AI Based Multicore Resource Allocation

Posted on March 12, 2025

Abstract

Asymmetric Multicore Processors (AMP) are becoming used in both advanced and basic computing systems due to their fundamental flexibility and outstanding computing possibilities. Among the subcategories of AMP, Performance Asymmetric Multicore (PAM) Architectures are unique in that they include various micro-architecture cores into a single chip. Nevertheless, the complex interplay between heterogeneous cores and a diverse range of applications presents a formidable challenge in determining the optimal hardware configuration, encompassing core selection and reducing energy consumption for each application. To tackle these multifaceted problems, this paper introduces a pioneering core prediction model known as the Dual branch Recurrent Neural Network (DB-RNN) tailored specifically for AMP. The DB-RNN model encompasses weight sharing facilitated by a hybrid optimization algorithm called African Vulture with Aquila Optimizer (AVAO) and a core prediction module. Additionally, the energy-delay product (EDP) will be evaluated to find the performance of each core. The implementation is performed using the MATLAB platform. In the final phase, DB-RNN dynamically predicts the most suitable cores for individual workloads at runtime, thereby elevating both energy efficiency and overall system performance. This innovative approach paves the way for enhanced efficiency and performance in PAMP systems, offering promising prospects for a wide range of computing applications.

Introduction

Since multi-core processors outperform complicated single-core CPUs in terms of performance per watt and processing capacity, they have gained popularity. While CPU chip manufacturers like AMD and Intel continue to build new CPU chips with more symmetric cores, experts are exploring other multi-core organizations, including AMC designs [1, 2, 3], where each core has varying processing speeds. The potential to improve system efficiency and reduce power consumption is what makes AMC desirable. An application with a diverse mix of workloads can be better served by an AMC architecture since it combines fast cores and slow cores [4,5]. For instance, the serial code parts may be performed on quick, sophisticated cores, whereas parallel number crunching could be done on slow, basic cores, which would consume less power. Additionally, many contemporary multi-core processors have Dynamic Voltage and Frequency Scaling (DVFS) [6,7], which allows for the dynamic adjustment of each core's operating frequency, converting PAM processors from symmetric multi-core devices.

In high-performance computing, the utilization of the AMC model creates an alluring solution to the

power wall. These designs have many processing core types intended to strive for efficiency or power reduction, increasing energy efficiency. Considering the variety of such systems, workload balancing and task allocation become two of the major difficulties [8,9]. Using task-based programming paradigms is one way to tackle these problems. According to the resources' accessibility, tasks are dynamically allocated resources in today's task-based programming paradigms [10, 11, 12]. They also enable the defining of task dependencies, enabling automated scheduling and synchronization decisions made by the runtime system. On an asymmetric system, it is still challenging to map available jobs efficiently to different types of cores despite task-based programming models being a potent method. Task-based parallel applications exhibit several traits that might impact the overall program time, such as challenging tasks with extended critical paths or varying levels of task cost fluctuation. These characteristics incite researchers to develop smart scheduling techniques within a task-based programming framework and speed up the overall application [13].

Task-sharing and task-stealing, on the other hand, assign jobs at random to various cores; this does not pose a problem for symmetric cores but might lead to unequal task distribution among asymmetric cores. While small work might be accomplished by a fast core, protracted work might be allocated to a slowcore, for instance. Unbalanced workloads are a crucial factor in the performance of parallel applications [14,15]. A lot of earlier research has been done to address this issue with the goal of enhancing the performance of parallel programs on AMP. PFT and WATS aim to use their own spawning algorithms to assign work effectively for AMP where the frequencies of cores are constant (referred to as "static asymmetry"). To overcome these drawbacks, this work introduces the DB-RNN to efficiently schedule the workloads. The DB-RNN may be able to decrease both energy usage and energy delay by determining the appropriate hardware configuration for a certain application at runtime.

Multi-objective problems like energy consumption, memory utilization, computation time, and load balancing are evaluated for each core to evaluate the performance of the cores.

To improve the weight function of the DB-RNN model, the African Vulture with Aquila Optimizer is utilized. This optimization is the hybrid version of the African Vulture and the Aquila Optimization algorithm.

