IEICE TRANS. INF. & SYST., VOL.E90–D, NO.1 JANUARY 2007

PAPER Special Section on Parallel/Distributed Processing and Systems

CPU Load Predictions on the Computational Grid**

Yuanyuan ZHANG[†*a)], **Wei SUN**[†], *Nonmembers*, *and* **Yasushi INOGUCHI**[††], *Member*

SUMMARY To make the best use of the resources in a shared grid environment, an application scheduler must make a prediction of available performance on each resource. In this paper, we examine the problem of predicting available CPU performance in time-shared grid system. We present and evaluate a new and innovative method to predict the one-step-ahead CPU load in a grid. Our prediction strategy forecasts the future CPU load based on the variety tendency in several past steps and in previous similar patterns, and uses a polynomial fitting method. Our experimental results on large load traces collected from four different kinds of machines demonstrate that this new prediction strategy achieves average prediction errors which are between 22% and 86% less than those incurred by four previous methods.

key words: *grid computing, task scheduling, CPU load prediction, polynomial fitting*

1. Introduction

The term computational grid [1], a hotspot in recent research field, derives its name from the analogy with the electric power grid from which we make use of the electricity without the knowledge of where it is from or how it is generated. A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to heterogeneous and dynamic resources. Task scheduling of the applications is an important component for achieving high performance in a grid environment, while most of such work involves predicting the performance of the tasks on the prospective resources.

CPU load on a host significantly influence the running time of computation-intensive tasks. In fact, for some applications the running time of a computational task is almost linearly related to the measured load during its execution [2]. If we could predict the load during the execution of a task on a host, we could predict easily the running time of the task on the host. Therefore, CPU load prediction can be useful for guiding job scheduling strategies to achieve high application performance and efficient resource use [3]–[5]. However, in grid since users with competing goal share resources, resources are upgraded, fail, and so on, host load and availability vary over time. Such dynamicity makes the load prediction problem more difficult. In this paper we propose a new one-step-ahead time series prediction strategy which behaves better than previously proposed techniques. Such one-step-ahead load prediction can be extended to multi-steps-ahead prediction, and then task running times can be predicted based on such CPU load predictions, so one-step-ahead load prediction is the basis for the task running time prediction, and in turn very important for the performance of applications and grid schedulers. We have achieved some preliminary results for using the work in this paper to solve the task running time prediction problem, which can be found in [6].

Our one-step-ahead CPU load prediction strategy is a kind of tendency-based method, which predicts the one-step-ahead load value based on the "ascending" or "descending" tendency of load signal several steps backward, and uses 2nd or 3rd order polynomial fitting to determine the prediction value. We also predict based on a search of previous similar "patterns". Our experimental results on a commonly used host load measurement dataset show that our proposed strategy consistently outperforms the previous Last Measurement predictor, Tendency-based predictor, AR linear models, and the Network Weather Service (NWS) system [7].

The paper structure is as follows. In Sect. 2, we discuss related work. In Sect. 3, we introduce our prediction strategy in detail. Section 4 describes the results of experiments in which our prediction method was compared with previous work. Finally in Sect. 5 we conclude the paper.

2. Related Work

In [8] Dinda presented a comprehensive analysis on the statistical properties of host load. The significant result is that although load exhibits complex properties, it is still consistently predictable to a very useful degree from past behavior. This work encourages us that there is abundant opportunity for prediction algorithms to improve matters.

Several CPU load prediction strategies have been proposed [9], [10]. Among them, NWS is a system that dynamically forecasts the performance of network and computational resources. For every resource, NWS uses several methods to forecast simultaneously. The overall best performed method at time t is then used to predict the load at

Manuscript received March 1, 2006.
Manuscript revised June 30, 2006.

[†]The authors are with the Graduate School of Information Science, JAIST, Nomi-shi, 923–1292 Japan.
[††]The author is with the Center for Information Science, JAIST and PREST, Japan Science and Technology Agency, Nomi-shi, 923–1292 Japan.
[*]The autor has moved to Fujitsu Laboratry Ltd.

[**]This research is conducted as a program for the "21st Century COE Program" by Ministry of Education, Culture, Sports, Science and Technology, Japan.

a) E-mail: yuanyuan@jaist.ac.jp

DOI: 10.1093/ietisy/e90–d.1.40

time t+1. The predictive methods currently used in NWS include running average, sliding window average, last measurement, adaptive window average, median filter, adaptive window median, α-trimmed mean, stochastic gradient, and auto-regression (AR).

Dinda et al. [2] evaluated different linear time series models [11], including AR, MA, ARMA, ARIMA, and ARFIMA, for predicting load from 1 to 30 seconds into the future. His conclusion is that considering both computational overhead and prediction accuracy, simple, practical models such as AR models with the model order higher than 16 are sufficient for host load prediction.

