

RESEARCH ARTICLE

Characterization and Machine Learning Classification of AI and PC Workloads

FADI N. SIBAI¹, ABU ASADUZZAMAN², (Senior Member, IEEE),
AND ALI EL-MOURSY³, (Senior Member, IEEE)

¹Department of Electrical and Computer Engineering, Gulf University for Science and Technology, Mubarak Al-Abdullah 32093, Kuwait

²Electrical and Computer Engineering Department, Wichita State University, Wichita, KS 67260, USA

³Electrical and Computer Engineering Department, University of Sharjah, Sharjah, United Arab Emirates

Corresponding author: Fadi N. Sibai (sibai.f@gust.edu.kw)

ABSTRACT To better design AI processors, it is critical to characterize artificial intelligence (AI) workloads and contrast them to normal personal computer (PC) workloads. In this work, we profiled the AIBench and PassMark PerformanceTest benchmarks with the Intel oneAPI VTune Profiler on a multi-core computer. We captured and contrasted the various CPU and platform metrics and event counts for these two distinct benchmarks. Using the Orange 3.0 data mining tool, and based on the captured profile metrics and event counts, we then trained and tested 9 machine learning (ML) models to classify the CPIs and elapsed times of the various tests of these two benchmarks, including inference and training tests in AIBench, and CPU, memory, graphics, and disk tests in PassMark. The linear regression machine learning model emerged as the best clocks per instruction (CPI) classifier, while the neural network model with 4 hidden layers was the best elapsed time classifier. This machine learning classification can help in predicting the CPI and elapsed time and distinguish between AI and standard PC workloads based on the profiled application(s) and captured profile metrics and event counts. The stressed computer units identified by this detailed profiling work and exercised by the benchmark tests can also guide future AI processor design improvements.

INDEX TERMS AI workloads, Tensorflow, PassMark PerformanceTest, AIBench, workload characterization, event counts, benchmark profiling, machine learning classification, VTune.

I. INTRODUCTION

The amount of research devoted to artificial intelligence (AI), machine learning (ML), and deep learning (DL) has accelerated. Applications of AI, ML and DL have spanned a wide range of fields including health, engineering, business, agriculture, and arts. A critical factor in successful AI deployment is the performance of the computing infrastructure. Computer benchmarks were created to assess the performance of computers. Computer performance has relied on benchmarks such as 3DMark [1], [2], SPEC [3], PCMark [4], [5], and Open Source Mark (OSMark) benchmarks [6]. These benchmarks attempt to stress the components of computers such as the CPU, memory system, or I/O system. Other performance studies have focused on narrower subjects such as the performance of the cache hierarchy [7], performance

impact of cache locking schemes [8] and thread scheduling and migration [9], or performance comparisons of libraries such as MPI vs Pthread [10], or OpenCL vs CUDA [11]. PerformanceTest 11.0 (PassMark) [12] is a computer benchmark similar to PCMark which measures the performances of the CPU, memory, 2D and 3D graphics, and the hard disk.

More recently, the performance of deep learning and machine learning systems has caught the attention of several researchers to characterize workloads, identify performance bottlenecks in hardware and software stacks, and assess the performance gains due to various accelerators. AI benchmarks include DawnBench, ParaDNN, HPL-AI, MLPerf, and AI Benchmark (AIBench). MLPerf [13], [14] includes seven benchmarks for training and five benchmarks for inference. Such benchmarks include workloads for image classification, translation, face recognition, image compression, recommendations, speech recognition, as well as other DL-related tasks. AIBench 0.1.2 [15], [16] encompasses 19 tests with training

The associate editor coordinating the review of this manuscript and approving it for publication was Antonio J. R. Neves¹.

and testing versions and reports the elapsed time with error. AIBench is free of cost unlike other benchmarks. It is the intent of our work to characterize the AIBench workload on a modern multi-core Intel Haswell processor, and compare it the PassMark benchmark workload, which is considered to be a standard computer workload. Although today's AI workloads are run on computer platforms equipped with GPUs, it is the intent of this work to run the benchmarks on a single AI-capable processor for personal computers, without any AI co-processor assistance, to pinpoint weaknesses and identify areas of future improvements. In this work, the methodology we followed consisted of these steps:

1. The AIBench and PassMark benchmarks were run on a modern PC and the overall and subtest times were captured;
2. The components of the central processing unit (CPU) and computer platform were monitored using the Intel oneAPI's VTune profiler 2023.1, and a list of events with event counts were captured for both benchmarks, allowing the stressed units to be identified. VTune is an Intel-based profiler providing analysis of algorithm, parallelism, platform, processor microarchitecture, and accelerator analyses. It employs statistical sampling to update a large variety of performance counters. This exercise allows one to contrast the signatures of AI DL workloads (AIBench) and standard computer workloads (PassMark) on a modern multi-core PC. Following the identification of stressed units, design, circuit, or firmware implementation enhancements conducted on computer units which are more frequently stressed, or particularly, which have reached their top performance capabilities and stalled earlier units in the pipeline, often results in higher overall speedups and likely raises user satisfactions. In that sense, the results of our study can help identify AI workload bottlenecks [34] and consequently guide future design enhancements of AI-capable processors.
3. Also using VTune, the multiple parallel thread activities were captured and visualized to assess the threading performance.
4. Next, Orange 3.0 [17] was used to generate the correlation coefficients of all the captured events for the two combined benchmark suites with the number of clocks per instruction (CPI) and the elapsed time. Orange is a graphical data mining tool supporting various machine learning models and providing a variety of data visualization functions.
5. Also, with the help of the Orange tool, nine ML models were optimized to classify the clocks per instruction (CPI; model target) and then the elapsed time based on the captured events (model features). The nine models are random forest (RF), support vector machine (SVM), k-nearest neighbor (kNN), decision tree (Tree), linear regression (LR), stochastic gradient descent (SGD), AdaBoost, and gradient boosting (GB). The best CPI and elapsed time ML classifiers were identified.

Our work differs from prior work in its focus on profiling and correlating the AIBench and PassMark benchmarks in order to contrast these two workloads and identify performance bottlenecks, and in its use of machine learning models to classify the CPI and elapsed time based on the various performance events in order to predict the performance from the captured events. Following the capturing of events of a workload, the captured performance event signature can help distinguish between AI and non-AI workloads. The workload's performance can be predicted assisted by the machine learning models to decide for instance whether to activate hardware units kept dormant for power saving in order to meet performance demands, or for capacity planning in AI clouds. Another key contribution of this study lies in its ability to integrate insights from both data-centric and model-centric AI perspectives. By leveraging benchmark datasets and employing machine learning models for performance classification, the research offers a comprehensive understanding of AI workload characteristics and hardware utilization patterns. This integrated approach enables not only the analysis of performance metrics but also the prediction of system behavior, thus facilitating optimization efforts for hardware design and system efficiency in handling AI tasks. In essence, this study exemplifies the synergy between data-centric and model-centric AI methodologies [35], showcasing a holistic approach towards advancing the field of AI performance analysis and optimization on modern computing platforms.

The paper is organized as follows. Section II provides some background information from the literature review. Section III discusses AIBench tests and their runs on an Intel core i5 quadcore platform. Section IV presents the PassMark tests and reports the results of running PassMark on the same platform. In Section V, we present the AIBench and PassMark benchmark profiling results obtained with the oneAPI VTune profiler including percentages of time spent in various units and detailed event counts. In Section VI, the two benchmarks are characterized via various graphically plots, and the captured event counts are correlated with both the CPI and the elapsed time, in order to identify the most impactful events on the performance. Section VII presents the ML-based classification of the CPI and the elapsed time with the 9 ML models. The paper concludes in Section VIII.

II. BACKGROUND

In [18], the AIC-Bench methodology for selecting workloads to benchmark AI processor performance based on their computational and memory operation intensities was described. The operational intensity is the ratio of computational operations (number of multiply-accumulate, add and divide, and compare in pooling, softmax, normalization, and convolution operators) over the number of memory accesses. FCN and VGG and Inception deep learning neural networks scored higher operational intensities, whereas ResNet and VOCNet scored the lowest operational intensities. Of all the 2036 operators in 8 sub-benchmarks, 79% of them were

found to be memory-intensive, in particular ReLU, Scale and BatchNorm. The convolution operators were found to be, in the majority, 92%-99% compute-intensive, and consequently with the highest operational intensities, rising more with window size. In the investigated deep neural networks (VGG, ResNet, Inception, SqueezeNet, DenseNet), between 70%-80% of the workloads were memory-intensive (due to the inner product operations on VGG and VOCNet, or due to convolution on Inception and SqueezeNet), and the rest were found to be CPU-intensive. The inner product operator had the highest number of memory accesses and among the top number of computational operations. Therefore, its operational intensity was 1, also reached by Tanh, ReLU, and Scale. Flatten, crop, and concat operators were found to be mainly memory-intensive with little computations. The GPU and Cambricon machine learning unit (MLU) outperformed the CPU on deep learning workloads. It was found that the CPU was not effective in compute-intensive operators, and the GPU underperformed the MLU in such operations. The MLU was found to underperform on memory-intensive operators.

The MLPerf benchmark [13], [14] is a widely used benchmark used in evaluating AI platforms with 7 tests spanning image recognition, object detection, object segmentation, recurrent translation with GNMT model, non-recurrent translation with Transformer model, recommendation, and reinforcement learning. Some AI benchmarks may suffer from high cost, scalability (due to fixed problem scaling), and repeatability. Specifically, regarding repeatability, training DL networks encompasses random factors in: model initialization, data augment, data shuffle, dropout and other stages. In particular, the 3D face recognition, image to text translation, text summarization and compression tests encompass more randomness than the other tests. It is important to select an AI benchmark free of the above weaknesses for detailed studies.

Tang et al. [19] implemented nineteen AI tasks with DL models and concluded that AIBench Training v1.1 outperformed MLPerf Training(v0.7 in terms of benchmark representativeness, computational cost, convergence rate, computation, distinctive memory access patterns, and number of hotspot functions. On AIBench, the tensor processing units (TPUs) outperformed the GPU, although the GPU has better model support. The specification, source code, and performance numbers are available from the AIBench homepage [20]. Notably, to MLPerf Training's text and image data, AIBench added 3D, audio and video data. MLPerf had 30 hotspot functions, to only 9 for MLPerf. In Tang et al's study, MLPerf converged in up to 96 epochs, compared to the MLPerf which converged in up to 49 epochs. In terms of millions of floating-point operations per second (MFLOPS) range covered, AIBench's range of 0.09-282830 outsized MLPerf's range of 0.213248-24500 MFLOPs. From the perspective of model complexity, the number of learnable parameters of AIBench learned 0.03-68.4 million parameters, a larger range than MLPerf's range of 5.2-49.53 million parameters. The authors [19] found the TPU to outperform

the GPU on image classifications. The general AIBench benchmark suite added real-world application scenarios from Datacenter, HPC, IoT, and Edge, into the scenario, training, inference, micro, and synthetic benchmarks. In particular, the AIBench Training and AIBench Inference benchmarks [15], which we focus on in this work, cover nineteen diverse AI tasks with DL models spanning image classification and generation, object detection, text to text and image to text translations, to speech recognition, and text summarization. Object detection and 3D object reconstruction consumed top FLOPs, while learning-to-rank generated the lowest FLOPs. Image-to-text was the most complex model, while the spatial transformer was the least. Text-to-text translation required the most epochs to converge. Both AIBench and MLPerf supported five optimizers, but AIBench supported 14 loss functions, while MLPerf supported only 6. The authors [19] concluded that AIBench captured distinct computation and memory patterns under different scenarios and covered a wider range of DL applications while costing less than MLPerf. Scores of runs of AIBench on a number of GPUs with CUDA and cudNN can be found in [21].

Wang et al. [22] compared the performances and energy efficiencies of CPU, GPU and TPU on DL workloads including convolutional neural networks (CNNs), recurrent neural networks (LSTM), Deep Speech 2, and Transformer. The compute-intensive operators in high-throughput kernels included matrix multiply and 2D convolution heavily used during DL training. The low-throughput kernels required serial operations and longer training times. Key learnings were that matrix multiplication operations were better optimized than convolution on the Intel CPU constrained by peak FLOPS and data fetching. The Tensor cores were underutilized on the GPU during training. Also, the TPU 2 to TUP 3 transition witnessed a much larger improvement in peak FLOPS than memory bandwidth.

