

University of Magdeburg
School of Computer Science



Master's Thesis

Cassiopeia - Andromeda-based Protein Identification on Fast Data Architecture

Author:

Anton Atanasov

June 17, 2019

Advisors:

Prof. Dr. rer. nat. habil. Gunter Saake
Institute of Technical and Business Information Systems (ITI)

M.Sc. Roman Zoun
Institute of Technical and Business Information Systems (ITI)

Atanasov, Anton:

Cassiopeia - Andromeda-based Protein Identification on Fast Data Architecture
Master's Thesis, University of Magdeburg, 2019.

Abstract

The constant evolution of the field of metaproteomics creates vast quantities of data. The ample amounts of new data provide the opportunity to study organic samples by analyzing the contained proteins thoroughly. There are already existing software tools used to identify peptide sequences, such as X!Tandem, Mascot, and Andromeda, but there is a lack of tools able to work with experimental data in real-time. We present a new protein identification software - Cassiopeia. It uses the same evaluation mechanism used in the open-sourced Max Quant product - Andromeda, and produces similar results. Our evaluations show how the scoring algorithm is used as a cross-validation mechanism, integrating it into an existing cloud environment, able to process mass spectrometry data as a stream. Cassiopeia is a part of an already existing protein search engine - MStream, which uses fast data architecture and is deployed onto existing SMACK stack environment.

Contents

List of Figures	xi
List of Tables	xiv
List of Code Listings	xv
1 Introduction	1
2 Background	5
2.1 Mass spectrometry	5
2.1.1 Metaproteomics and mass spectrometry	5
2.1.1.1 Proteomics and Metaproteomics	6
2.1.1.2 Mass spectrometry	8
2.1.1.3 Data formats used in mass spectrometry	8
2.2 Protein identification	13
2.2.1 Protein identification software - state of the art	13
2.3 Andromeda	16
2.4 Fast data	17
2.4.1 History of Big Data	19
2.4.2 Batch-Mode architecture	21
2.4.3 Lambda architecture	21
2.4.4 Streaming Technologies	23
2.5 SMACK stack	26
2.5.1 SMACK definition	27
2.5.1.1 Apache Spark	27
2.5.1.2 Apache Mesos	29
2.5.1.3 Akka	29
2.5.1.4 Cassandra	30
2.5.1.5 Kafka	31
2.6 Summary	32
3 Concept	33
3.1 Existing concept provided by MStream	33
3.1.1 Cassiopeia role in MStream	35
3.2 Cassiopeia concept	37
3.2.1 Cassiopeia validation	37
3.2.2 Cassiopeia Scoring	38
3.2.3 Cassiopeia Integration	42

3.3	Summary	43
4	Implementation	47
4.1	Cassiopeia technology stack	47
4.2	Cassiopeia architecture implementation	50
4.2.1	Creating peptide in-memory knowledge base	50
4.2.2	Scoring algorithm	52
4.2.2.1	Scoring for the validation	52
4.2.2.2	Scoring implementation	53
4.2.2.3	Preparation for the true probability calculation	57
4.3	Summary	57
5	Evaluation	59
5.1	Experiment set up	59
5.2	Validation Experiments	61
5.2.1	Experiment 1: Validation on sets without any modifications	61
5.2.2	Experiment 2: Validation for data sets with modifications	63
5.2.3	Experiment 3: Evaluation of performance on data sets with different sizes	69
5.3	Cloud evaluation	75
5.4	Summary	77
6	Discussion	79
6.1	Local environment	79
6.1.1	Validation results	79
6.2	Scaling	80
6.3	Cloud result analysis	81
6.4	Summary	82
7	Related Work	83
7.1	Q-Cloud	83
7.2	MS-PyCloud	83
7.3	Chorus	83
7.4	Summary	84
8	Conclusion	85
9	Future Work	87
	Bibliography	89

List of Figures

1.1	Visual representation of amino acids, peptides and proteins. Each circle color represents different amino acid. Chains of amino acids construct a peptide and groups of peptides form a protein.	2
1.2	Flow of a mass spectrometer analysis [RZS18]. The steps are as follows: 1. Biological preparation; 2. Measurement by a mass-spectrometer; 3. Conversion of data; 4. Identification of sample; 5. Validation of results	3
2.1	Mass spectrum result. The x-axis represents the mass to charge ratio and the y-axis is the relative intensity of the measured signal strength from the sample. High intensity values indicate possible presence of corresponding amino acid sequences. [Ree88]	9
2.2	The hydrolysis of protein (red) by the effects of water (blue) [RLLK06]. The properties of water manage to split the peptide into two at a specific amino acid site. By Thomas Shafee - Own work, CC BY 4.0, https://commons.wikimedia.org/w/index.php?curid=42390219 . . .	12
2.3	Process of matching peptides. An assumption is built from theoretical protein composition and the theoretical values are compared with the spectra measured by a mass spectrometer. Theoretical values are calculated in ideal condition, while in the real world the presence of noise and contaminants is to be expected.	14
2.4	Steps used in building the protein sequence database used as knowledge base in Andromeda [CNM ⁺ 11]. There are three distinct steps performed when building the index, a) sorts the proteins by name performs the initial splitting of proteins into peptides by applying experiment specific cleavage rules and writes the results onto storage while keeping an index file. Step b) sorts the peptide sequences by their strings and stores those into index files. The last step c) sorts the peptides by mass, while again keeping an index file showing where the data is allocated on the storage.	18
2.5	CAP Theorem	21
2.6	Hadoop Batch architecture	22

2.7	Lambda architecture workflow [BH]. Label 1 shows new data entering the system, 2 contains the batch layer with the master data set in it, 3 shows the serving layer managing incoming queries, 4 denotes the speed layer, showing intermediate progress for queries in real-time, 5 describes any incoming queries, which might target either the batch or serving layers.	23
2.8	Fast data (streaming) architecture	24
2.9	SMACK (Apache Spark, Mesos, Akkaa, Cassandra and Kakfa) [ER16a] overview. The graphic shows the roles and connections between each component from the technological stack.	27
2.10	Apache spark cluster managing three worker nodes	28
2.11	Apache Mesos architecture [Foua]: master daemon (only one at a time) interacts and manages slave agent daemons	30
2.12	Apache Kafka typical scenario [ER16b]. In the picture the connections between heterogeneous systems to a common message publisher (Kafka) is presented. External systems can read data published in topics of interest.	31
3.1	Architecture of MStream [RZS18], analytic platform for real-time diagnostic of mass spectrometry data	34
3.2	Concept architecture of MStream using multiple peptide identification algorithms. The spectra will be provided as a stream, theoretical peptides will be retrieved from a distributed peptide sequence database and given as an input to the search functions. After both tools have evaluated the potential matches the results will be compared to check the validity of the scores.	36
3.3	Andromeda stand-alone components. We can see the input batches for proteins in the form of FASTA, which create the sequence database and store it on disk. The spectra are iterated sequentially and fed into the search function, using the Andromeda scoring algorithm. Commonly considered theoretical peptides are stored into a cache. The end output of a PSM object is an extension by us to enable validation. One thing to note is the presence of a filter, controlling the resulting output.	39
3.4	Components of Cassiopeia. The structure is based on the one present in Figure 3.3. Instead of storing the peptide sequence database onto hard disk, we use an in-memory database. The spectra can be either APL or MGF formats, and the settings given to the search function are the same as Andromeda's. During evaluation by Andromeda's algorithm, peptides can also be stored into a cache structure. The output is returned as a PSM object without filters.	40

3.5	Cassiopeia scoring components. The present elements are a subset of those seen in Figure 3.4, where only the required for the MStream integration are kept. The input, consisting out of a spectrum and peptide, get passed along with application-specific settings to the scoring function. The evaluation uses Andromeda's algorithm to produce a PSM object	41
3.6	Influence of adding Cassiopeia into MStream's SMACK environment. There are now two Peptide Spectrum Match components, one indicating Cassiopeia and the other - the default X!Tandem implementation or any other integrated peptide identification software. The validation is also marked in red as the scores of multiple searching tools can be used for cross-validation.	43
3.7	Flow chart for MStream with integrated Cassiopeia, showing possible evaluation configuration	44
4.1	The IScorer interface. It produces a PSM object, which encompasses the input object and adds an attribute for the result score. This is the interface used to integrate Cassiopeia into MStream.	48
4.2	Usage of IScorer scoring functionality for MStream and for the validation of Cassiopeia against Andromeda	51
4.3	Formula used to calculate the probability of a match. It is the same one used in Andromeda [CNM ⁺ 11]. The scoring involves an optimization of the number of highest intensity peaks in a Spectrum. The peaks are considered in ranges of 100 Dalton units in the m/z interval and over the inclusion of modification-specific neutral losses.	54
4.4	Cloud Scorer	55
4.5	Scoring result class. The array have the size of how much peptide are given as input, in the cloud variant this array will be the size of one.	56
5.1	Correlation between Andromeda and Cassiopeia scores performed on 10% of the Ecoli_04_RD2_01_1275 data set. The graphs contains 40 random subsets out of the total 63 evaluated peptides, but the missing records behave in the same way. The y-axis displays the score given as output, and the x-axis displays the peptide sequence. Both Cassiopeia and Andromeda have produced the exact same result, therefore, the bins are the same height.	62
5.2	Evaluation for the whole Ecoli_04_RD2_01_1275 without modifications. The x-axis shows the index of the peptide, the y-axis is for the score value. The blue points indicate the average for each individual peptide. The gray line is present for peptides, which were considered as a potential match for more than one peak and had different score values, based on the spectrum considered. The gray line shows the range of the found scores, where the lowest point shows the minimum, the highest - the maximum and the blue dot displays the average of the scores assigned to the peptide.	64

5.3	Comparison between the maximum scores of identified peptides from Cassiopeia and Andromeda with Methionine and Cysteine as variable modifications. The x-axis displays the 109 distinct peptides shared by both Cassiopeia and Andromeda, while the y-axis indicates the maximum score achieved by each individual peptide sequence.	66
5.4	Comparison on the complete data set without modifications, displaying how many times distinct peptides have been considered as a match in Cassiopeia and in Andromeda. The y-axis displays how many times a sequence has been evaluated as a potential match. The blue line shows the result given by Cassiopeia, while the orange one is for Andromeda. The x-axis shows the index of the 109 shared by Cassiopeia and Andromeda peptides.	67
5.5	Average of all scores present for each distinct peptide, occurring both in Andromeda and Cassiopeia. The x-axis shows the indexes of the 109 shared by both Cassiopeia and Andromeda result evaluations. For each peptide the average is computed from the sum of all different scores for each individual peptide and divided by the number of times it has been evaluated as a potential match for more than one spectra. The resulting value is assigned to the y-axis, representing the score achieved.	68
5.6	Search performed on 10% of the protein sequence data set UPSP_Nov2017 and 10% of the Ecoli_04_RD2_01_1275 peak data set. The first 10% of both files was used to evaluate the duration of each phase on a small set of spectra with a small sequence data base used as a reference.	70
5.7	Search performed on fifty percent of the data.	71
5.8	Search performed using the complete data sets representing the spectra from the mass spectrometry experiment and the UPSP_Nov2017 protein knowledge base.	72
5.9	Time spent for the building of the peptide sequence database out of a FASTA file, containing protein sequence information. There are three different sizes of the initial UPSP_Nov2017 considered.	73
5.10	Time spent by Cassiopeia (blue) and Andromeda (orange) on evaluation peptide-spectra matches on different data set sizes. The x-axis shows the data set size and the x-axis displays the time taken in milliseconds.	73
5.11	The total time spent for an evaluation of different data set sizes for Ecoli_04_RD2_01_1275 without modifications. The time is represented in milliseconds and is the sum of pre-processing and search times.	74

5.12 The average of the sum of scores for the top 200 MStream PSMs reduced to 78 unique peptide sequences, which are present also in Cassiopeia's output for the cloud evaluation. The scores are ordered by descending for results produced by X!Tandem scoring and matched with their Cassiopeia counterparts. Both X!Tandem and Cassiopeia posses different scoring scales, where the top X!Tandem score achieved was 26 and the top Cassiopeia was 121. 77

List of Tables

2.1	List of the proteinogenic amino acids, the building blocks of protein and peptide sequences.	7
2.2	Batch-mode systems [Wam19]	21
2.3	Streaming systems	26
4.1	Components roles and implementations between MStream and Cassiopeia's validation	52
5.1	Experiment parameters for the experiments performed. The same parameters are used for Andromeda, Cassiopeia and MStream. Variable modifications are used only on Ecoli_02_RD2_01_2199 for one experiment, otherwise none were used	60
5.2	Variable modifications used in Andromeda, Cassiopeia and MStream	61
5.3	Number of times the same peptide has been identified as a match . .	63
5.4	Results of peptide search for Ecoli_04_RD2_01_1275 with Cysteine and Methionine as variable modifications.	65
5.5	Mismatched peptides between the maximum scores produced by Cassiopeia and Andromeda. We have 4 places in total where differences are present. Two of them produce scores with small difference, while the other 2 cases have a significant difference in favor of Cassiopeia. .	65
5.6	Identification time in milliseconds for 10% of the complete proteins and peaks data sets	69
5.7	Identification time in milliseconds for 50% of the complete proteins and peaks data sets	70
5.8	Identification time in milliseconds for the complete proteins and peaks data sets	71
5.9	Total time elapsed for the evaluation of the three different data set pair sizes.	74
5.10	MStream integration comparison. For the processing of 57,000 individual spectra Cassiopeia takes five times longer by evaluating ten times more peptide-spectrum-match objects.	75

5.11 Overlapping peptides identified by both protein identification algorithms	76
5.12 Overlapping peptides identified by both protein identification algorithms where the whole Cassiopeia data set is used	76
5.13 Overlapping peptides identified by both protein identification algorithms where the whole MStream data set is used	76

List of Code Listings

2.1	MGF format example, used to encode a single spectrum or also called peak. In this example the peptide sequence is present. The sequence can be obtained from the analysis of the "PEPMASS" and the mass-to-charge ratios represented by the triples of numbers.	10
2.2	APL format example used to encode a single spectrum. It contains the mass-to-charge ration, the charge and tuples of mass-intensity values. The fragmentation type used to obtain this peak is also present under "fragmentation"	11
2.3	FASTA format example. In this listing there two proteins described, a header is given after the '>' symbol, describing the protein, followed by its composition, represented by an amino-acid chained expressed as a string. Typical usage of this format is for a protein sequence database	13
4.1	Calculate delta of spectrum mass	53

1. Introduction

The study of proteins in living organisms provides an exciting opportunity in learning how cells behave and interact with their surroundings in different environmental conditions [PF17]. The presence of proteins gives insight into possible organisms behavior. Proteins can be used as features, indicating the presence of viruses or bacteria. They can help distinguish the presence of diseases so that appropriate actions can be taken. The biological field concerned with the study of proteins is called *proteomics* [AM03]. Proteomics concerns itself with the composition of one single organism. Distinguishing between many organisms is subject of *metaproteomics*. Both research fields are done with the help of *mass spectrometers* [WS07]. Mass spectrometers attempt to analyze the structure of an organism by subjecting the organic compound to an analysis. Organisms are composed of proteins, which are amino acid chains [SSM58]. Amino acids are chemical compounds consisting of different chemical elements and follow the rules based on chemistry to bond with other amino acids [Nus81]. A protein can be divided into sub-proteins, so-called peptides. A visual representation of the whole structure may be seen in Figure 1.1.

Mass spectrometers measure biological samples and produce mass spectra as output [AM03]. In Figure 1.2 the workflow of a metaproteomic experiment is displayed. *Step 1* shows the biological preparation of an organic sample. The step includes the acquisition of an organic sample, purification, separation, and possible other pre-processing steps. The sample is then fed into a mass spectrometer, which measures the data and records the measurements and stores the result into storage, represented by *step 2*. Performing an experiment measurement typically lasts around two hours, depending on the sample size and the environment variables, which need to be configured. [HSZ⁺17]. The transformation of results into electronic format is the digitization of the experiment outcome. In *step 3* the raw data from the experiment is converted into a suitable format, usable by a protein identification software. The experiment results can be then used by a protein identification process [NPPCC99]. The protein identification generally is composed of a protein knowledge database, which describes known protein sequences and additional meta information. A software tool analyzes the data and attempts to identify known chemical compounds, represented by step 4. The data generated by the mass spectrometer is then analyzed

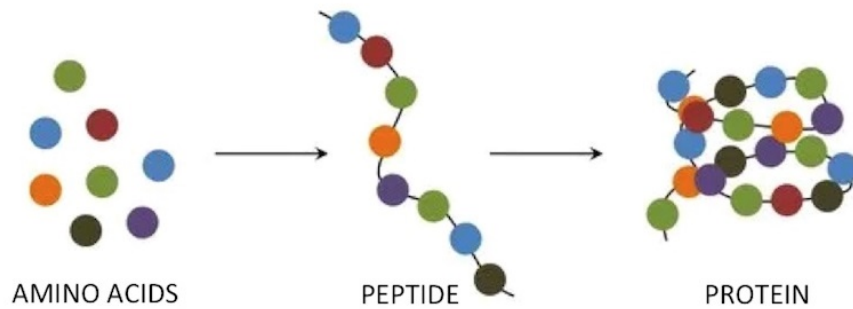


Figure 1.1: Visual representation of amino acids, peptides and proteins. Each circle color represents different amino acid. Chains of amino acids construct a peptide and groups of peptides form a protein.

and compared against the protein database. The best matches can be validated, and in the end, conclusions can be drawn. There are a couple of known protein identification software tools present, such as Mascot [NPPCC99], Andromeda [CNM+11], X!Tandem [BCC+08] and more. The final step 5 is the validation of the correctness of the results, in order to guarantee that the evaluation can be trusted.

The state of the art protein processing tools rely on having the whole data set on which analysis will be performed beforehand. For the analysis to be thorough, analytics need the whole volume of experimental data present. Most of the current software tools are not explicitly set up to work with parts of the experiment result and to produce valid results. A single mass-spectrometry experiment can also last a while, based on the time needed to prepare for the measuring step. This limits analysts as they need to wait until all measurement information is ready for the next stage. After the experimental data is gathered, it is transformed into a format, since each algorithm differs in implementations. This transformation step usually lasts another hour (step 3 from Figure 1.2). The transformed experimental data is also substantial in size, directly correlating to the experiment, and the processing time ends up taking another couple of hours [HSZ+17]. In the end state of the art tools can begin working with data after at least a couple of hours have gone by for the measurements, digitalization, and conversion (steps 1, 2 and 3 in Figure 1.2)

Some of the most known existing protein identification software tools, such as Andromeda, X!Tandem and Mascot ([CNM+11], [BCC+08], [CB03]), work with the transformed data in batches. The encapsulation of experimental data into batches is done so information can be processed in chunks and to offer eventual parallelization of work. Although efficient for areas, where the data may be already available beforehand, this approach fails to accommodate use cases, where the results need to be analyzed on the fly. An example situation would be a clinical trial, where breakthroughs may offer significant immediate gains of knowledge, and it is most beneficial to know about them as soon as possible. The real-time analysis presents a problem, which requires the design and implementation of a new pipeline, able to handle real-time data, instead of waiting for the whole measurement to complete.

Our paper focuses on an already existing solution by Zoun *et. al* ([RZS18]) in handling streams of data on the fly, where experiment data is transformed and ana-

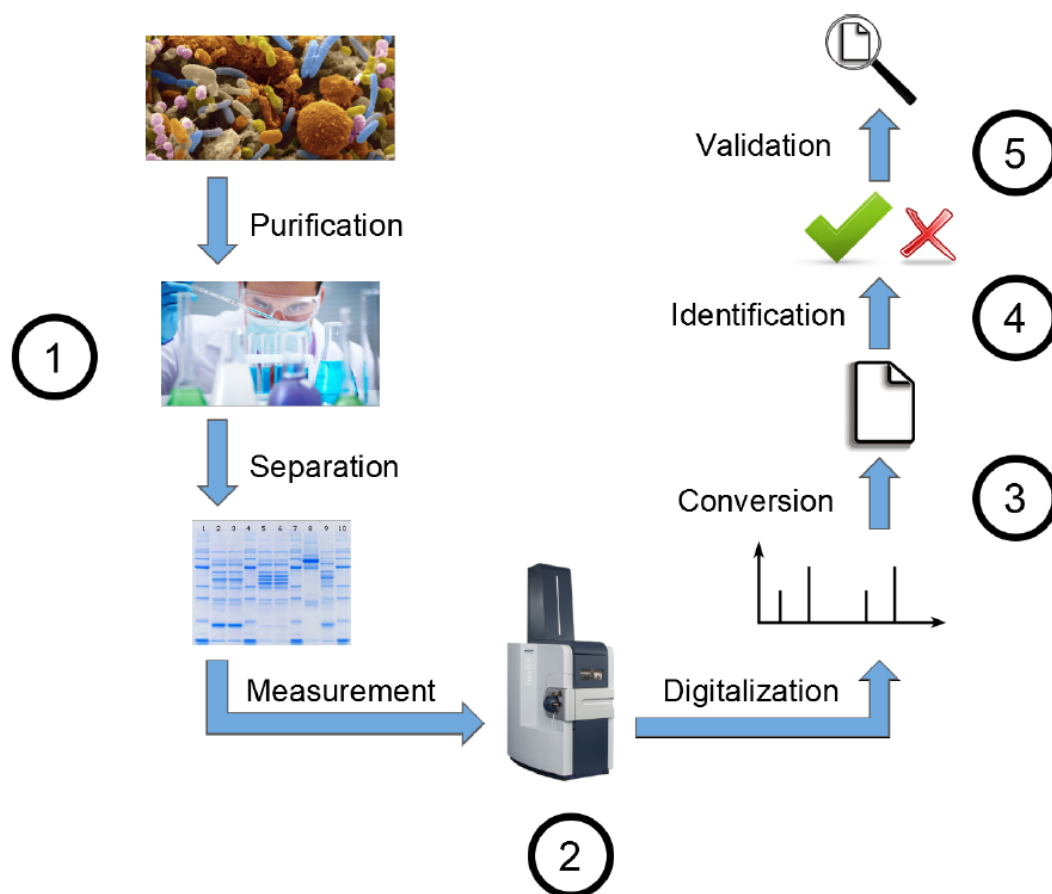


Figure 1.2: Flow of a mass spectrometer analysis [RZS18]. The steps are as follows: 1. Biological preparation; 2. Measurement by a mass-spectrometer; 3. Conversion of data; 4. Identification of sample; 5. Validation of results

lyzed in near real-time. The existing solution uses X!Tandem ([BCC⁺08]) as protein identification software and is deployed on an environment using SMACK ([All13]) environment. If we want to improve the analyzing process, only one search engine may not be enough. The system will benefit from an additional scoring approach, which can be used as a cross-validation mechanism. We focus on adding another open-source protein identification software to the identification step: Andromeda [CNM⁺11]. We intend to complement the existing solution by adding a second scoring mechanism. The usage of a second protein search tool will be used to cross-validate any results, thus increasing the confidence in the results and validating the software performance and reliability.

The original Andromeda algorithm has a validated, tested, and peer-reviewed performance. Therefore the scoring method will be enough. Should we be able to recreate the results and be able to analyze the data on the fly with a streaming architecture using fast data [Wam19] we will enable the analysis of data in real-time. Additionally, a new software tool could be used as a measure of cross-validation for protein identification. Achieving high scores on guesses obtained by multiple independent tools will boost the confidence rating of a match.

Goal of this Thesis

In this paper, we will present our new protein identification software: *Cassiopeia*. Based on the open-sourced stand-alone Andromeda, developed by MaxQuant, we attempt to integrate our new solution into a MStream environment. The MStream platform has previously been developed by Zoun et. al in [RZS18], as a proof of concept for the near-real-time analysis of tandem mass spectrometry data. The main goal is to provide answers to the following questions:

- How do the results of Cassiopeia compare against Andromeda and is the new tool supplying correct evaluations?
- Are Cassiopeia scoring results an improvement on the overall results produced by MStream?
- Is the performance of Cassiopeia better than MStream state of the art?

Structure of the Thesis

This work is structured as follows: in [Chapter 2](#) we give all the needed background knowledge to understand the biological terms, the software architecture concepts, and history of the evolution of the protein identification software. In [Chapter 3](#) we will present our concept for Cassiopeia, the goals and the target results. In [Chapter 4](#) we will present the concrete technologies used to implement the concept. After that we will present our achieved results and will discuss what knowledge we gained in [Chapter 6](#). Finally, we will sum up our work by covering the viability of our software, the correctness, and its efficiency when compared to other protein identification software, while taking into account the environment in which the tool is deployed.

2. Background

In our work, we present a new protein identification software. Its purpose is to analyze metaproteomic data measured by mass spectrometers. In this chapter, we cover the technological concepts of the underlying biological knowledge discussed in this paper. We also cover the inner workings of the Andromeda software and how it evaluates potential peptide matches.

2.1 Mass spectrometry

Protein analysis by MS has been gaining increasing popularity with the years. Previously the best results were obtained from small sets of proteins in an isolated environment. With the years passing by and people using already gained knowledge, the process has improved. Having access to better hardware and technology processes further sped up the knowledge gain in this field. It is still proving to be a difficult task because the structure of proteomes has significant and highly variable complexity. The number of proteins outnumbers the number of proteins in a species proteome vastly when compared with the corresponding genome. This is made possible by the ability for different protein splicing, presence of sequence polymorphisms and in general, the ability to accumulate proteins in various manners.

In the previous section, we covered the data-working process. In this chapter, we will focus on the *what* the data is representing. First, we will go over the biological concepts regarding metaproteomics, and then we will briefly describe how we measure them with the mass spectrometer.

2.1.1 Metaproteomics and mass spectrometry

A protein is a chemical compound composed of a sequence of amino acids [Sar92]. An amino acid is an organic molecule with known properties; they are the main building blocks of organic structure. There are roughly 500 naturally occurring amino acids [WM83] but in the term of biochemistry we consider only 22 different types of amino acids. We do this because organisms are always built by the combination of these 22 amino acids, a table of them can be seen in Table 2.1. Each amino acid

is represented by a different letter abbreviation and has a different mass property. In a computational environment, an amino acid chain is represented as a string, composed out of those letters. A protein is any sequence composed of more than 30 amino-acids; otherwise, the chain is regarded as a peptide. An example of a peptide would be the string "AEFVEVTK". This peptide will have a mass influenced by the sum of the masses of all present amino acids. Based on the mass-spectrometry experiment, some of the organic compounds may be modified. On top of that the modifications can be two types:

- fixed modifications: they represent a specific amino acid, that is modified in the experiment environment. Usually, a modification is added instead of one of the 22 available, where the default mass property has a modified value. Any protein strings, containing the modification will use its altered mass instead of the default one.
- variable modifications: they again represent some amino acids that are modified in the context of the mass-spectrometry analysis. They do not directly replace the default mass property but instead result in an amino acid chain that has two different masses: one with the default mass and one with the modified value. For example, if we have a variable modification on Alanine, the peptide "AEFVEVTK" will have two different masses.

Additionally, the same peptides may be arbitrarily present in many proteins. Those can be treated as biomarkers to identify the presence of already known proteins [HBS⁺93]. Identified peptide sequences are used to build protein sequence databases, which are later used to find the presence of known organic mixtures in an unknown sample. The scientific area, which is tasked with the finding of peptides, is named proteomics.

2.1.1.1 Proteomics and Metaproteomics

The biological field of *proteomics* is concerned with the study and identification of proteins expressions [AA98]. Its goal is to correctly classify protein structures and changes in their compositions occurring naturally over time or caused by environmental effects. The main benefit of the proteomic study is the ability to identify known proteins [BW99]. Those proteins can be the building blocks of a bacteria [VRS⁺08], be a biomarker for the presence of an illness [PA06] or a byproduct of a chemical process [WGS⁺09].

Metaproteomics is an extension of the field of proteomics [MRML07]. The added difference is the more significant scope of research, where it could perform experiments on samples of whole microbial communities. This is especially useful for areas like biogas plants [HKRB15] or research on human gut bacteria composition [VRS⁺08]. The field has been steadily gaining pace in terms of applicability in recent years, thanks to the technological advances in mass spectrometry instrumentation, creating more and more data and enabling more in-depth studies.

Amino acid	Letter abbreviation	Average mass (Dalton)
Alanine	A	89.09404
Cysteine	C	121.15404
Aspartic acid	D	133.10384
Glutamic acid	E	147.13074
Phenylalanine	F	165.19184
Glycine	G	75.06714
Histidine	H	155.15634
Isoleucine	I	131.17464
Lysine	K	146.18934
Leucine	L	131.17464
Methionine	M	149.20784
Asparagine	N	132.11904
Pyrrolysine	O	255.31
Proline	P	75.06714
Glutamine	Q	146.14594
Arginine	R	174.20274
Threonine	T	119.12034
Selenocysteine	U	168.053
Valine	V	117.14784
Tryptophan	W	204.22844
Tyrosine	Y	181.19124

Table 2.1: List of the proteinogenic amino acids, the building blocks of protein and peptide sequences.

2.1.1.2 Mass spectrometry

Mass spectrometry (*MS*) experiments are carried out on the organic samples. The samples are turned into their gas aggregate states and then ionized [AM03]. The effects are observed with a *mass spectrometer*. A mass spectrometer has an ion source and a mass analyzer which measures the mass-to-charge ratio (m/z) of the ionized sample. The count of present ions is taken as the final result, expressed in the sample's molecule mass and molecule formula. An example can be seen in Figure 2.1. The goal of the mass analyzer is to measure the sample with regards to sensitivity, accuracy of the mass measurement, and high resolution. These properties provide information in the form of ion mass spectra, derived from the peptide fragments (tandem mass or MS/MS spectra) [AM03] [PM00] [AG01] [MHP01]. Tandem mass spectrometers split the protein sample into peptide fragment ions and store the resulting fragment ion spectra [AG01]. The observed fragment ion spectra are generated by a process named collision-induced dissociation. During it, the peptide ion is fragmented and then analyzed in an isolated environment. This second stage is used to recognize familiar bio-markers to identify known or unknown peptides. The main benefit of tandem mass spectrometry is the extensive sequence information obtained on the whole protein chain. The means of extracting information by splitting is also straightforward when considering the overhead for splitting proteins and purifying the resulting samples [HYS+86]. The sample can be inspected without requirements of it to be purified and in a highly homogeneous state. This represents step 1 from Figure 1.2.

2.1.1.3 Data formats used in mass spectrometry

Measuring the data is the first step of protein identification. The measurements are then digitized and stored into files for future usage. The information provides links between the proteins and their coding genes which ties together the physiology and genetics of a microorganism. Information can be encoded in different file formats, either proprietary or open formats [Deu12]. An open data format allows the usage of the data by any tool, suited for said format, whereas proprietary ones are typically used for specific models of mass spectrometers or in a company. Open formats can be further separated as:

- officially recognized standard - gone through official peer reviews and multiple refinements
- *de facto* standard - without official recognition but commonly used by many protein identification tools
- other standards

A typical open official format in which the information is stored is the Mascot Generic File (MGF) format [MTS+04], an example can be seen in Listing 2.1.

