A Classification Model for Predicting Suitable Cache Level in a Multi-core Architecture

Thasneema V, Bhargavi R. Upadhyay, Sreebha Bhaskaran
Dept. Of Computer Science & Engineering
Amrita School of Engineering, Bengaluru
Amrita Vishwa Vidyapeetham, India
thasneema2414@gmail.com, bhargavi.upadhyay@gmail.com,b_sreebha@blr.amrita.edu

Abstract— **Cache memory has an important role in achieving system performance in multi-core architecture. Finding the best suitable cache configuration for an application is a very important step while designing a computer system to improve the performance. The commonly considered cache memory design parameters are the size of the cache, line size, associativity, type of cache, replacement policy, write policy and levels of cache. Selecting these parameters decides the design goals such as system performance, energy consumption, area, scalability and programmability of an application. Cache design space is a time consuming and complex process as it involves studies on the impact of all possible cache parameter configurations on system performance. This project mainly aims at creating a model that can predict an efficient cache configuration -cache level- that is best suitable for an application in terms of energy consumption and performance in a multi-core environment using machine learning techniques such as classification. The design goal here is to predict the optimum cache levels for an application by considering the design parameters such as cache sizes, associativity, block size etc. This method will be carried out in a multi-core environment for the studies on advancements in computer architecture. The required data set is generated using two simulators such as Gem5 and CACTI. The entire process of data collection is automated and applications from different domains will be considered here with different cache parameter combinations. The performance of the classifier is measured based on the evaluation metrics such as Precision, Recall, and F-measure. Performance measurement concerning power consumption and execution time would be the figure of merits of this project.**

Keywords—Design space exploration, simulation, Gem5, CACTI, cache recommendation, classification.

I. INTRODUCTION

The speed of the memory component is very low compared to the speed of the modern processors. To get good performance during program execution, achieving lower memory access time is of utmost importance. Hierarchical memory architecture is the solution introduced to get faster access to the data compared to the main memory. The main goal of a computer architect is to improve system performance during program execution. Here the performance means reducing the execution time. Another major design goal is reducing power consumption due to the underlying hardware. To improve computational power and execution time increase the number of cores in the processor chip. A higher level of the cache may be shared among multiple cores. Compared to single-core processors multi-core processor has better performance, power consumption and thermal effect [1].

a. Need for the design space exploration

Memory hierarchy consumes a significant amount of the processor's power. A program runs effectively when the resources utilized by the program consume lesser power and still giving faster execution. For this to happen the application must be able to utilize the available memory effectively. To achieve this design goal identifying an optimal cache configuration for an application is necessary. This method is known as the design space exploration of cache (CDE) memory [2][3]. Different cache variables need to be considered during CDE such as cache size, block size, associativity, cache type, cache policy etc.

CDE involves evaluating these parameters effect on cache performance. Quality of Service provided by a cache depends on memory demand of the application. The main goal of cache optimization is to reduce the energy consumed andexecution time. This design goal is highly important for the advancement in high-performance computing and the embedded system. In a multi-core environment, design space exploration needs to consider all the cache parameters as well as their different combinations to find an optimal cache parameter combination.

Finding the best suitable cache parameter combination for an application is a time-consuming process since it needs to consider all cache parameters and their different configurations [4]. Also, it may require extra hardware, and implementing software for this purpose may be highly complex.

b. Challenges need to be addressed

This paper mainly aims to predict the best suitable cache parameter combination in a multi-core environment with different levels of cache memory. Here we are considering a shared memory architecture. Different applications were run in a `simulation environment using GEM 5 and CACTI simulators. For each application, almost all possible combinations of cache parameters are considered to choose the best suitable cache variable combination for an application. We propose a classification model that can predict suitable cache variable combinations (cache level) by considering performance and energy consumption as the design parameters. The goal here is to come up with a cache prediction model to reduce search time in a multi-core environment. The data set is passed into many different classifiers to select the best classifier based on the evaluation metrics such as Precision, Recall, and F-measure. This work also has taken an account of real-time usage of the model. And also tried to reduce the model's miss prediction error as much as possible.

