Comparing Logs and Models of Parallel Workloads Using the Co-plot Method

David Talby*, Dror G. Feitelson*, Adi Raveh†

* Institute of Computer Science
† Department of Business Administration
The Hebrew University, Jerusalem, Israel
{davidt,feit}@cs.huji.ac.il

Abstract. We present a multivariate analysis technique called Co-plot that is especially suitable for samples with many variables and relatively few observations, as the data about workloads often is. Observations and variables are analyzed simultaneously. We find three stable clusters of highly correlated variables, but that the workloads themselves, on the other hand, are rather different from one another. Synthetic models for workload generation are also analyzed, and found to be reasonable; however, each model usually covers well one machine type. This leads us to conclude that a parameterized model of parallel workloads should be built, and we describe guidelines for such a model. Another feature that the models lack is self-similarity: We demonstrate that production logs exhibit this phenomenon in several attributes of the workload, and in contrast that the none of the synthetic models do.

1. Introduction

A notion of the workload a system will face is necessary in order to evaluate schedulers, processor allocators, or make most other design decisions. Two kinds of workloads are typically used: A trace of a real production workload, or the output of a synthetic statistical model.

Production logs have the advantage of being more realistic, and abstain as much as possible from making assumptions about the modeled system. At least three assumptions, however, are always there. First, we believe that we can draw conclusions from past workloads and learn about future ones. Second, we believe that we can infer from one installation – one scheduler, users set, and hardware configuration – about other ones. And third, we believe that the log contains no errors.

In order to use a production log as a model, we must answer yes to all three questions. In reality, the third issue – correctness of the log – is almost always questioned by mysterious jobs that exceeded the system's limits, undocumented downtime, dedication of the system to certain users, and other 'minor' undocumented administrative changes which distort the users' true wishes [6,15]. The first two assumptions – similarity between configurations and along time – will be shown in this paper to be generally unjustifiable as well.

Since using production traces suffers from apparent severe problems, researchers have turned to the other alternative, and have offered several synthetic models of parallel workloads [4,5,7,14,20]. The idea of basing models on measurements is not new [1,11], but practical work in the area was very little [3] until 1996. Models have advantages over production logs by putting all the assumptions "on the table", and by allowing their user to easily vary the model's parameters in order, for example, to generate a workload for a machine with a given number of processors.

The problem with models is, of course, the need to correctly build them. This paper compares ten production workloads with the generated output of five synthetic models, and maps each model to the production environment to which it is closest. But we also go further, and try to identify desired properties of parallel workload models for the models yet to come: Which variables should be modeled, and which should not? What can we tell about the distribution of these variables? What can we tell about the correlation of these variables and the unmodeled ones? Should we create a single model, or a customizable one? And how should the model be altered to match a needed number of processors, the load, and so forth?

In order to reach the answers, we use a new statistical method called Co-Plot, which is tailored for situations in which few observations but many variables about them are available. Section 2 presents Co-plot, and section 3 presents the data set used for the analysis. Sections 4, 5 and 6 analyze the production logs from several angles, which allows for a comparison with synthetic models available today in section 7, and a discussion about the implications in section 8. Section 9 deals with self-similarity, which is shown to differentiate between the production logs and the synthetic models.

2. Co-Plot

Classical multivariate analysis methods, such as cluster analysis and principal component analysis, analyze variables and observations separately. Co-Plot is a new technique which analyzes them simultaneously. This would mean, for example, that we'll be able to see, in the same analysis, clusters of observations (workloads in our case), clusters of variables, the relations between clusters (correlation between variables, for example) and a characterization of observations (as being above average in certain variables and below in others). The technique has been used before mostly in the area of economics [18,23].

Co-plot is especially suitable for tasks in which there are few observations and relatively many variables – as opposed to regression based techniques, in which the number of observations must be an order of magnitude larger than the number of variables. This is crucial in our case, in which there are few workloads (ten production ones and five synthetic ones), and just as many variables.

Co-plot's output is a visual display of its findings. It is based on two graphs that are superimposed on each other. The first graph maps the n observations into a two-dimensional space. This mapping, if it succeeds, conserves distance: observations that are close to each other in p dimensions are also close in two dimensions, and vice versa. The seconds graph consists of p arrows, representing the variables, and shows the direction of the gradient along each one.

Given an input matrix $Yn\times p$ of p variable values for each of n observations (see for example Table 1), the analysis consists of four stages. The first is to normalize the variables, which is needed in order to be able to relate them to each other, although each has different units and scale. This is done in the usual way. If Y_j is the j'th variable's mean, and D_j is its standard deviation, then Y_{ij} is normalized into Z_{ij} by:

$$Z_{ij} := (Y_{ij} - \overline{Y_j}) / D_j \quad (1)$$

In the second stage, we choose a measure of dissimilarity $S_{ik} \geq 0$ between each pair of observations (rows of $Zn\times p$). A symmetric $n\times n$ matrix is produced from all the different pairs of observations. To measure S_{ik}, we use city-block distance – the sum of absolute deviations – as a measure of dissimilarity:

$$S_{ik} = \sum_{j=1}^{p} \left| Z_{ij} - Z_{kj} \right| \quad (2)$$

In stage three, the matrix S_{ik} is mapped by means of a multidimensional scaling (MDS) method. Such an algorithm maps the matrix S_{ik} into an Euclidean space, of two dimensions in our case, such that 'close' observations (with a small dissimilarity between them) are close to each other in the map, while 'distant' ones are also distant in the map. Formally the requirement is as follows. Consider two observations, i and k, that are mapped a distance of d_{ik} from each other. We want this to reflect the dissimilarity S_{ik}. But this is actually a relative measure, and the important thing is that:

$$S_{ik} < S_{lm} \text{ iff } d_{ik} < d_{lm}$$

The MDS we use is Guttman's Smallest Space Analysis, or SSA [12]. SSA uses the coefficient of alienation Θ as a measure of goodness-of-fit. The smaller it is, the better the output, and values below 0.15 are considered good. The intuition for Θ comes directly from the above MDS requirement: A success of fulfilling it implies that the product of the differences between the dissimilarity measures and the map distances are positive. In a normalized form, we define:

$$\mu = \frac{\sum_{i,k,l,m} (S_{ik} - S_{lm})(d_{ik} - d_{lm})}{\sum_{i,k,l,m} \left| S_{ik} - S_{lm} \right| \left| d_{ik} - d_{lm} \right|} \quad (3)$$

Thus μ can attain the maximal value of 1. This is then used to define Θ as follows:

$$\Theta = \sqrt{1 - \mu^2} \quad (4)$$

The details of the SSA algorithm are beyond the scope of this paper, and presented in [12]. It is a widely used method in social sciences, and several examples along with intuitive descriptions can be found in [21].

In the fourth stage of the Co-plot method, p arrows are drawn on the Euclidean space obtained in the precious stage. Each variable j is represented by an arrow j,

emerging from the center of gravity of the n points. The direction of each arrow is chosen so that the correlation between the actual values of the variable j and their projections on the arrow is maximal (the arrows' length is undefined). Therefore, observations with a high value in this variable should be in the part of the space the arrow points to, while observations with a low value in this variable will be at the other side of the map.

Moreover, arrows associated with highly correlated variables will point in about the same direction, and vice versa. As a result, the cosines of angles between these arrows are approximately proportional to the correlations between their associated variables.

The goodness-of-fit of the Co-plot technique is assessed by two types of measures, one for stage 3 and another for stage 4. In stage 3, a single measure – the coefficient of alienation in our case – is used to determine the quality of the two-dimensional map. In stage 4, p separate measures – one for each variable – are given. These are the magnitudes of the p maximal correlations, that measure the goodness of fit of the p regressions. These correlations help in deciding whether to eliminate or add variables: Variables that do not fit into the graphical display, namely, have low correlations, should in our opinion be removed. Therefore, there is no need to fit all the 2^p subsets of variables as in other methods that use a general coefficient of goodness-of-fit. The higher the variable's correlation, the better the variable's arrow represents the common direction and order of the projections of the n points along the axis it is on.

