A Prediction Method for Job Runtimes on Shared Processors: Survey, Statistical Analysis and New Avenues

Menno Dobber[a], Rob van der Mei[a,b], Ger Koole[a]
[a]Vrije Universiteit, De Boelelaan 1081a, 1081HV Amsterdam, The Netherlands
[b]CWI, Kruislaan 413, 1098SJ Amsterdam, The Netherlands
{amdobber,mei,koole}@few.vu.nl

Abstract

Grid computing is an emerging technology by which huge numbers of processors over the world create a global source of processing power. Their collaboration makes it possible to perform computations that are too extensive to perform on a single processor. On a grid processors may connect and disconnect at any time, and the load on the computers can be highly bursty. Those characteristics raise the need for the development of techniques that make grid applications robust against the dynamics of the grid environment. In particular, applications that use significant amounts of processor power for running jobs need effective predictions of the expected computation times of those jobs on remote hosts. Currently, there are no effective prediction methods available that cope with the ever-changing running times of jobs on a grid environment. Motivated by this, we develop the Dynamic Exponential Smoothing (DES) method to predict running times in a grid environment. The main idea behind DES is that it dynamically adapts its prediction strategy to the height of the fluctuations in those running times. We have performed extensive experiments in a real global-scale grid environment to compare the effectiveness of DES. The results demonstrate that DES strongly and consistently outperforms existing prediction methods.

Keywords: Grid computing, Parallel computing, Forecasting, Prediction methods, Data analysis

1 Introduction

Over the past few decades a lot of effort has been devoted to the design and development of computational grids, consisting of centralized interconnected parallel processors. The operation of those clusters is well-predictable: the processors are homogeneous and a reservation system allocates dedicated processing resources to the jobs. Over the years, clusters have evolved, because of scalability and costs reasons, into grid environments that connect remote processors at a global scale via the Internet. Grids often do not make use of reservation systems and, therefore, grid applications effectively share the processing resources. Grid environments are highly unpredictable in several aspects: the load on processors changes all the time, the processor capacities and link rates are often unknown, and processors may connect and disconnect at any time. Variations in the available resources (e.g., computing power, bandwidth) may lead to a significant increase in the running times of parallel applications. This raises the need for the development of techniques that make applications robust for the continuously changing circumstances of the grid environment. Those techniques strongly rely on the effectiveness of prediction methods: significant speedups can be obtained by good scheduling schemes based on accurate predictions, as is demonstrated in an experimental setting in [2, 4, 13]. Similar observations are made in Dobber et al. [6] where we investigate the impact of fluctuations in processing speeds on running times of grid applications. There we show that significant reduction in the running times can be obtained by implementing dynamic load balancing schemes.
In this paper, we focus on the development of a new method to predict the running times of jobs on shared processors. To this end, we perform extensive data analysis of job running times generated from different grid nodes. Next, we analyze several existing methods to predict running times of

jobs, and analyze their weak and strong points. Based on these insights, we develop a new prediction method, called Dynamic Exponential Smoothing (DES). Extensive experimental results demonstrate that predictions from the DES strongly outperforms the existing methods. Partial and preliminary versions of the results presented in this paper have been presented in [8] and [9].

In the literature, a significant number of papers have been devoted to prediction methods. The prediction schemes can be broadly classified into two main categories: linear and non-linear predictors. For grid environments, several linear models have been proven successful for predicting future properties of a grid: Exponential Smoothing (ES) for predicting running times of jobs ([7, 17]), Autoregressive models (AR) for predicting the load [5] and network traffic ([16]), Linear Regression (LR) for predicting total running times of parallel applications ([12]), and a tendency-based predictor ([25]), also for predicting the load. Furthermore, the Network Weather Service (NWS) prediction algorithm that selects between different linear predictors (e.g. [22, 23]) predicts different grid entities. Wolski et al. [24] also focus on the prediction of load, and conclude that the measurement errors swamp the prediction errors; in this paper, however, the focus is on predictions of the job-running times, where the measurement errors are typically much lower than for load predictions. Moreover, in other research areas interesting predictors have been developed with possible applications in grid environments. As described in [18], several adaptive ES-based methods have been developed, which have shown to be very accurate in predicting, for example, economic and societal quantities. Currently, the applicability in grid environments of non-linear prediction models, like Neural Networks, has not been investigated.

A key requirement of applications running in a grid environment is robustness against the dynamically changing circumstances. For example, an effective means to deal with changing processor speeds is to implement dynamic load balancing schemes [6]. Prediction methods for running times used to trigger load re-balancing actions should be simple and fast. Moreover, since the monitoring capabilities in a real grid environment will be at best limited, prediction methods should be based on only a small number of measurement parameters. To predict the running times on the basis of the measured load, Dinda et al. [3, 5] analyze the relation between processor load and the running times. In [3] a threshold autoregressive model is used to specifically to deal with the non-stationarity of the signal. They find that in practice it is hard to predict the running times by the load, because many other factors (e.g., memory space) also have an influence on the running times, and those are hard, or even technically impossible, to gather in practice. The prediction of running times is even further complicated by the fact that in a real grid environment even the load (i.e., CPU utilization) often can not be measured at all. To circumvent this problem, we focus on methods to predict future running times *only* on the basis of the *past running times* of jobs, not requiring any additional - and possibly unavailable - measurement data.

In this paper, we develop a new method that effectively predicts the running times of successive jobs on nodes in a grid environment. To this end, we make use of the Planetlab [1], a global-scale collaborating testbed environment. We proceed along the following steps. First, in Section 2 we perform extensive data analysis with six datasets of job running times, generated from four different Planetlab nodes. The results show that the characteristics of the running times vary strongly among the different nodes, and moreover, that the characteristics (e.g., mean, burstiness) for each given node may differ strongly over different time periods. Second, to cope with these complexities in Section 3 we investigate a broad set of predictors (e.g., NWS, AR-methods, Whybark, STES predictors), and identify the strengths and weaknesses of those predictors on the basis of both theoretical analysis and extensive statistical analysis of experimental testbed data. The analysis reveals that the main sources of inaccuracy of these existing methods are (1) over reaction to sudden peaks in the running times, and (2) delayed reaction to "level switches". Level switches are significant changes in the height of the running times that hold for at least three measurements. Third, in Section 4, based on these observations we develop a new prediction method, called Dynamic Exponential Smoothing (DES). The main idea behind DES is that it uses ES where the interpolating factor α has a number of levels that can be dynamically adapted to the height of the outliers in the data. Fourth, in Section 5 we compare the accuracy of the predictions resulting from DES to that of the other prediction methods. For the comparison we use 45 datasets of 18 different Planetlab nodes. The results show that DES strongly outperforms the existing methods in the vast majority of the datasets. Finally, in Section 6 we address a number of topics for further research.

2 Data Analysis

In this section we analyze the statistical properties of six representative and real datasets of running times of jobs on shared processors. In 2.1 we describe the data collection procedure. In 2.2 we discuss the statistical properties of the data.

2.1 Data collection

To perform experiments on shared processors in a real grid environment, we used Planetlab [1], a commonly used grid test bed environment shared by many users. At the time of our experiments, Planetlab version 2.0 was installed on the nodes. We ran in total 51 runs on 18 different Planetlab nodes. Six datasets were generated for the data analysis phase, in which we analyze the statistical properties of the datasets and identify the shortcomings of the existing prediction methods and which serves as the basis for the development of our new prediction method, called DES. 45 additional datasets were generated for the comparison phase, in which we extensively compare the accuracy of the DES-based predictions with the predictions based on the existing methods.

Each dataset is generated by a single run and consists of the running times (i.e., wall-clock times) of 2000 consecutive and identical jobs. Hence, we suppose the processing times (i.e., the pure processor times that the jobs take) of all the jobs to be constant and measure the time the jobs take on shared processors (i.e., the job run times). Note that Kleinrock [11] shows that for given value of the load there is a linear relation between the job processing times and the job running times.

The average duration of a run is about two hours. We observed significant differences in the durations, some even exceed the 10 hours, especially the runs on the Arizona node. In order to correlate the statistical properties of the running times at the different nodes, at each day all runs were started *simultaneously* at 9:00 CET. For the data analysis phase, experiments were done during two consecutive days. During the first day, three runs were launched: one at Tel Aviv, Israel (telaviv); one at Tucson, USA (ar); and one at Sydney, Australia (au). During the second day, three additional runs were launched at Warsaw, Poland (warsch); Tucson; and Sydney. The abbreviations for the six datasets in the data analysis phase are shown in Table 1 below.

	Day	
Run number	1	2
Run 1	pre_telaviv01	pre_warsch01
Run 2	pre_ar01	pre_ar02
Run 3	pre_au01	pre_au02

Table 1: Run schedule for the data analysis phase.

For the comparison phase, experiments were done during six additional consecutive days. In addition to the four nodes we used in the analysis set we used nodes in Amsterdam, The Netherlands (ams); Beijing, China (china); Le Chesnay, France (inria); Copenhagen, Denmark (dk); Madrid, Spain (mad); Moscow, Russia (mos); Pasadena, USA (cal); San Diego, USA (sandiego); Santa Barbara, USA (santab); Singapore, Singapore (sing); Taipei, Taiwan (tw); Salt Lake City, USA (utah); Vancouver, Canada (ca); and Washington, USA (wash). In practice, we started more runs than the 51 represented in this table; interrupted jobs were omitted in the table. The selection of which nodes were used during which days is based on both the availability, and the global distribution of the nodes. Moreover, we never started a new run on a node on which a previous run was interrupted. Table 2 shows which nodes were used during these six days.

2.2 Statistical analysis

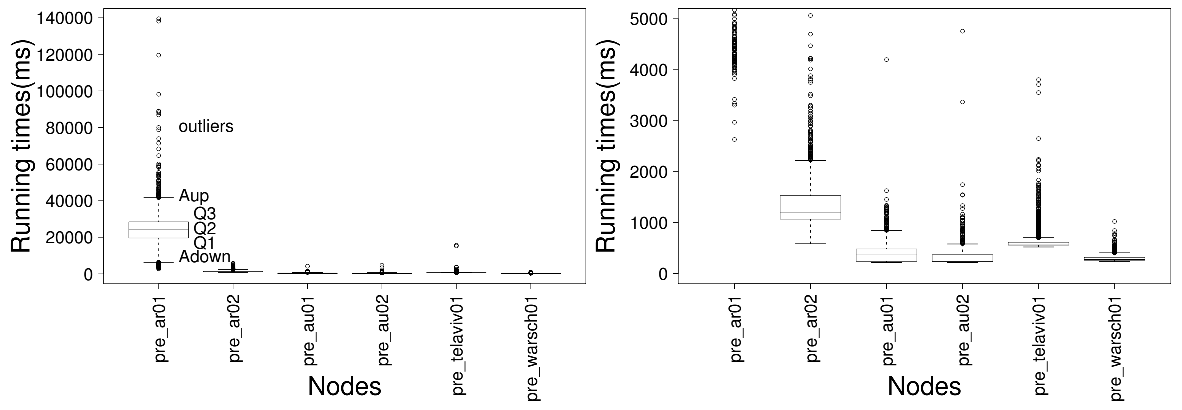
In this section we successively discuss the Box-and-Whisker plots, histograms, Auto Correlation Functions (ACFs), and other statistical properties of the six datasets, which are gathered for the data analysis phase, see Table 1.

	Day					
Run number	1	2	3	4	5	6
Run 1	ams01	ams02	ams03	ams04	ams05	inria01
Run 2	ar01	ar02	ar03	ar04	ar05	ar06
Run 3	warsch01	cal01	cal02	cal03	cal04	cal05
Run 4	china01	china02	wash01	wash02	wash03	wash04
Run 5	au01	au02	au03	sandiego01	sandiego02	sandiego03
Run 6	utah01	utah02	utah03	ca01	ca02	mad01
Run 7	mos01	mos02	telaviv01	telaviv02	tw01	tw02
Run 8	santab01	dk01	sing01			

Table 2: Run schedule for the comparison phase.

2.2.1 Box-and-Whisker plots

Constructing Box-and-Whiskerplots, commonly known as Boxplots (see Chapter 2C in [20]), is a common way to show the differences between the medians, the quartiles, the minimum, the maximum and the outliers of different datasets. Figure 5 depicts the Boxplots of the running times (in milliseconds) of the datasets. The left-hand side plot gives a macroscopic view of the data, and the plot at the right-hand side focuses on running times ranging from 0 to 5000 ms. More precisely, we consider the three quartiles: the 25%-percentile (denoted as $Q1$), the median (denoted as $Q2$) and the 75%-percentile, $Q3$. These quartiles are plotted as a long horizontal line. In addition, we consider the statistical measures $A_{down} := Q1 - 1.5(Q3 - Q1)$, $A_{up} := Q3 + 1.5(Q3 - Q1)$, which are indications of the data points that should not be considered as outliers, and indicated by short horizontal lines. Finally, the outliers are plotted by small circles. Figures 1(a) and 1(b) lead to the following observations: (1) the characteristics

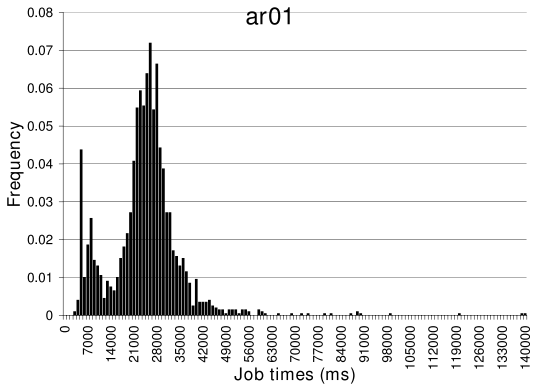


(a) **Boxplot of running times** (b) **Boxplot of running times with y-range 0. . . 5000**

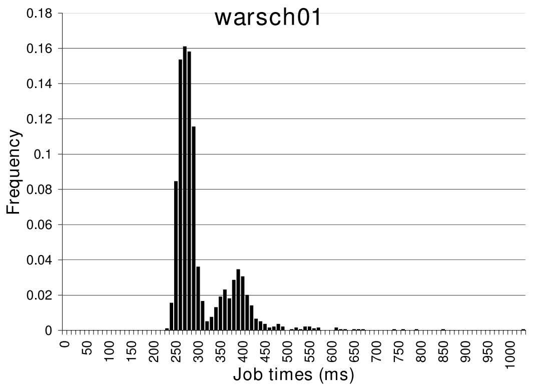
Figure 1: Boxplots of the 6 analysis phase datasets

of the running times at a given node on different days in some cases differ strongly (see for example the results for pre_ar01 and pre_ar02), but can be quite similar in other cases (see for example pre_au01 and pre_au02), (2) the running-time characteristics of different nodes are strongly different, even when experiments are done at the same time, (3) the running times at a given node within a given run are highly bursty, and have a large number of strong outliers, and (4) in most cases the outliers correspond to very large values of the running times, but in some cases (see for example pre_ar01) outliers correspond to very small running times.