II. RELATED WORKS

A lot of studies and researches are taking place in the field of CDE for the advancement in computations. As the size of data needs to be processed is increasing day by day, the execution must be faster along with with the efficient utilization of available resources [5]. Several studies have been conducted in optimizing the major design goals like power expenditure [6] and execution time. Different cache parameters need to be optimized during the designing of cache memory. This may vary for each application. Therefore, choosing the best cache parameter combination for an application is very important for a cost-effective system modeling and many methods have been proposed for the same. This includes Simulation-based, Analytical based, Configurable cache and evolutionary techniques.

a. Cache Design Parameters

Several cache design variables need to be adjusted for the efficient execution of each application. The main cache parameters are type of cache, size of cache, line size, associativity, cache levels, write policy and replacement policy [7]. Several combinations of these parameters are possible and choosing the best suitable parameter combination is very important for an application to best utilize the available resources. Designers main challenge is to identify the best suitable combination of these parameters for an application. Their usual practice is to select a cache configuration which gives an average performance for a series of applications. But this is not a good practice. This work will see how each parameter is affecting the performance of an application and try to predict the best suitable variables of cache for a new application. Existing methods of CDE are listed below.

- Simulation-based design: It is used for evaluating the system performance and correctness for the proposed hardware. Running the simulation environment using software will take very little time. It is of three types: Functional simulator: based on hit rate and miss rate. Full system simulator: It shows the actual system environment. The most commonly used full system simulators are SimOs [8] and Gem5 [9] It is more time taking as it considers almost all parameters while designing and Trace driven simulator.

- Analytical based exploration: It doesn't use simulation instead uses a mathematical formula. It finds hit rate and miss rate using analysis methods [8]. Here the cache design space can be created by changing cache size and associativity. The algorithm takes input as miss rates and produces the desired cache parameter combination. But there are some limitations like the accuracy of the result may be less and it is a complex model. It works based on some assumptions on the application [10][11].

- Configurable cache: Several cache reconfiguration methods are proposed for uni-core systems [12][13]. A self-tuning cache can reduce this energy consumption to some extent by configuring the cache during run time. We can configure cache variables such as block size, cache size, and associativity during the run time [14]. The cache tuning is applied in a specific interval of time. MASTER is an energy-efficient model with a reconfigurable cache used in a multi-core system [15]. A SMART cache is also proposed for implementing an energy-efficient cache model [16].

- Evolutionary techniques: It uses both hardware and software methods for solving optimization problems. This method investigates the parameters that determine the cache variable combination in terms of energy consumption and chip area. The performance evaluation criteria used here are hit rate, miss rate, and CPI. In a multiprocessor system-on-chips the optimizations are mainly done on cache hierarchy, block size and associativity. Genetic algorithm is a method proposed [17] for optimizing cache configuration in the system on chip architecture.

With the aim in reducing the search time for cache optimization, machine learning technologies are also understudying for the CDE of cache memory. This will be highly helpful for the efficient execution of an application that involves large and complex computations. In such applications optimizing the available storage will lead to the creation of a cost-effective system. Several machine learning techniques are proposed for the optimization in unicore and multi-core systems [19][20][21].

Cache recommendation in uni core system in which statistical classification (supervised learning technique) is used to predict the best cache variable combination for each application by considering energy consumption and performance. Machine learning techniques are also used to improve cache

replacement policies-LeCaR. (Learning Cache Replacement) method [22]. It is online learning with regret minimization. Two replacement policies considered here are LRU and LFU. Power management in a multi-core processor using supervised learning is also an important research area [23].

b. Multi-core system design challenges

But there are several challenges for the designing of a multi-core environment. Increasing the number of cores in a single chip provides faster execution but it introduces a new problem due to higher power consumption and heat dissipation [24]. Some of the issues associated with the multi-core environment are it requires larger memory subsystems, cache coherence, Multi-threading, and Parallel programming, etc. We need to consider all these aspects while designing a multi-core system and appropriate cache design need to be identified for a balance between execution time and energy consumption.

