4

Cloud Parallel Processing of Tandem Mass Spectrometry-based Proteomics

Yassene Mohammed,[1,2] Ekaterina Mostovenko,[1] Alex A. Henneman,[1] Rob J. Marissen,[1] André M. Deelder,[1] and Magnus Palmblad[1]

[1] Biomolecular Mass Spectrometry Unit, Department of Parasitology, Leiden University Medical Center, The Netherlands

[2] Distributed Computing Security Group and L3S, University of Hannover, Germany

Journal of Proteome Research 2012, 11 (10), 5101–5108

ABSTRACT

Data analysis in mass spectrometry-based proteomics struggles to keep pace with the advances in instrumentation and the increasing rate of data acquisition. Analyzing this data involves multiple steps requiring diverse software, using different algorithms and data formats. Speed and performance of the mass spectral search engines are continuously improving, although not necessarily as needed to face the challenges of acquired big data. Improving and parallelizing the search algorithms is one possibility, data decomposition presents another, simpler strategy for introducing parallelism. We describe a general method for parallelizing identification of tandem mass spectra using data decomposition that keeps the search engine intact and wraps the parallelization around it. We introduce two algorithms for decomposing mzXML files and recomposing resulting pepXML files. This makes the approach applicable to different search engines, including those relying on sequence databases and those searching spectral libraries. We use cloud computing to deliver the computational power and scientific workflow engines to interface and automate the different processing steps. We show how to leverage these technologies to achieve faster data analysis in proteomics and present three scientific workflows for parallel database as well as spectral library search using our data decomposition programs, X!Tandem and SpectraST.

INTRODUCTION

Mass spectrometry (MS), particularly tandem mass spectrometry (MS/MS), is currently the most used method for identifying unknown proteins present in biological samples. Advances in instrumentation have reduced acquisition time and increased resolution and sensitivity, which in combination with complementary fragmentation mechanisms[1, 2] and high resolving-power mass analyzers in both MS and MS/MS[3-5] have led to very complex data. This has brought new challenges to proteomics, i.e. how do we store and process these large data volumes. Standard desktop computers often cannot process data at the rate it is being generated, creating an additional bottleneck in the analysis pipeline. The analysis of the mass spectrometry data typically involves several steps. One essential and computationally expensive step is peptide identification, i.e. the mapping of each spectrum to a unique peptide or one or more peptides. In this manuscript we describe a method of handling mass spectrometry "big data" by outsourcing computationally intensive tasks using off-the-shelf open source tools and in-campus cloud resources. We introduce a method for parallelizing common search engines like X!Tandem and SpectraST that are part of the Trans-Proteomic Pipeline (TPP)[6], which can also work for most other available search engines. We show how peptide identification speed using workflow engines, cloud computing, and a new data decomposing/recomposing algorithm can easily be improved by a significant factor. In our tests we reached more than 30-fold speed improvement comparing X!Tandem running locally (one core) with the same program running on the cloud and a 7-fold improvement for the SpectraST spectral library search.

METHODS

One important step in the processing pipeline of mass spectrometry data is associating a particular (tandem) mass spectrum with a peptide sequence. There are three types of search engines for peptide identification, i.e. database, library, and *de novo*. Database search engines, like Mascot[7], SEQUEST[8], or X!Tandem[9], compare each spectrum obtained from the sample with theoretical spectra generated from a list of predicted peptides. The predicted peptides list is ideally derived from all of the protein sequences that could be expressed in the experiment sample. Library search

engines, like SpectraST[6] or X!Hunter[10] assume that the fragmentation of a particular molecule in a mass spectrometer is partially reproducible between analyses and instruments. One can therefore generate a library of ion fragmentation spectra with each spectrum being associated with a corresponding molecular structure. A library search engine assigns a specific structure to an experimental spectrum by comparing it with the entries in the library.

Peptide identification using a search engine is a main processing bottleneck in mass spectrometry based proteomics. A normal search of 30,000 spectra could take up to 40 minutes on common modern desktop with a 4-core processor, depending on the search parameters. In many cases this is impractical for scientists, especially if they want to include more modifications in their searches, which can significantly increase the search space. Enhancing search engine speed besides developing search algorithms for high performance computing environment are continuously under development.[11-14] While making faster algorithms is a main objective of several groups[14-17], we are only targeting the data itself leaving the search engine intact. This makes the approach applicable to many search engines. Search engines are legacy software that have gained acceptance and usability in the proteomics community and we therefore prefer to consider them as black boxes and not modify them in any way, but instead wrap the parallelization around them. In the rest of the section we describe the data formats used, the new decomposition and recomposition algorithms, the processing pipelines and how to scale these up to scientific workflows.

Data formats

To build on other efforts, such as the TPP[6], we chose to use common XML formats such as mzXML[18, 19] for input and pepXML[20] as output. mzXML and pepXML are two *de facto* open format standards still used for mass spectrometry data. Converters from almost any other format to mzXML or pepXML can be obtained.[19-21] Extension of our method to mzML[22] is also feasible, as only the logic in the data decomposition algorithm needs to be modified with no further changes. Using open standard formats maintains compatibility with other efforts and existing pipelines and avoids making this work an isolated solution.

