AALBORG UNIVERSITY

STUDENT REPORT

Department of Computer Science
Selma Lagerløfs Vej 300
DK-9220 Aalborg Ø

Title:
Integrating RAPL With BenchmarkDotNet and Benchmarking EFCore

Theme:
Energy Efficient Programming

Project Period:
February 2023 - June 2023

Project Group:
cs-23-pt-10-01

Participant(s):
Alfred Ilberg Ulvmose
Virmantas Cekys

Supervisor(s):
Bent Thomsen

Copies: 1

Number of Pages: 75

Date of Completion:
June 9, 2023

Abstract:

There are two goals in this project: developing a RAPL integration for BenchmarkDotNet, and using it to do a case study of different query processing settings of EFCore. An integration for BenchmarkDotNet is created that takes and processes RAPL measurements simultaneously on a client and server machine. The integration works for the case study, and can be further developed for submitting to BenchmarkDotNet. The case study reveals that the 'NoTracking' setting of EFCore almost always costs more energy on the client device.

Summary

This project has two directions in terms of contributing to the general field of knowledge about the energy efficiency of software. Firstly, we investigate how energy efficiency measurements can be integrated into one of the most popular benchmarking frameworks for C#: BenchmarkDotNet (BDN)[1]. Secondly, we attempt to do some energy efficiency benchmarks of Entity Framework Core (EFCore)[2] - both on a client and server device.

In order to create an energy efficiency integration for BenchmarkDotNet (BDN), we use RAPL as a software-based energy meter. To integrate RAPL with BDN we create a number of tools and interfaces such as the RAPL Diagnoser. The integration is designed in such way that would allow us to perform energy efficiency measurements both on a local and remote device.

This integration is then used to perform benchmarking of some query processing settings of EFCore: *Split* query and *Single* query, *NoTracking*, *ToList* and *Load*. To test these settings, a pair of select queries are tested - one that queries a single table, and one that has join operations. These queries are tested with the settings turned on and off, and in different data sizes: fetching from 10 to 100000 rows of data. The tests are performed on a lightweight sample PostgreSQL database called Chinook[3], running on the server device, that we seeded ourselves with some random data.

We end up creating an energy consumption integration for BDN that works for our case study, but the integration is not of sufficient quality to upstream into BDN. This is due to the fact that our integration requires some 'hacks' to work both with a client and a server device. On the other hand, we have outlined ways in which a clean RAPL integration could be built for BDN.

The results of the EFCore benchmarking case study show that *NoTracking* queries always consume more energy on the client device, even when, though not as much, there are no join operations in the query. The difference between *ToList* and *Load* settings fall within the margin of error, so no difference in energy consumption is observed. *Split* queries appear to consume more energy on the local device than *Single* queries.

Contents

Glossary

BDN Engine	BenchmarkDotNet Benchmark Engine - a child process of BDN Runner used for executing benchmarks.
BDN Runner	BenchmarkDotNet Benchmark Runner - the main process of BDN, used for benchmark setup, spawning child processes of BDN Engine, and analysing benchmark results.
PowerCap	A Linux Power Capping framework that is capable of reading RAPL power measurements.
RAPL	Running Average Power Limit - Software-based power consumption measurement tool on newer Intel CPUs.
RAPL Diagnoser	A custom implementation of the BDN *IDiagnoser* interface that is used to integrate RAPL with BDN.
RAPL Monitor	A custom C# interface for taking RAPL Measurements.
RAPL Reader	A custom C++ application made for taking and outputting PowerCap RAPL measurements.
software entity	A specific entity of software to be tested.
software entity class	A class of software entities considered for testing.

Acronyms

BDN	BenchmarkDotNet.
DUT	Device Under Test.
EFCore	Entity Framework Core.
ICT	Information and Communication Technology.
LP	Laboratory Package.
MAD	Median Absolute Deviation.
ORM	Object-Relational Mapper.
POSIX	Portable Operating System Interface.
RDBMS	Relational Database Management System.
SQL	Structured Query Language.
SSH	Secure Shell.
UUID	Universally Unique ID.

Chapter 1

Introduction

While we rely more and more on software in our day to day lives, the energy consumption of our Information and Communication Technology (ICT) devices is also increasing. One study even suggests that in their worst-case analysis, ICT would consume up to 51% of global electricity and contribute up to 23% of global greenhouse emissions by 2030[4]. Not only is this bad from a climate change perspective, but as the 2022 energy crisis illustrated, energy may be in very high demand at times, and we as software developers should do our very best not to waste it.

Awareness has been steadily increasing in both the industry and especially in academia around energy efficient programming, such as Pereira, Couto, et al. [5] comparing energy efficiency of different programming languages, or Lima, Soares-Neto, et al. [6] comparing data structures in Haskell.

A very common piece of software is databases, which are used everywhere from small databases embedded in locally running software, to big database systems running in data centers. Most common database systems are relational[7], but non-relational databases have also become increasingly popular.

To interface with databases, many developers choose to use tools such as an Object-Relational Mapper (ORM) to provide an abstraction over the database's layout and technicalities. These tools greatly simplify interfacing with the database system, and can in some cases even facilitate seamless swapping between different database systems. The use of ORMs do come with trade-offs in the form of hits to both time and energy efficiency, which has been shown in several studies[8, 9].

1.1 Problem Statement

Energy efficiency is a largely unexplored field in the software industry. There is a lack of accessibility to good tools related to this field, and lack of knowledge in the area in general. As such our aim is to do both: push the development of tools in regards to energy efficiency, and bring more knowledge about the energy efficiency of some commonly used software.

1.2 Research Questions

In regards to the problem statement, the research questions are the following:

1. Is it possible to create a RAPL integration for BDN?
2. How do different data processing settings of EFCore affect the energy consumption on the client and server device?

1.3 Expected Contributions

We expect to provide insight into how different data processing settings of EFCore affects energy efficiency, and we expect to make some progress towards getting RAPL integration into BDN.

Chapter 2

Related Works

In this section, we will discuss works that are related to this project. Firstly we will discuss energy efficiency measuring tools and techniques in general, then energy efficiency of databases and ORMs.

2.1 Energy Efficiency Measurement

While there are many excellent hardware-based energy measurement devices out there, we will focus on software-based approaches. In particular RAPL, which has been evaluated by several studies [10, 11, 12] to be at least closely matches the true power consumption of a system.

For energy measurement studies, there seems to be little consistency in the way of methodology and presentation of results. Mancebo, García, et al. [13] seeks to remedy this by proposing a process on how to plan, execute, and present energy efficiency studies, and is extended upon in [14]. We also used, and evaluated, the process in our previous report [15], and found it to be helpful compared to using an ad-hoc methodology.

2.1.1 Benchmarking

We are only aware of two RAPL-based benchmarking frameworks, namely jRAPL[16], a currently maintained Java framework, and CSharpRapl[17], a C# framework made by previous students at Aalborg University, which is no longer actively maintained.

BDN is a popular framework for benchmarking .NET applications, with it being able to automatically determine things such as iteration counts, and it can do statistics and graphing. Unfortunately there is currently no RAPL support for BDN, and is not suitable for energy benchmarking without modification.

2.2 Energy Efficiency of ORMs

While there exists several studies about the performance implications of using ORMs, such as Procaccianti, Lago, et al. [9] which suggests that not using an ORM at all is the most energy and time efficient, but that using a lightweight ORM, such as the TinyQueries library they use in the paper, might be worth it for the trade-off between efficiency and maintainability.

Calero, Polo, et al. [18] created two versions of an application, one using the Spring framework, one using raw Structured Query Language (SQL) and compared the two in terms of energy and time efficiency, and also in terms of problems found using static analysis. They found the Spring-version to have far fewer bugs and vulnerabilities before preventative maintenance, but do note that it comes with a hit in terms of energy and time efficiency. They conclude that software developers should consider whether using tools such as Spring is worth it for their application considering the added environmental impact.

While not about energy efficiency, Zmaranda, Pop-Fele, et al. [19] compares the execution times and memory usages of the .NET ORMs EFCore, Dapper, and nHibernate. They run different types of queries using each ORM, and conclude that depending on which type of queries your application executes the most, either ORM may be more efficient.

Chapter 3

Experiment Design

In order to streamline the experiment design, a seven-phase iterative process defined by Mancebo, García, et al. [13] is used. We are using this process because we had a positive experience using it in the past [15] - it is helpful in the definition of energy efficiency experiments. However, we are not following the process exhaustively, as not all parts are relevant to the project.

3.1 Phase I: Scope Definition

The scope definition consists of the experiment requirements, goal, the software entity to be tested, and all experiment variations and their explanation.

Requirements

The requirements of this experiment are:

- An ORM that has different settings for generating queries in regards to memory or database processing.
- The ORM should be implemented in C#, so that it can be benchmarked using BDN.
- It should be possible to compare the energy usage of the client and server machine.

Goal

The goals of this experiment are:

- Create an integration for BDN that is capable of taking and processing RAPL energy measurements for a given benchmark.
- Compare the energy efficiency of in-memory versus database processing of a popular ORM, both on a client and server machine.
- Find the margins where some ORM settings are more efficient than others when processing data.

Software Entity

The software entity class that fulfils our goals and requirements is a **C# ORM**, in combination with a supported **Relational Database Management System (RDBMS)**. The ORM chosen as our software entity is **EFCore**[2], in combination with **PostgreSQL**[20] as the RDBMS. The reasons for choosing these software entities is their popularity, support, and the fact that they are open source. EFCore is the most popular C# ORM, with more than 700 million downloads as a NuGet package[21]. It also has a number of relevant settings for data processing, such as *Single* and *Split* querying, *Tracking* and *Non-Tracking* etc. that are relevant to our goals and requirements.

Database

The database used for this experiment is a sample PostgreSQL database called Chinook[3]. We are using this database because it has an easy to use SQL script for setup, and also already includes a data model for EFCore. The data model had some minor issues that we had to fix, but other than that it worked well. To populate the database with some more data, we performed some random data seeding, section 4.1. The size of the database is shown in Table 3.1

Table	**Rows**
Artist	2775
Album	12903
Genre	25
MediaType	5
Track	122735
Employee	567
Customer	17021
Invoice	16412
InvoiceLine	210391
Playlist	2518
PlaylistTrack	634050

Table 3.1: Table showing all the tables in the database and the amount of rows in each. This is after seeding.

Test Cases

There are multiple settings that can be turned on or off in EFCore that we expect will affect both the execution time and the energy use of queries:

- *Single* and *Split* queries. This setting influences how join operations are handled in a query. With a single query setting, only a single query is sent to the database with all its corresponding join operations. This often results in duplicate data being sent, as all the joined one-to-many and many-to-many relations are exploded into their own rows - then parsed into corresponding objects in the ORM. Split query setting aims to solve this problem by creating a separate query for each join operation in order to avoid duplicate data being sent.
- *NoTracking* queries. By default, EFCore tracks database entities by mapping them to its database context. This is done for handling references and tracking changes. With the *No-Tracking* setting this behaviour can be disabled, then entities are created ”loose” - just as a standard object, with no tracking.

- *Load* and *ToList* queries. In the most common use case for fetching data from EFCore, the *ToList* method is used which queries the database and returns the result as a list. *Load* is effectively the opposite of using the *NoTracking* setting with *ToList* - it maps the entities to the context, but it does not return them as a list.

Two types of queries are used to test these settings. The intention is to observe how the selected ORM settings affect select queries with and without join operations:

- Tracks: fetching a list of tracks from the database. This does not include the tracks themselves, but only some of their metadata, such as the name and length of the track. Note that Single/Split query setting is inapplicable to queries that fetch data only from a single table - it is always single query.
- Tracks and albums: again fetching a list of tracks, but also joining the album that the track is part of. This results in some data duplication, as the same albums are joined to their corresponding tracks.

In order to observe how the data size affects the results, the same queries are tested with different limits on the amount of rows to fetch: 10, 100, 1000, 10000, and 100000. The final set of test cases is shown in Table 3.2, the query size is omitted for brevity.

Query	**Single/Split Query**	**NoTracking**	**Load/ToList**
Tracks	N/A	NoTracking	ToList
Tracks	N/A	Tracking	ToList
Tracks	N/A	Tracking	Load
Tracks and albums	Single	NoTracking	ToList
Tracks and albums	Single	Tracking	ToList
Tracks and albums	Single	Tracking	Load
Tracks and albums	Split	NoTracking	ToList
Tracks and albums	Split	Tracking	ToList
Tracks and albums	Split	Tracking	Load

Table 3.2: A list of all test cases. Each case is also repeated with a different query size: 10, 100, 1000, 10000, and 100000.

3.2 Phase II: Measurement Environment Setting

3.2.1 A2.1: Selecting a Measuring Instrument

For selecting a measuring instrument for our study, we have elected to use software-based measuring over hardware-based due to ease of use and widespread availability.

We will be using RAPL to measure energy consumption of our Devices Under Test (DUTs) While RAPL was initially an Intel-only feature, AMD does also have its own implementation of RAPL, which does have its limitations compared to its Intel counterpart. For one, AMD has support for fewer domains, only supporting the package and core zones, missing the DRAM, Psys, and uncore zones. The DRAM zone can be especially interesting for studies comparing CPU versus memory-intensive algorithms.

Fortunately, both Intel and AMD RAPL-implementations are supported in the Linux Power Capping Framework, allowing us to read the energy usage by interfacing with that.

3.2.1.1 Power Measuring Architecture

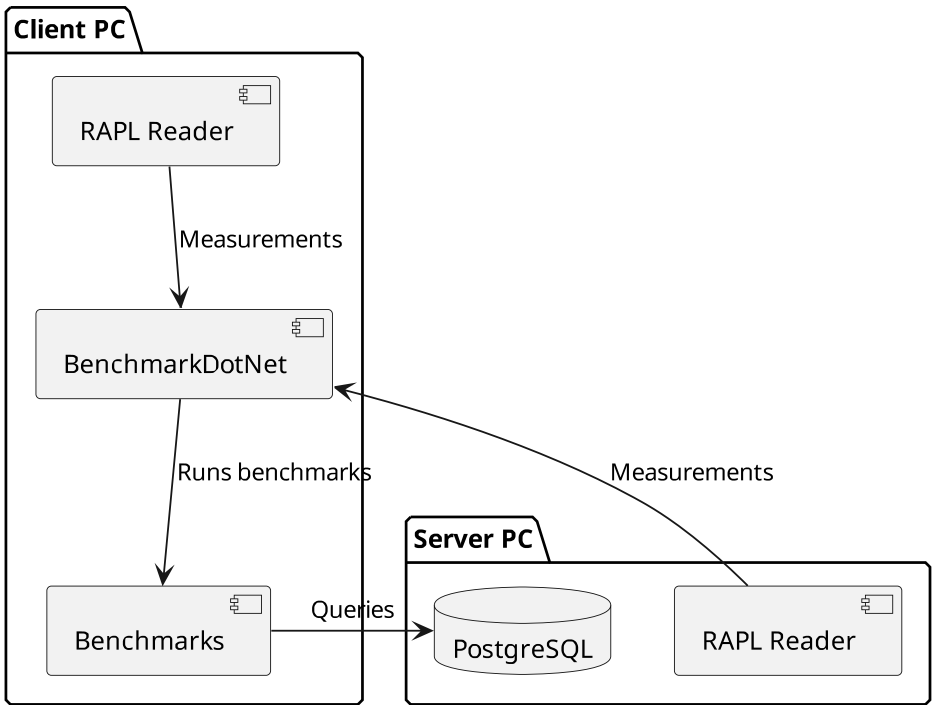

Figure 3.1: Overall architecture of the experiment setup.

Figure 3.1 shows the overall setup of the experiment setup.

The client PC is the computer running the application-side software entity, while the server PC runs the database-side software entity. The actual benchmarks are ran and managed by BDN, which also starts, stops, and gathers energy measurements form the RAPL Reader on both the server PC and locally.

3.2.1.2 RAPL Reader

The RAPL Reader is a program we wrote to read all the energy counters available on the system continuously while each benchmark runs and send them to BDN afterwards for processing.

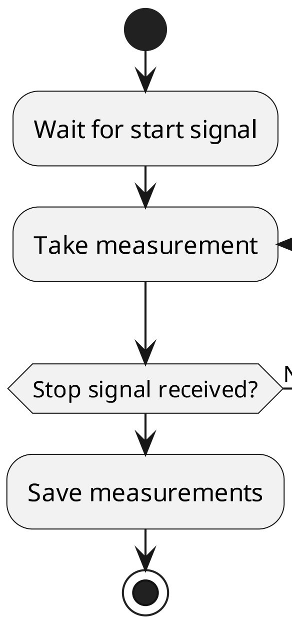

Figure 3.2: Simplified control flow of the RAPL Reader program.

