International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2022): 7.942

A Statistical Approach to Temperature Estimation in Multi-Core Systems for Task Scheduling

Parth Shingala[1], Amogh Zare[2], Dhruv Khandelwal[3], Y. S. Rao[4]

Abstract: ***In the past few decades, technologies such as cloud computing, IoT, blockchain and many more which were talked about only on paper a few years back, have now come to life. High performance and faster processors have played quintessential roles in the rapid development of these technologies. As the world moves from uni-core processors to multi-core processors, there is a dire need for efficient thermal management. One such thermal management scheme is that of thermal aware task schedulers. In this paper, we present a combination of two statistical models that will help to determine the temperature of individual cores in a multi-core system. Our linear regression model estimates the current temperature of a system from the commonly available parameters of CPU power, CPU usage, etc. Further, our ARIMA model, predicts the value of the temperature of the individual core for the next time instance. This acts like an input to the thermal aware task schedulers which can then allocate the task to the core with the minimum temperature. Such a two-step temperature estimation model can then help towards efficient thermal management in multi-core systems.***

Keywords: ARIMA, Linear Regression, Multi-Core Systems, Statistical Approach, Temperature Estimation

1. Introduction

In the past few decades, technological developments have picked up tremendous pace. Technologies such as cloud computing, IoT, blockchain and many more which were talked about only on paper a few years back, have now come to life. High performance and faster processors have played quintessential roles in the rapid development of these technologies. The demand for such devices is all set to increase as the world moves towards edge computing. According to [1], a forecast made in 2011, the revenue from multicore processors is estimated to grow from 5.4 Billion € in 2011 to 12.7 Billion € in 2020.

The ability of these processors to carry out intensive tasks in a faster manner can be attributed to its multi-processing capabilities. Naturally, the execution of such intensive tasks causes the temperature of the processors to rise. According to [2], in 2018, about 205 terawatt-hours of electricity or about 1 percent of all electricity consumed that year, was consumed by the world's data centres. As mentioned in [3], about 40 % of the total energy consumed in data centres is attributed to cooling mechanisms for the equipment. Thus, there is a need to keep the temperature of the CPU cores in check. Several thermal management schemes both on the hardware level [4] [5] as well the OS level [6] [7] have been suggested before to ensure efficient thermal management.

One such method on the OS Level is to develop a temperature based task scheduler. Our work helps the task scheduler in one aspect of the decision making process. It informs the scheduler about the expected behaviour of the temperatures of all the cores in a future time instant and thus helps the scheduler to determine which core will have the minimum temperature.

2. Literature Survey

Owing to the necessity of efficient thermal management, various kinds of thermal management schemes have been proposed before. The paper in [8], uses about 21-22 in-built hardware performance counters to estimate the temperature of the processor using Linear Regression. However, the scope of this work was restricted to uni-processor systems. The paper in [9] starts with an overview of the domain of Temperature Aware Task Scheduling. It helped us in understanding the challenges and multitude of possibilities that exist in this domain.

In [10], it is suggested that conventional thermal management techniques respond only when the threshold temperature is crossed. That's why they proposed a ARMA-SPRT based model which could dynamically adapt and could predict future temperature on each core. However, they have forecasted the future temperature only on the basis of previous core temperatures and haven't taken into consideration the impact of any other system variables on the core temperature. The paper in [11] was based on their observation that the different order of execution of the same hot and cool jobs can have different resulting CPU temperatures. One aspect covered in this paper was that there is a need to consider other associated parameters while predicting the CPU temperature in a future time interval.

In [6] a probabilistic approach to solve the problem of energy minimisation is proposed. This paper aims to determine the expected energy demand after the execution of a task using statistical execution profiles. The paper in [12] primarily dealt with systems having many processors. They utilised machine learning methods like Multi-Layer Perceptron, Lasso Linear Regression Gaussian Process Model to predict the thermal profile of the entire system. However, their focus was on developing the thermal profile of the entire system.

