Meeting Deadlines in Complex Systems: A Probabilistic Approach

Günther A. Hoffmann and Miroslaw Malek

Humboldt University Berlin, Department of Computer Science,

Computer Architecture and Communication Group, 10099 Berlin, Germany

[gunho|malek]@informatik.hu-berlin.de

May 2004

Abstract *Investigations into deadline guarantees in real-time systems have traditionally been dominated by scheduling using static and analytical approaches. This framework has been successfully applied to a wide range of applications. However, many real-world applications are too complex for such analysis and cannot be subject to absolute guarantees. Rigid analytical approaches are not practical when confronted with the degree of complexity and degrees of freedom of real software systems in use. In this paper we propose and employ a novel modeling technique based on Universal Basis Functions (UBF) to give probabilistic estimates of a system holding its deadlines (responsiveness). The algorithm is non-parametric and data driven rather than analytical. It builds a model of the system based on observations of system variables. We first show the practicality of our learning algorithm by modeling responsiveness of a simulated queuing system and by predicting CPU demand of the Baum-Welch algorithm based on its parametrization. We derive precise models to predict resource demand a-priori and give accurate responsiveness estimates for the queuing system. We then apply our modeling technique to real data of a commercial telecommunication platform. Deadlines in this system are introduced as serving incoming calls within a prespecified time frame. The data investigated includes time-continuously measured system states. We build probabilistic models and derive conditional probabilities of calls being processed within their prespecified deadline for given time intervals. Results are presented in terms of responsiveness, precision, recall and F-Measure. The models and results have been validated on synthetic as well as on a distributed real-world computing system. Our findings suggest significantly improved forecasting of timeliness compared to alternative approaches.*

Keywords responsiveness, timeliness, failure prediction, distributed real-time systems, non-parametric data-based modeling, stochastic modeling, universal basis functions (UBF)

Contact author: Günther A. Hoffmann (gunho@informatik.hu-berlin.de)

1 Introduction

Investigations into deadline guarantees in real-time systems have traditionally been dominated by scheduling using static and analytical approaches. This framework has been successfully applied to a wide range of applications. However, many real-world applications are too complex for such analysis and cannot be subject to absolute guarantees. Rigid analytical approaches are not practical when confronted with the degree of complexity and degrees of freedom of real software systems in use. Particularly fault tolerant systems can be inherently stochastic and can not be subject to absolute guarantees. Furthermore software systems in industrial environments need to be flexible and adaptive. Algorithms with a great variance in their parametrization and application domain are becoming more wide spread. Nonetheless we would like to give quantifiable statements about meeting deadlines in these systems.

In addition to meeting temporal constrains, introduced as deadlines, industrial real-time systems also must be fault tolerant and performant. For example architectural features such as caches, pipelines, out-of-order executions and branch predictions are implemented by these systems for performance reasons. Fail over, checkpointing, restart and rejuvenation techniques are implemented for fault tolerance. However, by introducing these features also stochastic dynamics are induced into the system thus decreasing the predictability of these systems by traditional methods. Particularly in large industrial software systems deadline guarantees have to be given to the customer despite the fact that rigid analytical approaches are non-manageable and non-applicable.

One approach to get quantifiable statements about meeting deadlines in the presence of fault tolerance mechanisms and performance optimising strategies is to observe the dynamics of the system, derive a statistical model and get probabilistic statements about deadlines. Indeed this is the approach we take in this paper.

We propose and employ a novel modeling technique based on Universal Basis Functions (UBF) to give probabilistic estimates of a system holding its deadlines. The algorithm is non-parametric and data driven rather than analytical. It builds a model of the system based on observations of system variables. We first show the practicability of our algorithm by modeling responsiveness of a simulated queuing system and by predicting CPU demand of the Baum-Welch algorithm based on its

parametrization. We then apply our modeling technique to real data of a commercial telecommunication platform. Deadlines in this system are introduced as serving incoming calls within a prespecified time frame.

The paper is structured as follows. In the next chapter we review related work on responsiveness and modeling software systems and further motivate our approach. In Chapter 3 we introduce our novel modeling technique and describe the modeling and forecasting objectives. To show the practicality of our approach we present two experiments with synthetic data in Chapter 4. In the first experiment we model and predict CPU usage of the Baum-Welch Algorithm, which is non-linearly dependent on its two input parameters. In the second experiment we model and predict responsiveness of a G/G/N queuing system. In Chapter 5 we apply our modeling technique to a real-world distributed telecommunication system. We derive conditional probabilities of calls being processed within their prespecified deadline. In Chapter 6 we present and discuss our results.

2 Related Work and Motivation

To predict the future state of any computing system based on observations about its dynamics we need methods that can handle the complexity of the, possibly non-linear, interactions between its components. One approach is to use rigorous methods. Rigorous methods imply to formally specify the properties of a systems intended behaviour and verify that the system conforms to that specification. Examples for rigorous methods include temporal logic and process algebra. The approach advocated in this paper is to observe the system in question and gather data which is believed to describe the functional relationship between elements of the system. Then use the gathered data to

1) describe and model the timeliness and functional behaviour of the system in question in terms of time series modelling and forecasting.
2) describe and model the probabilistic behaviour of the system in question in terms of conditional probability densities.

The dynamics of distributed real-time systems can be observed as some time series of measurements describing the evolution of the system over time. An example of such a time series can be given by resource utilization such as queue length, page faults or the number of page readings. The problem that is imposed by this approach is reduced to finding a procedure which can automatically detect the

functional relationship between measured data and which is also immune to noise corruption, limited data accessibility and non-linear dependencies. One approach to this type of statistical analysis is non-linear, non-parametric data analysis [Hertz et al (1991)][Weigend et al. (1994)][Bishop (1995)]. Statistical analysis of non critical computing systems has been carried out by [Terrasa et al. (2003)]. They analyze run-time traces. Data obtained by measurement is subjected to statistical analysis by [Burns et al. (2000)]. The authors model computation times and extent their approach to provide probabilistic scheduling [Burns et al. (2003)]. This work is related to our approach of modeling and predicting CPU requirements before the algorithm is scheduled. A probabilistic analysis technique has been developed by [Gardner (1999)]. The author introduces Stochastic Time Demand Analysis for determining a lower bound on the rate at which deadlines are met in fixed priority systems. Probabilistic models predicting resource demand of an algorithm, based on observations of its historic behavior are developed in [Hoffmann (1996)].

