Exploiting per user information for supercomputing workload prediction requires care

Tuan V. Dinh Lachlan L. H. Andrew Philip Branch
Swinburne University of Technology, Australia
{tdinh,landrew,pbranch}@swin.edu.au

***Abstract*—Efficient management of supercomputing facilities requires estimates of future workload based on past user behaviour. For supercomputers with large numbers of users, aggregate user behaviour is commonly assumed to be best in prediction of future workloads, however for systems with smaller numbers of users the question arises as to whether it is still suitable or if benefits can be derived from monitoring individual user behaviour to predict future workload. We compare using individual user behaviour, aggregate user behaviour and a hybrid approach where we track heavy users individually and cluster aggregate light users into a small number of clusters. We find that the hybrid approach produces the best results in both mean absolute error and mean squared error. However, treating all users separately provides slightly worse predictions. We also introduce a new approach to prediction based on the hazard function which is a significant improvement on previously used schemes based on autoregressive models. The schemes are investigated numerically using a two-year workload trace from a supercomputer with a population of 136 users.**

***Index Terms*—supercomputing, workload prediction, hazard rate function, user behaviours**

I. Introduction

Supercomputers are increasingly important for tasks ranging from weather forecasting and climate modelling to mapping the human genome and modelling the world economy. In some cases a supercomputer is built for a single task, but in most cases supercomputers are shared by many different users who submit jobs as the need arises.

Efficient management requires that the workload can be predicted at least a short time in advance. For example, some "elastic" jobs can dynamically increase the accuracy with which results are calculated if it is known that demand will be light until the job's deadline. Other jobs can be run with different degrees of parallelism, but cannot change the parallelism once they have started . Workload prediction is also important for managing "valley-filling", in which delay-insensitive jobs are run at times of low load, and in dynamic provisioning [1], [2], [3] in which some servers in a cluster are turned off during periods of low load to save energy. Most previous work on predicting workload has focused on systems with very large numbers of users, which averages out much of the burstiness of job arrivals. In contrast, we propose a new approach suitable to systems with a small user population. Although the reduced multiplexing makes this problem harder, it also allows us to exploit information about individual heavy users in the predictions. In particular, our predictor takes into account the length of time since each heavy user last submitted a job.

As a case study, we use a two-year workload trace taken from the Swinburne University's supercomputing cluster. This computer serves a small number of heavy users and a much larger number of light users. In order to obtain statistically meaningful estimates for the behaviour of light users, we cluster them into a small number of classes and base our predictions on the aggregate per-cluster information. This also reduces the computational requirements of the prediction process.

A. Related work

There are two bodies of work that are closely related to this paper — one characterizing supercomputing job arrivals and one predicting future CPU utilization — which we will discuss in turn.

An important application for characterizing supercomputing workloads is for generating the synthetic workload [4], [5]. This requires both a description of the jobs themselves and the arrival process in time. The former commonly [6], [7], [8] consists of statistical models of attributes such as the number of CPUs used by a job and job run times, and more subtle attributes such as cancellation rates [9].

The arrival process is usually characterized by the inter-arrivals times, which are either explicitly [6] or implicitly assumed to be independent. In early work, the arrival process was assumed to be Poisson [6]. Later, Lublin et at. [5] proposed modelling inter-arrival times as Gamma distributed. Heavy-tail distributions such as Log-normal or Pareto were suggested in [9].

Squillante et al. [10] is one of the few groups to study the dependence between inter-arrival times. They noted that arrivals tend to occur in *batches*. We also observed arrivals in batches at the Swinburne supercomputer.

This branch of work is primarily descriptive, and does not immediately lead to techniques for workload estimation. In contrast, the second branch considers predictive models, but focus on the accumulated load (total arriving work minus work served, or CPU load) rather than specifically the arrivals.

The second branch typically uses models based on (pth-order) autoregressive AR(p) processes. This model has found significant success in predicting the workload for CPU load [11] or in Grid computing [12]. Such models assume that workload process is stationary. That is, characteristics

such as the mean and autocorrelation of the workload are time invariant. This allows the AR model to be built using a historical workload. Several studies improve the prediction accuracy of such AR models by either adapting the model based on recent observations [13], [14], or employing "data refinement" [12], such as a Kalman filter or a Savitzky-Golay filter, to reduce the noise so that the AR model can capture the underlying workload more accurately.

AR models have typically been used to predict use of a single resource, such as CPU capacity. Liang et al. [13] proposed a cross-correlation model to predict the use of multiple kinds of resources such as CPU and memory.

Despite these two substantial bodies of work, little attention has been paid to the task of predicting the actual work arrival process. This paper seeks to address that lack.

B. Contributions

This paper makes the following contributions: In Section II, we characterise the workload of the Swinburne Supercomputer based on two year traces. Unlike most of the past work, we focus on the impact of user behaviours on the workload characteristics. We then present the user clustering results from experiments using well-recognised data mining software, WEKA [15].

