Locality of Sampling and Diversity in Parallel System Workloads

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

Abstract

Observing the workload on a computer system during a short (but not too short) time interval may lead to distributions that are significantly different from those that would be observed over much longer intervals. Rather than describing such phenomena using involved non-stationary models, we propose a simple global distribution coupled with a localized sampling process. We quantify the effect by the maximal deviation of the distribution as observed over a limited slice of time from the global distribution, and find that in real workload data from parallel supercomputers this deviation is significantly larger than would be observed at random. Likewise, we find that the workloads at different sites also differ from each other. These findings motivate the development of adaptive systems, which adjust their parameters as they learn about their workloads, and also the development of parameterized workload models that exhibit such locality of sampling, which are required in order to evaluate adaptive systems.

1 Introduction

Locality of reference is one of the best known and widely occurring attributes of computer workloads [6]. It means that if a certain memory location is referenced, there is high probability that it (or a nearby location) will be referenced again soon. This is the basis for the success of all caching schemes that maintain recently-referenced data in high-speed memory for possible reuse, including processor caches, virtual memory, file system buffer caches, and caching web pages on proxies. Likewise, complex functions may cache their result and reuse it rather than recomputing it if they are called again with the same argument [21], and freed memory blocks can be cached in anticipation of future requests for the same block sizes [33].

Systems can also use their history to learn about their environment [34]. For example, several algorithms have been devised to predict job runtimes based on historical data regarding previous executions, in the interest of providing schedulers with better information [12, 26]. Adaptive algorithms can be designed that observe the workload and change the system's behavior accordingly. For example, schedulers and load balancers may adapt to their workload and tune their settings so as to provide the best performance [22, 29, 30, 35].

The common thread uniting all these examples is the belief that the recent past is indicative of the near future, and that this can be exploited to advantage. As such, it is a generalization of the idea of locality of reference. In particular, a distinction can be made between the workload on short time scales, which tends to be repetitive and regular, and the workload on long time scales, which tends to be more random [11]. This distinction has a generative explanation: users of computer systems often perform the same tasks repeatedly many times before moving on to do something else; programs often repeatedly execute the same nested loops over and again before moving on to the next phase of the computation, etc.

Regrettably, synthetic workloads used for performance evaluation often do not have this property. The predominant approach to workload generation considers it as a process of sampling from some population or distribution [18, 15], and this approach is indeed used in parallel workload models [16, 7, 5, 20] A large population of possible workload items is postulated, and the next one to arrive is simply selected at random from this population. Likewise, the specification of service time distributions in queueing analysis assumes this model [17]. But such random sampling, where each workload item is independent of those that came before it, will not lead to any type of locality.

Instead, we suggest that the workload generation model should employ sampling with locality. This means that the sampling process has two levels: at the top level, one selects the part of the distribution or population on which to focus. Then, at the bottom level, you select workload items at random from this part of the distribution or population. In its simplest form, the top level selects a single workload element, and the bottom level repeats it many times. Such an approach matches the observed characteristics of workloads at both short and long time scales: the local sampling creates regularity and repetitiveness at the short time scales, while the shifts from one locality to another create the randomness of long time scales.

Using a workload model with such repetitions for performance evaluation is important because real systems operate in an on-line manner, and must contend with the workload as it appears on the immediate time scale. Thus a model based on random sampling from a global distribution will subject the system to a very different workload than one based on localized sampling from the same distribution. This has two consequences. First, a workload sampled from a global distribution will be more mixed, and have a higher probability that unusual events cancel each other out. Second, using a global workload will not enable a reliable evaluation of adaptive systems that *rely* on locality.

Adaptive systems are becoming more common in the quest for efficiency and autonomous management. For example, consider a batch scheduler in a parallel system that needs to set the relative priorities of its different queues to obtain the best possible performance. This can be done by simulating the results that would be obtained for the current workload had other settings been in effect, and switching if those other setting seem to be better [29, 30]. In reality, as workloads evolve, new settings would be needed, and each setting would stay in effect for a certain period of time due to the locality. But in a model that just uses the global distribution this is not the case. Thus such a model prevents the evaluation of the potential benefits of the adaptive scheduler.

The idea of two-level sampling is not new. Perhaps the closest related work is that on user-behavior graphs [11, 4]. These are motivated by the same arguments submitted here — that the workload should reflect the activity of those users that are active at each point in time. While this

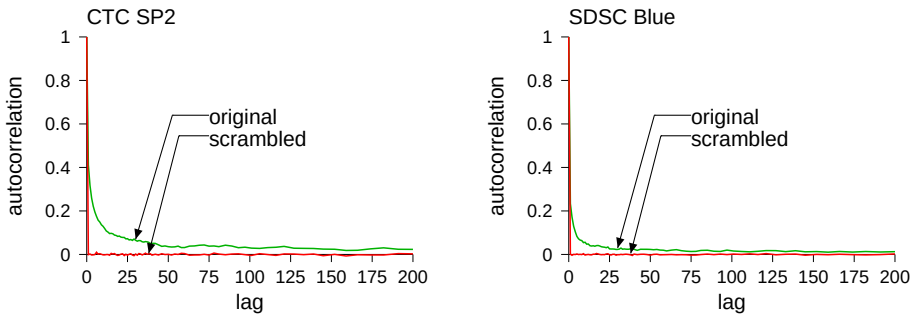

Figure 1: *Autocorrelation of runtimes of parallel jobs submitted to two parallel supercomputers. When the sequence of jobs is scrambled all correlation is lost. In this and other figures only a couple of examples are shown; similar plots can be drawn for the other logs too.*

device can be used to generate locality of sampling, the focus of user-behavior graphs is different: it is the desire to capture the *sequences* of jobs that typify different users.

The main contributions of this paper are

- To identify the phenomenon of locality of sampling as an important characteristic of computer workloads, distinct from known characteristics based on the autocorrelation function (Section 2),
- To quantify this effect based on the maximal deviation between the distributions observed during a limited slice of time from the global distribution (Section 3),
- To show how this effect can be generated in synthetic workload models by repeating selected jobs multiple times (Section 4), and
- To show that the same mechanism may be used to characterize how workloads differ from one location to another (Section 5).

2 The Phenomenon of Locality of Sampling

When analyzing and modeling empirical workload data, a chief concern is often that the data be stationary. This stems from the common approach of modeling workload generation as a sampling from a distribution [18, 15]. To be valid, this model requires observed workloads to look like random sampling from a distribution, and in particular, all workload items should come from the same distribution. The formal definition of stationarity requires that the correlation structure also be preserved, but this is not always verified.

Upon inspection, however, one finds that many workload datasets do not seem to be stationary. On long time scales of months and years, this can be caused by workload evolution — the change in workload as users learn to use a new system, or as the dominant application types change with time [14, 13]. On shorter time scales, it can be caused by workloads that have periodic cycles (e.g. as a result of the human daily work cycle).

