Platform independent software analysis for near memory computing

Document status and date:
Published: 24/06/2019

Document Version:
Accepted manuscript including changes made at the peer-review stage

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

Platform Independent Software Analysis for Near Memory Computing

Stefano Corda[1,2], Gagandeep Singh[1,2], Ahsan Javed Awan[3], Roel Jordans[1], Henk Corporaal[1]

[1]Eindhoven University of Technology   [2]IBM Research - Zurich   [3]Ericsson Research

{s.corda, g.singh, r.jordans, h.corporaal}@tue.nl, ahsan.javed.awan@ericsson.com

Abstract—Near-memory Computing (NMC) promises improved performance for the applications that can exploit the features of emerging memory technologies such as 3D-stacked memory. However, it is not trivial to find such applications and specialized tools are needed to identify them. In this paper, we present PISA-NMC, which extends a state-of-the-art hardware agnostic profiling tool with metrics concerning memory and parallelism, which are relevant for NMC. The metrics include memory entropy, spatial locality, data-level, and basic-block-level parallelism. By profiling a set of representative applications and correlating the metrics with the application's performance on a simulated NMC system, we verify the importance of those metrics. Finally, we demonstrate which metrics are useful in identifying applications suitable for NMC architectures.

Index Terms—LLVM, NMC, Spatial Locality, Memory Entropy, Data-Level Parallelism

I. Introduction

Big-data workloads, due to highly random access nature, are not able to efficiently exploit the cache hierarchy. Consequently, they cause huge cache-flushes and data retrievals from off-chip memory [1], [2]. This off-chip movement leads to substantial energy consumption and stalling of compute resources, which causes performance degradation. Recent advancements in technology, however, have enabled us to bring the compute units in the proximity of data, which has lead to renewed interest in near-memory computing (NMC) architectures. Existing literature shows that these architecture are effective for a wide range of application ranging from graph processing to data management [3]. However, it's not trivial to identify those kernels which would benefit from the NMC paradigm and most of the studies rely on profiling applications of interest using e.g. hardware performance counters. The influence of micro-architecture features limits this technique.

To avoid this pitfall, we propose the platform-independent approach of characterizing workloads from NMC paradigm perspective. The idea is to profile instruction traces and collect inherent application information related to memory behavior and parallelism. For that, we extend the capabilities of PISA [4], a platform-independent software analysis tool to extract characteristics directed towards NMC architectures [5] and we propose a method to validate the relevance of proposed metrics in selecting the kernels to offload on NMC hardware.

II. PISA-NMC

PISA is based on the LLVM Compiler framework. PISA's architecture is shown in *Figure 1*. Initially, the application

source code, e.g., C/C++ code, is translated into the LLVM's intermediate representation (IR) using a clang front-end. This IR is independent of the target architecture and has a RISC-like instruction set. PISA exploits the *opt* tool to perform LLVM's IR optimizations and to perform the instrumentation process using an LLVM pass. This process is done by inserting calls to the external analysis library throughout the application's IR. The last step consists of a linking process that generates a native executable. On running this executable, we can obtain analysis results for specified metrics in JSON format. PISA can extract metrics such as instruction mix, branch entropy, data reuse distance, etc. Moreover, PISA supports the MPI and OpenMP standards allowing the analysis of multi-threaded and multi-process applications.
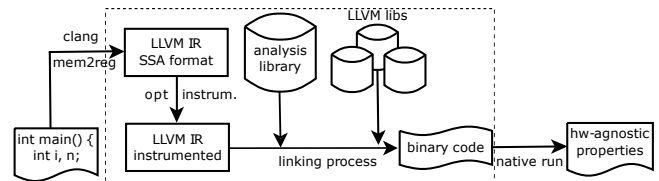
Fig. 1: Overview of the Platform-Independent Software Analysis Tool [6].

We extend PISA with metrics directed towards to NMC (PISA-NMC) [5]. We focus on the memory behaviour, which is essential to decide if an application should be accelerated with NMC architecture, and on the parallelism behaviour, which is crucial to decide if a specific parallel architecture should be integrated into an NMC system.

A. Memory metrics

We add memory entropy, which measures the randomness of the memory addresses accessed using an entropy formula adapted to the memory accesses. If the memory entropy is high, which means a higher cache miss ratio, the application may benefit from 3D-stacked memory bandwith because of the volume of data moved from the main memory to the caches.