III. Design and Methodology

The proposed work aims to develop a classification model that can predict the best suitable cache parameter combination for an application in terms of the time of execution and energy expenditure. Applications from different domains are run here to generate the required data set which will be passed into the classifier. The entire procedure is divided into three steps. 1) Setting up the simulation environment, 2) Data set generation and 3) Classification and evaluation of the model. First, two steps are completely automated to get an error-free data set (error due to manual intervention). Different applications were run in simulators with many different combinations of cache parameters to get the energy-related

and execution time-related components which are shown in Table I. After the feature extraction the data set will be passed onto many different classifiers to identify the best classifier with lesser error in prediction. The entire procedure is shown in the architecture diagram fig [1].

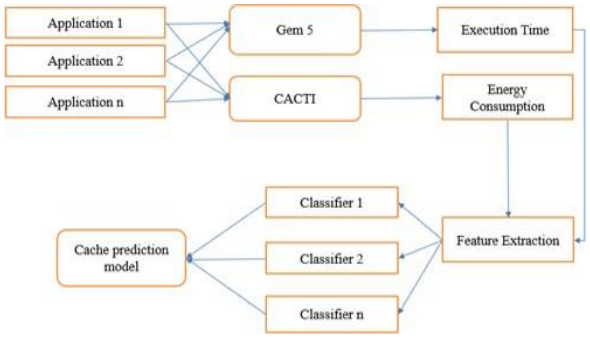

Fig [1]: Architecture diagram of the model.

a. Simulation Environment

The simulation is carried out in a multi-core environment where the cache variables are changed for different applications. The main goal of the simulation is to extract the time of execution and energy expenditure related parameters for an application with different cache parameter combinations. Gem 5 is used to generate execution time-related parameters and CACTI is used to generate Energy consumption related parameters. The parameters changed here are associativity, cache size at each level, block size and cache type. Since it is a multi-core environment number of cores also changed (2,4 and 8) to identify the effects of cores on performance. Different cache parameters are used in this work are listed in TABLE I.

TABLE I. Cache Configuration Parameters

Cache Parameters	***Values***
Size of cache	8,16,256...32768 kB
Line size	32,64,128 kB
Associativity	2,4,8,16
Cache type	Instruction cache, Data cache

The Gem5 Simulator [25] is a flexible and modular simulator capable of evaluating a broad range of architectures. The infrastructure provided by Gem5 helps computer architects to explore the diverse set of CPU models, system execution models and memory systems, models. In the GEM5 simulator, our first step is simulating more levels of cache before running the applications. This is to identify the best suitable cache level for an application. Here we are configuring level 3 and

level 4 caches. We can create caches with specific parameters in Gem5. Buses are configured for connecting different levels of cache and main memory. PARSEC benchmarks have run to obtain the parameters such as Hit rate, Miss rate, Frequency of each instruction type and Execution time in a multi-core environment. CACTI is a tool that takes several cache parameters to calculates the time of access, power, cycle time and area of a cache memory [26]. Several features supported by cacti include Power, delay, area, and cycle time model for direct-mapped, set associative and fully associative caches. The input parameters we can change in the CACTI include the capacity of cache, cache line size, cache associativity, Number of cache levels and the number of cores. From the output CACTI, we get details of cache variables that minimize power consumption, area etc.

b. Data profiling

To generate the required data profile from gem5 and cacti stats file we need to create a shell script which involves automatic compilation and simulation of each program with different cache variable combinations. Each instance in the data set represents an application. The flow chart to create a CSV file

from output files (TXT) of gem5 and cacti simulators are given below.

