Parallel Computing 57 (2016) 37–51

Contents lists available at ScienceDirect

Parallel Computing

journal homepage: www.elsevier.com/locate/parco

Hybrid-view programming of nuclear fusion simulation code in the PGAS parallel programming language XcalableMP

Keisuke Tsugane[a,*], Taisuke Boku[a,b], Hitoshi Murai[c], Mitsuhisa Sato[a,c], William Tang[d,e], Bei Wang[e]

[a] *Graduate School of Systems and Information Engineering, University of Tsukuba, 1-1-1 Tennodai, Tsukuba, Ibaraki, 305-8573, Japan*
[b] *Center for Computational Sciences, University of Tsukuba, 1-1-1 Tennodai, Tsukuba, Ibaraki, 305-8577, Japan*
[c] *Advanced Institute for Computational Science, RIKEN, 7-1-26, Minatojima-minami-machi, Chuo-ku, Kobe, Hyogo, 650-0047, Japan*
[d] *Princeton Plasma Physics Laboratory, Princeton University, 100 Stellarator Rd Princeton, NJ 08540, USA*
[e] *Princeton Institute for Computational Science and Engineering, Princeton University, 335 Lewis Science Library Washington Road and Ivy Lane Princeton, NJ 08544, USA*

A R T I C L E I N F O

Article history:
Received 4 November 2015
Revised 27 May 2016
Accepted 31 May 2016
Available online 1 June 2016

Keywords:
GTC-P
Partitioned global address space
Gyrokinetic PIC code
Nuclear fusion simulation
XcalableMP

A B S T R A C T

Recently, the Partitioned Global Address Space (PGAS) parallel programming model has emerged as a usable distributed memory programming model. XcalableMP (XMP) is a PGAS parallel programming language that extends base languages such as C and Fortran with directives in OpenMP-like style. XMP supports a global-view model that allows programmers to define global data and to map them to a set of processors, which execute the distributed global data as a single thread. In XMP, the concept of a coarray is also employed for local-view programming. In this study, we port Gyrokinetic Toroidal Code - Princeton (GTC-P), which is a three-dimensional gyrokinetic PIC code developed at Princeton University to study the microturbulence phenomenon in magnetically confined fusion plasmas, to XMP as an example of hybrid memory model coding with the global-view and local-view programming models. In local-view programming, the coarray notation is simple and intuitive compared with Message Passing Interface (MPI) programming while the performance is comparable to that of the MPI version. Thus, because the global-view programming model is suitable for expressing the data parallelism for a field of grid space data, we implement a hybrid-view version using a global-view programming model to compute the field and a local-view programming model to compute the movement of particles. The performance is degraded by 20% compared with the original MPI version, but the hybrid-view version facilitates more natural data expression for static grid space data (in the global-view model) and dynamic particle data (in the local-view model), and it also increases the readability of the code for higher productivity.

1. Introduction

The Message Passing Interface (MPI) is used widely as a parallel programming model for distributed memory systems. However, its low productivity due to complex local array indexing and error-prone coding during communication is a huge

* Corresponding author. fax.:+81 29 853 6912
E-mail address: tsugane@hpcs.cs.tsukuba.ac.jp (K. Tsugane).

http://dx.doi.org/10.1016/j.parco.2016.05.016
0167-8191/© 2016 Elsevier B.V. All rights reserved.

problem because it forces programmers to describe the data distribution explicitly and they must employ inter-node communication using primitive API functions.

Recently, the Partitioned Global Address Space (PGAS) parallel programming model has emerged as a usable distributed memory programming model. A PGAS language model contains a global address space where any processes can view the data globally in a similar manner to a shared memory system. The global address space is logically partitioned into each process, which can access its own data more efficiently, thereby allowing programmers to perform locality-aware parallel programming.

PGAS parallel programming languages such as UPC [1], Coarray Fortran [2], and X10 [3] have been proposed based on this model. XcalableMP [4–6], or XMP for short, is another PGAS parallel programming language, which is an extension of existing languages such as C and Fortran with directives similar to OpenMP. XMP enables the parallelization of serial source code with additional directives to describe the data distribution, synchronization, and consistency among them over multiple nodes with distributed memory architecture. This approach to extending base languages (C or Fortran) with directives makes it easier for programmers to develop parallel programs on distributed memory systems.

