Applied Computer Science, vol. 18, no. 2, pp. 16–27
doi: 10.35784/acs-2022-10

Submitted: 2022-03-22 | Revised: 2022-05-16 | Accepted: 2022-05-23

Keywords: proteomics, mass spectrometry, distributed computing, Apache Spark

Katarzyna ORZECHOWSKA [0000-0001-9797-1142]*,
Tymon RUBEL [0000-0002-1342-414X]*, *Robert KURJATA* [0000-0001-8547-910X]*,
Krzysztof ZAREMBA [0000-0002-4036-6459]*

A DISTRIBUTED ALGORITHM FOR PROTEIN IDENTIFICATION FROM TANDEM MASS SPECTROMETRY DATA

Abstract

Tandem mass spectrometry is an analytical technique widely used in proteomics for the high-throughput characterization of proteins in biological samples. Modern in-depth proteomic studies require the collection of even millions of mass spectra representing short protein fragments (peptides). In order to identify the peptides, the measured spectra are most often scored against a database of amino acid sequences of known proteins. Due to the volume of input data and the sizes of proteomic databases, this is a resource-intensive task, which requires an efficient and scalable computational strategy. Here, we present SparkMS, an algorithm for peptide and protein identification from mass spectrometry data explicitly designed to work in a distributed computational environment. To achieve the required performance and scalability, we use Apache Spark, a modern framework that is becoming increasingly popular not only in the field of "big data" analysis but also in bioinformatics. This paper describes the algorithm in detail and demonstrates its performance on a large proteomic dataset. Experimental results indicate that SparkMS scales with the number of worker nodes and the increasing complexity of the search task. Furthermore, it exhibits a protein identification efficiency comparable to X!Tandem, a widely-used proteomic search engine.

1. INTRODUCTION

Liquid chromatography-tandem mass spectrometry (LC-MS/MS) is nowadays the method of choice in proteomics, a field of studies aimed at the large-scale analysis of proteins expressed in cells, tissues, and organisms (Aebersold & Mann, 2003; Hernandez, Müller & Appel, 2006). Most proteomic studies follow a "bottom-up" strategy, in which the proteins are first digested into smaller peptides using a proteolytic enzyme. The peptides are then separated by liquid chromatography, ionized, and passed to a tandem mass spectrometer where both the masses of the parent ions and the corresponding mass spectra of their

* Warsaw University of Technology, Institute of Radioelectronics and Multimedia Technology, Poland, orzechowska@ire.pw.edu.pl, trubel@ire.pw.edu.pl, r.kurjata@ire.pw.edu.pl, k.zaremba@ire.pw.edu.pl

fragments are measured. The resulting fragmentation mass spectra (MS/MS spectra) are sequence-specific and can be used for peptide identification. In typical proteomic experiments, from several thousand to tens of millions of MS/MS spectra are collected.

Due to high complexity and sizes ranging up to several gigabytes, tandem mass spectrometry data must be processed in an automated manner. In the currently most popular approach, peptide amino acid sequences are determined by comparing the measured mass spectra with theoretical spectra derived from a database of known proteins (Hernandez, Müller & Appel, 2006; Sadygov, Cociorva & Yates, 2004). To obtain a list of potential peptides, the database is subjected to an *in silico* digestion, in which protein sequences are cut at cleavage sites specific for the proteolytic enzyme used in the experiment. Depending on the database search parameters, this initial list may be further significantly expanded to consider non-specific enzyme cleavage and post-translational modifications (PTMs) of amino acid residues. Given the complete list of peptides, for each MS/MS spectrum a two-step procedure is repeated. First, a set of candidates is selected, comprised of peptides with masses falling into a tolerance window around the mass of the parent ion. Next, a theoretical fragmentation mass spectrum is generated for each candidate peptide and compared with the experimental one. The candidate with the highest value of a scoring function is considered the best match and is assigned to the examined spectrum. With peptide sequences identified, a list of proteins present in the studied samples can be inferred. Numerous search engines, both commercial and free, have been developed over the last years to implement this generic scheme of protein identification (Perkins et al., 1999; Craig & Beavis, 2004; Cox et al., 2011; Kim & Pevzner, 2014).