Data decomposition and recomposition

Decomposition involves breaking down a complex system into smaller pieces. It is the basis for finding the tasks that can run concurrently in parallel applications. There are two major decomposition methods in parallel programming, i.e. functional and data decomposition.[23] Data decomposition is used more often and it depends mainly on the developer's knowledge about the data and how an algorithm processes the data. In order to facilitate parallelism of peptide identification of mass spectrometry data we developed two algorithms for decomposing and recomposing the inputs and outputs of an arbitrary search engine. The only assumption made is that each spectrum will be processed by the search engine independently from other spectra. This is true for many search algorithms, but not subsequent validation steps, such as PeptideProphet[24] and Percolator.[25] However, the latter are not nearly as computationally expensive as the initial peptide-spectrum matching. The search engines we used to demonstrate our parallelization approach, i.e. X!Tandem without model refinement[26] and SpectraST process each spectrum independently. OMSSA[27], MS-GFDB/MS-GF+[1] and Crux/Tide[16, 17] are other search engines that could also be parallelized in this way.

Processing Pipelines and Scientific Workflows

There are multiple software packages that allow stepwise processing of mass spectrometry data, such as TPP[6], Proteomatic[28] and ProteoWizard.[29] In this sense, processing pipelines and workflows are overloaded terms, and sometimes used synonymously. We use processing pipelines to refer to a multistep sequential processing of one dataset at a time, in which transitions from one step to the next happen with some manual interaction as in the TPP. Scientific workflows involve concurrency and parallel processing capabilities, in which the transition between the processing steps can happen automatically or with breakpoints according to the workflow design. Scientific workflow engines like Galaxy[30], Moteur[31], Kepler[32], and Taverna[33] were introduced in the last decade to facilitate interfacing modular processing steps, automating analysis pipelines, scaling them up to workflows, and make analyses reproducible and sharable. We have previously described[34] how Taverna can be used to automate analysis workflows in mass spectrometry based proteomics on a local machine. We also demonstrated how workflow and data decomposition can scale up processing pipelines to run in high performance computing environments.[23]

Here we use Taverna 2.4 to build our processing workflows and to perform job orchestration, i.e. to manage data and software transfer to and from the cloud. In this respect, we use Taverna not only as a workflow manager, but also as a technical enabler to build our *adhocratic*[35, 36] experiment oriented distributed computing environment using in-campus clouds. Taverna offers various kinds of processors.[33, 37] Scientists can chose between WSDL web services, Beanshell processors, REST Web services, Rshell processors, Tools and XPath processors. Details about these processors and how to use them can be found in literature[33] as well as in the Taverna documentation.[37] In the following we highlight the two processor types that are important for our implementation.

Beanshell processors enable executing small Java code snippets as part of a workflow. Typically they are used for small tasks like simple file and data manipulation, parsing and formatting, saving to a local directory, calling local program, interacting with the user, etc. Tool processors are very suitable to call commands in a shell on any machine, to which Taverna can obtain an SSH connection - including the local machine. We mainly use Beanshell processors to launch software with their correct inputs locally, and Tool processors to interact with the cloud resources, upload data, and retrieve results. We used cloud resources based on the open source cloud middleware OpenNebula.[38] These cloud resources are freely available for academic research users in the Netherlands. Such resources are common in various universities. A cloud environment in regard to our method can include any machine, to which Taverna could have an SSH connection.

Used Datasets for Testing

In order to profile our method and compare it with the local run of the used search engines we ran multiple tests from realistic database search scenarios. For these tests we used two ion trap datasets; the first consisted of 5 LC-MS/MS datasets from tryptically digested human serum samples and the second of LC-MS/MS data from 20 fractions of one *E. coli* whole cell lysate, also digested with trypsin. All data was acquired on amaZon ion trap mass spectrometer (Bruker Daltonics, Bremen, Germany). The five human datasets each contains around 27,000 spectra whereas the 20 *E. coli* datasets each contains around 10,600 spectra (see Table 4.1 and Figure 4.6). In the X!Tandem search, strict tryptic cleavage specificity were assumed (C-terminally or R and K, not N-terminally of P), the precursor mass measurement error tolerance -0.5 to 2.5 Da, 2 missed enzymatic cleavage

allowed, and carbamidomethylation as the only and fixed modification. Phosphorylation as variable modification and semi-tryptic cleavage were also considered in the performance tests. In the SpectraST search, average masses instead of monoisotopic masses were used and precursor mass measurement error tolerance of 3 Th. All other parameters for X!Tandem and SpectraST were as the defaults in the TPP package. For the X!Tandem

	One sample (human)		5 samples (human)		20 samples (*E. coli*)	
Size of file(s)	113.8 MB		565.8 MB		1,540 MB	
Number of spectra	27,436		139,211		212,141	
Search engine	**X!Tandem**	**SpectraST**	**X!Tandem**	**SpectraST**	**X!Tandem**	**SpectraST**
Size of database/ library	35.6 MB[40]	2,123 MB[41]	35.6 MB[40]	2,123 MB[41]	1.75 MB[39]	303 MB[42]
Number of protein/spectra entries	70,254[40]	310,688[41]	70,254[40]	310,688[41]	4,303[39]	50,369[42]
Wall time monolithic running locally[1] (1 core / 4 cores) in min:sec	39:32 / 10:17	27:43	191:53 / 55:13	52:32	40:16/ 28:09	43:22
Wall time[2] parallel running on the cloud in min:sec	2:15	4:17	5:42	7:09	10:34	11:31
Speedup in fold	18 / 4.6	6.8	34 / 10	7	3.8/2.7	3.8

[1] The used system to run all the local experiments was an HP Elite 8200 computer with Windows® 7 Enterprise 64bit operating system, Intel® i7-2600 processor running at 3.40 GHz, and 8 GB of RAM.

[2] Wall time here refers to the actual time experienced by the user, i.e. the time needed to decompose, transfer, analyze and recompose data, starting with the spectra in mzXML file(s) on the user local computer and ending with the peptide identification in pepXML format stored in the same directory as the mzXML file.