Besides scheduling, responsiveness issues and data based modeling techniques are closely related to our approach. We briefly revisit these two important topics in the next paragraphs.

2.1 Data-based Modeling

Literature on measurement-based approaches to modeling software systems has been dominated by approaches limited to a single or few observable variables not considering feature detection methods. Most statistical approaches focus on modeling the evolution of specific system variables not deadline guarantees. The majority of models also are either based on observations about workload, time, memory or file tables. A time-based model for detection of software aging is proposed by [Garg et al. 1998]. A workload-based model for prediction of resource exhaustion in operating systems is introduced by [Vaidyanathan et al. 1999]. The authors conclude that measurement-based software rejuvenation outperforms simple time statistics. [Li et al. 2002] collect data from a web server which they expose to artificial workload. They build linear ARMA (autoregressive moving average) models to detect aging and estimate resource exhaustion times. The Pinpoint system to monitor and predict network traffic is proposed by [Chen et al. (2002)]. They include a cluster-based analysis technique to correlate failures with components and determine which component is most likely to be in a faulty

state. A noteworthy approach is introduced by [Salfner et al. (2003)]. The authors build on log files as a valuable resource for system state modeling.

2.2 Responsiveness

A variable of particular interest in real-time systems is *responsiveness*. This attribute describes the combination of important non-functional system characteristics such as load, timeliness and fault tolerance. It has been defined as the conditional probability of correct execution of a task on time under a given fault and load hypothesis [Malek (1995)]. Responsiveness is difficult to obtain analytically with the exception of very simplistic systems. However, a real-time system can be described by a number of system variables, such as memory utilization, CPU usage, network and I/O activity. These variables are available or can be obtained with relative ease.

$$responsiveness = \boldsymbol{P}(\text{process will be finished before or at deadline} \mid \text{fault, load}) \quad (2.1)$$

This system information can then be used to model the underlying system dynamics and to predict responsiveness as a function of observed system variables. Indeed this is the approach we take in this paper.

3 Modeling Technique

3.1 Learning Task

The learning task in our scenario is straightforward. Given a set of labelled observations we compute the conditional probability that the system will serve incoming requests within a prespecified time frame, i.e. deadline. Based on this function approximation we a derive a classifier that predicts the target class label which is either "*will meet deadline*" or "*will not meet deadline*". As the function approximation and classifier mechanism we employ Universal Basis Functions (UBF).

3.2 Universal Basis Functions Approach (UBF)

To model continuous variables we employ a novel data-based modeling approach we call Universal Basis Functions (UBF) which was introduced in [Hoffmann (2004a)]. UBF models are a member of the class of non-linear non-parametric data-based modeling techniques. UBF operate with linear mix-

tures of bounded and unbounded activation functions such as Gaussian, sigmoid and multi quadratics. Non-linear regression techniques strongly depend on architecture, learning algorithms, initialization heuristics and regularization techniques. To address some of these challenges we have developed UBF. The kernel functions in a UBF can be adapted to evolve domain specific transfer functions. This makes them robust to noisy data and data with mixtures of bounded and unbounded decision regions. UBF produce parsimonious models which tend to generalize more efficiently than comparable approaches such as Radial Basis Functions [Hoffmann (2004b)].

3.3 Structural Review

Universal Basis Function networks (UBF) are structurally similar to Radial Basis Function (RBF) networks [Poggio et al. (1990)]. So we will revisit this topic only briefly. Suppose we obtain a set of data $g = \left\{ \left(\mathbf{x}_i, y_i \right) \in \Re^d \times \Re^1 \right\}$ with $i \in [1,N]$ by random sampling from a function f in the presence of noise. Where $\mathbf{x}$, $\boldsymbol{y}$, are observation vectors, d is the dimension of the observation vector, N is the number of observations. Usually we are interested in recovering that function from our sampled data g. Then the objective is to find a function which is close to our sampled data and is smooth according to a chosen smoothness criterion λ. Such a function f would minimize the following functional

$$\mathrm{H}[f] = \sum_{i=1}^{N} \left(f(\mathbf{x}_i) - y_i \right)^2 + \lambda \phi[f] \qquad (3.1)$$

where λ is a positive real number called the *regularization parameter*. The first term enforces closeness to our data, the second term smoothness. The regularization parameter controls the trade-off between the two. It can be any number in the interval [0,1]. For λ = 0 this results in plain interpolation, for $\lambda \to 1$ it would result in polynomial regression, where the degree of the polynomial depends on the smoothness term $\phi[f]$. It has been shown in [Girosi et al. (1993)] that the solution to f is given by

$$f(\mathbf{x}) = \sum_{i=1}^{N} c_i \, \mathrm{G}\left(\mathbf{W}\mathbf{x} - \mathbf{W}\mathbf{x}_i \right) \qquad (3.2)$$

where c is a coefficient vector, **W** is a $d \times d$ norm vector which rotates and resizes the input space to reflect possible linear combinations of original input variables and G is some nonlinear transformation function. The solution to (3.2) can be given by

$$f(\mathbf{x}) = \sum_{i=1}^{n} c_i \, \mathrm{G}\left(\left\|\mathbf{x} - \mathbf{t}_i\right\| \mathbf{w}\right) \qquad (3.3)$$

where $\|\bullet\|$ denotes a geometric distance measure, in our case the Euclidean. We can solve for **w** by matrix inversion.