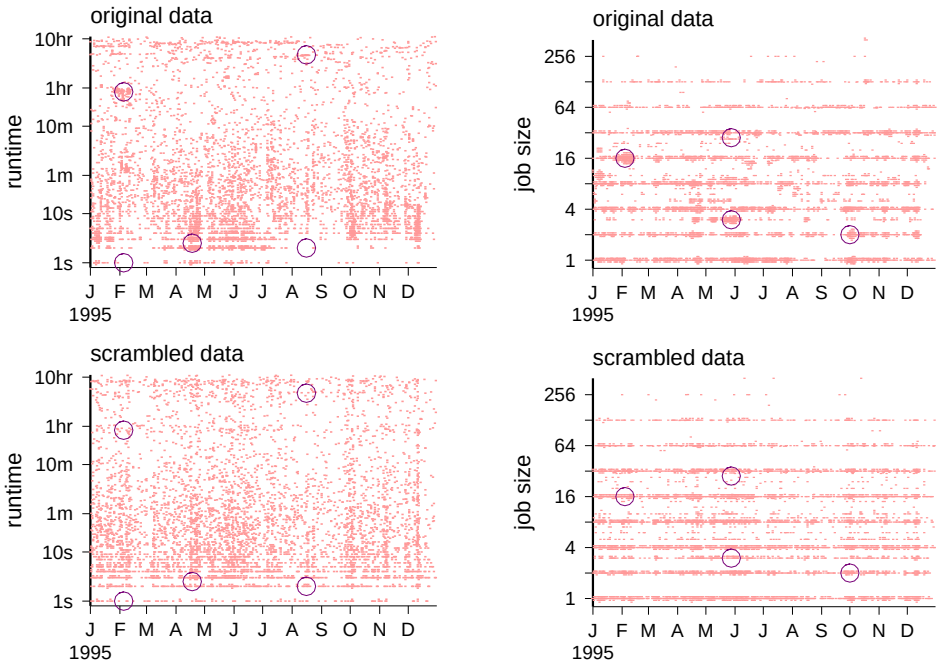

Figure 2: *Graphical demonstration of the existence of locality of sampling, by comparing the original data with scrambled data after a random permutation along the time axis. The original workloads typically do not look like a random sampling from a global distribution. Data from the SDSC Paragon.*

Another important characteristic of computer workloads is their locality — a persistent similarity between nearby items. Locality may be quantified by noting that if items are similar to each other there is a correlation between them. The degree of locality can then be measured by the autocorrelation function: the correlation of a list of items with itself after being shifted by a certain lag. An example is shown in Fig. 1. The items here are parallel jobs, and the quantity of interest is their runtime[1]. The autocorrelation function is pretty high even for large lags, showing that there is a correlation between jobs that are separated by dozens of other jobs. (Locality can also be quantified by the average stack distance — the average number of distinct items since seeing this one the last time, so named because it is easily calculated by keeping the items in an LRU stack [28]. However, this requires continuous quantities like runtime to be discretized and is therefore sensitive to the granularity of the discretization.)

The relationship of locality and stationarity is subtle. For example, workloads may exhibit bursty behavior at many different time scales, but this does not necessarily imply a lack of stationarity — just that there are long-range dependences among jobs. In fact, given finite data it is not always possible to distinguish between long-range dependence and a non-stationary trend. Like-

[1]Our study focuses on the workload on parallel supercomputers and clusters. Such workloads are composed of parallel jobs, that need a certain number of processors and run for a certain time. The data comes from the Parallel Workloads Archive, which contains accounting logs from large-scale production systems; it is summarized in Table 6.

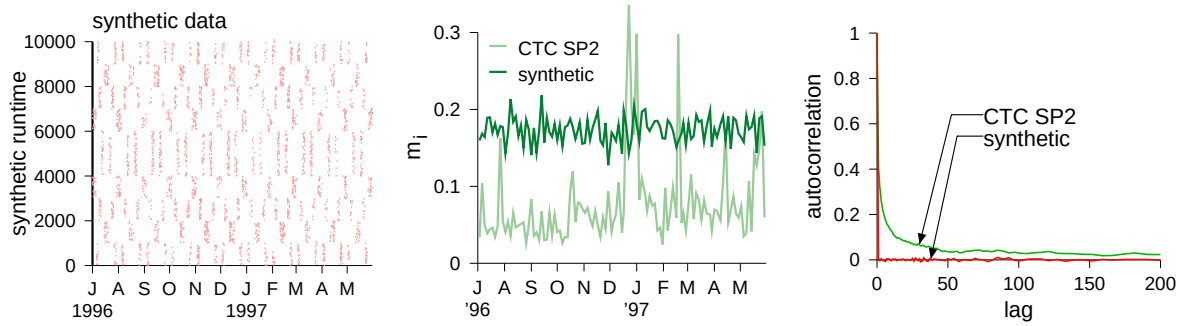

Figure 3: *Synthetic dataset displaying consistently high locality of sampling (as quantified in Section 3, middle plot), but no autocorrelation at positive lags.*

wise, the existence of locality does not contradict stationarity — it just means that there are local correlations. Locality of sampling focuses on these short-range dependences. While the exposition here assumes a stationary background distribution, this distribution can in principle change with time. Locality of sampling is therefore orthogonal to long-range dependence and non-stationarity, and can be combined with them if needed.

In essence, locality of sampling assumes a stationary global distribution (the marginal distribution), but a localized sampling process. In effect, this is a way to avoid the need to describe how the distribution changes with time. Instead, we describe how the sampling changes. The advantage is that this can be very simple, as shown in Section 4: simply replicating samples can lead to the desired effect. Moreover, it resonates with observations regarding how real workloads are generated. But of course other modeling approaches are also possible.

Note also that locality of sampling is a generalization of locality of reference, in that it refers to all aspects of workloads and not only to locations in storage. In particular, it may also imply a form of clustering, as we may consider sampling of complete multi-attribute workload items, rather than just values from each workload attribute distribution independent of other such distributions.

A simple way to visualize locality of sampling is by using scatter plots in which the X axis is time, and the Y axis is a workload attribute. Such a plot shows how the distribution of values of this attribute changes with time: a vertical slice of the plot shows the distribution at the time (position along the X axis) of this slice. To visualize multiple samples that fall at the same spot (that is, samples with the same value that occur at the same time) we create a disk; the more samples, the larger the disk.

An example of this device is shown in Fig. 2. This shows the distributions of job runtimes and sizes on the SDSC Paragon parallel supercomputer, and how they changed over a period of one year. To emphasize the concentrations of sampling certain values in a restricted span of time, we compare the given data with a scrambled version. The top two plots are the original ones. Some prominent concentrations of values are marked with circles to help guide the eye (but note that there are many other concentrations too). The bottom plots are scrambled, or randomized. This means that some random permutation was applied to the jobs in the log (but keeping the original arrival times, so the vertical streaks corresponding to high loads remain). As a result, jobs that

used to be next to each other may now be distant, and jobs that originally were unrelated are now next to each other.

The effect of scrambling on the scatter plots is that the locality properties are lost: at every time, we now observe a random sample from the global (marginal) distribution. In particular, the concentrations of very many samples of related values are spread out and create or contribute to horizontal streaks (this is especially common for powers of two in the size distribution). At the same time, gaps in the original horizontal streaks are now filled in. The fact that this is different from the original plots testifies to the fact that the original ones exhibit locality of sampling. Such phenomena are not unique to the SDSC Paragon machine. Using other datasets from the Parallel Workloads Archive leads to similar plots.

It should be noted that while locality of sampling can lead to autocorrelation, and it seems that the two indeed go hand in hand in real datasets, these two properties of the data are not equivalent to each other. This is demonstrated by the simple synthetic dataset shown in Fig. 3. The global distribution of this data is uniform. But in each time slice, only two small sub-ranges are sampled, leading to a strong locality (this is quantified in the next section). The trick is that these sub-ranges are symmetrically placed, thereby leading to essentially no autocorrelation at any positive lag.

