Exploring the CLAIX Monitoring Data for Statistical Modeling

Seminararbeit von Niko Sakic
(Matrikelnummer 3518708)

Studiengang Bachelor of Scientific Programming
im Fachbereich 9 “Medizintechnik und Technomathematik”
an der Fachhochschule Aachen, Campus Jülich

Diese Arbeit wurde betreut von:
1. Prüfer: Prof. Dr. rer. nat. Hans Joachim Pflug
2. Prüfer: Christian Wassermann - M. Sc.

Diese Arbeit wurde am IT Center der RWTH Aachen durchgeführt.

Aachen, den 19. Januar 2024

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die Seminararbeit mit dem Thema

Exploring the CLAIX Monitoring Data for Statistical Modeling

selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, alle Ausführungen, die anderen Schriften wörtlich oder sinngemäß entnommen wurden, kenntlich gemacht sind und die Arbeit in gleicher oder ähnlicher Fassung noch nicht Bestandteil einer Studien- oder Prüfungsleistung war. Ich verpflichte mich, ein Exemplar der Seminararbeit fünf Jahre aufzubewahren und auf Verlangen dem Prüfungsamt des Fachbereiches Medizintechnik und Technomathematik auszuhändigen.

Aachen, den 19. Januar 2024

Niko Sakic

Abstract

Efficient utilization of HPC clusters is crucial due to their high upfront costs and ongoing operational expenses. On the CLAIX systems at the RWTH Aachen University, performance monitoring data is collected to analyze hardware utilization for this reason. By predicting the power consumption and heat dissipation of the cluster with the monitoring data, the cooling controls of the cluster could be optimized and automated. To achieve this goal, it is first necessary to verify the reliability and correctness of the data. This work aims to fill the gap by analyzing the reliability of the time series data by searching for missing values. The correctness of the data is assessed by searching for implausible values. This work also presents preprocessing methods to repair missing and implausible values, as well as an analysis of the correlations between performance-related and power-related metrics which could be valuable for future predicting modeling approaches.

Contents

1 Introduction **1**

2 Background **3**

2.1 Hardware Architecture of CLAIX-2018 3
2.2 SLURM 5
2.3 Performance Data of the RWTH Cluster 5
2.4 Benchmarks 9

3 Methods **13**

3.1 Detection of Missing Data 13
3.2 Detection of Impossible Values 14
3.2.1 Identification of Theoretical Peak Performance 14
3.2.2 Identification of Experimental Peak Performances with Benchmarking 15
3.3 Preprocessing 16
3.3.1 Repair of Missing and Impossible Values 16
3.4 Sanity Checks 17
3.4.1 Comparison of Power Consumption Metrics 18
3.4.2 Comparison of Sidecooler Temperatures 18
3.4.3 Comparison of Node Temperatures 18
3.5 Correlations between Metrics 19
3.5.1 Generation of the Correlation Matrix 19

4 Results **21**

4.1 Overview of Missing Data 21
4.2 Overview of Impossible Values 21
4.3 Results of Reparation Strategies 23
4.4 Sanity Checks 29
4.4.1 Comparison of Power Consumption 29
4.4.2 Comparison of Sidecooler Temperatures 30
4.4.3 Comparison of Node Temperatures 32
4.5 Correlation Matrix 33
4.5.1 Comparison of Correlations after applying Repairs 33

5 Conclusion **37**

References **39**

1 Introduction

HPC clusters are expensive, both in terms of the initial purchase as well as the cost of their operation and maintenance. Thus, it is desirable to ensure the hardware is used efficiently. Performance monitoring data of clusters can be useful to analyze the utilization of the clusters hardware. Insight on cluster usage that is gained through monitoring can help to make informed decisions about the future of their HPC systems, like defining requirements for future acquisitions and helping to justify the total cost of ownership (TCO) of a cluster to stakeholders. Monitoring data also serves to document the performance metrics of research experiments and can be helpful for detecting performance issues. A cluster-wide performance monitoring can also enable the detection of malfunctions and failures of compute nodes.
The HPC cluster CLAIX-2018 at RWTH Aachen University uses non-invasive, background monitoring to collect the performance metrics of the entire cluster, which only minimally impacts the performance of the cluster. The different metrics of the performance monitoring data are gathered with various methods. Metrics from the CPUs, caches, memory devices and power-related metrics are collected with Hardware Performance Counters (HWPCs). Other metrics like disk or network-related metrics are collected by the Linux kernel or by the distributed file system software. The IT-Zauber research project of the RWTH Aachen University aims to improve the energy efficiency of hpc clusters by using performance monitoring data to develop digital twins of hpc clusters [1]. These digital twins model the power supply, cooling components and job load of hpc facilities. The digital twin has the potential to be used for a predictive analysis of the clusters power consumption and heat dissipation. By accurately predicting future heat dissipation of the cluster based on the most recent monitoring data, the cooling controls could be optimized and even automated. The available performance monitoring data holds the potential to be used for an accurate modeling approach. However, to achieve this goal, it is first necessary to assess the quality of the monitoring data. While the measurements of HWPCs are often accurate, their reliability depends on the architecture of the system being measured and on the tool used to access the counters. The reliability of the other measurement techniques is also unknown.
This work aims to perform an *Exploratory Data Analysis (EDA)* of the performance monitoring data of the CLAIX-2018 cluster, specifically to assess the correctness and reliability of the monitoring data. If the EDA raises data quality concerns, then methods for automatic preprocessing of the monitoring data need to be proposed. To support predictive modeling in the future, this work also includes an analysis of the relationship between performance and power supply metrics and searches for correlations. Chapter 2 describes the architecture of CLAIX-2018 and explains the various metrics and their different sources. Chapter 3 describes the methods used to detect missing and

invalid values across the various metrics. The chapter also covers how correlations between different metrics were examined. Chapter 4 presents the results of the methods described in Chapter 3 and compares the reliability of the various metrics and draws conclusions about the reliability of the different sources. This work concludes with a summary of the different methods and their findings, as well as an outlook for future work, in Chapter 5.

2 Background

The upcoming sections will present the essential information for conducting the EDA of the cluster monitoring data. The first section covers the architecture of the hardware of the cluster from which the data is collected. This will be followed by an explanation of the software which allows users to execute jobs on the cluster. It will also describe how meta-data about the jobs are collected and stored. Subsequently, the third section presents an overview of the different sources of performance-related metrics on the cluster. The metrics, which are collected by each source, will also be listed. The final section covers different benchmarks that can be used to gauge the maximum values the cluster is capable of achieving across various performance metrics.

2.1 Hardware Architecture of CLAIX-2018

CLAIX-2018 is the current compute cluster of the RWTH Aachen University and operates since February 2019 [2]. It is equipped with 1032 compute nodes. Each node contains 2 Intel Xeon Platinum 8160 CPUs with 24 cores each at a base clock frequency of 2.1 GHz, as well as 192 GB of RAM [3]. Each node is split into 2 sockets. Each socket has one CPU and its own dedicated memory. The cluster also contains 48 compute nodes which are equipped with 2 NVIDIA Volta V100 GPUs that can be used as accelerators for suitable applications.
The compute nodes are arranged in racks. The racks are split up into several rows and each row contains 7 racks. Within each rack, there are 18 chassis. Each chassis contains 4 nodes, meaning there are 72 nodes inside each rack. Every rack is supplied with power by 2 Power Distribution Units (PDUs) and each chassis is connected to the PDUs by a Power Supply Unit (PSU). The inside of a rack is shown in Fig. 2.1. The racks are cooled by the sidecoolers which are placed in the gaps between the racks. The leftmost rack in each row is only cooled by the one sidecooler to its right while the other racks including the one at the right end of the row is cooled by two sidecoolers. The structure of racks and sidecoolers in a row can be seen in Fig. 2.2. The sidecoolers use air cooling to reduce the temperatures of the nodes. During the cooling cylce, the fans of the sidecoolers blow air into the racks on each side of each sidecooler. The air enters the rack in the front and then travels through the rack to the back where it returns to the sidecooler. During this process, the air heats up which is why the air is the coldest at the front of the racks and hotter at the back. The sidecoolers also use water cooling to transport heat away. During the water cooling cycle, cold water enters each sidecooler and flows through the sidecooler where it absorbs heat. The water then leaves the sidecooler at a higher temperature compared to when it entered.

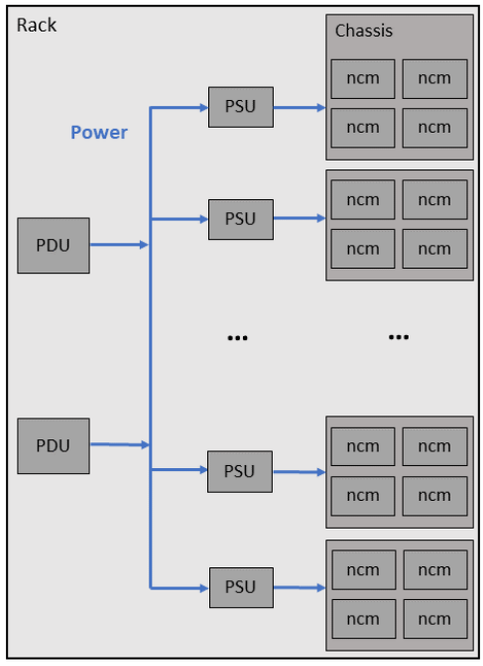

Figure 2.1: Front view of the inside of a rack row.

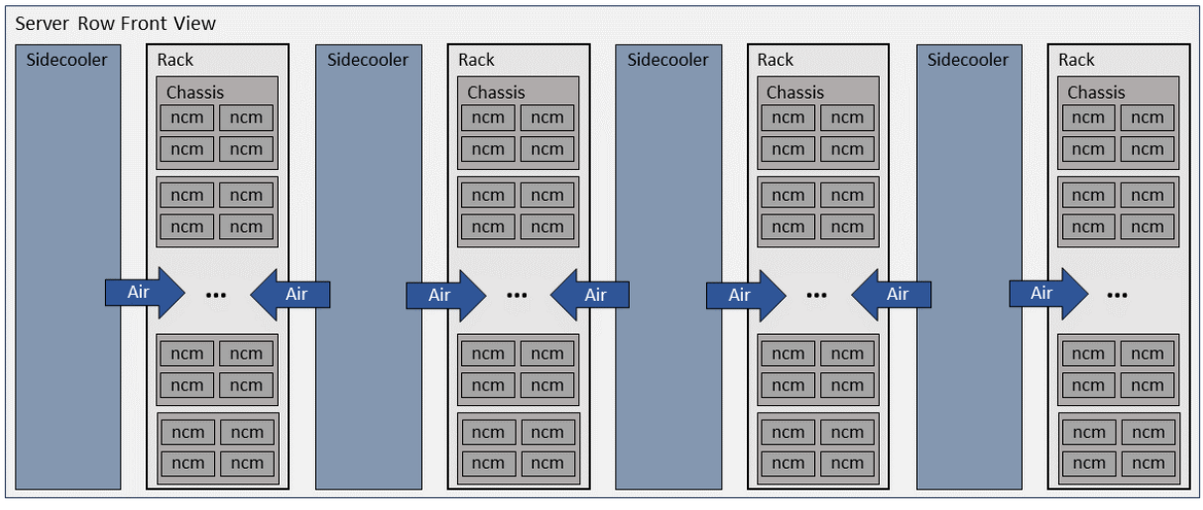

Figure 2.2: Front view of a row of racks and sidecoolers.

