Analysis of Individual HPC System Metrics through Audification

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

Li Ting Luong
liting.luong@stud.unibas.ch

December 21, 2021

Abstract

In an ideal world, systems and applications run efficiently, reliably, and with high performance. However, in the real world this is not always the case. Monitoring data generated by High Performance Computing (HPC) systems is analyzed to understand the corresponding problems and to have better understanding of the system and applications. Traditionally, monitoring data is analyzed and investigated through visualization and resulting plots. However, there exists an alternative approach that is not well explored, namely listening to the data. Listening to data can have advantages over simply looking at plots, because the human auditory perception is good at picking up on repeating patterns and changes in sound. The goal of this project is to convert HPC monitoring data to sound files through audification. We want to analyze the data acoustically, instead of the traditional visual analysis approach. Our results show that audification can assist in recognizing repeated executions of applications, comparing multiple different applications, and investigating different inputs and nodes of a specific application. We used the Python Wave Module to audify specific system metrics that are characterized by frequency and have a waveform. To systematically compare multiple application executions, we created an overview of the strongest frequency components of our chosen system metrics. Applying Principle Component Analysis (PCA) to our data and plotting the resulting clusters, exposed that some HPC applications repeat more consistently (reproducing similar system metric waveform compared to other executions), while other applications have more variation and change between repeated executions. Our discoveries can serve as a base for future studies aimed at recognizing applications based on frequency components in their execution behavior, similar to e.g. how Shazam recognizes songs. Our work also opens further research on why some applications have more variation and generate different system metric waveform for repeated executions compared to other applications.

Contents

1 Introduction **3**
- 1.1 Motivation and Challenges . . . 3
- 1.2 Goals and Research Questions . . . 4
- 1.3 Solution, Utility, and Findings . . . 4
- 1.4 Technical Background of FFT . . . 5

2 Related Work **6**
- 2.1 Data and Sound . . . 6
- 2.2 Sonification . . . 6
- 2.3 Audification . . . 7

3 Methods **8**
- 3.1 Audification . . . 8
- 3.2 Python Wave Module . . . 9
  - 3.2.1 Sampling Rate . . . 9
  - 3.2.2 Channels . . . 9
- 3.3 Method Justification . . . 10
- 3.4 Dataset: Taxonomist Dataset . . . 10
- 3.5 Implementation . . . 12

4 Results **14**
- 4.1 Experiments . . . 14
  - 4.1.1 Suitability of Audified Monitoring Data . . . 14
  - 4.1.2 Comparison between Multiple Applications . . . 15
  - 4.1.3 Comparison between Inputs to the same Applications . . . 15
  - 4.1.4 Comparison between Nodes of the same Execution . . . 15
  - 4.1.5 Comparison between Runs of the same Application-Input Pair . . . 15
- 4.2 Evaluation of the Solution . . . 16
  - 4.2.1 Node Frequency Table . . . 16
  - 4.2.2 Visualization of Node Frequency Table . . . 18

5 Discussion **24**
- 5.1 Sonification vs. Audification . . . 25
- 5.2 Limitations . . . 25

6 Conclusion **26**
6.1 Main Contributions 26
6.2 Future Work 27

Appendices **30**
Node Frequency Table 30
Visualization of Node Frequency Table 30

Chapter 1

Introduction

Scientific applications demand high computing power to solve problems. This is where High-Performance Computing (HPC) systems, which offer computing power, come into play. HPC systems generate a large amount of monitoring data, which reflects the system's response to running applications.

1.1 Motivation and Challenges

In an ideal world, systems and applications run efficiently, reliably and with high performance. However, this is not always the case. Monitoring data generated by HPC systems is analyzed to have a better understanding of the system and applications. In this bachelor thesis, a specific technique of analysis, audification, is explored. It enables the user to listen to data as if it were sound. Audification can help to: (a) explore an alternative to visual analysis of data, (b) identify repeated executions of applications, (c) compare multiple different applications, and (d) investigate different inputs and nodes of an individual application.

It is not trivial whether the sound produced by audifying data can be used to analyze monitoring data. Listening to the sound files may reveal that no repeated sound pattern can be used as a reference point, showing that audification is not effective. This exact problem was discovered when sonification was used to listen to data. But, what makes sonification different from audification? Sonification maps single data points to a musical scale, which is then played with a digital instrument, whereas audification treats data as if it were already sound.

Another challenge to overcome is the ability to convert monitoring data to sound. Because audification is only suitable for data with waveform, only monitoring data (system metrics) with waveform are considered. The next difficulty is to objectively state similarities and differences in sound using numbers rather than just listening. Furthermore, the numbers should be visually represented so that the desired information about similarities can be extracted easily.

1.2 Goals and Research Questions

To overcome the challenges defined in the previous section, a method for converting time series to sound is required. For this purpose, Python provides a library called 'wave module', which makes it possible to convert data to sound. Following that, a systematic method for comparing sound similarities and differences is needed as listening to all sound files would be beyond the scope of this project. Consequently, a table in which the most prominent frequencies from the sound files were recorded, was created. This allows to visually extract information about whether or not there are similarities.
The research questions are:

- Can we listen to monitoring data (system metrics) of applications through audification?
- Do we hear differences between multiple applications?
- Do we hear differences between inputs to the same applications?
- Do we hear differences between nodes of the same execution?
- Do we hear differences between runs of the same application-input pair?

1.3 Solution, Utility, and Findings

By using the Python Wave Module, the system metrics with waveform were converted successfully into sound files that can be listened to. This is beneficial for HPC reseachers, because rather than analyzing data visually, data can be analyzed through audification. It may also bring some information to the surface that would have remained hidden if visualization was used. Another advantage could be that listening to data is easier or faster than looking at plots. Lastly, but worth mentioning is that blind people can take part in research using audification. Because blind people's hearing abilities are enhanced when compared to hearing people, they may be able to detect even minor differences in sound. The findings include:

- Audification can be used to recognize repeated application executions.
- It can be further used to investigate applications by comparing different input sizes.
- Audification can also help to analyze individual application executions, by listening to the differences between several nodes of the same run.
- We found that applications generally produce the same frequency on all allocated nodes.
- We further found out that input sizes to applications change the frequency components of the system metrics.
- Lastly, we found out that some applications repeat more consistently than others, meaning they generate the same waveform for different repeated executions.

1.4 Technical Background of FFT

The terms 'Fast Fourier Transform' (FFT) and 'Spectrogram' will be defined briefly in this section, as well as their significance to the work.

Fast Fourier Transform (FFT)
The Fast Fourier Transform is a technique used for measurements regarding audio and acoustics measurements. Individual spectral components and therefore frequency information can be obtained by applying the Fast Fourier Transform to a signal. In more detail, the Fast Fourier Transform is an optimization of the 'Discrete Fourier Tranformation' (DFT). The data points collected over time form a signal, which is then divided into frequency components. Each of these represents an oscillation with amplitude and phase. [4]

Spectrogram
A spectrogram is a tool for visually displaying the frequency spectrum of a signal, or the "loudness" of a signal over time, in a waveform. Spectrograms are used in science to display the frequencies of sound waves produced by, for example, humans or animals. [9][19]

What is the significance of the 'Fast Fourier Transform' and 'Spectrogram' in this work? The spectrogram was generated applying the Fast Fourier Transform to a time series. This was done in order to obtain the strongest frequency components of the chosen system metrics, which would later be used for sound comparison.

Chapter 2

Related Work

2.1 Data and Sound

Listening to computing data is not a novel approach in the scientific community. By conducting a survey, Paul Vickers and James L. Alty [24] investigated whether computing information can be conveyed through sound and whether musical experience influences what participants can hear. According to survey results, participants with musical experience were able to recognize tone changes better. This finding is important for this work because it shows that listening to audified monitoring data can support in hearing similarities and differences in data. However, when it comes to monitoring data analysis, they do not look for similarities between applications or different settings, which is more important to High Performance Computing researchers that want to compare and analyze application behavior.

2.2 Sonification

The paper "Tuning Complex Systems by Sonifying Their Performance Data" [10] Henthorne et al. also discussed a possible method for conveying performance data, namely sonification. Sonification allows the programmer to listen to information via sound. While examining sonification, it was discovered that survey participants could detect changes in sound characteristics. Based on their newly acquired knowledge, participants were instructed to tune the configuration settings to see if they could outperform the standard configuration solely through sonification, which proved to be effective. All participants were able to tune the settings, resulting in improved performance. Even so, we run into the same issue as before because they don't make any comparisons between different parameters in the analysis.

Sonification and the potential benefits of combining it with High Performance Computing (HPC) was also investigated by Maarten Schenk in his Bachelor thesis "The Sound of Computing" [22]. High Performance Computing researchers must analyze a lot of data to optimize the performance of their applications. Instead of comparing data visually, this approach tries to convert monitoring data into sound to determine if this method of comparing data can be beneficial to researchers.

As a result, it was discovered that certain application parameters had characteristics that could be extracted through sound and that this approach has the potential to be investigated further. Nonetheless, the technique of sonification differs from the method of audification that we utilize. The main difference between these two methods is that sonification maps data points to a musical scale, whereas audification, which is used in this work, examines the data as if it were already sound.

By nature, sonification is more suited for constant system metrics, because a constant value can easily be mapped to a musical scale. System metrics that are changing a lot, e.g. with waveform, are not so easily mapped or result in pseudo-random sound. If the metrics already have waveform, the method of audification is more suited. In this thesis the focus is on audification and metrics with waveform.

2.3 Audification

Previous related works have shown that researchers have attempted to find several methods for working with computing information. Instead of using the traditional method of visually analyzing data, performance data was converted into sound to enable acoustic analysis. Audification is not a new technique; it has been used before, but not in the context of HPC monitoring data.

Chapter 3

Methods

3.1 Audification

This paper proposes an approach for analyzing HPC monitoring data using audification. What audification does in detail is to take a time series, which is defined as "[...] a sequence of data points collected over an interval of time" [23] and translate it directly to sound without altering the data. In this project, the time series are the HPC monitoring data where only those that resemble waveforms are investigated. This is due to the fact that the produced sound by audifying these time series are the best to experiment with. On the opposite, time series that are constant do not provide much diversity in sound.

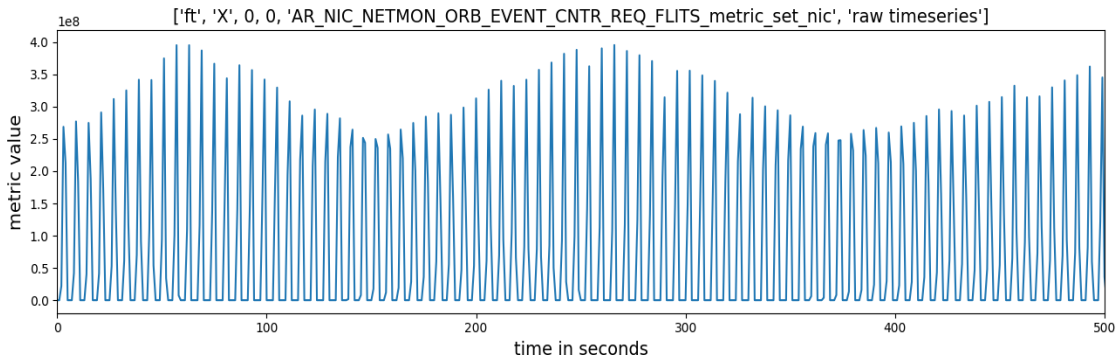

Figure 3.1: Waveform Time Series

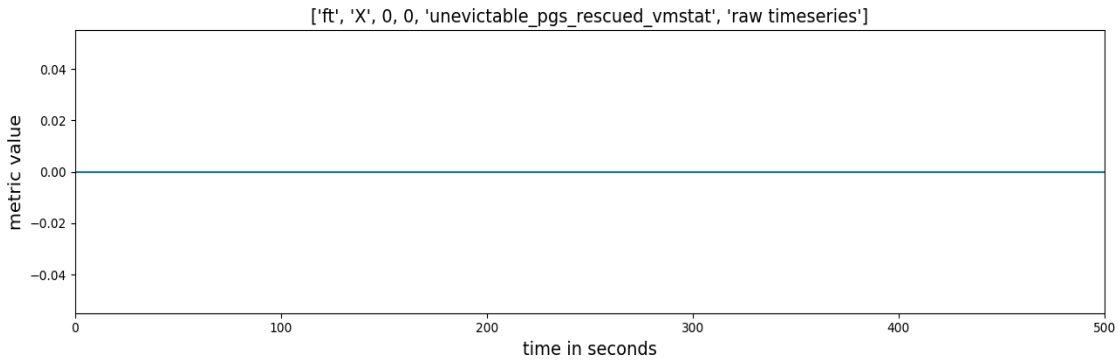

Figure 3.2: Constant Time Series

3.2 Python Wave Module

To enable listening to monitoring data, the time series can be saved as a wav file. Python comes in handy in this case because it has a module called 'wave' [14] that allows us to audify our data. This module provides a set of parameters that can be manually configured. Sampling rate and channels were the primary parameters investigated. They will be explained in the sections that follow.

3.2.1 Sampling Rate

The sampling rate is defined as "the number of samples per second [...] taken from a continuous signal to produce a discrete or digital signal" [12]. Various sampling rates were tested in order to determine the influence of sampling rate on the sound generated by audification. As a starting point, the sampling rate of 1000 hertz was randomly chosen and then increased by 200 or 300 hertz until the sampling rate of 2500 hertz was reached. Because when the sampling rate is set too high, it causes the pitch to rise, and the length of the sound file becomes too short. The opposite occurs if the sampling rate is set too low. As a result, the sampling rate of 1500 hertz was chosen as the best for distinguishing sound. This sampling rate was applied in all upcoming experiments.