Locality of sampling is also distinct from the well-known phenomenon of self similarity (e.g. [19, 23]). Self similarity refers to structure in the *arrival process*, i.e. in the series of time instances at which new work arrives; it refers to the fact that this sequence is bursty at many different time scales. Locality of sampling refers to *attributes of the arriving work items*, i.e. to what arrives at those time instances.

3 Quantifying Locality of Sampling

The essence of locality of sampling is that if we look at a short time scale, we will only see part of the global distribution. In other words, the distribution at a short time scale is different: it is less diverse in the values that it contains, or in other words, it is more modal. Moreover, this is time dependent: at different times we will observe different values.

The proposed metric for quantifying the degree of locality of sampling tries to formalize this intuition. This has to be done with care, because at extremely short time scales lack of diversity is expected: a small number of samples cannot represent the whole distribution. We will start by dividing the time line into equal-duration slices, and finding the distributions of workload items when considering each slice independently; we call these *slice distributions* because they are limited to a slice of time. The metrics are then based on direct measurement of the difference between these slice distributions and the global distribution.

3.1 Step 1: Deviation of Slice Distributions

Fig. 4 shows example slice distributions for 3 selected week-long slices of the SDSC Paragon log. Due to the locality of sampling, these distributions tend to be different and much more modal than the global distribution, as indicated by their more step-like shape. For example, the data for

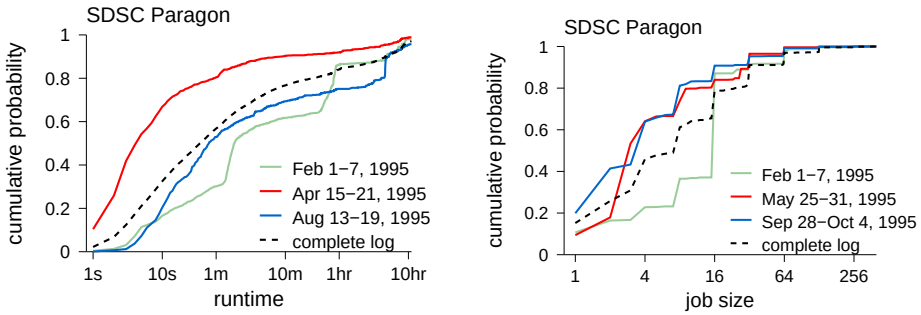

Figure 4: *The distributions of job runtimes and sizes on select weeks tend to be modal and different from each other and from the distribution of the whole log. Selected weeks correspond to markings in Fig. 2.*

February 1–7 indicates a preponderance of 16-node jobs, running for either a couple of minutes or about one hour.

Based on this, we can propose an actual measure of the divergence of the weekly (or other short-range) distributions from the global distribution. This is inspired by a combination of two tests for goodness of fit: the χ^2 test and the Kolmogorov-Smirnov test. The difference is that here we use the tests in reverse: we want to show that the distributions are different from each other, and quantify how different they are.

The essence of the χ^2 test is to divide the range of possible values into sub-ranges with equal probabilities, and verify that the number of samples observed in each are indeed nearly equal. Our quantification will therefore be to divide the overall range into subranges that have equal probabilities *according to the global distribution*, and observe the *maximal* probability for a single range *according to the slice distributions*. This measures a mode that is present in the slice distributions but not in the global one. We use the maximal deviation, as in the Kolmogorov-Smirnov test, as we are interested in the deviation between the global and slice distributions.

In order to be meaningful, there should be at least a few samples in each subrange. This places constraints on how the measurement is done. Assume the whole log contains a total of N samples (e.g. parallel jobs). If the length of the log is d days, there are N/d jobs per day on average. This has to be large enough to apply the χ^2 test using enough subranges. If it is too small, we need to consider a larger basic time unit. Using the 1995 SDSC Paragon log as an example, it contains 53,970 jobs (in the cleaned version) and spans a full year, for an average of 147.9 jobs per day. This should be enough for a resolution of more than 20 subranges. However, due to fluctuations in activity, some days will have much less jobs. It may therefore be better to use a somewhat longer time unit, say 3 days.

Given that we have selected a slice size of t time units and a resolution of dividing the range of values into r bins, the calculation proceeds as follows.

1. Create a histogram of the complete (global) data, and partition it into r equally likely ranges. This defines the boundary points of the ranges.
2. Partition the log into d/t successive slices of t time each (this need not be measured in days

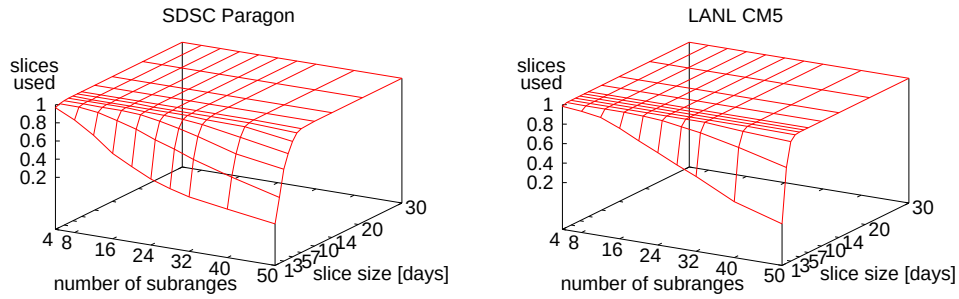

Figure 5: *Fraction of slices found to be usable for different parameter combinations.*

as in the above example — the time unit should match the type of data being considered).

3. For each of these slices of the log (indexed by i), do the following:

   (a) Find the number of workload items N_i in this slice of the log.

   (b) Create the slice histogram of these N_i workload items, and count how many of them fall into each of the r ranges defined in step 1. Denote these observed counts by $o_1, \ldots, o_r$.

   (c) By construction, the expected number of items in each range (assuming the global distribution) is $e_i = N_i/r$. We are interested in the deviations from this, and in particular, in the maximal relative deviation. Therefore we compute

$$m_i = \frac{\max_{j=1..r} \left\{ |o_j - e_i| \right\}}{N_i - e_i}$$

   This is slightly different from the conventional expression used in the χ^2 test. First, we use the max rather than a sum, as is done in the Kolmogorov-Smirnov test. This has the effect of being more sensitive to deviations, and emphasizing the concentration of values in a subrange. Second, we use the absolute value rather than the square to ensure that the result is positive. This retains a linear scale for comparison. Finally, we divide by $N_i - e_i$ rather than by e_i. This normalizes the result to the range $[0, 1]$, as the maximal value for any o_j is N_i, which occurs if *all* the samples appear in the jth subrange.

