2008 Eighth IEEE International Conference on Data Mining

Exploiting Local and Global Invariants for the Management of Large Scale Information Systems

Haifeng Chen[1] Haibin Cheng[2] Guofei Jiang[1] Kenji Yoshihira[1]
[1] NEC Laboratories America, Inc.
4 Independence Way, Princeton, NJ 08540
{haifeng, gfj, kenji}@nec-labs.com
[2] CSE Department, Michigan State University
East Lansing, MI 48824
chenghai@msu.edu

Abstract

This paper presents a data oriented approach to modeling the complex computing systems, in which an ensemble of correlation models are discovered to represent the system status. If the discovered correlations can continually hold under different user scenarios and workloads, they are regarded as invariants of the information system. In our previous work [12], we have developed an algorithm to automatically search the invariants between any pair of system attributes, which we call local invariants. However that method is unable to deal with the high order dependency models due to the combinatorial explosion of search space. In this paper we use Bayesian regression technique to discover those high order correlation models, called global invariants. We treat each attribute as a response variable in turn and express its dependency with the other attributes in a regression model. By adding the prior constraint of Laplacian distribution to the regression coefficients, we can find the solution in which only the correlated attributes with respect to the response have nonzero regression coefficients. After that we further consider the temporal dependencies of those extracted attributes by incorporating their past observations. We also provide a confidence metric and a validation procedure to measure the reliability of learned models. If the model does not break down in the validation, it is regarded as a true invariant of the system. Experimental results on a real wireless networking system show that the discovered invariants can be used to effectively detect system failures as well as provide valuable information about the failure source.

1 Introduction

With the rapid advances in networking and computing technology, we are facing an explosive growth of complexity in networked applications and information services. Typical systems such as Google, Yahoo! and Amazon consist of thousands of components including operating systems, applications software, servers, and storage and networking devices, which are widely distributed across the system. Since the system complexity appears to be approaching the limits of human capability, it is crucial to develop autonomic management tools to reduce the burden of system operators.

Meeting the grand challenges of such self-management requires advanced techniques to correctly characterize the system state. One approach to achieving this relies on the expert knowledge of system structure and turns such prior knowledge into a set of event-condition-action rules to model the system [1][8]. However, the models and rules provided by those approaches are usually system dependent and may be incomplete and inaccurate. Recently statistical techniques [5][4][6] have been drawing a surge of interest in modeling the system activity given that large amount of measurement data can be collected from system monitoring tools. This data contains many implicit evidences about the system structure and activity. By discovering those evidences from the data and correlating them to the high level system behavior, we can effectively characterize the system status and hence facilitate many management tasks. However, there are usually huge amount of attributes and observations in the measurement data due to the large scale of computing systems, which makes the data modeling a big challenge. For instance, commercial frameworks such as HP's OpenView [7] and IBM's Tivoli [11] aggregate attributes from a variety of sources including hardware, networking, operating systems, application servers, and so on. It is hard to create a single powerful model that can effectively describe the system and efficiently deploy in practice.

To address this issue, this paper presents an invariants based solution to model the behavior of information systems. We learn an *ensemble* of correlation models among system attributes from a large amount of measurement data. If the dependency models continually hold under different user scenarios and workloads, they are regarded as *invariants* of the computing system. For example, the frequency

1550-4786/08 $25.00 © 2008 IEEE
DOI 10.1109/ICDM.2008.51

of SQL queries issued in the backend database in a web based system is usually correlated with the frequency of HTTP requests issued at the front end as a result of fixed business logic of the application. Such relationship is usually difficult to be quantitatively described by the system operators. Our algorithm can automatically discover those implicit constraints in the system to assist operators to understand the system internal status. As a result, many management tasks can be facilitated such as failure detection, capacity planning, policy generation, and so on. For instance, by online checking the number of 'healthy' invariants, i.e., those models that fit the online measurements well, we can identify anomalies during system operation.

In our previous work [12], a brute force search method has been developed to explore all the pair-wised correlation models in the system, which we call *local* invariants. In distributed systems, however, some attributes may depend on more than one other attributes. For instance, the input of a load balancer is always equal to the summation of multiple output variables. If we use the method in [12] to explore high order correlation models, which we call *global* invariants, we need to check all the models with various combinations of system attributes. In order to avoid the problem of 'combinatorial explosion' for large scale computing systems, this paper uses a statistical mining based approach to discovering global invariant models for the system. Given the measurement data, we use the regression model to express the correlation between each attribute and its related partners. That is, we treat each attribute as a response variable in turn and express its dependency with the correlated attributes in a regression model. Since we do not know how many attributes are relevant to the response and what those attributes are, we put all the rest attributes in the regression as *possible* independent variables at the beginning. We add Laplacian priors on the regression coefficients and formulate the dependency discovery in the Bayesian regression framework. As a result, we can expose a subset of attributes that have statistically significant relationship with the response. If such high order relationship is strong enough, we further explore the temporal dependency between the response and discovered subset of relevant attributes by incorporating their past observations. We also provide a confidence metric and a validation procedure to measure the reliability of learned models. If the model does not break down in the validation, it is regarded as a true invariant of the system. Comparing with the local invariants as described in [12], the global invariants reveal more complex dependency relationships among the system attributes.

We have successfully applied the discovered invariants to the management of a 3G telecommunication system. It shows that the discovered global invariants are necessary additions to the previous learned local invariants. Combining local and global invariants can present a more comprehensive and effective characterization of the system behavior. The discovered invariants have been successfully applied to detect and localize the system failures. Experimental results based on the real system data demonstrate that the percentage of broken invariants is an effective indication of the failure occurrence. In addition, it is easy to pinpoint the suspicious attributes from the broken invariant set, which significantly facilitates the failure localization task.

2 Background and Related Work

To support the autonomic management of computing systems, several statistical approaches have been proposed in the literature. In the project of Magpie [2], Barham and his colleagues collected request traces from software components in the distributed system, and used the stochastic context free grammar to model the request's control flow for the purpose of detecting component failures as well as localizing performance bottlenecks. The Pinpoint project [5], a close relative to Magpie, used request path shapes and component interactions as two features to represent the user request flow across the system. The probabilistic context free grammar (PCFG) and χ^2 statistical test were employed to model those features for detecting high level software failures in application servers. In the same context of component interaction analysis as in [5], [4] put forward a subspace decomposition based algorithm to explore the correlations between interaction profiles of multiple components. A simplified Bayesian network model was presented by Cohen et al. [6] to discover the root cause of service level agreement (SLA) violations. In the work [3], Bodik et al. made use of the user access behavior as an evidence of system's health, and applied several statistical approaches such as the Naive Bayes to mining such information for detecting application level failures.

