The presented work was submitted to the
Chair for High-Performance Computing (Computer Science 12), IT Center

Performance Metrics for Access Pattern-aware Analysis of Heterogeneous Memory Power Consumption in HPC

Metriken zur Analyse des Energieverbrauchs von heterogenem Arbeitsspeicher im Hochleistungsrechnen unter Berücksichtigung der Zugriffsmuster

Master Thesis

Lukas Alt

Aachen, October 19, 2023

First Examiner:	Prof. Dr. rer. nat. Matthias S. Müller (‘)
Second Examiner:	Dr. rer. nat. Stefan Lankes (*)
Supervisor:	Anara Kozhokanova, M.Sc. (‘)

(‘) Chair for High-Performance Computing, RWTH Aachen University
IT Center, RWTH Aachen University
(*) Institute for Automation of Complex Power Systems
E.ON Energy Research Center, RWTH Aachen University

Communicated by Prof. Matthias S. Müller

I hereby declare that I wrote this thesis myself without sources other than those indicated herein. All parts taken from published and unpublished scripts are indicated as such. The paper has not been previously presented as an examination paper in any other form.

Aachen, October 19, 2023

Abstract

In response to the rising demand for memory performance and capacity, memory heterogeneity in HPC systems increased. In particular, technologies such as high-bandwidth memory (HBM) and high-capacity memory (HCM) are employed in addition to DRAM. While the power consumption of the memory subsystem was often neglected in node-level power optimizations in the past, the increased power consumption by HCM motivates studying the energy consumption of heterogeneous memory on the latest architectures under workloads with different memory access patterns.

A method for measuring the memory energy consumption using hardware instrumentation of memory slots is described and implemented. Measurements using this approach are compared to RAPL, a software interface for limiting and controlling power consumption on Intel systems. Results show that RAPL energy measurements for the memory domain can differ significantly - up to 120% - from reference measurements on Intel Ice Lake-SP systems. A discussion of possible reasons yields that the RAPL memory domain may include losses at the voltage regulator level. The accuracy of the reference measurements was validated by comparing the results from the literature to results obtained from a similar architecture (Broadwell-EP).

This thesis presents the new metrics DEL and DES for heterogeneous memory energy evaluation using different memory access patterns. Additionally, the BpW metric was utilized for memory energy efficiency characterization. The metrics are based on the instrumented energy measurements conducted on the Ice Lake architecture equipped with DRAM and Intel Optane Persistent Memory (PMem). The results demonstrate that the memory access pattern and the concurrency in memory accesses significantly impact the memory's dynamic energy consumption. Furthermore, it shows that PMem is more energy efficient per capacity than DRAM at idle and is better suited for storing rarely accessed data. When PMem is under load, DRAM is more energy efficient.

The proposed metrics are then used to estimate the energy consumption of real-world applications, followed by a discussion on the applicability of this approach and potential improvements.

Keywords: HPC, Heterogeneous Memory, RAPL validation, Energy Efficiency, Performance Metrics, Intel Optane Persistent Memory, Dynamic Energy per Load, Dynamic Energy per Store, Bandwidth per Watt

Kurzfassung

Als Reaktion auf die steigende Nachfrage nach Speicherkapazität und -performance hat die Heterogenität des Arbeitsspeichers im Hochleistungsrechnen zugenommen. Insbesondere werden neben DRAM auch Technologien wie High-Capacity Memory (HCM) und High-Bandwidth Memory (HBM) eingesetzt. Während der Energieverbrauch von DRAM in der Vergangenheit oft bei der Energieoptimierung auf Knotenebene vernachlässigt wurde, motiviert der erhöhte Energieverbrauch von HCM die Untersuchung des Energieverbrauchs von heterogenem Speicher auf aktuellen Architekturen mit unterschiedlichen Zugriffsmustern.

Es wird eine Methode zur Messung des Energieverbrauchs des Speichers durch Hardware-Instrumentierung der Speichersteckplätze beschrieben und implementiert. Die so durchgeführten Messungen werden mit RAPL verglichen, einem Interface zur Begrenzung und Messung des Energieverbrauchs auf Intel Systemen. Die Ergebnisse zeigen, dass die RAPL Messungen für den Arbeitsspeicher erheblich - um bis zu 120% - von Referenzmessungen auf Intel Ice Lake-SP Systemen abweichen. Eine Diskussion möglicher Gründe ergab, dass die RAPL-Speicherdomäne möglicherweise Verluste auf der Spannungsreglerebene enthält. Die Genauigkeit der Referenzmessungen wurde durch den Vergleich von Ergebnissen aus der Literatur mit den Ergebnissen auf einer ähnlichen Architektur (Broadwell-EP) im Rahmen dieser Arbeit validiert.

In dieser Arbeit werden die neuen Metriken DEL und DES für die Bewertung des Energieverbrauchs von heterogenem Arbeitsspeicher unter Berücksichtigung der Zugriffsmuster vorgestellt. Zusätzlich wurde die BpW-Metrik für die Charakterisierung der Speicherenergieeffizienz verwendet. Die Metriken basieren auf Energiemessungen, die auf der Ice Lake-Architektur mit DRAM und Intel Optane Persistent Memory (PMem) durchgeführt wurden. Die Ergebnisse zeigen, dass das Zugriffsmuster und die Anzahl der gleichzeitigen Speicherzugriffe den dynamischen Energieverbrauch erheblich beeinflussen. Außerdem zeigt sich, dass PMem im Idle-Betrieb pro Kapazität energieeffizienter als DRAM ist und sich somit besser für die Speicherung selten benötigter Daten eignet. Unter Last ist DRAM aber energieeffizienter als PMem.

Die präsentierten Metriken werden dann zur Schätzung des Energieverbrauchs realer Anwendungen verwendet, gefolgt von einer Diskussion über die Anwendbarkeit dieses Ansatzes und Verbesserungsmöglichkeiten.

Stichwörter: HPC, Heterogener Arbeitsspeicher, RAPL Validation, Energieeffizienz, Performancemetriken, Intel Optane Persistent Memory, Dynamic Energy per Load, Dynamic Energy per Store, Bandwidth per Watt

Contents

List of Figures

List of Tables

List of Listings

1. Introduction

The continuously increasing complexity and data intensity of HPC applications raise the demand for performance-optimized, energy-efficient, cost-effective, and high-capacity memory. In response to this, especially with exascale computing in mind, the memory heterogeneity in HPC is increasing. An emerging class of heterogeneous memory is high-capacity memory (HCM), which provides higher memory capacities than traditional Dynamic Random Access Memory (DRAM), often with persistence features.

One of the first commercially available types of HCM is Intel Optane Persistent Memory (PMem), a non-volatile, byte-addressable Dual Inline Memory Module (DIMM) with a performance similar to regular DRAM. PMem still requires DRAM to be installed on the system, resulting in two memory types with different features and characteristics to be present on a single system.

In the past, node-level power optimizations mainly focused on optimizing CPU and GPU power, and the memory system was often neglected due to its minor contribution to total power consumption. However, in modern, heterogeneous memory systems, particularly with HCM, the memory domain can have a power consumption similar to the CPU. For instance, a dual-socket node with 512 GB of DRAM and 2 TB of PMem has a peak CPU power consumption of 410 W, while the memory can consume up to 432 W[1].

With energy consumption being a limiting factor in HPC, especially in terms of costs, carbon footprint, and power availability, the increased energy demand for the memory subsystem cannot be neglected anymore and needs to be considered in node-level energy efficiency optimization. The energy consumption of memory, especially the impact of the memory access pattern, has yet to be investigated thoroughly.

The objective of this thesis is to gain a deeper understanding of memory power consumption, particularly how the memory access pattern impacts the power consumption and how it differs between DRAM and PMem. So far, no performance metrics have been well-established for heterogeneous memory energy consumption analysis. This thesis proposes performance metrics that enable a quantitative comparison between the memory energy efficiency of different memory types and access patterns and uses them for evaluating DDR4 DRAM and 2nd-generation PMem.

This requires measuring the energy consumption of the memory accurately. A prominent technique for this is Running Average Power Limit (RAPL), an interface for limiting and measuring power consumption on Intel systems. The accuracy of

[1]The 16 PMem modules have a TDP of 15W each. For the DRAM, no TDP values are provided by the manufacturers. This computation assumes a modest TDP of 3 W for 8 GB of DDR4 [6].

RAPL's memory domain has been assessed as overall accurate on the Haswell [10] and Skylake [23] platforms. Many studies in the past relied on these results and used RAPL for memory measurements and energy consumption modeling. As the CPU and system architecture has undergone several changes since the Skylake architecture, in particular, the added support for PMem and associated changes in power delivery, evaluating the accuracy of DRAM and PMem energy measurements using RAPL on recent platforms is highly motivated before using them.
This thesis makes the following main contributions:

- Design and implementation of an experimental setup to accurately measure the power consumption of DIMMs and synchronize the measurements with the execution of an application.
- Validation of RAPL memory energy measurements against reference measurements on an Intel IceLake-SP system with PMem and an Intel Broadwell-EP system as a reference point for older architectures.
- Design and computation of the DEL, DES, and BpW metrics using energy measurements of DRAM and PMem DIMMs.
- Energy efficiency evaluation and comparison between DRAM and PMem with varying memory access patterns.
- Design of an energy estimation model for DRAM and PMem using access pattern-aware metrics and evaluating its accuracy with real-world applications.

The remaining part of the thesis is structured a follows: The extensive background information is provided in chapter 2, followed by a discussion on related work in chapter 3. Chapter 4 covers an analysis of RAPL concerning temporal resolution and accuracy. In chapter 5, the new metrics for heterogeneous memory energy efficiency analysis are proposed. It also includes the results from the memory energy consumption measurements under different workloads used for a per-instruction energy characterization. Chapter 6 describes how these metrics can be used for estimating the memory energy consumption of real-world applications, along with an evaluation using two case studies. The results are concluded with a summary and an outlook to future work in chapter 7.

2. Background

This chapter starts with a description of modern processors and system architectures. Next, we review different memory types that are used in current HPC systems with respective performance and power characteristics. Then, common memory access pattern classifications are discussed. In this thesis, we focus on Running Average Power Limit (RAPL) and explain the foundations for an external power measurement technique to validate RAPL's accuracy on the latest Intel server system. Finally, a discussion on power distribution on such systems concludes the chapter.

2.1. Modern Processors' Memory Architectures

This section describes basic concepts in HPC architectures and the memory technologies DRAM and Intel Optane PMem that are referenced in this thesis.

2.1.1. Cache Hierarchy

In modern applications, different types of memory access patterns can be observed. To reduce the memory access latency, modern systems employ a layered hierarchy of small but fast caches. Regarding cache efficiency, two general locality paradigms for memory accesses can be defined [21]:
Spatial locality: A memory location that is close to the current memory location will be accessed soon.
Temporal locality: The current memory location will be reaccessed in the near future.

2.1.2. Memory Hierarchy

A shared-memory parallel computer is a computing system with multiple CPUs or CPU cores and a common physical memory address space. These systems can be classified into two memory access architectures based on how memory accesses are organized.

The `Uniform Memory Access (UMA)` model exhibits a flat memory model in which each processor can access the memory with approximately the same latency and bandwidth. In most systems, this is realized by connecting each processor to a global memory through a shared memory bus. However, this results in scaling issues with increasing numbers of processor cores. Thus, modern HPC systems employ a different memory access paradigm, especially those with multiple sockets [21].

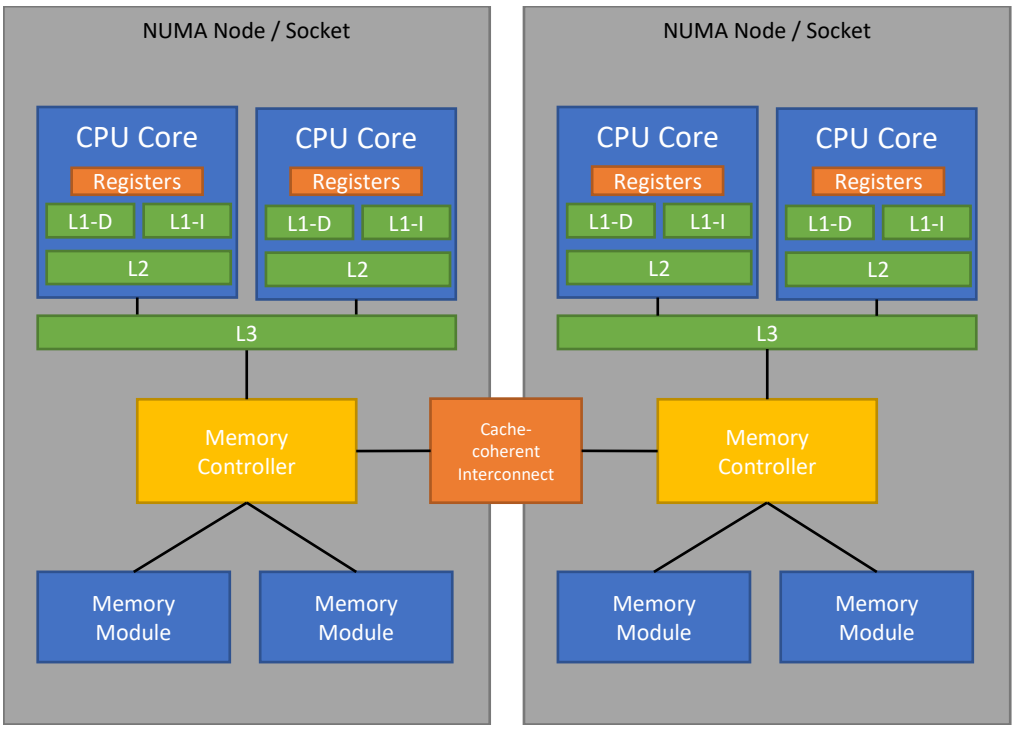


Figure 2.1.: Schematic visualization of a cache-coherent NUMA system [14, 21]

In a `Non-uniform Memory Access (NUMA)` system, the complete memory is still mapped to a single global address space. However, the processor cores are separated into different locality domains with a local memory attached to each domain. The cores of such a locality domain can still access the memory of another locality domain through an interconnect but with limited bandwidth and higher latency compared to local memory access [21].

In most multi-socket systems, each socket forms a locality domain with its local memory as visualized schematically in figure 2.1. Thus, the access to memory from the own socket is significantly faster compared to remote access, requiring awareness from developers to ensure that they get adequate memory performance. If multiple types of heterogeneous memory, for example, high-bandwidth memory or high-capacity memory (see section 2.1.4) are installed in the system, these might be exposed as separate NUMA nodes as well. The exact NUMA topology can be inspected using the `numactl -H` command on unix-based systems. Detailed information on controlling memory allocations on NUMA systems can be found in appendix A.1.

2.1.3. Dynamic Random Access Memory (DRAM) and Dual Inline Memory Modules (DIMM)

Dynamic Random Access Memory (DRAM) is the technology of choice for the main memory in desktop computers and most HPC systems currently. DRAM is available

in many generations, variants, and form factors, which can be confused easily. To remove complexity, we will first clarify the terminology based on *DDR4 SDRAM - RDIMM* as an example, which contains three different concepts: *DDR4* is the generation of the memory, *SDRAM* refers to the actual memory technology, and *DIMM* is the form factor of the memory.

DRAM is an integrated circuit that stores the actual data in volatile memory. Each DRAM chip consists of multiple banks that can operate concurrently. These banks are further divided into rows and columns for per-byte access granularity. Multiple DRAM chips can also be grouped into ranks [4, 16]. DRAM requires periodic refresh of the capacitors that store the data, resulting in a non-negligible static power consumption of the memory and making DRAM volatile - if power to the DRAM is interrupted, the stored data will be lost.

DRAM Generations Different generations of DRAM have been established in the last two decades and are listed in table 2.1. Significant performance improvements have been achieved with each generation, mainly by increasing the number of bytes sent at once (burst size) and usually doubling the transfer rate with each generation. At the same time, the supply voltage has been constantly reduced to achieve lower power consumption. A significant difference between Double Data Rate (DDR) and Single Data Rate (SDR) is that data is sent both on the rising and the falling edge of the clock signal, effectively doubling the transfer rate at the same clock speed compared to the SDR. The transfer rate is an essential criterion for memory performance, even within different generations of DRAM, but is often misreported as the clock frequency. However, the clock frequency is half of the transfer rate.

	Release Date	**Transfer Rate**	**Supply Voltage**	**Burst Size**
SDR	1993	66–133 MT/s	3.3 V	1
DDR	1998	200–400 MT/s	2.5 V	2
DDR3	2003	400–1066 MT/s	1.8 V	4
DDR3	2007	800–2666 MT/s	1.5 V	8
DDR4	2014	1500–5333 MT/s	1.2 V	8
DDR5	2020	4800–8000 MT/s	1.1 V	16

Table 2.1.: DRAM Generations [4, 7, 18, 80]

Although DDR5 memory became commercially available in 2020, it is still considered as a new technology that is subject to improvements in the upcoming years. As of 2023, most HPC installations are still using DDR4 [96] memory, but new installations will likely use DDR5 memory in the future. To cover special-purpose use-cases, different variants of DRAM have been developed, with the most important ones being *GDDR* and *LPDDR*. *GGDR*, for example, *GDDR5X* is a highly optimized and high bandwidth version mainly used as the main memory of GPUs. In contrast, *Low Power DRAM (LPDDR)* is a variant targeting mobile devices such

as smartphones or laptops and is optimized for lower power consumption and smaller module dimensions.

In this thesis, we will focus on DDR4-DRAM as it is the most used main memory type in HPC currently, in fact, most architectures listed in Top500 do not support DDR5 memory as of June 2023 [96] and the next 4th generation Intel Xeon code-named Sapphire Rapids supporting DDR5 it is not widely available to the public yet.

Dual Inline Memory Module (DIMM) DIMM is a form factor commonly used for modern DRAM memory modules both in HPC and desktop computers. The modules can be plugged into the corresponding DIMM slots on the mainboard of the computer to ensure a close connection to the CPU while maintaining replaceability.

DIMMs are also available in different variants. In contrast to unbuffered DIMMs (UDIMMs), which are mainly used in consumer-grade systems, registered DIMMs (RDIMMs) feature an additional register between the memory register and the memory modules to improve the reliability of the memory, reduce stress on the memory controller, and to allow higher module capacities. Load-reduced DIMMs (LRDIMMs) can support additional memory capacity by reducing the load on the memory even further compared to RDIMMs.

For all of these DIMM types, power supply and data communication between the computer and the memory is provided through multiple pins, e.g., 288 for DDR4, in the DIMM connector. The exact role of each of these pins is standardized by the JEDEC organization [36].

Power Supply of DDR4 RDIMMs

Most of the 288 pins are used as digital data rails. In addition, the following power-delivery related voltages are provided to the DIMM [36]:

- `VDD` is the main power supply for the SDRAM. It is provided through several, redundant pins and operates at 1.2 V. The maximum expected power drawn from this supply is about 14 W.
- `VPP` is the activation power supply/word-line boost for the SDRAM and operates at 2.5 V. The maximum expected power drawn from this pin is about 9 W.
- `12V` is a pin for optionally providing 12 V to the DIMM. It is unused on RDIMMs.
- `VDDSPD` is the power supply for the serial presence detect (SPD)[1] of the memory. It does not relate to the other voltages and is not considered relevant after the system has started.

[1]SPD stores essential information such as speed, capacity, manufacturer of the memory and provides it to the system for memory detection, configuration, and training purposes [36].

- `VTT` is the power supply of the I/O termination of the SDRAM.
- `VREFCA` is the reference voltage for digital signals (usually $\frac{1}{2}$ VDD).
- `VSS` is the ground.

Some of these pins have no or just a minor impact on the total power consumption of the DIMM. Thus, it is sufficient to measure the power consumption of `VDD` and `VPP` in order to measure the power consumption of a complete DDR4 DIMM [10].

2.1.4. Intel® Optane™ Persistent Memory (PMem)

Intel Optane Persistent Memory (a.k.a PMem, DCPMM, "Optanes") is a type of non-volatile, byte-addressable, high-capacity memory based on Intel's 3D XPoint technology. In contrast to the 3D XPoint SSDs, the Optane modules are available in a DIMM form factor with the same interface as DDR4 RDIMMs. However, they cannot be used interchangeably with DRAM and require specific support by the mainboard and the CPU, which was introduced with 2nd and 3rd Generation Intel Xeon Scalable Processors (codenamed Cascade Lake and Ice Lake respectively). Optane PMem modules are currently available in three different generations, namely 100, 200, and 300 series. In this thesis, we focus on the 200 series compatible with Ice Lake CPUs as 300 series supported by the Sapphire Rapids CPU are out of our reach currently.

Intel Optane PMem 200 series modules are available with 128GB, 256GB, and 512GB capacity and feature a transfer speed of 3200MT/s on dual-socket systems[2] at a Thermal Design Power (TDP) of 15 W per module [30].

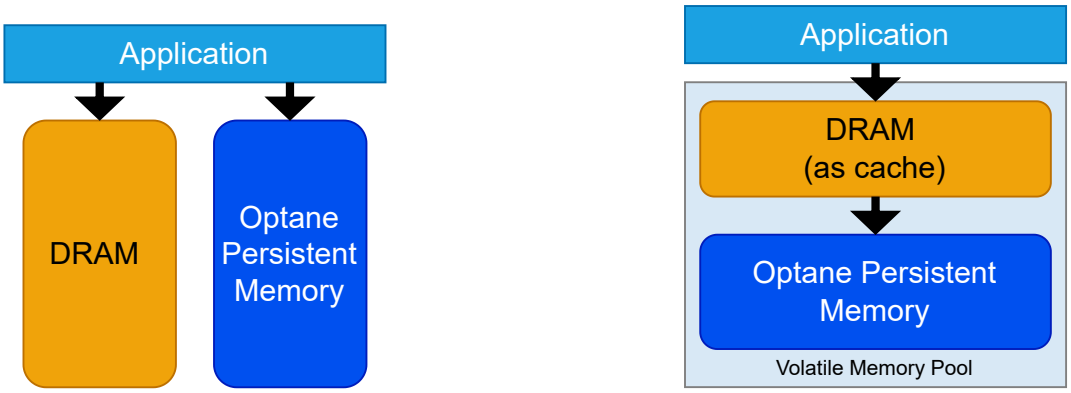
Operation Modes

In order to operate PMem, DRAM modules still have to be mounted on the system [31]. Depending on the use case, PMem can be configured in two operation modes: *Memory Mode* and *AppDirect* mode [30]:

In *Memory* mode, the capacity of only PMem (far memory) is available as a directly addressable memory. As it features a lower bandwidth and higher latency than DRAM, the DRAM (near memory) is used as a cache to the far memory. Consequently, a memory access that can be satisfied by the DRAM cache has the same performance characteristics in terms of latency and bandwidth as plain DRAM. *Memory* mode is a choice in scenarios in which the available memory capacity has to be increased and existing programs should remain unmodified. While the underlying Optane media is persistent, a data loss will occur on a power failure in *Memory* mode due to cache coherency.

In *AppDirect* mode, both DRAM and PMem are available as directly addressable memories and the persistence features of PMem can be utilized. If configured

[2] 4-socket compatible 3rd Generation Intel Scalable CPUs (codenamed Cooper Lake) only support a transfer speed of 2666 MT/s for PMem [30]. This also limits DRAM to 2666 MT/s.



(a) App-Direct mode (b) Memory mode

Figure 2.2.: Logical view on operation modes of Intel® Optane™ Persistent Memory [30, 69]

accordingly, both memory types are available as individual NUMA nodes and developers can explicitly control allocations on DRAM and PMem. This, however, requires modifications of the application to properly allocate memory on the different memory types depending on the use cases. A detailed explanation on how allocations can be controlled is given in appendix A.1.

A schematic visualization of the operation modes is given in figure 2.2. In this thesis, PMem is used in *AppDirect* mode as we are interested in using PMem as a large-capacity and DRAM as a directly addressable memory. In order to configure PMem as separate NUMA nodes in AppDirect mode, the open-source tools `ndctl` and `daxctl` are required. This configuration process has been described by Steve Scargall [79]. Furthermore, `ipmctl` is a tool for configuring and monitoring Intel persistent memory modules [79].

Performance Characteristics

While PMem shares the same form factor as DDR4 modules and suggests DRAM-like performance, it features some key performance characteristics.

The underlying 3D XPoint media has a minimum access granularity of 256 Bytes. This way, a read of a single Byte requires 256 Bytes to be read from the media, which is unproblematic in sequential access as the data loaded from the media is cached but can lead to a significant overhead for small, random accesses. A write access of a single Byte requires reading 256 Bytes, modifying the single Byte in this 256 Byte buffer, and writing the whole 256 Byte buffer back to the storage media [103], resulting in a significant write-amplification. The DIMM itself only supports accesses of a 64 Byte cache-line granularity but an on-dimm buffer (*XPBuffer*) can combine multiple 64 Byte accesses into a single 256 Bytes (*XPLine*) write to the media. An important performance metric is the Effective Write Ratio (EWR), which is the number of Bytes written by the memory controller divided by the number of actual

Bytes written to the 3D XPoint media. An EWR less than 1 means that writes are inefficient [103].

In addition, PMem shows a decreasing performance with a higher number of threads accessing memory concurrently [45, 98, 100]. Possible reasons for this are a contention at the read and write queues at the integrated memory controller (iMC) of the CPU and a contention of the XPBuffer [103].

Experiments by Kozhokanova et al. [45] revealed that PMem has a latency twice as high compared to DRAM with one thread and more than three times higher compared to with 32 threads in sequential access. For random access, the PMem latency can be more than eight times higher than DRAM in recommended system configurations. The bandwidth penalties when using PMem compared to DRAM are in the same order of magnitude.

Module Population Rules

PMem still requires traditional DRAM installed on the system - usually, a ratio between 4:1 and 16:1 between PMem and DRAM capacity is recommended for Memory mode. While, in theory, a single DDR4 DIMM paired with one PMem module is sufficient to operate PMem in AppDirect mode, mainboards may support only specific module combinations. For example, our Intel Server Board M50CYP2SB requires at least 6 DRAM modules installed to operate a single PMem DIMM in AppDirect mode per socket according to the technical product specification [31]. Other supported configurations are, for example, 4xDRAM + 4xPMem, 8xDRAM + 1xPMem, and 12xDRAM + 2xPMem. However, we found out that booting with 1xDRAM + 1xPMem also works with AppDirect mode, although it is not specifically recommended. In particular, the statement that "each memory channel [..] must be populated with at least one DRAM DIMM" regarding 1st generation PMem modules does not hold anymore for 2nd generation PMem [100].

Power Supply of Optane PMem

While the PMem modules share the same physical interface with normal DDR4 RDIMMs, its power supply is inherently different and not specified in detail in publicly available datasheets or documentation.

As described in section 2.1.3, DDR4 RDIMMs are powered primarily via an 1.2 V `VDD` line and an additional 2.5 V word-line boost rail called `VPP`. The main power to the PMem modules is delivered via a 12 V line, specified in the DDR4 interface but unused in normal RDIMMs. This 12 V rail powers a Power-Management Integrated Circuit (PMIC) that generates a number of different voltages required for the Optane Media, the Optane Controller, and the energy-storage supercaps[3] directly on the module [55]. This is a key difference to RDIMMs, which do not have any power conversion directly on the module[4]. The 1.2 V `VDD` rail is also used by PMem but

[3]These capacitors provide power to flush write queues on a power-loss

[4]DDR5 modules have an on-DIMM PMIC.

only shows a minor contribution to the total power consumption of the module[5].

Power Control and Monitoring using `ipmctl`

Similarly to Intel's RAPL (see section 2.3.2), Intel Optane PMem 200 Series modules feature detailed power limitation and reporting features, that can be controlled using the `ipmctl` command-line interface. This tool supports setting different properties for each DIMM. However, each DIMM within a system is recommended to be configured identically.

The average power consumption can be accessed for the 12 V and the 1.2 V rail individually by querying the `Average12vPower` and `Average1_2vPower` property respectively. The averaging interval is 1000 ms by default and can be increased in steps of 100 ms by modifying the `AveragePowerReportingTimeConstant` property. An example for querying the power for a single PMem module with id 0x001 is shown in listing 2.1. This query, however, results in a latency of 1–2 seconds on our test system.

```
$ ipmctl show -ddrt -d AveragePower,Average12vPower,Average1_2vPower
   ↪ -dimm 0x001
   AveragePower=3485 mW
   Average12vPower=2468 mW
   Average1_2vPower=1017 mW
```

Listing 2.1: Example output for PMem power measurements with `ipmctl`

The power consumption of the module can also be limited by modifying the `AvgPowerLimit` property, which is set to 15 W by default. Using the `MemoryBandwidthBoostMaxPowerLimit` a higher power limit for a short period can be specified. By default, 18 W can be drawn for up to 15 seconds [53].

2.2. Memory Access Pattern Classification

The presented memory types and memory hierarchies are expected to perform differently depending on how the memory accesses are organized. While the number of access patterns is practically unlimited, accesses are usually classified into four general patterns [67]. The *constant* memory access pattern is the simplest and best-performing access pattern. It occurs if the same memory location is accessed repeatedly, for example, in each loop iteration. Typically, the compiler places this memory value in a register, the fastest type of memory in a computer, and thus it can be accessed with very low latency. This pattern exhibits an excellent temporal locality but no spatial locality.
In *sequential* memory access, consecutive memory locations are accessed after each other. This pattern exhibits an ideal spatial locality but generally a poor temporal

[5] `VPP` consumes about 1 mW at idle and 1.25 mW under load according to our measurements

locality. Due to the organization of the memory into cache lines in combination with prefetching techniques, these memory accesses perform well on modern systems.
In a *strided* memory access, consecutive memory locations with a fixed gap, also called stride, in between are accessed. If the stride is 1, this is equivalent to the sequential memory access. A small stride with subsequent memory accesses that are in the same cache line will also result in good performance. However, a larger stride will decrease spatial locality, and if larger than a cache line or even greater than the memory page size will significantly impact performance.
An *irregular* access pattern, also called **random** access, is a sequence of memory accesses that is unpredictable or inherently irregular. These patterns usually exhibit a bad spatial locality and, thus, poor cache-line utilization. This increases the latency of each memory access, which usually cannot be hidden by pipelining techniques and thus decreases performance. A prominent example of irregular memory accesses is pointer-chasing. In such algorithms, the computation of the next memory access location depends on the value at the previous location. Another example is a linked data structure, e.g., linked lists of items in which each item contains the pointer to the next item or operations on sparse matrices, such as in compressed-row storage format.

2.3. Power Measurement Techniques

In this section, we describe different techniques to measure the power consumption of HPC systems on a node level. We distinguish between integrated power measurement techniques that can be accessed from the system under test without additional hardware and external measurement techniques that require the instrumentation of the system with physical power and energy sensors [23].

2.3.1. Power Consumption Basics

The two terms *power* and *energy* are often used interchangeably but describe different concepts. The power P, with units in Watts [W], is the rate at which energy is dissipated. By integrating the power P over time, we can compute the energy E in Joules [J] or Watt-Hours [Wh] units [77]. In an electrical circuit, the instantaneous power consumption can be computed using $P = U \times I$ with U being the voltage and I being the current.

The power consumption of computer components can be divided into static power and dynamic power. Static power (P_s) is the constant power required to maintain the operation at no computational load. The dynamic power (P_d) is the power drawn by the actual computational load, e.g., on CPUs, it scales with the CPU frequency.

2.3.2. Running Average Power Limit (RAPL)

Running Average Power Limit (RAPL) is an interface developed by Intel with two main functionalities. It allows measuring the energy consumption of different hardware components at a sampling rate of about 1 ms and supports capping their energy consumption within a time window to ensure that a system stays within certain power and thermal limits. RAPL was first introduced on the Intel Sandybridge micro-architecture in 2011 and evolved over different Intel architectures in the past [41]. Starting with the AMD Zen micro-architecture, AMD implemented a semi-compatible version of RAPL [44, 81].

RAPL groups different system components into so-called power planes (also called domains) and reports the aggregated energy consumption for each of these groups.

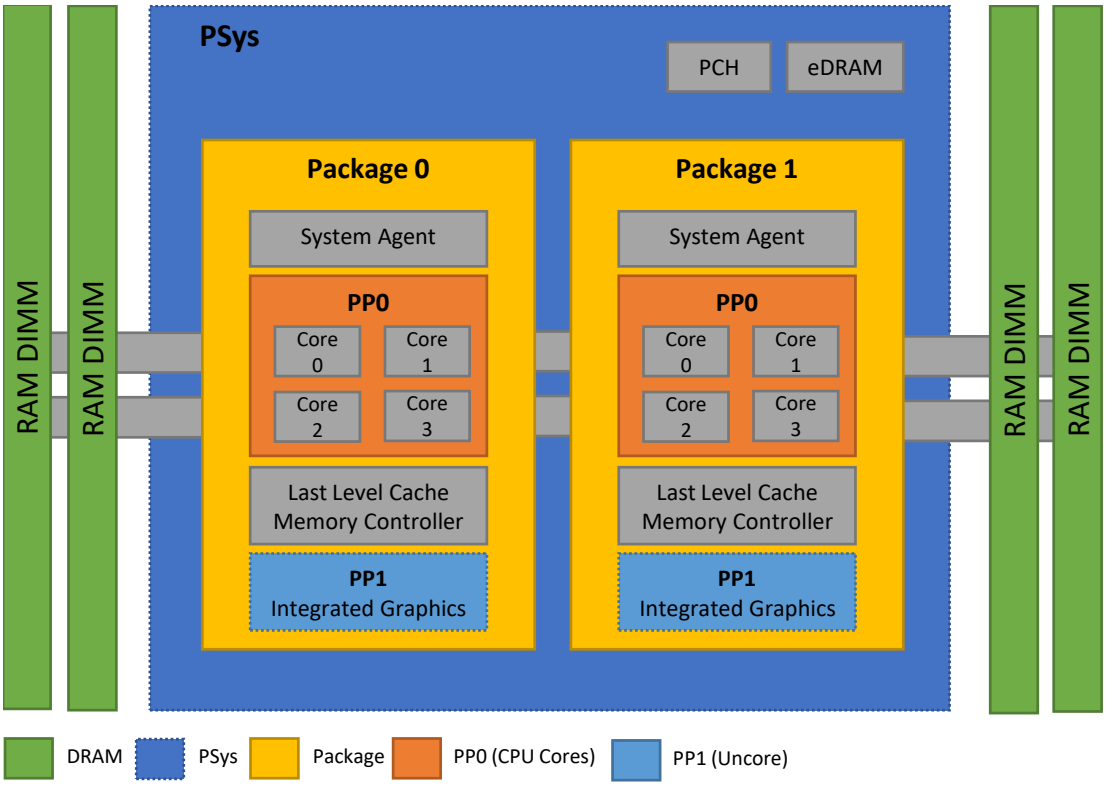


Figure 2.3.: RAPL power planes on recent Intel architectures [27, 43]

Figure 2.3 visualizes the power planes supported by RAPL. The power-planes defined by Intel are [27, 43]:

- The Package **(PKG)** power plane reports the energy consumption of the CPU package. This includes power consumption of CPU cores, integrated graphics, and some uncore components, such as memory controllers and last-level caches.
- The PowerPlane 0 **(PP0)** covers the power consumption of the CPU cores.

- The PowerPlane 1 **(PP1)** is only available on desktop CPUs and covers the power consumed by uncore components such as integrated graphics.
- The **DRAM** power plane reports the energy consumption of RAM directly connected to the memory controller of the package. Although the name does not suggest it, this also covers other DIMM types than DRAM, for example, Intel® Optane™ Persistent Memory. To reduce confusion, we will refer to this domain as the *Memory Domain*.
- The **PSys** power plane reports the energy consumed by the complete System-on-a-Chip (SoC) on single-socket systems. This includes the package and other domains, such as the Platform Control Hub (PCH) or the system agent.

Multi-socket systems report these power planes separately for each socket. On Intel systems, the available power planes differ between server and desktop CPUs and different micro-architectures. The Package domain is the only one supported on every Intel system. Table 2.2 shows an overview of the RAPL support on important Intel server micro-architectures [28]. Current HPC systems overall only support the package and memory power planes and, while the PSys domain is specified in theory, we did not find any HPC system that supports it.

Architecture	**Package**	**PP0**	**PP1**	**Memory / DRAM**	**PSys**
Knights Landing	✓	×	×	✓	×
Sandybridge-EP		✓			×
Haswell-EP		✓			×
Broadwell-EP		✓			×
Skylake-SP		×			(✓)
Cascade Lake-SP					
Ice Lake-SP					
Sapphire Rapids					

✓ Supported
× Not supported
(✓) Support depends on vendor-specific power delivery

Table 2.2.: RAPL power planes support on different Intel server micro-architectures [28].

Model-specific Registers

Low-level access to the RAPL measurements is provided through model-specific registers (MSRs), which are 32-bit special-purpose registers that are specific to certain processor generations. On Intel x86 systems, these registers are documented in the "Intel® 64 and IA-32 Architectures Software Developer's Manual" [27].

For each power plane, a set of MSRs is available to access power measurements and control the power capping. Each power plane features a set of MSRs that are similar across different power planes. As an example, the set of MSRs from the package (PKG) power plane is described below:

- `MSR_PKG_ENERGY_STATUS` is a dynamic, read-only register that contains the energy consumed since the start of the system and is updated approximately every 1 ms. To get the energy consumption in Joule, this value must be multiplied with the platform-specific energy unit, which can be read from the `MSR_RAPL_POWER_UNIT` register. For example, the energy unit on Intel Skylake and Ice Lake is 61.04 µJ. As the energy consumption is stored as a 32-bit unsigned integer, regular counter overflows must be expected and handled accordingly.
- `MSR_PKG_POWER_LIMIT` is a read/write register to control the power capping functionality. Two different limitation rules can be individually specified using four attributes here:
  - Whether the power limitation rule is enabled.
  - The average power usage limit in a time frame specified in the following field[6].
  - The wall time the power limit applies to[7].
  - A binary flag to control if power states that are lower than requested by the operating system can be entered in the given time window.
- `MSR_PKG_POWER_INFO` is a static, read-only register that contains four values:
  - The system's thermal specification power (similar to TDP) as a discrete value [6].
  - The minimum and maximum power limit values for a window[6].
  - The maximum time window for power limiting[7].
- `MSR_PKG_PERF_STATUS` is a dynamic, read-only register containing the last time the package was throttled due to power limits.