Yang et al. [12] proposed several homeostatic and tendency-based one-step-ahead prediction strategies. The idea of homeostatic methods is that the mean of a load series remains steady, while tendency-based strategies assume that a load series always retains its "tendency", that is, if current value increases, the next value will also increase, or vice versa. The increment/decrement used in the methods is adjusted according to the values of the last load measurement and the last prediction error.

In this paper we propose and evaluate a new one-step-ahead load prediction strategy. Our strategy is tendency-based, which predicts based on the variety tendency of load signal several steps backward, and uses polynomial fitting to determine the prediction value. We also predict based on the measurements in previous similar "patterns".

3. CPU Load Prediction

Based on our study on the statistical properties of host load traces, our prediction strategy predicts the next load value based on polynomial fitting and "similar" patterns.

We use the following notations in the description of the prediction strategy:

V_T: the measured load at the Tth measurement.

P_{T+1}: predicted value for the (T+1)th measurement.

N: number of historical data points used for the prediction of P_{T+1}.

3.1 Prediction Based on Polynomial Fitting

Polynomial models are a great tool for many engineering and manufacturing applications, hence polynomial fitting is sometimes used to estimate and predict the future data and their direction [13]. In most modern data analysis tools for example Matlab and Origin, the polynomial fitting is a basic function. In the modern computer science polynomial fitting has become a common approach for signal processing and data analysis [14].

Similar to some methods in [12], our prediction strategy is a kind of tendency-based method because we predict based on the increases or decreases of previous measurements. However, our method to predict the increment or decrement for next step is based on multi-order polynomial fitting, while the tendency-based method used in [12] is somewhat a 1st order polynomial fitting method, i.e., linear fitting. It is obvious that such linear fitting is not so consistent with reality, so we try the multi-order polynomial fitting method. To determine the order of the polynomial function, we have studied the rule that the CPU load traces vary and run a set of experiments. From our observations we found that 2nd or 3rd order polynomial fitting achieves much better (and the best) fitting effect than the 1st order (linear) fitting does when a load trace varies smoothly and monotonously; therefore in our prediction we try both 2nd and 3rd order polynomial fitting and choose the one that is closer to the current measurement as the predicted value for next load measurement. The detail is that when the last 3 (V_{T-2}, V_{T-1}, V_T) or 4 (V_{T-3}, V_{T-2}, V_{T-1}, V_T) load measurements increase or decrease monotonously, we use these measurements to produce a 2nd or 3rd order polynomial fitting function and then use the function value at the next time point as P_{T+1}.

3.2 Prediction Based on Similar Patterns

The polynomial fitting method can provide satisfying prediction when load traces vary smoothly and monotonously, however, such fitting does not work well for the prediction of a "turning point", the time when a load series changes its "direction", (that is, the point at which a load series begins to decrease after a number of increases, or starts to increase after some successive decreases), or when a "turning point" is used as one data point to fit the polynomial function. For such turning points, the tendency-based prediction method in [12] uses the average of all the past measurements as the threshold to judge if the point to be predicted will be a turning point or not; if yes, it adjusts the increment according to the magnitude of the last measurement. However, if the current point is a turning point, for example, if the load trace begins to decrease at this point after several increases, according to the tendency-based method the predicted value will still increase, although the real measurement decreases. Therefore such a prediction strategy introduces considerable prediction error.

To deal with this problem, we predict the load value at a possible turning point based on the information of previous similar "patterns". Here a "pattern" is the successive increases or decreases between two neighboring turning points. Two patterns with same number of data points and same tendency are thought to be "similar". We observe that there are many successive and similar "patterns" occurring in the load series, as shown in Fig. 1. Because of this observation, we can predict the value of a possible turning point based on the information of previous similar patterns. In detail, if the point to be predicted comes after several (at least 3) successive decreases (the first point of these successive decreases is a turning point), while there are same number of successive decreases before this series of decreasing points, then we think that the point to be predicted is quite possibly a turning point and these two decreasing series are similar "patterns", so we use the value of last turning point as the prediction value, P_{T+1}. If we can't find such similar patterns we will still use the polynomial fitting method to

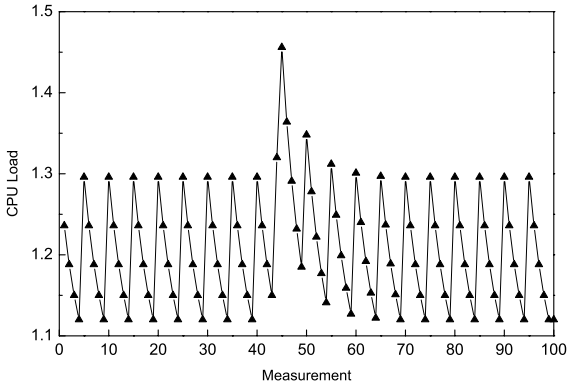


Fig. 1 Similar "patterns" in the load traces.

predict.