The program itself has a rather simple control flow shown in Figure 3.2, where the program waits for a signal after initializing itself. Upon receiving the signal, it begins taking measurements every 10ms, and then when it receives another signal telling it to stop, it saves the measurements and exits.

3.2.2 A2.2 Define Specifications of the DUTs

The DUTs are a pair of Lenovo ThinkPad T470s W10DG laptops - the only difference between them is the RAM capacity. Both devices are running a clean installation of Arch Linux, with no programs installed or processes running that are not required for the experiment. The following is a list of the DUT hardware specs and software versions. A detailed report of all the hardware and software is in Appendix A.

Local Client Device

- OS: Arch Linux x86_64
- Kernel: 6.3.5-arch1-1
- CPU: Intel i7-6600U 3.4GHz
- GPU: Intel Skylake GT2 [HD Graphics 520]
- RAM: 8GB DDR4
- .NET Core 7
- BDN taken directly from source[1], latest commit made on March 13, 2023
- EFCore v7.0.3

Remote Server Device

- OS: Arch Linux x86_64
- Kernel: 6.3.5-arch1-1
- CPU: Intel i7-6600U 3.4GHz
- GPU: Intel Skylake GT2 [HD Graphics 520]
- RAM: 20GB DDR4
- PostgreSQL v15.3

3.2.3 A2.3 Select a Set of Measures Provided By Measuring Instruments

With RAPL there are multiple things you can measure. RAPL has the concept of domains which can all have their own energy counters.

- PSys
  - Package
    - Core
    - Uncore
    - DRAM

Figure 3.3: Topology of RAPL domains.

Figure 3.3 shows the topology of the RAPL domains. The Core domain is the actual cores, the Uncore domain is for things like the L3 cache and integrated graphics, and the DRAM domain is for the memory.

The Package domain is the entirety of the CPU socket, while the PSys domain is supposed to be the entire system.

At a glance, getting readings from the PSys domain seems like the best option, but it is rather poorly documented what it does exactly, and is more system-dependent than the Package domain. Another issue is that the PSys domain is only available on newer Intel systems, while the Package domain is available on all RAPL-enabled Intel and AMD systems, and the experiment will thus be easier to replicate on other hardware.

So the data we will be measuring is energy use from the Package domains of both client and server machine, and the execution time gathered by BDN on the client machine.

3.2.4 A2.5: Obtain the Baseline Energy Consumption of the DUTs

The baseline energy consumption is the energy consumption of the DUTs when they are idle. In order to determine the baseline energy consumption of both the client and server machine, the RAPL Reader application is run on both machines for 8 minutes, taking measurements every 10ms.

Figure 3.4 and Figure 3.5 each show three graphs of the energy usage over time, one on the client and the other on the server machine. Each graph also shows details about the data - the min, max, interquartile range, median, mean, and the standard deviation. The first graph in each set is the raw data. The second graph shows the same power usage over time, but after decreasing the granularity by a factor of 10 - measuring every 100ms. The third graph shows the same power usage over time, but after removing outliers based on the Median Absolute Deviation (MAD).

The graphs in each set are comparable by the X axis. The X axis values in the first graph is time passed in increment of 10ms, and the second and third graph is time passed in increment of 100ms, so even if the values differ - they represent the same amount of time passed.

The raw data, taken at a granularity of 10ms, has very high fluctuation - a lot of the values are 0, and a lot of the values are approximately twice the mean - this suggests that a granularity of 10ms is too fine for the DUTs, at least under baseline load. The second and third graphs in each set show a realistic approximation of the energy use. There are noticeable fluctuations on each device where the energy usage increases by approximately 0.15W. Unfortunately, we could not determine what exactly causes this fluctuation.

After removing outliers - the mean energy use of the client device is 0.52W with a standard deviation of 0.05, and the mean energy use of the server device is 0.54W with a standard deviation of 0.06.

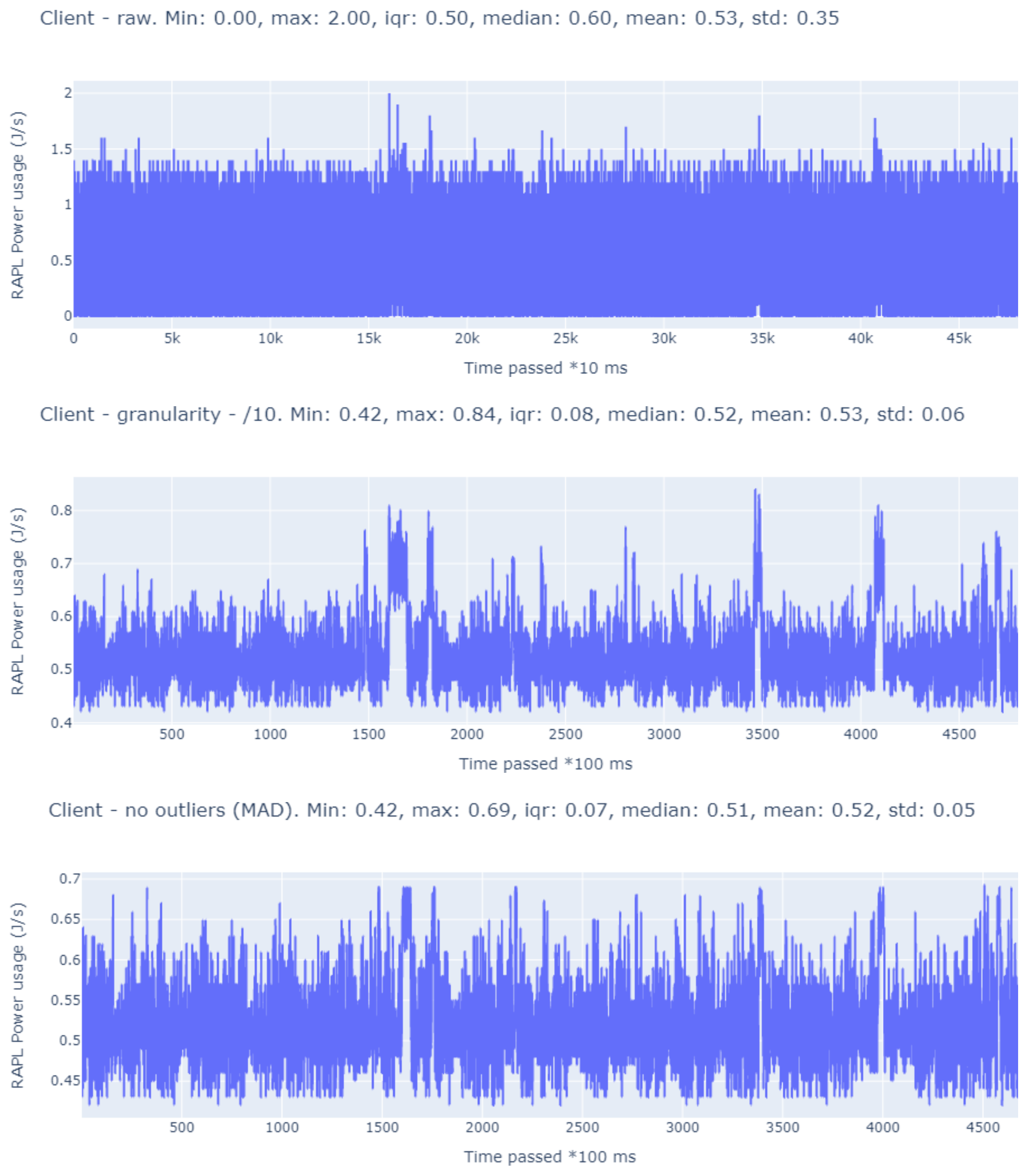

Figure 3.4: The baseline energy consumption of the local client machine.

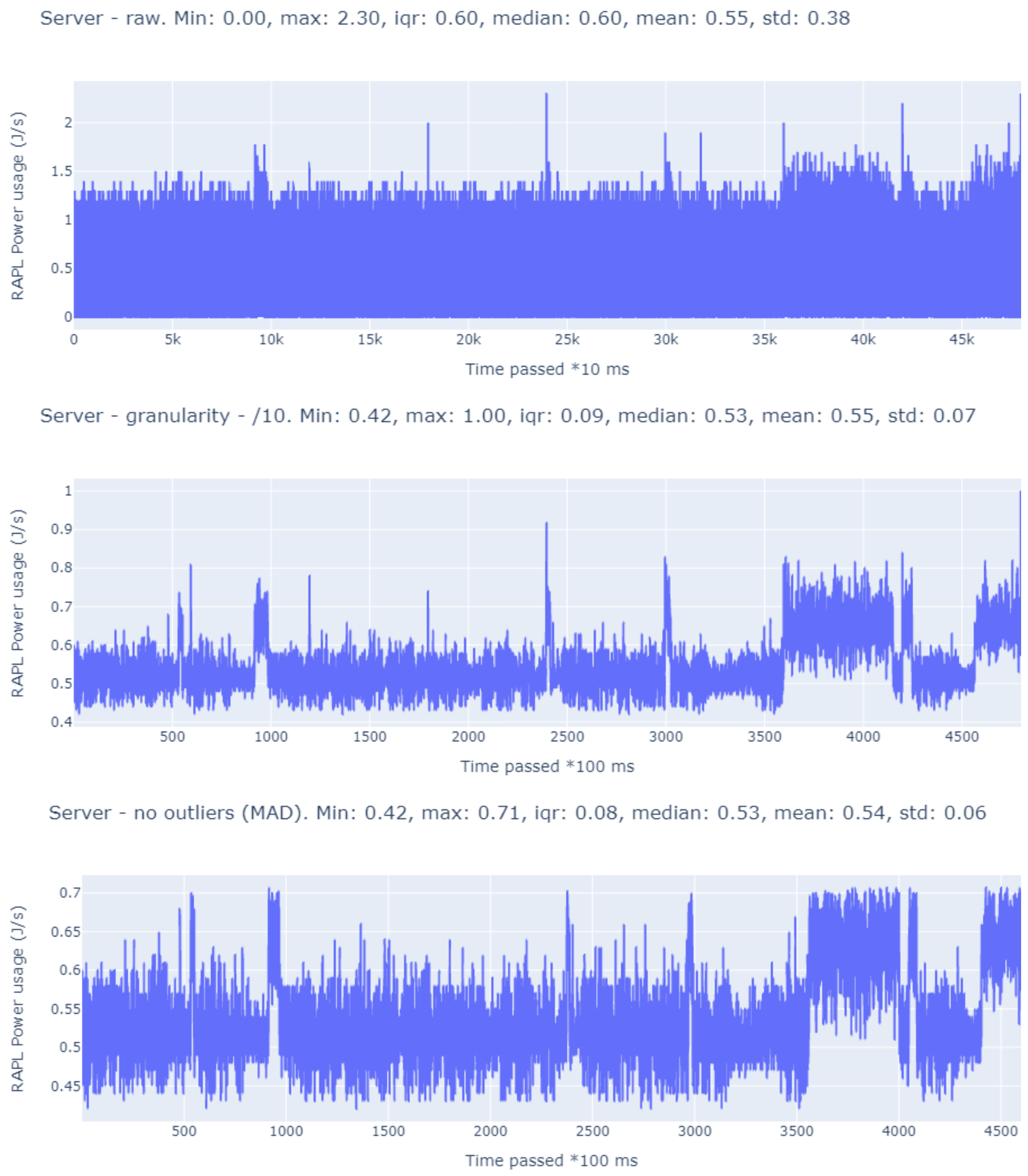

Figure 3.5: The baseline energy consumption of the remote server machine.

3.3 Phase III: Measurement Environment Preparation

In order to eliminate as much overhead as possible, the DUTs are running on a minimal installation of Arch Linux. Before performing the measurements, the DUTs are restarted and all unnecessary background processes are closed.

3.3.1 A3.2: Determine number of repetitions of measurements

BDN has built-in heuristic methods for determining the amount of repetitions of measurements. A single run of the benchmarked method in BDN terms is called an operation. In the pilot stage of the benchmark, BDN determines the amount of operations per iteration[22]. By taking an in-depth look at the source code[1], we have found that, by default, the operation count is doubled, if an iteration takes less than 500ms, or has a higher relative error than 0.02. Iteration count is determined in the workload stage. By default, iterations keep running until at least 10 iterations are complete and the max relative error is less than 0.02, or until the max amount of iterations is reached - 100.

Due to the involvement of network traffic in our benchmarks, the execution time is not very stable. While the operations per iteration count varies, the iteration count always maxes out at 100 iterations in all of our experiments. The iterations only stop between the min and the max if we run the benchmark with client and server on the same machine - eliminating network traffic.

3.4 Phase IV: Perform the Measurements

In this phase the experiments are performed and the results are gathered.

3.4.1 A4.1 Measure the Energy Consumption

Before the experiments are performed, the DUTs are prepared as described in section 3.3. Afterwards, the prepared BDN benchmark suite is executed on the Client device. The energy measurements are gathered, processed and exported automatically as a part of the benchmarking process.

3.4.2 A4.2 Collect the Raw Data

The raw data is collected automatically at a frequency of 10ms by RAPL Reader, which is automatically started and stopped by the benchmarking process.

3.5 Phase V: Test Case Data Analysis

After each benchmark, the RAPL Diagnoser (see subsection 4.4.4) processes the raw energy efficiency data and calculates the mean energy usage over time (W) and the mean energy usage per operation (J) on both the client and the server device. BDN automatically removes the iterations that are time based outliers from the actual result using Tukey outlier detector[23]. The RAPL Diagnoser discards these iterations as well. Moreover, the RAPL Diagnoser uses the same Tukey outlier detector for finding and removing outliers based on the calculated wattage on each iteration.

3.6 Phase VI: Software Entity Data Analysis

The final results are shown in figures 3.6 - 3.14. Note that the power consumption and wattage on the Remote machine is affected by the total execution time of the query. Due to the way the benchmarks are implemented (see chapter 4), the energy on the server device is measured from the start to the end of the benchmarked method on the client device, not the actual start and end of the query processing on the server device. Therefore, the database idle time while waiting for queries is also measured, which affects the final results. The following is a discussion of the effects of each of the settings listed in the Test Cases part of section 3.1.

NoTracking

Due to the behaviour of the *NoTracking* setting, the expectation was that it will produce energy savings when used with the Tracks query - because there are no join operations and therefore no duplicate data, and because the results do not need to be loaded into the context and can just be returned as a list. However, the NoTrack setting has produced a negative result in each of the experiments, increasing the energy cost of each operation on the client machine by 20 to 300%. Though, as expected, the energy increase is indeed much more noticeable when used with the Tracks and Albums query - which contains a join operation - resulting in the same album being recreated every time it is returned as a part of a track.

Load and ToList

There is no difference between the *Load* and *ToList* settings that is outside of the margin of error. This is expected, as *Load* and *ToList* has very similar behaviour, with the difference being that *ToList* returns the results as a list, right after *Load*-ing them.

Single and Split query

The *Split* query, in comparison to *Single* query, appears to increase the energy consumed on the client device and reduce the wattage on the server device. We believe that this is due to the fact that the *Split* query takes a longer amount of time due to more database round-trips, and some processing is moved from the server to the client. The total amount of energy consumed on the server device remains the same, but *Split* query is more expensive on the local device.

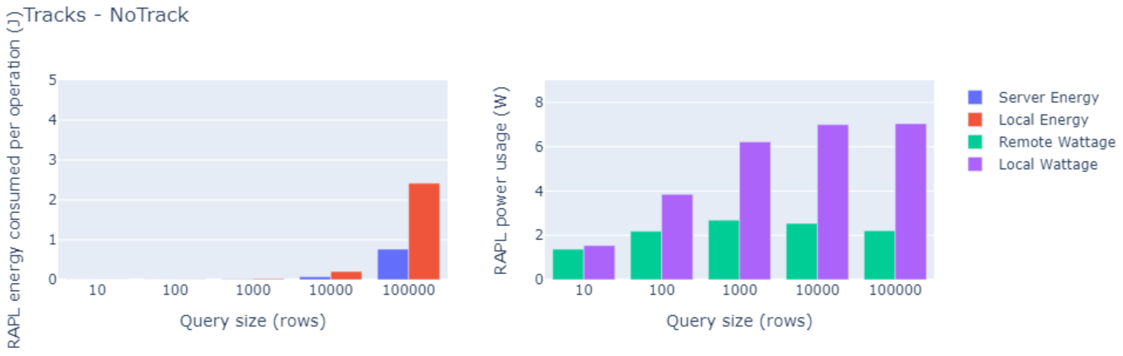

Figure 3.6: The energy efficiency of the Tracks query and NoTracking setting.

