ParCFD'2020
32nd International Conference on Parallel Computational Fluid Dynamics
May 11-13 2020, Nice FRANCE

EXPLICIT/IMPLICIT NAVIER-STOKES SOLVER ON CPU/GPU HETEROGENENOUS SUPERCOMPUTERS

G. OYARZUN[1], M. ÁVILA[2], R. BORRELL[3], G. HOUZEAUX[4], O. LEHMKUHL[5] AND H. OWEN[6]

[1,2,3,4,5,6] Barcelona Supercomputing Center
Department of Computer Applications in Science and Engineering
Jordi Girona 29, 08034, Barcelona Spain
[1]e-mail: guillermo.oyarzun@bsc.es, web page: https://www.bsc.es/es/oyarzun-guillermo
[2]e-mail: matias.avila@bsc.es, web page: https://www.bsc.es/avila-matias
[3]e-mail: ricard.borrell@bsc.es, web page: https://www.bsc.es/borrell-ricard
[4]e-mail: guillaume.houzeaux@bsc.es, web page: www.bsc.es/houzeaux-guillaume
[5]e-mail: oriol.lehmkuhl@bsc.es, web page: https://www.bsc.es/lehmkuhl-oriol
[6]e-mail: herbert.owen@bsc.es, web page: https://www.bsc.es/owen-herbert

Key words: Navier-Stokes, GPU, vectorization, explicit/implicit solver, LES, co-execution

Summary. Heterogeneous systems are a consolidated trend among the pre-exascale systems. High fidelity computational fluid dynamics simulations are one of the scientific problems with the potential to fully utilize these systems. New implementation models are required to engage the different components within the heterogeneous node. We propose a co-execution model that permit to utilize concurrently the CPUs and GPUs on high fidelity simulations. Using the co-execution, an acceleration of 25% is attained when comparing with a GPU-only execution.

1 INTRODUCTION

High fidelity Computational Fluid Dynamics simulations for turbulent flows entail very large computing requirements, which are progressively acute with each new generation of supercomputers. In order to efficiently exploit the available resources, specific implementations must then be investigated. In this work, we present the implementation of a parallel Navier-Stokes solver for heterogeneous architectures.

The numerical model is based on the low-dissipative scheme presented in [1], where a conservative discretisation of the convective term is considered. As far as time discretisation is concerned, a Runge-Kutta (RK) scheme is employed for the explicit solver and an implicit–explicit (IMEX) RK method for the explicit/implicit solver. The LES model is the Vreman model [2], and the wall model is implemented at the element level, thus avoiding an extra loop on the boundaries of the computational domain and the risky definition of local basis [3].

On the computational side, all levels of parallelism are exploited. At the supercomputer level, a classical MPI-based parallelization is considered together with dynamic partitioning. At the node level, the solution is based on a co-execution paradigm for the efficient use of heterogeneous CPU/GPU architectures, together with a dynamic load balancing mechanism to correctly account for the heterogeneity of the architecture [4]. The co-execution enables to maximise the use of the available devices by executing some kernels concurrently on both CPUs and GPUs. The assessment of the performance of all the proposed strategies has been carried out on the POWER9 architecture accelerated with NVIDIA Volta V100 GPUs

2 SIMD AND STREAM PROCESSING

Nowadays, all CPUs have integrated vector registers that allows to perform arithmetic operations in a Single Instruction Multiple Data (SIMD) execution model. On the other hand, GPUs are composed of stream multiprocessors (SM) that execute threads concurrently in an execution model that is similar to SIMD. Context switching is used to maintain thousands of threads active and hiding the memory latency. Keeping the threads active is known as occupancy and it is one of the main factors to attain maximum performance in GPU computing. For this purpose, the kernel workload has to be divided in threads with low register and shared memory requirements. Additionally, special attention is needed to the creation of new data structures. The SIMD model used by the CPUs and the stream multiprocessors needs of coalesced and aligned memory accesses for optimal performance. Random memory accesses serialize the execution of the memory calls and decrease the overall performance. A data structure that benefits CPU and GPU execution has been developed. First, are numbering is introduced with the objective of grouping the elements of the same geometrical type. Within each subgroup, the elements are packed in sets of VECTOR SIZE elements. If the number of elements in the subgroup is not multiple of VECTOR_SIZE, then zeros are padded at the end of the pack to maintain the regularity. Whenever is possible, the operations of elements within a pack are executed in SIMD mode. VECTOR_SIZE depends on the architecture in which the simulation will be executed. In CPUs, its size is related to the vector register width, and in GPUs determines the number of threads within a block.