The data encoded in MGF format describes individual spectra. These spectra, or also called peaks, can be used as indicators for the presence of peptides. From the observation of peptides, we can deduce possible original proteins present. To

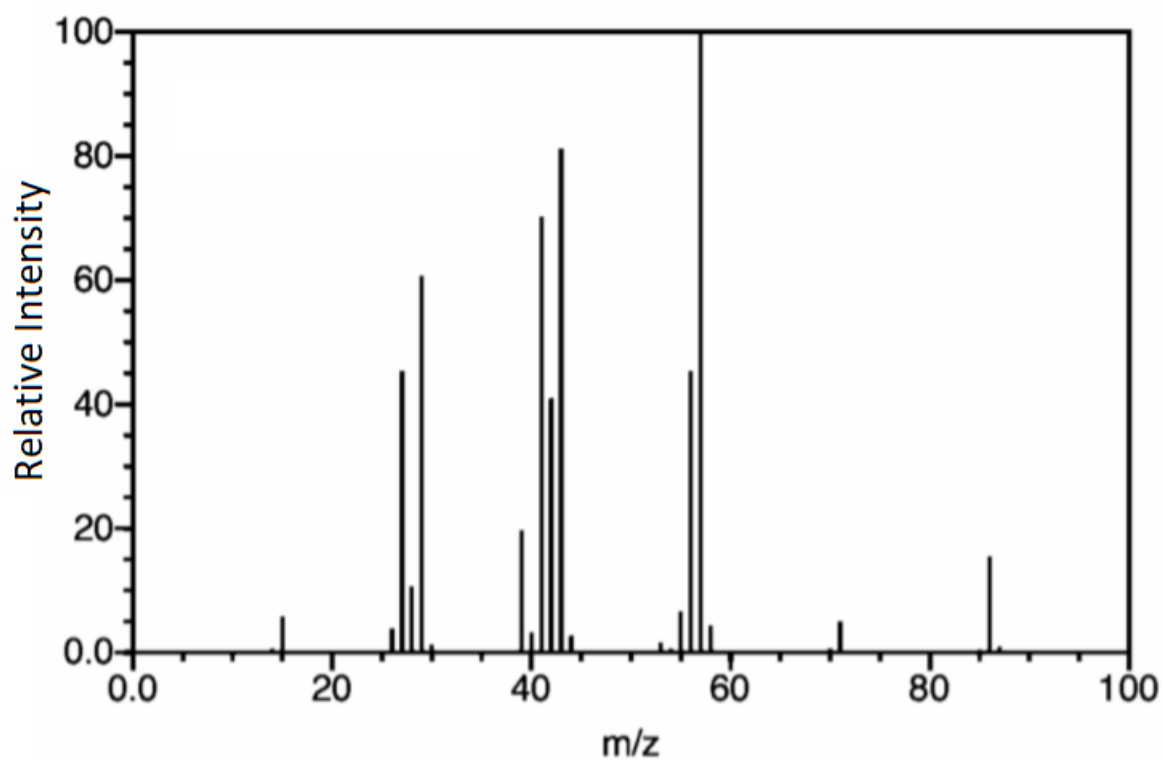


Figure 2.1: Mass spectrum result. The x-axis represents the mass to charge ratio and the y-axis is the relative intensity of the measured signal strength from the sample. High intensity values indicate possible presence of corresponding amino acid sequences. [Ree88]

```
1 BEGIN IONS
2 TITLE=Spectrum000101 AEFVEVTK +2 y- and b-series
3 PEPMASS=308.16757 2000000
4 CHARGE=3+
5 RTINSECONDS=100
6 SCANS=25000
7 36.52588 100000 1+
8 74.06009 100000 1+
9 101.04717 100000 1+
10 124.58392 100000 1+
11 174.11813 100000 1+
12 201.08703 100000 1+
13 224.11559 100000 1+
14 238.63943 100000 1+
15 288.17363 100000 1+
16 338.17109 100000 1+
17 348.15544 100000 1+
18 388.69493 100000 1+
19 447.22386 100000 1+
20 576.26645 100000 1+
21 675.33486 100000 1+
22 END IONS
```

Listing 2.1: MGF format example, used to encode a single spectrum or also called peak. In this example the peptide sequence is present. The sequence can be obtained from the analysis of the "PEPMASS" and the mass-to-charge ratios represented by the triples of numbers.

check if the experimental data indicates a peptide or protein presence, we construct theoretical spectra. We construct them by using a peptide sequence database. This database contains the amino-acid sequence information of proteins. The sequence database can be encoded in different formats, but the general concept is to represent a protein structure as a string of amino acids elements. The same information can also be encoded in other formats, for example APL (Listing 2.2). The information present in the MGF format is also present in the APL file, although some differences are present. For example, APL states the type of fragmentation used for deriving this peptide. A common fragmentation used is HCD fragmentation [LQ13], which is for short for high-energy collision dissociation. The fragmentation is used to split proteins into peptides and can be used as an identification method, since different fragmentation types follow strict patterns in producing amino acid sub-sequences.

Mass also has intensity, where different intensity levels signal the presence of ions located in the mass spectrometer [Dom06]. Low intensities can be assumed as noises of the signal. For example, in Listing 2.2, we have some tuples of masses with intensities. Higher intensities mean higher probability for meaningful particles. Lower intensities have a more significant probability of being noise. We look into the mz value is the monoisotopic weight of the ions in a peptide. The Cassandra database contains an ordered by mass list of known peptides. We expect to be supplied with a peptide which has a similar mass of the peak, where the closer - the better.

For example, we can consider the *FASTA* format example of a two protein entries seen in Listing 2.3. We can observe two distinct organic compounds, and to be

```
1 peaklist start
2 header=Precursor_13428
3 mz=401.7435005663026
4 charge=2
5 fragmentation=HCD
6 224.105 9.0
7 236.774 9.0
8 310.843 9.0
9 343.664 9.0
10 562.914 9.0
11 570.607 9.0
12 850.545 9.0
13 929.528 9.0
14 1385.651 9.0
15 peaklist end
```

Listing 2.2: APL format example used to encode a single spectrum. It contains the mass-to-charge ration, the charge and tuples of mass-intensity values. The fragmentation type used to obtain this peak is also present under "fragmentation"

precise: a header, containing some protein-specific information and an amino-acid chain sequence. We can use this protein string to obtain a set of peptides, usable by the mass spectrometer. We split the proteins with rules, consequences of the presence of an enzyme or other experimental factors. For example, when a particular amino acid is present in the protein sequence, we can utilize a rule to split the chain at the amino acid [SCH01]. After having split the protein chain, the matching can begin, where real-world results are compared with theoretical values. The picture seen in Figure 2.3 describes this process.

In the upper part, there is visualized the real-world preparation and analysis of organic samples. Before an organic compound is subjected to a metaproteomic analysis, it is purified, so only proteins remain. Having done that, they are subjected to enzymes and other environmental properties, specific to the experiment, so the protein amino acid chains break down into peptides. This is done by the process of proteolysis [DS10], which is the breakdown of protein chains into its derivatives. The natural process of proteolysis is slow but can be sped up by the introduction of enzymes, called protease. For example, a common protease is Trypsin [RLLK06], a serine protease, which is commonly found in the digestive systems of many vertebrates. Trypsin cleaves peptide chains at specific amino acids, which are predefined. This means the protein chain will be cut and the resulting sub-chains, an example can be seen in Figure 2.2. In the picture a protein is subjected to an enzyme which will alter the protein structure and produce two distinct peptides.

The enzyme cleaves the protein in specific positions, producing peptide strings. Each enzyme contains different rules, deciding at which position the peptide string will be split. For example, for Trypsin enzyme, the cleavage rules are the letters *K* and *R*, representing Lysine and Arginine. Should an amino acid chain contain those letters, the peptide will be considered in an array of substrings of the original organic compound. The enzyme also contains a restriction set of amino acids, which impede the split process as an extension to the rule.

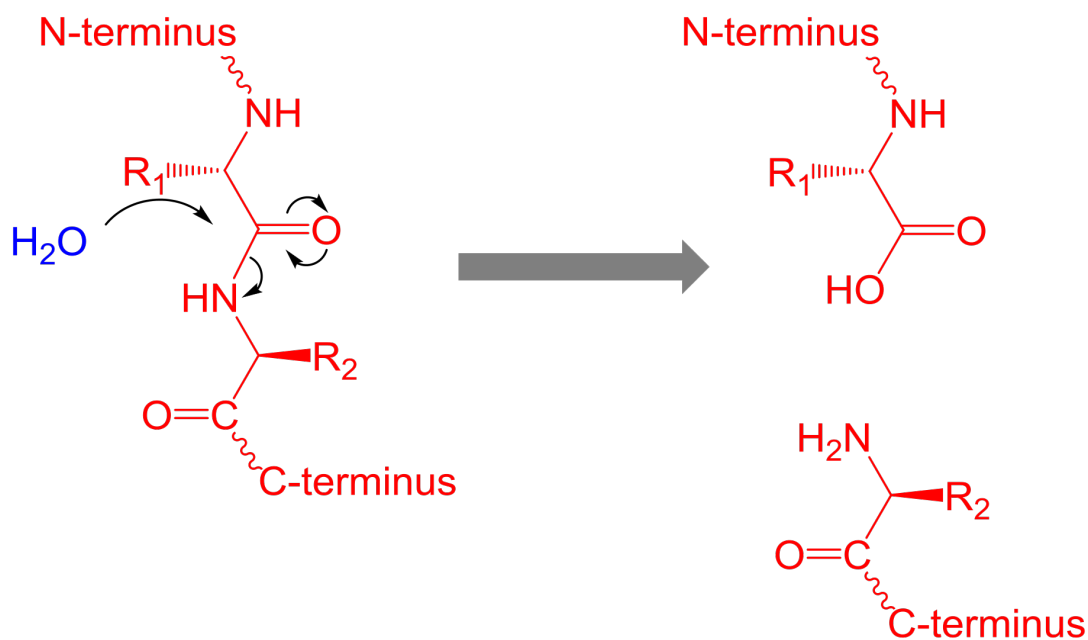


Figure 2.2: The hydrolysis of protein (red) by the effects of water (blue) [RLK06]. The properties of water manage to split the peptide into two at a specific amino acid site. By Thomas Shafee - Own work, CC BY 4.0, <https://commons.wikimedia.org/w/index.php?curid=42390219>

The end products are fed into the mass spectrometer, which measures the properties of samples in sequence and outputs a result for each measurement. This process, called protein sequencing [Met10], can be mimicked and applied to the sequence database. By applying the same rules, that are set to for the experiment phase, the resulting peptide set should be the same. In the bottom part of Figure 2.3, we attempt to do the same process but with software. We analyze a preexisting set of known protein chains and apply the same rules that should, in theory, occur in the real world experiment. We implement the rules that are a consequence of the presence of a digestive enzyme and factors in additional experimental setup properties. The rule set is then applied to the proteins, which are represented as strings of characters, encoding the exact amino acid sequence. The output is a peptide data set with theoretical properties, derived from the sub-string representing their amino acid sequence.

Having obtained a theoretical match for a single mass spectrometry spectrum, we evaluate the similarities of both real and theoretical spectra. Due to the real-world experiments being subjected to noise, fluctuations of the measuring digital signal, presence of contaminants or unforeseen environmental influences, it is challenging to have precisely the same properties with the theoretical counterpart. A good analogy for the comparison would be a picture of the object on a piece of paper and the object in real life - no matter how close the picture looks like the original, it lacks a whole dimension at least. The upside of this consequence is that excellent matches will less likely be wrongly classified.

2.2 Protein identification

In the real world, samples are rarely pure, and the presence of contaminants is to be expected. When analyzing a sample, ideally, the sequence database used should contain the composition of the protein, which is expected to be found. This is also a challenge, especially in the area of metaproteomics, where a set of various organisms are present. Identified possible peptides from the MS measurements are also often not contained in the knowledge base. The process of matching is finding the peptide from a spectrum, which has the highest probability to be a match, for the theoretical spectrum. The real-world samples may differ a bit or significantly from their theoretical counterparts [MBR⁺13]. Finding better ways to study samples and the ability to improve the matching phase has been a continuous driving force in the improvement of the metaproteomic field.

With improved experimental setups, the ever-increasing computational power and newly found architectures and algorithms for analysis result in the increase of the available data for analysis. Despite all the progress, there still remain problems to be solved, the most Important ones, as described in [SM19], are as follows:

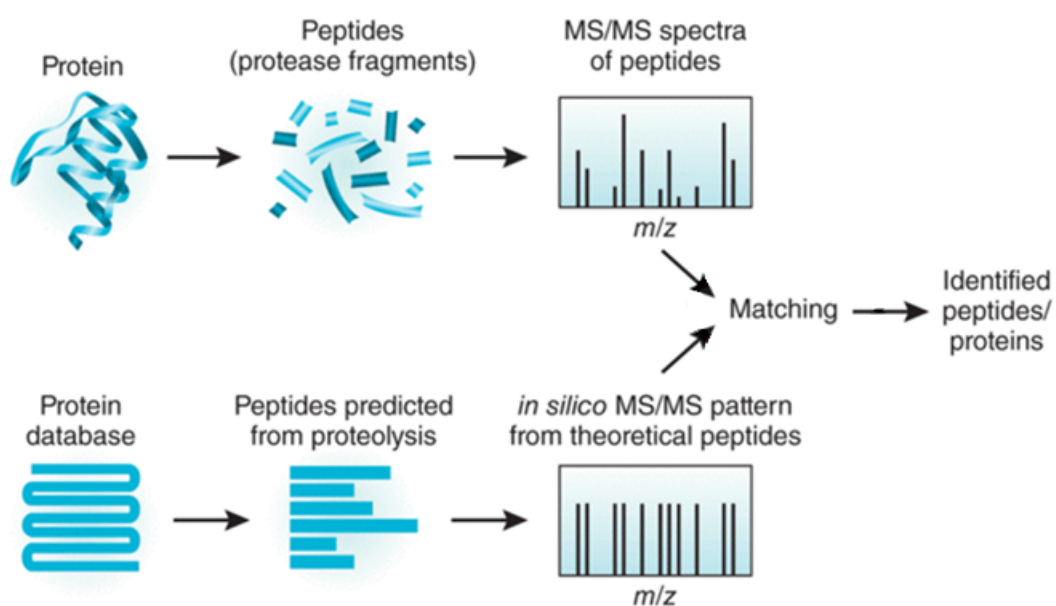
- difficulty in choosing representative sampling and protein extraction methods
- issues with selecting appropriate proteomic workflows and experimental settings
- item lack of tools and guidelines for optimal use of data analysis methods for identifying, annotating, and quantifying proteins to resolve community function accurately and to compare heterogeneous samples across time and space robustly

```
1 >sp|P02769|ALBU_BOVIN Serum albumin OS=Bos taurus GN=ALB PE=1 SV=4
2 AEFVEVTKLVTDLTK
3 >sp|Q6GZX4|001R_FRG3G Putative transcription factor 001R
4 OS=Frog virus 3 (isolate Goorha) GN=FV3-001R PE=4 SV=1
5 MAFSAEDVLKEYDRRRRMEALLLSLYYPNDRKLLDYKEWSPPRVQVECPKAPVEWNNPPS
6 EKGLIVGHFSGIKYKGEKAQASEVDVNMCCWVSKFKDAMRRYQGIQTCKIPGKVLSDLD
7 AKIKAYNLTVEGVEGFVRYSRVTKQHVA AFLKELRHSKQYENVNLIHYILTDKRVDIQHL
8 EKDLVKDFKALVESAHMRMQGHMINVKYILYQLLKKHGHGPDGPDILTVKTGSKGVLYDD
9 SFRKIYTDLGWKFTPL
```

Listing 2.3: FASTA format example. In this listing there two proteins described, a header is given after the '>' symbol, describing the protein, followed by its composition, represented by an amino-acid chained expressed as a string. Typical usage of this format is for a protein sequence database

2.2.1 Protein identification software - state of the art

A protein identification software needs to analyze vast amounts of metaproteomic experiment data obtained by mass spectrometry measurements. As input, the sample is digitized and may be encoded in different file formats. After this step, the



<https://media.nature.com/m685/nature-assets/nbt/journal/v28/n7/images/nbt0710-659-F1.gif>

Figure 2.3: Process of matching peptides. An assumption is built from theoretical protein composition and the theoretical values are compared with the spectra measured by a mass spectrometer. Theoretical values are calculated in ideal condition, while in the real world the presence of noise and contaminants is to be expected.

data may be used after converting it to a format, complying with the software, used to find known organic structures in the sample. This represents the third step in Figure 1.2.

Identifying proteins has been achieved by several different algorithms, such as Mascot [NPPCC99], Andromeda [CNM⁺11], X!Tandem [BCC⁺08], Sequest [Tab15] and others. Each differs in the approach towards recognizing a sample. However different these methods follow the same idea. First, there is a need to analyze the sample. Protein sequence databases have been built by analyzing proteins and methodically applying splitting (cleavage) rules, digesting the proteins into smaller parts - peptides.

During the splitting, the probability of a peptide occurring in a protein is measured, since a single peptide may be present in more than one protein. This knowledge base allows scientists to compare theoretical peptide with real-world experiment data. Masses are matched and scored with varying probability by analyzing the sample and counting the found known elements. If the recognized peptides are the building blocks of some known proteins, a guess might be deduced. This guess has various levels of confidence, based on the number of shared features a sample has to a theoretical peptide. Following a probability approach is beneficial in this case because the decision for significance can be obtained based on a set of rules. Those rules may have arbitrary complexity, mimicking the mass spectrometry experiment.

Matching a peptide and a spectrum produces a score based on a scale. The scale differs between algorithms, however the lower the probability of a match being found, the higher the assigned score will be. The significance of the match is also influenced by the data set being observed and the available protein knowledge base. Naturally, more knowledge leads to better assumptions, but this also means that a sample with a particular mass may also have much more candidate peptides or proteins. Additionally, the resulting mass might be just a result of chance, so the probability is further adjusted with constraints. For example, we might have identified 6 out of 10 known peptide masses, which belong to the same protein. Let us assume the probability for that would be a 10^{-5} . The chance may seem extremely small, but if we take into account that nowadays, sequence databases have well over million records in them, the scenario of finding a match by chance is quite significant. A common threshold for an observation to be made by chance is the probability of less than 5%. This means that for a data set consisting of 10^7 entries, the number of essential matches would be 5×10^{-9} . The resulting number is too small and varying, so it has been widely established to accept results with a score of $-10 \log_{10}(P)$ as a significant match [NPPCC99]. A match with a higher score represents a protein which is a better fit for the sample, and typically suitable matches have scored higher than 70.

The state-of-the-art protein identification software tools rely on having a complete digitized mass spectrometry experiment data before analysis begins. Depending on the tool and use case, knowledge bases represent a reference, for the experimental data is also built on the fly. This is decided based on the use case and the expected presence of known peptides. In the work by Zoun et. al. [RZS18], data is being processed as soon as results are digitized. This leads to a near-real-time data analysis, and this is also the target behavior of our own tool Cassiopeia. By using the already

validated and used in real-life open-source software Andromeda algorithm, we will implement its scoring function but process the data as soon as it arrives. Before diving deeper into the concept of our new proposed protein identification approach, we will take a look into the standalone Andromeda [CNM+11] software.

2.3 Andromeda

Andromeda is a probabilistic peptide search engine. Input data are mass spectrometer experiment results. Analyzing proteome data Andromeda achieves performance rivaling that of Mascot, which is commonly referred to as the gold standard when dealing with peptide identification, seen in the Andromeda paper [CNM+11]. This tool is also extensible, configurable, and open-sourced, it can work with simple protein data or one which contains arbitrary complex modifications present. Andromeda has a primary goal of providing flexibility when working with arbitrarily high fragmented mass accuracy, is capable of assigning and scoring intricate patterns of post-translation modifications, and works with data sets of any size.

The search process of Andromeda is as follows:

1. The search process begins with the construction of the protein and peptide sequence databases. Initially, the data set is given in a format, typically in a human-readable one like FASTA. This represents the theoretical database, which will be used as a comparison for the real world results. The current implementation transforms a sequential protein database into parts and creates indexes for easier access during the evaluation phase. This is done in three steps and can be seen in Figure 2.4.
 - (a) The first phase sorts the proteins by name and iterates over the whole presented set, creating indexes where each index points to a protein. The proteins are then broken down into peptides by applying environment-specific rules. These mimic the real-world experiment and what the theoretical outcome should be when a protein chain is put under specific conditions. Those conditions may be the presence of a digestive enzyme, which splits the protein at certain points, for example on encountering a specific amino acid.
 - (b) The second part of the sequence database preparation is the ordering of the resulting peptide set by the string representation of their amino acid chains. A protein is represented as a string. Therefore, a peptide would be a sub-string. By using lexicographical ordering, the peptides are stored, and indexes are created for the allocations of the ordered peptides.
 - (c) The last step is the ordering of the peptides by mass. This is a straightforward approach unless the experiment expects modifications. The modifications alter the masses of peptides by applying rules, where the weight of one or more amino acid elements might be altered. This results in the creation of a peptide with original mass and one for each possible modified mass. In the end, the whole set is ordered, stored onto storage, and indexed again.

2. The second step is the iteration of the present spectra. For each spectrum, we look up a corresponding peptide. This is done by looking for peptides with masses, which are in range of one the known peptides in the knowledge base. The range is calculated by a predefined tolerance, stating what kind of deviation from the peptide mass is acceptable. This is configured for each experiment, based on the environment used. If no suitable peptide is found, the spectra is skipped. Otherwise, it is transformed and submitted for evaluation.
3. The next step is the evaluation, where a spectrum is passed in, together with a theoretical peptide, which is deemed as a possible occurrence. We consider individual peaks in the spectra and include them in the evaluation. In the end, the probability of a match is scored and a result emits.
4. The last step is the storing of results. Usually, scores under a certain threshold are omitted. Should the score be high enough, the peak is classified as the recognized peptide and stored together with the origin protein, from which the peptide has originated.

The search process boils down into three simple steps - pre-processing, evaluation of feasibility for the spectrum to be present in the protein knowledge base and scoring the correlation between the peak and the theoretical peptide. The three steps are easily separated and have good granularity, which makes possible the delegation of each processes to separate functions. In Cassiopeia we have designed the architecture to be separate the areas of concern for each of the three tasks, with the idea of processes in parallel and in an immutable context. The next section describes the core idea.

2.4 Fast data

Currently, a single metaproteomic experiment takes around two hours to complete and produces up to 40 GBs and larger mass spectrometry data output [MBK⁺18]. This amount of data is the average produced by a metaproteomic experiment [SAB⁺08]. The sample is split, measured, and digitized, which may result in data sets of considerable sizes. One device can be managed by a local tool, but for a central solution with multiple devices, horizontally scalable architecture is needed.

The data quantity presents challenges when attempting to analyze it with conventional means. Architectures have been invented to deal with data sets of considerable sizes [PC15]. This encourages the usage of efficient algorithms for big data sets in handling the number of research results. In recent years, the term Big Data has become a buzz word for dealing with data sets composed of from gigabytes to petabytes and higher. The increasing popularity of problems dealing with vast quantities of data has sparked a lot of new approaches and is continuously evolving [Wam19].

There have been a couple of approaches in dealing with large amounts of data, by treating them either as batches or streams of data [MZK17]. The most common and most straightforward approach is to transform the data into pieces, which are easy to work with. The data set is split into batches, small enough to allow the usage

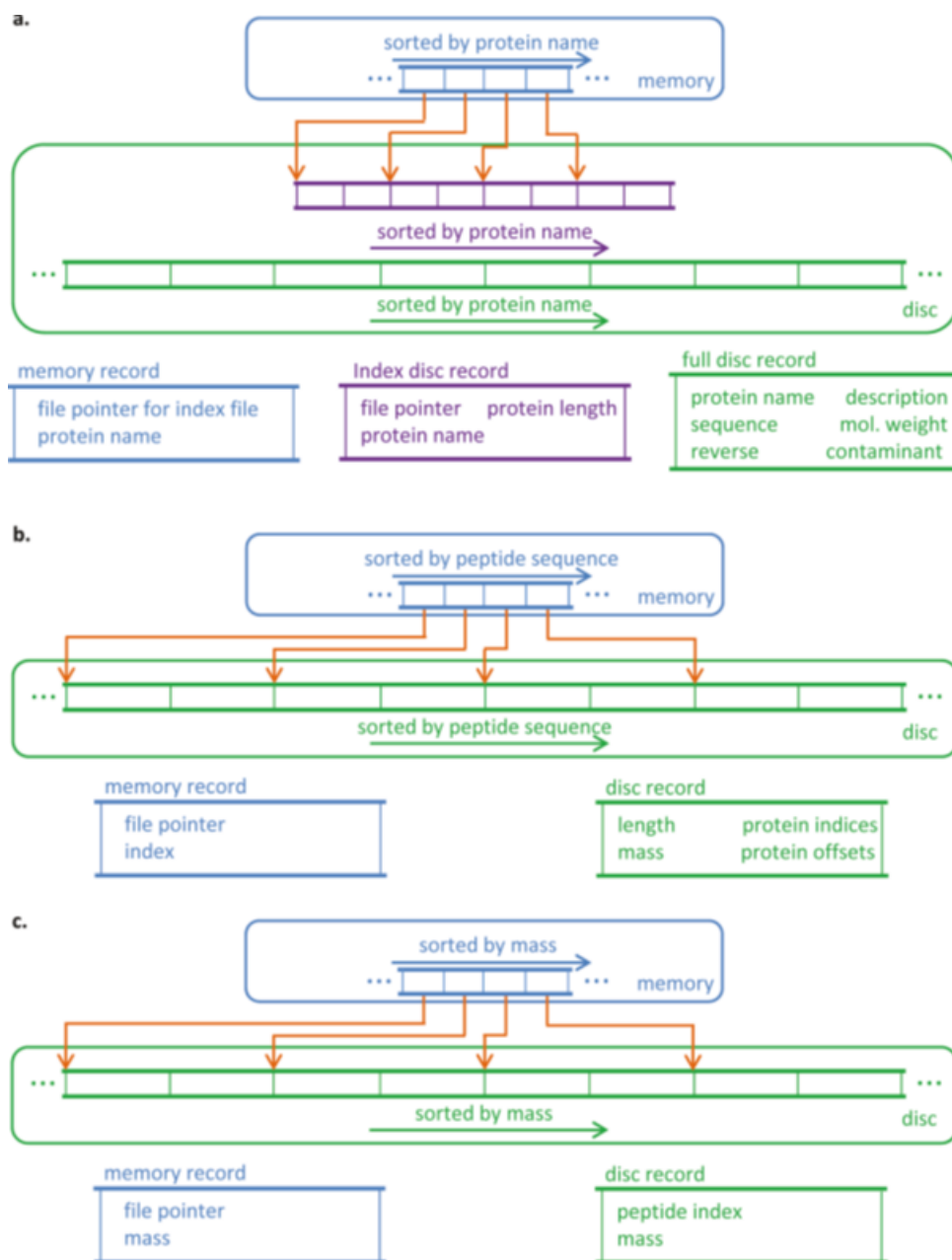


Figure 2.4: Steps used in building the protein sequence database used as knowledge base in Andromeda [CNM⁺11]. There are three distinct steps performed when building the index, a) sorts the proteins by name performs the initial splitting of proteins into peptides by applying experiment specific cleavage rules and writes the results onto storage while keeping an index file. Step b) sorts the peptide sequences by their strings and stores those into index files. The last step c) sorts the peptides by mass, while again keeping an index file showing where the data is allocated on the storage.

of usual functionality, which has already been used but has troubles when dealing with larger data sizes. Additionally, this enables possible parallelization.

Splitting the data however comes also with a drawback - we introduce additional overhead for splitting the big data and wrapping it into smaller chunks. This impedes real-time processing since the batching process takes some time, however small, which dilutes the data freshness.

Some use cases require to work with new data as soon as it is produced, and having a time offset could make the end cost too much for this approach to be a viable one. A simple example would be a stock exchange, where stock values vary each second. People buy servers close to the stock market itself because the latency of a couple of milliseconds may be the difference between loss and profit. In this scenario, a pre-processing step would not be feasible.

A second approach would be to treat the big data set as a stream of data. In this case, we could treat the data as a log [Wam19]. The received data is being handled at the moment of arrival, accepting potentially infinite data streams. This is what we call *fast data*. In the following sections, we give a brief overview of the Big Data evolution history, the emergence of Fast Data, and then we compare the traditional batch-based with the relatively new streaming approaches.

2.4.1 History of Big Data

After the creation of the internet, the resulting amounts of newly introduced information were not expected. Existing tools were not prepared for handling the constant introduction of new and existing knowledge. The internet was consistently supplying new information by allowing users to publish known and unknown knowledge. This happened in parallel to many different domains. The sheer volume of data was not expected to be of this magnitude [CO98]. This means that a need for reliable, highly available, and robust was present. The opportunity led to the emergence of the Big Data concept - unprecedented amounts of data are available and ready for utilization. As stated in [Wam19], in its core, the architectural design of Big Data can be represented by three components:

- Storage – the data has to be stored somewhere in a way that its fault-tolerant, minimizes transfer latency and to have the capacity to grow
- Compute – to use what is stored
- Control plane – managing and monitoring the resources dedicated to the task of handling data

These three layers were the core of Big Data. On top of it, other components were later introduced, but the idea remained the same. Big Data systems are divided into two general forms: databases and more general environment. An example of a database would be the non-relational databases, such as NoSQL, that were storing information in an agreed format. The concept was implemented in Apache Hadoop, which is an open-source software for reliable, scalable, and distributed computing.

The environment provides more significant flexibility in terms of usage at the cost of bigger management overhead. Additionally, in 2007, the Dynamo paper [DHJ⁺07] created many new NoSQL databases, using different data formats, such as XML, JSON, key-value storage, and others.

Different implementations had different trade-offs. Eric Brewer came up with a theorem describing said limitations, called CAP theorem. Working with distributed data systems brings a trade-off between consistency, availability, and partition tolerance. In 2002 Seth Gilbert and Nancy Lynch from MIT published a proof of Brewer's statement under "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services" [GL02]. In summary, the theorem postulates that a network shared-data systems can only guarantee two of the three properties with the following properties:

- Consistency – every node in a distributed database network will return the most recent successful write. Each client accesses the same view of the data.
- Availability – every node responds for all read and writes operations in a timely fashion. The focus here is to guarantee that absolutely all requests respond in a manner, attempting to avoid timeouts.
- Partition tolerant – the system always stays available, even in case of node failures. Should one element fail, there will be at least one to provide suitable substitution of functionality.

Abiding the rules from the above properties, there are three combinations, where at least two of the features will be according to the requirements. A system may be seen in the [Figure 2.5](#):

- Consistent and Partition Tolerant (CP) – the system will always be consistent and distribution tolerant, but no guarantees for availability can be made.
- Consistent and Available (CA) – consistency and availability are guaranteed, but there is no support for partitioning. These systems most often are single database servers.
- Available and Partition Tolerant (AP) – the environment is available and partition tolerant but not consistent.