In Section III, we introduce a load estimation method that employs the hazard function of the inter arrival time distribution and we explain how per user information can be incorporated. The procedure of building such model from the historical traces will be described. We then address the challenges of smoothing the hazard curves and how they were met.

In Section V, we evaluate our model by comparing it with autoregressive based models and show significant improvement with respect to the mean squared errors of the predictions. We also demonstrate that even with the comparatively small number of users on this system, aggregate behaviour is a better predictor than individual behaviour.

II. Properties of the workload traces

In order to predict job arrivals, it is important to have a thorough understanding of the workload. We now investigate the characteristics of the workload of Swinburne's supercomputing cluster. The focus will be on identifying heterogeneity between users, which we will attempt to exploit in the predictor of the following section.

A. Overview of the Swinburne Supercomputer

The Swinburne supercomputer comprises around 160 computing nodes, each with eight CPU cores except for a "head" node, which has 4 CPU cores, 16 GBytes of RAM and 2× 500GBytes of local disk storage. The cluster is controlled by that *head* node, which uses Moab [16] and Torque [17] for scheduling. Specifically, Torque keeps track of the system state (available CPUs, load on each computational node, etc). Moab regularly queries Torque to find the system state and then schedules jobs according to the configuration set up by the administrator. Torque eventually dispatches jobs to the computational nodes based on Moab's decisions.

The cluster is primarily used by astrophysicists for performing large scale physical simulations and for processing large observational data sets. This workload consists of some highly paralellizable jobs, such as inverting gravitational lensing, and some jobs that are either hard to parallelize or for which no parallel code exists. In addition, the cluster is available for other users at Swinburne, and other significant users run simulations for molecular dynamics and micro photonics research. The cluster is also used for teaching related purposes at the university.

Users are given accounts that allow them to submit jobs to the supercomputer via the resource manager (Torque) running on the head node. When submitting a job, the user must specify the resources that will be required. This includes, among other things, the required (a) total number of CPUs, (b) memory, (c) disk space, and (d) estimated run time (wall time). When a job is scheduled, it will be allocated exclusive use of the requested number of CPUs, memory and disk space until the job completes.

Users tend to submit jobs in batches, and so we seek to predict the batch arrival process instead of the individual job arrival process. We define a batch as a sequence of jobs such that (a) the inter arrival time between any consecutive jobs is less than $\delta = 10$ seconds, and (b) they come from the same user.

A common motivation for estimating job arrivals is to estimate the total amount of work likely to be in the system at a future time. For that, it appears that having users supply an estimate of the run time will be extremely useful. However, this turns out not to be the case. Whenever the job run time exceeds the requested walltime, the execution is aborted and this event is reported to the owner. Since there is minimal incentive for users to provide short estimates of run times, this results in users overestimating run times by an order of magnitude. This effect has been observed in other shared computing systems [8], [18].

B. Aggregate and individual user behaviour

A central contribution of this paper is an investigation into whether aggregate workload is still a suitable predictor for supercomputers with few users or whether tracking individual user behaviour gives better workload predictions. To this end, we first investigate how the load is distributed among users.

Workload data for this paper was collected over a two-year period, from January 2010 to December 2011. Over that period, 136 distinct users submitted jobs. Among them, 86 users submitted jobs in 2010, and 104 in 2011, of whom 54 users submitted jobs in both years. As is often the case, a minority of the users generated the majority of the workload, measured as the sum over jobs of the product of job run time and number of CPUs used. This is illustrated in Figure 1. Specifically, around 15% of users generated 85% of the workload, which is a slightly greater concentration of work than Pareto's famous "80:20" rule. However, as shown in

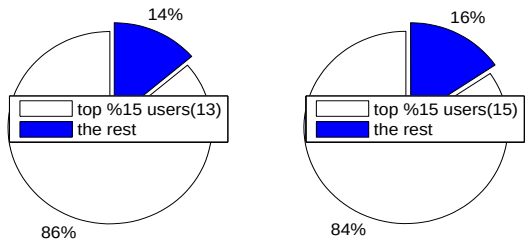

Fig. 1. Workload contribution by users in 2010(left) and 2011. Most workload was generated by a few users as 15% of users generated 85% of the total workload.

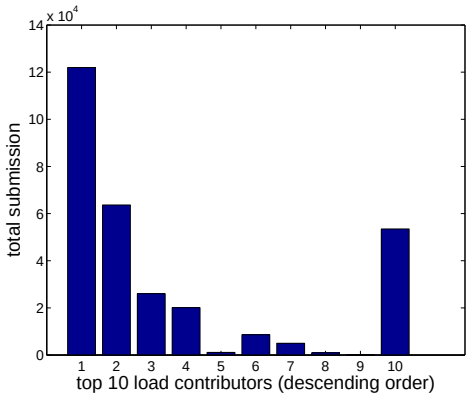

Fig. 2. The total number of submissions during 2011 of the 10 largest users in terms of total CPU hours. Note that the big users do not necessarily submit many jobs.