3.2.2 Channels

The Python Wave Module includes a method called setnchannels() that enables choosing two channel settings: mono and stereo. The distinction between mono and stereo sound [...] "is the number of channels used to record and playback audio" [21]. While mono uses one channel to convert a signal into sound, stereo uses two channels. This makes it possible for stereo to create width with sound, while mono cannot. Stereo sound can portray [...] "sound coming from different sources and positions" [...] [11]. Since the effect of stereo creating new dimensions is not primarily needed for audification, mono was chosen as the default channel setting.

3.3 Method Justification

This project's aim is to make it possible to listen to monitoring data. There are two methods available to achieve this: sonification and audification. Because sonification had already been investigated, a new approach to achieving the same goals was discovered. As it was decided that the code will be written in Python, it was only natural to select a Python library. The 'wave module' is a suitable library that matches audification because it can convert monitoring data to an audio file in wav format without altering the data, which is the opposite of what sonification does.

3.4 Dataset: Taxonomist Dataset

The data utilized in the project came from the Taxonomist dataset [2][3], which applied the machine learning technique random forest to classify known applications and detect unknown ones. This dataset contains a total of 11 applications, all of which are benchmarks, meaning that they do the same thing every execution, see Table 3.1.

It also provides several parameters that can be configured, the first of which is the input size, where the work changes depending on which input size is selected. These applications paired with specific inputs were run on multiple nodes, and each node collected 563 metrics as a time series. The applications can also be executed multiple times with the same configuration. It should be noted that there are 721 metrics and 16 applications in total. However, only 563 metrics and 11 applications have been published in open source.

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 Overview

The following application descriptions are taken from several sources:

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. [13][5]

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. [13][5]

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). [13][5]

LU - Gauss-Seidel solver
LU is simulating a CFD problem like BT, but solves regular-sparse, lower and upper triangular systems. [13][5]

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. [13][5]

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. [13][5]

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. [15]

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. [17]

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. [6]

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. [16]

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.) [1]

3.5 Implementation

The following Python code audifies the time series using the `audify()` function:

```
def audify():
    samplingRate = 1500  # hertz
    audio = wave.open('sound/ft/input/input_X.wav', 'w')
    audio.setnchannels(1)
    audio.setsampwidth(2)
    audio.setframerate(samplingRate)
    data = norm(timeseries) # normalize time series
    audio.writeframesraw(data.tobytes())
    audio.close()
```

The wave read object methods used in the code are described in detail below. The following is a description taken from tutorialspoint: [25]

- open(): This function opens a file to read/write audio data. The function needs two parameters - first the file name and second the mode. The mode can be 'wb' for writing audio data or 'rb' for reading.
- setnchannels() : Set the number of channels. 1 for Mono 2 for stereo channels
- setsampwidth(): Set the sample width to n bytes
- setframerate(): Set the frame rate to n
- writeframesraw(): Write audio frames, without correcting

For each execution, the `findMaxNodeFrequency()` function determines the frequency with the highest amplitude per node:

```
def findMaxNodeFrequency():
  max_freq = 0
  for peak in peaks:
    amplitude = spectrogram[peak]
    if amplitude > spectrogram[max_freq]:  # save frequency with highest amplitude
      max_freq = peak

  # save frequency to corresponding node dictionary
  if node_id == 0:
    node_0_dict[run_id] = max_freq
  elif node_id == 1:
    node_1_dict[run_id] = max_freq
  elif node_id == 2:
    node_2_dict[run_id] = max_freq
  elif node_id == 3:
    node_3_dict[run_id] = max_freq
```

The frequency with the highest amplitude, or frequency peak, can be determined using the spectrogram. In the spectrogram, the amplitude of each frequency peak is compared to the amplitude of all other frequency peaks. Only when the highest peak for a given node is found is it saved in the corresponding node dictionary.

Chapter 4

Results

4.1 Experiments

The time series was converted into sound using the Python Wave Module. The experiments were executed for the network metric_id 0 and the applications FT, MG, and SP. The applications FT, MG, and SP were chosen based on criteria. The first condition was that they came from the NAS Parallel Benchmark Suite [8], which is widely known and utilized for research. Second, the applications BT, LU and SP simulate the same problem with some variations, namely CFD (computational fluid dynamics) that solves independent equations without much communication. SP was chosen to be the representative for these applications. Because CG and FT both do long distance communications, FT was picked out of the two. And lastly, MG was chosen as a third application to be examined since it conducts a different computation and short distance communication.
Since the input size L can only be applied to certain applications, it was omitted in this study. Only these parameter settings were investigated because it would be too challenging to investigate every possible combination of parameter settings. Eventually, the questions in the following sections were answered subjectively, with the main goal of determining similarities and differences in hearing.

Here are some more details regarding metric 0: Aries is the name of a network hardware of the company Cray that has its own network performance counters. The relationship between Aries and metric 0 (AR_NIC_NETMON_ORB_EVENT_CNTR_REQ_FLITS_metric_set_nic) is that it is a network metric from the Aries NIC (Network Interface Controller). It represents the aggregate network traffic through the NIC into the High Speed Network (HSN) used as interconnection between the nodes of the system.

4.1.1 Suitability of Audified Monitoring Data

Various metrics were examined to see what sound they produced in order to establish whether monitoring data (system metrics) are suitable for audification. As described in Section 2.3, only time series that resemble waveforms were analyzed. Because the sound generated by a constant time series returned a constant tone, there was limited space for experimentation. In general, audification has shown to be effective in converting monitoring data to sound.

4.1.2 Comparison between Multiple Applications

Section 4.1 stated that the experiments were only conducted out for applications FT, MG, and SP. But, for this experiment all applications were taken into consideration.
Listening to the applications displayed that they can be distinguished and recognized based on sound. FT, CG, CoMD, and miniGhost are easily recognizable compared to other applications. The remaining applications are not so easily distinguished based on sound. The reason for that could be that they perform similar computations, but this needs further analysis. For future work, analyze why some applications sound similar.

4.1.3 Comparison between Inputs to the same Applications

In this study, the input sizes X, Y, and L were investigated in combination with applications to see if the input sizes affect monitoring data and, more importantly, sound. When evaluating the listening part, it can be heard that the input sizes do influence monitoring data. In other words, the same application with different input sizes does not sound the same. The correlation is between what was defined in Section 3.4, where it was stated that input sizes impact the performance that computers do. This was confirmed by the experiment.
It is worth noting that CoMD was found to be most stable, with only minimal changes in the values for the frequency peaks.

4.1.4 Comparison between Nodes of the same Execution

In order to analyze whether nodes with the same application and input size differ in sound, the four nodes of an execution were compared to one another. It was discovered that all nodes in FT sound extremely similar, if not identical, to the human ear. Minor changes in sound were 'only' visually detectable with the help of an audio player called "Resonic Player Beta" [20], which had an inbuilt feature that allowed us to visualize the sound file. On the other hand, the nodes for applications MG and SP also sounded similar but there was a pairwise similarity. The pairing node IDs for MG were 0, 2 and 1, 3. While SP's pairing node IDs were 0,3 and 1,2. One possible explanation for this result may be that the metric 0 belongs to the network group. As a result, the pairing nodes may represent clients who send packets to each other. While sending, the packet order stays; this is also reflected in the sound, with two nodes sounding similar.

In summary, the answer to the question whether differences between nodes of the same execution can be heard is yes.

4.1.5 Comparison between Runs of the same Application-Input Pair

The experiment was carried out for the different run IDs 0,1,2, and 3. The intention behind this experiment was to see if running the same application-input pair would resolve in different sound characteristics. When it comes to applications FT and SP, the sound produced for multiple runs is extremely similar, nearly identical, whereas MG sounded similarly in general with some minor differences. However, the difference isn't significant enough. As an outcome, even with different run IDs (repeated executions), the applications can be distinguished.

4.2 Evaluation of the Solution

Hearing was used to extract the similarities and dissimilarities from previous experiments. However, not everyone hears the same thing when listening to the same sound. As a result, the outcomes are quite subjective. To illustrate that the outcomes can be objectively verified an overview table called 'Node Frequency Table' was created to substantiate similarities and dissimilarities in numbers. In a later step, the table was plotted for additional visual analysis.

4.2.1 Node Frequency Table

The frequencies with the highest amplitude per node, or frequency peaks per node, were compared to see if the similarities or differences heard in sound could be verified by these numbers. Figure 4.1 shows the frequency peaks computed in a spectrogram for application FT.

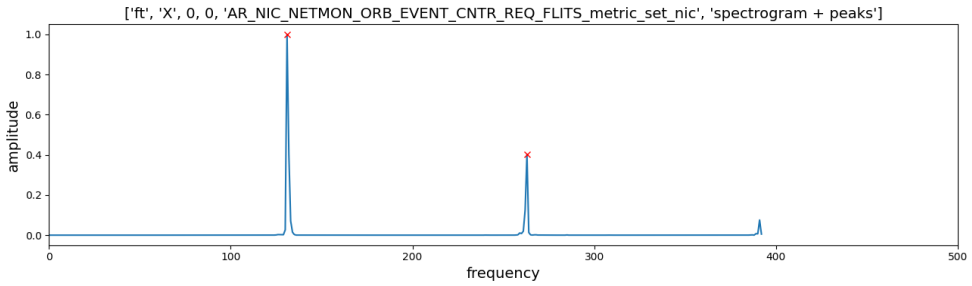

Figure 4.1: Frequency Peaks for Application FT

There are two frequency peaks denoted by a red 'x', with only the highest peak examined for all application-input pairs. The highest frequency peak is more likely to repeat for repeated executions of the same application. On the opposite, 'weaker' frequencies can be overshadowed or even completely hidden by background noise and perturbations through other processes or applications running on an HPC system.

Table 4.1 is the first version, which records the peak frequencies for each node and the metric 0. As an example of how the table works, the data in Figure 9 shows that the frequency peak for application FT with input size X and node ID 0 is at 131. Therefore, the value 131 was denoted in the first cell. For some application-input pairs there were two frequency peaks at the exact same height, which were separated by a semicolon. In the absence of a peak, the term 'N/A' was entered into the cell.

Comparing the values for application FT, MG and SP with input size X reveals that they are indeed distinct. The table also illustrates how different input sizes affect the work done by the applications. Finally, it proves that the peak frequencies of the nodes are similar or pairwise similar and that the executed runs are identical. In conclusion, the table confirms the findings of the experiments described in Section 4.1. and demonstrates that it is effective in expressing what can be heard in numbers.

A visual representation of the 'Node Frequency Table' was required because looking at so many numbers and attempting to find differences in numbers is not feasible. For this purpose, the first version of Table 4.1 was expanded so that it lists the values for all nodes when they are executed 30 times.

Highest Amplitude Frequency per Node				
App+Input	Node 0	Node 1	Node 2	Node 3
FT + X	131	131	131	131
FT + Y	258	258	258	258
FT + Z	216	216	216	216
MG + X	286	234	347	234
MG + Y	149	149	119	149
MG + Z	197	263	231	263
SP + X	26	26	26	26
SP + Y	84	84	84	84
SP + Z	168	168	168	168
LU + X	354	354	194	194
LU + Y	174	174	174	87
LU + Z	251	251	251	251
BT + X	96	96	96	96
BT + Y	103	103	103	103
BT + Z	223	223	223	223
CG + X	52	52	52	52
CG + Y	242	242	242	242
CG + Z	298	298	298	298
miniGhost + X	116	116	116	116
miniGhost + Y	101	101	101	101
miniGhost + Z	69	70	69 ; 139	70
miniAMR + X	75	75	75	75
miniAMR + Y	1	1	1	1
miniAMR + Z	17	8	8 ; 17	8
miniMD + X	1	1	1	1
miniMD + Y	139	139	227	139
miniMD + Z	44	44	44	44
Kripke + X	303	304	228	304
Kripke + Y	3	3	3	3
Kripke + Z	167	N/A	195	222
CoMD + X	233	233	233	233
CoMD + Y	169	169	169	169
CoMD + Z	228	228	228	228

Table 4.1: First version: 'Node Frequency Table'

Monika Multani provided in her thesis "Statistical Characterization of HPC Monitoring Data" [18] an overview table with metrics indicating which time series had the most waveform. A total of 20 metrics were recorded, and for each of these metrics a 'Node Frequency Table' was created. However, for illustration purposes, the table was divided into 11 sections, each of which describes a different application. Also used were the abbreviations 'Ips' (input size), 'N ID' (Node ID) and 'R' (Run). The remaining tables for the other applications are contained in the appendix. The Table 4.2 illustrates how the frequency with the highest amplitude per node was recorded. The metric analyzed in Table 4.2 has metric ID 0, and the application researched was FT. The 'Input' column defines the input that was paired with the application and executed 30 times for each of the four nodes.