To identify the best core for identifying the right cores for each task at runtime, two RNN models called Dual Band Recurrent Neural Network (DB-RNN) are utilized. The outputs of the hidden layers are interconnected; using this, the weight functions are shared among them. This may improve the speed of the prediction model.

To improve the prediction accuracy, the output is given to the self-attention mechanism. The Softmax activation function is used to make the final prediction of EDP.

The organization of the paper is as follows: section 2 discusses the recent existing papers in the literature review, section 3 gives a clear explanation of the proposed methodology, the results obtained for the proposed model are compared with the existing techniques in Section 4, and section

5 gives the detailed conclusion.

Literature Review

In 2020, Kim et al. [16] designed an asymmetric multi-core mobile device with an energy-saving real-time multi-core allocation method. In order to reduce the energy consumption of the mobile device while maintaining real-time functionality for the mobile application, the study proposed an energy-efficient big.LITTLE core allocation mechanism. We use an actual test bed of a ready-made smartphone to show how the recommended multi-core assignment strategy may maximize the energy-saving advantage while assuring real-time performance.

In 2021, Kumar and Vidyarthi [17] suggested a brand-new multi-core system scheduling methodology that uses less energy. Along with efficient core utilization, a scheduler also considers the energy aspect while allocating work to CPU cores. It addresses three issues: thermal management by balancing workload across processing units, asymmetric problem by effectively allocating workload to various processing units (using the AEt concept), and DVFS/DPM problem by implementing the execution unit's use of the DVFS approach for workloads that are CPU and memory units. The suggested scheduler was extensively assessed using a variety of criteria. The model's capacity to scale to an increasing number of processing units was also assessed.

In 2020, Yu et al. [18] introduced the distinctive COLAB scheduling architecture, which prioritises workloads for many threaded programs running on AMPs, which now make up a sizable portion of the processor industry, particularly in embedded systems. It analyses each thread's performance and power on each type of core, identifies patterns of communication, and identifies restriction strands. By using such data, the scheduler was able to organize an allocation of threads and core assignments while still giving each program a reasonable amount of processing time.

In 2021, Mahmood et al. [19] introduced a temporary priority real-time scheduling on AMPs. That study examines the scheduling of power AMP using dynamic-priority semi-partitioning and presents a unique method, EDFwC=D-TS. The task-allocation policy of the EDFwC=DTS algorithm features two rounds. When tasks were allocated, a core was initially given a portion of them, and then in the subsequent round, tasks were divided in order to maximize core utilization. The results of the empirical investigation support the EDFwC=D-TS algorithm's superiority to its rival.

In 2020, Chniter et al. [20] provided a scheduling method that utilizes multi-speed cores and has three possible outcomes, depending on the setup scenarios: job migration, period modification, and task partitioning. An ILP was created to provide an execution model that deals with the distribution of jobs across various cores while consuming the least amount of energy. The work fails to concentrate on an expanded execution model that takes into account mixed types of jobs (sporadic and aperiodic) connected by dependency constraints. Costs of migration and communication, which were frequently prevalent in real-time applications, should also be taken into account.

In 2023, Wu et al. [21] recommended three methods of processor allocation for EDF scheduling for multiprocessor systems with asymmetric performance. The homogenous computing environment was the foundation for the majority of the current task-scheduling algorithm research. The study analyzed how the three earliest deadline-first algorithms perform on asymmetric multiprocessors with various processor allocation methodologies. When using an allocation strategy, high-priority tasks are sent to the processor that is idle the most slowly, providing an effective schedulable analysis.

In 2022, Fang et al. [22] suggested a partitioned cache replacement policy for AMP with heterogeneity awareness. The paper suggests replacing the partition cache with a strategy that is heterogeneous-aware (HAPC); by detecting the various ways that heterogeneous cores access memory differently, HAPC dynamically modifies the weight given to the reuse of cache blocks. That minimizes the impact that small cores have on big cores' memory access behaviour and ensures that cache blocks consumed by powerful cores are not immediately replenished. Dynamic cache partitioning, which has been utilized to manage shared cache in the context of a multi-core CPU, was another successful technique.