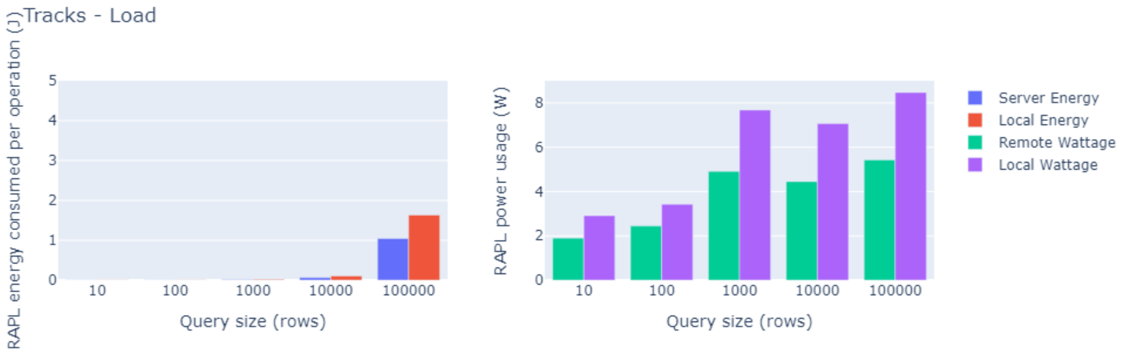

Figure 3.7: The energy efficiency of the Tracks query and Load setting.

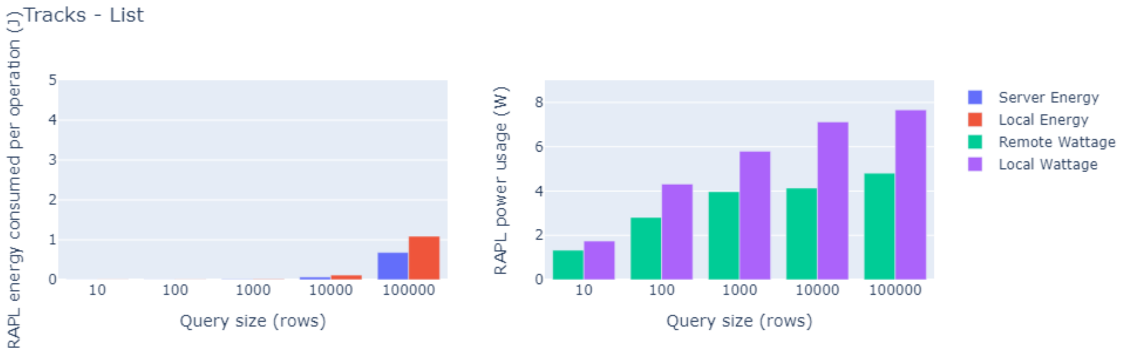

Figure 3.8: The energy efficiency of the Tracks query and ToList setting.

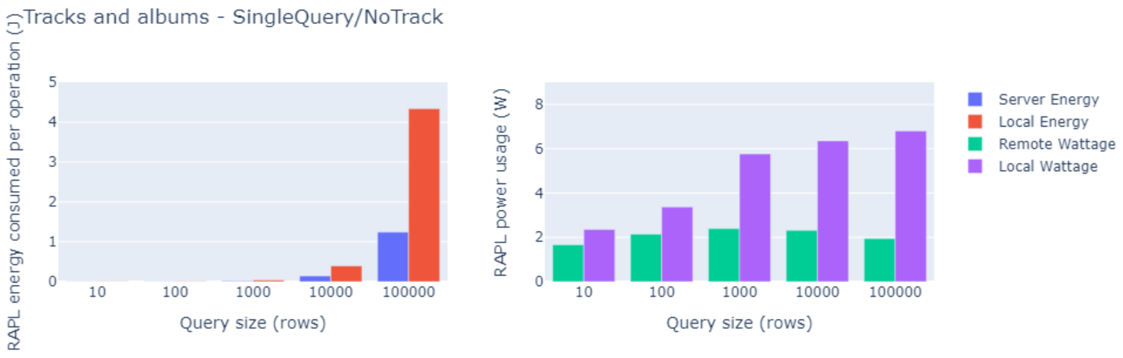

Figure 3.9: The energy efficiency of the Tracks and Albums query and NoTracking/Single query setting.

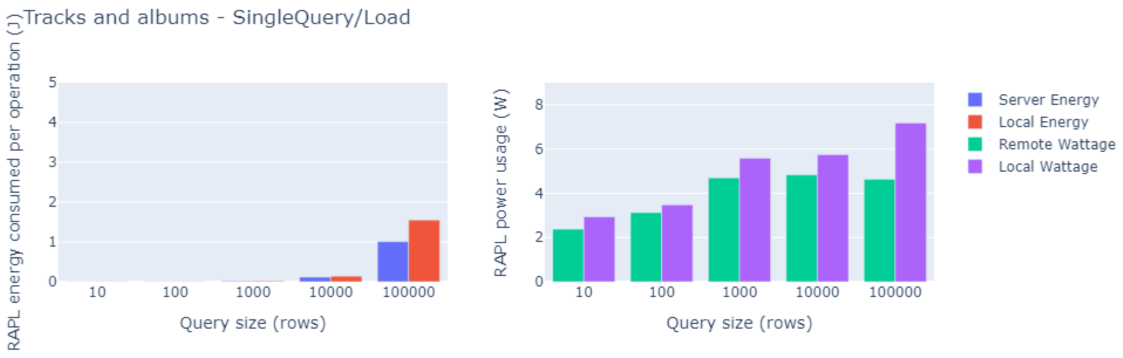

Figure 3.10: The energy efficiency of the Tracks and Albums query and Load/Single query setting.

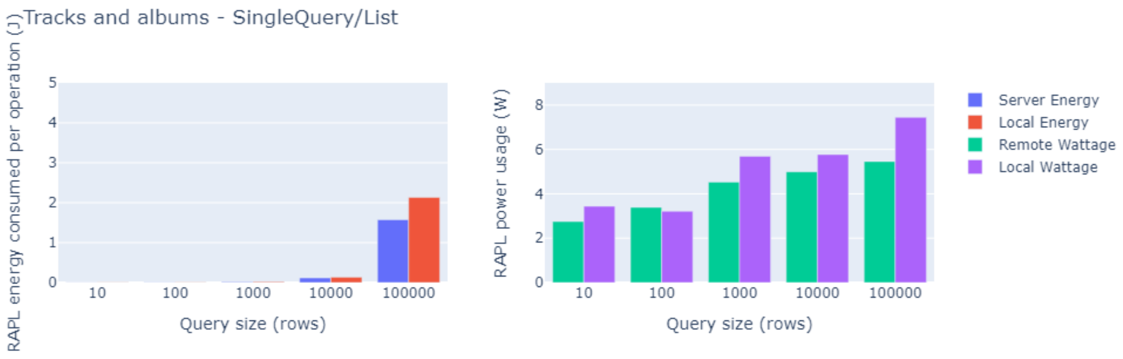

Figure 3.11: The energy efficiency of the Tracks and Albums query and ToList/Single query setting.

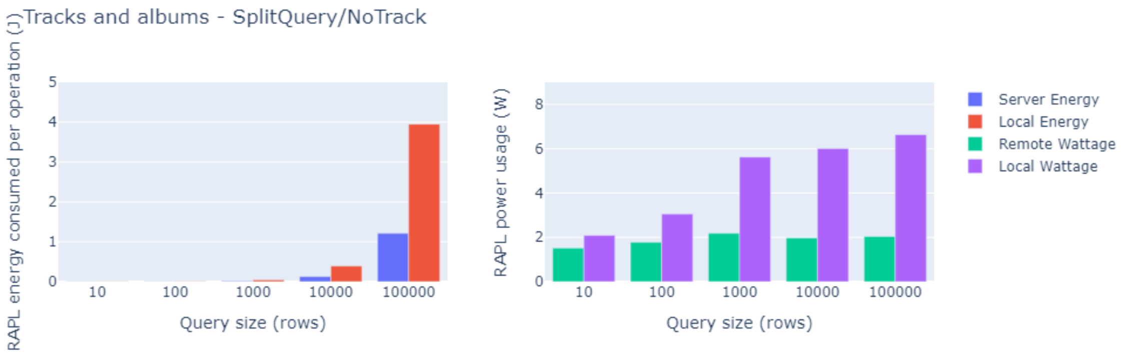

Figure 3.12: The energy efficiency of the Tracks and Albums query and NoTracking/Split query setting.

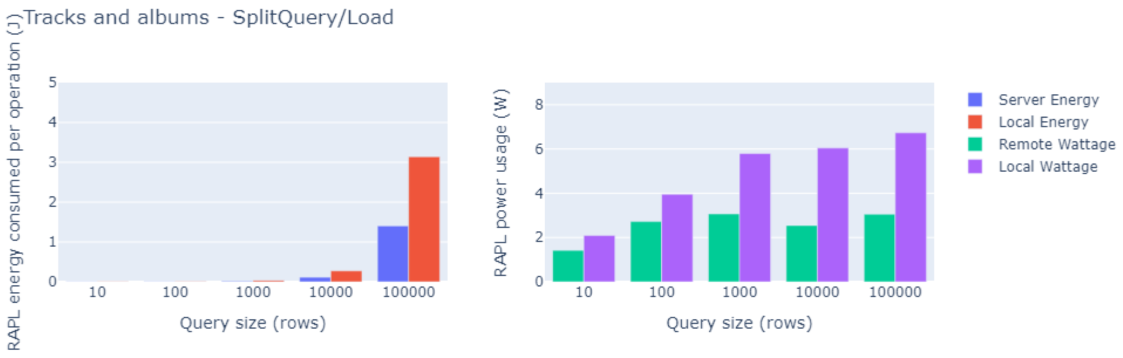

Figure 3.13: The energy efficiency of the Tracks and Albums query and Load/Split query setting.

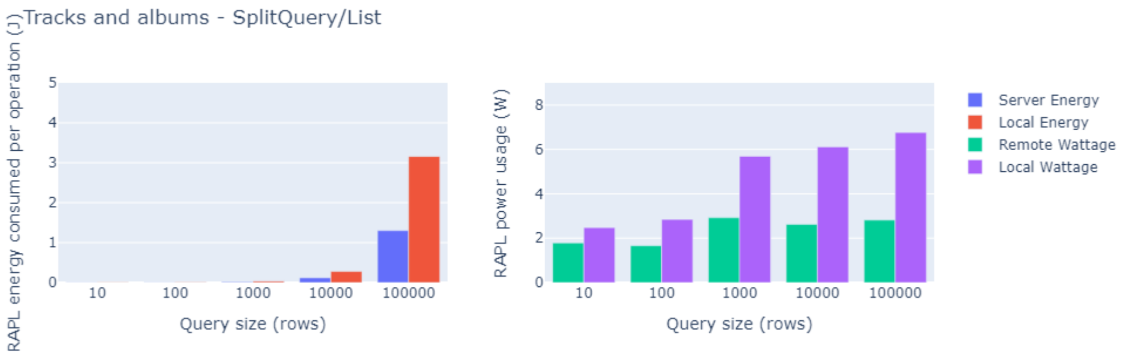

Figure 3.14: The energy efficiency of the Tracks and Albums query and ToList/Split query setting.

3.7 Phase VII: Reporting the Results

3.7.1 A7.1: Carry Out the Laboratory Package

The Laboratory Package (LP) has the role of making our experiments reproducible. Therefore, it includes both all the raw data, but also code and configuration to enable other researchers to reproduce the experiments. All of that is attached as a ZIP archive.

Chapter 4

Implementation Details

The tools that are used or created for this project are described in this chapter in greater detail. This includes the tools that are used for seeding the test database, an application for taking RAPL measurements, and a RAPL integration for BDN.

4.1 Database Seeding

As was discussed in section 3.1, we will need to have more data in our database than there is by default in Chinook. Therefore we will need to generate more data that fits into the Chinook schematic that is still somewhat representative of real data. For that we are writing a seeder program that will generate real-looking data under the scheme using the [24] framework, which extends the popular Bogus[25] framework.

In broad strokes, the seeder program works in the following way:

1. Open database connection.
2. Generate data adding onto existing data also including existing data.
3. Save generated data to database.

To create new data with AutoBogus, you use *Faker*s. The default *Faker* should be good enough for generating most simple types, but we use overrides to ensure the data fits the database schema.

```
 1  public class AlbumFaker
 2  {
 3      public static Faker<Album> GetGenerator(Artist artist)
 4      {
 5          return new Faker<Album>()
 6              .RuleFor(x => x.Title, x => x.Name.JobArea())
 7              .RuleFor(x => x.AlbumId, 0)
 8              .RuleFor(x => x.ArtistId, artist.ArtistId)
 9              .RuleFor(x => x.Artist, artist)
10              .RuleFor(x => x.Tracks, (_, x) => TrackFaker.Generate(x));
11      }
12
13      public static List<Album> Generate(Artist artist)
14      {
15          var gen = GetGenerator(artist);
16          var rnd = new Random();
17          return gen.Generate(rnd.Next(1, 10));
18      }
19  }
```

Listing 1: Implementation of *AlbumFaker*.

GetGenerator returns a faker with the rules set in lines 5-10. Notice how on line 7, the *AlbumId* is always set to 0. This is because the actual ID is determined by the database when it gets committed. On line 10, the generation of actual tracks for the album is delegated to another faker.

Generate simply takes an *Artist* as an argument, and creates between 1 and 10 albums for it.

The rest of the fakers follow the same pattern, and each class in the Chinook domain has its own faker.

4.2 Benchmarks

The benchmarks are implemented in the *SingleSplitQuery* class, each as their own method.

```
[Benchmark]
[ArgumentsSource(nameof(QuerySize))]
public async Task PlaylistsTracks_Single_NoTrack(int amount)
{
    await TracksWithAlbum()
        .AsSingleQuery()
        .AsNoTracking()
        .Take(amount)
        .ToListAsync();
}
```

Listing 2: *PlaylistTracks_Single_NoTrack* benchmark. All the benchmarks follow this structure.

Listing 2 shows the structure of all the benchmarks. They all have the BDN *Benchmark* annotation to mark them as benchmarks. They all have the *ArgumentsSource* annotation so BDN automatically

```
private IQueryable<Track> Tracks() => _dbContext.Track;

private IQueryable<Track> TracksWithAlbum() => _dbContext.Track
    .Include(x => x.Album);
```

Listing 3: The two queries being used for benchmarking.

runs all the test cases with each parameter setting how many objects to fetch. The method itself is simply just running the query with the different combinations of EFCore query options.

Listing 3 shows the two queries used in the benchmarks. *Tracks* is a simple query with no joins, and *TracksWithAlbum* also includes the album, thereby performing a join.

4.3 RAPL Reader

The RAPL Reader needs to run on both machines and needs to be relatively light-weight so as to not disturb measurements.

The first design consideration here is picking an appropriate programming language. The main contenders are C#, for more seamless integration into BenchmarkDotNet, or C/C++ for the lower overhead. Another thing to consider when picking a programming language is where there are existing libraries for taking RAPL measurements. While we were unable to find a C# library, we were able to find a C library called RAPLCap[26] [27]. Because of library availability, we chose to make the implementation of the RAPL Reader in C++. [1]

The basic control flow of the program is described in subsubsection 3.2.1.1 and shown in Figure 3.2.

Due to the need to eliminate overhead, the results are saved to disk rather than transmitted directly over STDIN or similar. This is mainly because it is easier to do for the remote monitor, but also because keeping a connection alive could potentially interfere with measurements. The results can then later be fetched by the BDN monitor.

4.3.1 Signal Handling

Communication between BDN and the RAPL Reader is handled using Portable Operating System Interface (POSIX)-signals. This is because we only plan to support Linux systems anyway, and POSIX-signals are both easy to receive and send between processes. They can be sent using either keyboard shortcuts or by using the *kill* system call.

[1]While using the Linux powercap interface directly would have worked, we did not have a supported machine yet as of writing this program, and thus had difficulties getting a complete understanding of the interface.

```
 1  namespace
 2  {
 3      volatile std::sig_atomic_t signalCode;
 4  };
 5
 6  void signal_handler(int signal)
 7  {
 8      signalCode = signal;
 9  }
10
11  int main(int argc, char* argv[])
12  {
13      std::signal(SIGINT, signal_handler);
14  ...
```

Listing 4: Code snippets showing how signals are handled.

Listing 4 shows the boilerplate for adding signal handling. Line 1-4 defines a global variable containing the signal code that is checked later in the program. It is volatile so the compiler does not try to re-order reads and writes to it, or optimize them away entirely, and is declared as an integer with ensured atomic writes.

On line 13, the simple signal handler shown from line 6-9 is set to be the handler for the *SIGINT* signal. *SIGINT* was mainly chosen because it is very easy to send using keyboard shortcuts on most systems using CTRL+C, for easier development and for ease of use when using the RAPL Reader without using BDN.

