Published in final edited form as:

J Proteome Res. 2012 July 6; 11(7): 3581–3591. doi:10.1021/pr300338p.

Tempest: GPU-CPU computing for high-throughput database spectral matching

Jeffrey A. Milloy^{§,¶}, Brendan K. Faherty^{‡,¶}, and Scott A. Gerber^{§,¥,‡,*}

[§]Norris Cotton Cancer Center, Dartmouth-Hitchcock Medical Center, Lebanon, NH 03756

[‡]Department of Genetics, Geisel School of Medicine at Dartmouth, Lebanon, NH 03756

[¥]Department of Biochemistry, Geisel School of Medicine at Dartmouth, Lebanon, NH 03756

Abstract

Modern mass spectrometers are now capable of producing hundreds of thousands of tandem (MS/MS) spectra per experiment, making the translation of these fragmentation spectra into peptide matches a common bottleneck in proteomics research. When coupled with experimental designs that enrich for post-translational modifications such as phosphorylation and/or include isotopically-labeled amino acids for quantification, additional burdens are placed on this computational infrastructure by shotgun sequencing. To address this issue, we have developed a new database searching program that utilizes the massively parallel compute capabilities of a graphical processing unit (GPU) to produce peptide spectral matches in a very high throughput fashion. Our program, named Tempest, combines efficient database digestion and MS/MS spectral indexing on a CPU with fast similarity scoring on a GPU. In our implementation, the entire similarity score, including the generation of full theoretical peptide candidate fragmentation spectra and its comparison to experimental spectra, is conducted on the GPU. Although Tempest uses the classical SEQUEST XCorr score as a primary metric for evaluating similarity for spectra collected at unit resolution, we have developed a new “Accelerated Score” for MS/MS spectra collected at high resolution that is based on a computationally inexpensive dot product but exhibits scoring accuracy similar to the classical XCorr. In our experience, Tempest provides compute-cluster level performance in an affordable desktop computer.

Keywords

search algorithm; peptide spectral matching; graphical processing unit; GPU; tandem mass spectrometry; peptide identification

Introduction

Peptide spectral matching of tandem (MS/MS) spectra to translated genomic sequence databases is the preferred method for identifying peptides in a proteomics experiment^{1–5}. However, recent advancements in mass spectrometry instrumentation have enabled the generation of very high numbers of MS/MS spectra per experiment, which has resulted in greater attention to the speed of peptide spectral matching algorithms. Indeed, in a recent

*To whom correspondence should be addressed: Scott A. Gerber, Tel: 603-653-3679, Fax: 603-653-9923, scott.a.gerber@dartmouth.edu.

[¶]These authors contributed equally.

Supporting Information

Supporting Information Available: This material is available free of charge via the Internet at <http://pubs.acs.org>

review on information science in proteomics, Nesvizhskii maintains that “database search time remains an important consideration” in proteomics data analysis ⁶.

Database searching programs perform two primary functions: digestion and scoring. Traditionally, these functions are performed serially on a CPU. Efforts to improve the performance of the overall process have been described that use task parallelism to distribute the work across many serial processors ^{2, 4, 7}. Some solutions have been implemented on multi-core CPUs; other run across a cluster of CPUs.

Previous efforts to improve the throughput of database search algorithms have focused primarily on their digestion functions. For example, Crux indexes digested peptides in memory at runtime, in order to quickly select candidate peptides for scoring ⁸. DBDigger and MacroSEQUEST explicitly describe the implementation of a “candidate-centric” approach in which the experimental MS/MS spectra are indexed at runtime, which allows for a single digestion pass through the database in order to assign candidate peptides to all MS/MS spectra in an experiment ^{9, 10}, although other algorithms likely also rely on a similar strategy. As a result of these improvements, similarity scoring has now become the most significant bottleneck for database searching. Similarity scoring generally involves two primary steps: creation of a theoretical MS/MS spectrum from peptide sequences (including potential modifications), and a vector calculation in which the theoretical and experimental spectra are compared. It should be noted that the generation of a fragmentation spectrum from peptide sequences comprises a significant portion of the computational work involved in peptide-spectrum scoring.

Graphical processing units (GPUs) have recently been introduced as alternative platforms for general purpose computing, with considerable potential for advancing scientific research in a fast and economical manner ^{11–13}. A typical desktop GPU contains hundreds of processing cores that are capable of functioning in parallel. However, in contrast to multiple cores on a CPU, the processing cores on a GPU are designed for data parallelism (single instruction, multiple data; SIMD). Thus, GPUs are ideally suited to compute solutions to problems that contain many independent and often repeated calculations. Indeed, peptide spectral matching lends itself to GPU programming in that millions of independent similarity scores are computed in a typical experiment.

Recently, Baumgartner et al. computed spectral comparisons on a GPU ¹⁴. However, their algorithm was restricted to only matrix multiplications required for spectral library matching, and thus contained no protein database digestion or generation of theoretical spectra. While that study did clearly demonstrate the performance gains associated with data parallelism for scoring, the successful implementation of the Smith-Waterman algorithm on GPUs indicates that GPU parallelism is not limited to vector-based calculations alone.

Here, we present a new database searching program, Tempest, that combines efficient database digestion and MS/MS spectral indexing with fast similarity scoring on a GPU. In our implementation, the entire similarity score (including the generation of full theoretical peptide candidate fragmentation spectra and its comparison to experimental spectra), as well as other data processing tasks, are conducted on the GPU. Tempest uses the classical SEQUEST XCorr score as a primary metric for evaluating similarity. When scoring MS/MS spectra collected at high resolution, Tempest uses a new “Accelerated Score” based on a dot product that closely recapitulates the more computationally expensive legacy XCorr without sacrificing score accuracy. In our experience, Tempest provides compute-cluster level performance in an affordable desktop computer.

