Automated performance profiling of software applications

Shamoona Imtiaz
AUTOMATED PERFORMANCE PROFILING
OF SOFTWARE APPLICATIONS

Shamoona Imtiaz

2023

School of Innovation, Design and Engineering
Abstract

For industrial systems performance, it is desired to keep the IT infrastructure competitive through the efficient use of computer resources. However, modern software applications are complex and often utilize a broad spectrum of available hardware resources. The way how applications utilize these resources may vary from platform to platform due to the different architectural features, requirements and performance levels guaranteed by the hardware as well as due to the type of application under analysis. It becomes challenging to predict how the deployed applications will perform on a particular platform, how to improve the hardware resource utilization, and how to meet the Quality of Service (QoS) requirements.

Computers these days enable us to precisely trace down the performance of applications using the Performance Monitoring Counters (PMCs) available in the Performance Monitoring Unit (PMU) of the processors. PMCs can record micro-architectural events, called PMU events, at the CPU cycle level. Tools like perf API and PAPI provide performance information using manual and selective function calls. Nevertheless, it is difficult for humans to make analyses, visualize performance over time and draw conclusions from this wealth of data without automatic and intelligent tools.

In this thesis, our first contribution is to propose a cross-platform automated approach to investigate the overall performance profile of the applications. Instead of relying on a static and pre-selected list of hardware and software performance events we avoid the selection bias by capturing the entire range of performance events specific to the platform on which applications are running.

The performance data being generated from shared resource environments and hierarchical resource utilization demands makes it harder to represent the
behavior in one model. That being the case, it was deemed appropriate to
demonstrate the compact representation of behavior. So, our next contribution is
to present a simplified model to understand the behavior of performance events.
Therefore, we determine segments in performance data by locating the points in
their data distribution using the change point detection method. The proposed
solution reduces the complexity of data handling, allows the application of
further statistical analyses and provides better visualization.

Lastly, to reveal the out-of-sight information, we present a customized
approach to automatically identify the groups of similar performance events
based on the change in their behavior. There can be several ways to group the
performance data, we opt to form the groups based on change points in the
behavior of the performance events. The knowledge can then be used by the
decision-makers as per their interests such as for load balancing, deployments,
scheduling and anomalous behavior detection.
Sammanfattning

Inom system som kräver hög prestanda är önskan att hålla IT-infrastrukturen konkurrenskraftig genom effektiv användning av datorresurser. Moderna mjukvaruapplikationer är komplexa och använder ofta ett brett spektrum av tillgängliga hårdvaruresurser. En applikations resursutnyttjande beteende kanske inte är detsamma eftersom de olika hårdvara arkitekturerna, hårdvarustödet och mängden information en maskin kan bearbeta samtidigt kan variera från en plattform till annan. Det blir en utmaning att prediktera hur de utplacerade applikationerna kommer att prestera på en viss plattform, vad är det bättre utnyttjandet av hårdvaruresurser och hur man uppfyller kvalitetskraven (QoS).

Moderna datorer nuförtiden möjliggör att vi kan spåra applikationernas prestanda exakt med hjälp av Performance Monitoring Counters (PMC) som finns tillgängliga i processorernas Performance Monitoring Unit (PMU). Verktygen som perf API och PAPI tillhandahåller prestandainformation med hjälp av manuella och selektiva funktionsanrop. Ändå är det svårt för det människan att analysera, visualisera prestanda över tid och dra slutsatser från denna mängd data utan tillgång till automatiska och intelligenta verktyg.

I den här avhandlingen är vårat första bidrag att föreslå ett lösning som undersöker automatiskt applikationernas övergripande prestandaprofil. Istället för att förlita oss på en statisk och förvald lista över hårdvaru- och mjukvaruprestandahändelser undviker vi urvalsbias genom att fånga upp hela utbudet av PMU-händelser som är specifika för plattformen där applikationer körs.

Prestandadata som genereras från delade resursmiljöer gör det svårare att representera beteendet i en modell. Så vårt nästa bidrag är att presentera en förenklad modell för att förstå beteendet hos prestandahändelser. Därför föreslår
ett automatiserat tillvägagångssätt genom att segmentera prestandadata i mindre dataserier och tillhandahålla en statistisk modell för varje segment i stället för hela spektrumet. Den föreslagen lösningen ger fördelar som minskad komplexitet för datahantering, tillämpning av ytterligare statistiska analyser och bättre visualisering.

Slutligen, för att avslöja den osynliga informationen, presenterar vi ett anpassat tillvägagångssätt för att automatiskt identifiera grupper av liknande prestationshändelser baserat på förändringen i deras beteende. Det kan finnas flera sätt att gruppera prestandadatala, vi väljer att bilda grupperna baserat på förändringspunkter i beteendehändelserna. Kunskapen kan sedan användas av beslutsfattarna enligt deras intressen till exempel lastbalansering, schemaläggning och upptäckt av avvikande beteende.
To my family
Acknowledgment

Confucius formulated that ‘We have two lives, and the second begins when we realize we only have one’. Moving to Sweden and starting my Ph.D. was like leaving the uncertainty behind by empowering myself with higher education. So first of all, I would like to express my gratitude to my first teacher, my father, who taught me how to face life. Thank you Abu G, your lessons never get old.

Thanks to MDU and all my supervisors for providing me with the best-suited environment to achieve this goal. Thank you Moris Behnam for your continuous support and for fostering my passion for cybersecurity with various opportunities. You have the eye and openness to shape a talent. Thank you Jan Carlson for your instant and valuable feedback, you have been like a catalyst throughout my mentorship. Thank you Gabriele Capannini for inspiring this journey with your mathematical skills and problem-solving ideas, you always have a way forward. Thank you Marcus Jägemar for always appreciating small steps to achieve bigger goals, your encouragement brought me the confidence to seek bigger challenges.

I deeply appreciate Jakob Danielsson for your help in onboarding my Ph.D. journey. I would also like to express my sincere gratitude to Robbert Jongeling for being so kind, helpful and approachable in providing quick information than the internal portal. I am also thankful to my colleagues and peers for the cheerful times at Java for lunches and Fika.

