Policy transformation techniques in policy-based systems management

Conference Paper · July 2004

DOI: 10.1109/POLICY.2004.1309146 · Source: IEEE Xplore

3 authors:

Mandis Beigi
IBM
31 PUBLICATIONS **737** CITATIONS
SEE PROFILE

Seraphin B. Calo
IBM
147 PUBLICATIONS **2,067** CITATIONS
SEE PROFILE

Darshika Verma
University of Petroleum & Energy Studies
100 PUBLICATIONS **1,934** CITATIONS
SEE PROFILE

Policy Transformation Techniques in Policy-based Systems Management

Mandis S. Beigi Seraphin Calo Dinesh Verma
IBM T.J. Watson Research Center
mandis@us.ibm.com *scalo@us.ibm.com* *dverma@us.ibm.com*

Abstract

Policy based systems management provides a means for administrators, end-users and application developers to manage and dynamically change the behavior of computing systems. One advantage of policy-based management is that it simplifies and automates the administration of IT environments. A significant part of the simplification is obtained by allowing the system administrator to specify only the objectives or goals that are to be met, rather than having to specify detailed configuration parameters for each of the different devices in the system. It may not be obvious to the administrator how the goals can be achieved without having to know the internals of the system. This knowledge thus needs to be captured in the policy driven actions. The existing algorithms for mapping policy objectives to specific configuration details tend to be specific to each policy discipline. This makes the policy-based approach harder to deploy for new disciplines. In this paper, we address different types of policy transformations and propose methods, which are not discipline specific for mapping objectives to system configurations.

1. Introduction

Most distributed computing environments today are very complex and time consuming for a human administrator to manage. The policy based management approach [1][16][18][19] provides a mechanism that can simplify the management of distributed computing systems, and can be applied in large computer networks [2][3], or in the context of Grid Computing [4]. This simplification of management functions is obtained primarily because of two aspects of the policy-based architecture employed, namely centralization and business level abstractions. Centralization refers to the process of defining all the device provisioning and configuration policies at a single point rather than separately provisioning and configuring each individual device in the distributed system. Business level abstractions make the job of the policy administrator simpler by allowing the policies to be defined in terms of a language that is closer to the business needs of an organization rather than in terms of the specific technology needed to implement them. Therefore, the administrator need not know all the details of the technology and 'how' these business needs can be met.

The mechanisms necessary for transforming service objectives into configuration parameters currently available in the literature are specific to the domains in which policy-based management is being applied [5][6][15]. The ad-hoc domain specific transformation mechanisms only increase the amount of work that is required to apply policy-based techniques to a new domain. Clearly, generic transformation mechanisms, which can be applied to many different disciplines, are needed. In this paper, we introduce the different types of policy transformations needed in a system managed by the use of policies, propose methods for performing such generic transformation mechanism, and study their effectiveness.

The paper is organized as follows: Section 2 describes a general policy based systems management architecture as employed in many environments; in Section 3 we show how this can be extended to incorporate policy transformation; in Section 4 we describe a number of different methods for determining the right set of system configuration parameters to meet a business objective or goal; Section 5 presents three solution scenarios for policy transformation; Section 6 describes experimental results for a policy transformation solution using case based reasoning; in Section 7 online or real time transformation is discussed; and, finally, conclusions and future work are presented in Section 8.

2. Policy based systems management architecture

A policy based systems management architecture based on the IETF framework consists of four main components: a policy management tool, a policy repository, a policy decision point, and a policy enforcement point. Figure 1 shows a general architecture for such a policy based management system [18]. Most current systems employing policies follow such an architecture.

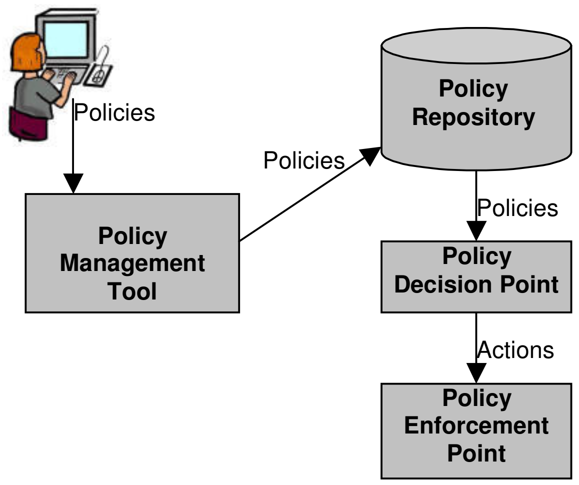

Figure 1. A general policy based systems management architecture

The system administrator enters the policies into the policy management tool. These policies are then sent to and are stored in the policy repository. The policies stored in the repository must be in a form that corresponds to the information model specified by the Policy Framework Working Group [1]. This ensures interoperability across products from different vendors. The decision points (PDPs) retrieve their policies from the repository. The PDP is responsible for interpreting the policies stored in the repository and communicating them to the policy enforcement points (PEPs).