3.4 Flexible and Domain Specific Kernel Functions

UBF use flexible mixture activation functions $G(\bullet)$ which are parametrised to change their shape smoothly from one functional form to another. Most commonly in RBF the activation function $G(r)$ is Gaussian as in (5.1) with $r = \|\mathbf{x} - \mathbf{t}\|$ and **t** being a vector of kernels. We will refer to this type of RBF as classical or traditional RBF.

$$G(r) = e^{-r^2/2\sigma^2} \qquad (3.4)$$

Other activation functions frequently investigated in literature include sigmoid types and multiquadratic functions. We propose a novel type of activation function which is a mixture of activation functions such as Gaussian, sigmoid and multiquadratics. This type of kernel function smoothly changes its shape from one activation function to another covering all transitional states. The mixture of two activation functions can be achieved by replacing the standard Gaussian kernel function (3.4) by a mixture function as in (3.5).

$$G(\mathbf{x};\mathbf{r},\boldsymbol{\sigma},\omega') = \omega'\Phi_1(\mathbf{x};\mathbf{r},\boldsymbol{\sigma}) + (1-\omega')\Phi_2(\mathbf{x};\mathbf{r},\boldsymbol{\sigma}) \qquad (3.5)$$

with

$$\omega' = \tanh\left(\frac{1}{2}\omega\right); \omega \in (-\infty,\infty) \qquad (3.6)$$

and $\Phi \in \Phi'_{1..3}$

$$\Phi'_1 = e^{-r^2/2\sigma^2} \qquad (3.7)$$

$$\Phi'_2 = \tanh\left(r / \sigma^2\right) \qquad (3.8)$$

$$\Phi'_3 = \sqrt{r^2 + \sigma^2} \qquad (3.9)$$

We call ω' the UBF slider because it smoothly adjusts the shape of the transfer function $G(\bullet)$. $\Phi'_{1..3}$ is a selection of frequently used transfer functions. In the case of a Gaussian / sigmoidal mixture (i.e. $\Phi_1 = \Phi'_1$ and $\Phi_2 = \Phi'_2$), we get a purely Gaussian transfer function for $\omega' >> 1$, for $\omega' << -1$ we get a purely sigmoid transfer function. Any value in between the two extreme generates a smooth superposition of both functions. In this paper we employ mixtures of sigmoid (3.8) and multiquadratic kernels (3.9).

3.5 Parameter Initialisation and Optimisation

The use of a mixture transfer function in the form of a linear mixture of activation functions as in (5.2) involves the estimation of a new parameter ω'. Conjugate gradient methods are a first choice to update the parameters of an UBF. Any parameter $\overline{\omega}$ would follow a steepest descent trajectory with respect to some error criterion. The disadvantage however, is that the gradient approach tends to get stuck in local minima resulting in inferior model quality. As an alternative we employ a global stochastic optimisation procedure, the de-randomised evolution strategy with full covariance matrix adaptation (CMA-ES). This procedure has been described in detail in [Rechenberg (1994)][Hansen et al. (2001)]. In UBF we have to optimise the joint set of parameters $\Theta = \{t, \sigma, \omega'\}$ with kernel positions ***t***, kernel widths σ and the UBF slider value ω' which controls the shape of the kernels.

The initial setting of ***t***, σ and ω' follows this procedure:

1. start with selecting the kernels using a standard k-means approach, e.g. [Bishop (1995)]
2. set the kernels width at[1]

$$\sigma = \frac{d}{\sqrt{2M}} \qquad (3.10)$$

where *M* is the number of kernels and *d* is the maximum distance between the chosen kernels. We have conducted experiments in which we let each kernel adapt its shape individually to the local con-

1 This is an initial heuristic value, which has been investigated in earlier work [Haykin (1994)].

ditions. This leads to an additional *k*d* number of parameters to be estimated compared to a standard RBF. Where *k* is the number of kernels and *d* is the number of input variables. We call this full covariance matrix adaptation. To reduce the number of parameters to be estimated and to make the models comparable in terms of number of free parameters we reduce the full covariance matrix of the UBF to a single parameter which is subsequently the same for all kernels. This value is estimated in an initialization step. In this step we iterate ω' through the interval of plausible values $\omega' \in [-1,1]$ and calculate the initial model error. We then clamp ω' at the value which generated the lowest initial model error. This leaves us essentially with a RBF with a modified kernel and the same number of parameters as a standard RBF. We call this partial covariance matrix adaptation.

4 Experiments

In this chapter we show the practicability of our learning algorithm by modeling responsiveness of a simulated queuing system and by predicting CPU demand of the Baum-Welch algorithm [Rabiner (1989)] based on its parametrization.

4.1 Experimental settings

In our first experiment we build a stochastic model of the CPU time needed to complete the Baum-Welch Algorithm as a function of its input parameters. The computational effort to complete this algorithm is non linearly dependent on its input parameters and the effects of its parameters are well understood. This way it serves as a non trivial test case. The motivation in modelling and predicting the resource requirements of a task is its potential to derive optimised scheduling and placement algorithms based on a-priori knowledge about a tasks resource demands. This knowledge can be extracted from a stochastic model of the task.

Our second application builds on the results we obtained from our prediction of a tasks CPU demand. One major challenge in real-time systems is the timing of all tasks involved. The question to be solved is: what is the probability that a given task with predicted resource demands will be finished within its deadline, given the load and fault status of the system. We investigate the effectiveness of Universal Basis Functions (UBF) to model conditional densities of execution times as a function of

the tasks required CPU time and the load status of the system. For this purpose we simulate a G/G/N queuing system to experimentally compare predicted and simulated responsiveness of the system.

