Efficient Statistical Performance Modeling for Autonomic, Service-Oriented Systems

Rui Zhang[1], Alan Bivens[2] and Iead Rezek[3]

[1]Oxford University
Computing Laboratory
Oxford OX1 3QD, England
ruiz@comlab.ox.ac.uk

[2]IBM Corporation
T.J. Watson Research Center
Hawthorne, NY 10532, USA
jbivens@us.ibm.com

[3]Oxford University
Department of Engineering Science
Oxford OX1 3PJ, England
irezek@robots.ox.ac.uk

Abstract

As service-oriented environments grow in size and complexity, managing their performance becomes increasingly difficult. To assist administrators, autonomic techniques have been adopted to permit these environments to be self-managing (problem localization, workload management, etc.). These techniques need a sense of system state and the ability to project a new state given some change within the environment. Recent work addressing this issue frequently used statistically learned models which were derived entirely from data. However, many environments already have management facilities in place that could provide precise and useful insights (e.g. workflows) into the system. This paper introduces a method of modeling service-oriented system performance using Bayesian networks and specifically addresses the benefits obtained by incorporating these insights into the model learning process. To further minimize model building costs, we devise a decentralized method to concurrently learn parts of the model where knowledge inclusion is impossible. Simulations and applications in actual environments show significant reductions in learning time, better accuracy and stronger tolerance to small learning data sets.

1 Introduction

Service-oriented computing [15] facilitates distributed application development across organizational and geographical boundaries through the promotion of self-describing and open software components. In these environments, user requests traverse multiple heterogeneous components drawn together on-the-fly, making it ever more challenging to manage end-to-end quality of service (QoS) (e.g. response time) and meet goals in service level agreements. In order to do so, autonomous management [10] software in service-oriented environments requires a model to capture the complex system dynamics and link individual component behaviors to end-to-end performance (typically response time) states. This model must address two challenges: 1) it must provide *accurate* guidance to autonomic activities such as resource provisioning, load balancing, and performance problem localization and remediation, and 2) due to the dynamic nature of autonomic environments, the model must be reconstructed from time to time, removing past and obsolete information to reflect most recent system states. This process must be accomplished at *little* cost with *minimal* human intervention.

Much work has been done in analytical modeling of computer system performance. Classic theories such as queuing networks, petri nets and control theory have been applied to real world systems [20, 7, 11]. Although these models are stable and mathematically sound, they may be very difficult to derive manually from certain domains. The modeling process can require substantial effort from human experts and may be subject to inadvertent human mistakes or unrealistic assumptions.

More recently, some groups have created techniques that "learn" models from performance data collected through instrumentation/profiling [9, 1]. These statistical learning approaches do not assume human involvement and require little domain analysis. While promising, such practices are still at an early stage, and can be computationally very expensive and rather data-sensitive.

Motivated by the contrasting natures of analytical modeling and statistical learning approaches, this paper attempts to harvest the strengths of both by encoding existing domain knowledge into the statistical learning framework. In this manner, not only can the vulnerabilities of analytical

1-4244-0910-1/07/$20.00 ©2007 IEEE.

modeling be compensated by information extracted from the data, but the cost and data-sensitivity of statistical learning can be reduced by informative domain knowledge guidance. In particular, a knowledge-enhanced Bayesian net is proposed to model the end-to-end response time of service-oriented systems. The model leverages readily available domain knowledge and decentralized learning to minimize model construction cost while optimizing accuracy.

In addition, the locality of data required to learn individual components of the BN has inspired us to seek a decentralized and concurrent way to derive these model parts locally on machines where the data required is collected.

This work has lead to the following research contributions:

- We have used easily attainable domain knowledge to establish an accurate Bayesian network response time model while eliminating the need for structure learning and easing the exercise of parameter learning.
- When no adequate domain knowledge can be utilized, we have formulated a novel way to concurrently learn model parameters from local data at distributed locations, thus further diminishing model construction costs.
- An implementation of the approach has been delivered to operate under a flexible model (re)construction schemes and can be integrated into autonomic solutions with minimal effort.
- Comprehensive simulations have been presented demonstrating the vastly reduced model construction time and data-sensitivity. Furthermore, the model has been further justified by its use in two real-world applications on a service-oriented Grid.

The remainder of the paper is structured as follows. The sequel provides necessary background. Section 3 presents the knowledge-enhanced Bayesian network for end-to-end response time modeling. The approach is experimentally evaluated using simulations in Section 4. Two real-world applications of the model are detailed in Section 5. Section 6 reviews related work. Section 7 concludes and discusses future research.

2 Performance Monitoring and Modeling for Service-Oriented Systems

The eDiaMoND Grid [3], an OGSA-enabled federated database of annotated mammograms, will be used as a reference to service-oriented systems throughout this paper. Figure 1 shows a common eDiaMoND scenario. In it a radiologist retrieves some mammograms for analysis. Six Grid services are involved. Having received a request from the radiologist client, the `image_list` service calls the `work_list` service, asking for information about the images assigned to the radiologist. Suppose that IDs and locations are returned of two images that need to be compared. Since the two images are stored in a local hospital `L` and a remote hospital `R`, respectively, the `image_list` service simultaneously issues two requests to the `image_locator` service on both sites. This leads to the invocation of local and remote `ogsa_dai` service (a service-oriented database wrapper) on both sites, to obtain the corresponding images from on-site databases. The retrieved images are returned as part of the service responses to the radiologist for viewing and comparing.

