2004 IEEE International Symposium on Cluster Computing and the Grid

Adaptive Multi-Resource Prediction in Distributed Resource Sharing Environment

Jin Liang, Klara Nahrstedt and Yuanyuan Zhou
Department of Computer Science
University of Illinois at Urbana-Champaign
{jinliang, klara, yyzhou}@cs.uiuc.edu

Abstract

Resource prediction can greatly assist resource selection and scheduling in a distributed resource sharing environment such as a computational grid. Existing resource prediction models are either based on the autocorrelation of a single resource or based on the cross correlation between two resources. In this paper, we propose a multi-resource prediction model *(MModel) that uses both kinds of correlations to achieve higher prediction accuracy. We also present two adaptation techniques that enable the MModel to adapt to the time-varying characteristics of the underlying resources. Experimental results with CPU load prediction in both workstation and grid environment show that on average, the adaptive MModel (called MModel-a) can achieve from 6% to more than 90% reduction in prediction errors compared with the autoregressive (AR) model, which has previously been shown to work well for CPU load predictions.*

1. Introduction

Grid computing [9] has made it possible for a user to access large number of geographically distributed resources. However, these resources are often time-shared with other users, and the amount of resources available to the user may vary greatly over time. This means before an application is launched to run on the grid, the user (or some application scheduler on behalf of the user) must carefully select a subset of the resources and schedule the application to run on these resources in order to maximize the application performance [3]. While such resource selection and scheduling must consider many factors such as application requirements and user preferences, its performance has been shown to depend largely on accurate resource predictions [3, 5, 17].

Although different models have been proposed for dynamic resource prediction [10, 7, 12, 11, 16], these models are either history based or based on a related resource. This means they either only make use of the autocorrelation of a single resource, or only utilize the cross correlation between two resources. In addition, many of the models are non-adaptive and cannot capture the time-varying characteristics of the underlying resources, which means they will have large prediction errors for highly dynamic resources.

In this paper, we propose a *multi-resource prediction model* (MModel) that uses both the autocorrelation (of a single resource) and the cross correlation (between two resources such as CPU and memory) to achieve higher prediction accuracy. We also present two adaptation techniques, namely the *mean adaptation* and the *cross correlation adaptation* that enable the MModel to adapt to the changing characteristics of the resources. We have evaluated our adaptive MModel (called MModel-a) by predicting the CPU load traces that we collected from workstation and the NCSA Alliance Grid [1] environment. Our results show that on average, MModel-a can achieve from 6% to more than 90% reduction in prediction errors compared with the autoregressive (AR) model. Our model also compares favorably to a tendency-based model [16]. Both the AR model and the tendency-based model have previously been shown to work well for CPU load predictions.

2. Background and related work

Existing statistical prediction models can be classified into two classes. The first class is history based and exploits the autocorrelation in a single resource. The second class is based on a related resource and exploits the cross correlation between the two resources.

Two simple prediction models based on the autocorrelation of a resource are the *LAST* model and the *exponential weighted moving average* (EWMA) model.

The LAST model predicts the next resource value as the previous measurement and can be expressed as

$$\hat{x}_k = x_{k\ \ 1}. \qquad (1)$$

The EWMA model predicts the next resource value as the weighted average of the previous prediction and the previous measurement. It can be expressed as

$$\hat{x}_k = \alpha \cdot \hat{x}_{k\ \ 1} + (1 - \alpha) \cdot x_{k\ \ 1}. \qquad (2)$$

Both the LAST model and the EWMA model are popular because they are simple. However, their simpleness also limits their prediction accuracy. Therefore, these models are often insufficient for resource predictions in a highly dynamic shared environment.

In [7], Dinda et al. have conducted extensive study of host load prediction. They use randomized test and evaluate the predictive power of a set of history based models. Their results show that the augoregressive (AR) model is the best among the models they studied, because of its low overhead and relatively high prediction accuracy. The AR model can be expressed as follows,

$$\hat{x}_k = \sum_{i=1}^{p} \phi_i x_{k\ \ i}. \qquad (3)$$

This means the resource value x_k at time k is predicted from the history values $x_{k\ \ 1}, x_{k\ \ 2}, \ldots, x_{k\ \ p}$. Here p is the order of the AR model. Like the LAST and the EWMA models, the AR model is based on the autocorrelation of a resource. However, AR does not prescribe the particular relationship between x_k and the history values. Instead, a sample data set is used to "fit" the model, i.e., to calculate the coefficients ϕ_i's.