Because of the huge number of candidate peptides scored against the input spectra, the database-aided interpretation of LC-MS/MS data is a computationally expensive and resource-demanding task. This is especially true when variable (i.e. potential) post-translational modifications are considered, as they cause a combinatorial growth of the search space. On the other hand, the coarse-grained nature of the problem makes it almost perfectly parallel. Thus, not surprisingly, several strategies have been proposed to speed up proteomic searches. Most of the popular protein identification systems are designed for multithreaded execution. Moreover, hardware-accelerated processing has been suggested to take advantage of the massively parallel nature of modern graphics processing units (GPUs) (Milloy, Faherty & Gerber, 2012). Large-scale computing architectures have also been employed, including clusters with worker nodes communicating using message-passing protocols, such as MPI (Duncan, Craig & Link, 2005; Bjornson et al., 2008). Unfortunately, none of these approaches are fully satisfactory: multithreading is inherently limited by the resources of a single machine, GPU computations can only accelerate the scoring phase of the algorithm, and the necessity to repeat the database digestion independently on each node of the cluster affects the scalability of MPI-based implementations.

More recently, distributed computing frameworks have been proposed to address the increasing complexity of proteomic analyses. Frameworks such as Apache Hadoop (Dean & Ghemawat, 2008) and Spark (Zaharia et al., 2010) provide an effective way to parallelize computational tasks across large heterogeneous clusters or cloud environments. They also facilitate the development of distributed applications by handling network communication, job scheduling, and failure recovery. Hadoop was first used in proteomics to create a simple distributed version of X!Tandem, a popular open-source tool for protein identification (Pratt et al., 2012). Afterwards, a more advanced search engine, called Hydra, was introduced by Lewis et al. (2012). The algorithm of Hydra is implemented following the MapReduce

programming paradigm and is designed to scale with both the number of input spectra and the size of the proteins database. Yet, its performance and flexibility are limited by the execution model of Hadoop, primarily developed and optimized for the processing of on-disk data.

Here, we introduce SparkMS, a novel distributed algorithm based on Apache Spark. This state-of-the-art framework considerably improves the restrictive MapReduce model of Hadoop by allowing in-memory caching of data and iterative operation. Spark was already employed in bioinformatic analyses (Guo et al., 2018) and utilized in proteomics for the unbiased search of PTMs in spectral libraries (Horlacher, Lisacek & Müller, 2016). In this paper, we present its application to the problem of database-aided peptide and protein identification.

The main idea behind the development of SparkMS was to create a high-performance algorithm capable of processing vast amounts of data provided by comprehensive MS-based proteomic studies. In this regard, it was essential to ensure high scalability resulting from both the parallelization of all the database search steps and the effective use of the resources of the distributed computer system. Achieving such a goal was possible by fully utilizing the Spark platform features and the appropriate design of the algorithm workflow, which also employs data structures and procedures that allow for the minimization of network transfers. At the same time, unlike in the case of Hydra, great emphasis was placed on the algorithm's flexibility and ensuring the possibility of easy adaptation of its operation to various experimental designs and different measuring instruments.

2. METHODS

The presented proteomic database search algorithm is specifically developed to be compatible with the distributed execution model of the Apache Spark framework. Spark uses a master/slave architecture, where the master node runs a driver program, which dispatches computational tasks to executors on worker nodes.

The primary data structure in Spark is the resilient distributed dataset (RDD), a collection of immutable objects or key-value pairs which can be processed in parallel. The RDDs are automatically divided into logical partitions, and a different executor may manage each partition on a separate cluster node. Spark offers two types of operations on RDDs: transformations and actions. Transformations create a new dataset from an existing one by applying some function to its records, while actions return the processing results to the driver.

SparkMS follows the above-described execution model by converting the list of input MS/MS spectra and the proteins database into RDDs and performing a set of succeeding parallel transformations. To balance the workload and limit transfers between worker nodes, a specialized partitioning scheme is used. Moreover, an interval tree is employed in the protein digestion step to reduce memory usage and accelerate the search of candidate peptides. The flowchart of the algorithm is presented in Figure 1, and its details are discussed below.

2.1. Input Data and Parameters

The input data for SparkMS consists of an MGF (Mascot Generic Format) file with experimental MS/MS spectra and a protein sequences database in FASTA format. An additional XML configuration file allows the user to specify various search parameters, including

the most important ones: the proteolytic enzyme, maximum mass deviation (MMD) of parent and fragment ions, peptide fragmentation rules, and the list of PTMs selected from the UniMod database (Creasy & Cottrell, 2004).