4.3.2 Initialisation

```
1  int main(int argc, char* argv[])
2  {
3      std::signal(SIGINT, signal_handler);
4  
5      std::string filename = argv[1];
6  
7      raplcap rc;
8      raplcap_init(&rc);
9  
10     auto measurements = new std::vector<measurement>();
11 
12     auto n_packages = raplcap_get_num_packages(&rc);
13 
14     auto dies = new std::vector<die>();
15 
16     for (int i = 0; i < n_packages; i++)
17     {
18         auto n_die = raplcap_get_num_die(&rc, i);
19         for (int j = 0; j < n_die; j++)
20         {
21             dies->emplace_back(die(i, j));
22         }
23     }
24 ...
```

Listing 5: Initialisation code.

Listing 5 shows the parts of the main function dedicated to initialise the RAPLCap library and prepare for taking measurements.

Lines 10-23 are mainly there for supporting more exotic configurations, such as multi-CPU servers or platforms with multiple CPU dies per package. It simply makes a list of all dies in the system, that will then later be iterated on when taking measurements.

4.3.3 Taking Measurements

After waiting for the starting signal of the benchmark, the actual measurement loop begins.

```
1  auto time_next_measurement = std::chrono::system_clock::now();
2  
3  while (true)
4  {
5      time_next_measurement += std::chrono::milliseconds(10);
6  
7      if (signalCode)
8      {
9          break;
10     }
11 
12     for (auto die: *dies)
13     {
14         measurements->emplace_back(
15                 measurement{die,
16                             raplcap_pd_get_energy_counter(&rc, die.package_n, die.die_n, RAPLCAP_ZONE_PACKAGE),
17                             raplcap_pd_get_energy_counter(&rc, die.package_n, die.die_n, RAPLCAP_ZONE_CORE),
18                             raplcap_pd_get_energy_counter(&rc, die.package_n, die.die_n, RAPLCAP_ZONE_UNCORE),
19                             raplcap_pd_get_energy_counter(&rc, die.package_n, die.die_n, RAPLCAP_ZONE_DRAM),
20                             raplcap_pd_get_energy_counter(&rc, die.package_n, die.die_n, RAPLCAP_ZONE_PSYS),
21                             static_cast<uint64_t>(std::chrono::duration_cast<std::chrono::milliseconds>(
22                                     std::chrono::system_clock::now().time_since_epoch()).count())
23                 });
24     }
25 
26     std::this_thread::sleep_until(time_next_measurement);
27 }
```

Listing 6: Measuring loop.

The measuring loop is shown in Listing 6.

At line 1, the *time_next_measurement* variable is initialised with the current time, and on line 5 the amount of time between measurements are added to it. It is important that the time is initialised outside the loop, and that no *now* calls are made inside, because otherwise it will risk introducing drift.

Lines 7-10 end the loop if the end of benchmark signal has been received. This illustrates the need for the signalCode variable to be volatile, because if not, the code would most likely be removed by the compiler, as it will never get changed in any reachable code from this function. It is instead changed via interrupt.

Lines 12-24 takes takes the actual measurements. As discussed in subsection 4.3.2, it iterates over all dies found on the system. For most personal computers[2] this will most likely just be one die. It gets the energy counters of all RAPL zones, and ones not supported on the system will be -1. Seeing as most zones are either not of interest to the user, or unavailable on the system (e.g. DRAM on AMD systems), there is some unnecessary overhead here. A potential improvement could be specifying zones of interest as a command-line argument when starting the RAPL Reader.

In the end, at line 26, the program sleeps until the next measurement is due. The real-time nature of these periodic measurements should be guaranteed by the use of *sleep_until* rather than a simple sleep.

After the measuring loop exits, the results are saved to disk and the RAPL Reader exits.

4.4 BenchmarkDotNet Integration

The research questions in this project have a slightly conflicting design approach. BDN does not support simultaneously benchmarking a client and a server machine, which is required to answer RQ2. At the same time, in order to answer RQ1 and create a clean RAPL integration for BDN, it is important not to introduce any breaking changes and make as few changes and overhead as possible to the benchmarking process, while also maintaining a good first user experience. Unfortunately, adding the capability for BDN to benchmark multiple machines at the same time, while also conforming to the listed requirements, cannot be achieved in the scope of this project. Therefore, in favor of RQ1, the focus is to extend the current BDN process to allow energy measurements just on the single machine. Then, in order to answer RQ2, a custom solution for measuring both devices is added for the scope of this project only.

4.4.1 Integration Design

BDN is composed of two main parts: BDN Runner[28] and BDN Engine[29]. The BDN Runner is the main benchmarking process that builds each benchmark, executes them in a separate process, and analyses the results. The BDN Engine is the child process of BDN Runner that executes the benchmark and reports back the results.

Diagnosers [30] are used in implementing features to extend the benchmarking capability of BDN. A diagnoser provides access to the results of the benchmark, events such as *BeforeAnythingElse*, *BeforeActualRun*, *AfterActualRun* etc., and allows outputting additional metrics to the final result. This interface is perfect for implementing a RAPL Diagnoser that adds RAPL measurements to the benchmark result.

To separate the logic of the RAPL Diagnoser and the way that RAPL measurements are taken, a new interface (RAPL Monitor) is created. This provides a layer of abstraction that can be used to differentiate between experiment setups - separate implementation for RQ1 and RQ2, as mentioned in section 4.4.

The resulting architecture is shown in Figure 4.1. First, the BDN Runner creates the benchmarking context and sequentially spawns a BDN Engine for each benchmark. The BDN Engine sends signals back to the BDN Runner for benchmark start and stop, which are passed to all the diagnosers, including RAPL Diagnoser. The RAPL Diagnoser then triggers all the RAPL Monitor instances that it contains to start/stop taking RAPL measurements. In this instance, there are separate RAPL Monitors for the local and remote machine, which are responsible for interfacing with a RAPL Reader, described in section 4.3. Once the benchmark is complete, the RAPL Diagnoser processes the results and outputs a new RAPL energy metric for the BDN Runner.

[2]Not necessarily servers or powerful workstations.

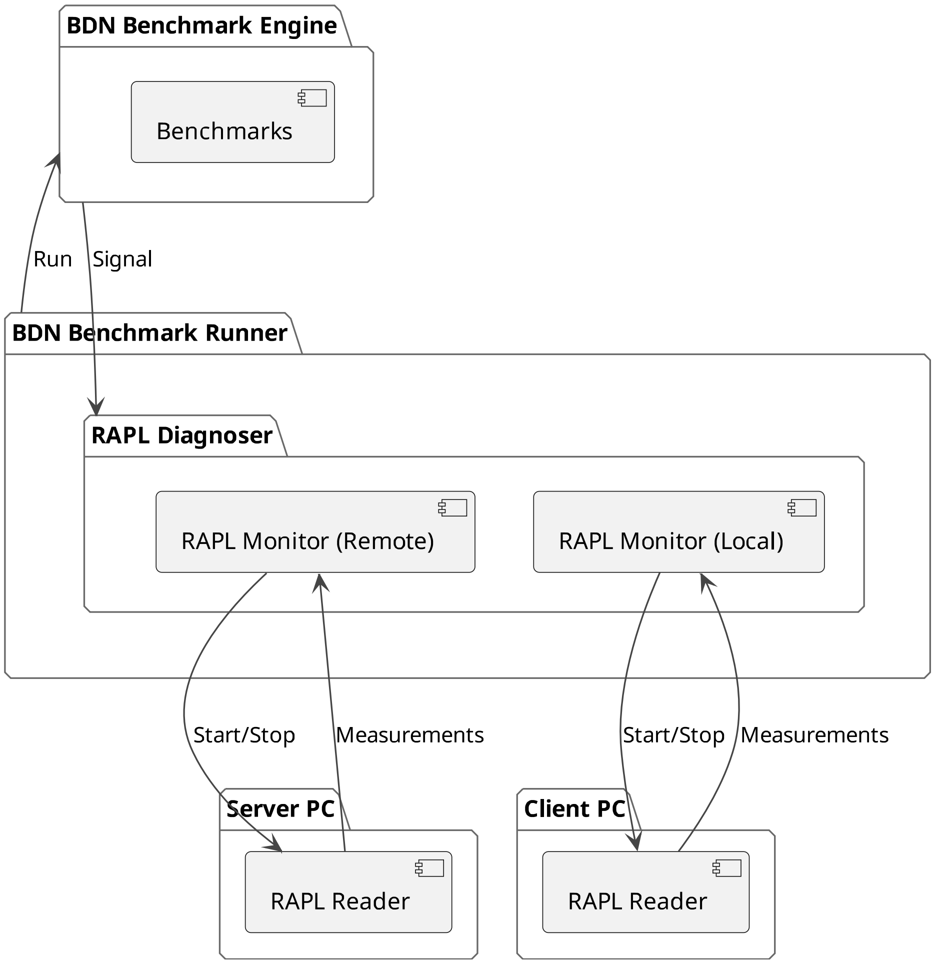

Figure 4.1: RAPL integration with BDN.

4.4.2 Aligning RAPL Measurements With the Benchmark

One of the challenges in the development of this tool is making sure that only the RAPL measurements taken during the execution of each benchmark iteration are taken into account when calculating the energy usage. The measurements that are taken before, after, or in-between iterations should not be taken into account, as they are not part of the benchmarked method.

In the current version of BDN, there is no exact time stamp taken, nor any event fired, at the start and end of each iteration. Therefore, this issue must be addressed with a custom implementation. There are multiple ways to achieve this:

1. Add a timestamp to BDN Engine for start and end of each benchmarking iteration. This requires the RAPL Reader app to continuously take measurements and their timestamp. The measurements that are taken outside of the iteration timestamps can then be discarded.
2. Output an event to BDN Runner, which is then consumed by the BDN diagnosers. The RAPL Diagnoser then implements a RAPL Monitor, in order to signal RAPL Reader.
3. Read the RAPL measurement directly from PowerCap, at the start and end of each iteration.

The first solution involves running RAPL Reader for the entire benchmarking process, even between iterations, and removing the out-of-scope measurements when processing the results. The second solution provides more granular signals to the RAPL Reader in order to control precisely when RAPL measurements should be taken. It would also result in an additional layer of interaction for the diagnosers already present in BDN - allowing them to use the same newly implemented signal. Both of these options modify the interaction between the external benchmarking process and the main process of BDN.

The third option could be implemented directly in BDN Engine and would require the least amount of changes to the BDN infrastructure. This option has the least amount of overhead, but it cannot be used in this experiment, because it does not implement RAPL Monitor, which is required for measuring the energy use of a remote device.

Therefore, option 1 or 2 must be chosen. The most reasonable way for implementing the first solution is by expanding the *Measurements.cs* class of BDN that contains the results of a benchmark iteration. The iteration start and end timestamp is then processed the same way as execution time - by passing it through the console, see Listing 8 and Listing 7. The *Measurement.cs* results are passed to the diagnosers, and therefore to the RAPL Monitor. The BDN Engine is modified to record the start and end timestamp, see Listing 9.

```
WorkloadResult 1: 2 op, 625701100.00 ns, 312.8505 ms/op
WorkloadResult 2: 2 op, 623259900.00 ns, 311.6300 ms/op
WorkloadResult 3: 2 op, 630686000.00 ns, 315.3430 ms/op
WorkloadResult 4: 2 op, 630164200.00 ns, 315.0821 ms/op
WorkloadResult 5: 2 op, 641134900.00 ns, 320.5675 ms/op
WorkloadResult 6: 2 op, 615457100.00 ns, 307.7285 ms/op
WorkloadResult 7: 2 op, 638407400.00 ns, 319.2037 ms/op
WorkloadResult 8: 2 op, 641247700.00 ns, 320.6239 ms/op
WorkloadResult 9: 2 op, 617673300.00 ns, 308.8367 ms/op
```

Listing 7: Example of the original BDN console output

```
WorkloadResult 1: 2 op, 634808200.00 ns, 1684327721854 ss, 1684327721219 es, 317.4041 ms/op
WorkloadResult 2: 2 op, 651551000.00 ns, 1684327722544 ss, 1684327721892 es, 325.7755 ms/op
WorkloadResult 3: 2 op, 646801400.00 ns, 1684327723230 ss, 1684327722583 es, 323.4007 ms/op
WorkloadResult 4: 2 op, 665663600.00 ns, 1684327724644 ss, 1684327723978 es, 332.8318 ms/op
WorkloadResult 5: 2 op, 650799700.00 ns, 1684327725345 ss, 1684327724694 es, 325.3999 ms/op
WorkloadResult 6: 2 op, 630605100.00 ns, 1684327726029 ss, 1684327725398 es, 315.3025 ms/op
WorkloadResult 7: 2 op, 640630800.00 ns, 1684327726694 ss, 1684327726053 es, 320.3154 ms/op
WorkloadResult 8: 2 op, 630216200.00 ns, 1684327727375 ss, 1684327726744 es, 315.1081 ms/op
WorkloadResult 9: 2 op, 636206500.00 ns, 1684327728039 ss, 1684327727402 es, 318.1033 ms/op
```

Listing 8: Example console output that is returned by the BDN benchmark process, after implementing option 1. The units marked with 'ss' (start stamp) and 'es' (end stamp) are added.

In order to implement option 2, a diagnoser can be made to listen for BDN *EngineEventSource* events by using a *PipeReader*. Inspiration for this implementation comes from the *EventPipeProfiler* [31] that is already built into BDN for the purposes of profiling benchmarks. The BDN Engine already contains events for tracing the iteration start and stop (see Listing 9, the event is triggered on lines 4 and 15), but there is no built-in way to listen for it.

```
1  // Code from BenchmarkDotNet.Engines.Engine.RunIteration method.
2  ...
3  if (EngineEventSource.Log.IsEnabled())
4      EngineEventSource.Log.IterationStart(data.IterationMode, data.IterationStage,
        ↪  totalOperations);
5  Span<byte> stackMemory = randomizeMemory ? stackalloc byte[random.Next(32)] :
   ↪  Span<byte>.Empty;
6
7  // Measure
8  var start = Clock.GetTimestamp();
9  var clock = Clock.Start();
10 action(invokeCount / unrollFactor);
11 var clockSpan = clock.GetElapsed();
12 var end = Clock.GetTimestamp();
13
14 if (EngineEventSource.Log.IsEnabled())
15     EngineEventSource.Log.IterationStop(data.IterationMode, data.IterationStage,
        ↪  totalOperations);
16 ...
```

Listing 9: Snippet of code where BDN Engine executes a benchmark iteration. The highlighted lines are part of implementing solution 1. If solution 3 was implemented, the highlighted lines would be replaced with reading RAPL energy from PowerCap. Lines 3-4, 14-15 are where BDN Engine fires an event for solution 2. Line 5 is for optional BDN memory benchmarking - it is unrelated to this project.

In conclusion, option 2 appears to be the best overall solution. Option 1 has the side effect of polluting the console output with data that is not interesting for the human reader. With option 2 it is also possible to modify the diagnoser in such a way that does not require the RAPL Reader to run continuously, or to omit the RAPL Reader entirely and take the energy reading directly from PowerCap, similar to option 3.

Initially, in order to obtain a minimum working product as quickly as possible, we have implemented option 1 before considering the other options. Even though option 2 might be better as an integration, option 1 suits the requirements of this project as well. Therefore, in order to save time and focus the development effort elsewhere, we are using option 1.

4.4.3 Modifications to BenchmarkDotNet

Since most of the implementation for a RAPL integration can be done by implementing a diagnoser (see subsection 4.4.4), very few modifications need to be made to the BDN source code. All of these modifications are required to solve the problem discussed in subsection 4.4.2.

Listing 9 shows where modification is required in order to record the timestamp of the start and end of a benchmark iteration. To pass this data back to the RAPL Diagnoser, the *Measurement* class of BDN must be extended to include it. This involves adding the variables to the class, and also updating its *ToString* and *Parse* methods to include the new variables.

4.4.4 RAPL Diagnoser

Diagnosers[30] are a component of BDN that can attach to a benchmark in order to provide some additional information. Therefore, a new diagnoser - RAPL Diagnoser is created to handle the taking and reporting of RAPL energy measurements.

Two functions in the BDN diagnoser interface[32] are the most relevant: *Handle* and *ProcessResults*. The *Handle* event is called when certain stages of the benchmarking process are reached, such as *BeforeAnythingElse*, *BeforeActualRun*, *AfterActualRun* etc. The *ProcessResults* is called after the benchmark is complete, and it accepts some output in the form of metrics, that are then presented in the benchmark results.

The RAPL Diagnoser uses the *Handle* method to invoke the RAPL Monitors at the start and end of each benchmark. The monitors then start/stop recording RAPL measurements by signaling the RAPL Reader.