Ips	N ID	R 0	R 1	R 2	R. 3	R 4	R 5	R 6	R 7	R 8	R 9	R 10	R 11	R 12	R 13	R 14	R 15	R 16	R 17	R 18	R 19	R 20	R 21	R 22	R 23	R 24	R 25	R 26	R 27	R 28	R 29
X	0	131	131	132	132	131	131	131	131	131	130	131	131	131	132	131	131	131	131	130	130	132	132	132	131	131	131	131	130	131	131
X	1	131	131	132	131	131	131	131	131	131	130	131	131	131	132	131	131	131	131	130	130	132	132	132	131	131	131	131	130	131	131
X	2	131	131	132	132	131	131	131	131	131	130	131	131	131	132	131	131	131	131	130	130	132	132	132	131	131	131	131	130	131	131
X	3	131	131	132	132	131	263	131	263	131	130	131	131	131	132	131	263	131	131	130	130	132	132	132	131	131	131	131	130	131	131
Y	0	258	258	257	258	258	258	258	258	257	258	259	258	259	259	258	258	259	258	258	258	258	258	258	258	257	259	258	259	258	259
Y	1	258	258	257	258	258	258	258	258	257	258	259	258	259	259	258	258	259	258	258	258	258	258	258	258	257	259	258	259	258	259
Y	2	258	258	257	258	258	258	258	258	257	258	259	258	259	259	258	258	259	258	258	258	258	258	258	258	257	259	258	259	258	259
Y	3	258	258	257	258	258	258	258	258	257	258	259	258	259	259	258	258	259	258	258	258	258	258	258	258	257	259	258	259	258	259
Z	0	216	217	217	218	216	217	216	216	215	216	217	215	217	216	216	216	216	216	216	216	216	216	216	215	216	217	217	216	216	216
Z	1	216	217	217	218	216	217	216	216	215	216	217	215	217	216	216	216	216	216	216	216	216	216	216	215	216	217	217	216	216	216
Z	2	216	217	217	218	216	217	216	216	215	216	217	215	217	216	216	216	216	216	216	216	216	216	216	215	216	217	217	216	216	216
Z	3	216	217	217	218	216	217	216	216	215	216	217	215	217	216	216	216	216	216	216	216	216	216	216	215	216	217	217	216	216	216

Table 4.2: Node Frequency Table FT

4.2.2 Visualization of Node Frequency Table

The goal of this section was to visualize the 'Node Frequency Table' with clustering. During this process, difficulties were encountered, which will be discussed here.

The first attempt was to plot the values for each run on the x-axis and the node IDs on the y-axis. This was done as a test for metric 0, and application FT using input X to determine if the generated plot produced helpful results, which it did not. Figure 4.2 contains no clusters, only lines parallel to the x-axis are drawn.

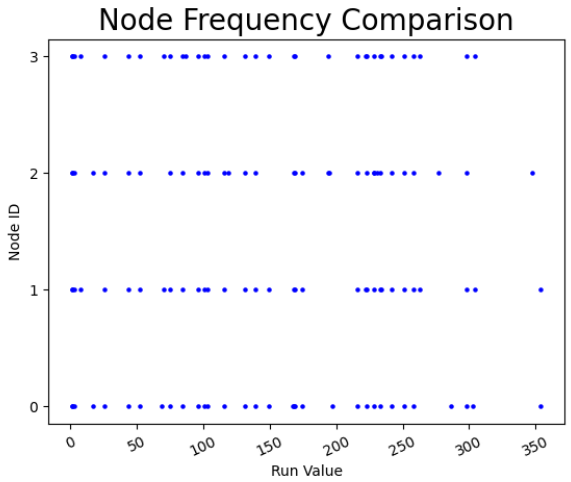

Figure 4.2: First Attempt: 'Node Frequency Comparison'

Plotting the values for the executed runs per node was the next concept to be tested for metric 0, and application FT with input X. However, adding a third dimension did not solve the problem of no clusters being found, as seen in Figure 4.3. Furthermore, there is no option to plot the values for node ID 3. As a consequence, this concept was abandoned.

Taken together, the attempts failed at visualizing the table where the frequency with the highest amplitude per node was recorded. The challenge was to represent the nodes' four dimensions in a two or three-dimensional layout. As a result, a new approach for displaying the frequency table was established that makes use of the "Principal Component Analysis" (PCA). PCA reduces the dimensionality of the data without losing information. It examines the correlation between dimensions and sets the goal of preserving as much information as possible about how the original data was distributed. This information is then saved in a minimum number of variables that are given for plotting. [7]

The four dimensions of the nodes were reduced to two dimensions since adding a third dimension would not contribute much as the nodes' values are mostly the same. For example, if three data points with value (40, 40, 40, 40) (50, 50, 50, 50), (60, 60, 60, 60) are to be plotted. The three data points will always be drawn in a line, no matter how many dimensions are added or removed.

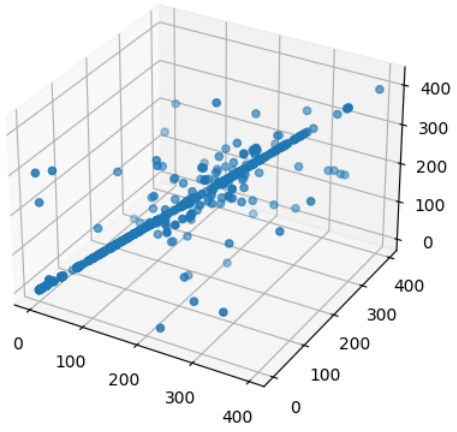

Figure 4.3: Second Attempt: 'Node Frequency Comparison'
x-axis: node ID 1 values, y-axis: node ID 0 values, z-axis: node ID 2 values

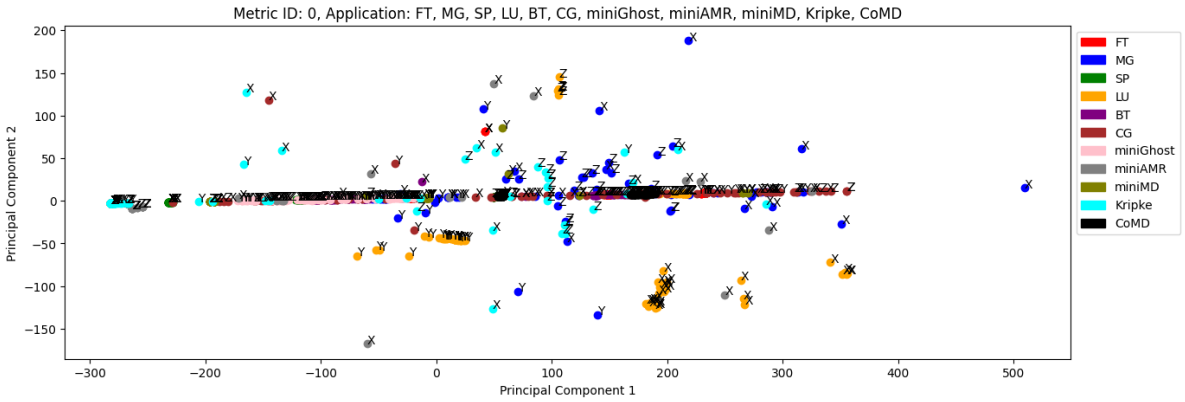

Figure 4.4: Frequency Peaks of all Applications

Figure 4.4 illustrates the 'Node Frequency Table' of metric 0. What is interesting in this data is that a line of data points can be seen. The reason for this was already established. It is also noticeable that applications MG and Kripke have more outliers, but application LU shows the desired input clusters. This plot, however, makes it difficult to see individual data points. As a consequence, instead of plotting all applications in a single frame, they were plotted for individual applications.

