Statistical Characterization of HPC Monitoring Data

Bachelor Thesis

University of Basel
Faculty of Science
Department of Mathematics
and Computer Science
HPC Research Group

Examiner: Prof. Dr. Florina M. Ciorba
Supervisor: Thomas Jakobsche

Monika Multani
monika.multani@stud.unibas.ch

December 21, 2021

Abstract

Scientific applications use high computing power offered by High Performance Computing (HPC) systems to solve research questions. The use of system resources is recorded by monitoring infrastructure. Data from monitoring activities can give us insights into application behavior and system response. The goal of this project is to characterize HPC monitoring data and its statistical properties. An assessment of which system metrics have frequency components is also made, because some monitoring metrics (e.g. CPU) have specific characteristics resembling the waveform of sound data. For this purpose, we implement code in Python to provide an overview of the system metrics and their characteristics. The system metrics are divided into the conceptual groups CPU, Memory and Network and into subgroups to better understand the type of metric and investigate statistical properties of each group. The results show that the majority of the system metrics are always zero, this is important for future research on monitoring data. Researchers have to pre-process monitoring data if they do not want to operate on data that is largely always zero. Furthermore, we show that some system metrics have waveform and the most metrics with waveform come from the network group. The amount of frequency components inside the waveform of a system metric can be dependent on the executed application. Frequency proofed to be a viable statistical feature that can complement traditional features like minimum, maximum, and average. Frequency can not only be used to analyze properties of system metrics, but also to characterize application-specific system response. We also found that some system metrics are very similar to specific musical instruments. This opens up further research opportunity in the direction of applying acoustic signal processing techniques on HPC monitoring data.

Contents

Chapter 1

Introduction

Scientific applications use high computing power offered by High Performance Computing (HPC) systems to solve research questions. The use of system resources is recorded by monitoring infrastructure. Data from monitoring activities can give us insights into application behavior and system response.

1.1 Motivation and Goal

Monitoring data is generated by executing applications and benchmarks on HPC systems. By analyzing this data, it is possible to gain insights into the applications and the HPC system. To better understand the applications and the HPC system (e.g., to detect errors) monitoring data is often analyzed with statistical features like the average load or the maximum CPU usage. But the HPC system still needs to be fully understood, therefore alternative analysis methods need to be investigated.

The goal of this project is to learn more about the metric groups in HPC systems, to gain insight into the system metrics of HPC monitoring data, as well as to characterize HPC data. Investigations for HPC monitoring data will be made and system metrics will be sorted into groups based on their statistical properties. Furthermore, an assessment of which system metrics have frequency components will be made, because some monitoring metrics (e.g. CPU) have specific characteristics resembling the waveform of sound data.

1.2 Research Questions

To achieve the goal, the monitoring data must be analyzed. At this point, the problem is that the methods for analyzing the monitoring data are not clear. A large number of different approaches exists such as the time series analysis with and without machine learning. Simpler methods involve analyzing statistical features like the average load or the maximum CPU usage. But these statistical features do not always capture all characteristics of the data. Additionally, a large number of system metrics exists to choose from, and it is not clear which system metrics are worth investigating.

Hence, the research questions of this project are:

– What kind of metrics are available on the HPC system?

– What are the characteristics of HPC monitoring data?

– Do system metrics have a specific frequency and waveform?

– How similar are system metrics to actual sound?

1.3 Solution and Answers

To solve the problem, the system metrics of HPC Monitoring Data are investigated. By looking at the system metrics, it becomes visible that some of them have sound wave characteristics. There exist monitoring metrics including CPU usage that have frequencies. The initial approach to find a method for analyzing monitoring data was to investigate the frequency of the HPC monitoring data. While investigating the frequency, many other characteristics of the metrics were observed. The project was thus further developed. In addition to frequencies, other characteristics of the system metrics were researched in order to create an overview of HPC Monitoring Data.

For any HPC researcher, a successful achievement could be of importance to get a better overview of the data. HPC data with the relevant characteristics for specialized experiments can then be filtered out. An example of using these characteristics is for researchers to focus on the system metrics with waveforms while analyzing the data through listening, since they are similar to sound waves.

The findings and answers to the research questions include:

- Groups of metrics have been found based on statistical properties that show, among other things, that a majority of system metrics is consistently zero.
- Some system metrics were found to have frequency and most metrics with waveform come from the network group.
- The amount of frequency components inside the waveform of a system metric can be dependent on the executed HPC application.
- Some system metrics are very similar to specific musical instruments.

Chapter 2

Related Work

2.1 Importance of Frequency and Waveform

One aspect of finding system metrics with waveform, is to exploit the waveform for audification in order to listen to the data and try to find out things that may be overlooked by visual analysis. The transformation of data into sound is not something that is uncommon. As mentioned in the handbook "The sonification handbook" by Dombois, Florian, et al., audification was used for example to analyse physical data, general acoustic data or sound recording data. The general goal of data audification is to make it possible to listen to the results of scientific measurements or simulations in order to understand them better. The audification of data is a method of decoding data that can be interpreted as an amplitude over time and be reproduced as sound. [6]

"Tuning Complex Systems by Sonifying Their Performance Data" [12] is a paper that investigates the use of sonification as a cognitive aid to support performance tuning, which was presented as a novel approach to simplify performance tuning of complex computer systems by using sound to transmit performance information during execution. According to this research, sound-based tuning approaches can have valuable solutions concerning these problems.

2.2 Analysis of System Metrics

In order to successfully audify data, first, system metrics, that are suitable for audification (metrics that have a frequency and waveform), must be found. This project focuses on finding these metrics and more generally on characterizing HPC monitoring data.

Maarten Schenk had already taken an approach of turning performance data into sound in his bachelor thesis ”The Sound of Computing” [29] also by investigating sonification through the development of a customized solution. His thesis is primarily based on proof of concept. In this thesis he assigned raw values to a musical scale and played the resulting notes on a digital instrument. The sound of applications can be used to distinguish them from one another. Findings of his work show that it is possible to perform sonification of HPC performance data, whereby new insights can be gained, and it makes the data more understandable.

2.3 HPC Monitoring Data of Applications

Researchers at Sandia National Laboratories and Boston University in the USA have presented the Taxonomist technique in a paper [2]. Taxonomist is a technique that uses machine learning to classify known and unknown applications that are based on easily accessible system monitoring data, even when they are being executed on a supercomputer. The technique is also capable of recognizing previously unknown input configurations from known applications.

As previously indicated, some researchers have already developed techniques to turn data into music in order to listen to the data. Therefore, in this thesis, a further approach is applied to investigate the HPC monitoring data. As Florian Dombois and Gerhard Eckel reported in their handbook [6], not every data is suitable for audification. One of the requirements is that the data has a waveform, which is why the new concept of investigating the frequency of HPC monitoring data metrics came up. For the inquiry, a public dataset, which contains several applications that have been executed on HPC systems, is used.

Chapter 3

Methods and Materials

3.1 Fast Fourier Transformation and Spectrogram

Fast Fourier Transformation (FFT) An important measurement method in audio and acoustic measurement technology is the ”Fast Fourier Transformation” (FFT). The FFT decomposes a signal into individual spectral components and thus indicates its composition. It is possible to assess mistakes, manage quality and monitor the condition of equipment or systems using FFTs. To be precise, the FFT is an optimized algorithm for implementing ”Discrete Fourier Transformation”, DFT for short. In this process, a time-limited section of a signal is broken down into its components. These components are individual sine oscillations at discrete frequencies whose amplitude and phase are determined. The FFT thus allows visualization of a signal in the frequency range. [3]
In very simplified terms, the frequency counts the number of full sine turns or peaks in the time series.

Spectrogram A spectrogram is a visual representation of the frequency spectrum of a signal on how it changes over time [9]. It is common to use spectrograms for representing frequencies of sound waves generated by people, machines, animals, etc. and recorded by microphones [26].

3.2 Time Domain vs. Frequency Domain Representation

The metrics were plotted in time domain and frequency domain representation. The title of the plots includes a list of the applied parameters. First, the application name and the input size are mentioned. Afterwards, the run ID, node ID, metric ID and the metric name are shown. Lastly, information about whether the plot is a ”raw timeseries” or a ”spectrogram + peaks” plot is presented.

The time domain plot visualizes the metric data with its values per second. The time series values are shown on the x-axis and the values of the metrics are shown on the y-axis. The Figure 3.1 shows an example plot for a time series array.

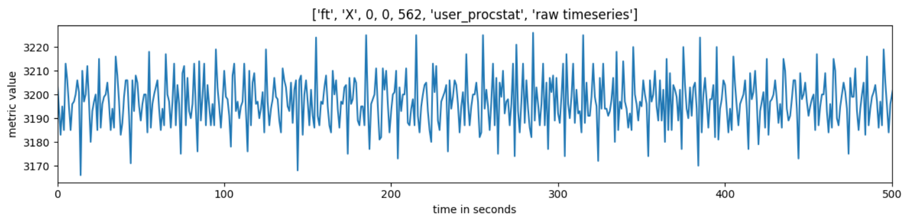


Figure 3.1: Time Domain Representation of Metric 562

To calculate the frequency representation, Spectrogram (see Section 3.1) and Fast Fourier Transform (FFT) (see Section 3.1) are used. The frequency values are presented on the x-axis, while the amplitude values are presented on the y-axis. For the transformation, the function scipy.signal.spectrogram(x, fsfloat) from the open-source software SciPy [35] was used with the parameters x and the optional parameter fsfloat. For the x parameter, the measured values of time series were used, and the sampling frequency of the time series was used for the fsfloat parameter. This function divides a time signal into shorter segments of equal length. To each segment, the FFT was applied and thus the representation of the spectrum on each segment was obtained [28]. An example plot for a frequency plot of a time series array is shown in the Figure 3.2.

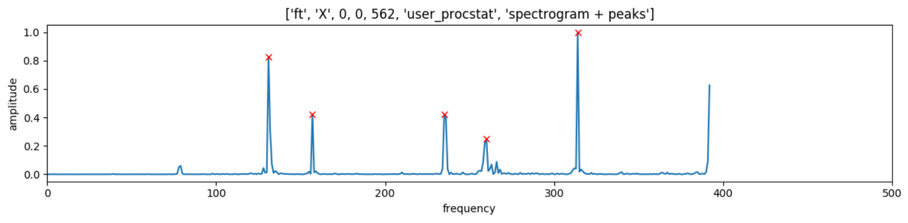


Figure 3.2: Frequence Domain Representation of Metric 562

For a better understanding of the transformation, consider the following example:

In Figure 3.3, a representation in the time domain with a waveform is visible. It can be observed that a wave is repeated several times. To be more precise, in this example the wave marked in red occurs 50 times.

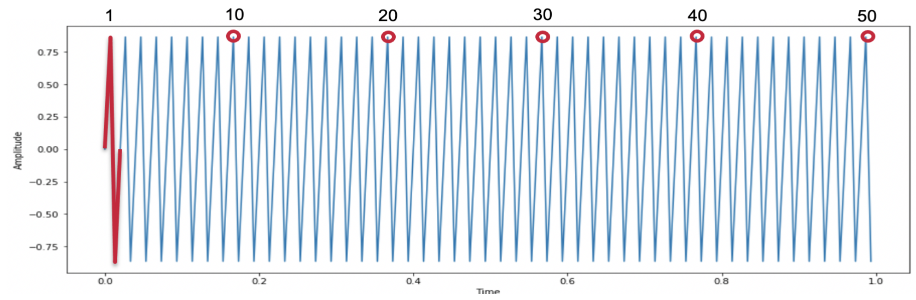


Figure 3.3: Time Domain Representation

The image below (Figure 3.4) illustrates the frequency representation. In the diagram it can be seen, that there is a frequency peak at 50. Therefore, in the frequency domain, the different waves or their proportions are represented as amplitude. Mathematically this is a delta function centred on the fundamental frequency of 50 Hz.

