MIST: Many-ISA Scheduling Technique for Heterogeneous-ISA Architectures

Prakhar Diwan[1], Suryakant Toraskar[1], Varun Venkitaraman[1], Nirmal Kumar Boran*[,1,2], Chandramani Chaudhary[2] and Virendra Singh[1]

[1]Indian Institute of Technology Bombay, India
{prakhar.diwan,suryamt,varunvenkat,viren}@ee.iitb.ac.in

[2]National Institute of Technology Calicut, India
{nirmalkboran,chandramanic}@nitc.ac.in

***Abstract*—Heterogeneous chip multiprocessors have shown significant promise in achieving better performance and energy efficiency than their homogeneous counterparts. Instruction set architecture (ISA) as a heterogeneity dimension has become increasingly popular in recent years due to the significant performance gains it offers. In order to achieve maximum benefits from heterogeneous-ISA architectures, a program must be divided into phases, with each phase then dynamically scheduled on the most affine ISA core. Determining the affine ISA for every program phase from numerous ISAs at runtime is crucial, as a larger set of ISAs enhances the potential of heterogeneous-ISA architectures. This work proposes MIST, a classification-based scheduling algorithm with minimal hardware overhead for many (more than two) ISA scenarios. It utilizes a smaller set of microarchitectural parameters obtained via online performance counters to determine the most affine ISA for upcoming program phases. Overall, MIST achieves a prediction accuracy of 88.1%, resulting in a 40.4% improvement in single-threaded performance compared to x86 ISA architecture. It achieves 9.9% additional performance gain compared to the current state-of-the-art scheduling technique.**

***Index Terms*—Chip multiprocessors, Dynamic scheduling, ISA, Heterogeneous architectures, Single-threaded performance**

I. Introduction

During the latter half of the previous century, the central aim of microprocessor technology was to achieve better performance. Performance was improved through extensive research in VLSI technology and computer architecture domains. With transistors shrinking in size due to technology advancements, significant performance gains were achieved due to larger transistor counts (more resources) and higher operating frequencies of processors. However, the transistor's threshold voltage drop saturated across technology nodes, which constrained frequency scaling due to increasing power dissipation. This halted the performance growth of cores from technology improvements.

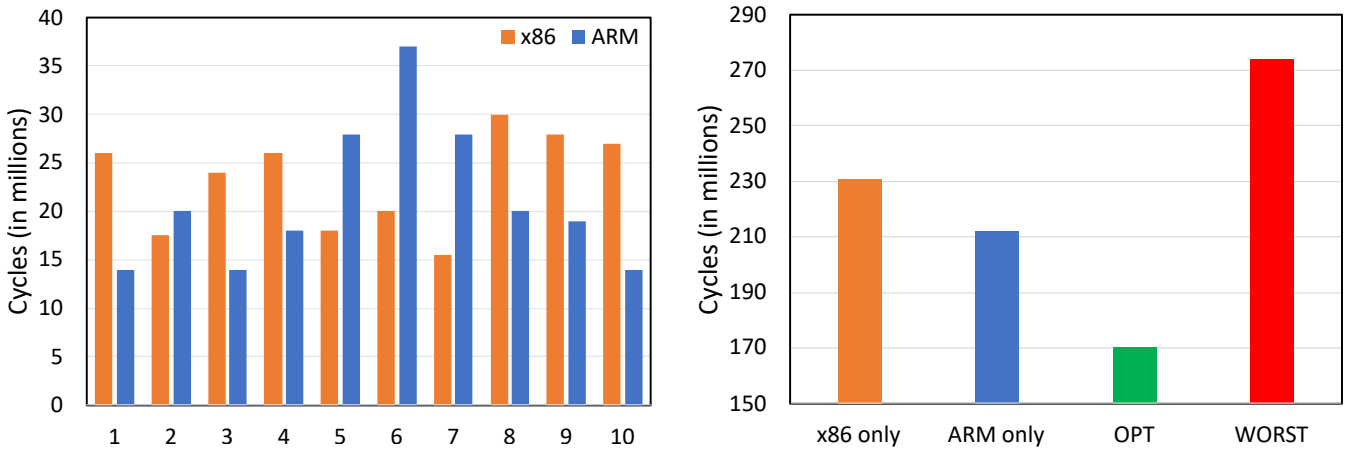
To further drive performance growth, multiple identical cores were integrated on the same chip to provide enhanced throughput for parallel workloads. However, single-threaded performance remained a bottleneck to overall performance highlighted by Amdahl's law [1], [2]. For improving single-threaded performance, various directions were pursued by computer architects, one such direction being dynamic multi-core architectures [3], [4], where single-threaded performance was improved by exploiting instruction-level parallelism (ILP) through a larger core (created by merging cores dynamically).

