IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 16, 2019, accepted September 4, 2019, date of publication September 9, 2019,
date of current version September 20, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2940044

Implementation of Hybrid Alignment Algorithm for Protein Database Search on the SW26010 Many-Core Processor

HAO ZHANG, YOU FU, LU-BIN FENG, YUE ZHANG, AND RONG HUA
College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China

Corresponding author: Rong Hua (huarong@sdust.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFB0202002, and in part by the Natural Science Foundation of Shandong Province, China, under Grant ZR2018BF001.

ABSTRACT In biological research, biology sequence alignment algorithm aims to find similarities between sequences. As the size of biological database increases exponentially, the complexity of sequence alignment process also increases rapidly, which results in a large amount of computational time. The Sunway TaihuLight is the world's first heterogeneous supercomputer with peak performance over 100 PFlops and provides a new hardware platform for database search. In this paper we present an efficient method of protein database search based on Sunway TaihuLight supercomputer. Furthermore, we also optimize protein database search on Sunway TaihuLight to give full play to the performance of the SW26010 processor. In our proposed approach, we design hybrid sequence alignment by combining the Smith-Waterman local alignment algorithm and the Needleman-Wunsch global alignment algorithm. The protein database search is paralleled by message passing interface (MPI) and accelerated thread library (Athread). Experiment results with the Swiss-Prot database show that our implementation can effectively leverage the SW26010 processor's special hardware architecture and achieve a speedup to 15.91 times on a single node. In addition, we expand the scale to 64 nodes to test the scalability of the parallel method on the Sunway TaihuLight system, and the results show that our parallel implementation of protein database search have a good expansibility and reliability.

INDEX TERMS Load balance, many-core processor, parallel scalability, sequence alignment, Sunway TaihuLight.

I. INTRODUCTION

Sequencing facilities, analytical departments and biological laboratories around the world generate huge amounts of data, such as gene and protein sequences, which are usually called as data explosion [1], [2]. In bioinformatics, it is very important to find out necessary information from the biological database in proper time [3]. In database search, sequence alignment algorithm is the most important component, which can affect the accuracy of the results. To compute pairwise similarity, the Smith-Waterman (SW) algorithm and the Needleman-Wunsch (NW) algorithm were proposed and achieved high matching sensitivity [4], [5]. Although both of these algorithms adopt the idea of dynamic programming (DP), the alignment strategies are quite different: one is local sequence alignment, and the other is global sequence alignment. Since their time complexity and space complexity are $O(n^2)$, therefore the two algorithms require massive running time and computational resources. To overcome these disadvantages, a series of alignment algorithms using heuristic strategy, such as Blast [6] and Fasta [7] were proposed, which are considerably faster. However, this kind of algorithms does not guarantee the optimal alignment. Although heuristics algorithms can improve computational efficiency, they often fail to provide effective solutions in the loss of sensitivity. With the rise and development of many-core architectures and supporting software technologies, researchers have

The associate editor coordinating the review of this manuscript and approving it for publication was Chong Leong Gan.

made lots of effort on accelerating the compute-intensive programs, for example DP algorithms. Experiments show that many-core acceleration can effectively reduce the running time on the premise of ensuring accuracy.

In recent years, with the failure of Moore's law, the main frequency increase of CPU only with single core is subject to heat radiation, transistor technology, etc., while many-core processor represented by Xeon Phi, GPU and others are becoming the mainstream. More and more researchers are working to take full advantage of the hardware to speed up programs. As we all know, nowadays GPUs are developed as coprocessors for general-purpose computing on the GPUs (GPGPU). For example, the Tesla P100 GPU based on Pascal architecture has 1792 computing units in double precision and 732 GB/sec stacked bandwidth that could provide strong supports for scientific computing [8]. Further, the development of general parallel computing architectures, such as CUDA [9] and OpenCL [10], provides an easy-to-use platform for programmers to develop applications on the GPUs. At the same time, Intel Xeon Phi processors was introduced, which have more processing cores than traditional CPUs. For instance, the Xeon Phi processor codenamed Knights Landing (KNL) has 72 computing cores and 475 GB/sec memory bandwidth, making it possible to parallel programs while maintaining the Intel architecture [11]. Previous studies on protein database search mainly focused on GPU, Intel Xeon Phi and other platforms, while few studies on the Sunway TaihuLight have been conducted. The Sunway TaihuLight architecture is significantly different from other many-core architectures, with more limitations on-chip cache and memory bandwidth, which makes porting and optimizing the hybrid alignment algorithm very difficult.

The Sunway TaihuLight, the fastest supercomputer in China, is the world's first supercomputer with a peak performance over 100 PFLOPS [12]. It is designed for large-scale computing in the fields of scientific computing and industry and built up based on the home-grown SW26010 many-core processor. The SW26010 processor comprises four Core Groups (CGs), each of which includes the Management Processing Element (MPE) and Computing Processing Elements (CPEs). As mentioned above, the architecture of Sunway TaihuLight is completely different from other existing heterogeneous architectures, such as CPU-GPU. And the system provides a unique programming language and compilation environment. Normal C/C++ or FORTRAN codes can directly execute on the supercomputer but not for optimal performance [13]. Therefore, how to effectively transplant the hybrid alignment algorithm to the Sunway TaihuLight is of great concerns.

In this paper, a parallel hybrid alignment algorithm for protein database search based on the Sunway TaihuLight is designed and implemented with MPI and Athread parallel programming interface. Our hybrid alignment algorithm integrates SW algorithm and NW algorithm into an alignment process to simultaneously achieve global and local alignment. Focus on the structure of Sunway TaihuLight, we design a special data structure for sequence alignment, which can effectively reduce the space of dynamic programming array and increase the space utilization rate of SPM. We design and optimize the hybrid alignment algorithm on a single SW26010 node. Experimental results demonstrate that the proposed parallel hybrid database search method achieves a 15x speedup on a single node. Besides, we scale our method proposed in this paper to multiple computing nodes. Compared to the single-node version, we achieve a 63x speedup on 64 SW26010 computing nodes.

The remaining of this paper is organized as follows. We first introduce the background of our research in Section II, then in Section III discuss the parallel design and optimization of our hybrid alignment algorithm on Sunway TaihuLight. In Section IV, the experimental method is presented and the performance of the database search is evaluated comprehensively. Finally, we conclude our work and propose the future work.

