Towards a real-time modeling of global ocean waves by the fully GPU-accelerated spectral wave model WAM6-GPU v1.0

Ye Yuan[1,2], Fujiang Yu[1,2], Zhi Chen[1,2], Xueding Li[3], Fang Hou[1], Yuanyong Gao[1,2], Zhiyi Gao[1,2], and Renbo Pang[1,2]

[1]National Marine Environmental Forecasting Center of China, Beijing, 100081, China
[2]Key Laboratory of Research on Marine Hazards Forecasting, Ministry of Natural Resources of China, Beijing, 100081, China
[3]Fujian Marine Forecasts

Correspondence: Ye Yuan (yuanye@nmefc.cn)

Abstract. The spectral wave model WAM (Cycle 6) is a commonly-used code package for ocean wave forecasting. However, it is still a challenge to include it into the long-term earth system modeling due to huge computing requirement. In this study we have successfully developed a GPU-accelerated version of the WAM model that can run all its computing-demanding components on GPUs, with a significant performance increase compared with its original CPU version. The power of GPU computing has been unleashed through substantial efforts of code refactoring, which reduces the computing time of a 7-day global 1/10° wave modeling to only 7.6 minutes in a single-node server installed with 8 NVIDIA A100 GPUs. Speedup comparisons exhibit that running the WAM6 with 8 cards can achieve the maximum speedup ratio of 37 over the dual-socket CPU node with 2 Intel Xeon 6236 CPUs. The study provides an approach to energy-efficient computing for ocean wave modeling. A preliminary evaluation suggests that approximately 90% of power can be saved.

1 Introduction

Ocean wave modeling has long been regarded as an important part of numerical weather prediction systems, not only due to its critical role in ship routing and offshore engineering, but also its climate effects (Cavaleri et al., 2012). Nowadays wave forecasts are generated from the spectral wave models routinely operating at the scales of ocean basins to continental shelves (Valiente et al., 2023). To assess uncertainty of wave forecasts, major national weather prediction centers also adopt ensemble approach to provide probabilistic wave forecasts (Alves et al., 2013). For example, the Atlantic Ensemble Wave Model of the Met Office issues probabilistic wave forecasts from an ensemble of 36 members at 20 km resolution. However, due to the limitation of the High Performance Computing (HPC) resource, these ensemble members have to be loaded in 2 groups with a latency of 6 hours for each (Bunney and Saulter, 2015). Increasing spatial resolution and conducting ensemble forecasting both require huge HPC resource (Brus et al., 2021).

In recent decades, wind-generated surface waves are revealing themselves more and more as the crucial element modulating the heat, momentum and mass fluxes between ocean and atmosphere (Cavaleri et al., 2012). The air-sea fluxes of momentum and energy are sea state dependent and the forecasts of winds over ocean are demonstrably improved by coupling atmosphere and wave models. Ocean circulation modeling can also be improved through coupling to wave models (Breivik et al., 2015; Couvelard et al., 2020). Waves absorb kinetic energy and momentum from the wind field when they grow and in turn release it when they break, thus enhancing the turbulence in the upper mixing layer significantly (Janssen, 2012). Non-breaking waves and swells stir the upper water and produce enhanced upper-layer mixing (Huang et al., 2011; Fan and Griffies, 2014). For high-latitude Marginal Ice Zones (MIZs), ice-wave interactions comprise a variety of processes such as wave scattering and dissipation, ice fracturing, which also have important impacts on both atmospheric and oceanic components (Williams et al., 2012; Iwasaki and Otsuka, 2021). The earth system model (ESM) developers are making efforts to explicitly include a surface wave model into their coupling framework. For example, as part of ECMWF's Earth system, the wave model component of the IFS is actively coupled to both the atmosphere and the ocean modeling subsystems (Roberts et al., 2018). The FIO-ESM (Qiao et al., 2013; Bao et al., 2020) and the Community Earth System Model Version 2 (Danabasoglu et al., 2020; Brus et al., 2021; Ikuyajolu et al., 2024) also include a spectral wave model as part of their default components.

The spectral wave models resolve wave phenomenon as a combination of wave components along a frequency and direction spectrum across space and time. Their generation, dissipation, and nonlinear interaction processes are described by a wave action transport equation with a series of source terms, which requires large computational resources and long computing time (Cavaleri et al., 2007). The first operational third-generation spectral model, the WAM (WAve Modeling) was developed in 1980s (The Wamdi Group, 1988). The WAM has been replaced by the WaveWatch III (Tolman, 1999) as the most commonly used global and regional wave model used by NWP centers. The SWAN (Simulating WAves Nearshore) is also a popular wave model which has advantages in the nearshore regions (Booij et al., 1999). All these wave models are under continued development by different communities, and their code packages are freely accessible.

As an efficient, portable yet commercially available substitute for HPC clusters, GPU computation has played a central role in implementing massive computation across a wide range of areas (Yuan et al., 2020, 2023). In the area of ocean and climate modeling, a variety of mainstream models and algorithms have been successfully ported to the GPU devices with substantial improvement on model performance reported (Xiao et al., 2013; Xu et al., 2015; Qin et al., 2019; Yuan et al., 2020; Jiang et al., 2019; Häfner et al., 2021). For brevity, in this paper we do not elaborate these pioneering works. Due to the complexity of code, GPU acceleration of spectral wave models has not been reported in public until recently. Ikuyajolu et al. (2023) has ported one of the WaveWatch III's source-term scheme W3SRCEMD to GPU with OpenACC. However, the GPU implementation only exhibited a 35 – 40 % decrease in both simulation time and usage of computational resources, which was not satisfactory. They suggested that, rather than a simple usage of OpenACC routine directives, a significant effort on code refactoring had to be done for better performance.

Since 2020, we have managed to fully port the WAM Cycle 6 wave model to GPUs through a significant amount of works. The purpose of this paper is to report the progress that has been made. The paper is organized as follows. Section 2 briefly reviews the governing equations and physical schemes adopted by the WAM; Then, the OpenACC implementation of the model

and optimization strategies are detailed in Section 3. Performance analysis through a global high-resolution wave hindcast is made in Section 4. A brief summary is presented in Section 5.

2 Wave model WAM

The ocean wave model WAM is a third-generation wave model that describes the evolution of the wave spectrum by solving the wave energy transport equation. In this study, the WAM Cycle 6 (hereinafter referred to as WAM6) is adopted. The WAM6 is one of the outcomes of European Union (EU) funded MyWave project, which brings together several major operational centers in Europe to develop a unified European system for wave forecasting based on the updated wave physics (Baordo et al., 2020). Generally, the third-generation wave models are based on the following wave action balance equation in spherical coordinates (Equation 1),

$$\frac{\partial}{\partial t}N + (cos\phi)^{-1}\frac{\partial}{\partial \phi}(\dot{\phi}cos\phi N) + \frac{\partial}{\partial \lambda}(\dot{\lambda}N) + \frac{\partial}{\partial \omega}(\dot{\omega}N) + \frac{\partial}{\partial \theta}(\dot{\theta}N) = S_{in} + S_{ds} + S_{nl} + S_{bot} + S_{brk}, \quad (1)$$

The left-hand side of Equation 1 accounts for spatial and intra-spectrum propagation of spectral wave energy along two-dimensional geographic space (ϕ for latitude and λ for longitude), angular frequency (ω) and direction (θ), respectively. The 5 terms in the right-hand side of the equation 1 is source terms, which represent the physics of wind input (S_{in}), dissipation due to whitecapping (S_{ds}), nonlinear wave-wave interaction (S_{nl}), bottom friction (S_{bot}), and depth-induced wave breaking (S_{brk}), respectively. Two sets of wave physics parameterizations are available in the WAM6, which are based on the works of Janssen (1989, 1991) and Ardhuin et al. (2010). A first-order upwind flux scheme is used for advection terms, and the source terms are integrated using a semi-implicit integration scheme. The full description of the governing equation, parameterization and numerical schemes can be referred to ECMWF (2023), Günther et al. (1992), and Behrens and Janssen (2013).

The flow diagram of the WAM6 model is illustrated in Figure 1. The WAM6 model has a pre-processing program (*PREPROC*) which generates the computing grids with two alternative options. The default option is to produce a reduced Gaussian grid. In this case, the number of grid points along the latitude reduces polarward with east-west grid spacing comparable for all latitudes. This approach not only benefits numerical stability required by the Courant–Friedrichs–Lewy (CFL) condition, but also improves model efficiency by reducing cell number considerably. The forward integration of the model at each grid point is implemented within the main program *CHIEF*, which in turn considers wave propagation, source term integration, and post-processing of wave spectra for a variety of wave parameters. These processed fields comprise parameters to describe the mean sea state, major wind sea and swell wave components, as well as freak waves. Other parameters also include radiation stresses, Stokes drifting velocities, and maximum wave/crest height (ECMWF, 2023). Wave spectra and wave parameters can then be written into the formatted or NetCDF files at specified intervals, and this is achieved by the *PRINT* programs.

