J. Ocean Univ. China (Oceanic and Coastal Sea Research)
https://doi.org/10.1007/s11802-022-4839-z
ISSN 1672-5182, 2022 21 (3): 737-747
http://www.ouc.edu.cn/xbywb/
E-mail:xbywb@ouc.edu.cn

Parallel Extraction of Marine Targets Applying OIDA Architecture

LIU Lin[1)], LI Wanwu[1), *], ZHANG Jixian[2)], SUN Yi[1)], and CUI Yumeng[1)]

1) *College of Geodesy and Geomatic*, *Shandong University of Science and Technology*, *Qingdao* 266590, *China*
2) *National Quality Inspection and Testing Center for Surveying and Mapping Products*, *Beijing* 100830, *China*

(Received November 16, 2020; revised December 24, 2020; accepted January 13, 2021)

Abstract Computing resources are one of the key factors restricting the extraction of marine targets by using deep learning. In order to increase computing speed and shorten the computing time, parallel distributed architecture is adopted to extract marine targets. The advantages of two distributed architectures, Parameter Server and Ring-allreduce architecture, are combined to design a parallel distributed architecture suitable for deep learning−Optimal Interleaved Distributed Architecture (OIDA). Three marine target extraction methods including OTD_StErf, OTD_Loglogistic and OTD_Sgmloglog are used to test OIDA, and a total of 18 experiments in 3 categories are carried out. The results show that OIDA architecture can meet the timeliness requirements of marine target extraction. The average speed of target parallel extraction with single-machine 8-core CPU is 5.75 times faster than that of single-machine single-core CPU, and the average speed with 5-machine 40-core CPU is 20.75 times faster.

Key words parallel computing; distributed architecture; deep learning; target extraction; PolSAR image

1 Introduction

The effect of marine target detection based on deep learning (DL) largely depends on the amount of training data and training scale. Therefore, with the help of parallel distributed architecture for model training, the training time of deep learning is reduced, which is beneficial to optimize the neural network model rapidly (Goyal *et al*, 2017), thus improving the speed and accuracy of marine target detection.

Parallel architecture has been applied to related researches of target detection and extraction, and has achieved certain results. Parallel architecture and parallel tempering algorithm were used to conduct multi-target tracking, and the results show that it can make full use of parallel computing capabilities of GPU and improve the computational efficiency of the algorithm (Wen, 2017). In order to solve the timeliness problem of video processing, parallel architectures were designed to detect moving targets in the videos, which significantly improved the computing speed (Peng *et al*., 2014; Lou *et al*., 2016; Zeng, 2017). Ling *et al.* (2016) proposed a parallel algorithm using sparse pulse-coupled neural network to detect moving targets based on Nvidia compute unified device architecture (CUDA), which improved the computational efficiency of the algorithm. Zhang (2017) changed traditional image processing algorithm based on CPU serial, and adopted GPU parallel algorithm to realize ship target detection in optical images. You (2016) derived fast detection algorithm for abnormal targets, proposed a parallel processing method under GPU architecture, and accelerated the calculation of hyperspectral data through CUDA, up to 33.2 times (Li, 2015). Different distributed architectures have been proposed to improve image processing speed (Hu *et al*., 2016; Quirita *et al*., 2016; Wu *et al*., 2016; Chen *et al*., 2017; Ye *et al*., 2017; Huang *et al*., 2018). And the parallel algorithm was enhanced from different angles to improve its applicability in deep learning (Aytekin *et al*., 2016; Mamidala *et al*., 2018; Cheng and Xu, 2019; Shen *et al*., 2019; Liu *et al*., 2020). Alqahtani and Demirbas (2019) established three parallel models of different system architectures including Parameter Server (PS), Peer-to-Peer, and Ring Allreduce (RA). Thao Nguyen *et al*. (2019) proposed two hierarchical distributed memory multi-lead AllReduce algorithms for GPU accelerated clusters optimization, and evaluated it on the discrete-event simulation simgrid. Bouzidi (2019) used MapReduce model to realize the parallel distributed algorithm based on the computing engine SPARK.

On the basis of existing research results, the PS architecture and the RA architecture are synthesized and a more efficient parallel distributed architecture is designed according to the characteristics of the polarized Synthetic Aperture Radar (SAR) image and its deep learning data set. A single computer uses multi-core CPU/GPU to achieve parallel computing, while multiple computers pre-load model training parameters through the parameter server to achieve uninterrupted distributed computing. In this way, parallel experiments are conducted on multiple extraction models of marine targets to test the performance of Optimal In-

* Corresponding author. E-mail: liwanwuqd@126.com

terleaved Distributed Architecture (OIDA).

2 Proposed Architecture

2.1 Parallel Distributed Architecture

Parallel distributed architectures mainly include two types: PS architecture and RA architecture.

In PS architecture, the nodes of cluster are divided into two categories: PS and Worker. PS stores the parameters of the model, and Worker calculates the gradient of the parameters. In each iterative process, Worker obtains the parameters from PS, and then returns the calculated gradient to PS. PS aggregates the gradients returned from Worker, then updates the parameters and passes the new parameters to Worker. PS architecture is the most commonly used distributed training architecture for DL. PS architecture using synchronous SGD is shown in Fig.1.

The devices of ring-allreduce architecture are all Workers and form a ring, as shown in Fig.2, and there is no central node to aggregate the gradients calculated by all Workers. In the iterative process, each Worker calculates the gradient after completing its own mini-batch training, and passes it to the next Worker in the ring. Worker also receives the gradient passed by the previous Worker at the same time. Each Worker updates the model parameters after receiving the gradients of all Workers except for itself. Compared with PS architecture, ring-allreduce architecture is bandwidth-optimized, which can make full use of the characteristics of BP algorithm to reduce training time.

2.2 OIDA Architecture

The PS and RA architectures are integrated in the paper, and on this basis OIDA is designed, as shown in Fig.3. Its working principle is as follows: A single computer makes full use of multi-core CPU and GPU to achieve parallel computing, while multiple computers pre-load intermediate training parameters of model through the PS to achieve uninterrupted distributed computing. In an iterative process, each Worker of the child node calculates the gradient after completing its own batch training, passes it to PServer, and reads the model from the model folder to continue training. The PServer of the child node passes the gradient and other parameters calculated by Worker at this node to the central node chief, receives the model parameters of the central node at the same time, updates the model in the model folder of the node for Worker to call, and passes the model parameters to the upper and lower nodes connected to it. The upper and lower nodes check the model parameters from the child nodes and compare with the model parameters of their own nodes to determine whether to update the models. PServer in the central node chief monitors and receives the model parameters passed by each child Node, passes them to Worker_DS in central node to test and evaluate model parameters, and updates the model parameters in the model folder for the child nodes to call. For OIDA deployment, first a cluster is created then job

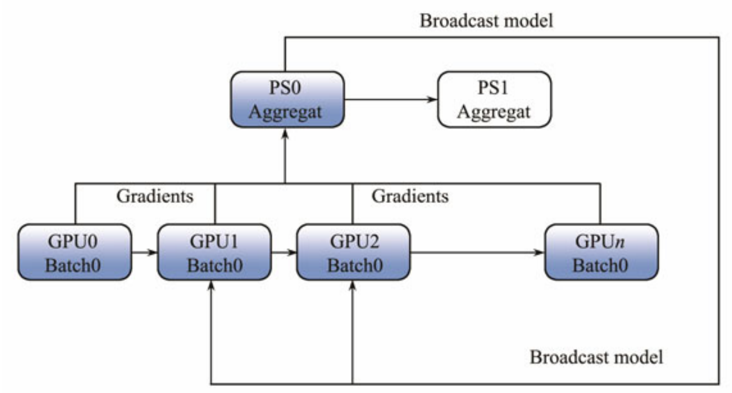

Fig.1 Synchronous SGD training method in PS architecture.