4. Record the m_i values of the different slices.

An important question regarding the suggested procedure is the choice of parameters. As noted above, one of the considerations is data availability: we need enough data items to populate every subrange in each slice. Using a threshold of 5 items as the minimal requirement (as is common for the χ^2 method, e.g. [1]) we may expect that using a large number of subranges and short slices will cause many of them to be unusable. An example of what happens with real data is given in Fig. 5. This shows the fraction of slices of duration t days that are usable when divided into r subranges, meaning that they contain enough jobs. The data indicate that only very short slices

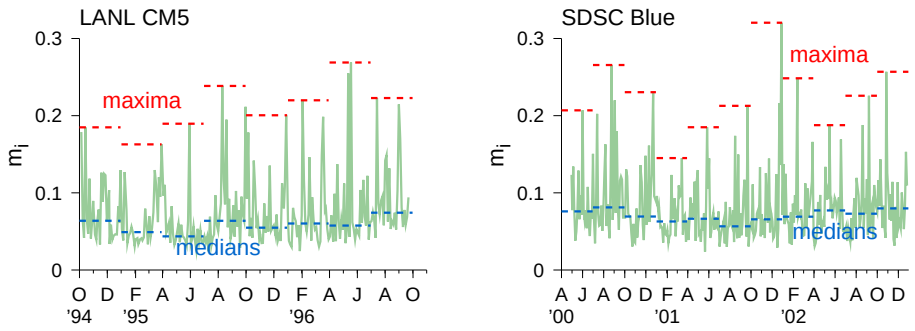

Figure 6: *The values obtained by m_i vary considerably in successive slices. As a result their maximum attains widely fluctuating values even in successive quarters. Their median is much more stable.*

create a problem. In particular, our default choice of 3-day slices and 24 subranges is in the middle of the rim of the plateau, and utilizes 94% of the slices for the SDSC Paragon log and 100% for the LANL CM5 log.

3.2 Step 2: Reduction to a Single Number

Given a year's worth of data, and using slices of 3 days, we end up with over 120 values of m_i for successive slices. The question then is how to best reduce them into a single metric.

Recall that we are looking to quantify the maximal divergence of slice distributions from the global distribution. This was the reason to define m_i based on the range with the maximal deviation from what was expected according to the global distribution. It would therefore be natural to continue with this approach, and define the final metric to be the maximal m_i value:

$$M'_{\text{max}} = \max_{1 \le i \le d/t} \{m_i\}$$

The problem with using the maximum is that extremal values are by their very nature unstable. An example is shown in Fig. 6. Obviously the values of m_i in the different slices varies widely. In a long log it may reach high values several times, but in between are slices with m_i values that are considerably lower. As a result the outcome of the calculation is very sensitive to the period being studied: if the log was shorter or longer, the maximal observed value could be considerably different.

Another demonstration of the volatility of results based on the maximum of the m_is is obtained as follows. Recall that we are using slices of 3 days for the slice distributions. These can be defined in three different ways, with shifts of one day relative to each other. Re-calculating the metrics for these different shifts leads to results like those shown in Table 1: a shift of one day can change the results by some 15%, or not at all.

The obvious alternative is to use the median of the m_is in the different slices. While this too fluctuates to a certain degree, the fluctuations are much smaller than for the maximum (Fig. 6 and

	shift		
metric	0	1	2
M'_{max}	0.3204	0.2692	0.2701
M'_{med}	0.0691	0.0698	0.0706

Table 1: *Effect of shifts of 1 or 2 days on the metric results, for the SDSC Blue Horizon log.*

log	M'_{max}	M'_{med}
LANL CM5	0.269	0.059
SDSC Paragon	0.429	0.098
CTC CP2	0.336	0.063
KTH SP2	0.218	0.085
SDSC SP2	0.577	0.101
Blue Horizon	0.320	0.069
DataStar	0.419	0.085

Table 2: *Results of measuring the degree of locality of sampling for the runtime distributions in different logs using the M'_{max} and M'_{med} metrics.*

Table 1). We may therefore define our metric to be

$$M'_{\text{med}} = m_{(d/2t)}$$

Applying the above to various logs of workloads from parallel supercomputers, when using 24 subranges and slices of 3 days, leads to the results shown in Table 2. For example, in the SDSC DataStar log the maximum-based metric is 0.419. This means that in one of the slices, 44.3% of the jobs were concentrated in a single subrange, rather than being equally dispersed among all 24 subranges. On the other hand, in the LANL CM5 log this metric was only 0.269, implying that the biggest concentration of jobs in one range was 29.9%. The values using the median are of course much lower. For example, the value of 0.085 for the SDSC DataStar log implies that for half of the slices, 12.3% or more of the jobs were concentrated in one range, which is about 3 times higher than would be expected by random sampling from the global distribution.

The question regarding all these results is whether they are actually significant. Obviously, even if the slice distributions are identical to the global one, some deviations are to be expected in a random sampling. We therefore need to compare our results to those that would be obtained via random sampling from the global distribution. This will be done using the bootstrap method [8, 9].

The bootstrap method is very simple: we just repeat the measurement a large number of times (say a thousand), using random samplings from the global distribution rather than the real slice distributions. In our case, each repetition creates N samples from the global distribution, partitions them into groups with sizes dictated by the N_is, creates the histogram for each group, calculates the m_i based on the deviation between the group's empirical distribution and the original global distribution, and finally finds the maximum and median of the m_is. Each repetition thus produces a single data point for each metric (the values of M'_{max} and M'_{med}) valid for a specific sampling from the global distribution. Repeating this a thousand times allows us to approximate the distribution of such results, that is, the distribution of M'_{max} and M'_{med} for random sampling.

We then check where the results for the real slice distributions falls in this distribution of results. If it is at the extreme end of the range (or beyond the end of the range), it is unlikely to have occurred by chance. Examples of the outcome of following this procedure are shown in Fig.

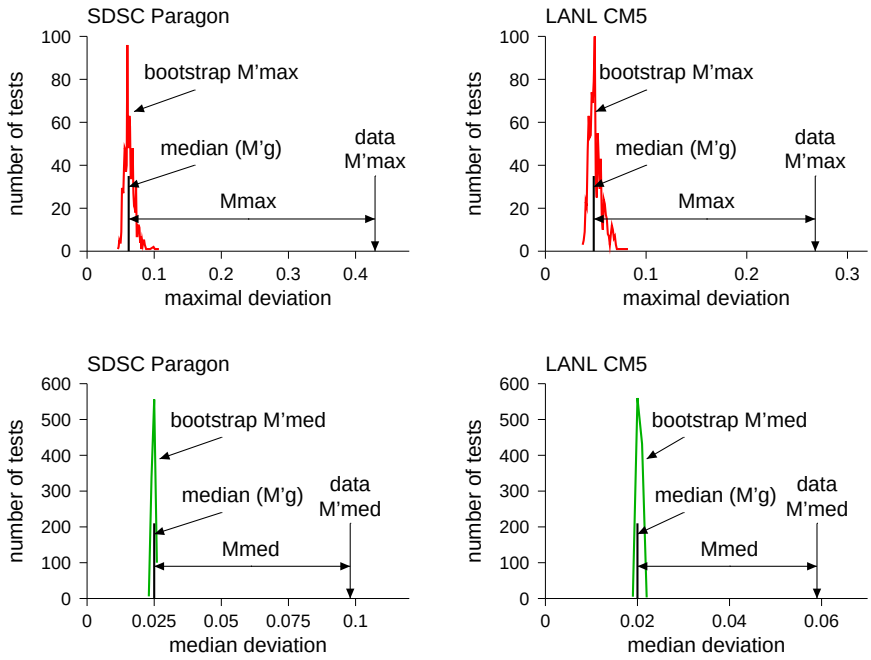

Figure 7: *Results of 1000 random tests of the maximal/median deviation observed when samples come from the global distribution of runtimes, compared to the maximal/median deviation observed in the slice distributions.*

log	M_{max}	M_{med}
LANL CM5	0.221	0.039
SDSC Paragon	0.367	0.073
CTC CP2	0.302	0.046
KTH SP2	0.159	0.056
SDSC SP2	0.511	0.072
Blue Horizon	0.271	0.052
DataStar	0.376	0.068

Table 3: *Results of measuring the degree of locality of sampling for the runtime distributions in different logs using the M_{max} and M_{med} metrics.*

7. Obviously, the actual results for the real slice data are way out of the scale of results that are obtained by random sampling from the global distribution, for both the maximum and median-based metrics. We can therefore claim that our results are highly significant.

