High performance sequence mining using pairwise statistical significance

Yuhong ZHANG [a,1], Feng CHEN [a]
[a] *College of Information Science and Engineering, Henan University of Technology, Zhengzhou, P.R. China*

Abstract. With the amount of sequence data deluge as a result of next generation sequencing, there comes a need to leverage the large-scale biological sequence data. Therefore, the role of high performance computational methods to mining interesting information solely from these sequence data becomes increasingly important. Almost everything in bioinformatics counts on the inter-relationship between sequences, structure and function. Although pairwise statistical significance (PSS) has been found to be capable of accurately mining related sequences (homologs), its estimation is both computationally and data intensive. To keep it from being a performance bottleneck, high performance computation (HPC) approaches are used for accelerating the computation. In this chapter, we first present the algorithm of pairwise statistical significance, then highlights the use of such HPC approaches in acceleration of estimation of pairwise statistical significance using multi-core CPU, many-core GPU, respectively, which both enable significant improvement of accelerating pairwise statistical significance estimation (PSSE).

Keywords. pairwise statistical significance, sequence mining, high performance computation (HPC), multi-core CPU, many-core GPU

1. Introduction

The recently years have witnessed the emergence of new sequencing platforms, higher accuracy, more reads, highlighting longer reads, and cheaper genomes [1], which results in a dramatically increasing trend in the quantity and variety of publicly available genomic and proteomic sequence data. As of October 2012, for instance, there are approximately 145,430,961,262 bases from 157,889,737 reported sequences in GenBank [2]. Furthermore, the number of bases in GenBank still continues to grow at an exponential rate, doubling in size every 18 months since 1982 (Moore's Law at work again). Worldwide sequencing capacities currently exceed 15 petabases per year [3]. "Big data" sets demand more computing power. Under the scenario of unprecedented flood of data generated by the next generation of DNA sequencers, how to handle these data, make sense of them, and enable them accessible to biologists working on a wide diversity of problems places unique challenges upon bioinformatics [4].

Data mining aims to recognize patterns and construct the relationships from a raw data and transform it into a comprehensible form for further use. Especially, the "raw

[1] Corresponding Author: College of Information Science and Engineering, Henan University of Technology, Zhengzhou, P.R. China, 450001; E-mail:zyh803@gmail.com

data" in bioinformatics refers to genomic sequence. There exist many complex data mining tasks, which often cannot be handled directly by traditional data mining algorithms. In bioinformatics, data handling is now becoming a bottleneck of knowledge discovery, which costs more to analyze a genome than to sequence a genome. In fact, genomic data mining and knowledge extraction for biological problems has become one of top 10 challenging problems in data mining research [5].

Almost everything in bioinformatics depends upon mining the inter-relationship between sequence, structure and function (all encapsulated in the term *relatedness*), which is far from being well understood [6].

The recently proposed pairwise statistical significance (PSS) has been shown to be a promising alternative to the database statistical significance (DSS) for the purpose of identifying homologs [7–9]. Although shown to be accurate, pairwise statistical significance estimation (PSSE) is both computationally and data intensive. An unscalable implementation will become a performance bottleneck to be practically useful for many large-scale applications. Thus, use of high-performance computing (HPC) techniques is highly conducive to accelerate the computation of PSSE.

Since the multi-core computers or laptops and clusters have become increasingly ubiquitous and more powerful, it is of interest to use high performance technologies to unlock the potential of computers or laptops and clusters. At the same time, with many-core Graphics Processing Unit (GPU) becoming increasingly powerful, inexpensive, and relatively easy to program, it has become a very attractive hardware acceleration platform. Those strongly motivate the use of multi-core CPUs [10, 11], many-core GPUs [12, 13] or FPGAs [14] to accelerate the PSSE.

Based on the above observation and motivation, in this chapter, we highlight to present OpenMP, MPI and their hybrid, and GPU implementations to accelerate the PSSE, which help. After careful performance analysis, we have efficiently distributed the compute-intensive kernels of the algorithm across processors (cores), so as to reap the maximum benefits of OpenMP or/and MPI paradigms, CUDA as well.

2. Background

2.1. Why Statistical Significance?

Pairwise sequence alignment (PSA) forms a crux of both DNA and protein sequence comparison techniques. It in turn builds the basis of many other applications in bioinformatics, such as database search, finding protein function, protein structure, phylogenetic analysis, etc, for making various high level inferences about the DNA and protein sequences. PSA is one of the most widely used methodology that tries to mine such *relatedness* hidden from proteomic and genomic data. Many bioinformatics applications have been developed on the basis of PSA, such as BLAST [15], PSI-BLAST [16], and FASTA [17].

Biologists usually use pairwise alignment programs to identify similar, or more specifically, related sequences or homologs (having common ancestor). A typical pairwise alignment program aligns two sequences and constructs an alignment with maximum similarity score. PSA produces a score for an alignment as a measure of the similarity between two sequences. In general, the higher the score, the more related the

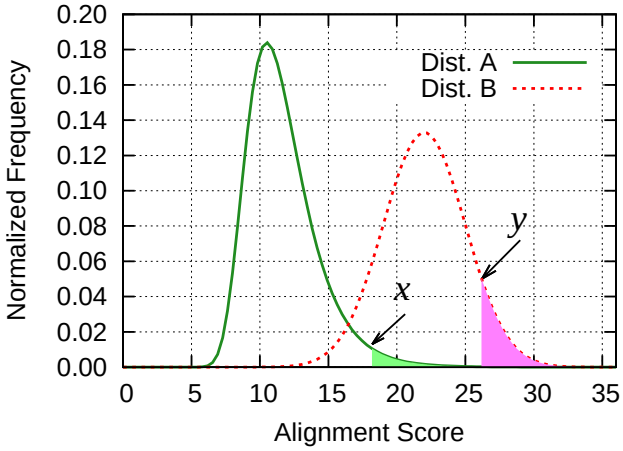


Figure 1. Two different distributions of alignment score

sequences. However, the alignment score counts on various factors such as alignment methods, sequence lengths, sequence compositions and scoring schemes, and so forth [18]. Therefore, estimating the relationship between two sequences solely depended on the scores may lead to a bias conclusion.

To determine whether the resulting similarity (represented by score) between two sequences implies an evolutionary link or not, one has to know how probable it is that we could have obtained the same score by aligning completely unrelated (e.g., randomly chosen) sequences [19]. Therefore, statistical approaches are often used to address the problem. The task related to estimating the significance of an alignment score, in essence, is estimating data reliability and how well the data support a specific hypothesis or prediction. Fig. 1 shows that two alignment score distributions (probability density functions) "Dist. A" and "Dist. B". Assuming that scores x and y are in the score distributions A and B, respectively. Obviously, $x < y$, but x is more statistically significant than y, since x lies more in the right tail of its distribution, meaning that it is less probable to have occurred by chance. The shaded region in this figure presents the probability which a score equal or higher could have been obtained by chance.