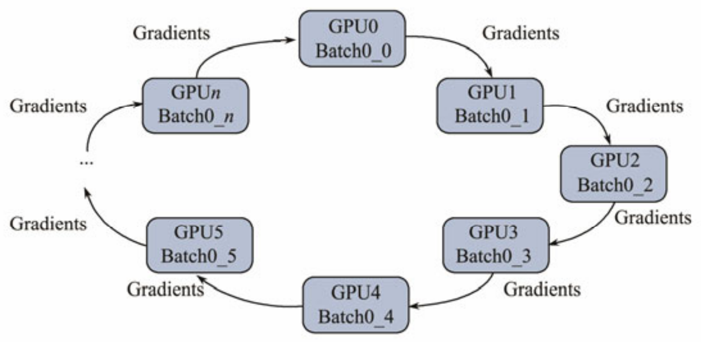

Fig.2 Ring-allreduce architecture.

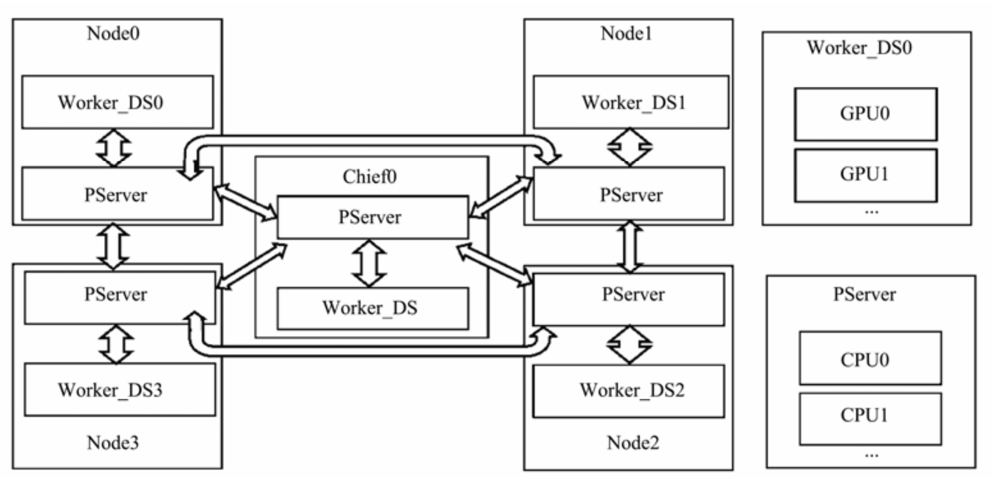

Fig.3 OIDA architecture.

and task are assigned. For each task, host addresses are assigned, and a server is created. Cluster must be imported when sever is created so that each server knows which hosts are included in the cluster it is in, and then servers can communicate each other. Servers must be created on their own host. Once all servers are created on their respective hosts, the entire cluster is constructed, and servers between clusters can communicate with each other. Each Server contains two components: Master and Worker. Master can mainly provide remote access (RPC protocol) to each device in the cluster. At the same time, it serves as a target for creating TF. Session. Worker can execute calculation subgraphs with local equipments.

3 Models and Methods

3.1 DL Model Construction

A DL model OceanTDA9 is constructed to detect marine targets in this paper. OceanTDA9 contains 4 convolutional layers, 1 convolutional group and 3 fully connected layers, as shown in Fig.4. The first 4 convolution forms are same, and each group is as Convolution2D-relu-Dropout-Maxpool. The organization form of the intermediate convolution group is as (Convolution2D-relu-Dropout)*2-Maxpool. The last three are fully connected dense layers, among which the first two sets of dense are as Dense-relu-Dropout, and the last fully connected dense layer only has dense. The kernel of each convolutional layer is a 3×3 small convolution kernel, and Dropout is set to 0.2. All pooling adopts maxpooling with a core size of 2×2 and a sliding step stride of 2.

3.2 Target Extraction Method

The marine target extraction methods are proposed in this paper, including Constant False Alarm Rate (CFAR) method based on the initial detection (OTD_StErf), marine target extraction method based on loglogistic (OTD_Loglogistic), the method based on Adjoint Covariance Correction Model (OTD_Sgmloglog) for complex sea conditions.

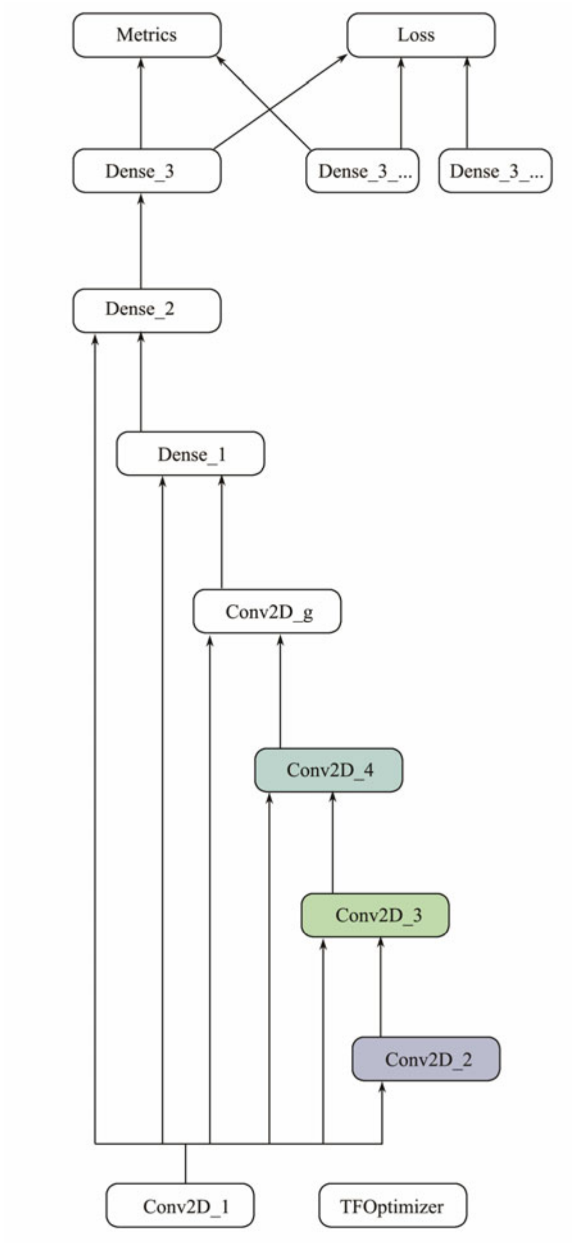

Fig.4 DL model OceanTDA9 for marine target detection.

