Future Generation Computer Systems 114 (2021) 679–691

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Highly scalable parallel genetic algorithm on Sunway many-core processors

Zhiyong Xiao [a], Xu Liu [a,b], Jingheng Xu [b,c], Qingxiao Sun [b,d], Lin Gan [b,c,*]

[a] School of Artificial Intelligence and Computer Science, Jiangnan University, Wuxi, China
[b] National Supercomputing Center in Wuxi, China
[c] Department of Computer Science and Technology, Tsinghua University, Beijing, China
[d] School of Computer Science and Engineering, Beihang University, Beijing, China

ARTICLE INFO

Article history:
Received 12 December 2019
Received in revised form 13 July 2020
Accepted 16 August 2020
Available online 28 August 2020

Keywords:
High performance computing
Genetic algorithm
Parallel optimization
Register communication
MPI communication

ABSTRACT

As a heuristic method, the genetic algorithm provides promising solutions with impressive performance benefits for large-scale problems. In this study, we propose a highly scalable hybrid parallel genetic algorithm (HPGA) based on Sunway TaihuLight Supercomputer. First, the Cellular model is presented on a thread level, so that each individual can be processed by a single computing unit which is in charge of the parallel fitness calculation, crossover, and mutation operations. The information exchange between individuals is realized by register communication. Second, the Island model is assigned to a process level, so that each process accounts for a single sub-population, and the migration among sub-populations is implemented using MPI communication. The proposed approach can fully exploit the individual diversity of the genetic algorithm and reasonably maintain the communication overhead. Based on the widely used CEC2013 benchmark, the experimental results show that the algorithm presents a sound performance in terms of both accuracy and convergence speed.

1. Introduction

With the increasing complexity of optimization problems in engineering, solving large-scale problems is challenging with traditional algorithms. Novel and sophisticated algorithmic schemes are required [1]. The genetic algorithm (GA), a stochastic search algorithm based on the principle of natural selection and recombination [2], has become a potential and promising option. GA uses genetic operators such as crossover and mutation to find the optimal chromosomes in the entire search space. Compared with traditional deterministic optimization algorithm, GA intrinsically has the advantages of parallelism and strong global optimization ability, so that it is efficient for solving combinatorial optimization problems on large-scale parallel machines [3].

In recent decades, with the fast development of the high-performance computing (HPC) systems, genetic algorithms evolved from serial to parallel and can fit well with not only traditional platforms (e.g. CPUs), but also some novel architectures (e.g. GPUs). Although those optimization problems, such as the knapsack problem [4], the shop scheduling problem [5], and the traveling salesman problem [6], are easy to be resolved using serial or original genetic algorithm, it is difficult to achieve the desired result in a short time when the objective problem becomes more complex due to huge data volume and complicated constraints. The parallel genetic algorithm is thereby attractive because of its intrinsic advantages [7] of parallelism. For example, Dorronsoro et al. [8] proposed a parallel genetic algorithm to solve the capacitive vehicle routing problem more efficiently, and Nitisiri et al. [9] reported a multi-threading technology to achieve parallel individual computing so that passengers could obtain the best railway scheduling solution in the shortest time. However, the computing power of CPUs is usually limited, and it leads to inadequate use of computing resources.

Benefiting from the development of heterogeneous multi-core supercomputers, Zhao et al. [10] proposed a parallel genetic algorithm of CPU + GPU to solve the task scheduling problem. Rathomi et al. [11] developed a two-level parallel algorithm of multi-core CPU + many-core GPU. Hou et al. [12] described an enhanced parallel genetic algorithm for hardware and software partitioning. However, none of the above algorithms considers the time consuming caused by data communication. Liu et al. [13] reduced the communication delay through the asynchronous migration strategy. However, the effectiveness is severely affected by specific problems. Huang et al. [14] proposed a fast parallel genetic programming framework by using the environment-vector-based multipopulation mechanism and the hierarchical parallel computing mechanism, but the data size of this method is small. Liu

* Corresponding author.
E-mail address: lingan@tsinghua.edu.cn (L. Gan).

https://doi.org/10.1016/j.future.2020.08.028
0167-739X/

et al. [15] designed a Hybrid Parallel Genetic Algorithm with Dynamic Migration Strategy Based on Sunway Many-Core Processor, but the convergence speed was still not ideal.

On the other hand, computing power has also been significantly increased, and we are almost reaching the Exa-scale computing era. However, the performance gap between the software and hardware is still large, and lots of computing power is wasted. Besides, the time complexity analysis of the genetic algorithm is still the main index to judge the performance of a algorithm [16]. Therefore, it is necessary to figure out efficient algorithms with good scalability to fit the architectures well.

The Sunway TaihuLight supercomputer was announced in June 2016, and it is the first system in the world that has a peak performance of 125 PFlops. Sunway TaihuLight has been widely applied in various applications, such as climate modeling [17,18], earthquake simulation [19,20], and life science [21,22].

The parallelism and scalability of GA take full advantage of heterogeneous many-core systems. We deploy GA on Sunway TaihuLight and perform a series of wise operations for it to reduce the gap between software and hardware performance, such an implementation could be a guidance for other analogous algorithms deployed on heterogeneous parallel machines.

In order to expand the individual diversity of GAs and reduce the communication overhead, we propose a large-scale parallel genetic algorithm for heterogeneous many-core processors. The major contributions include:

1. Cellular model and Island model are wisely performed on the lower and upper levels on Sunway processor, respectively, to better deal with bottlenecks from communication overhead, individual diversity, as well as premature convergence.

2. Fine-grained tuning mechanisms such as register communications are applied to further exploit the localities of the local memory, while the underlying Cellular model well implements the selection and crossover in a high parallel efficiency.

3. The hybrid parallel genetic algorithm can well handle a variety of problems, including those with large-scale data and simple fitness function, and also those with small-scale data and complex fitness function.

The remainder of this paper is organized as follows. In Section 2, we describe the related work and briefly introduce the development of genetic algorithms. Section 3 introduces the features and structure of Sunway TaihuLight and some preliminary knowledge of GA. Section 4 describes the proposed parallelization method. The parallelism implementation, the experimental results, and their analysis are in Sections 5 and 6, respectively. Finally, we conclude this paper in Section 7.

2. Related work

GA searches the optimal solution according to specific rules, and it has a straightforward executive process and strong computing power, which is widely used in complex combinatorial optimization problems.

GAs solved many optimization problems, such as the knapsack problem [4], the shop scheduling problem [5], and the traveling salesman problem [6]. In addition, prior data guidance and expansion of population size are introduced to avoid local optimum [23,24]. However, it is difficult to achieve the optimal result within the desired time for GAs when the population size, the problem constraints, and the computational complexity are increased.

The independent calculation of individuals makes GA have natural adaptability to parallelization. Parallel genetic algorithm [7] has been greatly researched and applied, and a series of new methods are proposed according to the parallel computer architecture. For example, Bernab et al. [8] used two-stage parallel strategies: Island + master–slave hybrid parallel genetic algorithm. They performed individual parallel computing on a computing platform that combines more than one hundred machines to solve the large-scale capacitated vehicle routing problem. Using the multi-core CPU parallel computing, Tansel et al. [25] proposed a highly scalable Island model, and effectively solve the NP-hard packing problem. However, the multi-core processor is not fully utilized because of the limitation of node resources and CPU computing power. To summarize, multi-CPU parallel execution would alleviate the pressure of the single CPU at the computing level. However, the CPU resources are not fully utilized because the advantages of CPU are scheduling and management abilities rather than computing.

With the development of heterogeneous multi-core supercomputers [26–29], Zhao et al. [10] proposed the parallel genetic algorithm and applied to CPU + GPU platform to solve the task scheduling problem in the physical simulation system. Under the management and coordination of CPU, the GPU performed parallel computations, effectively reduced the time cost from exponential level which was caused by the increasing data, to linear level. Further, Muhamad et al. [11] proposed a two-stage parallel algorithm, which deployed in multi-core CPU + many-core GPU architecture. One CPU processes a sub-population, and data migration between CPUs is taken by MPI, which could be theoretically extended indefinitely. SIMT creates multiple threads, and each thread processes an individual. Hou et al. [12] proposed an enhanced parallel genetic algorithm for hardware and software partitioning on Multi-Core CPU and Many-Core GPU. The execution position of each operation is specifically generated by graph division. However, the time consumption caused by data communication is not considered in the above algorithms. Liu et al. [13] proposed an asynchronous migration strategy, in which the migration parameters are dynamically adjusted by buffer communication and migration operations to reduce communication delay and overlap, however, it is still seriously affected by specific problems.

The above methods show that GA is suitable for solving large-scale optimization problems while it is affected by the data communication delay. We implement a two-stage parallel genetic algorithm for Sunway heterogeneous many-core processors. By doing this, we take the full advantages of the parallel computing capability of SW26010 heterogeneous multi-core processors and effectively reduce communication overhead.

3. Background

In this section, we first describe the structural characteristics of SW26010 heterogeneous multi-core processors which support the parallel algorithm proposed in this paper. We then introduce the process of various genetic algorithms as well as the advantages and disadvantages of them to make this study could be better understood.

3.1. Sunway TaihuLight and SW26010

The Sunway TaihuLight, as the world's leading supercomputer, is currently ranked third in the global supercomputer Top500 list.

The peak performance of Sunway system is 125 PFlops, and the sustained Linpack performance is 93 PFlops. The power efficiency is 6.05 GFlops/watt, which saving more than 60% energy compared with other computers of the same magnitude [30]. Sunway TaihuLight is the world's first supercomputer with a peak performance of more than 100 PFlops. It is also the first supercomputer built in China using domestic processors [31].

