Enhancing DEVITO GPU Allocator Using Unified Memory by NVIDIA

Gustavo Araujo Alvaro Coelho[1*], Atila Saraiva Quintela Soares[1], João Henrique Speglich[1], Marcelo Oliveira da Silva[1]
[1]Supercomputing Center for Industrial Innovation SENAI CIMATEC (CS2I); Salvador, Bahia, Brazil

DEVITO is a framework whose objective is to implement optimized stencil computing. Its execution can be carried out both in the CPU and in GPU. For this reason, the data must be manipulated correctly so that, in case of executions in the GPU, they are present in the memory of the GPU at the time of the execution. Natively, DEVITO transfers data every time the operator is executed from OpenACC pragmas. This approach results in performance degradation when the operator is executed repeatedly. To prevent redundant copies and alleviate this bottleneck, an allocator based on unified memory was implemented, which makes manual data transfer between CPU and GPU unnecessary, significantly reducing data transfer time in GPU applications.
Keywords: DEVITO. Unified Memory. GPU. Data Transfer.

Introduction

DEVITO [1,2] is a framework developed in Python, whose objective is to implement optimized stencil computation (e.g., finite differences, image processing). This tool uses Sympy code and automatic code generation, transforming the user's Python implementation into C code, which is lighter and faster to run computational kernels on different platforms, such as CPUs or GPUs. To perform data allocation, DEVITO implements different classes of memory allocators.

Each one with specific characteristics, from allocators based on the use of the POSIX library to allocators based on non-uniform memory access (NUMA). It is up to the user to choose an allocator that presents the best characteristics for his application. Given the available allocator options and their different characteristics, there is an essential similarity between them: they all allocate data within the CPU memory. Figure 1. Graphs illustrate the amount of data transfer between CPU and GPU before kernel execution. The blue bar represents kernel execution, and the green bar represents memory transfer between CPU and GPU.

Depending on the environment configuration, the DEVITO operator can run on both CPU and GPU. However, running on GPU requires data to be present within its memory region. For this reason, the device on which the data is initially allocated is crucial for the system's performance and the analysis of its functioning.

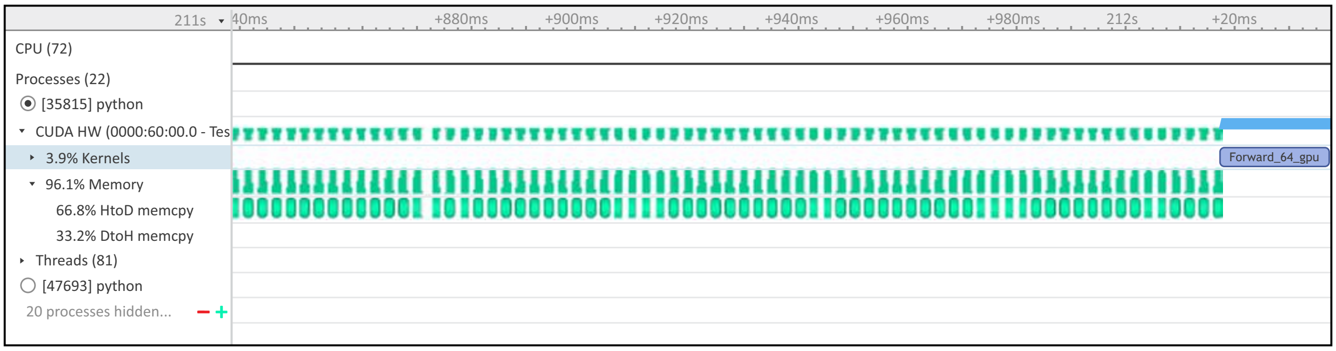
As DEVITO default allocators allocate data to the CPU, data transfer before and after operator computation is required. It uses pragma directives from the OpenACC library through the sub-directives copy in and copy out. The problem with this approach is that the data transfer process is defined within the operator, so if this operator is executed multiple times with the same arguments (e.g., execution using Checkpoint), data will also be transferred multiple times, generating redundant data stream that affects system performance. Figure 1 illustrates the amount of data transferred from CPU to GPU (Host to Device) before kernel execution.

Materials and Methods

In order to reduce the number of redundant copies mentioned above, a new allocator was created based on unified memory. Data allocated in unified memory can be accessed by CPU and GPU devices, making an exact copy of data unnecessary. Figure 2 illustrates the schematic unified memory,

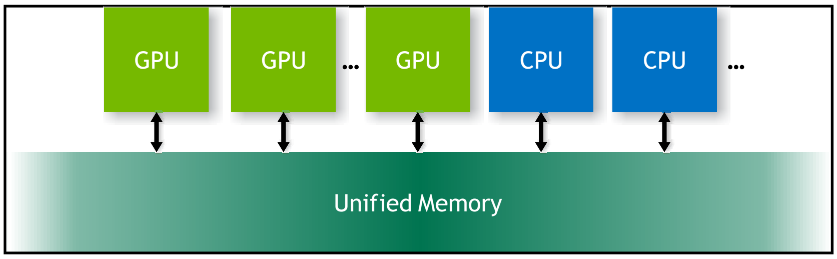
Received on 20 December 2022; revised 8 January 2023.
Address for correspondence: Gustavo Araujo Alvaro Coelho. Av. Artemia Pires Freitas, 8220, Cond. Viva Mais Master, Casa 6, Feira de Santana - BA. Zipcode: 44085-370. E-mail: gustavo.coelho@fieb.org.br.DOI 10.34178/jbth.v6iSuppl. 1.267.

Figure 1. Amount of data transfer between CPU and GPU prior to kernel execution.