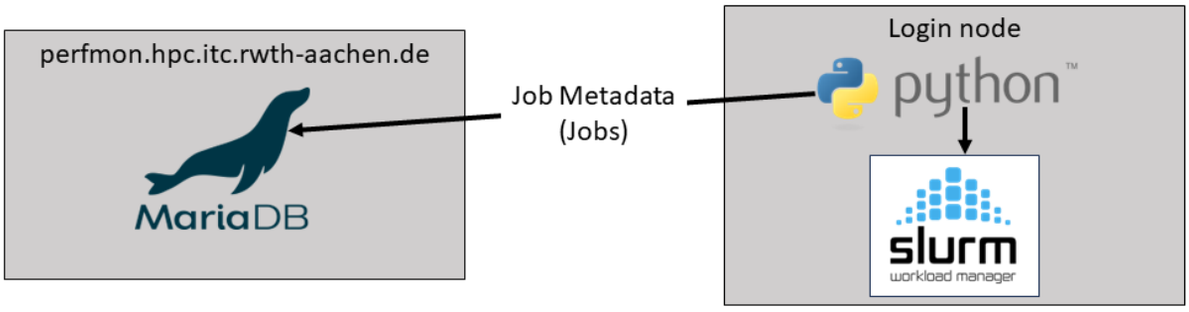

Figure 2.3: Collection of SLURM job metadata.

2.2 SLURM

The *Simple Linux Utility for Resource Management (SLURM)* is a job scheduler and workload manager, and it is commonly used for supercomputers and compute clusters [4]. On CLAIX-2018, users can use SLURM to request exclusive or non-exclusive access to the compute nodes to execute their jobs on the cluster. Users can request access to the compute nodes by submitting a jobscript to SLURM which contains the commands that the user wants to run on the nodes. The jobscript should also include information about the size of the job like the maximum runtime and the number of nodes that SLURM has to allocate. SLURM manages the queue of pending jobs and allocates nodes to users accordingly.
A Python script on the cluster regularly retrieves the data that SLURM tracks about the jobs and sends it to a MariaDB database. This process is illustrated in Fig. 2.3. For each job, the MariaDB contains the start and end time of its execution and the resources the job was given as well as a copy of the jobscript.

2.3 Performance Data of the RWTH Cluster

The CLAIX Monitoring data is collected every minute by instances of Telegraf which run on each node of the cluster. Telegraf is a server agent for collecting and sending data from computer systems and sensors to databases [5, 6]. Time Series data is stored in an InfluxDB instance which is a Time-Series Database (TSDB). Inside the InfluxDB instance the data is split up into two databases. The database *Cluster* contains the performance data of the nodes and the database *Environment* contains the operational data from the cluster environment like the power consumption and information about the cooling.
Most data in the *Cluster* database are measured for every node however some are measured per socket and others others even for every single CPU core. Table 2.1 shows the different metrics in the *Cluster* database, as well as the granularity and source of their measurement. The following list describes the meaning of the individual metrics in greater detail.

Table 2.1: Overview of metrics contained in *Cluster* database.

Metric	**Granularity**	**Source**	**Short Description**
PSU Power	per Chassis	Ipmitool	Power Consumtpion measured by individual PSUs.
RAPL Power	per Node	Likwid	Power consumption measured by individual nodes.
Memory Bandwidth	per Node	Likwid	Speed at which bytes can be written into or read from the memory.
Infiniband Bandwidth	per Node	OS	Speed of parallel read and write operations between different compute nodes.
Lustre Bandwidth	per Node	OS	Speed of read and write operations on the parallel distributed file system of the compute cluster.
Active Cores	per Node	Likwid	Number of active cores of each node.
Node Temperature	per Node	lm-sensors	Temperatures measured inside of compute nodes.
Flops DP	per Core	Likwid	Floating Point Operations of Double Precision Datatypes (64 Bit).
Flops SP	per Core	Likwid	Floating Point Operations of Single Precision Datatypes (32 Bit).
CPI	per Core	Likwid	Average number of clock cycles per instruction.
Clock Frequency	per Core	Likwid	Average number of clock cycles per second.
L3 Miss Rate	per Core	Likwid	Ratio of caches misses to total of memory requests.
CPU Usage	per Core	OS	Percentage of time during which the CPU core is active.

- **Flops**: The floating point operations per second refer to the number of mathematical operations performed with floating point datatypes, like doubles and floats. The Flops for single precision datatypes (32 Bit) and double precision datatypes (64 Bit) are measured separately.
- **Clock Frequency**: This metric refers to the number of instructions a CPU core performs per second. This value often deviates from the base frequency of 2.1 GHz since the clock frequency can be increased dynamically up to 3.7 GHz with Intel Turbo-Boost. The frequency can also be dynamically reduced through *Dynamic Voltage and Frequency Scaling (DVFS)* to optimize power consumption depending on the required performance.
- **Cycles per Instruction (CPI)**: CPI refers to the average number of clock cylces a CPU core needs to complete an instruction.
- **Memory Bandwidth**: The memory bandwidth refers to the speed at which bytes can be written into or read from the memory and is measured per node in MegaByte per second (MB/s) with Likwid.
- **PSU Power**: *PSU Power* refers to the power consumption for each chassis and is measured for each PSU.
- **RAPL Power**: *RAPL Power* refers to the power consumption that is measured for every node.
- **Lustre Bandwidth**: The lustre bandwidth refers to the speed of read and write operations on the parallel distributed file system of the compute cluster.
- **Infiniband Bandwidth**: InfiniBand is a high-speed, low-latency interconnect that often used in high-performance computing environments. The Infiniband Bandwidth refers to the speed of parallel read and write operations between different compute nodes.
- **Node Temperature**: refers to the temperature measured inside of the compute nodes.

The *Environment* database contains the data that is measured by the devices surrounding the compute nodes like the PDUs and the sidecoolers.

- *PDU Power* refers to the consumption of the power that is used for the racks.
- several metrics from the sidecoolers
  - Air and water temperature in Celsius at different times of the cooling cycle. Air is measured at the front and the back at the rack. Water is measured when it enters and when it leaves the sidecooler.
  - Fan speed of the sidecooler in revolutions per minute RPM.

- The valve percentage determines to what degree the valve is open to let the water enter the rack.
- The vent percentage determines how much air can enter the rack.

The monitoring data contains three types of metrics: performance, power supply and cooling. Power supply metrics measure the amount of power which is consumed by the hardware which is a result of the number and types of operations being performed as well as the rate at which operations are performed [7]. Operations include mathematical operations like the two types of *Floating Point Operations* and different types of memory access operations which are measured in the bandwidth metrics. The rate at which operations are performed is directly linked to the *Clock Frequency*. Other metrics like the CPI, *Memory Bandwidth* and the L3 cache metrics also influence the rate, as they can be bottlenecks for the execution of operations. Most of the performance related metrics are measured with Likwid [8]. Likwid is performance tool suite for the GNU Linux operating system. Likwid contains a tool called *likwid-perfctr* that can access and read performance counters of Intel, AMD, ARM and POWER processors and Nvidia GPUs. Performance Counters are special registers of modern multicoreprocessors which provide a low-level, hardware based mechanism for counting the occurrences of specific performance related events within computer systems. On CLAIX-2018, Likwid is used by the Telegraf instance to measure the performance counters of all CPUs. These measurements are collected every minute. Some metrics, like the Flops and CPI, are measured for every CPU core while others, like the Memory Bandwidth, are measured per socket and later aggregated inside the InfluxDB to node level.
The collection process of the performance data is illustrated in Fig. 2.4. Other metrics are collected through the OS. The Infiniband Bandwidth is collected by the OS through the Infiniband Driver HFI1 and written into logfiles on the cluster. Telegraf then reads the files and stores the data in the InfluxDB.
The values for the Lustre Bandwidth are provided by the Lustre clients that run on each node. The metric is collected for each Lustre mount. Each socket has one Lustre mount, so there are two values for every node. Each mount provides the information about the amount of data that was written and the amount that was read. Inside of the InfluxDB the values are aggregated to node level by taking the mean number of each bytes either written and bytes read and converting the results into the rate of change per second, resulting in the bandwidth.
The values of the PSU metric are reported by the *Intelligent Platform Management Interface (IPMI)*. IPMI is an open standard for monitoring and logging the activity of hardware that is implemented independent of the main CPU, BIOS, and OS. `Ipmitool` is a utility program for devices that support IPMI. `Ipmitool` provides a simple command-line interface which can be used to read and display the data that is monitored by IPMI. The Telegraf instances on the cluster run `ipmitool` to retrieve the power consumption of the PSUs and store the data in the InfluxDB in the *PSU Power* metric.
The node temperature values that are contained in the *Node Temperature* metric are collected with Linux-Monitoring Sensors (lm-sensors) which is an open-source software-tool that can be used to monitor temperatures, voltage, humidity, and fans.

Table 2.2: Standard Kernels of the STREAM benchmark.

Kernel name	Array Operation
Copy	c[j] = a[j]
Scale	b[j] = scalar * c[j]
Add	c[j] = a[j] + b[j]
Triad	a[j] = b[j] + scalar * c[j]

The PDU and sidecooler data is queried directly from the PDUs and sidecoolers using the *Simple Network Management Protocol (SNMP)* which is a standard protocol for collecting and organizing information about managed devices on IP networks. The PDUs measure the power consumption accurate to 1 %.