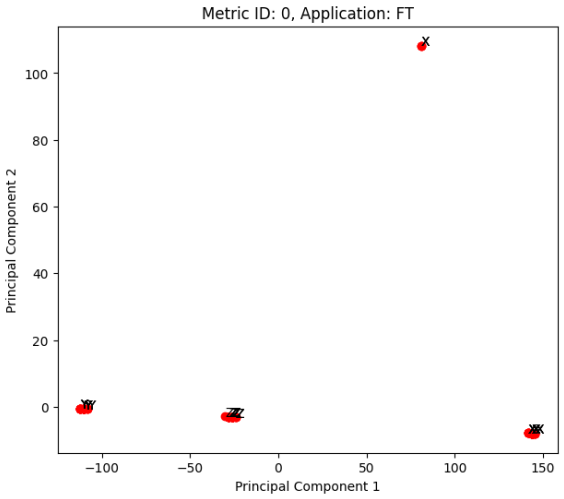

Figure 4.5: Frequency Peak Visualization: FT

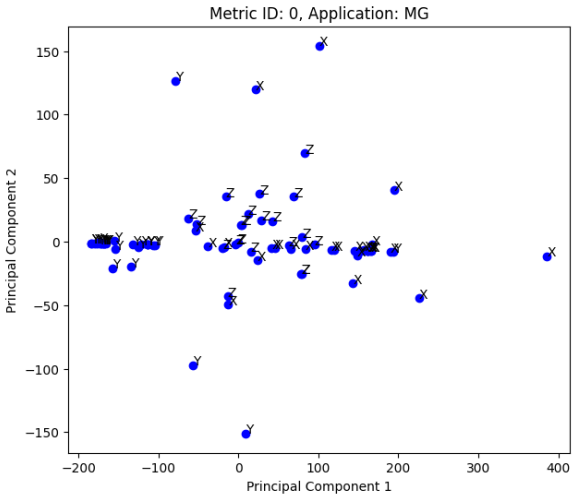

Figure 4.6: Frequency Peak Visualization: MG

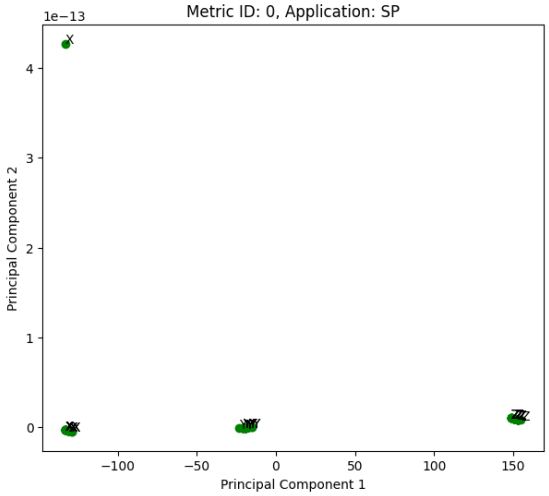

Figure 4.7: Frequency Peak Visualization: SP

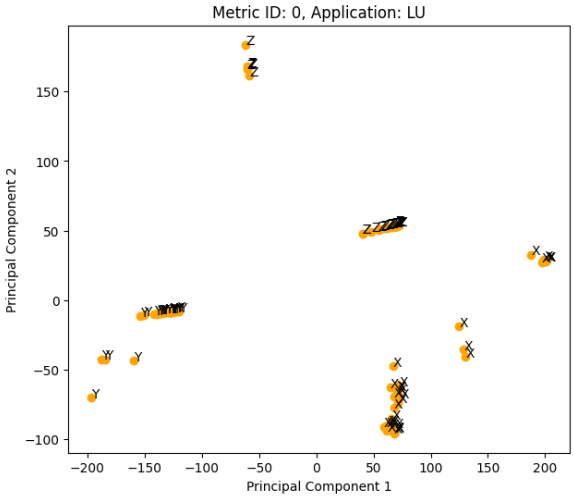

Figure 4.8: Frequency Peak Visualization: LU

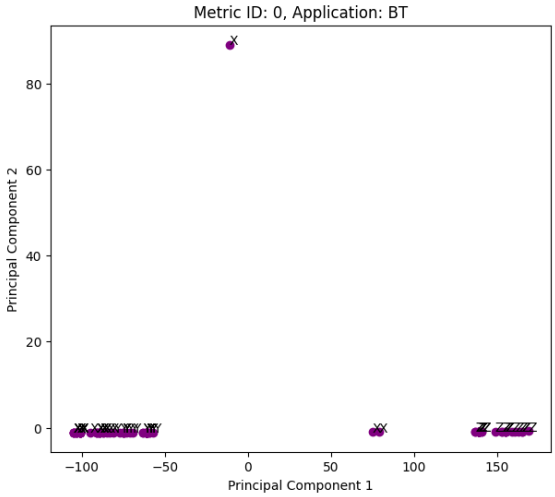

Figure 4.9: Frequency Peak Visualization: BT

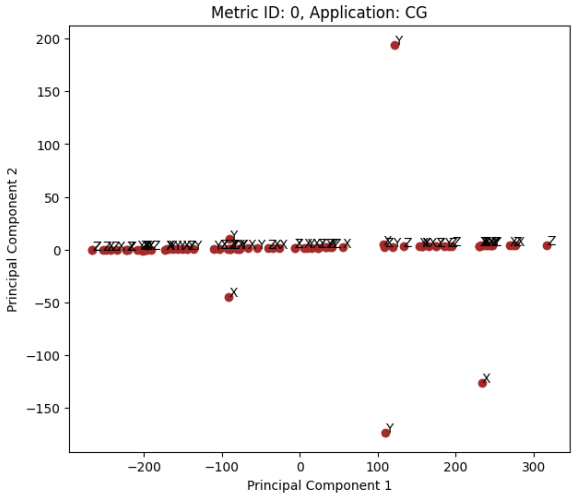

Figure 4.10: Frequency Peak Visualization: CG

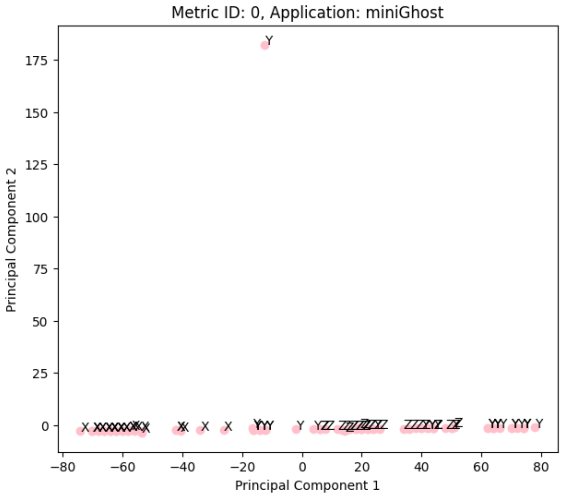

Figure 4.11: Frequency Peak Visualization: miniGhost

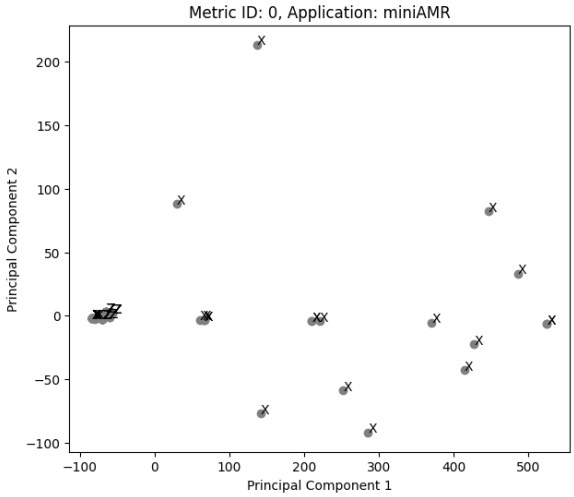

Figure 4.12: Frequency Peak Visualization: miniAMR

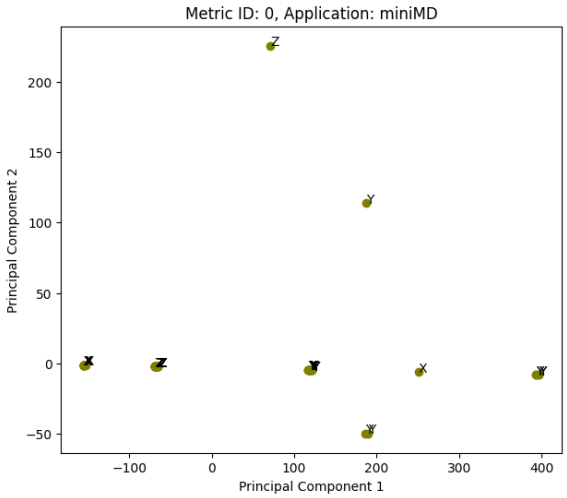

Figure 4.13: Frequency Peak Visualization: miniMD

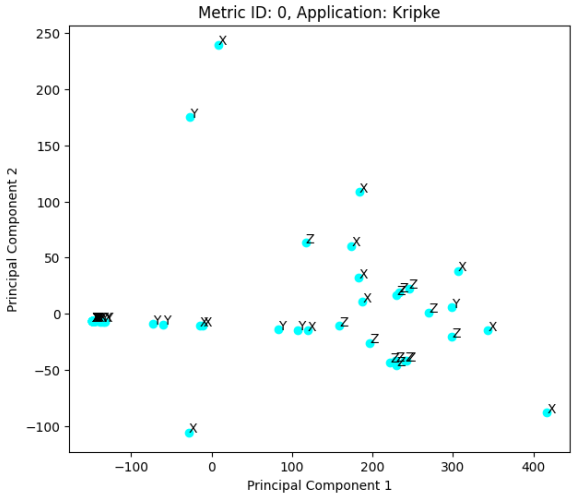

Figure 4.14: Frequency Peak Visualization: Kripke

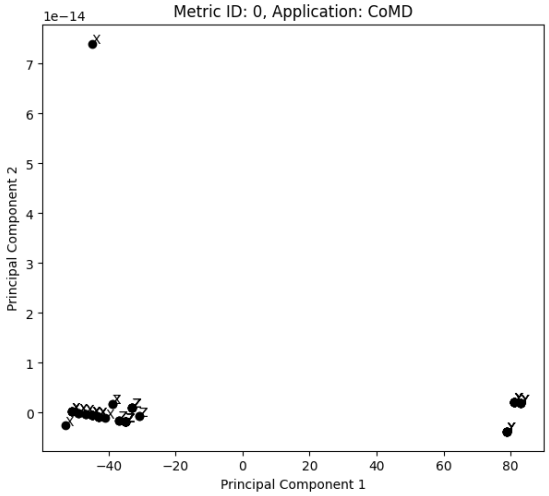

Figure 4.15: Frequency Peak Visualization: CoMD

As shown in Figure 4.5 for application FT, three clusters have been formed for the corresponding input sizes X, Y, and Z. Only for input size X is there an outlier; otherwise, FT always produces the same numbers for all executed runs per node. The same is true for the applications SP, LU, BT and CoMD. On the opposite, application MG does not always perform the same values. The data points are more scattered and different input sizes occasionally overlap. This also applies to miniAMR, miniMD and Kripke. The applications CG and miniGhost have a tendency to plot the data points in a straight line with a few isolated outliers.
The findings are not surprising when referring to Table 4.2. The values for applications FT and SP were practically identical for each execution, whereas the values for application MG varied greatly. This demonstrates that the visualization of the 'Node Frequency Table' generates the same results as the table itself. The plots for the remaining 19 metrics can be found in the appendix.

Chapter 5

Discussion

The initial objective of this project was to obtain answers to the following research questions:

- Can we listen to monitoring data (system metrics) of applications through audification?
- Do we hear differences between multiple applications?
- Do we hear differences between inputs to the same applications?
- Do we hear differences between nodes of the same execution?
- Do we hear differences between runs of the same application-input pair?

Utility of Audification
The first question is whether the audification of monitoring data (system metrics) can produce audible time series. This has been verified to be feasible.

Recognizing Applications
The answer to the second question, whether differences between multiple applications can be heard, is answered affirmatively. All applications can be distinguished based on sound. Some were easily recognizable due to their uniqueness, while others sounded similar but not identical to other applications.

Effect of Input Size
The next discovery about the correlation between different input sizes paired with the same application revealed that different input sizes do affect the output for all applications. When compared to other applications, CoMD proved to be the most stable.

Node Variation
For the research question, if differences between nodes of the same execution can be heard, the conclusion is, that for some applications the differences occurred pairwise. Other applications had practically no substantial difference because they sounded nearly identical.

Repeated Executions
The sound created by executing the same application-input pair indicated that the performed runs for FT and SP sounded practically identical, whereas the executed runs for MG sounded similar overall but with some minor changes.

5.1 Sonification vs. Audification

The main distinction between sonification and audification, as described in Section 2.2, is that sonification maps data points to a musical scale, whereas audification treats the data as if it were already sound. The aim of this section is to compare sound files created through sonification with audification. For the comparison, Maarten Schenk's sonification files (2021) are used for comparison, with just the applications CoMD, Kripke, MG and miniAMR from metric 0 being considered. [22] Moreover, all applications were executed with the input size X and the run ID 0.

The first noticeable change is the length of the sound file. The sonified time series has on average a duration of 5 minutes, but the audification files are in the seconds range. This enables the user to recognize applications within seconds. Listening to the sonified audio also revealed that the applications Kripke, MG and miniAMR could not be recognized, because all that can be heard are random piano tones being played without repetition. Only CoMD stood out because it uses an alternating rhythm of low and high tones. The uniqueness of CoMD can also be found in the audified version of CoMD. It stood out the most, among all applications.

According to Maarten Schenk (2021) discovered the same findings, that all applications besides CoMD were unrecognizable due to their similarity. CoMD, on the contrary, showed a unique pattern of anomalies. [22]

5.2 Limitations

The proposed method audification is a novel method of listening to monitoring data that has proven to be effective. This work, however, was limited in several ways. First, audification relies on users listening to sound, but not everyone hears things the same way. Some people can distinguish between sounds, while others cannot because the frequency range is too low or too high for them to hear. Second, the investigation part, which included listening, was limited to one metric with limited parameter changes since listening to all sound files that could be computed would be out of scope. Third, the results revealed that applications with specific input sizes have distinct sound characteristics that set them apart from other applications. But, this uniqueness implies that we can only recognize an application if the parameter settings (input and node configuration) are the same. Nonetheless, this feature can be advantageous in detecting cryptocurrency miners whose resource usage is always the same.

Chapter 6

Conclusion

6.1 Main Contributions

As a result of this work, it was discovered that the Python Wave Module is sufficient for converting time series to sound and that sound characteristics can be used for application analysis. For example, distinguishing applications and discovering similarities or differences between nodes can be achieved. This work also includes a comparison table in which the frequency with the highest amplitude per node for each execution was recorded, allowing sound similarities and differences to be extracted through reading rather than listening. The comparison table was also visualized as an additional step. Overall, the findings show how audio-based analysis can be used to complement visual analysis.

In terms of confirming the findings, the study extended our knowledge of traditional statistical features by using a valid alternative, frequency (instead of e.g. minimum, maximum and average). Additionally, the discoveries will serve as a base for future studies aimed at recognizing applications based on execution behavior, similar to e.g. how Shazam recognizes songs.

By applying PCA to the strongest frequency components of chosen system metrics and plotting the result, further insights were exposed. It was discovered that some applications (FT, SP, BT, CoMD) repeat more consistently, while others (MG, LU, CG, miniGhost, miniAMR, miniMD, Kripke) have more variation and generate different system metric waveform for repeated executions.

To summarize, the main contribution of this work is the ability to listen to HPC Monitoring data with audification and a systematic frequency-based analysis of multiple system metrics.

6.2 Future Work

Because the Python Wave Module contains parameters that have not been investigated in depth, further research on this topic is recommended. The parameter channels, for instance, can be explored in terms of multi-channel sound system settings: mono, stereo, and surround sound. This enables us to have a single audio file with different sounds on each speaker depending on the settings we select. To demonstrate how surround sound is used in real life: Sound can be used to simulate a car driving by in a movie.

This concept could be applied to applications running on multiple nodes. The experiment could be carried out in a room with speakers set up in each corner to see if we can hear different or similar sound coming from each corner.

When considering Shazam, the newly acquired knowledge that applications can be distinguished can be put to use. Shazam is a tool that allows you to identify the title of a music song or the artist by playing the song into the tool. Rather than identifying songs, it can be explored if applications can be identified using an approach similar to Shazam.

Furthermore, as stated in Section 4.1.2, it is unknown why the investigated applications sound similar. Because the primary goal was to determine whether or not applications sounded comparable at all. A further study with more focus on the applications themselves is therefore recommended. Our work also opens further research on why some applications have more variation and generate different system metric waveform for repeated executions compared to others.

Bibliography

[1] 3D Sn deterministic particle transport, README [Online]. https://github.com/LLNL/Kripke [Accessed: 24.11.2021].

[2] 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. https://doi.org/10.6084/m9.figshare.6384248.v1 [Accessed: 14.09.2021], 2018.

[3] 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.

[4] NTi Audio. Fast Fourier Transformation FFT. https://www.nti-audio.com/de/service/wissen/fast-fourier-transformation-fft [Accessed: 24.11.2021].

[5] 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. Technical report, RNR-94-007, NASA Ames Research Center, Moffett Field, CA, 03, https://www.nas.nasa.gov/assets/pdf/techreports/1994/rnr-94-007.pdf[Accessed: 22.11.2021], 1994.

[6] Classical molecular dynamics proxy application, README [Online]. https://github.com/ECP-copa/CoMD [Accessed: 24.11.2021].

[7] Luuk Derksen. visualising high-dimensional datasets using pca and t-sne in python.

[8] NASA Advanced Supercomputing (NAS) Division. NAS Parallel Benchmarks. https://www.nas.nasa.gov/software/npb.html, [Accessed: 24.11.2021], 2021.

[9] Wikimedia Foundation. Spectrogram. https://en.wikipedia.org/wiki/Spectrogram [Accessed: 24.11.2021], 2021.

[10] 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.

[11] Charles Hoffman. Mono vs. Stereo Sound: The Difference Explained (With Audio Examples). https://www.blackghostaudio.com/blog/mono-vs-stereo-sound-the-difference-explained-with-audio-examples [Accessed: 17.11.2021], 2020.

[12] Federal Agencies Guidelines Initiative. Sampling rate (audio). http://www.digitizationguidelines.gov/term.php?term=samplingrateaudio [Accessed: 23.11.2021].

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

[14] The Python Standard Library. wave — Read and write WAV files. https://docs.python.org/3/library/wave.html [Accessed: 22.11.2021], 2021.

[15] Adaptive Mesh Refinement Mini-App, README [Online]. https://github.com/Mantevo/miniAMR [Accessed: 24.11.2021].

[16] MiniGhost Halo Exchange Mini-Application, README [Online]. https://github.com/Mantevo/miniGhost ([Accessed: 24.11.2021].

[17] miniMD molecular dynamics mini-app, README [online].

[18] Monika Multani. Statistical Characterization of HPC Monitoring Data. Bachelor Thesis, University of Basel, Faculty of Science, Department of Mathematics and Computer Science, 2021.

[19] Pacific Northwest Seismic Network. What is a spectrogram? https://pnsn.org/spectrograms/what-is-a-spectrogram [Accessed: 25.11.2021].

[20] Resonic Player. https://resonic.at/home [Accessed: 24.11.2021].

[21] Rowkin. Mono vs Stereo Sound: What's the big difference? https://www.rowkin.com/blogs/rowkin/mono-vs-stereo-sound-whats-the-big-difference [Accessed: 22.11.2021], 2019.

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

[23] Tableau. Time series analysis: Definition, types, techniques, and when it's used. https://www.tableau.com/learn/articles/time-series-analysis [Accessed: 24.11.2021].

[24] Paul Vickers and James L Alty. Using music to communicate computing information. *Interacting with computers*, 14(5):435–456, 2002.

[25] Chandu Yadav. Read and write WAV files using Python (wave). https://www.tutorialspoint.com/read-and-write-wav-files-using-python-wave [Accessed: 24.11.2021], 2019.

