Load Prediction Using Hybrid Model for Computational Grid

Yongwei Wu[1], Yulai Yuan[2], Guangwen Yang[3], Weimin Zheng[4]

Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China

[1, 3, 4]{wuyw, ygw, zwm-dcs}@tsinghua.edu.cn

[2]yuan-yl05@mails.tsinghua.edu.cn

***Abstract*—Due to the dynamic nature of grid environments, schedule algorithms always need assistance of a long-time-ahead load prediction to make decisions on how to use grid resources efficiently. In this paper, we present and evaluate a new hybrid model, which predicts the *n*-step-ahead load status by using interval values. This model integrates autoregressive (AR) model with confidence interval estimations to forecast the future load of a system. Meanwhile, two filtering technologies from signal processing field are also introduced into this model to eliminate data noise and enhance prediction accuracy. The results of experiments conducted on a real grid environment demonstrate that this new model is more capable of predicting *n*-step-ahead load in a computational grid than previous works. The proposed hybrid model performs well on prediction advance time for up to 50 minutes, with significant less prediction errors than conventional AR model. It also achieves an interval length acceptable for task scheduler.**

I. INTRODUCTION

Grid computing [1] is the high-performance and internet-based infrastructure that aggregates geographically distributed and diverse resources to deliver computational power to users in a transparent way, supporting large-scale, resource-intensive and distributed applications. Task scheduling is an important factor on improving resource usage efficiency in such a dynamic distributed environment. Obviously, system load prediction can be used to forecast task run time [18] and guide scheduling strategies, thus to achieve high performance and more efficient resource usage [2] [3] [4].

In most previous works, load prediction models, such as mean-based methods, median-based methods, autoregressive (AR) models, polynomial fitting, etc., use point value to represent future load status of workstations, clusters and grids. Papers [5] [6] [10] describe such point value prediction models and their performances. Some other works, like [11], implement interval value prediction on structural model to forecast system performance in a distributed production environment.

Though point value prediction strategy has been widely adapted by recent system load prediction models, this kind of strategy do have some drawbacks for highly distributed environments. Because grid tasks usually take long run time, task scheduler in a computational grid needs a comparatively long-time-ahead prediction or an *n*-step-ahead prediction with large step intervals. Point value can hardly cover load variability in such a long time frame (i.e., step interval), and therefore these conventional point value prediction models do not perform well on *n*-step-ahead load prediction for large *n*. The interval value prediction model proposed in [11] addresses these two problems of point value prediction and provides valuable additional information that can be used by a task scheduler. But it does not achieve a reasonable prediction interval length, which is important for the task scheduler.

Furthermore, most of the previous prediction works also ignore some problems in the history load data, which significantly affects the prediction accuracy. One problem is the load measurement error, which inherently exists in all measurement methods and can be amplified by a prediction model; another problem is the load data noise introduced by the load fluctuation in a computational grid. The tendency of history data is used by some tendency-track models, like AR model or polynomial fitting, to forecast future status. But the future load tendency might be distorted or concealed in the noisy data, and therefore misleads the tendency–track models, which will consequently impair prediction accuracy.

Our contribution in this paper is that we propose a new hybrid model, which can forecast far more *n*-step-ahead load status than previous models and methods, with comparatively less prediction errors and acceptable prediction interval length for task scheduler and other utilities. To deal with the dynamic nature of computational grids and the needs of task schedulers, we integrate AR model with confidence interval estimate in our hybrid model, where AR model is used as a basic *n*-step-ahead point value prediction method, and the confidence interval of future load status is estimated based on both historical and predicted point values. In order to enhance prediction accuracy, Kalman Filter [12] is used to minimize the measurement errors of history load data, and Savitzky-Golay filter functions [17] are used as the tendency revealing tool for data noise eliminating. The results of our experiments on a real computational grid environment demonstrate that this hybrid model is of an excellent capability on *n*-step-ahead load prediction, and it also achieves much smaller prediction mean square error than conventional AR model. Furthermore, *n*-step-ahead load prediction is achieved for large *n* in our hybrid model as well. The corresponding prediction advance time can be up to 50 minutes, with low prediction mean square error (0.04 on average) and acceptable confidence interval length (less than 20%) for the task scheduler.

The rest of this paper is organized as follows. Section II introduces the related work. Section III analyses the causes of

measurement errors and describes an error minimize tool – Kalman filter. Section IV presents our hybrid model on load prediction, and also introduces a signal smoothing algorithm, Savitzky-Golay filter, which is used to eliminate data noise and reveal the tendency of the data more clearly. Section V describes the experiment results of our hybrid model on a real computational grid. Finally, we conclude our work and describe our future work in section VI.

II. Related Work

Previous works [7] [10] indicate that load has such properties as self-similarity and epochal behaviour, and is strongly correlated over time, which implies that system load is consistently predictable from the past behaviour. Therefore, correctly correlating the history data with the future values is the kernel to make accurate predictions.

Several load prediction strategies in the distributed environment have been proposed in the past. [14] extends the prediction by using seasonal variation and Markov model-based meta-predictor in addition to seasonal variation for 1-step-ahead prediction. [15] proposes a multi-resource prediction model that uses both autocorrelation and cross correlation to achieve higher prediction accuracy.

