Adaptive Workload Prediction of Grid Performance in Confidence Windows

Yongwei Wu, *Member*, *IEEE*, Kai Hwang, *Fellow*, *IEEE*, Yulai Yuan, and Weimin Zheng, *Member*, *IEEE*

Abstract—Predicting grid performance is a complex task because heterogeneous resource nodes are involved in a distributed environment. Long execution workload on a grid is even harder to predict due to heavy load fluctuations. In this paper, we use Kalman filter to minimize the prediction errors. We apply Savitzky-Golay filter to train a sequence of confidence windows. The purpose is to smooth the prediction process from being disturbed by load fluctuations. We present a new *adaptive hybrid method* (AHModel) for load prediction guided by trained confidence windows. We test the effectiveness of this new prediction scheme with real-life workload traces on the AuverGrid and Grid5000 in France. Both theoretical and experimental results are reported in this paper. As the lookahead span increases from 10 to 50 steps (5 minutes per step), the AHModel predicts the grid workload with a *mean-square error* (MSE) of 0.04-0.73 percent, compared with 2.54-30.2 percent in using the static point value *autoregression* (AR) prediction method. The significant gain in prediction accuracy makes the new model very attractive to predict Grid performance. The model was proved especially effective to predict large workload that demands very long execution time, such as exceeding 4 hours on the Grid5000 over 5,000 processors. With minor changes of some system parameters, the AHModel can apply to other computational grids as well. At the end, we discuss extended research issues and tool development for Grid performance prediction.

Index Terms—Grid computing, performance prediction, workload characterization, autoregression method, Kalman filter, Savitzky-Golay filter, and parallel applications.

1 Introduction

Aggregated grid performance is directly related to the collective workload to be executed on a large number of processors scattered on all participating grid sites. Predicting the collective grid workload is a very challenging task [1], [3], [13], [34], [43] because heterogeneous resources are widely distributed under the control of different administrations. We propose a new adaptive approach to predicting workload on computational grids within a confidence window, which is dynamically trained with the load variations.

The grid workload is represented by a collective *load index* among all processors. The load index $X(t)$ is the percentage of processors utilized within a unit time interval $[t-1, t]$. All discrete-time instances t are denoted by nonnegative integers. For simplicity, we assume 5 minutes per time step. Load index reflects the *CPU utilization rate* among all available processors in a grid platform. For example, $X(t) = 0.45$ implies that 45 percent processors are busy during the observation period.

Workload is difficult to predict due to the lack of runtime information on job scheduling and resource allocation on remote machines [22], [29]. Predicted workload may contain errors, if loading noises cannot be filtered out. Some previous workload prediction methods have ignored two problems: One is the workload measurement errors and another is the load data noise introduced by workload fluctuation [11], [19].

In this paper, we offer an adaptive method to predict the workload for parallel execution on grid resource sites. We use Kalman filter [23], [39] and Savitzky-Golay smoothing techniques [33] to filter out potential errors. Filtering out noises from workload fluctuation and measurement errors, one can predict grid workload more accurately. This prediction scheme can forecast execution time [44], [46] and guide the job scheduling strategies [21], [37].

Traditional *point value* prediction methods [11], [40], [41] apply a very short prediction window. Although they may work well to predict CPU load in centralized computer systems, they do not work well on large-scale production grids due to the long execution time expected. In fact, point value prediction can hardly cover workload fluctuation in a long time frame.

In general, the load index is used to estimate the percentage of peak performance achievable on a given computational grid. The management console of such a grid can monitor the CPU utilization. However, it is rather difficult to predict CPU utilization rate in the future [21]. The grid performance in Tflops can be predicted by dividing the total workload integrated over the entire observation period by the execution time of job completed [4], [11], [19].

Workload managers in large grid infrastructures are notoriously weak in determining correct scheduling scenarios, which affects the application execution time on the grid. This paper is about predicting the future workload within a reasonable confidence range. The narrower is the prediction range, the higher is the accuracy of prediction. Long-range workload prediction is bound to have some errors. But we

• *Y. Wu, Y. Yuan, and W. Zheng are with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China. E-mail: {wuyw, yuan-yl05, zwm-dcs}@tsinghua.edu.cn.*
• *K. Hwang is with the University of Southern California, 3740 McClintock Ave., EEB 212, Los Angeles, CA 90089-2562. E-mail: kaihwang@usc.edu.*

Manuscript received 17 Apr. 2009; revised 7 Aug. 2009; accepted 11 Aug. 2009; published online 18 Aug. 2009.
Recommended for acceptance by A. Zomaya.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2009-04-0170.
Digital Object Identifier no. 10.1109/TPDS.2009.137.

try to minimize the prediction errors by using lookahead filtering techniques within a trained confidence range.

The remainder of the paper is organized as follows: Section 2 introduces related works and outlines our unique approach. Section 3 discusses the causes of workload measurement errors. We suggest to use Kalman filter to minimize the errors. Section 4 presents our hybrid workload prediction scheme, which is denoted by HModel. Section 5 analyzes the impact of system and parameter values on the prediction accuracy. Section 6 presents an adaptive prediction scheme, AHModel. Section 7 reports the experimental results on real-life workload traces on Grid5000 and AuverGrid in France. Finally, in Section 8, we summarize the contributions and suggest directions for further research.

2 Related Work and Our New Approach

Related work is reviewed first. Then we introduce our unique approach to predicting grid workload accurately.

2.1 Related Work

In the past, many research groups attempted to predict grid performance [3], [5], [9], [11], [13], [14]. Grid workload varies with time but it is correlated in different time spans [10], [41]. Thus, system workload is predictable from checking the historical performance records. By correlating historical workload data, we predict the future workload, and thus, the Grid performance with controlled accuracy.

Feitelson has developed some workload models for parallel computers [12], [30]. Berman's group at UCSD has developed tools for prediction of parallel applications on Grids [34]. The Grid research group led by Fortes at University of Florida has developed a predictive application-performance model for computational grids [24]. Supercomputer workloads were characterized in [6] and [36].

Maheswaran et al. [31] have studied the dynamic mapping issues in heterogeneous computers. Risk-tolerant scheduling of parallel jobs on grids is studied by Song et al. [37]. Li and associates have reported progress on workload characterization for grids [24], [25], [26]. In particular, Li's PhD Thesis [25] has given a comprehensive treatment of this subject area.

The concept of *confidence window* was first proposed by Schopf and Berman [34] to address the accuracy issue. Historical workload data are used by tendency-track models, e.g., AR model or polynomial fitting methods [32], [33], to forecast future workload status. However, this tendency may be distorted or concealed in noisy data, which will consequently impair the accuracy of workload prediction.

Workload variation and resource consumption in grid environments exhibit a wide range of dynamics, such as *sudden local change*, *sudden level change*, etc., [27]. Adaptive techniques are introduced. The purpose is to capture the dynamic characteristics. Normally, two types of adaptation approaches are introduced to improve prediction accuracy.

One approach is the use of *adaptive static predictor* [41], [47] with fixed parameters as used in AR method. This scheme works under the hypothesis that predictor varies with resource types [40]. However, the predictor for a particular resource may also change over time. This approach is to select the best predictor among a number of predictors with the minimum prediction errors for a particular load pattern.

The second approach is adaptive prediction implemented with parameter adaptation [27]. When an adaptation is triggered by workload variation, this adaptive prediction automatically modifies some system parameters to adapt with the resource consumption patterns. The purpose of this approach is to achieve lower errors in workload prediction than using AR method with fixed parameters.

Jiang et al. [20] extend the prediction by using Markov model-based metapredictor in addition to seasonal variation recognition for 1-step-ahead prediction. A multiresource prediction model is proposed in [27], which uses both autocorrelation and cross correlation to achieve a higher prediction accuracy.

Several workload prediction methods [4], [10], [15], [16], [20], [34], [40], [41], [46] measured mean value or median performance, or using AR, polynomial fitting, Markov model, or seasonal variation to predict performance with various lookahead times. In a large-scale grid environment, tasks usually require long time to run, thus, the task scheduler needs a large lookahead span to predict the performance [28].

Network Weather Service (NWS) in US [40] provides a dynamically monitoring and forecasting method to implement 1-step-ahead prediction. Various prediction methods are used together in the NWS to forecast the performance of a Grid system. NWS tracks the prediction accuracies of all these predictors and selects the one exhibiting the lowest prediction error.

Dinda and O'Hallaron [10] evaluated the prediction power of several models, including the AR, *Moving Average* (MA), etc., methods. Their evaluation results show that a simple predictor model such as AR is sufficient for a single CPU load prediction. In [44], several 1-step lookahead predictors are evaluated. Static prediction methods are only effective with steady workload. Adaptive prediction is needed to handle time-varying workloads.

The prediction of future value is adjusted according to the magnitudes of the last workload measurement. Conventional point value prediction models are often inaccurate, since they can only represent one point in a range of possible behaviors. In [34], stochastic interval values are introduced to address this drawback. The interval value prediction model performs more effectively than point value prediction model in production grid environments.

Adaptive predictor integration [47] adopts an approach for predictor integration based on learning historical predictions. Classification algorithms like the *k-nearest neighbor* are based on supervised learning. This approach can achieve higher predictor accuracy. The NWS model ends up with lower cumulative error [40]. A multiresource prediction model was proposed in [27] using both autocorrelation of single resource and cross correlation over multiple resources.

2.2 Adaptive Prediction in Confidence Windows

We introduce in Fig. 1 an adaptive workload prediction process by a time series of events. Current time is marked at t. The real workload is shown by a series of dots up to time t. The future load indices can only be predicted beyond time t. The conventional prediction scheme, like the AR(p) method, predicts the future load indices using load information from the past p point values. The

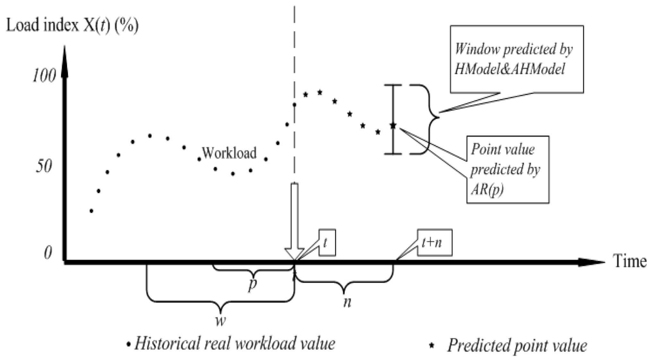

Load index X(t) (%)
100
50
0
Workload
Window predicted by HModel&AHModel
Point value predicted by AR(p)
t
t+n
Time
p
n
w
• Historical real workload value
* Predicted point value

Fig. 1. Concept of adaptive lookahead prediction of grid workload using confidence windows derived from historical load information and projected load values in the future.

predicted point values are marked by stars in Fig. 1. Details of the AR(p) scheme are given in Section 4.