2.4 Benchmarks

This section covers the different benchmarks and tools which were used to measure the highest possible values of various metrics. These values were used to assess the plausibility of the values in the time series data and to clean implausible values. The *Intel Memory Latency Checker (Intel MLC)* [9] is a tool for measuring the memory latency and bandwidth of computer systems. The Intel MLC provides several options to measure the latency or bandwidth at different levels of granularity. The option *max_bandwidth* measures the peak bandwidth of a system with varying amounts of read and write operations. The *max_bandwidth* option includes the bandwidth with only read operations as well as with a 3:1, 2:1 and 1:1 ratio of read and write operations.
Another benchmark that was employed is the STREAM benchmark. The STREAM benchmark is a tool that is commonly used to measure the sustained memory bandwidth of computer systems [10]. During execution, STREAM initializes several arrays of the same size and applies different combinations of read, write and mathematical operations to them. The combinations of operations are called kernels and the memory bandwidth is measured separately for each one. The STREAM benchmark contains 4 kernels, shown in Table 2.2. The STREAM benchmark is parallelized such that the workload can be spread between all of the individual CPU cores of a node. Executing the benchmark in parallel increases the rate at which memory accesses are performed and allows to measure the maximal memory bandwidth of the entire node.
LMBench was also used for this work. LMBench is benchmark suite for UNIX [11]. Most benchmarks in the suite are used to measure memory bandwidth or latency. The benchmark `lat_mem_rd` is used to measure memory load latency which refers to the time needed for the CPU to retrieve data from memory of the subsystem. To perform the measurement, the benchmark initializes an array and then loads individual values with a specific offset to each other, called stride length, into the CPU. The array size and the stride length are two mandatory arguments that must be provided when executing `lat_mem_rd`. The array size and stride length are passed as arguments because they

need to be adjusted depending on the characteristics of the system. The array size determines whether the benchmarks measures the load latency of the different caches or of the main memory. To measure the load latency correctly, the stride length must be set to a value large enough such that loading one value does not load the next one through cache prefetching.

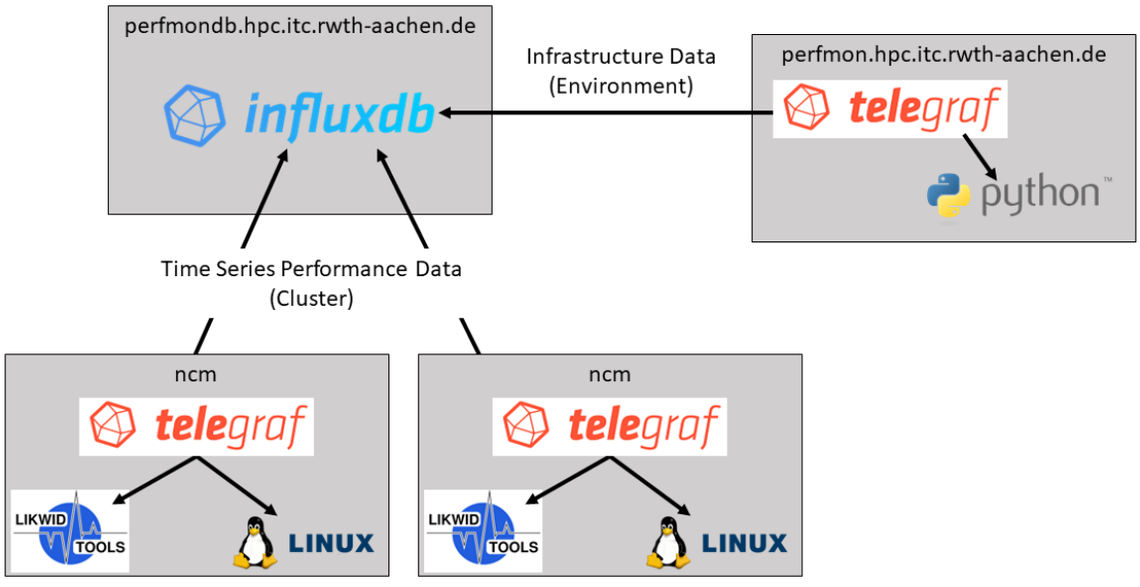

Figure 2.4: Collection of performance monitoring data through Telegraf.

3 Methods

The primary goal of this EDA is to determine what kind of preprocessing is necessary to prepare the Cluster Monitoring data for statistical modeling. At first, it is necessary to verify the integrity of the time-series data by detecting the amount of missing values for the various metrics. The data will also needs to be checked for impossible values. These refer to values that exceed what the cluster can achieve. To do this, the limits for the various metrics must be determined. All values that exceed these limits can be considered invalid and need to be filtered out or repaired. Repairing the dataset by filling in the missing and impossible values through various methods will be the main component of the data preprocessing. The EDA also includes other checks for the correctness of the data which are necessary for the metrics that do not possess a theoretical or experimental peak performance. The last step of the EDA will focus on correlations between the power consumption and the performance related metrics. The goal is to find the metrics with the highest correlation to the power consumption since these metrics should be the most relevant values for the statistical modeling of the power consumption and heat dissipation.

3.1 Detection of Missing Data

Determining the precise number can be achieved by calculating the expected values for a selected timeframe and the designated number of nodes and subtracting the actual number of values in the database. The number of expected values can be calculated by multiplying the number of minutes in the timeframe with the selected number of devices that are supposed to measure the metric every minute. For example, to calculate the expected number of Floating Point Operationss (Flops) for a single rack over the course of the week, we would need the amount of minutes contained in a weeks and the number of cores that the Flops are measured on. Since a rack has 72 nodes with 48 cores each, the total number of expected values equals:

$$72 * 48\,\text{cores} * 7 * 24 * 60\,\text{minutes} = 34,836,480\ \text{expected values}$$

This calculation can be used to create an overview of the amount of missing values across all metrics. This overview could be used to compare the integrity of metrics which are tested with different methods or with different granularities.

3.2 Detection of Impossible Values

To identify impossible values in the dataset, it is necessary to first establish the maximum possible values for the various measurements. Some of the maximum possible values can be inferred with knowledge of the hardware. In these cases the maximum values can either be obtained directly from the hardware specifications or calculated based on them. The maximum possible values of the *Memory Bandwidth* and *CPI* could be derived from hardware specifications, however, employing benchmarking can yield a more accurate representation of what can be achieved on the cluster.

3.2.1 Identification of Theoretical Peak Performance

For the *Clock Frequency* metric the limit can be set to 3.7 GHz which is the maximum clock frequency of Intel the Xeon Platinum 8160 CPUs on the cluster with Turbo-Boost [12]. The limits for the *Infiniband* and *Lustre Bandwidth* can both be set to 6.25 GB/s which is the limit for parallel I/O and network bandwidth with the Fat-Tree network topology on the cluster. For the *Floating Point Operations* the limit can be calculated. For other metrics the maximum possible value cannot be determined exactly but can be roughly tested with specific benchmarks. The theoretical limit for the Flops on a single CPU core can be calculated by multiplying the peak clock frequency of the CPU together with the number of operations per cycle. The maximum operations per cycle can be calculated by multiplying the number of possible multiply and add operations per cycle with the vectorization factor. Vectorization is a technique in which multiple operations are combined into one by treating the data as vectors. The CPUs on the cluster support AVX-512 vectorization which means that the maximum size of a vector is 512 Bit [12, 13]. Therefore the vectorization factor for double precision datatypes with a size of 64 Bit is 8 and the factor for single precision datatypes which are 32 Bit is 16. The amount of possible multiply and add operations per cycle depends on the number of Fuse-Multiply-Add FMA units which are units that can perform one multiply and one add floating point operation per cycle and the CPUs on the cluster each have 2 FMA units which means the number of operations per cylce is 4. Theoretically, the peak clock frequency for the CPUs on the cluster is 3.7 GHz, however, this frequency cannot be reached when vectorization is used, especially if multiple cores are active. The peak clock frequency when using AVX-512 with a single active core is 3.5 GHz.

$$\begin{aligned} \text{Maximum Flops} &= \text{Frequency} * \text{Vector} * \text{FMA} \\ \text{Flops DP}_{max} &= 3.5\,\text{GHz} * 8 * 4 = 112\,\text{GFlop/s} \\ \text{Flops SP}_{max} &= 3.5\,\text{GHz} * 16 * 4 = 224\,\text{GFlop/s} \end{aligned} \tag{3.1}$$

With this information, 112 GFlop/s can be employed as the limit for *Flops DP*, and 224 GFlop/s for Flops SP to search the dataset for impossible values.

3.2.2 Identification of Experimental Peak Performances with Benchmarking

There are two metrics that require benchmarking to find their maximum possible value on CLAIX: These are the *Memory Bandwidth* and *CPI*. However, the *Memory Bandwidth* must be differentiated between the read bandwidth and the write bandwidth. Because of this, it is necessary to run two different benchmarks, one with pure read operations and another with pure write operations. The Intel Memory Latency Checker has a `max_bandwidth` option that performs various measurements with different proportions of read and write operations. The `max_bandwidth` option does contain a measurement with only read operations. After running the Memory Latency Checker 10 times on the cluster, each time on one exclusive node the mean read bandwidth of all the runs was 222,260.29 MB/s and the maximum of the runs was 231,848.9 MB/s. It should be reasonable to employ 232,000 MB/s as the limit for the read bandwidth as it is the slightly rounded up maximum of all the runs.
Since the Memory Latency Checker does not have an option to measure the pure write bandwidth, another benchmark must be employed for this purpose. The STREAM benchmark is commonly used to measure the bandwidth of compute clusters. However, none of the four kernels is suited to purely measure the write bandwidth. For this reason, I changed the Copy kernel such that it sets the array element to a literal value instead of another array element then it becomes a pure write operation and can now be used to determine the limit of the write bandwidth. After executing the STREAM benchmark on one node with all 48 cores a total of 10 times, the maximum bandwidth for the new fill kernel was 152,897.4 MB/s. The rounded up value of 153,000 MB/s was then used as the limit for the write bandwidth.
To find the maximum possible value for the *CPI*, it is necessary to find a benchmark that maximizes the duration of its instructions. For this purpose, the `lat_mem_rd` benchmark from the LMBench suite was chosen, as the benchmark's instructions are prolonged since the data for each instruction must be retrieved from the systems memory. To make the instructions as long as possible, the data must be retrieved from the main memory, not from the caches. To achieve this the data must be larger than the capacity of the largest cache. The L3 Cache of the compute cluster of CLAIX-2018 has a capacity of 16 MByte. To measure the highest possible *CPI* values, the benchmark was executed with array sizes of up to 4 GByte. The measurement was performed with Likwid. The `lat_mem_rd` benchmark was executed with *likwid-perfctr* and after executing the benchmark 10 times, the highest measured *CPI* value was 58.6. This value was rounded up and 60 was then used as the limit for the *CPI*. To summarize, all the limits that were established for performance related metrics are shown in Table 3.1. These limits were then used to verify whether there are impossible values in the dataset and count the number of impossible values for the various metrics and compare the results. After the detection of missing and impossible values, different methods were tested to repair them in the preprocessing stage to improve the data quality.

Table 3.1: Established limits for identifying impossible values.

