Atlantis Highlights in Engineering (AHE), volume 2
International Conference on Industrial Enterprise and System Engineering (IcoIESE 2018)

Self-Adaptive Load Balancing System for Grid Computing

Irfan Darmawan
Department of Information System
School of Industrial and Systems Engineering
Telkom University Bandung, Indonesia
irfandarmawan@telkomuniversity.ac.id

Aradea
Department of Informatics Engineering
Faculty of Engineering, University Siliwangi
Tasikmalaya, Indonesia
aradea@unsil.ac.id

***Abstract*-Load balancing is necessary for computer network infrastructure services, with the aim of balancing computing resources to achieve optimal load processing time. There are many methods that can be used for load balancing, namely by adjusting the load on the computing resources that are provided. The strategy manages the coordination of each computing resources by looking at their ability. The problem that can arise here is that there is a diverse resource involved, yang. This is related to the problems of heterogeneity, scalability, and adaptability that is the fundamental issue in the process of determining the workload on the infrastructure of computing resources. When this has been many methods that can be used for load balancing techniques, namely by adjusting the load on the computing resources will be provided. The strategy does manage the coordination of each of computing resources by looking at their ability. The problem that can arise here is that there is diversity resource involved, as well as changes in workload that can happen quickly and in real time. This is related to the problems heterogeneity, scalability, and adaptability that is the fundamental issue in the process of determining the workload on the infrastructure of computing resources. This paper introduces the mechanism of load balancing approach, taking into account its ability to adapt to changing dynamic workload, as well as the diversity of computing resources involved. The strategy used is to integrate autonomic computing approach into the component-based software so that it can give birth to the ability of self-adaptive load balancing. Based on the experimental results, the proposed model can handle the workload factor of dynamism and diversity of resource through the description of system scalability. The strategy used is to integrate autonomic computing approach into the component-based software so that it can give birth to the ability of self-adaptive load balancing. Based on the experimental results, the proposed model can handle the workload factor of dynamism and diversity of resource through the description of system scalability. The strategy used is to integrate autonomic computing approach into the component-based software so that it can give birth to the ability of self-adaptive load balancing. Based on the experimental results, the proposed model can handle the workload factor of dynamism and diversity of resource through the description of system scalability.**

Keywords-Self-adaptive systems; autonomic computing; based component; load balancing

I. PRELIMINARY

Load balancing is a way to divide or balance work (workload) between two or more computing resources (resource), a computer network, CPU, which is connected to a network to get the optimal infrastructure throughput, maximum, and response time is small [1]. Therefore, by using load balancing mechanism, is expected to improve the reliability and redundancy. The problem that can arise in this mechanism is related to the characteristics of the workload and the resource itself, where workloads are dynamic depending on events that may happen quickly and in real time, while the resource in a network of heterogeneous nature depending on the specifications of the infrastructure each,

These conditions relate to the issues [2], heterogeneity where the system consists of a variety of resources to the nature of different hardware and software as well as domain administration, and scalability. The system is expected to be developed from resources that were little and great, and the last is the adaptability, where the system can adapt to a dynamic environment. In the network system, a resource that failure is the most important provisions, where the possibility of some resources may experience failure or damage will be very high.

Based on these descriptions, there was thought to how to develop self-adaptive load balancing system, so the system can automatically adjust to the dynamism and diversity of workload and resources, and ability to handle the issue of scalability. The strategy developed is through autonomic computing approach that is realized in the form of a component-based software system.

II. SELF-ADAPTIVE LOAD BALANCING MODEL

The proposed model is a continuation of our previous work related to load balancing system [3] [4], which is integrated into our studies related to self-adaptive systems [5] [6]. Models have four main activities, namely monitors, analyzers, planners, executors, and knowledge (MAPE-K), which is inspired by the conception of autonomic computing [7] [8], as can be seen in Figure 2. The MAPE-K is abstraction control loops, where the dynamic behavior of the management system is controlled using the autonomic manager.

Each phase of the MAPE-K can be described as follows: (a) monitor, collect and pre-process information relevant context of the entity in the execution environment, so as to affect the desired properties of the target system, (b) Analyze, support decision-making about the need self-adaptation, ie the current conditions or failure requirements (event), changes in the condition (condition), and adaptation action (action), (c) Plan,

generating actions that are expected to affect the target system, according to the support mechanisms for adaptation and result analyzer, (d) Execute, implement the action plan with a view to adjusting the target system for the changes happened, (e) Knowledge, allows for the sharing of data, persistence of data, decision-making, and communication between components of control loops,and to develop multiple control loops.