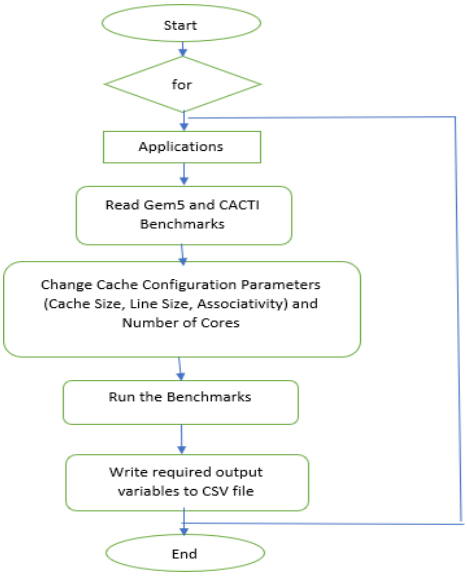

Fig [2]: Flow chart of the automated system

c. Feature Selection

Stats file of the simulation generate many features related to execution time and energy consumption. Out of these features the required features for our data set need to be selected which is done based on the performance equation and equation for energy consumption. This section explains how each feature is selected for the data set. The performance equation and equation for total energy consumption are given as

$$\text{Execution time} = IC * CPI * CT \quad (1)$$
$$\text{Ecache} = Es + Ed \quad (2)$$

Where IC is the Instruction Count, CPI is the Cycles Per Instruction and CT is the Clock Time. Es stands for Static Energy and Ed stands for Dynamic Energy. Hit rate and Miss rate are also taken into account since it affects the memory access time. Effective CPU time means actual execution time and memory access time. Frequency of different instruction type from multiple cores is also considered while calculating IC. This includes an average count of IntAdd, IntMult, Load, Store, Branch etc instructions from multiple cores. A new level of cache (l3 and l4) is simulated here to improve the accuracy in prediction. The data profile is shown in the below tables II and III. An application is represented using a row in TABLE III

TABLE II. INPUT PARAMETERS GIVEN TO THE SIMULATORS

L1i size	L1d size	L2 size	L3 size	Block size	Associativity	Number of CPUs
...		...		...		...
16	8	256	512	64	4	2
...		...		...		...

TABLE III. OUTPUT PARAMETERS OBTAINED FROM SIMULATORS

Int ALU	IntMult	Float Add	Load Instr	Store Instr	CPU Branch	Hit Rate	Miss Rate	Dynamic Energy	Leakage Energy
	...		...		...		...	...	
	12	759		...		0.87	0.12	0.171	0.53
	...		...		...		...	...	

d. Developing a Model for Prediction

The class label to be predicted here is the best suitable cache level (up to L4) for an application. A mapping functions from input variables to the output variable need to be identified in terms of frequency of instruction and hit rate. Here a range for the values of time of execution and energy expenditure is fixed for an application by taking an average of multiple simulations with different cache configuration. The cache level for which the least execution time and energy consumption under the given parameter specification are taken as the best suitable cache level. The data set generated using simulators will be passed on to different classifies to choose the best performing classifier in terms of the least number of miss prediction and accuracy while predicting. The procedure is shown in Fig [3].

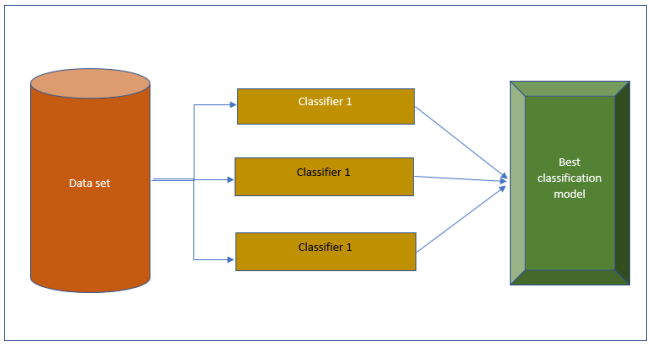

Fig [3]: Applying different classifiers to select the best classifier.