In 2020, Gomatheeshwari and Selvakumar [23] utilized deep learning algorithms to properly distribute workloads on PAM architectures. A special core prediction model for AMP called the low-weight deep neural network (LW-DNN) was developed. The proposed LW-DNN consisted of three steps: feature selection, feature optimization, and the main prediction module. In order to increase energy efficiency and performance, the third step forecasts the ideal cores for each activity at runtime. The first two stages collect and optimize workload characteristics using the preprocessing method.

In 2021, Balasekaran et al. [24] used deep learning, an intelligent task-scheduling system for autonomous vehicles. In order to overcome these difficulties, that research resulted in the development of an intelligent task-handling system for IoT-based autonomous vehicles. A managed resource predictor was used to select the best hardware cluster for each task's processing. The earliest hyper period first (EHF) scheduler was used to complete tasks in order to get the greatest efficiency in terms of task error rate and scheduling length. Employing the single-layer feedforward neural network (SLFN) and minimal learning methods, each job was routed to the suitable processor based on its urgency and CPU use.

In 2021, El Sayed et al. [25] introduced real-time task partitioning on multi-core platforms with energy-efficient design. The suggested blocking-aware-based partitioning (BABP) technique seeks to lower overall energy usage while preventing time-outs. The suggested approach is more energy-saving than existing partitioning algorithms. Comparing the proposed approach to well-known heuristic partitioning techniques, a number of tests were conducted to evaluate its performance. The suggested algorithm's efficiency and system scheduler-friendliness were compared to the most popular bin-packing techniques.

2.1. Problem statement

A significant problem in computing is effectively using the processing capacity of current multi-core processors, especially asymmetric ones while reducing energy consumption and guaranteeing real-time job scheduling. The research studies that are now available ([16] to [25]) jointly address this complex issue. Designing energy-efficient core assignment methods (such as big.LITTLE), controlling heat concerns, efficiently allocating workloads to various processing units, and optimizing dynamic voltage and frequency scaling (DVFS/DPM) for different workloads are major issues at the forefront of this challenge. The integration of deep learning algorithms for predicting task-core assignments is also explored in this research, along with scalability, cache management, and performance asymmetry. Intelligent task management and hardware selection have become crucial in the IoT and autonomous systems areas, where this issue also arises. On these complicated multi-core systems, the discipline is actively exploring how to combine energy efficiency, real-time performance, and optimal task scheduling. This will build the groundwork for effective and sustainable computing across a variety of applications and domains.

3. Proposed Methodology

Due to their versatility and good performance, AMP is being utilized more and more in both high-end and low-end computing devices. The variant of AMP known as PAM merges several micro-architecture cores onto a single chip. Finding the ideal core and voltage-frequency pairings for each application's hardware setup is still an NP-hard task due to the diversity of cores and applications. The EDP optimization in these devices is a challenging issue. The execution of various embedded workloads in accordance with the requirements of the proposed architecture is examined in this article. For each job, more than 60 performance counter (PC) measurements are gathered as part of the Mibench test, which regularly executes standard mathematical operations such as matric multiplication and sort algorithms on each cluster.

3.1. Network model

Platform Description: A platform using ARM big-little clustered multicore processors is the subject of the experimental verification. Performance-efficient cores (also known as "big" cores) and power-efficient cores (also known as "Little" cores) are both intended to be incorporated into the same system-on-chip (SoC) on this platform.

Cluster Structure: The platform is divided into clusters, each of which may have a unique mix of large and small cores. Each cluster may have a varied ratio of large and small cores, resulting in various core combinations. As a result, the platform may be easily customized to meet different demands.

Cluster Combinations: The experimental setup takes seven different clusters (core combinations) into

account. These clusters are known by their names: “4L4b, 0L4b, 3L2b, 4L0b, 2L2b, 4L3b, 3L1b.” Both “L” and “b” stand for the cores that are performance- and power-efficient in these names. The power and performance capabilities of each of these clusters probably vary.