The relatively narrow distributions of M'_{max} and even more so of M'_{med} obtained by random sampling from the global distribution present an opportunity for improving the metric of locality of sampling. Instead of measuring the absolute value of M' as computed above, we can measure the difference between M' and the value that would be obtained by chance. We define the latter as the median of the distribution of results obtained by the bootstrap method, because the median is

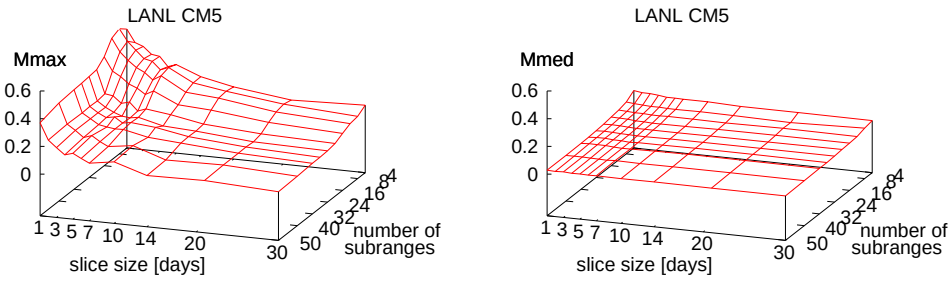

Figure 8: *The maximum-based metric is much more sensitive to the choice of parameter values than the median-based one.*

both representative and much more stable than the mean or max. Denote this median value by M'_g. Our final metric for locality of sampling is therefore

$$M = M' - M'_g$$

applied to either the maxima of the m_i or to their medians. This is demonstrated in Fig. 7, and the results are tabulated in Table 3.

The results shown in Fig. 6 indicate that the m_i values fluctuate widely among slices. Some of the values are very high, but many are actually quite low. This raises the concern that they may also be sensitive to the parameter values used in the measurement (in our case, 3-day slices and 24 runtime ranges). To check this, we repeated the measurements for a wide range of parameter values. The results shown in Fig. 8 indicate that M_{max} is indeed quite sensitive, and obtains much higher values when the slice size and number of ranges are reduced. The M_{med} metric, by contradistinction, is very stable. When combined with the previous results, this indicates that the M_{med} metric should be preferred as a measure of locality of sampling.

4 Modeling Locality of Sampling

Despite its importance, there has been relatively little work on introducing locality into workload models. Early work on virtual memory primarily used the LRU stack model [28]. Thiébaut et al. proposed a fractal model of memory traversal based on a hyperbolic law for the probability of jumps with different sizes [31]. Wang et al. claim that there is a correlation between burstiness in time (self similarity) and space (locality), and suggest the QPRS model to capture it, in which time-space is recursively divided into 4 and sampled with probabilities p, q, r, and s which sum to 1 [32]. Arlitt and Williamson used a stack model to investigate locality in Mosaic (WWW) conversations, and concluded that there isn't much locality [3]. Instead, they introduced a parameter that specifies the probability to repeat the last destination. A stack model was also used by Almeida et al. [2]. Shi et al. combined this with simulated large flows to recreate the observed temporal locality in web-traffic [25].

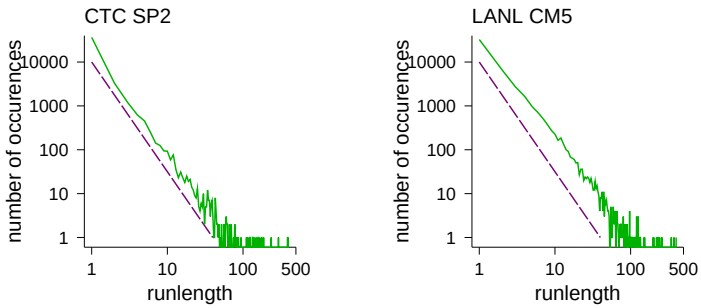

Figure 9: *Histograms of run lengths of similar jobs in production workloads. Note the use of logarithmic axes; the dashed line has a slope of −2.5.*

Our approach to modeling locality of sampling is both general and extremely simple. We initially sample from the global distribution. But then, instead of using each variate once, we repeat it a number of times. With enough repetitions we will get a sequence of samples that exhibits modal slice distributions. This approach is based on the observation that humans often repeat the same commands over and over again. For example, when writing and typesetting this paper I repeatedly execute the LaTeX command with only minor modifications to the input file, leading to a sequence of executions with a very similar profile. The same happens with compilations done when developing code, or when running production simulations of a set of related configurations. But on another day the work at hand changes, and we see repeated runs with a different profile.

Analyzing repetitions in workload logs leads to results like those shown in Fig. 9. In this analysis, we scan the workload data and partition it into separate streams of jobs submitted by different users. We then look for runs of equivalent jobs, defined to be jobs that execute the same application, and use the same number of nodes. The distribution of run lengths shows that many jobs are independent or are part of a short run, but on the other hand, some runs are very long.

Repeating the sampled workload items is designed to reproduce such runs; the only consideration is to correctly model the distribution of runlengths. This intuition may be formalized as follows. We are given a global distribution described by the pdf $f(\cdot)$. In addition, we need the distribution of repetitions, which will be denoted $f_{rep}(\cdot)$

1. Sample a variate $X = x$ with probability proportional to $f(x)$. Note that X may be a vector in case we are considering workload items that have multiple attributes, as is the case with parallel jobs, where two attributes are the runtime and size.
2. Sample a variate $R = r$ with probability proportional to $f_{rep}(r)$.
3. Repeat the X variate R times. This distorts the distribution locally.
4. Return to step 1 until the desired number of samples have been generated.

With a large enough number of samples, the number of times we will see a value of x will be proportional to $f(x)$, i.e. according to the global distribution, as desired. But these samples will come in bursts rather than being distributed evenly. This is qualitatively similar to the parameter used to specify the probability to repeat the last sample, suggested by Arlitt and Williamson [3].

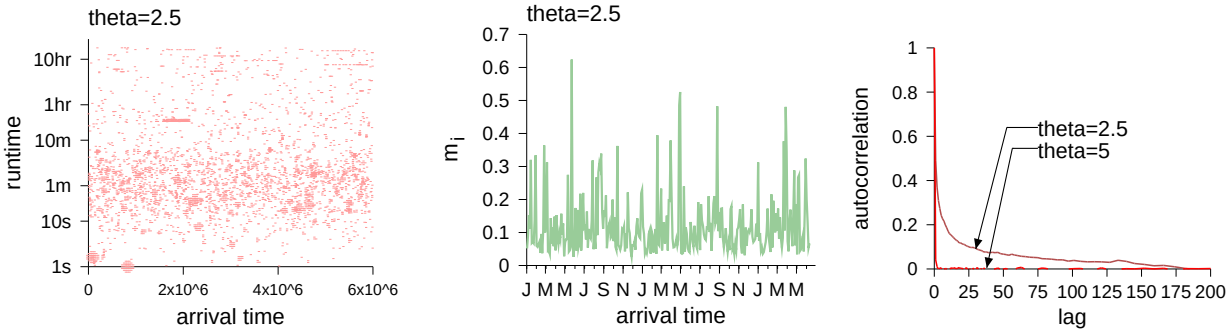

Figure 10: *Results of modeling locality of sampling by repeating workload items.*

However, such independent repetitions would lead to a geometric distribution of runlengths, rather than the power-law distribution suggested by the workload data.