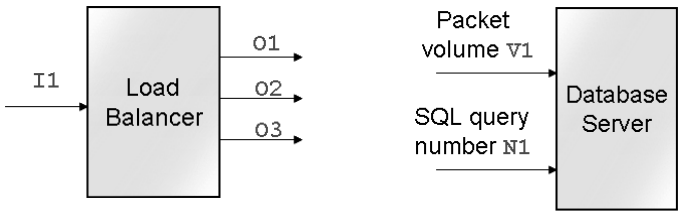


Figure 1. Examples of invariants in information systems. (a)The input-output relationship in a load balancer; (b)Packet volume vs. number of SQL query in a database.

The above approaches mostly focused on either a specific type of system measurements such as the request

traces, or a specific system management task. Compared with them, our invariant based solution can accept the data measurements from various sources such as software log files and network traffic statistics, and generalize well for many system management scenarios. We observe that there exist invariant properties of system measurements since most of the attributes in measurement data are strongly correlated. For example, the resource utilizations of the system such as CPU and memory usages always increase or decrease in accordance with the change of system workloads. Furthermore, the system structure and design also introduce a lot of correlated attributes. As shown in Figure 1(a), assuming that a load balancer has one input I_1 and three outputs O_1, O_2 and O_3, we should always have $I_1 = O_1 + O_2 + O_3$ because this is the physical property of the load balancer. Meantime, in a web system, the volume of network packets V_1 going through the database, as shown in Figure 1(b), should always be correlated with the number of queries issued in the database because of the underlying logic of system components. We believe there are many of such constant relationships among the measured attributes. As a consequence, an *ensemble* of invariant models can be discovered to correlate the large amount of monitoring data collected from various points of the system. Each invariant contains a specific dependency relationship among the measurement attributes, which reveals some specific aspects of system activity. If we put all the discovered invariants together we can achieve a comprehensive view of the system situation and facilitate many management tasks. For example,

- By monitoring the number of 'healthy' invariants during system operation, we can detect unknown system failures. The detected failures can also be isolated based on the correlation of broken invariants and their related system attributes.
- Since each invariant is usually represented as a dependency function, we can use such function for system capacity planning. That is, we can predict the capacity requirements for system resources such as CPU, memory and disks under heavy workload based on the dependency functions as well as those resource usages under light system workload.
- The discovered invariants can assist system administrators to better understand the system and define rules that distinguish different system activities.

The invariants are usually identified by analyzing the historical system measurements. Given a set of monitoring data $\mathbf{z}_l$, $l = 1, \ldots, n$, with each measurement containing p attributes, $\mathbf{z}_i \in R^p$, we divide the data into two parts: one is used for training, and the other is for the validation. The purpose of training is to build dependency relationships among the data attributes, which serve as the hypotheses of system invariants. The validation process uses the second part of data to mimic actual system dynamics and sequentially validate the effectiveness of hypotheses under those situations. The hypotheses that pass the validation process are regarded as true invariants of the information system. In Section 3 we describe the work [12] of modeling the local invariants in the training process. Section 4 then presents our proposed method of discovering system global invariants. The validation of those discovered correlation models are described in Section 5.

3 Learning Local Invariants

The proposed system invariant model has to consider both spatial and temporal dependencies among system attributes. While spatial relationship concerns the correlation across different attributes, the temporal correlation is related to dependency in consecutive measurements of each attribute along time. In the work [12], a brute force search method is used to build correlation models between any pair of attributes x and y from the data $\mathbf{z}$, $y = f(x)$. The relationship between $x(t)$ and $y(t)$ is learned by the AutoRegressive model with eXogenous inputs (ARX) [13]

$$y(t)+a_1y(t-1)+\cdots+a_ny(t-n) \;=\; b_0x(t)+\cdots+b_mx(t-m) \tag{1}$$

where $[n, m]$ is the order of the model, which determines how many previous steps are affecting the current output. a_i and b_j are the coefficient parameters that reflect how strongly a previous step is affecting the current output. Let's denote

$$\boldsymbol{\theta} \;=\; [a_1, \cdots, a_n, b_0, \cdots, b_m]^T, \tag{2}$$

$$\boldsymbol{\varphi}(t) \;=\; [-y(t-1), ..., -y(t-n), x(t), ..., x(t-m)]^T \tag{3}$$

then Equation (1) can be rewritten as:

$$y(t) = \boldsymbol{\varphi}(t)^T\boldsymbol{\theta}\,. \tag{4}$$

Assuming that we have observed the measurements over a time interval $1 \le t \le N$, $O_N = \{x(1), y(1), ..., x(N), y(N)\}$, the parameter θ can be obtained by the least squares solution

$$\hat{\theta}_N = [\sum_{t=1}^{N} \boldsymbol{\varphi}(t)\boldsymbol{\varphi}(t)^T]^{-1} \sum_{t=1}^{N} \boldsymbol{\varphi}(t)y(t). \tag{5}$$

In order to evaluate how well the learned model (1) fits the measurement data, the following fitness score is defined

$$F(\theta) = [1 - \sqrt{\frac{\sum_{t=1}^{N} |y(t) - \hat{y}(t|\theta)|^2}{\sum_{t=1}^{N} |y(t) - \bar{y}|^2}}] \cdot 100, \tag{6}$$

where $\hat{y}(t|\theta)$ is the estimated value of y based on equation (4) and $\bar{y}$ is the mean of the real output $y(t)$. A higher score

$F(\theta)$ indicates that the model fits the observed data better and its upper bound is 100. Only a model with high fitness score is selected to characterize the data relationship. We can set a range of the order $[n, m]$ to learn a list of model candidates and select a right model from them according to the highest fitness score. For more detailed description of local invariants extraction, please see [12].