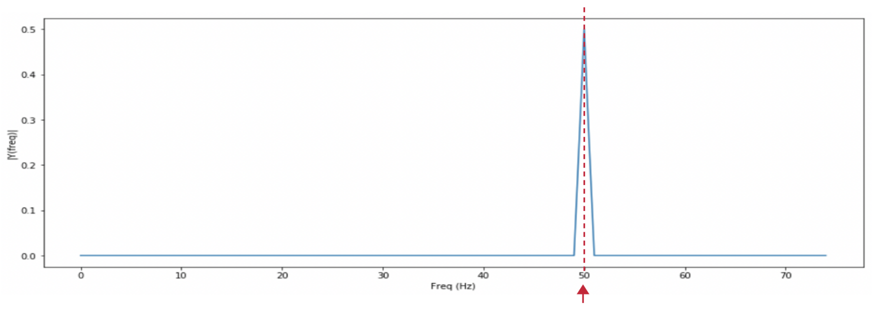


Figure 3.4: Frequency Domain Representation

3.3 Taxonomist Dataset

The public data from Ates et al. [1] published in 2018 was used to investigate frequencies in HPC monitoring data. This data was collected at the time to figure out which applications are running on modern supercomputers using the taxonomic technique mentioned above. HPC performance data from 11 benchmarks that are representative applications (see Section 3.3.1), and 5 different unwanted applications are included in the data. However for the investigation of frequency only the data from the representative applications are used, as the unwanted metrics have not been published. Each application has at least three different input sizes, depending on what they are doing. In total, 721 different metrics per seconds were collected as time series data from each node, of which only 563 have been published. Therefore the used data only includes 563 metrics. Several of these metrics are memory related, while others are CPU related and some are network related. These groups are described in the Section Conceptual Groups and Subgroups (see Section 3.4). The applications with the same configuration were repeated 30 times. In addition, the four applications miniAMR, miniMD, miniGhost and Kripke were executed 6 times with another input on 32 nodes as time series data. All these time series data are used for this research.

Taxonomist Dataset				
Applications	FT, MG, SP, LU, BT, CG, CoMD, miniGhost, miniAMR, miniMD, Kripke			miniGhost, miniAMR, miniMD, Kripke
Input Sizes	X	Y	Z	L
Number of Nodes	4	4	4	32
Metrics per Second	563	563	563	563
Repeated Executions	30	30	30	6

Table 3.1: Taxonomist Dataset

3.3.1 Representative Applications

Application	Description
BT	Block tri-diagonal solver
CC	Conjugate gradient
FT	Fourier transform
LU	Gauss-Seidel solver
MG	Multi-grid on meshes
SP	Scalar penta-diagonal solver
miniAMR	Adaptive mesh refinement miniApp
miniMD	MiniMD molecular dynamics miniApp
CoMD	Classical molecular dynamics proxy application
miniGhost	MiniGhost Halo Exchange Mini-Application
Kripke	3D $\mathrm{S_n}$ deterministic particle transport

Table 3.2: Representative Application

The following descriptions are taken from [17] and [4]:

BT - Block tri-diagonal solver: BT simulates a CFD (computational fluid dynamics) problem with two discrete versions of three-dimensional, unsteady, compressible Navier-Stokes equations. BT solves multiple, independent systems of non diagonally dominant, block tridiagonal equations.

CG - Conjugate gradient: Computation of the smallest eigenvalue of a large, sparse symmetric, positive- definite matrix with a conjugate gradient method. By using unstructured matrix vector multiplication, CG tests irregular long distance communication.

FT - Fourier transform: Testing the performance of long-distance communication. FT numerically solves a Poisson partial differential equation (PDE) using the fast Fourier transform (FFT).

LU - Gauss-Seidel solver: LU is simulating a CFD problem like BT, but solves regular-sparse, lower and upper triangular systems.

MG - Multi-grid on meshes: MG requires highly structured long distance communication and tests the performance of short and long distance data communication. MG computes an approximation for the solution to a three-dimensional scalar Poisson problem on a discrete grid, by using the V-cycle multi-grid algorithm.

SP - Scalar penta-diagonal solver: SP also simulates a CFD problem like BT, but solves multiple, independent systems of non diagonally dominant, scalar, penta-diagonal equations.

The following descriptions were taken from the respective github pages:

miniAMR - Adaptive Mesh Refinement Mini-App: miniAMR applies a stencil calculation on a unit cube computational domain, which is divided into blocks. The blocks all have the same number of cells in each direction and communicate ghost values with neighboring blocks. With adaptive mesh refinement, the blocks can represent different levels of refinement in the larger mesh. (translated from [23])

miniMD - MiniMD Molecular Dynamics Mini-App: miniMD is a parallel molecular dynamics (MD) simulation package written in C++ and intended for use on parallel supercomputers and new architectures for testing purposes. This simple code is a self-contained piece of C++ software that performs parallel molecular dynamics simulation of a Lennard-Jones or a EAM system and gives timing information. [25]

CoMD - Classical molecular dynamics proxy application: CoMD is a reference implementation of typical classical molecular dynamics algorithms and workloads. The code is intended to serve as a vehicle for co-design by allowing others to extend and/or reimplement it as needed to test performance of new architectures, programming models, etc. [5]

miniGhost - MiniGhost Halo Exchange Mini-Application: A broad range of scientific computation involves the use of difference stencils. In a parallel computing environment, this computation is typically implemented by decomposing the spacial domain, inducing a "halo exchange" of process-owned boundary data. MiniGhost represents 3D nearest neighbor halo-exchange communications that are present in a many HPC codes. [24]

Kripke - 3D S_n deterministic particle transport: Kripke is a simple, scalable, 3D S_n deterministic particle transport code. Its primary purpose is to research how data layout, programming paradigms and architectures effect the implementation and performance of S_n transport. (S_n: Discrete ordinates method of approximately solving radiative transfer equations.) [21]

3.4 Conceptual Groups and Subgroups

The following descriptions were taken from various sources.

Network The interconnection of several transmitters for the large-scale distribution of radio and television programs or individual broadcasts is referred to as a network [7]. The network usage information are collected from Cray network interface card (NIC) counters [2].

– **NIC (Network Interface Controller):** A network card, Network Interface Controller (NIC) or Network Interface Card (NIC), is a printed circuit board or chip that directly connects the network to the end device. It can be a plug-in card for a bus system used in the personal computer: for Industry Standard Architecture (ISA), Peripheral Component Interconnect (PCI) or PCI Express, for Versa Module Europe (VME) or microchannel. (translated from [16])

– **RDMA (Remote Direct Memory Access):** Remote Direct Memory Access (RDMA) is a networking technology that allows computers to exchange data in main memory without using either computer's processor, cache or operating system. Because it frees up resources, RDMA, like locally based Direct Memory Access (DMA), improves throughput and performance. In addition, on the one hand, RDMA enables faster data transfer rates and low-latency networks. On the other hand, it can be used in networking and storage applications. [30]

– **SMSG (Short Message Service Gateway):** A Short Message Service (SMS) gateway is a method that allows SMS messages to be sent and received. SMS processes for organizations are facilitated and streamlined with the help of SMS gateways. Some of the conversion into different formats is often done by the gateways. [33]

– **IPoGIF (IP over generic fabric interface):** The HPC system where the dataset for the Taxonomist was collected is a Cray system. Cray uses a reliable and high-performance IP network over the Aries network. This high-performance network is configured as an IP over generic fabric (IPoGIF) interface.

– **Other metrics including Energy and Power:** Energy is the ability to bring about change, for example creating motion.The unit of measure for energy is joule. Power is defined as the amount of energy divided by the amount of time it took to use the energy. Its unit is watt, which is equal to one joule of energy every second. [8]

    - **LOADAVG:** The load average is the CPU demand of a server. The server includes the sum of the running and waiting threads [19]. Depending on the system, the composition varies, as do the usual values. In general the following applies: The lower the load value, the less loaded a system is, i.e. more resources are available. A load average of 0 indicates that all resources are available and the system can process a user's tasks at maximum speed. However, as the system load increases, it becomes more and more difficult as all resources are permanently in use. [20]

Memory The memory metrics are collected by /proc/meminfo and /proc/vmstat [2], for which reason they are regrouped into memory info and virtual memory statistics here.

- **Memory info** (proc/meminfo): The memory info metrics are metrics that provide information about the distribution and use of storage [18].
    - **Active / Inactive:** Memory that is used by a particular process is called active memory. The inactive memory is the memory allocated to a process that is no longer running. There are pages that have not been accessed recently. [31]
    - **Huge Pages:** The memory pages are greater than default. Huge pages are memory blocks that exist in two sizes: 2MB and 1GB. The page tables used by 2MB pages are optimal for scaling to terabytes of memory, whereas the page tables used by 1GB pages are best for maintaining several gigabytes of memory. [22]
    - **Other metrics including cache and shared memory:** A small volatile computer memory that provides fast data access to a processor and stores frequently used computer programs, applications and data is considered as cache memory [13]. Shared memory is a type of memory that can be shared by several applications or processes for the purpose of enabling communication between applications or avoiding redundant data copies [14].
- **Virtual memory statistics** (proc/vmstat) The virtual memory statistics is a computer system monitoring tool that collects and displays summary information about operating system memory, processes, interrupts, paging and block I/O [39]. Virtual memory (VM) is a kernel feature that simulates additional main memory such as RAM (random access memory) or disk storage. This technique includes memory manipulation and management in order to allow the loading and execution of larger programs or multiple programs at the same time. It also allows each program to operate as if it had infinite memory, and it's often cheaper than purchasing more RAM. [34]
    - **Various virtual memory statistics:** This subgroup includes the various virtual memory statistics metrics.

- **kswapd (Kernal Swap Daemon) memory management:** The name swap daemon is a misnomer because the daemon does more than simply swap updated pages to the swap file. Its job is to ensure that the memory management system runs smoothly. Every time the timer expires, the kernel swap daemon (kswapd kernel init process) is started. The swap daemon monitors the number of available pages in the system to detect whether it is getting too low. [27]
- **NR metrics:** The metrics that determine the number of different virtual memory statistic are included here.
- **NUMA (Non-Uniform Memory Access):** Non-Uniform memory access abbreviated NUMA is a multiprocessing memory design in which the memory access time is dependent on the memory location relative to the processor. A CPU can access its own local memory faster than non-local memory when using NUMA. [11]
- **Other metrics including slab and page:** The amount by which a cache can grow or shrink is called a slab and is expressed as one memory allocation to the cache by the computer. Normally its size is a multiple of the page size. [38] A page, also known as a memory page or virtual page, is a contiguous block of virtual memory with a defined length that is described by a single entry in the page table. In a virtual memory operating system, it is the smallest unit of data for memory management. [37]

CPU usage The unit that does most of the processing in a computer is called the central processing unit (CPU) or the processor. It acts as a powerful calculator, processing all instructions received from software running on the PC and other hardware components. [32] The CPU usage information are collected from /proc/stat [2].

- **Various kernel / system statistics:** The kernel manages the computer and hardware operations and is therefore the central component of an operating system. In general, it manages memory and CPU time operations. Kernel acts as a gateway between applications and data processing performed at hardware level using inter-process communication and system calls. Managing communication between software (e.g. user-level applications) and hardware (CPU and disk storage) is the major goal of the kernel. [36]

– **Per Core Metric:** The per core metrics are listed as ranges over 0-47 cores. In this project, the per core metrics were disregarded. The reason behind it is that on the HPC system where the data was obtained, the following applies: a node has 48 CPU cores, and metrics have been recorded for each of these nodes (e.g., per_core_user0_procstat, per_core_user1_procstat, ..., per_core_user47_procstat). Besides that, there is a metric (e.g., user procstat) which represents all 48 cores merged into a single node; in other words, the CPU usage of all cores is practically included in this combined metric for the entire node. Therefore, the metrics containing the name 'per_core' can be omitted.