A review of novel AI accelerators from SambaNova, Cerebras, Graphcore, and Groq and their performance evaluation can be found in [23]. The paper mentioned several benchmarks for characterizing the performance of ML workloads, including Deep500, HPCAI500, HPL-AI, DAWNbench, DeepBench, Fathom, ParaDNN, HPE DLBS, XSP, Mahon, as well as MLPerf, CosmoFlow, DeepCAM, and OpenCatalyst. The researchers evaluated the SambaNova, Cerebras, Graphcore, and Groq accelerators on the following workloads on the general matrix multiply layers, the convolution layers, and the ReLU layers. The matrix multiply layers GEMMs are used in fully connected layers, convolution layers, and recurrent layers. More importantly from a performance perspective, convolutions account for the most FLOPS in DL networks that operate on images and videos and occupy significant portions of DL networks for speech recognition. ReLU is a popular activation operator in DL networks requiring one comparison and one multiplication per input.

The HPC AI500 benchmark included two AIBench tests and was introduced in [24]. Ten AI kernels were selected in [25] to measure massive AI performance across distributed

computer systems spanning the cloud and the edge. In [26], qualitative metrics for AI benchmark performance are discussed. Al-Ali et al. [27] made the case for using FPGA to run AI workloads at the edge, supporting their case with the AlexNet deep learning classifier whose inference on the FPGA surpassed ARM and i5-6400 CPUs, and a GPU. In [28], AI performance was measured on NVIDIA V-100 and A-100, AMD MI100, and Cerebras CS-2 and Graphcore accelerators. Davis [29] surveyed AI benchmarks and provides detailed description for 12 of them. Arora et al. [30] addressed the energy efficiency of AI workloads. Other researchers [31] investigated DL performance on FPGAs and presented DL acceleration benchmark circuits for FPGA architecture and CAD research with 19 circuits covering a wide variety of accelerated neural networks, design sizes, implementation styles, abstraction levels, and numerical precisions.

Regarding metrics reported for DL/ML accelerator performance, it is arguable that FLOPS is not the only relevant metric for assessing AI performance. For instance, in some cases, mixing high and low FP precision may improve the FLOPS performance at the expense of lowering the quality of the AI application. Benchmarks which therefore consider these various dimensions in their scoring will report more realistic scores.

III. AIBENCH

AIBench is an open source Python library for evaluating the training and inference performances of DL models running on various hardware platforms, including CPUs, GPUs and TPUs. We ran AIBench v. 0.1.2 on the following platform: HP EliteOne 800, Intel Core i5-4570S Haswell processor with 4 logical processors at 2.9 GHz, 64-bit Windows 10 Pro 21 H2 Windows-10-10.0.19044-SP0, 8GB RAM, and Intel HD 4600 integrated graphics. The benchmark required the installation of Python, and the TensorFlow [31] machine learning library, and in our work, we installed TensorFlow-intel 2.12.0. In total, AI Benchmark consists of a total of 42 tests (5 3-tests + 1 1-test + 13 2-tests = 15+1+26=42) distributed over 19 sections as listed below:

1. MobileNet-V2 (classification, 2 tests)
2. Inception-V3 (classification, 2 tests)
3. Inception-V4 (classification, 2 tests)
4. Inception-ResNet-V2 (classification, 2 tests)
5. ResNet-V2-50 (classification, 2 tests)
6. ResNet-V2-152 (classification, 2 tests)
7. VGG-16 (classification, 2 tests)
8. SRCNN 9-5-5 (image to image mapping, 3 tests)
9. VGG-19 (image to image mapping3 tests)
10. ResNet-RSGAN (image to image mapping, 3 tests)
11. ResNet-DPED (image to image mapping, 3 tests)
12. U-Net (image to image mapping, 3 tests)
13. Nvidia Spade (image to image mapping, 2 tests)
14. ICNet (image segmentation, 2 tests)
15. PSPNet (image segmentation, 2 tests)

16. DeepLab (image segmentation, 2 tests)
17. Pixel-RNN (image inpainting, 2 tests)
18. LSTM (sentence sentiment analysis, 2 tests)
19. GNMT (text translation, 1 test)

Table 1 displays the run times of the AIBench tests, along with their run time variations. Tests 1.1 and 1.2 are the two MobileNet-V2 tests, while test 19.1 is the sole GNMT test. These run times do not only help in assessing the contribution of each test to the final benchmark score, but also serve in contrasting the execution times of various deep neural networks with similar parameters. The final AIBench scores are

Device Inference Score: 346

Device Training Score: 302

Device AI Score: 648.

Note that the run times and scores were obtained while simultaneously running the Intel OneAPI VTune profiler 2023.1.0 in parallel with the benchmark run, which slightly raised the total run time to about 35 minutes for all 19 tests. Also note that the device AI score, at the end of the run, is the sum of the device AI inference score and the device training score. The installation and running steps for AIBench training or inference only are found in [32].

IV. PASSMARK PERFORMANCE TEST

On the same platform, and also while simultaneously running the VTune profiler, the PassMark PerformanceTest 11.0 (PassMark) benchmark produced a score of 786, 6th percentile, and consisted of the following tests and their scores on the tested platform. The results of the PassMark benchmark run with the test suite scores and individual test score breakdowns (no units) are shown in Table 2. Note that unlike the AIBench run times where smaller numbers are better, the PassMark individual test scores are such that higher scores are better. A computer scoring twice as much as another computer approximately means that it can process twice the amount of data in the same amount of time. The CPU Mark score is a measure of the CPU performance while the PassMark score is a measure of the overall computer platform performance and is limited by the weakest component performance. It is not the average or sum of the subscores. Instead, the overall PassMark v. 11 score is given by the following formula

$$\begin{aligned} \text{PassMark Rating} &= 1/((1/(\text{CPU Mark}^*0.33)) \\ &\quad + (1/(2\text{D Mark}^*50)) + (1/(3\text{D Mark}^*0.5)) \\ &\quad + (1/(\text{Memory Mark}^*1.92)) + (1/(\text{Disk Mark}^*0.37)))/5 \end{aligned} \quad (1)$$

V. BENCHMARK PROFILING AND WORKLOAD CHARACTERIZATION

The results of the VTune profiling including performance snapshot, microarchitectural exploration, and memory access are displayed in Table 3. Memory access profiling and the

TABLE 1. AIBench tests run times.

Test # & Type	Test Parameters	Run Time
1.1 - inference	batch=50, size=224x224	1253 ± 483 ms
1.2 - training	batch=50, size=224x224	5299 ± 44 ms
2.1 - inference	batch=20, size=346x346	1864 ± 24 ms
2.2 - training	batch=20, size=346x346	10676 ± 70 ms
3.1 - inference	batch=10, size=346x346	1810 ± 128 ms
3.2 - training	batch=10, size=346x346	10744 ± 93 ms
4.1 - inference	batch=10, size=346x346	2230 ± 13 ms
4.2 - training	batch=8, size=346x346	9923 ± 80 ms
5.1 - inference	batch=10, size=346x346	1388 ± 16 ms
5.2 - training	batch=10, size=346x346	6146 ± 23 ms
6.1 - inference	batch=10, size=256x256	1943 ± 15 ms
6.2 - training	batch=10, size=256x256	9584 ± 195 ms
7.1 - inference	batch=20, size=224x224	2538 ± 16 ms
7.2 - training	batch=2, size=224x224	3025 ± 13 ms
8.1 - inference	batch=10, size=512x512	2643 ± 16 ms
8.2 - inference	batch=1, size=1536x1536	2400 ± 14 ms
8.3 - training	batch=10, size=512x512	24906 ± 760 ms
9.1 - inference	batch=10, size=256x256	4317 ± 26 ms
9.2 - inference	batch=1, size=1024x1024	6983 ± 19 ms
9.3 - training	batch=10, size=224x224	24414 ± 179 ms
10.1 - inference	batch=10, size=512x512	4657 ± 35 ms
10.2 - inference	batch=1, size=1536x1536	4238 ± 29 ms
10.3 - training	batch=5, size=512x512	11287 ± 14 ms
11.1 - inference	batch=10, size=256x256	7038 ± 100 ms
11.2 - inference	batch=1, size=1024x1024	11274 ± 43 ms
11.3 - training	batch=15, size=128x128	15435 ± 43 ms
12.1 - inference	batch=4, size=512x512	11930 ± 62 ms
12.2 - inference	batch=1, size=1024x1024	12012 ± 33 ms
12.3 - training	batch=4, size=256x256	13522 ± 108 ms
13.1 - inference	batch=5, size=128x128	3602 ± 220 ms
13.2 - training	batch=1, size=128x128	5553 ± 32 ms
14.1 - inference	batch=5, size=1024x1536	2780 ± 38 ms
14.2 - training	batch=10, size=1024x1536	6912 ± 53 ms
15.1 - inference	batch=5, size=720x720	16488 ± 33 ms
15.2 - training	batch=1, size=512x512	8537 ± 51 ms
16.1 - inference	batch=2, size=512x512	4491 ± 22 ms
16.2 - training	batch=1, size=384x384	6104 ± 16 ms
17.1 - inference	batch=50, size=64x64	4862 ± 25 ms
17.2 - training	batch=10, size=64x64	8631 ± 255 ms
18.1 - inference	batch=100, size=1024x300	14274 ± 146 ms
18.2 - training	batch=10, size=1024x300	27686 ± 140 ms
19.1 - inference	batch=1, size=1x20	2890 ± 14 ms

“AIBench - First 5 tests only” ran until 1GB of data were collected and stopped (between 4 min. and 5 min. of run time). Table 3 also includes the VTune profiling results of AIBench with all tests, with only the inference tests, and with only the training tests.

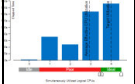
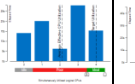
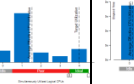
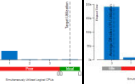
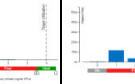
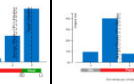

The major performance indices are:

TABLE 2. PassMark subtest scores.

Test Name	Test Type	Test Score
Test Suites		
CPU MARK	CPU intensive	4635, 16% percentile
2D GRAPHICS MARK	2D Graphics	302, 26th percentile
3D GRAPHICS MARK	3D Graphics	469, 10th percentile
MEMORY MARK	Memory	1533, 9th percentile
DISK MARK	Disk	2524, 16% percentile
Individual Tests		
CPU MARK	Integer Math	12468
CPU MARK	Prime Numbers	27
CPU MARK	Compression	73786
CPU MARK	Physics	286
CPU MARK	CPU Single Threaded	2020
CPU MARK	Floating Point Math	9545
CPU MARK	SSE	6167
CPU MARK	Encryption	1222
CPU MARK	Sorting	7856
2D GRAPHICS MARK	Simple Vectors	12
2D GRAPHICS MARK	Fonts and Text	152
2D GRAPHICS MARK	Windows Interface	39
2D GRAPHICS MARK	Image Filters	433
2D GRAPHICS MARK	Image Rendering	23
2D GRAPHICS MARK	Direct 2D	26
2D GRAPHICS MARK	PDF Rendering	43
2D GRAPHICS MARK	Direct 2D – SVG	26
3D GRAPHICS MARK	DirectX 9	9.9
3D GRAPHICS MARK	DirectX 10	1.7
3D GRAPHICS MARK	DirectX 11	5.8
3D GRAPHICS MARK	DirectX 12	N/A
3D GRAPHICS MARK	GPU Compute	367
MEMORY MARK	Database Operations	2001
MEMORY MARK	Memory Read Cached	17450
MEMORY MARK	Memory Read Uncached	9873
MEMORY MARK	Memory Write	4908
MEMORY MARK	Available RAM	3460
MEMORY MARK	Memory Latency	45
MEMORY MARK	Memory Threaded	11034
DISK MARK	Disk Sequential Read	544
DISK MARK	Disk Sequential Write	215
DISK MARK	IOPS 32kQD20	73
DISK MARK	IOPS 4KQD1	7.7

Front-End Bound is the percentage of time where the Front-End (i.e., instruction fetching, branch prediction, and decoding instructions into micro-operations) section of the CPU does not provide enough micro-operations (uOps) to its Back-End section (execution units). The Front-End Bound number reflects the amount of bubbles or empty issue slots where the Front-End delivered no uOps –due to instruction cache misses for example– when the Back-End could have accepted them.