Figure 2, the "big" users need not be those who submit the largest number of jobs.

The fact that a small number of users dominates the workload suggests that it may be useful to track those users in detail, as will be done in the following section.

III. Estimating arriving load

We now turn to estimating the job arrival process. Predicting the time of arrival of jobs on a continuous timescale is prohibitively difficult, and so we focus on estimating the probability that there will be a job arrival in a given discrete time slot.

We first assume a simple information structure, in which the estimator has knowledge only of the most recent job arrival time, and the marginal distribution of inter-arrival times. Due to this lack of information, it is natural to assume that inter-arrival times are independent and identically distributed (i.i.d.); this assumption will be relaxed in the numerical evaluations in Section V.

Assume without loss of generality that at time t, the most recent job arrival was at time 0. Let T be the random time of the next arrival, with probability mass function (PMF) $p(\cdot)$. At t, all that is known is that $T \geq t$. Thus, the probability of a job arriving at t (i.e., $T = t$) is

$$\mathbb{P}\{T = t | T \geq t\} = \frac{p(t)}{\mathbb{P}\{T \geq t\}} \equiv h(t). \quad (1)$$

This is called the *hazard rate* (HR) of the distribution p [19]. Without further information, (1) is the best possible estimate of the job arrival probability.

Next, consider a system with N users, which records statistics for each user, j. In particular, let $p_j(\cdot)$ be the p.m.f. of the distribution of inter-arrival times between batches of user j, and $h_j(\cdot)$ the hazard rate of p_j. Let μ_j be the mean number of required CPUs per request of user j, and $s_j(t)$ be the last time before t that j submitted a job. We propose to estimate the number of CPUs requested at time t by

$$\hat{x}_t = \sum_{j=1}^{N} \mu_j h_j(t - s_j(t)). \quad (2)$$

Here $h_j(t - s_j(t))$ is the probability of having an arrival from user j at time t.

A. Potential for enhancement

The estimator (2) is the exact mean number of CPUs requested if inter-arrival times are i.i.d.. However, there is typically a rich structure in the inter-arrival times. Numerical results in Section V suggest that (2) is a substantial improvement on alternatives, but it can be further improved by considering non-i.i.d. models.

Inter-arrival times usually have a strong diurnal pattern, with fewer jobs submitted at night, and a weekly pattern, with fewer jobs submitted on weekends. These violate the assumption that inter-arrival times are identically distributed. However, if the inter-arrival times are short compared with the time scale on which these variations occur, then these variations can be incorporated into (2) by replacing $p(\cdot)$ by a time-varying PMF.

Similarly, a user will have different modes of working. The main users of Swinburne's supercomputer are astrophysicists. They will often perform a spate of calculations when a new observational data set arrives, followed by longer periods of inactivity. Again, (2) can be applied to models in which a hidden Markov model tracks the state of individual users, and $p(\cdot)$ is a state-dependent PMF.

Each of these enhancements adds complexity to the model, making it harder to estimate the required PMFs, and hence hazard rate functions. We now consider how this estimation is done.

B. Estimating hazard function from the arrival records

The accuracy of (2) depends on having an accurate estimate of the inter-arrival time distribution, and hence its hazard rate. This is obtained from historical data. However, there is a limited amount of data, and so empirical PMF is subject to substantial sampling error. To alleviate this, we fit a simple parametric model to the observed PMF. Rather than using a maximum likelihood estimate, we perform the fitting in two steps, as follows.

1) Empirical hazard rate: The empirical hazard rate for individual user is estimated as follows. Let $T_{\max}$ be the maximum interarrival time, and divide the interval $[0, T_{\max}]$ into K bins, such that the width of the ith bin is w_i. From the past workload, calculate the inter-arrival times of all batches

from user j and group them into the bins. Let n_i be the number of samples in the ith bin. The hazard rate at time t can then be estimated as:

$$\hat{h}_j(t) = \frac{n_i}{w_i \sum_{k=i}^{K-1} n_k}. \quad (3)$$

If the number of samples is large and all w_i are small, then this is a good approximation of the true hazard rate [20].

2) Smoothing by parametric fitting: When the amount of historical data is small, the above fitting may not accurately reflect the true underlying inter-arrival time distribution. It is well known that fitting an appropriate model with few parameters can improve the generalization ability of a statistical model. To this end, we fit two-and three-parameter curves to the empirical hazard rate. As well as improving the generalization ability, this allows the hazard rate to be calculated for non-integer times t, and reduces the storage required to model a large user population. The drawback is that the user's hazard rate may not actually be well approximated by the chosen parametric family.