Lastly, the article in [13] talks about how machine learning is being used for the purpose of thermal management on both single as well as multi-core systems. Some methods that were identified in this article for the purpose of thermal management were: Bayesian Learning, Neural Networks, Reinforcement Learning and Regression.

In this paper, we present a combination of linear regression & ARIMA modelling to predict the temperature of all the individual cores in the future time instance. The linear regression model takes into consideration the impact of

Licensed Under Creative Commons Attribution CC BY

International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2022): 7.942

several core-wise parameters on core temperature. Whereas, the ARIMA model, predicts the core temperature in the next time instance based on previous core temperatures.

3. Methodology

a) Generation of Dataset

To estimate the core temperature, we decided on a set of some common parameters that we believe can easily be retrieved on any system irrespective of its architecture. The parameters that we had decided are mentioned in the Table-1 below. Due to the unavailability of datasets with our specific parameters online, we created our own dataset. For this purpose, we used the MSI Afterburner [14] utility. We ran the Sysbench [15] benchmark in the background and meanwhile MSI Afterburner logged the specified parameters into the file. The data was logged for a period of five minutes. The benchmark was executed again once it was completed until five minutes of data was recorded. Once this data was logged, the data was processed and then converted into a Current Sheet View (CSV) file in a format suitable for the model. A similar method of data logging was done in [12].

Table 1: System Parameters

Core 1 Temperature	Core 1 Clock	Core 1 Usage
Core 2 Temperature	Core 2 Clock	Core 2 Usage
Core 3 Temperature	Core 3 Clock	Core 3 Usage
Core 4 Temperature	Core 4 Clock	Core 4 Usage
Core Temperature	Core Clock	Core Usage
RAM Usage	CPU Power	

b) Training the model

In order to build an algorithm to select core on the basis of temperature for task scheduling, our end goal is to create a forecasting model that can determine the core that will have minimum temperature in the next time instance. However, instead of using just previous temperatures, we developed a linear regression model that will first estimate the current temperature based on current core parameters. In order to determine the value of individual core temperatures at a future time instant, it is important to understand and consider the impact of associated input features such as Core Clock, Core Usage, CPU Power and RAM Usage. This current temperature estimation model tries to capture exactly that.

Another important reason why this model is developed is because not all systems have an in-built temperature core wise sensor. Thus, this model tries to estimate the current temperature values based on the values of some of the commonly used parameters which have been described already in the previous section.

We started with observing the data of Core 1 input and output variables that we had collected. Figure 1 shows the correlation matrix between the CPU Core 1 temperature and all other associated variables.

	CPU1 temperature
CPU1 temperature	1.000000
CPU1_1 usage	0.420329
CPU1_2 usage	0.389283
CPU usage	0.482955
CPU1 clock	0.054799
CPU clock	0.050842
CPU power	0.358373
RAM usage	0.174760
CPU temperature	0.875011

Figure 1: Correlation Matrix of the CPU1 Temperature and associated values

Thus, to develop a current temperature estimation model we tried out three different models viz. Linear Regression Model, Polynomial Regression Model a neural network model which were developed using the scikit-learn [16] and tensorflow [17] libraries in python.

1) *Linear Regression Model*: Linear Regression is a mathematical model that tries to establish a linear relation between the input variables and the output variable. It does so by determining a best fit line such that the the cumulative perpendicular distance of all the points from the line is minimum. Figure 2 shows the output of the linear regression model obtained for core 1 temperature and Table 2 shows the errors that were obtained.

Table 2: Linear Regression Results for Core 1

Mean Absolute Error	0.7245736
Mean Squared Error	0.8126314
Root Mean Squared Error	0.9014607

	Actual	Predicted
257	52	51.332543
258	48	46.808059
259	47	45.731939
260	48	48.810954
261	47	46.988168

	Actual	Predicted
257	52	51.332543
258	48	46.808059
259	47	45.731939
260	48	48.810954
261	47	46.988168

Figure 2: Comparison of the Linear Model Predicted results and actual value of temperature for Core 1

2) *Polynomial Regression Model:* Mathematically, Polynomial Regression & Linear Regression is very similar to each other just that Polynomial Regression tries to establish a non- linear relationship between input and output variables. It tries to determine a higher-order (order > 1) function. Keeping the input and output variables same as before, we developed the polynomial regression model. Figure 3 shows the output that was obtained for core 1 and Table 3 shows the cumulative errors obtained for core 1.