4.2 Modeling and Predicting CPU demand of the Baum-Welch algorithm

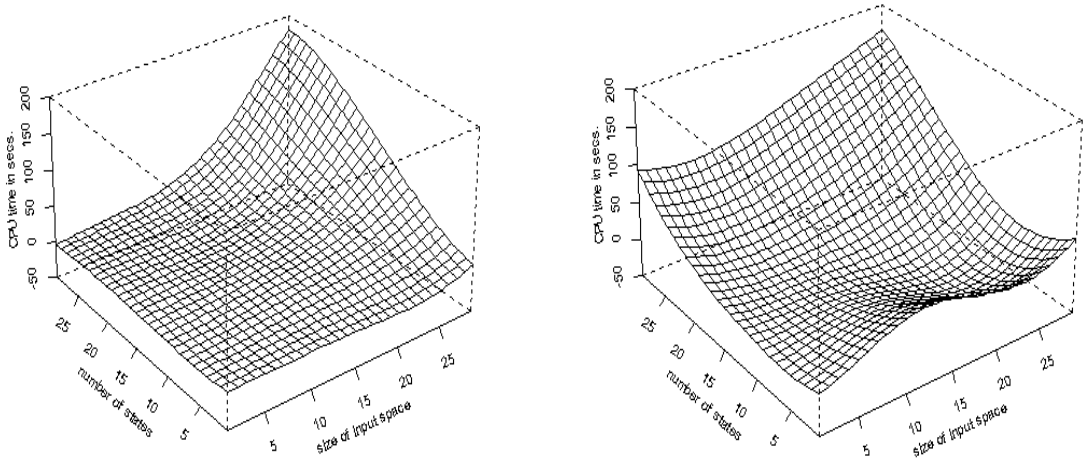
Load in a distributed computer systems is dependent on the utilization of given resources such as CPU, memory, storage etc. A first step towards solving the responsiveness equation (2.1) is therefore to find a load model. This is to give reliable estimates of the resource requirements of involved processes before execution. As an example process we choose the Baum-Welch algorithm. This algorithm is recursively defined and can be utilized for parameter optimisation in Hidden Markov Models. The important aspect is that the performance of this algorithm is difficult to analyse and non-linearly dependent on its parameters. Furthermore, the sample problem setting is low dimensional and intuitively verifiable since performance is only dependent on two parameters. The input parameters of the Baum-Welch algorithm are the number of states in a Hidden Markov Model and the size of the observation vector. However, we would like to stress at this point that the proposed method is process invariant and that we choose this particular example for illustrative purposes.

We obtained a matrix of CPU utilization for different parameter settings of our sample algorithm with the *UNIX rusage* command. Figure 1 shows a non-linear UBF model fitted to the complete set of data obtained this way. It can be seen that with increasing number of states and size of the observation vector a larger amount of CPU time is required to complete the sample process.

In real environments, complete information of parameter combinations are rather difficult to obtain. More likely the process will appear in a-priori unknown time intervals with possibly stochastic combination of parameters. In such a case the problem is ill-posed since only information of a small parameter subset is available. Figure 1 (b) shows one model that was fitted to 10% of the actually available data set. The data points were sampled randomly from the original and complete data set (Figure 1 (a)). This corresponds to observing the algorithms performance during an initial „burn in“ phase in which the algorithm is fed with random combinations of parameters. The CPU requirements measured this way could then be used to establish an input-output mapping. It is clear that the quality

of the model depends on how the data points used for building a model are selected. Further research in this area is needed to clarify the influence of data distributions from real world applications.

However, we find that even with a small subset of data we can build models which closely resemble the dynamics of the process in question. These models can be a base for intelligent and effective resource planning, scheduling and job placement in distributed real-time systems.



(a) (b)

Figure 1 (a) CPU time in seconds vs. the Baum-Welch algorithm's parameters. With increasing number of states and size of the observation vector a larger amount of CPU time is required to finish the process. The graph shows the complete enumeration of the sample solution space. (b) Ill-posed problem. Model built with sparse data. Approximately 10% randomly sampled data from the entire parameter space.

4.3 Responsiveness of a real-time queuing system

Most principles documented in literature try to speed up task computation and meet deadlines by some sort of intelligent scheduling. However, this still leaves open the question, how likely it is that a given task in a possibly dynamic environment meets timing constraints. Despite the developments in scheduling heuristics and other procedures to optimise computation times for a given task there are still major shortcomings. Typically, tasks are assumed to have deterministic execution times which are their worst case performance. However, these are very narrow assumptions which might over constrain matters. Additionally, we frequently do not even know the worst case performance of a task because this is dependent on system variables we can not directly observe. This leaves us with the

highly unsatisfactory status that for most real-world real-time applications we can hardly give performance profiles or give quantitative measures about their probable performance under given constraints. Furthermore, scheduling algorithms are not particularly useful if the problem is further complicated by a-periodic, sporadic and/or bursty activity. The scheduling approach is thus impractical for an important class of real-time systems involving multi-media applications or communication systems which can show stochastic activity.

One way of describing these systems is queuing theory. However, the problem with queuing theory is that only very simplistic systems can analytically be solved [Lehoczky (1996)]. Real-world real-time systems tend to be far more complex that what could be handled with queuing theory [Nelson (1995)].

4.3.1 Observation based modeling

In this part of the paper we investigate a Universal Basis Function approach to model a queuing system based solely on observations about its dynamics. The approach utilizes Universal Basis Function for non-linear, non-parametric modelling. Queuing systems can easily show multi-modal behaviour which makes the application of non-parametric UBF modelling even more appealing. We model task resource requirements based on the task's parametrisation. These models are then used to make forecasts of the system behaviour with out-of-sample parameter settings. For example we obtain complete probability density functions which are then utilized to calculate the probability that a certain task will be completed within a given time frame.

The following scenario is a simulation in which customers (tasks) arrive at a single node computing system with given resource requirements. For example, a task arrives with the request for a certain amount of CPU time. At the same time other tasks are being processed in the system already. In such a scenario it is interesting to answer the question *What is the probability of a given task being processed within a certain time frame?* This is also referred to as responsiveness as defined in (1.1). In order to get an answer to this question we let the probability density be a function of given or obtainable system parameters such as the number of tasks already in the system N, requested CPU time of the task in question t_c, scheduling policy and the distribution of computation times of other tasks in the system. We use a scheduling policy to minimize response time. That means the task which shows

the largest value of *waiting time + requested CPU time* will be scheduled first. The variable *waiting time* is the time which has passed since the task first entered the system, *requested CPU time* is a given value which is task specific. Other scheduling policies might be handled as well with this methodology.