An n-step lookahead process predicts the load index $X(t+n)$ using all load data in a *historical interval* w plus predicted load values in a lookahead span of n time steps beyond t. Instead of predicting a point value, which may not be accurate due to the load fluctuation, we use a *confidence window* to predict $X(t+n)$ within a range $\{X_l(t+n), X_u(t+n)\}$.

For example, a window {0.75, 0.82} implies that the predicted workload is considered totally accurate or acceptable if the value falls within this range. Any predicted workload outside the range should introduce some unacceptable errors. In other words, this range reflects the *error tolerance,* which can be accepted by the users of the workload prediction system. The sequence of confidence windows plays a crucial role to minimize the prediction errors by lookahead filtering within confidence windows. The *historical interval* is dynamically trained to support adaptive load prediction.

To benefit readers, we summarize in Table 1 the key parameters and their default values used. The time interval parameters $(\boldsymbol{p}, \boldsymbol{n}, \boldsymbol{w}, \boldsymbol{u})$ are used to trace back the past, look into the future, review the history, and train the historical interval size. The threshold values ε and $\boldsymbol{v}$ are used for historical interval validation and sample training, respectively. The total time span from historical load to predicted future load is simply $w+n$.

The basic idea of adaptive prediction and preliminary results of this work were presented in the *IEEE CCGrid 2008* [42]. The conference version, containing no theoretical results, overlaps with this version in less than 20 percent. A comprehensive set of new experiments is presented. In summary, we make original contributions in four technical aspects of dynamic workload prediction on large-scale grids.

1. We reveal the causes of sampling errors in workload prediction. We use Kalman Filter [23] to minimize sampling errors. We apply Savitzky-Golay filter [33] to smooth data noises from unexpected load fluctuation.
2. We propose a new *hybrid model* (HModel) to integrate the AR model within an estimated *confidence window*. The purpose is to forecast workload status n steps ahead of current time. This method results in less prediction error.

TABLE 1
Notations and Basic Terms Used in the Paper

Symbol	Basic Definition and Default Value Used
$X(t)$	***Load index*** at discrete time t: CPU resource utilization rate across the entire grid platform at time t
p	***Traceback span*** in AR(p) prediction method (Default value p = 32 in all grid trace experiments)
n	***Lookahead span*** : Time steps in lookahead prediction process (Default range $10 \le n \le 50$ applied)
$C(t)$	***Confidence window :*** The load index range $\{X_l(t+n), X_u(t+n)\}$ predicted at time t for a future workload at time $t+n$
w	***Historical interval:*** : Time span of historical workload used in the prediction process (Default range $0 \le w \le 130$)
u	***Training period :*** Time needed to update historical interval (Default value $u = n$ used in all experiments)
ε	***Tolerance threshold*** : Error tolerance on the predicted load index (default value ε = 5% used in experiments)
v	***Validation period :*** Time span needed to validate the modified historical interval

3. We use *mean-square error* (MSE) analysis and create an interval control process to generate an *adaptive hybrid model* (AHModel) for workload prediction. This model automatically adapts to the change of Grid workload.
4. Experiment results demonstrate the advantages of using the AHModel, which outperforms both the HModel and AR methods. The AHModel appeals to predicting long execution jobs on computational grids.

3 Workload Traces and Sampling Error

Workload traces on two French grids are introduced in this section. Sampling errors are analyzed. A poor load prediction method may amplify the errors. Wolski et al. [40] have introduced three prediction methods for workstations and servers. The mean sampling errors of their methods vary from 3.2 percent to 41.3 percent. Our sampling errors are around 6.9 percent on ChinaGrid nodes at Tsinghua University.

3.1 Job Characteristics on AuverGrid and Grid5000

Two grid workload traces come from the AuverGrid [3] and Grid5000 [18] in France. AuverGrid is a production grid, consists of five clusters. It has 475 CPUs in total, which are geographically scattered around Auvergne area in France. Grid5000 is an experimental grid platform, built over 5,000 processors in nine sites throughout France.

The *Grid Workloads Archive* (GWA) [17] collected job traces from these two grids. GWA records information of all jobs submitted to these two grid systems by 11 data items: (Job Number, Submit Time, Wait Time, Runtime, Number of Allocated Processors, Average CPU Time Used, used Memory, Requested Number of Processors, Requested Time, Requested Memory, Status, User ID, Group ID, Executable (Application) ID, and Site ID). These data items are used at different places of our AHModel.

Table 2 summarizes job statistics on the two French grids. The traces characterize the jobs submitted. Normalized workloads are calculated as the average CPU utilization rate

TABLE 2
Job Characteristics on AuverGrid and Grid'5000

Metrics	AuverGrid	Grid'5000
N_{total_job}	404176	1020195
$Rate_{error\_job}$	13.99%	8.67%
$CPUTime_{total}$ (sec)	8,755,024,087	20,549,246,683
$JobRunTime_{total}$ (sec)	8,755,024,087	2,512,363,279

TABLE 3
Load Trace Characteristics on AuverGrid

Trace Name	Mean Workload	Standard deviation	Load pattern
Trace1	68.83 %	22.62 %	*sudden level change*
Trace2	46.52 %	23.53 %	*Sudden fluctuation*
Trace3	70.69 %	13.58 %	*gradual level change*

in every 5 minutes over all processors in the grids. The workload traces on these two grids were collected in 2006 [3], [15]. Both AuverGrid and Grid'5000 have a large number of failed jobs in GWA. On AuverGrid, the job failure rate is 13.9 percent. Grid5000 has a 8.67 percent job failure rate.

3.2 Workload Traces on Two Grid Platforms

Two grid workload traces on *AuverGrid* and *Grid5000* are shown in Fig. 2. Job log entries of AuverGrid and Grid'5000 consist of two types: those run successfully and those for failed jobs. Measurement errors are possibly caused by failed jobs or erroneous job entries.

By a 2006 trace report [6], the average workload on AuverGrid ranges from 58.5 to 80 percent. As shown in Fig. 2a, AuverGrid has a heavy and fluctuated workload. Load distribution of Grid5000 is shown in Fig. 2b from January to October 2006. The average workload on Grid5000 is 10.6 percent. During these 10 months, Grid5000 was lightly loaded and fluctuated around the average load.

All workload traces applied in this section are collected from AuverGrid [3]. We apply three workload traces: Trace1 for *sudden level change*, Trace 2 for *sudden fluctuation*, and Trace 3 for *gradual level change*. These three traced load characteristics are summarized in Table 3. The sample space has collected 1,000 load index values. The mean value is related to the load level and the standard deviation reflects the fluctuations around the mean value.

3.3 Sampling Errors on Three Grid Platforms

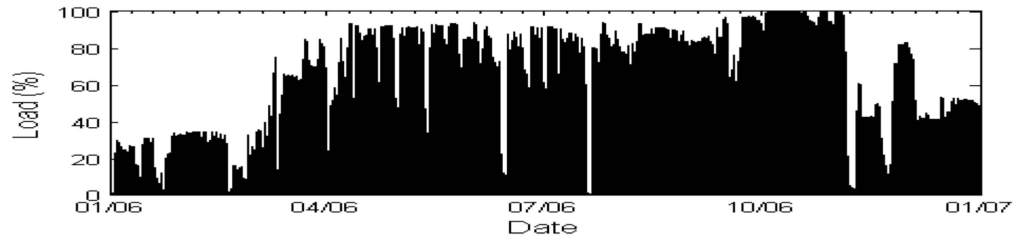
ChinaGrid integrates high-performance computers, data resources and tools, and high-end experimental facilities over 20 major universities in China [8]. Potentially, the system can provide 20 Teraflops of aggregated computing power over 200 terabytes of online archival data storage. The ChinaGrid Super Vision (CGSV) gathers nodal load data in every 60 seconds. The center records the point value at the beginning of every 60 seconds as the measured load shown by the solid line in Fig. 3.

We apply the test process method developed in [41] to measure the nodal load as plotted by the dotted lines in Fig. 3. The method reports CPU load in every 10 seconds. Fig. 3 plots both CGSV results and test process measurements at one ChinaGrid node at Tsinghua University in 2008. The mean sampling error on the load index of this node was measured as 6.9 percent, after comparing the two measured curves.

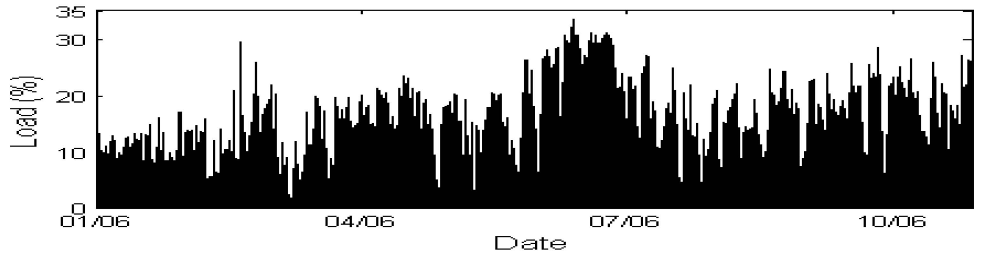
Based on Table 2, we define the *absolute measurement error* of AuverGrid and Grid5000 as follows:

$$AbsoluteMeasurementError = \frac{Rate_{error\_job}/2}{(1 - Rate_{error\_job}) + Rate_{error\_job}/2}. \quad (1)$$

By (1), the absolute measurement errors of AuverGrid and Grid'5000 are calculated as 7.5 and 4.5 percent, respectively.



(a)



(b)

Fig. 2. Workload variation in two French grid systems from January to October 2006. AuverGrid is heavily loaded with an average load exceeding 70 percent and the Grid5000 is lightly loaded with an average load of 10 percent. (a) AuverGrid. (b) Grid'5000.