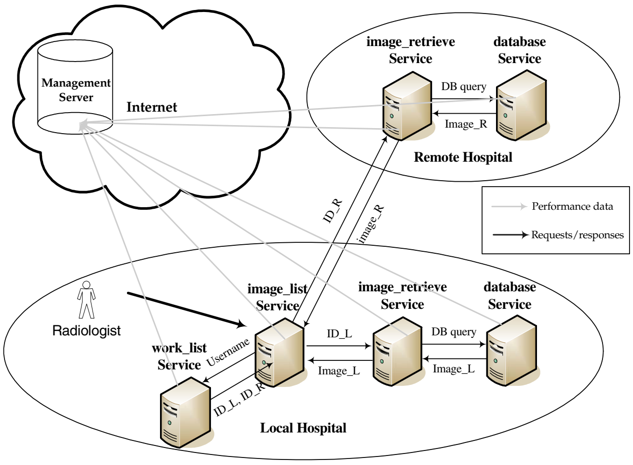


Figure 1. Performance data reporting in an eDiaMoND scenario. Monitoring infrastructure components are not shown.

As is described in previous work [21], standard OGSA-based middleware, are instrumented with monitoring points, which measure the time elapsed at middleware components. A monitoring agent resides on each machine, listening to the monitoring points for data, possibly batching them before reporting them to the management server.

The collected data are used to periodically construct end-to-end response time models on-the-fly, so as to reflect latest dynamics in the service-oriented environment and assist autonomic management. Although a strategy where models are updated with the latest data may appear less extreme, the disperse of old data is often not possible under current statistical frameworks such as Bayesian network [19, 8]. The result is that these out-of-date information lingers in the updated model and adversely impacts its accuracy, making a scheme purely based on updates inadequate. A hybrid scheme combining frequent updates in-between reconstructions may be more appropriate, but is beyond the scope of this paper, which focuses on the more complex and costly model (re)construction procedures.

In general, the model (re)construction should occur at interval, T_{CON}, using a sliding data window W, allowing the model to use the data of the current interval as well as $(K-1)$ previous intervals (see Equation 1).

$$W = K * T_{CON} \tag{1}$$

In Equation 1, K is an *Environmental Correlation Metric* based on the autonomic nature of the environment. It characterizes how often autonomic actions may happen in the environment, rendering the current environment states unrelated to their pasts. Environments where radical changes (e.g. resource allocation, failure recovery actions etc.) happen more often will have a lower K value making the model construction only use more recent data. If things change less dramatically, a larger K value will allow the model to consider a larger window of data .

T_{CON} is be partly based on the interval of data collection as shown in Equation 2.

$$T_{CON} = \alpha_{model} * T_{DATA} \tag{2}$$

where α_{model} is the *Model Construction Coefficient* and T_{DATA} is the *Data Collection Interval.* Models that can be built quickly will have a lower α_{model} value indicating that they could be applied in environments demanding more frequent and accurate model recycling. T_{DATA} is how often a data point is reported and dependent on the monitoring infrastructure. $K * \alpha_{model}$ gives the number of data points available for inferring the model.

3 A Knowledge-Enhanced Bayesian Network Model for Response Time

This section presents a Bayesian network model to capture response time in service-oriented systems. It is shown in the first three subsections how the model can be constructed using a combination of readily available domain knowledge and performance data collected via instrumentation (i.e. putting monitoring points into the system as is described in the previous section). Where appropriate, the model learning is decentralized as is described in Subsection 3.4.

3.1 Mathematical Framework

A Bayesian network is used to model the relationship between time elapsed on services and end-to-end response time, capitalizing on its support of automatic model inference from data. This formalism is also chosen because it reflects the stochastic nature of the problem while graphically representing the causality among elapsed time on services and end-to-end response time.

DEFINITION 1. A *Knowledge Enhanced Response Time Bayesian Network (KERT-BN)*, is a directed graph (DAG) representing the joint distribution $P(D, X_1, \cdots, X_n)$ of random variables X_i, $i = 1 \cdots, n$, the elapsed time of service i, and random variable D, the end-to-end response time, such that

$$P(D, X_1 \cdots X_n) = P_{\mathcal{D}}(D|\Phi(D)) \prod_{i=1}^{n} P_{\mathcal{X}_i}(X_i|\Phi(X_i)). \tag{3}$$

Here, $\Phi(X_i)$ and $\Phi(D)$ are the set of parents of node $X_i, i = 1, \ldots, n$ and D, respectively, and will be determined by workflow and resource sharing knowledge in the next subsection. $P_{\mathcal{X}_i}(X_i|\Phi(X_i)), i = 1, 2, \ldots, n$ and $P_{\mathcal{D}}(D|\Phi(D))$ are the *Conditional Probability Distributions (CPDs)* of X_i and D, respectively, and describe the dependencies among service elapsed times and their effect on the end-to-end response time (to be discussed further in Section 3.3).

A KERT-BN can be either continuous or discrete, depending on application needs. Since continuous CPDs (typically Gaussian CPDs) have relatively few parameters, a continuous KERT-BN allows the model to converge more quickly when data is lacking. This property is attractive in fast changing environments where model reconstruction must be performed very frequently before a lot of data can be gathered. If the environments are relatively stable, a discrete KERT-BN may be considered. Contrary to a continuous model, a discrete one imposes no assumption on the shape of the CPDs and is likely to achieve greater accuracy when there is sufficient data.