Statistical significance represents the likelihood that the resulting similarities in terms of alignment score between two sequences. It is very useful to judge whether the similarity is a functional or evolutionary link, or a chance event [18]. As a result, accurate estimation of statistical significance of sequence alignment has attracted a lot of investigation in recent years [8, 9, 12, 20–32].

The statistical significance of the resulting sequence alignment score between the sequences can be presented by the probability (i.e., *P-value*) that random or unrelated sequences could be aligned to generate the same score. The term "*P-value*" here represents the probability of an alignment with this score or higher occurring by chance alone. If an alignment score has a low probability of occurring by chance, the alignment is considered statistically significant, and hence potentially biologically significant.

These statistical approaches can be based upon theoretical models or upon permutation reconstructions of the observed sequences [33]. However, it is only possible when

the distribution of the alignment scores is known precisely, which can be obtained by aligning many pairs of random sequences.

2.2. Pairwise Statistical Significance

When two sequences are aligned optimally, the significance of a local alignment score can be tested on the grounds of the distribution of scores expected by aligning two *random* sequences of the same length and composition as the two test sequences [33–35].

The optimal alignment scores of random sequences, assessed from high scoring segments pairs, follow a Gumbel distribution (a type I extreme value distribution, EVD [18, 19, 33]). The EVD, to some extent, is like a normal distribution but with a positively skewed tail in the higher score range. It is a limiting distribution for the maximum or the minimum of a large collection of random observations from the same arbitrary distribution. In particular, herein the maximum observations refer to the optimal alignment scores, whose distribution can be deduced not only from reliability theory [36], but from Karlin-Altschul model [33].

The remarkable Karlin-Altschul model interprets the number of highest scoring matching regions above a threshold by a Poisson distribution. Briefly, considering the query sequence $Seq_1 = a_1 a_2 \cdots a_m$ and the subject sequence $Seq_2 = b_1 b_2 \cdots b_n$, where $m = |Seq_1|$ and $n = |Seq_2|$ are their lengths, respectively, the scoring scheme SC (substitution matrix, gap opening penalty, and gap extension penalty, etc.,), the expected number of distinct local alignments with score values of at least x is approximately Poisson distributed with mean

$$E(x) \approx Kmne^{-\lambda x}\,, \tag{1}$$

where λ and K are calculational constants depending on the scoring scheme and average compositions of sequences based on the Poisson distribution hypothesis [33, 37].

Based on Gumbel distribution, the probability of finding an ungapped segment pair with a score lower than x, can be defined by:

$$P(S(Seq_1, Seq_2) < x) = e^{-Kmne^{-\lambda x}}\,. \tag{2}$$

Correspondingly, *P-value* represents the probability of observing at least one score S greater than or equal to a score x in the set of alignment scores, which is given by:

$$\begin{aligned} P(S(Seq_1, Seq_2) > x) &= 1 - P(S(Seq_1, Seq_2) < x) \\ &= 1 - e^{-Kmne^{-\lambda x}} = 1 - e^{-E(x)}\,. \end{aligned} \tag{3}$$

Formula (3) can also be generalized to apply to multiple sequences, allowing the significance of specific sequence alignments to be evaluated [33]. Through the Taylor expansion of equation (3), the *P-value* can be approximated by the *E-value* when $E(x) < 0.01$.

In general, there are two primary methods to estimate the statistical significance of local sequence alignment. One is called database statistical significance (DSS) reported by many popular database search programs, such as BLAST [38], FASTA [17], and

SSEARCH (using full implementation of Smith-Waterman algorithm [39]). This method depends on the size and composition of the database being searched. The other method is called the pairwise statistical significance (PSS) proposed by Agrawal et al. [7], which is specific to the sequence-pair being aligned, and independent of any database.

In particular, pairwise statistical significance is an attempt to make the statistical significance estimation procedure more specific to the sequence pair being compared. In addition to not needing a database to estimate the statistical significance of an alignment, pairwise statistical significance is shown to be more accurate than database statistical significance reported by popular database search programs like BLAST, PSI-BLAST, and SSEARCH [7].

This brings us to the formal definition of pairwise statistical significance. Consider that the scoring scheme *SC* (substitution matrix, gap opening penalty, gap extension penalty), and the number of permutations *N*, the PSSE of two sequences is represented as:

$$PSSE(Seq_1, Seq_2, m, n, SC, N) \ . \tag{4}$$

Through permuting Seq_2 N times randomly, the function 4 generates N scores by aligning Seq_1 against each of the N permuted sequences and then fits these scores to an EVD [17, 18, 34, 35] using censored maximum likelihood [40].

Note that the EVD distribution only applies a gapless alignment. However, local sequence alignment methods are used to find the best-matching pairwise alignments with gap penalties. For the cases of gapped alignment, although no asymptotic score distribution has yet been established analytically, computational experiments strongly indicate these scores still roughly follow Gumbel law after pragmatic estimation of the λ and K parameters [17, 18, 34, 35].

In the sense of gapless alignment, the parameters of the EVD can be obtained by theoretical computation. However, we cannot get these parameters through the same way. As a thumb of rule, the fitting for parameters is often used. A good fit play a great role in performance improvement of pairwise statistical significance in terms of homolog detection.

Through permuting $Seq2$ N times randomly, the function 4 generates N scores by aligning $Seq1$ against each of the N permuted sequences and then fits these scores to an EVD [17, 34, 35] using censored maximum likelihood [40].

Note that the EVD distribution only applies to a gapless alignment. However, local sequence alignment methods are used to find the best-matching pairwise alignments with gap penalties. For the cases of gapped alignment, although no asymptotic score distribution has yet been established analytically, computational experiments strongly indicate these scores still roughly follow Gumbel law after pragmatic estimation of the λ and K parameters [17, 18, 34, 35]. In the sense of gapless alignment, the parameters of the EVD can be obtained by theoretical computation. However, we cannot get these parameters through the same way. As a thumb of rule, parameter fitting is commonly used. A good fit plays a crucial role in performance improvement of pairwise statistical significance in terms of homology detection.

3. HPC Solutions to Accelerate PSSE

Although the estimation of PSS has been shown to be accurate, it involves thousands of such permutations and alignments, which are enormously time consuming and can be impractical for estimating pairwise statistical significance of a large number of sequence pairs. For instance, for our experiments with 86 query sequences and 2771 subject sequences, the sequential implementation takes more than 32 hours.

Therefore, applying high performance computing (HPC) techniques provides more opportunities of accelerating the estimation in reasonable time. Multi-core computers and clusters have become increasingly ubiquitous and more powerful. Therefore, it is of interest to unlock the potential of multi-core desktops and clusters using high performance technologies. On the other hand, with general purpose graphics processing units (GPGPUs) becoming increasingly powerful, inexpensive, and relatively easy to program, it has become a very attractive hardware acceleration platform. Those strongly motivate the use of multi-core CPUs, many-core GPUs or their hybrid to accelerate the estimation of PSS.

