# Workload Sanitation for Performance Evaluation

Dror G. Feitelson Dan Tsafrir
School of Computer Science and Engineering
The Hebrew University, 91904 Jerusalem, Israel

## Abstract

*The performance of computer systems depends, among other things, on the workload. Performance evaluations are therefore often done using logs of workloads on current productions systems, under the assumption that such real workloads are representative and reliable; likewise, workload modeling is typically based on real workloads. We show, however, that real workloads may also contain anomalies that make them non-representative and unreliable. This is a special case of multi-class workloads, where one class is the "real" workload which we wish to use in the evaluation, and the other class contaminates the log with "bogus" data. We provide several examples of this situation, including a previously unrecognized type of anomaly we call "workload flurries": surges of activity with a repetitive nature, caused by a single user, that dominate the workload for a relatively short period. Using a workload with such anomalies in effect emphasizes rare and unique events (e.g. occurring for a few days out of two years of logged data), and risks optimizing the design decision for the anomalous workload at the expense of the normal workload. Thus we claim that such anomalies should be removed from the workload before it is used in evaluations, and that ignoring them is actually an unjustifiable approach.*

## 1. Introduction

The performance of a computer system depends not only on its design and implementation, but also on the workload to which it is subjected [9]. Different workloads may lead to different absolute performance numbers, and in some cases to different relative ranking of systems or designs. Using representative workloads is therefore crucial in order to obtain reliable performance evaluation results.

The need to use representative workloads for system evaluations has long been recognized, and has led to the practice of monitoring existing systems and characterizing their workloads for this purpose [11, 1, 6]. If current systems are available that have a similar functionality to the system being evaluated, one can assume that the same workloads may apply. One can then record the workloads on the current systems, and play back these recordings to drive a simulation of the new system. Alternatively, the recorded workloads can be used as the basis for constructing a workload model (e.g. [13, 8, 16]). This has the benefit of allowing for more flexible usage, e.g. by modifying model parameters so as to adapt it to different system configurations.

However, using recorded workloads also has its problems. Modeling is typically done by collecting workload traces, and creating a statistical model based on fitting the distributions of workload attributes [14]. But such an approach is questionable if the data is not stationary. For example, Chiang et al. analyze six non-consecutive months of data from the NCSA Origin 2000 machine, and find that the workloads in different months may be quite different from each other [7]. It is also well known that workloads at different installations differ, and that workloads evolve with time as users learn to better use the system [12].

This paper deals with another type of drawback: real workloads may be multiclass, meaning that they are a mixture of several different types of workload, some of which are undesirable. In particular, undesirable classes include abnormal behaviors that, though they do in fact occur on rare occasions, are not representative in general.

Previous work on computer workloads has focused on analyzing the workload as a whole, and trying to determine whether it is stationary and representative. We suggest a complementary approach in which the data is "cleaned up" by removing the non-representative or undesirable parts, and only retaining part of the data for use in the evaluation. This is not as far fetched as it may sound initially: for example, removing outliers from data is a common practice in many statistical analyses.

The following sections present several examples of multiclass workloads that we have encountered, from systems spanning large-scale parallel supercomputers to Unix servers staff at our department. These examples demonstrate the wide variety of different anomalous behaviors that can appear in computer workloads. Regrettably, this means that it is doubtful all such behaviors will ever be recognized automatically. Our work therefore serves mainly to raise awareness to the issue of workload sanitation. However, we also provide important tips and observations regarding how to go about spotting anomalous behavior in newly recorded workloads.

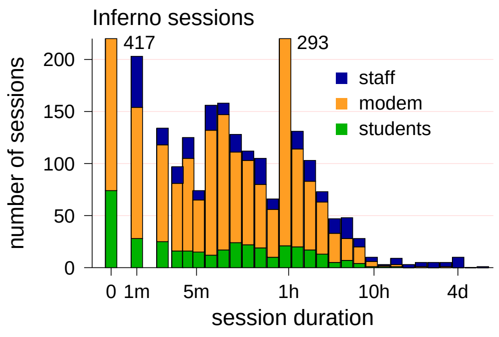

**Figure 1.** *The distribution of session durations for different classes of users. Only staff who have dedicated workstations generate sessions longer than 10 hours. (Data truncated at 250 sessions; there are more sessions of 0 or 1 hour.)*

## 2. Multiclass Workload Mixtures

Even without errors and abnormal events, it is often necessary to filter the collected data. One example of such a situation is when the workload is actually composed of multiple classes of workloads that have been merged together. If we want to model or use just one of these classes, we need to filter out the rest.

A common type of multiclass workload is the combination of prime-time and non-prime-time workloads. Most users are active during normal work hours: say 9 to 5 on weekdays. This is when most of the workload is generated. During the night, on weekends, and on holidays, much less work is submitted. In fact, it is not only the number of jobs that changes from prime-time to non-prime time; the characteristics of the jobs may also be different. For example, in systems that employ batch queueing systems, it is common to have different queues active at different times of day [10]. During working hours, only queues for relatively short jobs are active. But at night, queues that allow longer jobs become active. As a result, many long jobs tend to start execution when the system shifts to non-prime mode.

The effect of the prime/non-prime dichotomy depends on our goals. If we are interested in modeling the characteristics of prime-time load, we should focus on prime-time data, and filter out data that is generated at other times. Conversely, if we are interested in non-prime load, the data relating to prime time should be filtered out. If we want a full workload, we should include both, and in particular, retain the daily cycle.

As another example, consider the modeling of interactive user sessions. Data can be collected from a Unix server using the `last` command, which lists all user logins and the lengths of the ensuing sessions. Fig. 1 shows the results of doing so on a shared server called “inferno” in our department. The data contains many short and intermediate sessions, and a few extremely long ones. The longest sessions seem impossible, as they extend for several days. This mystery is solved by noting that the workload is actually composed of three classes:

1. Students working from student terminal labs. These sessions are of limited duration because of rules regulating the sharing of the terminals, and also because students need to leave the terminals to eat and sleep.
2. People accessing the server via remote dial-up networking. In this case sessions are automatically terminated if no activity has occurred for a certain amount of time, again limiting the maximal length of sessions.
3. Staff members working from their offices. As staff have private terminals that are connected to the server by a LAN, there is no a-priori limit on the length of sessions for this class of users.

