

# MambaCPU: Enhanced Correlation Mining with State Space Models for CPU Performance Prediction

1<sup>st</sup> Xiaoman Liu

*Data Center and Artificial Intelligence (DCAI)*

*Intel Corporation*

Shanghai, China

xiaoman.liu@intel.com

Personal Homepage

**Abstract**—Forecasting CPU performance, which involves estimating performance scores based on hardware characteristics during operation, is crucial for computational system design and resource management. This research field currently faces two primary challenges. First, the diversity of CPU products and the specialized nature of hardware characteristics make real-world data collection difficult. Second, existing approaches, whether reliant on hardware simulation models or machine learning, suffer from significant drawbacks, such as lengthy simulation cycles, low prediction accuracy, and neglect of characteristic correlations. To address these issues, we first gathered, preprocessed, and standardized historical data from the 4th Generation Intel Xeon Scalable Processors across various benchmark suites to create a new dataset named PerfCastDB. Subsequently, we developed a novel network, MambaCPU (MaC), as the baseline model for the PerfCastDB dataset. This model employs the mamba structure to explore global dependencies and correlations among multiple characteristics. The use of intra- and inter-group attention mechanisms further refines correlations within and between characteristic groups. These techniques enhance MaC’s capability to analyze and mine complex multivariate correlations. Comparative experiments on the PerfCastDB dataset demonstrate that MaC surpasses existing methods, confirming its effectiveness. Additionally, we have open-sourced part of the dataset and the MaC code at <https://github.com/xiaoman-liu/MaC> to facilitate further research..

**Index Terms**—CPU Performance Prediction, State Space, Deep Learning, Mamba

## I. INTRODUCTION

CPU performance prediction based on hardware characteristics is crucial for optimizing computational system design and resource management. It plays a pivotal role in tasks ranging from hardware design to resource allocation, as it allows for forecasting a CPU performance under various operational conditions. This capability is indispensable not only for CPU manufacturers, who seek to streamline the design and prototyping process, but also for consumers and enterprises that need to select the most suitable hardware for their specific workloads. In summary, CPU performance prediction possesses substantial theoretical research value and

practical application significance, representing one of the most valuable research areas.

Numerous studies have focused on understanding and predicting computer system performance. Early work [6], [14], [20], [21] employed statistical and sampling methods to analyze computer performance. Subsequently, research shifted towards using machine learning methods [8], [15], [17], [18], [25] for CPU performance prediction. The explosive growth of deep learning in text [26], speech [10], and image recognition [9] has introduced new avenues for performance prediction [4]. Dibyendu et al. [3] developed the deep neural network SpecNet, achieving high prediction accuracy. Yu Wang et al. [27] demonstrated that deep neural network models significantly outperform traditional linear models in benchmark performance prediction. Michael et al. [23] utilized a long short-term memory (LSTM) model for CPU and GPU performance prediction.

CPU performance prediction remains a challenging field due to several key limitations. First, the collection of real-world data is hindered by the sheer variety of available CPU products, each with highly specialized and diverse hardware characteristics. As a result, the field lacks a unified dataset that can comprehensively cover different hardware configurations and benchmarks. Without such a standardized dataset, it becomes difficult to compare and evaluate the effectiveness of different prediction models across varying CPU architectures. Second, the previous approaches, ranging from hardware simulation models to machine learning-based methods, suffer from notable drawbacks. Hardware simulation models are often computationally expensive, requiring lengthy test cycles that limit their practicality for real-time applications. The machine learning methods offer faster predictions but often struggle with low accuracy and an inability to fully capture the complex correlations between different hardware characteristics. These methods typically treat the features as independent or fail to exploit the intricate dependencies between them, which are critical for accurate performance prediction.