These observations address the need for prediction methods that can deal with heterogeneous and dynamically changing time series. Moreover, we reemphasize that the observed heterogeneity of characteristics of the running times in the grid environment differs *fundamentally* from the running-time characteristics in clusters of processors, which are homogeneous and well predictable.

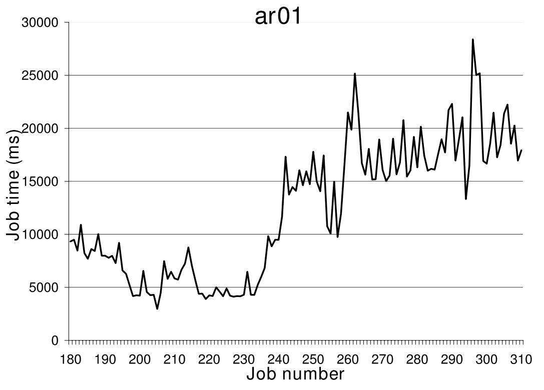


(a) **Histogram of pre_ar01**

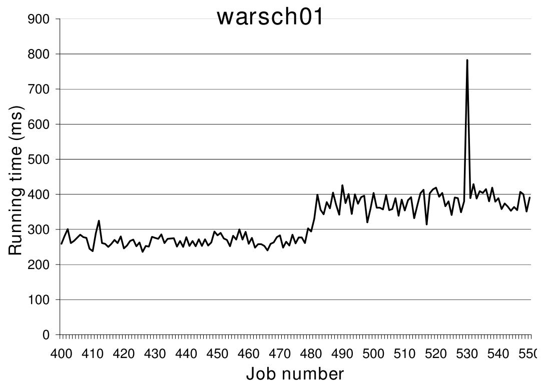


(b) **Histogram of pre_warsch01**

Figure 2: Histograms of datasets pre_ar01 and pre_warsch01



(a) **Level switch in pre_ar01**



(b) **Level switch in pre_warsch01**

Figure 3: Level switches in datasets pre_ar01 and pre_warsch01

2.2.2 Histograms

To analyze the frequency distribution of the running times in more detail, Figure 2 shows the histograms of the marginal running-time distributions of two representative datasets.

The results in Figure 2 show that the running-time distributions are multi-modal. This is caused by level switches in the running times over sustained time periods (ranging from minutes to hours). To illustrate this, Figure 3 gives a graphical representation of parts of datasets pre_ar01 and pre_warsch01. Figure 3(a) shows that from data points 235 to 260 a level switch occurs. From job number 180 to 230, the running times are between 3500 and 6000, which explains the first top in Figure 2(a). From job number 260 to 310 the measured running times are between 15000 and 30000, which explains the second top in Figure 2(a). Similar level switches are observed in Figure 3(b), which explains the bi-modal distribution of the running times in Figure 2(b). These level switches are presumably caused by changes in the processor load due to the launching or termination of other jobs on the same node. As stated by Dinda et al. [3, 5] much of the behavior of running times can be explained by changes in the load. However, they conclude that there are many other factors (e.g. memory space) that have their influence on the running times. In this paper, we focus on reacting to the resulting fluctuations and not on prevention of the causes of fluctuations.

2.2.3 Auto Correlation Function

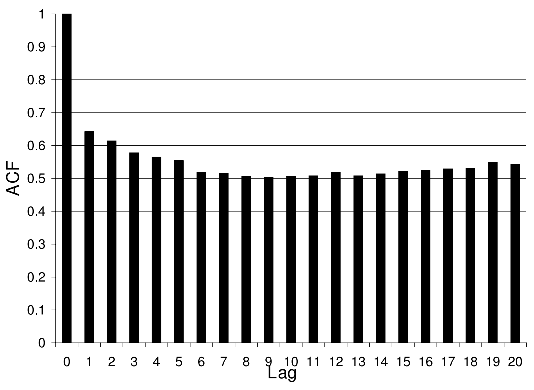
The Auto Correlation Function (ACF) is an effective way to investigate whether the data points show correlations over different time scales. For a given set of data points $\underline{y} = (y_1, \ldots, y_N)$, the ACF is defined as follows: For $h = 1, \ldots, N$,

$$\rho(h) \quad := \quad \frac{\gamma(h)}{\gamma(0)}, \tag{1}$$

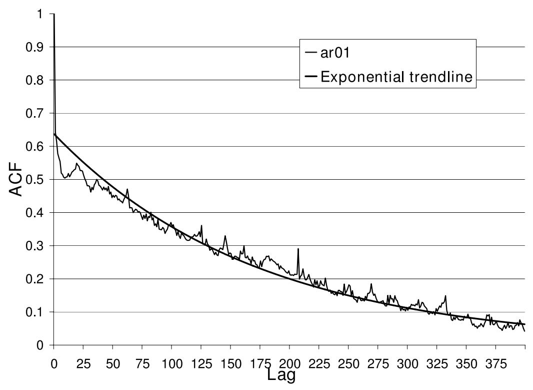
with

$$\gamma(h) \quad := \quad \frac{1}{N} \sum_{t=1}^{N-h} (y_{t+h} - \mu(\underline{y}))(y_t - \mu(\underline{y})), \tag{2}$$

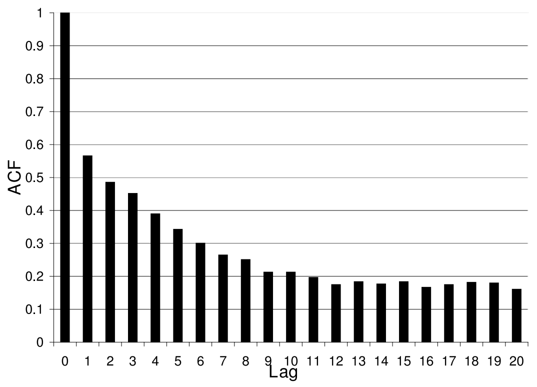
where h is the lag, and where $\mu(\underline{y}) := \sum_{t=1}^{N} y_t / N$. The ACFs of datasets pre_ar01, pre_au01, and pre_telaviv01 are shown in Figure 4. The results show that the ACFs do not follow predictable patterns, and show strong differences over the small (ranging from 1 up to 20) and large (higher than 20) lag values. For example, the ACF has significant autocorrelations over the small lag values of datasets pre_ar01 and pre_au01. On the contrary, the ACF for dataset pre_telaviv01 decreases very quickly to 0, even for small lag numbers. For the datasets with significant autocorrelations for the large lag values, we consistently observe exponentially decaying autocorrelations, as illustrated by Figure 4(b), which suggests that the successive running times are short-range dependent. Periodicity can be observed by ACFs if one or more lag values are higher than the other lag values. Although some parts of the datasets temporarily show periodicity, we conclude that in general the datasets do not show periodicity. However, we notice that the datasets represent job runtimes measured on one node with a time period of less than one day. For that reason, it is possible that the job runtimes show periodicity on longer times scales. We expect that if there exists a daily periodic behavior in the job runtimes of Planetlab nodes, this is not a strong behavior due to the fact that researchers at all times zones use the nodes.



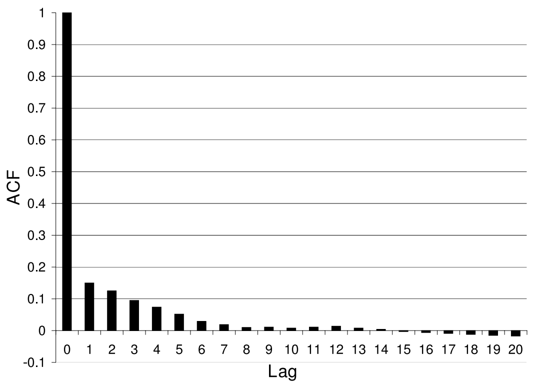
(a) **pre_ar01**



(b) **pre_ar01 (400 ACF values)**



(c) **pre_au01**



(d) **pre_telaviv01**

Figure 4: ACF plots of different datasets

2.2.4 Other statistical properties

In this section, we provide an analysis of another set of statistical properties of the datasets. First, we define the long-term fluctuations or level switches as changes in the height of the data points that continue for more than 4 consecutive data points. To analyze the changes in the statistical properties over time, the index set $I := \{1, \ldots, N\}$ is partitioned into

$$I = I_1 \cup \cdots \cup I_M, \text{ where } I_k := \{B(k-1)+1, \ldots, Bk\} \text{ for } k = 1, \ldots, M,$$

where B is the block size and $M := N/B$ (assuming that M is integer-valued). Let 1_E be the indicator function on the event E, i.e., $1_E = 1$ if E is true and $1_E = 0$ otherwise. Further, let $\underline{y}_{I_k} := (y_{B(k-1)+1}, \ldots, y_{Bk})$, for $k = 1, \ldots, M$. For a vector $\underline{v}$, we define $|\underline{v}|$ to be the number of elements in vector $\underline{v}$, and, therefore, $\underline{v} := (v_1, \ldots, v_{|\underline{v}|})$. Moreover, we define

$$\mu(\underline{v}) := \frac{1}{|\underline{v}|} \sum_{t=1}^{|\underline{v}|} v_t, \tag{3}$$

and

$$\sigma(\underline{v}) := \sqrt{\frac{1}{|\underline{v}|-1} \sum_{t=1}^{|\underline{v}|} (v_t - \mu(\underline{v}))^2}. \tag{4}$$

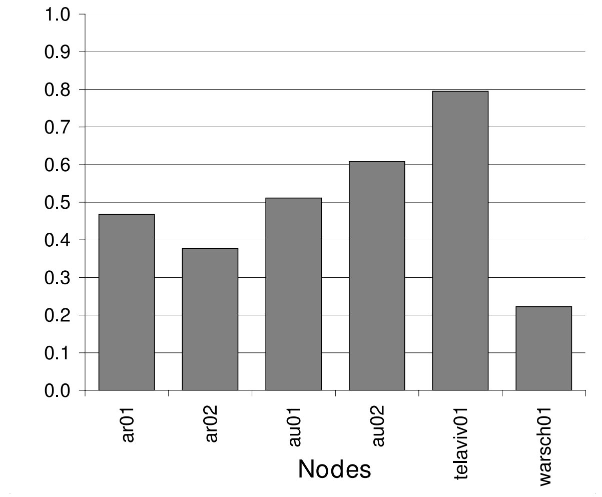
Using this notation, we consider the set of statistical properties defined in Table 3 below. Interpretations of the statistics and their formulas are given further on in this section.

Using the definitions in Table 3 we have calculated the statistics for the six datasets defined in Table 1 above, where the number of data points is $N = 2000$ and the block size $B = 20$. The results are shown in Table 4 below.

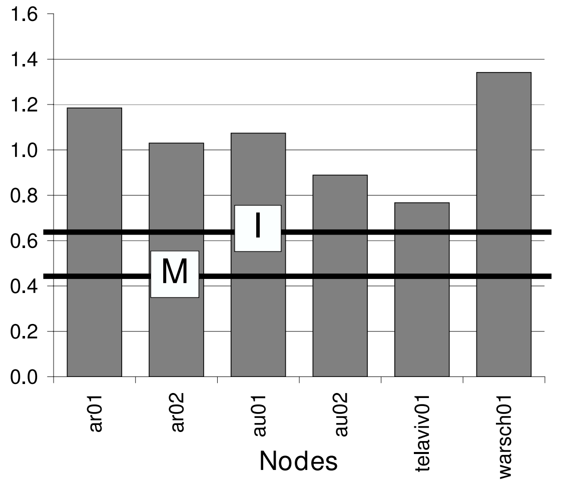
To highlight the most interesting statistical properties, Figure 5 below gives a graphical representation of the results for a number of statistics. From each group of statistics, the most important statistic is chosen to make a graphical representation. The results presented in Table 4 and Figure 5 are discussed in detail below.

The first group of statistics μ, σ, and CoV indicate the main characteristics of the datasets: statistic μ is the average, statistic σ the standard deviation, and statistic CoV is the Coefficient of Variance. The CoV equals the standard deviation divided by the average, and is a scale-invariant indicator for the variability of the data points.
Similar to the results in Section 2.2, we conclude that the average and the standard deviations of the running times of the jobs may differ strongly between the dataset. Datasets can show similarities when two runs are performed on the same node and on two consecutive days, which is illustrated by pre_au01 and pre_au02. Table 4 and Figure 5(a) show that the CoVs of the running times are fairly low, ranging between 0.22 and 0.79.