To summarize, the first two classes logoff immediately when they complete their work, and are responsible for the bulk of the data. This is the data that truly characterizes interactive sessions. The third class, that of university staff, simply open windows to the server on their desktop machines, and leave them open for days on end. This data is not representative of interactive work, and should be filtered out.

Incidentally, it may also be the case that this dataset should be cleaned due to non-representative items. In particular, the large number of 0-length and 1-hour modem sessions may reflect connection problems and automatic termination of non-active sessions, respectively, rather than the real lengths of user sessions. Notably, the interactive student sessions do not have these modes.

## 3. System Personnel Activity

Large-scale systems often require continuous support from the vendors that installed them. In some cases, a vendor employee is even stationed at the installation site, so as to be on hand in case of need. Such employees also perform monitoring tasks and take preventative measures to avert failures before they happen.

Given the presence of such system staff, the workload observed on the system is actually a mixture of two classes of workload: work submitted by real users (the systems “payload”), and work submitted by the system personnel as part of performing their tasks. As noted above, what we do about this depends on our goals. If we are only interested in user activity, system staff activity should be filtered out. But if we are interested in the complete system, then monitoring and maintenance tasks should be left in, because they are indeed part of what the system has to do.

However, sometimes system staff generate extraneous workload that is obviously bogus. One striking example was reported in the analysis of the workload on the NASA Ames iPSC/860 hypercube [10]. The histogram of job sizes on that 128-node machine indicated that more than half of the jobs were serial; moreover, most of the serial jobs were flagged

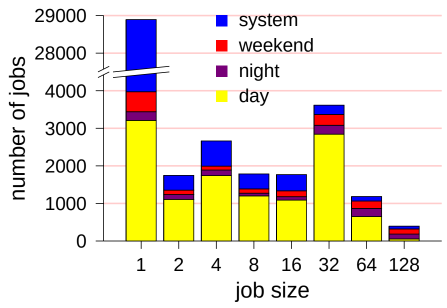

**Figure 2.** *Histogram of job sizes from the NASA Ames iPSC/860, showing abnormally many single-node system jobs.*

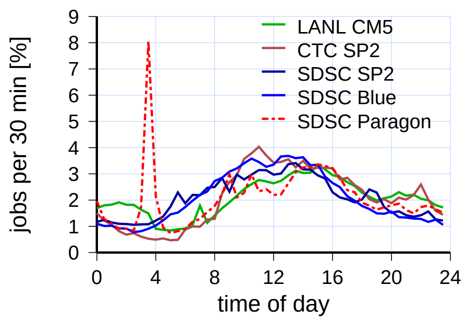

**Figure 3.** *Daily arrival pattern on 5 parallel supercomputers, showing abnormal spike at 3:30 AM on the SDSC Paragon.*

as being run by the system support staff (Fig. 2). This turned out to be a result of an ad-hoc method used to verify that the system was operational and responsive by running the Unix `pwd` command on a single node. Overall, a full 56.8% of the trace (24025 jobs) were such check-runs. This type of activity was not observed on any other parallel system. Thus it is quite obvious that these jobs should be removed if the trace is used to analyze parallel workloads or to evaluate parallel job schedulers.

## 4. Robot Activity

Another example is shown in Fig. 3. This compares the daily arrival cycle on 5 different parallel supercomputers. All display the expected periodic behavior, with load peaking during work hours and lower loads at night. But the SDSC Paragon machine has an additional and much higher peak between 3:30 and 4:00 AM. Upon investigation, it turned out that a set of 16 jobs with a distinct profile was executed during this time slice every day. While specific information is not available, it is reasonable to assume that these jobs served some system administration function and were executed automatically. It is again obvious that they should be removed when using the log for evaluations, so as to reduce the danger of optimizing for this abnormal behavior.

Robot activity, defined to be the automatic execution of programs, also occurs on desktops and departmental servers. Many enterprise networks use automatic scripts to update the software on all the machines, typically at night when this activity will not cause undue interference with user work. Backups are also often done in this way. A salient feature of this type of activity is its predictability: the scripts always run at the same time, and do the same thing.

An extreme example is shown in Fig. 4. This shows the activity at a departmental server called “pita” during 3 days towards the end of 1998 (the full dataset covers about 2 weeks). Remarkably, this includes two distinct types of robot activity, which together constitute a large fraction of the workload. Thus the complete workload actually has three classes:

1. A set of over 400 jobs submitted automatically by a root script each morning at 04:15AM. These are all Unix `rm` commands, most probably issued by a maintenance script as noted above.
2. Massive activity by user 1301, starting in the evening hours of December 22. Upon closer inspection, this was found to consist of iterations among half a dozen jobs, totaling about 200, that are submitted every 10 minutes. In subsequent days, this started each day at 6:40AM, and continued until 2:50AM the next day.
3. Normal behavior by other users, with its fluctuations and daily cycle.

A different type of robot activity occurs on the web. Here the term “robot” refers to software agents that roam the web and search for specific information. In some cases, this is just all the information that is available at a site, and it is collected for indexing purposes. Such robot activity may account for a large fraction of a site’s activity. But in most cases this is considered legitimate activity that the site should support, and therefore it should not be filtered out indiscriminatingly. Rather, it can be modeled as a separate class of workload, as we suggest below.

## 5. Singular Activity (Workload Flurries)

Many of the classes of workload mentioned in the previous sections have been noted already in the past. In this section we introduce a new type of workload pattern which we believe has not been identified before. We call these patterns “workload flurries”.