*Corresponding Author

Another popular direction remains heterogeneous computing, which utilizes cores of different performance and power characteristics to effectively cater to program phases. Workload is migrated to the best core (in terms of meeting phase-wise requirements) at runtime. Various heterogeneity dimensions such as core sizes [5], [6] and execution semantics [7], [8] have been utilized to achieve performance and energy efficiency gains over homogeneous multicore systems.

However, these architectures were restricted to a single ISA, which restrained the potential benefits, as ISAs are designed with diverse goals, such as reducing overall memory access and decreasing code size. DeVuyst et al. [9] presented migration techniques to facilitate dynamic migration of programs between different ISA cores. Venkat et al. [10] introduced heterogeneous-ISA chip multiprocessor (HISACMP) consisting of different ISA cores on the same chip. They observed that different program phases performed better on different ISAs, and referred to this as ISA affinity. This arises from factors such as register pressure, code density and dynamic instruction count, which vary across program execution on different ISAs.

We executed the *sjeng* benchmark from SPEC CPU2006 [11] to explore ISA affinity using ARM and x86 ISA cores. From Figure 1a, we observed that phases 2, 5, 6 and 7 are affine to x86 ISA, whereas others are affine to ARM ISA. Hence, running the program exclusively using either ISA is sub-optimal, as different program phases exhibit distinct behavior. From Figure 1b, we observe that with optimal (OPT) scheduling (always affine ISA used), a speedup of up to 24.7% (relative to best single ISA core) can be achieved. However, incorrect (WORST) scheduling (always non-affine ISA used) severely degrades performance up to 22.6% (relative to the best single ISA core). Therefore, scheduling is critical in Heterogeneous-ISA architectures. Heterogeneous-ISA CMPs [10] achieved significant performance benefits; however, they utilized oracular scheduling, which is impractical. To realize the benefits of ISA affinity, hardware-based scheduling mechanisms have been proposed. It has been observed that the most affine ISA core for each program phase can be predicted from mathematical models involving microarchitectural parameters [12], [13]. Perceptron-based scheduling mechanism [13] offers high prediction accuracy ($> 90\%$) for the dual ISA scenario. A many-ISA (larger number of ISAs) scenario is highly favorable since it provides more diversity and better potential performance benefits. Scheduling in many-ISA scenario [12] is more general compared to dual ISA scenario [13]. Boran et



(a) Execution time of different phases from *sjeng*

(b) Total execution time for various scheduling methods

Fig. 1: Impact of scheduling in heterogeneous-ISA Architectures

al. [12] employed a linear regression neural network (LRNN) based method to tackle this problem. We propose a many-ISA scheduling technique (MIST), a decision tree based approach to get higher prediction accuracy for many-ISA scenarios with lower hardware overheads than prior works [12], [13].

Paper organization: Section II describes prior related work, Section III details the proposed work and Section IV illustrates our evaluation methodology. Section V presents our analysis and results. Section VI concludes and provides future work.

II. Previous Work

Abundant amount of literature exists related to the proposed work; hence, we split this into two categories: first describes heterogeneous architectures and the second pertains to dynamic scheduling methods used in them.

A. Heterogeneous Architectures

Kumar et al. [6] demonstrated that heterogeneous multicore architectures, which consist of cores with different power and performance characteristics, offer substantial power reduction compared to homogeneous counterparts. This heterogeneous computing approach is widely utilized in industrial designs, exemplified by ARM's big.LITTLE [8] architecture. A refinement to this architecture was proposed by Lukefahr et al. [7], where they utilized finer migration opportunities between different execution semantics (in-order, out-of-order) to increase energy savings. Pricopi et al. [4] proposed the 'Bahurupi' architecture, which improved performance and energy efficiency over homogeneous multicore systems by dynamically meeting the diverse instruction-level parallelism (ILP)- thread-level parallelism (TLP) requirements of modern applications. However, these works were restricted to single ISA cores with microarchitectural differences.

