Homeostatic and Tendency-based CPU Load Predictions

Lingyun Yang[1] Ian Foster[1,2] Jennifer M. Schopf[2]
[1]Department of Computer Science, University of Chicago, Chicago, IL 60637
[2]Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439
[lyang, foster]@cs.uchicago.edu jms@mcs.anl.gov

Abstract

The dynamic nature of a resource-sharing environment means that applications must be able to adapt their behavior in response to changes in system status. Predictions of future system performance can be used to guide such adaptations. In this paper, we present and evaluate several new one-step-ahead and low-overhead time series prediction strategies that track recent trends by giving more weight to recent data. We present results that show that a dynamic tendency prediction model with different ascending and descending behavior performs best among all strategies studied. A comparative study conducted on a set of 38 machine load traces shows that this new predictor achieves average prediction errors that are between 2% and 55% less (36% less on average) than those incurred by the predictors used within the popular Network Weather Service system.

1 Introduction

In multi-user time-shared systems, applications are in active competition with unknown background workloads introduced by other users. The contention that results from the resource sharing tends to cause load and resource availability to vary over time. The impact can be particularly significant on a *computational Grid* [1], which links interconnected but geographically distributed computing resources as a cooperating set of resources. Clearly, predictions of system performance are necessary for efficient use of such resources. Performance predictions can be useful to both applications and schedulers. Applications can use predictions to adapt their behavior in response to changes in system status to get better performance[2, 3]. Schedulers can use predictions to guide their scheduling strategies and thus to achieve higher application performance and more efficient resource use[4-6].

Varying CPU load has a significant effect on the running time of CPU-bound applications. Indeed, for certain types of applications the running time of a compute-bound task is linearly proportional to the average CPU load it encountered during the execution [4, 7]. The focus of this paper is predicting the CPU load of shared computing resources. Our contribution is to introduce a new time series prediction technique that behaves better than other techniques used previously. Rather than giving the same consideration to the history data within a "sliding window" as do traditional linear models and mean- or median-based models, our one-step-ahead time series prediction strategies give more weight to more recent measurements than to other history data. We also allow for the use of different behaviors when "ascending" and "descending." Our experimental results show that, on a range of host load measurement datasets, our dynamic tendency strategy with different ascending and descending behavior consistently outperforms the nine predictors used within the Network Weather Service (NWS) [8-11], a widely used performance prediction system.

The rest of the paper is structured as follows. Section 2 introduces background and related work. Section 3 gives a detailed description of the prediction strategies we studied. Section 4 describes the experimental results when our prediction strategies are applied to actual measurements and compared with those of other researchers. Section 5 presents our conclusions and notes directions for further work.

2 Background and related work

Previous efforts [9, 12] indicate that CPU load is strongly correlated over time, which implies that history-based load prediction schemes are feasible. We believe that the key to making accurate predictions is to correctly model the relationship of the history data with the future values.

Time series modeling has been studied widely in many areas, including financial data prediction [13-15], earth and ocean sciences [16], biomedical signal processing [17], and networking [18, 19]. In the area of CPU load prediction, the Network Weather Service [8-11] provides one-step-ahead predictions for any time-series fed to its predictor module. NWS is a distributed system that periodically monitors and dynamically forecasts the

performance of various network and computational resources. NWS applies a collection of one-step-ahead prediction strategies to time series and chooses the strategy used for the "next" prediction dynamically according to which strategy has been most accurate over recent measurements. The prediction strategies used by NWS currently include running average, sliding window average, last measurement, adaptive window average, media filter, adaptive window media, α-trimmed mean, stochastic gradient, and autoregressive [8, 9]. Dynamical selection of the best prediction strategy on the fly has resulted in predictions equivalent to, or slightly better than, the best predictor in the set. We note that while for the purposes of comparison we measure the improvements achieved by our new prediction relative to those used within NWS, our new strategies could easily be included as predictors within the NWS framework. Thus, our work does not invalidate the NWS approach but rather shows that its choice of predictors can be improved.

Dinda et al. evaluated multiple linear models, including autoregressive (AR), moving average (MA), autoregressive moving average (ARMA), autoregressive integrated moving average (ARIMA), and autoregressive fractionally integrated moving average (ARFIMA) models [7]. Their results show that the simple AR model (also used in NWS) is the best model of this class because of its good predictive power and low overhead. More complex liner models are expensive to fit and hence difficult to use in a dynamic or real-time setting. Our approach, as shown in Section 4, performs better and has less overhead than these approaches.

3 Prediction strategies study

This section defines our prediction strategies. Each strategy predicts the one-step-ahead value based on a fixed number of immediately preceding history data measured at a constant-width time interval. We present two families of strategies: (1) homeostatic prediction strategies, Section 3.1, and (2) tendency-based prediction strategies, Section 3.2.

We use the following notation in the descriptions of the prediction strategies:

V_T: the measured value at the T_{th} measurement,

P_{T+1}: the predicted value for measurement value V_{T+1},

N: the number of data points of history data used in the prediction, also called the window size.

