**Authors' response to referee #1: egusphere-2024-169: "Towards a real-time modeling of global ocean waves by the fully GPU-accelerated spectral wave model WAM6-GPU"**

Hi, the anonymous referee,

Thank you for your constructive comments on our manuscript entitled 'Towards a real-time modeling of global ocean waves by the fully GPU-accelerated spectral wave model WAM6-GPU'. We feel indebted to you for your time on this manuscript. In the response below, all the comments and concerns are replied point by point, and the revised manuscript is attached as PDF supplements.

Kind Regards,
Ye Yuan, on behalf of the co-authors.

---

***I wish to congratulate the authors on a job well done. This is a very comprehensive piece of work, and the speedups achieved on GPU are indeed impressive. What I particularly enjoyed was the level of detail in which code optimisations were discussed. Some of the computational and memory access patterns present in WAM6 are also relevant to many scientific algorithms, and thus the optimisations demonstrated herein have a wider utility beyond the wave modelling community.***

***I have attached an annotated copy of the paper with some detailed comments. I have mainly requested further clarifications and/or evidence for some of the points made in the paper. The most significant of these is the request for further detail into the optimisation of the non-linear wave interaction. There are also a few typographical and grammar corrections.***

***1. Section 3.1: Can you please explain how unstructured data directives would prove inconvenient for model developers? Alternatively you could also simply remove this statement; you don't need it to justify the use of sturctured data directives.***

**Response**: I have removed the statement "[Line 122 in the revised manuscript] *The unstructured data region <acc enter/exit data}> is not used in the study, though it can create data region spanning different routines. It probably results in inconvenience for potential model developers.*". My personal experience was that some of my colleagues who were unfamiliar with the OpenACC might forget to complete the unstructured data directives that were spanning multiple subroutines.

***2. Line 151: While I do agree that adding simple openacc directives around the outer loop and relying simply on acc routine directives to deal with nested functions would lead to poor performance, this is not the same as saying GPU kernel size necessarily has a detrimental effect on performance. I suspect modern GPU compilers with link-time optimisations would go a long way towards mitigating any performance penalty related to large kernels with nested function calls. Therefore, this statement should either be removed or backed up by an example comparing an optimised large kernel versus many smaller ones.***

**Response**: I agree with your comments. Thanks a lot. I removed the statement from the manuscript [Line 158 in the revised manuscript]. Here I try to express that a very huge GPU kernel may claim

more on-chip resource, thus lower the GPU occupancy. My previous experience on FUNWAVE-GPU project with CUDA Fortran told me to reduce the GPU-kernel complexity when possible. Besides, this comment is related to the Comment 6. I have made time measurements of the experiments that are made for $S_{nl}$ term. Please refer to the response for Comment 6.

***3. Figure 3: why is there a '?' above the optimisation operator in subplot 3a? The RHS of figure 3b should read "compute MxKxIJ times"***

**Response**: The question mark which I placed here is explained between Line 156-161 in the revised manuscript. Here I would like to express the concerns that in some occasions this code refactoring is not appropriate, or there is no universal rule for code refactoring in GPU implementation. In the revised manuscript, we decide to remove the question mark [Figure 3 Algorithm (a) in the revised manuscript].

The typing error in the RHS of Figure 3b has been corrected. Thank you. Please also refer to the Figure R1 below.