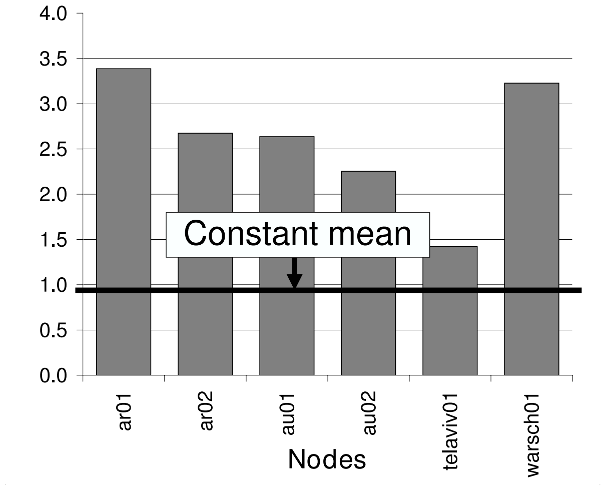
Statistics $rmse_{lv}$ and $\sigma/rmse_{lv}$ indicate (1) to what extent the standard deviation is caused by short- or long-term fluctuations, and (2) how well the last value performs as a predictor for running times of jobs. Statistic $rmse_{lv}$ is the so-called Root of the Mean Squared Error (RMSE) of the last value, which is the RMSE of a predictor that uses the last value to predict the next value. Moreover, this statistic indicates the total amount of fluctuations. Statistic $\sigma/rmse_{lv}$ is the standard deviation over the RMSE of the last value, and is represented in Figure 5(b). Statistic $\sigma/rmse_{lv}$ theoretically varies between 0.5 and infinity. For (1), when Statistic $\sigma/rmse_{lv}$ has the theoretical minimum value of 0.5 (see line M in Figure 5(b)), it indicates that the values alternate between two values; two random successive values show a correlation of -1. A value of $0.7 (= \frac{1}{2}\sqrt{2})$ (see line I in Figure 5(b)) indicates that the values are independent and identically distributed. When the value for this statistical property is higher than 0.7 it indicates short- and long-term fluctuations; the higher the value the more long-term fluctuations influence the standard deviation. For (2), a value of 1.0 for this property indicates that the average and the last value as predictors have the same accuracy. The higher this value the better is the last value as predictor in comparison with the average.



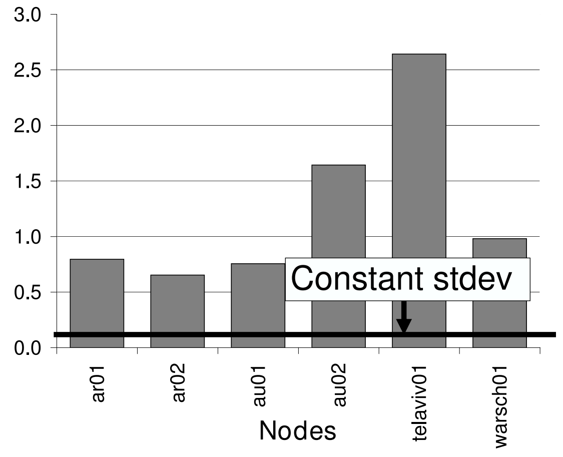
(a) **Coefficient of Variance**



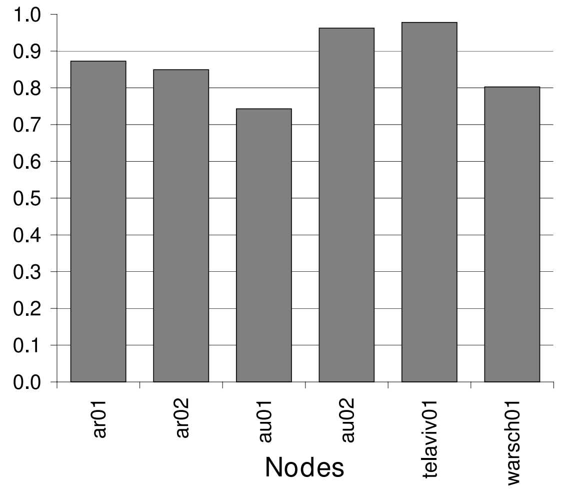
(b) **Standard deviation over RMSE**



(c) **Standard deviation of the averages over the standard deviation**



(d) **CoV of Standard Deviations**



(e) **Fraction of total amount of fluctuations caused by jumps**

Figure 5: Various statistical properties

Name	**Statistic**	**Formula**
μ	Average of whole dataset	$\mu(\underline{y}) := \frac{1}{N}\sum_{t=1}^{N} y_t$
σ	Standard deviation of whole dataset	$\sigma(\underline{y}) := \sqrt{\frac{1}{N-1}\sum_{t=1}^{N}(y_t - \mu(\underline{y}))^2}$
CoV	Coefficient of Variance	$\mathrm{CoV}(\underline{y}) := \frac{\sigma(\underline{y})}{\mu(\underline{y})}$
$rmse_{lv}$	RMSE of last value	$rmse_{lv}(\underline{y}) := \sqrt{\frac{1}{N-1}\sum_{t=2}^{N}(y_t - y_{t-1})^2}$
$\frac{\sigma}{rmse_{lv}}$	$\frac{\text{Standard deviation}}{\text{RMSE of last value}}$	$\frac{\sigma(\underline{y})}{rmse_{lv}(\underline{y})} := \frac{\sigma(\underline{y})}{rmse_{lv}(\underline{y})}$
$\sigma(\underline{\mu})$	Standard deviation of the averages	$\sigma(\mu(\underline{y_{I_1}}), \ldots, \mu(\underline{y_{I_M}})) := \sqrt{\frac{1}{M}\sum_{k=1}^{M}(\mu(\underline{y_{I_k}}) - \mu(\underline{y}))^2}$
$\mathrm{CoV}(\underline{\mu})$	CoV of the averages	$\mathrm{CoV}(\mu(\underline{y_{I_1}}), \ldots, \mu(\underline{y_{I_M}})) := \frac{\sqrt{B}\sigma(\mu(\underline{y_{I_1}}), \ldots, \mu(\underline{y_{I_M}}))}{\mu(\underline{y})}$
$\frac{\sigma(\underline{\mu})}{\sigma}$	$\frac{\text{Standard deviation of averages}}{\text{Standard deviation}}$	$S_8(\underline{y}) := \frac{\sqrt{B}\sigma(\mu(\underline{y_{I_1}}), \ldots, \mu(\underline{y_{I_M}}))}{\sigma(\underline{y})}$
$\sigma(\underline{\sigma})$	Standard deviation of standard deviations	$\sigma(\sigma(\underline{y_{I_1}}), \ldots, \sigma(\underline{y_{I_M}})) := \sqrt{\frac{1}{M}\sum_{k=1}^{M}(\sigma(\underline{y_{I_k}}) - \mu(\sigma(\underline{y_{I_1}}), \ldots, \sigma(\underline{y_{I_M}})))^2}$
$\frac{\sigma(\underline{\sigma})}{\mu}$	$\frac{\text{Standard deviation of standard deviations}}{\text{Average}}$	$S_{10}(\underline{y}) := \frac{\sigma(\sigma(\underline{y_{I_1}}), \ldots, \sigma(\underline{y_{I_M}}))}{\mu(\underline{y})}$
$\mathrm{CoV}(\underline{\sigma})$	CoV of the standard deviations	$\mathrm{CoV}(\sigma(\underline{y_{I_1}}), \ldots, \sigma(\underline{y_{I_M}})) := \frac{\sigma(\sigma(\underline{y_{I_1}}), \ldots, \sigma(\underline{y_{I_M}}))}{\mu(\sigma(\underline{y_{I_1}}), \ldots, \sigma(\underline{y_{I_M}}))}$
f_{jumps}	Fraction of jumps	$f_{jumps}(\underline{y}) := \frac{1}{N-1}\sum_{t=2}^{N} 1_{\{\lvert y_t - y_{t-1}\rvert > 2\sigma(\underline{y})\}}$
σ_{jumps}	Standard deviation impact of jumps	$\sigma_{jumps}(\underline{y}) := \sqrt{\frac{1}{N-1}\sum_{t=2}^{N} 1_{\{\lvert y_t - y_{t-1}\rvert > 2\sigma(\underline{y})\}}(y_t - \mu(\underline{y}))^2}$
$rmse_{jumps}$	RMSE impact of jumps	$rmse_{jumps}(\underline{y}) := \sqrt{\frac{1}{N-1}\sum_{t=2}^{N} 1_{\{\lvert y_t - y_{t-1}\rvert > 2\sigma(\underline{y})\}}(y_t - y_{t-1})^2}$
$\frac{\sigma_{jumps}}{\sigma}$	$\frac{\text{Standard deviation impact of jumps}}{\text{Standard deviation}}$	$S_{15}(\underline{y}) := \frac{\sigma_{jumps}(\underline{y})}{\sigma(\underline{y})}$
$\frac{rmse_{jumps}}{rmse_{lv}}$	$\frac{\text{RMSE impact of jumps}}{\text{RMSE of last value}}$	$S_{16}(\underline{y}) := \frac{rmse_{jumps}(\underline{y})}{rmse_{lv}(\underline{y})}$
$\frac{mse_{jumps}}{\sigma^2}$	$\frac{\text{MSE impact of jumps}}{\text{variance}}$	$S_{17}(\underline{y}) := \frac{\sigma_{jumps}(\underline{y})^2}{\sigma(\underline{y})^2}$

Table 3: Definitions of the statistics of the datasets

The results in Table 4 show a high variety of the values of Statistic $rmse_{lv}$, ranging from 50 to almost 9500. The average of Statistic $\sigma/rmse_{lv}$ is 1.05, which is significantly higher than 0.7, which indicates long-term fluctuations. Moreover, it indicates that the last value is on average a slightly better predictor than the average. For the two datasets pre_au02 and pre_tel01 that have a value lower than 1.0 an average predictor would predict better. We conclude from the high variety of the height of this statistic that the proportion long- and short-term fluctuations differs per dataset.

The next group statistics $\sigma(\underline{\mu})$, $\mathrm{CoV}(\underline{\mu})$, and $\sigma(\underline{\mu})/\sigma$ indicate to what extent the average of the running times of jobs changes during the run. Statistic $\sigma(\underline{\mu})$ is the standard deviation of the averages, Statistic $\mathrm{CoV}(\underline{\mu})$ is the CoV of the averages, and Statistic $\sigma(\underline{\mu})/\sigma$ is the standard deviation of the averages of the standard deviation. Statistics $\mathrm{CoV}(\underline{\mu})$ and $\sigma(\underline{\mu})/\sigma$ are multiplied by a correction factor $\sqrt{B}$ to compensate the fact that a standard deviation of an average of B independent, identically distributed values is by definition $\sqrt{B}$ times smaller than the standard deviation of a single value. The closer the value of Statistic $\sigma(\underline{\mu})$ is to 0, the more constant the average stays during the run. When statistic $\mathrm{CoV}(\underline{\mu})$ has a value close to 1.0 or higher, the averages fluctuate significantly. A value of 1.0 for statistic $\sigma(\underline{\mu})/\sigma$ indicates that the expectation stays constant during the run. Fluctuations in the standard deviations have no influence on this property.
Table 4 and Figure 5(c) show that for all datasets the averages fluctuate significantly: Statistic $\mathrm{CoV}(\underline{\mu})$ has an average value 1.19, and Statistic $\sigma(\underline{\mu})/\sigma$ shows values that are significantly higher than 1.0. Nevertheless, the datasets show a high diversity in to what extent the averages fluctuate. We conclude

Name	**pre_ar01**	**pre_ar02**	**pre_au01**	**pre_au02**	**pre_telaviv01**	**pre_warsch01**
μ	23922	1346	407	301	656	300
σ	11187	507	208	183	521	67
CoV	0.47	0.38	0.51	0.61	0.79	0.22
$rmse_{lv}$	9443	493	194	206	680	50
$\frac{\sigma}{rmse_{lv}}$	1.18	1.03	1.07	0.89	0.77	1.34
$\sigma(\underline{\mu})$	8471	303	123	92	166	48
CoV($\underline{\mu}$)	1.57	1.03	1.34	1.39	1.11	0.72
$\frac{\sigma(\underline{\mu})}{\sigma}$	3.39	2.67	2.64	2.25	1.42	3.23
$\sigma(\underline{\sigma})$	4665	230	105	139	479	34
$\frac{\sigma(\underline{\sigma})}{\mu}$	0.20	0.17	0.26	0.46	0.73	0.11
CoV($\underline{\sigma}$)	0.80	0.65	0.76	1.64	2.64	0.98
f_{jumps}	0.03	0.03	0.02	0.04	0.01	0.03
σ_{jumps}	6265	330	108	148	488	35
$rmse_{jumps}$	8240	418	144	198	665	40
$\frac{\sigma_{jumps}}{\sigma}$	0.56	0.65	0.52	0.81	0.94	0.53
$\frac{rmse_{jumps}}{rmse_{lv}}$	0.87	0.85	0.74	0.96	0.98	0.80
$\frac{mse_{jumps}}{\sigma^2}$	0.31	0.42	0.27	0.65	0.88	0.28

Table 4: Summary of the statistical properties of the datasets

from this group of statistics that all the datasets have many level switches.