3.1 Homeostatic prediction strategies

What we term *homeostatic prediction strategies* work on the assumption that if the current value is greater (less) than the mean of history values, then the next value is likely to decrease (increase). The basis of this approach is to be "self-correcting" so that they return to the mean of the history value. More formally, this kind of strategy can be expressed as following:

if (V_T > $Mean_T$) then
  $P_{T+1} = V_T$ – *DecrementValue;*
  [Optional DecrementValue adaptation process]
else if (V_T < $Mean_T$) then
  $P_{T+1} = V_T$ + *IncrementValue;*
  [Optional IncrementValue adaptation process]
else
  $P_{T+1} = V_T$;

where $Mean_T$ is the mean of the N history data points. It can be calculated by the following formula:

$$Mean_T = (\Sigma_{i=1..N} V_i)/N \quad (1)$$

At every prediction step, the increment value or decrement value can be an independent value or a relative value that is proportional to the current measurement. The increment or decrement value can be "static" such that it is fixed for all prediction steps, or "dynamic" such that it is adapted to the time series at each step. The different combinations of the above configurations result in four homeostatic prediction strategies: independent static, independent dynamic, relative static, and relative dynamic. We present the detailed description of these strategies next.

3.1.1 Independent static homeostatic prediction strategy. The independent static homeostatic strategy generates a prediction by changing the current value by a fixed amount, without any adaptation process. For this strategy, the decrement (increment) value remains fixed for the run of the experiment, and the increment or decrement value can be expressed by the following formula:

DecrementValue=DecrementConstant (2)
IncrementValue=IncrementConstant

The decrement (increment) constant may change depending on the training set. Values between 0.05 and 1 are reasonable. Our selection for this value is discussed in Section 4.1.

3.1.2 Independent dynamic homeostatic prediction strategy. The independent dynamic homeostatic strategy dynamically adjusts the amount of the increment or decrement value by means of an adaptation process:

DecrementValue adaptation process:
  Measure $V_{T+1;}$
  $RealDecValue_T = V_T - V_{T+1;}$
  $DecConstant_{T+1} = DecConstant_T + (RealDecValue_T - DecConstant_T) * AdaptDegree;$

IncrementValue adaptation process:
  Measure V_{T+1};
  $RealIncValue_T = V_{T+1} - V_{T;}$
  $IncConstant_{T+1} = IncConstant_T + (RealIncValue_T - IncConstant_T) * AdaptDegree;$

At each time step, after we measure the real data (V_{T+1}), we calculate the difference between the current

measured value and the last measured value, thus determining the real decrement (increment) we should have used in the last prediction in order to get the actual value. We adapt the value of the decrement (increment) value accordingly and use the adapted IncrementConstant (or DecrementConstant) to predict the next data point. The parameter AdaptDegree can range from 0 to 1 and expresses the adaptation degree of the variation. If AdaptDegree is equal to 0, the $DecConstant_T$ ($IncConstant_T$) is not adapted at all, and we have *nonadaptation* (or a static approach). If AdaptDegree is equal to 1, the $DecConstant_T$ ($IncConstant_T$) is equal to $RealDecValue_T$ ($RealIncValue_T$), or *full adaptation*.

We define the value of AdaptDegree by using an experimental method described in Section 3.3. The value that results in minimal average error rate is considered as suitable and the AdaptDegree will change depending on the training set. Values between 0.05 and 1 are reasonable. Our selection for this value is discussed in Section 4.1.

3.1.3 Relative static homeostatic prediction strategy. The relative static homeostatic strategy assumes that a larger load value has more potential to change than does a smaller load value. Thus, this strategy modifies the independent static homeostatic prediction strategy so that the increment or decrement applied to a prediction is proportional to the current value instead of a constant value. For this strategy, the increment or decrement value can be expressed by the following formula:

$$DecrementValue = V_T * DecrementFactor \quad (3)$$
$$IncrementValue = V_T * IncrementFactor$$

Increment or decrement factors between 0.05 and 1 are reasonable. Our selection for these values is discussed in Section 4.1.

3.1.4 Relative dynamic homeostatic prediction strategy. The relative dynamic homeostatic strategy alters the prediction value by a relative amount, as does the relative static homeostatic strategy, but allows the value of IncrementFactor and DecrementFactor to be adapted dynamically, using the same method as in the independent dynamic homeostatic prediction strategy. Again, the methodology for selecting the constants is described in Section 4.1.