3.1. The paradigms of high performance computation techniques

In 2006, Berkeley released a technology report [41] about parallel computing research. According to this report, since power is proving to be the dominant constraint for present and future generations of processing elements, the trend of future processors would be " Small is Beautiful". Now the semiconductor industry has settled on two main trajectories for designing microprocessors [42], i.e., multi-core trajectory and many-core trajectory. The multi-core trajectory, mainly walked by ©Intel and ©AMD, began with two-core processors, with the number of cores doubling with each semiconductor process generation. The many-core trajectory began with a large number of far smaller cores. An example is the NVIDIA C2050 Graphics Processing Unit (GPU) with 448 cores, each of which is a heavily multi-threaded, single instruction issue.

To harvest the benefits of all the new generation parallel hardware including multi-core CPU and many-core GPU, one has to develop a parallel program to work with the hardware together. As a result, the four important models used in developing applications are shared-memory model (OpenMP paradigm), distributed memory model (message passing model, that is, MPI paradigm), distributed-shared memory model (the hybrid of the previous two), and attached memory model (for example, CUDA, only works for NVIDIA GPU). These paradigms are discussed as follows.

3.1.1. OpenMP paradigm

In past several decades, most classes of applications have harvested free and regular performance gains, even without releasing new versions or doing anything special, since the CPU manufacturers and memory and disk manufacturers have reliably enabled ever-newer and ever-faster mainstream systems. However, As claimed by Herb Sutter, "the free lunch is over" [43].

In recent years, the several CPU vendors, such as AMD® and Intel®, have shifted gears away from heading for more clock speeds to adding parallelism support on-chip with multicore processors (i.e., putting more simpler and smaller cores on a single chip compared to before). This practice can bypass many of the technological obstacles (such

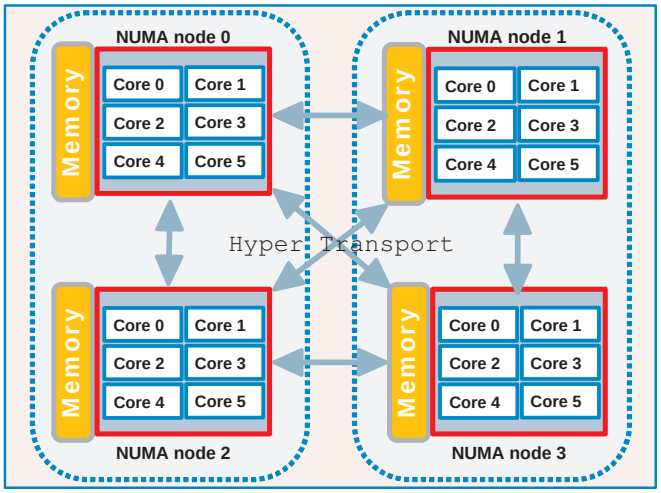


Figure 2. The non-uniform memory architecture for hybrid parallel programming paradigms

as energy and other constraints on processor designs) that CPU vendors are encountering while trying to boost speeds. But only if an application takes advantage of these multiple cores, it is able to harvest the benefits. This is where OpenMP paradigm sets foot in this picture

OpenMP is an application programming interface (API) that may be used to implement a multi-threaded, shared memory parallel algorithm. It supports multi-platform shared memory multiprocessing programming in C/C++ and Fortran. It consists of a set of compiler directives, environment variables, and library routines that determine runtime execution. OpenMP directives can be added to a sequential source code to instruct the compiler to execute the corresponding block of code in parallel. The main advantages of this technique are relative ease of use and portability between serial and multiprocessor platforms, as for a serial compiler the directives are ignored as comments [44].

3.1.2. MPI paradigm

Currently, the most often-used parallel programming paradigm is the *flat MPI* mode, in which each single-threaded MPI process is executed on one core. However, the growth in memory capacity is not keeping pace with the growth in the number of cores. Therefore, this programming model of MPI everywhere is not likely to be a viable model on these newer architectures simply because of the reduced amount of memory per core [45]. It is important to investigate a hybrid paradigm (for example, a combination of OpenMP and MPI) to avoid these issues [46].

3.1.3. Hybrid paradigm

The hybrid paradigm can match the characteristics of the cluster of multicore nodes very well, since it allows for a two-level communication pattern (i.e., intra-communications among cores within a node and inter-node communications among different nodes), as shown in Fig. 2. The main benefits of the hybrid paradigm is to take advantage of process

level coarse-grain parallelism, in which one MPI process executes on each multi-core processor, and fine grain parallelism on a loop level, in which each MPI process spawns a team of threads to occupy the multicore processor when encountering parallel sections of code using OpenMP.

3.1.4. CUDA

Although GPUs were originally designed as highly effective graphics accelerator, the modern programmable GPUs have evolved into highly parallel, many core processors with tremendous computational power and huge memory bandwidth [47]. This makes them more effective compared to general-purpose CPUs for a wide range of applications.

In 2007, NVIDIA released GPUs with Compute unified device architecture (CUDA), a software and hardware co-processing architecture. CUDA is a computing engine of GPUs that is accessible to developers through a minimalist set of variants of industry standard programming languages such as C/C++ and Fortran. Third party wrappers are also available for Python, Perl, Java, Ruby, and MATLAB, which allows application developers to write scalable multi-threaded programs for CUDA-enabled GPUs with familiar languages.

Although the current generation of GPUs provides much higher computational parallelism than the CPUs, memory management is still more complex compared to CPU. To obtain high performance, a developer needs to have a good understanding of the hierarchy and features of the GPU memory. An insightful discussion on memory management and optimizations of GPUs could be found in Ref. [48].

3.2. implementations

Careful analysis of the data pipelines of PSSE shows that the computation of PSSE can be decomposed into three computation kernels: (1) Permutation: generating N random sequences by permuting Seq_2; (2) Alignment: aligning the N permuted sequences of $Seq2$ (also known as subject sequence) with $Seq1$ (known as query sequence) using Smith-Waterman algorithm [39]; and (3) Fitting: obtaining statistical constants K and λ by fitting the scores produced in (2) into an EVD, then returning the PSS between sequence pair $Seq1$ and $Seq2$ according to Equation 3.

However, permutation and alignment comprise the overwhelming majority (more than 99.8%) of the overall execution time [12]. Therefore, efforts should be spent to optimize these two kernels to achieve high performance.

Therefore, efforts should be spent to optimize these two kernels, which will result in net significant improvements to the performance of the algorithm as a whole.

Also, we observe that permutation presents high degrees of data independency that are naturally suitable for single instruction, multiple threads (SIMT) architectures and therefore, can be mapped very well to task parallelism models of multi-core CPU and many-core GPU. In this part, we present MPI, OpenMP and GPU implementations. Furthermore, we analyze potential reasons for these performance differences.