3. The Data Set

Over several years we have obtained both a set of production workloads and the source codes to generate a number of synthetic workloads. As part of the current study, all workloads were translated to the standard workload format, and are freely available to all researchers in the field from the parallel workloads archive at URL http://www.cs.huji.ac.il/labs/parallel/workload. We would also like to encourage others to produce logs and source codes whose output is in this format, in order to create a growing library of quickly accessible and reliable data, that would ease validating a research on many workloads at once.

Traces of real production workloads were available from six machines: The NASA Ames iPSC/860 [9,24], the San Diego Supercomputing Center Paragon [24], the Cornell Theory Center SP2 [13], The Swedish Institute of Technology SP2, the Los Alamos National Lab CM-5, and the Lawrence Livermore National Lab Cray T3D. The Los Alamos and San Diego logs are displayed as three observations: The entire log, the interactive jobs only, and the batch jobs only. This gives a total of ten observations of production workloads. The characteristics of these workloads are given in table 1.

As Co-plot encourages it, the logs were tested for as many variables (attributes of the workloads) as possible. The following variables were measured for each workload:

1. The **number of processors** in the system.

2. **Scheduler Flexibility**. There were essentially three schedulers in this sample: the NQS batch queuing system, the EASY scheduler which uses backfilling, and gang schedulers. We ranked them in this ascending order, from 1 to 3.
3. **Processor Allocator Flexibity**. There were again three ranks, in this order of increasing flexibility: Allocation of partitions with power-of-2 nodes, limited allocation (meshes, etc.), and unlimited allocation (where any arbitrary subset of the nodes can be used).
4. **Runtime Load**, or the percent of available node seconds that were actually allocated to jobs. This is calculated as the sum of runtime multiplied by number of processors over all jobs, divided by the product of the number of processors in the machine multiplied by the log duration.
5. **CPU Load**, which is the percent of actual CPU work out of the total available CPU time during the log's lifetime. CPU times are the part of runtime in which the job actually processed; this however was missing from two workloads, and its definition is vague in some of the others, so focus was given to the runtime load.
6. **Normalized number of executables**. As some logs are much longer than others, it is not surprising that more executables are represented in them. We therefore normalize by dividing the number of observed executables by the total number of jobs in the log. A lower number indicated more repeated requests to run the same executable.
7. **Normalized number of users**. As above.
8. Percent of **successfully completed jobs**.
9. Median and 90% interval of the distribution of **runtimes** of jobs.
10. Median and 90% interval of the distribution of **degree of parallelism**, i.e. the number of processors used by each job.
11. Median and 90% interval of the distribution of **normalized degrees of parallelism**. This gives the number of processors that would be used out of a 128-processor machine. It is calculated as the percent of available processors that jobs used, multiplied by 128. It enables the decoupling of conclusions about the effect of machine size and the effect of parallelism.
12. Median and 90% interval of the distribution of **total CPU work** (over all processors of the job).
13. Median and 90% interval of the distribution of **inter-arrival times**.

As shown in [6], the average and standard deviation of these fields are extremely unstable due to the very long tail of the involved distributions. Removing the 0.1% 'taily' jobs from a workload, for example, could change the average by 5% and the CV by 40%. These findings follow similar ones in [16], and mean that the very big jobs must never be removed from workloads as outliers. On the other hand, most errors in traces are often in this area of the distribution (what do you do with a job that lasted more than the system allows?). Therefore, it is preferable to use order moments, such as the median and intervals. In this case, the 90% interval – difference between the 95% and 5% percentiles – was used; the 50% interval was also tested, and gave virtually the same results.

Using normalized degree of parallelism is preferable whenever possible, since it enables the decoupling of conclusions about the effect of machine size and the effect of parallelism. In our case we treat jobs as if they requested from a 128-node machine.

Variable		CTC	KTH	LANL	LANL inter.	LANL batch	LLNL	NASA	SDSC	SDSC inter.	SDSC batch
Machine processors	MP	512	100	1024	1024	1024	256	128	416	416	416
Scheduler flexibility	SF	2	2	3	3	3	3	1	1	1	1
Allocation flexibility	AL	3	3	1	1	1	2	1	2	2	2
Runtime load	RL	0.56	0.69	0.66	0.02	0.65	0.62	N/A	0.7	0.01	0.69
CPU load	CL	0.47	0.69	0.42	0	0.42	N/A	0.47	0.68	0.01	0.67
Norm. Executables	E	N/A	N/A	0.0008	0.0019	0.0012	0.0329	0.0352	N/A	N/A	N/A
Norm. Users	U	0.0086	0.0075	0.0019	0.0049	0.0032	0.0072	0.0016	0.0012	0.0021	0.0029
% Completed jobs	C	0.79	0.72	0.91	0.99	0.85	N/A	N/A	0.99	1.00	0.97
Runtime median	Rm	960	848	68	57	376.00	36	19	45	12	1812
Runtime interval	Ri	57216	47875	9064	267	11136	9143	1168	28498	484	39290
Processors median	Pm	2	3	64	32	64.00	8	1	5	4	8
Processors interval	Pi	37	31	224	96	480.00	62	31	63	31	63
Norm. proc. Median	Nm	0.76	3.84	8.00	4.00	8.00	4.00	1.00	1.54	1.23	2.46
Norm. proc. Interval	Ni	14.10	39.68	28.00	12.00	60.00	31.00	31.00	19.38	9.54	19.38
CPU work median	Cm	2181	2880	256	128	2944	384	19	209	86	9472
CPU work interval	Ci	326057	355140	559104	2560	1582080	455582	19774	918544	3960	1754212
Inter-arrival median	Im	64	192	162	16	169	119	56	170	68	208
Inter-arrival interval	Ii	1472	3806	1968	276	2064	1660	443	4265	2076	5884

Table 1. Data of production workloads

Since not all workload traces had all the required fields, missing values were approximated. These are all the assumptions that were made:

1. If one of CPU load and runtime load were missing, the other of the two fields was used. This was done in the NASA and LLNL workloads.
2. If the submit time of jobs was not known but the time the job started running (after possibly being in a queue) was, the inter-arrival time was based on this start time. This was necessary in NASA, LLNL, and the interactive workloads.
3. In the NASA log, total CPU work wasn't given and was approximated by the product of runtime and degree of parallelism. In the LLNL log, the opposite was done: The runtime was approximated by the total work divided by parallelism.

The synthetic models usually only offer values for the inter-arrival times, runtimes and degree of parallelism. They were only compared to the production workloads in these fields, of course, and the remaining variables were discarded.

4. Production Workloads: Variables

Running Co-plot on all variables resulted in some of them having low correlations, and they were removed. These variables were the number of processors in the machine, the scheduler flexibility, the normalized number of users, the normalized number of executables, and the percent of completed jobs. This basically means that

these attributes of a workload neither support nor refute the information derived from other variables. They are either irrelevant or belong to a different explanation universe.

Two other variables, the CPU load and the processor allocation flexibility, were also removed from the final map (Figure 1), but will still be analyzed. The correlations of these variables was slightly lower than that of the others, therefore removing them improved the output (in the goodness of fit sense), but we can still deduce about them from their would-be direction if they weren't removed. To represent the degree of parallelism, the normalized variant was use, although the un-normalized one gives almost exactly the same result. As discussed in the previous section, in such a case the normalized variant is preferable. The map in Figure 1 has a coefficient of alienation of 0.07 and an average of variable correlations of 0.88 with a minimum of 0.83. These are generally considered as excellent goodness of fit values.