Licensed Under Creative Commons Attribution CC BY

	Actual	Predicted
257	52	51.433046
258	48	47.962438
259	47	46.412287
260	48	48.571103
261	47	46.846243

Figure 3: Comparison of the Polynomial Model Predicted results and actual value of temperature for Core 1

Table 3: Polynomial Regression Results for Core 1

Mean Absolute Error	5.821
Mean Squared Error	278.83846
Root Mean Squared Error	16.698457

3) *Neural Network Model:* The Neural Network Model is basically the collection of neurons which are joined together with edges. All the neurons and edges have some random weights assigned which gets changed as and when learning proceeds. The output of a given neuron is calculated by the weight of the given neuron and the incoming input. These weights will then be changed using the back-propagation of error and gradient descent to get the closest possible result. Figure 4 shows the output that was obtained for core 1 and Table 4 shows the cumulative errors obtained for core 1.

Table 4: Neural Network Model Results for Core 1

Mean Absolute Error	1.3836713
Mean Squared Error	3.4914973
Root Mean Squared Error	1.8685549

4) *Current Temperature Estimation Model Summary:* The results that we obtained after applying all the above three models are summarised below in Table 5. Thus, as it can be seen from the above table, Linear Regression performed better in terms of root mean squared error as compared to the other two models. Thus, we decided on

	Actual	Predicted
257	52	51.778770
258	48	49.454193
259	47	47.523792
260	48	49.659706
261	47	49.961990

Figure 4: Comparison of the Neural Network Model Predicted results and actual value of temperature for Core 1

The linear regression model for the purpose of current temperature estimation. Thus, the flow of the training aspect of the current temperature estimation model can be summarised as shown in Figure 5

Table 5: Mean Square Error Results of Three Different Models

	Root Mean Squared Error
Linear Regression Model	0.9014607
Polynomial Regression Model	16.698457
Neural Network Model	1.8685549

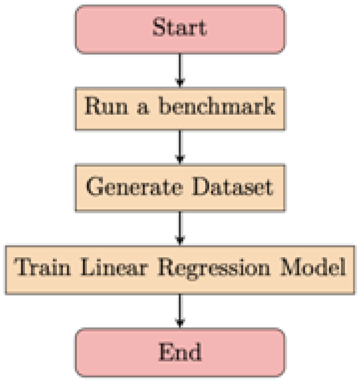

Figure 5: Flowchart for current temperature estimation of the system

c) Testing the Model

1) ARIMA Model: Auto-regressive Integrated Moving Average Model is a Time Series model which uses the past values of a variable and learns from it. It then uses it to forecast the value of the variable to give the closest result. ARIMA model is the combination of AR model that is Auto-regressive model which focuses only on the past values of the given variable and MA model that is Moving Average model which focuses only on the past forecasted error. The mathematical equation of ARIMA is as shown below:

$$Y_t = \alpha + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \beta_3 Y_{t-3} \ldots \beta_p Y_{t-p} + \epsilon_t + \varphi_1 \epsilon_{t-1} + \varphi_2 \epsilon_{t-2} \ldots \varphi_q \epsilon_{t-q} \quad (1)$$

Equation 1: ARMA mathematical equation

2) *Testing Algorithm:* During the testing phase, For the first 10 iteration i.e. till sufficient data is obtained, only linear regression is performed. This is done to consider the impact of associated variables on CPU Core temperature. However, after 10 iterations, both linear regression and ARIMA forecasting is used. The entire algorithm can be seen in Figure 6.