The OTD_StErf method is to first adopt DL model OceanTDA9 for initial detection, and save the obtained suspected targets. 3×3 images, which contains 28×28 pixel, are constructed with the suspected target points as the center in turn. Then, the two-parameter CFAR method was used to perform n tests on the 3×3 window. The value of t is calculated by Eq. (1), and the critical pixel value X_{tb} is found by the target detection model (Eq. (2)). The pixels whose values are larger than the critical value in the 3×3 image window are determined as suspected targets.

$$p_{fa} = \frac{1}{2} - \frac{1}{2} erf\left(\frac{t}{\sqrt{2}}\right), \tag{1}$$

$$X_{tb} = m_b + \sigma_b \times t . \tag{2}$$

OTD_Loglogistic method first adopts the DL model OceanTDA9 for initial detection, and constructs a 3×3 image window with the obtained suspected target as the center, and saves the pixels that are not the targets in the window to the array x. Calculate the frequency of each pixel in x and draw a histogram, extract the effective pixels and corresponding frequencies from the smallest pixel to the largest pixel in the histogram and save them to the array $x2$ and $y2$ as fitting data. Call loglogistic probability density function $f(x, \alpha, \beta)$ (Eq. (3)) to fit $x2$ and $y2$, obtain the parameter values of (α, β) for the fitted curve, and evaluate the goodness of fit such as Chi-square and absolute value error.

$$f(x,\alpha,\beta) = \frac{(\beta/\alpha)(x/\alpha)^{\beta-1}}{\left(1+(x/\alpha)^{\beta}\right)^2}. \tag{3}$$

The principle of the OTD_Sgmloglog ocean target extraction method is similar to the above two methods, except that the calling model is different. It calls Adjoint Covariance Correction Model $f(x, \alpha, \beta)$ constructed by the author to fit $x2$, $y2$, as shown in Eq. (4), where $x\in[0, 255]$, $\alpha>0$, $\beta>0$, the modified parameter $\varepsilon\in[0, 1]$, and σ is the standard deviation of the fitted curve. Finally, the parameter values of α, β and ε of the fitted curve are calculated, and the parameters indicating the goodness of fit such as Chi-square and absolute value error are calculated.

$$f(x,\alpha,\beta,\varepsilon) = f(x,\alpha,\beta) + \varepsilon\sigma = \frac{\frac{\beta}{\alpha}\left(\frac{x}{\alpha}\right)^{\beta-1}}{\left(1+\left(\frac{x}{\alpha}\right)^{\beta}\right)^2} + \frac{2\pi\varepsilon\alpha^2}{\beta\cdot\sin\frac{2\pi}{\beta}}\left(1-\frac{2\pi}{\beta\cdot\sin\frac{2\pi}{\beta}}\right). \tag{4}$$

4 Parallel Experiment

In the paper, the experimental area for marine target parallel extraction is in the Bohai Sea, located in 37˚07΄–40˚56΄N, 117˚33΄–122˚08΄E, and the polarized SAR data from Sentinel-1 in this area are used as the experimental data. Through experimental comparison, it is found that the VV (vertical transmission, vertical reception) polarized SAR image from Sentinel-1 IW (Interferometric Wide Swath Mode) is more suitable for marine target extraction. So VV polarized SAR image is selected for experiments. After the VV polarized SAR image is preprocessed by the procedure such as split, calibration, speckle filter, multi-looking, terrain correction, resampling and data conversion, the marine target detection data set is obtained, which is used as the data source of parallel distributed experiments for marine target extraction.

The marine target detection DL model OceanTDA9 is adopted to learn and train the preprocessed PolSAR dataset according to the above parallel distributed architectures, and the parameters of the neural network model for marine target detection are obtained. The model parameters are called to detect the research area, and the obtained suspected targets are saved in the library files. The suspected targets detected in the area with the range of 39˚52΄–40˚13΄N, 120˚44΄–121˚03΄E are marked by the magenta boxes in Fig.5, where the upper left corner is assigned as 0 rows and 0 columns, corresponding to 40˚13΄N and 120˚44΄E, the lower right corner is at 2715 rows and 2519 columns, corresponding to 39˚52΄N and 121˚03΄E. Similar situations in this paper will not be explained again. The OIDA architecture designed in this paper is adopted to do experiments on three methods, OTD_StErf, OTD_Loglogistic, and OTD_Sgmloglog, and extract distributed targets in the research area.

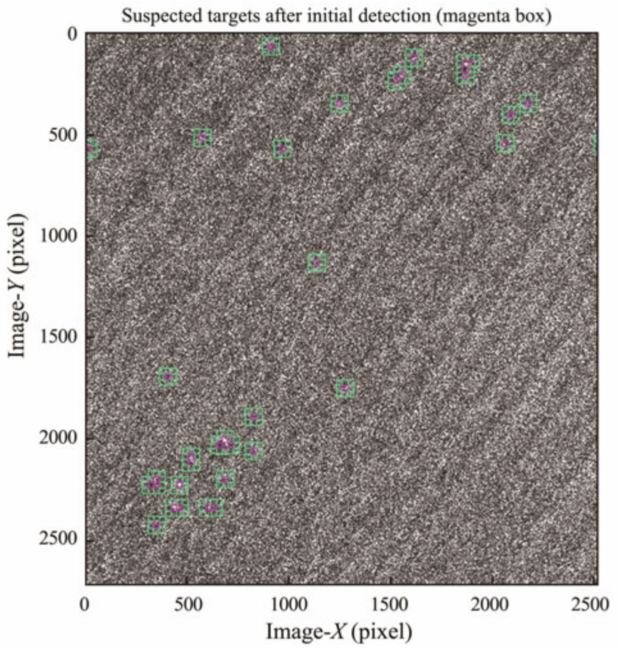

Fig.5 Suspected targets after initial detection (magenta box).

4.1 Experiment Design

The overall process of marine target extraction by using parallel distributed architecture is shown in Fig.6. Each computer in the cluster is configured with a 4-core 8-thread CPU with 32GB of memory, and two GPUs with 2GB of video memory. All computers form a Gigabit Ethernet. Each computer in the cluster is configured with network parameter models and PolSAR image data after DL to ensure that the necessary dynamic data with extremely low redundancy are transmitted in the network.

After the software and hardware environment is set up, start the service, and designate a node as chief, which is responsible for managing each node, coordinating the operation between each node, and completing the distribution of tasks, the collection of intermediate results and the integration and visualization of results. After the task of chief is received by other Workers, the corresponding function is initialized according to the task requirements. The target fitting parameter extraction function is used to complete the clustering of suspected targets, extract the position parameters such as the center coordinates and inclination of the suspected targets, and extract the shape parameters such as length, width, and area, then use a straight line to fit the central axis and use an ellipse to fit the shape of the suspected target. Three target parameter extraction methods including OTD_StErf method, OTD_Loglogistic method, and OTD_Sgmloglog method were used.

After each Worker node in the cluster complete initial detection task assigned by chief, the suspected targets in the unit of DL sub-image (28 pixels×28 pixels) are saved to the corresponding folder, and chief is notified that the tasks have been completed. According to the load condition of Workers in the cluster, the collected suspected targets are grouped and distributed to the corresponding Workers by chief. After the task of chief is received by Worker nodes, the target parameters are called to extract the function. According to the pre-set maximum and minimum length and aspect ratio, the target parameters that meet the conditions are extracted and saved to the corresponding folder, and Chief is notified that the tasks have been completed. The target parameters extracted by each Worker in the cluster are collected by chief and dynamically visualized after integration.