Finally, I would like to extend my gratitude to my husband for holding me through the ups and downs of life, to my brother for bringing all the positivity in me, to all my siblings for always being supportive and to my parents for their unconditional love and hard work they had done to make us strong and good.
individuals. Most importantly, to my kids, Saleh and Abdullah, to restore my energy on my way back home, You Are Life.

Shamoona Imtiaz
Västerås, October, 2023
List of Publications

Papers included in this thesis


---

1 The included papers have been reformatted to comply with the thesis layout.
Related publications, not included in this thesis

# Contents

## I Thesis 1

1 Introduction 3

2 Research Overview 7

2.1 Problem Overview 7

2.2 Research Goals 8

2.2.1 Research Method 10

2.2.2 Research Approach 10

3 Background and Related Work 13

3.1 Performance 13

3.2 Performance Monitoring Counters 14

3.3 Change Point Detection 16

3.4 Sequence Similarity - Similar Is Not Same 18

3.5 Bounded Cost Function 19

3.6 Related work 21

4 Research Results 25

4.1 Thesis Contributions 25

4.1.1 C1: An automated cross-platform mechanism to capture the overall performance of the applications 25

4.1.2 C2: An automated mechanism to identify the most relevant PMU events related to performance 27
4.1.3 C3: An automated approach to present a compact representation of complex performance events based on statistical methods .......................... 27
4.1.4 C4: An automated approach to group up similar performance events based on a measure of similarity ........................................ 28
4.2 Included papers .............................................................. 28
4.2.1 Paper A ................................................................. 29
4.2.2 Paper B ................................................................. 30
4.2.3 Paper C ................................................................. 30
4.3 Publications not included in the thesis ................................. 31
4.3.1 Paper X ................................................................. 32

5 Conclusion and Future Work .............................................. 33
5.1 Future Work ............................................................... 34

Bibliography ................................................................. 37

II Included Papers .......................................................... 43

6 Paper A: Automatic Platform-Independent Monitoring and Ranking of Hardware Resource Utilization ........................................... 45
6.1 Introduction .............................................................. 47
6.2 Background ............................................................ 49
  6.2.1 Performance Monitoring .......................................... 49
  6.2.2 Performance Monitoring Unit .................................. 50
  6.2.3 Perf and PAPI ...................................................... 51
  6.2.4 Multiplexing ....................................................... 53
6.3 Methodology ............................................................ 53
  6.3.1 Event fetch ......................................................... 54
  6.3.2 Application Characterization using Multiplexing ......... 55
  6.3.3 Ranking Events .................................................. 56
6.4 Implementation ......................................................... 57
6.5 Experiments ............................................................. 58
6.6 Discussion ................................................. 62
6.7 Conclusion ............................................... 64
6.8 Related work ............................................. 64
6.9 Future Work .............................................. 65
Bibliography .................................................. 67

7 Paper B:  
Automatic Segmentation of Resource Utilization Data  71
7.1 Introduction ............................................... 73
7.2 Background ................................................ 75
  7.2.1 Performance Monitoring Counters .................... 75
  7.2.2 Change Points Detection ......................... 76
  7.2.3 Statistical Methods ................................. 77
7.3 Proposed Solution ........................................ 77
  7.3.1 Segment Detection ................................. 78
  7.3.2 Segment-wise Statistical Model .................... 79
7.4 Implementation and Experiments ....................... 81
7.5 Discussion ............................................... 83
7.6 Related Work ............................................ 85
7.7 Conclusion and Future Work ............................ 85
Bibliography .................................................. 87

8 Paper C:  
Automatic Clustering of Performance Events  89
8.1 Introduction ............................................... 91
8.2 Background ............................................... 93
  8.2.1 Performance Monitoring Counters .................. 94
  8.2.2 Change Point Detection ......................... 94
  8.2.3 Sequence Similarity - Similar Is Not Same ........ 95
  8.2.4 Bounded Cost Function ............................ 96
  8.2.5 Clustering Analysis ............................... 96
8.3 Proposed Solution ....................................... 97
  8.3.1 Similarity Detection ............................... 97
  8.3.2 Group Identification .............................. 102
8.4 Implementation and Experiments ....................... 103
8.5  Discussion ....................................................... 107
8.6  Related Work .................................................... 108
8.7  Conclusion and Future Work ................................. 109
Bibliography .......................................................... 111
I

Thesis
Chapter 1

Introduction

To reduce human intervention and efforts, technology-driven solutions are on the rise. This adherence to automation has led to rapid scaling of applications consequently transforming the world into a smaller place (a global village [1]) which is a complex digital system. This increased complexity of software and hardware architecture affects the efficiency of deployment, creates difficulties in predicting the system’s behavior, makes it hard to optimize the systems against better usage of hardware, increases the power consumption, impacts the quality of service, influences the user experience and increases the cybersecurity risks. As a countermeasure to complexity, the community has proposed different approaches such as modularity [2, 3], abstraction, reusability [3], multilevel approach [4], standardization, and simplification [5]. These approaches are good design principles but achieving the desired functionality and better insight into hardware (multilevel cache, pipeline, internal busses, etc.) and software utilization is still a challenge due to complex interactions, diverse technologies, non-linear relationships, dynamicity, scale and size. To gather these insights from the designer and administrative perspective, load analysis and resource analysis are performed respectively by monitoring the performance of the systems [6].

System performance monitoring is categorized into processor utilization, memory usage, disk activity and network usage [6]. That implies it is related to the resource utilization demand of the software running over hardware. In
general, modern applications have complex resource utilization behavior such as in multicore systems a simultaneous demand for a shared cache can directly impact the system performance and can provide a variable user experience in a shared resource environment. To achieve the optimization goals it becomes interesting to know which hardware resources an application is utilizing the most, what hardware specifications can serve the demands efficiently, and how a resource demand may generate the request for the other resources. For such reasons, assessment of the expected performance is part of the performance monitoring routines. Detailed knowledge of hosted applications on a platform can forecast potential problems in addition to improved productivity. However, this information is not readily available without intelligent tools.