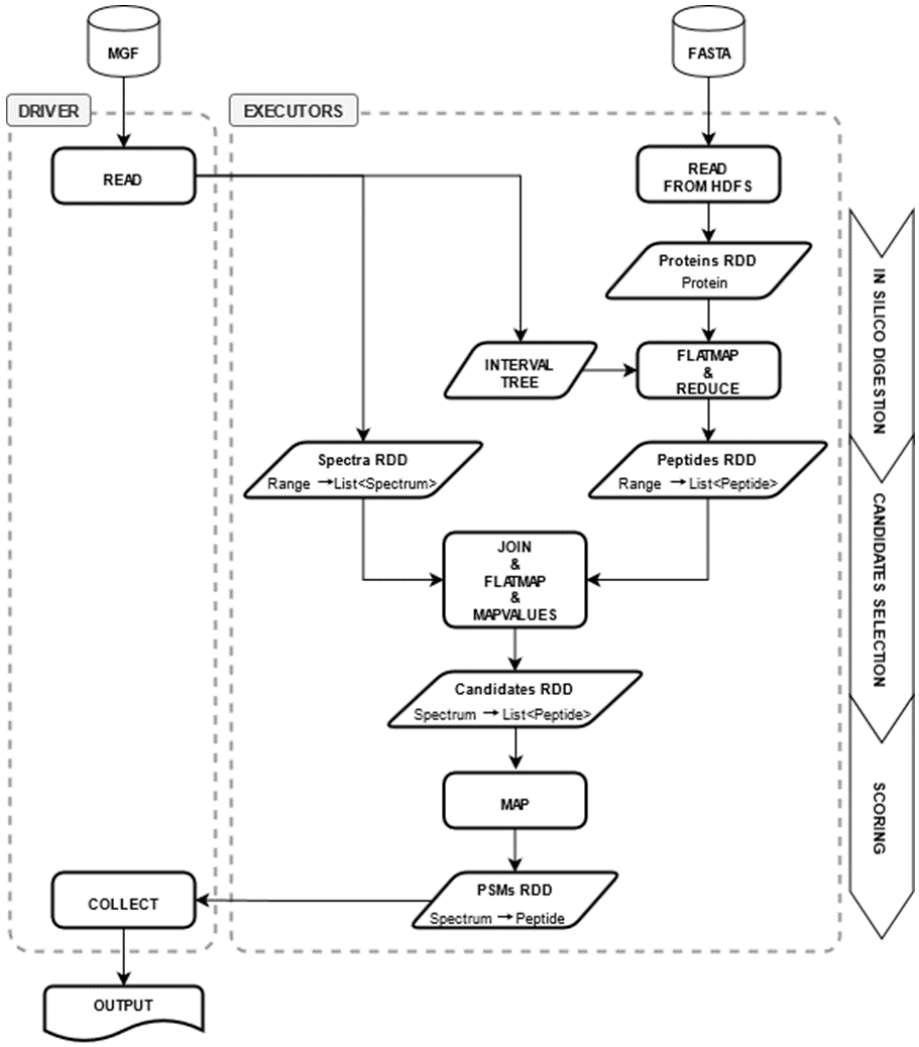

Fig. 1. Flowchart of SparkMS algorithm – the plot is divided into two parts representing sequential procedures performed on the master node (left frame) and operations executed in parallel on worker nodes (right frame) (on the right edge, the corresponding stages of the database identification process are presented; the notation *T→S* used in the description of the RDDs symbolizes *key→value* pairs with keys of type *T* and values of type *S*)

2.2. Creation of Resilient Distributed Datasets

Loading of the MGF input file takes place on the driver. Simultaneously the program creates an interval tree, i.e., a data structure that enables the efficient retrieval of intervals containing a given value. In our case, the tree stores the mass ranges of candidate peptides, calculated for each spectrum based on the parent ion mass and the user-defined MMD.

The tree is used to build a map of mass ranges (keys) and corresponding lists of experimental spectra (values), which is then parallelized to an RDD of pairs (depicted in Figure 1 as *Spectra RDD*). The interval tree itself is also provided to all executors as a broadcast variable and afterwards utilized in the *in silico* digestion step (see section 2.3).

The SparkMS implementation provides the ability to read the proteins database in FASTA format from both local or distributed filesystems. As a result, an RDD with protein records is created (*Proteins RDD* in Figure 1).

2.3. *In Silico* Protein Digestion

For each record in the proteins RDD, proteolytic peptides are generated according to the specificity of the selected enzyme and the list of considered PTMs. Isobaric modified variants of the same peptide are merged into single entries, as the exact locations of the PTMs are irrelevant for determining candidate sequences (location-specific variants will be generated only for candidate peptides before scoring – see section 2.4). To further reduce memory usage, all peptides are compared with the previously created interval tree, and only those found to fall within the mass range of at least one spectrum are stored.

