International Journal of Computer and Communication Technology

Volume 7 | Issue 4 Article 7

October 2016

SELF-MANAGING PERFORMANCE IN APPLICATION SERVERS – MODELLING AND DATA ARCHITECTURE

RAVI KUMAR G
HP Bangalore, Research Scholar JNTUH, Hyderabad, India,, ravikgullapalli@gmail.com

C. MUTHUSAMY
Yahoo, Bangalore, India, chelgeetha@yahoo.com

A.VINAYA BABU
College of Engineering, JNTUH, Hyderabad, India, dravinayababu@yahaoo.com

Recommended Citation
G, RAVI KUMAR; MUTHUSAMY, C.; and BABU, A.VINAYA (2016) "SELF-MANAGING PERFORMANCE IN APPLICATION SERVERS – MODELLING AND DATA ARCHITECTURE," *International Journal of Computer and Communication Technology*: Vol. 7 : Iss. 4 , Article 7.
DOI: 10.47893/IJCCT.2016.1379
Available at: https://www.interscience.in/ijcct/vol7/iss4/7

SELF-MANAGING PERFORMANCE IN APPLICATION SERVERS – MODELLING AND DATA ARCHITECTURE

RAVI KUMAR G[1], C.MUTHUSAMY[2] & A.VINAYA BABU[3]

[1]HP Bangalore, Research Scholar JNTUH, Hyderabad, India, [2]Yahoo, Bangalore, India
[3]College of Engineering, JNTUH, Hyderabad, India
E-mail : ravikgullapalli@gmail.com[1], chelgeetha@yahoo.com[2], dravinayababu@yahaoo.com[3]

Abstract - High performance is always a desired objective in computing systems. Managing performance through manual intervention is a well-known and obvious mechanism. The attempts to self-manage performance with minimal human intervention are predominant in the recent advances of research. Control Systems theory is playing a significant role in building such intelligent and autonomic computing systems. We are investigating in building Intelligent Application Servers by enabling control system as a first class feature in component software, at design time and runtime. In this direction, it is important to build efficient data access mechanisms that capture the control system models, performance data, analyze the data efficiently, identify patterns and build a knowledge base. In this paper we propose a data organization and architecture as a building block of developing Intelligent Application Servers.

Keywords - *self-managing performance; Intelligent Application Servers; Control Systems; modeling and data architecture;*

I. INTRODUCTION

High performance is most desired non-functional requirement of any computing system. The rapidly increasing usage of IT resources triggered self-managing and self-correcting IT architectures from hardware to the application layer. Such self-managing and self-correcting solutions are referred as Autonomic Computing. There are various solutions in providing such autonomic and self-managing mechanisms in the recent research investigations [1]. Control Systems is obvious choice due to the inherent feedback and adaptive control capabilities to build such autonomic IT systems [2], supported by mathematical foundations. Control Systems proved successful applications in networking, database systems, and data centers. There is a great emphasis of control systems application in application layer involving Application and Web Servers [3]. Most of the research is around using classic controllers, building hybrid controllers for automated performance management. In this paper we intend to exploit generic modeling and data architecture providing simple ways to model, capture, organize and analyze the real time data that help in building more robust self-managing abilities in Application Servers.

II. PROBLEM AND RELATED WORK

Application Layers play a significant role in today's IT environments. They host applications and provide services ranging from individual consumers to the Enterprise users. It is obvious that such Application Servers exhibit high performance all the times. There are solutions applying control systems to self-manage the performance using adaptive controllers. Many of them are specific solutions to manage the performance of web servers [4][5], web services [6], web caching [7][8], EJB Servers [9], JMS Servers [10], JDBC drivers [11][12]. Most of such solutions are specific to the components in the Enterprise Application Server stack. Although the controllers, control algorithms are important in building self-managing computing systems, it is even more necessary to provide simple, faster, efficient data organization mechanisms that support generic data modeling, data analysis, and pattern identification mechanisms and construct knowledge base. In this paper we address the above said issues by a simple Data Architecture for efficient functioning of these controllers. The proposed Data Architecture is a building block of a bigger research problem where we are investigating to build an Expert Control System for Application Servers [13][14].