Data reuse distance or data temporal reuse (DTR) is a helpful metric to detect cache inefficiencies. The DTR of an address is the number of unique addresses accessed since the last reference of the requested data. We integrate this metric into PISA for different cache line sizes in order to compute the spatial locality metric. Spatial locality measures the probability of accessing nearby memory locations. The key idea behind
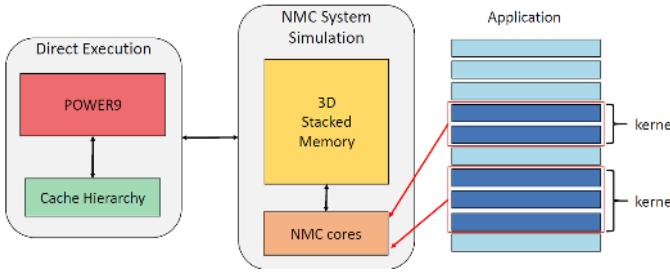
Fig. 2: Our NMC System

this spatial locality score is to detect a reduction in DTR when doubling the cache line size. Usually, application with low spatial locality perform very bad on traditional systems with cache hierarchies because a small portion of data is utilized compared to the data loaded from the main memory to the caches

B. Parallelism metrics

Data-level parallelism (DLP) measures the average length of vector instructions that are used to optimize a program. DLP could be interesting for NMC when employing specific SIMD processing units in the logic layer of the 3D-stacked memory. We specialize the instruction-level parallelism (ILP) per opcode in order to estimate the DLP.

A basic-block is the smallest component in the LLVM's IR that can be considered as a potential parallelizable task. Basic-block level parallelism (BBLP) is a potential metric for NMC because it can estimate the task level parallelism in the application. The parallel tasks can be offloaded to multiple compute units located on the logic layer of a 3D-stacked memory. We develop metrics similar to ILP considering the basic block as a set of instruction that can be run only sequentially.

We also aim to estimate the presence of data parallel loops. Data parallel loops consist of basic-blocks that are repeated without any dependencies among their instances. We develop a metric, PBBLP (potential basic-block-level parallelism) that tries in a fast and straightforward manner to estimate the basic-block level parallelism in data-parallel loops.

III. METHODOLOGY

To validate the relevance of our NMC specific metrics, we characterize the representative benchmarks using PISA-NMC and extract the numbers for memory entropy, spatial locality, data level parallelism, and basic-block level parallelism. We apply principal component analysis (PCA) [7], on the collected metrics to make it more understandable by reducing its dimensionality [8]. Next, we run the same benchmarks on an NMC system using a simulator and measure the improvement in energy-delay product and correlate this data with the output of PCA.

A. Host and NMC system

Figure 2 depicts the reference computing platform that we consider in this work. We run the applications both on a

traditional Von-Neumann Architecture using the latest *IBM Power 9* [9] and on an NMC system based on hybrid memory cube (HMC). HMC memory is divided into several vertical DRAM partitions, called vaults, each with its own DRAM controller in the logic layer. In this work, we model NMC PEs as in-order, single-issue cores with a private cache as proposed in previous work [10], [11]. *Table 1* lists the details of the host and NMC system used in our experiments. We extract the power consumption with AMESTER[1] tool on Power 9. We simulate the NMC system on an extended version of the memory simulator *Ramulator* [12] including the processing units. Each processing unit is assigned to a vault and operates on the data assigned to that vault. We collect dynamic execution traces of the instrumented code with a Pin tool. We feed the acquired traces to Ramulator.

Table 1: Host and NMC System Characteristics

Architecture	CPU Used	Cache per core	Memory
IBM Power9 (Host)	4 cores (SMT4) @ 2.3 GHz	L1 32 KB L2 256 KB L3 10 MB	DDR4, 32 GB RDIMM @ 2.7 GHz
NMC	32 single-issue in-order cores @ 1.25 GHz	L1-I/D 2-way 2 cache lines 64B per cache line	HMC, 4GB 8 stacked-layers, 32 vaults, 16-bit full duplex and SerDes I/O link @ 15 Gbps