When running in parallel, the WAM6 model splits the global computation domain into rectangular sub-domains with equal size. Only sea points are retained. The Message Passing Interface (MPI) is adopted to exchange wave spectra of the overlapping halos between connecting sub-domains. The WAM6 is able to run on a HPC cluster with up to thousands of processors with acceptable speedup.

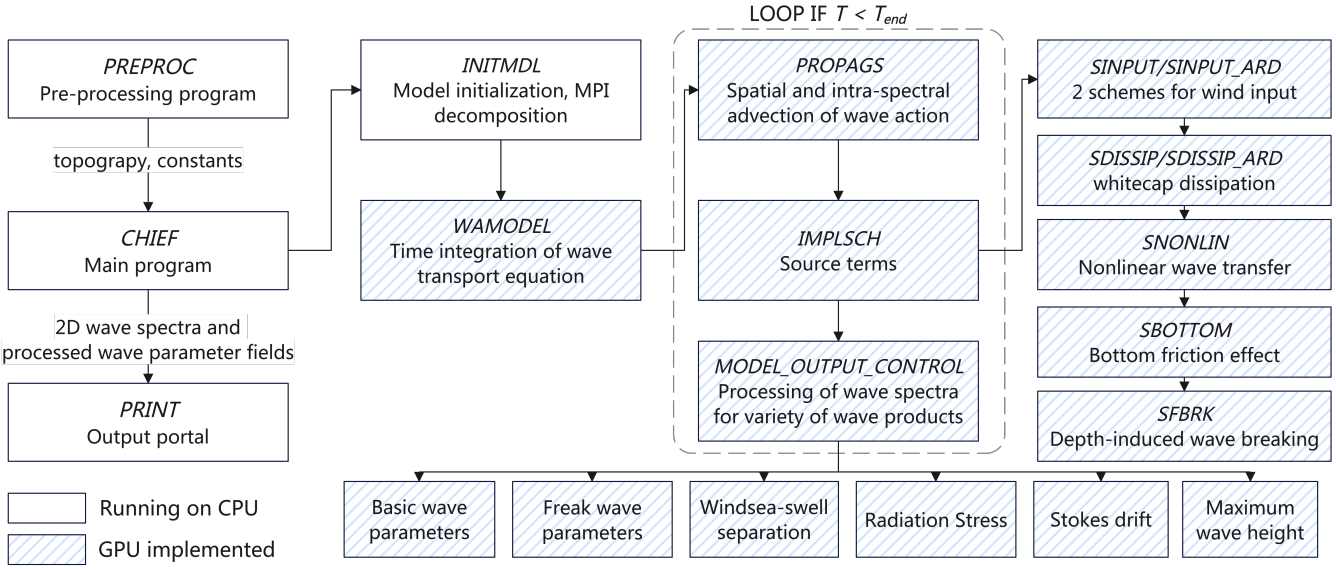

Figure 1. Flow diagram of the WAM6 for a regular wave modeling. The WAM6 modeling is implemented with a three-step procedure, including pre-processing (*PREPROC*), model integration (*CHIEF*), and output (*PRINT*). The shaded box denotes the compute-intensive modules that are ported to GPU.

3 Porting WAM6 to GPU

Before porting a complex code package to GPU, the developers should consider whether the core algorithm is suitable for GPU (Applicability), whether the programming effort is acceptable (Programmability), whether the ported code can fit the rapidly evolving and diverse hardware (Portability), and whether it can scale efficiently to multiple devices (Scalability). CUDA and OpenACC are two commonly used GPU programming tools. CUDA can access all features of the GPU hardware to enable more optimization. A number of studies on GPU-accelerated hydrodynamic models have been reported to use CUDA C or CUDA Fortran as coding languages (Xu et al., 2015; de la Asunción and Castro, 2017; Qin et al., 2019; Yuan et al., 2020). Substantial efforts have to be done to encapsulate the computing code with CUDA kernels. However, lessons learned by us is that using CUDA to rewrite a code package which is under community development is not always a smart choice. OpenACC is a parallel programming model which provides a collection of instructive directives to offload computing code to GPU and other multi-core CPU. Using OpenACC greatly shortens the development time, reduces the learning curve of non-GPU programmer and supports multiple hardware architectures. Therefore, we adopted OpenACC as the coding tool.

To fully accelerate the WAM6 model and improve performance, the best practice is to offload all compute-intensive code to GPU devices and maximize data locality by keeping necessary data resident on GPU. The spectral wave model is memory-consuming due to the existence of a four-dimensional prognostic variable N (Equation 1). For example, for a global 1/10° wave model consisting of 36 directions and 35 frequencies, the variable of spectral wave energy is as large as 36 Gigabyte

(GB). All the coding strategies should comply with the facts that GPU memory is limited and data transfer between the host and device is expensive. As shown in Figure 1, the overall strategy in this study is to run the entire $CHIEF$ program on GPUs without any data transfer between CPU and GPU memories. For multiple-GPU programming, the cuda-aware OpenMPI is used to conduct inter-GPU communication.

3.1 Data hierarchy

Unlike its predecessor, the WAM6 now is modernized with the FORTRAN 90 standard. All subroutines are organized as groups and then encapsulated as modules in accordance with their purposes. As the spatial and intra-spectrum resolutions increase, the memory usage for a global wave modeling shows a exponential growth. Hence, careful design of data structure and exclusion of unnecessary variables on GPU are important, which requires a panorama of the entire WAM6 model, as well as a thorough analysis of individual variable.

As illustrated in Figure 2, a three-level data hierarchy is established with each level having different lifetime and scope. The prognostic and state variables are constantly modified during the program runtime, and declared as global variables using ⟨*acc declare*⟩ directive with *create* clause. The module-level variables are private to the module where they are declared. They have the same lifetime of the residing module, and are also defined on GPU using ⟨*acc declare*⟩ directive. For the global and module-level variables, the FORTRAN *allocate* statement allocate the variables in both host and device memories simultaneously. The OpenACC uses ⟨*acc data*⟩ directive to create local variables that are only accessible locally inside a subroutine. They are allocated and freed within the ⟨*acc data*⟩ region.

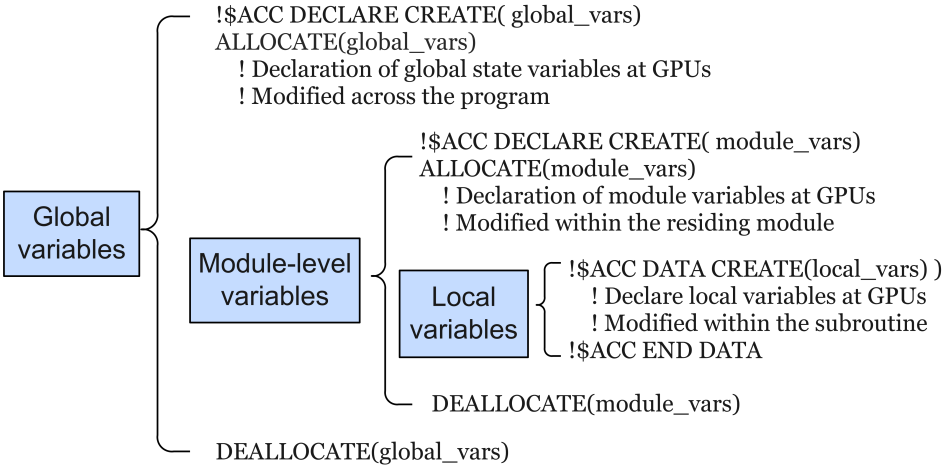

Figure 2. The three-level data hierarchy of the WAM6-GPU v1.0.

3.2 Optimization strategies

Literally, OpenACC implementation can be as simple as inserting directives before specific code sections to engage the GPUs. However, to fully utilize the GPU resource, optimizations such as code refactoring or even algorithm replacing are necessary, which require a wealth of knowledge of both GPU architecture and code package itself. Before introducing the optimizing strategies, some knowledge about the WAM6, especially that related to GPU computing, is reiterated here.

First of all, the WAM6 is compute-intensive, so GPU accelerating is certainly promising. A global 1/10° high-resolution wave modeling (corresponding to nearly 3 million grid points) with 35 frequencies and 36 directions means that approximately 4 billion equations have to be solved for a time step. In the WAM6, a mapping from the two-dimensional spherical grid to a one-dimensional array is performed. Thus, the computation of wave physics is mainly composed of a series of three-level nested loops iterating each grid, frequency and direction. Iteration on grid points is placed in the innermost loop for vectorization where appropriate. Nevertheless, as explained later, sometimes better performance requires loop on grid points placed to the outermost, leaving two another loop levels executing serially.

Secondly, the WAM6 can be memory- and I/O-bound, as it contains a dozen of three-dimensional (3D) variables with overall size amounting to hundreds of GB. Some optimizing strategies, such as loop collapse, contain trade-off between performance and memory occupancy. Besides, previous work (Ikuyajolu et al., 2023) only accelerates source terms of the wave model, and postprocessing of spectral wave energy for a variety of wave parameters has not been ported to GPUs. This leads to a mass device-to-host transfer of the 3D wave spectra at each output time step. The consequence is that the achieved speedup ratio is immediately neutralized by the transfers.