The PEP is the system component that actually applies and executes the policies. The PEP and the PDP may both be located on a single device or they can be on different physical devices. Different protocols can be used for various parts of the architecture; e.g., the COPS protocol [7] or the SNMP protocol [8] can be used for communication between the PDP and the PEP. A repository could be a network directory server accessed using the LDAP protocol [9].

3. The policy transformation module

In general, high-level business-oriented policies will need to be transformed into lower level technology-oriented policies in order for them to be used by the various components of the system. Such a policy transformation can be done by a module within the policy management tool that takes the policies entered by the system administrator and converts them from one form to another before they are deployed and interpreted by the enforcement points. This transformation module should be bi-directional in that it should be able to transform technology-oriented policies (i.e. configuration parameters) into high-level policies or business goals. The reverse transformation feature is useful for an advisor tool where an administrator is told what level of service (i.e. set of goals) he or she can achieve given the specified set of configuration values.

Transformation may be done in one of two places: either before the policies are sent to the repository, or at the decision point before the policies are sent to the enforcement point. Policy transformation can be done either offline or online. The former will be referred to as static transformation, and the latter will be referred to as real time transformation.

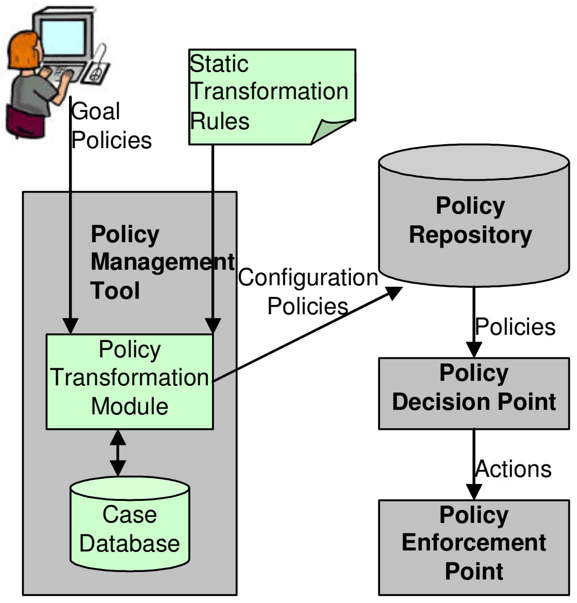

Figure 2. A policy based systems management architecture enabled with static policy transformation

Static (or offline) policy transformation uses static data and predetermined transformation rules to convert high level goals to sets of specific policies that can be applied to the components of the system. Figure 2 shows a policy-based systems management architecture incorporating static policy transformation within the policy management tool.

In real time policy transformation, the system uses an online component that dynamically monitors the behavior of prescribed elements within the system in order to ensure that the specified objectives are being met. In such a system, observations of system behavior are used by the transformation module to dynamically modify the configuration of the system to achieve the user's goals.

In the following sections of the paper, we describe different approaches for performing policy transformation and propose a generic algorithm that is discipline independent.

4. Existing Solution approaches

The problem of determining the right set of configuration parameters to meet a business objective or a goal can be solved by various approaches. We go over some of these approaches in the following subsections.

4.1. Analytic Models

If an analytic model exists that can be used to determine the business objective as a function of the configuration parameters, one could solve for the model parameters that satisfy the required business objective. As an example, if an analytical expression existed to relate a website's outbound bandwidth to its inbound traffic rate, one could invert the expression to obtain the requisite traffic rate for any desired limit on outbound bandwidth.

The problem with analytical expressions is that they do not exist in most real-life environments. In order to obtain a tractable analytical expression, one generally needs to make simplifying assumptions, which are rarely satisfied in practice. For example, in the case where one needs to model the relationship between the network bandwidth required and the connection rate supported, it is generally necessary to make assumptions such as: that the distribution of the arrival traffic is Poisson, or that the traffic characteristics are constant over time. Neither of these assumptions may actually hold, making it difficult to obtain tractable analytical expressions.