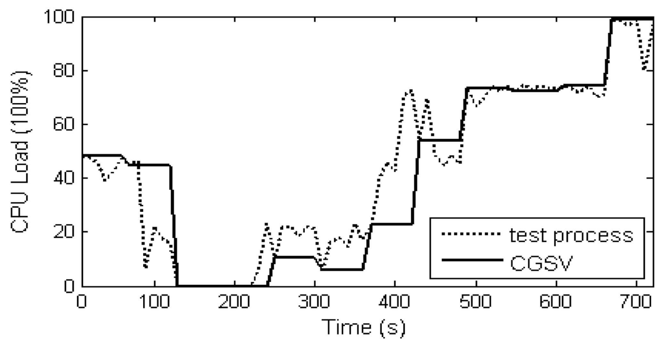


Fig. 3. CPU load sampling errors measured at a ChinaGrid node at Tsinghua University, Beijing, China, in 2008.

4 Hybrid Prediction Method

In this section, we introduce two filtering techniques by Kalman [23] and Savitzky-Golay [33]. We first specify the AR method for point value load prediction in Algorithm 1. Then we specify the hybrid model (HModel) for load prediction in confidence windows. The generation of successive confidence windows is specified in Algorithm 2.

4.1 Kalman Filtering and Savitzky-Golay Filtering

Kalman filter is characterized by a *time update process* and a *measurement update process*. The time update projects the process state and estimates the error covariance. The measurement update verifies the priori estimates and generates a posterior estimate. We port the Matlab Kalman filter packages into our prediction scheme. We compare the MSE of HModel using Kalman filter with the case of no Kalman filtering. The MSE difference is plotted in Fig. 4.

The difference lies between 0.1 and 0.3 percent. The MSE difference raises significantly with the increase of the lookahead span n. The MSE difference increases with longer *historical interval* w. The spike near the origin is a degenerate case, which should not be counted. These results clearly demonstrate that Kalman filter works very well on minimizing measurement errors.

Workload data are stable in large part of the time, but sometimes, they fluctuate as seen in Fig. 2. These noises are caused by monitoring sensor, process switching in the host system kernel, or network communications, etc. The purpose of data smoothing is to expose these errors. We use Savitzky-Golay filter to smooth workload data in several steps of our prediction model.

In the frequency domain, this filter is effective to preserve high-frequency parts. This filter is capable of preserving higher moments of the peak values. These characteristics make the Savitzky-Golay filter effective to smoothen the predicted workload. The filter coefficients are derived by historical information. In our work, we set Savitzky-Golay filter with a frame size $f = 51$ and a polynomial degree $d = 4$. [33].

4.2 Autoregression Method

This AR method was well described in [10] and [32]. The method applies a time sequence of load samples, $X(t)$. The coefficients used in the AR method are estimated from past load data sequences. These coefficients can be also used to predict future load sequence.

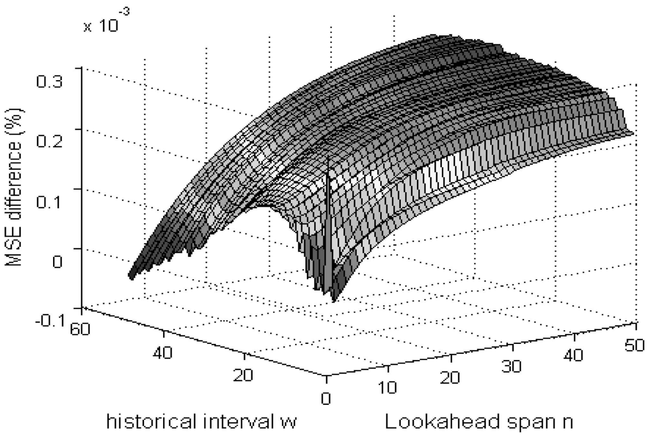


Fig. 4. Differences in mean square errors in workload prediction with and without Kalman filtering.

In [40], the MSE of different prediction models was reported. The AR method has been used to predict CPU workload. We specify an AR(p) method by the following load instance at time t in a polynomial format:

$$X(t) = \sum_{i=1}^{p} \varphi_i X(t-i) + \varepsilon_t, \tag{2}$$

where p is the traceback span, called the *order* of the prediction method. The first term gives an estimated value of the current load index by a weighted sum of past p load values. The p coefficients $\{\varphi_i\}$ are trained from past workload. The ε_t term covers the noise introduced at time t.

In essence, the $\mathrm{AR}(p)$ model predicts the load index $X(t+1)$ at time t based on the past p load values $\{X(t), X(t-1), \ldots, X(t-p+1)\}$ using the coefficients $\{\varphi_1, \varphi_2, \ldots \varphi_p\}$. The larger is the order p of the AR(p) method, the higher is the prediction accuracy expected. However, the cost to generate the weighting coefficients increases with the order p.

In theory, we should expect no errors if p approaches infinite, which is totally cost prohibitive. The real difficulty of this AR method lies in finding the accurate coefficients φ_i, given a series of past load indices $X(t-\mathrm{i})$. We will use Burg's algorithm [32] to compute these coefficients. Most AR methods assume that the series $X(t-\mathrm{i})$ is linear and stationary. The load series $X(t-\mathrm{i})$ should have a zero mean value. Otherwise, the error term ε_t is needed to offset the noises to achieve a zero mean. We specify the AR method in Algorithm 1.

Algorithm 1: Prediction using AR model

Input: Current time *t*, Traceback time span *p*, previous load index series {*X*(*t-p*),..., *X*(*t-1*), *X*(*t*) }, Lookahead time span *n*
Output: Predicted future load index series [X(*t*+1),t+2),...., X(*t+n*)] *n* steps ahead from the current time *t*.
Procedure:

```
1.  Compute coefficients $\varphi_1, \varphi_2, \ldots \varphi_p$ using Burg's Algorithm
    trace back over p load indices { X(t-p),...,x(t-1), X(t)}
2.  for i = 1 to n
3.      Predict future load index X(t+i) using
        Eq.(2) over the coefficients $\varphi_1, \varphi_2, \ldots \varphi_p$
4.  endfor
```

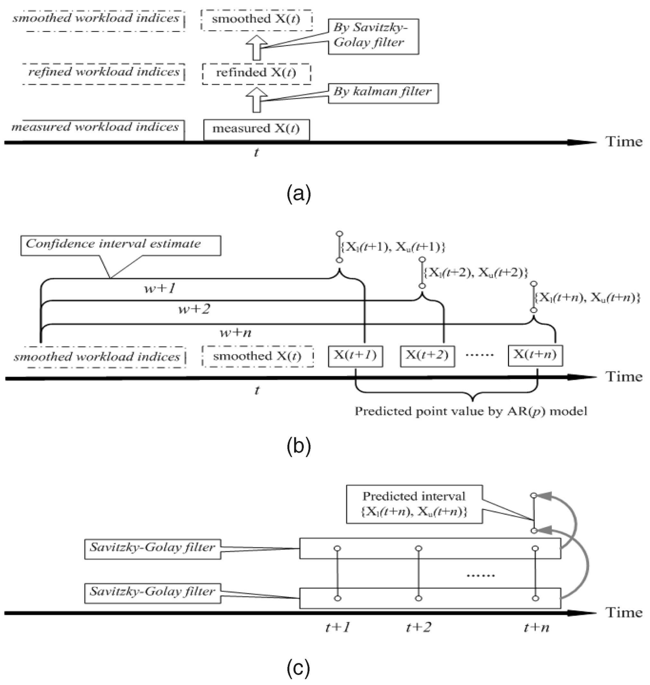


Fig. 5. The process of hybrid AR prediction method (HModel in Algorithm 2). (a) Workload processing at time t. (b) Confidence window from step 1 to step n. (c) Smoothing process in the lookahead window.

An n-step lookahead prediction ($n = 1, 2, 3, \ldots, 30, 40, 50, \ldots$) is desired to predict long execution workload. Two conditions were observed below based on our experiments.

a. The coefficients of AR(p) method vary very little with the historical load data $X(t), X(t-1), \ldots X(t-p+1)$ on the same workload suite.
b. To predict the load $X(t+n)$, we use data series

$$\{X(t+n-1), X(t+n-2), \ldots, X(t-1), X(t), X(t-1), \ldots\}.$$

Only $X(t)$, $X(t-1)$, and the earlier data points are measured data. The remaining $X(t+1), X(t+2), \ldots, X(t+n-1)$ are predicted ones. The load series $\{X(t-1), X(t), \ldots, X(t+n-1)\}$ is used to predict the load $X(t+n)$.

This AR method was first reported by Akaike [1]. More details can be found in [32], [33]. Our new methods in Algorithms 2 and 3 will use AR method as a kernel computation. The AR model is mostly applied to predict lighter workload with at most a few minutes of execution time. AR method is static with a fixed confidence window that does not change with workload.

4.3 Confidence Window Generation

We use Kalman filters from the Matlab to reduce the measurement errors in the load index $X(t)$ at each time step t. Kalman filter [23] applies feedback control to estimate a process. First, the filter estimates the process state at a future time. Second, it obtains feedback from the measurements. Our work applies Kalman filter to minimize the measurement errors at a series of the load indexes. The purpose is to improve the prediction accuracy.

AR(p) model predicts the future workload status by using point values measured in the past. In practice, point value is accurate only for a short time frame. Another type of load values is referred as *interval values.* The interval values provide an estimate of the load variation in a given time frame. The HModel for n-step-ahead workload prediction is specified in Algorithm 2.

```
Algorithm 2: Generation of Confidence Window (HModel)
Input: AR model parameter p, measured workload index X(t),
Lookahead span n, historical interval w
Output: Confidence window {X_l(t+n), X_u(t+n)} for n-step
        lookahead prediction
Procedure:
1.  Produce the refined workload index X(t);
      // using Kalman filter on workload index X(t) //
2.  Generate smoothed workload index X(t);
      // using Savitzky-Golay filter on refined workload index X(t)//
3.  Predict workload index series [X(t+1),.., X(t+n)]
      //using Algorithm 1 on smoothed loads [X(t-p),.., X(t)] //
4.      forall i =1 to n
5.          Form A(i) = smoothed load index [X(t-w+1),.. X(t)]
            followed by predicted load indices [X(t+1),.., X(t+i)]
6.          Compute confidence window {X_l(t+i), X_u(t+i)}
            of A(i) to yield A_l(i) = X_l(t+i) and A_u(i) = X_u(t+i)
7.      endforall
8.  Generate smoothed A_ls, A_us //using Savitzky-Golay filter//
9.  Generate X_l(t+n) = A_ls(n), the lower end of confidence window
    Generate X_u(t+n) = A_us(n), the upper end of confidence window
```

Two linear arrays $A_l(i)$ and $A_u(i)$ are used to represent the lower bound and upper bound of confidence window values. The confidence range is specified by a pair of two point values $\{X_l(t), X_u(t)\}$. After we compute the n-step lookahead confidence window, we use Savitzky-Golay filter to smooth the confidence windows.