TABLE 3. VTune profiling of AIBench and PassMark tests.

	AIBench– All tests	AIBench – First 5 tests	PassMark CPU Mark	PassMark Memory Mark	PassMark 3D Graphics Mark	PassMark Disk Mark	AIBench – Inference Only	AIBench – Training Only
PERFORMANCE SNAPSHOT								
Elapsed Time (sec)	2088.738	440.036	78.898	59.696	208.101	190.150	966.79	1179.454
Logical Core Utilization	83.2%=3.33/4		71.1%=2.84/4	59.6%	17.6%=0.70/4	14.4%=0.58/4		79%=3.16/4
Microarch. Usage (% pipe slots)	38.1%		37.9%	23.8%	24.1%	31.1%	47%	29%
Memory Bound (% pipe slots)	38.6%		25.9%	66.4%	30.2%	13.2%	28%	44%
Threading			71.1%	59.6%	17.6%	14.4%		
Memory Access	38.6%		25.9%	66.4%	30.2%	31.1%	28%	44%
MICROARCHITECTURAL EXPLORATION								
CPI Rate	0.654	0.630	0.595	0.885	0.948	1.086	0.522	0.859
Retiring (% of pipeline slots)	38.3%	39.2%	40.4%	25.6%	28.1%	24.7%	47.6%	29.2%
Front-End Bound (% pipe slots)	7.3%	8.4%	6.7%	5.7%	19.6%	50.6%	7.3%	7.7%
Bad Speculation (% pipe slots)	1.2%	1.4%	7.9%	0.8%	5.4%	6.4%	1.1%	1.5%
Back-End Bound (% pipe slots)	53.1%	51.0%	44.9%	67.9%	46.9%	18.3%	44%	61.5%
Memory Bound (% pipe slots)	38.6%	36.3%	24.6%	53.7%	37.6%	13.1%	28.4%	43.6%
L1 Bound (% clocks)	6.7%	6.3%	5.3%	4.1%	20.6%	11.7%	5.6%	8.1%
L2 Bound (% clocks)	0.0%	0.0%	0.0%	0.0%	0.0%	1.6%	0.0%	0.0%
L3 Bound (% clocks)	7.9%	7.4%	2.4%	1.1%	10.1%	11.8%	4.1%	11%
L3 Latency	18.1%	14.8%	5.5%	14.0%	11.5%	24.7%	12.5%	21.6%
DRAM Bound (% clk ticks)	27.3%	25.9%	17.8%	45.7%	22.4%	4.3%	24.3%	29.9%
Memory Bandwidth	37.0%	42.2%	54.9%	55.8%	37.4%	13.6%	35.9%	37.2%
Memory latency	43.3%	43.5%	37.0%	28.9%	42.3%	40.3%	46.3%	39.4%
LLC Miss	24.9%	20.1%	22.3%	20.2%	11.4%	4.7%	21.3%	29.3%
Store Bound (% clocks)	9.0%	9.8%	1.2%	7.4%	6.7%	2.7%	7.4%	10.5%
Core Bound (% pipe slots)	14.5%	14.7%	20.3%	14.3%	9.3%	5.2%	15.6%	17.9%
Port Utilization (% clocks)	18.0%	18.9%	21.8%	15.4%	10.8%	12.8%	22.2%	22.4%
CPU Utilization (out of 4)	3.33 (83.3%)	2.95 (73.6%)	2.14 (53.5%)	1.54 (38.5%)	0.18 (4.6%)	0.11 (2.7%)	3.46 (87%)	3.15 (79%)
CPU Utilization (graphical, per core)								
MEMORY ACCESS								
Elapsed time (sec)		315.607		60.396			262.147	349.545
Loads per sec.		4,993,906,922		854,557,393			6,963,148,384	3,879,237,231
Stores per sec.		681,363,059		266,072			852,339,729	580,484,021
LLC misses per sec.		10,236,138		3,430,794			5,383,401	12,544,684
Number of threads		12		18			11	13
Socket 0 to DRAM		60.6%		54.6%			50.9%	55.8%

Memory bound: is the fraction of execution pipeline slots stalled due to memory loads and stores.

Retiring: is the fraction of pipeline slots with useful (not speculative) work. When the Retiring number is high, this usually entails a larger Instruction-Per-Cycle (IPC=1/CPI) number.

Core bound: reflects how much non-memory issues were a bottleneck, such as contention to shared hardware resources, or instruction dependencies (for instance, RAR, WAR hazards).

Bad Speculation: is the fraction of pipeline slots wasted due to incorrect speculations, for instance, as a result of mispredicted branches and machine clears.

CPU utilization: indicates how much the logical processors were loaded.

Regarding the parallel thread executions and core utilizations, the thread activities versus time for the AIBench and CPU Mark benchmarks are plotted in Figures 1-2. We observe that AIBench with all 45 inference and training tests dis-

played the longest and most intense parallel logical core activities with high utilization compared to the PassMark tests.

The blue bar charts in Tables 3 reflect the core utilizations of the tests with the leftmost bar representing the elapsed time (in sec.) the test ran on 0 cores (i.e. all cores idle). The next 4 bars represent the elapsed time spent with 1, 2, 3 and 4 cores running simultaneously, respectively. It is clear that the tests which mostly took advantage of all processor cores were AIBench-inference only, AIBench-all tests, AIBench-first 5 tests. CPU Mark placed in second as most of its elapsed time was spent on 3 cores. In third place, Memory Mark and AIBench-training spent most of their elapsed times on only 1 core. In the last places were 3D Graphics Mark and Disk Mark which left all 4 processor cores idle for most of their elapsed times.

In terms of speculation performance, bad speculation was more pronounced in CPU Mark and Disk Mark than the rest.

Table 4 displays a sample of event counts in millions of events per second. The complete list of events

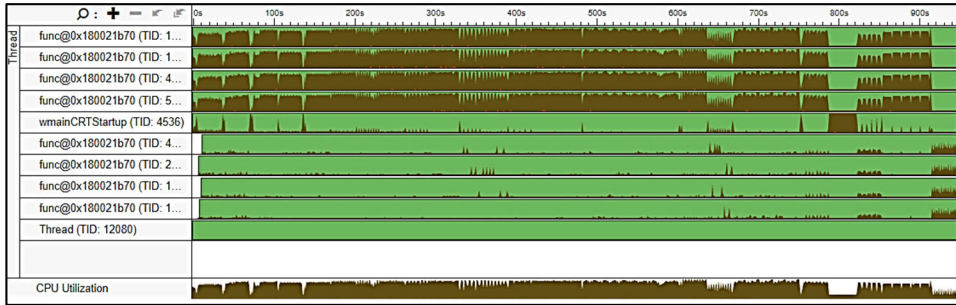


FIGURE 1. Thread activity of AIBench – all tests.

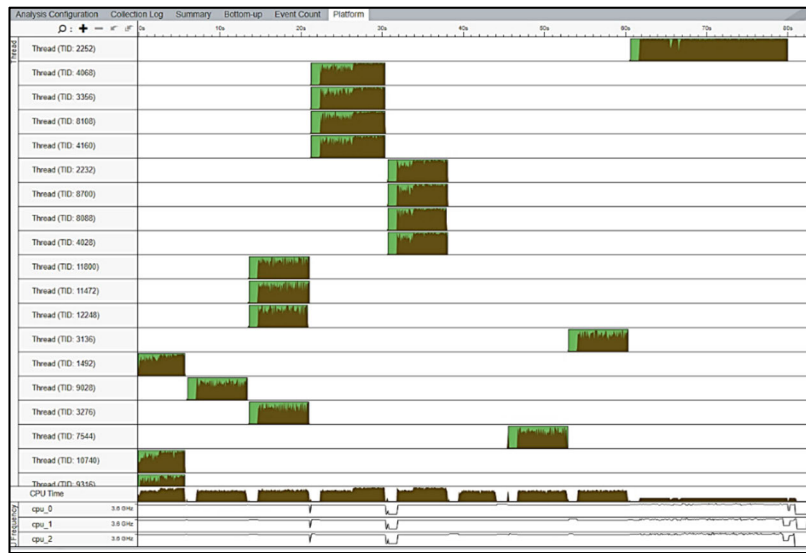


FIGURE 2. Thread activity of CPU mark.

supplied by VTune, and which were later normalized to display as millions of events per second, can be found in the Table in the Supplementary. Higher event counts (per sec.) directly correlate with more intense stressing of the related units. For instance, considering the event counts where the lowest benchmark test count (per sec.) is less than 0.1% of the highest benchmark test count (per sec.), the event count analysis reveals that both AIBench training and inference tests exceed all other considered benchmarks in CPU_CLK_UNHALTED.REF_TSC (number of reference cycles when the core is not in a halt state), IDQ_UOPS_NOT_DELIVERED.CYCLES_0 (number cycles during which the Front-End allocated zero uOps to the Resource Allocation Table while the Back-end of the processor is not stalled.), INT_MISC.RECOVERY_CYCLES (number of cycles spent waiting for a recovery after an event such as a processor nuke or assist), MEM_LOAD_UOPS_L3_HIT_RETIRED.XSNP_HIT (number of retired load uOps for which the data sources were L3 and cross-core snoop hits in core cache), and MEM_LOAD_UOPS_RETIRED.L1_HIT_PS (number of retired load uOps with L1 cache hits as data sources) counts (per

sec.). AIBench training only tests exceed all other tests in DTLB_LOAD_MISSES.WALK_DURATION (number of cycles when the page miss handler is servicing page walks caused by DTLB load misses), L1D_PEND_MISS.PENDING (increments the number of outstanding L1D misses every cycle), and L1D_PEND_MISS.REQUEST_FB_FULL (number of times a request needed a FB entry but there was no entry available for it) counts (per sec.). AIBench inference only tests exceeded all other tests in IDQ.ALL_DSB_CYCLE_ANY_UOPS (number of cycles where DSB is delivered at least one uOp), INST_RETIRED.PREC_DIST (number of precise instruction retired events), L2_RQSTS.RFO_HIT (number of store RFO requests that hit the L2 cache), and UOPS_RETIRED.RETIRE_SLOTS (number of retirement slots used each cycle). CPU Mark exceeded all other tests in ARITH.DIVIDER_UOPS (number of uOps which used the divider including divide and square root uOps), BR_MISP_RETIRED.ALL_BRANCHES (number of mis-predicted branch instructions at retirement), and MEM_UOPS_RETIRED.ALL_STORES (number of all retired store uOps) counts (per sec.). Memory Mark exceeded all other tests in CYCLE_ACTIVITY.STALLS

TABLE 4. Events per second sample.

Hardware Event Type	AIBench - all tests	AIBench- first 5 tests	CPU Mark - all tests	Memory Mark - all tests	3D Mark - all tests	Disk Mark - all tests	AIBench - all tests - inference only	AIBench- all tests - training only
	Millions of Events Per Sec.	Millions of Events Per Sec.	Millions of Events Per Sec.	Millions of Events Per Sec.	Millions of Events Per Sec.	Millions of Events Per Sec.	Millions of Events Per Sec.	Millions of Events Per Sec.
ARITH.DIVIDER UOPS	5.15	6.41	89.26	2.18	0.46	2.29	4.49	4.97
BR MISP RETRD.ALL BR	6.22	6.04	29.52	1.58	1.76	11.10	6.16	7.57
DTLB LD MISS.STLB HIT	151.15	132.44	7.92	2.45	1.65	13.09	87.07	202.61
ICACHE.IFDATA STALL	229.69	164.44	17.52	26.71	21.50	11.07	238.58	235.35
INST RETIRED.PREC DIST	16282.86	14880.21	11507.29	5666.52	512.32	6.85	20975.50	11704.86
MEM RTRD.ALL STORES	775.76	650.63	1082.20	560.36	93.61	0.58	928.87	659.40
RS EVNTS.EMPTY CYCLE	1021.34	947.33	281.71	1578.59	47.40	2.39	570.17	1343.38
UOPS RETRD.RETRE SLOT	16257.01	14728.59	11120.49	5206.46	564.13	0.25	20975.65	11726.18

_LDM_PENDING (number of cycles during which no instructions were executed in the execution stage of the pipeline and there were pending memory instructions waiting for data), and LSD.UOPS (number of uOps delivered by the loop stream detector in the branch prediction unit) counts (per sec.). Disk Mark exceeded all other tests in ILD_STALL.LCP (number of cycles where the decoder is stalled on an instruction with a length changing prefix), MEM_LOAD_UOPS_L3_HIT_RETIRED.XSNP_MISS (number of retired load uOps for which data sources were L3 hit and cross-core snoop missed in core cache), and OTHER_ASSISTS.ANY_WB_ASSIST (number of micro-code assists invoked by hardware upon uOp writeback) counts (per sec.).