II. BACKGROUND

A. THE SUNWAY TAIHULIGHT SUPERCOMPUTER

The Sunway TaihuLight is a new generation heterogeneous many-core supercomputer system developed by the National Research Center of Parallel Computer Engineering Technology (NRCPC) in Wuxi, China. The peak speed of the Sunway TaihuLight is 125.436 PFlops, which is the first supercomputer in the world with a speed exceeding 100 PFlops [14], [15]. Moreover, the Sunway TaihuLight supercomputer system adopts an extensible polymorphic composite architecture oriented to high-performance computing, adopts high-density assembly, high-efficiency dc power supply, and whole-machine water cooling key technologies, and is equipped with accurate resource scheduling management, rich parallel programming language and development environment.

The Sunway TaihuLight supercomputer system expands the system through the modes of computing plug-in board, computing super node and computer warehouse [16], [18]. Its architecture mainly consists of high-speed computing system, auxiliary computing system. In addition, there are high-speed computing interconnection network (corresponding with the high-speed computing system, 16GB/sec two-way bandwidth) and auxiliary computing interconnection network (corresponding with the auxiliary computing system, 112GB/sec two-way bandwidth). These four parts together with high speed computing storage system, auxiliary computing storage system and a large number of software systems make up the whole supercomputer system.

B. THE SW26010 PROCESSOR

The Sunway TaihuLight supercomputer adopts the SW26010 heterogeneous many-core processor, which is developed by Shanghai High Performance Integrated Circuit Design Center (HPICDC) through independent technology. The SW26010 processor uses 64-bit Sunway instruction set, with the standard working frequency of 1.45 GHz and the memory

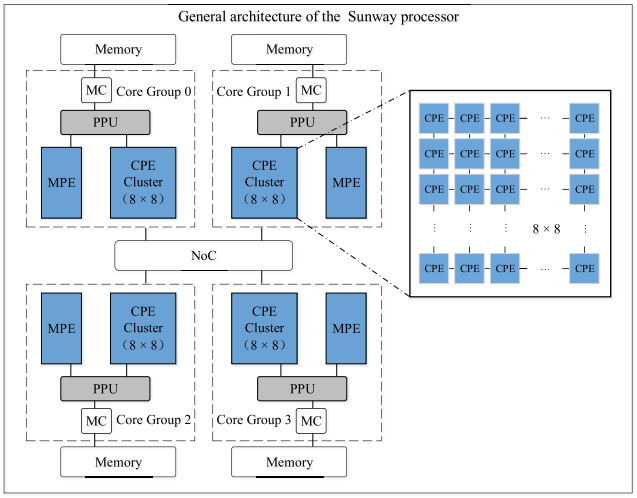

FIGURE 1. The SW26010 processor architecture.

bandwidth of 130 GB/sec. As shown in Figure 1, there are four Core Groups (CGs), each containing a Management Processing Element (MPE) and Computing Processing Element (CPE) cluster. Among them, the CPE cluster consists of 64 computing cores, the array controller and L2 instruction cache. Each CG has one memory controller (MC) and 8 GB physical memory. Every MPE has two caches for data and instruction, respectively. The MPE has a 256 KB L2 cache both for data or instruction. Every CPE has a 16 KB L1 cache for instruction and a CELL-like processor 64 KB Scratch Pad Memory (SPM) instead of the data cache. The SPM is controlled by the user, which is convenient for the user to program according to the characteristics of the program. The CPE cluster adopts DMA to transfer data between the main memory and the SPM.

Based on the SW26010 processor architecture, there are two user modes available including CG private mode and Chip-sharing mode. The main difference between these two modes is the memory running scheme. The CG private mode is the most common mode, with each CG sharing 8 GB private memory. Applications can run on four MPEs using MPI or four CGs using MPI plus OpenACC* / Athread (every MPE binds 64 CPEs). The Chip-sharing mode is designed to accommodate large memory requirements. Four CGs provide a 32 GB physical memory and each running process having access to 32 GB memory.

C. HYBRID SW AND NW DATABASE SEARCH BASED ON THE SW26010

Biological database search is a process of studying the similarity between a query sequence and target sequences in the database by calculating alignment scores. We present a SW26010-based database search method, which combines the Smith-Waterman algorithm and the Needleman-Wunsch algorithm for local and global alignments. Usually, the local alignment algorithm is adopted in the biological database search. Because the length of most query sequences differs greatly from that of the sequences in the database, so the local alignment algorithm is more suitable in this case. Our motivation for adopting a hybrid method for database search is as follows. Firstly, it is more reasonable to use the global alignment algorithm to study the similarity between the query sequence and target sequences in the database when their lengths are similar. Besides, the results obtained by simply using the local alignment algorithm may not reflect the similarity between sequences. Therefore, it is more reasonable to adopt a hybrid method combing local alignment and global alignment. At the same time, the hybrid method could save some time than adopting two types of algorithms respectively, because some computational steps can be reduced. Secondly, Polyanovsky *et al.* [19] proved that in some cases the results of the global alignment algorithm are more accurate and competent than those of the local alignment algorithm. From their research, we can see that when two sequences of similar length are aligned, the global alignment algorithm is more stable in the longer evolutionary distance and larger non-homologous parts than the local algorithm in the case that the position of non-homologous parts of each sequence is symmetric.

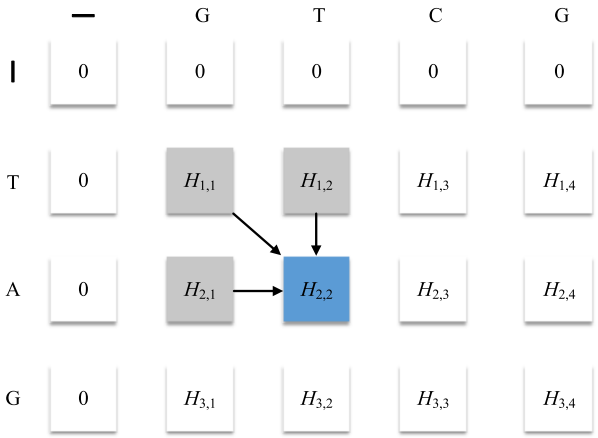

FIGURE 2. Assisted DP matrix of SW algorithm.

