I.J. Wireless and Microwave Technologies, **2022, 4, 15-25**
Published Online August 2022 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijwmt.2022.04.02

A Theoretical Graph based Framework for Parameter Tuning of Multi-core Systems

Surendra Kumar Shukla[1], Devesh Pratap Singh[2], Shaili Gupta[3], Kireet Joshi[4]
[1,2,4]Dept. of Computer Science and Engineering, Graphic Era Deemed to be University, Dehradun, Uttarakhand, India
[3]Dept. of Computer Science and Engineering, IMS Engineering College, Ghaziabad, Uttar Pradesh, India

Vishan Kumar Gupta
Dept. of Computer Science and Engineering, Graphic Era Deemed to be University, Dehradun, Uttarakhand, India
Email: vishangupta@gmail.com

Received: 15 March 2022; Accepted: 26 May 2022; Published: 08 August 2022

Abstract: Multi-core systems are outperforming nowadays. Therefore, various computing paradigms are intrinsically incorporated in the multicore domain to exploit its potential and solve well known computing problems. Parameter tuning is a well-known computing problem in the field of Multicore domain. Addressing the said hurdle would leverage in the performance enhancement of Multicore systems. Various efforts in this direction have been made through the conventional parameter tuning algorithms in a limited scope; however, the problem is yet not addressed completely. In this research article, we have addressed parameter tuning problem by employing applications of graph theory, especially Dijkstra shortest path algorithm to address the said issue. Dijkstra's principle has been applied to establish correlation among the parameters further tuning by finding the pair of suitable parameters. Two other algorithms which are based on application feedback (to provide performance goals to the system) has been introduced. The proposed algorithms collectively (as a framework), addressed the parameter tuning problem. The effectiveness of the algorithms is verified and further measured in distinct parameter tuning scenarios and promising outcome has been achieved.

Index Terms: Multi-core system, Performance parameters, Graph, Dijkstra algorithm.

1. Introduction

Multicore systems have created a boom in processor industry, and could be witnessed with most of the computing devices incorporated multicore chips. These systems are outperforming, however have some common challenges such as low resource utilization, run time resource crises, and resource management [14,15]. Further, "knowing the performance degradation causes of multicore system" is a well-known problem needed to be addressed. Performance degradation might be caused by any computing resource (shared) among the cores of the Multicore systems. To address the said issues distinct strategies has been employed in the research community [16]. In this perspective, Graph theory has been proved to be an effective approach to find out the solution of various computing problems including multicore systems [1]. Problems such as Application-to-core mapping, addressing resource crises at run time, are few instances where graph theory has outperformed. Similarly, large scale graphs are solving highly complex problems which involve fast computation [2,3]. Further, graph theory has been extensively used for scheduling activity in multi-core systems [4]. Another important area where graph theory has leveraged the multicore systems are in establishing the energy-performance trade-offs [5,17].

To know the resource crises effects in the multicore systems, there is a dire need to perform the parameter tuning activity. Interestingly, various efforts have been made through the tuning algorithms [8], however, this problem is not addressed completely. In most of the parameter tuning strategies, profiling further "parameters extraction" is a best approach employed so far [6, 7] which is incorporated in this research work also.

Extending the parameter extraction work, (which represents and models the performance issues), in this research work, the parameters has been analyzed and their correlation has been established employing the graph theory. Further, we have incorporated the classical Dijkstra algorithm [9] to perform parameter tuning which is not attempted in all previous research works. Additionally, the parameter tuning approach has been extended and two novel approaches which are based on the "parameter threshold", incorporate "learning" respectively, are also proposed [10].

The main contribution of this research work is enlisted below as-

- Parameter extraction and correlation by employing the graphs to find the possible performance tuning alternatives.
- Employing Dijkstra shortest path algorithm to find the best "parameter sets" which have low contradiction and helps in performance tuning.
- Proposed the novel thresholds aware parameter algorithm.
- Learning based act-react model for the parameter tuning.

Rest of the paper is organized as Section 2 details the parameter extraction process. In Section 3, method to create the parameter dependence graph has been elaborated. After the parameter extraction, Parameter classification has been detailed in Section 4. Section 5 elaborate the Graph traversal approach for parameter tuning. Parameter tuning models has been detailed in Section 6. Finally, article is concluded in Section 7.

2. Parameter Extraction

The parameter tuning process comprise of parameter extraction step which is shown in Figure 1. It could be noted that the program is consist with the instructions and related program logic. The program is first profiled using the profiler (here we have used gem5 simulator). Then the extracted parameters are traced to a particular shared resource which belongs to the specific core. For example, the cache hit parameter is related to the last level cache (LLC) would be redirected to the specific core. Here the Analysis and monitoring module is introduced to evaluate the parameter effects in the performance. Analysis module also establishes the correlation among the parameters. After the correlation of parameters, the scheduling activity considering the resource availability is performed. Here Dijkstra and other algorithms are the part of Analysis and monitoring module [11].