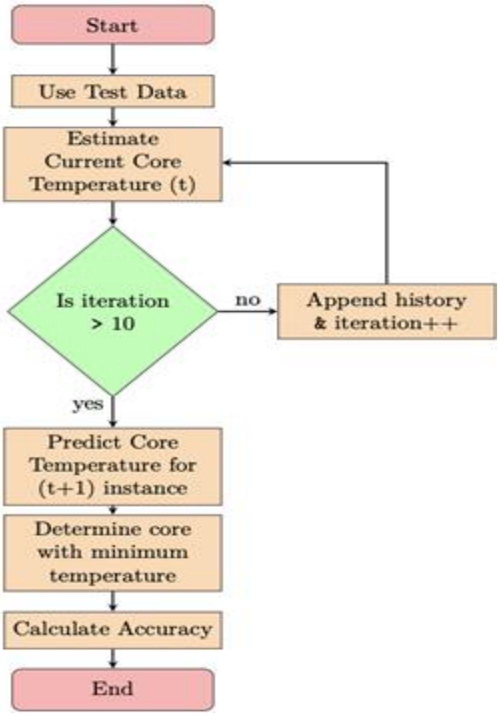

Figure 6: Flowchart for temperature prediction in the next time instance.

Licensed Under Creative Commons Attribution CC BY
Paper ID: MR22614065111 DOI: 10.21275/MR22614065111

4. Results and Analysis

After combining both the linear regression model and the future temperature forecasting model and following the method- ology mentioned in figure 6, following results, as can be seen in figure 7, were obtained. Both predicted and expected temperatures for all the cores are displayed. Further, our model gives a suggestion of the core which has the minimum predicted temperature for the next time instance. This suggestion can then be considered by a task scheduler, which can then allocate the task to that particular core.

For instance, in Figure 7, we can clearly interpret that in the pseudo task number 20, the predicted temperature of the core 2 is minimum so core 2 is selected. Same for task number 21 where core 4 has the minimum temperature so it is selected by the algorithm. The mean squared error between the predicted and the expected values of all the four cores are mentioned in Table 6.

```
================================================================================
Number: 20
--------------------------------------------------------------------------------
CORE 1: predicted=47.68    expected=51.00   ||   CORE 2: predicted=46.47    expected=48.00
CORE 3: predicted=47.39    expected=51.00   ||   CORE 4: predicted=46.61    expected=48.00
--> Core 2 Selected
================================================================================
================================================================================
Number: 21
--------------------------------------------------------------------------------
CORE 1: predicted=48.28    expected=49.00   ||   CORE 2: predicted=46.93    expected=48.00
CORE 3: predicted=48.35    expected=47.00   ||   CORE 4: predicted=46.67    expected=47.00
--> Core 4 Selected
================================================================================
================================================================================
Number: 22
--------------------------------------------------------------------------------
CORE 1: predicted=50.45    expected=50.00   ||   CORE 2: predicted=46.86    expected=48.00
CORE 3: predicted=49.20    expected=51.00   ||   CORE 4: predicted=46.43    expected=48.00
--> Core 4 Selected
================================================================================
```

Figure 7: Comparison of the temperatures of all 4 cores and selection of the core

Table 6: Error metrics calculated on the predicted results of all the individual cores

	Mean Squared Error
Core 1	1.7831255734914426
Core 2	1.2410225835357465
Core 3	3.174885572958408
Core 4	1.4193039729912649

	Mean Squared Error
Core 1	1.7831255734914426
Core 2	1.2410225835357465
Core 3	3.174885572958408
Core 4	1.4193039729912649

The expected and predicted values for all the individual cores can be visualised as shown below:

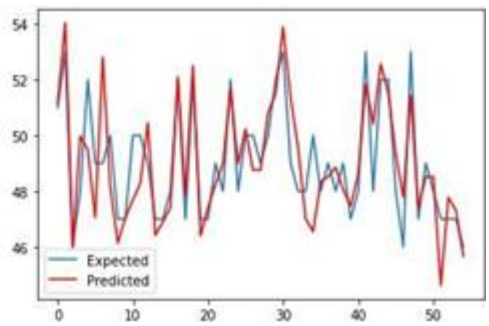

Figure 8: Results graph of CPU Core 1 comparing actual and predicted temperature

Thus, as it can be seen in figure 8, the expected and predicted temperature values of core1 seem to be overlapping. Even though it may not seem the same for other cores, there's not much difference in their predicted and expected temperature values with difference of 2°C at the most.