Materials and Methods

HeLa cell protein preparation

HeLa cell lysate was prepared by harvesting a confluent 15 cm dish of HeLa cells by trypsinization, washing 2x in PBS, and addition of 4 ml lysis buffer (0.5% Triton X-100, 50 mM Tris pH 8.1, 150 mM NaCl, 1 mM MgCl₂ Roche Mini-complete protease inhibitors), followed by sonication and clarification of the lysate in a centrifuge (14,000 × *g*) for 10 minutes at 4 °C. Both protein preps were reduced by addition of DTT to 5 mM and incubation in a water bath at 55 °C for 20 minutes, followed by alkylation of cysteines in 12.5 mM iodoacetamide at room temperature for 45 minutes. After quenching the alkylation reaction (addition of 2.5 mM DTT), the lysates were aliquotted, snap frozen in liquid nitrogen and stored at –80 °C until use. To prepare the protein digests, an aliquot of cell lysate was warmed rapidly under warm water and mixed with SDS-PAGE sample buffer to a protein concentration of ~0.25 mg/ml protein, followed by separation on 2 well, 4 – 12 % NOVEX minigels (for 2D separations) with a protein marker in the narrow lane. Proteins were visualized with Coomassie blue and the region between 80 and 125 kDa was excised, destained, and digested with trypsin. Peptide samples were analyzed on an LTQ Orbitrap (ThermoFisher Scientific, Bremen, Germany) per established procedures¹⁵. Two 240-minute LC-MS/MS runs were conducted on this digest: one with data collection in the LTQ (“unit resolution”, 34,945 total MS/MS spectra – Tranche hash: Be56RwE14zJOrkvS11WBMzValnJTjMgpULIXuxtVJdydrOJK3gCLj/T49pqz0CnA73V8BtaTOH ZIcVSpx0OU8Nzg8AAAAAAAAB4w==) and one with data collection in the Orbitrap (“high resolution”, resolution set to 7,500, 17,652 total MS/MS spectra – Tranche hash: yYWdqPWRk5gI7xvb6h+5Ewtfz18Fy+elfZSWrqcmjK9L2nJ6Crq3mJi2TuKeME6EzdcQuWLUJ2r4fHB94L0qHj2LWrMAAAAAAAAAB6Q==). Precursor ion mass-to-charge assignments were adjusted post acquisition with in-house software by averaging all chromatographic observations of precursor ion mass-to-charge values.

Tempest programming environment

All programs were written using ANSI C and standard libraries. GCC version 4.2.5 was used with the NVIDIA CUDA 3.2 C libraries in a Unix environment. The GPU used in this study was an EVGA NVIDIA “Fermi” GeForce GTX 480 with 480 discrete arithmetic logic units (ALU) distributed on 15 streaming multiprocessors (SM) and 1.5 gigabytes of global memory (\$300 retail). Input is provided as command line arguments and experimental data is read in .DTA file format. Candidate peptides are digested from FASTA-formatted target-decoy¹⁶ protein databases (UniProtKB, www.uniprot.org: yeast – download date 09/2010, 13,234 target plus reversed proteins; human – download date 09/2010, 40,558 target plus reversed proteins; all organisms – download date 06/2011, 1,058,112 target plus reversed proteins). Tempest outputs SEQUEST-like .out files for each input spectrum. Tempest program internals are described in the results section. All testing was performed on a desktop computer with an Intel i7 870 CPU (stock 2.93 GHz) and 16 GB of DDR3-1333 SDRAM running Ubuntu (version 11.04) and the NVIDIA GeForce GTX 480 GPU. Standard search parameters used a 1.1 Dalton precursor peptide mass tolerance, and included up to three missed cleavages per peptide (for fully and partially tryptic searches), static acetamide alkylation of Cys and dynamic oxidation of Met, with up to three dynamic modifications per peptide. For phosphorylation searches, the addition of phosphorylation on Ser, Thr and Tyr with up to three dynamic modifications per modification per peptide was considered. Other precursor peptide mass tolerances and enzyme specificities were employed as described in the text. All run times were collected in triplicate and averaged.

Results and Discussion

The Tempest database searching algorithm was designed to use both the CPU and GPU appropriately and efficiently, while minimizing data transfer between the two (Figure 1). Tempest executes protein digestion serially on the host CPU and computes similarity scores in parallel on the GPU. MS/MS scans are first indexed by precursor mass as previously described in the Macro project⁹, and peak information from the MS/MS spectra is transferred to the GPU. The index is used to quickly select candidate peptides (“candidates”), which are collected in buffers on the CPU for each MS/MS. Whenever a buffer is filled, its candidate sequences are transferred to the GPU and similarity scores are computed in parallel. A list of top scoring candidates and their scores is maintained on the GPU, and once all proteins have been digested and all candidate peptides scored, the top-ranked PSMs are transferred back to the CPU for publishing. In this way, Tempest achieves both task parallelism (by using the CPU for *in silico* protein digestion and the GPU for candidate scoring) and data parallelism (by scoring many candidates at a time against a single MS/MS spectrum).

Tempest provides two algorithms to generate similarity scores: the SEQUEST XCorr for low-resolution MS/MS and an Accelerated XCorr (presented here) for high-resolution MS/MS spectra. Both algorithms are implemented as CUDA kernels and are executed in a SIMD manner on a single GPU. Each kernel launch scores all of the candidate peptides from one buffer against a single observed MS/MS spectrum, with each thread computing the score for one candidate. To produce a SEQUEST XCorr score, a candidate is fragmented *in silico*, and a cross-correlation is used to compare the resulting theoretical MS/MS spectrum to the observed MS/MS spectrum. The cross-correlation can be accomplished by first transforming the observed spectrum and then computing the dot product of the two spectra. Importantly, the expensive transformation step can be factored out so that pre-transformed input spectra are reused for each candidate^{17, 18}. The scoring kernel is responsible for theoretical peptide fragmentation and the dot product. Figure 1A depicts a single scoring kernel launch in Tempest, in which 1024 threads (distributed across 480 processors) score 1024 candidates for a particular MS/MS spectrum in parallel. Each individual thread calculates fragment ions for one peptide and computes the dot product serially. In fact, each thread executes an identical set of instructions for each theoretical fragment. For example, many threads compute the mass of the predicted +2 b_2 fragment ion from their candidate peptide at the same time. This is possible because the calculations for any one candidate are distinct from those of any other candidate. After each scoring kernel, a custom reduction kernel on the GPU is used to find top scoring candidates and compute cumulative summary statistics.