3.2 Establishing the Structure with Workflow and Resource Sharing

A Bayesian network can be built in two steps. First determine the structure (i.e. the DAG which captures the statistical/causal dependencies between the random variables) and then obtain the parameters (CPDs). Whilst automated methods for Bayesian net structure learning exist (i.e. methods for finding a DAG topology that best fits the data [14]), they are typically computationally expensive. For instance, a KERT-BN for a system of n services will have $n+1$ nodes and exponential (in $n+1$) number of DAG structures. Such complexity makes it intractable to exhaustively search for the best DAG [5] in large environments. Even greedy algorithms like K2 [6] needs to explore $O((n+1)^2)$ possibilities.

If only one autonomic management product is using the response time model, then K can base this metric off its own interval of autonomic actions. If multiple autonomic managers are present in the environment, K should be a statistical combination of autonomic change intervals of the different products (e.g. taking the minimum of the autonomic change intervals may be appropriate)

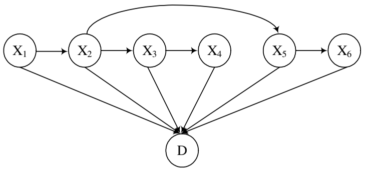
Intuitively, dependency is present between two random variables if there is some connection between them such that change in one variable can somehow be communicated to the other variable where a corresponding variation may be triggered. In the context of service-oriented systems, it appears reasonable to assume that such connections between the performance (elapsed time in this paper) of two services i and j can exist in two forms:

- While a number of relationships may be interpreted from workflow information, this paper attempts to reduce the number of dependencies considered by only accounting for direct and important relationships between services and their immediate upstream services. If one service (say, i) is the immediate upstream service of the other service (j) in the workflow graph, classic performance modeling literature [16] suggests that i's elapsed time can be tied to its throughput. Thus elapsed time of j is related to i through j's input from i (assuming that j's input can determine its elapsed time). A burst in i's workload or an increase in its service rate may prolong or shorten its elapsed time, and may also be reflected by change in j's elapsed time. The latter phenomenon corresponds to an important performance management issue known as "bottleneck shift" [2], which can now be captured by the KERT-BN model without gaining insights into the services as would be needed in [2].
- The two services are sharing a common resource (e.g. CPU, memory, network). Status of the common resource can be tied to the performance of both services, resulting in dependency between them.

Both the workflow and resource sharing information can be captured by existing monitoring infrastructures [21] or are well documented at system design stage. This knowledge can then used to determine the dependencies among KERT-BN elapsed time nodes $X_1, \ldots, X_n$ at little cost. In the case of service i directly affecting service j downstream, the KERT-BN must contain an edge between the random variables X_i and X_j. Note, that whilst service i and j may very well affect service k, say, further downstream to j, our focus here is on the simplest DAG representing the workflow, i.e. the Bayesian net structure should have as few loops as possible. Similarly, resource sharing may be represented by services forming the parents to a KERT-BN node embodying the resource they share.

Given that the elapsed time on each service is collectively forming the end-to-end response time, it is natural to assume that response time node D is dependent upon all elapsed time nodes $X_1, \ldots, X_n$. Thus, a KERT-BN should capture this dependency by a conditional dependence of D on the entire set $\mathbb{X}$, i.e. $P_{\mathcal{D}}(D|\Phi(D)) \equiv P_{\mathcal{D}}(D|\mathbb{X})$.

The full DAG structure of the KERT-BN for the eDiaMoND scenario in Figure 1 is depicted in Figure 2, where the `image_locator` services (and then the `ogsa_dai` services) on the two sites are invoked in parallel after the `image_list` and `work_list` services are called.



Variable	Service
X_1	image_list
X_2	work_list
X_3	image_locator_local
X_4	ogsa_dai_local
X_5	image_locator_remote
X_6	ogsa_dai_remote

Figure 2. KERT-BN for the eDiaMoND scenario.

3.3 Determining Parameters with Workflow

Having identified an appropriate DAG for the KERT-BN (i.e. the parent sets for each node), the CPDs in Equation 3 can now be determined, typically from collected data [14]. However, parameter learning can be quite expensive when nodes have many (discrete) parents (e.g. D in Figure 2) or (discrete) parents have high cardinality. In particular, the complexity of learning the CPD for a discrete node with n discrete parents is $O(n^m)$, where m is the largest number of states for nodes involved.

Special models (e.g. Naive Bayesian network and TAN) have been proposed to reduce the complexity of parameter learning, by focusing only on important parent-children dependencies that mainly affect model accuracy. One goal of this work is to minimize the cost of parameter learning without this accuracy compromise by making use of domain knowledge. Specific to a KERT-BN, the need for learning the heavyweight CPD, $P_{\mathcal{D}}(D|X_1, \ldots, X_n)$, can be largely eliminated by exploiting precise workflow information in the following manner:

$$\begin{aligned} P_{\mathcal{D}}(D = f(\mathbb{X})|\mathbb{X}) &= 1 - l \\ P_{\mathcal{D}}(D \neq f(\mathbb{X})|\mathbb{X}) &= l \end{aligned} \tag{4}$$

where f is a deterministic function (see the example in the next paragraph) that links elapsed times to response time; l is the probability of a "leak" situation [18] where response time D is not deterministically given by the workflow-derived function $f(\mathbb{X})$ due to noises (e.g. errors may be introduced into the prediction of D using $f(\mathbb{X})$, because the placement of monitoring points through code instrumentation can be restricted and subsequent measurements taken on $\mathbb{X}$ are not precise.

The deterministic function f can be easily derived from any workflow formed by any combination of four key workflow constructs: sequence, parallel, choice and loop, using a method developed by Cardoso et. al. [4]. For Figure 2, the deterministic function giving the CPD is: $D = X_1 + X_2 + max(X_3 + X_5, X_4 + X_6)$. The maximum operation is the result of parallel invocation of the `image_locator` services (and then the `ogsa_dai` services) on the two sites, whereas the sum operations are products of sequential service invocations.