Table 4.1. Performance tests of the described method comparing elapsed time for analyzing multiple input datasets.

search, the used databases for the human serum and for *E. coli* datasets were retrieved from UniProt.[39, 40] The spectral libraries for human and *E. coli* from National Institute of Standards and Technology (NIST) were used for the SpectraST searches.[41, 42]

Related Work

Duncan et al. have developed a parallel version of a X!Tandem for Message Passing Interface (MPI) enabled cluster.[11] It is beneficial to run a search engine on a cluster using MPI in terms of speed; this demands anyhow the availability of an MPI enabled server/cluster to the scientist. Pratt et al.[13] developed a cloud parallel peptide identification using parallel X!Tandem[11], Hadoop[43, 44], and MapReduce.[45, 46] They used a similar approach to X!!Tandem[12] in extending X!Tandem's threading onto a network, but used Hadoop and MapReduce instead of MPI. Their implementation is meant for Amazon Elastic Cloud and they achieved speedup of 31-fold using 200 Amazon cloud instances (corresponding to processing unit or a core). The current TPP version allows outsourcing X!Tandem searchs to Amazon Elastic Cloud to run multiple searches at the same time. We are not aware of any parallel implementation of SpectraST, but Baumgardner et al. have recently implemented their own spectral library search algorithm for GPUs using CUDA.[14] Our goal is to achieve data parallelism to accelerate peptide identification while preserving the search engine without any modification to its code. In principle, this makes the solution compatible also with closed-source algorithms.

RESULTS AND DISCUSSION

The employed technologies can be divided into three categories: data decomposition, cloud computing and scientific workflow engines. Data decomposition/recomposition is the parallelization *enabler*. The virtual and physical computers in the cloud delivers the processing and storage power. Finally, scientific workflows are used to imbed the logic of the data analysis into interfaced processing steps, to scale analysis pipelines up to workflows, and to orchestrate the parallel processing. In the following we explain how we are leveraging these technologies in our implementation.

Data decomposition and recomposition algorithms

Our decomposition algorithm splits an mzXML file into multiple smaller syntactically correct mzXML files. Syntactically correct here means that each daughter file is itself a valid mzXML file according to the mzXML schema.[19] The requested number of daughter files is passed to the algorithm as an input. Typically, LC-MS or LC-MS/MS datasets incorporate many low quality (information-poor) spectra; particularly at the beginning and near the end of the chromatographic gradient, while the good (information-rich) spectra are concentrated in the middle of the chromatographic run. Simply dividing the data in equal and sequential time intervals would therefore be suboptimal, as the early and late time intervals contains many spectra that would be immediately filtered out by the search engine. These data subsets would therefore process much faster than subsets from the middle of the gradient. To avoid this, we designed the algorithm to distribute the spectra from the original mzXML file randomly to all daughter files. This is an ad hoc approach to distribute good and bad spectra in order to divide the computational load evenly over the processing nodes. This also makes the method scaleable and independent on the chromatographic gradient and experimental design. Our data recomposition algorithm takes multiple pepXML files and composes them into one pepXML file. The algorithm takes into account the different original naming of the file and corrects the scan numbers to make the composed pepXML file schematically correct.[20] Both algorithms are written in Java and are available on ms-utils.org/decomposition.

Cloud computing

We used a dedicated infrastructure for cloud computing at SARA.[47] The infrastructure runs on OpenNebula cloud middleware. The instances/workers we used were minimal Ubuntu 11.04 server 64-bit virtual machines with Oracle (Sun) Java 6 build 1.6.0_26 installed. Depending on the workflow, a number of identical images can be initiated and used. For our tests we always used 8 instances, each of 8 virtual CPUs. Currently starting the workers from the workflow using OpenNebula Cloud Computing Interface services[48] is not permitted due to the security policy of the provider. All necessary software to run a workflow, for instance the search engines, will be deployed on the target machine from within the workflow. This keeps the cloud instances lightweight and the workflows easier to update and adjust to the target cloud architecture. In case one

prefers another version of the search engine, or using a 32-bit server, only the corresponding executable has to be provided as an input to the workflow.

Scientific workflows

The minimal workflow consists of three main processors: the mzXML decomposer, a search engine, and the pepXML composer (see Figure 4.1 and 4.2). One extra processor is needed to uncompress (unzip) the downloaded data from the cloud. Moving compressed (zipped) data between the cloud and the local machine and vice versa reduces the latency regarding the network speed. This is very helpful when the data is in ASCII format and can be compressed down to 68% of its original size like in mzXML and pepXML formats. The NIST spectral libraries can be compressed down to 32% of their original size. Each workflow processor includes the needed logic to run the corresponding program from the command line. The firing mechanism in Taverna is the availability of the data on the inputs of each processor. Taverna takes care of transferring the data between the processors. The data decomposing/recomposing processors are Beanshell