The computations and memory access pattern of each kernel function must be carefully organized in order to fully capitalize on the data parallelism offered by GPU computing. In scoring kernels, Tempest scores candidates for just one observed MS/MS scan in each launch. Because all threads access peak data from the same observed spectrum, kernel performance can benefit from memory caching on each GPU multiprocessor. Furthermore, we note that the number of predicted fragment ions for a given candidate depends on peptide length and precursor charge state, and a thread with more fragments will take longer to complete, resulting in idle threads and wasted GPU processing power. The UniProt human protein database contains fully tryptic peptides that vary in length by over 30 residues. But by scoring candidates for only one MS/MS spectrum at a time, Tempest minimizes the range of peptide lengths and charge states in each kernel launch. Peptides that are scored together are all selected from a window around a single mass: for a given precursor and a 1.1 Da precursor mass tolerance, the average range of candidate peptide lengths is just seven residues.

In addition to scoring kernels, Tempest includes two kernels for processing input MS/MS spectra in parallel on the GPU (Figure 1B). MS/MS peak information from SEQUEST DTAs is loaded and preprocessed (peak normalization and noise filtering) on the CPU and transferred to the GPU as a compact list of peaks in order to minimize data transfer. However, a full sparse data array of intensities must be constructed for the vector calculations of the cross-correlation score. First the full spectrum is built in a single parallel instruction in which each thread writes one peak, and then a second kernel performs the previously mentioned Fast-XCorr transformation in parallel by transforming a small portion of the spectrum in each thread. If there is sufficient space in GPU memory for the full spectra, all of the input MS/MS are built and transformed once before digestion begins. The spectra reside in GPU memory during the full execution of Tempest, poised for scoring when needed as depicted in Figure 1A. At unit-resolution, up to 75 thousand MS/MS fit in 1.5 GB of global GPU memory. For larger datasets Tempest switches to a slower “rebuild mode”, instead storing only the compact peak data. In this mode, the necessary spectrum is built and transformed immediately prior to each scoring kernel launch.

Task Parallelism

In CUDA, all kernel launches are asynchronous; it is appropriate to think of a kernel launch as “queuing” the kernel for execution. CUDA also enables asynchronous data transfer to and from the GPU device. Tempest uses both of these features to achieve concurrent execution of the digestion and scoring algorithms. During digestion, candidate peptides are collected in a buffer for each MS/MS spectrum. When a candidate peptide buffer is filled, the CPU initiates an asynchronous transfer of the candidate data to GPU memory and then launches a scoring kernel. The CPU is then immediately free to continue digesting proteins and collecting peptide candidates, as the transfer and scoring will occur as soon as the GPU is available. This concurrent execution model is in contrast to Macro—and any other database search program that uses serial processors—which must pause digestion to score and rank selected candidates. Figure 2A schematically compares digestion and scoring of candidate peptides on a single CPU in Macro and distributed across a CPU and GPU in Tempest. Digestion is executed serially in both programs and is expected to require approximately the same amount of total time (in white). Tempest’s data-parallel scoring, however, is both more efficient and occurs concurrent to continued digestion, resulting in shorter overall search times.

Each candidate buffer may be filled many times, and new candidates cannot be placed into a buffer until any previous transfer of that buffer has been completed. Tempest uses a double buffer system for each input MS/MS so that the CPU can continue filling one buffer while the second is waiting for transfer; additional buffers were evaluated and were found to be unnecessary. CUDA events are used to manage the double buffer system for each MS/MS spectrum and to query the GPU for progress. Page-locked host memory is used for the buffers to accelerate asynchronous transfers.

Performance: Digestion and Scoring Algorithms

To determine the benefit from Tempest’s GPU-enabled parallelism, we compared the performance of the digestion and scoring algorithms in Tempest to those in Macro using a dataset of unit-resolution HeLa MS/MS spectra. Digestion times were measured by running the programs without scoring. For Macro, the scoring time was calculated as the complete runtime minus the digestion time, while for Tempest we used CUDA events to time scoring kernels on the GPU with high precision. The performance of both algorithms is plotted against the total number of generated scores at increasing numbers of input spectra (Figure 2B). Tempest exhibits a slight amount of additional overhead during digestion due to the buffering of candidate peptides. The scoring algorithm shows a significant performance

improvement when executed in parallel: Tempest scores five thousand spectra (~40 million scores) ten times faster than Macro, requiring only 22 seconds to complete, while Macro takes three minutes and 36 seconds. It is important to note that Macro uses the same XCorr similarity score as Tempest.

Scoring takes far longer than digestion in Macro. However, it has been reduced in Tempest to nearly the same amount of time. At six thousand spectra, Macro spends ten times longer scoring candidates than digesting the database. The parallelization offered by a single GPU reduces this ratio to 1.8, which suggests that Tempest should achieve significant performance gains as a result of concurrent digestion and scoring.

Fine Tuning the Scoring Kernels

Scoring kernels are configured with enough threads to score all of the candidates from one buffer in a single launch. Increasing the buffer size means that more scores are computed in a single kernel, but fewer kernels are required to complete the search. We assessed this balance by testing scoring kernel run time and total (cumulative) scoring time as a function of buffer size (Figure 3). As buffer size increases, the average time for a single scoring kernel increases with a slope that is less than one, which is due to increased parallelism on the GPU. That is, a single kernel configured to score 1,024 candidates takes less than twice the time it takes a kernel configured to score 512, or half as many, candidates. The trend continues up to 2,048 scores per kernel, where the capabilities of our test GPU (a GTX 480) are saturated. Ultimately, total scoring time in Figure 3 (light green circles) shows that scoring is most efficient across an entire search at a buffer size of 2,048 candidates.