The Sunway TaihuLight is equipped with 40,960 "SW- 26010" heterogeneous many-core processors. A two-level approach is

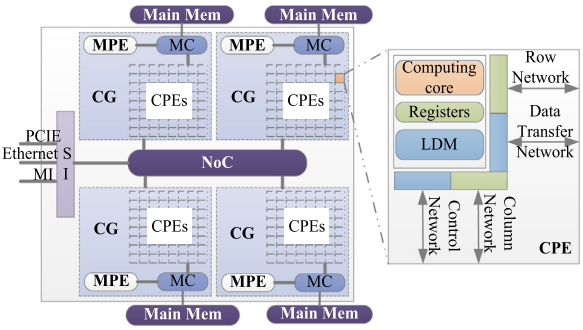


Fig. 1. The architecture of the many-core SW26010 CPU.

adopted by the machine to set up the network. Inside a supernode, the customized network board is applied to fully connect all the 256 processors. In addition, above the supernode, the central network switches are utilized to deal with the communication packets. By this strategy, all the nodes can be efficiently connected.

The basic architecture of the SW26010 is shown in Fig. 1. Each processor contains 4 computing core groups (CGs) with a total of 260 computing cores. Each core group consists of one memory controller (MC) and 65 computing cores, one 8*8 computing processing element (CPEs) and one management processing element (MPE). Both CPE and MPE are complete 64-bit RISC cores, but they handle different tasks in computing. MPE supports complete interrupt functions, memory management and out-of-order problem execution for management, task scheduling, and data communication; CPE is used for ultra-high performance and computational tasks. The MPI and CPEs in the core group realize parallel cooperation through the master and slave cores. The core groups are connected through a network on chip (NoC), and the system interface is used to connect with the off-chip system [32, 33]. The SW26010 many-core processor is different from other multi-core or many-core processors. In the memory hierarchy, MPE uses the traditional 32KB L1 instruction cache, 32KB L1 data cache and 256 KB L2 cache for instruction and data. In contrast, CPE only has 16 KB L1 instruction cache, and uses 64 KB Local Data Manager (LDM) as a fast buffer controlled by users. However, this brings more programming challenge, but it can improve overall performance. In addition, each CPE contains a control unit, a data transfer unit, an 8-row communication bus, and an 8-column communication bus. The 8-row, 8-column communication bus provides conditions for register data communication between 8 by 8 CPEs, handling CPEs data sharing, reducing memory access time, and effectively increasing bandwidth [30,34].

3.2. Genetic algorithm

The genetic algorithm is a metaheuristic algorithm based on natural genetics and natural selection principles. The core elements of the genetic search include reproduction, crossover, and mutation. Through continuous iterative evolution, the very good solutions is found throughout the search space. The specific process is shown in Fig. 2. Compared with other evolutionary algorithms, GAs have the characteristics of natural parallelism and strong global optimization ability, which is very useful for solving combinatorial optimization problems on large-scale parallel machines.

3.2.1. Traditional genetic algorithm

In the genetic algorithm, each problem parameter is encoded. Each individual is composed of gene strings, indicating a feasible

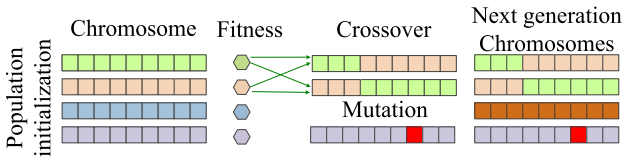


Fig. 2. Traditional genetic algorithm.

solution to the problem. Randomly generating the initial population has a significant impact on the results and efficiency of the genetic algorithm. Therefore, the initial population needs to be spread over the entire feasible solution.

Fitness is an indicator used to evaluate an individual and to assess the degree of randomization of the individual. The fitness function is often determined according to the objective function, and directly determines the optimization performance of GA by selecting individuals in population. Therefore, the fitness function is generally characterized to be continuous, non-negative, reasonable, simple, and versatile.

The selection operator selects the father and mother to inherit the gene to the next generation of individuals based on fitness. Therefore, the choice of father and mother determines the pros and cons of the offspring. In order to select individuals with advantages, the standard selection methods are roulette-wheel selection, optimal individual retention, and sorting selection.

The selected parents perform crossover with a certain probability to produce the next generation of individuals. The choice of crossover needs to preserve the individual's great genes and produce descendants with diversity and extensiveness. Conventional methods include single point crossing, two-point crossing, and uniform crossing. In order to prevent falling into local optimum, individuals are selected based on a probability to perform mutation operations on their genes, such as the random selection of mutations and inverse transformations.

After multiple generations of evolution, GA jumps out of the loop by a defined threshold or iteration number and obtains an solution whose quality is very good.

3.2.2. Master–slave model

The master–slave model is called the global parallel model. In this model, the fitness function calculation part of genetic algorithm is processed in parallel. In a simple implementation, this model uses a node as the main thread and creates some nodes as slave threads. The master thread performs the overall GA and assigns individuals to the slave nodes, which independently calculates the fitness in parallel without communicating with other slave nodes. In detail, first, the master thread initializes individuals and sends individuals to slave nodes. Then the slave threads compute the fitness function of individuals in parallel and send back to the master node. Finally, the master thread performs selection, crossover, and mutation. As a result, this model is commonly used in the situation of time-consuming fitness calculation.

3.2.3. Island model

Granularity in parallel computing is measured by the ratio of computation to communication [35]. The Island model is also called the coarse-grained model. The population is divided into multiple sub-populations (Islands), which independently perform a large amount of computation to evolve. Migration is carried out after a certain number of iterations in islands, and information exchange in an island increases population diversity and prevents premature convergence. The important factors affecting the performance of migration in the Island model are migration topology, migration scale, and migration strategy.

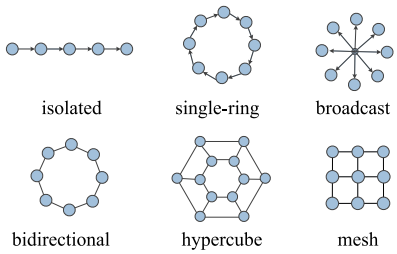


Fig. 3. Common migration topology: The migration topology is the individual migration path of each island. Common topologies include entirely isolated, single-ring, broadcast, bidirectional, hypercube, and mesh.

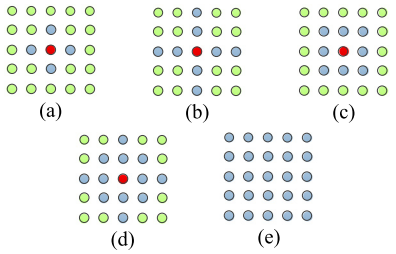


Fig. 4. Cellular model neighborhood structure: Neighborhood structure restricts the interaction between individuals. There are five commonly used domain structures, (a) four neighborhood (Von Neumann), (b) eight neighborhood, (c) Moore neighborhood, (d) diamond neighborhood, and (e) Compact neighborhood.

The migration topology is related to the parallel computer structure, which determines the migration path of individuals on the island and has a severe impact on communication events. Fig. 3 shows some commonly used topologies, such as entirely isolated, single-ring, broadcast, bidirectional, hypercube, and mesh.

The migration scale includes migration cycle and migration rate, among them; the migration cycle refers to the migration interval of communication between islands, which can be fixed or not fixed. The migration rates determine the amount of communication between islands, typically choosing a specific value or percentage of sub-populations.

The migration strategies identify migrating individuals. Generally, three strategies are commonly used to select the best individuals: (1) the optimal individual is selected directly, (2) the sub-population with high fitness value is selected proportionally, (3) the migrating individual is randomly selected. The selected best individuals then replace the worst individuals in the migration topology, and the worst individuals can be selected by similar strategies.

3.2.4. Cellular model

The sub-population in the Cellular model is divided finely. Ideally, one processor processes an individual, and the selection and crossover are performed within the domain of the individual. The neighborhood structure of the Cellular model, which includes the neighborhood topology and the neighborhood radius, affects the performance of GA. The commonly used neighborhood structures are shown in Fig. 4. In general, large-scale communication neighborhood like Fig. 4(e) is often used in small population, and hyper-ring meshes of 4 and 8 communication neighborhoods like Fig. 4(a) and (b) are often used in large-scale population.

According to the topology of the Cellular model, the computation work between communication events is relatively small in the design of the Cellular model. Also, handling a large number of communication events ensures workload balancing, allowing the Cellular model to further facilitate load balancing. However, this design also causes the model to be severely affected by communication overhead and tie down the overall speed of the algorithm [11].

From the above cases, we can see that genetic algorithms can effectively solve various NP problems. The increase of individual diversity is beneficial to the performance of genetic algorithms but also brings more communication and computing overhead. Besides, the implementation of the genetic algorithm itself has many problems, such as complicated strategy, numerous parameters, and a large amount of computation. Also, heterogeneous multi-core processors have complex structures, among which the cost of accessing memory is prohibitive. How to balance the load and control the communication overhead is the main challenge of highly scalable parallel genetic algorithm. To counter

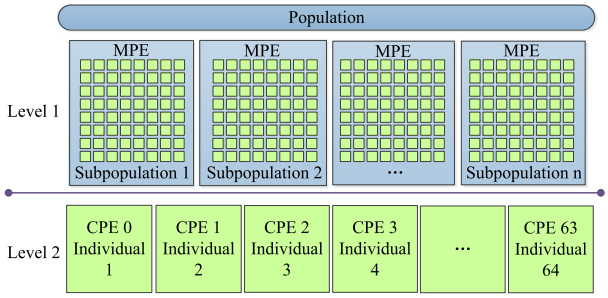


Fig. 5. Island-Cellular hybrid parallel genetic algorithm structure: In the upper layer, the population is divided into n sub-populations according to the number of processors, and one MPE processes one sub-population. In the lower layer, one MPE corresponds to 64 CPEs, and one CPE processes one individual.

the problems above, a large-scale hierarchical hybrid parallel genetic algorithm for Sunway many-core Processors is proposed. This hybrid parallel genetic algorithm makes full use of the characteristics of heterogeneous multi-core processors and realizes the communication of upper MPEs and information exchange of lower CPEs by using MPI and register communication, respectively. Theoretically speaking, the individual scale of this method is infinitely expanding, and the communication overhead is considerable.