The detailed steps in Algorithms 2 are illustrated in Fig. 5. Fig. 5a shows the load processing at time t. The confidence windows estimated from beginning to n-step lookahead are shown in Fig. 5b. The smoothing decides the final confidence window illustrated in Fig. 5c.

5 Mean-Square Error Analysis

To analyze the accuracy of a prediction method, we evaluate below the HModel prediction accuracy under three workload traces. The evaluation is carried out by the impacts of different trace time span p in the kernel AR(p) computation (Algorithm 1) against the historical interval w used.

These two parameters $\{p, w\}$ have major impact on prediction accuracy of HModel. The optimal choice of the widow size w varies with the workload. The track span p has less impact on the prediction error. Fixed parameter values in HModel may not predict accurately as we want. We analyze the effects of historical interval w on the confidence window.

5.1 Mean-Square Error in Prediction

In point value prediction, error means the difference between true workload data and predicted point value. We define the error function $Error(t)$ at time t for a window-based prediction scheme in (3), where $X(t)$ is the real load index. The error is zero if it is inside the confidence window. Otherwise, the error is visible:

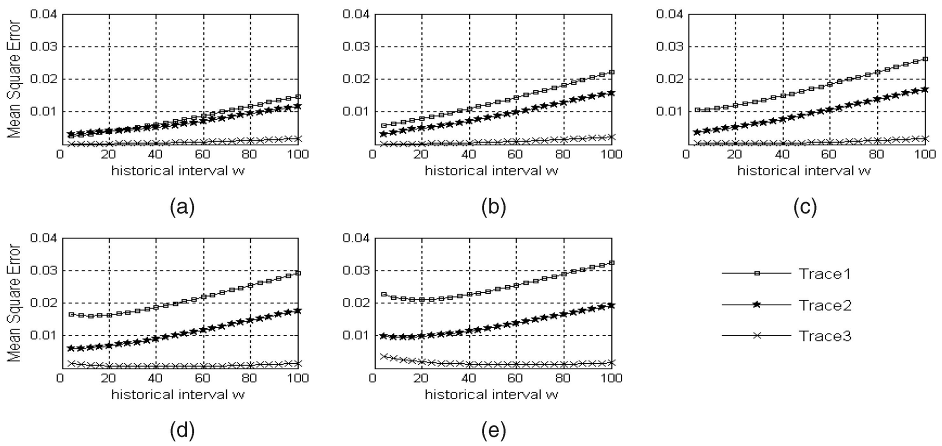


Fig. 6. Impact of historical interval (w) and lookahead span (n) on the MSE measured in the HModel on the AuverGrid for three load traces in Table 3. (a) $n = 10$. (b) $n = 20$. (c) $n = 30$. (d) $n = 40$. (e) $n = 50$.

$$\boldsymbol{Error(t)} = \begin{cases} 0 \text{ if } \boldsymbol{X_l(t) \leq X(t) \leq X_u(t)} \\ \mathrm{Min}\{|\mathrm{X(t)} - \mathrm{X}_l(t)|,\ |\mathrm{X(t)} - \mathrm{X}_u(t)|\} \end{cases} \quad (3)$$

The MSE is defined by taking the average of the square of the ***Error(i)*** function over t time steps from the beginning of the measurement:

$$\boldsymbol{MSE} = \frac{1}{t+1}\sum_{1}^{t}[\boldsymbol{Error(i)}]^2. \quad (4)$$

But the results will be plotted in absolute percentage of error. For example, instead of showing an MSE of $(0.02)^2$, we plot the absolute prediction error as 0.0004. This reflects the average behavior of the entire time interval. Different historical interval sizes w applied in the HModel will impact the prediction MSE and the confidence window used.

Unlike point value prediction, interval-based prediction scheme has lower MSE using confidence window. According to an estimation by Schopf and Berman [34], large sample size of load index $X(t)$ is modeled by a normal distribution. From our experience, for the small sample size ($n < 30$) data, t-distribution is more appropriate. Let $\bar{X}$ be *mean value*, and σ be *standard deviation* of these distributions. The following equations can used to calculate these two parameters, given the values of w and n:

$$\overline{X} = \frac{1}{w+n}\sum_{i=t-w+1}^{t+n} X(i), \quad (5)$$

$$\sigma = \sqrt{\frac{1}{w+n-1}\sum_{i=t-w+1}^{t+n}(X(i) - \overline{X})^2}. \quad (6)$$

The confidence window $C(t+n)$ is characterized as $\overline{X} \pm L/2$, where L is the length of the confidence window predicted for time $t+n$. Throughout our trace experiments, we have applied a default value of $\varepsilon = 5\%$. The following theorem shows the existence of a confidence window $C(t+n)$ centered around the mean value $\bar{X}$, which is guaranteed within the error tolerance threshold ε:

Theorem 1. *Given the Normal distribution or the t-distribution of the load process $X(t)$ being predicted with a historical interval w and lookahead time span n and an error tolerance threshold ε. We can choose the window length L as follows, which is always upper bounded by ε:*

$$L = 2\sigma q/(w+n)^{1/2} \leq \varepsilon, \quad (7)$$

where q is the quantile of the Normal distribution for large lookahead span $n > 30$ steps (150 minutes or greater) or for the t-distribution within $n \leq 30$ steps. Recall that each time step was 5 minutes in all of our trace experiments.

Proof. The confidence window L has a length calculated with (7). What we need is to prove that it can be made less than ε. By definition of standard deviation σ, we must choose a historical interval w that is lower bounded as follows: $w \geq (2\sigma q/\varepsilon)^2 - \mathrm{n}$. This implies that $(w+n)^{1/2} \geq (2\sigma q/\varepsilon)$. Thus, the upper bound on L is proved by rearranging the above inequality. □

From Theorem 1, for any tolerance threshold ε, there always exists w that can make the confidence window upper bounded by the threshold ε. Therefore, we can minimize the prediction MSE by choosing w as such to satisfy Theorem 1.

In general, we should use smaller historical interval w, when n has smaller or moderate value (say, below 30 time steps). The smaller is the confidence window used, the higher the chance to aid task scheduling or load balancing in a production grid computing environment.

5.2 Impact of Historical Interval Size

In Fig. 6, we report the impact of changing the key parameters w, n on the MSE using traces in Table 3. The MSE increases with window size steadily for all five cases shown. Trace 1 has the highest MSE for having sudden level changes. Trace 3 has very small MSE for having only gradual level changes. Trace 2 sits in the middle for having sudden fluctuations around a steady level.

The MSE gaps among the three traces widen as the historical interval increases. It is clear that HModel prefers to apply smaller w on all load traces when lookahead span n is smaller than 30 in Figs. 6a, 6b, and 6c. In using long lookahead spans ($n > 30$) as shown in Figs. 6d and 6e, the impact of w becomes flatter and the better choice of w is to

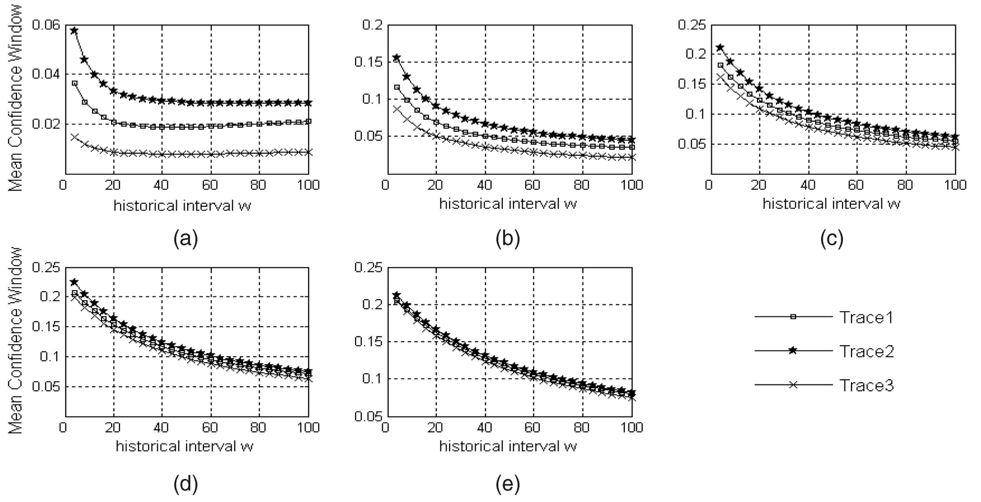


Fig. 7. Impact of historical interval (w) and lookahead span (n) on the mean confidence window used in HModel on the AuverGrid under three load traces given in Table 3. (a) $n = 10$. (b) $n = 20$. (c) $n = 30$. (d) $n = 40$. (e) $n = 50$.

achieve least MSE, which may vary among different load traces, as one sees the big gaps among the three traces in Figs. 6d and 6e.

Fig. 7 reports the mean confidence window $\bar{X}$ used in three load traces using the HModel to predict workload. It is obvious that when n is smaller or moderate value (e.g., 10 or less) as shown in Figs. 7a, 7b, and 7c, we set a small tolerance threshold (ε), such as 0.03 in Fig. 7a and 0.08 in Fig. 7b. With fixed mean confidence window, the better choice of w in using the HModel on the three traces is quite different in magnitude as seen in Figs. 7a, 7b, and 7c.