The digestion step is implemented as a combination of two transformations: `flatMap` and `reduceByKey`. The first one maps proteins to their proteolytic peptides, while the purpose of the latter is to create an RDD of pairs with mass ranges as keys and aggregated lists of unique peptides as values (*Peptides RDD* in Figure 1). Both transformations subdivide the required computations among executors responsible for processing separate equal-sized partitions of the input proteins set.

2.4. Candidate Peptides Selection

In order to find candidate sequences, first, a `leftOuterJoin` is performed on the *Spectra RDD* and *Peptides RDD* using the mass range as a shared key. Next, a `flatMap` applied to the join result creates a new RDD of pairs (referenced as *Candidates RDD* in Figure 1), which assigns MS/MS spectra (keys) to lists of their candidate peptides (values). Finally, the lists of peptides are extended by a `mapValues` transformation to include sequence variants with all the possible PTM locations. The transformations are parallelized at the level of *spectrum→candidates* pairs, and a dedicated partitioner ensures even workload distribution among executors by equalizing the number of spectra in partitions. This partitioning scheme is also maintained in the scoring step of the algorithm.

2.5. Candidate Peptides Scoring

The scoring phase of the algorithm is realized by a single `map` transformation. The first task of the function applied in parallel to each entry of the *Candidates RDD* is to generate the theoretical spectra of candidate peptides following the fragmentation rules specified by the user. Next, the theoretical spectra are compared with the experimental one, and similarity scores are computed. The candidate with the highest score is assigned to the examined spectrum to form a peptide-spectrum match (PSM).

The spectra similarity is expressed in terms of a simple probabilistic score dependent on the number k of matches between the peaks of the experimental spectrum and the theoretical one (two peaks are deemed to be a match if the difference of their positions is less than the

allowed MMD of fragment ions). The score value is calculated as the probability of matching at least *k* out of the *n* theoretical peaks by pure chance, reported in logarithmic scale (Taus et al., 2011):

$$score = -10 \log_{10}[\sum_{i=k}^{n} \binom{n}{i} p^i (1-p)^{n-i}] \quad (1)$$

where: p – the probability of randomly matching a single experimental peak, estimated as:

$$p = \frac{Nd}{w} \quad (2)$$

where: N – the total number of measured peaks,
d – the fragment ions MMD,
w – the full mass range of the peaks in the experimental spectrum.

After scoring all spectra against their candidates, the resulting dataset (the *PSMs RDD* in Figure 1) is collected to the driver and saved in a human-readable text file.

3. RESULTS

The SparkMS algorithm has been implemented in the Java programming language and tested on a computer cluster. We have also implemented an alternative variant of the algorithm, which, to some extent, mimics the behaviour of the Hydra search engine (Lewis et al., 2012). This modified version (further denoted as "SparkMS (no IT)") does not use an interval tree to limit the number of peptides stored in memory after *in silico* digestion. Instead, the complete set of proteolytic peptides is preserved, divided into constant-width mass bins, and co-grouped by mass range with the input spectra to eventually form the lists of candidate sequences. Both implementations share the same source code of basic procedures (such as protein digestion, scoring function, etc.), thus they allow for an unbiased comparison of different approaches to database search on a common distributed computing platform. The two variants of SparkMS have been evaluated in regard to their protein identification performance, horizontal scalability, understood as the ability to efficiently use additional computational resources, and scalability with increasing task complexity (expressed in the number of scored candidate sequences).

3.1. Test Data and Settings

The computing infrastructure for the tests consisted of a Spark/HDFS cluster (Apache Spark version 2.3.0, Hadoop 2.9.0) composed of 6 workers and one master node connected with gigabit Ethernet. Each node was equipped with two 4-core AMD Opteron 2384 processors and 16 GB of RAM. The Spark cluster manager was configured to lunch two independent executors per node, assigning 4 cores and 6 GB of RAM to each executor. The master node was running the Spark driver process with 12 GB of RAM available.

Test data originated from a recently published study on the proteome composition of pancreatic cyst fluid (Paziewska et al., 2018). The dataset consisting of 715 972 high-resolution MS/MS spectra is available in the PRIDE repository (Vizcaíno et al., 2016) under the ID: PXD005248. The experimental spectra were searched against a database containing

173 843 target human proteins from the release 2020.06 of the UniProt/TrEMBL database (UniProt Consortium, 2019) and the same number of reversed decoy entries. Such an approach, commonly referred to as the target/decoy strategy, enables the estimation of the false discovery rate (FDR) of protein identification results (Käll et al., 2008). The search parameters were as follows: enzyme specificity – trypsin; maximum number of missed cleavages – 2; parent ions MMD – 10 ppm; fragment ions MMD – 0.01 Da; fixed PTMs – methylthiolation of C; variable PTMs – oxidation of M and acetylation of K (unless otherwise stated in the description of a particular test), and at most 3 modified sites per peptide were allowed.

