REVIEW

Distributed computing and data storage in proteomics: Many hands make light work, and a stronger memory

Kenneth Verheggen[1,2], *Harald Barsnes*[3] *and Lennart Martens*[1,2]

[1] Department of Medical Protein Research, VIB, Ghent, Belgium
[2] Department of Biochemistry, Faculty of Medicine and Health Sciences, Ghent University, Ghent, Belgium
[3] Proteomics Unit, Department of Biomedicine, University of Bergen, Norway

Modern day proteomics generates ever more complex data, causing the requirements on the storage and processing of such data to outgrow the capacity of most desktop computers. To cope with the increased computational demands, distributed architectures have gained substantial popularity in the recent years. In this review, we provide an overview of the current techniques for distributed computing, along with examples of how the techniques are currently being employed in the field of proteomics. We thus underline the benefits of distributed computing in proteomics, while also pointing out the potential issues and pitfalls involved.

Received: July 12, 2013
Revised: September 9, 2013
Accepted: September 23, 2013

Keywords:
Bioinformatics / Cloud computing / Crowdsourcing / Distributed computing / Parallelized computing

1 Background

As most scientists who come into contact with current high-throughput proteomics know, it is not always straightforward to process and store such data. Furthermore, because of continuous advances in MS and proteomics methods, the size of the generated data will continue to increase in the foreseeable future [1–3]. As a result, the field of proteomics has encountered computational requirements that far exceed the specifications of most common desktop computers [4]. Indeed, for any computational process, the overall speed is determined by the size and amount of the input data, the amount of compute cores that can simultaneously perform calculations, and the availability of the necessary storage capacity. Additionally, the data storage has to be as efficient and accessible as possible, while the processing should be stable and powerful in order to avoid bottlenecks in the data analysis.

Fortunately, proteomics does not have to start from scratch as there are precedents in fields outside of proteomics, such as the Galaxy genomic workbench [5] and TranSmart (http://www.transmartproject.org). Both these frameworks have been developed to store, distribute, and analyze genomics data. The principles used in these examples should be amenable for reuse in solving the issues for distributed proteomics. Inspired by genomics, distributed proteomics is now starting to take off, yet a lot of innovation and new adaptations will likely emerge to deliver solid, field-changing applications. It is important to note that computers were originally intended to sequentially execute tasks. This was essentially a technical limitation, due to the fact that a computer possessed a single central processing unit (CPU). However, in the recent decade single-CPU designs have been replaced by parallel systems with multiple compute cores, able to run multiple processes simultaneously. As a result, parallel processing has become the standard way of speeding up the transformation and storage of data, and is the main innovation that has allowed computers to keep up with Moore's law [6]. This is important, because even though the total transistor count on a chip has kept increasing, as Moore famously predicted, the speed of the individual processing units has been more or less stagnant for well over 5 years.

Software programs that only make use of a single compute core are therefore no longer speeding up with the purchase of a new computer, despite the fact that the new computer

Correspondence: Professor Lennart Martens, Department of Medical Protein Research, Ghent University, Albert Baertsoenkaai 3, B-9000 Ghent, Belgium
E-mail: lennart.martens@UGent.be

Abbreviations: AMI, Amazon Machine Image; **AWS**, Amazon Web Service; **CPU**, central processing unit; **GPGPU**, general purpose graphics processing unit; **GPU**, graphics processing unit

Colour Online: See the article online to view Figs. 1 and 2 in colour.

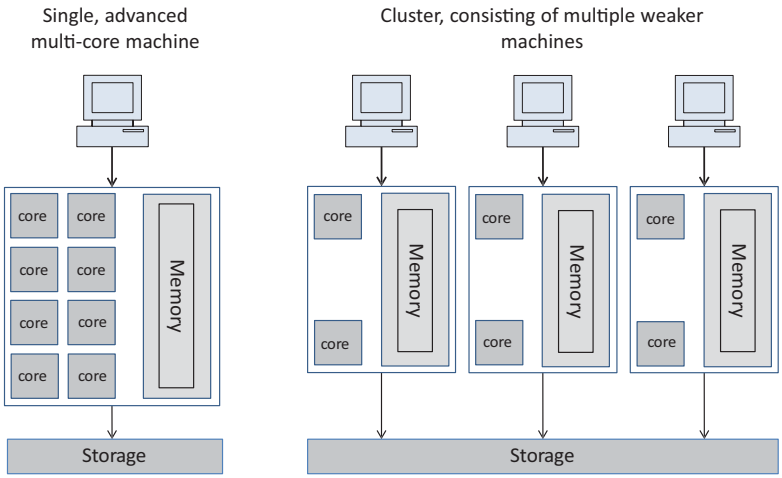

Figure 1. The biggest difference between a multicore computer and a multinode architecture is the memory usage. In the first case, shared memory is used between local CPUs, while in the latter, every node has CPUs linked by local memory, yet the nodes themselves do not share memory.

contains double or more transistors. The only way to capitalize on the presence of these additional transistors is to split the work over multiple processing units, thus executing the overall task as parallel subtasks. Consequentially, the biggest benefit of distributed systems is their scalability. And while it is not always an option to add additional CPUs to a system, adding additional machines to a coherent working group is in most cases straightforward. This ensures that in multimachine distributed computing, the size of the computer pool can be scaled to match the requirements of the given problem. It is equally important to note that the actual compute time is not reduced in such a parallel computation approach, as the total running time summed across the different processing units is equal to, or even slightly higher than the running time on a single processor. Of course, the waiting time for the user is shortened by parallelization, and one therefore distinguishes between: wall time (as measured by the clock on the wall) for the user-perceived time to finish the task, and compute time for the total amount of time spent in computation by all enlisted processing units.

The field of high-performance computing has also moved toward parallelization, by increasingly using multiple, relatively weak computers in parallel instead of a single supercomputer. Note that this does not imply that a collection of individual units is intrinsically more parallel than the internal workings of a single supercomputer, but rather that there is a tendency to stop increasing the processing power of a single device, and instead expanding into greater numbers of more standardized components. Indeed, where very large and expensive supercomputers were originally the main method of providing high-end computation in science [7], the combination of numerous (relatively) inexpensive computers into so-called clusters via networking has since proven to be a valuable and highly popular substitute [8]. The culmination of this evolution has been an optimization of both strategies in hybrid systems that build clusters of (mid-range) supercomputers [9].

