Journal of Advanced Computing Research Vol. 3, Issue 2 (2018) 13-15

Parallel Processing Problem and Solution - A Case Study on MATLAB Parallel Computing Toolbox Performance Profiling

Ayu Fazillah Alias [#1], Mohamed Faidz Mohamed Said [#2]

[#] *Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA*

70300 Seremban, Negeri Sembilan, MALAYSIA

[1] ayufazillahalias@gmail.com

[2] faidzms@ieee.org

***Abstract*—Forecasting the execution time of computer programs is an important but challenging problem in the society of computer systems. Code analysis tools are indispensable to comprehend program behaviour. Profile tools utilize the aftereffects of time estimations in the execution of a program to increase comprehension and in this way help in the advancement of the code. This research paper is about a case study on MATLAB parallel computing toolbox performance profiling. The parallel profiler runs an extension of the profile command and the profile viewer exclusively for communicating jobs, to allow users to see how much time each worker spends estimating each function and how much time communicating or waiting for communications with the other workers. Thus, the programmers can improve their system so that the system can run smoothly. Besides, it can increase the quality of code to become better than before. This will conclude how important the profiling whiles doing the programming.**

Keyword: **MATLAB, parallel, computing, toolbox, profiling**

I. Introduction

Huge plan models contain a huge numbers of model components. The engineers effectively get overpowered keeping up the consistency of such outline models after some time. Not just it is difficult to distinguish new irregularities when the model changes, it is likewise difficult to find known irregularities [1]. Software profiling is the investigation of PC program performed by measuring the time spent on each of code, code scope or memory use amid its execution. Profiling is the initial move towards effective programming. Concentrating the streamlining just on the bottlenecks is known to amplify productivity in both advancement time and program runtime. Since components that influence the execution time are hard to anticipate previously, and the bottleneck are particularly hard to distinguish, the need of profiling instrument are evident [2].

A. Definition

Profiling is an approach to measure where a program spends time. After distinguishing which functions are devouring the most time, it can assess for conceivable execution changes. Besides, it can profile the code as a debugging tool. For instance, figuring out which lines of code MATLAB does not run can help create test cases that activite that code. On the off chance that there is a blunder in the file when profiling, it will demonstrate what ran and what did not to help segregate the issue [3].

II. Historical Background

A. MATLAB history

The main MATLAB was composed in 2000 lines of Fortran, with matrices as the main information sort, 80 functions, no .m documents and no tool stash. Jack Little, one of Moler's understudies saw MATLAB possibilities in control frameworks and signal processing. They established together Mathworks, Inc. in 1980. Mathworks is currently in charge of improvement, deal and support for MATLAB. MATLAB was revised in C with greater usefulness, for example, plotting schedules, and now it contains more than 80,000 functions [4].

B. Profiling Process and Guidelines:

1. Run the Profiler on the code
2. In the Profile Summary report, search for capacities that utilization a lot of time or that are called generally often
3. View the Profile Detail report for those functions, and search for the lines of code that take the most time or are called frequently
4. Decide if there are changes that can make to those lines of code to enhance execution
5. Execute the potential execution upgrades in the code. Spare the document and run clear all. Run the Profiler again and contrast the outcomes with the first report [3].

III. Literature review

In a research paper [5], the aim of the *gprof* profiling tool is to enable user to assess elective executions of deliberations. Researchers built up this device because to enhance a code generator they were composing. The profile can be utilized to look at and evaluate the expenses of different usage. The profiler keeps running on a period sharing framework utilizing just the ordinary administrations given by the working framework and compilers.