The implications brought by the CAP's theorem made dealing with data from the always-on internet functionality to accept eventual consistency in exchange for better availability, fault tolerance, storage capabilities and latency minimization brought by partition tolerance. Eventually, SQL as a query language started catching up to NoSQL ones, due to the steady query optimization improvements. Nonetheless, loading the whole data was not feasible. Fast Data and streaming technologies were iterative results from technological advances in the field. The earlier approach was to split the data and use batches, a methodology present in the Hadoop tools suite. In the next section, we give a brief overview of how this was actually implemented.

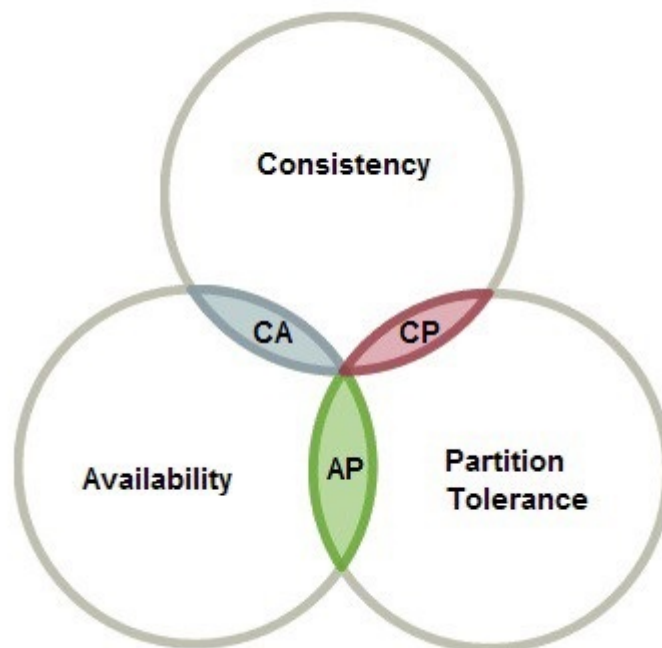


Figure 2.5: CAP Theorem

Metric	Sizes and units
Data sizes per job	TB to PB
Time between data arrival and processing	Minutes to hours
Job execution times	Seconds to hours

Table 2.2: Batch-mode systems [Wam19]

2.4.2 Batch-Mode architecture

In Figure 2.6 the standard Hadoop architecture used for batch operations is visualized. In the figure, the areas separated by dashed rectangles show the logical subsystem separation.

The resources may be shared across physical systems. The data is persisted into distributed file or database systems like Hadoop Distributed File System [Fouc] (HDFS) or other cloud or local based SQL or NoSQL tools – they are adjusted according to the scope of the problem being tackled. Current technologies to ease the work with data are, for example, Flume [Foub] for log operations and Sqoop [Foue] to ensure database interoperability between the nodes. Analysis jobs implemented in Spark or MapReduce [Foud] are run on the worker nodes. The final result is that we have the following performance in batch-mode systems:

2.4.3 Lambda architecture

Lambda architecture [BH] can be seen as an intermediate step between batch-mode and streaming architectures. The concept was first introduced by Nathan Marz in 2011 in his post "How to beat the CAP theorem" [Mar]. It encompasses the ability to work with bulky data, for which processing time does not play too big of a role,

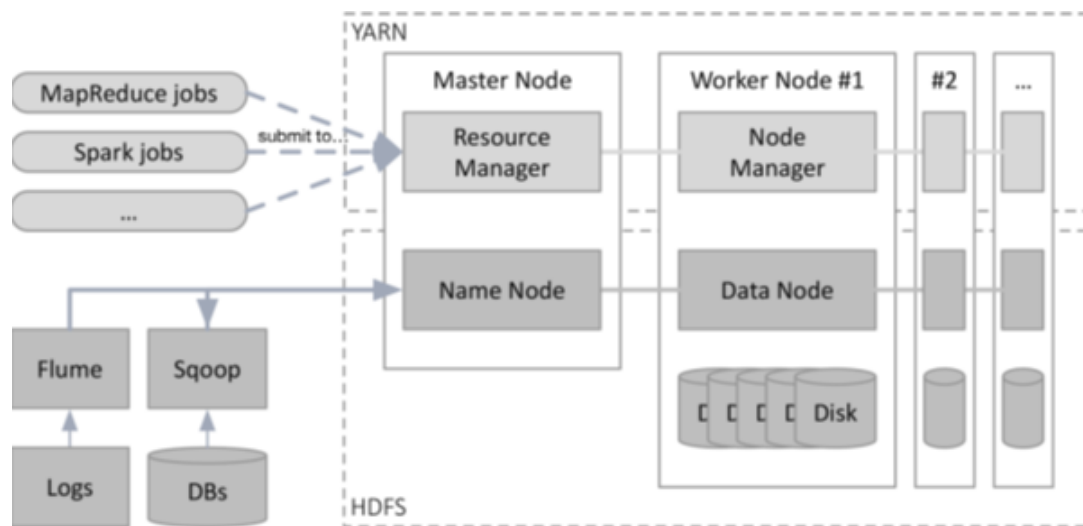


Figure 2.6: Hadoop Batch architecture

=

while also having a layer, which can be used for real-time processing. In essence, it is as a hybrid model with three layers:

- *Batch layer*: usable for large-scale analytical processes
- *Speed layer*: to work with newly arrived data in minimal time
- *Serving layer*: providing a query/view capability, managing both the speed and batch layers.

The goal of Lambda architecture is to provide a robust, fault-tolerant system while providing capabilities to work across the whole spectrum of data sizes. It provides ability to process data for Online analytical processing (OLAP) [CCS93], targeting huge data sets with possibly long-running calculations, while also being able to work with data in real-time, for tasks requiring low latency read and write operations. An example flow can be seen in Figure 2.7. The numbers in the picture describe the flow of data in the system and are as follows:

1. New data entering the system that is submitted to the batch layer and speed layer.
2. The batch layer manages the master data set, which is an immutable collection of data to which only addition of new data is allowed. It also pre-computes batch view results.
3. The serving layer is reading data from the batch layer, indexing batch-views for fast querying.
4. Speed layer is used to compensate for the high latency of new updates by the serving layer. It deals only with recent data only.

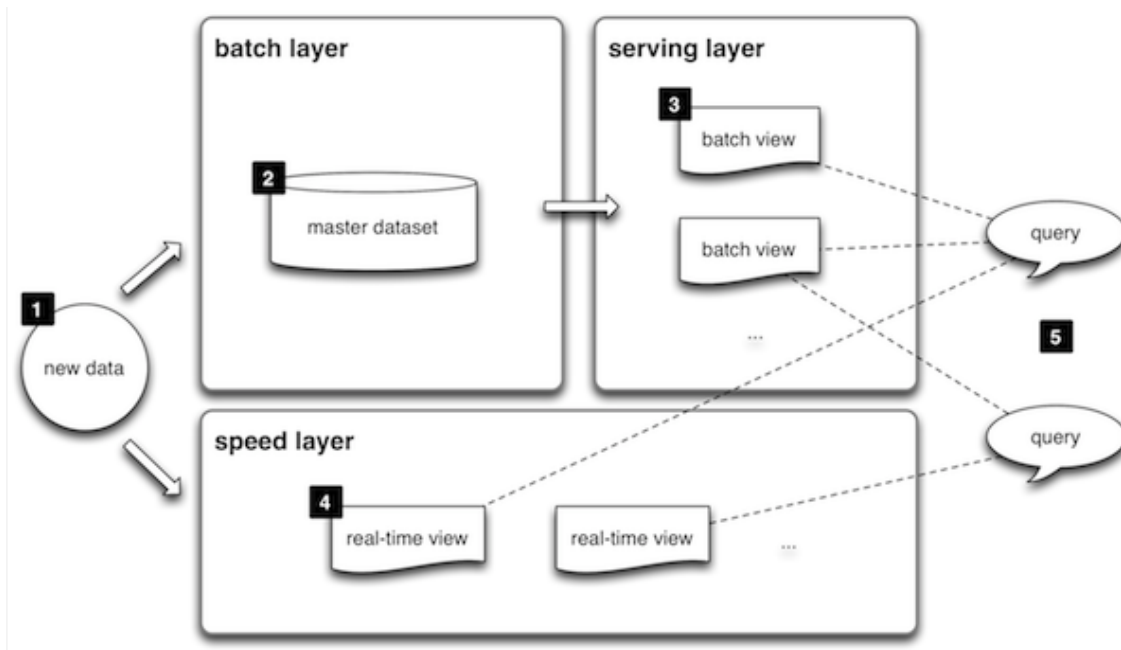


Figure 2.7: Lambda architecture workflow [BH]. Label 1 shows new data entering the system, 2 contains the batch layer with the master data set in it, 3 shows the serving layer managing incoming queries, 4 denotes the speed layer, showing intermediate progress for queries in real-time, 5 describes any incoming queries, which might target either the batch or serving layers.

5. Queries can be issued towards the serving and the speed layer, allowing work with batch-views and real-time data.

One could theoretically implement a fast data environment using a lambda architecture. The architecture however has some issues. The implementation of both batch and speed layers are separate and treat the data differently. This means the same logic might need to be implemented twice. The serving layer also needs to be able to handle two sources of data, which imposes additional overhead.

The field dealing with Big Data has also improved throughout the years. Previously there were no reliable strategies in working with data in a streaming context. Streams of data were treated as sub-batches, and there were no valid approaches to dealing with infinite streams. The assumption was that the streaming calculations might only be an approximation of the results and to have a complete and definitive answer, batches would always be needed. This has changed; the new strategies in dealing with unbound data can be seen in the posts by *Tyler Akidau* in his "Streaming 101: The world beyond batch" [Aki15] and "Streaming 102: The world beyond batch" [Aki16]. The lambda architecture played an essential role in the dealings with infinite streams of data.

2.4.4 Streaming Technologies

There are many different approaches in implementing a batch-mode system, but the implementation will always lack the ability to provide the absolute most recent

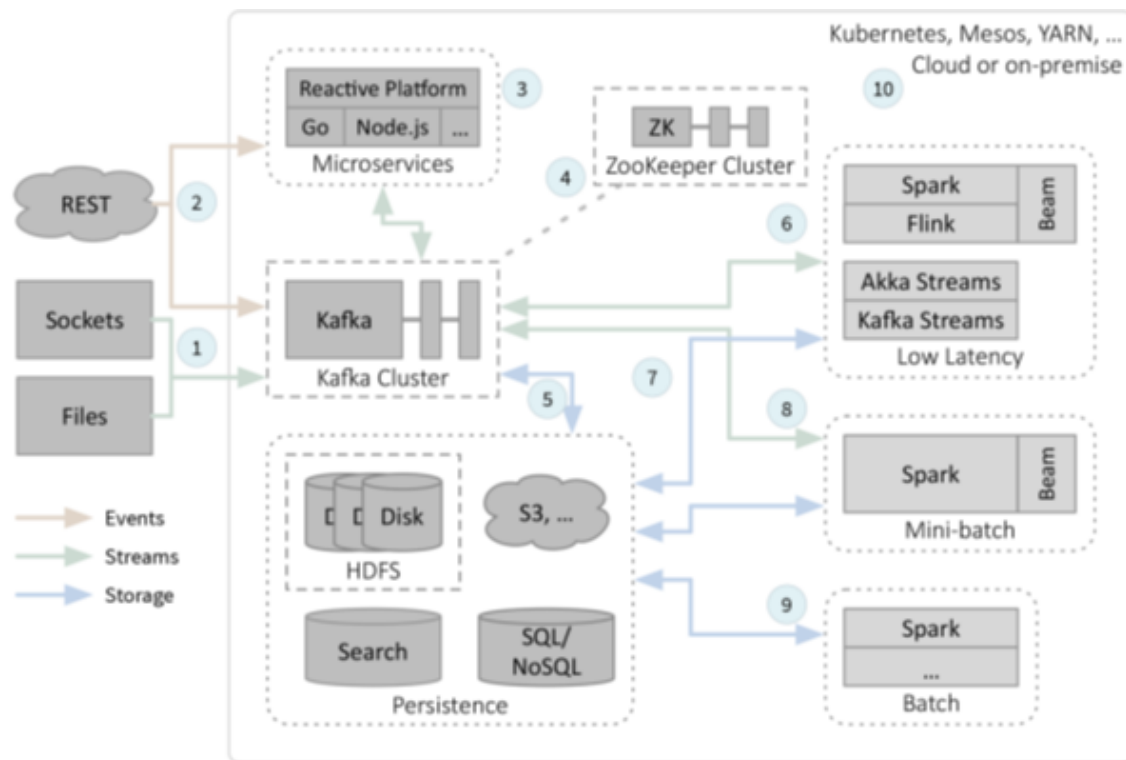


Figure 2.8: Fast data (streaming) architecture

data. The data will always lack freshness. Streaming workflow tracks the data live. This means that the processes or jobs analyzing data are running in near-real or real-time. This process can also be kept alive for as long as needed.

Streaming also introduces new semantics for analytics [ACC⁺16]. For example, in one of the most common data processing language SQL the usage of windowing. Window functions perform calculations over data, while not mutating the observed data set - the data is immutable. In contrast to that, using an aggregate function mutates the transformed data, while window functions keep the original data state. Since we are working with streams of data, we can not be certain when and if an end will come. Performing aggregating functions on the whole available data set will always mean that the operations are conducted on an incomplete knowledge base. This means that the analytic process might miss some yet to be introduced important features, outliers or even terminal states. A balance between the correctness of data and the need to analyze the data must be struck. In Figure 2.8 a sample streaming architecture is shown:

In contrast with the batch-mode system architecture seen in Figure 2.6 more elements are present. The elements in the Figure are numbered, and their functionality is as follows:

1. Input streams can have various sources of origin, such a socket connection or reading from a file. The data is represented in an acceptable data format and should be consumed in the same fashion. In our scope, the data is the

output from the mass spectrometer measurements and fed into Kafka topics using Kafka producers [Kre11]. Kafka helps in handling the data and making it scalable, reliable, takes care of the partition distribution and durability of the data. It also provides control over how the data should be distributed across the network, where and how it should be stored, and while caring for the load balancing.

2. REST (Representational State Transfer) [BB08] represents immediate payloads with data for processing or configurational actions to the Kafka cluster. The usual use case here is the triggering of events changing the state of processing. Working with REST includes more overhead for working with records of data and web socket connections are preferred for feeding new data into the system. Nevertheless, this protocol allows a bit more flexibility in the means of communication.
3. A suite of microservices is used as a form of control and management mechanism [AAE16]. The applications here are endless; this allows for an extension of the normal workflow by introducing adapters, services, and external means for communication or storage. Whatever is needed for the given task can be added as functionality – from parameter control to state changes in the analytics.
4. Apache Kafka [Kre11] is a distributed system, and it lives in a ZooKeeper (ZK) container. It provides the needed log processing to achieve the streaming behaviour. In its essence, Kafka is a distributed messaging system. It obfuscates the means to keep track of where the message is currently physically stored, it manages different domains of information in the form of topics and simplifies the implementation process. The ZooKeeper container helps Kafka in caring for how and where the data from Kafka will be stored. Additionally, the cluster used to supply the environment may also be used for other processes.
5. Using Kafka Connect data can be persisted from Kafka to long-term, persistent storage. For example, newly identified peptide sequences may be stored for future reference. The arrow in the Figure 2.8 is two-way because data from the persistence layer may be fed into the Kafka flow.
6. The Kafka cluster provides the data and makes it ready for processing. Working with the information is done by streaming engines, which can be Apache Spark, Apache Flink, or other implementations, such as a micro-service architecture. There is quite a lot of flexibility in the architecture. The Flink and Spark tools can be thought of like software which deals with jobs. Data is passed with instructions of what needs to be done, and the task will be done somewhere in the future. Apache Beam also inspires them in providing a considerable analytical toolset. The other option to just use Akka [NAR⁺19] or Kafka streams brings less functionality out of the box but also less processing overhead. However, any implementation will have low latency as a feature.
7. The processing engine has access to already persisted information while having the ability also to introduce new entries. This is useful since it might skip the

Metric	Sizes and units: Batch	Sizes and units: Streaming
Data sizes per job	TB to PB to PB	MB to TB
Processing time	Seconds to hours	Microseconds to minutes
Job execution times	Minutes to hours	Microseconds to minutes

Table 2.3: Streaming systems

process of sending results first to the Kafka cluster or any similar approach, but instead store the information and send a reference of the newly written data.

8. Initially streaming was thought of to be implemented by incorporating mini-batches while using Apache Spark and Beam. The batching process leads to bigger latency since there were defined periods when the batching should take place, and when the result would be sent. Although a drawback, this can be used to fit into expensive calculations which can be useful when the latency is not a primary concern.
9. Although the architecture is tailored towards streaming capabilities, the option always remains to await sufficient information to wrap it into a batch and work like a batch-mode architecture, the flexibility for that is present.
10. All this architecture can be located arbitrarily: it can be on a physical server in the near vicinity or deployed in the cloud. Different tools can manage the cluster resources; flexibility is present yet again.

A comparison between batch-mode and streaming architecture can be seen in [Table 2.3](#). We can see that the processes in streaming take less time due to the different distribution of processes. Generally, the approach is more granular and focused on continuous analysis of smaller chunks of data for smaller time frames. This is useful for any domains, which target working with data in real-time.

2.5 SMACK stack

Working with streams of metaproteomic data requires much data to be processed. A single sample can be composed of multiple proteins, which can be further split into smaller building blocks - peptides. The size of the original data plays an exponential role in the final result set. On top of that, the protein sequence database has to be built by applying the same protein cleave rules, used in the real mass spectrometer experiment. The theoretical protein set gets split into an even higher quantity of peptides since it usually contains an even bigger set of proteins because it needs to be used for the identification of unexpected peptide occurrences.

In this chapter, we covered an already possible computational concept, which may enable a fast and near real-time [RZS18] performance - fast data. The streaming approach of a fast data architecture enables the analysis of newly analyzed data as soon the measurement results are digitized. In this section, we take a look into a possible environment, which may enable the usage of a fast data approach - a SMACK (Spark, Mesos, Akka, Cassandra, Kafka) stack.



Figure 2.9: SMACK (Apache Spark, Mesos, Akkaa, Cassandra and Kakfa) [ER16a] overview. The graphic shows the roles and connections between each component from the technological stack.

2.5.1 SMACK definition

The acronym *SMACK* describes the following five technologies:

- *Apache Spark* - plays the role of the engine
- *Mesos* - represents the container
- *Akka* - encompasses the model
- *Cassandra* - is used for the data storage
- *Kafka* - relays messages and is a broker

The SMACK stack represents a concrete implementation of an architecture. The above-described technologies are all open-sourced and readily available to developers. The main task of the architecture is to provide an extensible and scalable solution for tackling large quantities of data. Additionally, any solutions, implemented with this technology stack, work as a distributed software. The geographical location of any servers, on which the implementation is located, can be anywhere as long as it minimizes latency and maximizes throughput. It is also scalable, where computational power is directly linked to the computational power dedicated to the supporting hardware. Having these properties makes this architecture ideal for the usage in a cloud environment. An overview of how the individual technologies are intertwined can be seen in Figure 2.9. In the following sections we give a brief description regarding the role of each component in the SMACK stack.

2.5.1.1 Apache Spark

Apache Spark plays the role of processing engine in the architecture of SMACK. It is used to analyze data in real-time and performs analytical workloads. It performs

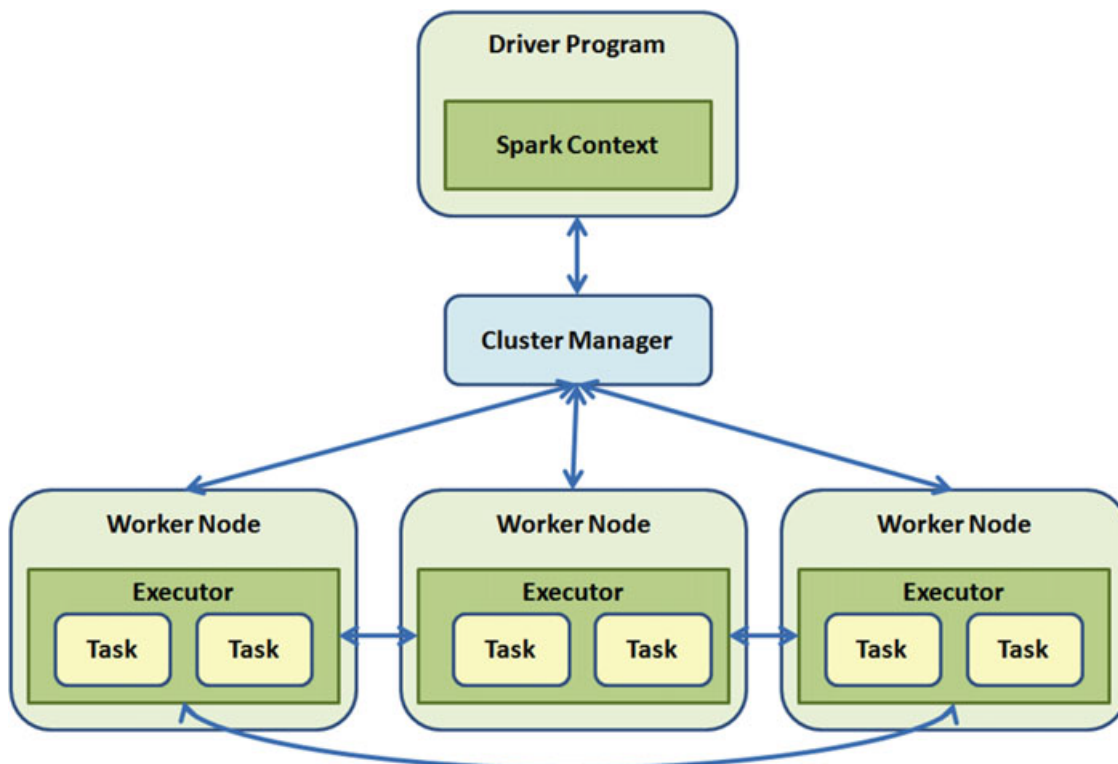


Figure 2.10: Apache spark cluster managing three worker nodes

poorly when considering OLTP (Online Transactional Processing) tasks and is suited for aggregations and other analytical tasks. There are other alternatives to it, the most common one being Apache Hadoop or Apache Flink. It boasts 100 times faster performance on batch and streaming processes [VMJ16] however, compared to Hadoop. This has been achieved by a new state-of-the-art Directed Acyclic Graph Scheduler.

Additionally, the Spark engine is very tolerable regarding the supported languages. In contrast to Hadoop, which accepts only Java code, programs may be developed in Scala, Python, Java, R or SQL. This simplifies the development cycle, giving flexibility towards new implementation. If the solution has high enough granularity, components may be implemented parallelly by separate teams of people, using different programming languages but working with the same engine.

Spark takes care of the infrastructure and the running of the processes of a program. It can also be used as a data science tool, able to handle massive data sets and perform operations on them. It works with resilient distributed data sets (*RDDs*) [CM15], which provide fault tolerance, efficiency, speed, and in-memory data storage [ER16b]. Additionally the RDDs enforce immutability so no side effects from parallel running jobs may interfere. When running on a cluster, a spark driver program delegates the work as jobs to its subsidiary worker nodes (Figure 2.10). This enables scalability and is perfect for working in the cloud environment. Spark can work with different types of cluster managers, but in the SMACK stack, Mesos takes care of resource management.

2.5.1.2 Apache Mesos

Apache Mesos [HKZ⁺11] represents the abstraction layer over the available computer resources, such as CPU speed, available memory, and storage allocation. It can work with physical or virtual environments and plays the role of a distributed system kernel. It manages available system resources and is build upon the Linux kernel principles as abstraction guideline. In a distributed system finding the best approach for execution is by no means an easy task [UAK18]. A distributed system may contain nodes on local machines and cloud environments. The work has to be split in such a way that the subsequent worker nodes may be utilized as much as possible. The resources should also be available to use while also providing reliability.

Mesos can be classified as a general-purpose cluster manager. It is used to manage batch processes and others like streaming or one-time jobs. When deployed into a data center, Mesos abstracts the resources available and directs the workload for ongoing jobs. The architecture of Mesos is composed out of a master daemon, which can handle up to 10,000 slave daemons [Foua]. The slaves manage software components called Mesos frameworks, which hold the responsibility for task execution. The master agent initially offers each slave daemon an option of resources and manages to CPU, memory, and storage space in this way.

The framework running on top of Mesos container consists of two parts: a scheduler that registers with the master in order to obtain resources and an executor process, which performs the actual work. The master decides how much resources to delegate to a worker node, and the node itself dictates the utilization of available resources.

2.5.1.3 Akka

Akka [NAR⁺] represents the model and run time for the distributed system of the SMACK stack. It adds the actor model functionality to the SMACK stack. The actor model is a mathematical model was first described in "A *universal modular ACTOR formalism for artificial intelligence*" [HBS73] developed by Carl Hewitt, Peter Bishop, and Richard Steiger at MIT in 1973. Its purpose is to handle parallel processing in a high-performance network. Traditional object-oriented programming (OOP) has already been used as a tool to work with distributed data systems [BGL98].

There are some downfalls to classical OOP when working in a multi-threaded system in a distributed environment. For example, the encapsulation principle, one of the four OOP principles [Bla13], is troublesome to adhere. Objects can guarantee encapsulation only in a single-threaded environment. When working with multiple threads with singletons [GHJV95], for example, we need to use locks [BB89]. Locks are expensive CPU operation and inhibit the performance of the process because one has to wait until their release, before working on the locked data. Additionally, it introduces other problematic concepts: the deadlock and livelocks [Sif80]. Handling these problems introduce additional overhead for dealing with them [HFV17].

Other problems for OOP design on distributed systems would be how the data is stored. To simplify object properties are stored into the cache lines of a CPU [CNL⁺08]. Most of those are local to the CPU doing the calculation in order to

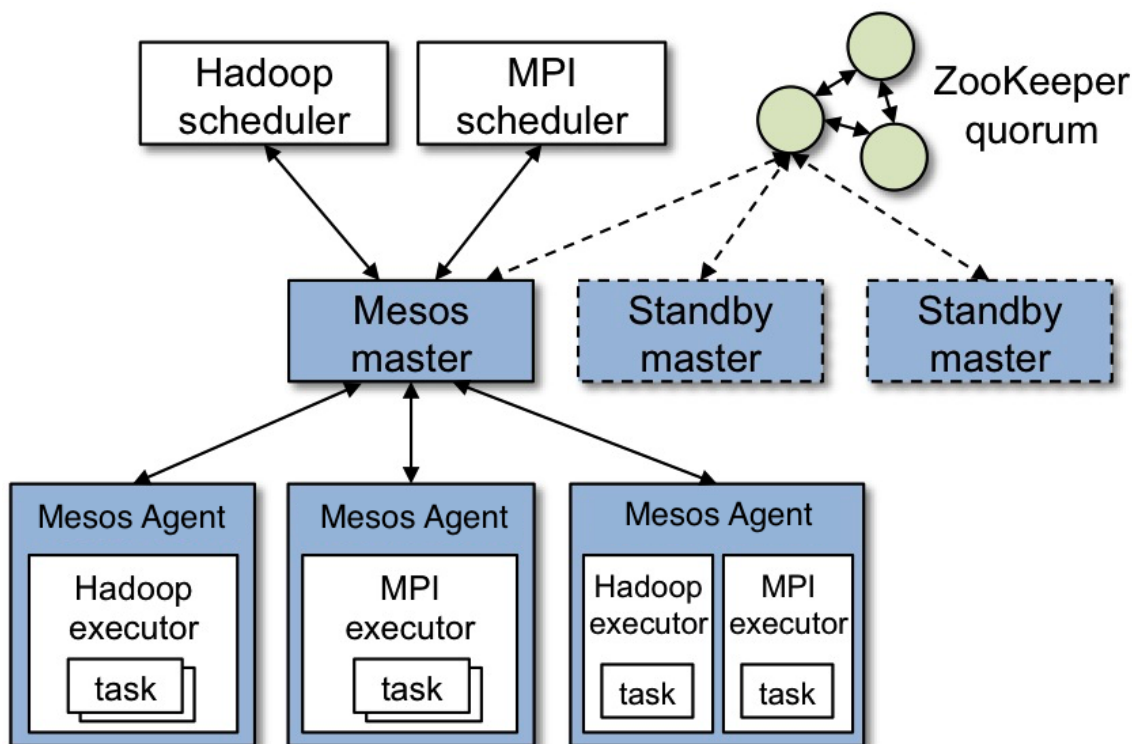


Figure 2.11: Apache Mesos architecture [Foua]: master daemon (only one at a time) interacts and manages slave agent daemons

propagate the new changes to a new core or thread in an atomic fashion. A better variant would be to keep the state locally and send data or events via messages.

Using an actor model helps mitigate some shortcomings of the traditional OOP approaches. We can adhere to encapsulation without resorting to locks. The actor models also contain their state locally and communicate changes in the state via messages. An actor can send an asynchronous message without blocking the process execution. Sending a message does not await a response, in contrast to a method, which role may be to provide a value upon which the future work is dependent. Actors react to an incoming message independently, not mutating its state in a multi-threaded environment. They process new messages sequentially with a queue, ensuring a correct concurrent flow of the program. Instead of locks, they use synchronization principles to ensure consistency. Having no blocking conditions, actors are better suited for a fast data processing pipeline.

2.5.1.4 Cassandra

Apache Cassandra [Fou16] plays the role of a distributed database in the SMACK stack. It is extremely fast and scalable, can be deployed over multiple data center to ensure the resilience of the data and is very easy to use. Similar products are DynamoDB [AWS], MongoDB [Mon], Redis [Lab] and others. The data contained in Cassandra is in a NoSQL format. Using the NoSQL format makes any underlying database to be without a schema, non-relational and cluster friendly, as described in *NoSQL distilled* by Martin Fowler and Pramod J. Sadalage [SF13].