According to the above analysis, we make contributions from two aspects to promote the development of the CPU

performance prediction field. First, we organize a novel dataset for further research. Concretely, we collect the historical CPU benchmark data of the 4th Generation Intel® Xeon® Scalable Processors, including data samples with 83-dimensional the hardware characteristics and the 1-dimensional corresponding performance prediction scores under different benchmark suites. Following the data cleaning, data standardization and feature engineering processes, we can generate standard data instances in the dataset. As a result, we organize a novel CPU performance prediction dataset called PerfCastDB, each instance contains 35 hardware characteristics, and 1 ground truth prediction scores under 6 testing suites. For better understanding of the organized data, we provide a sub-benchmark sample at the link Intel Sapphire Rapids sample. Second, we present MaC, a state-of-the-art model designed to uncover and leverage the complex dependencies between multiple hardware characteristics. The MaC model is built on the Mamba structure [2], [7], [28], [29], which is tailored to mine global dependencies across multivariate data.

In general, the contributions of this article can be summarized as:

- We organize a novel dataset called PerfCastDB, which suits for the CPU performance prediction task. This dataset offers comprehensive coverage of hardware characteristics and performance benchmarks, providing a solid foundation for future research in CPU performance prediction
- We propose a novel network MambaCPU(MaC) under the PerfCastDB dataset. MaC is built on the Mamba structure, which can leverage the complex dependencies and global correlations between multiple hardware characteristics. The intra- and inter-group attention mechanisms are introduced to refine the group-wise correlations. The comprehensively utilization of the character correlations ensures the prediction accuracy and interpretability of the MaC model.
- We compare the proposed MaC with several traditional approaches. The experimental results show that MaC significantly outperforms existing approaches, which evaluates the effectiveness of MaC.

## II. DATASET ORGANIZATION

Data was collected from Sapphire Rapids (SPR), the 4th Generation Intel® Xeon® Scalable Processors based on Intel 7 technology [13], from September 27, 2022, to October 27, 2023. This dataset encompasses various stock-keeping units within the SPR product line and employs multiple testing suites, including SPECrate2017, Memory Latency Checker (MLC), Stream, and High Performance Conjugate Gradients (HPCG). The SPECrate suite [24] is divided into "SPECrate2017\_int\_base" and "SPECrate2017\_fp\_base." MLC [12] provides metrics on local and cross-socket latencies and bandwidth, offering 9 levels of latency and 9 types of bandwidth data. Stream [1] evaluates memory efficiency under various scenarios, while HPCG [5] assesses the performance and efficiency of high-performance computing systems.

We standardize the dataset through five key stages: outlier cleaning, multi-output conversion, feature trimming, feature expansion, and normalization/tokenization, resulting in a clean and high-quality dataset for model training. In outlier cleaning, we apply a z-score filtering method to remove data points with z-scores exceeding a threshold ( $|z| > 3$ ), where  $z = (x - \mu) / \sigma$ . Multi-output conversion consolidates six types of benchmark performance data into a unified 11-dimensional vector, reducing redundancy and improving training efficiency. Feature trimming involves removing redundant and non-significant hardware characteristics to enhance generalizability. Feature expansion enriches memory-related features by extracting detailed specifications using identifiers like "DIMM.PartNo" from manufacturers such as Samsung [22], Hynix [11], and Micron [19]. Finally, normalization and tokenization convert categorical features into numerical tokens and scale numerical features consistently, enhancing training efficiency and prediction accuracy. For detailed data formats, refer to our open-source code [16].

## III. METHOD

### A. State Space Model

State space methods provide a mathematical framework for modeling. In a linear time-invariant system with input  $x(t)$ , state  $h(t)$ , and output  $y(t)$ , the system equations are expressed as:

$$\begin{aligned} h'(t) &= Ah(t) + Bx(t), \\ y(t) &= Ch(t). \end{aligned} \quad (1)$$

where  $A$  is the state matrix influencing the state rate of change,  $B$  is the input matrix affecting the state, and  $C$  is the output matrix determining the output from the state. Mamba has discretized  $A$  and  $B$  into a structured state space model by a timescale parameter  $\Delta$ , the tranformation by zero-order hold (ZOH) can be defined as follows:

$$\begin{aligned} \bar{A} &= \exp(\Delta A), \\ \bar{B} &= (\Delta A)^{-1}(\exp(\Delta A) - I) \cdot \Delta B. \end{aligned} \quad (2)$$

