ADVANCES IN FOREST FIRE RESEARCH

2022

Edited by
DOMINGOS XAVIER VIEGAS
LUÍS MÁRIO RIBEIRO

A Study of FDS Computational Performance in Heterogeneous Hardware Architectures -Applied for grassland fires

Donghyun Kim[1,2]; Jae-Ryul Shin[*3]; Hwang-Hui Jeong[3]

[1]Jeonju University, Jeonju-si Jeonbuk-do, Rep. of Korea, {72donghyunkim@jj.ac.kr}
[2] IIASA(International Institute for Applied Systems Analysis). Laxenburg, Austria, {dhkim@iiasa.ac.at}
[3]NEXTfoam, Geumcheon, Seoul, Rep. of Korea, {jrshin, hhjeong}@nextfoam.co.kr

**Corresponding author*

Keywords

FDS, Computational Fluid Dynamic, Message Passing Interface, Open Accelerator, Kokkos

Abstract

Fire Dynamic Simulator (FDS), a fire simulation program, applies Message Passing Interface (MPI) and Open Multi Processing (OpenMP) libraries for large-scale simulation. FDS can be executed by dividing simulation problems in a computing cluster using MPI. The main point is to divide the entire domain to be interpreted into several sub-domains and allow each sub-domain to be calculated by an individual computer with an individual processor. When performing parallel computation, FDS first decomposes each sub-domain, then supports two-step parallelization in which multi-threading is applied within each sub-domain, and uses the OpenMP library to implement multi-threading. In this study, OpenACC, a parallelization technique capable of using heterogeneous hardware architectures, was partially applied to FDS. As an application problem, the calculation performance is evaluated through CSIRO Grassland Fires, a verification case of FDS. The hardware for evaluation was a personal computer consisting of dual Xeon 2678-V3 and GeForce GTX 1070. The FDS source code applies OpenACC using PGI Fortran as a compiler in Linux environments. In calculation performance, calculations using CPU and GPU together show 1.89 times faster performance than calculations using a single CPU. In case of using 1 GPU and 16 CPUs (MPP + OpenACC), the analysis result is 21 times faster. In this regard, analysis of grassland fire of WFDS was performed.

1. Introduction

Central Process Unit (CPU), the brain that performs calculations, has been increasing exponentially in unit performance in accordance with Moore's law over the past half century, but the improvement trend has been saturated by the 21st century. Since then, performance has been increased by increasing the number of cores, and now it is trying to overcome the scaling limitations of systems composed only of CPU by accelerators. Therefore, heterogeneous hardware architectures are widely used, and the well-known AlphaGo has 280 Graphic Process Units (GPU) of Tesla accelerators for deep learning. The adaptive parallel technique in heterogeneous hardware architecture proposed in this study is a new technique using the Kokkos library, in which the practical version was released by 2014, and is a very experimental and innovative technology. Therefore, few literatures have been published so far, and its application has recently been introduced by conferences in the field of high-performance supercomputing and international combustion conferences. There is currently the only alternative called Open Computing Language (OpenCL) in a way that does not use Kokkos while pursuing the same goal of adapting to heterogeneous hardware architectures, but this is a new programming language. Compilers that support this must develop together, but there have been no cases applied to the engineering or computational fluid fields yet. On the other hand, because Kokkos uses templates, which are existing features of general-purpose C++ languages, it has the advantage of being able to use dedicated high-performance compilers that are optimized according to architectures such as GNU, PGI, Intel, Cray, and CUDA.

Fire Dynamics Simulator (FDS) is a dedicated fire simulation program written in the Fortran programming language by introducing computational fluid dynamics techniques to analyze the fluid flow phenomenon driven by fire. FDS numerically solves the Navier-Stokes equation for low-velocity heat-driven flows focusing on smoke and heat transfer from fire.Therefore, FDS provides a model that can analyze basic fire dynamics and combustion. FDS has been developed by the National Institute of Standards and Technology in the United States with the goal of interpreting real fire problems. However, the computational cost increases to analyze very