The Smith-Waterman algorithm is the most famous algorithm that uses dynamic programming strategy for local sequence alignment. The score of each element in the DP matrix is determined by its diagonal, column, or row direction score. As shown in Figure 2, the query sequence "T-A-G" and the target sequence "G-T-C-G" construct a scoring matrix **H** with the size of 4 x 5 as an assisted DP matrix. The assisted **DP** matrix will be computed after initializing the first row and first column to zero. The final best alignment result doesn't need to include the whole sequence. Any position in the sequence may be the starting backtracking point of the alignment result, and any position may be the ending point of the alignment result. Given two sequences S_1 and S_2, m and n represent the length of S_1 and S_2, respectively. The SW algorithm formula is listed below (see Eq. 1). Matrix **E** and **F** are two assisted matrices when calculating matrix **H**, recording the horizontal and vertical gap extending penalty, respectively. Parameter ρ is the gap opening penalty and σ is

the gap extending penalty. The *sbt* is the score matrix which is used for obtaining the similarity score between the *i*th residue in S_1 and the *j*th residue in S_2.

$$\begin{aligned} \boldsymbol{E}(i,j) &= \max(\boldsymbol{E}(i,j-1), \boldsymbol{H}(i,j-1) - \rho) - \sigma \\ \boldsymbol{F}(i,j) &= \max(\boldsymbol{F}(i-1,j), \boldsymbol{H}(i-1,j) - \rho) - \sigma \\ \boldsymbol{H}(i,j) &= \max(0, \boldsymbol{E}(i,j), \boldsymbol{F}(i,j), \boldsymbol{H}(i-1,j-1) \\ &\quad + sbt(\boldsymbol{S}_1[i], \boldsymbol{S}_2[j])) - \sigma \end{aligned} \tag{1}$$

The Needleman-Wunsch algorithm is a kind of algorithm widely used in global sequence alignment with dynamic programming strategy. By this algorithm, we could obtain the optimal global alignment between two sequences. The matrix elements $\boldsymbol{H}_{i,j}$ will be computed (see Eq. 2) after the $\boldsymbol{H}_{0,0}$ is initialized to zero. Compared to local alignment SW algorithm, there has the following two main aspects of the differences. First, the penalty points between prefix strings and spaces must be included in the global alignment matrix. Second, the penalty points between suffix strings and spaces must be included in the global alignment matrix.

$$\begin{aligned} \boldsymbol{E}(i,j) &= \max(\boldsymbol{E}(i,j-1), \boldsymbol{H}(i,j-1) - \rho) - \sigma \\ \boldsymbol{F}(i,j) &= \max(\boldsymbol{F}(i-1,j), \boldsymbol{H}(i-1,j) - \rho) - \sigma \\ \boldsymbol{H}(i,j) &= \max(\boldsymbol{E}(i,j), \boldsymbol{F}(i,j), \boldsymbol{H}(i-1,j-1) \\ &\quad + sbt(\boldsymbol{S}_1[i], \boldsymbol{S}_2[j])) - \sigma \end{aligned} \tag{2}$$

D. RELATED WORK

In recent years, many-core processor has become the mainstream processor in the academic world and industrial world. More and more researchers with a need for large-scale arithmetic computations began to use many-core processor to parallel programs. In bioinformatics, as the powerful computing capacity of many-core processor, it has been widely applied to accelerate calculation. In order to improve the performance of sequence alignment in biological databases, following works (e.g. [20]–[27]) have studied the sequence alignment algorithm adopting the related parallel acceleration technology, and achieved the desired results.

Zhou *et al.* [20], [21] propose a hybrid alignment method based on the SW algorithm and NW algorithm on GPU platform. They test their method on a single GPU with GF 750Ti card or GTX 1070 card. And they carry out experiments on GPU clusters when aligning in the Swiss-Prot database. The hybrid alignment implementation can achieve a 159.89 times speedup over the serial solution.

Liu *et al.* [22] present a parallel protein alignment method of SW algorithm with the OpenGL on GPU platform, and they adopt two stages in the OpenGL rendering pipeline: geometry transformation and fragment rasterization. They carry out experiments by using a protein sequence including 16384 acids with a database of 983 protein sequences. Compared to the serial algorithm on a 3.2 GHz Pentium D 840 processor, the method presented in this paper achieved a 4x speedup on one GF 7800 GTX card.

Manavski and Valle [23] propose a parallel SW sequence alignment algorithm based on CUDA framework. In the sequence alignment process, they select BLOSUM50 score matrix to participate in the calculation. In the experiment, they adopt the Swiss-Prot database with more than 250000 sequences. The experiment results show that their method achieves a more than 30 times speedup over a CPU implementation using a single Pentium IV 3.0 GHz processor.

Rucci *et al.* [24], [25] design and implement the SW algorithm with OpenCL in a FPGA platform. The implementation of the Smith-Waterman algorithm take full advantage of data and pipeline parallelism on the FPGA platform. The experiments demonstrate that the FPGA version of the Smith-Waterman algorithm can reach up to 114 GCUPS in less than 25 watt power requirements.

Liu and Schmidt [26] present SWAPHI-LS to accelerate the long DNA sequences by exploiting emerging Xeon Phi many-core processors. In this paper, they adopt instruction-level parallelism within a single Xeon Phi and exploit thread-level parallelism over the many cores. The performance evaluation in this paper shows that the computing performance of their implementation can achieve up to 111.4 GCUPS on four Xeon Phi processors.

Siriwardena *et al.* [27] propose a GPU heterogeneous solution for NW alignment algorithm based on CUDA programming framework. They conduct all experiments on a single Nvidia Geforce 8800 GT card and the solution presented in this paper can achieve a 4.2 times speedup over a CPU implementation with a 2.4 GHz Intel Quad Core processor.

There are a lot of studies on accelerating sequence alignment algorithm on many-core platforms, but none of them can be directly applied to the Sunway TaihuLight architecture. Compared with other many-core processors like GPU and Xeon Phi, the smaller capacity of on-chip cache and more limited memory bandwidth make our porting work more challenging. In other words, it is necessary to make full use of special architecture of the Sunway TaihuLight to parallel the alignment algorithms.

III. PARALLEL DESIGN AND OPTIMIZATION

The main component of our protein database search method is the SW-NW hybrid sequence alignment algorithm. Therefore, how to successfully transplant and optimize SW-NW hybrid sequence alignment algorithm on the Sunway TaihuLight is the key issue we need to discuss. First, this paper proposes parallel implementation of SW-NW hybrid alignment algorithm that combines sliding window technique and multi-threading on single SW26010 node (Section III-A). Second, we optimize the parallel sequence alignment method based on special architecture of the Sunway TaihuLight (Section III-B). Finally, we present parallel design of multi-node on the Sunway TaihuLight (Section III-C).