One way to precisely trace down the performance of applications is using Performance Monitoring Counters (PMCs) [7, 6, 8]). PMCs are hard-wired registers in Performance Monitoring Units (PMUs) of modern processors and are both architecture and vendor specific. These PMCs are responsible for monitoring micro-architectural PMU events at the CPU cycle level. The type and number of PMU events is dependent on the underlying architecture and so are the number of PMCs. Regardless of the architectural variations, the number of available PMCs is always significantly lower than the number of PMU events. These limitations restrict the engineers from monitoring more PMU events than the number of available PMCs. To them, it is a challenge to monitor numerous performance events without multiplexing many events on each PMC. Moreover, typical practices are static solutions based on a pre-selected list of PMU events with respect to engineers’ interests and knowledge. This brings up the challenge of an increased likelihood of missing out-of-sight information. Nevertheless, such inconsistencies in the assessments can be reduced with the help of an automated cross-platform mechanism capable of determining the performance profile of the applications based on data-driven solutions instead of relying on general practices only. A cross-platform approach is what is not limited to a particular hardware architecture, is not restricted to some vendor specifications and does not require significant modification in the method when is applied across different platforms. This ultimately allows to make better decisions if engineers have the provision of such information.

In particular, our main goal was to analyze how the applications utilize
computer resources. Here the first deliberation was to avoid selection bias (also called Survivorship bias, an error identified by Abraham Wald) [9] by capturing the overall behavior native to the platform on which the application is running. One of the challenges is that an application may not show the same behavior if it has been tested on a different platform and is running on a different one. Either way, measuring all returns a huge number of performance events. So our obvious goal was to 'look beyond the data in front of us' and funnel down the data from 'all' to the 'interesting ones' in particular to the underlying platform. While processing this broad spectrum of performance events one of the main considerations was to rank the performance events with respect to performance. The measure of performance is described differently in different scenarios but it is always related to time such as the number of instructions executed by the processor in a certain time or the number of packets sent or received in a particular time-frame over the network [10] are two different indicators of performance in two different scenarios. We consider the number of instructions passed through the 5-stage Reduced Instruction Set Computer (RISC) pipeline to serve as a key indicator of system performance (also explained in Section 3.1). Identifying the most frequent events with respect to performance would immediately determine the significant resource utilization behavior.

However, identifying significant performance events with respect to performance does not state how the resource utilization was performed. From hundreds of different performance events, any one can create thousands of data points per second of execution. This makes it hard to draw conclusions due to the given amount of data. Hence, the next challenge was to automatically analyze the acquired measurements. The captured data is complex for the reason that there is a logical hierarchy in the generation of different performance events coming from a particular hardware. It occurred as a result of multiple experiments that it is hard to find one single model that can describe the overall behavior of a performance event. Hence our interest moved toward simplification which is splitting the behavior into segments and providing the compact representation of dynamic-length segments based on statistical methods. The goal was not to quantify the magnitude of change but to have simple working sets based on the change in their data distribution.

To extend the analysis, the next goal was to automatically identify if per-
formance events can be grouped together based on their behavior. We consider PMU events can be related if they experience the changes in their behavior in a similar time fashion. The grouping criteria was not ‘exactly the same’ but ‘somewhat similar’. Therefore by applying the change point detection method, the number of changes for each performance event in comparison can come differently. This mismatch in the number of changes restricts the use of traditional clustering algorithms because they need an identical feature as a classification rule. Besides, complex models are data-hungry for training themselves before making predictions and this deters the use of machine learning algorithms. Therefore, the challenge was to design and develop fuzzy matching criteria that can provide a symmetric feature of comparisons for finding the structure in the unlabeled data.

In summary, the thesis presents a mechanism to mitigate the hardware limitations through an automated approach that pinpoints the most relevant events for the applications’ performance. The mechanism then enables us to understand the behavior through the compact representation of segments of performance events. Finally, an automated approach is presented to see if there are any groups of similar performance events based on changes in their behavior.

**Thesis outline** The thesis consists of two parts. Part I starts with presenting the overall research goals including the research approach and research method in Chapter 2. For a better reading experience, a technical background and related work are presented next in Chapter 3. Followed by the research methodology, results are presented in Chapter 4 to explain the contributions made through the published papers. Towards the end of Part I, a conclusion is presented along with prospective future work in Chapter 5.

Finally, the included papers are presented in Part II of the thesis.
Chapter 2

Research Overview

In this chapter, a brief description of the problem overview and research goals is provided. The chapter then continues to present the research method used to achieve the goals.

2.1 Problem Overview

Traditionally, a comprehensive view of systems performance is required for optimized and predictable usage of computer resources so we categorize the problems addressed through this thesis into the following three areas:

1. **Unstructured data and selection bias** - Specialized skill and training are prerequisites to a good analysis. So contextual understanding of both the tool and the examined architecture is required to draw accurate conclusions. Nevertheless, due to limited cross-platform compatibility, the current approaches are subject to selection bias. A more transparent data-driven solution is needed to automate the processes.

2. **Difficulty of analysis and visualization** - Data interpretation is hard due to the high volume of data generated by the complex system architectures. On one hand large volume of data is required for better analysis on the other hand sampling overhead serves as one of the grounds of statisti-
analyzing complex data without visualization and analysis tools is challenging.

3. **Classification** - For complex data, underlying patterns and relationships are hard to reveal. Classification related to the change in behavior implicates a corresponding cause-and-effect relationship between different performance events such as resource dependence, concurrency, and resource contention. Simply applying traditional classification methods may not report the deep insights, we may need to tailor them to suit our specific requirements.

### 2.2 Research Goals

Performance events are important as they are tied to the resource utilization behavior of the applications. We consider the uncertainty gets higher in a shared resource environment if one does not know how the limited available resources are being utilized among different applications. Moreover, to encounter the challenges of the widely accepted practice of performing stand-alone tests based on a static list of performance events, an automated solution is a way forward to avoid human errors, expertise shortage, and manual effort. An evidence-based insight and understanding of the run-time behavior of the applications is indeed valuable. Therefore, we started with an overall goal of investigating the performance profile of applications. To achieve the overall goal we have divided it into three sub-goals.