3.2.1. implementation setup

We carried out the MPI and OpenMP implementations on Cray XE6 machines (provided by Hopper system at NERSC), each with 24 cores and 32 GB memory. Each node

contains two twelve-core AMD® "Magny-Cours" @2.1 GHz processors. As for GPU implementations, they are carried out using Intel© Core™ i7 CPU 920 processors running at 2.67 GHz. The system has 4 cores (each with 2 threads), 4 GB of memory, dual Tesla C2050 GPU (each with 448 CUDA cores). All the implementations are running in 64-bit Linux operating system, and the program has been compiled using gcc 4.4.1 and CUDA 4.1. All the performance in terms of speedup is computed over the corresponding sequential implementation.

The sequences data used in this work comprise of a non-redundant subset of the CATH 2.3 database [49]. This dataset consists of 2771 domain sequences as our subject sequences library, which represents 1099 homologous superfamilies and 623 topologies and includes 86 CATH queries serving as our query set.

3.2.2. MPI implementations

(1) Intuitive strategy

As discussed in the previous section, the kernels of permutation and alignment are independent of each other during the pairwise statistical significance estimation procedure, which map very well to programming models capable of expressing MPI task parallelism. The pairwise statistical significance estimation (PSSE) of single-pair sequences processes only one pair of query and subject sequences. The idea of computing single-pair PSSE is as follows.

Given the query sequence A and the subject sequence B, to compute PSSE, thousands (say N) of randomly permutations of B are needed. To obtain these N random sequences of B, first, a set of N random numbers is generated. Each MPI process then generates new random numbers using its own seed and swap the characters of B accordingly to obtain a permuted sequence. Thus the N random permutations of B are obtained in parallel. The algorithm then uses Smith-Waterman to compute alignment score of A and the N permuted copies of B in parallel. The scores are then gathered by process 0 for fitting.

The multi-pair PSSE processes multiple queries and subject sequences. In the rest of chapter, we mainly discussed multi-pair PSSE since the benefits of HPC are harvested usually when the bigger data are available.

The intuitive strategy for multi-pair estimation of PSS is simply extended from Ref. [50]. That paper discusses the parallelization of estimation of PSS of a single pair of query and subject sequences. We just execute that for each query and subject sequence pair. The main disadvantage of this policy is that a higher overhead of communication among processes has to be paid, as the query and subject sequences have to be broadcasted to all the processes. Also, communication is required for gathering alignment scores for each pair. Moreover, only the *root* performs the task of fitting. See Ref. [11] for more details about the MPI implementation.

(2) Tiling strategy

Unlike in the intuitive strategy, the tiling strategy employs a more coarse-grained parallelism. Tiling strategy partitions the sequence database into different disjoint sets of subject sequences and assigns one set to each process. Therefore, each process has a subset of subject sequences and all the query sequences and can estimate the PSS of each pair independent of other processes. This enhances data locality on single node and reduces the overhead of communication.

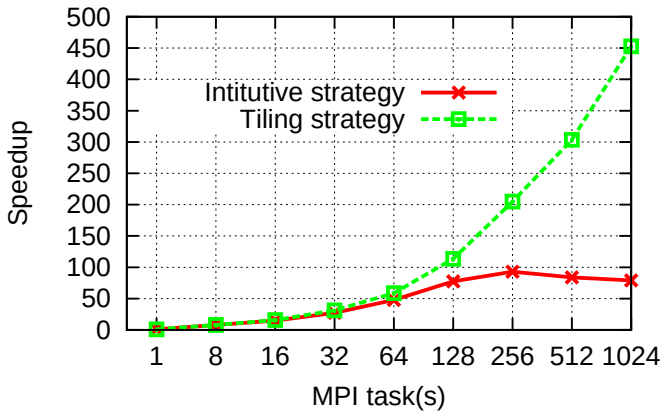


Figure 3. The speedup of MPI implementations using two different strategies

In order to achieve higher efficiency for this parallelization, the workload of each node should be roughly equal. The workload of the estimation of PSS process mainly depends on the length of query and subject sequence. Therefore, instead of distributing roughly equal number of subject sequences to different processes, this method partitions sequences based on the total length of sequences. This helps to balance the workload across nodes and improve the performance. Unlike the previous strategy, there is no need to gather alignment scores generated from every node as each node can estimate the pairwise statistical significance separately. The overhead of communication is much smaller, which is expected to result in a higher performance and better scalability. More details about this strategy can be found in Ref. [11]

We show the speedups of the two MPI implementations using intuitive and tiling strategies, respectively as shown in Fig 3. When MPI tasks $P < 64$, there are not many differences in term of speedup between the two strategies. However, when $P > 64$, due to lower overhead of communications and better load balancing, the tiling strategy implementation shows a much better scalability and higher performance than the intuitive strategy. Their maximum speedups are 452.93$\times$ and 92.92$\times$ when using 1024 MPI processes and 256 MPI processes, respectively.

3.2.3. OpenMP implementation

Compared to the implementation of MPI paradigm, OpenMP implementation uses a similar strategy to accelerate PSSE. However, OpenMP paradigm currently applies to shared memory machine, therefore, there are some differences in terms of the overhead of communications. All of the OpenMP experiments are executed on single node with 24 cores. Since the amount of computation required for a query sequence depends on its length, herein we compared the performance using multiple lengths of query sequences. Four query sequences of length 200, 400, 800, and 1600 are chosen from CATH to align against all the 2771 subject sequences. We have tested different number of OpenMP threads, T, and MPI tasks, P. Fig. 4 shows the experimental results of the OpenMP implementation. All speedups are computed over the corresponding sequential implementation. The maximum speedup is 6.10$\times$, 9.61$\times$, 14.55$\times$, and 18.94$\times$, corresponding to the

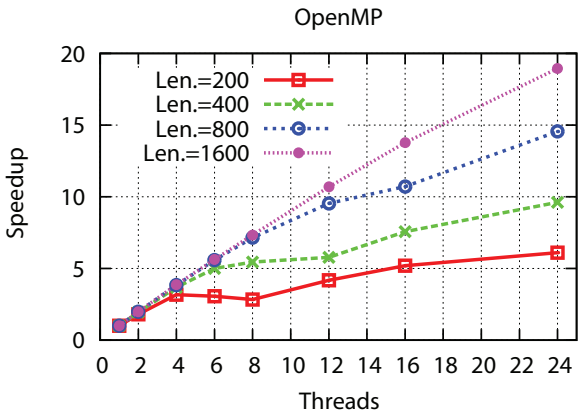


Figure 4. The speedup of the OpenMP implementation

length of query equal to 200, 400, 800 and 1600, respectively. Observe that, performance increases with the increase in length of query sequences.