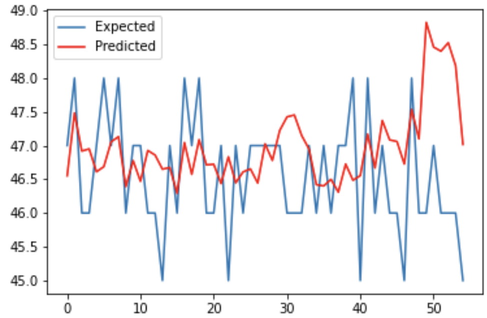

Figure 9: Results graph of CPU Core 2 comparing actual and predicted temperature

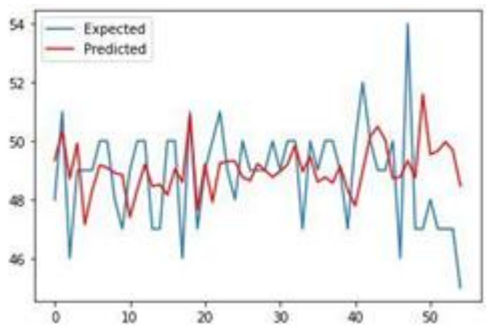

Figure 10: Results graph of CPU Core 3 comparing actual and predicted temperature

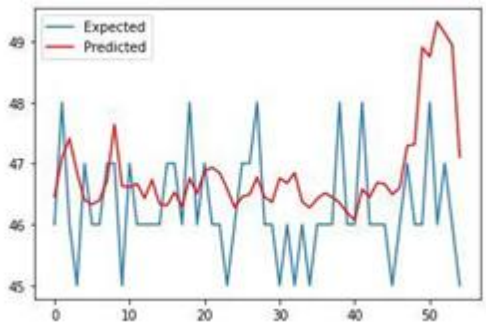

Figure 11: Results graph of CPU Core 4 comparing actual and predicted temperature

5. Conclusion and Future Scope

Our model has been able to predict the temperature of all the four cores with an average error of 1.65 °C. Our project uses a combination of linear regression and ARIMA model to consider the impact of associated processor features while predicting the future temperature of the core. This serves as an input to the thermal aware task schedulers

This work is by no means a comprehensive solution to the problem of temperature estimation in multi-core systems. Even though we have proposed a methodology for that challenge, we believe this is a preliminary work and requires some more efforts in this direction. Some suggestions from our side are: Firstly, the dataset was developed while running only one benchmark. However, a

Licensed Under Creative Commons Attribution CC BY

more representative dataset can be formed while running a suite of benchmarks. Secondly, only processor level features are considered for determining the temperature in the future time instance. However, task features can also be included for accurate prediction. Further, the temperatures are determined only for the (t+1) instance. However, the time instance for which the value is to be predicted should be determined by the characteristics of the tasks.

Acknowledgement

We are extremely grateful to the Electronics & Telecommunication Department at Sardar Patel Institute of Technology for their constant support and guidance.