Given measurements with p attributes, the above method needs to construct $p(p-1)/2$ models as all possible candidates of local invariants, i.e., pair wise dependency models. If we want to discover the high order dependency models among attributes, however, the above brute force search does not scale well because it requires the search of all the models with various combinations of system attributes. In the following we propose a statistical mining based method to discover those high order correlations.

4 Learning Global Invariants

While the local invariants reveal the correlations between any pair of measurement attributes, the global method explores the relationship between each attribute y and *multiple* other attributes $\tilde{\boldsymbol{x}} = [x_1, x_2, \cdots, x_s]$, $2 \leq s \leq p-1$, where p is the total number of attributes in the measurement data $\mathbf{z}$. Since all the attributes in the data are measured as time series, we have to model both the spatial and temporal correlations among the attributes. To simplify the problem, we separate these two stages in the algorithm. In the first step, we explore all the contemporary dependencies among attributes, e.g. identify the subsets of attributes that exhibit strong spatial correlations. Given the measurements $\mathbf{z}$ with p attributes, we treat each attribute y as the response in turn and express the dependency between y and other attributes in a regression model. Since we do not know how many attributes are relevant to y and what those attributes are, at the beginning we put the other $p-1$ attributes in the model as *possible* independent variables

$$y(t) = \beta_1 x_1(t) + \beta_2 x_2(t) + \cdots + \beta_{p-1} x_{p-1}(t) \quad (7)$$

We use the Bayesian technique to trim the regression coefficients β_is. That is, by choosing proper prior distribution for the regression coefficients, we can force the coefficients whose associated attributes have no relationship with the response y to be exactly zeroes. As a result, we can extract a subset of variables, whose corresponding coefficients are not zero, as correlated variables with respect to the response y.

We use the fitness score as described in (6) to measure the significance of the statistical relationship between y and the extracted attributes $\tilde{\boldsymbol{x}}$. If it is strong enough, we further explore their dependency by incorporating their past observations

$$y(t) = f(\tilde{\boldsymbol{x}}(t), \tilde{\boldsymbol{x}}(t-1), \cdots, \tilde{\boldsymbol{x}}(t-m), y(t-1), \cdots, y(t-n)) \,. \quad (8)$$

Note this time only a small number of attributes, $x_1, \cdots, x_s$, are involved in the time series modeling. We still use the Bayesian approach but with slight modification to identify the function (8). Eventually we obtain a model that contains both the spatial and temporal correlations between y and its related attributes.

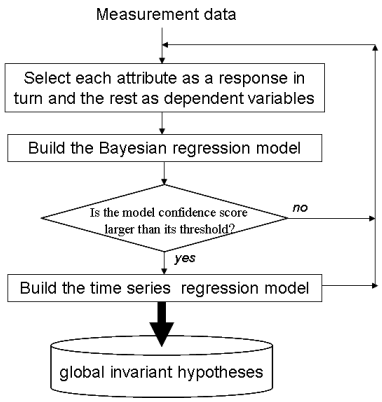


Figure 2. The workflow of extracting global invariant hypotheses.

Figure 2 presents the workflow of extracting global dependency models. In Section 4.1, we focus on the extraction of spatial correlations among attributes. Section 4.2 then models the temporal correlations among attributes.

4.1 Modeling Spatial Correlations

This section focuses on the modeling of spatial dependencies among attributes. We rewrite equation (7) in a simplified way

$$y = \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_{p-1} x_{p-1} \quad (9)$$

where $\boldsymbol{\beta} = [\beta_1, \beta_2, \cdots, \beta_{p-1}]^\top$ is the regression coefficients. The common solution such as the least squares estimation for equation (9) is to maximize the likelihood

$$p(y|\boldsymbol{\beta}) = \prod_{l=1}^{n} p(y_l|\boldsymbol{\beta}) \;\propto\; (\frac{1}{\sqrt{2\pi\sigma^2}})^n \prod_{l=1}^{n} exp\{-\frac{(y_l - \boldsymbol{x}_l^\top \boldsymbol{\beta})^2}{2\sigma^2}\} \,, \quad (10)$$

where $[\boldsymbol{x}_l \;\; y_l]$ is the lth observation of measurement data and σ^2 is the variance of residuals $\epsilon_l = y_l - \boldsymbol{x}_l^\top \boldsymbol{\beta}$, $l = 1, \cdots, n$. In order to extract a subset of variables that have relationship with the response y, we make a prior assumption of the distribution on regression coefficients $\boldsymbol{\beta}$ and use Bayesian method to find the maximum a posterior solution of (9). It has been shown in [9] that if we choose the Laplacian prior on each coefficient β_i

$$p(\beta_i|\gamma) = \frac{\sqrt{\gamma}}{2} exp(-\sqrt{\gamma}|\beta_i|) \quad (11)$$

we can shrink the coefficients of irrelevant input variables to exactly zero. That is, given the data set $D = [(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_n, y_n)]$, by maximizing the following posterior distribution

$$p(\boldsymbol{\beta}|D, \gamma) \propto p(y|\boldsymbol{\beta}) p(\boldsymbol{\beta}|\gamma) \tag{12}$$

instead of equation (10), we will get a model in which the coefficients of irrelevant attributes are zeroes. Since the variance σ^2 in (10) is also unknown, we incorporate it into the optimization process and revise the posterior (12) as

$$p(\boldsymbol{\beta}, \sigma^2|D, \gamma) \propto p(y|\boldsymbol{\beta}, \sigma^2) p(\boldsymbol{\beta}|\gamma) \; . \tag{13}$$

The parameter γ in the Laplacian prior (11) is a predefined constant. Here we choose $\gamma = 0.1$.