III. INTELLIGENT CONTROL ARCHITECTURE – OVERVIEW

The Fig 1 below shows Intelligent Control Architecture consisting of Data Architecture block in the outer loop and a Feedback block in the inner loop. The Feedback block consists of a set of classic controllers such as P, PI, PID [15] and advanced controllers such as Time-Series, Fuzzy and other intelligent controllers. The Data Architecture block contains the functional blocks to capture the performance data of the Application Server component control system models, analyze data, identify patterns and create knowledge base. The subsequent sections explain this block in detail and the data that is exchanged between these blocks.

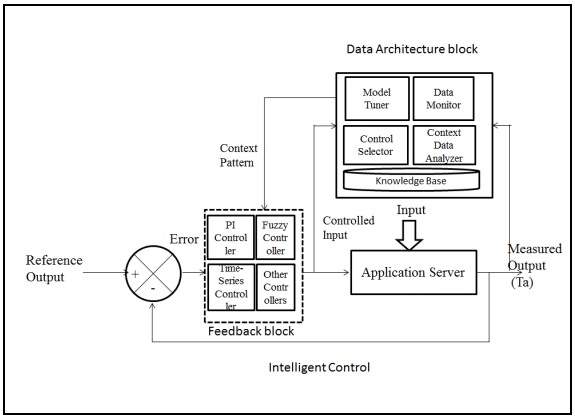

Fig. 1: Intelligent Control Architecture

At every sample interval the control loops shown are run, where the data architecture block analyzes the real time data, sends a "Context Pattern" to the Feedback block. The context pattern object contains the controller to be used and other information required for tuning the controlled input to the Application Server to achieve the desired performance.

IV. MODEL TEMPLATES – UML DESIGN

Control System solutions require the Application Server components models to tune their performance. The typical models adopted are ARMA models [16]. Equation (1) below shows a simple SISO system model.

$$y(t + 1) = ay(t) + bu(t) \qquad (1)$$

A. Model Templates

The feedback control system uses ARMA models and we have considered the same to represent the Application Server components in our solution.

The models of different JEE server [17] components such as JMS Providers [18], EJB Servers [19] , Web Servers are captured as SISO or MISO model [20] templates consisting of all possible output parameters for the server components. These are pre-defined models used to capture the associated performance data in the faster access database.

The pre-built model templates are separately packaged as XML resource files by the server component developers along with the compiled source. These templates are used at the Application Server initialization to create the database and estimate the different model parameters at runtime. The Fig 2 below shows the definition of a model template

```
<model>
 <component-name>
  <outputParam> </outputParam>
  <inputParam1> </inputParam1>
  <inputParam2> </inputParam2>
  <inputParam3> </inputParam3>
  <inputParam4> </inputParam4>
</component-name>
</model>
```

Fig. 2 : Sample MISO Model Template in xml

B. Example Models – JMS Providers

The following Fig 3 shows the xml definition of representing the ARMA model of JMS Provider.

```
<model>
 <jms>
  <outputParam>
     msgTPut
  </outputParam>
  <inputParam1>noSub</inputParam1>
</jms>
</model>
```

Fig. 3 : JMS Provider SISO Model Template.

.

The above shown model template can be represented by the following (2) as SISO model

$$\text{msgTput}\,(t + 1) = a\ \text{msgTput}(t) + b\ \text{noSub}(t) \qquad (2)$$

'*a*' and '*b*' are the model parameters. This model will be transformed into a column oriented database during the initialization of the Application Servers.

V. DATA ARCHITECTURE

This section explains the Data Architecture block of the Intelligent Control Architecture in the Fig 1. Effective data organization is important for analyzing the performance data; infer meaningful patterns to enable tuning the managed system. We propose to use a Column oriented database to capture performance data and will be periodically moved to a persistent storage when its patterns are defined as rules in the knowledge base. The Fig 4 below shows the proposed Data Architecture with two parts in it

- Data Organization: It deals with the parsing the model templates and capturing the data. The Data Modeler deals with transforming model templates of the Application Server components into the column oriented database. The Data Monitor updates the performance data.
- Data Analysis: It has a generic data access layer to fetch the performance data, which will be consumed by the Data Analyzer to analyze the data, identify patterns and generate Context Patterns.