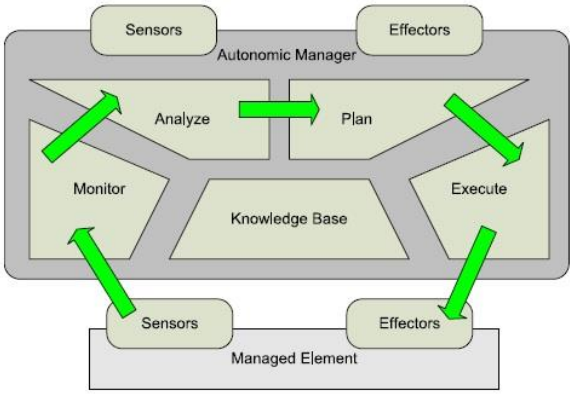

Figure 1. Process MAPE-K [7] [8].

Each stage of the activity of the computing process in Figure 1, is implemented by utilizing the architectural description languages (ADL) models Darwin [9] [10]. This model is a declarative component-based ADL to realize the hierarchical model of a system software components. There are two abstractions in this model the components and ports. The component contains a description of the software functions that can be used or receive input other components. The port is a means of components to interact; this port has two types of interfaces, namely Provided service port and required service port. To realize the ability of self-adaptive load balancing system, we create a model for system components as shown in Figure 2.

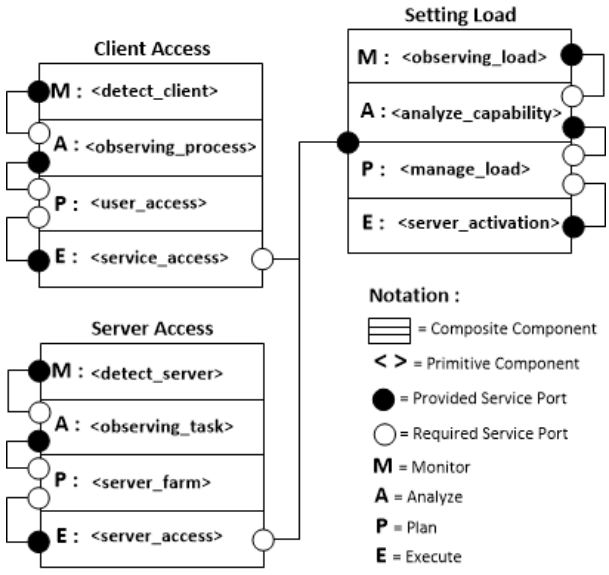

Figure 2. Self-adaptive load balancing models.

The model consists of three MAPE-K control loop, (a) client access defines the pattern for the management needs of the workload of the user as a client. (b) server access defines the pattern for the management requirements of the resource from the server farm as a service provider. (c) Setting load defines the pattern for load balancing management needs as a process of adaptation. A detailed discussion of this model of computing mechanisms is discussed in Part III, where the model is applied to the case of load balancing workload that has the dynamism and diversity of resource.

The contribution of this study is the improvement of the concept of adaptation to the diversity of resources in the process of load balancing computing, taking into consideration its evolutionary capabilities in a sustainable way. The view used is that a load balancing system in its journey will continue to grow and this becomes a major problem to be solved.

III. CASE STUDY

The case studies discussed load balancing system, is the issue of the case detailing previous investigators [11]. In the study, a description of the case was limited to discuss the application of the concept design pattern through recommendations interaction models. While in this case study, a description of the case is discussed in more detail, including the addition of the complexity of the case and its application into the proposed model, and equipped with the experimental results of the simulation. It also does well as a comparison of the results of these studies with the proposed model.

A. Specifications Case

The main target of this case study is its scalability. Several properties can identify in this matter, namely: (a) the property load and throughput, which is associated with the server component, load expressing how many processes are waiting queue to access the server or can be measured with CPU usage. Throughput measures are the number of messages the process server during a predetermined time. (b) The property of latency, which is associated with components of the client. Latency is measured from the client machine is the time it takes to send a request to receive a reply. (c) The property cost, which is associated with the server components or the number of servers active.

System Requirements this case can be illustrated as follows:

- Some system properties should be read and measured, among others (a) Property server farm that consists of a server (s_1, s_2, ... s_n) as a service provider. (b) users, comprised of some client (c_1, c_2. .. c_n) as the users of the service.
- State runtime system is represented by a combination of the value of the property. Possible changes to the property that may occur, including changes in load measurement are influenced by request for access from any client.