Since the Laplacian distribution (11) is non-convex, we rewrite the Laplacian prior (11) as a hierarchical decomposition of two other distributions: a zero-mean Gaussian prior $p(\beta_i|\tau_i) = \mathcal{N}(\beta_i|0, \tau_i)$ with a variance τ_i that has an exponential hyper prior

$$p(\tau_i|\gamma) = \frac{\gamma}{2} exp\{-\frac{\gamma}{2}\tau_i\} \; . \tag{14}$$

As a result, the distribution (13) can be rewritten as

$$p(y|\boldsymbol{\beta}, \sigma^2) p(\boldsymbol{\beta}|\gamma) = p(y|\beta, \sigma^2) p(\beta|\boldsymbol{\tau}) p(\boldsymbol{\tau}|\gamma) \; . \tag{15}$$

Suppose we could observe the values of new parameter $\boldsymbol{\tau} = [\tau_1, \tau_2, \cdots, \tau_{p-1}]^\top$, then $p(\boldsymbol{\tau}|\gamma) = 1$ and the posterior distribution (15) is simplified because both $p(\boldsymbol{y}|\beta, \sigma^2)$ and $p(\beta|\boldsymbol{\tau})$ in the right side of equation (15) are Gaussian distributions. We write the log-posterior of the right side in equation (15)

$$\log\{p(\boldsymbol{y}|\beta, \sigma^2) p(\beta|\boldsymbol{\tau})\} \; \propto \; -n \log \sigma^2 - \frac{\|\boldsymbol{y} - \mathbf{H}\boldsymbol{\beta}\|_2^2}{n} - \boldsymbol{\beta}^\top \Gamma(\boldsymbol{\tau}) \boldsymbol{\beta} \tag{16}$$

where the matrix $\mathbf{H} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n]^\top$, and $\Gamma(\boldsymbol{\tau}) = diag(\tau_1^{-1}, \cdots, \tau_{p-1}^{-1})$ is the diagonal matrix with the inverse variances of all β_is. By taking the derivatives with respect to $\boldsymbol{\beta}$ and σ^2 respectively, we obtain the solution that maximizes (16).

In reality, however, since we do not know the values of $\boldsymbol{\tau}$ (and hence the matrix $\Gamma(\boldsymbol{\tau})$ in (16)), we can not maximize the equation (16) directly. Instead the following expectation-maximization (EM) algorithm is used to find the solution. The EM algorithm is an iterative process which computes the expectation of hidden variables $\boldsymbol{\tau}$ and uses such expectation as the estimation of $\boldsymbol{\tau}$ to find the optimal solution. Each iteration consists of two steps.

- the *E-step* computes the conditional expectation of $\Gamma(\boldsymbol{\tau})$ given $\boldsymbol{y}$ and the current estimate $\hat{\sigma^2}_{(t)}$ and $\hat{\boldsymbol{\beta}}_{(t)}$

$$\begin{aligned} \mathbf{V}(t) &= E[\Gamma(\boldsymbol{\tau})|\boldsymbol{y}, \hat{\sigma^2}_{(t)}, \hat{\boldsymbol{\beta}}_{(t)}] \\ &= diag\{E[\tau_1^{-1}|\boldsymbol{y}, \hat{\sigma^2}_{(t)}, \hat{\boldsymbol{\beta}}_{(t)}], \cdots, E[\tau_p^{-1}|\boldsymbol{y}, \hat{\sigma^2}_{(t)}, \hat{\boldsymbol{\beta}}_{(t)}]\} \; . \end{aligned} \tag{17}$$

Since

$$\begin{aligned} & E[\tau_i^{-1}|\boldsymbol{y}, \hat{\sigma^2}_{(t)}, \hat{\boldsymbol{\beta}}_{(t)}] \\ &= \frac{\int_0^\infty \frac{1}{\tau_i} \mathcal{N}(\hat{\boldsymbol{\beta}}_{i,(t)}|0, \tau_i) \frac{\gamma}{2} exp\{\frac{\gamma}{2}\tau_i\} d\tau_i}{\int_0^\infty \mathcal{N}(\hat{\boldsymbol{\beta}}_{i,(t)}|0, \tau_i) \frac{\gamma}{2} exp\{\frac{\gamma}{2}\tau_i\} d\tau_i} = \frac{\gamma}{|\hat{\boldsymbol{\beta}}_{i,(t)}|} \end{aligned} \; , \tag{18}$$

thus

$$\mathbf{V}(t) = \gamma diag\{|\hat{\boldsymbol{\beta}}_{1,(t)}|^{-1}, \cdots, |\hat{\boldsymbol{\beta}}_{p-1,(t)}|^{-1}\} \; . \tag{19}$$

- The *M-step* carries out the maximization of (16) with respect to σ^2 and $\boldsymbol{\beta}$ except that the matrix $\Gamma(\boldsymbol{\tau})$ is replaced with its conditional expectation $\mathbf{V}(t)$. We obtain

$$\begin{aligned} \hat{\sigma^2}_{(t+1)} &= arg \max_{\sigma^2} (-n \log \sigma^2 - \frac{\|\boldsymbol{y} - \mathbf{H}\boldsymbol{\beta}\|_2^2}{\sigma^2}) \\ &= \frac{\|\boldsymbol{y} - \mathbf{H}\hat{\boldsymbol{\beta}}_{(t)}\|_2^2}{n} \end{aligned} \tag{20}$$

and

$$\begin{aligned} \hat{\boldsymbol{\beta}}_{(t+1)} &= arg \max_{\beta} \left(-\frac{\|\boldsymbol{y} - \mathbf{H}\boldsymbol{\beta}\|_2^2}{\sigma^2} - \boldsymbol{\beta}^\top \mathbf{V}(t) \boldsymbol{\beta} \right) \\ &= (\hat{\sigma^2}_{(t+1)} \mathbf{V}(t) + \mathbf{H}^\top \mathbf{H})^{-1} \mathbf{H}^\top \boldsymbol{y} \end{aligned} \tag{21}$$

The EM algorithm is easy to implement, and converges to the maximum of posterior probability (13) quickly.