Accessing these MSRs requires elevated privileges, which usually are only available to the `root` user. The energy consumption reported by RAPL is not associated with any timestamps, and the sampling frequency of it is subject to jitter [22] (see section 4.2). Thus, oversampling of the energy readouts, which might introduce a significant overhead [43], is required to counter this.

RAPL measurements have been integrated into several performance measurement tools such as *LIKWID*, *PAPI* [59], and *perf* [52]. In appendix A.2, we describe how these tools can be used for higher-level access to RAPL measurements.

[6]Value needs to be multiplied by the power unit specified in `MSR_RAPL_POWER_UNIT`

[7]Value needs to be multiplied by the time unit specified in `MSR_RAPL_POWER_UNIT`

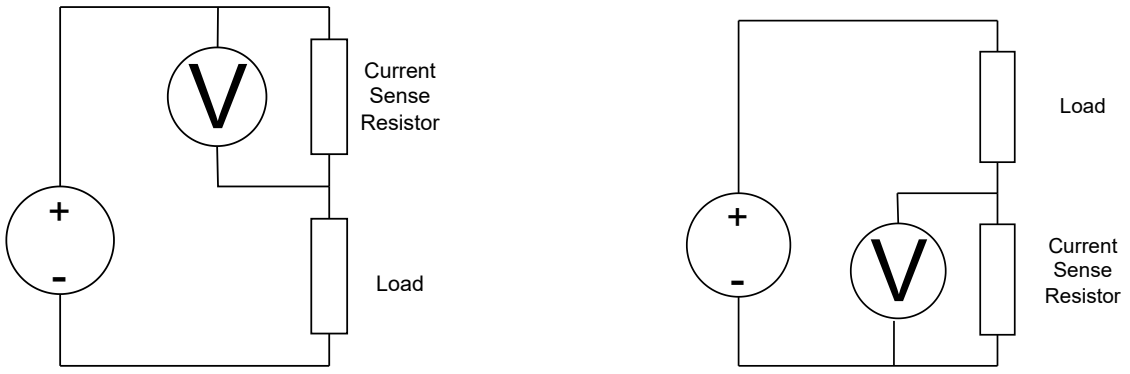
2.3.3. Physical Power Measurements

If no suitable integrated power measurement techniques are available, the only feasible solution is often using an external power measurement sensor. Two popular techniques in direct current (DC) circuits are direct measurements by integrating a current-sense resistor in the circuit or indirect measurements using a Hall Effect Current Sensor, each with different application scenarios. The latter technique utilizes the magnetic field induced by current flow to measure the current without invasive circuit modifications. This thesis only focuses on current sensing using resistors and describes how this can be implemented in detail.

Current Sensing using Resistors

A current-sense resistor (CSR) is a resistor with a very low resistance, often also called a shunt. The shunt is placed in the circuit to measure the power consumption of the consumers in the circuit. Due to the known resistance R of the shunt, a voltage drop ΔU will occur at the resistor. As this voltage drop ΔU is proportional to the electrical current I, Ohm's law can be used to compute the current in the circuit: $I = \frac{\Delta U}{R}$. Using the formula $P = U \times I$, the power can be computed with I being the current calculated from the voltage drop and U being the voltage before the shunt drop [77]. For example, if we measure a voltage drop of $\Delta U = 10mV$ across a $5\,\mathrm{m\Omega}$ current-sense resistor in a $5\,\mathrm{V}$ circuit, the current drawn by the load in the circuit will be $A = \frac{\Delta U}{R} = \frac{10\,\mathrm{mV}}{5\,\mathrm{m\Omega}} = 2\,\mathrm{A}$. The power drawn by the load can then be computed: $P = U \times I = 5\,\mathrm{V} \times 2\,\mathrm{A} = 10\,\mathrm{W}$.

High-side vs. Low-side Sensing In DC circuits, the shunt can be either placed on the high side (positive rail) or the low side of the load (negative rail or ground) for current measurements, which is visualized in figure 2.4.



(a) High-side current sensing

(b) Low-side current sensing

Figure 2.4.: High-side vs. Low-side current-sense resistor placement [5]

Choice of Current Sense Resistors The correct choice of the resistor is crucial for correct power measurement results [5].

First, the resistance of the CSR is important as a high resistance will also induce a high voltage drop that can affect the other components of the circuit. For example, using a 1 Ω resistor and having 2 A of current flowing through the resistor will create a voltage drop of 2 V. In a 5 V circuit, for example, on a computer's mainboard, this will mean that only 3 V instead of 5 V are available to the device we want to measure the power for. To reduce the impact on other components, the resistance of the shunt needs to be very low, usually in the order of milliohms.

Second, the voltage drop at the resistor will cause a power dissipation of $P = U \times I = I^2 \times R$ in the form of heat, which needs to be lower than the resistor's power rating. For example, a 5 mΩ resistor with a power rating of 0.5 W will only sustain a maximum current of 10 A.

Third, the exact resistance of the shunt needs to be known to compute the current I. Resistors are usually specified with a nominal resistance and a relative tolerance. For instance, a 1 kΩ resistor with 5% tolerance has an actual resistance between 0.995 kΩ and 1.005 kΩ. As this error is linearly propagated in power measurements, resistors with a low tolerance should be selected.

Voltage-drop Amplification

While very low resistance of the shunt and the resulting low voltage drop is required to limit the impact on the circuit, it also introduces the challenge of accurately measuring this voltage drop [93]. As most measurement devices, especially in lower price ranges, only offer accurate measurements in the order of millivolts, the voltage drop needs to be amplified, i.e., transformed to a voltage that is proportional to the voltage drop but also in a voltage range that can be measured well.

For this, instrumentation amplifiers (in-amps) or current-sense amplifiers can be used [93]. Both types are small integrated circuits (ICs), which usually take a differential voltage signal between two pins as an input and output a voltage proportional to the difference to another pin. The factor by which the voltage difference is amplified is called *gain*. These ICs also require a separate power supply.

Terminology for Amplification ICs While the basic concept of these amplifiers is inherently simple, several parameters need to be considered to ensure correct measurements and device compatibility. For this, some terminology has to be defined. The common-mode voltage (U_{CM}) between two differential voltages U_-, U_+ is defined as the voltage that is common between them, i.e, the average of them $U_{CM} = \frac{U_- + U_+}{2}$. The common-mode rejection ratio is the ratio to which the common-mode signal is rejected and should be as high as possible. Similarly, the power-supply rejection ratio describes by which degree the supply voltage is suppressed in the output. Input Bias Current I_B is the current flowing through the input pins of the amplifier and should be as low as possible. Typical values are in the order of pico to microampere. The quiescent current I_Q is the current for powering the amplifier. Amplifiers only guarantee accurate amplification until a specific frequency in the input signal. At higher frequencies, the gain gradually reduces. Usually, at higher configured gains,

the maximum supported input frequency drops. The slew rate is closely related to this and determines how fast the output voltage can change. It is usually given in the unit $\frac{V}{\mu s}$, i.e., it describes by how many volts the output voltage can change within a microsecond [88].

Current Sense Amplifiers and Instrumentation Amplifiers Both current-sense amplifiers (cs-amps) and instrumentation amplifiers (in-amps) can amplify a voltage drop at a shunt for digital current sensing. The critical difference between them is the input topology: Cs-amps support common-mode voltages significantly higher than supply voltages, while in-amps feature a higher input impedance and, thus, lower bias current. A high bias current can introduce significant measurement errors at low load currents. While most current-sense amplifiers have a fixed gain, the gain of many in-amps can be configured by connecting a gain resistor to the Integrated circuit (IC). The value of this resistor defines the gain, which increases flexibility but also introduces another source of errors [88, 90, 92].

Data Acquisition

To measure the power consumption of the load in the circuit, both the voltage drop at the resistor (ΔU) and the common-mode voltage (U) need to be measured. Both voltages need to be converted to a discrete, digital voltage value for further digital processing, such as integrating power over time to report the energy consumption or power sampling over time. Here, many devices, such as analog-digital coverters (ADC) ICs, data acquisition cards, high-quality digital multimeters, and oscilloscopes, are available, each with different use cases, precision, sampling intervals, and cost.

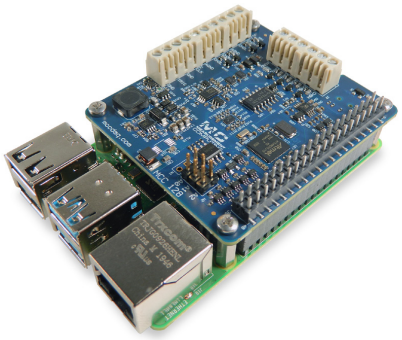


Figure 2.5.: MCC 128 voltage measurement DAQ HAT for Raspberry Pi [60]

The *MCC 128* is a data acquisition (DAQ) device for monitoring analog voltages on eight single-ended channels at an aggregated sampling rate of 100,000 samples per second [60]. Singe-ended means that the voltage is measured between the pin for a channel and ground. It also supports differential measurements in which the difference between two pins is measured, resulting in up to 4 differential channels. The device can be installed as a HAT on a Raspberry Pi, which also controls the data acquisition (see figure 2.5). A special pin of the MCC128 can be configured to trigger the measurements when a configurable digital signal (e.g., a rising edge) is detected.

Some products facilitate the energy measurements using CSRs. For example, the

LTC2947[8] combines the shunt, analog to digital conversion, and accumulated energy reporting into a single device. This way, the energy consumption over time and the minimum and maximum power consumption can be accessed easily as well. Other manufacturers offer comparable products, for example, the *INA260*[9] digital power monitor.

Current-sense resistors are a favorable choice in many use cases due to their low cost, excellent integrability, and high accuracy. However, the power and heat dissipation of the resistor can quickly become problematic. Thus, other solutions, such as hall-effect sensors, need to be considered if power dissipation exceeds the rating of the resistor.

2.4. Platform Power Delivery

After introducing how RAPL can be used for power measurements and budgeting, we now explain how power is delivered to the different components on modern servers. This is important for understanding how RAPL works internally. Modern HPC nodes feature a large number of components, each requiring specific DC supply voltages, which need to be provided by the platform's power delivery system. For this, the 110 V to 240 V alternating circuit (AC) input from the wall socket or the power-distribution unit (PDU) is first converted into 12 V DC by the power-supply unit (PSU) [31], which then is converted into a set of different voltages to power the CPU package, the memory, and many more consumers including disks, fans, network-interface cards (NICs), and the baseboard management controller (BMC).

The power delivery on a dual-socket Intel Ice Lake server system is schematically visualized in figure 2.6, which we created based on several sources and documentations. Traditional CPUs require several voltages delivered by the mainboard for different on-die components, such as CPU cores, IO, and other uncore components. However, the Intel Haswell and Ice Lake architectures feature a Fully Integrated Voltage Regulator (FIVR) on the CPU socket. Such a FIVR takes a single voltage, usually 1.8 V produced by another voltage regulator on the mainboard, as an input and converts it to all voltages required by on-package components, requiring less power delivery components on the mainboard and granting more control of power delivery to the CPU. However, this also increases the temperature dissipation on the CPU die [18, 26].

On the *Intel M50CYP* [31] server, which supports eight memory channels with 2 DIMM slots each per socket, the channels 0 to 3 and 4 to 7 are powered independently by the mainboard. For this, the 12 V input from the PSU is converted into 1.2 V VDD, 2.5 V VPP, and 12 V by a voltage regulator module (VRM) (see figure 2.6). An additional 0.6 V termination voltage (VTT) is generated from the 1.2 V line. In addition, a 1.2 V supply voltage is fed into the CPU package. While it is not

[8] `https://www.analog.com/en/products/ltc2947.html`
[9] `https://www.ti.com/product/de-de/INA260`

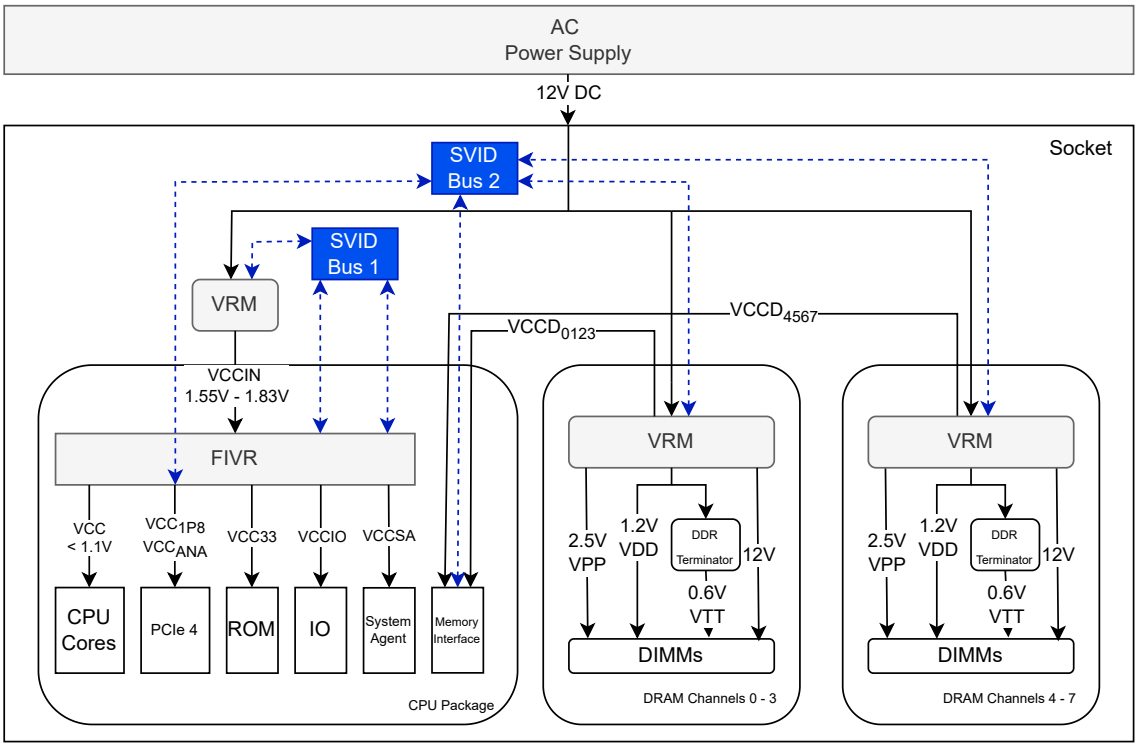


Figure 2.6.: Schematic visualization of CPU and memory power delivery on Intel Ice Lake-SP server systems [18, 26]

explicitly specified, it can be expected that this supply voltage is also provided by the same voltage regulators that power the memory modules.

DC-to-DC Power Converters The conversion of the 12 V line to lower voltages requires DC-to-DC power conversion. A widely used type of these converters is a Buck Converter, which steps down an input voltage, for example, 12 V, to a constant, usually configurable output voltage, for instance, 3.3 V. Such a converter consists of a high-side MOSFET that switches between outputting a 12 V and 0 V voltage at a high frequency. This voltage then passes a low-pass filter, which consists of an inductor and a capacitor, to produce a smooth output voltage, and an additional low-side MOSFET (or a diode) ensuring that current can flow through the filter [18].

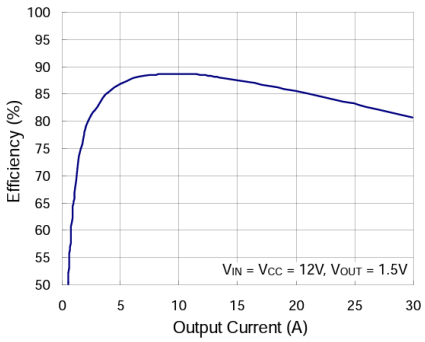


Figure 2.7.: Output current vs. efficiency for the *RT8120* voltage regulator [74]

Buck Converters can achieve a high efficiency of up to 90% at higher currents, but efficiency drops significantly at lower load

currents as shown in figure 2.7 [18]. Multiphase Buck Converters are often used in scenarios with higher current demand and smooth output voltage requirements, for example, delivering power to memory or the CPU. This is achieved by combining multiple buck converters coupled with a controller chip that produces the switching signal for the high-side MOSFET of the different converters and ensures that the signals are phase shifted, in particular, that at most one of the signals is high simultaneously. Controllers of modern multiphase converters often feature phase shedding, which automatically turns off phases at a low output current to ensure high conversion efficiency even at low load.

Overall, buck converters have become highly optimized and complex components that are important in energy-efficient power delivery on mainboards. For this thesis, only a fundamental understanding of buck converters and their energy efficiency characteristics is required [18].

Communication between VR and CPU On Intel server systems, Voltage Regulators (VRs) used to power CPUs or the memory are required to implement the Serial Voltage Identification (SVID) interface. This bus protocol enables communication between the VR controller on the mainboard and the CPU package. SVID sets the required voltage that the VR outputs to a specific rail, which is essential for CPU power management. Furthermore, SVID supports reporting voltages and current (IMON) at the VR to the CPU for monitoring purposes. This forms the base for RAPL (see section 2.3.2), which provides power measurements based on current and voltage measurements at the VR and implements power limiting based on these measurements [18, 32]. Figure 2.6 shows the communication paths of the SVID bus schematically on the Ice Lake-SP platform. The specific mechanism behind SVID and RAPL is not specified publicly and is probably highly platform-dependent. Intel specifies the exact requirements for voltage regulators (VRs) on mainboards for the Ice Lake platform in the VR 13.HC document, which is a version of VR 13 with support for monitoring higher currents [26].

Apart from SVID, a couple of other bus protocols are often used for communication between system components, particularly for power monitoring and management. The Power Management Bus (PMBus) is a bus protocol that is based on an I2C protocol for monitoring, e.g., the input and output voltage and current of compatible PSUs. System Management Bus (SMBus) is a similar protocol for lightweight communication between different platform components, particularly for system monitoring and control. It is also used for reporting temperature sensor readings, for example, at DIMMs or voltage regulators, and power measurement data from compatible voltage regulators similar to SVID. Monitoring data from both buses is usually processed and exposed through the Intelligent Platform Management Interface (IPMI) [18].

All this background is required to understand how power measurements using RAPL work internally and how we can physically measure the power consumption of the memory by instrumenting DIMMs to validate RAPL.

3. Related Work

This chapter provides a comprehensive overview of the research conducted on power consumption analysis of different workloads, RAPL validation, power models for memory, and energy metrics in the field of HPC.

3.1. Workload-specific Power Consumption Analysis

Arafa et al. [2] measured the energy consumption of different instructions on GPUs of four distinct architectures. The energy consumption is measured using different software interfaces that were validated using physical power measurements at the GPU with current clamps at the ATX power connector and PCIe power supply via a custom PCIe extender. Allen et al. [1] evaluated the power drawn by a GPU for different types of memory accesses, such as sequential and random reads or writes. Ghose et al. [16] studied the energy consumption of different generations and low-power variants of DRAM on multiple workloads. They used power models based on power specifications from memory vendors for energy consumption estimations. The same authors also assessed that these power models are unreliable because the actual current drawn by the memory can differ significantly from the current specified in the memory's datasheet and that other parameters, such as the kind of data read, e.g., whether only ones or zeros are read from memory, affects power consumption [17].

Schöne et al. [83] evaluated how data access to different types of caches and the main memory affects the cache access rate, the instructions per cycle, and the total system power consumption.

3.2. Validation of RAPL Energy Measurements

This section describes previous work in validating the performance and accuracy of RAPL.

Desrochers et al. [10] evaluated the accuracy of the RAPL package and memory domains individually on different Intel Haswell platforms by instrumenting it with power measurement sensors. They measured the power consumption drawn by the CPU by instrumenting the P4 power connector, i.e., the cable that powers the CPU from the power supply, with Hall Effect Current Sensors. Server systems usually do not feature such a connector, so the authors did not measure the power consumption of server CPUs. In addition, the authors used DIMM riser cards both

for DDR3 and DDR4 memory with current-sense resistors on the VDD and VPP pins to measure the power consumption of a single memory DIMM. The voltage drop at the resistor, which is proportional to the current flowing through the pin, was amplified by a factor of 100 for VDD and 300 for VPP using a *Texas Instruments INA 122* instrumentation amplifier and then fed into a *USB-1208FS-Plus* data acquisition device. Power drawn from the wall socket is measured using the WattsUpPro USB power measurement device. All connections are made on breadboards. These three power measurements are collected by a Raspberry Pi and synchronized with RAPL measurements of the package and memory power plane by outputting a signal over a serial port to the data acquisition device when RAPL measurements are started on the machine under test. The authors collected RAPL measurements using the *perf* tool at a comparably low frequency of 10 Hz (100 ms sampling interval) due to a higher overhead of this tool at higher sampling rates. They proposed the use of a different interface for measuring RAPL (e.g., LIKWID) or directly reading the RAPL MSRs for lower overhead access. Their results showed that RAPL measurements match the actual power consumption by a margin of at most 20% for 8 GB DDR4 memory and by a margin of less than ±6% for 16 GB modules under load on the Haswell-EP server platform. At idle, this error is significantly higher and can exceed 75%. For Haswell desktop systems, the error of RAPL is significantly higher and either subject to a constant offset or the RAPL memory power consumption is not modeled well overall depending on the memory modules used [10].

Hackenberg et al. [20] evaluated the accuracy of the RAPL measurements on Intel SandyBridge-EP with DDR3 memory and on Intel Haswell-EP with DDR4 memory. For this, the power drawn by the whole system was measured using the LMG450 AC power analyzer and was compared to the sum of power reported by the RAPL package and memory power planes for different workloads. As the external power measurements also included off-package power consumption, a good fit was defined as the existence of a continuous function between both measurements. While this was not the case on the SandyBridge-EP platform for some workloads, a strong agreement between these values was assessed for the Haswell-EP platform. The authors also carried out this evaluation on an AMD Zen 2 system [81], which offers a power-limiting and reporting interface semi-compatible with Intel's RAPL. Here, similarly to the Intel Sandybridge-EP platform, a poor correlation has been observed for most workloads, indicating that RAPL readouts were modeled and not measured.

T. Ilsche's dissertation on the topic "Energy Measurements of High Performance Computing Systems: From Instrumentation to Analysis" [23] includes - apart from many other contributions - techniques to instrument HPC systems for detailed and high-resolution power measurements. This includes the instrumentation of different DC consumers on the mainboard, such as voltage regulators, processor power connectors, ATX mainboard power supply, SATA connectors, fans, and memory DIMMs for power measurements. The accuracy of RAPL counters for package and memory was validated on Intel SandyBridge and Skylake architectures by instrumenting the 12V voltage lines to the CPU and memory with the LMG670 power analyzer. The author also presents a novel approach for synchronization between power traces of

different systems, for example, a server under load and a microcontroller collecting external power measurements of that server, using correlation sequences. This is achieved by generating a distinctive pattern in the power consumption signal of the machine under test before and after the execution of the actual compute kernel. Identifying this pattern in the power traces ensures a tight synchronization between both traces.

Pitz et al. [70] analyzed the accuracy of different power sensors on an AMD Zen 3 system with different NVIDIA Tesla GPUs by comparing the measurements of integrated sensors with reference measurements conducted using a ZES Zimmer LMG671 power analyzer that measured the total power drawn by the redundant power supplies of the server. They also included package power measurements using the `zenpower3` driver, which is based on RAPL-like measurements on AMD-based systems. The authors conclude that sensor measurements can increase the system's power consumption and that platform power sensors, i.e., sensors that measure the power drawn by the whole node, feature an insufficient sampling rate. Additionally, they point out that an accuracy analysis of power sensors is mandatory for conducting meaningful measurements and optimizing energy efficiency.

In addition to the named work, we list several other evaluations conducted on the accuracy of RAPL measurements in table 3.1.

Authors	**Year**	**Platform**	**Power Plane**	**Measurement Methodology**
Dongarra et al. [12]	2012	Sandy Bridge (SB)	Pkg	PowerPack
Demmel et al. [9]	2012	SB	Pkg, Memory	Wall power measurements
Rotem et al. [78]	2012	SB	Pkg, CPU, Graphics	Wall power measurements
Mazouz et al. [58]	2014	SB Ivy Bridge (IB)	Pkg	Wall power measurements
Thomas et al. [72]	2014	SB	Pkg	ATX connector
Khan et al. [42]	2016	Haswell (HSW), SB	Pkg, Memory	Smart Wall Plug
Paniego et al. [68]	2018	HSW	Pkg	12V connector using Hall sensor
Fahad et al. [15]	2019	HSW, Skylake (SKL)	Pkg	WattsUpPro

Table 3.1.: Studies on RAPL validation using external power measurements

Several other works evaluated RAPL's other aspects. Khan et al. [41, 43] conducted a more general evaluation of RAPL. They assessed that the update interval of the RAPL CPU counter (PP0) is around 60 µs on Intel Skylake compared to

1000 µs on Intel Haswell and that the counter update rate is subject to significant jitter. Furthermore, they discussed the limitations of RAPL, such as a possible overflow of registers, the lack of timestamps associated with RAPL, and unpredictable timings of RAPL register updates. Hähnel et al. [22] discussed how RAPL can be used for power profiling of short code kernels.

Overall, RAPL measurements are inaccurate up to the Intel Sandybridge architecture or Intel Haswell desktop systems. The reason for this is that the energy consumption provided by RAPL was based on a calibrated energy model based on performance counters [8], often also referred to as a digital power meter (DPM) [18]. While an initial evaluation from Intel [8] showed an overall good approximation within 1% to reference measurements, later evaluations assessed that it can be inaccurate for certain workloads. With RAPL being based on actual measurements on more recent architectures, it is considered overall "useful" and "valuable" in the literature [10, 20]. However, the accuracy of the RAPL memory domain has not been evaluated yet on modern architectures, such as 3rd Generation Intel Xeon Scalable CPUs (e.g., Ice Lake-SP) with different power distribution topologies than previous architectures and high-capacity memory support.

3.3. Memory Power Modeling

Micron provides a technical note on how the power consumption of DDR4 SDRAM memory can be estimated based on the signals and commands sent to the memory. This is based on the current flowing through the 1.2 V VDD and 2.5 V VPP pins specified in the *IDD* section in the DIMM's datasheet. This computation requires a deep understanding of how DDR4 SDRAM works internally and is difficult for determining the power consumption of system applications [62]. In addition, the specified *IDD* values have been shown to differ significantly from the actual current by Goose et al. and that power modeling based on these values can be subject to errors as high as 160.6% [17]. Instead, the authors propose the *VAMPIRE* DRAM power model that estimates DRAM power consumption at a mean absolute percentage error of at most 7.1%. In response to the pessimistic *IDD* values in the datasheets, Mathew et al. [56] developed a bank-wise DRAM power model based on calibration data from their own measurements. Zhang et al. [105] developed a Dynamic Adaptive Model of CPU and memory power based on hardware performance counters. Overall, we are not ware of other approaches utilizing an access-pattern-based memory energy prediction.

3.4. HPC Energy Metrics

In HPC, performance metrics play an important role in quantifying the performance of an application, a hardware component, a compute node, or even a whole supercomputer. Prominent examples are, for example, the number of floating-point

operations per second, the arithmetic intensity, or the sustained memory bandwidth.

With increasing awareness regarding energy efficiency, energy or power consumption has been fused with existing metrics that assess compute performance. The most popular metric of this type is the number of floating-point operations per power consumption, i.e., `GFlops/Watt`, which is used as the primary metric for ranking supercomputers by their energy efficiency in the Green 500 ranking [95]. The energy-delay product (EDP) is a well-established metric for assessing an application's energy efficiency. It is a fused metric that is based on the time-to-solution `T`, i.e., the wall time required to solve the problem, and the energy-to-solution `E`, i.e., the energy for solving the problem [47]:

$$EDP = E \times T^{\omega}$$

The scalar ω is a hyperparameter to describe the impact of the performance to the metric. In scenarios where a higher performance is more important than low energy consumption, ω can be increased, usually in the range of 1 to 3 [47]. Roberts et al. [76] argue that the class of EDP metrics is "unsuitable for energy-aware software optimization".

Zhang et al. [104] characterized the memory energy efficiency (MEE) using the Bandwidth per Watt (BpW) metric, which is the perceived memory bandwidth divided by the memory power consumption:

$$MEE = BpW = \frac{Bandwidth[MB/s]}{MemoryPower[W]}$$

Several studies used the Energy per Instruction (EPI) [19] metric, defined as the CPU or GPU's energy consumption to execute a specific instruction, or variants of it. Kulkarni et al. [46] presented an approach for measuring the instruction-level power consumption in battery-operated embedded systems. Molka et al. [63] characterized the energy consumption of the CPU for memory transfer instructions to different layers of the memory hierarchy on Intel Westmere and AMD Istanbul systems based on external node-level power measurements. Overall, we did not identify any previous work that developed metrics characterizing the dynamic energy consumption of the memory.

4. Validation of RAPL Measurements on Intel Ice Lake Architecture

As described in chapter 3, numerous researchers previously evaluated the accuracy of the RAPL package power measurements on different Intel and AMD architectures. Two studies also evaluated the accuracy of RAPL energy values for the memory domain [10, 23]. However, no work validating RAPL on the Intel Ice Lake platform equipped with DRAM or non-volatile memory such as Intel® Optane™ Persistent Memory is available. In this chapter, we validate the memory power measurements using RAPL on the Intel Ice Lake platform. We evaluate the timing of RAPL updates and validate the RAPL power measurements of the memory domain against reference measurements at the memory DIMMs by instrumenting the memory slot with a riser card.

4.1. Test System

The main test system on which we conduct our experiments is a dual-socket Intel Ice Lake system from the Intel Optane Cluster that is part of the RWTH HPC cluster. In addition, we conduct measurements on older Intel Broadwell and KNL systems as a reference point. Specifications of these systems with the default memory populations are given in table 4.2. Three different kinds of DIMMs, as shown in table 4.1, are available for installation on these systems.

Vendor	**Name**	**Capacity**	**Transfer Rate**	**Part Number**
Samsung	DDR4 SDRAM ECC RDIMM	32GB	3200 MT/s	M393A4K40DB3-CWE[1]
SK hynix	DDR4 SDRAM RDIMM	16GB	2400 MT/s	HMA42GR7AFR4N[87]
Intel	Optane™ PMem 200 Series	128GB	3200 MT/s	NMB1XXD128GPSU4[1]

[1] No datasheets are available publicly.

Table 4.1.: Memory modules used for experiments

Intel architecture	**Ice Lake-SP**	**Broadwell-EP**	**Knight Landing**
Mainboard	Intel M50CYP2SB[1]	Supermicro X10DRT-H	Supermicro K1SPi
No. sockets	2	2	1
CPU model	Xeon Gold 6338	Xeon E5-2650 v4	Xeon Phi 7210
Base frequency	2.0 GHz	2.2 GHz	1.3 GHz
Cores	32	12	64[2]
L1-D Cache	48 kB	64 kB	4 MB
L1-I Cache	32 kB		
L2 Cache	2.8 MB	3 MB	32 MB
L3 Cache	48 MB	30 MB	-
TDP	205 W	105 W	215 W
Main memory	16x32GB Samsung DDR4 3200 MT/s	8x16GB SK hynix DDR4 2400 MT/s	6x32GB SK hynix DDR4 2133 MT/s
Other memory	16x128GB Intel® Optane™ PMem	-	4x4GB MCDRAM 6400 MT/s
Power governor	performance		powersave
OS	Rocky Linux 8.8 (Green Obsidian)		CENTOS 7.9
Kernel	6.4.3-1.el8.elrepo		3.10.0-1160.95.1.el7

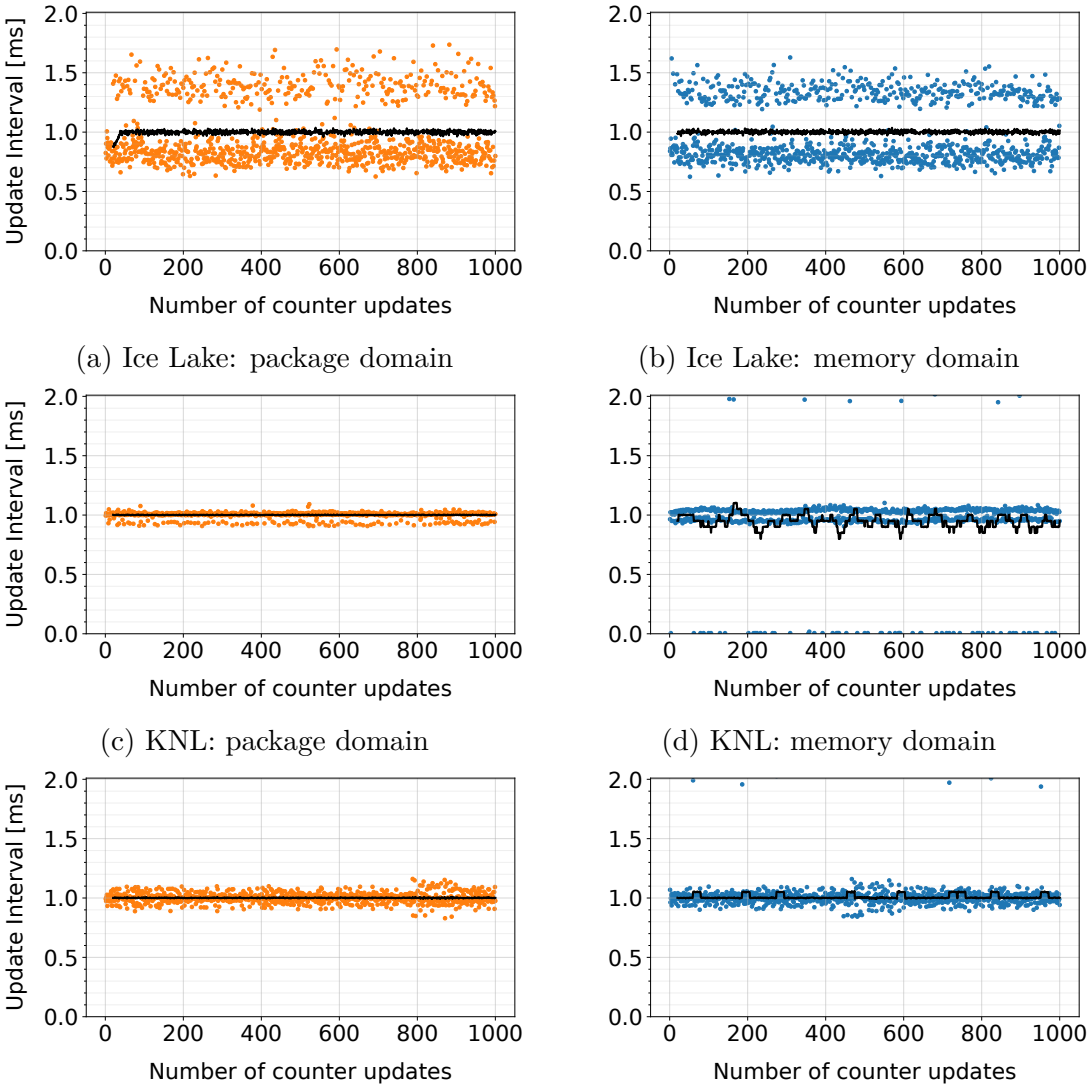
[1] Engineering sample
[2] Sub-NUMA Clustering enabled

Table 4.2.: Default hardware configuration of the Ice Lake, Broadwell, and KNL systems

4.2. Update Interval of RAPL Counters

According to the official documentation by Intel [27], RAPL values should update approximately every 1 ms. As previous work on other architectures showed a considerable variance and jitter in the update rate [41, 84], conducting the same validations on the architectures we use in this thesis, namely Intel Ice Lake and older platforms such as Intel Knights Landing (KNL) and Intel Broadwell, is highly motivated. For this, we used the code by Schöne et al. [81], that busy waits for an update of the RAPL MSRs and measures the number of cycles[1] between each update. A code snippet can be found in listing B.1. We convert the number of clock cycles between each counter update into a wall time delay by multiplying it with the clock speed and present the results for these experiments in the following.

[1]The timestamp counter using the `RDTSCP` instruction is used for low overhead measurements.



(a) Ice Lake: package domain

(b) Ice Lake: memory domain

(c) KNL: package domain

(d) KNL: memory domain

(e) Broadwell: package domain

(f) Broadwell: memory domain

Figure 4.1.: RAPL update interval distribution for RAPL package and memory domains on the Intel Ice Lake, Broadwell, and KNL platforms

4.2.1. Experimental Results

The described experiments have been executed five times on Intel Ice Lake, Broadwell, and KNL systems (see table 4.2). The results here cover a short yet representative excerpt of the data to facilitate visualization. The results of the experiments are shown in figure 4.1. Each plot shows the gap between consecutive RAPL counter updates in milliseconds for 1000 counter updates observed. The black line shows the running average of 20 data points.

4.2.2. Discussion

For the Ice Lake system, the RAPL counters for both the package domain (figure 4.1a) and the memory domain (figure 4.1b) are updated every 1 ms on average. However, the update interval is subject to significant jitter, with almost no update observed at an exact 1 ms gap. Instead, most updates are registered either after a gap of about 1.4 ms or after 0.8 ms averaging 1 ms. While this is consistent with the documentation and previous work, the jitter is more significant than on the Skylake or Haswell platform [41].

The results for the KNL system are inherently different. As shown in figure 4.1c, the RAPL counter for the package domain is updated roughly every 1 ms with some updates being delayed by several microseconds and a couple of updates coming in earlier after a delay of about 0.95 ms. This is more consistent than on the Ice Lake system. In contrast, the measurements for the RAPL memory counters on the KNL platform shown in figure 4.1d show that this counter is not updated every 1 ms on average. While a majority of the updates is observed either after 0.95 ms or 1.05 ms, some significant outliers either at about 2 ms or a few microseconds can be observed. This means that some updates are skipped while sometimes two updates come in basically at the same time. The repetitive execution of the same experiments showed this consistently. On the Broadwell system (figures 4.1e and 4.1f), the update interval is similar to the KNL but more evenly distributed between 0.9 ms and 1.1 ms.