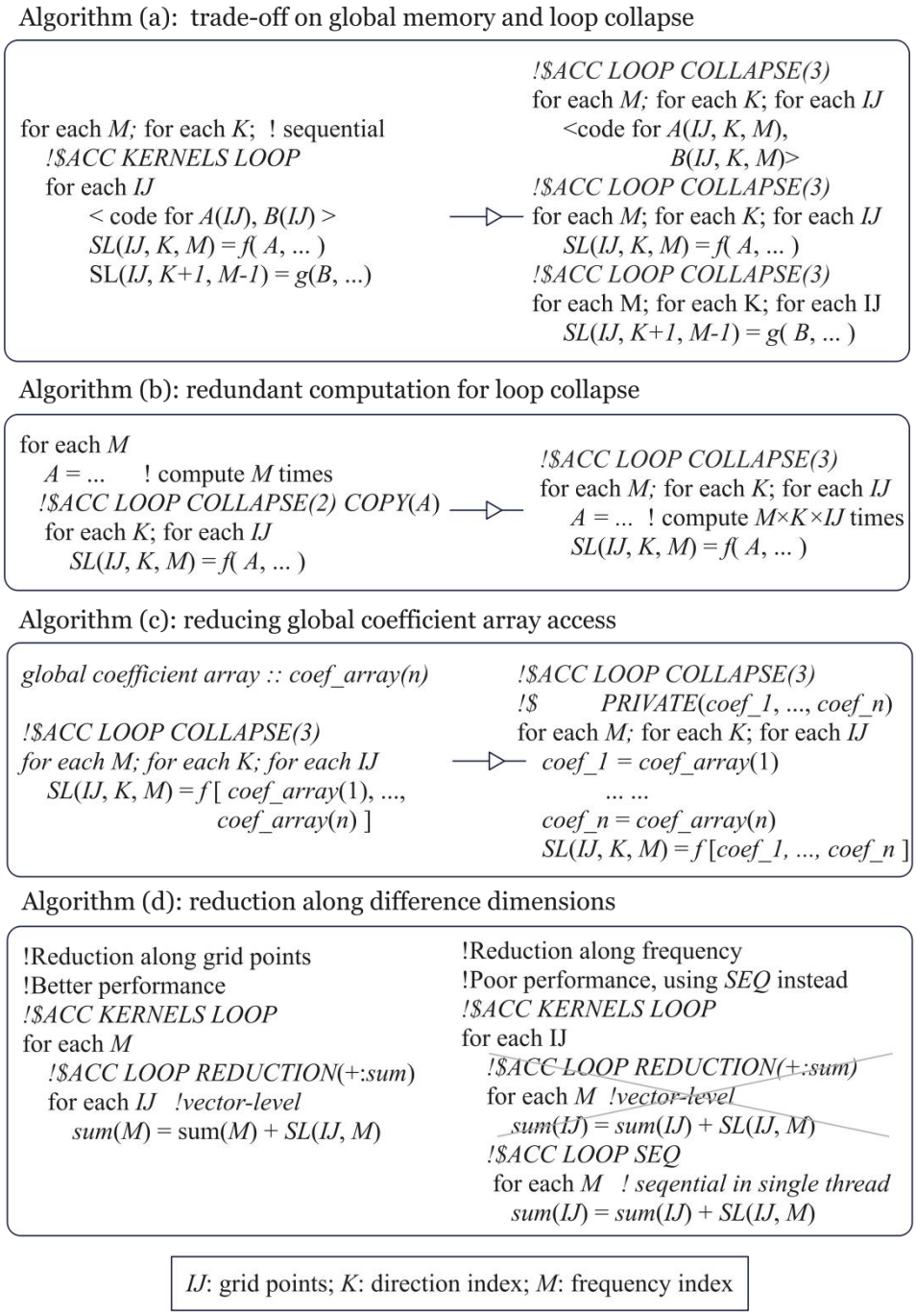


Figure R1. Optimizing strategies to improve performance in the WAM6-GPU.

***4. Line 187: Does this mean the CPU baseline was established using only one of the two Intel xeon chips in each node? Please clarify.***

**Response**: Each Intel Xeon CPU has 16 physical cores. Thus in our single-node GPU server, the total number of the CPU cores are 32. To avoid misunderstanding, the sentence is rephrased as,

[Line 195 in the revised manuscript] *"The GPU server is hosted by 2 Intel Xeon 6236 CPUs with 32 (2 ×16) physical cores running at 2.9 GHz."*

***5. Line 216: "Averagely" is incorrect English. The sentence can be rephrased as follows: "The average wall-time taken to run a 1-day forecast is 2393.2 seconds."***

**Response**: Adopted. The sentence has been rephrased as,

[Line 225 in the revised manuscript] *"The average wall-time taken to run a 1-day forecast is 2393.2 seconds."*

***6. Line 249-255: Considering that the non-linear wave interaction is the most expensive kernel on GPU, this section should be backed up with code examples detailing the three optimisation levels. Moreover, given that clearly a great deal of performance analysis has been carried out, key metrics from GPU profiles (e.g. occupancy) of the three approaches should also be shown.***

**Response**: Adopted. By combining two reviewers' suggestions, the subsection 4.3 has been supplemented with the pseudocode and the wall time measurement for each experiment, which are also shown here [Line 250-274 in Section 4.3].

- *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.*
- *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 global arrays within a kernel is detrimental to performance.*
- *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.*

(a) $S_{nl}$ Exp1

```
! S_nl : define the transfer of energy among wave components.
! index_array: 2D arrays defining indexes for interacting frequencies
! coef_array: 2D array used for computing wave interactions
 1  for each M;   ! Unable to parallalize
 2    index_1 = index_array(1, M) ...  index_n = index_array(n, M)
 3    coef_1 = coef_array(1, M) ... coef_array(n, M)
 4   for each KH;  for each K;   ! Unable to parallalize
 5     ! defining indexes for interacting wave components
 6     MM, MP, MM1, K1, K2, K21 ...
 7     !$ACC KERNELS LOOP
 8     for each IJ;
 9        ! K' and M' defined by index_array
10       S_nl (IJ, K, M) = f ( coef_n, S_nl (IJ, K', M'), ... )
11        S_nl (IJ, K1, M) = f ( coef_n, S_nl (IJ, K', M'), ... )
12              ...
13        S_nl(IJ, K21, MM) =  f ( coef_n, S_nl (IJ, K', M'), ... )
14              ...
```

(b) $S_{nl}$ Exp2

```
1 !$ACC KERNELS LOOP
2  for each IJ;
3    for each M;   ! Unable to parallalize
4     for each KH; for each K;  ! Unable to parallalize
5       MM, MP, MM1, K1, K2, K21
6       !Noting that accessing the global coef_array directly instead.
7       S_nl (IJ, K, M) = f ( coef_array, S_nl (IJ, K', M'), ... )
8       S_nl (IJ, K1, M) = f ( coef_array, S_nl (IJ, K', M'), ... )
9             ... ...
```

(c) $S_{nl}$ Exp3

```
 1  for each M;
 2    index_1 = index_array(1, M) ...  index_n = index_array(n, M)
 3    coef_1 = coef_array(1, M) ... coef_array(n, M)
 4    for each KH;
 5     !$ACC KERNELS LOOP COLLAPSE(2)
 6      for each IJ;  for each K;
 7        K1 = ...   ! defining indexes for interacting wave components
 8        S_nl (IJ, K1, M) = f ( coef_n, Snl (IJ, K', M'), ... )
 9     !$ACC LOOP COLLAPSE(2)
10     for each IJ;  for each K;
11       K2 = ...;  S_nl (IJ, K2, M) = f ( coef_n, S_nl (IJ, K', M'), ... )
12             ... ...
```

Figure R2. Pseudocode of optimizing experiments on source term describing nonlinear wave interaction ($S_{nl}$)