RAPID COMMUNICATIONS IN MASS SPECTROMETRY
Rapid Commun. Mass Spectrom. 2010; **24**: 1791–1798
Published online in Wiley InterScience (www.interscience.wiley.com) DOI: 10.1002/rcm.4578

RCM

An efficient parallelization of phosphorylated peptide and protein identification

Leheng Wang[1,2†], Wenping Wang[1,2,3†], Hao Chi[1,2,3], Yanjie Wu[1,2,3], You Li[1,2,3], Yan Fu[1,2], Chen Zhou[1,2,3], Ruixiang Sun[1,2], Haipeng Wang[1,2,3], Chao Liu[1,2,3], Zuofei Yuan[1,2,3], Liyun Xiu[1,2,3] and Si-Min He[1,2*]

[1]Key Lab of Intelligent Information Processing, Chinese Academy of Sciences, Beijing 100190, P.R. China
[2]Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, P.R. China
[3]Graduate University of Chinese Academy of Sciences, Beijing 100049, P.R. China

Received 8 November 2009; Revised 19 March 2010; Accepted 12 April 2010

Protein sequence database search based on tandem mass spectrometry is an essential method for protein identification. As the computational demand increases, parallel computing has become an important technique for accelerating proteomics data analysis. In this paper, we discuss several factors which could affect the runtime of the pFind search engine and build an estimation model. Based on this model, effective on-line and off-line scheduling methods were developed. An experiment on the public dataset from PhosphoPep consisting of 100 RAW files of phosphopeptides shows that the speedup on 100 processors is 83.7. The parallel version can complete the identification task within 9 min, while a stand-alone process on a single PC takes more than 10 h. On another larger dataset consisting of 1 366 471 spectra, the speedup on 320 processors is 258.9 and the efficiency is 80.9%. Our approach can be applied to other similar search engines. Copyright © 2010 John Wiley & Sons, Ltd.

Shotgun proteomics based on tandem mass spectrometry coupled with liquid chromatography (LC/MS/MS) have become key techniques for biological research. An important task in proteomics is to automatically identify peptides and proteins.[1,2] The database search approach addresses this problem by assigning known peptide sequences to observed tandem mass spectra.[3] The popular commercial search engines include SEQUEST,[4] Mascot[5] and Phenyx.[6,7] Some open-source tools have also been developed, e.g. X!Tandem[8] and OMSSA.[9] In our earlier work, we have developed the software pFind,[10–14] which is employed as a platform in this paper.

There are many reasons leading to a significant explosion of computational demand in database search. Firstly, with the development of liquid chromatography and mass spectrometry, the generating rate of spectra greatly exceeds the upgrading rate of computer hardware.[15] The size of protein databases is also increasing significantly,[16,17] as depicted in Fig. 1. Moreover, the variable modifications cause combinatorial explosion when searching the database, such as phosphorylation which occurs frequently in eukaryotic proteins, as illustrated in Fig. 2 by pScan.[18] In addition, in proteogenomics,[19,20] tandem mass spectra may be searched against a multi-species genome database directly.[20–24]

Since the computational demand is increasing remarkably, the identification speed is becoming a bottleneck in high-throughput proteomics. The following are the common speedup methods:

1. Spectra preprocessing: filter out the noise peaks, remove low-quality tandem mass spectra and adjust the charge states of precursor ions;[25–29]
2. Spectra clustering: integrate similar tandem mass spectra into a representative one to carry out the database search;[30,31]
3. Iterative identification: first, launch a pre-search with enzyme-specific digestion and common modifications. Then, run a series of sub-searches to prune search space iteratively with more and more stringent settings, such as non-specific digestion and more modifications.;[6,7,32,33]
4. Filtration via *de novo* technique: integrate *de novo* sequencing with database search to filter the candidate peptides;[34–38]
5. Space-for-time substitution: use pre-computing and indexing techniques, instead of repeated calculating;[39–45]
6. Hardware acceleration: use *graphics processing units* (GPU), *field programmable gate arrays* (FPGA) and other hardware technologies to speed up the 'hot spot' modules of a search engine;[46–48]
7. Parallel technology: distribute the computational load efficiently among a lot of computers. With in-depth software design, a 100-fold acceleration may be obtained in a large-scale cluster.

Some of the methods, such as iterative identification, have the potential to affect the accuracy of identification. On the other hand, parallel computing is a lossless accelerating method, which means the search results of the parallel version and the stand-alone version should be exactly the

*Correspondence to: S.-M. He, Key Lab of Intelligent Information Processing, Chinese Academy of Sciences, Beijing 100190, P.R. China.
E-mail: smhe@ict.ac.cn
†These authors contributed equally to this work.

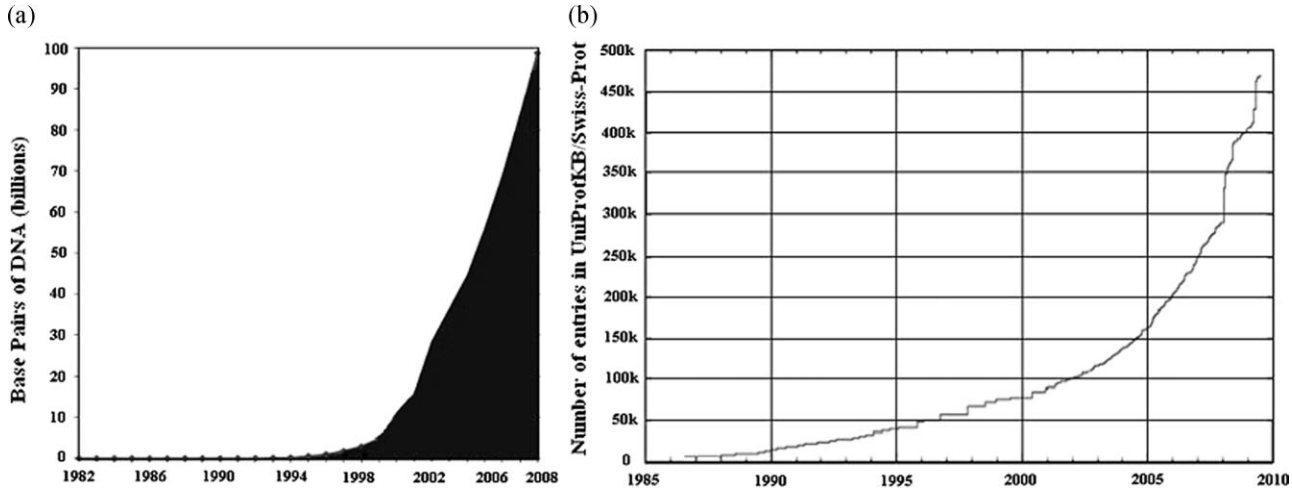

Figure 1. The size of protein sequence databases grows significantly. (a) The growth of the base pairs of DNA in GenBank (1982–2008). (b) The growth of the UniProtKB/Swiss-Prot protein knowledge database (1985–2010).

same. Moreover, parallel computing can be combined with other methods. For instance, the iterative identification method has been successfully combined with parallel computing in X!Tandem[49,50] and Phenyx.[6,7] With the popularity of cluster hardware, parallel computing has become a practical solution for speeding up protein identification, which is the focus of this paper.