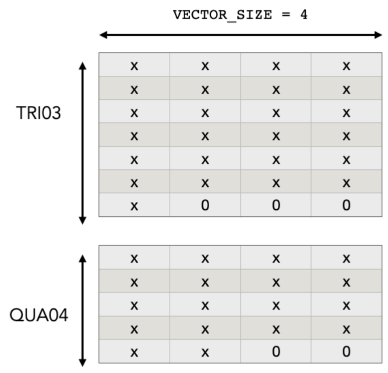

Figure 1 shows the data structure organization for a mesh with 25 elements TRI03, 18 elements QUAD04 and a VECTOR_SIZE=4

3 COMPUTING FACILITIES

Following the current trend in HPC node design, the POWER9 nodes are high throughput heterogeneous nodes. Each compute node has 2xPOWER9 8335 @3.0 GHz with 20 physical cores each. With 4 SMT threads per core, each node can run up to 160 CPU threads. The 512 GB of main memory is distributed in 16 DIMMS of 32 GB each operating at 2666MHz. The 4 Volta V100 accelerators have a 7.8 TFLOPS double precision peak performance giving each one a total of 30.8 TFLOPS. The POWER9 nodes include high throughput NVLINK 2.0 connectors. Each GPU has 6 NVLINK connectors, which are attached to the neighboring GPU as well as CPU, giving an aggregate bandwidth of 150GB/s for GPU to GPU as well as GPU to CPU communication. Details on the node architecture can be found in Figure 2.

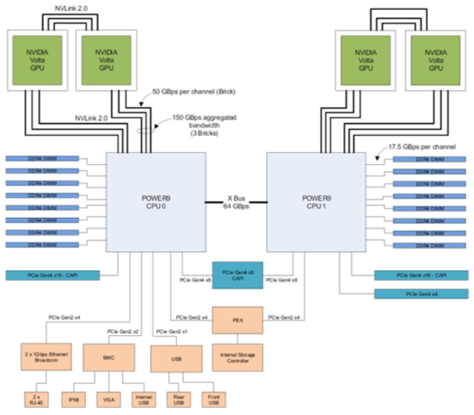

Figure 2 Power 9 cluster configuration.

4 PRELIMINARY RESULTS

The co-execution strategy consists in generating two executables of the CFD code. Each executable is compiled with the optimized parameters for a computing architecture (CPU or GPUs). Then, the execution follows a Multiple Program Multiple Data (MPMD) strategy in which each compiled version resolves a proportional part of the full simulation. A load balancing strategy is utilized for finding the workload on CPU and GPU. The preliminary results using the co-execution show that an airplane simulation is solved 25% faster than using only the four GPUs of the Power9 node. This encourages us to continue studying this execution strategy.

REFERENCES

[1] O. Lehmkuhl, G. Houzeaux, H. Owen, G. Chrysokentis, and I. Rodrìguez. A low-dissipation finite element scheme for scale resolving simulations of turbulent flows. *J. Comp. Phys.*, 390:51-65, 2019.
[2] A. Vreman, An eddy-viscosity subgrid-scale model for turbulent shear flow: Algebraic theory and applications, Physics of Fluids 16:3670-3681. 2004.
[3] H. Owen, G. Chrysokentis, M. Ávila, D. Mira, G. Houzeaux, J.C. Cajas, and O. Lehmkuhl. Wall-modeled

large-eddy simulation in a finite element framework. *Int. J. Numer. Meth. Fluids*, 92:20-37, 2020.

[4] R. Borrell, D. Dosimont, M. Garcia-Gasulla, G. Houzeaux, O. Lehmkuhl, V. Mehta, H. Owen, M. Vázquez, and G. Oyarzun. Airplane Simulation using Heterogeneous CPU/ GPU co-Execution targeting the POWER9 Architecture. *Future Generation Computer Systems,* Volume 107, 2020, Pages 31-48.