On the contrary, the impacts of w on prediction MSE and confidence window in HModel are reversed when n is large. Figs. 7d and 7e demonstrate the mean confidence window of HModel for large lookahead span, such as n greater than 30. The variation of optimal w follows the same trend in the three traces. Considering both prediction MSE and confidence window, better choice of w varies with load patterns, independent of the lookahead span n.

6 Adaptive Prediction Scheme (AHModel)

We propose an AHModel by implementing the HModel in Algorithm 2 using a historical interval, which is dynamically trained and validated from past workload series. Two adaptations are integrated here. One is the *mean adaptation* and the other is the historical *interval adaptation.* This will capture the workload changes dynamically. For simplicity, we assume white noises in the embedded AR(p) kernel computations.

The mean value of historical workload is kept in AR(p) as an internal state. We first compute the mean of load indices $\{X(t-p), X(t-p+1), \ldots X(t-1), X(t)\}$. Then, we subtract the mean from all input values. Finally, the future workload value is predicted by adding the white noise with predicted load state.

6.1 Adaptive Choice of Historical Interval

The historical interval w should be selected to correlate future workload with historical workload. Large w means that workload is stable for a long time, and it has more possibility to keep the stable in future. The HModel method predicts future workload effectively only if it has a stable workload pattern. From traces on AvuerGrid [3] and Grid5000 [18], we see the sharp variations on a daily basis.

HModel applies a fixed historical interval w, which cannot capture the dynamic workload changes. Therefore, we develop an adaptive method to adjust the historical interval w, as workload fluctuates. We assume that the workload is periodically stable. We apply both MSE adaptation and window control to form the AHModel.

By periodical checking of prediction MSE and its window under different historical intervals, the AHModel applies a suboptimized historical interval value. This leads to a minimum MSE with an acceptable mean confidence window. We introduce a notion of *tolerance threshold* (ε) to limit the confidence window. First, the historical interval w must keep the mean of the confidence window below a threshold ε. Also we should choose w to achieve a minimum MSE.

Fig. 8 shows the variation of the confidence window selection in AuverGrid. At the upper part of the plotted curves, we show the successive confidence widows, which are upper bounded by the dashed curve and lower bounded by the dotted curve. The actual workload is shown by the solid curve in the middle.

The suboptimal historical interval varies in quantum jumps shown by the thick flat lines at the lower part of the plot. From time instances 61 to 120, the workload fluctuates widely. The suboptimal historical interval is chosen rather small. From time instances 121 to 180, the workload becomes relatively stable. Thus, a longer historical interval can be chosen without hurting the MSE.

6.2 Training and Validation of Historical Interval

From the results plotted in Figs. 6 and 7, we realize that the proper choice of the historical interval w does make a big difference in reducing the MSE under workload fluctuation. To achieve an optimal choice of w is an NP-hard problem because too many conflicting factors are involved. We pursue for a suboptimal solution by historical interval training. We

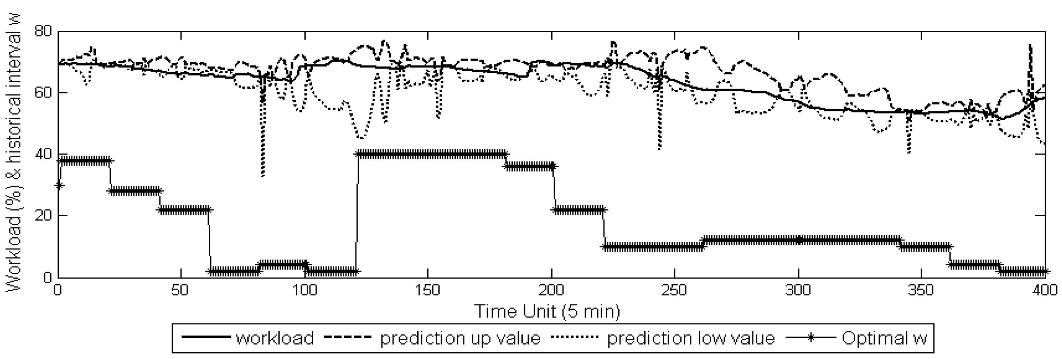


Fig. 8. Successive confidence widows are upper bounded by dashed curve and lower bounded by dotted curve. The actual workload is shown by solid curve. The suboptimal choice of the historical interval is shown by thick flat line that quantum jumps from time to time as the workload fluctuates.

introduce below a training process in Algorithm 3. The purpose is to choose a suboptimal historical interval w.

The training period u can be simply set equal to the lookahead span n. This means that we use the past u input load indices $\{X(t-u-v+1), \ldots, X(t-1), X(t)\}$ from the past steps to predict n future indices $\{X(t+1), X(t+2), \ldots, X(t+n)\}$ with n steps ahead of the current time t. Algorithm 3 shows the training process steps of the AHModel method. This is a pseudocode aided by inline comments to illustrate the training process.

Let $D = \{w_i \; for \; i = 1, 2, \ldots, k\}$ be a given set of preselected historical interval sizes. These candidate historical intervals are obtained from past prediction experiences on a given grid platform. Thus, the set may vary from platform to platform. Let S be the load series $\{X(t-u-v+1), \ldots, X(t-u-1), X(t-u)\}$ used for historical interval training and H be the load series $\{X(t-v+1) \ldots, X(t-1), X(t)\}$ used for window validation.

ALGORITHM 3: TRAINING AND VALIDATION OF HISTORICAL INTERVAL FOR USE IN THE AHMODEL PREDICTION

Input: Historical workload series $\{X(t\text{-}u\text{-}v\text{+}1),\ldots, X(t\text{-}v\text{+}1),\ldots, X(t\text{-}1), X(t)\}$, lookahead span n, training period u, validation period v, threshold error ε. A set $D = \{ w_i$ for $i = 1, 2, \ldots, k\}$ of candidate historical interval sizes.
Output: Historical interval w trained to minimize the MSE in workload prediction for using the AHModel
Procedure:

```
1.  Initialize $A_w$ to hold all acceptable interval sizes
2.  for each $w_i$ in $D = \{ w_i$ for $i = 1, 2, \ldots, k\}$
3.      Compute the mean $\bar{X}$ of the confidence window
        using this $w_i$ in Algorithm 2 over the load series
        $S = \{X(t\text{-}u\text{-}v\text{+}1),\ldots, X(t\text{-}u\text{-}1), X(t\text{-}u)\}$
4.      If ($\bar{X} < \varepsilon$), then store $w_i$ in $A_w$.
5.      else apply the next $w_{i+1}$ in $D$ to generate a new $\bar{X}$
          by repeating Step 3.
6.  endfor
7.  If none of the candidate window size in set $D$ can produce
    a mean value lower than $\varepsilon$, then choose w in set $D$ which
    achieve minimum mean value of the confidence window.
8.  else validate the historical window over the past $v$ load
    series $H = \{X(t\text{-}v\text{+}1),\ldots, X(t\text{-}1), X(t)\}$ to choose w in $A_w$
    which minimizes the MSE on the validation series $H$.
9.  Output the window w so trained and validated
```

The training and validation process is specified in Algorithm 3. The AHModel is implemented with simultaneous application of Algorithms 2 and 3. In other words, the AHModel is built on top of the HModel, which is, in turn, built with AR(p) model as the kernel computation at the innermost loop.

The AHModel prediction process is illustrated in Fig. 9a. We validate the choices of historical interval w in every v step, where v is the validation period. We apply the HModel with a different w over past load indices $\{X(t-u-v+1), \ldots, X(t-u)\}$. The process of the AHModel is shown in Figs. 9a, 9b, and 9c. The prediction is validated using load indices $\{X(t-v+1), \ldots, X(t)\}$. To implement n-step lookahead prediction, we apply HModel to the next u load indices as shown in Fig. 9c.

6.3 Suboptimality of Trained Historical Interval

The historical interval w predicts the future workload from checking the historical workload. If the workload is unstable and fluctuates widely, w should be chosen small. This is reasonable because when workload varies rapidly, future

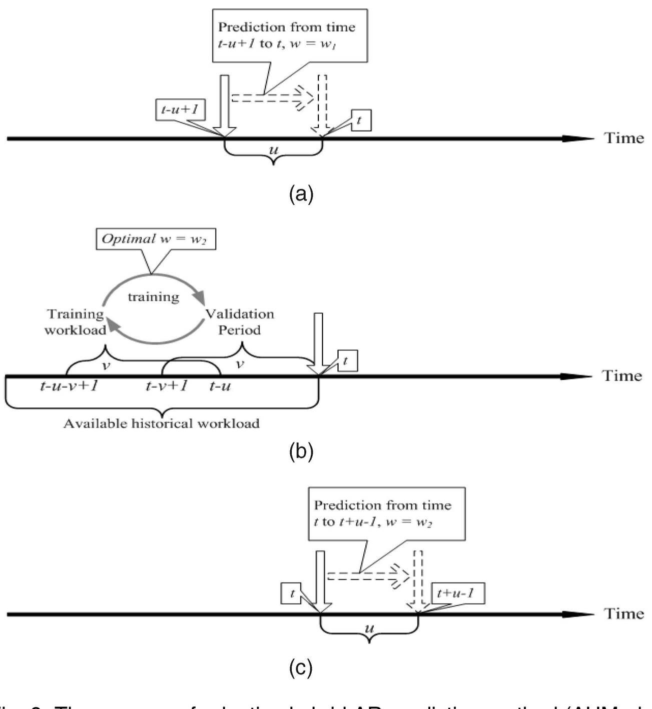


Fig. 9. The process of adaptive hybrid AR prediction method (AHModel using Algorithms 2 and 3). (a) Prediction by using HModel. Update at time $t-u+1$. (b) Adaptation triggered at time t. (c) Prediction by using HModel. Next update at time t.