The CPD format given by Equation 4 and the workflow-defined nature of deterministic function f also apply to other transaction-oriented performance metrics such as timeout request count (i.e. a counter of how many requests timeout), only with a different mapping from the workflow to f. In the case of timeout request count, D will stand for the count for end-to-end transactions, $\mathbb{X}$ will hold per-service sub transaction counts, and it appears that f should take the form of $D = \sum_{i=1}^{6} X_i$ for Figure 2. Equation 4 is unlikely to hold, if D and $\mathbb{X}$ are concerning different metrics, for instance, if D encodes the response time and $\mathbb{X}$ represents resource consumption level on the components. The CPD has to be learned from data in this case.

3.4 Decentralized Parameter Learning

The rest of the KERT-BN CPDs, $P_{\mathcal{X}_i}(X_i|\Phi(X_i)), i = 1, 2, \ldots, n$ can be acquired by collecting the number of data instances $X_i = x_i \wedge \Phi(X_i) = \Phi(x_i)$, where x_i and $\Phi(x_i)$ are instances of elapsed times of X_i and $\Phi(X_i)$, and using any maximum likelihood (i.e. normalized counts) or Bayesian method [14].

Interestingly, the computation only requires data about X_i and all of its KERT-BN parents $\Phi(X_i)$. This data locality suggests that the computation of each $P_{\mathcal{X}_i}(X_i|\Phi(X_i))$ can actually be performed on service i. To this end, communication is set up between the monitoring agent (see Section 2) for service i and those agents for services corresponding to $\Phi(X_i)$. For services where $\Phi(X_i) = \emptyset$, no communication is needed. Such communication takes place periodically at a frequency that will not flood the network. The communicated data are batched together with locally collected data on each service i (services where $\Phi(X_i) = \emptyset$ batch local data only). In doing so, the parameter computation is initiated concurrently on all services and the results reported every T_{CON}.

Decentralizing the parameter learning computations has several attractions:

- The (potentially large) computational expense does not fall on a central management node which may become a bottleneck as the system scales. Even though the entire KERT-BN structure is still maintained in the central server, it will prove far more lightweight than storing and computing the CPDs and should not compromise scalability to a large extent.

- The computation can be performed concurrently reducing overall time taken.

It can be also noted from Figure 1 that user requests are sent from immediate upstream services (e.g. `image_locator_local`) to a downstream service (e.g. `ogsa_dai_local`). These communications can be leveraged to send elapsed time data from parents $\Phi(X_i)$ to X_i, by attaching the data in an extra SOAP segment at the end of the application request messages. The computation of $P_{\mathcal{X}_i}(X_i|\Phi(X_i))$ can then be conducted on service i when needed. This possibility will be explored further in the future.

4 Evaluation through Simulations

This section evaluates our approach through simulations under a comprehensive set of settings that are difficult to replicate in real-world systems. First, the performance of KERT-BN is compared against *Naive Response Time Bayesian Network (NRT-BN)* (i.e. learned purely from data via both structure learning with K2 [6] and parameter learning) in a decentralized process. Then the extra benefit of learning unknown (from domain knowledge) KERT-BN parameters in a decentralized manner is assessed.

4.1 Evaluation Settings and Metrics

Experiments in this section were conducted within a service-oriented system simulated in Matlab[12]. The simulated services receive and send calls among other and randomly generate a processing delay upon receiving calls. They are assembled together by different workflows to constitute simulated applications. The simulated delays (and response times) are used to form training and testing data sets. *Continuous* KERT-BN and NRT-BN models (where l in Equation 4 is assumed to be 0) with Gaussian CPDs were constructed using training data and then run against the testing data, so as to gauge their performance in terms of the following two metrics:

- *Construction time* - the time it takes to build the entire Bayesian network (i.e. including the structure and parameter values)

- *Data-fitting accuracy* - the log likelihood of testing data given the Bayesian network model,

It is worth mentioning that a mistake was actually made in formulating $D = X_1 + X_2 + max(X_3 + X_5, X_4 + X_6)$ in the initial version of the paper. The incident serves to reinforce the earlier argument that analytical model is prone to human errors. The use of a hybrid model in this paper has limited chances of such mistakes being made. However, certain risks are still being taken in the knowledge-given parts of the model in exchange for faster model construction time and superior accuracy. The risks are minimum because, unlike during the edition of the paper, the CPD is automatically generated by software.

$log_{10}p(TestData|BN)$ [14]. The higher the likelihood, the better the model fits (i.e. accurately represents) the testing data.

The experiments were implemented using the Matlab Bayesian network tool-box [13] and conducted on a Redhat Linux machine with two 3.0 GHZ dual core CPUs and 1GB memory. In all experiment runs, the environmental metric (see Section 2) was set to $K = 3$ to emulate model training in environments where data more than 2 construction intervals (see Section 2) old were deemed uncorrelated with present environment status. The data collection interval had the value of $T_{DATA} = 10$ seconds, a high frequency at which the reported data will not flood the network in a heavily loaded environment. All data-fitting accuracy numbers were measured against a training set of 100 data points.