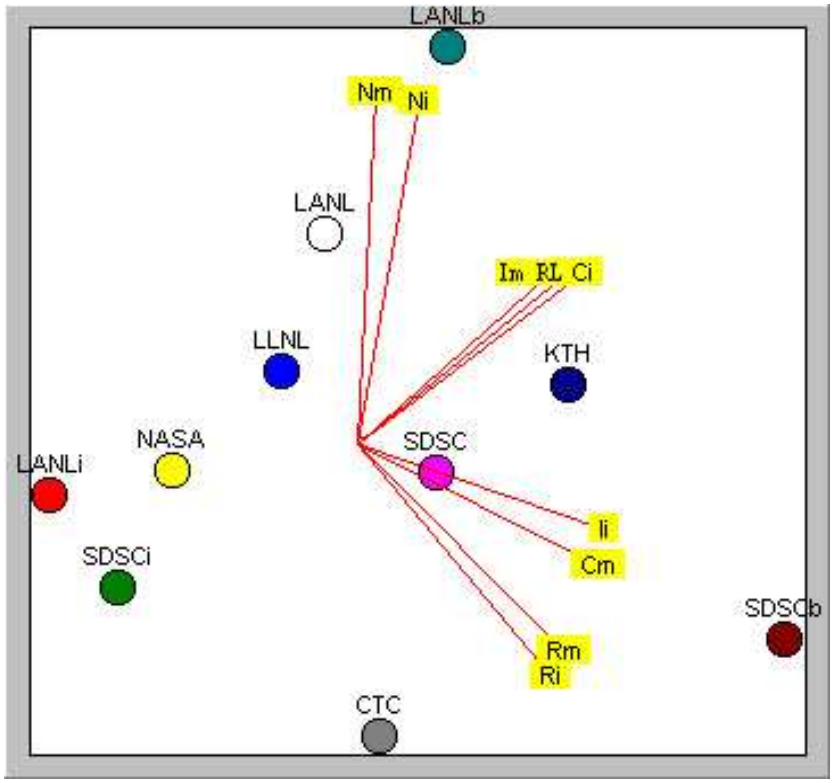

Fig. 1. Co-plot output of all production workloads
See Table 1 for variable signs

First, it is clear that there are four clusters of highly correlated variables. Clockwise:

1. The median and interval of the normalized degree of parallelism.
2. The median of inter-arrival times, interval of total CPU work, and runtime load.

3. The median of total CPU work and the interval of inter-arrival times. The CPU load (uncharted here) also belongs to this cluster.
4. The median and interval of job runtimes. The processor allocation flexibility (uncharted here) also belongs to this cluster.

It should be noted, however, that in some of the other runs (with more variables included, or some workloads excluded), the third cluster disappears: The CPU work median (Cm) joins the fourth cluster, and the inter-arrival times interval (Ii) joins the second.

The map tells a lot about the types of distributions that should be used for modeling workloads. Both runtimes and the degree of parallelism exhibit a high positive correlation between the median and interval of the distribution. This means that systems that allow higher runtimes and parallelism also exhibit a more varied stream of requests. This probably occurs because common administrative tools, such as limiting the maximal runtime, affect both the median and interval of the observed runtime in the same way. We suspect these phenomena to be linked to administrative constraints in general, but this could not be quantitatively checked.

Inter-arrival times and total CPU work, however, require a different treatment. Although their median and interval are positively correlated, the correlation is far from being full. This result repeats for other analyses for these two variables. Since none of the synthetic models published so far model runtime and actual CPU time separately, the focus has been put on runtimes.

The fact that the processor allocation flexibility measure and the median and interval of runtimes fall in the same cluster hints that systems which are more flexible in their allocation attract, on average, longer jobs. Note that the scheduler, in contrast, was not found to have a significant effect on the other variables. This is a first move towards a highly needed research about the changes in users' requests due to their system's constraints – users learn the system over time, and change their behavior accordingly. When looking at production workloads, we only see the distorted picture, not the "true workload" by which we wish to design future systems [15].

Apart from defining variable clusters, we can also infer about the correlation between clusters. The first cluster, for example, is positively correlated with the second cluster, has a small negative correlation with the third, and a strong negative one with the fourth. The second cluster is positively correlated with the third cluster, but not correlated with the fourth one. The third and fourth clusters are positively correlated.

One should be careful not to misinterpret these findings – note that they relate to whole workloads, not jobs. For example, the negative correlation between runtime (fourth cluster) and degree of parallelism (first cluster) means that systems with high average parallelism exhibit lower average runtimes, not that jobs that use many processors are shorter (in fact, the opposite of this was demonstrated in [6,10]). The reason for this may, for example, be that systems with more processors tend to enforce tighter runtime limits on jobs. This is merely a hypothesis, triggered by the observation that systems with fewer processors often try to compensate for this by offering more flexible policies.

Another significant finding from the correlations between variable clusters is that the processor allocation flexibility of a system is positively correlated to the median of the total CPU work done in it, and is uncorrelated to its runtime load. Or, in short,

systems whose allocation schemes are more flexible allow bigger jobs to run without affecting the average load. In contrast, neither the scheduler nor the number of processors in the system seems to have such an effect.

There is more data to be studied from the correlations between the second, third and fourth cluster, but since as mentioned before the third cluster sometimes melts into the other two, any such conclusions are dangerous. Only stable findings are reported.

5. Production Workloads: Observations

Two workloads in Figure 1 seem to be outliers, which 'stretch' the map in order to accommodate them – the batch jobs of LANL and SDSC, marked 'LANLb' and 'SDSCb' respectively. The LANL batch jobs are way above average in the normalized number of used processors, and also exhibit high inter-arrival times combined with low runtimes and runtime intervals. The high degree of parallelism is probably a result of the fact that the system had static partitions, all powers of two, of which the smallest one has 32 processors. The SDSC workload has very long runtime and total CPU work averages, and an extreme interval of inter-arrival times as well. The SDSC is also relatively inflexible in its processor allocation.

In order to understand the relations between the other workloads, and get another picture of the variables arena, another analysis is presented here that includes the same workloads as those used in figure 1, but without the two batch workloads. The variables are also the same, except the degree of parallelism which is now not normalized – the normalized variables had too low correlations. The Figure 2 map has a coefficient of alienation of 0.01 and an average of variable correlations of 0.88.

Comparing the variable clusters here to those of Figure 1 shows that what was there the third cluster indeed broke; the interval of inter-arrival times variable joined the second cluster, and the median of CPU work variable joined the fourth. All our conclusions remain the same.

At the lower left part of the map are the two interactive workloads, again of LANL and SDSC respectively. Along with the NASA Ames log[1], these three workloads seem to form the only natural observations cluster in this map: All other observations are evenly spread out across the map. This means that apart from a grouping of the interactive workloads – based on two observations only – the workloads exhibited by different systems are very different from one another.

Since Co-plot analyzes observations and variables together, it is not only possible to see clusters of observations, but also to identify their nature. For example, the interactive jobs are characterized by being way below average on all variables: They have a shorter average inter-arrival time, and also shorter runtimes and degrees of parallelism. Such facts are deduced in Co-plot from the arrows: The projection of a point (workload, in our case) on a variable's arrow should be proportional to its

[1] A caveat is that 57% of the jobs in the NASA log were small jobs used to periodically check the system availability, which obviously causes a strong bias towards low varuable values.

distance from the variable's average, where above average is in the direction of the arrow and vice versa.