Statistic $\sigma(\underline{\sigma})$, $\sigma(\underline{\sigma})/\mu$, and CoV($\underline{\sigma}$) indicate how much the standard deviation of the running times changes during the run. Statistic $\sigma(\underline{\sigma})$ is computed by taking the standard deviation of standard deviations of B successive values. Statistic $\sigma(\underline{\sigma})/\mu$ equals statistic $\sigma(\underline{\sigma})$ divided by the average, and Statistic CoV($\underline{\sigma}$), which is graphically represented in Figure 5(d), equals Statistic $\sigma(\underline{\sigma})$ divided by the average of the standard deviations. In general, the closer these statistics are to 0, the more the standard deviation stays constant during the run. If the running times are independent, identical normally distributed, statistic $c(\underline{\sigma})$ equals 0.16, which is independent of the height of the average and the standard deviation. We note that 0.16 depends on the value of B: the higher the B, the lower the value.
Table 4 and Figure 5(d) show that all the datasets have substantial fluctuations in the standard deviations. Nevertheless, there is diversity in the fluctuations of the standard deviations between the different sets. The running times on the node in Telaviv shows the most fluctuations in the standard deviation. For that node the high value for Statistic CoV($\underline{\sigma}$) is mainly caused by the many big outliers, as we see in Figure 1(b).

The last group of Statistics f_{jumps}, σ_{jumps}, $rmse_{jumps}$, σ_{jumps}/σ, $rmse_{jumps}/rmse_{lv}$, and mse_{jumps}/σ^2 indicates how many jumps the datasets contain and what the impact of those jumps is on the total amount of fluctuations, the standard deviation, and the variance. The first statistic of this group is the fraction of jumps, where a jump is defined as a value that differs (up- or downwards) from its previous value more than 2 times the standard deviation. A jump can be a peak or a level switch. The second statistic is the impact of the jumps on the standard deviation. This statistic sums up all the squares of the differences between the jumps and the averages, and takes the square root of that sum. The next statistic is the RMSE impact of jumps. This statistic sums up all the squares of the differences between the jumps and their previous values, and takes the square root of that sum. Statistic 15 indicates the fraction of the standard deviation that is caused by the jumps. It is computed by taking Statistic σ_{jumps} over the standard deviation. Statistic $rmse_{jumps}/rmse_{lv}$ indicates the fraction of the total amount of fluctuations that is caused by the jumps. This property equals Statistic $rmse_{jumps}$ divided by Statistic $rmse_{lv}$. The last Statistic, mse_{jumps}/σ^2, indicates how much influence the jumps have on the variance. This value equals the square of Statistic σ_{jumps}/σ.
The results show that the fraction of the jumps does not differ that much between the datasets, and ranges between 1% and the 4%. The fraction of jumps is relatively low compared to the normal distribution (16%) and to the exponential distribution (13%). Statistics σ_{jumps} and $rmse_{jumps}$ illustrate huge differences between the different impacts of jumps on the standard deviation and on the variance. Statistics σ_{jumps}/σ and mse_{jumps}/σ^2 show that the fraction of the standard deviation and the variance that is caused by the jumps (respectively 67% and 47%) is quite similar to the fractions of the normal distribution (respectively 60% and 36%) and the exponential distribution (respectively 73% and 53%). Moreover, the graphical representation of statistic $rmse_{jumps}/rmse_{lv}$, Figure 5(e), shows that on average 87% of the total amount of fluctuations (i.e. RMSE of last value) is caused by the jumps and that for two of the nodes the fraction is even higher than 95%. This value is significantly higher than those of the normal distribution (76%) and the exponential distribution (81%).

To summarize, the statistical data analysis of the datasets shows that: (1) there are significant differences between the characteristics of the datasets, (2) the datasets show on average more long-term than short-term fluctuations and the proportion differs per dataset, (3) the averages fluctuate significantly during the run, with differences in the amount of fluctuations between the different nodes, (4) the standard deviations fluctuate significantly during the run, with differences in the amount of fluctuations between the different nodes, and (5) the datasets contain a small amount of jumps that have a huge influence on the standard deviation, the total amount of fluctuations, and the variance.

3 Analysis of Existing Prediction Methods

In this section, we analyze a variety of existing methods that can be used to predict the running times of jobs on shared processors. The most commonly used predictors in grid studies are ES, NWS and AR (these predictors will be described below). Nevertheless, for predicting the running times of tasks there are many other useful predictors applicable from other areas (e.g., economics). We selected the

Adaptive Exponential Smoothing Predictors (AESP) and Smooth Transition Exponential Smoothing (STES) predictors as potential methods that can give accurate predictions in grids. From the AESP we discuss Trigg and Leach [19], Whybark [21], Mentzer [14], and from the STES predictors we discuss STES $|e|$ with $\gamma < 0$, STES $|e|$, STES e^2, and STES Whybark, which we selected from [18]. Below, we describe the different prediction methods and assess their strong and weak points for making forecasts based on the characteristics of the datasets discussed in the previous section.

3.1 Common grid predictors

In this section, we consider the most commonly used predictors in grid environments: Exponential Smoothing (ES), Network Weather Service (NWS) and Autoregression (AR).

3.1.1 Exponential Smoothing

ES is a simple prediction method that surprisingly often works very good in practice. We define y_t as the measured value at time t, $\hat{y}_t$ as the prediction for y_t, α as a chosen parameter between 0 and 1. Next, the prediction for y_t is defined as:

$$\hat{y}_t = \alpha y_{t-1} + (1-\alpha)\hat{y}_{t-1}, \text{ where } 0 \leq \alpha \leq 1. \tag{5}$$

ES is a good method because it does not react directly to peaks while it reacts fast enough to level switches (which are observed frequently in the datasets discussed in Section 2). The most suitable value of the interpolation parameter α depends on the characteristics of the dataset. A weak point is that once a parameter is chosen, it always reacts in the same way to peaks and level switches, even when the structure is changed completely, which were found to occur frequently in Section 2. For example, Figure 6 shows that ES with parameter 0.5 does not react properly to a peak, even when there were many peaks in the history data. In this paper we use the ES parameter 0.5, because in [6] is stated that 0.5 is the value that leads to the most accurate predictions for running times of jobs.

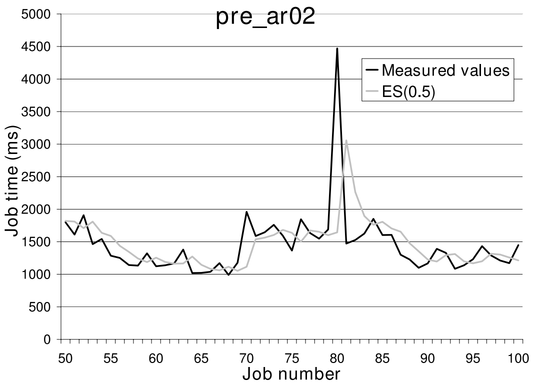
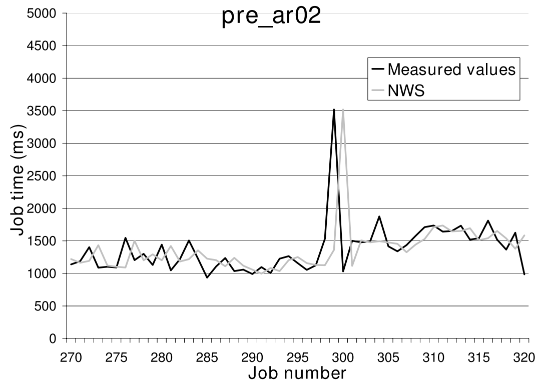


Figure 6: ES reacting on a peak on pre_arizona02

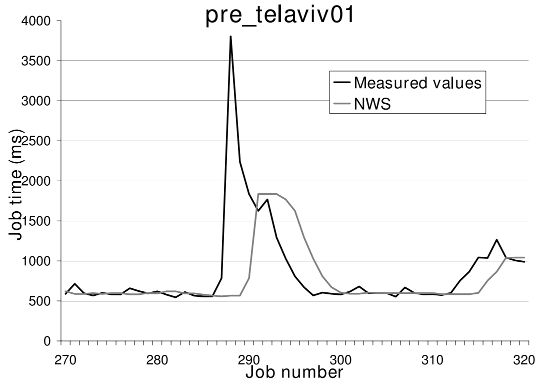
3.1.2 The Network Weather Service

The Network Weather Service (NWS) prediction method conducts postcasts using different windows of previous data (always starting with the most recent data and working backwards in time) and records the "winning" forecaster for each window size. A postcast is a forecast of a property at a historical point in time. The prediction accuracy can be immediately measured. Each window with historic data is subsequently treated as a separate forecaster and a final accuracy tournament determines which forecaster will be used. The predictor selects from the following set of predictors: the adaptive median window 5-21 and 21-51, 30% trimmed median window 31 and 51, sliding median window 5 and 31, median window 5 and 31, the running mean, the last value, exponential smoothing with trend 10% and parameter 30%, 20%, 15%, 10%, and exponential smoothing with parameter 90%, 75%, 50%, 40%, 30%, 20%, 15%, 10%, 5%. Further details can be found in [23].

A strong point of this method is that it contains a large set of predictors. They represent many different characteristics of the datasets. Therefore, the NWS prediction method is able to deliver accurate predictions in many situations. Further investigations with the 6 analysis-phase datasets show that some parts of the datasets contain more than 20% jumps, or show an alternating characteristic. For those cases a predictor based on the average is the most accurate. Some parts of the datasets show a homeostatic character, for which the sliding median with a window size of 31 gives the best predictions. Both types (i.e., the average and the sliding median with a window size of 31) of predictors are represented in the NWS set. Another good point of this selection method is the case of changing characteristics, this predictor rapidly chooses another predictor that predicts the new situation more accurately. A weak point is that it does not always react properly to peaks. Figure 7(a) shows that when there were small level switches in history, the NWS method often chooses a predictor that over reacts to peaks. That is even the case when peaks in history never introduced big changes. Moreover, Figure 7(b) shows that the NWS prediction method reacts too slowly on the new level, despite many peaks introduced a new level for previous values. Clearly, peaks in datasets have to be interpreted differently than regular values. However, the NWS does not make a distinction between those types of values.



(a) **Peak on pre_arizona02**



(b) **Level switch on pre_telaviv01**

Figure 7: NWS-predictor reacting on a peak and a level switch in the running times

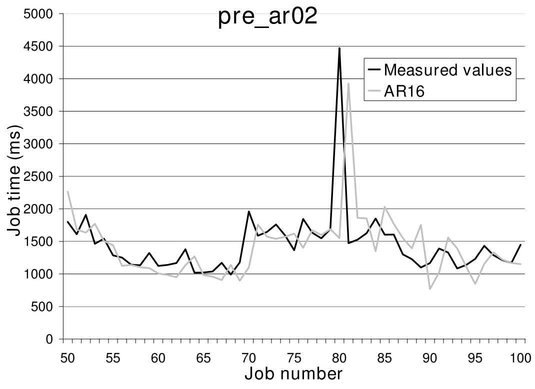
3.1.3 Autoregression

Autoregressive (AR) predictors multiply previous data points with some parameter between 0 and 1 to compute the next prediction. AR models have a parameter p which indicates the number of data points it uses from the history. Generally, AR predictions (denoted as $AR(p)$) are calculated in the following way:

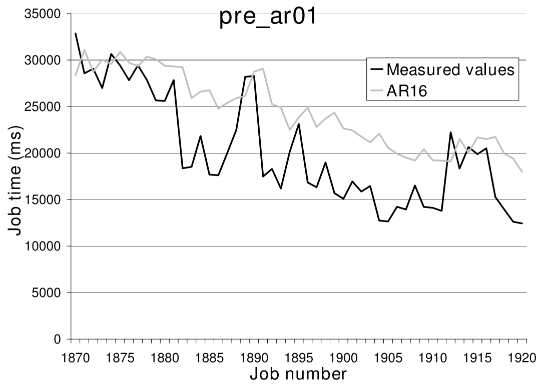
$$\hat{y}_t = \alpha_1 y_{t-1} + \alpha_2 y_{t-2} + \ldots + \alpha_p y_{t-p}, \text{ with } \sum_{i=1}^{p} \alpha_i = 1 \text{ and } 0 \leq \alpha_i \leq 1. \tag{6}$$

The α parameters are carefully chosen, based on historical correlations between y_t's over different time scales. When the expectation of the y_t's does not equal 0, the mean of the y_t's is subtracted from those y_t's and finally added to the $\hat{y}_t$. The sum of the α_i values does not necessarily equals 1.0. Dinda [5] has also analyzed a set of other prediction models, related to AR models, and showed that $AR(16)$ performs the best in forecasting processor loads and running times of tasks. However, as stated in [5], they only work well in case of periodicity. As we have seen in Figures 4(a) to 4(d), we notice that the datasets do not show periodicity. Dinda used k-steps-ahead prediction methods and implemented the Yule-Walker technique. However, we only use the part of Dinda's software that implements the one-step ahead prediction methods for the following two reasons. The first reason is the ability to compare the predictions with the one-step-ahead predictions of other predictors and the second reason is that one-step ahead predictions are significantly more accurate.

A strong point is that AR methods adapt the parameters in such a way that it efficiently reacts to periodicity. In case of many peaks, this method will adapt the parameters in a more 'averaging' way, such that the peaks do not influence the predictions too much. When more level switches occur, this method chooses for higher values for the first α parameters, such that it reacts fast on those switches. However, as we have seen in Section 2.2.3, the datasets do not clearly show periodicity. Therefore, the aspect of adapting to periodicity of AR methods is not an advantage for our datasets. Another weak point is that the choice of parameters is not optimal. For example, when the AR method takes a high first parameter because the set shows some small level switches, it will give a highly inaccurate prediction when a higher peak occurs (see Figure 8(a)). Another drawback of this method is that when the structure of data points is changed it takes a long time before the method is adapted due to the fact that this method uses more than 100 data points to fit the parameters. Also in case of a trend upwards or downwards (see for example Figure 8(b)) this method adapts the predictions too slowly, because a significant part of the prediction is based on the average of the whole dataset.