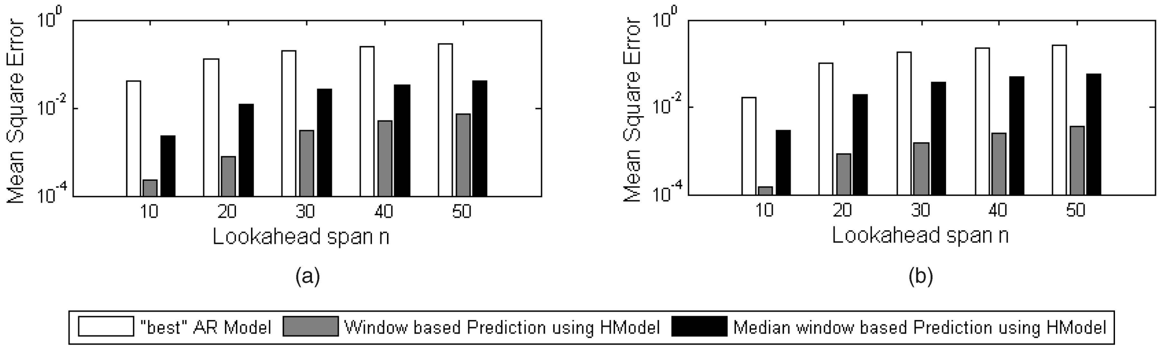


Fig. 10. Average prediction MSE in using the HModel, compared with the best results in using the AR model. (a) AuverGrid. (b) Grid'5000.

workload is not likely to rely on past historical workload. The MSE of HModel increases with the increase of w.

Narrower confidence window like 5 percent produces larger MSE than the wider ones. If we set a larger threshold ε on the confidence window, we may find a suboptimal historical interval size to minimize the MSE in using the HModel. This relaxation enables the AHModel method to cover wider load variations. The following theorem proves the suboptimality of the historical interval w generated by Algorithm 3:

Theorem 2. *The historical interval w generated in Algorithm 3 is suboptimal in that it results in minimal MSE in minimized confidence windows, among all admissible historical window sizes $D = \{w_i \; for \; i = 1, 2, \ldots, k\}$.*

Proof. Algorithm 3 leads to two possible choices of the historical interval w from set $\boldsymbol{D}$. The first choice is when we succeed to find a $\boldsymbol{w}$ in the ***for*** loop (Steps 3-6) that leads to a acceptable mean $\bar{X}$ of confidence window and minimized MSE. The second choice is to choose the minimum among all possible choices $\{w_i \; for \; i = 1, 2, \ldots, k\}$. Either choice will lead to a minimal mean $\bar{X}$ of the confidence window. Thus, the interval size w so obtained becomes suboptimal in this sense. □

7 Prediction Trace Results on the AuverGrid and Grid5000

HModel implements only the AR(p) adaptation. AHModel adjusts the historical interval w periodically. We report below the experimental results in using the HModel and AHModel.

7.1 Performance Results on Using HModel

We evaluate HModel using workload traces from the AuverGrid and Grid'5000. The results are plotted in Fig. 10. The HModel results are obtained using two different value types: one is governed by the confidence window and the other is selected by the median of the confidence window. We compare their MSE results with the best AR(p) model by choosing the best case out of different values of the traceback span $p = 4, 8, 12, \ldots, 64$.

We applied 12 traces from AuverGrid and 10 traces from Grid5000. As shown in Fig. 10, the HModel outperforms the best AR(p) model with the lowest MSE on both AuverGrid and Grid5000 load traces. The MSE of HModel is 98.5 and 98.9 percent lower than that of the best AR results on both grid platforms. The MSE in using the median of the confidence window is 88.7 and 79.9 percent lower than that of the best AR results in these experiments.

7.2 Validation Period in AHModel Experiments

To predict n future workload values, the AHModel uses $u = n$ historical load to yield a suboptimal historical interval w. We introduce a validation mechanism to make sure that the modified historical interval is sufficient to trace back. The time span v needed to validate the suboptimality of w is called the *validation period.* AHModel applies the latest v historical workload indices to validate the prediction accuracy. The idea is to train from different historical interval sizes.

Fig. 11 shows the impacts of validation period v on the accuracy (MSE) and the mean confidence applied with a lookahead span of $n = 50$ steps. The workload traces cover 8,000 samples from AuverGrid and Grid'5000. As the validation period v increases, Fig. 11a shows that the MSE drops steadily from 0.92 to 0.82 percent on the AuverGrid. The Grid5000 achieves the lowest MSE of 0.56 percent as v increases to 50 steps in Fig. 11b. When v equals $n + 60$, the AHModel results in the highest accuracy on both grid platforms. So we set $\mathrm{v} = \mathrm{n} + 60$ in all load prediction experiments.

In Figs. 11c and 11d, we plot the mean confidence window size as a function of the validation period. On the AuverGrid, the mean window size drops to the lowest value of 12.25 percent as v increases to 25 steps. The mean confidence then increases steadily to 13 percent as v increases to 150 steps. On the Grid5000, the mean window is lowered to 11.85 percent as v approaches 50 steps. Then the mean value stays at this level with almost no increase as v goes beyond 50 steps.

7.3 Performance of Using AHModel for Prediction

We report experimental results on implementing the AHModel in both AuverGrid and Grid5000. The average MSEs of AHModel in two implementations are compared with the best AR model in Fig. 12. The AR method leads to MSE in the range 2-30 percent on both machines. The window-based AHModel has the lowest MSEs between 0.1 and 0.7 percent.

The median window has MSE in 0.2-0.9 percent. In the best case, the AHModel outperforms the best AR method by a factor of 300 times. For $\mathrm{n} = 10$ steps, the window-based AHModel performs 50 percent better than the median-based scheme on the AuverGrid, while 20 percent better on the Grid5000. For a 50-step lookahead, the window-based scheme has MSE below 1 percent, while the best AR method may introduce 30 percent MSE

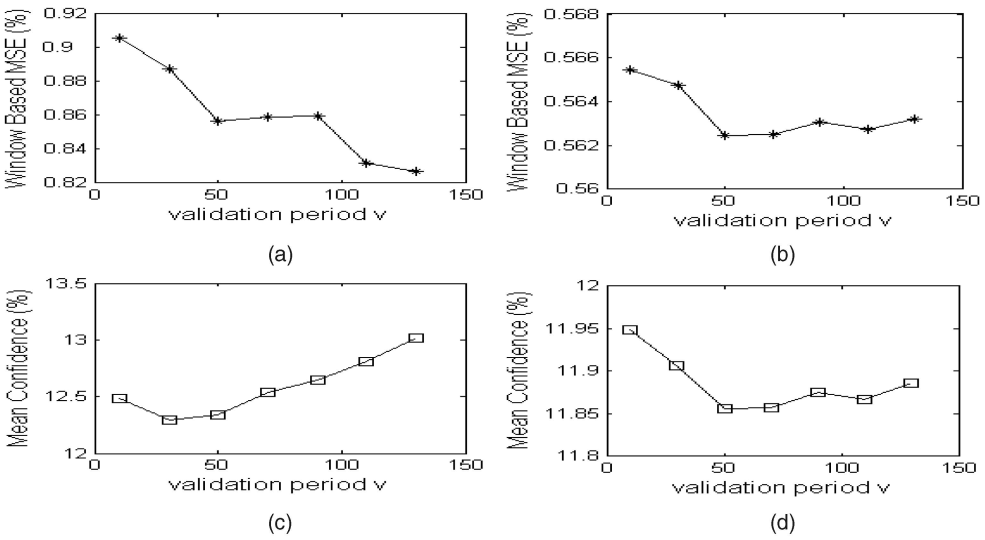


Fig. 11. Variation of the prediction MSE and mean confidence windows used in implementing the AHModel on the AuverGrid and Grid5000 platforms. (a) AuverGrid. $n = 50$. (b) Grid5000. $n = 50$ (c) AuverGrid. $n = 50$ (d) Grid5000. $n = 50$.

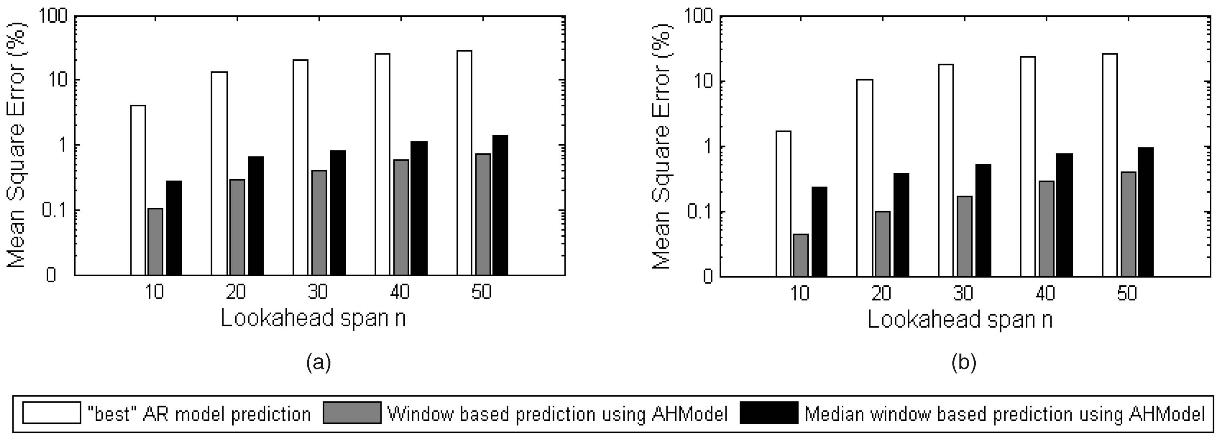


Fig. 12. Mean-square errors of two implementations of AHModel, compared with the best MSE by using the AR model. (a) AuverGrid. (b) Grid'5000.

Every n-step lookahead prediction in using the AHModel requires few hundreds of input load values. The execution time of one prediction is done in a few milliseconds. This computational cost is rather low, and thus, applicable in real practices.

7.4 Relative Performance of AHModel and HModel

In this section, we reveal the relative performance of the adaptive prediction scheme AHModel, compared with the hybrid prediction scheme HModel. To simplify the comparison, both schemes apply the median of the confidence window in making predictions. The results are shown in Figs. 13a and 13b, based on trace experiments on the AuverGrid and Grid5000, respectively.

We use the first 1,000 sample loads of each month to train the historical interval w. The suboptimal w becomes the initial input of the HModel to process the rest of the data trace. Overall, the AHModel outperforms the HModel by a factor in the range 20-700 percent. On the AuverGrid, the MSE of AHModel is below 1.2 percent, compared with 4 percent in using the HModel for a long lookahead span of 50 steps (250 minutes). On the Grid5000, the same MSE gap is even greater. The AHModel has an MSE 0.8 percent, while the HModel leads to an MSE of 5.9 percent, about 7 times greater.