Perhaps the most striking result of the evolution toward cluster-based systems in high-performance computing is that the availability of such systems is no longer confined to a short list of institutions, but is now well within the financial means of most individual research groups. It is however important to note that running software on a cluster is not equivalent to running the same software on a multicore machine. For example, the memory is not shared in the case of a cluster, with each machine only having direct access to its own memory, while the multicore approach allows all cores to access the same memory directly. This important difference is illustrated in Fig. 1. Both versions of memory architecture have their benefits. When memory is shared, there is little latency when programs communicate, but multiple instances may attempt to access the same data simultaneously, potentially creating so-called concurrency issues. Using distributed memory on the other hand leads to the exclusion of such issues at the cost of higher latency due to the necessary network communication overhead.

Perhaps the best known architecture for parallel computing using off-the-shelf computers is GRID computing. The term itself was first used in the early 90s, as an analogy to the power grid [10], but where a traditional power grid allows a user to plug in and acquire electrical power at will, a computational GRID is designed to deliver CPU time and processing capacity upon request. In principle, cluster- and GRID-based systems are very similar, with the former essentially a simplified version of the latter [11].

The culmination of the GRID architecture is known as cloud computing. The term cloud first appeared in 2006, with the inception of Amazon's Web Service (AWS) (http://aws.typepad.com/aws/2006/08/amazon_ec2_beta.html). It refers to the unknown nature of the trajectory that data go through to get to its point of destination. In essence, the cloud is a return to an early concept in computing, when large and powerful mainframes connected to much less powerful terminals were the preferred setup in both industry and research. In both early distributed computing and in cloud computing,

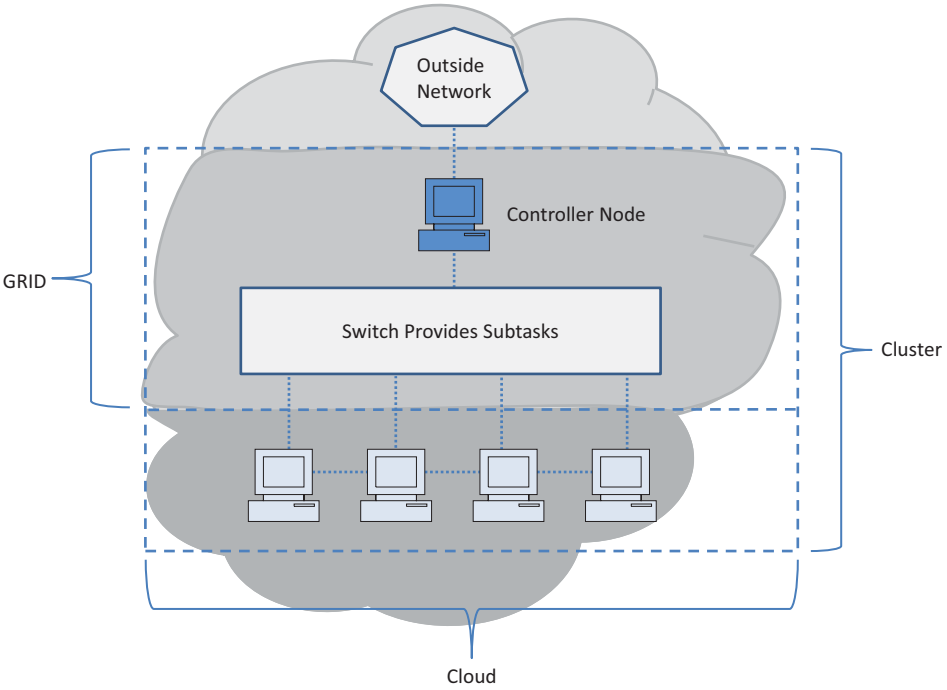

Figure 2. A computer cluster is built up out of several connected computers that cooperate as a single powerful unit. These computers are per definition in the same geographical location. The GRID methodology strongly resembles the cluster architecture, but makes use of geographically separated computers, allowing nodes to be dispersed across institutes. Cloud-based computing implies that resources are not accessed directly, but are queried via an intermediate service.

the main compute power was thus provided off-site, likely at a geographically distinct location. Rather than contacting a single mainframe however, a compute cloud consists of a large group of computers that can be located anywhere in the world, and that are connected in a GRID-like architecture via the internet, as shown in Fig. 2.

Yet computers are not the only thing connected through the internet; users too can be reached via this global network. Rather than simply calling upon (spare) compute cycles through the internet, it is now also possible to obtain services, ideas, or content from databases and users worldwide, prompting the term crowdsourcing. In a way, crowdsourcing is a logical next step in distributed scientific computing, as science itself has thrived on the sharing of ideas and results. And while the term might be new, the fundamental concept has been part of the life sciences, and of the studies of proteins in particular, for quite some time. Indeed, scientists have been sharing the results of their protein 3D structure determination experiments for decades [12–15] and have been building impressive databases of annotated genomes and proteomes through a combination of a few large-scale efforts and numerous small-scale contributions [16, 17]. The public availability of data is of course a key precondition for the success of crowdsourcing, and its importance has therefore been emphasized repeatedly, for instance in the field of proteomics [18, 19].

In order to provide insight into the distributed computing techniques and their use in proteomics, this review presents a brief overview of solutions that have been developed to address both processing and storage issues, along with tangible examples of successful applications of these techniques in the field. This review can thus act as a guidebook to determine whether distributed computing techniques can be useful for future proteomics informatics projects, and if so, help select the most suitable approach.

2 Divide and conquer

It is not always necessary to invest in new hardware to maximize the speed of an application. In many cases, simply multithreading the software can significantly augment efficiency on modern CPUs. When multithreaded, a program will not execute its operations in series, but in parallel. Needless to say, this excludes inevitably serial operations, such as the writing to magnetic hard drives. The basic concept of multithreading is explained in Fig. 3.

Examples of multithreaded applications for high-throughput proteomics include the parallelization of commonly used database search engines such as Mascot [20], X!Tandem [21], and OMSSA [22]; phosphoRS [23] performing phosphorylation site assignment in peptides based on the corresponding tandem mass spectra; and the proteome discovery pipeline by Gough et al. [24].

