

Generalized Spring Tensor Model: A New Improved Load Balancing Method in Cloud Computing

Shahrzad Aslanzadeh¹, Zenon Chaczko¹

¹ University of Technology, Sydney, Center of Real Time Information Network (CRIN), Sydney, Australia

{Shahrzad.Aslanzadeh, Zenon.Chaczko}@uts.edu.au

Abstract. Significant characteristics of cloud computing such as elasticity, scalability and payment model attract businesses to replace their legacy infrastructure with the newly offered cloud technologies. As the number of the cloud users is growing rapidly, extensive load volume will affect performance and operation of the cloud. Therefore, it is essential to develop smarter load management methods to ensure effective task scheduling and efficient management of resources. In order to reach these goals, varieties of algorithms have been explored and tested by many researchers. But so far, not many operational load balancing algorithms have been proposed that are capable of forecasting the future load patterns in cloud-based systems. The aim of this research is to design an effective load management tool, characterized by collective behavior of the workflow tasks and jobs that is able to predict various dynamic load patterns occurring in cloud networks. The results show that the proposed new load balancing algorithm can visualize the network load by projecting the existing relationships among submitted tasks and jobs. The visualization can be particularly useful in terms of monitoring the robustness and stability of the cloud systems.

Keywords: cloud computing, load balancing, collective behavior, dynamic pattern recognition

1 Introduction

Over the last decade, business and academic requirements from technology perspective have changed substantially with a greater emphasis on more powerful computing techniques. In IT, much of these changes have been driven by prompt success in Internet improvement and economical IT infrastructure development, which resulted in novel structured computational models [1]. Cloud computing is one of these newly emerged paradigms for hosting and delivering services over the Internet.

In cloud computing mapping a proper load-balancing algorithm was always an important challenge. The load on the network can be forced by CPU load, memory load, bandwidth load and tasks load [2]. Therefore due to the extensive load volume, the load balancer should prioritize the information using the distributed and heuristic algorithms. Moreover, the load should be managed in a real time manner to prevent

the overloaded pipelines and respond to the user requirements as quick as possible [3]. Reviewing the literature, different collections of algorithms have been proposed by researchers. Although the algorithms can optimize the load balancing methods, still there is a need for designing a method that can forecast the load patterns according to application types.

The purpose of this research is to architect a nature base heuristic load balancing algorithm, which can anticipate the fluctuation and magnitude of the load with various mathematical apparatus. The proposed approach can help to alleviate the problem of un-balanced load by visualizing the task dependencies pattern that can result in effective load management and resource monitoring.

2 Problem Formulation

There is a need to develop an efficient load management tool in cloud computing that can monitor the load in an elastic and scalable cloud. The load monitoring tool must not only consider the optimization method to distribute the load effectively, but also it should have anticipatory characteristics that can perform an optimal decision making.

Different types of load balancing methods have been designed using static, dynamic and hybrid algorithms. However, there are limited numbers of examination on load balancing algorithms with dependent patterns. This could be due to the complex nature of the workflow tasks and its vague behavior in terms of resource management [4]. Despite of the limited number of the works, valued solutions have been proposed which shaped the research direction in workflow load scheduling and highlighted the main gaps and their possible solutions.

Today, workflow scheduling is considered to be a key tool for automating the e-business and e-science applications. Critical applications such as earthquake modeling, climate forecasting and online booking systems for hotels and aircrafts are the example of these groups [5]. Therefore due to the complex procedure of data processing in these applications there is a need to architect a comprehensive tool incorporated with the heuristic algorithms to predict the direction and magnitude of the load changes on workflow structured applications.

3 Proposed Solution: Generalized Spring Tensor Model (STeM)

In this research we aim to evaluate the fluctuation and magnitude of the load changes in cloud computing through generalized spring Tensor algorithm. The scope of the experiment will be limited to workflow tasks, where tasks and jobs have certain dependencies to each other. Generalized spring tensor (STeM) is part of the coarse grained models. It is composed of two main components [6-7]:

- Gaussian Network Model (GNM)
- Anisotropic Network Model (ANM)

GNM is designed to predict the magnitude of the load while ANM is concentrating on the direction of the fluctuation.

Basically GNM will be effective if the tasks are located in certain distances, or in other words they are connected to each other in somehow [8].