4.3.2 Implementation

In order to investigate the effectiveness of our modelling technique we compare simulation results with the results given by our procedure. We proceed as follows. We simulate system behaviour for a given parameter range, in our example $N \in [1...100]$ and $t_c \in [40...60]$. We consider a scheduling policy which minimizes response time and a Gaussian distribution of task execution times. Complete enumeration of the parameter space leaves us with 2.000 states to consider. From these simulation results we take 2% or 40 data samples to build a conditional density model of the system. The samples we take evenly distributed from the parameter space. We then compare the models predictions with the outcome of the simulation for the out-of-sample parameter space. From the models outcome we can directly derive probabilities for execution times of a particularly parametrised task.

The probability density function $P(x)$ specifies that the probability of variable x lying in the interval between a and b is given by

$$P(x \in [a,b]) = \int_a^b p(x)dx \qquad (4.1)$$

The function $p(x)$ is normalized such that

$$\int_a^b p(x)\,dx = 1 \qquad (4.2)$$

In our case $a = 40$ and $b = 130$ meaning that we restrict the parameter space to tasks with execution times between 40 and 130 time units. This restriction is merely for illustrative purposes. The results are shown in Figure 2 to 5. Figure 2 shows probability density functions which give the probabilities of a task being executed within x time units. The dotted graph gives the models estimation (based on an evenly sampled 2% of all available data), the solid line graph gives the simulation results. Parameters for this example where $N = 11$ and $t_c = 50$. Figure 2 shows the integrated probability density func-

tions *F*. This graph gives estimates on the responsiveness of the system. Figure 4 and 5 show a more complex scenario with bimodal density shapes.

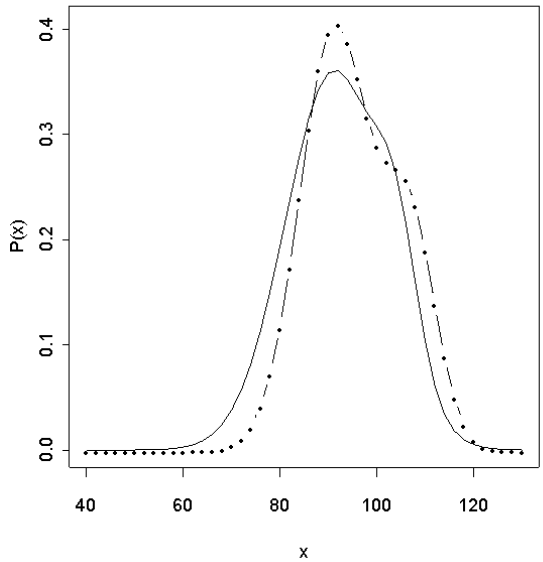


Figure 2: Probability density functions which give the probabilities of a task being executed within *x* time units. The dotted graph gives the models estimation based on an evenly selected sample of 2% of all available data. The solid line graph gives the simulation results. Parameters for this example where *N* =11 and t_c = 50.

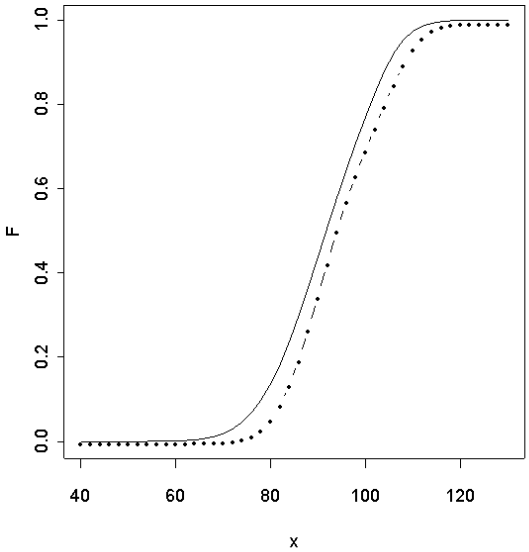


Figure 3: Accumulated probability density functions. This graph gives estimates on the responsiveness of the system and gives the probability of a task being executed in x time units. The dotted graph gives the models estimation, the solid line graph gives the simulation results. Parameters for this example where N =11 and t_c = 50.

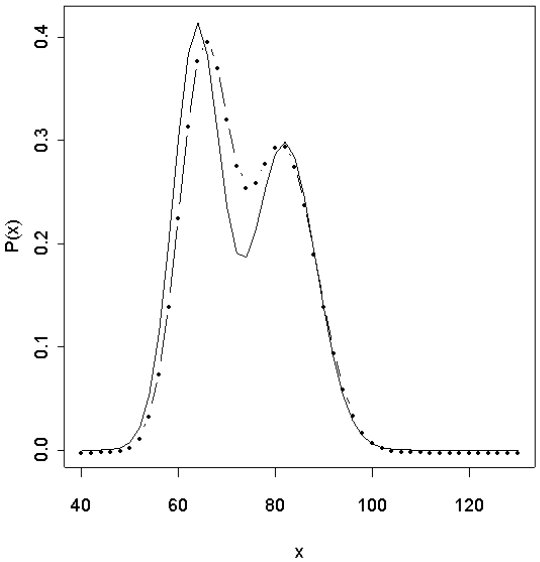


Figure 4: Parameters for this multi modal example where N =3 and t_c = 40.

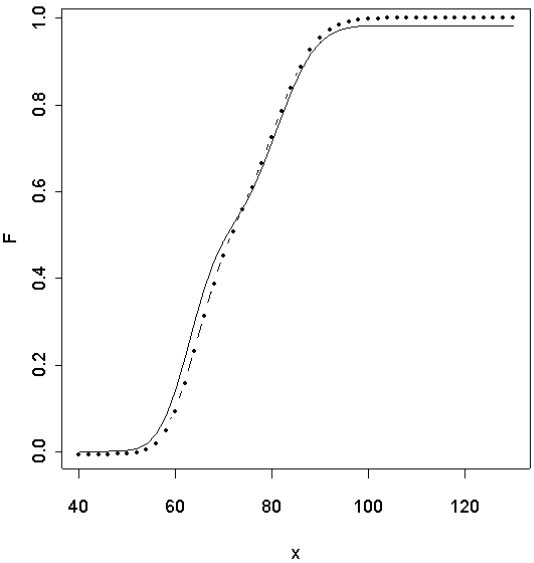


Figure 5: Accumulated probability density function for the same parameters as in Figure 4

5 Modeling a Real-World Telecommunication System

We apply the UBF modeling technique to data of a commercial telecommunication platform. The primary objective is to model and predict the probability that the availability of the system drops below 99.99% in successive five minute intervals. This corresponds to 0.01% of calls missing their prespecified deadline in any given interval.