Most of the peptide and protein search engines have their own parallel implementations: SEQUEST uses *parallel virtual machine* (PVM) to build its cluster system,[29] while Mascot and Phenyx use *message passing interface* (MPI). As an open-source software, X!Tandem has more than one parallel implementation.[49,50] Additionally, these systems have been integrated into higher-level application frameworks, such as web service or grid.[51–54] Furthermore, in the field of high-performance parallel computing, the so-called cloud computing is becoming more and more popular. Halligan has migrated X!Tandem and OMSSA to the Amazon cloud computing platform.[55]

For parallel computing, scheduling is the key issue. Its main objective is load balancing among all computing nodes. In other words, the tasks at each node should complete at exactly the same time if possible, in order to minimize the total runtime.[56–58] Most of the parallel protein identification systems only adopt random-like scheduling methods, in which the efficiency of load balancing cannot be assured. Deciu *et al*. pointed out that the effectiveness of any scheduling method depended on the accurate prediction of the search time for each tandem mass spectrum.[59] They investigated SEQUEST and built a model to estimate the search time. The model was used in their scheduling algorithm and reduced the total runtime significantly. However, more and more search engines, including pFind, applied the large-scale indexing and dispatching technology to speed up the identification.[42,44,45] The indexing has greatly improved the identification speed of the stand-alone version, but has also brought more complexity to the parallel scheduling. It is not enough to use the existing methods.

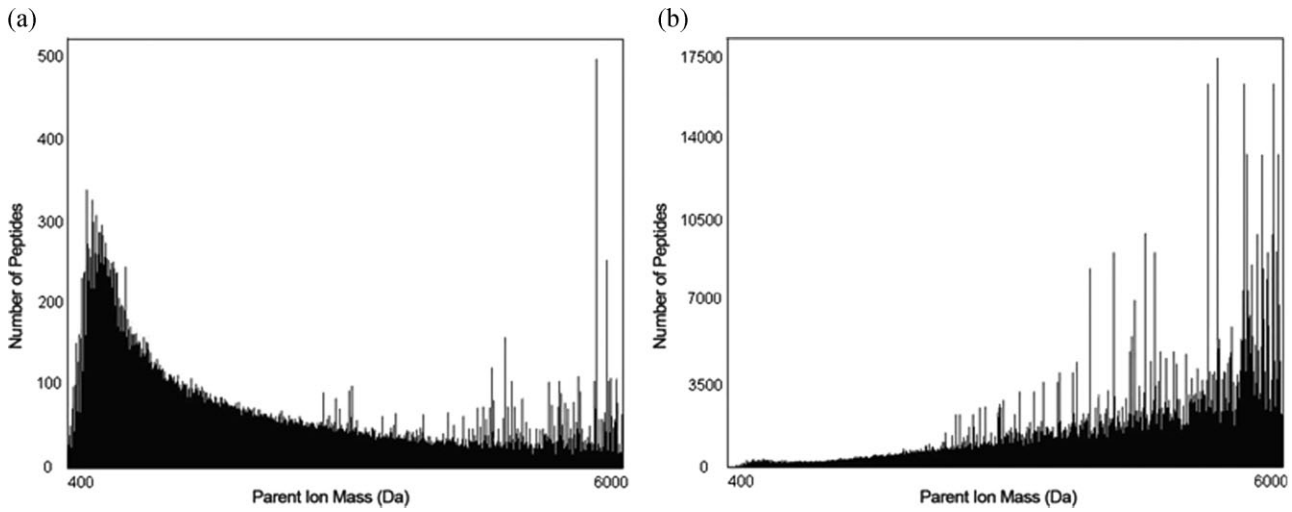

Figure 2. The mass distribution of phosphorylated and unmodified peptides. The number of candidates of phosphorylated peptides is hundreds of times more than that of the unmodified ones. IPI.HUMAN.v3.55 is used to calculate the distribution by pScan. The parameters include phosphorylation as variable modification. For the *in silico* digestion, trypsin is specified as protease. Up to two missed cleavages are allowed for the peptides. (a) The distribution of precursor ion masses of all unmodified peptides. (b) The distribution of precursor ion masses of all phosphorylated peptides.

DOI: 10.1002/rcm

Based on the runtime estimation of pFind, we have designed two different scheduling algorithms. An experiment is conducted on a public dataset from PhosphoPep[60] which consists of 100 RAW files of phosphorylated peptides. It shows that the speedup on 100 processors is 83.7 and the efficiency is 83.7%. In another experiment on a larger dataset consisting of 1 366 471 tandem mass spectra, the speedup on 320 processors is 258.9 and the efficiency is 80.9%. Both scheduling methods showed good scalability. The methods and results are detailed in the following sections.

EXPERIMENTAL

Dataset and parameters

Three different phosphopeptides datasets were used in our experiments: The first dataset is mouse liver phosphopeptides[61] containing 108 870 tandem mass spectra. The database is IPI Mouse 3.68, which contains 56 729 protein sequences. The second dataset is selected from PhosphoPep,[60] which contains 190 711 tandem mass spectra. The database is FlyBase 5.7, which contains 21 129 protein sequences. The largest dataset of synthetic phosphopeptides is provided by the *Shanghai Institutes for Biological Sciences* (SIBS). It contains 1 366 609 individual tandem mass spectra totaling 7.36 GB in size. The database is *E. coli* proteins with additional standard sequences and contains 4448 protein sequences in all.