3.5 Definition and Description of Metric Groups

In order to investigate the frequencies of HPC Monitoring Data, the time series arrays must be analyzed in the beginning. Looking at the graphical representation of the data in a plot, it is evident that there are different types of plots. For that reason, the metrics can be classified into different groups.

3.5.1 Zero Constant and Nonzero Constant Metric Group

The initial step is to identify very general groups of metrics. In the graphical representation in some figures of the time series arrays, horizontal graphs can be observed. These can be summarized to a group of constant metrics. In some of these time series metrics a constant value of 0 occurs. Therefore, they can be assigned to the zero constant metric group. The remaining ones, which always have the same number in the time series array, are assigned to the nonzero constant metric group. These two groups can be extracted relatively easily with if statements. The remaining time series arrays potentially have a waveform. One sample plot for each group is shown in the Figure 3.5 and Figure 3.6.

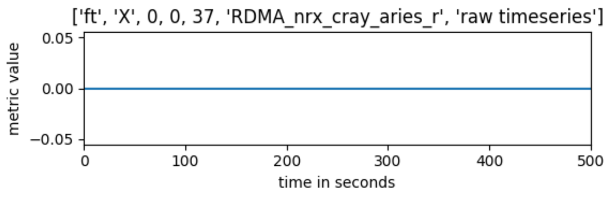


Figure 3.5: Metric 37 - Example Plot for Zero Constant

3.5.2 Increasing and Decreasing Monotonic Metric Group

Monotonic characteristics can be discovered in some of the time series arrays. The values either consistently increase and never decrease or decrease and never increase over time. Therefore, they could be categorized into two subgroups:

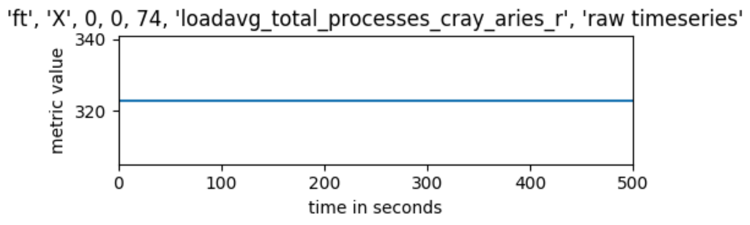


Figure 3.6: Metric 74 - Example Plot for Nonzero Constant

increasing monotonic and decreasing monotonic. By determining whether the next value is greater or equal to the current value, the increasing monotonic group can be recognized. In the opposite case, if the next value is smaller or equal to the current value, the array corresponds to the decreasing monotonic group. In Figure 3.7 and Figure 3.8, one sample plot for each group is provided.

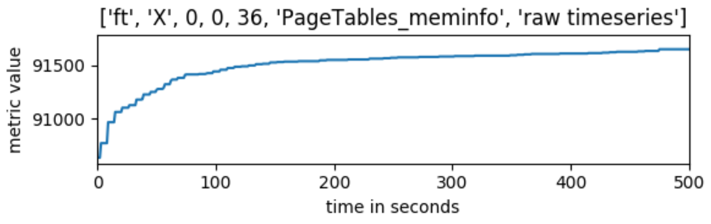


Figure 3.7: Metric 36 - Example Plot for Increasing Monotonic

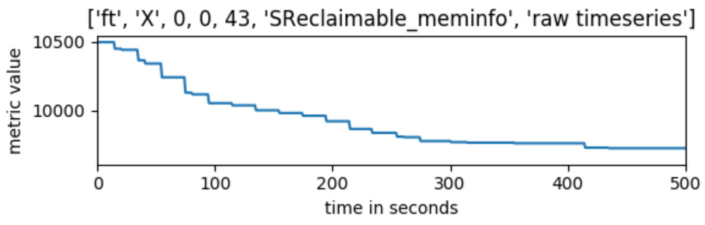


Figure 3.8: Metric 43 - Example Plot for Decreasing Monotonic

3.5.3 Waveform Metric Group

The last group needed to investigate the frequency characteristics of the metrics is to determine the group of actual waveform metrics. In some cases, a graph with several amplitudes can be seen through the frequency representation of a time series array which is calculated using the Spectrogram (see more in Section

3.1 and the Fast Fourier Transform (FFT) 3.1). According to the explanation of the Fast Fourier Transform (FFT), the time series array is assumed to have a waveform in the time domain representation, if the frequency representation contains at least one frequency peak that is not equal to 1. The Figure 3.9 represents a sample plot for a time series array in the time domain. The frequency representation of the corresponding time series array is shown in the Figure 3.10.

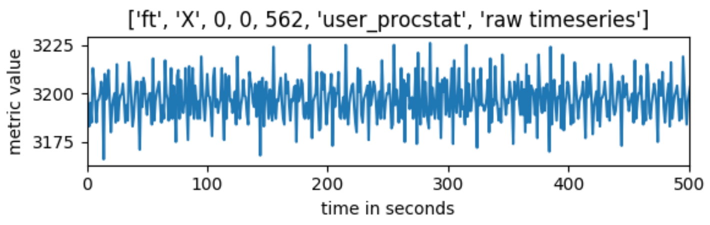


Figure 3.9: Metric 562 - Example Plot for Waveform in Time Domain

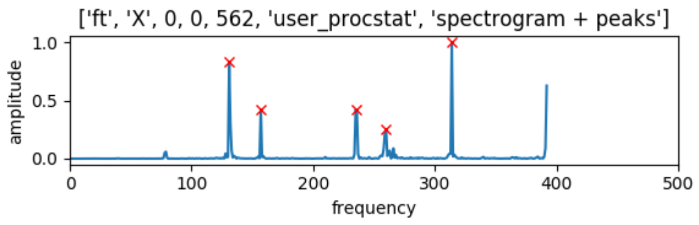


Figure 3.10: Metric 562 - Example Plot for Waveform in Frequency Domain

3.5.4 Remaining or Other Metrics

There may be a few remaining metrics that seem to have an unusual time domain plot and therefore belong to the category of other metrics. Here are the time series arrays that have no peaks in the frequency plot or have exactly one frequency peak at 1, which is the reason why these metrics cannot be assigned to the other metric groups. As an example, a plot for other metrics is shown in the Figure 3.11.

3.6 Metrics with Waveform Characteristics

For the purpose of investigating frequency in monitoring data from HPC applications and benchmarks, the metrics requirement is to have waveform characteristics.

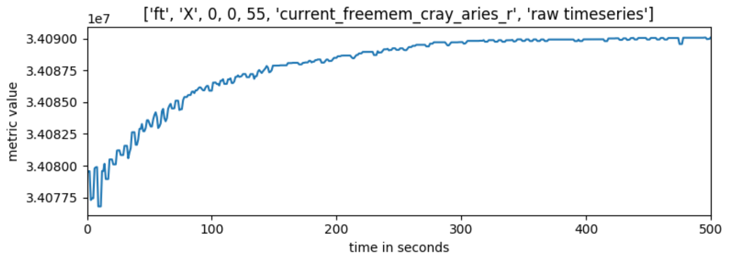


Figure 3.11: Metric 55 - Example Plot for Other Metrics

3.6.1 First Approach: Regular vs. Irregular Waveforms

The first approach was to look for waveform metrics that were regular. With the help of the spectrogram, it is possible to extract an array of frequency peaks from a time series array of a metric.

If the array of frequency peaks has only a few entries, the time series array can be considered as a regular waveform. Thus, all time series array are called regular if they have less than four frequency components. The network metric 0 with the four parameters application 'ft', input 'X', node ID 0 and run ID 0 is an example of a regular waveform (Figure 3.12). In the frequency representation, it can be seen that the graph has three frequency peaks (Figure 3.13).

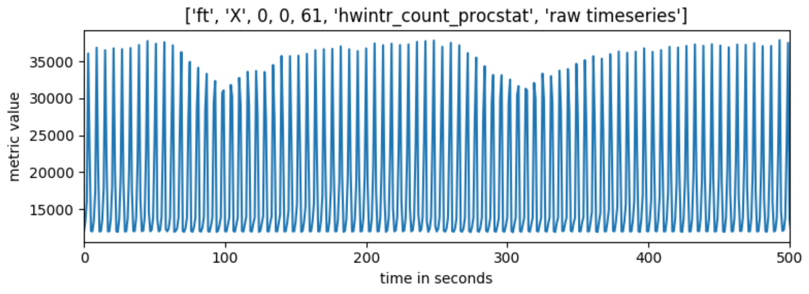


Figure 3.12: Metric 61 - Example Plot for Regular Waveform (Time Domain)

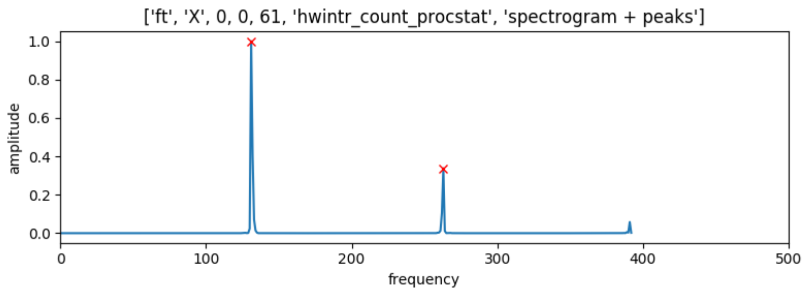


Figure 3.13: Metric 61 - Example Plot for Regular Waveform (Frequency Domain)

However, if the array involves multiple entries, the time series array will have many different overlayed frequency components and will therefore be irregular. An example for an irregular waveform of a time series array is the metric 562 with the parameters application 'ft', input 'X', node ID 0 and run ID (Figure 3.14). The graph has more than three frequency components, which can be seen in the frequency representation (Figure 3.15).

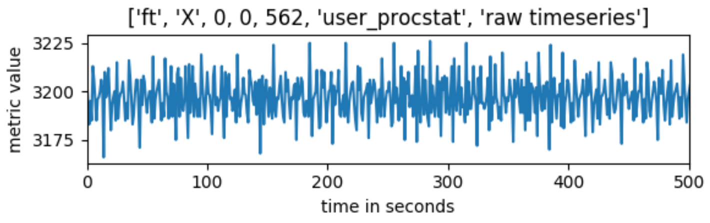


Figure 3.14: Metric 562 - Example Plot for Irregular Waveform (Time Domain)

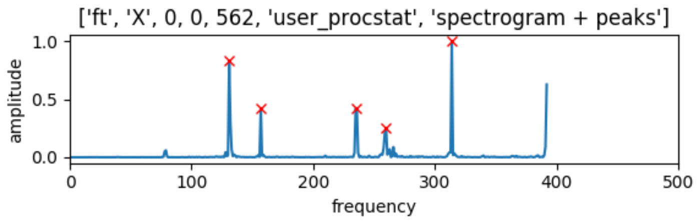


Figure 3.15: Metric 562 - Example Plot for Irregular Waveform (Frequency Domain)

The frequency of HPC monitoring data should be easier to investigate using regular waveform metrics. But unfortunately, it was discovered that depending on the parameter, the network metric 0 has more than 3 entries for the array of frequency peaks, and that this metric is actually a good metric for the analysis of frequencies. The problem is that different applications may produce different waveform plots for the same metric, of which some are regular while others are irregular. Therefore this approach is not useless, but it cannot be used for sorting out metrics with waveform. Instead, this approach is used to compare the applications.

3.6.2 Second Approach: Majority Waveform Metrics

The second approach used was finding metrics consisting of the most waveform characteristic time series arrays because these are the metrics that are considered to be most suitable for investigating the frequency of HPC monitoring data. The metrics which are selected for this purpose are those that consist of more than 95 percent waveform time series.