Possible reasons for this variation are overhead introduced by oversampling the RAPL counters, inaccurate updates of the CPU timestamp counter, or jitter introduced by the operating system. In any of these cases, RAPL counters cannot be expected to be updated or read continuously at a sampling interval of 1 ms, mainly because the updates are not associated with any timestamps. However, for most use cases, a lower sampling rate is sufficient as the updates come in about every millisecond on average. Nevertheless, users sampling RAPL counters at high rates need to be aware that updates of counters can be delayed by up to 1 ms. We expect that the exact update interval and variation is highly platform-specific.

4.3. Accuracy of RAPL for the Memory Domain

This section describes how the accuracy of the RAPL memory domain can be assessed. For this, we measure the power consumption of the memory DIMMs by instrumenting the system with power sensors. As mentioned in section 3.2, mainly two instrumentation points for this have been used in the past: T. Ilsche [23] instrumented the voltage regulators (VR) on the mainboard that deliver power to the memory, while Desrochers et al. [10] used DIMM riser cards with integrated current-sense shunts. As the instrumentation at the VRs is more invasive and requires soldering cables directly onto the mainboard, we chose the second approach.

We carried our or experiments on the Ice Lake and Broadwell systems (see table 4.2) but used different memory populations by physically exchanging the modules

in the memory slots. The exact memory slot population with memory modules from table 4.1 is specified for each experiment individually.

4.3.1. Hardware Instrumentation for Reference Power Measurements of DIMMs

This section describes our experimental setup for collecting reference power measurements of DIMMs.

Power Measurements using DIMM Riser

Both DDR4 DRAM and Intel Optane Persistent Memory modules come in the same DIMM format and are plugged into the mainboard through memory slots. In order to measure the power consumption of the memory modules through a current-sense shunt (see section 2.3.3), such a shunt needs to be installed at the pins of the DIMM. However, the pins plug into the memory slot of the mainboard and are inaccessible. To solve this problem, we used DDR4 riser cards that plug into the regular memory slots of the mainboard and forward all traces to the pins of the memory module that is plugged into the riser. We chose the DDR4-rev3-L-CSR riser by Adex Electronics[2], which is depicted in figure 4.2.

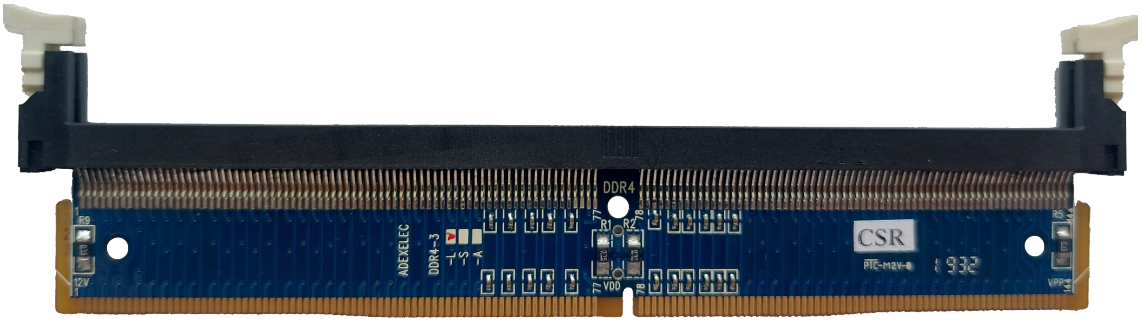


Figure 4.2.: Front of Adex DDR4 riser card - Revision 3

This riser card has 5 mΩ, 1% tolerance current-sense shunts with a power rating of 0.5 W at several pins, namely `VDD`, `VPP`, `12V`, `VPPSPD`, `VTT`, and `VREFCA`. On the `VDD` pin, two of these shunts are installed in parallel. Redundant pins are connected to the same shunt. This doubles the maximum supported power drawn from `VDD` and enables support for modern and high-capacity DDR4 modules with a significantly higher power consumption than early generations of DDR4 memory. In the work by Desrochers et al. [10], revision 2 of the riser card has been used, which only has one shunt at the VDD pin. As only `VDD`, `VPP`, and `12V` show a relevant contribution to the overall power consumption during runtime, we only consider these pins for our measurements and conduct high-side current sensing by measuring the voltage drop

[2] `https://www.adexelec.com/ddr4-3`

at the current-sense shunt, which is proportional to the current flowing through the pin (see section 2.3.3). Our Ice Lake test system automatically trains and configures the memory modules when the system boots. If the system successfully initialized the memory with exactly the same memory configuration, in particular also the identical DIMMs, during the previous boot, the initialization is partially skipped and the system boots significantly faster. When we booted the system without any riser cards and then rebooted in the same memory configuration, except that one PMem module was plugged into the memory slot using a riser, the system did not boot successfully. After a successful Baseboard Management Controller (BMC) and memory initialization according to the video output of the server, the server was stuck at the POST code `EE 00` and did not proceed. At some point, the CPU fault LED turned on and the system shut off. A possible reason for this is the increased contact and trace resistance caused by the riser card resulting in an invalid memory configuration, which is not detected by the system as the memory module and the mainboard did not change. Thus, a retraining of the memory needs to be forced whenever risers are added or removed. One option for this is to remove the module first, boot the system once, and then insert the module with the riser and boot as usual.

Current-sense Amplification To measure the voltage drop at the shunts at the `VDD`, `VPP`, and `12V` pins, we soldered cables to each side of these resistors. As this voltage can be too low for correct measurements using most analog-to-digital converters, e.g. 0.4 mV for a power draw of 1 W at `VDD`, we amplify the voltage by a factor of 100 for `VDD` and by 200 for `VPP` and `12V`. We use the *INA2180* [92] current-sense amplifier evaluation board for this. These evaluation boards have a fixed gain of 20, 50, 100, or 200, support the amplification on two channels, and have decoupling capacitors directly installed [92]. A picture of the variant with a gain of 100 is shown in figure 4.3. In our case, we powered them using the 5 V voltage line from the Raspberry Pi.

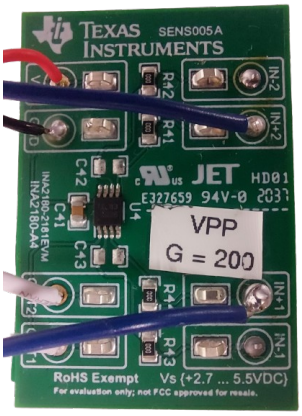


Figure 4.3.: *INA2180-A4* current-sense amplifier evaluation module used for amplifying 12 V and 2.5 V. Gain: 200

We also tried different types of instrumentation amplifiers such as the *INA122* used by Desrochers et al. [10], the *INA128* [91], and the *LT1167* used by T. Ilsche [23]. The *INA 122*, which was our first choice, did not amplify the voltage drop as expected. In a manual experiment in which we installed the *INA122* on a breadboard and generated different stable input signals using voltage divider networks, the in-amp amplified this signal as expected. However, when we used the *INA122* for measuring the `VDD` signal, the resulting amplified voltage was about 15% less than expected. As this error directly propagates to the power consumption calculation, the power consumption of the DIMM was

underestimated with the *INA122*. A possible reason for this is the high frequency of the power signal, which can be in the order of kHz [23, p. 62f]. The *INA122* has a limited bandwidth and a slew rate of only $0.08\,\frac{\mathrm{V}}{\mu\mathrm{s}}$. This causes a drop in gain at input frequencies higher than 2 kHz at a configured gain of 100, which decreases to effectively 10 at an input frequency of about 50 kHz [90, p. 4]. The higher the gain is configured, the lower the cutoff frequency at which the gain drops is [90, p. 2ff].

The *INA2180* shows significantly better results and amplifies the voltage drop as expected. A reason for this could be the significantly higher slew rate of $2\,\frac{\mathrm{V}}{\mu\mathrm{s}}$, which supports a bandwidth greater than 100 kHz for each supported gain. We confirmed this again using a voltage divider network on a breadboard, and by measuring the input voltage and the output voltage to the amplifier using a multimeter while the amplifier was connected to the current-sense resistor of the DIMM.

Analog-to-Digital Conversion This amplified signal was then measured using an MCC-128 data acquisition HAT [60], which was mounted on a Raspberry Pi 3B and used to log up to 8 analog voltage signals concurrently at an aggregated sampling rate of up to 100,000 samples per second. The internal analog-digital conversion features a 16b resolution and different input voltage ranges. We configured it to sample the voltages every 1 ms at a voltage range between 0 V and 5.0 V.

In order to measure the power drawn by a single pin of a DIMM, measuring two voltages is required: The voltage drop at the current-sense resistor (see section 2.3.3) and the voltage at the pin, which can either be measured before or after the voltage drop at the shunt. As we are dealing with relatively low voltages (1.2 V to 12 V) and higher currents (up to 11 A), we decided to conduct voltage-correct measurements by measuring the voltage before the shunt. More information on this is discussed by T. Ilsche [23, pp. 13ff]. When measuring DDR4 DRAM DIMMs, both voltage and current of the 1.2 V `VDD` and 2.5 V `VPP` pins are monitored. For PMem modules, the 1.2 V `VDD` and the 12 V supply rails are measured. `VPP` is not measured here as our measurement infrastructure only supports two channels per DIMM. We measured that the power consumption at the `VPP` pin is well below 15 mW for PMem and can be neglected. As four voltages need to be measured per DIMM, the power consumption of up to 2 DIMMs can be measured by a single MCC-128 device at a time [3].

Measurement Synchronization While the data of the physical measurements is collected by the Raspberry Pi, the RAPL measurements need to be performed by the system under test itself, introducing the challenge of synchronizing the timestamps

[3] This can be extended to up to 16 memory modules by stacking 8 MCC-128 modules on the same Raspberry Pi. One option to reduce the number of required channels would be using current-sense amplifiers with reference inputs. This way, the current of multiple rails of the same voltage can be summed up by the analog circuit, and only one signal needs to be processed digitally. In our case, this would allow us to measure the power of infinitely many DRAM and PMem modules with only six data-acquisition channels, but per-module spatial resolution will be lost. Furthermore, a sophisticated circuit design, ideally on a Printed Circuit Board (PCB), would be required.

of measurement samples between both devices. However, it is crucial to associate the power samples of RAPL with corresponding power samples collected by the Raspberry Pi. Several approaches have been used to solve this problem. For example, T. Ilsche [23] generated a distinctive pattern in the power consumption profile of the memory before and after the measurements, which is then detected in the collected power traces and used to line up both measurements. Another option is to directly synchronize the clocks of the Pi and the server under the test, for example, using the Network Time Protocol (NTP).

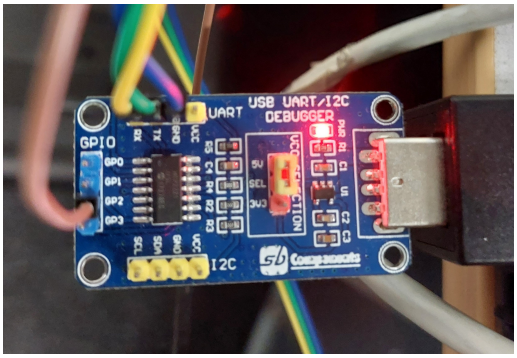


Figure 4.4.: *MCP2221* USB to I2C/UART converter with GPIO

For this work, an *MCP2221* [61] Integrated circuit (IC), which is connected to a male USB-A port and plugged into the host machine, is utilized. This IC, which is depicted in figure 4.4, features a serial UART interface, an I2C interface, and four GPIO pins. When the RAPL measurements are started on the machine under test, it outputs an 1 to one of the GPIO pins, that is connected to the digital trigger input of the MCC128 data acquisition card. This device has been configured to automatically start measurements when a rising edge[4] is detected at the trigger input with a latency of 1 µs [60]. Once the measurements end, the GPIO is set back to 0, which is then handled by the Pi to stop the measurements. The latency to stop the measurements is significantly higher compared to the start synchronization using the digital trigger pin. However, the stop of measurements does not need to be synchronized tight for our purposes.

Another challenge was that corresponding RAPL measurements and reference measurements had to be merged after the measurements were done. To facilitate this, each workload was identified by a unique ID, which was transmitted to the Pi using the UART serial interface of the *MCP2221* along with the name of the workload. This serial interface is exposed by the Linux kernel as a virtual file, for example, located in `/dev/ttyACM0`, which can be read and written from to transmit data between the Raspberry Pi and the machine under test.

Common Reference Ground between Circuits In the described measurement setup, we have two different AC power sources: The power supply that powers the Raspberry Pi and the redundant power supplies of the machine under test. Although all power supplies are connected in the same building, the ground potential of both DC circuits might differ significantly, usually between 10 mV to 200 mV [66]. When measuring single-ended analog voltages, e.g., voltages with respect to ground, of two

[4]rising edge: Digital signal switches from 0 to 1

different devices, this can severely impact the quality of the measurements. This could result in a significant offset between the actual voltage and measured voltage. Thus, the analog ground input of the MCC128 HAT is connected to the ground pin of the *MCP2221* USB device to establish a common reference point for the measurements.

Engineering Aspects of Measurement Setup In our first prototypes, we used a breadboard to connect all components, similar to Desrochers et al. [10] did. However, the breadboard was not making reliable connections in our case and made the circuit unstable and very sensitive to external influences, in particular, the high airflow produced by nearby fans of the server. Thus, we soldered all connections whenever possible to maintain reliable and low-impedance connections. To maintain the replaceability of the components, we also added screw terminals to the current-sense amplification boards and created a custom perf-board that provides power and ground to the different components and handles the digital trigger signal.

Special care must be paid to the wires transferring the voltage drop signal of the current sense-shunts and the bus voltages. Ideally, these connections would be realized using short PCB traces, which was infeasible for our setup. The voltage drop at the probing wires, which we realized through kelvin connections[5], is expected to be very low due to a high input impedance of the amplifiers and a low bias current. Still, the pairs of wires used for probing the voltage drop at the shunts should ideally be the same length, as short as possible, and low-impedant to ensure correct current measurements. In our case, we were using solid, tinned, 20 AWG[6] copper wires with approximately the same length for each pair of probes.

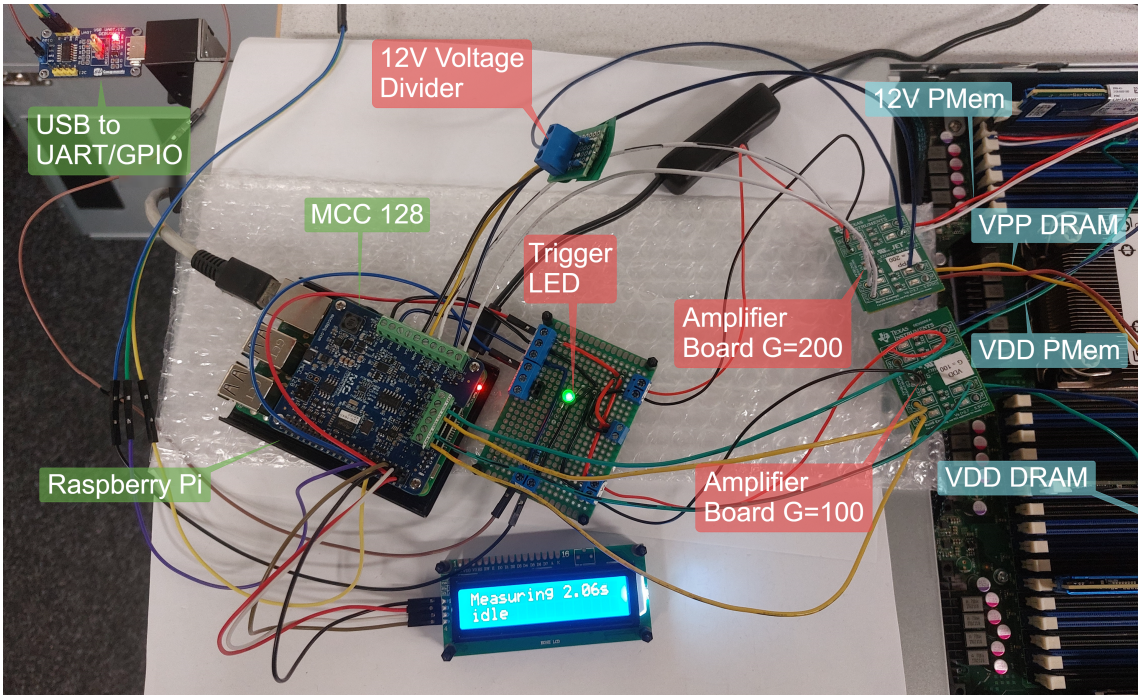
Because the 12 V bus voltage used for powering PMem exceeds the 10 V input limit of the MCC128 device, we use a custom voltage divider realized using an 815 Ω and a 2680 Ω resistor, each with 0.1% tolerance to regulate a nominal 12 V power line down to about 2.8 V, which is well within the configured 5 V input range of the DAQ device[7].

A circuit schematic of all components and connections can be found in figure B.5 in the appendix. Furthermore, a guide on reproducing these experiments and how the setup can be modified for measuring the power consumption of DDR5 memory can be found in appendix B.2. Figure 4.5 shows annotated photos of the data processing and voltage amplification as well as the instrumentation of a DDR4 and a PMem module on the Ice Lake system. Additional photos of the hardware setup are shown in figure B.1 for the Ice Lake system and figure B.4 for the Broadwell system in the appendix.

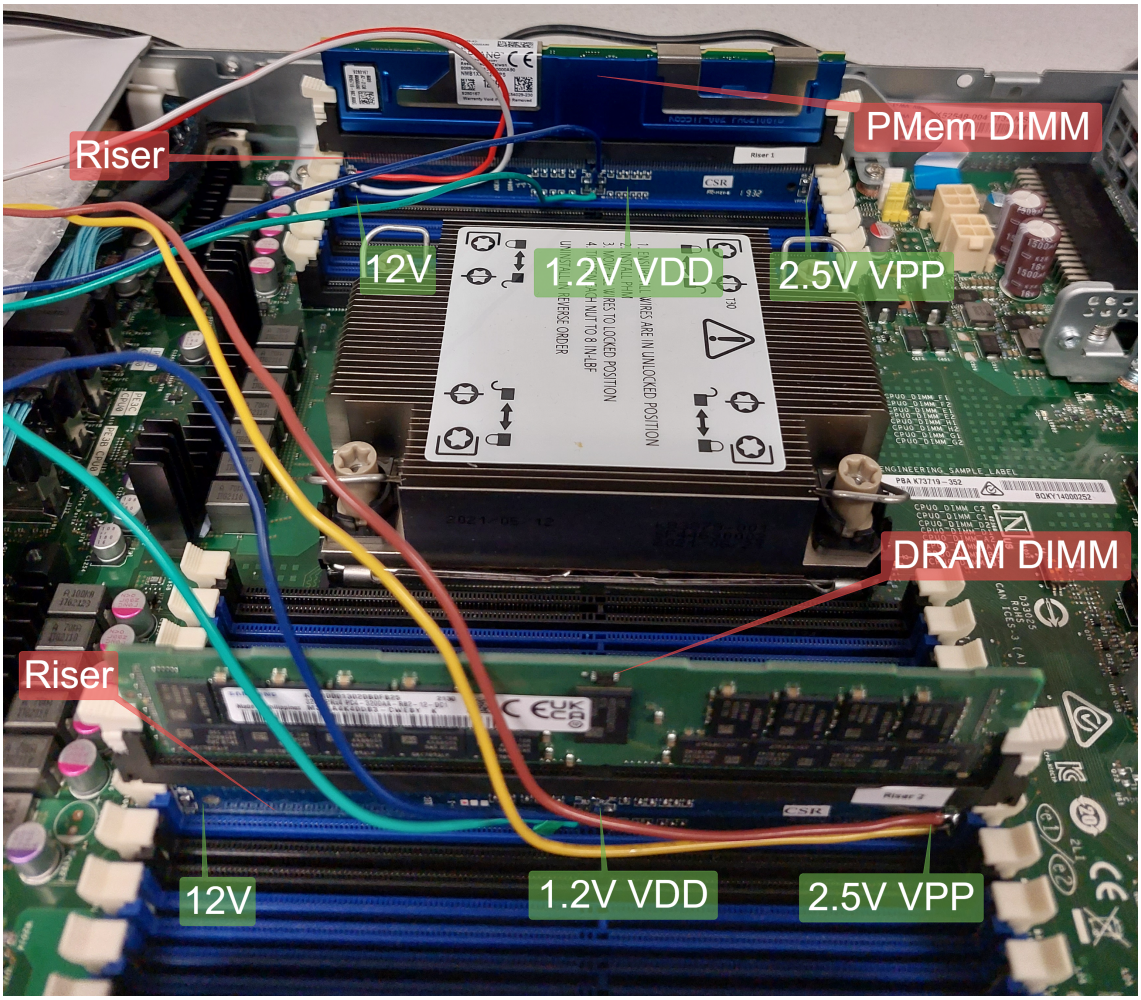
[5]In Kelvin sensing, also called four-terminal sensing, an additional pair of probes is used for measuring the voltage drop at the shunt. The current in these probes is very low, so the voltage drop in the probes is usually neglectable.

[6]0.81 mm inner cable diameter

[7]The input range can be configured to a lower limit than 10 V if all input voltages are lower.



(a) Data processing setup



(b) Memory modules and risers

Figure 4.5.: DIMM hardware instrumentation on the Ice Lake system

RAPL Measurements To validate the accuracy of RAPL measurements, we need to sample the RAPL energy counter for the memory domain regularly and compare it to the reference power measurements with the riser card. Because RAPL provides an energy counter rather than instantaneous power consumption, we need to divide the energy consumption of the sampling interval by the length of that interval to get the average power consumption.

As described in section 2.3.2, there are several high-level tools and libraries to access RAPL counters. Similar to Desrochers et al. [10], we first used the `perf stat` command, which is part of the Linux kernel tools, to sample RAPL counters regularly. We modified the tool to output an 1 to a GPIO pin of the *MCP2221* when the measurements start and set it back to 0 when measurements are completed for synchronization purposes. This approach faced two problems: First, `perf` was not able to sample RAPL counters at precise time intervals at sampling rates greater than 100 Hz. Second, the delay between setting the GPIO to 1 and starting the actual RAPL sampling was too high, resulting in a noticeable shift of about 150 ms between the RAPL measurements and reference measurements.

To counter both problems, we decided to use a low-level approach and wrote a tool that directly reads the MSR values for RAPL. This tool is based on a popular C script by V. Weaver [99], which we extended with support for Intel Ice Lake and regular sampling of RAPL counters. Furthermore, our tool takes a bash command as an input and executes it in a subprocess. When the subprocess terminates, the sampling of the RAPL counters is stopped and the results are written to a file. As the exact timing of RAPL updates cannot be predicted [41] (see section 4.2), we use a sampling interval of 5 ms as a trade-off between unpredictable timings and low sampling rate.

Data Post-processing The previously described measurement method produced raw measurement data for the reference measurements and the sampled RAPL counter values, which had to be further processed to compare the power consumption measured by reference measurements and RAPL.

For the reference measurements, we have a CSV file with the voltages measured by the MCC128 data acquisition HAT at each measurement clock tick, namely the amplified voltage drop ΔU^* at the shunt and the supply voltage U for the measured line, e.g., `VDD`. We divide ΔU^* by the gain used for the respective pin, e.g. $G = 100$ for `VDD`, to get the unamplified voltage drop ΔU. Then, we can compute the current at the pin using $I = \frac{\Delta U}{R}$ with R being the known resistance of the shunt. For `VPP` and `12V`, we can use the value $R = 5\,\text{m}\Omega$ but for `VDD`, we need to consider that we have two $5\,\text{m}\Omega$ shunts in parallel. The resistance of both shunts in parallel can be computed as follows: $R = (\frac{1}{5\,\text{m}\Omega} + \frac{1}{5\,\text{m}\Omega})^{-1} = 2.5\,\text{m}\Omega$ [77, p. 35f]. Power can then be computed using $P = U \times I$.

The RAPL measurements are also written to a CSV file. Each sample contains the energy consumption since the previous one and the time elapsed since the start of the measurements in seconds as a floating-point value. By dividing the energy

consumption by the delay to the previous sample, the average power during this interval can be computed.

The different sampling rates for the reference measurements (1 ms) and the RAPL measurements (5 ms) require resampling both measurements to a common interval in order to compare values of both measurements at the same time. In our case, whenever required, we resampled the data to 200 ms intervals by averaging all samples within this interval.

4.3.2. Experimental Setup Validation

This section describes the validation of the experimental setup with respect to power limits, overhead, and accuracy.

Power Limits of the Riser Cards The current-sense shunts on the riser card have a very low resistance of 5 mΩ to limit the impact on the system. Still, the shunts cause a voltage drop at each pin, which can be computed using $\Delta U = I \times R$ and should not bring the voltage below the minimum operating voltage of the memory. In addition, the power dissipated at the resistor must not exceed its power rating, which is 0.5 W in this case, and should also leave some tolerances. This power dissipation can be computed using $P = U \times I = I^2 \times R$. Table 4.3 shows the voltage drop and the power dissipation at the highest expected current specified by the JEDEC standard for RDIMMs [36].

Pin	Typical Voltage	Shunt Resistance	Max Current	Min Voltage	Max Voltage Drop	Max Power Dissipation	Shunt Rating
VDD	1.21 V	2.5 mΩ	11.7 A	1.16 V	0.029 V	0.3422 W	0.5 W
VPP	2.50 V	5 mΩ	3.75 A	2.41 V	0.019 V	0.0703 W	0.5 W
12V	12.00 V	5 mΩ	1.17 A	10.2 V	0.005 V	0.0068 W	0.5 W

Table 4.3.: Voltage drop and power dissipation at shunts on DDR4 riser card

As the voltage after the drop is above the minimum expected voltage and the maximum power dissipation is below the rating of the resistor, it is safe to operate the riser with DDR4 RDIMMs. However, it needs to be ensured that the mainboard provides the typical voltage, especially for the `VDD` rail because a voltage close to the minimum supported voltage provided by the mainboard can result in an undervoltage supply with the riser card. The PMem modules can be safely operated as they are primarily powered using the `12V` rail, which offers a large margin for both voltage drop and power dissipation as the current is 10 times lower compared to a 1.2 V supply.

DIMM Riser Overhead The DDR4 riser cards increase the length of the traces to the memory and also introduce an additional contact resistance. In addition,

the case of the server needs to stay open when the risers are installed due to the increase in height. This leads to poor airflow in the system. Both could result in degraded performance for the memory access, which we evaluated by running the Intel Memory Latency Checker (MLC) [29] with no risers installed in a closed case and then compared the benchmark results to the results obtained with one riser installed on both sockets and the rear side of the top case cover open. Two memory configurations were considered: 1x32GB Samsung DDR4 on both sockets and a riser installed on socket 0, as well as 1x128GB Optane PMem + 1x32GB DRAM per socket and the PMem module on socket 0 being mounted using a riser. The results of both runs are shown in table 4.4.

	(a) DRAM		(b) PMem	
	No Riser Case closed	**With Riser Case open**	**No Riser Case closed**	**With Riser Case open**
Local Latency	61.5 ± 0.1 ns	61.4 ± 0.5 ns	183.8 ± 0.1 ns	183.4 ± 0.3 ns
Remote Latency	120.3 ± 0.1 ns	120.4 ± 0.3 ns	252.3 ± 1.3 ns	252.0 ± 1.2 ns

Table 4.4.: Memory access latency comparison between the operation with and without risers used for DRAM and PMem. Mean of 5 measurements with standard deviation.

As shown in table 4.4, the risers and the opened case do not significantly impact the memory performance. The minor differences between the results are well within statistical variations and can be ignored for our experiments in accordance with the results by Desrochers et al. [10].

Error Margin Analysis To correctly interpret the results of the physical power measurements, we need to conduct an error margin analysis. Three primary error sources can affect the power measurements here: The tolerance of the shunt at the riser card, the current-sense amplifier, and the MCC128 data acquisition device. Texas Instruments provides an Excel tool to compute an upper error bound for their current-sensing products [89], which we utilized to get an error estimation for the `12V` and `VDD` pins for the cs-amps *INA2180* and *INA296*.

As shown in figure 4.6 the current-sensing is subject to an increasing total error at low current for the *INA2180*. In our case, the minimal expected current is about 1.5 A for the 1.2 V line and 0.3 A for the 12 V line, at which the theoretical maximum error is about 30% for both rails. The *INA296*, which is represented by the dashed lines, shows a significantly lower error here and converges to the tolerance of the shunt at a load current of less than 0.25 A and should be preferred over the *INA2180* in terms of accuracy. Unfortunately, the *INA296* showed some problems in practice, mainly caused by a constant offset in output voltage - possibly caused by a wrong reference voltage. When we were able to get reliable results with the *INA296* at

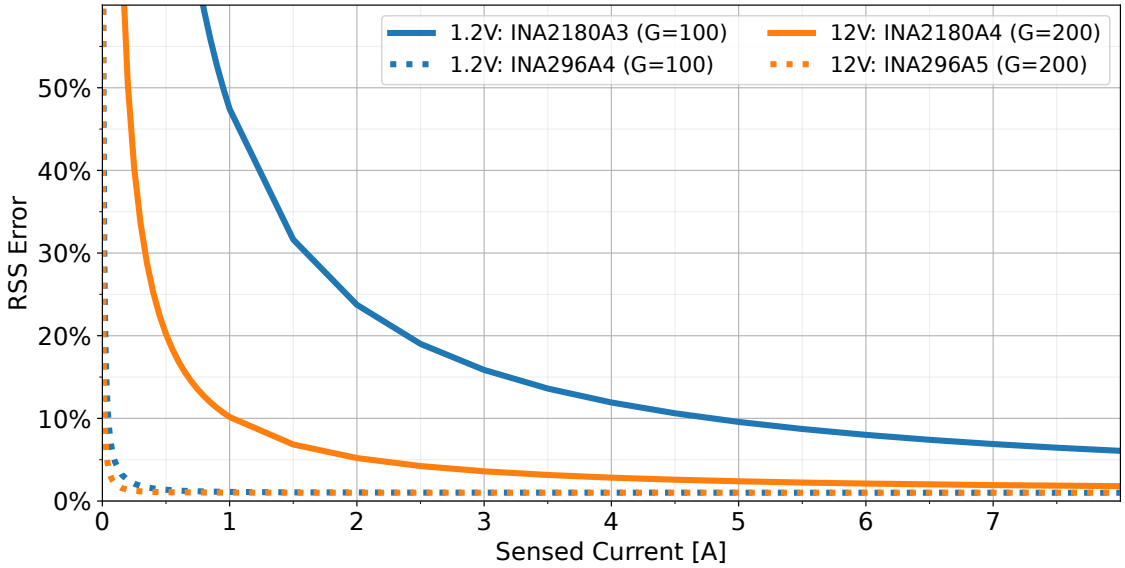


Figure 4.6.: Theoretical residual sum of squares (RSS) error of *INA2180* vs. *INA296* in high-side current sensing at 25°C. $V_s = 5.1V$ Shunt Tolerance: 1%. Comparison between $V_{CM} = 12V$, $R_{shunt} = 5m\Omega$, Gain = 200 and $V_{CM} = 1.2V$, $R_{shunt} = 2.5m\Omega$ Gain = 100

some point, only a minor difference to the measurements with the other amplifier was observed. This indicates that the error model is more of a theoretical nature and that the actual error is lower in practice. As we were not able to get consistent and reliable results with the *INA296*, we decided to use the *INA2180* regardless of the error. The *INA2180* also supports amplifying two channels at once, while the other amplifier only has single-channel support. This allows measuring two DIMMs at the same time.

In addition to the sensing error, which includes errors caused by amplification and the shunt itself, a maximum absolute error of 3 mV is introduced by the analog-digital conversion at the data acquisition device at 25 °C room temperature [60]. If we assume a current of 2 A at the CSR for `VDD`, a voltage drop of 5 mV, which is amplified to 1 V, is measured by the DAQ device. This results in a relative error in the A/D conversion of 0.3%. The voltage divider for the `12V` line uses resistors with 0.1% tolerance and introduces an error of at most 0.154% to the power measurements.

4.3.3. Evaluation Experiments

For validating RAPL, we need to compare the RAPL measurements with reference power measurements for different workloads. These workloads can be classified into three types: idle workloads, CPU-intensive workloads, and memory-intensive workloads. As we are only interested in the energy consumption of the heterogeneous

memory domain, the focus lies on the memory-centric workloads. A collection of kernels is given in table 4.5.

Workload	Boundness	Threads (DRAM)	Threads (PMem)
Stream (All)	Memory	32	4
Stream (Copy)	Memory	1,4,8,16,24,32	1,2,4
Stream (Scale)	Memory	32	4
Stream (Add)	Memory	32	4
Stream (Triad)	Memory	32	4
Mem Write	Memory	32	4
Mem Copy	Memory	32	4
Dot Product	Memory	1	1
SSE2 ADD	Compute	1	1
SSE2 MUL	Compute	1	1
DGEMM Mat. Mult.	Compute	1	1
Sleep	Idle	1	1
Busy wait	Idle	1	1

Table 4.5.: Kernels used for RAPL validation

For idle measurements, the `sleep()` system call and busy waiting were used. Some CPU-intensive workloads were extracted from the *roco2*[8] workload-generation tool, which was used to validate the accuracy of node-level RAPL measurements by Schöne et al. [81]. In addition, the four separate kernels from the STREAM benchmark with different numbers of threads were used as memory-intensive workloads. The different thread count is essential as the power consumption of the memory will increase with more threads. As we only consider configurations with a single PMem DIMM, we limited the number of threads concurrently accessing PMem to 4. For higher thread counts, the performance and bandwidth would drop significantly.

We performed these experiments on the Ice Lake and Broadwell systems as described in table 4.2. OpenMP threads are pinned to the CPU cores for all experiments using `OMP_PLACES=cores` and `OMP_PROC_BIND=true`. Using the `numactl -m 0 -N 0` command, the threads are placed on socket 0, and all allocations are forced on the DRAM of socket 0. For allocations on PMem, `numactl -m 2 -N 0` is used to allocate memory on the local PMem memory attached to socket 0. The `GCC 8.5.0` compiler with default flags is used for compiling the binaries.

[8] https://github.com/tud-zih-energy/roco2/

4.3.4. Experimental Results

The experimental instrumentation setup was used for power measurements of up to two DIMMs and the comparison to RAPL. DRAM-only configurations on Intel Ice Lake and Broadwell systems and configurations with both PMem and DRAM on the Ice Lake system were considered.

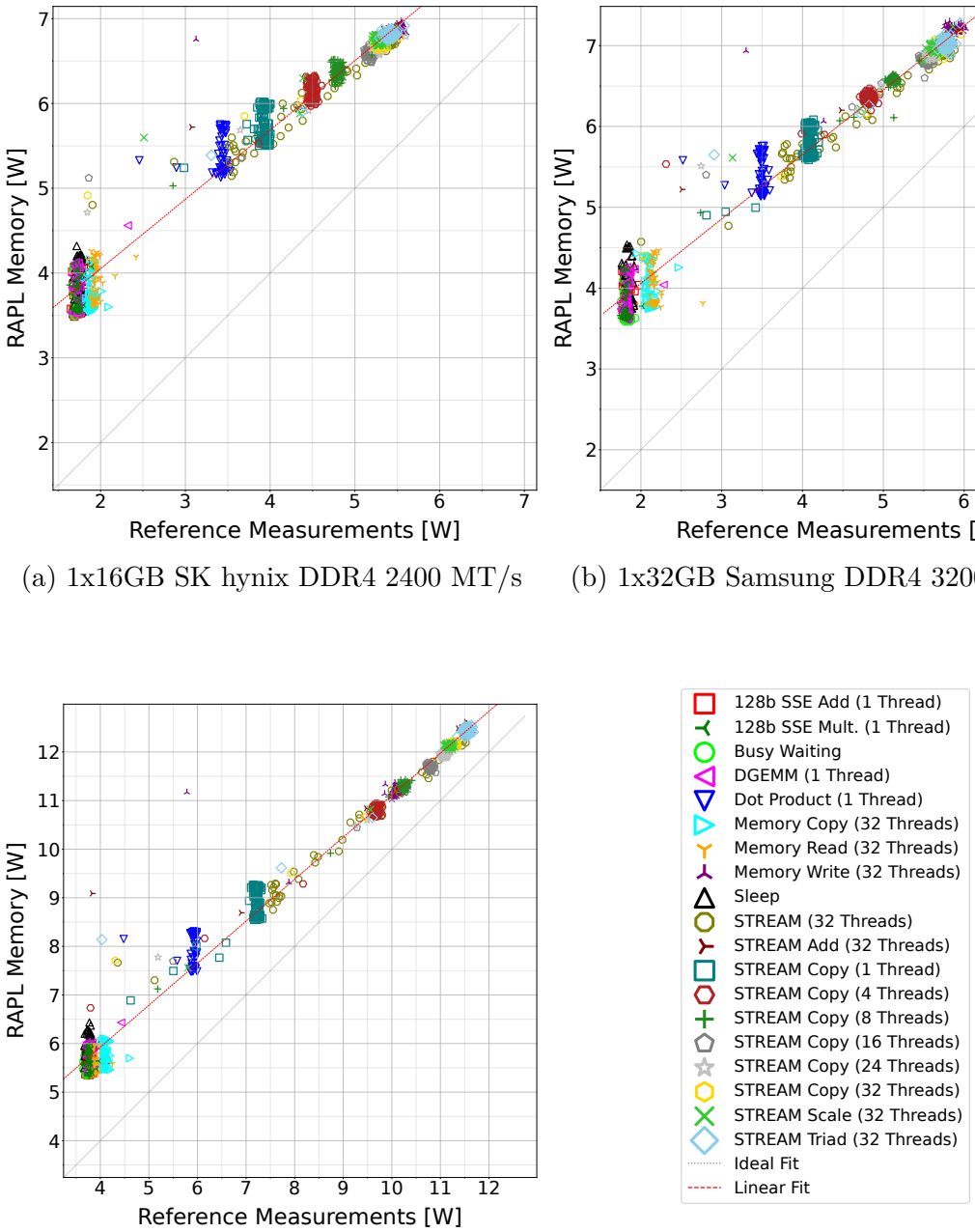
Intel Ice Lake-SP

Figure 4.7 shows the correlation between RAPL and the reference measurements for the workloads described in table 4.5 with different memory module populations. The graphs have been created by resampling both the RAPL and reference measurement samples to a common interval of 200 ms by averaging all measurements in this interval. At most 100 samples, each corresponding to a 200 ms interval, are shown for each workload to aid visualization. Figure 4.7a shows results with a single 16GB DRAM module per socket, figure 4.7b with a single 32GB DRAM module per socket, and figure 4.7c for 2x32GB DRAM modules per socket. The power consumption of modules installed in socket 0 is measured with the risers and compared to RAPL measurements of socket 0.