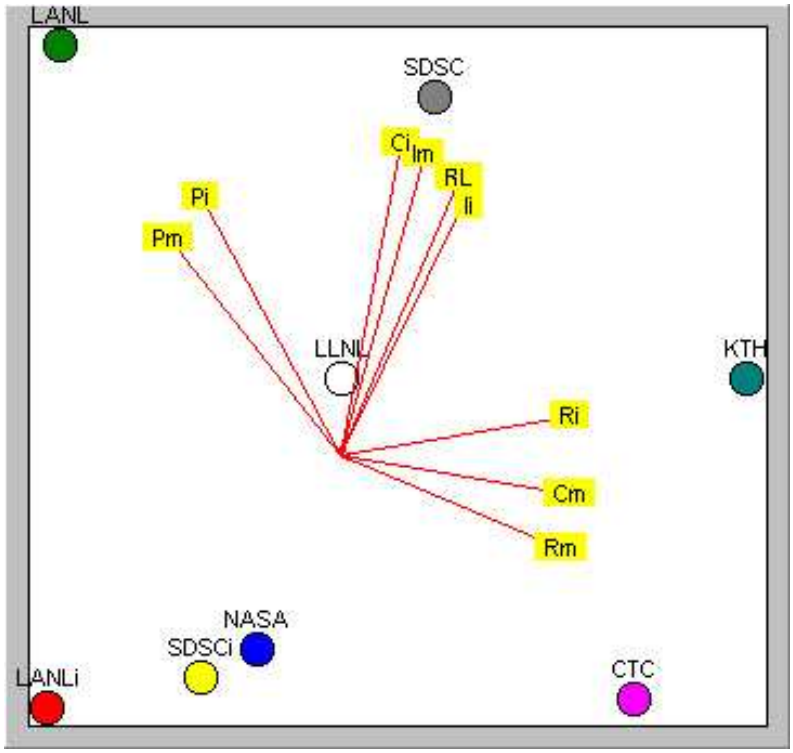

Fig. 2. Production workloads of all except the batch workloads

In the same manner, the CTC workload (in the lower right side of the map) has very long runtimes but little parallelism, while the LANL workload (upper left) has a very high degree of parallelism, but below average runtimes. The LLNL workload seems to be the average – it is very close to the center of gravity in almost all variables.

Note that although we can see that the workloads are 'far' from each other, this notion of distance is always relative to the other observations in this analysis. This happens because all variables must be normalized – otherwise we can't compare relations between them – and means that we should beware if attaching 'real' distance to Co-plot's output.

6. Production Workloads Over Time

Logs of production workloads from previous years are typically used as a model of the pattern of requests for next year. But since users learn the system as time passes and adjust their behavior accordingly, administrators fine-tune the system

continuously, and the dominant projects on machines change every few months, it is unclear whether this is justified.

Co-plot allows us to test just that, by mapping several consecutive periods of logged work on the same machine, and mapping them together with the other workloads. If past workloads were indeed good indications of the near future, we would expect consecutive workloads on the same machine to be mapped close to each other. Only two workloads in our sample were long enough to test this – the LANL and the SDSC logs were each divided to four periods of six months each. The data for these partial logs is given in Table 2.

	LANL				SDSC			
	10/94-3/95	4/95-9/95	10/95-3/96	4/96-9/96	1/95-6/95	7/95-12/95	1/96-6/96	7/96-12/96
Runtime Load	0.76	0.83	0.24	0.73	0.66	0.67	0.76	0.65
CPU Load	0.43	0.52	0.16	0.48	0.65	0.66	0.72	0.63
Executables per Job	0.0016	0.0014	0.0034	0.0016	N/A	N/A	N/A	N/A
Users per Job	0.0038	0.0038	0.0076	0.0042	0.0021	0.0019	0.0023	0.0023
% of Completed Jobs	0.93	0.93	0.82	0.90	0.99	0.99	0.98	0.97
Runtime Median	62	65	643	79	31	21	73	527
Runtime Interval	7003	7383	11039	11085	29067	20270	30955	25656
Processors Median	64	32	64	128	4	4	4	8
Processors Interval	224	224	480	480	63	63	63	63
Norm. Procs. Median	8	4	8	16	1.23	1.23	1.23	2.46
Norm. Procs. Interval	28	28	60	60	19.38	19.38	19.38	19.38
CPU Work Median	128	256	7648	384	169	119	295	1645
CPU Work Interval	300320	394112	1976832	1417216	504254	612183	1235174	1141531
Inter-Arrival Median	159	167	239	89	180	39	92	206
Inter-Arrival Interval	1948	1765	2448	1834	2422	5836	4516	5040

Table 2: Data of production workloads divides to six months

Figure 3 includes the same workloads as Figure 1, except the addition of the eight new workloads. The four parts of the LANL workload are marked L1 through L4, and the four parts of the SDSC workload are marked S1 through S4. The full, interactive and batch workloads of both sites were also kept. Two variables were removed because of low correlation: The runtime load, and the interval of the inter-arrival times. This does not mean that they shouldn't be used in general – it may very well be the case that they do not fit well only with the LANL or SDSC logs, which are together 14 out of the 18 observations in Figure 3.

It is clear that the SDSC jobs are rather clustered, apart possibly from the last workload S4, which has slightly higher runtimes, degrees of parallelism, and inter-arrival times. The original full SDSC workload is some kind of average of its four parts, as was expected. On the other hand, the LANL workloads have a quiet first year (workloads L1 and L2, close to the original full LANL workload), but the second year is wildly different with L3 and L4, which are definite outliers.

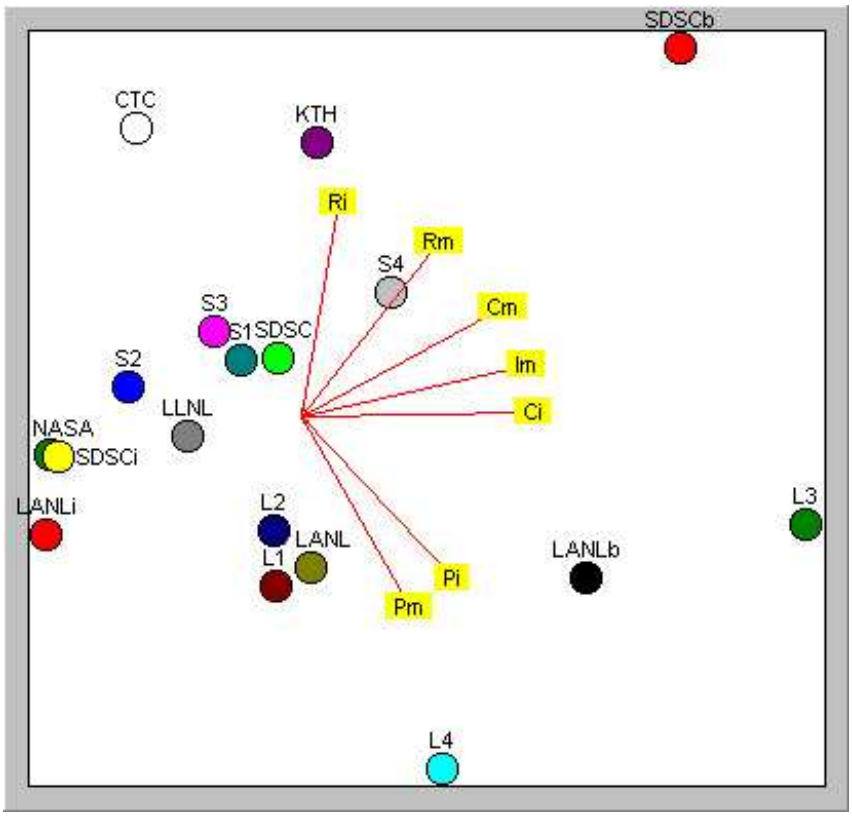

Fig. 3. Production workloads change over time

A clarification with Curt Canada of LANL indeed revealed that at the end of 1995 there was a significant change of usage of the CM-5. It approached the end of its life for grand challenge jobs, and only a couple of groups remained on the machine for special projects that were trying to finish. Fewer jobs of fewer users, mostly very long ones, were run in 1996.

Co-Plot could be used in this manner to test any new log, by dividing it into several parts and mapping it with all the other workloads. This should tell whether the log is homogeneous, and whether it contains time intervals in which work on the logged machine had unusual patterns.

7. Synthetic Workloads

Having taken a serious look at the raw data from production logs, we now turn to inspect what research has done so far, namely the synthetic models that are currently available. To make a long story short, the models are quite good – none is an

outrageous outlier – but each one is more representative of one or two production logs than of the others. Usually every researcher based his or her model on one log, and the model reflects this.

The five synthetic models available are the following. The first model was proposed by Feitelson in '96 [7]. This model is based on observations from several workload logs. Its main features are the hand-tailored distribution of job sizes (i.e. the number of processors used by each job), which emphasizes small jobs and powers of two, the correlation between job size and running time, and the repetition of job executions. In principle such repetitions should reflect feedback from the scheduler, as jobs are assumed to be re-submitted only after the previous execution terminates. Here we deal with a pure model, so we assume they run immediately and are resubmitted after their running time. The second model is a modification from '97 [8].