A basic arithmetic program like matric multiplication is frequently conducted in each cluster as part of the Mibench benchmark, which also includes the automotive, office, and communications industries. For each task, more than 60 PC metrics are recorded. Additionally, benchmarks for IoMT, which includes advanced encryption standard (AES), sleep apnea detection (update), histogram equalization (imghist), inverse radon transformations (radon), and QRS detection in ECG (sqrt), are also run and analyzed. The benchmark for CoreMark involves a core matrix, a core util conducted through many iterations, and data collection. Each cluster runs these programs, with the identical procedure being followed each time.

3.2. Multi-objective problems

From the input data, the characteristics like IPC, L1 I Cache miss rate (%), L1 D Cache accesses pti, L1 D Cache miss rate (%), L1 I Cache accesses pti, LLC accesses pti, LLC miss rate (%), Branches instructions (%) and Branch miss rate (%) are evaluated. To share the workload among the multiple cores, the multi-objective problems are evaluated by considering the above characteristics from the dataset. By evaluating these parameters, the optimal core for each workload is selected.

3.2.1. Energy consumption

Voltage and frequency squared are directly proportional to energy usage. Because the frequency and voltage are strongly correlated, lowering these two variables will have a considerable impact on lowering energy consumption, and as a result, processors’ dynamic energy consumption (P) will take the form of Eq.

The number of switches in each clock cycle, along with capacitance and voltage, is represented by A in this equation.

3.2.2. Memory utilization

The amount of instructions the core has processed up to completion time is equal to the resource used. Consequently, it is used to its full potential.

3.2.3. Computation time

A workflow’s makespan is its total completion time. The structural design makes it possible to calculate a solution’s makespan using the exit task’s completion time. A workflow’s critical route,

which is also its longest path, can, on the other hand, be used to calculate how long it will take to complete. As a result, the makespan may be defined as the time it takes for the last job on the critical route to complete. To define the start time and end time of the task, we define two functions.

Then, the computation time of a given data to reach the destination is given in Eq. (6),

3.2.4. Load balancing

In multicore systems, load balancing aims to evenly distribute computational tasks over numerous CPU cores to ensure efficient resource utilization. It is essentially an algorithmic and heuristic process, but because it is dynamic and context-dependent, it is challenging to distil it into a single formula. However, load balancing may also be modelled in a more ethereal fashion.

The imbalance in workload allocation between cores may be easily measured using this formula. While a larger number denotes a greater load imbalance, a lower value denotes a more evenly distributed load. By dynamically dividing workloads or work units among cores, load balancing algorithms really seek to reduce this Load Imbalance Index. Keeping all cores as close to the average load as feasible can ensure effective utilization of the available processing power. As noted in the preceding comments, this objective may be accomplished using a variety of load-balancing methodologies and algorithms.

3.3. Optimal Core Prediction using DB-RNN

By properly forecasting, allocating, and carrying out tasks, this prediction model's major goal is to optimize energy use and performance. It is recommended to utilize a VF pair with a minimized optimal EDP rate for each job at runtime or any random combination of them. Based on the features of the workload and the hardware properties, a DB-RNN network was built in this study to anticipate the best configuration ahead of time with a low EDP rate. The AVAO method is used to train RNN in order to improve network performance and increase the accuracy of EDP prediction.

Traditional RNNs either use supervised learning or unsupervised learning. The input sequence data might have a length as long as its depth in this model. An essential feature of the RNN model architecture is a feedback loop that joins each layer and has the ability to retain information from the previous input. Consequently, it might improve the model's dependability. The fact that RNNs do the same task for each element in an order, with the outcome depending on the computations made before, is what makes them recurrent. To put it another way, the RNN acts as a memory for the information that has been computed up to this point.

In this model, the hidden layer with index is called at time, whereas the input layer with index is called. Similarly, the output layer with index is called. Input is connected to a hidden layer with an index by a weight matrix called. The previously concealed and hidden layers with the indexes are

connected by the weight matrix. The number of input units is hidden in the output layer and connected by the weight matrix, which has an index of. The number of concealed units is. The results of the two hidden layers are shared among two RNN models, which can improve the training speed of the prediction model.