For comparison purposes X!Tandem (version 2015.12.15.2) was used. The output files of both SparkMS and X!Tandem were post-processed with MScan, a software tool available at http://proteom.ibb.waw.pl/mscan.

3.2. Peptide and Protein Identification Performance

After searching against the TrEMBL database, SparkMS assigned sequences to 80 721 spectra (out of 715 972, 11.3% of the dataset) with an FDR equal to 0.01. The created peptide-spectrum matches (PSMs) corresponded to 3 548 unique peptides originating from 1 231 proteins, of which 923 were represented by at least two peptides. To ensure complete consistency, the sets of identified PSMs, peptides, and proteins were cross-checked between the two versions of the distributed algorithm and an additional single-threaded implementation. The results were also confronted with those obtained from X!Tandem, executed with corresponding parameters on the same input data. The comparison indicated a generally similar identification performance for both search engines. As it is presented in table 1, SparkMS created 0.4% fewer PSMs than X!Tandem, but simultaneously, it detected 0.8% more unique peptides and 3.4% more proteins. Noteworthy, we observed a significant overlap between the results of X!Tandem, and SparkMS, as 83.2% of the total number of peptides and 73.5% of proteins were common for the two search engines. Such a degree of commonality is typical for the comparison of different proteomic database-aided identification systems (Paulo, 2013).

Tab. 1. Comparison of SparkMS and X!Tandem in terms of the numbers of PSMs, peptides and proteins identified at an FDR threshold of 0.01.

Search engine	PSMs	Peptides	Proteins
X!Tandem	81071	3534	1189
SparkMS	80721	3548	1231

3.3. Horizontal Scalability

On a single 8-core machine, SparkMS completed the database search in 1 583 seconds, and it was 2.57 times faster than its modified version, which does not employ the interval tree (Figure 2). Moreover, it outperformed by a factor of 1.33 the X!Tandem search engine running on the same machine. With the adding of worker nodes, the execution time decreased, ultimately leading to a 4.81-fold speedup on the 6-node cluster versus a single machine. An analysis of Apache Spark runtime statistics indicated that the speedup was to some

degree limited by the relatively low network throughput of the test cluster. So, it is possible that the horizontal scalability could be improved by simply replacing the gigabit ethernet with a faster connection.

The two compared variants of SparkMS presented almost equal horizontal scalability, however, the version with the interval tree was on average 2.56 times faster. Since the numbers of protein records and scored peptides are equal in both cases, the acceleration is mostly due to a more effective candidate sequences selection. The interval tree also reduced the amount of data stored after *in silico* digestion (17.7% fewer peptides stored) and subsequently shuffled between the executors (data transfer reduced by 34.0%).

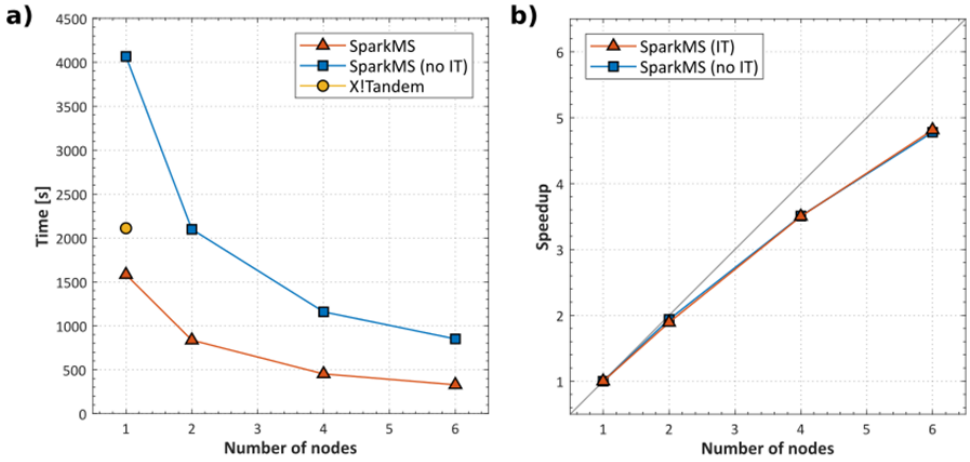

Fig. 2. Execution time in seconds (a) and speedup (b) with the number of worker nodes increasing from 1 to 6 (8 to 48 cores) – the plot presents the results of both SparkMS versions: with and without the interval tree (execution time of X!Tandem on a single node is also presented for reference; points are averaged over 5 test runs)

3.4. Scalability with Growing Task Complexity

To validate SparkMS against larger search spaces, we ran a series of tests with the number of variable PTMs increasing from none to 8 (apart from the two PTMs used in previous tests, six more were included: phosphorylation of S/T and Y, methylation of R and K, and dimethylation of R and K). This parameter greatly influences the number of possible candidate peptides, and therefore, it is well suited for verifying the ability of the algorithm to handle computationally intensive tasks. The test results are presented in Figure 3.