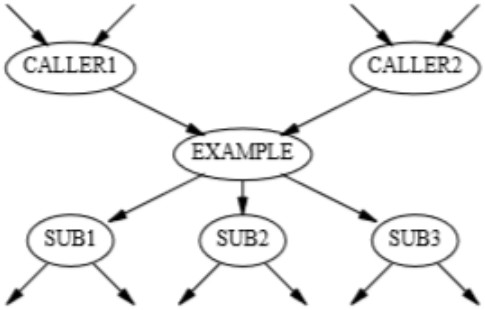

Figure 1. The call graph profile

index	%time	self	descendants	called/total called+self called/total	parents name children	index
		0.20	1.20	4/10	CALLER1	[7]
		0.30	1.80	6/10	CALLER2	[1]
[2]	41.5	0.50	3.00	10+4	EXAMPLE	[2]
		1.50	1.00	20/40	SUB1 <cycle1>	[4]
		0.00	0.50	1/5	SUB2	[9]
		0.00	0.00	0/5	SUB3	[11]

Figure 2. Profile entry

In another research paper [2], the researchers review the distinctive accessible bundles to profile code and demonstrate the focal points and burdens of each of them. The review shows that, regardless of being a long way from the instinct and ease of use of the MATLAB profiler, the distinctive created devices are getting nearer to it. For instance, the graphical show of the connection between capacities in proftools is helpful and instinctive and MATLAB does not give anything like this.

	HTML report	Graphical output	Function nesting	Line profiling
GUIProfiler	✓	✓	✓	✓
summaryRprof	✗	✗	✗	✓
`proftable`	✗	✗	✓	✓
aprof	✗	✓	✓	✓
proftools	✗	✓	✓	✗
profr	✗	✓	✓	✗
lineprof	✓	✓	✓	✓
MATLAB	✓	✓	✓	✓

Figure 3. Comparison between different profilers

In another research paper [6], the work done is to register the vitality devoured by each assignment in parallel application. TPROF powerfully follows the parallel execution and utilizes a novel method to gauge the per-assignment vitality utilization. TPROF can precisely register and imagine nitty gritty breakdowns of the vitality devoured by each undertaking and can enable the software engineer to comprehend where the vitality is spent inside his parallel application.

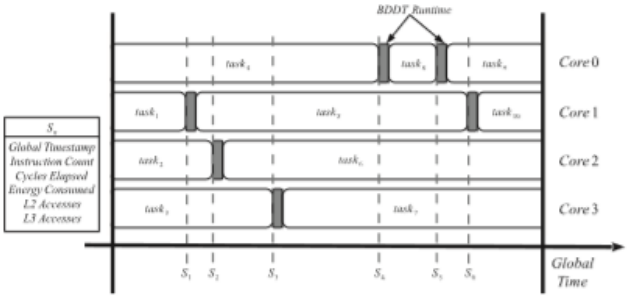

Figure 4. Example execution on a four-core processor

In another research paper [7], computing system today are widespread and extend from the little device to the vast servers, server farms and computational networks. At the heart of such frameworks are administration parts that choose how timetable to execution of various projects additional time, how to assign to each program assets, for example, memory, stockpiling and systems administration. In this paper, they proposed the SPORE to fabricate the connection between execution time of PC projects and components of the projects.

In another research paper [8], FOLD Profiler is a MATLAB code for arranging the states of profiles of collapsed surfaces as per an assortment of existing strategies. The client is offered a decision of four techniques, each in light of an alternate sort of capacity that are cubic Bezier bends, conic segments, control capacities and super circles. Thus, the investigation of an overlap appendage is fast and takes under two minutes.

IV. Conclusion

Overall this paper describes the uses of profiling in MATLAB software. The profiling tool is very important to measure time spent on each of the code so that the programmers will know what the problems are about related to the codes, and try to reduce the time spent. The profiler is a valuable tool for enhancing arrangement of schedules that execute a deliberation. It can be useful in recognizing inadequately coded schedules, and in assessing the new calculations and codes that supplant them. Taking full favorable position of the profiler requires a watchful examination of the call diagram profile, and an exhaustive information of the deliberations hidden in the program. The most effortless enhancement that can be performed is a little change to a control development or information structure that enhances the running time of the program. A conspicuous beginning stage is a normal that is called commonly. For instance, assume a yield routine is the main parent of a standard that composes the arrangements of the information. There is possibility that this arrangement routine is extended inline in the yield schedule, the overhead of a capacity call and return can be put something aside for every datum that should be organized. Eventually, the good code depends on less time spent while run the code.