Let  $\bar{A} = e^{A\Delta}$ ,  $\bar{B} = (\Delta A)^{-1}(e^{A\Delta} - I)\Delta B$ , then the previous equations can be transformed into the following form:

$$\begin{aligned} h(t_{k+1}) &= \bar{A}h(t_k) + \bar{B}x(t_k) \\ y(t_{k+1}) &= Ch(t_{k+1}). \end{aligned} \quad (3)$$

Finally, through a global convolution operation, the output can be calculated as follows:

$$\begin{aligned} \bar{K} &= (\bar{CB}, \bar{CAB}, \dots, \bar{CA}^k \bar{B}, \dots) \\ y(k) &= x * \bar{K} \end{aligned} \quad (4)$$

### B. Feature Division Module

In this module, input in Figure.1 features are categorized into numerical and character features. Mamba blocks are then

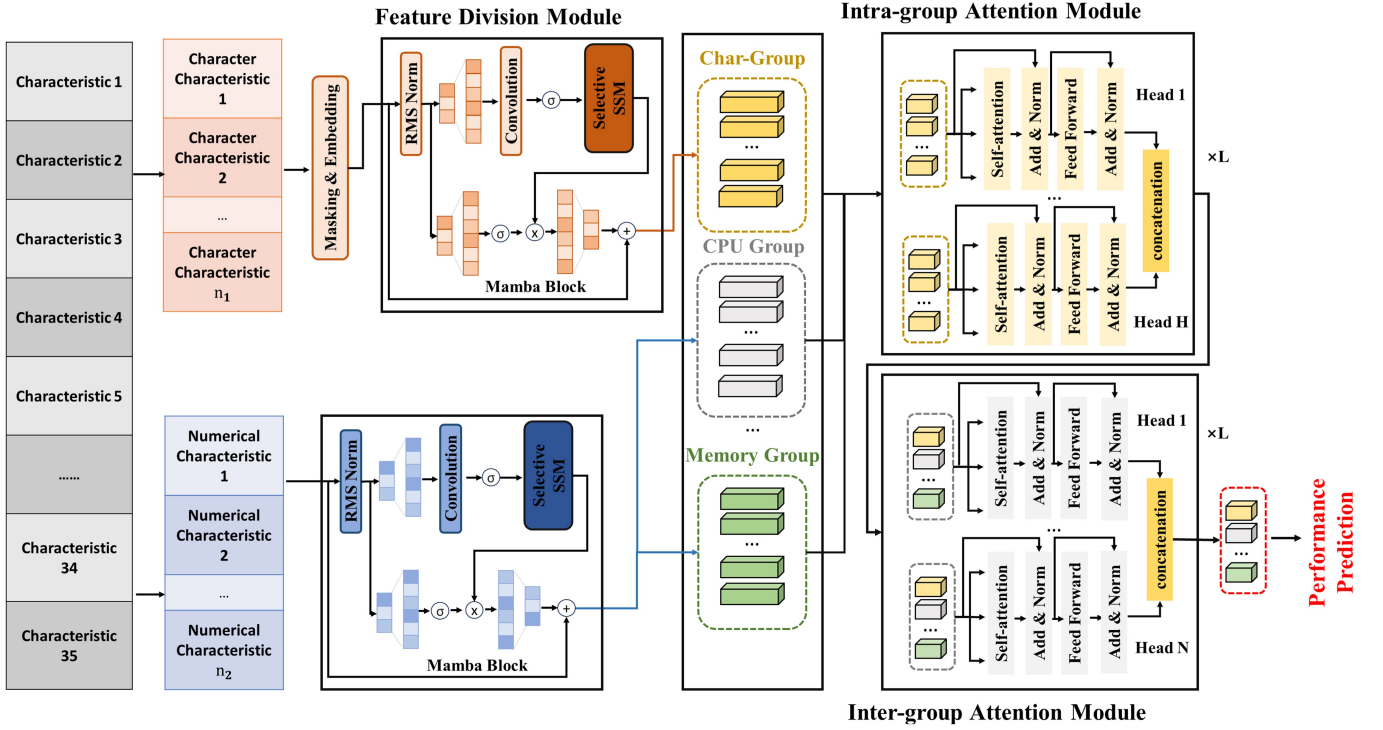


Fig. 1: MaC Architecture: Feature Extraction from Character and Numerical Inputs Using a Selective state Space with Mamba Block and Multiple Attention Mechanisms

used for feature extraction from both categories, which is defined as follows:

$$\begin{aligned}
 \text{RMS Norm: } x_1^{(n)} &= \text{Proj}(\text{RMSNorm}(x^{(n)})) \\
 \text{Convolution: } x_2^{(n)} &= \text{Conv}(x_1^{(n)}) \\
 \text{Selective SSM: } x_3^{(n)} &= \sigma(\text{SSM}(x_2^{(n)})) \\
 \text{Projection: } x_4^{(n)} &= \text{Proj}(x_3^{(n)}) \\
 \text{Multiplication: } x_5^{(n)} &= x_4^{(n)} \cdot \sigma(x_1^{(n)}) \\
 \text{Addition: } y^{(n)} &= x_5^{(n)} + x^{(n)}
 \end{aligned} \tag{5}$$

where  $x^{(n)}$  represents the  $n$ -th input, Proj denotes linear projection, Conv is convolution,  $\sigma$  is the activation function, SSM stands for the state space model, and  $y^{(n)}$  is the  $n$ -th final output.