The Network Weather Service (NWS) provides a dynamically monitoring and forecasting method to implement 1-step-ahead prediction on network and computational resources [5]. NWS uses various prediction methods, such as mean-based methods, median-based methods, and AR methods to forecast the future system status at the same time. NWS tracks the accuracy of all predictors, and selects the one exhibiting the lowest cumulative error measure at any given moment to generate a forecast. In this way, NWS automatically identifies the best forecasting technique for any given resource.

In Dinda's paper [6], the prediction power of several linear models, including AR, MA, ARMA, ARIMA, and ARFIMA are evaluated in detail. Their results show that simple, practical models such as AR are sufficient for load prediction and AR(16) models or better are recommended for CPU load prediction.

In [16], several homeostatic and tendency-track 1-step-ahead prediction strategies are presented and evaluated. Homeostatic methods assume that the load of a system always remains steady in a given time frame, while tendency-track strategies are based on the assumption that the tendency of the load variability exists in the load data, and can be properly revealed by the tendency-track models. The prediction of future value is adjusted according to the magnitudes of the last load measurement and the last prediction error, and the evaluations of this technique show that it outperforms NWS for CPU load prediction.

The prediction strategies mentioned above in this section are all using point value to represent the future load status. Conventional point value prediction models are often inaccurate since they can only represent one point in a range of possible behaviours. In [11], stochastic interval values are introduced in the area of system load prediction, which can address this problem. Authors of the paper define stochastic values using intervals, and then they define the interval arithmetic formulas and implement interval value on the point value prediction model. The interval value prediction model in [11] performs more effectively than point value prediction models in production environments.

III. Measurement Error and Error Minimization

Measurement error is unavoidable in every measurement method. On one hand, the monitoring sensors bring some degrees of disturbers into target system; on the other hand, some measurement methods gain performance value by estimation, for example, computing CPU availability from the average load of the system, or using the value of a time point in a period to represent the average performance of that period. Because of these, measurement would inevitably be inaccurate. Even worse, a prediction model would amplify these measurement inaccuracies in the prediction result.

In this section, we ignore the errors caused by the monitoring sensors, considering that these errors are relatively small compared with those caused by the measurement methods. In this section, the measurement error reported by the Computer Science and Engineering Department at UCSD and measurement error observed by us in Bioinformatics Application Grid (BioGrid) of ChinaGrid [20] are discussed first, followed by the introduction of the solution to the problem, which use the famous Kalman filter to minimize the error observed in the BioGrid.

A. Measurement methods and errors observed by UCSD

In [10], three kinds of measurement methods are given and the results gathered by monitoring a collection of workstations and compute servers in the Computer Science and Engineering Department at UCSD are reported. To determine the accuracy of these three methods, they compared the measurements of these three methods with the readings generated by an independent ten-second, CPU –bound process which they refer to as the *test process*. The mean absolute measurement errors of these three methods observed for different hosts at UCSD vary from as small as 3.2% to as big as 41.3%, which would inevitably deteriorate the prediction accuracy.

B. Measurement methods and errors observed in the BioGrid of ChinaGrid

The BioGrid of ChinaGrid (see section V for details) uses a simple method to generate the average performance value of every fixed time frame. The monitor centre gathers various performance data from the nodes of the Grid every 60 seconds, and uses the point value observed at the beginning of the monitor interval to represent the average performance of this time frame. Because the tasks running on the BioGrid are time-consuming and the CPU load of each node in the BioGrid is stable in a moderate period, we assume that the load data detected by the node sensors at a certain time point can be used to represent the average load of the time frame to which this time point belongs to.

We also use the *test process* method in [10] to determine the measurement accuracy. We set a *test process* to report

CPU load every 10 seconds, whereas the original BioGrid sensors report the monitoring information every 60 seconds. Fig. 1 details the measurement errors observed from a node of the BioGrid.

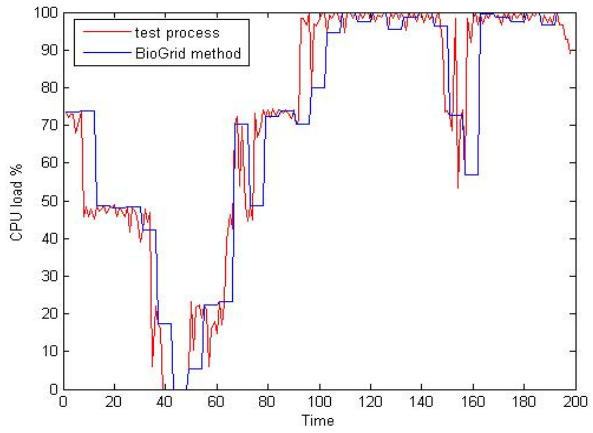


Fig. 1 CPU load measurement errors observed on one node (in Tsinghua University) of the BioGrid. Mean absolute Measurement Error is 0.069543, and Mean Square Measurement Error is 0.0145

C. Using Kalman filter to minimize measurement errors

In 1960, R.E. Kalman proposed a recursive solution to the discrete-data linear filtering problem in his famous paper [12]. Kalman filter has been extensively researched and applied, particularly in the area of autonomous or assisted navigation [9]. It provides an efficient recursive method to estimate the state of a process, and minimizes mean square error.