Even though multithreading provides a straightforward way of speeding up tasks that can be divided into smaller subtasks, multithreaded applications remain tied to the limited resources available in the host computer. Indeed, very large projects can easily result in impractically long wall times on modern desktop computers, despite their ability to handle from two to eight parallel processes. In order to overcome the limitations inherent to a single computer, the

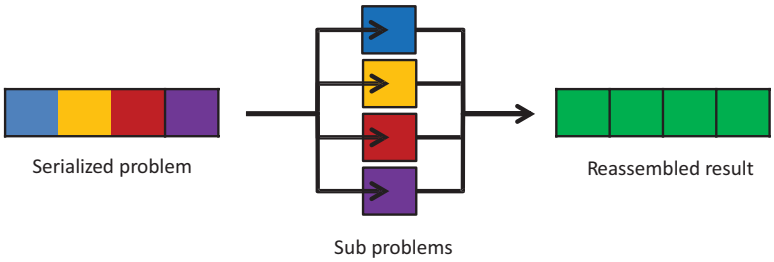

Figure 3. A multithreaded approach to solving a computational problem. Initially, a serialized sequence of operations is fed to a program. Then, the whole is broken down into smaller, subproblems that can be computed simultaneously. Finally, the results of each subproblem are merged to provide the final solution.

move to a truly distributed architecture must be made. Such a transition was made for X!Tandem, where the multithreaded capacities of the database search engine were expanded to allow parallelization across machines. First, multiple instances of X!Tandem were combined in the Parallel Virtual Machine (PVM) environment, and in the Message Passing Interface (MPI) system to distribute the work [25]. This type of load balancing has been thoroughly investigated and proven to reduce the overall searching time, without compromising the final output [26]. Elaborating on the Parallel Tandem project, X!!Tandem was developed to further optimize the operation of X!Tandem searches [27]. Here the parallelism was optimized by centralizing the remaining sequential steps. This illustrates that the level at which parallelization is implemented can have an effect on the efficiency of the overall process.

Classic database searching algorithms are not alone in putting a high combinatory load on processors. Another example is de novo sequencing of peptides, a classic problem in proteomics bearing high combinatory complexity [28]. These algorithms (reviewed in [29–31]) therefore typically come with large computational demands, but the de novo sequencing of each spectrum can be performed as independent subtasks. And even though multithreading of de novo algorithms presents a clear opportunity for increasing the performance, no such efforts have been published to date. A third, hybrid approach to identifying tandem mass spectra combines short stretches of de novo sequencing called sequence tags with subsequent (often error-tolerant) database searching. An excellent example is provided by DirecTag [32], where parallelization is used to improve the efficiency of the tag detection.

Thus far, we have only discussed the usage of CPUs as the main processing workhorse. Another option is to use the proficiency of graphics processing units (GPUs) to perform parallel calculations. Even though GPUs were designed to accelerate the computation of 3D graphics, thus relieving the CPU of this task, technical advances have made it possible to instruct a GPU to also perform general purpose computations. It is also no coincidence that most of the GPU-based applications work with NVIDIA drivers using Compute Unified Device Architecture (CUDA). CUDA is a platform developed to allow programmers to enhance the performance of their software by using the processing powers of the GPU. Such GPUs are called general purpose graphics processing units (GPGPUs) [33] because they use an architecture originally developed for graphics for more generic calculation purposes. The two main benefits of these GPGPUs are that they are essentially massively parallel linear algebra processors that can perform more exact mathematical operations than normal CPUs, allowing for substantial speedups and higher quality results [34].

Despite offering great potential and the promise of higher quality results, GPGPUs also come with a potentially serious drawback: writing algorithms that tap into the power offered by these processors, or porting existing algorithms to do so, can be quite challenging, requiring extensive expertise [35]. The usage of GPGPUs in proteomics data processing is therefore currently still in its infancy, with only limited examples available in the field. Nevertheless, these applications do demonstrate both the benefits and the difficulties of GPGPU processing, such as FastPaSS [35], an emulated form of a tandem mass spectral library search algorithm called SpectraST [36]. In a comparison between identical searches performed on a single CPU and a single GPGPU, FastPaSS outperformed SpectraST in terms of speed. However, the authors compared a single CPU against a single (but internally highly parallel) GPGPU that means that speed differences might be less dramatic on computers possessing multiple cores or CPUs.

The next logical step is to combine the expertise of both CPUs and GPUs. As discussed earlier, parallelizing a task over multiple cores is one of the pillars of distributed computing. Obviously, GPUs are also capable of this. It is thus logical to combine the strengths and weaknesses of both types of processing units, as shown in Fig. 4. One project that explores this benefit of synergetic CPU-GPGPU processing is Tempest [37]. Contrary to the focus placed on using solely a GPGPU, as is the case in FastPaSS, Tempest acknowledges the synergy between CPUs and GPUs, executing less straightforward calculations in the ranks of database digestion and MS/MS spectral indexing on a CPU, while reserving the GPGPU for the more cumbersome similarity scoring. This way, both types of processing units can be assigned those subtasks that fit their specialty. Although GPGPU driven proteomics data processing is currently in its infancy, the recent developments illustrated above show that this type of parallelization holds substantial promise.

3 Enter the GRID

A highly efficient, yet inexpensive way to achieve the parallel, distributed processing of datasets is to make optimal use of already existing resources. By employing idle computers in

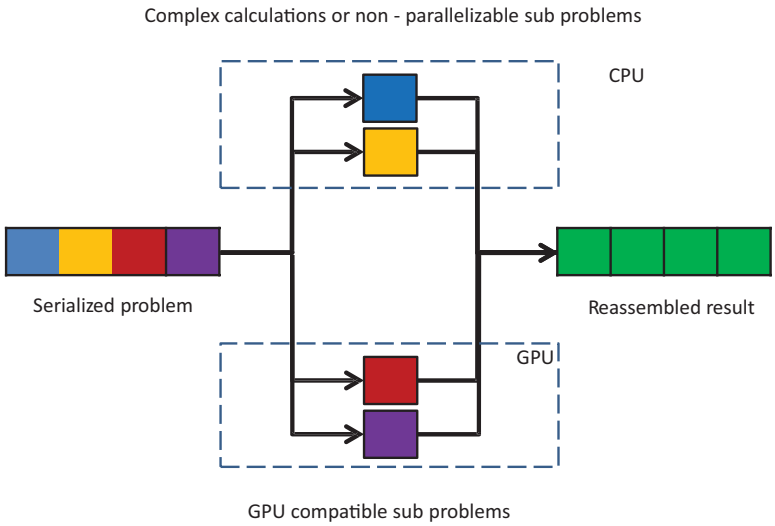

Figure 4. Multithreading is not confined to only generic CPUs. A parallelizable problem can be solved by both CPUs and GPUs, according to the complexity of the calculation. Often, there is even synergy between the efficiency in parallelized calculations of a GPU and the versatility of a CPU to handle complex calculations.

an existing network, a quite powerful cluster computer can be created ad hoc without extra resources. Since modern laboratories already have the required network infrastructure in place, which typically contains many computers that are often idle (even if only at night), this dormant computational power can easily be harvested using a GRID-computing approach. This strategy is used by the proTurbo framework, as part of the ProteomeGRID pipeline [38], developed to automate the analysis and mining of 2D gel images, a highly demanding computational challenge that has been the focus of considerable research for years.