To predict a point one or two steps after a turning point, a polynomial fitting based prediction will also result in a large prediction error because such prediction uses the turning point, so we predict such points also based on similar patterns: we use the increment for the point in same position in last pattern as the increment for current point if last two patterns are similar.

To predict a point after several (at least 3) successive increases, if we can find a successively increasing pattern immediately before these increasing points, we will use the information of this pattern to judge if current point is a turning point or not and then predict: if the number of points in this pattern has been same as that of last pattern, then we think that next point will be a turning point and use the value of last turning point as P_{T+1}; if the number is less than that of last pattern, then we use polynomial fitting to predict next value; if we can't find a successively increasing series we will choose a "conservative" strategy: we use V_T as P_{T+1}.

For the other cases we also choose a conservative prediction strategy that sets the increment/decrement between V_T and P_{T+1} at 0.

3.3 Analysis of Time Complexity

It is obvious that the time complexity of our proposed prediction algorithm mainly depends on polynomial fitting, while since we only use 3 or 4 load measurements to fit a 2nd or 3rd order polynomial fitting function, the computational cost is very small. This complexity is comparative with Last Measurement and Tendency-based method, and much lower than that of AR linear models and NWS.

4. Experimental Evaluations

4.1 Experimental Environment

To validate if our proposed prediction strategy performs better than the other methods or not in the context of CPU load prediction, we ran a series of experiments to compare our method with Last Measurement, Tendency-based, AR models, and NWS on a large set of CPU load time series with a variety of statistical properties. The load traces we used are the load time series set on a Unix system and were collected by Dinda [15]. The load here refers to the number of processes that are running or ready to run. The kernel samples the number at some period and exponentially averages some previous samples to produce a load average. To satisfy the Nyquist criterion, Dinda chose a 1 HZ sample rate and exponentially averaged with a time constant of five seconds.

In the first and second part of this section, we evaluate these five time series prediction strategies on CPU load traces collected from four heterogeneous machines. These four machines demonstrate different CPU load statistical properties, as illustrated in Fig. 2. There are two groups of machines. The first is in an Alpha cluster. In this group, the machine *axp0* is an interactive machine, while *axp7* is a batch machine. The machines in the second group are a compute server named *sahara* and a desktop workstation named *themis*.

axp0.psc.edu is a heavily loaded, highly variable interactive machine with mean CPU load 1 and standard deviation 0.54. The total number of data in this time series is 1,296,000 (for 75 days).

axp7.psc.edu is a more lightly loaded batch machine which reveals interesting epochal behavior. The average of CPU load is only 0.12 and its standard deviation is 0.14. The total number of data is 1,123,200 (for 65 days).

sahara.cmcl.cs.cmu.edu is a moderately loaded (mean load 0.22), big memory compute server. The standard deviation of the CPU load is 0.33. The total number of data is 345,600 (for 20 days).

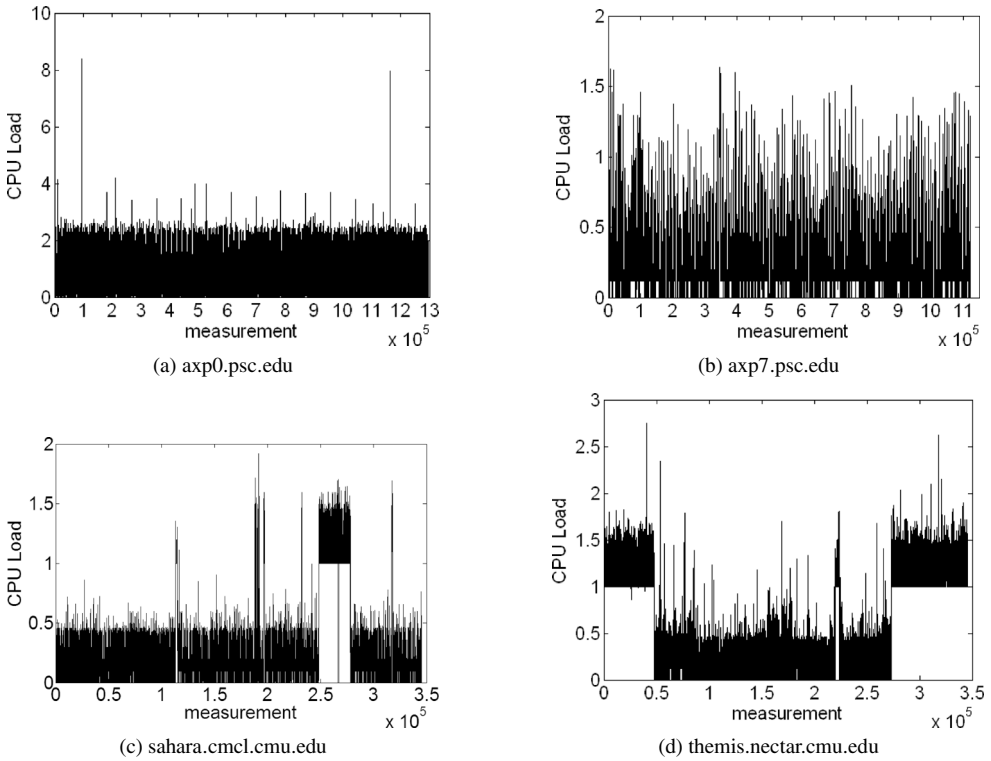
themis.nectar.cs.cmu.edu is a moderately loaded (mean load 0.49) desktop machine. The load on this machine has high standard deviation of 0.5. The total number of data is 345,600 (for 20 days).