Kalman filter uses feedback control to estimate a process: firstly the filter estimates the process state at certain time and secondly obtains feedback from the measurements. Thus we can divide the equations of Kalman filter into two groups: *time update equations* and *measurement update equations*. The specific equations for time and measurement updates are presented in [12] and [9].

Our work applies Kalman filter to minimize measurement errors, and therefore enhance prediction accuracy, which is evaluated in section V. Kalman filter has two initial parameters, *process noise covariance Q* and *measurement noise covariance R*. In practice, these two parameters might vary with each time step or measurement, but we assume they are constants here. Presuming a very small process variance, we let Q = 1e – 5, and fix the measurement variance $R = (0.062)^2 = 0.0036$. Because this is the "true" average measurement error variance we observed from the pervious load data in the BioGrid of ChinaGrid, we would expect the "best" performance in terms of balancing responsiveness and estimate variance.

IV. PREDICTION

Task schedule and load balance strategies can benefit a lot from accurate load prediction. Because the load variability and resource consuming situation on a computational grid are strongly correlated over time and have the properties such as self-similarity, period stability, long time load prediction should be feasible. In this section, we introduce our hybrid model on load prediction for computational grid, which integrates AR model with confidence interval estimate. We use AR model as the basic linear model to predict an *n*-step-ahead load point value, and then history load data and prediction load data are used to estimate the possible load interval of an *n*-step-ahead future time frame.

Even with Kalman filter applied to eliminate the measurement errors, the noise still exists in the load data, and the tendency of load variability is concealed in the fluctuating data noise. We need a smoothing algorithm to expose the features of load data, and provide a reasonable starting approach for parametric fitting. If the curve of load data is smoothed, the tendency of future state would be more obvious and the prediction would be more accurate.

A. Smooth history load data using Savitzky-Golay filter

Savitzky-Golay smoothing filter [17] is typically used to "smooth out" a noisy signal with a large frequency span. In our work, we use Savitzky-Golay smoothing filter to smooth load data in several steps of our prediction model.

Savitzky-Golay filter can be regarded as a generalized moving average. The filter coefficients are derived by implementing an unweighted linear least square fit using a polynomial of a given degree. A higher degree polynomial makes it possible to achieve a high level of smoothing without attenuating data features. For frequency data, this filter is effective at preserving the high-frequency components of the signal. In our work, we set the parameters of Savitzky-Golay filter with *span*=101, *pd* (polynomial degree) = 4.

B. Predict future state using AR models and interval value

In [5], the performance prediction minimum mean square error and mean percentage error values for each network setting of different prediction models are summarized. Although best predictor of each performance characteristic is, in general, not obvious and varies from resource to resource, with a series evaluation in [6], AR models are recommended to predict computing resource related metric, such as CPU load.

1) Using AR models to predict a future point value

The main idea behind using a linear time series model in load prediction is to treat the sequence of periodic samples of performance, $<Z_t>$, as a realization of a stochastic process that can be modelled by linear stationary models, such as AR, MA, ARMA, ARIMA, ARFIMA, etc. [6]. The coefficients of the model can be estimated by observing past data sequence. If the action of the model can cover most of the data sequence variability, its coefficients could be used to estimate future data sequence values with low mean squared error.

AR model is a model used to find an estimation of a data based on previous inputs of the data. The general form of a *p*th-order AR(*p*) model is as follows:

$$X_t = \sum_{i=1}^{p} \varphi_i X_{t-i} + \varepsilon_t \qquad (4.1)$$

AR model consists of two parts: an error or noise part ε_t, and an autoregressive summation $\sum_{i=1}^{p} \varphi_i X_{t-i}$. The summation represents a fact that current input value depends only on the previous input values. The variable p is the order of AR model. The higher the order of AR model, the more accurate a representation will be. As the order of the model approaches infinity, we get almost an exact representation of the input data; however, the computational expense on calculating the AR(p) coefficients increases with the p order. Therefore, we should balance the prediction accuracy and the computational expense. In [6], the authors collected a large number of 1 Hz benchmark load traces, which capture all the dynamics of load signal, and subjected them to a detailed statistical analysis, drew a conclusion that AR(16) models or better are recommended for host load prediction.

The problem in AR(p) analysis is to derive the "best" values for φi, given a series X_{t-i}. The majority of methods assume that the series X_{t-i} is linear and stationary. By convention the series X_{t-i} is assumed to be zero mean, if not this is simply another term ε_t in the summation of the equation (4.1). Even for relatively large values of p, with the Burg algorithm [13] that we used to compute the AR(p) coefficients φ_i, this can be done almost instantaneously. Then the 1-step-ahead value X_t can be easily computed.

However, in highly distributed computational grid environments, 1-step-ahead load forecasting can not satisfy the demands of task scheduler. The further the load prediction of a computational grid can reach, the more appropriate the task scheduler can make scheduling strategies and balance the load. So what the scheduler in these large distributed environment needs is the n-step-ahead prediction (n = 2,3,4…or even 30,40,50..), which can forecast the load variability far more ahead before it really happens.

