Analytic Cloud Platform for Near Real-Time Mass Spectrometry Processing on the Fast Data Architecture

DISSERTATION

zur Erlangung des akademischen Grades

Doktoringenieur (Dr.-Ing.)

angenommen durch die Fakultät für Informatik
der Otto-von-Guericke-Universität Magdeburg

von Msc. Roman Zoun

geb. am 01.03.1987 in Tscherepovez

Gutachterinnen/Gutachter

Prof. Dr. Gunter Saake
Prof. Dr. Mourad Elloumi
Dr. Dirk Benndorf
Prof. Dr. Stefan Conrad

Eingereicht am:
Magdeburg, den 18.10.2019

Verteidigt am:
Magdeburg, den 30.06.2020

Zoun, Roman:
Analytic Cloud Platform for Near Real-Time Mass Spectrometry Processing on the Fast Data Architecture.
Dissertation, University of Magdeburg, 2019.

Inhaltsangabe

Ein Massenspektrometer ist ein Gerät zum Messen von Biomarkern in biologischen Gemeinschaften. Diese Biomarker können genutzt werden, um Biogasanlagen zu optimieren und Energieeffizienter zu machen oder auch Krankheiten zu diagnostizieren. Die aktuellen Analysearbeitsabläufe eines Massenspektrometers sind sequenziell und beinhaltet Stunden von Wartezeiten zwischen den einzelnen Arbeitsschritten. Diese Situation ist nicht akzeptabel, besonders wenn es um klinische Diagnostik geht. Zusätzlich ist die softwaregestützte Datenanalyse sehr komplex und braucht eine stabile Hardwareinfrastruktur, welche mit hohen Kosten und Aufwand einhergeht.

In unserer Arbeit präsentieren wir einen konzeptionellen Beweis einer analytischen Plattform für Echtzeitanalyse von Massenspektrometer-Experimenten. Wir implementierten MStream, eine Cloudbasierte Plattform, welche auf der Fast-Data-Architektur aufsetzt und eine skalierbare, streambasierte Protein Identifikation ermöglicht. Wir diskutieren und lösen alle Herausforderungen, die für die Adaption einzelner Schritte einer streambasierter Lösung auf einer Cloudarchitektur benötigt waren. Die erste Herausforderung war die Konzepterstellung, die zweite Herausforderung war das streamen der Daten während der Messung direkt von dem Massenspektrometer. Die dritte Herausforderung war die Strukturierung der Daten für einen schnellen Durchsatz und die letzte Herausforderung war die streambasierte Validierung der Ergebnisse. Das Resultat ist MStream, eine prototypische Umsetzung einer Cloudplattform zur Echtzeitverarbeitung von Massenspektrometerdaten. Am Ende der Arbeit wird die Plattform evaluiert und die Ergebnisse zeigen eine bessere Performance im Vergleich zu aktuellen Software-Alternativen.

Abstract

A mass spectrometer is a device which can measure biomarkers of biological environments such as sea, biogas plant, human gut or a just a blood. Using these biomarkers, it is possible to optimize biogas plants in order to maximize the energy production or to diagnose diseases of thousands of patients with only one mass spectrometer. Unfortunately, the mass spectrometry data analysis pipeline is sequentially including hours of waiting time between the workflow steps. This situation is not applicable especially for use cases such as clinical diagnostics. Additionally, the data analysis is complex and needs a stable infrastructure, which involves very high costs and effort.

In our work, we present a proof of concept of an analytic platform for real-time analysis of mass spectrometry experiments. We implemented MStream, a cloud-based platform on the fast data architecture for scalable streamlined protein identification. We discuss and solve all challenges in the thesis for adapting the components to a streaming cloud-based pipeline. First challenge is the concept of cloud-based architecture for streaming mass spectrometer data analysis pipeline. Next challenge is the possibility to stream from a device during the measurement. The third challenge is to structure and transform the data in the database systems and the last challenge is the streaming validation of the results. Finally, we implement MStream, a prototype of real-time cloud platform for analyzing mass spectrometry data during the measurement. Furthermore, our evaluation results show the performance gain for the analysis process in comparison to the state-of-the-art software.

Contents

List of Figures

List of Tables

List of Code Listings

1. Introduction

Mass spectrometers are devices to digitize real world samples, helping to solve puzzles of the humanity with growing success on the market[1]. The mass spectrometry technology measures proteins to identify protein biomarkers of biological environments, such as oceans, humans and other microbial communities, which are used in the research fields proteomics and metaproteomics [AM03, Ast, HKRB15, AA98]. These biomarkers are similar to a fingerprint and can be used to identify viruses, bacteria or specific proteins in the sample [PF17, SAB+08]. The mass spectrometer analysis results can be used for pharmacy research, security issues on an airport or research on energy industry [ZLY+15, HSS+19, PF17, AM03]. Additionally, the mass spectrometer reveals the possibility to identify proteins of known diseases such as cancer, Alzheimer's disease, and even lupus. Hence, it is used for clinical diagnostic [LQD14, NKH+17, PJW+14, PF17, ECL+12, BFWSS+15, XYF+17]. Since diagnostics are time critical, research is striving for an ever-increasing performance [HSZ+17].

Due to the fast quality upgrades of the mass spectrometer, they produce ever-increasing amounts of data, resulting in terabytes of output data by a single machine. This data alone is useless and cannot diagnose anything without the analysis and post-processing that bring insights into these samples. Due to the huge data sizes and the complexity of the algorithms, the current sequential analysis takes hours to complete [HSZ+17, HBS+93, MBR+13, PPCC99].

Current mass spectrometers need two hours for a measurement followed by a conversion step of the data of up to one hour and an additional analysis step which takes several hours to complete [HSZ+17, AM03, Cla19, Ast].

The state-of-the-art software that is used for the downstream analysis works with file-based input data such as X!Tandem, Andromeda or Mascot [PPCC99, CNM+11, RR03, Tab15]. Accordingly, the algorithms are specialized to process the data in a bulk processing fashion. Nevertheless, the state-of-the-art data analysis (protein

[1] Mass spectrometry growth from 2016 - 2025 from 5.3$ billion dollar to 10.5$ billion [res17].

identification) needs all the experimental data at once. As a consequence the analysis step, which itself takes hours, is further delayed by another several hours because it has to wait until all the measurement data is available. While these delays are tolerable in many research applications, in clinical diagnostics they are not. Therefore, a real-time analysis is needed, so that the data can be processed during the measurement. Among other benefits, a real-time analysis could be stopped if a specific result is identified, which reduces the overall time of the analysis.

Goal of this Thesis

In the thesis, our goal is a new streaming workflow for processing mass spectrometry data that analyses the data **during the measurement** of the mass spectrometer on a cloud based **fast data architecture**. The research can be divided into two levels — the feasibility and the performance:

Feasibility level: At this level, the goal is to identify the needed components of the analysis pipeline and transform the input data from the complete experiment file into one single dataset. This level answers the question: "Is a streaming fast data architecture feasible for mass spectrometry analysis workflows?"

Performance Level: This level is the connection of all components in one system in order to evaluate the performance of the whole system compared to the state-of-the-art solution. This level answers the question: "Does a streaming fast data architecture increase the performance of the workflow?"

In the following, both levels are explained in detail.

Level 1: Feasibility

During our research, we have found several challenges that have to be solved. The first challenge is the streaming protein identification, the next one is the device-streaming interface, the third challenge is the distributed data model and the fourth is the validation[2] of the results on streaming data.

Streaming Protein Identification: The first issue is to connect the protein identification algorithm to a per-spectrum data source instead to the current complete file source. In addition, an adoption of the similarity-scoring function and its parallelization can be used for an identification process as a pipeline. This goal leads to the following research question.

RQ 1: What are design choices that are essential for adopting the analysis pipeline to a fast data architecture?

Device-Streaming Interface: The next challenge is to stream a mass spectrum immediately when it is measured from the mass spectrometer. However, the inner workings of each mass spectrometer are special for each manufacturer and highly involved. Hence, the challenge will be to create a general, manufacturer independent interface, which satisfies the needs of all users.

[2]The state of the art target-decoy approach cannot be applied on streaming data, a decoy-free validation is needed, see Chapter 6

RQ 2: How to stream the data during the measurement without losing information or functionality in the current or further processing steps?

Distributed Data Model: Since our system collects a huge amount of data asynchronously, a horizontally scalable database management system is needed. The storage should provide the experimental spectra and the protein data for the protein identification procedure.

RQ 3: How to structure the data to increase the performance of the identification process from a streaming mass spectrometer?

Decoy-Free False-Discovery-Rate calculation: The next challenge is the False-Discovery-Rate (FDR) calculation. The FDR is the expected rate of false positives in the result. The target-decoy method repeats searches of the whole sample data again for the FDR calculation. This step produces target-result collection and a decoy result collection[3]. Therefore, all experiment data is needed, to provide these collections. In our parallel approach, we cannot provide all the information about all the spectra from a sample, because the mass spectrometer streams each spectrum separately. Hence, we need new methods for calculating the FDR without the decoy process.

RQ 4: How to avoid target decoy method without compromising the result quality of the validation results?

Level 2: Performance

At this level, the main task is to implement a prototype and evaluate the performance of the whole system. To this end, we will evaluate the process regarding performance and scalability. Scalability will be measured from the user perspective (increasing the number of devices) and from the data perspective (increasing the amount data from the database).

RQ 5: How does protein identification scale with an increasing number of connected devices?

RQ 6: How does protein identification scale with an increasing amount of protein data?

With our result, we reveal a new direction of future development of mass spectrometry data analysis and its application in medical environment or other areas. In summary, we transform the mass spectrometry data analysis to a streaming cloud architecture in two steps. Firstly, solving all biological and technical problems in every single step of the analysis pipeline and secondly, evaluating the performance of the complete system.

[3]Decoy collection contains wrong results and is used for validation of the target results.

Structure of the Thesis

In order to present the research contributions of the thesis in understandable parts, we divide the thesis into 10 chapters. In the following, we give a small overview of the content chapters Chapter 2 to Chapter 7 as well as related work in Chapter 8. The thesis is rounded up by a conclusion part in Chapter 9 and future work in Chapter 10.

Chapter 2 – Background

In Chapter 2, we present the basics that should be well understood before going deeper into the topic of mass spectrometry analysis and fast data architectures. Hence, we give biological basics about mass spectrometry, spectrum analysis and metaproteomics. Furthermore, we review big data technologies and the fast data architecture. This chapter is based on:

> Robert Heyer, Kay Schallert, Roman Zoun, Beatrice Becher, Gunter Saake, and Dirk Benndorf. Challenges and Perspectives of Metaproteomic Data Analysis. *Journal of Biotechnology*, Volume 261:24 – 36, November 2017
>
> Atin Janki, Roman Zoun, Kay Schallert, Rohith Ravindran, David Broneske, Wolfram Fenske, Robert Heyer, Dirk Benndorf, and Gunter Saake. Connecting X! Tandem to a Database Management System. In *GI-Workshop Grundlagen von Datenbanken*, GvDB, pages 77–82, May 2018
>
> Robert Heyer, Kay Schallert, Corina Siewert, Fabian Kohrs, Julia Greve, Irena Maus, Johanna Klang, Michael Klocke, Monika Heiermann, Marcus Hoffmann, Sebastian Püttker, Magdalena Calusinska, Roman Zoun, Gunter Saake, Dirk Benndorf, and Udo Reichl. Metaproteome Analysis Reveals that Syntrophy, Competition, and Phage-Host Interaction Shape Microbial Communities in Biogas Plants. *Microbiome*, Volume 7(1):69, April 2019

Chapter 3 – Challenges of Streamlining the Mass Spectrometry Analysis

The Chapter 3 focuses on the feasibility goal of this thesis. In this chapter, we review the idea of a streaming pipeline and extract the components of the mass spectrometry analysis pipeline. Furthermore, we present a general streaming architecture for the whole system and define the challenges of the single components. Additionally, we propose a technology stack and finally present an interactive visualization of mass spectrometer protein results. As a result, we present concepts for answering **RQ 1**. This chapter is based on:

Roman Zoun. Internet of Metaproteomics. In *IEEE 34th International Conference on Data Engineering*, pages 1714–1718, April 2018

Roman Zoun, Gabriel C. Durand, Kay Schallert, Apoorva Patrikar, David Broneske, Wolfram Fenske, Robert Heyer, Dirk Benndorf, and Gunter Saake. Protein Identification as a Suitable Application for Fast Data Architecture. In *Database and Expert Systems Applications*, pages 168 – 178. IEEE, September 2018

Roman Zoun, Kay Schallert, David Broneske, Robert Heyer, Dirk Benndorf, and Gunter Saake. Interactive Chord Visualization for Metaproteomics. In *Database and Expert Systems Applications*, pages 79–83, August 2017

Chapter 4 – Streaming Mass Spectrometer

The Chapter 4 focuses on the mass spectrometer and how to stream the data during the measurement. To reach the goal, we explain how we collaborate with a mass spectrometry manufacturer. As a result, we present concepts for answering **RQ 2**. This chapter is based on:

Roman Zoun, Kay Schallert, David Broneske, Wolfram Fenske, Marcus Pinnecke, Robert Heyer, Sven Brehmer, Dirk Benndorf, and Gunter Saake. MSDataStream - Connecting a Bruker Mass Spectrometer to the Internet. In *Datenbanksysteme für Business, Technologie und Web*, pages 507 – 510. Gesellschaft für Informatik, March 2019

Chapter 5 – Managing the Protein Sequence Data

The Chapter 5 focuses on the protein data for later use for the protein identification. We discuss the data dependencies and show the data structure and index structure for the data to improve the performance for later identification processes. In addition, we show several ways to transform the protein data into that structure evaluating the best method. As a result, we present concepts for answering **RQ 3**. This chapter is based on:

Roman Zoun, Kay Schallert, David Broneske, Ivayla Trifonova, Xiao Chen, Robert Heyer, Dirk Benndorf, and Gunter Saake. Efficient Transformation of Protein Sequence Databases to Columnar Index Schema. In *Database and Expert Systems Applications*, pages 67–72. Springer International Publishing, August 2019

Chapter 6 – Validation of Streaming Peptide Spectrum Matches

The Chapter 6 focuses on the validation of an identified hit. We discuss why the usual target-decoy approach is not feasible for streaming systems and present a machine learning based solution for validation. As a result, we present concepts for answering **RQ 4**. This chapter is based on:

Roman Zoun, Kay Schallert, Atin Janki, Rohith Ravindran, Gabriel C. Durand, Wolfram Fenske, David Broneske, Robert Heyer, Dirk Benndorf, and Gunter Saake. Streaming FDR Calculation for Protein Identication. In *Advances in Databases and Information Systems*, pages 80 – 87, September 2018

Chapter 7 – Analytic Platform for Near-Real-Time Mass Spectrometry Data Analysis

The Chapter 7 focuses on the scoring function of the protein identification and the connection of all system components. In this chapter, we evaluate the complete throughput from the mass spectrometer to the identified data. As a result, we present the proof of concept implementation MStream for answering **RQ 5** and **RQ 6**. This chapter is based on:

> Roman Zoun, Kay Schallert, David Broneske, Sören Falkenberg, Robert Heyer, Sabine Wehnert, Sven Brehmer, Dirk Benndorf, and Gunter Saake. MStream: Proof of Concept of an Analytic Cloud Platform for Near-Real-Time Diagnostics using Mass Spectrometry Data. Technical Report 002-2019, Otto-von-Guericke-University Magdeburg, August 2019

Chapter 8 – Related Work

In Chapter 8, we review the related work for this thesis. This related work includes a description of different approaches for mass spectrometry analysis using modern technologies.

Chapter 9 – Conclusion

In Chapter 9, we conclude each content chapter regarding the goals defined in the introduction chapter. Finally, we conclude the overall results of the thesis.

Chapter 10 – Future Work

In Chapter 10, we show some possible future improvements. This future work contains a description of different additional components to the mass spectrometry analysis.

2. Background

The mass spectrometry data analysis is a growing topic not only in the research community, but also in industry areas such as clinical diagnostic, security on airports or energy industry [res17, Ast, HKRB15, PF17]. In this chapter, we describe the basic biological knowledge for understanding the analysis pipeline as well as the basics of cloud computing technologies.

In the following, we first review the typical analysis workflow of mass spectrometry analysis in metaproteomics/proteomics research area. Secondly, we introduce the protein identification method and the validation of the identification process. Thirdly, we describe the cloud architecture we will use for our implementation, as well the technologies.

It is important to understand the necessary steps of the analytic pipeline in order to follow the challenges on the new architecture. These parts explain the fundamental knowledge to understand our design decisions in the later chapters and to contrast our contribution from the others.

2.1 Mass Spectrometry Workflow

The mass spectrometry field deals with the analysis of protein biomarkers. The workflow is sequential and the smallest parallelizable unit is the whole experiment itself. In Figure 2.1, we show the experiment pipeline.

The sample could originate from a stool sample, a biogas plant sample or from an ocean, so everything that could contain living organisms. After the sample is collected, the proteins are extracted (Figure 2.1–1). This preparation is done in a laboratory and is not the focus of this work [Ast, MRML07, AM03, BVJ+09, WAWH05]. For clinical sample preparation, these steps could be automated [PF17]. Further, the prepared sample is measured in the mass spectrometer (Figure 2.1–2). The digitized data is collected in a manufacturer specific RAW[1] format. The duration of the measurement is between one and two hours and results in gigabytes

[1] Containing the raw signal data and need to be processed and aggregated to get usable information from it.

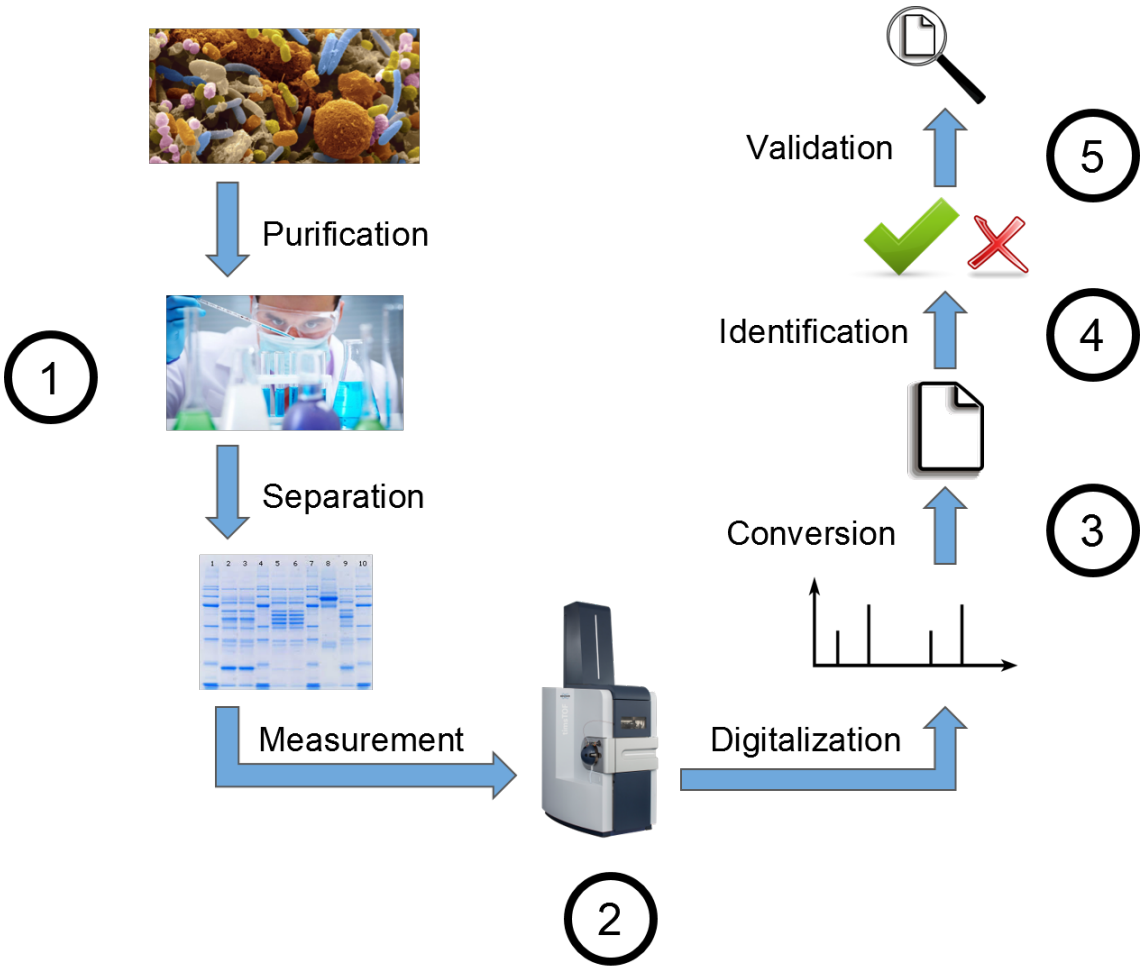

Figure 2.1: State-of-the-art workflow of mass spectrometry experiments with the biological preparation (1), the measurement of the mass spectrometer (2) and the data processing pipeline (3-5) [mur19, the19, kom19].

of data, containing thousands mass spectrometry datasets, so called mass spectra [AM03, LBK+17, BM11, SBD+08]. In the next step, the RAW data is converted into a standard file format[2] (Figure 2.1–3). During the conversion, several preprocessing steps, such as noise reduction and pre-filtering are executed in order to increase the quality of the data for further analysis [AM03, LBK+17]. This step takes up to 1 hour of the processing time. In the next step (Figure 2.1–4), the data is compared to a sequence database such as UniProtKB or a user defined protein sequence database. The protein data in UniProtKB are manually annotated and reviewed by scientists and can be assumed as high qualitative [ABW+04]. The knowledge base is usually in a fasta[3] file format [Nat02, Whi05]. The validation of the hits is the last step of the process (Figure 2.1–5). Further analysis such as visualizations, reasoning by a physicians or biological researchers is done based on those identification results [HKRB15, MRML07, BW99, VRS+08, BVJ+09].

[2]Mascot Generic File (MGF), a lightweight text format or mzML, a XML-based structured text format [Mat16, Deu12, MTS+04].

[3]Text file containing the protein sequences and meta information of proteins and is developed as a exchange format for the software suite FASTA in 1985 [Nat02, Whi05].

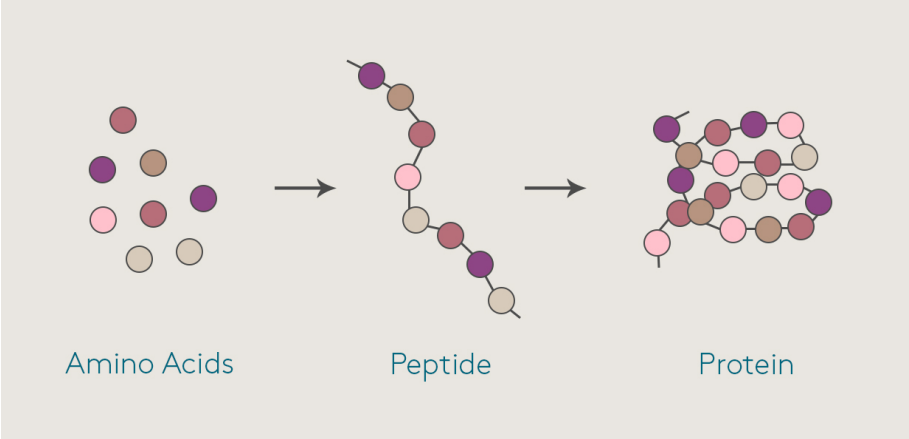

Figure 2.2: Visual model of amino acids, peptides and proteins. Each circle color expresses a different amino acid. A collection of amino acids builds a peptide and a collection of peptides builds a protein [col19].

The central component of the data processing pipeline is the protein identification step shown in Figure 2.1–4. In our following discussions, we skip the laboratory part, Figure 2.1–1, since this is not part of this work [HKRB15, HSS+19]. In the next section, we explain briefly the measurement using a mass spectrometer.

2.2 Measuring the World with a Mass Spectrometer

A mass spectrometer is a device to measure molecules such as proteins and digitize a biological sample. Proteins are the building blocks of living organisms and to understand how they function and interact with each other, the proteins need to be analyzed. The method was proposed by Hunt et al. and describes the workflow in for a mass spectrometer measurement [HYS+86]. Each organism can be specified as a collection of amino acids [PF17, HKRB15, ZLY+15, res17, HSS+19].

In Figure 2.2, we show the connection between amino acids, peptides and proteins. Amino acids are organic molecules with known properties and they represent the building blocks of any organic structure. There exist about 500 occurring amino acids [WM83] but in the term of biochemistry and mass spectrometry, we consider only 22 different types of amino acids. This is enough, since proteins in living organisms are always containing a combination of these 22 amino acids [HBS+93]. A collection of amino acids forms a peptide and a collection of peptides forms a protein.

For mass spectrometry, the biological sample is purified and the proteins are cleaved[4] into peptides. The goal of the mass spectrometer is to measure the mass of the amino acids in these peptides. In the mass spectrometer, the sample is turned into their gas phase and get ionized [AM03, RGSP07]. As next the ionized particles are separated creating the first mass spectrum (MS1 spectrum) [AM03, KNR19, HSS+19].

[4]separated into sub-sequences, so called peptides

Figure 2.3: Visual model of a tandem mass measurement [HYS+86].

Each measurement represents a peptide in this phase. For each MS1 spectrum, the particles of it are fragmented into smaller ions using collisions. The measurement of the fragments after the collisions results in a second spectrum (MS2 spectrum). The MS2 spectrum represents the chain of amino acids of the peptide separated by their differences between the fragment masses. In Figure 2.3, we show a sketch of the tandem mass spectrometer measurement workflow [AM03, HYS+86].

For the thesis, it is important to know the connection between amino acids, peptides and proteins and the results of the mass spectrometry measurement. In the next Section, we explain the output data of the mass spectrometer.

2.2.1 Mass Spectrum Data

In this section, we explain the experimental output data from the mass spectrometer device. The device measures the masses of the peptides (parts of a protein) and their amino acids. The mass of the particles is represented by a mass spectrum, see Figure 2.4. On the x-axis, the mass to charge ratio can be seen and on the y-axis the intensity of the signal, measured by the device, is represented. The intensity signal represents the collisions, which are detected by the digitizer of the mass spectrometer device. The mass of a peptide is a list of peaks that can be represented in a textual format or as a plot of the peaks.

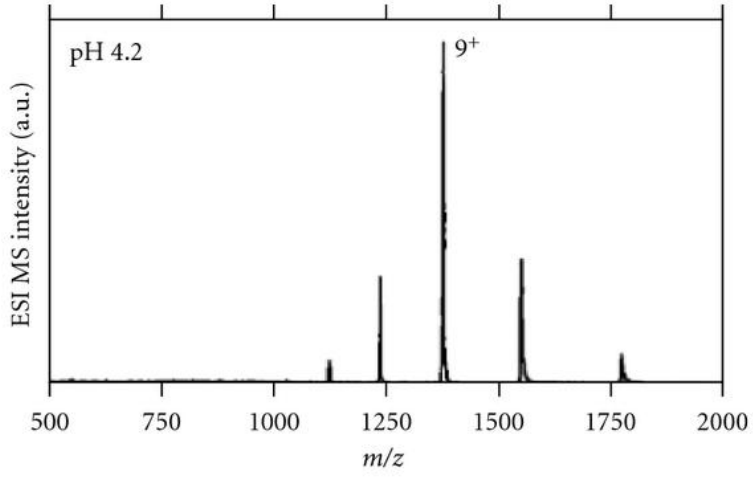

Figure 2.4: A plot of a mass spectrum [BM11].

In our work, we analyze the MS2 spectrum datasets. Each mass spectrometer manufacturer has its own data format of the digital signal [LBK+17, ZSB+19b]. The digital RAW data needs a conversion step, in which the data are pre-processed due to noise reduction and pre-filtering. The digital signal is converted in one of the textual file formats used as input data in further software tools of the analysis pipeline [MTS+04].

Nowadays, the XML standard mzML[5] and the Mascot Generic File[6] (MGF) are typically used for the experimental data of a mass spectrometer. The MGF format are very famous to describe the mass spectrum data in a lightweight format [Deu12, MTS+04, Mat16]. A mass spectrum contains the precursor mass (the total mass of a peptide) and the fragment mass differences of the amino acids in the peaks. Hence, the textual representation of a mass spectrum is a collection of the peaks of the mass spectrum, the total mass of the peptide and some descriptive information. One mass spectrometer produces hundred thousands of spectra every two hours, resulting in up to 100GB files [HSZ+17, LBK+17].

```
1  BEGIN IONS
2  TITLE=Spectrum000101 AEFVEVTK +2 y- and b-series
3  PEPMASS=308.16757   2000000
4  CHARGE=3+
5  36.52588    456
6  74.06009    24
7  101.04717   3511
8  124.58392   342
9  174.11813   787
10 201.08703   3
11 224.11559   66
12 238.63943   2361
13 288.17363   74
14 338.17109   12
15 348.15544   785
16 388.69493   5361
17 447.22386   12
18 576.26645   106
19 675.33486   4976
20 END IONS
```

Listing 2.1: MGF format example

In Listing 2.1, we show the textual representation of a mass spectrum in MGF format. A typical MGF file contains thousands of such mass spectra. Each spectrum is separated with the start word “BEGIN IONS” and the end word “END IONS”. The properties of a mass spectrum are “TITLE”, “CHARGE”, “PEPMASS” and the peak list. The “TITLE” names and identify the mass spectrum. The property “PEPMASS” defines the MS1 peak and the summed mass of fragment mass differences of the amino acids. The “CHARGE” is the charge value and finally

[5]XML based textual representation of mass spectrum data.
[6]Lightweight textual representation of mass spectrum data

the peak list describes pairwise the peaks with mass-to-charge-ratio on X-axis and intensity on Y-axis. Additional properties exist, but are not important for this work [HSZ+17, HSS+19, Deu12, Mat16].

In the next section, we explain the protein data and how they are involved in the mass spectrometry analysis pipeline. The proteins are the reference data for the analysis pipeline and it is important to know how to compare the protein data with the mass spectrometer results.

2.2.2 Protein Data

The protein sequence database is another involved data source, which is needed for the analysis of mass spectrometry data [DAC10, Whi05]. This is usually in FASTA format, including a textual representation of the protein sequences and their descriptions [Nat02]. This data is the theoretical data and is created manually by the research community or by algorithms. In Listing 2.2, two proteins are shown with their unstructured text description (Listing 2.2 line 1) and their protein sequence (Listing 2.2 line 2 – 4). Each letter of the protein sequence represents one of the 22 amino acids. Each protein contains a sequence of amino acids and a description. The uniqueness of a protein is given by the tuple of sequence and description. However, the proteins are used to perform comparison of the resulting data of a mass spectrometer to them. Since the output data of the mass spectrometer are peptides, the protein sequences need to perform the same separation (digestion) [DAC10]. The protein digestion separates the protein sequence into smaller peptides. Several enzymes exists to split the protein on specific positions. The most popular is using "trypsin". The enzyme interacts with specific proteins and cleaves it. In Figure 2.5, we show the trypsin enzyme digestion rules and how the peptides are created from a protein sequence [uni19, Ell98]. Trypsin cuts the protein sequence at the amino acids lysine (K) and arginine (R) except when either is followed by proline (P) [DAC10, AM03, uni19, RGSP07].

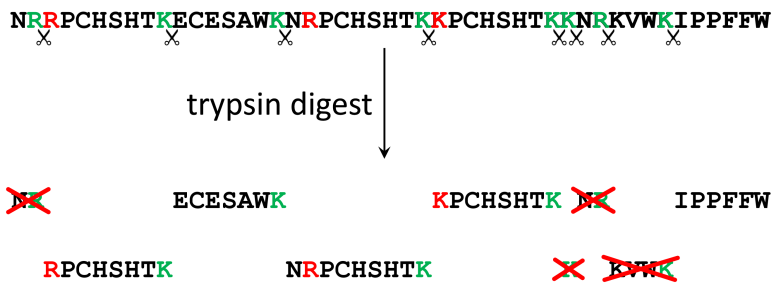

Figure 2.5: Trypsin digestion of a protein. The sequence is splitted on C-terminus of the amino acid K or R but only if they are not followed by P [DAC10, AM03, uni19, RGSP07].

Since the trypsin digestion rules are based on the reaction of an enzyme on a biological sample, the yield of cleavage is not 100%. Hence, the separation is not 100% accurate in the real world. Consequentially, the digestion needs to consider missed cleaves for the splitting algorithm. The so called missed cleavage describes that behavior [TLM+00, RGSP07]. Using missed cleavage value bigger than zero $MCV > 0$

results in a possible peptides that are not splitted on every possible position. The value $MCV = 0$ is following the trypsin rule from Figure 2.5. The value $MCV = 1$ follows the rule, but allows to skip one cleavage and hence, adds more possible peptides. In Figure 2.6, we show an example of missed cleavage on tryptic digestion of a protein sequence. In the example, the protein sequence "ACEDFHSAKDFQEASDFPKQWFE" cleaves with $MCV = 0$ on position 8 and 18. Hence, the peptides are "ACEDFHSAK", "DFQEASDFPK" and "QWFE". The value $MCV = 1$ allows to skip a cleavage and adds another possible peptides. Accordingly, the peptides are identical to $MCV = 0$ "ACEDFHSAK", "DFQEASDFPK" and "QWFE". Additionally, we add peptides that skip one cleave in position 8 or position 18. Consequently, the peptides "ACEDFHSAKDFQEASDFPK" and "DFQEASDFPKQWFE" are added. All the combination of cleavages are the result of the digestion. The resulted peptides can be used to compare to the mass spectrometry output data (mass spectra) to identify the sample [TLM+00, RGSP07, Whi05].

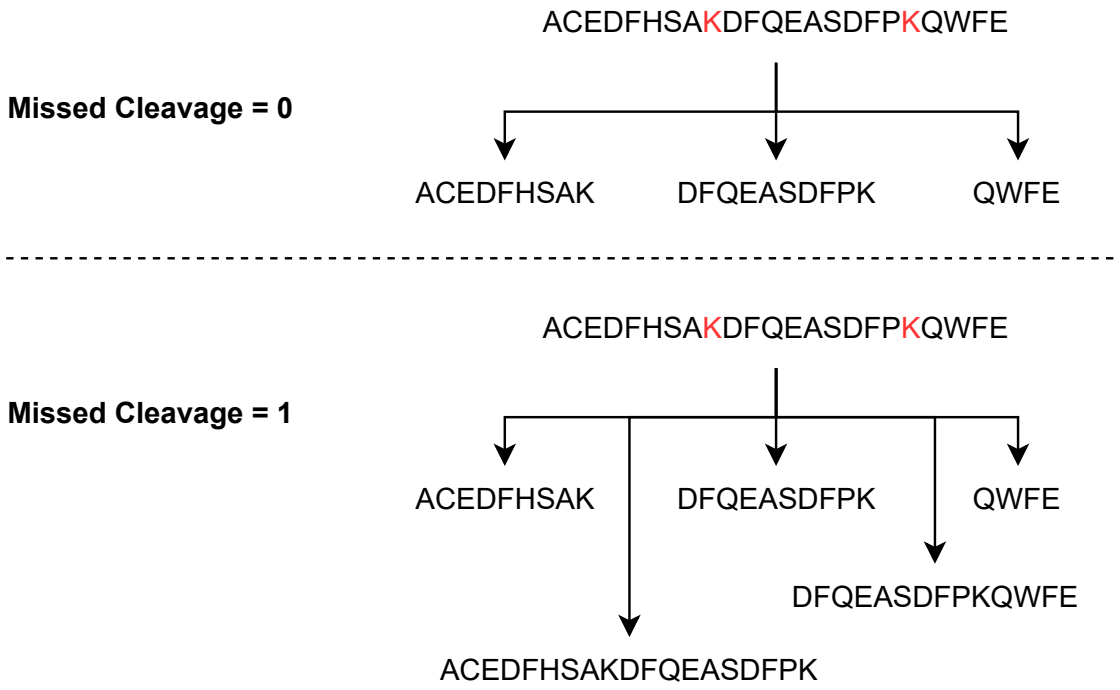

Figure 2.6: Example of missed cleavage on tryptic digestion of a protein [RGSP07].

A popular protein sequence database is SwissProt from UniProtKB[7] with 559,077 proteins. After the digestion, the SwissProt proteins are divided into 37,403,696 peptides (using missed cleavage value two). This digestion increases the amount of data from 500MB proteins to over 2GB of peptides.

Since the peptides are amino acid chains and cannot be directly compared to a mass spectrum, a theoretical spectrum needs to be generated. In the next section, we explain the special properties of the theoretical spectrum.

[7]UniProtKB in January 2019 contains 559,077 reviewed and 139,694,261 unreviewed protein entries (https://www.uniprot.org/)

```
1  >sp|P31946|1433B_HUMAN 14-3-3 protein beta/alpha OS=Homo sapiens
       GN=YWHAB PE=1 SV=3
2  MTMDKSELVQKAKLAEQAERYDDMAAAMKAVTEQGHELSNEERNLLSVAYKN
3  VVGARRSSWRVISSIEQKTERNEKKQQMGKEYREKIEAELQDICNDVLELLD
4  KYLIPNATQPESKVFYLKMKGDYFRYLSEVASGDNKQTTVSNSQQAYQEAFE
5  ISKKEMQPTHPIRLGLALNFSVFYYEILNSPEKACSLAKTAFDEAIAELDTL
6  NEESYKDSTLIMQLLRDNLTLWTSENQGDEGDAGEGEN
7
8
9  >sp|Q4R572|1433B_MACFA 14-3-3 protein beta/alpha OS=Macaca
       fascicularis GN=YWHAB PE=2 SV=3
10 MTMDKSELVQKAKLAEQAERYDDMAAAMKAVTEQGHELSNEERNLLSVAYKN
11 VVGARRSSWRVISSIEQKTERNEKKQQMGKEYREKIEAELQDICNDVLELLD
12 KYLIPNATQPESKVFYLKMKGDYFRYLSEVASGDNKQTTVSNSQQAYQEAFE
13 ISKKEMQPTHPIRLGLALNFSVFYYEILNSPEKACSLAKTAFDEAIAELDTL
14 NEESYKDSTLIMQLLRDNLTLWTSENQGDEGDAGEGEN
```

Listing 2.2: Textual representation of one protein in a FASTA format

2.2.3 Theoretical Spectrum

A theoretical spectrum is the result of the reverse process of the mass spectrometry measurement. After digestion of the proteins the peptides are transformed into a spectrum. The fragment peaks are calculated from the known mass and sequence of the amino acids. In Table 2.1, we show the list of amino acids and their masses. Following the rules as published by Eng et al., it is easy to generate theoretical peak lists [EMY94, RR03]. The difference to the experiment spectrum is the intensity. Since in the real measurement the intensity is measured by the device, in a theoretical spectrum the intensity is for each peak 100%. Additionally, the theoretical spectrum does not contain any noises or false peaks in the spectrum [KNR19]. Some of the amino acids have same mass and the identification gets more complex. Furthermore, modifications change the mass, which can lead to same mass between the particles. These problems are tackled during the protein identification process [MBR+13].

In Figure 2.7, we show a comparison of experimental spectrum on the upper side and a theoretical spectrum on the lower side of the image. On the one hand, the experimental spectrum has different intensity values and more peaks, which are false and are measured due to noises. On the other hand, the theoretical spectrum contains only optimal peaks with maximum intensity. Furthermore, the theoretical mass spectra peaks are without noises. The comparison of these two datasets produces very low similarity score and is prone to errors.

To conclude, we explained the mass spectrometry and the experimental as well as the theoretical data involved in further analysis [ABW+04, Ast]. The comparison of the data is very complex and results in low similarity score [MFT+13, MBR+13]. The theoretical mass spectra are reverse engineered from the protein data and the experimental data is measured by a mass spectrometer [KNR19, Ast]. In the next section, we will explain the main analysis step in the mass spectrometry analysis pipeline, the so called protein identification.

Amino acid	Short	Formula	Mon. mass§ (Da)	Avg. mass (Da)
Alanine	A	C3H5NO	71.03711	71.0779
Cysteine	C	C3H5NOS	103.00919	103.1429
Aspartic acid	D	C4H5NO3	115.02694	115.0874
Glutamic acid	E	C5H7NO3	129.04259	129.1140
Phenylalanine	F	C9H9NO	147.06841	147.1739
Glycine	G	C2H3NO	57.02146	57.0513
Histidine	H	C6H7N3O	137.05891	137.1393
Isoleucine	I	C6H11NO	113.08406	113.1576
Lysine	K	C6H12N2O	128.09496	128.1723
Leucine	L	C6H11NO	113.08406	113.1576
Methionine	M	C5H9NOS	131.04049	131.1961
Asparagine	N	C4H6N2O2	114.04293	114.1026
Pyrrolysine	O	C12H19N3O2	237.14773	237.2982
Proline	P	C5H7NO	97.05276	97.1152
Glutamine	Q	C5H8N2O2	128.05858	128.1292
Arginine	R	C6H12N4O	156.10111	156.1857
Serine	S	C3H5NO2	87.03203	87.0773
Threonine	T	C4H7NO2	101.04768	101.1039
Selenocysteine	U	C3H5NOSe	150.95364	150.0489
Valine	V	C5H9NO	99.06841	99.1311
Tryptophan	W	C11H10N2O	186.07931	186.2099
Tyrosine	Y	C9H9NO2	163.06333	163.1733

Table 2.1: Table of amino acids with short letter representation and their mass. The mass is used to generate a peak in a theoretical mass spectrum.

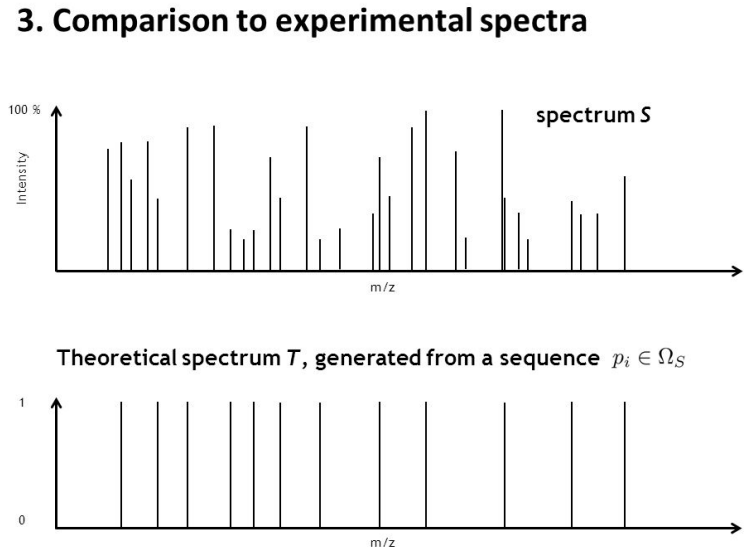

Figure 2.7: The figure shows a comparison of experimental spectra, containing false peaks or noises and a perfect theoretical spectrum of the same peptide [KNR19].

2.3 Protein Identification

The data measured by a mass spectrometer is transformed into a digital signal, containing masses of the particles. The next step is to map the masses to the real world data, the experimental mass spectrometry data. This is the comparison of the protein data and the experimental data [MFT+13].

For a better understanding, we will explain the procedure on an abstract example: we assume that three persons from one department are working in our building during the night. We want to know, who is working in the night in our office building. The only information about the persons is the mass, so his or her weight. To identify the hard working person, we can use the whole list of the employees for the company with the masses (which is quite hard to get) and search for similar weights. After searching, we can identify several possible persons, but excessively much to identify the person. Therefore, as next step, we look that department of our office building contains all three weights. Then, the combined masses of the persons could identify a department. Hence, the employees. In this sample, the employees are peptides and the departments are the proteins, while the whole building is a living organism. On a microscopic level, the problem is much more complex.

The incoming experiment data is a digital signal from real world proteins. The data is without any meaning and the data has to be identified compared to a protein knowledge base. Therefore, the protein identification approach is used to identify the measured spectra [MFT+13].

In the following, we first explain the general protein identification approach, followed by a state-of-the-art implementation of the method.

2.3.1 Concept of Protein Identification

The protein identification compares the real world data, which is represented as measured mass spectra from the mass spectrometer and the theoretical data from the protein sequence database, which contains already known real world proteins [MFT+13, DAC10, Whi05].