**Survivorship Bias - The Tale of the Forgotten Ones**

Our first goal was to establish an automatic cross-platform mechanism that can capture the overall resource utilization behavior of the applications with respect to the performance metric. Rather than relying on selected performance events, the goal was to involve all of them, including the ones that usually do not pass the selection criteria due to engineers’ knowledge and interest during the analysis. In short, the motivation was to include out-of-sight information while providing solutions and making decisions. Selection bias results in misleading insights and consequently leads to optimization misdirection, misguided resource allocation
and performance bottlenecks. An automated cross-platform approach is to reduce development costs, enhance collaboration, provide flexibility to adaption, reach a wider user audience and deliver consistent user experience.

**RG1** – To establish an automated cross-platform mechanism to profile the performance of the applications.

**Because Simple is Beautiful ...**

Our next goal was to understand how the application performance and related system resource utilization evolve at runtime. The obstacle was finding a best-fit model for such a complex resource utilization behavior. So we aimed to countermeasure the complexity in the context of simplified data. Therefore, the goal was established to divide the behavior into dynamic-length segments such that the segments are being identified based on abrupt changes in data distribution.

**RG2** – To establish an automated cross-platform mechanism for compact representation of performance events of applications.

**What Makes it Related ...**

We advanced the research by defining the next goal as identifying groups of performance events performing similarly. The data distribution may vary within each but if changes in their data distribution are occurring at similar points in time then we consider them related. So our focus was to investigate if they are related with respect to the time of change in their behavior.

**RG3** – To automatically group up similar performance events of applications related to the time of change in their data distribution.
2.2.1 Research Method

In persuasion of an effective and practical research approach, empirical research inspired by design science [11] was conducted. The advantage is evidence-based insights to further explore the topic. We employed different kinds of applications for our experiments ranging from computationally heavy applications to memory-bound applications [12]. We have also tested a malicious application [13] that is known for the absurd exploitation of computer resources. The motive behind their selection is likelihood and significant use in industrial systems. Such applications can enormously impact the system performance due to their eager resource utilization demands. Table 2.1 maps the goals achieved in the corresponding papers.

<table>
<thead>
<tr>
<th>Papers</th>
<th>RG1</th>
<th>RG2</th>
<th>RG3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paper A</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Paper B</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Paper C</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 2.1. Mapping of research goals into papers

In this thesis, Paper A proposes an approach to obstacle the architectural limitations of the underlying platform and aims to capture the overall cross-platform performance profile. The findings of Paper A endorsed the complexity of resource utilization behavior. So in the temptation of understanding the captured behaviour Paper B targets to model the system performance using statistical methods. In continuation to simplified behavior, the work continued in Paper C to explore the relationship between different performance events to see the signs of parallelism or influence on each other.

2.2.2 Research Approach

To conduct this research, an empirical approach inspired by design science is used. The overall research process is shown in Figure 2.1. We describe the steps as follows:

1. To achieve the overall goal, we formulated the problem to capture and
investigate the overall performance profile of applications.

2. Defined and outlined the mechanism to reach the research goals.

3. Designed and developed an artefact to provide the solution using statistical methods.

4. Preliminary results were obtained for further analysis using the developed artefact.

5. The preliminary results were evaluated to see whether the developed artefact solves the problem defined in step 1. There were two possible outcomes at this step:

   (a) The evaluation may give the ground to iterate back to step 2 to redefine the artefact.

   (b) The reasoning during the evaluation process can also advocate how the research can be expanded or if there is a need to go back to step 1 to reformulate the problem definition.

The evaluation of preliminary results together with industrial partners determined when to stop this iterative process.

6. A proposed solution is presented as an overall contribution upon receiving satisfactory results.

---

**Figure 2.1. Research approach**
Chapter 3

Background and Related Work

3.1 Performance

For improved performance of computers, the Reduced Instruction Set Computer (RISC) pipeline is a fundamental architectural feature of modern microprocessors [14]. And Instruction is the elementary unit a modern processor processes in one cycle, so the desire is to process more and more instructions in a given time. An instruction is marked retired as soon as it completes the 5-stage RISC pipeline. Concerning the current study, the more instructions retired in a certain time indicates higher performance.

The RISC pipeline is 5-stage process that allows concurrent execution of each stage, also shown in Figure 3.1. An instruction enters the first stage, called Instruction Fetch (IF), in which the processor fetches the instruction from memory using the program counter (PC). After the current instruction is fetched, the process moves to the next stage called Instruction Decode (ID) where the processor determines what operations are to be performed. Here for a concurrent approach, the next instruction is fetched in the IF stage as soon as the current instruction is moved to ID stage. In continuation to the ID stage, now instruction is executed based on the determined operation in the ID stage and this stage is called Execute (EX). During EX, if required, data is being written into or read from memory. Finally, after the execution, results are written back to the register at Write Back (WB) stage. The instruction is completed once it has passed the
write-back stage and is then marked as retired.

Figure 3.1. RISC pipeline for single instruction

3.2 Performance Monitoring Counters

The current state of modern computers enables us to precisely trace down the applications’ resource usage at run-time. Modern computers have special built-in hardware in the Performance Monitoring Unit (PMU) in the form of direct memory access (registers). These hard-wired Model-Specific Registers (MSRs) can be configured to monitor the events occurring during a specific time interval in a system. An event is an observable activity, state or signal whose occurrence can be from different sources such as hardware, software, kernel etc [6]. The registers are generally named Performance Monitoring Counters (PMCs) and events are called PMU events or performance events [7, 6, 8].

PMCs are grouped into fixed-function counters and general-purpose counters, where fixed-function counters are hard coded and general-purpose counters can be programmed to monitor any kind of PMU event. The performance events are implemented by using processor-specific codes. These codes along with other attributes of events are provided by the vendor in JSON files i.e. arch event definition file. The number of available performance counters varies depending on the hardware architecture. A typical Intel processor contains 3 fixed-function and 4 general-purpose counters per PMU [7] and the IBM BlueGene supercomputer has 64 in total [15]. However, in a multi-core system, each core has its own set of PMCs. PMCs are not only available for CPUs but sometimes also for other components of the computer such as GPUs, network interface cards (NICs), network switches etc [16]. By using these PMCs, micro-architectural
performance events can be monitored in the processor pipeline, such as the branch predictor unit (BPU), internal memory events, off-core events, network resource utilization, network problems, etc even for the different components in parallel.