detailed physical phenomena. To overcome this, FDS is written so that the central processing unit (hereinafter referred to as CPU) is executed on various hardware platforms and operating systems sequentially or in parallel. FDS uses a Message Passing Interface (MPI) to divide and execute analysis problems in a computing cluster. This is a method in which the entire area to be analyzed is divided into multiple meshes and each mesh is computed by a separate processor on a separate computer. FDS supports two-step parallelization by first decomposing each computational domain into a mesh of distributed memory and then applying multi-threading within each mesh in performing parallel analysis, and uses the OpenMP (Open Multi Processing) library to implement multi-threading.In this study, Open ACCelerator (OpenACC), a parallelization technique that can use heterogeneous hardware architectures, was partially applied to FDS to speed up FDS operation. The speed verification application case was conducted on Grassland fire tested in Australia in 2010. The computational performance is evaluated through the verification case of FDS, CSIRO Grassland Fires. The hardware for evaluation was configured in parallel with a personal computer consisting of dual Xeon 2678-V3 and GeForce GTX 1070. FDS source code applies OpenACC using PGI Fortran as a compiler in Linux environment.

2. Parallel Computing Method

MPI and OpenMP are standard methods for performing parallel computation in distributed memory system and shared memory system environment. Parallel computation has been supported since FDS version 5.4, and two parallel processing methods are provided, OpenMP and MPI. These two methods can be simultaneously applied to FDS according to the hardware configuration. In section 3.1.2 of the FDS User's Guide, it was reported that the maximum speed improvement of OpenMP is approximately doubled, and that when MPI and OpenMP are used together, most of the reduction in analysis time is achieved by MPI.

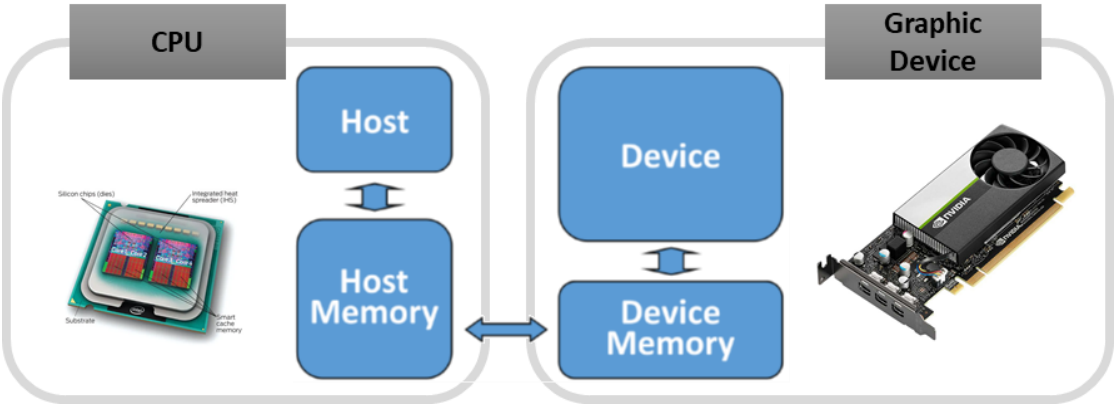

Figure 1- OpenACC's Abstract Accelerator Model

2.1. MPI

MPI is a standard that describes information exchange in distributed and parallel processing. The Message Passing method is an operation model in which data to be exchanged between processors is exchanged using a message passing function. MPI defines the standard of the Message Passing Library, which is a collection of these functions, and several MPI libraries have been developed accordingly. Subroutines that require parallelization in sequence code are constructed in parallel through appropriate parallelization techniques.

2.2. OpenMP

OpenMP is an application programming interface that supports shared memory multi processing of programs in programming languages such as C, C++, and Fortran on various computer platforms. OpenMP provides users with a simple and flexible interface for developing a variety of parallel applications from desktops to supercomputers, and is easily extensible. And by using OpenMP and MPI, the application program can also be built as a Hybrid OpenMP/MPI model.

2.3. OpenACC

OpenACC is a data parallel programming model designed to provide portability between CPU/GPU hardware architectures. Like OpenMP, OpenACC is directive-based, so there are fewer changes to the original code

compared to CUDA or OpenCL. Also, compilers such as PGI and GCC support OpenACC. OpenACC is a method in which the compiler detects directives in a program and determines how to parallelize loops.

3. Computing Performance

The computational performance is evaluated through the verification case of FDS, CSIRO Grassland Fires. As hardware for evaluation, a personal computer composed of dual Xeon 2678-V3 and GeForce GTX 1070 was used. FDS source code applies OpenACC using PGI Fortran as a compiler in Linux environment. In the FDS source code, '!$ACC PARALLEL LOOP' of OpenACC was partially applied to 'PRESSURE_ITERATION_LOOP', the part that determines convergence during calculation.

