Enhancing Devito GPU allocator using unified memory by Nvidia: An application in Elastic Reverse Time Migration (E-RTM)

Gustavo Araujo Alvaro Coelho*, Supercomputing Center for Industrial Innovation SENAI CIMATEC
Atila Saraiva Quintela Soares, Supercomputing Center for Industrial Innovation SENAI CIMATEC
João Henrique Speglich, Supercomputing Center for Industrial Innovation SENAI CIMATEC
Marcelo Oliveira da Silva, Supercomputing Center for Industrial Innovation SENAI CIMATEC
Peterson Nogueira Santos, Supercomputing Center for Industrial Innovation SENAI CIMATEC

Abstract

The Devito open-source version tool currently needs more effective memory management mechanisms for checkpointing technique-dependent applications running on GPUs. This deficiency can lead to significant performance degradation, particularly in memory-intensive applications that frequently access data stored in CPU memory. Devito employs a specific data allocation process to ensure that the objects defined by the tool are accessible in an equitable manner for both Python environment execution and the generated code execution by the Devito Operator. Devito has different ways of data allocation, and all of them are in CPU memory. Thus, when an application is executed in the GPU, it is necessary that, at each operator execution, the data be transmitted to the GPU memory. This CPU-GPU data transference makes Devito's memory management inefficient, mainly in high data movement applications, since the high amount of data transfer can impact the execution time. Running GPU applications performs data transfer between CPU and GPU every time a particular operator is executed. In situations where, repeatedly, the operator is executed with the same dataset, redundant data transfers occur, where the same dataset is transferred from the CPU to the GPU and then from the GPU to the CPU in each of these executions. Our proposed solution for the problem of redundant data transfers in Devito is introducing a new data allocator that employs the Unified Memory (UM) interface. The primary objective of this approach is to reduce the frequency of unnecessary data transfers by enabling data persistence within the application. When data is first transferred from the CPU to the GPU, it is retained in GPU memory, even after the operator has been executed. Consequently, subsequent executions of the operator can directly access the data without transferring it again, resulting in improved efficiency and faster processing. This approach significantly reduces the time and resources required for data transfers, ultimately enhancing performance and improving the overall user experience. By leveraging the UM interface, Devito can take advantage of data persistence to maximize efficiency and optimize the use of resources, resulting in faster and more reliable computations. To implement UM, we rely on the CuPy library, developed by Nvidia, which offers UM support and features a structure that closely resembles NumPy, a library extensively used in Devito. This choice helps to minimize compatibility issues and enables us to integrate the UM approach into the Devito framework seamlessly. We conducted an efficiency test of the UM implementation using a seismic dataset (400x400x505) from the SEG/EAGE SALT 3D model in an elastic RTM application. Our experiment with the UM allocator demonstrated significant improvements in execution time, resulting in a gain of 7.5 times compared to the conventional Devito allocators. These findings underscore the effectiveness of the UM approach in optimizing performance and enhancing the overall efficiency of the application.