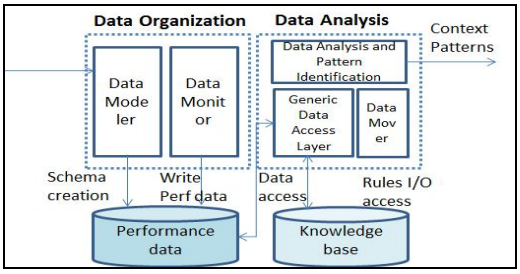

Fig. 4 : Data Architecture

A. Context Pattern

Context Pattern is an object that contains the predicted values for the future output parameter values, that reflects the behavior of the components of the Application Server. Additional parameters are also predicted such as percentage of CPU usage, Memory usage. Additionally for such a predicted behavior the suitable controller required to be applied is also set. The Fig 5 below shows a sample Context Pattern.

```
<ContextPattern>
  <Pattern>SuddenVariations</Pattern>
  <OutParam>MessageThroughput</OutParam>
  <OutParamVal>50</OutParamVal>
  <InParam>Smax</InParam>
  <InParamVal>75</InParamVal>
  <ControllerType>Fuzzy</ControllerType>
  <CPU>75</CPU>
  <MEM>68</MEM>
</ContextPattern>
```

Fig. 5 : Context Pattern

The context pattern either determines to choose an appropriate controller or it suggests an action to be taken to cater to the future needs. The Table I below shows the action to be taken based on the context pattern generated. The mapping between the Context Pattern values and the Controller to be chosen or the action to be taken is stored as control selection rules in the Data Architecture block in Fig 1.

Table I : Context Patterns Controller Mapping

Sl. No	**Context Patterns**		
	Pattern Identified	***Controller Type***	***Action***
1.	Sudden Variations	Fuzzy	
2.	Quick Adaptation	PID	
3.	Gradual Increase	Time-Series	
4.	Constant	Time-Series	
5.	Resource shortage		New Server Instance
6.	Drastic Performance Degrdation		System Audit

B. Data Organisation – Performance Data

The model templates defined for different server components will have a lot of data associated during runtime. It is desired that there is a faster data query mechanism, and the growing data volume should not be an overhead on the self-managing architecture.

The first challenge of faster data access is addressed by choosing a column oriented database [21]. It is used to capture the data that provides data sequence for a given parameter which is a useful input for easier data analysis and identifying the patterns. The second challenge is handled by moving the data to a persistent storage periodically. The data analyzer generates context patterns periodically, generates rules from those context patterns, capture in the knowledge base. Such rules represent the patterns associated with the data and hence the data will be moved to a persistent storage. Thus the amount of data associated to study the past data; the mechanism to analyze and predict the future patterns is much simpler and light weight mechanism.

The following are the different data elements involved based on the model templates:

- Input parameters
- Output parameters
- Model Parameter Constants
- Controller Gains

The Column oriented database supports SQL [22] and fetches the data column wise rather than row wise thereby providing faster data access. Such a data structure enables the Pattern Analysis algorithms to easily extract the behavior. The following Table II below shows a sample Table structure of a column oriented database for a JMS Provider.

TABLE II : JMS PROVIDER COLUMN ORIENTED DATA

Message Throughput	***Subscribers***	***Publishers***	***brokers***	***CPU***

It is easy to add additional input parameters at runtime when it is observed that there are other factors affecting the system performance.

C. Data Analysis - Knowledge base

Data Analysis and Pattern Identification is another important functional building block of the Data Architecture shown in Fig 1. Once the performance data is available in the database, through generic data access layer, the Data Analyzer fetches the data. Initially the knowledge base is empty and every time a Context Pattern is generated it is written to the knowledge base. We used a simple custom rule mechanism which is XML based that stores condition and action to be taken. The following Fig 6 shows a XML rule template Context Pattern used to store as a rule.

```
<rule>
 <name> </name>
 <component> </component>
```

```
<if>
 <property-1>
  <name> </name>
  <value> </value>
 </property-1>
 <property-2>
  <name> </name>
  <value> </value>
 </property-2>
 <property-3>
  <name> </name>
  <value> </value>
 </property-3>
 <property-n>
  <name> </name>
  <value> </value>
 </property-n>
</if>
<then>
 <ControllerType></ControllerType>
</then>
</rule>
```

Fig. 6 : Sample Rule Template

There is a rich collection of Pattern Recognition [23] and Data Mining [24] techniques and we use some of them in our solution such as Time-Series, Episode Discovery, Outlier Analysis, and Association Rules.