4. Memory layout of the HPGA

For heterogeneous many-core processors, we propose a large-scale hybrid parallel genetic algorithm. In the upper layer, the Island model is deployed in MPEs, and an MPE is in charge of one sub-population evolution. Sub-populations carries out the migration by MPI. In the lower layer, the Cellular mode is deployed in CPEs, and one CPE processes an individual. The information exchange in CPEs is performed using register communication, which completes individuals selection and crossover. For the whole Genetic Algorithm, individuals can be expanded indefinitely while the communication overhead is small. In addition, the two-layer model is independent and can be selected according to specific problems. In other words, in Sunway multi-core system, there are four main cores (MPE) in each node, and each MPE is a processor to process a process. Each MPE has 64 slave cores (CPE), and each CPE processes one thread. The specific structure is shown in Fig. 5:

4.1. MPEs parallel structure

In this paper, the Island model is implemented in the upper layer. We divide the population into multiple sub-populations according to the number of MPEs, and sub-populations perform genetic manipulation in parallel. An MPE treats one sub-population

as an island that evolves independently. The algorithm of this layer is implemented by using MPI, and the structure of this part corresponds to the contents of level 1 in Fig. 5.

For the migration topology, we choose the single-ring structure. Migration realizes information transmission in islands using MPI. In detail, the master node broadcasts the size of subpopulation to all slave nodes that produces its sub-population. The sub-population completes one iteration; the best individual on the island is chosen to replace the worst one in the right domain. Finally, the master node obtains the result of all subpopulations. The equal exchange data of each island ensures MPI communication load balancing. For the SW26010 heterogeneous many-core processor, the number of processors can be fully expanded. Therefore, in the upper Island model, the population size is fully expanded while ensuring a considerable communication overhead. Specific algorithm implementation is shown in Algorithm 1:

Algorithm 1 Upper layer parallel genetic algorithm

```
Require:
    numb : the number of subpopulation
    subpopsize : the number of individuals in one subpopulation
    P_c, P_mu, P_mi : the probability of crossover, mutation and migration
    maxgen : the maximum number of iterations
    gen : the Current iterations
    fit[gen][i] : the fitness of i-th individual in the n-th generation
    parents[gen] : the parents of the n-th generation
    child[gen][i] : the i-th child of the n-th generation

Phase 1: Initialization
        Input parameter : numb, subpopsize, maxgen, P_c, P_mu, P_mi
        gen = 0
        parents[], child[][], fit[][]

Phase 2: MainLoop
        //Parallel execution on all MPEs
        for gen in range maxgen :
            for i in range subpopsize :
                Roulette Wheel Selection → parents[gen]
                Crossover --P_c--> child[gen][i], Child[gen][i + 1]
            Mutation --P_m--> child[gen][]
            Evaluate the fitness of new individuals → fit[gen][i]
            Pick the best and worst → best, worst
            //Exchange information between MPEs
            for i in range (numb):
                Migration
                subpopulation_i(best), subpopulation_{i+1}(worst)
            gen = gen + 1

Phase 3: Submit
        Submit the final subpopsize individuals to node 0
```

4.2. CPEs parallel structure

The selection and crossover of GA require the exchange of individual information within the sub-population. Considering the characteristics of the SW26010 many-core processor, to reduce the performance loss caused by communication overhead, a Cellular model is implemented on CPEs. The structure of this part corresponds to level 2 is shown in Fig. 5. The algorithm implementation of Cellular model is shown in Algorithm 2:

Considering the access bottleneck when the CPE accesses the main memory, we store the chromosome and fitness values of an individual in the LDM of the corresponding CPE. Information exchange is performed within the domain topology of CPEs using register communication. Combining with the register structure characteristics of the Sunway heterogeneous system, the Cellular model adopts the topology of the eight communication fields. In the topological area, the roulette-wheel selection is used to select parents randomly. The parents perform a crossover to produce a new individual, the individual with high fitness value is retained after a comparison between the new individual and the original one. Then, individuals randomly perform mutation. After multiple iterations, the CPEs pass the information to the MPE and execute the upper-level island model.

Algorithm 2 Lower layer parallel genetic algorithm

```
Require:
    numb : the number of subpopulation
    subpopsize : the number of individuals in one sub-population
    P_c, P_mu, P_mi : the probability of crossover, mutation and migration
    maxgen : the maximum number of iterations
    gen : the Current iterations
    fit[gen][i] : the fitness of i-th individual in the n-th generation
    parents[gen] : the parent of the nth generation
    child[gen][i] : the i-th child of the n-th generation
     old[gen][j] : the j-th individual of the n-th generation
     new[gen][j] : the j-th individual of the n-th generation

Phase 1: Initialization
        Input parameter : numb, subpopsize, maxgen, P_c, P_mu, P_mi
        gen = 0, subpopsize = 64
        parents[], child[][], fit[][]

Phase 2: MainLoop
        //Parallel execution on all MPEs
        for gen in range maxgen :
            //Parallel execution on all CPEs
            for i in range subpopsize :
                for j in range 8 :
                    Roulette Wheel Selection → parents[gen]
                    Crossover --P_c--> new[gen + 1][j]
                    Evaluate the fitness → fit[gen][i]
                    child[gen][i] = MAX(old[gen][j] : new[gen][j])
            Mutation --P_m--> child[gen][]
            Pick the best and worst → best, worst
            //Exchange information between MPEs
            for i in range (numb):
                Migration
                subpopulation_i(best), subpopulation_{i+1}(worst)
            gen = gen + 1

Phase 3: Submit
        Submit the final subpopsize individuals to node 0
```

In the SW26010 heterogeneous many-core processor, one core group contains 8 * 8 CPEs, and each CPE has a 64 KB LDM. When the data stored in CPEs are similar in horizontal and vertical directions, and the more saturated the storage is, the higher the utilization of CPEs, the higher the transmission efficiency, and the higher the algorithm performance [36]. In this paper, each individual occupies one CPE, and all 64 CPEs are used. CPEs perform the fitness assessment, crossover, and mutation in parallel. Individual information is stored in LDM, and register communication is used to exchange data between CPEs instead of accessing main memory. By doing this, the communication overhead is reduced significantly. Moreover, the cellular model has a more compact topology and makes the genetic algorithm obtain and retain excellent genes.

4.3. Register communication in cellular model

In SW26010 core processors, there are 4 row transfer registers, 4 column transfer registers, 6 row shared receive registers, and 6 column shared receive registers in each CPE, each register size is 256 bits. These registers are used as send buffers and receive buffers, this layout could minimize the impact of register communication on the core calculation. The specific structure is shown in Fig. 6. The data of source CPE and purpose CPE are passed in a producer–consumer relationship. In detail, CPE stops and waits for data when all receiver registers are empty and CPEs

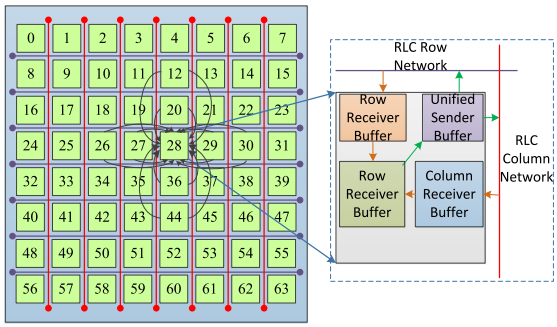


Fig. 6. Register communication structure in Sunway TaihuLight: The CPE of the SW26010 processor has four rows transmit registers, and four columns transmit registers. To avoid deadlock, the cellular model uses the eight-neighbor topology.

with a receive command. When all transmit registers are full, and CPE has a transfer command, CPE remains for data transmission in the transmit register until the idle transmit register is present. In this paper, we adopt the eight-domain topology to avoid the deadlock validly in data transmission between CPEs.

Cellular model is deployed in the lower layer to make full use of the many-core systems and reduce communication overhead. The selection and crossover within the model are realized by register communication.

Due to the limitation of memory access, the individual information is stored in the LDM of the CPE. However, the LDM space is limited and needs to be utilized reasonably. Register communication is used to perform data transmission from the same row or the same column, and indirectly achieve the purpose of accessing adjacent LDMs.

For the Sunway TaihuLight, the main core needs 94 Cycles to access the LDM, while only 10 Cycles are required for register point-to-point communication, and only 14 Cycles are required for row broadcasting and column broadcasting. The access speed is greatly improved as compared with directly accessing main memory because of the small-time consumption of register communication and accessing data from LDMs. Implementation of register communication not only utilizes the limited resources of LDM but also improves the efficiency of access.

4.4. Multi-layer structure selection

The hybrid parallel genetic algorithm can be performed separately depending on the specific application. The hybrid parallel model is used for solving the problem with large-scale data and straightforward fitness function. When dealing with small-scale data and complex fitness function models, we use the Island model in upper layer; furthermore, the Master–Slave model is deployed in CPEs.