4.2 Evaluation Results for AR Models

For the data set on each machine we want to compare our proposed method with AR linear models using different numbers of data from 1 day to the total number of the data to evaluate its effectiveness. We choose AR model other than other linear or nonlinear time series models is because that as Dinda has mentioned, AR models can achieve considerably accurate prediction with much lower computational cost comparing with other models.

For an AR model, if the current value of a time series is only related to the last p values, then the model is called a p-order model, or AR(p) model. Since the order is the most important parameter for an AR model, first of all we want to know which order we should choose for prediction with an AR model considering both the prediction accuracy and the computational overhead. To do this, we conducted prediction experiments on the load traces collected from the above four machines with the model order of AR models increasing from 1 to 20 to try to disclose the relationship between AR model order and the prediction accuracy.

The result is shown in Fig. 3. Here the prediction error



(a) axp0.psc.edu (b) axp7.psc.edu

(c) sahara.cmcl.cmu.edu (d) themis.nectar.cmu.edu

Fig. 2 CPU load time series collected from four machines.

for a measurement is the ratio between the absolute value of prediction error (the difference between a predicted value and the measurement) and the measurement. The mean of prediction errors is the average error ratio for the prediction error ratios of all the data in a time series. From Fig. 3 we can see that the prediction error doesn't decrease monotonously as the increase of the order of AR models. This means that a higher order AR model does not always correspond to more accurate prediction. This is an interesting phenomenon which is beyond our expectation.

What should also be noticed is that the curves which represent the prediction errors with different numbers of data on each of the four machines are very similar, that is, the orders of AR models corresponding to the best predictions with different numbers of data on each machine are same, although the prediction curves on different machines vary significantly. For example, on machine axp7, traces with data of 1 day, 5 days, 10 days, 15 days, 20 days and 65 days achieve minimum prediction error all using AR(6) model. On machine axp0, although the best AR models for traces with different numbers of data are different, the differences of prediction errors between the best AR models with different numbers of data are negligible. For example, trace with data of 1 day on axp0 achieves best prediction with AR(17) model, while the best order of AR model for trace with data of 5 days on same machine is 7, but the prediction errors for AR(7) and AR(17) models with either data of 1 day or 5 days are almost same. This is because although the numbers of data for traces on a machine are different, these traces reveal similar statistical characteristics; therefore the best AR models for them are same. This hints us that if we use AR model for CPU load prediction, we can use only a small part of the load trace history as training data to find the best AR model for this trace, and then use the model to predict the future CPU load.

4.3 Comparison of Five Prediction Strategies

The evaluation results, using our prediction strategy to predict the CPU load on the above four machines, are shown in Table 1. Here the standard deviation (SD) of prediction errors is the standard deviation for the prediction error ratios of all the data in a time series. We conduct a comparison for our proposed method with Last Measurement, Tendency-based, AR models, and NWS.

We try AR models with the model order increases from 1 to 20. For AR model in Table 1, the data in the bracket following the prediction error is the order of AR model which achieves best prediction for AR models from AR(1) to AR(20).

For Last Measurement, AR models and NWS, since NWS uses the best performed method among methods including Last Measurement and AR models to predict next value, its performance usually should be no worse than that of the other two. Our evaluation results shown in Table 1 validate this. For the Tendency-based method and NWS, Table 1 shows that usually NWS achieves better performance compared with the Tendency-based strategy. While for Last Measurement, Tendency-based and AR models, there is no strict ordering relationship among them. Notice that the pre-