4.2 KERT-BN vs. NRT-BN

In the first experiment, the construction time and data-fitting accuracy of both KERT-BN and NRT-BN built were compared for 30 simulated services. The training sets contained from 36 data points (i.e. $K * \alpha_{Model} = 3 * 12 = 36$ and model construction interval $T_{CON} = \alpha_{Model} * T_{DATA} = 2$ minutes) to 1080 data points (with $K * \alpha_{Model} = 3 * 360 = 1080$ and model construction interval $T_{CON} = 60$ minutes). For each training set size considered, the experiment was repeated 10 times (each with fresh training and testing data) to obtain average model performance.

The results of this experiment are shown in Figure 3. As one would expect, the construction time of both NRT-BN and KERT-BN grows linearly with the training set size. *KERT-BN consistently outperforms NRT-BN, and the advantage becomes increasingly evident, as the training set grows.* This observation can be explained by the fact that KERT-BN only requires partial parameter learning with the training data, whereas NRT-BN must go through both an expensive structural learning phase and a full parameter learning phase.

Despite skipping structural learning completely and parameter learning partly, KERT-BN still achieves a slightly superior accuracy as is illustrated in the right half of Figure 3. In addition, NRT-BN appears to be quite sensitive to training data size and require a relatively large number of data points (about 600 in Figure 3 with $\alpha_{Model} = 180$ and model construction interval $T_{CON} = 1800$ seconds $=$ 30 minutes) to reach a stable accuracy level. As a result, *if data collection is not sufficiently frequent, short model construction intervals (thus small α_{model}) might not be used with NRT-BN so as to avoid data accuracy loss.* In contrary, KERT-BN is not liable to such restrictions, as its accuracy converges much more quickly.

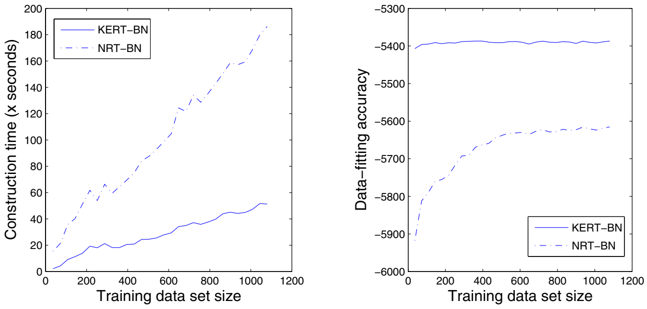


Figure 3. KERT-BN performance vs. NRT-BN performance with different training set sizes.

In the second experiment, the performance of KERT-BN and NRT-BN when applied to up to 100 simulated services (embodying small to medium data centers) was plotted. The models were trained with training sets of size 36 data points (again $\alpha_{Model} = 12$ and $T_{CON} = 2$ minutes) to study how practical it is to learn the models in fast changing environments of different sizes. This experiment was also repeated 10 times.

Figure 4 illustrates the outcome of the second experiment. An undesired non-linear correlation (discussed in Subsection 3.2) between NRT-BN construction time and the number of nodes in the model can clearly be seen in the left half of the figure. This correlation DE deteriorate even faster in bigger environments not illustrated in the figure, taking over 2 hours for 200 services, over 10 hours for 300, and more than 2 days for 500 services. *Consequently, for large environments, NRT-BN may simply be impossible to build at short model construction intervals, its learning time eclipsing how often it is supposed to be renewed.* In this experiment for example, NRT-BN being constructed at the designated interval of $T_{CON} = 2$ minutes will not be feasible for any environment with more than 60 services. In contrast, the construction time of KERT-BN remains flat across different environment sizes, as only relatively cheap parameter learning among service nodes is required. The right half of the figure once more confirms that KERT-BN achieves greater accuracy than NRT-BN under various settings (number of services in this experiment).

We also considered the possible use of a learning-free NRT-BN - one that simply settles for an arbitrary structure (e.g. the classic Naive Bayesian Network [14]) - in an attempt to offset its performance deficit against KERT-BN as manifested in the experiments in this section. The idea was nevertheless quickly dismissed, as not only is a learning-free NRT-BN even less accurate (than a NRT-BN) by construction, but its use will result in complete loss of model interpretability (i.e. the casual relationships among service

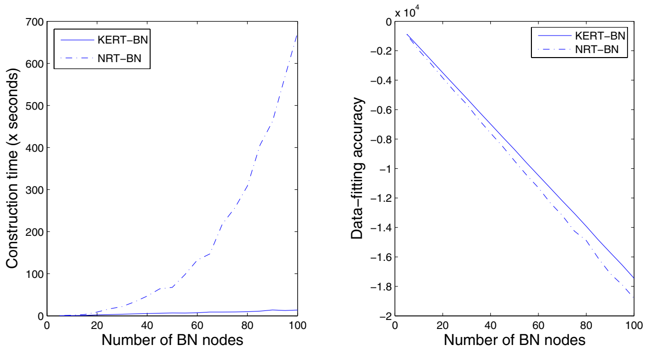


Figure 4. KERT-BN performance vs. NRT-BN performance with different environment sizes (i.e. service numbers).

elapsed time and response time that a KERT-BN or the original NRT-BN will encode) which is a fundamental strength of BN models.

4.3 Decentralized Parameter Learning vs. Centralized Parameter Learning

Having shown that KERT-BN outperforms NRT-BN in various centralized learning situations, it is the aim of this subsection to show that this advantage is indeed further extended by learning the unknown KERT-BN parameters in a decentralized manner, as is proposed in Subsection 3.4. To this end, the time taken to learn each CPD in KERT-BNs is measured. Since these CPDs will be computed in parallel on monitoring agents in practice, the decentralized learning time is the maximum of individual learning times across all CPDs, and is compared against the learning time for regular centralized KERT-BN parameter learning in Figure 5. The accuracy of these two KERT-BN parameter learning methods is not plotted on the grounds that they produce principally the same parameters.