Although in most cases, the non-uniform memory architecture (NUMA) used by Hopper machine provides advantages over traditional symmetric multiProcessing (SMP) solutions in terms of low level memory characteristics (such as memory bandwidth) and scalability, it also brings its disadvantages, like the NUMA effect [46]. Each NUMA node we are using is essentially a four-chip node, one of which contains a six-core die. All the memory in NUMA is transparently accessible, but if a process running on chip node i accesses the memory connected to a different chip node j (where $i \neq j$), a performance penalty will be incurred (i.e., NUMA effect). When f is lower, the performance will be more sensitive to this extra overhead. This can shed a light on why the speedup using 8 threads is even a little lower than using 6 threads when the length of query sequence is equal to 200, because the benefits of increasing threads is lower than the overhead of NUMA effect. But when the length of query sequence increases, the NUMA effect decreases. In [46], authors claimed that it is very hard to address the performance implications of the NUMA effect while using more than 6 OpenMP threads on Hopper. Our experiments confirmed this claim again.

In short, using more OpenMP threads can potentially save memory usage while it may also increase the synchronization cost (implicit or explicit) among these threads and the activation/deactivation overhead. Therefore, it is hard to generalize a common rule for setting of the optimized number of OpenMP threads to deliver the best performance, because it also depends on specific computer architectures, platforms and problem size.

See Ref. [10] for more details about OpenMP implementation of PSSE. The hybrid scheme (using OpenMP and MPI paradigms together) is also possible, whose details can be found in Ref. [51]

3.2.4. GPU implementation

In this subsection, we will discuss the solution of using GPU to accelerate pairwise statistical significance estimation. We first present the main design issues (i.e., GPU memory access optimization and maximizing *occupancy*), which play a great role in the whole performance of GPU implementations. Then we present the experimental results and discuss the reasons of performance difference.

(1) Design

A good understanding of the hierarchy and features of GPU memory is required in order to obtain good performance. It is especially important to optimize global memory access as its bandwidth is low and its latency is hundreds of clock cycles [47]. Moreover, global memory coalescing is the most critical optimization for GPU programming [48]. Since the kernels of PSSE usually work over large numbers of sequences that reside in the global memory, the performance is highly dependent on hiding memory latency.

When a GPU kernel is accessing global memory, all threads in groups of 32 (i.e. *warp*) access a bank of memory at one time. A batch of memory accesses is considered coalesced when the data requested by a warp of threads are located in contiguous memory addresses. For example, if the data requested by threads within a warp are located in 32 consecutive memory addresses (such that the *i-th* address is accessed by the *i-th* thread), the memory can be read in a single access. Hence, this memory access operation runs 32 times faster. If the memory access is not coalesced, it is divided into multiple reads and hence serialized. Whether data is coalesced or not has a significant impact on an application's performance, as it determines the degree of memory access parallelism [52].

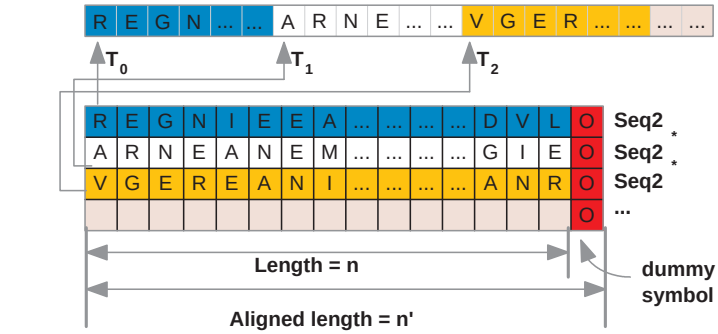
After permutation, if the sequence $Seq2$ and its N permuted copies were stored contiguously one after another in the global memory, the intuitive memory layout would be as shown in Fig. 5-(a). Note that, we need one byte (*uchar*) to store each amino acid residue. Moreover, GPU can read four-byte (packed as a CUDA built-in vector data type *uchar4*) of data from the global memory to registers in one instruction. To achieve high parallelism of global memory access, *uchar4* is used to store the permuted sequences. Dummy amino acid symbols are padded in the end to make the length of sequences a multiple of 4.

Considering inter-task parallelism, where each thread works on the alignment of one of the permuted copies of $Seq2$ to $Seq1$, in this layout the gap between the memory accesses by the neighboring threads is at least the length of the sequence. For example, in the intuitive layout, if thread T_0 accesses the first residue (i.e., 'R'), and thread T_1 accesses the first residue (i.e., 'E'), the gap between the access data is n. This results in non-coalesced memory reads (i.e., serialized reads), which significantly deteriorates the performance.

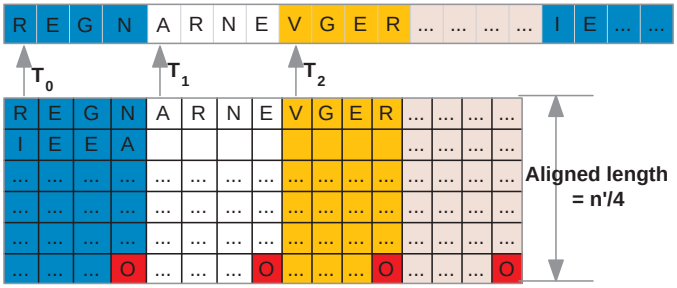
We therefore reorganize the layout of sequence data in memory to obtain coalesced reads. Now, to achieve coalesced access, we reorganize layout of sequences in memory as aligned structure of arrays, as shown in Fig. 5-(b). In the optimized layout, the characters (in granularity of 4 bytes) that lie at the same index in different permuted sequences stay at neighboring positions. Then if the first *uchar4* of the first permuted sequence (i.e. 'REGN') is requested by thread T_0, the first *uchar4* of the second permuted sequence (i.e. 'ARNE') is requested by T_1, and so on. This results in reading a consecutive memory (each thread reads 4 bytes) by a warp of threads in a single access. Thus the global memory access is coalesced, and therefore high performance is achieved.

As the sequences remain unchanged during the alignment, they can be thought of as read-only data, which can be bound to texture memory. For read patterns, texture memory fetches is a better alternative to global memory reads because of texture memory cache, which can further improve the performance.

Hiding global memory latency is very important to achieve high performance on the GPU. This can be done by creating enough threads to keep the CUDA cores always



(a) The intuitive layout that one sequence is appended after another



(b) The reorganized layout such that the uchar4 at the same indices in different sequences stay at neighoring positions

* denotes Seq2 permuted copies

T_n denotes Thread n

Figure 5. Two GPU memory layout strategies.

occupied while many other threads are waiting for global memory accesses [48]. GPU *occupancy*, as defined below, is a metric to determine how effectively the hardware is kept busy.

$$Occupancy = (B \times T_{num})/T_{max} \ , \tag{5}$$

where T_{max} is maximum number of resident threads that can be launched on a SM (which is a constant for a specific GPU), T_{num} is the number of active threads per block and B is the number of active blocks per Streaming Multiprocessor (SM).