As explained above, the GRID architecture is essentially an expanded cluster architecture. This allows a GRID system to run on traditional cluster setups while simultaneously allowing it to exploit the idle time on local, network-connected desktop computers. In the case of the proTurbo software, this property was used to great effect by running the system in a local facility, tapping into the idle CPU time of computers in a student lab. This increased the overall processing speed of the system dramatically while operation costs were reduced by the reuse of existing, idle equipment, thus nicely illustrating the usefulness and efficiency of GRID computing.

Another recent application of a GRID-based architecture in proteomics research is the Hydra search engine [39], a system built around a modified X!Tandem version designed to handle both very large amounts of spectra, as well as very large databases. Hydra is built on the Hadoop (http://hadoop.apache.org) distributed computing framework, famous for its scalability, as it allows for the efficient distribution of large datasets among many compute nodes connected in a GRID architecture. The development of tools such as Hydra will likely play a key role in the future of proteomics, as data volumes from typical experiments are set to continue to increase [40], and as meta-analysis becomes even more popular [41–43].

For those needing help in setting up GRID-computing systems, there is now a growing community aimed at providing support, with the ultimate goal of being able to process data easily on remote locations. The BOINC [44] system has proven its value in achieving this goal, as it allows users to donate compute cycles to one or more projects of choice without requiring technical know-how. As such, it provides a solid platform to develop GRID-based tools with access to a vast pool of users. One of the best known uses of the BOINC platform is the Rosetta@Home project [45, 46] providing a very successful illustration of the raw power of GRID computing by tackling the notoriously difficult problem of ab initio protein structure determination [47, 48]. With approximately 60,000 active computers donating their spare compute cycles, Rosetta@Home effectively functions, on an average basis, as a 57 teraflops (Floating Point Operations Per Second) supercomputer (http://boincstats.com/en/stats/14/project/detail).

Finally, it should be noted that moving from a single computer to a GRID setup also has an impact on the complexity of the required error handling. Failure recovery is thus an important aspect for large, ad hoc, distributed systems such as BOINC and Hadoop (http://hadoop.apache.org). This is due to the fact that failure during certain subprocesses cannot always be avoided, and large efforts have therefore gone into correct error handling and failure recovery in these systems. If such an advanced framework is not used however, the internal structure of the overall process should be meticulously planned and tested to account for subtask failure.

4 Take it to the cloud

The advent of the global internet has enabled computers all over the world to be connected to a common network, thus doing away with the requirement that computers need to be co-located in order to function as a coordinated whole. It is therefore a logical sequence of steps from the cluster architecture, via GRID computing to the current cloud computing systems. As cloud computing is a relatively young concept, the

definition has changed quite a bit during its evolution. This can cause confusion and eventually misuse of the term. According to the comprehensive NIST definition, cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [49].

Furthermore, the definition states that a cloud system should possess five key features. First of all, it should be a self-contained, readily available service, requiring little to no manual intervention. Second, the service should be independent of the user and their end devices, with an adequate network connection to sustain high performance for that particular service. Third, the required resources should be simultaneously available to multiple consumers, using technologies such as virtualization. Fourth, cloud systems must be capable of both providing and disposing resources rapidly, with no manual intervention required. Lastly, there is a more economical prerequisite to cloud computing. Each abstracted service should be measurable in terms of used resources. De facto, cloud systems are capable of automatically controlling and optimizing resource use by monitoring certain parameters appropriate to the service (e.g. storage, bandwidth, and active user accounts). Resource usage can then be managed easily, which leads to a billable, consumption-based system. Indeed, using cloud systems, the geographic locations of the various computers, whose processing and storage capabilities are combined into a coherent whole, are no longer important. Dedicated servers can be combined with idle computer time from desktop machines or even gaming consoles anywhere in the world, as long as they are connected to the internet. This approach to large-scale systems architecture has led to the provision of computing-as-a-service, where the necessary hardware is no longer owned or maintained by the user, but by the service provider, and where compute power and storage space are simply rented out. Of the many cloud computing providers, the popular Amazon Elastic Compute Cloud (EC2) has already proven its value in proteomics.

One example is the Virtual Proteomics Data Analysis Cluster (ViPDAC) [50] that provides a cost-efficient means to process large amounts of MS-based proteomics data. The system is built on a standard Amazon Machine Image (AMI), a medium performance virtual computer that can be rented from the Amazon cloud, which was modified by installing database search programs (OMSSA and X!Tandem) and protein sequence databases. Users can thus request as many such virtual computers as required, and use these to process the acquired mass spectra. It should be noted however that the task of distributing these spectra across different virtual computers is left to the user, and that substantial manual interaction remains to setup and coordinate the system, and to retrieve and parse the results after the searches are done.

Two more recent systems have been built that provide more convenient, pipelined access to cloud computing resources. The approach taken by Mohammed et al. [51] is to provide convenient access to the trans-proteomic pipeline (TPP) tools [52] in a GRID fashion by decomposing and recomposing the input data and output results automatically. As a result, a generic workflow engine like Taverna [53, 54] can be used to create ad hoc pipelines that can take full advantage of GRID or cloud architectures. Another solution, developed by Trudgian and Mirzaei [55] extends their existing, special purpose Central Proteomics Facilities Pipeline (CPFP) system to allow access to cloud computing resources.

A fourth cloud-based identification system, called ProteoCloud [56] takes a slightly different approach to user convenience, by providing a stand-alone system that covers the entire analysis workflow, including cloud-based database storage of the results, and full control of the system through a user-oriented graphical interface. ProteoCloud makes use of the previously mentioned Amazon EC2 cloud. In addition to the typical search engines provided by the four above-mentioned proteomics cloud-based approaches, ProteoCloud also includes more exhaustive methods for spectrum identification using both tag-based searches through the InSPECT tool [57] and de novo sequencing through PepNovo [58]. The output of the different algorithms is unified and merged based on the *q*-values determined using QVality [59].

A different example of cloud computing in proteomics is provided by Stanford University's Computing Cloud to process MS datasets and compare protein abundance in order to discover possible biomarkers [60]. The discovery of biomarkers using MS is a hot topic in proteomics and has been considered for a long time [61–63]. In this system, source data are uploaded through a web portal, followed by selection of peaks and statistical analysis on a cloud server with sufficient capacity to handle these computationally demanding operations. This approach allows researchers to quickly scan datasets for differentially expressed proteins, so that potential biomarkers can be picked up efficiently. However, it should be noted that the uploading of ever larger proteomics datasets can be a serious bottleneck for such setups.