One advantage of using PMCs is the low overhead of data extraction [17] for performance events like branch instructions retired, mispredicted branches, cache hit/misses or floating point operations. There are tools for data extraction such as perf API which is a performance analysis tool and the official Linux profiler for both kernelspace and userspace. The perf API was originally developed for monitoring PMCs but evolved into a tool capable of tracing kernel activities too [6]. It uses processor-specific raw hardware descriptors for the performance events. These codes can then be translated into aliases (low-level human-readable event names) by using an event mapping table [18, 19]. The hardware vendor provides these hardware details in the form of JSON files (arch event definition files), as shown in Figure 6.2. In Linux, these JSON files can be located at `tools/perf/pmu-events/arch/foo`. The information is then used by Performance Application Programming Interface (PAPI) which aims to provide consistent and OS-independent access to PMCs.

PAPI was introduced as an abstraction layer to access the PMCs using the perf API interface. Over time PAPI has evolved into a component-based architecture, which can monitor data from multiple components like CPU, thermal sensors, network, virtual machines etc [16, 20]. PAPI extracts perf event codes and maps them into human-readable names based on the underlying platform to save users from low-level architectural details. These performance events are divided into two categories named `presets` and `native`. Presets are events that are common and consistent across the majority of the platforms (also called `architectural` [7]). However native events are specific to a given platform on which they are running (also called `non-architectural` [7]). Due to rapid advancements in technology and version changes, static solutions require frequent checks and updates which can directly influence the system performance. So using the ’native mask’ non-architectural performance events can be extracted directly from the underlying hardware. The categorization of performance events and organization of profiling tools are also illustrated in Figure 3.2.
In short, when an event occurs it generates data that can further be utilized for statistical analysis as a metric or to generate an alert. These metrics are the result of evaluation or monitoring processes and can be used by technicians for system tuning and detection of faults. Events related to execution time, application memory-footprint size, memory latency, and error status can also present important insights.

### 3.3 Change Point Detection

Statistical methods are a conventional approach to analyze, interpret, and present huge amounts of data into a brief notation. Some of the common measures are standard deviation, mean and root mean square level to get valuable insights into data. However, there are numerous advanced methods also depending on the need and objective of analysis. Such as for determining segments in a data series, there are several methods like change point detection, cluster analysis, and time series segmentation.

Change point detection is one of the methods used for identifying distinct
points for partitioning a continuous data series. The method locates structural
and distributional changes based on statistical methods like mean, standard
deviation, and variance, also shown in Figure 3.3. The analysis can be parametric
or non-parametric. A parametric analysis estimates by explicitly providing the
location and/or the number of change points which is somewhat vulnerable to
deviation [21]. On the other hand, the non-parametric analysis does not require
a probability distribution assumption beforehand. These methods can be offline
or online. Online methods use a subset of data series whereas offline methods
use complete data series, from start to end, to make an analysis.

![Figure 3.3. Change points detection for a measured performance event i.e., TLB_FLUSH](image)

Some of the commonly used methods are likelihood ratio and Bayesian point
of view for single change point and multiple change point detection respectively. From a Bayesian point of view, it is possible to update the probability of the hypothesis with more data and a penalized contrast function [22]. The process is offline and the penalized contrast function starts with splitting the data series into two. Empirical estimation of statistical property (such as standard deviation, root mean square level, slope) is then calculated for each. Next, the sum of deviation from all the points in each part is calculated to see how much residual error still exists. The sum of aggregated deviations of each part gives a total residual error. This process is repeated until the final residual error is minimum [23]. Therefore, the Bayesian point befits the aim of our study. The result of the described process is also elaborated in Figure 3.3 for a measured performance event e.g., TLB_FLUSH.

It is also good to note some of the popular applications of change point detection are signal processing, genome, trend analysis, time series, intrusion detection, spam filtering, website tracking, quality control, step detection, edge detection, and anomaly detection.

3.4 Sequence Similarity - Similar Is Not Same

Sequence analysis or sequence similarity analysis is a popular method of identifying DNA similarity, a span of life trajectories & career and text similarity, alignment distances, document similarity and classification [24]. Some known methods are distance function (Chi-Squared, Euclidean), common attributes (Hamming distance, Longest common subsequence), Edit distance, Cosine similarity and Jaccard similarity. These methods are usually based on the measure of distance, order, position, time, duration and/or the number of repetitions [24]. Edit distance can be an appropriate choice if the aim is to quantify inequality. The method applies a weight for each edit function (insertion, deletion, substitution) until a sequence becomes identical to the other one. Cosine similarity is useful when similarity is not intended in terms of the size of the data. It can also be used in situations when the data sets are of different lengths and the orientation of the data is more important than the magnitude of the data [25]. Another method is Jaccard similarity which is a proximity measurement of shared properties i.e., size of intersection over the size of union [26].
3.5 Bounded Cost Function

All of these methods are based on an exact match of elements. However, in a classification problem, it is possible that some items are similar but not the same. Things are the same if they are identical to each other. Things can still be similar if they are not exactly the same. For example, if there are four sequences as below:

\[
S_1 = \{3, 5, 7, 1700\}, \\
S_2 = \{3, 5, 7, 1700\}, \\
S_3 = \{2, 5, 9, 1700\}, \\
S_4 = \{3, 5, 1700\}
\]

We can see that sequence \(S_1\) and sequence \(S_2\) are the same because all of their elements are an exact match to each other. Yet \(S_3\) and \(S_4\) are similar to \(S_1 \& S_2\) because \(S_3\) is slightly different for one element and \(S_3\) is missing one value for an exact match.