We define a workload flurry to be a pattern of activity with the following characteristics:

1. It causes a level of activity that is significantly higher than usual, thus dominating the workload
2. It exists for a limited period of time
3. It significantly changes the distributions of workload attributes
4. It is caused by a single user.

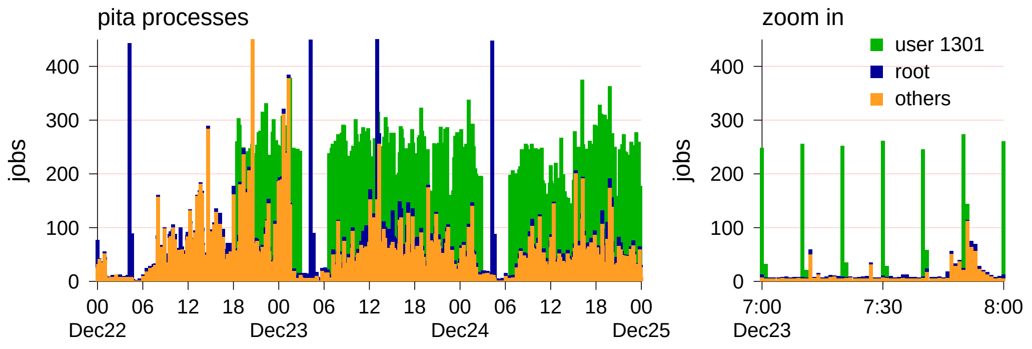

**Figure 4.** *Arrival pattern to a departmental server showing two distinct robot activities.*

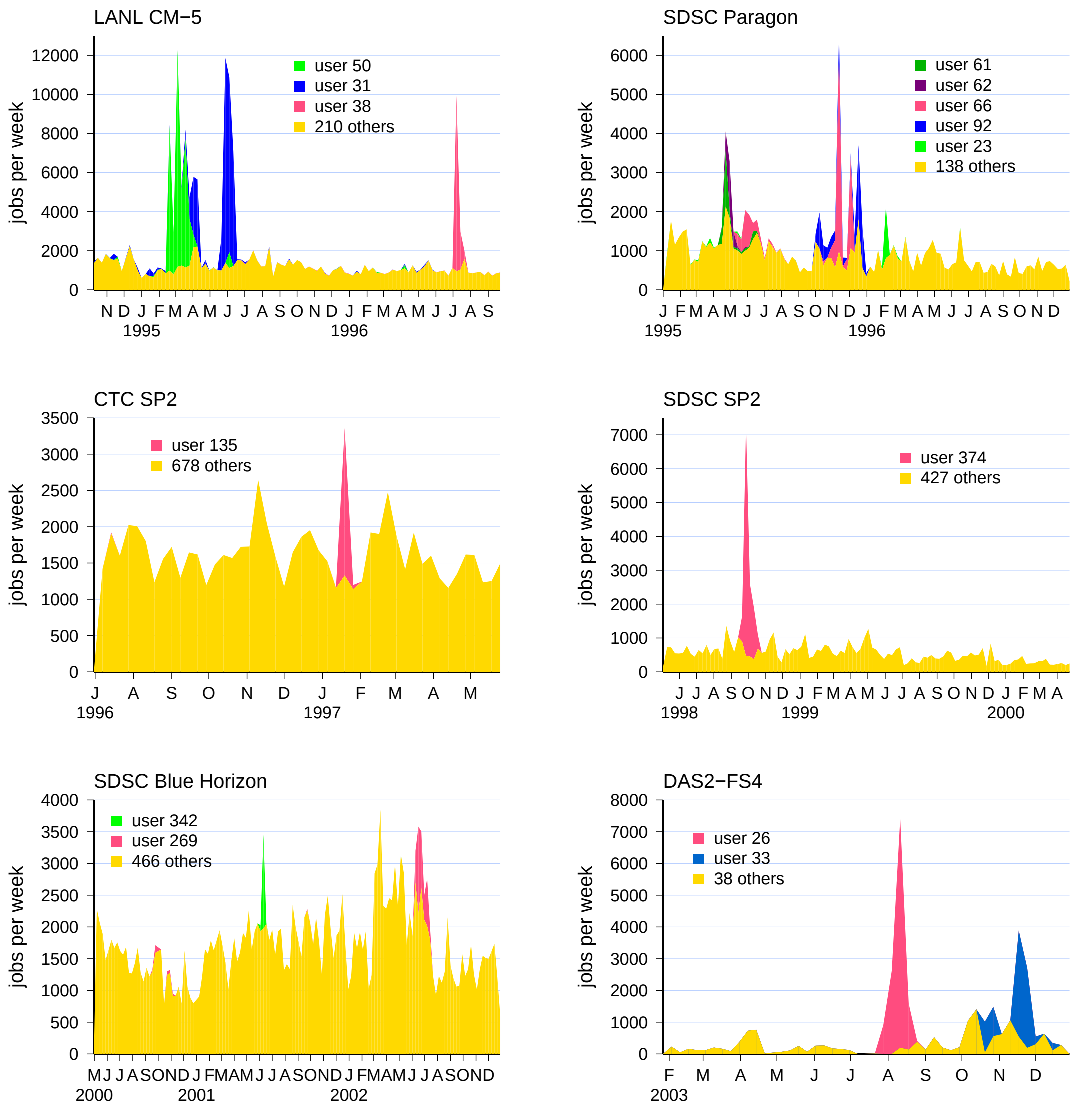

**Figure 5.** *Arrivals per week on six parallel supercomputers, showing flurries of activity due to single users.*

We have observed workload flurries in a wide variety of computer systems. Fig. 5 shows the job load (the number of jobs submitted) as a function of time, at the granularity of weeks, from six logs of activity on large-scale parallel supercomputers available from the Parallel Workloads Archive [17]. These logs come from different places, different machine types, and different times: the 512-node IBM SP2 installed at Cornell Theory Center (CTC), the 1024-node Thinking Machines CM-5 installed at Los Alamos National Lab (LANL), the DAS2 cluster at Utrecht University, and three machines from the San Diego Supercomputer Center (SDSC): the 400-node Intel Paragon, the 128-node IBM SP2, and the 144-node Blue Horizon machine (also an IBM SP, but with 8-way SMP nodes). The CTC and DAS logs are one year long, the Blue Horizon log is $2\frac{1}{2}$ years long, and the other three cover two years each. Together, these logs span the period from 1995 to 2003.