Results from figure 4.7 show a significant offset between RAPL measurements of the memory domain and our reference measurements consistently throughout all memory populations we investigated. For the population with a single 32GB Samsung DDR4 module (figure 4.7b), RAPL reports almost twice the power reported by the reference measurements when the memory is idle or under light load. At higher load, this gap slightly reduces and the relative error decreases.

In addition, an increased variation in power consumption can be observed for RAPL measurements. For example, RAPL reports between 3.5 W and 4.3 W for idle workloads, while our reference measurements report a power consumption between 1.6 W and 1.9 W. This gap decreases at higher power consumption. In general, a monotonicity in the correlation between RAPL and reference measurements can be observed: If reference power measurements increase significantly, the power consumption reported by RAPL increases significantly, too. As indicated by the red curve, the relationship between RAPL and reference measurements can be roughly approximated linearly for all three memory populations. The absolute offset between the measurements decreases at higher power consumption, and, in particular, with higher memory capacities installed. An evaluation of higher memory capacities would be of interest here but is limited by our measurement setup, which only supports measuring two DIMMs at once[9].

[9]We also ran the measurements with 4, 6, and 8 DRAM DIMMs per socket and measured the power consumption of only two using risers. We assumed that the power consumption of all DIMMs of the same model is identical under the same workload and computed an estimation for all memory modules based on measurements of the two DIMMs accordingly. However, under this assumption, RAPL would underestimate the reference power measurements. As the power consumption between the two measured DIMMs differed significantly under load, we concluded that this assumption is invalid. That is in line with previous work [17].



(a) 1x16GB SK hynix DDR4 2400 MT/s (b) 1x32GB Samsung DDR4 3200 MT/s

(c) 2x32GB Samsung DDR4 3200 MT/s

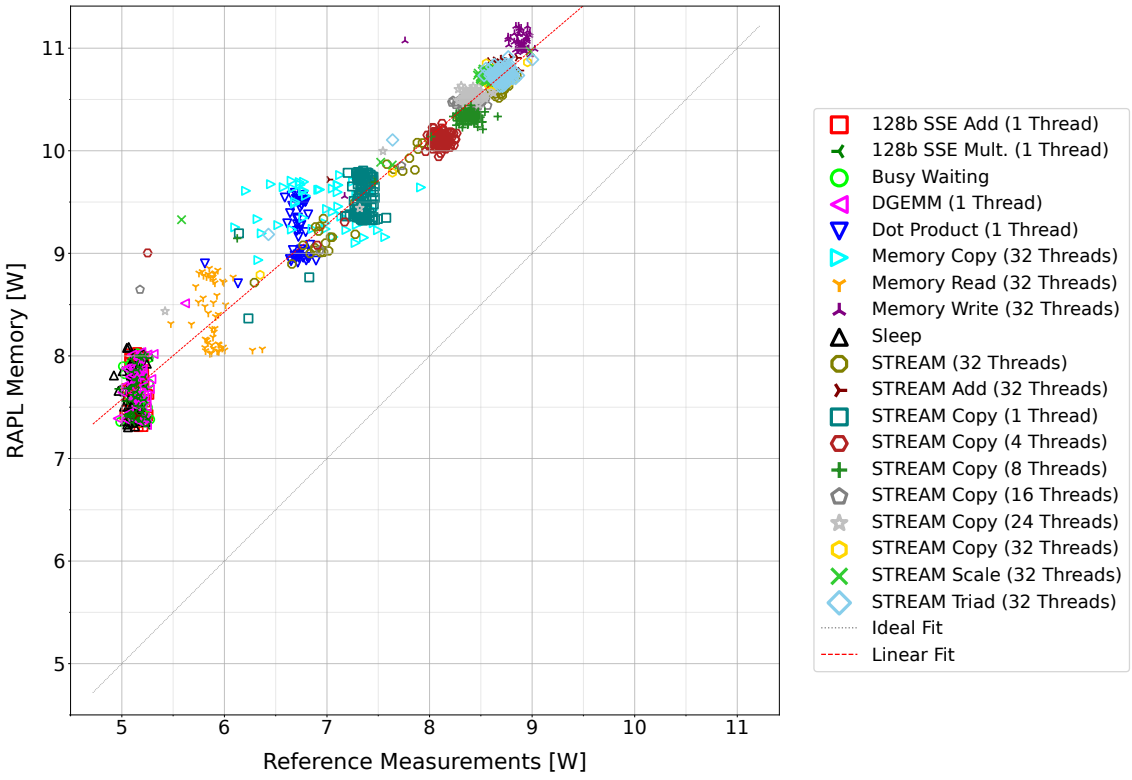
Figure 4.7.: RAPL memory measurements vs. reference on Intel Ice Lake system: DRAM only

Results for experiments with 1x32GB DDR4 and 1x128GB PMem per socket are shown in figure 4.8 with the power of both the DRAM and the PMem modules of socket 0 measured using riser cards. For figure 4.8a, allocations were made on DRAM, while in figure 4.8b on local PMem. When allocating on DRAM, the correlation between RAPL and reference measurements is similar to when no PMem is installed (see figure 4.7). However, an overall higher power consumption is reported

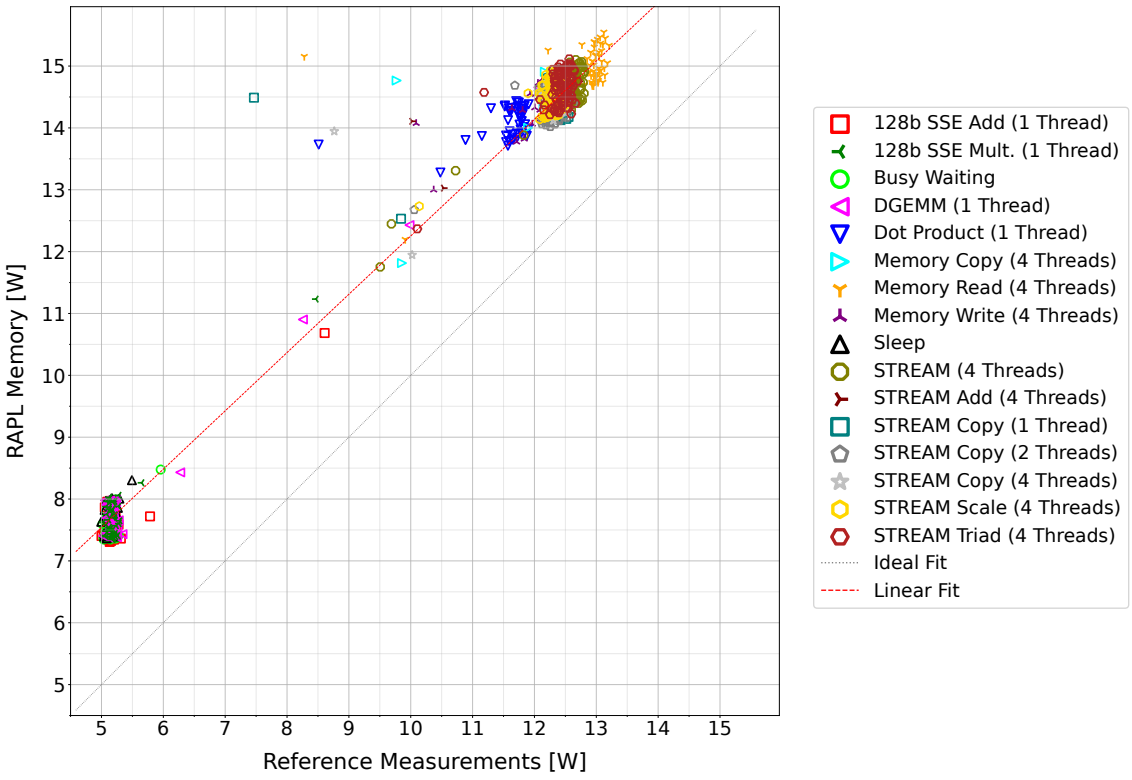
and RAPL measurements are again spread wider at low load on memory. When allocating on PMem, the overall variance in power consumption is lower. Workloads with little load on memory consume about 5.2 W, the Dot Product kernel consumes 11.8 W, and the Memory Read workload consumes the highest power of about 13 W. All other workloads consume approximately 12.5 ± 0.3 W. Figure 4.9 shows RAPL and reference measurements for socket 0 on the Ice Lake platform populated with 1x32GB DRAM when idle and during execution of the STREAM triad kernel with 32 threads. When idle, the reference measurements report a power between 1.6 W and 1.9 W, while the power reported by RAPL is more than two times higher and varies between 3.5 W and 5 W. In addition, RAPL reports peaks in the power consumption, e.g., between 1300 ms to 2600 ms, which are not reflected in reference measurements. Overall, no strong correlation between small changes in the reference measurements and RAPL measurements can be observed. During the execution of the STREAM triad kernel with 32 threads, reference measurements report about 5.75 W, while RAPL reports around 7 W. Here, large spikes in power consumption are reflected in both the reference and RAPL measurements, for example, during the first 300 ms. Figure 4.10 shows the RAPL measurements and reference measurements for socket 0 on the Ice Lake platform with 1x32GB DRAM and 1x128GB PMem. Allocations are done on PMem and results are shown for when idle and during the execution of the STREAM triad kernel with 4 threads. The green, dotted curve is the sum of the power consumption of DRAM (shown in blue) and PMem (shown in orange) and should be compared to the purple curve that represents corresponding RAPL measurements. For the DRAM, the power consumption is similar at idle compared to when PMem is under load. At idle, the DRAM consumes about 1.8 W while the PMem module consumes 3.4 W. When the STREAM triad benchmark is executed with 4 threads, RAPL reports about 2.0 W to 3.0 W more than what reference measurements report. In addition, the RAPL measurements are again subject to temporal variations.

Figure 4.11 compares RAPL with reference measurements on both sockets of the Ice Lake system when idle. Each socket was populated with a single 32GB DDR4 module installed through a riser for power measurements. The reference measurements report about 1.8 W for the DRAM modules on each socket, with a slightly higher temporal variation on socket 0. For socket 0, RAPL reports between 3.7 W and 4.5 W for the memory domain, while, in contrast, for socket 1 RAPL reports between 5.25 W and 5.7 W. While RAPL diverges, reference measurements are overall close between both sockets. This is reinforced on the same system with full DIMM population (see table 4.2) for which RAPL reports 35.37 W for the memory domain of socket 0 and 41.93 W for socket 1 when idle.[10].

[10]Measured repeatedly using `perf stat -e power/energy-ram/ -I10000 -per-socket`, i.e., averaging over 10 seconds for each socket.

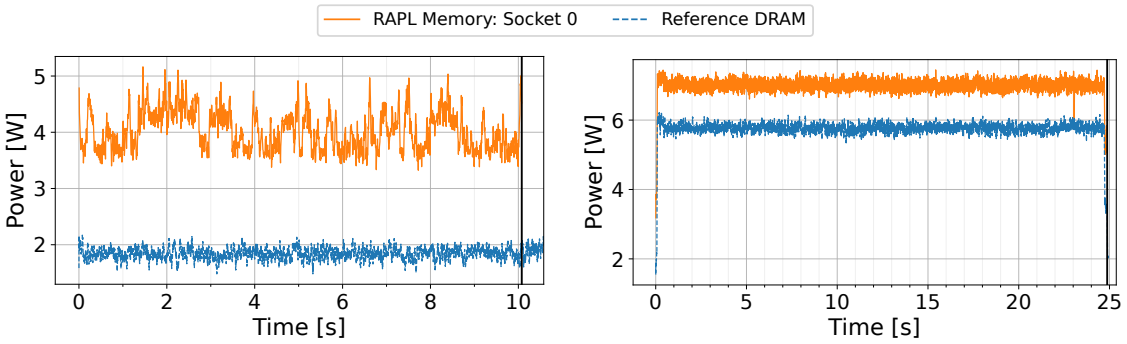


(a) Allocation on DRAM



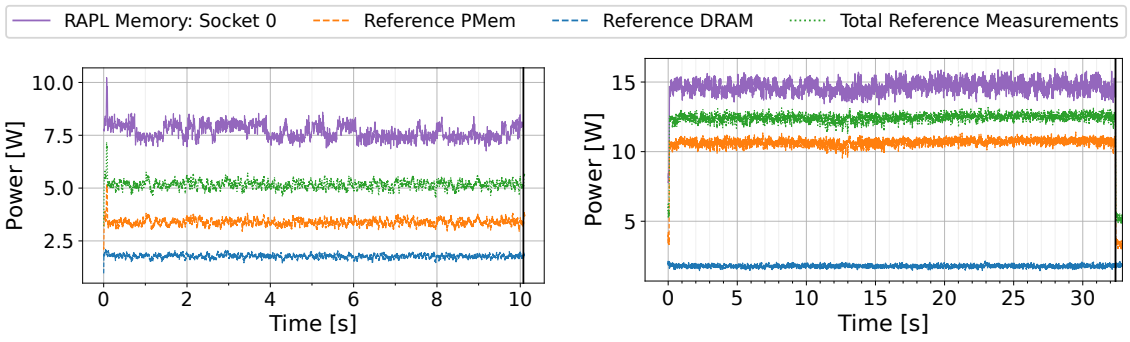
(b) Allocation on PMem (AppDirect)

Figure 4.8.: RAPL memory measurements vs. reference on Intel Icel Lake system



(a) Idle (sleep) (b) STREAM Triad (32 Threads)

Figure 4.9.: RAPL memory measurements vs. reference on Intel Ice Lake system over time: 1x32 GB DDR4



(a) Idle (sleep) (b) STREAM Triad (4 Threads)

Figure 4.10.: RAPL memory measurements vs. reference on Intel Ice Lake system over time: 1x32 GB DDR4 + 1x128GB PMem in AppDirect Mode. PMem memory allocation

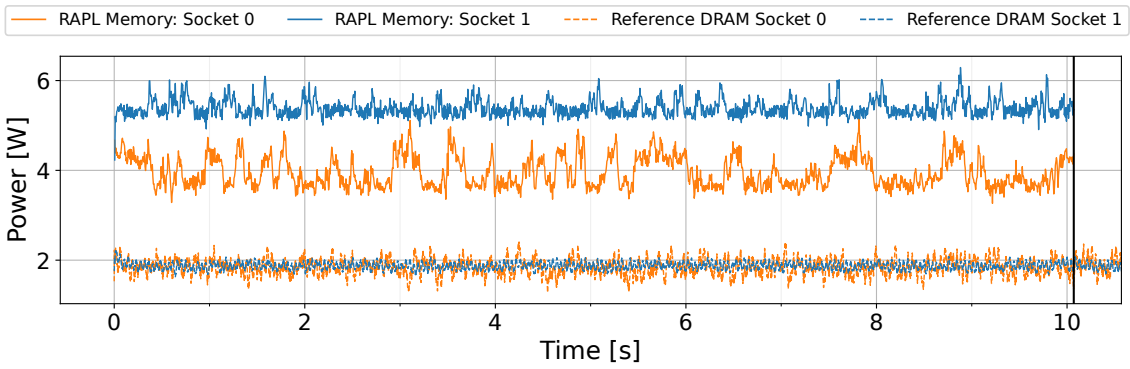


Figure 4.11.: RAPL vs. reference power measurements on the Intel Ice Lake system with 1x32GB DDR4 DRAM on each socket and reference power measurements at both DIMMs at idle state

Intel Broadwell-EP

We conducted the same experiments as on the Ice Lake system also on the Intel Broadwell-EP system (see table 4.2) in order to verify the accuracy of our reference measurements with the results from a similar platform from the literature [10]. Due to the position and orientation of the memory slots on the mainboard and the limited space available in the case, only one riser card could be used per socket. In our case, the `D1` slot of socket 0 and the `F1` slot of socket 1 have been populated with a riser card for power measurements and 16GB of SK hynix DDR4 memory. All other slots remain unpopulated. The correlation between RAPL measurements and reference measurements on socket 0 when allocating on socket 0 is shown in figure 4.12. When

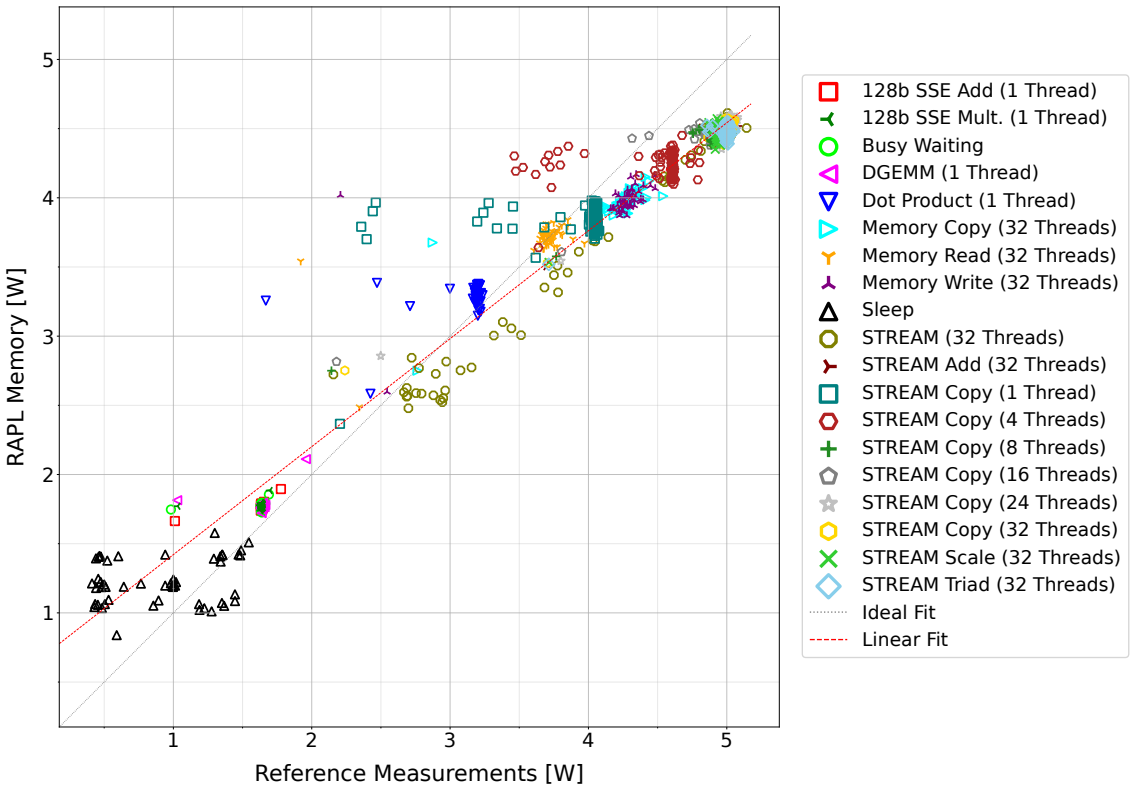
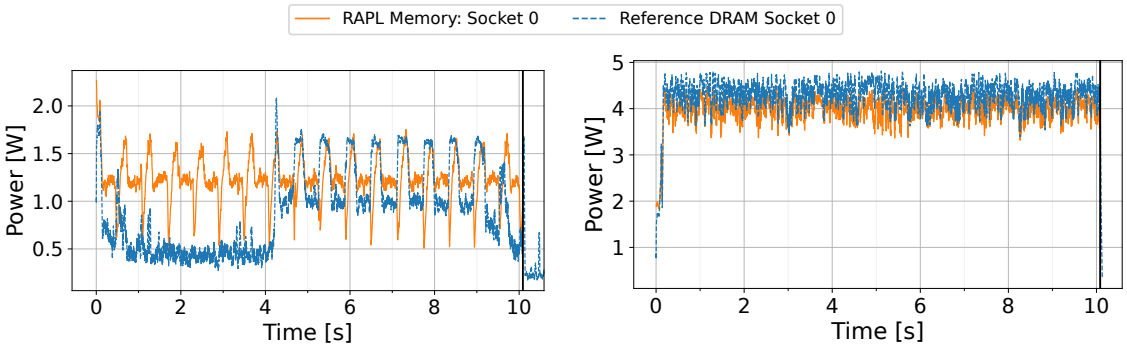


Figure 4.12.: RAPL memory measurements vs. reference on socket 0 of Intel Broadwell system: 1x16GB DDR4.

the system was idle, the reference measurements differed significantly and inconsistently from what RAPL reported. For workloads with low load on the memory, such as Busy Waiting, DGEMM, the Add Kernel, and the Multiplication kernel, only minor variations in the power consumption were observed. Here, the reference measurements report about 1.6 W, while RAPL reports 1.7 W. For the dot product kernel, both RAPL and reference measurements reported about 3.3 W with minor variations very similar to the Memory Read kernel, for which both RAPL and reference measurements reported about 3.7 W. For workloads that consumed more than

4 W, RAPL reported slightly less power and a wider spread of RAPL measurements could be observed while reference measurements were more consistent. The single-threaded STREAM copy, for example, consumed about 4.1 W according to reference measurements, while RAPL reported a varying power consumption between 3.7 W and 4 W. In addition, some outliers for which RAPL reported significantly more power than reference measurements could be observed for the STREAM copy kernel with up to 24 threads.

Figure 4.13 shows corresponding RAPL and reference measurements for socket 0 over time during idle and during the execution of Memory Copy with 32 threads. When idle, RAPL reports a consistent, repetitive pattern of a negative spike to 0.6 W



(a) Idle (sleep)

(b) Memory Copy (32 Threads)

Figure 4.13.: RAPL memory measurements vs. reference on Intel Broadwell system over time. 1x16GB DDR4 per socket. Measurements on socket 0.

followed by a positive spike up to 1.6 W and a roughly constant power consumption of 1.25 W for 300 ms–400 ms. The reference measurements match this pattern partially after 4400 ms. In particular, the power consumption peaks are present in RAPL and reference measurements and align. In addition, reference measurements report about 0.2 W less during the constant power phase.

When executing the Memory Copy kernel with 32 threads, RAPL and reference measurements both report a power consumption of around 4.1 W with spikes between 3.5 W and 4.6 W[11]. These spikes line up between both measurement methods.

To better compare the RAPL error between the Ice Lake and the Broadwell-EP platform across the different workloads, the total energy-to-solution has been computed by integrating the power samples over time[12]. For the Ice Lake platform, three different DRAM-only memory populations have been considered. The relative energy difference between RAPL and reference measurements is shown in table 4.6. A positive value indicates that RAPL overestimates the actual energy consumption, while a negative value indicates underestimation.

[11]For other workloads that put stress on the memory, a slightly larger gap between RAPL and reference measurements can be observed (see figure 4.12).

[12]The energy-to-solution includes both the average power and the execution time. This way, we can compare both systems quantitatively, although the execution time might differ.

	Ice Lake			Broadwell
	2x32GB DDR4	**32GB DDR4**	**16GB DDR4**	**16GB DDR4**
Busy Wait	50%	107%	118%	8%
Sleep	58%	118%	122%	35%
Mat. Mult.	50%	112%	119%	8%
Dot Product	33%	55%	58%	4%
Memory Read	45%	87%	102%	0%
SSE 128 ADD	46%	113%	119%	8%
SSE 128 MUL	49%	109%	116%	8%
Stream Copy (1 Thr.)	23%	43%	46%	-2%
Stream Copy (8 Thr.)	10%	29%	33%	-9%
Stream Copy (32 Thr.)	8%	23%	27%	-10%

Table 4.6.: RAPL error comparison between the Ice Lake and Broadwell systems

As shown in table 4.6, the relative error of RAPL decreases with increasing memory capacity. For example, the error during sleep with 2x32GB DDR4 installed is 58% compared to 122% with only 16GB of memory installed. At higher load, this error decreases to less than 10% with 2x32GB memory and less than 20% with 16GB of memory. Nevertheless, RAPL overshoots in every case on the Ice Lake system. On the Broadwell-EP system, the RAPL error is significantly lower and less consistent: The error can reach up to 35% at idle, while RAPL overestimates by up to 10% at low load and underestimates by up to 10% at high load. An absolute error of less than 2% can be observed for a medium load on the memory, for example, single-threaded, memory-bound workloads.

4.3.5. RAPL Error Analysis

The results for the RAPL validation show that RAPL reports a significantly higher power consumption for the memory domain than physical measurements at the DIMM on the Intel Ice Lake-SP platform both for DRAM and PMem in all tested memory populations. The absolute difference decreases slightly at higher current demand, and as a result, the relative offset decreases to about 10%. In contrast, RAPL matches the reference measurements significantly better on Broadwell-EP, which is similar to the Haswell-EP architecture, which was evaluated by Desrochers et al. [10]. Additionally, we have shown that RAPL measurements are inconsistent between different sockets of the Ice Lake-SP system. As shown in figure 4.11, the reference measurements at the DIMM report roughly the same power consumption for both sockets while the corresponding measurements with RAPL differ significantly between both sockets. In contrast, RAPL memory measurements are consistent between both sockets on the Broadwell-EP system.

As indicated in section 4.3.2, our reference measurements cannot be considered as a ground truth as they can be subject to a significant theoretical error as high as 30 % when the load on the memory is low. However, for the population with a single 16GB SK hynix DDR4 module (see figure 4.7a) our reference measurements report between 1.6 W and 1.8 W when idle. This matches the 1.7 W *active standby* power consumption according to the DIMM's datasheet [87, p.74]. Thus, our measurements are in the correct order of magnitude and match the expected power consumption significantly better compared to RAPL. For the other memory modules, such datasheets are not available publicly. Furthermore, different current-sense amplifiers with significantly lower theoretical errors gave us very similar results, indicating that this high error margin from section 4.3.2 is more of a theoretical nature.

Our results only cover configurations with up to two DIMMs populated per socket, which is well below the maximum number of 16 DIMMs supported per socket on our Ice Lake system. At least one DIMM for each memory channel, i.e., eight DIMMs per socket in this case, would be considered relevant for use in production. In particular, our memory configuration is synthetic and would not be used in production. While our approach could be easily extended to measure 16 DIMMs at the same time, it would require 16 DDR4 riser cards and 8 MCC-128 data acquisition devices. In addition, the extended footprint of the riser cards and the attached wires prevents us from placing two risers in slots next to each other.

As extending our measurement infrastructure in a way that we could measure all DIMMs in a real-world memory population of the server is infeasible, we discuss possible reasons for the large gap between RAPL and reference measurements for single and dual DIMM population. We aim to reverse engineer how RAPL works internally and explain how this knowledge can be applied to a system with full memory population.

Possible Reasons for RAPL Inaccuracy

As discussed previously, RAPL has been shown to significantly overestimate the power consumption of memory in the tested configurations on the Ice Lake-SP system. In contrast, the results with the same measurement setup on the Broadwell-EP system are closer to RAPL and match the observations by Desrochers et al. [10] closely. To explain these differences, several possible reasons that can explain the variations have been identified and will be explained in the following.

Intel SGX/Energy Filtering In previous work, the RAPL energy readouts were successfully used for side-channel attacks, e.g., monitoring the control flow of applications or accessing cryptographic keys [54]. To prevent this, Intel implemented Energy Filtering, which adds random noise to the measured power and randomly changes the update interval of counters if Intel Software Guard Extensions (SGX) are enabled or energy filtering is explicitly switched on[13]. The platform still uses the

[13]Energy filtering can be switched on explicitly until the next processor reset by setting a specific MSR

unfiltered measurements for power management. While this filtering added an error of up to 50% with the initial patch by Intel in 2020, this error was reduced with an update from 2021 to 2% at sampling intervals of 1 second or up to 15% when sampling RAPL every 100 ms [32]. As stated by Intel, this filtering only affects the package and PP0 RAPL domain, however, the 3rd Generation Scalable CPUs (e.g., Ice Lake-SP) are not explicitly listed as affected CPUs. On the systems used in this thesis, both Intel SGX and explicit energy filtering are disabled. Thus, the filtering cannot explain the high gap between RAPL and reference measurements.

Sensing Point at Voltage Regulator As described in figure 2.6, memory DIMMs are primarily powered by the voltage regulators (VRs) on the mainboard, even on systems with a Fully Integrated Voltage Regulator (FIVR). Such a VR internally measures the current and voltage and communicates it to the CPU package over SVID (see section 2.4), which utilizes this data for RAPL. This feature is called *IMON* and usually refers to the measurement of the output current of the VR. In this case, power measurements based on it would be expected to be close to our reference measurements, which are based on measurements directly at the DIMM pins. The exact specification of the demands of voltage regulators and their current sensing capabilities are specified by Intel in the non-disclosed VR 13 / VR 13.HC standard for the Ice Lake platform. The latest published version of the VR specification is available for the VR 11.1 standard [34] from 2009[14]. According to this document, *IMON* explicitly measures the "output load current" of the voltage regulator.

After an inspection of the server board of the Ice Lake system, the *PXE1410CDM-G003* VR controller (figure 4.14) has been identified as a candidate responsible for controlling power delivery to the memory. Again, a datasheet for this chip has not been published, but the datasheets for variants of the controller with different number of phases are available [25]. All these controllers feature multiple output channels that each can be configured for specific output voltages using a digital interface. The controller measures the input power consumption by measuring the current and actual voltage of the 12 V input. In addition, the output current of each channel, which can be used for computing the output power, is monitored separately. According to the datasheet [25], the current measurements are realized using an inductor-based current sensing circuit that achieves "best in class noise immunity" by oversampling the signal. The support for input power monitoring motivates that RAPL memory measurements on the Ice Lake system are

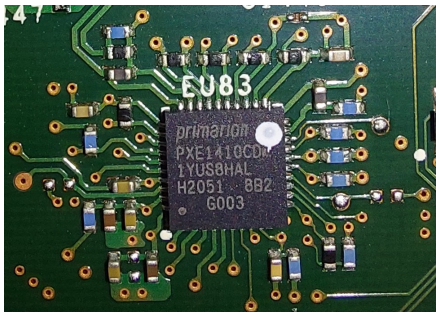


Figure 4.14.: PXE1410CDM voltage regulator controller chip on the Intel M50CYP2SB server board

[14]The VR 11.1 standard applies, for example, to the antiquated Intel Nehalem architecture.

likely to include the power loss at the voltage regulators in addition to the actual power consumption of the DIMMs. As shown in figure 2.7, typical voltage regulators for powering the memory have an efficiency in the order of 90% at medium output load that slightly decreases at higher loads. For low output current, the efficiency significantly decreases, resulting in a potentially high inaccuracy of RAPL. This is in line with section 4.3.4, where an increasing error of RAPL can be observed for lower power consumption.

The chosen memory population of only up to two DIMMs, which is limited by the number of riser cards and data acquisition channels we have available, reinforces this effect. While the voltage regulators are designed to power up to 16 DRAM DIMMs or 8 DRAM DIMMs plus 8 PMem DIMMs at once, the population with only two DIMMs results in a load current way below the voltage regulator's optimal efficiency.

To further investigate whether RAPL memory measurements are based on VR input or output monitoring, we collected a non-representative list of available voltage regulators with Serial Voltage Identification (SVID) support. A list of these regulators with target use cases and supported current sensing features is shown in table B.1 in the appendix. All listed VR13-compliant devices feature both input and output current monitoring, while some VR 12/12.5 products only provide output current sensing. For the *TPS544C26* [94], which specifically targets DDR5 power supply, the RAPL memory measurements are explicitly based on input power measurements, i.e., it features "Input power monitoring for DDR5 memory RAPL". Overall, it is likely that the RAPL memory measurements are based on input power sensing rather than output sensing on the Ice Lake-SP system. While both Skylake and Ice Lake server CPUs require the VR 13/VR 13.HC standard, RAPL memory measurements are based on output power measurements in the Skylake platform [23, p.75]. Thus, RAPL memory measurements are not based on input power sensing on every VR 13 system. The datasheet for a VR13 compatible controller [73] also explicitly states that it supports "input current sensing required for NVDIMM", which indicates that the VR13 specification might require input current sensing rather than output current sensing for persistent memory technologies. A possible reason for this is that these DIMMs are usually powered using three voltage rails, i.e., 1.2 V, 2.5 V, and 12 V, and using input current sensing allows monitoring all three power rails by power monitoring a single point.

This could also explain the phenomenon that RAPL sometimes reports a diverging power consumption while the reference measurements are the same (see figure 4.11). As the voltage regulator might automatically shut down phases to increase its efficiency (*phase shedding*, see page 19), the RAPL measurements can differ depending on the number of phases that are active. For example, if the output current is are close to the threshold for toggling a phase, the phase could continuously switch on and off during a roughly constant output power consumption.

With this in mind, the accuracy of RAPL memory energy measurements strongly depends on the installed voltage regulators and controllers, the memory population, and the current that is drawn by the installed memory modules. Our experimental

results (see table 4.6) confirmed that the gap between RAPL measurements and reference measurements at the risers, which is approximately the same instrumentation point as output current sensing at the VR, reduces with increasing current drawn by memory. The installation of additional DIMMs, which also increases the current, also decreases this error. This observation differs from previous work in which the accuracy of RAPL was mainly characterized in terms of the used CPU architecture [10, 23].

Accuracy of Current-sensing at Voltage Regulators Another error source is the actual accuracy of the current sensing at the VR (IMON). As already stated by Gough et al. [18] in 2016, the current sensing feature at VRs is typically optimized for the highest current it can deliver and is subject to an almost constant offset error. However, if the load current reduces, the relative error of the current sensing will increase significantly, resulting in high inaccuracies of RAPL measurements if systems are populated with significantly lower memory capacity than they have been designed for [18, p. 63]. The *VR11.1 Design Guidelines* give concrete values of the maximum current monitoring error on older architectures. At a load of more than 50%, the error needs to stay below 15%, and at a load up to 30%, an error of 24% is allowed, while the error is recommended to stay below 50% at a load of 15% [34]. Such requirements are not published for modern Intel systems, but some device vendors provide typical accuracy values of their current sensing circuits in their datasheets. For example, the *MP86901C* [101], which is advertised as very "reliable and accurate", has an accuracy of ±2% at 30 A load, ±4.0% at 10 A load, and ±7% at 5 A load. However, if only a single DDR4 module is used, the current for the 1.2 V line is usually less than 2 A, and the error is higher. The Infineon *TDA38640* [24], for example, has a rated accuracy of ±6% for output current monitoring (IMON) at maximum load - the accuracy is expected to be lower at light load.

Components Included in RAPL Measurements In addition to the `VCCIN` voltage delivered to the FIVR of the CPU, the memory VRs deliver two 1.2 V lines called VCCD_{0123} and VCCD_{4567} to the CPU. These voltages are used for powering the "memory interface", which is not specified further [26]. Potentially, this could mean that the memory VRs also power the integrated memory controller (iMC) of the CPU, which in turn may suggest that the RAPL memory domain also includes the power consumption of the iMC.

Another reason for the difference between reference measurements and RAPL is that we only measure the power consumption of two pins per DIMM, and the other pins are neglected due to their minor contribution to the total power consumption and the limited number of data acquisition channels we have available. However, the power consumption of the other pins and the power dissipation at the current-sensing shunts are expected to be included in the VR IMON. Thus, it is included in RAPL measurements, while it is not covered by our reference measurements. Overall, this difference is expected to be well below 500 mW and cannot be used to

explain the significant gap between RAPL and reference measurements.

While we discussed potential reasons for the offset in RAPL measurements in this section, we will discuss what our findings imply for users of RAPL with respect to temporal resolution and accuracy of the memory domain in the following section.

4.4. Discussion

The temporal resolution of RAPL has been evaluated and discussed in section 4.2. As pointed out, the update interval of RAPL counters is, on average, 1 ms but can be subject to a significant temporal variation. While previous work already has shown this on older architectures, this variation is up to 4 times higher on the Ice Lake platform compared to Broadwell-EP, yet the delay between two updates is at most 2 ms. Overall, RAPL counters should either be oversampled significantly or a sampling interval of at least 2 ms or higher should be chosen to prevent excessive aliasing. This also limits the applicability of RAPL for measuring the energy consumption of short code paths.

Based on the RAPL validation experiments from section 4.3, we have shown that the RAPL memory measurements are fundamentally different on the Ice Lake-SP system compared to previous architectures such as Intel Haswell or Broadwell. On previous platforms, the RAPL measurements were based on the output current monitoring of the voltage regulators powering the memory. This approximately matches the instrumentation point for our reference measurements at the DIMM riser. The Ice Lake system, which supports high-capacity PMem, however, uses the input power monitoring at the voltage regulators for power reporting and budgeting, resulting in a significant gap between RAPL and reference measurements. This gap, as we believe, primarily corresponds to the power losses at the voltage regulator, i.e., the difference between the input and the output of the voltage regulator in terms of power.

