Platform Independent Software Analysis for Near Memory Computing

Stefano Corda\textsuperscript{1,2}, Gagandeep Singh\textsuperscript{1,2}, Ahsan Javed Awan\textsuperscript{3}, Roel Jordans\textsuperscript{1}, Henk Corporaal\textsuperscript{1}
\textsuperscript{1}Eindhoven University of Technology \quad \textsuperscript{2}IBM Research - Zurich \quad \textsuperscript{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 \cite{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 \cite{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 \cite{4}, a platform-independent software analysis tool to extract characteristics directed towards NMC architectures \cite{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 \ref{fig:overview}. 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 \cite{6}](image-url)

We extend PISA with metrics directed towards to NMC (PISA-NMC) \cite{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 bandwidth 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
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 (BBPLP) 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, PBPLP (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)\(^1\), on the collected
metrics to make it more understandable by reducing its
dimensionality\(^2\). 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

![Fig. 2: Our NMC System](image)

traditional von-Neumann Architecture using the latest *IBM
Power 9*\(^3\) 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\(^4\).\(^5\). Table 1 lists the details of
the host and NMC system used in our experiments. We
extract the power consumption with AMESTER\(^6\) tool on
Power 9. We simulate the NMC system on an extended
version of the memory simulator Ramulator\(^7\) 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

<table>
<thead>
<tr>
<th>Architecture</th>
<th>CPU Used</th>
<th>Cache per core</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM</td>
<td>4 cores</td>
<td>L1 32 KB</td>
<td>DDR4, 32 GB</td>
</tr>
<tr>
<td>Power9 (Host)</td>
<td>(SMT4) @ 2.3 GHz</td>
<td>L2 256 KB, L3 10 MB</td>
<td>RDIMM @ 2.7 GHz</td>
</tr>
<tr>
<td>NMC</td>
<td>32 single-issue cores @ 1.25 GHz</td>
<td>L1/1D 2-way cache lines 64B per cache line</td>
<td>HMC, 4GB, 8 stacked-layers, 32 vaults, 16-bit full duplex and SerDes I/O link @ 15 Gbps</td>
</tr>
</tbody>
</table>

### B. Benchmarks

Existing literature is devoid of proper benchmarks to evalu-
ate NMC systems and explore the design space. Most of the
studies instead design NMC systems tailored to improve
the performance of specific workloads\(^8\).\(^9\). 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\(^10\)–\(^16\) 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, correla-
tion, 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\(^1\) we characterize a set of applications from Polybench
and Rodinia for the added metrics (see Figure 2). Memory
entropy, in Figure 2a, and spatial locality, in Figure 2b,
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 3c 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.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig3}
\caption{Application characterization results:(a) Memory Entropy; (b) Spatial Locality; (c) Parallelism.}
\end{figure}

### 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 7 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.

\begin{table}[h]
\centering
\caption{Benchmarks Parameters}
\begin{tabular}{|c|c|c|}
\hline
Applications & Parameters & Values \\
\hline
Polybench & atax, gemver, gesummv & dimensions 8000 \\
& cholesky, gramschmidt, lu, mvt, syrk, trmm & dimensions 2000 \\
Rodinia & bfe & nodes 1.0m \\
& bp & layer size 1.1m \\
& kmeans & data size 819k \\
\hline
\end{tabular}
\end{table}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig4}
\caption{EDP improvement}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig5}
\caption{Metric derived from memory entropy}
\end{figure}

C. Correlation between NMC Metrics and NMC Performance

Spatial locality in Figure 3b provides insights on which application could be better for the NMC system we considered. Applications that show the lowest spatial locality such as gramschmidt, bfe, 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 3a gives similar insights. For instance, applications with the highest entropy such as gramschmidt and bp shows benefit executing on an NMC system. However, also applications with low entropy seem to benefit from NMC. Parallelism analysis in Figure 3c 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 BBL 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.
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], [15]–[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

This work is funded by European Commission under Marie Skłodowska-Curie Innovative Training Networks European Industrial Doctorate (Project ID: 676240).

REFERENCES