The model by Downey is based mainly on an analysis of the SDSC log [4,5]. It uses a novel log-uniform distribution to model service times (that is, the total computation time across all nodes) and average parallelism. This is supposed to be used to derive the actual runtime based on the number of processors allocated by the scheduler. Again, as we are dealing with a pure model here, we instead use the average parallelism as the number of processors, and divide the service time by this number to derive the running time.

Jann's model is based on a careful analysis of the CTC workload [14]. Both the running time and inter-arrival times are modeled using hyper Erlang distributions of common order, where the parameters for each range of number of processors are derived by matching the first 3 moments of the distribution.

The last model, by Lublin [20], is based on a statistical analysis of 4 logs. It includes a model of the number of processors used which emphasizes powers of two, a model of running times that correlates with the number of processors used by each job, and a model of inter-arrival times. While superficially similar to the Feitelson models, Lublin based the choice of distributions and their parameters on better statistical procedures in order to achieve a better representation of the original data.

Figure 4 is the Co-Plot output of all the production workloads and the five synthetic models that were tested. Since all models only model the inter-arrival times, runtimes, and degrees of parallelism, then the median and interval of each of these along with the implied used CPU times (runtimes multiplied by the degree of parallelism) were the only eight variables that could be used. All eight showed high correlations to the map; the average correlation is 0.89 and the coefficient of alienation is 0.06.

First, Uri Lublin's model places itself as the ultimate average. This result repeated in analyses under different variables and observations. So this model represents "the workload on the street", and when used to compare schedulers, for example, we can be sure that results will not be distorted by one out-of-line feature of the workload (of the variables analyzed here). However, most of the production workloads do have such out-of-line features, and are far from the center of gravity. Only the LLNL workload is close enough as to accept the model as a match.

Downey's model, as well as the two versions of Feitelson's model, match well the interactive workloads and the NASA one. This is probably due to the fact that the NASA log was the first to be published and seriously analyzed, and had a major effect on the creation of the earlier (e.g. 1997) models. In order to try to differentiate the

three models, the batch workloads, which were the outliers removed in Figure 2 as well, were removed and the analysis was re-run. The result was essentially the same, with a "zoom in" on the lower left part of Figure 4. Feitelson's 1997 model remained the closest to the interactive and NASA workloads, while his 1996 model was closer to the center of gravity and Downey's model was further out from it.

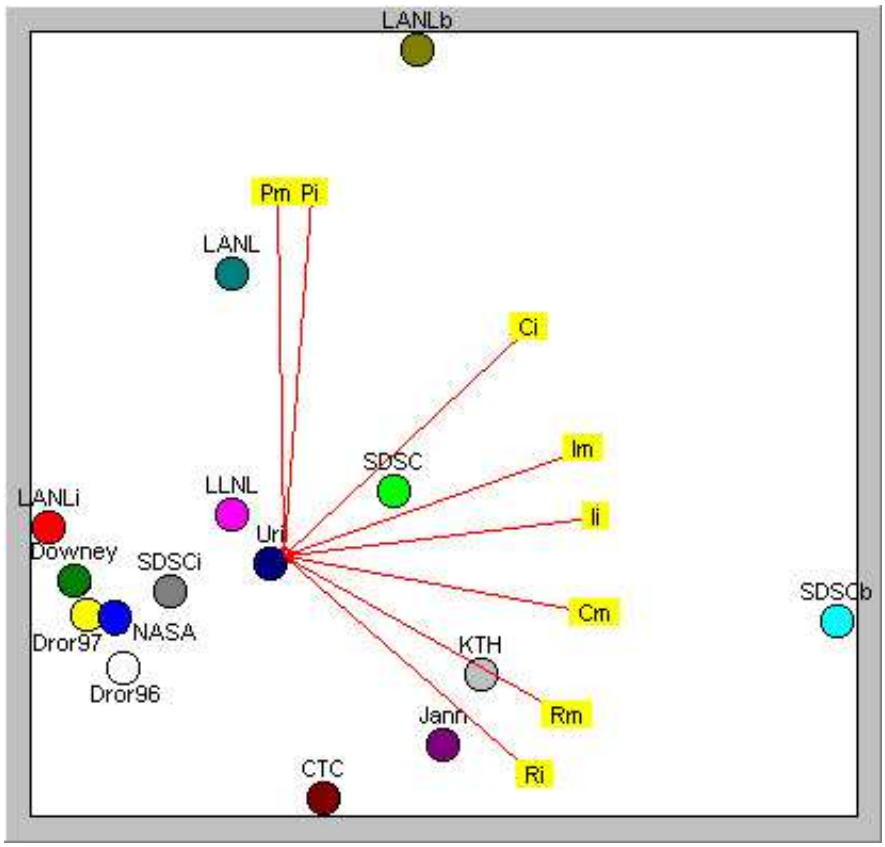

Fig. 4. Comparison of production and synthetic workloads

Jann's model was designed specifically to match the CTC workload. It is indeed the closest to it, but is also close to the KTH workload. The CTC and KTH offer very similar environments: Both are IBM SP2 machines, with slightly different versions of the RS/6000 processor, using LoadLeveler with EASY for scheduling, and offering a totally flexible processor allocation scheme. With a limited warranty – since only two observations support this – it seems that Jann's model is appropriate for this kind of environment.

The LANL and SDSC workloads have no model close to them, and the batch workloads of these two systems are still lonely outliers as well. This means that no workload model as it stands today models well the heavier batch jobs that these large

machines see. The focus seems to be the interactive jobs, which, according to Table 1, provide only a fraction of the total load a parallel system has to handle.

A quick look at the variable arrows of Figure 4 is also worthy. It is almost the same as that of Figure 1, which is good news: The synthetic workload models do not distort the 'real world' picture by assuming wildly incorrect distributions or correlations.

8. Implications for Modeling

From what we have seen so far, three major conclusions concerning the correct modeling of parallel workloads can be stated:

1. Model the runtime and degree of parallelism by a distribution that has an almost full correlation between the median and interval. A high positive such correlation also exists for the inter-arrival time.
2. A single model cannot truly represent all systems. It is better to parametrize by three variables: The medians of total CPU work, degree of parallelism, and inter-arrival time.
3. In order to alter the average load of a modeled workload, do not use any of the common techniques of multiplying the inter-arrival time, runtimes, or parallelism by a factor, or changing the lambda of an exponential distribution.

The first statement is supported by the map of Figure 1, which says much about the types of distributions that should be used for modeling workloads. Both runtimes and the degree of parallelism are characterized by the fact that the median and interval of the distribution are highly correlated. An analogous feature is found in the exponential distribution, in which the mean and variance are equal. Although the exponential distribution lacks the very long tail property workload modeling requires, variants such as two- and three-stage hyper-exponential distributions have been used in several recent models [7,20], which seems to be rationalized here.

Using an exponential distribution for the inter-arrival time is also popular, but it seems that although the correlation between the median and interval of it is positive, it is not full and a more sophisticated model is required.

The second statement is clear from Figure 2. Apart from the NASA workload being similar to the interactive ones, all the workloads are scattered across the space that they define. While Lublin's model represents the average, it does not closely represent any single model. But there are good news as well: Beyond telling us that a generic model must be a parameterized one, the Co-plot method also helps us find upon which variables we should parameterize. We should take one representative from each variables cluster, such that the representatives conserve the previously known map, and that their correlation is highest.

In our case, the best results (with a coefficient of alienation of 0.02 and an average correlation of 0.94) were achieved by using the processor allocation flexibility and the medians of the (un-normalized) degree of parallelism and the inter-arrival time (Figure 4). The processor allocation flexibility could be replaced by the median of total CPU work, to give a slightly lower but still excellent goodness of fit.