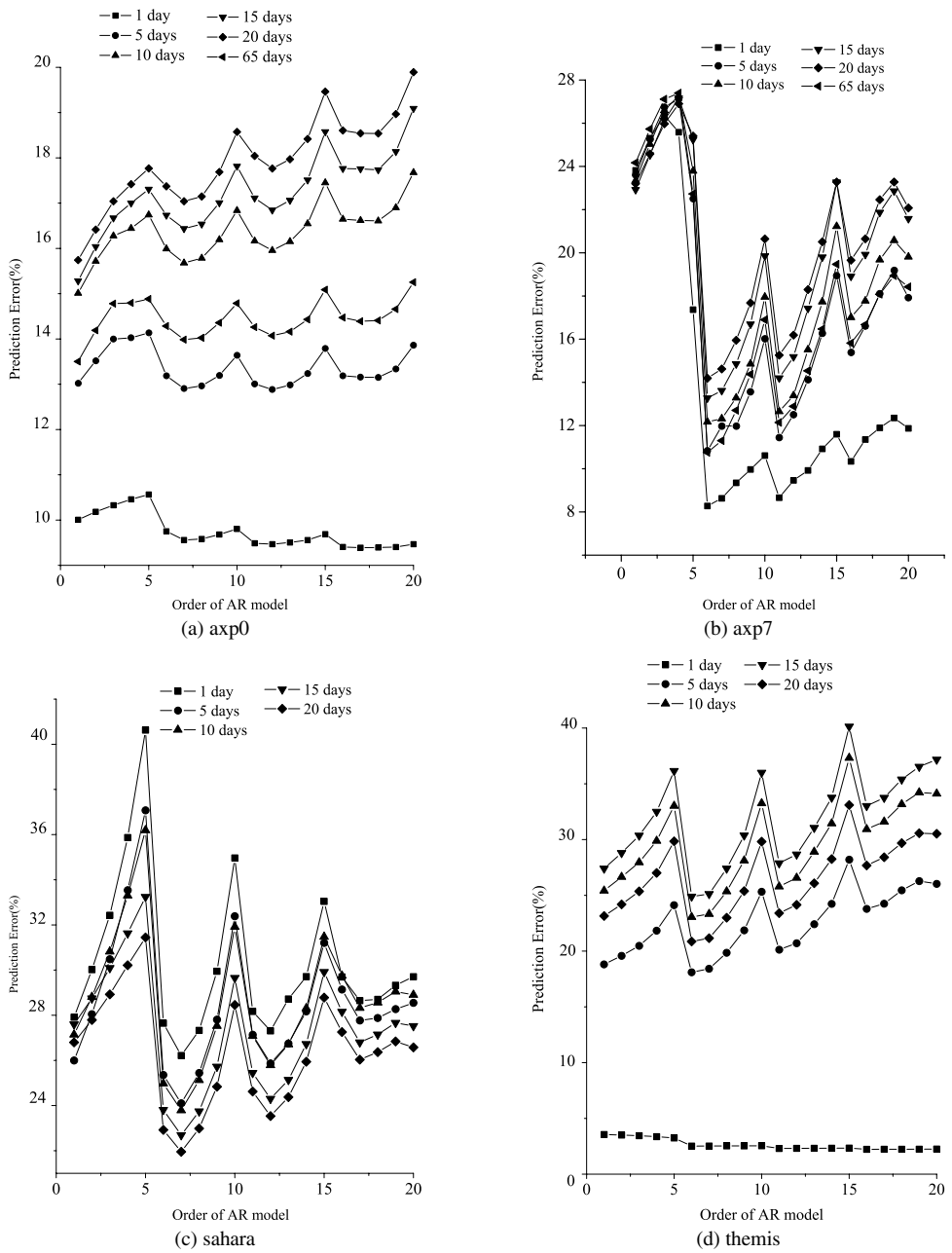


Fig. 3 Prediction errors for the AR models from AR(1) to AR(20) on the four machines.

diction result for AR model we show here is actually the best result for all the AR models from AR(1) to AR(20), therefore the overhead of AR models is much higher than that of the other two methods.

For the data set on each machine we evaluate the five strategies using different numbers of data from 1 day to the total number of the data. The results show that the prediction error has no direct relationship with the amount of data, but from the experimental results we can see that in all cases our prediction strategy performs quite better than the other four methods for all of the four load traces which have different load properties. Specifically, for the total number of data on the four machines, our proposed method gives an average prediction error rate of 10.57% on axp0 which is 22% less than that incurred by NWS and the best AR model between AR(1) and AR(20), 44% less than that of Tendency-based method, and 39% less than that of Last Measurement method; 2.73% for load series collected from axp7 which is 75% less than that incurred by NWS and the best AR model, 82% less than that of Tendency-based method, and 86% less than that of Last Measurement method; 9.2% for load series collected from sahara which is 47% less than the error rate of NWS, 58% less than that of the best AR model, 50% less than that of the Tendency-based method, and 56% less than that of Last Measurement; and 10.6% for load series collected from themis which is 35% less than that of

Table 1 Mean and SD of prediction errors of different prediction strategies.