(a) **Peak on pre_arizona02**



(b) **Trend down on pre_arizona01**

Figure 8: AR16-predictor reacting on a peak and a trend down in the running times

3.2 Adaptive Exponential Smoothing Predictors

Adaptive Exponential Smoothing Predictors (AESP) [18] use the following formula:

$$\hat{y}_t = \alpha_{t-1} y_{t-1} + (1 - \alpha_{t-1})\hat{y}_{t-1}. \tag{7}$$

and focus on adapting α_t such that the α_t will always get a good value that is independent of the start value, and adapts when the structure of the values changes. We consider the following variants of AESPs: Trigg and Leach [19], Whybark [21], Mentzer [14], Pantazopoulos and Pappis [15], and the STES predictors [18].

3.2.1 Trigg and Leach

The method of Trigg and Leach [19] defines the smoothing parameter as the absolute value of the ratio of the smoothed forecast error to the smoothed absolute error.

$$\hat{y}_t = \alpha_{t-1} y_{t-1} + (1 - \alpha_{t-1})\hat{y}_{t-1}, \tag{8}$$

with

$$\begin{aligned} \alpha_t &:= \left|\frac{A_t}{M_t}\right|, & (9)\\ A_t &:= \phi(y_t - \hat{y}_t) + (1-\phi)A_{t-1}, & (10)\\ M_t &:= \phi|y_t - \hat{y}_t| + (1-\phi)M_{t-1}, & (11) \end{aligned}$$

where ϕ is set arbitrarily, with 0.2 being a common choice [19]. Trigg and Leach explain that this formulation enables α_t to vary according to the degree to which biased forecasts are obtained.

A main advantage of this prediction method is that it chooses a low α parameter when there is

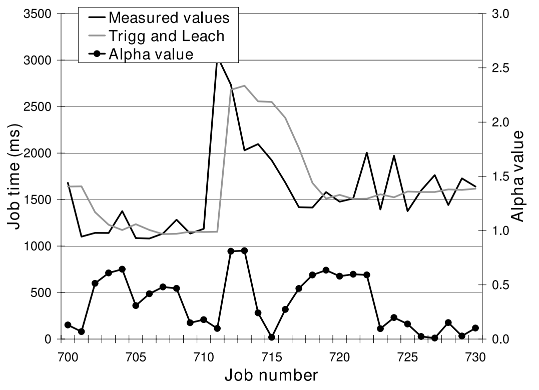


Figure 9: A trend up and down on pre_arizona02

a lot of noise in the dataset and a high α when there is less noise and there are some level-switches. These properties improve the accuracy of the predictor. A disadvantage of the proposed method is that it is difficult to find a suitable choice of the parameter ϕ and that the approach sometimes delivers unstable forecasts [10]. Another weak point is that when a few successive data points show a trend up (in Figure 9 from value 710 to 711) and then a trend down (see Figure 9 from value 711 to 717) in the α value will first increase to close to 1, which increases the accuracy, and then decreases to 0 and finally increases back to 1. Consequently, as is shown by the Trigg and Leach prediction values 713 to 718 in Figure 9, the predictor does not give very accurate predictions. In this case, it would be better when the α-value was kept fixed to 1. Similar problems occur when there are many level switches up- and downwards.

3.2.2 Whybark

AESP Whybark [21] defines the control limits in terms of multiples of the forecast error standard deviation, σ. An indicator variable, δ_t, is defined as:

$$\delta_t := \begin{cases} 1 & \text{when } |y_t - \hat{y}_t| > 4\sigma, \\ 1 & \text{when } |y_t - \hat{y}_t| > 1.2\sigma,\ |y_{t-1} - \hat{y}_{t-1}| > 1.2\sigma, \text{ and } (y_t - \hat{y}_t)(y_{t-1} - \hat{y}_{t-1}) > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

The value of δ_t determines whether α_t takes a base value, B, a medium value, M, or a high value, H. Whybark suggests $B = 0.2, M = 0.4$ and $H = 0.8$. Altogether, the Whybark prediction is computed in the following way:

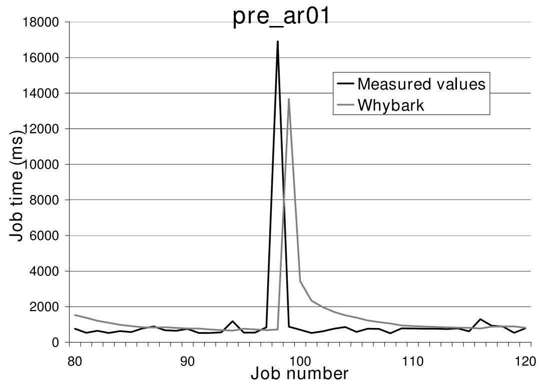
$$\hat{y}_t = \alpha_{t-1} y_{t-1} + (1 - \alpha_{t-1})\hat{y}_{t-1}, \tag{13}$$

with

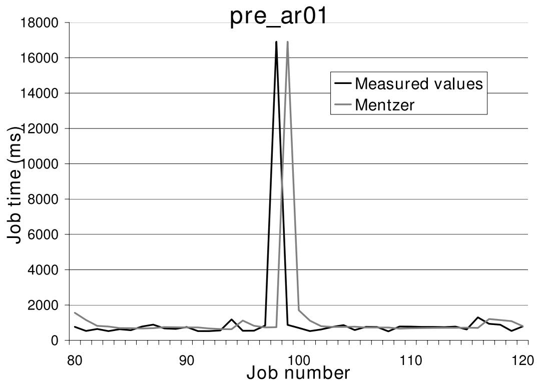
$$\alpha_t := \begin{cases} H & \text{when } \delta_t = 1, \\ M & \text{when } \delta_t = 0, \text{ and } \delta_{t-1} = 1, \\ B & \text{otherwise.} \end{cases} \tag{14}$$

A strong point of Whybark is that it distinguishes between normal situations and those where there was a huge difference between the prediction and the measured value; characteristics of predictions are always different when big differences occur. However, further analysis with the 6 datasets shows that the Whybark predictor gets a higher accuracy when it would distinguish between (1) 'normal' fluctuations, (2) fluctuations that are higher than 2 times the measured standard deviation, and (3) fluctuations that are higher than 10 times the measured standard deviation. A second strong point is that Whybarks' predictor contains a *correction part*; that is, when two successive measurements both deviate in the same direction with more than 1.2 times the standard deviation from the predictions, a different α_t value has to be applied. Further analysis with our analysis-phase datasets shows that this correction part increases the accuracy of the predictor, but that a value of 1.0 makes better distinctions. A weak point is that it does not distinguish between up- and downward fluctuations; it is not likely that those situations can be interpreted the same. Moreover, Whybark reacts in the same way to huge peaks as to two successive small differences between the prediction and the measured value. Furthermore, Whybark always uses the same parameters for the same kind of peaks; 0.8 for high peaks, 0.4 for the value after a high peak and also for two successive differences, and 0.2 as a base value. In practice, however, characteristics for each dataset are different. Figure 10(a) shows that the Whybark predictor does not react properly to peaks. Whybark will only work well when peaks introduce a new 'level' of values. It would be smarter when Whybarks predictor learns from historical data whether a huge difference introduces a new level or is a peak.

Finally, we conclude that the Whybark predictor contains a useful fluctuations classification, but that the following improvements can be made: (1) other decision rules to distinct between the 3 classes of fluctuations, (2) a correction part parameter of 1.0, (3) different treatment of down- and up-wards fluctuations, (4) different treatment of a data point after a huge peak and two successive significant, but not enormous, differences, and (5) adaptive parameters.



(a) **Whybark reacting on the peak**



(b) **Mentzer reacting on the peak**

Figure 10: Two predictors reacting on a peak in pre_au01

3.2.3 Mentzer

AESP Mentzer [14] uses the absolute forecast error fraction from the most recent period as α_t. In order to restrict α_t to the interval [0, 1], if the absolute error fraction exceeds 1.0, then α_t is set to 1:

$$\hat{y}_t := \alpha_{t-1} y_{t-1} + (1 - \alpha_{t-1})\hat{y}_{t-1}, \text{ with } \alpha_t := \min\left(\left|\frac{y_t - \hat{y}_t}{y_t}\right|, 1\right). \tag{15}$$

A strong point of this method is that when the forecasting error increases (decreases), the α value also increases (decreases) to improve the forecast. A weak point of this method is the reaction on peaks: when a peak occurs, the predictor will have a large forecasting error, which leads to a high next prediction, which in turn causes another large prediction error. This is illustrated in Figure 10(b). Another weak point is that it is not clear why there should be a 1- 1 relation between the α-value and the last forecasting error. For example, a forecasting error of 50% is a worse prediction and probably needs an α value that is higher than 0.5. The height of the α value also depends on the height of the standard deviation of the value.

3.2.4 Pantazopoulos and Pappis

Pantazopoulos and Pappis [15] argue that, since the ideal value for α_t would lead to $\hat{y}_{t+1} = y_{t+1}$, this ideal value can be derived by substituting y_{t+1} for $\hat{y}_{t+1}$ in (7) and solving for α_t to give

$$\alpha_t := \left(\frac{y_{t+1} - \hat{y}_t}{y_t - \hat{y}_t}\right). \tag{16}$$

Since y_{t+1} is unknown at time t, the ideal value of α_t for period $t-1$ is used for period t to give

$$\alpha_t := \left(\frac{y_t - \hat{y}_{t-1}}{y_{t-1} - \hat{y}_{t-1}}\right), \tag{17}$$

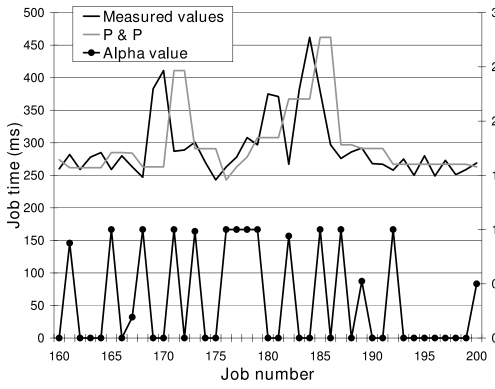


Figure 11: P&P-predictor in pre_warsch01

which can be substituted into the standard ES-Formula (7). In order to restrict α_t to the interval [0, 1], Pantazopoulos and Pappis propose that if $\alpha_t \notin (0, 1)$ then α_t is set to either 0 or 1, whichever one is closer.

A strong point of this predictor is that it calculates the optimal value of the α_t's for the last predictor-value combination. With high probability, that optimal choice of α_t is also a very good choice for the next value. Unfortunately, brief consideration of (17) suggests that the approach is unlikely to be of use. Since the one-step-ahead forecast is also the multi-step-ahead forecast for simple exponential smoothing, the numerator is a two-step-ahead forecast error, while the denominator is a one-step-ahead forecast error. This suggests that the expression in (17) will very often lead to values larger than 1. Following the rule of Pantazopoulos and Pappis (P&P), α_t would then take a value of 1. The result is that α_t very often takes a value of 1, which is shown by Figure 11. Moreover, this figure shows that the interpolation parameter α of the P&P predictor is quite unstable, and in many cases results in inaccurate predictions, see for example data points 171, 172, 185 and 186.

3.3 STES predictors

A smooth transition exponential smoothing (STES) method [18] has a smoothing parameter α_t defined as a logistic function of a user-specified transition variable, V_t. Mathematically, a STES method is defined as:

$$\hat{y}_t = \alpha_{t-1} y_{t-1} + (1 - \alpha_{t-1})\hat{y}_{t-1}, \tag{18}$$

where

$$\alpha_t := \frac{1}{1 + e^{\beta + \gamma V_t}}. \tag{19}$$

If $\gamma < 0$, then α_t is a monotonically increasing function of V_t. Hence, as V_t increases, the weight on y_t increases, and consequently, the weight on $\hat{y}_t$ decreases. The logistic function restricts α_t to lie between in the interval (0,1). Historical data is used to calibrate the adaptive smoothing parameter, α_t, through the estimation of β and γ in (19). The derived values for β and γ in (19), govern the degree to which the variation in the transition variable influences the STES smoothing parameter. The choice of the transition variable, V_t, is of crucial importance to the success of the method. Consideration of the adaptive methods described in the previous section leads to a number of different possible transition variables. The value of the smoothing parameter in all of the existing adaptive methods depends to varying degrees on the magnitude of the most recent periods forecast error. At the end of this section we describe the different methods we used: STES $|e|$, STES e^2, and STES Whybark.

In [18], 80 data points are used to fit the parameters, and those parameters are used to make one-step-ahead predictions of the next 20 values. To make a fair comparison between the STES- and the other predictors, we fitted the β and the γ after every new measured value. Moreover, for the STES predictors, we compared 10, 80, and “all values” as the number of data points used for the fit.

Next, we describe the three specific STES prediction methods we used in the predictor analysis.

STES |e|

An obvious choice for the transition variable is the absolute value of the forecast error from the most recent period:

$$V_t := |y_t - \hat{y}_t|. \tag{20}$$

In our analyses in Section 5 we make distinction between the situation in which we let the predictor fit β and γ freely, and in which we restrict the γ to be lower than 0. The purpose of restricting the γ to negative values is to get a high α value when the absolute error is high.