A. PARALLEL IMPLEMENTATION DETAILS ON SINGLE NODE

The SW-NW hybrid alignment algorithm is a typical compute-intensive algorithm, and in the protein database search, sequence alignment processes are not correlated with

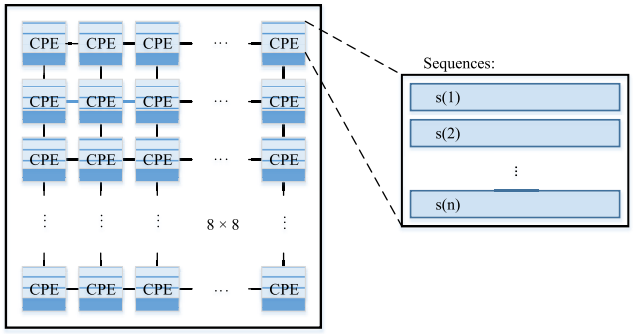

FIGURE 3. Mapping manner between CPE cluster and alignment sequences.

each other, which is very suitable for parallelization. As mentioned above, the SW26010 processor has four CGs which are independent of each other. Similarly, within the CG, CPEs are independent of each other. Therefore, we adopt a SIMD manner [28], [29] to accelerate the database search algorithm. In our method, CGs can execute the same sequence alignment procedure, however, each CG independently carries out the alignment of separate database sequences. As shown in Figure 3, the CPE cluster aligns $64 \cdot n$ sequences in the separate database. Another manner is to calculate along anti-diagonal direction of the DP matrix [27], [30], [31]. According to the manner, computations on the anti-diagonals are independent of each other and be easily parallelized. It can be seen that anti-diagonal manner is more effective for fine-grained computation, especially when aligning two long sequences. If we use four CGs for the pairwise alignment of long sequences, operations such as thread synchronization and process synchronization will greatly reduce the performance of the program. To sum up, our database search method using SIMD will be more conducive to parallel acceleration. The SW-NW hybrid alignment algorithm on the SW26010 processor is shown in Algorithm 1.

The design and implementation of the SW-NW hybrid alignment algorithm on single CG is our main task. In our design, the whole database search process is divided into two parts: MPE code and CPE code. The MPE code is responsible for IO operation, control flow and data collection in database search. Meanwhile, in our proposed database search method, the hybrid SW and NW alignment algorithm costs the most time, which is placed in the CPE cluster for parallel acceleration. Figure 4 shows the design of our parallel database search on single node. First, we divide the database data into 64 blocks in the MPE. Next we call the multi-thread function to open the CPE cluster. Second, we transfer data blocks bound to the specific CPEs and transfer query sequences from the main memory to SPM through the DMA mode. Because of the limited memory, it is difficult to transfer the entire sequences in the data block to the SPM at one time, so we need to transfer multiple times. After completing the hybrid sequence alignment, we can transfer the optimal value from the SPM to the main memory through DMA. Finally, we close the CPE cluster after getting the final result by comparing the returned optimal values.

Algorithm 1 The Pseudo-Code of SW-NW Hybrid Alignment Algorithm on the SW26010 Processor

```
Divide the target sequence database into 4 CGs
//The following code running in each CPE
//Number of target sequences → n_ts
//Length of query sequence → len_qs; Length of the ith target sequence → len_ts[i]
for i = 0 to n_ts do
  Initialize the DP matrix H, E and F for SW and NW algorithm
  for j = 0 to len_qs do
    for k = 0 to len_ts[i] do
      Load E(i, j-1) and H(i, j-1), and calculate E (i, j-1) for SW and NW algorithm
      Load F(i, j-1) and H(i, j-1), and calculate F (i, j-1) for SW and NW algorithm
      Load H(i-1, j-1), and load sbt(S1[i], S2[j])for SW and NW algorithm
      Compute H(i, j) for SW and NW algorithm:
      H(i, j) = max(0, E(i, j), F(i, j), H(i-1, j-1) +sbt(S1[i], S2[j]))-σ
      Obtain the optimal alignment score by comparing the previous maximum H_sw and H_nwvalue
      Store H(i, j), F(i, j) and E(i, j) for SW and NW algorithm
    End do
  End do
  Obtain H_max by comparing the previous optimal alignment score for SW and NW algorithm
End do
```

Considering that the CPEs use gld/gst discrete mode to access the main memory is too slow, we take full advantage of local data memory in the CPEs to perform many-core acceleration on the SW-NW hybrid alignment algorithm. We find it is easy to put sequences from the database into the SPM, but fail to apply the memory of the DP matrix in the CPEs. As described in Section II-C, if the query sequence and the target sequence have lengths of n and m respectively, we need to construct a scoring matrix with the size of $(n + 1)^*(m + 1)$. When the query sequence or the target sequence exceeds a certain length, the scoring matrix we construct will be larger than 64 KB SPM capacity. We study the dynamic programming process and find that each element in the matrix is updated only with respect to the previous and current rows, so that we could optimize memory usage by using sliding window technology. Obviously, the size of the scoring matrix we construct changes to $2^*(m+1)$, which saves $(n+1)/2$ times space. At the same time, we can also transfer more target sequences from the main memory to the SPM, which reduces the number of transfers and saves more running time.

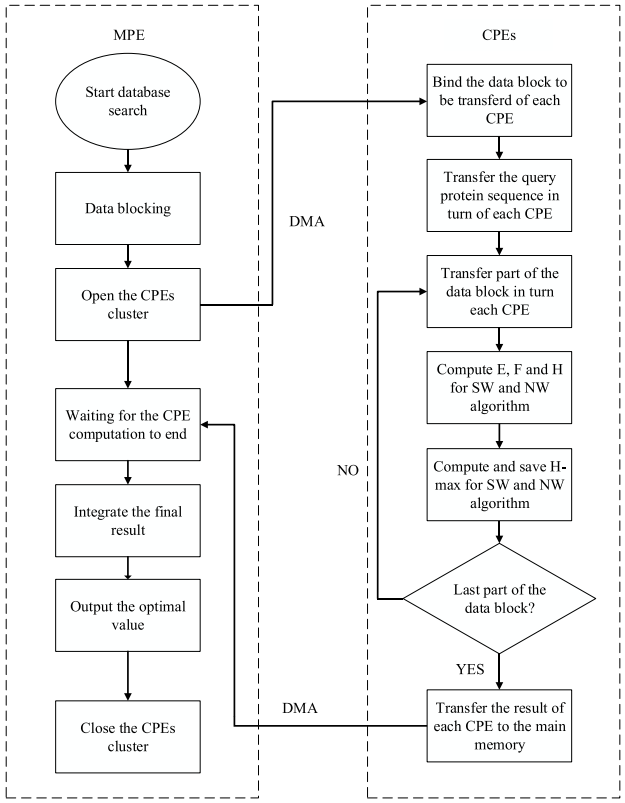

FIGURE 4. The flowchart of implementation in one CG.