On the other hand, increasing the buffer size means increased memory requirements in CPU RAM. Two candidate buffers are used for each input MS/MS spectrum, so the number of MS/MS spectra that Tempest can search in a single run is limited by the amount of RAM and the buffer size. There is a tradeoff between the increased efficiency derived from high buffer sizes and the number of MS/MS spectra that can be searched in a single run. As a result, Tempest uses a buffer size of 1,024 by default, which requires 192 kilobytes of CPU RAM per tandem mass spectrum. In addition, the buffer size is configurable at run time; the buffer size can be reduced for systems with limited RAM or to search more MS/MS spectra in a single run, although it may sometimes be faster to divide very large numbers of MS/MS spectra into two searches at a higher buffer size, in particular if Tempest is also running in “rebuild mode”. At the default buffer size, we can search 80,000 spectra in one run on a machine with 16 GB RAM without rebuilding.

Performance: Concurrent Digestion and Scoring

To evaluate the performance of a complete Tempest search, we used the CUDA Event API to record the absolute total run time, the digestion and non-digestion times on the CPU, and the transfer and scoring times on the GPU. A key factor in run time performance is GPU-CPU concurrency, and so we calculated concurrent execution time by summing the isolated processing times and subtracting the total search time:

$$t_{concurrent} = (t_{cpu} + t_{gpu}) - t_{total}$$

In addition, we also calculated GPU idle time, CPU idle time, the ratio of scoring time and digestion time, and the relative overhead of data transfer. Figure 4A provides a holistic breakdown of three common searches by task and device and depicts GPU-CPU concurrency during each search. While not perfect, Tempest’s division of labor is effective and scalable in a range of database searching scenarios. In unit-resolution searches using the

SEQUEST XCorr, scoring time is always greater than digestion time, and the ratio is typically three and is always less than four. In particular, note that the ratio is similar for the 10 ppm, 1.1 Dalton, and 10 Dalton searches, even though total run time varies as a function of mass tolerance. Data transfer accounts for just 2–3% of the total GPU run time. A full listing of detailed run times and concurrency measurements can be found in Supplementary Table 1 (Supporting Information).

In most Tempest searches of unit resolution spectra, the GPU is active for over 90% of the total run time. Besides non-digestion CPU tasks such as loading MS/MS data and publishing results, the GPU is only idle when all launched scoring kernels have been completed and the CPU has not yet filled any new buffers. This GPU idle time (during digestion) is just 3% or less in most Tempest searches. In addition to GPU idle time, there is CPU idle time in each unit-resolution search because the scoring time is greater than the digestion time. Depicted in Figure 4A as blank space above the CPU work, CPU idle time occurs when digestion must wait for a queued buffer transfer before buffering the next peptide candidate. The result is concurrent execution for 20–40% of the total run time. We note that when there is both CPU and GPU idle time, the unrealized potential concurrency is the minimum of the two idle times. With the exception of 10 ppm and small database searches, unrealized concurrency during digestion is under 2% of the total run time. We note that searches with low concurrency are fast searches with very low digestion and scoring times. For example, in a 10 ppm search of 10,000 unit resolution spectra, the GPU is idle for a high percentage of digestion because no buffers are completely filled, but the total run time is only 16 seconds with just two seconds of unrealized concurrency.

The run time behavior of a search depends on the rate at which candidate buffers are filled (i.e. the rate at which candidate peptides are produced from the database) and the speed of the scoring kernels. To explore this behavior, Tempest was altered to record the time of each scoring kernel launch, and we used the kernel launch times and the average kernel run time to simulate the queue depth over the lifetime of the search. Figure 4B shows the queuing and scoring behavior for three of the search types in 4A for 1,000 of the MS/MS spectra in Figure 4A. In each search, queue depth is rarely if ever zero once digestion starts, meaning that the GPU always has candidates to score. There is a significant spike after completing digestion (red line) as partially filled buffers are all queued at once. While the queue depth in the standard 1.1 Da search remains fairly level throughout digestion, in the 10 Da search the depth gradually rises to nearly 800 by the end of digestion because searching with a larger precursor tolerance significantly increases the rate of peptide candidate production without altering the kernel run time. In fact, the 10 Da search produces candidates so quickly that digestion must pause while scoring completes, resulting in oscillations in the queue depth throughout the search (Figure 4B, middle). On the other hand, the phosphorylation search requires more calculations to produce modified candidates and produces candidates less quickly; the average queue depth drops from 175 in the standard search to 130 when allowing for dynamic phosphorylation on Ser, Thr and Tyr. Dips in the queue depth are intrinsic to the database used; for example, the N-terminal decoy peptide ATLSVDSGQASDSSAAQSYSGGKGGSSK is repeated in 23 proteins in a row in the reversed portion of the Swiss-Prot human FASTA database and causes the most significant dip at 13.88 seconds. This sequence requires significant computation on the CPU for considering the combinatorial expansion of constitutional isomers, but does not produce any candidates peptide matches in our dataset. Importantly, Tempest's buffering and queuing of candidates maintains GPU activity despite such lulls in candidate production.

Unit-Resolution Performance

Tempest was designed to be a scalable, flexible, and practical database searching algorithm that can be used for the majority of real-world experiments. Table 1 summarizes Tempest's

run times in comparison to Macro on a dataset of 30,000 unit-resolution MS/MS spectra under a variety of commonly-encountered search scenarios in proteomics applications. Tempest completes a standard human database search in 2 minutes 48 seconds and a taxing nonspecific enzyme search with 300 million peptides and over 20 billion separate scores in less than three and a half hours. Search runtime improves by up to 15 times as a result of Tempest's GPU parallelism. Importantly, note that Tempest can efficiently search databases of any size and can handle extremely large input data sets of 100,000 MS/MS spectra or more in a single run. The program also supports user-defined enzymes with full or partial specificity at the peptide termini, user-defined PTMs, and configurable precursor mass tolerance.