Thirdly, the nonlinear wave-wave interaction (S_{nl}) redistributes energy over wave components with similar frequencies and directions. Thus parallelism along the dimensions of frequency and direction cannot be performed. Optimizing strategies on S_{nl} should be considered carefully. Spectral separation of windsea and swell also involve manipulation between different wave components, which cannot be vectorized either.

Generally, we pursue a high GPU occupancy by loop reordering, collapse and refactoring. The key principles are to tune how the loops are mapped to the GPU hardware, and allocate optimal workload and on-chip resource for each CUDA core. We take full advantage of on-chip memories and caches by evaluating the necessity of each global memory access, and whether it can be converted to on-chip access. We eliminate device-host memory copies completely during the runtime of the program by migrating the whole wave physics and postprocessing modules to the GPUs. It should be noted that too much code refactoring may be detrimental to code readability in terms of its physical concept. Based on the full absorption of model physics, loop reordering and collapse are the main strategies for better speedup, which have not altered the code structure in most cases. Heavy code refactoring only occurs in computing source term of nonlinear wave interactions, as well as in some MPI interface functions. More specifically, some of the code optimization is presented by a few examples as follows.

Restructuring the complex loops The wave physics of the WAM6 consists of complex loops with nested functions and a number of private scalars or local variables. A simplest implementation is to insert OpenACC directives before the outer loops to generate a huge GPU kernels, and using ⟨*acc routine*⟩ to automatically parallelize the nested functions. The implementation

does work but comes with poor performance generally. A GPU kernel with appropriate size helps to achieve better performance. When appropriate, we split complex loops into a series of short and concise loops to increase parallelism and GPU occupancy. For nested subroutines, a naive usage of ⟨*acc routine*⟩ is not encouraged. A rule of thumb is to make loop-level (fine-grained) parallelism inside the nested subroutines with complex structure.

Nevertheless, splitting the outermost loop into pieces sometimes leads to an dilemma. In WAM6, evaluation of the spectral wave energy is a 3D computation. As shown in Figure 3(a), local, intermediate scalars or variables have to be declared as 3D variables to store the temporary information. It is commonplace in subroutine to compute the nonlinear wave-wave interactions S_{nl}. Although splitting the loop can achieve vectorization along all loop levels (Right panel of Algorithm a), the space of GPU memory may be limited in this case, not to mention the overhead for creating/freeing the memory each time. Thus, trade-off between parallelism and global memory must be considered.

Increasing parallelism by loop collapse Except aforementioned loop splitting, loop reordering is commonly used to organize the nested loops. We ensure that the 3D global variables are being accessed consecutively along the dimensions with an increasing manner. OpenACC then collapses the nested loops into one dimension. For the nested loops without any dependency, we place the gridcell-based loop (IJ) to the innermost, immediately followed by the direction- (K) and frequency-based (M) loops. Loop collapse is limited to the tightly nested loops. As shown in Figure 3(b), for loosely nested loops, some scalars are computed between the nested loops. If appropriate, these scalars are placed into the innermost loop at the cost of computing them redundantly. In this way, the loops are converted to the tightly-nested loops.

Reducing global memory allocation and access Global memory allocation and access are expensive operations on GPU, which may incur performance degradation. One widely-used strategy is to convert 2D or 3D local variables in the nested loops to loop-private scalars if appropriate. In the subroutine to evaluate S_{nl}, there are parameter arrays that store dozens of factors for spectral components of the wave. As exhibited in Figure 3(c), the best practice is to assign each value of the array to a temporary scalar which is private to the loop, rather than offloading the array to the global memory of the device. The latter would increase global memory access substantially. Using too many private scalars inside one kernel raises the concerns of a possible lower GPU occupancy due to spills of registers. The fact is that OpenACC tries to handle it properly by manipulating the physical resource (i.e., registers, L1/L2 cache) for these local variables, without degrading performance notably.

Choosing longest dimension for reduction Reduction operations are ubiquitous in the WAM6. A simple usage of ⟨*acc reduction*⟩ construct along the dimensions of frequency or direction should be avoided. It requires a parallel level of *vector* in OpenACC to do the reduction at each grid point, leaving the majority of threads idle. Instead, we should map each thread to a grid point and instruct the thread to do the reduction in a serial way (Figure 3(d)). The approach can fully occupy the GPU resource. Using ⟨*acc reduction*⟩ for global reduction along the longest dimension IJ is encouraged.

For multiple-GPU MPI communication, we fill the sending and receiving buffers by GPU, and using ⟨*acc host_data*⟩ construct for device-to-device data exchange. This reduces MPI communication time dramatically (Figure 5).

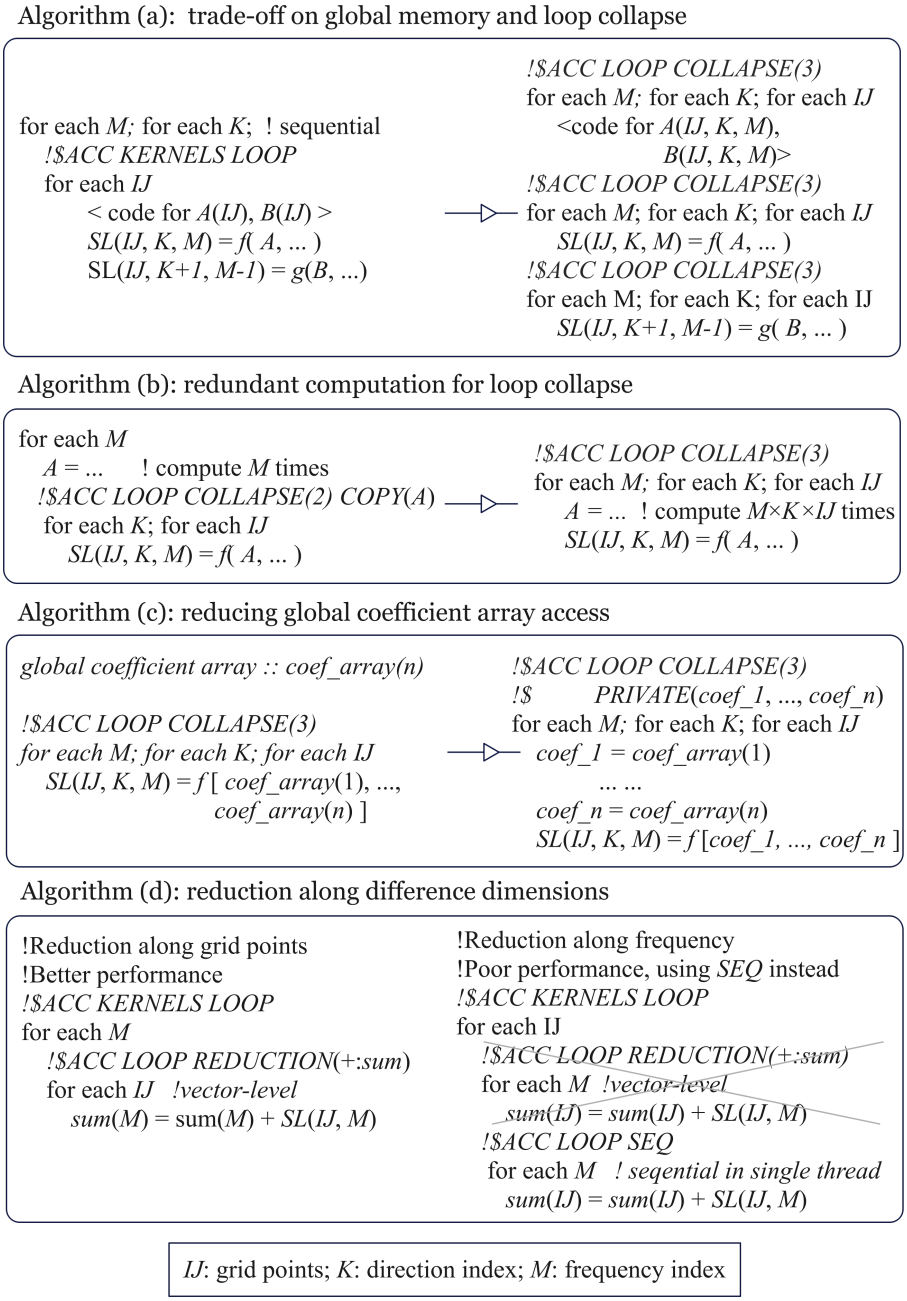

Figure 3. Optimizing strategies to improve performance in the WAM6-GPU v1.0. In the model, the three-level nested loops are extremely common, and IJ, M, K denote indexes of grid cell, frequency and direction.