In all six logs, large flurries are observed. They range in size from double the average activity to 10 times the average activity, are caused by a single user, and extend from a few days to several weeks. The flurries in the CTC log and the Blue Horizon log seem similar to normal fluctuations, but nevertheless turn out to have an important effect (at least for CTC), as shown in Fig. 11.

It should be noted that this dataset includes all the long logs in the Parallel Workloads Archive. Flurries were not observed in the shorter logs, including the NASA Ames iPSC/860 (3 months), the KTH SP2 (11 months), the LLNL T3D (4 months), and the LANL Origin 2000 Cluster (5 months). Indeed, periods several months long with no flurry occur also in the logs that do include flurries.

Fig. 6 illustrates the process load on two of the machines, showing that job flurries do not necessarily correspond to process flurries (on parallel supercomputers a job can be composed of dozens of processes). In the Blue Horizon log, the largest process flurry is even much more distinctive than any job flurry. In fact, what exactly constitutes a flurry depends on the context in which the question is asked. "High load" can also be defined in terms of CPU utilization, memory usage, disk operations, or network bandwidth consumed. But here we focus only on job and process flurries, for which data is available, and which can be identified unequivocally.

Flurries are not unique to parallel supercomputers. For example, we have observed a case where 539,253 of 661,452 jobs (82%) submitted to a departmental server over a 30 hour period in March 2003 were all invocations of the Unix ps command by the same user. This turned out to be the result of a bug in implementing an exercise in the operating systems course[1]. A large flurry was also observed in the session log for March 2004 of a Unix server used by students (Fig. 7). In this case it turned out to be the result of ftp'ing

[1]Indeed, it is possible that some of the flurries on supercomputers are also the result of runaway scripts rather than being intentional. However, this does not detract from the importance of the phenomenon. On the contrary, situations in which flurries are unintentional add motivation to the need to identify them before using the workload as representative of normal work.

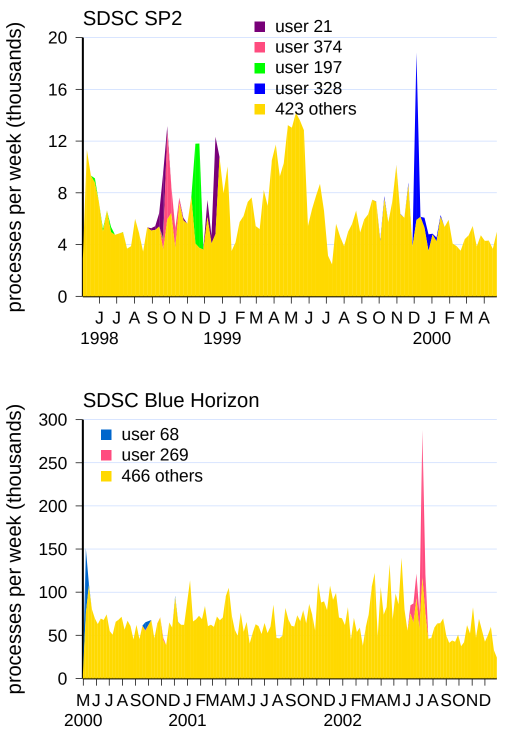

**Figure 6.** *Process arrivals per week also display flurries, which may be different from the job-arrival flurries.*

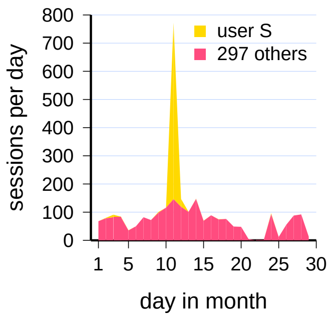

**Figure 7.** *A flurry in the sessions on a Unix server.*

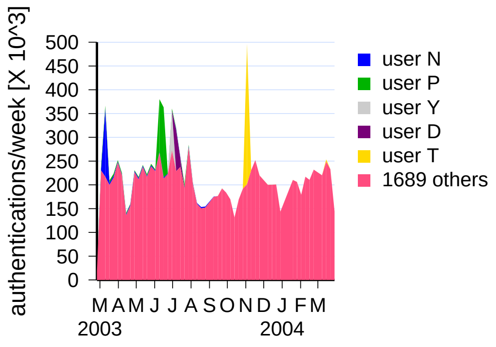

**Figure 8.** *Flurries of activity on an authentication server. Those by users D and T are failures, and swamp the normal level of about 2000 failed authentications a week.*

a large directory structure by a certain student one afternoon; the implementation automatically opened a new ftp session for each directory, and this was logged as a distinct user session. Obviously, this data does not represent normal user sessions, and would cause misleading results if used as the basis of a model of interactive user sessions. A third example is the activity on our authentication server (Fig. 8). In this case data covering a long period was available, and several distinct flurries were observed. Two of them consisted of failed authentications. In particular, the flurry attributed to user T was traced to a bug in Windows, where an authentication failure led to an infinite loop of retries.

A generalization of flurries is to consider any unique high-volume event. There are many accounts of such flurry-like events on the Internet. For example, new releases of software by Microsoft have caused the "midnight madness" phenomenon, where users flocking to download the new version (typically released at midnight) saturate the network overwhelming the servers [18]. Other examples include the surge of activity on CNN's servers on September 11, 2001. Ari et al. model such unique activity, which they call "flash crowds", with the aim of evaluating schemes to survive them [2]. This is especially important for sites set up specifically to cover one-time events, such as the Olympic games or the World Cup finals [3]. Targeted attacks on specific web servers, especially denial of service attacks, are also unique high-volume events. In this case, an analysis of the attack workload patterns is not only useful for evaluation of servers, but also as a tool in identifying such attacks [5].