In addition to being self-correlated, resources are often correlated with one another, and this correlation can also be used for prediction purpose. For example, Vazkudai et al. [12, 13] have used linear regression to predict the data transfer time from network bandwidth or disk throughput. The linear regression model between two resources is as follows.

$$\hat{y} = a + b \cdot x. \qquad (4)$$

This means the resource value y is predicted from the measured resource value x. The constants a and b characterize the correlation between x and y, and can be determined from history values of x and y.

In [11], a different technique is used to utilize the correlation between resources. Specifically, the cumulative distribution functions (CDFs) of two resources are computed from history data and the prediction of one resource is derived from its CDF, based on the measurement of the other resource.

Most of the existing prediction models are non-adaptive, this means the model is fixed once the prediction begins. As a result, if the resources have time-varying characteristics, these models will be unable to produce accurate predictions.

The Network Weather Service (NWS) [15, 14] uses a combination of several models for the prediction of one resource. NWS allows some adaptation by dynamically choosing the model that has performed the best recently for the next prediction, but its adaptation is limited to the selection of a model from several candidates.

Yang et al. [16] have examined a tendency-based technique that predicts the next value by adding/subtracting some value to/from the current measurement. The increment/decrement value may depend on the current measurement and some other dynamic information such as whether the resource is increasing or decreasing. This technique is adaptive and is shown to outperform NWS for CPU load predictions. However, its main drawback is that it is ad hoc and difficult to use.

3. Multi-resource prediction model

From the previous section we can see that correlation is key to the statistical prediction models. Specifically there are two kinds of correlations, the *autocorrelation* that characterizes the statistical relationship of the same resource at different times, and the *cross correlation* that characterizes the statistical relationship between two resources (possibly at different times). To achieve higher prediction accuracy, we should make use of both kinds of correlations (if they are present in the resources). This has motivated us to design the following multi-resource prediction model (denoted as MModel(p, q)):

$$\hat{x}_k = \sum_{i=1}^{p} a_i x_{k\ \ i} + \sum_{i=1}^{q} b_i y_{k\ \ i}, \qquad (5)$$

$$\hat{y}_k = \sum_{i=1}^{p} c_i y_{k\ \ i} + \sum_{i=1}^{q} d_i x_{k\ \ i}. \qquad (6)$$

This means the resource value x_k is predicted from the history values of y as well as the history values of x (The same is true for the prediction of y_k. In the following we will only discuss the prediction of x). Here p is the autoregression order and q is the cross regression order.

Clearly many of the prediction models discussed in Section 2 can be thought of as special cases of the MModel. For example, MModel(1,0) with $a_1 = 1$ is the LAST model, MModel$(p, 0)$ is the AR(p) model, and MModel$(0, 1)$ can be thought of as the linear regression model.

Given the MModel (for x) as in (5), if we multiply it by x_{k-j} (for $j = 1, 2, \dots, p$) and take expectation on both sides, we get p linear equations (for the purpose of model derivation, we substitute x_k for $\hat{x}_k$):

$$r_x(j) = \sum_{i=1}^{p} a_i r_x(|j-i|) + \sum_{i=1}^{q} b_i r_{xy}(i-j) \qquad (7)$$

Here $r_x(j) = E[x_{t+j}x_t]$ is the autocorrelation function (ACF) of x, and $r_{xy}(j) = E[x_{t+j}y_t]$ is the cross correlation function (XCF) of x and y.

If we multiply (5) by y_{k-l} (for $l = 1, 2, \dots, q$) and take expectation on both sides, we get another q linear equations:

$$r_{xy}(l) = \sum_{i=1}^{p} a_i r_{xy}(l-i) + \sum_{i=1}^{q} b_i r_y(|l-i|) \qquad (8)$$

Here $r_y(j) = E[y_{t+j}y_t]$ is the ACF of y.