To estimate the *false discovery rate* (FDR) of peptide-spectrum matches, the target-decoy strategy has been used in searching. This widely adopted strategy is based on the principle that incorrect matches have an equal probability of being derived from either the target or the decoy database.[62–64] In this strategy, the composite databases contain protein sequences in both forward and reverse orientation. After scoring, the FDR can be estimated as $FDR = 2 * \#R/(\#R + \#F)$, where #R is the number of spectra identified from the reverse sequences, and #F is the number of spectra identified from the forward ones.

The search parameters include carbamidomethylation on cysteine as fixed modification and phosphorylation on serine, threonine and tyrosine and oxidation on methionine as variable modifications. For the *in silico* digestion, trypsin is specified as protease. Up to two missed cleavages are allowed for the peptides. The precursor ion mass tolerance is ± 3 Da, and the fragment ion tolerance is ± 0.5 Da.

Cluster platform

A cluster at the *National Institute of Biological Sciences, Beijing* (NIBS) was used for the experiments. Each node of this cluster has 4 GB RAM and 2 CPU. Each CPU has four cores with a clock speed of 1.95 GHz. The nodes are interconnected via switched gigabit Ethernet.

Another cluster, *Dawning 5000*,[65] was also used for the experiments. Each node has 64 GB RAM and 4 CPU. Each CPU has four cores with a clock speed of 1.9 or 2.2 GHz. The nodes are interconnected via InfiniBand. It is a heterogeneous cluster, where nodes do not have the same computing power.

The software tools for the experiments involve pFind 2.2, MPICH 2-1.0.8, ACE 5.6, GCC compiler 4.2.0.

We implement the pFind cluster architecture based on MPI. A pFind cluster consists of a single master node and multiple of slave nodes. The master node assigns search tasks to particular slave nodes and manages a registry service maintaining information.

METHODS AND RESULTS

Splitting tandem mass spectra into subtasks

In a database search, one frequently invoked but time-consuming step is the dispatching module. In this module, candidate peptides whose masses match the m/z values within a mass tolerance window are assigned to the corresponding tandem mass spectra. The peptide indexing technique has been applied in the query process of dispatching module of many search engines, such as pFind. Before identification, the indexing process digests proteins *in silico*, removes redundant peptides, sorts unique peptides by mass and stores them into indexing file. Then the dispatching module of search engine sorts all spectra by their precursor ion masses, sequentially loads peptide sequences from indexing file, generates candidate peptides with all possible compositions of modifications for each sequence, and matches them with the corresponding tandem mass spectra. This workflow scans the peptide index only once to assign candidate peptides to each tandem mass spectrum so that the search efficiency is increased. The time complexity of the dispatching module is $O(N + M)$, where the variable N indicates the number of peptides and the variable M indicates the number of tandem mass spectra. Considering that N is usually far larger than M, the performance of dispatching mainly depends on the number of scanned peptides. The denser the precursor ion masses of the tandem mass spectra are, the fewer peptides are scanned, and the higher the performance achieved.

Tasks have to be assigned to every node of the cluster in parallel computing. How to split the tandem mass spectra into subtasks is a key issue. As described above, those tandem mass spectra with similar precursor ion mass should be assigned to the same subtask, in order to improve the performance of the dispatching module. Therefore, we should sort the spectra by mass before dividing them into subtasks. An experiment on the mouse liver dataset shows that without sorting, the total serial searching time of all subtasks will be at least twice as long as in the general case (Table 1).

Parallel scheduling

Scheduling is a critical component of any parallel algorithm. A parallel algorithm has to divide the large number of spectra into smaller tasks and schedule the tasks onto the machines.[66] Two important measurements of parallel quality are speedup and efficiency. If T_1 is the time of running the fastest serial algorithm on a single processor and T_n is the

Table 1. The performance of different splitting methods on the mouse liver dataset

	Serial runtime (s)
Subtasks formed from sorted spectra	54014.1
Random subtasks	150190.1

DOI: 10.1002/rcm

time of a parallel algorithm running on N processors, then the speedup is defined as Speedup = T_1/T_n and the efficiency is defined as Efficiency = Speedup/N * 100%. Efficiency shows how well all processors are utilized. An ideal efficiency of 100% means that all processors are being fully used all the time. In our experiments, the scheduling algorithms are carried out on a different number of processors and speedups and efficiencies are calculated respectively.

The goal of scheduling is to minimize the completion time of a parallel search engine by balancing the load over the various nodes, because an inappropriate scheduling cannot exploit the true potential of the system, and the benefit gained from parallelization will be offset.[67] Generally speaking, there are two ways of scheduling: off-line[57,58] and on-line,[56] also named static and dynamic scheduling. With the off-line method, all the tasks have to be assigned to the proper working nodes before executing. The off-line method assumes that the runtime estimation is good enough and we can create a single meta-task for each processor. On the other hand, the on-line method assigns each task in real time. The master process initializes the slave processes and the subtask list. The subtasks are assigned to slave processes and searched respectively. When a slave finishes its current subtask, it will signal the master and receive another subtask until all subtasks are finished. With the on-line method, although the runtime estimation is not perfect, the parallelization will still be satisfactory. Different from the off-line method, the on-line method needs smaller, more numerous subtasks to get load balancing. Parallel SEQUEST[29] is based on the on-line scheduling method, while the two parallel implementations of X!Tandem[49,50] use off-line scheduling methods.

We have implemented two different scheduling algorithms in the parallel version of pFind, on-line and off-line scheduling, as detailed in the following sections.

On-line scheduling algorithm

From the point of view of load balance, it is desirable to split the spectra into as many subtasks as possible. However, as discussed previously, the dispatching module gains in efficiency by grouping together many spectra of similar ion masses. Therefore, we must find a balance between these two considerations. Experiments show that 100 is a reasonable number of spectra to group into a single task, as Fig. 3 shows. This number will obviously vary somewhat by dataset.