In the master–slave model, the MPEs perform the selection, crossover, mutation, and migration; and the CPEs complete the time-consuming fitness calculation. This method is relatively simple to implement; however, the evolutionary operation is carried out within the entire population. The MPI is used for parallelization, and the communication cost increases with the increase individuals, which leads to the reduction of the acceleration ratio. Hence, this model is suitable for the situation that has a small scale and complicated fitness evaluation.

In order to verify the applicability of the model, the hybrid parallel model is tested in the CEC2013 benchmark and used to complete nonlinear prestack seismic propagation. The island model is used to optimize the performance of RTM programs. Both models perform well in accuracy and convergence speed.

5. Implementation of the HPGA

As mentioned in the previous chapter, the algorithm in this paper has a two-level decomposition structure, based on which a parallel structure is constructed. MPI, as a powerful distributed parallel tool, enables different CPUs to communicate by sending and receiving information, and then completes tasks cooperatively, which significantly reduces the running time. Communication between CPUs is expensive, so choosing a reasonable communication topology is an effective way to balance information exchange and computing allocation. Register communication is adopted to avoid repeated access to data from main memory and considerably shorten the data exchange time between cores by combining with the characteristics of the SW26010 heterogeneous multicore processor.

The large-scale hybrid parallel genetic algorithm is motivated by the optimization problem in the real-world such as seismic inversion. The algorithm is tested on the CEC2013 benchmark, and the communication load is balanced in case of the vast expansion of the data scale. Using the CEC2013 benchmark as an example, we introduce the optimization design of MPI communication, register communication, and the genetic algorithm.

5.1. Chromosome design and initial population

The individual is represented by binary code, which is appropriate for the needs of crossover and mutation. We regard the value of one dimension as a gene, which ranges between -100 and 100, and is accurate to 4 decimal places. Multiple dimensional Genes constitute a chromosome, which represents a feasible solution to the problem. There are 11 different dimensions of data in the CEC2013 benchmark, and the length of the chromosome code is based on the dimension requirement.

We use an 18-bit binary code to represent a gene. The formula for converting a binary represented by a binary string into a decimal number is as follows:

$$(b_0 b_1 \ldots b_M) = \left(\sum_{i=0}^{M} b_i \cdot 2^i \right)_{10} = X^t, \tag{1}$$

where M is the length of the gene, and the length of our gene is 18.

The formula for converting to the corresponding real number is as follows:

$$X = min + X^t \frac{(max - -min)}{2^{M+1} - 1}, \tag{2}$$

where *max* and *min* is the maximum and minimum of the gene, respectively. The *max* is -100 and the *min* is 100 in this case.

This paper randomly initializes the population according to the characteristics of the SW26010 heterogeneous multi-core processor. The upper layer implements the Island model, and One MPE is used as an island. The population information is stored in the main memory, and MPE uses MPI for communication when migration. In addition, the number of sub-populations is equal to the number of MPI processes. The Cellular model is deployed in CPEs, and one CPE processes an individual. The individual information is stored in the LDM and the CPEs exchange information through register communication. There are 64 CPEs in an MPE, and so there are 64 individuals in a subpopulation.

In this case, the data processed between each CPE is the same size, and also the data size processed by each MPE is the same, which ensures load balancing. In theory, the number of processors can be expanded infinitely to ensure the diversity of the population.

5.2. Fitness evaluation

The fitness evaluation in the algorithm is completed in the Cellular model, and the CPEs in the same sub-population complete the individual fitness assessment in parallel. The fitness function is determined based on the specific problem. There are 28 benchmark functions in CEC2013 benchmark [37], and all test functions are minimization problems defined as follows:

$$minf(x), x = [x_1, x_2, \ldots, x_D]^T, \tag{3}$$

where D is the dimension. In the CEC2013 benchmark, the orthogonal matrix generated from standard normally distributed entries by Gram–Schmidt orthonormalization. Matrix data is saved in "M_D2.txt", "M_D10.txt", "M_D30.txt", "M_D50.txt", "M_D100.txt", and each file concludes ten D*D orthogonal matrices. So, the value of D is 2, 10, 30, 50, 100. Here, the fitness function is determined by the reference benchmark function value. We set the rule, that the individual with lower the function value has the higher the fitness, to guide the evolution of the hybrid parallel genetic algorithms.

In the program of seismic inversion, the fitness function is the reciprocal of the squared difference from the target inversion curve. For the segmentation performance experiment of reverse time migration, the performance result is directly used as the fitness function.

5.3. Genetic operators

In this paper, the island model in the first parallel layer completes the migration operation through MPI communication. The primary genetic operations are accomplished in the second level of parallel, and the selection, crossover and mutation operations in the cellular model are performed in parallel by register communication. The hybrid model effectively improves the speed and performance of the genetic algorithm.

5.3.1. Register communication

For reducing communication overhead, we implement a Cellular model in the lower layer. When performing selection and crossover, we use register communication to realize information exchange between CPEs.

In the SW26010 heterogeneous many-core processor system, the CPE provides four-row transmit registers and four-column transmit registers in the design, and six-row and column shared receive registers for the transmit buffer and receive buffer respectively. These registers are 256 bits in size. The Cellular model in this paper uses the eight-neighbor topology, and it guarantees that there is no deadlock when data is transferred from the CPEs. However, the receiving CPE does not know the source of the data, and the register communication also needs to occupy the LDM resources.

In response to the above problems, we design a register communication protocol for recording the data source, and this protocol achieves the data sharing between CPEs in the eight-neighbor topology. The specific communication protocol is as follows: (see Fig. 7).

Limited by the width of the communication register, the entire packet has a width of 256 bits. The front is the data part, retaining 6 data locations, each of which occupies 32 bits and occupies 192 bits in total. "respos" occupies 16 bits, and it is the offset position of this data packet in the receiving buffer of the CPE. "src" is used to record the data sender, and it occupies 8 bits. "dst" is used to record the data receiver, and it occupies 8 bits. "cva" occupies 16 bits, which is used to indicate the valid bit of the data. "cvb" occupies 16 bits and it is not used yet.

Fig. 7. Register communication protocol: The width of the whole packet is 256 bits. The front is 6 data locations, which occupies 192 bits. "Respos" occupies 16 bits and stores the offset of the packet in the CPE receiving buffer. "Src" records the data sender with 8 bits. The "dst" uses an 8-bit recording data receiver. "Cva" records 16 bits of valid data. "Cvb" is empty, which occupying 16 bits.

According to the protocol, the individual information is transmitted in the topology domain. In a cellular model, the chromosome of each individual is stored in the LDM. When exchanging information between CPEs, we divide the data into 75 transmissions, caused by the limitation of the communication register width. When performing selection in parallel, each individual receives information from eight CPEs in the neighborhood each time. Each CPE receives 600 data transfer packets at a time to perform crossover. All individuals in sub-population perform selection and crossover in parallel by using register communication, and this method reduces communication overhead effectively.

The selection uses the roulette-wheel selection algorithm to select parents within the eight neighborhoods. The roulette-wheel selection algorithm is also called the proportional selection algorithm, in which the probability of the selected individual is proportional to its fitness. The specific operation is to calculate the fitness of each individual in the neighborhood and sum them and calculate the probability that each individual is inherited to the next generation. The specific formula is as follows:

$$P(x_i) = \frac{f(x_i)}{\sum_{j=1}^{N} f(x_j)}, \tag{4}$$

where N is the number of populations, and in this paper is the number of an individual within the neighborhood. This paper uses eight neighborhoods, but the value can be 4 to 7 when the individual is at the boundary.

Then, we calculate the cumulative probability of each individual:

$$q_i = \sum_{j=1}^{i} P(x_j). \tag{5}$$

Parents perform a single-point crossover to generate the new individual by randomly selecting chromosome locations. We calculate the fitness value of a new individual, and the value is compared with the value of the original individual, and the individual with higher fitness is retained. Besides, individuals are randomly selected for mutation at a probability of 0.005.

5.3.2. MPI communication

In the first layer, each sub-population can be deployed to different MPEs by using the characteristics that the population can be divided into several loosely dependent sub-populations. Due to the limitation of heterogeneous systems, MPE cannot communicate with each other. We use MPI communication to complete the migration. MPI (The message passing interface) is a common tool in high-performance computing, which is often used in parallel computing and other complex function operations [38,39].

Assuming that there are N_{MPI} MPIs in parallel for population decomposition, N_{subpop} is the number of subpopulations. N_i^{MPI} is the number of CPEs available for each subpopulation, and there are:

$$N_i^{MPI} = \frac{N_{MPI}}{N_{subpop}}, i = 1, 2, \ldots, N^{subpop}, \tag{6}$$

Besides, according to the algorithm structure, if the number of individuals in the sub-population is NP, $N_{i,j}^{CPE}$ is the number of CPEs used by each individual, then:

$$N_{i,j}^{CPE} = \frac{NP}{N_i^{MPI}}, j = 1, 2, \ldots, N_i^{MPI}, \quad (7)$$

According to the algorithm design in this paper, the number of MPI equals the number of sub-populations when an individual is processed from a CPE. That is to say, an MPE deals with a sub-population, and there are 64 individuals in a sub-population.

The sub-population distribution affects the execution speed of the parallel genetic algorithm. When designing the topology, we should try to ensure that the amount of data exchanged between each sub-population is the same and that the speed of each MPE processing is relatively close. In this paper, the migration topology uses a single-ring topology. After one iteration of the low-level cellular model, migration is carried out, replacing one individual in each sub-population. The best individuals in the core group exchange the worst individuals in the right neighboring core group, and the worst individuals in the core group are also exchanged by the best individuals in the left neighboring core group.