(2) Implementations

Smith-Waterman is a dynamic programming algorithm to identify the optimal local alignment between a pair of sequences. In general, there are two different methods for parallelizing the alignment task[53]. The first method is regarded as *inter-task parallelism*. In this case, each thread performs alignment of one pair of sequences. Hence, in a thread block, multiple alignment tasks are performed in parallel [54]. The second one is *intra-task parallelism* . Here, alignment of each pair of sequences is assigned to a block of threads, splitting the whole task into a number of sub-tasks. Each thread in the thread

block then performs its own sub-tasks, cooperating to exploit the parallel characteristics of cells in the anti-diagonals of the local alignment matrix.

Through analyzing our experimental results of the single-pair PSSE, we observe that inter-task parallelism performs better than intra-task parallelism (results shown later). Hence, we consider inter-task parallelism in computing multi-pair PSSE. Based on the guidelines for optimizing memory and occupancy described earlier, we compare three implementation strategies.

- **Intuitive strategy**

Given Q query sequences and S subject sequences, the intuitive strategy is to simply perform the same 'single-pair' procedure $Q \times S$ times. In other words, in each iteration, we send a single pair of query and subject sequences to the GPU. The GPU processes that pair and returns the result to the CPU. The same procedure is repeated for all query and subject sequence pairs. However, this strategy suffers from low occupancy. We analyze the cause along with its performance results in the next section.

- **Data reuse strategy**

In the first strategy, the subject sequences from the database are permuted for every query-subject sequence pair. Hence, the same subject sequence is permuted every time it is sent to the GPU. A better strategy is to create permutations of each subject sequence only once and reuse them to align with all the queries. "One permutation, all queries" is the idea of the second strategy. Because of the reuse of permuted sequences, higher performance is expected than the first strategy. However, the occupancy of GPU, to be shown in the next section, is still not elevated.

- **Adaptively tile-based strategy**

The low occupancy of the above two strategies is due to the underutilized computing power of GPU. In addition, these two strategies do not work well when the size of subject sequence database becomes too big to be fitted into GPU global memory. For instance, if the size of the original subject sequence database is 5 MB, it becomes 5000 MB when each of the sequences is permuted, say, 1000 times. This prohibits transfer of all the subject sequences to GPU at the same time. We therefore need an optimal number of subject sequences to be shipped to GPU keeping in mind that the subject sequences and their permuted copies fit in global memory. Moreover, the number of new generated sequences should be enough to keep all CUDA cores busy, i.e., keep a high occupancy of GPU, which is very important to harness the GPU power. Herein we develop a memory tiling technique that is self-tuning based on the hardware configuration and can achieve a close-to-optimal performance. The idea behind the technique is as follows. In out-of-core fashion, the data in the main memory is divided into smaller chunks called *tiles* and transferred to the GPU global memory. In our case, the tiles are the number of subject sequences to be transferred to the GPU at a time. The tile size T can be calculated using the following equation.

$$T = \lfloor \frac{SM_{num} \times T_{max}}{N} \rfloor \tag{6}$$

where SM_{num} is the total number of SMs in GPU, T_{max} is the maximum number of resident threads per SM, and N is the number of permutations.

In Tesla C2050 used in our experiments, there are 14 SMs and the maximum number of resident threads per SM is 1024. Let $N = 1000$, then, $T = \lfloor 14 \times 1024/1000 \rfloor = 14$, which means that there are 14 distinct subject sequences to be transferred to the GPU's global memory at a time. Based on the second strategy (i.e., data reuse), 14 subject sequences and their permuted copies are aligned against one query sequence at a time, until all the query sequences are processed. As a result, 14×1000 alignment scores in total are obtained in each round, which are subsequently transferred to CPU for fitting. The CPU takes the 1000 alignment scores for each subject-query sequence pair and uses them to compute the corresponding pairwise statistical significance *pss*. The tile-based strategy has been described in Fig. 6.

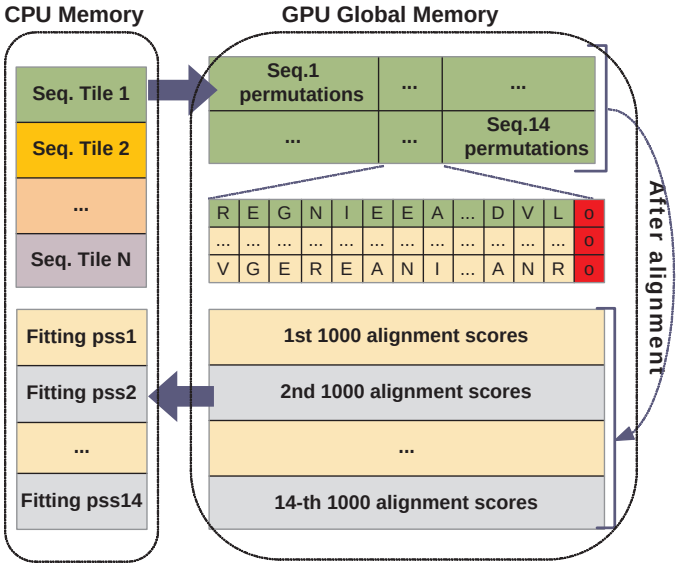


Figure 6. Adaptive tile-based strategy

(3) Performance analysis

In the intuitive and data reuse strategies, most of the SMs suffer from the same low occupancy using the inter-task parallelism method. As expected, we observe poor performance for both strategies, as shown in Fig. 7. The data reuse strategy produces higher performance than the intuitive strategy, because the number of permutations is reduced by a factor of Q, the number of query sequences. To alleviate the low occupancy problem, our proposed tile-based strategy uses a carefully tuned tile size that can effectively improve the occupancy. Recall that, as a result of tiling, we send 14 subject sequences and one query sequence to the GPU in each round. After the permutation step, we have 14×1000 alignments to be performed. Consequently, each SM has 1000 alignments to perform. Also, note that, the maximum number of blocks that can be launched simultaneously on an SM is 8. Therefore, when the number of threads per block is 64, the number of blocks per SM is $min(\lceil 1000/64 \rceil, 8) = 8$, resulting in an occupancy of $(8 \times 64)/1024 = 50\%$, based on Equations (5). This is a significant improvement over the cases using the intuitive and data-reuse strategies. The maximum theoretical occupancy of an SM for the three strategies is given in Table 1.

Table 1. The maximum occupancy for three strategies.