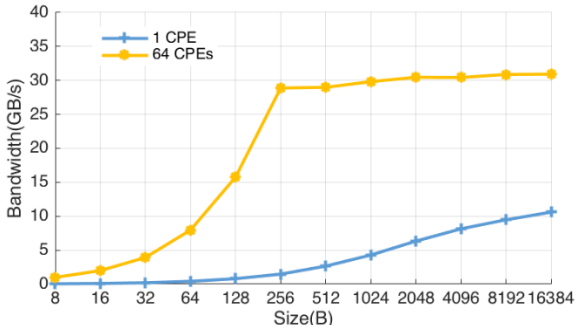

FIGURE 5. The memory access bandwidth between main memory and CPEs.

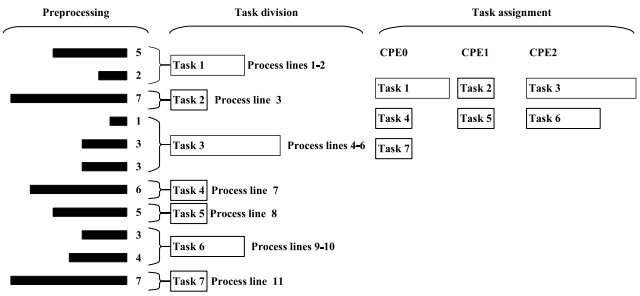

FIGURE 6. The improved task assignment strategy.

B. OPTIMIZATION METHODS ON SINGLE NODE

Based on the SW26010 processor architecture and our parallel approach, we propose three kinds of optimizations, including memory access optimization, manual prefetching and load balancing strategy. Since the SW-NW hybrid alignment algorithm is the most time-consuming part in database search, we focus our optimization on the CPE part. The details of optimization methods are briefly discussed below.

As described in Section II-B, each CPE core has a user-controlled SPM, which gives programmers more programming possibilities. Comparing with automatic cache design in other architectures, using SPM has more controllability and subjectivity for programmers. The Sunway TaihuLight system supports DMA communication by specific instructions that allow faster data transfer between the MPE and CPEs [32]. The DMA operations can only be issued by CPEs. At the same time, we test the memory access bandwidth of CPEs on the Sunway TaihuLight and the results are shown in Figure 6. It can be seen that when the data in the main memory is aligned with 128 B and the number of transfers is larger than 2KB in 64 CPEs, DMA intrinsic will achieve the best performance. There are five kinds of DMA modes, such as the single-CPE mode, the broadcast mode, the single-row mode, the row-broadcast mode and the array mode. According to the characteristics of our parallel approach, each CPE aligns its own sequences separately, so we adopt the single-CPE mode.

In CPE parallel region, although DMA intrinsic can greatly reduce the overhead of memory access, memory access often restricts the parallel efficiency. In order to further improve the parallel efficiency, we use manual prefetching technology to give full play to the asynchronous performance of DMA intrinsic. When there are multiple rounds of DMA operations, the CPE needs to apply double memory space to buffer the data of MPE and CPE. The CPE initially starts the transfer, and then performs computing operations during the transfer. Therefore, we can execute our parallel algorithm while transferring the next data block.

At the same time, the basic task allocation strategy is to divide tasks coarsely according to the number of sequences, and does not consider the task load balancing among different CPEs. So, we divide tasks into fine-grained tasks according to the sequence length in the database. As shown in Figure 6, the task assignment process consists of three parts. In preprocessing part, we need to calculate the length of protein sequences in the database. Task division part divides the whole sequences in the database into several subtasks according to the sequence information and the capacity of the SPM. Figure 6 illustrates that each CPE can process multiple sequences with a total length of 7 in a computing cycle. In the last part, we adopt a static way to assign subtasks to each CPE.

C. PARALLEL DESIGN ON THE SUNWAY TAIHULIGHT

For a large number of sequences, it is very important to study the scalability of our database search method in the Sunway TaihuLight supercomputer. In this paper, we study and discuss the scalability of parallel database search on the Sunway

TaihuLight with different number of the SW26010 processor node.

The parallel design based on single node does not need complex data exchange, which provides a useful idea for us to realize multi-node version. We implement the parallel program based on multiple SW26010 processors with MPI in the same way as the single-node implementation. The parallel program first divides sequences in the database evenly to each SW26010 processor node, and then process them in parallel within processor nodes. The parallel processing procedures within the node are consistent with the description of Algorithm 1.

IV. EXPERIMENT AND RESULTS

A. EXPERIMENT METHOD

To evaluate the effectiveness of our method, we first test on single node of the Sunway TaihuLight, and obtain their running time, the GCUPS and speedup ratio compared to the serial version. Due to the huge difference between the Sunway TaihuLight system and other many-core platforms, there is no research significance for direct performance comparison between them. So, we do all the experiments and performance comparison on the Sunway TaihuLight. And then we study the scalability of our method on multiple nodes. The specifications of Sunway processor are shown in Table 1. All the algorithms implemented in this paper are edited in C language and MPI 3.0.

TABLE 1. The Sunway Taihulight system specifications.

CPU	SW26010 processor
SW26010 Processor	4 CGs (4 MPEs and 256 CPEs)
Operating System	Sunway Raise OS 2.0.5
Memory	8 GB DDR3 per CG
Instruction Set	Sunway-64 Instruction Set
Programming Language	C, C++, Fortran
Parallel Programming Language	MPI 3.0, OpenMP 3.1, OpenACC 2.0

TABLE 2. The information of query sequences.

Sequence	B0M3E5	Q3E7A0	A4J3F7	P5944 5	A8NVA4	P50909
Length	15	39	81	99	199	389

The dataset used in experiments is from March 2019 release of the Swiss-Prot database, and the threshold value of sequence length is set as 1000. In the experiments, there are five randomly selected sequences as query sequences. The information of query sequences is shown in Table 2. And we set the gap open penalty ρ to 5 and the gap extension penalty σ to 1. In order to eliminate accidental errors in the Sunway TaihuLight system, we perform the algorithm ten times for each query sequence and take the average as the final execution time of the algorithm. To illustrate the performance of the protein database search algorithm more intuitively, the GCUPS and parallel speedup are adopted to measure the effectiveness of program parallelism. The GCUPS is often used to evaluate the performance of the sequence alignment algorithm, as defined in Eq. (2). The variables m and n represent the length of two sequences involved in alignment, respectively. The speedup is used to measure performance and effectiveness of parallelization of a parallel system or program. The specific formula is shown in Eq. (3).

$$\text{GCUPS} = \frac{mn}{t \times 10^9} \tag{3}$$

$$\text{Speedup} = \frac{\text{Serial Running Time}}{\text{Parallel Running Time}} \tag{4}$$

B. SINGLE-NODE PERFORMANCE EVALUATION

Figure 7 shows the parallel performance comparison of the serial algorithms and the parallel algorithms for different query sequences. The six sub-figures from left to right represent the running time of different query sequence, respectively. It can be concluded from all sub-figures that the serial run time of the database search algorithm increases as the sequences length increases. Besides, we can get from each sub-figure that the computing performance of the parallel algorithm is much higher than that of the serial algorithm. Compared to the running time of the serial algorithm, the optimized single-node algorithm could save 93.10%, 93.14%, 93.38%, 93.49%, 93.62% and 93.71% the time when executing the protein database search algorithm, respectively. Meanwhile, we count the major MPI time overheads included in the running time, as listed in Table 3.