5 Combining wetware with hardware

Despite the ability to simply rent compute time and storage space on globally distributed commodity hardware as the need arises, there remain challenges that cannot easily be solved by simply increasing computational power. In such cases, crowdsourcing often provides a more efficient means to solve the problem [64, 65]. Crowdsourcing relies on making an appeal to the collective online community to help solve a particular problem, either by asking for idle CPU time to create a GRID-based supercomputer, or by recruiting volunteers to carry out a particular task that computers do much more slowly than humans. One example depending on the use of idle CPUs is the already mentioned Rosetta@Home project [45, 46]. Another structure determination project, called Foldit (http://fold.it), goes beyond mere computational power

and requests its human volunteers to perform a more active task in solving protein structures. Foldit is essentially a multiplayer, online game that allows players to manipulate the structure of proteins, based on three simple biochemical rules understandable by nonscientists: (i) the smaller the resulting protein the higher the resulting structure scores; (ii) the hydrophobicity of residues should be respected, that is shielded from the surface; and (iii) two atoms cannot be in the same place at the same time. By thus turning a scientific problem into an online puzzle game, interested individuals from the general population can actively contribute to state-of-the-art research, thus generating an enormous working capacity while promoting and communicating science to nonscientists at the same time. The scientific results so far can be illustrated by the recently solved crystal structure of a retroviral protease called M-PMV [66] and the improved enzyme design for Diels alderase [67].

Projects such as Foldit effectively fuse two overall approaches for maximum benefit: the integration of human pattern-detection and problem-solving capabilities with modern advanced computational algorithms [68]. And thus, as the division of complex problems into subtasks formed the basis for parallel computing, problems are here split into new types of tasks: human-solvable and machine-solvable tasks. The system distributes these subtasks over a global population of active volunteers who donate both brain power as well as compute power. This type of parallelism, spanning both human brains (wetware) as well as computer processor cores (hardware) holds immense promise for future scientific research.

6 Is the sky the limit?

Although the above examples of distributed computing might lead one to think that there are no limitations in parallelizing complex problems, there are clearly defined hard limits to the benefits offered by distributed computing. Indeed, the efficiency of adding more computational power to a parallelized pool is not linearly correlated to the amount of cores employed. Additionally, not all processes can be effectively parallelized. Indeed, while computing the dot product of a single fragmentation spectrum against multiple proposed candidates from a spectral library can easily be parallelized, the subsequent consolidation of these scores into a ranked list is an inherently serial operation. Both aspects contribute to the maximal acceleration for a given algorithm in a parallel computing environment. This fact is described by Amdahl's law [69]. There is a strong resemblance between the theory that governs the reaction rate of a chemical process, where the slowest step determines the total speed of the entire reaction, and Amdahl's law. In the latter, it is assumed that any given program can be divided into subprocesses that can be either parallelizable or sequential. The degree in which a problem can be split into parallel subproblems without changing the outcome is called parallelizability. As shown in Fig. 5, the

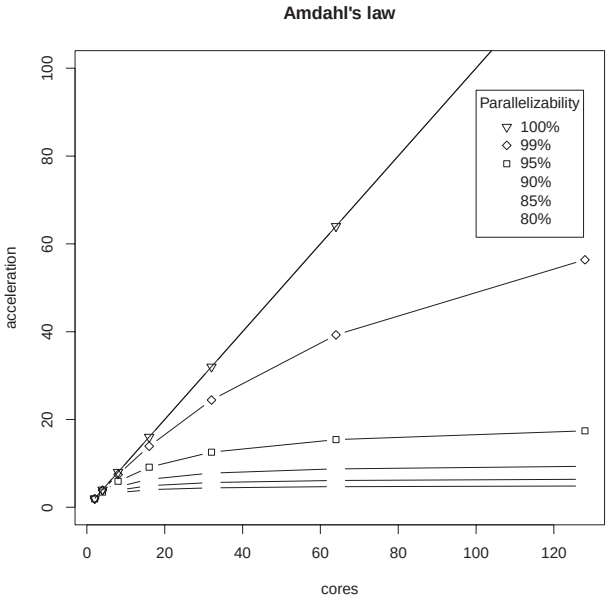

Figure 5. Amdahl's law [66] shows that the acceleration of processing is directly linked to the amount of available processors and the percentage of the computing process that is parallelizable. Any process that is not perfectly parallelizable will always encounter an upper limit where addition of further compute cores becomes essentially useless.

number of parallelizable tasks strongly determines the potential benefit of parallelization on the total process [70]. Yet even in nearly perfectly parallelizable problems, the acceleration of the process by adding more cores can still turn marginal quite fast.

Another limitation to parallel computing, which is not explicitly covered by Amdahl's law, involves the latencies inherent in data storage access and network communications. To reduce the impact of such limitations, it is possible to move away from a central point of storage by distributing the data across the system. The Hadoop framework mentioned above is perhaps the best known example of a strategy where the spread of the data is employed to speed up calculations across very large volumes of data. As a rule of thumb, the copying of data from a central location to the working unit's local storage becomes beneficial as soon as more than seven nodes are present in a cluster [71].

There is also an entirely orthogonal limitation that can affect distributed computing, and that relates to possible licensing issues. Many commercial software packages, such as the popular Mascot search engine, limit the number of computers the software can be installed on, and further place limitations on the number of CPUs and CPU cores that can be employed. Licenses to use additional compute power can be purchased in addition, but this often involves a substantial cost. It therefore comes as no surprise that the cloud-based proteomics platforms discussed earlier rely exclusively on freely available, open-source software packages. Furthermore, free or commercial closed-source packages that were

not originally designed to exploit parallel or distributed computing cannot easily be adapted, an issue that is largely absent in open-source software. The utility of being able to dive into the code and make changes is nicely shown throughout this review for the X!Tandem software, which several researchers have already adapted to better suit a particular parallel or distributed architecture.

7 Distributed databases

Apart from using the data, storing the data is another important aspect to consider when discussing distributed proteomics. Currently, there are two trends that "omics-scientists" follow: (i) specialist databases specific to a certain domain; or (ii) large databases encompassing a plethora of global data types or sources.

On the one hand, databases are often designed for a specific goal. Such is the idea behind SoyProDb [72] for instance, a database consisting of proteins derived from soy and intended to aid scientists in their efforts to genetically enhance soyplants, keeping track of the changed proteome. Another example is EPIC-DB [73], a publicly available database specifically built around the proteome and genome of Apicomplexan organisms. Another form of specialization is found in UniCarb-DB [74]. UniCarb-DB is a comprehensive LC MS/MS library for N- and O-linked glycans derived from glycoproteins. Furthermore, proteomics data repositories like PRIDE [3] and PeptideAtlas [75, 76] specialize in capturing and disseminating MS-based proteomics data.