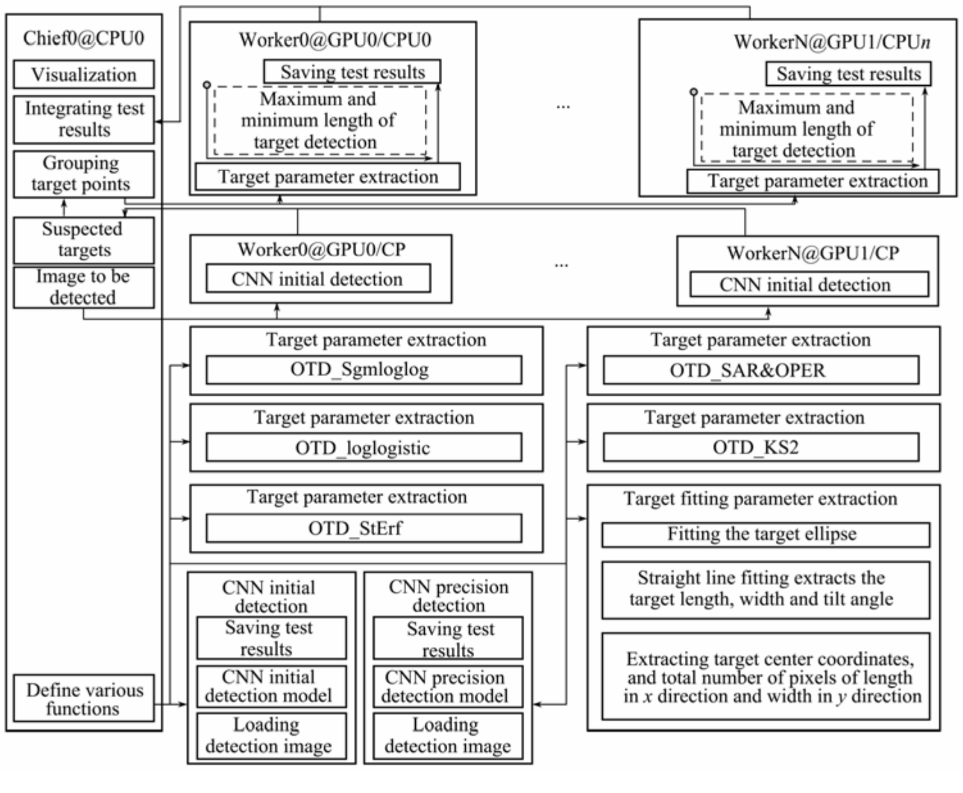
Chief0@CPU0
Visualization
Integrating test results
Grouping target points
Suspected targets
Image to be detected
Define various functions
Worker0@GPU0/CPU0
Saving test results
Maximum and minimum length of target detection
Target parameter extraction
...
WorkerN@GPU1/CPUn
Saving test results
Maximum and minimum length of target detection
Target parameter extraction
Worker0@GPU0/CP
CNN initial detection
WorkerN@GPU1/CP
CNN initial detection
Target parameter extraction
OTD_Sgmloglog
Target parameter extraction
OTD_loglogistic
Target parameter extraction
OTD_StErf
Target parameter extraction
OTD_SAR&OPER
Target parameter extraction
OTD_KS2
Target fitting parameter extraction
Fitting the target ellipse
Straight line fitting extracts the target length, width and tilt angle
Extracting target center coordinates, and total number of pixels of length in x direction and width in y direction
CNN initial detection
Saving test results
CNN initial detection model
Loading detection image
CNN precision detection
Saving test results
CNN precision detection model
Loading detection image

Fig.6 Marine target extraction procedure by using parallel distributed architecture.

4.2 Experiment on OTD_StErf

For comparison, single-machine single-core device is used to extract marine target parameters for 35 suspected targets detected initially by OTD_StErf. It takes 381.45 s to extract 22 targets. The time sequence is shown in Fig.7, and each target extraction task takes approximately 6 s. In order to ensure the display effect and data storage, each target is set to delay 5 s after displayed and then turned off.

The OIDA designed in this paper is used for marine target extraction experiment based on OTD_StErf. The times for every CPU in every node of the experiment to display the suspected targets and extract targets are shown in Fig.8, where Fig.8(a) is the parallel computing with single-machine 8-core CPU. The 35 suspected targets are allocated randomly to 0–7-core CPU, 4–5 tasks for one core. After each CPU receives the tasks, it will display positions of suspected targets in the research marine area in turn, and extract targets according to the target size and aspect ratio specified by the user. From Fig.8(a), it can be found that the suspected target detection takes approximately 6–7 s, and each target extraction takes approximately 5–7 s.

Fig.8(b) is the parallel computing with dual-machine 16-core CPU, and 1–3 tasks are randomly allocated to 0–15 core CPU. The times for suspected target detection and each target extraction are approximately 6–7 s. The first to complete the task is the 5-core and 6-core CPU, which are allocated 2 tasks and take 10 s. The last one to complete the task is 1-core CPU, which takes approximately 33 s. The CPU is assigned a total of 3 tasks, that is, 3 suspected targets need to be detected, of which 2 suspected targets meet the conditions. The extraction times of these 2 targets are 10 s and 25 s, respectively.

Fig.8(c) is the parallel computing with 3-machine 24-core CPU, and each CPU is randomly assigned 1–2 tasks. The first CPU that completes tasks takes 3 s, and the last takes approximately 27 s. Fig.8(d) is the parallel computing with 4-machine 32-core CPU, 1–2 tasks are randomly allocated to each CPU. The first CPU that completes tasks takes 1 s, and the last takes approximately 23 s. Fig.8(e) is the parallel computing with 5-machine 40-core CPU. Since there are 35 tasks, 35 CPU cores are involved in the calculation, and 1 task is randomly allocated to one CPU. The first CPU that completes tasks takes 1 s, and the last takes approximately 9 s. The 0–7 core CPUs in the calculation shown in Fig.8(b–e) are the chief CPUs. Since chief is responsible for assigning tasks to other Workers, the time for the chief CPU to display the first suspected target is when the program runs for 3 s, and the time for other Workers to display the first suspected target is when the program runs for 1 s, 0 or 3 s, 1 or 2 s, 1 or 2 s.

The CPU task execution status of each node is shown in Table 1. In single-machine 8-core CPU parallel architecture, each CPU takes approximately 53.77 s, all CPUs take a total of 430.23 s, and the total time to complete the task is 67.07 s, which is 17.58% of the time consumed by a single machine with one core, *i.e.*, 82.49% faster. In 5-machine 40-core CPU parallel architecture, the cluster composed of CPUs take 17.66 s to complete the task, which is 4.62% of the time consumed by a single machine, and 95.38% faster. The CPU execution status in other clusters are shown in Table 1.

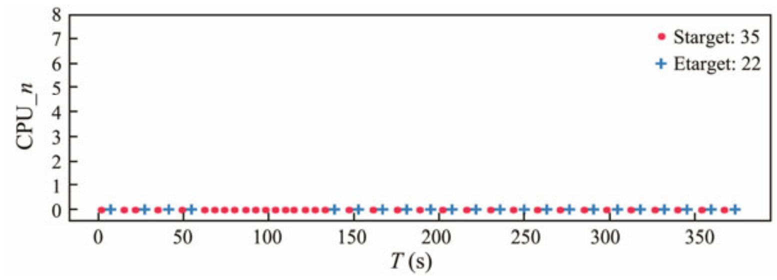

Fig.7 The display time of target parameter extraction when OTD_StErf method is carried out under single-machine single-core condition. The suspected targets (Starget) are represented by the red dot '•', and the extracted targets (Etarget) are represented by the blue plus '+'. Here in after the same.