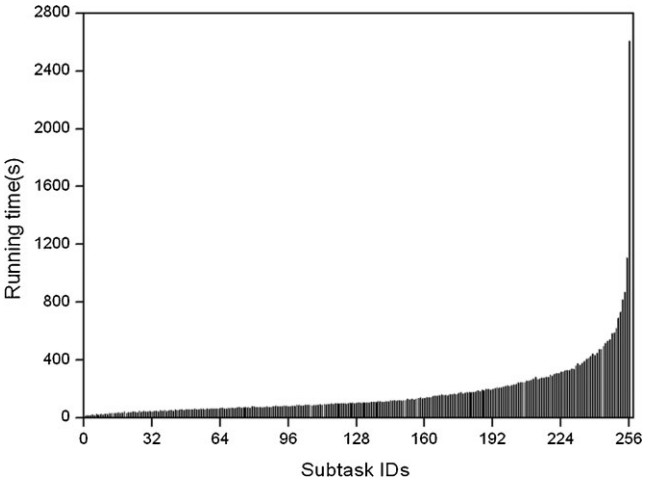

Figure 4. The runtime of each subtask. On the dataset from PhosphoPep, if the tandem mass spectra are sorted by precursor ion mass and divided equally into 256 subtasks, the runtime of each subtask has an incremental relationship. Thus, the average precursor ion mass is a good predictor of the relative runtime for the subtask.

For the on-line scheduling, one efficient heuristic is the *longest processing time* (LPT) first rule, which has been shown to be a 4/3-approximation.[67,68] In other words, LPT is guaranteed to generate a solution no worse than 4/3 optimal. The LPT rule assigns the n largest jobs first. After that, whenever a node is free, the longest job among those not yet processed is assigned to that node. This heuristics places shorter jobs toward the end of the schedule, where they can be used to balance the loads.[66] For the LPT rule, we do not need to estimate the runtime accurately. It is sufficient to order the subtasks by runtime. As an experiment (Fig. 4)

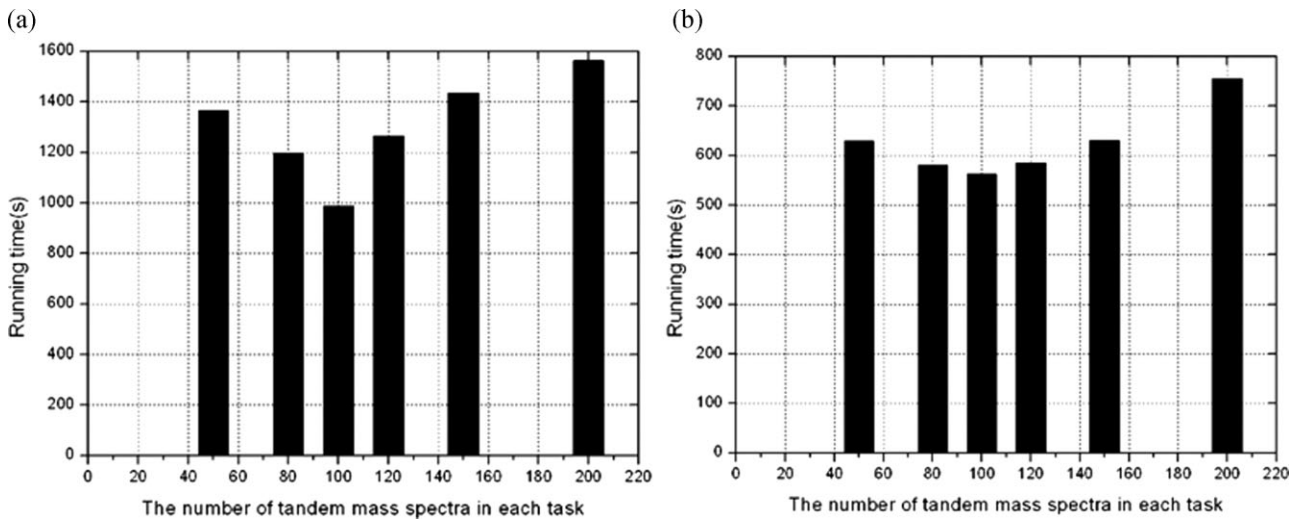

Figure 3. The relationship between the subtask size and the performance. Based on different subtask sizes, e.g. 50, 80, 100, two different datasets were searched on 96 processors with on-line scheduling, both of which indicated that the number 100 is a satisfactory trade-off point. (a) The experiment on the dataset of mouse liver phosphopeptides. (b) The experiment on the dataset from PhosphoPep.

DOI: 10.1002/rcm

Table 2. The performance of on-line scheduling using the mouse liver dataset

#processors	1	16	32	48	64
runtime (s)	54014.1	3397.1	1759.4	1286.1	1044.8
speedup	~	15.9	30.7	42.0	51.7
efficiency (%)	~	99.4	95.9	87.5	80.8

shows if the spectra are sorted by precursor ion masses and divided equally into subtasks, the runtime of subtasks will increase with the average ion masses. The average precursor ion mass is a good predictor of the relative runtime for the subtask. As a result, the tandem mass spectra are sorted by precursor ion mass in ascending order first, and then are divided equally into subtasks, each of which contains approximately 100 tandem mass spectra. If subtasks are assigned to slave processes and searched in descending order, then the LPT rule is satisfied.

The experiments on the three datasets show good speedup and scalability (Tables 2, 3, 4).

Off-line scheduling algorithm

The on-line scheduling algorithm described above cannot perfectly predict the runtime. We have investigated the various factors affecting the runtime, and designed an off-line scheduling algorithm.

Firstly, an open-source performance analysis tool, OProfile, was utilized to measure the proportion of runtime for each module in the search engine. In general, more than 70% of the runtime is occupied by the scoring module alone. Therefore, the scoring module is the 'hot spot'. Furthermore, the proportion of runtime of the scoring module will be more than 85% when searching data of phosphorylated peptides, as showed in Fig. 5, since billions of candidate peptides may be matched with the tandem mass spectra by the scoring module. Consequently, we can approximate the overall runtime by estimating the runtime of the scoring module alone.