B. Benchmarks

Existing literature is devoid of proper benchmarks to evaluate NMC systems and explore the design space. Most of the studies instead design NMC systems tailored to improve the performance of specific workloads [13]. We select a set of applications from two benchmark suites that are representative of the most common kernels and have been used previously in other related studies: Rodinia [14]–[16] and Polybench [16]. Rodinia [17] is a benchmark suite for heterogeneous computing. Rodinia workloads cover a wide range of different behaviors which help a developer in building new systems. Polybench [18] is a collection of a large number of common kernels like matrix multiplication, stencil, covariance, correlation, etc. Each kernel is in a single file, tunable at compile-time. This makes instrumentation easier.
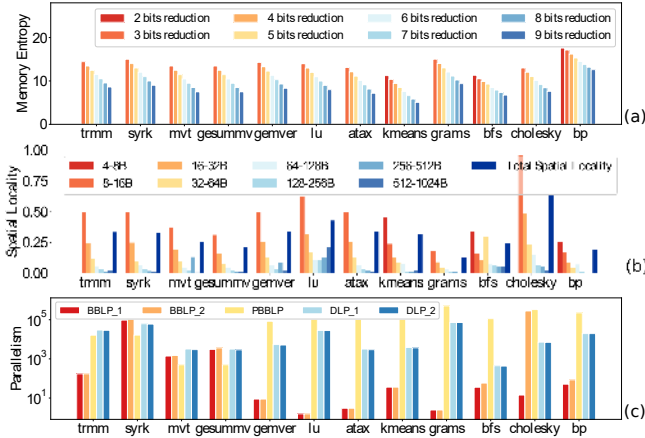
IV. RESULTS

We present the NMC specific characterization of selected applications from PolyBench and Rodinia benchmark. Then, we show the correlation between the added metrics to NMC performance.

A. Application characterization

In [5] we characterize a set of applications from Polybench and Rodinia for the added metrics (see *Figure 3*). Memory entropy, in *Figure 3.a*, and spatial locality, in *Figure 3.b*, show respectively high and low values for applications such as `bp` and `gramschmidt` that could benefit from NMC

---

[1] https://github.com/open-power/amester

because of their poor memory performance. *Figure 3.c* show the parallelism characterization of these kernels. In particular a group of applications shows good level of DLP, lowest level of BBLP and highest level of PBBLP. They could benefit from NMC architecture that exploit data-level parallel processing element.

Fig. 3: Application characterization results:(a) Memory Entropy; (b) Spatial Locality; (c) Parallelism.

### B. Applications evaluation on NMC system

We compare our metrics to the performance achieved by running the applications on an NMC system. We perform a single-thread analysis to estimate our metrics and then evaluate the execution time on the considered architectures. Table 2 lists the parameter levels for the evaluated applications on the Power 9 and NMC system. We consider only single-thread analysis here to avoid the side effects of a multi-threaded analysis in metrics such as memory entropy and spatial locality, e.g., averaging the numbers from multiple threads tend to mask the true behavior of applications. Since the analysis trend is similar for different dataset sizes and the memory analysis is highly time-consuming we use smaller dataset than the one simulated for the NMC system in line with similar work on application characterization [4], [19]. *Figure 4* shows the energy-delay product (EDP) ratio between the IBM Power 9 and the NMC system we simulated. We use EDP as our major metric of reference in this analysis because both energy and performance are critical criteria for evaluating NMC suitability. Applications with EDP reduction less than 1 are not suitable for NMC.

Table 2: Benchmarks Parameters

| Applications		Parameters	
Benchmarks	**Kernels**	**Param.**	**Values**
Polybench	atax, gemver, gesummv	dimensions	8000
	cholesky, gramschmidt, lu, mvt, syrk, trmm	dimensions	2000
Rodinia	bfs	nodes	1.0m
	bp	layer size	1.1m
	kmeans	data size	819k

