Future Generation Computer Systems 51 (2015) 132–141

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

A high performance framework for modeling and simulation of large-scale complex systems

Feng Zhu [a,b], Yiping Yao [a,b], Wenjie Tang [b], Dan Chen [c,d,*]

[a] State Key Laboratory of High Performance Computing, National University of Defense Technology, Changsha 410073, China
[b] College of Information System and Management, National University of Defense Technology, Changsha 410073, China
[c] National Engineering Research Center for Multimedia Software, School of Computer, Wuhan University, Wuhan 430072, China
[d] Research Institute of Wuhan University in Shenzhen, China

HIGHLIGHTS

- We presented a CCM framework for modeling complex systems.
- We developed a vectorized extension of CCM framework.
- We proposed a two-level composite parallel execution method.

ARTICLE INFO

Article history:
Received 25 June 2014
Received in revised form
17 November 2014
Accepted 27 November 2014
Available online 23 December 2014

Keywords:
Complex systems
Modeling and simulation
Parallel computing
Composite parallel method
Discrete event simulation

ABSTRACT

Due to the quick advances in the scale of problem domain of complex systems under investigation, the complexity of multi-input component models used to construct logical processes (LP) has significantly increased. High-performance computing technologies have therefore been extensively used to enable parallel simulation execution. However, the traditional multi-process parallel method (MPM) executes LPs in parallel on multi-core platforms, which ignores the intrinsic parallel capabilities of multi-input component models. In this study, a vectorized component model (VCM) framework has been proposed. The design aims to better utilize the parallelism of multi-input component models. A two-level composite parallel method (CPM) has then been constructed within the framework, which can sustain complex system simulation applications consisting of multi-input component models. CPM first employs MPM to dispatch LPs onto a multi-core computing platform. It then maps VCMs to the multiple-core platform for parallel execution. Experimental results indicate that (1) the proposed VCM framework can better utilize the parallelism of multi-input component models, and (2) CPM can significantly improve the performance comparing to the traditional MPM. The results also show that CPM can effectively cope with the size and complexity of complex simulation applications with multi-input component models.

1. Introduction

Computer simulation is one of the most important tools for studying problems involved in complex systems. The last decade has witnessed quick advances in the scale of problem domain of complex systems under investigation. The complexity of simulation systems, such as large-scale social simulations [1,2], next-generation network simulations [3], and especially simulations of complex military scenarios [4], has dramatically increased. Subsequently, the requirement for high-performance computing power increases quickly.

The parallel discrete event simulation (PDES) [5] technology adopts an event scheduling strategy, which refers to as-fast-as-possible simulations. Thus it had long been an important approach to sustaining complex system simulation applications. At present, the main trend along this direction adopts the Logical Process (LP) paradigm [6–9]. A physical process, e.g., a detection radar or a battle plane, is modeled by an LP, and interactions between physical processes are modeled by scheduling events between the corresponding LPs [5]. A simulation application includes multiple LPs, and each LP is composed of several independent component models which can be independently developed. Some component models may only involve simple mathematical calculations, while others can model the complicated behavior of computational systems over time [10]. As component models become more and more compute-intensive [11], the demand for computing

* Corresponding author.
E-mail addresses: ypyao@nudt.edu.cn (Y. Yao), dan.chen@ieee.org (D. Chen).

http://dx.doi.org/10.1016/j.future.2014.11.018
0167-739X/

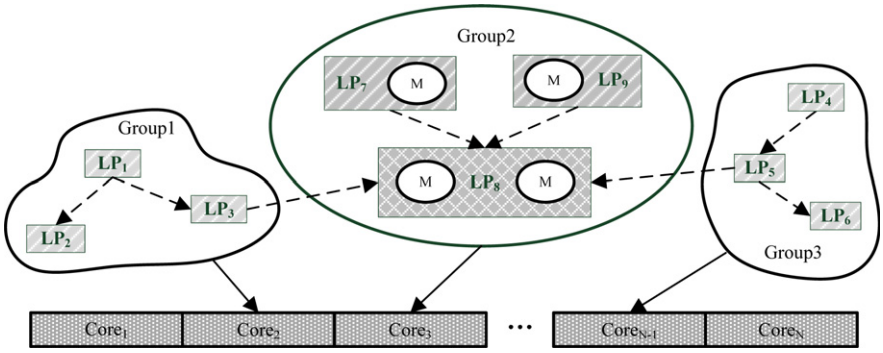

Fig. 1. LPs run on a multi-core computing platform using MPM.

capacity increases quickly, especially for the LPs that include multi-input component models. A typical example is the multi-target radar detection model in decision-making simulation systems for military purpose [12]. These simulations do not include humans or physical devices, and users need to complete the execution of the simulation as quickly as possible. As multi-target detection usually needs to iterate multiple times [13], the LPs which include such multi-target detection models would consume much time in processing events, then the overall simulation will be slowed down by processing such multi-input component models. Recently, studies of these complex simulation systems including multi-input component models (MICS) still suffer from a lack of (1) an appropriate modeling approach to better utilizing the parallelism of multi-input component models and (2) an effective parallel approach to sustaining MICS.