3.2 Tendency-based prediction strategies

Unlike the homeostatic strategies that predict the future value according to the current value and the mean of the history value. Our second family of prediction strategies predicts the next value according to the tendency of the time series change. This approach assumes that if the current value increases, the next value will also increase and that if the current value decreases, the next value will also decrease. Formally, tendency based prediction strategies can be expressed as follows:

```
//Determine Tendency
if ((V_T - V_T-1)<0)
      Tendency="Decrease";
else if ((V_T-1 - V_T )<0)
      Tendency="Increase";
if (Tendency="Decrease") then
      P_T+1 = V_T – DecrementValue;
      DecrementValue adaptation process
else if (Tendency="Increase") then
      P_T+1 = V_T + IncrementValue;
      IncrementValue adaptation process
```

Like the homeostatic prediction strategies, the variation (DecrementValue and IncrementValue) can be an independent or relative value that is proportional to the current value. Since the static prediction strategies always give worse results than does a simple last-value prediction strategy in the initial experiments, we exclude the static case from this discussion.

Tendency-based strategies have an additional possible source of error. It is impossible to predict when a time series is going to "change direction" – that is, when an increasing time series will become a decreasing one, or vice versa. Because of this, a large error can occur at the *turning point*, or that value when the time series changes direction. For example, if the time series value begins to drop after several successive increases, the tendency prediction strategy will still predict the value to increase at this turning point, when the value has actually decreased. In this case, the prediction error is the sum of the predicted increasing variation and the actually decreasing variation. If the increment variation is adapted to a very big value, a large error can occur.

To minimize this kind of error, we need to reduce the variation at the possible turning points. The basic idea is that if the current value increases to a very high value (or decreases to a very low value), it is possible that a turning point is about to occur. In this case, instead of adapting the variation according to the measured value as usual, we would like to adapt it to a smaller degree to minimize possible errors.

In our implementation, we use the mean of the history data as the threshold value. In the increase phase, if the current data is smaller than the threshold value, the variation will be adapted normally; if the time series increases to a value that is bigger than the threshold value, it is possible that the next step is a turning point. We calculate the percentage of the history data that is greater than the current data and use this value as the possibility of current data *not* to be a turning point. The larger the current value is, the more possible that it is the turning point, and the less the percentage of the history data bigger than it is. So the IncrementValue adaptation process can be expressed in the following way:

$$Mean_T = (\Sigma_{i=1..N} V_i)/N;$$

$RealIncValue_T = V_{T+1} - V_T$;
$NormalInc = IncValue_T + (RealIncValue_T - IncValue_T) *$ AdaptDegree;
if ($V_{T+1} < Mean_T$) // normal adaptation
$IncrementValue_{T+1} = NormalInc$;
else
$PastGreater_T$=(the number of past data points greater than V_T) /N;
$TurningPointInc = IncValue_T * PastGreater_T$;
$IncrementValue_{T+1}$=Min(abs(NormalInc), abs(TurningPointInc));

NormalInc is the value of the $IncrementValue_{T+1}$ in the case of normal adaptation. When the current value is higher than $Mean_T$, it may be a turning point, and the value of $PastGreater_T$ (the percentage of the past time series values greater than the current value) will be small (<0.5). Hence, the possibility that the current value is *not* the turning point is small, so we adjust the increment value accordingly. If we predict the value to go in the wrong direction, the error is still small.

Similarly, the DecrementValue can be adapted in the same way by using the percentage of the history data smaller than current value when the current value decreases to a value that is smaller than the threshold value.

3.2.1 Independent dynamic tendency prediction strategy. The independent dynamic tendency strategy predicts the next step value by adding or subtracting an independent increment or decrement value from the current value according to the tendency of the value change. For this strategy, we determined the increment and decrement values just as we determined them for the static and dynamic independent homeostatic prediction approaches, discussed in Sections 3.1.1 and 3.1.2.

3.2.2 Relative dynamic tendency prediction strategy. The relative dynamic tendency strategy is similar to the independent dynamic tendency prediction strategy except that the increment value or decrement value is in proportion with the current value. The increment and decrement values are determined in the same way as for the relative static or dynamic homeostatic approaches, discussed in Sections 3.1.3 and 3.1.4.

3.2.3 Dynamic tendency prediction strategy (mixed variation). During initial experiments, we observed that the independent tendency prediction strategy resulted in better predictions during an increase phase and that the relative tendency prediction strategy generally resulted in better predictions during a decrease phase. One possible explanation of this phenomenon is that while a CPU time series is increasing, the independent tendency strategy better tracks the behavior due to very small increases that are independent of the actual value of the prediction, but that during the decrease phase the relative prediction strategy applies a value is proportional to the current value more in keeping with the trend of the load behavior. The results of our further experiments (Section 4) also confirmed this tentative explanation. This may suggests some intrinsic property of CPU load data. The further explanation is left as an open question.

Because of this initial result, we define a mixed tendency-based prediction strategy that predicts the next value for an increase phase using the independent tendency prediction strategy and for a decrease phase uses the relative tendency prediction strategy, that is,

$$DecrementValue = V_T * DecrementFactor \quad (4)$$
$$IncrementValue = IncrementConstant$$

For completeness, we examined the use of the independent constant in the decrement phase and a relative value in the increment phase, but worse predictions resulted in all cases.