As expected, expanding the list of variable PTMs caused a highly nonlinear growth of task complexity. Consequently, the time needed to accomplish the database search with 8 PTMs reached 2 hr 48 min on a 48-core cluster. The speedup over a single 8-core node was 4.72, which is a value very close to the one observed in section 3.3. This indicates that the horizontal scalability of SparkMS remains unaffected even for high-complexity tasks. Notably, the relationship between the number of scored peptides and the search time was almost perfectly linear ($R^2 > 0.99$) for both SparkMS variants. As was the case earlier, the interval tree considerably accelerated the search (from 1.68 to 2.58 times). Separate tests revealed that similar conclusions also apply to the algorithm's scalability with the increasing number of input spectra.

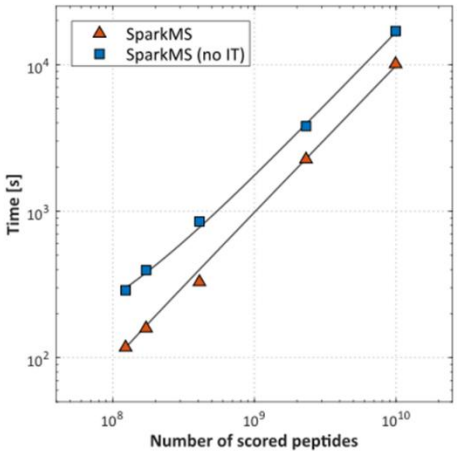

Fig. 3. Execution time on a cluster with 6 worker nodes (48 cores) as a function of task complexity, expressed in the number of scored candidate peptides (points are averaged over 5 test runs; grey lines represent linear regression fits)

4. DISCUSSION

The basic idea of the SparkMS algorithm is somewhat similar to the one used in Hydra, a previously presented search engine build on Hadoop. First and foremost, both solutions are designed to operate on clusters of heterogeneous hardware managed by distributed computing frameworks. Therefore, they inherit features such as horizontal scalability and fault tolerance. They also enable the parallelization of all stages of the protein identification process – this is a clear advantage over MPI-based or GPU-accelerated approaches.

However, despite certain similarities between the two search engines, there are also essential differences in both the algorithm design and the used programming framework. Hydra divides the search into two independent tasks, separated by disk write and read operations. In the first task, all the possible proteolytic peptides are generated, binned by mass value, and saved to a distributed filesystem. The second task is aimed at scoring the input spectra against the formerly stored peptides. Both steps are implemented as series of MapReduce operations with intermediate results serialized to disk. Such behaviour is consistent with the Hadoop workflow but is not necessarily optimal for proteomic searches.

In contrast to Hydra, SparkMS is prepared for in-memory operation, and it does not require any time-consuming disk access, aside from reading the input data and writing the results. The algorithm also does not rely upon a precalculated list of proteolytic peptides (which in the case of Hydra must be recreated each time the search parameters are changed) or any additional assumptions limiting its flexibility. In fact, its implementation is fully parameterized and can be easily adapted by the user to any specific experimental design or mass spectrometer. Moreover, the algorithm's performance and scalability benefit from the lower overheads of the Spark platform in respect to Hadoop. Further acceleration is achieved by using a custom partitioning scheme and an interval tree to select candidate peptides.

SparkMS was tested on a computer cluster against a large real-world proteomic dataset. On a single 8-core machine, SparkMS achieved an execution time comparable to X!Tandem, a standard multi-threaded search engine (notably, both programs also exhibited a similar

peptide and protein identification performance). However, the distributed algorithm provided an added value in the ability to efficiently utilize multiple worker nodes. Its horizontal scalability allowed to shorten the calculations approximately 4.8 times on a 6-node cluster, and the speedup did not deteriorate with increasing task complexity. Moreover, the observed execution times scaled linearly with the number of candidate peptides scored against the input spectra. Thus, it is possible to assume that SparkMS would accommodate even larger datasets/databases and more exhaustive search parameters. The test also clearly demonstrated the gain in speed and memory consumption resulting from employing an interval tree for candidate sequences selection, instead of using a straightforward join of mass-binned spectra and peptides, as proposed in Hydra.