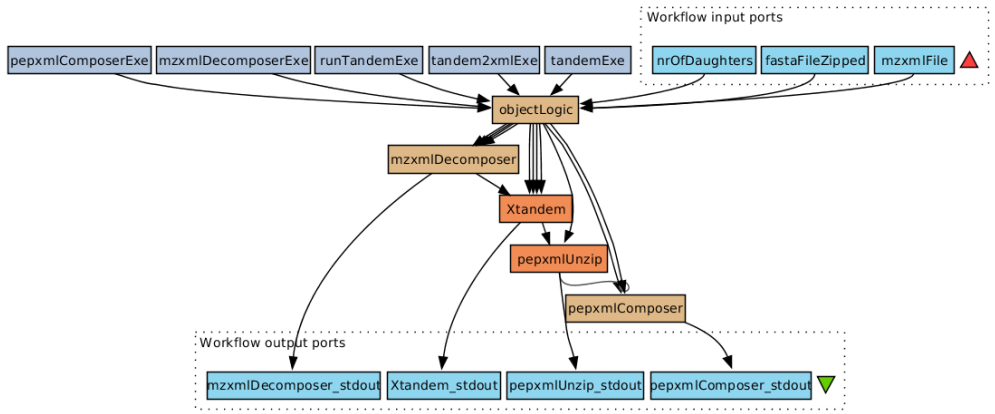


Figure 4.1. A scientific workflow for searching LC-MS/MS mass spectrometry data using X!Tandem on the cloud. The workflow consists of 5 processors. The *objectLogic* processor prepares all inputs in the right format, i.e. keeping or converting strings into file object according to the following processor. The *mzxmlDecomposer* and *pepxmlComposer* run the decomposing/recomposing algorithms. *objectLogic*, *mzxmlDecomposer* and *pepxmlComposer* are Beanshell processors and they run locally. *Xtandem* runs X!Tandem on a remote machine and *pepxmlUnzip* unzip the pepXML files to a local directory; both are Tool processors.

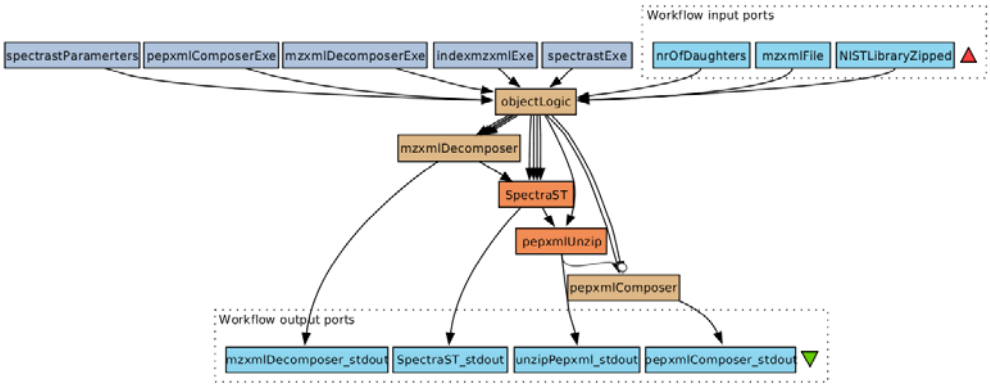


Figure 4.2. A scientific workflow for searching LC-MS/MS data using SpectraST on the cloud. The processor *mzxmlDecomposer*, *pepxmlUnzip* and *pepxmlComposer* are identical to the one in the X!Tandem workflow (Figure 5.1). The only difference is that the *Xtandem* processor is exchanged with the *Spectrast* processor and the constant inputs are adjusted to SpectraST. This approach is also possible for other search engines as described in the *Data decomposition and recomposition* paragraph

processors and run locally. The search engine is a tool processor and runs on the cloud. Taverna stores the IP addresses and passwords of the cloud worker nodes in its credential management. The password repository is protected with a master password, i.e. the user need to authenticate only once when starting Taverna.

Figure 4.1 shows a workflow to run X!Tandem on the cloud. The workflow takes the mzXML file(s), zipped search data base file in FASTA format and the number of the daughter mzXML files as inputs. Ideally the number of the daughter files is an integer factor of the available cloud workers. The search engine parameters are included in the *runTandemExe* processor. Figure 4.2 shows a simple scientific workflow to run SpectraST on the cloud. Similarly, the workflow takes the mzXML file(s), the zipped search library files (including the .splib, .spidx and .pepidx files) and the number of daughter mzXML files as inputs. SpectraST search parameters are included in the *spectrastParameters* processor, which is a string and is adjustable for different experiments. The processing logic of both workflows is very similar. The decomposition, recomposition and unzip pepXML processors are identical. The search engine calling processors are adjusted to each search engine, but are still logically very similar. This processor can be readjusted for other search engines. It is sometimes beneficial to separate the preprocessing/decomposing of the mzXML files from the logic of