### 3.5 Bounded Cost Function

In a matching principle, when the objects in comparison are not exactly the same the similarity is quantified with probability. There are many ways to compute the probability such as Binary step function, Linear functions, and Non-linear functions. The binary step function applies a static cost if a certain threshold is passed. The drawback is that it does not provide back-propagation. Linear functions are mean, variance, and covariance and they also do not offer back-propagation and the absence of one value can augment the cost of the others.

In comparison, there is a variety of non-linear cost functions such as Sigmoid, Hyperbolic tangent (Tanh), Rectified linear unit (ReLU), and Exponential linear unit (ELU) [27]. These functions have the advantage of proposing a smooth and bounded cost. For example, Sigmoid converts the number on a scale of 0 and 1 and gives the probability value as output, also shown in Figure 3.4. Its smooth scale gives the rate of change based on the gradient descent. The s-shaped
curve has one inflection point where the curve changes the shape from convex to concave. This point can serve as a decision boundary for classification.

The Sigmoid function is also important in artificial neural networks and logistic regression. Logistic regression is used to predict binary classification where Sigmoid plays the role of the activation function using its bounded scale. The bounded scale is reasonable to estimate the likelihood of probability which is why they are considered reliable to use with analysis algorithms for optimization purposes also. Therefore, we opt to use the non-linear Sigmoid function to calculate the cost to be applied while matching the sequences.

\[
c_1(X) = 0.5 \times \frac{kX - kG}{\sqrt{(kX - kG)^2 + 1}} + 0.5
\]

Figure 3.4. Graph of sigmoid function
3.6 Related work

Here we present some of the related work in comparison to our research work and state-of-the-art. Most of the work in comparison is the one whose motivation comes mainly from the use of PMCs for various purposes. We also compare the ones who propose customized grouping and classification approaches.

The researchers [10] employed PMCs for the characterization of system performance and determined resource dependence of an application based on architectural events which are common across many platforms. One of the limitations of their study was to explicitly feed the performance events list for the characterization of the application. Their eventual focus was last-level caches only. Whereas our study focuses on all native events coming from the underlying platform to provide an automatic cross-platform solution.

Not only the cache but to explore other resources also there are studies that have used PMCs to estimate the power and bandwidth consumption [28, 17, 8] and to check the performance of applications in terms of CPU load and enhance quality of service by improving the performance [29]. Another study has used performance counters for the safety and security of the systems by proposing an attack mitigation model [30]. But as per our knowledge, these studies did not monitor all performance events of the platform they are running on. An interesting study performed by [15] on Blue Gene/P was on a supercomputer to monitor the massive number of performance events (256 concurrent 64b counters). Although the capability to monitor performance was increased it is not very commonly available architecture across many Small and Medium Enterprises (SMEs) so the solution is not generally applicable.

There are situations when splitting the data set into segments becomes appropriate to provide a solution to the given problem. Several researchers have introduced various methods of segment detection. The well-known algorithms for change point detection are E-Agglomerative, Wild binary segmentation, Bayesian analysis of change points and Iterative robust detection of change points [31]. E-Agglomerative is a cluster-based approach to estimate change points depending on the goodness of fit [32]. The method is used to detect multiple change points within a data set. However, many of the methods require pre-screening to exclude the irrelevant points for improved accuracy which is
Yao considered multiple change points with the Bayesian point of view [33]. The Bayesian point of view is a form of statistical reasoning based on calculated probabilities to provide the best possible prediction. It is used when the inputs and information are not sufficient to determine the output. Yao also presented graph-based change point detection for high dimensional and non-euclidean data [34]. He studied the single-point case to estimate a change even when there is noise in data. The method can even estimate when the number of jumps is unknown and they are within defined bounds.

Another study [35] used randomly sampled basic block frequencies (sparse) without any dedicated hardware support and using PMCs. They propose Precise Event Based Sampling (PEBS) to reduce the run time overhead as one of the prime goals of their study. But it requires extensive normalization of data before processing.

To identify similarities and differences between multiple data sets some of the standard methods are least square and likelihood. However, it is not possible to directly apply the concepts due to inconsistencies in data and complicated requirements and conditions. There are other existing similarity approaches such as DNA similarity, Cosine similarity, Edit distance, and Jaccard index but they have preconditions like identical or different lengths, same data structure or exact match [24, 25, 26]. The way they compare is more strict and can be applied in absolute conditions. When it is not the case researchers like Fletcher and Islam [36] have used the Jaccard index for comparing the patterns coming from different techniques. Their proposed method converts each pattern into a single element which is also the commonality between their and our solution. However, the method to get a discrete value of similarity is different. Their method translates each pattern into an element of its own set whereas we compute the similarity based on element-wise weighted distance with respect to the lengths of the sequences. This is an additional strength of our proposed mechanism to handle the inconsistencies of data.

In situations involving limited data and diverse conditions for grouping, an approach has been applied by Koch, Zemel and Salakhutdinov [37] for one-shot image recognition where very limited or sometimes single example
is available to compare in supervised machine learning. They employed the sigmoid function in siamese convolutional neural networks to find the similarity between the final and hidden layer of the twin network. Their approach was to scale the absolute distance between 0 and 1 with the help of training parameters. Since their problem was binary classification so instead of utilizing real-value output the values from 0.5 to 1 were regarded as dissimilar. Whereas we use the resultant weighted cost as a probability of similarity. Moreover, their working sets were of the same length so one-to-one comparisons were directly possible which on the contrary was not a viable option for us. So we provide additional functionality to find the closest possible match with our holistic and intelligent approach.
Chapter 4

Research Results

4.1 Thesis Contributions

This section lists the contributions made through the goals achieved in this research, also shown in Figure 4.1.

• C1: An automated cross-platform mechanism to capture the overall performance of the applications.

• C2: An automated mechanism to identify the most relevant PMU events related to performance that describes what computer resources have been utilized most by the application under investigation.