Here we are using a machine learning tool WEKA for training and evaluating the model. WEKA is used for data mining which includes a lot of machine learning algorithms and evaluation techniques which is used to select the best classification model. Many different classifiers are tested with different combinations of features. The different classifiers used here are BayesNet, Naive bayes, Ada boost, Bagging, Random subspace, Stacking, Decision table, JRip, PART, OneR, ZeroR, J48, and Random forest.

IV. RESULTS AND VALIDATION

With four levels of cache (including instruction and data cache at L1 cache), the possible combinations of different cache sizes obtained are 5,00,000. Different applications are run with

these combinations along with the number of cores (1,2,4,...) . Here the cache level prediction is based on the design goals such as Execution time and Energy consumption. A histogram is plotted against the number of applications fall in different cache levels is shown in Fig [3**].** From the figure, it is clear that most of the applications fall in the cache levels of L1, L2, and L3. This is because even if we provide more levels of cache or increase the cache size, the optimum performance application may utilize a lower level of cache or lesser cache size. It means increasing the cache size may not increase the performance all the time. That is the reason we need to choose the best suitable cache variable combination for an application. The threshold for choosing the application for training the model is 3. That is an application is selected for the model only if it falls in a particular cache level at least three times with the given cache parameters.

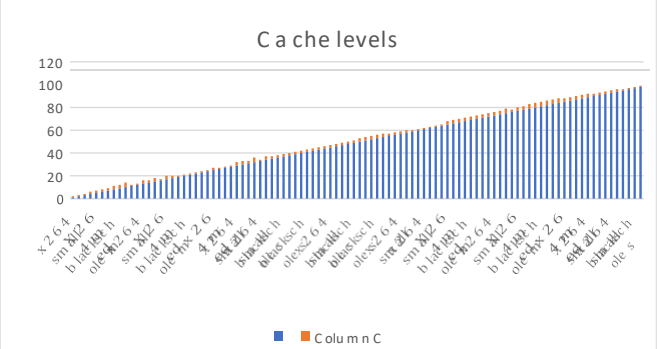

Fig [3]: Histogram showing applications per class (cache level). Series1 indicates different applications and Series2 indicate class labels.

We got a balanced data set where each of the cache variable combinations and cache levels has a fair number of applications falls in it. After the data set generation feature extraction is carried out to make the data set more practical. This includes feature extraction and feature creation (miss rate) and feature reduction. Using the WEKA tool different classification algorithms are applied to the data set to select the best prediction model in terms of the time of execution and energy expenditure. The label predicted here is the best suitable cache level for an application. The evaluation methods used here are 10-fold cross-validation and evaluation metrics used include Precision, Recall, and F-Measure.

TABLE IV. RESULTS FROM DIFFERENT CLASSIFIERS

SI.No	Classifier	Precision	Recall	F-Measure	Number of miss classified instances
1.	Random Forest	0.943	0.933	0.938	80
2.	Decision Tree	0.900	0.904	0.901	89
3.	Random Subspace	0.950	0.934	0.941	55
4.	Logistic Regression	0.814	0.811	0.830	65
5.	SVM	0.875	0.845	0.871	102

The best performing classifier is Random Subspace which has a Precision of 95% along with Recall and F-measure are 0.934 and 0.941 respectively. Random Forest Classifier also shows a precision of 94%, Recall of 0.933 and F-measure of 0.938 which is almost near to Random Subspace. We can say that our model gives better performance because almost classifiers show comparatively very little difference in their Precision, Recall, and F-measure values. The number of wrongly classified instances for Random Subspace is 55. Other classifiers also show a near number of wrongly classified instances which is 0.0098%. our model is optimized to have a very small number of wrongly classified instances. This is done by feature extraction and 10- fold cross-validation.