Given a sample data set of x and y, we can fit the model by first estimating the ACFs and the XCF from the data set, then solving the above $p + q$ linear equations for the coefficients of the model. Once we have these coefficients, we can proceed to make predictions for x.

The above derivation is based on the assumption that the characteristics of the sample data set will remain unchanged over time. This is true for any model that is fitted from a sample data set. In the case of MModel, our model fitting is similar to that of the AR model. Therefore we share the same assumptions with AR that the mean and autocorrelation of the resources are time invariant [4]. Our model fitting also uses cross correlation, therefore we have an additional assumption that the cross correlation is also time invariant. Since the resources have constant means, it is often convienent to treat them as "zero-mean" series plus a constant, so that MModel can always work on zero-mean series.

4. Adaptation techniques

The MModel as presented in Section 3 assumes that the mean and cross correlation of the resources are time invariant. However, this is often not true for resources in a shared environment. For example, we will see (in Section 5.1) that both the mean and cross correlation of the traces we collected may change rapidly over time. This means we must dynamically adapt our MModel, so that its predictions are based on the *current* mean and cross correlation of the resources instead of those computed from the sample data set.

We first consider the *mean adaptation* that addresses the changing mean of the resource. The mean value of a resource is kept in MModel as an internal state. Whenever a new measurement is available, we feed the model with the new data. MModel first subtracts the mean from the new data, this converts the resource to the "zero mean" form. MModel then makes the prediction for the converted data, and adds the mean back to form the final prediction. For resources with constant means, this conversion does not affect the prediction error. For a resource with changing mean, however, this provides us an opportunity to address the problem. Specifically, as new measurement data are made available, we can first update the estimation of the mean using the following EWMA algorithm, the prediction is then made with respect to the updated mean.

$$\begin{aligned} est\_mx &= \alpha \cdot est\_mx + (1-\alpha) \cdot obsx, \\ est\_my &= \alpha \cdot est\_my + (1-\alpha) \cdot obsy. \end{aligned}$$

Here est_mx and est_my are the dynamically estimated means of x and y. $obsx$ and $obsy$ are the new measurement data. α determines how quickly the model responds to the changes in the mean. In our experiments, we find $\alpha = 0.99$ to work fairly well for our traces (Note $\alpha = 1$ corresponds to the case without mean adaptation).

To adapt to the changing cross correlation between two resources is more difficult. This is because the XCF is used to solve the linear equations in (7) and (8), but it is not directly involved in the prediction. One way to adapt to the changing XCF is to dynamically estimate the XCF and solve the linear equations for the new coefficients. But this may be computationally too expensive, especially if multiple streams are being predicted. In order to adapt to the changing XCF in an efficient way, we have developed the following heuristic *cross correlation adaptation* algorithm.

First, we limit the cross regression order q of MModel to be 1. This may limit the potential benefit of using cross correlation, but it also greatly simplifies the adaptation technique. Now the equations in (7) and (8) reduce to the following:

$$\begin{aligned} r_x(j) &= \sum_{i=1}^{p} a_i r_x(|j-i|) + b_1 r_{xy}(1-j), && (9) \\ r_{xy}(1) &= \sum_{i=1}^{p} a_i r_{xy}(1-i) + b_1 r_y(0). && (10) \end{aligned}$$

Next, since the cross correlation changes rapidly, we should probably give less weight to the cross regression term y_{k-1} in (5). Therefore, when we first fit the model or when we do dynamic adaptation, we damp the cross correlation by a damping ratio *dampr*. In our experiments, we find a damping ratio of 4 to be good enough for our traces.