From mathematical point of view this connectivity can be explained with Kirchhoff Matrix, also illustrated in equation 1 where r_c is representing the cut-off distance.

$$\tau_{ij} = \begin{cases} -1 & \text{if } i \neq j \cap r_{0,ij} \leq r_c \\ 0 & \text{if } i \neq j \cap r_{0,ij} > r_c \\ \sum_{j,j \neq i}^N \tau_{ij} & \text{if } i = j \end{cases} \quad (1)$$

ANM, however, was suggested as a coarse gained model which is using the simpler Hookian potentials to bypass the energy minimization needed for measuring the direction of the load [9].

ANM is using Hessian matrixes shown in equation 2 with $N \times N$ super elements, where each element is a 3×3 tensor and H_{ij} is the interaction tensor between i and j [10-11].

$$H_{ANM} = \begin{bmatrix} H_{1,1} & \cdots & H_{1,N} \\ \vdots & \ddots & \vdots \\ H_{N,1} & \cdots & H_{N,N} \end{bmatrix} \quad (2)$$

Both ANM and GNM models have their own advantages and disadvantages. They are considered as croase-gained models which do not require any energy minimization techniques. However as it is described earlier, ANM is focusing on direction of the fluctuation, while GNM is highlighting the magnitude of that.

Therefore to take the advantages of these two coarse-gained models and overcome their limitations, generalized spring tensor (STeM) was proposed, as a result of ANM and GNM combination. STeM can calculate the magnitude and direction of the load fluctuation in multi-dimensional environment base on nodes interactions [12].

To explore STeM model on cloud computing, a simple workflow application in a static structure should be considered. As shown in figure 1, each node in this workflow structure is representing a particular task, while combination of the tasks are representing group of jobs.

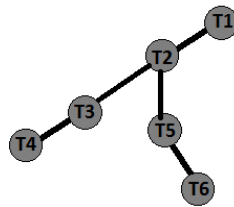


Fig. 1. - Simple workflow job modeling

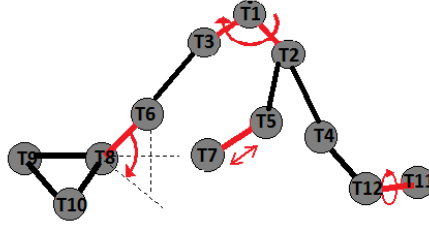


Fig. 2. - Go-Like potentials main parameters on workflow job modeling

STeM algorithm is functioning base on Go-like model. In this model, it is recognized that each task has a defined location comparing to other neighbors while they are connected to each other by a single spring [13].

Therefore in early step of applying the STeM algorithm, the first derivative of the Go-like potential will be determined. “Chain connectivity”, “Bond Angle”, “Bond dihedral” and “Non-local” interaction between tasks are the main parameters that should be computed using equation 3.

$$V(X, X_0) = \sum \text{Bond } V_1(r, r_0) + \sum \text{Angles } V_1(\theta, \theta_0) + \sum \text{Dihedral } V_3(\Phi, \Phi_0) + \sum \text{Non - Local } V_4(r_{ij}, r_{0,ij}). \quad (3)$$

Figure 2 is interpreting the main parameters of Go-like potential; bond, Angle, Dihedral and non-local interactions; on workflow tasks model. Details of each parameter are illustrated in table 1. It should be mentioned that, each task is composed of different threads with variety of commands and functionalities. Each of these threads should have priority numbers while they should start and finish by a certain time. Therefore in workflow model each task has its own depth, start-time and finish-time. Depth is depicting task’s priorities in terms of execution, while start-time and finish-time show the period needed to lock a particular resource for task completion. Figure 3 is interpreting the workflow model in hierarchical time slots.