C. Correlation between NMC Metrics and NMC Performance

Spatial locality in *Figure 3.b* provides insights on which application could be better for the NMC system we considered. Applications that show the lowest spatial locality such as `gramschmidt`, `bp`, `bfs` show a considerable EDP improvement (see *Figure 4*) using the NMC system. Contrariwise also `cholesky`, that has the highest spatial locality among the chosen applications, benefits from the NMC architecture. Memory entropy in *Figure 3.a* gives similar insights. For instance, applications with the highest entropy such as `gramschmidth` and `bp` shows benefit executing on an NMC system. However, also applications with low entropy seem to benefit from NMC. Parallelism analysis in *Figure 3.c* highlights that most of the applications that benefit from NMC have the lowest values for $BBLP_1$ and a good level of DLP. However, there are some exceptions such as `lu`, that has the lowest BBLP values, and `bfs` that has the lowest DLP values. The above shows that a single metric can not explain NMC appropriateness. To get more insights into what combinations of metrics can predict NMC applicability, we apply PCA to our metric results.
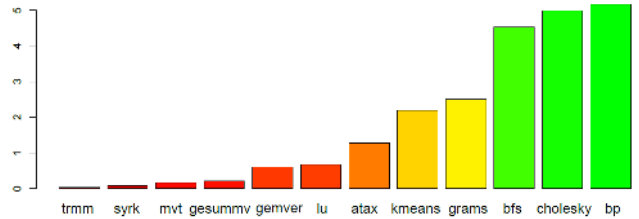

Fig. 4: EDP improvement

For this, we derive another metric from the memory entropy exploiting the granularity. For each application, we first compute the difference between each couple of consecutive memory entropy values with different granularities (see *Figure 3.a*, larger granularity represents larger cache line size). Then, we compute the average of these values that represents a spatial locality variation increasing the cache line size. *Figure 5* shows this metric. This metric compared to the EDP values shows that the major part of the applications not suitable for NMC has the highest values.
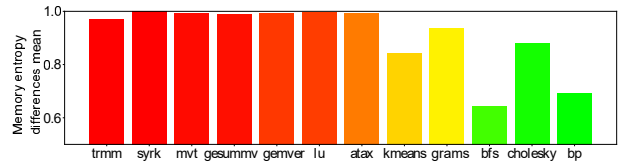
Fig. 5: Metric derived from memory entropy.

Figure 6 shows the PCA applied to the most promising subset of presented metrics. We use 4 input features for the PCA: $BBLP_1$, $PBBLP$, $entropy\_diff\_mem$ (the value proposed above) and $spat\_8B\_16B$ (spatial locality doubling the cache line size from 8B to 16B). We highlight that all the

applications that benefit from Power9 are in the II quadrant (top-left) except for `lu` that is in the III quadrant. In its code diagonal matrix accesses are present and they should be critical for traditional CPUs. It could be an NMC application candidate employing a larger dataset size. The applications that benefits from NMC are in the other quadrants. In particular `bfs` and `bp` seem having similar characteristic and are located in the I quadrant. Similarly `gramschmidt` and `kmeans` located in the IV quadrant. These metrics show good potential in discriminating NMC applications.
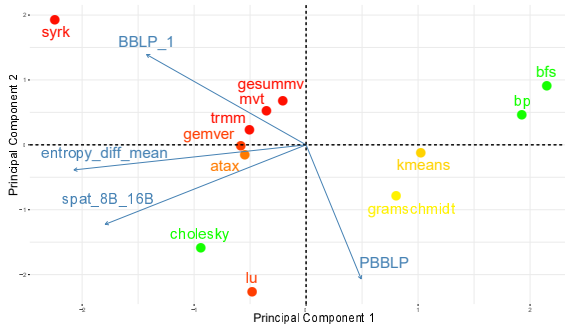
Fig. 6: PCA using the added metrics. Blue arrows quantify the contribution and direction to the PCs.

V. Related Work

Existing studies primarily rely on hardware performance counters available in the modern processors to understand the memory access behavior of the applications and identify the kernels suitable for offload to NMC architectures [14], [20]–[23]. Others have used a dynamic binary instrumentation framework like Pin [24] or estimation at the compile time [15], [16] for the same purpose. PISA-NMC showed a different approach to workload characterization applied to NMC. We used a target-agnostic workload characterization technique to extract metrics directed towards NMC. Then, we used PCA and NMC simulation to show the relevance of the metrics proposed.

VI. Conclusion

We extend PISA with NMC specific metrics such as data-level parallelism, basic-block level parallelism, memory entropy, and spatial locality. By correlating the principal components of metrics as mentioned above with the energy-delay product of benchmark kernels on an NMC system, we show that PISA-NMC can help to identify the kernels that can benefit from NMC in a platform agnostic manner. As future work, we will investigate more workloads and perform a more exhaustive analysis.

Acknowledgments

References

[1] A. J. Awan *et al.*, "Performance characterization of in-memory data analytics on a modern cloud server," in *2015 IEEE Fifth International Conference on Big Data and Cloud Computing*. IEEE, 2015, pp. 1–8.