Table 3. The performance of on-line scheduling using the dataset from PhosphoPep

#processors	1	16	32	64	96	100	128	160
runtime (s)	44833.3	2817.6	1417.1	752.4	562.5	535.4	467.2	414.6
speedup	~	15.9	31.6	59.6	79.7	83.7	96.0	108.1
efficiency (%)	~	99.4	98.8	93.1	83.0	83.7	75.0	67.6

Table 4. The performance of on-line scheduling using SIBS dataset

#processors	1	32	64	96	128	160
runtime (s)	259136.5	8123.4	4084.6	2790.0	2155.4	1790.3
speedup	~	31.9	63.4	92.9	120.2	144.7
efficiency (%)	~	99.7	99.1	96.8	93.9	90.4

#processors	192	224	256	288	320
runtime (s)	1533.4	1353.2	1218.3	1096.2	1000.9
speedup	169.0	191.5	212.7	236.4	258.9
efficiency (%)	88.0	85.5	83.1	82.1	80.9

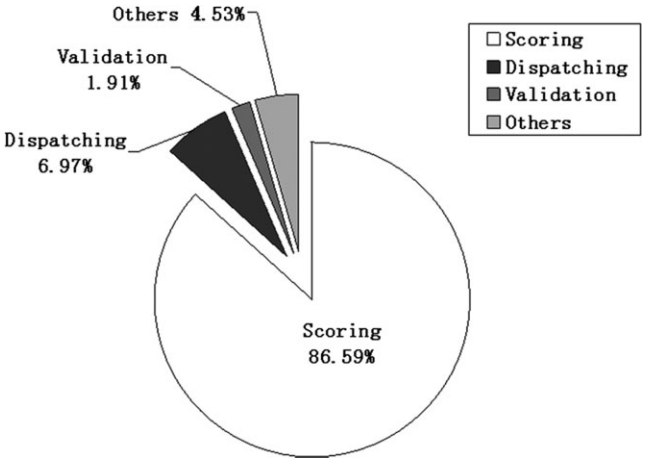

Figure 5. The proportion of runtime of each module in the search engine. More than 85% of the runtime is occupied by the scoring module alone.

Secondly, to estimate the runtime of the scoring module, we have predicted the time cost for each match between a candidate peptide and a tandem mass spectrum. The time cost involves many factors, such as the length of peptide, the number of modification sites and the charge states. The time complexity of each peptide-spectrum match is O(N), where the variable N indicates the size of the theoretical fragment ion list (determined by the length of peptide, the number of modification sites and the charge state). An experiment (Fig. 6) has been carried out to support this theoretical prediction. The estimation needs some necessary detail information, i.e. the charge state, the modification sites and the length of each candidate peptide of each spectrum. The problem is how to get the information before the actual database searching with the off-line method. We can run the dispatching module in parallel (but without the scoring and validation modules), and collect the necessary information in advance.

Thirdly, the distribution of the tandem mass spectra whose masses are above 3500 Da becomes very sparse in most cases. Figure 7(a) illustrates this trend. As described above, a decrease in density leads to a performance decline of the dispatching module. Therefore, when it comes to the mass region higher than 3500 Da, the proportion of runtime for the scoring module should be adjusted lower, as showed by

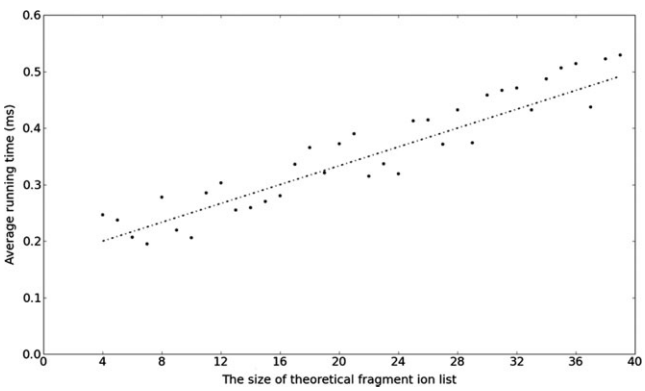

Figure 6. A linear relationship between the size of theoretical fragment ion list and the scoring runtime.

DOI: 10.1002/rcm

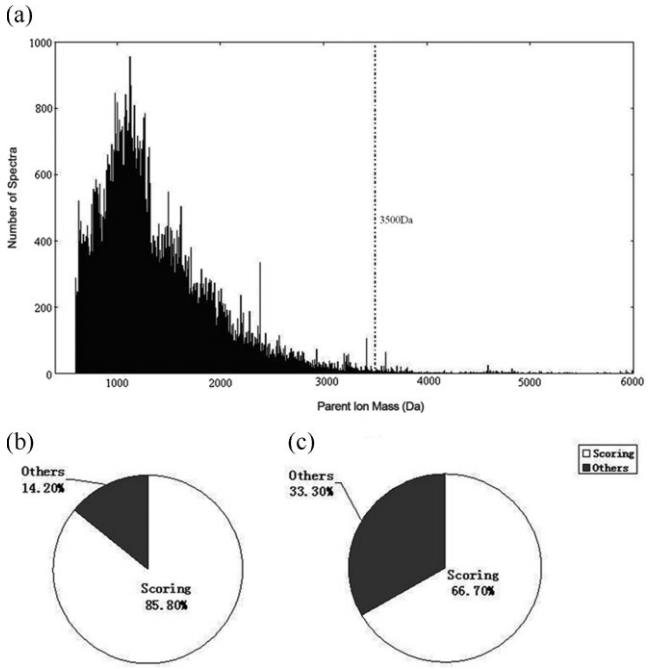

Figure 7. The runtime of high mass range. (a) The distribution of the tandem mass spectra whose masses are above 3500 Da (whose proportion is only 1%) becomes very sparse. (b) For the general tandem mass spectra, the proportion of runtime of the scoring module is over 85%. (c) When it comes to the mass region higher than 3500 Da, the proportion of runtime of the scoring module will decline to 65%.

Figs. 7(b) and 7(c). So the penalty for high mass must be added into the runtime estimation algorithm.

Considering the above, a model has been built to estimate the identification time (see Supplementary Algorithm 1, Supporting Information).

If we get the runtime estimation, load balancing will become the makespan problem, which is defined as $\max(C_1, \ldots, C_n)$, where C_j is the computation time for the jth task, is equivalent to the completion time of the last job to leave the system. A minimum makespan usually implies a high utilization of the machines. The makespan problem is a partition problem[69] which is to decide whether a given multiset of integers can be partitioned into two or more subsets that have the same sum. It is an NP-hard problem.[66,70,71] In our solution, a simple greedy algorithm for partition (see Supplementary Algorithm 2, Supporting Information) was applied and acceptable effect was achieved. Different from the on-line scheduling, a single meta-task is created for each processor so that the communications between master and slaves are less.