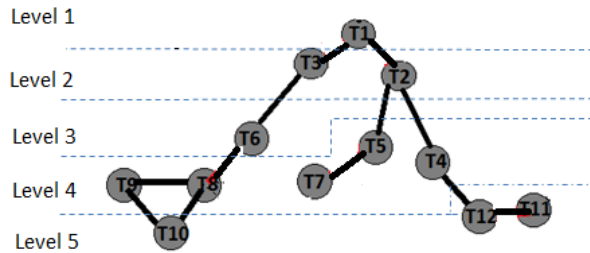
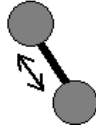
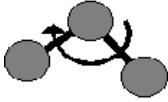

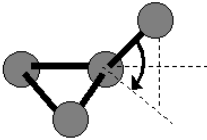


Fig. 3. – Workflow job modeling in hierarchical time manner

In next step, the second derivative of the obtained Go-like model will be calculated to determine the projection of the magnitude and direction of the load fluctuation on workflow load balancing [14]. This value can be obtained using equation (4).

$$H_{ij} = \begin{bmatrix} \frac{\partial^2 V_{1(r,r_0)}}{\partial X_i \partial X_j} & \frac{\partial^2 V_{1(r,r_0)}}{\partial X_i \partial Y_j} & \frac{\partial^2 V_{1(r,r_0)}}{\partial X_i \partial Z_j} \\ \frac{\partial^2 V_{1(r,r_0)}}{\partial Y_i \partial X_j} & \frac{\partial^2 V_{1(r,r_0)}}{\partial Y_i \partial Y_j} & \frac{\partial^2 V_{1(r,r_0)}}{\partial Y_i \partial Z_j} \\ \frac{\partial^2 V_{1(r,r_0)}}{\partial Z_i \partial X_j} & \frac{\partial^2 V_{1(r,r_0)}}{\partial Z_i \partial Y_j} & \frac{\partial^2 V_{1(r,r_0)}}{\partial Z_i \partial Z_j} \end{bmatrix} + \begin{bmatrix} \frac{\partial^2 V_{2(\theta,\theta_0)}}{\partial X_i \partial X_j} & \frac{\partial^2 V_{2(\theta,\theta_0)}}{\partial X_i \partial Y_j} & \frac{\partial^2 V_{2(\theta,\theta_0)}}{\partial X_i \partial Z_j} \\ \frac{\partial^2 V_{2(\theta,\theta_0)}}{\partial Y_i \partial X_j} & \frac{\partial^2 V_{2(\theta,\theta_0)}}{\partial Y_i \partial Y_j} & \frac{\partial^2 V_{2(\theta,\theta_0)}}{\partial Y_i \partial Z_j} \\ \frac{\partial^2 V_{2(\theta,\theta_0)}}{\partial Z_i \partial X_j} & \frac{\partial^2 V_{2(\theta,\theta_0)}}{\partial Z_i \partial Y_j} & \frac{\partial^2 V_{2(\theta,\theta_0)}}{\partial Z_i \partial Z_j} \end{bmatrix} + \begin{bmatrix} \frac{\partial^2 V_{3(\varphi,\varphi_0)}}{\partial X_i \partial X_j} & \frac{\partial^2 V_{3(\varphi,\varphi_0)}}{\partial X_i \partial Y_j} & \frac{\partial^2 V_{3(\varphi,\varphi_0)}}{\partial X_i \partial Z_j} \\ \frac{\partial^2 V_{3(\varphi,\varphi_0)}}{\partial Y_i \partial X_j} & \frac{\partial^2 V_{3(\varphi,\varphi_0)}}{\partial Y_i \partial Y_j} & \frac{\partial^2 V_{3(\varphi,\varphi_0)}}{\partial Y_i \partial Z_j} \\ \frac{\partial^2 V_{3(\varphi,\varphi_0)}}{\partial Z_i \partial X_j} & \frac{\partial^2 V_{3(\varphi,\varphi_0)}}{\partial Z_i \partial Y_j} & \frac{\partial^2 V_{3(\varphi,\varphi_0)}}{\partial Z_i \partial Z_j} \end{bmatrix} + \begin{bmatrix} \frac{\partial^2 V_{4(r_{ij},r_{0,ij})}}{\partial X_i \partial X_j} & \frac{\partial^2 V_{4(r_{ij},r_{0,ij})}}{\partial X_i \partial Y_j} & \frac{\partial^2 V_{4(r_{ij},r_{0,ij})}}{\partial X_i \partial Z_j} \\ \frac{\partial^2 V_{4(r_{ij},r_{0,ij})}}{\partial Y_i \partial X_j} & \frac{\partial^2 V_{4(r_{ij},r_{0,ij})}}{\partial Y_i \partial Y_j} & \frac{\partial^2 V_{4(r_{ij},r_{0,ij})}}{\partial Y_i \partial Z_j} \\ \frac{\partial^2 V_{4(r_{ij},r_{0,ij})}}{\partial Z_i \partial X_j} & \frac{\partial^2 V_{4(r_{ij},r_{0,ij})}}{\partial Z_i \partial Y_j} & \frac{\partial^2 V_{4(r_{ij},r_{0,ij})}}{\partial Z_i \partial Z_j} \end{bmatrix} \quad (4)$$