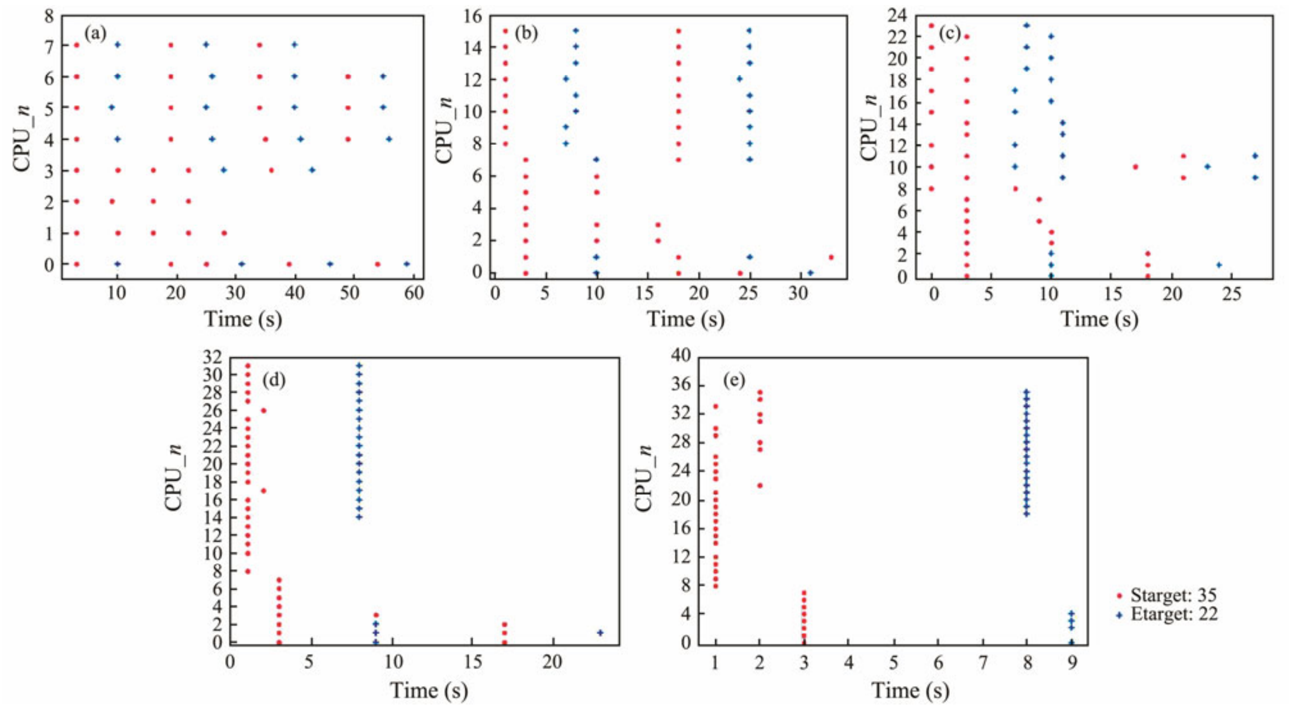

Fig.8 Parallel extraction CPU-*t* diagram for OTD_StErf.

Table 1 Parallel extraction CPU status for OTD_StErf

Categories	Number of working cores	Node task (%)	Total time of node (s)	Time per core (s)	Node server	Total time (s)
Single-machine 8-core CPU	8	100.00	430.23	53.77	Local	67.07
Dual-machine 16-core CPU	8	50.00	190.77	23.84	Local	37.96
	8	50.00	279.62	34.95	192.168.69.93:35000	
3-machine 24-core CPU	8	33.33	149.59	18.69	Local	32.76
	8	33.33	148.12	18.51	192.168.69.93:35000	
	8	33.33	170.33	21.29	192.168.69.30:35001	
4-machine 32-core CPU	8	25.00	129.34	16.16	Local	31.34
	8	25.00	113.13	14.14	192.168.69.35:35001	
	8	25.00	112.18	14.02	192.168.69.30:35002	
	8	25.00	105.28	13.16	192.168.69.93:35000	
5-machine 40-core CPU	7	19.44	79.78	11.39	192.168.69.93:35000	17.66
	7	19.44	91.38	13.05	192.168.69.9:35003	
	7	19.44	73.28	10.46	192.168.69.30:35002	
	7	19.44	91.40	13.05	192.168.69.35:35001	
	8	22.22	106.45	13.30	Local	

4.3 Experiment on OTD_Loglogistic

Single-machine single-core architecture is adopted to extract marine target parameters for 35 suspected targets, detected initially after DL, by OTD_Loglogistic, which takes 389.02 s and extracts 22 targets. As shown in Fig.9, each extraction takes approximately 6–9 s, and the first suspected target is displayed when the program runs for 1 s.

The OIDA framework designed in this paper is used to extract marine target parameters by OTD_Loglogistic method. The times that the CPU of each core at each node displays the suspected targets and extracts them are shown in Fig.10. Among them, Fig.10(a) is the parallel computing with single-machine 8-core CPU. The 35 suspected targets are randomly allocated to 0–7 core CPU, 4–5 tasks for one core. The first CPU that completes tasks takes 22 s, and the last takes approximately 60 s. Fig.10(b) is the parallel computing with dual-machine 16-core CPU. 1–3 tasks are randomly allocated to 0–15 core CPU. The first CPU that completes tasks takes 3 s, and the last takes approximately 33 s. Fig.10(c) is the parallel computing with 3-machine 24-core CPU. 1–2 tasks are randomly allocated to each CPU. The first CPU that completes tasks takes 7 s, and the last takes approximately 25 s. Fig.10(d) is the parallel computing with 4-machine 32-core CPU. 1–2 tasks are randomly allocated to each CPU. The first CPU that completes tasks takes 2 s, and the last takes approximately 24 s. Fig.10(e) is the parallel computing with 5-machine 40-core CPU. 1 task is randomly allocated to each CPU. The first CPU that completes tasks takes 1 s, and the last takes approximately 11 s. In Figs.10b–e, the first suspected target is displayed by the chief CPU when the program runs for 3 s or 2 s, and the first suspected target is displayed by the other working nodes when the program runs for 2 s, 1 s, 2 s, 1 s or 2 s, respectively.

In the parallel extraction process of marine targets by single-machine multi-core/multi-machine multi-core OTD_Loglogistic method, the CPU task execution status at each node is shown in Table 2. When the single-machine 8-core CPU is used, each CPU takes approximately 54.89 s, all CPUs take a total of 439.18 s, and the total time to complete the task is 68.40 s, which is 17.58% of the time consumed by a single machine with single core, and 82.42% faster. The cluster composed of 5-machine 40-core CPU takes 18.91 s to complete the task, which is 4.86% of the time consumed by a single machine with single core, and 95.14% faster. The CPU execution status in other clusters are shown in Table 2.

4.4 Experiment on OTD_Sgmloglog

An OTD_Sgmloglog method for marine target extraction in complex sea conditions is proposed by the author. The designed OIDA architecture is tested on this proposed method and the results are compared with the single-machine single-core mode.

Single-machine single-core is used to extract marine

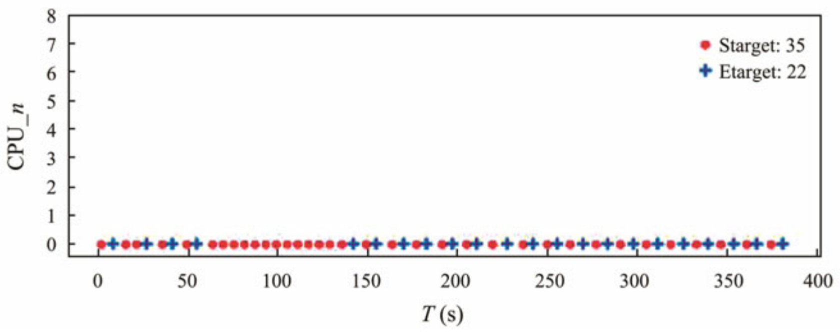

Fig.9 Display time of target parameter extraction for single-machine single-core OTD_Loglogistic method.