Various works [14], [15] have analyzed the impact of ISA (CISC vs. RISC) in different processors, concluding similar energy efficiency and performance results. These claims were weak due to infrastructural limitations and no ISA-specific compiler optimizations for the evaluation. However, we utilize Heterogeneous-ISA CMP [10], which consists of different ISA cores built on heterogeneous hardware. In order to obtain maximum gains from ISA affinity, the process must be able to migrate freely between different ISA cores. However, cross-ISA process migration is a challenging problem [16].

DeVuyst et al. [9] demonstrated compiler and runtime techniques to enable cross-ISA migration with minimal changes to the compiler back end. They achieved a high density of equivalence points throughout binary execution, where the program's memory image was consistent across different ISAs executables. Switching at these points reduced the migration overhead, making dynamic cross-ISA migration feasible in heterogeneous-ISA CMPs. Building on this, Venkat et al. [10] performed an extensive design space exploration and showcased significant performance and power consumption improvements through heterogeneous-ISA architectures over single ISA heterogeneous architectures.

B. Dynamic Scheduling in Heterogeneous Architectures

Various techniques have been proposed in the literature to determine the optimal core in heterogeneous architectures for the execution of program phases. Kumar et al. [6], [17] predicted optimal core among different cores through a sampling-based technique, where small portions of the phase execute on each of the different cores, and then the remaining portion ran on the best-determined core. However, this approach is not scalable and performs poorly due to dynamically changing program behavior. To overcome this, researchers [18], [19] have explored modeling the parameter "cycles per instruction" (CPI) for the inactive cores. Craeynest et al. [20] have utilized MLP (memory-level parallelism) and ILP parameters to develop a regression-based performance impact estimator. This estimator uses microarchitectural parameters to determine the performance of inactive cores for the program phases. It was further employed by Lukefahr et al. [7] and Pricopi et al. [21]. However, these techniques were particularly suited to single-ISA heterogeneous architectures, providing sub-optimal performance for heterogeneous ISA architectures.

Various scheduling schemes [12], [13], [22], [23] have been proposed for heterogeneous ISA architectures. Boran et al. [23] presented a general regression-based model to predict the execution time of program phases on inactive ISA cores based on the set of microarchitectural parameters observed for the active ISA core's execution. First, microarchitectural parameters for the inactive-ISA core were estimated from the active-ISA core's parameters. Then the execution time for the inactive-ISA core was obtained using the estimated parameters. This two-step approach was, however, shown to be unnecessary, with a linear regression-based approach [12] providing better accuracy using lesser hardware for the many-ISA problem. Further, the perceptron-based solution [13] achieved more than 90% migration decision accuracy. However, it was optimized for the dual ISA scenario. Performance and power benefits from ISA affinity magnify upon using more diverse ISAs [10]. Hence, optimal scheduling is critical for heterogeneous ISA architectures with numerous ISAs. This work aims to tackle this problem by proposing MIST, a low hardware overhead scheduling technique based on decision trees. It accurately predicts the program phases to the most affine ISA. We use LRNN [12] as the baseline for the proposed work.

III. Proposed Work

Each program phase must be carefully scheduled on the most affine ISA core to maximize performance benefits from ISA heterogeneity. We assume that the groups of phases affine to the different ISAs are linearly separable and view scheduling as a classification problem. Boran et al. [13] proposed a perceptron algorithm for classifying program phases between ARM and x86 ISAs based on ISA affinity. It achieved high accuracy ($> 90\%$) for binary classification. However, considering that more and diverse ISAs further increase the potential performance gains for heterogeneous-ISA architectures [10]. This makes dynamic scheduling between more ISAs (> 2) a critical and more generic problem. We propose a many-ISA scheduling technique (MIST), which uses decision trees for scheduling the program phases on the most affine ISA. The proposed algorithm for dynamic scheduling is realized as a hardware-based mechanism. MIST improves over perceptron [13] and LRNN [12] for the many-ISA case by achieving higher prediction accuracy with lower hardware overhead (by using lesser microarchitectural parameters for making decisions). LRNN-based scheduler [12] is used as the baseline for this work and is trained with the corresponding larger set of microarchitectural parameters. We also compare the migration decision accuracy with perceptron [13], trained for many-ISA cases. We consider three ISAs for evaluating our proposal: ARM(v7), x86 and Alpha.