In this case, the task size and the amount of data transmitted by MPI communication are equal, which effectively guarantees load balancing.

6. Experimental result and analysis

The experiment was performed on the SW26010 heterogeneous many-core processor, using up to 2048 compute nodes. The experiment mainly consists of three parts. The first part is the performance and scalability testing of the hybrid parallel genetic algorithm on the CEC2013 benchmark. The second part is the hybrid parallel genetic algorithm for nonlinear pre-stack seismic inversion. The third part is dealing with the complex fitness objective function using the single island model.

6.1. Evaluation about the CEC2013 benchmark

In this part, the test function is the CEC2013 large-scale global optimization benchmark problems. 12 benchmark functions picked from the CEC2013 benchmark are tested experimentally, including unimodal functions, basic multimodal functions, and composition functions. We carried out a series of tests on the CEC2013 benchmark to evaluate the convergence and accuracy of HPGA.

6.1.1. Evaluation of HPGA

To evaluate the performance of our high parallel genetic algorithm, we use multiple processes to experiment on 12 benchmark functions. In this part, we discuss the convergence of hybrid parallel genetic algorithm when the number of processes increases. In this group of experiments, the crossover probability is 0.8, and the mutation probability is 0.005. The number of iterations is 2000 generations, and the results are recorded every 50 generations. There are 64 individuals in a sub-population, and processes number of 2, 16, 128, 1024, and 8192 is tested. Note that HPGA may generate different results each time, depending on the randomly selected generations, so we run it three times for each benchmark function and compute the average.

In Fig. 8, the horizontal axis is the number of iterations, and the vertical axis is the reference function value. As can be seen from the graph, with the increase of iteration times, the fitness values of the 12 benchmark functions gradually tend to converge and generate the solution whose quality is very good between 500 and 2000 generations, which proves that the algorithm in this paper is universal. In addition, this group of experiments on the number of processes 2, 16, 128, 1024 and 8192 cases, we can see from the graph that with the increase of the number of processes, that is, the population number increases, the convergence rate of the benchmark function is better, and it has better optimal performance. This also conforms to the theory that the higher the diversity of individuals in the genetic algorithm, the better the performance.

Table 1
Comparative table of convergence time and accuracy with the state of the art works.

Function	Method	Time (s)	Accuracy (10^4)
F_1	GA-DM	1789	10.033
	HPGA	633	9.946
F_2	GA-DM	2189	1.241
	HPGA	804	1.224
F_3	GA-DM	2068	0.012
	HPGA	576	0.012
F_4	GA-DM	2185	1.023
	HPGA	1031	0.915
F_5	GA-DM	2165	0.163
	HPGA	1084	0.158
F_6	GA-DM	2254	0.201
	HPGA	1051	0.170
F_7	GA-DM	2153	1.728
	HPGA	1050	1.718
F_8	GA-DM	2472	7.768
	HPGA	1064	7.674
F_9	GA-DM	2773	0.059
	HPGA	1120	0.058
F_{10}	GA-DM	2137	0.082
	HPGA	868	0.061
F_{11}	GA-DM	2374	0.397
	HPGA	890	0.366
F_{12}	GA-DM	2431	1.534
	HPGA	887	1.463

6.1.2. Comparison with state of the art works

Table 1 shows the comparison results between the proposed method and the state of the art works, the performance of the algorithm is discussed when using 8192 nodes in a 100-dimensional case. In this part of the experiment, the method(GA-DM) proposed by Liu et al. [15] is implemented on the Sunway TaihuLight. Both GA-MD and HPGA adopt 8192 MPEs, each of which has 64 CPEs. The parameters of both methods use crossover probability 0.9, mutation probability 0.005, the number of migration is one individual at a time, and the number of iterations is 2000 generations. For the 12 benchmark functions, compared with the algorithm proposed in [15], the method proposed in this paper improves the accuracy slightly, and the convergence speed has been greatly improved. Comparing the codes of the two methods, it is found that GA-DM stores data in the master process, sends the data to the slave process when assigning tasks, and sends the results back to the master process after the slave process completes the calculation. The HPGA proposed in this paper stores the data and results in each process, and the master process only needs to release a few control parameters, which greatly reduces the communication overhead.

6.1.3. Weak expansion experiment

In this experiment, six benchmark functions were selected from the CEC2013 benchmark to test the weak scalability. In this group of experiments, the crossover probability and mutation probability are 0.8 and 0.005, respectively. Experiments were carried out on 2, 16, 128, 1024, and 8192 processes, respectively. Record the entire execution time of all processes after the iteration of 2000 generations, and retain the maximum execution time.

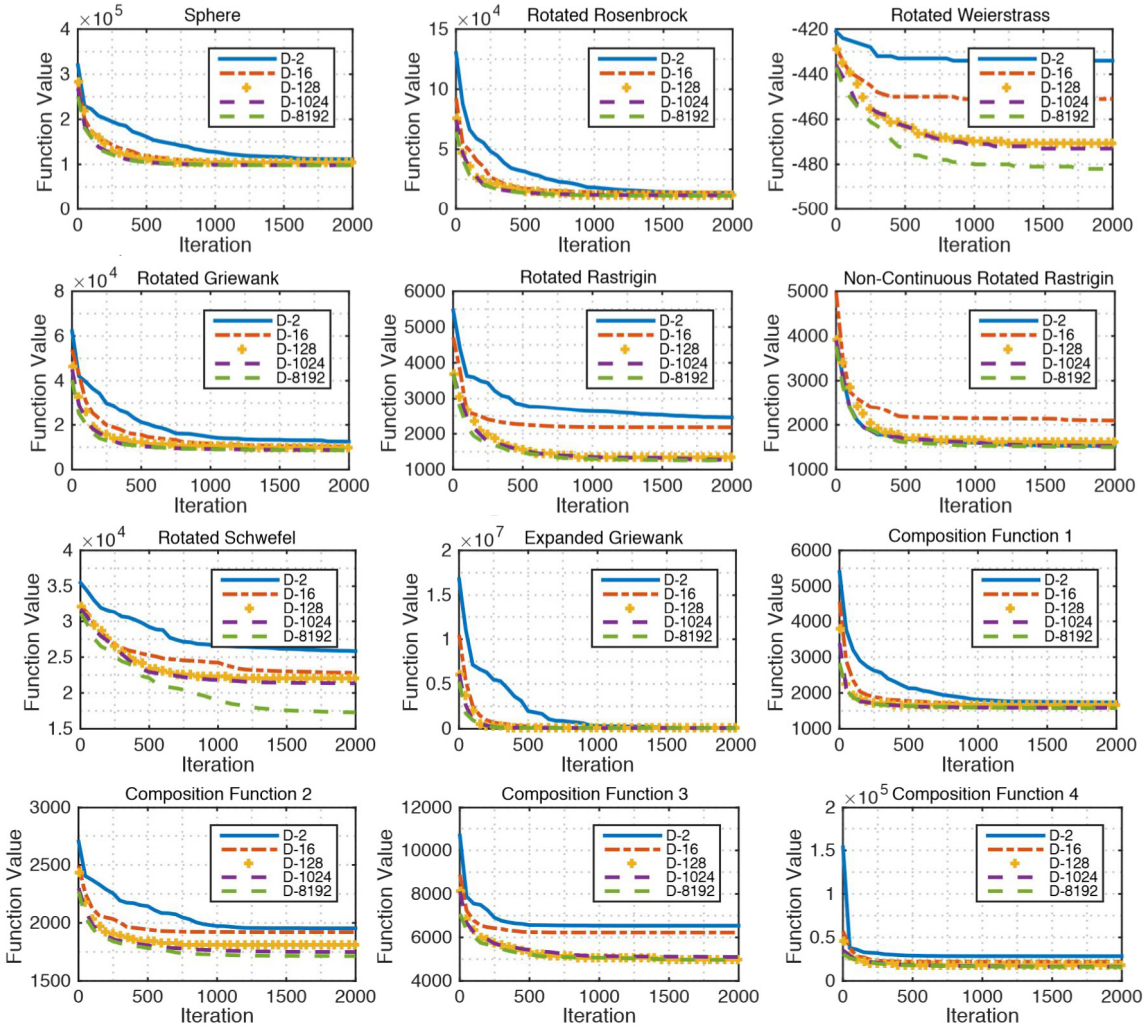


Fig. 8. Function value: This part of the experiment discussed the convergence speed and performance of HPGA as the number of iterations and processes increased. In the CEC2013 benchmark test, 12 100-dimensional benchmark functions were tested. As can be seen from the figure, the 12 benchmark functions tend to the minimum as the number of iterations increases. Also, as the number of processes increases from 2 to 8192, the convergence performance and speed of the 12 benchmark functions are improved obviously.

In Fig. 9(a), the horizontal axis is the number of processes, and the vertical axis is the normalized acceleration ratio based on the number of processes is 2. The results show that the acceleration ratio functions of the six benchmark functions maintain similar linear functions, which proves that the hybrid large-scale parallel genetic algorithm has good scalability.

In Fig. 9(b), we show the scalability comparison results of six benchmark functions under the same conditions between the single island model and the parallel model. It can be seen that when the number of processes is more than 10^3, the running time of the single island model increases significantly, while the running time of the hybrid parallel model still keeps linear growth. The graph shows that the hybrid model proposed in this paper has stronger scalability than the single island model.