Figure 8 presents the GCUPS results of each query sequence. It can be observed from the figure that our proposed algorithm could obtain a better GCUPS scores than the serial algorithm. The database search algorithm achieves the highest score of 9.85 GCUPS and the lowest score of 9.03 GCUPS when aligning the sequences B0M3E5 and P50909, separately.

The speedup ratio of the basic parallel method and the optimized method based on a single node are illustrated in Figure 9. As we can see in Figure 9, compared with the serial algorithm, the basic parallel algorithm achieves a higher speedup, while the optimized parallel algorithm further improves the speedup ratio and the parallel efficiency. In the six sub-figures, we can get the highest speedup of 15.91 times when query sequence P50909 is aligned.

C. MULTI-NODE PERFORMANCE EVALUATION

The scalability study of the algorithms on the Sunway TaihuLight is crucial, and the Sunway TaihuLight system provides MPI interface, so this paper designed and conducted parallel experiments on multiple SW26010 processors. In the experiments, we guarantee the task load of all nodes on the Sunway TaihuLight is balanced. We select the sequence P50909 as the query sequence, and the target sequence database also uses the Swiss-Prot database, including 541122 sequences. Since we only prove scalability of the protein database search algorithm, we do not perform all the node tests on the whole Sunway TaihuLight supercomputer, but only on 1, 2, 4, 8, 16, 32, 64 processor nodes.

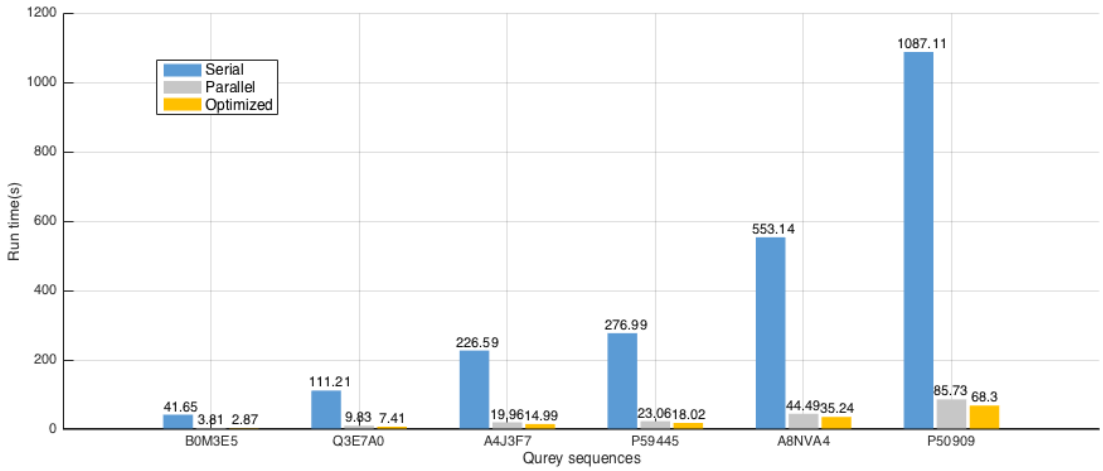

FIGURE 7. Comparison results of running time.

TABLE 3. The overhead of major MPI operations for each sequence.

Sequence	B0M3E5	Q3E7A0	A4J3F7	P59445	A8NVA4	P50909
Broadcast query sequence(s)	0.016	0.018	0.018	0.018	0.018	0.017
Send search results(s)	0.023	0.023	0.022	0.021	0.022	0.021
Integrate search results(s)	0.097	0.094	0.098	0.096	0.101	0.096

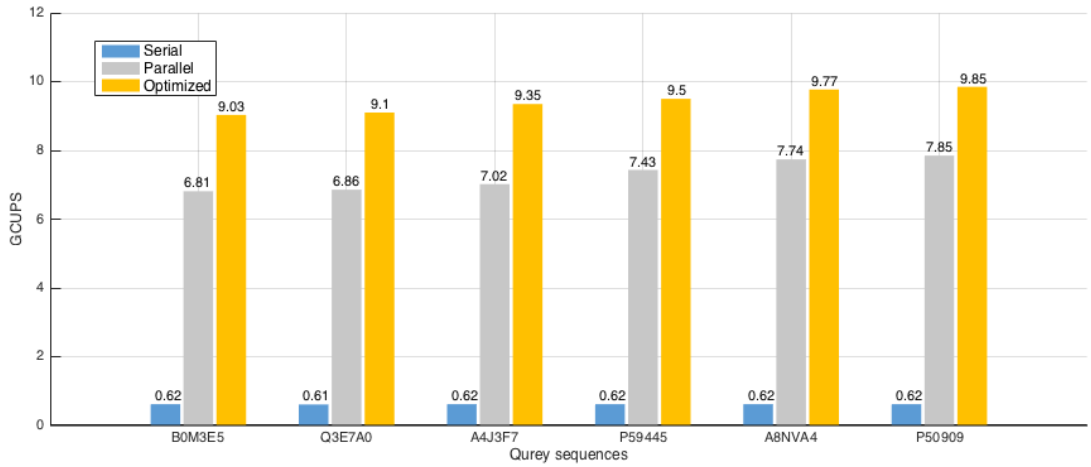

FIGURE 8. Performance results measured in GCUPS.

TABLE 4. The running time of database search on different numbers of nodes.

Number of Nodes	Execution Time(s)
1	68.32
2	34.50
4	17.20
8	8.73
16	4.37
32	2.16
64	1.07

Table 4 presents the running time of the protein database search algorithm with the number of SW26010 processor nodes increasing. We can clearly see that the running time of our algorithm decreases as the number of nodes increases gradually, and the running time decreases to 1.07 seconds at 64 nodes. This illustrates that our algorithm can achieve better performance by using more processor nodes.