3.7 Code Implementation

In order to generate the statistics, a code in the Python environment was created. A file called metricGroups.py, that gives an output of 4 different tables as CSV files and also the plots of the metrics, was written. At the beginning, arrays for all applications, input sizes and metrics were initialised according to the data from the Taxonomist data set (see Section 3.3). With the aid of for-loops, the program is executed over all arrays as well as over every node ID and run ID.

The Python CSV Module was used to create CSV files. The plots were displayed using matplotlib.pyplot. As mentioned in Section 3.2, spectrogram from the SciPy module was used to visualise the spectrogram.

To extract the metric groups, the time series arrays were separated using if statements with the respective conditions. For each group, a default dict was created from the collection module to count the number of metric groups occurring in the metric, which was later used to calculate different values.

The following code has been implemented for extracting the metric groups.

```
# zero constant metrics
if all(x == 0 for x in timeseries):
    zero_dict[metric_id] += 1

# non-zero constant metrics
elif all(x == timeseries[120] for x in timeseries):
    constant_dict[metric_id] += 1

# increasing monotonic metrics
elif all(x <= y for x, y in zip(timeseries, timeseries[1:])):
    increasing_monotonic_dict[metric_id] += 1

# decreasing monotonic metrics
elif all(x >= y for x, y in zip(timeseries, timeseries[1:])):
    decreasing_monotonic_dict[metric_id] += 1

# wave form metrics
elif len(peaks) > 1 or (len(peaks) == 1 and peaks[0] != 1):
    waveform_dict[metric_id] += 1

# other metrics
else:
    other_dict[metric_id] += 1
```

In addition, a Python file called WAVfileConverter.py was created for the conversion of a WAV file. With the help of the Python file, a WAV file can be displayed visually by first converting it into an array. This array is then displayed in time series and frequency representations. The purpose of this programme is to compare the system metrics with real sound.

Chapter 4

Results

4.1 Overview of Metric Groups

The following table contains percentage of each metric group over all metrics. The percentages reported are calculated in terms of the total number of time domain arrays produced for each metric. For each metric, a total of 4728 time domain arrays can be constructed by combining all different possibilities of applications, inputs, node IDs and run IDs parameters. The metrics are listed with leading metric ID, as it is defined in the Taxonomist data. Metrics might be grouped disregarding the metric ID ordering for readability. The table was divided up into multiple tables based on the subgroup of the conceptual group for better illustration in the document (Table 4.1 - 4.15). In addition, the first column 'Metric ID' was abbreviated as 'ID'.

ID	Metric Name prefix1: AR_NIC_NETMON_ORB_EVENT_CNTR_REQ prefix2: AR_NIC_RSPMON_PARB_EVENT_CNTR	Zero constant %	Nonzero constant %	Increasing monotonic %	Decreasing monotonic %	Wave %	Other %
0	prefix1_FLITS_metric_set_nic	0%	0%	0%	0%	95.73%	4.27%
1	prefix1_PKTS_metric_set_nic	0%	0%	0%	0%	94.97%	5.03%
2	prefix1_STALLED_metric_set_nic	0.21%	0%	0%	0%	94.5%	5.29%
3	prefix2_AMO_BLOCKED_metric_set_nic	0%	0%	0%	0%	97.65%	2.35%
4	prefix2_AMO_FLITS_metric_set_nic	0%	0%	0%	0%	96.11%	3.89%
5	prefix2_AMO_PKTS_metric_set_nic	0%	0%	0%	0%	96.17%	3.83%
6	prefix2_BTE_RD_BLOCKED_metric_set_nic	76.35%	0%	0%	0%	23.08%	0.57%
7	prefix2_BTE_RD_FLITS_metric_set_nic	18.44%	0%	0.04%	0.04%	77.35%	4.12%
8	prefix2_BTE_RD_PKTS_metric_set_nic	18.44%	0%	0.04%	0.04%	77.35%	4.12%
9	prefix2_IOMMU_BLOCKED_metric_set_nic	0%	0%	0%	0%	97.44%	2.56%
10	prefix2_IOMMU_FLITS_metric_set_nic	0%	0%	0%	0%	97.46%	2.54%
11	prefix2_IOMMU_PKTS_metric_set_nic	0%	0%	0%	0%	97.46%	2.54%
12	prefix2_PI_FLITS_metric_set_nic	0%	0%	0%	0%	96.21%	3.79%
13	prefix2_PI_PKTS_metric_set_nic	0%	0%	0%	0%	96.7%	3.3%
14	prefix2_PI_STALLED_metric_set_nic	0%	0%	0%	0%	96.17%	3.83%
15	prefix2_WC_BLOCKED_metric_set_nic	17.77%	0%	0%	0%	79.93%	2.31%
16	prefix2_WC_FLITS_metric_set_nic	17.77%	0%	0%	0%	79.65%	2.58%
17	prefix2_WC_PKTS_metric_set_nic	17.77%	0%	0%	0%	79.65%	2.58%

Table 4.1: Network - NIC (Network Interface Controller)

ID	Metric Name	Zero constant %	Nonzero constant %	Increasing monotonic %	Decreasing monotonic %	Wave %	Other %
37	RDMA_nrx_cray_aries_r	100.0%	0%	0%	0%	0%	0%
38	RDMA_rx_bytes_cray_aries_r	100.0%	0%	0%	0%	0%	0%

Table 4.2: Network - RDMA (Remote Direct Memmory Access)

ID	Metric Name	Zero constant %	Nonzero constant %	Increasing monotonic %	Decreasing monotonic %	Wave %	Other %
39	SMSG_nrx_cray_aries_r	0%	0%	0%	0%	100.0%	0%
40	SMSG_ntx_cray_aries_r	0%	0%	0%	0%	100.0%	0%
41	SMSG_rx_bytes_cray_aries_r	91.9%	0%	0.04%	0.02%	7.23%	0.8%
42	SMSG_tx_bytes_cray_aries_r	91.9%	0%	0.04%	0.02%	7.23%	0.8%

Table 4.3: Network - SMSG (Short Message Service Gateway)

ID	Metric Name	Zero constant %	Nonzero constant %	Increasing monotonic %	Decreasing monotonic %	Wave %	Other %
64	ipogif0_rx_bytes_cray_aries_r	21.19%	0%	0.13%	0.06%	73.63%	4.99%
65	ipogif0_tx_bytes_cray_aries_r	21.09%	0%	0.13%	0.08%	73.5%	5.2%

Table 4.4: Network - IPoGIF (IP over Generic Fabric Interface)

ID	Metric Name	Zero constant %	Nonzero constant %	Increasing monotonic %	Decreasing monotonic %	Wave %	Other %
55	current_freemem_cray_aries_r	0%	0%	0%	0%	15.99%	84.01%
56	energy(J)_cray_aries_r	0%	0%	0%	0%	99.6%	0.4%
57	freshness_cray_aries_r	0%	3.0%	0%	0%	96.21%	0.78%
550	power(W)_cray_aries_r	0%	0.27%	0.06%	0%	93.23%	6.43%

Table 4.5: Network - Other Metrics Including Energy and Power

ID	Metric Name	Zero constant %	Nonzero constant %	Increasing monotonic %	Decreasing monotonic %	Wave %	Other %
71	loadavg_5min(x100)_cray_aries_r	0%	0.06%	91.62%	3.76%	0.21%	4.34%
72	loadavg_latest(x100)_cray_aries_r	0%	0%	45.85%	3.34%	1.9%	48.9%
73	loadavg_running_processes_cray_aries_r	0%	37.61%	0.08%	0.15%	59.31%	2.86%
74	loadavg_total_processes_cray_aries_r	0%	100.0%	0%	0%	0%	0%

Table 4.6: Network - LOADAVG

ID	Metric Name	Zero constant %	Nonzero constant %	Increasing monotonic %	Decreasing monotonic %	Wave %	Other %
18	Active(anon)_meminfo	0%	43.02%	48.96%	0%	7.7%	0.32%
19	Active(file)_meminfo	0%	100.0%	0%	0%	0%	0%
20	Active_meminfo	0%	43.02%	48.96%	0%	7.7%	0.32%
29	Inactive(anon)_meminfo	0%	72.14%	27.81%	0.04%	0%	0%
30	Inactive(file)_meminfo	0%	100.0%	0%	0%	0%	0%
31	Inactive_meminfo	0%	72.14%	27.81%	0.04%	0%	0%

Table 4.7: Memory (MEMINFO) - Active/Inactive

ID	Metric Name	Zero constant %	Nonzero constant %	Increasing monotonic %	Decreasing monotonic %	Wave %	Other %
25	HugePages_Free_meminfo	100.0%	0%	0%	0%	0%	0%
26	HugePages_Rsvd_meminfo	100.0%	0%	0%	0%	0%	0%
27	HugePages_Surp_meminfo	0%	100.0%	0%	0%	0%	0%
28	HugePages_Total_meminfo	0%	100.0%	0%	0%	0%	0%

Table 4.8: Memory (MEMINFO) - Huge Pages (Memory Pages Greater Than Default)

ID	Metric Name	Zero constant %	Nonzero constant %	Increasing monotonic %	Decreasing monotonic %	Wave %	Other %
21	AnonPages_meminfo	0%	47.48%	38.37%	0%	7.83%	6.32%
22	Cached_meminfo	0%	69.31%	30.69%	0%	0%	0%
23	CommitLimit_meminfo	0%	100.0%	0%	0%	0%	0%
24	Committed_AS_meminfo	0%	64.57%	35.43%	0%	0%	0%
32	KernelStack_meminfo	0%	100.0%	0%	0%	0%	0%
33	Mapped_meminfo	0%	49.6%	50.4%	0%	0%	0%
34	MemFree_meminfo	0%	0%	0%	0%	16.16%	83.84%
35	Mlocked_meminfo	0%	100.0%	0%	0%	0%	0%
36	PageTables_meminfo	0%	0%	100.0%	0%	0%	0%
43	SReclaimable_meminfo	0%	0%	0%	100.0%	0%	0%
44	SUnreclaim_meminfo	0%	0%	0%	9.56%	7.8%	82.64%
45	Shmem_meminfo	0%	69.31%	30.69%	0%	0%	0%
46	Slab_meminfo	0%	0%	0%	9.56%	7.59%	82.85%
47	Unevictable_meminfo	0%	100.0%	0%	0%	0%	0%
48	VmallocChunk_meminfo	0%	100.0%	0%	0%	0%	0%
49	VmallocUsed_meminfo	0%	100.0%	0%	0%	0%	0%

Table 4.9: Memory (MEMINFO) - Other Metrics Including Cache and Shared Memory

ID	Metric Name	Zero constant %	Nonzero constant %	Increasing monotonic %	Decreasing monotonic %	Wave %	Other %
50	allocstall_vmstat	100.0%	0%	0%	0%	0%	0%
51	compact_blocks_moved_vmstat	100.0%	0%	0%	0%	0%	0%
52	compact_pagemigrate_failed_vmstat	100.0%	0%	0%	0%	0%	0%
53	compact_pages_moved_vmstat	100.0%	0%	0%	0%	0%	0%
60	htlb_buddy_alloc_success_vmstat	100.0%	0%	0%	0%	0%	0%

Table 4.10: Memory (VMSTAT) - Various Virtual Memory Statistics

ID	Metric Name	Zero constant %	Nonzero constant %	Increasing monotonic %	Decreasing monotonic %	Wave %	Other %
67	kswapd_high_wmark_hit_quickly_vmstat	100.0%	0%	0%	0%	0%	0%
68	kswapd_inodesteal_vmstat	100.0%	0%	0%	0%	0%	0%
69	kswapd_low_wmark_hit_quickly_vmstat	100.0%	0%	0%	0%	0%	0%
70	kswapd_steal_vmstat	100.0%	0%	0%	0%	0%	0%