Table 1. - Interpretation of Bond, Angle, Dihedral and non-local connections between task on workflow model

Model	Description
	<i>Bond:</i> This parameter is defining the chain connectivity between tasks in workflow models. It will mainly highlight the potential connections between a task and its neighbors [15].
	<i>Angle:</i> This parameter is defining the angles between tasks in workflow model. The angle can be interpreted as the interval between finishing time of one task comparing with starting time of the neighbor task.
	<i>Dihedral:</i> The parameter is describing the location of the tasks after they forced by external load. Torsion can disconnect the current connections between two nodes and it can substitute that with a new relation between non-neighboring nodes [16].
	<i>Non-local interaction:</i> The parameter is defining the task's connectivity with other tasks through non-local interactions. With this model it is implied that, the forces of changes between non-neighboring tasks can be calculated.

4 Results and Discussion

In this work, we simulated a deployment of homogenous set of nodes to capture the global behavior of cloud load. Using Matlab, simulation was performed and evaluated for hundreds of nodes that represent the masses of the workflow tasks. Figure 3 is representing 100 workflow tasks, recognized by their depth, start-time, and finish-time. This workflow task has 8 main levels of time slots. The details of the first and the last time-slots of the tasks are shown in table 2.

Table 2. -Depth, start-time and finish of the selected tasks in 8 time slots

Cloud-Let ID	Depth	Start-Time	Finish Time
100	0	0.1	0.21
1	1	0.21	13.34
40	1	0.21	14.08
22	2	13.48	24.7
78	2	45.36	56.17
79	3	56.17	61.13
85	4	61.13	66.47
82	5	66.47	77
96	5	66.47	77.45
97	6	77.45	85.71
98	7	85.71	95.13
99	8	95.31	102.22

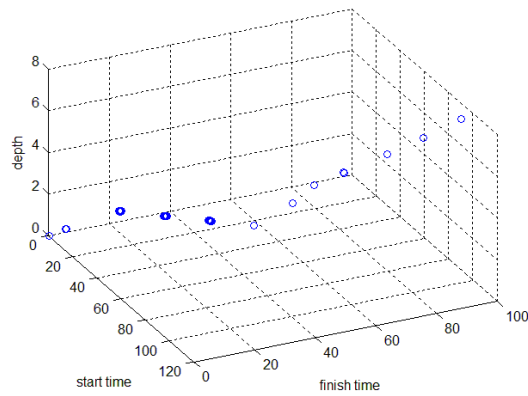


Fig. 4. -Simulated workflow tasks with depth, start time, finish time

Applying the STeM algorithm, figure 4 shows the determinant values of the analyzed Hessian for the data set of 100 tasks in one set of job. It shows the direction of the load on each task. The magnitude of the load is shown with red color, illustrating the greater value of change. As the future work, several scenarios will be considered to evaluate the behavior of the load in cloud.

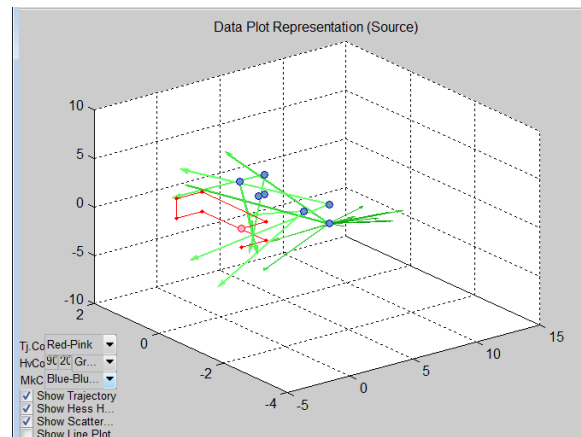


Fig. 5. -Determinant values of analyzed Hessian for data set of 100 tasks in one set of job

5 Conclusion

This research explores the adoption of STeM algorithm for visualizing the behavior of the load fluctuations on workflow tasks, in localized and globalized pattern.