A classical neural network may be trained similarly to a DB-RNN. However, there was a subtle change in how this model applied the backpropagation technique. The hidden layers in the structure allow for the sharing of parameters across all time steps; as a result, the gradient at every result depends on both the assessments of the present-time steps and the processing of the previous time steps.

After the output layers of two RNN models, a Self-Attention (SA) layer is presented. The final output of this layer (not the final output of the model) is obtained by employing an attention approach to dynamically generate the weights of many interactions between the input and output of the same layer. The SA mechanism considers how logically connected the top and lower sequences are. By using a linear transformation, three vector sequences are extracted from the DB-RNN model.

The optimal core is predicted using the EDP calculation model. To achieve this, developed an EDP cost function, which is determined as the sum of the IPC cycles of each task and the average power utilized by the total amount of work within all clusters.

Workloads are set up for the projected cluster before runtime based on this forecast. To determine each workload's minimized EDP function, the RNN training model was utilized to repeatedly explore all potential coefficients (configurations).

Result And Discussion

This section evaluates and compares the outcomes of the findings attained using the suggested model with the existing techniques like RNN, GRU, LSTM, MLP and DNN. The Mibench benchmark dataset is taken for implementation [26]. The embedded control systems have processors in them. These processors need knowledge of simple data organization, bit manipulation, input/output, and basic arithmetic operations. Some of the typical applications include airbag controllers, engine efficiency monitors, and sensor systems. An algorithm for sorting and a bit counting check, a form recognition program, and a simple math exam are the tests utilized to evaluate these circumstances. The Matlab platform is used for implementation. The error metrics are used for evaluation.

4.1. Performance metrics

MAE

By averaging all observations, MAE calculates the distance between the findings (the dataset elements) and the regression coefficient predictions.

(20)

MAPE

Utilizing the MAPE formula, demand is divided by the total number of distinct absolute errors (each period separately). It reflects an average of percentage errors.

RMSE

For each data point, get the residual (difference between forecast and reality), the mean of residuals and the norm of residual, and then calculate the RMSE by taking the square root of that mean.

MSE

MSE is used to determine how well forecasts or estimations match actual values. The MSE is measured, and the lower it is, the closer the forecast is to reality. Regression models employ this as a model assessment metric, and a lower value denotes a better fit.

NMSE

The NMSE is derived from the MSE, and the forecast value, the number of summing iterations that are performed, is the actual value, is the forecast value

4.2. Overall comparison of the proposed method

The proposed model is compared with the existing techniques like RNN, Gradient Recurrent Unit (GRU), Long Short-Term Memory (LSTM), Deep Neural Network (DNN) and Multi-Layer Perceptron (MLP) in terms of the above-discussed performance metrics. The comparison is given in Table 1.

Table 1: overall comparison of the proposed technique and the existing techniques

Techniques	MAE	MAPE	RMSE	NMSE	Correlation	R-Square
Proposed	1.2398	3.4157	1.4675	0.0010	0.9978	0.9954
RNN	3.2275	8.7386	3.7392	0.0064	0.9855	0.9699

GRU	5.1838	14.4212	6.0071	0.0167	0.9628	0.9211
LSTM	6.1502	16.8794	7.2833	0.0240	0.9500	0.8834
DNN	5.9711	16.5268	7.4144	0.0274	0.9510	0.8793
MLP	6.9942	19.2368	8.2279	0.0326	0.9361	0.8521

In the provided table, various techniques or models are compared based on several performance metrics. The “Proposed” technique stands out as it exhibits superior predictive accuracy with a low MAE of 1.2398, indicating an average absolute difference of approximately 1.2398 units between predicted and actual values. Additionally, it boasts a MAPE of 3.4157%, implying an average percentage deviation of 3.4157% from actual values. The RMSE for the “Proposed” technique is 1.4675, representing the average magnitude of prediction errors. The exceptionally low NMSE of 0.0010 signifies remarkable accuracy relative to actual values. Furthermore, the high correlation (0.9978) demonstrates a strong positive linear relationship between predictions and actuals, while the R-Square (0.9954) indicates that the model explains approximately 99.54% of the variance in the actual data. Overall, the “Proposed” technique excels in accuracy and explanatory power, making it a compelling choice for the task compared to other models like RNN, GRU, LSTM, DNN, and MLP.