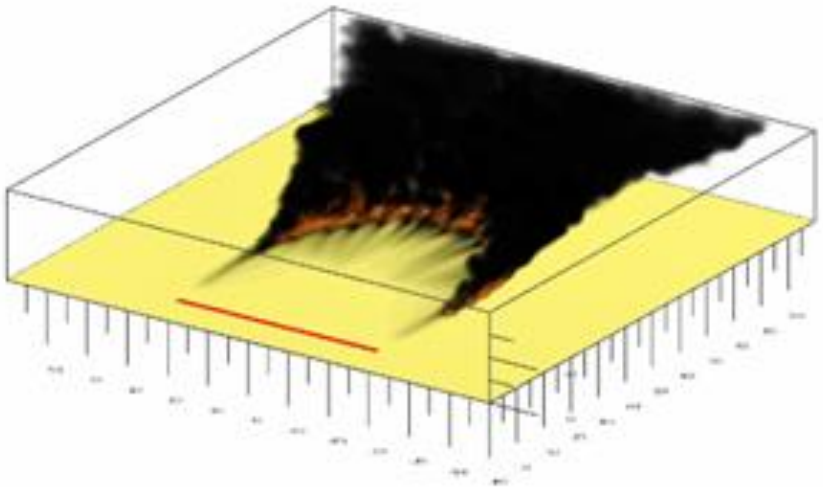

Figure 2- WFDS result of Grassland Fires in CSIRO Austrailia

Table 1- Computing Performance

MPI	Analysis time [hr]	OpenACC with MPI	Analysis time [hr]
1 CPU	109	1 CPU + 1 GPU	57
2 CPU	59	2 CPU + 1 GPU	32
4 CPU	31	4 CPU + 1 GPU	16
8 CPU	17	8 CPU + 1 GPU	9
16 CPU	9	16 CPU + 1 GPU	5

In the case of MPI, when 16 CPUs are used compared to one CPU, the analysis time is 12 times faster from 109 hours to 9 hours. On the other hand, when one CPU and one GPU are used together by applying OpenACC, the calculation time is 1.89 times faster than the result using one CPU. In the case of using 1 GPU and 16 CPUs (MPP + OpenACC), the analysis time is 5 hours, which is about 21 times faster.

Table 2 shows the results of measuring the parallel performance of MPI+OpenACC according to the increase in the size of the computation area. When parallelization was performed with MPI, the performance improvement was evident as the number of CPUs and the size of the problem increased. On the other hand, in the case of MPI+OpenACC, the performance of MPI+OpenACC with 2 CPUs or more is lower than that of MPI. It can be seen that the amount of CPU allocated during MPI parallelization is also allocated to the GPU as much as the number of CPUs, which increases the load and decreases performance. In addition, if the size of the calculation area is about 100 million and 8 CPUs are used, the memory performance of the GPU, GTX 1070, is exceeded, and calculation cannot be performed. However, when one or two CPUs are used, it can be seen that the performance of MPI+OpenACC is significantly superior to that of MPI depending on the size of the calculation area.

Table 2- 2D Heat Conduction SpeedUp with Problem Size and Parallel Method

Problem Size	1024x1024 (@1 million)		2048x2048 (@4 million)		10240x10240 (@100 million)	
Parallel Method	MPI	MPI + OpenACC	MPI	MPI + OpenACC	MPI	MPI + OpenACC
1 CPU	1	1.48	1	2.93	1	4.96
2 CPU	1.96	1.41	1.99	2.82	2.63	4.69
4 CPU	3.70	0.85	4.15	1.70	4.06	2.88
8 CPU	7.21	0.59	6.46	1.08	8.02	Out of GPU memory
16 CPU	15.12	0.31	16.84	0.74	16.71	

4. Conclusion

In this study, OpenACC, a parallelization technique that can use heterogeneous hardware architectures, was partially applied to FDS. As an application problem, computational performance was evaluated through CSIRO Grassland Fires, a verification case of FDS. As hardware for evaluation, a personal computer composed of dual Xeon 2678-V3 and GeForce GTX 1070 was used. In computational performance, the computation using the CPU and GPU is 1.89 times faster than the computation using one CPU. In the case of using 1 GPU and 16 CPUs (MPP + OpenACC), the analysis time is 5 hours, which is about 21 times faster. In the future, it is expected that the hardware parallelization technology can be used to shorten the calculation time for forest fire spread fire analysis using a personal computer.