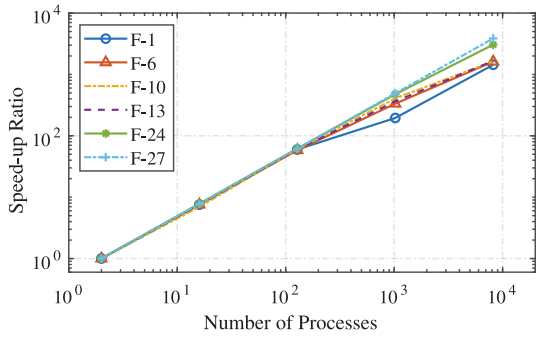
6.1.4. Multi-dimensional experiment

The experiments show the performance of the hybrid algorithm in different dimensions. The crossover probability and mutation probability of genetic algorithms are 0.8 and 0.005, respectively. There are 1024 MPI processes and 64 individuals in the sub-population. The number of iterations is 2000, and the output is recorded every ten generations. In the CEC2013 benchmark, the size can reach 11 dimensions. We conducted experiments in 10, 30, 50, and 100 dimensions (see Fig. 10).

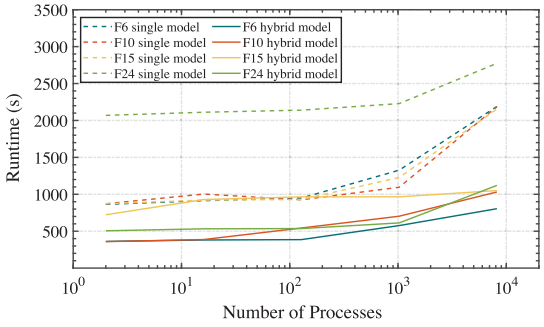
From the graph ref Fig: function-4, it can be seen that the convergence speed is the fastest in the 10-dimensional case, and the convergence rate is less than 100 generations, and the results reach the optimal value. With the increase of dimension, the search range becomes larger, and the convergence speed and results of the algorithm decrease relatively. From the graph, we can see that the algorithm still has good performance in both 30 and 50 dimensions. It can converge within 500 generations and approach optimal performance. The algorithm also shows convergence performance when tested in a 100-dimensional case, and generally converges within 1000 generations.

6.2. Seismic inversion

To further test the performance of the hybrid algorithm model, we also use this algorithm to realize the non-linear prestack seismic inversion. Seismic inversion is to infer the structure, shape, and material composition of the earth's interior based on various geophysical observation data and to calculate various geophysical parameters quantitatively. Inversion technology has been applied in many fields and is widely used in complex experimental processes of various scales and levels. For example, the estimation of underground structure and mineral deposits, the estimation of hydrocarbon accumulation parameters, the determination of



(a) Weak scalability



(b) Weak scalability comparison

Fig. 9. (a) Weak scalability: the elements in each process of the six benchmark functions are fixed to 2, 16, 128, 1024, and 8129, respectively. With the increase of the number of processes, the six benchmark functions maintain similar linear functions, showing good weak scalability; (b) Weak scalability comparison: Weak scalability comparison between the single island model and mixed parallel model, following As the number of processes increases, the weak scalability of hybrid parallel model is better than that of the single island model.

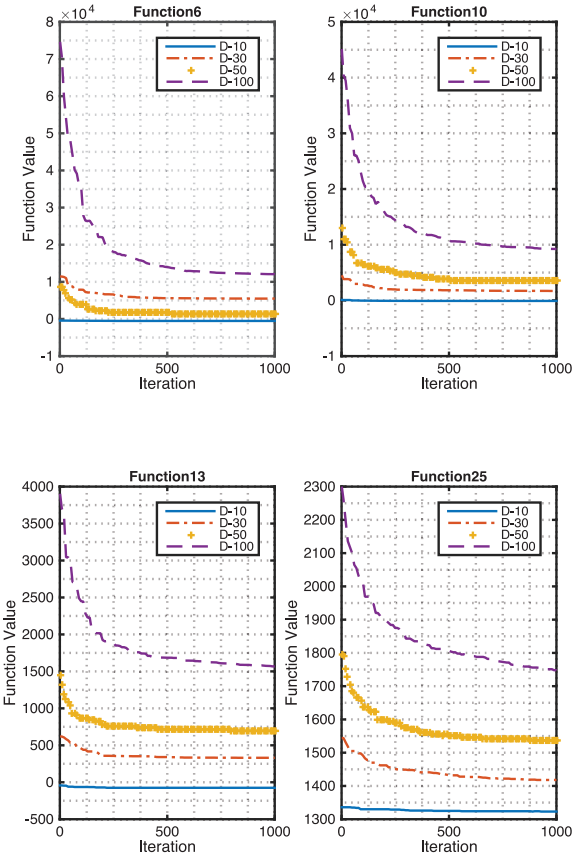


Fig. 10. Multidimensional experiments: The elements of each dimension were fixed to 10, 30, 50, and 100, respectively. Under 10-dimensional conditions, it converges within 100 generations and reaches the optimal value; under 30 and 50-dimensional conditions, it converges within 500 generations, and the final result is close to the optimal value; under 100-dimensional conditions, it can converge within 1000 generations.

focal location by wave arrival time, medical tomography, and so on. In many cases, inversion improves the conventional seismic resolution, improves the research conditions of reservoir parameters to varying degrees, obtains optimized data volume, improves the evaluation ability of resources, and puts forward favorable well location suggestions. Therefore, people's interest in seismic inversion technology is growing. Seismic inversion has become a conventional technology in oil and gas exploration and development and is becoming a key link in reservoir characterization [40, 41].

Conventional AVA seismic inversion uses the approximate formula of the Zoepritz equation. In this part, the exact equation is used to optimize the whole model space. The global optimization algorithm has better adaptability in geophysical inversion with fewer parameters. For more complicated problems, especially large-scale parameter seismic inversion such as LSRTM and FWI, gradient method, and the Newton method are generally used. In this paper, AVA seismic inversion is used to evaluate the reliability of the algorithm under the condition of small-scale computing resources. This paper considers the horizontal interface between two uniformly isotropic elastic half-spaces on the welding interface. Among them, the wave velocity, shear wave velocity and density in the upper half-space are respectively expressed as α_1, β_1 and ρ_1; the wave velocity, shear wave velocity and density in the lower half-space are respectively expressed as α_2, β_2 and ρ_2. The reflection coefficient and transmission coefficient of plane waves are given by Zoeppritz equation. In this paper, only the reflection coefficient Rpp of polypropylene is considered, the specific calculation formula is as follows [42,43]:

$$R_{pp} = \frac{Q^2 - r_4T_0T_3 + r_4T_1T_2 - (1+Q)^2\,T_0T_2 + (r_4-Q)^2\,T_1T_3 - (r_4-Q-1)^2\,T_0T_1T_2T_3}{Q^2 + r_4T_0T_3 + r_4T_1T_2 + (1+Q)^2\,T_0T_2 + (r_4-Q)^2\,T_1T_3 + (r_4-Q-1)^2\,T_0T_1T_2T_3}, \tag{8}$$

where

$$Q = 2sin^2\theta\left(r_4r_3^2 - r_2^2\right), \tag{9}$$

$$T_0 = r_0sin\theta/\sqrt{1 - r_0^2sin^2\theta}, \tag{10}$$

$$T_1 = r_1sin\theta/\sqrt{1 - r_1^2sin^2\theta}, \tag{11}$$

$$T_2 = r_2sin\theta/\sqrt{1 - r_2^2sin^2\theta}, \tag{12}$$

$$T_3 = r_3sin\theta/\sqrt{1 - r_3^2sin^2\theta}, \tag{13}$$

in which,

$$r_0 = \frac{\alpha_1}{\alpha_1}, r_1 = \frac{\alpha_2}{\alpha_1}, r_2 = \frac{\beta_1}{\alpha_1}, r_3 = \frac{\beta_2}{\alpha_1}, r_4 = \frac{\rho_2}{\alpha_1}, \tag{14}$$

where θ is the angle of incidence, and internal consistency and symmetry are maintained by $r_0 = 1$ and $T_0 = Tan\theta$.

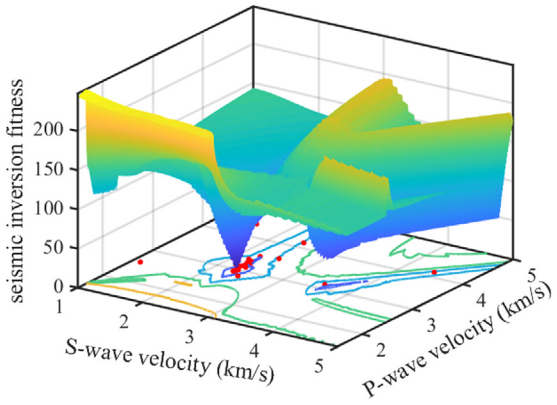


Fig. 11. Seismic inversion: The PHGA is used for seismic inversion. The fitness function is the difference between the observed value and the measured value. The graph shows that the algorithm searches in the global scope, can effectively jump out of the local optimum, and eventually converges to the better global solution.

```
The forward propagation
1:  for all time-step do
2:      for all main-grid do
3:          compute wavefield
4:      end for
5:      for all source location do
6:          add source wavelet
7:      end for
8:      apply FD8 computation
9:  end for
The back propagation
1:  for all time-step do
2:      for all main-grid do
3:          compute wavefield
4:      end for
5:      for all receivers location do
6:          add receivers data
7:      end for
8:      apply FD8 computation
9:      if time-step < nt
10:         add wavelet
11: end for
```

```
1:  for ix: d -> nx-d
2:      for iz: d -> nz-d
3:          temp[]  = FD8 computation
4:      end for
5:  end for
6:  for ix: d+1 -> nx-d
7:      for iz: d+1 -> nz-d
8:          perv[]  = FD2 computation
9:      end for
10: end for
```

Fig. 12. Demonstration of the RTM algorithm:

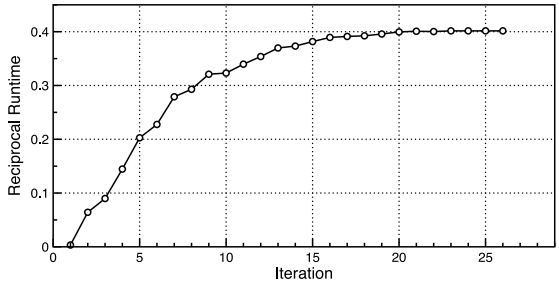


Fig. 13. RTM Segmentation optimization: As the number of iterations increases, the time of the RTM algorithm decreases gradually, and the single island model convergence achieves the optimal effect in the 21st iteration.