In Figure 2.8, we show on the upper side, the biological preparation of the sample. After the purification of the sample from a patient, only proteins are left. Protein data consists of a protein sequence and the meta information. The protein sequence is split (digested by specific enzymes) into peptides. Then the mass spectrometer measures the peptides and generates their digital signal, which represents the experimental spectra.

The lower side in Figure 2.8 shows the same procedure for the known proteins. The protein sequences from a protein sequence database are cleaved into peptides, using same splitting rules and reconstructing each peptide into theoretical spectra. This ensures the comparability of the data. The matching is then between each peptide (theoretical spectrum) and each experimental spectrum (peptide from the experimental sample), that results in a peptide-spectrum-match (PSM). Due to the unsteady digital signal during the measurement and the noises, the experimental spectrum can never be as good as the theoretical one, so the comparison ends up in low similarity score. The matching is similar to the comparison of a photography

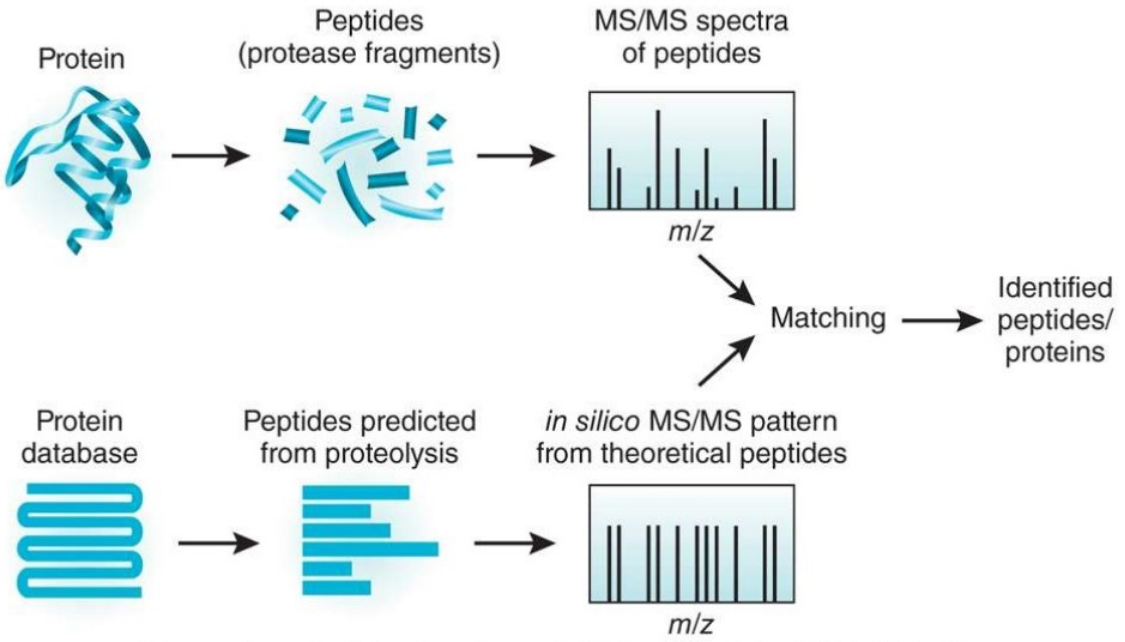

Figure 2.8: A general peptide centric protein identification method, which compares the experimental spectra and the theoretical ones [MFT+13].

and a clip-art, which is always ambiguous to some degree. In conclusion, the scoring needs to be validated for true matches or false matches with the idea that wrong hits have even lower score [MFT+13].

The protein sequence databases are growing regularly, containing millions of entries[8]. For the protein identification, the proteins of the knowledge base are divided into peptides, since the experimental data measures on peptide level. Hence, the split of the proteins increases the amount of data. The complete annotated knowledge base SwissProt includes over a half of a million protein entries, which results in millions of peptides (23,934,321). For the mass spectrometry data analysis a scalable architecture is needed, since a comparison against all known proteins consumes a lot of computation resources [HSZ+17].

2.3.2 Protein Identification in X!Tandem

Currently, many protein identification software tools exist under proprietary and free license, such as Mascot or X!Tandem [PPCC99, RR03]. Mascot is a proprietary software from the company matrix science and X!Tandem is an open source software, which is popular in the research areas of proteomics and metaproteomics community and provides very fast processing of the data. Both rely on the sequential data processing pipeline. In this section, we present the processing of the data in X!Tandem, because we could analyze and measure the process in the free available source code. Mascot is another very famous protein search engine, unfortunately the source is not open and the software is commercial.

[8]UniProtKB in January 2019 contains 559,077 reviewed and 139,694,261 unreviewed protein entries (https://www.uniprot.org/)

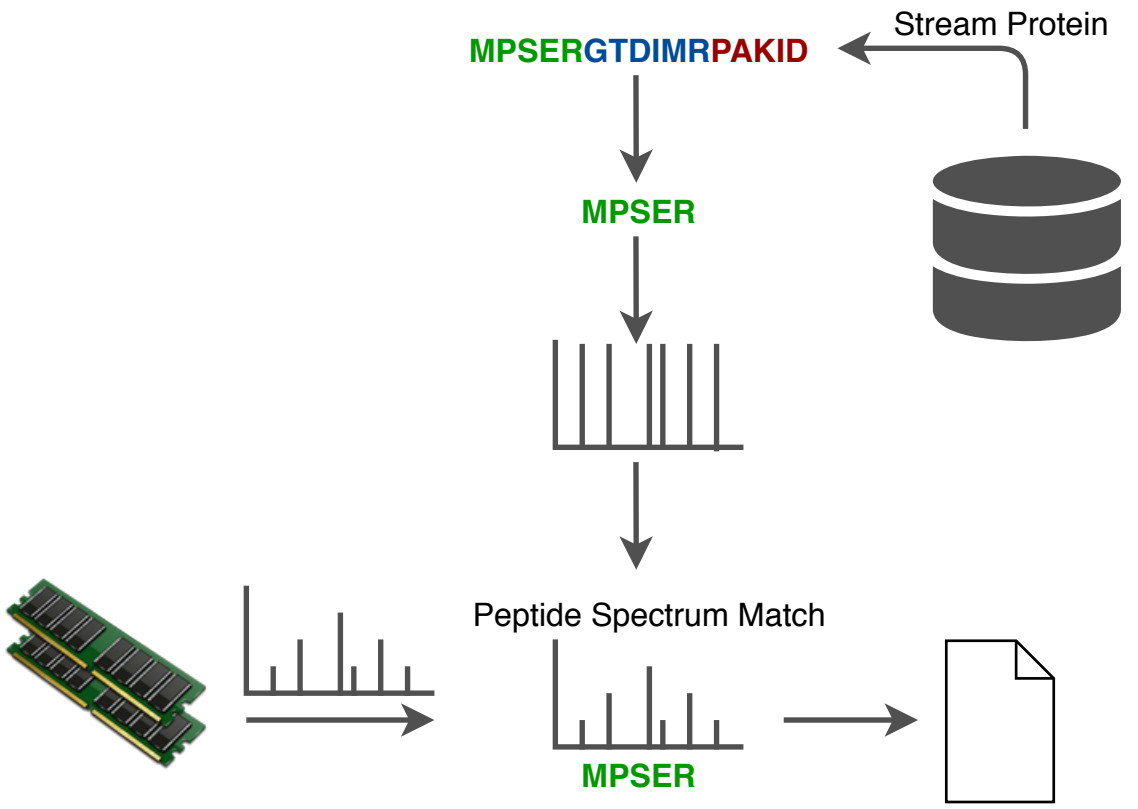

Figure 2.9: General processing of the protein identification in X!Tandem protein engine [RR03].

In Figure 2.9, we show the process of the comparison in X!Tandem. The protein data comes from the hard disk, while the mass spectrometry data is in the main memory. The software iterates once over the protein database and multiple times over the spectra. The following steps are carried out: firstly, the software loads the experimental data at once[9], filtering out low quality data, based on user-defined parameters. Secondly, the software processes data from files batchwise. Thirdly, X!Tandem splits each protein in smaller peptides and fourthly, the algorithm compares them with the experimental spectra. Lastly, the best peptide-spectrum-matches are stored in the end of the process in an XML based result file [RR03].

The main difference to other tools is the scoring function and the pre-filtering or output format. The procedure is parallelizable and can be applied to cluster and cloud platforms to increase the performance through horizontal scalability [BCC+08, Hai88]. Furthermore, we showed already the possibility to connect the X!Tandem software to a database management system and created an interface for the software [JZS+18]. Nevertheless, the identified PSMs need a validation to ensure the quality of the results. The validation step is needed in all protein identification software.

[9]MetaProteomeAnalyzer uses X!Tandem as protein identification engine and allows to split experimental files [MBH+15].

2.3.3 Protein Identification in Andromeda

Another software for protein identification is Andromeda, a probabilistic peptide search engine. As known, the input data are the Mass spectrometer data and the protein sequence data. The primary goal of Andromeda is to provide flexibility for working with high fragment mass accuracy [CNM+11].

In contrast to X!Tandem, Andromeda starts with the construction of the protein database. The data from a textual representation (FASTA file format) are transformed into parts, which are indexed for easier access during the evaluation. Andromeda preprocesses the protein data before to start the identification process. Firstly, the proteins are indexed and the index is used to skip comparisons with the theoretical mass spectra. In X!Tandem, this is done during the comparison. Secondly, the sequences are digested into peptides. This step reduces the effort to split the protein data during the runtime. Thirdly, the peptides are sorted by their mass, including the modifications, which is beneficial, because only specific theoretical masses are used for the comparison. Each of the measured experimental spectrum is compared to the theoretical peptides. The comparison results in a PSM, which need to be validated.

The method of Andromeda applies the behavior of a streaming environment, but tackles the problems of non-distributed platform. Each of the three steps of Andromeda is synchronized for each experiment and a distributed architecture is aimed with more suitable persistence [CNM+11]. In our work, we parallelize these steps and include the validation to the processing pipeline.

2.4 Target Decoy Validation

The validation method processes each peptide-spectrum-match (PSM) and tests whether the PSM is a true positive or false positive match. Since every experimental data is individual and no general rule exists, the state-of-the-art validation uses a target-decoy approach for each experiment [EG09]. The idea is to allow only a small amount of wrong hits in the result using approximation of the false discovery rate (FDR).

In this method, the identification of experimental spectra against the knowledge base results in target PSMs. Additionally, a wrong knowledge base[10] is created and then used in an additional identification process for the experimental data to produce decoy PSMs. Later, the target and the decoy PSMs get stored in a collection, sorted by the similarity score in descending order. Afterwards, the bottom PSMs are iteratively removed until the desired false discovery rate is reached, using the formula from Figure 2.10 for the FDR calculation in every step. All removed target PSMs are false positives, the rest of the target PSMs are valid hits [HSS+19, HSZ+17, EG09].

In Figure 2.10, we show a sample of a collection of target and decoy hits. The FDR at the beginning of the method in this sample is 66% (two decoy hits divided by three target hits). After removing the last element, only one decoy hit is left and changes the FDR to 33% (one decoy hit divided by three target hits). Next bottom element

[10] Usually inverts the protein sequences to create wrong proteins [EG09].

is a target hit and changes the FDR to 50% (one decoy hit divided by two target hits). The next removed decoy element changes the FDR in this sample to 0% (zero decoy hits divided by two target hits). In a real world scenario millions of elements are in the collection and the method stops usually with an FDR of 1% and 10%, but it depends on the sample and is chosen by a biologist [HSS+19, HSZ+17, EG09].

For repeating use cases such as diagnostics, the FDR is a static value, since the sample as well as the identification process is well known. However, for each experiment the FDR need to be calculated, because in this approach, the method depends on the experimental data. However, the drawback of this approach is that it needs the complete experimental data at once and is not applicable to our prototype, which requires real-time processing.

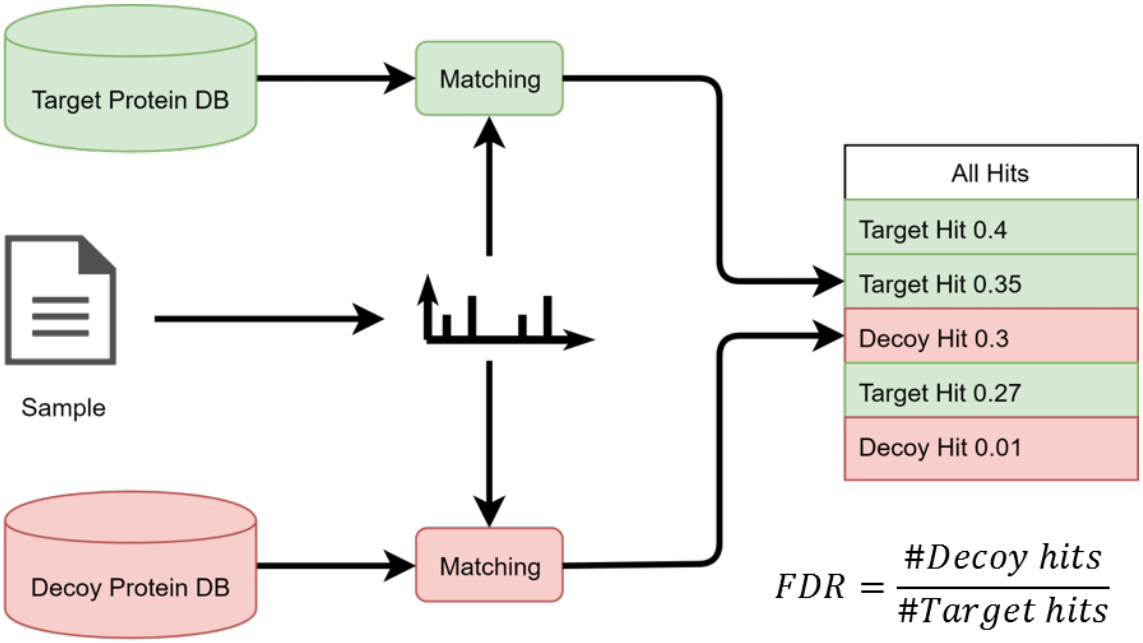

Figure 2.10: General target decoy validation [Bio19, EKT+12, EG09].

2.5 Machine Learning Classification

A machine learning classification is used to provide a decision on complex data. The real world data needs a conversion into a vector. This vector with defined features describes the data [Han07]. For supervised leaning, each of the vectors get a label of the class. This information is used for the training. The training shapes a model depends on the class labels. However, the model can predict the result for new incoming data, based on the training data. The incoming data need the conversion to a feature vector, too. Depends on the training, the class label for the new feature vector gets predicted. In Case of protein identification results, the features are the combined values of the comparison and properties of the experimental spectrum and the theoretical peptide. Hence, the validation of PSMs can be defined as a binary classification, because only true positive and false positives are the class labels. In this section, we describe logistic regression, a machine learning approaches for binary classification [MBY+16].

Logistic Regression

There is a plethora of machine learning approaches that classify different datasets. The goal is a function, which can predict the outcome, based on the training model. For this work, we will not explain multi-label classifiers and explain only binary classifiers because in the validation of the protein identification only two results are possible. In Figure 2.11, we see the two possible labels 1 and 0 on the Y-axis and the data points on X-axis. After an evaluation of different binary classifiers, we determined logistic regression as a binary classifier for our work because this method reached the best accuracy in our tests [Qui86, Bre01, KMF+17, RRK+90, DP97, PLI02].

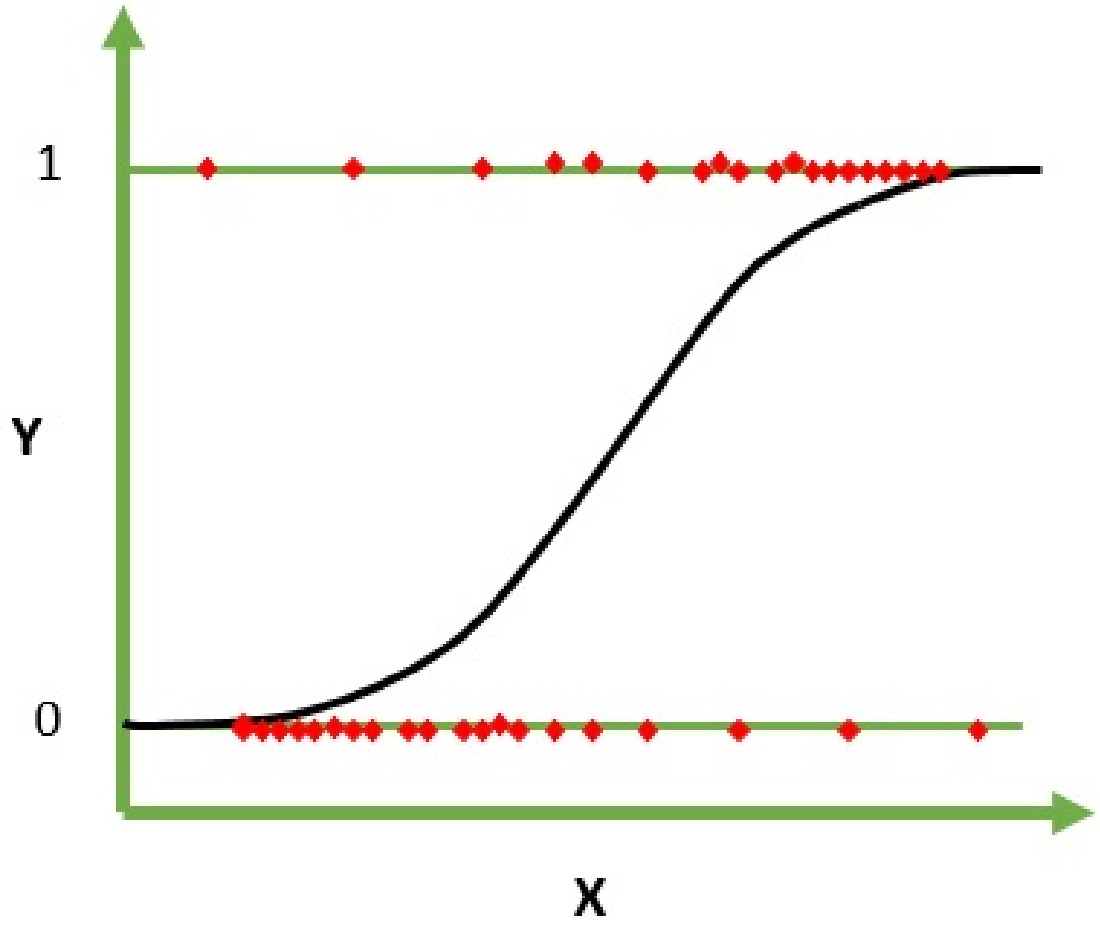

Figure 2.11: Plot of logistic regression, which predict a binary label [PLI02, tec19, PW78].

The logistic regression technique contains dependent variables. The variables are binary values, which means the outcome could only be true or false. Typical use is to find the probability of a successful or failed event [PLI02, PW78].

Logistic regression uses logistic functions to identify the probability P of an event. This event is affected by one or more variables [PLI02]. In Figure 2.11, we show a plot of the classifier.

We will use a binary classification, since a match can be true positive, or false positive [Nas07]. Specifically, we train a logistic regression model on data with known outcomes. This model is later applied in the production system to predict whether a PSM is a true or a false positive [ZSJ+18, MBY+16].

2.6 Big Data Architecture

The huge amount of data generates a need for reliable, highly available and robust systems. This leads to big data concept - unpredictable amounts of data are collected and made ready for processing. In general, big data systems can be divided into three components [Wam16, KKKG14, CML14, Cur16]. The first component is the storage. The data needs to be stored to minimize transfer, growing capacity and improve fault tolerance that is mostly realized using a database management system. Second component is the computation, to process the stored data as fast as possible. The last component is the controlling, which is needed for managing and monitoring the tasks and the resources [Wam16, YHG+16, CML14].

Different network systems have different trade-offs. The CAP-theorem describes, that network shared-data systems can guarantee two of the three properties of consistency[11], availability[12] and partition tolerance[13].

Hence, only two of the properties can be combined. A visualization of the CAP theorem is shown in Figure 2.12 [Bre00].

Based on the CAP theorem, a system can be implemented as follows:

- Consistent and Partition Tolerant (CP) – banking, because the transactions need to be consistent and the data should not get lost if some parts of the system crash.
- Consistent and Available (CA) – relational database systems, because these database systems are consistent and replicated. These systems are vertically scalable and prone to crashes.
- Available and Partition Tolerant (AP) – cloud computing, because in this case several servers are connected and the processes as well as the data is distributed over the cloud. Some crashes would not disturb the availability but it takes time to spread the data over all nodes.

In this work, we use a cloud computing architecture, which is placed in the area of availability and partition tolerance (AP) in Figure 2.12. The databases in this area are distributed NoSQL systems and follow the BASE criteria (Basically Available, Soft state, Eventual Consistency). In contrast to the ACID[14] principle of relational database management systems, the BASE principle has more flexible consistency that brings the benefits of horizontal scalability and parallel computing [Pri08].

In the following, we focus on streaming systems and explain first the architecture for cloud systems and then some of the technologies that can be used.

[11] Every node in the network is consistent, and every user has same view on the data.
[12] Availability guarantee a response of the system in acceptable time.
[13] The system stays available if nodes crashes.
[14] Atomicity, Consistency, Isolation, Durability

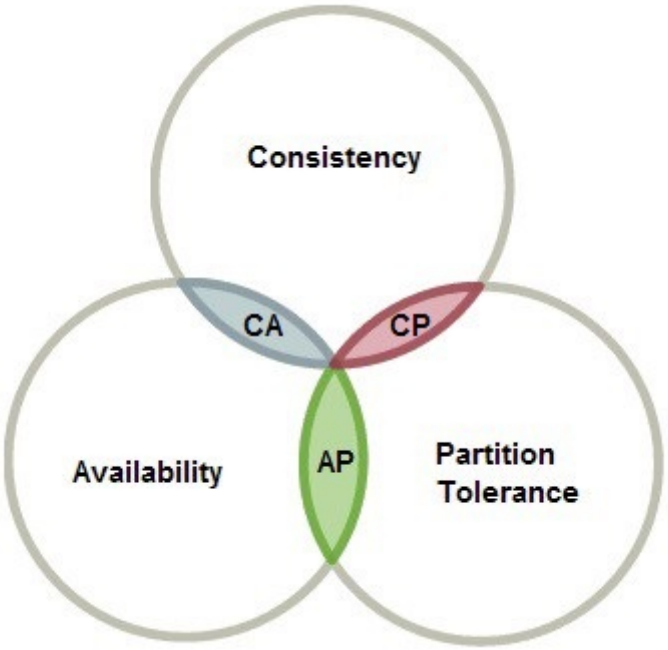

Figure 2.12: CAP theorem, which says that a network shared-data system can guarantee only two components of consistency, availability and partition tolerance [Bre00].

Fast Data Architecture

Processing the data of multiple mass spectrometers is a task with unpredictable amount of data and high availability is required. In addition, the data and the processing needs to be horizontally scalable to effort the calculations on the one hand and to distribute the data on the other hand. In this case, a highly consistent system is not required. Hence, our work is a cloud system with the properties "Availability" and "partition tolerance".

In general, the following properties are required for the stream-based mass spectrometry analysis platform:

- **BIG DATA:** Distributed storage for unpredictable high amount of data from multiple mass spectrometers, users and other sources
- **AVAILABLE:** High availability to allow processing of data in a fault-tolerant way
- **FAST PROCESSING:** Scalable processing engines to calculate results in real time and scale on demand
- **MESSAGE QUEUE:** Fault tolerant message service to queue the incoming messages
- **SCALABLE SYSTEM:** Extensible cloud operating system to scale on demand

To transform the mass spectrometry workflow to a streaming asynchronous pipeline, the fast data architecture seems to be applicable. The fast data architecture is a technology pipeline to process possibly infinite data streams in near real time, using state-of-the-art big data technologies [Wam16, Wam15, KPB+17, KPB+19, AÇ09, Eva11]. A popular fast data implementation is the SMACK stack.

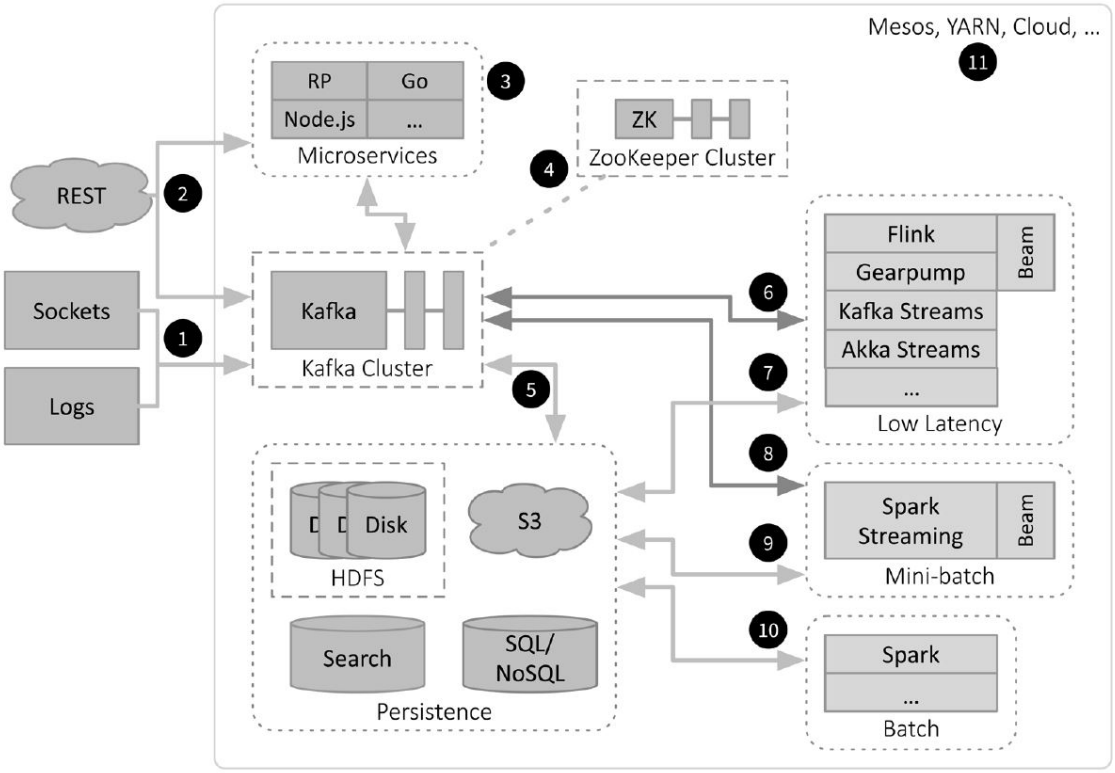

Figure 2.13: A general Fast Data streaming architecture [Wam16].

In Figure 2.13, we show the general fast data architecture. Incoming data enters directly as messages via HTTP or socket (Figure 2.13–2), directly from a device (Figure 2.13–1), or microservice (Figure 2.13–3). A distributed cluster manages the incoming data (Figure 2.13–4). The data stream is either directly stored (Figure 2.13–5) or analyzed by a cloud processing engine (Figure 2.13–6, 8). The processing components communicate with the persistence layer (Figure 2.13–7, 9 and 10). The whole system runs on a cloud operating system in order to schedule the components (Figure 2.13–11) [Wam16].

The Fast Data architecture is the so called “evolution” of Big Data, which proposes an additional streaming component to the state-of-the-art cloud technologies. The architecture proposes a real-time analysis of data streams, using horizontally scalable applications on each step of the pipeline [Wam16, Wam15, Cur16, AÇ09, Eva11].

The SMACK stack is a popular implementation of the fast data architecture. The technologies of the stack are Apache licensed, free and open source. The main goal of the stack is to enable an extensible and horizontally scalable software solution for tackling probably unlimited amount of data. An overview of how each of the

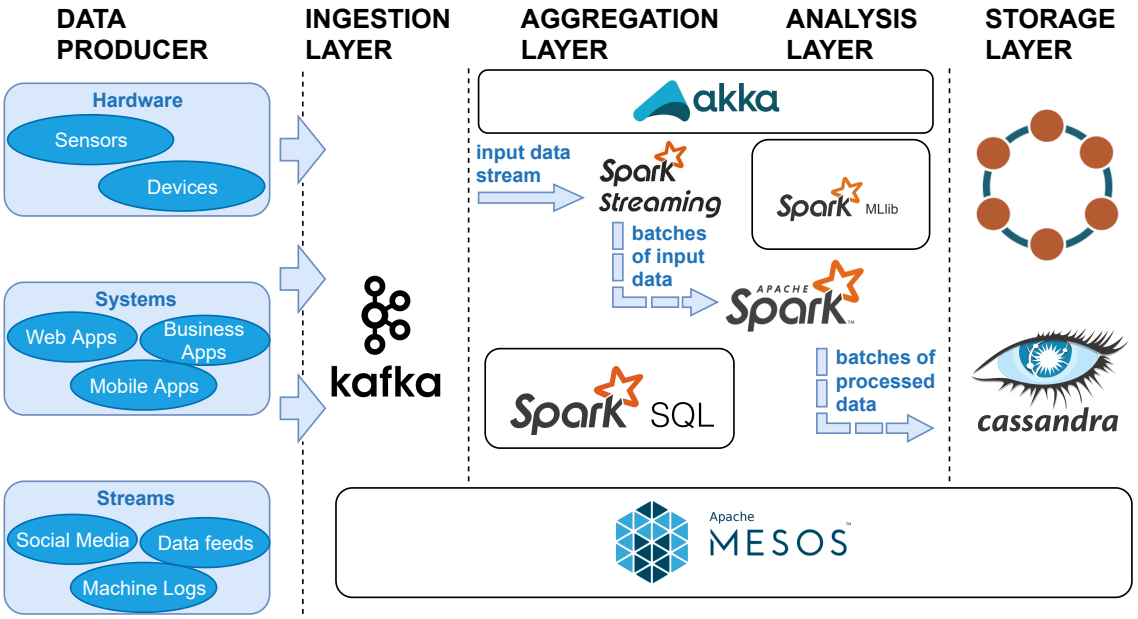

Figure 2.14: SMACK (Apache Spark, Mesos, Akka, Cassandra and Kafka) stack overview containing the roles and connections between the components [Est16].

technologies interacts with each other can be seen in Figure 2.14 [Est16, Wam15, Wam16, ZWLS15].

Several producers send messages to the message broker, which is in this stack Apache Kafka. The data producers can be HTTP[15] messages, websocket streams or other protocols from devices and sensors. Kafka queue all the messages in topics. Apache Akka and Apache Spark consumes the messages from the Kafka topics and process it. Processing the data includes database access or access to external data sources as well as transformation or storage of the incoming data. To process the data Apache Akka and Apache Spark is used in this stack. As database management system, Apache Cassandra is used. Apache Mesos schedules all the services on the platform. This stack is one possible technology pipeline and the components needs to be chosen individually [Est16, Wam15, Wam16, LLP+12, BGM+17].

Hence, we meet all the required properties using the SMACK stack.

- **BIG DATA:** Apache Cassandra
- **AVAILABLE:** Apache Mesos, Docker
- **FAST PROCESSING:** Apache Spark
- **MESSAGE QUEUE:** Apache Kafka
- **SCALABLE SYSTEM:** Apache Mesos

Finally, all the technologies are replaceable if the required properties are fulfilled.

[15] Hypertext Transfer Protocol

2.7 Big Data Technologies

The opportunity of technologies grows fast and results into the so-called big data landscape. The big data landscape, which is shown in Figure 2.15, presents possible technologies that can be installed as different components in the big data stack [Tur18, SÇZ05, BGM+17].

For each task and goal, there are many different configurations, which result in a huge pool of possible technologies, represented with the help of the landscape model. In the right bottom corner of Figure 2.15, a piece of the landscape has been zoomed in to show how many technologies exist only for a special data structure graph. Eight different databases can be chosen – Neo4J, Amazon Neptune, IBM Janus Graph, Oracle Spatial and Graph, OrientDB, Apache Giraph, InfiniteGraph, Objectivity [Tur18]. This means, for implementing a pipeline, the technologies are not important. The architectural technology decision is based on the data structure and on the applicability for a specific use case, licenses and knowledge [Tur18, Wam16].

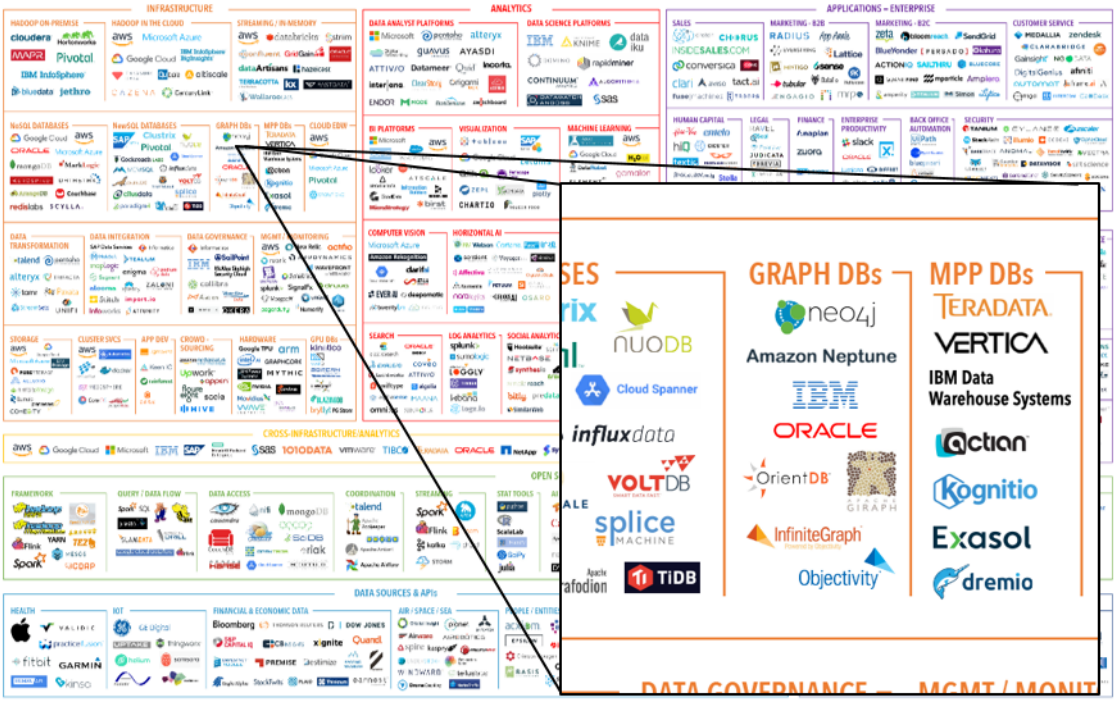

Figure 2.15: The big data landscape and shown graph database technologies [Tur18].

In our work, we analyze the protein identification process. Therefore, we use cloud architecture to allow horizontal scalability. Firstly, a cloud operation system is needed. Secondly, our system needs a column oriented database system to allow range queries on the protein data. Thirdly, a distributed processing engine is needed to perform the search and lastly, our system needs a message broker to manage the incoming messages. A SMACK stack is a possible technology stack for our platform, but other technologies with equal properties can be used for the components.

As the big data landscape has been shown, further in this section, we focus on the significant technologies used in our work.

2.7.1 Docker

In a cloud environment with many components it is hard to solve the "dependency hell". Every application has different dependencies and libraries in use. Docker tries to solve such problems isolating the application together with all the needed dependencies. A Docker container is like a small virtual machine containing one complete application. For example, a java program needs PostgreSQL as a data storage, Redis for caching and Apache as a Web server. Each of these components has own libraries and dependencies that might conflict on one machine with other components. By packaging everything in one container, each component and its dependencies are isolated [Mer14, Neg15].

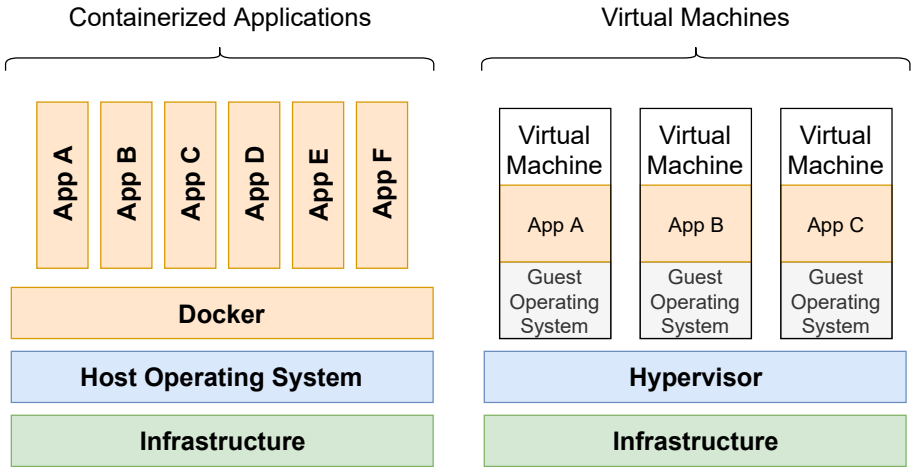

Figure 2.16: Docker architecture runs the application in isolated containers instead of running it in virtual machines with own operating system [Kra19, Mer14, Neg15].

In Figure 2.16, we show the isolated Docker containerized applications and the virtual machine organization of applications. Main difference is, that Docker uses the host operating system and minimize the size of each container and accelerate the boot operations of the container. On the other side, a virtual machines has own operating system, which is slow on startup and has a full copy of the operating system taking gigabytes of storage. On Docker, each application contains all dependencies that are needed to run. Docker is an additional layer that manages the containers. Each Application is in one virtual machine on the Docker environment. Hence, multiple container applications share only one operating system kernel on a single machine. Apache Mesos includes the Docker layer on each node, which allows to manage micro services on the whole Mesos cluster. The service is reachable through port forwarding from the host operating system to the virtual container network of the application.

2.7.2 Mesos

Apache Mesos represents the operating system and abstraction layer over the available computer resources. It monitors hardware resources such as CPU, available memory and storage allocation [HKZ+11]. It can operate on physical or virtual machines. Apache Mesos manages available system resources and orchestrates containers and jobs on the distributed system [UAK18]. Mesos can be classified as

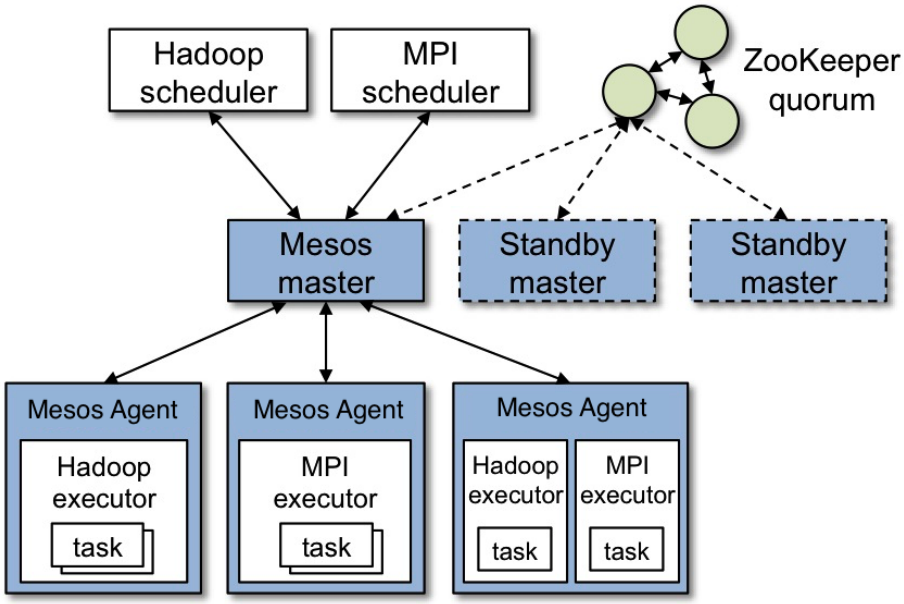

Figure 2.17: Apache Mesos master node interacts and manages slave nodes [The19b, HKZ+11].

cloud operating system and manages services among the nodes like tasks on a single machine.

The architecture of Mesos is build on a master node, which handles multiple slaves [The19b, HKZ+11]. The slaves manage service components, so called Mesos frameworks and the master monitors and orchestrates them. The Mesos master agent offers available resources to slave node in order to run services. It is recommended to use more than one master node in cases of failure.

2.7.3 Spark

Apache Spark plays the role of the processing engine in the big data architecture. The main task of Spark is to analyze data in real time and perform analytic workloads. Apache Hadoop or Apache Flink are alternative technologies to it. Apache Spark promises to be 100 times faster than Hadoop on batch and streaming processes [VMJ16, ZCD+12].

Apache Spark is available in Scala, Python, Java, R and SQL, which brings good flexibility from developer perspective.

Spark provides the infrastructure and the running of the worker of a program. The key property is the way of data handling in Spark. It loads the data into resilient distributed data sets (*RDDs*), that allow fault tolerance, efficiency, speed and in-memory data storage [ER16, ZCD+12, Pam16]. Furthermore, the RDDs enforce immutability and have no negative effects from parallel running jobs that interfere. When Spark is running on a cluster, a Spark driver program delegates the work as jobs to its worker nodes (Figure 2.18). This enables horizontal scalability and is perfect for working in the cloud environment. The master node of Apache Mesos can be used as Spark driver and spreads the workers over the Mesos cluster. Hence, on a SMACK stack, Mesos takes care for the resource management.

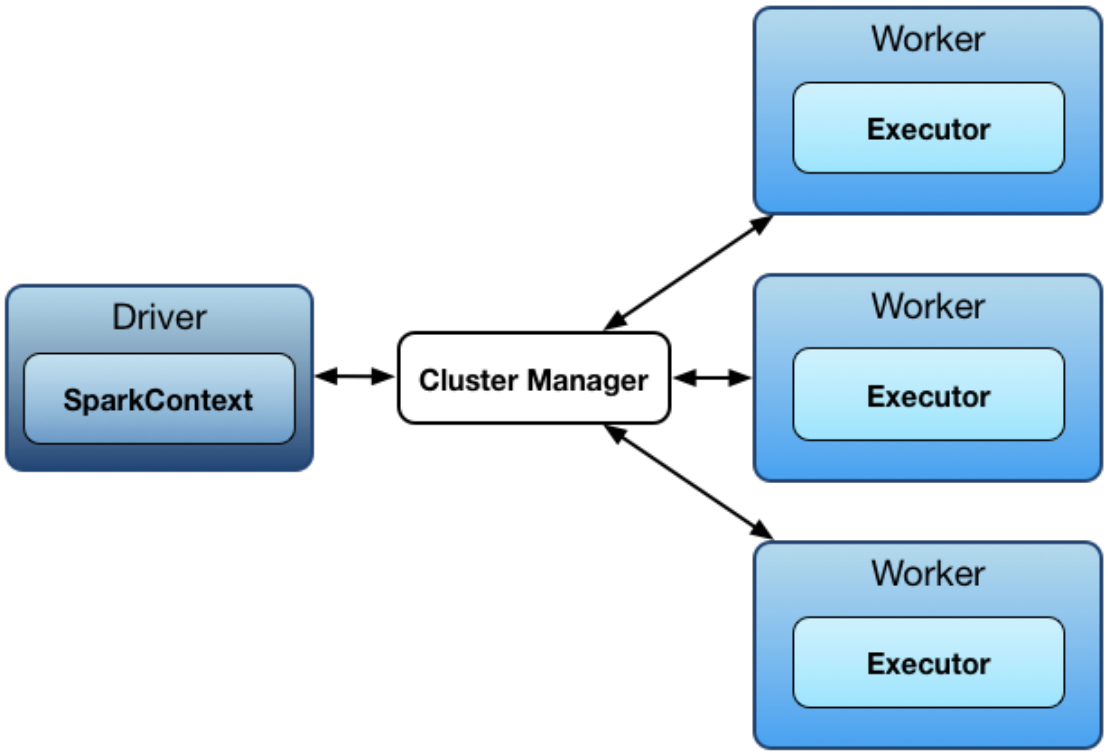

Figure 2.18: Apache Spark cluster managing three worker nodes [Las19].

2.7.4 HDFS

Hadoop Distributed File System (HDFS) is a highly available file system for the storage of a huge amount of data. The file system is distributed on multiple nodes and, hence, their files are stored separated on multiple nodes. Therefore, HDFS provide master and slave nodes [SKRC10, Whi09].