A possible reason for the changed instrumentation point for RAPL is that PMem modules are powered inherently different by the mainboard compared to DRAM. While DDR4 DIMMs are mainly powered using 1.2 V and 2.5 V provided by the mainboard, PMem DIMMs take a 12 V input from the mainboard and have an on-dimm Power-Management Integrated Circuit (PMIC), which regulates all required voltages for on-DIMM consumers. In addition, these modules use the 1.2 V and 2.5 V lines from the mainboard as secondary power supply. This different power supply topology of DDR4 DRAM and Optane PMem modules, that both share the same form factor and can be plugged interchangeably into the same memory slots on the mainboard, puts additional requirements on the mainboard's power delivery and voltage regulation components. On one hand, the voltage regulator should be able to power a system fully populated with high capacities of DRAM and no PMem, which requires a significant fraction of the power delivered by a 1.2 V line at high current. On the other hand, it needs to power heterogeneous populations with DRAM and PMem, which is primarily powered by a 12 V line at lower currents

compared to a 1.2 V supply.

While our results do not necessarily mean that RAPL measurements are wrong on the Ice Lake platform, it means that the instrumentation point for RAPL memory measurements may has changed without clear announcements by Intel. For example, this change is beneficial for estimating the power consumption of a complete node by summing up the power consumption of all RAPL domains, since RAPL also reflects power conversion losses. Conversely, it limits the usability of RAPL for measuring the actual power consumed by the memory, as the power losses during voltage conversion distort the power consumption of the actual memory. The nonlinearity of the losses (see figure 2.7) results in a non-linear correlation between the power consumed by the memory modules themselves and the power consumption reported by RAPL. This is confirmed by figure 4.7c, which shows that the gap between RAPL and reference measurements reduces with higher memory power consumption.

In addition, RAPL measurements are often inconsistent, i.e., RAPL reports different values for power consumption while the reference measurements are the same over time. This can be explained by phase shedding, an energy efficiency mechanism of the voltage regulators that regulate the number of active phases to be close to the maximum efficiency point of the regulator at different load levels. The hereby achieved reduction in power losses at the VR also decreases the gap between RAPL and reference measurements.

The difference in RAPL measurement methodology also has implications on power budgeting and limiting. If, for example, the power for all memory modules is capped to 200 W using RAPL because the system can only dissipate up to 200 W of heat directly at the DIMMs, the memory modules will be throttled way before they reach a power consumption of 200 W, resulting in lower memory performance than expected.

In this chapter, we have shown how to measure the power consumption of DDR4-compatible DIMMs by instrumenting the memory slots with risers. This approach is not perfect since it is unsuitable for full production scale. However, it gave us a sophisticated insight into the DIMM-level power consumption of DRAM and PMem. We also evaluated RAPL's accuracy on the Broadwell system, which is rather dated but the most similar system to the one used in a previous study [10] we had available. Our results are in line with that study. Thus, our experimental setup provides accurate measurements. Our results also raise awareness of the difference between RAPL and actual power measurements. We furthermore provided a detailed insight into how the mainboard powers DRAM and PMem memory modules, how the mainboard's power measurements are used for RAPL, and how RAPL's accuracy depends on installed memory capacity, population schemes, the mainboard, and the load of the memory.

5. Performance Metrics for Energy Consumption of Memory

In this chapter, we propose the Dynamic Energy per Load (DEL) and the Dynamic Energy per Store (DES) metrics for describing the instruction-level energy consumption of heterogeneous memory. After introducing the metrics, we describe an approach to measure the energy consumption and conduct memory energy measurements of synthetic workloads with sequential, strided, and random accesses on both PMem and DRAM. Based on the collected data, we evaluate DEL and DES with different access patterns and compare them between both memory types. In addition, we compute the Bandwidth per Watt (BpW) metric to assess the energy efficiency impact of various memory access patterns on heterogeneous memory systems.

5.1. Dynamic Energy per Load/Store Metrics

As discussed in chapter 3, the Energy per Instruction (EPI) is a well-known metric for characterizing the energy consumption of individual CPU and GPU instructions. This metric has already been used for characterizing CPU energy consumption of memory load and store instructions in previous work [63].

The Dynamic Energy per Load (DEL) and Dynamic Energy per Store (DES) metrics characterize the average dynamic energy consumed by the memory to complete a load or store instruction executed by the CPU because these memory access instructions are the primary cause of the memory energy consumption. The dynamic energy is the total energy consumption of the instruction minus the energy that the memory would consume during the same period when no memory access occurs, i.e., the idle power consumption times the execution time of the instruction. These metrics can be universally computed for different scenarios, such as different memory access patterns on heterogeneous memory.

Both metrics cover the dynamic energy consumed by an instruction on average. In practice, the dynamic energy consumption can differ significantly between consecutive instructions. For example, in sequential loads, a cache miss will cause a high dynamic energy consumption of the memory, while a cache hit is free in terms of dynamic memory energy consumption. These metrics amortize this effect, as our goal is to focus on the main memory domain. In addition, both metrics also include typical variations in energy consumption and, thus, should always be reported with a measure of uncertainty, such as confidence intervals.

5.2. Energy Measurement Methodology

A reliable method for measuring the energy consumption of the memory modules is required to evaluate the DEL and DES metrics on different heterogeneous memory types under varying access patterns. Initially, we utilized Running Average Power Limit (RAPL), which reports the accumulated energy consumption of all DIMM memory modules over a software interface (see section 2.3.2), for this. In this section, we discuss why RAPL cannot be used for measuring energy in this case and present an alternative methodology using physical power measurements directly at the memory modules.

The dynamic power of each memory type can be approximated when memory is only allocated on either DRAM or PMem by subtracting the idle power consumption from the total power consumption under load. While this was the initial plan for this thesis, the results from the RAPL validation (see chapter 4) emphasized that we cannot use RAPL for accurate and consistent power measurements of the memory modules themselves. In particular, the fact that the RAPL memory energy includes significant power delivery losses, which are non-linear in terms of power drawn and make RAPL report more power than consumed by the memory modules, rules out RAPL for our purposes.

5.2.1. Physical DIMM Energy Measurements

As discussed above, we decided that RAPL cannot be used for our purposes, so we adopted our reference measurement approach using riser cards from section 4.3.1 for measuring the energy drawn by the memory domain during the execution of code kernels.

For this, we increased the power sampling interval at the MCC128 DAQ device from 1 kHz to 25 kHz and integrated these samples over time to get the energy consumption. The integral is computed numerically using the `trapz` function from *NumPy*. The significantly increased sampling rate, which is also the maximum the device supports when sampling with four channels, reduces the effect of aliasing and improves the overall accuracy of the energy measurements.

As we have only two riser cards available, the power of only two DIMMs can be measured simultaneously. In particular, we cannot measure the power consumption of a system with fully populated memory slots. Thus, we decided to focus on single DIMM's power consumption only. While this synthetic scenario is not considered practically relevant because only the bandwidth of a single memory channel is available, it is the best solution with the equipment we have access to and gives a more generalizable view of memory performance and energy efficiency.

Another challenge is that we are only interested in the power consumption during the execution of the actual compute kernel rather than in the initialization or finalization phases, as these would significantly impact our results. In many cases, the energy consumption of the initialization can exceed the energy consumed by the kernel drastically in synthetic benchmarks. A common technique for solving this

is to instrument the code so that only the energy consumption during the execution of a specific code region is measured by wrapping the part with markers in the code. This technique is implemented by several performance monitoring tools such as LIKWID [97] or PAPI [59] that provide high-level APIs for this purpose and support energy monitoring using RAPL. As we were not using RAPL for our purposes and our own energy measurement methodology could not be implemented into the abovementioned tools, we developed a C header file with macros for starting and stopping energy measurements. These markers can be placed in the code to measure the energy consumption only between the `MemEnergyMeasure_Start` and `MemEnergyMeasure_End` marker. An example of how this can be used for a simple array reduction is shown in listing 5.1. Our approach is currently limited to only one instrumented region per executable.

```
#include "measure_memory_energy.h"
#include <stdlib.h>

int main(int argc, char** argv) {
    size_t N = 1024 * 1024 * 1024;
    MemEnergyMeasure_Init
    double* data = (double*) malloc(sizeof(double) * N);
    for(int i = 0; i < N; i++) {
        data[i] = (double) rand();
    }
    double sum = 0.0;
    MemEnergyMeasure_Start
    for(int i = 0; i < N; i++) {
        sum += data[i];
    }
    MemEnergyMeasure_End
    free(data);
    MemEnergyMeasure_Finalize
    return 0;
}
```

Listing 5.1: Code instrumentation for energy measurements of a specific kernel

The overall energy measurements work this way:

1. The Python script `measure-energy.py`, which takes the name of the application to monitor and the executable or bash command to monitor as a parameter, is executed on the server.

2. The application name and a unique identifier are sent to the Raspberry Pi over a configurable serial port.

3. The Pi prepares the measurements by configuring the MCC128 accordingly. The measurements will start when the server sends a trigger signal to the MCC128.

4. A new process that executes the executable is started on the server.

a) The connection to the Pi is started using `MemEnergyMeasure_Init`, and workload-specific initialization is performed (e.g., allocating and initializing data).

b) The trigger signal is sent to the Pi using `MemEnergyMeasure_Start`, and the kernel to monitor is executed.

c) A finalization signal is sent to the Pi using `MemEnergyMeasure_End` immediately after the termination of the kernel.

d) The application prints the number of memory loads and stores during the execution of the compute kernel to the standard error stream for processing by the parent script and performs some finalization steps.

5. The Pi sends the execution time of the kernel, the aggregated energy consumption, and the average power to the server over the serial interface.

6. The parent script aggregates the results from the Pi and the child process and outputs the final energy measurements.

The trigger signal sent by the application under test is inherently optional as the parent script can be configured to directly send the trigger signal before the child process starts and after it terminates. This can be used for measuring the energy consumption of applications that cannot be modified for instrumentation or in cases where the measurement of a specific kernel is not required.

5.3. Synthetic Workloads for Measuring Instruction Level Memory Energy Consumption

The very short execution time of memory load and store instructions, usually in the order of nanoseconds, makes it impossible to measure the energy consumption of individual instructions with our measurement infrastructure. To overcome this, we created synthetic workloads with a large, known number of load and store instructions and measured the energy consumption of the target memory during the execution of the whole workload. Then, we subtracted the static energy consumption, i.e., the energy that the memory consumes when idle, and divided the resulting dynamic energy consumption by the number of loads or stores, which is known from the code. In our case, each workload consists of a repetitive execution of a kernel until at least 3 seconds of wall time elapsed. An execution time of at least several seconds is required to hide possible synchronization latencies between the system under test and the data-acquisition device. While the synchronization of the measurement start is tight with a synchronization delay of less than 1 ms due to the digital trigger functionality of the MCC-128 device, a higher delay can be expected for synchronizing the end of the measurements as no trigger signal for stopping measurements is supported by the MCC128. Instead, the Pi waits for a falling edge, i.e., a transition from a high to low signal, at the trigger pin and instructs the MCC128

to stop measurements when the falling edge is detected. Additional processing by the Pi can add a synchronization delay in the order of several milliseconds.

For sequential and strided loads and stores, an array of fixed length is allocated and initialized with constant data using first-touch in parallel. Then, the array elements are accessed in an OpenMP parallelized `for` loop with a possible stride that is implemented as the number of elements skipped between consecutive loop iterations. Loads are realized using the `movq` instruction in inline assembler to prevent the compiler from optimizing away the operation. Stores are implemented as simple array element assignments. For sequential loads and stores, we varied the number of threads and considered every number of threads between 1 and 32. Strided accesses, on the other hand, were executed with 1, 4, 8, 16, and 32 threads but with varying strides between 1 element (8 Byte) and 34 elements (272 Byte).

Random loads are implemented by initializing an array with each element's index as its value, e.g., the value 42 is assigned to array index 42. Then, this array is permuted using the Fisher-Yates shuffling algorithm to create an index permutation with only a single cycle. In the actual compute kernel, multiple threads are spawned using an OpenMP parallel region that each perform pointer-chasing on the array with permuted indices. The start indices are different for each thread and are evenly spread over the whole array. For random loads, we varied the number of threads and considered each thread count between 1 and 32.

We only consider 64 bit loads and stores here. All workloads have been written in C and have been compiled using the `gcc` compiler (version 13.2.0) with default parameters and OpenMP enabled. Memory allocations are performed using *libnuma* on the local memory of socket 0 (either NUMA node 0 for DRAM or node 2 for PMem), and CPU threads are pinned to socket 0 using `numactl`. Each workload has been executed three times, and we report the arithmetic mean and the standard deviation for each.

All experiments have been carried out on the Intel Ice Lake system (table 4.2). To evaluate DRAM, we installed a single 32GB DDR4 module with a riser for power measurements in memory slot `A1` on socket 0. For the PMem evaluation, we mounted one DRAM and one PMem module in memory slots `A1` and `E1`, respectively, on socket 0. The power consumption of the PMem module was measured using a riser. Socket 1 was populated symmetrically but without risers.

5.4. Experimental Results

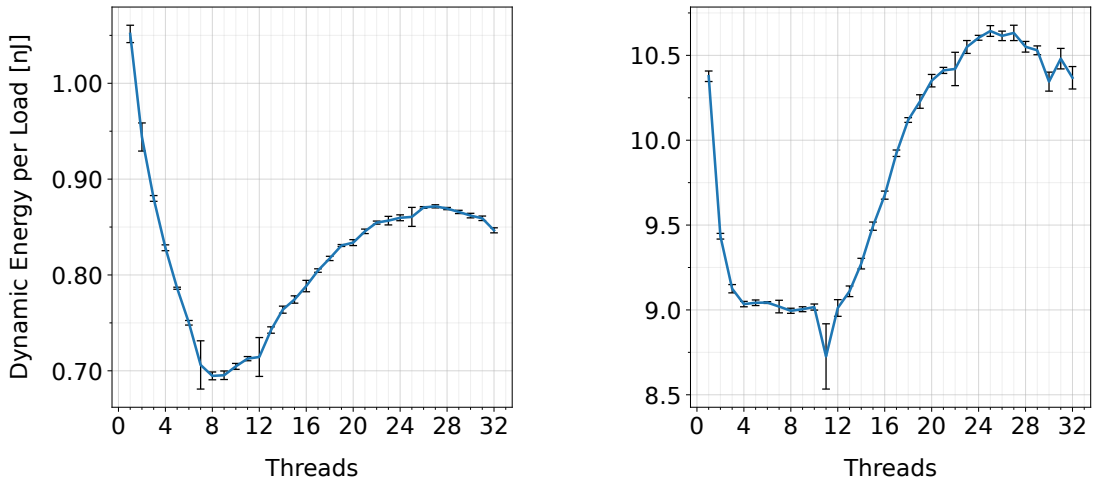
To compute the dynamic energy consumption of a memory module, the static energy needs to be subtracted from the total energy consumption. The static energy consumption can be computed by multiplying the power drawn by the module when idle and the workload's execution time. We measured an idle power consumption of 1.8575 W for Samsung 32GB DDR4 modules and 2.9668 W for the 128GB PMem modules. Both values here correspond to the minimum power consumption measured during five measurements of 10s on an idle system. This corresponds to the

typical idle power consumption when the operating system itself is active and regularly accesses the memory, but no workload is executed.

Taking the capacity of the memory modules into account, the DRAM module consumes 58 $\frac{\text{mW}}{\text{GB}}$ at idle while PMem consumes 23 $\frac{\text{mW}}{\text{GB}}$.

5.4.1. Sequential Access

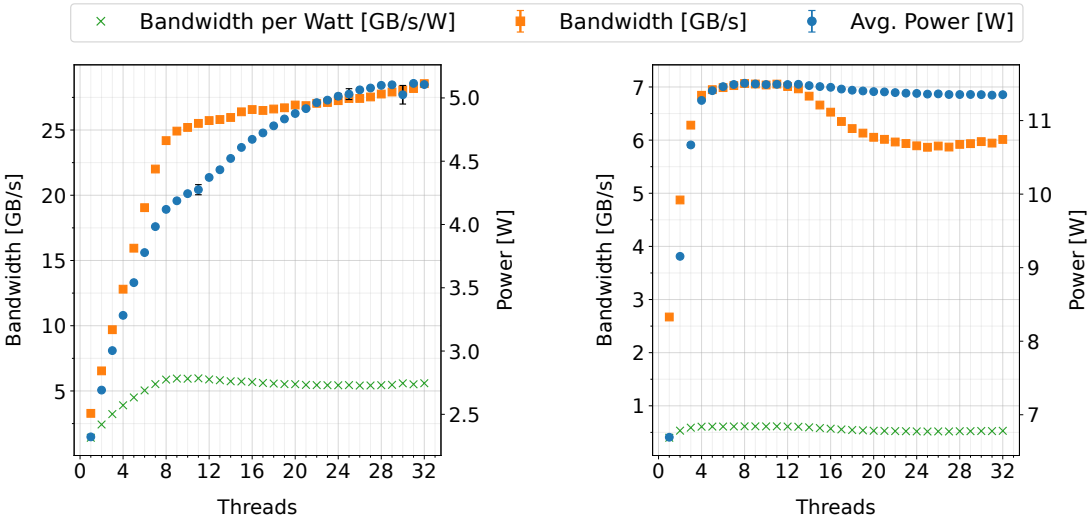
Figure 5.1 shows the mean DEL for sequential double-precision load instructions for both DRAM and PMem at different numbers of threads concurrently accessing the memory. A different view on the same data is provided by figure 5.2, which shows the effective memory bandwidth and corresponding power consumption at varying thread counts. In addition, the BpW metric, which relates the memory bandwidth to the power drawn by the memory, is shown. The corresponding measurements for the DES in sequential accesses are visualized in figure 5.3 and figure 5.4.



(a) DRAM

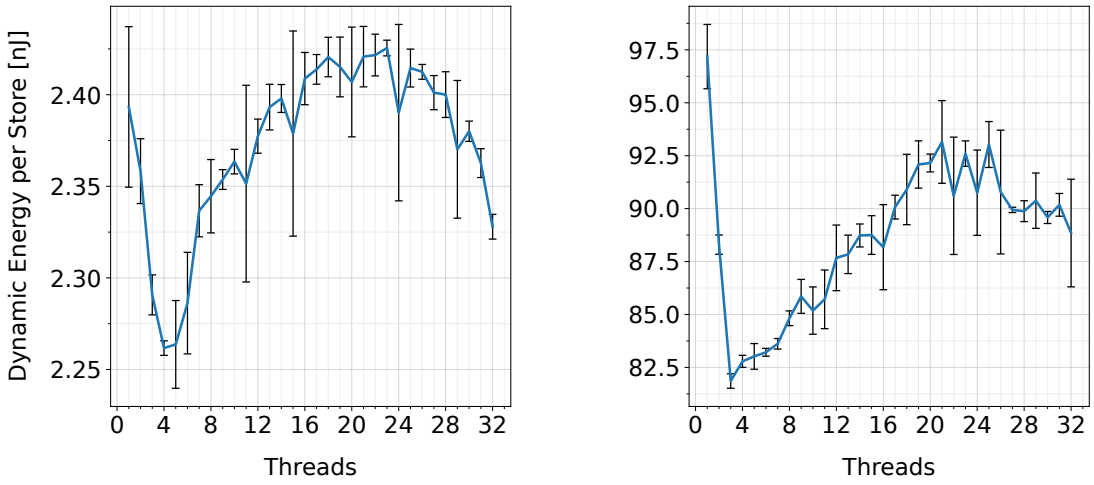
(b) PMem

Figure 5.1.: Dynamic Energy per Load (DEL) with standard deviation for sequential access at different numbers of threads



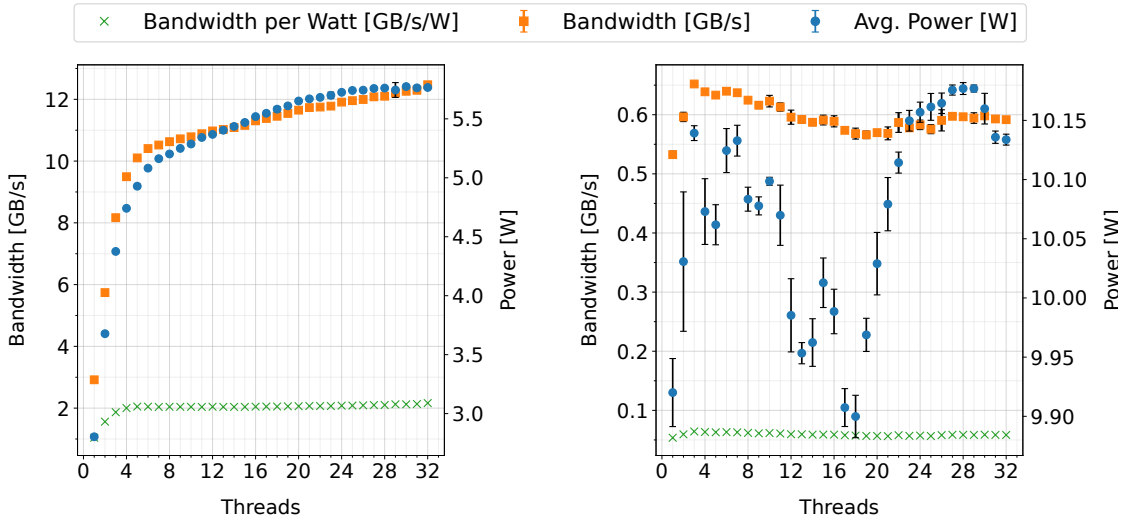
(a) DRAM (b) PMem

Figure 5.2.: Bandwidth per Watt (BpW) for sequential loads at different numbers of threads.



(a) DRAM (b) PMem

Figure 5.3.: Dynamic Energy per Store (DES) with standard deviation for sequential access at different number of threads.

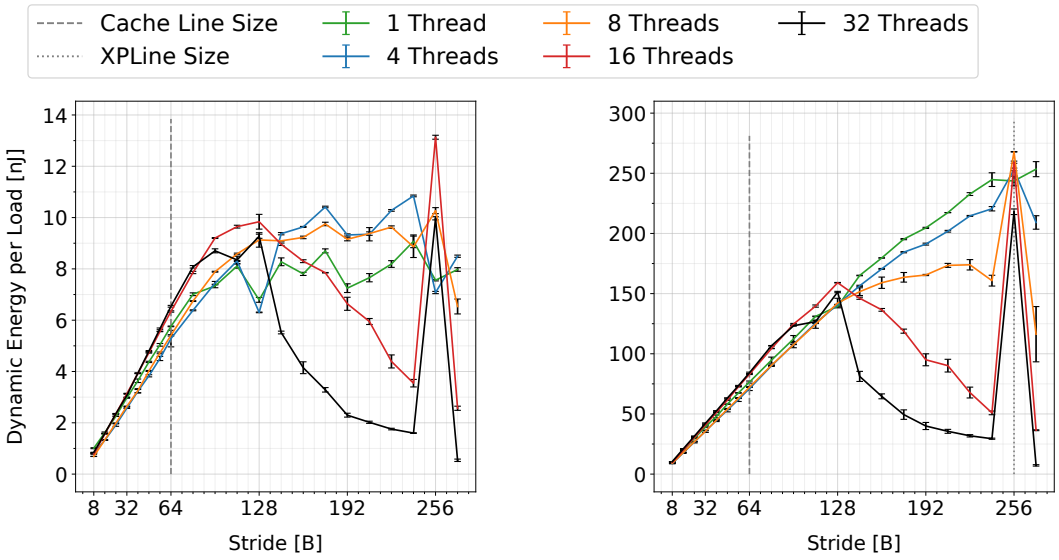


(a) DRAM (b) PMem

Figure 5.4.: Bandwidth per Watt (BpW) for sequential stores at different numbers of threads

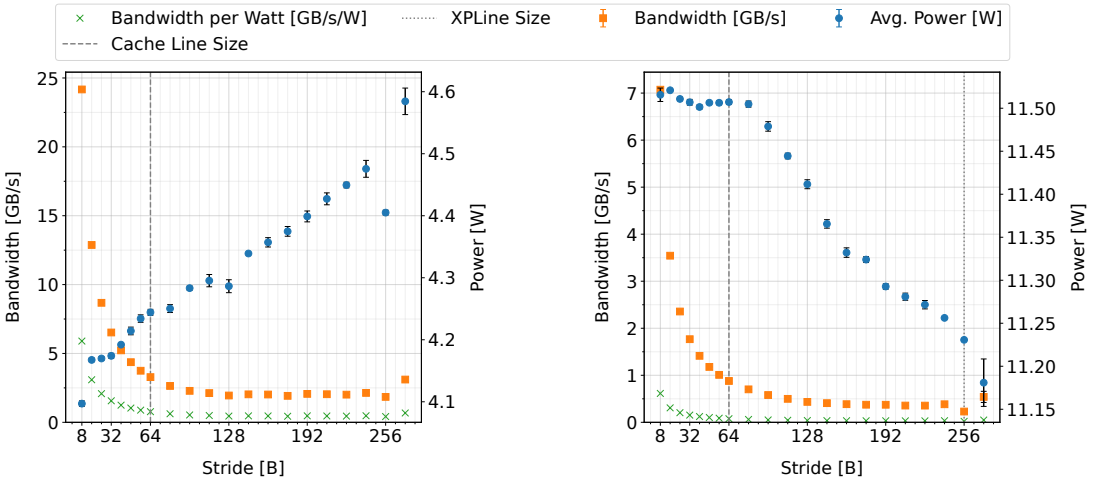
5.4.2. Strided Access

The DEL and DES for both DRAM and PMem have been measured for varying strides between 8 B and 256 B and with 1, 4, 8, and 16 threads and are visualized in figure 5.5 and figure 5.7. In addition, figure 5.6 and figure 5.8 show the memory energy efficiency in terms of BpW for strided loads and stores at 8 threads.



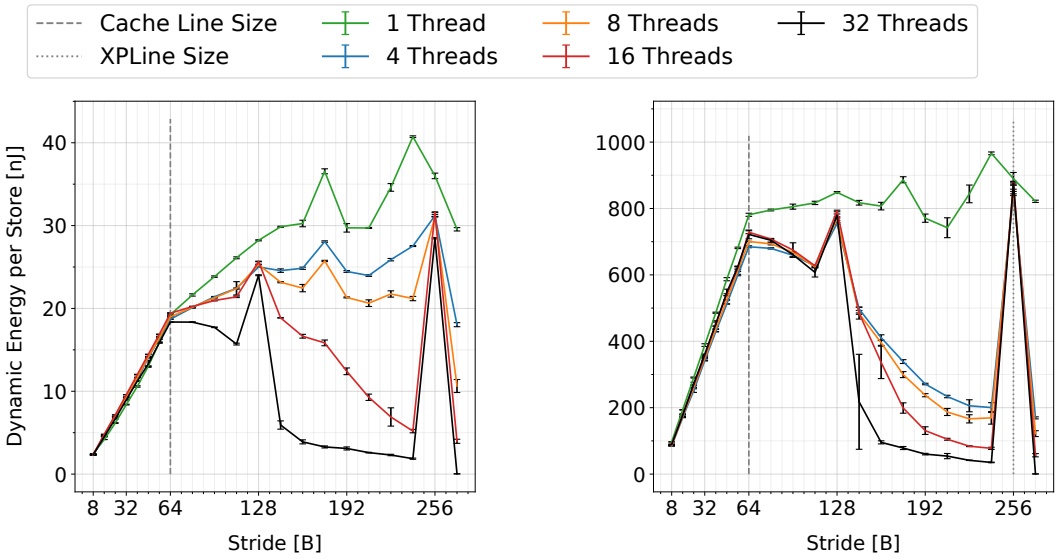
(a) DRAM (b) PMem

Figure 5.5.: Dynamic Energy per Load (DEL) for different stride lengths and thread counts



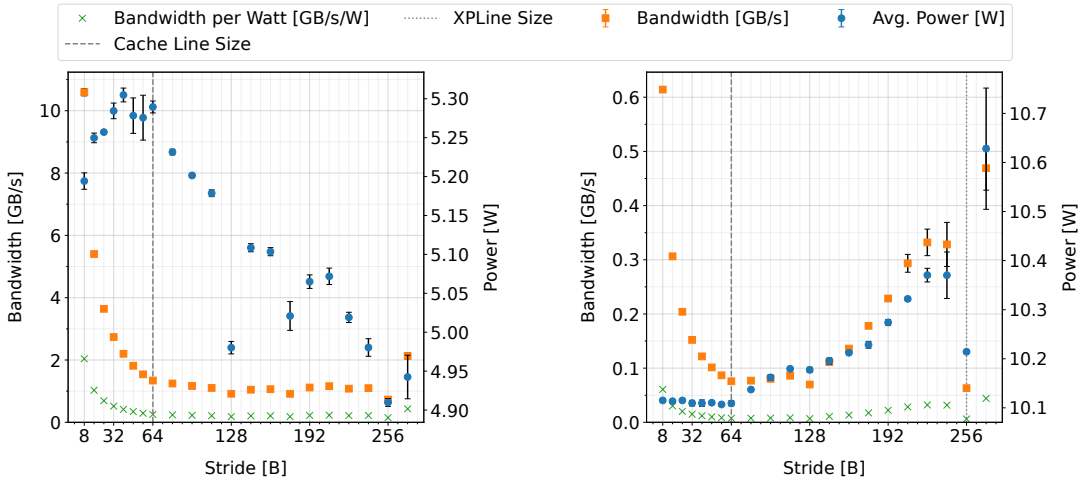
(a) DRAM (b) PMem

Figure 5.6.: Bandwidth per Watt (BpW) for strided loads with 8 threads



(a) DRAM (b) PMem

Figure 5.7.: Dynamic Energy per Store (DES) for different stride lengths and thread counts

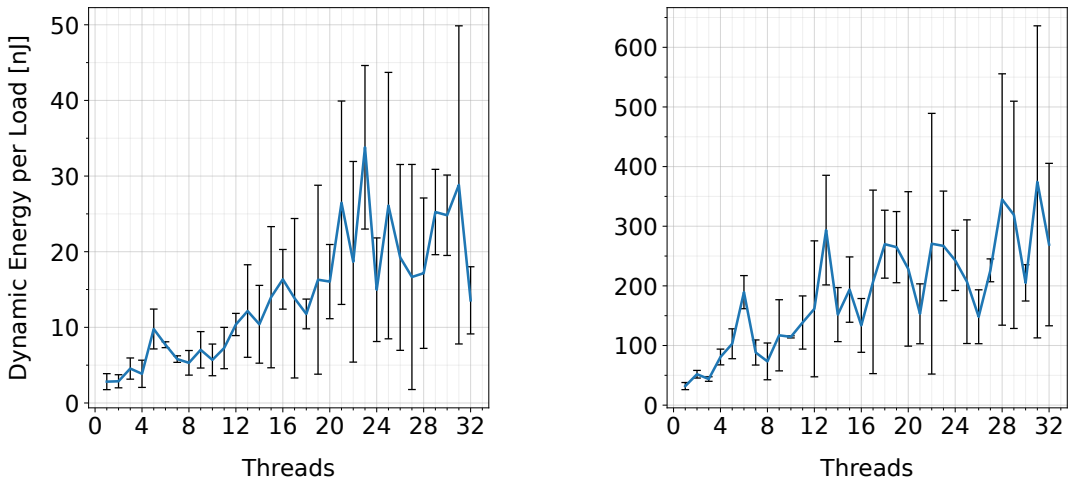


(a) DRAM (b) PMem

Figure 5.8.: Bandwidth per Watt (BpW) for strided stores with 8 threads

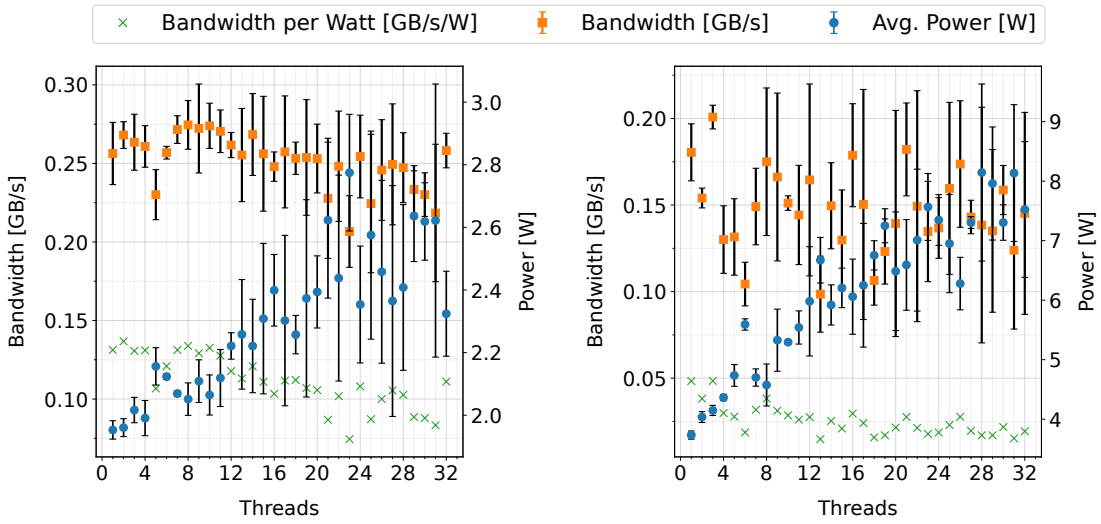
5.4.3. Random Access

The mean DEL over three measurements for random loads is shown in figure 5.9 and the corresponding BpW is depicted in figure 5.10.



(a) DRAM (b) PMem

Figure 5.9.: Dynamic Energy per Load (DEL) for different thread counts for random access



(a) DRAM (b) PMem

Figure 5.10.: Bandwidth per Watt (BpW) for random loads at different numbers of threads

5.4.4. DRAM vs. PMem Comparison

Figure 5.11 shows the mean DEL and DES with respective standard deviations for different access patterns for DRAM and PMem on an exponential scale. The results presented in the previous part of this section have been grouped by memory type

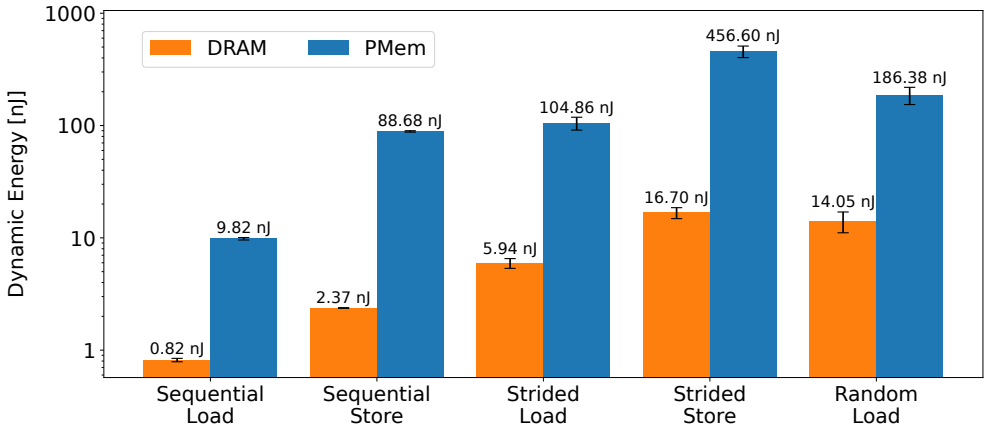


Figure 5.11.: Average energy per load and store instructions comparison between DRAM and PMem on an exponential scale.

and access pattern, and the resulting mean has been computed regardless of thread count or stride for this particular figure.

The dynamic energy consumption per load and store is significantly higher on PMem compared to DRAM. For example, it is about 12 times higher for sequential loads, 37 times higher for sequential stores, 17 times higher for strided loads, 27 times higher for strided stores, and 13 times higher for random loads.

5.5. Discussion

This chapter introduced the DEL and DES metrics, which characterize the dynamic energy consumption of heterogeneous memory during load and store instructions. The metrics can be used to quantitatively compare the energy consumption between different access patterns or memory architectures, taking both the power consumption and the instruction throughput into account. For example, if the power drawn is constant and the throughput (e.g., the read bandwidth) doubles, DEL reduces by 50%. When the power consumption doubles, and the bandwidth reduces by a factor of 4, DEL increases by a factor of 8.

The BpW metric, on the other hand, gives a different view of energy efficiency. It relates the total power consumption, which includes the static and dynamic power, to the memory bandwidth[1]. DEL and DES describe the costs in terms of dynamic energy, whereas BpW relates to memory energy efficiency. Consequently, lower DEL values are desired, while a higher BpW value is considered better.

[1]The Energy per Load metric would be equivalent to BpW. DEL and BpW are not equivalent however.

5.5.1. Sequential Access

For sequential loads, as shown in figure 5.1, the DEL reduces with increasing threads until it reaches a minimum with 8 threads for DRAM and 4 threads for PMem. The BpW metric also reflects it as it becomes maximal with 8 threads on DRAM and 4 threads on PMem, respectively, according to figure 5.2. With more than 12 threads, the loads become less efficient again, particularly well-visualized in figure 5.2b. With six threads, both the power and the bandwidth reached a plateau until bandwidth dropped with more than 12 threads, while the power consumption stayed roughly constant. This can be explained by the contention of the different buffers, particularly the 256B XPBuffer of the PMem module or the CPU's integrated memory controller (iMC) [103]. This effect is not present on DRAM, for which both power consumption and bandwidth increase continuously with more threads while the BpW slightly reduces with more than 10 threads. Overall, the best energy efficiency for sequential accesses to PMem directly corresponds to the best performance. Thus, users only need to focus on the performance when programming PMem to get the best energy efficiency for sequential memory access.

In compute-bound applications, the memory bandwidth is not saturated, and the memory accesses might not be highly efficient. However, as the CPU's power consumption generally exceeds the memory's power consumption in CPU-bound scenarios, more focus on the energy efficiency of the CPU should be put instead. In particular, optimizing only for memory energy efficiency can lead to energy efficiency penalties at the CPU.

DES and BpW (see figures 5.3 and 5.4) follow an overall similar pattern for sequential stores. While the DES is about three times higher for DRAM and ten times higher for PMem compared to the read-only scenario, the best energy efficiency can be observed with 4 threads for both memories. The BpW graph for PMem (figure 5.4b) stands out here as the total power consumption of the memory seems to be very inconsistent and subject to significant variations at first sight. However, the power drawn by the module is only between 9.92 W at a single thread and 10.13 W at 32 threads, while the power consumption ranges from 7 W to 11.5 W for sequential loads on PMem. The total DRAM consumption is significantly higher for writes compared to reads. Surprisingly, the opposite can be observed for PMem: The total module power consumption during writes is lower than during reads.