Metric	Granularity	Limit	Unit
Clock Frequency	per Core	3.7	GHz
Flops SP	per Core	112	GFlop/s
Flops SP	per Core	224	GFlop/s
CPI	per Core	60	
Memory Bandwidth - Read	per Host	232000	MB/s
Memory Bandwidth - Write	per Host	153000	MB/s
Lustre Bandwidth	per Host	6.25	GB/s
Infiniband Bandwidth	per Host	6.25	GB/s
CPU Usage	per Core	100	%

3.3 Preprocessing

The goal of the preprocessing is to improve the quality of the data which is reduced by the existence of both missing and impossible values. One approach would be to simply ignore the missing values and filter out the impossible values with the established limits. However, if too many individual values on the core or node level are missing throughout the time series data, that could falsify the total activity on the rack level that is observed at any given time. And this might negatively impact the correlations that can be observed between the power related metrics and the power consumption and temperature related metrics and therefore decrease the usefulness of the data when it comes to predicting power consumption and heat dissipation in the future. Because of this, it is necessary to fill the gaps that are caused by missing and impossible values and attempt to reconstruct the original information that was lost. In the future, the preprocessing should be applied when collecting samples for statistical modeling. It should also be applied to the recent monitoring data when it is used to predict the power consumption and heat dissipation of the cluster.

3.3.1 Repair of Missing and Impossible Values

Missing values result in entire rows being absent from the database, meaning the timestamp and device information are missing as well. To repair the missing values, it is first necessary to recreate the missing rows with the correct timestamps and device information. The actual values in the newly recreated rows are first set to NaN so that they can easily be identified and repaired. The impossible values of a metric are also set to NaN so that they can be repaired alongside the missing rows. For this process, I first checked if all of the nodes that are expected to be in the dataset are present. This is necessary because sometimes nodes can be shut down for periods of time. And if one or multiple nodes are missing then it would be a waste of resources to insert an entire time series of NaN values into the datasets since those values cannot be repaired anyway. Now that the dataset is filled with NaN values, the pandas library can be

employed since it offers functions specifically to fill in NaN values in dataframes. In total, 3 different strategies were used to fill in the NaN values:

- **Constant Fill**: The NaN values are filled with a constant values, in this case zero.
- **Forward Fill**: Each NaN value is set to the last valid value in the time series. Consecutive NaN values will be set to the same value.
- **Linear Interpolation**: The NaN values are interpolated between the previous and the next valid value. Consecutive NaN values will be evenly spaced on the linear slope between the previous and next valid value.

These different strategies can be applied to the dataframe with the `pandas.fillna` as well as the `pandas.interpolate` method. When using *Constant Fill* all the NaN values will be always be corrected. However, since *Forward Fill* requires a previous valid value and *Linear Interpolation* requires both a previous and a next value, these methods are not able to repair NaN values at the edges of the selected timeframe. To determine the effectiveness of both methods, the number of NaN values was counted before and after applying the repair operation. After applying the three different methods, the impact of the fill operations on the time series data will be compared. The distribution of values can be compared between the different strategies as well as with the raw data. This can be achieved visually with the use of histograms but also with the use of the five-number summary which is a set of descriptive statistics that is commonly used to provide a statistical overview of a dataset. The five-number summary contains the 0 %, 25 %, 50 %, 75 %, and 100 % quantiles of a dataset. Ideally, the impact of the preprocessing methods on the distribution of the metrics should be as small as possible.

3.4 Sanity Checks

In previous sections, the correctness of the various metrics could only be tested through comparison to the peak performance of the cluster. However, the metrics listed below are unique since it is possible to formulate concrete expectations of how they should behave compared to other metrics based on the knowledge of the architecture of CLAIX-2018.

- The power consumption measured by each the PDUs, the PSUs and with RAPL.
- The sidecooler temperatures which measure the air temperature at the bottom and at the top of each rack. The sidecooler data also contains the water temperature which measures the temperature of the water entering the sidecooler as well as the water leaving it.
- The Sensor LT temperatures which measure the temperature inside the compute nodes.

The following sections describe various sanity checks which were employed to verify the correctness of these metrics.

3.4.1 Comparison of Power Consumption Metrics

While the measurements PDU, PSU and RAPL all measure the power consumption on the cluster, they each measure it at different levels of granularities. Each of the two PDUs in each rack measures the power that they distribute to the PSUs. The PSUs measure the consumption of the four nodes that are contained in their chassis while RAPL measures the power consumption per node. However, RAPL only measures the power consumption in parts of the node. To compare the three different metrics I aggregated them to the rack-level to verify if the sums of the three metrics are equal to one another. In theory, for each rack the sum of both PDUs should be equal to the sum of the 18 PSUs and equal to power consumption of the 72 node measured with RAPL. Since RAPL does not measure the entire power consumption of the nodes, it should be expected for the RAPL measurement to be much lower than both PDU and PSU. PDU and PSU should be noticeably higher and closer together and since the PDUs distribute power to the PSUs, the power consumption of the PSUs cannot be higher than that of the PDUs. To see if the power consumption metrics meet this expectation I plotted their sums at individual timestamps against the time series.

3.4.2 Comparison of Sidecooler Temperatures

The sidecoolers measures four temperature values in total. The air and the water temperature both have two different measurements, a *cold* and a *warm* value. The *cold* air temperature is measured at the front of the rack and the *warm* at the back. For the water temperature, the *cold* value refers to the temperature of the water entering the sidecooler and the *warm* value measures the water leaving the sidecooler. Naturally, the *cold* temperature is expected to be lower than its warm counterpart. Another expectation is that the air temperature is considerably higher than the water temperature since the air is measured inside the rack and has direct contact to the source of the heat, the nodes, while the water only flows through the sidecooler. To verify if the sidecooler data meets these expectations, I compared the mean values of the sidecooler temperatures for a few individual racks in a bar plot.

3.4.3 Comparison of Node Temperatures

The Sensors LT metrics measures the temperature for each of the 72 nodes within each rack. The temperature is expected to increase with the physical height of the nodes in the rack, since hot air rises to the top of the rack, making the air cooling less effective. To check if there are significant differences between the temperatures of nodes in the same chassis, the mean difference between the maximum and minimum *Node Temperature* values in each chassis was determined. The mean difference turned out to be quite high at 13.8 °C. Observing the minimum node temperatures revealed several extreme outliers with relatively cold node temperatures of only 40 °C while the majority of the temperatures was well above 60 °C. These outliers are most likely caused by inactive nodes, which might have been in maintenance or shut down temporarily, and cooled

down during this period. I decided to use both the maximum and mean value of the node temperatures in each chassis for this sanity check. The comparison was performed with the mean node temperatures of a one week long timeframe to compare the different chassis temperatures based on their height.

3.5 Correlations between Metrics

To detect correlations between the performance related metrics and the power consumption metrics, I decided to make use of the Pearson correlation coefficient. The Pearson correlation coefficient is commonly used to test different metrics in a dataset for positive or negative linear correlation [14]. The coefficient of two features X and Y is calculated by dividing the covariance *cov(X, Y)* of X and Y with the product of the standard deviations S_X and S_Y of X and Y.

$$X = x_1, ..., x_n \text{ and } Y = y_1, ..., y_n$$

$$\text{with } \overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \text{ and } \overline{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

$$\text{and } cov(X,Y) = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}$$

$$\text{and } S_X = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2} \text{ , } S_Y = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(y_i - \overline{y})^2}$$

The Pearson correlation coefficient is then defined as:

$$r_{XY} = \frac{cov(X,Y)}{S_X S_Y} = \frac{\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}}{\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2}\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(y_i - \overline{y})^2}}$$

The coefficient r_{XY} produces results in the range between -1 and 1. If it equals 0, then no correlation between X and Y exists. A value of 1 means that X and Y have a strong positive correlation, meaning an increase in X causes an increase in Y. If r_{XY} equals -1, it signifies a strong negative correlation, indicating that as Y increases, X decreases and vice versa.

3.5.1 Generation of the Correlation Matrix

To compare the Pearson correlation coefficients of various performance metrics with the power consumption during a selected timeframe, I aggregated the various metrics so that each metric has one value for each timestamp. For most metrics, including the power consumption metrics, I chose to use the sum of all measurements across all devices for each timestamp. But, for the following metrics I used the mean across the

devices instead. Other metrics count absolute values that were measured during each minute which makes it reasonable to use their sum to aggregate the values. But since these metrics are already mean values it makes more sense to use their mean.

- CPI
- *Clock Frequency*
- *Memory Bandwidth*
- *Infiniband Bandwidth*
- *Lustre Bandwidth*
- *Node Temperature*

I used the `pandas.corr` method to calculate the Pearson correlation coefficient for the aggregated metrics. This function returns a matrix which contains the coefficient for every pair of metrics in the original data it is used on. The matrix contains all of the correlation coefficients between the three power consumption metrics and all performance metrics and can be used to determine which metrics show the highest positive or negative linear correlation.

4 Results

This chapter presents the results of the EDA which was described in Chapter 3. The first two sections cover the detection of missing and impossible values and compare the correctness of the different metrics as well as their sources. These sections are followed by a comparison of the reparation strategies which aim to reconstruct the missing and impossible values. The fourth section covers the results of the sanity checks and the final section discusses the correlations that were found between the performance and power-related metrics. All of the tables and figures presented in this chapter were created with a single rack, rack 605, during a timeframe of one week, from the 1st to the 7th September 2023.

4.1 Overview of Missing Data

Table 4.1 shows an overview of the amount of missing values in the monitoring data. *RAPL Power* and *Memory Bandwidth* are measured per node with Likwid and both have the same amount of missing values. The PSU, PDU, Lustre and Infiniband Bandwidth measurements all have no missing values during the selected time period and all of them are measured by the OS. This shows that measurements taken with Likwid are less reliable than other methods and might require more preprocessing. Among the measurements that are made with Likwid there seems to be a big discrepancy between those measured per node and those per CPU core. The metrics *Flops DP* and *Flops SP*, *Clock Frequency*, *CPI* and *CPU Usage* are the only ones measured per CPU core and they have the highest percentage of missing values. All five are missing roughly 9 % of their values while *RAPL Power* and *Memory Bandwidth* are measured per node and both are only missing 0.68 % of their values. The *Flops DP*, *Clock Frequency* and *CPI* stand out, as all three metrics contain the exact same number of missing values with 3,158,246. A closer inspection of these metrics revealed that all of the 3,158,246 missing values occurred at the same timestamps and on the same hardware.

4.2 Overview of Impossible Values

The following Table 4.2 shows an overview of the amount of missing values in the monitoring data. Most of the metrics in the overview have a total of zero impossible values which should make them very reliable. The only two exceptions are the *CPI* and *Clock Frequency*. The *Clock Frequency* only shows a very small percentage of impossible values with only 0.0055 %. Nonetheless, these impossible values still lower

Table 4.1: Overview of missing values for rack 605.