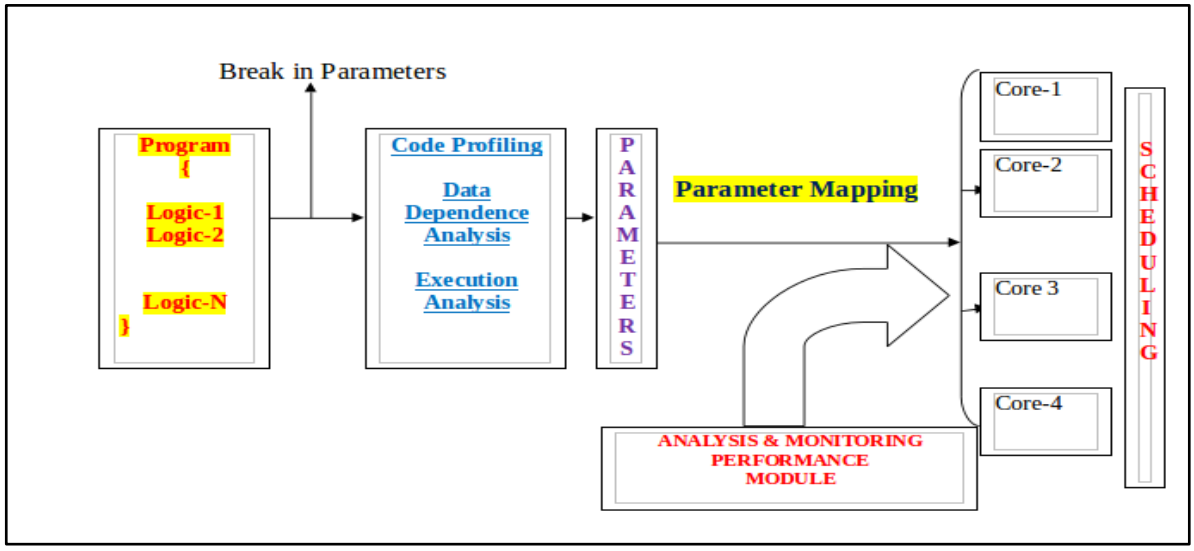

Fig. 1. Parameter extraction from the source code

3. Parameter Dependence Graph (PDG)

In order to perform the parameter Analysis and tuning, establishing the dependency among the parameters is essential. This dependency could be established through the Parameter dependence graph [12]. The procedure of creating the graph is detailed below. Identified parameters have been represented through the vertices. Additionally, the dependency among the Parameters is represented through the edges. Let us say, there are two parameters P1 & P2, if there is a dependency between Parameter 1 and Parameter 2 then an edge will be drawn from P1 to P2. The edge could follow the directed graph pattern. There might be three types of dependency among the parameters, first, increasing the performance and another, decreasing the performance, and finally, no change in the performance. Increase/decrease among the parameters would be measured by varying the respective parameters. The dependency types have been detailed below [13]:

3.1 Increasing the Performance

Changing (increasing) the value of one parameter might increase (automatically) the value of another parameter, further improvement on the performance would be measured. i.e., increasing the parallelism parameter value affects and increases the throughput of the system [18].

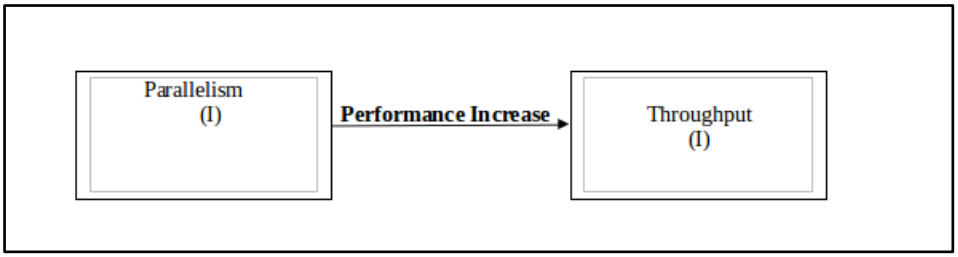

Fig. 2. Dependency between parameters (dependency Increases Performance)

3.2 No Change in Performance

Increasing the value of one parameter increases or decreases the value of another parameter. However, the impact on the performance might not be measured. For example, if we decrease the value of parallelism, will result in decrease the traffic rate in the interconnection network. However, it will not be resulted in improvement [19].

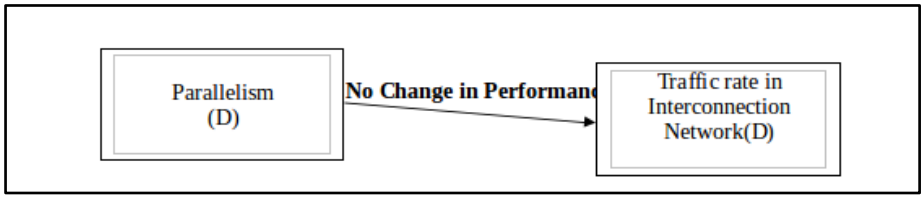

Fig. 3. Dependency among parameters (dependency not changes performance)

Another example of increasing the performance is denoted in Figure 4. Let say, we increase the cache size it will decrease the cache miss rate and then performance will increase.