4.2. Online Adaptive Control

One could use concepts from control theory [10] to determine online schemes for controlling and fine-tuning the parameters needed to obtain a business objective. Another approach would be to develop a neural network model [11] as a mechanism for determining the impact of specific configuration parameters on the requisite business objective. The neural network or the adaptive control scheme could then be used to modify the business parameters dynamically online [12]. While these approaches seem attractive from the point of view of flexibility, there are two principal drawbacks. The first is that the control algorithm or the neural network needs to be trained on a characteristic workload for it to operate effectively, and it is often impossible to generate a synthetic workload that will have the same characteristics as the real workload in the system. The second drawback is that the development of the training model for a discipline is a laborious discipline-specific process, and it is hard to develop generic software that could be used readily across multiple disciplines.

4.3 Simulation Approach

An alternative to developing an analytic or a control model is to use a simulation system to model the behavior of a system, and use simulation based schemes to build a scheme for transforming higher level business objectives into lower level system configuration parameters. The simulation approach can offer a more flexible approach than an analytic model, and can encompass more realistic assumptions than the analytic model. Simulation based approaches do not suffer from the limitations of a training set in order to learn how to transform the system.

The significant limitation of the simulation approach is its lack of generality. Developing a good simulator for meaningful policy transformation requires a significant amount of effort, and depends upon many nuances of the domain and environment in which policies are being used. A simulation based approach can be used to build a customized policy transformation engine, but is hard to use in a generic manner.

5. Our Proposed Approach

We categorize the usage of policy transformation into three different scenarios. We go over these scenarios in the following subsections and propose a generic solution.

5.1. Scenario 1: Transformation using static rules

Transformation using static rules is the simplest type of transformation but can be very useful in simplifying the policy language as seen by the system administrator. In this scenario, we assume that there is set of static transformation rules for converting policies in terms of high-level goals into policies in terms of low-level configuration parameters understandable by the system. The rules follow a policy language that is more detailed and complicated than the one used by the goal policies as seen by the system administrator. These rules are defined by an expert user, who knows the details of the system and the definitions of the various objectives, such as what it means to provide gold level service in terms of performance, security, etc. The transformation module simply transforms the objectives to low-level configuration parameters using the definitions specified by the transformation rules. This kind of transformation is well suited for a policy management system in which there are groupings of administrators, and different administrators have different authorities and access rights for retrieving and editing policies.

As an example, the transformation module would receive a policy of the form: If the *user* is from *Schwab*, then provide *Gold* level service. This policy would be transformed into the following more specific policy: If the *user* is from the subnet *9.10.3.0/24*, then reserve a *bandwidth* of *20 Mbps* and provide an *encryption of 128 bits*. In this case, the transformation rule specifies that a Schwab user is on the 9.10.3.0/24 subnet. It also specifies that a Gold service be defined to provide a bandwidth of 20 Mbps and an encryption of 128 bits. Since policy transformation is bi-directional, a user is also able to query the module to find out the level of goals that can be achieved given a set of low-level configuration parameters.

5.2. Scenario 2: Transformation by policy table lookup

In this scenario, we assume the transformation module holds a table of policies appropriate for the system. The system administrator queries the module with a set of configuration parameters in order to obtain a set of goals that can be achieved given the input.

In order to perform transformation, each policy in the table needs to be first mapped into an N-dimensional hyperspace object, where N or the dimension of the space is the number of configuration parameters in the policy. Each policy is considered to be a region in the hyperspace, which may be connected or disconnected. Each hyper-cube points to a set of goals or actions. The system needs to match the hyper-cube corresponding to the policy being queried against all the hyper-cubes representing all the policies in the table. It finds the hyper-cube from the table that fully contains the specified hyper-cube.

Note that the specified hyper-cube might not be fully contained in any single hyper-cube from the table. In other words, it might overlap several hyper-cubes, in which case the incoming hyper-cube needs to be split into smaller hyper-cubes where each new hyper-cube now corresponds to a different set of goals. In order to make sure we find a match for all segments of the incoming hyper-cube, we need to perform a coverage check on the policy table.

The coverage checker is based on the ability to subtract two regions (i.e. perform group difference): A - B = {x | x in A but not in B} where A and B are two regions and A-B denotes the difference between A and B. Coverage checking is performed by subtracting from the region of interest all the regions defined by the group of policies. Once the region of interest becomes empty, then coverage is deemed *complete*. If, after iterating through all the policies the region of interest is not empty, then the coverage is deemed *incomplete.*

Figure 3 shows a simple 2-dimensional hyperspace with 4 policies shown as A, B, C, and D. The policy being queried is shown with a dashed pattern and is shown to overlap policies B, C and D.