HDFS is mostly used to store files from different nodes and services in one environment. Additionally, the distributed file system can provide parallel access, which depends on the amount of nodes used for the system. Most of the processes in mass spectrometry analysis are file based. Hence, HDFS is a good way to store the files and transform the data from the files into the structured database management system such as Cassandra [SKRC10, Whi09, Has15].

2.7.5 Cassandra

Apache Cassandra is a column oriented, wide-column distributed database management system [The19a, Nee15]. The NoSQL DBMS Cassandra distributes the data over nodes in partitions and guarantees fault tolerance and horizontal scalability. Similar technologies are HBase or Google Cloud Big Table [Gar14, SF13, CDG$^+$08, SSH10].

Apache Cassandra has no relation between the data and stores the data in a schema free fashion. The architecture of the system is a ring hierarchy of nodes, where each node can fulfill the same role [WLZZ14, LM10, Nee15]. In Figure 2.19, we show the ring architecture of Cassandra database management system. In the ring, each node

can be the coordinator for the client and the each row data is replicated through the nodes. If a node is not reachable, the replicated data can be used.

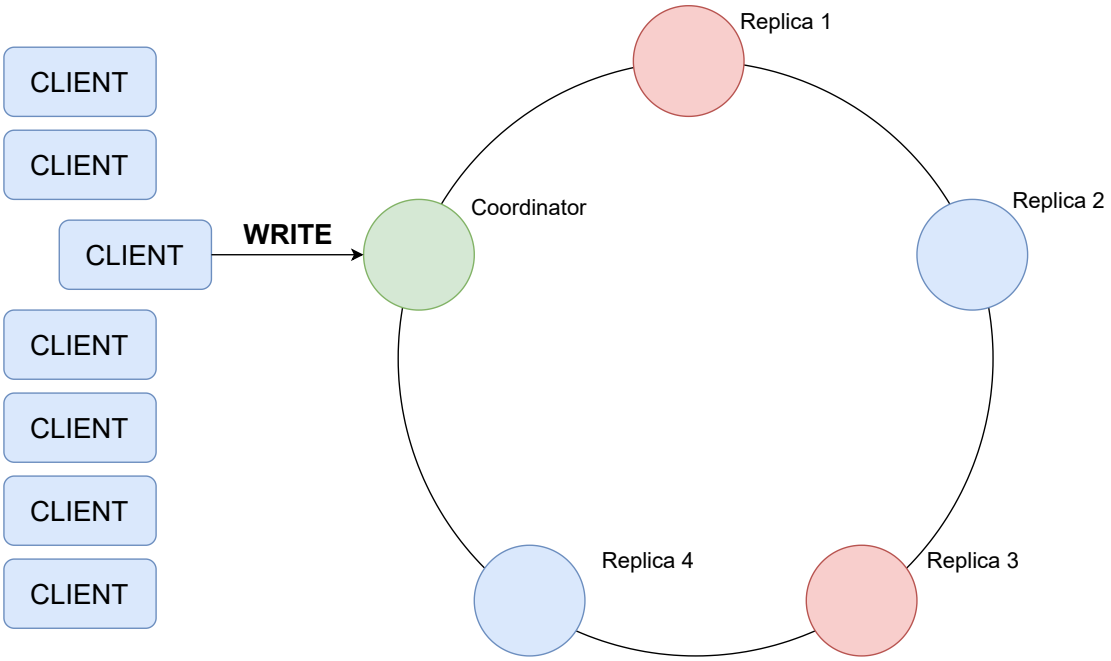

Figure 2.19: The ring architecture of Apache Cassandra database management system [avd19, Nee15].

2.7.6 Kafka

Apache Kafka is a message broker providing a horizontally scalable cluster and several APIs to interact with the cluster. A Kafka cluster, so called broker, stores messages as a key-value message in topics. Topics are stored in partitions, which are distributed and replicated on the brokers. The messages are queued in the partitions. The producer sends messages to a specific topic and does not need an answer. The consumer subscribes on a topic and gets the messages as they appear in the topic [KNR11, NSP17, Gar13].

The Kafka message processing can be seen in Figure 2.20. Multiple producers send messages to one or more topics. The messages in the topic are stored fault tolerant on the hard disk. Consumers are subscribed to topics and react on messages as they arrive. Additionally, consumers can be grouped, which allows more flexible event processing in the system [ER16, KNR11, NSP17, Gar13].

2.8 Conclusion

In this chapter, we introduce the basic biological background for understanding the state of the art processes of mass spectrometry analysis and possible software technologies to implement the processes. We focus only on the processing pipeline of mass spectrometry data, which is for protein identification on the biological side and we introduced the fast data architecture and possible technologies of the architecture that can be used for the implementation on the information technology side. We

already propose the technology stack, which can be used for the implementation of our work. In the next chapter, we introduce the challenges and requirements of the work.

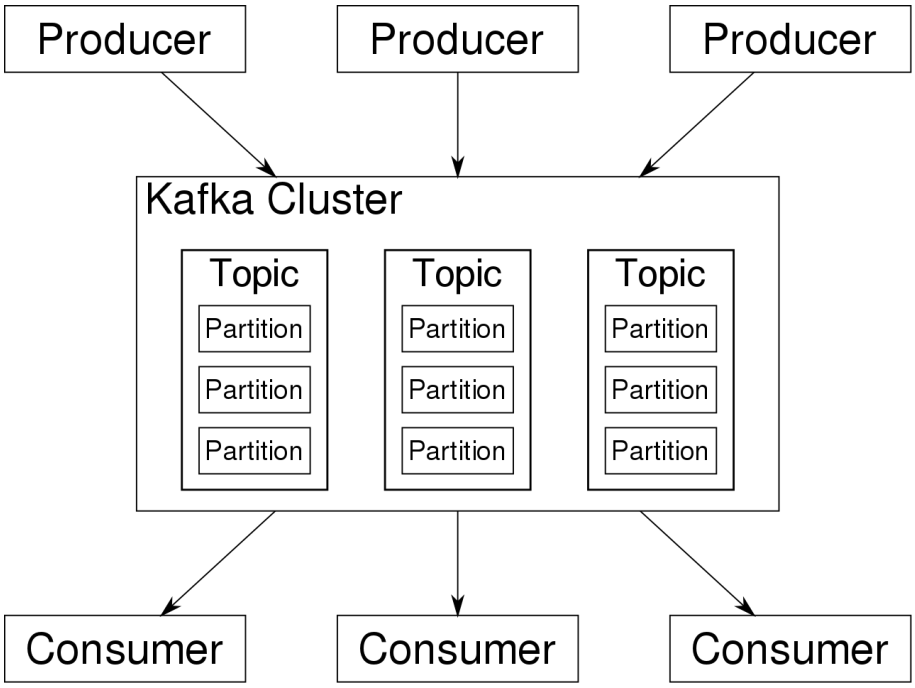

Figure 2.20: Apache Kafka architecture with sending producer, the brokers and the consumers [wU19, ER16, KNR11, NSP17, Gar13].

3. Challenges of Streamlining the Mass Spectrometry Analysis

The growing quality of mass spectrometers and the growing amount of data reveal new use cases of mass spectrometry. For instance, researchers show the applicability of proteomics and metaproteomics research for clinical diagnostic [PF17, ECL+12, BFWSS+15, XYF+17]. However, the amount of data and the performance of data processing is an additional bottleneck for the applicability of such use cases [HSZ+17, HBS+93, MBR+13, CNM+11, RR03, Tab15].

In this thesis, we implement a real-time highly scalable analytic platform for mass spectrometer analyses showing the applicability for real-time use cases such as clinical diagnostics [PF17, ECL+12, BFWSS+15, XYF+17]. Additionally, we leverage the complete analytic platform to a cloud system.

As a first step, we describe the challenges and the research tasks of the thesis in this chapter. Firstly, we clarify shortly the sequential state-of-the-art workflow and the performance of the workflow. Secondly, we explain possible optimizations of the workflow. Finally, we show possible analysis of the result data proposing the complete pipeline [Zou18].

In Summary, we make the following contributions in order to answer research question RQ1:

- Protein identification on fast data: *Analyzing the applicability of protein identification process on a fast data architecture.*
- Challenges: *Analyzing the components and the challenges to apply the data analysis pipeline to a fast data architecture.*

3.1 Sequential State of the Art Mass Spectrometer Workflow

This section gives an overview of the typical mass spectrometry workflow in metaproteomics [HKRB15, EMY94]. The workflow consists of four steps, where an asynchronous processing of the third and fourth step is the goal of the thesis. We show the complete state-of-the-art metaproteomics workflow in Figure 3.1. In contrast to Figure 2.1, we add the approximated duration of these steps. The duration is based on the experience of scientists, who work with a mass spectrometer. Additionally, the identification and validation is combined in the fourth step in Figure 3.1 [HKRB15].

The first step is the sample preparation in the laboratory and needs days to complete, the second step is the measurement of the samples, which takes up to two hours per measurement of one sample and the third step is the conversion of the digital signal data into a readable standard format. The third step needs hours to write the data. The last step is the *identification step* and analysis of the data. This step can take days depending on the tools used for the analysis and data exploration [HKRB15, AM03, LBK+17, Ast]. In the following, we explain each step in detail.

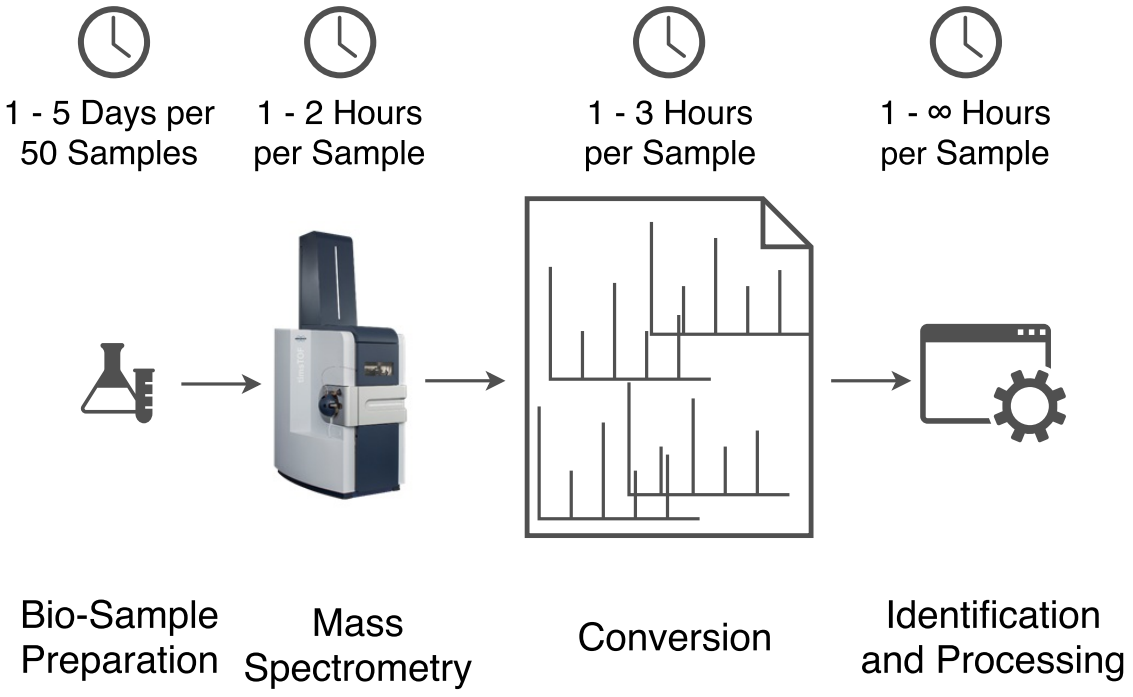

Figure 3.1: The summary of the current, sequential mass spectrometry workflow with approximated processing time.

3.1.1 Bio-Sample Preparation

The input of a metaproteomics workflow is an experimental sample from a biological environment, for example a sample from a biogas plant or human gut. In a laboratory, the proteins are separated and cleaved (broken up) into smaller peptides, which are the output of this step [HKRB15, AM03, LBK+17, Ast].

Usually a metaproteomics experiment requires the preparation of approximately 50 samples. This includes replicas to increase the fault tolerance in the analysis of biological data. This procedure takes one to five days of preparing the samples for the mass spectrometer [HKRB15, AM03, LBK+17, Ast].

3.1.2 Mass Spectrometry

In proteomics, the technique of tandem mass spectrometry is commonly used. However, it measures the mass of the peptide (MS1) and of fragment mass differences (MS2) [Ast], which are stored as a so-called spectrum in a file. The important information are the MS2 experimental spectra and the connection between the precursor intensities (MS1) to it. The attributes mass-over-charge-ratio and intensity pairwise, represent one peak of a mass spectrum [HKRB15, AM03, LBK+17, Ast].

A mass spectrometry analysis takes two to three hours per sample and the measurement of the typical 50 samples would take about a week of mass spectrometry measurements. Each experiment resulting from the sample measurement is written to a RAW file. This RAW file is in manufacturer depending format and stores all the data collected from the mass spectrometer. Hence, the RAW file contains all the signal data, which includes many noises. Furthermore, depending on the mass spectrometer device and the measurement method, the data needs to be aggregated and filtered before data analysis can. Finally, it is not recommended to start the analysis with the RAW files and furthermore, the most analysis tools start their process with the standard format as input [HKRB15, AM03, LBK+17, Ast].

3.1.3 Conversion to Readable Format

The measured spectrum data are in a proprietary RAW format, which is not readily usable for further processing. Each mass spectrometer device company has its own RAW file format. This is the reason why the workflow needs to convert it into a standard exchange format. As each company has its own RAW structure, different types of tools are used to convert the data, leading also to different output formats. The most-commonly used data formats for tandem mass spectrometry data are the Mascot Generic File or mzML [MTS+04, Deu10]. The conversion step takes up to one hour per sample. During the conversion, the data gets filtered and thus the quality is increased [LBK+17, Deu12, The17, Mat16, HSZ+17].

3.1.4 Protein Identification and Validation

Protein identification is a method to relate the measured experimental MS2 data to the peptides forming a protein with a certain similarity. Since our database (a FASTA[1] file of already identified proteins) contains the amino acids of the proteins, i.e., not peptides, we have to transform them to theoretical mass spectra that can be compared with our sample mass spectra [HSZ+17, HBS+93, MBR+13, CNM+11, RR03, Tab15].

A big overhead is produced because the measured spectra do not exactly map to a theoretical spectrum due to noise. Hence, an N-M similarity comparison is

[1] Text file that contains amino acid sequences as char arrays

needed and a final ranking of similarity scores will produce the resulting identified protein [DAC10, MFT+13]. The result of this step is a set of identified spectra, which relate to a protein in the FASTA file. This identification process is the foundation of further analysis and has to be adapted to the new fast data architecture [HSZ+17, HBS+93, MBR+13, CNM+11, RR03, Tab15].

The general approach of the protein identification is implemented by several groups and companies such as Mascot, Andromeda, Sequest and X!Tandem [RR03, EMY94, PPCC99, CNM+11].

Because of the measurement errors and artifacts, the false discovery rate (FDR) of the result of the identification is needed to assess the quality of the current identification. Briefly, our identified proteins are compared to nonexistent proteins (decoy proteins) which are retrieved by inverting the protein sequence of the FASTA file. A drawback of this is that the protein identification process runs twice, thus, doubling the runtime of this step [EKT+12, MFT+13]. Furthermore, decoy FDR calculation needs the whole sample data at once and does not work for single spectrum data [HSZ+17, MBR+13].

The identified spectra are the results that can be analyzed by the scientists. Especially the interactivity of visualizations have a positive effect for the workflow exploration by a researcher [ZSB+17, CCL+16].

3.1.5 Optimization of the Mass Spectrometry Analysis Workflow

After an overview of the typical state-of-the-art workflow of mass spectrometry analysis, we introduce our idea of the new parallel mass spectrometry analysis workflow. In our concept, the mass spectrometer writes each measured spectrum to the file during the measurement. If we stream the measured experiment spectra to the cloud during the second step of the workflow, we can parallelize the whole procedure after the second step. Hence, the further processing could run for each single spectrum independently after the mass spectrometer measurement (Figure 3.2). This is a typical behavior of Internet of Things (IoT) technology. IoT describes a method, where devices are connected to a cloud system and send their sensor data to the central system [GBMP13]. Since the further calculation should run fast on a streamed input we decided to use the Fast Data architecture [Wam15, KPB+17]. The goal is to overlap calculations with the mass spectrometry measurement time. The best case would be a near real-time processing and the results are being finalized when the measurement is finished [Zou18].

To this end, our concept needs a local service running on the mass spectrometer side. Our platform would collect the converted spectra instead of writing the measured spectrum into the RAW file. The device converts the RAW data into a readable format using existing scripts of the manufacturer and sends the data to a local service, which streams the spectrum as a structured MS1 spectrum into the cloud system. The MS1 spectrum contains several precursors of MS2 spectra. In this way, we can connect a mass spectrometer to the new infrastructure, without changing the existing system. However, procedures have to be adapted to the Fast Data architecture, which is a major challenge for us [HKRB15, AM03, LBK+17, Ast, ZSB+19b, Zou18].

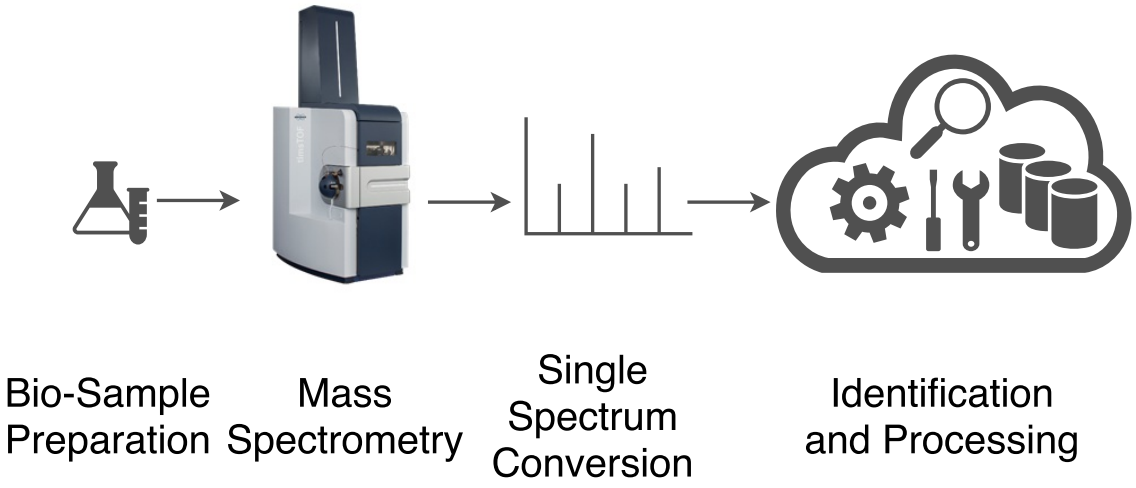

Figure 3.2: The parallel workflow of the metaproteomics procedure. Each spectrum will be processed one by one on a scalable cloud-based system.

3.2 Challenges of a Parallel Metaproteomics Workflow

During our research, we have found several challenges that have to be solved. In this Section, we explain each of the challenges in detail.

3.2.1 Device Streaming Interface

The first challenge is to stream a mass spectrum immediately when it is output from the mass spectrometer. However, the inner workings of each mass spectrometer are special for each manufacturer and highly involved. Hence, the challenge will be to create a general, manufacturer-independent interface, which satisfies the needs of all users [ZSB+19b].

3.2.2 Distributed Data Model

Since our system collects a huge amount of data asynchronously, a horizontally scalable database management system is needed. The storage should provide the experimental spectra and the protein data for the protein identification procedure. Additionally, we evolve the system to a central system with several devices and users connected to it. Therefore, a NoSQL system, which can store such data amounts, is needed [ZSB+19].

3.2.3 Streaming Protein Identification

The next issue is to connect the protein identification algorithm to a per-spectrum data source compared to the current batched file source. In addition, an adoption of the similarity-scoring function and their parallelization can be used for an identification process as a pipeline. To increase the scalability and the performance of the algorithms, we use modern distributed cloud computing techniques [Zou18, ZDS+18, ZSB+19a, MBR+13].

3.2.4 Decoy-Free FDR Calculation

The next challenge is the FDR calculation. As mentioned in Section 2.4, the decoy method needs to search the whole sample data again for the FDR calculation. In our parallel approach, we cannot provide all the information about all the spectra from a sample, because the mass spectrometer streams each spectrum separately. Hence, we need new methods for calculating the FDR without the decoy process. This also halves the run time, due to the skipped decoy search. A decoy free approach using machine learning seems to be promising, but is only used in the Mascot search engine [GSV+15]. We aim to use a cloud-based machine learning system to decide whether the identification is true or not, based on the idea of the decoy free method. First we implement a feature extractor for the results of X!Tandem search engine. Then we generate training data using the state-of-the-art decoy FDR method and evaluate the results using different classificators [ZSJ+18].

3.3 One Stack to SMACK Them All

A streaming fast data architecture evolves the field of big data and shall show a high performance and scalability on streaming data scenarios [Wam15, KPB+17, KPB+19].

All the requirements we have generated from the challenges and the optimization analysis can be fulfilled with the so-called SMACK stack, an implementation of the fast data architecture [Est16, ER16, KPB+17, KPB+19]. The SMACK stack is one implementation of the fast data architecture and the components can be replaced.

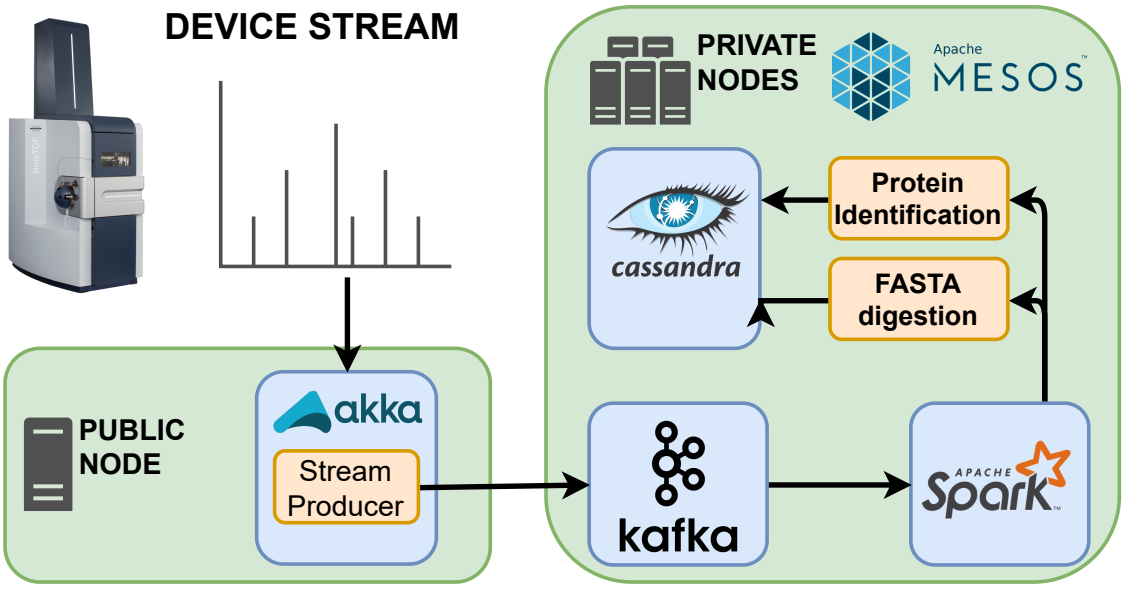

Figure 3.3: New fast data architecture using a SMACK stack with active services for a protein identification task. Green marks the infrastructure, blue marks the technology and yellow marks the applications.

The idea is to combine cloud technologies for creating a highly scalable pipeline for mass spectrometry analysis (see Figure 3.3). In the following, we match each used technology with the challenges that they should address.

Since the data is streamed per MS1 spectrum, which is a collection of MS2 mass spectra, a batch processing cloud engine is needed (see Section 4.2.1). However, Apache Spark seems promising for this component, since the Spark Streaming library processes incoming batch data faster than other engines [Las19, KPB+17, KPB+19, MBY+16].

Consequently, the device streaming needs a collaboration with a manufacturer, because the conversion needs data insights of the measured signal data. Therefore, a collaboration with Bruker Daltonik GmbH is needed, because they are the manufacturer of the in-house mass spectrometer. Since our platform should work with many connected devices, the Apache Kafka server seems very well suited for the communication [NSP17].

The Apache Spark is chosen for our prototype, Apache Mesos is a good choice, because the Mesos master node can be used as spark master node to acquire the maximum resources of the cluster for the processing engine [HKZ+11].

Accordingly, the protein identification processes spectrum-wise, we need to query all possible peptides from the database. Therefore a column-oriented index is needed, which provides wide tables for the data. In this case, the Apache Cassandra database management system fulfills all the requirements for the task [LM10].

Finally, all additional services can be implemented using Apache Akka, software toolkit, which implements the actor model or other microservice frameworks [ER16].

Of course each of technology can be replaced by similar tools. Finding the best combination, however, is one of the future tasks, since we use the standard stack of the fast data architecture for our prototype.

3.4 Interactive Web-based Analysis of Metaproteomics Results

The proposed cloud architecture provides benefits for data processing. However, for full cloud based solution, the researchers need also the capability to analyze the results. Especially explorative analysis needs interactive visualizations. In this section, we focus on complex metaproteomics results and show how to increase the quality of the explorative analysis of the data using web-based interactive visualization [ZSB+17].

A key challenge of metaproteomics is the discovery of major relationships between microbial taxonomies and biological functions. Proteins detected by metaproteomics experiments can provide a quantitative relationship between functions and taxonomies [HKRB15]. A typical question is which family of gut bacteria (taxonomy) is linked to a biochemical process that is associated with a disease (function). In a study that finds over 10,000 proteins, a comprehensive visualization of the relationships would improve the analysis process, because this currently involves sifting through extensive lists of proteins [PF17, ECL+12, BFWSS+15, XYF+17].

Metaproteomics experiments yield mass spectra, which identify the proteins. These proteins are associated with multiple functions and a single taxonomy (see Figure 3.4). Protein functions are represented by keywords, which are grouped into

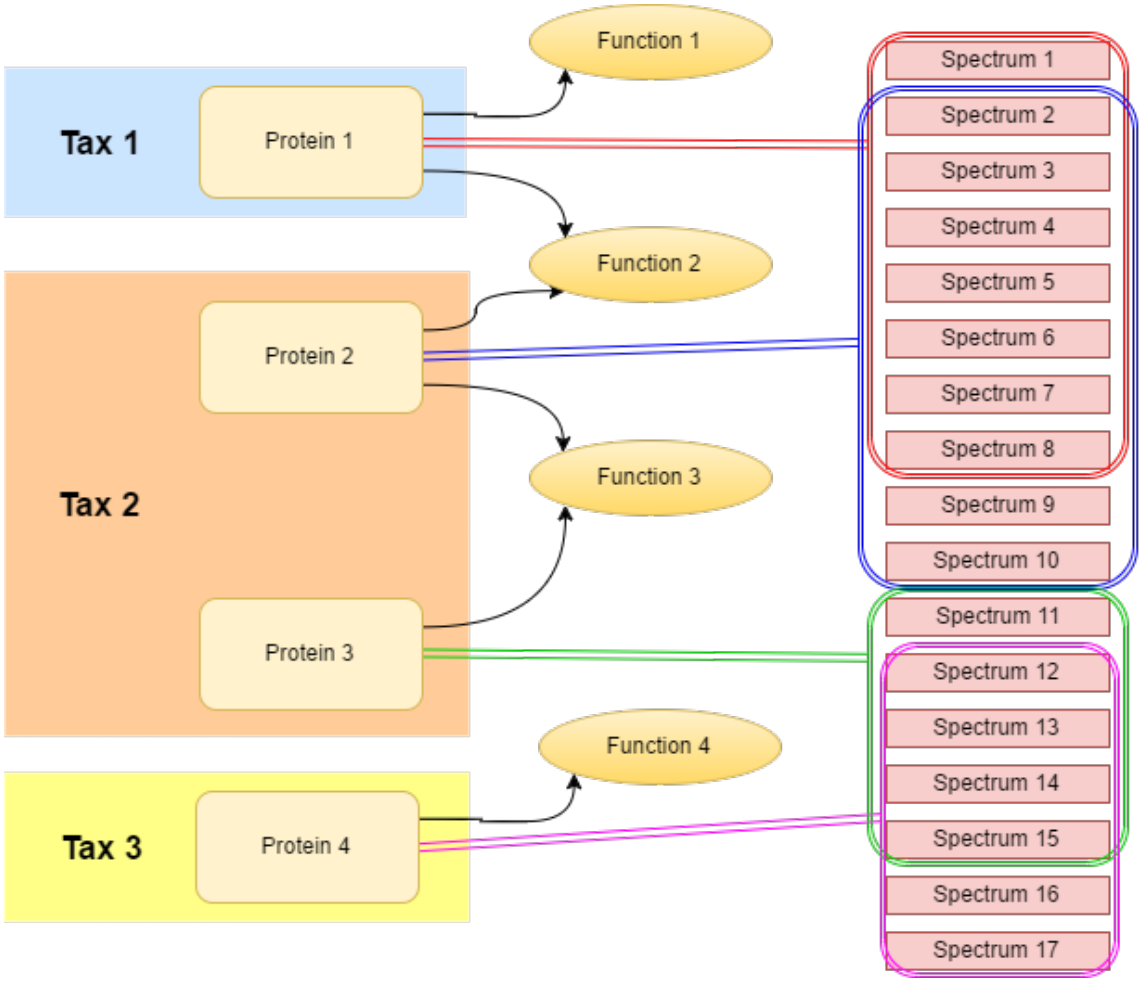

Figure 3.4: The relation between mass spectra, proteins, taxonomy and biological function, which is represented by the crossmaps.

categories. Taxonomies are arranged in a directed acyclic graph and are categorized into ranks. Approximately 1.5 million taxonomies are currently stored in the NCBI protein database[2] alone. An intersection matrix that represents the relation between taxonomies and functions has to be created from the protein data. This intersection matrix as visualized by the crossmap in Figure 3.5 has been established previously using the MetaProteomeAnalyzer software [MBH+15].

The crossmap plots taxonomies of a certain taxonomic rank against functions of a certain category. This visualization approach has several drawbacks for the metaproteomics use case. First, the vast majority of theoretically possible relationships is not present in typical experiments, but will still be drawn in a crossmap and dilute the actual information. Second, a crossmap will visualize absolute quantification, but the use case would typically require a relative quantification (i.e. how much percent does each taxonomy contribute to a certain function). This problem worsens, as experimental data can span several orders of magnitude. Finally, the amount of data poses a problem for crossmap visualization. A large experiment can quickly produce a relationship-matrix with size 200x200 of which most entries would be empty. In

[2]National Center for Biotechnology Information

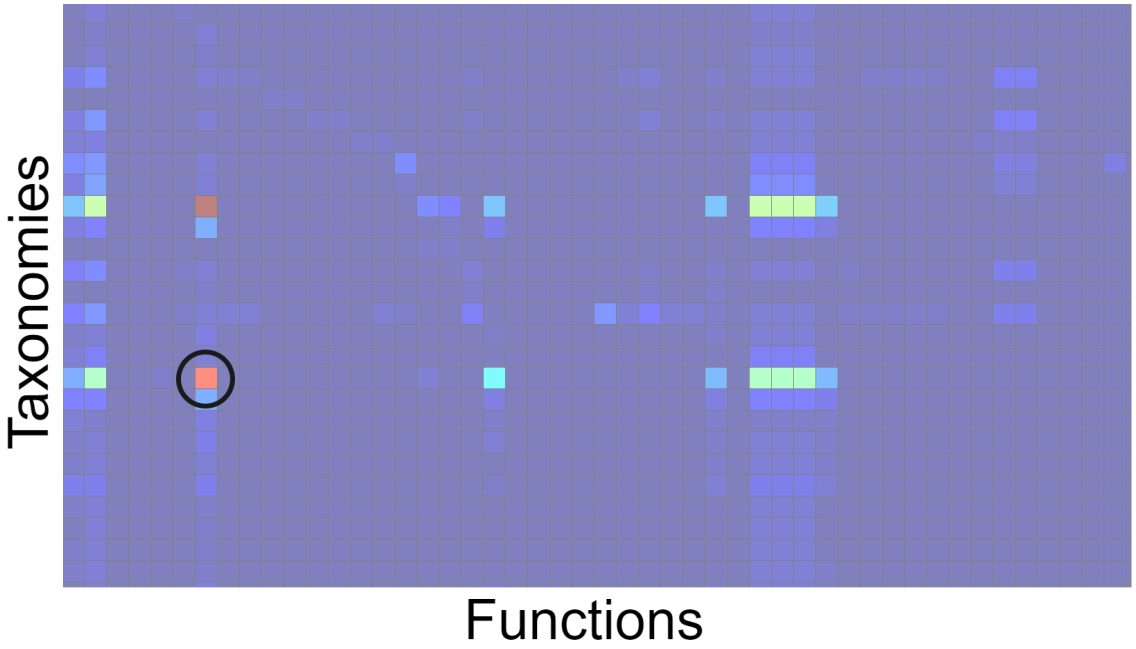

Figure 3.5: A snippet of a crossmap generated from metaproteomics data using the MetaProteomeAnalyzer software [MBH+15]. The highlighted red intersection marks the function-taxonomy relation with the highest amount of spectra. The intersection matrix contains mostly empty entries.

conclusion, a new approach is desired, to deal with the challenges of big data and present large amounts of data in a condensed human-readable format [MY04].

Different methods are used to visually present relational data. One visualization technique is a chord visualization approach of related objects, which is based on the hierarchical edge bundles plots. These plots show a network of entities in a circle diagram [Hol06]. Another approach is the crossmap method, which is used for cross relationships between objects. Two entities of two different entity groups are cross-plotted, to show the impact of their relationship on the intersection point (Figure 3.5) [MY04].

There are different tools to visualize relational data based on both crossmaps and chord diagrams. Circos is a chord visualization tool, used for the identification and analysis of the results of genome comparison [KSB+09]. This kind of visualization becomes more and more popular and new frameworks and tools are being created. One of them is the BioCircosJS framework [CCL+16]. This API is built based on the d3 visualization framework and is used for creating Circos plots of biological data. The primary use of this framework is to plot genome data and explore it. These tools focus on genome analysis and are difficult to use for other data. Furthermore, the Circos tools cannot generate interactive visualization. In addition, both tools require programming skills and are not user-friendly. The crossmap visualization can become confusing with increasing amount of relations to be displayed. For our work, we used the d3-chord library for the d3 framework, that provides tools for creating different kinds of chord diagrams [Bos16], unrelated to the domain of biology.

3.4.1 Transformation from CSV to Chord Diagram

In this section, we describe the format of the input data and how we compute the matrix for the visualization. We also show how the user can choose the color spectrum of the diagram.

3.4.1.1 Data Model

The first step of the transformation is to create a suitable data model. The relational data between functions and taxonomies needs to be comparable and therefore only taxonomies of a specific rank and protein functions of a specific category are used. In our approach, we use an intersection matrix as the input format for the chord diagram as described by Holten et al. and Krzywinski et al. [Hol06, KSB+09]. However, most of the entries in this matrix are zeroes making it unreadable for the user. Furthermore, the creation of this matrix can be complex depending on the data stock. To counter these drawbacks we use a relational CSV storage that only stores data of related entries. This concept is similar to the relational storage in data warehouse cubes [Col96]. In Figure 3.6 we show an example of our data model.

Object 1 and Object 2 are related. Impact 1 is the value, which describes how often object 1 occurs in object 2 (number of spectra). Impact 2 describes exactly the opposite: how often object 2 occurs in object 1. Both values are real numbers bigger than zero. In our tool the value of impact 1 and impact 2 are equal. Nevertheless, we let the possibility to control the visualization with the second value for the future. The benefit of our simple data model is its human-readable format and its general applicability.

3.4.1.2 Transformation Computation

The next step of the transformation is the client side computation of the chord matrix, which is done similarly to the code by delimited.io [Del14]. Regarding the relational input format from the section before, we need to transform it to the intersection matrix described in [Bos16]. The object pairs are built together and the impact value is used as their cross product. During this step, the local filter is installed. This filter contains all objects and controls their visibility in the chord visualization. The client side development makes the connection between different data sources possible (cloud storage, web service). With a GET request, we can import CSV strings and files using the general CSV format defined above.

3.4.1.3 Dynamic Color Range

The last step of the transformation is to define the dynamic color variety of the chord visualization, which allows the user to create an individual representation of the data. The colors are interpolated using an RGB interpolation function. The interpolation function assigns a color to an element in the chord visualization. The example in Figure 3.7 shows the interpolation between red and blue. Thereby, adjacent elements are assigned different colors to better distinguish them.

The definition of the color spectrum is done with different blocks, which we call two-color-gradient blocks and the offset to pick the colors. As shown in the figure, the

Id	Object1	Object2	Impact1	Impact2
1	Bacillaceae	Cell Devision	130	130
.	.	.	.	.
.	.	.	.	.
n	StringObject1	StringObject2	IntNumber1	IntNumber2

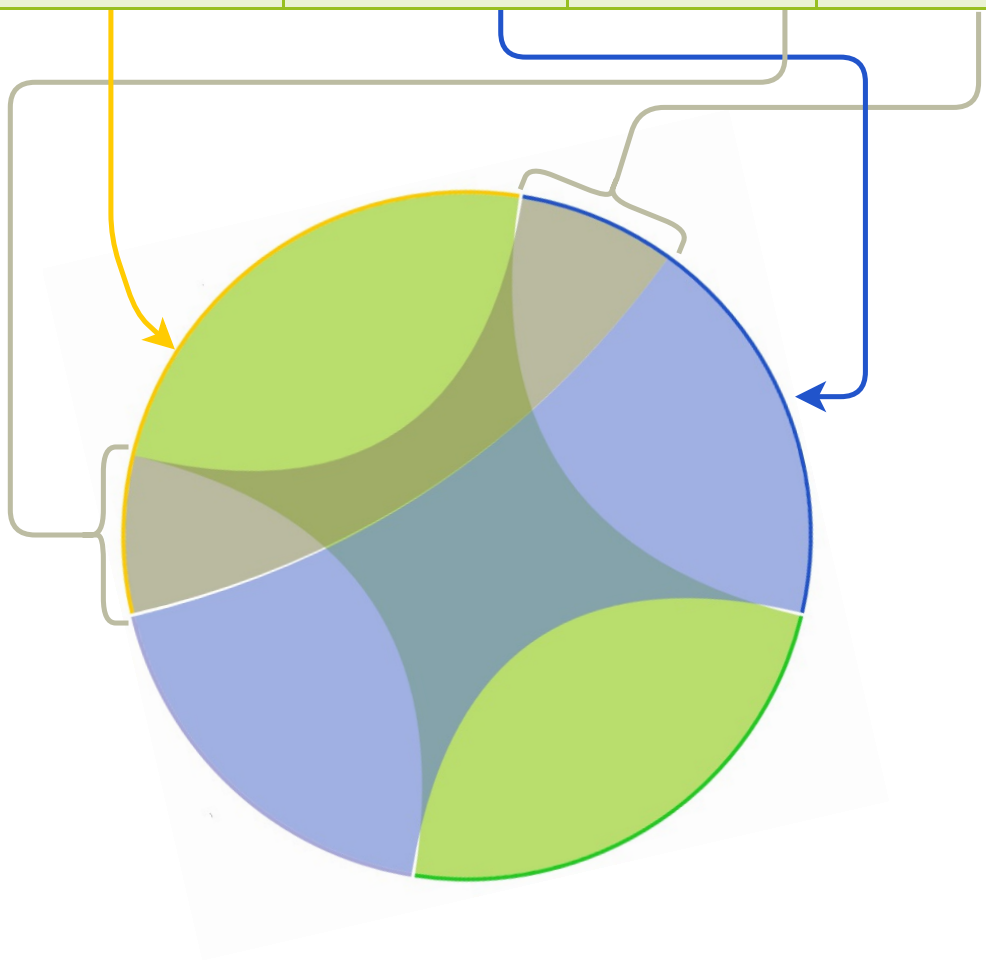

Figure 3.6: Transformation of the input data to the visualization

color picker iterates through all blocks and picks an interpolated color for an element in the chord visualization. The color picker starts back from the beginning after the last block. The user can define the different two-color-blocks and determine their colors. Additionally, the user also defines the offset for the color picker. The color schema is represented by a JSON array and can be used as an exchange format.

3.4.2 Features of the Interactive Chord Diagram

The usage of interactive diagrams has many advantages compared to non-interactive ones. For example, interactive diagrams are better for analysis of big data and for

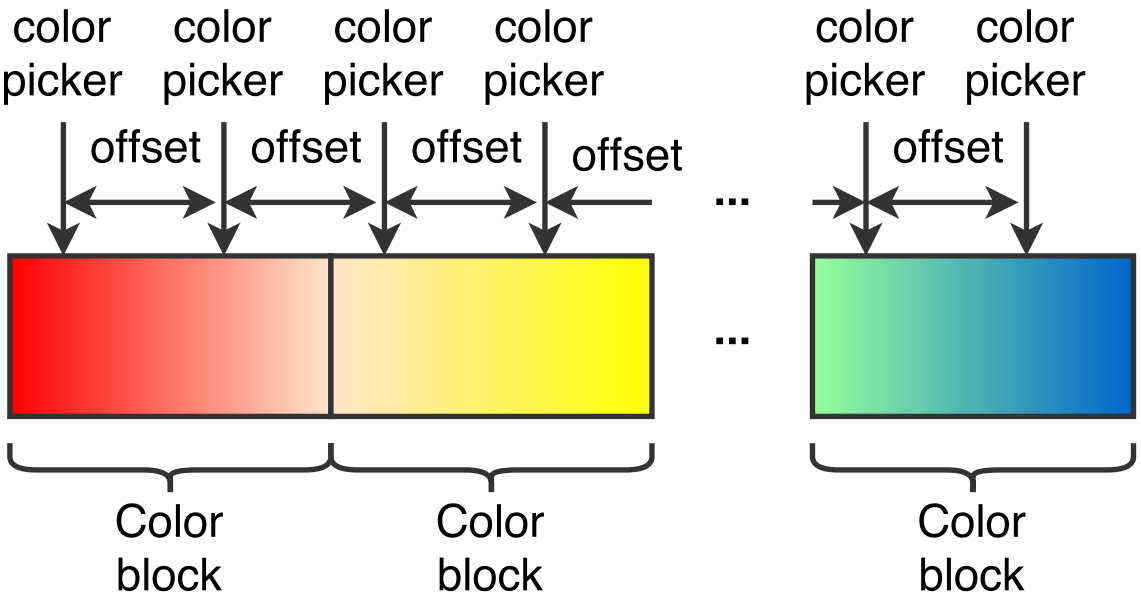

Figure 3.7: Concept of the color spectra for the chord visualization

cognition [SH15]. The following features of interactive chord diagrams prove their better application in different situations: highlighting, animations, tooltips, filter search and exclusion search.

3.4.2.1 Analyzing with Highlighting

Highlighting allows an interactive analysis of the visualization. To highlight a path in the diagram, the user can hover with the mouse over the path to get more information. With the help of this interaction, the relation between different objects can be easily found. It is also possible to highlight a complete group to find out fast which objects are related to each other. For example, the highlighting of one taxonomy reveals all its functions. The user can clearly recognize the relationships and evaluate the data. In our function-taxonomy use case, a user can for instance recognize the third largest function “Purine biosynthesis” and bring all related taxonomies into the foreground by a simple mouse over.

3.4.2.2 Local filtering for Better Understanding

The next feature of interactive chord diagrams is the local filtering, which ensures better clarity of the data. We control the visibility of the elements with the filter object. An initial filter (top 50 impact values), which was applied to the data at the startup of our application, creates a clear starting point from which the data can be explored. The filter functionality is accessible through an interactive clicking of the text in the diagram or through the filter list. The filter list shows every object of the diagram. In this way, elements can be added and removed freely by the user without changing the original data. Finally, the filter object can also be used as an exchange format (Figure 3.8). A biologist may select the functions “ATP synthesis” and “Glycolysis” to be displayed and easily determine the taxonomies related to them.