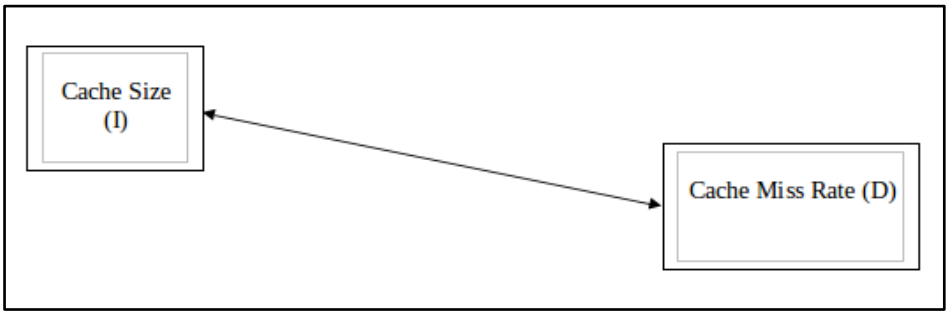

Fig. 4. increasing the cache size decreases cache miss rate (increasing performance)

Similarly, if we decrease the parameter value of one parameter and other parameter value is increased & Performance is also increasing, which is depicted in the Figure 5. The possible combination has been summarized in Table 1, and the parameter dependency graph has been illustrated in Figure 5:

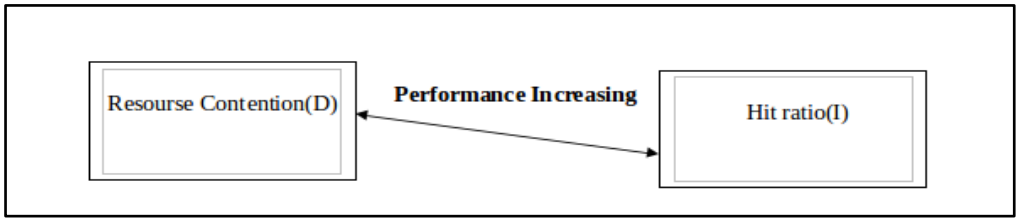

Fig. 5. Decreasing the resources contention increases the hit ratio (increasing performance)

Table 1. Parameter effects on the performance (For 2 Parameters)

Parameter 1	Parameter 2	Performance
Decrease	Decrease	No change
Decrease	Increase	Increase
Increase	Decrease	Increase
Increase	Increase	Increase

Table 2. A real scenario with parameters

Parameter 1	Parameter 2	Performance
Parallelism (Decrease)	No of Cores (Decrease)	No change
Cache Size (decrease)	Level of cache (Increase)	Increase
Cache Size (Increase)	Cache miss rate (Decrease)	Increase
No of cores (Increase)	Parallelism (Increase)	Increase

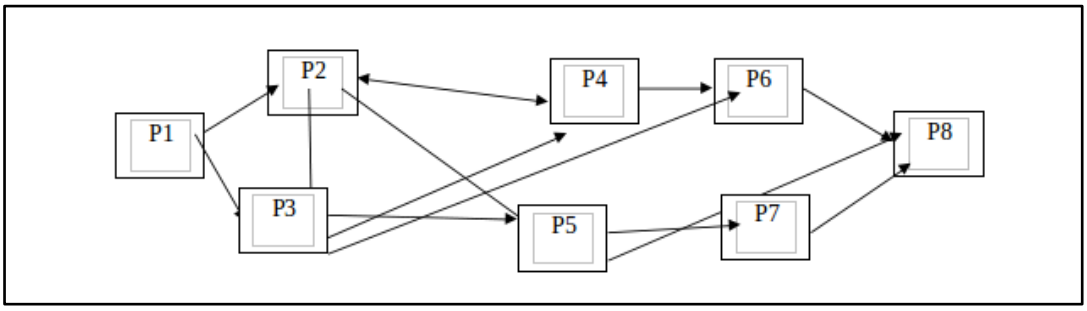

Fig. 6. Graph traversal for finding the increased performance path

Considering the parameter scenario of Table 2, the final graph has been prepared and is depicted in Figure 6. The whole graph denotes the dependency among the parameters. Further, to find the effects of parameter increasing and decreasing, the graph could be broken as a set of spanning trees [20]. Each spanning tree would denote the performance increasing or decreasing effects of the parameters. After the creation of the graph, the traversal of the graph would help to find out the performance increasing and decreasing effects. The traversal algorithm would traverse the graph aiming to create the set of vertices to improve the performance. The Figure 6 has been broken into two spanning trees and the set of vertices are given below. (Refer Figure 6)

P1->P3->P6->P7 **SET-A**
P1->P2->P5->P8 **SET-B**

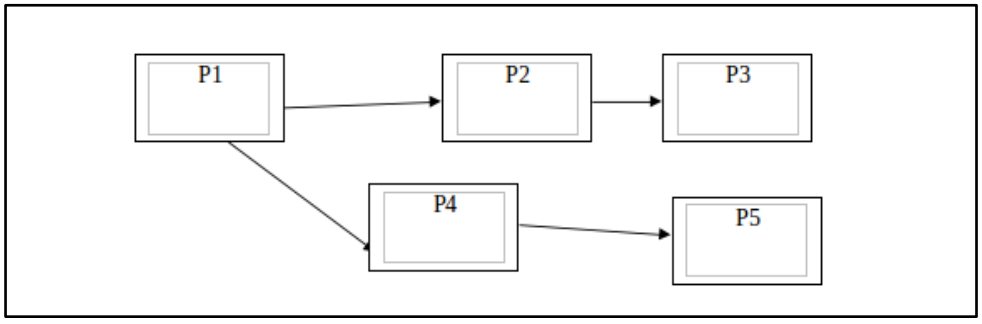

Fig. 7. Spanning tree of parameter set increasing the performance

It could be noted in Figure 7 that, we can make the sets of increasing and decreasing parameters by changing their order, we can make the set that always increases the performance. Our aim is to make the sets which always increase the performance. Figure 7 illustrate the possible parameters correlation which would always be resulted in performance improvement [21].

4. Parameter Classification

After analyzing the parameters and their correlation, the parameters were classified into three categories illustrated in Figure 8. It could be noted that an application might have N parameters and further classified in to three categories.