### C. Group Attention Module

Features from the Feature Division Module are categorized into Char, CPU, Memory, and Other groups for specialized processing. X is the attention input, the intra group attention mechanism can be expressed by:

$$\begin{aligned}
 \text{Self-Attention} &= \text{softmax} \left( \frac{Q^{(l)}(K^{(l)})^T}{\sqrt{d_k}} \right) V^{(l)} \\
 \text{head}_i^{(l)} &= \text{Self-Attention}(Q_i^{(l)}, K_i^{(l)}, V_i^{(l)}) \\
 \text{MultiHead}^{(l)} &= \text{Concat}(\text{head}_1^{(l)}, \dots, \text{head}_h^{(l)}) W_O^{(l)}
 \end{aligned} \tag{6}$$

$W_Q^{(l)}$ ,  $W_K^{(l)}$ , and  $W_V^{(l)}$  are trainable weight matrices for layer  $l$ , with  $d_k$  as the embedding dimension. The outputs of the

self-attention heads are concatenated and linearly transformed. The inter-group attention mechanism mirrors the intra-group mechanism, using the output of the intra-group attention module. The final output is fed into a dense layer with units equal to the number of benchmarks.

### D. Loss Function

In this study, we utilize the huber loss instead of Mean Absolute Error (MAE) or Mean Squared Error (MSE) due to its robustness to outliers. The formula of huber loss can be expressed by:

$$\text{huber loss} = \begin{cases} \frac{1}{2} \times (y - \hat{y})^2 & \text{for } |y - \hat{y}| \leq \delta \\ \delta \times (|y - \hat{y}| - \frac{1}{2}\delta) & \text{otherwise} \end{cases} \tag{7}$$

where  $y$  is the true value,  $\hat{y}$  is the predicted value, and  $\delta$  is a hyperparameter that determines the threshold for the transition between the MAE and MSE loss functions.