VI. RESULTS ANALYSIS

A. GRAPHICAL ANALYSIS

In this section, in order to better assess the results, we conduct a graphical analysis of the AIBench and PassMark benchmarks by plotting the CPI (one of the factors in the elapsed time product) Vs. various captured metrics to better contrast the AIBench and components of the PassMark benchmarks.

In order to identify bottlenecks, we start by analyzing the CPI versus the core utilization, and the major components of Table 3: Core Bound, Memory Bound, Memory Bandwidth, Percent Retiring, Front End Bound, and Back End Bound. The CPI vs CPU utilization plot is shown in Fig. 3. The bubble colors indicate the percent port utilization value. The ports on a modern Intel microprocessor attach to either an arithmetic logic unit (ALU), a floating-point unit (FPU), a load-store (LS) unit, and an address generation unit (AGU). A regression line with $r=-0.81$ (slope) is also drawn and shown to reflect the relationship. Low port utilization results in a large CPI. A high CPU utilization results in a low CPI. The AIBench-Training only, CPU Mark, and AIBench-Inference only are the furthest away points from the regression line. This results from these tests being highly memory bound which raises the CPI despite a high CPU utilization.

The CPI vs Core Bound (which reflects non-memory core cycles) plot is shown in Fig. 4 along with a regression line with $r=-0.74$. The bubble color reflects the

UOPS_ISSUED.ANY value. As expected, the CPU Mark and AIBench were the most core bound tests and suffered from contention to shared hardware resources. Disk Mark was the least core bound, followed by the 3D Graphics Mark test. The number of uOps issued correlates positively with the core bound value.

In Fig. 5, we plot CPI vs Memory Bound and a very weak regression line with $r=-0.04$. The bubble color reflects the % DRAM Bound value, i.e., the amount of cycles where the CPU stalled on DRAM. As expected, Memory Mark was the most DRAM bound followed by AIBench, 3D Graphics Mark. Disk Mark and CPU Mark were the least DRAM bound. Disk Mark seems to be the test going against the trend of the higher the memory bound value, the higher is the CPI. Without the Disk Mark test, the regression line would have a positive r .

Although 3D Graphics Mark and AIBench-all tests seem to have a close memory bound value, 3D Graphics Mark shows a much higher CPI given that it is more front end bound with less threading than AIBench.

In Fig. 6, we plot the CPI vs the memory bandwidth with a regression line with $r=-0.48$. The bubble color reflects the memory latency value. Memory Mark followed by CPU Mark were the most memory bandwidth consuming benchmarks with the highest stalls due to approaching the bandwidth limits of the DRAM. The AIBench and 3D Graphics Mark came next. Disk Mark was in the distant last position. The benchmarks which accumulated the largest percentage of stalls due to the memory latency were AIBench-Inference, AIBench-all tests, followed by 3D Graphics Mark and Disk Mark, then AIBench-Training and CPU Mark. Memory Mark was a distant last.

The CPI vs % Retiring graph is displayed in Fig. 7 with a regression line with a strong $r=-0.96$. The bubble color is uOps retired. AIBench, in particular AIBench-Inference, had the highest number of retired uOps slots followed by CPU Mark. Disk Mark, Memory Mark, and 3D Graphics Mark occupied the last positions. The CPI correlated negatively with the % retiring and the number of retired uOps. We also observe that due to its memory bound nature,

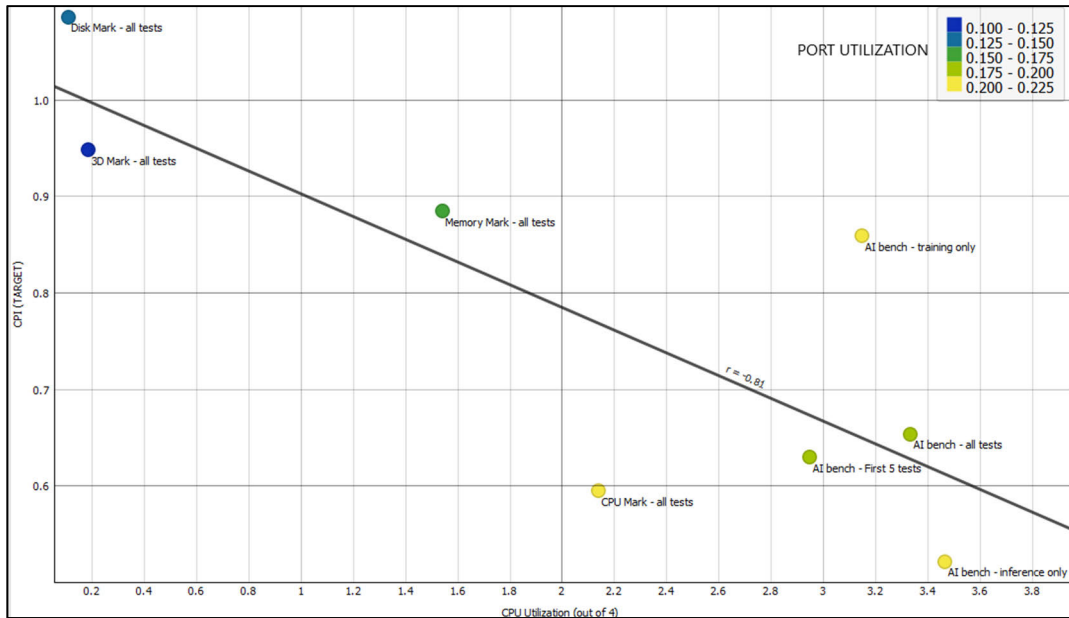


FIGURE 3. CPI vs. CPU utilization (and Port Utilization).

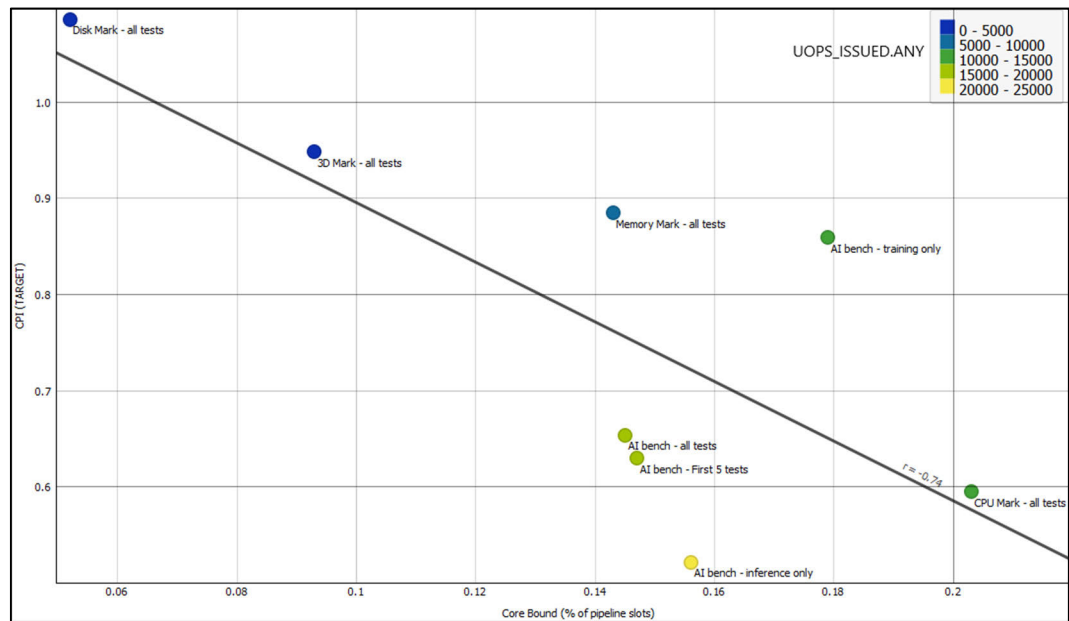


FIGURE 4. CPI vs. core bound (and uOps_Issued).

AIBench-Training had a much lower % retiring and number of uOps retired than its AIBench-Inference counterpart.

In Fig. 8, we plot CPI vs Front End Bound, along with a regression line with slope $r=0.73$. The bubble color reflects the mispredicted branched retired value. Disk Mark was a distant first given its high number of stalled cycles due to high Icache misses (16.4%) and branch rasteers (21.5%, as a result of branch mispredictions). 3D Graphics Mark came in second place. Memory Mark, CPU Mark, and AIBench were in the last spots with little front end issues.

In Fig. 9, we plot CPI vs Back End Bound, with a regression line of slope $r=-0.27$. The bubble color reflects the

Memory Bound value. Memory Mark, with high memory bandwidth and latency, and with a high -59%- port 0 utilization, came in first place due to data (L1, L2, L3) cache misses, followed by AIBench (due its high DRAM bound percentage, high -49%- port 0 utilization), 3D Graphics Mark (memory latency 42% and memory bandwidth 37%, L1 -20%- and L3 -10%- cache bound, and high -38%- port 0 utilization) and CPU Mark (core bound, high port utilization indicating execution unit contention, and high memory bandwidth).

For comparison purposes, Fig. 10 displays the Radviz diagram for the benchmarks with four axes: % retiring, % front end bound, % back end bound, % bad speculation. We make

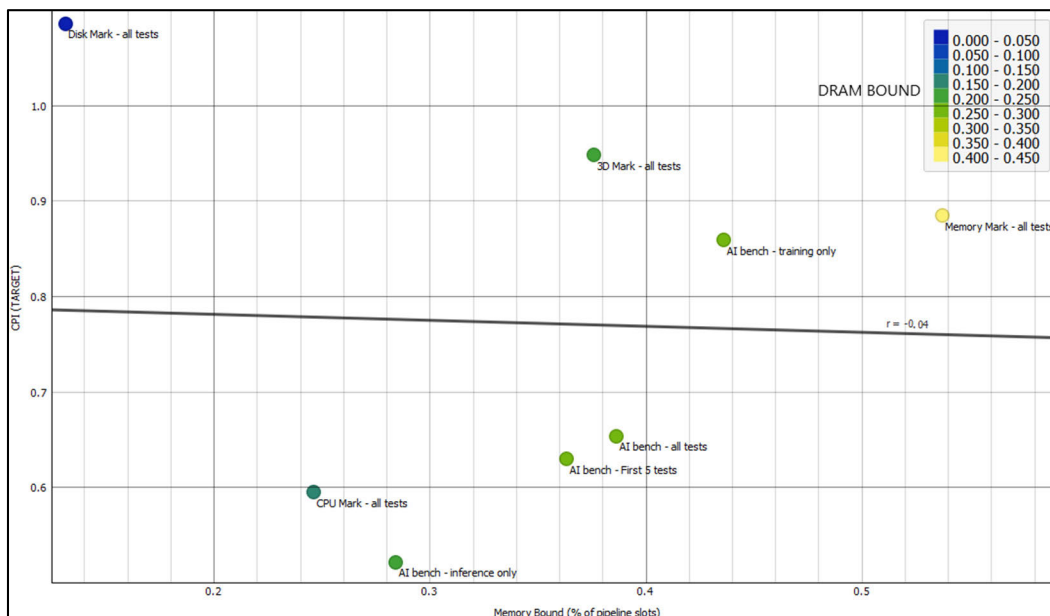


FIGURE 5. CPI vs. memory bound (and DRAM bound).