In this part of the experiments, the fitness function of the genetic algorithm is the difference between the observed value and the measured value, that is, the closer the perceived value is, the better the inversion effect will be. In this experiment, the crossover probability is 0.8, and the mutation probability is 0.005. The number of MPI processes is 128, and there are 64 individuals in each subpopulation. The number of iterations is 1000, and the output is recorded every ten generations. In this group of experiments, the surface S-wave velocity and P-wave velocity, as well as the surface and underground density are determined. Then the underground S-wave velocity and P-wave velocity are inverted according to the determined conditions. In Fig. 11, the individuals in GA are represented by red dots. The graph shows that the individuals in the algorithm search globally and finally find the optimal solution. Besides, it can be seen from the graph that the algorithm can also effectively jump out of the local optimum and find a better solution in global. The parameter optimization strategy proposed in this paper can effectively improve the overall calculation efficiency and is a more practical application in the field of geophysics.

6.3. Single island model experiment

According to the scale requirements of different problems, a single island model was used to optimize RTM. Seismic wave propagation is a very complex physical phenomenon, which requires a very accurate description of the complex wave equation. Reverse time migration imaging (RTM) is one of the most commonly used migration algorithms in seismic modeling [44–46]. RTM uses the two-way wave to generate topographic images, showing significant advantages in high-steep structures and under-salt imaging. The flow chart of RTM algorithm implementation in this paper is shown in Fig. 12. The performance bottleneck of reverse-time migration imaging algorithms mainly comes from memory access and computation. In the RTM algorithm, only one direction of data is stored in memory continuously. This kind of non-uniform memory access leads to a considerable time overhead. In this paper, the RTM algorithm is implemented on the Sunway TaihuLight. The performance of the RTM algorithm is seriously affected by the selection of three dimensional parameters from the core specifications. In this case, a single island model is used to automatically optimize the parameters of the three-dimensional slave core of the reverse-time migration imaging algorithm to make full use of hardware resources to reduce time overhead.

In this case, we use the binary code of a three-dimensional combination of parameters from nuclear specifications as chromosomes. Restricted by the architecture of the SW26010 heterogeneous multi-core processor and the condition of reverse time migration imaging algorithm, the range of values of three-dimensional parameters NX, NY and NZ are integers in [1,500], [1,1000] and [1,1000], among which NX must be a multiple of 4. Moreover, the product of NX, NY, and NZ does not exceed $3 * 10^7$. We use the reciprocal of the running time of the three-dimensional reverse-time migration imaging algorithm as the fitness function of genetic algorithm. In this experiment, the number of iterations is 400, the crossover probability is 0.8, and the mutation probability is 0.1. Graph 13 is the parameter optimization of RTM data block segmentation. The graph shows that with the increase of iteration times, the performance of the algorithm improves effectively and achieves the optimal effect at the 21st iteration, then keeps the stable value. It is proved that the single island model is very effective in solving parameter optimization problems with small data scale and complex fitness calculation.

7. Conclusion

With the development of processors and the complexity of various optimization problems, genetic algorithm has developed from serial to parallel, from single to hybrid. Sunway TaihuLight, as the third supercomputer in the world, deploys many high parallel and intensive computing algorithms on it. At this time,

it is very meaningful to deploy high parallel genetic algorithm on SW26010 heterogeneous multi core processors to help solve a variety of parallel optimization problems. Against this background, This paper proposes a highly scalable hybrid parallel genetic algorithm, which runs on the light of Sunway TaihuLight.

In order to implement this method, this paper implements the islanding model at the upper level and the fine-grained model at the lower level. MPI and register communication achiever are used for two levels parallelism and information exchange between MPEs and CPEs. This algorithm effectively reduces the communication overhead while fully demonstrating individual diversity. The data size processed by different CPEs is the same and the data size processed by different MPEs are the same as also, which ensure the load balancing.

To verify the performance of the algorithm, performance tests and scalability experiments are carried out on CEC2013 benchmark. The hybrid parallel algorithm model perform excellent scalability and performance on CEC2013 benchmark. The hybrid parallel algorithm is compared with a single island model and shows superiority in both performance and time. In order to further prove the practicability of the algorithm, this paper also uses the nonlinear pre-stack seismic inversion algorithm to verify, the results show that the algorithm can quickly find the global optimal solution. In addition, according to the data size and the complexity of the objective function, the single island model is also well used in parameters optimization for RTM program.

This novel design fully exploits the hardware potential of heterogeneous many-core processors, and has a good reference for the optimization and design of similar algorithms under heterogeneous many-core processors.

CRediT authorship contribution statement

Zhiyong Xiao: Conceptualization, Methodology. **Xu Liu:** Datacuration, Writing - original draft. **Jingheng Xu:** Visualization, Investigation. **Qingxiao Sun:** Writing-review and editing, Supervision. **Lin Gan:** Software, Validation. The corresponding author is responsible for ensuring that the descriptions are accurate and agreed by all authors.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors are grateful to the reviewers for their valuable comments, which greatly improved the paper. Special thanks to Wubing Wan and others for their technical help and Mengyao Sun for her writing guidance.

This work was supported by Natural Science Foundation of Jiangsu Province, China under Grant No. BK20190079, National Key Research and Development Project of China under Grant No. 2017YFE0128500 and Center for High Performance Computing and System Simulation of Pilot National Laboratory for Marine Science and Technology (Qingdao), China.

References

[1] Y. Zuo, M. Gong, J. Zeng, L. Ma, L. Jiao, Personalized recommendation based on evolutionary multi-objective optimization [research frontier], IEEE Comput. Intell. Mag. 10 (1) (2015) 52–62.

[2] J.H. Holland, Genetic algorithms, Scholarpedia 7 (12) (2012) 1482, http://dx.doi.org/10.4249/scholarpedia.1482.

[3] M. Gen, R. Cheng, Genetic Algorithms and Engineering Optimization, 1997.

[4] P.C. Chu, J.E. Beasley, A genetic algorithm for the multidimensional knapsack problem, J. Heuristics 4 (1) (1998) 63–86, http://dx.doi.org/10.1023/A:1009642405419, https://doi.org/10.1023/A:1009642405419.

[5] J.F. Gonçalves, J.J. de Magalhães Mendes, M.G.C. Resende, A hybrid genetic algorithm for the job shop scheduling problem, European J. Oper. Res. 167 (1) (2005) 77–95, http://dx.doi.org/10.1016/j.ejor.2004.03.012.

[6] L.V. Snyder, M.S. Daskin, A random-key genetic algorithm for the generalized traveling salesman problem, European J. Oper. Res. 174 (1) (2006) 38–53.

[7] E. Cantú-Paz, A survey of parallel genetic algorithms, Calc. Paralleles Res. Syst. Repar. 10 (2) (1998) 141–171.

[8] B. Dorronsoro, D. Arias, F. Luna, A.J. Nebro, E. Alba, A grid-based hybrid cellular genetic algorithm for very large scale instances of the CVRP, in: 2007 High Performance Computing & Simulation Conference (HPCS 2007), 2007, pp. 759–765.

[9] K. Nitisiri, M. Gen, H. Ohwada, A parallel multi-objective genetic algorithm with learning based mutation for railway scheduling, Comput. Ind. Eng. 130 (2019) 381–394, http://dx.doi.org/10.1016/j.cie.2019.02.035, http://www.sciencedirect.com/science/article/pii/S0360835219301214.

[10] Y. Zhao, L. Chen, G. Xie, J. Zhao, J. Ding, GPU implementation of a cellular genetic algorithm for scheduling dependent tasks of physical system simulation programs, J. Comb. Optim. 35 (1) (2018) 293–317, http://dx.doi.org/10.1007/s10878-016-0007-y.

[11] M. Rathomi, R. Pulungan, A coarse-grained parallelization of genetic algorithms, Int. J. Adv. Intell. Inf. 4 (1) (2018) 1–10, http://dx.doi.org/10.26555/ijain.v4i1.137, http://ijain.org/index.php/IJAIN/article/view/137.

[12] N. Hou, F. He, Y. Zhou, Y. Chen, X. Yan, A parallel genetic algorithm with dispersion correction for HW/SW partitioning on multi-core CPU and many-core GPU, IEEE Access 6 (2018) 883–898, http://dx.doi.org/10.1109/ACCESS.2017.2776295.

[13] Y.Y. Liu, S. Wang, A scalable parallel genetic algorithm for the generalized assignment problem, Parallel Comput. 46 (2015) 98–119, http://dx.doi.org/10.1016/j.parco.2014.04.008, http://www.sciencedirect.com/science/article/pii/S0167819114000519.