Threads/blocks	64	128	256	512
Intuitive Occu.	12.5%	6.25%	6.25%	6.25%
Data-reuse Occu.	12.5%	6.25%	6.25%	6.25%
Tile-based Occu.	50.0%	100%	100%	100%

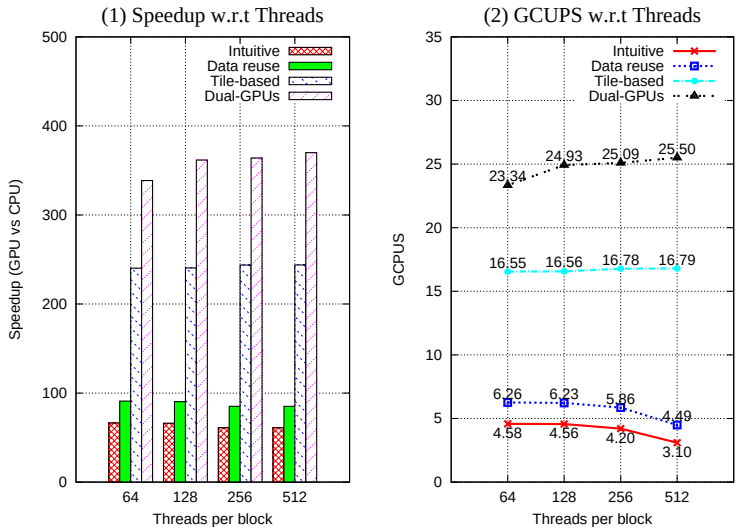


Figure 7. Performance of three strategies for multi-pair PSSE. All experiments are run using 2771 subject sequences and 86 query sequences from the CATH 2.3 database.

Due to a high occupancy, the tile-based strategy using single (dual) GPU(s) achieves significant speedups of 240.31 (338.63)×, 243.51 (361.63)×, 240.71 (363.99)×, and 243.84 (369.95)× for 64, 128, 256, and 512 threads per block, respectively, as shown in Fig. 7.

The billion cell updates per second (GCUPS) value is another commonly used performance measure in bioinformatics [53]. The tile-based strategy using single (dual) GPU(s) achieves performance results from 16.55 (23.34) to 16.79 (25.50) GCUPS, as shown in the chart of Fig. 7.

In summary, low occupancy is known to interfere with the ability to hide latency on memory-bound kernels, causing performance degradation. However, increasing occupancy does not necessarily increase performance. In general, once a 50% occupancy is achieved, further optimization to gain additional occupancy has little effect on performance. Our experiments verify this claim. More details about the GPU implementation can be found in Ref. [12, 13].

All the implementations of the proposed method and related programs are available for free academic use at `http://cucis.ece.northwestern.edu/projects/PSSE/`.

4. Conclusion

In this chapter, we first present a new method of mining genomic sequence data for related sequences using pairwise statistical significance. Then we give some high performance solutions to accelerate the estimation of the pairwise statistical significance of local sequence alignment, Our accelerator harvests computation power of multi-core CPU and many-core GPUs and by using OpenMP/MPI programming and CUDA paradigms, respectively, which results in high end-to-end speedups for PSSE. The proposed optimizations and efficient framework are also applicable to a wide variety of next-generation sequencing comparison based applications, such as DNA sequence mapping and database search.

The shorter read lengths and larger data volumes of high-throughput sequencing technologies have created substantial new requirements for bioinformatics. The "Big data" in terms of biological sequence demands more computing power. Therefore, researchers are expected to use parallelized computational workflows to mine meaningful information from the sequences, for which our work can serve as a useful stepping stone to making Big Data work in genetics.

Acknowledgement

This work is supported in part by National Natural Science Foundation of China (Contract No. 61203265) and Key Project of Henan Province (Contract No. 122102110106). This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

References

[1] C. Mason and O. Elemento, Faster sequencers, larger datasets, new challenges, *Genome Biology*. **13**(3), 314 (2012).

[2] NCBI. GenBank release notes (2012). `ftp://ftp.ncbi.nih.gov/genbank/gbrel.txt`.

[3] M. C. Schatz and A. M. Phillippy, The rise of a digital immune system, *J. of Molecular Biology*. **1**(4), 1–4 (2012).

[4] D. S. Roos, COMPUTATIONAL BIOLOGY: Bioinformatics-Trying to Swim in a Sea of Data, *Science*. **291**, 1260–1261 (2001).

[5] Q. Yang and X. Wu, 10 challenging problems in data mining research, *Intl. J. of Information Technology and Decision Making*. **5**(4), 597–604 (2006).

[6] A. Agrawal, A. Choudhary, and X. Huang. Sequence-specific sequence comparison using pairwise statistical significance. In *Software Tools and Algorithms for Biological Systems*, vol. 696, *Advances in Experimental Medicine and Biology*, pp. 297–306. Springer New York (2011).

[7] A. Agrawal, V. Brendel, and X. Huang. Pairwise statistical significance versus database statistical significance for local alignment of protein sequences. In *Bioinformatics Research and Applications*, vol. 4983, pp. 50–61, Springer Berlin/Heidelberg (2008).

[8] A. Agrawal and X. Huang, Pairwise statistical significance of local sequence alignment using sequence-specific and position-specific substitution matrices, *IEEE/ACM TCBB*. **8**(1), 194–205 (2011).

[9] A. Agrawal and X. Huang, Pairwise statistical significance of local sequence alignment using multiple parameter sets and empirical justification of parameter set change penalty, *BMC Bioinformatics*. **10**(Suppl 3), S1 (2009).

[10] Y. Zhang, F. Zhou, J. Gou, et al., Accelerating pairwise statistical significance estimation using numa machine, *J. of Computational Information Systems*. **8**(9), 3887–3894 (2012).

[11] Y. Zhang, M. M. A. Patwary, S. Misra, , et al., Par-PSSE: Software for pairwise statistical significance estimation in parallel for local sequence alignment, *Intl. J. of Digital Content Technology and its Applications*. **6**(5), 200–208 (2012).

[12] Y. Zhang, S. Misra, D. Honbo, et al. Efficient pairwise statistical significance estimation for local sequence alignment using GPU. In *ICCABS*, pp. 226–231 (2011).

[13] Y. Zhang, S. Misra, A. Agrawal, et al., Accelerating pairwise statistical significance estimation for local alignment by harvesting GPU's power, *BMC Bioinformatics*. **13**(Suppl 3), S1 (2012).

[14] D. Honbo, A. Agrawal, and A. N. Choudhary. Efficient pairwise statistical significance estimation using FPGAs. In *Proc. of Intl Conf. on Bioinformatics and Computational Biology*, pp. 571–577 (2010).

[15] C. Camacho, G. Coulouris, V. Avagyan, et al., BLAST+: architecture and applications, *BMC Bioinformatics*. **10**, 421 (2009).

[16] A. Schäffer, L. Aravind, T. Madden, et al., Improving the accuracy of PSI-BLAST protein database searches with composition-based statistics and other refinements, *Nucleic Acids Res.* **29**(14), 2994–3005 (2001).

[17] W. R. Pearson, Empirical statistical estimates for sequence similarity searches, *J. of Molecular Biology*. **276**, 71–84 (1998).

[18] R. Mott, Accurate formula for P-values of gapped local sequence and profile alignments, *J. of Molecular Biology*. **300**, 649–659 (2000).

[19] R. Bundschuh, Rapid significance estimation in local sequence alignment with gaps, *J. of Computational Biology*. **9**(2), 243–260 (2002).

[20] S. F. Altschul, R. Bundschuh, R. Olsen, et al., The estimation of statistical parameters for local alignment score distributions, *Nucleic Acids Res.* **29**(2), 351–361 (2001).