So, a general model of parallel workloads will accept these three parameters as input. It would use the highly positive correlations with other variables to assume

their distributions. But what should we give it? The processor allocation flexibility is usually known, since it is one of the published characteristics of the modeled computer. Therefore it seems like the best estimator for the level of total CPU work in the modeled system. As for the two other medians, a way is needed to determine whether it would be above, below or around average. While we can offer little guidance on how that could be done, apart from using past knowledge as section 6 suggests, we can tell what the averages are – the LLNL log seems to represent it, and Lublin's model looks like an even more accurate choice (Figure 4).

The third statement we make deals with the preferable way to alter a workload's load. There are basically three ways to raise the load: Lowering the inter-arrival time, raising the runtimes, and raising the degree of parallelism. The most common [19, 22] technique is to expand or condense the distribution of one of these three fields by a constant factor. Note that by doing so the median and interval (any interval) is also multiplied by the same factor.

Our choice, of which field to alter, should be derived from the correlations between the runtime load and the three variables that are candidates to change it. We would choose lowering inter-arrival times if it were negatively correlated with load, and raising runtimes or parallelism if they were positively correlated with it. By doing so, we minimize the side effects of raising the load on other features of the workload.

Using this logical criterion, we have mostly bad news. First, from Figure 1 it is clear that systems with a higher average load have a higher inter-arrival time median, not a lower one. Second, the runtimes of job are not correlated to the load. And third – this is the only optimistic part – the degree of parallelism is indeed positively correlated with load, but the correlation is far from full.

This means that a correct way to raise a system's load would end up with higher inter-arrival times, about the same runtimes, and somewhat more parallelism. None of the three simplistic ways to alter the load satisfy these conditions – they all contradict it. Correctly varying a given workload model's load is not as simple as it looks; it seems to require changes to the distributions of the inter-arrivals, runtimes, and parallelism that the variables we chose for our analysis do not expose.

Such findings, in this case about the right method to change a workload's average load, call for a generalization to a methodological rule. Since most modeled variables are correlated to each other, any assumption of the kind "in order to change X I'll change Y, and everything else will remain the same" is bound to be wrong.

9. Self-Similarity

Recent studies of traces of Ethernet network traffic [17], web server traffic [25], and file system requests [26] have revealed an unexpected property of these traces, namely that they are self-similar in nature. Intuitively, self-similar stochastic processes look similar and bursty across all time scales. Physical limitations, such as the finite bandwidth and lifetime of any network or server, inhibit true self-similar behavior, but the presence of self-similarity over considerably long amounts of time (months, in the case of multi-computers), makes this phenomenon of practical importance.

The major consequences of self-similarity are in the area of scheduling: The common theory of workload modeling usually assumes that while a single resource may exceed its resource consumption average, its aggregate resource requirements over time will have a low variance. In a self-similar system, this is not the case. Therefore, the discovery of self-similarity in parallel workloads should lead to a reassessment of known schedulers, to determine their compatibility with long-range dependence and highly variant workloads.

We have tested both the production and synthetic workloads for self-similarity using three different methods. A description of these methods – R/S analysis, Variance-time plots and Periodogram analysis – is given in the appendix, together with a formal definition of self-similarity. For a more thorough theoretical presentation of the issue, see [27].

The results are summarized in Table 3. The value given for each test and each variable is the Hurst parameter estimation for that variable by the test. In short, the Hurst parameter measures the degree of self-similarity in a time series: It is 0.5 if the series is not self-similar, and is closer to 1.0 the more self-similar it is. Although all three tests are only approximations and do not give confidence intervals to the value of the Hurst parameter, it seems certain that most workloads are self-similar in all of the tested variables. This new result adds to similar findings in networks and file systems, and hints that most "human generated" workloads, in which tens or more of people are involved in creating, will exhibit self-similarity to some degree.

	Used Processors			Run Time			Total CPU Time			Inter-Arrival Time		
	R/S	V-T	Per.	R/S	V-T	Per.	R/S	V-T	Per.	R/S	V-T	Per.
Variable	rp	vp	pp	rr	vr	pr	rc	vc	pc	ri	vi	pi
CTC	0.71	0.71	0.68	0.55	0.75	0.76	0.29	0.65	0.56	0.42	0.63	0.68
KTH	0.74	0.87	0.67	0.68	0.58	0.79	0.61	0.67	0.56	0.48	0.69	0.71
LANL	0.60	0.90	0.82	0.74	0.90	0.77	0.65	0.88	0.76	0.67	0.91	0.68
LANLi	0.96	0.81	0.91	0.80	0.80	0.84	0.71	0.79	0.70	0.86	0.59	0.84
LANLb	0.52	0.78	0.78	0.66	0.81	0.71	0.68	0.80	0.71	0.71	0.79	0.66
LLNL	0.84	0.74	0.84	0.88	0.74	0.69	0.77	0.69	0.72	0.56	0.43	0.71
NASA	0.61	0.68	0.84	0.53	0.66	0.56	0.43	0.60	0.55	0.60	0.35	0.51
SDSC	0.50	0.77	0.68	0.54	0.85	0.70	0.53	0.83	0.60	0.66	0.96	0.67
SDSCi	0.61	0.59	0.94	0.83	0.61	0.58	0.62	0.59	0.56	0.80	0.74	0.64
SDSCb	0.68	0.83	0.72	0.84	0.76	0.68	0.83	0.79	0.58	0.82	0.84	0.56
Lublin	0.47	0.47	0.48	0.55	0.80	0.67	0.55	0.80	0.67	0.45	0.49	0.47
Feitelson '97	0.64	0.62	0.80	0.72	0.62	0.72	0.67	0.58	0.70	0.49	0.49	0.54
Feitelson '96	0.72	0.57	0.65	0.26	0.61	0.69	0.26	0.60	0.68	0.55	0.48	0.50
Downey	0.46	0.49	0.50	0.54	0.48	0.49	0.60	0.47	0.49	0.55	0.46	0.49
Jann	0.69	0.57	0.59	0.49	0.49	0.49	0.64	0.51	0.51	0.61	0.50	0.54

Table 3: Estimations of Self-Similarity

As self-similarity was unknown when the synthetic models we use were created, it is plausible that they will not exhibit the phenomenon. We ran Co-plot on Table 3, without any of the variables used in the earlier sections. Adding the previous group of variables resulted in a high degree of alienation and low correlations for many variables, in both variable groups. This happens because of the two dimensional nature of Co-plot: When using too many variables that are not well correlated, two dimensions are just not enough to present a coherent picture. It is then necessary to use less variables, or split the available variables into groups that belong to different "explanation universes".

Figure 5 does not include three estimators out of the twelve given in Table 3: The R/S analysis estimation of the degree of parallelism and of the total used CPU time, and the periodogram estimator of the total used CPU time. These variables were removed because of relatively low correlations; however, they also pointed to the left direction of the map, as do all the other variables. All estimators of the degree of parallelism are equal by definition of self-similarity for the absolute and normalized versions, so there was no need to test both variables separately.