4.2 Modeling Temporal Dependencies

The coefficient vector $\hat{\boldsymbol{\beta}}$ computed from the above EM algorithm contains many zeroes. By selecting those attributes with nonzero coefficients, we obtain a subset of variables $\tilde{\boldsymbol{x}} = [x_1, x_2, \cdots, x_s]$ that are correlated with y as well as a function $\tilde{f}$ to express such dependency

$$y = \tilde{f}(x_1, \cdots, x_s) \; . \tag{22}$$

We measure the fitness of function $\tilde{f}$ using the metric (6), in which $\hat{y}$ is the estimated value of y based on equation (22). If the confidence score is low, we discard that model. Otherwise, we further consider the temporal dependency between y and the subset $\tilde{\boldsymbol{x}} = [x_1, x_2, \cdots, x_s]$ by incorporating their past observations. Such dependency can be described in the following multivariate autoregressive model with eXogenous inputs

$$\begin{aligned} & y(t) + a_1 y(t-1) + \cdots + a_n y(t-n) = \\ & b_{01} x_1(t) + \cdots + b_{0s} x_s(t) + b_{11} x_1(t-1) + \cdots + b_{1s} x_s(t-1) \\ & \quad + \cdots + b_{m1} x_1(t-m) + \cdots + b_{ms} x_s(t-m) \; . \end{aligned} \tag{23}$$

The values $[n, m]$ denote the maximum time lags for y and x_is respectively in the autoregressive modeling. If we define

$$\boldsymbol{\theta} = [a_1, \cdots, a_n, b_{01}, \cdots, b_{0s}, b_{11}, \cdots, b_{1s}, b_{m1}, \cdots, b_{ms}]^T \tag{24}$$

$$\begin{aligned} \boldsymbol{\varphi}(t) = [& -y(t-1), ..., -y(t-n), x_1(t-k), ..., x_s(t-k), \\ & \cdots, x_1(t-k-m), ..., x_s(t-k-m)]^T \end{aligned} \tag{25}$$

the equation (23) can be rewritten in the same format as (4), and its solution can be obtained by the equation (5).

Note we put the same maximum time lag m for all attributes $x_1, \cdots, x_s$ in equation (23). In reality, however, different attributes may have temporal dependency with $y(t)$ up to different maximum time lags. For example, some attributes may only have contemporaneous dependency with $y(t)$, e.g. $m = 0$. If we place the same maximum order m for all variables and solve the equation (23) by traditional least squares method, the results may not be accurate because many irrelevant variables are involved in the model (23). Those irrelevant parts have to be removed in the estimation process.

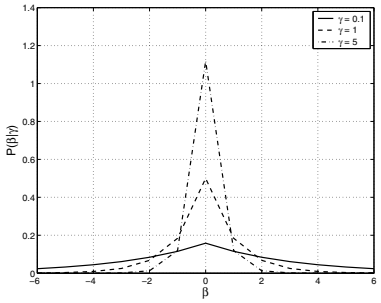


Figure 3. The effects of different γ values on the Laplacian distribution.

We still use the Bayesian method in Section 4.1 to prune the irrelevant variables. The difference here is that we are more interested in pruning those inputs with longer time delays because those inputs are more likely to be irrelevant to the response $y(t)$. As a result, we choose different prior parameter γ values in the Laplacian distribution (11) for different coefficients to control the level of penalties for those coefficients. Figure 3 plots the shape of Laplacian distribution with different γ values. It shows that a large γ value strengthens the constraint that the coefficient β will be zero. On the other hand, small γ value conveys our expectation that the corresponding variable is correlated to the response. We utilize such knowledge and choose different γ values for the variables with different time lags in (23). While the coefficients $a_1, \cdots, a_n$ and $b_{01}, \cdots, b_{0s}$ in (24) are added with Laplacian prior (11) with parameter γ, the coefficients $b_{i1}, \cdots, b_{is}$, $i = 1, \cdots, m$ are imposed with the same prior distribution with parameter $\sqrt{i+1}\gamma$. By doing so, the old observations are penalized more severely. The same EM algorithm described in Section 4.1 is used to obtain the MAP solution $\boldsymbol{\theta}$. The only modification is that in the E-step the matrix $V(t)$ in equation (19) is replaced with

$$\mathbf{V}(t) = diag\{\gamma_1 |\hat{\boldsymbol{\beta}}_{1,(t)}|^{-1}, \cdots, \gamma_{p-1} |\hat{\boldsymbol{\beta}}_{p-1,(t)}|^{-1}\} \quad (26)$$

because of different γ values. We learn a list of model candidates from a range of orders $[n, m]$ and then select the one with the highest fitness score (6) as the final global invariant hypothesis associated with $y(t)$.

5 Invariants Validation

Given the measurement data, we treat each attribute as the response variable in turn and use the algorithm presented in Section 4 to obtain following high order correlation model

$$\begin{aligned} & y(t) + a_1 y(t-1) + \cdots + a_n y(t-n) \\ = & b_{0,1} x_1(t) + \cdots + b_{k_1,1} x_1(t-k_1) + b_{0,2} x_2(t) + \cdots + \\ & b_{k_2,2} x_2(t-k_2) + \cdots + b_{0,s} x_1(t) + \cdots + b_{k_s,s} x_s(t-k_s) \end{aligned} \quad (27)$$

where $k_1, \cdots, k_s$ are the maximum time lags for each variable. As a consequence we get a set of correlation models. However, those models only fit the training data well. They may not work properly for other data that contains different user scenarios and system workloads. The purpose of validation is to use another data set called 'validation data' to test the validity of invariant hypotheses and hence remove the unqualified ones. Those correlation models that survive the validation process are regarded as system invariants. For simplicity we express the equation (27) as the $y(t) = F(\theta)$, where $\boldsymbol{\theta}$ denotes all the coefficients in the model

$$\boldsymbol{\theta} = [-a_1, \cdots, -a_n, b_{0,1}, \cdots, b_{k_1,1}, \cdots, b_{0,s}, \cdots, b_{k_s,s}] \ . \quad (28)$$

We divide the validation data into K windows, each of which contains l measurements. For every window W_i, we use its data to fit each hypothesis $\boldsymbol{\theta}$ and calculate the fitness score $F(\theta)$ given by Equation (6). Further we select a threshold $\widetilde{F}$ to determine whether the model fits that segment of data or not

$$f_i(F(\theta)) = \begin{cases} 1 & \text{if } F(\theta) > \widetilde{F} \ , \\ 0 & \text{if } F(\theta) \leq \widetilde{F} \ . \end{cases} \quad (29)$$

After receiving k such windows of data, $1 \leq k \leq l$, we can calculate an overall confidence score for the invariant hypothesis $\boldsymbol{\theta}$ which is the average of fitness scores computed from previous k windows

$$p_k(\theta) = \frac{\sum_{i=1}^{k} f_i(F(\theta))}{k} \ . \quad (30)$$

We compare the value $p_k(\theta)$ with predefined threshold P. If $p_k(\theta) < P$, the invariant hypothesis $\boldsymbol{\theta}$ is discarded. Those hypotheses that survive in all the test windows are finally regarded as the invariants of system. Note in the implementation the confidence $p_k(\theta)$ in (30) can be online updated for each window of dataset

$$p_k(\theta) = \frac{p_{k-1}(\theta) \cdot (k-1) + f(F_k(\theta))}{k} \ . \quad (31)$$

6 Experimental Results

We use the real monitoring data from a Universal Mobile Telecommunication System(UMTS) to demonstrate the results of extracting both local and global invariants in the system. We also apply the discovered invariants to detect and localize failures in the system.