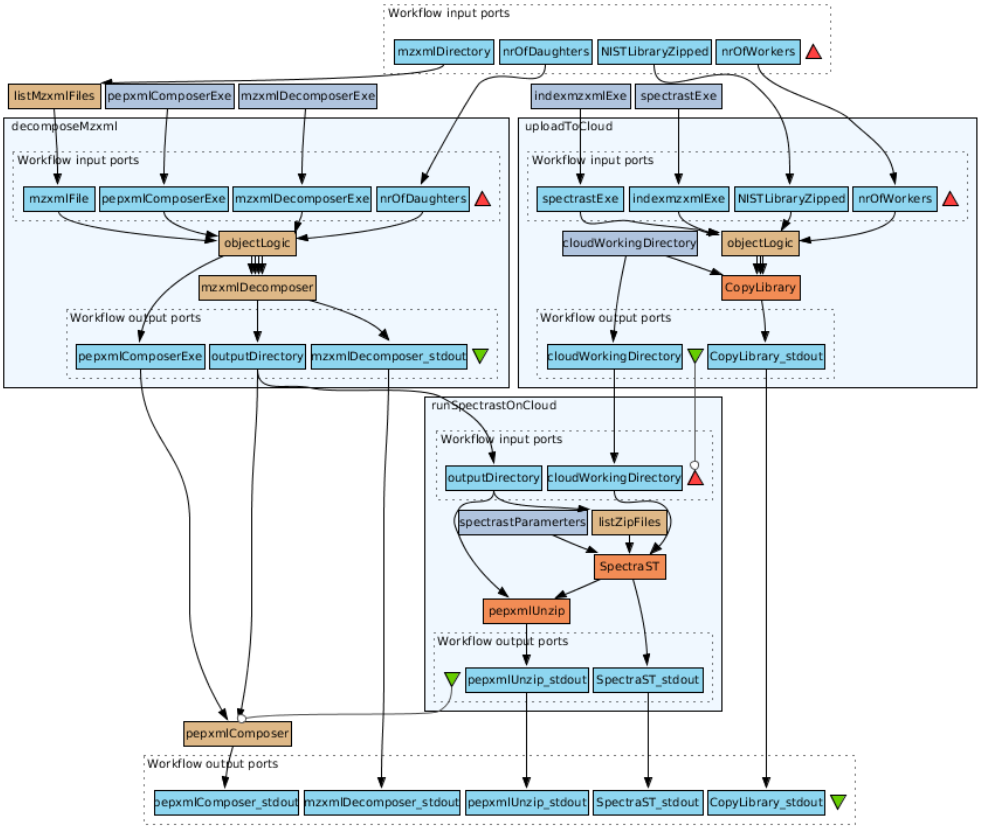


Figure 4.3. An advanced scientific workflow for searching LC-MS data using SpectraST on the cloud. Uploading the libraries is optimized to achieve better performance, which makes this workflow more suitable for processing mzXML spectra files from human samples, as the corresponding NIST library needed by SpectraST is larger than 2 GB. Here we connect 3 nested workflows, in which the first 2, i.e. *decomposeMzxml* and *uploadToCloud* run in parallel while the third nested workflow, i.e. *runSpectrastOnCloud* will start only if *uploadToCloud* finished all iteration. *runSpectrastOnCloud* and *decomposeMzxml* can still run in parallel.

uploading big data like the NIST human spectral libraries[41, 42] for SpectraST. Figure 4.3 illustrates an advanced workflow for SpectraST, in which the needed library and executables for the processing are uploaded simultaneously while decomposing the input mzXML files. The three workflows are available from ms-utils.org/cloud.

Speed performance comparison

We compared the elapsed wall clock time needed to analyze one file of the human dataset, the whole human data set, and the whole *E. coli* dataset on a local workstation and on the cloud using our method. The results are summarized in Table 4.1 and Figure 4.6. In the simplest scenario of analyzing one mzXML file we achieved speedup of 11-fold in case of X!Tandem running on single core and of 6-fold in case of SpectraST.

In order to exploit our implementation and test it for possible future big data challenges, we used spectra from fractioned sample to construct a single large mzXML file of 213,788 spectra. We profiled the number of cores in relation to the elapsed wall clock time needed to process these spectra and the results are illustrated in Figure 4.4. We were able to perform the peptide identification using X!Tandem and 8 cloud machines each of 8 processors within 12 min, a 26-fold faster than running it on a single core machine. When allowing phosphorylation as a variable modification, it was possible to obtain identical results within 72 minutes using 64 CPUs, or 5 hours on a single 8 CPU cloud node (comparable to an 8 CPU local machine).

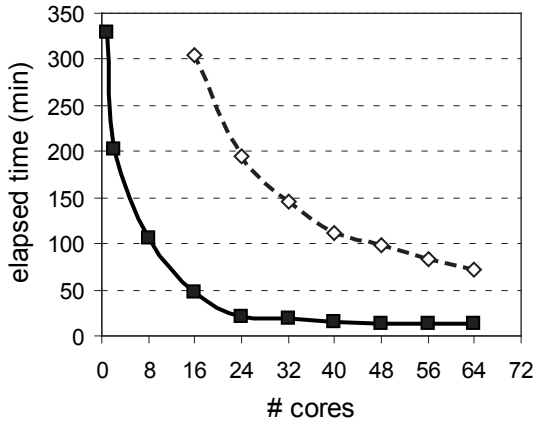


Figure 4.4. The wall time needed to search a large input file against a human sequence database[40] as a function of the number of cores with only fixed modifications (solid, squares) and with phosphorylation as variable modification (dashed, diamonds). By implementing data parallelism on the workflow level, it was possible to process 213,788 spectra in a 1.3 GB mzXML file with a search window of -0.5 to 2.5 Da in 12 minutes with only fixed modifications, a job which would take more than 5 hours on a desktop computer. When allowing phosphorylation (on serine, threonine and tyrosine) as a variable modification, the search took 72 minutes using 64 processors.

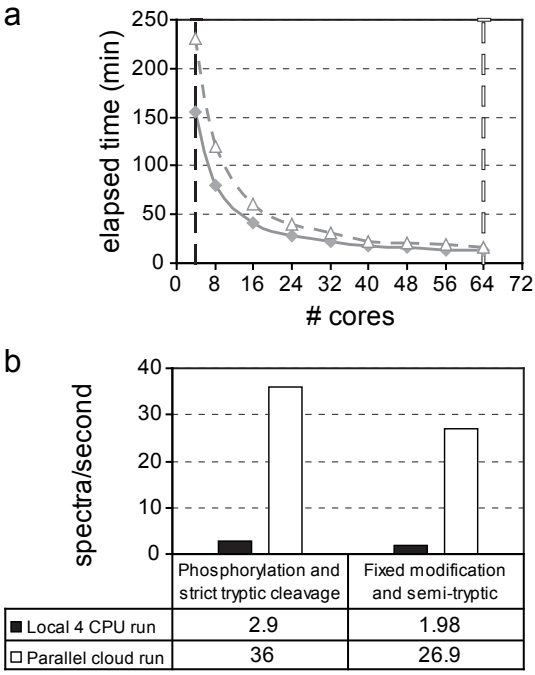
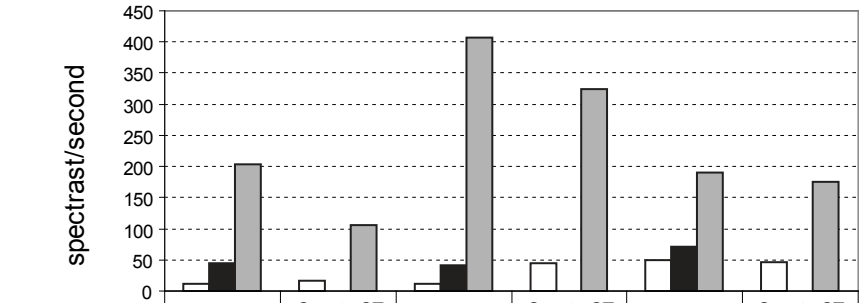