For each KERT-BN size, the parameters of 20 randomly generated KERT-BNs were learned, with the average decentralized learning time and average centralized learning time plotted in the Figure. It can be observed that the decentralized learning time is constantly superior to the centralized learning time. The figure also reveals an obvious trend of this superiority becoming more and more considerable as the number of services (thus the number of KERT-BN CPDs) increases.

5 Applications

This section demonstrates the use of KERT-BN in tackling real-world performance management problems on

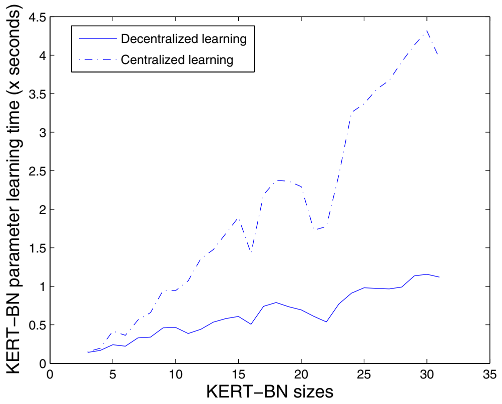


Figure 5. Decentralized learning time vs. centralized learning time for various environment sizes (i.e. service numbers).

the eDiaMoND test-bed, and further justifies the approach presented in this paper. The eDiaMoND services shown in Figure 1, `ogsa_dai_local`, `image_locator_local`, `ogsa_dai_remote`, and `image_locator_remote` were hosted by four AIX machines with two 3.0 GHZ dual core CPUs and 2GB memory. The `image_list` and `work_list` services were run on two separate 3.0 GHZ dual core CPUs of a Redhat Linux server with 1GB memory. Since the entire test-bed is within the same sub-net, extra routing was imposed between `image_list` and `image_locator_remote` through request forwarding to simulate a connection to a remote site.

Given that eDiaMoND fosters a relatively dedicated and stable system where $T_{DATA} = 20$ seconds, a bigger correlation metric $K = 10$ and a longer construction interval $T_{CON} = 20$ minutes ($\alpha_{Model} = 120$) were chosen. *Discrete* rather than continuous (as in Section 4) models are built for two reasons: 1) there are comparatively many data points to work with; 2) MATLAB BNT does not support non-linear deterministic CPDs that contain maximum relationships.

5.1 dComp: Compensating for Missing Data

In large distributed systems like the Grid, performance data from some system components may go missing due to 1) lack of instrumentation, 2) failure in the act of data reporting, or 3) the need to reduce monitoring overhead. To cope with the consequences, mechanisms must be provided to support estimation of performance states of *unobservable components* using data from the *observable* ones.

Typically, the only knowledge possibly available about elapsed time on unobservable components (whose latest performance data go missing) is from historical measurements or component providers. Such prior knowledge is likely to be obsolete or imprecise. *dComp* is developed as a technique that applies KERT-BN to update this prior knowledge with current elapsed time measurements from observable components as well as response time measurements. dComp computes a *Posterior (Probability) Distribution* by $p(Y|\mathbb{O} = E(\mathbf{o}))$ using standard BN-based inference [14] based on Equation 3 for each Y where no elapsed time data was available ($\mathbb{O}$ is the observable set of services and $E(\mathbf{o})$ is the current measurement mean). In the absence of using full blown fill-in methods (like Expectation Maximization), it suffices to just use the summary of observation statistics (the mean, $E(\mathbf{o})$) to assess Y.

To illustrate dComp, the KERT-BN in Figure 2 is considered. It is assumed that no service is observable. In Figure 6, dComp is employed to infer the posterior distribution for X_4 using observations on the rest of the variables. The figure compares the inferred posterior distribution of X_4 with the prior distribution of X_4, and demonstrates how the posterior distribution has updated our knowledge about X_4. The posterior distribution has shifted (from the prior distribution) toward the actual elapsed time and become more deterministic and precise with a narrower shape.

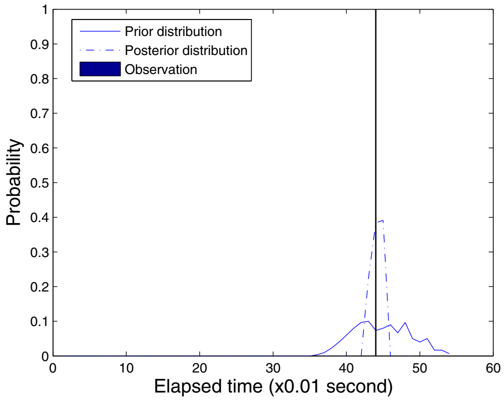


Figure 6. dComp example: posterior distribution of X_4 vs. prior distribution of X_4.

5.2 pAccel: Accelerating Key Service Performance

Due to the complexity of service-oriented environments, a significant performance boost for a particular service may not lead to system -wide benefits. For instance, if service A is being invoked in parallel with another service B that has a significantly longer elapsed time, reducing A's elapsed time can do little to improve the overall performance. A method to assess the end-to-end impact of local actions is essential to avoiding spending a lot of effort in accelerating those services that will bring little response time improvement.

pAccel is a means to achieve this goal through the use of KERT-BN. Based on Equation 3, pAccel computes the posterior response time distribution $p(D|Z = E(z))$ given the mean of a prediction about, Z, the elapsed time of any service. The observation mean is sufficient for the same reason as was argued in the previous subsection.