4 Performance evaluation

4.1 Hardware and model configuration

The WAM6-GPU v1.0 is tested on a GPU server with 8 NVIDIA A100 GPUs. The compilation environment is NVIDIA HPC SDK version 22 and NetCDF 4.3. The NVIDIA A100 consists of 6912 CUDA cores sealed on 108 streaming multiprocessors. These cores access 80 GB of device memory in a coalesced way. The GPU server is hosted by 2 Intel Xeon 6236 CPUs with 32 (2×16) physical cores running at 2.9 GHz. In this study, the test case first runs on these 2 CPUs with 32 cores fully occupied, and the corresponding execution time (measured in second) is chosen as the baseline for speedup computation. Speedup ratio is defined as a ratio of the baseline time over the execution time of the same test case on GPUs

Besides, 2 HPC clusters are used to test the MPI scalability of the WAM6. The National Marine Environmental Forecasting Center of China's (NMEFC's) cluster has 120 active computing nodes, and each has dual Intel Xeon E5-2680 v4 CPUs with 28 cores. The Beijing Super Cloud Computing Center's (BJSC's) cluster provides us with 32-node computing resource. Each node has dual AMD EPYC 7452 CPUs with 64 cores. The hardware configurations is listed in Table 1.

Table 1. Hardware configurations for performance evaluation of the WAM6-GPU v1.0.

Resource	Hardware Specifications			
	CPU cores per node	GPUs per node	Available nodes	CPU cores in total
NMEFC's GPU server	32 Intel Xeon 6326	8 NVIDIA A100	1	32
NMEFC's HPC cluster	28 Intel Xeon 2680v4	0	120	3360
BJSC's HPC cluster	64 AMD EPYC 7452	0	32	2048

To evaluate performance of the WAM6-GPU v1.0, 3 test cases (hereafter referred as to T1-T3) are configured to conduct a global high-resolution wind hindcast. The model configuration is summarized in Table 2. In T1, the global model runs on a reduced Gaussian grid at horizontal resolution of 0.1°, and the wave spectrum at each grid point is discretized with 24 directions and 25 frequencies. The modeling area extends to latitude of 89.9° in Arctic and Antarctic regions. The total grid points is 2,923,286. The resolution of wave spectrum changes to 36 directions and 35 frequencies for T2. In T3, regular grid is adopted, and the number of grid points increase to 4,271,075. According to CFL condition, the time step for propagation reduces from 43 second to 0.9 second. In short, the computing workload gets heavier as spectral resolution and grid points increase from T1 to T3. Generally, regular grid adopted in T3 is not used for a practical global modeling.

The hourly ECMWF ERA5 reanalysis (Hersbach et al., 2020) is used as atmospheric forcing with a spatial resolution of 0.25°. Output parameters are generated at an interval of 3 hours. The WAM6-GPU v1.0 is configured as a stand-alone

model which do not consider sea ice-wave interaction and current refraction. Two schemes for wave growth and dissipation parameterization (Janssen, 1991; Ardhuin et al., 2010) are used with integrating time step of 5 minutes. Ardhuin et al. (2010)'s scheme is slightly more compute-intensive. In the study the performance metric is based on the average computing time required for a 1-day global wave hindcast collected from the 30-day simulations (January 1-30, 2021).

Table 2. Summary of the configuration for the global wave hindcast cases.

	Configuration of WAM6		
	T1	T2	T3
Model grid	0.1°, 89.9°S - 89.9°N		
Grid type	Gaussian grid 2,923,286 grids	Gaussian grid 2,923,286 grids	Regular grid 4,271,075 grids
Bathymetry	ETOPO2 interpolated to 0.1°		
Wave spectrumn	24 directions 25 frequencies	36 directions 35 frequencies	36 directions 35 frequencies
Source time step	5 minutes		
Source schemes	Janssen, 1991	Janssen, 1991	Ardhuin et al., 2010
Wind forcing	ECMWF ERA5, 0.25°, 1-hour		
Hindcast duration	January 1-30, 2021		
Sea ice concentration	Sea ice masking is not implemented		
Ocean current	Current refraction is not considered		

4.2 Speedup comparisons

First of all, MPI scalability of the WAM6 is evaluated on NMEFC's HPC cluster using 84, 168, 336, 672, 1344 and 2688 cores, respectively. The T1 case is used. As shown in Figure 4, the WAM6 exhibits perfect linear scaling up to 1344 cores. The drop in scalability becomes significant when the T1 case is split onto 2688 cores. In this case, 1-day global wave hindcast requires at least 72.0 seconds. Communication time for the exchange of the overlapping halo cells among the neighboring sub-domains (upper panel in Figure 4) declines, and the curve gradually flattens as expected.

In Figure 5, the bar plot displays the execution time and speedup ratios of the WAM6-GPU v1.0 when allotted with different hardware resource. The T1 case is used. The original code is first placed on NMEFC's GPU server with only 32 CPU cores fully used. The average wall-time taken to run a 1-day forecast is 2393.2 seconds. Time for model initialization is excluded. We then choose it as baseline to compute speedup ratios, which are annotated as gray text in Figure 5.

Overall, when running on 2 GPUs, the WAM6-GPU v1.0 can achieve a speedup of 10 over the dual-socket CPU node, and a near-linear scaling can be observed when 4 and 8 GPUs are allocated. This means when the 8-card GPU server is at full strength, the maximum speedup ratio can achieve 37, which greatly shortens the execution time of the 1-day global wave modeling to 65.2 second. As a comparison, the T1 case has been tested on the NMEFC's and BJSC's HPC clusters (Table 1).

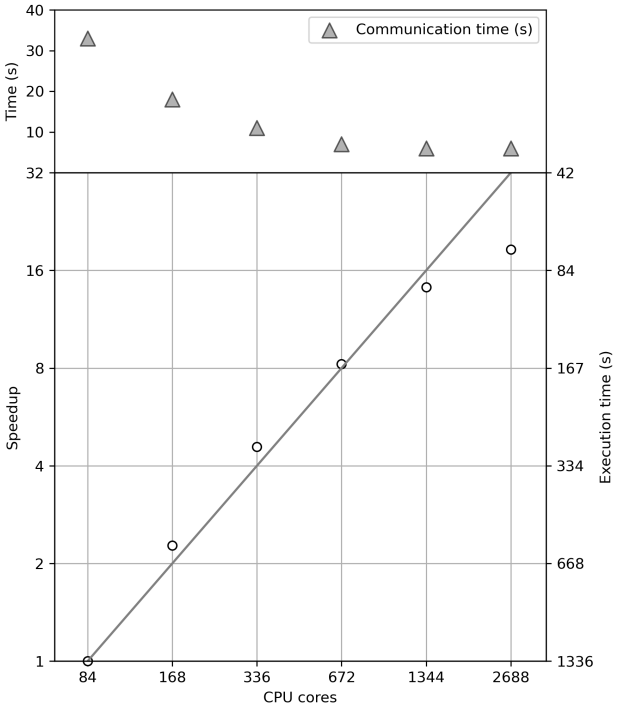

Figure 4. MPI scalability of WAM6 for a global 0.1°, 1-day wave hindcast case (T1) on NMEFC's HPC cluster with Intel Xeon E5-2680 v4 processors. Diagonal line denotes linear scalability. Both speedup (left y-axis) and corresponding execution time in second (right y-axis) are shown by open cycles.

The execution times are 72.0 and 74.5 second separately when 2688 and 2048 cores are allocated at each cluster, corresponding to speedup ratios of 33 and 32. The GPU-accelerated WAM6 model shows its superiority over the original code with regard to computing efficiency. It should be noted that how to showcase the performance of GPU acceleration over CPU is somewhat difficult. The reported speedup ratios vary in terms of the hardware and workload.

For operational 7-day wave forecast, the global model usually adopts reduced Gaussian grid and spectral resolution of 36 directions and 35 frequencies, which is similar to the T2 case in the study. Therefore, the energy consumption (E) of a 7-day global wind forecast using T2 case is measured by $E = P_{chip} \cdot N \cdot T$ in terms of chip's thermal design power (TDP). P_{chip} is the TDP value of the GPU/CPU chip, N is the number of devices, and T is the computing time of the model. For NVIDIA A100 GPU and Intel Xeon 2680v4 CPU, the TDP values are 350 and 120 watt, respectively. In the NMEFC, the operational global wave model runs for 2.5 hours on the NMEFC's cluster (Table 1) with 560 cores. A global wave forecast consumes 0.84 kWh on the 8-card GPU server (Table 3), and approximately 12 kWh on the NMEFC's cluster. That means at least 90% of power could be saved if a 8-card GPU server is used.