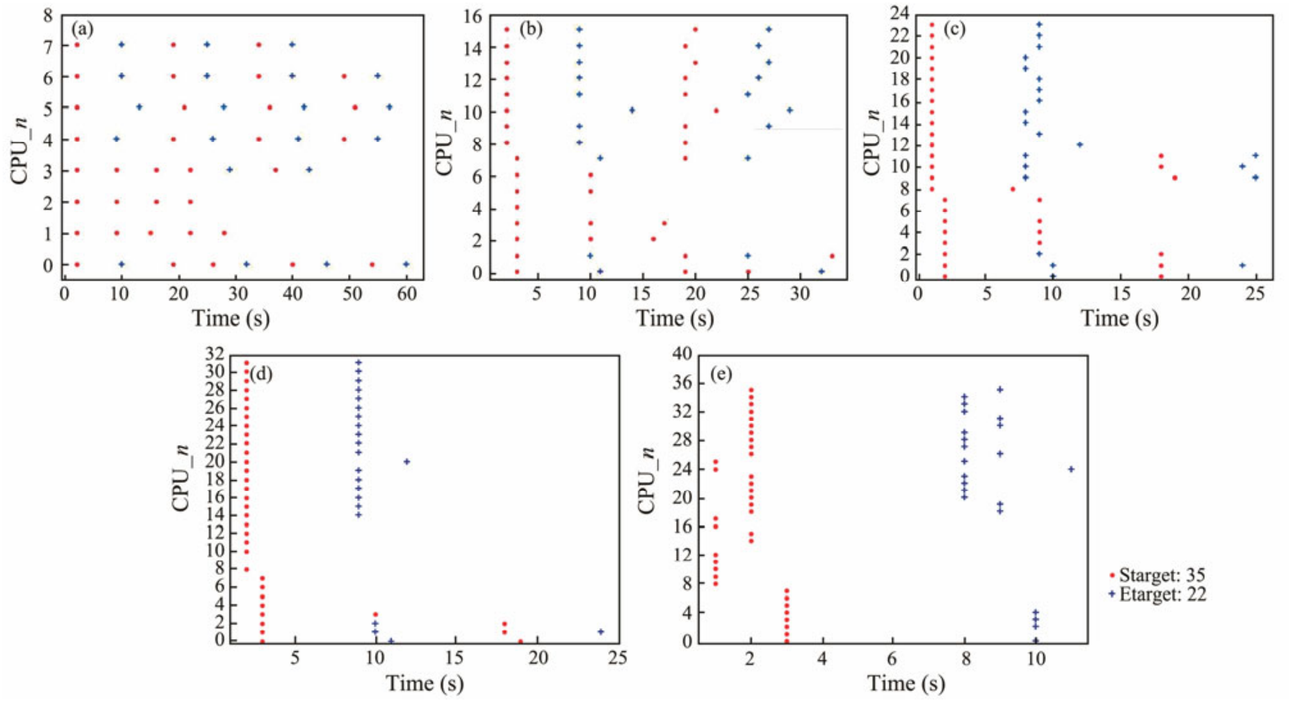

Fig.10 Parallel extraction CPU-*t* diagram for OTD_ Loglogistic.

Table 2 Parallel extraction CPU status for OTD_Loglogistic method

Categories	Number of working cores	Node task (%)	Total time of node (s)	Time per core (s)	Node server	Total time (s)
Single-machine 8-core CPU	8	100.00	439.18	54.89	Local	68.40
Dual-machine 16-core CPU	8	50.00	281.72	35.21	192.168.69.93:35000	39.08
	8	50.00	197.27	24.65	Local	
3-machine 24-core CPU	8	33.33	172.85	21.60	192.168.69.35:35001	32.83
	8	33.33	152.16	19.02	192.168.69.93:35000	
	8	33.33	154.15	19.26	Local	
4-machine 32-core CPU	8	25.00	103.84	12.98	192.168.69.35:35001	31.74
	8	25.00	107.15	13.39	192.168.69.93:35000	
	8	25.00	133.41	16.67	Local	
	8	25.00	128.38	16.04	192.168.69.30:35002	
5-machine 40-core CPU	7	19.44	84.54	12.07	192.168.69.93:35000	18.91
	7	19.44	93.98	13.42	192.168.69.30:35002	
	7	19.44	93.40	13.34	192.168.69.9:35003	
	7	19.44	75.81	10.83	192.168.69.35:35001	
	8	22.22	109.82	13.72	Local	

target parameters by OTD_Sgmloglog method for 35 suspected targets detected initially after DL. It takes 389.52 s to extract 22 targets. As shown in Fig.11, each target extraction takes approximately 6–7 s, and the first suspected target is displayed when the program runs for 1 s.

The designed OIDA architecture is used to extract marine target parameters by OTD_ Sgmloglog method. The times for each CPU at each node to display the suspected targets and extract targets are shown in Fig.12. Fig.12(a) is the parallel computing with single-machine 8-core CPU. The 35 suspected targets are randomly allocated to 0–7 core CPU, 4–5 tasks for one core. The first CPU that completes tasks takes 23 s, and the last takes approximately 60 s. Fig.12(b) is the parallel computing with dual-machine 16-core CPU. 1–3 tasks are randomly allocated to 0–15 core CPU. The first CPU that completes tasks takes 3 s, and the last takes approximately 33 s. Fig.12(c) is the parallel computing with 3-machine 24-core CPU. 1–2 tasks are randomly allocated to each CPU. The first CPU that completes tasks takes 3 s, and the last takes approximately 25 s. Fig.12(d) is the parallel computing with 4-machine 32-core CPU. 1–2 tasks are randomly allocated to each CPU. The first CPU that completes tasks takes 1 s, and the last takes approximately 24 s. Fig.12(e) is the parallel computing with 5-machine 40-core CPU. 1 task is randomly allocated to each CPU. The first CPU that completes tasks takes 1 s, and the last takes approximately 12 s. The 0–7 core CPU in Figs.12b–e is the chief CPU and the first suspected target is displayed when the program runs for 3 or 2 s. The first suspected target for other working nodes is displayed when the program runs for 2 s, 1 s or 2 s, 1 s or 2 s, 1 s or 2 s, respectively.

In the parallel extraction of marine target parameters by single-machine multi-core/multi-machine multi-core OTD_ Sgmloglog method, the CPU status of each node is shown in Table 3. Each CPU of the single-machine 8-core takes approximately 54.23 s, all CPUs take a total of 433.85 s, and the total time to complete the task is 67.40 s, which is 17.30% of the time consumed by a single machine with single core, and 82.70% faster. The cluster composed of 5-machine 40-core CPU takes 20.17 s to complete the task, which is 5.18% of the time consumed by a single machine with single core, and 94.82% faster. The CPU execution status in the other clusters is shown in Table 3.

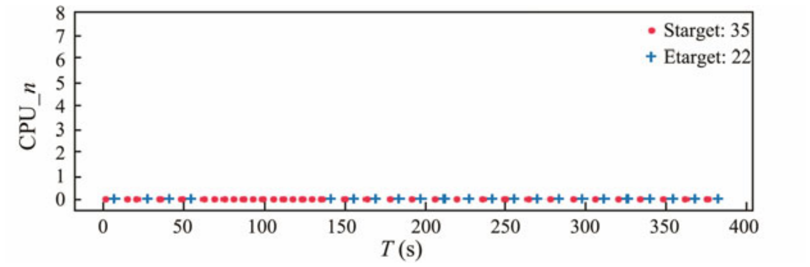

Fig.11 CPU-*t* diagram of parameter extraction for single-machine single-core OTD_ Sgmloglog method.