Figure 4.5. Comparison of different search times with either variable modification or semi-specific cleavage (a). The X!Tandem workflow was used to search a human dataset of 27,436 spectra against the human sequence database,[40] with strict tryptic cleavage and allowing for phosphorylation (dashed gray, triangles), and with semi-tryptic cleavage and only fixed modifications (solid gray, diamonds) as variable modification. Speed improvement of parallel processing in the 64-CPU cloud compared to 4 local CPUs of the same searches (b).

To evaluate the performance in common practice, where the enzyme fidelity is not known a priori or other proteases may have been active, we ran a series of tests with semi-specific cleavage on a smaller set of 27,000 spectra and measured a 27-fold increase in speed (see Figure 4.5). When allowing three variable PTMs, the 64 CPU cloud finished searching these spectra 36 times faster than a 4-core local machine.

CONCLUSIONS

In data mining, data decomposition is considered the "most useful form of transformation of datasets".[49, 50] With our approach of wrapping data parallelism via decomposition and recomposition around the search engine, we were able to parallelize more than one peptide identification software. We demonstrated this using a common database search engine - X!Tandem - and a spectral library search engine - SpectraST. We achieved the parallelism by an ad hoc approach using off-the-shelf open source software for scientific workflow, i.e. Taverna workbench, and OpenNebula for cloud computing. We believe that such adhocratic cloud implementations can scale with future needs to handle big data in mass spectrometry based



	One sample (human)		5 samples (human)		20 samples (*E. coli*)	
	X!Tandem	SpectraST	X!Tandem	SpectraST	X!Tandem	SpectraST
Local run 1 CPU	11	16	12	44	50	46
Local run 4 CPUs	44	n/a	42	n/a	71	n/a
Parallel cloud run	203	106	407	324	191	175

Figure 4.6. The performance of the cloud compared to local runs for the same search engines and data. Three experiments are compared, the details for which are listed in Table 5.1.

proteomics. A single large mzXML file of 1.3 GB containing 213,788 spectra was searched using our cloud parallel X!Tandem in 12 min. Compared to other parallel implementation of search engines like the Message Passing Interface (MPI) enabled parallel X!!Tandem, or the Hadoop MapReduce deployment on Amazon web services – MR-Tandem, our method does not require dedicated MPI hardware or rewriting of the search algorithm. We designed our method to be generally applicable to any software that searches spectra independently and demonstrated this with X!Tandem and SpectraST. The method can possibly be used in combination with the other parallel programs. The decomposition/recomposistion algorithms and slightly modified workflows can then be used to distribute an mzXML file to multiple machines with the Hadoop MapReduce X!Tandem deployment or multiple machines with the MPI-parallel X!!Tandem. In this case the workflow can be modified to use the already installed search engine. In comparing speed performance by running identical searches on local machines and in parallel, our method achieved more than twice the increase in speed reported by the MPI-parallel X!!Tandem. Compared to the 31-fold speedup on 200 processors reported by the Hadoop MR-Tandem implementation, our method achieved 36-fold speedup on 64 processors.

To make the implementation useful to the research community, we used common standards for input and output. Furthermore, cloud instances from providers like Amazon or in-campus clouds can be used as long as they are associated with public IP addresses. In such cases our implementation can be used without modification. Where the instances have private IP

addresses, one can still launch the workflow from one of these instances without modification. Researchers can also build their own cloud environment by accessing accounts on different Linux machines without the need to install additional software; only Java is required, which is available for nearly all platforms. The decomposition and recomposition algorithms can be used in other scenarios, with or without clouds. For instance, when using a computer cluster or a computer with a multi-core CPU, the researcher can still use the data decomposition and recomposition with single-threaded algorithms such as SpectraST to gain parallelism.

We are currently working with the developers of scientific workflow managers and cloud providers to address different issues including starting and shutting down the virtual machines on the cloud entirely from within the workflow, using certificates authentication, and enhancing the security on the cloud. We are convinced the 36-fold speedup reported here is still not exploiting the available resources to their full potential and also work to further improve the acceleration of these algorithms using cloud.

ACKNOWLEDGMENTS

This work was supported by the Dutch Organization for Scientific Research (De Nederlandse Organisatie voor Wetenschappelijk Onderzoek, NWO), grants NRG-2010.06, BG-043-11 and VI-917.11.398. Used cloud resources are part of the Dutch e-Science Grid – "BigGrid".

SUPPLEMENTARY MATERIAL

The used data decomposition and recomposition algorithms are written Java and are available from www.ms-utils.org/decomposition. The Taverna workflows are available from www.ms-utils.org/cloud and on www.myExperiment.org.