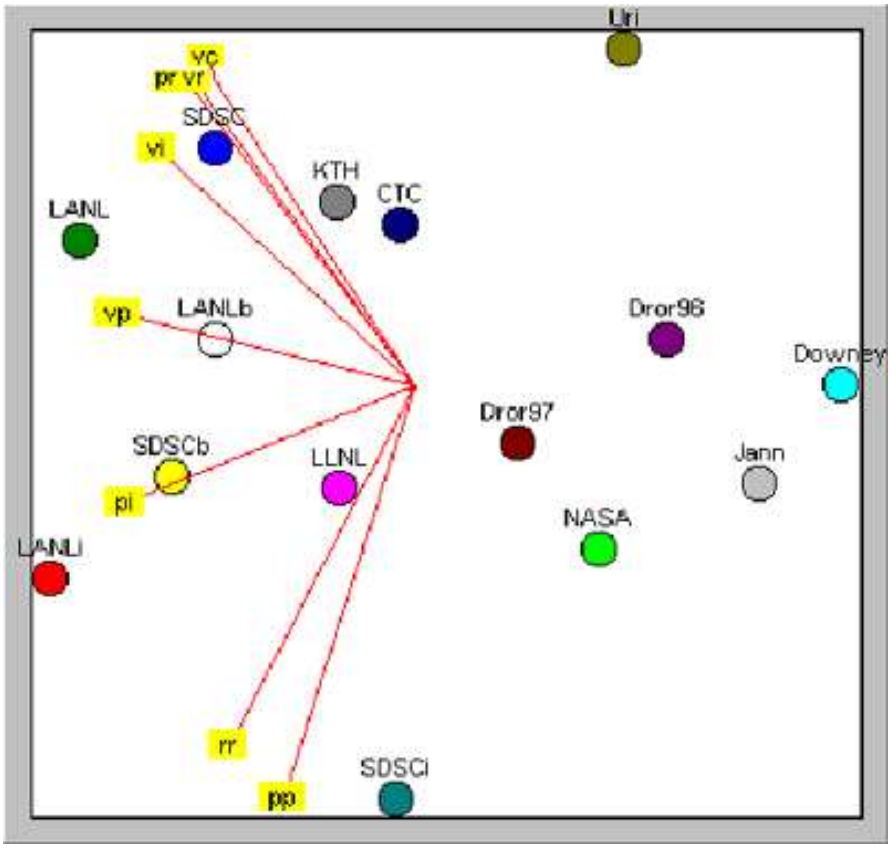

Fig. 5. Self Similarity Estimations of Logs and Models

There are two clear results. First, all the production workloads except for NASA Ames show some degree of self-similarity, while all the synthetic models do not. Uri Lublin's model, on the upper right corner, is apart from the other models, but this is not because of high Hurst parameter estimators but because of very low ones, which goes against what is seen in the production logs. The Feitelson '97 model has the highest self-similarity, possibly due to the inclusion of repeated job executions. In general, it is clear that the models do not capture the self-similar nature of parallel computer workloads.

Second, the three different estimators for the Hurst parameter have major differences. We expected the three estimators of each variable to be high correlated, but this happens only occasionally. For example, while the Variance-Time and Periodogram estimators for run time self-similarity are highly correlated, the R/S analysis estimator for the same variable is uncorrelated to them. The two estimators for inter-arrival time are almost uncorrelated, and the same goes for the degree of parallelism.

Therefore, it is best to refrain from conclusions in the spirit of "workload X has high run time self-similarity but low inter-arrival self-similarity". The only conclusion that is supported by all estimators is the fact that the production workloads are self-similar while the synthetic models are not, because all the arrows point leftwards – where the production workloads are.

Although further checking is required, it seems that computers with similar attributes produce similar self-similarity levels. For example, the CTC and KTH logs, which are both SP2 machines scheduled with EASY, are very close to one another, and the batch jobs of LANL and SDSC are also neighbors. We defer this issue for now, since finding out the definite causes of self-similarity requires more workloads, particularly ones coming from similar computers.

10. Conclusions and Future Research

This paper makes two important contributions. First, it presents the Co-plot technique, a multivariate statistical method tailored to the demands of the workload modeling field: It works well given few observations and many variables, dependent or not. Second, it provides new insights about the majority of production workloads and synthetic models available today, giving a clear view of what needs to be done next.

One path to continue by is to find more variables that correlate well with the ones already found, and are known for the modeled system. The major problem with the parametric model approach suggested in section 8 is the need to estimate the medians of the degree of parallelism and inter-arrival times. We wish to model a future system, so these are not known in advance. They should be replaced by known variables – we have tried the processor allocation scheme, scheduler, number of processors in the system, and the expected number of users. We intend to look further into robust estimators of the third moment, user or multi-class modeling attributes [2], and the self-similarity of the distributions [17].

A second path to be taken should try to estimate these unknown medians of a future system based on similar systems from the past. As we have seen, this approach seems to work in some cases but breaks down in others, and it remains to be discovered which changes made to a system – adding processors, replacing the scheduler, changing policies and so forth – cause which changes to the resulting workload.

A third question raised is the issue of changing the load of a workload. It was shown that the currently used techniques cause harmful (in the modeling sense) side effects to the workload, by contradicting the expected correlations between the altered variables. Finding the right way to control load is of practical concern to many experiments and statistical tests in this field of research.

Self-similarity is expected to play a significant role in future synthetic models, not only in the area of parallel computer workloads. The lack of a suitable model that represents self-similarity is apparent, and a new model is a near future requirement. However, although it is clear that none of the models exhibit self-similarity, the effect of this absence has not yet been determined, and this needs to be done as well.

We'd be more than happy to share the data sets and tools we used. The production workloads in standard workload format and the source codes of synthetic models are available from the online archive at http://www.cs.huji.ac.il/labs/parallel/workload. The Co-Plot program and workload analysis program (both under Windows) are available from the authors.

11. Acknowledgements

This work would not be possible without the help of many people who supplied logs, source codes, and explanations. We'd like to personally thank Bill Nitzberg, Reagan Moore, Allen Downey, Lars Malinowsky, Curt Canada, Moe Jette, and Joefon Jann.

The third author was partially supported by the Racannati Foundation.

References

1. A.K. Agrawala, J.M. Mohr and R.M. Byrant, "An approach to the workload characterization problem". *Computer* **9**(6), pp.18-32, June 1976.
2. Maria Calzarossa and Giuseppe Serazzi, "Construction and Use of Multiclass Workload Models". *Performance Evaluation* **19**(4), pp. 341-352, 1994.
3. Maria Calzarossa and Giuseppe Serazzi, "Workload Characterization: A Survey". *Proc. IEEE* **81**(8), pp. 1136-1150, Aug 1993.
4. Allen B. Downey, "A Parallel Workload Model and Its Implications for Processor Allocation". *6th Intl. Symp. High Performance Distributed Comput.*, Aug 1997.
5. Allen B. Downey, "Using Queue Time Predictions for Processor Allocation". In *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph (Eds.), Springer-Verlag, 1997, Lect. Notes Comput. Sci. vol. 1291, pp. 35-57.
6. Allen B. Downey and Dror G. Feitelson, "The Elusive Goal of Workload Characterization". *Perf. Eval. Rev.* **26**(4), pp. 14-29, Mar 1999.

7. D. G. Feitelson, "Packing schemes for gang scheduling". In *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph (Eds.), Springer-Verlag, 1996, Lect. Notes Comput. Sci. vol. 1162, pp. 89-110.
8. Dror G. Feitelson and Morris A. Jette, "Improved Utilization and Responsiveness with Gang Scheduling", In *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph (Eds.), Springer-Verlag, 1997, Lect. Notes Comp. Sci. vol. 1291, pp. 238-261.
9. D. G. Feitelson and B. Nitzberg, "Job characteristics of a production parallel scientific workload on the NASA Ames iPSC/860". In *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph (Eds.), Springer-Verlag, 1995, Lect. Notes Comput. Sci. vol. 949, pp. 337-360.
10. Dror G. Feitelson and Larry Rudolph, "Metrics and Benchmarking for Parallel Job Scheduling". In *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph (Eds.), Springer-Verlag, 1998, Lect. Notes Comput. Sci. vol. 1459, pp. 1-24.
11. D. Ferrai, "Workload characterization and selection in computer performance measurement". *Computer* **5**(4), pp. 18-24, Jul/Aug 1972.
12. Guttman, L., "A general non-metric technique for finding the smallest space for a configuration of points", *Psychometrica* **33**, pp. 479-506, 1968.
13. Steven Hotovy, "Workload Evolution on the Cornell Theory Center IBM SP2". In *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph (Eds.), Springer-Verlag, 1996, Lect. Notes Comput. Sci. vol. 1162, pp. 27-40.
14. Joefon Jann, Pratap Pattnaik, Hubertus Franke, Fang Wang, Joseph Skovira, and Joseph Riodan, "Modeling of Workload in MPPs". In *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph (Eds.), Springer-Verlag, 1997, Lect. Notes Comput. Sci. vol. 1291, pp. 95-116.
15. E.J. Koldinger, S.J. Eggers, and H.M. Levy, "On the validity of trace-driven simulation for multiprocessors". In *18th Ann. Intl. Symp. Computer Architecture Conf. Proc.*, pp. 244-253, May 1991.
16. E.D. Lazowska, "The use of percentiles in modeling CPU service time distributions". In *Computer Performance*, K.M. Chandy and M. Reiser (eds.), 53-66, North Holland, 1977.
17. 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.
18. G. Lipshitz, and A. Raveh, "Applications of the Co-plot method in the study of socioeconomic differences among cities: A basis for a differential development policy", *Urban Studies* **31**, pp. 123-135, 1994.
19. V. Lo, J. Mache, and K. Windisch, "A comparative study of real workload traces and synthetic workload models for parallel job scheduling". In *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph (Eds.), Springer Verlag, 1998. Lect. Notes Comput. Sci. vol. 1459, pp. 25-46.
20. Uri Lublin, "A Workload Model for Parallel Computer Systems", Master Thesis, Hebrew University of Jerusalem, 1999, in preparation.
21. S. Maital, "Multidimensional Scaling: Some Econometric Applications", *Journal of Econometrics* **8**, pp. 33-46, 1978.
22. S. Majumdar, D.L. Eager, and R.B. Bunt, "Scheduling in multiprogrammed parallel systems". In *Sigmetrics Conf. Measurement & Modeling of Computer Systems*, pp. 104-113, May 1988.
23. A. Raveh, "The Greek banking system: Reanalysis of performance", European Journal of Operational Research, (forthcoming).
24. K. Windisch, V. Lo, R. Moore, D. Feitelson, and B. Nitzberg, "A comparison of workload traces from two production parallel machines". In *6th Symp. Frontiers Massively Parallel Comput.*, pp.319-326, Oct 1996.