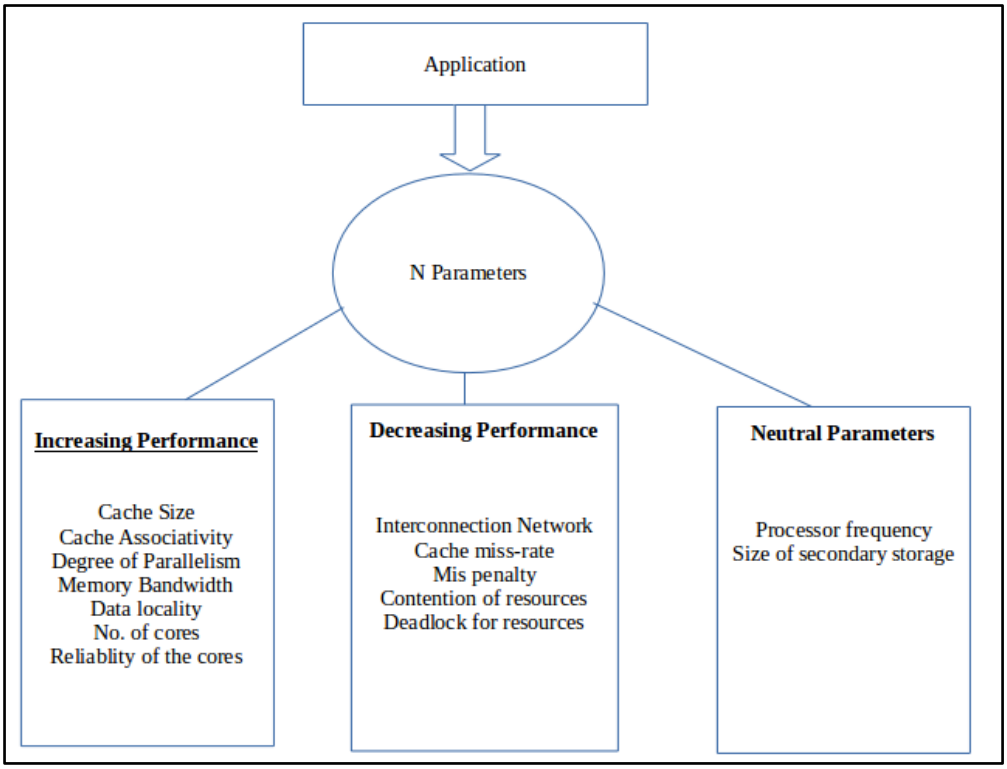

Fig. 8. Classification of parameters

5. Graph Traversal

To create the graph there are some assumptions has been set. There are three types of vertices which have been selected in the graph. The vertices are- increasing performance, decreasing performance and neutral vertex. The rule for traversing the graph follows the Dijkstra algorithm. Thus, from the current vertex, the next vertex must be the vertex, that would possibly help to improve performance, known as improve-vertex [22]. The graph shown in Figure 9 depicts the said concept.

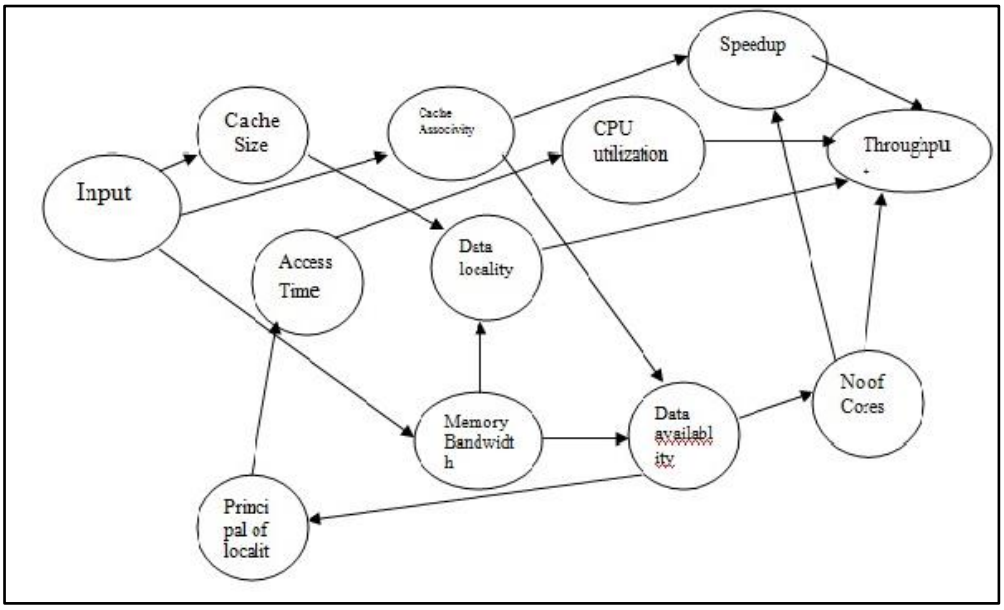

Fig. 9. Graph Traversal increasing the performance

5.1 Dijkstra Graph Traversal Algorithm

In this approach, we will first traverse the graph according to the Dijkstra principle (finding the next parameter from the current vertex whose value is high). Different cases have been discussed as below:

Case-I

For example, starting from the input, we will have 2 choices: cache size parameter whose value is (supposed to be) 80% and other parameter parallelism having 30% in the program, obviously, the concept would recommend and further choose the cache size parameter for the next node [23]. At the same time, it will be informed to the OS that the value of the parallelism parameter is less, and the OS will try to find the reason/increase the value. In the next iteration, necessary measures would be taken for performance improvement. In this way, we will have almost all the parameters whose value is high and performance would be improved. This process will be repeated till we will not get a satisfactory performance.