REFERENCES

1. Kim, S.; Mischerikow, N.; Bandeira, N.; Navarro, J. D.; Wich, L.; Mohammed, S.; Heck, A. J.; Pevzner, P. A., The generating function of CID, ETD, and CID/ETD pairs of tandem mass spectra: applications to database search. Mol Cell Proteomics 2010, 9, (12), 2840-52.
2. Swaney, D. L.; McAlister, G. C.; Coon, J. J., Decision tree-driven tandem mass spectrometry for shotgun proteomics. Nat Methods 2008, 5, (11), 959-64.
3. Resemann, A.; Wunderlich, D.; Rothbauer, U.; Warscheid, B.; Leonhardt, H.; Fuchser, J.; Kuhlmann, K.; Suckau, D., Top-down de Novo protein sequencing of a 13.6 kDa camelid single heavy chain antibody by matrix-assisted laser desorption ionization-time-of-flight/time-of-flight mass spectrometry. Anal Chem 2010, 82, (8), 3283-92.
4. Michalski, A.; Damoc, E.; Hauschild, J. P.; Lange, O.; Wieghaus, A.; Makarov, A.; Nagaraj, N.; Cox, J.; Mann, M.; Horning, S., Mass spectrometry-based proteomics using Q Exactive, a high-performance benchtop quadrupole Orbitrap mass spectrometer. Mol Cell Proteomics 2011, 10, (9), M111 011015.
5. Frese, C. K.; Altelaar, A. F.; Hennrich, M. L.; Nolting, D.; Zeller, M.; Griep-Raming, J.; Heck, A. J.; Mohammed, S., Improved peptide identification by targeted fragmentation using CID, HCD and ETD on an LTQ-Orbitrap Velos. J Proteome Res 2011, 10, (5), 2377-88.
6. Keller, A.; Eng, J.; Zhang, N.; Li, X. J.; Aebersold, R., A uniform proteomics MS/MS analysis platform utilizing open XML file formats. Mol Syst Biol 2005, 1, 2005 0017.
7. Perkins, D. N.; Pappin, D. J. C.; Creasy, D. M.; Cottrell, J. S., Probability-based protein identification bu searching sequence databases using mass spectromery data. Electrophoresis 1999, 20, (18), 3551-2567.
8. Yates, J. R., 3rd; Eng, J. K.; McCormack, A. L.; Schieltz, D., Method to correlate tandem mass spectra of modified peptides to amino acid sequences in the protein database. Anal Chem 1995, 67, (8), 1426-36.
9. Craig, R.; Beavis, R. C., TANDEM: matching proteins with tandem mass spectra. Bioinformatics 2004, 20, (9), 1466-7.
10. Craig, R.; Cortens, J. C.; Fenyo, D.; Beavis, R. C., Using annotated peptide mass spectrum libraries for protein identification. J Proteome Res 2006, 5, (8), 1843-9.
11. Duncan, D. T.; Craig, R.; Link, A. J., Parallel tandem: A program for parallel processing of tandem mass spectra using PVM or MPI and X!Tandem. Journal of Proteome Research 2005, 4, (5), 1842-1847.
12. Bjornson, R. D.; Carriero, N. J.; Colangelo, C.; Shifman, M.; Cheung, K. H.; Miller, P. L.; Williams, K., X!!Tandem, an improved method for running X!tandem in parallel on collections of commodity computers. J Proteome Res 2008, 7, (1), 293-9.
13. Pratt, B.; Howbert, J. J.; Tasman, N. I.; Nilsson, E. J., MR-Tandem: parallel X!Tandem using Hadoop MapReduce on Amazon Web Services. Bioinformatics 2012, 28, (1), 136-7.

14. Baumgardner, L. A.; Shanmugam, A. K.; Lam, H.; Eng, J. K.; Martin, D. B., Fast parallel tandem mass spectral library searching using GPU hardware acceleration. J Proteome Res 2011, 10, (6), 2882-8.

15. Pratt, B., GPU-ACCELERATED PEPTIDE SEARCH. In Funded by Department of Health and Human Services, 1R43HG006414-01: 2011.

16. Park, C. Y.; Klammer, A. A.; Kall, L.; MacCoss, M. J.; Noble, W. S., Rapid and accurate peptide identification from tandem mass spectra. J Proteome Res 2008, 7, (7), 3022-7.

17. Diament, B. J.; Noble, W. S., Faster SEQUEST searching for peptide identification from tandem mass spectra. J Proteome Res 2011, 10, (9), 3871-9.

18. Pedrioli, P. G.; Eng, J. K.; Hubley, R.; Vogelzang, M.; Deutsch, E. W.; Raught, B.; Pratt, B.; Nilsson, E.; Angeletti, R. H.; Apweiler, R.; Cheung, K.; Costello, C. E.; Hermjakob, H.; Huang, S.; Julian, R. K.; Kapp, E.; McComb, M. E.; Oliver, S. G.; Omenn, G.; Paton, N. W.; Simpson, R.; Smith, R.; Taylor, C. F.; Zhu, W.; Aebersold, R., A common open representation of mass spectrometry data and its application to proteomics research. Nat Biotechnol 2004, 22, (11), 1459-66.

19. Seattle Proteome Center/Institute for Systems Biology mzXML Format. http://tools.proteomecenter.org/wiki/index.php?title=Formats:mzXML (June 13),

20. Seattle Proteome Center/Institute for Systems Biology pepXML Format. http://tools.proteomecenter.org/wiki/index.php?title=Formats:pepXML (June 13),

21. List of free software for analysis of mass spectrometry data. www.ms-utils.org (June 13),

22. Martens, L.; Chambers, M.; Sturm, M.; Kessner, D.; Levander, F.; Shofstahl, J.; Tang, W. H.; Rompp, A.; Neumann, S.; Pizarro, A. D.; Montecchi-Palazzi, L.; Tasman, N.; Coleman, M.; Reisinger, F.; Souda, P.; Hermjakob, H.; Binz, P. A.; Deutsch, E. W., mzML--a community standard for mass spectrometry data. Mol Cell Proteomics 2011, 10, (1), R110 000133.