Accelerated High Resolution Scoring

The classic SEQUEST cross-correlation score (XCorr) was developed originally to improve spectral match accuracy over a simple dot product by correcting for background correlation³. This allows for improved discrimination of true peptide fragmentation spectra from spectra that were full of chemical "noise" instead of peptide ion fragments. Given the historical context of mass spectrometry instrumentation during the development and implementation of SEQUEST, this aspect of the XCorr was a key component of the success of the algorithm, in particular for assigning PSMs to MS/MS spectra collected at unit resolution.

Advancements in mass spectrometry instrumentation have now brought high resolution tandem mass spectrometry closer to many proteomics laboratories^{19, 20}. We previously developed an improved method for processing high-resolution MS/MS spectra using the classic SEQUEST XCorr that conserves increases in mass accuracy by representing the spectra at ~40 data points per Thompson, or a bin width 0.025 m/z⁹. As the "resolution" of spectral processing increases, the observed MS/MS spectra become more sparse and the likelihood of observed peaks randomly matching the theoretical MS/MS spectrum decreases, yielding increased discrimination of correct peptide sequences against other candidates²¹. However, we also realized that this increased sparseness should decrease the reliance of the XCorr on the correction offered by the cross-correlation function, given the decreased probability of spectral fragments contributing to background correlation. We hypothesized that as sparseness increases due to increased resolution, the XCorr score should approach a simple dot product.

In the classic SEQUEST XCorr score at low resolution, it was necessary to correct for cases in which many theoretical fragment ions resolve to the same m/z bin. With sparse experimental MS/MS spectra, any theoretical ion has a decreased likelihood of matching to a nonzero peak in the experimental spectrum. We surmised that the work required to store and check the theoretical spectrum as every new ion is generated, which was essential at unit resolution, could be removed at high resolution because high resolution MS/MS spectra without the cross-correlation correction would be sufficiently sparse.

To test these hypotheses, we combined these two features into a new "Accelerated Score" for high-resolution MS/MS spectra in order to recapitulate the SEQUEST XCorr with limited loss in productivity. This Accelerated Score uses a simple dot product, which replaces the expensive cross-correlation calculation in the XCorr for spectral comparison. To measure the degree of correlation between the legacy SEQUEST XCorr and the Accelerated Score, 30,000 unit-resolution MS/MS spectra and 15,000 high-resolution MS/MS spectra were searched with Tempest at ~1 bin per Thompson ("unit" resolution) and at ~40 bins per Thompson ("high" resolution), respectively, using both the classic XCorr and Accelerated Scores. The scores were then plotted against each other for each spectrum with matching PSMs (where both algorithms agreed on the top-ranked peptide) (Figure 5A).

While the scores differ significantly at low resolution, Tempest's simplified Accelerated Score correlates very well with the SEQUEST XCorr at high resolution. The scores agree poorly in the unit-resolution dataset (black), where the XCorr cross-correlation appropriately normalizes noisy spectra but the dot product in the accelerated algorithm produces inflated scores. However, when searched at high resolution (red), the two scores are in near perfect agreement with a correlation coefficient (R^2) of 0.9982. We note that a few of the Accelerated Scores also suffer from score inflation, likely for the same reason.

Spectra receiving mismatching PSMs between the two scores account for 36% of the unit-resolution results but only 3% of the high-resolution results (Supplementary Figure 1, Supporting Information). To further investigate the qualitative differences between the two scores, we generated receiver operator curves (ROC) for the two scores for searches performed at unit and high resolution (Figure 5B). We observed that the XCorr score outperformed the dot-product based Accelerated score at unit resolution, but was indistinguishable from the Accelerated Score when high resolution spectra were analyzed. We note here that while the XCorr score produced 15,293 PSMs at a 1% FDR for the unit resolution search, the Accelerated Score produced only 7,576 PSMs at the same level of precision, or a 50% reduction in sensitivity. However, when searching high resolution spectra, the XCorr produced 10,662 PSMs while the Accelerated Score produced 10,660 PSMs at a 1% FDR. Clearly, SEQUEST's cross-correlation scorer is more accurate for spectra collected at unit resolution, although this advantage disappears at high resolution.

To determine the point at which the Accelerated Score accurately recapitulates the XCorr, we measured the correlation between the two scores as function of resolution (bins per Thompson) by searching the 15,000 high resolution spectra using both scores at varying bin widths (Supplemental Figure 2, Supporting Information). The R^2 value approaches 1.0 as the processing resolution increases; the R^2 value was > 0.99 at 10 bins per Thompson (a bin width of 0.1 m/z) and above. In Tempest, the decision of whether to use the Accelerated Score is left to the user, but we suggest that spectra collected with at least ± 0.05 m/z precision can be scored using the Accelerated Score in conjunction with appropriately small bin widths.

The Accelerated Score is much faster to calculate than the SEQUEST XCorr and can be used to significantly improve the efficiency of Tempest searches (Figure 6A). Because the Accelerated Score uses a dot product instead of a cross-correlation to compare spectra, Tempest can simply skip the transformation step, which is much more costly when using high resolution spectra, and use the untransformed fragmentation spectra directly. Furthermore, fully built high-resolution spectra require ~40 times more memory on the GPU than unit-resolution spectra, so at high resolution Tempest is often forced to run in its rebuild mode in which spectral processing (Figure 1B) recurs prior to each scoring kernel launch. A search using the Accelerated Score can avoid many expensive spectral transformations per input MS/MS spectrum. (See Supplementary Table 2 for comparison of Tempest run times in normal search mode and in rebuild mode, in Supporting Information). In addition, the Accelerated Score algorithm generates fragment ions transiently, simply allowing ions that resolve to the same m/z bin to accumulate with a negligible effect on the final score and a considerable improvement in speed. Because the algorithm does not check for previously predicted peaks, it does not need to store the theoretical spectrum in memory at all, and slow global memory access is dramatically reduced. In each thread, the Accelerated algorithm cuts the number of global reads by half and, more importantly, eradicates all global writes other than the final score. Furthermore, 1,024 high resolution theoretical spectra require 820 MB of GPU memory, over half of the available memory on a GTX 480. When using the Accelerated Score, these spectra become unnecessary, and twice as many observed MS/MS spectra can be stored on the GPU before Tempest must switch to

rebuild mode. Table 2 summarizes the run time performance of searches using the XCorr and Accelerated scores for high resolution MS/MS. Total runtime improvement varies based on search type, but the total GPU scoring time consistently improves around 130 fold using the Accelerated Score.