Case-II

In another context, let's say we have chosen a parameter whose value is high and after a while, their value starts decreasing, then we will discard that path and we will choose another path for the same. This approach of graph traversal will optimize the performance if the process of execution will be monitored continuously, and effective measures have been taken.

6. Parameter Tuning Models

To perform the parameter tuning, we have proposed distinct models. In the previous section, it is identified that, increasing/decreasing parameters value may be higher and lower. Generally, higher value for increasing the performance parameter denotes "performance increase".

6.1 Dijkstra Parameter Tuning Model

The proposed Algorithm follows the key principle: move to the vertex whose value is higher, which may increase or decrease the performance. Later on, in successive iterations, the error values would automatically be mitigated. The parameters along with their values as a weight to the edges are represented in Figure 10.

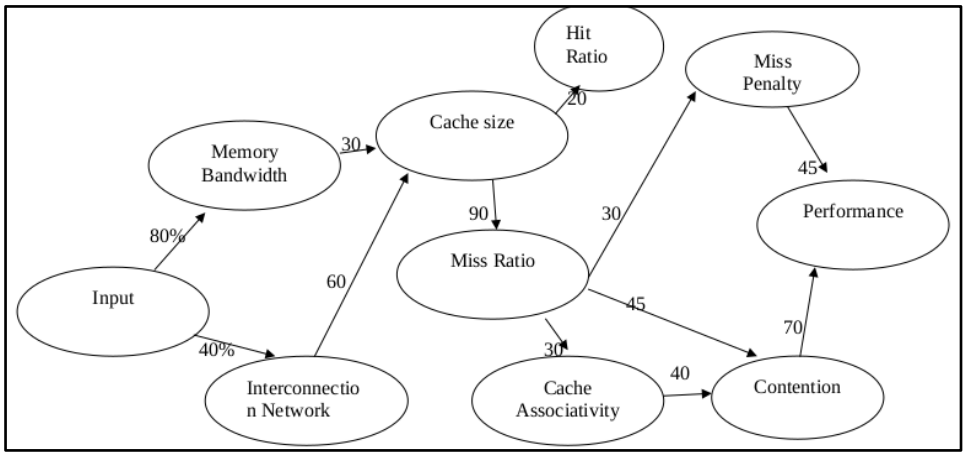

Fig. 10. Parameter weights and traversal of graph that following Dijkstra principal

The parameter tuning algorithm which is inclined to Figure 10 and next vertex of higher value is searched. The parameter tuning process is shown in Figure 11. It could be noted first, application parameters are extracted and then classified in any of the categories- increasing, decreasing, and neutral. Further, these parameters are represented through the Graph. In graph, we would apply the Dijkstra principle to tune the parameters. The performance is verified after each iteration till the execution of the program.

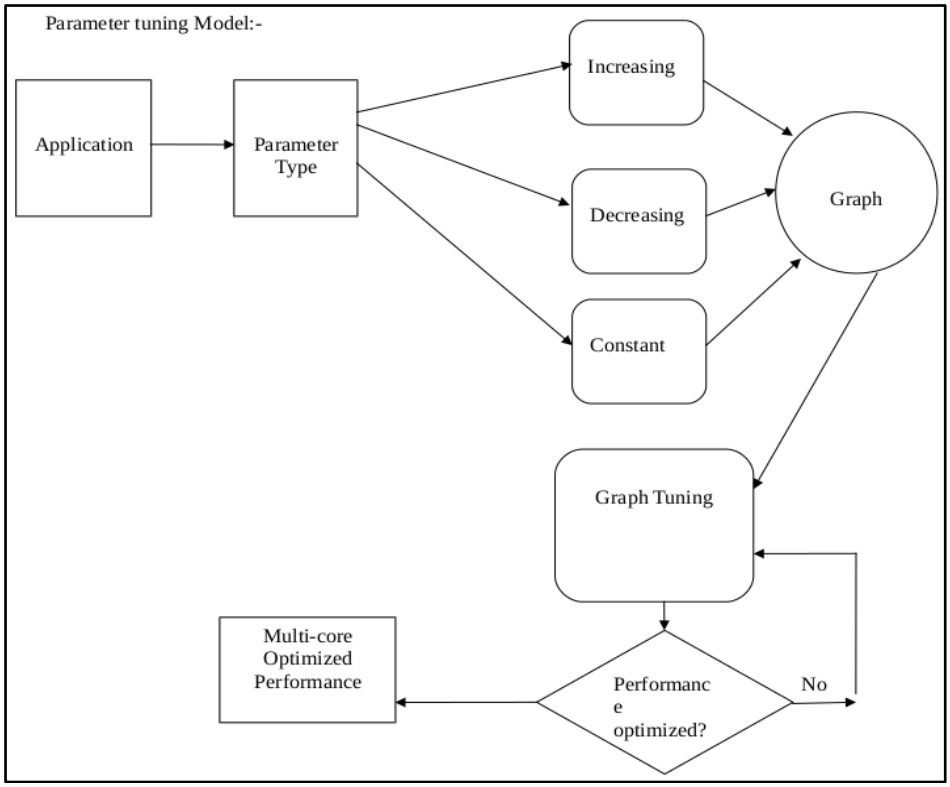

Fig. 11. Parameter tuning model

6.2 Threshold Parameter Model (TPM)

To tune the parameters, it is essential to have a proper classification of the parameters. In this model, parameters have been classified into two categories, firstly, Parameters that are directly proportional to other parameters. Secondly, Parameters are indirectly proportional to others. It means, changing the value of one parameter increases/decreases the other parameter in the same proportion is known as direct proportion. These parameter relations could be distinguished in the graph by coloring or any other graphical notations. The idea of direct and indirect proportion has been expressed with the help of the truth table, refer to Table 3