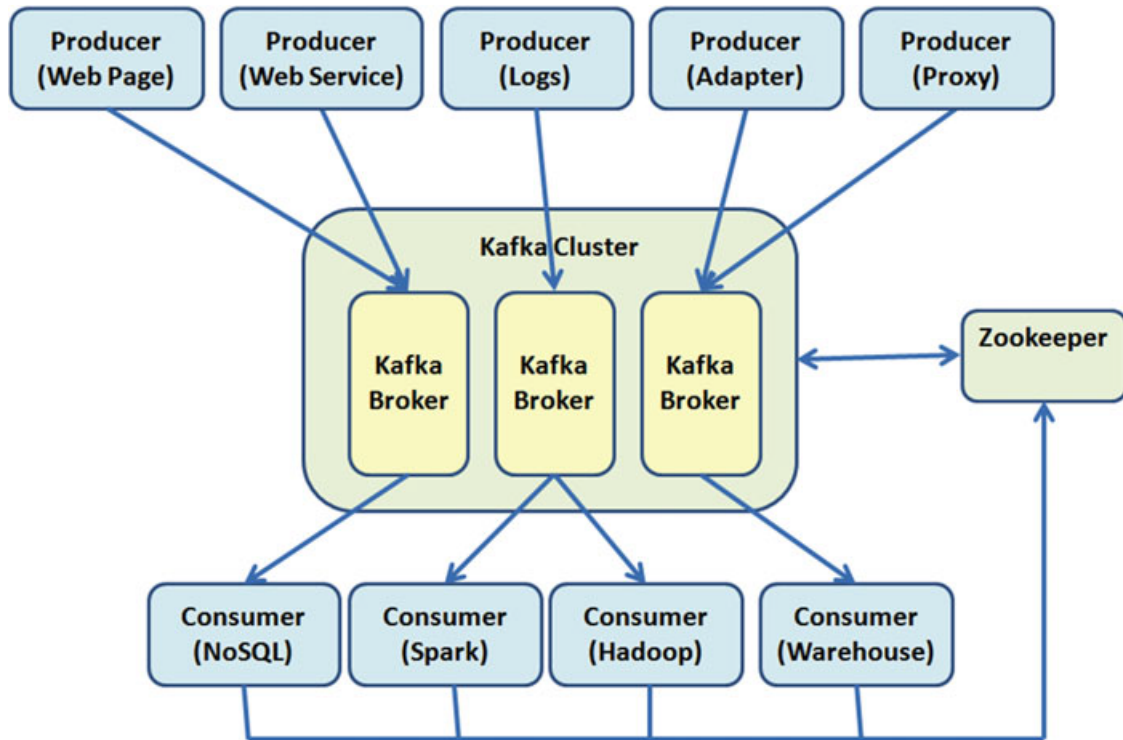


Figure 2.12: Apache Kafka typical scenario [ER16b]. In the picture the connections between heterogeneous systems to a common message publisher (Kafka) is presented. External systems can read data published in topics of interest.

Having no data scheme, the data stored can be accessed easier and manipulated a higher degree of freedom, compared to traditional SQL Databases. Cassandra also implements the "no single points of failure" principle. This is achieved by having redundant nodes and data stored. The architecture can be described as a ring hierarchy of nodes, where each one fulfills the same role. Having data redundancy supplies higher availability of data because it is much easier to maintain data replication [WLZZ14], when the copy of the data is present across other data nodes.

2.5.1.5 Kafka

As we stated previously at the beginning of the section, Apache Kafka [Kre11] plays the role of message broker in the SMACK environment. It provides a smooth implementation for delivering messages across heterogeneous systems. Dealing with Big Data generally poses two challenges: gathering the data and then analyzing it. Message publishing is a way of tackling the first problem. Apache Kafka presents a medium through which messages may be forwarded toward systems of arbitrary type, so long they are configured in advance to be able to listen for new data.

The Kafka message broker needs to provide data in a non-blocking fashion and also obfuscate any present and active subscribers for each topic. Its characteristics are as follows:

- *Distributed* - The system is designed around a cluster concept to support a distributed database system. It allows to grow the cluster horizontally, without downtime.

- *Multiclient* - It provides high interoperability, where clients using different technologies may subscribe to the message broker.
- *Persistent* - Possessing its replication strategy Kafka guarantees persistence with data access of $O(1)$ performance
- *real-time* - Any messages published are made available in real-time and the access can be instantaneous, limited only by the latency of the data center.
- *Very high throughput* - Being based in a cluster environment Kafka can handle hundreds of I/O operations per second from multiple clients. The only precondition is a load-balancing strategy, ensuring constant availability and no starvation of resources.

A typical Kafka architecture can be seen in [Figure 2.12](#). There are multiple producers, which can be of various types. All of them are generating messages and publish messages towards the message broker. They can be services located on the web or program services, used to log messages and perform communication by emitting events in the form of messages. The consumer can be anything, suitable for processing messages and able to subscribe itself for a topic. It can be a database or a program, analyzing newly published data in real-time.

2.6 Summary

In this chapter, we covered the essential concepts that are needed to understand the concepts and terminology covered throughout this paper. We cover the basics for the mass-spectrometry, the metaproteomic fields and how those fields can generate vast amounts of data. We also covered the concepts of fast data and a concrete implementation - the SMACK stack. In the next chapter we go over how the above mentioned-concepts can be used to utilize Andromeda's scoring algorithm and implement it into a cloud-based environment, while keeping the original scoring accuracy.

3. Concept

In chapter [Chapter 2](#), we have gone over the necessary steps needed for the protein identification process. In this chapter, we go over the conceptual design of software used for peptide identification. We present the use case scenario for our tool and discuss what requirements are present and what kind of constraints may apply. We begin by analyzing another protein search engine: the Andromeda stand-alone solution [[CNM⁺11](#)]. We plan on utilizing its scoring algorithm and integrating it in a fast data architecture, but an arbitrary software tool in this domain should be able to be used, as long as it adheres to previously agreed format. We then look then into what parts are needed for our use case to be beneficial.

We covered the process of protein identification in detail in chapter [Chapter 2](#), section [Section 2.2](#). Based on the processes needed, the work of Zoun et. al. in the face of the *MStream* protein analysis software [[RZS18](#)] and the Andromeda stand-alone implementation [[CNM⁺11](#)] we propose a new combination of two approaches - using the Andromeda scoring mechanism on a fast data architecture. As stated in [Chapter 2](#) and in the previous section, state-of-the-art protein identification tools require the whole mass spectrometry experiment data set to be present on start. This is a considerable bottleneck for analysis, especially if there is a need for real-time data processing. In Andromeda, the scoring is also done in a batch manner, where peaks are iterated sequentially. This means that during an evaluation, only a single core is utilized. Nowadays, CPUs rarely have a single core, therefore a non-blocking, asynchronous approach gives a significant increase in performance. Running locally also is a constraint, since we have limited vertical computational power. Having a fast data architecture, deployed on a cloud environment, promotes horizontal scalability, limited only by available resources. The software represented by MStream provides a concrete environment supplying such scalability and promoting availability and resilience by reducing possible down-time.

3.1 Existing concept provided by MStream

The MStream provides a solution for the bottleneck imposed by the need of state of the art protein identification software to have the whole experiment result batch

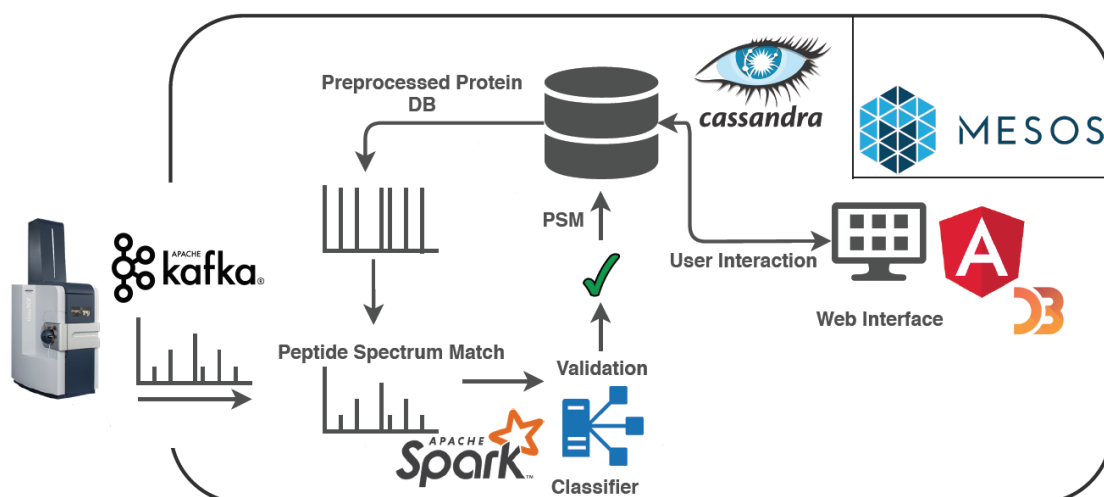


Figure 3.1: Architecture of MStream [RZS18], analytic platform for real-time diagnostic of mass spectrometry data

before an accurate evaluation may be performed. The software by Zoun et. al. is focused on streaming individual spectra from a mass spectrometer experiment in near-real-time. Its architectural overview can be seen in Figure 3.1. The flow is focused on finding peptide matches, validating them and storing good matches, while allowing users to see the results in near-real-time.

The experiment run-time and the digitizing phase and the conversion of data, seen as steps 2 and 3 in Figure 1.2, can be optimized if there is no need to wait for the whole batch to be done. It is better to send intermediate results for analysis. Raw data can be shipped for conversion as soon as a spectrum is generated. The conversion can be then delegated to a process, which takes care of transforming measurements in format, suitable for the protein identification software. If this is done by a message queue, sent to an API or published to a message broker, the new data can be distributed across data centers. This allows the results to be accessible across systems, scale the accessibility of data, and minimize latency.

For the process of identification, shown as step 4 in Figure 1.2, we rely on having a theoretical peptide, which should have features closely resembling the spectrum. The theoretical peptide is retrieved from a protein knowledge base, composed out of protein sequences string representations, put under the same theoretical conditions as the original metaproteomic experiment, to which the same experimental rules are applied. This produces sequences of peptides, which are sub-strings of the original proteins. This takes time, dependent on the total number of proteins and the complexity to recreate the experimental conditions. After composing the information in a format, which is suitable for the used protein identification software, this database might be reused for future experiments.

On a cloud environment, we are dealing with distributed management systems. The scalability of the cloud and the state-of-the-art database technologies offer a new and a better solution - having centralized storage for the protein knowledge base.

The sequence database can be reused across experiments, as long as the environment variables are the same, and can be updated with newly identified suitable peptide matches. However, in order to have a good peptide sequence knowledge base we need to be confident in the correctness of the results.

The general idea of MStream can be summarized as the following:

- Stream the analyzing phase: to find a way to stream results, produced by the mass spectrometry experiment and to send it for processing and analysis as soon as a part of the sample is complete. Publish the data in a manner which is suitable to be accessed by a distributed system or systems.
- Newly added results need to be processed as a stream. The goal is to consume results in real-time and analyze them as soon as available.
- Find a better way to supply a theoretical peptide, which possesses similar features for the observed spectrum.
- Derived scores should also be submitted in a near-real-time. Ideally, newly found peptides, which have a high score and not found in the existing protein knowledge base, should also be included. In this step, we need to adapt Andromeda scoring function to the new environment and add to the MStream pipeline.

3.1.1 Cassiopeia role in MStream

The solution provided by MStream is distributed across systems in a cloud environment and highly scalable. It could benefit from an improvement in the validation of the results, which can be a role fulfilled by Cassiopeia. By introducing a new protein search engine, it can be used as a cross-validation mechanism. As long as both methods for the scoring of peptide matches are good individually, the confidence will also scale upwards. We propose the following extension of MStream architecture, seen in [Figure 3.2](#).

The difference from the original work of Zoun et. al. is that we can add additional protein search engines, usable for cross-validation. In [Figure 3.2](#) we have only two distinct ones but in theory if additional ones are introduced they should be usable too, as long as they adhere to the same standards. The main requirement is to have a protein search engine, which has good performance, scalability and is able to be validated. Cassiopeia should be able to accept a peptide and a spectrum and derive a result object, which adheres to same standard imposed by the architecture so it can be cross-validated with the results of other protein search engines. Deriving scores does not mean much unless the results are validated and the outcome is reproducible. Having a high score for certain peptides does not bring much if unrelated peptides are ranked high. This architectural design should also be usable for a local environment the concept is the same, although lacking in scalability.

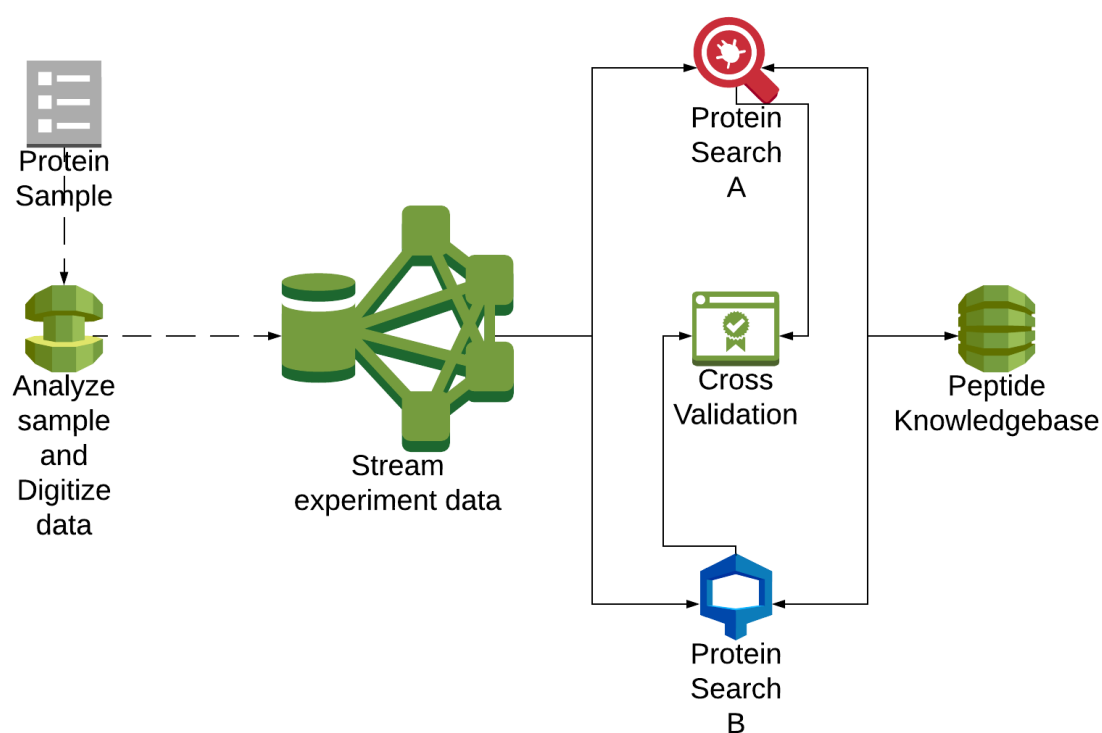


Figure 3.2: Concept architecture of MStream using multiple peptide identification algorithms. The spectra will be provided as a stream, theoretical peptides will be retrieved from a distributed peptide sequence database and given as an input to the search functions. After both tools have evaluated the potential matches the results will be compared to check the validity of the scores.

3.2 Cassiopeia concept

In [Chapter 2](#) we introduced the open-sourced Andromeda protein search engine. The goal of Cassiopeia is to implement its scoring function and to integrate it into the existing MStream environment. It will be then used as a cross-validation mechanism to verify the correctness of matched theoretical peptides with individual spectra against the existing protein scoring mechanism. Therefore our new tool also to comply with the architectural standards seen in [Figure 3.1](#). In the picture, the concrete technologies are shown, giving an overview of the used technology stack, so Cassiopeia has to fit well with the used technologies. The current Andromeda stand-alone software is written using Microsoft .NET. It is possible to wrap its functionality and use it in the existing environment. However, it is not suitable for working with streams of mass spectrometry data. It also requires to have the whole spectra batch beforehand, together with a protein sequence collection for the peptide knowledge base. Since the overhead for adapting the Andromeda software could very well outweigh the benefits of its scoring functionality, we have decided to use it instead as an inspiration for a new implementation of a protein identification engine.

The first thing to consider was the programming language for the implementation of Cassiopeia. Since we have an environment running Apache Spark jobs, we decided that the Scala language is a suitable fit. Although any other language could also be theoretically used to implement the Andromeda scoring function, we have chosen Scala for its simplicity and ease of integration. More about the concrete implementation and technologies is discussed in [Chapter 4](#).

We showed a concept architecture where Cassiopeia is used in [Figure 3.2](#). In it, spectra are given as input with theoretical peptide matches, supplied by a protein sequence database. Since it is integrated into an existing ecosystem, it has to conform to the present predefined data formats. Additionally, the new scoring functionality has to be validated with regards to the output of the original Andromeda software. This means that the challenge of Cassiopeia is two-fold:

- To fit easily and efficiently in the existing MStream ecosystem
- To have validated output, completely matching the results of the Andromeda software

3.2.1 Cassiopeia validation

Before we can use the new implementation of the Andromeda scoring algorithm, we have to verify the correctness of the output. Andromeda has already evaluated and known performance in the metaproteomic field. Implementing a new software with a known to work algorithm brings nothing if the resulting output is not the same as the original. To have an unbiased and good comparison, we have decided to recreate the flow of the standalone Andromeda algorithm as closely as possible. We have also attempted to introduce improvements, where possible. In Andromeda the search of suitable peptides is boiled down to looking for a theoretical peptide, which has a mass in an acceptable range of the inspected spectrum. A representation of

the components can be seen in [Figure 3.3](#). The protein knowledge base is built from files in FASTA format, containing information for known protein sequences. Those are processed sequentially and digested, by splitting the protein string into sub-strings. This "digestion" mimics the effects an enzyme would have on a protein in the real world, which ends up splitting the sequences into smaller sub-sequences. The outcomes are analyzed and stored onto the persistent storage, where indexes are also created for easier data traversal. The spectra are also loaded from a batch and given as an input to the Andromeda search function. The evaluation performed in it boils down to iterating each peak and looking for theoretical peptides, sharing features of the spectrum. Found peptides are stored into a local cache data structure for re-usability and to skip the same evaluations. The process finishes by scoring the similarities between the theoretical peptide and the peak, then produces a result. Before emitting a result, it goes through a filter, which is adjusted by application-specific settings given on input. To validate the results, we have extended the default result output to be converted into a peptide-sequence-object (PSM).

The goal of the validation is to ensure that Cassiopeia produces the same output as Andromeda under the same conditions. If the algorithm is implemented correctly and the same results are achieved, there is the possibility of improvement with regards to performance. Although the stand-alone product of Max Quant has tested results, the computing technology keeps improving. We generally followed the design patterns used in Andromeda but also made some improvements, which we considered beneficial. A diagram of the components can be seen in [Figure 3.4](#). The structure present is almost similar, however there are 2 noticeable differences - an in-memory database has been used to store the peptides, and there are no filters on the output. Instead of creating indexes and writing into files we the sequence knowledge base is stored in memory. The spectrum iterator works with one additional file format - MGF, which is the default standard used by MStream.

Additionally, we have skipped the filtration of the output and directly give back a PSM object. Those modifications are implemented in this way because of the real goal of this work is to provide an additional scoring mechanism to MStream. Therefore we have kept only the crucial components, which benefit the cloud peptide identification tool.

3.2.2 Cassiopeia Scoring

The goal of the scoring is to provide correct results with good performance and be easy to integrate into the existing environment of MStream. For the scoring function, we have stripped away the components dealing with batch data and the construction of a peptide sequence database, resulting in a minimal set of elements, seen in [Figure 3.5](#). The only elements present are the scoring function, which evaluates the possibility of a match between a spectrum and a peptide, supplied on input while considering the set settings.

MStream does not rely on having the complete experiment batch beforehand. It works with streams of spectra, following the fast data concept. Therefore for it is best to implement the scoring mechanism as a function, which can be used on tuples composed by a single spectrum and a theoretical peptide.

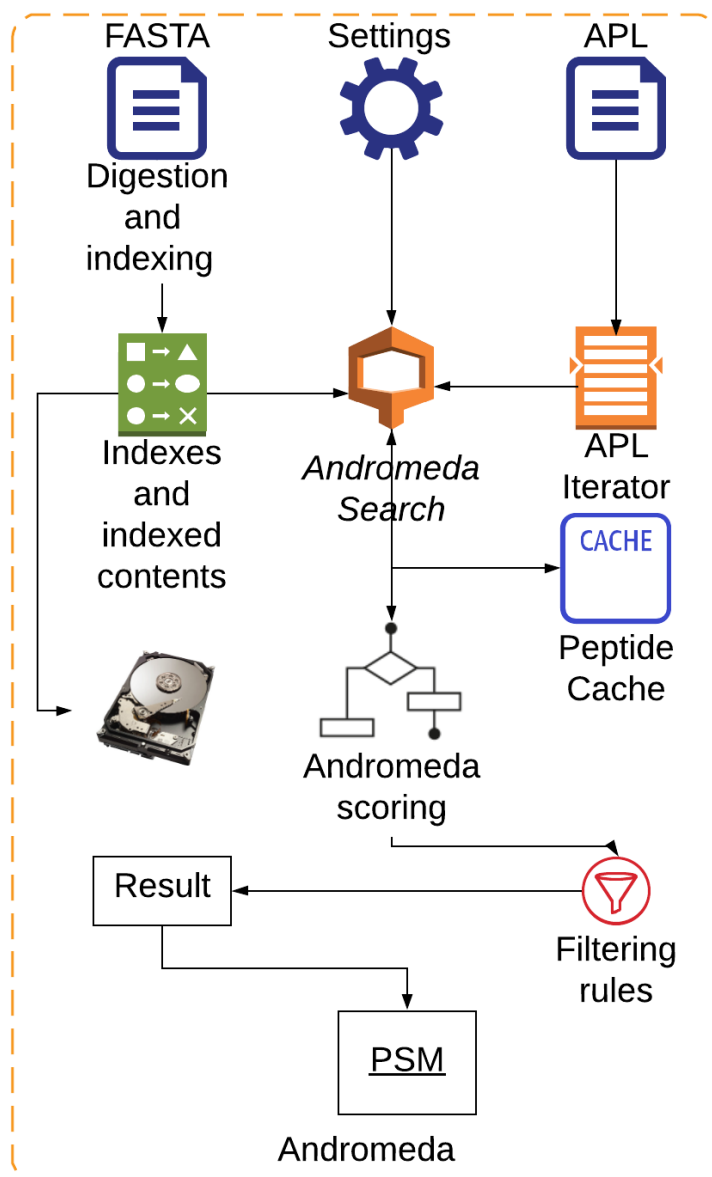


Figure 3.3: Andromeda stand-alone components. We can see the input batches for proteins in the form of FASTA, which create the sequence database and store it on disk. The spectra are iterated sequentially and fed into the search function, using the Andromeda scoring algorithm. Commonly considered theoretical peptides are stored into a cache. The end output of a PSM object is an extension by us to enable validation. One thing to note is the presence of a filter, controlling the resulting output.

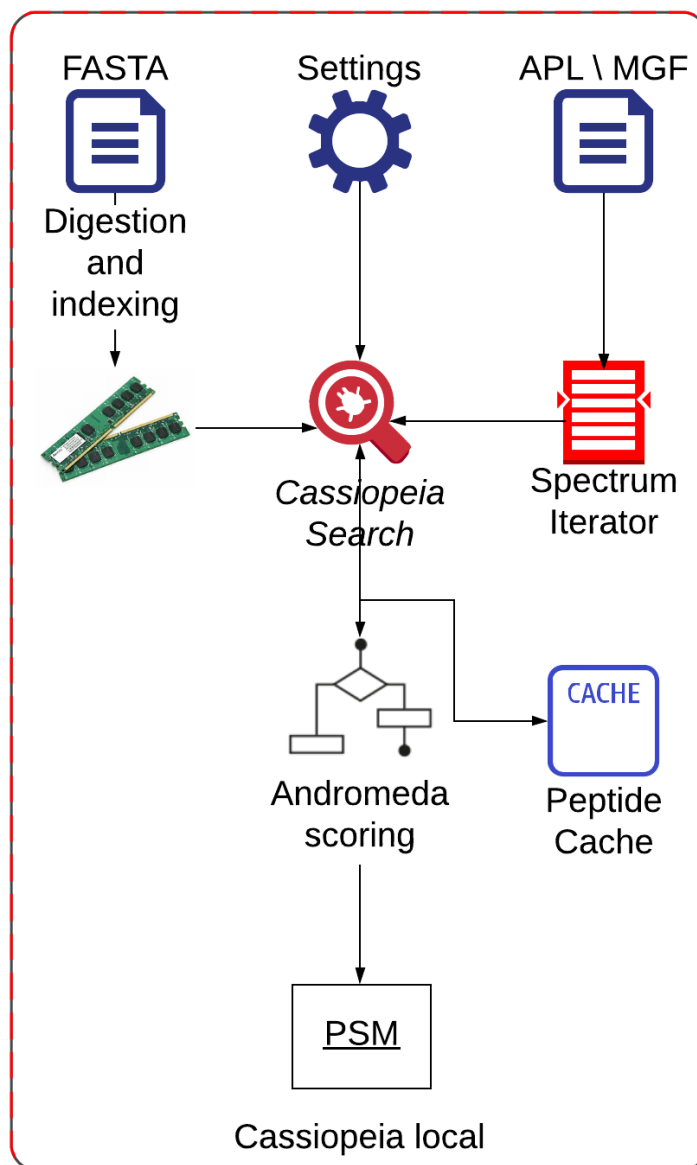


Figure 3.4: Components of Cassiopeia. The structure is based on the one present in Figure 3.3. Instead of storing the peptide sequence database onto hard disk, we use an in-memory database. The spectra can be either APL or MGF formats, and the settings given to the search function are the same as Andromeda's. During evaluation by Andromeda's algorithm, peptides can also be stored into a cache structure. The output is returned as a PSM object without filters.

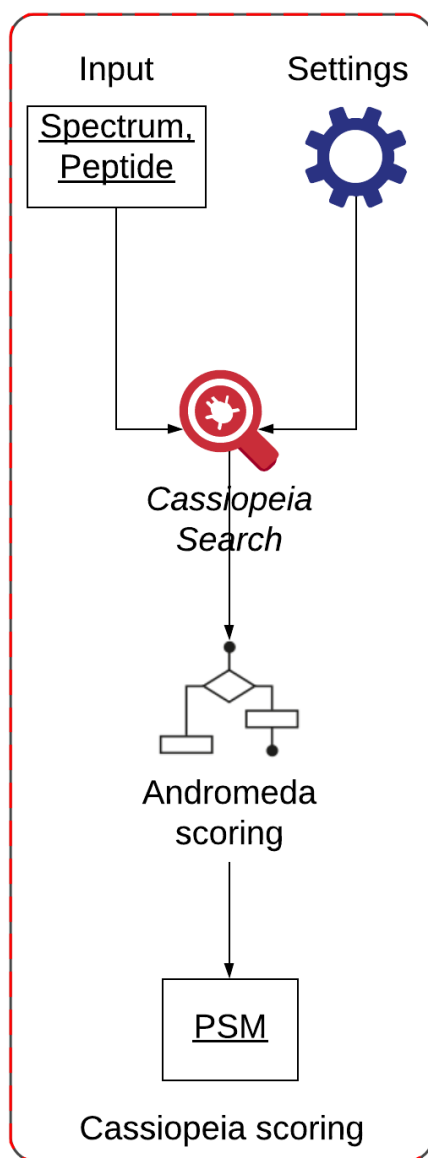


Figure 3.5: Cassiopeia scoring components. The present elements are a subset of those seen in Figure 3.4, where only the required for the MStream integration are kept. The input, consisting out of a spectrum and peptide, get passed along with application-specific settings to the scoring function. The evaluation uses Andromeda’s algorithm to produce a PSM object

In this way, for the validation, we can represent the mass spectrometry batch results as stream and reuse the underlying scoring functionality for stream operations.

In [Figure 3.2](#) the protein search engines communicate with a peptide sequence knowledge base to obtain suitable possible matches. While the spectrum is supplied by a mass spectrometer figuring out a good theoretical peptide sequence needs to be computed. In MStream this is done by a functionality, retrieving peptide sequences with similar properties to the peak given as input from an existing peptide knowledge base. Andromeda builds its own protein sequence database, along with its derivatives and iterates all theoretical matches in a tolerable range of the spectrum attributes. We have covered the exact methodology in detail in [Chapter 4](#). In the end, scores are always derived from the tuples of the spectrum - theoretical peptide. Therefore it is easy to implement the scoring mechanism to work for both the validation against Andromeda and Cassiopeia's integration into MStream when we keep the scoring components decoupled. Accepting the same input also minimizes the areas where errors could occur, should there be any deviations in the results.

If the results of Cassiopeia are validated and the scoring function is able to derive good match evaluations based on the same input format used in MStream we can fit our tool into the architecture shown in [Figure 3.2](#). By using Scala as a programming language our scoring function can be wrapped in a service, playing the role of a job. Given that MStream uses Apache Spark ensures that our new protein identification functionality fits perfectly in the existing ecosystem. Since the system is distributed and located in the cloud, we have guaranteed horizontal scalability.

Having Cassiopeia scoring function hidden behind a standard scoring interface, containing a method accepting an input of a spectrum and a peptide, can also promote the usage of additional protein search engines, so long as they implement the interface. This will obfuscate the concrete implementation of our new tool. Ideally, it will provide better vertical scalability by utilizing the system resources better than the stand-alone software, designed for 32 bit systems.

3.2.3 Cassiopeia Integration

The purpose of integrating Cassiopeia into the SMACK stack of MStream is to provide flexibility for the evaluation and result validation of mass spectrometry data. Adding the new scoring methodology will influence the existing architecture of MStream by changing the normal flow of the following components, marked with red squares in [Figure 3.6](#). Introducing a second evaluation mechanism increases the capabilities of the cloud solution. The validation is also changed since now there is the option for cross-validating evaluated scores.

There are various choices and flows, presenting themselves by having one more tool available for validation. We have visualized some of them briefly in [Figure 3.7](#). The process always starts with the arrival of new spectra, which needs to be analyzed. Potential peptides are retrieved from the peptide sequence knowledge base and given as input further down the line. Based on the pre-configured settings, there are multiple approaches possible:

- Evaluate only with X!Tandem

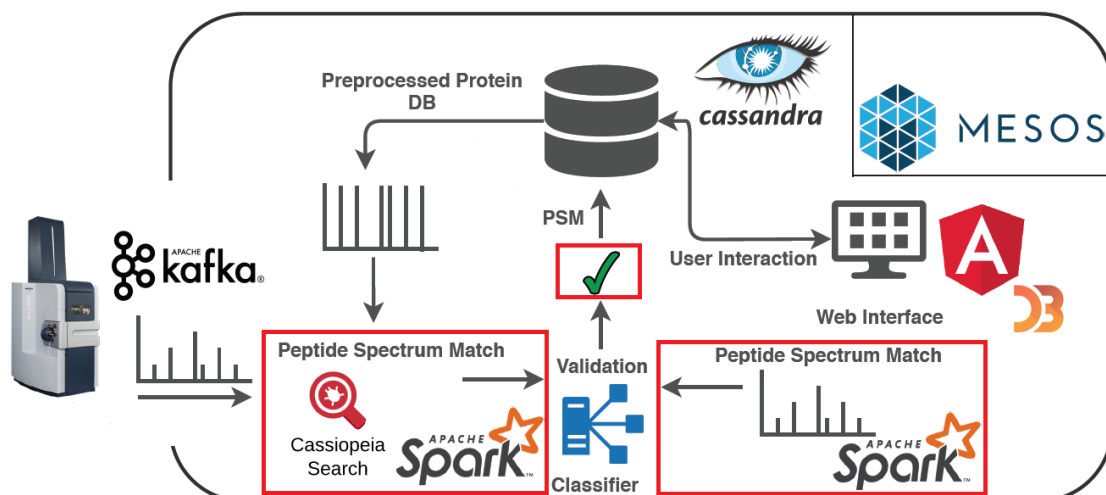


Figure 3.6: Influence of adding Cassiopeia into MStream’s SMACK environment. There are now two Peptide Spectrum Match components, one indicating Cassiopeia and the other - the default X!Tandem implementation or any other integrated peptide identification software. The validation is also marked in red as the scores of multiple searching tools can be used for cross-validation.

