.cs.huji.ac.il/labs/parallel/workload/.
10. 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.
11. 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.
12. 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.
13. 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.
14. E. Shriver and M. Hansen, *Search Session Extraction: A User Model of Searching.* Tech. rep., Bell Labs, Jan 2002.
15. C. Silverstein, M. Henzinger, H. Marais, and M. Moricz, "*Analysis of a very large web search engine query log*". *SIGIR Forum* **33(1)**, pp. 6–12, Fall 1999.
16. J. Zilber, O. Amit, and D. Talby, "*What is worth learning from parallel workloads? a user and session based analysis*". In 19th *Intl. Conf. Supercomputing*, pp. 377–386, Jun 2005.