The speed of the Accelerated Score kernel alters the balance of work between the CPU and GPU (Figure 6B). The ratio of scoring time to digestion time when using the Accelerated Score is below 1, in comparison to 150 in high resolution XCorr searches and 3 or 4 in unit resolution XCorr searches. In searches using the Accelerated score, the CPU does not fill candidate buffers quickly enough to keep the GPU active during digestion, resulting in an increase in GPU idle time. Unlike unit resolution and high resolution searches using the XCorr, the CPU idle time in searches using the Accelerated score is negligible, and GPU idle time is always greater than CPU idle time. However, the unrealized potential concurrency during digestion is still under 2% of the total search time. Detailed run time data can be found in Supplementary Table 1 (Supporting Information).

In summary, by combining all of these features of Tempest together, we note that high resolution MS/MS spectra can be scored with the Accelerated Score in less time than unit resolution spectra are scored using the legacy SEQUEST XCorr. This relationship is also true when searching with dynamic post-translational modifications, such as phosphorylation (Figure 6C). Here, 15,000 high resolution MS/MS spectra can be searched for phosphorylation on S, T, or Y in less than half the time that the same number of unit resolution spectra can be searched using XCorr.

Conclusions

The desktop computer we used for this study cost \$2,500, which includes the NVIDIA GPU. With it, Tempest was able to search large datasets collected at high and low resolution very quickly. We searched a dataset of 100,000 MS/MS spectra collected on an LTQ Orbitrap Velos instrument against peptides digested in real time from the UniProt human database (forward and reverse) in 16 minutes. Using the Accelerated Score, Tempest searched the same dataset in 6 minutes. Clearly, Tempest is capable of handling datasets typical of large-scale proteomics experiments.

We anticipate that other database matching algorithms could also benefit from GPU-based acceleration, using the Tempest program design as a framework. Tempest's scoring kernels are modular, which allows for support of different fragmentation methods (e.g. ETD, ECD), as well as different spectral comparison algorithms. We also expect that our reporting of how our Accelerated Score functions to improve efficiency for high resolution scoring is likely to improve other database matching algorithms that operate with or without GPU acceleration that work on datasets that contain fragmentation spectra collected at high mass precision. In addition, in order to facilitate access to Tempest for the greater proteomics community, we are currently working with ThermoFisher Scientific to place a Tempest node in their ProteomeDiscoverer framework. Tempest allows for digestion of databases of any size using conventional and user-defined proteases, as well as partial and fully non-specific protease cleavages, at variable precursor and fragment ion mass tolerances, and supports all primary post-translational modifications as both static and dynamic masses.

In this study, we compared the performance of Macro (running on a single CPU core) to Tempest (running on a single CPU core and a GPU). In practice, many database search algorithms are written specifically to take advantage of multiple processing cores on a single chip. For example, the Intel i7 870 CPU used in the present work has four independent processing cores available for use. Even in the absence of deliberate multithreading, it is

possible to divide the number of MS/MS spectra in an analysis by the number of available cores and launch multiple independent Macro processes simultaneously on a multicore CPU. Other algorithms, such as OMSSA, are written to employ multithreading as an optional input parameter at launch. We note that the practical "real-world" utilization of all available cores in either of these modes of operation in a quad-core i7 CPU reduces the apparent performance benefit of Tempest considerably (Supplemental Figure 3, Supporting Information). However, comparisons between such different platforms are always "apples and oranges" – in the example above, this version of Tempest leaves three additional cores on the CPU that are idle that could also be performing additional work, either to support a faster, multithreaded version of Tempest, or to perform other functions, such as webserver support or database management that might be required in a production environment in a working proteomics lab or core facility.

In general, unit resolution scoring still takes as much as three times longer than database digestion, which suggests that increased parallelism through the use of additional CPU cores or GPUs could further improve performance. With the release of CUDA 4.0 in May, 2011, a single program can now access multiple GPUs. We anticipate that with three GPUs on a single platform, unit resolution scoring could be folded entirely within the time of digestion. We note that for high resolution scoring, the Accelerated Score already reduces the scoring time to a fraction of the digestion time. Importantly, CUDA 4.0 allows multiple CPUs to access a GPU, which opens the door to multithreaded digestion with GPU parallel scoring. Future versions of Tempest are planned that take advantage of these and other advancements in technology.

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

Acknowledgments

We thank Mike Senko, Justin Blethrow, and colleagues at ThermoFisher Scientific for their support and general discussions. We thank Arminja Kettenbach for preparing the HeLa cell lysate digest and Jason Gilmore for critical insights and discussions. This work was supported by the National Institutes of Health grant P20-RR018787 from the IDeA Program of the National Center for Research Resources and grant R01-CA155260 from the National Cancer Institute (to S.A.G.) and a predoctoral fellowship from the National Institute of General Medical Sciences Grant T32-GM008704 (to B.K.F.).