Measurement	Device	**Expected Number of Values**	**Missing Values**	**Missing Percentage**
PDU	PDU	20158	0	0.0 %
PSU	Chassis	181440	0	0.00 %
RAPL	Node	725760	4957	0.68 %
Memory Bandwidth	Node	725760	4957	0.68 %
Lustre	Node	725760	0	0.00 %
Infiniband	Node	725760	0	0.00 %
Active Node	Node	725760	4957	0.68 %
Flops DP	CPU Core	34836480	3158246	9.06 %
Flops SP	CPU Core	34836480	3097169	8.90 %
CPI	CPU Core	34836480	3158246	9.06 %
Clock	CPU Core	34836480	3158246	9.06 %
CPU Usage	CPU Core	34836480	48	0.0001 %
L3 Miss Rate	CPU Core	34836480	3182208	9.13 %

Table 4.2: Overview of impossible values for rack 605.

Metric	Device	Limit	**Total values**	**Found**	**Percentage**
Read Bandwidth	Node	232,000 MB/s	720803	0	0.0000 %
Write Bandwidth	Node	153,000 MB/s	720803	0	0.0000 %
Lustre	Node	6.25 GB/s	725760	0	0.0000 %
Infiniband	Node	6.25 GB/s	725760	0	0.0000 %
Flops DP	CPU Core	112 GFlop/s	31678234	0	0.0000 %
Flops SP	CPU Core	224 GFlop/s	31739311	0	0.0000 %
CPI	CPU Core	60	31678234	6872	0.0217 %
Clock Frequency	CPU Core	3.7 GHz	31678234	1755	0.0055 %

Table 4.3: Overview of values that cannot be repaired with FFill and Linear Interpolation.

Metric	NaN values	FFill		Interpolation	
RAPL	4957	4896	98.77 %	4896	98.77 %
Flops DP	3158246	239696	7.59 %	239696	7.59 %
Flops SP	3097169	242007	7.81 %	242007	7.81 %
CPI	3158246	239697	7.59 %	239697	7.59 %
Clock Frequency	3158246	239820	7.59 %	239820	7.59 %
Read Bandwidth	4957	4896	98.77 %	4896	98.77 %
Write Bandwidth	4957	4896	98.77 %	4896	98.77 %

the correctness of the data and should therefore be repaired. Especially since the limit for the *Clock Frequency* was derived through information about the hardware and is most likely correct. In the case of the *CPI*, the amount of impossible values is also quite low with 0.53 %. It is possible that the amount of impossible values is actually lower, since the benchmark that was used in Section 3.2.2 was not designed to detect the highest CPI values. Because of this the limit that was chosen might be too low. However, when looking at the distribution of the *CPI*, we can see that there are many values that are magnitudes higher than the current limit of 15, with quite a few even going above 1000. Because of this, we can be certain that there are impossible values in the *CPI* metric that must be corrected.

4.3 Results of Reparation Strategies

As described in Section 3.3, I used three different methods to reconstruct the missing and impossible values that were detected in Section 4.1 and Section 4.2. In this section, I will explore the effectiveness of the different methods and compare their impact on the data. The method *Constant Fill* can be used to fill in all of the missing and impossible values since it always inserts the constant value 0 and does not need to take the position of the missing value or its previous and subsequent values into account. The same cannot be said for the other methods *FFill* and *Linear Interpolation*. These methods cannot be used to fill in values at the edges of a dataset which might be problematic if there are many consecutive values. Table 4.3 shows the amount of values that could be repaired with *FFill* and *Linear Interpolation*. In the case of *Linear Interpolation*, the metrics measured per core all had roughly 93 % of their missing values filled in. However, because of the large size of these metrics, a large absolute number of values remains missing. For all of the four metrics, there are still roughly 240,000 values left that could not be repaired. In total, the number of missing values of the metrics measured per core was reduced from 8.9 % to 0.27 %. For the other two measurements *RAPL Power* and the *Memory Bandwidth*, things look very different. For both, only 2.3 % of their missing values could be repaired. The most likely explanation would be that the major-

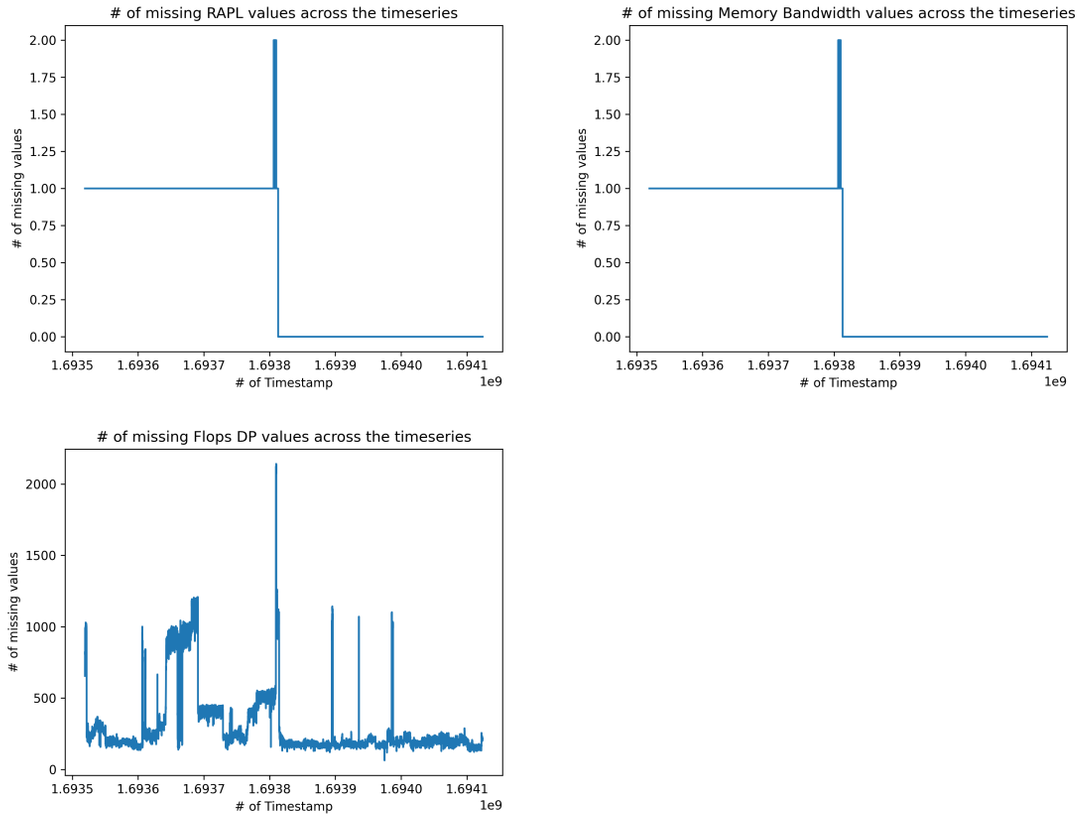

Figure 4.1: Plot of the number of missing values across the time series for metrics RAPL Power, Memory Bandwidth and Flops DP.

ity of missing values is at the edges of the measurement period since *FFill* and *Linear Interpolation* require the surrounding valid values to fill in the NaN values. Both metrics only had 0.68 % of their data missing and 0 impossible values. It appears that the missing values for the two metrics were not as spread out and more concentrated at the edges. To confirm this, I plotted the number of missing values for each timestamp for the *Memory Bandwidth* and for *RAPL Power*, shown in Fig. 4.1. For comparison I also included the metrics *Flops DP* in the graph. In the plots *RAPL Power* and *Memory Bandwidth* behave exactly the same. They both are missing exactly one missing data point from the beginning of the timeframe up until the middle where they have a short peak of 2 missing values and after that there are no more missing values. Assuming that these values are all missing from the same node, this would explain why so many values could not be filled with *FFill* and *Linear Interpolation*. The most likely explanation why so many values from the same node are missing consecutively, is that a job is being executed on that node that wants to use Likwid manually. If users use Likwid manually during a job, the performance counter monitoring will be disabled during the runtime of the job. It is also possible for individual nodes to be missing from the monitoring data for extended periods of time when they are undergoing maintenance. Temporarily missing nodes are relatively common in the monitoring data, as can be seen in Fig. 4.2

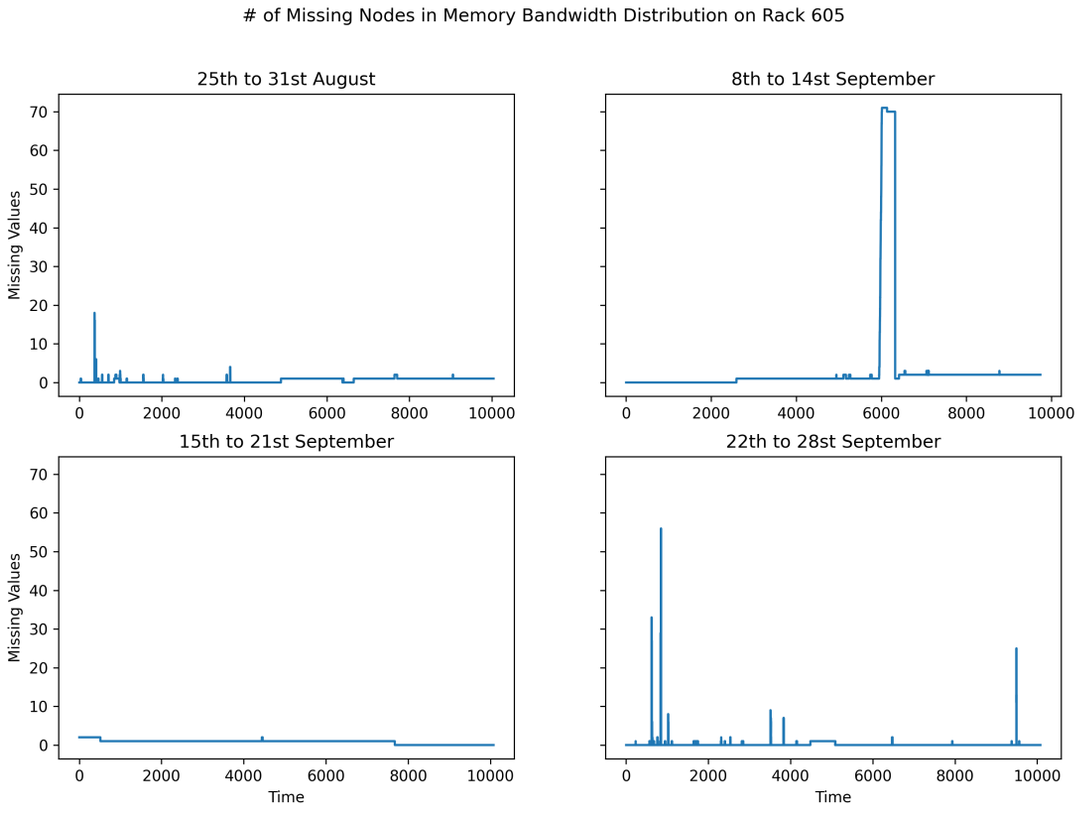

Figure 4.2: Number of missing nodes in Memory Bandwidth in four different one week timeframes.

which shows the number nodes for the metric *Memory Bandwidth* in the surrounding weeks of the usual timeframe. Unfortunately, it is not possible for the *Memory Bandwidth* and *RAPL Power* to be filled because there are no valid values for these metrics at the beginning of the timeframe. This result matches quite well with the number of unchanged values when applying the two methods which was 4896. The timeframe contains 7 days $*$ 24 hours $*$ 60 minutes $=$ 10800 data points per node and since one node is missing its values for roughly the first half of the timeframe, these values should be the 4896 values that could not be corrected through *FFill* and *Linear Interpolation*.