(1) Mean and SD of the prediction errors on time series collected from *axp0.psc.edu*.

axp0	1 day		5 days		10 days		15 days		20 days		75 days	
	Mean (%)	SD	Mean (%)	SD	Mean (%)	SD	Mean (%)	SD	Mean (%)	SD	Mean (%)	SD
Last Measurement	10.09	0.41	13.08	0.48	15.02	0.44	15.26	0.48	15.71	0.46	17.25	0.50
Tendency-based	10.58	0.43	13.73	0.54	15.93	0.40	16.32	0.47	16.80	0.48	18.99	0.44
AR Model (order of model)	9.39 (17)	0.37 (17)	12.90 (7)	0.45 (7)	15.00 (1)	0.43 (1)	15.20 (1)	0.47 (1)	15.74 (1)	0.35 (1)	13.50 (1)	0.47 (1)
NWS	9.39	0.37	12.9	0.45	13.47	0.41	14.73	0.44	15.2	0.41	13.5	0.47
Our method	6.54	0.20	7.86	0.22	8.31	0.33	9.33	0.28	10.10	0.39	10.57	0.37

(2) Mean and SD of the prediction errors on time series collected from *axp7.psc.edu*.

axp7	1 day		5 days		10 days		15 days		20 days		65 days	
	Mean (%)	SD	Mean (%)	SD	Mean (%)	SD	Mean (%)	SD	Mean (%)	SD	Mean (%)	SD
Last Measurement	24.44	0.27	22.37	0.24	21.32	0.37	19.18	0.34	17.56	0.36	18.97	0.33
Tendency-based	20.45	0.22	18.00	0.26	17.02	0.39	15.34	0.29	20.45	0.32	15.02	0.38
AR Model (order of model)	8.28 (6)	0.19 (6)	10.83 (6)	0.17 (6)	12.17 (6)	0.29 (6)	13.27 (6)	0.20 (6)	14.19 (6)	0.35 (6)	10.76 (6)	0.25 (6)
NWS	8.28	0.19	10.83	0.17	11.37	0.21	13.27	0.20	14.19	0.35	10.76	0.25
Our method	3.14	0.16	3.06	0.23	2.92	0.13	2.83	0.07	2.81	0.08	2.73	0.08

(3) Mean and SD of the prediction errors on time series collected from *sahara.cmcl.cs.cmu.edu*.

sahara	1 day		5 days		10 days		15 days		20 days	
	Mean (%)	SD	Mean (%)	SD	Mean (%)	SD	Mean (%)	SD	Mean (%)	SD
Last Measurement	27.39	0.41	24.8	0.39	24.09	0.34	22.14	0.35	20.99	0.31
Tendency-based	23.59	0.33	20.90	0.35	20.20	0.26	18.74	0.34	18.28	0.37
AR Model (order of model)	26.22 (7)	0.36 (7)	24.10 (7)	0.47 (7)	23.79 (7)	0.23 (7)	22.69 (7)	0.39 (7)	21.95 (7)	0.42 (7)
NWS	13.52	0.31	15.14	0.31	16.32	0.19	18.15	0.26	17.3	0.29
Our method	7.54	0.25	9.86	0.31	10.31	0.12	11.33	0.21	9.20	0.23

(4) Mean and SD of the prediction errors on time series collected from *themis.nectar.cs.cmu.edu*.