References

1. Craig R, Beavis RC. A method for reducing the time required to match protein sequences with tandem mass spectra. *Rapid Commun Mass Spectrom*. 2003; 17(20):2310–2316. [PubMed: 14558131]
2. Craig R, Beavis RC. TANDEM: matching proteins with tandem mass spectra. *Bioinformatics*. 2004; 20(9):1466–1467. [PubMed: 14976030]
3. Eng JK, McCormack AL, Yates JR III. An approach to correlate tandem mass spectral data of peptides with amino acid sequences in a protein database. *Journal of the American Society for Mass Spectrometry*. 1994; 5(11):976–989.
4. Geer LY, Markey SP, Kowalak JA, Wagner L, Xu M, Maynard DM, Yang X, Shi W, Bryant SH. Open mass spectrometry search algorithm. *J Proteome Res*. 2004; 3(5):958–964. [PubMed: 15473683]
5. Perkins DN, Pappin DJ, Creasy DM, Cottrell JS. Probability-based protein identification by searching sequence databases using mass spectrometry data. *Electrophoresis*. 1999; 20(18):3551–3567. [PubMed: 10612281]

6. Nesvizhskii AI. A survey of computational methods and error rate estimation procedures for peptide and protein identification in shotgun proteomics. *J Proteomics*. 2010; 73(11):2092–2123. [PubMed: 20816881]
7. Sadygov RG, Eng J, Durr E, Saraf A, McDonald H, MacCoss MJ, Yates JR. Code developments to improve the efficiency of automated MS/MS spectra interpretation. *Journal of Proteome Research*. 2002; 1(3):211–215. [PubMed: 12645897]
8. Park CY, Klammer AA, Kall L, MacCoss MJ, Noble WS. Rapid and accurate peptide identification from tandem mass spectra. *J Proteome Res*. 2008; 7(7):3022–3027. [PubMed: 18505281]
9. Faherty BK, Gerber SA. MacroSEQUEST: efficient candidate-centric searching and high-resolution correlation analysis for large-scale proteomics data sets. *Anal Chem*. 2010; 82(16):6821–6829. [PubMed: 20684545]
10. Tabb DL, Narasimhan C, Strader MB, Hettich RL. DBDigger: reorganized proteomic database identification that improves flexibility and speed. *Anal Chem*. 2005; 77(8):2464–2474. [PubMed: 15828782]
11. Liu YC, Schmidt B, Liu WG, Maskell DL. CUDA-MEME: Accelerating motif discovery in biological sequences using CUDA-enabled graphics processing units. *Pattern Recogn Lett*. 2010; 31(14):2170–2177.
12. Manavski SA, Valle G. CUDA compatible GPU cards as efficient hardware accelerators for Smith-Waterman sequence alignment. *BMC Bioinformatics*. 2008; 9(Suppl 2):S10. [PubMed: 18387198]
13. Schatz MC, Trapnell C, Delcher AL, Varshney A. High-throughput sequence alignment using Graphics Processing Units. *BMC Bioinformatics*. 2007; 8:474. [PubMed: 18070356]
14. Baumgardner LA, Shanmugam AK, Lam H, Eng JK, Martin DB. Fast parallel tandem mass spectral library searching using GPU hardware acceleration. *J Proteome Res*. 2011; 10(6):2882–2888. [PubMed: 21545112]
15. Haas W, Faherty BK, Gerber SA, Elias JE, Beausoleil SA, Bakalarski CE, Li X, Villen J, Gygi SP. Optimization and use of peptide mass measurement accuracy in shotgun proteomics. *Mol Cell Proteomics*. 2006; 5(7):1326–1337. [PubMed: 16635985]
16. Elias JE, Gygi SP. Target-decoy search strategy for increased confidence in large-scale protein identifications by mass spectrometry. *Nat Methods*. 2007; 4(3):207–214. [PubMed: 17327847]
17. Venable JD, Xu T, Cociorva D, Yates JR 3rd. Cross-correlation algorithm for calculation of peptide molecular weight from tandem mass spectra. *Anal Chem*. 2006; 78(6):1921–1929. [PubMed: 16536429]
18. Eng JK, Fischer B, Grossmann J, MacCoss MJ. A fast SEQUEST cross correlation algorithm. *J Proteome Res*. 2008; 7(10):4598–4602. [PubMed: 18774840]
19. Makarov A, Denisov E, Kholomeev A, Balschun W, Lange O, Strupat K, Horning S. Performance evaluation of a hybrid linear ion trap/orbitrap mass spectrometer. *Anal Chem*. 2006; 78(7):2113–2120. [PubMed: 16579588]
20. Michalski A, Damoc E, Hauschild JP, Lange O, Wiegand A, Makarov A, Nagaraj N, Cox J, Mann M, Horning S. Mass spectrometry-based proteomics using Q Exactive, a high-performance benchtop quadrupole Orbitrap mass spectrometer. *Mol Cell Proteomics*. 2011
21. Mann M, Kelleher NL. Precision proteomics: the case for high resolution and high mass accuracy. *Proc Natl Acad Sci U S A*. 2008; 105(47):18132–18138. [PubMed: 18818311]

Figure 1. The Tempest program

A) Schematic of the Tempest database searching algorithm. *In silico* protein digestion occurs on the CPU, where digested peptides are matched to indexed MS/MS scans and added to candidate peptide buffers. When an MS/MS scan's buffer is full (in this example, the buffer for index 400 has reached 1,024 candidates and is ready for scoring, while the others remain incomplete), the candidate peptides are transferred to the GPU for scoring without interrupting digestion. On the GPU, each candidate is fragmented and scored against the MS/MS spectrum at the index 400 in parallel, and the topscoring candidate (LSSDVLTLIIK) is determined using a custom reduction kernel. The overall top-scoring

PSM for each MS/MS spectrum is stored on the GPU and must be updated after each scoring kernel. **B)** Schematic of Tempest's MS/MS spectral data processing. Experimental MS/MS spectra are transferred to the GPU once prior to digestion as a compact list of peaks. For each spectrum, the full, sparse data array is first built in a single parallel step and then transformed in parallel in preparation for subsequent Fast XCorr calculations. If possible, the full transformed data arrays are stored on the GPU for the duration of the search (as depicted in **A**).

Figure 2. Task and data parallelism

A) Depiction of digestion and scoring behavior in Macro compared to the same work done in Tempest. The total digestion time in Macro and Tempest is expected to be the same, but total scoring time should be decreased in Tempest due to data parallelism. In addition, Macro must switch back and forth between digestion and scoring, while Tempest is able to perform both concurrently. **B)** Search times for Macro and Tempest scoring a dataset of 5,000 HeLa MS/MS spectra. Compared to Macro, Tempest suffers from slight additional overhead while digesting, but fragments and scores candidate peptides much faster. Tempest is able to produce 40 million scores in 22 seconds while Macro does the same work in 3 minutes and 36 seconds.