Singular events lead to unique traffic patterns. We claim that it would be wrong to use workload data including such singular events to analyze the performance of web servers under normal conditions, just as it would be wrong to use normal data for an evaluation of how systems would behave under unique conditions. Of course, in these particular cases the unique high-load conditions may be more important and meaningful than the normal conditions; but the main point, of distinguishing between two workload classes, remains.

## 6. The Importance of Workload Sanitation

The examples from the previous sections demonstrate that many real workloads are actually mixtures of different classes of workloads. Moreover, in many cases the main dichotomy is between just two classes, which may be labeled as a "normal" workload and an "abnormal" workload. When we perform workload modeling and system evaluation, we typically want to use a normal workload — one that is generally representative of what systems will encounter during normal usage. But our results show that in practice using logs from real systems runs the risk of unwittingly using a mixture of normal and abnormal workloads.

Systems are usually evaluated for one of two reasons: to guide a purchase decision, or a design decision. In both cases using the wrong workload means that the decision may be optimized for the wrong conditions. This implies that it is imperative that multiclass workloads be identified as such, and that they be sanitized by removing the abnormal workload component before being used for modeling or in evaluations.

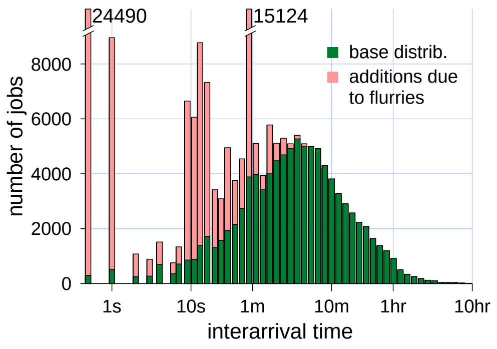

**Figure 9.** *Flurries turn the lognormal distribution of interarrival times in the LANL CM-5 log into a noisy and modal distribution.*

To justify this approach, we now present some examples of the effect that abnormal data may have. The most obvious effect is on statistical workload modeling. Such modeling attempts to generalize from specific workloads by finding invariants that are common to workloads from similar settings [4]. The presence of an abnormal workload class interferes with this endeavor by modifying the workload statistics. A couple of examples have been shown already above: the distribution of job sizes on the NASA Ames iPSC/860 has many bogus serial jobs due to the actions of system personnel (Fig. 2), and arrivals to the SDSC Paragon have a strange peak at 3:30 AM, probably due to robot activity (Fig. 3).

Additional examples are provided by workloads with flurries. Due to their repetitive nature, these flurries tend to modify the workload distributions by adding modes. Focusing on the LANL CM-5 interarrival times as an example, we find that the distribution for the whole log is distinctly modal, with several values that are extremely common and each come from a different flurry (Fig. 9). After the flurry-related data is removed, the underlying distribution can easily be characterized as lognormal.

The fact that flurries are not only different from the normal workload but also different from each other leads to severe non-stationarity. This is demonstrated in Fig. 10. This compares the distributions of four different workload attributes in the 1995 and 1996 portions of the LANL CM-5 log. For example, in 1996 the log contained a large flurry of activity by user 38 as seen in Fig. 5. The flurry consisted of jobs that were about 10 seconds long, arrived about 12 seconds apart, ran on 128 nodes, and used either very little memory or about 1.84 MB per node. This accounted for 12,344 (29%) of the total of 42,702 jobs in this part of the log, and thus had a decisive effect on the distributions of these workload attributes.

For comparison, during 1995 the log contained two other flurries, by users 31 and 50, which accounted for 71,161

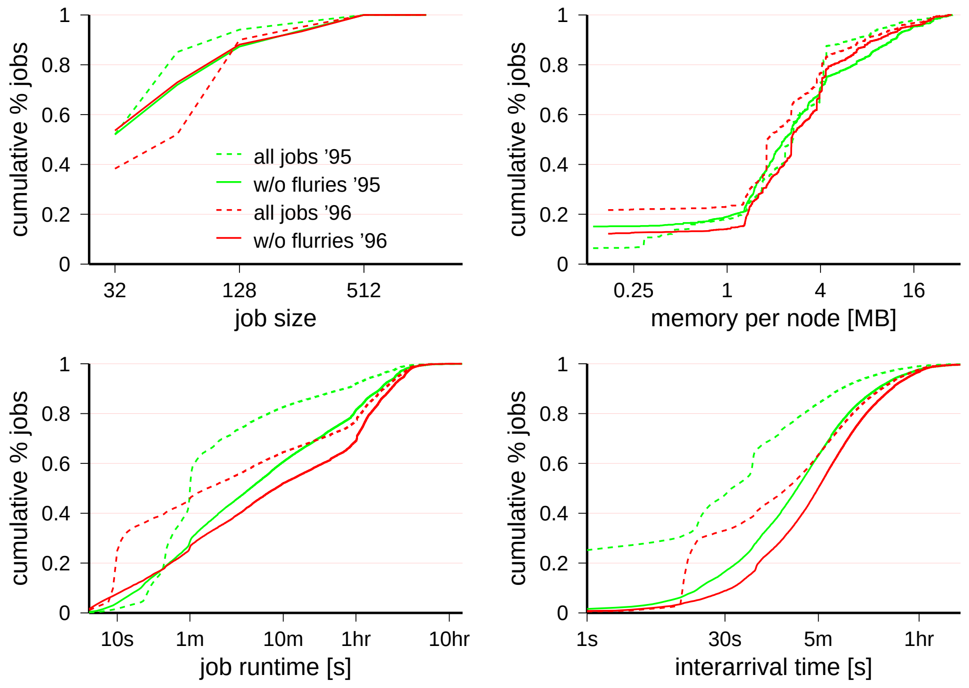

**Figure 10.** *Changes to the distributions of workload attributes when flurries are removed from the LANL CM-5 log. Most differences between 1995 and 1996 may be attributed to the different flurries.*