• C3: An automated approach to present a compact representation of complex performance events based on statistical methods

• C4: An automated approach to group up similar performance events based on a measure of similarity

4.1.1 C1: An automated cross-platform mechanism to capture the overall performance of the applications

One of the core contributions made through Paper A [38] and Paper B [39] is the automated profiling of applications which has an enhanced ability to capture
Figure 4.1. Mapping of research goals into corresponding contributions and papers

RG1: To establish an automated cross-platform mechanism to profile the performance of the application

RG2: To establish an automated cross-platform mechanism for compact representation of performance events of applications

RG3: To automatically group up similar performance events of applications related to the time of change in their data distribution

Overall Goal:
Survivability Bias: The role of the Forgotten Ones...

Paper A

RG1

C1

Because Simple is Beautiful...

Paper B

RG2

C3

What Makes it Related...

Paper C

RG3

C4

Chapter 4. Research Results
the overall behavior of applications on the platform where the application is running. The ability to capture the performance events native to the underlying computer architectures makes it consistent across different platforms. Besides, the automated solution is equally interesting for experts and non-experts in terms of ease of use, efficiency, competence, dynamicity, consistency and decision-making.

The consistent mechanism also ensures reduced human intervention and manual effort by characterizing all available performance events for the entire execution period of the application through re-run multiplexing. Whereas existing temporal multiplexing approaches are prone to blind spots due to which critical times may go unnoticed during event evaluation. Moreover, re-run multiplexing enables to characterize the short duration applications which can be problematic in the case of temporal multiplexing.

4.1.2 C2: An automated mechanism to identify the most relevant PMU events related to performance

The work in Paper A [39] continues to describe what computer resources have been utilized most by the application under investigation. The way the applications utilize computer resources indicates the likelihood of resource boundness, resource contention and security threats. Since different resources can be involved during the execution of an application the proposed mechanism is capable of reporting which hardware or software resources the application was utilizing most. Using statistical methods the mechanism ranks all characterized events with respect to performance to report the most relevant ones for the particular application under investigation.

4.1.3 C3: An automated approach to present a compact representation of complex performance events based on statistical methods

The work in Paper B [39] contributes not only to capture but also to understand the behavior of the applications. Interpretation of data generated as a result of complex resource utilization behavior was challenging to translate into one model. Various simplification and abstraction methods like polynomial and curve
fitting techniques provided unsatisfactory results. Realizing the complexity, a sophisticated approach to simplify the working data set was determined i.e., to decompose the data into segments. There are many approaches for segmentation, current work emphasizes abrupt changes in data distribution as segmentation points. These changes can be identified using statistical methods like change point detection which enables to locate the significant moments in data shifts. Instead of investigating each point in time, considering the critical moments helps eliminate the impact of statistical noise caused by sampling errors.

4.1.4 C4: An automated approach to group up similar performance events based on a measure of similarity

A further contribution was made through Paper C [40] to group up similar performance events. There are different classification criteria to identify the groups. However, due to quite a few limitations, applying a traditional clustering algorithm was not a viable solution in our case. The main challenge was to group up based on a 'not exactly same' but a 'somewhat similar' basis considering data is coming from complex, sensitive and rational resource utilization demands.

Another challenge was to deal with uneven lengths of series in comparison. Therefore, we have proposed a novel approach by tweaking the Sigmoid function and Jaccard index that considers the real value differences as cost while computing the weights of similarity. The methods can efficiently augment the existing clustering algorithms as a similarity function to compute the pairs of performance events. These pairs are then represented in a tree-structure i.e. dendrogram where the height of each linkage represents how different the performance events are from each other.

4.2 Included papers

A mapping of contributions to the corresponding papers is illustrated in Table 4.1.
4.2 Included papers

<table>
<thead>
<tr>
<th>Paper A</th>
<th>Paper B</th>
<th>Paper C</th>
</tr>
</thead>
<tbody>
<tr>
<td>RG1</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>RG1</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>RG2</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>RG3</td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 4.1. Mapping of contributions made through the papers for each research goal

4.2.1 Paper A

**Title:** Automatic Platform-Independent Monitoring and Ranking of Hardware Resource Utilization

**Authors:** Shamoona Imtiaz, Jakob Danielsson, Moris Behnam, Gabriele Capannini, Jan Carlson, Marcus Jägemar

**Status:** Published in proceedings of 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2021)

**Abstract:** In this paper, we discuss a method for automatic monitoring of hardware and software events using performance monitoring counters. Computer applications are complex and utilize a broad spectra of the available hardware resources, where multiple performance counters can be of significant interest to understand. The number of performance counters that can be captured simultaneously is, however, small due to hardware limitations in most modern computers. We suggest a platform independent solution to automatically retrieve hardware events from an underlying architecture. Moreover, to mitigate the hardware limitations we propose a mechanism that pinpoints the most relevant performance counters for an application’s performance. In our proposal, we utilize the Pearson’s correlation coefficient to rank the most relevant performance counters and filter out those that are most relevant and ignore the rest.

**My Contribution:** Our industrial partner and co-author Marcus Jägemar has initiated the need for the problem to be solved. Following that, I was the main driver and author of this work. All authors have also contributed to the planning of the paper through productive and joint discussions. I have extended and implemented the characterization prototype provided by Jakob Danielsson. I have also written the first draft of the paper and all authors have contributed to
improving the content with their valuable feedback.

### 4.2.2 Paper B

**Title:** Automatic Segmentation of Resource Utilization Data  
**Authors:** Shamoona Imtiaz, Moris Behnam, Gabriele Capannini, Jan Carlson, Marcus Jägemar  
**Status:** Published in proceedings of 1st IEEE Industrial Electronics Society Annual On-Line Conference (ONCON 2022)  
**Abstract:** Industrial systems seek advancements to achieve required level of quality of service and efficient performance management. It is essential though to have better understanding of resource utilization behaviour of applications in execution. Even the expert engineers desire to envision dependencies and impact of one computer resource on the other. For such reasons it is advantageous to have fine illustration of resource utilization behaviour with reduced complexity. Simplified complexity is useful for the management of shared resources such that an application with higher cache demand should not be scheduled together with other cache hungry application at the same time and same core. However, the performance monitoring data coming from hardware and software is huge but grouping of this data based on similar behaviour can display distinguishable execution phases. For benefits like these we opt to choose change point analysis method. By using this method our study determines an optimal threshold which can identify more or less same segments for other executions of same application and same event. Furthermore the study demonstrates a synopsis of resource utilization behaviour with local and compact statistical model.  
**My Contribution:** I am the main driver and author of this work. All authors have also contributed to planning the paper through productive discussion. I have implemented the prototype and written the first draft of the paper and all authors have contributed to improving the content with their valuable feedback.