MAE

The MAE statistic calculates the average absolute difference between the expected and actual values. It provides a measurement of the model’s correctness, with lower values indicating a higher degree of accuracy. Typically, lower MAE values are preferred. The MAE values are compared in Figure 2.

Figure 2: Comparison of MAE

The “Proposed” approach has the lowest MAE (1.2398), which means that, on average, its forecasts are 1.24 units off the actual values. In terms of the other approaches, RNN has the second lowest MAE (3.2275), followed by GRU, DNN, LSTM, and MLP, in increasing order of MAE values. This indicates that the “Proposed” strategy often has the fewest prediction errors.

MAPE

The average percentage variance between predicted and actual values is calculated using a statistic known as MAPE. This metric, which is expressed as a percentage, is used to assess the relative

accuracy of the models. Lower MAPE values are preferable. Figure 3 represents the values of the MAPE.

Figure 3: Comparison of MAPE

The technique denoted as “Proposed” exhibits the most minimal Mean Absolute Percentage Error (MAPE) at 3.4157%. This indicates that, on average, the predictions derived from this technique deviate by approximately 3.42% from the actual values. In comparison to the other approaches, RNN demonstrates the second lowest MAPE at 8.7386%, succeeded by GRU, DNN, LSTM, and MLP, in ascending order of MAPE values. This implies that, on average, the “Proposed” technique possesses the lowest percentage of prediction errors.

RMSE

The metric known as RMSE calculates the square root of the mean of the squared differences between the anticipated and observed values. In a manner analogous to MAE, this measure provides a gauge of precision, where smaller values correspond to greater accuracy. The RMSE values for the proposed and the existing techniques are compared in Figure 4.

Figure 4: Comparison of RMSE

The technique labelled “Proposed” demonstrates the smallest overall spread or variability in its predictions, as evidenced by its lowest RMSE value of 1.4675. Following this, the RNN technique exhibits the next lowest RMSE of 3.7392, while GRU, LSTM, DNN, and MLP techniques follow in increasing order of RMSE values.

NMSE

The technique labelled “Proposed” demonstrates the smallest overall spread or variability in its predictions, as evidenced by its lowest RMSE value of 1.4675. Following this, the RNN technique exhibits the next lowest RMSE of 3.7392, while GRU, LSTM, DNN, and MLP techniques follow in increasing order of RMSE values, which are shown in Figure 5.

Figure 5: Comparison of NMSE

The technique denoted as “Proposed” displays the smallest NMSE value of 0.0010, signifying that its predictions exhibit minimal error in comparison to the variability of the true values. A reduced NMSE

value implies a commendable level of accuracy in forecasting. Amongst the alternative techniques, RNN demonstrates the subsequent smallest NMSE value of 0.0064, succeeded by GRU, LSTM, DNN, and MLP, in ascending order according to their NMSE values.

Correlation

This measure evaluates how linearly expected and observed values relate to one another. The correlation coefficient, which ranges from -1 (totally negative correlation) to 1 (absolutely positive correlation), demonstrates this. If the correlation is close to 1, which denotes a substantial positive correlation, then the model's predictions are likely to be closely related to the actual values. The correlation between the proposed and existing techniques is shown in Figure 6.

Figure 6: Comparison of correlation

The "Proposed" method has the greatest correlation (0.9978), demonstrating a very strong positive linear connection between its predictions and the observed data. RNN, followed by DNN, LSTM, GRU, and MLP, in decreasing order of correlation values, has the next-highest correlation among the other methods (0.9855).

R-Square

The R-Square statistic determines how much of the variance in the dependent variable's actual values can be explained by the independent variable's expected values. This parameter's value ranges from 0 to 1, and a greater number indicates that the model and the data are more closely aligned. This table favours higher R-Square values. Figure 7 compares the R-Square error metrics of the discussed techniques.