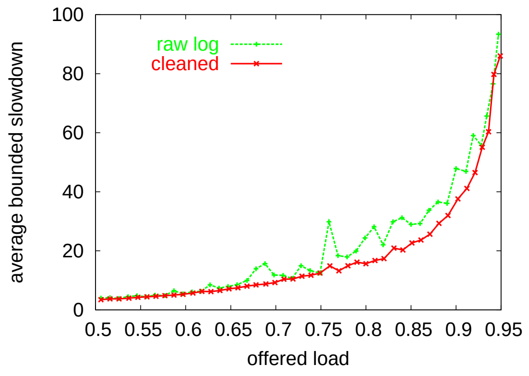

**Figure 11.** *Simulation of EASY backfilling on the CTC log. Flurries tend to be sensitive to exact simulation conditions, leading to instabilities. Simulations using a cleaned log are smoother.*

(58%) of that year's total of 123,058 jobs. By comparing the 1995 and 1996 distributions in Fig. 10, we see that the workload seems to be non-stationary, as the distributions for the two years are quite different (dashed lines). But if the flurries are removed, we find that in reality the base workloads are actually quite similar to each other (solid lines). Thus the major differences between 1995 and 1996 are actually the result of flurries introduced by 3 users out of a total population of 213. Including the flurry data gives the actions of these 3 users significant sway over the results.

Abnormal workload classes do not only affect workload statistics. They may also have an effect on the results of evaluations. For example, simulations of parallel job scheduling can be extremely sensitive to the exact workload conditions. An example is given in Fig. 11, using the CTC workload trace. This is a simulation of the performance of EASY backfilling [15], showing how it depends on the system's offered load (i.e. the fraction of the machine's capacity that is used). The way to create different offered load conditions is to multiply the job interarrival times by a constant. For example, if the original offered load is $\rho$, multiplying all arrival times by a factor of $\rho/0.8$ will change the offered load to 0.8. Naturally, this causes the produced schedule to change, as the space available for backfilling is changed. As Fig. 11 shows, such changes to the schedule cause large fluctuations in the performance results as measured by the bounded slowdown metric. It would be ludicrous to take such effects at face value, and claim that, say, the expected performance at a load of 0.77 is much better than at a load of 0.76. In fact, these fluctuations are caused by an interaction between the flurry of activity of user 135 and other jobs that appear earlier in the log (for a detailed examination of this interaction, see [19]). If the 2000-job flurry of activity by user 135 is removed (this is 2.5% of the total of 78,500 jobs in the log) the result becomes a smooth curve similar to those produced in queueing analysis.

## 7. Identifying Abnormal Workload Classes

Our thesis is that when new workload data is collected, it should be subjected to a test of whether it is actually a mixture of normal and abnormal workloads. If this is indeed the case, the abnormal workload can then be filtered out, leaving only the normal workload for use in system evaluations. The question is how to perform this test and identify abnormal workloads.

The principle of the "normal workload" test is simple. Observing the examples given above, it is evident that abnormal workload classes tend to be unique. For example, in Fig. 3 we saw the daily arrival pattern to 5 different parallel supercomputers. All of them include a daily cycle, with more jobs arriving during work hours than at night. Only one had an additional sharp peak of numerous jobs that arrive at 3:30 AM. If we look only at this log, we cannot say that this peak is abnormal. But comparing this log to the other four, we find that this is indeed the case. Thus searching for abnormalities is the flip side of searching for invariants [4], which lies at the base of all workload modeling.

The problem is that there is no a-priori way to anticipate where abnormalities will show up. The only way to find them is to investigate the workload at hand from various angles, and compare it with other workloads and with experience. This may include one or more of the following actions, which we have used to find the abnormalities reported in this paper:

- Many workloads are actually a time series, with arrivals that occur over some span of time. This can be visualized by tabulating arrivals per unit time, as was done in Figs. 5 and 6. Such a visualization may identify suspicious events of high activity (like flurries) or low activity (like holidays or down time).
- The same data may be re-drawn as an average per time unit, based on a known cycle like the daily cycle. This leads to graphs like those of Fig. 3. Such graphs are especially useful for identifying repetitive robotic work, such as scripts that are fired at the same time each day.
- Workloads typically have a relatively small number of main attributes. For example, parallel jobs have their size (degree of parallelism, or number of processors used) and runtime; packets and files have their size (this time measured in bytes); and user sessions have their duration. The distributions of such attributes should be tabulated in the form of a histogram or a CDF, as was done in Figs. 1 and 2. This can then be used to search for events that don't make sense, such as interactive user sessions that are more than a day long, or a preponderance of parallel jobs that only use a single processor. Finding such anomalies provides the analyst with a starting point for further investigation.

A certain difficulty with the approach outlined above is that it is labor intensive: one has to display the data in various forms and inspect them for unexpected patterns. The question is whether this process can be automated. The answer is that full automation is probably impossible, as the identification of abnormalities requires judgment regarding what should be regarded as normal. However, partial automation is possible and beneficial.

For example, the identification of the flurries portrayed in Figs. 5 and 6, and the claim that such flurries do not appear in some other logs, used an automated analysis that performed the following steps:

1. Partition the log data into weeks, and count the number of jobs submitted in each week.
2. Sort the weeks by the level of activity in them, and focus on the most active weeks (the exact number depends on the length of the log). These naturally include those weeks that have abnormally high loads.
3. Tabulate the number of jobs submitted by each user in each of the high-activity weeks.
4. For each user, find the maximal number of jobs submitted in any of the high-activity weeks.
5. Sort the users according to their maximal activity, and focus on the 6 to 10 most active users. These are the users that were the most active in the weeks with abnormally high levels of activity, and are therefore good candidates for having generated flurries.
6. Create data for graphs like Fig. 5: for each week in the log, output the level of activity of those users who were the most active users in one of the high-activity weeks.