We considered two families of curve: the Weibull hazard function and a custom function inspired by the Gamma hazard function:

$$h_W(t) = \frac{k}{\alpha}\left(\frac{t}{\alpha}\right)^{k-1} \quad (4)$$

and

$$h_C(t) = at^b e^{-ct}. \quad (5)$$

The simple form of the hazard rate of the Weibull distribution makes it a common choice, and it is often found to be sufficiently accurate as in the field of reliability engineering [21]. However, hazard rates we observed typically increase to a mode and then decrease, a characteristic which cannot be captured by a Weibull distribution. That weakness motivated us to develop the second form, which is reminiscent of a Gamma function.

For each family, we obtained a least-squares fit. To ensure that each measured sample is given equal weight, the probability density was estimated by binning the data into bins that had an approximately equal number of samples — two samples in our case — subject to the requirement that bin boundaries be multiples of the discrete time step.

The two fitted curves are shown in Figure 3, along with the empirical data. Both families show comparable fitting quality for user 1,3 and 4. However, for user 2 and 5, the Weibull curves show some limitations. As for user 6, both fitted curves appear undesirable as none candidate can fit well with both head and tail of the fitting curve. But in overall (considering all six users), the "Gamma" fitting appears to be the better choice. Note that these top six users contributed roughly 70% of total workload in 2010.

IV. Clustering small users

When a users submits very few jobs, it becomes impossible to estimate the hazard rate with useful statistical significance. Instead, we cluster small users together and treat each cluster as a single user. Since the purpose of the prediction is to improve the efficiency of resource management and the key resource is the number of CPUs, we clustered based on the mean and standard deviation of the mean and mean-square inter-arrival times.

TABLE I
User clusters with WEKA using k-means algorithm

Normalised data	Clus. 1 30%(21)	Clus. 2 11%(8)	Clus. 3 59%(42)	Full data 100%(71)
Mean inter-arrival	0.043	0.365	0.009	0.059
Std. dev. inter-arrival	0.146	0.609	0.020	0.124
Mean sq. inter-arrival	0.018	0.437	0.001	0.055
Std. dev. sq. inter-arrival	0.056	0.539	0.003	0.079

Each of the top 15 users is allocated to its own cluster. To cluster the remaining 71 users, we used the Weka Data Mining Software [15] with iterative k-means algorithm [22]. In this algorithm, the number of the clusters k is provided in advance, then k random points are chosen as the initial centroids of the clusters. Users are added to the cluster to which they have the smallest Euclidean distance among the k clusters. At the end of each iteration, the new "mean" or centroid is calculated for each cluster. The whole process is repeated until the clusters are stable, meaning that each point is assigned to the same cluster as the last iteration.

Table I shows the clustering results, completed after 5 iterations. The clusters are of comparable sizes, although Cluster 3 is substantially larger than the other two. The clustering is on four feature dimensions, and so it is not easy to visualise the clusters directly.

Although these clusters are not clearly delineated, the numerical results in Section V will show that this clustering actually results in improved load estimation. That is because the statistical behaviour of individual small users cannot be determined accurately from their own histories, but the behaviour of the clusters can be estimated more accurately.

With this clustering, the workload estimate becomes

$$\hat{x}_t = \sum_{j=1}^{\hat{N}} \mu_j h_j(t - s_j(t)) + \sum_{c=1}^{C} \mu_c h_c(t - s_c(t)) \quad (6)$$

where $\hat{N}$ is the number of users who contributed the majority of the workload ($N = 15$ in our case) and C is number of clusters of less significant users ($C = 3$ in our case).

V. Numerical evaluation

A. Objectives

The derivation of (2) assumed that inter-arrival times were i.i.d., which we know not to be the case. In this section, we will evaluate the effectiveness of the resulting estimator on a real workload to investigate the impact of that assumption. Let time be divided into M timeslots, and let x_t denote the number of CPUs requested in the tth timeslot. The objective is to be able to forecast x_{t+1} at time t, based only on knowledge available at time t. Such forecasts play an important role in efficient allocation of resources [1], [2], [3].