Some PGAS languages support the global-view model that allows programmers to define global data and to map them to a set of processors, which execute the distributed global data as a single thread. In XMP, the global-view model allows programmers to define global arrays, which are distributed to processors by adding the directives. Some typical communication patterns are supported by directives, such as data exchange between neighbor processors in stencil computations.

In contrast to the global-view model, the local-view model describes remote memory access using the node (processor) index. This operation is implemented as one-sided communication. XMP employs the coarray concept from Coarray Fortran as a local-view programming model. A coarray is a distributed data object, which is indexed by the coarray dimension that maps indices to processors. In XMP, the coarray is defined in C as well as Fortran. In the local-view model, a thread on each processor executes its own local computations independently with remote memory access to data located in different processors by coarray access. The local-view model requires that programmers define their algorithms by explicitly decomposing the data structures and controlling the flow in each processor. The data view is similar to that in MPI, but coarray remote access provides a more intuitive view of accessing the data in different processors, thereby increasing productivity.

In this study, we consider a hybrid-view programming approach, which combines the global-view and local-view models in XMP according to the characteristics of the distributed data structure of the target application. The global-view model allows programmers to express regular parallel computations such as domain decomposition with stencil computation in a highly intuitive manner simply by adding directives to a serial version of code. However, it is difficult to describe parallel programs in the global-view model when more irregular communication patterns and complex load balancing are required on the processing. Thus, local-view programming is necessary in these situations.

We utilize this hybrid-view programming for Gyrokinetic Toroidal Code - Princeton (GTC-P) [7], which is a large-scale plasma turbulence code that can be applied at the International Thermonuclear Experimental Reactor (ITER [8]) scale and beyond for next-generation nuclear fusion simulation. The GTC-P is an improved version of the original GTC [9] and it is a type of gyrokinetic Particle-in-Cell (PIC) code with two basic data arrays: global grid data that corresponds to the physical problem space and particle data that corresponds to particles moving around the grid space. The original GTC-P was written in C as a form of hybrid programming with OpenMP and MPI. In this code, the grid data and particle data are mapped onto MPI processes and exchanged. As found with most codes of this type, it is difficult to manage complex data distributions and communication for both grid data and particle data during code development. Furthermore, to simulate the microturbulence phenomenon in plasmas for magnetically confined fusion devices, non-flat domain decomposition is necessary in one dimension, as well as parallelizing multiple dimensions, to obtain accurate large-scale simulations. Therefore, the number of computations becomes extremely large for next-generation and large-scale reactors such as ITER.

We consider both types of data models in XMP, i.e., global-view and local-view models, which are suitable for representing grid space data and particle data, respectively, because of their data distribution and communication pattern. In this study, we implement the GTC-P code in two ways: using XMP with a local-view only model, and with a combination of local-view and global-view models, where we evaluate the performance and productivity of these approaches. As the preliminary result, we implemented and evaluated the GTC-P in XMP hybrid-view model [10]. Moreover, we indicate the causes of performance degradation for GTC-P in XMP and evaluate the GTC-P of hybrid versions written in XMP+OpenMP and MPI+OpenMP in this study.

The remainder of this paper is organized as follows. We provide the related research and an overview of XMP with an example of code in Section 2 and Section 3, and we briefly describe the GTC-P nuclear fusion simulation code in Section 4. Section 5 describes the implementation of GTC-P using XMP. We report the performance and productivity evaluation in Section 6, and we conclude our study in Section 7.

2. Related research

Chapel [11] is widely known as PGAS language having the global-view programming model designed and developed by Cray Inc. As a feature, Chapel employs the original language where programmers can describe several different parallelisms, such as inter/inner node, instruction level threads, and accelerator. In addition, the communication of data based on the global-view model is executed implicitly. While the automatic communication is useful for programmers without experience in parallel programming, it is difficult to tune its performance. In contrast, XMP is an extension of existing languages by