The experiments (Tables 5 and 6) for off-line scheduling show good performance. The experiment on the public dataset from PhosphoPep shows that the speedup on 100 processors is 81.2. When it comes to more than 100 processors, the off-line scheduling method shows a higher speedup than the on-line scheduling method.

Table 5. The performance of off-line scheduling using the mouse liver dataset

#processors	1	16	32	48	64
runtime (s)	54014.1	3577.1	1869.0	1304.7	1034.8
speedup	~	15.1	28.9	41.4	52.2
efficiency	~	94.4%	90.3%	86.3%	81.6%

Table 6. The performance of off-line scheduling using the dataset from PhosphoPep

#processors	1	16	32	64	96	100	128	160
runtime (s)	44833.3	3350.6	1678.0	844.1	575.3	552.1	444.1	381.2
speedup	~	13.4	26.7	53.1	77.9	81.2	101.0	117.6
efficiency (%)	~	83.8	83.4	83.0	81.2	81.2	78.9	73.5

RESULTS

Figures 8, 9 and 10 depict the trends of the two scheduling methods. Both of them are close to linear speedup. On a smaller public dataset from PhosphoPep consisting of 100 RAW files, when the number of processors increases to 100, the speedup is 83.7. This means that the parallel version on a usual cluster can complete the task in 535 s, while the runtime of a stand-alone process on a single PC is more than 10 h. On another larger dataset, the speedup on 320 processors is 258.9 and the efficiency is 80.9%.

The two scheduling methods have their own features and scope of application. With more than 100 processors, the off-line scheduling method showed better scalability. The reason is that the on-line scheduling of thousands of subtasks requires more coordination than the off-line scheduling method which incorporated only one subtask for each processor. However, the off-line scheduling is only suitable for a homogeneous cluster environment, in which all nodes are exactly the same (discussed below). On a heterogeneous cluster, the on-line scheduling has better adaptability.

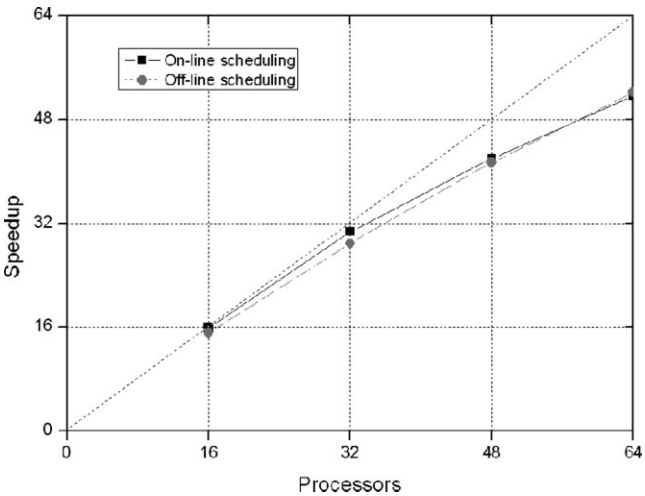

Figure 8. The speedups on the dataset of mouse liver phosphopeptides. The speedups of on-line and off-line scheduling methods on the dataset of mouse liver phosphopeptides. The experiments were run on the Dawning 5000 cluster.

DOI: 10.1002/rcm

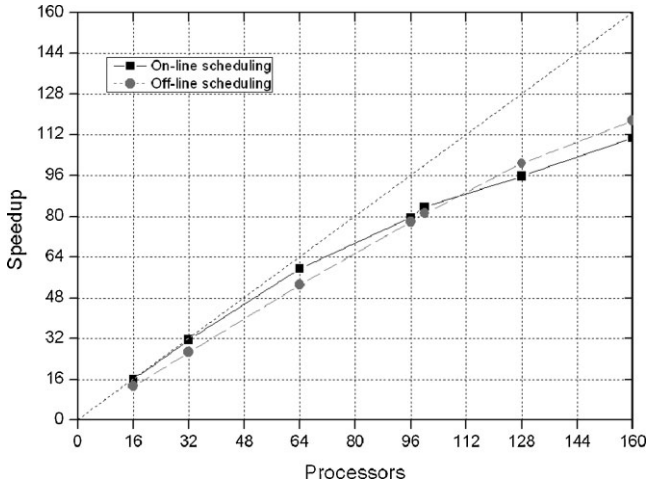

Figure 9. The speedups on the dataset from PhosphoPep. The speedups of on-line and off-line scheduling methods on the dataset from PhosphoPep. The experiments were run on the NIBS cluster.

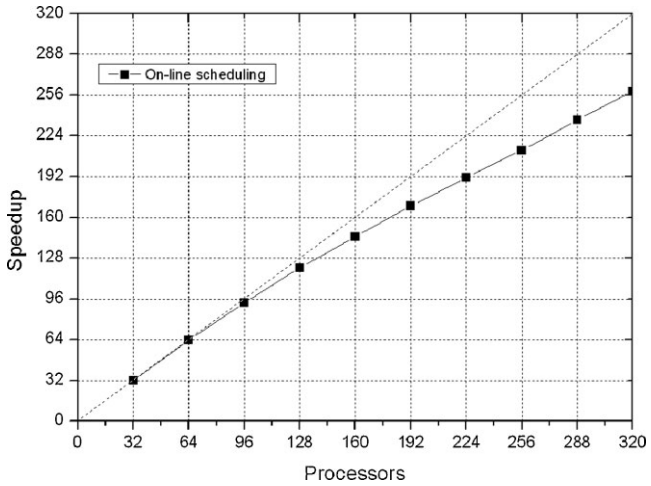

Figure 10. The speedups on the dataset of synthetic phosphopeptides from SIBS. The speedups of on-line scheduling methods on the dataset of synthetic phosphopeptides from SIBS. The experiments were run on the Dawning 5000 cluster.

DISCUSSION

Parallel computing is an important solution for speeding up tandem mass spectrometry-based protein identification. Based on the runtime estimation of tandem mass spectra, we have investigated two different scheduling algorithms.

The off-line scheduling method described in this paper is applied in a homogeneous cluster environment. When in a heterogeneous cluster environment, one option is to estimate the computing power of each node using a standard dataset *a priori*.

This paper mainly discussed the strategy for parallel search. However, it can also be combined with other acceleration technologies, like preprocessing and iterative identification strategies. With the improvement of hardware (like multi-core) technology, the combination of the distributed technology and the multi-threading technology will become important. Inside a single process, multi-threading can be used to accelerate the 'hot spot' modules, like the scoring module. FPGA or GPU technology can also be used to speed up the 'hot spot' modules.