AR(p) model attempts to predict an output X_t of a system based on the previous inputs (X_{t-1}, X_{t-2}, X_{t-3}…) and the coefficients ($\varphi_1, \varphi_2, ...\varphi_p$). Before we use AR($p$) model to implement the n-step-ahead prediction, the following two important assumptions have to be addressed:

- The coefficients of AR(p) model represent the variability of the history load data X_{t-1}, X_{t-2},…, and they are also suitable to represent the tendency of the load in a future period, and the variability of future unknown data X_t, X_{t+1}…, X_{t+n-2}, and X_{t+n-1}.
- The prediction of the X_{t+n-1} is based on the data X_{t+n-2}, X_{t+n-3}, …, X_t, X_{t-1}, X_{t-2},…,and only data X_{t-1}, X_{t-2},…is the true measured data; X_t, X_{t+1},…, X_{t+n-2} must be computed step by step before predicting the X_{t+n-1}, and then X_t, X_{t+1},…, X_{t+n-2} can be assumed as the "true" data when we predict the X_{t+n-1}.

Based on these two assumptions, the n-step-ahead performance point value prediction is generated using the following algorithm in Table I.

TABLE I *N*-STEP-AHEAD AR(*P*) MODEL PREDICTION ALGORITHM

Compute the AR coefficients $\varphi_1, \varphi_2, ...\varphi_p$ based on the true monitoring data X_{t-1}, X_{t-2},…;
for i = 1 to n
  use the AR coefficients $\varphi_1, \varphi_2, ...\varphi_p$ and $X_{t-1+i-1}$, $X_{t-1+i-2}$,…, to compute the X_{t-1+i};
end

2) Confidence interval value prediction

AR(p) model predicts the future load status by using point value. In practice, point value is often an ideal estimate, or a value that is accurate only for a given time frame [11]. It often represents the average load in a given time frame, and can not reflect the variability of a system in this time frame. In the highly distributed systems like grids, point values can not convey enough dynamic load information for the task scheduler to make scheduling strategies.

Another type of value with the form of $X \pm Y$, or $X1 \sim X2$, is referred as interval value. The load of a computational grid varies over time, and the interval value can provide an estimate of this variability in a given time frame.

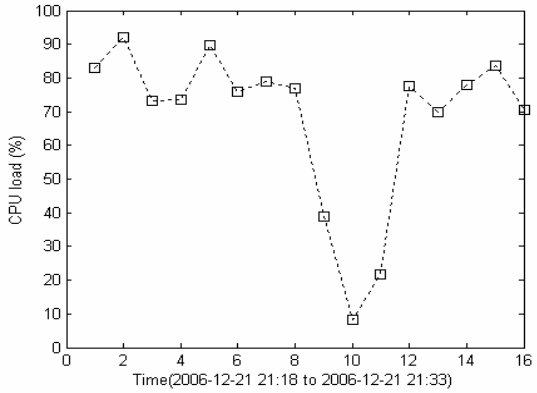


Fig. 2 CPU load variability of THU node cluster of the BioGrid

Unfortunately, it is difficult to cover all the data of a given time frame in an interval value. Sometimes the load can be extremely abnormal and fluctuant in a wide range, as shown in Fig. 2. If we want to use an interval value to represent all the load variability in a time frame, it would lengthen the interval and be meaningless for grid task scheduler. In [11], many large-sample-size real phenomena generate distributions which are close to normal distribution. Therefore we can use confidence interval to estimate this large sample size normal distribution of load, and for the small sample size data, t-distribution is more suitable. This approach can eliminate some abnormal behaviour of load variability, and dramatically shorten the interval length in these situations.

Confidence intervals represent a means of providing a range of values in which the fluctuation value can be expected to lie. This estimated range is calculated from a given set of sample data. Here, we set the sample size to be *CIwindow+N*, where *CIwindow* (Confidence interval window) is the basic sample windows size, and N is the additional windows size, which

includes the *n*-step-ahead prediction values generated by AR(*p*) model.

We set the confidence level to be 95% in our experiments. The confidence interval value here is described as a pair of two point values ($X_{lower}(t)$, $X_{upper}(t)$), where $X_{lower}(t)$ denotes the lower limit of the load prediction of time frame t, while $X_{upper}(t)$ means the upper limit of the load prediction of time frame t.

In practice, suddenly abnormal load fluctuation in a computational grid does not happen frequently and often lasts transitorily, which makes this phenomenon hard to be predicted. In this paper, we assume that load is stable in most of the time, and therefore the load confidence interval prediction values should also display as two smoothness curves which reflect the stability of the load. So after we compute the *n*-step-ahead load confidence interval, we use Savitzky-Golay filter introduced in formula 4.1 to smooth the interval data, the lower limit data and the upper limit data, to do some corrections on the original load confidence interval prediction values.

TABLE II LOAD PREDICTION PROCESS OF HYBRID MODEL

1. Using Kalman filter to minimize the measurement errors of the history data $X_{measure}(t)$, generate the more accuracy data $X_{filter}(t)$.

2. Using Savitzky-Golay filter to smooth the data $X_{filter}(t)$ with the former data $X_{filter}(t-1)$, $X_{filter}(t-2)$, ..., generate the data $X_{smooth}(t)$.

3. Compute the AR(*p*) coefficients $\varphi_1, \varphi_2, ...\varphi_p$ based on the $X_{smooth}(t)$, $X_{smooth}(t-1)$...

4. For the given *n*-step-ahead parameter *N*, using the $X_{smooth}(t)$, $X_{smooth}(t-1)$... and AR(*p*) coefficients $\varphi_1, \varphi_2, ...\varphi_p$ to compute the predict future value of $X_{predict}(t+1)$, $X_{predict}(t+2)...X_{predict}(t+n)$.