As modern applications display varying requirements across execution, they are split into multiple phases, and each phase is allocated to the appropriate core in heterogeneous architectures. For ISA heterogeneity, the phase length (number of dynamic instructions) must be large enough to amortize the migration overheads but small enough to utilize the benefits of ISA affinity effectively. We use phase length of ≈100 million dynamic instructions as in prior works [9], [10], [12], [13], but with switching only done at equivalence points [12], [13]. This is shown in Fig 2. We use x86 ISA as a reference for obtaining the phases based on dynamic instruction count and take equivalent phases for ARM and Alpha ISAs. The distance between two consecutive migration points ($E^i{}_{\text{x86}}$ and $E^{i+1}{}_{\text{x86}}$) is ≈100 million dynamic instructions for x86 ISA execution. Equivalent migration points for other ISAs like ARM ($E^i{}_{\text{ARM}}$ and $E^{i+1}{}_{\text{ARM}}$) are obtained by mapping migration points in x86 ISA execution to high-level language (HLL), and then mapping these points to corresponding points in ARM and Alpha ISA executions. At the end of each phase, the scheduler makes the migration decision.

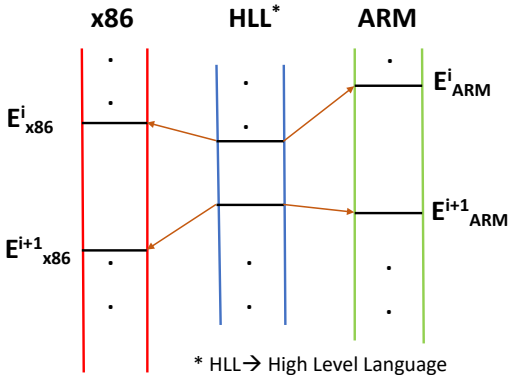


Fig. 2: The program state is consistent across different ISAs at equivalence points. We keep them ≈100M instructions apart at function call sites in compiler intermediate representation.

A. Extraction of Microarchitectural Parameters

Parameter	Purpose
L1-I misses	Stalls due to cache misses
L1-D misses	
L2 misses	
ROB full events	Execution pipeline stalls
IQ full events	
SQ full events	
ILP	Parallelism of the executing program
MLP	
Branch mispredictions	Performance impact of branch predictor
Float instruction count	ISA-specific parameters

TABLE I: Microarchitectural parameters used by MIST

The proposed algorithm utilizes ten microarchitectural parameters to classify the program phases to their most affine ISA. The microarchitectural parameters are listed in Table I with their purpose. These parameters are extracted during the simulation of benchmarks, and the model is trained and tested accordingly. We use fewer microarchitectural parameters than perceptron, the state-of-the-art scheduler for dual-ISA case [13] and LRNN-based scheduler for many-ISA case [12].

Modern processors already house performance counters, i.e., the necessary hardware to track most of these features [24], [25]. We calculate the more intricate features like

ILP (instruction-level parallelism) and MLP (memory-level parallelism) in a similar way as described in [7]. ILP is approximated as the inverse of the number of instructions in the issue stage of the core's pipeline, which are delayed due to dependencies on other incomplete instructions. If these dependencies did not exist, no instructions would be delayed, indicating perfect parallelism. On the other hand, MLP is estimated using a counter that keeps track of the entries in the miss status handling register (MSHR). This counter is updated whenever there is a cache miss, representing the number of ongoing memory accesses processed in parallel.