This clearly shows that the adaptive model predicts much more accurately when large workload with long execution time is expected. For short time workload with 10 lookahead steps (50 minutes), the two prediction models perform about the same. Overall, we declare that the adaptive AHModel is more suitable for use in predicting large workload on a computational grid with more than 20 lookahead steps (from 100 to 250 minutes). The HModel is cheaper to implement and more suitable for predicting smaller workload in fewer lookahead steps.

7.5 Confidence Window Distribution

In this section, we reveal the distribution of the confidence window, when the HModel and AHModel were implemented on AuverGrid and Grid5000. The confidence window can be characterized by a *mean confidence* and a *standard deviation* (STD). Based on our load tracing experiments, we report in Fig. 14 the mean and STD for five lookahead spans from $n = 10$ steps with less than 50 minutes to 50 steps with more than 4 hours.

Figs. 14a and 14b compare the mean confidence windows used in HModel and AHModel over the two

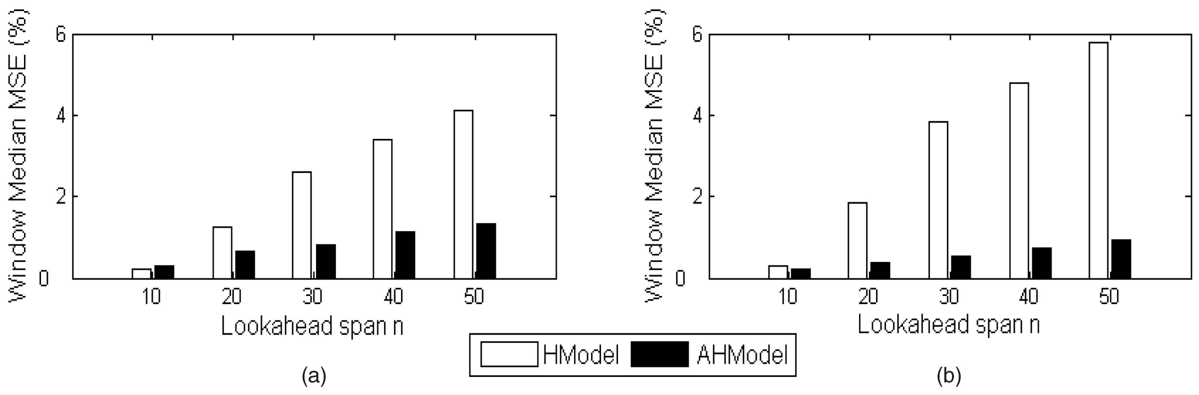


Fig. 13. Mean-square error of using AHModel and HModel with median of the confidence window for workload prediction. (a) AuverGrid. (b) Grid'5000.

French grid platforms. The mean confidence used in AHModel was a constant 7 percent for all lookahead spans. On the AuverGrid, the AHModel always applies shorter mean confidence than that used by HModel as shown in Fig. 14a. On the Grid5000, the mean confidence converges for both AHModel and HModel as n increases to 30 steps or greater as shown in Fig. 14b.

The standard deviation of the confidence window becomes rather small (about 2 percent) for both models, when large lookahead span is applied as shown in Figs. 14c and 14d. In summary, we see a mean confidence around 7 percent with a STD of less than 2 percent when $n = 50$-step lookahead prediction scheme is applied.

8 Conclusions and Further Work

The traditional point value prediction strategy, such as the AR method, is inadequate to predict workload in computational grids because they cannot cover load variations in a long execution environment. We developed two new lookahead workload prediction schemes for assessing long-term grid performance under fluctuating loads.

We obtained encouraging results to demonstrate the effectiveness of our new prediction schemes. The main idea is to extend the point value to a confidence window, which is dynamically adjusted against load variations. We make adaptation by adjusting the historical interval w used. This is the key to enable adaptive workload prediction to match with historical load variations.

As the lookahead span increases from 10 to 50 steps (5 minutes per step), AHModel predicts the grid workload with an MSE of 0.04-0.73 percent, compared with a much greater MSE of 2.54-30.2 percent in using the point value AR prediction method.

The significant gain in prediction accuracy makes the new AHModel model very attractive to predict Grid

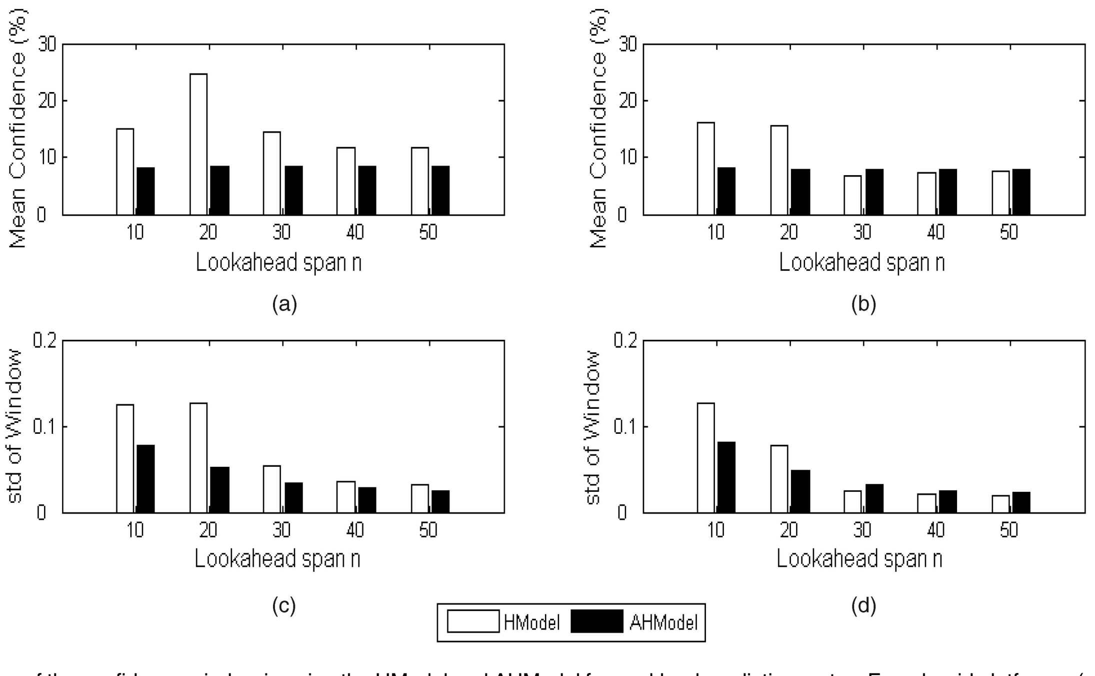


Fig. 14. Variation of the confidence window in using the HModel and AHModel for workload prediction on two French grid platforms: (a) and (b) Mean confidence window. (c) and (d) Standard deviation (STD) of the confidence window applied in Grid trace experiments. (a) AuverGrid. (b) Grid'5000. (c) AuverGrid. (d) Grid'5000.

performance. The model was proved especially effective to large workload that demands very long execution time, such as exceeding 4 hours on the Grid5000 with more than 5,000 processors. With minor changes of some key system parameters, the AHModel can be applied to other computational grids as well.

8.1 Summary of Research Contributions

Our research contributions are summarized in five technical aspects as follows:

1. ***Introduction of the confidence window approach:*** The confidence window allows the workload prediction to be conducted within a tolerable range of load index values. This interval-based prediction scheme leads to higher accuracy and enables the adaptive prediction method.
2. ***Kalmam filter and Savitzky-Golay filter are proved effective tools for Grid workload prediction:*** For the first time, these two Matlab tools are used to reduce measured load index errors and smooth the predicted load indexes. The effectiveness was proved by CPU load trace experiments on the AuverGrid and Grid5000.
3. ***A hybrid prediction method (HModel)***: This prediction scheme extends the static AR *method* to work in a confidence window, dynamically.
4. ***An adaptive prediction method (AHModel):*** This scheme solves the load fluctuation problem in large-scale grids. When the grid load changes rapidly, the prediction confidence window also changes. The adaptive scheme improves the prediction accuracy significantly.
5. ***Benchmark trace programs tested on AuverGrid and Grid5000:*** The prediction results are obtained from trace experiments on AuverGrid and Grid'5000 in France. These trace experiments can be extended to evaluate other computational grids with only minor modification in system and workload parameters.

8.2 Suggestions for Further Work

In the future, we plan to apply the AHModel to larger infrastructure like EGEE. We suggest to extend the above work in three aspects. These will generate useful tools for grid performance assessment over real benchmark applications.

1. ***Extending the adaptive prediction scheme to multiple levels of adaptation:*** A multilevel adaptation scheme can predict the performance of very large-scale grids that are hierarchically constructed. This must address the grid scalability and reliability issues: When a grid platform grows hierarchically by merging with other grids, one can consider adding another level of adaptation. This opens up a wide open area for further research.
2. ***Developing benchmarks for grid performance evaluation:*** The grid community needs to develop a standard set of application programs for performance evaluation or workload prediction purposes. The trace programs we have experimented are not representative to cover the kernel Grid applications.
3. ***Building production systems based on the HModel and AHModel in real-life grid applications:*** Our experimental software could be prototyped toward this end. We suggest to perfect the Matlab tools (Kalman filter and Savitzky-Golay filter) for the Grid community to use in specific grid workload prediction.

ACKNOWLEDGMENTS

This research work was supported by the National Science Foundation of China (NSFC) under Grant Nos. 60673152, 60773145 and by China's National Basic Research (973) Program under Grant No. 2004CB318000. The authors would like to thank Dr Alexandru Iosup, Dr. Emmanuel Medernach, Dr. Frank Cappello, and Dr. Olivier Richard for their assistance to obtain the Workload Archive Traces of AuverGrid and Grid5000. Dr. Hwang acknowledges the support from the EMC Endowment during his sabbatical visits of Tsinghua University in 2008-2009.

REFERENCES

[1] H. Akaike, "Fitting Autoregressive Models for Prediction," *Annals of the Inst. of Statistical Math.*, vol. 21, no. 1, pp. 243-247, Dec. 1969.