Table 4.11: Memory (VMSTAT) - kswapd (Kernal Swap Daemon) Memory Management

ID	Metric Name	Zero constant %	Nonzero constant %	Increasing monotonic %	Decreasing monotonic %	Wave %	Other %
76	nr_active_anon_vmstat	0%	43.02%	49.03%	0%	7.7%	0.25%
77	nr_active_file_vmstat	0%	100.0%	0%	0%	0%	0%
78	nr_anon_pages_vmstat	0%	47.44%	38.43%	0%	7.85%	6.28%
79	nr_dirty_background_threshold_vmstat	0%	0%	0%	0%	16.16%	83.84%
80	nr_dirty_threshold_vmstat	0%	0%	0%	0%	16.14%	83.86%
81	nr_file_pages_vmstat	0%	69.31%	30.69%	0%	0%	0%
82	nr_free_pages_vmstat	0%	0%	0%	0%	16.14%	83.86%
83	nr_inactive_anon_vmstat	0%	72.14%	27.81%	0.04%	0%	0%
84	nr_inactive_file_vmstat	0%	100.0%	0%	0%	0%	0%
85	nr_isolated_anon_vmstat	100.0%	0%	0%	0%	0%	0%
86	nr_isolated_file_vmstat	100.0%	0%	0%	0%	0%	0%
87	nr_kernel_stack_vmstat	0%	100.0%	0%	0%	0%	0%
88	nr_mapped_vmstat	0%	49.6%	50.4%	0%	0%	0%
89	nr_mlock_vmstat	0%	100.0%	0%	0%	0%	0%
90	nr_page_table_pages_vmstat	0%	0%	100.0%	0%	0%	0%
91	nr_shmem_vmstat	0%	69.31%	30.69%	0%	0%	0%
92	nr_slab_reclaimable_vmstat	0%	0%	0%	100.0%	0%	0%
93	nr_slab_unreclaimable_vmstat	0%	0%	0%	9.56%	7.8%	82.64%
94	nr_unevictable_vmstat	0%	100.0%	0%	0%	0%	0%

Table 4.12: Memory (VMSTAT) - NR Metrics

ID	Metric Name	Zero constant %	Nonzero constant %	Increasing monotonic %	Decreasing monotonic %	Wave %	Other %
95	numa_foreign_vmstat	100.0%	0%	0%	0%	0%	0%
96	numa_hit_vmstat	0%	0%	0%	0%	90.99%	9.01%
97	numa_interleave_vmstat	100.0%	0%	0%	0%	0%	0%
98	numa_local_vmstat	0%	0%	0%	0%	90.99%	9.01%
99	numa_miss_vmstat	100.0%	0%	0%	0%	0%	0%
100	numa_other_vmstat	100.0%	0%	0%	0%	0%	0%

Table 4.13: Memory (VMSTAT) - NUMA (Non-Uniform Memory Access)

ID	Metric Name	Zero constant %	Nonzero constant %	Increasing monotonic %	Decreasing monotonic %	Wave %	Other %
101	pageoutrun_vmstat	100.0%	0%	0%	0%	0%	0%
534	pgactivate_vmstat	99.96%	0%	0%	0%	0.04%	0%
535	pgalloc_dma32_vmstat	98.1%	0%	0%	0%	1.02%	0.89%
536	pgalloc_normal_vmstat	0%	0%	0%	0%	95.24%	4.76%
537	pgdeactivate_vmstat	100.0%	0%	0%	0%	0%	0%
538	pgfault_vmstat	0%	0%	0%	0%	78.26%	21.74%
539	pgfree_vmstat	0%	0%	0%	0%	98.98%	1.02%
540	pgmajfault_vmstat	100.0%	0%	0%	0%	0%	0%
541	pgrefill_dma32_vmstat	100.0%	0%	0%	0%	0%	0%
542	pgrefill_normal_vmstat	100.0%	0%	0%	0%	0%	0%
543	pgrotated_vmstat	100.0%	0%	0%	0%	0%	0%
544	pgscan_direct_dma32_vmstat	100.0%	0%	0%	0%	0%	0%
545	pgscan_direct_normal_vmstat	100.0%	0%	0%	0%	0%	0%
546	pgscan_kswapd_dma32_vmstat	100.0%	0%	0%	0%	0%	0%
547	pgscan_kswapd_normal_vmstat	100.0%	0%	0%	0%	0%	0%
548	pgsteal_dma32_vmstat	100.0%	0%	0%	0%	0%	0%
549	pgsteal_normal_vmstat	100.0%	0%	0%	0%	0%	0%
554	slabs_scanned_vmstat	100.0%	0%	0%	0%	0%	0%
558	unevictable_pgs_culled_vmstat	100.0%	0%	0%	0%	0%	0%
559	unevictable_pgs_mlocked_vmstat	100.0%	0%	0%	0%	0%	0%
560	unevictable_pgs_munlocked_vmstat	100.0%	0%	0%	0%	0%	0%
561	unevictable_pgs_rescued_vmstat	100.0%	0%	0%	0%	0%	0%

Table 4.14: Memory (VMSTAT) - Other Metrics Including Slab and Page

ID	Metric Name	Zero constant %	Nonzero constant %	Increasing monotonic %	Decreasing monotonic %	Wave %	Other %
54	context_switches_procstat	0%	0%	0%	0%	91.05%	8.95%
58	guest_nice_procstat	100.0%	0%	0%	0%	0%	0%
59	guest_procstat	100.0%	0%	0%	0%	0%	0%
61	hwintr_count_procstat	0%	0%	0%	0%	97.38%	2.62%
62	idle_procstat	60.93%	0%	0%	0.02%	38.87%	0.17%
63	iowait_procstat	99.85%	0%	0%	0%	0.02%	0.13%
66	irq_procstat	100.0%	0%	0%	0%	0%	0%
75	nice_procstat	100.0%	0%	0%	0%	0%	0%
551	processes_procstat	100.0%	0%	0%	0%	0%	0%
552	procs_blocked_procstat	100.0%	0%	0%	0%	0%	0%
553	procs_running_procstat	0%	22.91%	0%	0.06%	74.64%	2.39%
555	softirq_count_procstat	0%	0%	0%	0%	99.89%	0.11%
556	softirq_procstat	62.99%	0%	0.02%	0.06%	35.6%	1.33%
557	sys_procstat	0%	0%	0%	0%	99.79%	0.21%
562	user_procstat	0%	0%	0%	0%	100.0%	0%

Table 4.15: CPU Usage (PROCSTAT) - Various Kernel / System Statistics

In this table there should be 563 entries, since there is one entry for each metric. But as already mentioned, the per core metrics are omitted, therefore it now has 131 metrics, as well as 131 entries. By multiplying 4726 by 563, we can calculate a total of 2,661,864 time series arrays of all metrics, including per core metrics, together. Out of these a total of 69.3% time series arrays are zero constant metrics and 4.52% are non-zero constant metrics. It also has 1.61% monotonic metrics that are increasing and 0.42% monotonic metrics that are decreasing. Furthermore, 22.53% time series arrays have a waveform. The remaining time series arrays were classified to the other group.

Total Time Series Arrays	2661864
Zero constant %	69.3%
Nonzero constant %	4.52%
Increasing monotonic %	1.61%
Decreasing monotonic %	0.42%
Wave %	22.53%
Other %	1.62%

Table 4.16: Percentage of Each Group over all Metrics

According to the percentage of metric groups calculated over all time series arrays, more than half of the data cannot be used for a frequency based investigation because the majority of the metrics are zero constants and therefore have no frequency.

Caution: The Table 4.16 is the only table where per_core metrics were not excluded. This was done to gain an overview of the whole dataset of the Taxonomist. Most of the zero constant metrics come from the per_core metrics which are a part of the PROCSTAT group. They are excluded from other tables.

4.1.1 Percentage of Metric Groups for Each Conceptual Group

The calculated percentages of the metric groups in each conceptual group are presented in this section. The percentages are based on the number of time series arrays in each Conceptual Group, which is listed in the last column.

Metric Group	Network	MEMINFO	VMSTAT	CPU usage	Total Time Series Arrays
Zero constant	11.82%	3.99%	67.76%	16.43%	237093
Nonzero constant	5.54%	60.13%	33.43%	0.9%	120341
Increasing Monotonic	15.27%	48.56%	36.17%	0.0%	42753
Decreasing Monotonic	3.18%	50.41%	46.35%	0.06%	11181
Waveform	59.31%	1.56%	14.98%	18.1%	166471
Others	24.93%	29.18%	44.08%	1.81%	41529

Table 4.17: Conceptual Groups

By looking at the Table 4.17, the most waveform metrics come from the network group and the most constant metrics, rounded up to 70%, come from the memory group.

4.1.2 Metric Characteristics of Application Execution

To compare the applications, the percentages of the metric groups in the application were calculated (Table 4.18). In addition, the approach of dividing regular and irregular waveforms was used here. As mentioned, the time series arrays that have less than four peaks in the frequency representation are classified as regular. In the opposite case, those with more than three peaks are classified as irregular.

Application	Zero constant %	Nonzero constant %	Increasing monotonic %	Decreasing monotonic %	Regular Waveform %	Irregular Waveform %	Other %	Total Time Series Arrays
FT	38.25%	21.72%	5.08%	1.56%	17.09%	9.43%	6.88%	47160
MG	37.04%	20.96%	5.63%	2.04%	4.08%	23.45%	6.8%	47160
SP	37.28%	24.08%	2.62%	2.37%	9.51%	17.5%	6.65%	47160
LU	38.61%	18.4%	8.14%	1.54%	9.27%	18.93%	5.12%	47160
BT	37.38%	24.13%	2.78%	2.41%	11.32%	15.84%	6.13%	47160
CG	37.4%	24.22%	2.64%	2.32%	9.08%	18.09%	6.25%	47160
miniGhost	38.8%	21.92%	4.53%	1.55%	10.02%	16.14%	7.04%	72312
miniAMR	37.55%	15.11%	11.19%	1.58%	2.32%	23.32%	8.92%	72312
miniMD	39.28%	16.14%	10.14%	1.69%	8.74%	15.84%	8.18%	72312
Kripke	39.93%	13.86%	12.96%	1.66%	2.97%	20.39%	8.24%	72312
CoMD	38.26%	18.88%	4.25%	1.55%	15.13%	21.33%	0.6%	47160

Table 4.18: Percentage of Each Metric Group From Each Application

Most HPC applications produce irregular waveform in their system metrics. FT is the only application with more regular waveform metrics than irregular waveform metrics. The application CoMD generates more system metrics with waveform with about 35% compared to all other applications, which have on average about 25% waveform system metrics.

4.2 Metrics with Majority Waveform

All the majority waveform metrics are listed in a table. This table was also divided up into multiple tables based on the subgroup of the conceptual groups for better illustration in the document (Table 4.19 - 4.23). In addition, the first column 'Metric ID' was abbreviated as 'ID'.