REFERENCES

[1] A. Egyed, "Automatically Detecting and Tracking Inconsistencies in Software Design Models," *IEEE Transactions on Software Engineering*, vol. 37, 2011.

[2] A. Rubio and F. de Villar, "Code Profiling in R: A Review of Existing Methods and an Introduction to Package GUIProfiler," *R JOURNAL*, vol. 7, pp. 275-287, 2015.

[3] (June 14, 2017). *Profile to Improve Performance*. Available: https://www.mathworks.com/help/matlab/matlab_prog/profiling-for-improving-performance.html#responsive_offcanvas

[4] C. Moler. (2014). *The Origins of MATLAB*. Available: https://www.mathworks.com/company/newsletters/articles/the-origins-of-matlab.html

[5] S. L. Graham, P. B. Kessler, and M. K. McKusick, "gprof," *ACM SIGPLAN Notices*, vol. 39, p. 49, 2004.

[6] I. Manousakis, F. S. Zakkak, P. Pratikakis, and D. S. Nikolopoulos, "TProf: An energy profiler for task-parallel programs," *Sustainable Computing: Informatics and Systems*, 2013.

[7] L. Huang, J. Jia, B. Yu, B.-G. Chun, P. Maniatis, and M. Naik, "Predicting execution time of computer programs using sparse polynomial regression," in *Advances in neural information processing systems*, 2010, pp. 883-891.

[8] R. J. Lisle, J. L. Fernández Martínez, N. Bobillo-Ares, O. Menéndez, J. Aller, and F. Bastida, "FOLD PROFILER: A MATLAB ®—based program for fold shape classification," *Computers and Geosciences*, vol. 32, pp. 102-108, 2006.

[9] K. Whipple, C. Wobus, B. Crosby, E. Kirby, and D. Sheehan, "New tools for quantitative geomorphology: extraction and interpretation of stream profiles from digital topographic data," *GSA Short Course*, vol. 506, 2007.

[10] B. Vermeulen, A. J. F. Hoitink, and M. G. Sassi, "On the use of horizontal acoustic Doppler profilers for continuous bed shear stress monitoring," *International Journal of Sediment Research*, vol. 28, pp. 260-268, 2013.

[11] C. Sharmistha, K. N. Jukka, and S. Matti, "Design of energy-efficient location-based cloud services using cheap sensors," *International Journal of Pervasive Computing and Communications*, vol. 9, pp. 115-138, 2013.

[12] G. Pryor, B. Lucey, S. Maddipatla, C. McClanahan, J. Melonakos, V. Venugopalakrishnan, *et al.*, "High-level GPU computing with Jacket for MATLAB and C/C++," in *SPIE Defense, Security, and Sensing*, 2011, pp. 806005-806005-6.

[13] D. S. Mueller, "extrap: Software to assist the selection of extrapolation methods for moving-boat ADCP streamflow measurements," *Computers and Geosciences*, vol. 54, pp. 211-218, 2013.

[14] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Infocom, 2012 Proceedings IEEE*, 2012, pp. 945-953.

[15] A. Klöckner, N. Pinto, Y. Lee, B. Catanzaro, P. Ivanov, and A. Fasih, "PyCUDA and PyOpenCL: A scripting-based approach to GPU run-time code generation," *Parallel Computing*, vol. 38, pp. 157-174, 2012.

[16] M. Garg and L. Dewan, "Non-recursive Haar Connection Coefficients Based Approach for Linear Optimal Control," *Journal of Optimization Theory and Applications*, vol. 153, pp. 320-337, 2012.

[17] A. F. Alias. (2017, May 23). *170525 CSC580 AFA*. Retrieved from https://www.youtube.com/watch?v=1jhQVK2qx8I