25. M.E. Crovella and A. Bestavros, "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes". In *Sigmetrics Conf. Measurement & Modeling of Computer Systems*, pp. 160-169, May 1996.
26. S.D. Gribble, G.S. Manku, D. Roselli, E.A. Brewer, T.J. Gibson and E.L. Miller, "Self-Similarity in File Systems". *Performace Evaluation Review* ***26(1)*** , pp. 141-150, 1998.
27. Jan Beran, *Statistics for Long-Memory Processes*. Monographs on Statistics and Applied Probability. Chapman and Hall, New York, NY, 1994.

Appendix: Theory of Self-Similarity

This section briefly describes the theory behind self-similarity, and three methods for finding it in a given time series. For a thorough introduction, see [27].

Consider a stochastic process $X = (X_1, X_2, \ldots)$ with mean $\mu = E[X_i]$, variance $\sigma^2 = E[(X_i - \mu)^2]$, and auto-correlation function:

$$r(k) = \frac{E[(X_i - \mu)(X_{i+k} - \mu)]}{E[(X_i - \mu)^2]} \qquad k \geq 0 \tag{5}$$

The process is said to exhibit long-range dependence if:

$$r(k) \approx k^{-\beta} L(t) \quad \text{for some } 0 < \beta < 2 \quad \text{as } k \to \infty \tag{6}$$

where L(t) is a slowly varying function with the property:

$$\lim_{t \to \infty} \frac{L(tx)}{L(t)} = 1 \qquad \forall x > 0 \tag{7}$$

Self-similar processes possess long-range dependence, but also satisfy stronger constraints on the form of the auto-regression function, to be defined now. For a time series $\{X\}$, A new aggregated series $X^{(m)} = (X_k^{(m)} : k = 1, 2, \ldots)$ for each $m = 1, 2, 3\ldots$ is obtained by averaging non-overlapping blocks of size m from the original series:

$$X_k^{(m)} = \frac{X_{km-m+1} + \ldots + X_{km}}{m} \qquad k \geq 1 \tag{8}$$

The process X is said to be exactly second-order self-similar if there exists $0 < \beta < 2$ such that the following two conditions hold:

$$\begin{aligned} &\mathrm{Var}(X^{(m)}) \propto m^{-\beta} && \text{for all } m = 1, 2, 3, \ldots \\ &r^{(m)}(k) = r(k) && \text{for all } k = 1, 2, 3, \ldots \end{aligned} \tag{9}$$

The process is said to be asymptotically second-order self-similar if the following weaker conditions holds instead (note that only the second condition changes):

$$\begin{aligned} &\mathrm{Var}(X^{(m)}) \propto m^{-\beta} && \text{for all } m = 1, 2, 3, \ldots \\ &r^{(m)}(k) \to r(k) && \text{as } m \to \infty \end{aligned} \tag{10}$$

For historic reasons, Self-similar processes are characterized by their Hurst parameter, defined:

$$H = 1 - \frac{\beta}{2} \qquad (11)$$

The rescaled adjusted range (R/S) statistic for a series X having average A(n) and sample variance $S^2(n)$ is given by:

$$R(n) / S(n) = [\ 1 - S(n)\ ] \times [\ \max(0, W_1, \ldots, W_k) - \min(0, W_1, \ldots, W_k)\ ] \qquad (12)$$

Where:

$$W_k = (X_1 + X_2 + \ldots + X_k) - kA(n) \qquad (k \geq 1) \qquad (13)$$

Short-range dependent observations seem to satisfy $E[\ R(n) / S(n)\ ] \approx c_0 n^{0.5}$, while long-range dependent data, such as self-similar processes, are observed to follow:

$$E\ [\ R(n) / S(n)\ ] \approx c_0 n^H \quad (0 < H < 1) \qquad (14)$$

This is known as The Hurst Effect, and can be used to differentiate self-similar from non-self-similar processes. In general, the Hurst parameter can be in one of three categories: $H < 0.5$, $H = 0.5$ and $H > 0.5$. When $H = 0.5$ a random walk is produced, and no long-range dependence is observed. When $H > 0.5$, the values produces are self-similar with positive correlation, or persistent; when $H < 0.5$ the values are self-similar with negative correlation, also called anti-persistent. Most observed self-similar data to date is persistent.

Equation 14 can be used to produce an estimate of the Hurst parameter of a given trace, observing that the following is derived from it:

$$\log [\ R(n) / S(n)\ ] = c_1 + H \log(n) \qquad (15)$$

Plotting R(n)/S(n) against log(n) for increasing values of *n* should therefore result in a roughly linear line, with a slope equal to the estimated Hurst parameter. Such a graph is called a Pox Plot, and the technique is called R/S analysis.

As you recall from their definition, self-similar processes must satisfy:

$$\mathrm{Var}(X^{(m)}) \propto m^{-\beta} \qquad \text{for all } m = 1, 2, 3, \ldots \qquad (16)$$

Taking the logarithm of both sides of the equation gives:

$$\log [\ \mathrm{Var}(X^{(m)})\ ] = c_2 - \beta \log(m) \qquad (17)$$

Again, plotting $\log(\mathrm{Var}(X^{(m)}))$ against log(m) for a self-similar process should result in a linear series of points with a slope of $-\beta$. The Hurst parameter estimate is $H = 1 - \beta/2$, therefore a slope between –2 and 0 indicates self-similarity ($0.5 < H < 1$). This plot is known as a Variance-Time Plot.

Finally, the Periodogram is a statistical method to discover cycles in time series. The periodogram of a time series $\{X\}_{1..N}$ for a given frequency $-\Pi \leq \omega \leq \Pi$ is defined as:

$$\mathrm{Per}(w) = \frac{2}{N} \times \left[\left(\sum_{k=1}^{N} X_i \cos(wk) \right)^2 + \left(\sum_{k=1}^{N} X_i \sin(wk) \right)^2 \right] \qquad (\mathbf{18})$$

The periodogram graph of a time series is computed by plotting a log-log graph of $\mathrm{Per}(\omega_i)$ against the following frequencies:

$$\omega_i = \frac{2\Pi i}{N} \qquad i = 0..N \qquad (\mathbf{19})$$

For a self-similar time series, the slope of the periodogram is a straight line with slope $\beta - 1 = 1 - 2H$ close to the origin.