[21] S. Sheetlin, Y. Park, and J. L. Spouge, The gumbel pre-factor k for gapped local alignment can be estimated from simulations of global alignment, *Nucleic Acids Res.* **33**(15), 4987–4994 (2005).

[22] A. Agrawal and X. Huang, PSIBLAST_PairwiseStatSig: reordering PSI-BLAST hits using pairwise statistical significance, *Bioinformatics*. **25**(8), 1082–1083 (2009).

[23] A. Agrawal, S. Misra, D. Honbo, et al. MPIPairwiseStatSig: Parallel pairwise statistical significance estimation of local sequence alignment. In *Proc. of High-Performance Parallel and Distributed Computing*, pp. 470–476 (2010).

[24] A. Agrawal, A. Choudhary, and X. Huang. Non-conservative pairwise statistical significance of local sequence alignment using position-specific substitution matrices. In *Proc. of Intl. Conf. on Bioinformatics and Computational Biology*, pp. 262–268 (2010).

[25] A. Agrawal and X. Huang. Pairwise DNA alignment with sequence specific transition-transversion ratio using multiple parameter sets. In *Intl. Conf. on Information Technology*, pp. 89 –93 (2008).

[26] A. Agrawal and X. Huang. Conservative, non-conservative and average pairwise statistical significance of local sequence alignment. In *Proc. of IEEE Intl. Conf. on Bioinformatics and Biomedicine*, pp. 433–436 (2008).

[27] A. Agrawal and X. Huang. DNAlignTT:pairwise DNA alignment with sequence specific transition-transversion ratio. In *Proc. of IEEE Intl. Conf. on Electro/Information Technology* (2008).

[28] A. Agrawal, A. Ghosh, and X. Huang. Estimating pairwise statistical significance of protein local alignments using a clustering-classification approach based on amino acid composition. In *Bioinformatics Research and Applications*, vol. 4983, pp. 62–73, Springer Berlin/Heidelberg (2008).

[29] A. Agrawal, V. P. Brendel, and X. Huang, Pairwise statistical significance and empirical determination of effective gap opening penalties for protein local sequence alignment, *Intl. J. of Computational Biology and Drug Design*. **1**(4), 347–367 (2008).

[30] A. Agrawal and X. Huang. Pairwise statistical significance of local sequence alignment using multiple parameter sets. In *Proc. of ACM 2nd Intl. Workshop on Data and Text Mining in Bioinformatics*, pp. 53–60 (2008).

[31] A. Agrawal and X. Huang. Pairwise statistical significance of local sequence alignment using substitution matrices with sequence-pair-specific distance. In *Proc. of Intl. Conf. on Information Technology*, pp. 94–99 (2008).

[32] A. Agrawal, A. Choudhary, and X. Huang. Derived distribution points heuristic for fast pairwise statistical significance estimation. In *ACM Intl. Conf. On Bioinformatics and Computational Biology*, pp. 312–321 (2010).

[33] S. Karlin and S. F. Altschul, Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes, *PNAS*. **87**(6), 2264–2268 (1990).

[34] S. F. Altschul and W. Gish, Local alignment statistics, *Methods in Enzymology*. **266**, 460–480 (1996).

[35] M. S. Waterman and M. Vingron, Rapid and accurate estimates of statistical significance for sequence database searches, *PNAS*. **91**(11), 4625–4628 (1994).

[36] O. Bastien and E. Marechal, Evolution of biological sequences implies an extreme value distribution of type I for both global and local pairwise alignment scores, *BMC Bioinformatics*. **9**(1), 332 (2008).

[37] A. Dembo, S. Karlin, and O. Zeitouni, Limit distribution of maximal non-aligned two-sequence segmental score, *Annals of Probability*. **22**, 2022–2039 (1994).

[38] S. Altschul, T. Madden, A. Schäffer, et al., Gapped BLAST and PSI-BLAST: A new generation of protein database search programs, *Nucleic Acids Res.* **25**, 3389–3402 (1997).

[39] T. F. Smith and M. S. Waterman, Identification of common molecular subsequences, *J. of Molecular Biology*. **147**(1), 195–197 (1981). ISSN 0022-2836.

[40] S. R. Eddy, Maximum likelihood fitting of extreme value distributions (1997). URL `citeseer.ist.psu.edu/370503.html`. unpublished work.

[41] K. Asanovic, R. Bodik, B. C. Catanzaro, et al. The Landscape of Parallel Computing Research: A View from Berkeley. Technical report, Electrical Engineering and Computer Sciences University of California at Berkeley (12, 2006).

[42] W. Hwu, K. Keutzer, and T. Mattson, The concurrency challenge, *Design & Test of Computers, IEEE*. **25**(4), 312–320 (2008).

[43] H. Sutter, The free lunch is over: a fundamental turn toward concurrency in software, *Dr. Dobb's Journal*. **30**(3), 202–210 (2005).

[44] Openmp tutorial. URL `https://computing.llnl.gov/tutorials/openMP/` .

[45] H. Shan, H. Jin, K. Fuerlinger, et al. Analyzing the Effect of Different Programming Models upon Performance and Memory Usage on Cray XT5 Platforms. In *Cray User's Group Meeting* (2010).

[46] N. J. Wright, H. Shan, F. Blagojevic, et al. The NERSC-Cray center of excellence: Performance optimization for the multicore era. In *Cray User's Group Meeting* (2011).

[47] *NVIDIA CUDA C: Programming Guide 4.0*. NVIDIA (2011).

[48] S. Ryoo, C. Rodrigues, S. Baghsorkhi, et al. Optimization principles and application performance evaluation of a multithreaded GPU using CUDA. In *ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pp. 73–82 (2008).

[49] C. Orengo, A. Michie, S. Jones, et al., CATH - a hierarchic classification of protein domain structures, *Structure*. **5**(8), 1093 – 1109 (1997).

[50] A. Agrawal, S. Misra, D. Honbo, et al., MPIPairwiseStatSig: Parallel pairwise statistical significance estimation of local sequence alignment, *High-Performance Parallel and Distributed Computing*. pp. 470–476 (2010).

[51] Y. Zhang, M. M. A. Patwary, S. Misra, et al. Enhancing parallelism of pairwise statistical significance estimation for local sequence alignment. In *Workshop on Hybrid Mutl-core Computing*, pp. 1–8 (2011).

[52] *NVIDIA CUDA C: Best Practices Guide 4.0*. NVIDIA (2011).

[53] Y. Liu, D. L. Maskell, and B. Schmidt, CUDASW++: optimizing Smith-Waterman sequence database searches for CUDA-enabled graphics processing units, *BMC Research Notes*. **2**(1), 73 (2009). ISSN 1756-0500.

[54] S. Manavski and G. Valle, CUDA compatible GPU cards as efficient hardware accelerators for Smith-Waterman sequence alignment, *BMC BIOINFORMATICS*. **9** (S-2) (2008).