D. Model and Data Organisation - Algorithm

The following Fig 7 shows the algorithm of creating templates the knowledge base is created. The control loop is run periodically and in each loop 'n' columns of the output parameter is analyzed. After each loop context pattern is transformed into a rule, written to the knowledge base.

- Create model templates
- Set the periodicity of control loop (p)
- For every interval 'p'
  - Read 'n' columns of output param
  - Analyze the data and generate Context Pattern(prediction algorithms are run)
  - Convert the Context Pattern into rules of the knowledge base
  - After 'm' control loops, call Data Mover to move all the performance data into a persistent storage

Fig. 7 : Data Organisation Algorithm

VI. IMPLEMENTATION

A primitive implementation of the proposed solution is implemented using Java and XML.

A. Data Model

The Data Modeler is a simple Java class implemented to read model templates and create tables in the column oriented database. We have used MonetDB [25] to store the model templates. But we are exploring to identify a light weight in memory column database. Currently the XML model templates have to be created manually. A tool implementation is in progress that allows creating the templates easily.

B. Knowledge base

A simple custom rule engine is developed that stores the knowledge base in XML format. A set of java classes are implemented to create the rules using the context pattern object generated. The following Fig 8 shows a sample knowledge base that is created for a context pattern of JMS Providers.

```
<rule>
 <name>JMS-MsgTput-Pattern1</name>
 <component>jms</component>
  <if>
   <property-1>
    <name>Smax</name>
    <value>125</value>
   </property-1>
   <property-2>
    <name>MsgTput</name>
    <value>160</value>
   </property-2>
   <property-3>
    <name>Pattern</name>
    <value>SuddenVariation</value>
   </property-3>
   <property-4>
    <name>CPU</name>
    <value>64</value>
   </property-4>
   <property-4>
    <name>ThresholdViolation</name>
    <value>Yes</value>
   </property-4>
  </if>
 <then>
  <ControllerType>Fuzzy</ControllerType>
    <NwSrvrInstance>False</NwSrvrInstance>
 <modelparam-1>1</modelparam-1>
 <modelparam-2>0.28</modelparam-2>
</then>
</rule>
```

Fig. 8 : A sample Rule for JMS Provider tuning

VII. DISCUSSION AND FUTURE WORK

In this paper we proposed control system model and data architecture that provides templates to represent

the various server components of the Application Server. Also, the solution exploited the mechanisms to capture, organize and use the data associated with these models for effective prediction of the future patterns of the components. The current solution is a subset of the bigger research problem that we are trying to address dealing with development of a generic end-to-end framework that enables in creating robust Application Servers that are more adaptive and self-managing in performance management. We have implemented various controllers for the JEE server components with encouraging results [10][11]112] but most of the implementation is simulation based. We intend to extend our work in validating our theory by running in actual Application Server environments. The end goal our research is to enable the design, modeling and runtime of Application Servers with inherent self-managing capabilities. We have implemented a prototype solution that supports control system concepts as first class elements in UML modeling [26]. We inted to integrate the model templates creation discussed in this paper during the UML design of the server components.

.

Additionally we want to explore Java based rule engines such as Jess [27] and evaluate against our custom rule engine. The proposed architecture implementation is primitive and it requires enahancement to complete the implemenation of all the components shown and evalute the results in different run time environments.