Cloud computing [14,15], grid computing [16,17], datacenter computing [18,19], cluster computing [20–23] and multi-core computing platforms may achieve high performance for complex scientific and engineering applications [24]. However, according to *Amdahl's Law* [25], the speedup is limited by the sequential proportion of a program. Furthermore, the multi-process parallel method (MPM) is only suitable for executing coarse-grained LPs due to the higher overhead of communication and synchronization compared to multi-threads [26,27]. The intrinsic parallel capabilities of multi-input component models are ignored. Meanwhile, the performance will decrease using MPM to dispatch LPs onto multi-core computing platforms when the number of LP processes is greater than the number of cores [28] (shown in Fig. 2). Therefore, for MICS, a new parallel method is needed to use multi-core computing resources to achieve high performance.

In this study, we propose a computational component model (CCM) framework to develop multi-input component models. A vectorized extension of this framework is introduced to take advantage of the parallelism of these multi-input component models. Multiple inputs of a component model are divided into several groups, and each element of the VCM contains a copy of CCM and an input group. Finally, we propose a two-level composite parallel execution method (CPM) and develop the VCM acceleration algorithm (VCA). In CPM, the first level uses MPM to schedule LP groups to run on some cores of the multi-core computing platform, and the second level uses VCA to map VCMs onto other cores to gain accelerations.

Experiments with a multi-target radar detection model have been executed on a *Linux Cluster* with 6 cores. The complexity of the MICS is higher than that the traditional MPMs can sustain. The results indicate that the approach successfully alleviates the bottleneck of executing multi-input component models.

The organization of the paper is as follows: Section 2 discusses the motivation and related work. Section 3 presents the CCM framework and the execution flow, and it introduces the vectorized extension of the CCM framework. Section 4 discusses the composite parallel method and details the algorithm for accelerating VCM. Section 5 describes a case study of a radar detection model. Section 6 presents the performance evaluation. Finally, Section 7 concludes the paper with a summary and proposes a plan for the future work.

2. Motivation and related work

In this section, we discuss the motivation of the VCM framework and the VCM acceleration algorithm. Then we summarize some typical related work along this direction.

2.1. Motivation

The traditional MPM is usually used to dispatch LPs to run on multi-core computing platforms. Fig. 1 shows that LPs run on multi-core platforms using MPM. There are three groups, *Group1*, *Group2* and *Group3*. Some LPs only do simple calculations, such as LP_1, LP_2 and LP_3 in *Group1*, while others contain complex multi-input component models, such as LP_7, LP_8 and LP_9 in *Group2*. LP_8 is composed of two component models, while LP_7 and LP_9 contain one model respectively. Dotted lines represent event scheduling between two LPs, while solid lines represent the distribution from an LP group to the corresponding core.

We use the area of a rectangle to represent the computational complexity of an LP. Fig. 1 shows that the computing loads of LP_7, LP_8 and LP_9, which include multi-input component models, are much heavier than other LPs in *Group1* and *Group3*. Thus, their execution will be the bottleneck of the whole application. From Fig. 1, we can also see that the two component models in LP_8 run in a sequential manner, which may not be efficient when the number of LPs rises to a certain size. Thus, we can draw the conclusion that MPM ignores the independence between different inputs of multi-input component models, which causes difficulties in exploiting fine-grained parallelism for multi-input component models. Hence, it is necessary to find a component model framework to decompose the multi-input component model into several subprocesses, and then map them onto multi-core computing platforms for parallel execution.

We have executed the multi-target radar detection model within a military simulation application on a multi-core computing platform, which is a single computing node of a *Linux Cluster* with 6 cores. The blue curve in Fig. 2 shows that the execution time of the radar model detecting one target increases quickly when the number of LP groups increases from 6 to 7. Meanwhile, the green curve in Fig. 2 shows that the execution time of the whole simulation application increases when the number of LP groups increases from 6 to 7. Hence, one important conclusion can be drawn that the performance will decrease when the number of LP groups is larger than the number of cores. To improve the performance, it is necessary to use a lightweight dispatch approach, which will make full use of multi-core computing platforms to accelerate the execution of the LP which includes multi-input component models.