5.1 Characteristics

The main characteristics of the software platform investigated here is its component-based software architecture running on top of a fully featured clustering environment consisting of two to eight nodes. We measure time continuous data of a *two* node cluster non-intrusively using the Unix SAR (system activity reporter) tool. To label the data sets we use an external stress generator to keep track of both the call load and responsiveness during system tests.

5.2 Data

We have monitored 53 days of operations over a four month period providing us with approximately 30GB of data. Results presented in the next section originate from a three days excerpt. We split the three days of data into equally proportioned segments (one day per segment). One data segment we use to build the models, the second segment we use to cross validate the models, the third segment is our test data which we kept aside. We gathered the numeric values of 42 operating system variables once per minute and per node. This leaves us with 84 variables in a time series describing the evolution of the internal states of the operating system, thus in a 24-hour period we collect n = 120.960 readings. Figure 7 depicts plots of two variables over a time period of six hours.

5.3 Modeling Task

Our real world modeling task is straight forward. Given a set of observed variables describing the status of the system at any given point in time we estimate the probability at the prediction time t_1 that within the *prediction period* Δt_p all calls will meet their deadlines. The prediction period occurs some time after the prediction is made, we call this the *lead time* Δt_l. This *lead time* is necessary for

a prediction to be of any use. The prediction period defines how far the prediction extents into the future. The value of the *lead time* Δt_l critically depends on the problem domain, e.g., how long does it take to restart a component, to initiate a fail over sequence or to initiate any other preventive measure. The value of the prediction period can be adapted more flexibly. The larger this value becomes the easier the prediction problem, but the less meaningful the prediction will be. The *embedding dimension* Δt_e specifies how far the observations extend into the past.

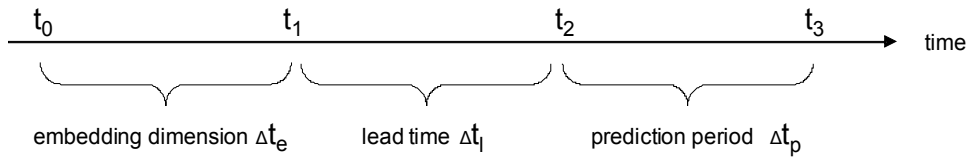
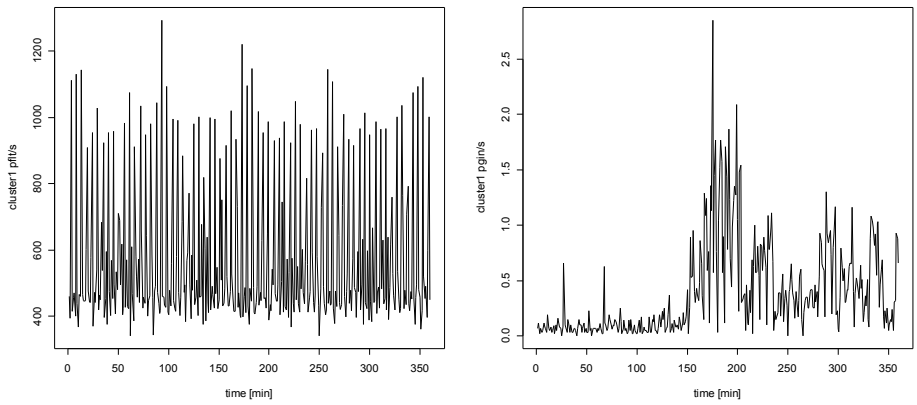


Figure 6: The embedding dimension specifies how far the labelled observations extend into the past. The lead time specifies how long in advance a failure is signalled. A prediction is correct if the target event occurs at least once within the prediction period.

For practicability reasons in our example we choose the prediction period to be five minutes, the lead time we set also at five minutes and we used an embedding dimension of ten minutes.



(a) (b)

Figure 7: Showing two system variables over a sample period of six hours. (a) shows "page faults per second" on cluster node one, (b) shows "pages In" per second on the same node.

5.4 Metrics: Precision, Recall and F-Measure

The quality of our responsiveness model can not be measured directly. In reality a deadline is either met or not met. This concept is closely related to a classifier system which either predicts "*will meet*

deadline" or "*will not meet deadline*". Thus we threshold our model to turn it into a classifier. The optimal threshold we find by sampling from our validation data. The resulting classifier we use to categorize conditional densities. We assess the performance of the classifier with respect to the number of correctly predicted events (*true positives*), missed events (*false negatives*) and also to the number of incorrectly predicted events (*false positives*). An event corresponds with *responsiveness = 0*.

In our sample data we count 96 events where deadlines are not met within a total of 1080 observations. The majority class thus contains 984 observations indicating a responsiveness equal to one. The minority class contains 96 entries indicating a responsiveness equal to zero. By always considering the majority class we would achieve a predictive accuracy of roughly 89%. Thus the predictive power of the widely used quality metric *predictive accuracy* is of limited use in a scenario where we would like to model and forecast rare events as in our case – they yield an overly optimistic model quality. Therefore, we need a metric optimised for rare event classification rather than predictive accuracy. We report *precision*, *recall* and the integrating *F-Measure* [Rijsbergen 1979].

Precision and recall, originally defined to evaluate information retrieval strategies, are frequently used to express the classification quality of a given model. Applications can be found for example in [Weiss 1999] [Trivedi et al. 2000]. Precision is the ratio between the number of correctly identified failures and predicted failures:

$$precision = \frac{\text{true positives}}{\text{true positives} + \text{false positives}} \qquad (5.1)$$

Recall is defined as the ratio between correctly identified faults and actual faults. Recall sometimes is also called sensitivity or positive accuracy.