Table 3. Parameter tuning represented through the digital logic

Parameter-1	Parameter-2	Effect of correlation of Parameters
I	I	I
I	D	I
I	D	I
D	D	I
D	D	I
I	I	I
D	I	I

It could be noted that having 2 parameters we can have 2^3 combinations, therefore a total of 8 combinations of such parameters could be formed. Consequently, if n parameters are given, there would be a total of 2^{n+1} combinations.

Suppose, we have

- Parameters=2, 2^{2+1} combinations=8
- Parameters=3, 2^{3+1} combinations=16

Here, we are introducing the concept of M (Maturity), which denotes the presence of an optimal spanning tree in the graph. The optimal graph represents the best performance for a set of parameters (vertices), arranged in the form of a spanning tree. The value of M could be calculated as-

M1=f(memory)
M2=f(Cache)
M3=f(Processor)
M4=f (interconnection network)

$$\mathbf{M=M1+M2+M3+....M_n} \quad \text{(i)}$$

If the value of M is in the range of T (Threshold), it would be assumed that the performance is up to the mark. The value of T is the performance expected by an application during execution.

The above mathematical analysis is represented in Figure 12.

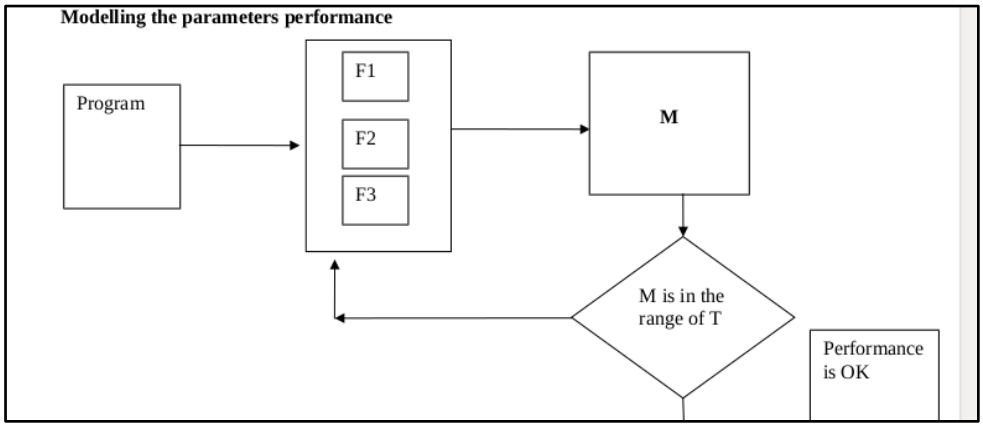

Fig. 12. Threshold parameter tuning model (TPTM)

6.3 Observe-Act-Learn (OAL) Model

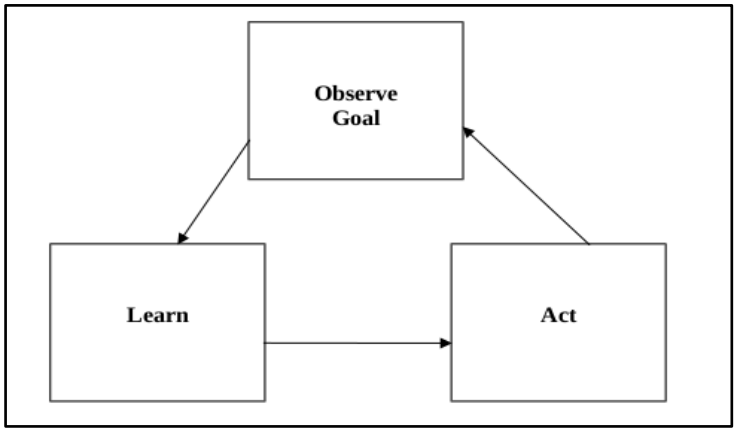

Fig. 13. OAL model for parameter tuning

In the threshold parameter model, the value of M and T should be verified periodically. To address this issue, another novel model has been proposed named OAL (Observe, Act, and Learn) (Refer Figure-13). This model is an adaptive one, which could adapt itself as per the performance requirement of the system. The goal of this model is to continuously perform analysis and verify the performance for possible improvement. After that, incorporate learning to which parameters should be changed to improve the performance. The Observer stage is used to note the behavior of the system periodically. For example, the observation that the parallelism parameter is creating the performance issue. Observation is performed to achieve the goal.

The benefit of this model is that just set the target of the performance, and everything would be monitored by the analysis module that which parameter is going to decrease the performance.