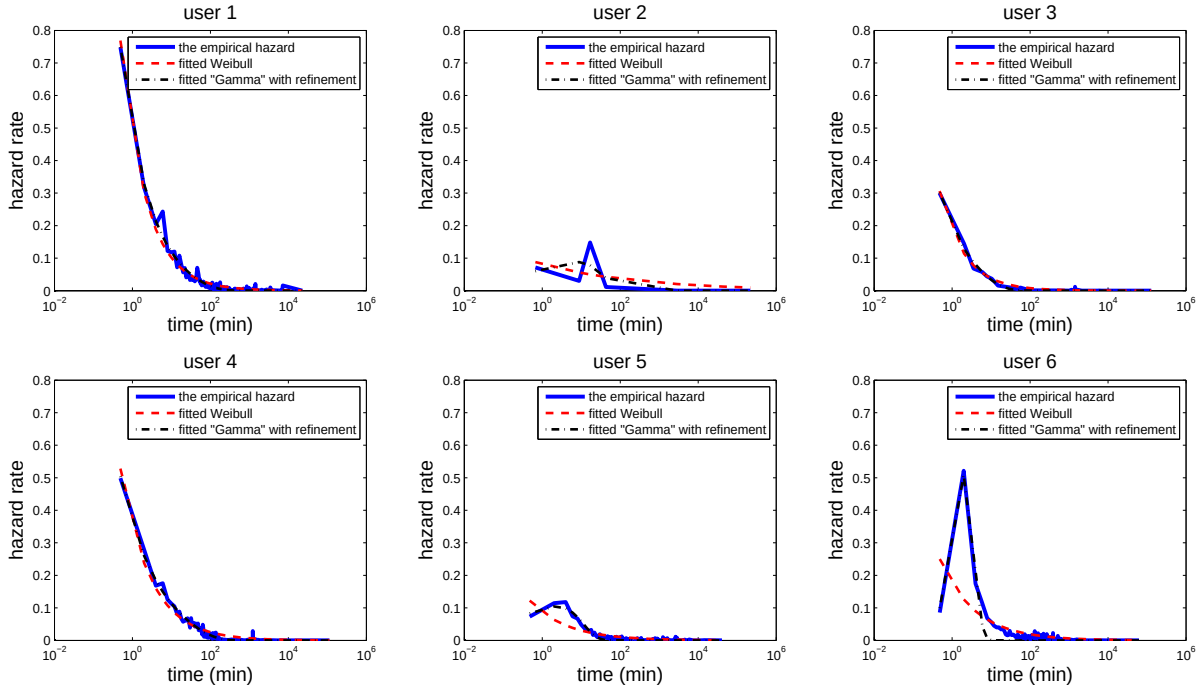

Fig. 3. Curve fittings for six top users

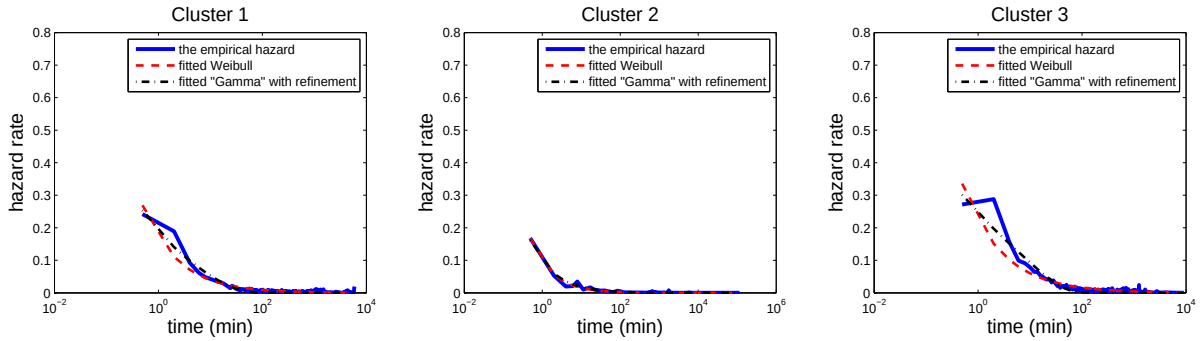

Fig. 4. Curve fittings for 3 clusters

The experiment used a two year trace of work from the Swinburne supercomputer. To be consistent through our experiment, for all models (including our hazard function based model), we will use the first year traces as the sample workload to build the models and the second year traces are used for evaluation.

B. Benchmark models

In order to calibrate the effectiveness of our proposed scheme, we would like to compare it against an existing estimator. However, there has been little prior work on estimating *arrivals* to supercomputing clusters (although see [23], [24]). Instead we will use a technique that has been developed for estimating a different notion of "workload", namely the number of ongoing jobs. Note that our task is substantially harder than estimating the number of ongoing jobs; the arrival process is related to the *derivative* of the number of ongoing jobs, and it is well known that differentiating a noisy quantity leads to high errors.

As a benchmark, we will compare the performance of (2) against two variants of autoregressive (AR) models, which are generalizations of the familiar exponentially weighted moving average (EWMA) filter. These models have found substantial success in predicting ongoing workload [11], [13], [12]. We shall compare our model with the classical AR(p) model and an enhancement version with adaptive mean that is described in [13].