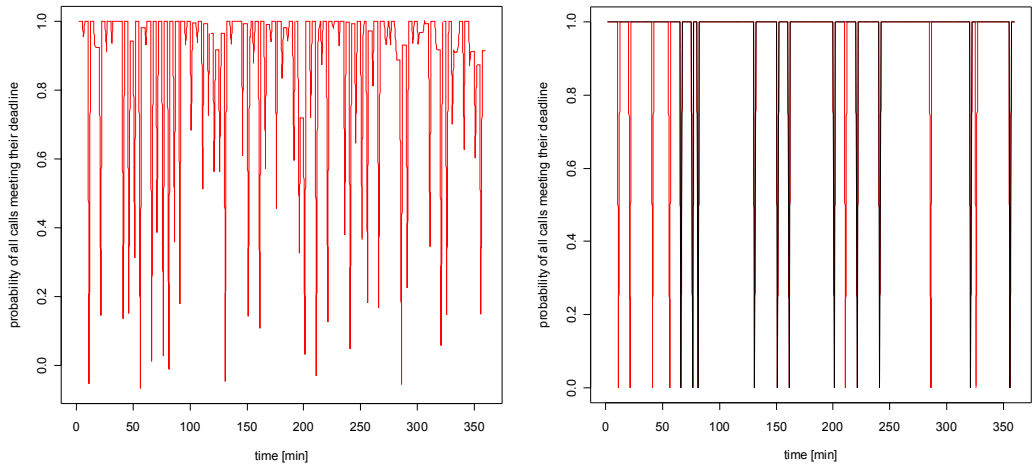
$$recall = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \qquad (5.2)$$

Following [Weiss 1999] we use *reduced precision.* There is often tension between a high recall and a high precision rate. Improving the recall rate, i.e. reducing the number of false negative alarms, typically also decreases the number of true positives, i.e. reducing precision. A widely used metric which integrates the trade-off between precision and recall is the F-Measure.

$$\text{F - Measure} = \frac{2 \cdot precision \cdot recall}{precision + recall} \qquad (5.3)$$

6 Results and Discussion

We investigated the effectiveness of Universal Basis Functions (UBF) to model two important aspects of distributed real-time systems which are a) a-priori knowledge of resource demand of tasks and b) responsiveness. We give a proof-of-concept of our modeling technique by applying it to real as well as synthetic data. In the first example we model and forecast CPU demand of the Baum-Welch algorithm. In the second application we derive precise quantification of responsiveness of a simulated G/G/N queuing system.



(a) (b)

Figure 8 showing *responsiveness* or conditional probabilities at each point in time in (a) and a classifier system (b) build on top of the model used in (a) with a threshold of 0.2. Dotted lines in (b) represent the classifiers forecast probability that calls will meet their deadlines. Solid lines show actual probabilities. The model in (b) hits all *real* missed deadlines and also predicts some additionally missed deadlines.

We show that models based on few observations, in the order of 2% of the entire variable space, obtained during an initial „burn in“ phase, give near optimal estimates of out-of-sample resource demands. These estimates can be the basis for probabilistic real-time scheduling algorithms.

In our second example we modelled the probabilistic behaviour of a single node queuing system. Based on few system parameters such as number of tasks in the system and resource require-

ments of the task in question we derive conditional probability densities which show the probability that a given task will be completed by the system within a given time frame (i.e., *responsiveness*).

To show the effectiveness of our modeling approach in an industrial distributed real-time environment we model the timely behavior of a commercial telecommunication platform. Deadlines in this platform are introduced as calls which have to be handled within a prespecified time frame. We derive probabilistic models which give the probability that 0.1% of calls within a five minute interval will meet their deadline. We achieve a recall rate of 82% and a precision rate of 49% which compares most favorably to the standard MTBF based model (typically used in this environment) which yields a recall rate of 20% and a precision rate of 25% (see Table 1 for details).

Model	***Type of Data***	***Precision***	***Recall***	***F-Measure***
UBF	SAR	0.4912	0.8295	0.6088
naive	MTBF	0.2500	0.2000	0.2222

Table 1: Precision, recall and F-Measure for the universal basis function approach (UBF) and naive model. In the case of UBF we report mean values, because the UBF additionally yields model error estimates. See Figure 9 for a complete distribution of values. The naive model was build using mean-time-between-failures (MTBF) as timing estimates. The reported results were generated on previously unseen test data.

Furthermore, we derive responsiveness values for the telecommunication system giving precise estimates of the probability that calls within successive five minutes intervals will meet their deadline or not (see Figure 8 for details). Additionally we derive error boundaries for our models as depicted in Figure 9. The granularity of five minutes time intervals is introduced by the telecommunication system and reflects data handling capacities of the platform and platform configuration.

With the proposed modeling method we cannot only give statements whether or not a system is working, we can also give a complete probabilistic description on how well the system is working. By utilizing Universal Basis Functions for building non-linear, data-driven models and for estimating conditional densities we provide a tool for intelligent resource management. This has two implications. First, based on this tool forecasts of resource demands can be made before a task is executed, thus providing a platform for improved task placement. Second, observation-based models of system behavior can be build in cases where simulations would be computationally too expensive or simply impractical because the specifications of system components are not readily available to the user.

Based on our method probabilistic models of the system in question can be built by observing the systems dynamics.

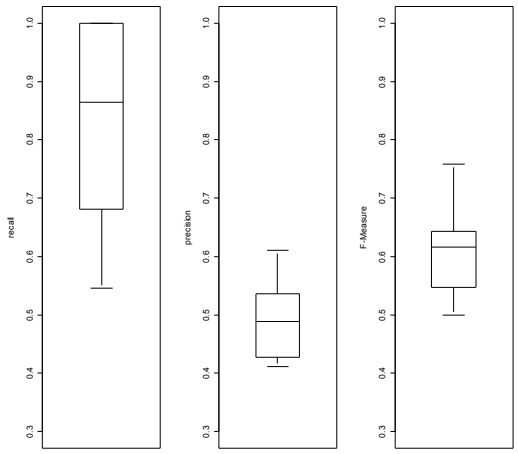


Figure 9: Box-Whisker plots of recall, precision and F-Measure for the UBF model applied to previously *unseen test data*. Shown are the mean, minimum, maximum and quantiles of the respective measure The F-Measure is a cumulative measure integrating precision and recall into a single value.