Now if we look at the p equations in (9), since $b_1 r_{xy}(1-j)$ is likely to be small compared with other terms in the equation (due to the damping of $r_{xy}(1-j)$), we can ignore the effect on the coefficients a_i when we adapt b_1. Therefore, we can solve b_1 from the single equation in (10) as the follows,

$$b_1 = \frac{1}{r_y(0)}(r_{xy}(1) - \sum_{i=1}^{p} a_i r_{xy}(1-i)). \qquad (11)$$

Here we make another simplification. We assume that $r_{xy}(j)$ is constant and equal to $r_{xy}(0)$ (which is almost true for our trace data, due to the strong autocorrelation of the resources), then the above can be written as

$$b_1 = \frac{r_{xy}(0)}{r_y(0)}(1 - \sum_{i=1}^{p} a_i). \qquad (12)$$

Thus we finally have the following XCF adaptation algorithm that requires little computation overhead:

$$\begin{aligned} \Delta_x &= obsx - est\_mx \\ \Delta_y &= obsy - est\_my \\ xcf &= \alpha_1 \cdot xcf + (1-\alpha_1) \cdot \Delta_x \cdot \Delta_y \\ vary &= \alpha \cdot vary + (1-\alpha) \cdot \Delta_y^2 \\ b_1 &= xcf/vary \cdot (1 - suma)/dampr \end{aligned}$$

Here xcf is the dynamically estimated cross correlation between x and y. $vary$ is the dynamically estimated variance of y, which is the same as $r_y(0)$ when y is zero mean. $suma = \sum_{i=1}^{p} a_i$ can be precomputed once the model is fit. α is the same as in mean adaptation, and α_1 is chosen to be 0.9 because we would like the model to respond more quickly to the changes in cross correlation.

5. Model evaluation

To evaluate the adaptive MModel (MModel-a), we have chosen to experiment with CPU load prediction because CPU resource is one of the most important resources in a grid environment. Many of today's grid applications are CPU bound, yet the computing power has been growing at a slower rate than other resources such as data storage capacity and network bandwidth [8].

In the following we first describe the traces we collected and their time-varying characteristics. Next our model evaluation method is given. Finally we present the performance results of MModel-a for the traces.

5.1. Traces and their characteristics

We have collected two sets of trace data. The first set is collected on the Sun workstations in the CS Instructional Lab (CSIL) at UIUC. We collected the 1 minute CPU load average and free memory on five of the machines (*suna1, suna5, suna12, sunb2* and *sunb21*) for several days starting from Nov. 8, 2002. One measurement is made for every second.

The second set is collected from the NCSA Origin 2000 supercomputer arrays. These are distributed shared memory architectures and were part of the NCSA Allicance Grid at the time we collected our traces. [1] NCSA has an online webpage [2] that publishes the dynamic information about these machines such as the 15 second host load (*r15s*) and the 1 minute host load (*r1m*). We have used a data collection program that retrieves the webpage every five seconds and extracts the *r15s* load and *r1m* load for our experiments. Our traces include twelve machines of the array (*aegir, brono, eir, forseti1, hermod, hod1, huldra, jord1, mimir, modi2, nerthus* and *saga1*) for more than one week starting from Mar. 20, 2003.

Before using the traces to evaluate MModel-a, we have used a simple sliding window technique to test if the traces have time-varying characteristics . We find that for the CSIL CPU load traces, if we randomly select a window of 600 points in a trace and compute its mean, and compare it with the mean of the next window with the same length, on average there will be a change of more than 20%. For the grid traces, we also find a change in the mean for most of the cases, although the changes come in different ways (See our classification of these traces in Section 5.3.2).

We have used the same technique to test if the cross correlation between two resources also changes. We find that for the CSIL traces, the cross correlation between CPU load and free memory changes rapidly between consecutive windows, not only in magnitude, but also in the way the resources are correlated. For example, it is often the case that the two resources are negatively correlated in the first time window, but positively correlated in the next one. This is because the resources are shared by many applications, and different applications may have very different resource usage patterns. For the grid traces, since the *r15s* load and *r1m* load are two different measurements of the real

[1] Recently the Origin arrays have been replaced by the more powerful IBM pSeries 690 servers, which are also shared memory machines and part of the Alliance Grid.

load (both are computed using EWMA, but with different parameter α), their correlation is more stable. But on average there are still near 10% changes between consecutive windows. The changing characteristics of the traces means our adaptation techniques in Section 4 are necessary for accurate resource predictions.