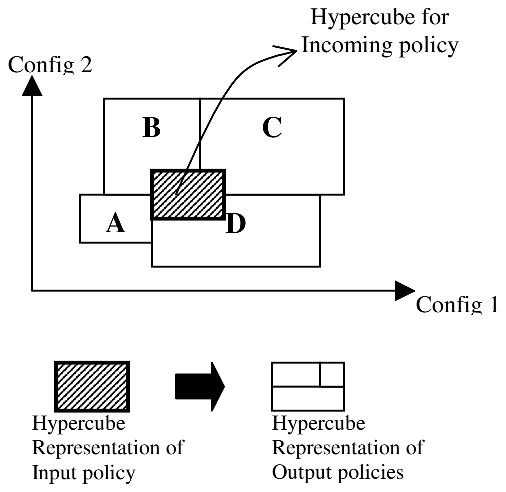

Figure 3. Hypercube representation of policies for a 2-dimensional space

5.3. Scenario 3: Transformation using case based reasoning

One can use a case database or history of system behavior to provide an experiential basis for the transformation of high-level policies or goals into low-level configuration parameters and vice versa. In this approach, the transformation module uses the knowledge learned from the behavior of the system in the past to predict its present and future behavior.

Case based reasoning is widely used in many applications such as diagnostics, planning, prediction, object classification, and in electronic commerce for such functions as product selection [13].

In the case based reasoning approach, the system learns experientially from the operational behavior it has seen in the past. The system maintains a database of past cases, where each case is a combination of the system configuration parameters and the business objectives that are achieved by the specific combination of the system configuration parameters. When the configuration parameters needed for a new business objective are required, the case database is consulted to find the closest matching case, or an interpolation is performed between the configuration parameters of a set of cases to determine the appropriate configuration parameters that will result in the desired business objective. The adaptation to each discipline requires that the discipline identify the configuration parameters and the business objectives that constitute a case and identify the strategies for interpolation among cases, but the same case manipulation software and algorithms can be used across all the different disciplines.

However, the effectiveness of the case based reasoning approach depends on having a rich enough set of cases that can be consulted in order to map the business objectives into the desired set of configuration parameters. In a system that has been running for an extended period of time, it is possible to build cases from prior experiences. However, at bootstrap time, the system has no case history to use. Thus, one has to solve the bootstrap problem by using some heuristic (an analytical expression or rules of thumb), or by pre-populating the system with a set of cases observed in pre-operational tests. The bootstrapping case suffers from the same limitations as the previously mentioned approaches described in section 4. Later on in this section, we will describe methods that can help increase the accuracy of the cased based approach when there is only a small amount of data available. Despite this bootstrapping issue, it is felt that the case based reasoning approach offers the best solution for a generic mapping of business objectives to the configuration parameters.

The format of the case database – The case database contains measurements of various parameters of a system over a long period of time. Each case contains an N-dimensional set of configuration parameters and an M-dimensional set of corresponding goal values. Each case corresponds to measurements taken at one point in time. The cases may or may not be ordered chronologically and may or may not be associated to timestamps. Table 1 shows an example of a case table in a 2-tiered web server measuring the user response times as a function of the number of disks and nodes in tiers 0 and 1. In this example N=4 and M=1.

N: # of configuration parameters

M: # of measured goal values

It is assumed that the table may contain more information than necessary, such as configuration parameters that have no correlation with any of the corresponding goal values, or goal values that have no correlation with any of the configuration parameters. If this is the case, there is a need for an automated management system that with little or no human interaction can deal with such ambiguities in the data. Ideally, the user of the system would not need to know which measurements are necessary and relevant.

We also assume that the database may contain inconsistent cases. Two cases are considered inconsistent when the same values of configurations result in different goal values. This may be the result of bad measurements or it may be caused by other unknown variables affecting the system behavior, which are not present in the case database. Conversely, in cases when different values of configuration parameters map to the same set of goals and objectives, one may combine these cases together to form a hyper-cube as described in section 5.2 in order to increase the performance of the table lookup.

It may also occur that some measurements might be missing in the table due to measurement problems. We will present later in this section a way of handling such difficulties by including data pre-processing steps that remove noise and redundant data.