5.5.2. Strided Access

For strided accesses, we measured the memory energy consumption for 1, 4, 8, 16, and 32 threads but also varied the stride between 1 and 34 `double` elements. As shown in figure 5.5, the DEL increases linearly with increasing stride as long as the stride is smaller than the cache line length, which is 64 B in our case. If the stride exceeds 128 B (i.e., 16x double), the energy efficiency decreases for higher thread numbers, while it increases or stays roughly constant for fewer threads. For a stride of 256 B, the DEL spikes for 8 or more threads both on DRAM and PMem. While

this is expected for PMem, as subsequent memory accesses are in different XPLines, we do not have an explanation for this on DRAM.

The corresponding BpW graph (figure 5.6) is particularly interesting here. For PMem, the power consumption is roughly constant for strides up to 64 B. A possible reason for this is that two consecutive memory accesses are within the same XPLine of 256 B. The bandwidth, on the other hand, decreases with increasing strides within a cache line as the cache line utilization at the CPU drops. For a stride greater than 64 B, the power consumption of the PMem surprisingly drops gradually, while the power consumption of the DRAM increases. A possible reason for this is a contention of on-DIMM buffers on the PMem.

Strides should generally be kept as short as possible to improve memory bandwidth and energy efficiency. However, strided accesses are often a fundamental element of an algorithm and cannot be optimized. If larger strides, for example, between 128 B and 256 B, are used, the thread number should be greater than 16 to improve energy efficiency.

Strided stores show an overall similar pattern compared to strided loads. However, the DES is significantly higher than DEL. In contrast to strided loads, however, the power consumption of the memory slightly increases for strides larger 64 B on PMem, while it decreases for DRAM (see figure 5.8).

5.5.3. Random Access

For random accesses, we only considered random loads and neglected writes. The main reason for this is the difficulty of creating synthetic write-only workloads with random accesses without utilizing another array or random number generators during the execution. Figure 5.9 shows the DEL for random loads from DRAM and PMem at different numbers of threads. In this case, the energy consumption is subject to a large variance at 12 threads or more on both memory types. Overall, the dynamic energy per load increases with more threads and is minimal for single-threaded access. Based on figure 5.10, this is mainly caused by an increase in power consumption of the memory and only a less substantial decrease in bandwidth at a higher thread count. Highly concurrent random reads should be avoided if possible, as they perform poorly on both DRAM and PMem in terms of bandwidth and energy efficiency.

The larger standard deviation in figures 5.9 and 5.10 also indicates that the energy consumption and bandwidth strongly differ between different workloads with random accesses. In this thesis, we only investigated a single application with random memory accesses. However, a large number of varying access patterns are classified as “random access“ [67]. The workload used for this thesis serves as a worst-case scenario, as the memory accesses are random and spread over a comparably large memory address range but cannot be considered representative. In particular, some random access patterns might exhibit some temporal and spatial locality and are expected to perform comparably better as a result.

5.5.4. Energy Efficiency of Intel Optane PMem

Our previous experiments have been designed to directly compare the energy efficiency of DRAM and PMem on the same workloads using the proposed DEL and DES metrics.

For the memory modules evaluated in this thesis, we measured an idle power consumption per capacity of 58 $\frac{\text{mW}}{\text{GB}}$ and 23 $\frac{\text{mW}}{\text{GB}}$ for DRAM and PMem, respectively. Thus, although PMem modules draw more static power than DRAM modules, PMem can be considered more energy efficient compared to DRAM when idle[2]. The per-capacity perspective on power consumption, however, cannot be transferred to the dynamic power consumption of the memory.

As shown in figure 5.11, PMem is significantly less energy efficient with respect to per load and store dynamic energy consumption. Compared to DRAM, writes exhibit significantly lower energy efficiency on PMem. For example, DES divided by DEL is on average 2.93 nJ for DRAM while it is 9.06 nJ on PMem.

Overall, we conclude that PMem in AppDirect mode cannot compete with DRAM as a technology for main memory in classical HPC applications in terms of performance and overall energy efficiency. For applications requiring large main memory capacities that cannot be provided with DRAM on a single node, for example, training and evaluating machine learning models on enormous datasets, PMem may still be considered as it offers up to 6 TB of combined main memory capacity on a single server [30] at the cost of lower energy efficiency compared to DRAM. PMem was initially designed as a competitor to traditional solid state disks, primarily in database applications, and to fill the gap between an SSD and DRAM. For these use cases, in which large amounts of data need to be stored persistently, PMem offers comparable sequential read performance to a high-end PCIe4 SSD [33] and offers "significant performance and energy efficiency gains compared to a typical SSD" [40]. The Samsung PM9A3, for instance, offers a 283 (MB/s)/W energy efficiency while a single 128 GB PMem module offers an energy efficiency of about 500 (MB/s)/W. An in-depth energy-efficiency evaluation of PMem compared to traditional solid-state disks is, however, out of scope for this thesis.

[2]When we observed the highest power consumption under load in the previous experiments, DRAM consumed 181 $\frac{\text{mW}}{\text{GB}}$ and PMem 90 $\frac{\text{mW}}{\text{GB}}$.

6. Memory Access Pattern-based Energy Estimation & Metrics Evaluation

The goal of this chapter is to showcase how the Dynamic Energy per Load (DEL) and Dynamic Energy per Store (DES) metrics can be utilized for estimating the energy consumption of real-world applications. Furthermore, the evaluation of the metrics' accuracy is discussed. We follow the same experimental setup as in the previous chapter. The power consumption of a single DRAM (slot `A1`) or PMem (slot `E1`) DIMM mounted on socket 0 is measured using a riser on the Ice Lake system. The other socket is populated symmetrically, but memory is allocated on socket 0, and threads are pinned to socket 0.

6.1. Metrics-based Energy Estimation

The metrics DEL and DES can be used for straightforward energy estimation. For this, we need to analyze the application of interest regarding the number of load and store instructions for each memory access pattern based on the classification presented in section 2.2. In other words, the number of load and store instructions in sequential, strided, and random access, along with the corresponding number of threads and possible stride length, needs to be counted. By multiplying the number of loads with the measured DEL and the number of stores with the measured DES and computing the sum of both results, an estimate for the total dynamic energy consumption of the memory during the kernel execution can be computed.

```
long N = 1000000000;
#pragma omp parallel for num_threads(8)
for (long i = 0; i < N; i++) {
    x[i] = y[idx[i]];
}
```

Listing 6.1: Example parallelized *Gather* kernel consisting of sequential loads and stores and random loads

For example, each iteration of the loop in the gather access, as shown in listing 6.1, consists of one sequential read to `idx`, one sequential write to `x`, and one random read to `y`. The computation of the energy estimation for this example is visualized in table 6.1. On DRAM, the code kernel includes 10^9 sequential load

			DRAM		**PMem**	
Access Pattern	Threads	Count	DEL/DES	Total	DEL/DES	Total
Seq. Load	8	10^9	0.69 nJ	0.69 J	8.99 nJ	8.99 J
Seq. Store			2.34 nJ	2.34 J	84.82 nJ	84.82 J
Random Load			5.31 nJ	5.31 J	122 nJ	122 J
				8.34 J		215.81 J

Table 6.1.: Exemplary dynamic energy estimation for Gather kernel with 8 threads and 10^9 elements accessed.

accesses, and we measured that a single double-precision load consumes 0.69 nJ on average in sequential access. The total dynamic energy consumption is estimated at $0.69\,\text{nJ} \times 10^9 = 0.69\,\text{J}$ for all sequential loads in the kernel. By summing the dynamic energy for the sequential loads (0.69 J), sequential stores (2.34 J), and random loads (5.31 J), the estimated dynamic DRAM energy consumption of the whole kernel is 8.34 J. Predicting the total energy consumption furthermore requires an estimation of the static energy consumption during code execution. Because the static energy consumption ideally only depends on the execution time, it can be computed by multiplying the idle power of the DRAM module, which is 1.8575 W according to section 5.4, with the execution time of this kernel.

To facilitate this computation, we provide a calculator for the energy estimation model in the supplementary material of the thesis (see figure C.1). Using the described approach, the energy consumption of the evaluated DRAM and PMem modules during the execution of an application, a specific part of an application, or a kernel can be estimated. This requires that the number of load and store instructions is deterministic and can be estimated. Furthermore, the specific access patterns of the different kernels need to be known and decomposable into the "atomic" access patterns for which we collected energy measurements in the previous chapter. However, it is unclear how reliable and accurate this energy model is. While the DEL and DES metrics are based on load and store instructions only, a more detailed study on mixed-instruction metrics is necessary to better estimate the energy consumption for real-world applications, which are inherently mixed-instructions workloads. We leave this for future work as it requires additional investigation. However, we evaluate the DES and DEL metrics with the following two real-world applications to demonstrate the methodology for evaluating such metrics in energy consumption estimations.

6.2. Case Study: Sparse Matrix Vector Product (SpMV)

The matrix-vector multiplication is an essential element of several matrix diagonalization algorithms. Real-world applications often need to deal with sparse matrices, i.e., with most entries being zero. Several sparse matrix representations have been developed to represent these matrices in memory and on disk efficiently. Apart from the well-known Compressed Row Storage (CRS) format, the Coordinate Format (COO) is another format for representing such sparse matrices. The COO format stores each non-zero entry of the matrix in a tuple containing the column and row indices of the element and its value. For example, the 4×4 matrix A can be represented in the sparse COO format as shown in A_{COO}:

$$A = \begin{bmatrix} 1 & 0 & 0 & 6 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 11 & 0 & 0 \end{bmatrix} \qquad A_{COO} = [(0,0,1),\ (0,3,6),\ (1,1,2),\ (2,2,3),\ (3,1,11)]$$

Listing 6.2 shows how the product of such a COO-encoded matrix and a vector `B` can be computed and stored in the array `C` using OpenMP shared-memory parallelization. This implementation was created by S. Muneeb [85] and was optimized and restructured for this thesis.

```
1  auto temp = new double[threads * mtx->cols];
2  // [..] Initialize 'temp' with 0
3
4  #pragma omp parallel for default(none) shared(mtx,temp,B,cout)
5  for (long i = 0; i < mtx->nonzeros; i++) {
6      int tid = omp_get_thread_num();
7      long row_index = mtx->row_idx[i];
8      long col_index = mtx->col_idx[i];
9      double val = mtx->vals[i];
10     temp[tid * mtx->cols + row_index] += B[col_index] * val;
11 }
12
13 #pragma omp parallel for default(none) shared(mtx,temp) shared(C)
14 for (long i = 0; i < mtx->cols; i++) {
15     double sum = 0;
16     for (int j = 0; j < omp_get_num_threads(); j++) {
17         sum += temp[j * mtx->cols + i];
18     }
19     C[i] = sum;
20 }
```

Listing 6.2: Sparse matrix-vector product in Coordinate Format (COO)

In our case study, we measure the dynamic energy consumption (see section 5.2) during the execution of this algorithm on a large, real-world matrix and compare it to the estimated energy consumption based on our proposed model. We have chosen

the `Flan_1565` matrix as a real-world example matrix, which represents hexahedral finite elements [35]. This $1,564,794 \times 1,564,794$ matrix contains $59,485,419$ non-zero entries, i.e., only 0.002% of the elements are populated. While a dense matrix representation requires at least 17.8 TiB of storage, the sparse representation only occupies 1.57 GiB of disk space. In our case study, we multiply this matrix in COO representation by a vector containing 1.0 values in double-precision floating-point format.

To apply our model, we first need to count the number of memory loads and stores in listing 6.2 for each memory access pattern we investigated in section 5.4. For a sparse matrix of structural size $N \times N$ with N_z non-zero elements and T threads, the main kernel consists of $3 \times N_z$ sequential loads (lines 7–9) and N_z random loads (line 10). In addition, line 11 contains a random load-modify-store construct, which we model as a random load plus a sequential write. This adds an additional N_z random loads and N_z sequential writes to the main kernel.

The initialization of the `temp` array in line 2 consists of $T \times N$ sequential stores, and the final postprocessing consists of N sequential stores and $T \times N$ strided loads with a stride of N elements. As N exceeds 32, which is the highest stride we measured in chapter 5, we model it as a random load[1].

Table 6.2 shows the total count of estimated loads and stores, along with the measured DEL and DES values from chapter 5 and the resulting dynamic memory energy estimation for four threads.

		DRAM		**PMem**	
Access Pattern	Count	DEL/DES	Total	DEL/DES	Total
Seq. Load	$3 \times N_z$	0.83 nJ	1.48 J	9.03 nJ	16.11 J
Seq. Store	$N_z + (T+1) \times N$	2.26 nJ	1.52 J	82.79 nJ	55.73 J
Random Load	$2 \times N_z + T \times N$	3.86 nJ	4.83 J	80.74 nJ	101.11 J
			7.83 J		172.95 J

Table 6.2.: Dynamic energy estimation for SpMV computation on the Flan_1565 matrix with $T = 4$, $N = 1,564,794$, $N_z = 59,485,419$

As a next step, we ran the actual application with different numbers of threads (1, 4, and 8). We measured the mean energy consumption of the memory during three repetitions using the methodology described in chapter 5 and computed the dynamic energy consumption. In addition, the dynamic energy consumption from RAPL was collected[2]. Both measurements are compared to the metrics-based estimation in table 6.3. A significant difference between the measured dynamic energy

[1] This is an approximation as accesses with large strides can usually be better prefetched than a truly random access.

[2] For this, we measured the idle power according to RAPL. Based on RAPL measurements during workload execution, we subtracted the estimated static energy consumption from the total energy consumption. According to RAPL, a single DRAM DIMM draws 4.33 W when idle. When

Thr.	Measured		RAPL		Estimation	
	DRAM	**PMem**	**DRAM**	**PMem**	**DRAM**	**PMem**
1	2.1 ± 0.4 J	21.6 ± 2.0 J	-1.3 ± 7.1 J	19.4 ± 0.3 J	6.8 J	117.9 J
4	2.2 ± 0.1 J	29.2 ± 0.6 J	1.5 ± 0.1 J	27.8 ± 0.2 J	7.8 J	173.0 J
8	2.7 ± 0.1 J	37.5 ± 0.3 J	1.8 ± 0.1 J	35.7 ± 0.1 J	9.9 J	174.9 J

Table 6.3.: Dynamic energy of sparse matrix-vector product - measurements vs. RAPL vs. prediction for 1, 4, and 8 threads

consumption and the estimated values can be observed. For DRAM, the actual energy consumption is more than 3 times lower than estimated, and for PMem, it is about 5.5 times lower for the single-threaded execution. We expected that the estimation based on the DES and DEL metrics would result in higher dynamic energy values than the measured because the read- or write-only workloads tend to yield higher energy consumption than mixed-instruction workloads. RAPL reports an overall lower dynamic energy consumption than the reference measurements here. This is caused by the unreliable idle power measurements using RAPL, i.e., that RAPL reports a higher power consumption for a single DRAM module compared to the same DRAM module with an additional PMem module. While we have shown that RAPL overestimates the memory's power consumption (see chapter 4), the dynamic energy characterization using RAPL makes it appear to underestimate energy consumption.

6.3. Case Study: Conjugate Gradient (CG) for Solving Poisson Equation

As a second case study, we present an application that solves the Poisson equation using the Conjugate Gradient (CG) optimization algorithm. This source code by T. Econonom [13] was updated such that it only operates on 64b data types, i.e., `double` and `long`. The application contains several kernels with different access patterns. Table 6.4 shows each kernel's number of loads and stores and how often it is executed in a single iteration. N is the size of the matrix, N_z is the number of non-zero entries in the sparse matrix, and i is the number of iterations.

The energy consumption of this application was measured when allocating on DRAM or PMem five times each. We computed the mean dynamic energy consumption for both memory types and different numbers of threads. Furthermore, we stored the number of iterations in each experiment as this directly influences the energy consumption. The results, along with the mean execution time, are shown in table 6.5. For 1500 iterations, which is the approximate number of iterations in our

both DRAM and PMem are installed, RAPL reports 4.2 W for the memory domain of socket 0 when idle, which is unexpectedly lower compared to DRAM-only. These measurements were consistent throughout several reboots of the system.

Kernel	#Seq. Load	#Seq. Store	#Rnd. Load	#Invocations
vector_copy	N	N	0	1
vector_norm	N	0	0	$2 \times i$
dot_product	$2 \times N$	0	0	$3 \times i$
daxpy	$2 \times N$	N	0	$2 \times i + 1$
sp_mv_product	$N + 2 \times N_z$	n	N_z	$i + 1$

Table 6.4.: Number of memory loads and stores with memory different access patterns for the kernels in the CG application

	Dynamic Energy [J]		Time [s]	
Threads	**DRAM**	**PMem**	**DRAM**	**PMem**
1	0.29 ± 0.12	4.03 ± 1.90	18.22 ± 0.17	18.27 ± 0.06
4	0.12 ± 0.04	2.41 ± 0.56	4.44 ± 0.01	4.46 ± 0.02
8	0.07 ± 0.02	1.35 ± 0.33	2.27 ± 0.01	2.26 ± 0.01
16	0.01 ± 0.01	0.07 ± 0.01	1.24 ± 0.00	1.24 ± 0.01
32	0.01 ± 0.01	0.04 ± 0.02	0.84 ± 0.00	0.85 ± 0.01

Table 6.5.: Measured dynamic energy consumption and execution time of the CG application. Mean of 5 repetitions.

experiments, the estimated dynamic energy consumption for single-threaded access is 176 J for DRAM and 3353 J for PMem[3]. These estimates significantly exceed the measured dynamic energy consumption.

An obvious explanation for this phenomenon is that the application only allocates about 32 MiB of data, which is accessed repeatedly throughout the iterations. As the allocated data completely fits into the CPU's 48 MB last-level cache, almost all memory accesses can be satisfied from cache, and accesses to the underlying memory are not required. This also explains why the execution time remains constant regardless of the allocation on DRAM or PMem.

6.4. Discussion

The presented results indicate that the DEL and DES metrics report a higher dynamic energy than measured in both mixed-instruction case studies. Possible reasons for this overestimation of the model are:

- Our model does not consider temporal cache locality, i.e., the identical memory location is accessed multiple times within a short time. While the sequential accesses to the `row_idx`, `col_idx`, and `vals` arrays (see listing 6.2) exhibit no temporal locality, the random load from the `B` array and the accesses to `temp`

[3]The exact computation is analogous to table 6.2 and is omitted here.

are likely to regularly being satisfied from cache, in particular, because the CPU has a comparably large L3 cache of 48 MB. This is particularly true for the CG case study.

- The DEL and DES metrics can be seen as upper bounds of dynamic energy consumption for the read- or write-only workloads with corresponding memory access patterns. As mentioned above, mixed-instruction workloads must be evaluated with the relevant read-write ratios for each memory type to estimate the precise dynamic energy demands for real-world applications.

- The measurements from chapter 5 are based on workloads with only memory access instructions and essentially no arithmetic operations. The main kernel for the frist case study (lines 4–11 in listing 6.2), however, has six memory accesses and three arithmetic operations per loop iteration, resulting in an arithmetic intensity (AI) of 0.5. The application is memory-bound, but, the arithmetic operations still significantly impact its execution and reduce the load on the memory interface and memory itself. In particular, it also avoids a possible contention of the integrated memory controller (iMC) or on-DIMM buffers, for example, the *XPBuffer* of PMem.

- Our model does not reflect compiler optimizations. While the compiler and its configuration remained consistent throughout all experiments, more complex applications generally allow for more compiler optimization.

Overall, we have shown how to evaluate the proposed metrics and how to analyze the dynamic energy consumption of different memory types. The different measurement methods have shown that one should exercise caution when utilizing RAPL as the foundation for performance and power modeling on heterogeneous memory systems. While the proposed memory access pattern-aware metrics are subject to further improvements, they are still well-suited for comparing the instruction-level dynamic energy consumption during the execution of the same application on different memory technologies, for example, different DRAM generations and vendors or persistent memory architectures.

7. Conclusions & Future Work

This thesis presents an in-depth methodology for correctly measuring memory power consumption and an access-pattern-based analysis of dynamic energy consumption of heterogeneous memory systems.

RAPL's memory energy measurement accuracy was validated against physical measurements at the memory DIMMs for DRAM and PMem in chapter 4. The results show that RAPL significantly overestimates the memory modules' power consumption on the Ice Lake-SP platform. A likely reason for this is that the measurement point for the RAPL memory domain changed compared to previous architectures and now also includes power losses at the voltage regulator level. Measurements using the same methodology on the older Broadwell architecture, which was used for validating the correctness of the reference measurements, showed overall better agreement between both measurements, which aligns with results from the literature [10]. The changed measurement point would be beneficial for estimating a node's total power consumption but would make RAPL impractical for accurately measuring the power consumption of the memory modules. Due to a significant variance in the internal update rate, RAPL counters either need to be oversampled or sampled at a rate significantly lower than the advertised 1 kHz.

Chapter 5 presents the Dynamic Energy per Load (DEL) and Dynamic Energy per Store (DES) metrics for characterizing instruction-level dynamic energy consumption. Based on power measurements of single DRAM and PMem modules with different memory access patterns, the DEL and DES metrics, as well as the Bandwidth per Watt (BpW) metric, were computed. The results show that the access pattern significantly impacts energy consumption. While Intel Optane Persistent Memory (PMem) is four times more energy efficient per capacity compared to DRAM when idle, DRAM significantly outperforms PMem in terms of memory bandwidth and energy efficiency under load.

An access-pattern-based estimation model for dynamic energy consumption was proposed based on the DEL and DES metrics. Although designed for read- and write-only scenarios, these metrics were used to evaluate the energy consumption of real-world applications. As expected, the model showed higher dynamic energy consumption than the measured results. To ensure the model's accuracy on real-world applications, mixed-instruction workloads need to be evaluated. The metrics, however, can be used to estimate the upper bound of dynamic energy consumption.

7.1. Future Work

While RAPL is considered a reliable and accurate method for power measurements in the literature, this work motivates investigating the accuracy of RAPL further on modern systems and upcoming architectures. In particular, the accuracy of package power measurements and the accuracy of the memory domain for DDR5 memory, which is primarily powered using a single 5 V (UDIMM) or 12 V (RDIMM) line from the mainboard and has an on-DIMM Power-Management Integrated Circuit (PMIC) that delivers all required voltages and also measures power consumption [37, 38], are candidates for future work. Further experiments should also be carried out with fully populated memory channels.

Future work could also involve measuring and evaluating DEL and DES with different ratios of mixed instructions to better cover real-world applications and future memory architectures, e.g., the high-bandwidth memory (HBM) of Saphire Rapids CPUs, DDR5 memory, and Intel Optane PMem 300 series. Also, memories compatible with the emerging Compute Express Link (CXL) technology and, specifically, the `CXL.mem` protocol, which provides high-bandwidth and low latency access to volatile and persistent memory expansion devices in a cache-coherent way [86], are of interest here.

Acknowledgements

I want to thank Prof. Müller and Dr. Lankes for advising my thesis, as well as all the members of the Chair for High-performance Computing and the IT Center for providing access to compute resources and hardware and their help in troubleshooting technical problems. In particular, I want to thank Anara Kozhokanova for supervising me during my work on the thesis and for her constant support and valuable advice. This work probably would not have been possible without her input and help. A special thanks goes to my family and friends, who continuously supported me not only during the thesis but also throughout my studies. Simulations were performed with computing resources granted by RWTH Aachen University under project `thes1426`.

A. Additional Background Information

A.1. Memory Allocation on Heterogeneous Memory Architectures

With increasing diversity of memory architectures and topologies, developers are required to put more effort into programming the systems in order to benefit from them. A crucial part is allocating the memory correctly, e.g., allocating small and frequently required data on local memory with high bandwidth. However, if no special attention is put into this and memory is allocated using the plain `malloc` function from the standard library, the performance of the application will usually be unpredictable. In this section, we present three different methods how developers can control on which memory their data is allocated.

libnuma

libnuma [50] is a library that is available on most NUMA systems and that can be used easily in C/C++ programs by include the header (`#include <numa.h>`) and linking the library (`-lnuma`). Instead of using `malloc` from the C standard library, *libnuma* provides several methods to control on which NUMA node data should be allocated. The most important methods are:

- `numa_alloc_onnode(size, numa_node)` to allocate memory of `size` Bytes on the NUMA node `numa_node`.
- `numa_alloc_local(size)` to allocate memory of `size` Bytes on the local NUMA node of the thread invoking the memory.
- `numa_alloc_interleaved(size)` to allocate memory of `size` Bytes that is spread across all available NUMA nodes with page-size granularity.

All allocations using *libnuma* need to be explicitly freed using `numa_free(pointer, size)` instead of `free(pointer)` from stdlib. The library supports several others features, such as binding threads to NUMA nodes or getting information on the current NUMA structure. Readers can find detailed information on the interface in the official Linux manpage [50]. If modification of the program's source code is possible, *libnuma* is a good choice for programming heterogeneous memory systems as long as different memory types are configured as individual NUMA nodes.

numactl

`numactl` [51] is a command line tool for controlling NUMA placement policies and NUMA thread scheduling as well as inspecting NUMA topology of a system using `numactl -H`. Configuration done by `numactl` is propagated to all child processes. This tool is often the only option to control NUMA policies when modification of the application's source code is not possible. The most important features, that can also be combined, and, that we will use in thesis as well, are:

- `numactl -m <Nodes> -- <command>` to only allocate memory on the NUMA nodes specified by a comma-separated list when executing the provided command.
- `numactl -l -- <command>` to only allocate memory on the local NUMA node when executing the provided command.
- `numactl -n <Nodes> -- <command>` to only execute the specified command on specific NUMA nodes given by a comma-separated list.

Detailed information on the command usage can be found in the manual page [51].

memkind

memkind [71] is an advanced memory allocator that allows sophisticated control over the memory allocations. It supports a wide variety of memory technologies such as DRAM (see section 2.1.3), persistent memory like Intel Opane PMem (see section 2.1.4), and high-bandwidth memory like MCDRAM. As it is based on *jemalloc* [49], an implementation of malloc that aims to avoid memory fragmentation and improves scalable concurrency, an improved performance compared to allocations with *libnuma* can be expected but are not guaranteed. The library can be included using `#include <memkind.h>` and linked using `-lmemkind`.

In order to allocate memory using *memkind*, the type of memory needs to be defined by initializing a `memkind_t` object. This object is passed to all allocation calls and can be either created manually or one can be chosen out of large number of predefined kinds, such as:

- `MEMKIND_DEFAULT`: Allocate on default memory with default page size.
- `MEMKIND_HIGHEST_CAPACITY_PREFERRED`: Allocate on NUMA node with the highest capacity.
- `MEMKIND_HBW`: Allocate on the closest NUMA node with HBM.
- `MEMKIND_LOWEST_LATENCY_LOCAL`: Allocate on the local NUMA node with the lowest latency.
- `MEMKIND_DAX_KMEM`: Allocate on the closest persistent memory NUMA node.

Furthermore, the function `int memkind_create_pmem(const char *dir, size_t max_size, memkind_t *kind)` can be used to allocate memory backed by file located in `dir` in the file system. Similar to the standard library and *libnuma*, memory can than be allocated using `void *memkind_malloc(memkind_t kind, size_t size)` and freed using `void memkind_free(memkind_t kind, void *ptr)`

Another method for heterogeneous memory allocators, especially with OpenMP shared memory parallelization, is the memory allocation extension introduced in OpenMP 5.0 [11]. Out of the presented methods, *memkind* features the largest control of the memory access and the best performance but requires modification of the source code and is also more advanced than *libnuma*.

A.2. Performance Counters

Performance counters are special purpose hardware registers that get incremented when a specific event occurs. Popular examples of such events are the number of cache misses, the number of cache accesses, or the number of floating-point operations.

perf_event

perf, often also called *perf_event*, is a lightweight command-line profiling tool for performance counters that is part of the Linux kernel tools. Apart from other features, the `perf stat` command can be used to sample the performance counters over time, for example during the execution of a program. It can be configured to report the counter values at a fixed sampling interval of up to 1 ms or aggregated over the complete execution of the program. *perf* does not only implement hardware performance counters but also other software counter values or system sensor data. Access to energy measurements provided through RAPL is also supported. The following command can be used to sample the RAPL energy counters every 500ms for the package and memory domains and report it for all sockets independently during the execution of the STREAM benchmark.

```
perf stat -e power/energy-pkg/,power/energy-ram/ -I500 -per-socket
./stream
```

With the `-summary` option, the aggregated energy consumption during the execution of the entire application is reported at the end. Detailed information on how to use the *perf* tool can be found in the manual page [52].

PAPI

PAPI (Performance Application Programming Interface) is an API for efficient access to hardware performance counters on modern microprocessors and monitoring system information, e.g., network cards or GPUs. It provides both a low-level and

a high-level API, each designed for different use cases and enabling greater flexibility. For example, using the high-level API provided by PAPI, the programmers can mark a "measurement region" identified by a name in the code using `PAPI_hl_region_begin(const char*)` and `PAPI_hl_region_end(const char*)` before and after regions. PAPI measures performance counters at the start and end of the region and computes the difference across the region. The performance counters are represented as named events in PAPI, for example "PAPI_TOT_INS" for the number of CPU instructions processed or "rapl:::DRAM_ENERGY:PACKAGE0" for the memory energy consumed by socket 0 according to RAPL. The events PAPI tracks can be set using the `PAPI_EVENTS` environment variable. PAPI is organized in a modular fashion and can be extended by new components. For example, a component for directly accessing RAPL measurements and the `powercap` component, which also exposes RAPL measurements, can be used for accessing RAPL energy counters. The `rapl` component missed Ice Lake-SP support up until PAPI 7.0.1. After submitting a pull request that fixes this problem to the project, PAPI can be used for accesing RAPL measurements [59, 102].

B. Additional Material for RAPL Validation

B.1. Experimental Setup

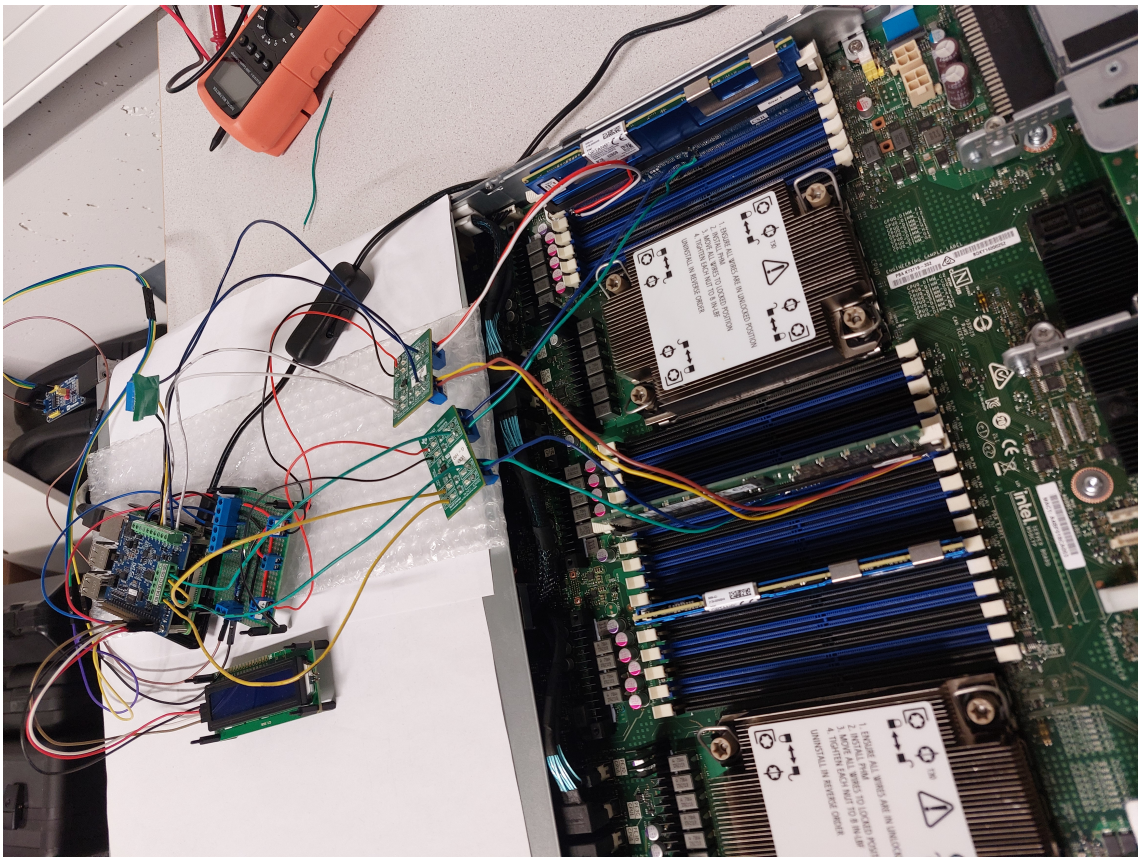


Figure B.1.: Hardware Instrumentation of the Ice Lake system with a DRAM (slot A1) and PMem (slot E1) module mounted using a riser for current-sensing in socket 0. At the time this photo was taken, the risers had only probing wires attached to two pins. A photo with three pairs of wires is shown on the next page.

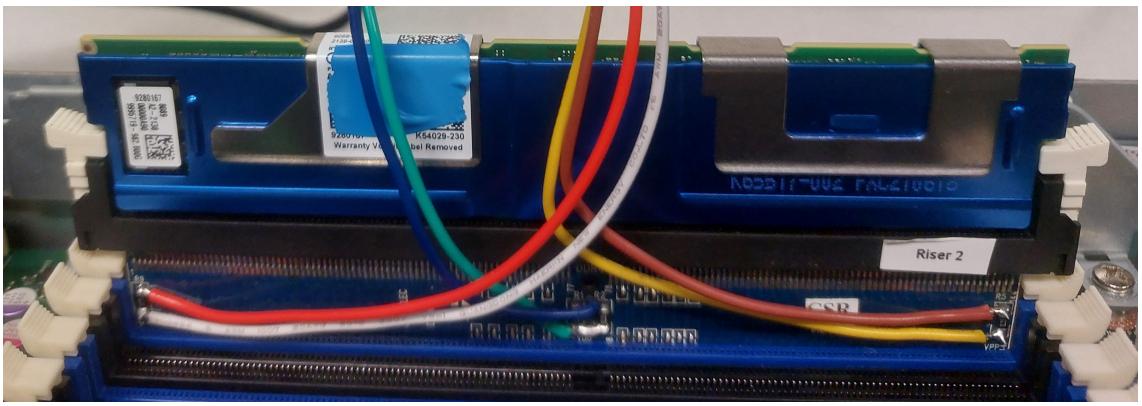


Figure B.2.: Installed riser with a PMem module. Probing wires have been soldered to the `12V` (left), `VDD` (middle, two resistors in parallel), and `VPP` (right) current-sense resistors. `VPP` is not measured for PMem. The blue sticker is used to identify the module.

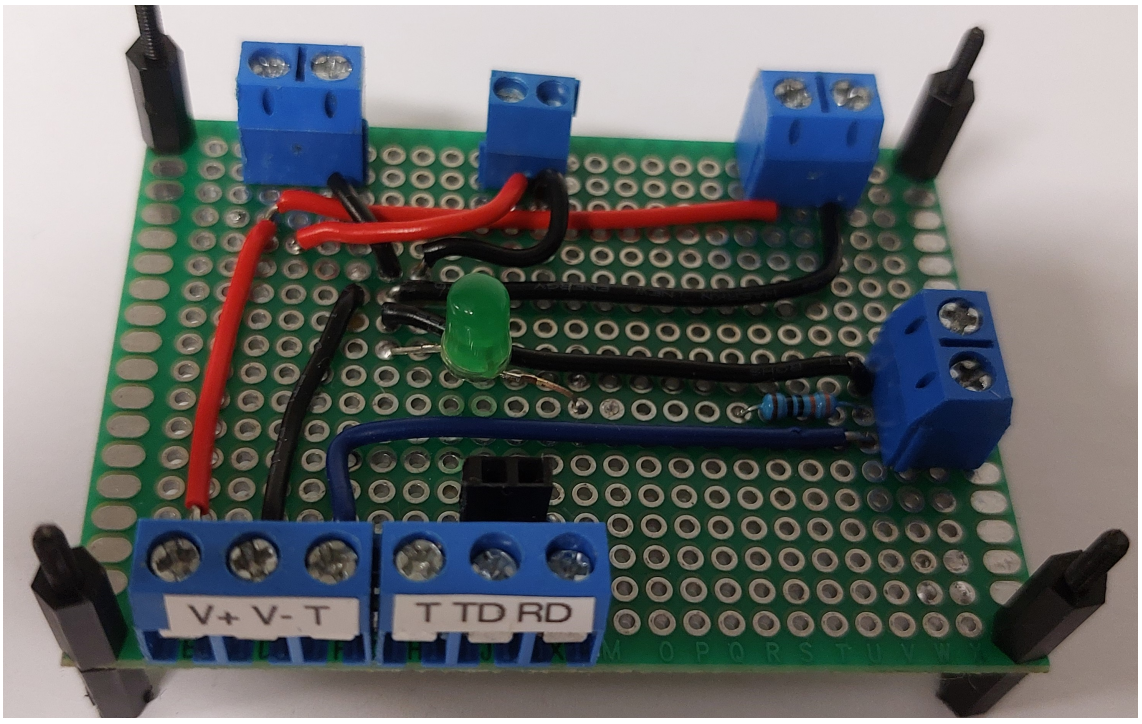


Figure B.3.: Custom through-hole board for distributing power and ground to the different components. The LED turns on when the trigger signal is active.