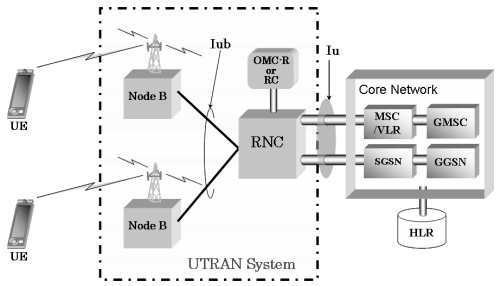


Figure 4. The architecture of UMTS network.

The UMTS is the third generation (3G) wireless network, which is divided into a Core Network (CN) and UMTS Terrestrial Radio Access Network (UTRAN) as shown in Figure 4. The CN is responsible for switching/routing calls and data connections to external networks. The UTRAN consists of a set of radio network subsystems(RNSs) connected to the CN. Figure 4 plots one of the RNS subsystems, which is composed of a radio network controller (RNC) and one or more Node Bs. While RNC is in charge of controlling the use of radio resources, the Node B is responsible for radio transmission/reception to/from the User Equipment(UE). Asynchronous transfer mode (ATM) is chosen as the transport technology to interconnect the UTRAN elements such as Node Bs and RNCs.

Two sets of data have been collected from two radio network subsystems(RNSs) in the UTRAN network system. Table 1 gives a rough description of the two data sets. We see that both RNSs cover over 550 radio cells and the data are sampled at 15 minutes interval. While the data from RNS-A contains 556 attributes, there are 731 attributes in the RNS-B data set. They contain comprehensive information about the system behavior such as load, capacity, resource availability and accesses, and service quality. Figure 5 plots some of those attribute curves. With this large amount of data, it is hard for operators to check each individual attribute and their correlations manually. In the following we extract both the local and global invariants in each RNS and use those invariants to monitor the status of RNSs.

Table 1. Raw Data Description

	RNS-A data	RNS-B data
number of covered cells	> 550	> 550
number of attributes	556	731
sampling rate	15 min	15 min
starting date	6/16/2006	3/7/2006
data length	15 days	5 days
failure existence	no failure	1 failure at 5th day

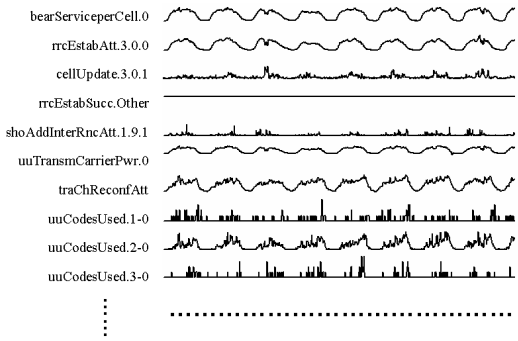


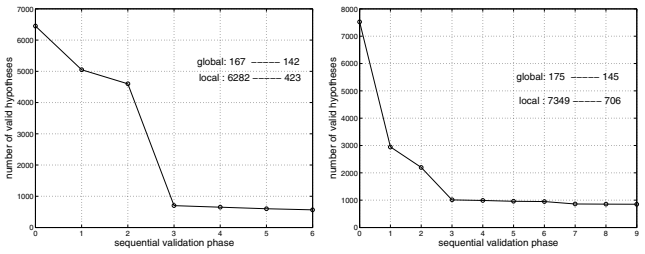
Figure 5. Some attribute curves in the data from RNS-A.

6.1 Invariants Extraction

We divide each data set into three parts for training, validation and testing respectively. For the data from RNS-A, we use the first 5 days measurements as training data, the next 6 days data as the validation set, and the last 4 days as the test set. For the data from RNS-B, both training and validation sets contain 2 days measurements and the test data contains 1 day measurements.

Figure 6(a) presents the results of invariants extraction in RNS-A. After initial training, we have extracted 6449 invariant hypotheses from the data, in which 6282 candidates are learned by the local method and the remaining 167 are from the global method. The 6 days validation data is then divided into 6 validation phases, with each phase containing 1 day measurements. In each validation window, we calculate the confidence score for every invariant hypothesis and discard those hypotheses with confidence lower than 85. The curve in Figure 6 exhibits the number of remaining invariant hypotheses after each sequential validation phase. We see that although the number of valid hypotheses shows a rapid decrease at the first three validation windows, it changes a little during the last three validation days. This tells us that the remaining candidates after validation are stable under different system variations and can be regarded as invariants of the system. Eventually we discover 565 invariant models, in which 423 are local invariants and 142 are global ones. Compared with the initial number of invariant hypotheses, we see that a large percent of local hypotheses disappear after the validation due to their sensitivity to system dynamics. This is because many local relationships identified from the training data only reveal partial dependencies between attributes, which can not always hold under different system variations. On the other hand, the global models are not likely to be affected by various

system dynamics. Most of the global hypotheses continue to hold after the validation process.



(a) (b)

Figure 6. The invariants sequential validation results in (a) RNS-A and (b) RNS-B.

The invariants extraction in RNS-B presents similar results, which are shown in Figure 6(b). After training with 2 days data, we obtain 7524 invariant hypotheses including 7349 local models and 175 global candidates. We then divide the validation data into 9 windows to sequentially validate those hypotheses. It shows that the number of valid hypotheses becomes almost stable after the third validation phase. Eventually 851 invariants have been discovered in which 706 invariants are local ones and the remaining are global invariants. One interesting observation is that the percentages of survived local and global invariant hypotheses are very close to those from the RNS-A data. It is probably because of the similar underlying physical properties of two systems even though they have different capacities and configurations.