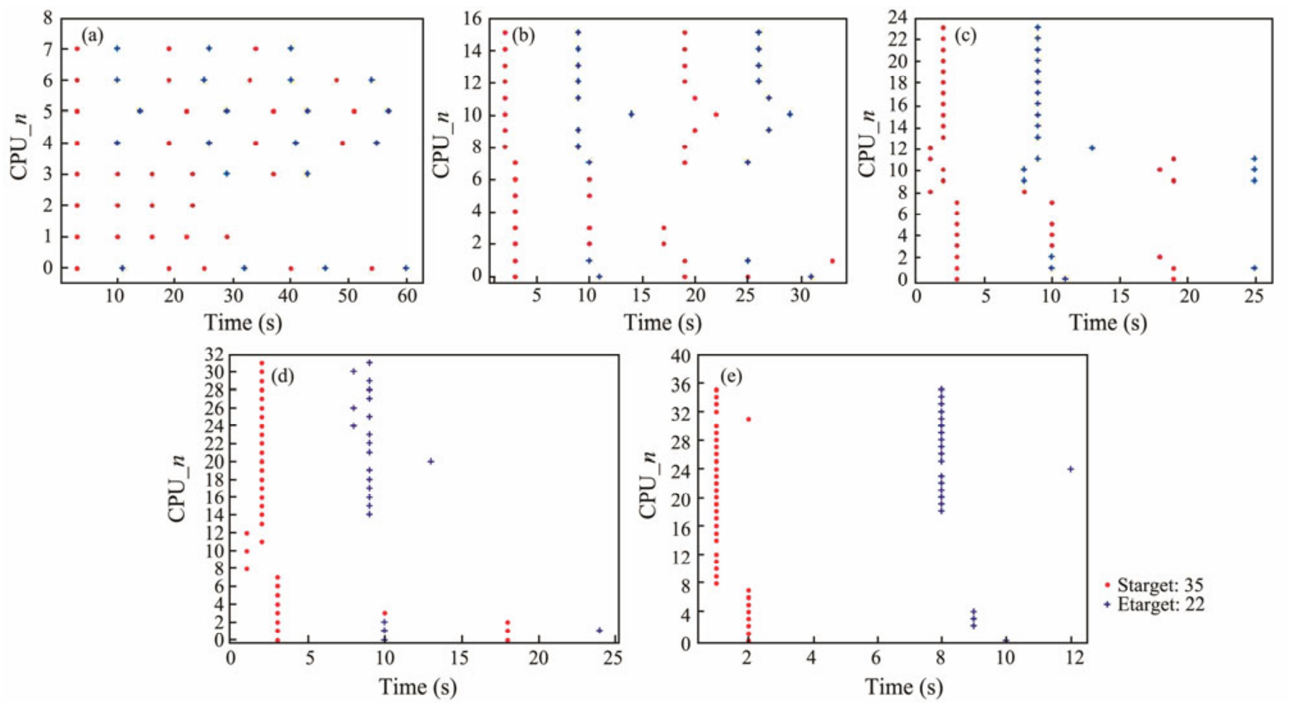

Fig.12 Parallel extraction CPU-*t* diagram for OTD_ Sgmloglog method.

Table 3 Parallel extraction CPU status for OTD_Sgmloglog method

Categories	Number of working cores	Node task (%)	Total time of node (s)	Time per core (s)	Node server	Total time (s)
Single-machine 8-core CPU	8	100.00	433.85	54.23	Local	67.40
Dual-machine 16-core CPU	8	50.00	282.91	35.36	192.168.69.93:35000	38.79
	8	50.00	195.71	24.46	Local	
3-machine 24-core CPU	8	33.33	151.49	18.93	192.168.69.93:35000	32.97
	8	33.33	173.20	21.65	192.168.69.35:35001	
	8	33.33	153.90	19.23	Local	
4-machine 32-core CPU	8	25.00	115.66	14.45	192.168.69.93:35000	31.80
	8	25.00	133.37	16.67	Local	
	8	25.00	107.85	13.48	192.168.69.30:35002	
	8	25.00	114.25	14.28	192.168.69.35:35001	
5-machine 40-core CPU	6	16.67	74.11	12.35	192.168.69.9:35003	20.17
	8	22.22	102.94	12.86	192.168.69.35:35001	
	8	22.22	109.84	13.73	Local	
	7	19.44	93.56	13.36	192.168.69.30:35002	
	7	19.44	74.55	10.65	192.168.69.93:35000	

4.5 Results for Marine Target Extraction

The above method is used to extract target parameters from the relevant SAR data in the research marine area, and the parameters of target position and shape are fitted by ellipses. The fitting results are shown in Fig.13, and the area marked by blue box is enlarged in Fig.14. The detailed ellipse fitting parameters of the extracted targets are listed in Table 4.

5 Discussion and Conclusions

The OIDA parallel distributed architecture is designed in the paper, and comparison experiments are conducted on three extraction methods of marine targets: OTD_StErf, OTD_Loglogistic, and OTD_Sgmloglog. Each method is applied under 6 deployments of single-machine single-core CPU, single-machine 8-core CPU, dual-machine 16-core CPU, 3-machine 24-core CPU, 4-machine 32-core CPU, and 5-machine 40-core CPU, including 18 experiments. Experimental results show that the parallel distributed OIDA architecture which is designed in this paper is efficient in the applications of marine target extraction. The average speed of parallel target extraction for a single-machine 8-core CPU is 5.75 times the speed of a single-machine single-core CPU, where the highest is 5.84 times, and the lowest is 5.78 times. The average speed of parallel target extraction for a 5-machine 40-core CPU is 20.75 times the speed of a single-machine single-core CPU, where the highest is 21.53 times and the lowest is 19.31 times. The parallel distributed OIDA architecture shortens the time required for the operation of the marine target extraction method. The average response time per thousand square kilometers is approximately 2 s, with the fastest response time of 1 s, and the slowest response time of 3 s. The designed parallel distributed architecture OIDA realizes the high-precision rapid extraction of target features.

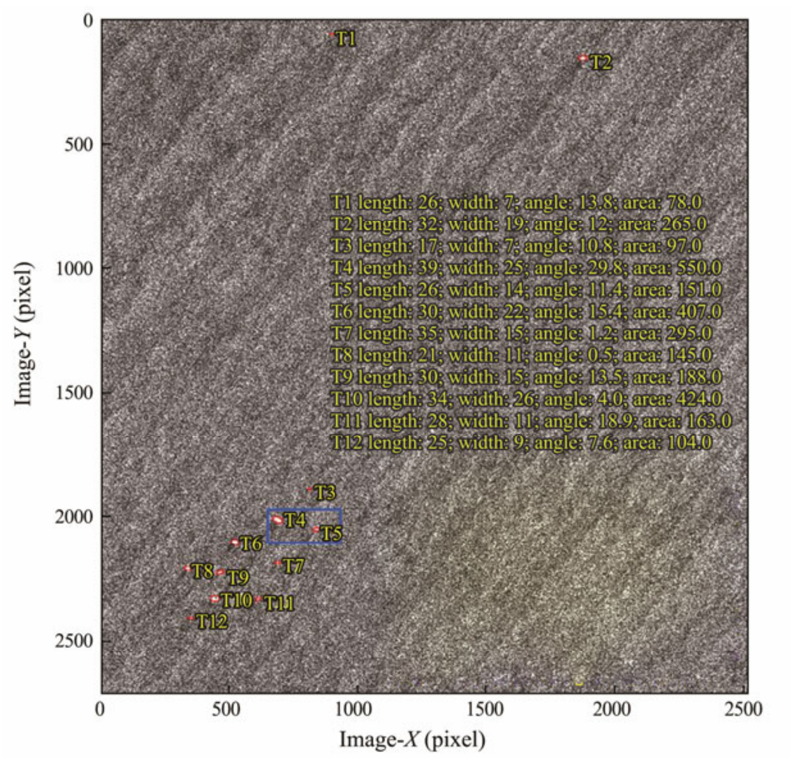

Fig.13 Marine target extraction results.