In our study, Elastic Network Model (ENM) analysis was selected to evaluate the dynamics of the cloud load. Amongst the popular algorithms of ENM, Generalized Spring Tensor (STeM) was adopted to satisfy our objectives.

The foundation of STeM algorithm is based on Gaussian Network Model (GNM) and Anisotropic Network Model (ANM) which is able to detect of the motions of tasks in cloud network. The approach helps us to identify and model the magnitude and direction of the load fluctuations at a single task in a workflow application model. Simulation result tested on 100 tasks demonstrates that STeM algorithm can visualize the tasks connectivity with both local and non-local interactions. The expected benefit of our proposed model shows that STeM algorithm can be applied in designing an effective, dynamic and autonomous load balancer that is able to support optimal decision making in critical situations.

References

1. Zhang, Q., Cheng, L. & Boutaba, R. 2010. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1, 7-18
2. M. Armbrust et al., Above the Clouds: A Berkeley View of Cloud Computing, tech. report EECS--28, Univ. of California, Berkeley, 2009.
3. Yike, G., Ghanem, M. & Rui, H. Does the Cloud need new algorithms? An introduction to elastic algorithms. *Cloud Computing Technology and Science (CloudCom)*, IEEE 4th International Conference on, 3-6 Dec. 2012. 66-73.
4. Barrett, E., Howley, E. & Duggan, J. A Learning Architecture for Scheduling Workflow Applications in the Cloud. *Web Services (ECOWS)*, Ninth IEEE European Conference on, 14-16 Sept. 2011. 83-90.
5. Gupta, A., Sarood, O., Kale, L. V. & Milojicic, D. Improving HPC Application Performance in Cloud through Dynamic Load Balancing. *Cluster, Cloud and Grid Computing (CCGrid)*, 13th IEEE/ACM International Symposium on, 13-16 May 2013. 402-409.
6. Tu-Liang, L. & Guang, S. Generalized spring tensor models for protein fluctuation dynamics and conformation changes. *Bioinformatics and Biomedicine Workshop, BIBMW*. IEEE International Conference on, 1-4 Nov. 2009. 136-143.
7. Xing, L., Karimi, H. A., Yang, L. W. & Bahar, I. Protein functional motion query and visualization. *Computer Software and Applications Conference, COMPSAC 2004*.
8. Gregorcic, G. & Lightbody, G. 2007. Local Model Network Identification With Gaussian Processes. *Neural Networks*, IEEE Transactions on, 18, 1404-1423.
9. Chaczko, Z. & Aslanzadeh, S. C2EN: Anisotropic Model of Cloud Computing. *Systems Engineering (ICSEng)*, 21st International Conference on, 16-18 Aug. 2011. 467-473.
10. Guang, Spring tensor model source code, 2012, <http://www.cs.iastate.edu/~gsong/CSB>
11. Aslanzadeh, S & Chaczko, Z. Article: Generalized Spring Tensor Algorithms: with Workflow Scheduling Applications in Cloud Computing. *International Journal of Computer Applications* 84(7):15-17, December 2013. Published by Foundation of Computer Science, New York, USA
12. Kaya, H.; Liu, Z.R.; Chan, H.S. Chevron Behavior and isostable enthalpic barriers in protein folding: Successes and limitations of simple Go-like modeling. *Biophys. J.* 2005, 89, 520-535
13. Gashler, M.; Martinez, T. (2011). "Tangent Space Guided Intelligent Neighbor Finding". *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN'11)*. pp. 2617–2624.
14. Nakamura, H. K., Sasai, M. & Takano, M. 2004. Scrutinizing the squeezed exponential kinetics observed in the folding simulation of an off-lattice Go-like protein model. *Chemical Physics*, 307, 259-267.
15. Hamelryck T, Kent JT, Krogh A (2006) Sampling Realistic Protein Conformations Using Local Structural Bias. *PLoS Comput Biol* 2(9): e131.
16. Blondel, A., & Karplus, M., 1998. "New formulation for derivatives of torsion angles and improper torsion angles in molecular mechanics: Elimination of singularities". *Journal of Computational Chemistry* 17 (9): 1132–1141.