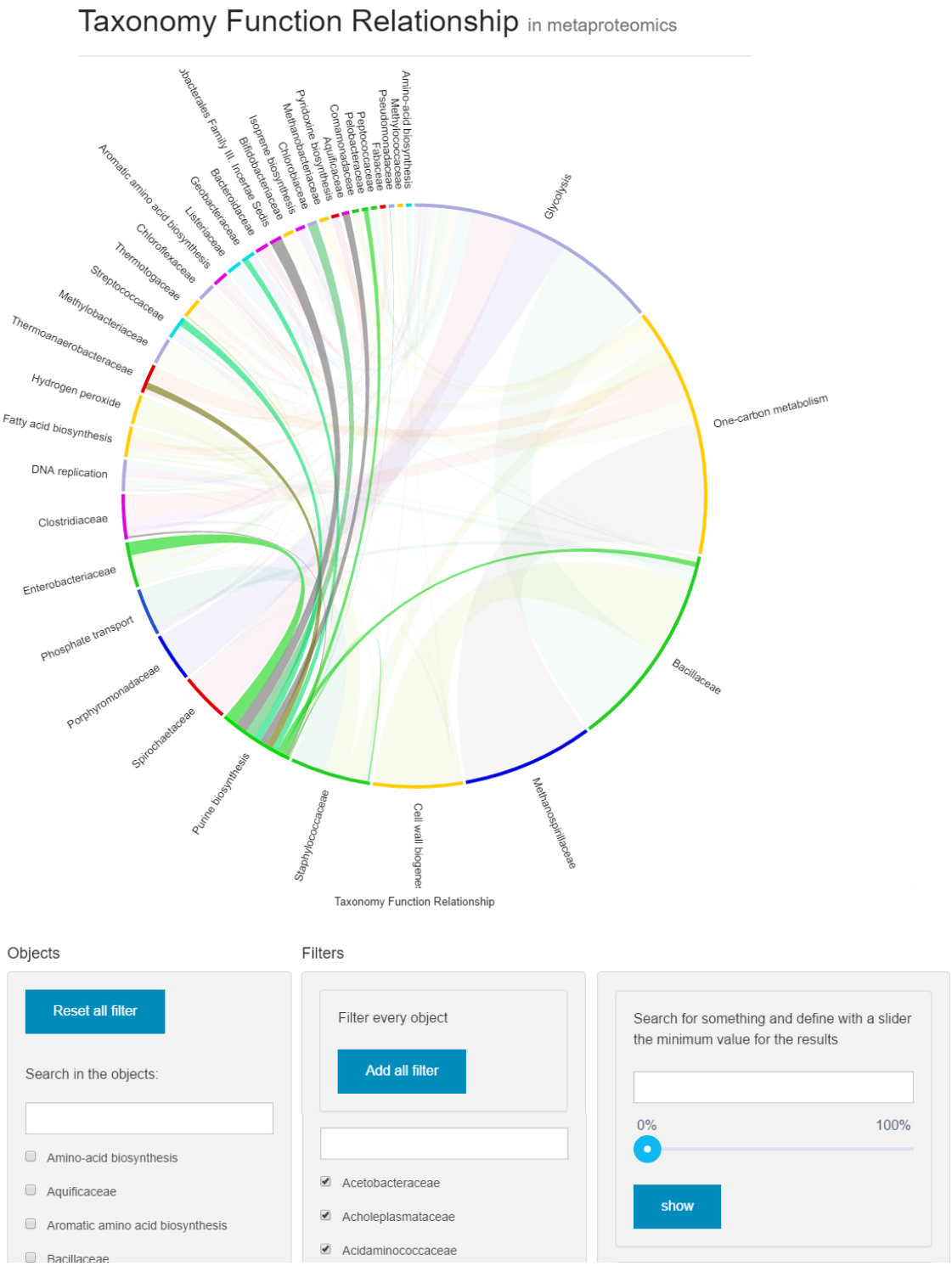

Figure 3.8: Demonstration of group highlighting: all relationship are visualized for a single object.

3.4.2.3 Search Functions for Specific Results

In addition to the interactive features, we implemented specific functions for improved data exploration. One function is a search that finds a specific object. The visualization will show all objects related to this specific one. Another function

finds an object excluding another object (exclusion search). For example, searching for object A excluding object B means that the visualization will show all objects related to A, but not related to B. Both of the search functions have an individual filter to cut off the lowest values with a threshold (Figure 3.8). The search functions are crucial to explore the function-taxonomy data. Taxonomies with smaller impact, such as *Methanococcaceae*, can be singled out and displayed alone.

3.4.3 Animations

The last feature of interactive chord diagrams are the animations. They are used to make changes in data easily understandable. It has been proven that animated graphics make the user feel involved in the content. Animations are especially useful to better visualize data in diagrams [HR07]. In our chord visualization, every change is animated. For example, when applying a filter, the object is excluded from the visualization and the other segments will be enlarged due to new rations between the remaining segments. This reorganization is animated by slowly enlarging all remaining segments.

3.4.4 Performance Evaluation

We begin with the evaluation of the start performance by measuring the time needed to load the visualization. We also analyze the dependency of the input data size and the time. In the next part, we will show the performance of the interaction in the visualization. We finish the performance evaluation chapter with the analysis of the results.

3.4.4.1 Performance for the Initial Loading of the Input Data

The input data is a table with five columns as described in the section Section 3.4.1.1, which contains the relationship between taxonomy and function. The initial loading time of the visualization with 10 sets of input data, which was 118 ms. Loading 3000 sets of data took 27,500 ms. To load the complete test input data required 75,700 ms. Figure 3.9 shows the result of the measurement for the initial loading. The diagram shows the average measurements and the correlation between data size and performance, which appears to be linear.

3.4.4.2 Performance for the Interaction Functions

In order to evaluate the performance of the interactions and the search functions we measured their individual calculation time. The best performance was measured for highlighting the path and the slowest one for searching for a specific element. A sixfold increase in the amount of data effected performance minimally as shown in Figure 3.10. Furthermore, as all calculation times are below 250 ms, latency time experienced by the user is negligible.

3.4.4.3 Results

The input data for the visualization is in a human readable CSV format. The transformation time of this format into the intersection matrix for the chord visualization correlates with the amount of data. This loading process is time consuming, but

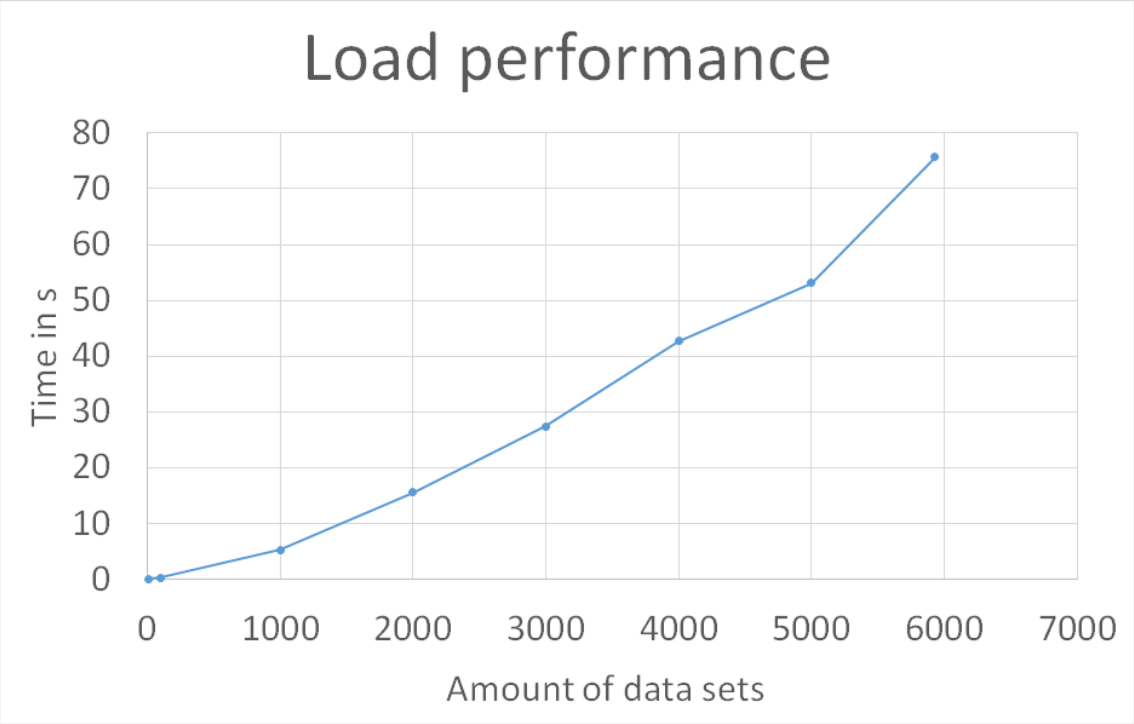

Figure 3.9: Performance of initial loading time dependent on the size of the input data set.

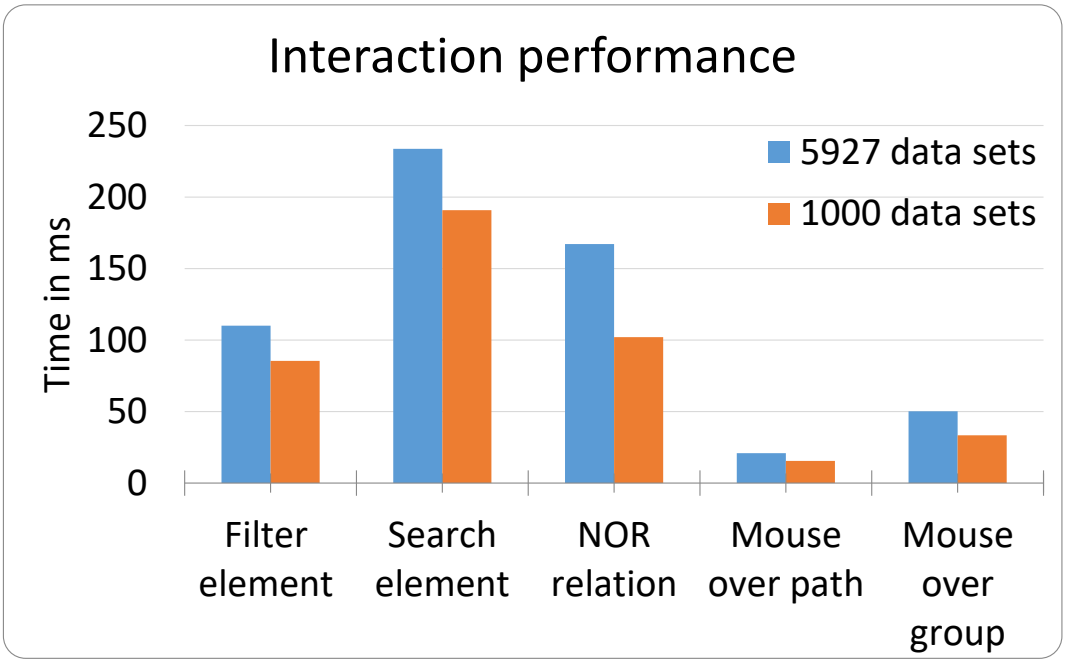

Figure 3.10: Evaluation of the calculation time required for interaction functions.

might only rise linearly with larger data sets. To achieve better loading speed the initial matrix calculation should be moved to the server side. The interactions and functions of the visualization are processed almost instantly. Once loaded, other

aspects of the client interface already perform in real time. An approach to decrease the loading time is a server-sided implementation of the transformation.

3.4.5 Empirical Evaluation

We conducted an empirical user study to test the effectiveness of the visualization prototype in comparison to the previously employed approach using a spreadsheet. Initially, we intended to provide a crossmap created by the MetaProteomeAnalyzer software for comparison, but preliminary tests showed, that the tasks cannot be solved using the crossmap. A typical big data set from a metaproteomics study, generated over the course of 150 days, was prepared resulting in an intersection matrix with 338 taxonomic families and their relations to 232 biological functions.

3.4.5.1 User Study

Participants from the local work group were given a six tasks, that reflect typical use cases for biologist. Test subjects were familiar with metaproteomics data and proficient in the use of spreadsheets. The first task was designed to familiarize the test subjects to the visualization and the underlying problem. Tasks two to five required the user to answer increasingly complex questions autonomously using either the spreadsheet or the chord-visualization. Solving all tasks required approximately 45 min per test subject.

3.4.5.2 Results

The time to answer individual tasks was measured and the boxplot shows the results, where green refers to spreadsheets and orange refers to chord visualization (Figure 3.11). As the boxplots show, task two and six were significantly easier to solve with the chord visualization, while task three and four were comparable in difficulty and task five was solved easier using the spreadsheet. Another observation is that the range and standard deviation for time to solve the tasks with the spreadsheet is significantly larger then for the chord visualization.

Test users rated the tool generally positive and preferred it over the use of a spreadsheet for all tasks except task four, which required the users to perform arithmetic operations. The time for solving the tasks with the spreadsheet varied more between testers, which can be explained by different skill levels of testers using spreadsheet software. In contrast, testers performed similar using the chord visualization, reflecting the fact that they used this method for the first time. The study showed, that the most important aspect of the chord visualization is the inclusion of several filter and search functions. Initial filtering allowed the users to start from a clear overview. Through the filter tools, users could add and remove any element to explore the data set. Through the implementation of specialized search functions such as the example of the exclusion search, entire workflows can be mapped onto a single interaction to increase the user experience. Some aspects of the user interface, such as the placement of certain user interface elements, were criticized by test users for not being intuitive.

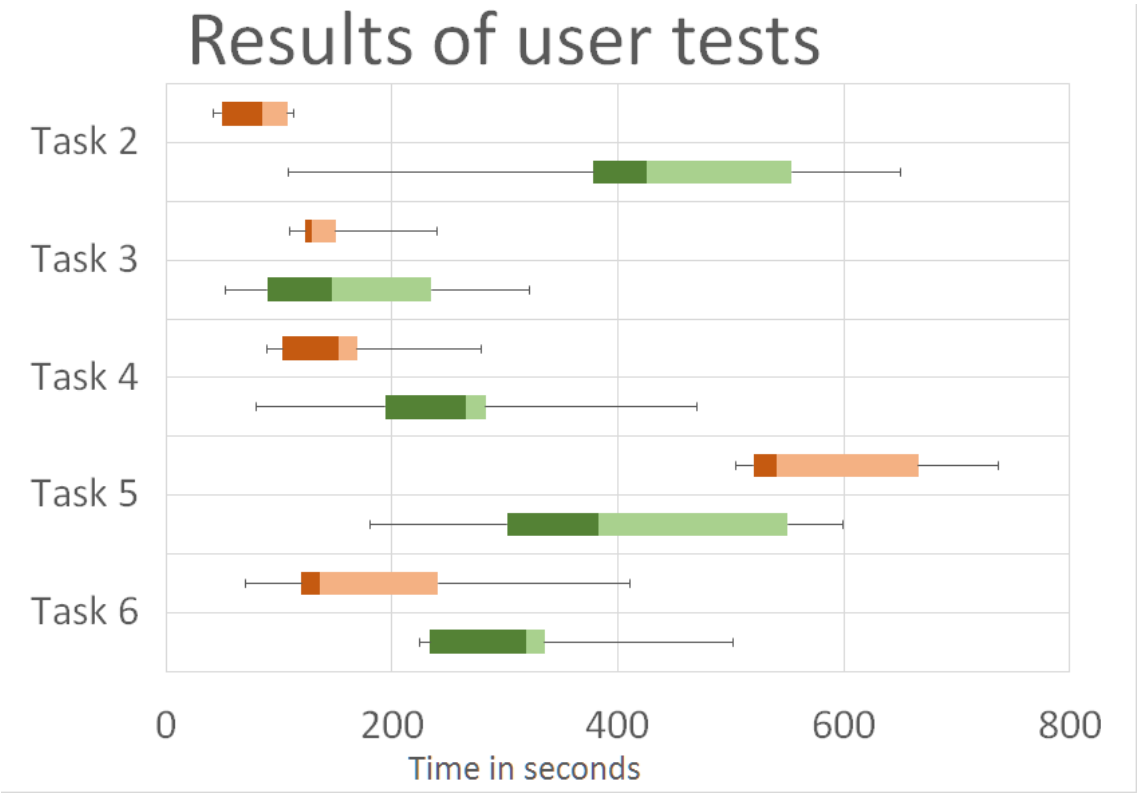

Figure 3.11: Boxplot showing the time required for individual tasks. Orange refers to tasks solved via chord visualization and green refers to tasks solved via spreadsheet.

3.5 Conclusion

The mass spectrometry data analysis can be separated into four steps. The first step is the biological sample preparation, the second one is the mass spectrometry, the third one is the conversion to an exchange format and the last one is the identification. This thesis provides research of the optimization of the mass spectrometry protein search workflow [HKRB15, Ast, Zou18]. Instead of optimizing each step on its own, we want to parallelize the last three steps, using an approach that combines fast data with protein search engines. We expect to gain performance from these improvements. However, the parallel workflow brings some challenges that we have to solve [Zou18, HSZ+17].

After the processing, the analysis and the interpretation of the data needs to be interactive using visualizations.

To analyze the results is in some cases very complex and an explorative analysis is needed. Therefore, we propose an interactive visualization such as the chord diagram [ZSB+17].

The prototype presented in this section was shown to improve the ability of users to solve metaproteomics use cases. Our chord visualization can be applied to other use cases through an easy exchange format. Regarding performance, the loading time still requires optimization. Search and filtering functions were received positively by users and constitute the starting point for the data exploration.

In the future, more attention should be focused on ready-to-use functions to increase the user experience. Additionally, support for numerical evaluation of the data should be implemented. Users should have the option to easily associate visualization elements with exact values. Tests with the prototype also revealed that improvements of the user interface are necessary. Finally, research can be expanded to other visualizations using larger test groups. This visualization need to applied to the platform visualize the results from the database in real-time. Hence, the interactive visualization is still a challenge.

In general, the goal of this thesis is to revise and improve the current workflow, revolutionizing the mass spectrometry procedures and developing a prototype of MStream, a real-time analytic platform for analyzing mass spectrometry data. We focus on the data processing and only store the result data for further explorative or autonomous analyses in the system. The interpretation of the data is part as well as the interactive visualizations are challenges of future work.

4. Streaming Mass Spectrometer

Since the goal of this thesis is to enable near-real-time mass spectrometry analysis, it is required that the main task of protein identification can be processed as a streaming task. This work will focus on the real-time processing of the data and not analyzing the results. Hence, we need a producer to stream the mass spectrometry data from the device during the measurement [ZDS+18, ZSB+19b, Zou18]. In this chapter, we analyze the current state-of-the-art protein identification software, showing the limitations of the local solution and propose a new architecture of the software using a fast data architecture. Finally, we show our software MSDataStream. This software is implemented in collaboration with Bruker Daltonik GmbH and connects a mass spectrometer with the cloud. Our tool MSDataStream allows streaming of the experimental data during the measurement.

In Summary, we make the following contributions in order to answer research question RQ2:

- Streaming Protein Identification: *Proposal of a fast data architecture for the protein identification task.*
- Stream producer: *Streaming mass spectrometry experiment data during the measurement to the cloud.*

4.1 Streamlined Spectrum Centric Protein Identification

In the metaproteomics workflow, the prepared biological sample is measured using a mass spectrometer. It measures the mass of the molecules of the biological input sample. The identification can be done with so-called protein identification engines. These are software tools, which compare the measured mass spectra with in silico calculated fragment mass spectra based on the amino acid sequence from a database and create a similarity score for each comparison. Possible protein search engines are

X!Tandem, OMSSA, Andromeda and Mascot as mentioned in Chapter 2 [HSZ+17, HBS+93, MBR+13, CNM+11, RR03, Ast, HKRB15].

The standard algorithms use a file of the experimental spectra data and a textual representation of protein databases to generate a result file of identified spectra data. Since the mass spectrometry devices are constantly upgraded, the size of the measured data has increased from Megabytes to Gigabytes, which leads to an increased number of comparisons [LBK+17, HSZ+17, MBR+13]. For example, the database of known proteins of a human in UniProt TrEMBL[1] contains around 140 thousand proteins, whereas it is approximated that the number of entries for all organisms exceeds 100 million – more than 700 times as much (around 40GB) [ABW+04]. Additionally protein databases with known proteins grow because sequence data expand. Hence, the performance of locally executed protein identification in a processing workflow reaches its limits. Furthermore, the possible future use cases of especially metaproteomics in the clinical environment for patient diagnostics need near-real-time processing [LQD14, NKH+17, PJW+14, PF17, ECL+12, BFWSS+15, XYF+17].

The current protein identification software X!Tandem needs several hours for one search and uses main memory to store all the experimental spectra data at once. Since the spectra can reach several gigabytes per experiment with modern mass spectrometer devices, protein searches, where each experimental spectra is scored against possible thousands of similar matches (thus increasing significantly the memory footprint), take a lot of time for calculation on a local system. Additionally, the resulting identifications are stored during the runtime in main memory, which increases the RAM usage a lot [HSZ+17, MBR+13, RR03].

Pratt et al. tackle the performance problem of X!Tandem and implement parallel X!Tandem using Hadoop and show, that big data technologies improve performance [PHTN12]. However, the parallel approach needs a preparation time for partitioning the data and uploading it to the system.

An alternative approach to process incoming data, which promises real time analysis of sensor data constitutes the fast data architecture [Wam15, KPB+17, KPB+19, Est16]. In this section we take a closer look at the X!Tandem protein identification tool and analyze the feasibility of the X!Tandem scoring algorithm on a Fast Data Architecture (see Section 2.6).

We found, that X!Tandem already exhibits streaming behavior for the proteins and streams the theoretical spectra for the scoring. For a central fast data architecture, the data processing step has to change to streaming experimental spectra data. Since the spectra data is user dependent and the protein database is static data, that is used by several users, we recommend an X!Tandem fast data architecture with streaming experimental spectra data and a persistent protein database. The new suitable data processing pipeline does not change the logic of the software, only the way of pairwise processing. Besides performance, this method brings some positive effects such as removing redundant peptides and enabling further analyses of measured spectra to the pipeline [ZDS+18, Zou18].

[1]The mission of UniProt is to provide the scientific community with a comprehensive, high-quality and freely accessible resource of protein sequences and functional information [ABW+04].

4.1.1 Improvable Components of X!Tandem

As described before, X!Tandem consists of a component to load and digest the proteins, a spectra component to load and loop through the spectra data and the scorer, which calculates the score and the expectation value of the proteins. The first step is to load all spectra. We ran X!Tandem 100 times with a protein database (6,158,917 entries) and spectra data (33,227 entries) on our system[2] to evaluate the current state. The spectrum loader component needs on average 219.6 milliseconds to load one spectrum into the memory. The proteins are loaded batch wise (default 1,000 proteins per batch) and for each experimental spectrum X!Tandem consumes the batch and calculates the score. The protein and the spectrum loader map the textual representation to a programmatic object and prefilter them to reduce the amount of comparisons. The protein loader needs on average 0.0026 ms to load one protein and each scoring takes on average 644.3 ms. The whole search took on average 5.6 hours. The current generation of mass spectrometers can produce more than 300 thousands spectra for one experiment and, as mentioned before, a big protein database has more than 100 million entries. Searches on this data would take over 24 hours. Additionally, the results, which are stored in an XML format, are not usable for analytical queries or further analysis [RR03, ZDS+18].

Overall, the current X!Tandem approach loads the spectra data into memory and "stream" batch wise the proteins and calculates the scores pairwise, using mapping and filter functions at runtime. This reveals following improvable components:

RAM Usage: The complete experimental data need to be loaded in memory, which is not necessary.

Protein Data Reiteration: The protein data is iterated for every experiment, which multiply the effort.

Uniform Result: The result data is in a specific XML format and it is not comparable to other search software result file formats.

Distributed Architecture: The whole process is single node application and is not horizontally scalable. Hence, it is limited by the machine performance it runs.

4.1.2 Streamification of X!Tandem

The integration of X!Tandem in the fast data streaming architecture would bring benefits for usability, performance, efficient storage and for further processing of the data.

The fast data architecture brings a central cloud solution with a central database management system (DBMS). We observed that the protein data is static and should be stored centrally and accessible for all users. This extends collaboration between biologists and allows reuse and non-redundant storage of protein data [Nat02, Whi05, RGSP07]. The ingesting streaming data will be the experimental spectra

[2]CPU 48x Intel Xeon E5-2650 v4; 512GB RAM

instead of the proteins. Streaming these data gives a possibility of further calculation with the experimental spectrum during the protein search such as DeNovo or clustering [KCS+17, SZBL10, GPRL+16]. A better option would be to digest the proteins, and store the distinct peptide information only once. Making digestion results persistent in this way would create a non-redundant peptide database and reduce the amount of comparisons [ZDS+18, Zou18, CNM+11]. The measured spectra are independent from each other and there is no need to load all of them into main memory. Processing of the data needs a stream processing engine for prefiltering, mapping and scoring functions.

The integration of X!Tandem into a fast data architecture would bring an application that streams spectra batch wise and loops them through all proteins in the database (Figure 4.1). The spectra can be streamed from a user using a file upload stream (Figure 4.1 – B) or a directly connected mass spectrometer (Figure 4.1 – A). The spectra are collected in a message queue (Figure 4.1 – C). The stream processing engine consumes the messages, maps the spectra into programmatic objects and prefilters them (Figure 4.1 – D). In the next step, the consumed batch loops through all the peptide sequences from the database (Figure 4.1 – E) and the calculated score and the expectation value are stored in the database (Figure 4.1 – 5). All the components are managed by a cloud operating system (Figure 4.1 – F) [ZDS+18, Zou18, RR03, MBR+13].

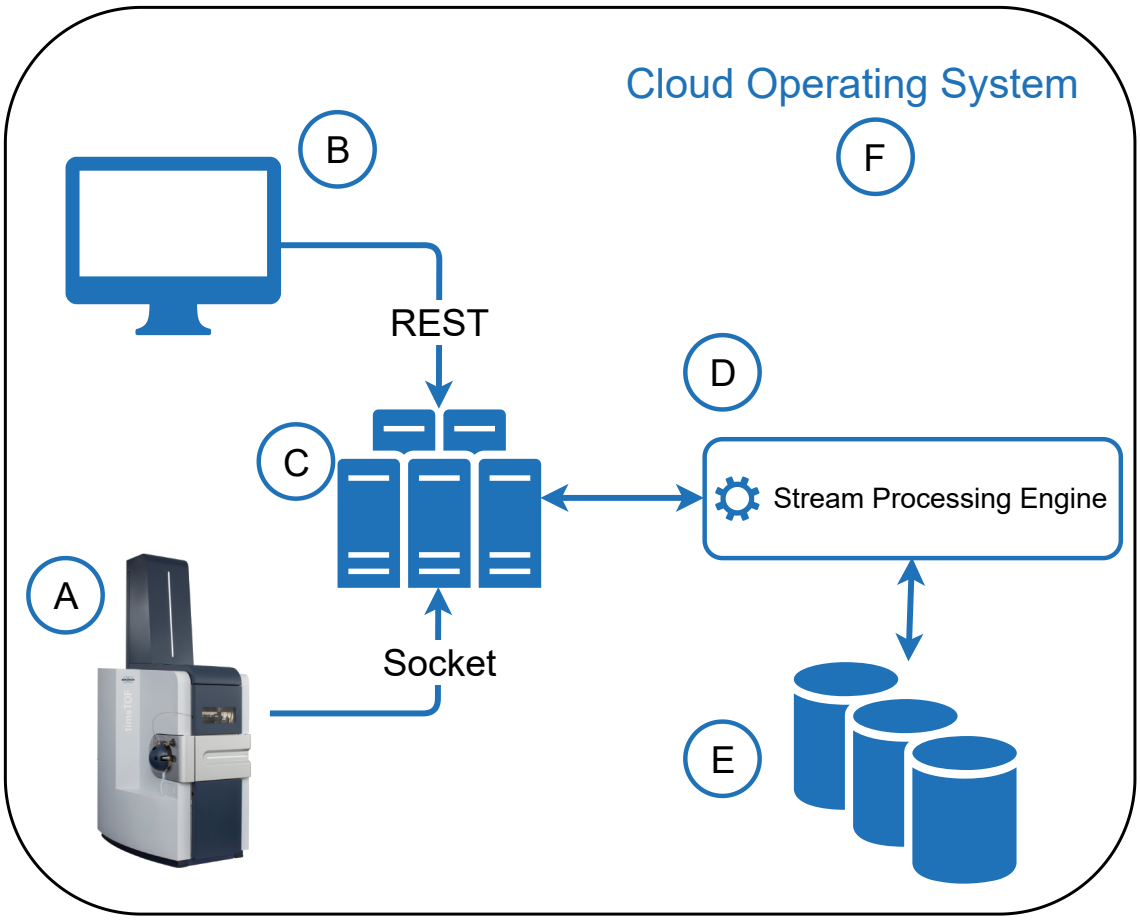

Figure 4.1: The general fast data architecture for X!Tandem algorithm.

However, on a theoretical basis, our architecture shown in Figure 4.1 should remove X!Tandem's deficiencies due to the following design choices. Overall, the integration brings following improvements to the components:

RAM Usage: Only a small time window of the experimental data needs to be loaded in RAM.

Protein Data Reiteration: Once the protein data is transformed in the database, only needed peptides are selected.

Uniform Result: The result data is in a general data structure in a database and it is comparable to other search software integrated in the system. Furthermore, it enables further analysis and comparisons.

Distributed Architecture: The whole process is horizontally scalable and is not limited by the machine it runs.

4.2 Mass Spectrometer as Stream Producer

Currently, the smallest parallelizable unit for the mass spectrometry data analysis pipeline is a whole experiment file. Since the mass spectrometer measurement and digitalization duration (approx. 2 hours) cannot be avoided, we shrink the smallest unit to a single spectrum instead of a whole experiment. This means, we connect a mass spectrometer to the cloud for outsourcing the calculation and overlap mass spectrometer processing with data processing by using a streaming-based architecture, i.e., a fast data architecture [Wam16, Wam15].

In this section, we present one important cornerstone of our architecture, our tool MSDataStream. The tool is responsible for grabbing the single spectrum data from the mass spectrometer as it arrives, converting it into a readable format and streaming the data to the cloud for processing [ZSB+19b, NSP17].

The software MSDataStream combines as a local tool the possibility to upload batch-wise an already completed experiment file or stream the data during the measurement. Hence, this component tackles the tasks in Figure 4.1 – A and Figure 4.1 – B.

4.2.1 Producer Architecture

The system architecture is shown in Figure 4.2. In the following, we describe the architecture in detail based on the background information described in Chapter 2.

For our development, we collaborate with Bruker Daltonik GmbH, a mass spectrometer company from Bremen, Germany. Each of their devices are connected via a digitizer to a computer. The digital signal is collected in a proprietary RAW file format [LBK+17, Ast]. Additionally, the measurement software provides structure and meta data to index spectrum data that belongs together. Since each manufacturer uses their own RAW file format, Bruker provided us with a library (DLL file) to access the digital spectrum data. The index data is stored in an SQlite database and provides the location of spectrum data in the RAW file. The index structure

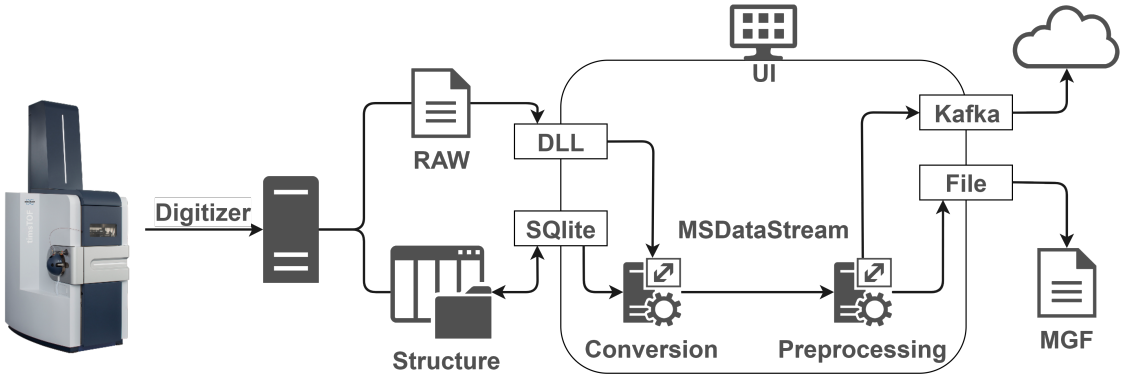

Figure 4.2: The flow of the mass spectrometer data through the MSDataStream software

consists of 16 tables that describe meta-information and the spectrum location for each single spectrum. MSDataStream reads the meta-information for a single spectrum from the SQlite index and extracts the spectrum data from the RAW file via the DLL [Mat16, Gar13, ZSB+19b].

The data of the mass spectrometer is grouped by MS1 data. In order to send complete MS2 datasets as a batch, it is necessary to collect all MS2 mass spectra that belong to a specific MS1 mass spectrum. During the measurement, it is possible to collect MS2 data. In our software, we implemented a waiting queue until the MS1 signal changes. The change notifies the next dataset and all collected MS2 data can be forwarded [Mat16, Deu12, MTS+04, Ast].

The collection of the batch happens before the conversion step while reading out the data from RAW files and the SQlite database. After the batch is completed, the collection of the MS2 datasets, each mass spectrum is converted into MGF format. MGF is chosen since it fits to the current pipeline. It can be further extended by other conversion formats in future work [Mat16, Deu12, MTS+04, Ast].

Then, several preprocessing methods increase the quality of the spectrum data. The preprocessing step would have increased performance if the signal data is directly used before converting the data into a readable format, but for future use with different devices, it is decided to perform the preprocessing after the conversion. Otherwise, the adapters for different devices need more functionality than the conversion. Hence, in the application with multiple devices, the processing of the data should be completely outsourced to the cloud [ZCD+12, Wam16, KPB+17].

Finally, MSDataStream sends the data to the cloud via Kafka broker for further processing or to write the data into a file. Summarized, MSDataStream checks periodically (e.g. every 2 seconds) for new measured data, collects it and produces messages [Gar13].

Our MSDataStream software requires several parameters for the execution. We implemented a JavaFX interface, which queries MSDataStream tasks and helps the user to configure the parameters. After sending the first data messages, the user waits until the results are generated from the fast data system and are shown in a live updated visualization [ZSB+17].

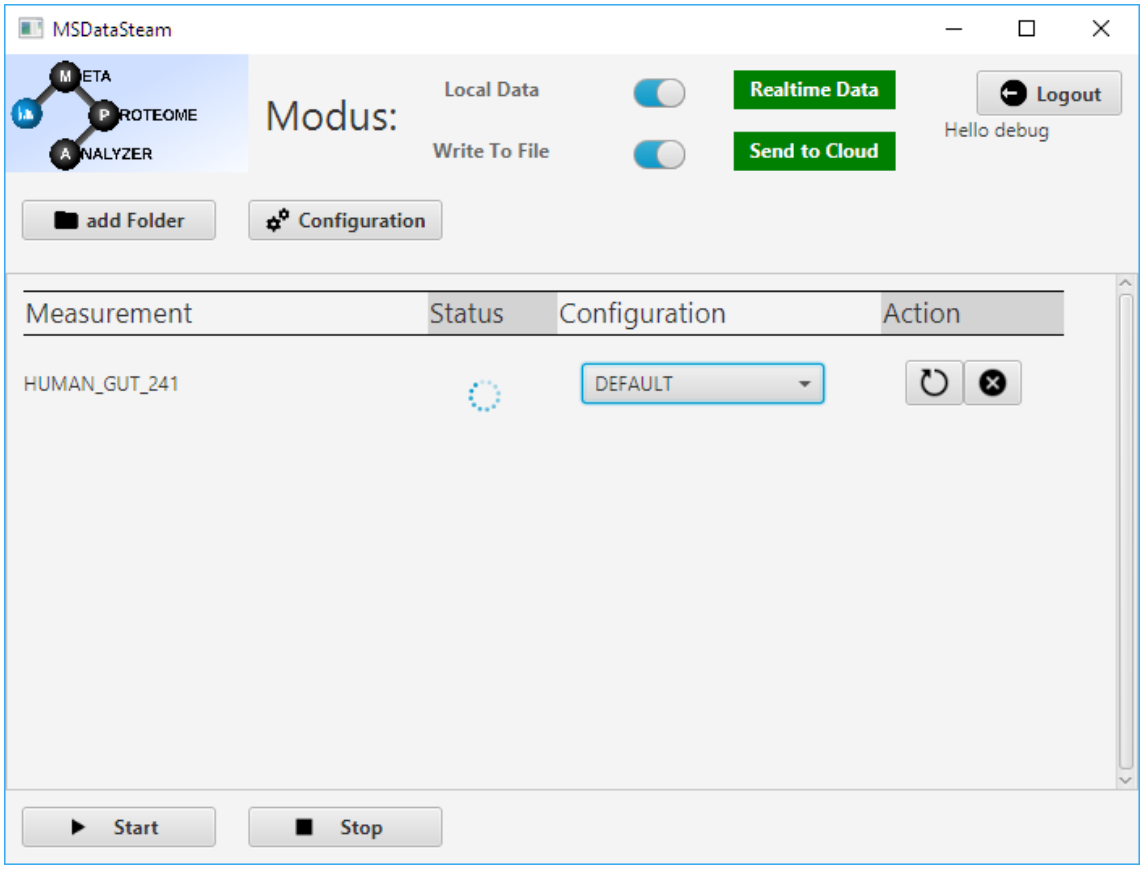

Figure 4.3: User interface of MSDataStream. The control elements are on the top and the experiment queue management is in center of the user interface.

In Figure 4.3, we show the user interface (UI) implemented in JavaFX. The UI supports user management and queue management of experiment streams and uploads. Two toggle buttons control the properties "stream" and "direction". The "stream" attribute decides whether to stream the data during an experiment or upload an already completed experiment. The "direction" attribute decides to save the data into a file or send the experiment data to the internet. Hence, our tool can be used as a standalone client software producing MGF files from mass spectrometer measurements.

Since the conversion happens locally to a MGF format, it is possible to add complete experiments in that file format. Finally, a complete measured experiment in Bruker RAW format is compatible, too.

4.2.2 Integration of MSDataStream

Integrating the MSDataStream software, the producer for the experimental data changes the general architecture regarding the clients. The connection happens over the Apache Kafka broker installed on the cloud platform. In Figure 4.4, we show the upgraded architecture. MSDataStream Software is a part of the streamlined analytic pipeline.

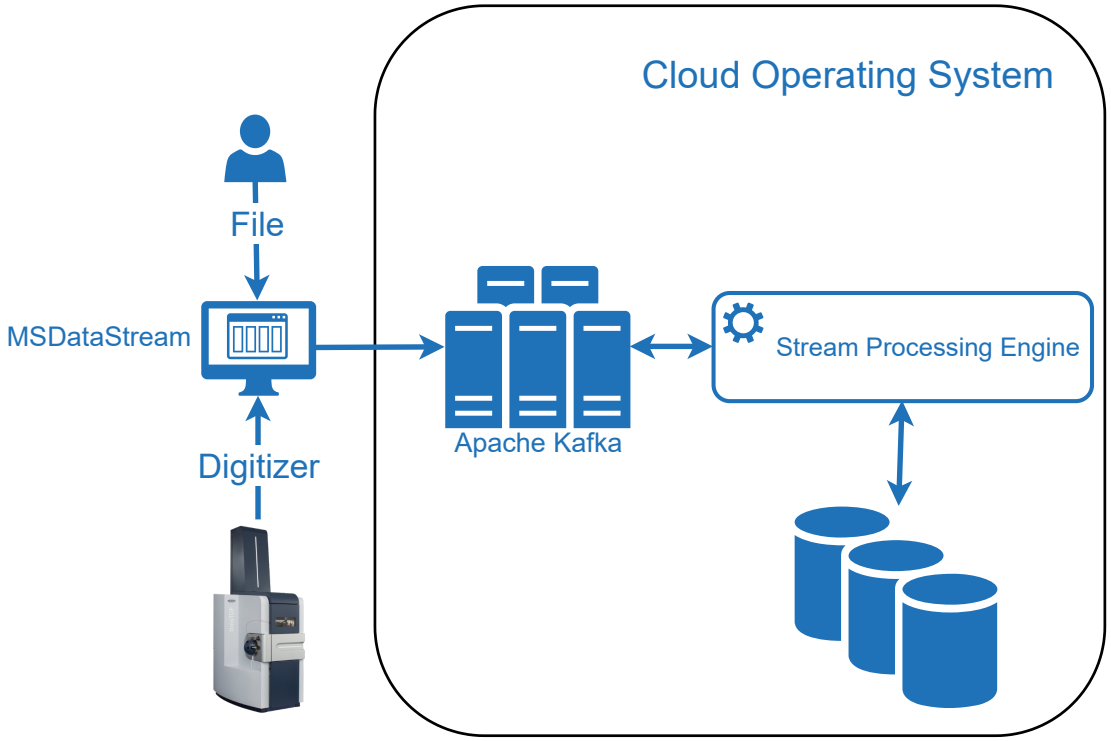

Figure 4.4: The general architecture including MSDataStream.

4.3 Related Work

In this section, we explore related work regarding improvement of the protein identification software X!Tandem. We begin with a distributed solution following by the implementation using modern hardware.

Pratt et al. implemented parallel X!Tandem using Hadoop MapReduce on Amazon Web Services, which is the first parallel implementation of X!Tandem to exploit the scalability and fault tolerance of Hadoop to create large on-demand compute clusters on commodity hardware [PHTN12].

Another work implemented the algorithm using GPUs to increase the performance and parallelize the comparisons [BSL+11]. In our work, we propose to implement the X!Tandem protein search on a mini service in the fast Data streaming architecture.

Regarding the streaming data, Zhanlin et al. presented in his work a cloud based IoT platform for car parking [JGO+14]. In the work, the authors describe how to stream the data via Apache Kafka and analyze them. The streamed data in the work is not complex, because only orientation, positions and information of the vehicle are the data from the clients.

Finally, we could not find related work for the stream producer during the measurement. All converters work file based and need all the data at once. Hence, even the conversion software from the company Bruker Daltonik GmbH is file based [LBK+17].

The related work of the overall MStream platform is described in Chapter 8.

4.4 Conclusion

We show in this section pros and cons of the current file-based version of X!Tandem. In addition, we show that the current workflow of the algorithm already has a streaming behavior and we make the case that performance benefits could be achieved by using modern fast data technologies [RR03, MBR+13]. We point out that an integration on a fast data architecture increases not only performance but also usability and enables further analyses besides protein identification. The state of the art of fast data shows clearly the benefits of streaming near-real-time applications [ZSB+19b].

Additionally, we show the streaming producer MSDataStream. The software is installed on the machine, which is connected to the mass spectrometer. Using manufacturer specific scripts, the data is read out during the measurement. Each dataset is converted directly into the MGF text format that we use as general interface for preprocessing and further streaming into the cloud [ZSB+19b].

5. Managing the Protein Sequence Data

The state-of-the-art protein identification approach uses, from the algorithmic perspective, a peptide-centric approach comparing the experimental data (mass spectra) with a protein sequence database. For this purpose, the proteins are divided into peptides, which results in billions of data sets [HKRB15, AM03, LBK+17, Ast]. Each peptide is compared to every spectrum to find the highest similarity. All those comparisons take several hours to complete and as long as the measurement data is not written, the protein identification cannot start. This leads to a further delay of the protein identification. A method which identifies experimental data individually allows to analyze each single spectrum during the measurement as this one needs to be compared to all possible candidates [HSZ+17, HBS+93, MBR+13, CNM+11, RR03, Tab15]. Hence, an index schema is needed to query only suitable candidates of the sequence data and reduce the search area to a minimum [ZDS+18, ZSB+19b, LM10, ZSB+19].

In Figure 5.1, we marked the component that is concerned in this chapter. In this chapter, we present an index schema for the sequence data of a protein sequence database, using a column-based index in a distributed database management system (DBMS) that allows streamlining of the analysis step. This leads to the problem of how to transform the protein sequence data from the current state-of-the-art format to the indexed schema in the DBMS. The transformation involves separation of the sequence data, deduplication of the sequence data and many mass calculation steps, which have to be calculated before inserting the sequence data into the DBMS. To be applicable in a real-world scenario, the data preparation has to be efficient regarding their memory consumption and runtime [ZSB+19, Nat02, Whi05, RGSP07].