References

[1] The Impact of the Introduction of Multi-core Technologies on the Computing Market and Opportunities for Europe: http://publications.europa.eu/resource/cellar/d2eeb993 - 5e7c-403d-b0bc-fda57388d211.0001.01/DOC 1
[2] Data Centers Responsible for 1 Percent of All Electricity Consumed Worldwide: https://www.datacenterknowledge.com/energy/study-data-centers-responsible-1-percent-all-electricity-consumed- worldwide
[3] Z. Song, X. Zhang, C. Eriksson, Data Center Energy and Cost Saving Evaluation, EnergyProcedia, Volume 75, 2015, Pages 1255-1260, ISSN 1876-6102, https://doi.org/10.1016/j.egypro.2015.07.178.
[4] D. Brooks and M. Martonosi, ”Dynamic thermal management for high-performance microprocessors,” Proceedings HPCA Seventh International Symposium on High-Performance Computer Architecture, Monterrey, Nuevo Leon, Mexico, 2001, pp. 171- 182, doi: 10.1109/HPCA.2001.903261.
[5] K. Skadron, M. R. Stan, W. Huang, Sivakumar Velusamy, Karthik Sankaranarayanan and D. Tarjan, ”Temperature-aware microarchitecture,” 30th Annual International Symposium on Computer Architecture, 2003. Proceedings., San Diego, CA, USA, 2003, pp. 2-13, doi: 10.1109/ISCA.2003.1206984.
[6] Thorsten Zitterell and Christoph Scholl. 2010. A probabilistic and energy-efficient scheduling approach for online application in real-time systems. In Proceedings of the 47th Design Automation Conference (DAC ’10). Association for Computing Machinery, New York, NY, USA, 42–47. DOI:https://doi.org/10.1145/1837274.1837287
[7] A. K. Coskun, T. S. Rosing and K. Whisnant, ”Temperature Aware Task Scheduling in MPSoCs,” 2007 Design, Automation Test in Europe Conference Exhibition, Nice, 2007, pp. 1-6, doi: 10.1109/DATE.2007.364540.
[8] A. Kumar, Li Shang, Li-Shiuan Peh and N. K. Jha, ”Hyb-DTM: a coordinated hardware-software approach for dynamic thermal management,” 2006 43rd ACM/IEEE Design Automation Conference, San Francisco, CA, 2006, pp. 548-553, doi: 10.1145/1146909.1147052.
[9] Chrobak M., Du¨rr C., Hurand M., Robert J. (2008) Algorithms for Temperature-Aware Task Scheduling in Microprocessor Systems. In: Fleischer R., Xu J. (eds) Algorithmic Aspects in Information and Management. AAIM 2008. Lecture Notes in Computer Science, vol 5034. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-68880-8_13
[10] A. K. Coskun, T. S. Rosing and K. C. Gross, ”Utilizing Predictors for Efficient Thermal Management in Multiprocessor SoCs,” in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 28, no. 10, pp. 1503-1516, Oct. 2009, doi: 10.1109/TCAD.2009.2026357.
[11] Xiuyi Zhou, Jun Yang, Marek Chrobak, and Youtao Zhang. 2010. Performance-aware thermal management via task scheduling. ACM Trans. Archit. Code Optim. 7, 1, Article 5 (April 2010), 31 pages. DOI:https://doi.org/10.1145/1736065.1736070
[12] K. Zhang et al., ”Machine Learning-Based Temperature Pre- diction for Runtime Thermal Management Across System Components,” in IEEE Transactions on Parallel and Distributed Systems, vol. 29, no. 2, pp. 405-419, 1 Feb. 2018, doi: 10.1109/TPDS.2017.2732951.
[13] S. Pagani, P. D. S. Manoj, A. Jantsch and J. Henkel, ”Ma- chine Learning for Power, Energy, and Thermal Management on Multicore Processors: A Survey,” in IEEE Trans- actions on Computer-Aided Design of Integrated Circuits and Systems, vol. 39, no. 1, pp. 101-116, Jan. 2020, doi: 10.1109/. TCAD.2018.2878168.
[14] MSI Afterburner: https://www.msi.com/Landing/afterburner
[15] SysBench Benchmark: https://github.com/akopytov/sysbench
[16] F. Pedregosa , et al., “Scikit-learn: Machine learning in python ” J. Mach. Learn. Res., vol. 12, pp. 2825 – 2830, 2011.
[17] Tensorflow: https://www.tensorflow.org/

Licensed Under Creative Commons Attribution CC BY