Furthermore, large-scale tandem mass spectrometry data analysis and large-scale text information retrieval have many similarities. GFS,[72] MapReduce,[73] Bigtable[74] and Chubby[75] introduced by Google are successful industrial applications for large-scale parallel storage, indexing and query technologies. The architecture of Google's cluster,[76] which consists of thousands of computing nodes, is also valuable for parallel identification of tandem mass spectra.

Our future work will focus on large-scale parallel protein identification running on thousands of CPUs. Cloud computing will be introduced to pFind for providing the *software-as-a-service* (SAAS) searching.

SUPPORTING INFORMATION

Additional supporting information may be found in the online version of this article.

Acknowledgements

The authors thank Mengqiu Dong, Rong Zeng, Sujun Li, Quanhu Sheng, Chuan Wang, Bing Yang, Zhiyi Jing and Shengbo Fan for valuable discussions. This work was supported by the National High Technology Research and Development Program (863) of China under Grant Nos. 2007AA02Z315, 2008AA02Z309, the National Key Basic Research & Development Program (973) of China under Grant Nos. 2002CB713807 and 2010CB912701, and the CAS Knowledge Innovation Program under Grant No. KGGX1-YW-13.

REFERENCES

1. Aebersold R, Mann M. *Nature* 2003; **422**: 198.
2. Steen H, Mann M. *Nat. Rev. Mol. Cell. Biol.* 2004; **5**: 699.
3. Xu C, Ma B. *Drug Discov. Today* 2006; **11**: 595.
4. Eng JK, McCormack AL, Yates JR III. *J. Am. Soc. Mass. Spectrom.* 1994; **5**: 976.
5. Perkins DN, Pappin DJ, Creasy DM, Cottrell JS. *Electrophoresis* 1999; **20**: 3551.
6. Colinge J, Masselot A, Cusin I, Mahe E, Niknejad A, Argoud-Puy G, Reffas S, Bederr N, Gleizes A, Rey PA, Bougueleret L. *Proteomics* 2004; **4**: 1977.
7. Colinge J, Masselot A, Giron M, Dessingy T, Magnin J. *Proteomics* 2003; **3**: 1454.
8. Craig R, Beavis RC. *Bioinformatics* 2004; **20**: 1466.
9. Geer LY, Markey SP, Kowalak JA, Wagner L, Xu M, Maynard DM, Yang X, Shi W, Bryant SH. *J. Proteome Res.* 2004; **3**: 958.
10. Fu Y, Yang Q, Sun R, Li D, Zeng R, Ling CX, Gao W. *Bioinformatics* 2004; **20**: 1948.
11. Fu Y, Gao W, He SM, Sun RX, Zhou H, Zeng R. *Pacific Symposium on Biocomputing* 2007; **12**: 421.
12. Li D, Fu Y, Sun R, Ling CX, Wei Y, Zhou H, Zeng R, Yang Q, He S, Gao W. *Bioinformatics* 2005; **21**: 3049.
13. Wang LH, Li DQ, Fu Y, Wang HP, Zhang JF, Yuan ZF, Sun RX, Zeng R, He SM, Gao W. *Rapid Commun. Mass Spectrom.* 2007; **21**: 2985.
14. Jia W, Lu Z, Fu Y, Wang HP, Wang LH, Chi H, Yuan ZF, Zheng ZB, Song LN, Han HH, Liang YM, Wang JL, Cai Y, Zhang YK, Deng YL, Ying WT, He SM, Qian XH. *Mol. Cell. Proteomics* 2009; **8**: 913.
15. Mann M, Kelleher NL. *Proc. Natl. Acad. Sci. USA* 2008; **105**: 18132.
16. http://www.ncbi.nlm.nih.gov/Genbank/genbankstats.html.