### 4.2.3 Paper C

**Title:** Automatic Clustering of Performance Events  
**Authors:** Shamoona Imtiaz, Gabriele Capannini, Jan Carlson, Moris Behnam, Marcus Jagemar
Status: Published in proceedings of 28th Annual Conference of the IEEE Industrial Electronics Society (ETFA)

Abstract: Modern hardware and software are becoming increasingly complex due to advancements in digital and smart solutions. This is why industrial systems seek efficient use of resources to confront the challenges caused by the complex resource utilization demand. The demand and utilization of different resources show the particular execution behavior of the applications. One way to get this information is by monitoring performance events and understanding the relationship among them. However, manual analysis of this huge data is tedious and requires experts’ knowledge. This paper focuses on automatically identifying the relationship between different performance events. Therefore, we analyze the data coming from the performance events and identify the points where their behavior changes. Two events are considered related if their values are changing at approximately the same time. We have used the Sigmoid function to compute a real-value similarity between two sets (representing two events). The resultant value of similarity is induced as a similarity or distance metric in a traditional clustering algorithm. The proposed solution is applied to 6 different software applications that are widely used in industrial systems to show how different setups including the selection of cost functions can affect the results.

My Contribution: I was the main driver and author of this work. All co-authors have also contributed to the planning of the paper through productive and joint discussions. Together with Gabriele Capannini and Jan Carlson, we discussed different approaches for the proposed method that I have been implementing to evaluate their validity. Finally, the prototype provided by Gabriele Capannini was selected and then implemented by me. I have also written the first draft of the paper after which I and Gabriele Capannini have rewritten the methodology to its final version. All authors have also contributed to improving the content with their valuable feedback.

4.3 Publications not included in the thesis
4.3.1 Paper X

Title: Towards Automatic Application Fingerprinting Using Performance Monitoring Counters
Authors: Shamoona Imtiaz, Jakob Danielsson, Moris Behnam, Gabriele Capannini, Jan Carlson, Marcus Jägemar
Status: Published in proceedings of 7th International Conference on the Engineering of Computer Based Systems (ECBS 2021)
My Contribution: I am the main driver and author of this work. All authors have also contributed to planning the poster paper through productive discussion. I have written the first draft of the paper and all authors have contributed to improving the content with their valuable feedback.
Chapter 5

Conclusion and Future Work

The main goal of this thesis is to understand the resource utilization behavior of the applications when running on a particular platform. Various performance events are available per platform, we aimed for an automated approach to providing better insights of application behavior. In system performance monitoring, CPU and memory are among the main sources of information [6] however the information can be collected from software, hardware and kernel as well. We have identified the challenges coming from hardware limitations, vendor specifications and lack of documentation while collecting and processing this information. The proposed mechanism does not require significant modifications to implement across different platforms since it unfolds the platform-specific list of available performance events to start its operations. Moreover, the proposed mechanism is resistant to selection bias which otherwise can be a reason to miss signs, clues and details of determining undesired state. These challenges were handled in the best possible way to establish a good foundation for comprehension and knowledge about application behavior. The proposed solution has also made an advancement to approach the aim of reduced investigation time and effort through an automated approach.

We have further explored and developed a solution to understand different phases in the behavior of each performance event. It was hard to present the data using one model so a simplified approach was determined i.e., segmentation. This enabled us to only consider the points in time where a change in resource
utilization demand is expected after a stable behavior. All these times can be considered during resource management to handle the parallel resource utilization demand from concurrent applications. This improves efficiency and saves from continuous monitoring. The proposed method does not require normalization which is a pre-condition for many methods. As a result of segmentation, the simplified subsets allow focused analysis, simplified visualization and deeper insights.

Finally, a contribution is made towards a relative picture of performance events with respect to changes in their behavior. It is hard to see a relationship between different performance events due to the vast variety of architectures and available performance events per platform. If the data is small then it still can be analyzed with extra effort but in case of the sheer volume of data, which is a result of complex behavior also, is time-consuming, error-prone and challenging. So a mechanism is proposed here that can compute the proximity of similarity between performance events by applying weighted real-value costs to relate different performance events. The automated mechanism reports groups of similar performance events with a decent accuracy based on concurrent changes in their behavior. The proposed solution can serve as a baseline feature to determine the relation and dependency between different resources.

5.1 Future Work

Solving complex performance problems usually requires holistic approaches. Finding an issue is not a problem, finding what matters the most at the time of decision is a challenge. The aim is to create a fingerprint of applications using its performance events. Having a fingerprint of an application can serve various purposes such as identification, detection or even decision-making.

In the future, we will extend the investigation for the segment-wise understanding of performance events. We are also interested in exploring how the segmentation can be performed with less number of measurements. The more the data the better examples are for comparison and change detection. One possible direction was to apply machine learning models but even the basic machine learning models are data-hungry to learn from the previously seen data. And when we are working on a very low level then it becomes a big overhead...
to run these computationally heavy models. So the aim is to look for more sophisticated and lightweight statistical methods that can provide better insights even when there are not many or any examples to be compared.

Another immediate extension can be relating the trends in the data before and after the segmentation points to identify the impact between different resources. This way one can focus on the magnitude of the change for budget management which is another performance management problem that results in over-provision of the resources. Moreover, a future step is to provide a user-friendly publicly available tool to benefit from its capabilities. Making it open-source would allow improvements over time with the help of the wider community.

This also needs to be further explored what insights can be drawn to identify any relation at different resource levels. Currently, available documentations are poorly maintained that even the manual analysis using the performance event name is a problem. So a solution that can already relate the different data sets of performance events to their names, codes and details can supplement the documentation and analysis activities.

Finally, our aim is to present a model that can portray the best overall resource utilization behavior of the application.
Bibliography