1) AR(p): Autoregressive filters are the simplest in a family of filters that include, in increasing order of complexity, ARMA (autoregressive moving average), ARIMA (autoregressive integrated moving average) and ARFIMA (autoregressive fractional integrated moving average). Their structure allows them to incorporate substantial memory with only a small number of parameters. Having fewer parameters allows those

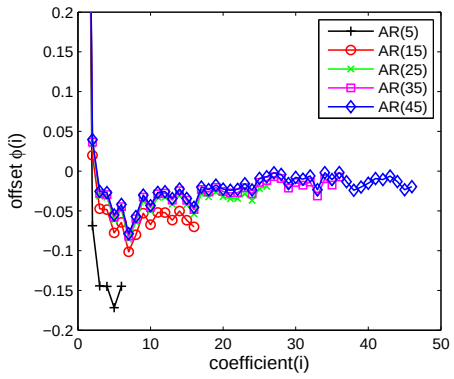

Fig. 5. AR coefficients for different values of p, calculated using Yule-Walker method and the workload traces of Swinburne Supercomputer in 2010 as the sample workload.

parameters to be estimated more accurately, and they were found in [11] to be superior to ARMA and its generalizations for the task of predicting CPU workload.

The pth-order AR model, AR(p), predicts the next request for CPU resources, $\hat{x}_k$, by the recurrence

$$\hat{x}_k = \sum_{i=1}^{p} \phi_i x_{k-i} \tag{7}$$

where $x_{k-1}, x_{k-2}, ..., x_{k-p}$ are the previous measurements and $\{\phi_i\}$ are the AR coefficients.

The AR coefficients $\{\phi_i\}$ can be determined from a historical sample of the workload. To do this, we assume that the true arrival counts satisfy

$$x_k = \sum_{i=1}^{p} \phi_i x_{k-i} + \epsilon_k \tag{8}$$

where ϵ_k is a residual error signal. If $\{\epsilon_k\}$ are zero mean white Gaussian noise, then the AR coefficients can be determined using the Yule-Walker method [25], which finds the $\{\phi_i\}$ in terms of the autocorrelation functions of $\{x_k\}$.

Figure 5 presents the AR coefficient sets $\{\phi_i\}$ for several values of p for our sample workload. It shows a strong positive correlation with the last immediate observation in the process, and a weaker negative correlation with recent observations which then decays into noise. The figure shows that estimators with $p \geq 15$ are all quite similar, and in particular that $p = 35$ is sufficient to capture all of the autoregressive structure in this data set. For that reason, we will use AR(35) as our benchmark.

2) AR(p) with adaptive mean: The AR(p) model assumes that the workload process is stationary. However, workload data typically has a strong diurnal variation. To account for this, we adopt the technique used in [13], which separates out a slowly-varying mean from a faster-varying AR process.

The AR coefficients are obtained the same way as for the simple AR(p) model. In addition, there is a parameter α, such that $1 - \alpha << 1$, which governs the drift in the overall mean denoted by $\{\mu_k\}$. The steps to estimate $\hat{x}_k$ are as follows.

First, the expected value of x_k is estimated as

$$\mu_k = \alpha\mu_{k-1} + (1 - \alpha)x_{k-1}.$$

Next, the perturbation from the mean is found. Given observations x_{k-p} to x_{k-1}, a zero-mean sequence

$$x^k_{k-i} = x_{k-i} - \frac{1}{p}\sum_{j=1}^{p} x_{k-p}$$

is constructed, and then passed through the AR filter too obtain

$$x'_k = \sum_{i=1}^{p} \phi_i x^k_{k-i}.$$

Finally, the load at time k is estimated by

$$\hat{x}_k = \mu_k + x'_k. \tag{9}$$

C. Estimating rates rather than arrivals

A consequence of having a small user population is that arrivals are very bursty. The majority of time slots will have no arrivals, whereas a few will have arrivals requesting hundreds of CPUs. This can be seen in Figure 6, which shows the data for an 5 hour interval of the Swinburne supercomputer workload. As a result, it is not feasible to estimate the exact amount of work that will arrive in a given time slot. Instead, we estimate the *expected rate* of arrivals. Many resource allocation algorithms take as input a time-varying Poisson process, and so estimating the time-varying rate of that process is of practical benefit.

We can think of the observed number of arrivals x_t in each slot t as a realization of a random process. We wish to compare our estimated number of arrivals $\hat{x}_t$ against the expected number of arrivals under that random process, rather than against the actual number of arrivals. To do this, we follow the "refinement" approach of [12], and compare against a moving average of the arrivals. Specifically, select an averaging window W and construct

$$y_k = \frac{1}{2K+1}\sum_{i=-W}^{W} x_{i+k}. \tag{10}$$

Note that this cannot be constructed online, but can be calculated after all the data is known, and used for evaluation. In our experiments, the time step was $\Delta = 5$ minutes, and $W = 6$ was selected so that y_t represents the average over a one-hour period.

The primary figure of merit we will use is then the mean squared error. The mean squared error of the estimate $\{\hat{x}_t\}$ with respect to the smoothed data is defined as

$$\frac{1}{M}\sum_{t=1}^{M} |y_t - \hat{x}_t|^2, \tag{11}$$

where M is the number of timeslots used for evaluation. The two years of data was divided into one year for training the predictor, and one year for evaluation. Thus the horizon was one year, giving $M = 365 \times 24 \times 60/5$.