B. Scheduling Model

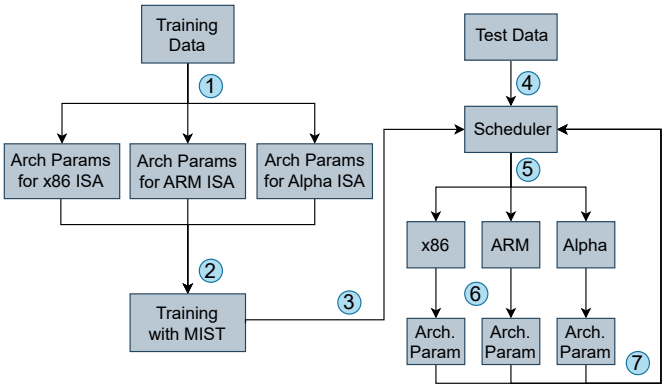


Fig. 3: A schematic representation of the scheduling model

The flow diagram presented in Figure 3 illustrates the schematic representation of our scheduling model's workflow. Each benchmark was divided into two distinct sets of data: training and test data, to ensure their mutual exclusivity. Step 1 in Figure 3 represents the extraction of parameters used as input for training MIST. These parameters are utilized by MIST to predict the affinity of a program phase towards a specific ISA. It incorporates the migration overhead of switching in its decision making, switching only if overall performance is improved. Through experimentation, we have determined that analyzing 10 million instructions is sufficient to determine the affinity of the subsequent phase, consisting of 100 million instructions. Therefore, based on the microarchitectural parameters derived from the last 10 million instructions of the current phase, the scheduler (MIST) predicts the affinity of the next phase.

We assume that the program's behavior will remain consistent during the execution of subsequent phase. In the scheduling model depicted in Figure 3, steps 1 to 3 are conducted offline, while steps 4 to 7 are performed online dynamically. The scheduler determines the migration between ISAs for the initial phase using the initial training data (step 2). This is denoted by step 5. Subsequently, based on the analysis of this phase's last 10 million instructions (step 6), it makes migration decisions for the following phase, which consists of 100 million instructions (step 7).

C. MIST: a decision-tree based approach

Decision tree algorithm was chosen for multi-class classification based on it's high accuracy of prediction and low complexity. The decision tree recursively splits the data based on the best features and thresholds, aiming to create partitions that are as pure as possible. The tree traversal starts at the root node, follows the branches based on feature values, and ends at leaf nodes that provide the final predictions or classifications. It does not involve weight updates like back-propagation and can be trained in a single pass through the data. Feature selection and pre-processing are essential steps to ensure effective decision tree construction. Feature set was finalized by iteratively testing all the features (from [13]) and filtering based on their impact on prediction accuracy. Training dataset was obtained offline by tagging each phase i.e. corresponding feature set value with the most affine ISA.

We employed the Gini index as the basis for splitting in decision trees. The tree's depth is contingent on achieving pure leaves or when leaf nodes contain fewer than 2 samples. To split an internal node, a minimum of 2 samples is required, while a leaf node can contain as few as 1 sample.

IV. Evaluation Setup

We consider three ISAs in this work - ARM(v7), Alpha and x86. Benchmarks were compiled for x86 and cross-compiled for ARM and Alpha similar to [9], [10], [12], [13]. The configurations of the three different ISA cores are shown in Table II. We modeled heterogeneous-ISA CMP using these cores, each with a private 2-level cache hierarchy [10]. The clock rate for all the cores is set to 2GHz. We analyzed SPEC CPU2006 [11] benchmarks on the three cores, simulated using gem5 architectural simulator [26]. Phase length and migration overhead are taken as in prior works [9], [10], [12], [13].

Design Parameter	ARM	Alpha	x86
Architectural Registers	32 GPR	64 GPR	16 GPR
Cache line size(bytes)	64	64	64
LSQ size (bytes)	32	32	32
Superscalar width	4	4	4
Instruction Queue entries	64	64	64
ROB entries	192	192	192
DCache,ICache size	32KB	32KB	32KB
L2 Cache size	256KB	256KB	256KB
SIMD Support	No	Yes	Yes

TABLE II: Core configurations

V. Results

We present the results for scheduling techniques implemented for heterogeneous-ISA chip multiprocessor (HISACMP). We consider LRNN [12], perceptron [13], MIST and oracle. Oracle indicates the perfect assignment of all phases to corresponding optimal ISA cores.