REFERENCES

[1] Mohammad Reza Nami, Koen Bertels, "A Survey of Autonomic Computing Systems", ICAS '07

[2] What Does Control Theory Bring to Systems Research? Xiaoyun Zhu, Mustafa Uysal, Zhikui, Wang , Sharad Singhal, Arif Merchant, Pradeep Padala, Kang Shin, ACM SIGOPS Operating Systems Review, Volume 43 Issue 1, January 2009

[3] Ravi Kumar G, C.Muthusamy, A.Vinaya Babu, "Control Systems application in Java based Enterprise and Cloud Environments – A Survey", IJACSA Vol 2 No 8, 2011

[4] N. Gandhi and D. M. Tilbury, Y. Diao, J. Hellerstein, and S. Parekh "MIMO Control of an Apache Web Server, Modeling and Controller Design", IEEE American Control Conference, 2002

[5] C. Lu, T.F. Abdelzaher, J.A. Stankovic and S.H. Son, "A feedback control approach for guaranteeing relative delays in web servers" Proc. of the 7th IEEE Real-Time Technology and Applications Symposium, pp 51-62, 2001

[6] Tarek Abdelzaher. Yina Lu, Ronahua Zhana, Dan Henriksson, "Practical Application of Control Theory to Web Services", American Control Conference, 2004

[7] Ying Lu, Avneesh Saxena and Tarek E Abdelzaher Differentiated Caching Services; A Control-Theoretical Approach, IEEE International Conference on Distributed Sysytems, 2001.

[8] Ying Lu, Tarek Abdelzaher and Gang Tao, "Direct Adaptive Control of A Web Cache System", Proceedings of the American Control Conference, Denver, Colorado, 2003

[9] Yan Zhang, Wei Qu, Anna Liu, "Adaptive Self-Configuration Architecture for J2EE-based Middleware", Vol 9, HICSS□06

[10] Ravi Kumar G, Dr.Chelliah Muthusamy and Dr.A.Vinaya Babu , "Self-regulating Message Throughput in Enterprise Messaging Servers – A Feedback Control Solution" , IJACSA, Volume 3, No 1, Jan2012

[11] Ravi Kumar Gullapalli, Dr.Chelliah Muthusamy, Dr.A.Vinaya Babu and Raj N. Marndi, "A FEEDBACK CONTROL SOLUTION IN IMPROVING DATABASE DRIVER CACHING", IJEST, Vol 3, No 7, July 2011

[12] Ravi Kumar G, Dr.Chelliah Muthusamy, Dr.A.Vinaya Babu and Raj N. Marndi, "Autonomic Database Driver – An Adaptive Control Solution", Proc. ICITEC 2012, Mar 2012,pp 40-44, Ravi Kumar G, Dr.Chelliah Muthusamy, Dr.A.Vinaya Babu "Intelligent Application Servers – A Vision of Self-managing of performance" – unpublished, accepted in ICAdC 2012

[13] Ravi Kumar G, Dr.Chelliah Muthusamy, Dr.A.Vinaya Babu "SELF-MANAGING THE PERFORMANCE OF DISTRIBUTED COMPUTING SYSTEMS – AN EXPERT CONTROL SYSTEM SOLUTION" - unpublished

[14] Z.Vukic, Ognejen Kuljaca:"Lecture on PID Controllers", http://arri.uta.edu/acs/jyotirmay/EE4343/Labs_Projects/pidcontrollers.pdf , Apr 2002

[15] ARMA: http://en.wikipedia.org/wiki/ Autoregressive_moving_average_model

[16] JEE Specification : http://www.oracle.com/ technetwork/ java/javaee/tech/index.html"

[17] JMS Providers :http://en.wikipedia.org/wiki/ Java_Message_Service

[18] EJB:" http://en.wikipedia.org/wiki/ Enterprise_JavaBeans"

[19] System Analysis and Modeling: "http://en.wikipedia.org/wiki/System_analysis"

[20] DJ.Abadi, PA.Boncz, S. Harizopoulos, "Column-oriented Database Systems", Proc ACM, VLDB'09

[21] SQL:, http://en.wikipedia.org/wiki/SQL

[22] Robert J. Scholkoff "Pattern Recognition: Statistical, Structured, Neural Approaches", John Wiley 1992

[23] Jiawei Han, Micheline Kamber, , "Data Mining – Concepts and Techniques", Morgan Kaufmann Publishers, 2006

[24] MonetDB: "http://www.monetdb.org/"

[25] Ravi Kumar Gullapalli, Dr.Chelliah Muthusamy, Dr.A.Vinaya Babu, "Design and Modeling Autonomic aware Software in UML – A Control System Solution", ICCIT,July 2012 - inpress

[26] Jess Rule Engine , "http://www.jessrules.com/"

◈◈◈