5. Compute the confidence interval ($X_{lower}(t+n)$, $X_{upper}(t+n)$) with sample size *CIwindow+N*, at confidence level 95%, using the data $X_{filter}(t-CIwindow+1)...,X_{filter}(t-2)$, $X_{filter}(t-1)$, $X_{predict}(t)$, $X_{predict}(t+1)...X_{predict}(t+n)$.

6.Using Savitzky-Golay filter to smooth the prediction interval value ($X_{lower}(t+n)$, $X_{upper}(t+n)$) with the former data ($X_{lower}(t+n-1)$, $X_{upper}(t+n-1)$), ($X_{lower}(t+n-2)$, $X_{upper}(t+n-2)$),…, generate the smoothed prediction interval value($X_{smoothed\_lower}(t+n)$, $X_{smoothed\_upper}(t+n)$)

3) The whole prediction process

The whole process of our hybrid model on *n*-step-ahead load prediction is described in Table II.

V. EVALUATION

We evaluated our hybrid model in an real grid environment, BioGrid. This computational grid provides bioinformatics supercomputing services for bioinformatics researchers through Web interface transparently. It is part of the China Education and Research Grid Project – ChinaGrid, an important project funded by Chinese Ministry of Education. ChinaGrid aims to integrate heterogeneous mass resources distributed in the China Education and Research Network (CERNET) into a public platform for research and education in China. The BioGrid has 7 nodes distributed in 6 distant cities of China (Beijing, Shanghai, Wuhan, Ji'nan, Lanzhou, Xi'an), and each of them is a cluster with processor number from 64 to 256. Without the loss of generality, we use a set of 7 series of CPU load data of the BioGrid from the ChinaGrid Super Vision (CGSV [19]) to evaluate our hybrid model, the total sample size is 13470. These CPU load data is gathered from the front-end machines of 7 nodes in the BioGrid every 60 seconds, and the CPU load of each node(cluster) is denoted as 0% to 100%, using point value to represent the average CPU load of every time frame. All the following evaluation results are generated from this set of CPU load data.

A. The optimal parameters of the AR(p) model

There is no doubt that more accurate point value prediction in our work will lead to a nicer confidence interval value forecasting. In the conclusion of [6], AR(16) model or better are appropriate for the host load prediction. We measured the accuracy of the AR *n*-step-ahead prediction with different *n*, where the result of this measurement reveals the capability of AR(*p*) model in the further *n*-step-ahead prediction.

Define the mean square prediction error as (5.1)

$$MSE_{\text{point\_value\_prediction}}(t) = \frac{1}{t+1}\sum_{i=0}^{t}(error_{\text{point\_value\_prediction}}(i)^2) \quad (5.1)$$

TABLE III MEAN SQUARE ERROR OF AR(*P*) *N*-STEP-AHEAD POINT VALUE PREDICTION, CPU LOAD DATA FROM THE BIOGRID, TOTAL 100%

P	1 step ahead	10 step ahead	30 step ahead	50 step ahead
4	0.1621	0.1658	0.2372	0.2523
8	0.0535	0.0685	0.1244	0.1498
16	0.0393	0.0651	0.1194	0.1331
32	0.0406	0.0635	0.1034	0.1346
64	0.0501	0.0583	0.1052	0.1234

From Table III we notice that AR(*p*) model performs well at $p \geq 16$ and step ahead between 1 to 10, where the mean square error could be smaller than 0.065. The results suggest that AR(*p*) model can not predict much further future CPU load of the node in a computational grid. Considering the prediction accuracy and computational expense, we use $p=32$ in our evaluations below.

B. The accuracy of the confidence interval value prediction using hybrid model

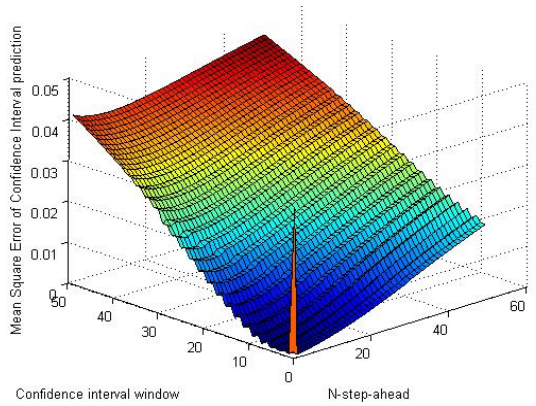
Unlike the point value prediction using AR(*p*) model which estimates a possible point value of a future time point, confidence interval value prediction gives a view of possible values in a future time frame. If the true CPU load measured by the monitoring system in a future time frame falls in the predicted confidence interval value, we define the prediction accuracy of this situation as 100%, in other words, the

prediction error is 0. The mean square error of the confidence interval value prediction is defined as (5.2), and Fig. 3 shows the evaluation results.