To compare the impact of the reparation strategies on the metrics, I first compared the means of the metrics before and after applying the three strategies, as can be seen in Table 4.4. The metrics *RAPL Power* and *Memory Bandwidth* only show minor deviations within their mean for all three strategies, for all the three the mean value deviates less than 1 % from the initial value. This minor difference stems from the fact that these metrics only had 0.68 % of their data missing. *FFill* and *Linear Interpolation* also had less of an impact on the mean value than *Constant Fill* did which can be explained by the fact that these methods could only repair around 2 % of the missing values of these

metrics.

For the other metrics, the mean deviates much more for all three methods than it did with *RAPL Power* and *Memory Bandwidth*. Especially *Constant Fill* mean values deviate much from the initial mean values, close to 10 % in fact.
The mean values of the *Clock Frequency*, *Flops DP* and *SP* all change similarly. *Constant Fill* reduced the mean the most, of course. But the mean of *FFill* and *Linear Interpolation* was also decreased and both are actually closer to the *Constant Fill* value than to the original. The Flops values deviate between 6 and 7 % while the *Clock Frequency* deviates around 3 %. In general, the mean values of *FFill* and *Linear Interpolation* are very close to each other for almost every metric. The only metric, where the difference in deviation of *FFill* and *Linear Interpolation* is more than 1 %, is the *CPI*. The *CPI* stands out in general, as it is also the only metric where *FFill* and *Linear Interpolation* increase the mean value and not decrease it. *FFill* increases the mean value by 18.25 % while *Linear Interpolation* increases it by 13.06 %. These deviations are also the largest absolute deviations of all the metrics. With *Constant Fill* however, the mean decreases, as it does for the other metrics. If *FFill* and *Linear Interpolation* increase the mean value, then this must mean that the majority of missing values had surrounding values above the initial mean. To confirm this, I plotted the distribution of only the formerly missing *CPI* values after applying the methods *FFill* and *Linear Interpolation*. The results can be seen in figure Fig. 4.3. The figure shows that the distributions when applying *FFill* or *Linear Interpolation* both look quite similar. In both histograms, the occurrences of values above the original mean of 1.64 increases considerably. At the same time, the amount of values between 0 and 1 does not appear to increase at all. The mean of only the reconstructed values is 1.98 for both methods which is higher than the mean value of the raw values, being 1.64.
To gain more information about the changes that come with the reparations strategies I compared the Five-Number Summary Statistics before and after applying the strategies *FFill* and *Linear Interpolation*, especially to find out more about what might cause the decrease of the mean values. The results can be seen in Table 4.5.

Table 4.4: Overview of deviations to the mean value caused by Constant Fill, FFill and Linear Interpolation.

Metric	Unit	**Initial Mean**	**Constant Fill**	**FFill**	**Lin. Inter.**
RAPL	W	286.34 W	−0.68 %	−0.0067 %	−0.0067 %
Flops DP	MFlop/s	443.60 MFlop/s	−9.06 %	−7.17 %	−7.90 %
Flops SP	MFlop/s	83.96 MFlop/s	−8.89 %	−6.37 %	−7.80 %
CPI		1.64	−10.17 %	18.25 %	13.06 %
Clock Frequency	MHz	2563.61 MHz	−9.26 %	−3.21 %	−3.03 %
Read Bandwidth	MB/s	24,310.97 MB/s	−0.68 %	−0.0085 %	−0.0085 %
Write Bandwidth	MB/s	6783.47 MB/s	−0.68 %	−0.0084 %	−0.0084 %

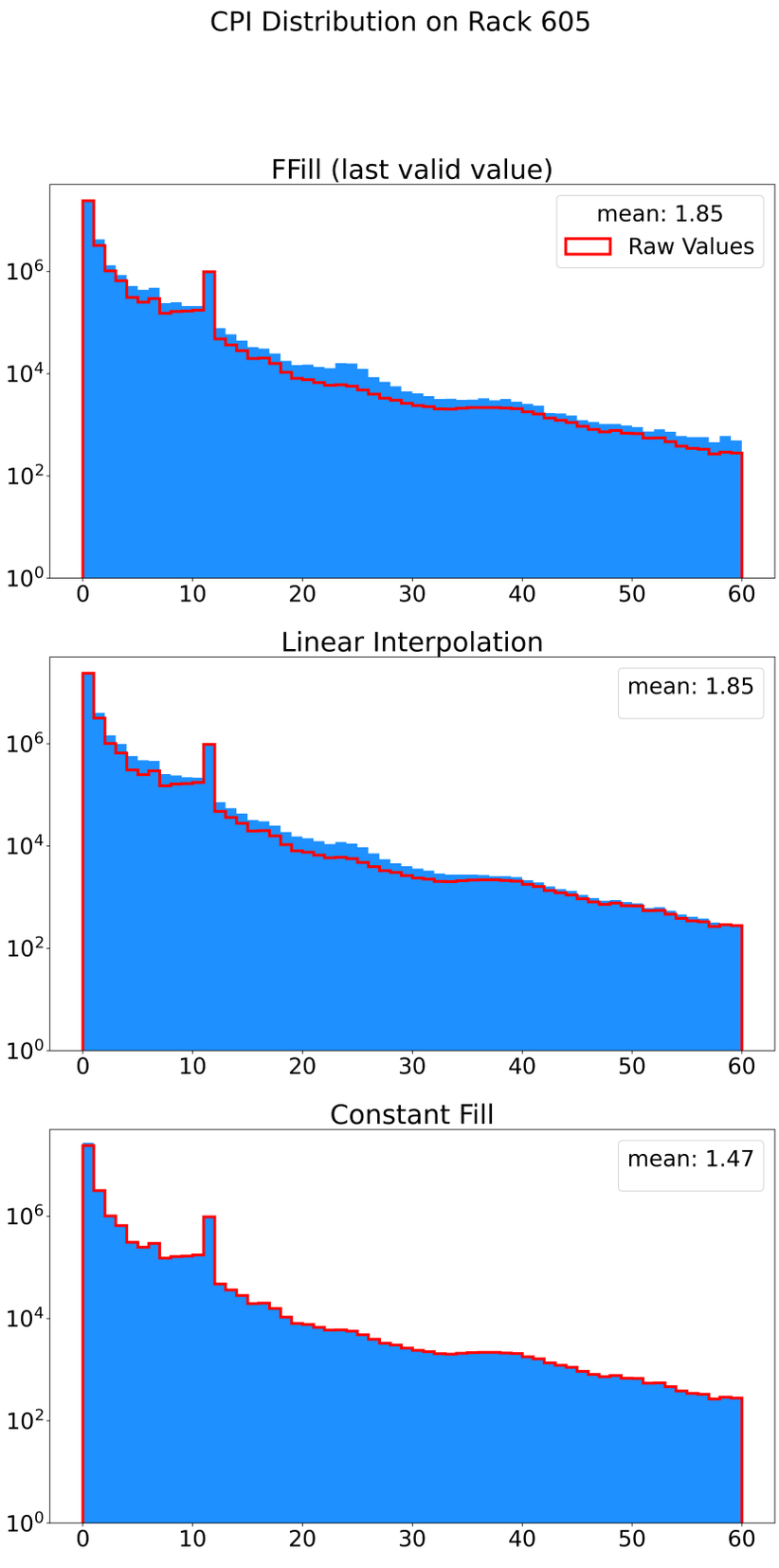

Figure 4.3: Histogram showing the distribution of CPI values after the preprocessing during the timeframe. The original mean is 1.64. The CPI metric has 0.0217 % impossible values and 9.06 % missing values.

Table 4.5: Overview of deviations in the five-number summary statistics after applying Constant Fill, FFill and Linear Interpolation.

Metric	Method	Unit	0 %	25 %	50 %	75 %	100 %
RAPL	Raw	W	36.87	289.78	324.72	332.69	392.12
Power	FFill	W	36.87	289.76	324.72	332.69	392.12
	Interpolation	W	36.87	289.76	324.72	332.69	392.12
Read BW	Raw	MB/s	4.22	2095.42	17399.32	40245.69	209262.64
	FFill	MB/s	4.22	2092.09	17396.50	40244.67	209262.64
	Interpolation	MB/s	4.22	2092.09	17396.50	40244.67	209262.64
Write BW	Raw	MB/s	4.33	628.12	4902.65	11010.30	76420.14
	FFill	MB/s	4.33	628.12	4902.65	11010.30	76420.14
	Interpolation	MB/s	4.33	628.12	4902.65	11010.30	76420.14
Flops DP	Raw	MFlop/s	0.00	32.30	102.34	215.26	96103.72
	FFill	MFlop/s	0.00	11.45	87.03	203.78	96103.72
	Interpolation	MFlop/s	0.00	11.69	87.03	203.72	96103.72
Flops SP	Raw	MFlop/s	0.00	0.00	0.00	0.00	38953.84
	FFill	MFlop/s	0.00	0.00	0.00	0.00	38953.84
	Interpolation	MFlop/s	0.00	0.00	0.00	0.00	38953.84
CPI	Raw		0.25	0.55	0.66	0.97	3990.49
	FFill		0.25	0.56	0.68	1.26	59.99
	Interpolation		0.25	0.56	0.68	1.30	59.99
Clock	Raw	MHz	644.00	2554.70	2639.23	2717.46	3942.43
Frequency	FFill	MHz	644.00	2529.84	2632.13	2716.52	3500.00
	Interpolation	MHz	644.00	2527.38	2630.73	2714.82	3500.00

The five-number summary of the metrics *RAPL Power*, *Flops SP*, *Clock Frequency* and the *Read* and *Write Memory Bandwidth* remains almost entirely the same when applying the methods, deviating less than 1 % from the original values if at all. The metric Flops SP stands out from the others, as the majority of values appear to be zero, indicated by the 75 % quantile of 0 MFlop/s. This is most likely caused by the fact that users of the cluster typically prefer double precision datatypes for their calculations. The maximum values of *Clock Frequency* and *CPI* is reduced from implausible values and now matches their respective limit. The distribution of values only changes significantly for *Flops DP* and for the *CPI*. The upper quartile of the *CPI* increased from 0.97 to 1.26 and 1.3, meaning that the repaired values are on average slightly higher than the median of the original value. For *Flops DP* the opposite is true, the lower and upper quantiles and the median have all been decreased with the biggest change in the lower quartile which is almost reduced to a third from 32 MFlop/s to roughly 11 MFlop/s in both cases.
Overall, the preprocessing methods *FFill* and *Linear Interpolation* keep the distribution of most metrics very similar. These two methods are certainly preferable to *Constant Fill* which seems much more likely to falsify the distribution of metrics. In theory, *Linear Interpolation* should yield more accurate results by blending between the previous and next valid value. However, the method is limited, as it requires both a previous and next valid value. Especially the next valid value might not always be available, e.g. when applying the preprocessing methods to the most recent monitoring data during predictive modeling. Going forward, it might be advantageous to combine *FFill* and *Linear Interpolation*. *Linear Interpolation* could be used first for more accurate results. Afterwards *FFill* could be used to fill invalid values without a previous valid value.