4 Experimental evaluation

We ran a series of experiments using our predictors in order to validate their effectiveness under a variety of conditions. We break these experiments into two sets. In the first set, we ran all of our predictors on a small set of time series over which we had complete control, and we evaluated the effect of different collection rates on our own predictors, on a simple last-value predictor, and on the Network Weather Service. In the second set of experiments, we ran a much larger set of 38 load traces and evaluated only our best predictor and the NWS.

The last-value predictor uses the current measured value as the predicted value of the next measurement. It can be expressed by the following formula:

$$P_{T+1} = V_T$$

Harchol-Balter and Downey [20] show that this is a useful prediction strategy for CPU resources. It has low computation and storage overhead and is the default predictor in several current systems because of its simplicity.

The NWS [8-11] dynamically selects the best predictor from a set that includes mean-based prediction strategies, median-based prediction strategies, and AR model-based prediction strategies. It has been shown to yield forecasts that are equivalent to, or slightly better than, the best forecaster in the set. This implies that if our prediction strategy performs better than NWS predictor, it can perform better than all the prediction techniques in the set.

We did no model fitting for any of the experiments, as is commonly needed in linear regression techniques. The parameters were defined by using training data off-line before the experiments, as described in Section 4.1, and were not redefined during the experiments. Thus, we minimized the run-time cost of these strategies; on average, this cost is only a few milliseconds per prediction.

4.1 Input parameters

All of the prediction strategies we defined need certain input parameters to determine how much to increment or decrement a value or how strongly to change a prediction over time. In all cases, we determined these values by running a set of experiments to exhaustively search the space of feasible selections. We did this using training data that was not a part of the data used for our experiments. We used 25 one-hour-long time series and evaluated increment and decrement values at intervals of 0.05 between 0 and 1 using the error formula

$$\text{Average Error Rate} = \frac{\sum_{i=1..N} abs(P_i - V_i)/V_i}{N} * 100\% \quad (5)$$

Where N is the number of data in the time series to be tested. To avoid the " division overflow" error, we add a small value -delta to V_i (measured value) in case it is equal to zero. We set delta equal to 0.01 in our experiment. The value that results in minimal average error rate is considered as suitable.

For our experiments, we found the best results with

IncrementConstant= DecrementConstant = 0.1
IncrementFactor = DecrementFactor = 0.05
AdaptDegree = 0.5

These values, while somewhat ad hoc, worked well over a wide set of traces. We used the same values for all of our predictions regardless of the characteristics of the trace each experiment was being run on. It would be possible to tune these parameters for each trace, or to change them on the fly to better fit a particular load behavior, but we felt that was unnecessarily complicated and would significantly add to the overhead.

We also studied the sensitivity of the mixed variation prediction strategy to a selection of AdaptDegree parameter values. We did this by running this strategy on 38 time series collected by Dinda [12], while changing the value of the AdaptDegree from 0 (nonadaptation) to 1 (full adaptation), increasing 0.1 at each step. Figure 1 shows the resulting prediction errors, while Figure 2 shows the means and standard deviations. We find that no one value of the parameter is the best for all time series: the average prediction error shows different varying tendency while the value of AdaptDegree changes. Moreover, except in the case of nonadaptation, the variation of the prediction error for different AdaptDegree values is not a significant fraction of the total. We conclude that the value of the parameter does not have a significant influence on prediction capability of our prediction strategy, as long as extremes are avoided, and select an intermediate value of 0.5 for future studies.

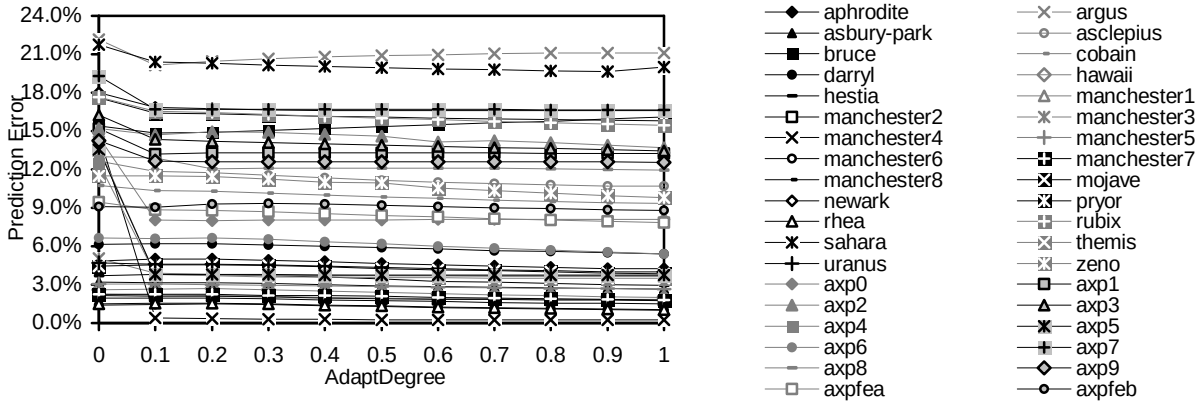


Figure 1: The prediction error of the mixed tendency prediction strategy as a function of the AdaptDegree parameter on 38 different time series.