A. Migration Decision Accuracy

We define *accuracy* as the fraction of time when the scheduler chooses the most affine ISA. Migration decision

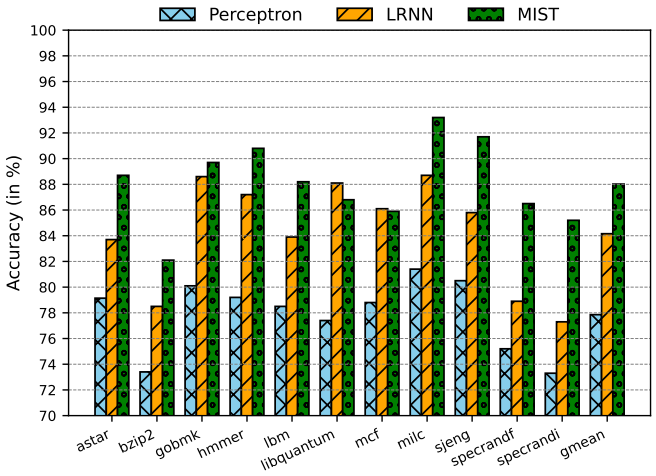


Fig. 4: Migration decision accuracy for different benchmarks

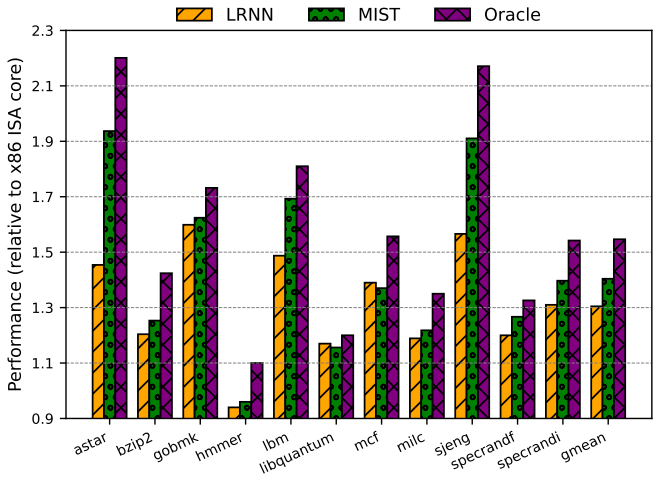


Fig. 5: Comparison of single-threaded performance for HISACMP with MIST, LRNN and Oracle schedulers

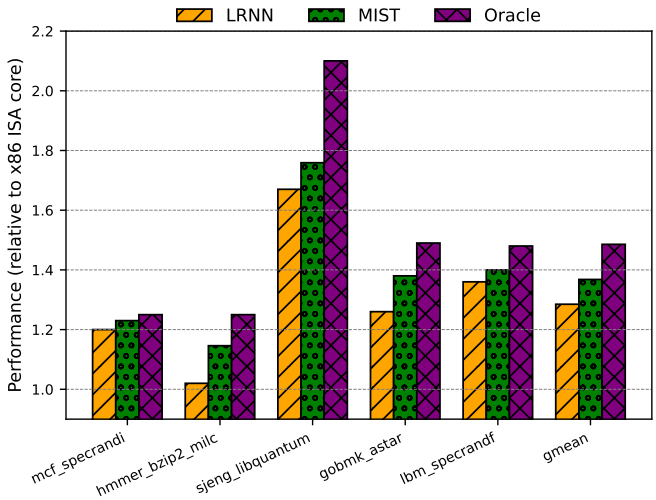


Fig. 6: Performance gains when training and testing is done using different data sets

accuracies of LRNN [12], perceptron [13] and MIST are plotted in Figure 4. MIST predicted the most affine ISA with an accuracy of 88.1%, performing better than LRNN and perceptron, which show accuracies of 84.2% and 77.8%, respectively. This leads to performance improvement with MIST, as shown in the next section. We show comparisons with LRNN from the following section onwards as it predicts better than perceptron for the many-ISA scenario. Perceptron performs poorly for the many-ISA case compared to MIST, as it requires more data for training, along with a higher amount of pre-processing. Also, we observed that the training time in the decision tree algorithm was significantly reduced compared to LRNN and perceptron.

B. Performance Results

The impact of MIST on performance was analyzed through two investigations. First, we trained MIST with data obtained from 70% phases and then tested MIST on the remaining 30% phases for each benchmark. MIST, built as a classification-based algorithm, directly assigns the affine ISA to the program phase. In contrast, LRNN uses regression, first approximating the execution time and then deciding affine ISA based on a fixed threshold. The direct assignment in classification helps, as noted in prior work [13]. We observe from Figure 5 that MIST improves single-threaded performance by 40.4% relative to only x86 ISA execution, which is 9.9% higher than LRNN [12].