Once the benchmark is finished and the *ProcessResults* method is invoked, the RAPL Diagnoser fetches the measurements from each of its RAPL Monitors and calculates the results.

As discussed in subsection 4.4.2, the RAPL measurements that are taken in time-frames outside of the actual benchmark must be discarded. This is done in the scope of the RAPL Diagnoser's *ProcessResults* method, as this is the only area in the code where both the RAPL and BDN measurements are available. The RAPL results are a flat list of RAPL readings and the timestamp that they are taken on (see section 4.3). The BDN results contain the duration of each iteration, and after adding them - the start and end timestamp of each iteration. Using the timestamps, the RAPL Diagnoser groups the RAPL measurements by iteration (Listing 10).

```
// Implementation relies that both input lists are ordered by timestamp
var raplEnumerator = raplMeasurements.GetEnumerator();
raplEnumerator.MoveNext();
foreach (var iteration in iterations)
{
    var raplGroup = new List<RaplMeasurement>();
    while (raplEnumerator.Current.TimeStamp < iteration.StartTimeStamp)
    {
        raplEnumerator.MoveNext();
    }
    while (raplEnumerator.Current.TimeStamp <= iteration.EndTimeStamp)
    {
        raplGroup.Add(raplEnumerator.Current);
        raplEnumerator.MoveNext();
    }
    yield return raplGroup;
}
raplEnumerator.Dispose();
```

Listing 10: RAPL Diagnoser *GroupByIteration* method. Grouping RAPL measurements by iteration and discarding measurements that are out of range.

Once the measurements are grouped, the diagnoser removes the outliers using Tukey outlier detection method and calculates the average RAPL energy use per operation and per second. The resulting values are then returned as a metric for each of the monitors. The whole *ProcessResults* method is shown in Listing 11

```csharp
// filter only the result iterations
var iterations = results.Measurements
    .Where(x => x.Is(IterationMode.Workload, IterationStage.Result))
    .ToList();
var operationsPerIteration = iterations.First().Operations;
foreach (var monitor in monitors)
{
    // grouping rapl measurements by iteration
    var raplMeasurements = monitor.ProcessResults();
    var raplByIteration = GroupByIteration(raplMeasurements, iterations,
    ↪  monitor.GetType()).ToList();

    // remove outliers
    var outlierDetector = TukeyOutlierDetector
        .FromUnsorted(raplByIteration.Select(x => x.Wattage()).ToArray());
    raplByIteration = raplByIteration
        .Where(x => !outlierDetector.IsOutlier(x.Wattage()))
        .ToList();

    // energy use per operation
    yield return new Metric(
        new EnergyPerOpMetricDescriptor(monitor.MonitorType, "Package"),
        raplByIteration.Select(x => x.EnergyConsumedActual()).Average() /
        ↪  operationsPerIteration
    );

    // wattage
    yield return new Metric(
        new RaplMetricDescriptor(monitor.MonitorType, "Package"),
        raplByIteration.Select(x => x.Wattage()).Average()
    );
}
```

Listing 11: RAPL Diagnoser *ProcessResults* method. Takes the results of the benchmark and produces metrics for each of its monitors. The highlighted line shows where the method shown in Listing 10 is called.

4.4.5 RAPL Monitor

The RAPL Monitor is introduced as a layer of abstraction between the RAPL Diagnoser and taking RAPL measurements (see subsection 4.4.1). Doing this allows us to create a clean implementation for measuring the energy use of a local and remote device at the same time. The RAPL Monitor interface is shown in Listing 12.

```
public interface IRaplMonitor
{
    string MonitorType { get; }
    RaplMeasurements ProcessResults();
    void Launch(string id);
    void Start();
    void Stop();
}
```

Listing 12: RAPL Monitor interface.

There are three implementations of the RAPL Monitor: a stub, local, and remote. The stub monitor is only used internally for debugging purposes and does not use actual RAPL, but simply runs a counter to simulate energy use. The local monitor spawns a child RAPL Reader (see subsubsection 3.2.1.2) process on the local machine. The remote monitor starts a RAPL Reader process on a remote machine by connecting to it through SSH.

More monitors can be added on demand that could do something different, such as when implementing the other options mentioned in subsection 4.4.2.

4.4.6 Handling RAPL Overflow

The RAPL energy register keeps incrementing as it calculates the energy consumption. This means that the value eventually overflows. The handling of this is implemented directly in the parser that processes the RAPL Reader output. This is achieved by creating a custom class called *OverflowDouble*. *OverflowDouble* has a single method called *Next*, that accepts and returns a double. It works by creating a buffer that has an initial value of 0, which it increments with the max RAPL value of 262143.32885, when it detects an overflow. The buffer is added to the input value, before returning it, see Listing 13.

```
internal class OverflowDouble
{
    private double Buffer = 0;
    private double Last = 0;

    public double Next(string value) =>
        Next(double.Parse(value, CultureInfo.InvariantCulture));

    public double Next(double value)
    {
        if (value != -1 && value < Last)
        {
            Buffer += 262143.32885;
        }

        Last = value;
        return Buffer + value;
    }
}
```

Listing 13: OverflowDouble made for handling RAPL measurement overflow.

4.4.7 Calculating Energy Consumption

Since the RAPL measurements are taken every 10ms, and the measurements that are taken outside of an iteration are removed before calculation (see Listing 10), the margins at the start and end of each iteration between the iteration start and first measurement taken, and between the last measurement taken and iteration end are unaccounted for. This results in some additional error in the measurements as that margin can range from 0 to 20ms, and the energy consumed during that time is not calculated. To solve this, the multiplier of the time difference between the start and end of the iteration and the first and last measurement taken is applied to the resulting energy consumption. The method is shown in Listing 14. This method is used by the RAPL Diagnoser, see Listing 11, line 22.

```
public static double EnergyConsumedActual(this RaplMeasurements measurements)
{
    var last = measurements.Last();
    var first = measurements.First();

    // time and energy between first and last measurement
    var energy = last.Package - first.Package;
    double time = last.TimeStamp - first.TimeStamp;

    // time between the start and end of the iteration
    double timeActual = measurements.EndTimestampActual -
    ↪  measurements.StartTimestampActual;

    var multiplier = timeActual / time;
    return energy * multiplier;
}
```

Listing 14: Projecting the energy consumption based on iteration start and end. *timeActual* is guaranteed to always be greater or equal to *time*, therefore the multiplier is always greater or equal to 1.

4.4.8 Export of Raw Energy Data Per Iteration

Because of the way we chose to integrate RAPL into BDN using a diagnoser, and because we did not have time to properly integrate our results with the statistical engine of BDN, we have to export the energy usage of each iteration to manually do our own statistics.

BDN has built-in *Exporters*, that can export benchmark results in different formats. There are several exporters, such as *RPlotExporter*, which exports plots of execution time compared to the parameters of the benchmark, or most of the other exporters, like *CsvExporter* or HtlmExporter, that exports data based on the metrics of the benchmark, i.e. summarized data after statistical analysis is performed. There are exporters that export all measurement data, but seeing as how we implement RAPL support using the RAPL Diagnoser, they have no access to energy measurements, as they rely on the *Measurement* struct, also mentioned in subsection 4.4.2, where it is used to add timestamp data. Luckily, due energy values only being needed in the BDN Engine, and not the BDN Runner, the fields do not need to be serialised, and thus do not need to clutter the console output.

Fortunately, if the *Measurement* struct is instead changed to a class, due to the way they are passed in the program, they would all become references to the same object when they are passed on down to the diagnosers, exporters, etc. This means that we can set the values in the diagnoser, and then the exporters will be able to access it.

```
switch (monitorType)
{
    case not null when monitorType == typeof(RaplMonitor):
        iteration.Energy = raplGroup.Last().Package - raplGroup.First().Package;
        break;

    case not null when monitorType == typeof(RemoteRaplMonitor):
        iteration.EnergyR = raplGroup.Last().Package - raplGroup.First().Package;
        break;
}
```

Listing 15: Code inserted into the *GroupByIteration* method. Assigns energy usage of each iteration into the *Measurement* object.

The energy values for each iteration is updated in the code shown in Listing 15, that is inserted into *GroupByIteration*. It was the easiest place to do it, because it is the place in the diagnoser where the most convenient link between *Measurement* and its RAPL data exists. The *GroupByIteration* method has been updated to take the type of monitor whose measurements are currently being processed, and then there is a switch statement on the type so it can set the appropriate energy field depending on whether it is from the local or the remote monitor.

```
...
columns.Add(new MeasurementColumn("Measurement_Operations", (summary, report, m) =>
↪  m.Operations.ToString()));
columns.Add(new MeasurementColumn("Measurement_Value", (summary, report, m) =>
↪  (m.Nanoseconds / m.Operations).ToString("0.##", summary.GetCultureInfo())));
columns.Add(new MeasurementColumn("Energy", (summary, report, m) =>
↪  (m.Energy.ToString(summary.GetCultureInfo())) ));
columns.Add(new MeasurementColumn("Energy_Remote", (summary, report, m) =>
↪  m.EnergyR.ToString(summary.GetCultureInfo())));
columns.Add(new MeasurementColumn("Power", (summary, report, m) => (m.Energy /
↪  (m.Nanoseconds / 1000000000)).ToString(summary.GetCultureInfo())));
columns.Add(new MeasurementColumn("Power_Remote", (summary, report, m) =>
↪  (m.EnergyR / (m.Nanoseconds /
↪  1000000000)).ToString(summary.GetCultureInfo())));
...
```

Listing 16: *CsvMeasurementsExporter* gets configured to output both energy and power consumption.

Listing 16 shows how the measurement data gets exported using the *CsvMeasurementsExporter*, where we can access the energy values we set in the diagnoser due to *Measurement* now being a mutable reference type.   b

Chapter 5

Discussion

Initially, the goal of this project was only to do some ORM energy efficiency benchmarking on a client and server device (RQ2). We have not tried using BDN before, and since we have heard many good things about it, we wanted to try using it for this project. After learning more about BDN, we realized that it would be possible to integrate RAPL energy efficiency measurements into it (RQ1). We believe that such an integration would be much more valuable to the community than a few benchmarks that we could make. Therefore, our focus has shifted from the benchmark to the integration.

Unfortunately, as discussed in section 4.4, BDN does not provide the capability to benchmark both a client and a server machine. We ended up having conflicting goals: making a clean energy efficiency integration for BDN, and measuring both a client and server device, which BDN cannot do. This has cost us a lot of time, because we not only had to make a BDN integration, but also make it work on separate machines. The quality of our research has suffered greatly because of this, as a lot of the time spent on 'hacks' could have been better used in other areas of the project.

The solution to this problem could have been to modify RQ2, so that our benchmarks were localized to a single machine - as intended by BDN. This could have been achieved by either benchmarking the ORM on a single machine, or coming up with different kinds of benchmarks entirely. This way we could make the best RAPL integration we could, and also use it as is for the benchmark.

5.1 Future Works

5.1.1 Towards Upstreaming

Getting any RAPL integration into BDN would greatly lower the barrier to entry for .NET developers to get into testing the energy efficiency of the software they develop. Unfortunately, our solutions has several problems that would prevent it from getting upstreamed.

We would likely need to switch to approach 2 as discussed in subsection 4.4.2, i.e. use an event-based approach to know when an iterations starts and stops instead of attaching timestamps to each *Measurement*. Not only would this stop us polluting the console output, an *IterationStart* and *IterationStop* event could be useful for other diagnosers made in the future.

The RAPL Reader code should also be moved to C#, and should not depend on RAPLCap, but rather read from the PowerCap interface. The overhead from doing readings from PowerCap through the filesystem should not be an issue, because we could certainly get away with just a before and after measurement because the continuous is only really necessary for profiling. It is not a problem that this method would not easily work for the remote RAPL Monitor, because benchmarking remote machines is likely not a feature needed for most software, and thus could be scrapped.

Considerations on how to perform statistics on the energy measurements will have to be made, and if energy measurements should be used in outlier removal, and if it is also necessary to take energy into account when deciding on how many iterations to run.

5.2 Threats to Validity

Network Data Transfer

We are not measuring the energy cost of transferring data over the network. Only the client and server machine is measured, and nothing in-between. Even though the client and server machine are on the same local network, there is still some unpredictability in how long the data transfer takes - affecting the execution time of the results. The energy usage results on the client and server machine might be affected as well, if e.g. data transfer takes more/less time, then CPU time, and consequently energy use, on each machine might be affected accordingly.

Another cause of concern rising from the network data transfers is that we do not take the energy cost of the actual network traffic into account. While it should not be too much of an issue for this experiment, due to the server and client being on the same local network (even on the same switch), our results might not be applicable to scenarios where the requests go onto the internet.

No Validation of Our Tool

Because we do not validate our tool against another measurement device, like comparing against a hardware measuring device, we cannot be certain that our results correspond to the actual energy consumption of the devices. As discussed in chapter 2, RAPL itself should be accurate enough, but we cannot know for certain that our data processing is valid.

Server Idle Time

Because of the way the energy measurements are being performed, the time where the server idles after executing the query, but the client is not yet done, we capture of a lot of measurements where the server is not doing any execution. This means that the energy use for the server will be inflated, while the wattage is seemingly reduced. It would be an improvement to only count the server energy data from when it is serving the query, thus getting more accurate measurements.

Micro-benchmarks

Because we use micro-benchmarks, we cannot know if our results will generalize to real-world applications. The queries we run are also quite simple, so they themselves may also be a problem if we want to generalize the results. For example, we do not know if these results also apply to queries with more complex joins.

Hardware

We ran our benchmarks on two low-power laptops, which is certainly not reminiscent of the actual servers that a production database would run on in a data center.

Chapter 6

Conclusion

We managed to make a RAPL integration for BenchmarkDotNet that works for our case study. The integration is not of sufficient quality to upstream into BenchmarkDotNet, but we have outlined ways in which it could be done. So to answer RQ1, it is possible to create a RAPL integration for BenchmarkDotNet.

We have found that the *NoTrack* option for Entity Framework Core always has a negative impact on the energy consumption of the client device. The *Load* and *ToList* settings appear to be within margin of error, which shows that there is little cost to adding the objects to a list compared to just loading them into the database context.

The *Split* query setting, in comparison to *Single* query, appears to increase the energy consumption on the client device. It also appears to reduce the wattage on the server, but we suspect that this is only due to the operation taking longer in general, as the server idles while it is waiting for the client to complete execution. The results on *Split* and *Single* are mostly inconclusive, and further testing would have to be made.

Bibliography

[1] *BenchmarkDotNet - source code.*
URL: `https://github.com/dotnet/BenchmarkDotNet` (visited on 06/06/2023).

[2] *Overview of Entity Framework Core - EF Core.*
URL: `https://learn.microsoft.com/en-us/ef/core/` (visited on 05/30/2023).

[3] *Chinook Database - source.*
URL: `https://github.com/lerocha/chinook-database` (visited on 06/06/2023).

[4] Anders SG Andrae and Tomas Edler.
"On global electricity usage of communication technology: trends to 2030".
In: *Challenges* 6.1 (2015), pp. 117–157.

[5] Rui Pereira, Marco Couto, Francisco Ribeiro, Rui Rua, Jácome Cunha, João Paulo Fernandes, and João Saraiva. "Energy Efficiency across Programming Languages: How Do Energy, Time, and Memory Relate?" In: *Proceedings of the 10th ACM SIGPLAN International Conference on Software Language Engineering.* SLE 2017.
Vancouver, BC, Canada: Association for Computing Machinery, 2017, pp. 256–267.
ISBN: 9781450355254. DOI: `10.1145/3136014.3136031`.
URL: `https://doi.org/10.1145/3136014.3136031`.

[6] Luís Gabriel Lima, Francisco Soares-Neto, Paulo Lieuthier, Fernando Castor, Gilberto Melfe, and João Paulo Fernandes.
"Haskell in green land: Analyzing the energy behavior of a purely functional language".
In: *2016 IEEE 23rd international conference on Software Analysis, Evolution, and Reengineering (SANER).* Vol. 1. IEEE. 2016, pp. 517–528.

[7] solidIT. *DB-Engines Ranking.* `https://db-engines.com/en/ranking`.
[Online; Accessed 2023-05-17]. 2023.

[8] Coral Calero, Macario Polo, and Mª Ángeles Moraga. "Investigating the impact on execution time and energy consumption of developing with Spring".
In: *Sustainable Computing: Informatics and Systems* 32 (2021), p. 100603. ISSN: 2210-5379.
DOI: `https://doi.org/10.1016/j.suscom.2021.100603`.
URL: `https://www.sciencedirect.com/science/article/pii/S2210537921000913`.

[9] Giuseppe Procaccianti, Patricia Lago, and Wouter Diesveld.
"Energy efficiency of orm approaches: an empirical evaluation". In: *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement.* 2016, pp. 1–10.