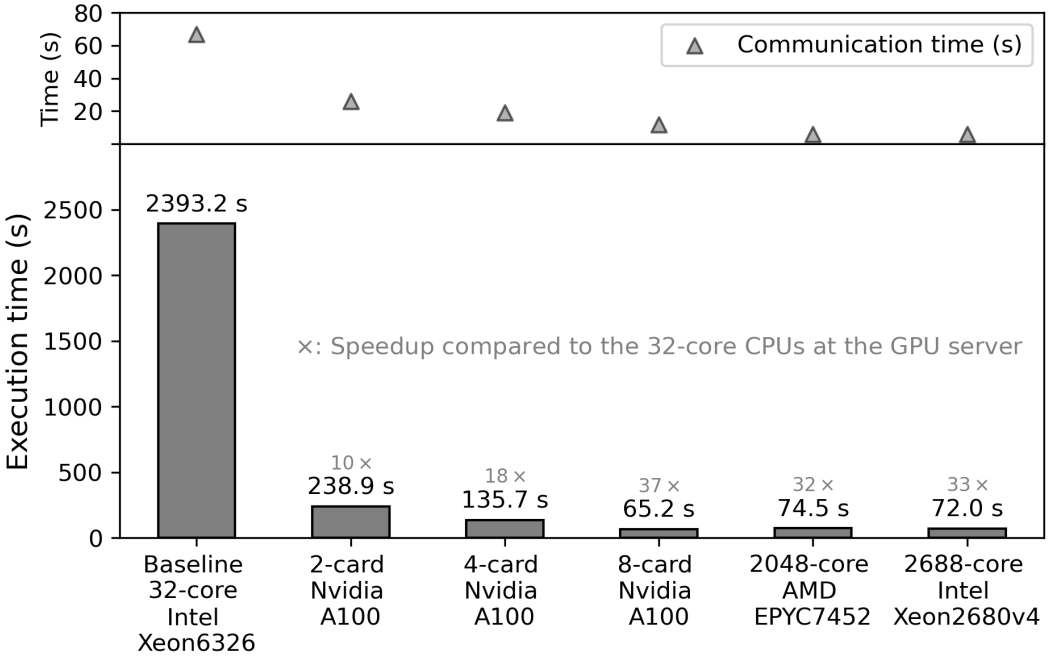

Figure 5. Speedup ratios of the WAM6-GPU v1.0 on 2, 4 and 8 NVIDIA A100 cards over its original code on a 32-core node with 2 Intel Xeon 6326 processors for 1-day global 0.1° wave hindcast. The computing time of the CPU code on the NMEFC's and BJSC's HPC clusters is also included. Speedup ratios are labeled over the bars. MPI communication time is denoted as solid triangles.

Table 3. GPU memory and computing time required for a 7-day global 0.1° wave forecast on the GPU server[a].

Model configurations	GPU memory (GB)	Computing time (min)
T1[b]	102.1	7.6
T2	199.9	17.9
T3	312.8	49.2

[a]Specification of the GPU server is presented in Table 1.

[b]Model configurations are presented in Table 2.

4.3 GPU profiling

The major processes of the WAM6 and its GPU version are profiled and their performance is illustrated by timeline plots in Figure 6. The WAM6 runs with 2 CPUs with 32 cores, and its GPU version runs with 4 GPUs. In T1 case, the time intervals for wave propagation, source term integration, and wave parameter computation are 43 second, 5 minute, and 3 hour, respectively. The profiling of wave propagation and source term integration in a 5-minute period is presented in Figure 6a.

At each integration time step, the model begins with MPI exchange of 3D spectral energy, which is followed by the computation on propagation and source terms. Generally, a complete model integration on CPU takes 8.589 second while its GPU counterpart takes only 0.435 second on 4 A100 cards. This corresponds to a speedup ratio of 20. In more details, for original CPU version, wave propagation and wind input (S_{in}) are 2 major computing-intensive processes, which contributes 27.9% and 38.0% of the runtime, respectively. For GPU version, the most prominent feature is the share of nonlinear transfer (S_{nl}) soars to 36.4%. By adopting optimizing strategies introduced in Section 3.2, OpenACC implementation of wave propagation and S_{in} can take full advantage of GPU resource. On the contrary, the computation of S_{nl} requires 5 layers of loops, and the dependency along frequency and direction inhibits multi-level parallelism. The original algorithm only allows parallelization on IJ loop. An naive implementation of OpenACC results in very poor performance and substantially lower speedup ratio. Extensive optimizing experiments on S_{nl} have been made to reduce its share from nearly 60% to 36.4%. Among these experiments, three of them (Exp1 - 3) are summarized below, with their time measurements shown in bracket. The pseudocode is shown in Figure 7.

1. Exp1: Placing the IJ loop to the innermost (Not adopted; 0.214 s): It leads to too much overhead for serially launching thousands of kernels in a single time step.

2. Exp2: Placing the IJ loop to the outermost and accessing global coefficient and index arrays directly (Not adopted; 0.252 s): It leads to lower GPU occupancy and higher GPU latency due to spilling of local memory, and frequent access of these arrays within a kernel are detrimental to performance.

3. Exp3: Placing the IJ loop between the M (frequency) and K (direction) loops (Adopted; 0.171 s for loop collapse on IJ and K, and 0.151 s for parallelism on IJ and sequential execution on K): It overcomes the shortcomings of the above experiments. Besides, actually two tests have been conducted in Exp3. By reorganizing the code substantially, we managed to collapse the IJ and K loops at first. As the second test, we did not do the loop collapse, and simply inserted ⟨*acc loop seq*⟩ before the nested K loop. Surprisingly, the second test took 0.02 s less time. Although loop collapse on IJ and K may increase code parallelism, it seems that the reorganized code leads to increased overhead.

Furthermore, in contrast with the common sense that MPI exchange is a limiting factor for speedup, in this study its GPU implementation exhibits a considerable speedup, with its share reducing from 2.8% to 1.5%. The best practice is to pack and decode the MPI buffers through GPU acceleration, and transfer them using ⟨*acc host_data*⟩ construct.

The WAM6 has several postprocessing modules to derive a series of wave parameters from wave spectra at each output time step (ECMWF, 2023). Some of the derived parameters serves as interface variables for coupling with atmospheric, oceanic and sea ice models (Breivik et al., 2015; Roberts et al., 2018). It is found that even though these modules are only used at each output time step (normally 1-3 hours), the device-to-host transfer of the 3D spectral energy is so expensive that the overall performance can be affected. Besides, as shown in the timeline, the postprocessing runs 6.068 second on the CPU node. Considering the postprocessing is both computing-intensive and I/O-bound, these modules are ported to the GPU in the study. The profiling of these modules is shown in Figure 6b. The postprocessing is categorized as 4 types shown in the figure legend.

GPU implementation exhibits acceptable acceleration with a speedup metric of 12. As expected, the spectral separation take the largest share of the runtime (77.0%), as it involves a succession of frequency-dependent loops. Not much optimization has been done at this time.

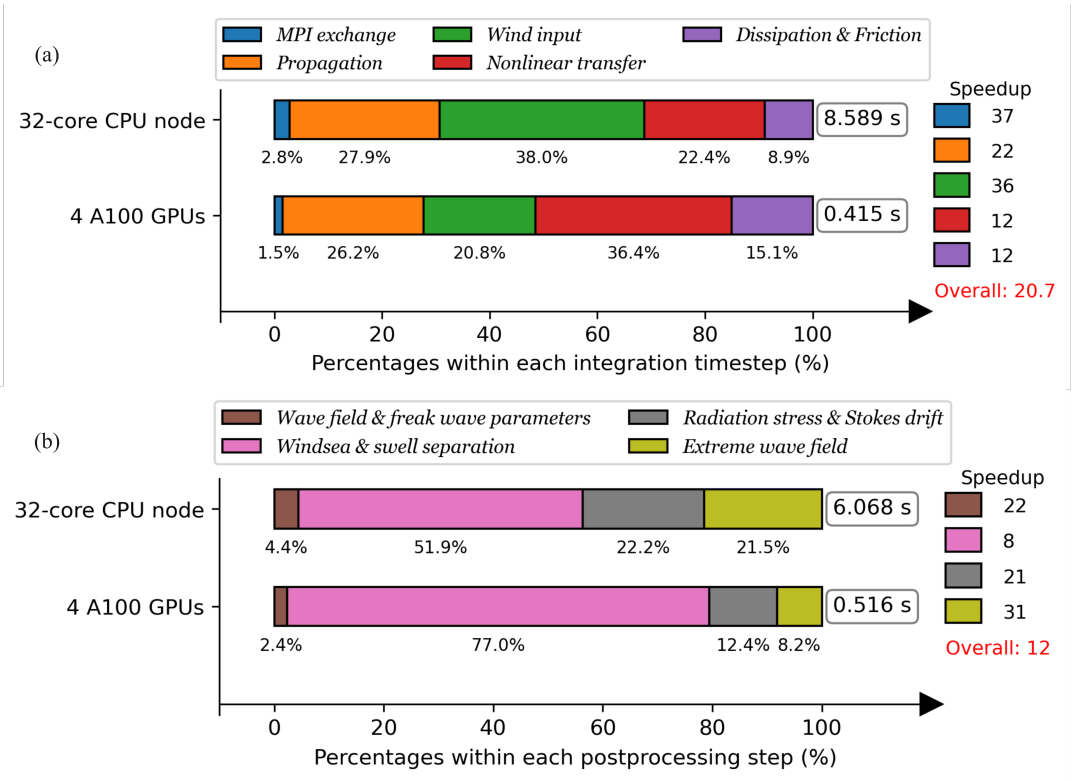

Figure 6. GPU runtime analysis within the integration (a) and postprocessing (b) time steps for a global 0.1° wave hindcast case.