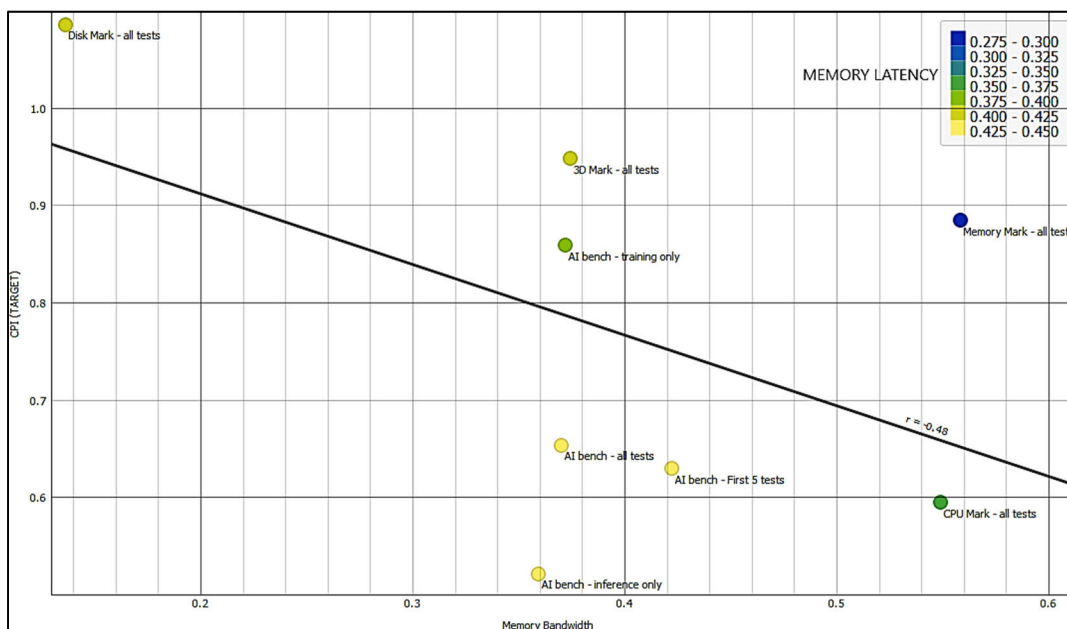


FIGURE 6. CPI vs. memory bandwidth (and Memory Latency).

the following observations. The AIBench and CPU Mark tests achieve higher % retiring from the rest. Most benchmarks are more back end bound than front end bound with the exception of Disk Mark. Memory Mark is the most back end bound. In terms of bad speculation, CPU Mark, 3D Graphics Mark, and Disk Mark exceed the rest, while the Memory Mark and AIBench tests generate the lowest percentages of bad speculations.

Fig. 11 shows another Radviz diagram for the benchmarks with four axes: memory latency, memory bandwidth, number of last level cache (LLC) misses, and % store bound. In terms of memory performance, AIBench is the least sensitive to

memory bandwidth and is in the center. In terms of memory latency, Disk Mark is the most sensitive and is in distant first place, while Memory Mark is the least sensitive. The opposite can be said regarding % store bound, with Memory Mark in first place, while Disk Mark in last place. In terms of number of last level cache (LLC) misses, AIBench is in first place, in particular AIBench-Training. Furthermore, according to the memory access profiles of Table 3, the AIBench tests performed 6x more loads/sec, 2560x more stores/sec, and 3x more LLC misses than the Memory Mark tests.

In Fig. 12, the last Radviz diagram for the benchmarks with four axes: % L1 Bound, % L2 Bound, % L3 Bound,

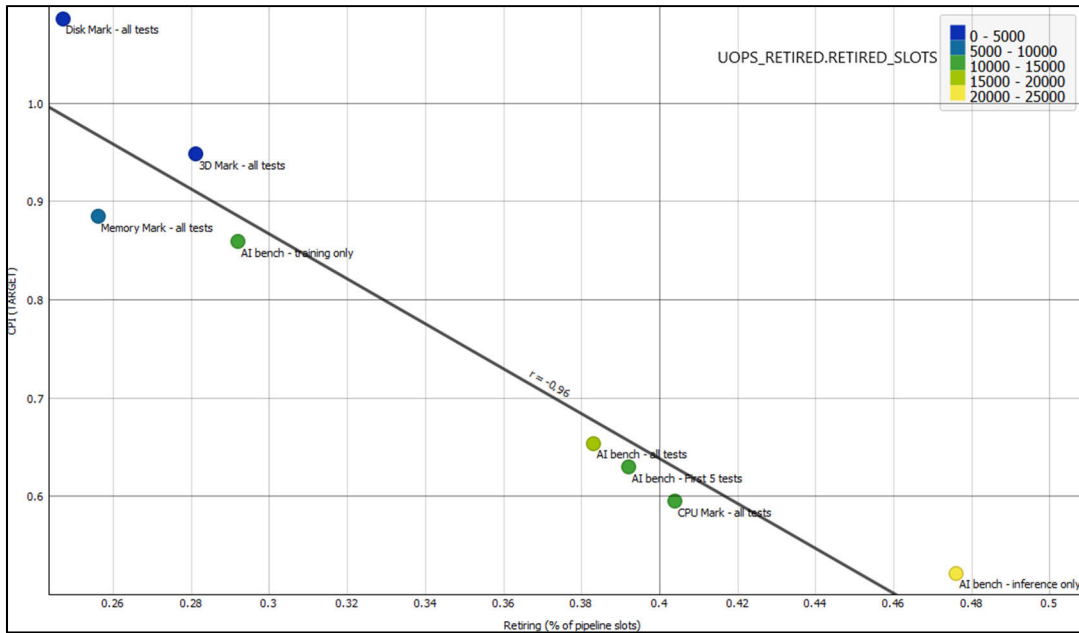


FIGURE 7. CPI vs. retiring (and uOps Retired).

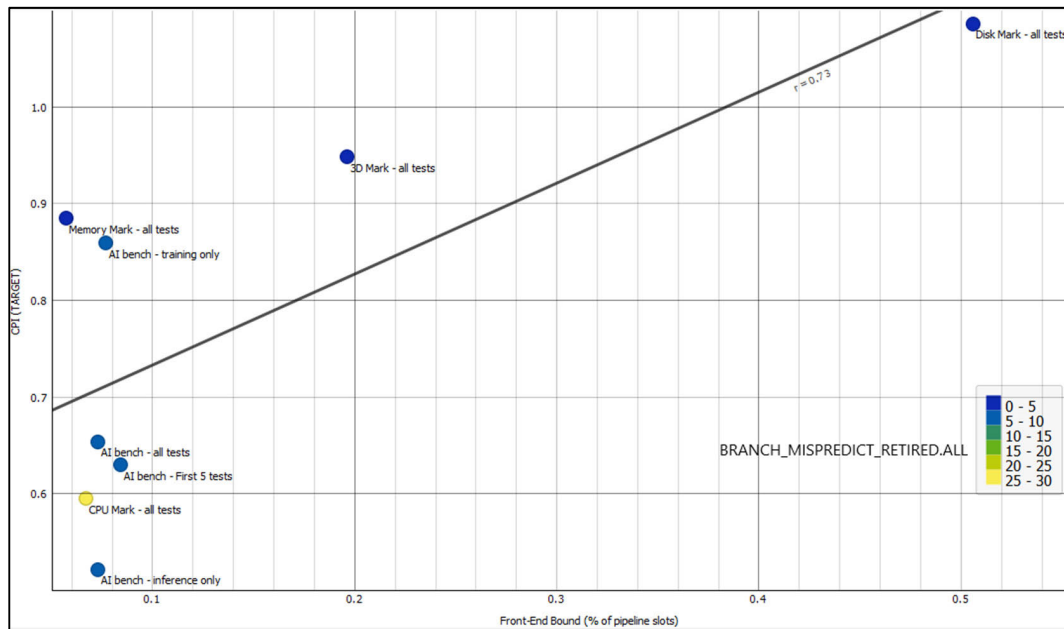


FIGURE 8. CPI vs. front end bound (and Branch Mispredicted Retired).

and LLC misses, reflects the cache performance. The 3D Graphics Mark stalled the most on the L1 cache without missing it. The Memory Mark tests stalled the least on the L1 cache. The Disk Mark benchmark stalled the most on the L2 cache. The Disk Mark tests, followed by 3D Graphics Mark, stalled the most on the L3 cache shared by the 4 cores either due to contention between the cores or due to conflict misses. The AIBench, CPU Mark, and Memory Mark tests missed the LLC the most, while the Disk Mark tests missed the LLC the least, and was more L3 bound and L1 bound than L2 bound. In general, the 3D Graphics Mark was closest to the center.

B. HOTSPOT ANALYSIS

Fig. 13 displays the results of the hotspots and function call stack for the AIBench-Inference benchmark revealing that Tensorflow roughly occupies 25% of the CPU time exercising the AVX instruction set. The Tensorflow-ML was reported to improve the performance AI workloads on CPUs.

C. EVENT CORRELATIONS WITH CPI AND ELAPSED TIME

We fed the VTune performance snapshot metrics and event counts of the AIBench and PassMark benchmarks to Orange 3.0 after normalizing the data by dividing the event counts by the elapsed time * 10⁶, and dividing the elapsed time by 100.

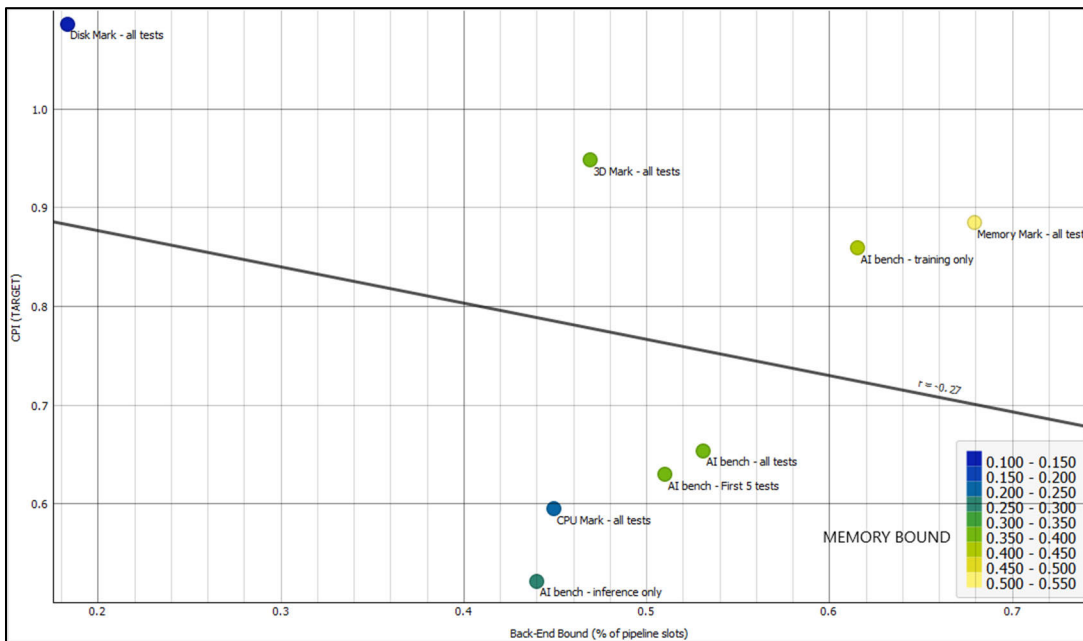


FIGURE 9. CPI vs. back end bound (and Memory Bound).