After presenting the transformation process, we describe four methods to aggregate the data into the index structure. The first one is the naive in-memory approach, the second one is the structured hard disk approach, the third method uses DBMS queries and the last one is the radix-trie-based method. At the end, we evaluate

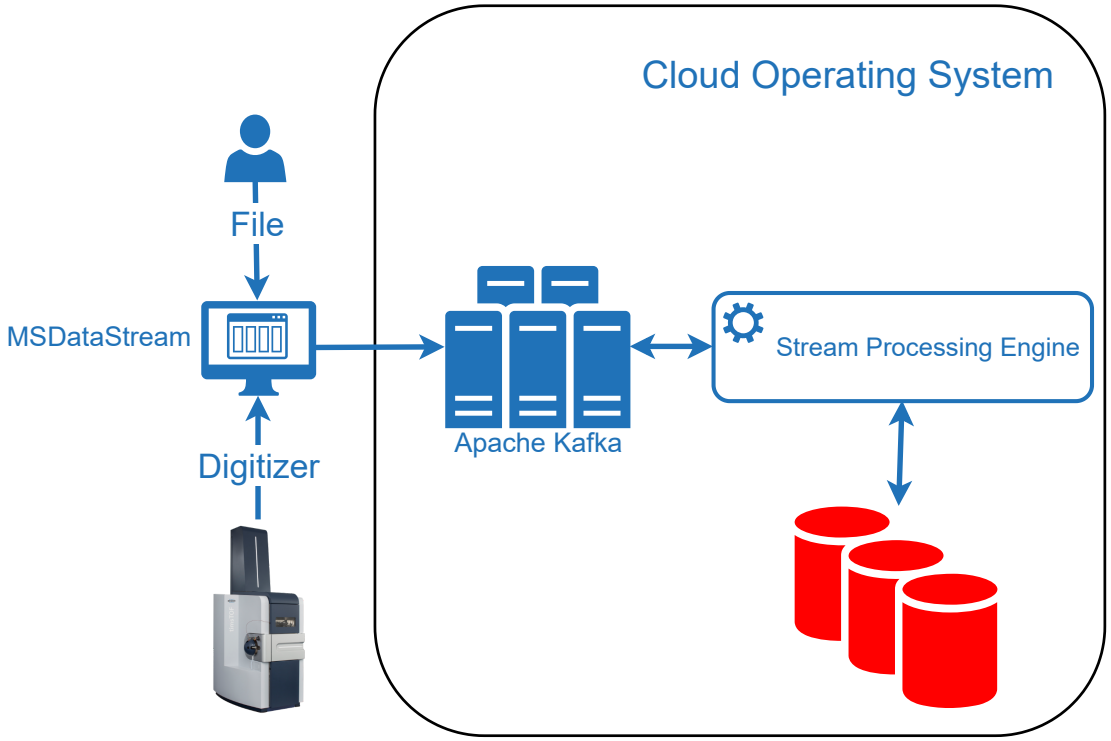

Figure 5.1: Marked database management system component in the architecture of the mass spectrometry analytic platform.

those methods and show that a trie structure is very efficient for the storage of sequence data and has the best overall performance among all approaches.

In Summary, we make the following contributions in order to answer research question RQ3:

- Index schema: *Proposing a suitable index schema for protein sequence data*
- Protein data transformation: *Evaluate the best method to transform the protein sequence data*

5.1 Data Preparation for Real-Time Protein Identification

To perform near-real-time processing, an index structure is needed to reduce the search space of suitable candidates for each spectrum. Using an index on the data, the software can query the needed data instead of performing a linear search. Hence, the search space is minimized. In this section, we explain first the schema followed by the transformation methods into the indexed schema [ZSB+19, Nat02].

5.1.1 Indexed Masses of Peptides

To enable fast access to suitable candidates without losing all the information from the protein sequence database, we introduce our data structure. Our schema for the

prot_id	FILE_1	FILE_2	prot_seq
PROTEIN_1	null	['{json-data}']	TEMRTEQAFY
PROTEIN_2	['{json-data}']	['{json-data}']	TEMRTEMQG

Listing 5.1: Sample data of a protein table.

protein data consists of three tables – the protein table, the peptide table and the pepmass table and the relations between them (Figure 5.2). This table schema is needed in the database and used in the analysis platform [Nat02, RGSP07, ZSB+19].

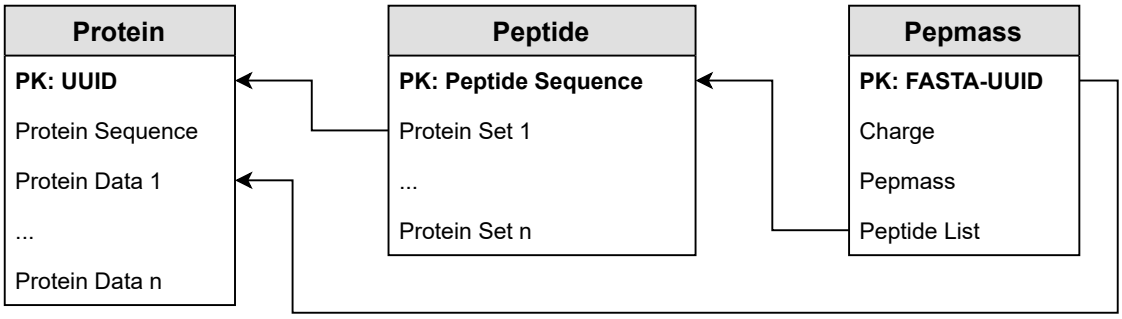

Figure 5.2: General schema of preprocessed protein data in the mass spectrometry analytic cloud system.

5.1.1.1 Protein Table

The `protein` table uses the protein sequence as a unique id and contains a list of description texts, such as species or functions of the proteins, in each row, since one sequence can define more than one protein description or can be mapped to more than one protein description (see, Chapter 2). The UUID is created like a hash from the protein sequence. Hence, the unique protein sequence is reduced and can be used as identifier for the protein data. The primary key is the "UUID" column in the `protein` table. Each protein sequence database appends a new column in the `protein` table, which results in a wide table. This fact leads to the necessity to have an individual column for each FASTA file, since the protein identification step is only processed against a single database. The `protein` table consists of minimum two columns – "UUID" and "Protein Sequence". For each uploaded protein sequence database, a new column "Protein Data" is added to the table, which contains the description from the FASTA file. Hence, uploading new data brings new descriptions but not necessarily new protein sequences. Therefore, we can save storage in the DBMS since the sequence data is much bigger than the description data (Figure 5.2). In Listing 5.1, we show a sample data of a protein table. Two files are loaded into the table, which creates two columns "FILE_1" and "FILE_2". For each protein in the files, a row is added with generated UUID in column "prot_id" and the sequence in column "prot_seq".

pep_seq	FILE_1	FILE_2
TEMR	[PROTEIN_2]	[PROTEIN_1, PROTEIN_2]
TEQAFY	null	[PROTEIN_1]
TEMQG	[PROTEIN_2]	[PROTEIN_2]

Listing 5.2: Sample data of a peptide table.

5.1.1.2 Peptide Table

The `peptide` table consists of the peptide sequence as a primary key and has a set of protein UUIDs, which relates to the `protein` table. Similar to the `protein` table, each protein sequence database is stored in an additional column. Hence, the `peptide` table consists of minimum one column – "Peptide Sequence". Each uploaded protein sequence database appends a new column – "Protein Set", to the `peptide` table, which contains a non-redundant collection of protein UUIDs. The UUIDs describe where the peptide comes from. This relational information is needed for further analysis of the data. Using the relation information, the tool can reconstruct the proteins from the identified peptides (Figure 5.2). In Listing 5.2, we show the sample data of the `peptide` table regarding the sample data from `protein` table (see Listing 5.1).

5.1.1.3 Pepmass Table

The `pepmass` table is the table, which groups the peptides with the proposed parameters "FASTA-UUID", "Charge" and "Pepmass". All data of one FASTA should be on one partition. This is the reason why we use this value as a partition key in the `pepmass` table. For the peptides, we store all possible charges (we consider charges one, two and three, higher charges are untypical for the measurement results.) and calculate the mass of the peptides. Because of the possible modifications of the peptide, the total mass can be different. Modifications are possible changes of ions in the peptides, which change the mass of the ion and therefore the total mass of the peptide. For example, oxidation during the preparation process could modify the experimental data and should be considered during the identification process. Hence, each modification is similar to an additional letter in the sequence and increases the amount of peptide sequences drastically. In our work, we calculate the masses for two typical modifications. The precalulated charges and all the precalulated modified and unmodified masses generate a huge amount of peptides in the `pepmass` table. For example, Swissprot contains 23,934,321 peptides, which produce 14,579,004 non-redundant peptides after deduplication of the peptide sequences. The grouping and calculation of all possible masses results in 111,183,434 peptides in 4,814,243 rows of the `pepmass` table. Hence, the grouping transforms 500MB protein sequence database into 2GB of precalculated data in our schema. The peptides are stored as a list of strings in the column "Peptide List".

In Listing 5.3, we show the sample data of the `pepmass` table regarding the sample data from `protein` table and `peptide` table (see Listing 5.1 and Listing 5.2).

```
 fasta   | charge | pepmass | peptide_list
---------+--------+---------+--------------
 FILE_1  |      1 |  536    | ['TEMR']
 FILE_1  |      1 |  565    | ['TEMQG']
 FILE_2  |      1 |  536    | ['TEMR']
 FILE_2  |      1 |  768    | ['TEQAFY']
 FILE_2  |      1 |  565    | ['TEMQG']
```

Listing 5.3: Sample data of a peptide table.

In Table 5.1, we show a set of real world example data from the `pepmass` table. One query over all possible candidates takes in average 19 millisecond.

FASTA	charge	pepmass	Peptide List
UUID_1	1	489.2415	['GGGGGGK']
UUID_1	1	503.2572	['AGGGGGK', 'GGGGGAK']
UUID_1	1	504.2412	['SGAGAAA', 'TGAAAGG', 'SAAGGAA']
UUID_1	1	514.2619	['AGAAPAG']
UUID_1	1	517.2728	['GAGGGAK', 'AAGGGGK']

Table 5.1: Example data from the pepmass table.

5.1.2 Data Transformation

Our proposed schema increases the query performance to get the suitable candidates for each spectrum in a few milliseconds, which leads to increased storage because of the precalculations of all masses. As mentioned, we have to consider more than one protein sequence database. Hence, an efficient technique is needed, which allows to upload new databases and to transform their data into our schema [ZSB+19]. In Figure 5.3, we show the four steps to transform the data from the FASTA format into our schema.

The first step is to deduplicate the protein sequences and merge the descriptions of the entries with a similar protein sequence (Step 1 in Figure 5.3). The next step is the protein digestion, which splits the protein sequence into smaller peptides. Equal peptide sequences can be extracted from different proteins, which leads to a many-to-many relationship between proteins and peptides. Due to the high number of those relationships, the protein digestion is conducted with a list of protein id's in the table (Step 2 in Figure 5.3). The next step is the deduplication of the peptides during which only the unique peptides are left within the relation to the proteins that they come from (Step 2 in Figure 5.3). For each of those peptides, the mass needs to be calculated with all the possible modifications and charges. Afterwards all the data is stored into the DBMS [ZSB+19].

For the data transformation, we evaluate a map structure in memory and on disk and the radix tree structure. As next, we explain the steps in detail, because it should be clear to follow the evaluation.

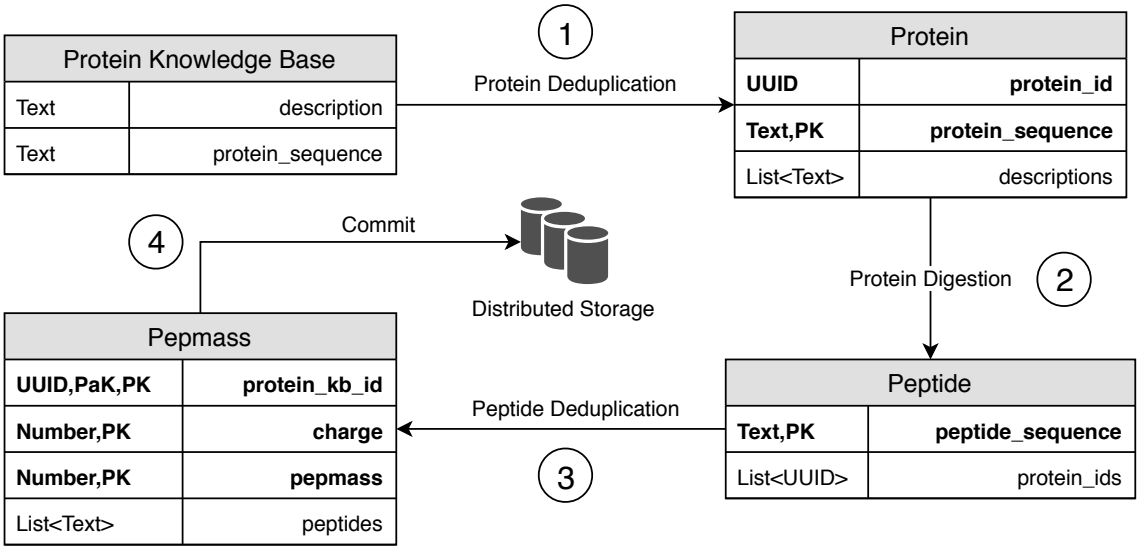

Figure 5.3: Transformation steps from FASTA format to our indexed schema.

```
1 UPDATE protein SET prot_seq='proteinSequence', "fastaID"= "fastaID
    " + ['proteinData'] WHERE prot_id = prot_uuid;
```

Listing 5.4: Query for protein deduplication in the DBMS.

5.1.2.1 Protein Deduplication

The goal of this step is to map all descriptions to a unique protein sequence. The naive approach is to create a map with the protein sequence as a key and a collection of descriptions as a value. In this case, a map would only consider the currently loaded protein sequence database. However, since we want to persist the results in the database, this deduplication step needs to be executed on the DBMS side. Hence, we have to use update queries to append data to an existing protein sequence or insert the protein sequence if the data set does not exist. A sample query is shown in Listing 5.4. The name of the new column is "fastaID" and the query inserts a new entry or updates the list in the new column by a new description of the protein. In Cassandra query language, an UPDATE statement can INSERT datasets if the data not exists.

5.1.2.2 Protein Digestion

After the deduplication of the protein sequences, the peptides need to be extracted, using the digestion method. Hence, the sequences are divided on specific letters of the sequence. This method simulates the digestion from a laboratory on the digital sequence. Additionally, we have to consider missed cleavages (MC) that define the possibility of missed cutouts in the sequence. As an example, let us consider a protein sequence with a length of 300 would be split into 23 peptide when defining MC to zero. Defining MC to the value of one would generate 56 peptides. The missed cleavage parameter increases the amount of peptides linearly [MFT+13]. In our work, we define this parameter with the value of two, which is enough for most of the mass spectrometer experiments [RGSP07, TLM+00, Nat02, Whi05].

5.1.2.3 Peptide Deduplication and Mass Calculation

The peptide deduplication is conducted only for the currently used protein sequence database file and does not depend on the data from the other columns. During this step, each peptide has to be deduplicated without losing the information about the protein that the peptide comes from. The protein sequences are already mapped to an identifier in our DBMS. Hence, there are four possible approaches to deduplicate peptides: The naive approach is to create a map with the peptide sequence as a key and a collection of the protein identifiers as a value. The map approach can be divided into (1) an in-memory map approach and (2) a file-based map approach. (3) The next approach is to update the list of protein identifiers in the DBMS and (4) the last approach is to use a radix tree for the sequences with relations to the protein identifiers on each end node.

Afterwards, the peptide masses need to be calculated. For each unique peptide sequence, the mass for all combinations of possible modifications and for all charges has to be calculated and stored in the DBMS. Therefore, the sequences of the peptides have to be mapped to the precalculated masses. Since each peptide generates many different masses, this peptide is stored multiple times in the pepmass table. In the last step, all the grouped and precalulated data needs to be inserted into the DBMS.

Due to the fast data architecture and the need of fast accessing of the protein data, it is required to implement an efficient service for uploading protein sequence databases into our system. In order to achieve this, the transformation has been implemented using different approaches. Firstly, we implemented the naive approach using in-memory hash maps. Secondly, we implemented the steps, using a structured storage on the hard disk. In the third approach, we implemented the steps using DBMS queries and in the last one, we implemented an extended radix tree as an in-memory data structure.

In summary, the four steps of the transformation process can be implemented in different ways, see Table 5.2. The first and the last step are done with the database engine method. An evaluation is needed, to find the best method to transform the data.

Method	Protein Deduplication	Protein Digestion	Peptide Deduplication	Commit
In-Memory Map	No	Yes	Yes	No
HDD Map	No	Yes	Yes	No
Database Engine	Yes	Yes	Yes	Yes
Radix Tree	No	Yes	Yes	No

Table 5.2: An overview of protein data transformation approaches.

5.2 Implementation

We implemented the transformation of a protein sequence database as a cloud service, using Java Jetty application server and Cassandra DBMS on an Apache Mesos

```
1 UPDATE peptide SET "fastaID" = "fastaID" + ["protID"] where
     pep_sequence='pep_seq';
2 UPDATE pepmass SET "peptide_list" = "peptide_list" + {'
     peptideSequence'} where "fasta"=fastaID AND "charge"=charge AND
      "pepmass"=totalMass;
```

Listing 5.5: Queries to transform protein data from FASTA format into the table schema using DBMS.

system. Additionally, SQlite is used as a storage for the local structured data. The in-memory approach, the file system approach (SQlite) and the DBMS query approach use standard methods (CQL[1] queries) and data structures (hash maps and sets) for the deduplication and other transformation steps. In this section, we give a brief description of the implementation of the naive approaches and a detailed explanation of the extended radix tree approach [ZSB+19].

5.2.1 Transformation using a Map Structure

This approach uses a map and a key value structure for the peptide deduplication and the protein relationship. The first approach is the implementation of the map in-memory and the second implementation uses an SQLite table as key value storage. We implemented the approach using the peptide sequence as a key and a set of protein identifiers as value [ZSB+19].

5.2.2 Transformation using DBMS Queries

Since the result data needs to be inserted into the database management system (DBMS), we implemented queries to transform the data directly in the DBMS. We use UPDATE queries, which add a new row if the key does not exist or otherwise, append a value to the list in the column. The deduplication of proteins is already implemented, using UPDATE queries on the `protein` table. In this approach, we implemented additional UPDATE queries for the `peptide` table and for the `pepmass` table. In Listing 5.5, we present the UPDATE queries used in this method for the `pepmass` table and for the `peptide` table [ZSB+19].

5.2.3 Transformation using Extended Radix Tree Structure

In contrast, the ideas behind using a radix tree are more complex and, hence, we give a detailed explanation on the relational radix tree approach in the following [SOAA97, LKN13, DLB59, ZSB+19].

In Figure 5.4, we show the approach on an example and in Listing 5.6, we show the algorithm of the transformation. At the beginning, a new column is added to the `protein` table and to the `peptide` table (Listing 5.6 line 2). For each protein, an identifier is generated and the protein descriptions in the protein table get updated (Listing 5.6 line 6,7). Next, the proteins are divided and we generate a

[1] Cassandra Query Language

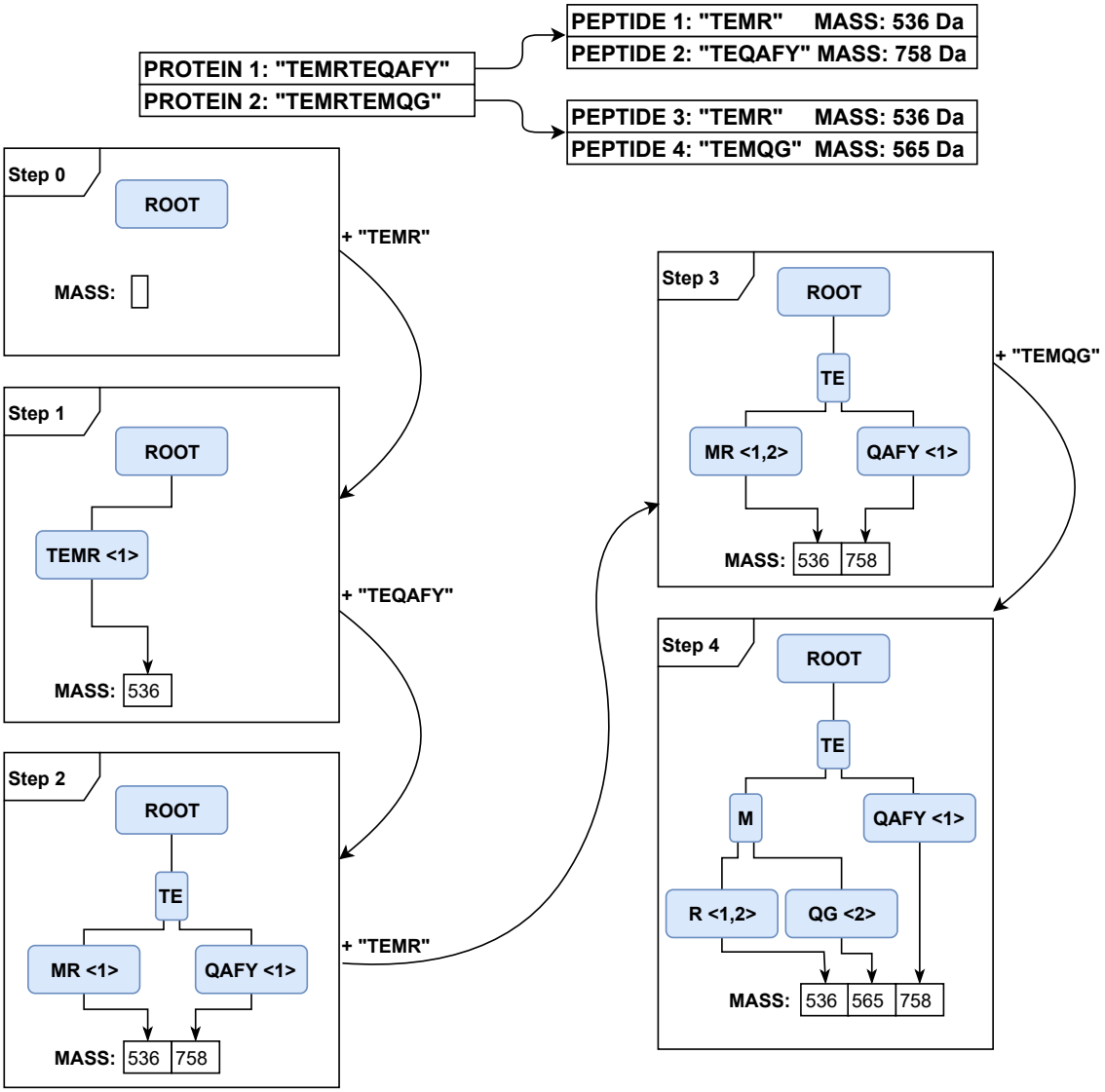

Figure 5.4: Sample of a radix tree as peptide storage in the data transformation process.

list of peptides for each protein (Listing 5.6 line 8). Hence, the two sample proteins "TEMRTEQAFY" and "TEMRTEMQG" are divided into peptides. The first protein generates the peptide list: "TEMR", "TEQAFY", and the second protein generates the peptide list: "TEMR", "TEMQG". For the ease of explanation, we choose this fictional separation, which does not necessarily reflect real world digestion but consider the trypsin separation rules. Each of the peptides is inserted into the trie with the additional parameter of the protein identifier (Listing 5.6 line 10).

Following steps are shown in the sample in Figure 5.4. In step 0, the tree is empty as well as the set of masses. In step 1, we add the peptide "TEMR" to the tree, the end-node get a relation to the protein it comes from and a pointer points to the set of masses. In step 2, we add the peptide "TEQAFY" to the tree. Because both peptides share the prefix "TE", the first node is splitted and two end-nodes are created. Because the mass is different to other values in the set of masses, a new entry in the set of masses is created. In step 3, we add the peptide "TEMR" to the

```
1  Foreach Protein sequence database, FASTA
2      Add column to protein-table and peptide-table
3      TRIE <- null
4      massMap <- null
5      for each protein in FASTA
6          UUID <- fromString(protein.sequence)
7          UPDATE protein-table where protein.uuid = UUID
8          peptides <- digest(protein.sequence)
9          for each pep in peptides
10             TRIE.insert(pep.sequence, UUID)
11     for each Node in TRIE
12         if Node.proteins not empty
13             INSERT * INTO peptide-table
14             massMap.put(mass,Node)
15     for each entry in massMap
16         INSERT * INTO pepmass-table
```

Listing 5.6: Transformation algorithm to transform protein data from FASTA format into the table schema using the radix-trie-data structure.

tree. The complete sequence is already available and only a new relation to another protein is created in the last node. In the last step, we add the peptide "TEMQG" to the tree. All the shared prefixes create an additional node, and only the suffix nodes are left with the relation to the protein the peptide comes from and a pointer to the unique mass value [ZSB+19]..

However, the difference to the original radix tree is that we are not indexing the peptides, we link protein relationship to the peptides. To adapt the radix tree to our use case of peptide deduplication, we extend the usual radix tree, using a set of protein identifiers in each node. Hence, the trie contains at this moment all information for the **peptide** table (Listing 5.6 line 13). The link to the proteins is needed to identify the proteins, based on the peptide identification. The peptide sequences are automatically deduplicated.

The next step is to calculate masses for each peptide. Hence, an additional map with the mass as key and a list of pointer to the end nodes in the radix tree as value is needed. For each of the end nodes, several masses are calculated (Listing 5.6 line 14). First, the different masses regarding the modifications and second for each of those masses three charges. This ends up in a high growth of the peptides from the trie. For each mass, the peptide sequences are calculated from the end node recursively. Beginning with the end node, which represents the end of a peptide sequence, it is needed to traverse over all nodes to the top of the trie to get the sequence completely. The last step is to insert the data from the map with masses and the peptide sequences into the **pepmass** table (Listing 5.6 line 16) [ZSB+19]..

As we show in the example in Figure 5.4, the trie needs only few nodes to store multiple peptide sequences, while the map approach gets a new entry for each different sequence. Especially, if only one letter is different, it will be only one additional node in the trie structure which is more compressed than a complete additional entry in the map. Hence, the trie structure is very promising for peptide sequence data.

5.3 Evaluation

For the evaluation, we use two different protein sequence databases - Homo sapiens and SwissProt. In Table 5.3, we show the characteristics of the data sets. Homo sapiens is a small data set with 4794 proteins and a storage size of 2.88 MB. The Homo sapiens data contains 484,479 overall peptides and 248,996 unique peptides. Finally, these peptides result in a trie structure with 295,602 nodes. SwissProt has the usual size of the productively used protein sequence databases with 255 MB storage size and 556,196 proteins and 37,403,696 peptides. After deduplication, only 23,254,068 peptides are left using 28,481,207 nodes in the radix tree.

	Homo sapiens	SwissProt
Size in MB	2,88	255
Number of proteins	4794	556,196
Number of peptides	484,479	37,403,696
Number of unique peptides	248,996	23,254,068
Number of radix nodes	295,602	28,481,207

Table 5.3: Statistical data about our two datasets for the evaluation.

The goal of the evaluation is to find the best approach regarding the applicability as an online service in the peptide deduplication step, which has the most impact on the overall transformation time. Hence, we evaluate the resource consumption and the calculation time in our evaluation in order to compare all approaches.

5.3.1 Time Evaluation

For the time measurement, we measure the overall time of the approaches. In Figure 5.5, we present the average results of 50 runs on both data sets and explain their difference on the two datasets in the following.

5.3.1.1 Homo sapiens Data Set

Regarding the Homo sapiens data set, the naive in-memory approach is 5 seconds faster than the radix tree method. Furthermore, we can see that the SQlite and the DBMS approach are very slow and need hours of calculation time. The runtime is explainable by of the amount of writes on the slow hard disk.

5.3.1.2 SwissProt Data Set

The Naive approach needs 60 seconds for the Homo sapiens data set and 58 minutes for the SwissProt data set. The tree method takes 65 seconds for the smaller data set and 60 minutes for the SwissProt data set. The differences come from the calculation of the sequence recursively from the end node. In the trie structure, the mass is calculated traversing over the nodes while in the naive approach the whole sequence is accessible. This is not needed in the naive approach. The SQlite and the DBMS approach took days on the SwissProt data and is not comparable for such sizes of protein sequence databases. Hence, the naive in-memory approach and the radix tree approach seem to be promising. The in-memory approach is a few seconds faster than the radix tree approach.

Figure 5.5: Evaluation of the runtime of the transformation process on the data sets Homo sapiens and SwissProt.

5.3.2 Memory Consumption

In the memory consumption evaluation, we only consider the in-memory approach and the radix tree method. The other two methods failed due to their bad performance. In Figure 5.6, we show the results of the memory evaluation. For both data sets, the radix tree needs less memory to store all the peptide information with the relation to the proteins and their masses compared to the naive in-memory approach. The in-memory approach needs around one gigabyte for the Homo sapiens data set and over 100 gigabytes for the SwissProt data set. For the same data, a radix tree approach needs less then 300 megabytes for the Homo sapiens data set and around 14 gigabytes for the SwissProt data set. Moreover, the memory consumption of the radix tree increases between the data sets by factor 50, while the naive in-memory approach has a factor of 92. Overall, the radix tree is very beneficial for the peptide sequence data due to its inherent duplicate compression. Hence, the radix tree method combines in-memory speed and efficient data compression for sequence data.

5.3.3 Result

The evaluation shows the benefits of the radix tree for peptide sequence data and for its transformation in our system. Using this tree structure for the peptide sequences revealed three benefits. Firstly, most of the differences of peptides are in the first levels and, hence, in higher levels of the tree the sequences end up in an end node.

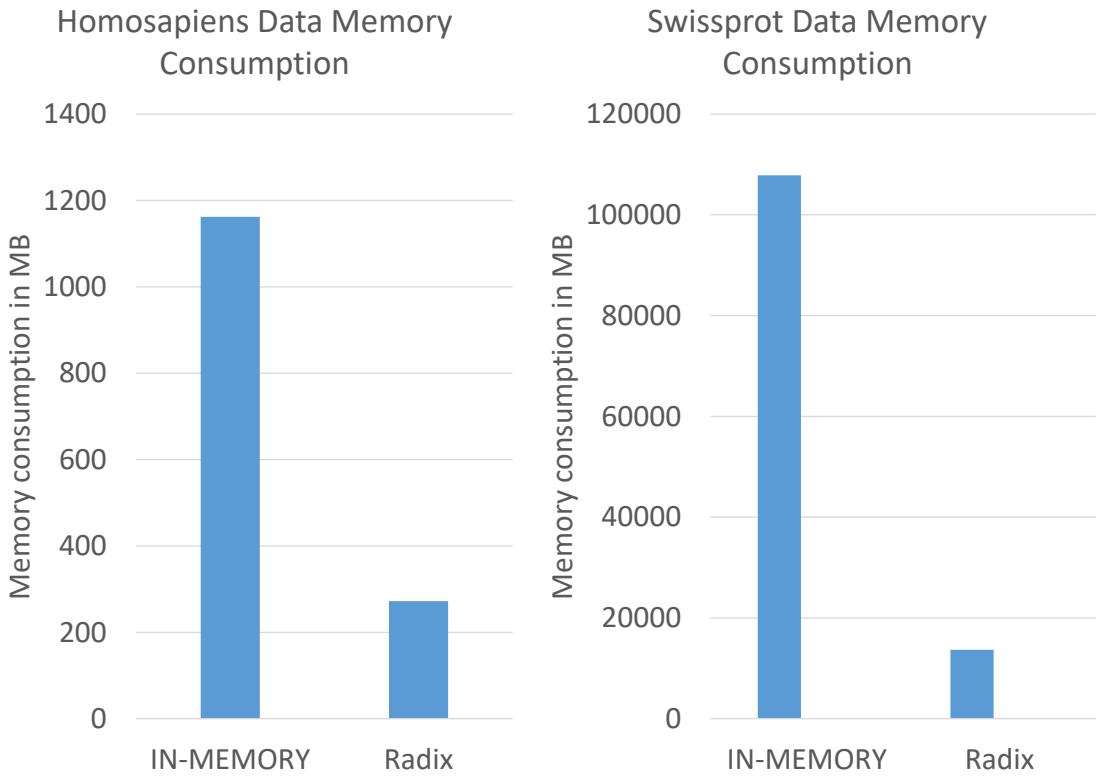

Figure 5.6: Evaluation of the memory consumption during the transformation process on the data sets Homo sapiens and SwissProt.

Due to this fact, the tree consumes less memory. Secondly, many of the differences are only due to one letter. Representing such a difference results in only one node in the tree compared to two separate peptide sequences being stored in the naive approach. Thirdly, insertion of new data automatically resolves redundancy, which is needed for further processing. We extend the attribute of the end node by a set of relations to the proteins. Due to its minimized memory consumption, it is possible to process the data completely in RAM, reaching acceptable performance.

5.4 Related Work

In this section, we present related work - the protein data indexing approach of Andromeda and the use case of the radix tree structure as a storage structure for sequence data.

The protein search engine Andromeda, part of MaxQuant Software suite, uses an indexing method on peptide masses to reduce the search space during the identification. This approach points to proteins in the protein sequence database files [CNM+11]. Nevertheless, the data is still in a FASTA file, while we structure and transform it completely. Furthermore, we remove redundancies over all the protein databases uploaded into our system.

The software MetaProteomeAnalyzer propose a relational structure to store the results in a relational database management system [MBH+15]. To process the

data, the whole data was loaded in a map structure into the memory, which took many resources of the system and corresponds our in-memory map approach.

Enrico Siragusa proposed in his thesis the radix tree as a structure to store DNA sequences as a preprocessing step [Sir15], which is similar to our approach. In our work, however, we use the radix tree firstly to store the peptide sequences and secondly to extend the end nodes with the relationship to the proteins.

The related work of the overall MStream platform is described in Chapter 8.

5.5 Conclusion

Mass spectrometry analytic platform is a future application of mass spectrometry data analysis [Ast, HKRB15]. To achieve high performance a combination of scalable cloud algorithms and fast access to the data are needed [HSZ+17]. The goal of this chapter is to analyze transformations to index protein data. The index schema brings benefits for the performance of the overall mass spectrometry analysis pipeline [SSH10]. Therefore, an efficient transformation of the protein sequence database into the index schema is needed. In our work, we investigate how to transform the protein sequence databases into the final index schema. Therefore, we show the overall transformation and different implementation approaches of this transformation. After the evaluation of the four different methods, we conclude that an extended radix tree is the best structure for the peptide sequence data in order to transform the protein data into the index schema. The radix tree for peptide sequences combines nearly the best performance and minimal memory consumption. Hence, it proves to be beneficial for our use case of peptide deduplication. We visualize the best method of the transformation in Table 5.4. The best choices of methods are bold in the table.

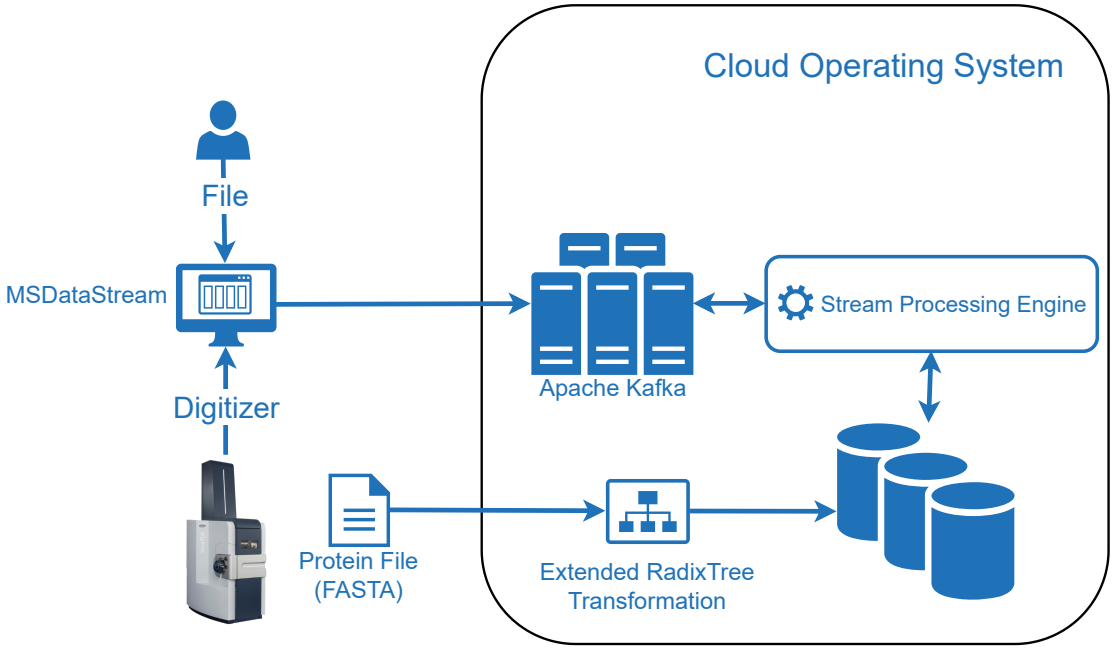

Figure 5.7: Integrated transformation service in the analytic platform.

Method	Protein Deduplication	Protein Digestion	Peptide Deduplication	Commit
In-Memory Map	No	Yes	Yes	No
HDD Map	No	Yes	Yes	No
Database Engine	**Yes**	Yes	Yes	**Yes**
Radix Tree	No	**Yes**	**Yes**	No

Table 5.4: An overview of protein data transformation approaches with the best method in bold.

In future, it is possible to use the radix tree as a storage system for protein databases and build a query engine on it. Consequently, we assume that we can speed up the current processes based on the FASTA file by using this tree structure [Nat02].

The transformation is a cloud service and the protein data need to be uploaded before the later identification process can start. In Figure 5.7, we show the integrated transformation component. The Data need to be uploaded and the radix tree us used to perform the transformation in acceptable time and efficient hardware consumption [ZSB+19].

6. Validation of Streaming Peptide Spectrum Matches

Protein identification software tools compare the measured mass spectra with already known proteins from a database and create a similarity score for each comparison. Possible protein search engines are X!Tandem, Sequest, and Mascot [PPCC99, DAC10, EMY94, RR03, MBR+13]. Protein identification is a pairwise comparison and statistically it contains false positive peptide-spectrum matches (PSM) due to measurement errors and artifacts. A common approach to remove the false positive matches is the target-decoy approach [EG09]. In this approach, a complete search runs on the experimental spectra and a protein database, which is called target search. Additionally, a second search is conducted on the same experimental spectra but this time against a decoy protein database (generated by reversing, shuffling or randomizing the original protein sequences [EG09]), called decoy search. A collection of these two search results is used to approximate the false discovery rate (FDR) for the experiment under the assumption that all hits of the decoy search are false positives.

Protein identification using the target-decoy FDR approximation is inflexible and hardly parallelizable, because the workflow needs the whole experiment data as input and incurs two searches. Flexibility and parallelism are needed to improve the state of the art performance and enable protein identification for highly parallel and horizontally scalable cloud infrastructures. Since the mass spectrometer device produces experiment data continuously, a fast data streaming architecture seems promising for the metaproteomics use case. On a fast data environment involving a huge amount of independent streaming experiment data, a target-decoy approach is no longer feasible. Therefore, we need an approach without decoy search that is still able to identify false positive matches [EG09, HSZ+17].

Gonnelli et al. show a possible usage of logistic regression on specific proteomics (only a single species) data using ranked PSMs of the Mascot search engine for the training [GSV+15]. Their decoy-free approach for specific single-species data results in 99% accuracy. However, the question arises whether similar accuracies are possible

for metaproteomics data, which contain multiple species. Addressing this question, we create a pipeline with four components for the decoy-free approach. Using our decoy-free pipeline, we developed a general classifier for metaproteomics that takes as input data from any arbitrary protein identification engine [ZSJ+18].

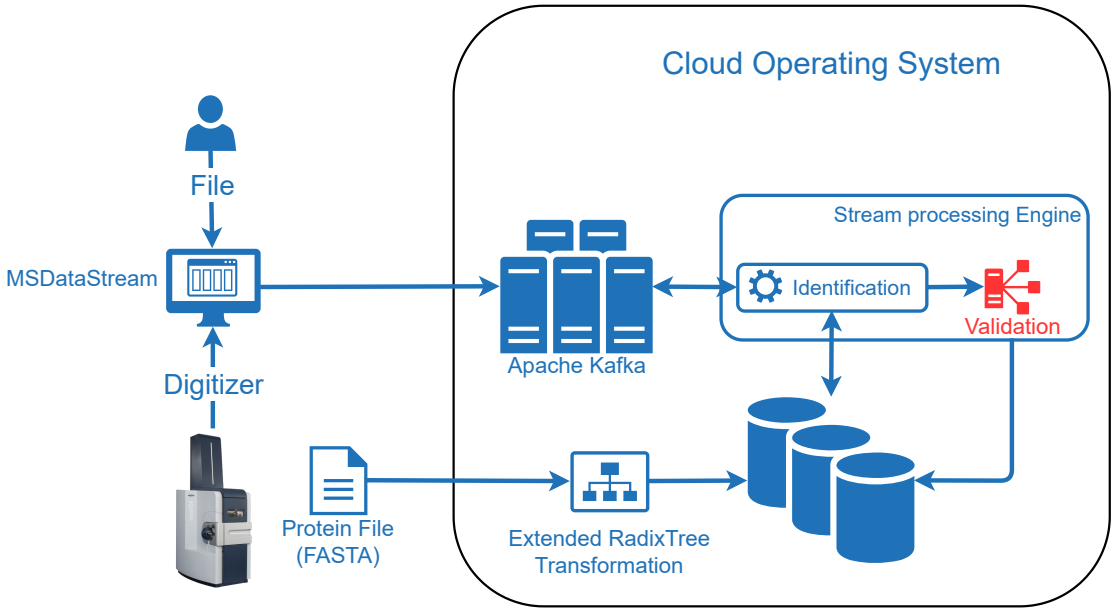

Figure 6.1: Target architecture of a classifier, which decides in real-time if a result is valid or not valid.

In Figure 6.1, we show the aimed architecture and mark the component for the validation. The validation is part of the stream processing engine and runs after the identification. Every experimental spectrum that arrives in our system is compared against the protein database. This results in a comparison between a theoretical and an experimental spectrum. The best result is classified and the valid hit can be stored in the database [ZSJ+18, RGSP07, EG09, HSZ+17].

Such classifier consists of four components. The first component is the feature extractor. To this end, we created a list of features that we need from the results of the protein search engine. Based on the feature list, we created an X!Tandem feature extractor. The next component is the CSV exchange format of the features. The third component is the trainer with the exchange format as input and a classifier as the result [PW78, PLI02]. This classifier is the last component and is used for productive systems. Using the Apache Spark library, the classifier is usable in a fast data architecture, such as the SMACK stack[1] [Est16, Wam15, MBY+16].

In this chapter, we implement a feature detector for X!Tandem results and extend the X!Tandem code to get ranked information in the result file. We use the X!Tandem results as training sets for our decoy-free classifier, which can be produced using the target-decoy method or ranked method. Furthermore, we evaluate the streaming decoy-free pipeline on different metaproteomics data. Additionally, we evaluate the process using ranked PSMs training and the target-decoy training method. We show

[1]Spark, Mesos, Akka, Cassandra and Kafka as streaming pipeline for real time data processing [Est16, Wam15]

that our approach can reach over 95% accuracy for general metaproteomics while raising horizontal scalability and effectively doubling the processing speed, since no decoy search is needed after the trained classifier.

In Summary, we make the following contributions in order to answer research question RQ4:

- Decoy-free validation: *Evaluation of a decoy-free validation approach.*
- Decoy-free pipeline: *Software pipeline to create decoy-free models.*
- Applying decoy-free to multi-species data: *Generating data from microbial bacteria.*

6.1 The Requirement of Streaming Protein Identification

Since protein identification needs a pairwise comparison and already uses the streaming implementation patterns, such as batch-wise processing, a fast data architecture seems plausible, which promises real time analysis of sensor data [Ast, MBR+13, Wam15, KPB+17, KPB+19, Est16]. For the case of metaproteomics, the mass spectrometer device produces the sensor data and streams it to the cloud system. The identification of the incoming spectra data produces peptide-spectrum matches (PSMs). The system must then decide whether the match is below the false discovery rate or not (see Figure 6.2). The goal of this chapter is to create a general classifier to decide whether the PSM is a true or a false positive for metaproteomics data.