- Evaluate only with Cassiopeia
- Evaluate with X!Tandem and if good scores are obtained perform cross-validation against Cassiopeia
- Evaluate with Cassiopeia and decide based on high evaluated PSMs to perform cross-validation on top scoring peptides
- Run both X!Tandem and Cassiopeia, filter to common peptides, select top N and validate the results
- Other configurations

All those options enable flexibility in dealing with different data sets. Depending on the scores produced and the preconditions present the flow of the search algorithm can be adjusted accordingly. This allows for optimization of the scalability and the utilization of resources. The different tools available also allow for performing different analysis while also increasing the confidence in the produced results, should a PSM be ranked high.

3.3 Summary

In this chapter we presented a concept for the implementation of a peptide scoring mechanism, which produces similar results as Andromeda. We have displayed the individual components and what needs to be present for having correct results. We also proposed improvements to the architecture by utilizing the increased performance capabilities of state of the art technology. We also stripped down parts of the

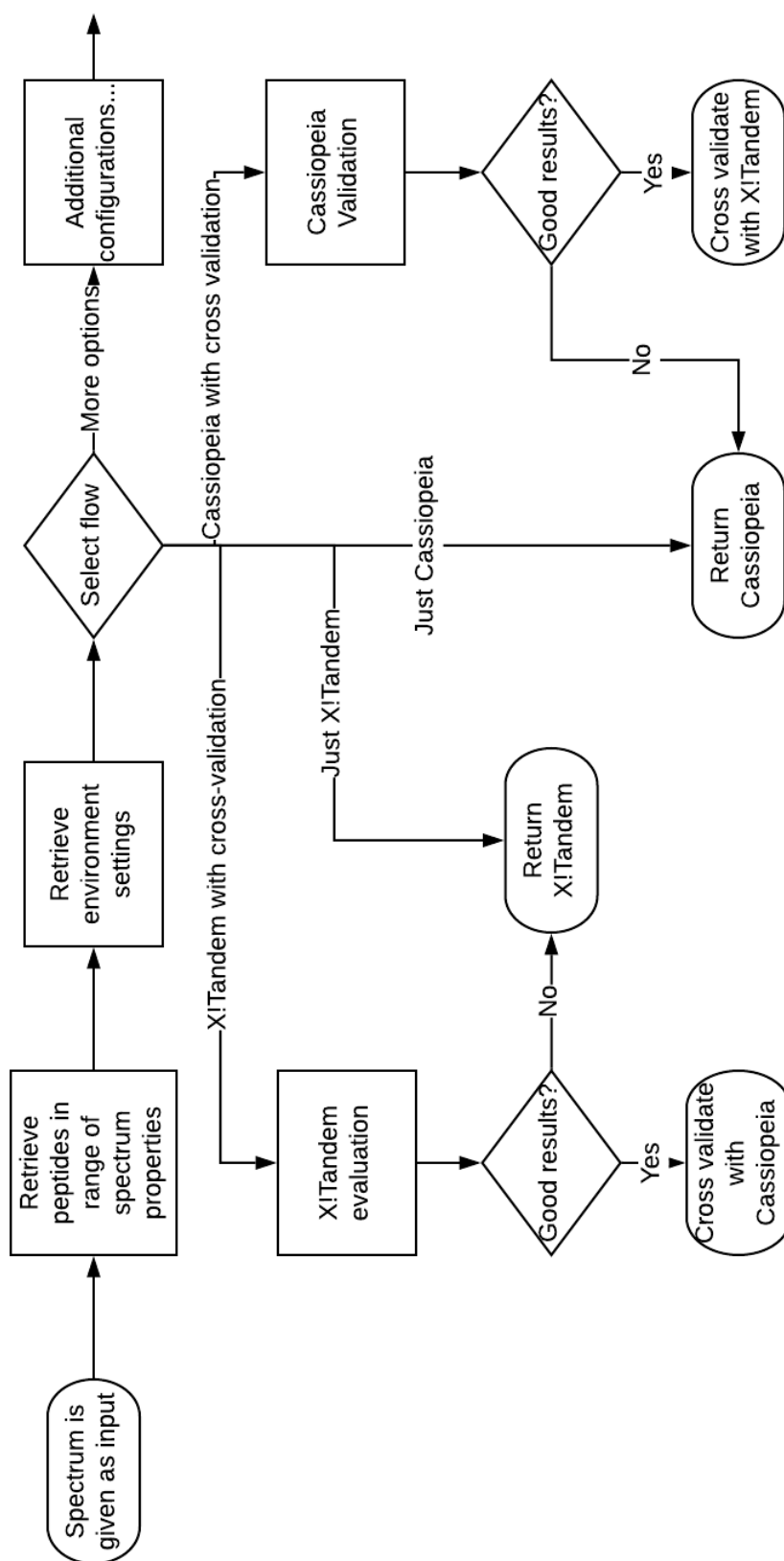


Figure 3.7: Flow chart for MStream with integrated Cassiopeia, showing possible evaluation configuration

original Andromeda software, which are not beneficial for work in a cloud environment. The data manipulation was also adjusted to fit into the fast data architecture environment in the face of working with stream data and not using batch files as input. By abstracting the scoring mechanism, we intend to produce a solution, which can be easily integrated into individual jobs in a cloud environment, maximizing the utilization of available resources. In the next chapter, we delve deeper into the concrete implementations for the validation against Andromeda and the integration of our new scoring mechanism into the existing cloud architecture.

4. Implementation

In the previous chapter we covered the main idea behind protein identification and the challenges of state-of-the-art protein identification software. We presented a concept which builds upon the Andromeda protein search engine and integrates its scoring methodology in the MStream environment. The idea of our new tool, *Cassiopeia*, is to use Andromeda's scoring and use it as a cross-validation mechanism for the already present peptide evaluation on a cloud environment, presented in the work of Zoun et. al. The goal is to increase the confidence of results by validating the output while also processing mass spectrometry experiment data as a stream in a near-real time fashion.

In the [Chapter 3](#) we presented the concept for Cassiopeia and how it can be used to validate the correctness of peptide evaluations. We have mentioned that before integrating it in the existing ecosystem of MStream the first challenge to overcome is to prove the usefulness of our new scoring mechanism. In this chapter we will offer concrete implementations for evaluating the possible match between a theoretical peptide, retrieved from a protein sequence knowledge base and a spectrum, supplied from a mass-spectrometry experiment. Then we will show how we have compared Cassiopeia's results against the output of Andromeda.

4.1 Cassiopeia technology stack

We have decided to use Scala as a programming language to implement Cassiopeia. This decision was made based on the existing technology stack used in MStream, seen in [Figure 3.1](#). In contrast to Andromeda .NET implementation it can be easier to expose the scoring function with Scala since it is naively supported by Apache Spark and there the overhead of wrapping the Andromeda scoring can be mitigated.

As described in [Chapter 2](#) Andromeda processes a protein sequence knowledge base by iterating the proteins and breaking them down by recreating the theoretical environment of the mass spectrometry experiment. MStream has a Cassandra database, containing a peptide sequence knowledge base. In both solutions they are used to

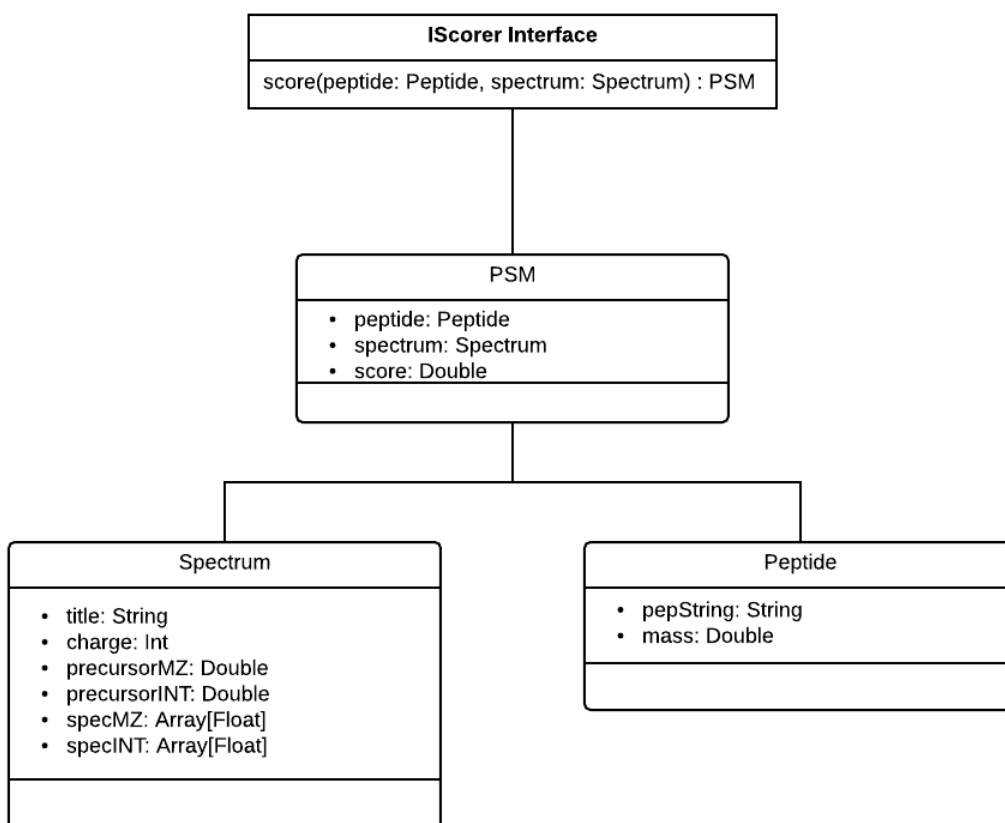


Figure 4.1: The IScorer interface. It produces a PSM object, which encompasses the input object and adds an attribute for the result score. This is the interface used to integrate Cassiopeia into MStream.

evaluate a potential match for a spectrum. Based on that fact we have developed our scoring function as an implementation of the interface seen in Figure 4.1.

In the figure we can see three distinct objects and with the following usages:

- *PSM* - short for peptide-spectrum-match. This is the output used in MStream [RZS18] and the output we use in Cassiopeia. It is composed out of the input theoretical peptide, mass spectrometry spectrum and a value indicating the probability for a match.
- *Spectrum* - the digitized information of a mass-spectrometry output. It encodes the information of a single spectrum (Figure 2.1). The information contained is a *title*, providing information regarding the protein, from which the peptide is derived. There are also properties for the peptide *charge*, the mass of the peptide encoded in the *precursorINT*. The *precursorINT* denotes the intensity for the whole peptide. The two arrays *specMZ* and *specINT* are used to encode the 2 dimensions of each individual peak, present in the spectrum (in Figure 2.1 for example each bin corresponds to one peak).
- *Peptide* - this object encapsulates the peptide amino acid sequence as a chain of amino acids, each encoded with their corresponding characters (Table 2.1). The mass is also present, where it could be unmodified - sum of the default masses of all present amino acids or could have a different value, depending on the mass spectrometry set up.

The objects used by Andromeda are a bit different than the ones presented in Figure 4.1, where the difference is only in the naming and the data type of individual peaks (Double is used instead of Float). Hiding the functionality behind an interface however simplifies the communication with external services and allows interchangeable usage of different concrete implementation of said interface. Another thing worth noting is that the mass spectrometry data is encoded in an APL (Listing 2.2) format. The concrete implementation of the IScorer interface and the adjustment of formats is represented in 1. In it we can see a method, which calls the IScorer interface but accepts classes named *WorkingPeptide* and *WorkingSpectrum*. These objects are used to note the differences in the validation and MStream calling the IScorer *Score* method.

Algorithm 1 Cassiopeia Scoring

```

1: procedure SCORE(workingPeptide : workingPeptide, workingSpectrum :
   WorkingSpectrum)
2:   peptide  $\leftarrow$  CassiopeiaPeptide.Map(peptide)
3:   spectrum  $\leftarrow$  CassiopeiaSpectrum.Map(workingSpectrum)
4:   score  $\leftarrow$  IScorer.score(peptide, spectrum)
5:   return score
6: end procedure

```

4.2 Cassiopeia architecture implementation

Cassiopeia is completely based on the Andromeda .NET stand alone software. This means we have researched and adapted the original probabilistic algorithm found in Andromeda. The configuration parameters present in Andromeda are kept the same in order to keep to integrity of the algorithm. There are two distinguishing changes to the original stand alone peptide search engine and those are the usage of Scala with Apache Spark context and the addition of concurrent data processing for the evaluation purpose. Another smaller change, which is worth mentioning is the usage of in-memory database to store the protein sequence knowledge base, but this plays a role only for the validation phase. The scoring is implemented in a way which can be used interchangeably for validation and the integration as an protein search engine in a cloud environment. A concept mock-up can be seen in [Figure 4.2](#).

In the picture it the scoring function can be called by MStream or our validation implementation. Since we covered the basics of MStream implementation in [Chapter 3](#) we will focus on the inner workings of the validation approach and how the actual peptide is evaluated. The difference between the validation implementation and the cloud one is that the first one works not with just a single pair, but a whole data set of proteins. In the *Cassiopeia Validation* part of [Figure 4.2](#) we have a spectrum iterator, which supplies the spectrum and a peptide database, which is store in memory. The Andromeda software is building a knowledge base and indexing known peptides, then storing them in files. This keeps track of already know amino acid sequences but in the end relies on first having the data. The software expects the complete information on the peak data set and known protein database.

4.2.1 Creating peptide in-memory knowledge base

In [Figure 4.2](#) we see the two different software architectures, which will with Cassiopeia's scoring function. Although the technologies are different they fulfill the same roles. A comparison of the components can be seen in [Table 4.1](#). We can wrap the functionality into four distinct tasks, which need to be done in order to evaluate if the peptide and the spectrum match. We have two task which are used for data preparation and forward it as input to the scoring function. Since the scoring will adhere to [Figure 4.1](#) we need a spectrum and a peptide to retrieve and give as input. The peptides are located in a peptide knowledge base, where they are ordered by mass to have an optimal search structure. Although the information is stored in different manners, the idea is the same - to provide peptides, which masses resemble a spectrum the most. This spectrum is supplied from the mass spectrometry example result. In the context of MStream it is as a stream of data, published by a Kafka producer. Andromeda requires the whole batch to be available before evaluation, so to validate the correctness we iterate over each spectrum.

The peptide database is build on the fly for Cassiopeia. In Andromeda once created, it can be reused, due to the data being stored in intermediate files. We covered in [Chapter 2](#) the details of how the sequence database is created. For the validation of Cassiopeia we have followed the approach of building a knowledge base out of a FASTA file, containing the protein sequences. Instead of writing into files we have stored the information in-memory. Andromeda uses indexes, pointing to the

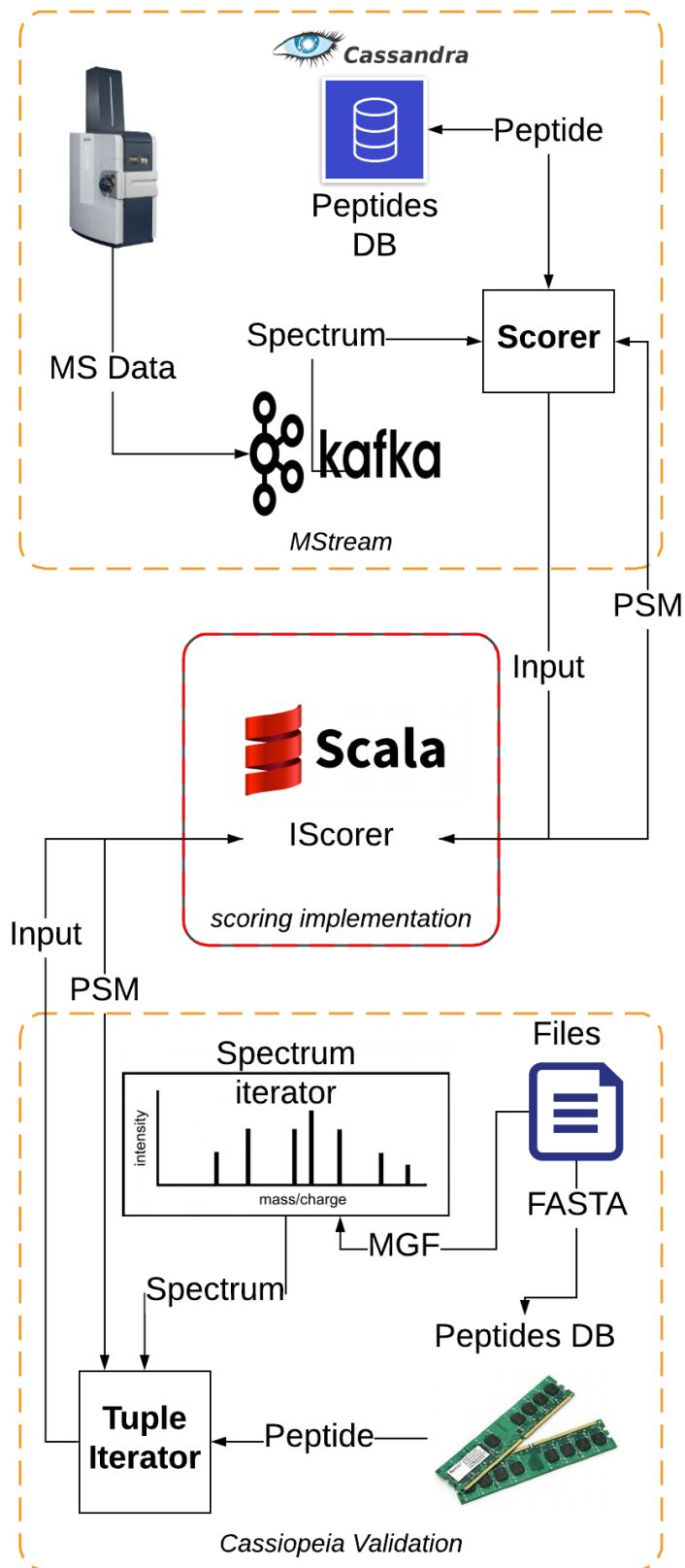


Figure 4.2: Usage of IScorer scoring functionality for MStream and for the validation of Cassiopeia against Andromeda

Task	MStream	Cassiopeia validation
Retrieving peptides	Cassandra database	In memory database
Retrieving spectrum	Kafka message	Iterating over input spectra
Calling the scoring function	Scorer Object	Scoring Facade
Output	PSM	PSM

Table 4.1: Components roles and implementations between MStream and Cassiopeia’s validation

locations of entries, stored on disk. Due to the significant differences between I/O operations of the two types of memory, having it on the RAM omits the need to go look up the information on the hard disk drive and materialize the object. Each entry in the FASTA data is loaded as a resilient distributed dataset (RDD). We use a Spark Context in Cassiopeia so we can work with datasets, allowing efficient processes. Each protein sequence is inspected and peptides are derived out of it, based on the predefined conditions, such as the digestive enzyme used, the fragmentation rules or additional settings. Like Andromeda, Cassiopeia has a pre-configured knowledge base, ordered by mass value so searches will take at most $O(N \log N)$ time, when we use binary search. MStream, Andromeda and Cassiopeia keep the peptides as an ordered database, where the mass is the key and the value is the theoretical peptide.

4.2.2 Scoring algorithm

We have encapsulated the scoring algorithm under an interface (Figure 4.1) and the overview can be seen in Algorithm 1. This is to follow the separation of concern and to follow the best practices to program against interfaces and not against concrete implementations. This makes the changing of strategy scoring algorithms quite easy, particularly for the cloud environment, where we might want to cross-validate the evaluation of each peptide. The usage of Interface structures is also very flexible, allowing to construct a Strategy pattern [GHJV95] for different classes, implementing the IScorer interface.

4.2.2.1 Scoring for the validation

For the validation we first process the protein sequence knowledge, store it into memory and then we begin iterating over the available spectrum data. The peaks are in MGF format and get translated into a Spectrum object, seen as one of the inputs in Figure 4.1. For each spectrum an according peptide is searched and this is done by looking into the sequence database for peptide which are in tolerable range from the spectrum mass. This is the purpose of having a database ordered by the mass values of peptides, since we can just take a range of peptides. The maximum tolerance for the deviation from the spectrum’s mass is decided by input settings, which we will go over in the next section. In the end having a spectrum we iterate over peptides which are bigger or equal than the spectrum mass minus the tolerable deviation allowed for the mass. The function can be seen in Listing 4.1, where we first check what type of unit we have configured for the tolerance used and then return the according value. When we have narrowed down a spectrum and a couple of peptides, which could be a fitting match we begin the scoring process.

```
1 function Tolerance deltaMass(input: Double) : Double
2     if (unit == Dalton) {
3         return value
4     } else {
5         return value * (0.000001 * input)
6     }
7 }
```

Listing 4.1: Calculate delta of spectrum mass

Additionally, as described in [Chapter 3](#) we have also implemented a cache structure by using a Map object. The key of this map is the sequence of a peptide and the contents of this collection are updated each time a new peptide is being passed down for evaluation. The intent is to minimize the calculations when calculating the individual properties and possible splits of peptides.


4.2.2.2 Scoring implementation

The actual score calculation is done by using the Andromeda formula ([Figure 4.3](#)). The formula described works with a spectrum object and analyzes the present peaks in a spectrum. We consider the peaks in chunks, as we split the whole spectrum into ranges and then perform an optimization, where we look for peaks which may correspond to a similarity present in a peptide chain structure. The peptide is decomposed in an array which has the same dimension n as the spectrum. This is done based on the fragmentation type used in the experiment to cleave the peptide sequence. In general we recreate the theoretical processes, which would have happened during the mass spectrometry experiment and measure how similar results we have. The less deviation there is from the theoretical peptide used as a reference the better the resulting score.


In Cassiopeia the concrete implementation for both validation and the one used as a cross-validation mechanism in MStream. The difference is only in handling the environmental variables and some pre-processing, but in the end both implementation reuse the same scoring. The *IScorer* interface accepts *peptide* and *spectrum* objects and uses the common scoring method *getBestN()*, which is the same for both implementations and uses the formula described in [Figure 4.3](#). A diagram for the method may be seen in [Figure 4.4](#). The input parameters in [Figure 4.4](#) are spread for the sake of clarity. We will go over each individual parameter as follows.

- peptide: input peptide
- spectrum: input Spectrum
- tolerance: this is a Tolerance object which is composed by a tolerance value and tolerance unit. This unit may be *Dalton* or *PPM*. It is used to calculate the acceptable deviation of a value used to decide whether a amount is in tolerable range. This is defined by the user and is an input parameter. The difference between the two is that for the tolerance *PPM* the delta is calculated as $toleranceValue \times (input \times 10^{-6})$ where the *toleranceValue* is the configurable deviation of the tolerance and the input is the given mass for which the acceptable fluctuation is calculated


n = total number of theoretical ions
 k = number of matching ions in spectrum


 Approx. probability of getting at least k matches by chance

$$s(q, \text{loss}) = -10 \log_{10} \sum_{j=k}^n \left[\binom{n}{j} \left(\frac{q}{100} \right)^j \left(1 - \frac{q}{100} \right)^{n-j} \right]$$


 Optimize inclusion of losses

$$s(q) = \max_{\text{loss} = \text{true/false}} s(q, \text{loss})$$


 Optimize q (peaks per 100 Da)

$$s = \max_{p \geq 2} s(q)$$

Figure 4.3: Formula used to calculate the probability of a match. It is the same one used in Andromeda [CNM⁺11]. The scoring involves an optimization of the number of highest intensity peaks in a Spectrum. The peaks are considered in ranges of 100 Dalton units in the m/z interval and over the inclusion of modification-specific neutral losses.

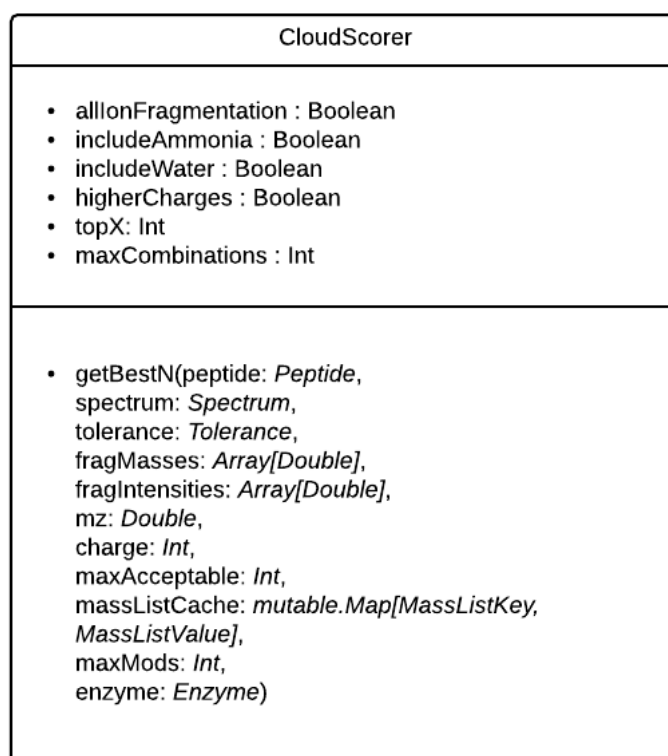


Figure 4.4: Cloud Scorer

- fragMasses: the fragmented masses from the spectrum object
- fragIntensities: the intensities from the spectrum object
- mz: the m/z ratio of the precursor
- charge: the charge of the peak
- maxAcceptable: number of maximum acceptable scores (refactor array peptides)
- massListCache: a mutable map used to store already found peptides for future reference. Should a peptide with the same properties is peptide property calculations are skipped and direktly utilized from the map
- maxMods: maximum number of modifications
- enzyme: enzyme used to cleave the peptide, user configurable. In our case we used *Trypsin*

As seen in 2 initially we set up the ScoringResult object (Figure 4.5). It is composed of arrays because we have used this object to represent the score results for the validation and cloud solutions. The scores contain the evaluations of individual input peptides. The sequences contain the amino acid sequences. The theoretical masses are obtained by calculating the mono isotopic masses from the peptide string.

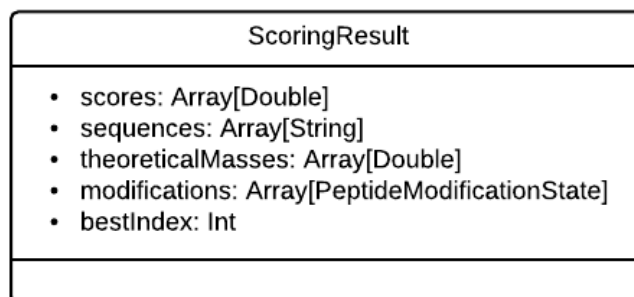


Figure 4.5: Scoring result class. The array have the size of how much peptide are given as input, in the cloud variant this array will be the size of one.

The modifications array represent the modification state of each peptide. These are influenced by the presence of fixed and variable modifications. The *bestIndex* shows at which index the score is highest.

Another initialization step is to configure the tolerance used to calculate the maximum deviation from a mass value. Tolerance values are defined as input parameters and the types of those values are covered in [Chapter 2](#). We also set up additional parameters: the maximum scores to be considered : \mathcal{C} in our and Andromeda's case, the presence of water, amonia and higher charges. We split the peptide into smaller peptides by applying the enzyme *Trypsin* rules on the amino acid chain. The peptide string get split accordingly and the resulting sub-peptides are stored into a list. We use this as our new array of peptides to work with.

After calculating the tolerance the only step left before the actual scoring is the application of any present modifications. In the cloud variant of Cassiopeia there are none but the option to use them is there. Should the modification count be greater than the maximum allowed the execution is broken and the next peptide may be cleaved. The maximum is an environment variable, configurable by the user. If we have an acceptable modification count we continue by giving the information to the next step, which works with the sub-peptide array. Having performed all those preparation steps the only thing left is to evaluate how likely the peptide is to be corresponding to the input spectrum.

Algorithm 2 Get Best Scores Cassiopeia cloud environment

```

procedure GETBESTN
    scoringResult ← ScoringResult.init()
3:   fragmentationType ← HcdFragmentationType()
    initTolerance()
    Enzyme.getNCleave()
6:   applyModifications()
    calculateScore()
end procedure

```

4.2.2.3 Preparation for the true probability calculation

The next step further transforms the data and calculates the effect of the input parameters on the newly cleaved peptide array. The modifications, if present, adjust again the values of the peptide. Then for each peptide string a score is calculated. First we apply the fragmentation rules to each peptide (Algorithm 3). In our tool we have considered the usage of only HcdFragmentation, which is just a collection of rules, splitting the peptide into smaller subsets. This produces a new array of masses and is done by the method *getQueryMasses()*. The presence of water, ammonia, the state of the N- and C-terminus are considered and adjust each mono isotopic peptide mass. Having performed this step we can apply our formula and extract a score. We pass as parameters the cleaved and modified masses array originating from the input peptide. The spectrum masses and intensities are also inputs to this function. In 3 we call the *score* function three times, dependent on user input parameters. First is whether neutral loss is included. When this is set to false the other two calls are skipped. In the depiction of the algorithm we have displayed only two Boolean parameters as intake to keep the pseudo code clean. Those variables are *includeNeutralLoss* and *secondLoss*. These two parameters influence the calculation of mono-isotopic masses performed on the split by the Enzyme peptide chain. The altered masses are given as input to the scoring function. If there are neutral losses the results of the three possible scores is compared and the highest is returned. The precise calculation is explained in the next sub section.

The score function encapsulates in itself the application of the formula provided by the Andromeda software (Figure 4.3). To simplify the software first looks for matches in the cleaved array masses. We count the number of matches k found when comparing the cleaved peptide theoretical masses n and the peaks in the input spectrum. Higher k with regards to n represent better probability to have found a suitable matching amino acid chain. The n and k values are used to calculate the probability as -10 times the logarithm of the probability of matching at least k out of n theoretical masses by chance. Again, we use the same approach from [CNM⁺11]. We take this probability while accounting for the correlation between masses, intensities and the cleave position for found matches.

The calculated score is given back up the chain and assigned to the *ScoringResult* object, along with information, that can be used later for analysis, such as peak annotations and the role of modifications.

The *ScoringResult* is an intermediate object, which gets translated into a PSM object. This can be then given back as input, which fits both the validation concept and the one where Cassiopeia plays the role of an additional protein search engine, used as a cross-validation mechanism. In the next chapter we will go over the results we have obtained with our implementation.