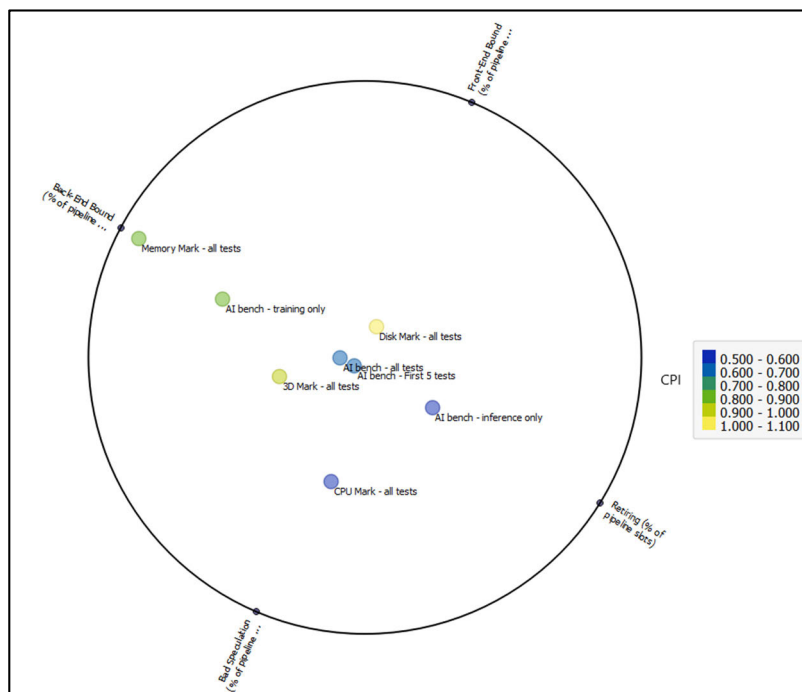


FIGURE 10. Radviz diagram with four axes.

Fig. 14 displays the Pearson correlation coefficients of the features with CPI, where - or blue mean negative correlation, and + or green mean positive correlation. Some of the events highly correlating with the CPI are uOps dispatched to Port 5 / 1 / 7 / 0 / 4, uOps executed, cycles delivering at least one uOp or 4 uOps to the decode stream buffer, instructions retired, % retiring, retired store uOps, and % port

utilization. More details on the meaning of the events can be found in [33]. Interestingly, the highest negatively correlating events with the CPI, are the number of dispatched uOps on the Haswell port 5 (Integer ALU/shift, vector Integer ALU, and 256-bit FP shuffle and blend), number of issued uOps, number of instruction decode queues (IDQ) decode stream buffer (DSB) uOps. Among the highest positively correlating

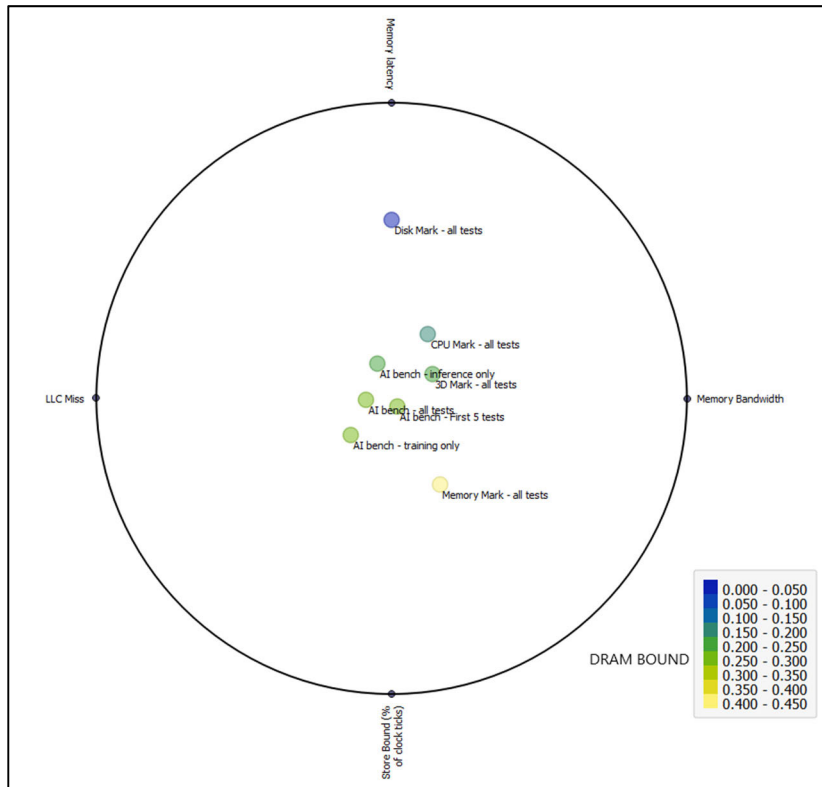


FIGURE 11. Second radviz diagram with four axes – memory performance.

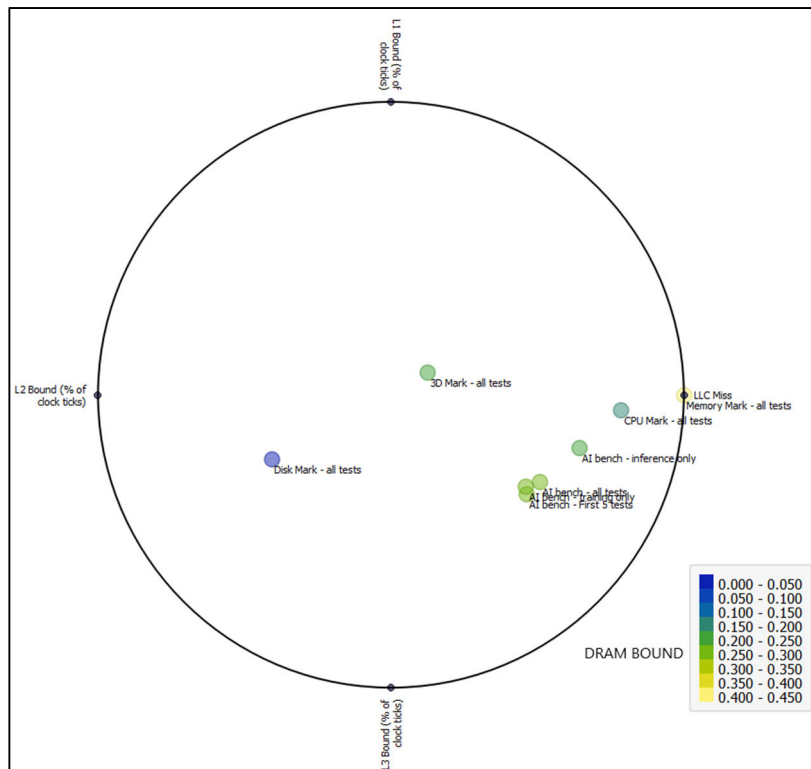


FIGURE 12. Third radviz diagram with four axes – cache performance.

events with the CPI, are the % front end bound, and L2 cache bound.

Similarly, we correlated the captured events of the AIBench and PassMark benchmarks with the normalized

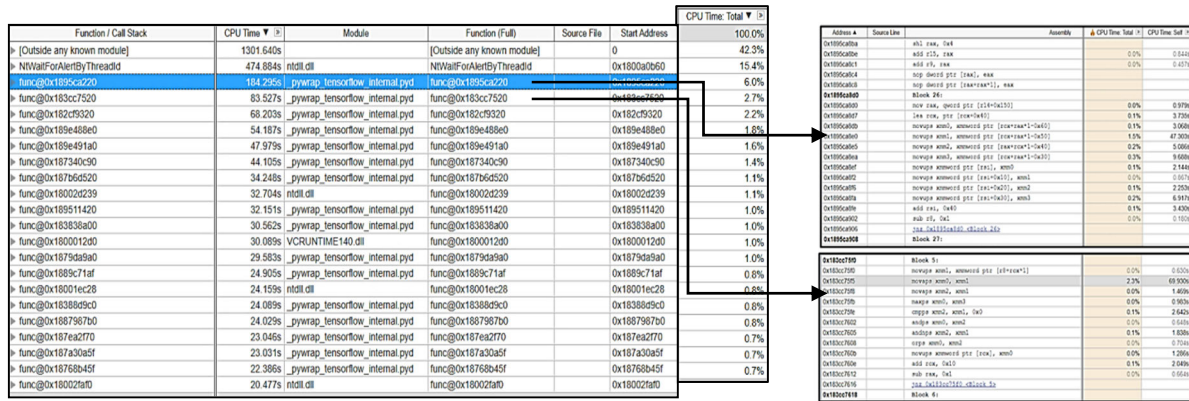


FIGURE 13. Hotspots and function call stack in AIBench-Inference only.

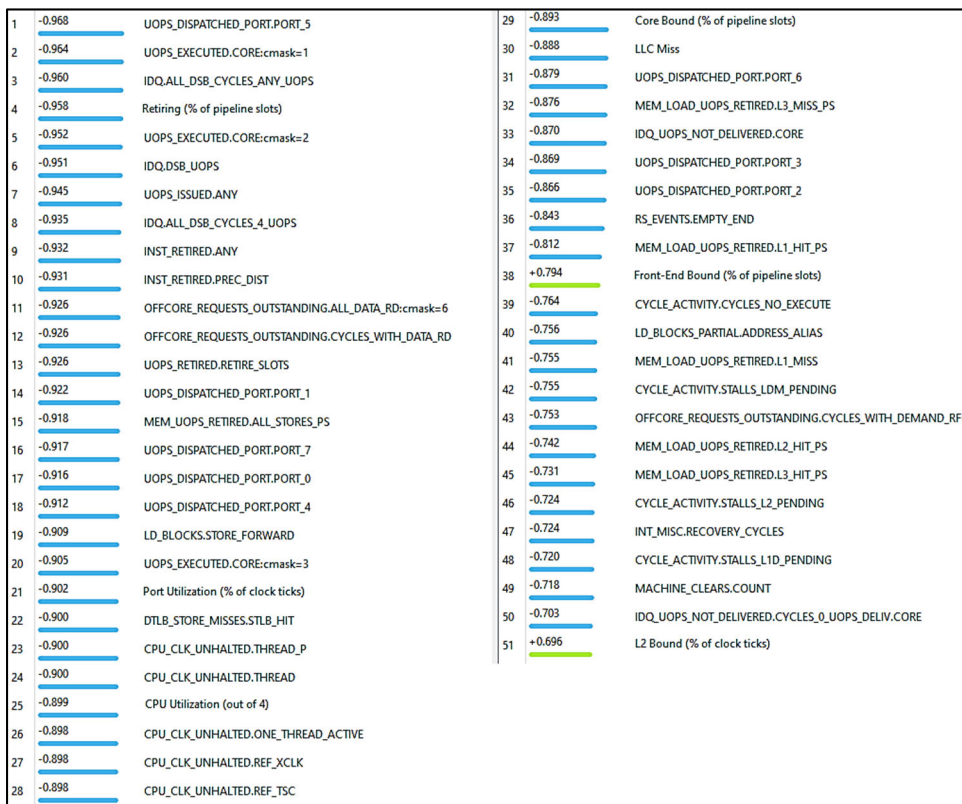


FIGURE 14. Correlation coefficients of the event counts and metrics with the CPI.

elapsed time. The Pearson correlation coefficients of the features with elapsed time (scaled by 1/100; - or blue means negative correlation, + or green mean positive correlation) are displayed in Fig. 15. Some of the events which highly correlate with the normalized elapsed time are write back assists, ITLB misses, STLB and DTLB load misses, memory uOps retired, and “OFFCORE_RESPONSE:request= DEMAND_RFO:response=L3_hit.hitm_other_core” which reflects the number of all data writes –request for ownership RFOs – which hit in the L3 cache, and which also hit in another core cache

in the Modified state requiring a line forwarding. Write back assists reflect the number of microcode assists invoked by the hardware when writing back, exclude FP assists, and include assists pertaining to page access dirty and AVX.