5.2. Evaluation method

We evaluate MModel-a by comparing its performance with that of the AR model using randomized test similar to those in [7]. For a given trace, we randomly select a starting position in the trace and fit the models to a sample data set of size 600. We then do prediction for the next 9000 values (i.e., for each of these values, we feed it into the model and produce a prediction, and compute the prediction error, until we have finished 9000 predictions). The sample data set size is chosen to be 600 so that the AR model can adequately capture the autocorrelation of the resources. The number of predictions is chosen to be 9000 so that it is large enough to show the benefits of adaptation techniques. In order to examine the benefits of the two adaptation techniques individually, we also implement an AR model with mean adaptation only (ARM) and show its prediction performance (note ARM is also a special case of MModel-a). We do this test for a certain number of times (e.g., 200) and calculate the expected sum of square error (SSE), which is equivalent to the mean square error (MSE) since the number of predictions is fixed.

Both MModel-a and the AR model can be parameterized with different orders. Since MModel-a can be thought of as an AR model with additional cross regression terms, we would like to know how this added information would improve the prediction. Therefore, we compare MModel-$a(p, 1)$ with AR(p) for different p values in our experiments.

5.3. Performance results

In this subsection, we present experimental results to evaluate the performance of MModel-a. We first present the results for the workstation traces, where we use the CPU load and free memory for the prediction of CPU load. Next we look at the results for the grid traces, where we use the *r15s* and *r1m* load for the prediction of the *r1m* load. The performance of MModel-a for n-step ahead predictions is then presented. And finally we briefly compare MModel-a with the tendency-based model [16].

5.3.1. Results for workstation traces.

Figure 1 shows the expected sum of square error (SSE) of AR, ARM and MModel-a for the *suna1* CPU load prediction. The AR and ARM are based on the CPU load trace only. MModel-a is based on the CPU load and free memory traces.

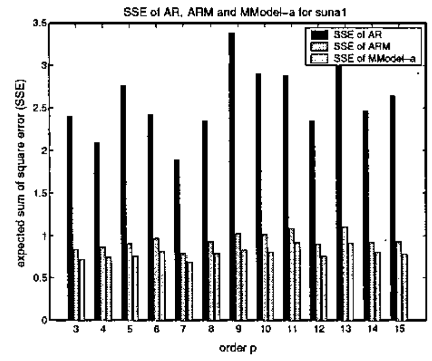

Figure 1. Expected sum of square error (SSE) of AR, ARM and MModel-a for the *suna1* trace.

From the figure we can clearly see the benefit of adaptive multi-resource prediction. The SSE of MModel-a is consistently smaller than that of ARM, and the SSE of ARM is consistently smaller than that of AR. The use of mean adaptation seems to achieve more significant improvements than the use of cross correlation (with adaptation). With mean adaptation only, we can achieve more than 55% reduction in SSE compared with AR. With cross correlation adaptation, we can achieve another 13% to 18% reduction in SSE. Overall, the improvement of MModel-a over AR is around 70%.

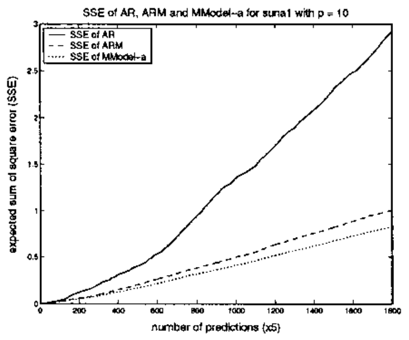

Figure 2. Trend of expected SSE for AR, ARM and MModel-a with $p = 10$.

Figure 2 shows for $p = 10$, how the expected SSE of AR, ARM and MModel-a increases with time. We

Table 1. Characteristics of the workstation load traces and the performance of MModel-a

	suna1	suna5	suna12	sunb2	sunb21
mean	.3746	.0596	.2049	.0329	.0389
var	.2836	.0048	.2970	.0007	.0011
impr_m	62%	12%	70%	6%	4%
impr_x	15%	2%	9%	3%	2%
impr_mx	68%	13%	73%	9%	6%

can see that initially the prediction error of all three models are very close to each other. After about 600 predictions, however, the SSE of AR begins to grow more quickly. The SSEs of ARM and MModel-a continue to grow at linear rate, which means their MSE is constant over time, due to their ability to adapt to the changes of the resources.