Appendices

Node Frequency Table

Ips	N ID	R 0	R 1	R 2	R 3	R 4	R 5	R 6	R 7	R 8	R 9	R 10	R 11	R 12	R 13	R 14	R 15	R 16	R 17	R 18	R 19	R 20	R 21	R 22	R 23	R 24	R 25	R 26	R 27	R 28	R 29
X	0	26	26	28	27	26	26	26	26	27	27	26	28	26	26	26	26	26	26	26	27	26	27	26	26	26	26	26	27	27	26
X	1	26	26	28	27	26	26	26	26	27	27	26	28	26	26	26	26	26	26	26	27	26	27	26	26	26	26	26	27	27	26
X	2	26	26	28	27	26	26	26	26	27	27	26	28	26	26	26	26	26	26	26	27	26	27	26	26	26	26	26	27	27	26
X	3	26	26	28	27	26	26	26	26	27	27	26	28	26	26	26	26	26	26	26	27	26	27	26	26	26	26	26	27	27	26
Y	0	84	83	84	83	84	84	84	83	84	82	82	82	81	84	84	84	84	84	84	83	83	83	83	82	82	83	85	84	85	84
Y	1	84	83	84	83	84	84	84	83	84	82	82	82	81	84	84	84	84	84	84	83	83	83	83	82	82	83	85	84	85	84
Y	2	84	83	84	83	84	84	84	83	84	82	82	82	81	84	84	84	84	84	84	83	83	83	83	82	82	83	85	84	85	84
Y	3	84	83	84	83	84	84	84	83	84	82	82	82	81	84	84	84	84	84	84	83	83	83	83	82	82	83	85	84	85	84
Z	0	168	167	167	168	169	168	167	169	168	169	169	168	169	167	169	167	169	169	167	168	168	167	168	169	167	169	168	170	168	168
Z	1	168	167	167	168	169	168	167	169	168	169	169	168	169	167	169	167	169	169	167	168	168	167	168	169	167	169	168	170	168	168
Z	2	168	167	167	168	169	168	167	169	168	169	169	168	169	167	169	167	169	169	167	168	168	167	168	169	167	169	168	170	168	168
Z	3	168	167	167	168	169	168	167	169	168	169	169	168	169	167	169	167	169	169	167	168	168	167	168	169	167	169	168	170	168	168

Table A1: Node Frequency Table SP

Ips	N ID	R 0	R 1	R 2	R 3	R 4	R 5	R 6	R 7	R 8	R 9	R 10	R 11	R 12	R 13	R 14	R 15	R 16	R 17	R 18	R 19	R 20	R 21	R 22	R 23	R 24	R 25	R 26	R 27	R 28	R 29
X	0	354	361	361	341	353	340	342	355	353	361	346	325	343	348	359	359	348	361	362	357	339	348	357	354	354	335	359	339	359	362
X	1	354	361	194	206	191	206	203	194	196	194	346	211	205	348	359	195	200	361	193	193	339	203	191	192	193	204	359	208	359	194
X	2	194	361	194	206	191	206	203	194	196	194	205	211	205	348	359	195	200	361	193	193	207	203	191	192	193	204	359	208	190	194
X	3	194	194	194	206	191	206	203	194	196	194	205	211	205	204	195	195	200	193	193	193	207	203	191	192	193	204	191	208	190	194
Y	0	174	176	171	173	170	172	172	156	175	167	166	171	154	156	163	171	158	169	165	172	174	166	174	172	173	167	165	169	171	171
Y	1	174	176	171	173	86	172	172	156	175	167	166	171	154	156	163	171	158	169	165	172	174	166	174	172	173	167	165	169	171	171
Y	2	174	176	171	173	86	172	172	78	175	167	166	171	77	156	163	171	158	169	165	172	174	166	174	172	86	167	165	169	171	171
Y	3	87	88	85	86	86	86	78	78	88	84	83	86	77	78	82	86	79	85	83	86	87	83	87	86	86	83	82	85	85	86
Z	0	251	249	65	250	250	78	79	247	243	250	235	251	79	84	249	250	248	250	245	250	249	246	250	242	235	251	249	239	249	249
Z	1	251	249	241	250	250	236	236	247	243	250	235	251	235	234	249	250	248	250	245	250	249	246	250	242	235	251	250	239	249	249
Z	2	251	249	241	250	250	236	236	247	243	250	235	251	235	234	249	250	248	250	245	250	249	246	250	242	235	251	250	239	249	249
Z	3	251	249	241	250	250	236	236	247	243	250	235	251	235	234	249	250	248	250	245	250	249	246	250	242	235	251	250	239	249	249

Table A2: Node Frequency Table LU

Ips	N ID	R 0	R 1	R 2	R 3	R 4	R 5	R 6	R 7	R 8	R 9	R 10	R 11	R 12	R 13	R 14	R 15	R 16	R 17	R 18	R 19	R 20	R 21	R 22	R 23	R 24	R 25	R 26	R 27	R 28	R 29
X	0	96	97	90	89	91	100	90	97	91	89	97	91	91	181	90	94	96	89	91	179	91	96	98	99	97	89	98	91	91	90
X	1	96	97	90	89	91	100	90	97	91	89	97	181	91	181	90	94	96	89	91	179	91	96	98	99	97	89	98	91	91	90
X	2	96	97	90	89	91	100	90	97	91	89	97	181	91	181	90	94	96	89	91	179	91	96	98	99	97	89	98	91	91	90
X	3	96	97	90	89	91	100	90	97	91	89	97	91	91	181	90	94	96	89	91	179	91	96	98	99	97	89	98	91	91	90
Y	0	103	105	110	111	111	107	101	111	104	111	110	111	111	111	111	106	104	104	111	111	111	111	112	111	104	112	111	113	111	111
Y	1	103	105	110	111	111	107	101	111	104	111	110	111	111	111	111	106	104	104	111	111	111	111	112	111	104	112	111	113	111	111
Y	2	103	105	110	111	111	107	101	111	104	111	110	111	111	111	111	106	104	104	111	111	111	111	112	111	104	112	111	113	111	111
Y	3	103	105	110	111	111	107	101	111	104	111	110	111	111	111	111	106	104	104	111	111	111	111	112	111	104	112	111	113	111	111
Z	0	223	212	211	222	211	211	212	211	210	211	211	210	211	219	219	224	219	211	211	216	211	211	224	218	226	221	218	211	219	211
Z	1	223	212	211	222	211	211	212	211	210	211	211	210	211	219	219	224	219	211	211	216	211	211	224	218	226	221	218	211	219	211
Z	2	223	212	211	211	222	211	212	211	210	211	211	210	211	219	219	224	219	211	211	216	211	211	224	218	226	221	218	211	219	211
Z	3	223	212	211	211	222	211	212	211	210	211	211	210	211	219	219	224	219	211	211	216	211	211	224	218	226	221	218	211	219	211

Table A3: Node Frequency Table BT

Ips	N ID	R 0	R 1	R 2	R 3	R 4	R 5	R 6	R 7	R 8	R 9	R 10	R 11	R 12	R 13	R 14	R 15	R 16	R 17	R 18	R 19	R 20	R 21	R 22	R 23	R 24	R 25	R 26	R 27	R 28	R 29
X	0	52	71	20	238	204	104	287	190	200	273	184	179	262	298	48	71	252	70	69	227	159	108	285	272	132	304	311	288	220	67
X	1	52	71	20	238	204	104	287	190	200	273	184	179	262	298	48	71	198	70	69	227	159	108	285	272	136	304	311	288	220	67
X	2	52	71	299	238	204	104	287	190	200	273	184	179	262	298	48	71	221	70	69	227	159	108	285	272	135	304	311	288	220	67
X	3	52	71	223	238	204	104	287	190	200	273	184	179	262	298	48	71	258	70	69	227	159	108	285	272	132	304	311	288	220	67
Y	0	242	287	255	175	271	214	265	268	167	110	64	27	108	259	227	52	291	63	66	99	285	108	127	170	226	225	231	222	94	182
Y	1	242	287	255	175	271	214	265	268	167	110	64	223	108	259	245	223	291	63	66	99	285	108	127	170	226	225	231	42	94	182
Y	2	242	287	255	175	271	214	265	268	167	110	64	223	108	259	227	224	291	63	66	99	285	108	127	170	226	225	231	42	94	182
Y	3	242	288	255	175	271	214	265	268	167	110	64	27	108	259	227	51	291	63	66	99	285	108	127	170	226	225	231	222	94	182
Z	0	298	169	259	28	132	120	63	231	287	50	71	91	282	99	89	320	313	233	166	64	232	175	190	289	170	207	297	238	308	320
Z	1	298	169	259	28	132	120	63	231	287	50	71	91	282	99	89	320	313	233	166	64	232	175	190	289	170	207	297	238	308	320
Z	2	298	169	259	28	132	120	63	231	287	50	71	91	282	99	89	320	313	233	166	64	232	175	190	289	170	207	297	238	308	320
Z	3	298	169	259	28	132	120	63	231	287	50	71	91	282	99	89	320	313	233	166	64	232	175	190	289	170	207	297	238	308	320

Table A4: Node Frequency Table CG

lps	N ID	R 0	R 1	R 2	R 3	R 4	R 5	R 6	R 7	R 8	R 9	R 10	R 11	R 12	R 13	R 14	R 15	R 16	R 17	R 18	R 19	R 20	R 21	R 22	R 23	R 24	R 25	R 26	R 27	R 28	R 29
X	0	286	249	246	207	227	288	195	263	328	283	277	285	351	287	286	299	281	285	285	171	397	185	287	96	282	264	262	225	138	237
X	1	234	181	246	207	227	288	195	286	247	277	277	285	292	287	286	299	281	285	285	342	397	185	287	292	283	264	262	225	138	237
X	2	347	181	246	415	227	288	195	327	247	277	277	285	334	287	286	299	281	285	285	171	397	185	287	292	283	264	262	225	291	237
X	3	234	181	246	250	227	144	195	327	328	277	277	285	292	287	286	299	281	285	285	342	397	185	287	334	283	264	262	225	138	237
Y	0	149	301	114	282	147	116	118	138	119	151	112	120	120	147	119	120	120	147	152	112	118	143	301	115	121	116	119	147	119	152
Y	1	149	301	114	141	147	118	118	138	119	151	112	150	120	118	149	120	120	147	122	112	118	143	301	115	121	116	119	147	119	152
Y	2	119	120	114	141	147	116	118	138	119	151	112	120	120	118	149	120	120	147	152	112	118	143	301	115	121	116	119	147	119	152
Y	3	149	120	114	141	147	118	118	138	119	151	112	120	120	118	119	120	120	147	122	112	118	143	301	115	121	116	119	147	297	152
Z	0	197	160	229	157	210	195	257	162	189	225	215	194	158	193	193	227	260	258	230	260	159	260	186	260	211	194	185	212	196	195
Z	1	263	187	196	184	184	260	257	189	162	226	188	226	185	225	194	227	227	226	197	227	159	194	213	227	211	194	212	212	228	260
Z	2	231	187	229	N/A	210	195	225	216	162	193	161	226	185	193	226	260	260	194	197	259	186	260	213	260	211	226	212	159	228	194
Z	3	263	187	326	184	210	228	225	216	162	258	214	227	184	258	226	227	259	194	197	227	186	260	213	227	158	194	212	185	195	227

Table A5: Node Frequency Table MG

lps	N ID	R 0	R 1	R 2	R 3	R 4	R 5	R 6	R 7	R 8	R 9	R 10	R 11	R 12	R 13	R 14	R 15	R 16	R 17	R 18	R 19	R 20	R 21	R 22	R 23	R 24	R 25	R 26	R 27	R 28	R 29
X	0	233	233	230	234	232	234	235	231	233	235	233	233	236	233	236	235	237	234	232	236	236	233	232	233	236	232	232	232	233	236
X	1	233	233	230	234	232	234	235	231	233	235	233	233	236	233	236	235	237	234	232	236	236	233	232	233	236	232	232	232	233	236
X	2	233	233	230	234	232	234	235	231	233	235	233	233	236	233	236	235	237	234	232	236	236	233	232	233	236	232	232	232	233	236
X	3	233	233	230	234	232	234	235	231	233	235	233	233	236	233	236	235	237	234	232	236	236	233	232	233	236	232	232	232	233	236
Y	0	169	170	169	169	170	170	170	170	170	170	169	171	171	170	170	171	171	171	170	171	170	170	170	170	170	169	171	169	170	170
Y	1	169	170	169	169	170	170	170	170	170	170	169	171	171	170	170	171	171	171	170	171	170	170	170	170	170	169	171	169	170	170
Y	2	169	170	169	169	170	170	170	170	170	170	169	171	171	170	170	171	171	171	170	171	170	170	170	170	170	169	171	169	170	170
Y	3	169	170	169	169	170	170	170	170	170	170	169	171	171	170	170	171	171	171	170	171	170	170	170	170	170	169	171	169	170	170
Z	0	228	228	228	228	226	228	227	229	228	228	228	229	228	227	228	228	228	228	228	227	229	226	228	228	227	228	230	228	227	227
Z	1	228	228	228	228	226	228	227	229	228	228	228	229	228	227	228	228	228	228	228	227	229	226	228	228	227	228	230	228	227	227
Z	2	228	228	228	228	226	228	227	229	228	228	228	229	228	227	228	228	228	228	228	227	229	226	228	228	227	228	230	228	227	227
Z	3	228	228	228	228	226	228	227	229	228	228	228	229	228	227	228	228	228	228	228	227	229	226	228	228	227	228	230	228	227	227