6.2 Failure Detection

Since the learned invariants reflect the system internal property and are robust to normal system dynamics such as the workload variations, we can monitor the system runtime status by checking the consistencies of those invariants during system operation. That is, we use the test data as samples to test the invariant model. As shown in Figure 7, given a new measurement $\mathbf{z}_{new}$, we select relevant attributes from $\mathbf{z}_{new}$ to get $\boldsymbol{x}_{new}$ and y_{new} based on the invariant equation $y = f(\boldsymbol{x})$. The $\boldsymbol{x}_{new}$ is then fed to the invariant model to generate the estimated output $\hat{y}_{new}$. We calculate the difference between $\hat{y}_{new}$ and the real value y_{new} as the residual

$$R = y_{new} - \hat{y}_{new} \ . \tag{32}$$

We set a threshold for the residual R to determine whether the invariant model is broken or not. The threshold value is set as 1.2 times the maximum value of historical residuals collected from the validation phase. We count the number of broken invariants for each coming measurement as the evidence of system's health. Only when that number is larger than certain threshold, which is the 10 percent of total invariants in our experiment, we regard that the system has encountered an unexpected failure.

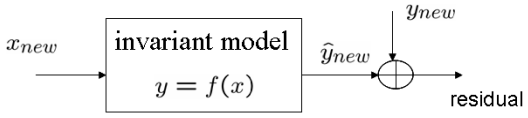
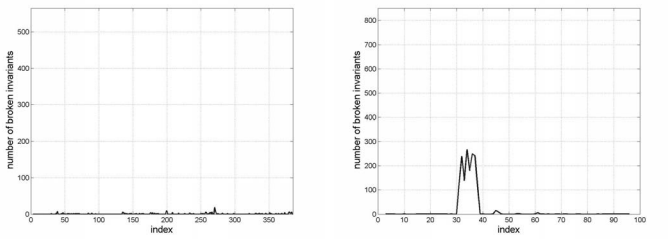


Figure 7. Residual generation for each new measurement based on the invariant model.

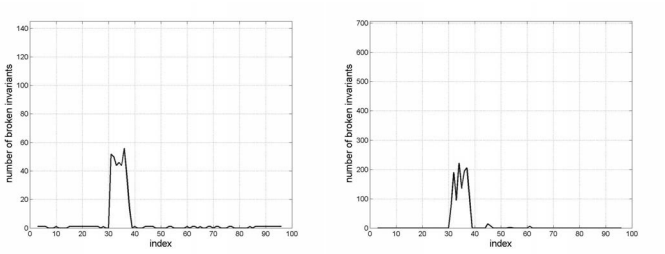
Figure 8(a) presents the number of broken invariants for each test sample from RNS-A, in which the X axis is the index of test samples and the Y axis denotes the number of broken invariants. The maximum of Y axis is the total number of discovered invariants in RNS-A. From that Figure, we see that only a small portion of invariants (less than 5 percent) are occasionally broken for the test samples. Therefore, we conclude that the system works well during the testing period, which is actually consistent with the groundtruth we got from the original data source.



(a) (b)

Figure 8. The number of broken invariants on the test data from (a)RNS-A (b)RNS-B.

We use the same strategy to monitor the status of RNS-B on one day test samples. In this case, we are told that the test data of RNS-B contains a failure which happened at 7:30am and was resolved two hours later. The results of our detector for that test data are presented in Figure 8(b). It shows that the number of broken invariants suddenly increases to over 200 at the 30th test sample, which is exactly the time when the actual failure occurred given that each measurement is sampled at 15 minutes interval. Furthermore the number of broken invariants immediately drops at the 38th test sample, two hours after the detected failure which coincides with the actual time of failure settlement. From this example we see that the set of discovered invariants does provide a real time view of the system internal state. We also plot the number of broken global and local invariants in Figure 9(a) and (b) respectively. It shows that both global and local invariants present strong evidences about the incidence of failure. Combining the global and local invariants can present more comprehensive description of the system behavior.



(a) (b)

Figure 9. The number of (a) broken global invariants and (b) broken local invariants on the test data from RNS-B.

After a failure has been detected, we look into the broken invariants and their related attributes to provide information about the failure source. We build a dependency matrix and use the market data analysis [10] to determine a list of attributes that are likely to co-occur with the failure. An example is shown in Figure 10 to illustrate such process. There are five attributes $m_1, m_2, \cdots, m_5$ in the figure, and six invariants have been discovered among those attributes which are plotted as lines connecting the related attributes. After the failure, four invariants are broken. To find the most suspicious attributes, we build a dependency matrix in Table 2. Each row in that matrix presents the information of one invariant: the column titled 'broken' represents whether the invariant is broken ('1') or not ('0'), and the other five columns show whether each of the five attributes is involved in the invariant ('1') or not ('0'). We define the values in the 'broken' column as a vector T_b and those in other columns as $T_i, i = 1, \cdots, 5$. The Jaccard coefficient is applied to indicate the relevance of each attribute to the failure

$$J_i = \frac{|T_i \cap T_b|}{|T_i \cup T_b|} \quad i = 1, \cdots, 5 \tag{33}$$

which is based on the number of 1s in the intersection set divided by the number of 1s in the union set between T_b and T_i. The attributes with higher Jaccard coefficient are considered more suspicious with respect to the failure. In the example of Figure 10, the Jaccard coefficients for the five attributes are 0.2, 0.2, 1, 0.2, and 0.2 respectively. Obviously m_3 is the most suspicious attribute.

Table 2. Data Dependency Matrix

invariant	broken	m_1	m_2	m_3	m_4	m_5
1	0	1	1	0	0	0
2	1	0	1	1	0	0
3	1	1	0	1	0	0
4	1	0	0	1	1	0
5	1	0	0	1	0	1
6	0	0	0	0	1	1

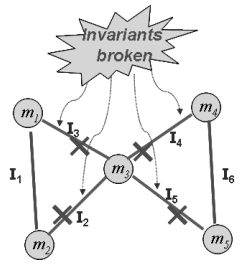


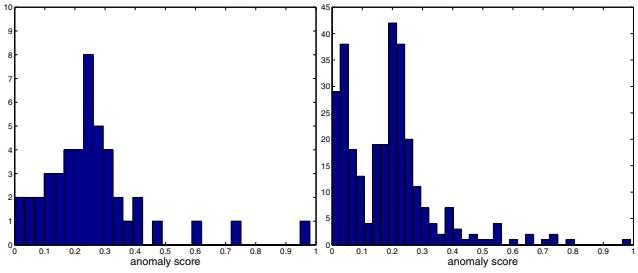
Figure 10. Illustration of broken invariants after the component m_3 breaks down.