D. RESULT DISCUSSION

In Summary, the AIBench benchmark was characterized by having among the highest levels of CPU utilization particularly with all 4 cores active, microarchitectural usage, retiring, port utilization, back end (memory, DRAM) bound, memory latency, store bound, and core bound percentages

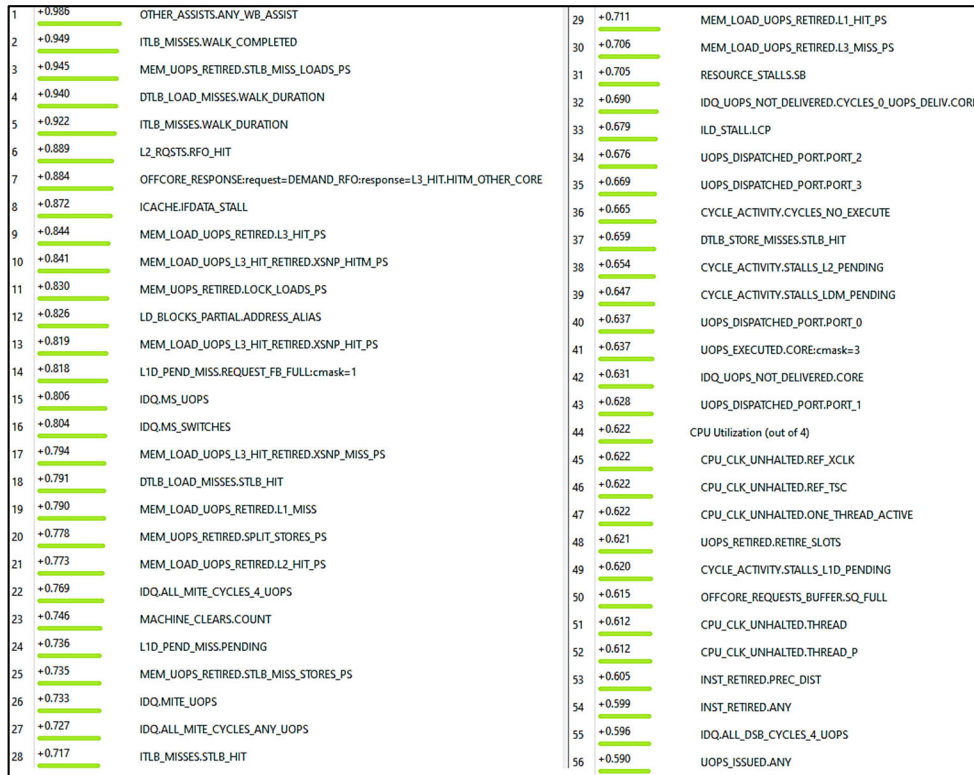


FIGURE 15. Correlation coefficients of the event counts and metrics with the elapsed time.

among the covered benchmarks. It scored among the lowest CPIs. In particular, the AIBench-Training generated the highest L1D and L2 cache pending, DTLB miss, and machine clear counts. The AIBench-Training exhibited a larger CPI, was more memory bound, and exhibited lower retiring and logical core utilization than its AIBench-Inference counterpart.

The PassMark CPU Mark benchmark was characterized by having among the highest levels of microarchitectural usage, threading, retiring, port utilization, memory bandwidth, and core bound percentages. It also scored among the lowest CPIs.

The Memory Mark benchmark was characterized by having among the highest levels of back end (memory, DRAM) bound, and memory bandwidth percentages. It also scored among the lowest percentages of microarchitectural usage, retiring, port utilization, front end bound, L1 and L3 cache bound, and memory latency.

The 3D Graphics Mark benchmark exhibited the highest L1 and L3 cache bound, and the lowest CPU utilization, microarchitectural usage, threading, and core bound percentages.

The Disk Mark benchmark was the most front end bound, L1 and L3 cache bound, and among the lowest in CPU utilization, retiring, port utilization, back end bound, memory bandwidth, and store bound.

Generally, the performance of hardware units identified and exercised by the above workload characteristics can be

enhanced by increasing hardware resources, raising capacities, reducing latencies, or increasing pathway bandwidths.

In terms of core activity, the Disk Mark test followed by the 3D Graphics Mark marked the lowest core utilization with all 4 cores idle during most of the elapsed time. AIBench-training only and Memory Mark followed with most of their elapsed times spent on 1 core only. Next came CPU Mark which ran most of the time on 3 cores. The AIBench-all tests, AIBench-inference only, and AIBench-first 5 tests took the crown with the top utilizations of all 4 cores. Although AIBench-all tests and CPU Mark marked high core utilizations, the thread activity plots of both benchmarks revealed a much higher execution intensity of the AIBench benchmark.

VII. MACHINE LEARNING CLASSIFICATIONS

A. CLASSIFYING THE CPI

Using Orange 3.0 to train and test 9 ML models, we classified the different AIBench and PassMark workload CPIs and elapsed times based on the above event counts (events/sec), and the following captured metrics (as is): retiring, front end bound, bad speculation, back end bound, memory bound, L1 bound, L2 bound, L3 bound, contested accesses, data sharing, L2 latency, SQ full, DRAM bound, memory bandwidth, memory latency, LLC miss, store bound, core bound, port utilization, and CPU utilization. In this classification investigation, the AIBench-inference only and AIBench-training only data were skipped and excluded. Instead, the

TABLE 5. Performance of the 9 ML models in classifying the CPI.

ML Method	MSE	RMSE	MAE	R ²
Random Forest	0.023	0.151	0.142	0.150
SVM	0.033	0.182	0.165	-0.240
kNN	0.008	0.088	0.081	0.711
Neural Networks	0.029	0.169	0.148	-0.065
Decision Tree	0.038	0.195	0.179	-0.422
Linear Regression	0.006	0.077	0.051	0.778
SGD	0.087	0.294	0.237	-2.232
AdaBoost	0.013	0.116	0.086	0.497
Gradient Boosting	0.015	0.123	0.094	0.435

TABLE 6. Optimal ML model parameters for classifying the CPI.

ML Method	Parameter Settings
Random Forest	# trees=800 Limit depth of individual trees=3
SVM	v-SVM Regression Cost =0.2, Complexity bound=0.1 Kernel: Sigmoid, kernel: tanh(g x y +c), g =auto, c=1 Numerical tolerance=0.01
kNN	# neighbors=2; metric =Euclidean; Weight=Distance
Neural Networks	Neurons in hidden layers= 40, 70, 50, 10, 20, 10 Activation=Logistic; Solver=Adam Regularization a=0.3 Max. # iterations=200
Decision Tree	Min # instance in leaves=3 Limit the maximal tree depth=10
Linear Regression	Fit intercept; Elastic Net regularization L1 0.37:0.63 L2
AdaBoost	# estimators=5; learning rate=0.51; Classification algorithm= SAMME; Regression loss function=Linear
Gradient Boosting	# trees=8; learning rate=1; Limit depth of individual trees=9; Do not split sunsets smaller than 2
Stochastic Gradient Descent	Classification: ϵ insensitive; ϵ =0.1; Regression: Squared loss; Regularization: Ridge (L2)

AI Bench-all tests metrics were included in the dataset. With 66% random sampling (i.e. 66% of data rows were used for training, and 34% of the rows used for testing), and based on the mean square error (MSE) and R² metrics, the linear regression (LR; with MSE=0.006; R² = 0.778) turned out to be the best CPI classifier, followed by kNN (with MSE=0.008; R² = 0.711), as shown in Table 5. SGD performed the worst.

The above results were obtained by trial and error after repeated experiments by varying the ML model parameters to optimize the mean square error (MSE) and R² metrics. The optimal parameters which resulted in the best MSE and R² are displayed in Table 6.

TABLE 7. Performance of the 9 ML models in classifying the CPI with trimmed parameters.

ML Method	MSE	RMSE	MAE	R ²
Random Forest	0.016	0.125	0.101	0.450
SVM	0.022	0.148	0.123	0.230
kNN	0.019	0.137	0.118	0.340
Neural Networks	0.032	0.180	0.153	-0.140
Decision Tree	0.033	0.181	0.159	-0.160
Linear Regression	0.008	0.092	0.072	0.701
SGD	0.105	0.325	0.313	-2.724
AdaBoost	0.045	0.211	0.182	-0.578
Gradient Boosting	0.014	0.117	0.098	0.512

After trimming the number of features in the dataset, and only leaving as features the following 10 features which highly correlate with the CPI:

- OFFCORE_REQUESTS_OUTSTANDING.ALL _
DATA_RD:cmask=6,
- OFFCORE_REQUESTS_OUTSTANDING.CYCLES
_WITH_DATA_RD,
- UOPS_DISPATCHED_PORT.PORT_5,
- UOPS_EXECUTED.CORE:cmask=1,
- UOPS_RETIRED.RETIRE_SLOTSRETIRING,
- FRONT-END BOUND,
- LLC Miss,
- CORE BOUND,
- PORT UTILIZATION, and
- CPU UTILIZATION,

the 9 ML models performed as shown in Table 7. After pruning the features and using only the 10 above features, all ML classifiers dropped in performance except for LR and NN which improved. LR remained in the top spot as the best CPI classifier, with a slight performance drop in R² and MSE. One important consequence of training with only 10 features, is a large reduction in training time.

B. CLASSIFYING THE ELAPSED TIME

Next, classifying the normalized elapsed time (target), with only the following features which correlated well with the elapsed time:

- DTLB_LOAD_MISSES.WALK_DURATION,
- ICACHE.IFDATA_STALL,
- ILD_STALL.LCP,
- ITLB_MISSES.WALK_COMPLETED,
- L2_RQSTS.RFO_HIT,
- LD_BLOCKS_PARTIAL.ADDRESS_ALIAS, and
- MACHINE_CLEAR.S.COUNT,

the 9 ML models performed as displayed in Table 8.

NN emerged as the best classifier with a RMSE of 5 and R² of 0.64. The optimal NN parameter settings are 4 hidden layers with 80, 70, 30, and 20 neurons, respectively, and with ReLU activation, regularization=0.02 and Adam solver.

TABLE 8. Performance of the 9 ML models in classifying the normalized elapsed time.

ML Method	MSE	RMSE	MAE	R ²
Random Forest	55.271	7.434	5.620	0.204
SVM	76.524	8.748	7.175	-0.102
kNN	38.922	6.239	4.712	0.439
Neural Networks	25.215	5.021	3.595	0.637
Decision Tree	39.477	6.283	5.264	0.431
Linear Regression	46.574	6.825	5.402	0.329
SGD	47.313	6.878	5.252	0.318
AdaBoost	62.902	7.931	5.770	0.094
Gradient Boosting	75.726	8.702	6.531	-0.091

Again, the performance numbers of Table 8 were obtained after several runs to optimize the MSE and R² metrics.

VIII. CONCLUSION

In this work, the performance metrics and events counts for the AIBench and PassMark PerformanceTest benchmarks along with their components were captured. AIBench represents AI workloads with training and inference contents of deep neural networks such as MobileNet, Inception, ResNet, VGG, and LSTM. On the other hand, PassMark measures standard PC applications such as compression, encryption, sorting, 2D and 3D graphics, and memory and disk reads and writes. The events most highly correlated with the CPI and elapsed time were also identified.

Representing deep learning workloads with tests in classification, image to image mapping, image segmentation, sentiment analysis, and text translation, the AIBench was characterized by having among the highest levels of CPU utilization, microarchitectural usage, retiring, port utilization, back end bound, memory latency, and core bound percentages among the covered benchmarks, while ticking the lowest CPIs. In particular, the AIBench-Training generated the highest L1D and L2 cache pending, DTLB miss, and machine clear counts. The AIBench-Training had larger CPI, was more memory bound, and had less retiring and logical core utilization than its AIBench-Inference counterpart. Comparatively, CPU Mark exhibited the highest levels of threading and memory bandwidth. Memory Mark stressed the back end (memory, DRAM), and the memory bandwidth. The 3D Graphics Mark benchmark exhibited the highest L1 and L3 cache bound and core bound percentages. In contrast, the Disk Mark benchmark was the most front end bound, L1 and L3 cache bound, while exhibiting low CPU utilization and back end activities.

Based on the collected performance metrics and event counts, nine machine learning models were trained and tested to classify the CPI and elapsed time. In classifying the CPI, the linear regression classifier was the top classifier to minimize the MSE and maximize the R² metrics. In classifying the elapsed time, the neural network model with 4 hidden layers was the best performer.

This machine learning-based classification can help predict the CPI and elapsed time of profiled applications and distinguish between applications with and without deep learning content based on captured performance metrics and event counts. Additionally, predicting the workload performance based on captured performance event counts can have other uses such as activating dormant hardware units when desired, and capacity planning in clouds. Furthermore, the above findings identified the stressed and overloaded hardware units and can help guide future CPU and accelerator hardware design enhancements for the various workloads.