[14] Z. Huang, J. Zhong, L. Feng, Y. Mei, W. Cai, A fast parallel genetic programming framework with adaptively weighted primitives for symbolic regression, Soft Comput. (2019) 1–17.

[15] Y. Liu, R. Zhao, K. Zheng, S. Wang, Y. Liu, H. Shen, Q. Zhou, A hybrid parallel genetic algorithm with dynamic migration strategy based on sunway many-core processor, in: 2017 IEEE 19th International Conference on High Performance Computing and Communications Workshops (HPCCWS), IEEE, 2017, pp. 9–15.

[16] P.S. Oliveto, C. Witt, Improved time complexity analysis of the simple genetic algorithm, Theoret. Comput. Sci. 605 (2015) 21–41.

[17] L. Gan, H. Fu, C. Yang, W. Luk, W. Xue, O. Mencer, X. Huang, G. Yang, A highly-efficient and green data flow engine for solving euler atmospheric equations, in: 2014 24th International Conference on Field Programmable Logic and Applications (FPL), IEEE, 2014, pp. 1–6.

[18] L. Gan, H. Fu, W. Luk, C. Yang, W. Xue, G. Yang, Solving mesoscale atmospheric dynamics using a reconfigurable dataflow architecture, IEEE Micro 37 (4) (2017) 40–50.

[19] H. Fu, L. Gan, R.G. Clapp, H. Ruan, O. Pell, O. Mencer, M. Flynn, X. Huang, G. Yang, Scaling reverse time migration performance through reconfigurable dataflow engines, IEEE Micro 34 (1) (2013) 30–40.

[20] H. Fu, C. He, B. Chen, Z. Yin, Z. Zhang, W. Zhang, T. Zhang, W. Xue, W. Liu, W. Yin, et al., 18.9-pflops nonlinear earthquake simulation on sunway taihulight: enabling depiction of 18-Hz and 8-meter scenarios, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, vol. 2, ACM, 2017.

[21] B. Chen, H. Fu, Y. Wei, C. He, W. Zhang, Y. Li, W. Wan, W. Zhang, L. Gan, W. Zhang, Z. Zhang, G. Yang, X. Chen, Simulating the Wenchuan earthquake with accurate surface topography on Sunway TaihuLight, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis, SC 2018, Dallas, TX, USA, November 11-16, 2018, pp. 517–528.

[22] W. Xue, C. Yang, H. Fu, X. Wang, Y. Xu, L. Gan, Y. Lu, X. Zhu, Enabling and scaling a global shallow-water atmospheric model on tianhe-2, in: 2014 IEEE 28th International Parallel and Distributed Processing Symposium, IEEE, 2014, pp. 745–754.

[23] A. Rezoug, M. Bader-El-Den, D. Boughaci, Guided genetic algorithm for the multidimensional knapsack problem, Memetic Comput. 10 (1) (2018) 29–42, http://dx.doi.org/10.1007/s12293-017-0232-7.

[24] A.J. Umbarkar, M.S. Joshi, W. Hong, Multithreaded parallel dual population genetic algorithm (MPDPGA) for unconstrained function optimizations on multi-core system, Appl. Math. Comput. 243 (2014) 936–949, http://dx.doi.org/10.1016/j.amc.2014.06.033.

[25] T. Dökeroglu, A. Cosar, Optimization of one-dimensional bin packing problem with island parallel grouping genetic algorithms, Comput. Ind. Eng. 75 (2014) 176–186, http://dx.doi.org/10.1016/j.cie.2014.06.002.

[26] J. Luo, D.E. Baz, A survey on parallel genetic algorithms for shop scheduling problems, in: 2018 IEEE International Parallel and Distributed Processing Symposium Workshops, IPDPS Workshops 2018, Vancouver, BC, Canada, May 21-25, 2018, 2018, pp. 629–636, http://dx.doi.org/10.1109/IPDPSW.2018.00103.
[27] J.R. Cheng, M. Gen, Accelerating genetic algorithms with GPU computing: A selective overview, Comput. Ind. Eng. 128 (2019) 514–525, http://dx.doi.org/10.1016/j.cie.2018.12.067.
[28] L. Zheng, Y. Lu, M. Guo, S. Guo, C. Xu, Architecture-based design and optimization of genetic algorithms on multi- and many-core systems, Future Gener. Comput. Syst. 38 (2014) 75–91, http://dx.doi.org/10.1016/j.future.2013.09.029.
[29] P. Pospichal, J. Jaros, Gpu-based acceleration of the genetic algorithm, GECCO competition (2009).
[30] 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, G. Yang, The sunway taihulight supercomputer: system and applications, Sci. China Inf. Sci. 59 (7) (2016) 072001, http://dx.doi.org/10.1007/s11432-016-5588-7.
[31] L. Gan, H. Fu, W. Xue, Y. Xu, C. Yang, X. Wang, Z. Lv, Y. You, G. Yang, K. Ou, Scaling and analyzing the stencil performance on multi-core and many-core architectures, in: 2014 20th IEEE International Conference on Parallel and Distributed Systems (ICPADS), IEEE, 2014, pp. 103–110.
[32] C. Yang, W. Xue, H. Fu, H. You, X. Wang, Y. Ao, F. Liu, L. Gan, P. Xu, L. Wang, et al., 10M-Core scalable fully-implicit solver for nonhydrostatic atmospheric dynamics, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE Press, 2016, p. 6.
[33] L. Li, T. Yu, W. Zhao, H. Fu, C. Wang, L. Tan, G. Yang, J. Thomson, Large-scale hierarchical k-means for heterogeneous many-core supercomputers, in: SC18: International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE, 2018, pp. 160–170.
[34] H. Fu, J. Liao, N. Ding, X. Duan, L. Gan, Y. Liang, X. Wang, J. Yang, Y. Zheng, W. Liu, et al., Redesigning CAM-SE for peta-scale climate modeling performance and ultra-high resolution on sunway taihulight, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, ACM, 2017, pp. 1–12.
[35] B. Barney, et al., Introduction to parallel computing, Lawrence Livermore Natl. Lab. 6 (13) (2010) 10.
[36] S. Xu, Y. Xu, W. Xue, X. Shen, F. Zheng, X. Huang, G. Yang, Taming the" Monster": Overcoming program optimization challenges on SW26010 through precise performance modeling, in: 2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS), IEEE, 2018, pp. 763–773.
[37] J. Liang, B. Qu, P. Suganthan, A.G. Hernández-Díaz, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Technical Report, 201212, 34, 2013, pp. 281–295.
[38] W. Gropp, W.D. Gropp, A.D.F.E.E. Lusk, E. Lusk, A. Skjellum, Using MPI: Portable Parallel Programming with the Message-Passing Interface, vol. 1, MIT press, 1999.
[39] W. Gropp, R. Thakur, E. Lusk, Using MPI-2: Advanced Features of the Message Passing Interface, MIT press, 1999.
[40] G.T. Schuster, Seismic Inversion, Society of Exploration Geophysicists, 2017.
[41] Y. Wang, Seismic Inversion: Theory and Applications, John Wiley & Sons, 2016.
[42] X. Zhu, G.A. McMechan, Elastic inversion of near-and postcritical reflections using phase variation with angle, Geophysics 77 (4) (2012) R149–R159.
[43] B. Ursin, E. Tjåland, The information content of the elastic reflection matrix, Geophys. J. Int. 125 (1) (1996) 214–228.
[44] M. Araya-Polo, J. Cabezas, M. Hanzich, M. Pericas, F. Rubio, I. Gelado, M. Shafiq, E. Morancho, N. Navarro, E. Ayguade, et al., Assessing accelerator-based HPC reverse time migration, IEEE Trans. Parallel Distrib. Syst. 22 (1) (2010) 147–162.
[45] F. Ortigosa, Q. Liao, A. Guitton, W. Cai, Speeding up RTM velocity model building beyond algorithmics, in: SEG Technical Program Expanded Abstracts 2008, Society of Exploration Geophysicists, 2008, pp. 3219–3223.
[46] J. Xu, H. Fu, W. Shi, L. Gan, Y. Li, W. Luk, G. Yang, Performance tuning and analysis for stencil-based applications on POWER8 processor, ACM Trans. Archit. Code Opt. 15 (4) (2018) 1–25.

Zhiyong Xiao received his B.S. degree from Shandong University of China in 2008, and received the Ph.D. degree in Optics, Physics and Image Processing from the Ecole Central de Marseille, France in 2014. He was an assistant research fellow at Fresnel Institute, CNRS France. He is an associate professor at Jiangnan University now. His research interests include parallel computing, machine learning and artificial intelligence.

Xu Liu received the Master degree from Jiangnan University. She is pursuing the Ph.D. degree in Jiangnan University. Her research interests include high parallel computing and evolutionary algorithm.

Jingheng Xu is pursuing the Ph.D. degree in School of Computer Science and Engineering, Tsinghua University. He has participated in many state funded research projects and published several papers in international journals and conferences. His research interests include high parallel computing and evolutionary algorithm.

Qingxiao Sun is pursuing the Ph.D. degree in School of Computer Science and Engineering, Beihang University. He is currently working on GPU hardware extension and performance optimization. His research interests include computer architecture, HPC and deep learning.

Lin Gan received the Ph.D. from Tsinghua University. He is an assistant professor in School of Computer Science and Engineering, Tsinghua University. He has participated in many state funded research projects and published several papers in international journals and conferences. His current research areas include parallel computing and machine learning.