We have shown that the UBF approach a) outperforms standard modeling techniques significantly, b) derives accurate forecasts of tasks resource requirements and c) gives precise estimates of a systems responsiveness, that is the probability of meeting deadlines in the presence of faults, for synthetic as well as real data. Future research topics will include the validation of our models as a function of system configuration and time and will also focus on integrating additional data sources such as log files.

References

[Bishop (1995)] Bishop C. M. , Neural Networks for Pattern Recognition, Clarendon Press, London, 1995

[Burns et al. (2000)] Burns A., Edgar S. (2000), Predicting Computation Time for Advanced Processor Architectures, Real-time systems research group University of York, 2000

[Burns et al. (2003)] Burns A. And G. Bernat and I. Broster, A Probabilistic Framework for Schedulability Analysis, Proceedings of the Third International Embedded Software Conference, EMSOFT, Lecture Notes in Computer Science, pg. 1-15, 2003

[Chen et al. (2002)] Chen Mike , Emre Kiciman, Eugene Fratkin, Armando Fox and Eric Brewer (2002), Pinpoint: Problem Determination in Large, Dynamic Systems, Proceedings of 2002 International Performance and Dependability Symposium, Washington, DC, June 23-26, 2002, 2002

[Gardner (1999)] Gardner Mark (1999), Probabilistic Analysis and Scheduling of Critical Soft Real-Time Systems, PhD Thesis at University of Illinois at Urbana-Champaign, 1999

[Garg et al. 1998] Garg S., Puliofito A., Telek M., Trivedi K., Analysis of preventive Maintenance in Transaction based Software Systems, , 1998

[Girosi et al. (1993)] Girosi Federico, Jones Michael, Poggio Tomaso (1993), Regularization Theory and Neural Networks, MIT Cambridge, AI Memo 1430, 1993

[Hansen et al. (2001)] Hansen N., Ostermeier A. (2001), Completely derandomized self-adaptation in evolution strategies , Evolutionary Computation 9(2):159-195, 2001

[Haykin (1994)] Haykin S. (1994), Neural Networks - A Comprehensive Foundation, MacMillan, 1994

[Hertz et al (1991)] Hertz J., Krogh P., Palmer R. (1991), Introduction to the Theory of Neural Computation, A Lecture Notes Volume in the Santa Fe Institute Studies of Complexity, Vol. I, 1991

[Hoffmann (1996)] Hoffmann Günther Andreas (1996), A Radial Basis Function Approach to Modeling Resource Demands in Distributed Computing Systems, ISMIP - International Symposium on Multi Technology Information Processing; September 1996; Taiwan, 1996

[Hoffmann (2004a)] Hoffmann Günther Andreas (2004), Adaptive Transfer Functions in Radial Basis Function Networks, in Lecture Notes in Computer Science LNCS by Springer; (ICCS2004), June 2004, Krakau, Eds.: Bubak M., Dick van Albada, Peter M.A. Sloot, Jack J. Dongarra, 2004

[Hoffmann (2004b)] Hoffmann Günther Andreas (2004), Data specific mixture kernels in Basis Function networks, ICONIP 2004 submitted for publication, 2004

[Lehoczky (1996)] Lehoczky J. P., Real-Time Queueing Theory, Proceedings of Real Time Systems Symposium, 1996

[Li et al. 2002] Li L., Vaisyanathan, Trivedi , An Approach for Estimation of Software Aging in a Web Server, , 2002

[Malek (1995)] Malek, M. (1995), Omniscience, Consensus, Autonomy: Three Tempting Roads to Responsiveness, Keynote Address, 14th IEEE Symposium on Reliable Distributed Systems SRDS, 1995

[Nelson (1995)] Nelson R. (1995), Probability, Stochastic Processes and Queueing Theory - The Mathematics of Computer Performance Modeling, Springer Verlag, 1995

[Poggio et al. (1990)] Poggio T., Girosi F., A Theory of Networks for Approximation and Learning, Proc. of IEEE 78(9), 1990

[Rabiner (1989)] Rabiner L. R. (1989), A Tutorial on Hidden Markov Models, Proc. of IEEE, Vol.77, No.2, 1989

[Rechenberg (1994)] Rechenberg I. (1994) , Evolutionsstrategie '94, Frommann-Holzboog, Stuttgart, 1994

[Rijsbergen 1979] Rijsbergen Van C.J., Information Retrieval, Butterworth, 1979

[Salfner et al. (2003)] Salfner F., Tschirpke S., Malek M. (2003), Comprehensive Logfiles for Autonomic Systems, IEEE Proceedings of IPDPS 2004 (International Parallel and Distributed Processing Symposium), 2003

[Terrasa et al. (2003)] Terrasa Andres, Bernat Guillem (2003), Extracting Temporal Properties from Real-Time Systems by Automatic Tracing Analysis, Technical University Valencia, Real-time systems research group University of York, 2003

[Trivedi et al. 2000] Trivedi Kishor S., Vaidyanathan Kalyanaraman and Go¡seva-Popstojanova Katerina, Modeling and Analysis of Software Aging and Rejuvenation, , 2000

[Vaidyanathan et al. 1999] Vaidyanathan K. and K. S. Trivedi, A Measurement-Based Model for Estimation of Resource Exhaustion in Operational Software Systems, , 1999

[Weigend et al. (1994)] Weigend A. S., Gershenfeld N. A., eds. (1994), Time Series Prediction, Proceedings of the Santa Fe Institute, Vol. XV, 1994

[Weiss 1999] Weiss G. M., Timeweaver: a Genetic Algorithm for Identifying Predictive Patterns in Sequences of Events, , 1999