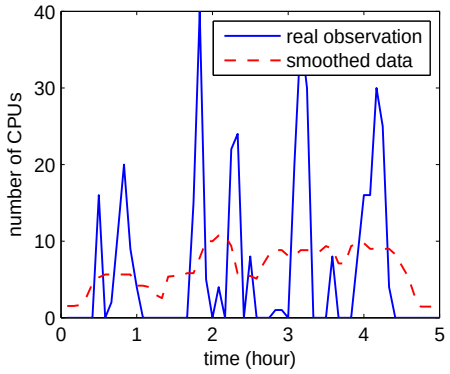

Fig. 6. Arrival workload of Swinburne Supercomputer for a five-hour interval. The arrivals appear to be very bursty but the overall characteristics are typical for such supercomputer systems ($\Delta = 5$ minutes, $W = 6$)

TABLE II
MEAN SQUARED ERROR AND MEAN ABSOLUTE ERROR FOR DIFFERENT PREDICTION SCHEMES.

Schemes	no smoothing		smoothing	
	MSE	MAE	MSE	MAE
AR(p=35)	4,300	11.67	2,310	9.29
AR(p=35)-adapt. mean	4,325	11.81	2,290	9.86
HR, per user	2,749	10.68	381	7.46
HR, clustered	2,733	8.44	371	5.86
HR, aggregate	2,738	8.60	377	6.10

D. Performance evaluation

With these preliminaries completed, we can now compare the performance of our hazard rate estimator (2) against the two benchmarks: AR(35) and AR(35) with adaptive mean. The latter used $\alpha = 0.99$, which corresponds to a time-constant for μ_k of 500 minutes (6 hours); this allows it to track diurnal variation but smooths out fast fluctuations.

The data from 2010 and 2011 was divided into slot of $\Delta = 5$ minutes. The data from 2010 was used to train both the benchmark estimators and our HR estimator, and the 2011 data was used for testing. At each time slot, the three estimators were given the most up-to-date information (recent arrival rates for AR, and the time of the last submission for HR), but their parameters ϕ and $h(\cdot)$ were not updated.

Those loads predicted by the three schemes were then compared with the refined version of the real load obtained from the traces using (11).

Table II summarizes the results. In this table, "HR, per-user" refers to the scheme (2) in which each user is its own cluster, "HR, clustered" refers to (6) in which users other than the top 15 were partitioned into three clusters, and "HR, aggregate" refers to (6) in which all users are placed in a single cluster.

The three HR-based schemes have a significantly lower MSE than the AR-based estimators: in the worst case it is 83% lower with data smoothing and 36% lower without data smoothing. Since those large MSEs may raise the concern about the potential impact of the schemes, it is important to emphasize that such large MSEs are a result of several large prediction errors due to the highly bursty nature of the workload. The mean absolute error (MAE) is less affected by the very large bursts, and is consequently more optimistic. Among the HR-based schemes, clustering small users together and separating out large users (HR, clusterd) appears to provide the minimum in both MSE and MAE, while grouping all users into one cluster is the second best (HR, aggregate). The "HR, per user" scheme, where all users are modelled separately, provides the worst performance; this is presumably due to the difficulty of estimating the hazard rate of light users accurately.

As expected, the mean squared error and mean absolute error are substantially improved after the smoothing process for all cases. Note that these quantitative conclusions appear sensitive to the workload; for example, changes in the sampling interval Δ can even change the order of HR per-user and HR-aggregate. However, since HR-clustered performed best, even without optimizing the clustering parameters, we expect that if the clustering is optimized then the conclusion that a hybrid HR-clustered approach will continue to be the best for most workloads.

VI. CONCLUDING REMARKS

Accurate workload prediction is useful for efficient management of supercomputer resources. In this paper, we have described an approach for exploiting per user information for workload estimation. The approach takes as input the most recent submission time from each user, and produces an estimate of the expected arriving workload in the next time step. Since predicting individual job arrivals is intractable, this is evaluated by comparison with the average arrival rate over short time periods. Testing on the workload of the Swinburne Supercomputer for a two year period showed that the proposed scheme yields lower mean squared error than two autoregressive schemes of order 35: AR(35) and AR(35) with adaptive mean.

We also investigated per-user prediction and k-means clustering of users who contributed little to the load, and were consequently hard to predict. In our example, clustering small users together resulted in better median prediction accuracy than either a scheme that aggregates all users or a scheme that attempts to estimate the future arrivals of all users individually.

Neither the proposed scheme nor the benchmark consider predictable diurnal fluctuations in load. The approach taken here is in principle easy to extend to incorporate such fluctuations, but doing so would require an estimate of the non-i.i.d. inter-arrival time distributions. This is an important direction for future study. Even without this extension, the proposed scheme demonstrates the usefulness of per-user information in workload prediction.