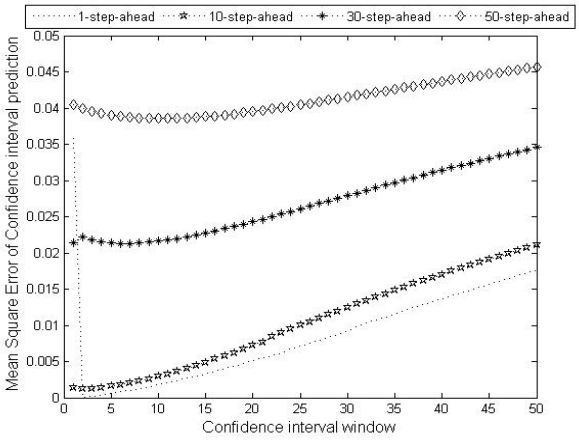
$$MSE_{\text{int}erval\_value\_prediction}(t) = \frac{1}{t+1}\sum_{i=0}^{t}(error_{\text{int}erval\_value\_prediction}(i)^2) \quad (5.2)$$

The $error_{\text{int}erval\_value\_prediction}(i)$ here is slightly different from the *error* in the point value prediction. We compute the $error_{\text{int}erval\_value\_prediction}(i)$ using the definition of (5.3).

$$\begin{aligned}&\text{if } X_{lower}(i) <= X_{true}(i) <= X_{upper}(i)\\ &\quad error_{\text{int}erval\_value\_prediction}(i) = 0 \ ;\\ &\text{Else}\\ &\quad error_{\text{int}erval\_value\_prediction}(i)\\ &\quad = Min\{abs(X_{true}(i) - X_{lower}(i)), abs(X_{true}(i) - X_{upper}(i))\}\end{aligned} \quad (5.3)$$



(a)

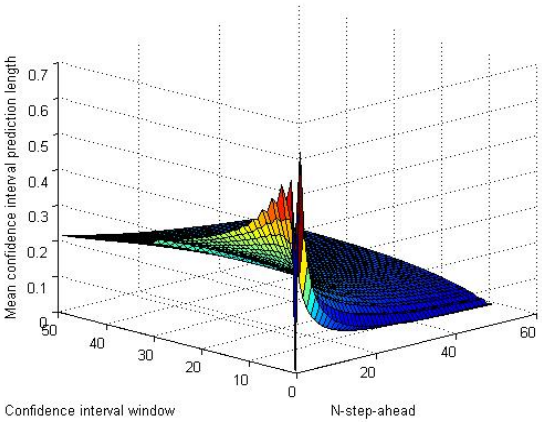


(b)

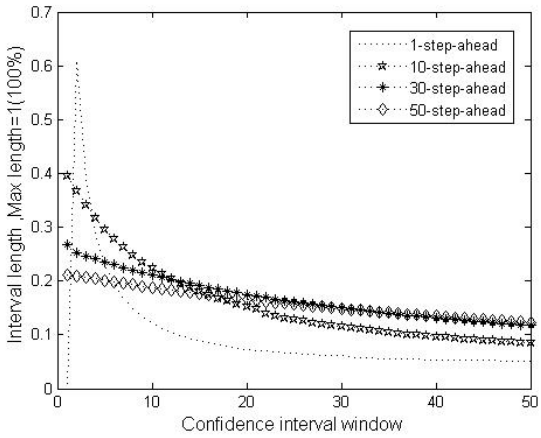
Fig. 3 Mean square error of hybrid model prediction

C. The prediction confidence interval length using hybrid model

As shown in Fig. 3, our hybrid model is of an outstanding forecasting capability. It illustrates that we should use as small as possible the *CIwindow* to achieve small prediction mean square error. But there is another problem about the confidence interval length of the prediction, which is often ignored in previous interval prediction models. If the confidence interval is too wide, it may be meaningless for the task scheduler. For example, the range of CPU load is described as 0% to 100%, but the prediction load confidence interval length of a future time is as wide as 60%. This prediction is so faint that the task scheduler can hardly arrange the task dispatching based on it. On the contrary, if this interval length of prediction can be limited as small as 15% or 20%, it is much easier for the scheduler to recognize the future status of the system CPU load.



(a)



(b)

Fig. 4 Mean predicting confidence interval length

Fig. 4 demonstrates the influence of the parameters, *CIwindow* and N, on the prediction interval length. Noticeably, the mean confidence interval length is almost below 20% when the *CIwindow* is bigger than 15.

As a compromise between the prediction accuracy and interval length, we set the *CIwindow* as 20 when using our model to forecast the CPU load of the BioGrid.

D. Comparison between hybrid model and AR model

We compared the prediction accuracy of our hybrid model and AR model on a set of 7 series of CPU load data collected from the BioGrid. The prediction mean square error of our

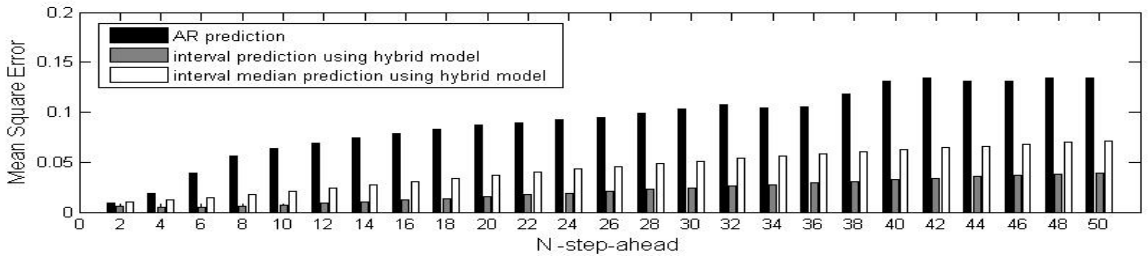


Fig. 5 Prediction errors compare on 7 series of CPU load data collected from the BioGrid