We have experimented with other workstation traces. For the *suna12* trace, we have similar results as for the *suna1* trace. For the other three traces, however, the improvement of MModel-a over AR is not as large.

We have examined the reason for the small improvement of the MModel-a for these traces. We find that it is because these traces have very small mean and variance in load. Table 1 shows the mean and variance of the load traces and the improvement of the two adaptation techniques. *impr_m* is the improvement of ARM over AR, *impr_x* is the improvement of MModel-a over ARM, and *impr_mx* is the improvement of MModel-a over AR. The percentage improvement is the median for all the p orders.

From Table 1 it is clear that the performance of MModel-a is largely related to the variance and mean of the CPU load. Generally, if the variance of the load is large, we would expect large improvement from mean adaptation. Since large variance often implies more changes in the mean, it is not surprising that mean adaptation works better for such traces. The improvement from cross correlation adaptation seems to be related to the mean of the load. This is because the CPU load and free memory are correlated only when there are processes running. Therefore, heavily loaded traces are likely to have more cross correlation, and hence more improvement is achievable from cross correlation adaptation. However, we should note that this is only general rule and it may not hold if there are only small differences between two load traces. Since distributed shared environment is often highly dynamic and heavily loaded, we expect the benefit of MModel-a would be even more significant in such environment.

5.3.2. Results for the grid traces. In this subsection, we show the performance results of MModel-a for the *r1m* load prediction for the grid traces based on the *r15s* and the *r1m* load traces. Before we present the performance results, we note that these traces exhibit wide range of dynamics. Therefore summary statistics such as mean and variance are no longer adequate to characterize the traces. However, we can still identify three kinds of load changes that distinguish the traces:

(1).*Sudden local change*, this means the load suddenly rises to a very high value or drops to a very low value, and then quickly (within several minutes) returns to the previous level.

(2).*Sudden level change*, this means the load suddenly changes to a different level and stays there for significant amount of time (tens of minutes or even longer).

(3).*Gradual level change*, this means the level of the load gradually changes over time.

Based on these characteristics, we can roughly classify the traces into four classes:

Class 1: The predominant dynamics of the trace are sudden local changes. *brono* is the only trace in this class.

Class 2: The predominant dynamics of the trace are sudden level changes. *hod1* is the only trace in this class.

Class 3: The predominant dynamics of the trace are gradual level changes. *aegir, eir, huldra, jord1, mimir* and *saga1* belong to this class. Among these, *aegir* and *saga1* also exhibit some sudden local changes.

Class 4: There are both sudden local and level changes. *forseti1*, *hermod*, *modi2* and *nerthus* belong to this class.

Given that we are using the 15 second and 1 minute CPU load for the 1 minute CPU load prediction, we would expect that when there are sudden load changes (either local or level changes), the cross correlation adaptation should be able to improve the prediction. Because any sudden load change will first reveal itself in the 15 second trace, and eventually in the 1 minute trace. Similarly, when there are changes in the level of the load (either sudden or gradual changes), the mean adaptation should be able to improve the prediction, because the mean adaptation is designed to address the changes in the mean (or level) of the traces.

Table 2 shows the performance results of MModel-a for the grid traces (The notations are the same as in Table 1). From the table we can see that the results are largely as expected. For the *brono* trace, since sudden local change is the predominant changes in the load, we see a large improvement of 55.5% from the cross correlation adaptation. The improvement from mean

Table 2. MModel-a performance results for the grid traces

class	machine	impr_m	impr_x	impr_mx
1	brono	14.68%	55.50%	61.78%
2	hod1	93.02%	14.11%	94.01%
3	aegir	68.79%	9.66%	71.80%
	saga1	62.35%	17.96%	69.11%
	eir	26.55%	1.52%	27.80%
	jord1	35.77%	2.38%	37.38%
	huldra	46.25%	4.36%	48.40%
	mimir	65.72%	2.52%	66.42%
4	hermod	83.41%	31.47%	88.57%
	forseti1	86.10%	37.56%	91.31%
	modi2	93.74%	33.73%	95.94%
	nerthus	92.97%	33.63%	95.37%

adaptation for *brono* is the smallest among all traces, because the overall level does not change over time. For *hod1*, since the predominant load change is sudden level change, we see a large improvement from mean adaptation. The cross correlation also improves the prediction, because there are sudden changes in load.