- Any deviations from the desired state are detected from any violation of the threshold goals each property. Possible violations are analyzed based on the possible symptom of an unwanted event.
- Symptom or event that can be identified in this case is the high load, very low load, and unresponsive.
- There are some rules for cases of load balancing this (change plan). (a) Adding new servers when high load event is raised, with a record number of new server active does not exceed the threshold. (b) Remove the server from the system, when a very low load event or unresponsive event one of them appeared.

B. Simulation Case

Some clients access the service application. At runtime, the number of clients and its load can be continuously increased or decreased, and the system must adapt to provide application services via the activation setting some servers (server farm). For example, 24 clients will access the applications with different loads, and ten servers with various specifications as well, as shown in table I and II. The system will have a consideration in determining the plan (P) to set up "user access" via the monitor component (M) "client detection" and analyze the component (A) "process observation." Also, the system also will determine the plan server farm, through the component monitor (M) "detection server" and analyze the component (A) "observation task".

TABLE I
DATA CLIENT

client (N)	Load	client (N)	Load	client (N)	Load	client (N)	Load
C1	19	C7	67	C13	43	C19	43
C2	21	C8	12	C14	65	C20	65
C3	2	C9	34	C15	23	C21	23
C4	20	C10	56	C16	56	C22	43
C5	34	C11	34	C17	34	C23	65
C6	45	C12	67	C18	67	C24	23

TABLE II
DATA SERVER FARM

Server (x)	Processors (v) = ms	Memory (m) = ms	The distance (d) = ms
S1	10	10	34
S2	6	6	2
S3	8	4	2
S4	2	8	4
S5	4	2	3
S6	6	6	2
S7	8	4	2
S8	2	8	2
S9	4	4	1
S10	6	8	2

The data in Table I and II will be the input variables for the system plan "set weight" server, through the component "observation load (M)". The following equation processes variable input:

$$f(x) = \sum_{x=1}^{n}(c_n) = \sum_{x=1}^{24}(c_x) = 961$$

Based on these equations obtained a total number of tasks that must be executed task 961. The next stage component "capability analysis (A)" determine the ability of each server to the entire task that must be processed, the analysis conducted by the equation:

$$f(x) = \frac{(\sum_{x=1}^{n}(c_n) + d_x)}{m_x \; x \; v_x}$$

The next step, the system will sort the ability of the server that can do the task as quickly as possible. Constraints of the desired process are k = 2 ms so that the equation obtains its fitness value:

$$fitness(s_x) = \frac{k}{sort(f(x))}$$

The rule for managing server load, as a response to the event, then set as follows:

- High load event is detected when the server load becomes greater than 80%.
- Very low load event is detected when the server does not perform the computing process.

The rule is based on the system "activation server (E)" by considering events for high load and the very low load, through the equation:

$$balancing = \frac{\min(fitness(S_x))}{roundup(\frac{\min(fitness(S_x))}{threshold\ high\ load})}$$

Based on these phases, the system can adjust the balance of the server and specify the number of servers as shown in Table III. Overview of execution of these functions in the form of graphs can be seen in Figure 3. The figure to the left is a condition of maximum CPU capability, after a process of balancing the obtained servers needed to balance a load of each as shown in the picture to the right.

TABLE III
SERVER FARM PROPERTY

Number server	detection Server	Sort server		Without balancing		Balancing Server	Number of Servers
	f (x) second	*f (x) second*	*Fitness*	*server*	*rest*	*balancing*	
1	9.95	9.95	498%	80%	418%	71%	1
2	26.75	20.06	1003%	80%	338%	71%	1
3	30.09	26.75	1338%	80%	258%	71%	1
4	60.31	26.75	1338%	80%	178%	71%	1
5	120.50	30.09	1505%	80%	98%	71%	1
6	26.75	30.09	1505%	80%	18%	71%	1
7	30.09	60.13	3006%	18%	0%	71%	1
8	60.19	60.19	3009%	0%	0%	0%	0
9	60.13	60.31	3016%	0%	0%	0%	0
10	20.06	120.50	6025%	0%	0%	0%	0
			Total	498%		498%	7

C. Evaluation

The problem of load balancing system adaptability highlights the importance of change management context information related to the workload is very dynamic, and the diversity of computing resources involved. In the case of simulation, it is anticipated that autonomic computing approach through a set of rules variability. Process simulation shows that the dynamic workload and resource diversity can occur continuously at run-time, and the system can handle the growing number of the stable. Thus, the load balancing system model developed can handle scalability issues as the impact of change and growth.