V. CONCLUSION

This work mainly aimed to come up with a model that can predict the suitable cache level for an application while considering the time of execution and energy expenditure as the design goals. Several benchmarks are run which included applications from different domains in two different simulators to generate the data set in terms of the time of execution and energy expenditure. A higher level of cache is simulated (up to level 4 cache) in a multi-core environment to obtain a better design space of cache for the applications. Major cache parameters considered here are the size of the cache, line size, and associativity. The number of cores is also changed here to identify the better performance.

Many different classifiers have been evaluated here to identify the best model for predicting suitable cache level for an application. A 10-fold cross-validation method is applied in which 10% of the data is used for testing. Most of the classifiers given almost similar accuracy, which shows that our model is a good model to be used as the prediction model. The classifier with the highest accuracy is chosen here is the Random Subspace with a precision of 95%. This model can be used to predict a suitable cache level for a new application using which time and cost for simulation to identify a suitable configuration while deigning hardware can be reduced.

This opens up a large area of studies and research in CDE of cache for applications from different domains including complex machine learning applications. The accuracy of the proposed model can be improved using better feature engineering and using an improved data set for classification. Further studies are possible by applying the above method in a heterogeneous environment. This model can be used to optimize other design goals in computer architecture such as the area utilized by the memory components. By considering applications from many more different domains and taking all possible combinations of cache parameters in CDE of cache can be further improved.

REFERENCES

[1] A. Gamatie, R. Ursu, M. Selva, G. Sassatelli, "Performance prediction of application mapping in manycore systems with artificial neural networks", *Proc. 10th Int. Symp. Embedded Multicore/Many-Core Syst. Chip*, pp. 185-192, 2016.

[2] Bhargavi R. Upadhyay, Sudarshan TSB, "Design Space Exploration of Cache Memory –A Survey", International Conference on Electrical

Electronics and Optimization Techniques (ICEEOT), pp. 2294-2297, 2016.

[3] Belwal, Meena, and T. S. B. Sudarshan. "A survey on design space exploration for heterogeneous multi-core." *2014* International Conference on Embedded Systems (ICES). IEEE, 2014.

[4] Navarro O., Mori J., Hoffmann J., Stuckmann F., Hübner M. A Machine Learning Methodology for Cache Recommendation. In: Wong S., Beck A., Bertels K., Carro L. (eds) Applied Reconfigurable Computing. ARC 2017. Lecture Notes in Computer Science, vol 10216. Springer, Cham.

[5] Upadhyay, Bhargavi R., and T. S. B. Sudarshan. "Task-Enabled Instruction Cache Partitioning Scheme for Embedded System." Proceedings of First International Conference on Smart System, Innovations and Computing. Springer, Singapore, 2018.

[6] R. G. Kim, J. R. Doppa, P. P. Pande, D. Marculescu and R. Marculescu, "Machine Learning and Manycore Systems Design: A Serendipitous Symbiosis," in Computer, vol. 51, no. 7, pp. 66-77, 2018.

[7] Lars Wenker , Sebastian Altmeyer, "Design-Space Exploration for Embedded System Caches through Simulation", University of Amserdam, June 9, 2017.

[8] Rosenblum, m., bugnion, e.,devine, s., and herrod s.a. 1997. Using the simos machine simulator to study complex computer systems. Acm trans. Model. Comput. Simul. 7, 1.78–103.

[9] N. Binkert et al., "The GEM5 simulator," CAN, vol. 39, pp. 1–7, 2011.

[10] N. Kumar, C. Pascoe, C. Hajas, H. Lam, G. Stitt and A. George, "Behavioral Emulation for Scalable Design-Space Exploration of Algorithms and Architectures," Lecture Notes in Computer Science, pp. 5-17, 2016.