## IV. EXPERIMENTS

### A. Data Splitting and Cross-Validation

We collect data across six suites, dividing it into 60% for training, 20% for validation, and 20% for testing, as detailed in Table I. To improve prediction, we fine-tuned MaC model parameters for each suite. Using 5-fold cross-validation, we based the final evaluation on the average results, which helps prevent overfitting and optimizes hyperparameters for better generalization.

TABLE I: the data distribution for the training, validation, and testing sets on the perfcasdb dataset under different suites.

Suite Name	TrainSet	ValSet	TestSet
SPECrate2017 Integer base	816	204	254
SPECrate2017 FP base	756	189	236
Memory Latency Checker Latency	140	35	42
Memory Latency Checker Bandwidth	922	230	294
Stream	2858	714	897
HPCG	2815	704	886

### B. Evaluation Metrics

We utilize the MAE, MSE, and mean absolute percentage error (MAPE) as the evaluation metrics. These metrics can be defined as the following equations:

$$\begin{aligned}
 \text{MAE} &= \frac{1}{n} \sum_{i=0}^n |x_i - \hat{x}_i| \\
 \text{MSE} &= \frac{1}{n} \sum_{i=1}^n (x_i - \hat{x}_i)^2 \\
 \text{MAPE} &= \frac{1}{n} \sum_{i=1}^n \left| \frac{x_i - \hat{x}_i}{x_i} \right|
 \end{aligned} \tag{8}$$

where  $x_i$  is the ground truth,  $\hat{x}_i$  is the predicted value, and  $n$  is the sample size. MAE quantifies average error in output units, MSE is sensitive to outliers, and MAPE offers percentage-based error interpretation.

### C. Implementation Details

We conducted our experiments on NVIDIA GeForce RTX 4090 GPU using the Adam optimizer with an exponential decay learning rate strategy. Our initial learning rate was set at 0.001, utilizing a batch size of 1, and we trained the model over 300 epochs.

### D. Ablation Study

In this subsection, we conduct the ablation study to demonstrate the effectiveness of each part in MaC. Specifically, the main components in MaC are mamba blocks and attention blocks. The results in Table. II indicate that the mamba-based MaC method, which combines intra-group and inter-group attention, performs best across all metrics.

TABLE II: Evaluating the impact of mamba block and inter-group & intra-group attention mechanisms on model accuracy

Models	MAE	MAPE	MSE	Median SE
Mamba	7.23	1.51%	226.81	15.42
Mamba + intra-group attention	8.63	1.70%	349.33	24.49
MaC	<b>5.90</b>	<b>1.18%</b>	<b>174.60</b>	<b>10.59</b>
MaC w/o char-group	6.51	1.31%	184.71	12.83
MaC w/o mem-group	6.01	1.21%	121.52	12.09
MaC w/o cpu-group	6.22	1.23%	181.86	11.47
MaC w/o other-group	7.23	1.43%	201.94	15.32

### E. Hyperparameters Study

In this subsection, we conduct the comparison experiments on different settings of important hyperparameters, focusing on the size of  $\Delta$  projection  $S$ , the number of attention layers

$L$ , and the weight of huber loss  $\delta$  on MAE, MSE, Median SE and MAPE. Table. III presents the corresponding results on "SPECrate2017\_FP\_base" suite. The results show that the combination of proj=2, state=8, layer=3, head=4 and  $\delta=10$  performs best across almost all metrics.

When proj=4 and state=8, the MSE reached a group minimum of 102.54, yet the MAE, Median SE, and MAPE were not minimized. This suggests that low MSE alone does not guarantee superior predictive performance, necessitating the consideration of detailed SE metrics. As shown, the Median SE for this setup exceeds that of proj=2. Additionally, SEs at the 75%, 90%, and 95% percentiles were higher than those for proj=4, though these are not presented due to space constraints. This is due to MSE's sensitivity to outliers present in the test set. Therefore, in contexts like CPU performance prediction, where outliers exist, careful metric selection is essential, as MSE alone is inadequate.