4.3 Summary

In this chapter we covered the concrete implementation of Cassiopeia scoring and the architecture required for the integration into MStream. We have developed our tool to be decoupled so the underlying approach can be used not only in a cloud

Algorithm 3 Main Scoring

▷ The *score* function accepts always the same parameters, the only difference are the boolean values *includeNeutralLoss* and *secondLoss*.

```

procedure SCOREQMAIN(spectrumMasses, spectrumIntensities, ...)
    scoringResult ← ScoringResult.init()
    scoreFirst ← score(false, false)
4:   if ( hasNeutralLosses == false )
    return scoreFirst
    scoreSecond ← score(true, false)
    scoreThird ← score(true, true)
8:   if ( scoreFirst > scoreSecond & scoreFirst > scoreThird )
    return scoreFirst
    else if ( scoreSecond > scoreFirst & scoreSecond > scoreThird )
    return scoreSecond
12:  else
    return scoreThree
end procedure

```

environment but also easy to validate against results, produced by the Andromeda stand alone solution. Our goal is to obtain correct results in a timely fashion, while also adhering to the pre-existing SMACK stack architecture. We have implemented both validation and integration with the same components, where we use only the required for the use case. The goal is to produce a software, which produces the same evaluations as Andromeda and can be integrated with the IScorer interface into the MStream ecosystem. In the following chapter we will evaluate the performance of our implementation and how it behaves in a local or in a cloud environment.

5. Evaluation

Our tool Cassiopeia is an implementation of the Andromeda peptide scoring algorithm intended to be used as cross-validation mechanism in a cloud environment. Since Andromeda performs at least as good as Mascot, which is considered as the golden standard in the proteomic field, we believe Cassiopeia may be close in usefulness to the scientists working within the proteomic and metaproteomic domain. This will be true as long as our tool performs just as good as Andromeda. Therefore in this chapter, we first evaluate the correctness and performance of Cassiopeia compared to Andromeda. The second part of this chapter shows the scalability of Cassiopeia when we consider a local or a cloud environment. Lastly, we compare the cloud variant of our tool to an existing X!Tandem solution in the same environment, where Cassiopeia will be running, evaluated in [RZS18].

This chapter begins by describing the experiment environment and the data sets used as input and reference. Then we go over the different evaluations we performed against Andromeda and X!Tandem in a cloud environment and compare the results.

5.1 Experiment set up

We begin by firstly describing the data sets used to perform our evaluations. The Andromeda software uses *APL* format (Listing 2.2) and *MStream*, deployed on a *SMACK* environment, works with data in *MGF* format (Listing 2.1). The protein knowledge base is encoded in *FASTA* format (Listing 2.3). We have used peaks in *APL* format for local experiments, but Cassiopeia can work with both *APL* and *MGF* formats interchangeably. For the verification of our correctness with regards to the performance of Andromeda, we have used two different experiment situations - one for a data set with modifications and one without. For both of the cases, we have used the same *FASTA* file to generate our peptide sequence knowledge base: the *UPSP_Nov2017* data set, which was downloaded from the Uniprot web site [O:U]. It contains 560,414 distinct proteins, and the whole data set size is 270 MBs.

As for the mass spectrometry results data set, we have used two distinct spectrum data sets. For the case with no modification, we have used the *Ecoli_04_RD2_01_1275*

Parameter	Value
Enzyme	Trypsin
Peptide mass tolerance	10 PPM
Fragmentation mass tolerance	0.5 Dalton
Fragmentation type	HCD Fragmentation
Fixed modifications	None
Variable modifications (when applicable)	Methionine, Cysteine
Max missed cleavages	2
Modification position	Anywhere

Table 5.1: Experiment parameters for the experiments performed. The same parameters are used for Andromeda, Cassiopeia and MStream. Variable modifications are used only on *Ecoli_02_RD2_01_2199* for one experiment, otherwise none were used

data set, containing 40,738 individual spectra and with a size of 77 MBs. We have also performed tests on different sizes of data to observe the performance influence. For the evaluation of dealing with data sets with modifications present, we have used a different spectrum data set: *Ecoli_02_RD2_01_2199*. It contains 27,759 distinct spectra and 110 MBs in size. For the evaluation of a cloud environment performance in the context of MStream, we compared the default scoring - X!Tandem and Cassiopeia. *UPSP_Nov2017* was also used as the initial peptide knowledge base, and spectra were obtained from *Ecoli_02_RD2_01_2199*.

We have also evaluated the performance of Cassiopeia in two environments: one local to compare the performance with the software used as an inspiration: Andromeda, and then we compare how our tool fares in the MStream cloud environment, compared to an Apache Spark job running the X!Tandem algorithm. We also compare the performance of Cassiopeia when ran on a local set up or one in the cloud. We wish to make note that we have two separate implementations - one that accepts a FASTA file to create its own peptide knowledge base and a second one, which accepts a peak entry and a peptide to be evaluated. This is covered more in detail in [Chapter 4](#) and [Chapter 3](#).

Throughout the experiments, we have reused the same experiment variables, which roles are covered in [Chapter 2](#), can be seen in [Table 5.1](#). We have used the same peptide mass tolerance when creating the peptide sequence database and the same fragmentation mass tolerance for the matching of spectra against theoretical spectra. The fragmentation type used to cleave the proteins and peptides is the same for all three different environments - HCD fragmentation and allowing a maximum of two cleavages. There are no fixed modification present, only variable modifications for the evaluation of performance on a set with variable modifications and in the MStream environment. Additionally, the modification may be present anywhere in the peptide string.

For the local experiment, we performed the evaluation on a personal laptop. The machine is a Dell Latitude 5580 with an *Intel Core i7-7820HQ* processor, 32 GBs of RAM with 2666 MHz frequency and a Toshiba NVMe2 512GB.

Modification	Andromeda representation	Composition	Delta mass	Site
Cysteine	Carbamidomethyl (C)	C_2H_2NO	57.0214637236	C
Methionine	Oxidation (M)	O	15.9949146221	M

Table 5.2: Variable modifications used in Andromeda, Cassiopeia and MStream

For the cloud environment, we had a cluster which was running up to 4 Spark Jobs at a time, each job was running on a machine of 1 Core with 4 GBs of dedicated RAM. As experiment data for the evaluation of X!Tandem and Cassiopeia scores on a cloud environment we used *Ecoli_02_RD2_01_2199* set. The peptides used as input were supplied by our SMACK environment through a Kafka producer.

5.2 Validation Experiments

Our first task was to ensure the correctness of Cassiopeia scores. If we are not sure about the validity of our results, further comparisons will be meaningless. That is why we first have focused on validating whether our results match those produced by Andromeda. For this purpose, we begin with the evaluation of the result of both Andromeda and our proposed implementation on the exact same data set with the same parameters.

We have performed experiments for two different cases - one without modifications and one with two variable modifications. For the second case with we used the following variable modifications seen in Table 5.2. The modification table contains the type, composition, and the altered mass of the amino acid, which is modified. The allowed modification position is anywhere in the peptide. The modifications create two versions of the peptide - one with the original mass, calculated as a sum of all available amino acids and a second one, where the amino acids with the specified sites use the delta mass instead of their default one.

5.2.1 Experiment 1: Validation on sets without any modifications

Our first experiment was to see whether the results of both Andromeda and Cassiopeia will match on a local environment with a batch of spectrum data. We have performed four experiments in total for this, three with different sizes for an experiment without modifications and one experiment with modifications.

For the first experiment, we considered using only subsets of the total data. We created three distinguish sub-sets of both spectra and protein sequences files - one where we took only the first 10% of the datasets, a second where we considered the first half of the files and a third case for the complete files. The first evaluation was done on the correlation between the best-found peptides and their scores for first 10% of protein records found in UPSP_Nov2017 and evaluated the first 10% of the peak records in *Ecoli_04_RD2_01_1275* . There were no modifications given as an input. The comparison of the results can be seen in Figure 5.1. The small data set produces 63 total peptide match evaluations, where we have displayed only 40 for the sake of clarity. The y-axis is the result score for the match between

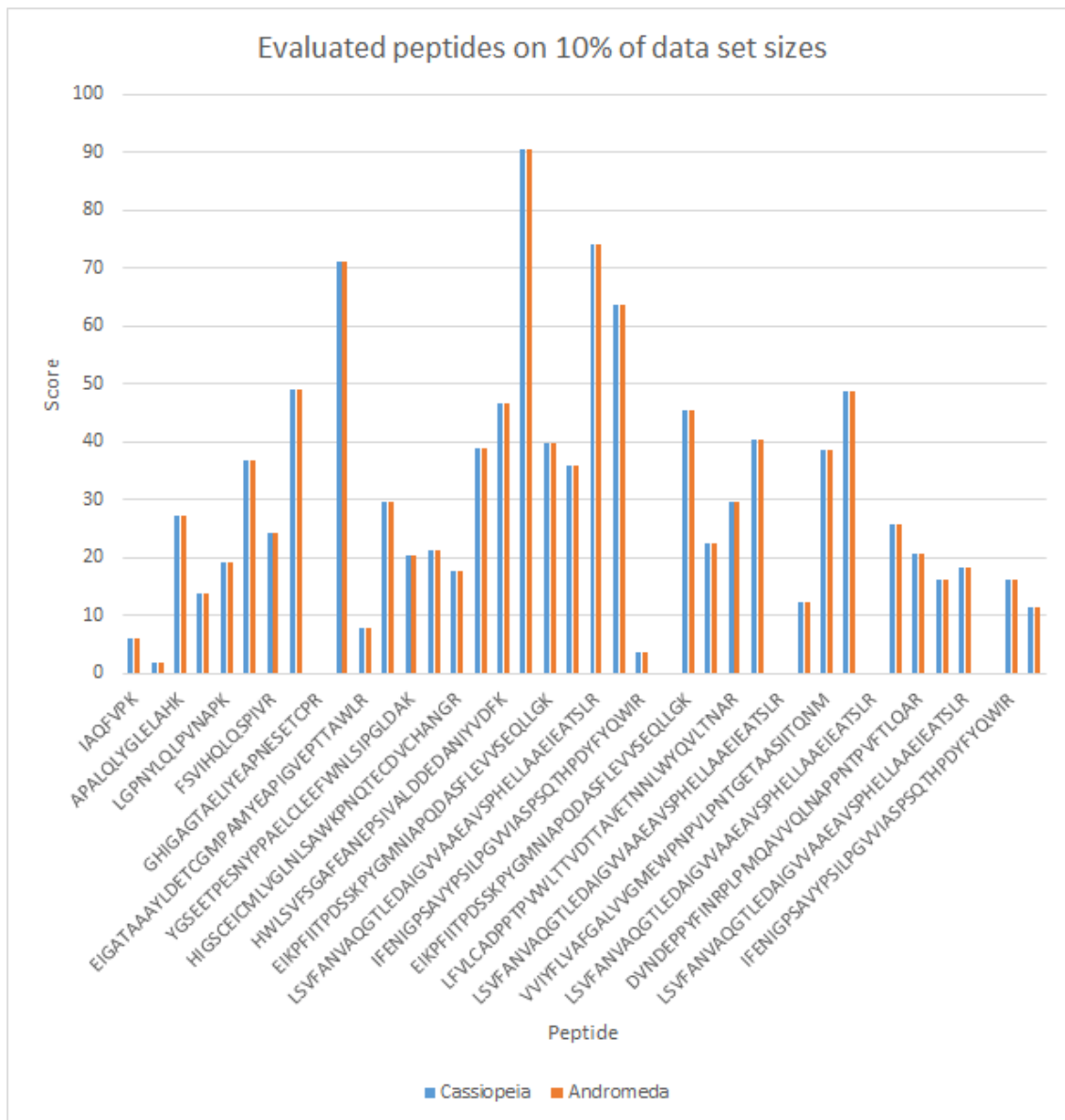


Figure 5.1: Correlation between Andromeda and Cassiopeia scores performed on 10% of the Ecoli_04_RD2_01_1275 data set. The graphs contains 40 random subsets out of the total 63 evaluated peptides, but the missing records behave in the same way. The y-axis displays the score given as output, and the x-axis displays the peptide sequence. Both Cassiopeia and Andromeda have produced the exact same result, therefore, the bins are the same height.

Occurrence number	Times encountered
1	36
2	7
3	4

Table 5.3: Number of times the same peptide has been identified as a match

the theoretical input peptide and the spectrum. The x-axis denotes the peptide sequence. We have plotted both scores as 2 bins with different colors - Cassiopeia (blue) and Andromeda (orange). Both scoring functions produce the same results and all bins are with equal height. This is also true for the 23 not displayed from all 63 distinct strings.

We also performed the same experiment on the complete data set. The ordering was not the same, but again the same unique peptides were identified, although there was one more evaluated PSM object. It resulted in 47 unique peptide sequences, and 64 total evaluated potential matches. Peptides appeared multiple times with different scores for different spectra, but at most three times. (Table 5.3) contains the distribution of how many times a peptide has been scored. We can still observe that the majority of the peptides are unique and only roughly one-quarter of the peptides were considered as a possible contents in more than one spectrum. Peptides, which occur more than once have, may also have different scores. A representation of the deviations in the score can be seen in Figure 5.2. The average of the scores is indicated by a blue dot. The minimum and maximum is displayed as a black diamond. The y-axis indicates the score value for the peptide. In this graph we have displayed the variation of a score with a gray line, starting at the minimum score and the maximum score achieved. There are 8 present gray lines, which means that 3 out of the 11 peptides, identified more than once have a large difference between the scores. The records which were repeated but did not show themselves in Figure 5.2 are at number 17, 21 and 26. For number 21 we have a peptide scored thrice as 0 for a potential match and for the other 2 results, they were identified by two distinct spectra, and the deviation is in range of 1.5 of their score. Their averages are 41.95 for peptide number 17 and 15.51 for peptide number 26. All other blue dots represent the average of the scores, where for single peptides the average is equal to the minimum and maximum score.

5.2.2 Experiment 2: Validation for data sets with modifications

After performing tests without including any modifications, we have run an additional one, including two variable modifications to the experiment environment: Cysteine and Methionine. There were differences in the number of peptides discovered in both Cassiopeia and Andromeda. An overview of the result can be seen in Table 5.4. For this experiment, the number of evaluated peptides differ - Andromeda has considered 882 in total, which is 78 less than those produced by Cassiopeia - 960. The common distinct peptides discovered are 109 in total, where Andromeda finds one more peptide additional. This means that we can boil down the sets down to around 110 peptides and that a peptide has occurred on average 9 times. Although

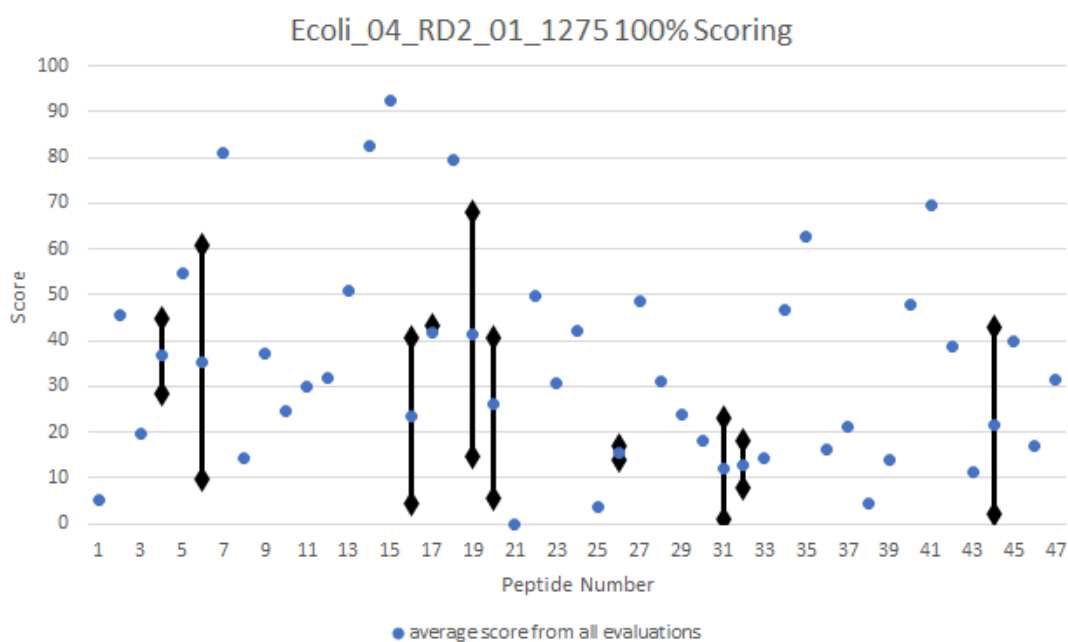


Figure 5.2: Evaluation for the whole Ecoli_04_RD2_01_1275 without modifications. The x-axis shows the index of the peptide, the y-axis is for the score value. The blue points indicate the average for each individual peptide. The gray line is present for peptides, which were considered as a potential match for more than one peak and had different score values, based on the spectrum considered. The gray line shows the range of the found scores, where the lowest point shows the minimum, the highest - the maximum and the blue dot displays the average of the scores assigned to the peptide.

Measurement	Cassiopeia	Andromeda
Total peptides evaluated	960	882
Distinct peptides count	109	110
Peptides missing	1	0

Table 5.4: Results of peptide search for Ecoli_04_RD2_01_1275 with Cysteine and Methionine as variable modifications.

Position	Cassiopeia	Andromeda	Difference
15	58.03	31.98	26.05
33	28.18	69.15	40.97
71	22.79	20.2	2.59
72	40.36	37.45	2.91

Table 5.5: Mismatched peptides between the maximum scores produced by Cassiopeia and Andromeda. We have 4 places in total where differences are present. Two of them produce scores with small difference, while the other 2 cases have a significant difference in favor of Cassiopeia.

having evaluated fewer peptides, Andromeda has evaluated one less. The missing key is a result of a peak which evaluates 14 different peptides in Andromeda and is 13th when ordered by produced score. The missing peptide also appears only once in this spectrum.

In [Figure 5.3](#) we have compared the maximum scores given by both protein identification tools for each distinct peptide. There are a total of 109 unique peptides displayed. We have made a side-by-side comparison for the peptides, which are produced as results and have plotted how they fare against each other. The x-axis is used to indicate the peptide index, and the y-axis show the value for the score. The orange line notes the maximum scores for peptides given by Andromeda and the blue line displays the maximum score achieved by Cassiopeia for the same peptide. The maximum value achieved is 77.48 by both software tools, and they also share 12 peptides with zero scores. We see again that Cassiopeia produces at least as high results, as those gathered by Andromeda. The blue line, representing Cassiopeia almost completely overlaps with the orange one, indicating Andromeda results. They are 4 peptides out of the 109 total, which have bigger value than their counterparts. Those different can be seen in [Table 5.5](#). We observe 2 distinct peptides to be evaluated quite higher than their counterpart in Andromeda - we have a difference of 40.97 higher result than one peptide evaluated with 28.18. The other two results deviate less, by 2.59 and 2.91, which is under 10% deviation for both cases. All other 105 peptides share the same maximum score assigned by both search tools.

In [Figure 5.4](#), we have plotted how many times each peptide is evaluated as a potential match. The figure again displays the shared by both scoring algorithms 109 distinct peptides and the correlation between their score values. We have plotted the number of evaluations for each peptide, found during the experiment run time. On the x-axis, we have displayed the index of the peptide up to the maximum of 109. In the y-axis, we have displayed the number of times a peptide has shown up during the evaluation. The occurrences for Cassiopeia are in blue and Andromeda is plotted

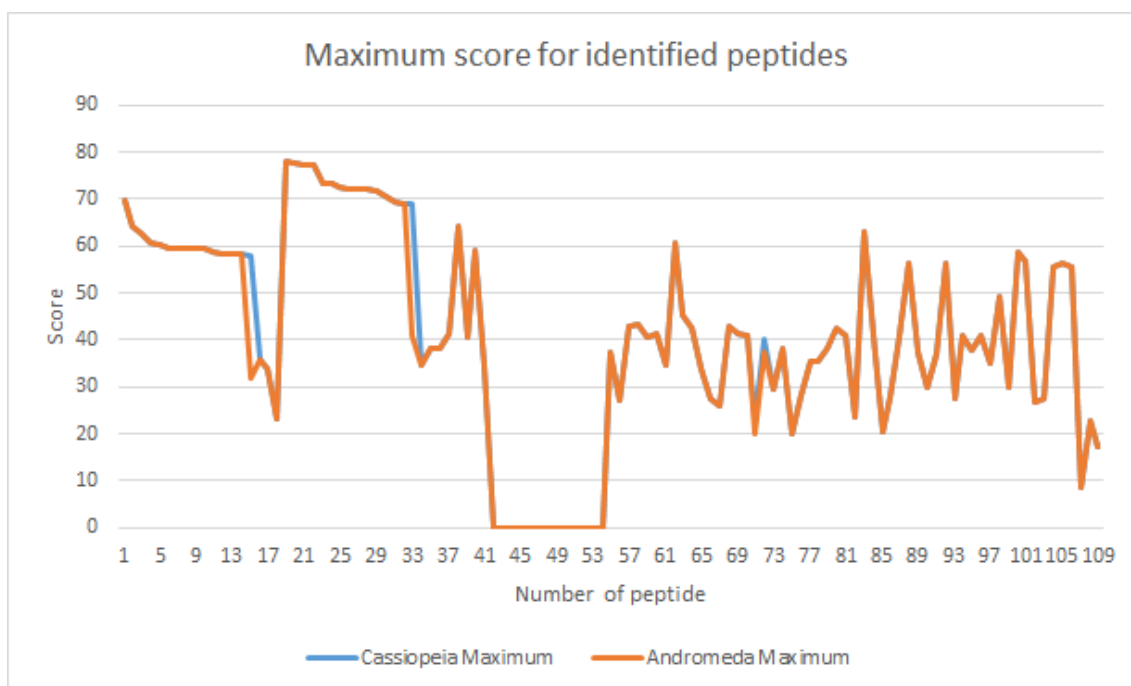


Figure 5.3: Comparison between the maximum scores of identified peptides from Cassiopeia and Andromeda with Methionine and Cysteine as variable modifications. The x-axis displays the 109 distinct peptides shared by both Cassiopeia and Andromeda, while the y-axis indicates the maximum score achieved by each individual peptide sequence.

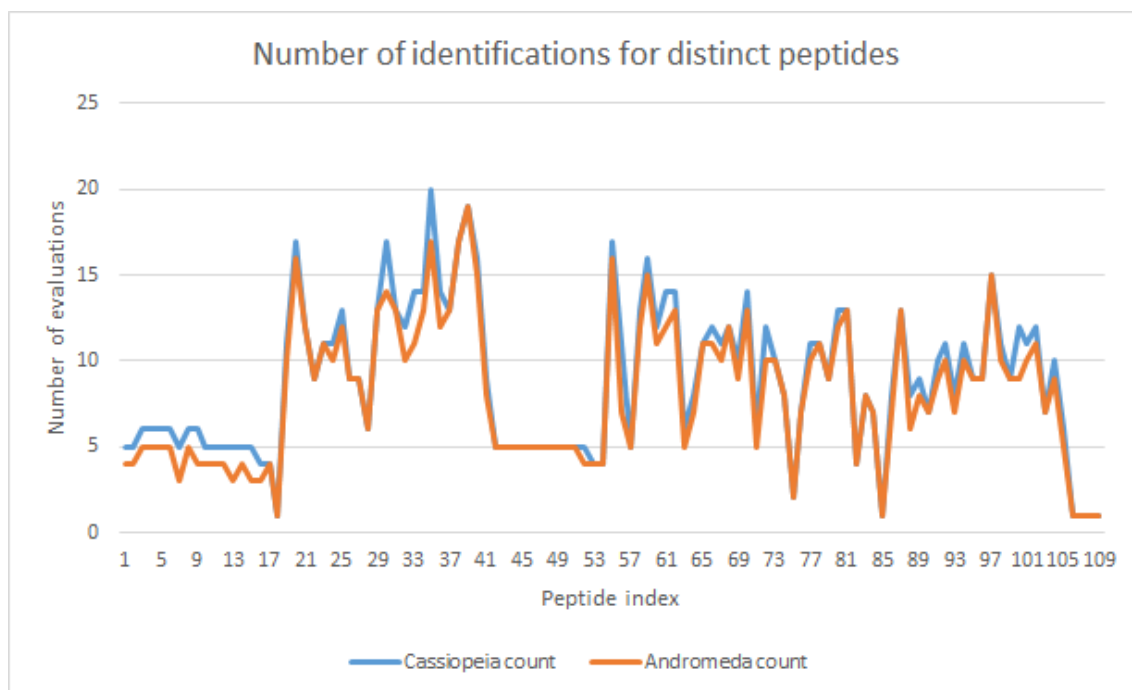


Figure 5.4: Comparison on the complete data set without modifications, displaying how many times distinct peptides have been considered as a match in Cassiopeia and in Andromeda. The y-axis displays how many times a sequence has been evaluated as a potential match. The blue line shows the result given by Cassiopeia, while the orange one is for Andromeda. The x-axis shows the index of the 109 shared by Cassiopeia and Andromeda peptides.

as orange. We see that the maximum amount a peptide has been shared is 20 and the least amount of occurrences is 1. The most interesting thing to observe is that Cassiopeia has at least as many evaluations when compared to Andromeda. There are some cases where Cassiopeia produces a larger quantity of results found, but in general, even if there are such, the difference between the two scoring mechanisms are no more than 4 per distinct peptide. This indicates how the Cassiopeia PSM evaluations manage to outnumber those, produced by Andromeda.

Since peptides in Cassiopeia are with roughly 10% more than those, evaluated by Andromeda we have also inspected the average results of each distinct sequence. In Figure 5.5 we have computed of the average for each one of the 109 distinct peptides. The x-axis denotes the index of the entry, and the y-axis displays the average score achieved. Cassiopeia (blue) almost overlaps with Andromeda (orange) for the bigger part of the graph. There are some deviations, however, especially for the first part of the figure. We also see an average of 0 given to peptide numbers between 41 and 55. This is also present in Figure 5.3, where the peptides again share a maximum score of zero. The fluctuations between the two lines reduce after the first 17 elements and are almost the same, apart for some differences.

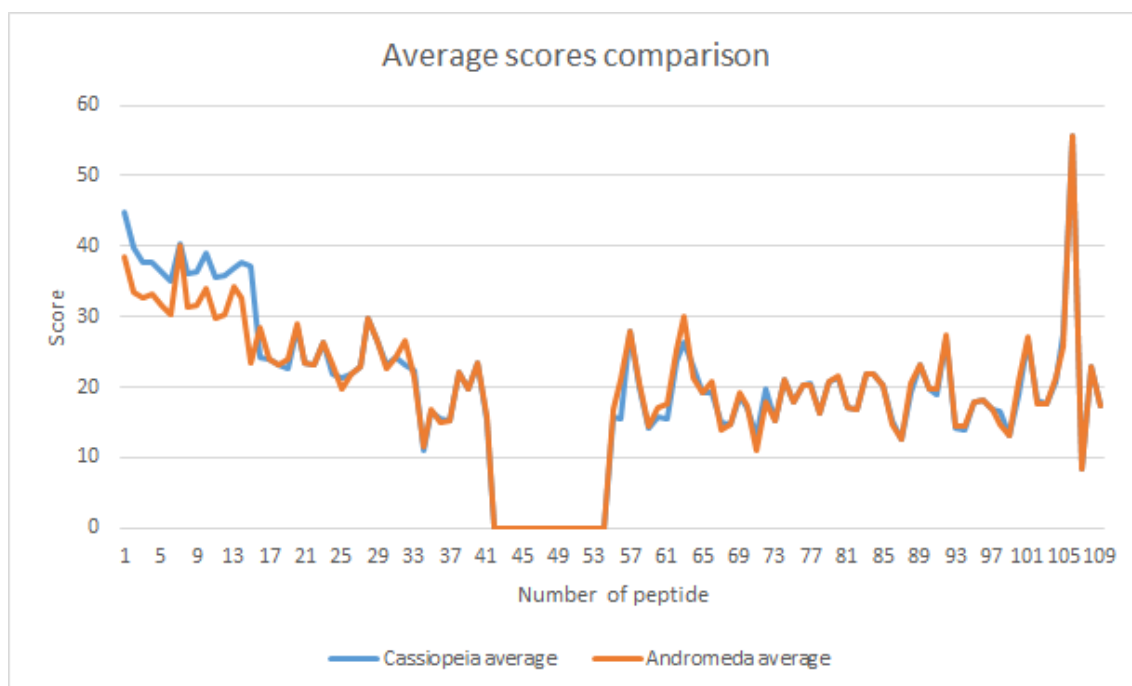


Figure 5.5: Average of all scores present for each distinct peptide, occurring both in Andromeda and Cassiopeia. The x-axis shows the indexes of the 109 shared by both Cassiopeia and Andromeda result evaluations. For each peptide the average is computed from the sum of all different scores for each individual peptide and divided by the number of times it has been evaluated as a potential match for more than one spectra. The resulting value is assigned to the y-axis, representing the score achieved.

Metrics	Cassiopeia	Andromeda
Pre-processing time	24,932 ms	107,517 ms
Search time	7,693 ms	6,151 ms
Total time	32,625 ms	113,668 ms

Table 5.6: Identification time in milliseconds for 10% of the complete proteins and peaks data sets

5.2.3 Experiment 3: Evaluation of performance on data sets with different sizes

After comparing the accuracy of Cassiopeia’s output against Andromeda, we also wanted to see the performance and scalability of our tool when working with data sets of different sizes. To validate the process, we have performed three separate tests on three different data sizes. The complete FASTA *UPSP_Nov2017* file takes 270 MBs of storage space and contains 560,414 individual proteins, which are considered in normal and reverse order, amounting to a total of 1,120,828 proteins. The peptides composed produced by these proteins, after applying the cleavage rules and omitting same peptides, end up being 13,413,201. The complete set of spectra *Ecoli_04_RD2_01_1275* is composed of 40,738 individual peaks and take 77 MBs of hard disk space.

Each graph below has been achieved by using the absolute same input parameters and settings. Since we evaluated the relation of the scores produced by a data set without modifications to produce the same results in Section 5.2.1 the graphs below do not focus on the individual scores but instead indicate the elapsed time. Therefore the y-axis is showing the time spent in milliseconds instead of indicating the score. To test the scalability of both approaches, we have used data sets of different sizes for both peptide knowledge base and the sizes of the spectra to be identified.

Figure 5.6 is performed with 10% of the total sizes for both FASTA and APL sets, amounting to 27 MBs in size and composed out of 107,176 individual proteins, cleaved into 1,593,078 individual peptides. For evaluation, we also used 10% of the spectrum data set, amounting to 3,898 individual entries. The graph displays the time required to build the peptide sequence database, labeled as "Preprocessing time", where Cassiopeia is shown in blue and Andromeda in orange. The y-axis displays the elapsed time in milliseconds. The search duration comparison, where the peaks are evaluated from the peptide knowledge base, is displayed as "Search time". The last bin tuple displays the total time required for the whole experiment. The values achieved by each peptide identification software for the three phases can be seen in Table 5.6. We can see that Cassiopeia finishes faster than Andromeda, needing 32,625 ms compared to 113,668 ms.