Another key contribution of this study lies in its ability to integrate insights from both data-centric and model-centric AI perspectives, offering a comprehensive understanding of AI workload characteristics and hardware utilization patterns, and exemplifying the synergy between data-centric and model-centric AI methodologies.

It is important to recall that performance enhancements are not free, for instance, FP4 arithmetic boosts floating-point performance at the expense of precision and/or accuracy, while general platform enhancements may be accompanied by increased costs and power consumptions. One limitation of this study is that, although the general workload signatures and identified stressed units apply across computing platforms, the actual performance numbers are specific to the Intel Haswell processor. Future work includes extending this work to other PC processors, and to mobile computing platforms.

ACKNOWLEDGMENT

The authors wish to thank GUST for covering the APC fee for the publication of their work.

REFERENCES

- [1] F. N. Sibai, "Performance analysis and workload characterization of the 3DMark05 benchmark on modern parallel computer platforms," *ACM SIGARCH Comput. Archit. News*, vol. 35, no. 3, pp. 44–52, Jun. 2007.
- [2] F. N. Sibai, "3D graphics performance scaling and workload decomposition and analysis," in *Proc. 6th IEEE/ACIS Int. Conf. Comput. Inf. Sci. (ICIS)*, Jul. 2007, pp. 604–609.
- [3] *SPEC Benchmarks*. [Online]. Available: <https://www.spec.org>
- [4] F. N. Sibai, "Evaluating the performance of single and multiple core processors with PCMARK[®]05 and benchmark analysis," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 35, no. 4, pp. 62–71, Mar. 2008.
- [5] F. Sibai, "Dissecting the PCMark[®]05 benchmark and assessing performance scaling," in *Proc. Innov. Inf. Technol.*, Dubai, United Arab Emirates, Nov. 2006, pp. 1–5.
- [6] F. N. Sibai, "Gauging the OpenSourceMark benchmark in measuring CPU performance," in *Proc. 7th IEEE/ACIS Int. Conf. Comput. Inf. Sci. (ICIS)*, Portland, OR, USA, May 2008, pp. 433–438.
- [7] F. N. Sibai, "On the performance benefits of sharing and privatizing second and third level cache memories in homogeneous multi-core architectures," *Microprocess. Microsyst., Embedded Hardw. Design*, vol. 32, no. 7, pp. 405–412, Oct. 2008.
- [8] A. Asaduzzaman, I. Mahgoub, and F. N. Sibai, "Impact of L1 entire locking and L2 way locking on the performance, power consumption, and predictability of multicore real-time systems," in *Proc. IEEE/ACS Int. Conf. Comput. Syst. Appl.*, Morocco, May 2009, pp. 705–711.
- [9] F. N. Sibai, "Simulation and performance analysis of multi-core thread scheduling and migration algorithms," in *Proc. Int. Conf. Complex, Intell. Softw. Intensive Syst.*, Krakow, Poland, Feb. 2010, pp. 895–900.

- [10] A. Asaduzzaman, F. N. Sibai, and H. El-Sayed, "Performance and power comparisons of MPI vs pthread implementations on multicore systems," in *Proc. 9th Int. Conf. Innov. Inf. Technol. (IIT)*, Mar. 2013, pp. 1–6.
- [11] A. Asaduzzaman, A. Trent, S. Osborne, C. Aldershof, and F. N. Sibai, "Impact of CUDA and OpenCL on parallel and distributed computing," in *Proc. 8th Int. Conf. Electr. Electron. Eng. (ICEEE)*, Turkey, Apr. 2021, pp. 238–242.
- [12] PassMark. *PerformanceTest 11.0*. [Online]. Available: <https://www.PassMark.com/products/performance-test/index.php>
- [13] P. Mattson, V. J. Reddi, C. Cheng, C. Coleman, G. Damos, D. Kanter, P. Micikevicius, D. Patterson, G. Schmuelling, H. Tang, G.-Y. Wei, and C.-J. Wu, "MLPerf: An industry standard benchmark suite for machine learning performance," *IEEE Micro*, vol. 40, no. 2, pp. 8–16, Mar. 2020, doi: [10.1109/MM.2020.2974843](https://doi.org/10.1109/MM.2020.2974843).
- [14] V. J. Reddi et al., "MLPerf inference benchmark," in *Proc. ACM/IEEE 47th Annu. Int. Symp. Comput. Archit. (ISCA)*, Valencia, Spain, May 2020, pp. 446–459.
- [15] *BenchCouncil AIBench*. [Online]. Available: <http://www.benchcouncil.org/aibench>
- [16] *AIBench Tutorial*. [Online]. Available: <https://www.benchcouncil.org/aibench-tutorial/aspl0s21>
- [17] J. Demšar and B. Zupan, "Orange: Data mining fruitful and fun—A historical perspective," *Informatica*, vol. 37, no. 1, pp. 55–60, 2013.
- [18] Z. Quan, X. Chen, and Y. Han, "AIC-bench: Workload selection methodology for benchmarking AI chips," in *Proc. IEEE 24th Int. Conf. High Perform. Comput. Communications; 8th Int. Conf. Data Sci. Syst., 20th Int. Conf. Smart City, 8th Int. Conf. Dependability Sensor; Cloud Big Data Syst. Appl. (HPCC/DSS/SmartCity/DependSys)*, Hainan, China, Dec. 2022, pp. 687–694, doi: [10.1109/HPCC-DSS-SmartCity-DependSys57074.2022.00117](https://doi.org/10.1109/HPCC-DSS-SmartCity-DependSys57074.2022.00117).
- [19] F. Tang et al., "AIBench training: Balanced industry-standard AI training benchmarking," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS)*, Stony Brook, NY, USA, Mar. 2021, pp. 24–35.
- [20] *AIBench Training*. [Online]. Available: <https://www.benchcouncil.org/aibench-training/index.html>
- [21] *Benchmarking GPUs for Machine Learning*. [Online]. Available: <https://www.ml4au.community/tools/benchmarking-gpus>
- [22] Y. Wang, Q. Wang, S. Shi, X. He, Z. Tang, K. Zhao, and X. Chu, "Benchmarking the performance and energy efficiency of AI accelerators for AI training," in *Proc. 20th IEEE/ACM Int. Symp. Cluster, Cloud Internet Comput. (CCGRID)*, Melbourne, VIC, Australia, May 2020, pp. 744–751, doi: [10.1109/CCGrid49817.2020.00-15](https://doi.org/10.1109/CCGrid49817.2020.00-15).
- [23] M. Emami et al., "A comprehensive evaluation of novel AI accelerators for deep learning workloads," in *Proc. IEEE/ACM Int. Workshop Perform. Model., Benchmarking Simul. High Perform. Comput. Syst. (PMBS)*, Dallas, TX, USA, Nov. 2022, pp. 13–25.
- [24] Z. Jiang, W. Gao, F. Tang, L. Wang, X. Xiong, C. Luo, C. Lan, H. Li, and J. Zhan, "HPC AI500 V2.0: The methodology, tools, and metrics for benchmarking HPC AI systems," in *Proc. IEEE Int. Conf. Cluster Comput. (CLUSTER)*, Portland, OR, USA, Sep. 2021, pp. 47–58.
- [25] T. Hao, K. Hwang, J. Zhan, Y. Li, and Y. Cao, "Scenario-based AI benchmark evaluation of distributed cloud/edge computing systems," *IEEE Trans. Comput.*, vol. 72, no. 3, pp. 719–731, Mar. 2023.
- [26] W. Dai and D. Berleant, "Benchmarking contemporary deep learning hardware and frameworks: A survey of qualitative metrics," in *Proc. IEEE 1st Int. Conf. Cognit. Mach. Intell. (CogMI)*, Los Angeles, CA, USA, Dec. 2019, pp. 148–155.
- [27] F. Al-Ali, T. D. Gamage, H. W. Nanayakkara, F. Mehdipour, and S. K. Ray, "Novel casestudy and benchmarking of AlexNet for edge AI: From CPU and GPU to FPGA," in *Proc. IEEE Can. Conf. Electr. Comput. Eng. (CCECE)*, London, ON, Canada, Aug. 2020, pp. 1–4.
- [28] M.-Y. Wang, J. Uran, and P. Buitrago, "Deep learning benchmark studies on an advanced AI engineering testbed from the open compass project," in *Proc. Pract. Exper. Adv. Res. Comput.*, Portland, OR, USA, Jul. 2023, pp. 1–5.
- [29] E. Davis, "Benchmarks for automated commonsense reasoning: A survey," *ACM Comput. Surv.*, vol. 56, no. 4, pp. 1–41, Oct. 2023.
- [30] A. Arora, A. Boutros, D. Rauch, A. Rajen, A. Borda, S. A. Damghani, S. Mehta, S. Kate, P. Patel, K. B. Kent, V. Betz, and L. K. John, "Koios: A deep learning benchmark suite for FPGA architecture and CAD research," in *Proc. 31st Int. Conf. Field-Programmable Log. Appl. (FPL)*, Dresden, Germany, Aug. 2021, pp. 355–362, doi: [10.1109/FPL53798.2021.00068](https://doi.org/10.1109/FPL53798.2021.00068).
- [31] *TensorFlow Installation*. [Online]. Available: <https://www.tensorflow.org/install>
- [32] *AIBench Installation*. [Online]. Available: <https://pypi.org/project/aibenchmark/>
- [33] *Intel Perfmom Events*. [Online]. Available: <https://perfmon-events.intel.com/>
- [34] S. Ward-Foxton. (2021). *Solving the AI Memory Bottleneck*. EE Times. [Online]. Available: <https://www.eetimes.com/solving-ais-memory-bottleneck/>
- [35] O. H. Hamid, "Data-centric and model-centric AI: Twin drivers of compact and robust Industry 4.0 solutions," *Appl. Sci.*, vol. 13, no. 5, p. 2753, Feb. 2023.



FADI N. SIBAI received the B.S. degree in electrical (computer) engineering from The University of Texas, Austin, and the M.S. and Ph.D. degrees in electrical (computer) engineering from Texas A&M University. He joined GUST as the Associate Dean of the College of Engineering and Architecture in 2022. Prior to that, he served as the acting Dean of the School of Engineering, American International University, the Dean of the College of Computer Engineering and Science, Prince Mohammad Bin Fahd University, and the Program Director of the College of Information Technology, UAE University. He also taught Engineering at the University of California, San Jose State University, and The University of Akron, USA. He also has extensive industrial experience, having worked for Aramco and Intel Corporation. He authored or coauthored more than 250 publications and technical reports. He also served in various capacities on program and organizing committees of more than 20 international conferences. He is a member of PMI and Eta Kappa Nu. He received IBM and nVIDIA equipment grants, the IBM Faculty award, and research grants from the Emirates Foundation and The University of Akron.



ABU ASADUZZAMAN (Senior Member, IEEE) received the M.S. and Ph.D. degrees in computer engineering from Florida Atlantic University, Florida. Currently, he is an Associate Professor of computer engineering with Wichita State University, Kansas. He has authored more than 20 refereed journals and more than 80 peer-reviewed conference papers out of his research work. His research interests include high-performance computing, machine learning, and data analysis.

He has received research grants from Kansas NSF EPSCoR, Nvidia, and NetApp. As an Invited Speaker, he has presented his research work in many professional forums at institutions, including the Nara Institute of Science and Technology, Japan, the Old Dominion University, Norfolk, VA, USA, the International Society for Engineering Research and Development, Thailand, and the IEEE Wichita Professional Section in Kansas. He serves as the Undergraduate Program Director in his department. He has served as a reviewer of NSF programs and IEEE journals.



ALI EL-MOURSY (Senior Member, IEEE) received the Ph.D. degree from the University of Rochester, USA. He is currently a Professor and the Head of the Computer Engineering Department, University of Sharjah, United Arab Emirates. His research interests include high-performance computer architecture, multi-core multi-threaded micro-architecture, high-performance computing, parallel computing, cloud computing, architecture modeling and simulation, and performance evaluation. Previously, he worked with the Software Solution Group, Intel Corporation, Santa Clara, CA, USA, and the Electronics Research Institute, Egypt.