Despite the usefulness of these diverse kinds of specialized resources, the most interesting biological questions typically require combined data retrieval, where the ability to cross-link information across different fields of research or organisms is required. Yet it is very complex to build integrated databases that can hold enough data to satisfy such complex queries, and it is even harder to keep them up-to-date. For this reason, the decentralized approach is typically chosen, where a shared integrative layer across a variety of repositories serves to tie all the data together. The quintessential example in the life sciences is BioMart [77–80], which allows users to access and combine data across many specialist databases, including HapMap [81], Wormbase [82], Gramene [83], dictybase [84] among others. Particularly important is the inclusion of hub databases such as Ensembl [85] and UniProt [86], which are already quite connected, and thus allow reliable links to be made between data sources, and even across domains in the life sciences. Finally, inclusion of repositories holding original research data such as PRIDE [3], allow the provenance of information to be verified down to the initial discovery. Novel technologies in the field of data storage, such as in-memory databases (e.g. https://www.proteomicsdb.org), will furthermore speed up access to increasingly large data resources, promising almost instantaneous results for even the most wide-ranging and complex queries.

8 A distributed future for proteomics

Distributed computing and parallelization of calculations have long been recognized to provide scientists with extremely powerful approaches to perform massive computations. This review therefore highlighted several recent approaches in which proteomics problems have been successfully parallelized. Since both the size and complexity of data analysis problems are likely to continue to grow in the field of proteomics, the popularity of clustered, GRID, or cloud-based data processing and analysis platforms is likely to increase further over the next few years.

Indeed, as laboratories become increasingly populated by computers for various tasks, and as increasingly sophisticated instruments continue to generate more data, the ready availability of spare compute power will almost certainly fuel the development of novel grid-based solutions for typical proteomics data analysis tasks. Furthermore, with cloud computing prices constantly dropping due to strong competition between providers, the renting of large, on-demand clusters for specific, demanding processing tasks will become more mainstream as well.

It will also be very interesting to see if the field of proteomics can successfully embrace the power of crowdsourcing. As shown by the Foldit project, a well-designed crowdsourcing project need not rely only on contributors with expertise in the applied field, vastly increasing the target audience of potential volunteers. Indeed, Foldit is now firmly rooted among the most popular scientific crowdsourcing project, rivaling perennial crowdsourcing giants like SETI@Home (http://setiathome.berkeley.edu) solidly planting proteins into the collective consciousness as a worthwhile target for study.

Finally, it has also become clear that the availability of open-source projects is of high importance. This is beautifully illustrated by the amount of available examples using the X!Tandem search engine across the entire palette of parallelized computing. Surely, easy to access open-source software has contributed to the novel, but clearly booming field of distributed computational proteomics. It will now be up to the field at large to devise strategies that can capitalize on this momentum, and harness the most of distributed computing for the purposes of MS-based proteomics!

K.V. is thankful to Ghent University and VIB. H.B. is supported by the Research Council of Norway. L.M. acknowledges the support of Ghent University (Multidisciplinary Research Partnership "Bioinformatics: from nucleotides to networks"), the IWT SBO grant 'INSPECTOR' (120025), the PRIME-XS project, grant agreement number 262067, and the "ProteomeXchange" project, grant agreement number 260558, both funded by the European Union 7th Framework Program.

The authors have declared no conflict of interest.

9 References

[1] Han, X., Aslanian, A., Yates, J. R., III, Mass spectrometry for proteomics. *Curr. Opin. Chem. Biol.* 2008, *12*, 483–490.

[2] Csordas, A., Ovelleiro, D., Wang, R., Foster, J. M. et al., PRIDE: quality control in a proteomics data repository. *Database* 2012, *2012*, doi: 10.1093/database/bas004.

[3] Vizcaino, J. A., Cote, R. G., Csordas, A., Dianes, J. A. et al., The Proteomics Identifications (PRIDE) database and associated tools: status in 2013. *Nucleic Acids Res.* 2013, *41*, D1063–D1069.

[4] Martín, H., J. A., Solving hard computational problems efficiently: asymptotic parametric complexity 3-coloring algorithm. *PLoS One* 2013, *8*, e53437.

[5] Lazarus, R., Taylor, J., Qiu, W., Nekrutenko, A., Toward the commoditization of translational genomic research: design and implementation features of the Galaxy genomic workbench. *Summit on Translat. Bioinforma.* 2008, *2008*, 56–60.

[6] Moore, G. E., Cramming more components onto integrated circuits, Reprinted from Electronics, volume 38, number 8, April 19, 1965, pp. 114 ff. *IEEE Solid-State Circuits Soc. Newsl.* 2006, *11*, 33–35.

[7] Missbach, M., Stelzel, J., Gardiner, C., Anderson, G., Tempes, M., *SAP Cloud*, Springer, Berlin, Heidelberg 2013, pp. 1–13.

[8] Oyanagi, Y., Future of supercomputing. *J. Comput. Appl. Math.* 2002, *149*, 147–153.

[9] Sadashiv, N., Kumar, S. M. D., *2011 6th International Conference on Computer Science and Education (ICCSE)*, SuperStar Virgo, Singapore 2011, pp. 477–482.

[10] Boukerche, A., Al-Shaikh, R. A., Notare, M. S. M. A., Towards highly available and scalable high performance clusters. *J. Comput. Syst. Sci.* 2007, *73*, 1240–1251.

[11] Foster, I., Kesselman, C., *The Grid 2: Blueprint for a New Computing Infrastructure*, Elsevier, Morgan Kaufmann, San Francisco, California, 2003.

[12] Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G. et al., The Protein Data Bank. *Nucleic Acids Res.* 2000, *28*, 235–242.

[13] Berman, H., Henrick, K., Nakamura, H., Announcing the worldwide Protein Data Bank. *Nat. Struct. Mol. Biol.* 2003, *10*, 980.

[14] Berman, H. M., Coimbatore Narayanan, B., Costanzo, L. D., Dutta, S. et al., Trendspotting in the Protein Data Bank. *FEBS Lett.* 2013, *587*, 1036–1045.

[15] Meyer, E. F., The first years of the Protein Data Bank. *Protein Sci.* 1997, *6*, 1591–1597.

[16] Hodis, E., Prilusky, J., Martz, E., Silman, I. et al., Proteopedia—a scientific "wiki" bridging the rift between three-dimensional structure and function of biomacromolecules. *Genome Biol.* 2008, *9*, R121.

[17] Tusnády, G. E., Dosztányi, Z., Simon, I., PDB_TM: selection and membrane localization of transmembrane proteins in the Protein Data Bank. *Nucleic Acids Res.* 2005, *33*, D275–D278.

[18] Prince, J. T., Carlson, M. W., Wang, R., Lu, P., Marcotte, E. M., The need for a public proteomics repository. *Nat. Biotechnol.* 2004, *22*, 471–472.