Figure 10 presents the speedup performance of the protein database search algorithm with the number of SW26010 processor nodes increasing. Among them, we choose the running time of the optimized algorithm in a single node as the benchmark to calculate the speedup ratio. As seen, the speedup increases along with the increase of the number of SW26010 processor nodes. Since there is little data communication between processor nodes in our parallel algorithm design, the curve in the figure will keep exponential growth

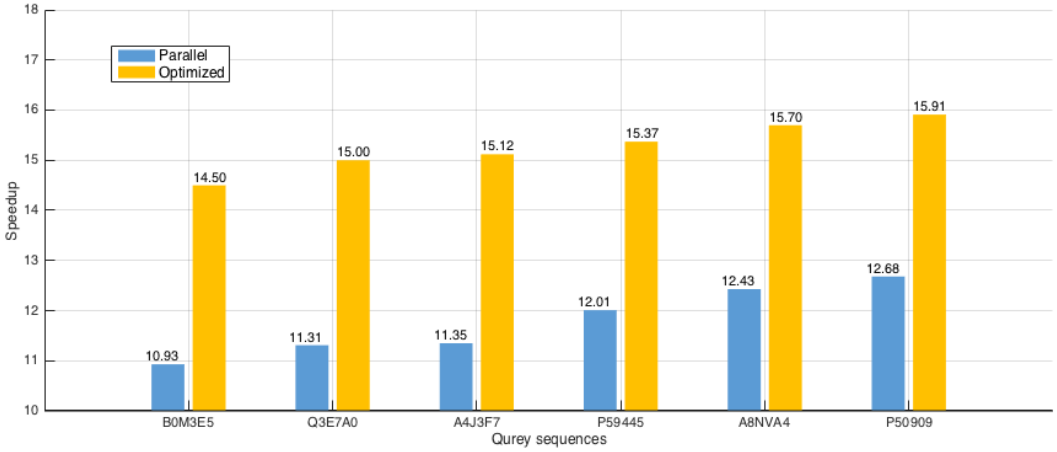

FIGURE 9. Speedup results of protein database search algorithm.

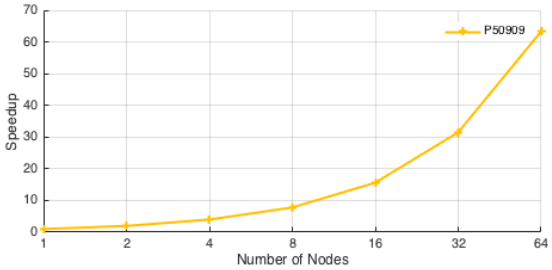

FIGURE 10. Speedup of aligning P50909 query sequence on different numbers of nodes.

as the number of nodes increases. Experiments show that our algorithm design has good scalability and parallel efficiency on SW26010 processors.

V. CONCLUSION AND FUTURE WORK

In this paper, we design and implement a hybrid SW and NW algorithm for the protein database search based on the Sunway TaihuLight. The strategy of the proposed method is to merge the SW local algorithm and NW global algorithm into one process. Specifically, we implement the parallel method of database search on one SW26010 processor, which can allow full play to the SW26010 processor's computing power. In addition, we optimize the parallel database search in three aspects: memory access optimization, manual prefetching and load balancing strategy. Finally, we test the effectiveness of our method on a single node and the scalability of the method on multiple nodes. Specifically, it not only can achieve top speedup of 15.91 times on a single node, but also has good scalability and parallel efficiency on multiple nodes.

For future work, we would study the parallel algorithm based on other heterogeneous platforms, such as FPGA and Xeon Phi, and propose a parallel model of sequence alignment. In addition, we will further explore the special architecture of the Sunway TaihuLight, especially how to use SPM more efficiently, and further optimize the parallel database search method at assembly level.

REFERENCES

[1] H. Li, "Minimap2: Pairwise alignment for nucleotide sequences," *Bioinformatics*, vol. 34, no. 18, pp. 3094–3100, 2018.
[2] A. Zielezinski, S. Vinga, J. Almeida, and W. M. Karlowski, "Alignment-free sequence comparison: Benefits, applications, and tools," *Genome Biol.*, vol. 18, no. 1, 2017, Art. no. 186.
[3] D. W. Mount, *Bioinformatics: Sequence and Genome Analysis*. Cold Spring Harbor, NY, USA: Cold Spring Harbor Laboratory Press, 2004.
[4] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *J. Mol. Biol.*, vol. 48, no. 3, pp. 443–453, 1970.
[5] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *J. Mol. Biol.*, vol. 147, no. 1, pp. 195–197, Mar. 1981.
[6] S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman, "Gapped BLAST and PSI-BLAST: A new generation of protein database search programs," *J. Nucleic Acids Res.*, vol. 25, no. 17, pp. 3389–3402, 1997.
[7] W. R. Pearson, "Rapid and sensitive sequence comparison with FASTP and FASTA," *Methods Enzymol*, vol. 183, pp. 63–98, 1990.
[8] *NVIDIA Tesla P100*. Accessed: Apr. 2016. [Online]. Available: https://www.nvidia.com/object/tesla-p100.html
[9] J. Cheng, M. Grossman, and T. McKercher, *Professional CUDA C Programming*. Indianapolis, IN, USA: Wrox, 2014.
[10] J. E. Stone, D. Gohara, and G. Shi, "OpenCL: A parallel programming standard for heterogeneous computing systems," *Comput. Sci. Eng.*, vol. 12, no. 3, pp. 66–73, 2010.
[11] A. Sodani, R. Gramunt, J. Corbal, H.-S. Kim, K. Vinod, S. Chinthamani, S. Hutsell, R. Agarwal, and Y.-C. Liu, "Knights landing: Second-generation intel Xeon Phi product," *IEEE Micro*, vol. 36, no. 2, pp. 34–46, Mar./Apr. 2016.
[12] J. Dongarra. (Jun. 2016). *Report on the Sunway Taihulight System*. [Online]. Available: http://www.netlib.org/utk/people/JackDongarra/PAPERS/sunway-report-2016.pdf
[13] H. Fu, J. Liao, J. Yang, L. Wang, Z. Song, X. Huang, C. Yang, W. Xue, F. Liu, F. Qiao, W. Zhao, X. Yin, C. Hou, C. Zhang, W. Ge, J. Zhang, Y. Wang, C. Zhou, and G. Yang, "The sunway taihulight supercomputer: System and applications," *Sci. China Inf. Sci.*, vol. 59, no. 7, 2016, Art. no. 072001.
[14] Y. Chen, K. Li, W. Yang, G. Xiao, X. Xie, and T. Li, "Performance-aware model for sparse matrix-matrix multiplication on the Sunway TaihuLight supercomputer," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 4, pp. 923–938, Apr. 2019.
[15] L. Jiang, C. Yang, Y. Ao, W. Yin, W. Ma, Q. Sun, F. Liu, R. Liu, and P. Zhang, "Towards highly efficient DGEMM on the emerging SW26010 many-core processor," in *Proc. 46th Int. Conf. Parallel Process. (ICPP)*, Aug. 2017, pp. 422–431.
[16] G. Xiao, K. Li, Y. Chen, W. He, A. Zomaya, and T. Li, "CASpMV: A customized and accelerative SpMV framework for the Sunway TaihuLight," *IEEE Trans. Parallel Distrib. Syst.*, to be published.