ID	Metric Name prefix1:AR_NIC_NETMON_ORB_EVENT_CNTR prefix2:AR_NIC_RSPMON_PARB_EVENT_CNTR	Zero constant %	Nonzero constant %	Increasing monotonic %	Decreasing monotonic %	Wave %	Other %
0	prefix1_REQ_FLITS_metric_set_nic	0%	0%	0%	0%	95.73%	4.27%
3	prefix2_AMO_BLOCKED_metric_set_nic	0%	0%	0%	0%	97.65%	2.35%
4	prefix2_AMO_FLITS_metric_set_nic	0%	0%	0%	0%	96.11%	3.89%
5	prefix2_AMO_PKTS_metric_set_nic	0%	0%	0%	0%	96.17%	3.83%
9	prefix2_IOMMU_BLOCKED_metric_set_nic	0%	0%	0%	0%	97.44%	2.56%
10	prefix2_IOMMU_FLITS_metric_set_nic	0%	0%	0%	0%	97.46%	2.54%
11	prefix2_IOMMU_PKTS_metric_set_nic	0%	0%	0%	0%	97.46%	2.54%
12	prefix2_PI_FLITS_metric_set_nic	0%	0%	0%	0%	96.21%	3.79%
13	prefix2_PI_PKTS_metric_set_nic	0%	0%	0%	0%	96.7%	3.3%
14	prefix2_PI_STALLED_metric_set_nic	0%	0%	0%	0%	96.17%	3.83%

Table 4.19: Majority Waveform Network - NIC (Network Interface Controller)

ID	Metric Name	Zero constant %	Nonzero constant %	Increasing monotonic %	Decreasing monotonic %	Wave %	Other %
40	SMSG_ntx_cray_aries_r	0%	0%	0%	0%	100.0%	0%

Table 4.20: Majority Waveform Network - SMSG (Short Message Service Gateway)

ID	Metric Name	Zero constant %	Nonzero constant %	Increasing monotonic %	Decreasing monotonic %	Wave %	Other %
56	energy(J)_cray_aries_r	0%	0%	0%	0%	99.6%	0.4%
57	freshness_cray_aries_r	0%	3.0%	0%	0%	96.21%	0.78%

Table 4.21: Majority Waveform Network - Other Metrics Including Energy and Power

ID	Metric Name	Zero constant %	Nonzero constant %	Increasing monotonic %	Decreasing monotonic %	Wave %	Other %
536	pgalloc_normal_vmstat	0%	0%	0%	0%	95.24%	4.76%
539	pgfree_vmstat	0%	0%	0%	0%	98.98%	1.02%

Table 4.22: Majority Waveform VMSTAT - Other Metrics Including Slab and Page

ID	Metric Name	Zero constant %	Nonzero constant %	Increasing monotonic %	Decreasing monotonic %	Wave %	Other %
61	hwintr_count_procstat	0%	0%	0%	0%	97.38%	2.62%
555	softirq_count_procstat	0%	0%	0%	0%	99.89%	0.11%
557	sys_procstat	0%	0%	0%	0%	99.79%	0.21%
562	user_procstat	0%	0%	0%	0%	100.0%	0%

Table 4.23: Majority Waveform CPU Usage - Various Kernel / System Statistics

Without considering the per core metrics, a total of 20 metrics were found which consist of more than 95% waveform. In addition, it can be observed that the majority of these metrics are network metrics.

4.2.1 Time Domain Plots of Majority Waveform

The following Figures (Figure 4.1 - 4.20) are time series plots for the selected metrics, the so-called majority waveform metrics. The time series arrays that have the parameters application 'ft', input 'X', node ID 0 and run ID 0 are the only ones that have been used for the graphical representation of these metrics, which will be compared to the real sound in the next section.