TABLE III: Optimization of specrate2017 fp hyperparameters on MAE, MSE, Median SE, and MAPE. The parameters proj, state, layer, and head correspond to the expansion factor, state vector dimension size, number of attention layers, and number of attention heads, respectively.

Hyperparameter	Value	MAE	MSE	Median SE	MAPE
proj=S, state=8	1	6.79	186.47	15.20	1.38%
	<b>2</b>	<b>5.90</b>	174.60	<b>10.59</b>	<b>1.18%</b>
	4	6.01	<b>102.54</b>	12.88	1.21%
	8	6.48	144.43	15.31	1.33%
	16	6.40	196.72	14.74	1.30%
proj=2, state=N	1	7.03	164.32	17.27	1.50%
	2	6.47	175.96	13.83	1.32%
	4	6.82	169.95	17.86	1.44%
	<b>8</b>	<b>5.90</b>	174.60	<b>10.59</b>	<b>1.18%</b>
	16	6.15	<b>106.86</b>	14.81	1.27%
layer=L, head=4	1	7.17	319.77	13.68	1.45%
	2	6.12	136.69	12.52	1.25%
	<b>3</b>	<b>5.28</b>	<b>65.02</b>	<b>12.24</b>	<b>1.08%</b>
layer=3, head=H	1	6.17	115.57	15.11	1.27%
	2	6.22	223.80	<b>10.92</b>	1.25%
	3	5.88	89.13	13.55	1.19%
	<b>4</b>	<b>5.28</b>	<b>65.02</b>	12.24	<b>1.08%</b>
	5	5.87	78.60	14.62	1.20%
$\delta$	0.1	7.22	194.87	17.42	1.50%
	1	6.95	204.39	15.58	1.41%
	<b>10</b>	<b>5.28</b>	<b>65.02</b>	<b>12.24</b>	<b>1.08%</b>

### F. Comparison of Prediction Performance

We compare our model with various methods, including machine learning and deep learning techniques. Lasso and Ridge regression address multicollinearity, with ElasticNet combining their strengths. SVM performs well in high-dimensional spaces, while XGB handles missing data and provides feature importance. In deep learning, LSTM and GRU excel at learning long-term dependencies. Next, we present experimental results comparing the performance of our MaC model with these baseline methods. The corresponding results are shown in Table.IV. As a result, MaC achieves the best evaluation results on most suites, which directly reflects the superiority of it.

TABLE IV: Comparison performance across six benchmarks, including SPECrate2017 Integer base (S-Int), SPECrate2017 FP base (S-FP), Stream, HPCG, Memory Latency Checker Latency (MLC-L), Memory Latency Checker Bandwidth (MLC-B).

Models	S-Int	S-FP	Stream	HPCG	MLC-L	MLC-B
MAE ↓						
Lasso	9.49	11.45	12.92	4.01	3.68	3.56
Ridge	11.61	16.47	11.91	3.32	4.05	4.92
EN	23.32	22.24	28.52	5.21	5.61	6.94
SVM	21.53	19.06	9.31	3.13	4.18	3.52
XGB	7.81	6.29	4.48	3.31	3.66	<b>1.24</b>
LSTM	31.68	30.21	31.10	2.84	7.92	8.88
GRU	9.70	8.47	4.17	5.42	3.45	8.89
MaC	<b>7.29</b>	<b>5.28</b>	<b>1.99</b>	<b>2.44</b>	<b>2.67</b>	1.36
MAPE ↓						
Lasso	2.42	2.50	3.20	5.74	3.49	4.63
Ridge	2.86	3.73	2.93	4.61	3.28	5.02
EN	5.94	4.70	7.03	7.87	5.68	8.78
SVM	5.44	3.68	2.44	4.38	3.81	3.01
XGB	1.99	1.32	1.08	4.30	3.05	<b>0.87</b>
LSTM	7.80	6.13	7.68	3.78	7.39	9.41
GRU	2.59	1.77	1.05	8.22	3.38	9.41
MaC	<b>1.89</b>	<b>1.08</b>	<b>0.48</b>	<b>3.10</b>	<b>2.81</b>	1.09
MSE ↓						
Lasso	339.86	471.17	365.63	33.27	32.42	48.80
Ridge	620.16	995.37	446.59	31.03	40.73	814.41
EN	1160.41	868.87	1137.69	66.36	59.45	170.98
SVM	1520.52	902.13	322.29	21.16	54.64	149.03
XGB	<b>233.81</b>	97.11	98.48	22.61	43.75	<b>12.09</b>
LSTM	2027.33	1577.07	1378.09	12.86	113.81	377.25
GRU	1133.63	164.96	403.15	73.49	41.68	377.15
MaC	240.55	<b>65.02</b>	<b>48.45</b>	<b>11.52</b>	<b>26.30</b>	16.75