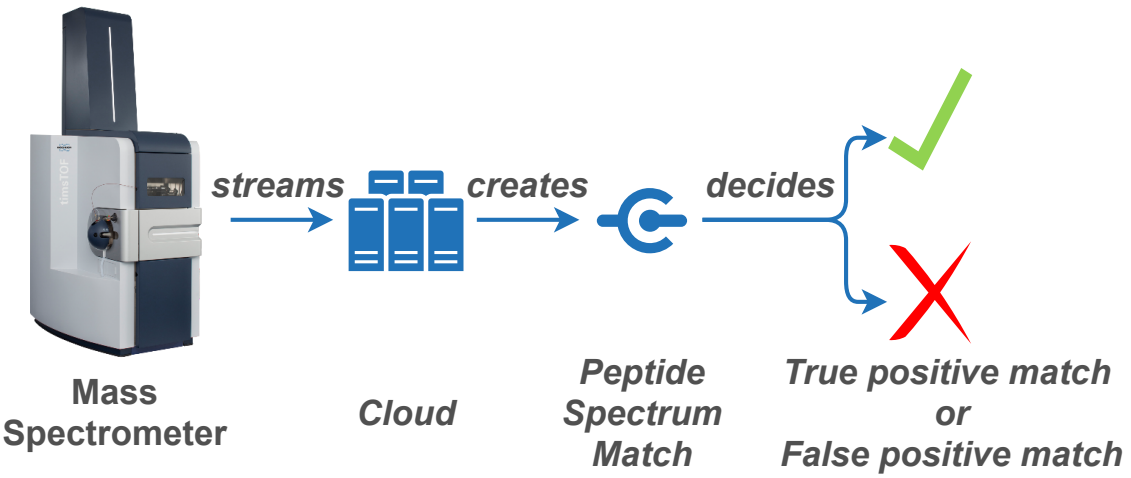

Figure 6.2: General fast data streaming architecture for protein identification with decoy-free classification of the matches.

Unfortunately, the common target-decoy FDR calculation needs the whole experiment data and makes the protein search as a streaming application unfeasible.

Hence, a decoy-free FDR estimation is necessary. Furthermore, the decoy-free approach should work generally for all proteomics and metaproteomics mass spectrometry data and with all protein search engines. To this end, we introduce four components of our decoy-free systems [HSZ+17, GSV+15, ZSJ+18].

The first component is the "Feature Extractor", the second component is the "Feature List", the third component is the "Trainer" and last component is the "Classifier".

Feature Extractor: The first component is the feature extractor, which extracts the numeric features of the matches from the results of a protein search engine. In our work we focus on the results of the X!Tandem protein search engine.

Feature List: We defined a CSV formatted feature list, which is the next component. These features represent information from the spectrum and from the peptide. In addition, it includes identification results.

Trainer: The next part of the system is the trainer. The input for the trainer component is a labeled feature list in CSV format and results in a trained model.

Classifier: The last component is the classifier, that takes the feature data and predicts whether the match is true positive or false positive according to the trained model.

Overall, the first component, the feature extractor, can be replaced, so our approach works for different search engines. The last component is the one which can be used in the productive system and classifies in O(1). The architecture with the four components is shown in Figure 6.3.

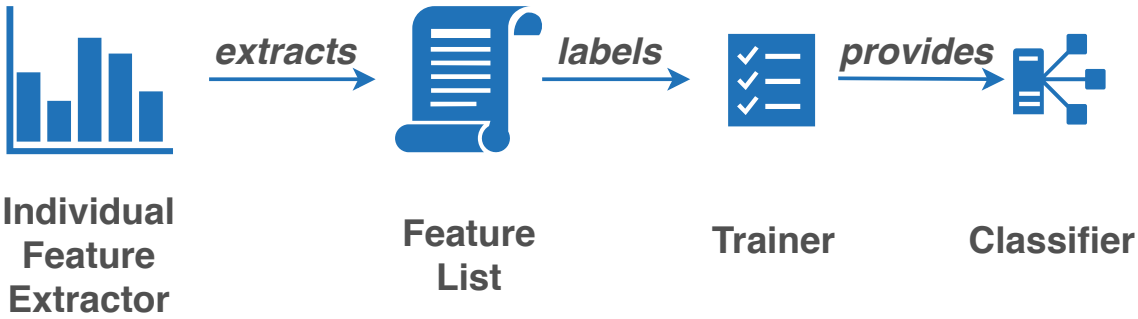

Figure 6.3: General decoy-free approach with the provided four components.

6.1.1 Individual Feature Extractor and the Feature List

The goal of these two components is to read the features that describe one peptide-spectrum-match and transform them into our defined format. The list of the selected features with their descriptions are shown in Section 6.1.1. The features are mostly not provided directly in the results, but can be calculated with the given values. The chosen features are from the decoy-free approach [GSV+15].

Computing the values for each feature is the task of the individual feature extractor. This component depends on the protein search engine. The goal is to read or calculate the features from the result data of a match. In this work we concentrate on the protein search engine X!Tandem. Hence we implemented an X!Tandem feature extractor. X!Tandem provides the results in XML format based on BIOML [Fen99, Deu12]. We mark in the type column of Section 6.1.1 if the features are directly contained in the result data (direct) or if we could calculate them (derived).

The feature list provides these features in a CSV format with 30 columns. In our exchange format, we use the first two columns for user-specific data of the PSM. In these two values we store the id's provided by X!Tandem. The last column is for labeling the PSM with TRUE or FALSE. All columns in between are the features shown in Section 6.1.1. Each row in the CSV file represents one PSM [MBY+16, PLI02].

6.1.2 The Trainer

The third component uses the labeled PSM feature list to train a classifier. We implement two different ways for the training. The first way is the ranked PSM training. In this case, for each spectrum a ranked list of matches is provided in the results. The first match is labeled as TRUE and the second rank is labeled as FALSE (see Figure 6.4). In X!Tandem, the results contained only the first rank, so we had to implement the ranked output into the X!Tandem code [ZSJ+18, PLI02].

In this method, no target-decoy-approach is required to train the model. Nevertheless, the top hits are assumed as true hits, which is also not correct. In the next approach, we take the FDR approach as ground truth.

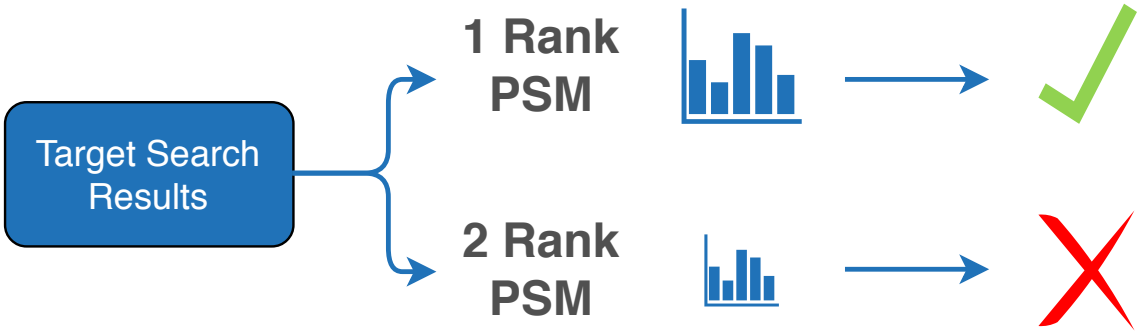

Figure 6.4: The method ranked labeling method.

The second way is the target-decoy training. For the target-decoy training, we need to extract features from the target result and from the decoy result. Using these feature lists, a target-decoy approach decides if a PSM of the target result is TRUE or FALSE. This method is independent of the protein searches and uses only the first-ranked results (see Figure 6.5). The target and decoy search results are transformed to our CSV format and are used as input for the labeling [ZSJ+18, PLI02].

6.1.3 The Classifier

The last component is the classifier, which relies on a trained logistic regression model to estimate the quality of the PSM, see Section 2.5. The classifier is the

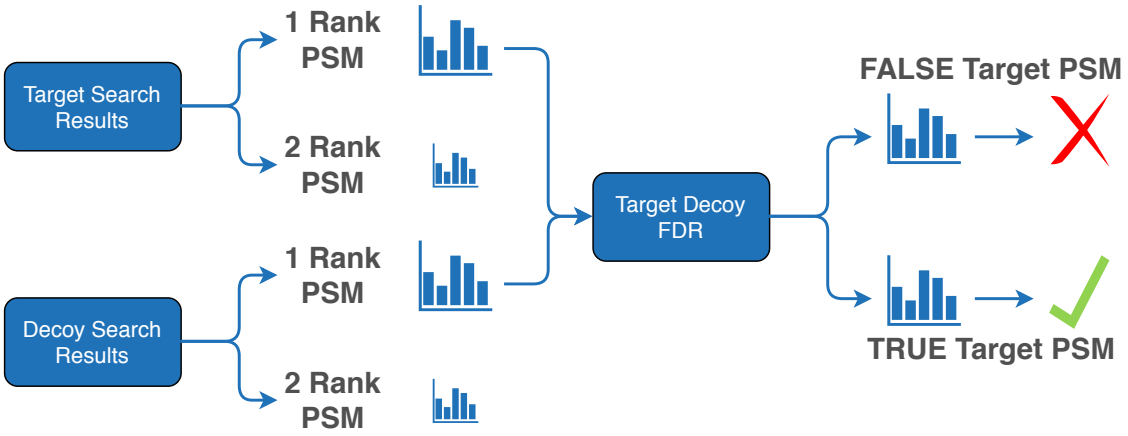

Figure 6.5: The ranked labeling method.

component which is integrated into the platform and validates the identification results as it arrives. This is the resulted service, which is used in the analytic platform. The other three components "Feature Extractor", "Feature List" and "Trainer" are not part of the productive analysis pipeline. The parts are needed for the creation of the classifier. Several classifiers can be stored in the system and loaded dynamically depending on the used data [ZSJ+18, PLI02, Wam16].

6.2 Evaluation

Evaluation of the decoy-free streaming pipeline was done using two different sets of training data and a modified version of the search engine X!Tandem. The training steps are only done once resulting in a classifier. Only the classifier is part of the production system, with negligible influence on overall performance.

To evaluate the decoy-free approach for the streaming metaproteomics workflow, we generated data from different experimental datasets using X!Tandem. We used three experiment datasets of a biogas plant and three experiment datasets from a human gut

In Table 6.2, we show the datasets. The biogas plant datasets have more PSMs than the human datasets.

In our evaluation, we tested the accuracy of the decoy-free approach on metaproteomics data. We evaluate our two training methods: the ranked PSM method and the target-decoy PSM method. In the evaluation, we test the accuracy and compare the two training methods to find the best decoy-free classifier for metaproteomics data.

6.2.1 Accuracy Evaluation

The accuracy reported corresponds to the average value of 50 runs. We combined the data to evaluate the metaproteomics use case for the general purpose. To this end, we trained the classifier using only biogas or human gut data and additionally, we mixed the datasets to show different training scenarios.

The results using the ranked training method is presented in Table 6.3. The best accuracy is 96% and the worst result is 93%. The average accuracy using the ranked training method is 95%.

The results using the target-decoy training method is presented in Table 6.4. The best accuracy is 95% and the worst is 93%. The average accuracy using the target-decoy training method is 94%.

6.2.2 Performance Evaluation

The next experiment is to measure the performance of the different approaches. In Figure 6.6, we show the performance evaluation of the approaches. The target-decoy approach needs two searches to perform the classification and needs overall approximately 1.8 times more runtime. Overall, after the training of a model, the performance is better, due to the unnecessary second decoy search.

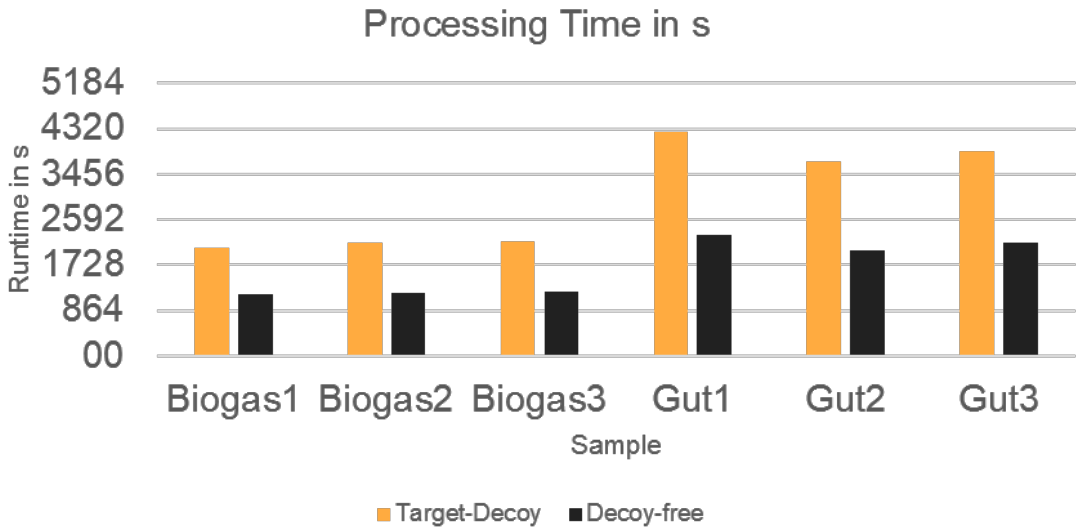

Figure 6.6: Runtime in seconds of the target-decoy classification and the decoy-free classification on different samples..

6.2.3 Evaluation Results

Our evaluation shows, both training methods bring good accuracy results to explore the experimental data. The accuracy of 95% can be considered good, because it matches the statistical uncertainty typical for metaproteomics experiments. For general purposes, the target-decoy training can be implemented for any protein identification engine, while the ranked approach has to be implemented in the software itself. For instance, Mascot provides ranked results, but X!Tandem does not. Hence, we had to add this feature in the code of the search engine. The classifier decides fast and works on independent streaming data. Using our general pipeline, we created a general metaproteomics classifier that is highly parallelizable and suitable for a streaming architecture. Since the accuracy in all scenarios is comparable, we propose to use the target-decoy method for training data generation, as it is the easiest method to apply to other search engines.

6.3 Related Work

The work done by Elias et al. on the target-decoy search strategy for providing a reasonable estimation of false discovery rate in the peptide-to-spectrum matches is a widely used approach in metaproteomics [EG09]. Another model was proposed by Gonnelli et al. for the Mascot Search Engine. This model allows fast yet reliable decoy-free separation of correct from incorrect peptide-to-spectrum matches (PSMs) when compared to the traditional decoy database paradigm, using a binary classification algorithm [GSV+15].

Granholm et al. presents a cross validation scheme for proteomics results [GNK12]. In the work, the authors present a validation of PSMs using different machine learning classificators, based on training of target-decoy-approximation.

In this chapter, we created a generalized pipeline to create a decoy-free classifier for complex mass-spectrometry data on all protein search engines. Furthermore, we used the stream processing library Apache Spark that allows integrating the classification in a fast data cloud system. To this end, we implemented an X!Tandem connector to our pipeline and evaluate the classifier using two training methods on the complex metaproteomics use case [ZSJ+18].

The related work of the overall MStream platform is described in Chapter 8.

6.4 Conclusion

The identification is a required step in the protein identification, that needs a validation of the results [MBR+13, EMY94, RR03]. To improve the performance, a fast data streaming architecture seems promising but the currently used target-decoy validation is not parallelizable because it needs all the experiment data at once [ZSJ+18, ZSB+19b, EG09].

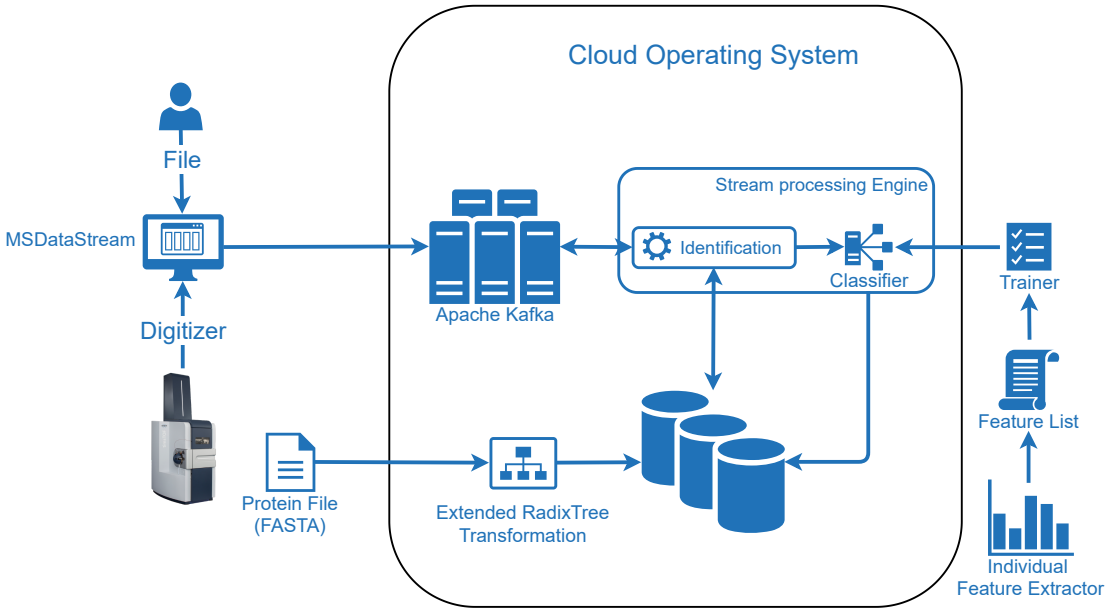

Figure 6.7: The streaming architecture including the decoy-free classifier.

In this Chapter, we show that a decoy-free approach is feasible for general, multi-species metaproteomics data with over 95% of accuracy, which can be considered as good, because it matches the typical statistical uncertainty for metaproteomics data. We created a system to build a machine learning classifier for every protein search engine. Our approach proposes an individual entry point that relies on the chosen protein search engine. We implemented one feature extractor for X!Tandem, a popular protein search software. The features describe one peptide-spectrum-match in the results of the search software. For the training, we used two different methods: the target-decoy and the ranked training method. Our evaluation shows, that both training methods have comparable accuracy and we recommend the target-decoy training method, because most protein search engines do not provide a ranked result. The final result is a classifier that removes false positive peptide-spectrum matches (PSM) from measured metaproteomics data. To classify the data, no second decoy search is needed cutting the runtime by half. The classification can be made on only one PSM, which is important for the feasibility on streaming systems.

In Figure 6.7, we show the complete pipeline including the classifier. The blue part on the right is the training that happens once. However, the resulting classifier is integrated in the identification process for each PSM.

In the future, several quality improvements on the input data and on the system itself are possible. Some of the attributes of one peptide-spectrum match (PSM) are calculated using other features. Therefore, we plan to research different weights and combinations of the PSM features. This can result in a higher accuracy or in a reduction of the input data. Furthermore, we will implement feature extractors for different protein search engines such as mascot and OMSSA [PPCC99, BLY+07]. Additionally, we will investigate whether a new classifier or a deep learning approach is feasible and brings better results to the validation of proteomics and metaproteomics data [MBY+16].

X! Tandem features	Description	Type
charge	The parent ion charge from the spectrum	*Direct*
sumI	The log10 value of the sum of all of the fragment ion intensities	*Direct*
highest peak intensity	Intensity of the highest peak in the spectrum	*Direct*
num of modifications	Number of modified residues	*Direct*
peptide length	Number of amino acids in the peptide sequence	*Direct*
modifications per peptide	Ratio of number of modifications to pep length	*Derived*
mh(group)	The parent ion mass (plus a proton) from the spectrum	*Direct*
mh(domain)	The calculated peptide mass + a proton	*Direct*
uniqueDM	(mh(domain) - mh(group)) with isotope correction in Dalton	*Derived*
uniqueDMppm	(mh(domain) - mh(group)) with isotope correction in ppm	*Derived*
sum of matched intensities	Sum of the intensities of all matched fragment ions	*Derived*
log of sum of matched intensities	Logarithm of sum of matched intensities	*Derived*
fragmented ion sum of match intensities	Sum of intensity of the matched fragment ions for the following ion series: y+,y++,b+ and b++	*Derived*
fragmented ion ratio	Ratio of number of matched fragment ions of each series to the total number of matched fragment ions of the spectrum	*Derived*
ion series count	Number of matched fragment ions for each ion series	*Derived*
longest consec match in ion series	Longest consecutive fragment ion matches for each ion series, i.e.,for (y1, y2, y3, y6) longest = 3 as (y1, y2,y3)	*Derived*
median of matched frag ion errors	Median value of all matched fragment ion errors in Dalton	*Derived*
mean of matched frag ion errors	Mean value of all matched fragment ion errors in Dalton	*Derived*
iqr matched frag ion errors	Interquartile range of all matched fragment ion errors in Dalton	*Derived*

Table 6.1: The features of a peptide-spectrum-match and how to get them from X!Tandem result file

Dataset	Amount of PSMs
BIOGAS 1	5984
BIOGAS 2	8367
BIOGAS 3	8921
HUMAN GUT 1	4819
HUMAN GUT 2	2317
HUMAN GUT 3	2685

Table 6.2: Three biogas datasets and the three human gut datasets with the amount of identified PSMs.

Training	**Test**	Accuracy in %
BIOGAS12	BIOGAS3	96
GUT12	GUT3	94
BIOGAS123	GUT3	93
GUT123	BIOGAS3	95
BIOGAS123(80)	BIOGAS123(20)	96
GUT123(80)	GUT123(20)	93
BIOGAS12_GUT12	GUT3	94
BIOGAS12_GUT12	BIOGAS3	96
BIOGAS_GUT123(80)	BIOGAS_GUT123(20)	96

Table 6.3: Evaluation Results for the ranked PSM training method.

Training	**Test**	Accuracy in %
BIOGAS12	BIOGAS3	94
GUT12	GUT3	94
BIOGAS123	GUT3	92
GUT123	BIOGAS3	93
BIOGAS123(80)	BIOGAS123(20)	95
GUT123(80)	GUT123(20)	94
BIOGAS12_GUT12	GUT3	95
BIOGAS12_GUT12	BIOGAS3	93
BIOGAS_GUT123(80)	BIOGAS_GUT123(20)	94

Table 6.4: Evaluation Results for the target-decoy PSM training method.

7. Analytic Platform for Near-Real-Time Mass Spectrometry Data Analysis

In this thesis, we present a new approach of processing mass spectrometry data on a cloud system using a fast data environment [Est16, Wam16]. We described each component of the pipeline and how to solve the challenges. In this Chapter, we combine the components in order to create the MStream prototype – an analytic cloud-based platform to process mass spectrometer data in near-real-time [ZSB+19a]. Especially in time critical use cases such as clinical diagnostics, a near-real-time processing is necessary. The metaproteomics and proteomics research area can identify marker proteins that belong to specific deceases [LQD14, NKH+17, PJW+14, PF17, ECL+12, BFWSS+15, XYF+17]. These biomarkers are similar to a fingerprint. Hence, this circumstance makes a mass spectrometer to a powerful diagnostician. Because one analysis can identify multiple diseases, a near real time analysis platform is very beneficial for this use case [PF17, NKH+17, PJW+14, LQD14, ECL+12, BFWSS+15, XYF+17].

The Chapter begins with a brief explanation of the components and how the challenges are solved. Finally, we evaluate the whole platform and the mass spectrometer analysis pipeline using MStream.

In Summary, we make the following contributions in order to answer the research questions RQ5 and RQ6:

- MStream: *Presenting the prototype MStream with all components*
- Performance evaluation: *Evaluation of MStream regarding performance and scalability*

7.1 MStream: Cloud-based Mass Spectrometry Data Analysis Platform

The mass spectrometry workflow consists of three main parts: the preparation of the experiment (Step 1 in Figure 2.1), the measurement (Step 2 in Figure 2.1) and the data processing (Steps 3, 4 and 5 in Figure 2.1). In our work, we focus on the data processing pipeline, because it is the bottleneck of the sequential workflow. Our approach starts directly at the data digitization step and parallelizes the measurement and the data processing steps, bringing the fast performance of cloud-based data processing to a mass spectrometer. In this section, we describe the challenges of the new MStream system and how we solved them [HKRB15, MBR+13, LBK+17, Ast].

7.1.1 Architecture of MStream

For a better visualization of the challenges, we show in Figure 7.1 the concept of our system. Our MStream platform needs a stream producer (Figure 7.1–1, see Chapter 4), horizontally scalable scorer (Figure 7.1–4, see Chapter 7), distributed structured protein knowledge base (Figure 7.1–3, see Chapter 5) and a smart validator of peptide-spectrum-matches (Figure 7.1–2, see Chapter 6).

The challenge of the stream producer is to stream the mass spectrometry data during the measurement without blocking the current measurement process and converting the RAW data into a readable format. These steps process each spectrum individually before sending it to the cloud-based MStream system [ZSB+19b, Mat16, Eva11]. The scorer, in turn, has to scale out horizontally to perform near-real-time analysis even with high throughput or future devices with higher resolution. For performance reasons, it is impossible to traverse all peptides every time a spectrum arrives in the system [ZCD+12, Wam16, RR03, ZSB+19a]. Therefore, a smart indexing and aggregation technique is needed to reduce the search area of the peptides. We use the criteria of current protein identification tools to define an index to the sequence database [ZSB+19, Nat02, RGSP07]. Furthermore, the smart validation method should allow the MStream system to validate every PSM individually without compromising the quality [EG09, GSV+15, ZSJ+18]. Finally, the personal data security is very important especially for a remote cloud system.

The stream producer, the data structure and the validation were already presented Chapter 4, Chapter 5 and Chapter 6. Hence, we will give only an overview of the results. For clinical use cases, the security of patient data is crucial. For this case, we assume the security as solved by mapping the experimental data in the stream producer to a specific patient without revealing or sending any personal data to the cloud. Hence, in this thesis no security aspects are tackled. The contribution of this chapter is the scalable scorer and the overall evaluation of the MStream analytic platform. In this section, we will describe our solution for the challenges beginning with the validation, followed by the stream producer and the persistence layer. Finally, we will describe the identification process and the MStream system [ZSB+19a].

7.1.2 Streaming Validation of Peptide Spectrum Matches

The validation is needed to trust the results but the state-of-the-art target-decoy method is not applicable to streaming experiment data. For a streaming validation method, the goal is to classify a peptide-spectrum match as valid or non-valid

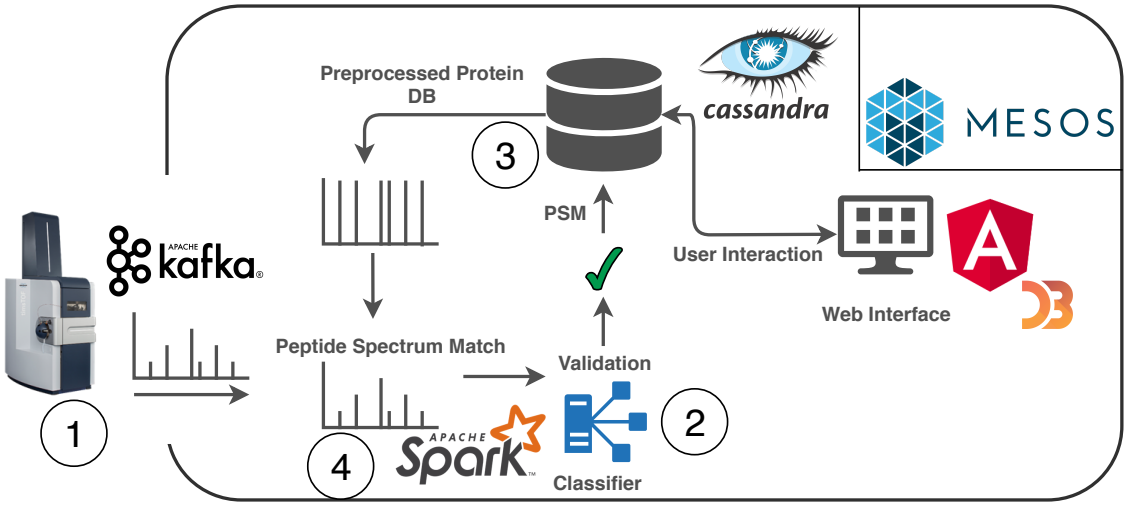

Figure 7.1: Architecture of the MStream prototype, the analytic platform for real-time diagnostic of mass spectrometry data [ZSB+19a].

directly as they are generated. We solved this classification problem using logistic regression, a machine learning technique. In Chapter 6, we showed that our solution can speed up the identification by a factor of 1.8. At the same time, we reach an accuracy over 95%, which is enough for this most uses cases, see Chapter 6 [ZSJ+18]. Additionally, the classifier is independent from the experiment; it only has to be trained once for a given device. As a result, our classifier helps us overcome the first challenge towards a stream-based analysis workflow.

7.1.3 MSDataStream: Stream Producer for a Bruker Mass Spectrometer

Since every manufacturer has its own file format to store the measured data, we collaborated with the Bruker Daltonik GmbH to access the data. We adapted the given API to read the data during the measurements in batches without blocking the actual measurement. Furthermore, we added preprocessing steps to increase the quality and reduce the noise of the data. While in the current pipeline, the conversion to a readable format begins after the measurement, we added the conversion into the stream producer to transform each data item, which we send to the cloud or write to a file. The application runs locally on the mass spectrometer computer and collects the data continuously. Further adapters for the manufacturers are planned for future work and would need additional collaborations with the device companies, see Chapter 4 [ZSB+19b].

7.1.4 Preparation of Protein Data

In order to implement the streaming identification process, we analyzed other tools for protein identification. The first requirement is that all the data should be available at once and the second one is that the experimental data should be small and fit into the main memory. In our system, the input data is not completely available and we can access only a few spectra at the same time. Accordingly, in our system, we iterate once over the mass spectrometry data and have to access the protein and

peptide sequence data for each spectrum. Due to the data size of the protein knowledge base, we reduce the searching area, using a sorted index on the total mass of peptides, based on the work of Cox et al., see Chapter 5 [CNM+11].

In our system, we extend this indexing by the charge[1] property of peptides and additionally persist the peptide modifications[2] in our system. Furthermore, we store the meta information of each protein of the knowledge base in an additional column, creating wide tables mapped to the sequence in a row.

As shown in Figure 7.2, the row key of the protein table is the protein sequence, while the meta information of the proteins are stored in additional columns. Additional information of a protein describe the functionality and the biological environment of its origin. Each knowledge base has an own column for the meta information but is mapped to the unique protein sequence in the protein table. This strategy removes redundant protein sequences across multiple protein databases without losing the meta information. In the peptide table, the meta information is the collection of proteins, which contains the peptide. In the pepmass table, the data is clustered by the protein knowledge base and indexed, using sorted charge value of the peptide, and sorted by the total mass of the peptide. Since modifications of the peptides change the mass, we store the modified peptide additionally in the pepmass table as a redundant peptide sequence. The parts of the peptides (Amino acids) can have a modification which changes the mass of the amino acid and hence, the total mass of the peptide. As described in Chapter 5, we precalculate the modified masses in beforehand and store the peptide with different masses redundantly. Accordingly, no further calculation during the similarity function is needed.

While state-of-the-art software iterates through the protein data and applies those operations during the processing, we use this indexing as a bottom-up filtering method for the protein identification engines. For the search, the query collects the data with the same charge and total mass with a tolerance defined by the user and the mass spectrometer.

Our storage schema of the protein data is specialized to reduce the amount of comparisons for the diagnostic use case. The index structure is column-based such as the one used by Apache Cassandra [LM10] or in Elf [BKSS17]. The preparation of the data is not time critical, since every disease has one knowledge base, which is used for every sample [ECL+12, BFWSS+15, XYF+17]. Of course, the schema of MStream has additional components for user management, data management and results. The complete schema is not part of this work, because it has no relevance for the thesis topic.

7.1.5 Online Processing of Streaming Spectra

The core of the identification process is the scoring function. This scorer component unites all other components of the system.

[1] Each spectrum has a charge that needs also to be applied on the theoretical spectrum of the peptide for the comparison [MFT+13].

[2] Depending on the biological preparation and the properties of the samples, some parts of the peptides are modified and changed the overall total mass [MFT+13].

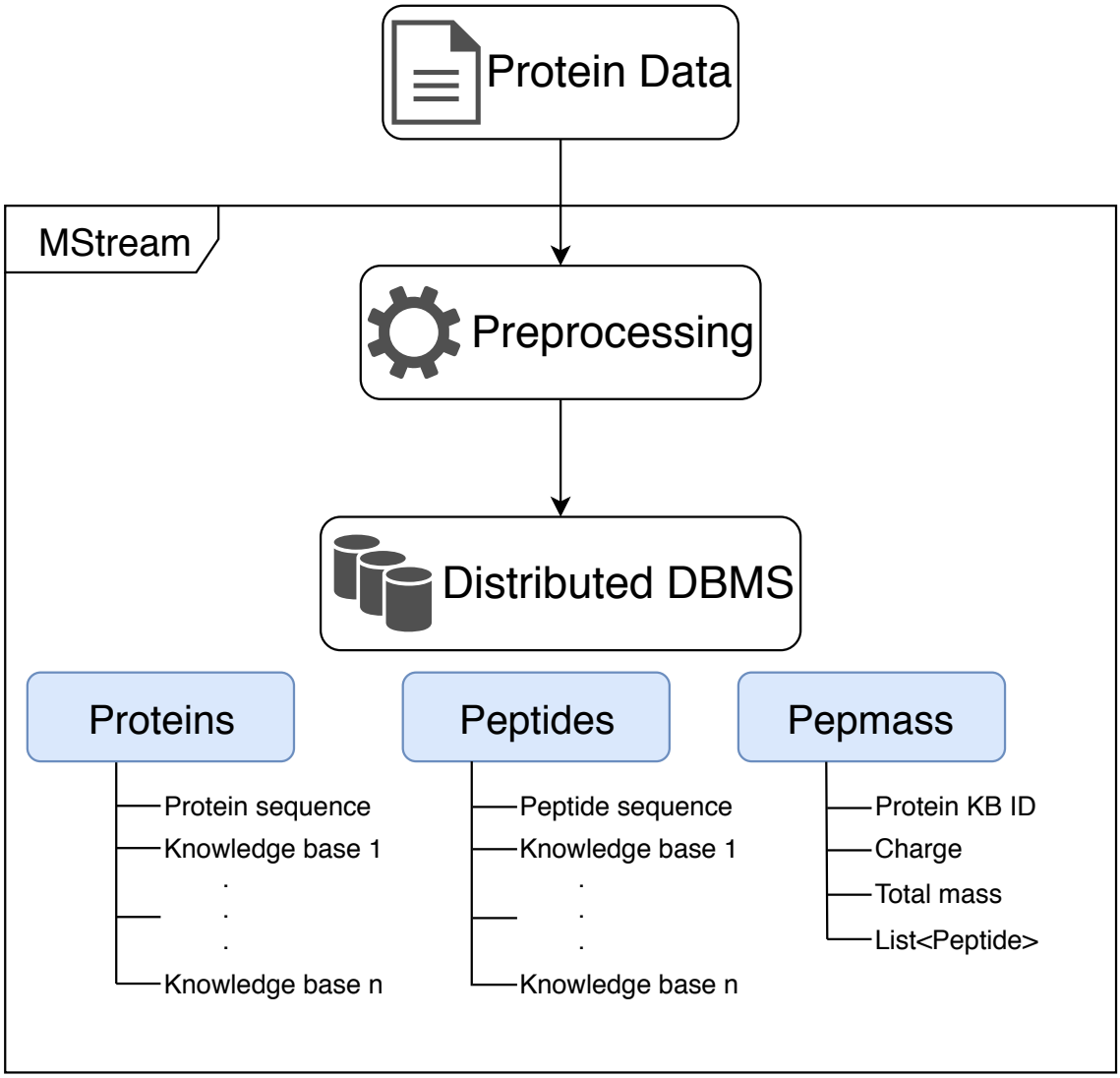

Figure 7.2: General structure of preprocessed protein data in the MStream system.

In the protein identification step of state-of-the-art tools, the experiment data is processed in one bulk. Whereas in our architecture, the experimental spectra are not available completely and arrive in our cloud system periodically. In the Scorer, the following components are required: the stream producer (Chapter 4), the validator (Chapter 6) and the storage (Chapter 5).

Since we have to serve multiple clients in parallel, we add user-dependent parameters into each incoming message, in addition to the experimental data. In this way, we separate each spectrum and use one message channel for the experimental data, making the spectrum message the smallest parallelizable unit in the system. The parameters define the protein data for the identification and the validation model. Additionally, we provide the error mass tolerance in the spectrum message to define the range of the corresponding peptides, which we query from the storage system.

In Listing 7.1, we describe the algorithm of one thread (worker service in the cloud). First, the worker collects a batch of messages. Each message contains the textual representation of the spectrum. Next, each spectrum of the batch is processed

individually. For each spectrum, the algorithm selects all peptides that must be considered for a comparison from a specific protein knowledge base. Hence, each peptide in the knowledge base is compared to the spectrum and the match with the best score is kept as a result. Using our logistic regression classifier, the best peptide-spectrum match is validated. Finally, the PSM and the spectrum are stored in our storage system.

The scoring function is based on the weighted dot product, which is adapted from the X!Tandem protein engine. Other similarity functions can of course be applied in our system as a part of future work.

```
1.  batch ← incoming messages
2.  for spectrum in batch do
3.   minMass ← spectrum.mass − error
4.   maxMass ← spectrum.mass + error
5.   peptides ← minMass < ? < maxMass
6.   maxPSM ← −∞
7.   for peptide in peptides
8.    psm ← score(spectrum, peptide)
9.    if (psm.score > maxPSM.score)
10.    maxPSM ← psm
11.   endif
12.  endfor
13.  validate(maxPSM)
14.  store(spectrum)
15.  store(maxPSM)
16. endfor
```

Listing 7.1: Algorithm of the MStream scoring worker [ZSB+19a].

7.1.6 Putting It All Together

The state-of-the-art technology needs the experimental data as a file and the protein data as a file, while our system has additional requirements on the technical side and on the infrastructure level. When the data processing starts, a protein knowledge base must be already in our storage, see Section 7.1.4. Consequently, the protein database must be uploaded and prepared for further processing in our system. Furthermore, we need a trained model for the specific device that enables validation of the experimental data, see Section 7.1.2. In the end, our stream producer must be installed on the mass spectrometer side, see Section 7.1.3.

The system is deployed on the SMACK stack, as our implementation of the fast data architecture, see Figure 7.1. Additionally, a lightweight web UI show us the results of an experiment during the identification process.

In Figure 7.3, we show the communication between the components and how they operate in a sequence diagram. The stream producer asks for new data periodically

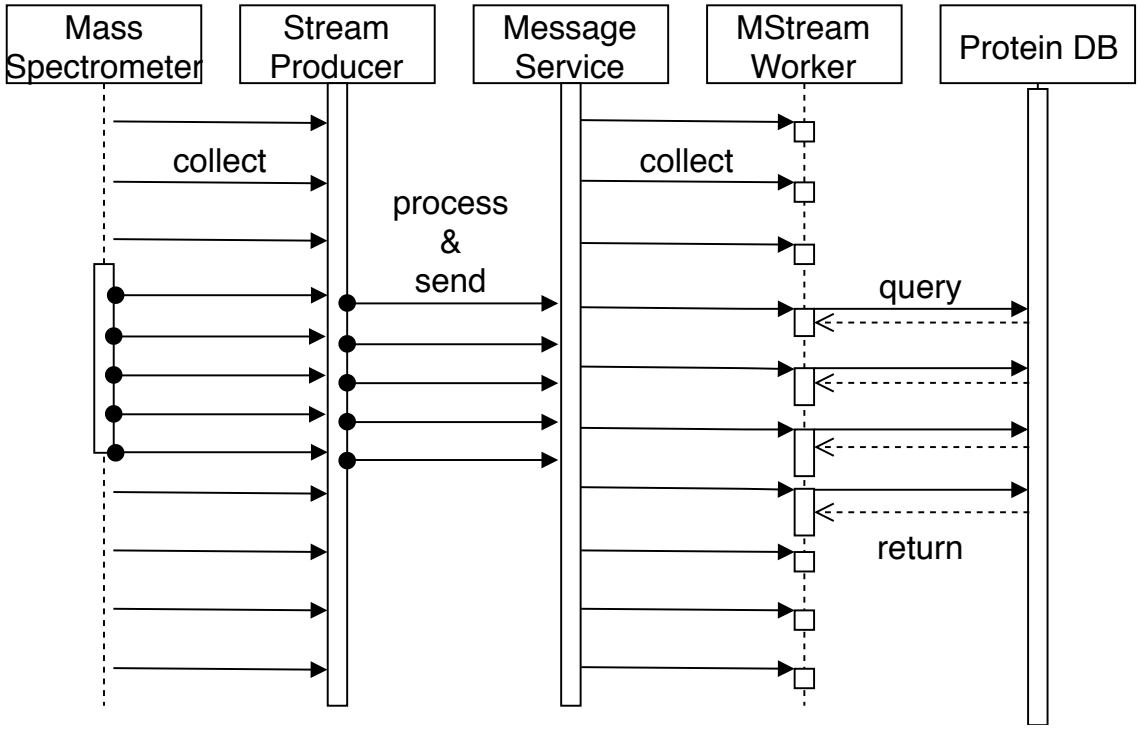

Figure 7.3: Sequence diagram of MStream components [ZSB+19a].

and starts to collect the data as the mass spectrometer begins with the measurement. The producer processes and transforms every experimental spectrum and sends the data to the message service. The MStream worker with the scoring task subscribes to the message service and asks for new data periodically. The subscription time of the scorer defines in which time periods the system asks for a new data batch from the message service. For the protein identification, the scorer queries peptides from the protein knowledge base and stores the results.

7.2 Evaluation

From a biological perspective, our system provides same results as other protein search engines, since the similarity function and the pre-filtering are based on the state-of-the-art approach. Hence, in this chapter, we focus on the evaluation of the performance.

This section is structured as follows: firstly, we present the evaluation setup of the prototype. Secondly, we describe the experiments. Finally, we present our results.

7.2.1 Evaluation Setup

In our work, we implemented the MStream system prototype[3], which we use for the evaluation. We used a de.NBI[4] OpenStack system to run 15 virtual machines, using Harshi Corp Terraform for fast management of the infrastructure. Overall, our OpenStack project offers 160 virtual CPUs and 720GB RAM. We use Apache

[3]https://git.iti.cs.ovgu.de/zoun/mstream
[4]German Network for Bioinformatics Infrastructure. (www.denbi.de)

Mesos as cloud operation system. On Mesos, we deployed all the services and components of MStream, such as Apache Cassandra for the structured storage, HDFS to store our trained validation models, Apache Spark for processing the data and scoring and Apache Kafka as a message broker. For additional services, we used the Spring framework and SparkJava on a Jetty application server. The stream producer software is a JavaFX implementation and runs on the machine of the mass spectrometer. The stream producer can currently only read Bruker experimental data, but can be easily extended for further devices.

For the state-of-the-art tool, we used a local computer. As a local machine, we used an ASUS UX360 with Intel I7-7500U/BGA with 2 cores and 16GB DDR3-RAM.

For the evaluation data we used PASEF experiments of *Escherichia coli* (*E.coli*) bacteria with 57,758 spectra (5.37GB) to evaluate the scalability and to compare to the state of the art. The data is generated by a PASEF TIMSTOF mass spectrometer from Bruker Daltonik GmbH. As knowledge base of proteins, we adopted the commonly used UniProt SwissProt knowledge base, containing 556,196 proteins, which results in 23,934,321 peptides. The chosen knowledge base contains much more proteins than necessary for diagnostics of specific disease, because we test the performance and the impact of the amount of proteins to the platform [ECL+12, BFWSS+15, XYF+17, PF17]. In the next Section, we describe the evaluation of our MStream platform.

7.2.2 Evaluation Experiments

The goal of the evaluation is to show the applicability for near-real-time identification that is especially needed for medical diagnostics. MStream performs in near real time if the overall processing rate in amount of spectra per second is the same as the measurement rate of the mass spectrometer in amount of spectra per second. Hence, we first analyze the persistence component (see Section 7.1.4) regarding the data size that we create, using our index approach. Additionally, we analyze how indexing reduces the search range and how fast our system queries the data from the persistence layer.

As we propose a central system, we observe if our system can handle multiple devices in the same time, providing real-time analysis. In this experiment, we analyze the scoring time by measuring the performance of each scorer, each step in a scorer and the scalability factor of the system.