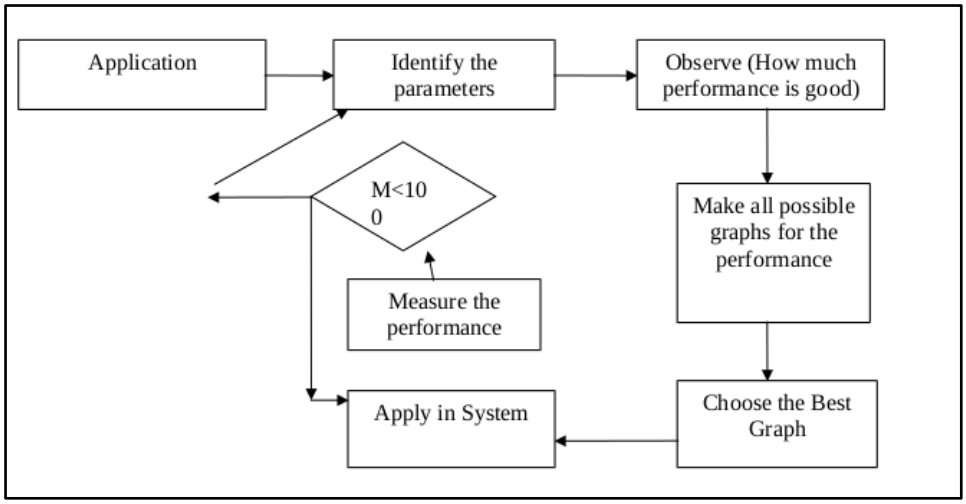

Fig. 14. Observe decide act parameter model (ODAPM)

7. Conclusion

Performance is a need of state-of-the-art applications, identifying various performance issues and working for their optimization could help to fulfil this need. To track the performance degradation causes in this paper parameter tuning based concept has been proposed. It is identified that parameter is an important aspect of the system which needs to be modelled and quantified to find insights of the system. In this work, we have incorporated the graphs to model the parameters and denote the performance degradation aspects. Further, the graph traversal is used to find out the possible performance path, in terms of spanning tree to perform the parameter tuning. To explore the graph traversal and obtain the parameter tuning, three distinct and unique models have been proposed, which identifies the spanning tree, maturity value of performance and observe and learn feature among the system parameters of the multi-core systems. The proposed algorithms are measured and verified through the mathematical modelling. The results shows that, the performance could be easily tracked by modelling through the graphs and related algorithms. The implementation aspects of the proposed algorithms have not been addressed in this paper, which could be further explored in future. Additionally, incorporating and executing these algorithms under gem5 simulator which facilitates the profiling, implementation aspects in python would certainly change the performance gap of the multi-core systems.

References

[1] Chien, S., Chang, Y., Yang, C., Hwang, Y., and Lee, J.K., 'Graph Support and Scheduling for OpenCL on Heterogeneous Multi-core Systems', In Proceedings of the 47th International Conference on Parallel Processing Companion (ICPP '18). Association for Computing Machinery, New York, NY, USA, Article 14, 2018, pp. 1–7.

[2] Huang, J., Qin, W., Wang, X. and Chen, W., 'Survey of external memory large-scale graph processing on a multi-core system', The Journal of Supercomputing', Vol. 76, 2020, pp. 549–579.

[3] Qiang, F., and Huang, H. H., 'Automatic Generation of High-Performance Inference Kernels for Graph Neural Networks on Multi-Core Systems', In 50th International Conference on Parallel Processing (ICPP 2021), Association for Computing Machinery, New York, NY, USA, Article 33, 2021, pp. 1–11.

[4] Bhuiyan, A., Liu, D., Khan, A., Saifullah, A., Guan N. and Guo, Z., 'Energy-Efficient Parallel Real-Time Scheduling on Clustered Multi-Core', IEEE Transactions on Parallel and Distributed Systems, Vol. 31, No. 9, 2020, pp. 2097-2111.

[5] Bylina, B., Potiopa, J., Klisowski, M. and Bylina, J., 'The impact of vectorization and parallelization of the slope algorithm on performance and energy efficiency on multi-core architecture', 16th Conference on Computer Science and Intelligence Systems (FedCSIS), 2021.

[6] Arndt, O. J., Lüders, M., Riggers, C. and Blume, H., 'Multicore Performance Prediction with MPET', Journal of Signal Processing System Vol. 92, 2020, pp. 981–998.

[7] Shukla, S. K., Murthy, C. and Chande, P.K., 'Parameter Trade-off and Performance Analysis of Multi-core Architecture', Published in: Progress in Systems Engineering, Springer, 2015.

[8] Huang, C., Li Y. and Yao X., 'A Survey of Automatic Parameter Tuning Methods for Metaheuristics', in IEEE Transactions on Evolutionary Computation, Vol. 24, No. 2, 2020, pp. 201-216.

[9] Zhang B. and Hu, D.J., 'Research on the construction and simulation of PO-Dijkstra algorithm model in parallel network of multicore platform', EURASIP Journal on Wireless Communications and Networking, Vol. 85, 2020.

[10] Kounev, S., Lewis, P., Bellman, K. L., Bencomo, N., Camara, J., Diaconescu, A., Esterle, L., Geihs, K., Giese, H., Götz, S., Inverardi, P., Kephart J.O. and Zisman, A., 'The notion of self-aware computing', In Self-Aware Computing Systems, Springer, pp. 3-16, 2017.

[11] Rathod, A., Thakker, R., 'Parameter Extraction of PSP MOSFET Model in Multi-core Zynq SoC Platform', Procedia Computer Science, vol. 171, 2020, pp. 1027-1036.