In Figure.2, we present the heatmaps for attention weights within and between feature groups. Figure.2a illustrates the intra-group attention for the CPU group, comprising 20 feature sets, where lighter colors indicate stronger correlations. Figure.2b displays the inter-group attention across memory, CPU, char, and other groups. These patterns highlight the capability of the model to effectively capture and represent feature interactions, demonstrating the ability of attention mechanism to discern complex relationships.

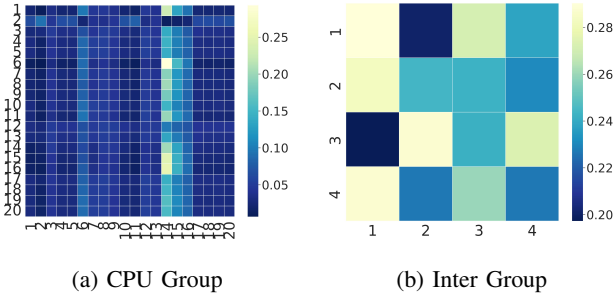


Fig. 2: Attention Matrix Visualization in CPU Intra-groups and Inter-group of the First Head for a SPEC CPU 2017 FP Data Sample

## V. CONCLUSION

In this paper, we introduce PerfCastDB, a comprehensive benchmark dataset specifically designed for CPU performance prediction tasks. This dataset refines historical data to create a model-ready training set. We also present MaC as a baseline model, developed using mamba state-space equations combined with group attention mechanisms to enhance prediction accuracy. Extensive experiments demonstrate that MaC surpasses traditional methods, validating its effectiveness. Furthermore, employing diverse metrics is essential for accurately evaluating CPU performance prediction, particularly in the presence of outliers. We aim for our work to establish a robust foundation for future research in CPU performance prediction.