Another example is the automated identification of outlier days suggested by Cirne and Berman [8]. They noticed that workload logs may contain periods that are highly abnormal, and suggested that such periods be deleted. Their methodology was to partition the log into days, and characterize each day's workload by a vector of length $n$ (specifically, this was applied to modeling the daily arrival cycle, and the vector contained the coefficients of a polynomial describing it). The days were then clustered into two clusters in $R^n$. If one of the clusters turned out to contain a single day, that day was deleted, and the process was repeated. Note, however, that this approach would not catch some of our examples: both the NASA Ames iPSC system activity and the SDSC Paragon activity at 3:30 AM span their respective logs, and are not confined to a single day. Flurries on parallel supercomputers are also typically longer than a single day, but this can be handled by modifying the Cirne and Berman procedure to also delete short spans of consecutive days.

## 8. Discussion and Conclusions

Exploratory data analysis is a field of statistics that deals with exploring new data sets in order to unravel the information that they contain. An important part of this is to detect and remove outliers — observations that lie an abnormal distance from others in a random sample. While some statistical tests have been devised for this, they are typically limited to cases where the assumed model is simple and well-defined, e.g. when samples come from a normal distribution. In other cases, visual techniques such as scatter plots are employed.

A similar situation occurs when using measured computer workloads for performance evaluation. Such workloads often contain a mixture of two or more types of workloads, where one is the "main" workload that reflects normal usage and conditions, and others are various anomalies and special cases. By using the workload as it was recorded the analyst runs the risk of optimizing the system to these unknown anomalous workload components, which may never be repeated. This motivates the need to identify and remove singular and unrepresentative workload components.

We have found numerous examples of workloads mixtures like this, including

- System personnel who created a large volume of bogus work as a test that the system was functional
- Scripts that are run automatically at night to perform cleanup and maintenance tasks, and are not representative of normal user activity
- Flurries of intense activity by individual users that dominate the workload for a limited time (in one case, from a Unix server, this was identified as a bug)
- Unanticipated mixtures of several workload types, e.g. data regarding login sessions that included sessions left open and unused for long periods

Importantly, these examples were not hard to find. In fact, a large fraction of measured workloads we looked at were easily found to contain such anomalies. For example, the two datasets from our department's Unix servers (inferno and pita) were originally collected as an exercise in a performance evaluation course, with the goal of showing a simple example of collecting data regarding a local workload. In both cases, this example turned out to be much more involved than intended.

The existence of such workload mixes raises the question of how to identify them. We have used the methods of exploratory data analysis: investigate the data from many different angles, with an eye for phenomena that seem extraordinary. Of course, "extraordinary" is subjective, and may also depend on your experience with similar workloads. It is therefore desirable to compare the suspect workload with other similar workloads, or with other segments of the same workload trace, to verify that the suspect phenomenon indeed violates the invariants seen elsewhere.

An important question for future work is how to take steps toward the automation of this process. We have identified one promising approach, that of tabulating levels of activity in successive time slices, and focusing on high-activity slices. Cirne and Berman have proposed a related technique, in which the descriptions of the workloads in different slices are clustered, in order to identify those that are unique and different [8]. However, much more work will be needed to fully explore the options for automating such analyses. Another approach is to search for deviations from an assumed statistical model of how the workload should look, where the model is based on previous observations. For example, the deviations from the lognormal distribution in Fig. 9 could be detected if the lognormal model was known.

Once abnormal workload components are identified, the question remains of what to do about it. We have suggested that the abnormal activity be deleted from the record, a process we call "workload sanitation". This is in line with removing outliers, which is common practice in statistical analysis, and with the proposal of Cirne and Berman [8]. However, our approach may be considered as more radical, as it may involve larger fractions of the recorded workload. For example, in the LANL CM-5 workload, the three flurries we have identified amount to 39% of the total jobs that appear in the log. This raises the question of whether removing such a large fraction of the workload is ever justified.

We claim that removing anomalies is justified for several reasons. First, the anomalies we have found are all unique. For example, we have identified nearly a dozen different flurries in parallel supercomputer workloads, but all of them are different from each other in terms of the characteristics of the jobs in them. Thus using the recorded logs as-is amounts to using workloads with unknown large-scale perturbations. Removing the anomalous flurries may be expected to lead to more reliable results, as the remaining workload is closer to workloads as observed at other sites and at other times — thus better reflecting observed workload invariants.

Another argument is that in some cases the anomalous activity actually originates from a subset of users that is of no interest. The system personnel activity on the NASA Ames iPSC, the daily maintenance activity on the SDSC Paragon and the pita server, and the staff sessions on the inferno server, all reflect (relatively small) groups of users that are disjoint from the main group. If we are interested in genuine user activity, the additional work generated by these users constitutes troublesome interference.

A third argument is that the anomalies are often generated by a very small fraction of the user population. For example, the flurries are each generated by a single user, while the rest of the workload on those same systems reflects the combined activity of hundreds of users. The same holds for the pita server data, where one user continually submitted sets of about 200 jobs every 10 minutes. In such situations, a very large fraction of the activity is due to a vanishingly small fraction of the users; using the data as is risks optimizing for the activity of these users, at the expense of others.

Removing anomalies is especially justifiable in cases where they are actually unintentional, and arise due to runaway scripts or other bugs in the workload creation process. But it is also justifiable if they represent real work. The heart of the matter is that anomalies are rare and unique. Using a workload with some anomaly may emphasize the rare and unique event at the expense of the normal conditions. To argue for evaluations based on workloads with anomalies, one must argue that the activity of a specific user during a short time should indeed dominate the evaluation results. Also, one must be satisfied with results that may change considerably if the span of time covered by the evaluation is shifted such that the anomaly happens to be excluded. Thus *leaving the anomaly in* is actually the unjustifiable approach.