Figure 5.7 is done with 50% of the protein and spectra data sets. The y-axis is for the duration in milliseconds, and the colors are blue for Cassiopeia and orange for Andromeda. The data quantities amount to 20,658 individual peak entries with a protein sequence database containing 505,494 proteins, which in turn get digested into 6,744,717 peptides. The experiment duration is again grouped into three categories - pre-processing, evaluation, and total time spent. The scores of the

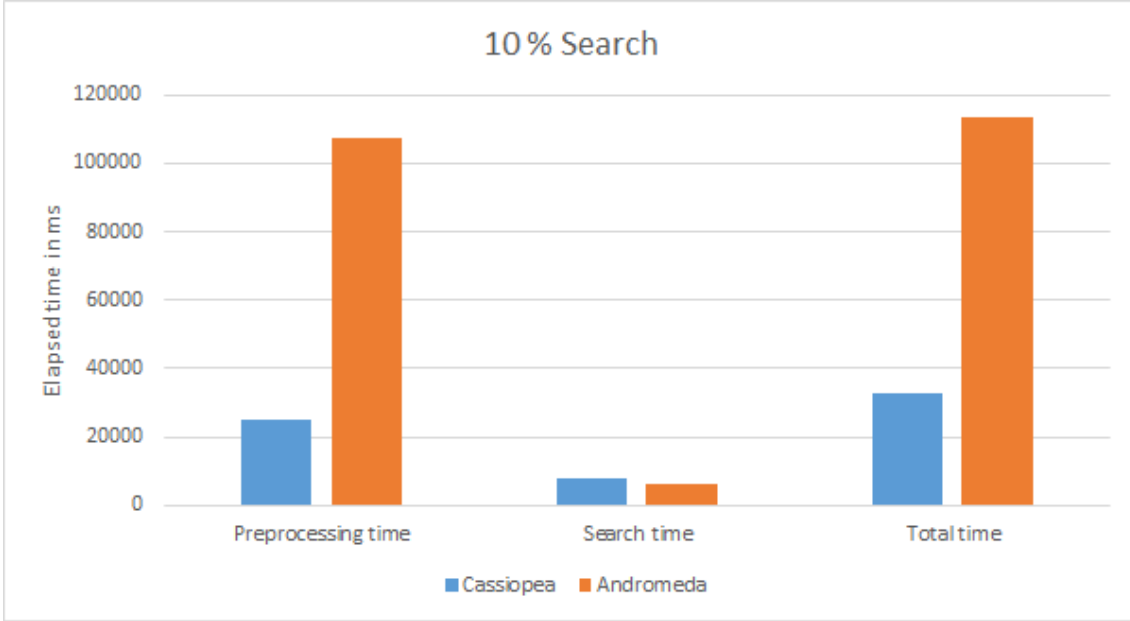


Figure 5.6: Search performed on 10% of the protein sequence data set UPSP_Nov2017 and 10% of the Ecoli_04_RD2_01_1275 peak data set. The first 10% of both files was used to evaluate the duration of each phase on a small set of spectra with a small sequence data base used as a reference.

Metrics	Andromeda	Cassiopeia
Pre-processing time	457,039 ms	138,113 ms
Search time	11,960 ms	114,679 ms
Total time	468,999 ms	252,793 ms

Table 5.7: Identification time in milliseconds for 50% of the complete proteins and peaks data sets

pre-processing, search, and total times can be seen in Table 5.7. We see that the pre-processing times for Cassiopeia takes one-fourth of the time - 138,113 ms, compared with Andromeda needing 457,039 ms to compose the sequence knowledge base. The scoring correlation, however, has increased drastically, where Cassiopeia requires 114,679 ms, which is 10 times longer than Andromeda (11,960 ms) to evaluate the same amount of spectra. The total time for the complete evaluations by Cassiopeia (252,793 ms) is less than Andromeda (468,999 ms).

The experiment run-time is shown in Figure 5.8 is done with both complete sets, using the whole 77 MBs of data for the spectra and the 270 MB protein set. The complete data set works with all 1,120,828 proteins and evaluates all 40,738 spectra. The times presented are grouped into three graphs, where one is for the pre-processing time needed to prepare the sequence database and to load the spectra from the mass spectrometry experiment results. Additionally, we have displayed the time spent on evaluating the spectra, grouped into Search time in milliseconds and also shown the end sum of the experiment run-time under Total time.

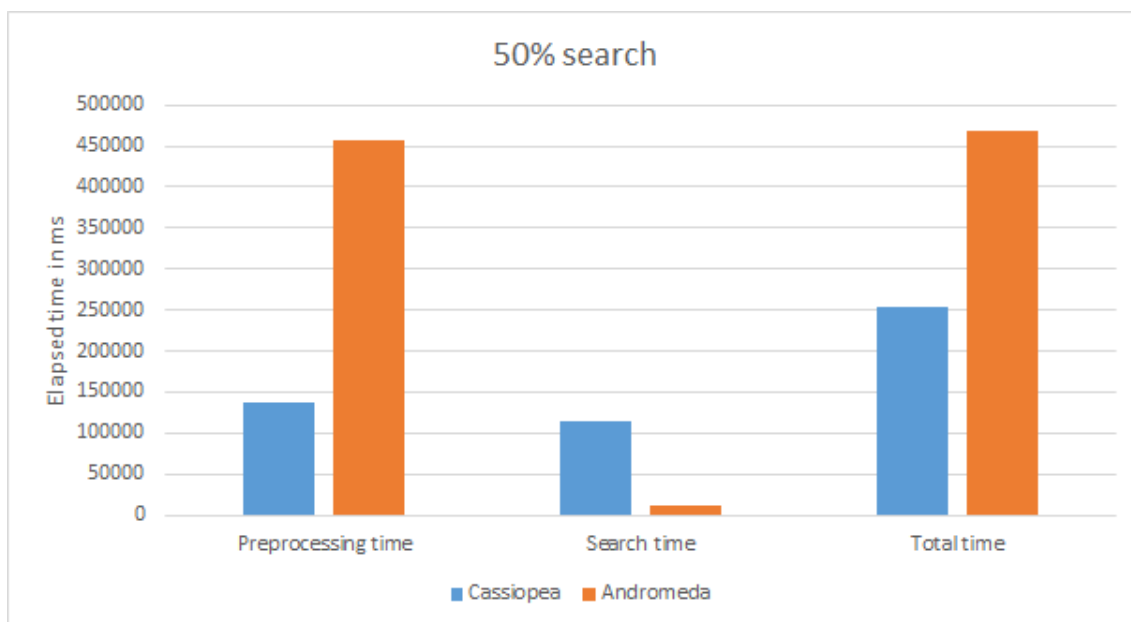


Figure 5.7: Search performed on fifty percent of the data.

Metrics	Andromeda	Cassiopeia
Pre-processing time	817,529 ms	268,309 ms
Search time	12,757 ms	328,302 ms
Total time	830,286 ms	596,612 ms

Table 5.8: Identification time in milliseconds for the complete proteins and peaks data sets

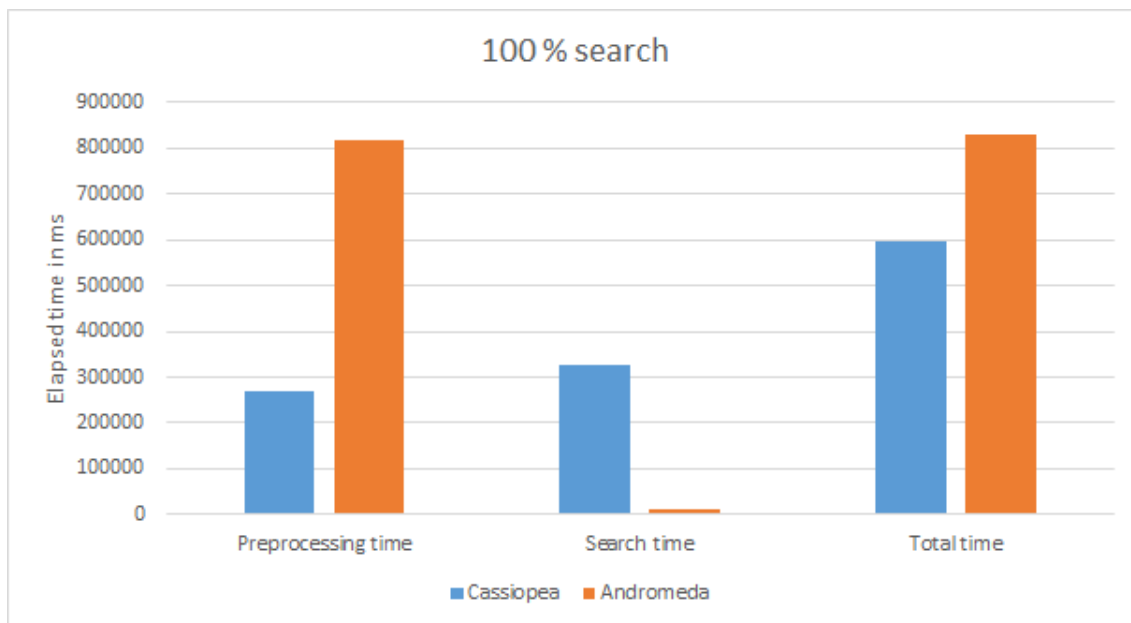


Figure 5.8: Search performed using the complete data sets representing the spectra from the mass spectrometry experiment and the UPSP_Nov2017 protein knowledge base.

The results achieved between the different phases is shown in Figure 5.8 and the time elapsed is visualized in Table 5.8. We see that the pre-processing trend is kept and Cassiopeia (268,309 ms) requires a bit over one-quarter of the time Andromeda needs (817,529 ms). The search time ratio has further increased, where the search time accounts for 55 % of the total duration of Cassiopeia - 328,302 ms, while Andromeda's takes about 1% of the experiment run time - 12,757 ms.

In Figure 5.9, we have displayed the relationship between the time elapsed for the preparation of the inputs. The x-axis shows the data set size used, and the y-axis is for the elapsed time in milliseconds. Andromeda trend has been drawn with orange and Cassiopeia with blue. We see that the pre-processing scaling ratio is the same for all different experiments, where Cassiopeia requires 1/4th of the time needed for Andromeda to prepare the sequence database.

Figure 5.10 displays the trends for the evaluation phase. The x-axis displays the percentage of data sets used, and the y-axis displays the time spent. We see that Andromeda (orange) is the better performer since the data set almost doesn't influence the search time, and the time required is linear. This isn't the case for Cassiopeia (blue), where search times initially follow closely, but after the 50% mark explode and continue increasing exponentially.

In Figure 5.11 we have plotted the time scaling of each step of the experiment - the pre-processing time, the duration of the evaluation phase and in the end the total duration of the test run. The x-axis is used to indicate the input file sizes, and the y-axis displays the total time elapsed in milliseconds. We can see that in the end, the total time of Cassiopeia (blue) is still lower, thanks to the advantages during the pre-processing phase. The values for each experiment run are also displayed in

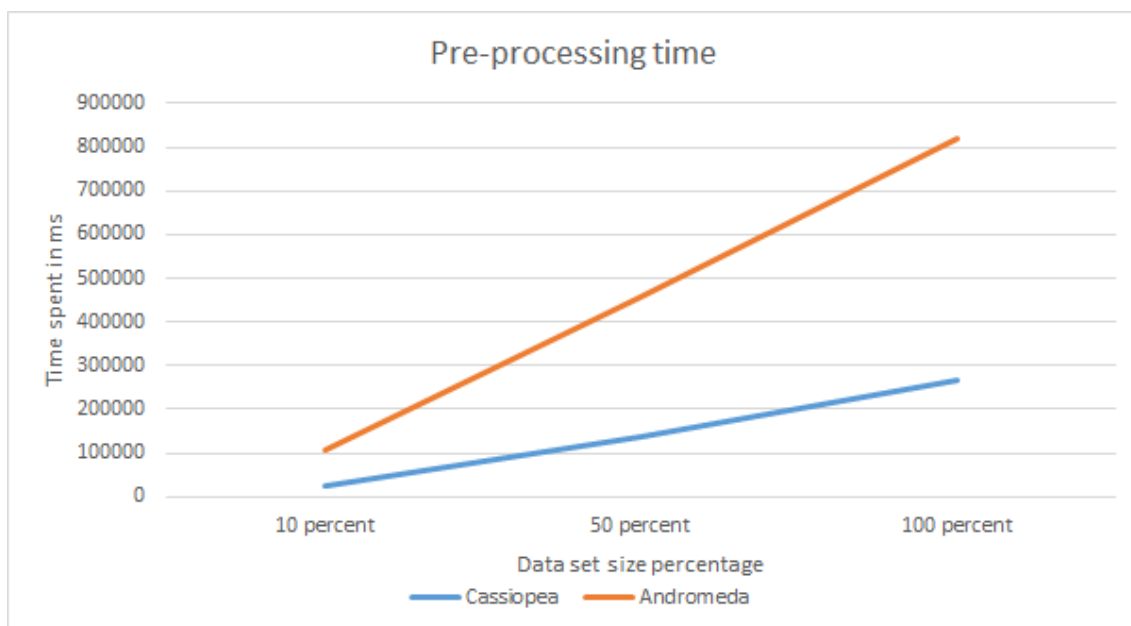


Figure 5.9: Time spent for the building of the peptide sequence database out of a FASTA file, containing protein sequence information. There are three different sizes of the initial UPSP_Nov2017 considered.

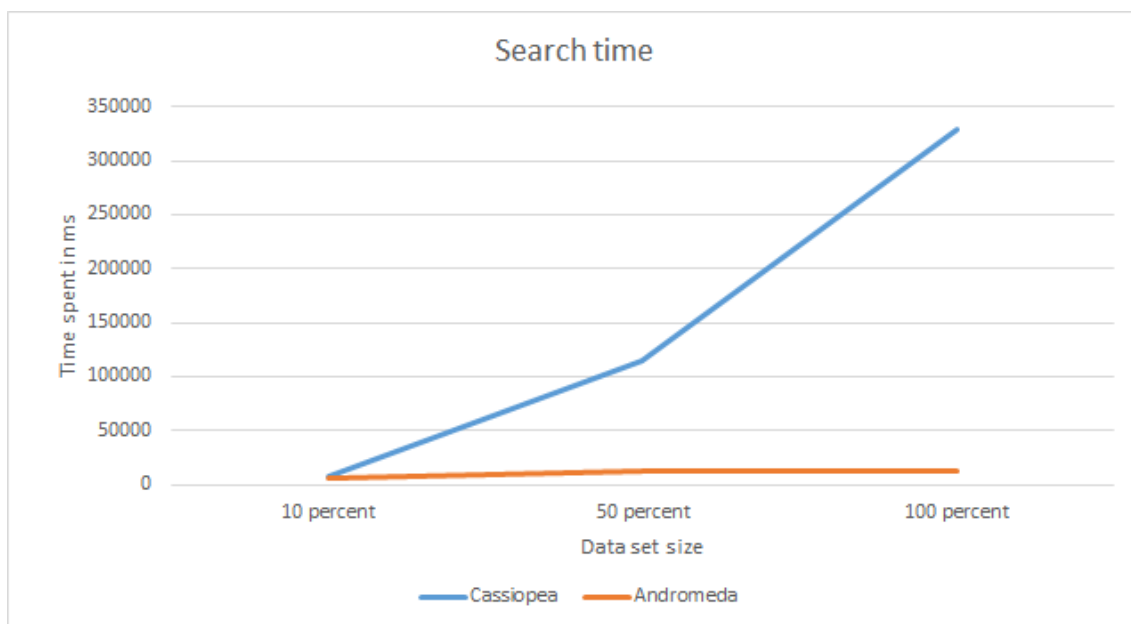


Figure 5.10: Time spent by Cassiopeia (blue) and Andromeda (orange) on evaluation peptide-spectra matches on different data set sizes. The x-axis shows the data set size and the x-axis displays the time taken in milliseconds.

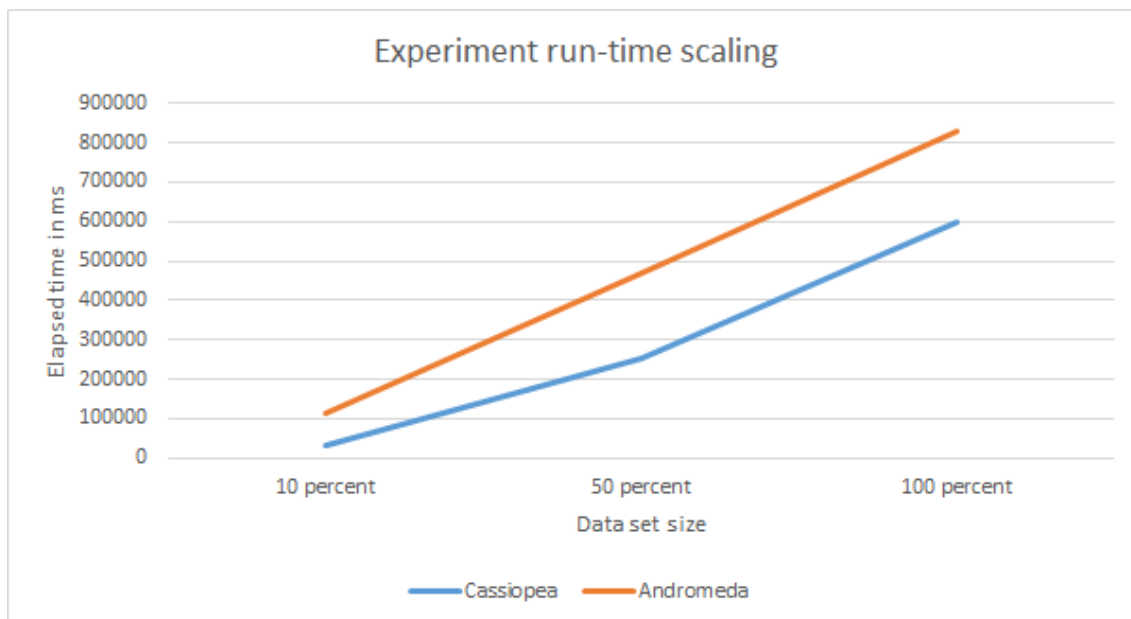


Figure 5.11: The total time spent for an evaluation of different data set sizes for Ecoli_04_RD2_01_1275 without modifications. The time is represented in milliseconds and is the sum of pre-processing and search times.

Total time	Andromeda	Cassiopeia
10 % proteins and peaks sets	113,668 ms	32,625 ms
50 % proteins and peaks sets	468,999 ms	252,793 ms
Complete proteins and peaks sets	830,286 ms	596,612 ms

Table 5.9: Total time elapsed for the evaluation of the three different data set pair sizes.

Table 5.9. For the smallest data set, Cassiopeia takes 27% of the time needed by Andromeda, for half of the data set - 54% and for the whole data set - 71 %. We see that the bigger the data set, the less advantage does our new tool have, with a trend to keep increasing.

We would also like to note the differences in resources used by both systems. Since Andromeda has been implemented with working on 32-bit Windows environment in mind, it does not exceed 2 GBs RAM usage. The CPU utilization also keeps a single core busy, which peaks at 17% utilization. In contrast, the scaling of RAM usage for Cassiopeia is exploding with the data set sizes. When considering the 10% experiment, the tool was peaking at 4 GBs of RAM. For 50% the needed memory space was around the 12 GB mark and for the complete data set peaks at 26 GBs of RAM usage. This happens at the beginning of analysis until the peptide database is built, after that step, the utilized memory is halved, but the heap size is kept until the last stages of the evaluation phase. For the use case of working with the complete data set sizes, we encountered the second peak much earlier during the evaluation stage. As far as CPU utilization goes for the pre-processing phase, it is capped, due to the parallel loading of input data. This also happens during the evaluation phase,

Measurement	Cassiopeia	MStream
Records count	57,000	57,000
Total time in seconds	4,842 s	899 s
Average records processed per second	12	63
Total PSM evaluations	179,841	17,173

Table 5.10: MStream integration comparison. For the processing of 57,000 individual spectra Cassiopeia takes five times longer by evaluating ten times more peptide-spectrum-match objects.

but it's worth to note that for the complete data set a more significant part of the work was assigned to garbage collection and when using less than 25 GBs of ram we encountered out of memory exceptions.

5.3 Cloud evaluation

After performing evaluations on how Cassiopeia works on a local environment, compared to Andromeda, we have evaluated how our tool performs as a peptide scoring mechanism when integrated into the existing MStream environment. Instead of using the default X!Tandem scoring algorithm we have performed the tests with our Cassiopeia implementation. In this experiment, we have used the variable modifications described in Table 5.1. The data sets used were Ecoli_02_RD2_01_2199 for the peaks, and UPSP_Nov2017 was used to construct the peptide sequence database, stored into Cassandra.

The output summary can be seen in Table 5.9. Although the initial processed records are the same - 57,000, the evaluated data sets differ significantly in records produced. The amount produced by both peptide scoring methods differs in orders of magnitude - the original scoring of MStream with X!Tandem considers 17,173 records, where Cassiopeia evaluates 179,841. The size naturally also impacts the performance, where for the whole duration of the evaluation, the time required with X!Tandem (899s) as a scoring mechanism is five times faster than Cassiopeia (4,842 s).

After taking a look at the performance, we checked out what kind of overlap do the two sets produce. In Table 5.11, we can see the number of unique peptides identified and how much of them overlap when only the top 1000, ordered in a descending manner by score, are considered. Since Cassiopeia's set is a bit over 10 times larger it also produces a bit over 10 times the unique sequences of MStream (12,942) - 137,861 distinct peptides are the result of flattening the whole PSM set to contain unique only sequence keys. After considering the top 1000 records by score we receive 64 shared peptides between the 812 unique strings from Cassiopeia and 526 resulting from X!Tandem.

Since the data sets differ quite a lot in sizes, we have evaluated how parts of the top results occur when considering the complete counterpart data set. In Table 5.12 we have taken the top 100 to 1000 keys from X!Tandem scoring and evaluated how much of them are contained in the 137,861 unique peptide sequences, evaluated by Cassiopeia. We see that at most, we have a 74% overlap of keys when we take the

Use case	Cassiopeia peptides	MStream peptides	Common peptides
Total	137,861	12,942	1,427
Top 1000 from both	812	526	64

Table 5.11: Overlapping peptides identified by both protein identification algorithms

Use case	Cassiopeia peptides	MStream peptides	Common peptides	%
Top 100	137,861	42	30	71%
Top 200	137,861	105	78	74%
Top 500	137,861	281	189	67%
Top 750	137,861	413	259	62%
Top 1000	137,861	526	321	61%

Table 5.12: Overlapping peptides identified by both protein identification algorithms where the whole Cassiopeia data set is used

first 200 X!Tandem scores and filter them to keep only the unique keys present. After that, the shared sequences dwindle as a percentage and go lower and lower.

When considering ordering the scores for Cassiopeia’s output and comparing them against the complete X!Tandem the maximum overlap is less and the number of common peptides deteriorates much faster. The best case is when the top 100 scored PSM are considered - they are boiled down to 88 unique peptides and have a 60% overlap. When increasing the considered keys, even with 200 we end up having 35% shared identification which goes to 16% with the top 500 best PSMs and continue dropping. One thing to note is that top scores by Cassiopeia produce a bigger percentage of unique peptides. When comparing [Table 5.13](#) we have 80+% unique strings, when in [Table 5.12](#) the results are between 42% and 55%.

Since we achieved the highest overlap for the comparison with the top 200 scoring PSMs using X!Tandem scoring mechanism we have compared the average of the evaluated scores in [Figure 5.12](#). The x-axis shows the distinct 78 shared by Cassiopeia and X!Tandem peptide sequence results. The y-axis displays the score achieved. The MStream line in orange is the average of the scores for the peptide sequence present in the data set. The same has also been done for Cassiopeia’s scoring function, colored in blue. In the figure, we observe 78 distinct peptides, obtained by extracting the unique keys out of the top 200 X!Tandem results. We have averaged the scores of each key for both scoring mechanisms and plotted it on the figure. The MStream

Use case	Cassiopeia peptides	MStream peptides	Common peptides	%
Top 100	88	12,942	53	60%
Top 200	173	12,942	60	35%
Top 500	433	12,942	68	16%
Top 750	632	12,942	77	12%
Top 1000	812	12,942	85	10%

Table 5.13: Overlapping peptides identified by both protein identification algorithms where the whole MStream data set is used

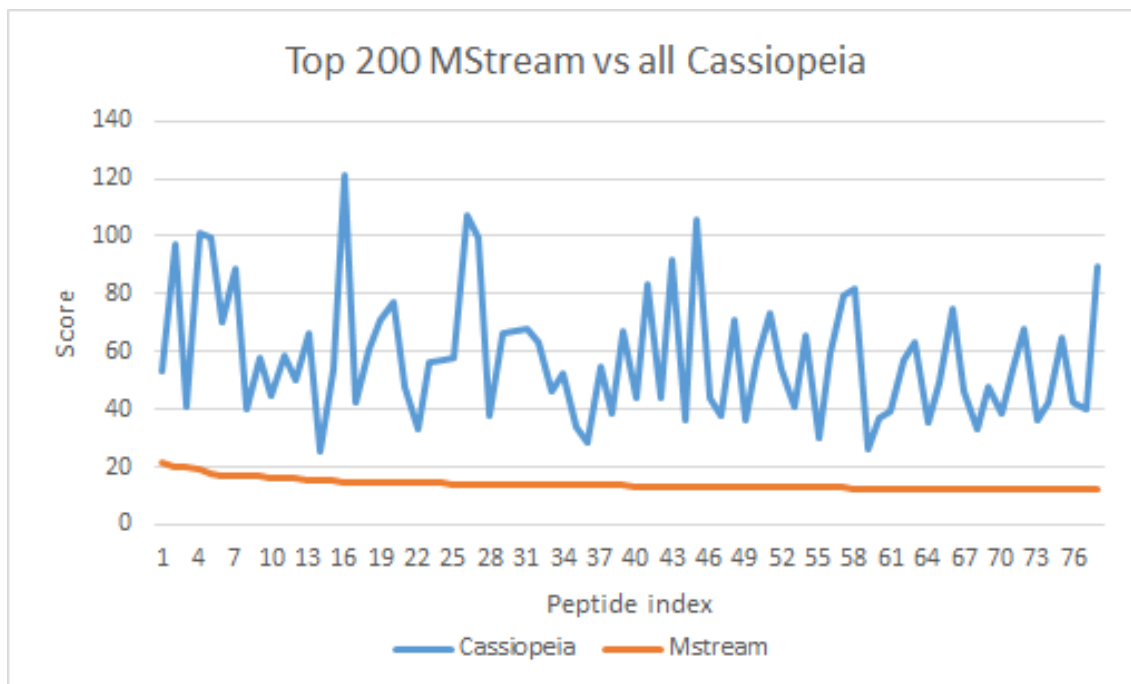


Figure 5.12: The average of the sum of scores for the top 200 MStream PSMs reduced to 78 unique peptide sequences, which are present also in Cassiopeia’s output for the cloud evaluation. The scores are ordered by descending for results produced by X!Tandem scoring and matched with their Cassiopeia counterparts. Both X!Tandem and Cassiopeia possess different scoring scales, where the top X!Tandem score achieved was 26 and the top Cassiopeia was 121.

graph (orange) appears in descending fashion, due to the ordering. The Cassiopeia line (blue) has a lot more peaks because we have represented the averages of scores by using the X!Tandem peptide sequence as a key. Both tools have different scoring scales, so it is normal that they do not overlap in this case. One thing to note is that the top-scoring MStream PSM with the score of 26 is not present in the graph, whereas the top result from Cassiopeia with the value 121 does occur.

5.4 Summary

In this chapter, we showed the evaluations for both local and cloud environments. We obtained almost the same results during the validation with Andromeda, where missing results were present only for the complete data set with present modifications. For the case without modifications, we had the same unique peptide data set. In contrast - the set with modifications resulted in Cassiopeia producing roughly 10% more PSMs, but the unique peptide sequences were the same, apart from 1 missing peptide in Cassiopeia, produced by Andromeda. The missing peptide, however, was part of a one spectrum evaluation, producing 14 distinct PSM objects and was 13th by score when ordered from highest to lowest score.

For the time evaluations, we displayed that Cassiopeia takes a quarter of the time needed by Andromeda. The scoring phase was, however, in favor of Andromeda,

where it took almost constant time. In contrast, the time required by Cassiopeia was rising exponentially. In the end, Cassiopeia still managed to outperform Andromeda for the data sets sizes considered.

As far as the cloud evaluations, we displayed a significant difference in the results produced and time required. Cassiopeia produced a bit over 10 times for both complete PSM evaluations and distinct peptide sequences. Cassiopeia also took 5 times longer than the default MStream implementation to finish the evaluation. Due to the large data set sizes, the overlap percentage was low. We found out, however, that when an according filtering is applied, we can achieve an overlap of 74% at most. In the next chapter we go over the results and discuss the reasons behind the presented evaluations.

6. Discussion

We presented a set of experiments in [Chapter 5](#), intended to compare the performance of Cassiopeia's scoring algorithm against a local Andromeda stand-alone protein identification software and also the X!Tandem scoring method used in MStream, deployed on a cloud environment.

6.1 Local environment

The experiments performed can be divided in two types - one for local environment, meant to validate the correctness of our results and another experiment, evaluating what is the output when Cassiopeia is used instead of the default X!Tandem scoring. We first discuss what can be extracted from the results in a local environment.

6.1.1 Validation results

We have begun the evaluation phase initially in a limited environment by analyzing parts of the data sets and data without modifications. We began the evaluation on a small data set with no modifications. Both the proteins for the sequence database and the evaluated spectra were 10 % of the total entries count. The 63 distinct results were of 47 different peptides. In [Section 5.2.1](#) we showed that almost one third of peptides occur more than once and in [Figure 5.2](#) we showed the deviations in averages for the PSM evaluations. Those occurrences do appear in the same fashion in both scoring functions, which is a good sign. The deviation can be attributed to different spectra containing different parts of the protein chain. This means that not the whole peptide may be contained, but is a good enough indication of a discovered peptide. We have also evaluated the complete data set without modifications, and we again obtain the exact same results, where we find one more entry. Since we considered always the first 10% of the files in both use cases we can state, that the all noteworthy peptide sequences appear in the first 10% of both data sets. Additionally, no differences in the maximum scores are observed. Therefore we can be certain that, as far as for data sets without modifications, Cassiopeia produces the exact same output.