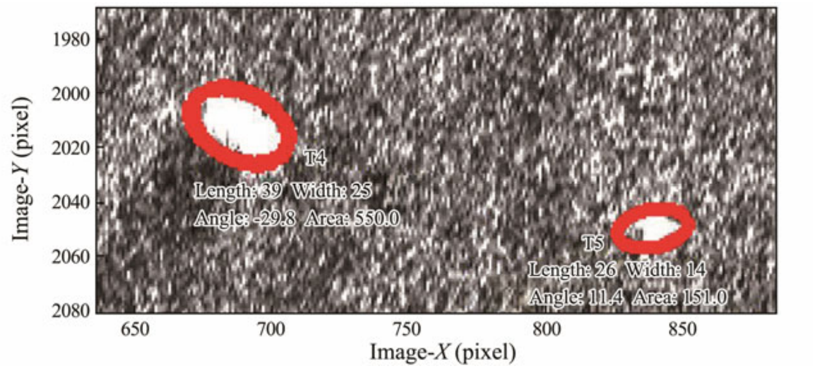

Fig.14 Local enlargement of marine target extraction results.

Table 4 Parameters of target extracted by experiment

Row	Column	Rotation angle	Semi-major axis	Semi-minor axis	Pixel amount
63.0	902.0	0.245	13.340	3.885	78.0
158.0	1874.0	0.021	16.050	9.945	265.0
1888.0	813.0	0.191	8.820	3.835	97.0
2012.0	689.0	−0.573	19.720	12.585	550.0
2050.0	839.0	0.202	13.240	7.450	151.0
2102.0	524.0	−0.276	15.390	11.185	407.0
2206.0	341.0	−0.020	17.985	7.510	295.0
2184.0	689.0	0.009	10.510	5.500	145.0
2222.0	465.0	0.240	15.385	7.875	188.0
2327.0	445.0	−0.069	17.390	13.440	424.0
2330.0	612.0	0.343	14.230	5.865	163.0
2407.0	351.0	0.133	12.720	4.595	104.0

Acknowledgements

The study was supported by the Natural Science Foundation of Shandong Province (No. ZR2019MD034), and the Education Reform Project of Shandong Province (No. M2020266). This study was supported by data from European Space Agency (ESA) and the Institute of Remote Sensing and Digital Earth, Chinese Academy of Sciences.

References

Alqahtani, S., and Demirbas, M., 2019. Performance analysis and comparison of distributed machine learning systems. *ArXiv Preprint ArXiv*: 1909.02061.

Aytekin, A., Feyzmahdavian, H. R., and Johansson, M., 2016. Analysis and implementation of an asynchronous optimization algorithm for the parameter server. *ArXiv Preprint ArXiv*: 1610.05507.

Bouzidi, S., 2019. Parallel and distributed implementation on SPARK of a spectral-spatial classifier for hyperspectral images. *Journal of Applied Remote Sensing*, **13** (3): 034501.

Chen, H., Wei, A., and Zhang, Y., 2017. Three-level parallel-set partitioning in hierarchical trees coding based on the collaborative CPU and GPU for remote sensing images compression. *Journal of Applied Remote Sensing*, **11** (4): 045015.

Cheng, Z., and Xu, Z., 2019. Bandwidth reduction using importance weighted pruning on ring allreduce. *ArXiv Preprint ArXiv*: 1901.01544.

Goyal, P., Dollár, P., Girshick, R., and Noordhuis, P., 2017. Accurate, large minibatch SGD: Training imagenet in 1 hour. *ArXiv Preprint ArXiv*: 1706.02677.

Hu, H., Shu, H., Hu, Z., and Xu, J., 2016. Using compute unified device architecture-enabled graphic processing unit to accelerate fast fourier transform-based regression kriging interpolation on a MODIS land surface temperature image. *Journal of Applied Remote Sensing*, **10** (2): 026036.

Huang, Y., Jin, T., Wu, Y., Cai, Z., and Cheng, J., 2018. Flexps: Flexible parallelism control in parameter server architecture. *Proceedings of the VLDB Endowment*, **11** (5): 566-579.

Li, T., 2015. Parallel optimization and application research on moving object detection and recognition algorithms. Master thesis. National University of Defense Technology.

Ling, B., Deng, Y., and Yu, S. B., 2016. Processing for accelerated sparse PCNN moving target detection algorithm with CUDA. *Computer Engineering and Design*, **37** (12): 3300-3305.

Liu, Y. H., Zhou, J., Qi, W. H., Li, X. L., Gross, L., Shao, Q., *et al.*, 2020. ARC-Net: An efficient network for building extraction from high resolution aerial images. *IEEE Access*, **8**: 154997-155010, DOI: 10.1109/ACCESS.2020.3015701.

Lou, X. H., Guo, C. S., Song, S. L., and Qi, L. Q., 2016. Parallel implementation of video moving object detecion algorithm based on CUDA. *Journal of Hangzhou Dianzi University (Natural Sciences)*, **36** (3): 23-26.

Mamidala, A. R., Kollias, G., Ward, C., and Artico, F., 2018. MXNET-MPI: Embedding MPI parallelism in parameter server task model for scaling deep learning. *ArXiv Preprint ArXiv*: 18 01.03855.

Peng, B., Zhang, C. Y., Zheng, S. B., and Tian, G., 2014. Multilevel parallel optimization of moving object detecion and feature extraction algorithm. *Video Engineering*, **38** (13): 173-177.

Quirita, V., da Costa, G., Happ, P., Feitosa, P., da Silva Ferreyra R., Oliveira, D., *et al.*, 2016. A new cloud computing architecture for the classification of remote sensing data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **10** (2): 409-416.

Shen, S., Xu, L., Liu, J., Liang, X., and Cheng, Y., 2019. Faster distributed deep net training: Computation and communication decoupled stochastic gradient descent. *ArXiv Preprint ArXiv*: 1906.12043,

Thao Nguyen, T., Wahib, M., and Takano, R., 2019. Efficient MPI-allreduce for large-scale deep learning on GPU-clusters. *Concurrency and Computation: Practice and Experience*: e5574.

Wen, S., 2017. The parallel design and implementation of MCMC multi-object tracking algorithm. Master thesis. Xidian University.

Wu, Z., Li, Y., Plaza, A., Li, J., Xiao, F., and Wei, Z., 2016. Parallel and distributed dimensionality reduction of hyperspectral data on cloud computing architectures. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **9** (6): 2270-2278.

Ye, L. H., Wang, L., Sun, Y. X., Zhao, L. P., and Wei, Y. W., 2017. Parallel multi-stage features fusion of deep convolutional neural networks for aerial scene classification. *Remote Sensing Letters*, **9**: 3, 294-303.

You, W., 2016. Research on hyperspectral remote sensing target detection parallel processing. Master thesis. Harbin Engineering University.

Zeng, T., 2017. Research on parallel algorithm for moving object contour extraction in video sequences. Master thesis. Central China Normal University.

Zhang, F., 2017. Target recognition and parallel acceleration with GPU in marine remote sensing image. Master thesis. Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences.

(Edited by Chen Wenwen)