We then calculate the Jacarrd coefficient for each attribute after we detected the RNS-B failure shown in Figure 9. Based on the system operator, such failure is caused by the broken DHT component in the system. In our diagnosis results, the DHT related attribute is ranked as the third highest among all system attributes. Note the attribute with the highest Jaccard coefficient may not be the exact failure root cause because of the cascading effects of failures. Our strategy is to provide a list of highly suspicious attributes to the system operator to help them for the failure localization. Although the final root cause is determined by the system operator, our ranked attributes can still provide useful clues to guide the localization.

6.3 More Real Data Testing

Besides the previous two data sets, we have tested our invariant based failure detection technique by using more than one year's system operation data (ranging from 12/2004 through 4/2006) from nine radio network systems(RNSs). Along with the data, a trouble ticket file is provided by the system operator in which 337 different system failures have been recorded. The logged failures belong to either of two categories based on their confidence level. The first category, which includes 46 cases, is related to the failures with strong confidence. Each failure in that category has a clear problem description such as DHT failure, partial power outage and so on. The second failure category contains the remaining 291 cases, which do not have clear problem descriptions and may be just false alarms. According to the original source, only 30 percent of the second category failures are real ones and the others are false alarms.

We first extract and validate the invariants for each RNS based on their normal data. The test data for each failure case is retrieved based on the failure equipment ID, problem start time, and the resolved time recorded in the trouble ticket file. We use the percentage of broken invariants as the anomaly score to determine whether there exists a failure or not. Figure 11 plots the histogram of the anomaly score for the two categories of failures respectively. We see that the anomaly score for the first category of failures shows a



(a) (b)

Figure 11. Histogram of the anomaly score for (a) the first category and (b) the second category of failures.

uni-modal distribution with the peak at around 0.25. On the other hand the histogram for the second category of failures shows a bi-modal distribution. This is consistent with our original knowledge because the second category contains a large amount of false alarms. If we choose the anomaly score 0.1 as the threshold, around 84.8 percent of recorded cases in the first category are reported as failures, and 33.6 percent of cases in the second category are regarded as failures. Here we assume the first category failures are all true failures. Therefore our technique can achieve the detection rate 0.848. For the second failure category, however, we do not have any groundtruth for the recorded cases. If we assume 30 percent of recorded cases are true failures, as described by the field operators, we can achieve the false positive rate 0.336 - 0.3*0.848 = 0.08. Compared with the current detection tool used by the system operators, our detection technique can reduce the false alarms significantly while still keep the satisfactory detection rate. Due to the confidentiality of the test data, we do not plan to describe the testing results in more detail.

7 Conclusions and Future Work

This paper has presented an invariants based framework to model complex information systems. In addition to the brute force local invariant search, we have proposed a statistical mining based algorithm to extract global invariants that represent multivariate dependencies among attributes. Combining local and global invariants can provide a more comprehensive and effective characterization of the system behavior, and hence facilitate many system management tasks. Experimental results from a real wireless system have demonstrated that the discovered system invariants can be used to effectively detect and localize failures in the system.

Since the discovered invariants capture the essentials of system state, the set of broken invariants under failure can serve as a 'signature' of system anomaly condition. One direction of our future work is to associate each signature with its related problem and solution, and build a database to store such information. By defining an appropriate metric to measure the similarity between different signatures, we can quickly identify and solve recurrent system failures by retrieving similar signatures in the database.

References

[1] G. A. Alvarez, E. Borowsky, S. Go, T. H. Romer, R. Becker-Szendy, R. Golding, A. Merchant, M. Spasojevic, A. Veitch, and J. Wilkes. Minerva: An automated resource provisioning tool for large-scale storage systems. *ACM Transactions on Computer Systems*, 19(4):483–518, 2001.

[2] P. Barham, R. Isaacs, R. Mortier, and D. Narayanan. Magpie: real-time modelling and performance-aware systems. In *9th Workshop on Hot Topics in Operating Systems (HotOS IX)*, May 2003.

[3] P. Bodik, G. Friedman, L. Biewald, and et al. Combining visualization and statistical analysis to improve operator confidence and efficiency for failure detection and localization. In *Proceedings of the Second International Conference on Autonomic Computing (ICAC 2005)*, pages 89–100, Seattle, WA, June 2005.

[4] H. Chen, G. Jiang, C. Ungureanu, and K. Yoshihira. Failure detection and localization in component based systems by online tracking. In *the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 750–755, Chicago, IL, 2005.

[5] M. Chen, E. Kiciman, E. Fratkin, A. Fox, and E. Brewer. Pinpoint: Problem determination in large, dynamic systems. In *2002 International Performance and Dependability Symposium*, Washington, DC, June 2002.

[6] I. Cohen, S. Jeffrey, M. Goldszmidt, T. Kelly, and J. Symons. Correlating instrumentation data to system states: A building block for automated diagnosis and control. In *6th Symposium on Operating Systems Design and Implementation (OSDI04)*, San Francisco, CA, December 2004.

[7] H.P. Corporation. H.P. Openview. *http://www.openview.hp.com/*.

[8] R. Doyle, J. Chase, O. Asad, W. Jin, and A. Vahdat. Modelbased resource provisioning in a web service utility. In *Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems (USITS '03)*, 2003.

[9] M. Figueiredo and A.K. Jain. Bayesian learning of sparse classifiers. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, volume 1, pages 35–41, Kauai, Hawaii, 2001.

[10] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.

[11] IBM. Tivoli business system manager. *http://www.tivoli.com/*.

[12] G. Jiang, H. Chen, and K. Yoshihira. Discovering likely invariants of distributed transaction systems for autonomic system management. In *Proceedings of the 3rd International Conference on Autonomic Computing (ICAC 2006)*, pages 199–208, Dublin, Ireland, June 2006.

[13] L. Ljung. *System Identification - Theory for The User*. Prentice-Hall, second edition, 1998.