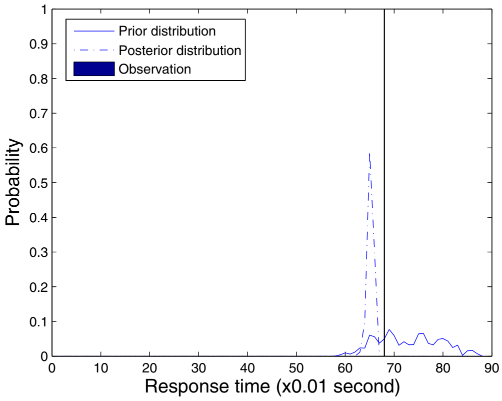


Figure 7. pAccel example: projected response time vs. observed response time, after accelerating X_4.

Recall the example in the previous subsection. A posterior response time distribution can now be computed with X_4 reduced to about 90% of what it was (e.g. after local resource allocation actions). Figure 7 shows that the posterior response time provides a good approximation of the actual improved response time mean. The difference between the posterior response time distribution and the prior response time distribution can be used to gauge the benefit of resource actions and guide autonomic decision making.

5.3 Justification

The efficacy of KERT-BN during model re-constructions in the eDiaMoND scenario in Figure 1 is examined by contrasting the response time distribution projected using pAccel with both KERT-BN and NRT-BN against real response time measurements.

Considering that both human users and autonomic software are likely to be interested in assessments like “What is the probability that response time will exceed the threshold(s)?”, we draw the comparison using *Relative Threshold*

Violation Probability Error defined as follows:

$$\epsilon = \frac{|P_{bn}(D > h) - P_{real}(D > h)|}{P_{real}(D > h)} \quad (5)$$

where $P_{real}(D > h)$ is the real probability that response time D will exceed threshold h, whereas $P_{bn}(D > h)$ is the threshold violation probability estimated with either KERT-BN or NRT-BN.

Figure 8 depicts the violation errors for KERT-BN and NRT-BN, when employed to make the violation probability projection for six different thresholds. Training sets of size 1200 ($K * \alpha_{Model} = 10 * 120 = 1200$) are used for model training purpose. Since there are few services and not many data points, the construction of both NRT-BN and KERT-BN can be finished quickly (relative to T_{CON}). Hence we can afford to repeatedly run K2 with different random orderings [6] until the next model construction is due, in order to optimize the accuracy of the learned NRT-BNs. Despite this optimization, nevertheless, the inferred NRT-BN is still inferior to KERT-BN as far as ϵ is concerned.

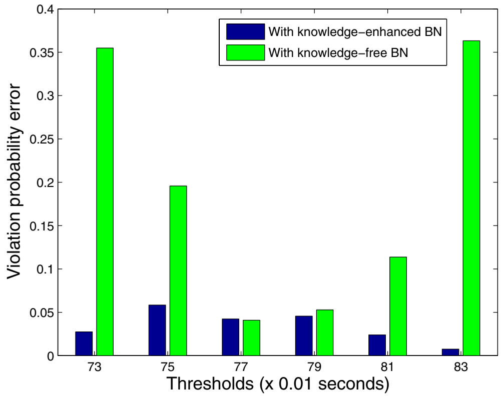


Figure 8. Relative threshold violation error (KERT-BN vs. NRT-BN) for the projected response time after accelerating X_4 in the eDiaMoND scenario in Figure 1.

6 Related Work

There are principally two schools of works in the literature concerning end-to-end response time modeling for multi-component distributed systems. The first school consists of analytical models based on queuing networks [20], petri nets [7] or control theory [11], that are constructed by human experts using domain analysis. Urgaonkar et. al. [20] regard a three-tier commercial environments as a queuing network and perform dynamic server allocations based on this model. Dingle et. al. [7] derives response time distributions given a Petri-net describing any distributed process, while Lu et. al [11] uses feedback control loop to guide server resource provisioning decisions. Although mature and mathematically sound, these models suffer from a number of shortcomings: 1) the models may be difficult to build, for instance, due to lack of information from some black-box components. 2) model construction can be prone to human errors; 3) models built are a priori and may not adapt well to workload or system changes; 4) assumptions that deviate significantly from real system conditions (e.g. the system being in stable state [20]) may have to be made.

The second school is embodied by emerging research rooted in statistical learning techniques using performance data collected through instrumentation. For example, tree-augmented naive Bayesian networks are learned from data to correlate system component resource state (CPU usage, memory usage etc.) with service level agreement violation/compliance states [9]. Unlike those in the first school, statistically learned models mostly assume very little or no domain knowledge and can be programmatically updated/reconstructed in response to system changes. Not only are these models often learned entirely from scratch at potentially substantial computational cost, but they can be rather sensitive to noisy or missing data. Moreover, statistical learning methods for response time modeling are still immature, with relatively little effort invested thus far. For example, to our best knowledge, there is yet any model of this type that describes end-to-end response time to a precision of more than two states.

The approach described in this paper overcomes these weaknesses, by incorporating domain knowledge in the statistical learning framework and decentralizing the learning procedure. Rish et. al [17] has also adopted a knowledge-enhanced strategy to construct a fault determination model. However, applying Rish's strategy here would be difficult given its dependency on binary state variables, binary domain relationships and strong independence assumptions.