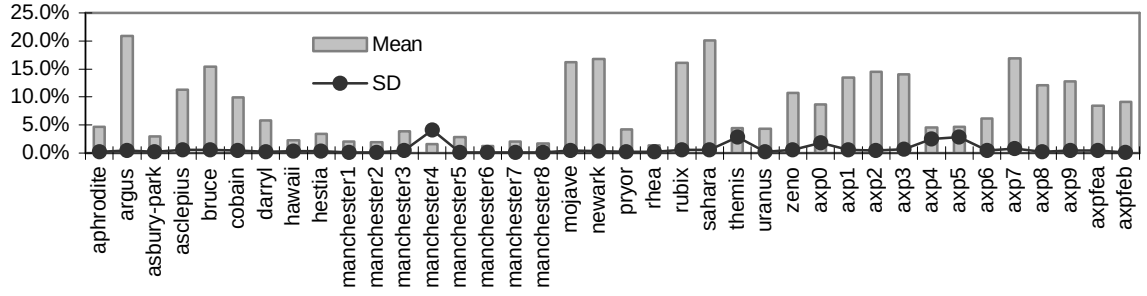


Figure 2:Mean and standard deviation of the prediction errors shown in Figure 1 for each time series.

4.2 Prediction strategy evaluation

We ran a set of experiments to evaluate the six prediction strategies presented in Section 3 and to determine what effect, if any, different measurement rates have on the predictions. Specifically, we evaluated one-step-ahead time series prediction strategies on CPU load time series collected from four machines on different time scales but during the same period of time. For each machine, we collected one set of data (spanning roughly 28 hours) and then examined it as three different time series: 0.1 Hz (measure the data every 10 seconds) with roughly 10,000 data points; 0.05 Hz (measure the data every 20 seconds), so throwing out every other data point from the 0.1 Hz trace, and 0.025Hz (measure the data every 40 seconds) or throwing out 3 of every 4 data points from the 0.1 Hz collection.

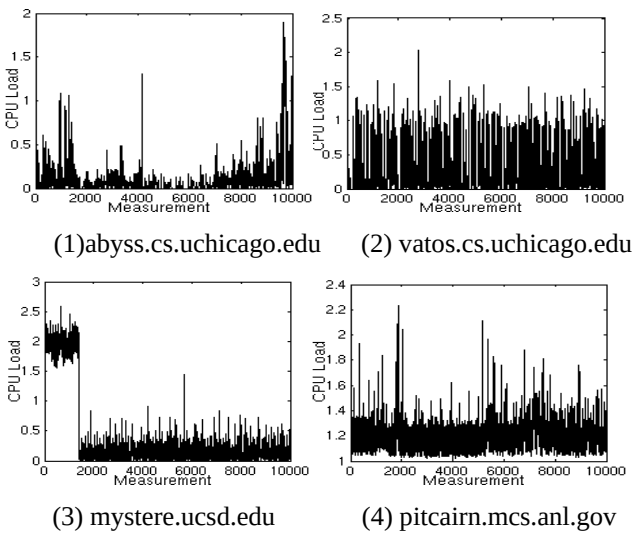
The four machines show different CPU statistical properties during our measurement, as illustrated in Figure 3. We discuss only the 0.1 Hz time series; the other two resolutions share the similar properties, since the data were collected during the same period.

abyss.cs.uchicago.edu is a server machine at the University of Chicago. This machine had very low CPU load with about 40% data equal to zero and mean equal to 0.08 during our measurement. The time series are relatively stable. The standard deviations are about 0.1755 for the 0.1 Hz time series.

vatos.cs.uchicago.edu is a server machine at the University of Chicago. This machine had low mean CPU load (0.24) during our measurement. On this machine, the time series are not stable and hence have higher standard deviations: about 0.3188 for the 0.1 Hz time series.

mystere.ucsd.edu is a server machine at UCSD. This machine had low CPU load (with occasional peaks) during our measurement (0.4). As the figure shows, on the left part of the 0.1Hz resolution time series, there is a dramatic change, so that the time series is divided into two stages. But within each stage, the CPU load is relatively stable. The standard deviations are 0.0755 for the first part of the time series, 0.1522 for the second part of the time series.

pitcairn.mcs.anl.gov is a server machine at Argonne National Laboratory (ANL). The CPU load on this machine was relatively high during our measurement. All the measurements are more than 1, and the mean CPU load is greater than 1.2. The time series are relatively stable, with some occasional peaks. The standard deviations are 0.1355 for the 0.1 Hz time series.



(1)abyss.cs.uchicago.edu (2) vatos.cs.uchicago.edu

(3) mystere.ucsd.edu (4) pitcairn.mcs.anl.gov

Figure 3: 0.1 HZ CPU load time series collected from four machines.