[11] MEklov, d., black-schaffer, d., and hagersten, e. 2011. Fast modeling of shared cache in multicore systems.in proceedings of the 6th international conference on high performance and embedded architectures and compilers.Acmnewyork,ny,147–157.

[12] Xu, C., Chen, X. Dick, R. P., And Mao, Z. M. 2010. Cache contention and application performance prediction for multi-core systems. In Proceedings of the IEEE International Symposium on Performance Analysis of Systems & Software (ISPASS). IEEE, Piscataway, New Jersey, 76–86.

[13] Xiang, x., bao, b., bai, t., ding, c., and chilimbi, t. 2011. All-window profiling and composable models of cache sharing. In proceedings of the 16th acm symposium on principles and practice of parallel programming. Acm new york, ny, 91–102..

[14] A. Gordon-Ross and F. Vahid, "A self-tuning configurable cache," DAC, 2007.

[15] W. Wang et al., "Dynamic cache reconfiguration for soft real-time systems," ACM TECS, 2011

[16] M. Biglari, K. M. Barijough, M. Goudarzi and B. Pourmohseni, "A fine-grained configurable cache architecture for soft processors," 2015 18th CSI International Symposium on Computer Architecture and Digital Systems (CADS), 2015.

[17] S. Mittal et al., "MASTER: A Multicore Cache Energy Saving Technique using Dynamic Cache Reconfiguration", IEEE TVLSI, 2013.

[18] K.T. Sundararajan, T.M. Jones, N. Topham, "Smart Cache: A Self Adaptive Cache Architecture for Energy Efficiency", Proc. Int'l Conf. Embedded Computer Systems (SAMOS), 2011.

[19] J. Diaz, J. I. Hidalgo, F. Fernendez, O. Garnica, S. LÓpez, "Improving SMT performance: An application of genetic algorithms to configure resizable caches", Proceedings of the 11th Annua Conference, 2009.

[20] J.D. Eklov and E. Hagersten. StatStack: E_cient Modeling of LRU caches. In Proceedings of the 2010 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS-2010), White Plains, NY, USA, Mar. 2010.

[21] H. Jung and M. Pedram. Supervised Learning Based Power Management for Multicore Processors. IEEE Transactions on Computer-Aided Design, September 2010, 29(9):1395-1408.

[22] Karthik T. Sundararajan, Timothy M. Jones, Nigel P. Topham, The Smart Cache: An Energy-Efficient Cache Architecture Through Dynamic Adaptation,International Journal of Parallel Programming, 2013, Volume 41, Number 2, Page305

[23] G. Theocharous, et al. Machine Learning for Adaptive Power Management. Intel Technology Journal, 2006.

[24] Giuseppe Vietri, Liana V. Rodriguez, Wendy A. Martinez, Steven Lyons,Jason Liu, Raju Rangaswami, Ming Zhao, Giri Narasimhan, Driving Cache Replacement with ML-based LeCaR, 10th {USENIX} Workshop on Hot Topics in Storage and File Systems (HotStorage 18), 2018.

[25] H. Jung and M. Pedram. Improving the Efficiency of Power Management Techniques by Using Bayesian Classification. In Proc. Intl. Symposium on Quality of Electronic Design, pp. 178-183, 2008.

[26] Anil Sethi1, Himanshu Kushwah, "Multi core processor technology advantages and challenges", IJRET: International Journal of Research in Engineering and Technology, eISSN: 2319-1163 | pISSN: 2321-7308, Sep 2015.

[27] J. Power, J. Hestness, M. S. Orr, M. D. Hill, D. A. Wood, "Gem5-gpu: A heterogeneous cpu-gpu simulator", *IEEE Comput. Archit. L.*, vol. 14, no. 1, pp. 34-36, Jan. 2015.

[28] S. Thoziyoor et al. A comprehensive memory modeling tool and its application to the design and analysis of future memory hierarchies. In ISCA, pages 51-62, 2008.