## REFERENCES

- [1] AMD, “Stream benchmark,” <https://www.amd.com/en/developer/zen-software-studio/applications/spack/stream-benchmark.html>.
- [2] T. Dao and A. Gu, “Transformers are ssms: Generalized models and efficient algorithms through structured state space duality,” *arXiv preprint arXiv:2405.21060*, 2024.
- [3] D. Das, P. Raghavendra, and A. Ramachandran, “Specnet: Predicting spec scores using deep learning,” in *Companion of the 2018 ACM/SPEC International Conference on Performance Engineering*, 2018, pp. 29–32.
- [4] J. Dean, D. Patterson, and C. Young, “A new golden age in computer architecture: Empowering the machine-learning revolution,” *IEEE Micro*, vol. 38, no. 2, pp. 21–29, 2018.
- [5] J. Dongarra, M. A. Heroux, and P. Luszczek, “A new metric for ranking high-performance computing systems,” *National Science Review*, vol. 3, no. 1, pp. 30–35, 2016.
- [6] P. Ein-Dor and J. Feldmesser, “Attributes of the performance of central processing units: A relative performance prediction model,” *Communications of the ACM*, vol. 30, no. 4, pp. 308–317, 1987.
- [7] A. Gu and T. Dao, “Mamba: Linear-time sequence modeling with selective state spaces,” *arXiv preprint arXiv:2312.00752*, 2023.
- [8] G. Hamerly, E. Perelman, J. Lau, B. Calder, T. Sherwood, and H. Hirsh, “Using machine learning to guide architecture simulation,” *Journal of Machine Learning Research*, vol. 7, no. 2, 2006.
- [9] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [10] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups,” *IEEE Signal processing magazine*, vol. 29, no. 6, pp. 82–97, 2012.
- [11] S. hynix, “Sk hynix dimm details,” <https://product.skhynix.com/>.
- [12] Intel, “Intel memory latency checker v3.11,” <https://www.intel.com/content/www/us/en/developer/articles/tool/intel-memory-latency-checker.html>.
- [13] —, “Products formerly sapphire rapids,” <https://ark.intel.com/content/www/us/en/ark/products/codename/126212/products-formerly-sapphire-rapids.html>.
- [14] B. C. Lee and D. M. Brooks, “Illustrative design space studies with microarchitectural regression models,” in *2007 IEEE 13th International Symposium on High Performance Computer Architecture*. IEEE, 2007, pp. 340–351.
- [15] K. P. Lin, “Application of least-squares support vector regression with pso for cpu performance forecasting,” *Advanced Materials Research*, vol. 630, pp. 366–371, 2013.
- [16] MaC, “Characteristic description,” [https://github.com/xiaoman-liu/MaC/blob/main/data/raw/SPR/characteristic\\_description.md](https://github.com/xiaoman-liu/MaC/blob/main/data/raw/SPR/characteristic_description.md).
- [17] P. Malakar, P. Balaprakash, V. Vishwanath, V. Morozov, and K. Kumar, “Benchmarking machine learning methods for performance modeling of scientific applications,” in *2018 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)*. IEEE, 2018, pp. 33–44.

- [18] A. Mankodi, A. Bhatt, B. Chaudhury, R. Kumar, and A. Amrutiya, "Evaluating machine learning models for disparate computer systems performance prediction," in *2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*. IEEE, 2020, pp. 1–6.
- [19] Micron, "Micron dimm details," <https://www.micron.com/>.
- [20] S. Nussbaum and J. E. Smith, "Modeling superscalar processors via statistical simulation," in *Proceedings 2001 International Conference on Parallel Architectures and Compilation Techniques*. IEEE, 2001, pp. 15–24.
- [21] R. H. Saavedra and A. J. Smith, "Analysis of benchmark characteristics and benchmark performance prediction," *ACM Transactions on Computer Systems (TOCS)*, vol. 14, no. 4, pp. 344–384, 1996.
- [22] Samsung, "Samsung dimm details," <https://semiconductor.samsung.com/dram/module/>.
- [23] M. Siek, "Benchmarking cpu vs. gpu performance in building predictive lstm deep learning models," in *AIP Conference Proceedings*, vol. 2510, no. 1. AIP Publishing, 2023.
- [24] SPEC, "Spec cpu 2017," <https://www.spec.org/cpu2017/Docs/overview.html#benchmarks>.
- [25] A. Tousi and M. Luján, "Comparative analysis of machine learning models for performance prediction of the spec benchmarks," *IEEE Access*, vol. 10, pp. 11 994–12 011, 2022.
- [26] A. Vaswani, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017.
- [27] Y. Wang, V. Lee, G.-Y. Wei, and D. Brooks, "Predicting new workload or cpu performance by analyzing public datasets," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 15, no. 4, pp. 1–21, 2019.
- [28] W. Yu and X. Wang, "Mambaout: Do we really need mamba for vision?" *arXiv preprint arXiv:2405.07992*, 2024.
- [29] L. Zhu, B. Liao, Q. Zhang, X. Wang, W. Liu, and X. Wang, "Vision mamba: Efficient visual representation learning with bidirectional state space model," *arXiv preprint arXiv:2401.09417*, 2024.