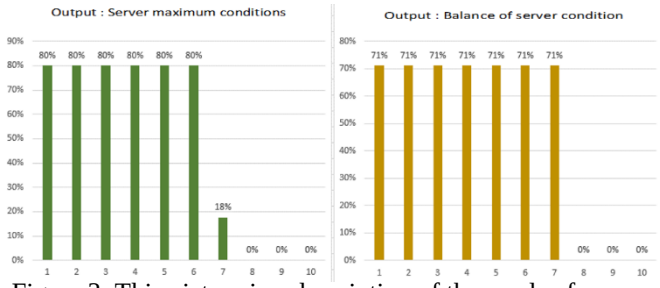

Figure 3. This picture is a description of the needs of server and load balancing functionality

Scalability load balancing system described by growth in the number of clients and the load dynamically respectively, and the average size load balance and the number of servers needed. As shown in Figure 4, with the total number of task 961 of 24 clients, required seven pieces of server load average of 71%, but if the whole task changed, for example, decreased or increased, the need for servers and the average balance of the load would adapt automatically. So the graphic shows the evaluation results scale linearly with the number of the number of servers and their clients balance the size of the load.

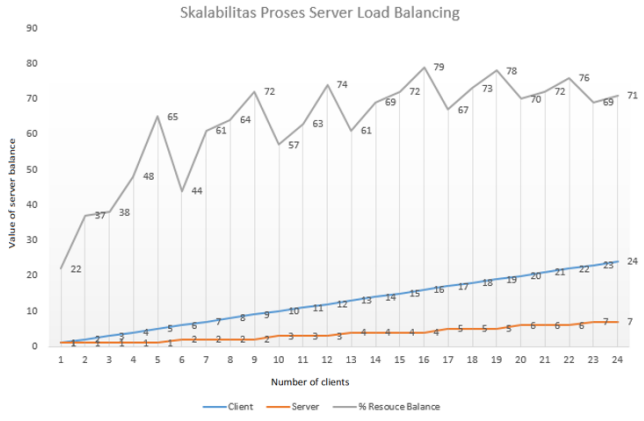

Figure 4. Scalability load balancing process.

IV. Related Jobs

Currently, there are diverse approaches proposed researchers for adaptation need load balancing system. For example, Abuseta [11] developed a design pattern concept of self-adaptive systems by taking the case of load balancing system; the proposed model reasonably reflects the adaptability at run-time through classes he developed. Here we try to modify the examples of such situations by adding complexity associated heterogeneous resource, also, the software component model that we have developed is also very possible to the needs of the dynamic evolution that have not been included in the model.

Beaulah [12], proposed a model of software load balancer based on availability and load-checker Reporters (LB-ACLRs), To reduce server overhead and load balancing. While the model we are proposing, these needs can only be implanted or handled in each control loop defined, and the process can be performed in parallel, so this will affect the performance and better scalability. Vasconcelos [13] did the design and implementation of an autonomous mechanism for load balancing of mobile data streams, models can handle variable amount and great wireless connection and automatically adapt to variations in flow volume data. The approach adopted in this model also utilizes the concept of MAPE-K, but in the model that we are proposing,

Keshvadi [14], introduced a scheduling strategy on VM load balancing by using multiple monitors and mobile agents, in this way can achieve the best load balancing and reduce or avoiding dynamic migration, thus completing the load balancing issues and migration costs are high. Bokharia [15], developed a dynamic load balancing strategy used to hypercube network in a multiprocessor system. Both studies highlight the importance of the concept of decentralization process, which is associated with the process execution time enhancement speed. It is also of concern to us forward in expanding the context inference mechanism, namely by setting the strategy for the formulation of domain knowledge.

Zarina [16], this study proposes a replication model for the federation data grid system to improve access latency by accessing data from identified areas. The strategy is to know the data from the nearest node by using the concept of network core area, this is done by diverting the search to the closest node, so the need for greater bandwidth is reduced and minimize the expected latency data access. This model is a good alternative to the implementation of collaboration and data sharing in the data grid system. This model relates to the dynamic data replication strategy, so that if our approach is integrated to this need, then it can provide more capability in terms of capturing data from the diversity of resources and dynamics of the process.