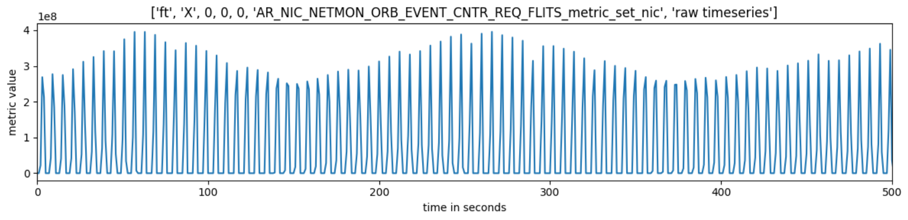


Figure 4.1: Metric 0

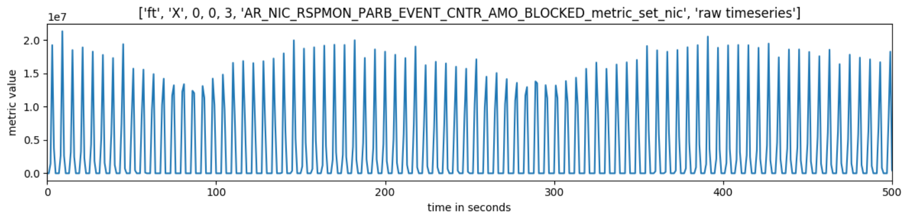


Figure 4.2: Metric 3

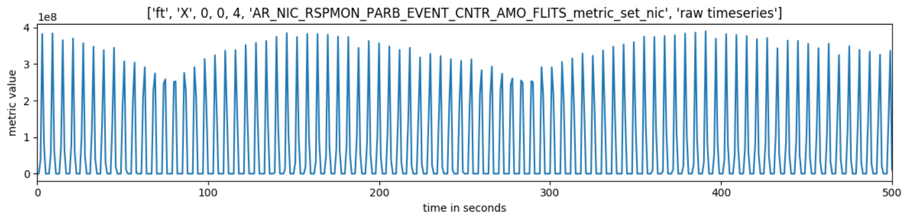


Figure 4.3: Metric 4

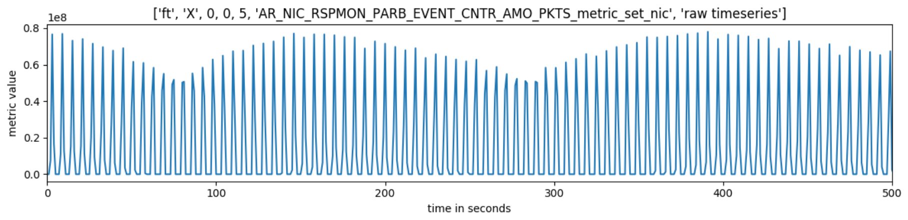


Figure 4.4: Metric 5

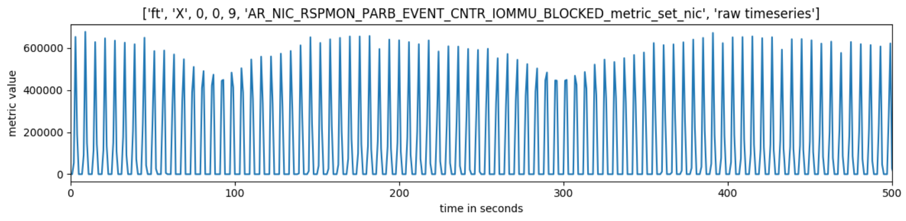


Figure 4.5: Metric 9

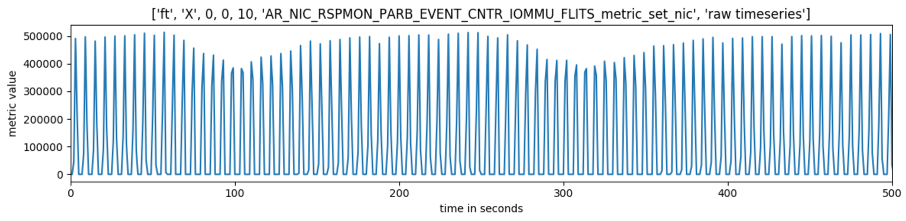


Figure 4.6: Metric 10

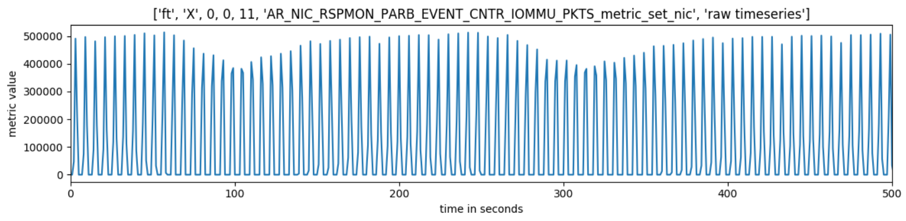


Figure 4.7: Metric 11

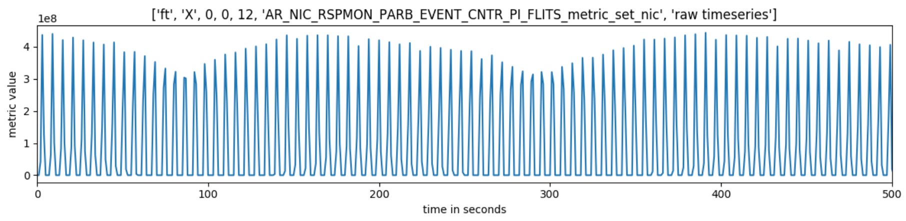


Figure 4.8: Metric 12

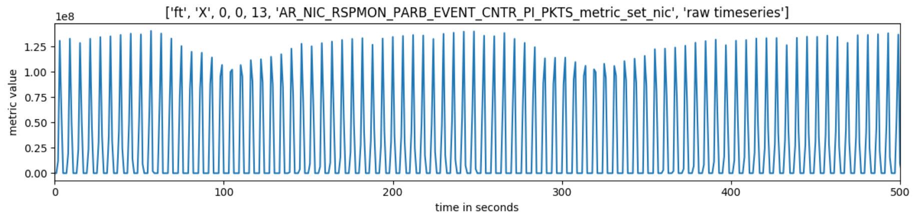


Figure 4.9: Metric 13

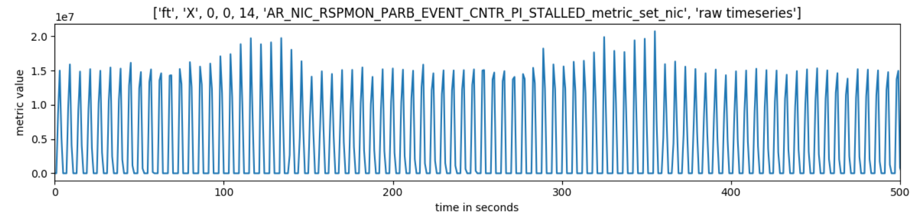


Figure 4.10: Metric 14

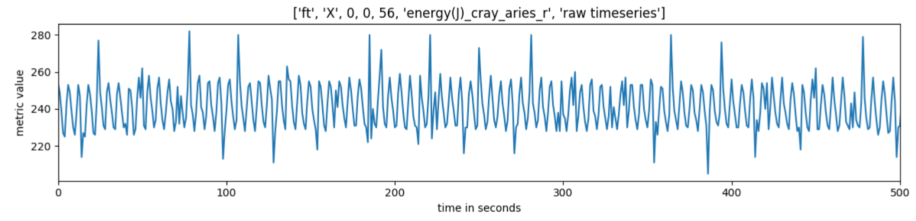


Figure 4.11: Metric 56

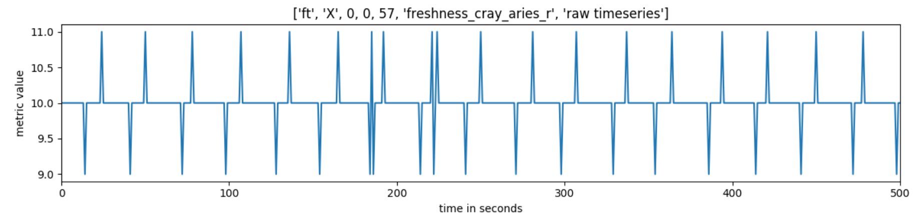


Figure 4.12: Metric 57

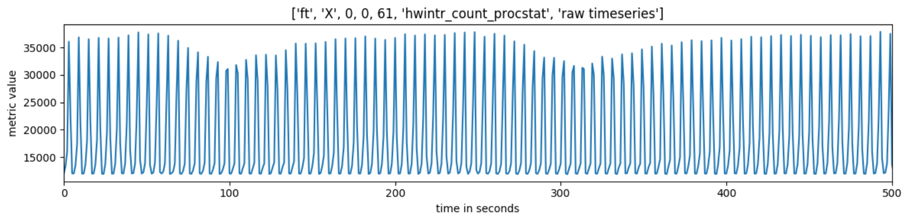


Figure 4.13: Metric 61

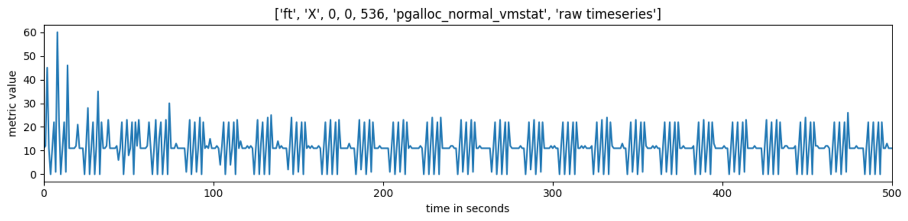


Figure 4.14: Metric 536

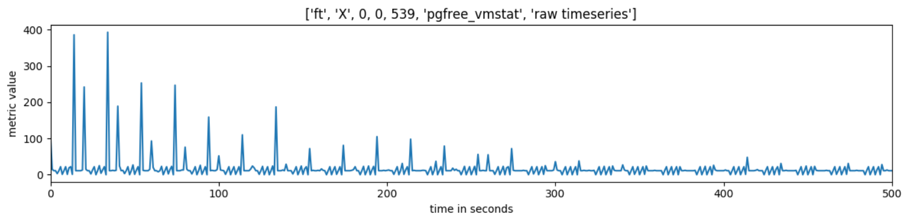


Figure 4.15: Metric 539

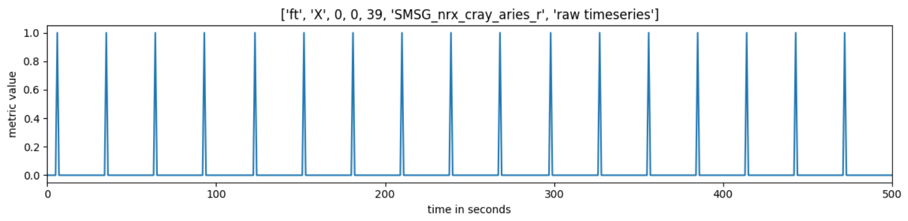


Figure 4.16: Metric 39

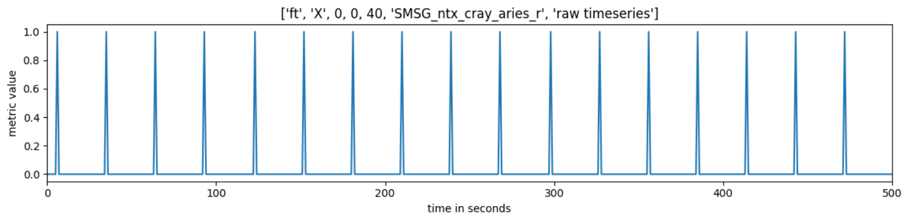


Figure 4.17: Metric 40

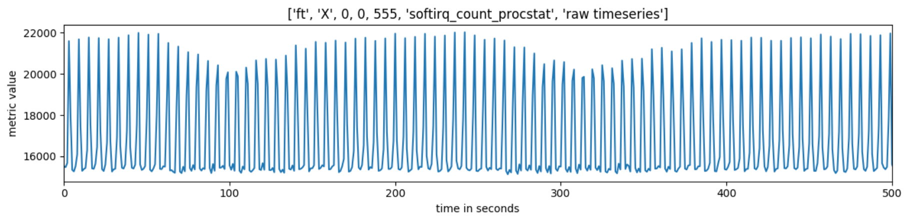


Figure 4.18: Metric 555

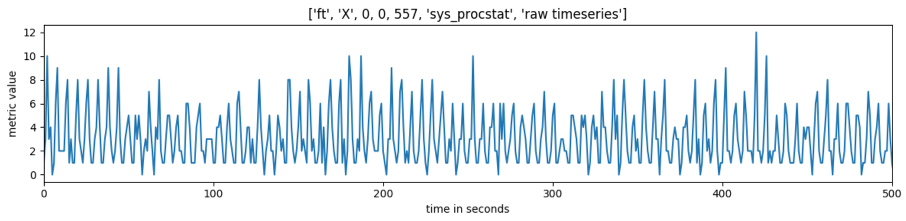


Figure 4.19: Metric 557

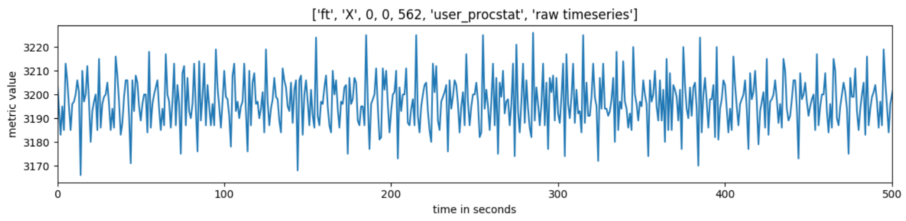


Figure 4.20: Metric 562

4.2.2 Similarity of HPC Waveforms to Actual Sound

This section contains the results gained from the analysis of the similarity between waveform systems metrics and sound, speech and music.

In order to compare the HPC monitoring data with real sound, the first step was to create a method that receives a WAV file as input parameter and converts it to an array. This array is visualised in time domain as well as in frequency representation with the frequency peaks. The idea was to compare the frequency peaks of the sound with the peaks of the majority waveform metrics. The images below represent a conversion example of a piano pop WAV file [10] into time domain(Figure 4.21) and frequency domain (Figure 4.22) plots.

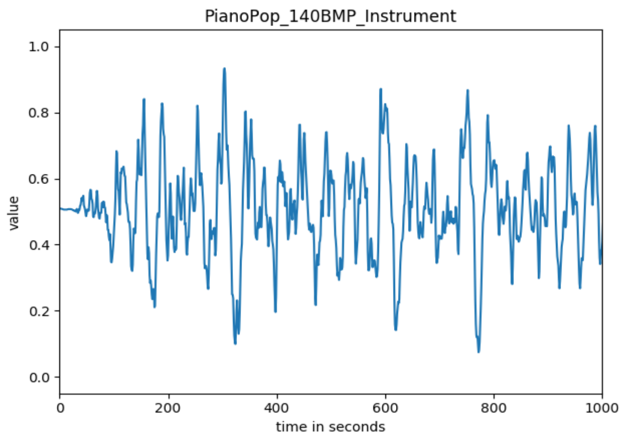


Figure 4.21: Piano Pop (Time Domain)

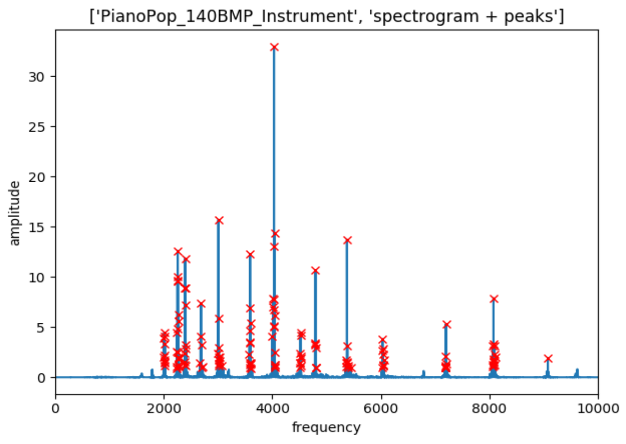
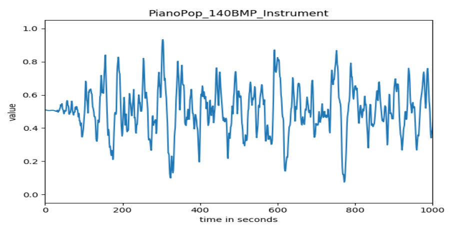


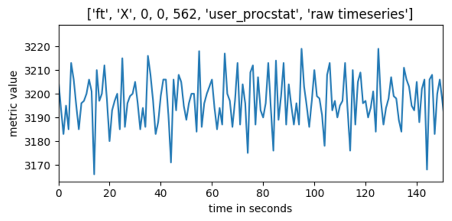
Figure 4.22: Frequency Domain Representation

However, there was a problem: the WAV file contains more samples per second than our time series data, whereas the sampling frequency of our dataset was 1/s, while normal sound sampling is done at multiple 1000/s. Without a large number of data transformations, the comparison between the frequency peak values of the arrays makes no sense.

But if the sampling frequency is disregarded, a certain similarity with the waveform of the metric user_procstat becomes visible (Figure 4.23).



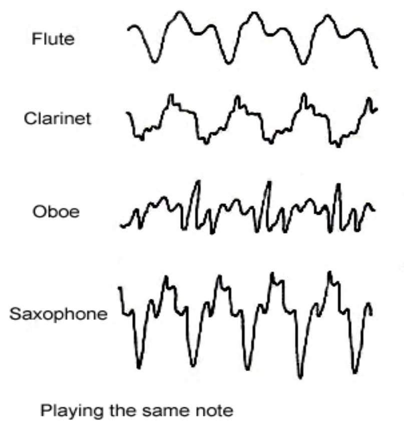
(a) Piano Pop



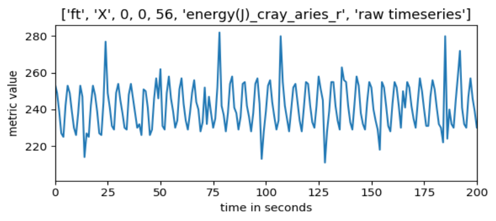
(b) Metric 562

Figure 4.23: Visual Similarity of HPC Waveform to Actual Sound

Hence, a new approach was to find the sound waves produced by different instruments and compare these views with the plots of the 20 most waveform. In the following Figure (4.24) it can be observed that there is a certain similarity between the metric energy(J)_cray_aries_r and the waveform of an oboe.



(a) Sound Waves of Instruments [15]



(b) Metric 56

Figure 4.24: Similarity of Instruments with Metric 56

Unfortunately, this was also the only similarity regarding the found sound waves of instruments.

Chapter 5

Discussion

This section summarizes the findings and insights gained through this project. Statistical properties, majority waveform, regular or irregular waveform, and sound similarity are all aspects of this work.

5.1 Statistical Properties of System Metrics

The majority of all system metrics in the dataset were found to be zero-constant. Most of the zero-constant metrics come from the per_core metrics of the PROCSTAT group. Furthermore, the memory groups (MEMINFO and VMSTAT) contribute the most to the constant system metrics category and the waveform category receives the most contributions from the network group.

Majority of Zero Constant Metrics That the majority of the data inside the dataset is zero constant, is not good news for anyone who wants to analyze the dataset, because a lot of the metrics are useless. Most of the zero constant metrics come from the per_core metrics and have to be excluded for meaningful analysis and statistical results (as done in this project).

5.2 Majority Waveform System Metrics

Most system metrics with waveform come from the network group, specifically the subgroup Network Interface Controller. But also the network subgroups SMSG, IPoGIF, and Energy and Power have high waveform shares. Other metric groups only have specific individual system metrics with waveform, but are otherwise mostly non-waveform. Examples of these are numa_hit and pgfree from VMSTAT, and user_procstat from PROCSTAT. Every major conceptual group (CPU, Memory, Network) contains at least some system metrics that have waveform. In total, 20 system metrics were found that consistently produce time series data with waveform. The time series data for these individual metrics includes over 95% waveform, while other metrics do not always have waveform.

Network Group mostly Waveform It was expected that the Network group of metrics has the most time series data with waveform, followed by the PROCSTAT group. The nature of communication inside an application is dynamic. It was also expected that the memory group has mostly constant metrics, because certain metrics such as requested memory do not change during execution.