Table A6: Node Frequency Table CoMD

Ips	N ID	R 0	R 1	R 2	R 3	R 4	R 5	R 6	R 7	R 8	R 9	R 10	R 11	R 12	R 13	R 14	R 15	R 16	R 17	R 18	R 19	R 20	R 21	R 22	R 23	R 24	R 25	R 26	R 27	R 28	R 29
X	0	116	125	127	115	126	128	126	123	132	130	123	122	130	112	127	125	124	129	126	130	128	127	129	116	127	108	122	124	128	126
X	1	116	125	127	115	126	128	126	123	132	130	124	122	130	112	127	125	124	129	126	130	128	127	129	116	127	108	121	124	128	126
X	2	116	125	127	115	126	128	126	123	132	130	124	122	130	112	127	125	124	129	126	130	128	127	129	116	127	108	122	124	128	126
X	3	116	125	127	116	126	128	126	123	132	130	123	122	130	112	127	125	124	129	126	130	128	127	129	116	127	108	122	124	128	126
Y	0	101	73	93	102	63	58	103	74	56	74	96	64	70	64	59	83	103	58	75	101	45	62	60	63	103	63	103	102	58	60
Y	1	101	73	93	102	63	58	104	74	56	74	96	64	70	64	59	83	103	58	75	101	260	62	60	63	103	63	103	102	58	60
Y	2	101	73	93	102	63	58	103	74	56	74	96	64	70	64	59	83	103	58	75	101	51	62	60	63	103	63	103	102	58	60
Y	3	101	73	93	102	63	58	103	74	56	74	96	64	70	64	59	83	103	58	75	101	45	62	60	63	103	63	103	102	58	60
Z	0	69	89	92	85	91	70	85	86	76	87	71	89	85	73	84	77	88	92	78	75	85	73	92	87	83	85	86	84	82	73
Z	1	70	89	92	85	91	70	85	86	76	87	71	88	85	73	84	77	87	92	78	75	86	73	92	87	83	85	86	84	82	73
Z	2	139	89	92	85	91	70	85	86	76	87	71	88	85	73	84	77	88	92	78	75	85	73	92	87	83	85	86	84	82	73
Z	3	70	89	92	85	91	70	85	86	76	87	71	88	85	73	84	77	88	92	78	75	85	73	92	87	83	85	86	84	82	73

Table A7: Node Frequency Table miniGhost

Ips	N ID	R 0	R 1	R 2	R. 3	R 4	R 5	R 6	R 7	R 8	R 9	R 10	R 11	R 12	R 13	R 14	R 15	R 16	R 17	R 18	R 19	R 20	R 21	R 22	R 23	R 24	R 25	R 26	R 27	R 28	R 29
X	0	116	125	127	115	126	128	126	123	132	130	123	122	130	112	127	125	124	129	126	130	128	127	129	116	127	108	122	124	128	126
X	1	116	125	127	115	126	128	126	123	132	130	124	122	130	112	127	125	124	129	126	130	128	127	129	116	127	108	121	124	128	126
X	2	116	125	127	115	126	128	126	123	132	130	124	122	130	112	127	125	124	129	126	130	128	127	129	116	127	108	122	124	128	126
X	3	116	125	127	116	126	128	126	123	132	130	123	122	130	112	127	125	124	129	126	130	128	127	129	116	127	108	122	124	128	126
Y	0	101	73	93	102	63	58	103	74	56	74	96	64	70	64	59	83	103	58	75	101	45	62	60	63	103	63	103	102	58	60
Y	1	101	73	93	102	63	58	104	74	56	74	96	64	70	64	59	83	103	58	75	101	260	62	60	63	103	63	103	102	58	60
Y	2	101	73	93	102	63	58	103	74	56	74	96	64	70	64	59	83	103	58	75	101	51	62	60	63	103	63	103	102	58	60
Y	3	101	73	93	102	63	58	103	74	56	74	96	64	70	64	59	83	103	58	75	101	45	62	60	63	103	63	103	102	58	60
Z	0	69	89	92	85	91	70	85	86	76	87	71	89	85	73	84	77	88	92	78	75	85	73	92	87	83	85	86	84	82	73
Z	1	70	89	92	85	91	70	85	86	76	87	71	88	85	73	84	77	87	92	78	75	86	73	92	87	83	85	86	84	82	73
Z	2	139	89	92	85	91	70	85	86	76	87	71	88	85	73	84	77	88	92	78	75	85	73	92	87	83	85	86	84	82	73
Z	3	70	89	92	85	91	70	85	86	76	87	71	88	85	73	84	77	88	92	78	75	85	73	92	87	83	85	86	84	82	73

Table A8: Node Frequency Table miniAMR

Ips	N ID	R 0	R 1	R 2	R 3	R 4	R 5	R 6	R 7	R 8	R 9	R 10	R 11	R 12	R 13	R 14	R 15	R 16	R 17	R 18	R 19	R 20	R 21	R 22	R 23	R 24	R 25	R 26	R 27	R 28	R 29
X	0	1	1	204	1	1	N/A	1	1	1	N/A	1	1	N/A	N/A	N/A	2	1	1	N/A	1	1	1	1	1	1	1	1	N/A	1	N/A
X	1	1	1	204	1	1	N/A	1	1	1	N/A	1	1	N/A	N/A	N/A	2	1	1	N/A	1	1	1	1	1	1	1	1	N/A	1	N/A
X	2	1	1	204	1	1	N/A	1	1	1	N/A	1	1	N/A	N/A	N/A	2	1	1	N/A	1	1	1	1	1	1	1	1	N/A	1	N/A
X	3	1	1	204	1	1	N/A	1	1	1	N/A	1	1	N/A	N/A	N/A	2	1	1	N/A	1	1	1	1	1	1	1	1	N/A	1	N/A
Y	0	139	138	138	138	138	138	277	138	138	137	277	138	139	275	138	275	138	137	139	139	138	137	138	275	138	138	138	139	137	138
Y	1	139	138	138	138	138	138	277	138	138	137	277	138	139	275	138	275	138	275	139	139	138	137	138	275	138	138	138	139	137	138
Y	2	277	138	138	138	138	138	277	138	138	137	277	138	139	275	138	137	138	275	139	139	138	137	138	275	138	138	138	139	137	138
Y	3	139	138	138	138	138	138	277	138	138	137	277	138	139	275	138	137	138	275	139	139	138	137	138	275	138	138	138	139	137	138
Z	0	44	46	46	46	44	46	46	44	46	44	46	44	44	46	46	44	46	46	44	46	44	44	46	46	46	44	44	46	46	46
Z	1	44	46	46	46	44	46	46	44	46	44	46	44	44	46	46	44	46	312	44	46	44	44	46	46	46	44	44	46	46	46
Z	2	44	46	46	46	44	46	46	44	46	44	46	44	44	46	46	44	46	46	44	46	44	44	46	46	46	44	44	46	46	46
Z	3	44	46	46	46	44	46	46	44	46	44	46	44	44	46	46	44	46	46	44	46	44	44	46	46	46	44	44	46	46	46

Table A9: Node Frequency Table miniMD

Ips	N ID	R 0	R 1	R 2	R 3	R 4	R 5	R 6	R 7	R 8	R 9	R 10	R 11	R 12	R 13	R 14	R 15	R 16	R 17	R 18	R 19	R 20	R 21	R 22	R 23	R 24	R 25	R 26	R 27	R 28	R 29
X	0	1	1	204	1	1	N/A	1	1	1	N/A	1	1	N/A	N/A	N/A	2	1	1	N/A	1	1	1	1	1	1	1	1	N/A	1	N/A
X	1	1	1	204	1	1	N/A	1	1	1	N/A	1	1	N/A	N/A	N/A	2	1	1	N/A	1	1	1	1	1	1	1	1	N/A	1	N/A
X	2	1	1	204	1	1	N/A	1	1	1	N/A	1	1	N/A	N/A	N/A	2	1	1	N/A	1	1	1	1	1	1	1	1	N/A	1	N/A
X	3	1	1	204	1	1	N/A	1	1	1	N/A	1	1	N/A	N/A	N/A	2	1	1	N/A	1	1	1	1	1	1	1	1	N/A	1	N/A
Y	0	139	138	138	138	138	138	277	138	138	137	277	138	139	275	138	275	138	137	139	139	138	137	138	275	138	138	138	139	137	138
Y	1	139	138	138	138	138	138	277	138	138	137	277	138	139	275	138	275	138	275	139	139	138	137	138	275	138	138	138	139	137	138
Y	2	277	138	138	138	138	138	277	138	138	137	277	138	139	275	138	137	138	275	139	139	138	137	138	275	138	138	138	139	137	138
Y	3	139	138	138	138	138	138	277	138	138	137	277	138	139	275	138	137	138	275	139	139	138	137	138	275	138	138	138	139	137	138
Z	0	44	46	46	46	44	46	46	44	46	44	46	44	44	46	46	44	46	46	44	46	44	44	46	46	46	44	44	46	46	46
Z	1	44	46	46	46	44	46	46	44	46	44	46	44	44	46	46	44	46	312	44	46	44	44	46	46	46	44	44	46	46	46
Z	2	44	46	46	46	44	46	46	44	46	44	46	44	44	46	46	44	46	46	44	46	44	44	46	46	46	44	44	46	46	46
Z	3	44	46	46	46	44	46	46	44	46	44	46	44	44	46	46	44	46	46	44	46	44	44	46	46	46	44	44	46	46	46