Tier0 # of Disks	**Tier1 # of Disks**	**Tier0 # of Nodes**	**Tier1 # of Nodes**	**User Response Time**
1	4	1	2	0.039 sec
3	2	2	4	0.029 sec
2	3	2	4	0.082 sec
4	1	1	2	0.015 sec
2	1	3	3	0.042 sec
2	4	1	2	0.053 sec
1	2	2	4	0.032 sec
3	2	3	4	0.098 sec

Table 1. Sample of a case database of the performance of a 2-tiered web site

Due to the high sensitivity of the case based reasoning approach to noise in the data, we consider grouping the data values into clusters. Clustering also improves the performance of the system by decreasing the data lookup time.

K-nearest neighbor clustering – We use the k-nearest neighbor clustering technique to cluster the data points into a fixed number (k) of clusters. The clusters are initially created by taking k data points randomly from the database and assigning each point to a cluster. The rest of the data points are then subsequently all assigned to these clusters. There are two different assignment techniques. The first and easier method is to find the closest cluster to the new point (i.e., the one for which the distance between the new point and the mean of the cluster is smallest). The second method is more accurate, especially when the data has lots of large overlaps (i.e., variances are large). It assumes that each cluster has a Gaussian distribution, and so we find the most probable cluster for the new point using the Gaussian density function. Every time a point is assigned to a cluster, the mean of the cluster is recalculated. The points may be reassigned from one cluster to a closer cluster. The reassigning of the points continues until the means of the clusters remain the same or vary by a very small amount.

The clustering technique is more computationally intensive than the regular CBR technique but is more robust to noise in the data. It also can be used to effectively limit the search space to distinctly different cases, and thereby improve performance. In most systems, cases that are very close should be coalesced, since they do not capture any significantly different behavior. Figure 4 shows simple clustering in a 2-dimensional space.

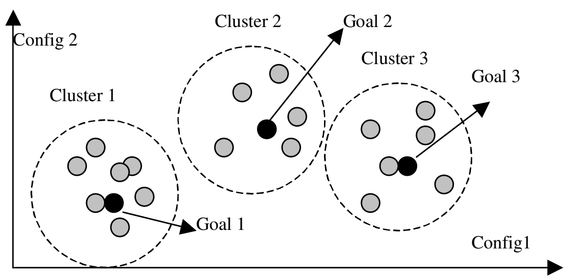

Figure 4. Clustering in a simple 2-dimensional space

Data pre-processing – In order to improve the usefulness of the case database, it is desirable to perform various pre-processing techniques on the data. This is needed to remove noise in the data as well as to remove irrelevant or redundant [20] data. The techniques that we are considering are discussed in the following.

Dimensionality reduction: A managed system may have hundreds of configuration knobs or *features*. Also, based on our assumption that the database may contain all the features, related or unrelated to any of the goal values, the dimensionality of the data may become too large thus increasing the computational overhead of the subsequent processing stages. A method for reducing the dimensionality will be of great use in our case based approach. In order to get accurate and meaningful results, we need many cases. The minimum number of cases needed depends on the number of dimensions [1]. Therefore, another way of getting accurate results, without having lots of data (i.e. many cases or instances) is to reduce the number of dimensions.

One form of dimensionality reduction technique is called feature selection. In this technique, all features are evaluated and weighted using some criteria and the features with the highest scores are selected. In our experiments, we have used accuracy as the evaluation criteria. First, we start with all the features in place and evaluate the accuracy of the clustering technique by comparing our predicted goal values with the actual values the simulator calculates for the same set of configuration values. We then take one feature out at a time and reevaluate the accuracy. This is called a backward feature generation. There are other generation schemes such as forward and bi-directional, which will not be discussed here. The search strategy we have used is a complete search, which means we

[1] The rule of thumb is to have 10 data points per dimension

cover all combinations of all the features. However, a complete search is time consuming and one can instead use heuristic or non-deterministic search strategies to improve the performance of the system.

Another method we have considered for reducing the dimensions is performing principal component analysis (PCA) [17] on the data. PCA is a powerful and well-established technique for finding the most informative or explanatory features in a collection of data[2]. The basic idea in PCA is to find those components, which explain the maximum amount of variance in a set of linearly transformed components. In particular, PCA substantially reduces the complexity of the data in which a large number of variables (e.g. thousands) are interrelated. PCA accomplishes this by computing a new, much smaller set of uncorrelated variables that best represent the original data.