hybrid model (using equation (5.2)) and AR model (using equation (5.1)) on this set of load data are shown in Fig. 5. The experimental results show that the hybrid model outperforms AR model on the CPU load prediction. The prediction mean square error of hybrid model is 77.07% less on average than that of AR model. We also evaluated the prediction mean square error of the interval median prediction by using the hybrid model (using equation (5.1)). This variation from our hybrid model also shows better prediction accuracy that the prediction mean square error is 51.31% less on average than that of AR(32) model. In these evaluations, we set the parameter of the hybrid model as *P*=32, *CIwindow* =20.

E. The power of Kalman filter in minimizing the measurement errors

In section III, we analyse the causes of the measurement errors. Measurement error is unavoidable and may be amplified by prediction models, thus impairing the prediction accuracy. In our work, we use Kalman filter to minimize these measurement errors. We compare the mean square error of our hybrid model using Kalman filter with the one without using Kalman filter. The results are shown in Fig. 6, which demonstrates that Kalman filter works very well on minimizing the measurement errors and the prediction accuracy is improved consequently.

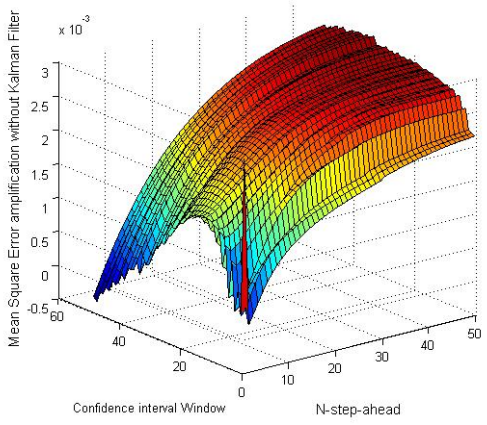
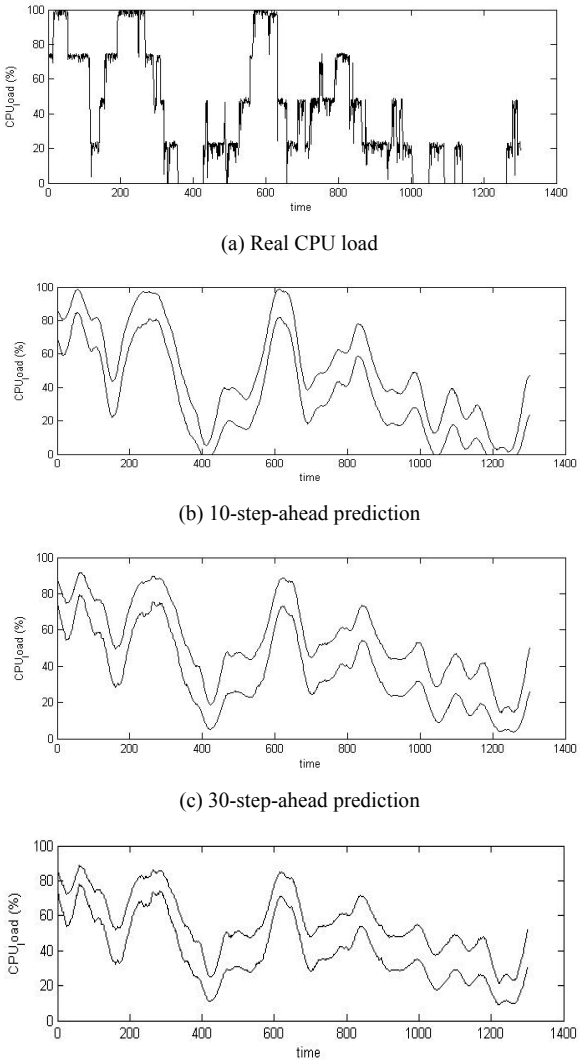


Fig. 6 Mean square error amplification without using Kalman filter, where Z-label = $MSE_{no\_kalman\_filter} - MSE_{using\_kalman\_filter}$

F. A sample of CPU load prediction using hybrid model

Fig. 7 is a sample of the CPU load prediction using our hybrid model. Fig. 7(a) is the CPU load measurement data of the BioGrid. Sample time is from 2006-12-23 16:27 to 2006-12-24 14:08, total 1301 sample point, with sample interval of 60 seconds, in Tsinghua University node. Fig. 7(b), (c), (d) are the 10, 30 and 50 step-ahead CPU load confidence interval prediction by hybrid model, using the same parameter *p*=32 and *CIwindow*=20. Table IV summarizes the mean square error of these predictions at different *n*-step-ahead and the mean interval length.