Table A10: Node Frequency Table Kripke

Visualization of Node Frequency Table

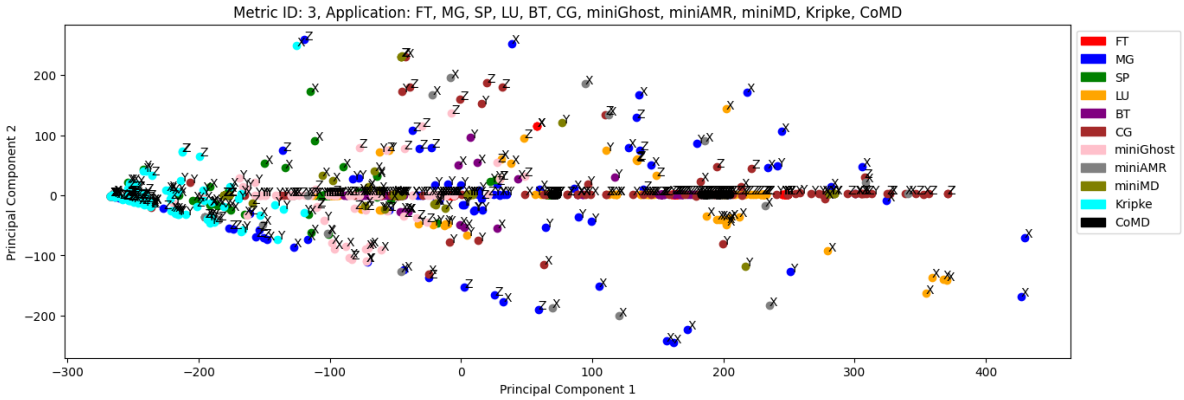

Figure A1: AR_NIC_RSPMON_PARB_EVENT_CNTR_AMO_BLOCKED_metric_set_nic

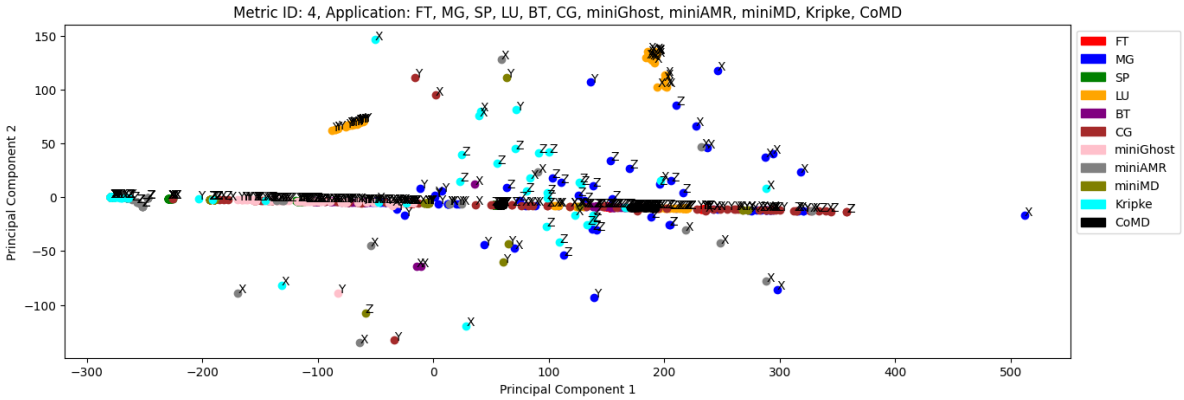

Figure A2: AR_NIC_RSPMON_PARB_EVENT_CNTR_AMO_FLITS_metric_set_nic

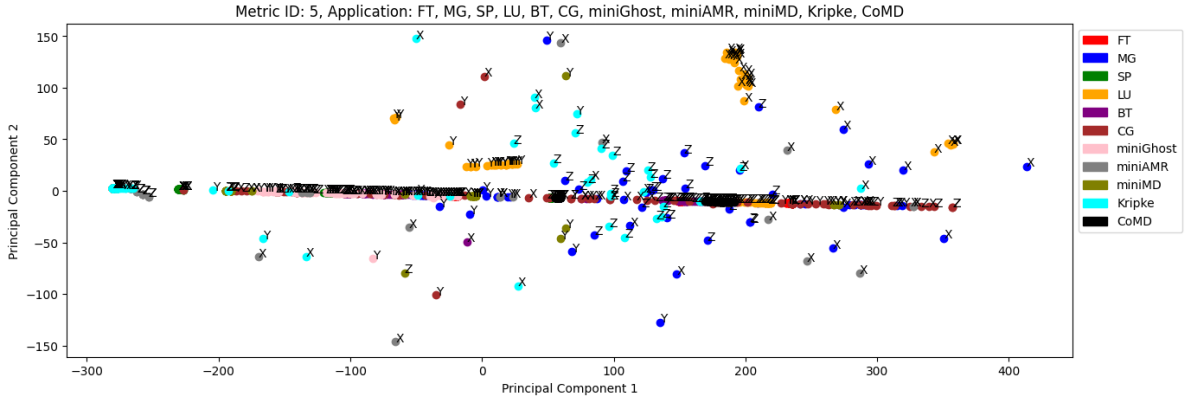

Figure A3: AR_NIC_RSPMON_PARB_EVENT_CNTR_AMO_PKTS_metric_set_nic

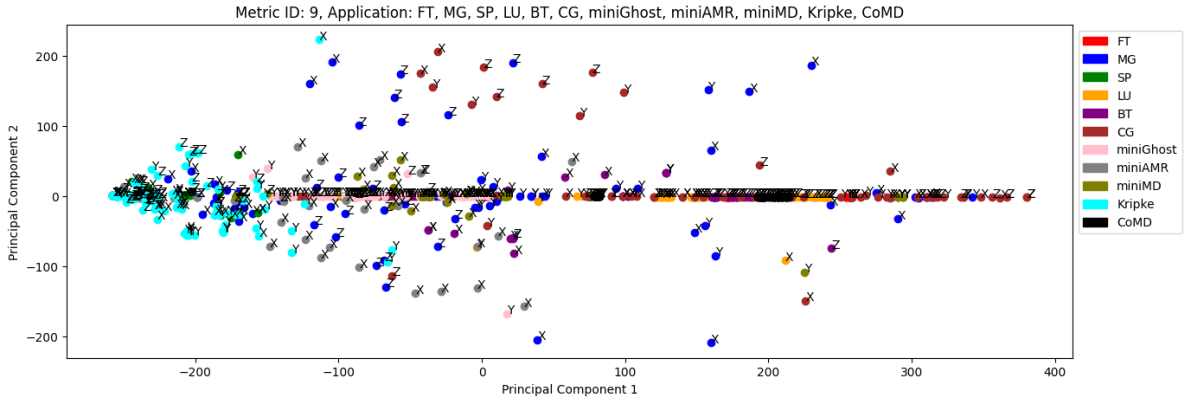

Figure A4: AR_NIC_RSPMON_PARB_EVENT_CNTR_IOMMU_BLOCKED_metric_set_nic

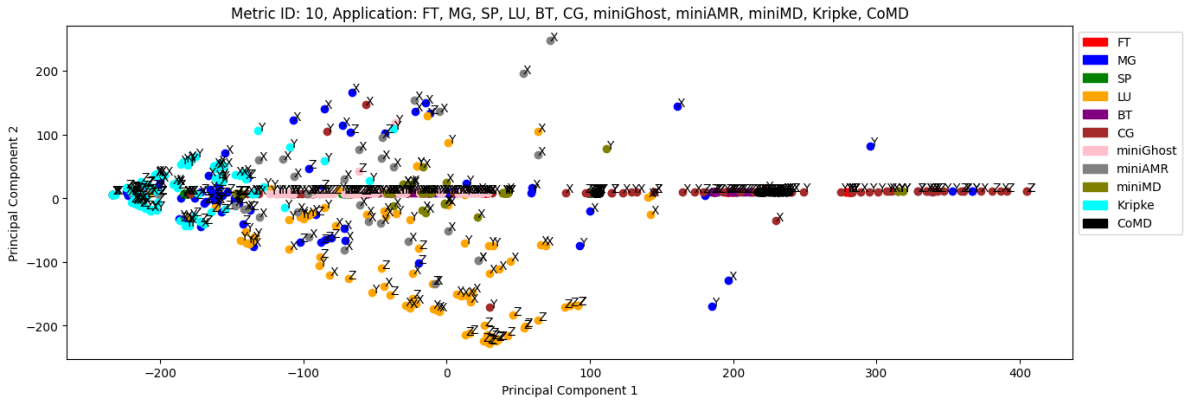

Figure A5: AR_NIC_RSPMON_PARB_EVENT_CNTR_IOMMU_FLITS_metric_set_nic

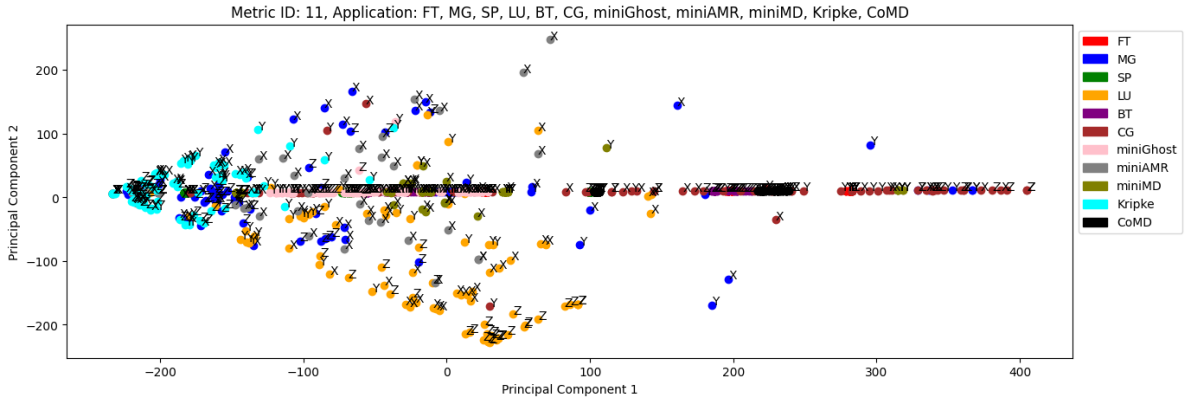

Figure A6: AR_NIC_RSPMON_PARB_EVENT_CNTR_IOMMU_PKTS_metric_set_nic

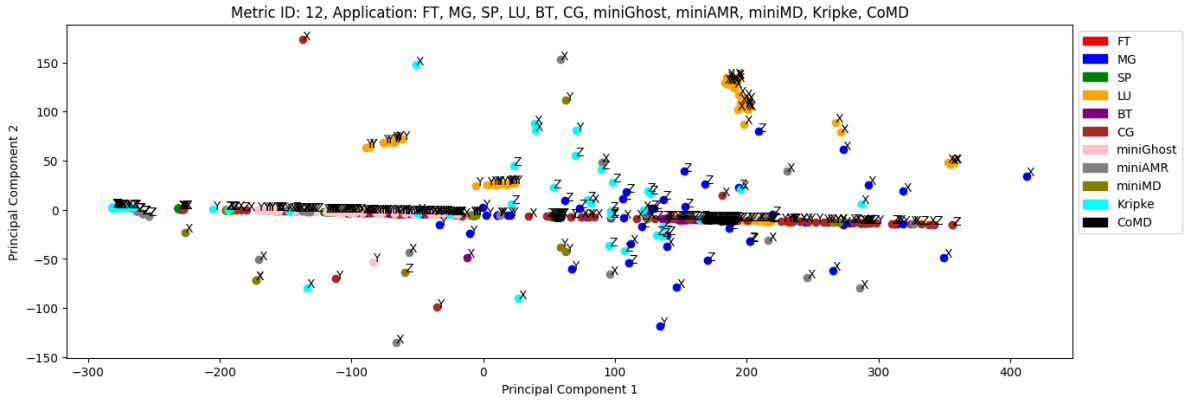

Figure A7: AR_NIC_RSPMON_PARB_EVENT_CNTR_PI_FLITS_metric_set_nic

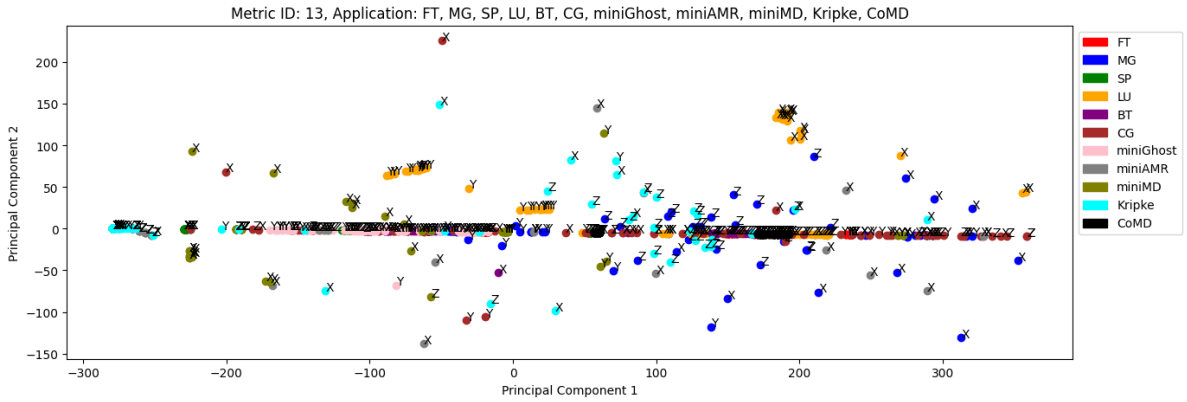

Figure A8: AR_NIC_RSPMON_PARB_EVENT_CNTR_PI_PKTS_metric_set_nic

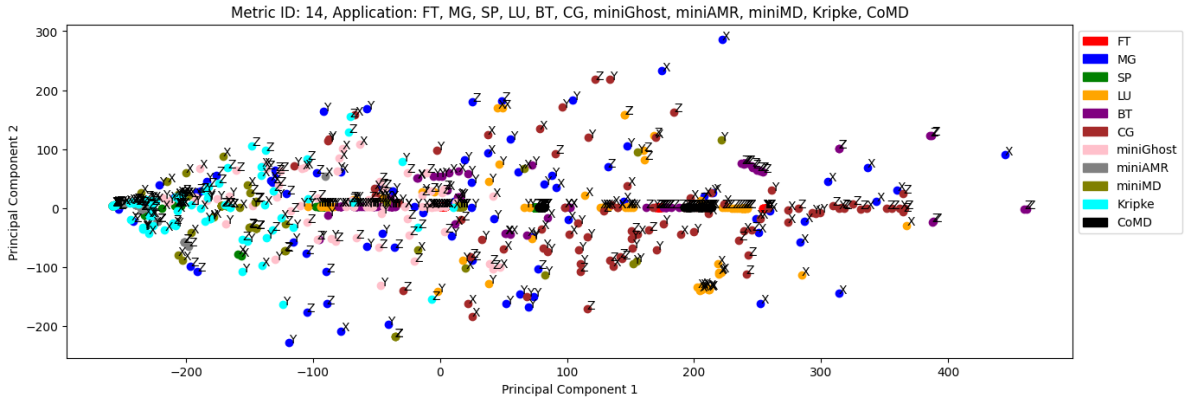

Figure A9: AR_NIC_RSPMON_PARB_EVENT_CNTR_PI_STALLED_metric_set_nic

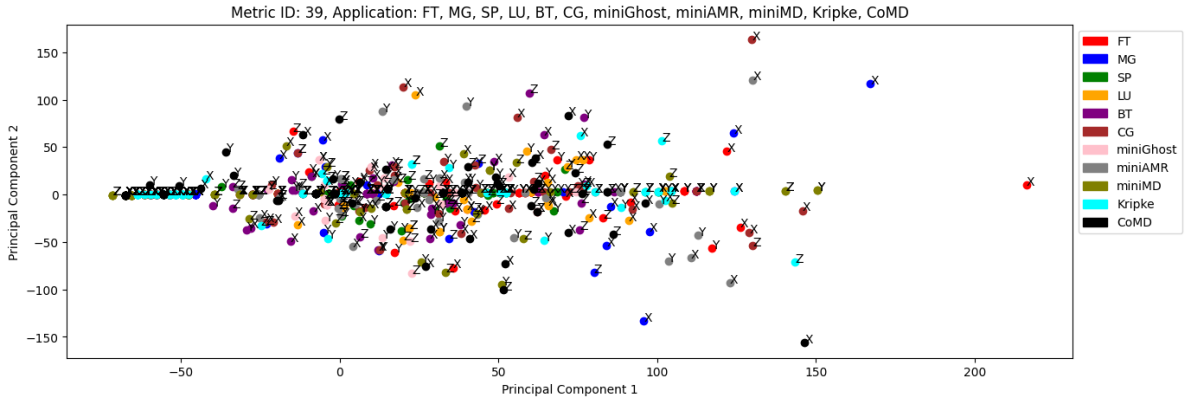

Figure A10: SMSG_nrx_cray_aries_r

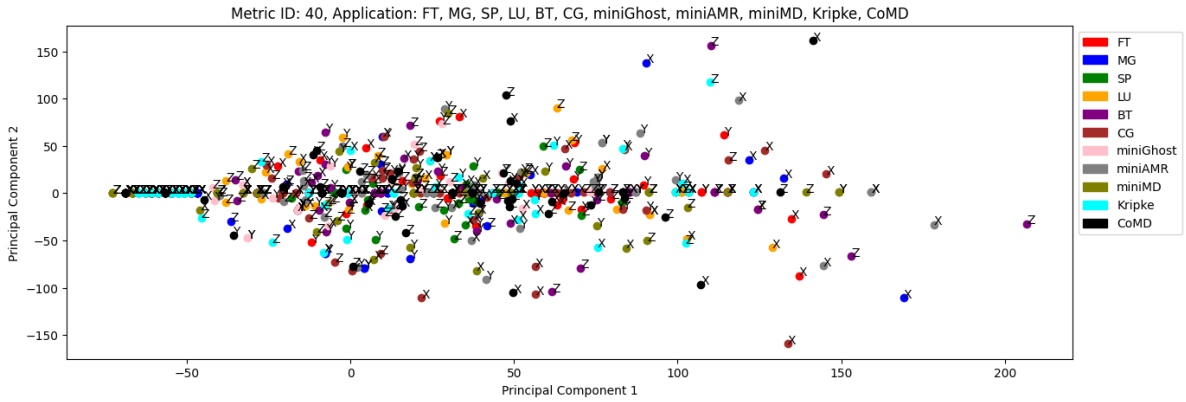

Figure A11: SMSG_ntx_cray_aries_r