STES $\mathbf{e^2}$

Another obvious choice for the transition variable is the square of the forecast error from the most recent period:

$$V_t := (y_t - \hat{y}_t)^2. \tag{21}$$

STES Whybark

It is possible to use the α_t outcome values of all the AESPs. We choose to use the Whybark parameter as the transition variable, because short analysis showed that the Whybark parameter was the best transition variable to choose:

$$V_t := \begin{cases} H & \text{when } \delta_t = 1, \\ M & \text{when } \delta_t = 0 \text{ and } \delta_{t-1} = 1, \\ B & \text{otherwise,} \end{cases} \tag{22}$$

where

$$\delta_t := \begin{cases} 1 & \text{when } |y_t - \hat{y}_t| > 4\sigma, \\ 1 & \text{when } |y_t - \hat{y}_t| > 1.2\sigma,\ |y_{t-1} - \hat{y}_{t-1}| > 1.2\sigma, \text{ and } (y_t - \hat{y}_t)(y_{t-1} - \hat{y}_{t-1}) > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{23}$$

A strong point of a STES predictors discussed above is that it does not assume a linear relation between the transition variable and the value α_t. With the logistic function and the freedom of the parameters β and γ many different relations can be created. Another strong point is that the relation between the transition variable and the α_t's is fitted with two parameters, based on the history. Therefore, the α_t is adapted in a way that it is optimal according to the history. Moreover, when the transition variable has no correlation with the right α_t, the method automatically adapts the γ to 0 and the smoothing parameter will be constant. The STES method, therefore, enables re calibration of the existing adaptive methods.

A weak point, however, is that it is not clear why a logistic function would work better than other functions. Although previous experimental results have shown that logistic functions work quite well, it is the question whether it is an optimal function for the values of running times of jobs. Further on, this method uses the same formula for peaks as for stable situations. Figure 12 shows the predictions around a peak of three different STES predictors, which are described below. The figure illustrates that using the same parameters for peaks as for stable situations leads to bad predictions. Besides, the peaks have a high impact on the optimal parameters β and γ. Consequently, the α will behave in a stable situation highly influenced by previous peaks.

3.4 Other predictors

As stated in [25] a homeostatic or a tendency-based predictor can be very effective in predicting CPU load. A homeostatic predictor uses the assumption that if the current value is greater (less) than the mean of the history values, then the next value is likely to decrease (increase). A tendency-based predictor uses the assumption that when a current value is greater (less) than the mean of the history values, then the next value is likely to increase (decrease), because of a trend up- or downwards.
A disadvantage is that before the prediction process is started, a choice between homeostatic or tendency

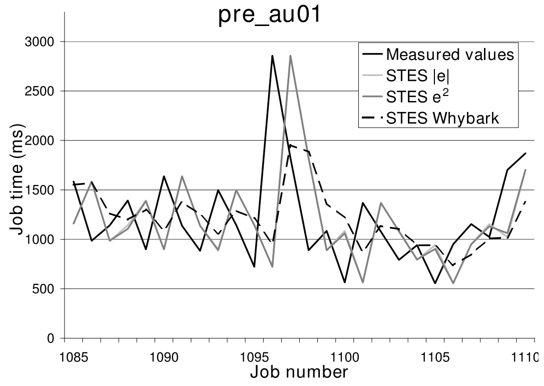


Figure 12: STES predictors reacting on a peak in pre_au01

has to be made. We know from the data analysis that it is possible that the characteristics in the dataset change, and that during the run the other type of predictor seemed to be more accurate. During our data analysis we noticed that trends do not appear very often in the datasets, there are many level switches, and sometimes homeostatic situations appear. For that reason we think that tendency-based predictors are not very accurate in predicting the running times of jobs. A homeostatic predictor will probably work fine for some datasets, but for the nodes with many level switches this method is very unlikely to be effective in the grid context. For these reasons, we have not implemented this prediction method.

Another well-known method is Linear Regression (LR). This method is quite similar to AR, when only historical data is used: the predictor is computed by multiplying historical data with parameters, which are fitted by using a least-squares method and the history. When not only data from the previous history is used, but also other information, like load, this method differs from AR. The parameter used to multiply with other data will also be fitted by its historical data. Since we focus on prediction methods based on the use historical data of running times of jobs only (see Section 1), LR is beyond the scope of this study.

Besides the linear prediction-methods, some experiments on a non-linear prediction method, called a Neural Network, have been performed. The results of the experiments illustrate that Neural Networks are computationally infeasible and are not accurate enough to be useful for prediction purposes in grids.

3.5 Conclusions of analyses

We conclude that although each prediction method has its strong points, none of the predictors properly reacts to peaks and level switches, which are two of the most important characteristics of the evolution of running times in a grid environment. This raises the need for the development of a new prediction method particularly suited for the specifics of a grid environment. For this reason, in the next section we use the insights in the pros and cons of the existing prediction methods to develop a new prediction method that effectively reacts to the peaks and level switches observed in a real grid environment.

4 New prediction method

4.1 DES prediction method

In this section we propose a method that effectively reacts to the peaks and level switches: the Dynamic Exponential Smoothing (DES) method. To this end, we take into account the benefits and drawbacks of the existing prediction methods. The basic idea of the DES prediction method is that it (1) treats different classes of fluctuations differently, (2) always computes the optimal parameter for those classes, and (3) has the possibility to select another predictor from a set, because of performance reasons. The parameter details are described below the mathematical description of the method. Furthermore, the method to derive the optimal historical α parameter, α_t^*, is described in Section 4.2. Finally, in Section 4.3 we discuss the effectiveness of the proposed predictor. We propose the following complete DES prediction method for $t = 2, 3, \ldots$:

$$\hat{y}_t = \begin{cases} \hat{y}_{DES,t} & \text{if } \kappa_{DES} = min_{j\epsilon\{DES,\mu,median\}}(\kappa_j), \\ \hat{y}_{\mu,t} & \text{if } \kappa_{\mu} = min_{j\epsilon\{DES,\mu,median\}}(\kappa_j) \neq \kappa_{DES}, \\ \hat{y}_{median,t} & \text{else,} \end{cases} \tag{24}$$

with

$$\hat{y}_{DES,t} := \alpha^*_{t-1} y_{DES,t-1} + (1 - \alpha^*_{t-1})\hat{y}_{DES,t-1}, \tag{25}$$

$$\hat{y}_{\mu,t} := \mu((y_1, \ldots, y_{t-1})), \tag{26}$$

$$\hat{y}_{median,t} := median((y_{t-l}, \ldots, y_{t-1})), \tag{27}$$

$$\kappa_i := \sum_{s=1}^{t-1}(y_s - \hat{y}_{i,s})^2, \ i\,\epsilon\,\{DES, \mu, median\}, \tag{28}$$

where

$$\alpha^*_t := \frac{1}{\sum_{s=2}^{t}(y_{s-1} - \hat{y}_{s-1})^2 1_{\{\delta_s=\delta_t\}}} \sum_{s=2}^{t} \alpha_{s-1,opt}(y_{s-1} - \hat{y}_{s-1})^2 1_{\{\delta_s=\delta_t\}}, \tag{29}$$

and for $l < t$:

$$median((y_{t-l}, \ldots, y_{t-1})) := \begin{cases} x_{(l+1)/2} & \text{if } l \text{ is odd,} \\ \frac{1}{2}(x_{l/2} + x_{1+l/2}) & \text{if } l \text{ is even,} \end{cases} \tag{30}$$

where

$$\alpha_{s-1,opt} := \frac{y_s - \hat{y}_{s-1}}{y_{s-1} - \hat{y}_{s-1}} \tag{31}$$

$$\delta_s := \begin{cases} H1 & \text{if} & |y_s - \hat{y}_s| & > 10\sigma((y_{s-k}, \ldots, y_{s-1})), \\ H2 & \text{if} & (y_s - \hat{y}_s) & > 2\sigma((y_{s-k}, \ldots, y_{s-1})), \\ H3 & \text{if} & (y_s - \hat{y}_s) & < -2\sigma((y_{s-k}, \ldots, y_{s-1})), \\ M & \text{if} & |y_s - \hat{y}_s| & > \sigma((y_{s-k}, \ldots, y_{s-1})), \\ & & |y_{s-1} - \hat{y}_{s-1}| & > \sigma((y_{s-k}, \ldots, y_{s-1})), \\ & \text{and} & (y_s - \hat{y}_s)(y_{s-1} - \hat{y}_{s-1}) & > 0, \\ B & \text{otherwise.} \end{cases} \tag{32}$$

$$x_1 := min((y_{t-l}, \ldots, y_{t-1})), \ldots, x_l := max((y_{t-l}, \ldots, y_{t-1})). \tag{33}$$

Note that $\mu((y_1, \ldots, y_{t-1}))$ is defined in (3), and $\sigma((y_{t-k}, \ldots, y_{t-1}))$ is defined in (4).

As can be seen in (24), the DES prediction method selects the best prediction method from a set of 3 predictors, by comparing the sum of the squared errors of the previous measurements. The set contains the running average, the l-sliding window median, and the DES predictor. The l-sliding window median takes the median of the last l measurements. The DES-predictor part classifies different types of fluctuations: 3 huge fluctuations classes ($H1$, $H2$, and $H3$), 1 medium fluctuations class (M), and 1 base class (B). When the deviation between the measured value and its prediction is more than 10 times the standard deviation of the last k values, the measurement is of class $H1$. Furthermore, when the deviation between the measurement and the prediction is higher than two times (smaller than minus two times) the standard deviation, the measurement is of class $H2$ ($H3$). Two consecutive measurements that both deviate in the same direction with more than standard deviation from the predictions are of class M. B is the base class and is defined for the rest of the cases. This classification is defined in (32). When a measurement fits into multiple classes, the first mentioned class in (32) will be chosen. Subsequently, the DES predictor computes the prediction by using the exponential-smoothing equation with an α parameter that equals the weighted average of the optimal α's (see (29)) of the previous measurements that are of the same class as the most recent measurement. Formula (29) is derived below.

We implemented the predictor with $k = 20$, and $l = 31$, and fixed the maximal number of data points to estimate each of the optimal α values for classes $H1$, $H2$, $H3$, M and B at 500. When no fluctuations of a certain type were registered before, the default value of 0.5 is taken. On the one hand, when k is higher than 20, the predictor reacts slower on changes in standard deviations. On the other hand, when less values are taken, the estimated standard deviations get less accurate. While 20 is a good value for k, a higher or lower value does not affect the results significantly. The value 31 as the value for l corresponds to one of the set of medians in [22]. Accuracy comparisons showed that this sliding-window median performs better than medians with other l parameters. The same holds for the choice to use all previous measurements for the computation of the average. The number 500 ensures that on the one hand the α_t^* is reliable and stable, because it is computed by enough measurements, and that on the other hand the α_t^* is still able to adapt to new characteristics in the dataset.

4.2 Computation of α_t^*

In this section, we derive (29). To this end, let $\hat{y}_{s,opt}$ be the best possible prediction of y_s, for $s \epsilon \{2,\ldots,t\}$. Then by definition,

$$\hat{y}_{s,opt} = y_s. \tag{34}$$

Moreover, let $\alpha_{s-1,opt}$ be the best possible parameter for prediction of y_s. Then

$$\hat{y}_{s,opt} = \alpha_{s-1,opt} y_{s-1} + (1-\alpha_{s-1,opt})\hat{y}_{s-1}. \tag{35}$$

We compute the optimal exponential smoothing parameter for the prediction of y_s, $\alpha_{s-1,opt}$, by solving (35) and replacing y_s by $\hat{y}_{s,opt}$, according to (34). We derive the following $\alpha_{s-1,opt}$.

$$\alpha_{s-1,opt} = \begin{cases} \frac{y_s-\hat{y}_{s-1}}{y_{s-1}-\hat{y}_{s-1}} & \text{if } y_{s-1}-\hat{y}_{s-1} \neq 0, \\ 0 & \text{else.} \end{cases} \tag{36}$$

At a given time step s, if y_s is known, we are able to calculate the Squared Error (SE_s) of one prediction in the following way:

$$\begin{aligned} SE_s &= (y_s-\hat{y}_s)^2 = (\hat{y}_{s,opt}-\hat{y}_s)^2 && (37)\\ &= (\alpha_{s-1,opt}y_{s-1}+(1-\alpha_{s-1,opt})\hat{y}_{s-1}-(\alpha_{s-1}y_{s-1}+(1-\alpha_{s-1})\hat{y}_{s-1}))^2 && (38)\\ &= (\alpha_{s-1,opt}-\alpha_{s-1})^2(y_{s-1}-\hat{y}_{s-1})^2. && (39) \end{aligned}$$

These formulas hold for all $s \in \{1,\ldots,t\}$, with t the current time step. Subsequently, we derive the total MSE of all predictions y_s, for all $s \in \{1,\ldots,t\}$. To this end, we assume that $\alpha_s = \alpha$ for all $s \epsilon \{2,\ldots,t\}$.

$$MSE_t = \frac{1}{t-1}\sum_{s=2}^{t}(\alpha_{s-1,opt}-\alpha)^2(y_{s-1}-\hat{y}_{s-1})^2. \tag{40}$$

Next, we calculate α_t^*, i.e., the value of α in (40) that minimizes the MSE_t, by taking the setting derivative of (40) to α to 0. This leads to the following expression for α_t^*:

$$\alpha_t^* = \frac{1}{\sum_{s=2}^{t}(y_{s-1}-\hat{y}_{s-1})^2}\sum_{s=2}^{t}\alpha_{s-1,opt}(y_{s-1}-\hat{y}_{s-1})^2. \tag{41}$$

Note that this is equivalent to the weighted average of the values of $\alpha_{s-1,opt}$, with weights $(y_{s-1}-\hat{y}_{s-1})^2$, for $s=2,\ldots,t$. Finally, to derive (29), an indicator $1_{(\delta_s=\delta_t)}$ is added to (41) to distinguish between the different classes.