5.3 Regular and Irregular Waveform

System metrics that have less than 4 frequency components are called regular waveform and those with more than 3 frequency components are called irregular waveform. Most HPC applications produce an irregular waveform in their system metrics. FT is the only application that has more metrics with regular waveform than irregular waveform. The application CoMD generates more system metrics with waveform (around 35%) compared to all other applications (averaging at around 25%).

HPC Applications and different Waveform It was unexpected to find that different applications produce different kinds of waveform. For example, FT produces more regular waveform time series data than other applications. This could be due to FT solving a numerical equation and using long distance communication in repeating and cyclic patterns.

5.4 System Metric Similarity to Sound

Complex musical pieces have multiple frequency components and are more similar to the definition of irregular waveform, and thus have similar characteristics as most applications in the dataset with the exception of FT, which has more regular waveform. The metric energy(J)_cray_aries_r is particularly similar to a waveform of the sound produced by an oboe. The sampling frequency of our dataset was 1/s, while normal sound sampling is done at multiple 1000/s. This makes comparison beyond visual analysis difficult without a lot of data transformations. However, if the sampling frequency is disregarded, the waveform of the metric user_procstat has a certain similarity to the waveform of pop music on the piano.

Specific Metric Similarity to Instruments It is disappointing that beyond the current findings, no further similarities between system metrics and sound could be found or investigated. But the results are encouraging enough to leave space for future work.

Chapter 6

Conclusion

The main contribution of this thesis is a systematic analysis of system metrics from HPC monitoring data. The results exposed metric groups with similar characteristics, such as being zero-constant or having waveform. To complement the statistical analysis of metrics, this work also contains a conceptual overview sorting metrics into the higher-level groups of CPU, memory, network, as well as lower-level conceptual groups such as RDMA (Remote Direct Memory Access) or NIC (Network Interface Controller). Additionally, the waveform characteristics of applications were investigated, exposing different waveform depending on the application (regular vs. irregular waveform). A small exploration of the system metric to sound was also provided.

The findings and answers to the research questions include:

– Groups of metrics have been found based on statistical properties that show, among other things, that the majority of system metrics are constant zero.

– Some system metrics were found to have frequency and most metrics with waveform come from the network group.

– The amount of frequency components inside the waveform of a system metric can be dependent on the executed HPC application.

– Some system metrics are very similar to specific musical instruments.

6.1 Future Work

There is still room for optimization and future work. Some of them are mentioned below:

For example, an audio player could be created for monitoring data by treating the data as if it were already sound. The challenge here would be on how this data could actually be listened to. Factors such as the scale and speed of the sound play a role here, and therefore a further approach could perhaps be to use audification for the investigation of HPC monitoring data which includes wavelike-shape data.

In addition, if the ability to listen to data exists, a tool similar to Shazaam could be developed that would be able to detect certain events and conditions, including performance degradation and idle time by listening to the sounds of the application.

Further looking at regular and irregular waveform could bring us better understanding of the HPC monitoring data. Research could be done to find out what the reasons and triggers are for some metrics being regular and others not, or why some applications generate more regular waveform than others.

Bibliography

[1] Emre Ates, Ozan Tuncer, Ata Turk, Vitus J Leung, Jim Brandt, Manuel Egele, and Ayse K Coskun. Artifact for Taxonomist: Application Detection through Rich Monitoring Data [online]. Available: https://doi.org/10.6084/m9.figshare.6384248.v1, [Accessed: September 14, 2021], 2018.

[2] Emre Ates, Ozan Tuncer, Ata Turk, Vitus J Leung, Jim Brandt, Manuel Egele, and Ayse K Coskun. Taxonomist: Application detection through rich monitoring data. In *European Conference on Parallel Processing*, pages 92–105. Springer, 2018.

[3] NTi Audio. Fast Fourier Transformation FFT [online]. Available: https://www.nti-audio.com/de/service/wissen/fast-fourier-transformation-fft [Accessed: November 22, 2021].

[4] D. Bailey, E. Barszcz, J. Barton, D. Browning, R. Carter, D. Dagum, R. Fatoohi, S. Fineberg, P. Frederickson, and T. Lakinski et al. The NAS Parallel Benchmarks [online]. Technical report, 1994.

[5] "Classical molecular dynamics proxy application", README [online]. Available: https://github.com/ECP-copa/CoMD [Accessed: November 22, 2021].

[6] Florian Dombois, Gerhard Eckel, Thomas Hermann, Andy Hunt, and John G Neuhoff. The Sonification Handbook, 2011.

[7] Duden. Network [online]. Available: https://www.duden.de/rechtschreibung/Network [Accessed: November 24, 2021], 2021.

[8] J.M.K.C. Donev et al. Energy Education - Energy vs power [online]. Available: https://energyeducation.ca/encyclopedia/Energy_vs_power [Accessed: November 28, 2021], September 2021.

[9] Wikimedia Foundation. Spectrogram [online]. Available: https://en.wikipedia.org/wiki/Spectrogram [Accessed: November 22, 2021], October 2021.

[10] Free-Loops.com. Free Piano Loops and Piano Sounds [online]. Available: http://free-loops.com/free-loops-find.php?term=piano&page=2 [Accessed: November 5, 2021], November 2007.

[11] GeeksforGeeks. Kernel in Operating System [online]. Available: https://www.geeksforgeeks.org/kernel-in-operating-system/ [Accessed: November 28, 2021], July 2020.

[12] Cody Henthorne, Ivica Ico Bukvic, Pardha S Pyla, and Eli Tilevich. Tuning Complex Systems by Sonifying Their Performance Data. Technical report, Department of Computer Science, Virginia Polytechnic Institute & State . . . , 2013.

[13] Techopedia Inc. What is Cache Memory? [online. Available: https://www.techopedia.com/definition/6307/cache-memory [Accessed: November 28, 2021], August 2016.

[14] Techopedia Inc. What is shared Memory? [online]. Available: https://www.techopedia.com/definition/13279/shared-memory [Accessed: November 28, 2021], December 2016.

[15] The Travel Insider. What Makes Digital Music Good (or Bad) Quality?[online]. Available: http://blog.thetravelinsider.info/2014/06/makes-digital-music-good-bad-quality.html [Accessed: November 24, 2021], June 2014.

[16] ITWissen.info. Netzwerkkarte [online]. Available: https://www.itwissen.info/Netzwerkkarte-network-interface-controller-NIC.html [Accessed: November 26, 2021], May 2019.

[17] T. Jakobsche. "Benchmark scheduling and communication behavior". Master Project, University of Basel, Faculty of Science, Department of Mathematics and Computer Science, 2018.

[18] Bodo Bauer Shen Feng Stefani Seibold Jorge Nerin, Terrehon Bowden. The /proc Filesystem - The Linux Kernel documentation [online]. Available: https://www.kernel.org/doc/html/latest/filesystems/proc.html#meminfo [Accessed: November 24, 2021], October 1999.

[19] Mathangi K. Load average: What is it, and what's the best load average for your Linux servers? [online]. Available: https://www.site24x7.com/blog/load-average-what-is-it-and-whats-the-best-load-average-for-your-linux-servers [Accessed: November 24, 2021], 2020.

[20] Mathangi K. Load [online]. Available: https://de.wikipedia.org/wiki/Load [Accessed: November 24, 2021], April 2021.

[21] "3D Sn deterministic particle transport", README [online]. Available: https://github.com/LLNL/Kripke [Accessed: November 22, 2021].

[22] Don Domingo Laura Bailey Marek Suchánek, Milan Navratil. Non-uniform memory access [online]. Available: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/performance_tuning_guide/s-memory-transhuge [Accessed: November 28, 2021], July 2018.

[23] ”Adaptive Mesh Refinement Mini-App”, README [online]. Available: https://github.com/Mantevo/miniAMR [Accessed: November 22, 2021].

[24] ”MiniGhost Halo Exchange Mini-Application”, README [online]. Available: https://github.com/Mantevo/miniGhost [Accessed: November 22, 2021].

[25] ”MiniMD Molecular Dynamics Mini-App”, README [online]. Available: https://github.com/Mantevo/miniMD. [Accessed: November 22, 2021].

[26] Pacific Northwest Seismic Network. What is a spectrogram? [online]. Available: https://pnsn.org/spectrograms/what-is-a-spectrogram [Accessed: November 22, 2021].

[27] David A. Rusling Nikos Drakos. The Kernel Swap Daemon (kswapd) [online]. Available: http://www.science.unitn.it/~fiorella/guidelinux/tlk/node5.html [Accessed: November 28, 2021], 1996.

[28] Vibration Research. What is a Spectrogram? - Signal Analysis[online]. Available: https://vibrationresearch.com/blog/what-is-a-spectrogram/ [Accessed: November 26, 2021], October 2021.

[29] Maarten Schenk. The Sound of Computing. Bachelor Thesis, University of Basel, Faculty of Science, Department of Mathematics and Computer Science, 2021.

[30] Robert Sheldon. Remote Direct Memory Access (RDMA) [online]. Available: https://searchstorage.techtarget.com/definition/Remote-Direct-Memory-Access [Accessed: November 26, 2021], November 2021.

[31] Stackoverflow. What is active memory and inactive memory [online]. Available: https://stackoverflow.com/questions/18529723/what-is-active-memory-and-inactive-memory [Accessed: November 28, 2021], August 2013.

[32] Techopedia. What is a Central Processing Unit (CPU)? [online]. Available: https://www.techopedia.com/definition/2851/central-processing-unit-cpu [Accessed: November 26, 2021], 2021.

[33] Techopedia. What is an SMS Gateway (SMSG)? [online]. Available: https://www.techopedia.com/definition/2978/sms-gateway [Accessed: November 26, 2021], 2021.

[34] Techopedia. What is Virtual Memory (VM)? [online]. Available: https://www.techopedia.com/definition/4773/virtual-memory-vm [Accessed: November 26, 2021], 2021.

[35] The SciPy community. scipy.signal.spectrogram - scipy v1.7.1 manual [online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.spectrogram.html [Accessed: November 26, 2021], August 2021.

[36] Wikipedia. Non-uniform memory access [online]. Available: https://en.wikipedia.org/wiki/Non-uniform_memory_access [Accessed: November 28, 2021], November 2021.

[37] Wikipedia. Page (computer memory) [online]. Available: https://en.wikipedia.org/wiki/Page_(computer_memory) [Accessed: November 29, 2021], November 2021.

[38] Wikipedia. Slab allocation [online]. Available: https://en.wikipedia.org/wiki/Slab_allocation [Accessed: November 29, 2021], November 2021.

[39] Wikipedia. Vmstat [online]. Available: https://www.kernel.org/doc/html/latest/filesystems/proc.html#meminfo [Accessed: November 24, 2021], July 2021.

University of Basel

Faculty of Science

Declaration on Scientific Integrity

(including a Declaration on Plagiarism and Fraud)
Translation from German original

Title of Thesis:	Statistical Characterization of HPC Monitoring Data
Name Assesor:	Prof. Dr. Florina M. Ciorba
Name Student:	Monika Multani
Matriculation No.:	18-061-663

With my signature I declare that this submission is my own work and that I have fully acknowledged the assistance received in completing this work and that it contains no material that has not been formally acknowledged. I have mentioned all source materials used and have cited these in accordance with recognised scientific rules.

Place, Date: Aesch, 29.11.2021 Student: [illegible]

Will this work be published?

No [✓ Yes.] With my signature I confirm that I agree to a publication of the work (print/digital) in the library, on the research database of the University of Basel and/or on the document server of the department. Likewise, I agree to the bibliographic reference in the catalog SLSP (Swiss Library Service Platform). (cross out as applicable)

Publication as of: 01.12.2021

Place, Date: Aesch, 29.11.2021 Student: [illegible]

Place, Date: ______ Assessor: ______

Please enclose a completed and signed copy of this declaration in your Bachelor's or Master's thesis .