[2] ——, "Micro-architectural characterization of apache spark on batch and stream processing workloads," in *2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud)*. IEEE, 2016, pp. 59–66.

[3] G. Singh *et al.*, "A review of near-memory computing architectures: Opportunities and challenges," in *2018 21st Euromicro Conference on Digital System Design (DSD)*, Aug 2018, pp. 608–617.

[4] A. Anghel *et al.*, "An instrumentation approach for hardware-agnostic software characterization," *International Journal of Parallel Programming*, vol. 44, no. 5, pp. 924–948, Oct 2016. [Online]. Available: https://doi.org/10.1007/s10766-016-0410-0

[5] S. Corda *et al.*, "Memory and parallelism analysis using a platform-independent approach," in *ACM 22nd International Workshop on Software and Compilers for Embedded Systems (SCOPES '19)*. Sankt Goar, Germany: ACM, May 2019.

[6] A. Anghel *et al.*, "An instrumentation approach for hardware-agnostic software characterization," *International Journal of Parallel Programming*, vol. 44, pp. 924–948, 2015.

[7] I. Jolliffe, *Principal Component Analysis*. Springer Verlag, 1986.

[8] K. Hoste *et al.*, "Comparing benchmarks using key microarchitecture-independent characteristics," *2006 IEEE International Symposium on Workload Characterization*, pp. 83–92, 2006.

[9] IBM. Ibm power 9. [Online]. Available: https://www.ibm.com/it-infrastructure/power/power9

[10] J. Ahn *et al.*, "A scalable processing-in-memory accelerator for parallel graph processing," in *ISCA 2015*.

[11] M. Gao *et al.*, "Practical near-data processing for in-memory analytics frameworks," in *PACT 2015*.

[12] Y. Kim *et al.*, "Ramulator: A fast and extensible dram simulator," *IEEE Computer Architecture Letters*, vol. 15, no. 1, pp. 45–49, Jan 2016.

[13] G. Singh *et al.*, "A review of near-memory computing architectures: Opportunities and challenges," 08 2018.

[14] D. Zhang *et al.*, "TOP-PIM: throughput-oriented programmable processing in memory," in *Proceedings of the 23rd international symposium on High-performance parallel and distributed computing*. ACM, 2014, pp. 85–98.

[15] K. Hsieh *et al.*, "Transparent offloading and mapping (TOM): Enabling programmer-transparent near-data processing in GPU systems," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, June 2016, pp. 204–216.

[16] A. Pattnaik *et al.*, "Scheduling techniques for GPU architectures with processing-in-memory capabilities," in *2016 International Conference on Parallel Architecture and Compilation Techniques (PACT)*, Sept 2016, pp. 31–44.

[17] S. Che *et al.*, "Rodinia: A benchmark suite for heterogeneous computing," in *2009 IEEE International Symposium on Workload Characterization (IISWC)*, Oct 2009, pp. 44–54.

[18] L.-N. Pouchet, "Polybench: The polyhedral benchmark suite," *URL: http://www. cs. ucla. edu/pouchet/software/polybench*, 2012.

[19] X. Gu *et al.*, "A component model of spatial locality," in *Proceedings of the 2009 International Symposium on Memory Management*, ser. ISMM '09. New York, NY, USA: ACM, 2009, pp. 99–108.

[20] A. J. Awan *et al.*, "Identifying the potential of near data processing for apache spark," in *Proceedings of the International Symposium on Memory Systems*. ACM, 2017, pp. 60–67.

[21] J. Ahn *et al.*, "PIM-enabled instructions: a low-overhead, locality-aware processing-in-memory architecture," in *Proceedings of the 42nd Annual International Symposium on Computer Architecture*. ACM, 2015, pp. 336–348.

[22] A. J. Awan, "Performance characterization and optimization of in-memory data analytics on a scale-up server," Ph.D. dissertation, KTH Royal Institute of Technology and Universitat Politècnica de Catalunya, 2017.

[23] A. Boroumand *et al.*, "Google workloads for consumer devices: Mitigating data movement bottlenecks," *SIGPLAN Not.*, vol. 53, no. 2, pp. 316–331, Mar. 2018.

[24] J. Ahn *et al.*, "A scalable processing-in-memory accelerator for parallel graph processing," in *2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA)*, June 2015, pp. 105–117.