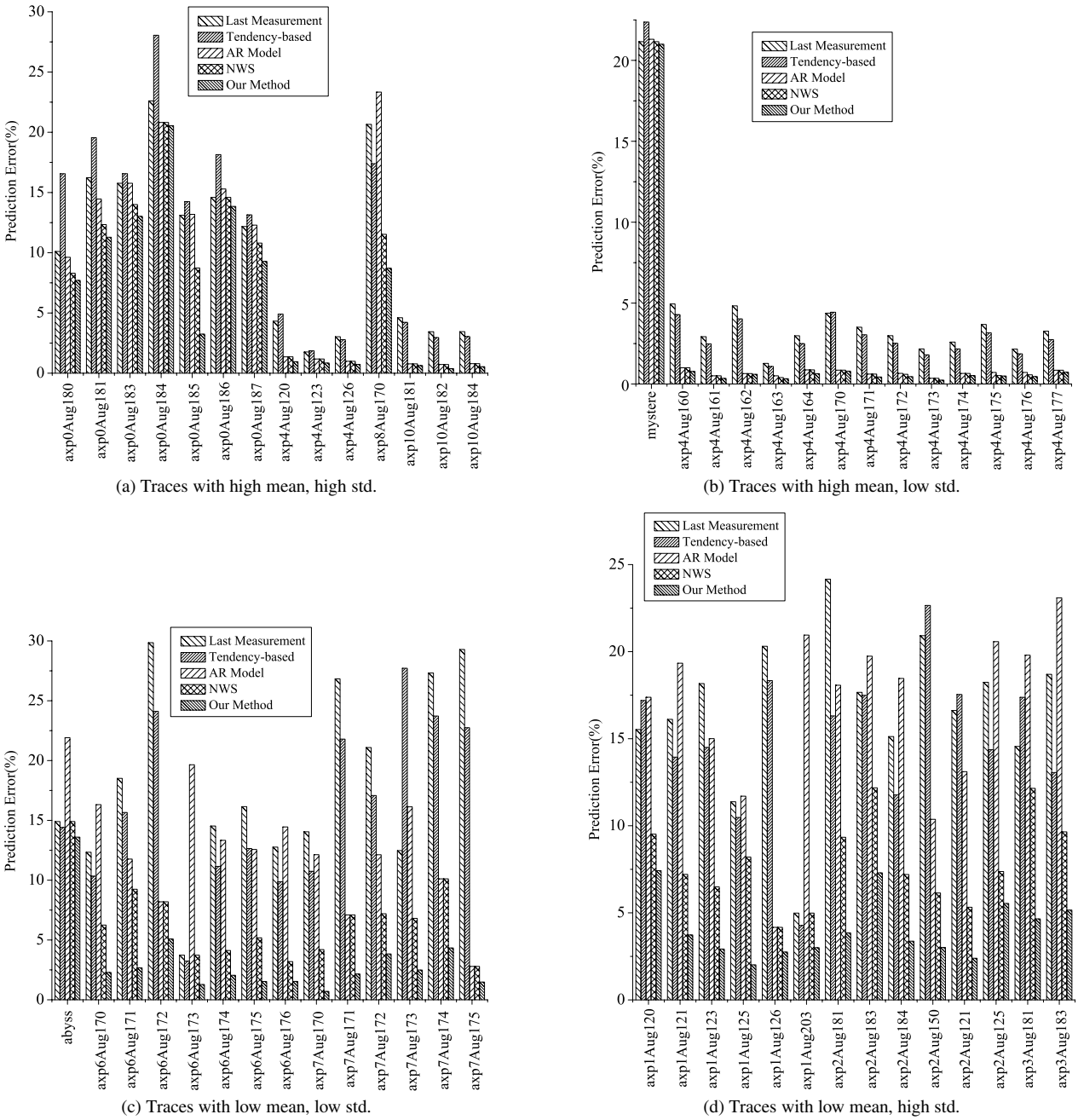
themis	1 day		5 days		10 days		15 days		20 days	
	Mean (%)	SD	Mean (%)	SD	Mean (%)	SD	Mean (%)	SD	Mean (%)	SD
Last Measurement	3.63	0.07	17.73	0.60	22.51	0.39	24.06	0.37	20.33	0.34
Tendency-based	3.13	0.05	20.30	0.55	25.85	0.34	27.48	0.51	27.37	0.48
AR Model (order of model)	2.23 (6)	0.05 (6)	18.10 (6)	0.48 (6)	23.06 (6)	0.30 (6)	24.89 (6)	0.41 (6)	20.86 (6)	0.40 (6)
NWS	2.23	0.05	14.25	0.46	15.42	0.24	17.11	0.21	16.21	0.35
Our method	1.97	0.05	8.93	0.40	9.51	0.04	11.39	0.14	10.60	0.25

NWS, 49% less than that of the best AR model, 61% less than that of the Tendency-based strategy, and 48% less than that of Last Measurement. Compared with other methods, our strategy is especially effective for predicting CPU load on machine axp7.

4.4 Varied Load Series Comparisons

Dinda provides traces on 38 machines in his homepage. We hope to conduct a complete comparison between our method and the previous strategies on all of these traces, but unfortunately the other traces are not available currently, therefore to further testify the effect of our prediction strategy, we downloaded traces from the homepage of Yang [16]. The number of data in every trace is 10,000. We classify these traces into four groups according to the values of their means and standard deviations. The mean that is larger than 1 is called "high" mean; while one which is smaller than 0.5 is called a "low" mean. On the other hand, the standard deviation which is larger than 0.25 is called a "high" standard deviation, or in short high std.; while std. smaller than 0.2 is judged as "low" std.. We classify the traces into groups of "High mean" and "High std." (HH), "High mean" and "Low std." (HL), "Low mean" and "Low std." (LL), and "Low mean" and "High std." (LH). There are 14 traces in each group.

The experimental results for the above five strategies on these four groups of traces are shown in Fig. 4. The caption of horizontal axis in each figure represents the names of the load traces. Among Last Measurement, Tendency-based, AR models and NWS, we can see that NWS achieves the best prediction performance for all of the four groups, while there is no clear ordering relationship among the other three. For all of the traces, our prediction strategy performs better than all of the other four strategies. The reduction of prediction errors is especially significant in Fig. 4 (c) and Fig. 4 (d).



(a) Traces with high mean, high std.