Figure 7: Comparison of R-Square

The "Proposed" method has the greatest R-Square (0.9954), which demonstrates that it fits the data the best and that it accounts for 99.54% of the variation in the actual values. RNN, followed by GRU, LSTM, DNN, and MLP, in decreasing order of R-Square values, has the next-highest R-Square among the other methods (0.9699).

Table 2: Execution time and energy consumption of the clusters

Core type	Execution time	Energy consumption

4L4B	☒	☒
0L4B	☒	☒
4L3B	☒	☒
4L0B	☒	☒
2L2B	☒	☒
3L2B	☒	☒
3L1B	☒	☒

Conclusion

In conclusion, the advent of AMP and, particularly, PAMP has ushered in a new era of computing characterized by flexibility and remarkable performance capabilities. However, the inherent complexity of managing heterogeneous cores and optimizing them for a diverse array of applications has posed significant challenges. This includes the intricate task of configuring the optimal hardware setup, encompassing core selection and voltage-frequency pairings, all while striving to efficiently optimize the EDP. To address these multifaceted challenges, we have introduced the DB-RNN model, a pioneering solution tailored specifically for AMP.

The DB-RNN framework, incorporating weight sharing through the innovative AVAO hybrid optimization algorithm, has demonstrated its potential to revolutionize core prediction and workload management. This model, implemented using the versatile MATLAB platform, dynamically predicts the most suitable cores for individual workloads at runtime, thereby enhancing energy efficiency and overall system performance. The DB-RNN model offers a promising path forward, one that has the potential to significantly enhance efficiency and performance in PAMP systems. As we continue to explore and refine such cutting-edge approaches, we look ahead to a future where computing applications across the spectrum can benefit from the advanced capabilities and adaptability offered by asymmetric multicore architectures.

Reference

Wei, L., Ning, Z., Quan, L., Wang, A. and Gao, Y., 2022. Research on Parameter Matching of the

Asymmetric Pump Potential Energy Recovery System Based on Multi-Core Parallel Optimization Method. Processes, 10(11), p.2298.

Salami, B., Noori, H. and Naghibzadeh, M., 2020. Fairness-aware energy efficient scheduling on heterogeneous multi-core processors. IEEE Transactions on Computers, 70(1), pp.72-82.

Sustran, Z. and Protic, J., 2021. Migration in hardware transactional memory on asymmetric multiprocessor. IEEE Access, 9, pp.69346-69364.

Assis, Í.A., Fernandes, J.B., Barros, T. and Xavier-De-Souza, S., 2020. Auto-tuning of dynamic scheduling applied to 3D reverse time migration on multicore systems. IEEE Access, 8, pp.145115-145127.

Foadaddini, A., Zolfaghari, S.A., Mahmoodi Darian, H. and Saadatfar, H., 2020. An efficient GPU-based fractional-step domain decomposition scheme for the reaction-diffusion equation. Computational and Applied Mathematics, 39, pp.1-35.

Bratek, P., Szustak, L., Wyrzykowski, R. and Olas, T., 2023. Reducing energy consumption using heterogeneous voltage frequency scaling of data-parallel applications for multicore systems. Journal of Parallel and Distributed Computing, 175, pp.121-133.

Khriji, S., Chéour, R. and Kanoun, O., 2022. Dynamic Voltage and Frequency Scaling and Duty-Cycling for Ultra Low-Power Wireless Sensor Nodes. Electronics, 11(24), p.4071.

Iranmanesh, A. and Naji, H.R., 2021. DCHG-TS: a deadline-constrained and cost-effective hybrid genetic algorithm for scientific workflow scheduling in cloud computing. Cluster Computing, 24, pp.667-681.

Asghari, A., Sohrabi, M.K. and Yaghmaee, F., 2021. Task scheduling, resource provisioning, and load balancing on scientific workflows using parallel SARSA reinforcement learning agents and genetic algorithm. The Journal of Supercomputing, 77, pp.2800-2828.