4.4 Comparison of CPU and GPU results

The numerical models running on CPU and GPU are supposed to produce slightly different results due to multiple factors. For example, the execution order of the arithmetic operations are different on CPU and GPU for a global reduction. Besides, iterative algorithms lead to the accumulation of the errors. Substantive difference is not allowed, and ideally it should not grow rapidly for a long-term simulation. Absolute difference ($AD = CPU - GPU$) and Relative difference ($RD = (CPU - GPU)/CPU$) are used to measure the difference between the CPU and GPU results. Both the WAM6 and its GPU version run with single precision for one-month period.

As many as 72 different wave parameters can be computed by the WAM6-GPU v1.0, and some of them are visualized in Figure 8. The field-mean AD and RD values for a one-day period are presented in Figure 9 as gray cycles and open triangles, respectively. Generally, GPU implementation is proven to be a success, which is demonstrated by the well-controlled

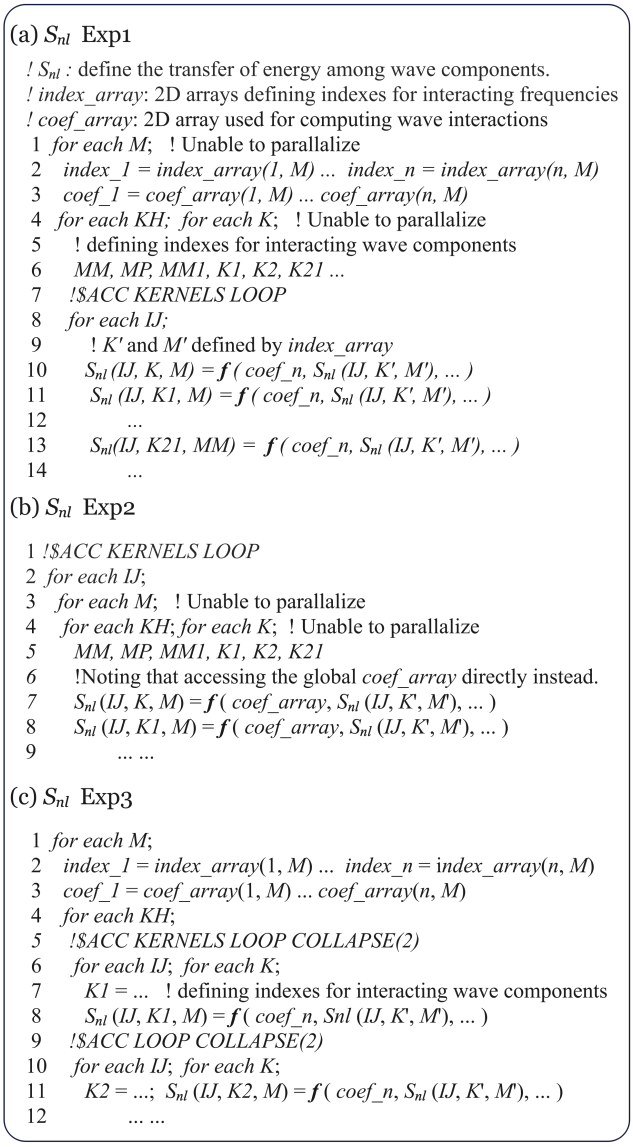

Figure 7. Pseudocode of optimizing experiments on source term describing nonlinear wave interaction (S_{nl}). Algorithms (a)-(c) correspond to Exp1 - 3 separately.

differences. The mean AD values are well below 10^{-6} with several outliers observed for some output parameters with their values several orders of magnitude larger than others (i.e., radiation stress components and wave direction in degree). In this case the RD metric is more appropriate for comparison. Time series of mean relative differences (RD) from January 1 to 30, 2021 for significant wave height, peak & mean wave period, and wave direction are shown in Figure 10. Two dashed lines

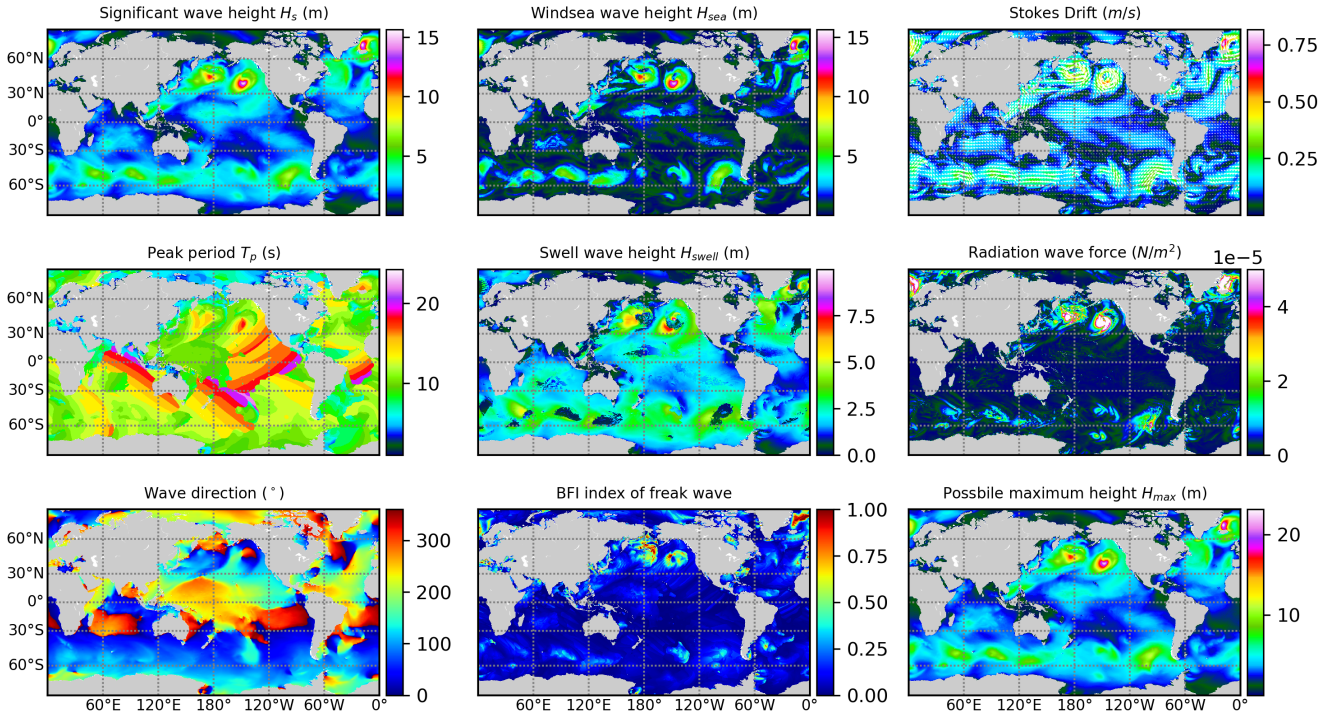

Figure 8. A series of output parameters generated by the WAM6-GPU v1.0.

marking the RD values of 10^{-6} are plotted. It is satisfying that the differences between CPU and GPU results do not show a growing trend for the whole simulation.

5 Conclusions

In recent years giant HPC clusters have been replaced by dense and green ones in an increasing pace. Moreover, programming barrier to port numerical models to GPUs are now incredibly low with the aid of high-level programming tools such as OpenACC. These progresses pave fast lane to energy-efficient computing for climate and earth system modeling. It is anticipated that the transition of ESMs from CPU-based infrastructure to GPU-based computing devices is inevitable.