Calculating the cross correlation matrix of the entire data allows us to select and be able to inform the user of the correlated parameters. A correlation matrix contains the strength as well as the direction of the relationship between all the variables. The correlation values are all in the range [-1, 1]. A positive number indicates a direct relationship between the variables, where as one variable goes up the other goes up as well. On the other hand, a negative number indicates an inverse relationship between the two variables. A number at or near zero indicates there is virtually no relationship at all between the variables. The cross correlation matrix is of size (M+N) x (M+N) where M is the number of configuration parameters and N is the number of goals.

Normalizing: Some configuration parameters may have values that are several orders of magnitude larger than others. For example a user response time is usually on the order of milliseconds while a bandwidth or throughput value is usually tens of Mega bits per second. Therefore we need to normalize the data for all the dimensions before we perform clustering in order to get accurate results.

Data unit consistencies: In order for the data to be meaningful, it needs to have consistent units of measures.

6. Experiments and Results

We have used the IBM High Volume Web Sites simulator [14] to produce a database containing a rich set of observations or cases. The simulator estimates the performance of complex configurations of a multi-tiered web site. Figure 5 shows a 3-tiered web site configuration. The simulator can accommodate different workload patterns such as online shopping, trading, reservations, auctions etc. It also allows changing of other aspects of the simulation such as varying the user session characteristics, as well as the software and hardware characteristics in each tier. In the experiments performed, we have modified 21 configuration parameters and observed 16 different output values or goals. Therefore, in this case, N=21 and M=16.

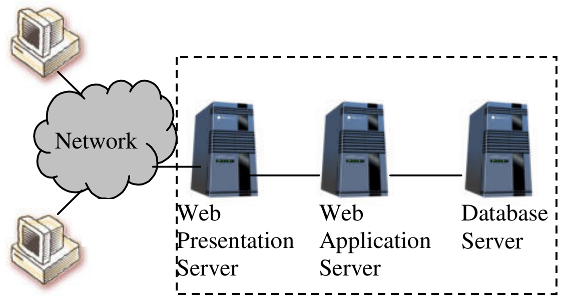

Figure 5. A 3-tier web sites architecture

Figure 6 shows the components of the transformation module. To generate each instance of data or a case, we generate a set of 21 uniformly distributed random variables. The configuration parameters of the simulator are set to these values and the resultant values or goals are measured. We have generated as many as 100,000 cases in this fashion and placed them in a repository (i.e., a case database). The case database is then passed to the pre-processing components, then to the feature reduction component, and finally into the clustering component of the transformation module.

[2] PCA is only effective for dimensions that have linear dependencies with each other

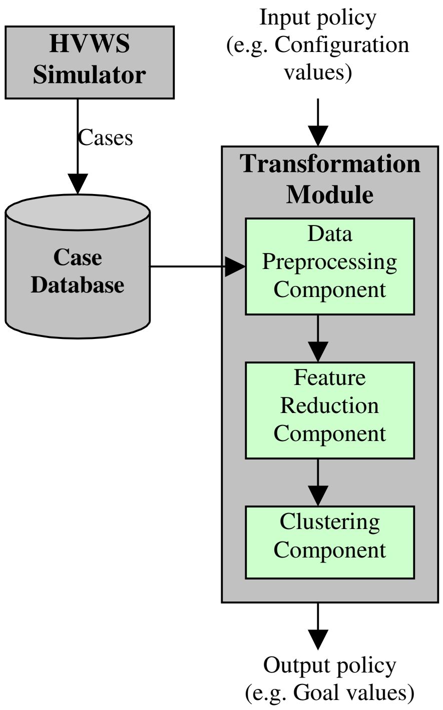
HVWS Simulator
Input policy (e.g. Configuration values)
Cases
Transformation Module
Case Database
Data Preprocessing Component
Feature Reduction Component
Clustering Component
Output policy (e.g. Goal values)

Figure 6. Components of the policy transformation module

Figure 7 shows a plot of the accuracy of the system using the clustering technique as a function of the data size or the number of instances learned by the system. The y-axis is the Euclidean distance of the predicted goal vector from the actual goal vector. As expected, the distance tends to decrease as the system collects more data or instances. As mentioned earlier, the input vector or the configuration vector has a dimension of 21 (e.g. N=21) and the goal vector has a dimension of 16 (e.g. M=16). The configuration values are normalized to be real values between 0 and 100. The number of clusters, k is set to be dataSize/100. We have used three different distance metrics: Euclidean distance, weighted Euclidean distance, and Mahalanobis distance [17]. All the three mentioned distance metrics resulted in the same accuracy levels.