[10] Kashif Nizam Khan, Mikael Hirki, Tapio Niemi, Jukka K. Nurminen, and Zhonghong Ou.
"RAPL in Action: Experiences in Using RAPL for Power Measurements".
In: *ACM Trans. Model. Perform. Eval. Comput. Syst.* 3.2 (Mar. 2018). ISSN: 2376-3639.
DOI: `10.1145/3177754`. URL: `https://doi.org/10.1145/3177754`.

[11] Jack Dongarra, Hatem Ltaief, Piotr Luszczek, and Vincent 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*. IEEE. 2012, pp. 274–281.

[12] Mohammed El Mehdi Diouri, Manuel F Dolz, Olivier Glück, Laurent Lefèvre, Pedro Alonso, Sandra Catalán, Rafael Mayo, and Enrique S Quintana-Ortí. “Assessing power monitoring approaches for energy and power analysis of computers”. In: *Sustainable Computing: Informatics and Systems* 4.2 (2014), pp. 68–82.

[13] Javier Mancebo, Félix García, and Coral Calero. “A Process for Analysing the Energy Efficiency of Software”. In: *Information and Software Technology* 134 (2021), p. 106560. ISSN: 0950-5849. DOI: `https://doi.org/10.1016/j.infsof.2021.106560`. URL: `https://www.sciencedirect.com/science/article/pii/S0950584921000446`.

[14] Javier Mancebo, Coral Calero, Felix Garcia, M^{a} Angeles Moraga, and Ignacio Garcia-Rodriguez de Guzman. “FEETINGS: Framework for Energy Efficiency Testing to Improve Environmental Goal of the Software”. In: *Sustainable Computing: Informatics and Systems* 30 (2021), p. 100558. ISSN: 2210-5379. DOI: `https://doi.org/10.1016/j.suscom.2021.100558`. URL: `https://www.sciencedirect.com/science/article/pii/S2210537921000494`.

[15] Alfred Ilberg Ulvmose, Peter Tamas Sebok, Valerie Luna Dickson, and Virmantas Cekys. “Testing Energy Consumption on Mobile-Like Devices with a Modular Step Counting Algorithm”. In: *Aalborg University Student Report* (Dec. 2022).

[16] *jRAPL*. URL: `https://github.com/aservet1/jRAPL` (visited on 06/04/2023).

[17] *CSharpRAPL*. URL: `https://gitlab.com/ImDreamer/CsharpRAPL` (visited on 06/04/2023).

[18] Coral Calero, Macario Polo, and M^{a} Ángeles Moraga. “Investigating the impact on execution time and energy consumption of developing with Spring”. In: *Sustainable Computing: Informatics and Systems* 32 (2021), p. 100603. ISSN: 2210-5379. DOI: `https://doi.org/10.1016/j.suscom.2021.100603`. URL: `https://www.sciencedirect.com/science/article/pii/S2210537921000913`.

[19] Doina Zmaranda, Lucian-Laurentiu Pop-Fele, Cornelia Gyorödi, Robert Gyorödi, and George Pecherle. “Performance comparison of crud methods using net object relational mappers: A case study”. In: *International Journal of Advanced Computer Science and Applications* 11.1 (2020).

[20] *PostgreSQL: The World’s Most Advanced Open Source Relational Database*. URL: `https://www.postgresql.org/` (visited on 05/31/2023).

[21] *Microsoft.EntityFrameworkCore*. URL: `https://www.nuget.org/packages/Microsoft.EntityFrameworkCore` (visited on 05/31/2023).

[22] *BenchmarkDotNet - How it works*. URL: `https://benchmarkdotnet.org/articles/guides/how-it-works.html` (visited on 06/06/2023).

[23] *BenchmarkDotNet - statistics*. URL: `https://github.com/dotnet/BenchmarkDotNet/blob/master/src/BenchmarkDotNet/Mathematics/Statistics.cs` (visited on 06/08/2023).

[24] Nick Dodd. *AutoBogus*. May 24, 2023. URL: `https://github.com/nickdodd79/AutoBogus`.

[25] Brian Chavez. *Bogus*. May 24, 2023. URL: `https://github.com/bchavez/Bogus`.

[26] *Powercap - source*. URL: `https://github.com/powercap/raplcap` (visited on 06/06/2023).

[27] Connor Imes, Huazhe Zhang, Kevin Zhao, and Henry Hoffmann. "CoPPer: Soft Real-Time Application Performance Using Hardware Power Capping". In: *2019 IEEE International Conference on Autonomic Computing (ICAC)*. 2019, pp. 31–41. DOI: 10.1109/ICAC.2019.00015.

[28] *BenchmarkDotNet - Benchmark Runner source code.* URL: https://github.com/dotnet/BenchmarkDotNet/blob/master/src/BenchmarkDotNet/Running/BenchmarkRunnerClean.cs (visited on 05/24/2023).

[29] *BenchmarkDotNet - Benchmark Engine source code.* URL: https://github.com/dotnet/BenchmarkDotNet/blob/master/src/BenchmarkDotNet/Engines/Engine.cs (visited on 05/24/2023).

[30] *BenchmarkDotNet - Diagnosers.* URL: https://benchmarkdotnet.org/articles/configs/diagnosers.html (visited on 05/23/2023).

[31] *BenchmarkDotNet - EventPipeProfiler.* URL: https://benchmarkdotnet.org/articles/features/event-pipe-profiler.html (visited on 05/17/2023).

[32] *BenchmarkDotNet - IDiagnoser source code.* URL: https://github.com/dotnet/BenchmarkDotNet/blob/master/src/BenchmarkDotNet/Diagnosers/IDiagnoser.cs (visited on 05/30/2023).

Appendix A

DUT Hardware Specifications and Software Versions

A.1 Client Specifications

A.1.1 Hardware

```
laptop1
    description: Notebook
    product: 20JTS1DM00 (LENOVO_MT_20JT_BU_Think_FM_ThinkPad T470s W10DG)
    vendor: LENOVO
    version: ThinkPad T470s W10DG
    serial: PC0VA0Q6
    width: 64 bits
    capabilities: smbios-3.0.0 dmi-3.0.0 smp vsyscall32
    configuration: administrator_password=disabled chassis=notebook family=ThinkPad T470s W10DG powe
  *-core
       description: Motherboard
       product: 20JTS1DM00
       vendor: LENOVO
       physical id: 0
       version: SDK0J40709 WIN
       serial: L3HF85J03GG
       slot: Not Available
     *-memory
          description: System Memory
          physical id: 3
          slot: System board or motherboard
          size: 8GiB
        *-bank:0
             description: SODIMM DDR4 Synchronous Unbuffered (Unregistered) 2133 MHz (0,5 ns)
             product: M471A5244BB0-CRC
             vendor: Samsung
             physical id: 0
             serial: 00000000
             slot: ChannelA-DIMM0
             size: 4GiB
             width: 64 bits
             clock: 2133MHz (0.5ns)
        *-bank:1
             description: SODIMM DDR4 Synchronous Unbuffered (Unregistered) 2133 MHz (0,5 ns)
             product: M471A5244CB0-CRC
             vendor: Samsung
             physical id: 1
             serial: 928395D9
             slot: ChannelB-DIMM0
             size: 4GiB
             width: 64 bits
             clock: 2133MHz (0.5ns)
     *-cache:0
          description: L1 cache
          physical id: 7
          slot: L1 Cache
          size: 128KiB
          capacity: 128KiB
          capabilities: synchronous internal write-back unified
          configuration: level=1
     *-cache:1
          description: L2 cache
          physical id: 8
          slot: L2 Cache
          size: 512KiB
          capacity: 512KiB
          capabilities: synchronous internal write-back unified
          configuration: level=2
     *-cache:2
          description: L3 cache
          physical id: 9
          slot: L3 Cache
          size: 4MiB
          capacity: 4MiB
          capabilities: synchronous internal write-back unified
          configuration: level=3
     *-cpu
          description: CPU
          product: Intel(R) Core(TM) i7-6600U CPU @ 2.60GHz
          vendor: Intel Corp.
          physical id: a
          bus info: cpu@0
          version: Intel(R) Core(TM) i7-6600U CPU @ 2.60GHz
          serial: None
          slot: U3E1
          size: 2714MHz
          capacity: 3400MHz
          width: 64 bits
          clock: 100MHz
          capabilities: lm fpu fpu_exception wp vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca
          configuration: cores=2 enabledcores=2 threads=4
```

```
*-firmware
     description: BIOS
     vendor: LENOVO
     physical id: b
     version: N1WET70W (1.49 )
     date: 12/26/2022
     size: 128KiB
     capacity: 16MiB
     capabilities: pci pnp upgrade shadowing cdboot bootselect edd int13floppy720 int5printscree
*-pci
     description: Host bridge
     product: Xeon E3-1200 v5/E3-1500 v5/6th Gen Core Processor Host Bridge/DRAM Registers
     vendor: Intel Corporation
     physical id: 100
     bus info: pci@0000:00:00.0
     version: 08
     width: 32 bits
     clock: 33MHz
     configuration: driver=skl_uncore
     resources: irq:0
   *-display
        description: VGA compatible controller
        product: Skylake GT2 [HD Graphics 520]
        vendor: Intel Corporation
        physical id: 2
        bus info: pci@0000:00:02.0
        version: 07
        width: 64 bits
        clock: 33MHz
        capabilities: pciexpress msi pm vga_controller bus_master cap_list rom
        configuration: driver=i915 latency=0
        resources: irq:131 memory:eb000000-ebffffff memory:a0000000-afffffff ioport:e000(size=64
   *-usb
        description: USB controller
        product: Sunrise Point-LP USB 3.0 xHCI Controller
        vendor: Intel Corporation
        physical id: 14
        bus info: pci@0000:00:14.0
        version: 21
        width: 64 bits
        clock: 33MHz
        capabilities: pm msi xhci bus_master cap_list
        configuration: driver=xhci_hcd latency=0
        resources: irq:125 memory:ec220000-ec22ffff
      *-usbhost:0
           product: xHCI Host Controller
           vendor: Linux 6.3.5-arch1-1 xhci-hcd
           physical id: 0
           bus info: usb@1
           logical name: usb1
           version: 6.03
```

```
          capabilities: usb-2.00
          configuration: driver=hub slots=12 speed=480Mbit/s
        *-usb:0
             description: Bluetooth wireless interface
             product: Bluetooth wireless interface
             vendor: Intel Corp.
             physical id: 7
             bus info: usb@1:7
             version: 0.01
             capabilities: bluetooth usb-2.00
             configuration: driver=btusb maxpower=100mA speed=12Mbit/s
        *-usb:1
             description: Video
             product: Integrated Camera
             vendor: SunplusIT Inc
             physical id: 8
             bus info: usb@1:8
             version: 0.10
             capabilities: usb-2.00
             configuration: driver=uvcvideo maxpower=500mA speed=480Mbit/s
     *-usbhost:1
          product: xHCI Host Controller
          vendor: Linux 6.3.5-arch1-1 xhci-hcd
          physical id: 1
          bus info: usb@2
          logical name: usb2
          version: 6.03
          capabilities: usb-3.00
          configuration: driver=hub slots=6 speed=5000Mbit/s
        *-usb
             description: Mass storage device
             product: USB3.0-CRW
             vendor: Generic
             physical id: 3
             bus info: usb@2:3
             version: 2.04
             serial: 20120501030900000
             capabilities: usb-3.00 scsi
             configuration: driver=usb-storage maxpower=800mA speed=5000Mbit/s
  *-generic
       description: Signal processing controller
       product: Sunrise Point-LP Thermal subsystem
       vendor: Intel Corporation
       physical id: 14.2
       bus info: pci@0000:00:14.2
       version: 21
       width: 64 bits
       clock: 33MHz
       capabilities: pm msi cap_list
       configuration: driver=intel_pch_thermal latency=0
       resources: irq:18 memory:ec248000-ec248fff
```

```
*-communication:0
     description: Communication controller
     product: Sunrise Point-LP CSME HECI #1
     vendor: Intel Corporation
     physical id: 16
     bus info: pci@0000:00:16.0
     version: 21
     width: 64 bits
     clock: 33MHz
     capabilities: pm msi bus_master cap_list
     configuration: driver=mei_me latency=0
     resources: irq:133 memory:ec249000-ec249fff
*-communication:1
     description: Serial controller
     product: Sunrise Point-LP Active Management Technology - SOL
     vendor: Intel Corporation
     physical id: 16.3
     bus info: pci@0000:00:16.3
     version: 21
     width: 32 bits
     clock: 66MHz
     capabilities: msi pm 16550 cap_list
     configuration: driver=serial latency=0
     resources: irq:19 ioport:e060(size=8) memory:ec24b000-ec24bfff
*-pci:0
     description: PCI bridge
     product: Sunrise Point-LP PCI Express Root Port #1
     vendor: Intel Corporation
     physical id: 1c
     bus info: pci@0000:00:1c.0
     version: f1
     width: 32 bits
     clock: 33MHz
     capabilities: pci pciexpress msi pm normal_decode bus_master cap_list
     configuration: driver=pcieport
     resources: irq:122 ioport:2000(size=4096) memory:d4000000-ea0fffff ioport:b0000000(size=
*-pci:1
     description: PCI bridge
     product: Sunrise Point-LP PCI Express Root Port #3
     vendor: Intel Corporation
     physical id: 1c.2
     bus info: pci@0000:00:1c.2
     version: f1
     width: 32 bits
     clock: 33MHz
     capabilities: pci pciexpress msi pm normal_decode bus_master cap_list
     configuration: driver=pcieport
     resources: irq:123 memory:ec100000-ec1fffff
   *-network
        description: Wireless interface
        product: Wireless 8260
```

```
           vendor: Intel Corporation
           physical id: 0
           bus info: pci@0000:3a:00.0
           logical name: wlp58s0
           version: 3a
           serial: 38:de:ad:92:17:92
           width: 64 bits
           clock: 33MHz
           capabilities: pm msi pciexpress bus_master cap_list ethernet physical wireless
           configuration: broadcast=yes driver=iwlwifi driverversion=6.3.5-arch1-1 firmware=36.c
           resources: irq:134 memory:ec100000-ec101fff
     *-pci:2
          description: PCI bridge
          product: Sunrise Point-LP PCI Express Root Port #9
          vendor: Intel Corporation
          physical id: 1d
          bus info: pci@0000:00:1d.0
          version: f1
          width: 32 bits
          clock: 33MHz
          capabilities: pci pciexpress msi pm normal_decode bus_master cap_list
          configuration: driver=pcieport
          resources: irq:124 memory:ec000000-ec0fffff
        *-nvme
           description: NVMe device
           product: INTEL SSDPEKKF256G7L
           vendor: Intel Corporation
           physical id: 0
           bus info: pci@0000:3c:00.0
           logical name: /dev/nvme0
           version: 131P
           serial: BTPY815406U4256D
           width: 64 bits
           clock: 33MHz
           capabilities: nvme pm pciexpress msix nvm_express bus_master cap_list
           configuration: driver=nvme latency=0 nqn=nqn.2014.08.org.nvmexpress:80868086BTPY8154(
           resources: irq:16 memory:ec000000-ec003fff
         *-namespace:0
              description: NVMe disk
              physical id: 0
              logical name: hwmon4
         *-namespace:1
              description: NVMe disk
              physical id: 2
              logical name: /dev/ng0n1
         *-namespace:2
              description: NVMe disk
              physical id: 1
              bus info: nvme@0:1
              logical name: /dev/nvme0n1
              size: 238GiB (256GB)
          capabilities: gpt-1.00 partitioned partitioned:gpt
          configuration: guid=e6b06645-d6ce-9646-b771-2242aeac170e logicalsectorsize=512 sec
        *-volume:0
             description: Windows FAT volume
             vendor: mkfs.fat
             physical id: 1
             bus info: nvme@0:1,1
             logical name: /dev/nvme0n1p1
             logical name: /boot
             version: FAT32
             serial: 24fa-3062
             size: 1022MiB
             capacity: 1023MiB
             capabilities: boot fat initialized
             configuration: FATs=2 filesystem=fat mount.fstype=vfat mount.options=rw,relatim
        *-volume:1
             description: Linux swap volume
             vendor: Linux
             physical id: 2
             bus info: nvme@0:1,2
             logical name: /dev/nvme0n1p2
             version: 1
             serial: 80fd307d-799b-4b4e-a217-518c616881f3
             size: 8190MiB
             capacity: 8191MiB
             capabilities: nofs swap initialized
             configuration: filesystem=swap pagesize=4095
        *-volume:2
             description: EXT4 volume
             vendor: Linux
             physical id: 3
             bus info: nvme@0:1,3
             logical name: /dev/nvme0n1p3
             logical name: /
             version: 1.0
             serial: da431963-203d-4628-81bc-5c8994427ee4
             size: 229GiB
             capabilities: journaled extended_attributes large_files huge_files dir_nlink re
             configuration: created=2023-03-22 15:35:11 filesystem=ext4 label=arch_os lastmo
*-isa
     description: ISA bridge
     product: Sunrise Point-LP LPC Controller
     vendor: Intel Corporation
     physical id: 1f
     bus info: pci@0000:00:1f.0
     version: 21
     width: 32 bits
     clock: 33MHz
     capabilities: isa bus_master
     configuration: latency=0
*-memory UNCLAIMED
```

```
           description: Memory controller
           product: Sunrise Point-LP PMC
           vendor: Intel Corporation
           physical id: 1f.2
           bus info: pci@0000:00:1f.2
           version: 21
           width: 32 bits
           clock: 33MHz (30.3ns)
           configuration: latency=0
           resources: memory:ec244000-ec247fff
      *-multimedia
           description: Audio device
           product: Sunrise Point-LP HD Audio
           vendor: Intel Corporation
           physical id: 1f.3
           bus info: pci@0000:00:1f.3
           version: 21
           width: 64 bits
           clock: 33MHz
           capabilities: pm msi bus_master cap_list
           configuration: driver=snd_hda_intel latency=64
           resources: irq:135 memory:ec240000-ec243fff memory:ec230000-ec23ffff
      *-serial
           description: SMBus
           product: Sunrise Point-LP SMBus
           vendor: Intel Corporation
           physical id: 1f.4
           bus info: pci@0000:00:1f.4
           version: 21
           width: 64 bits
           clock: 33MHz
           configuration: driver=i801_smbus latency=0
           resources: irq:16 memory:ec24a000-ec24a0ff ioport:efa0(size=32)
      *-network
           description: Ethernet interface
           product: Ethernet Connection I219-LM
           vendor: Intel Corporation
           physical id: 1f.6
           bus info: pci@0000:00:1f.6
           logical name: enp0s31f6
           version: 21
           serial: 8c:16:45:86:cd:39
           size: 1Gbit/s
           capacity: 1Gbit/s
           width: 32 bits
           clock: 33MHz
           capabilities: pm msi bus_master cap_list ethernet physical tp 10bt 10bt-fd 100bt 100bt-
           configuration: autonegotiation=on broadcast=yes driver=e1000e driverversion=6.3.5-arch1-
           resources: irq:132 memory:ec200000-ec21ffff
   *-pnp00:00
        product: Motherboard registers
```

```
          physical id: 0
          capabilities: pnp
          configuration: driver=system
     *-pnp00:01
          product: Motherboard registers
          physical id: 1
          capabilities: pnp
          configuration: driver=system
     *-pnp00:02
          product: Motherboard registers
          physical id: 2
          capabilities: pnp
          configuration: driver=system
     *-pnp00:03
          product: Motherboard registers
          physical id: 4
          capabilities: pnp
          configuration: driver=system
     *-pnp00:04
          product: AT Real-Time Clock
          physical id: 5
          capabilities: pnp
          configuration: driver=rtc_cmos
     *-pnp00:05
          product: PnP device INT3f0d
          vendor: Interphase Corporation
          physical id: 6
          capabilities: pnp
          configuration: driver=system
     *-pnp00:06
          product: PnP device LEN0071
          vendor: Lenovo Group Limited
          physical id: c
          capabilities: pnp
          configuration: driver=i8042 kbd
     *-pnp00:07
          product: PnP device LEN007f
          vendor: Lenovo Group Limited
          physical id: d
          capabilities: pnp
          configuration: driver=i8042 aux
     *-pnp00:08
          product: Motherboard registers
          physical id: e
          capabilities: pnp
          configuration: driver=system
     *-pnp00:09
          product: Motherboard registers
          physical id: f
          capabilities: pnp
          configuration: driver=system
     *-pnp00:0a
          product: Motherboard registers
          physical id: 10
          capabilities: pnp
          configuration: driver=system
     *-pnp00:0b
          product: System Board
          physical id: 11
          capabilities: pnp
          configuration: driver=system
  *-battery:0
       product: 00HW023
       vendor: SMP
       physical id: 1
       slot: Front
       capacity: 23540mWh
       configuration: voltage=11,4V
  *-battery:1
       product: 01AV406
       vendor: SMP
       physical id: 2
       slot: Rear
       capacity: 26060mWh
       configuration: voltage=11,5V
  *-scsi
       physical id: 3
       bus info: scsi@0
       logical name: scsi0
       capabilities: scsi-host
       configuration: driver=usb-storage
```