4.3 Discussion

Formula (24) shows the selection algorithm of the DES prediction method. As we have seen for the NWS method in Section 3.1.2, the datasets sometimes show characteristics for which the average or the sliding median with window size 31 gives the most accurate predictions. Around 10 % of the measurements show these kinds of characteristics. Analysis show that measuring the κ_i's (see Formula (28)) is an effective way of comparing the different predictors: in all the 10 % of the cases the average or the sliding-window median is chosen. We note that comparing the κ_i's is the same as comparing the Root of the Mean Squared Errors (as will be described in Section 5).

In Section 2, we observed that the datasets may have completely different and continuously changing characteristics. Therefore, as can be seen in Formula (29)), we chose to adapt the α_t's to the situation of the characteristics. We concluded from the analysis of the AESPs, the STES predictors and the AR method in Section 3 that adapting the α_t to the characteristics is an effective way to make the predictor robust against all the completely different and ever-changing characteristics of the datasets. However, we showed that improvements were possible on the following aspects: the choice of the α_t was not always clear and the α_t's are unstable in many predictors. The choice of taking the optimal α of the measured data is very clear and leads to stable α_t's within the classes of fluctuations.

Next, as shown in (32) we classified different types of fluctuations. We concluded in Section 3.2.2 that Whybark consists of useful classification elements, but that improvements are necessary to be made to develop a useful classification of running times. We incorporated those improvements in our prediction method, as can be seen in (32).

Moreover, easy to measure statistics can be included in the model that compute the MSE and the RMSE of the DES predictor by computing the squared differences between the predictions from (24) and the data measurements. These statistics can be applied in an online scheduling system that dynamically decides on the basis of expectations of job runtimes which nodes in the resource set are used.

Finally, we would like to address that this method can be extended for more general situations. We have planned to make those analyses in further research, which is described in Section 6.

5 Experimental Results

In this section we compare the performance of the DES prediction method with those of the predictors described in Section 3.

5.1 Quality metrics for prediction methods

We need to define a metric that quantifies the improvement of a predictor in comparison to another predictor. As we have seen in Figure 5(e), jumps have a strong impact on the standard deviation or the RMSE of the dataset, which is highly correlated with the standard deviation. When the peaks or level switches do not show up periodically, it hard to predict when they occur. Consequently, jumps always have a strong effect on the RMSE of a predictor. For that reason, we are interested in a measure that is able to compare the performance of different predictors and is independent of the influence of the jumps on the standard deviation. Therefore, to compare different predictors we define $\Delta\%(A, B)$ to be the percentage improvement of predictor A versus B:

$$\Delta\%(A, B) := \frac{RMSE_B - RMSE_A}{RMSE_B - RMSE^*} * 100\%, \tag{42}$$

where the value $RMSE^*$ is the optimal *postcast* selected from the set of the NWS predictors. It indicates the theoretically maximal forecasting performance (minimum error) that the method could have achieved if the best predictor at each step was known (see for more details about the $RMSE^*$ the definition of the *Optimum* in [22]). When a jump occurs, even the optimal forecasting method has the property that it was not able to predict that jump. But for the successive measurement, the optimal forecast mostly has a low RMSE. Consequently, the $RMSE^*$ is the RMSE that a prediction method would have when it would predict the behavior after jumps perfectly.

5.2 Comparison results

To compare the performance of DES with the existing prediction methods, we have gathered experimental data from 45 datasets, as discussed in Section 2. We first compare the predictors on the basis of the time that it takes to compute each prediction. Second we extensively compare their accuracies. The results are outlined below.

Table 5 shows us the ranking for the different prediction methods in terms of the time that it

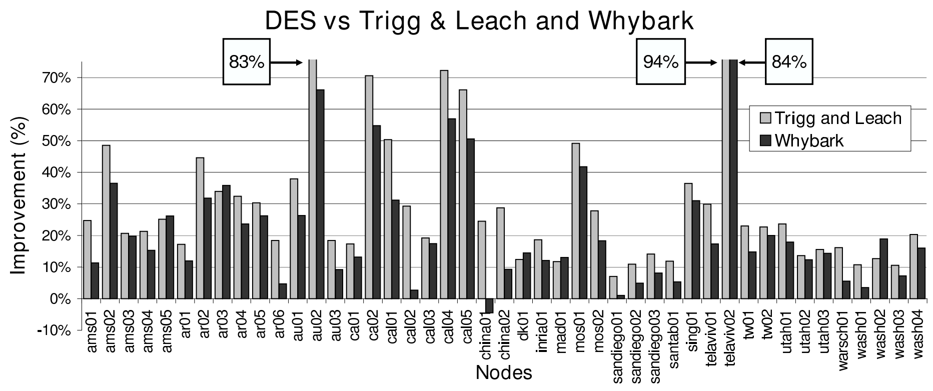


Figure 13: DES-predictor improvements compared to the Trigg & Leach and the Whybark predictor

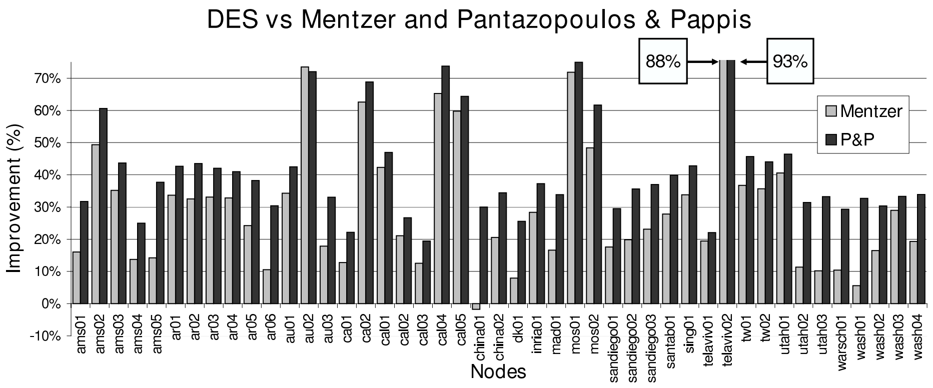


Figure 14: DES-predictor improvements compared to the Mentzer and Pantazopoulos & Pappis predictor

takes to compute the prediction for the next value, and it shows the number of computations needed per prediction. For the NWS and the AR predictor those times have been measured in [5] and [23], which is shown by the last column. The table illustrates that the DES needs to perform a lower number of computations than the NWS, which takes 161 microseconds on a Pentium III laptop. Consequently, we conclude that the computational costs of the DES predictor are low enough such that it can feasibly be implemented in online systems.

Rank	**Pred**	**Computations per prediction**	**Time**
1	ES	two multiplications and one summation	$<$ 161 microseconds
2	AESP	number of multiplications and summations	$<$ 161 microseconds
3	DES	number of multiplications and summations (order of ten)	$<$ 161 microseconds
4	NWS	number of multiplications and summations (order of hundred)	[23]: 161 microseconds on an unloaded 750 MHz Pentium III laptop.
5	AR	fitting procedure	[5]: 1.4 ms on an unloaded 500 MHz Alpha 21164-based workstation
6	STES	fitting procedure	$>$ 1.4 ms

Table 5: Comparison of the computational costs of the prediction methods

Next, we compare the accuracy of the different prediction methods. Figure 13 shows the performance improvements of the DES prediction method compared to the Trigg and Leach predictor and the Whybark predictor. Similarly, Figure 14 shows the results for DES compared to the Mentzer and Pantazopoulos & Pappis predictor.

The results presented in Figures 13 and 14 show that the DES prediction method strongly and consistently outperforms the other predictors. The improvements are remarkably high: for many

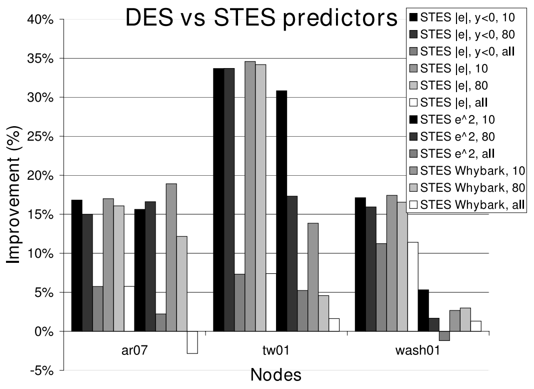


Figure 15: DES-predictor improvements compared to the STES predictors

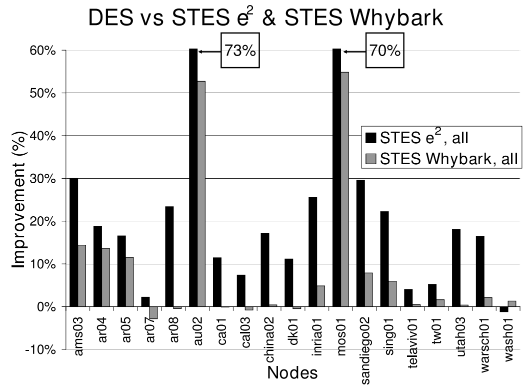


Figure 16: DES-predictor improvements compared to the STES e^2 and STES Whybark predictors

datasets the DES prediction method shows improvements of more than 30%. From the four AESP predictors considered here, the Whybark predictor gives the most accurate predictions, but is still strongly outperformed by our DES predictor.

Figure 15 shows the prediction performance of in total 12 STES predictor-parameter combinations for the three randomly-chosen datasets ar07, tw01, and wash01. We tested the STES |e| predictor with parameter $\gamma < 0$, the STES |e| predictor with no restrictions on γ, the STES e^2 predictor and the STES Whybark predictor. For all the four predictors we used 10, 80 and 2000 data points for the fit. We tested those predictors only with three datasets due to the fact that the STES predictors take too much time (a whole day per dataset). We observe considerable differences in the prediction results of the different methods. Although three datasets are not enough to draw conclusions that are statistically significant, we expect because of those significant differences that the following observations hold for almost all the datasets. The results in Figure 15 show that it is always better to use all the data (at least 2000 values) for the parameter fit, and that the STES e^2 and the STES Whybark are the best STES predictors. Those methods need further comparison analysis with more datasets to conclude more about the quality of their predictions. We used the following randomly-chosen datasets for the further analysis: ams03, ar04, ar05, ar07, ar08, au02, ca01, cal03, china02, dk01, inria01, mos01, sandiego02, sing01, telaviv01, tw01, utah03, warsch01, and wash01.

In Figure 16 we compare the improvements of the DES prediction method in comparison with the STES e^2 and the STES Whybark predictor for all those datasets. We clearly observe that the DES prediction method outperforms the other two predictors. For two of the datasets the DES shows even more than 50% improvements for both of the other predictors.

Figure 17 below shows the improvements of the DES prediction method compared to the ES predictor with $\alpha = 0.5$, denoted as the ES(0.5) predictor. Despite the fact that DES is based on ES, the method shows a completely different accuracy: the difference between the RMSEs of the ES(0.5) predictions and the DES predictions ranges from -17% to $+70\%$ and differs for each dataset. Only for

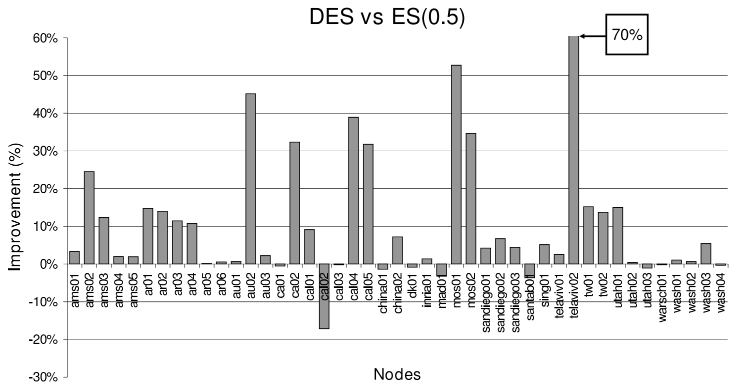


Figure 17: DES-predictor improvements compared to the ES(0.5) predictor

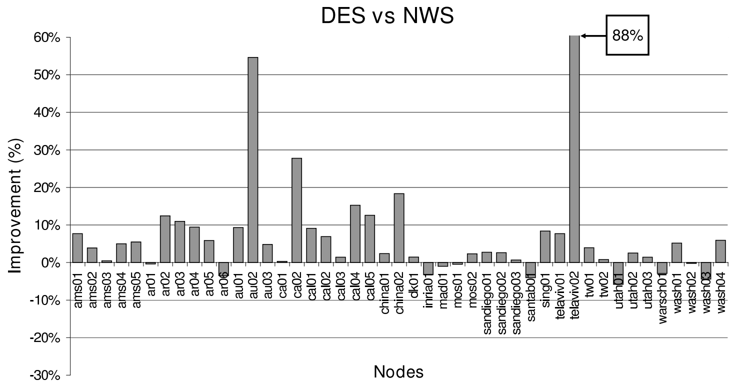


Figure 18: DES-predictor improvements compared to the NWS prediction method

single dataset the ES(0.5) predicts significantly better than the DES. On average the DES shows 11% improvement.

Figure 18 illustrates the improvement of the DES prediction method in comparison with the NWS prediction method. The figure shows that DES mostly outperforms the NWS prediction method. For 6 of the 45 datasets the NWS prediction method is only 1 to 5 percent more accurate than the DES prediction method. However, on average 8% improvements can be performed by implementing the DES prediction method instead of the NWS prediction method.

Figure 19 illustrates the performance improvement of the DES prediction method in comparison with the AR(16) predictor. Figure 19 shows more fluctuations than Figure 18, because the DES and the NWS prediction methods have more similarities. For five datasets the AR(16) predictor shows more than 5% accuracy improvements compared to the DES prediction method. In fourteen cases the DES prediction method shows more than 10% improvement. For three datasets the DES prediction method is even more than 40% more accurate than the AR(16) predictor. On average, the DES prediction method is 9% more accurate than the AR(16).