4.4 Sanity Checks

In this section, I will go over the results of the sanity checks which were discussed in Section 3.4. The first subsection will cover the differences in the power consumption measured by the PDUs, PSUs and with RAPL. The second subsection will focues on the comparison of the different sidecooler temperatures. The third and final section will show the comparison of node temperatures at different heights. While these sanity checks cannot prove the absolute correctness of the data, they can help to determine whether or not the data aligns with our expectations on the rack level. If these data do not behave as expected, it would greatly diminish the confidence in the reliability of the data.

4.4.1 Comparison of Power Consumption

Fig. 4.4 shows the differences in power consumption between the three different metrics. The PDUs consistently measured the highest power consumption across the entire timeframe. For the entire time period, the PSU values are only slighty smaller than those of the PDUs. The mean of the PDU metric is 33,564.39 W, while the mean PSU value is

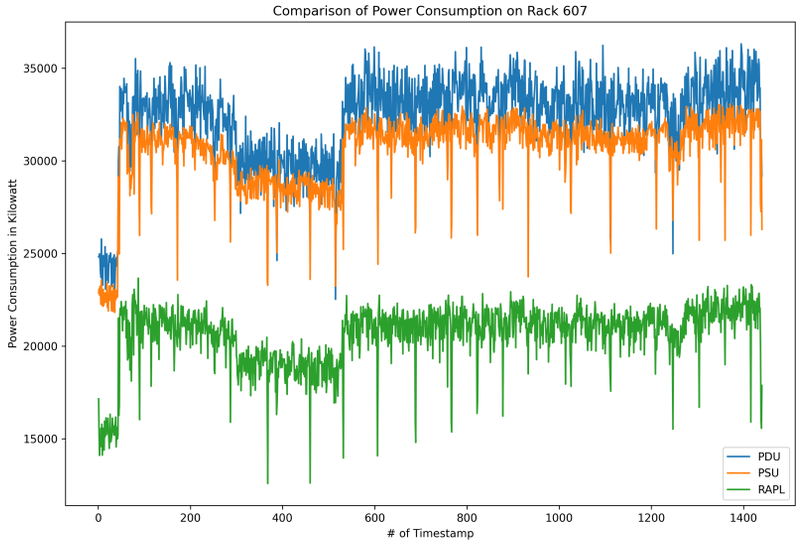

Figure 4.4: Comparison of power consumption metrics: PDU, PSU and RAPL.

6.5 % lower at 31,387.76 W. The *RAPL Power* values are significantly lower than both the PDU at and PSU values throughout the entire timeframe. This is reflected in the mean *RAPL Power* value of 20,476.12 W which is 39 % lower than the mean PDU value. When only examining the graph, it is not possibly to say exactly that the PDU values are always higher than the PSU values. Throughout the entire timeframe there are 49 timestamps out of the total 1440 timestamps with higher PSU power consumption than PDU. This means that 3.4 % of the timestamps do not show the desired behavior. This unexpected behavior might be caused by fluctuations in the exact timing of measurement readings between PSU and PDU. Overall, the power consumption metrics align with the expectations that were established in Section 3.4.1.

4.4.2 Comparison of Sidecooler Temperatures

The bar plot in Fig. 4.5 shows the mean temperatures of the water and air in the cooling cycle of the sidecoolers. The *cold* air metric refers to the air temperature measured at the front of the rack where the fans are, the *warm* air temperature to the air measured at the back. The *cold* water temperature is taken where water enters the rack, the *warm* one is taken when water leaves the rack. The mean of the *cold* air temperature is roughly 11 degrees lower than the *warm* temperature. For the water temperature, the *cold* value is roughly 12 degrees lower than the *warm* temperature. When looking at the five-number summary of the temperatures in Table 4.6, it can be observed, that in both cases the minimum of the *warm* temperatures is several degrees higher than the maximum *cold* temperature. This shows that the sidecooler temperatures meet the expectations established in Section 3.4.2. When comparing the *cold* air and water and

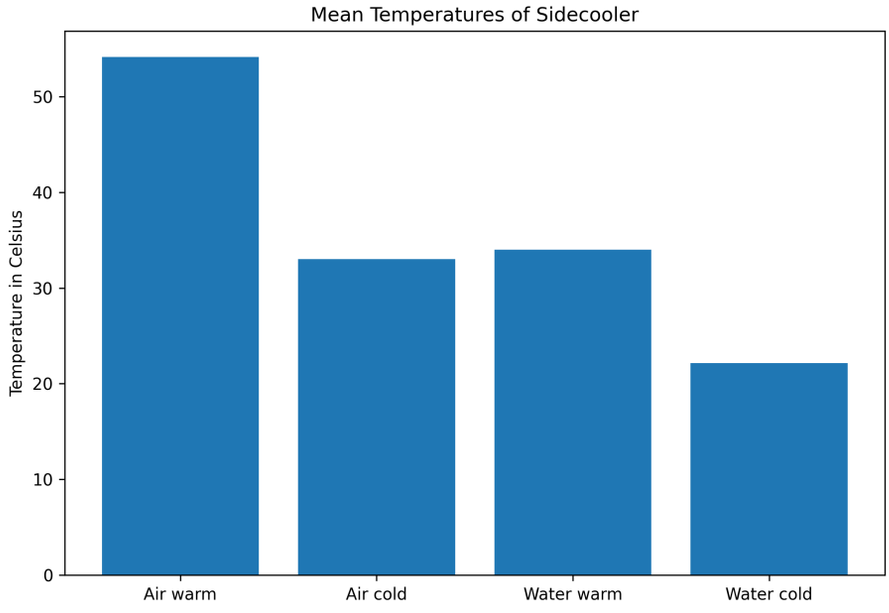

Figure 4.5: Plot of the mean sidecooler temperatures of rack 605.

Table 4.6: Five-number summary of sidecooler temperatures (in Celsius).

Metric		Min	25 % Quantile	Median	75 % Quantile	Max
Air	Cold	31.9	32.9	33	33.1	34.2
Air	Warm	46.3	53.2	55	55.4	56.5
Water	Cold	19.6	22.1	22.1	22.2	23.6
Water	Warm	32.3	33.9	34	34.2	35.3

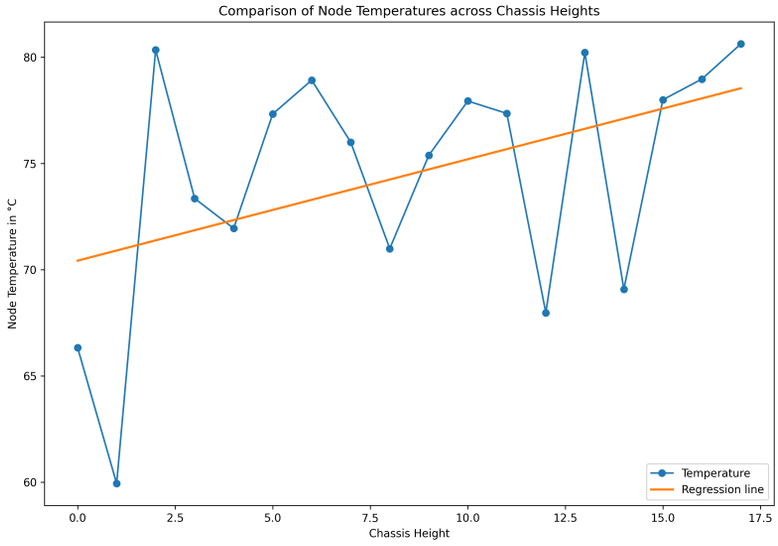

Figure 4.6: Mean node temperatures at different chassis heights.

the *warm* air and water temperature, it is clear that the air temperature is significantly higher than its water temperature counterpart. This is also in line with the expectations for the sidecooler data.

4.4.3 Comparison of Node Temperatures

Fig. 4.6 and Fig. 4.7 shows the mean and maximum node temperatures of each chassis of rack 605 plotted against the physical height of the chassis. While there there is a general trend of the mean temperature rising with the chassis height, as seen in the regression line, it is far less linear than initially expected, which is most likely caused by the outlier values of inactive nodes. The maximum values display an even stronger trend upwards. However, the graph still displays multiple instances of temperatues at higher chassis heights being lower than those of lower chassis heights. This effect could also be caused by those nodes being inactive for some time during the week. Another factor might be the placement of the sidecooler fans. Each sidecooler has 6 evenly spaced fans which would likely result in an uneven cooling of the 18 chassis in one rack. However, in general, these finding still meet the expectation from Section 3.4.3.

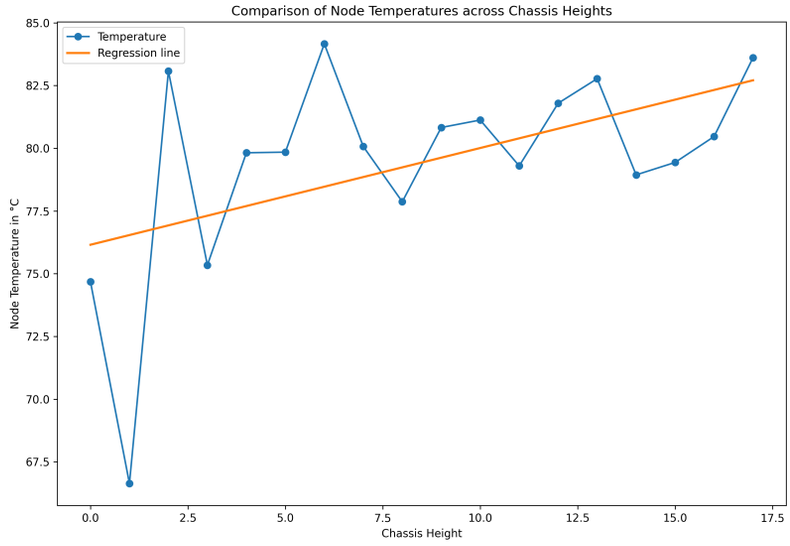

Figure 4.7: Maximum node temperatures at different chassis heights.