[12] Shukla S. S. and Chande, P.K., 'Parameter Analysis of Interfering Applications in Multi-Core Environment for Throughput Enhancement', International Journal of Engineering and Advanced Technology (IJEAT), Vol.9, Issue.2, 2019, pp. 1272-1286.
[13] Shukla, S. S. and Chande, P.K. 'Investigating Policies for Performance of Multi-core Processors', International Journal of Computer Sciences and Engineering, Vol.7, Issue.2, 2019, pp.964-980.
[14] Ezzati-Jivan, N., Fournier, Q., Dagenais M. R. and Hamou-Lhadj, A., 'DepGraph: Localizing Performance Bottlenecks in Multi- Core Applications Using Waiting Dependency Graphs and Software Tracing', IEEE 20th International Working Conference on Source Code Analysis and Manipulation (SCAM), pp. 149- 159, 2020.
[15] Jadon, S., Yadav, R.S., 'Load Balancing in Multicore Systems using Heuristics Based Approach', International Journal of Intelligent Systems and Applications (IJISA), Vol.10, No.12, pp.56-68, 2018.
[16] Khari, M., Kumar, R. L, Dac-Nhuong, Chatterjee, J. M., 'Interconnect Network on Chip Topology in Multi-core Processors: A Comparative Study', International Journal of Computer Network and Information Security, Vol.9, No.11, 2017, pp.52-62
[17] Umbarkar, A. J., Rothe, N. M., Sathe, A.S., 'OpenMP Teaching-Learning Based Optimization Algorithm over Multi-Core System', International Journal of Intelligent Systems and Applications (IJISA), vol.7, no.7, pp.57-65, 2015.
[18] Shukla, S. K., Gupta, V. K., Joshi, K., Gupta, A., & Singh, M. K. (2022). Self-aware Execution Environment Model (SAE2) for the Performance Improvement of Multicore Systems. International Journal of Modern Research, Vol. 2, No. 1, pp. 17–27.
[19] Gupta, V. K., and Rana, P. S. 'Toxicity prediction of small drug molecules of androgen receptor using multilevel ensemble model, Journal of Bioinformatics and Computational Biology', 2019. Vol. 17, pp. 1–26.
[20] Jaiswal, N., Gupta, V. K., and Mishra, A., Survey paper on various techniques of recognition and tracking. In 2015 International Conference on Advances in Computer Engineering and Applications, IEEE, 2015, pp. 921-925.
[21] Yadav, P., Varshney, R., and Gupta, V. K., Diagnosis of breast cancer using decision tree models and SVM. International Research Journal of Engineering and Technology (IRJET) e-ISSN, 2018, pp. 2395-0056.
[22] Gupta, V. K., Shukla, S.K., Anupriya, and Rawat, R.S., Crime Tracking System and People's Safety in India using Machine Learning Approaches. International Journal of Modern Research, 2022, Vol. 2, pp. 1–7.
[23] Gupta, V.K., Gupta, A., Jain, P. and Kumar, P., 2022. Linear B-cell epitopes prediction using bagging based proposed ensemble model. International Journal of Information Technology, pp.1-10.

Authors' Profiles

Dr. Surendra Kumar Shukla is currently working as an Associate Professor, at Graphic era Deemed to be University, Dehradun. He completed his B.E in Computer Engineering from SGSITS College Indore, India in 2004, M.E in Computer Engineering from IET, Devi Ahilya University, Indore, India in 2011. And Ph.D. from DAVV Indore, India in 2021. He has worked with different engineering universities in a span of 15 years in the Department of Computer Engineering. His research area is Multi-core architectures, Parallel Computing, Biomedical image processing and information security.

Dr. Vishan Kumar Gupta has received his Ph.D. degree in Computer Science and Engineering Department, Thapar Institute of Engineering and Technology, Patiala, Punjab, India. He has received his M.Tech degree in Information Technology from ABV-Indian Institute of Information Technology and Management, Gwalior, MP, India and Bachelor of Engineering in CSE from Rajiv Gandhi Technical University, Bhopal, MP, India, Currently, he is working as an Associate Professor in Graphic Era Deemed to be University, Dehradun, Uttarakhand, India. His areas of research are Computational Biology, Data Mining, and Machine learning.

Dr. Devesh Pratap Singh received the M.Tech. degree in computer science and engineering from Uttarakhand Technical University, Dehradun, India, in 2009, and the Ph.D. degree, in 2015. He is currently a Professor and the Head of the Computer Science and Engineering Department, Graphic Era deemed to be University, Dehradun, India. He has published more than 50 research articles in his area of expertise. His research interests include information security, wireless sensor networks, the Internet of Things, and soft computing.

Ms. Shaili Gupta has completed her M.Tech from Banasthali university, Rajasthan in 2010. Currently she is working in IMS Engineering College, Ghaziabad. Her area of interest is machine learning and deep learning.

Mr. Kireet Joshi has received his B.Tech in Computer Science & Engineering in Honors from H.N.B Garhwal University, Uttarakhand, and M.Tech in Computer Science & Engineering in Honors. from Govt. Kumaon Engineering College, Dwarahat, Uttarakhand. He has more than 9 years of professional experience. His research interests include Algorithm Design, Machine Learning. Currently he is working as an Assistant Professor, at Graphic Era Deemed to be University, Dehradun, India.

How to cite this paper: Surendra Kumar Shukla, Devesh Pratap Singh, Shaili Gupta, Kireet Joshi, Vishan Kumar Gupta, "A Theoretical Graph based Framework for Parameter Tuning of Multi-core Systems", International Journal of Wireless and Microwave Technologies(IJWMT), Vol.12, No.4, pp. 15-25, 2022. DOI:10.5815/ijwmt.2022.04.02