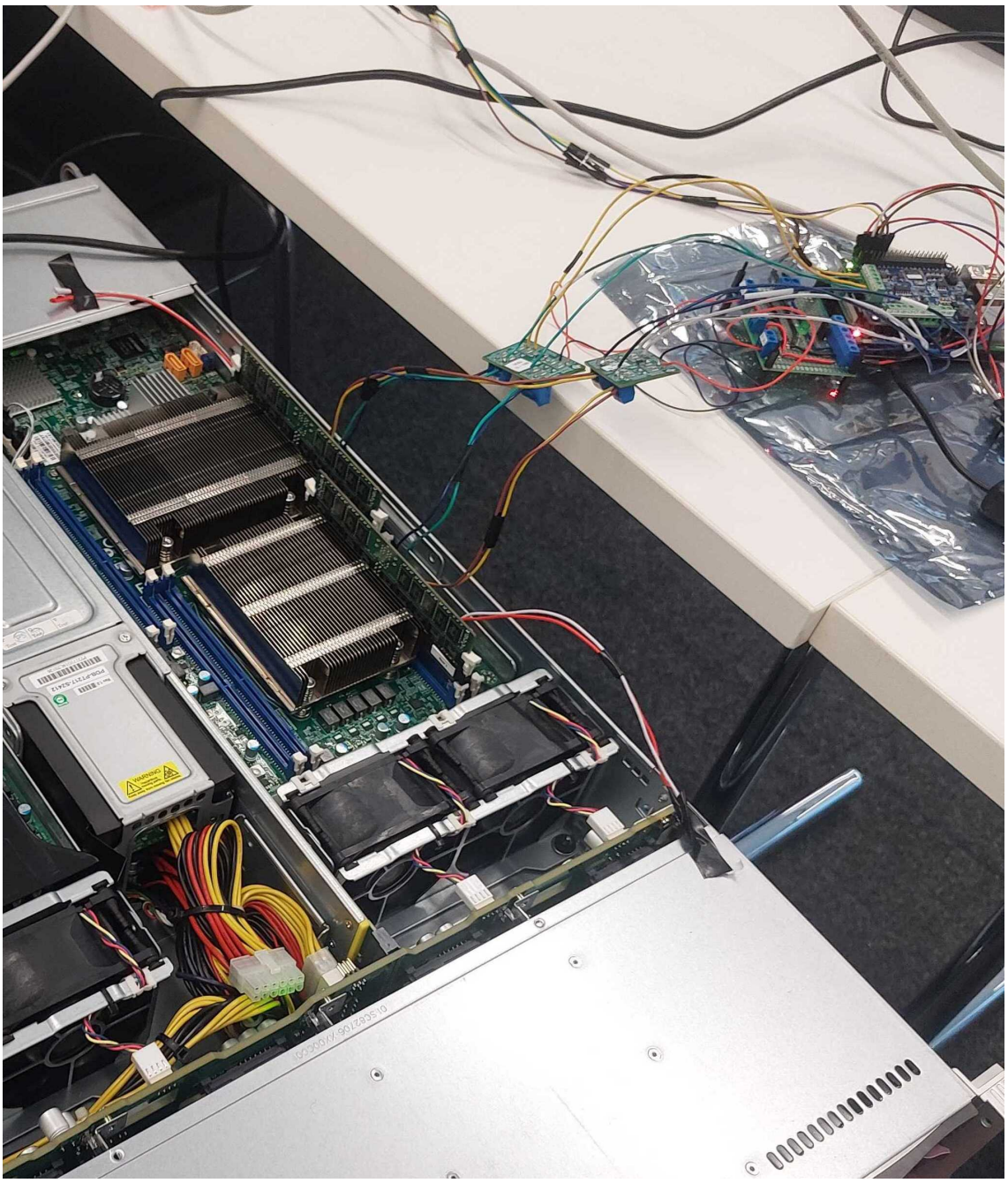


Figure B.4.: Hardware Instrumentation of the Broadwell system. The red and white wires are connected to `12V` of the riser and are not used here.

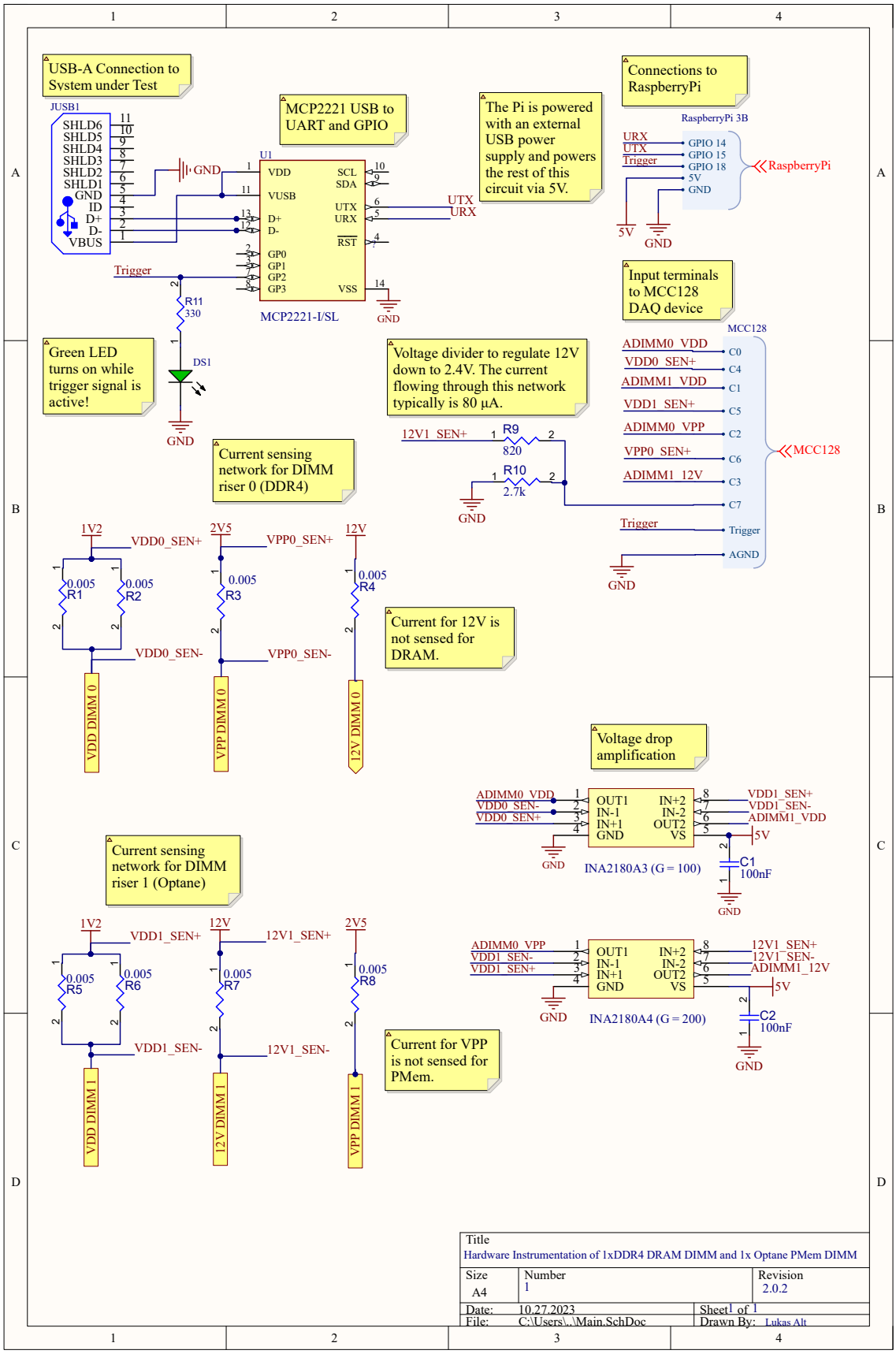


Figure B.5.: Schematic for current-sensing circuit with two DIMMs (1xDRAM, 1xP-Mem)

Name	VR	Sensing	Use Case	Notes
Infineon PXE1110CDN [25]	12.5/13	In/Out	DDR3/DDR4	Temperature compensation of current-sensing and oversampling
Infineon PXE1610CDN [25]	12.5/13	In/Out	DDR3/DDR4	
TI TPS544C26 [94]	13	In/Out	DDR5	Power measurements calibration supported
Analog MAX16602 [57]	13.HC	In/Out	Server CPUs	Max. Input sensing accuracy $\pm 0.8\%$
MPS MP2935 [64]	12.5	Out	CPU	End of life
MPS MP2965 [65]	13.HC	In/Out	CPU/GPU cores	Input sensing accuracy $\pm 1\%$ at $0.1mA$
Richtek RT8167A [75]	12	Out	Laptop/Desktop CPU supply	IMON update interval 1.6ms
Renesas ISL69133 [73]	13	In/Out	Server Memory/CPU supply	"Supports external input current sense required for NVDIMM"

Table B.1.: Different voltage regulators along with supported VR standards and current sensing features

```
#include <unistd.h>
#include <stdio.h>
#include <fcntl.h>
#include <string.h>
#include <stdlib.h>
#include <assert.h>
#define NR_MEASUREMENTS 10000
static long long cycles[NR_MEASUREMENTS+1];
#define MSR_DRAM 0x619
#define MSR_PACKAGE 0x611
static long long MSR=MSR_PACKAGE;

#define sync_rdtsc2(val) \
   do {\
      unsigned int cycles_low, cycles_high;\
      asm volatile("RDTSCP\n\t"\
      "mov %%edx, %0\n\t"\
      "mov %%eax, %1\n\t"\
      "CPUID\n\t": "=r" (cycles_high), "=r" (cycles_low):: "%rax", "%rbx", "%rcx", "%rdx");\
      (val) = ((unsigned long) cycles_low) | (((unsigned long) cycles_high) << 32);\
   } while (0)

void main() {
        long long start_cyc = 0, end_cyc = 0;
        sync_rdtsc2(start_cyc);
        sleep(1);
        sync_rdtsc2(end_cyc);
        printf("Cycles per second: %d\n", end_cyc - start_cyc);
        long long status;
        long long cur_status;
        char buffer[256];
        sprintf(buffer,"/dev/cpu/0/msr");
        int fd = open(buffer,O_RDWR);
        pread(fd,&status,8,MSR);
        sync_rdtsc2(cycles[0]);
        for (int i=1; i<NR_MEASUREMENTS; i++) {
                do {
                        pread(fd,&cur_status,8,MSR);
                } while(cur_status==status);
                status=cur_status;
                sync_rdtsc2(cycles[i]);
        }

        for (int i=2 ; i<NR_MEASUREMENTS-1; i++) {
                printf("%lli\n",cycles[i]-cycles[i-1]);
        }

}
```

Listing B.1: Adopted RAPL update interval measurement code [82]

B.2. Experiments Reproduction Guide

In this section, we describe how the experimental setup and the measurements for the RAPL validation (see chapter 4) and metrics (see chapter 5) can be reproduced. First, the DIMM risers need to be installed in the designated memory slots, and the MCC128 needs to be mounted on the Pi while following applicable ESD guidelines. Then, all wiring between the risers, the current-sense amplifiers, the MCC128, and the MCP2221 need to be established according to figure B.5. Some photos of the setup can be found in figures 4.5 and B.1.

Software Setup on the Raspberry Pi

The following steps need to be followed to set up a Raspberry Pi 3B or later for power and energy measurements.

- Power the Pi using a recommended power supply. We used the official one. Powering via a standard USB port usually does not provide enough current and can lead to an undervoltage.
- Install any Linux installation on the Pi. We used Raspberry Pi OS for this thesis.
- Install python3, e.g., `sudo apt install python3 python3-dev python3-pip` and `sudo apt install screen`
- Install required python packages: `pip install pandas numpy pyserial RPi.GPIO`
- Copy the `sourcecode/rapl-validation/power-collector/` folder from the thesis artifacts to the Pi.
- To capture power consumption traces, run `screen -dmS power-measurements python3 power-collector.py`. If the accumulated energy (at a higher internal sampling rate) should be measured only, run `screen -dmS power-measurements python3 power-collector.py` instead.
- Both scripts require a configuration containing which voltages and signals from which DIMM are connected to the input pins of the data acquisition device. Different profiles can be configured in `power-collector/scripts/channel_config.py`. The profile can then be configured in the global variable `channel_config` in the collector scripts.
- The sampling frequency can be controlled by setting the `scan_rate` variable, which defines the number of samples per second, in the collector scripts.
- Both scripts act as an agent and do not need to be explicitly controlled on the Pi once they are started.

Software Setup on the Machine under Test

The following steps need to be followed to set up the host system for lining up the power measurements taken by the Pi with the execution of applications on the host system.

- Ensure a Linux installation with a recent kernel is available and that root access is available.
- Install `numactl`, `libnuma`, `python3`, `python3-dev`, `python3-devel`, `hidapi`. All installed `RHEL 8` packages can be found in `data/system-info/yum_packages.txt`.
- Install the python packages `pip install numpy pandas PyMCP2221A hidapi==0.14.0 libusb pyserial`. In our case, the installation of `PyMCP2221A` was challenging and required a couple of other dependencies. A list of all installed Python packages can be found in `data/system-info/pip_packages.txt`.
- Copy the `sourcecode/rapl-trace/` folder from the thesis artifacts and build it using `make`.
- Copy the `sourcecode/rapl-validation/` folder from the artifacts to the system as well.
- Switch to the `workloads` folder and run `make`.
- Update the path to the `workloads` folder in the `workloads/run.sh` script.
- Run the `measure-all-optane.sh` and `measure-all-dram.sh` scripts to run the RAPL validation experiments we have evaluated in the thesis for PMem and DRAM, respectively.
- Run `python3 measure-power-rapl.py` to collect RAPL measurements and reference measurements of the specified workload. A description of the parameters can be requested by specifying the `-h` flag.
- Run `python3 measure-energy.py` to measure the aggregated memory energy consumption during the execution of the provided application. Again, a description of the parameters can be requested using the `-h` flag. If the `-T` flag is set, the delegate applications need to implement the measurement instrumentation (see listing 5.1) itself. This allows measuring the memory energy consumption of only a specific part of the code. The energy consumption, the execution time, and the average power consumption are printed to the standard output afterward.

The RAPL validation scripts will create a folder for each workload with the reference measurements on the Pi and the RAPL measurements on the system under test. For further processing, both folders need to be merged, and a `config.json` file that contains the mapping of the different voltage channels to the pins and risers needs to be created. The existing configs in the `data/rapl-validation/icelake` folder serve as a baseline. A Jupyter notebook that creates the scatter plots (e.g., figure 4.7c) and phase plots (e.g., figure 4.9) from this raw data can be found in `data/rapl-validation/Visualize RAPL Results.ipynb`.

Required Changes for DDR5 Measurements

Our setup can be modified to measure the power consumption of DDR5 DIMMs. As the DDR5 interface is incompatible with DDR4, our riser cards do not work here, and DDR5-compatible alternatives can be used. Currently, we only found the DDR5-R riser from Adex Electronics[1], which, however, only supports 4800 MT/s memory. Also, validate that the current-sense resistors are rated for the voltage and current through them (see section 4.3.2). If too much heat is dissipated at the shunt, the riser, the memory module, or even the mainboard can be damaged. Registered DDR5 DIMMs are powered using a single 12 V supply, and unbuffered DDR5 memory is powered using 5 V. Thus, only a single current-sense amplifier (ideally with a gain >= 200) and two data-acquisition channels are required for measuring the power consumption of a single DDR5 DIMM. If the bus voltage (e.g., 12V for RDIMMs) exceeds the maximum input voltage of the DAQ device, a voltage divider (see figure B.5) is required to step down the voltage to fit into the supported range.

IT Setup and Configuration

Due to the required space for the hardware instrumentation, the server could not be operated in the rack in the server room. Thus, we moved it to another room where we set up our measurements. As no network connection to the cluster infrastructure was available in this room, a local network using an ethernet switch was set up. The Pi was used to bridge WiFi to this local network, and we connected the server to the switch via a USB to Ethernet adapter as the Ice Lake system did not have an RJ45 public network port. The management port was also connected to the switch to access the Intelligent Platform Management Interface (IPMI).

Because the server was not connected to the cluster anymore, the distributed file system and the local operating system of the server could not be used. Thus, we had to install our own operating system, which did not work at first as the server did not detect any USB boot devices. After installing Rocky Linux on an old SATA SSD and connecting this SSD via a SATA to USB adapter to the server, the system

[1] `https://adexelec.com/ddr5-r`

correctly detected it and booted successfully. We installed PAPI v7.0.1 libnuma v2.0.16, and libmemkind v1.14.0 from sources.

Do not hesitate to contact the author if questions or problems arise during the reproduction of the experiments.

C. Dynamic Energy per Load and Store Data

Threads	DEL [nJ] DRAM		DEL [nJ] PMem		DES [nJ] DRAM		DES [nJ] PMem	
1	1.05	[1.04;1.06]	10.38	[10.35;10.40]	2.39	[2.34;2.44]	97.18	[95.46;98.90]
2	0.94	[0.93;0.96]	9.43	[9.42;9.45]	2.36	[2.34;2.38]	88.3	[87.78;88.81]
3	0.88	[0.88;0.88]	9.12	[9.10;9.15]	2.29	[2.28;2.30]	81.86	[81.47;82.24]
4	0.83	[0.82;0.83]	9.03	[9.02;9.05]	2.26	[2.26;2.27]	82.79	[82.46;83.11]
5	0.79	[0.78;0.79]	9.04	[9.02;9.06]	2.26	[2.24;2.29]	83.02	[82.33;83.71]
6	0.75	[0.75;0.75]	9.04	[9.04;9.05]	2.29	[2.25;2.32]	83.21	[83.00;83.42]
7	0.71	[0.68;0.73]	9.02	[8.98;9.06]	2.34	[2.32;2.35]	83.62	[83.33;83.90]
8	0.69	[0.69;0.70]	8.99	[8.98;9.01]	2.34	[2.32;2.37]	84.82	[84.43;85.22]
9	0.7	[0.69;0.70]	9.0	[8.99;9.02]	2.35	[2.35;2.36]	85.85	[84.95;86.76]
10	0.7	[0.70;0.71]	9.02	[9.00;9.04]	2.36	[2.36;2.37]	85.18	[83.92;86.45]
11	0.71	[0.71;0.72]	8.73	[8.51;8.94]	2.35	[2.29;2.41]	85.71	[84.15;87.28]
12	0.71	[0.69;0.74]	9.01	[8.96;9.07]	2.38	[2.37;2.39]	87.67	[85.92;89.43]
13	0.74	[0.74;0.75]	9.11	[9.07;9.15]	2.39	[2.38;2.41]	87.84	[86.81;88.87]
14	0.76	[0.76;0.77]	9.27	[9.24;9.31]	2.4	[2.39;2.41]	88.73	[88.12;89.35]
15	0.77	[0.77;0.78]	9.49	[9.47;9.52]	2.38	[2.32;2.44]	88.75	[87.72;89.79]
16	0.79	[0.78;0.80]	9.68	[9.66;9.70]	2.41	[2.39;2.42]	88.18	[85.90;90.45]
17	0.8	[0.80;0.81]	9.92	[9.90;9.94]	2.41	[2.40;2.42]	90.07	[89.44;90.71]
18	0.82	[0.81;0.82]	10.12	[10.10;10.13]	2.42	[2.41;2.43]	90.9	[89.02;92.79]
19	0.83	[0.83;0.83]	10.23	[10.18;10.27]	2.42	[2.40;2.43]	92.09	[90.82;93.35]
20	0.83	[0.83;0.84]	10.35	[10.31;10.39]	2.41	[2.37;2.44]	92.16	[91.68;92.63]
21	0.85	[0.84;0.85]	10.41	[10.39;10.43]	2.42	[2.40;2.44]	93.15	[90.94;95.36]
22	0.85	[0.85;0.86]	10.42	[10.31;10.53]	2.42	[2.41;2.43]	90.61	[87.47;93.74]
23	0.86	[0.85;0.86]	10.55	[10.51;10.59]	2.43	[2.42;2.43]	92.6	[91.92;93.28]
24	0.86	[0.86;0.86]	10.6	[10.59;10.62]	2.39	[2.34;2.44]	90.75	[88.47;93.03]
25	0.86	[0.85;0.87]	10.64	[10.61;10.68]	2.41	[2.40;2.43]	93.03	[91.80;94.26]
26	0.87	[0.87;0.87]	10.61	[10.58;10.65]	2.41	[2.41;2.42]	90.78	[87.48;94.09]
27	0.87	[0.87;0.87]	10.63	[10.58;10.68]	2.4	[2.39;2.41]	89.94	[89.80;90.08]
28	0.87	[0.87;0.87]	10.55	[10.52;10.59]	2.4	[2.39;2.41]	89.88	[89.32;90.44]
29	0.87	[0.86;0.87]	10.53	[10.50;10.56]	2.37	[2.33;2.41]	90.37	[88.90;91.85]
30	0.86	[0.86;0.86]	10.35	[10.28;10.41]	2.38	[2.37;2.39]	89.58	[89.26;89.90]
31	0.86	[0.86;0.86]	10.48	[10.41;10.55]	2.36	[2.35;2.37]	90.18	[89.57;90.79]
32	0.85	[0.84;0.85]	10.37	[10.31;10.42]	2.33	[2.32;2.34]	88.85	[85.97;91.72]

Table C.1.: Dynamic Energy per Load (DEL) and Dynamic Energy per Store (DES) for sequential access to DRAM and PMem with 95% asymptotic confidence intervals. Mean of 3 reptitions.

Stride	DEL [nJ]				DES [nJ]			
[Bytes]	**DRAM**		**PMem**		**DRAM**		**PMem**	
8	1.01	[0.98;1.04]	10.39	[10.36;10.41]	2.39	[2.34;2.44]	97.16	[97.11;97.21]
16	1.61	[1.57;1.65]	19.94	[19.89;19.98]	1.31	[1.29;1.32]	193.62	[192.75;194.48]
24	2.26	[2.24;2.27]	29.61	[29.50;29.73]	2.70	[2.65;2.75]	290.85	[289.16;292.53]
32	2.98	[2.95;3.00]	39.13	[39.01;39.26]	4.18	[4.15;4.21]	388.52	[383.90;393.13]
40	3.64	[3.54;3.73]	47.92	[46.57;49.27]	5.42	[5.37;5.47]	486.05	[484.09;488.01]
48	4.37	[4.34;4.41]	58.15	[58.00;58.29]	6.68	[6.60;6.75]	587.01	[581.68;592.34]
56	5.04	[4.97;5.11]	67.48	[67.25;67.71]	8.13	[8.11;8.15]	681.35	[678.85;683.85]
64	5.76	[5.74;5.78]	76.47	[75.48;77.47]	9.66	[9.61;9.71]	781.02	[775.80;786.23]
80	7.01	[6.95;7.07]	94.71	[91.86;97.56]	10.47	[10.44;10.50]	795.38	[792.41;798.35]
96	7.33	[7.26;7.41]	112.32	[109.10;115.54]	11.49	[11.45;11.52]	805.51	[796.84;814.18]
112	8.11	[8.02;8.20]	131.08	[130.80;131.37]	12.77	[12.70;12.84]	816.78	[811.14;822.43]
128	6.79	[6.69;6.90]	139.40	[138.94;139.85]	12.31	[11.76;12.86]	848.52	[845.85;851.18]
144	8.28	[8.10;8.45]	165.07	[164.81;165.33]	13.68	[13.47;13.89]	817.29	[809.32;825.27]
160	7.80	[7.74;7.87]	179.64	[179.47;179.82]	13.32	[13.02;13.63]	807.12	[794.23;820.01]
176	8.70	[8.61;8.79]	195.29	[194.74;195.83]	11.73	[11.13;12.32]	886.54	[876.17;896.91]
192	7.25	[7.06;7.44]	204.63	[204.09;205.17]	12.43	[12.37;12.48]	770.82	[756.80;784.85]
208	7.66	[7.47;7.84]	217.32	[217.26;217.37]	12.33	[12.20;12.47]	742.06	[708.16;775.96]
224	8.19	[8.04;8.34]	232.74	[231.41;234.06]	13.10	[13.01;13.18]	843.95	[813.75;874.15]
240	9.08	[8.79;9.36]	244.73	[238.27;251.18]	11.96	[11.68;12.24]	966.11	[961.45;970.77]
256	7.56	[7.53;7.59]	243.61	[239.00;248.22]	8.45	[8.16;8.73]	889.76	[868.68;910.84]
272	7.98	[7.90;8.06]	253.43	[246.35;260.52]	10.21	[10.00;10.42]	821.57	[817.74;825.39]

Table C.2.: DEL and DES for strided access with 1 thread to DRAM and PMem with 95% asymptotic confidence intervals. Mean of 3 reptitions.

Stride [Bytes]	DEL [nJ] DRAM		DEL [nJ] PMem		DES [nJ] DRAM		DES [nJ] PMem	
8	0.81	[0.81;0.82]	8.79	[8.54;9.04]	1.41	[1.41;1.41]	84.97	[84.53;85.41]
16	1.32	[1.32;1.32]	17.72	[17.11;18.33]	2.93	[2.91;2.95]	170.31	[169.88;170.73]
24	1.91	[1.86;1.97]	26.61	[25.83;27.39]	4.46	[4.46;4.46]	251.47	[245.36;257.59]
32	2.57	[2.56;2.59]	35.61	[34.69;36.53]	5.99	[5.96;6.02]	341.82	[340.58;343.06]
40	3.23	[3.15;3.31]	44.54	[43.55;45.54]	7.48	[7.31;7.65]	428.82	[428.21;429.44]
48	3.83	[3.78;3.88]	53.16	[51.54;54.79]	9.18	[9.15;9.21]	512.43	[512.19;512.67]
56	4.51	[4.41;4.60]	61.94	[60.14;63.73]	10.71	[10.65;10.77]	599.55	[597.17;601.94]
64	5.28	[4.92;5.65]	71.04	[69.23;72.85]	12.20	[12.17;12.22]	684.75	[681.48;688.02]
80	6.38	[6.36;6.41]	89.99	[89.40;90.58]	13.02	[12.96;13.08]	679.84	[677.62;682.06]
96	7.44	[7.37;7.52]	107.85	[107.60;108.11]	13.80	[13.76;13.83]	659.12	[653.22;665.03]
112	8.31	[8.29;8.33]	125.41	[125.07;125.75]	14.39	[13.82;14.96]	627.25	[622.68;631.81]
128	6.30	[6.29;6.32]	140.96	[140.80;141.13]	15.41	[15.39;15.43]	755.00	[751.69;758.32]
144	9.37	[9.32;9.42]	156.48	[155.97;157.00]	15.60	[15.46;15.74]	497.59	[492.19;502.99]
160	9.64	[9.61;9.66]	170.55	[170.14;170.96]	15.71	[15.63;15.79]	410.97	[401.26;420.67]
176	10.41	[10.38;10.45]	184.42	[183.85;184.99]	17.42	[17.32;17.53]	338.89	[332.40;345.39]
192	9.32	[9.27;9.37]	191.15	[190.24;192.05]	15.23	[15.19;15.27]	271.37	[268.82;273.91]
208	9.35	[9.06;9.64]	201.84	[201.03;202.65]	14.96	[14.86;15.06]	233.71	[229.54;237.87]
224	10.28	[10.24;10.32]	214.59	[214.17;215.01]	15.96	[15.87;16.05]	205.83	[185.32;226.34]
240	10.84	[10.81;10.88]	220.53	[218.49;222.57]	16.91	[16.87;16.95]	200.39	[183.60;217.17]
256	7.08	[7.03;7.13]	253.25	[252.02;254.47]	18.49	[18.45;18.53]	841.51	[838.57;844.46]
272	8.49	[8.44;8.54]	209.14	[202.80;215.49]	11.05	[10.86;11.24]	169.40	[165.92;172.88]

Table C.3.: DEL and DES for strided access with 4 threads to DRAM and PMem with 95% asymptotic confidence intervals. Mean of 3 reptitions.

Stride [Bytes]	DEL [nJ] DRAM		DEL [nJ] PMem		DES [nJ] DRAM		DES [nJ] PMem	
8	0.69	[0.69;0.69]	1.56	[1.54;1.59]	9.00	[8.98;9.02]	85.99	[84.70;87.27]
16	1.33	[1.33;1.34]	3.14	[3.14;3.14]	17.95	[17.86;18.03]	173.65	[172.55;174.76]
24	1.98	[1.97;2.00]	4.68	[4.67;4.70]	26.89	[26.71;27.06]	260.70	[260.31;261.09]
32	2.64	[2.64;2.65]	6.29	[6.28;6.30]	35.53	[34.58;36.47]	349.27	[347.29;351.25]
40	3.25	[3.17;3.34]	7.90	[7.86;7.93]	44.81	[44.60;45.01]	436.63	[435.44;437.83]
48	4.02	[4.00;4.04]	9.56	[9.51;9.61]	53.87	[53.61;54.13]	524.65	[523.84;525.46]
56	4.73	[4.69;4.76]	11.11	[11.00;11.21]	63.00	[62.79;63.20]	612.63	[609.72;615.54]
64	5.41	[5.38;5.44]	12.85	[12.80;12.90]	72.23	[71.96;72.49]	700.45	[699.65;701.25]
80	6.74	[6.73;6.76]	13.53	[13.47;13.59]	90.43	[90.21;90.66]	693.51	[689.95;697.07]
96	7.89	[7.87;7.91]	14.22	[14.19;14.25]	107.62	[104.68;110.56]	667.94	[666.54;669.34]
112	8.58	[8.55;8.62]	14.92	[14.85;14.98]	124.58	[120.74;128.41]	624.27	[621.46;627.07]
128	9.13	[8.81;9.45]	16.31	[16.30;16.32]	142.15	[137.65;146.64]	767.70	[763.71;771.69]
144	9.09	[9.09;9.10]	15.26	[15.20;15.32]	151.64	[148.00;155.27]	483.28	[480.94;485.62]
160	9.23	[9.17;9.28]	14.78	[14.44;15.12]	159.04	[153.72;164.36]	395.75	[386.69;404.81]
176	9.75	[9.67;9.83]	16.71	[16.59;16.82]	163.55	[159.04;168.07]	299.64	[289.42;309.86]
192	9.15	[9.09;9.22]	13.95	[13.93;13.97]	165.47	[164.91;166.03]	237.62	[232.30;242.95]
208	9.38	[9.30;9.45]	13.51	[13.24;13.79]	173.47	[171.60;175.33]	187.04	[175.12;198.96]
224	9.63	[9.57;9.68]	14.12	[13.85;14.38]	173.96	[169.08;178.83]	166.47	[152.51;180.43]
240	8.86	[8.39;9.33]	13.66	[13.46;13.87]	160.77	[155.70;165.85]	169.46	[147.94;190.98]
256	10.27	[10.14;10.40]	19.78	[19.77;19.78]	267.86	[267.66;268.06]	854.75	[850.98;858.52]
272	6.53	[6.21;6.86]	6.81	[6.26;7.36]	116.28	[90.30;142.26]	121.91	[111.89;131.93]

Table C.4.: DEL and DES for strided access with 8 threads to DRAM and PMem with 95% asymptotic confidence intervals. Mean of 3 reptitions.

Stride [Bytes]	DEL [nJ] DRAM		DEL [nJ] PMem		DES [nJ] DRAM		DES [nJ] PMem	
8	0.78	[0.78;0.79]	9.71	[9.69;9.72]	1.66	[1.62;1.70]	88.58	[86.42;90.74]
16	1.57	[1.57;1.57]	19.95	[19.92;19.98]	3.36	[3.36;3.37]	178.98	[178.11;179.84]
24	2.34	[2.33;2.35]	30.33	[30.28;30.38]	5.02	[5.00;5.03]	267.91	[266.37;269.44]
32	3.13	[3.11;3.15]	40.73	[40.58;40.87]	6.75	[6.72;6.78]	355.69	[354.78;356.60]
40	3.93	[3.91;3.94]	51.20	[51.13;51.27]	8.48	[8.46;8.51]	453.55	[444.13;462.96]
48	4.73	[4.71;4.75]	61.48	[61.37;61.59]	10.20	[10.16;10.24]	540.42	[537.36;543.48]
56	5.56	[5.55;5.57]	72.60	[72.43;72.76]	11.89	[11.85;11.94]	622.07	[617.30;626.85]
64	6.34	[6.31;6.37]	83.00	[82.71;83.28]	13.63	[13.57;13.69]	728.45	[723.45;733.46]
80	7.86	[7.81;7.91]	104.39	[104.20;104.57]	14.14	[14.09;14.20]	707.91	[706.70;709.13]
96	9.20	[9.18;9.22]	124.66	[124.01;125.31]	14.60	[14.51;14.70]	674.89	[650.25;699.54]
112	9.65	[9.58;9.71]	139.65	[138.61;140.69]	14.86	[14.77;14.95]	627.32	[624.99;629.66]
128	9.84	[9.51;10.16]	159.05	[158.80;159.30]	17.42	[17.38;17.46]	792.50	[789.83;795.18]
144	8.96	[8.91;9.01]	146.15	[145.15;147.15]	13.03	[13.00;13.06]	485.12	[464.54;505.70]
160	8.30	[8.24;8.36]	136.50	[135.21;137.79]	11.47	[11.31;11.62]	336.49	[281.29;391.69]
176	7.85	[7.84;7.86]	118.80	[116.87;120.73]	10.82	[10.58;11.06]	199.03	[181.37;216.69]
192	6.64	[6.35;6.93]	95.01	[89.38;100.63]	8.49	[8.19;8.79]	131.05	[117.99;144.12]
208	5.95	[5.82;6.08]	90.13	[84.15;96.11]	6.33	[6.04;6.61]	105.01	[101.75;108.28]
224	4.39	[4.10;4.68]	67.78	[62.69;72.87]	4.67	[3.82;5.52]	84.02	[82.53;85.52]
240	3.54	[3.38;3.71]	50.87	[49.13;52.61]	3.49	[3.34;3.64]	77.67	[74.01;81.34]
256	13.13	[13.04;13.22]	259.20	[258.17;260.22]	21.27	[21.22;21.31]	878.24	[876.31;880.16]
272	2.57	[2.47;2.66]	36.46	[36.05;36.87]	2.67	[2.46;2.88]	56.89	[51.37;62.40]

Table C.5.: DEL and DES for strided access with 16 threads to DRAM and PMem with 95% asymptotic confidence intervals. Mean of 3 reptitions.

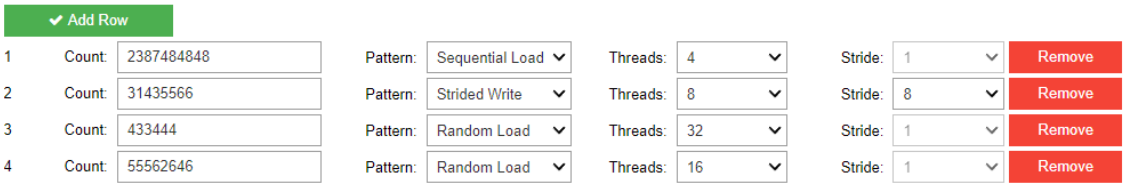
Stride [Bytes]	DEL [nJ] DRAM		DEL [nJ] PMem		DES [nJ] DRAM		DES [nJ] PMem	
8	0.85	[0.84;0.85]	10.33	[10.25;10.42]	1.68	[1.67;1.68]	89.14	[88.31;89.97]
16	1.59	[1.58;1.59]	21.01	[20.81;21.21]	3.22	[3.20;3.23]	177.23	[172.09;182.37]
24	2.35	[2.34;2.37]	31.46	[31.19;31.73]	4.77	[4.75;4.79]	271.65	[265.98;277.33]
32	3.07	[2.99;3.15]	42.12	[41.65;42.59]	6.42	[6.42;6.42]	360.86	[359.88;361.83]
40	3.93	[3.92;3.95]	52.46	[52.15;52.77]	8.06	[8.03;8.09]	452.17	[445.60;458.74]
48	4.80	[4.77;4.82]	63.20	[63.02;63.38]	9.54	[9.21;9.87]	547.15	[535.94;558.35]
56	5.66	[5.63;5.70]	73.40	[73.05;73.76]	11.40	[11.35;11.44]	624.76	[623.81;625.72]
64	6.54	[6.50;6.58]	84.20	[83.60;84.81]	13.21	[13.18;13.24]	721.89	[706.97;736.81]
80	8.08	[8.02;8.14]	106.66	[106.43;106.89]	13.21	[13.17;13.25]	704.31	[698.11;710.51]
96	8.70	[8.62;8.79]	123.26	[122.93;123.58]	12.74	[12.70;12.78]	660.97	[653.42;668.52]
112	8.35	[8.19;8.51]	126.73	[125.58;127.87]	11.27	[11.15;11.38]	607.56	[591.82;623.31]
128	9.27	[9.18;9.36]	151.34	[150.72;151.96]	16.89	[16.85;16.93]	779.19	[774.35;784.02]
144	5.52	[5.46;5.58]	81.19	[76.44;85.93]	4.19	[3.79;4.59]	217.76	[55.93;379.59]
160	4.15	[3.89;4.41]	64.63	[62.32;66.94]	2.74	[2.55;2.94]	95.61	[90.36;100.87]
176	3.29	[3.19;3.38]	49.45	[44.99;53.90]	2.30	[2.20;2.40]	78.53	[73.31;83.75]
192	2.30	[2.21;2.38]	39.98	[36.64;43.31]	2.17	[2.02;2.31]	60.21	[57.56;62.86]
208	2.02	[1.97;2.07]	35.55	[33.69;37.40]	1.80	[1.77;1.83]	54.25	[45.39;63.11]
224	1.76	[1.72;1.79]	31.83	[30.73;32.93]	1.59	[1.51;1.67]	41.51	[40.88;42.14]
240	1.60	[1.58;1.61]	29.42	[28.89;29.96]	1.29	[1.24;1.34]	35.17	[34.11;36.23]
256	9.97	[9.90;10.04]	218.03	[215.41;220.66]	19.98	[19.90;20.06]	875.95	[869.31;882.59]
272	0.54	[0.49;0.59]	7.25	[6.58;7.92]	*	*	0.55	[0.15;0.94]

* Collected data was erroneous

Table C.6.: DEL and DES for strided access with 32 threads to DRAM and PMem with 95% asymptotic confidence intervals. Mean of 3 reptitions.