4.5 Correlation Matrix

Fig. 4.8 shows the Pearson correlation coefficients between the power consumption metrics and the performance-related metrics in the form of a matrix which was created in Section 3.5. The metrics with the highest correlation to the power metrics are the *Node Temperature*, the *CPU Usage*, the number of active cores and the *Clock Frequency*. This makes sense because the number of active cores, *CPU Usage*, the *Clock Frequency* can be viewed as direct indicators of the activity level and performance of the cluster which should causally be related to the power consumption. On the other hand, the *Node Temperature* can be seen as the result of high power consumption and performance which explains the high correlation. The *L3 Miss Rate* stands as the only metric with strong negative correlation of up to -0.83 since a high miss rate will slow down the execution of instructions and thereby decrease the power consumption. The *CPI* has a lower but still significant amount of negative correlation. This can be explained by the fact that a higher *CPI* value means the CPU is idling while waiting for memory and therefore not able to perform instructions which should cause a decrease in power consumption. The Memory and Infiniband Bandwidth show similar levels of moderate positive correlation.

4.5.1 Comparison of Correlations after applying Repairs

Previously, I only looked at the correlations of the performance data with the power consumption before applying the preprocessing methods. Table 4.7 shows the Pearson correlation coefficient of the metrics that were repaired during the preprocessing. For

simplicity, the table only shows the correlation coefficients with the power consumption metric PDU since the differences of the coefficients between the power consumption metrics were only minor in Section 4.5. The changes when applying the different methods are mostly minor. *Constant Fill* does not change any of the coefficients in a significant manner. *FFill* and *Linear Interpolation* also only show minor effects on most metrics. Both slightly increase the negative correlation of *Flops DP*. Otherwise, the *CPI* and *Clock Frequency* stand out. The correlation changes from -0.32 to -0.48 with *FFill* and -0.49 with Interpolation which is almost a 50 % increase. The *Clock Frequency* coefficient decreases from 0.83 to 0.77 with both methods. These two metrics whose coefficients change the most are also the only two that possess impossible values and both have roughly 9 % missing values which explains the larger deviations relative to other metrics.

Table 4.7: Pearson correlation coefficient with the PDU consumption before and after filling missing and impossible values.

Metric	**Raw**	**Constant Fill**	**FFill**	**Interpolation**
Flops DP	-0.026	-0.026	-0.03	-0.028
Flops SP	0.23	0.23	0.23	0.23
Mean CPI	-0.32	-0.32	-0.48	-0.49
Mean Frequency	0.83	0.83	0.77	0.77
Mean Read Bandwidth	0.53	0.53	0.53	0.53
Mean Write Bandwidth	0.42	0.43	0.42	0.42

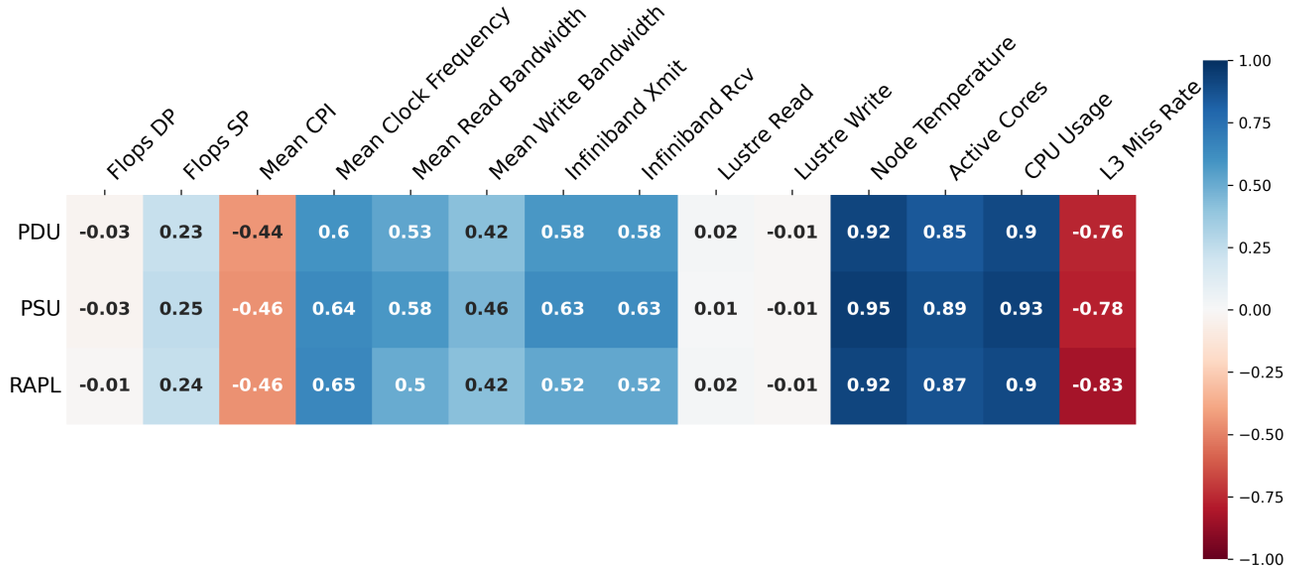

Figure 4.8: Heatmap of Pearson correlation coefficients between performance-related and power consumption metrics.

5 Conclusion

This work presents the results of the exploratory data analysis of the CLAIX-2018 monitoring data. The analysis focusses on the completeness and correctness of the data. Data of individual racks was analyzed within limited timeframes, usually a one-week period, to represent the overall data characteristics, as conducting a cluster-wide analysis across the entire time series was considered impractical. While the quantity of missing values for most metrics was either zero or very close to it, the metrics measured with Likwid per CPU core which are Flops DP and SP, CPI and Clock stood out with high quantities of missing values up to 9.06 %. Whenever possible, the maximum performance values for various metrics were determined, either with theoretical limitations of the hardware or through experimental benchmark measurements. These values were then used as limits to search the metrics for impossible values. The majority of the metrics contained exactly zero impossible values. The metrics CPI and Clock Frequency were the only ones which had impossible values in the selected timeframes and both contained less then 0.5 % impossible values.
Both missing and impossible values were repaired by applying the fill methods Constant Fill, Forward Fill and Linear Interpolation to the timeseries data. The effects of the repair on the distribution of the values were only minor. However, the repairs were successful in increasing the completeness and quality of the data and should from now on always be used as a preprocessing step before processing the monitoring data.
Besides the impossible values, the correctness of the data was also investigated through various sanity checks. Some metrics, for which no peak performance value could be determined, were compared either against directly related metrics or against general expectations which were formulated based on the knowledge about the cluster architecture. Overall, the behavior of the metrics matched the expectations. While the sanity checks cannot definitively prove the correctness of the metrics, their alignment with expectations at the rack level strongly supports their general accuracy.
The metrics were also investigated for linear correlations amongst each other. Specifically, the data was searched for correlations between the power consumption metrics and performance-related metrics.

The metrics with the highest degree of correlation were the *Clock Frequency* with 0.86, *Active Cores* with 0.94, the *CPU Usage* with 0.93 and *Node Temperature* with 0.89. The *L3 Miss Rate* also stood out with a high amount of negative linear correlation of -0.83. Apart from these metrics, most metrics showed a medium amount of correlation to the power metrics. While some performance metrics show far less correlation than expected, the majority showed some amount of correlation with a select few showing very strong linear correlation with the power consumption. In general, most performance metrics act as indicators of power consumption and thus give hope for the

potential of predictive modeling approaches of future works.
My subsequent bachelor thesis will cover the creation of such predictive statistical models for power consumption and heat dissipation. The benchmark suite SPEC-HPC will be used to collect a representative data sample with varying load characteristics to capture all relevant types of system behavior in a condensed timeframe. The preprocessing methods introduced in this thesis will be used to improve the data quality of the sample. Afterwards, the bachelor thesis will aim to highlight the differences of multiple modeling approaches as well as differences between the artificially collected data sample compared to organically collected time series data.

References

[1] *IT-Zauber*. Mar. 27, 2005. URL: https://www.acs.eonerc.rwth-aachen.de/cms/E-ON-ERC-ACS/Forschung/Forschungsprojekte/Simulation-Infrastructure-HPC/~baqzjc/IT-Zauber/?lidx=1 (visited on 01/02/2024).

[2] N. B. Filla. *CLAIX-2018 – schnellster Hochleistungsrechner an einer deutschen Universität!* Feb. 1, 2019. URL: https://blog.rwth-aachen.de/itc/2019/02/01/claix-2018-schnellster-hochleistungsrechner-an-einer-deutschen-universitaet/ (visited on 12/02/2023).

[3] *New compute cluster CLAIX-2018 at RWTH Aachen University*. Nov. 7, 2018. URL: https://www.itc.rwth-aachen.de/cms/it-center/IT-Center/Aktuelle-Meldungen/Aktuelle-Meldungen/~rxnl/CLAIX-2018/?lidx=1 (visited on 11/30/2023).

[4] *SLURM Workload Manager - Overview*. Aug. 6, 2021. URL: https://slurm.schedmd.com/overview.html (visited on 12/02/2023).

[5] *Telegraf*. URL: https://www.influxdata.com/time-series-platform/telegraf/ (visited on 01/04/2024).

[6] *Monitoring*. URL: https://git-ce.rwth-aachen.de/hpc-operations/monitoring (visited on 12/20/2023).

[7] B. Wang et al. "Performance Prediction for Power-Capped Applications based on Machine Learning Algorithms". In: *2019 International Conference on High Performance Computing & Simulation (HPCS)*. 2019, pp. 842–849. DOI: 10.1109/HPCS48598.2019.9188144.

[8] *LIKWID Performance Tools*. URL: https://hpc.fau.de/research/tools/likwid/ (visited on 12/20/2023).

[9] *Intel® Memory Latency Checker v3.11*. URL: https://www.intel.com/content/www/us/en/developer/articles/tool/intelr-memory-latency-checker.html (visited on 12/19/2023).

[10] *STREAM Benchmark*. URL: https://www.amd.com/de/developer/zen-software-studio/applications/spack/stream-benchmark.html (visited on 12/20/2023).

[11] *LMbench - Tools for Performance Analysis*. Dec. 11, 2005. URL: https://lmbench.sourceforge.net/ (visited on 12/19/2023).

[12] *Xeon Platinum 8160 - Intel*. Aug. 24, 2022. URL: https://en.wikichip.org/wiki/intel/xeon_platinum/8160 (visited on 12/05/2023).

[13] *Advanced Vector Extensions 512 (AVX-512) - x86*. Mar. 16, 2023. URL: https://en.wikichip.org/wiki/x86/avx-512 (visited on 12/05/2023).

[14] M. Witkowski et al. "Practical power consumption estimation for real life HPC applications". In: *Future Generation Computer Systems* 29.1 (2013). Including Special section: AIRCC-NetCoM 2009 and Special section: Clouds and Service-Oriented Architectures, pp. 208–217. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2012.06.003. URL: https://www.sciencedirect.com/science/article/pii/S0167739X12001392.