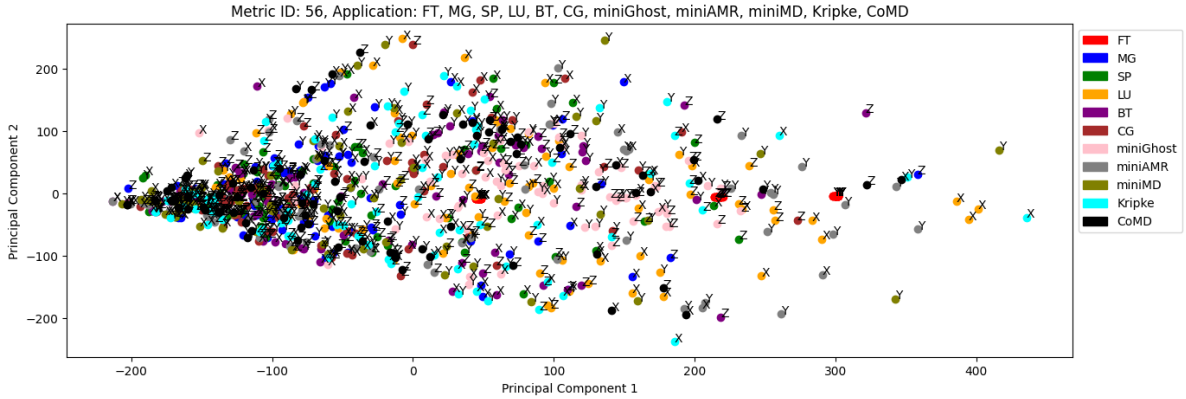

Figure A12: energy(J)_cray_aries_r

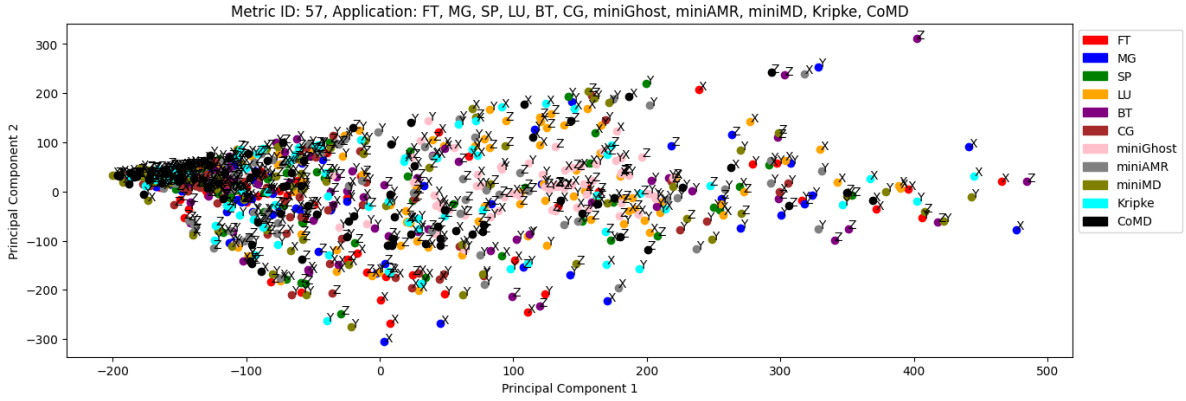

Figure A13: freshness_cray_aries_r

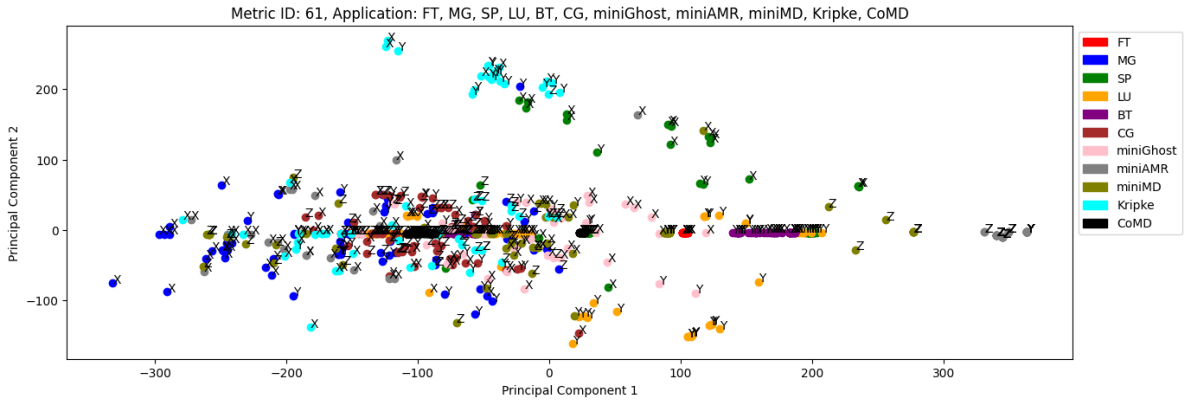

Figure A14: hwintr_count_procstat

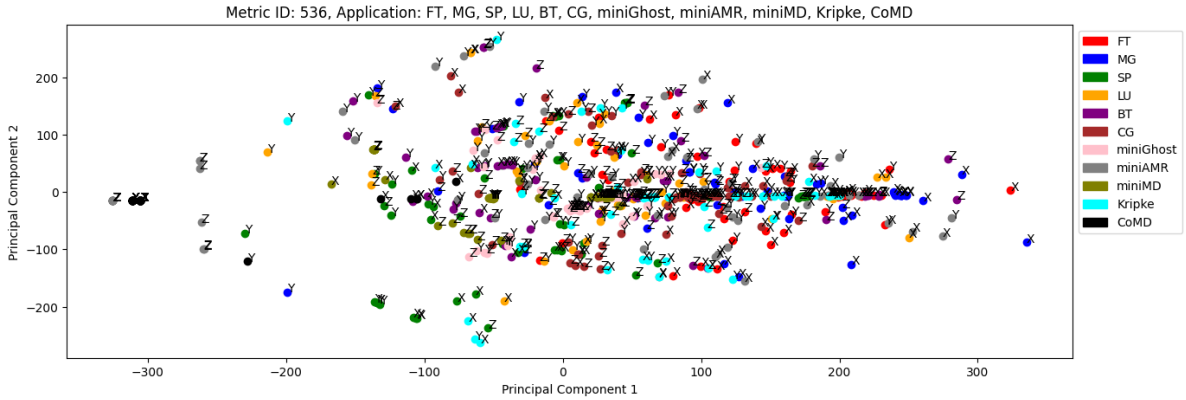

Figure A15: pgalloc_normal_vmstat

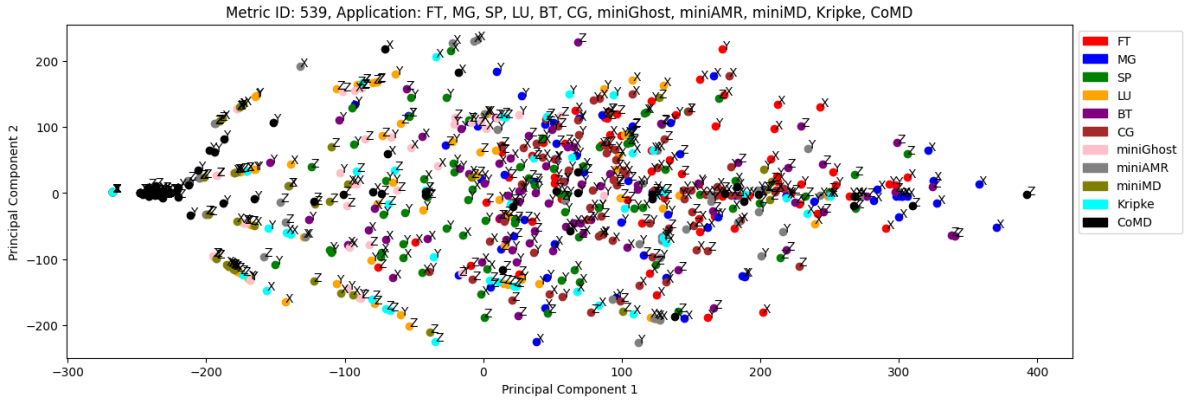

Figure A16: pgfree_vmstat

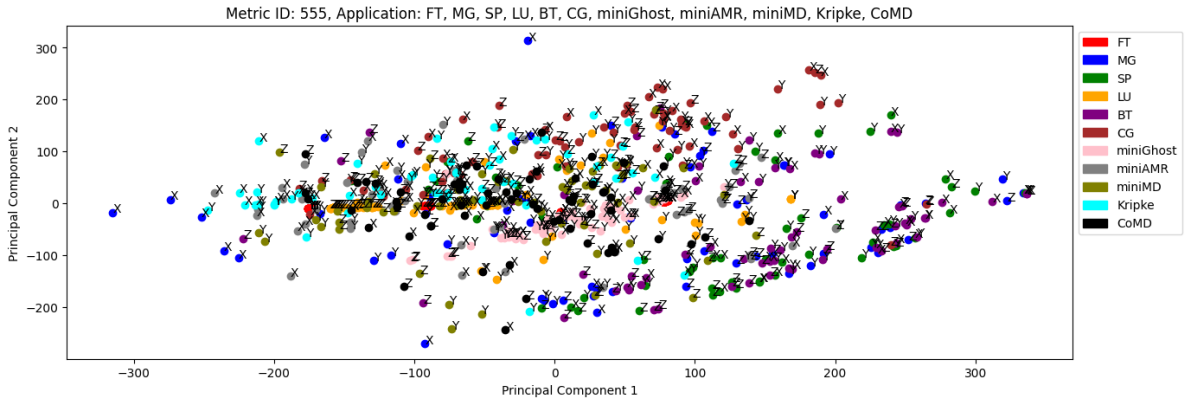

Figure A17: softirq_count_procstat

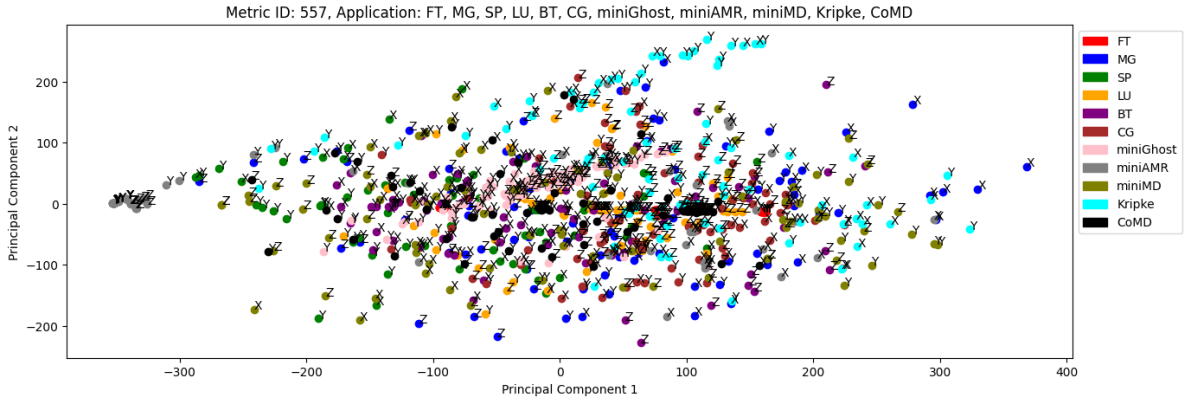

Figure A18: sys_procstat

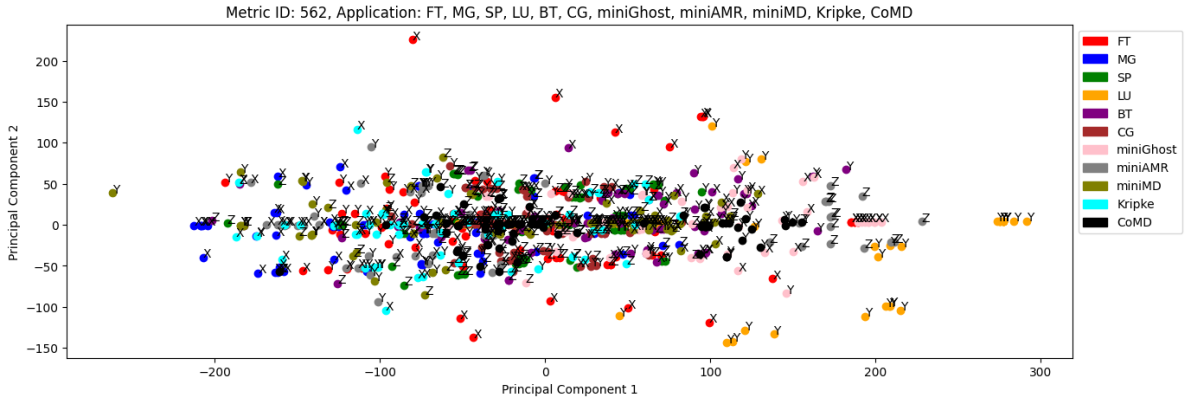

Figure A19: user_procstat

University of Basel

Faculty of Science

Declaration on Scientific Integrity

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

Title of Thesis: Analysis of Individual HPC System Metrics through Audification

Name Assesor: Prof. Dr. Florina M. Ciorba

Name Student: Li Ting Luong

Matriculation No.: 18-050-724

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: Muttenz, 29.11.2021 Student: L. Luong

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: Muttenz, 29.11.2021 Student: L. Luong

Place, Date: _____ Assessor: _____

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