V. Conclusion

This paper proposes a model self-adaptive to the needs of load balancing functionality. Strategies developed the integration approach is through autonomic computing into software components. This model consists of four fundamental components, namely component monitor, analyze, plan, and execute, which is controlled by the

knowledge through the rule engine. Each component primitive can form composite components as a control loop, thus setting the whole system can be represented through the coordination of multiple control loops.

Based on simulation results, the model makes it possible to anticipate changes in the dynamic workload with resource diversity, so the system can automatically determine the server needs to manage the task choppy. As a form of evaluation, description of system scalability on a simulated case has demonstrated the ability of the system to handle the growing workload and resource dynamically.

We believe the proposed model can contribute to the computational domain network, where the view of the system must be based on the life cycle that will continue to grow and develop.

REFERENCE

[1] I. Darmawan, Kuspriyanto, and Y. Priyana, "The design of load balancing algorithms on dynamic tree topology network grid computing", a National Seminar on Information Technology Application (SNATI), ISSN: 1907-5022, Information Engineering Indonesian Islamic University, Yogyakarta, June 20, 2009.

[2] E. Deelman, A. Chervenak and al., "High-performance remote access to climate simulation data is: a challenge problem for a data grid technologies", In Proceedings of the 22nd parallel computing, volume 29 (10), pages 13-35, 1997.

[3] I. Darmawan, Kuspriyanto, Priyan Y., M. Ian Joseph, "Grid computing process improvement through resource scheduling using a genetic algorithm and Tabu Search integration", 7th International Conference on Telecommunication Systems, Services, and Applications (TSSA), 2012.

[4] I. Darmawan, Kuspriyanto, Priyan Y., M. Ian Joseph, "Integration of Genetic and Tabu Search algorithm based load balancing for heterogeneous grid computing", International Conference on Computer, Control, Informatics and Its Applications (IC3INA), 2013.

[5] Aradea, I. Supriana, K. Surendro, and I. Darmawan, "Variety of approaches in self-adaptation requirements: a case study", Recent Advances in Soft Computing and Data Mining, Advances in Intelligent Systems and Computing, Volume 549, pp , 253-262, Springer, 2017.

[6] Aradea, I. Supriana, K. Surendro, and I. Darmawan, "Integration of self-adaptation approach on requirements modeling," Recent Advances in Soft Computing and Data Mining, Advances in Intelligent Systems and Computing, Volume 549, pp. 233-243, Springer, 2017.

[7] JO Kephart, and DM Chess, "The vision of autonomic computing", IEEE Computer, 36 (1), pp. 41-50, 2003.

[8] IBM Corporation, "An architectural blueprint for autonomic computing", White Paper, 4th ed. IBM Corporation, 2005.

[9] J. Magee, N. Dulay, S. Eisenbach, and J. Kramer, "Specifying distributed software architectures" In Fifth European Software Engineering Conference (ESEC95), Barcelona, September 1995.

[10] D. Hirsch, J. Kramer, J. Magee, and S. Uchitel. Modes for software architectures. In EWSA, pages 113-126. LNCS 2006.

[11] Abuseta and K. Y. Swesi, "Design patterns for self-adaptive systems engineering," International Journal of Software Engineering & Applications (IJSEA), Vol.6, No.4, July 2015.

[12] Beaulah P. S., S. Rani A, RK Sahai, J. Thriveni, KR Venugopal and LM Patnaik. "Load balancing and load with availability checker Reporters (LB-ACLRs) for improved performance in distributed systems," 2nd International Conference on Devices, Circuits And Systems, (ICDCS), At Coimbatore, India, in 2014.

[13] RO Vasconcelos, M. Endler, BTP Gomes, FJ da Silva e Silva, "Design and evaluation of an autonomous load balancing system for mobile data stream processing," International Journal of Adaptive, Resilient and Autonomic Systems, 5 (3), 1- 19, IGI Global, 2014.

[14] Keshvadi S., and B. Faghih, "A Multi-Agent-based Load balancing system in IaaS cloud environment," International Journal of Robotics and Automation, Volume 1, Issue 1, November 2016.

[15] MU Bokharia, M. Alam, and F. Hasana, "Performance analysis of dynamic load balancing algorithm for multiprocessor interconnection network," Perspectives in Science (2016) 8, pp. 564-566, Published by Elsevier, 2016.

[16] M. Zarina, F. Ahmad, ANM, M. Nordin, and M. Mat Deris, "Job Scheduling for Dynamic Data Replication Strategy in Heterogeneous Federation Data Grid Systems," Second International Conference on Informatics and Applications (ICIA), 2013, DOI: 10.1109/ICoIA.2013.6650256.