ACKNOWLEDGEMENT

The authors thank Dr. Yoni Nazarathy for extensive and fruitful discussions and anonymous reviewers for their insightful comments and suggestions to help improve our paper. This work was funded by ARC grant FT0991594.

References

[1] D. Kusic, J. Kephart, J. Hanson, N. Kandasamy, and G. Jiang, "Power and performance management of virtualized computing environments via lookahead control," *Cluster Computing*, vol. 12, pp. 1–15, 2009.

[2] Minghong Lin and Adam Wierman and Lachlan L. H. Andrew and Eno Thereska, "Dynamic right-sizing for power-proportional data centers," in *Proc. IEEE INFOCOM*, Shanghai, China, Apr. 2011, pp. 1098–1106.

[3] M. Lin, Z. Liu, A. Wierman, and L. L. H. Andrew, "Online algorithms for geographical load balancing," in *International Green Computing Conference(IGCC)*, 2012.

[4] 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*, ser. Lecture Notes in Computer Science, 1998, vol. 1459, pp. 25–46.

[5] U. Lublin and D. G. Feitelson, "The workload on parallel supercomputers: modeling the characteristics of rigid jobs," *Journal of Parallel and Distributed Computing*, vol. 63, no. 11, pp. 1105 – 1122, 2003.

[6] D. Feitelson, "Packing schemes for gang scheduling," in *Job Scheduling Strategies for Parallel Processing*, ser. Lecture Notes in Computer Science, 1996, vol. 1162, pp. 89–110.

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

[8] A. Mu'alem and D. Feitelson, "Utilization, predictability, workloads, and user runtime estimates in scheduling the ibm sp2 with backfilling," *IEEE Transactions on Parallel and Distributed Systems*, vol. 12, no. 6, pp. 529 –543, Jun. 2001.

[9] H. Li, D. Groep, and L. Wolters, "Workload characteristics of a multi-cluster supercomputer," in *Job Scheduling Strategies for Parallel Processing*, ser. Lecture Notes in Computer Science, 2005, vol. 3277, pp. 33–53.

[10] M. S. Squillante, D. D. Yao, and L. Zhang, "Analysis of job arrival patterns and parallel scheduling performance," *Performance Evaluation*, vol. 36 - 37, pp. 137 – 163, 1999.

[11] P. A. Dinda and D. R. O'Hallaron, "Host load prediction using linear models," *Cluster Computing*, vol. 3, pp. 265–280, 2000.

[12] Y. Wu, K. Hwang, Y. Yuan, and W. Zheng, "Adaptive workload prediction of grid performance in confidence windows," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 7, pp. 925–938, Jul. 2010.

[13] J. Liang, K. Nahrstedt, and Y. Zhou, "Adaptive multi-resource prediction in distributed resource sharing environment," in *IEEE International Symposium on IEEE International Symposium on CCGrid 2004.*, Apr. 2004, pp. 293–300.

[14] L. Yang, I. Foster, and J. Schopf, "Homeostatic and tendency-based cpu load predictions," in *Proceedings International Parallel and Distributed Processing Symposium.*, Apr. 2003, p. 9 pp.

[15] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, Nov. 2009.

[16] "Moab cluster scheduler." [Online]. Available: http://www.clusterresources.com/products/moab-cluster-suite/workload-manager.php

[17] "TORQUE Resource Manager." [Online]. Available: http://www.clusterresources.com/products/torque-resource-manager.php

[18] D. Tsafrir, Y. Etsion, and D. Feitelson, "Modeling user runtime estimates," in *Job Scheduling Strategies for Parallel Processing*, ser. Lecture Notes in Computer Science, 2005, vol. 3834, pp. 1–35.

[19] M. Shaked, J. G. Shanthikumar, and J. B. Valdez-Torres, "Discrete hazard rate functions," *Computers & Operations Research*, vol. 22, no. 4, pp. 391 – 402, 1995.

[20] J.-L. Wang, *Smoothing Hazard Rates*. John Wiley & Sons, Ltd, 2005.

[21] E. A. Elsayed, *Reliability Engineering*, 2nd ed. John Wiley & Sons, 2012.

[22] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005.

[23] C. Leangsuksun, L. Shen, T. Liu, H. Song, and S. L. Scott, "Availability prediction and modeling of high availability oscar cluster," *IEEE International Conference on Cluster Computing*, vol. 0, p. 380, 2003.

[24] S. A. Jarvis, D. P. Spooner, H. N. L. C. Keung, J. Cao, S. Saini, and G. R. Nudd, "Performance prediction and its use in parallel and distributed computing systems," *Future Gener. Comput. Syst.*, vol. 22, no. 7, pp. 745–754, Aug. 2006.

[25] P. Stoica and R. Moses, *Introduction to spectral analysis*. Prentice Hall Upper Saddle River, NJ, 1997, vol. 89.