[19] Martens, L., Nesvizhskii, A. I., Hermjakob, H., Adamski, M. et al., Do we want our data raw? Including binary mass spectrometry data in public proteomics data repositories. *Proteomics* 2005, *5*, 3501–3505.

[20] Perkins, D. N., Pappin, D. J., Creasy, D. M., Cottrell, J. S., Probability-based protein identification by searching sequence databases using mass spectrometry data. *Electrophoresis* 1999, *20*, 3551–3567.

[21] Fenyö, D., Beavis, R. C., A method for assessing the statistical significance of mass spectrometry-based protein identifications using general scoring schemes. *Anal. Chem.* 2003, *75*, 768–774.

[22] Geer, L. Y., Markey, S. P., Kowalak, J. A., Wagner, L. et al., Open mass spectrometry search algorithm. *J. Proteome Res.* 2004, *3*, 958–964.

[23] Taus, T., Köcher, T., Pichler, P., Paschke, C. et al., Universal and confident phosphorylation site localization using phosphoRS. *J. Proteome Res.* 2011, *10*, 5354–5362.

[24] Gough, E., Oh, C., He, J., Riley, C. P. et al., Proteome discovery pipeline for mass spectrometry-based proteomics. *BMC Bioinformatics* 2008, *9*, P21.

[25] 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. *J. Proteome Res.* 2005, *4*, 1842–1847.

[26] Deciu, C., Sun, J., Wall, M. A., On the predictability of protein database search complexity and its relevance to optimization of distributed searches. *J. Proteome Res.* 2007, *6*, 3443–3448.

[27] Bjornson, R. D., Carriero, N. J., Colangelo, C., Shifman, M. et al., X!!Tandem, an improved method for running X!Tandem in parallel on collections of commodity computers. *J. Proteome Res.* 2008, *7*, 293–299.

[28] Hughes, C., Ma, B., Lajoie, G. A., De novo sequencing methods in proteomics. *Methods Mol. Biol.* 2010, *604*, 105–121.

[29] Allmer, J., Algorithms for the de novo sequencing of peptides from tandem mass spectra. *Expert Rev. Proteomics* 2011, *8*, 645–657.

[30] Vaudel, M., Sickmann, A., Martens, L., Current methods for global proteome identification. *Expert Rev. Proteomics* 2012, *9*, 519–532.

[31] Pevtsov, S., Fedulova, I., Mirzaei, H., Buck, C., Zhang, X., Performance evaluation of existing de novo sequencing algorithms. *J. Proteome Res.* 2006, *5*, 3018–3028.

[32] Tabb, D. L., Ma, Z.-Q., Martin, D. B., Ham, A.-J. L., Chambers, M. C., DirecTag: accurate sequence tags from peptide MS/MS through statistical scoring. *J. Proteome Res.* 2008, *7*, 3838–3846.

[33] Dematté, L., Prandi, D., GPU computing for systems biology. *Brief. Bioinform.* 2010, *11*, 323–333.

[34] Hussong, R., Gregorius, B., Tholey, A., Hildebrandt, A., Highly accelerated feature detection in proteomics data

sets using modern graphics processing units. *Bioinformatics* 2009, *25*, 1937–1943.

[35] 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*, 2882–2888.

[36] Lam, H., Deutsch, E. W., Eddes, J. S., Eng, J. K. et al., Development and validation of a spectral library searching method for peptide identification from MS/MS. *Proteomics* 2007, *7*, 655–667.

[37] Milloy, J. A., Faherty, B. K., Gerber, S. A., Tempest: GPU-CPU computing for high-throughput database spectral matching. *J. Proteome Res.* 2012, *11*, 3581–3591.

[38] Dowsey, A. W., Dunn, M. J., Yang, G.-Z., ProteomeGRID: towards a high-throughput proteomics pipeline through opportunistic cluster image computing for two-dimensional gel electrophoresis. *Proteomics* 2004, *4*, 3800–3812.

[39] Lewis, S., Csordas, A., Killcoyne, S., Hermjakob, H. et al., Hydra: a scalable proteomic search engine which utilizes the Hadoop distributed computing framework. *BMC Bioinformatics* 2012, *13*, 324.

[40] Malik, R., Dulla, K., Nigg, E. A., Körner, R., From proteome lists to biological impact—tools and strategies for the analysis of large MS data sets. *Proteomics* 2010, *10*, 1270–1283.

[41] Gonnelli, G., Hulstaert, N., Degroeve, S., Martens, L., Towards a human proteomics atlas. *Anal. Bioanal. Chem.* 2012, *404*, 1069–1077.

[42] Matic, I., Ahel, I., Hay, R. T., Reanalysis of phosphoproteomics data uncovers ADP-ribosylation sites. *Nat. Methods* 2012, *9*, 771–772.

[43] Hahne, H., Moghaddas Gholami, A., Kuster, B., Discovery of O-GlcNAc-modified proteins in published large-scale proteome data. *Mol. Cell. Proteomics* 2012, *11*, 843–850.

[44] Anderson, D. P., *Fifth IEEE/ACM International Workshop on Grid Computing 2004 Proceedings*, Pitsburgh, PA, 2004, pp. 4–10.

[45] Kaufmann, K. W., Lemmon, G. H., DeLuca, S. L., Sheehan, J. H., Meiler, J., Practically useful: what the Rosetta protein modeling suite can do for you. *Biochemistry* 2010, *49*, 2987–2998.

[46] Rohl, C. A., Strauss, C. E. M., Misura, K. M. S., Baker, D., in: Brand, L., Johnson, M. L. (Eds.), *Methods in Enzymology*, Vol. *383*, Academic Press, Salt Lake City, Utah, 2004, pp. 66–93.

[47] Rost, B., Review: protein secondary structure prediction continues to rise. *J. Struct. Biol.* 2001, *134*, 204–218.

[48] Dill, K. A., MacCallum, J. L., The protein-folding problem, 50 years on. *Science* 2012, *338*, 1042–1046.

[49] Mell, P., Grance, T., The NIST Definition of Cloud Computing n.d. http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf.

[50] Halligan, B. D., Geiger, J. F., Vallejos, A. K., Greene, A. S., Twigger, S. N., Low cost, scalable proteomics data analysis using Amazon's cloud computing services and open source search algorithms. *J. Proteome Res.* 2009, *8*, 3148–3153.

[51] Mohammed, Y., Mostovenko, E., Henneman, A. A., Marissen, R. J. et al., Cloud parallel processing of tandem mass spectrometry based proteomics data. *J. Proteome Res.* 2012, *11*, 5101–5108.

[52] Deutsch, E. W., Mendoza, L., Shteynberg, D., Farrah, T. et al., A guided tour of the Trans-Proteomic Pipeline. *Proteomics* 2010, *10*, 1150–1159.