A.1.2 Software

```
acl 2.3.1-3
archlinux-keyring 20230504-1
argon2 20190702-5
attr 2.5.1-3
audit 3.1.1-1
autoconf 2.71-4
automake 1.16.5-2
base 3-1
base-devel 1-1
bash 5.1.016-4
binutils 2.40-6
bison 3.8.2-5
blas 3.11.0-2
blosc 1.21.3-1
brotli 1.0.9-12
bzip2 1.0.8-5
ca-certificates 20220905-1
ca-certificates-mozilla 3.89.1-1
ca-certificates-utils 20220905-1
```

```
cairo 1.17.8-2
cmake 3.26.4-1
cmark-gfm 0.29.0.gfm.11-1
cmatrix 2.0-2
coreutils 9.3-1
crypto++ 8.7.0-1
cryptsetup 2.6.1-3
curl 8.1.2-1
db5.3 5.3.28-2
dbus 1.14.6-2
debugedit 5.0-5
device-mapper 2.03.21-1
dhcpcd 10.0.1-1
diffutils 3.10-1
dnssec-anchors 20190629-3
dotnet-host 7.0.5.sdk105-1
dotnet-runtime 7.0.5.sdk105-1
dotnet-sdk 7.0.5.sdk105-1
dotnet-targeting-pack 7.0.5.sdk105-1
e2fsprogs 1.47.0-1
expat 2.5.0-1
fakeroot 1.31-2
file 5.44-3
filesystem 2023.01.31-1
findutils 4.9.0-3
flex 2.6.4-5
fontconfig 2:2.14.2-1
freetype2 2.13.0-1
fribidi 1.0.13-1
gawk 5.2.2-1
gc 8.2.4-1
gcc 13.1.1-1
gcc-fortran 13.1.1-1
gcc-libs 13.1.1-1
gdal 3.7.0-2
gdbm 1.23-2
geos 3.11.2-1
gettext 0.21.1-5
giflib 5.2.1-2
git 2.40.1-1
glib2 2.76.3-1
glibc 2.37-3
gmp 6.2.1-2
gnu-free-fonts 20120503-8
gnupg 2.2.41-1
gnutls 3.8.0-1
go 2:1.20.4-2
gpgme 1.20.0-3
graphite 1:1.3.14-3
grep 3.11-1
groff 1.22.4-10
```

```
guile 3.0.9-1
gzip 1.12-2
harfbuzz 7.3.0-1
hicolor-icon-theme 0.17-3
htop 3.2.2-1
hwdata 0.370-1
iana-etc 20230405-1
icu 72.1-2
intel-ucode 20230516.a-1
iproute2 6.3.0-2
iptables 1:1.8.9-1
iputils 20221126-2
jansson 2.14-2
json-c 0.16-1
jsoncpp 1.9.5-2
kbd 2.5.1-2
keyutils 1.6.3-2
kmod 30-3
krb5 1.20.1-1
lapack 3.11.0-2
ldns 1.8.3-2
less 1:633-1
libarchive 3.6.2-2
libassuan 2.5.5-2
libbpf 1.2.0-1
libcap 2.69-1
libcap-ng 0.8.3-2
libdatrie 0.2.13-2
libdeflate 1.18-1
libedit 20221030_3.1-1
libelf 0.189-1
libevent 2.1.12-4
libffi 3.4.4-1
libfreexl 1.0.6-2
libgcrypt 1.10.2-1
libgeotiff 1.7.1-2
libgpg-error 1.47-1
libgudev 237-2
libice 1.1.1-2
libidn2 2.3.4-3
libimobiledevice 1.3.0-9
libisl 0.26-1
libjpeg-turbo 2.1.5.1-1
libksba 1.6.3-1
libldap 2.6.4-2
libmnl 1.0.5-1
libmpc 1.3.1-1
libnetfilter_conntrack 1.0.9-1
libnfnetlink 1.0.2-1
libnftnl 1.2.5-1
libnghttp2 1.53.0-1
```

```
libnl 3.7.0-3
libnsl 2.0.0-3
libp11-kit 0.24.1-1
libpcap 1.10.4-1
libplist 2.3.0-2
libpng 1.6.39-1
libpsl 0.21.2-1
librttopo 1.1.0-5
libsasl 2.1.28-4
libseccomp 2.5.4-2
libsecret 0.20.5-2
libsm 1.2.4-1
libspatialite 5.0.1-3
libssh2 1.10.0-3
libsysprof-capture 3.48.0-2
libtasn1 4.19.0-1
libthai 0.1.29-2
libtiff 4.5.0-4
libtirpc 1.3.3-2
libtool 2.4.7+4+g1ec8fa28-3
libunistring 1.1-2
libunwind 1.6.2-2
libusb 1.0.26-2
libusbmuxd 2.0.2-3
libuv 1.44.2-1
libverto 0.3.2-4
libx11 1.8.4-1
libxau 1.0.11-2
libxcb 1.15-2
libxcrypt 4.4.33-1
libxdmcp 1.1.4-2
libxext 1.3.5-1
libxft 2.3.8-1
libxml2 2.10.4-4
libxmu 1.1.4-1
libxrender 0.9.11-1
libxslt 1.1.37-3
libxt 1.3.0-1
licenses 20220125-2
linux 6.3.5.arch1-1
linux-api-headers 6.3-1
linux-firmware 20230404.2e92a49f-1
linux-firmware-whence 20230404.2e92a49f-1
llvm-libs 15.0.7-3
lshw B.02.19.2-5
lz4 1:1.9.4-1
lzo 2.10-5
m4 1.4.19-3
make 4.4.1-2
minizip 1:1.2.13-2
mkinitcpio 36-1
```

```
mkinitcpio-busybox 1.35.0-1
mpfr 4.2.0.p9-1
nano 7.2-1
ncurses 6.4-1
neofetch 7.1.0-2
netstandard-targeting-pack 7.0.5.sdk105-1
nettle 3.9-1
npth 1.6-4
ntp 4.2.8.p15-3
ocl-icd 2.3.1-1
openssh 9.3p1-1
openssl 3.0.9-1
p11-kit 0.24.1-1
pacman 6.0.2-7
pacman-mirrorlist 20230410-1
pam 1.5.3-3
pambase 20221020-1
pango 1:1.50.14-1
patch 2.7.6-10
pciutils 3.10.0-1
pcre2 10.42-2
perl 5.36.1-1
perl-error 0.17029-4
perl-mailtools 2.21-6
perl-timedate 2.33-4
pinentry 1.2.1-1
pixman 0.42.2-1
pkgconf 1.8.1-1
popt 1.19-1
postgresql 15.3-1
postgresql-libs 15.3-1
procps-ng 3.3.17-1
proj 9.2.0-1
psmisc 23.6-1
qhull 2020.2-4
r 4.3.0-1
readline 8.2.001-2
rhash 1.4.3-1
sed 4.9-3
shadow 4.13-2
snappy 1.1.9-2
sqlite 3.42.0-1
sshpass 1.10-1
sudo 1.9.13.p3-1
systemd 253.4-1
systemd-libs 253.4-1
systemd-sysvcompat 253.4-1
tar 1.34-2
texinfo 7.0.3-1
tpm2-tss 4.0.1-1
tree 2.1.0-1
```

```
ttf-liberation 2.1.5-1
tzdata 2023c-2
unixodbc 2.3.11-1
unzip 6.0-19
upower 1.90.0-4
usbmuxd 1.1.1-3
util-linux 2.39-4
util-linux-libs 2.39-4
which 2.21-6
xcb-proto 1.15.2-3
xerces-c 3.2.4-2
xorgproto 2022.2-1
xz 5.4.3-1
zip 3.0-10
zlib 1:1.2.13-2
zstd 1.5.5-1
```