For the Class 3 traces, since the level of the load only gradually changes, the improvement from mean adaptation is not as large as that for Class 2 and Class 4. And except for *aegir* and *saga1*, which have more sudden local changes, the improvement from cross correlation adaptation is marginal. This is because compared with the 15 second load, the 1 minute load is a more smoothed version of the real load. Therefore small changes in the 15 second load may be smoothed away and do not exhibit in the 1 minute load (i.e., the correlation between the two observations is small).

For the last class of traces, there are both sudden local and level changes, therefore we have large improvement from both mean adaptation and cross correlation adaptation. Overall, the improvement of MModel-a over AR for this class is around 90%.

5.3.3. N-step ahead prediction. In the previous experiments we only do one-step ahead predictions. In a distributed shared environment, however, it is more important to have medium or long term predictions. The n-step ahead prediction can be produced by first generating the one-step ahead prediction. Treat this prediction as a new measurement and generate the 2-step ahead prediction, etc., until the n-step ahead prediction is reached [4]. [2]

[2]An alternative way to do long term prediction is to aggregate the resource measurements into coarse grained time series and perform one-step ahead perdiction on this series. This is unlikely to result in more accurate predictions because there would be less correlation in the aggregated series. In addition, the method considered here allows resource prediction for a small interval of time in the future, which is impossible for the aggregated series.

In this subsection, we examine how MModel-a performs as the lead time n increases. Using the grid traces, we conduct n-step ahead prediction for $n = 1, 3, 5, 7, 10, 15, 20$ and 30. Figure 3 shows the percent-

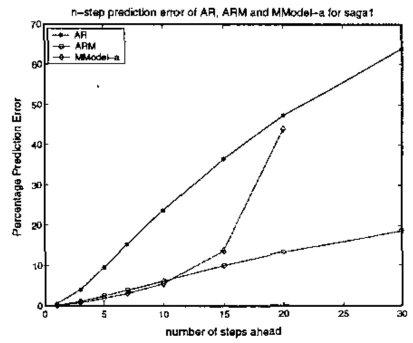

Figure 3. Percentage prediction error of AR, ARM and MModel-a (with $p = 10$) for the *saga1* trace.

age prediction error (i.e., the expected mean square error divided by the variance of the trace) of AR, ARM and MModel-a for the *saga1* trace (with $p = 10$).

From the figure we can see that for one-step ahead predictions, all the three models have small MSEs compared with the variance of the load itself. However, as the number of steps n increases, the prediction error of AR quickly increases. When $n > 5$, the n-step ahead percentage prediction error of AR is greater than 10%. The prediction error of ARM only slowly increases with n, and for 30-step ahead prediction (which is 150 seconds into the future), the percentage prediction error of ARM is still about 18%. The prediction error of MModel-a remains the smallest until $n = 10$. After that it quickly increases. The figure does not show the prediction error of MModel-a for $n = 30$ because it has already diverged.

Our experiments with other grid traces and different orders show the same pattern. In summary, we have the following observations:

1. The AR model is unfit for long term predictions, because its prediction error will quickly increase when the number of steps is greater than 1.

2. The MModel-a performs the best for near term predictions (e.g., for $n \leq 8$). Thereafter its prediction error will increase explosively. This is because we do not do adaptation when a predicted

value is fed back into the model. Therefore as n increases, the cross regression coefficients become increasingly inaccurate, which leads to large prediction errors (Remember these coefficients are not accurate even for one-step ahead predictions due to the simplications we made for the adaptation technique).

3. The ARM seems to be suitable for long term predictions. In fact, except for *brono*, the 30-step ahead prediction errors for all the traces are less than 20%. Therefore, if long term prediction is desired, we should probably disable the use of cross regression terms in the MModel-a.