	DEL [nJ]			
Threads	**DRAM**		**PMem**	
1	2.82	[1.62;4.02]	31.95	[25.19;38.72]
2	2.87	[1.89;3.85]	51.83	[44.58;59.08]
3	4.55	[2.96;6.14]	43.78	[39.46;48.09]
4	3.86	[1.82;5.89]	80.74	[65.78;95.69]
5	9.78	[6.80;12.76]	102.91	[74.55;131.26]
6	7.7	[7.26;8.14]	189.47	[158.06;220.88]
7	5.81	[5.32;6.29]	88.28	[64.45;112.10]
8	5.31	[3.47;7.15]	73.36	[38.47;108.25]
9	7.03	[4.31;9.75]	117.11	[49.60;184.62]
10	5.7	[3.33;8.07]	114.47	[112.46;116.48]
11	7.26	[4.17;10.36]	138.55	[88.12;188.99]
12	10.38	[8.72;12.04]	161.49	[32.50;290.47]
13	12.16	[5.23;19.08]	293.5	[189.49;397.51]
14	10.4	[4.59;16.22]	151.82	[100.56;203.08]
15	13.98	[3.43;24.54]	193.76	[131.78;255.74]
16	16.35	[11.89;20.81]	133.67	[82.65;184.70]
17	13.85	[1.92;25.77]	206.71	[32.61;380.80]
18	11.77	[9.55;14.00]	269.88	[205.43;334.32]
19	16.3	[2.16;30.43]	264.92	[197.40;332.43]
20	16.05	[10.50;21.60]	228.43	[81.90;374.95]
21	26.47	[11.26;41.69]	153.15	[96.33;209.97]
22	18.67	[3.66;33.67]	270.67	[23.35;517.99]
23	33.8	[21.57;46.04]	266.98	[162.94;371.02]
24	14.97	[7.22;22.72]	242.69	[185.64;299.75]
25	26.09	[6.17;46.02]	207.0	[89.67;324.34]
26	19.24	[5.34;33.15]	148.22	[97.08;199.36]
27	16.66	[-0.18;33.49]	226.07	[204.37;247.76]
28	17.16	[5.91;28.41]	344.82	[106.34;583.31]
29	25.25	[18.86;31.63]	319.11	[103.49;534.73]
30	24.82	[18.80;30.83]	205.05	[170.60;239.49]
31	28.83	[5.04;52.63]	374.41	[78.35;670.47]
32	13.57	[8.55;18.60]	269.25	[115.15;423.36]

Table C.7.: DEL for random access to DRAM and PMem with 95% asymptotic confidence intervals. Mean of 3 reptitions.



dram

Count	Pattern	Threads	Stride	DEL/DES	Dynamic Energy
2387484848x	Sequential Load	4		0.83	19.8161 J
31435566x	Strided Write	8	8	18.98	5.9665 J
433444x	Random Load	32		13.57	0.0588 J
55562646x	Random Load	16		16.35	9.0845 J
2474916504x	Total				**34.9259 J**

optane

Count	Pattern	Threads	Stride	DEL/DES	Dynamic Energy
2387484848x	Sequential Load	4		9.03	215.5899 J
31435566x	Strided Write	8	8	700.45	220.1904 J
433444x	Random Load	32		269.25	1.1670 J
55562646x	Random Load	16		133.67	74.2706 J
2474916504x	Total				**511.2179 J**

Figure C.1.: Screenshot of the energy estimation calculator, provided as a Jupyter notebook.

Figure C.1 shows a screenshot of the calculator for estimating the dynamic memory energy. This calculator can be found in the form of a Jupyter notebook inside the `data` folder of the supplementary material of this thesis.

Acronyms

Glossary

Buck Converter Switching power converter that steps down an input voltage to a stable output voltage 19

CSV Row-based file format with each line forming a row and each column being separated using a comma 37

Dual Inline Memory Module DIMM describes the form-factor of memory modules 1, 6, 107

Dynamic Random Access Memory Volatile memory, often used as main memory of computers 1, 4, 107

GPIO Pin that can be configured to either output or read digital signals 34

Hall Effect Current Sensor Sensor that measures the magnetic field generated by the current passing through the circuit and outputting a proportional voltage. This allows non-inversive current measurements 15, 21

I2C A protocol for communication between integrated circuits 20, 34

IMON Current monitoring capability of voltage regulators 20, 51, 53

Intel Optane Persistent Memory Byte-addressible, persistent memory in DIMM form factor 1, 81, 107

MOSFET A commonly used type of field-effect transistors, which allows current to flow between two terminals if a certain voltage is present at a third pin 19, 20

Running Average Power Limit An interface for monitoring and limiting power consumption on Intel systems 1, 3, 12, 58, 107

Serial Voltage Identification Protocol for communicating between voltage regulators and CPUs. Allows setting output voltages and measuring power consumption at the VR 20, 52, 108

Thermal Design Power Maxmium thermal disspiation of a device 7, 108

UART Bidirectional and asynchronous communcation protocol between two devices 34

Voltage Regulator Integrated circuit that outputs a configured, stable output voltage 108

XPLine The 256 Byte XPLine determines the minimal access granularity of PMem. The XPBuffer combines subsequent accesses to the same XPLine 8, 70

References

[1] T. Allen and R. Ge. Characterizing Power and Performance of GPU Memory Access. In *2016 4th International Workshop on Energy Efficient Supercomputing (E2SC)*. 2016 4th International Workshop on Energy Efficient Supercomputing (E2SC), pages 46–53, Salt Lake City, UT, USA. IEEE, Nov. 2016. ISBN: 978-1-5090-3856-5. DOI: 10.1109/E2SC.2016.012.

[2] Y. Arafa, A. ElWazir, A. ElKanishy, Y. Aly, A. Elsayed, A.-H. Badawy, G. Chennupati, S. Eidenbenz, and N. Santhi. Verified instruction-level energy consumption measurement for NVIDIA GPUs. In *Proceedings of the 17th ACM International Conference on Computing Frontiers*, CF '20, pages 60–70, New York, NY, USA. Association for Computing Machinery, May 23, 2020. ISBN: 978-1-4503-7956-4. DOI: 10.1145/3387902.3392613.

[3] E. Arima, T. Hanawa, and M. Schulz. Toward Footprint-Aware Power Shifting for Hybrid Memory Based Systems (Extended Abstract), 2020. URL: https://oaciss.uoregon.edu/icpp18/publications/pos118s2-file1.pdf (Accessed: 10/08/2023).

[4] I. Bhati, M.-T. Chang, Z. Chishti, S.-L. Lu, and B. Jacob. DRAM Refresh Mechanisms, Penalties, and Trade-Offs. *IEEE Transactions on Computers*, 65(1):108–121, Jan. 1, 2016. ISSN: 0018-9340. DOI: 10.1109/TC.2015.2417540.

[5] Components101. Current Sense Resistors - Types, Specifications, Selection, and Application Guide, Components101, July 23, 2020. URL: https://components101.com/articles/what-is-current-sense-resistors-types-specifications-and-selection (Accessed: 10/08/2023).

[6] Crucial. How much power does memory use? Crucial. 2019. URL: https://www.crucial.com/support/articles-faq-memory/how-much-power-does-memory-use (Accessed: 10/11/2023).

[7] Crucial. RAM Generations: DDR2 vs DDR3 vs DDR4 vs DDR5. Crucial. 2022. URL: https://www.crucial.com/articles/about-memory/difference-among-ddr2-ddr3-ddr4-and-ddr5-memory (Accessed: 07/11/2023).

[8] H. David, E. Gorbatov, U. R. Hanebutte, R. Khanna, and C. Le. RAPL: memory power estimation and capping. In *Proceedings of the 16th ACM/IEEE International Symposium on Low Power Electronics and Design*. ISLPED'10: International Symposium on Low Power Electronics and Design, pages 189–194, Austin Texas USA. ACM, Aug. 18, 2010. ISBN: 978-1-4503-0146-6. DOI: 10.1145/1840845.1840883.

[9] J. Demmel and A. Gearhart. Instrumenting Linear Algebra Energy Consumption via On-Chip Energy Counters, EECS Department, University of California, Berkeley, June 2012. URL: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-168.pdf (Accessed: 10/03/2023).

[10] S. Desrochers, C. Paradis, and V. M. Weaver. A Validation of DRAM RAPL Power Measurements. In *Proceedings of the Second International Symposium on Memory Systems*. MEMSYS '16: The Second International Symposium on Memory Systems, pages 455–470, Alexandria VA USA. ACM, Oct. 3, 2016. ISBN: 978-1-4503-4305-3. DOI: 10.1145/2989081.2989088.

[11] B. R. de Supinski, T. R. W. Scogland, A. Duran, M. Klemm, S. M. Bellido, S. L. Olivier, C. Terboven, and T. G. Mattson. The Ongoing Evolution of OpenMP. *Proceedings of the IEEE*, 106(11):2004–2019, Nov. 2018. ISSN: 1558-2256. DOI: 10.1109/JPROC.2018.2853600.

[12] J. Dongarra, H. Ltaief, P. Luszczek, and V. M. Weaver. Energy footprint of advanced dense numerical linear algebra using tile algorithms on multicore architectures. In *2012 Second International Conference on Cloud and Green Computing*, pages 274–281, 2012. DOI: 10.1109/CGC.2012.113.

[13] T. D. Economon. OpenMP implementation that solves the Poisson equation with CG using a CSR matrix representation. June 9, 2014. URL: https://web.stanford.edu/class/cs315b/projects/conjugate_gradient/conjugate_gradient.cpp (Accessed: 10/06/2023).

[14] S. Erlbeck and D. Schürhoff. NUMA, RWTH Aachen University - IT Center, 2018. URL: https://hpc-wiki.info/hpc/NUMA (Accessed: 10/03/2023).

[15] M. Fahad, A. Shahid, R. R. Manumachu, and A. Lastovetsky. A Comparative Study of Methods for Measurement of Energy of Computing. *Energies*, 12(11):2204, 11, Jan. 2019. ISSN: 1996-1073. DOI: 10.3390/en12112204.

[16] S. Ghose, T. Li, N. Hajinazar, D. S. Cali, and O. Mutlu. Understanding the Interactions of Workloads and DRAM Types: A Comprehensive Experimental Study. Oct. 18, 2019. arXiv: 1902.07609 [cs]. URL: http://arxiv.org/abs/1902.07609 (Accessed: 07/09/2023). preprint.

[17] S. Ghose, A. G. Yaglikçi, R. Gupta, D. Lee, K. Kudrolli, W. X. Liu, H. Hassan, K. K. Chang, N. Chatterjee, A. Agrawal, M. O'Connor, and O. Mutlu. What Your DRAM Power Models Are Not Telling You: Lessons from a Detailed Experimental Study. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 2(3):1–41, Dec. 21, 2018. ISSN: 2476-1249. DOI: 10.1145/3224419.

[18] C. Gough, I. Steiner, and W. Saunders. *Energy Efficient Servers: Blueprints for Data Center Optimization*. Apress, Berkeley, CA, 2015. ISBN: 978-1-4302-6638-9. DOI: 10.1007/978-1-4302-6638-9.

[19] E. T. Grochowski and M. Annavaram. Energy per Instruction Trends in Intel ® Microprocessors. In Intel Corporation, 2006. URL: https://www.intel.com/pressroom/kits/core2duo/pdf/epi-trends-final2.pdf (Accessed: 09/21/2023).

[20] D. Hackenberg, R. Schone, T. Ilsche, D. Molka, J. Schuchart, and R. Geyer. An Energy Efficiency Feature Survey of the Intel Haswell Processor. In *2015 IEEE International Parallel and Distributed Processing Symposium Workshop*. 2015 IEEE International Parallel and Distributed Processing Symposium Workshop (IPDPSW), pages 896–904, Hyderabad, India. IEEE, May 2015. ISBN: 978-1-4673-7684-6. DOI: 10.1109/IPDPSW.2015.70.

[21] G. Hager and G. Wellein. *Introduction to High Performance Computing for Scientists and Engineers*. CRC Press, 0th edition, July 2, 2010. ISBN: 978-1-4398-1193-1. DOI: 10.1201/EBK1439811924. (Accessed: 07/09/2023).

[22] M. Hähnel, B. Döbel, M. Völp, and H. Härtig. Measuring energy consumption for short code paths using RAPL. *ACM SIGMETRICS Performance Evaluation Review*, 40(3):13–17, Jan. 4, 2012. ISSN: 0163-5999. DOI: 10.1145/2425248.2425252.

[23] T. Ilsche. *Energy Measurements of High Performance Computing Systems: From Instrumentation to Analysis*, July 31, 2020. URL: https://nbn-resolving.org/urn:nbn:de:bsz:14-qucosa2-716000 (Accessed: 06/12/2023).

[24] Infineon. TDA38640 OptiMOS iPOL 40A Single-voltage Synchronous Buck Regulator with SVID and I2C, Infineon, Nov. 16, 2022. URL: https://www.infineon.com/dgdl/Infineon-TDA38640-0000-DataSheet-v02_04-EN.pdf (Accessed: 08/28/2023).

[25] Infineon. VR13 and VR12.5 Multi-rail / Multiphase Digital Controllers, Infineon, July 20, 2020. URL: https://www.infineon.com/cms/en/product/power/dc-dc-converters/digital-multiphase-controllers/pxe1610cdn-g003/ (Accessed: 10/15/2023).

[26] Intel. 3rd Gen Intel® Xeon® Scalable Processor, Codename Ice Lake-SP Datasheet, Volume One: Electrical, Intel, Aug. 2022. URL: https://www.intel.com/content/www/us/en/content-details/732800/3rd-gen-intel-xeon-scalable-processor-codename-ice-lake-sp-datasheet-volume-one-electrical.html (Accessed: 08/17/2023).

[27] Intel. Intel® 64 and IA-32 Architectures Software Developer's Manual Combined Volumes: 1, 2A, 2B, 2C, 2D, 3A, 3B, 3C, 3D, and 4, Intel, Mar. 2023. URL: https://cdrdv2-public.intel.com/774494/325462-sdm-vol-1-2abcd-3abcd.pdf (Accessed: 09/04/2023).

[28] Intel. Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 4: Model-Specific Registers, Oct. 2019. URL: https://www.intel.com/content/dam/develop/external/us/en/documents/335592-sdm-vol-4.pdf (Accessed: 09/04/2023).

[29] Intel. Intel® Memory Latency Checker v3.9a. Intel. July 20, 2021. URL: https://www.intel.com/content/www/us/en/developer/articles/tool/intelr-memory-latency-checker.html (Accessed: 07/16/2023).

[30] Intel. Intel® Optane™ Persistent Memory 200 Series Brief. Intel. URL: https://www.intel.com/content/www/de/de/products/docs/memory-storage/optane-persistent-memory/optane-persistent-memory-200-series-brief.html (Accessed: 08/23/2023).

[31] Intel. Intel® Server Board M50CYP2SB Technical Product Specification, Intel Corporation, Aug. 2022. URL: https://www.intel.com/content/dam/support/us/en/documents/server-products/single-node-servers/m50cyp2sb-server-board-tps.pdf (Accessed: 10/15/2023).

[32] Intel. Running Average Power Limit Energy Reporting / CVE-2020-8694 , CVE-2020-8695 / INTEL-SA-00389. Intel Corporation. Feb. 8, 2022. URL: https://www.intel.com/content/www/cn/zh/developer/articles/technical/software-security-guidance/advisory-guidance/running-average-power-limit-energy-reporting.html (Accessed: 08/25/2023).

[33] Intel. Upgrade Your Enterprise with the SSD that Redefines Performance - PM9A3 SSD, Samsung Electronics Co., Ltd., Aug. 2021. URL: https://download.semiconductor.samsung.com/resources/white-paper/PM9A3_SSD_Whitepaper_230327_2.pdf (Accessed: 10/10/2023).

[34] Intel. Voltage Regulator Module (VRM) and Enterprise Voltage Regulator-Down (EVRD) 11.1 - Design Guidelines, Intel, Sept. 2009. URL: https://www.intel.it/content/dam/doc/design-guide/voltage-regulator-module-enterprise-voltage-regulator-down-11-1-guidelines.pdf (Accessed: 09/02/2023).

[35] C. Janna and M. Ferronato. Janna/Flan_1565 3D model of a steel flange, hexahedral finite elements. In collaboration with T. Davis. 2011. URL: https://sparse.tamu.edu/Janna/Flan_1565 (Accessed: 10/12/2023).

[36] JEDEC. JEDEC Standard No. 21C, Release 29 - DDR4 SDRAM Registered DIMM Design Specification, version 1.50, May 2019. URL: https://www.jedec.org/standards-documents/docs/module4_20_28 (Accessed: 09/06/2023).

[37] JEDEC. JEDEC Standard No. 305 DDR5 Load Reduced (LRDIMM) and Registered Dual Inline Memory Module (RDIMM) Common Specification, Jan. 2022. URL: https://www.jedec.org/standards-documents/docs/jesd305 (Accessed: 10/18/2023).

[38] JEDEC. JEDEC Standard No. 308 DDR5 Unbuffered Dual Inline Memory Module (UDIMM) Common Standard, June 2022. URL: https://www.jedec.org/standards-documents/docs/jesd308 (Accessed: 10/18/2023).

[39] S. Kargar and F. Nawab. Hamming Tree: The Case for Energy-Aware Indexing for NVMs. *Proceedings of the ACM on Management of Data*, 1(2):182:1–182:27, June 20, 2023. DOI: 10.1145/3589327.

[40] M. Katsaragakis, C. Baloukas, L. Papadopoulos, V. Kantere, F. Catthoor, and D. Soudris. Energy Consumption Evaluation of Optane DC Persistent Memory for Indexing Data Structures. Apr. 3, 2023. arXiv: 2304.00953 [cs]. URL: http://arxiv.org/abs/2304.00953 (Accessed: 08/05/2023). preprint.

[41] K. N. Khan, M. Hirki, T. Niemi, J. K. Nurminen, and Z. Ou. RAPL in Action: Experiences in Using RAPL for Power Measurements. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, 3(2):1–26, June 30, 2018. ISSN: 2376-3639, 2376-3647. DOI: 10.1145/3177754.

[42] K. N. Khan, Z. Ou, M. Hirki, J. K. Nurminen, and T. Niemi. How much power does your server consume? Estimating wall socket power using RAPL measurements. *Computer Science - Research and Development*, 31(4):207–214, Nov. 1, 2016. ISSN: 1865-2042. DOI: 10.1007/s00450-016-0325-4.

[43] N. K. Khan. *Energy Measurement and Modeling in High Performance Computing with Intel's RAPL*. Doctoral thesis, Aalto University / School of Science, 2018. ISBN: 978-952-60-7892-2. URL: http://urn.fi/URN:ISBN:978-952-60-7892-2 (Accessed: 10/15/2023).

[44] S. Köhler, B. Herzog, T. Hönig, L. Wenzel, M. Plauth, J. Nolte, A. Polze, and W. Schröder-Preikschat. Pinpoint the joules: Unifying runtime-support for energy measurements on heterogeneous systems. In *2020 IEEE/ACM International Workshop on Runtime and Operating Systems for Supercomputers (ROSS)*, pages 31–40, 2020. DOI: 10.1109/ROSS51935.2020.00009.

[45] A. Kozhokanova, B. Wang, C. Terboven, and M. Mueller. Power-aware Computing with Optane Persistent Memory Modules. In *2023 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. 2023 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), pages 26–31, May 2023. DOI: 10.1109/IPDPSW59300.2023.00017.

[46] V. Kulkarni. Instruction Level Power Consumption Estimation – Issues and Review. In 2017. URL: https://www.semanticscholar.org/paper/Instruction-Level-Power-Consumption-Estimation-%E2%80%93-Kulkarni/17103a7401ed8b5d4fd99bd464f647a4e29e03ec (Accessed: 09/21/2023).

[47] J. H. Laros Iii, K. Pedretti, S. M. Kelly, W. Shu, K. Ferreira, J. Vandyke, and C. Vaughan. *Energy Delay Product*. In J. H. Laros Iii, K. Pedretti, S. M. Kelly, W. Shu, K. Ferreira, J. Van Dyke, and C. Vaughan. *Energy-Efficient High Performance Computing*. Springer London, London, 2013, pages 51–55. ISBN: 978-1-4471-4492-2. DOI: 10.1007/978-1-4471-4492-2_8.

[48] D. Li, J. Wan, J. Wang, J. Zhou, K. Lu, P. Xu, F. Wu, and C. Xie. Disperse Access Considered Energy Inefficiency in Intel Optane DC Persistent Memory Servers. In *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*. 2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS), pages 921–931, Nov. 2020. DOI: 10.1109/ICDCS47774.2020.00107.

[49] Linux. Jemalloc. URL: https://jemalloc.net/ (Accessed: 07/09/2023).

[50] Linux. Numa(3) - Linux manual page. URL: https://man7.org/linux/man-pages/man3/numa.3.html (Accessed: 07/09/2023).

[51] Linux. Numactl(8) - Linux manual page. URL: https://man7.org/linux/man-pages/man8/numactl.8.html (Accessed: 07/09/2023).

[52] Linux. Perf-stat(1) - Linux manual page. URL: https://man7.org/linux/man-pages/man1/perf-stat.1.html (Accessed: 07/31/2023).

[53] Linux. Ubuntu Manpage: ipmctl-show-device - Shows information about one or more DCPMMs. URL: https://manpages.ubuntu.com/manpages/focal/man1/ipmctl-show-device.1.html (Accessed: 08/23/2023).

[54] M. Lipp, A. Kogler, D. Oswald, M. Schwarz, C. Easdon, C. Canella, and D. Gruss. PLATYPUS: Software-based Power Side-Channel Attacks on x86. In *2021 IEEE Symposium on Security and Privacy (SP)*. 2021 IEEE Symposium on Security and Privacy (SP), pages 355–371, San Francisco, CA, USA. IEEE, May 2021. ISBN: 978-1-72818-934-5. DOI: 10.1109/SP40001.2021.00063.

[55] L. Looi, J. J. Xu, A. Altman, M. Arafa, K. Balasubramanian, K. Cheng, P. Damle, S. Datta, K. Gibson, B. Graniello, J. Grooms, N. Gurumoorthy, I. C. Escareno, T. Kasanicky, K. Khochare, Z. Li, S. Mandava, R. Mangold, S. Muralidhara, S. Najnin, B. Nale, J. Pickett, S. Qawami, T. Quach, B. Querbach, C. Raad, A. Rudoff, R. Saffores, I. Steiner, S. Thakkar, V. Viswanathan, D. Wu, and C. Xu. Intel® Optane™ Data Center Persistent Memory. Aug. 19, 2019. URL: https://old.hotchips.org/hc31/HC31_1.6_OptaneDCPMM_2019_FinalSlides_v4b.pdf (Accessed: 08/23/2023).

[56] D. M. Mathew, É. F. Zulian, S. Kannoth, M. Jung, C. Weis, and N. Wehn. A Bank-Wise DRAM Power Model for System Simulations. In *Proceedings of the 9th Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools*, RAPIDO '17, pages 1–7, New York, NY, USA. Association for Computing Machinery, Jan. 23, 2017. ISBN: 978-1-4503-4840-9. DOI: 10.1145/3023973.3023978.

[57] Maxim Integrated. MAX16602 VR13.HC and AI Cores Dual-Output Voltage Regulator Chipset, 2020. URL: https://www.analog.com/media/en/technical-documentation/data-sheets/MAX16602.pdf (Accessed: 09/28/2023).

[58] A. Mazouz, B. Pradelle, and W. Jalby. Statistical validation methodology of CPU power probes. In *Revised Selected Papers, Part I, of the Euro-Par 2014 International Workshops on Parallel Processing - Volume 8805*, pages 487–498, Berlin, Heidelberg. Springer-Verlag, 2014. ISBN: 978-3-319-14324-8. DOI: 10.1007/978-3-319-14325-5_42.

[59] H. McCraw, J. Ralph, A. Danalis, and J. Dongarra. Power monitoring with PAPI for extreme scale architectures and dataflow-based programming models. In *2014 IEEE International Conference on Cluster Computing (CLUSTER)*. 2014 IEEE International Conference On Cluster Computing (CLUSTER), pages 385–391, Madrid, Spain. IEEE, Sept. 2014. ISBN: 978-1-4799-5548-0. DOI: 10.1109/CLUSTER.2014.6968672.

[60] Measurement Computing Corporation. MCC128 - 16-bit Voltage Measurement DAQ HAT for Raspberry Pi, Measurement Computing, Dec. 2022. URL: https://www.mccdaq.com/PDFs/specs/DS-MCC-128.pdf (Accessed: 06/15/2023).

[61] Microchip Technology, Inc. MCP2221 USB 2.0 to I2C/UART Protocol Converter with GPIO, Microchip Technology, Inc., 2017. URL: http://ww1.microchip.com/downloads/en/devicedoc/20005292c.pdf (Accessed: 10/15/2023).

[62] Micron Technology. Calculating memory power for DDR4 SDRAM, Micron Technology, Inc., 2017. URL: https://www.micron.com/-/media/client/global/documents/products/technical-note/dram/tn4007_ddr4_power_calculation.pdf (Accessed: 10/15/2023).

[63] D. Molka, D. Hackenberg, R. Schone, and M. S. Muller. Characterizing the energy consumption of data transfers and arithmetic operations on x86-64 processors. In *International Conference on Green Computing*. 2010 International Conference on Green Computing (Green Comp), pages 123–133, Chicago, IL, USA. IEEE, Aug. 2010. ISBN: 978-1-4244-7612-1. DOI: 10.1109/GREENCOMP.2010.5598316.

[64] Monolythic Power Systems. MP2935 4-Phase PWM Controller for VR12.5 Applications, Oct. 4, 2019. URL: https://www.monolithicpower.com/en/documentview/productdocument/index/version/2/document_type/Datasheet/lang/en/sku/MP2935/ (Accessed: 09/28/2023).

[65] Monolythic Power Systems. MP2965 Dual-Loop, Digital, Multi-Phase Controller with PMBus Interface for VR13.HC/AVSBUS, Oct. 4, 2019. URL: https://www.monolithicpower.com/en/documentview/productdocument/index/version/2/document_type/Datasheet/lang/en/sku/MP2965/document_id/5134/ (Accessed: 09/28/2023).

[66] National Instruments. Grounding Considerations - Intermediate Analog Concepts. Oct. 14, 2022. URL: https://www.ni.com/en/support/documentation/supplemental/06/grounding-considerations---intermediate-analog-concepts.html (Accessed: 08/16/2023).

[67] C. Ntogkas. Design and implementation of a tool for memory access pattern visualization, 2017. URL: https://core.ac.uk/download/pdf/157700634.pdf (Accessed: 10/14/2023).

[68] J. M. Paniego, S. Gallo, M. Pi Puig, F. Chichizola, L. De Giusti, and J. Balladini. Analysis of RAPL Energy Prediction Accuracy in a Matrix Multiplication Application on Shared Memory. In A. E. De Giusti, editor, *Computer Science – CACIC 2017*, Communications in Computer and Information Science, pages 37–46, Cham. Springer International Publishing, 2018. ISBN: 978-3-319-75214-3. DOI: 10.1007/978-3-319-75214-3_4.

[69] I. B. Peng, M. B. Gokhale, and E. W. Green. System evaluation of the Intel optane byte-addressable NVM. In *Proceedings of the International Symposium on Memory Systems*. MEMSYS '19: The International Symposium on Memory Systems, pages 304–315, Washington District of Columbia USA. ACM, Sept. 30, 2019. ISBN: 978-1-4503-7206-0. DOI: 10.1145/3357526.3357568.

[70] M. Pitz, N. Eiling, F. Wege, I. Köster, S. Lankes, and A. Monti. Accuracy Analysis of Power Sensors in Compute Servers Using a High Precision Power Analyzer. *ISC High Performance 2023*, ISC 2023, May 21, 2023. DOI: 10.18154/RWTH-2023-05754.

[71] PMem.io. MEMKIND. URL: https://pmem.io/memkind/manpages/memkind.3/ (Accessed: 07/09/2023).

[72] T. Rauber, G. Rünger, M. Schwind, H. Xu, and S. Melzner. Energy measurement, modeling, and prediction for processors with frequency scaling. *The Journal of Supercomputing*, 70(3):1451–1476, Dec. 2014. ISSN: 0920-8542, 1573-0484. DOI: 10.1007/s11227-014-1236-4.

[73] Renesas. ISL69133 Digital, Dual Output, 4-Phase Configurable, VR13/IMVP8 PWM Controller, Renesas, Feb. 2018. URL: https://www.renesas.com/us/en/document/sds/isl69133-data-short (Accessed: 10/15/2023).

[74] Richtek. RT8120 - Single-Phase Synchronous Buck PWM Controller, 2021. URL: https://www.richtek.com/assets/product_file/RT8120/DS8120-09.pdf (Accessed: 09/26/2023).

[75] Richtek. RT8167A Dual Single-Phase PWM Controller for CPU Core/GFX Power Supply, 2014. URL: https://www.richtek.com/assets/product_file/RT8167A/DS8167A-02.pdf (Accessed: 09/28/2023).

[76] S. I. Roberts, S. A. Wright, S. A. Fahmy, and S. A. Jarvis. Metrics for Energy-Aware Software Optimisation. In J. M. Kunkel, R. Yokota, P. Balaji, and D. Keyes, editors, *High Performance Computing*. Volume 10266, pages 413–430. Springer International Publishing, Cham, 2017. ISBN: 978-3-319-58667-0. DOI: 10.1007/978-3-319-58667-0_22.

[77] C. R. Robertson. *Fundamental Electrical and Electronic Principles*. Newnes, Amsterdam, 3rd ed edition, 2008. 291 pages. ISBN: 978-0-7506-8737-9.

[78] E. Rotem, A. Naveh, A. Ananthakrishnan, E. Weissmann, and D. Rajwan. Power-Management Architecture of the Intel Microarchitecture Code-Named Sandy Bridge. *IEEE Micro*, 32(2):20–27, Mar. 2012. ISSN: 0272-1732. DOI: 10.1109/MM.2012.12.

[79] S. Scargall. *Programming Persistent Memory: A Comprehensive Guide for Developers*. Springer Nature, 2020. ISBN: 978-1-4842-4932-1. DOI: 10.1007/978-1-4842-4932-1.

[80] S. Schlachter and B. Drake. Introducing Micron DDR5 SDRAM: More Than a Generational Update, 2019. URL: https://www.micron.com/-/media/client/global/documents/products/white-paper/ddr5_more_than_a_generational_update_wp.pdf (Accessed: 09/26/2023).

[81] R. Schone, T. Ilsche, M. Bielert, M. Velten, M. Schmidl, and D. Hackenberg. Energy Efficiency Aspects of the AMD Zen 2 Architecture. In *2021 IEEE International Conference on Cluster Computing (CLUSTER)*. 2021 IEEE International Conference on Cluster Computing (CLUSTER), pages 562–571, Portland, OR, USA. IEEE, Sept. 2021. ISBN: 978-1-72819-666-4. DOI: 10.1109/Cluster48925.2021.00087.

[82] R. Schone, T. Ilsche, M. Bielert, M. Velten, M. Schmidl, and D. Hackenberg. Energy Efficiency Aspects of the AMD Zen 2 Architecture measurement and validation code - RAPL resolution - source code. URL: https://github.com/tud-zih-energy/2021-rome-ee/blob/main/RAPL_resoultion/test.c (Accessed: 10/17/2023).

[83] R. Schone, M. Schmidl, M. Bielert, and D. Hackenberg. FIRESTARTER 2: Dynamic Code Generation for Processor Stress Tests. In *2021 IEEE International Conference on Cluster Computing (CLUSTER)*. 2021 IEEE International Conference on Cluster Computing (CLUSTER), pages 582–590, Portland, OR, USA. IEEE, Sept. 2021. ISBN: 978-1-72819-666-4. DOI: 10.1109/Cluster48925.2021.00084.

[84] R. Schöne, T. Ilsche, M. Bielert, A. Gocht, and D. Hackenberg. Energy efficiency features of the intel skylake-SP processor and their impact on performance. In *2019 International Conference on High Performance Computing & Simulation (HPCS)*, pages 399–406, 2019. DOI: 10.1109/HPCS48598.2019.9188239.

[85] M. Shahid. Sparse-Matrix-Vector-Mul, May 17, 2022. URL: https://github.com/muneeb706/Sparse-Matrix-Vector-Mul (Accessed: 09/30/2023).

[86] D. D. Sharma. Compute Express Link®: An open industry-standard interconnect enabling heterogeneous data-centric computing. In *2022 IEEE Symposium on High-Performance Interconnects (HOTI)*. 2022 IEEE Symposium on High-Performance Interconnects (HOTI), pages 5–12, Aug. 2022. DOI: 10.1109/HOTI55740.2022.00017.

[87] SK hynix. DDR4 SDRAM Registered DIMM Based on 4Gb A-die, SK hynix, June 2016. URL: https://gzhls.at/blob/ldb/4/3/7/f/60f0bf5e41d678988473cd82e746d9a27454.pdf (Accessed: 09/06/2023).

[88] D. Terrazas. Difference Between an Instrumentation Amplifier and a Current Sense Amplifier, Texas Instruments, Apr. 2020. URL: https://www.ti.com/lit/an/sboa374/sboa374.pd (Accessed: 09/26/2023).

[89] Texas Instruments. CS-AMPLIFIER-ERROR-TOOL Calculation tool | TI.com. May 9, 2023. URL: https://www.ti.com/tool/CS-AMPLIFIER-ERROR-TOOL (Accessed: 08/16/2023).

[90] Texas Instruments. INA122 - Single Supply, MicroPower INSTRUMENTATION AMPLIFIER, Texas Instruments, Oct. 1997. URL: https://www.ti.com/lit/ds/symlink/ina122.pdf (Accessed: 10/03/2023).

[91] Texas Instruments. Ina128.INA12x Precision, Low-Power Instrumentation Amplifiers, May 2022. URL: https://www.ti.com/lit/ds/symlink/ina128.pdf (Accessed: 10/03/2023).

[92] Texas Instruments. INAx180 Low- and High-Side Voltage Output, Current-Sense Amplifiers, Texas Instruments, June 2022. URL: https://www.ti.com/lit/ds/symlink/ina2180.pdf (Accessed: 10/03/2023).

[93] Texas Instruments. Simplifying Current Sensing - How to design with current sense amplifiers. Oct. 28, 2020. URL: https://www.ti.com/lit/slyy154 (Accessed: 09/04/2023).

[94] Texas Instruments. TPS544C26 4-V to 16-V Input, 35-A Synchronous Buck Converter With SVID And I2C Interfaces, Texas Instruments, Sept. 2022. URL: https://www.ti.com/lit/ds/symlink/tps544c26.pdf (Accessed: 09/03/2023).

[95] Top500. GREEN500 June 2023. URL: https://www.top500.org/lists/green500/2023/06/ (Accessed: 07/18/2023).

[96] Top500. List Statistics | TOP500. URL: https://www.top500.org/statistics/list/ (Accessed: 07/18/2023).

[97] J. Treibig, G. Hager, and G. Wellein. LIKWID: Lightweight Performance Tools. In C. Bischof, H.-G. Hegering, W. E. Nagel, and G. Wittum, editors, *Competence in High Performance Computing 2010*, pages 165–175. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN: 978-3-642-24025-6. DOI: 10.1007/978-3-642-24025-6_14.

[98] Z. Wang, X. Liu, J. Yang, T. Michailidis, S. Swanson, and J. Zhao. Characterizing and Modeling Non-Volatile Memory Systems. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), pages 496–508, Athens, Greece. IEEE, Oct. 2020. ISBN: 978-1-72817-383-2. DOI: 10.1109/MICRO50266.2020.00049.

[99] V. M. Weaver. RAPL Userspace Access without perf - Source Code. RAPL Userspace Access without perf - Source Code. URL: https://web.eece.maine.edu/~vweaver/projects/rapl/rapl-read.c (Accessed: 09/03/2023).

[100] M. Weiland, H. Brunst, T. Quintino, N. Johnson, O. Iffrig, S. Smart, C. Herold, A. Bonanni, A. Jackson, and M. Parsons. An early evaluation of Intel's optane DC persistent memory module and its impact on high-performance scientific applications. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. SC '19: The International Conference for High Performance Computing, Networking, Storage, and Analysis, pages 1–19, Denver Colorado. ACM, Nov. 17, 2019. ISBN: 978-1-4503-6229-0. DOI: 10.1145/3295500.3356159.

[101] M. Willeit. Solutions for Powering Intel and AMD SoCs. Apr. 22, 2020. URL: https://media.monolithicpower.com/mps_cms_document/m/p/mps_solutions_for_powering_intel_and_amd_socs_22.04.2020.pdf (Accessed: 10/15/2023).

[102] F. Winkler. Redesigning PAPI's High-Level API, 2020. URL: https://icl.utk.edu/files/publications/2020/icl-utk-1322-2020.pdf (Accessed: 09/27/2023).

[103] J. Yang, J. Kim, M. Hoseinzadeh, J. Izraelevitz, and S. Swanson. An empirical guide to the behavior and use of scalable persistent memory. In *18th USENIX Conference on File and Storage Technologies (FAST 20)*, pages 169–182, Santa Clara, CA. USENIX Association, Feb. 2020. ISBN: 978-1-939133-12-0. DOI: 10.5555/3386691.3386708.

[104] K. Zhang, D. Ou, C. Jiang, Y. Qiu, and L. Yan. Power and Performance Evaluation of Memory-Intensive Applications. *Energies*, 14(14):4089, July 6, 2021. ISSN: 1996-1073. DOI: 10.3390/en14144089.

[105] Y. Zhang, Y. Dong, J. Chen, Z. Ou, and Y. Yuan. PMC-Based Dynamic Adaptive CPU and DRAM Power Modeling. In M. Qiu, editor, *Algorithms and Architectures for Parallel Processing*, Lecture Notes in Computer Science,

pages 92–111, Cham. Springer International Publishing, 2020. ISBN: 978-3-030-60245-1. DOI: 10.1007/978-3-030-60245-1_7.