As a concrete example, consider modeling the arrivals of jobs with a certain distribution of runtimes. The runlengths are taken from a Zipf-like distribution with parameter $\theta = 2.5$, chosen according to the data in Fig. 9. This means that the probability of a runlength of r is proportional to $r^{-2.5}$. This was generated by creating a table with normalized values running up to a maximal runlength of 1000.

Note that it is important to incorporate time in the model, in order to correctly interleave the generated sequences. Assume that each workload item is associated with an arrival time, and has some duration (its runtime). We then sample items as above, assigning them arrival times according to the model. But the repetitions of each item are assigned arrival times that are staggered according to each one's duration. Assuming the durations are large enough relative to the interarrival times, the result will be to interleave different repeated sequences. This interleaving leads to certain distances between repetition instances, as in the stack model.

The results of using this procedure are shown in Fig. 10, and exhibit essentially the same behavior as our original workloads. In particular, applying the m_i measure of locality of sampling to the output of this model indicates that the dynamic behavior over time of the model is similar to that of real workloads, as shown in Fig. 6. The autocorrelation structure is also similar to that of real workloads, as was shown in Fig. 1.

Additional improvements to the model are possible to make the repetitions more realistic and less deterministic. For example, we can add think times between the repetitions instead of having each one arrive immediately when the previous one terminates. In addition, we can introduce some variability between the repetitions, instead of making them identical. Naturally, such adjustments should be based on an analysis of the repetitions found in workload logs. This is currently left for future research.

Fig. 11 shows that the degree of locality of sampling depends on the prevalence of long sequences of repetitions. This, in turn, depends on the parameter θ of the distribution of repetitions: small θ lead to very long runlengths, while larger θ cause the distribution of runlengths to decay quickly, producing few repetitions if any.

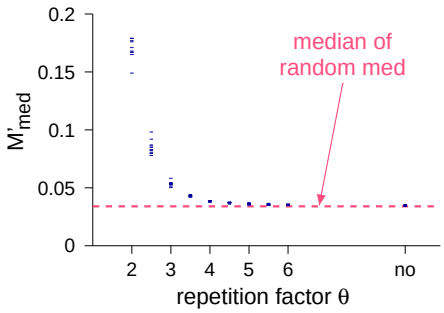

Figure 11: *The measured locality of sampling as a function of the parameter θ of the distribution of repetitions.*

To characterize the effect of θ on M'_{med}, we used the model to create 10 independent sequences of 100,000 jobs, and measured M'_{med} for each one. This was repeated for 9 values of θ between 2 and 6, and again for a model where repetitions are turned off completely. As Fig. 11 shows, values of θ from 2 to 4 lead to values of M'_{med} from 0.180 down to 0.038. The model default of $\theta = 2.5$ produces an M'_{med} in the range 0.078–0.098, of which the lower end is a bit high given the data in Table 2. Different workloads are therefore best modeled by different values in the range $\theta \in [2.5, 3]$. The data also show that when $\theta > 4$ this is essentially the same as having no repetitions. In this case, M'_{med} comes out at essentially the median value seen for random sampling from the global distribution.

Modeling locality of sampling by using job repetitions as suggested above has two important advantages: it is parsimonious, and it is generative.

Sampling with repetitions is as simple as a model can be, as it only requires the distribution of repetitions, which is described by a single parameter — the slope of the histogram (Fig. 9). Other models for locality are typically more complex. For example, Shi et al. find that the best model for a distribution of stack distances is a mixture of a Weibull distribution and a Pareto distribution, so five parameters are needed. Locality of sampling can also be achieved by a user behavior graph [11] or an HMM [24, 27]. However, this complicates the model as we need to describe the complete dynamics and what workload items correspond to each state. For example, when using an HMM we need to define the transition matrix among the states, and the output distribution for each state; the number of required parameters is at least linear in the number of states. Sampling with repetitions is much simpler, albeit this simplicity may come at the price of not capturing potential non-repetition sequencing properties.

The fact that the model is generative is even more important than parsimony. The alternative to a generative model is a descriptive one. The difference is that descriptive models just describe a certain situation, without explaining its mechanics. Thus they do not provide any clues regarding how the model should change under different conditions. For example, consider what may happen when the load on a system changes. If a (descriptive) stack model is used, the same stack depth distribution would be used for all load conditions. But a repetitions-based generative model shows that this is probably wrong. When the load is extremely low, there will be little if any overlap

log	*max deviation*
LANL CM5	0.043
SDSC Paragon ’95	0.282
SDSC Paragon ’96	0.135
CTC CP2	0.071
KTH SP2	0.113
SDSC SP2	0.055
Blue Horizon	0.043
DataStar	0.040

Table 4: *The maximal deviations observed when comparing each log with a global distribution composed from all of them.*

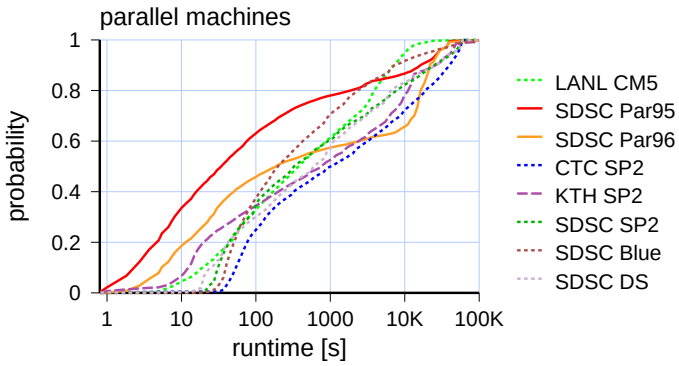

Figure 12: *Runtime distributions on different parallel machines.*

between repeated sequences of jobs, so stack distances should be very small. But when the load is high, more other jobs will intervene between repetitions, leading to higher stack distances. With a generative model we only need to create more sequences to increase the load, and the modification of the locality follows automatically.

Note that our approach to modeling locality of sampling can be viewed as a component in a more general model that simulates the process that generates the workload. Taking additional steps in this direction can be expected to lead to even better models that also exhibit locality of sampling. In particular, it is desirable to combine job repetitions with mechanisms that induce self-similarity. At present such comprehensive modeling is left for future work.

5 Workload Diversity

Our exposition so far has focused on locality of sampling in the time domain, where the distributions of workload attributes during a limited slice of time differ from the global distribution observed over a much longer period. In this section we generalize this idea, and show that the same technique may be applied to the spatial domain, by quantifying the difference between workloads of the same type that come from different sites.