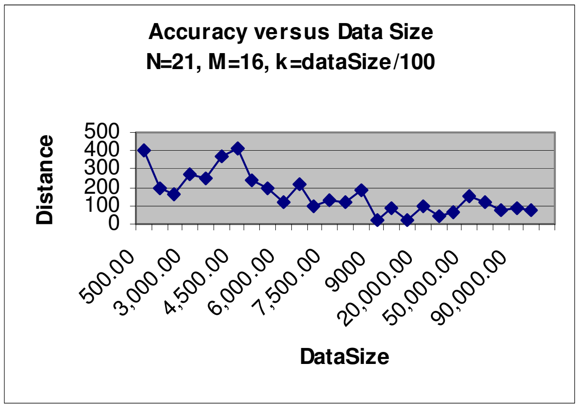

Figure 7. Accuracy of the system versus the size of the data used for learning (e.g. history of the system)

The feature reduction component using the PCA technique has been able to reduce M (e.g. the dimension of the measured goal values) from 16 to 6. This is quite substantial, and leads to a corresponding reduction in the complexity of the data analysis. This method can be used to reduce N as well since in a real environment the configuration parameters or knobs are correlated to each other as well. But in our simulations, since we use a uniform distribution to generate our random values for the configuration knobs, we are not able to show that.

We have calculated the cross correlation matrix for 10,000 instances of data. The correlation matrix shows a strong direct correlation value (e.g. +0.949) between the configuration knob 'ThinkTime' and the goal value 'SessionTime'. It also shows a direct high correlation value (e.g. +0.805) between the configuration knob 'BackgroundUtilization' and the goal value 'CPUUtilization'.

Table 2 shows the correlation values between various configuration and goal parameters. As it shows, the average response time of the web site has a higher dependency on the number of nodes in tier 1 than it does on the number in tiers 0 and 2. This shows that one can improve the web site's performance by adding more nodes to the application server tier rather than the database and the Web presentation server tiers.

	Tier0 #ofNodes	**Tier1 #ofNodes**	**Tier2 #ofNodes**
AvgRespTime	0.08	0.18	0.13

Table 2. Correlation values between various configuration (i.e. Tier0-Tier2 # of nodes) and a goal value (i.e. AvgRespTime)

In order to reduce the dimensionality one can throw away all goal parameters that show very little or no dependency on any of the configuration parameters.

7. Online or real time policy transformation

In Real time policy transformation an online monitoring component is used which dynamically monitors the behavior of its agents to ensure the objectives are being met. In such a system, the transformation module dynamically modifies the configuration of the system based on its observations in order to achieve the user's goals.

Figure 8 shows an adaptive policy-based systems management architecture, which enables transformation of business level objectives into system configuration parameters. The policy management tool allows the system administrator to specify business level objectives or goal level policies. These policies are transformed into system configuration parameters by the policy transformation module and then sent to the repository.

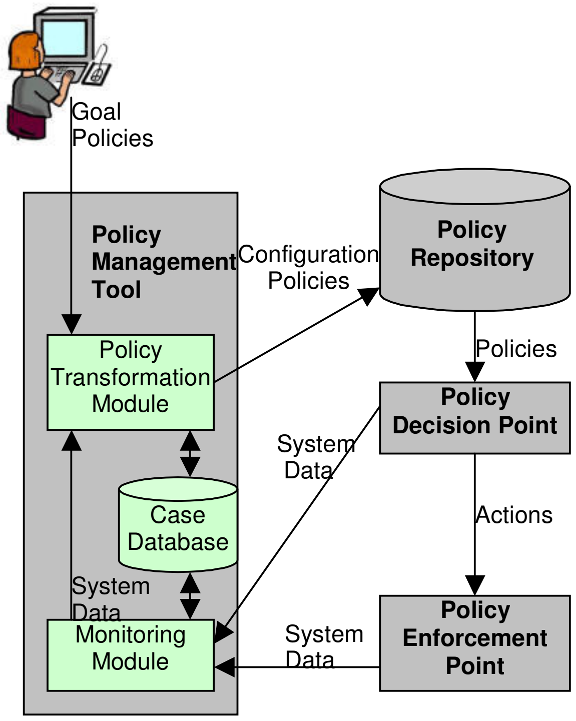

Figure 8. A policy based systems management architecture enabled with static policy transformation

The transformation module interacts with a monitoring module, which is responsible for monitoring the current behavior of the system and determining whether the specified business objectives are being met. The transformation module obtains the current system configuration from the repository, and changes it if needed to a new system configuration, which is better suited for meeting the performance goals. The transformation module can also send the new modified system configuration to the policy repository.