Based on all this, we suggest that workloads should be separated into "normal" workload and anomalous workload components. Modeling and evaluation of the normal workload may be expected to lead to reliable and consistent results that are applicable most of the time — that is, all the time during which anomalies are not present. Comparing evaluation results using the cleaned log with results based on the original log will identify whether the removed anomalies actually have a significant effect in the specific case being studied. Indeed, an interesting question is how to model or evaluate the effect of anomalies on a system designed and optimized for the more common normal workload. As anomalies are unique, it is doubtful whether using known anomalies can predict the effect of other potential anomalies. Important future work is therefore to develop methods to extend and generalize the results obtained with specific anomalies, and try to derive bounds on the effects of other potential anomalies.

An important byproduct of our work has been to identify a new type of workload anomaly: workload flurries. The essence of flurries is a huge surge of activity, due to a single user, which dominates and changes the workload for a limited period of time. It is dangerous to use logs with unknown flurries, because this may significantly reduce the reliability of the results. Specifically, flurries may amplify performance effects, thereby causing the evaluation to be very sensitive to fine details of the input, to the point of obscuring the effects related to the system issues being studied [19]. However, there are also smaller peaks of activity that are caused by single users; how do we decide when they qualify as a "flurry"? In the present paper we have used subjective judgment to pick out only the largest flurries, but this issue also begs for further research.

To summarize, it is extremely important to use real data regarding the workload on computer systems. But it is equally important to ensure that this is high-quality and representative data. Using measured workloads indiscriminatingly risks the introduction of unknown anomalies that may lead to unknown effects. As finding anomalies is not trivial, this information should be shared together with the original data. In other words, when workload data is made available, it should be accompanied by all the accumulated knowledge regarding problems with its use, and specifically, with information regarding anomalies that occur in it. As a first step, we have added our data to the parallel workloads archive [17], from which much of our original data comes, and which is used by many researchers for studies of parallel job scheduling.

**Acknowledgment** This research was supported in part by the Israel Science Foundation (grant no. 167/03).

## References

[1] A. K. Agrawala, J. M. Mohr, and R. M. Bryant, "*An approach to the workload characterization problem*". *Computer* **9(6)**, pp. 18–32, Jun 1976.

[2] I. Ari, B. Hong, E. L. Miller, S. A. Brandt, and D. D. E. Long, "*Managing flash crowds on the Internet*". In 11th *Modeling, Anal. & Simulation of Comput. & Telecomm. Syst.*, pp. 246–249, Oct 2003.

[3] M. Arlitt and T. Jin, "*A workload characterization study of the 1998 world cup web site*". *IEEE Network* **14(3)**, pp. 30–37, May/Jun 2000.

[4] M. F. Arlitt and C. L. Williamson, "*Web server workload characterization: the search for invariants*". In *SIGMETRICS Conf. Measurement & Modeling of Comput. Syst.*, pp. 126–137, May 1996.

[5] M. Burgess, H. Haugerud, S. Straumsnes, and T. Reitan, "*Measuring system normality*". *ACM Trans. Comput. Syst.* **20(2)**, pp. 125–160, May 2002.

[6] M. Calzarossa and G. Serazzi, "*Workload characterization: a survey*". *Proc. IEEE* **81(8)**, pp. 1136–1150, Aug 1993.

[7] S-H. Chiang and M. K. Vernon, "*Characteristics of a large shared memory production workload*". In *Job Sched. Strat. for Parallel Proc.*, pp. 159–187, Springer Verlag LNCS 2221, 2001.

[8] W. Cirne and F. Berman, "*A comprehensive model of the supercomputer workload*". In 4th *Workshop on Workload Characterization*, Dec 2001.

[9] D. G. Feitelson, "*Metric and workload effects on computer systems evaluation*". *Computer* **36(9)**, pp. 18–25, Sep 2003.

[10] D. G. Feitelson and B. Nitzberg, "*Job characteristics of a production parallel scientific workload on the NASA Ames iPSC/860*". In *Job Sched. Strat. for Parallel Proc.*, pp. 337–360, Springer-Verlag LNCS 949, 1995.

[11] D. Ferrari, "*Workload characterization and selection in computer performance measurement*". *Computer* **5(4)**, pp. 18–24, Jul/Aug 1972.

[12] S. Hotovy, "*Workload evolution on the Cornell Theory Center IBM SP2*". In *Job Sched. Strat. for Parallel Proc.*, pp. 27–40, Springer-Verlag LNCS 1162, 1996.

[13] J. Jann, P. Pattnaik, H. Franke, F. Wang, J. Skovira, and J. Riodan, "*Modeling of workload in MPPs*". In *Job Sched. Strat. for Parallel Proc.*, pp. 95–116, Springer Verlag LNCS 1291, 1997.

[14] A. M. Law and W. D. Kelton, *Simulation Modeling and Analysis*. McGraw Hill, 3rd ed., 2000.

[15] D. Lifka, "*The ANL/IBM SP scheduling system*". In *Job Sched. Strat. for Parallel Proc.*, pp. 295–303, Springer-Verlag LNCS 949, 1995.

[16] U. Lublin and D. G. Feitelson, "*The workload on parallel supercomputers: modeling the characteristics of rigid jobs*". *J. Parallel Distrib. Comput.* **63(11)**, pp. 1105–1122, Nov 2003.

[17] *Parallel workloads archive*. URL http://www.cs.huji.ac.il/labs/parallel/workload/.

[18] E. Schooler and J. Gemmel, *Using multicast FEC to solve the midnight madness problem*. Technical Report MS-TR-97-25, Microsoft Research, Sep 1997.

[19] D. Tsafrir and D. G. Feitelson, "*Instability in parallel job scheduling simulation: the role of workload flurries*". In 20th *Intl. Parallel & Distrib. Proc. Symp.*, Apr 2006.