5.3.4. Comparison with tendency-based model. In [16], a tendency-based model has been shown to perform well for the prediction of the CPU load traces collected by Dinda [6]. We have experimented with the same traces and the same error metric using our ARM model (MModel-a requires two related resources which are not available from Dinda's traces). For space reasons the detailed results are not shown here. However, we note that for those traces for which the tendency model has small prediction errors, ARM has comparably small errors. For those that the tendency model has large errors, however, ARM can often achieve as much as 50% reduction in the prediction error. We believe this is because ARM is better at capturing the autocorrelation of the resources than at hoc techniques.

6. Conclusion

We have designed a multi-resource prediction model that uses both autocorrelation and cross correlation to achieve higher prediction accuracy. We have also presented two adaptation techniques that enable the MModel to adapt to the changing characteristics of the underlying resources. Experimental results show that MModel-a can achieve more accurate predictions than other models, especialy for highly dynamic resources and long term predictions.

We should note that correlation of the resources is the basis for statistical predictions. If two resources are not statistically correlated, we can not expect the use of cross correlation to improve the prediction. However, MModel-a provides an important framework to make use of such correlations, if they are present in the resources.

The idea of adaptive prediction is also important, due to the highly dynamic nature of a shared environment. Our mean adaptation makes no assumption about the resources and should be applicable to other resources. Our cross correlation adaptation has made some simplifications that may not be valid for other resources, for which new adaptation techniques may need to be developed.

References

[1] NCSA Alliance Grid, http://www.ncsa.uiuc.edu /projects/alliance/.

[2] NCSA Origin 2000 online load monitoring webpage. https://internal.ncsa.uiuc.edu/cgi-bin/CC/systems/ info-sgi.sgi?load.

[3] F. Berman and R. Wolski. Scheduling from the perspective of the application. In *The 5th IEEE Symposium on High Performance Distributed Computing (HPDC5)*, Syracuse, New York, August 1996.

[4] G. Box, G. Jenkins, and G. Reinsel. *Time Series Analysis, Forecasting and Control.* Prentice Hall, 1994.

[5] H. Dail, H. Casanova, and F. Berman. A decoupled scheduling approach for the grads program development environment. In *Supercomputing '02*, 2002.

[6] P. Dinda. The statistical properties of host load. Technical report, CMU, 1998.

[7] P. Dinda and D. O'Hallaron. Host load prediction using linear models. In *The 8th IEEE Symposium on High Performance Distributed Computing*, 1999.

[8] I. Foster. The Grid: A new infrastructure for 21st century science. *Physics Today*, 55(2):42–47, 2002.

[9] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the Grid: Enabling scalable virtual organizations. *International J. Supercomputer Applications*, 2001.

[10] W. Rich, S. Neil, and H. Hayes. Predicting the cpu availability of time-shared unix systems on the computational grid. In *The 8th IEEE Symposium on High Performance Distributed Computing*, 1999.

[11] M. Swany and R. Wolski. Multivariate resource performance forecasting in the network weather service. In *Supercomputing '02*, 2002.

[12] S. Vazhkudai and J. Schopf. Predicting sporadic grid data transfers. In *The 11th IEEE Symposium on High Performance Distributed Computing*, 2002.

[13] S. Vazhkudai and J. Schopf. Using disk throughput data in predictions of end-to-end grid data transfers. In *The 3rd International Workshop on Grid Computing (GRID 2002)*, November 2002.

[14] R. Wolski. Dynamically forecasting network performance using the network weather service. In *Cluster Computing*, 1998.

[15] R. Wolski, N. T. Spring, and J. Hayes. The network weather service: A distributed resource performance forecasting service for metacomputing. *The Journal of Future Generation Computing Systems*, 1999.

[16] L. Yang, I. Foster, and J. M. Schopf. Homeostatic and tendency-based cpu load predictions. In *Proc. of IPDPS 2003.*

[17] L. Yang, J. M. Schopf, and I. Foster. Conservative scheduling: Using predicted variance to improve scheduling decisions in dynamic environment. In *Supercomputing '03*, Phoenix, 2003.