We evaluated our time series prediction strategies on twelve CPU load time series. The prediction error was calculated by using Equation 5. The error rates and the standard deviations of the prediction strategies when tested against these time series are shown in Table **1**, with the best predictors shown in boldface

Table 1: The error of different prediction strategies, with the best in each case shown in boldface.

(1) Mean and standard deviation of the prediction errors on time series collected from abyss.cs.uchicago.edu

	0.1 Hz		0.05 Hz		0.025 Hz	
	Mean	SD	Mean	SD	Mean	SD
Independent Static Homeostatic	496.10%	4.2855	492.26%	4.3583	488.90%	4.4384
Independent Dynamic Homeostatic	12.50%	0.2369	25.51%	0.4153	56.70%	0.9756
Relative Static Homeostatic	13.40%	0.2115	24.85%	0.2771	44.37%	0.3960
Relative Dynamic Homeostatic	13.53%	0.2585	28.67%	0.6984	59.57%	1.5305
Independent Dynamic Tendency	11.42%	0.2097	21.45%	0.2742	40.44%	0.3949
Relative Dynamic Tendency	11.54%	0.2338	20.40%	0.3203	36.15%	0.4799
Mixed Tendency	**11.13%**	0.2094	**19.48%**	**0.2741**	**34.23%**	**0.3941**
Last Value	14.40%	**0.2068**	25.84%	0.2742	45.62%	0.3984
NWS	13.43%	0.2071	25.08%	0.2760	45.89%	0.4315

(2) Mean and standard deviation of the prediction errors on time series collected from vatos.cs.uchicago.edu

	0.1 Hz		0.05 Hz		0.025 Hz	
	Mean	SD	Mean	SD	Mean	SD
Independent Static Homeostatic	333.75%	4.0129	340.31%	4.0151	360.14%	3.9996
Independent Dynamic Homeostatic	12.76%	0.2067	26.19%	0.3531	66.62%	1.0480
Relative Static Homeostatic	16.46%	0.1929	30.16%	0.2561	57.52%	0.3906
Relative Dynamic Homeostatic	15.48%	0.4531	33.73%	0.8334	102.55%	3.5787
Independent Dynamic Tendency	12.38%	0.1926	22.78%	0.2583	43.16%	0.3699
Relative Dynamic Tendency	11.77%	0.2722	20.25%	0.3735	36.85%	0.5569
Mixed Tendency	**10.78%**	0.1947	**18.74%**	0.2607	**34.31%**	**0.3628**
Last Value	16.50%	**0.1879**	29.40%	**0.2510**	57.14%	0.3874
NWS	15.53%	0.1883	25.00%	0.2515	57.33%	0.3913

(3) Mean and standard deviation of the prediction errors on time series collected from mystere.ucsd.edu

	0.1 Hz		0.05 Hz		0.025 Hz	
	Mean	SD	Mean	SD	Mean	SD
Independent Static Homeostatic	158.09%	1.9350	167.71%	1.9891	185.06%	2.1680
Independent Dynamic Homeostatic	21.24%	0.2655	38.47%	0.3867	70.20%	0.5989
Relative Static Homeostatic	22.21%	**0.1929**	37.94%	0.2329	63.09%	0.3731
Relative Dynamic Homeostatic	43.81%	1.5344	85.09%	2.2558	156.26%	4.3681
Independent Dynamic Tendency	18.38%	0.2097	34.96%	0.2632	62.10%	0.4109
Relative Dynamic Tendency	29.01%	0.8312	55.81%	1.2062	103.45%	2.0504
Mixed Tendency	**17.31%**	0.2639	**32.21%**	0.3773	**55.81%**	0.5749
Last Value	19.86%	0.2045	35.56%	**0.2270**	99.47%	**0.3445**
NWS	18.88%	0.1945	34.92%	0.2288	96.96%	1.4816

(4) Mean and standard deviation of the prediction errors on time series collected from pitcairn.mcs.anl.gov

	0.1 Hz		0.05 Hz		0.025 Hz	
	Mean	SD	Mean	SD	Mean	SD
Independent Static Homeostatic	6.94%	0.0352	6.29%	0.0425	7.83%	0.0482
Independent Dynamic Homeostatic	2.54%	0.0262	4.23%	0.0407	7.70%	0.0568
Relative Static Homeostatic	2.73%	0.0248	4.45%	**0.0364**	7.17%	0.0462
Relative Dynamic Homeostatic	2.68%	0.0242	4.48%	0.0371	7.29%	0.0515
Independent Dynamic Tendency	2.43%	0.0239	4.11%	0.0365	**7.07%**	0.0476
Relative Dynamic Tendency	**2.29%**	**0.0237**	**3.91%**	0.0409	7.39%	0.0575
Mixed Tendency	**2.29%**	**0.0237**	**3.91%**	0.0409	7.38%	0.0574
Last Value	2.69%	0.0242	4.46%	**0.0364**	7.24%	**0.0473**
NWS	2.69%	0.0242	4.49%	0.0365	7.47%	0.0479

From the experimental results we observe that all the prediction strategies gave less accurate prediction on average for the traces with lower frequency. We attribute this result to (a) data points being more widely spaced in time, so the last data points are not as "current" as the traces where there is more data, and (b) the prediction point being farther in the future. We also see that the independent static homeostatic strategy, without any dynamic adjustment, always gives worse results than the other strategies. Tendency prediction strategies outperform other prediction strategies almost in all cases. Moreover, the strategy using mixed variation gives better performance on average than the other two tendency prediction strategies for time series collected from different machines; specifically, it gives an average error rate of 21.64% for time series collected from *abyss*, 21.28% for time series collected from *vatos*, 35.11% for time series collected from *mystere*, and 4.53% for time series collected from *pitcairn*. It also achieves the smallest or near smallest standard deviation of prediction error on twelve time series.