23. Mohammed, Y.; Shahand, S.; Korkhov, V.; Luyf, A. C. M.; Schaik, B. D. C. v.; Caan, M. W. A.; Kampen, A. H. C. v.; Palmblad, M.; Olabarriaga, S. D., Data Decomposition in Biomedical e-Science Applications. In IEEE 7th International Conference on E-Science, e-Science 2011, Workshop Proceedings, Stockholm, Sweden, 2011.

24. Keller, A.; Nesvizhskii, A. I.; Kolker, E.; Aebersold, R., Empirical statistical model to estimate the accuracy of peptide identifications made by MS/MS and database search. Anal Chem 2002, 74, (20), 5383-92.

25. Kall, L.; Canterbury, J. D.; Weston, J.; Noble, W. S.; MacCoss, M. J., Semi-supervised learning for peptide identification from shotgun proteomics datasets. Nat Methods 2007, 4, (11), 923-5.

26. Craig, R.; Beavis, R. C., A method for reducing the time required to match protein sequences with tandem mass spectra. Rapid Commun Mass Spectrom 2003, 17, (20), 2310-6.

27. Geer, L. Y.; Markey, S. P.; Kowalak, J. A.; Wagner, L.; Xu, M.; Maynard, D. M.; Yang, X.; Shi, W.; Bryant, S. H., Open mass spectrometry search

algorithm. J Proteome Res 2004, 3, (5), 958-64.

28. Specht, M.; Kuhlgert, S.; Fufezan, C.; Hippler, M., Proteomics to go: Proteomatic enables the user-friendly creation of versatile MS/MS data evaluation workflows. Bioinformatics 2011, 27, (8), 1183-4.

29. Kessner, D.; Chambers, M.; Burke, R.; Agus, D.; Mallick, P., ProteoWizard: open source software for rapid proteomics tools development. Bioinformatics 2008, 24, (21), 2534-6.

30. Goecks, J.; Nekrutenko, A.; Taylor, J., Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. Genome Biol 2010, 11, (8), R86.

31. Maheshwari, K.; Montagnat, J. In Scientific Workflow Development Using Both Visual and Script-Based Representation, Services (SERVICES-1), 2010 6th World Congress on, 5-10 July 2010, 2010; 2010; pp 328-335.

32. Altintas, I.; Berkley, C.; Jaeger, E.; Jones, M.; Ludascher, B.; Mock, S., Kepler: An Extensible System for Design and Execution of Scientific Workflows. In Proceedings of the 16th International Conference on Scientific and Statistical Database Management, IEEE Computer Society: 2004; p 423.

33. Oinn, T.; Addis, M.; Ferris, J.; Marvin, D.; Senger, M.; Greenwood, M.; Carver, T.; Glover, K.; Pocock, M. R.; Wipat, A.; Li, P., Taverna: a tool for the composition and enactment of bioinformatics workflows. Bioinformatics 2004, 20, (17), 3045-3054.

34. de Bruin, J. S.; Deelder, A. M.; Palmblad, M., Scientific Workflow Management in Proteomics. Mol Cell Proteomics 2012.

35. Waterman, R. H., Jr., 'Adhocracy': lessons from the changemasters. Hospitals 1991, 65, (1), 56.

36. Waterman, R. H., Jr., Adhocracy W. W. Norton & Company: 1993; p 128.

37. Taverna Website. www.taverna.org.uk/ (June 13),

38. OpenNebula Website. www.opennebula.org/ (June 13),

39. Uniprot canonical sequence in FASTA format, obtained from www.uniprot.org on June 18, 2012 with the search string: “organism:Escherichia AND coli AND keyword:181 AND keyword:1185 AND reviewed:yes”.

40. Uniprot canonical sequence in FASTA format, obtained from www.uniprot.org on June 4, 2012 with the search string: “organism:"Homo sapiens" AND keyword:181”.

41. Eds. S.E. Stein and P.A. Rudnick, NIST Peptide Tandem Mass Spectral Libraries. E. coli Peptide Mass Spectral Reference Data, E. coli, ion trap, Official Build Date: April 20, 2012. National Institute of Standards and Technology, Gaithersburg, MD, 20899. Downloaded from http://peptide.nist.gov on June 18, 2012. In.

42. Eds. S.E. Stein and P.A. Rudnick, NIST Peptide Tandem Mass Spectral Libraries. Human Peptide Mass Spectral Reference Data, H. sapiens, ion trap, Official Build Date: May 26, 2011. National Institute of Standards and Technology, Gaithersburg, MD, 20899. Downloaded from http://peptide.nist.gov on June 6, 2012. In.

43. Apache Hadoop. http://hadoop.apache.org/ (June 13),

44. White, T., Hadoop: The Definitive Guide. O'Reilly Media, Inc.: Sebastopol, CA 95472., 2009.

45. Dean, J.; Ghemawat, S., MapReduce: simplified data processing on large clusters. Commun. ACM 2008, 51, (1), 107-113.

46. Taylor, R. C., An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics. BMC Bioinformatics 2010, 11.

47. SARA cloud. www.cloud.sara.nl (May 30),

48. Open Grid Forum, Open Cloud Computing Interface Specification. In 2009.

49. Kusiak, A., Decomposition in data mining: An industrial case study. Ieee Transactions on Electronics Packaging Manufacturing 2000, 23, (4), 345-354.

50. Maimon, O.; Rokach, L., Decomposition Methodology for Knowledge Discovery and Data Mining. In DATA MINING AND KNOWLEDGE DISCOVERY HANDBOOK, Maimon, O.; Rokach, L., Eds. Springer: New York, 2005; pp 981-1003.