The performed experiments confirmed the potential of the Spark-based algorithm to deal with proteomic searches of various sizes, including those involving the processing of vast amounts of data. However, it should be noted that the presented computational strategy also has some limitations. Specifically, in a distributed environment, the actual speedup is a function of task complexity, the number of worker nodes, and network throughput. For small inputs or restricted search parameters, the initialization overhead and the finite speed of data transfer are likely to limit the scalability, especially when the cluster size increases. Therefore, the distributed approach is best suited for computationally intensive jobs.

5. CONCLUSIONS

With constantly growing sizes of protein databases and the broader availability of fast and sensitive high-resolution spectrometers, the LC-MS/MS data processing is increasingly becoming a “big data” problem that requires an efficient and highly scalable computational strategy. To address this issue, we developed SparkMS, a new distributed algorithm for the database-aided peptide and proteins identification. The proposed algorithm takes full advantage of the Apache Spark framework to effectively parallelize all the stages of the identification process. Experimental results demonstrated that SparkMS scales with both the increasing task complexity and the number of worker nodes. Therefore, given a big enough cluster infrastructure, the algorithm is ready to perform fast and comprehensive proteomic searches, leading to a better insight into the biological problems under study.

Although the paper presents a fully functioning proteomic search engine, there is still room to improve the underlying algorithm. Its further development will primarily concern increasing the efficiency of using the cluster's computing resources. This will be achieved by introducing a more advanced data partitioning scheme in the scoring stage. The new partitioner will ensure equal numbers of candidate peptides (instead of spectra, as in the current version) in the partitions, allowing for shortened runtimes and increased scalability due to better load balancing between worker nodes.

Work is also underway to introduce an additional phase to the algorithm related to the statistical evaluation of peptide-spectra matches and to use machine learning to improve identification performance. The preliminary results of the proposed support vector machine demonstrated its ability to increase the number of correctly identified peptides at a given FDR threshold, albeit optimization is still required to exploit the full potential of the Apache Spark platform (Orzechowska & Rubel, 2021).

Finally, it is worth noting that an interesting direction of further research is the adaptation of SparkMS to the processing of data from experiments using mass spectrometry combined with cross-linking to study protein-protein interactions in complexes (Rappsilber, 2011). It is a problem of growing importance for proteomics and structural biology, the solution of which requires high computational effort, which makes it well suited to the distributed environment.

REFERENCES

Aebersold, R., & Mann, M. (2003). Mass spectrometry-based proteomics. *Nature*, *422*(6928), 198–207. https://doi.org/10.1038/nature01511

Bjornson, R. D., Carriero, N. J., Colangelo, C., Shifman, M., Cheung, K. H., Miller, P. L., & Williams, K. (2008). X!!Tandem, an improved method for running X!tandem in parallel on collections of commodity computers. *Journal of proteome research*, *7*(1), 293–299. https://doi.org/10.1021/pr0701198

Cox, J., Neuhauser, N., Michalski, A., Scheltema, R. A., Olsen, J. V., & Mann, M. (2011). Andromeda: a peptide search engine integrated into the MaxQuant environment. *Journal of proteome research*, *10*(4), 1794–1805. https://doi.org/10.1021/pr101065j

Craig, R., & Beavis, R. C. (2004). TANDEM: matching proteins with tandem mass spectra. *Bioinformatics (Oxford, England)*, *20*(9), 1466–1467. https://doi.org/10.1093/bioinformatics/bth092

Creasy, D. M., & Cottrell, J. S. (2004). Unimod: Protein modifications for mass spectrometry. *Proteomics*, *4*(6), 1534–1536. https://doi.org/10.1002/pmic.200300744

Dean, J., & Ghemawat, S. (2008). MapReduce: Simplified Data Processing on Large Clusters. *Communications of the ACM*, *51*(1), 107–113. https://doi.org/10.1145/1327452.1327492

Duncan, D. T., Craig, R., & Link, A. J. (2005). Parallel tandem: a program for parallel processing of tandem mass spectra using PVM or MPI and X!Tandem. *Journal of proteome research*, *4*(5), 1842–1847. https://doi.org/10.1021/pr050058i

Guo, R., Zhao, Y., Zou, Q., Fang, X., & Peng, S. (2018). Bioinformatics applications on Apache Spark. *GigaScience*, *7*(8), giy098. https://doi.org/10.1093/gigascience/giy098

Hernandez, P., Müller, M., & Appel, R. D. (2006). Automated protein identification by tandem mass spectrometry: issues and strategies. *Mass spectrometry reviews*, *25*(2), 235–254. https://doi.org/10.1002/mas.20068