From the experimental results we also see that tendency prediction with the mixed variation method outperforms the NWS predictor on all time series with different frequency. Our prediction strategy achieves a prediction error that is 20.68% less than that of the NWS predictor on average. We also observe that, for time series with lower frequency (and thus less information), the benefits gained by using our prediction strategy rather than the NWS predictor are as significant as those for time series with similar properties but higher frequency. In particular, our prediction strategy outperforms the NWS predictor by 17.73% on average on four time series with 0.1 Hz frequency, by 17.01% on average on four time series with

0.05 Hz frequency, and by 27.3% on average on four time series with 0.025 Hz frequency.

4.3 Varied time-series comparison

To show that our mixed tendency prediction strategy performs better than NWS in the context of CPU load prediction, we compared the techniques on an much larger set of CPU load time series collected by Dinda [12].

These week-long, 1 Hz resolution time series (available at www.cs.nwu.edu/~pdinda/LoadTraces) represent 38 different machines, including production and research cluster machines, computer servers, and desktop workstations. The series exhibit very high standard deviation and maxima. The standard deviation is typically at least as large as the mean, while the maxima are as much as two orders of magnitude larger. The time series have complex, rough, and often multimodal distributions that are not well fitted by analytic distributions such as the normal or exponential distributions. All of the time series exhibit a high degree of self-similarity and epochal behavior. Detailed statistical properties of these CPU load time series can be found in [12].

For our experiments, we selected 38 one-day time series collected on August 18, 1997. The prediction errors of our prediction strategy and NWS strategies on this set of time series are shown in Figure 4.

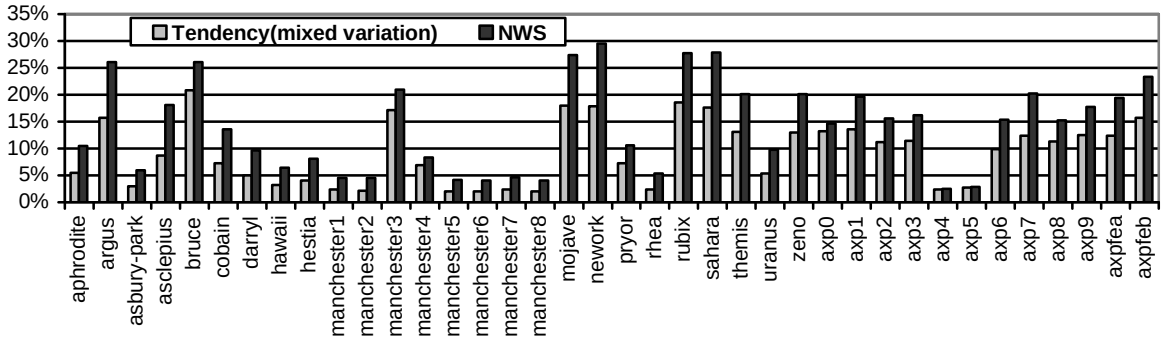


Figure 4: Prediction error of our predictor and NWS predictor on 38 time series collected from a variety of machines and with different statistical properties.

The experimental results show that the mixed tendency prediction strategy outperforms the NWS predictors on all of the 38 time series with different properties. It achieves a prediction error that is 36% lower on average than that achieved by NWS.

5 Conclusion and future work

Applications and schedulers each can benefit from accurate predictions of future resource availability when making decisions concerning how to use time-shared resources. In this paper, we have presented and evaluated two families of novel one-step-ahead time series prediction strategies that weight recent data in various ways. We presented experimental results that allow us to identify one such strategy as the best of the two families and to demonstrate that this strategy outperforms the widely used NWS predictor by 36% on average. While not every prediction is better, performance is clearly better on average. Comparison of prediction results on over 50 CPU load time series demonstrated that giving more weight to the most recent values significantly affects prediction accuracy.

Although our prediction strategy has been described and evaluated in the context of CPU load, we expect that it will also prove effective in other contexts. We plan to extend its use to network bandwidth and latency predictions. In addition, we are using this information to guide a scheduler designed to make efficient data placement approaches on a wide area network, where good predictions determine how well an application can run.