[53] Hull, D., Wolstencroft, K., Stevens, R., Goble, C. et al., Taverna: a tool for building and running workflows of services. *Nucleic Acids Res.* 2006, *34*, W729–W732.

[54] Wolstencroft, K., Haines, R., Fellows, D., Williams, A. et al., The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud. *Nucleic Acids Res.* 2013, *34*, 729–732.

[55] Trudgian, D. C., Mirzaei, H., Cloud CPFP: a shotgun proteomics data analysis pipeline using cloud and high performance computing. *J. Proteome Res.* 2012, *11*, 6282–6290.

[56] Muth, T., Peters, J., Blackburn, J., Rapp, E., Martens, L., ProteoCloud: a full-featured open source proteomics cloud computing pipeline. *J. Proteomics* 2013, *88*, 104–108.

[57] Tanner, S., Shu, H., Frank, A., Wang, L.-C. et al., InsPecT: identification of posttranslationally modified peptides from tandem mass spectra. *Anal. Chem.* 2005, *77*, 4626–4639.

[58] Frank, A., Pevzner, P., PepNovo: de novo peptide sequencing via probabilistic network modeling. *Anal. Chem.* 2005, *77*, 964–973.

[59] Käll, L., Storey, J. D., Noble, W. S., QVALITY: non-parametric estimation of q-values and posterior error probabilities. *Bioinformatics* 2009, *25*, 964–966.

[60] Ji, J., Ling, J., Jiang, H., Wen, Q. et al., Cloud-based solution to identify statistically significant MS peaks differentiating sample categories. *BMC Res. Notes* 2013, *6*, 109.

[61] Liotta, L. A., Ferrari, M., Petricoin, E., Clinical proteomics: written in blood. *Nature* 2003, *425*, 905–905.

[62] Kussmann, M., Raymond, F., Affolter, M., OMICS-driven biomarker discovery in nutrition and health. *J. Biotechnol.* 2006, *124*, 758–787.

[63] Pang, J. X., Ginanni, N., Dongre, A. R., Hefta, S. A., Opiteck, G. J., Biomarker discovery in urine by proteomics. *J. Proteome Res.* 2002, *1*, 161–169.

[64] Good, B. M., Su, A. I., Crowdsourcing for bioinformatics. *Bioinformatics* 2013, *29*, 1925–1933.

[65] Barsnes, H., Martens, L., Crowdsourcing in proteomics: public resources lead to better experiments. *Amino Acids* 2013, *44*, 1129–1137.

[66] Khatib, F., DiMaio, F., Foldit Contenders Group, Foldit Void Crushers Group et al., Crystal structure of a monomeric retroviral protease solved by protein folding game players. *Nat. Struct. Mol. Biol.* 2011, *18*, 1175–1177.

[67] Eiben, C. B., Siegel, J. B., Bale, J. B., Cooper, S. et al., Increased Diels-Alderase activity through backbone remodeling guided by Foldit players. *Nat. Biotechnol.* 2012, *30*, 190–192.

[68] Cooper, S., Khatib, F., Treuille, A., Barbero, J. et al., Predicting protein structures with a multiplayer online game. *Nature* 2010, *466*, 756–760.

[69] Amdahl, G. M., *Proceedings of the Spring Joint Computer Conference, April 18–20, 1967*, ACM, New York, NY, USA 1967, pp. 483–485.

[70] Hill, M. D., Marty, M. R., Amdahl's law in the multicore era. *Computer* 2008, *41*, 33–38.

[71] Van der Burgt, Y. E. M., Taban, I. M., Konijnenburg, M., Biskup, M. et al., Parallel processing of large datasets from NanoLC-FTICR-MS measurements. *J. Am. Soc. Mass Spectrom.* 2007, *18*, 152–161.

[72] Tavakolan, M., Alkharouf, N. W., Khan, F. H., Natarajan, S., SoyProDB: a database for the identification of soybean seed proteins. *Bioinformation* 2013, *9*, 165–167.

[73] Madrid-Aliste, C. J., Dybas, J. M., Angeletti, R. H., Weiss, L. M. et al., EPIC-DB: a proteomics database for studying Apicomplexan organisms. *BMC Genomics* 2009, *10*, 38.

[74] Campbell, M. P., Nguyen-Khuong, T., Hayes, C. A., Flowers, S. A. et al., Validation of the curation pipeline of UniCarb-DB: building a global glycan reference MS/MS repository. *Biochim. Biophys. Acta* 2013, doi:pii: S1570-9639(13)00172-6.

[75] Desiere, F., Deutsch, E. W., King, N. L., Nesvizhskii, A. I. et al., The PeptideAtlas project. *Nucleic Acids Res.* 2006, *34*, D655–D658.

[76] Deutsch, E. W., Lam, H., Aebersold, R., PeptideAtlas: a resource for target selection for emerging targeted proteomics workflows. *EMBO Rep.* 2008, *9*, 429–434.

[77] Smedley, D., Haider, S., Ballester, B., Holland, R. et al., BioMart—biological queries made easy. *BMC Genomics* 2009, *10*, 22.

[78] Zhang, J., Haider, S., Baran, J., Cros, A. et al., BioMart: a data federation framework for large collaborative projects. *Database* 2011, *2011*, bar038.

[79] Kasprzyk, A., BioMart: driving a paradigm change in biological data management. *Database* 2011, *2011*, bar049.

[80] Guberman, J. M., Ai, J., Arnaiz, O., Baran, J. et al., BioMart Central Portal: an open database network for the biological community. *Database* 2011, *2011*, bar041.

[81] Thorisson, G. A., Smith, A. V., Krishnan, L., Stein, L. D., The International HapMap Project Web site. *Genome Res.* 2005, *15*, 1592–1593.

[82] Harris, T. W., Chen, N., Cunningham, F., Tello-Ruiz, M. et al., WormBase: a multi-species resource for nematode biology and genomics. *Nucleic Acids Res.* 2004, *32*, D411–D417.

[83] Jaiswal, P., Gramene database: a hub for comparative plant genomics. *Methods Mol. Biol.* 2011, *678*, 247–275.

[84] Kreppel, L., Fey, P., Gaudet, P., Just, E. et al., dictyBase: a new *Dictyostelium discoideum* genome database. *Nucleic Acids Res.* 2004, *32*, D332–D333.

[85] Kinsella, R. J., Kähäri, A., Haider, S., Zamora, J. et al., Ensembl BioMarts: a hub for data retrieval across taxonomic space. *Database* 2011, *2011*, bar030.

[86] Apweiler, R., Bairoch, A., Wu, C. H., Barker, W. C. et al., UniProt: the Universal Protein knowledgebase. *Nucleic Acids Res.* 2004, *32*, D115–D119.