[17] Y. Chen, G. Xiao, and W. Yang, "Optimizing partitioned CSR-based SpGEMM on the Sunway TaihuLight," *Neural Comput. Appl.*, vol. 31, pp. 1–12, Mar. 2019. doi: 10.1007/s00521-019-04121-z.
[18] Y. Chen, K. Li, X. Fei, Z. Quan, and K. Li, "Implementation and optimization of AES algorithm on the Sunway TaihuLight," in *Proc. 17th Int. Conf. Parallel Distrib. Comput., Appl. Technol. (PDCAT)*, Dec. 2016, pp. 256–261.
[19] V. Polyanovsky, M. A. Roytberg, and V. G. Tumanyan, "Comparative analysis of the quality of a global algorithm and a local algorithm for alignment of two sequences," *Algorithms Mol. Biol.*, vol. 6, no. 1, p. 25, 2011.
[20] W. Zhou, Z. Cai, B. Lian, J. Wang, and J. Ma, "Protein database search of hybrid alignment algorithm based on GPU parallel acceleration," *J. Supercomput.*, vol. 73, no. 10, pp. 4517–4534, 2017.
[21] W. Zhou, Z. Cai, B. Lian, J. Wang, J. Ma, B. Sun, and Q. Yu, "A multi-GPU protein database search model with hybrid alignment manner on distributed GPU clusters," *Concurrency Comput., Pract. Exper.*, vol. 30, no. 18, 2018, Art. no. e4522.
[22] W. Liu, B. Schmidt, G. Voss, A. Schroder, and W. Müller-Wittig, "Bio-sequence database scanning on a GPU," in *Proc. 20th IEEE Int. Parallel Distrib. Process. Symp. (HiCOMB)*, Apr. 2006, p. 8.
[23] S. A. Manavski and G. Valle, "CUDA compatible GPU cards as efficient hardware accelerators for Smith–Waterman sequence alignment," *BMC Bioinf.*, vol. 9, Mar. 2008, Art. no. S10. doi: 10.1186/1471-2105-9-S2-S10.
[24] E. Rucci, C. Garcia, G. Botella, A. De Giusti, M. Naiouf, and M. Prieto-Matias, "Accelerating Smith–Waterman alignment of long DNA sequences with OpenCL on FPGA," in *Proc. Int. Conf. Bioinf. Biomed. Eng.*, 2017, pp. 500–511.
[25] E. Rucci, C. Garcia, G. Botella, A. De Giusti, M. Naiouf, and M. Prieto-Matias, "OSWALD: OpenCL Smith–Waterman on altera's FPGA for large protein databases," *Int. J. High Perform. Comput. Appl.*, vol. 32, no. 3, pp. 337–350, 2018.
[26] Y. Liu and B. Schmidt, "SWAPHI: Smith–Waterman protein database search on Xeon Phi coprocessors," in *Proc. IEEE 25th Int. Conf. Appl.-Specific Syst., Archit. Processors*, Jun. 2014, pp. 184–185.
[27] T. R. P. Siriwardena and D. N. Ranasinghe, "Accelerating global sequence alignment using CUDA compatible multi-core GPU," in *Proc. 5th Int. Conf. Inf. Automat. Sustainability (ICIAFs)*, Dec. 2010, pp. 201–206.
[28] M. Zhao, W.-P. Lee, E. P. Garrison, and G. T. Marth, "SSW library: An SIMD Smith–Waterman c/c++ library for use in genomic applications," *PLoS ONE*, vol. 8, no. 12, 2013, Art. no. e82138.
[29] J. Blazewicz, W. Frohmberg, M. Kierzynka, E. Pesch, and P. Wojciechowski, "Protein alignment algorithms with an efficient backtracking routine on multiple GPUs," *BMC Bioinf.*, vol. 12, no. 1, 2011, Art. no. 181.
[30] Y. Liu, W. Huang, J. Johnson, and S. Vaidya, "GPU accelerated Smith–Waterman," in *Proc. Int. Conf. Comput. Sci. (ICCS)*, 2006, pp. 188–195.
[31] A. Khajeh-Saeed, S. Poole, and J. B. Perot, "Acceleration of the Smith–Waterman algorithm using single and multiple graphics processors," *J. Comput. Phys.*, vol. 229, no. 11, pp. 4247–4258, 2010.
[32] S. Xu, Y. Xu, W. Xue, X. Shen, F. Zheng, X. Huang, and G. Yang, "Taming the 'Monster': Overcoming program optimization challenges on SW26010 through precise performance modeling," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, May 2018, pp. 763–773.

HAO ZHANG was born in Shandong, China, in 1994. He received the B.S. degree from the Shandong University of Science and Technology, China, in 2017, where he is currently pursuing the M.S. degree. His research interests include high performance computing and bioinformatics.

YOU FU was a Visiting Scholar with the karlsner institute of technology (KIT), Germany, from November 2006 to November 2007. She is currently a Full Professor with the College of Computer Science and Engineering, Shandong University of Science and Technology, China. Her research interests include high performance computing, distributed computing, and intelligent computing.

LU-BIN FENG was born in Shandong, China, in 1992. He received the B.S. degree from the Shandong University of Science and Technology, China, in 2016, where he is currently pursuing the M.S. degree. His research interests include high performance computing and bioinformatics.

YUE ZHANG was born in Shandong, China, in 1996. She received the B.S. degree from the Shandong University of Science and Technology, China, in 2018, where she is currently pursuing the M.S. degree. Her research interests include high performance computing and bioinformatics.

RONG HUA is currently an Associate Professor with the College of Computer Science and Engineering, Shandong University of Science and Technology, China. His research interests include high performance computing, artificial intelligence, and big data processing.