Finally, we compare our system to the state-of-the-art protein identification software X!Tandem to evaluate the competitiveness of MStream.

7.2.2.1 Evaluation 1: Structured Protein Knowledge Base

In this experiment, we present the results collected from the knowledge-base component (see Section 7.1.4). After uploading the data, in this case the SwissProt data, the amount of data changes because we remove duplicates on the protein level and on the peptide level. This reduces the amount of peptides from 23,934,321 to 14,579,004 non-redundant peptides. Considering the precalculation of charge property and the modifications for each peptide, the amount of the peptides in the pepmass table

rises to 111,183,434. Hence, our system increases a protein sequence database from original size of 500MB proteins to 2GB, but contains aggregated and ready to use peptides in the pepmass table.

In Figure 7.4, we show the distribution of all the peptide sequences in our storage, regarding the charge property and analyze the load balancing of the current system. All of the 111,183,434 peptide sequences are stored in 4,814,243 rows, grouped by charge and by the total mass. The amount of peptides with charge 3 (blue line) are in the range of 170 and 1300 Dalton. The peptides with charge 2 are in the range of 300 and 1950 Dalton, while data with charge 1 is between 600 and 3700 Dalton.

Grouping by the charge attribute, distributes the data across the rows. Nevertheless, the amount of peptides is not equal which results in unbalanced query results. Of course, the amount of data returned by the query for each spectrum has an influence on the runtime. We analyze this influence as next.

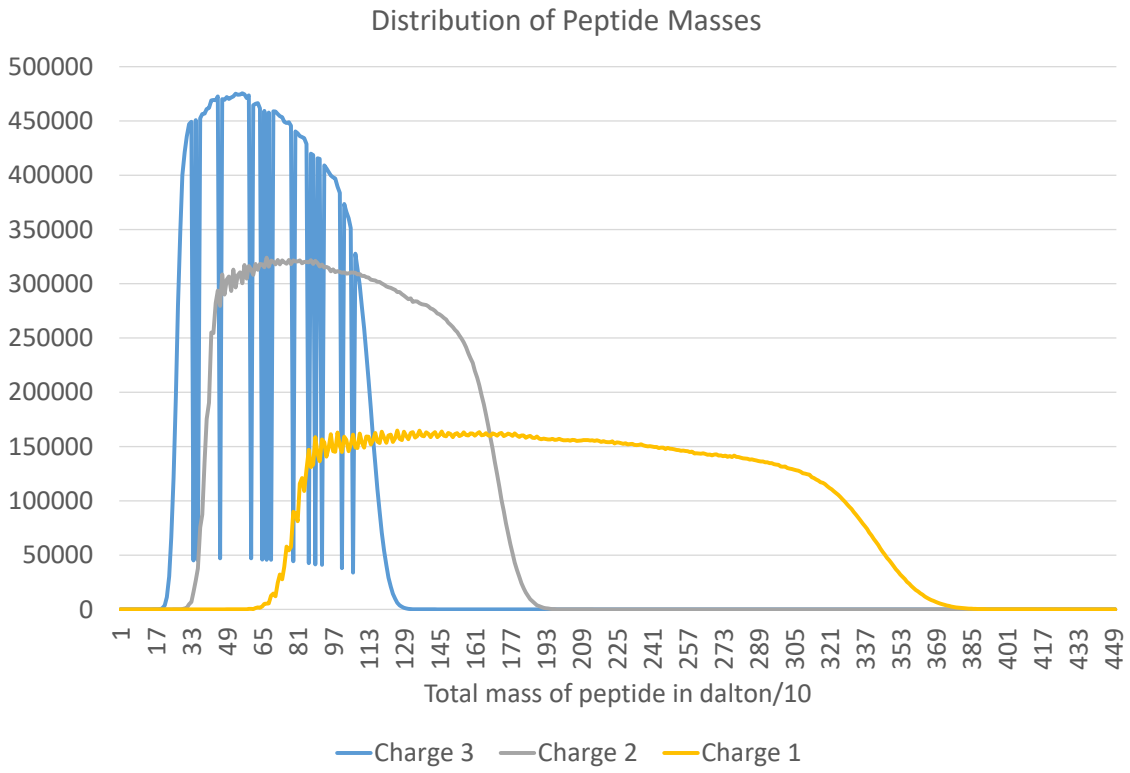

Figure 7.4: Histogram of peptide masses in the database of MStream.

We test the query time to select the data with different tolerances on their masses. The error tolerance is calculated in parts per million (ppm) and depends on the given device. The range of 100 ppm can be used for older devices, 20 ppm is for current devices, 10 ppm is for current high class devices and 1 ppm is for future devices with even higher precision.

In this experiment, the data input rate is 189.42 experimental spectra per second, which describes the rate of six or seven mass spectrometers based on our experience with the Bruker mass spectrometer (on average 27 spectra per second). In Figure 7.5, we show the query time on the left y-axis (orange bars) and the amount

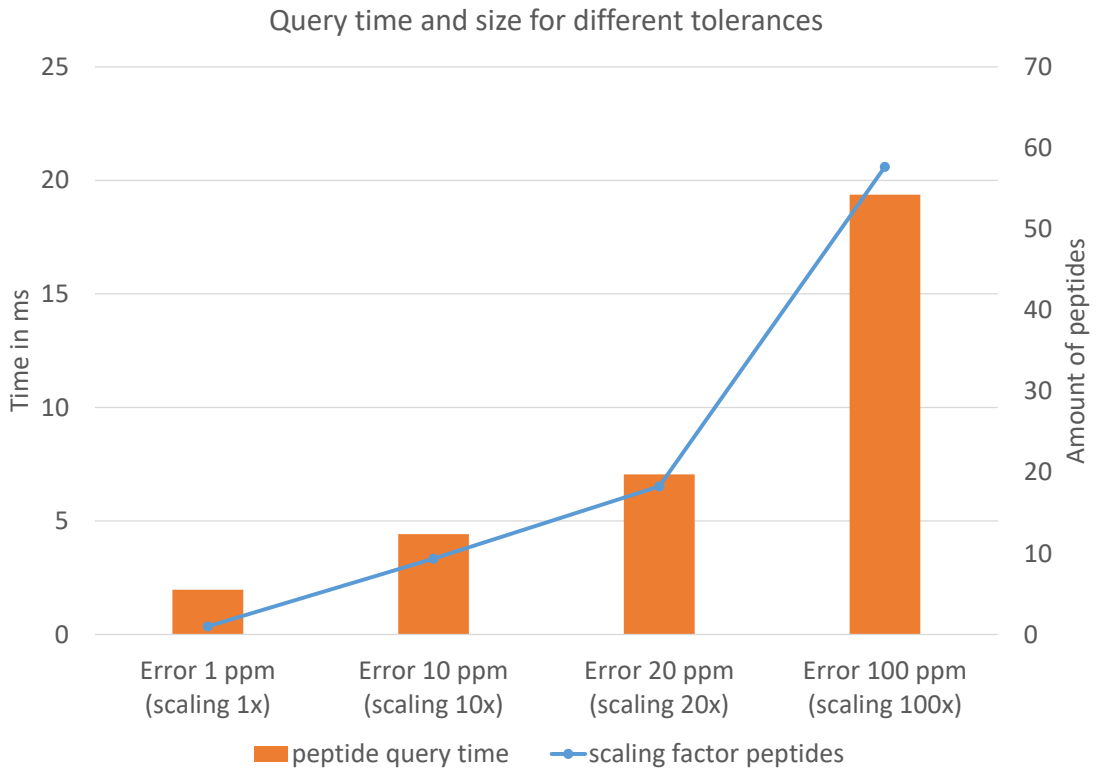

Figure 7.5: Query time and the query size regarding different error tolerances.

of the peptides on the right y-axis (blue line), selected when querying for different tolerances. The evaluation is an average of multiple runs (50) and measures the time for each spectrum. For 1 ppm, the query amount of rows that are selected is 12 on average, the amount of peptides is around 250 and the query is performed in 2 ms. Using 100 ppm, the query time increases to 19 ms, the amount of peptides grows to 14,420 selected from 818 rows in average. For 10 ppm the query needs 4.5 ms in average and 2,337 peptides are selected, while for 20 ppm the query took 7 ms with 4569 peptides in average. While the scaling factor from 1 ppm to 100 ppm is 100, the factor for the query time for these experiments is 9.9, and the factor for the peptides is 57.7. The MStream platform scales linearly allowing more workers work in parallel.

We show the influence of the tolerance to the query time and propose to optimize this parameter for specific real world scenarios. Especially for repetitive measurements, such as diagnostics of a disease, it is beneficial to use the minimum tolerance, based on experience of the measurements [PF17, ECL+12, BFWSS+15, XYF+17]. This will reduce the calculation time and the costs of the analysis.

7.2.2.2 Evaluation 2: The Scorer Performance

In this section, we evaluate the worker (the scorer) scalability and the performance of the MStream system. Our goal is to analyze the possibility to operate with multiple mass spectrometers and provide the results in near-real-time for every user.

To this end, we increased the spectra input rate on the one hand, which will increase the amount of spectra per batch and on the other hand, we increased the error tolerance, which will increase the amount of peptides.

This experiment runs with the error tolerance of 20 ppm and a subscription time of 10 seconds.

In the first experiment of the scorer evaluation, we use different spectra input rates to simulate multiple devices and users. For each run, we measure the time of each step of the scorer. The first step is the query of the peptides (Listing 7.1, line 5), the next step is the scoring of the spectrum and the peptides (Listing 7.1, line 7 to 12), the third step is the validation of the PSMs (Listing 7.1, line 13), followed by the insert functions of the PSM (Listing 7.1, line 15) and the spectrum (Listing 7.1, line 14).

All runs achieve real-time processing with the time deviation of the processing time of the last mass spectrometry data, usually a few seconds. In this experiment, we increase the input rate and simulate multiple devices sending data at the same time. Our goal is to scale out the workers to serve each user's device in real-time. Therefore, we analyze how the system adapts to additional mass spectrometers, which reflects the individual performance of the worker and the overall performance of the system.

In Figure 7.6, we present the results of this experiment. The measured time is the average value in ms per worker and per spectra. We evaluate our system with simulated input rates 41, 104, 196 and 727 spectra per second on average. Because one mass spectrometer sends on average 27 spectra per second, which can be processed by a single worker, the input rates are generated using already measured experiments processing the data with the stream producer (see Section 7.1.3). We measured the time for each part in the algorithm to analyze the differences between different input rates. Additionally, we measure the minimum number of workers that are needed to perform the analysis in near-real-time.

In the chart in Figure 7.6, we can observe that the input rate does not influence the single processing steps of the worker, but influences the amount of workers. The individual performance deviation is around 1 ms. We lead back this deviation to the outscaled workers and the cloud latency. Additionally, we can observe that a worker scales linearly with the input rate. The processing speed of one worker with the parameters of these experiments (using tolerance with 20 ppm) processes around 45 spectra per second. Increasing the rate of incoming spectra, the amount of processing workers scales linearly without influence on the individual worker performance. To this end, the scorer spends most of its time in the pairwise comparison.

The different sections of the algorithm are summed up to show the overall time for one spectrum. The growing input rate increases the amount of spectra in one batch, thus, increasing the amount of comparisons of the worker but not the time of the single spectrum analysis. This bottleneck is handled without scaling the workers. Hence, the processing time of one batch is increased and leads to longer waiting time of the last batch of the mass spectrometer.

In the next chart, we show the performance of one worker regarding the error tolerance. We ran this experiment with an input rate of around 200 spectra per second

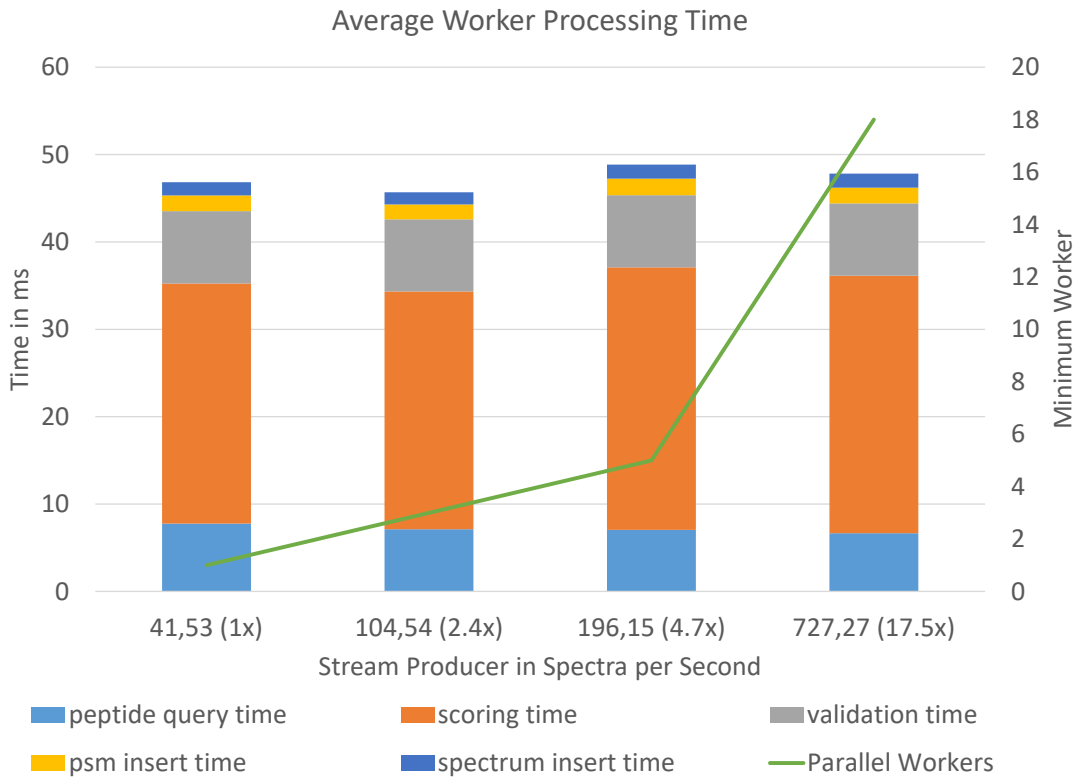

Figure 7.6: Average runtime in ms depending on the data input rate.

in each run and a subscription time of 10 milliseconds. The input rate simulates high utilization, which usually needs around 5 workers to perform in real-time. In Figure 7.7, we show the average performance of a single worker in relation to the error tolerance. For 1 ppm the scorer is faster, since the amount of peptides is decreased. Additionally, the scorer performs very fast with 335 spectra per second for one worker. However, for 100 ppm, one worker processes 15 spectra per second and one worker cannot perform in real-time anymore. Hence, the scorer scales out up to 17 workers to reach near-real-time processing speed.

For the next chart in Figure 7.8, we analyze the influence of the tolerance of the single steps on the scorer. Again, we ran the experiments with different error tolerances, measuring the time and relating the scaling factors to the amount of peptides. We can see that the time for insertion and validation does not change but the tolerance influences the amount of peptides. Accordingly, the time for querying the peptides and scoring increases. The scoring time scaling factor is 39 at 100 ppm. Hence, the scoring scales slower than the error tolerance regarding the scaling factor, because the input rate increases the amount of data comparisons more than the tolerance. This experiment shows that the influence of the input rate is much higher with the weight of 1 than of the error tolerance with weight 0.17 regarding the scaling factors[5] of the worker.

[5]Scaling the spectra input rate by factor 18, increases the workers from 1 to 18, while scaling the tolerance by factor 100 increases the workers from 1 to 17

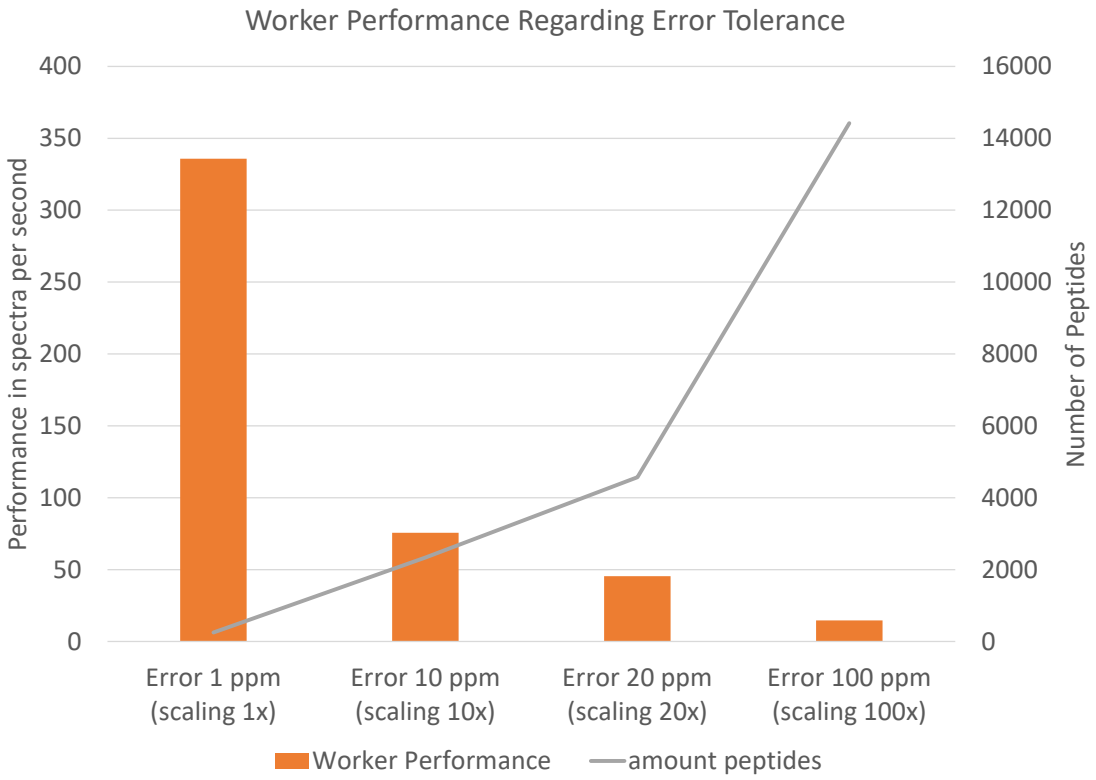

Figure 7.7: Average worker performance in spectra per second by increasing the error tolerance for the peptide query in MStream.

In the end, we show that increasing the error tolerance or increasing the input rate affect the runtime of the worker and the overall system. Additionally, we see that the scaling factors are different and the amount of mass spectrometry data affects the process more than the amount of peptides.

An evaluation of different subscription times is not necessary, since it changes the amount of spectra in a batch, which leads to the same results as the evaluation of the input rate.

7.2.2.3 Experiment 3: Comparison to X!Tandem

After evaluating the MStream platform, we now compare the runtime to the state-of-the-art tool X!Tandem which runs on a local machine and uses one thread, to the runtime of one worker of our system. X!Tandem needs all mass spectrometry data as input at the beginning, while our worker can start processing a single spectrum. Nevertheless, we compare only the identification time.

In Figure 7.9, we show the average performance of a mass spectrometer based on the experience with a Bruker device (on average 27 spectra per second), compared to the physical maximum of the device (100 spectra per second). Those rates reflect the speed at which one device and possible future devices deliver spectrum data.

X!Tandem performs on average 18 spectra per second, using one thread and our single worker performs 45 spectra per second. The search parameters were similar

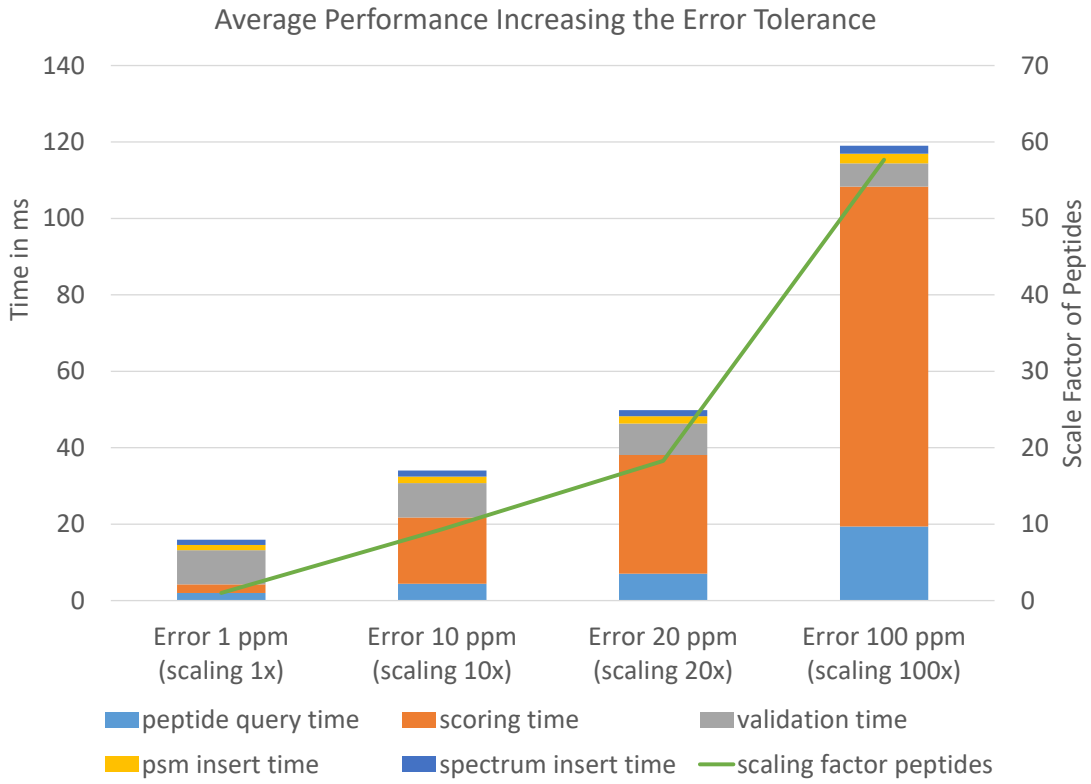

Figure 7.8: Average performance in ms by increasing the error tolerance for the peptide query in MStream.

for both tools. Overall, our single worker performed better than single-threaded X!Tandem software.

The comparison is not entirely fair, since X!Tandem runs on a single machine, using one thread and our system runs on a cloud environment with at least 37 virtual CPUs. Based on the scaling factors from this experiment, we can calculate, when it is worth to change to our system regarding the CPU usage. In Figure 7.10, we show that on 58 CPUs and 990 spectra per second our system performs faster. Hence, our system performs better if the data amount comes from 30 or more mass spectrometers.

The results showed that MStream has the capability to perform near-real-time analysis by fast querying of the peptide data and scales out if needed. We targeted out the three parameters - error tolerance, input rate and subscription time, to optimize the performance of the system. In the end, we examined the hardware consumption, which is acceptable for a central cloud-based analytic platform that serves thousands of mass spectrometers for emerging use cases such as clinical diagnostics.

7.3 Discussion

Regarding the goal of our MStream system to implement a central system for real-time diagnostics of mass spectrometer data, we discuss the results of the evaluation.

In Section 7.2.2.1, we show that the amount of data for the preparation explodes by factor 30 even if we use only two modifications, three charges and one kind

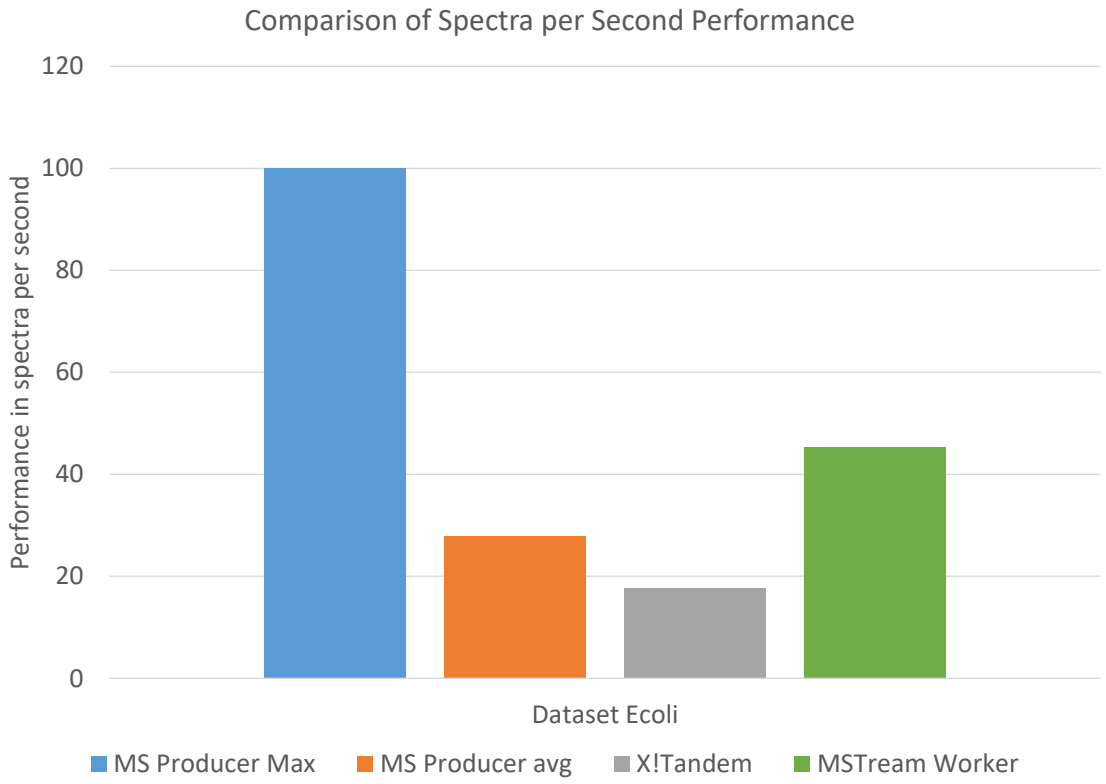

Figure 7.9: Comparison of the performance of X!Tandem, MStream and a Mass Spectrometer.

of splitting method. This shows that our data preparation greatly increases the requirements on hard disk memory. However, given the low prices for hard disks, we nevertheless argue that the speed-ups enabled by our data preparation step justify the increase in memory requirements.

Hence, the data preparation needs to be deployed for each protein sequence database that can be used for the analysis. For research, more flexible data preparation is needed at the expense of performance. The distribution of the peptides in Figure 7.4 reveal the differences between the charges. Especially peptides with charge 3 have a high number of peptides in a smaller range. Further research can optimize the balancing of the data among the rows in the pepmass table.

Next, we examined the query performance and the query size regarding different tolerances, see Figure 7.5. The scaling factor of the query time and size scales linear and slower than the error tolerance. Since the error tolerance depends on the precision of the mass spectrometer, this should be considered during the experiments using the minimum tolerance.

In Section 7.2.2.2, the MStream system provides the results in near-real-time. In this experiment, we analyze the weighted impact of the number of peptides and spectra. First, we increased the input rate to analyze the out scaling and how it influences a single worker. We examined in Figure 7.6 that the scaling factor of the workers is linear and each worker delivers the same performance during the processing. In Figure 7.7, we examined the influence of the peptide size to the worker

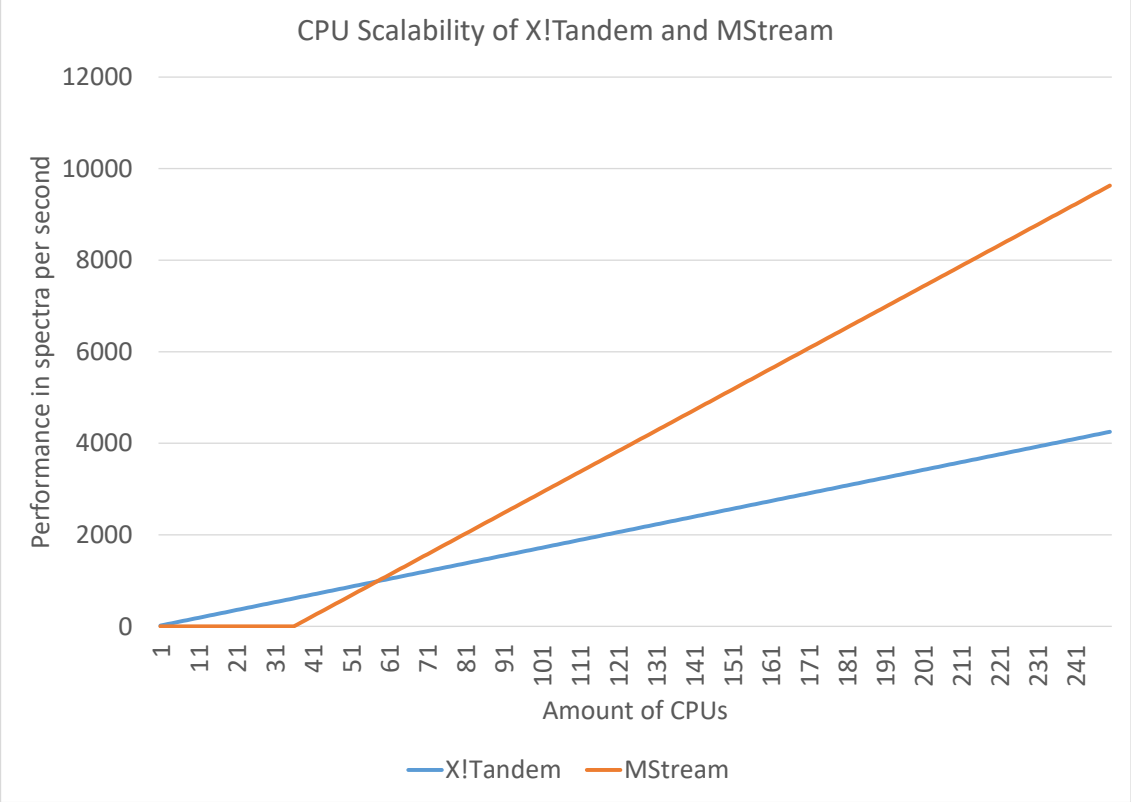

Figure 7.10: The performance regarding the CPU consumption of MStream and X!Tandem.

performance. In Figure 7.8, we show the influence on the single components in the scorer. In the end, we showed that the amount of spectra has a weight of 1 while the peptides influence with 0.17. We propose to adjust the prefiltering methods in the stream producer (Section 7.1.3) for individual diagnostic scenarios to reduce the calculation effort and costs on the cloud.

In our last experiment in Section 7.2.2.3, we compare our system to the state-of-the-art software X!Tandem. In this experiment, we only compared the performance of one MStream worker and ran X!Tandem in a single-thread mode. In this case, our system could process on average 45 spectra per second, while X!Tandem only processed 17. Furthermore, MStream outperforms X!Tandem upon the input rate of 990 spectra per second with respect to the CPU consumption which is produced by 36 parallel mass spectrometry measurements. Hence, as a central platform for real-time diagnostic platform MStream outperforms the state-of-the-art tool X!Tandem.

7.4 Related Work

In this section, we want to present the related work — the cluster based protein identification X!!Tandem and the indexing approach of Andromeda.

The protein identification software X!Tandem has a parallel cluster version, called X!!Tandem. In X!!Tandem, data is partitioned on available nodes with each node processing batches with the standard X!Tandem. In the end the results are added

to a combined result file [BCC+08]. In contrast to that, our system scales with the amount of data from the experimental side or from the knowledge base, while X!!Tandem scales only from the experimental side.

The protein search engine Andromeda uses an index on masses to reduce the search area during the identification. The index refer to the proteins in the protein knowledge base files [CNM+11]. Nevertheless, the data is in a file, while we structure it and remove redundancy over all the protein knowledge bases uploaded to our system. Finally, we process each spectrum separately and allow the data processing during the measurement whereas both X!!Tandem and Andromeda have to wait until the measurement has completed.

Further related work of the overall platform is described in Chapter 8.

7.5 Conclusion

In this chapter, we combine all components to a platform and evaluated the throughput and the scalability of the whole system.

This motivated us to evolve the data analysis of mass spectrometers to the cloud and break through the standard sequential pipeline. We present a proof of concept for a central mass spectrometry analytic platform MStream to process the data in near-real-time on the SMACK stack.

As part of this evolution, we implemented in collaboration with Bruker Daltonik GmbH an adapter to read the experimental data during the measurement of the mass spectrometer. We further implemented a stream-based validation method based on logistic regression. Additionally, we prepared the protein data and used a column-based index structure to perform fast range queries on the protein data. Finally, we connected the components in the MStream scorer and evaluated the system.

We showed, that the platform analyses in real time the data from multiple mass spectrometers during the experiment, which improves the performance of mass spectrometry analysis pipeline. Especially for use cases such as clinical diagnostics, the real-time processing is desired.

In this chapter, we successfully implemented the proof of concept for a mass spectrometry analytic platform for mass spectrometry analysis use cases such as clinical diagnostics on fast data architecture in near-real-time [ZSB+19a].

8. Related Work

In this chapter, we will analyze related work on the overall solution and not only related work for each component. The target is to analyze cloud solutions for mass spectrometry analysis. Our literature research shows three related platforms to our solution Chorus project, MS-PyCloud and Q-cloud. The criteria for the related work analysis are “Stream”, “Horizontal Scalability”, “Structured Input” and “Structured Output”.

The “Stream” property describes the ability of batch-wise processing of the incoming experimental data, which is needed for real-time processing of the measurement. The “Horizontal Scalability” property describes the ability to add new instances on the cloud to increase the throughput. The “Structured Input” property describes a structured storage of the experimental data. Hence, each single spectrum dataset is accessible and not only the whole experiment file. The last “Structured Output” property describes the storage of the result data. Results are the matches between spectrum and peptide and additional information of the match.

8.1 Chorus

Chorus project is a software suite hosted on Amazon Web Services. It uses the standalone software tools for analyzing the mass spectrometer files and allows sharing files with other researchers. The strategy is different to MStream, since in the Chorus project no adaptation of algorithms is done to perform better performance on a cloud system, which means Chorus does not support streaming data during the measurement. Additionally, the experimental data and protein data are stored in files on the S3 file system and the user management on a relational database management system. The storage of the files on S3 does not allow queries on the data. Chorus provides an additional sharing system over the database, to share and offer result files to other users in the research community. In contrast, MStream uses structured data storage for the data to perform faster analysis on single datasets and not on whole files and allows stream processing [cho19].

8.2 MS-PyCloud

MS-PyCloud, similar to the Chorus project, is a collection of python tools to apply mass spectrometry analysis. Same as in chorus, the data is on a file system and no adaptation of algorithms is done [CZS+18]. MS-PyCloud differs a lot to our MStream platform. In the MS-PyCloud streaming data is not supported and the input is stored in file format, which means unstructured.

8.3 Q-Cloud

In Q-Cloud, the authors integrated OpenMS, a local command line interface tool and transfer the result data into a MySQL database management system. The input is still file based and needs to be uploaded to the system via FTP [COB+18]. In contrast to MStream, no algorithms are adopted to the cloud environment and the input data are complete files in a standard format.

8.4 Summary

We analyzed several systems on a cloud environment. We found out that the cloud is not used properly, since other groups try not to achieve performance improvements using modern techniques of the cloud computing area. Instead, the functionality is in the foreground and the local tools are running on the cloud infrastructure connected as a pipeline. Hence, the cloud platforms are file based and perform as fast as a local machine with the difference of file upload. Hence, a new methodical approach is not provided.

In Table 8.1, we provide a summary of the tools. On the top row, we specify the properties stream, a horizontal scalability, structured input and structured output. In the first column, we specify the different tools. In following, we explain the properties.

Stream property means the ability to send the sample data in parts and not only complete files. Only on MStream platform, the property exists, because other tools use state-of-the-art software, which is not applied to single spectrum data.

Horizontal scalability property means the system can scale on demand over machines in the cluster. Every tool does this, since every computation node can be applied to a user. Only in MStream, the computation is dynamic using a cloud processing engine such as spark.

Structured input property means the input files are separated into datasets and single spectrum can be accessed. This is important for streaming, but also for later analyses to analyze the relations. In MStream the data is stored in a database management system. Each single spectrum can be referenced to the result. In other tools, the incoming data is processed by a local tool, which does not provide this feature.

Structured output property means the output files. In all tools the output is in a database management system and structured, since further analytic is done on the results.

	Stream	Horizontal Scalability	Structured Input	Structured Output
MStream	✓	✓	✓	✓
Chorus	✗	✓	✗	✓
MS-PyCloud	✗	✓	✗	✓
Q-Cloud	✗	✓	✗	✓

Table 8.1: Mass spectrometry analysis cloud platforms overview.

Overall, the main difference to related work is not an integration of tools on a cloud infrastructure, but adopt algorithms and structures on modern cloud computing technologies.

9. Conclusion

In this chapter, we conclude our work. Hence, we summarize the contributions and results of each chapter in detail and refer them to the goals presented in the introduction.

9.1 Level 1: Feasibility

At this level, we identify the needed components of the mass spectrometry data analysis pipeline and defined the challenges of adaptation of the components to a new streaming cloud-based architecture. Overall, four components are identified and each of the adaptation is a contribution on this level.

9.1.1 Identification

RQ 1: What are design choices that are essential for adopting the analysis pipeline to a fast data architecture?

To solve this research question, we analyzed the state-of-the-art protein identification and identified the components to perform the algorithm. In Chapter 3, we showed the applicability of protein search algorithms to a streaming platform by changing the peptide centric search algorithm to a spectrum centric process and apply the pipeline to the fast data architecture. Furthermore, we defined the different structures of the datasets to propose a technology to each component and identify the challenges that need to be solved. Finally, we discussed possible result visualization of complex samples.

9.1.2 Device Data Stream

RQ 2: How to stream the data during the measurement without losing information or functionality or functionality in further processing steps?

We could provide an architecture for the analysis pipeline. The next question is how to stream the data from a mass spectrometer. In collaboration with Bruker Daltonik GmbH, a mass spectrometer manufacturer, we could provide a user friendly software to read out the RAW file during the measurement. In Chapter 4, we present the tool, called MSDataStream, which transforms the RAW data into a textual MGF representation of the spectrum and sends the MS2 spectra text as a batch to the cloud. Finally, we send an MGF textual representation of the mass spectrometry data. Hence, other manufacturers can integrate their converting scripts in our MSDataStream tool. To stream the data directly, firstly, a manufacturer dependent adapter is needed, which collects all MS2 spectra that belongs to the currently measured MS1 spectra. Secondly, filtering functions are needed to reduce the data amount on the client side and increase the quality of the data in the cloud and thirdly, a user friendly graphical interface is needed to raise the acceptance of the users.

9.1.3 Persistence Layer

RQ 3: How to structure the data to increase the performance of the identification process from a streaming mass spectrometer?

At this point, the mass spectrometry data arrives to the cloud, but the protein sequence data needs to be uploaded to the cloud. To reach this goal, we firstly defined a data structure for optimizing the search and secondly, we implement efficient transformation services of the protein data. In Chapter 5, we evaluated the transformation service and improved the service regarding the need of memory and CPU consumption. The data structure is using a column-oriented index structure on the mass and the charge to shrink the search area for each spectrum to a minimum. This is why a column-oriented data structure for protein and peptide sequences is chosen.

9.1.4 FDR calculation

RQ 4: How to avoid target decoy method without compromising result quality of the validation results?

The biggest challenge was the validation, since the target-decoy-approach is only possible if the whole experiment data arrives at once. Hence, a binary classification is needed to perform the validation on a single incoming spectrum. In Chapter 6, we reach this goal by using logistic regression in the identification process, which seems promising to evolve the state-of-the-art target-decoy validation. We provide a workflow for creating training models to perform instant classification in our identification model, answering the RQ4.

9.1.5 Level 1 Summary

Level 1: Is a streaming fast data architecture feasible for mass spectrometry analysis workflow?

As a result, we achieve the first goal of our thesis — we adopt each component to a new streaming cloud-based pipeline and show the feasibility of a cloud stream architecture on a mass spectrometry analysis pipeline. In summary, our contributions answer the first four research questions RQ1, RQ2, RQ3 and RQ4 concerning the question — is a streaming fast data architecture feasible for mass spectrometry analysis workflow?

9.2 Level 2: Performance

At this level, we evaluate the whole system and compare it to the current software. In Chapter 7, all components are combined in our prototype MStream, hosted on multiple machines.

9.2.1 Increasing Number of Devices

RQ 5: How does protein identification scale with an increasing number of connected devices?

To answer the question, we increased the number of incoming spectra and simulate a throughput of up to 28 mass spectrometer devices simultaneously sending data to the cloud. We evaluate different search parameters and conclude a linear scalability regarding the data input. In average, one worker can process ~27 spectra per second in our system.

9.2.2 Increasing Number of Protein Data

RQ 6: How does protein identification scale with an increasing amount of protein data?

We show the influence of the tolerance parameter to the overall query time and recommend to optimize this parameter for specific use cases in real world. Especially for repetitive measurements, which are the usual case in clinical diagnostics of a disease, it is beneficial to use the minimum tolerance. This parameter can be defined based on experience of the measurements for a use case. Hence, the overall performance is acceptable and linear scalability is approved.

9.2.3 Level 2 Summary

Level 2: Does a streaming fast data architecture increase the performance of the workflow?

As a result, we achieve the second goal of our thesis – we evaluate the performance and scalability of the MStream analytic platform for real-time mass spectrometry data analysis. Furthermore, we compared the result with the state-of-the-art protein engine X!Tandem, which performs not as good as MStream on a single machine and X!Tandem is not applicable for a central cloud solution. Unfortunately, we

have to admit, that MStream basic hardware consumption with all the services and technologies is much greater than the local tool X!Tandem. MStream needs at least 30 virtual CPUs to run one worker, while X!tandem, as a local software, needs one CPU. Hence, we evaluate when it makes sense to use MStream regarding CPU consumption. MStream performs better if we scale the throughput over 30 devices, which is reached easily nowadays.

In summary, our contributions answer the last two research questions RQ5 and RQ6 concerning the question – Does a streaming fast data architecture increase the performance of the workflow?

9.3 Thesis Conclusion

As an overall conclusion, the contribution of all chapters evolves the data analysis of mass spectrometers to the cloud and break through the standard sequential pipeline. We present a proof of concept for a central mass spectrometry analytic platform MStream to process the data in near-real-time on the SMACK stack.

As part of this evolution, we implemented the software MSDataStream, which enables the possibility to stream the data from the mass spectrometer during the measurement and reveals new ways of data processing in that area. By preparing the protein data on a column indexed distributed storage the whole system is capable to score the incoming spectrum in real-time. Furthermore, by integrating logistic regression classifier to the validation process, the validation is not the bottleneck of the pipeline. Finally, we implemented a proof of concept MStream, which combines all the components in a cloud-based system. MStream shows the feasibility and the performance improvement in comparison to state-of-the-art software engines.

Hence, the contribution of the thesis is an analytic cloud platform for mass spectrometry data analysis and the challenges that need to be solved to enable the analysis pipeline in near real-time. The platform implements the foundation of real-time mass spectrometry data analysis on a central cloud platform with a good performance.

10. Future Work

In this chapter, we give an overview of open or newly arisen challenges and goals that can be inferred from the results of the thesis. Of course, the future work for this thesis spans across all levels that we worked on in the thesis. Hence, we first review future work for the feasibility of the fast data version of mass spectrometry data analysis pipeline and afterwards new ideas on extending our evaluation with more scenarios. Overall, evaluation that is more technical is needed in order to fine tune the software and hardware components. Although the thesis may give the impression that there are only small optimizations left for improving the applicability of MStream, however, based on our research, complete new applications are possible.

To provide an application for the future, our system will need some new components and improvements. Firstly, the real-time processing enables to generate results, but in the current version of MStream, a manual analysis is needed based on the PSMs in the database. For the diagnostic use case, where it is clear what to search for, a machine learning approach is needed to classify the disease. For example, to find out which diseases are related to the stool sample, the incoming data is directly classified to specific sickness.

Secondly, the current results are stored in a non-relational database, which seems good for the signal data, but not for the relational result datasets. In the future, a polyglot persistence solution needs to be considered to perform analysis on the search results. We propose a graph structured DBMS for the future task, but further experiments are needed. This new aggregation of data is needed to allow to collaborate with other researchers on a visualization.

Thirdly, the explorative analysis of the result is in most cases in the area of research and the researcher need a good interactive visualization of the data [ZSB+17]. We propose one kind of interactive visualization, which improves the quality of the research and further explorative methods are necessary, see Section 3.4. Furthermore, an interface should provide data access to external sources and other tools as export or query access to the data.