Using the metrics devised in Section 3 to measure “spatial” rather than “temporal” locality means that the slices are not data from the same site at different times, but rather data from different sites (possibly but not necessarily at the same time). We exemplify this idea by comparing the parallel system logs we have used throughout this study. We can define a global distribution that includes the runtimes of jobs from *all* the logs, and then check the deviation between each log’s distribution and this global average. The results of doing so are shown in Table 4; note that each log is taken as a single slice, so the maximal deviation is given. Obviously the SDSC Paragon log is the farthest from the average, especially in 1995. The KTH log is the second farthest. All the rest are not so far, but still their distance is significant: checking what deviations we may expect to

		max deviation	
cluster	*jobs*	*job wgt*	*same wgt*
bruce	20,629	0.432	0.307
narwhal	269,902	0.067	0.078
tiger	152,950	0.109	0.110
bull	68,108	0.111	0.071
megaladon	7,998	0.103	0.066
dolphin	8,337	0.123	0.072
requin	50,448	0.062	0.052
whale	589,251	0.036	0.132
zebra	5,749	0.101	0.084
bala	10,280	0.129	0.091

Table 5: *The maximal deviations observed when comparing each log with a global distribution composed from all of them.*

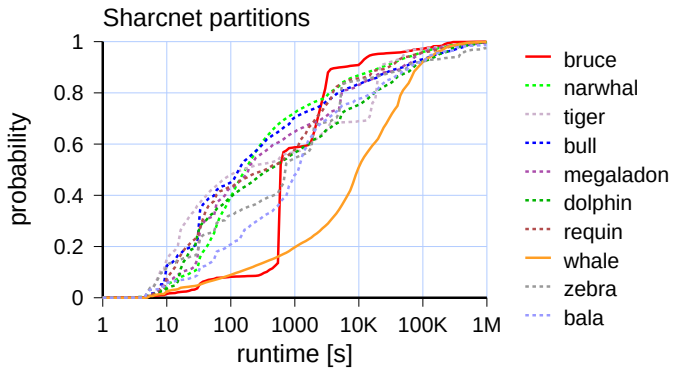

Figure 13: *Runtime distributions on different SHARCNET clusters.*

get due to random sampling led to results around 0.021 for all logs. The distributions themselves are shown in Fig. 12 and support these findings. They indicate that the main divergence of the Paragon and KTH distributions is at very short jobs of up to about 30 seconds.

An especially interesting dataset for studying workload diversity is provided by the SHARCNET system. As opposed to the above workloads, which come from different installations of different architectures at different times, SHARCNET is a collection of relatively new clusters based on the same architecture and located in a group of related academic institutions in Ontario, Canada. Would such uniformity lead to more similar workloads? As shown in Table 5 and Fig. 13, the answer is no.

Looking at the runtime distributions shows that eight of the ten clusters have similar distributions, but two are different: bruce has sharp modes at about 10m and 30–60m, and whale has much fewer jobs shorter than 1h and many more in the range of 1h to 1d. However, when looking at the divergence metric, whale has the lowest score! This happens because whale has a much higher load than any of the other clusters, and represents nearly 50% of all the jobs. As a result the global distribution is heavily biased towards the distribution of jobs that ran on whale, making whale appear closer and all the other clusters appear farther away.

A possible solution to such a bias is to redefine the global distribution. Instead of just tabulating all the jobs from all the sources, we can resample each source the same number of times. This puts all the sources on an equal footing. The results, also shown in Table 5, indeed reflect a global distribution that better represents the eight clusters that are more similar to each other.

6 Discussion and Conclusions

Locality of sampling refers to the fact that workloads often display a specific internal structure: successive samples are not independent of each other, but rather tend to be similar to each other. This applies to all workload attributes, and not only to those that denote location, such as memory addresses. In terms of a model whereby the workload is sampled from a distribution, this implies that successive samples be taken from the same part of the distribution.

A major contribution of this work is to suggest locality of sampling as a new way to look at and quantify short-range dependence. This was done by comparing the general distributions of the workload attributes of interest with the distributions that are observed during limited slices of time. We also showed that locality of sampling is distinct from short-range dependence and long-range dependence: a workload may display locality of sampling but zero autocorrelation at positive lags, and even if the autocorrelation is non-zero, it does not necessarily decay polynomially.

A second contribution was to suggest interleaved sequences of job repetitions as a generative model that can produce locality of sampling. This has two important advantages. First, it provides control over the degree of locality of sampling, by modifying the distribution of the number of repetitions used. Second, by virtue of being a generative model it adapts correctly when the workload as a whole is modified, e.g. when the load is increased. Of course, other models are also possible.

Locality of sampling in general, the job repetitions in particular, imply that on short time scales workloads display much less diversity than on long time scales, leading to a much more modal distribution. This lack of diversity on short time scales is an important phenomenon, as it may be exploited by systems that benefit from regularity in the workload. In particular, it is conjectured that such effects are required for the success of systems that adapt dynamically to workload conditions. If the workload were totally random, trying to adapt to the workload would be futile.

To date, job repetitions and localized deviations in workload distributions have received little if any attention. We contend that these important phenomena deserve to be taken into account and used as components of a more general workload model. Specifically, job repetitions should be a component of workload generation, alongside other means to create trends, correlations, and long-range dependence (an example of this is the work of Song et al. who combine repetitions with a Markovian model [27]). And our metrics for locality of sampling should be used both to characterize real workloads and to verify that synthetic workloads posses the desired attributes.

Our results also highlight the diversity observed in parallel system workloads, both along time in the same workload, and between workloads from different systems. This diversity implies that extreme care must be taken when trying to generalize performance results. Performance should always be evaluated for many different workloads, especially those that are known to be different from each other. Moreover, it might make sense to separate given workloads into shorter segments (that are each relatively homogeneous but different from each other) and consider the performance observed for each such segment. Such an analysis may also uncover the dependence of observed performance on specific workload features.

As locality of sampling is a newly identified phenomenon, much remains to be done. Directions for further research include the following. First, there is a need for a deeper exploration of the phenomenon itself, including the development of alternative metrics to measure it. For example, it may be necessary to consider other metrics when studying phenomena at very fine time scales.

log file	*proc's*	*duration*	*jobs*
LANL-CM5-1994-3.1-cln	1024	10/1994–9/1996	122,055
SDSC-Par-1995-2.1-cln	400	1/1995–12/1995	53,970
SDSC-Par-1996-2.1-cln	400	1/1996–12/1996	32,135
CTC-SP2-1996-2.1-cln	430	6/1996–5/1997	77,222
KTH-SP2-1996-2	100	9/1996–8/1997	28,489
SDSC-SP2-1998-3.1-cln	128	4/1998–4/2000	59,725
SDSC-BLUE-2000-3.1-cln	1152	4/2000–1/2003	243,314
SDSC-DS-2004-1	1664	3/2004–4/2005	96,089
SHARCNET-2005-1	6828	12/2005–1/2007	1,195,242

Table 6: *Main data logs used in this paper (available from the Parallel Workloads Archive).*

Second, there is a need to integrate locality of sampling with other workload attributes, such as self similarity, as comprehensive workload models should incorporate all known features of real workloads. Finally, we are just beginning to investigate the effects of locality of sampling on system behavior, and the opportunities for exploiting this behavior to improve system performance. In this context, it is also interesting to conduct a thorough comparison of locality of sampling and other models for locality, such as autoregressive, moving average, and stack models.

Acknowledgments

This research was supported in part by the Israel Science Foundation (grant no. 167/03). The idea that workloads should contain repetitive jobs was suggested to me by my PhD adviser, Larry Rudolph, more than 15 years ago, but at the time we didn't have any data with which to work. Many thanks are therefore due to all those who deposited their workload logs in the Parallel Workloads Archive, and made this research possible.