Leandro Nesi, L., da Silva Serpa, M., Mello Schnorr, L. and Navaux, P.O.A., 2020. Task-based parallel strategies for computational fluid dynamic application in heterogeneous CPU/GPU resources. Concurrency and Computation: Practice and Experience, 32(20), p.e5772.

Elshazly, H., Ejarque, J. and Badia, R.M., 2022. Storage-heterogeneity aware task-based programming models to optimize I/O intensive applications. IEEE Transactions on Parallel and Distributed Systems, 33(12), pp.3589-3599.

Minhas, U.I., Woods, R., Nikolopoulos, D.S. and Karakonstantis, G., 2021. Efficient, dynamic multi-task execution on FPGA-based computing systems. IEEE Transactions on Parallel and Distributed Systems, 33(3), pp.710-722.

Bosch, J., Álvarez, C., Jiménez-González, D., Martorell, X. and Ayguadé, E., 2020. Asynchronous runtime with distributed manager for task-based programming models. Parallel Computing, 97, p.102664.

Peng, B., Yang, M., Yao, J. and Guan, H., 2020. A throughput-oriented nvme storage virtualization with workload-aware management. IEEE Transactions on Computers, 70(12), pp.2112-2124.

Zhang, H., Geng, X. and Ma, H., 2020. Learning-driven interference-aware workload parallelization for streaming applications in heterogeneous cluster. IEEE Transactions on Parallel and Distributed Systems, 32(1), pp.1-15.

Kim, D., Ko, Y.B. and Lim, S.H., 2020. Energy-efficient real-time multi-core assignment scheme for asymmetric multi-core mobile devices. IEEE Access, 8, pp.117324-117334.

Kumar, N. and Vidyarthi, D.P., 2021. A novel energy-efficient scheduling model for multi-core systems. Cluster Computing, 24(2), pp.643-666.

Yu, T., Zhong, R., Janjic, V., Petoumenos, P., Zhai, J., Leather, H. and Thomson, J., 2020. Collaborative heterogeneity-aware OS scheduler for asymmetric multicore processors. IEEE Transactions on Parallel and Distributed Systems, 32(5), pp.1224-1237.

Mahmood, B., Ahmad, N., Khan, M.I. and Akhunzada, A., 2021. Dynamic priority real-time scheduling on power asymmetric multicore processors. Symmetry, 13(8), p.1488.

Chniter, H., Mosbahi, O., Khalgui, M., Zhou, M. and Li, Z., 2020. Improved multi-core real-time task scheduling of reconfigurable systems with energy constraints. IEEE Access, 8, pp.95698-95713.

Wu, P., Li, Z., Yan, T. and Li, Y., 2023. Three Processor Allocation Approaches towards EDF Scheduling for Performance Asymmetric Multiprocessors. Applied Sciences, 13(9), p.5318.

Fang, J., Kong, H., Yang, H., Xu, Y. and Cai, M., 2022. A Heterogeneity-Aware Replacement Policy for the Partitioned Cache on Asymmetric Multi-Core Architectures. Micromachines, 13(11), p.2014.

Gomatheeshwari, B. and Selvakumar, J., 2020. Appropriate allocation of workloads on performance asymmetric multicore architectures via deep learning algorithms. Microprocessors and Microsystems, 73, p.102996.

Balasekaran, G., Jayakumar, S. and Pérez de Prado, R., 2021. An intelligent task scheduling mechanism for autonomous vehicles via deep learning. Energies, 14(6), p.1788.

El Sayed, M.A., Saad, E.S.M., Aly, R.F. and Habashy, S.M., 2021. Energy-efficient task partitioning for real-time scheduling on multi-core platforms. Computers, 10(1), p.10.

Guthaus, M.R., Ringenberg, J.S., Ernst, D., Austin, T.M., Mudge, T. and Brown, R.B., 2001, December.

MiBench: A free, commercially representative embedded benchmark suite. In *Proceedings of the fourth annual IEEE International workshop on workload characterization. WWC-4 (Cat. No. 01EX538)* (pp. 3-14). IEEE.