Blue bar represents kernel execution and green bar represents memory transfer between CPU and GPU.

Figure 2. Schematic unified memory, which can be accessed by both CPU and GPU.



which can be accessed by both CPU and GPU [3]. The implementation of the new allocator was based on the use of CuPy [4], which, in addition to implementing unified memory in its structure, is very familiar with the NumPy library, which is widely used within DEVITO. The similarity between the two meant that implementing CuPy inside the DEVITO structure presented minimal difficulties.

The default allocator of CuPy API is set to use unified memory; any object created by it will have its data allocated in MU. Therefore, once the allocator was changed, a CuPy array of the same size as the desired data was created to allocate the data.

In order to analyze the obtained results and verify how much the unified memory improves the system performance, a test was applied. This test ran an algorithm responsible for direct propagation and adjunct calculation for a single source using the DEVITO and PyRevolve tools. This algorithm was run twice, once using the standard DEVITO allocator and once using the unified memory-based allocator.

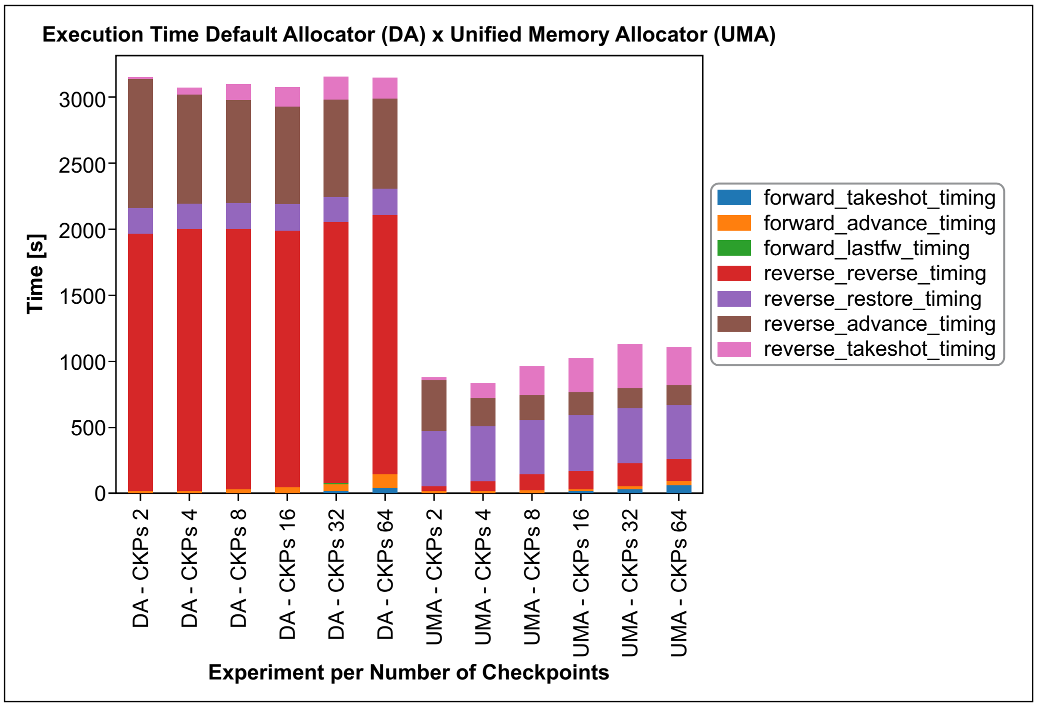
The following environmental variables were used to enable GPU execution with OpenACC:

- DEVITO_PLATFORM=nvidiaX;
- DEVITO_ARCH=nvc;
- DEVITO_LANGUAGE=openacc;

Results and Discussion

Based on the execution of the previously mentioned test, it was possible to evaluate the execution time of the algorithm. More specifically, the execution time of the forward and reverse methods, responsible for forward and reverse propagation, respectively. These are the methods that are most affected by the redundant transfer of data

Figure 3. Execution times of the overthrust script, varying the number of checkpoints, using the default DEVITO default allocator and the unified memory allocator.



when performing the checkpoint. Figure 3 shows the graph with the execution times of these functions. The unified memory allocation directly impacted the reverse propagation processing time, showing a significant decrease. This behavior is since using unified memory removes redundant copies of data made by the Devito tool. In addition, considering the entire process of calculating forward and reverse propagation, the application runtime with unified memory performs three times better than the standard implementation of the DEVITO tool.

Conclusion

DEVITO allocates data directly to the memory of CPU memory, making it necessary that, when running an application on the GPU, the data is transferred at each execution. As a result, when the operator is executed several times, data transfers become redundant, affecting application performance. A unified memory-based allocator was developed to solve this problem. Execution using this allocator makes data available to GPU and CPU, making redundant data transfers unnecessary. As a result, unified memory utilization showed significantly better results than the tool of the DEVITO default allocator.

References

1. Loubotin M. et al. DEVITO (v3.1.0): An embedded domain-specific language for finite differences and geophysical exploration. Geoscientific Model Development 2019;12(3):1165–1187.OpenACC Programming and Best Practices Guide. Available at: openacc.org. Accessed on: 20 Nov. 2022.
2. Luporini F. et al. Architecture and performance of DEVITO, a system for automated stencil computation. CoRR 2018;abs/1807.03032.
3. Harris M. Unified Memory for CUDA Beginners. Available at: <https://developer.nvidia.com/blog/unified-memory-cuda-beginners/>. 2017. Accessed on: 14 Dec. 2022.
4. Nisinho R, Loomis SHC. Cupy: A numpy-compatible library for nvidia gpu calculations. 31st Conference on Neural Information Processing Systems 2017;151(7).