As we see in Figures 17 to 19 DES consistently outperforms all other predictors for datasets au02 and telaviv02. Further analysis show that the jumps in those datasets have a huge influence on the standard deviation. We found that the main reason for the enormous performance improvement is that the DES prediction method is the only method that reacts properly on the high jumps. Nevertheless, when we filter out the jumps with a deviation above 10 times the standard deviation the DES still clearly outperforms the others predictors.

Next, we investigate the reason for the performance improvement of the DES prediction method in comparison with the best of the other predictors, the NWS prediction method. To this end, we analyze the situations in the datasets where DES clearly outperforms NWS. We observe two main reasons why DES outperforms the NWS prediction method: (1) DES reacts more properly on peaks,

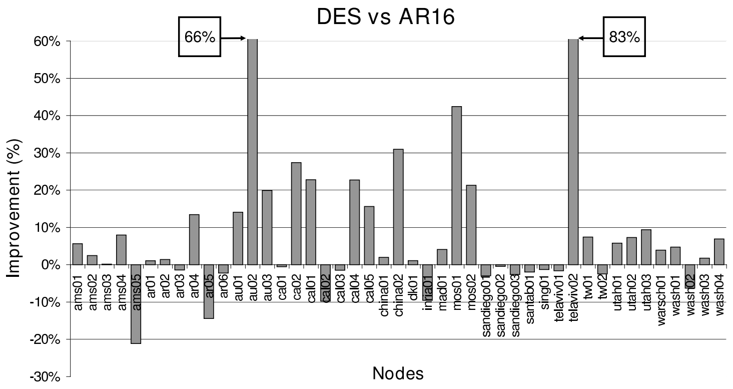
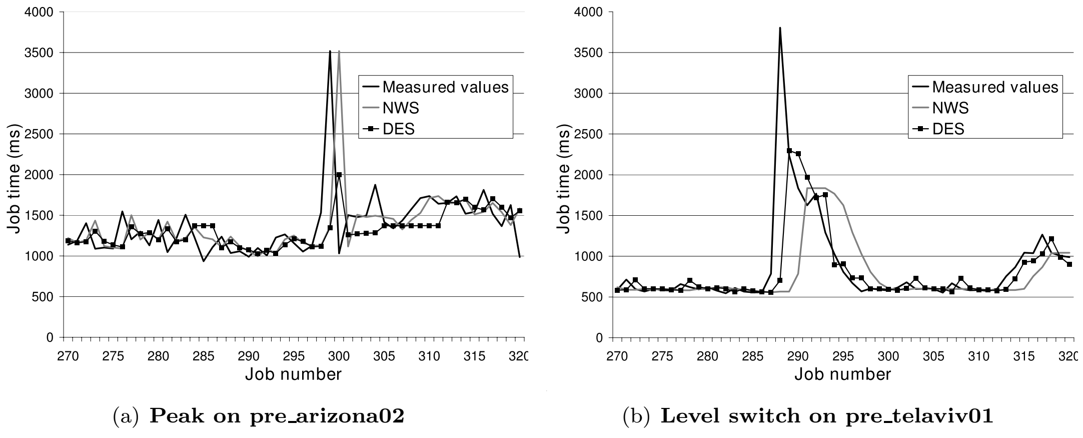


Figure 19: DES-predictor improvements compared to the AR16 predictor

and (2) DES reacts more properly on level switches. To investigate this more carefully, we add the predictions of the DES prediction method in Figure 7. The results are graphically represented in Figure 20.



(a) **Peak on pre_arizona02** (b) **Level switch on pre_telaviv01**

Figure 20: DES- and NWS prediction method reacting on a peak and a level switch in the running times

We notice the following from Figure 20(a). Due to the values before the jump, the NWS chooses a predictor that gives a high weight to the last value. When the jump appears, the NWS prediction for the next running time also gets very high, even though there is a high probability that it is a peak, because almost all the jumps in history were peaks and not level switches. The DES prediction method also gives a prediction that is slightly too high due to the fact that one jump in history did introduce a new level. However, as can be seen in Figure 20(a) at data points 300-312, the DES predictor sometimes shows a too stable pattern after a peak. In that case DES interprets every fluctuation as a jump without level switch and therefore does not react on the fluctuations. But, nevertheless, the DES prediction method is far more accurate than the NWS prediction method.

In Figure 20(b), we have a different situation: jumps introduce a new level of running times. In this example we see that the NWS also shows a bad performance. Due to the fact that the values before the jumps need a predictor that has a more averaging property, the NWS also uses the average to predict the value after the jump despite the fact that the jumps in history very often introduced a new level. It takes three values (i.e., jobs durations) before the NWS prediction method realizes that the predictions are not accurate. The DES prediction method clearly shows a better pattern of predictions in the level switch.

Despite the two previous mentioned situations where the NWS prediction method does not react very properly on jumps, there are a lot of situations where the NWS prediction method does react appropriately. There are three situations where the NWS does react very accurately: (1) when a more averaging predictor is chosen because of the values before a jump, and it does not introduce a new level, (2) when a last value predictor is chosen before a jump and the jump introduces a new level, and (3) when the jumps appear in quick succession, the NWS still *remembers* what the best predictor was during the last jump, because jumps have a high impact on the RMSE, and therefore on the choice of the predictor.

To summarize, the comparison results show that in general the DES prediction method outperforms the existing predictions methods. Another well performing predictor for predicting running times of jobs is the NWS prediction method.

6 Conclusions and challenges

Extensive statistical analysis of experimental datasets generated from a real, global-scale, grid environment has shown that (1) the characteristics of the datasets differ enormously from node to node, (2) the datasets contain in general more long- than short-term fluctuations and the proportion differs per dataset, (3) the datasets illustrate level switches (i.e., changes in the average) during the run, with differences in the amount of level switches between the different nodes, (4) the standard deviations fluctuate significantly during the run, with differences in the amount of fluctuations between the different nodes, and (5) the datasets contain a small amount of jumps, which have a considerable influence on the standard deviation, the variance, and the total amount of fluctuations. This has raised the need for the development of effective methods for prediction of running times that are able to deal with those characteristics. We have analyzed the performance of a variety of prediction methods. Our results show that none of these methods is well-suited for dealing with the specifics of running times in a grid environment, including the presence of level switches and sudden peaks. To this end, we have developed a new prediction method, called DES, that overcomes the shortcomings of the existing methods by properly reacting to level switches and peaks. The power of DES lays in the fact that it is able to deal with both peaks and level switches.
Extensive comparisons with a large number of datasets show that DES is a highly effective method for predicting running times of jobs on shared processors which are heterogeneous. DES consistently outperforms the common grid prediction methods, such as the NWS predictor, AR(16), and ES, and moreover, gives much more accurate predictions than the Adaptive Exponential Smoothing Predictors Trigg and Leach, Whybark, Mentzer, Pantazopoulos and Pappis, and four kinds of promising Smooth Transition Exponential Smoothing predictors. Consequently, the predictor has proved its robustness against the completely different characteristics of the job runtimes that are measured on heterogeneous nodes.

The results presented in this paper lead to a number of challenges for further research. First, it is interesting to investigate the correlations between the job runtimes on different processors. An effective statistical method to quantify these relations is the Cross Correlation Function (CCF). We plan to compute the CCFs of the job runtimes on different nodes and make contour plots, which will provide insight in those correlations. Second, there is room for refinement of the DES predictor. For the running times of jobs, the current choice of parameters for the heights of jumps (i.e. a difference between two consecutive values of more than 2 times the standard deviation is a jump, and more than 10 times the standard deviation is a high jump) leads to significant improvements in the predictions. We expect that those parameters partly depend on the properties of the dataset. We plan to investigate whether it is possible to adapt these parameters to the statistical properties of the datasets. Consequently, this means that predictors for job runtimes which are measured on different nodes choose different values for the above mentioned parameters. Moreover, this strategy can be applied if the job size is adapted. For this case it is reasonable that the level switches and peaks change their shapes and therefore it is necessary that the parameters adjust to the new characteristics. Third, we aim to apply the DES prediction method for predicting other types of grid property measurements (e.g., latency, CPU utilizations) or in other large-scale grid environments. Finally, the ultimate goal of the development of effective prediction methods in the context of the computational grid is to decrease the effective running times of distributed applications by triggering effective load re-balancing actions. Therefore, the next step is to quantify the actual improvements that can be obtained in the running times of distributed applications, which addresses a challenging area for further study.

References

[1] http://www.planet-lab.org.

[2] H. Dail, G. Obertelli, F. Berman, R. Wolski, and A. Grimshaw. Application-aware scheduling of a magnetohydrodynamics application in the legion metasystem. In *Proceedings of the 9th Heterogeneous Computing Workshop (HCW '00)*, pages 216–228, Cancun, Mexico, May 1, 2000. IEEE Computer Society.

[3] P. A. Dinda. Online prediction of the running time of tasks. *Cluster Computing*, 5(3):225–236, July 2002.

[4] P. A. Dinda. A prediction-based real-time scheduling advisor. In *Proceedings of the 16th International Parallel and Distributed Processing Symposium (IPDPS '02)*, pages 35–42, Fort Lauderdale, FL, USA, April 15-19, 2002. IEEE Computer Society.

[5] P. A. Dinda and D. R. O'Hallaron. Host load prediction using linear models. *Cluster Computing*, 3(4):265–280, 2000.

[6] A. M. Dobber, G. M. Koole, and R. D. van der Mei. Dynamic load balancing for a grid application. In *Proceedings of the 11th International Conference on High Performance Computing (HiPC '04)*, pages 342–352, Bangalore, India, December 19-22, 2004. Springer-Verslag.

[7] A. M. Dobber, G. M. Koole, and R. D. van der Mei. Dynamic load balancing experiments in a grid. In *Proceedings of the 5th IEEE International Symposium on Cluster Computing and the Grid (CCGrid '05)*, pages 123–130, Cardiff, Wales, UK, May 9-12, 2005. IEEE Computer Society.

[8] A. M. Dobber, R. D. van der Mei, and G. M. Koole. Effective prediction of job processing times in a large-scale grid environment. In *Proceedings of the 15th IEEE International Symposium on High Performance Distributed Computing (HPDC '06) (poster)*, pages 359–360, Paris, France, June 19-23, 2006. IEEE Computer Society.

[9] A. M. Dobber, R. D. van der Mei, and G. M. Koole. Statistical properties of task running times in a global-scale grid environment. In *Proceedings of the 6th IEEE International Symposium on Cluster Computing and the Grid (CCGrid '06)*, pages 150–153, Singapore, May 16-19, 2006. IEEE Computer Society.

[10] R. Fildes. Quantitative forecasting-the state of the art: extrapolative models. *Journal of the Operational Research Society*, 30:691–710, 1979.

[11] L. Kleinrock. Time-shared systems: A theoretical treatment. *Journal of the Association for Computing Machinery*, 14:242–261, 1967.

[12] B. Lee and J. Schopf. Run-time prediction of parallel applications on shared environments, poster paper. In *Proceedings of the 5th IEEE International Conference on Cluster Computing (Cluster '03) (poster)*, pages 487–491, Kowloon, Hong Kong, China, December 1-4, 2003. IEEE Computer Society.

[13] C. Liu, L. Yang, I. Foster, and D. Angulo. Design and evaluation of a resource selection framework for grid applications. In *Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing (HPDC '02)*, pages 63–75, Edinburgh, Scotland, UK, 2002. IEEE Computer Society.

[14] J. T. Mentzer. Forecasting with adaptive extended exponential smoothing. *Journal of the Academy of Marketing Science*, 16:62–70, 1988.

[15] S. N. Pantazopoulos and C. P. Pappis. A new adaptive method for extrapolative forecasting algorithms. *European Journal of Operational Research*, 94:106–111, 1996.

[16] Y. Qiao and P. Dinda. Network traffic analysis, classification, and prediction. Technical Report NWU-CS-02-11, Department of Computer Science, Northwestern University, January 2003.

[17] K. H. Shum. Adaptive distributed computing through competition. In *Proceedings of the 3th IEEE International Conference on Configurable Distributed Systems (CDS '96)*, pages 200–227, Annapolis, MD, USA, May 6-8, 1996. IEEE Computer Society.

[18] J. W. Taylor. Smooth transition exponential smoothing. *Journal of Forecasting*, 23:385–404, 2004.

[19] D. W. Trigg and A. G. Leach. Exponential smoothing with an adaptive response rate. *Operations Research Quarterly*, 18:53–59, 1967.

[20] J. W. Tukey. *Exploratory Data Analysis*, pages 39–43. Addison-Wesley, Reading, MA, 1977.

[21] D. C. Whybark. Comparison of adaptive forecasting techniques. *Logistics Transportation Review*, 8:13–26, 1973.

[22] R. Wolski. Forecasting network performance to support dynamic scheduling using the Network Weather Service. In *Proceedings of the 6th IEEE International Symposium on High Performance Distributed Computing (HPDC '97)*, pages 316–325, Portland, OR, USA, August 5-8, 1997. IEEE Conputer Society.

[23] R. Wolski. Experiences with predicting resource performance on-line in computational grid settings. *SIGMETRICS Performance Evaluation Review*, 30(4):41–49, 2003.

[24] R. Wolski, N. T. Spring, and J. Hayes. Predicting the CPU availability of time-shared unix systems on the computational grid. *Cluster Computing*, 3(4):293–301, 2000.

[25] L. Yang, I. Foster, and J. M. Schopf. Homeostatic and tendency-based cpu load predictions. In *Proceedings of the 17th International Symposium on Parallel and Distributed Processing (IPDPS '03)*, pages 42–50, Nice, France, April 22-26, 2003. IEEE Computer Society.