DOI: 10.1002/rcm

17. http://www.expasy.ch/sprot/relnotes/relstat.html.
18. Wang W, Wang L, Wang H, Yuan Z, Chi H, Li Y, Xiu L, Liu C, Fu Y, Sun R, He S-M. *BMC Informatics* 2010; submitted.
19. Ansong C, Purvine SO, Adkins JN, Lipton MS, Smith RD. *Brief Funct. Genomic Proteomics* 2008; **7**: 50.
20. Gupta N, Benhamida J, Bhargava V, Goodman D, Kain E, Kerman I, Nguyen N, Ollikainen N, Rodriguez J, Wang J, Lipton MS, Romine M, Bafna V, Smith RD, Pevzner PA. *Genome Res.* 2008; **18**: 1133.
21. Yates JR III, Eng JK, McCormack AL. *Anal. Chem.* 1995; **67**: 3202.
22. Choudhary JS, Blackstock WP, Creasy DM, Cottrell JS. *Proteomics* 2001; **1**: 651.
23. Choudhary JS, Blackstock WP, Creasy DM, Cottrell JS. *Trends Biotechnol.* 2001; **19**: S17.
24. Gupta N, Tanner S, Jaitly N, Adkins JN, Lipton M, Edwards R, Romine M, Osterman A, Bafna V, Smith RD, Pevzner PA. *Genome Res.* 2007; **17**: 1362.
25. Zhang J, Gao W, Cai J, He S, Zeng R, Chen R. *IEEE/ACM T. Comp. Biol. Bioinfo.* 2005; **2**: 217.
26. Zhang JF, He SM, Cai JJ, Cao XJ, Sun RX, Fu Y, Zeng R, Gao W. *Genomics, Proteomics & Bioinformatics* 2005; **3**: 231.
27. Zhang J, He S, Ling CX, Cao X, Zeng R, Gao W. *Rapid Commun. Mass Spectrom* 2008; **22**: 1203.
28. Yang C, He Z, Yu W. *BMC Bioinformatics* 2009; **10**: 4.
29. Sadygov RG, Eng J, Durr E, Saraf A, McDonald H, MacCoss MJ, Yates JR III. *J. Proteome Res.* 2002; **1**: 211.
30. Beer I, Barnea E, Ziv T, Admon A. *Proteomics* 2004; **4**: 950.
31. Frank AM, Bandeira N, Shen Z, Tanner S, Briggs SP, Smith RD, Pevzner PA. *J. Proteome Res.* 2008; **7**: 113.
32. Creasy DM, Cottrell JS. *Proteomics* 2002; **2**: 1426.
33. Craig R, Beavis RC. *Rapid Commun. Mass Spectrom.* 2003; **17**: 2310.
34. Mann M, Wilm M. *Anal. Chem.* 1994; **66**: 4390.
35. Sunyaev S, Liska AJ, Golod A, Shevchenko A, Shevchenko A. *Anal. Chem.* 2003; **75**: 1307.
36. Kim S, Gupta N, Bandeira N, Pevzner PA. *Mol. Cell. Proteomics* 2009; **8**: 53.
37. Tabb DL, Saraf A, Yates JR III. *Anal. Chem.* 2003; **75**: 6415.
38. Chi H, Sun R-X, Yang B, Song C-Q, Wang L-H, Liu C, Fu Y, Yuan Z-F, Wang H-P, He S-M, Dong M-Q. *J. Proteome Res.* 2010; submitted.
39. Edwards N, Lippert R. *WABI 2002* Algorithms in Bioinformatics: Second International Workshop, 2002; 68.
40. Lu B, Chen T. *Bioinformatics* 2003; **19**: 113.
41. Tang WH, Halpern BR, Shilov IV, Seymour SL, Keating SP, Loboda A, Patel AA, Schaeffer DA, Nuwaysir LM. *Anal. Chem.* 2005; **77**: 3931.
42. Li D, Gao W, Ling CX, Wang X, Sun R, He S. *Bioinformatics* 2006; **22**: 2572.
43. Dutta D, Chen T. *Bioinformatics* 2007; **23**: 612.
44. Li Y, Chi H, Wang LH, Wang HP, Fu Y, Yuan ZF, Li SJ, Liu YS, Sun RX, Zeng R, He SM. *Rapid Commun. Mass Spectrom.* 2010; **24**: 807.
45. Zhou C, Chi H, Wang L-H, Li Y, Wu Y-J, Fu Y, Sun R-X, He S-M. *Rapid Commun. Mass Spectrom.* 2010; submitted.
46. Bogdan I, Coca D, Rivers J, Beynon RJ. *Bioinformatics* 2007; **23**: 724.
47. Dandass YS, Burgess SC, Lawrence M, Bridges SM. *BMC Bioinformatics* 2008; **9**: 197.
48. Hussong R, Gregorius B, Tholey A, Hildebrandt A. *Bioinformatics* 2009; **25**: 1937.
49. Bjornson RD, Carriero NJ, Colangelo C, Shifman M, Cheung KH, Miller PL, Williams K. *J. Proteome Res.* 2008; **7**: 293.
50. Duncan DT, Craig R, Link AJ. *J. Proteome Res.* 2005; **4**: 1842.
51. Dominic B, David Sigfredo A. *J. Parallel Distrib. Comput.* 2006; **66**: 1503.
52. Quandt A, Hernandez P, Kunzst P, Pautasso C, Tuloup M, Hernandez C, Appel RD. *Stud. Health Technol. Inform.* 2007; **126**: 13.
53. Zosso D, Podvinec M, Muller M, Aebersold R, Peitsch MC, Schwede T. *Stud. Health Technol. Inform.* 2007; **126**: 3.
54. Quandt A, Masselot A, Hernandez P, Hernandez C, Maffioletti S, Appel RDD, Lisacek F. *Proteomics* 2009; **9**: 2648.
55. Halligan BD, Geiger JF, Vallejos AK, Greene AS, Twigger SN. *J. Proteome Res.* 2009; **8**: 3148.
56. Ishfaq A, Arif G. *IEEE Trans. Software Eng.* 1991; **17**: 987.
57. Gajski DD, Jib-Kwon P. *Computer* 1985; **18**: 9.
58. Chu WWL, Min-Tsung; Hellerstein, Joseph. *Computers, IEEE Trans. Computers* 1984; 691.
59. Deciu C, Sun J, Wall MA. *J. Proteome Res.* 2007; **6**: 3443.
60. Bodenmiller B, Malmstrom J, Gerrits B, Campbell D, Lam H, Schmidt A, Rinner O, Mueller LN, Shannon PT, Pedrioli PG, Panse C, Lee HK, Schlapbach R, Aebersold R. *Mol. Syst. Biol.* 2007; **3**: 139.
61. Villen J, Beausoleil SA, Gerber SA, Gygi SP. *Proc. Natl. Acad. Sci. USA* 2007; **104**: 1488.
62. Peng J, Elias JE, Thoreen CC, Licklider LJ, Gygi SP. *J. Proteome Res.* 2003; **2**: 43.
63. MacCoss MJ. *Curr. Opin. Chem. Biol.* 2005; **9**: 88.
64. Elias JE, Haas W, Faherty BK, Gygi SP. *Nat. Methods* 2005; **2**: 667.
65. Sun N-H, Li K, Chen M-Y. *Chin. J. Computers* 2008; **31**: 1503.
66. Pinedo M. *Scheduling: Theory, Algorithms, and. Systems*, (2nd edn). Prentice Hall: New Jersey, 2002.
67. Yu-Kwong K, Ishfaq A. *ACM Comput. Surv.* 1999; **31**: 406.
68. Li Liu YY, Wanbing Shi, Wumeng Lin, Lian Li. *First International Conference on Semantics, Knowledge and Grid (SKG'05)*, 2005.
69. Mertens S. *The Easiest Hard Problem: Number Partitioning*. Oxford University Press: New York, 2003.
70. Garey MR, Johnson DS. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Co.: New York, 1979.
71. Cormen TH, Leiserson CE, Rivest RL, Stein C. *Introduction to Algorithms*, (2nd edn). MIT Press: Cambridge, MA, 2001.
72. Sanjay G, Howard G, Shun-Tak L. *SIGOPS Oper. Syst. Rev.* 2003; **37**: 29.
73. Dean J, Ghemawat S. *Commun. ACM* 2008; **51**: 107.
74. Chang F, Dean J, Ghemawat S, Hsieh WC, Wallach DA, Burrows M, Chandra T, Fikes A, Gruber RE. *7th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2006; 205.
75. Burrows M. *7th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 2006.
76. Barroso LA, Dean J, Holzle U. *Micro, IEEE* 2003; **23**: 22.

DOI: 10.1002/rcm