When we consider working with validations in [Section 5.2.2](#) however, differences begin to be present. We obtain 960 PSM evaluations by Cassiopeia, compared by the 882 produced by Andromeda. The amount of peptides considered by Cassiopeia is equal or larger than those from Andromeda. This explains also how our tool has acquired more evaluations. When we filter the peptides down to only a set of unique peptide sequences, we observe that a single key from Andromeda is missing in Cassiopeia's output. Although having more results, the missing peptide is a result of an evaluation which has considered 14 distinct peptides and the missing element is 13th when ordered by score. Since all other sequences are present, we can attribute this to be an edge case, which does not invalidate the correctness of the resulting subset. Had it been further ahead, this would be a definitive problem, but due to the higher number of peptide results supplied by Cassiopeia we can attribute this to the filtering conditions not present in Cassiopeia.

To double-check how the extra evaluation influences the score, we have also compared the maximum scores and averages of both protein identification tools. The average fluctuates a bit, but mainly for peptides which have an extra evaluation. The PSMs with the biggest average differences share a common value for the maximum score, apart from one. The peptide with the most significant difference in maximum score has the largest difference in the times it has been considered as a match and the average score. Thus by having a large score and more distinct results, this explains the big difference in the average scores. Concerning different mismatches for the maximum scores, there are three additional peptides, which differ, where two of the differ by only a little. The averages of those three are however closer, when compared to the most significant difference. Apart from those four distinct cases, the results are the same or differ by very little with regards to the maximum and average score, and the times they have been evaluated as a match. Considering the data set contains 109 unique peptides, and 105 are almost identical, we are confident that Cassiopeia is a viable protein identification method.

6.2 Scaling

In [Section 5.2.3](#) we performed evaluations with different amounts of input data on a local environment. Our goal was to show that our tool can scale with more resources. In terms of scaling, we observed significant improvement over the method used to construct protein sequence databases for a local environment. This

When we consider the time needed for analyzing 10 Percent of the complete data sets we can see that the elapsed time for the identification of both Andromeda and Cassiopeia is quite close, the difference is only 0.5 Seconds. Considering the size of the data set however, this shows a trend for our software to be slower in the scoring phase when compared to the Max Quant peptide identification tool. Cassiopeia saves a considerable amount of time during the pre-processing phase. Not working with a file-based system and operating only with objects stored in the main memory naturally skips the biggest bottleneck in nowadays computing, this being the read / write operations done on the hard drive. Even with a *NVMe* drive capable of doing around 435MB/s mixed I/O operations, it does not come close to the speed provided by dual-channel 2666 MHz RAM. This is one of the main reasons that we

see a pre-processing time that take 4 times longer. This comes at a high cost, since during the building of the sequence knowledge base, 25 GBs of RAM are required at the peak. This happens only during the digestion of proteins and creating a map object, which binds the peptide and stores their origin. They can be present in multiple proteins, therefore the usage of a mutable list has been used. Scala generally does not encourage the usage of mutable objects and also having heavy objects as classes, which is the case of the peptide object used during the sequence database build. These factors contribute to the large memory footprint.

When we consider the time needed for the evaluation of the search time takes almost the same amount of time for small data sets, but rises exponentially with the increase of spectra. The total time for experiment runs is in favor of Cassiopeia for all input sizes. Although Cassiopeia is generally faster than Andromeda when it has to build its own peptide knowledge base, the margin of the total time keeps reducing with the increase of the data sets sizes (Table 5.9). The biggest bottleneck for Andromeda appears to be the working with the hard drive, but the correlation is consistent between different data set sizes. The search time for Andromeda also appears constant, while Cassiopeia search time keeps increasing exponentially. We attribute this to the current implementation of Cassiopeia being in an object-oriented fashion and Scala promoting functional approach. The usage of indexes, although slower, allows sequential iteration over the known peptide sequence knowledge base in a much more efficient way than. Andromeda achieves this by using an index and an offset indicating where the start point of a search can be. Instead of always re-materializing entries from the in-memory.

Having a slow scoring function still results in Cassiopeia performing better for different data set sizes, although the margin of improvement decreases, it does scale with the presence of sufficient enough amount of RAM to keep up the advantage in performance. If a sufficient amount is present, this reduces the influence of garbage collection occurring, although an implementation with a better consideration for memory makes more sense.

6.3 Cloud result analysis

Since the intended use of Cassiopeia is as a cross-validation mechanism to be used in MStream, which is located in a cloud environment, we analyze the results from the cloud experiment performed. In Section 5.2.1, Section 5.2.2 and we showed that we have results with a validity, that can be used for cross-validation. Figure 5.9 evaluated the rates of scalability and how our tool performs under bigger loads. In Section 5.3 there are significant differences present. The difference between the PSM count being evaluated is almost tenfold. We attribute the enormous difference in the numbers is due to the number of results not being filtered by Cassiopeia. We are currently returning all possible results for an identified peptide. For the cloud implementation, we have also omitted the cache list, since each spark job creates a new instance of the scoring function and the cache might not be so useful. This results in the same sequence being evaluated multiple times. The difference in the resulting output also ends up influencing the time required to perform the whole experiment. Cassiopeia takes 5 times longer than when MStream uses X!Tandem

for evaluation. The positive side is that Cassiopeia appears to evaluate twice the amount of PSMs for the same amount of time needed by X!Tandem.

When we filter the results to contain only unique peptides, we still keep the margin of difference in unique sequences present - ten times more. There is an overlap to be observed between the output of both tools. The ratio is rather small when considering all unique sequences. When we filter out the worst evaluations, the margin increases, but is again depending on what is filtered out. The best performance was achieved when considering only the top 200 scores produced by X!Tandem - we have 74% shared peptide sequences. X!Tandem sequences tend to be present more often in the other set, but this is expected, considering the second output is ten plus times larger. Additionally, X!Tandem records have different amplitudes in the score, since we ordered them in a descending manner. When we compare the scores shared by both the default MStream scoring mechanism and by Cassiopeia produces a set with large fluctuations in Cassiopeia's results. The best scoring peptides from Cassiopeia are present, which is a good indicator for the relevance of scores. Thus we can deduce, that although Cassiopeia produces much larger data sets for a longer duration, it can be easily used as a validation approach when we want to observe if some results are indeed better matches.

6.4 Summary

The evaluations performed to confirm the validity of Cassiopeia's results with regards to Andromeda. We produce the same relevant results as Andromeda. Although more resource-intensive it does provide a faster analysis run-time, as long as the data sets don't get too large. The best features of Cassiopeia, packed behind its in-memory database, also extract the most substantial toll on performance during the actual search for matching peptides regarding a single spectrum. This can be attributed more to the current utilization of the knowledge base.

As far as performance in a cloud environment, we observe much larger sets of PSMs being evaluated. This increases the total run time and confines the overlap between Cassiopeia and X!Tandem. When inspecting the top X!Tandem results, however, we obtain results with higher fluctuations in scores for the cloud implementation of the Andromeda scoring algorithm. The score differences in Cassiopeia can be used to adjust the meaningfulness of X!Tandem result, fulfilling the role of a cross-validation mechanism. The performance could be improved by the introduction of the filtering method, used by Andromeda, but this will reduce the data set available for comparison. Cassiopeia offers flexibility in the potential use, which is a valuable asset for a cloud environment. The cloud's horizontal scalability may be utilized to adjust the results needed for each individual experiment, thus improving the confidence and usability of MStream results. In the next chapter we will present some related peptide identifications software tools, also deployed in a distributed data systems.

7. Related Work

As related work, we can mention a couple of distinct researches that are also protein search tools, deployed on a cloud environment. Their approaches can be summarized as approaches, which use existing peptide identification mechanisms by wrapping them into a container, which is used for the evaluation. We will go over three distinct software solutions using this approach: *Q-Cloud*, *MS-PyCloud* and *Chorus*.

7.1 Q-Cloud

The Q-Cloud [CRE⁺18] tool wraps the usage of OpenMS [SBG⁺08] software as a scoring mechanism. OpenMS is a framework intended to be of use for developers working on applications handling mass spectrometry data. The scoring functionality is exposed by a local client interface, run as a script with giving input information as batches. In contrast to MStream results are not sent to a message queue but are directly inserted into a database. In order to perform an evaluation, the script is executed, where multiple instances of the script can run simultaneously.

7.2 MS-PyCloud

MS-PyCloud [LBM⁺18] is another container for existing protein identification algorithms, implemented using Python. It is also deployed on a cloud environment, but the input data are still batches of information and are required to be present before the valuation can take place.

7.3 Chorus

The Chorus project [cho19] is similar to both Q-Cloud and MS-PyCloud since it is again a facade to use existing protein search engines. It has been deployed on an Amazon container environment and provides the evaluations as a web service. Living in the Amazon ecosystem, it benefits of state-of-the-art load balancing and scalability. The input files for the spectra and proteins are retrieved from a S3

file system, which is also used for the storage of performed evaluations. Using the commercial suite of tools, it provides ease in sharing research data. Additionally, multiple files can be given as an input, in contrast to MStream working on a single data set at a time.

7.4 Summary

The above presented related tools share the approach in using existing tools already commonly used throughout the metaproteomic field and by researchers using mass spectrometers. They all offer either single or an ensemble of peptide evaluation mechanisms by abstracting their functionality into services, which can be called from a cloud environment. Although all offer scalability and availability due to the nature of the cloud architectures, the software tools are not implemented with the use case in mind. In contrast to these 3 software solutions, MStream uses naively integrated scoring mechanisms in the attempt to give better performance by not encapsulating tools intended for environments, not native to a cloud architecture.

8. Conclusion

The nowadays go-to approach of analyzing organisms in the field of metaproteomics is to use a mass spectrometer. To help analyze the results from metaproteomic experiments, researchers use protein identification software tools. Although there are available open-sourced and proprietary software solution for dealing with the mass spectrometry data, there is a significant downside - nearly all of them require to have the complete experiment spectrum data set present beforehand. Since the measurement may take a couple of hours or more before completion, we were compelled to analyze new measurements in real-time as each spectrum is evaluated. Additionally, the majority of those tools are also limited in horizontal scalability by being usable only in a local environment. In the work of Zoun et. al. - MStream, data is analyzed in near-real-time. It is also implemented in a SMACK stack ecosystem, situated in a cloud environment, ensuring a distributed software which is highly scalable, due to the nature of the cloud architecture.

Our goal was to extend the functionality of MStream by the introduction of a new peptide evaluation mechanism. We implemented Cassiopeia - a protein search engine, based on Max Quant's Andromeda stand alone. With this work, we have tried to answer three questions.

Our first goal was to compare the correctness of our tool Cassiopeia against the Andromeda stand-alone open-sourced solution, from which we have utilized the peptide scoring algorithm. We managed to reproduce 99% of the same results for different data sets in different conditions. Therefore we can conclude that our tool is on par with the Andromeda, regarding the correctness of the results.

Cassiopeia main goal is to be a cross-validation mechanism in the existing MStream environment. So after validating the output results, our second goal was to prove the viability of introducing our new tool into an existing ecosystem. Although the result sizes differ significantly between Cassiopeia and MStream, we have obtained common peptide sequences, after some filtration of the output. Cassiopeia's score had a higher amplitude and more significant diversity in the averages scores, compared to the top scores produced by MStream. Following the statement, that our tool

correctness is on Andromeda's level, we can deduce that by producing scores with large deviations, the high-ranking matches can be a viable indicator for a viable match. Those high scores can be used as a cross-validation feature for result entries produced by X!Tandem, when looking at the top MStream scoring evaluations in descending order.

Our third objective was to check how Cassiopeia performs compared to the state of the art MStream implementation. Our tool did take five times longer to complete the experiment but ended up performing ten times more evaluations. Even if the output is filtered to contain unique peptide sequences, the margin remains the same. It may not be suitable for all use cases, but the flexibility offered by the presence of an additional scoring mechanism in MStream can be a significant benefit. The Scala implementation also offers horizontal and vertical scalability, which can speed up the performance if there are resources to spare.

In our work we successfully recreated the evaluation approach used in Andromeda, integrated it into the existing MStream environment and managed to work on streams of data supplied in near-real-time, producing relevant and correct peptide evaluations, usable for protein identification or cross-validation purposes.

9. Future Work

Extracting top peptides, which are present in Cassiopeia but are missing for Andromeda can be attributed to the skipped filtering mechanisms in Cassiopeia. Limiting the evaluation of peptides can reduce the evaluated peptides and constrain the differences in data sets for example between our software and X!Tandem.

We also introduced the concept of using an in-memory data base for the peptide sequence knowledge base. This definitively proved to be valuable, since it attributes the most gain of performance when compared with the original Andromeda software. In MStream we use a Cassandra database

Additionally, we can always improve our tool by reducing the object-oriented methodologies used throughout and adopt a more functional approach, since this is encouraged by Scala. This can reduce the memory footprint and the effort needed for garbage collection of objects, this reducing the run-time of our tool.

Since Cassiopeia proved a potentially valuable asset to the MStream stack it is worth considering adding more peptide identification tools. This can increase the confidence of results exponentially, additionally it can be implemented as the go-to suite of protein identification software, which can handle mass spectrometry in near-real time as a stream. Basing itself on a SMACK stack will not only ensure resilience and availability due to the distributed nature of the cloud, it can be also scaled easily with tools that fit best for a particular experiment. Cassiopeia can work with two different formats as spectra input, which can be potentially extracted into a service, suitable to process any open-sourced or other type of data format. Having a bundle of possible protein search engines to choose from will offer flexibility, while allowing to adjust the confidence levels and adapt the run-time for maximum time utilization.

Bibliography

- [AA98] N. Leigh Anderson and Norman G. Anderson. Proteome and proteomics: New technologies, new concepts, and new words. *Electrophoresis*, 19(11):1853–1861, August 1998. (cited on Page 6)
- [AAE16] Nuha Alshuqayran, Nour Ali, and Roger Evans. A Systematic Mapping Study in Microservice Architecture. In *2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)*, pages 44–51, Macau, China, November 2016. IEEE. (cited on Page 25)
- [ACC⁺16] Marie-Aude Aufaure, Raja Chiky, Olivier Curé, Houda Khrouf, and Gabriel Kepekian. From Business Intelligence to semantic data stream management. *Future Generation Computer Systems*, 63:100–107, October 2016. (cited on Page 24)
- [AG01] Ruedi Aebersold and David R. Goodlett. Mass spectrometry in proteomics. *Chemical Reviews*, 101(2):269–296, February 2001. (cited on Page 8)
- [Aki15] Tyler Akidau. Streaming 101: The world beyond batch, August 2015. (cited on Page 23)
- [Aki16] Tyler Akidau. Streaming 102: The world beyond batch, January 2016. (cited on Page 23)
- [All13] Jamie Allen. *Effective Akka*. O’Reilly Media, Inc., 2013. (cited on Page 4)
- [AM03] Ruedi Aebersold and Matthias Mann. Mass spectrometry-based proteomics. *Nature*, 422(6928):198–207, 2003. (cited on Page 1 and 8)
- [AWS] Inc. Amazon Web Services. Amazon DynamoDB - Overview. (cited on Page 30)
- [BB89] Andrew D. Birrell and Andrew D. Birrell. An introduction to programming with threads. 1989. (cited on Page 29)
- [BB08] Robert Battle and Edward Benson. Bridging the semantic Web and Web 2.0 with Representational State Transfer (REST). *Journal of Web Semantics*, 6(1):61–69, February 2008. (cited on Page 25)

- [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*, 7(1):293–299, 2008. PMID: 17902638. (cited on Page 2, 4, and 15)
- [BGL98] Jean-Pierre Briot, Rachid Guerraoui, and Klaus-Peter Lohr. Concurrency and distribution in object-oriented programming. *ACM Computing Surveys*, 30(3):291–329, September 1998. (cited on Page 29)
- [BH] Nathan Bijmens and Michael Hausenblas. Lambda Architecture lambda-architecture.net. (cited on Page viii, 21, and 23)
- [Bla13] Andrew P. Black. Object-oriented programming: Some history, and challenges for the next fifty years. *Information and Computation*, 231:3–20, October 2013. (cited on Page 29)
- [BW99] Walter P Blackstock and Malcolm P Weir. Proteomics: quantitative and physical mapping of cellular proteins. *Trends in Biotechnology*, 17(3):121–127, March 1999. (cited on Page 6)
- [CB03] Robertson Craig and Ronald C. Beavis. A method for reducing the time required to match protein sequences with tandem mass spectra. *Rapid Communications in Mass Spectrometry*, 17(20):2310–2316, 2003. (cited on Page 2)
- [CCS93] E. F. Codd, S. B. Codd, and C. T. Salley. Providing OLAP (On-Line Analytical Processing) to User-Analysts: An IT Mandate. E. F. Codd and Associates, 1993. (cited on Page 22)
- [cho19] chorusproject.org. About chorus project. <https://chorusproject.org/pages/about.html>, jul 2019. (cited on Page 83)
- [CM15] Irving Cordova and Teng-Sheng Moh. DBSCAN on Resilient Distributed Datasets. In *2015 International Conference on High Performance Computing & Simulation (HPCS)*, pages 531–540, Amsterdam, Netherlands, July 2015. IEEE. (cited on Page 28)
- [CNL⁺08] Jatin Chhugani, Anthony D. Nguyen, Victor W. Lee, William Macy, Mostafa Hagog, Yen-Kuang Chen, Akram Baransi, Sanjeev Kumar, and Pradeep Dubey. Efficient implementation of sorting on multi-core SIMD CPU architecture. *Proceedings of the VLDB Endowment*, 1(2):1313–1324, August 2008. (cited on Page 29)
- [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*, 10(4):1794–1805, 2011. PMID: 21254760. (cited on Page vii, ix, 2, 4, 15, 16, 18, 33, 54, and 57)

- [CO98] Kerry Coffman and Andrew Odlyzko. The work of the encyclopedia in the age of electronic reproduction. *First Monday*, 3(10), October 1998. (cited on Page 19)
- [CRE⁺18] Cristina Chiva, Roger Olivella, Eva Borrás, Guadalupe Espadas, Olga Pastor, Amanda Sole, and Eduard Sabido. Qcloud: A cloud-based quality control system for mass spectrometry-based proteomics laboratories. *PLOS ONE*, 13(1):1–14, 01 2018. (cited on Page 83)
- [Deu12] Eric W. Deutsch. File Formats Commonly Used in Mass Spectrometry Proteomics. *Molecular & Cellular Proteomics*, 11(12):1612–1621, December 2012. (cited on Page 8)
- [DHJ⁺07] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. Dynamo: amazon’s highly available key-value store. In *IN PROC. SOSPP*, pages 205–220, 2007. (cited on Page 20)
- [Dom06] B. Domon. Mass Spectrometry and Protein Analysis. *Science*, 312(5771):212–217, April 2006. (cited on Page 10)
- [DS10] Bart De Strooper. Proteases and Proteolysis in Alzheimer Disease: A Multifactorial View on the Disease Process. *Physiological Reviews*, 90(2):465–494, April 2010. (cited on Page 11)
- [ER16a] Raul Estrada and Isaac Ruiz. *Big Data SMACK*. Apress, Berkeley, CA, 2016. (cited on Page viii and 27)
- [ER16b] Raul Estrada and Isaac Ruiz. *Big Data SMACK*. Apress, Berkeley, CA, 2016. (cited on Page viii, 28, and 31)
- [Foua] The Apache Software Foundation. Apache Mesos. (cited on Page viii, 29, and 30)
- [Foub] The Apache Software Foundation. Flume 1.9.0 User Guide — Apache Flume. (cited on Page 21)
- [Fouc] The Apache Software Foundation. HDFS Architecture Guide. (cited on Page 21)
- [Foud] The Apache Software Foundation. MapReduce Tutorial. (cited on Page 21)
- [Foue] The Apache Software Foundation. Sqoop -. (cited on Page 21)
- [Fou16] The Apache Software Foundation. Apache Cassandra, 2016. (cited on Page 30)
- [GHJV95] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995. (cited on Page 29 and 52)

- [GL02] Seth Gilbert and Nancy Lynch. Brewer’s conjecture and the feasibility of consistent, available, partition-tolerant web services. *ACM SIGACT News*, 33(2):51, June 2002. (cited on Page 20)
- [HBS73] Carl Hewitt, Peter Bishop, and Richard Steiger. A universal modular actor formalism for artificial intelligence. In *Proceedings of the 3rd International Joint Conference on Artificial Intelligence, IJCAI’73*, pages 235–245, San Francisco, CA, USA, 1973. Morgan Kaufmann Publishers Inc. (cited on Page 29)
- [HBS⁺93] W. J. Henzel, T. M. Billeci, J. T. Stults, S. C. Wong, C. Grimley, and C. 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*, 90(11), Jun 1993. (cited on Page 6)
- [HFV17] Waqar Haque, Matthew Fontaine, and Adam Vezina. Adaptive Deadlock Detection and Resolution in Real-Time Distributed Environments. In *2017 IEEE 19th International Conference on High Performance Computing and Communications; IEEE 15th International Conference on Smart City; IEEE 3rd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 571–577, Bangkok, December 2017. IEEE. (cited on Page 29)
- [HKRB15] Robert Heyer, Fabian Kohrs, Udo Reichl, and Dirk Benndorf. Metaproteomics of complex microbial communities in biogas plants. *Microbial Biotechnology*, 8(5):749–763, April 2015. (cited on Page 6)
- [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, NSDI’11*, pages 295–308, Berkeley, CA, USA, 2011. USENIX Association. (cited on Page 29)
- [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*, 261:24–36, November 2017. (cited on Page 1 and 2)
- [HYS⁺86] D F Hunt, J R Yates, J Shabanowitz, S Winston, and C R Hauer. Protein sequencing by tandem mass spectrometry. *Proceedings of the National Academy of Sciences*, 83(17):6233–6237, 1986. (cited on Page 8)
- [Kre11] Jay Kreps. Kafka : a distributed messaging system for log processing. 2011. (cited on Page 25 and 31)
- [Lab] Redis Labs. Redis. (cited on Page 30)

- [LBM⁺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. *bioRxiv*, 2018. (cited on Page 83)
- [LQ13] Miao Liu Lingdong Quan. CID,ETD and HCD Fragmentation to Study Protein Post-Translational Modifications. *Modern Chemistry & Applications*, 01(01), 2013. (cited on Page 10)
- [Mar] Nathan Marz. How to beat the CAP theorem - thoughts from the red planet - thoughts from the red planet. (cited on Page 21)
- [MBK⁺18] Florian Meier, Andreas-David Brunner, Scarlet Koch, Heiner Koch, Markus Lubeck, Michael Krause, Niels Goedecke, Jens Decker, Thomas Kosinski, Melvin A Park, Nicolai Bache, Ole Hoerning, Juergen Cox, Oliver Räther, and Matthias Mann. Online parallel accumulation–serial fragmentation (pasef) with a novel trapped ion mobility mass spectrometer. *Molecular & Cellular Proteomics*, 17:mcp.TIR118.000900, 11 2018. (cited on Page 17)
- [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. *Mol. BioSyst.*, 9(4):578–585, 2013. (cited on Page 13)
- [Met10] Michael L. Metzker. Sequencing technologies — the next generation. *Nature Reviews Genetics*, 11(1):31–46, January 2010. (cited on Page 12)
- [MHP01] Matthias Mann, Ronald C. Hendrickson, and Akhilesh Pandey. Analysis of proteins and proteomes by mass spectrometry. *Annual Review of Biochemistry*, 70(1):437–473, June 2001. (cited on Page 8)
- [Mon] Inc MongoDB. The most popular database for modern apps. (cited on Page 30)
- [MRML07] Pierre-Alain Maron, Lionel Ranjard, Christophe Mougel, and Philippe Lemanceau. Metaproteomics: A new approach for studying functional microbial ecology. *Microbial Ecology*, 53(3):486–493, March 2007. (cited on Page 6)
- [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. 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*, 18(18):2162–2168, September 2004. (cited on Page 8)

- [MZK17] Stathis Maroulis, Nikos Zacheilas, and Vana Kalogeraki. ExpREsS: Energy Efficient Scheduling of Mixed Stream and Batch Processing Workloads. In *2017 IEEE International Conference on Autonomic Computing (ICAC)*, pages 27–32, Columbus, OH, USA, July 2017. IEEE. (cited on Page 17)
- [NAR⁺] Patrik Nordwall, Johan Andrén, Johannes Rudolph, Arnout Engelen, Christopher Batey, and Helena Edelson. Akka: build concurrent, distributed, and resilient message-driven applications for Java and Scala | Akka. (cited on Page 29)
- [NAR⁺19] Patrik Nordwall, Johan Andrén, Johannes Rudolph, Arnout Engelen, Christopher Batey, and Helena Edelson. Akka. <https://akka.io>, 2019. (cited on Page 25)
- [NPPCC99] David N. Perkins, D.J.C. Pappin, David Creasy, and John Cottrell. Probability-based protein identification by searching sequence databases using mass spectrometry data. *ELECTROPHORESIS*, 20:3551–3567, 12 1999. (cited on Page 1, 2, and 15)
- [Nus81] Ruth Nussinov. The universal dinucleotide asymmetry rules in DNA and the amino acid codon choice. *Journal of Molecular Evolution*, 17(4):237–244, July 1981. (cited on Page 1)
- [O:U] Uniprot downloads website. (cited on Page 59)
- [PA06] Malu Polanski and N. Leigh Anderson. A list of candidate cancer biomarkers for targeted proteomics. *Biomarker Insights*, 1:117727190600100, January 2006. (cited on Page 6)
- [PC15] Witold Pedrycz and Shyi-Ming Chen, editors. *Information Granularity, Big Data, and Computational Intelligence*. Springer International Publishing, 2015. (cited on Page 17)
- [PF17] Bernardo A. Petriz and Octávio L. Franco. Metaproteomics as a complementary approach to gut microbiota in health and disease. *Frontiers in chemistry*, 5:4–4, Jan 2017. 28184370[pmid]. (cited on Page 1)
- [PM00] Akhilesh Pandey and Matthias Mann. Proteomics to study genes and genomes. *Nature*, 405(6788):837–846, June 2000. (cited on Page 8)
- [Ree88] W.Preston Reeves. Organic chemistry (Wade, L.G.Jr.). *Journal of Chemical Education*, 65(6):A169, June 1988. (cited on Page vii and 9)
- [RLLK06] E. S. Radisky, J. M. Lee, C.-J. K. Lu, and D. E. Koshland. Insights into the serine protease mechanism from atomic resolution structures of trypsin reaction intermediates. *Proceedings of the National Academy of Sciences*, 103(18):6835–6840, May 2006. (cited on Page vii, 11, and 12)

- [RZS18] David Broneske Wolfram Fenske Robert Heyer Sven Brehmer Dirk Bendorf Roman Zoun, Kay Schallert and Gunter Saake. Mstream: Proof of concept of an analytic cloud platform for near-real-time diagnostics on the smack stack for bruker mass spectrometry data. *Proceedings of the VLDB Endowment*, Vol. 12, 2018. (cited on Page vii, viii, 2, 3, 4, 15, 26, 33, 34, 49, and 59)
- [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*, 5(2):135–146, February 2008. (cited on Page 17)
- [Sar92] Raghupathy Sarma. Introduction to protein structure. carl branden , john tooze. *The Quarterly Review of Biology*, 67(1):44–45, 1992. (cited on Page 5)
- [SBG⁺08] Marc Sturm, Andreas Bertsch, Clemens Gröpl, Andreas Hildebrandt, Rene Hussong, Eva Lange, Nico Pfeifer, Ole Schulz-Trieglaff, Alexandra Zerck, Knut Reinert, and Oliver Kohlbacher. OpenMS – An open-source software framework for mass spectrometry. *BMC Bioinformatics*, 9(1):163, December 2008. (cited on Page 83)
- [SCH01] Mark R. Segal, Michael P. Cummings, and Alan E. Hubbard. Relating Amino Acid Sequence to Phenotype: Analysis of Peptide-Binding Data. *Biometrics*, 57(2):632–643, June 2001. (cited on Page 11)
- [SF13] Pramod J. Sadalage and Martin Fowler. *NoSQL distilled: a brief guide to the emerging world of polyglot persistence*. Addison-Wesley, Upper Saddle River, NJ, 2013. (cited on Page 30)
- [Sif80] Joseph Sifakis. Deadlocks and livelocks in transition systems. In P. Dembiński, editor, *Mathematical Foundations of Computer Science 1980*, volume 88, pages 587–600. Springer-Verlag, Berlin/Heidelberg, 1980. (cited on Page 29)
- [SM19] Jana Seifert and Thilo Muth. Editorial for Special Issue: Metaproteomics. *Proteomes*, 7(1):9, March 2019. (cited on Page 13)
- [SSM58] D. H. Spackman, W. H. Stein, and Stanford. Moore. Automatic recording apparatus for use in chromatography of amino acids. *Analytical Chemistry*, 30(7):1190–1206, July 1958. (cited on Page 1)

- [Tab15] David L. Tabb. The SEQUEST Family Tree. *Journal of The American Society for Mass Spectrometry*, 26(11):1814–1819, November 2015. (cited on Page 15)
- [UAK18] Saeed Ullah, M. Daud Awan, and M. Sikander Hayat Khiyal. Big data in cloud computing: A resource management perspective. *Scientific Programming*, 2018:1–17, 2018. (cited on Page 29)
- [VMJ16] Ankush Verma, Ashik Hussain Mansuri, and Neelesh Jain. Big data management processing with Hadoop MapReduce and spark technology: A comparison. In *2016 Symposium on Colossal Data Analysis and Networking (CDAN)*, pages 1–4, Indore, Madhya Pradesh, India, March 2016. IEEE. (cited on Page 28)
- [VRS⁺08] Nathan C Verberkmoes, Alison L Russell, Manesh Shah, Adam Godzik, Magnus Rosenquist, Jonas Halfvarson, Mark G Lefsrud, Juha Apalahti, Curt Tysk, Robert L Hettich, and Janet K Jansson. Shotgun metaproteomics of the human distal gut microbiota. *The ISME Journal*, 3(2):179–189, October 2008. (cited on Page 6)
- [Wam19] Dean Wampler. *Fast Data Architectures for Streaming Applications*. O’Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472., 2019. (cited on Page xiii, 4, 17, 19, and 21)
- [WGS⁺09] Alexander Wepf, Timo Glatter, Alexander Schmidt, Ruedi Aebersold, and Matthias Gstaiger. Quantitative interaction proteomics using mass spectrometry. *Nature Methods*, 6(3):203–205, February 2009. (cited on Page 6)
- [WLZZ14] Huajin Wang, Jianhui Li, Haiming Zhang, and Yuanchun Zhou. Benchmarking Replication and Consistency Strategies in Cloud Serving Databases: HBase and Cassandra. In Jianfeng Zhan, Rui Han, and Chuliang Weng, editors, *Big Data Benchmarks, Performance Optimization, and Emerging Hardware*, volume 8807, pages 71–82. Springer International Publishing, Cham, 2014. (cited on Page 31)
- [WM83] Ingrid Wagner and Hans Musso. New naturally occurring amino acids. *Angewandte Chemie International Edition in English*, 22(11):816–828, 1983. (cited on Page 5)
- [WS07] J. Throck Watson and O. David Sparkman. *Introduction to Mass Spectrometry*. John Wiley & Sons, Ltd, October 2007. (cited on Page 1)

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Magdeburg, den 17.07.2019