Appendix: Data Usage

The data analyzed in this paper comes from the Parallel Workloads Archive[2]. This contains accounting logs describing the activity on large scale production parallel supercomputers for periods of up to $2\frac{1}{2}$ years. The logs used are described in Table 6. The cleaned versions of the logs were used when available, which means that flurries of activity belonging to single users were removed [10], as well as other non-representative data such as records of site-specific automatic cleanup scripts that are fired up at the same time each day. This is significant as such singular activities tend to be repetitive and may therefore make a significant contribution to metrics aimed at measuring locality of sampling.

[2]http://www.cs.huji.ac.il/labs/parallel/workload/

References

[1] A. O. Allen, *Probability, Statistics, and Queueing Theory with Computer Science Applications*. Academic Press, 1978.

[2] V. Almeida, A. Bestavros, M. Crovella, and A. de Oliveira, "*Characterizing reference locality in the WWW*". In *Parallel & Distributed Inf. Syst.*, pp. 92–103, Dec 1996.

[3] M. F. Arlitt and C. L. Williamson, "*A synthetic workload model for Internet Mosaic traffic*". In *Summer Computer Simulation Conf.*, pp. 852–857, Jul 1995.

[4] M. Calzarossa and G. Serazzi, "*Construction and use of multiclass workload models*". *Performance Evaluation* **19(4)**, pp. 341–352, 1994.

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

[6] P. J. Denning, "*The locality principle*". *Comm. ACM* **48(7)**, pp. 19–24, Jul 2005.

[7] A. B. Downey, "*A parallel workload model and its implications for processor allocation*". *Cluster Computing* **1(1)**, pp. 133–145, 1998.

[8] B. Efron, "*Computers and the theory of statistics: thinking the unthinkable*". *SIAM Rev.* **21(4)**, pp. 460–480, Oct 1979.

[9] B. Efron and G. Gong, "*A leisurely look at the bootstrap, the jackknife, and cross-validation*". *The American Statistician* **37(1)**, pp. 36–48, Feb 1983.

[10] D. G. Feitelson and D. Tsafrir, "*Workload sanitation for performance evaluation*". In *IEEE Intl. Symp. Performance Analysis Syst. & Software.*, pp. 221–230, Mar 2006.

[11] D. Ferrari, "*On the foundation of artificial workload design*". In *SIGMETRICS Conf. Measurement & Modeling of Comput. Syst.*, pp. 8–14, Aug 1984.

[12] R. Gibbons, "*A historical application profiler for use by parallel schedulers*". In *Job Scheduling Strategies for Parallel Processing*, pp. 58–77, Springer Verlag, 1997. Lect. Notes Comput. Sci. vol. 1291.

[13] F. Hernández-Campos, K. Jeffay, and F. D. Smith, "*Tracking the evolution of web traffic: 1995–2003*". In 11th *Modeling, Anal. & Simulation of Comput. & Telecomm. Syst.*, pp. 16–25, Oct 2003.

[14] S. Hotovy, "*Workload evolution on the Cornell Theory Center IBM SP2*". In *Job Scheduling Strategies for Parallel Processing*, pp. 27–40, Springer-Verlag, 1996. Lect. Notes Comput. Sci. vol. 1162.

[15] R. Jain, *The Art of Computer Systems Performance Analysis*. John Wiley & Sons, 1991.

[16] 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.

[17] L. Kleinrock, *Queueing Systems, Vol II: Computer Applications*. John Wiley & Sons, 1976.

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

[19] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "*On the self-similar nature of Ethernet traffic*". *IEEE/ACM Trans. Networking* **2(1)**, pp. 1–15, Feb 1994.

[20] 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.

[21] D. Michie, "*Memo functions and machine learning*". *Nature* **218(5136)**, pp. 19–22, Apr 6 1968.

[22] T. D. Nguyen, R. Vaswani, and J. Zahorjan, "*Parallel application characterization for multiprocessor scheduling policy design*". In *Job Scheduling Strategies for Parallel Processing*, pp. 175–199, Springer-Verlag, 1996. Lect. Notes Comput. Sci. vol. 1162.

[23] V. Paxson and S. Floyd, "*Wide-area traffic: the failure of Poisson modeling*". *IEEE/ACM Trans. Networking* **3(3)**, pp. 226–244, Jun 1995.

[24] L. R. Rabiner, "*A tutorial on hidden Markov models and selected applications in speech recognition*". *Proc. IEEE* **77(2)**, pp. 257–286, Feb 1989.

[25] W. Shi, M. H. MacGregor, and P. Gburzynski, "*Synthetic trace generation for the Internet: an integrated model*". In *Intl. Symp. Performance Evaluation of Computer and Telecommunication Syst.*, pp. 471–477, Jul 2004.

[26] W. Smith, I. Foster, and V. Taylor, "*Predicting application run times using historical information*". In *Job Scheduling Strategies for Parallel Processing*, pp. 122–142, Springer Verlag, 1998. Lect. Notes Comput. Sci. vol. 1459.

[27] B. Song, C. Ernemann, and R. Yahyapour, "*Parallel computer workload modeling with Markov chains*". In *Job Scheduling Strategies for Parallel Processing*, pp. 47–62, Springer Verlag, 2004. Lect. Notes Comput. Sci. vol. 3277.

[28] J. R. Spirn, *Program Behavior: Models and Measurements*. Elsevier North Holland Inc., 1977.

[29] A. Streit, "*A self-tuning job scheduler family with dynamic policy switching*". In *Job Scheduling Strategies for Parallel Processing*, pp. 1–23, Springer Verlag, 2002. Lect. Notes Comput. Sci. vol. 2537.

[30] D. Talby and D. G. Feitelson, "*Improving and stabilizing parallel computer performance using adaptive scheduling*". In 19th *Intl. Parallel & Distributed Processing Symp.*, Apr 2005.

[31] D. Thiébaut, J. L. Wolf, and H. S. Stone, "*Synthetic traces for trace-driven simulation of cache memories*". *IEEE Trans. Comput.* **41(4)**, pp. 388–410, Apr 1992. (Corrected in *IEEE Trans. Comput.* **42(5)** p. 635, May 1993).

[32] M. Wang, A. Ailamaki, and C. Faloutsos, "*Capturing the spatio-temporal behavior of real traffic data*". *Performance Evaluation* **49(1-4)**, pp. 147–163, Aug 2002.

[33] P. R. Wilson, M. S. Johnstone, M. Neely, and D. Boles, "*Dynamic storage allocation: a survey and critical review*". In *Intl. Workshop Memory Management*, Sep 1995.

[34] L. Zhang, Z. Liu, A. Riabov, M. Schulman, C. Xia, and F. Zhang, "*A comprehensive toolset for workload characterization, performance modeling, and online control*". In *Computer Performance Evaluations, Modelling Techniques and Tools*, P. Kemper and W. H. Sanders (eds.), pp. 63–77, Springer-Verlag, Sep 2003. Lect. Notes Comput. Sci. vol. 2794.

[35] Q. Zhang, A. Riska, W. Sun, E. Smirni, and G. Ciardo, "*Workload-aware load balancing for clustered web servers*". *IEEE Trans. Parallel & Distributed Syst.* **16(3)**, pp. 219–233, Mar 2005.