Acknowledgments

We thank Peter Dinda for the use of his set of time series. Thanks also to our colleagues within the GrADS project for providing access to testbed resources. This work was supported in part by the Grid Application Development Software (GrADS) project of the NSF Next Generation Software program, under Grant No. 9975020, and in part by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy, under contract W-31-109-Eng-38.

References

[1] I. Foster and C. Kesselman, "The Grid: Blueprint for a New Computing Infrastructure," Morgan Kufmann, 1998, pp. 701.

[2] M. Faerman, A. Su, R. Wolski, and F. Berman, "Adaptive Performance Prediction for Distributed Data-Intensive Applications," presented at Proceedings of the ACM/IEEE SC99

Conference on High Performance Networking and Computing, Portland, OR, 1999.
[3] S. V. Adve, A. F. Harris, C. J. Hughes, D. L. Jones, R. H. Kravets, K. Nahrstedt, D. G. Sachs, R. Sasanka, J. Srinivasan, and W. Yuan, "The Illinois GRACE Project: Global Resource Adaptation through Cooperation," presented at in the Proceedings of the Workshop on Self-Healing, Adaptive, and self-MANaged Systems (SHAMAN), 2002.
[4] C. Liu, L. Yang, I. Foster, and D. Angulo, "Design and Evaluation of a Resource Selection Framework for Grid Applications," presented at Proceedings of the 11th IEEE International Symposium on High-Performance Distributed Computing (HPDC 11), Edinburgh, Scotland, 2002.
[5] H. Dail, G. Obertelli, F. Berman, R. Wolski, and A. Grimshaw, "Application-Aware Scheduling of a Magnetohydrodynamics Application in the Legion Metasystem," presented at Proceedings of the 9th Heterogeneous Computing Workshop, 2000.
[6] P. A. Dinda, "A Prediction-based Real-time Scheduling Advisor," presented at Proceedings of the 16th International Parallel and Distributed Processing Symposium (IPDPS 2002), 2002.
[7] P. A. Dinda and D. R. O'Hallaron, "An Evaluation of Linear Models for Host Load Prediction," presented at Proceedings of the 8th IEEE International Symposium on High-Performance Distributed Computing (HPDC-8), Redondo Beach, CA, 1999.
[8] R. Wolski, "Dynamically Forecasting Network Performance Using the Network Weather Service," *Journal of Cluster Computing*, 1998.
[9] R. Wolski, N. Spring, and J. Hayes, "Predicting the CPU availability of Time-shared Unix Systems," presented at Proceedings of 8th IEEE High Performance Distributed Computing Conference (HPDC8), 1999.
[10] R. Wolski, N. Spring, and J. Hayes, "The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing," *Journal of Future Generation Computing Systems*, pp. 757-768, 1998.
[11] R. Wolski, N. Spring, and C. Peterson, "Implementing a Performance Forecasting System for Metacomputing: The Network Weather Service," presented at Proceedings of SC97, 1998.
[12] P. A. Dinda and D. R. O'Hallaron, "The Statistical Properties of Host Load," presented at Fourth Workshop on Languages, Compilers, and Run-time Systems for Scalable Computers (LCR 98), Pittsburgh, PA, 1998.
[13] F. Virili and B. Freisleben, "Neutral Network Model Selection for Financial Time Series Prediction," *Computational Statistics*, vol. 16, pp. 451-463, 2001.
[14] A. Lendasse, E. d. Bodt, V. Wertz, and M. Verleysen, "Non-linear Financial Time Series Forecasting-Application to the Bel 20 Stock Market Index," *European Journal of Economic and Social Systems*, vol. 14, pp. 81-91, 2000.
[15] P. d. Almeida and L. Torgo, "The Use of Domain Knowledge in Feature Construction for Financial Time Series Prediction," presented at Portuguese Conference on Artificial Intelligence(EPIA 2001), Porto, Portugal, 2001.
[16] L. M. Lawson, E. E. Hofmann, and Y. H. Spitz, "Time Series Sampling and Data Assimilation in a Simple Marine Ecosystems Model," in *Deep Sea Research*, vol. 43, 1996, pp. 625-651.
[17] H. Kato and H. Kawahara, "An Application of the Bayesian Time Series Model and Statistical System Analysis for F0 Control," *Speech Communication*, vol. 24, pp. 325-339, 1998.
[18] N. K. Groschwitz and G. C. Polyzos, "A Time Series Model of Long-Term NSFNET Backbone Traffic," presented at Proceedings of the IEEE International Conference on Communications(ICC'94), 1994.
[19] H.-W. Braun, B. Chinoy, K. C. Claffy, and G. C. Polyzos, "Analysis and Modeling of High Speed Networks: 1993 Annual Status Report," San Diego Supercomputer Center and University of Californina, San Diego CS92-237, 1993.
[20] M. Harchol-Balter and A. Downey, "Exploiting Process Lifetime Distributions for Dynamic Load Balancing," presented at Proceedings of ACM Sigmetrics' 96 Conference on Measurement and Modeling of Computer Systems, Philadelphia, PA, 1996.