Second, to assess the robustness of MIST, we trained it with data from a subset of SPEC CPU2006 benchmarks [11] and then tested it on an unexposed set of benchmarks. Figure 6 depicts this analysis, with benchmarks used only for testing depicted on the x-axis. Overall, MIST achieves a significant improvement of 36.8% compared to LRNN's 28.5% relative to x86 ISA. It establishes the robust nature of proposed technique.

C. Energy Results

We analyzed the energy consumption with MIST and LRNN scheduling techniques, and Figure 7 shows the same. We used McPAT [27] to obtain phase-wise power consumption for different ISA cores. Overall, HISACMP achieves 1.43% lesser energy consumption with MIST compared to the case where LRNN scheduling was used. The energy benefits are minimal as the proposed scheduling is optimized for performance.

D. Hardware Overhead

The proposed technique, MIST, has lesser hardware overhead than earlier techniques [12], [13] as it uses fewer microarchitectural parameters than them. This reduces the number of registers from 14 [12] and 12 [13] to 10 for storing the weight matrix. One register stores the bias. MIST requires an 8-bit multiplier, a 32-bit adder, a comparator and a 32-bit register (to store intermediate values) to perform computation serially.

VI. Conclusion & Future Work

In the context of heterogeneous CMPs, instruction set architecture (ISA) has gained popularity as a dimension for

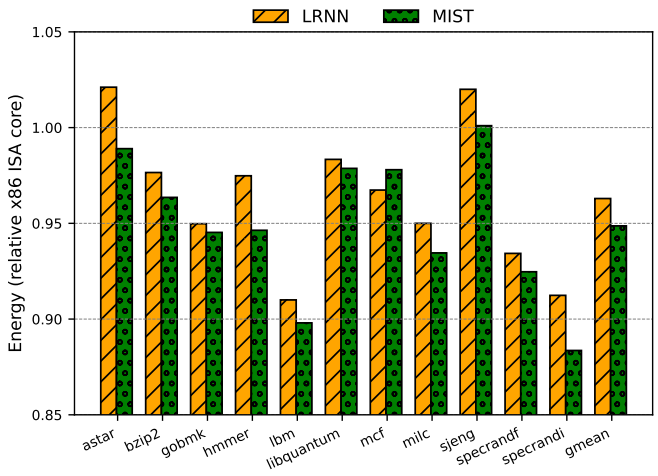


Fig. 7: Comparison of energy consumption for HISACMP using MIST and LRNN-based schedulers

improving performance and energy efficiency. Scheduling program phases to the optimal core in such architectures is critical to extracting the potential gains. This work proposes MIST, a scheduling algorithm with minimal hardware overhead for many-ISA scenarios involving more than two ISAs. MIST utilizes a smaller set of microarchitectural parameters obtained via performance counters to determine the most affine ISA for upcoming program phases. It achieves a prediction accuracy of 88.1%, leading to a 40.4% improvement (on average) in single-thread performance relative to only x86 ISA execution. It achieves 9.9% more gain compared to LRNN-based scheduling [12] for heterogeneous-ISA CMPs, which outperforms perceptron [13] for many-ISA cases. Overall, it leverages ISA affinity with dynamic scheduling to achieve substantial performance gains over heterogeneous-ISA architectures. As these architectures enhance energy efficiency as well, scheduling aimed for it is critical and remains part of our future work.

Acknowledgement

This work was supported in part by Indo Japanese Joint Lab Grant and AI powered adaptive cyber defence framework sponsored by NSCS, Government of India.

References

[1] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, ser. AFIPS '67 (Spring). New York, NY, USA: ACM, 1967, p. 483–485.

[2] M. D. Hill et al., "Amdahl's law in the multicore era," *Computer*, vol. 41, no. 7, pp. 33–38, 2008.

[3] E. Ipek et al., "Core fusion: Accommodating software diversity in chip multiprocessors," ser. ISCA '07. New York, NY, USA: ACM, 2007, p. 186–197.

[4] M. Pricopi et al., "Bahurupi: A polymorphic heterogeneous multi-core architecture," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 8, no. 4, Jan 2012.