[2] S. Akioka and Y. Muraoka, "Extended Forecast of CPU and Network Load on Computational Grid," *Proc. IEEE Int'l Symp. Cluster Computing and Grid (CCGrid '04)*, pp. 765-772, 2004.

[3] AuverGrid Workload Report, http://gwa.ewi.tudelft.nl/pmwiki/reports/gwa-t-4/trace_analysis_report.html, 2009.

[4] *Grid Computing: Making the Global Infrastructure a Reality*, F. Berman, G.C. Fox, and T. Hey, eds. Wiley, 2003.

[5] L. Carrington, A. Snavely, and N. Wolter, "A Performance Prediction Framework for Scientific Applications," *Future Generation Computer Systems*, vol. 22, pp. 336-346, 2006.

[6] W. Cirne and F. Berman, "A Comprehensive Model of the Supercomputer Workload," *Proc. IEEE Fourth Ann. Workshop Workloads Characterization*, 2001.

[7] A.A. Chien, X. Sun, and Z. Xu, "Viewpoints on Grid Standards," *J. Computer Science and Technology*, vol. 20, no. 1, 2005.

[8] ChinaGrid Website, http://www.chinagrid.edu.cn, 2006.

[9] M.J. Clement and M.J. Quinn, "Analytical Performance Prediction on Multicomputers," *J. Supercomputing*, pp. 886-894, 1993.

[10] P.A. Dinda and D.R. O'Hallaron, "Host Load Prediction Using Linear Models," *Cluster Computing*, vol. 3, pp. 265-280, 2000.

[11] *Sourcebook of Parallel Computing*, J. Dongarra, I. Fister, G. Fox, W. Gropp, K. Kennedy, L. Torczon, and A. White, eds. Kaufman Publishers, 2002.

[12] D.G. Feitelson, *Workload Modeling for Computer Systems Performance Evaluation, Draft Version 0.7*, Hebrew Univ. of Jerusalem, 2006.

[13] I. Foster and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure.* Kaufmann Publishers, 2004.

[14] G. Fox, D. Gannon, and M. Thomas, "Overview of Grid Computing environments," *Grid Computing*, F. Berman, G. Fox, and T. Hey, eds., Chapter 20, Wiley, 2003.

[15] L. Gong, S.H. Sun, and E.F. Watson, "Performance Modeling and Prediction of Non-Dedicated Network Computing," *IEEE Trans. Computers*, vol. 51, no. 9, pp. 1041-1055, Sept. 2002.

[16] G.C. Goodwin and K.S. Sin, *Adaptive Filtering Prediction and Control.* Prentice-Hall, 1984.

[17] Grid Workloads Archive, http://gwa.ewi.tudelft.nl/, 2009.

[18] *Grid'5000, ALADDIN-G5K: Ensuring the Development of Grid '5000*, https://www.grid5000.fr/, 2009.

[19] K. Hwang and Z. Xu, *Scalable Parallel Computing.* McGraw-Hill, 1998.

[20] S. Jang, X. Wu, and V. Taylor, "Using Performance Prediction to Allocate Grid Resources," GriPhyN Technical Report 2004-25, pp. 1-11, 2004.

[21] M.A. Iverson, F. Ozguner, and L. Potter, "Statistical Prediction of Task Execution Time through Analytical Benchmarking for Scheduling in a Heterogeneous Environment," *IEEE Trans. Computers*, vol. 48, no. 12, pp. 1374-1379, Dec. 1999.

[22] M. Kalantari and M. Akbari, "Fault-Aware Grid Scheduling Using Performance Prediction by Workload Modeling," *J. Supercomputing*, vol. 46, no. 1, pp. 15-39, Oct. 2008.

[23] R.E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Trans. ASME—J. Basic Eng.*, vol. 82, pp. 35-45, Mar. 1960.

[24] N.H. Kapadia, J.A. Fortes, and C.E. Brodley, "Predictive Application-Performance Modeling in a Computational Grid Environment," *Proc. IEEE Int'l Symp. High-Performance Distributed Computing (HPDC)*, 1999.

[25] H. Li, "Workload Characterization, Modeling, and Prediction in Grid Computing," PhD thesis, ASCI Graduate School, Univ. of Leiden, Jan. 2008.

[26] H. Li and R. Buyya, "Model-Driven Simulation of Grid Scheduling Strategies," *Proc. Third IEEE Int'l Conf. e-Science and Grid Computing (eScience)*, 2007.

[27] J. Liang, K. Nahrstedt, and Y. Zhou, "Adaptive Multi-Resource Prediction in Distributed Resource Sharing Environment," *Proc. IEEE Int'l Symp. Cluster Computing and Grid (CCGrid '04)*, pp. 1-8, 2004.

[28] C. Liu, L. Yang, I. Foster, and D. Angulo, "Design and Evaluation of a Resource Selection Framework for Grid Applications," *Proc. 11th IEEE Int'l Symp. High-Performance Distributed Computing (HPDC '02)*, 2002.

[29] D. Lu, H. Sheng, and P. Dinda, "Size-Based Scheduling Policies with Inaccurate Scheduling Information," *Proc. 12th IEEE Int'l Symp. Modeling, Analysis, and Simulation of Computer and Telecomm. Systems (MASCOTS '04)*, pp. 31-38, 2004.

[30] U. Lublin and D.G. Feitelson, "The Workload on Parallel Supercomputers: Modeling the Characteristics of Rigid Jobs," *J. Parallel and Distributed Computing (JPDC)*, vol. 63, no. 11, pp. 1105-1122, Nov. 2003.

[31] M. Maheswaran, H.J. Siegel, D. Haensgen, and R.F. Freud, "Dynamic Mapping of a Class of Independent Tasks onto Heterogeneous Computing Systems," *J. Parallel and Distributed Computing (JPDC)*, vol. 59, no. 2, pp. 107-131, Feb. 1999.

[32] S.L. Marple Jr., *Digital Spectral Analysis with Applications.* Prentice-Hall, 1987.

[33] S.J. Orfanidis, *Introduction to Signal Processing.* Prentice-Hall, 1996.

[34] J. Schopf and F. Berman, "Performance Prediction in Production Environments," *Proc. 12th Int'l Parallel Processing Symp.*, pp. 647-653, Apr. 1998.

[35] W. Smith, I. Foster, and V. Taylor, "Predicting Application Run Times with Historical Information," *J. Parallel and Distributed Computing (JPDC)*, vol. 64, no. 9, pp. 1007-1016, 2004.

[36] M. Snir and D.A. Bader, "A Framework for Measuring Supercomputer Productivity," *Int'l J. High-Performance Computer Applications*, vol. 18, no. 4, pp. 417-432, 2004.

[37] S. Song, K. Hwang, and Y. Kwok, "Risk-Tolerant Heuristics and Genetic Algorithms for Security-Assured Grid Job Scheduling," *IEEE Trans. Computers*, vol. 55, no. 6, pp. 703-719, June 2006.

[38] V. Taylor, X. Wu, J. Geisler, and R. Stevens, "Using Kernel Computings to Predict Parallel Application Performance," *Proc. Int'l Symp. High-Performance Distributed Computing (HPDC)*, 2002.

[39] G. Welch and G. Bishop, "An Introduction to the Kalman Filter," Univ. of North Carolina at Chapel Hill, 1995.

[40] R. Wolski, "Dynamically Forecasting Network Performance Using the Network Weather Service," *J. Cluster Computing*, vol. 1, pp. 119-132, Jan. 1998.

[41] R. Wolski, N. Spring, and J. Hayes, "Predicting the CPU Availability of Time-Shared Unix Systems," *Proc. Eighth IEEE High Performance Distributed Computing Conf. (HPDC)*, 1999.

[42] Y. Yuan, Y. Wu, G. Yang, and W. Zheng, "Adaptive Hydrid Model for Long-Term Load Prediction in Computational Grid," *Proc. IEEE Eighth Int'l Conf. Cluster Computing and Grid (CCGrid '08)*, pp. 340-347, May 2008.

[43] Z. Xu and K. Hwang, "Early Prediction of MPP Performance: SP2, T3D, and Paragon Experiences," *J. Parallel Computing*, vol. 22, no. 7, pp. 917-942, Oct. 1996.

[44] L. Yang, I. Foster, and J.M. Schopf, "Homeostatic and Tendency-Based CPU Load Predictions," *Proc. Int'l Parallel and Distributed Processing Symp. (IPDPS '03)*, pp. 42-50, 2003.

[45] L. Yang, X. Ma, and F. Muller, "Cross-Platform Performance Prediction of Parallel Applications Using Partial Execution," *Proc. Supercomputing Conf.*, 2005.

[46] Y. Zhang, W. Sun, and Y. Inoguchi, "CPU Load Predictions on the Computational Grid," *Proc. IEEE Sixth Int'l Conf. Cluster Computing and Grid (CCGrid '06)*, May 2006.

[47] J. Zhang and R.J. Figueiredo, "Adaptive Predictor Integration for System Performance Prediction," *Proc. IEEE Int'l Parallel and Distributed Processing Symp. (IPDPS '07)*, Mar. 2007.

Yongwei Wu received the PhD degree in applied mathematics from the Chinese Academy of Sciences in 2002. He is currently an associate professor of computer science and technology at Tsinghua University, Beijing, China. His research interests include grid and cloud computing, distributed processing, and parallel computing. He is a member of the IEEE.

Kai Hwang received the PhD degree from the University of California, Berkeley, in 1972. He is a professor of electrical engineering and computer science at the University of Southern California, Los Angeles. He specializes in computer systems, parallel processing, Internet security, and distributed computing. He is the founding editor-in-chief of the *Journal of Parallel and Distributed Computing* (Elsevier). He is a fellow of the IEEE and the IEEE Computer Society.

Yulai Yuan received the BS degree in computer science and engineering from the Beijing Information Technology Institute in 2005. He is currently working toward the PhD degree in computer science at Tsinghua University. His research interests include distributed systems, performance modeling and prediction, and data replication.

Weimin Zheng received the BS and MS degrees, respectively, in 1970 and 1982 from Tsinghua University, China, where he is currently a professor of computer science and technology. He is the research director of the Institute of High Performance Computing at Tsinghua University, and the managing director of the Chinese Computer Society. His research interests include computer architecture, operating system, storage networks, and distributed computing. He is a member of the IEEE.