(b) Traces with high mean, low std.

(c) Traces with low mean, low std.

(d) Traces with low mean, high std.

Fig. 4 Prediction results of different prediction strategies for traces with different statistical characteristics.

5. Conclusion

Prediction of future system performance is helpful and important for job scheduling and resource management in a dynamic, resource-sharing grid environment. In this paper we introduce and evaluate a one-step-ahead CPU load prediction strategy which predicts based on the variety tendency of a number of past steps and on previous similar patterns, using a polynomial fitting method. The experiment results we conducted on over 50 CPU load series demonstrate that this new prediction strategy outperforms the previously methods significantly. Specifically, for four large traces on four different types of machines, its average prediction errors are between 22% and 86% less than those incurred by four existing prediction strategies. Our future work is to extend the application of our prediction strategy to other resource signals.

References

[1] I. Foster and C. Kesselman, The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann Publishers, San Fransisco, CA, 1999.

[2] P.A. Dinda and D.R. O'Hallaron, "Host load prediction using linear

models," J. Cluster Computing, vol.3, no.4, pp.265–280, 2000.
[3] P.A. Dinda, "A prediction-based real-time scheduling advisor," Proc. 16th Int'l Parallel and Distributed Processing Symp. (IPDPS 2002), pp.1–8, 2002.
[4] S. Jang, X. Wu, and V. Taylor, "Using performance prediction to allocate grid resources," Technical report, GriPhyN 2004-25, pp.1–11, 2004.
[5] L. Yang, J.M. Schopf, and I. Foster, "Conservative scheduling: Using predicted variance to improve scheduling decisions in dynamic environment," Proc. ACM/IEEE SC2003 Conf. on High Performance Networking and Computing, pp.1–16, 2003.
[6] Y. Zhang, W. Sun, and Y. Inoguchi, "Predict running time of grid tasks based on CPU load predictions," The 7th IEEE/ACM Int'l Conf. on Grid Computing (Grid2006), pp.1–7, 2006.
[7] M. Swany and R. Wolski, "Multivariate resource performance forecasting in the network weather service," Proc. ACM/IEEE SC2002 Conf. on High Performance Networking and Computing, pp.1–10, 2002.
[8] P.A. Dinda, "The statistical properties of host load," Technical Report, CMU, pp.1–23, 1998.
[9] S. Akioka and Y. Muraoka, "Extended forecast of CPU and network load on computational grid," 2004 IEEE Int'l Symp. on Cluster Computing and the Grid, pp.765–772, 2004.
[10] J. Liang, K. Nahrstedt, and Y. Zhou, "Adaptive multi-resource prediction in distributed resource sharing environment," 2004 IEEE Int'l Symp. on Cluster Computing and the Grid, pp.1–8, 2004.
[11] G. Box, G. Jenkins, and G. Reinsel, Time Series Analysis, Forecasting and Control, Prentice Hall, 1994.
[12] L. Yang, I. Foster, and J.M. Schopf, "Homeostatic and tendency-based CPU load predictions," Proc. 17th Int'l Parallel and Distributed Processing Symp. (IPDPS 2003), pp.42–50, 2003.
[13] http://www.itl.nist.gov/div898/handbook/
[14] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, Introduction to Algorithms, Second ed., The MIT Press, 2001.
[15] http://www.cs.cmu.edu/˜pdinda/LoadTraces/
[16] http://people.cs.uchicago.edu/˜lyang/Load/

Yuanyuan Zhang received the B.E. degree in School of Mechano-Electronic Engineering, and M.E. degree in School of Computer Science and Technology from Xidian University, China, in 2000 and 2003, respectively. She received Ph.D. from Graduate School of Information Science, JAIST (Japan Advanced Institute of Science and Technology) in 2006. She is a researcher of Fujitsu Laboratory Ltd since 2006. Her current research interest is about resource management and information service in grid computing.

Wei Sun received his B.E. and M.E. degrees from Tianjin University, China, in 1998 and 2005. From 1998 to 2002 he was an engineer in the Sixth Research Institute (Electronics) of Ministry of Information Industry of China. He is currently a PhD candidate at JAIST. His research deals with large scale distributed system, parallel and heterogeneous computing.

Yasushi Inoguchi received his B.E. degree from Department of Mechanical Engineering, Tohoku University in 1991, and received M.S. degree and Ph.D from Japan Advanced Institute of Science and Technology (JAIST) in 1994 and 1997, respectively. He is currently a Associate Professor of Center for Information Science at JAIST. He was a research fellow of the Japan Society for the Promotion of Science from 1994 to 1997. He is also a researcher of PRESTO program of Japan Science and Technology Agency since 2002. His research interest has been mainly concerned with parallel computer architecture, interconnection networks, and high performance computing on parallel machines. Dr. Inoguchi is a members of IEEE and IPS of Japan.