[5] R. . Kumar et al., "Single-isa heterogeneous multi-core architectures for multithreaded workload performance," in *Proceedings. 31st Annual International Symposium on Computer Architecture, ISCA 2004.*, 2004, pp. 64–75.

[6] R. Kumar et al., "Single-isa heterogeneous multi-core architectures: the potential for processor power reduction," in *Proceedings. 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003. MICRO-36.*, 2003, pp. 81–92.

[7] A. Lukefahr et al., "Composite cores: Pushing heterogeneity into a core," in *2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, 2012, pp. 317–328.

[8] B. Jeff et al., "Big.little system architecture from arm: saving power through heterogeneous multiprocessing and task context migration," in *Proceedings of the 49th Annual Design Automation Conference*. ACM, 2012, pp. 1143–1146.

[9] M. DeVuyst et al., "Execution migration in a heterogeneous-ISA chip multiprocessor," in *Proceedings of the Seventeenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS XVII. New York, NY, USA: ACM, 2012, pp. 261–272.

[10] A. Venkat et al., "Harnessing ISA diversity: Design of a heterogeneous-ISA chip multiprocessor," in *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*, June 2014, pp. 121–132.

[11] J. L. Henning, "SPEC CPU2006 benchmark descriptions," *SIGARCH Comput. Archit. News*, Sep. 2006.

[12] N. K. Boran et al., "Performance modelling and dynamic scheduling on heterogeneous-ISA multi-core architectures," in *International Symposium on VLSI Design and Test*. Springer, 2019, pp. 702–715.

[13] N. Boran et al., "Classification based scheduling in heterogeneous ISA architectures," in *2020 24th International Symposium on VLSI Design and Test (VDAT)*. IEEE, 2020, pp. 1–6.

[14] R. P. Colwell et al., "Computers, complexity, and controversy," *Readings in computer architecture*, p. 144, 2000.

[15] E. Blem et al., "Power struggles: Revisiting the risc vs. cisc debate on contemporary arm and x86 architectures," in *High Performance Computer Architecture (HPCA2013), 2013 IEEE 19th International Symposium on*. IEEE, 2013, pp. 1–12.

[16] P. Smith et al., "Heterogeneous process migration: The tui system," *Software: Practice and Experience*, vol. 28, no. 6, pp. 611–639, 1998.

[17] R. Kumar et al., "Processor power reduction via single-isa heterogeneous multi-core architectures," *IEEE Computer Architecture Letters*, vol. 2, no. 1, pp. 2–2, 2003.

[18] T. S. Karkhanis et al., "Automated design of application specific superscalar processors: An analytical approach," in *Proceedings of the 34th Annual International Symposium on Computer Architecture*, ser. ISCA '07. New York, NY, USA: ACM, 2007, p. 402–411.

[19] S. Eyerman et al., "Mechanistic-empirical processor performance modeling for constructing cpi stacks on real hardware," in *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software*, ser. ISPASS '11. USA: IEEE Computer Society, 2011, p. 216–226.

[20] K. Van Craeynest et al., "Scheduling heterogeneous multi-cores through performance impact estimation (pie)," in *2012 39th Annual International Symposium on Computer Architecture (ISCA)*, 2012, pp. 213–224.

[21] M. Pricopi et al., "Power-performance modeling on asymmetric multi-cores," in *Proceedings of the 2013 International Conference on Compilers, Architectures and Synthesis for Embedded Systems*, ser. CASES '13. IEEE Press, 2013.

[22] N. K. Boran et al., "Fine-grained scheduling in heterogeneous-isa architectures," *IEEE Computer Architecture Letters*, 2020.

[23] N. Boran et al., "Performance modelling of heterogeneous isa multicore architectures," in *2016 IEEE East-West Design & Test Symposium (EWDTS)*, 2016, pp. 1–4.

[24] D. Terpstra et al., "Collecting performance data with papi-c," in *Tools for High Performance Computing 2009*, M. S. Müller, M. M. Resch, A. Schulz, and W. E. Nagel, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 157–173.

[25] K. Singh et al., "Real time power estimation and thread scheduling via performance counters," *SIGARCH Comput. Archit. News*, vol. 37, no. 2, p. 46–55, jul 2009.

[26] N. Binkert et al., "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, Aug. 2011.

[27] S. L. et al., "Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *2009 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Dec 2009, pp. 469–480.