Horlacher, O., Lisacek, F., & Müller, M. (2016). Mining Large Scale Tandem Mass Spectrometry Data for Protein Modifications Using Spectral Libraries. *Journal of proteome research*, *15*(3), 721–731. https://doi.org/10.1021/acs.jproteome.5b00877

Käll, L., Storey, J. D., MacCoss, M. J., & Noble, W. S. (2008). Assigning significance to peptides identified by tandem mass spectrometry using decoy databases. *Journal of proteome research*, *7*(1), 29–34. https://doi.org/10.1021/pr700600n

Kim, S., & Pevzner, P. A. (2014). MS-GF+ makes progress towards a universal database search tool for proteomics. *Nature communications*, *5*, 5277. https://doi.org/10.1038/ncomms6277

Lewis, S., Csordas, A., Killcoyne, S., Hermjakob, H., Hoopmann, M. R., Moritz, R. L., Deutsch, E. W., & Boyle, J. (2012). Hydra: a scalable proteomic search engine which utilizes the Hadoop distributed computing framework. *BMC bioinformatics*, *13*, 324. https://doi.org/10.1186/1471-2105-13-324

Milloy, J. A., Faherty, B. K., & Gerber, S. A. (2012). Tempest: GPU-CPU computing for high-throughput database spectral matching. *Journal of proteome research*, *11*(7), 3581–3591. https://doi.org/10.1021/pr300338p

Orzechowska, K., & Rubel, T. (2021). An SVM-based peptide identification algorithm integrated into a database search engine. *Proceedings of the XXII Polish Conference on Biocybernetics and Biomedical Engineering.*

Paulo, J. A. (2013). Practical and Efficient Searching in Proteomics: A Cross Engine Comparison. *WebmedCentral*, *4*(10), WMCPLS0052. https://doi.org/10.9754/journal.wplus.2013.0052

Paziewska, A., Polkowski, M., Rubel, T., Karczmarski, J., Wiechowska-Kozlowska, A., Dabrowska, M., Mikula, M., Dadlez, M., & Ostrowski, J. (2018). Mass Spectrometry-Based Comprehensive Analysis of Pancreatic Cyst Fluids. *BioMed research international*, *2018*, 7169595. https://doi.org/10.1155/2018/7169595

Perkins, D. N., Pappin, D. J., Creasy, D. M., & Cottrell, J. S. (1999). Probability-based protein identification by searching sequence databases using mass spectrometry data. *Electrophoresis*, *20*(18), 3551–3567. https://doi.org/10.1002/(SICI)1522-2683(19991201)20:18<3551::AID-ELPS3551>3.0.CO;2-2

Pratt, B., Howbert, J. J., Tasman, N. I., & Nilsson, E. J. (2012). MR-Tandem: parallel X!Tandem using Hadoop MapReduce on Amazon Web Services. *Bioinformatics (Oxford, England)*, *28*(1), 136–137. https://doi.org/10.1093/bioinformatics/btr615

Rappsilber, J. (2011). The beginning of a beautiful friendship: Cross-linking/mass spectrometry and modelling of proteins and multi-protein complexes. *Journal of Structural Biology*, *173*(3), 530–540. https://doi.org/10.1016/j.jsb.2010.10.014

Sadygov, R. G., Cociorva, D., & Yates, J. R., 3rd (2004). Large-scale database searching using tandem mass spectra: looking up the answer in the back of the book. Nature methods, 1(3), 195–202. https://doi.org/10.1038/nmeth725

Taus, T., Köcher, T., Pichler, P., Paschke, C., Schmidt, A., Henrich, C., & Mechtler, K. (2011). Universal and confident phosphorylation site localization using phosphoRS. *Journal of proteome research*, *10*(12), 5354–5362. https://doi.org/10.1021/pr200611n

UniProt Consortium. (2019). UniProt: a worldwide hub of protein knowledge. *Nucleic acids research*, *47*(D1), D506–D515. https://doi.org/10.1093/nar/gky1049

Vizcaíno, J. A., Csordas, A., Del-Toro, N., Dianes, J. A., Griss, J., Lavidas, I., Mayer, G., Perez-Riverol, Y., Reisinger, F., Ternent, T., Xu, Q. W., Wang, R., & Hermjakob, H. (2016). 2016 update of the PRIDE database and its related tools. *Nucleic acids research*, *44*(22), 11033. https://doi.org/10.1093/nar/gkw880

Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., & Stoica, I. (2010). Spark: Cluster Computing with Working Sets. *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing* (*HotCloud'10*). USENIX Association.