7 Conclusions and Future Work

This paper presents an automated and cost-effective solution to end-to-end performance modeling for service-oriented systems, an essential issue towards providing autonomic capabilities in these environments. We leverage a combination of easily attainable domain knowledge (mainly the workflow in this paper) and performance data collected system-wide, to efficiently induce a knowledge-enhanced response time Bayesian network (KERT-BN) that accurately correlates system component performance to end-to-end QoS goals. For model elements that can not be determined in this way, a decentralized learning mechanism is proposed to reduce learning overhead incurred. Experiments in both simulation and real-world contexts show that the utilization of domain knowledge 1) largely reduces the model building cost without sacrificing model accuracy; and 2) rapidly guides model training towards an accurate product with few data. These features are particularly valuable in highly dynamic and complex environments where the current model quickly expires and must be rebuilt.

The approach is automated through employing an instrumentation-based technique to the automatic extraction of workflow and resource sharing information [21], programmatically feeding this knowledge together with performance data into the Bayesian network framework, and subsequently inducing the model under a periodical scheme. The automated model construction can serve to further minimize the need for human engagements in model-driven management procedures and steer them closer to complete autonomic solutions.

The KERT-BN approach can be effortlessly generalized and applied to any instrumented distributed system featuring user transactions that traverse system components. With minor adjustments, it may also be used to model the relationship between component-level metrics other than elapsed time (e.g. CPU or memory usage) and end-to-end performance goals.

Another important extension of our work is employing domain knowledge and decentralization techniques to reduce the cost of probability assessment after the model is constructed. Crucial autonomic routines such as resource provisioning and problem localization will profit greatly on rapid response time assessment. Adding this value to the present work will lead to accurate performance models that are inexpensive to build and use, and largely facilitate efficient, scalable autonomic system management.

8 Acknowledgement

Rui Zhang was funded by the UK Department of Trade and Industry project: Heterogeneous Workload Management and Grid Integration (Grant THBB/C/008/00025).

References

[1] P. Barham, A. Donnelly, R. Isaacs, and R. Mortier. Using magpie for request extraction and workload modelling. In *Proceedings of the 6th Symposium on Operating Systems Design and Implementation (OSDI'04),*, San Francisco, USA, December 2004. USENIX Association.

[2] A. C. Bhuvan Urgaonkar, Prashant Shenoy and P. Goyal. Dynamic provisioning of multitier internet applications. In *Proceedings of 2nd IEEE International Conference on Autonomic Computing*, pages 217 – 228. IEEE Computer Society Press, June 2005.

[3] J. M. Brady, D. J. Gavaghan, A. C. Simpson, M. Mulet-Parada, and R. P. Highnam. eDiaMoND: A grid-enabled federated database of annotated mammograms. In F. Berman, G. C. Fox, and A. J. G. Hey, editors, *Grid Computing: Making the Global Infrastructure a Reality*, pages 923–943. Wiley Series, 2003.

[4] J. Cardoso, J. Miller, A. Sheth, and J. Arnold. Modeling quality of service for workflows and web service processes. Technical report, LSDIS Lab, Computer Science, University of Georgia, May 2002.

[5] G. F. C. E. Clark Glymour (Editor). *Computation, Causation and Discovery*. MIT Press, 1999.

[6] G. Cooper and E. Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, pages 309–347, 1992.

[7] N. J. Dingle, P. G. Harrison, and W. J. Knottenbelt. Response time densities in generalised stochastic petri net models. In *Proceedings of 3rd international workshop on Software and performance (WOSP'02)*, pages 46–54, New York, NY, USA, 2002. ACM Press.

[8] N. Friedman and M. Goldszmidt. Sequential update of Bayesian network structure. In *13th Conf. on Uncertainty in Artificial Intelligence*, pages 165–174, 1997.

[9] C. Ira, G. Moises, K. Terence, S. Julie, and C. Jeffrey. Correlating instrumentation data to system states: A building block for automated diagnosis and control. Technical report, HP Labs, November 2004.

[10] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.

[11] Y. Lu, T. Abdelzaher, C. Lu, L. Sha, and X. Liu. Feedback control with queueing-theoretic prediction for relative delay guarantees in web servers. In *Proceedings of the The 9th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'03)*, page 208, Washington, DC, USA, 2003. IEEE Computer Society.

[12] C. Moler. *Numerical Computing with MATLAB*. Society for Industrial and Applied Mathematics, 2004.

[13] K. Murphy. Bayes net toolbox for matlab. *Computer Science and Statistics*, 2001.

[14] R. Neapolitan. *Probabilistic Reasoning in Expert Systems*. Wiley Interscience, 1989.

[15] M. P. Papazoglou and D. Georgakopoulos. Introduction. *Commun. ACM*, 46(10):24–28, 2003.

[16] N. M. P. Peter G. Harrison. *Performance modelling of communication networks and computer architectures*. Wokingham, Addison-Wesley, 1992.

[17] I. Rish, M. Brodie, and S. Ma. Accuracy vs. efficiency tradeoffs in probabilistic diagnosis. In *18th national conference on Artificial intelligence*, pages 560–566. American Association for Artificial Intelligence, 2002.

[18] S. Russell and P. Norvig. *Artificial Intelligence: A Morden Approach*. Prentice Hall, 2003.

[19] D. J. Spiegelhalter and S. L. Lauritzen. Sequential updating of conditional probabilities on directed graphical structures. *Networks*, pages 579–605, 1990.

[20] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, and A. Tantawi. An analytical model for multi-tier internet services and its applications. *SIGMETRICS Performance Evaluation Review*, 33(1):291–302, 2005.

[21] R. Zhang, S. Heisig, S. Moyle, and S. McKeever. Ogsa-based grid workload monitoring. In *Proceedings of 5th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid05)*, pages 668–675. IEEE Computer Society Press, May 2005.