In this study, the WAM6, one of the most commonly-used spectral ocean wave models, has been ported to multiple GPUs by OpenACC with considerable efforts of code restructuring and refactoring. To evaluate model performance, a global wave hindcast case has been configured with horizontal resolution of 1/10°, and intra-spectrum resolution of 25×24 or 35×36. It shows that when using an 8-card GPU server (NVIDIA A100), a speedup of 37 over a fully-loaded CPU node can be achieved, which reduces the computing time of a 7-day global wave modeling from more than 2 hours to 7.6 minutes. We expect even higher performance of the WAM6-GPU v1.0 on the latest NVIDIA H100 GPU. The study emphasizes that for a complicated

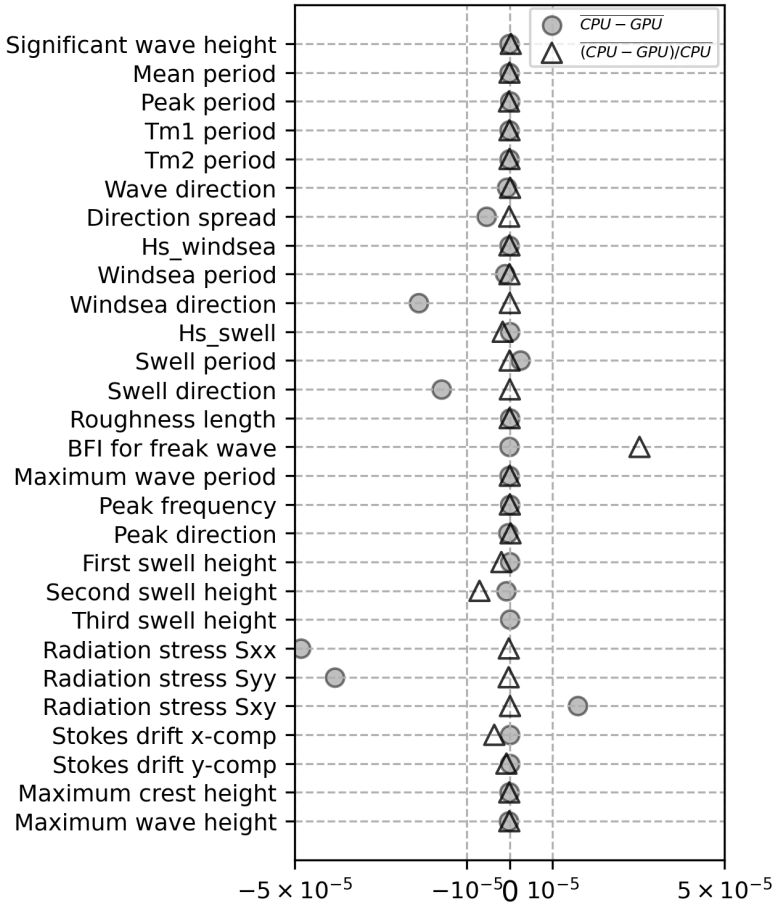

Figure 9. Mean difference (cycles) and mean relative differences (triangles) between the WAM6 and WAM6-GPU v1.0 for major output parameters for a one-day global simulation. Both versions use single precision for computation. Outliers in mean difference can be observed for parameters with several orders of magnitude larger than others.

model, GPU performance may not be desirable if OpenACC directives are implemented without substantial consideration on GPU's features and their combination with code itself.

To move the study forward, the ongoing works include adding a new finite-volume propagation scheme to support Voronoia unstructured grid, and incorporating seaice-wave interaction source term to the WAM6's physics. These works will extend the application of the code to the polar areas and nearshore.

Code and data availability. The WAM6-GPU is maintained at https://doi.org/10.5281/zenodo.10453368 (Yuan, 2024). The latest version is v1.2. Assistance can be provided through yuanye@nmefc.cn. The original WAM6 source code is maintained at https://github.com/mywave/WAM.

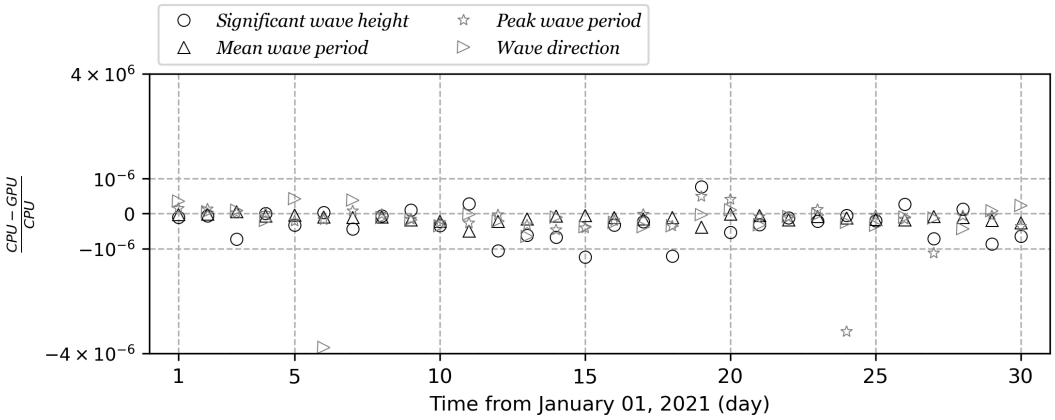

Figure 10. Time series of mean relative differences (RD) between the WAM6 and its GPU version from January 1 to 30, 2021 for significant wave height (circles), peak wave period (stars), mean wave period (triangle), and wave direction (left-pointing triangles).

Author contributions. The WAM6 was GPU-accelerated by Ye Yuan, Yuanyong Gao, and Fang Hou. The manuscript was prepared by Ye Yuan. Fujiang Yu and Zhi Chen conceived the research and revised the manuscript. The validation was carried out by Xueding Li and Zhiyi Gao.

Competing interests. The authors declare that they have no conflict of interest.

Acknowledgements. This work was funded by the National Science and Technology Major Project of China (2023YFC3107803) and Innovative Youth Talents Program, Ministry of Natural Resources of China. The authors would like to thank Dr. Arno Behrens at Helmholtz-Zentrum Geesthacht for his support on model developments.

References

Alves, J.-H. G. M., Wittmann, P., Sestak, M., Schauer, J., Stripling, S., Bernier, N. B., McLean, J., Chao, Y., Chawla, A., Tolman, H., Nelson, G., and Klotz, S.: The NCEP–FNMOC Combined Wave Ensemble Product: Expanding Benefits of Interagency Probabilistic Forecasts to the Oceanic Environment, Bulletin of the American Meteorological Society, 94, 1893 – 1905, https://doi.org/10.1175/BAMS-D-12-00032.1, 2013.

Ardhuin, F., Rogers, E., Babanin, A. V., Filipot, J.-F., Magne, R., Roland, A., van der Westhuysen, A., Queffeulou, P., Lefevre, J.-M., Aouf, L., and Collard, F.: Semiempirical Dissipation Source Functions for Ocean Waves. Part I: Definition, Calibration, and Validation, Journal of Physical Oceanography, 40, 1917 – 1941, https://doi.org/10.1175/2010JPO4324.1, 2010.

Bao, Y., Song, Z., and Qiao, F.: FIO-ESM Version 2.0: Model Description and Evaluation, Journal of Geophysical Research: Oceans, 125, e2019JC016 036, https://doi.org/10.1029/2019JC016036, 2020.

Baordo, F., Clementi, E., Iovino, D., and Masina, S.: Intercomparison and assessement of wave models at global scale, technical notes, CMCC/ODA, Italy, 2020.

Behrens, A. and Janssen, P.: Documentation of a web based source code library for WAM, 2013.

Booij, N., Ris, R. C., and Holthuijsen, L. H.: A third-generation wave model for coastal regions: 1. Model description and validation, Journal of Geophysical Research: Oceans, 104, 7649–7666, https://doi.org/10.1029/98JC02622, 1999.

Breivik, Ø., Mogensen, K., Bidlot, J.-R., Balmaseda, M. A., and Janssen, P. A. E. M.: Surface wave effects in the NEMO ocean model: Forced and coupled experiments, Journal of Geophysical Research: Oceans, 120, 2973–2992, https://doi.org/10.1002/2014JC010565, 2015.

Brus, S. R., Wolfram, P. J., Van Roekel, L. P., and Meixner, J. D.: Unstructured global to coastal wave modeling for the Energy Exascale Earth System Model using WAVEWATCH III version 6.07, Geoscientific Model Development, 14, 2917–2938, https://doi.org/10.5194/gmd-14-2917-2021, 2021.

Bunney, C. and Saulter, A.: An ensemble forecast system for prediction of Atlantic–UK wind waves, Ocean Modelling, 96, 103–116, https://doi.org/10.1016/j.ocemod.2015.07.005, 2015.

Cavaleri, L., Alves, J.-H., Ardhuin, F., Babanin, A., Banner, M., Belibassakis, K., Benoit, M., Donelan, M., Groeneweg, J., Herbers, T., Hwang, P., Janssen, P., Janssen, T., Lavrenov, I., Magne, R., Monbaliu, J., Onorato, M., Polnikov, V., Resio, D., Rogers, W., Sheremet, A., McKee Smith, J., Tolman, H., van Vledder, G., Wolf, J., and Young, I.: Wave modelling – The state of the art, Progress in Oceanography, 75, 603–674, https://doi.org/10.1016/j.pocean.2007.05.005, 2007.

Cavaleri, L., Fox-Kemper, B., and Hemer, M.: Wind Waves in the Coupled Climate System, Bulletin of the American Meteorological Society, 93, 1651 – 1661, https://doi.org/10.1175/BAMS-D-11-00170.1, 2012.

Couvelard, X., Lemarié, F., Samson, G., Redelsperger, J.-L., Ardhuin, F., Benshila, R., and Madec, G.: Development of a two-way-coupled ocean–wave model: assessment on a global NEMO(v3.6)-WW3(v6.02) coupled configuration, Geoscientific Model Development, 13, 3067–3090, https://doi.org/10.5194/gmd-13-3067-2020, 2020.