(a) Real CPU load

(b) 10-step-ahead prediction

(c) 30-step-ahead prediction

(d) 50-step-ahead prediction

Fig. 7 CPU load confidence interval prediction using hybrid model

TABLE IV CONFIDENCE INTERVAL PREDICTION ERRORS AND INTERVAL LENGTH STATISTICS OF FIGURE 7

Evaluation metrics	**10-step -ahead**	**30-step -ahead**	**50-step -ahead**
Confidence interval prediction MSE	0.0063	0.0259	0.0428
Mean interval length of prediction (confidence level 95%)	17.34% ±0.21%	18.05% ±0.19%	16.72% ±0.18%

Fig. 7 and Table IV show that our hybrid model works excellently in CPU load prediction, which is always the bottleneck for a computational grid. Using the parameter p=32 and *CIwindow*=20, the interval length is limited to below 20% and the mean square error is smaller than 0.05 even for 50-step-ahead (50 minutes ahead) CPU load.

VI. CONCLUSION AND FUTURE WORK

To predict the load of a computational grid, we have developed a hybrid model which integrates AR model with the confidence interval estimate. Whereas the point value prediction is always the ideal estimate of the load in the future time point, the confidence interval prediction adopted in this hybrid model can reflect the load variability in a future time frame and convey more information to the task scheduler of a computational grid.

In order to enhance the prediction accuracy, we use Kalman filter to minimize the load measurement errors, and Savitzky-Golay filter to smooth the history data. The evaluation results demonstrate that these noise eliminating tools perform very well and lead to a significant improvement on prediction accuracy.

Considering the trade-off between the prediction accuracy and the confidence interval length, we use parameters p=32 and *CIwindow* = 20 to predict the CPU load of the BioGrid. The prediction advance time can be even 50-step-ahead long, with significant less prediction mean square error than the conventional AR model and has acceptable interval length for the schedule algorithm. The optimal parameter values may be slightly different according to the different computational applications and programs running on a computational grid, and we will evaluate the prediction performance variability of our hybrid model under different applications and parameters in the future. Some machine learning mechanism and parameter auto-adaptation function could also be added in our model to fit in different conditions.

ACKNOWLEDGEMENT

This Work is supported by Natural Science Foundation of China (90412006, 90412011, 60573110, 90612016, 60673152), National Key Basic Research Project of China (2004CB318000, 2003CB317007), National High Technology Development Program of China (2006AA01A108, 2006AA01A111, 2006AA01A101), and IST programme of the European Commission (DG Information Society and Media, project n° 034442). The authors acknowledge Ms. Jing Zhu of UCSD and Dr. Ying Zhao of Tsinghua University for their revision to this paper.

REFERENCES

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

[2] C. Liu, L. Yang, I. Foster, and D. Angulo, "Design and Evaluation of a Resource Selection Framework for Grid Applications," Proceedings of the 11th IEEE International Symposium on High-Performance Distributed Computing (HPDC 2002), Edinburgh, Scotland, 2002.

[3] D. Lu, H. Sheng, and P. Dinda, "Size-based scheduling policies with inaccurate scheduling information," 12th IEEE Int'l Symp, on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS 2004), pp. 31-38, 2004.

[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] R. Wolski. "Dynamically Forecasting Network Performance Using the Network Weather Service," Journal of Cluster Computing, 1:119–132, January 1998.

[6] P. A. Dinda and D. R. O'Hallaron, "Host load prediction using linear models," Cluster Computing 3, 4 (2000).

[7] P. A. Dinda and D. R. O'Hallaron, "The Statistical Properties of Host Load," Fourth Workshop on Languages, Compilers, and Run-time Systems for Scalable Computers (LCR 1998), Pittsburgh, PA, 1998.

[8] Y. Zhang, W. Sun, and Y. Inoguchi, "CPU Load Predictions on the Computational Grid," IEEE, Proc. in 6th International Conference on Cluster Computing and the Grid (CCGrid 2006), pp. 321-326, May 2006.

[9] Greg Welch , Gary Bishop, "An Introduction to the Kalman Filter," University of North Carolina at Chapel Hill, Chapel Hill, NC, 1995

[10] R. Wolski, N. Spring, and J. Hayes, "Predicting the CPU availability of Time-shared Unix Systems," Proceedings of 8th IEEE High Performance Distributed Computing Conference (HPDC 1999), 1999.

[11] J. Schopf and F. Berman, "Performance prediction in production environments," in: 12th International Parallel Processing Symposium, Orlando, FL (April 1998) pp. 647-653.

[12] Kalman, R. E. 1960. "A New Approach to Linear Filtering and Prediction Problems," Transaction of the ASME—Journal of Basic Engineering, pp. 35-45 (March 1960).

[13] Marple, S. L., Jr., *Digital Spectral Analysis with Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1987

[14] 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 (CCGrid 2004), pp. 765-772, 2004.

[15] 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 (CCGrid 2004), pp. 1-8, 2004.

[16] L. Yang, I. Foster, and J.M. Schopf, "Homeostatic and tendency-based CPU load predictions," Int'l Parallel and Distributed Processing Symp. (IPDPS 2003), pp. 42-50, 2003.

[17] Orfanidis, S.J., *Introduction to Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1996.

[18] Y. Zhang, W. Sun, and Y. Inoguchi, "Predicting Running Time of Grid Tasks based on CPU Load Predictions," 7th IEEE/ACM International Conference on Grid Computing(Grid 2006) ,Sept. 2006

[19] (2006) The ChinaGrid Super Vision website. [Online]. Available: http://www.chinagrid.edu.cn/CGSV/.

[20] (2006) The ChinaGrid website. [Online]. Available: http://chinagrid.hust.edu.cn or http://www.chinagrid.edu.cn.