A.2 Server Specification

A.2.1 Hardware

```
laptop2
    description: Notebook
    product: 20JTS1DM00 (LENOVO_MT_20JT_BU_Think_FM_ThinkPad T470s W10DG)
    vendor: LENOVO
    version: ThinkPad T470s W10DG
    serial: PC0VGBDR
    width: 64 bits
    capabilities: smbios-3.0.0 dmi-3.0.0 smp vsyscall32
    configuration: administrator_password=disabled chassis=notebook family=ThinkPad T470s W10DG power
  *-core
       description: Motherboard
       product: 20JTS1DM00
       vendor: LENOVO
       physical id: 0
       version: SDK0J40709 WIN
       serial: L3HF85J05KJ
       slot: Not Available
     *-memory
          description: System Memory
          physical id: 3
          slot: System board or motherboard
          size: 20GiB
        *-bank:0
             description: SODIMM DDR4 Synchronous Unbuffered (Unregistered) 2133 MHz (0,5 ns)
             product: M471A5244BB0-CRC
             vendor: Samsung
             physical id: 0
             serial: 00000000
             slot: ChannelA-DIMM0
             size: 4GiB
             width: 64 bits
             clock: 2133MHz (0.5ns)
        *-bank:1
             description: SODIMM DDR4 Synchronous Unbuffered (Unregistered) 2133 MHz (0,5 ns)
             product: 9905663-031.A00G
             vendor: Kingston
             physical id: 1
             serial: B1A28B4B
             slot: ChannelB-DIMM0
             size: 16GiB
             width: 64 bits
             clock: 2133MHz (0.5ns)
     *-cache:0
          description: L1 cache
          physical id: 7
          slot: L1 Cache
          size: 128KiB
          capacity: 128KiB
          capabilities: synchronous internal write-back unified
          configuration: level=1
     *-cache:1
          description: L2 cache
          physical id: 8
          slot: L2 Cache
          size: 512KiB
          capacity: 512KiB
          capabilities: synchronous internal write-back unified
          configuration: level=2
     *-cache:2
          description: L3 cache
          physical id: 9
          slot: L3 Cache
          size: 4MiB
          capacity: 4MiB
          capabilities: synchronous internal write-back unified
          configuration: level=3
     *-cpu
          description: CPU
          product: Intel(R) Core(TM) i7-6600U CPU @ 2.60GHz
          vendor: Intel Corp.
          physical id: a
          bus info: cpu@0
          version: Intel(R) Core(TM) i7-6600U CPU @ 2.60GHz
          serial: None
          slot: U3E1
          size: 3381MHz
          capacity: 3400MHz
          width: 64 bits
          clock: 100MHz
          capabilities: lm fpu fpu_exception wp vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca
          configuration: cores=2 enabledcores=2 threads=4
```

```
*-firmware
     description: BIOS
     vendor: LENOVO
     physical id: b
     version: N1WET70W (1.49 )
     date: 12/26/2022
     size: 128KiB
     capacity: 16MiB
     capabilities: pci pnp upgrade shadowing cdboot bootselect edd int13floppy720 int5printscree
*-pci
     description: Host bridge
     product: Xeon E3-1200 v5/E3-1500 v5/6th Gen Core Processor Host Bridge/DRAM Registers
     vendor: Intel Corporation
     physical id: 100
     bus info: pci@0000:00:00.0
     version: 08
     width: 32 bits
     clock: 33MHz
     configuration: driver=skl_uncore
     resources: irq:0
   *-display
        description: VGA compatible controller
        product: Skylake GT2 [HD Graphics 520]
        vendor: Intel Corporation
        physical id: 2
        bus info: pci@0000:00:02.0
        version: 07
        width: 64 bits
        clock: 33MHz
        capabilities: pciexpress msi pm vga_controller bus_master cap_list rom
        configuration: driver=i915 latency=0
        resources: irq:131 memory:eb000000-ebffffff memory:a0000000-afffffff ioport:e000(size=64
   *-usb
        description: USB controller
        product: Sunrise Point-LP USB 3.0 xHCI Controller
        vendor: Intel Corporation
        physical id: 14
        bus info: pci@0000:00:14.0
        version: 21
        width: 64 bits
        clock: 33MHz
        capabilities: pm msi xhci bus_master cap_list
        configuration: driver=xhci_hcd latency=0
        resources: irq:125 memory:ec220000-ec22ffff
      *-usbhost:0
           product: xHCI Host Controller
           vendor: Linux 6.3.5-arch1-1 xhci-hcd
           physical id: 0
           bus info: usb@1
           logical name: usb1
           version: 6.03
```

```
          capabilities: usb-2.00
          configuration: driver=hub slots=12 speed=480Mbit/s
        *-usb:0
             description: Bluetooth wireless interface
             product: Bluetooth wireless interface
             vendor: Intel Corp.
             physical id: 7
             bus info: usb@1:7
             version: 0.01
             capabilities: bluetooth usb-2.00
             configuration: driver=btusb maxpower=100mA speed=12Mbit/s
        *-usb:1
             description: Video
             product: Integrated Camera
             vendor: SunplusIT Inc
             physical id: 8
             bus info: usb@1:8
             version: 37.37
             capabilities: usb-2.00
             configuration: driver=uvcvideo maxpower=500mA speed=480Mbit/s
     *-usbhost:1
          product: xHCI Host Controller
          vendor: Linux 6.3.5-arch1-1 xhci-hcd
          physical id: 1
          bus info: usb@2
          logical name: usb2
          version: 6.03
          capabilities: usb-3.00
          configuration: driver=hub slots=6 speed=5000Mbit/s
        *-usb
             description: Mass storage device
             product: USB3.0-CRW
             vendor: Generic
             physical id: 3
             bus info: usb@2:3
             version: 2.04
             serial: 20120501030900000
             capabilities: usb-3.00 scsi
             configuration: driver=usb-storage maxpower=800mA speed=5000Mbit/s
  *-generic
       description: Signal processing controller
       product: Sunrise Point-LP Thermal subsystem
       vendor: Intel Corporation
       physical id: 14.2
       bus info: pci@0000:00:14.2
       version: 21
       width: 64 bits
       clock: 33MHz
       capabilities: pm msi cap_list
       configuration: driver=intel_pch_thermal latency=0
       resources: irq:18 memory:ec248000-ec248fff
```

```
*-communication:0
     description: Communication controller
     product: Sunrise Point-LP CSME HECI #1
     vendor: Intel Corporation
     physical id: 16
     bus info: pci@0000:00:16.0
     version: 21
     width: 64 bits
     clock: 33MHz
     capabilities: pm msi bus_master cap_list
     configuration: driver=mei_me latency=0
     resources: irq:133 memory:ec249000-ec249fff
*-communication:1
     description: Serial controller
     product: Sunrise Point-LP Active Management Technology - SOL
     vendor: Intel Corporation
     physical id: 16.3
     bus info: pci@0000:00:16.3
     version: 21
     width: 32 bits
     clock: 66MHz
     capabilities: msi pm 16550 cap_list
     configuration: driver=serial latency=0
     resources: irq:19 ioport:e060(size=8) memory:ec24b000-ec24bfff
*-pci:0
     description: PCI bridge
     product: Sunrise Point-LP PCI Express Root Port #1
     vendor: Intel Corporation
     physical id: 1c
     bus info: pci@0000:00:1c.0
     version: f1
     width: 32 bits
     clock: 33MHz
     capabilities: pci pciexpress msi pm normal_decode bus_master cap_list
     configuration: driver=pcieport
     resources: irq:122 ioport:2000(size=4096) memory:d4000000-ea0fffff ioport:b0000000(size=
*-pci:1
     description: PCI bridge
     product: Sunrise Point-LP PCI Express Root Port #3
     vendor: Intel Corporation
     physical id: 1c.2
     bus info: pci@0000:00:1c.2
     version: f1
     width: 32 bits
     clock: 33MHz
     capabilities: pci pciexpress msi pm normal_decode bus_master cap_list
     configuration: driver=pcieport
     resources: irq:123 memory:ec100000-ec1fffff
   *-network
        description: Wireless interface
        product: Wireless 8260
```

```
          vendor: Intel Corporation
          physical id: 0
          bus info: pci@0000:3a:00.0
          logical name: wlp58s0
          version: 3a
          serial: 74:70:fd:57:1d:04
          width: 64 bits
          clock: 33MHz
          capabilities: pm msi pciexpress bus_master cap_list ethernet physical wireless
          configuration: broadcast=yes driver=iwlwifi driverversion=6.3.5-arch1-1 firmware=36.c
          resources: irq:134 memory:ec100000-ec101fff
     *-pci:2
          description: PCI bridge
          product: Sunrise Point-LP PCI Express Root Port #9
          vendor: Intel Corporation
          physical id: 1d
          bus info: pci@0000:00:1d.0
          version: f1
          width: 32 bits
          clock: 33MHz
          capabilities: pci pciexpress msi pm normal_decode bus_master cap_list
          configuration: driver=pcieport
          resources: irq:124 memory:ec000000-ec0fffff
        *-nvme
             description: NVMe device
             product: INTEL SSDPEKKF256G7L
             vendor: Intel Corporation
             physical id: 0
             bus info: pci@0000:3c:00.0
             logical name: /dev/nvme0
             version: 131P
             serial: BTPY81540DJL256D
             width: 64 bits
             clock: 33MHz
             capabilities: nvme pm pciexpress msix nvm_express bus_master cap_list
             configuration: driver=nvme latency=0 nqn=nqn.2014.08.org.nvmexpress:80868086BTPY8154(
             resources: irq:16 memory:ec000000-ec003fff
           *-namespace:0
                description: NVMe disk
                physical id: 0
                logical name: hwmon4
           *-namespace:1
                description: NVMe disk
                physical id: 2
                logical name: /dev/ng0n1
           *-namespace:2
                description: NVMe disk
                physical id: 1
                bus info: nvme@0:1
                logical name: /dev/nvme0n1
                size: 238GiB (256GB)
           capabilities: gpt-1.00 partitioned partitioned:gpt
           configuration: guid=125cdf1c-0fc6-5c42-94e2-2f12a0a12c11 logicalsectorsize=512 sec
         *-volume:0
              description: Windows FAT volume
              vendor: mkfs.fat
              physical id: 1
              bus info: nvme@0:1,1
              logical name: /dev/nvme0n1p1
              logical name: /boot
              version: FAT32
              serial: 0ec3-df9d
              size: 1022MiB
              capacity: 1023MiB
              capabilities: boot fat initialized
              configuration: FATs=2 filesystem=fat mount.fstype=vfat mount.options=rw,relatim
         *-volume:1
              description: Linux swap volume
              vendor: Linux
              physical id: 2
              bus info: nvme@0:1,2
              logical name: /dev/nvme0n1p2
              version: 1
              serial: 21787f87-6dd9-46f5-b574-8f843d53ccd2
              size: 19GiB
              capacity: 19GiB
              capabilities: nofs swap initialized
              configuration: filesystem=swap pagesize=4095
         *-volume:2
              description: EXT4 volume
              vendor: Linux
              physical id: 3
              bus info: nvme@0:1,3
              logical name: /dev/nvme0n1p3
              logical name: /
              version: 1.0
              serial: d63a8eca-4774-4d21-b20c-78fcfd9a90fc
              size: 217GiB
              capabilities: journaled extended_attributes large_files huge_files dir_nlink re
              configuration: created=2023-03-22 16:37:25 filesystem=ext4 label=arch_os lastmo
*-isa
     description: ISA bridge
     product: Sunrise Point-LP LPC Controller
     vendor: Intel Corporation
     physical id: 1f
     bus info: pci@0000:00:1f.0
     version: 21
     width: 32 bits
     clock: 33MHz
     capabilities: isa bus_master
     configuration: latency=0
*-memory UNCLAIMED
```

```
          description: Memory controller
          product: Sunrise Point-LP PMC
          vendor: Intel Corporation
          physical id: 1f.2
          bus info: pci@0000:00:1f.2
          version: 21
          width: 32 bits
          clock: 33MHz (30.3ns)
          configuration: latency=0
          resources: memory:ec244000-ec247fff
     *-multimedia
          description: Audio device
          product: Sunrise Point-LP HD Audio
          vendor: Intel Corporation
          physical id: 1f.3
          bus info: pci@0000:00:1f.3
          version: 21
          width: 64 bits
          clock: 33MHz
          capabilities: pm msi bus_master cap_list
          configuration: driver=snd_hda_intel latency=64
          resources: irq:135 memory:ec240000-ec243fff memory:ec230000-ec23ffff
     *-serial
          description: SMBus
          product: Sunrise Point-LP SMBus
          vendor: Intel Corporation
          physical id: 1f.4
          bus info: pci@0000:00:1f.4
          version: 21
          width: 64 bits
          clock: 33MHz
          configuration: driver=i801_smbus latency=0
          resources: irq:16 memory:ec24a000-ec24a0ff ioport:efa0(size=32)
     *-network
          description: Ethernet interface
          product: Ethernet Connection I219-LM
          vendor: Intel Corporation
          physical id: 1f.6
          bus info: pci@0000:00:1f.6
          logical name: enp0s31f6
          version: 21
          serial: 8c:16:45:86:d2:d1
          size: 1Gbit/s
          capacity: 1Gbit/s
          width: 32 bits
          clock: 33MHz
          capabilities: pm msi bus_master cap_list ethernet physical tp 10bt 10bt-fd 100bt 100bt-
          configuration: autonegotiation=on broadcast=yes driver=e1000e driverversion=6.3.5-arch1
          resources: irq:132 memory:ec200000-ec21ffff
  *-pnp00:00
       product: Motherboard registers
```

```
          physical id: 0
          capabilities: pnp
          configuration: driver=system
     *-pnp00:01
          product: Motherboard registers
          physical id: 1
          capabilities: pnp
          configuration: driver=system
     *-pnp00:02
          product: Motherboard registers
          physical id: 2
          capabilities: pnp
          configuration: driver=system
     *-pnp00:03
          product: Motherboard registers
          physical id: 4
          capabilities: pnp
          configuration: driver=system
     *-pnp00:04
          product: AT Real-Time Clock
          physical id: 5
          capabilities: pnp
          configuration: driver=rtc_cmos
     *-pnp00:05
          product: PnP device INT3f0d
          vendor: Interphase Corporation
          physical id: 6
          capabilities: pnp
          configuration: driver=system
     *-pnp00:06
          product: PnP device LEN0071
          vendor: Lenovo Group Limited
          physical id: c
          capabilities: pnp
          configuration: driver=i8042 kbd
     *-pnp00:07
          product: PnP device LEN007f
          vendor: Lenovo Group Limited
          physical id: d
          capabilities: pnp
          configuration: driver=i8042 aux
     *-pnp00:08
          product: Motherboard registers
          physical id: e
          capabilities: pnp
          configuration: driver=system
     *-pnp00:09
          product: Motherboard registers
          physical id: f
          capabilities: pnp
          configuration: driver=system
     *-pnp00:0a
          product: Motherboard registers
          physical id: 10
          capabilities: pnp
          configuration: driver=system
     *-pnp00:0b
          product: System Board
          physical id: 11
          capabilities: pnp
          configuration: driver=system
  *-battery:0
       product: 00HW023
       vendor: SMP
       physical id: 1
       slot: Front
       capacity: 23540mWh
       configuration: voltage=11,4V
  *-battery:1
       product: 01AV406
       vendor: SMP
       physical id: 2
       slot: Rear
       capacity: 26060mWh
       configuration: voltage=11,5V
  *-scsi
       physical id: 3
       bus info: scsi@0
       logical name: scsi0
       capabilities: scsi-host
       configuration: driver=usb-storage
```

A.2.2 Software

```
acl 2.3.1-3
archlinux-keyring 20230504-1
argon2 20190702-5
attr 2.5.1-3
audit 3.1.1-1
autoconf 2.71-4
automake 1.16.5-2
base 3-1
base-devel 1-1
bash 5.1.016-4
binutils 2.40-6
bison 3.8.2-5
brotli 1.0.9-12
bzip2 1.0.8-5
ca-certificates 20220905-1
ca-certificates-mozilla 3.89.1-1
ca-certificates-utils 20220905-1
cmake 3.26.4-1
coreutils 9.3-1
```

```
cryptsetup 2.6.1-3
curl 8.1.2-1
db5.3 5.3.28-2
dbus 1.14.6-2
debugedit 5.0-5
device-mapper 2.03.21-1
dhcpcd 10.0.1-1
diffutils 3.10-1
dnssec-anchors 20190629-3
dotnet-host 7.0.5.sdk105-1
dotnet-runtime 7.0.5.sdk105-1
dotnet-sdk 7.0.5.sdk105-1
dotnet-targeting-pack 7.0.5.sdk105-1
e2fsprogs 1.47.0-1
expat 2.5.0-1
fakeroot 1.31-2
file 5.44-3
filesystem 2023.01.31-1
findutils 4.9.0-3
flex 2.6.4-5
gawk 5.2.2-1
gc 8.2.4-1
gcc 13.1.1-1
gcc-libs 13.1.1-1
gdbm 1.23-2
gettext 0.21.1-5
git 2.40.1-1
glib2 2.76.3-1
glibc 2.37-3
gmp 6.2.1-2
gnupg 2.2.41-1
gnutls 3.8.0-1
go 2:1.20.4-2
gpgme 1.20.0-3
grep 3.11-1
groff 1.22.4-10
guile 3.0.9-1
gzip 1.12-2
hicolor-icon-theme 0.17-3
htop 3.2.2-1
hwdata 0.370-1
iana-etc 20230405-1
icu 72.1-2
intel-ucode 20230516.a-1
iproute2 6.3.0-2
iptables 1:1.8.9-1
iputils 20221126-2
jansson 2.14-2
json-c 0.16-1
jsoncpp 1.9.5-2
kbd 2.5.1-2
```

```
keyutils 1.6.3-2
kmod 30-3
krb5 1.20.1-1
ldns 1.8.3-2
less 1:633-1
libarchive 3.6.2-2
libassuan 2.5.5-2
libbpf 1.2.0-1
libcap 2.69-1
libcap-ng 0.8.3-2
libedit 20221030_3.1-1
libelf 0.189-1
libevent 2.1.12-4
libffi 3.4.4-1
libgcrypt 1.10.2-1
libgpg-error 1.47-1
libidn2 2.3.4-3
libisl 0.26-1
libksba 1.6.3-1
libldap 2.6.4-2
libmnl 1.0.5-1
libmpc 1.3.1-1
libnetfilter_conntrack 1.0.9-1
libnfnetlink 1.0.2-1
libnftnl 1.2.5-1
libnghttp2 1.53.0-1
libnl 3.7.0-3
libnsl 2.0.0-3
libp11-kit 0.24.1-1
libpcap 1.10.4-1
libpsl 0.21.2-1
libsasl 2.1.28-4
libseccomp 2.5.4-2
libsecret 0.20.5-2
libssh2 1.10.0-3
libsysprof-capture 3.48.0-2
libtasn1 4.19.0-1
libtirpc 1.3.3-2
libtool 2.4.7+4+g1ec8fa28-3
libunistring 1.1-2
libunwind 1.6.2-2
libuv 1.44.2-1
libverto 0.3.2-4
libxcrypt 4.4.33-1
libxml2 2.10.4-4
libxslt 1.1.37-3
licenses 20220125-2
linux 6.3.5.arch1-1
linux-api-headers 6.3-1
linux-firmware 20230404.2e92a49f-1
linux-firmware-whence 20230404.2e92a49f-1
```

```
llvm-libs 15.0.7-3
lshw B.02.19.2-5
lz4 1:1.9.4-1
m4 1.4.19-3
make 4.4.1-2
mkinitcpio 36-1
mkinitcpio-busybox 1.35.0-1
mpfr 4.2.0.p9-1
nano 7.2-1
ncurses 6.4-1
neofetch 7.1.0-2
netstandard-targeting-pack 7.0.5.sdk105-1
nettle 3.9-1
npth 1.6-4
ntp 4.2.8.p15-3
openssh 9.3p1-1
openssl 3.0.9-1
p11-kit 0.24.1-1
pacman 6.0.2-7
pacman-mirrorlist 20230410-1
pam 1.5.3-3
pambase 20221020-1
patch 2.7.6-10
pciutils 3.10.0-1
pcre2 10.42-2
perl 5.36.1-1
perl-error 0.17029-4
perl-mailtools 2.21-6
perl-timedate 2.33-4
pinentry 1.2.1-1
pkgconf 1.8.1-1
popt 1.19-1
postgresql 15.3-1
postgresql-libs 15.3-1
procps-ng 3.3.17-1
psmisc 23.6-1
python 3.11.3-1
readline 8.2.001-2
rhash 1.4.3-1
sed 4.9-3
shadow 4.13-2
sqlite 3.42.0-1
sudo 1.9.13.p3-1
systemd 253.4-1
systemd-libs 253.4-1
systemd-sysvcompat 253.4-1
tar 1.34-2
texinfo 7.0.3-1
tpm2-tss 4.0.1-1
tzdata 2023c-2
ufw 0.36.2-1
```

```
util-linux 2.39-4
util-linux-libs 2.39-4
which 2.21-6
xz 5.4.3-1
zlib 1:1.2.13-2
zstd 1.5.5-1
```