Danabasoglu, G., Lamarque, J.-F., Bacmeister, J., Bailey, D. A., DuVivier, A. K., Edwards, J., Emmons, L. K., Fasullo, J., Garcia, R., Gettelman, A., Hannay, C., Holland, M. M., Large, W. G., Lauritzen, P. H., Lawrence, D. M., Lenaerts, J. T. M., Lindsay, K., Lipscomb, W. H., Mills, M. J., Neale, R., Oleson, K. W., Otto-Bliesner, B., Phillips, A. S., Sacks, W., Tilmes, S., van Kampenhout, L., Vertenstein, M., Bertini, A., Dennis, J., Deser, C., Fischer, C., Fox-Kemper, B., Kay, J. E., Kinnison, D., Kushner, P. J., Larson, V. E., Long, M. C., Mickelson, S., Moore, J. K., Nienhouse, E., Polvani, L., Rasch, P. J., and Strand, W. G.: The Community Earth System Model Version 2 (CESM2), Journal of Advances in Modeling Earth Systems, 12, e2019MS001 916, https://doi.org/10.1029/2019MS001916, 2020.

de la Asunción, M. and Castro, M. J.: Simulation of tsunamis generated by landslides using adaptive mesh refinement on GPU, Journal of Computational Physics, 345, 91–110, https://doi.org/10.1016/j.jcp.2017.05.016, 2017.

ECMWF: IFS Documentation CY48R1 - Part VII: ECMWF wave model, 7, ECMWF, https://doi.org/10.21957/cd1936d846, 2023.

Fan, Y. and Griffies, S. M.: Impacts of Parameterized Langmuir Turbulence and Nonbreaking Wave Mixing in Global Climate Simulations, Journal of Climate, 27, 4752 – 4775, https://doi.org/10.1175/JCLI-D-13-00583.1, 2014.

Günther, H., Hasselmann, S., and Janssen, P.: The WAM Model cycle 4, https://doi.org/10.2312/WDCC/DKRZ_Report_No04, 1992.

Häfner, D., Nuterman, R., and Jochum, M.: Fast, Cheap, and Turbulent—Global Ocean Modeling With GPU Acceleration in Python, Journal of Advances in Modeling Earth Systems, 13, e2021MS002 717, https://doi.org/10.1029/2021MS002717, 2021.

Hersbach, H., Bell, B., Berrisford, P., Hirahara, S., Horányi, A., Muñoz-Sabater, J., Nicolas, J., Peubey, C., Radu, R., Schepers, D., Simmons, A., Soci, C., Abdalla, S., Abellan, X., Balsamo, G., Bechtold, P., Biavati, G., Bidlot, J., Bonavita, M., De Chiara, G., Dahlgren, P., Dee, D., Diamantakis, M., Dragani, R., Flemming, J., Forbes, R., Fuentes, M., Geer, A., Haimberger, L., Healy, S., Hogan, R. J., Hólm, E., Janisková, M., Keeley, S., Laloyaux, P., Lopez, P., Lupu, C., Radnoti, G., de Rosnay, P., Rozum, I., Vamborg, F., Villaume, S., and Thépaut, J.-N.: The ERA5 global reanalysis, Quarterly Journal of the Royal Meteorological Society, 146, 1999–2049, https://doi.org/10.1002/qj.3803, 2020.

Huang, C. J., Qiao, F., Song, Z., and Ezer, T.: Improving simulations of the upper ocean by inclusion of surface waves in the Mellor-Yamada turbulence scheme, Journal of Geophysical Research: Oceans, 116, https://doi.org/10.1029/2010JC006320, 2011.

Ikuyajolu, O. J., Van Roekel, L., Brus, S. R., Thomas, E. E., Deng, Y., and Sreepathi, S.: Porting the WAVEWATCH III (v6.07) wave action source terms to GPU, Geoscientific Model Development, 16, 1445–1458, https://doi.org/10.5194/gmd-16-1445-2023, 2023.

Ikuyajolu, O. J., Roekel, L. V., Brus, S. R., Thomas, E. E., Deng, Y., and Benedict, J. J.: Effects of Surface Turbulence Flux Parameterizations on the MJO: The Role of Ocean Surface Waves, Journal of Climate, https://doi.org/10.1175/JCLI-D-23-0490.1, in press, 2024.

Iwasaki, S. and Otsuka, J.: Evaluation of Wave-Ice Parameterization Models in WAVEWATCH III Along the Coastal Area of the Sea of Okhotsk During Winter, Frontiers in Marine Science, 8, https://doi.org/10.3389/fmars.2021.713784, 2021.

Janssen, P. A. E. M.: Wave-Induced Stress and the Drag of Air Flow over Sea Waves, Journal of Physical Oceanography, 19, 745 – 754, https://doi.org/10.1175/1520-0485(1989)019<0745:WISATD>2.0.CO;2, 1989.

Janssen, P. A. E. M.: Quasi-linear Theory of Wind-Wave Generation Applied to Wave Forecasting, Journal of Physical Oceanography, 21, 1631 – 1642, https://doi.org/10.1175/1520-0485(1991)021<1631:QLTOWW>2.0.CO;2, 1991.

Janssen, P. A. E. M.: Ocean wave effects on the daily cycle in SST, Journal of Geophysical Research: Oceans, 117, https://doi.org/10.1029/2012JC007943, 2012.

Jiang, J., Lin, P., Wang, J., Liu, H., Chi, X., Hao, H., Wang, Y., Wang, W., and Zhang, L.: Porting LASG/ IAP Climate System Ocean Model to GPUs Using OpenACC, IEEE Access, 7, 154 490–154 501, https://doi.org/10.1109/ACCESS.2019.2932443, 2019.

Qiao, F., Song, Z., Bao, Y., Song, Y., Shu, Q., Huang, C., and Zhao, W.: Development and evaluation of an Earth System Model with surface gravity waves, Journal of Geophysical Research: Oceans, 118, 4514–4524, https://doi.org/10.1002/jgrc.20327, 2013.

Qin, X., LeVeque, R. J., and Motley, M. R.: Accelerating an adaptive mesh refinement code for depth-averaged flows using graphics processing units (GPUs), Journal of Advances in Modeling Earth Systems, 11, 2606–2628, https://doi.org/10.1029/2019ms001635, 2019.

Roberts, C. D., Senan, R., Molteni, F., Boussetta, S., Mayer, M., and Keeley, S. P. E.: Climate model configurations of the ECMWF Integrated Forecasting System (ECMWF-IFS cycle 43r1) for HighResMIP, Geoscientific Model Development, 11, 3681–3712, https://doi.org/10.5194/gmd-11-3681-2018, 2018.

The Wamdi Group: The WAM Model—A Third Generation Ocean Wave Prediction Model, Journal of Physical Oceanography, 18, 1775 – 1810, https://doi.org/10.1175/1520-0485(1988)018<1775:TWMTGO>2.0.CO;2, 1988.

Tolman, H.: User manual and system documentation of WAVEWATCH III TM version 3.14, technical notes, NOAA / NWS / NCEP / OMB, 1999.

Valiente, N. G., Saulter, A., Gomez, B., Bunney, C., Li, J.-G., Palmer, T., and Pequignet, C.: The Met Office operational wave forecasting system: the evolution of the regional and global models, Geoscientific Model Development, 16, 2515–2538, https://doi.org/10.5194/gmd-16-2515-2023, 2023.

Williams, T. D., Bennetts, L. G., Dumont, D., Squire, V. A., and Bertino, L.: Towards the inclusion of wave-ice interactions in large-scale models for the Marginal Ice Zone, https://doi.org/10.48550/arXiv.1203.2981, 2012.

Xiao, H., Sun, J., Bian, X., and Dai, Z.: GPU acceleration of the WSM6 cloud microphysics scheme in GRAPES model, Computers & Geosciences, 59, 156–162, https://doi.org/10.1016/j.cageo.2013.06.016, 2013.

Xu, S., Huang, X., Oey, L.-Y., Xu, F., Fu, H., Zhang, Y., and Yang, G.: POM.gpu-v1.0: a GPU-based Princeton Ocean Model, Geoscientific Model Development, 8, 2815–2827, https://doi.org/10.5194/gmd-8-2815-2015, 2015.

Yuan, Y.: The WAM6-GPU: an OpenACC version of the third- generation spectral wave model WAM (Cycle 6), https://doi.org/https://doi.org/10.5281/zenodo.10453368, 2024.

Yuan, Y., Shi, F., Kirby, J. T., and Yu, F.: FUNWAVE-GPU: Multiple-GPU Acceleration of a Boussinesq-Type Wave Model, Journal of Advances in Modeling Earth Systems, 12, e2019MS001 957, https://doi.org/10.1029/2019MS001957, 2020.

Yuan, Y., Yang, H., Yu, F., Gao, Y., Li, B., and Xing, C.: A wave-resolving modeling study of rip current variability, rip hazard, and swimmer escape strategies on an embayed beach, Natural Hazards and Earth System Sciences, 23, 3487–3507, https://doi.org/10.5194/nhess-23-3487-2023, 2023.