This type of transformation is useful for handling those disciplines where the mapping from business level objectives to system configuration is dependent on the current state of the system. As an example, mapping the response time objectives to configuration parameters like network bandwidth limits is dependent on the current traffic load on the system, and an initial configuration tool can only provide estimates. Therefore, an adaptive architecture is much more appropriate for such a task. Depending on the nature of the discipline, an adaptation component may communicate changed configuration parameters directly to the PEPs, or may be incorporated as part of the management tool running as a system daemon process.

The monitoring module is responsible for monitoring the system and obtaining its current measure of the business objective that is supported, as well as reporting the current set of configuration parameters that are effective within the system. The information captured by the monitoring module is averaged over several measurement intervals to produce a new case history based on the averages observed during that period. The cases thus observed are stored in the case database.

8. Conclusions and future work

In this paper, we have addressed different types of policy transformations and proposed discipline independent methods for mapping objectives to system configurations and vice versa. We have described different scenarios for using a transformation module in a policy-based management system. A transformation module may use static data or it may be used real time in an adaptive system that uses a monitoring component for capturing real time data. We have used a high volume web site simulator to produce a database containing a rich set of observations for testing the case based reasoning approach. The proposed case-based reasoning approach seems to be practical for performing offline as well as real time transformation.

Our future work in this area is to explore a system modeling approach that predicts the impact of the system configurations on the goal values. One way to do this would be to come up with a state transition model for the system.

9. References

[1] The IETF Policy Framework Working Group: Charter available at the URL: http://www.ietf.org/html.charters/policy-charter.html.

[2] L. Lymberopoulos, E. Lupu and M. Sloman, "An adaptive Policy based Management Framework for Differentiated Services Network", IEEE International Policy Workshop 2002, Monterey, CA, June 2002.

[3] A. Ponnappan, L. Yang, R. Pillai, and P. Braun, "A Policy Based QoS Management System for the IntServ/DiffServ Based Internet", Policy Workshop 2002, Monterey, CA, June 2002.

[4] D. Verma, S. Sahu, S. calo, M. Beigi, and I. chang, "A Policy Service for GRID Computing", Grid Workshop 2002, Baltimore, MD, Oct. 2002.

[5] M. Bearden, S. Garg, and W. Lee, "Integrating Goal Specification in Policy-Based Management", IEEE International Workshop on Policies for Distributed Systems and Networks, Bristol, U.K., January 2001, pp.153-170.

[6] D. Verma, "Simplifying Network Administration using Policy based Management", IEEE Network Magazine, March 2002.

[7] The IETF Resource Allocation Protocol Working Group. Charter available at the URL http://www.ietf.org/html.charters/rap-charter.html.

[8] W. Stallings, "*SNMP, SNMPv2, SNMPv3, and RMON 1 and 2",* Addison Wesley, ISBN 0201485346, 1999.

[9] T. Howes, M. Smith, and G. Good, "Understanding and Deploying LDAP Directory Services", MTP, ISBN 1578700701, 1999.

[10] W. Brogran, "Modern Control Theory", Prentice Hall, October 1990.

[11] S. Haykin, "Neural networks: A Comprehensive Foundation", Prentice Hall, July 1998.

[12] Y. Diao, N. Gandhi, J. Hellerstein, S. Parekh and D. Tilbury, "Using MIMO Feedback Control to enforce Policies for interrelated metrics with application to the Apache Web Server", Proceedings of NOMS, April 2002.

[13] http://www.cbr-web.org/

[14] http://www.ibm.com/websphere/developer/zones/hvws

[15] D. Verma, M. Beigi, and R. Jennings, "Policy Based SLA Management in Enterprise Networks", IEEE International Workshop on Policies for Distributed Systems and Networks, Bristol, U.K., January 2001, pp.137-152.

[16] http://www-dse.doc.ic.ac.uk/research/policies

[17] R. Duda, P. Hart, and G. Stork, "Pattern Classification", John Wiley & Sons, Second edition, 2001.

[18] RFC 3060: Policy Core Information Model -- Version 1 Specification

[19] RFC 3198: Terminology for Policy-Based Management

[20] H. Liu and H. Motoda, "Feature selection for knowledge discovery and data mining", Kluwer Academic Publishers c1998, ISBN: 079238198X.