Figure 3. Scoring kernel performance

The total scoring time (light green circles) for Tempest to search 1,000 MS/MS spectra from the HeLa dataset is plotted as a function of buffer size (number of scores per kernel). As the size of a buffer increases, fewer kernel launches are required to score the 1,000 spectra. The average time for each kernel to score a buffer as a function of buffer size (dark green squares) is plotted on the second axis. Note that this GPU becomes saturated at 2,048 candidates per buffer.

Figure 4. CPU and GPU concurrency

A) Breakdown of cumulative time data for database searching tasks in Tempest. 10,000 MS/MS spectra from the HeLa dataset were searched either at 10 ppm, 1.1 Da. or a 10 Da. precursor mass tolerance, or at a 1.1 Da. precursor mass tolerance and with dynamic phosphorylation on S, T, and Y. The time spent by the CPU in non-digestion tasks (initialization and publication) is shown in dark green, while digestion is shown in light green. GPU fragmentation and scoring time is shown in blue. The total search time is the top of the GPU time. The ratio of GPU to CPU time is displayed above the column data. **B)** Number of queued scoring kernels over the runtime for 1,000 MS/MS spectra for the three

search types in **A** that queue buffers during digestion. Kernel launch times and the average kernel run times were used to simulate queue depth throughout a search. The vertical red line marks the end of CPU digestion after which any partially filled buffers are rapidly queued for scoring.

Figure 5. Accuracy of legacy XCorr and Tempest's Accelerated Score at unit and high resolution
A) Correlation plots of legacy XCorr and Tempest's Accelerated score. HeLa lysate datasets of 30,000 spectra collected at unit resolution (LTQ), and 15,000 spectra collected at high resolution (Orbitrap; $R = 15,000$) were searched using the legacy SEQUEST XCorr kernel (cross correlation score) and Tempest's Accelerated Scoring kernel (dot product). Spectra for which the top-ranked PSMs were identical between the two searches were evaluated for the agreement between the two scores (for mismatched PSMs, see Supplementary Figure 1). Note the dramatic increase in numerical agreement between the two scores at high resolution. **B)** Score accuracy as a function of search resolution. The results of the searches

performed in **A** were analyzed and receiver operator curves (ROC) were plotted for each MS/MS search resolution. Data from the searches using legacy XCorr are in orange while the searches using the Accelerated score are dashed blue. Note that the XCorr score dramatically outperforms the dot-product based Accelerated score at unit resolution, but that the two scoring methods are indistinguishable when analyzing spectra at high resolution.

Figure 6. High resolution runtime performance

A) The performance benefit of the Accelerated Score. Subsets of the high resolution dataset of 15,000 MS/MS spectra were searched with both the SEQUEST XCorr kernel and the Accelerated Scoring kernel, and the total search time was plotted as a function of the number of MS/MS spectra searched. The Accelerated Score takes just 52 seconds to score 15,000 high resolution spectra, while the XCorr takes 54 minutes. **B)** A breakdown of cumulative time data when searching the 10,000 spectra dataset used in A with the XCorr Score and the Accelerated Score. The searches differ only in scoring time on the GPU; digestion and non-digestion execution times are the same regardless of scoring algorithm

used. The ratio of scoring time to digestion time, shown above each plot, is reduced to ~1 using the Accelerated Score. C) Tempest is able to score spectra at high resolution with the Accelerated score more efficiently than at low resolution with legacy XCorr. The high resolution dataset of 15,000 MS/MS spectra were searched in 2 minutes 10 seconds at high resolution with the Accelerated Score, while searching the same dataset took 7 minutes 36 seconds at unit resolution with the legacy XCorr.

Unit resolution searches typical of proteomics experiments.

30,000 unit resolution MS/MS were collected from HeLa lysate and used to assess the benefit of Tempest in comparison to Macro in both “real world” and artificially extreme use cases. In general, as the total number of scores required increases, the overall benefit of data parallelization via GPU scoring increases.

Database	Precursor Tolerance	Enzyme	Additional Modifications *	Digested Peptides	Scoring Kernels	Tempest Runtime	Macro Runtime	Runtime Improvement
Human	1.1 Dalton	Trypsin		11 million	247 thousand	2.8 minutes	23.2 minutes	8×
Human	1.1 Dalton	Trypsin	Phosphorylation	124 million	1 million	10.8 minutes	2.7 hours	15×
Human	1.1 Dalton	Non-Specific		311 million	20 million	3.4 hours	35.7 hours	15×
Human	40 PPM	Trypsin		11 million	35 thousand	1.4 minutes	5.6 minutes	4×
Human	40 PPM	Trypsin	Phosphorylation	124 million	107 thousand	4.7 minutes	33.4 minutes	7×
Human	40 PPM	Non-Specific		311 million	2 million	31.0 minutes	8.0 hours	15×
Human	3.0 Dalton	Trypsin		11 million	315 thousand	5.1 minutes	1.0 hours	12×
Yeast	1.1 Dalton	Trypsin		3 million	82 thousand	1.2 minutes	6.7 minutes	5×

* All searches were run with reduced and alkylated cysteines (static modification) and oxidized methionine (dynamic modification), with up to three dynamic modifications per modification per peptide. The phosphorylation search was run with the addition of dynamic modifications for phosphorylated serines, threonines and tyrosines.

High resolution searching using the accelerated score

15,000 high resolution MS/MS were collected from HeLa lysate and searched at a 1.1 Dalton precursor ion tolerance with Tempest using both the SEQUEST XCorr score and the Accelerated Score.

Table 2

Database	Additional Modifications*	Tryptic Peptides	Scoring Kernels	XCorr Runtime	Accelerated Runtime	Runtime Improvement	Scoring Improvement
Human		11 million	124 thousand	53.7 minutes	52 seconds	62×	134×
Human	Phosphorylation	121 million	326 thousand	3.9 hours	2.2 minutes	106×	125×