Next, another scoring algorithms are needed, which can be applied as quality gate. Since the peptides are queried on same way, a committee of scoring algorithms can filter out bad matches. Nevertheless, since the validation is not 100% accurate, the platform need another classifier for the validation to create another committee. This two quality gates could improve the overall quality of the results and further research is needed.

Further future works are about to implement interfaces to other exchange formats, including different experiment data formats and protein data formats. Additionally, a monitoring tool is important for the platform, to find bottlenecks, improve the usability and quality of the platform or add new functionalities. Furthermore, for production release, parameter tuning is needed and evaluation with other big data technologies and services to find best possible technology stack for MStream.

Finally, the research for the pipeline is done, but a lot of improvement is still possible which tackles new components and new challenges in the future.

Bibliography

[AM03] Ruedi Aebersold and Matthias Mann. Mass Spectrometry-Based Proteomics. *Nature*, Volume 422(6928):198, March 2003. (cited on Page 1, 7, 8, 9, 10, 12, 34, 35, 36, and 61)

[AÇ09] Yanif Ahmad and Uğur Çetintemel. Streaming Applications. *Encyclopedia of Database Systems*, pages 2847–2848, 2009. (cited on Page 24)

[AA98] N. Leigh Anderson and Norman G. Anderson. Proteome and Proteomics: New Technologies, new Concepts, and new Words. *Electrophoresis*, Volume 19(11):1853–1861, August 1998. (cited on Page 1)

[ABW+04] Rolf Apweiler, Amos Bairoch, Cathy H. Wu, Winona C. Barker, Brigitte Boeckmann, Serenella Ferro, Elisabeth Gasteiger, Hongzhan Huang, Rodrigo Lopez, Michele Magrane, Maria J. Martin, Darren A. Natale, Claire O'Donovan, Nicole Redaschi, and Lai-Su L. Yeh. UniProt: The Universal Protein Knowledgebase. *Nucleic Acids Research*, Volume 32:115–119, January 2004. (cited on Page 8, 14, and 52)

[Ast] Astbury Centre for Structural Molecular Biology, The University of Leeds, Alison E. Ashcroft. An Introduction to Mass Spectrometry. http://www.astbury.leeds.ac.uk/facil/MStut/mstutorial.htm. Accessed: 2019-10-16. (cited on Page 1, 7, 14, 34, 35, 36, 49, 52, 55, 56, 61, 74, 79, and 90)

[avd19] avdeo.com. Cassandra 2.0 Architecture. https://advait.files.wordpress.com/2015/06/cassandra_architecture2.png, 2019. Accessed: 2019-10-16. (cited on Page 30)

[BGM+17] Peter Bailis, Edward Gan, Samuel Madden, Deepak Narayanan, Kexin Rong, and Sahaana Suri. MacroBase: Prioritizing Attention in Fast Data. *Proceedings of the ACM International Conference on Management of Data*, pages 541–556, May 2017. (cited on Page 25 and 26)

[BLY+07] Brian M. Balgley, Tom Laudeman, Li Yang, Tao Song, and Cheng S. Lee. Comparative Evaluation of Tandem MS Search Algorithms Using

a Target-Decoy Search Strategy. *Molecular & Cellular Proteomics*, Volume 6(9):1599–1608, May 2007. (cited on Page 85)

[BM11] Shibdas Banerjee and Shyamalava Mazumdar. Electrospray Ionization Mass Spectrometry: A Technique to Access the Information beyond the Molecular Weight of the Analyte. *International Journal of Analytical Chemistry*, Volume 2012, Published online December 2011. (cited on Page 8 and 10)

[BSL+11] Lydia Ashleigh Baumgardner, Avinash Kumar Shanmugam, Henry Lam, Jimmy K. Eng, and Daniel B. Martin. Fast Parallel Tandem Mass Spectral Library Searching using GPU Hardware Acceleration. *Journal of Proteome Research*, Volume 10(6):2882–2888, October 2011. (cited on Page 58)

[BFWSS+15] Pedro Belda-Ferre, James Williamson, Áurea Simón-Soro, Alejandro Artacho, Ole N. Jensen, and Alex Mira. The Human Oral Metaproteome Reveals Potential Biomarkers for Caries Disease. *Proteomics*, Volume 15(20):3497–3507, October 2015. (cited on Page 1, 33, 39, 52, 89, 92, 96, and 98)

[BVJ+09] Dirk Benndorf, Carsten Vogt, Nico Jehmlich, Yvonne Schmidt, Henrik Thomas, Gary Woffendin, Andrej Shevchenko, Hans-Hermann Richnow, and Martin von Bergen. Improving Protein Extraction and Separation Methods for Investigating the Metaproteome of Anaerobic Benzene Communities within Sediments. *Biodegradation*, Volume 20(6):737–750, November 2009. (cited on Page 7 and 8)

[Bio19] Bioinformatics Solutions Inc. False Discovery Rate (FDR) Tutorial. http://www.bioinfor.com/fdr-tutorial/, 2019. Accessed: 2019-10-16. (cited on Page 20)

[BCC+08] Robert D. Bjornson, Nicholas J. Carriero, Christopher Colangelo, Mark Shifman, Kei-Hoi Cheung, Perry L. Miller, and Kenneth Williams. X!!Tandem, an Improved Method for Running X!Tandem in Parallel on Collections of Commodity Computers. *Journal of Proteome Research*, Volume 7(1):293–299, January 2008. (cited on Page 18 and 105)

[BW99] Walter P. Blackstock and Malcolm P. Weir. Proteomics: Quantitative and Physical Mapping of Cellular Proteins. *Trends in Biotechnology*, Volume 17(3):121–127, March 1999. (cited on Page 8)

[Bos16] Mike Bostock. D3-chord. https://github.com/d3/d3-chord, 2016. Accessed: 2019-10-16. (cited on Page 41 and 42)

[Bre01] Leo Breiman. Random Forests. *Machine Learning*, Volume 45(1):5–32, October 2001. (cited on Page 21)

[Bre00] Eric A. Brewer. Towards Robust Distributed Systems. In *Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing*, pages 7–. ACM, January 2000. (cited on Page 22 and 23)

[BKSS17] David Broneske, Veit Köppen, Gunter Saake, and Martin Schäler. Accelerating Multi-Column Selection Predicates in Main-Memory – the Elf Approach. In *IEEE International Conference on Data Engineering*, pages 647 – 658, April 2017. (cited on Page 92)

[CDG+08] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. Bigtable: A Distributed Storage System for Structured Data. *ACM Transactions on Computer Systems*, Volume 26(2):4:1–4:26, June 2008. (cited on Page 29)

[CZS+18] Li Chen, Bai Zhang, Michael Schnaubelt, Punit Shah, Paul Aiyetan, Daniel Chan, Hui Zhang, and Zhen Zhang. MS-PyCloud: An Open-Source, Cloud Computing-Based Pipeline for LC-MS/MS Data Analysis. *Bio-Archive*, May 2018. (cited on Page 108)

[CML14] Min Chen, Shiwen Mao, and Yunhao Liu. Big data: A Survey. *Mobile Networks and Applications*, Volume 19(2):171–209, January 2014. (cited on Page 22)

[COB+18] Cristina Chiva, Roger Olivella, Eva Borras, Guadalupe Espadas, Olga Pastor, Amanda Sole, and Eduard Sabido. QCloud: A Cloud-Based Quality Control System for Mass Spectrometry-Based Proteomics Laboratories. *Public Library of Science*, Volume 13(1):1–14, January 2018. (cited on Page 108)

[cho19] chorusproject.org. About Chorus Project. https://chorusproject.org/pages/about.html, 2019. Accessed: 2019-10-16. (cited on Page 107)

[Cla19] Jim Clark. The Mass Spectrometer. https://www.chemguide.co.uk/analysis/masspec/howitworks.html, 2019. Accessed: 2019-10-16. (cited on Page 1)

[Col96] George Colliat. OLAP, Relational, and Multidimensional Database Systems. *ACM SIGMOD Record*, Volume 25(3):64–69, September 1996. (cited on Page 42)

[col19] colorescience.com. Amino Acids Peptides Proteins. https://www.colorescience.com/learn/files/2018-07/amino-acids-peptide-protein.jpg, 2019. Accessed: 2019-10-16. (cited on Page 9)

[CNM+11] Jürgen Cox, Nadin Neuhauser, Annette Michalski, Richard A. Scheltema, Jesper V. Olsen, and Matthias Mann. Andromeda: A Peptide Search Engine Integrated into the MaxQuant Environment. *Journal of Proteome Research*, Volume 10(4):1794–1805, April 2011. (cited on Page 1, 19, 33, 35, 36, 52, 54, 61, 73, 92, and 105)

[CCL+16] Ya Cui, Xiaowei Chen, Huaxia Luo, Zhen Fan, Jianjun Luo, Shunmin He, Haiyan Yue, Peng Zhang, and Runsheng Chen. BioCircos.js: An Interactive Circos JavaScript Library for Biological Data Visualization on Web Applications. *Bioinformatics*, Volume 32(11):1740–1742, January 2016. (cited on Page 36 and 41)

[Cur16] Edward Curry. The Big Data Value Chain: Definitions, Concepts, and Theoretical Approaches. *New Horizons for a Data-Driven Economy: A Roadmap for Usage and Exploitation of Big Data in Europe*, pages 29–37, 2016. (cited on Page 22 and 24)

[DLB59] Rene De La Briandais. File Searching using Variable Length Keys. In *Western Joint Computer Conference*, pages 295–298. ACM, 1959. (cited on Page 68)

[Del14] Delimited Technologies, 2014. Interactive Chord Diagrams in D3. http://www.delimited.io/blog/2014/11/18/interactive-chord-diagrams-in-d3, 2014. Accessed: 2019-10-16. (cited on Page 42)

[Deu10] Eric W. Deutsch. Mass Spectrometer Output File Format mzML. *Methods in Molecular Biology*, 2010. (cited on Page 35)

[Deu12] Eric W. Deutsch. File Formats Commonly Used in Mass Spectrometry Proteomics. *Molecular & Cellular Proteomics*, Volume 11(12):1612–1621, 2012. (cited on Page 8, 11, 12, 35, 56, and 81)

[DP97] Pedro Domingos and Michael Pazzani. On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. *Machine Learning*, Volume 29(2):103–130, November 1997. (cited on Page 21)

[DAC10] Mark W. Duncan, Ruedi Aebersold, and Richard M. Caprioli. The Pros and Cons of Peptide-Centric Proteomics. *Nature Biotechnology*, Volume 28(7):659–664, July 2010. (cited on Page 12, 16, 36, and 77)

[EKT+12] Martin Eisenacher, Michael Kohl, Michael Turewicz, Markus-Hermann Koch, Julian Uszkoreit, and Christian Stephan. Search and Decoy: the Automatic Identification of Mass Spectra. *Quantitative Methods in Proteomics*, pages 445–488, May 2012. (cited on Page 20 and 36)

[EG09] Joshua Elias and Steven Gygi. Target-Decoy Search Strategy for Mass Spectrometry-Based Proteomics. *Methods in Molecular Biology*, Volume 604:55–71, 2010, Published online December 2009. (cited on Page 19, 20, 77, 78, 84, and 90)

[Ell98] Mourad Elloumi. Comparison of Strings Belonging to the Same Family. *Information Sciences*, Volume 111(1):49 – 63, November 1998. (cited on Page 12)

[EMY94] Jimmy K. Eng, Ashley L. McCormack, and John R. Yates. An Approach to Correlate Tandem Mass Spectral Data of Peptides with Amino Acid Sequences in a Protein Database. *Journal of the American Society for Mass Spectrometry*, Volume 5(11):976–989, November 1994. (cited on Page 14, 34, 36, 77, and 84)

[ECL+12] Alison R. Erickson, Brandi L. Cantarel, Regina Lamendella, Youssef Darzi, Emmanuel F. Mongodin, Chongle Pan, Manesh Shah, Jonas Halfvarson, Curt Tysk, Bernard Henrissat, Jeroen Raes, Nathan C. Verberkmoes, Claire M. Fraser, Robert L. Hettich, and Janet K. Jansson. Integrated Metagenomics/Metaproteomics Reveals Human Host-Microbiota Signatures of Crohn's Disease. *PLOS ONE*, Volume 7(11):1–14, November 2012. (cited on Page 1, 33, 39, 52, 89, 92, 96, and 98)

[ER16] Raul Estrada and Isaac Ruiz. *Big Data SMACK*. Apress Media LLC, 2016. (cited on Page 28, 30, 31, 38, and 39)

[Est16] Raúl Estrada. *Fast Data Processing Systems with SMACK Stack*. Packt Publishing, 2016. (cited on Page 25, 38, 52, 78, 79, and 89)

[Eva11] Dave Evans. The Internet of Things: How the Next Evolution of the Internet is Changing Everything. *Cisco IBSG white paper*, pages 1–11, April 2011. (cited on Page 24 and 90)

[Fen99] David Fenyö. The Biopolymer Markup Language. *Bioinformatics*, Volume 15(4):339–340, April 1999. (cited on Page 81)

[Gar13] Nishant Garg. *Apache Kafka*. Packt Publishing, 2013. (cited on Page 30, 31, and 56)

[Gar14] Nishant Garg. *HBase Essentials*. Packt Publishing, 2014. (cited on Page 29)

[GSV+15] Giulia Gonnelli, Michiel Stock, Jan Verwaeren, Davy Maddelein, Bernard De Baets, Lennart Martens, and Sven Degroeve. A Decoy-Free Approach to the Identification of Peptides. *Journal of Proteome Research*, Volume 14(4):1792–1798, February 2015. (cited on Page 38, 77, 80, 84, and 90)

[GNK12] Viktor Granholm, William Stafford Noble, and Lukas Käll. A Cross-Validation Scheme for Machine Learning Algorithms in Shotgun Proteomics. *BioMed Central Bioinformatics*, Volume 13(S3):1471–2105, November 2012. (cited on Page 84)

[GPRL+16] Johannes Griss, Yasset Perez-Riverol, Steve Lewis, David L. Tabb, José A. Dianes, Noemi del Toro, Marc Rurik, Mathias Walzer, Oliver Kohlbacher, Henning Hermjakob, Rui Wang, and Juan Antonio Vizcaíno. Recognizing Millions of Consistently Unidentified Spectra Across Hundreds of Shotgun Proteomics Datasets. *Nature Methods*, Volume 13:651 EP –, June 2016. (cited on Page 54)

[GBMP13] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions. *Future Generation Computer Systems*, Volume 29(7):1645–1660, September 2013. (cited on Page 36)

[Hai88] Max Hailperin. Load balancing for massively-parallel soft real-time systems. *Proceedings of 2nd Symposium on the Frontiers of Massively Parallel Computation*, pages 159–163, October 1988. (cited on Page 18)

[Han07] David J. Hand. Principles of data mining. *Drug Safety*, Volume 30(7):621–622, 2007. (cited on Page 20)

[Has15] Hashem, Ibrahim Abaker Targio and Yaqoob, Ibrar and Anuar, Nor Badrul and Mokhtar, Salimah and Gani, Abdullah and Khan, Samee Ullah. The Rise of "Big Data" on Cloud Computing: Review and Open Research Issues. *Information Systems*, Volume 47(Supplement C):98 – 115, 2015. (cited on Page 29)

[HR07] Jeffrey Heer and George Robertson. Animated Transitions in Statistical Data Graphics. *IEEE Transactions on Visualization and Computer Graphics*, Volume 13(6):1240–1247, November 2007. (cited on Page 46)

[HBS+93] William. J. Henzel, Todd M. Billeci, John T. Stults, Susan C. Wong, Christopher Grimley, and Colin Watanabe. Identifying Proteins from Two-Dimensional Gels by Molecular Mass Searching of Peptide Fragments in Protein Sequence Databases. *Proceedings of the National Academy of Sciences of the United States of America*, Volume 90(11), June 1993. (cited on Page 1, 9, 33, 35, 36, 52, and 61)

[HKRB15] Robert Heyer, Fabian Kohrs, Udo Reichl, and Dirk Benndorf. Metaproteomics of Complex Microbial Communities in Biogas Plants. *Microbial Technology*, Volume 8, April 2015. (cited on Page 1, 7, 8, 9, 34, 35, 36, 39, 49, 52, 61, 74, and 90)

[HSS+19] Robert Heyer, Kay Schallert, Corina Siewert, Fabian Kohrs, Julia Greve, Irena Maus, Johanna Klang, Michael Klocke, Monika Heiermann, Marcus Hoffmann, Sebastian Püttker, Magdalena Calusinska, Roman Zoun, Gunter Saake, Dirk Benndorf, and Udo Reichl. Metaproteome Analysis Reveals that Syntrophy, Competition, and Phage-Host Interaction Shape Microbial Communities in Biogas Plants. *Microbiome*, Volume 7(1):69, April 2019. (cited on Page 1, 9, 12, 19, and 20)

[HSZ+17] Robert Heyer, Kay Schallert, Roman Zoun, Beatrice Becher, Gunter Saake, and Dirk Benndorf. Challenges and Perspectives of Metaproteomic Data Analysis. *Journal of Biotechnology*, Volume 261:24 – 36, November 2017. (cited on Page 1, 11, 12, 17, 19, 20, 33, 35, 36, 49, 52, 61, 74, 77, 78, and 80)

[HKZ+11] Benjamin Hindman, Andy Konwinski, Matei Zaharia, Ali Ghodsi, Anthony D. Joseph, Randy Katz, Scott Shenker, and Ion Stoica. Mesos: A Platform for Fine-grained Resource Sharing in the Data Center. In *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation*, pages 295–308. USENIX Association, March 2011. (cited on Page 27, 28, and 39)

[Hol06] Danny Holten. Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data. *IEEE Transactions on Visualization and Computer Graphics*, Volume 12(5):741–748, September 2006. (cited on Page 41 and 42)

[HYS+86] Donald F. Hunt, John R. Yates, Jeffrey Shabanowitz, Scott Winston, and Charles R. Hauer. Protein Sequencing by Tandem Mass Spectrometry. *Proceedings of the National Academy of Sciences*, Volume 83(17):6233–6237, September 1986. (cited on Page 9 and 10)

[JZS+18] Atin Janki, Roman Zoun, Kay Schallert, Rohith Ravindran, David Broneske, Wolfram Fenske, Robert Heyer, Dirk Benndorf, and Gunter Saake. Connecting X! Tandem to a Database Management System. In *GI-Workshop Grundlagen von Datenbanken*, GvDB, pages 77–82, May 2018. (cited on Page 18)

[JGO+14] Zhanlin Ji, Ivan Ganchev, Máirtín O'Droma, Li Zhao, and Xueji Zhang. A Cloud-Based Car Parking Middleware for IoT-Based Smart Cities: Design and Implementation. *Sensors*, Volume 14(12):22372–22393, December 2014. (cited on Page 58)

[KKKG14] Karthik Kambatla, Giorgos Kollias, Vipin Kumar, and Ananth Grama. Trends in Big Data Analytics. *Journal of Parallel and Distributed Computing*, Volume 74(7):2561–2573, July 2014. (cited on Page 22)

[KMF+17] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Li. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems 30*, pages 3146–3154. Curran Associates, Inc., December 2017. (cited on Page 21)

[KPB+17] Andreas Kipf, Varun Pandey, Jan Boettcher, Lucas Braun, Thomas Neumann, and Alfons Kemper. Analytics on Fast Data: Main-Memory Database Systems versus Modern Streaming Systems. *EDBT*, pages 49–60, March 2017. (cited on Page 24, 36, 38, 39, 52, 56, and 79)

[KPB+19] Andreas Kipf, Varun Pandey, Jan Böttcher, Lucas Braun, Thomas Neumann, and Alfons Kemper. Scalable Analytics on Fast Data. *ACM Transactions on Database Systems*, Volume 44(1):1:1–1:35, January 2019. (cited on Page 24, 38, 39, 52, and 79)

[KNR19] Oliver Kohlbacher, Sven Nahnsen, and Knut Reinert. Computational Proteomics and Metabolomics. https://slideplayer.com/slide/5682644/, 2019. Accessed: 2019-10-16. (cited on Page 9, 14, and 15)

[KCS+17] Raymond F. Kokaly, Roger N. Clark, Gregg A. Swayze, K. Eric Livo, Todd M. Hoefen, Neil C. Pearson, Richard A. Wise, William M. Benzel, Heather A. Lowers, Rhonda L. Driscoll, and Anna J. Klein. USGS Spectral Library Version 7. Technical report, U.S. Geological Survey Data Series 1035, April 2017. (cited on Page 54)

[kom19] komabiotech.co.kr. http://www.komabiotech.co.kr/www/product/pImages/sdspage1.jpg, 2019. Accessed: 2019-10-16. (cited on Page 8)

[Kra19] Dennis Krannich. HEM auf RPI mit Docker und Hypriot. https://blog.krannich.de/wp-content/uploads/2018/12/docker-was-ist-ein-container-600x520.png, 2019. Accessed: 2019-10-16. (cited on Page 27)

[KNR11] Jay Kreps, Neha Narkhede, and Jun Rao. Kafka: A Distributed Messaging System for Log Processing. *Proceedings of 6th International Workshop on Networking Meets Databases*, June 2011. (cited on Page 30 and 31)

[KSB+09] Martin Krzywinski, Jacqueline Schein, İnanç Birol, Joseph Connors, Randy Gascoyne, Doug Horsman, Steven J. Jones, and Marco A. Marra. Circos: An Information Aesthetic for Comparative Genomics. *Genome Research*, Volume 19(9):1639–1645, September 2009. (cited on Page 41 and 42)

[LM10] Avinash Lakshman and Prashant Malik. Cassandra: A Decentralized Structured Storage System. *The ACM Special Interest Group on Operating Systems*, Volume 44(2):35–40, April 2010. (cited on Page 29, 39, 61, and 92)

[LLP+12] Wang Lam, Lu Liu, Sts Prasad, Anand Rajaraman, Zoheb Vacheri, and AnHai Doan. Muppet: MapReduce-style Processing of Fast Data. *Proceedings of the VLDB Endowment*, Volume 5(12):1814–1825, August 2012. (cited on Page 25)

[Las19] Jacek Laskowski. The Internals of Apache Spark 2.4.2. https://legacy.gitbook.com/book/jaceklaskowski/mastering-apache-spark/details, 2019. Accessed: 2019-10-16. (cited on Page 29 and 39)

[LKN13] Viktor Leis, Alfons Kemper, and Thomas Neumann. The Adaptive Radix Tree: ARTful Indexing for Main-memory Databases. In *Proceedings of the IEEE International Conference on Data Engineering*, ICDE '13, pages 38–49, April 2013. (cited on Page 68)

[LQD14] Yahui Liu, Hong Qing, and Yulin Deng. Biomarkers in Alzheimer's Disease Analysis by Mass Spectrometry-Based Proteomics. *International Journal of Molecular Sciences*, Volume 15(5):7865–7882, May 2014. (cited on Page 1, 52, and 89)

[LBK+17] Markus Lubeck, Scarlet Beck, Heiner Koch, Stephanie Kaspar-Schoenefeld, Niels Goedecke, Oliver Raether, Nicole Drechsler, Michael Krause, Florian Meier, Jürgen Cox, and Matthias Mann. PASEF™ on a timsTOF Pro Defines new Performance Standards for Shotgun Proteomics with Dramatic Improvements in MS/MS Data Acquisition Rates and Sensitivity. Technical report, Bruker Daltonics, September 2017. (cited on Page 8, 11, 34, 35, 36, 52, 55, 58, 61, and 90)

[MRML07] Pierre-Alain Maron, Lionel Ranjard, Christophe Mougel, and Philippe Lemanceau. Metaproteomics: A New Approach for Studying Functional Microbial Ecology. *Microbial Ecology*, Volume 53:486—493, April 2007. (cited on Page 7 and 8)

[Mat16] Matrix Science. Data file format. http://www.matrixscience.com/help/data_file_help.html, 2016. Accessed: 2019-10-16. (cited on Page 8, 11, 12, 35, 56, and 90)

[MTS+04] W. Hayes McDonald, David L. Tabb, Rovshan G. Sadygov, Michael J. MacCoss, John Venable, Johannes Graumann, Jeff R. Johnson, Daniel Cociorva, and John R. Yates III. MS1, MS2, and SQT—Three Unified, Compact, and Easily Parsed File Formats for the Storage of Shotgun Proteomic Spectra and Identifications. *Rapid Communications in Mass Spectrometry*, Volume 18(18):2162–2168, September 2004. (cited on Page 8, 11, 35, and 56)

[MBY+16] Xiangrui Meng, Joseph Bradley, Burak Yavuz, Evan Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, DB Tsai, Manish Amde, Sean Owen, Doris Xin, Reynold Xin, Michael J. Franklin, Reza Zadeh, Matei Zaharia, and Ameet Talwalkar. MLlib: Machine Learning in Apache Spark. *Journal of Machine Learning Research*, Volume 17(1):1235–1241, January 2016. (cited on Page 20, 22, 39, 78, 81, and 85)

[Mer14] Dirk Merkel. Docker: Lightweight Linux Containers for Consistent Development and Deployment. *Linux Journal*, Volume 2014(239), March 2014. (cited on Page 27)

[MFT+13] Renato Millioni, Cinzia Franchin, Paolo Tessari, Rita Polati, Daniela Cecconi, and Giorgio Arrigoni. Pros and Cons of Peptide Isolectric Focusing in Shotgun Proteomics. *Journal of Chromatography*, Volume 1293:1 – 9, June 2013. (cited on Page 14, 16, 17, 36, 66, and 92)

[MY04] Steven A. Morris and Gary G. Yen. Crossmaps: Visualization of Overlapping Relationships in Collections of Journal Papers. *Proceedings*

of the National Academy of Sciences of the United States of America, Volume 101(Suppl 1):5291–5296, April 2004. (cited on Page 41)

[mur19] murphyandson.co.uk. https://www.murphyandson.co.uk/wp-content/uploads/2015/04/slider3-1-700x400.jpg, 2019. Accessed: 2019-10-16. (cited on Page 8)

[MBH+15] Thilo Muth, Alexander Behne, Robert Heyer, Fabian Kohrs, Dirk Benndorf, Marcus Hoffmann, Miro Lehtevä, Udo Reichl, Lennart Martens, and Erdmann Rapp. The MetaProteomeAnalyzer: A Powerful Open-Source Software Suite for Metaproteomics Data Analysis and Interpretation. *Journal of Proteome Research*, Volume 14(3):1557–1565, February 2015. (cited on Page 18, 40, 41, and 73)

[MBR+13] Thilo Muth, Dirk Benndorf, Udo Reichl, Erdmann Rapp, and Lennart Martens. Searching for a Needle in a Stack of Needles: Challenges in Metaproteomics Data Analysis. *Molecular BioSystems*, Volume 9(4):578–585, April 2013. (cited on Page 1, 14, 33, 35, 36, 37, 52, 54, 59, 61, 77, 79, 84, and 90)

[NSP17] Neha Narkhede, Gwen Shapira, and Todd Palino. *Kafka: The Definitive Guide Real-Time Data and Stream Processing at Scale*. O'Reilly Media, Inc., 1st edition, 2017. (cited on Page 30, 31, 39, and 55)

[Nas07] Nasser M. Nasrabadi. Pattern Recognition and Machine Learning. *Journal of Electronic Imaging*, Volume 16(4):049901, October 2007. (cited on Page 22)

[Nat02] National Center for Biotechnology Information. FASTA format. https://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&PAGE_TYPE=BlastDocs&DOC_TYPE=BlastHelp, 2002. Accessed: 2019-10-16. (cited on Page 8, 12, 53, 61, 62, 63, 66, 75, and 90)

[Nee15] Nishant Neeraj. *Mastering Apache Cassandra*. Packt Publishing, 2nd edition, 2015. (cited on Page 29 and 30)

[Neg15] Christopher Negus. *Docker Containers*. Addison-Wesley Professional, 2nd edition, 2015. (cited on Page 27)

[NKH+17] Orthodoxia Nicolaou, Andreas Kousios, Andreas Hadjisavvas, Bernard Lauwerys, Kleitos Sokratous, and Kyriacos Kyriacou. Biomarkers of Systemic Lupus Erythematosus Identified using Mass Spectrometry-Based Proteomics: A Systematic Review. *Journal of Cellular and Molecular Medicine*, Volume 21(5):993–1012, May 2017. (cited on Page 1, 52, and 89)

[Pam16] Juan Martín Pampliega. Towards an Architecture for Real-Time Event Processing. *II Simposio Argentino de Grandes Datos*, pages 22–35, September 2016. (cited on Page 28)

[PLI02] Joanne Peng, Kuk Lida Lee, and Gary M. Ingersoll. An Introduction to Logistic Regression Analysis and Reporting. *Journal of Educational Research*, Volume 96:3–14, September 2002. (cited on Page 21, 78, 81, and 82)

[PPCC99] David N. Perkins, Darryl J.C. Pappin, David Creasy, and John Cottrell. Probability-Based Protein Identification by Searching Sequence Databases using Mass Spectrometry Data. *Electrophoresis*, Volume 20:3551–3567, December 1999. (cited on Page 1, 17, 36, 77, and 85)

[PF17] Bernardo A. Petriz and Octávio L. Franco. Metaproteomics as a Complementary Approach to Gut Microbiota in Health and Disease. *Frontiers in Chemistry*, Volume 5:4, July 2017. (cited on Page 1, 7, 9, 33, 39, 52, 89, 96, and 98)

[PHTN12] Brian Pratt, J. Jeffry Howbert, Natalie I. Tasman, and Erik J. Nilsson. MR-Tandem: Parallel X!Tandem using Hadoop MapReduce on Amazon Web Services. *Bioinformatics*, Volume 28(1):136–137, January 2012. (cited on Page 52 and 58)

[PW78] S. James Press and Sandra Wilson. Choosing between Logistic Regression and Discriminant Analysis. *Journal of the American Statistical Association*, Volume 73(364):699–705, December 1978. (cited on Page 21 and 78)

[PJW+14] DaRue A. Prieto, Donald J. Johann, Bih-Rong Wei, Xiaoying Ye, King C. Chan, Dwight V. Nissley, R. Mark Simpson, Deborah E. Citrin, Crystal L. Mackall, W. Marston Linehan, and Josip Blonder. Mass Spectrometry in Cancer Biomarker Research: A Case for Immunodepletion of Abundant Blood-Derived Proteins from Clinical Tissue Specimens. *Biomarkers in Medicine*, Volume 8(2):269–286, February 2014. (cited on Page 1, 52, and 89)

[Pri08] Dan Pritchett. BASE: An ACID Alternative. *Queue*, Volume 6(3):48–55, May 2008. (cited on Page 22)

[Qui86] J. Ross Quinlan. Induction of Decision Trees. *Machine Learning*, Volume 1(1):81–106, March 1986. (cited on Page 21)

[res17] researchandmarkets.com. Global Mass Spectrometry Market Size, Market Share, Application Analysis, Regional Outlook, Growth Trends, Key Players, Competitive Strategies and Forecasts, 2015 to 2025. https://www.researchandmarkets.com/reports/4313373/global-mass-spectrometry-market-size-market, May 2017. Accessed: 2019-10-16. (cited on Page 1, 7, and 9)

[RR03] Craig Robertson and C. Beavis Ronald. A Method for Reducing the Time Required to Match Protein Sequences with Tandem Mass Spectra. *Rapid Communications in Mass Spectrometry*, Volume 17:2310–2316, October 2003. (cited on Page 1, 14, 17, 18, 33, 35, 36, 52, 53, 54, 59, 61, 77, 84, and 90)

[RGSP07] Jesse Rodriguez, Nitin Gupta, Richard D Smith, and Pavel A Pevzner. Does Trypsin Cut Before Proline? *Journal of Proteome Research*, Volume 7(01):300–305, December 2007. (cited on Page 9, 12, 13, 53, 61, 63, 66, 78, and 90)

[RRK+90] Dennis W. Ruck, Steven K. Rogers, Matthew Kabrisky, Mark E. Oxley, and Bruce W. Suter. The Multilayer Perceptron as an Approximation to a Bayes Optimal Discriminant Function. *IEEE Transactions on Neural Networks*, Volume 1(4):296–298, December 1990. (cited on Page 21)

[SSH10] Gunter Saake, Kai-Uwe Sattler, and Andreas Heuer. *Datenbanken - Konzepte und Sprachen, 4. Auflage.* MITP, 2010. (cited on Page 29 and 74)

[SF13] Pramod J. Sadalage and Martin Fowler. *NoSQL distilled: a brief guide to the emerging world of polyglot persistence.* Addison-Wesley, 2013. (cited on Page 29)

[SH15] Sanju Saha and Santochi Halder. Does Interactive Visualization Affect Motor Cognition and Learning Outcomes of Students? *Conference to Review Research on Science Technology and Mathematics Education*, Volume 6:110 – 116, January 2015. (cited on Page 44)

[SBD+08] Andreas Schlüter, Thomas Bekel, Naryttza N. Diaz, Michael Dondrup, Rudolf Eichenlaub, Karl-Heinz Gartemann, Irene Krahn, Lutz Krause, Holger Krömeke, Olaf Kruse, Jan H. Mussgnug, Heiko Neuweger, Karsten Niehaus, Alfred Pühler, Kai J. Runte, Rafael Szczepanowski, Andreas Tauch, Alexandra Tilker, Prisca Viehöver, and Alexander Goesmann. The Metagenome of a Biogas-Producing Microbial Community of a Production-Scale Biogas Plant Fermenter Analysed by the 454-Pyrosequencing Technology. *Journal of Biotechnology*, Volume 136(1):77 – 90, August 2008. (cited on Page 8)

[SZBL10] Joerg Seidler, Nico Zinn, Martin E. Boehm, and Wolf D. Lehmann. De Novo Sequencing of Peptides by MS/MS. *Proteomics*, Volume 10(4):634–649, February 2010. (cited on Page 54)

[SOAA97] Masami Shishibori, Mikiya Okuno, Kazuaki Ando, and Jun-Ichi Aoe. An Efficient Compression Method for Patricia Tries. In *IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, volume Volume 1, pages 415–420, October 1997. (cited on Page 68)

[SKRC10] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. The Hadoop Distributed File System. In *IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, pages 1 –10, May 2010. (cited on Page 29)

[Sir15] Enrico Siragusa. *Approximate String Matching for High-Throughput Sequencing*. PhD thesis, Freie Universität Berlin, 2015. (cited on Page 74)

[SÇZ05] Michael Stonebraker, Uğur Çetintemel, and Stan Zdonik. The 8 Requirements of Real-Time Stream Processing. *ACM SIGMOD Record*, Volume 34(4):42–47, December 2005. (cited on Page 26)

[SAB+08] Structural Genomics Consortium, Architecture et Fonction des Macromolécules Biologiques, Berkeley Structural Genomics Center, China Structural Genomics Consortium, Integrated Center for Structure and Function Innovation, Israel Structural Proteomics Center, Joint Center for Structural Genomics, Midwest Center for Structural Genomics, New York Structural GenomiX Research Center for Structural Genomics, Northeast Structural Genomics Consortium, Oxford Protein Production Facility, Protein Sample Production Facility, Max Delbrück Center for Molecular Medicine, RIKEN Structural Genomics/Proteomics Initiative, and SPINE2-Complexes. Protein Production and Purification. *Nature Methods*, Volume 5(2):135–146, February 2008. (cited on Page 1)

[Tab15] David L. Tabb. The SEQUEST Family Tree. *Journal of The American Society for Mass Spectrometry*, Volume 26(11):1814–1819, November 2015. (cited on Page 1, 33, 35, 36, and 61)

[tec19] techdifferences.com. Difference Between Linear and Logistic Regression. https://techdifferences.com/difference-between-linear-and-logistic-regression.html, 2019. Accessed: 2019-10-16. (cited on Page 21)

[The19a] The Apache Software Foundation. Apache Cassandra. http://cassandra.apache.org/, 2019. Accessed: 2019-10-16. (cited on Page 29)

[The19b] The Apache Software Foundation. Apache Mesos. http://mesos.apache.org/, 2019. Accessed: 2019-10-16. (cited on Page 28)

[The17] The Proteomics Standards Initiative. mzML 1.1.0 Specification. http://www.psidev.info/mzml_1_0_0%20, 2017. Accessed: 2019-10-16. (cited on Page 35)

[the19] themocracy.com. http://themocracy.com/wp-content/uploads/2016/07/Microbial-Community.jpg, 2019. Accessed: 2019-10-16. (cited on Page 8)

[TLM+00] Bernd Thiede, Stephanie Lamer, Jens Mattow, Frank Siejak, Christiane Dimmler, Thomas Rudel, and Peter R. Jungblut. Analysis of Missed Cleavage Sites, Tryptophan Oxidation and N-Terminal Pyroglutamylation after in-Gel Tryptic Digestion. *Rapid Communications in Mass Spectrometry*, Volume 14(6):496–502, March 2000. (cited on Page 12, 13, and 66)

[Tur18] Matt Turck. Great Power, Great Responsibility: The 2018 Big Data & AI Landscape. https://mattturck.com/bigdata2018/, 2018. Accessed: 2019-10-16. (cited on Page 26)

[UAK18] Saeed Ullah, M. Daud Awan, and M. Sikander Hayat Khiyal. Big Data in Cloud Computing: A Resource Management Perspective. *Scientific Programming*, Volume 2018:1–17, May 2018. (cited on Page 27)

[uni19] unipept.ugent.be. Case study: Taxonomic Analysis of a Tryptic Peptide. https://unipept.ugent.be/, 2019. Accessed: 2019-10-16. (cited on Page 12)

[VRS+08] Nathan C. Verberkmoes, Alison L. Russell, Manesh Shah, Adam Godzik, Magnus Rosenquist, Jonas Halfvarson, Mark G. Lefsrud, Juha Apajalahti, Curt Tysk, Robert L. Hettich, and Janet K. Jansson. Shotgun metaproteomics of the human distal gut microbiota. *Multidisciplinary Journal of Microbial Ecology*, Volume 3(2):179–189, October 2008. (cited on Page 8)

[VMJ16] Ankush Verma, Ashik Hussain Mansuri, and Neelesh Jain. Big data management processing with Hadoop MapReduce and spark technology: A comparison. In *IEEE Symposium on Colossal Data Analysis and Networking*, pages 1–4, Indore, Madhya Pradesh, India, March 2016. IEEE. (cited on Page 28)

[WM83] Ingrid Wagner and Hans Musso. New Naturally Occurring Amino Acids. *Angewandte Chemie International Edition in English*, Volume 22(11):816–828, November 1983. (cited on Page 9)

[Wam15] Dean Wampler. Fast Data: Big Data Evolved. *LIGHTBEND INC.*, 2015. (cited on Page 24, 25, 36, 38, 52, 55, 78, and 79)

[Wam16] Dean Wampler. *Fast Data Architectures for Streaming Applications*. O'Reilly Media, first edition, September 2016. (cited on Page 22, 24, 25, 26, 55, 56, 82, 89, and 90)

[WLZZ14] Huajin Wang, Jianhui Li, Haiming Zhang, and Yuanchun Zhou. Benchmarking Replication and Consistency Strategies in Cloud Serving Databases: HBase and Cassandra. In *Big Data Benchmarks, Performance Optimization, and Emerging Hardware*, volume Volume 8807, pages 71–82. Springer International Publishing, March 2014. (cited on Page 29)

[Whi09] Tom White. *Hadoop: The Definitive Guide*. O'Reilly Media, Inc., 1st edition, May 2009. (cited on Page 29)

[Whi05] David Whitford. *Proteins: Structure and Function*. John Wiley and Sons Inc., 2 edition, March 2005. (cited on Page 8, 12, 13, 16, 53, 61, and 66)

[wU19] Ch.ko123 wikimedia User. Apache Kafka. https://commons.wikimedia.org/w/index.php?curid=59871096, 2019. Accessed: 2019-10-16. (cited on Page 31)

[WAWH05] M.R. Wilkins, R.D. Appel, K.L. Williams, and D.F. Hochstrasser. *Proteome Research: Concepts, Technology and Application*. Springer, 2 edition, 2005. (cited on Page 7)

[XYF+17] Mingming Xiao, Junjun Yang, Yuxin Feng, Yan Zhu, Xin Chai, and Yuefei Wang. Metaproteomic strategies and applications for gut microbial research. *Applied Microbiology and Biotechnology*, Volume 101(8):3077–3088, April 2017. (cited on Page 1, 33, 39, 52, 89, 92, 96, and 98)

[YHG+16] Ibrar Yaqoob, Ibrahim Abaker Targio Hashem, Abdullah Gani, Salimah Mokhtar, Ejaz Ahmed, Nor Badrul Anuar, and Athanasios V. Vasilakos. Big Data. *International Journal of Information Management*, Volume 36(6):1231–1247, December 2016. (cited on Page 22)

[ZCD+12] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy Mccauley, M. Franklin, Scott Shenker, and Ion Stoica. Fast and Interactive Analytics over Hadoop Data with Spark. *Usenix Login*, Volume 37(4):45–51, August 2012. (cited on Page 28, 56, and 90)

[ZLY+15] Ji Zhang, Yanna Liang, Peter Yau, Rohit Pandey, and Satya Harpalani. A Aetaproteomic Approach for Identifying Proteins in Anaerobic Bioreactors Converting Coal to Methane. *International Journal of Coal Geology*, Volume 146:91–103, July 2015. (cited on Page 1 and 9)

[ZWLS15] Zhigao Zheng, Ping Wang, Jing Liu, and Shengli Sun. Real-Time Big Data Processing Framework: Challenges and Solutions. *Applied Mathematics & Information Sciences*, Volume 9(6), January 2015. (cited on Page 25)

[Zou18] Roman Zoun. Internet of Metaproteomics. In *IEEE 34th International Conference on Data Engineering*, pages 1714–1718, April 2018. (cited on Page 33, 36, 37, 49, 51, 52, and 54)

[ZDS+18] Roman Zoun, Gabriel C. Durand, Kay Schallert, Apoorva Patrikar, David Broneske, Wolfram Fenske, Robert Heyer, Dirk Benndorf, and Gunter Saake. Protein Identification as a Suitable Application for Fast Data Architecture. In *Database and Expert Systems Applications*, pages 168 – 178. IEEE, September 2018. (cited on Page 37, 51, 52, 53, 54, and 61)

[ZSB+19a] Roman Zoun, Kay Schallert, David Broneske, Sören Falkenberg, Robert Heyer, Sabine Wehnert, Sven Brehmer, Dirk Benndorf, and

Gunter Saake. MStream: Proof of Concept of an Analytic Cloud Platform for Near-Real-Time Diagnostics using Mass Spectrometry Data. Technical Report 002-2019, Otto-von-Guericke-University Magdeburg, August 2019. (cited on Page 37, 89, 90, 91, 94, 95, and 105)

[ZSB+19b] Roman Zoun, Kay Schallert, David Broneske, Wolfram Fenske, Marcus Pinnecke, Robert Heyer, Sven Brehmer, Dirk Benndorf, and Gunter Saake. MSDataStream - Connecting a Bruker Mass Spectrometer to the Internet. In *Datenbanksysteme für Business, Technologie und Web*, pages 507 – 510. Gesellschaft für Informatik, March 2019. (cited on Page 11, 36, 37, 51, 55, 56, 59, 61, 84, 90, and 91)

[ZSB+17] Roman Zoun, Kay Schallert, David Broneske, Robert Heyer, Dirk Benndorf, and Gunter Saake. Interactive Chord Visualization for Metaproteomics. In *Database and Expert Systems Applications*, pages 79–83, August 2017. (cited on Page 36, 39, 49, 56, and 115)

[ZSB+19] Roman Zoun, Kay Schallert, David Broneske, Ivayla Trifonova, Xiao Chen, Robert Heyer, Dirk Benndorf, and Gunter Saake. Efficient Transformation of Protein Sequence Databases to Columnar Index Schema. In *Database and Expert Systems Applications*, pages 67–72. Springer International Publishing, August 2019. (cited on Page 37, 61, 62, 63, 65, 68, 70, 75, and 90)

[ZSJ+18] Roman Zoun, Kay Schallert, Atin Janki, Rohith Ravindran, Gabriel C. Durand, Wolfram Fenske, David Broneske, Robert Heyer, Dirk Benndorf, and Gunter Saake. Streaming FDR Calculation for Protein Identication. In *Advances in Databases and Information Systems*, pages 80 – 87, September 2018. (cited on Page 22, 38, 78, 80, 81, 82, 84, 90, and 91)