Computer Physics Communications 194 (2015) 64–75

Contents lists available at ScienceDirect

Computer Physics Communications

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

Parallel implementation of inverse adding-doubling and Monte Carlo multi-layered programs for high performance computing systems with shared and distributed memory☆

Svyatoslav Chugunov, Changying Li *

Bio-Sensing and Instrumentation Laboratory, College of Engineering, University of Georgia, 200 D.W. Brooks Dr., Athens, GA 30602, USA

ARTICLE INFO

Article history:
Received 21 November 2014
Received in revised form
4 February 2015
Accepted 21 February 2015
Available online 1 April 2015

Keywords:
Parallel computing
Monte Carlo
Simulation
Inverse Adding-Doubling
Tissue
Photon

ABSTRACT

Parallel implementation of two numerical tools popular in optical studies of biological materials – Inverse Adding-Doubling (IAD) program and Monte Carlo Multi-Layered (MCML) program – was developed and tested in this study. The implementation was based on Message Passing Interface (MPI) and standard C-language. Parallel versions of IAD and MCML programs were compared to their sequential counterparts in validation and performance tests. Additionally, the portability of the programs was tested using a local high performance computing (HPC) cluster, Penguin-On-Demand HPC cluster, and Amazon EC2 cluster. Parallel IAD was tested with up to 150 parallel cores using 1223 input datasets. It demonstrated linear scalability and the speedup was proportional to the number of parallel cores (up to 150x). Parallel MCML was tested with up to 1001 parallel cores using problem sizes of 10^4–10^9 photon packets. It demonstrated classical performance curves featuring communication overhead and performance saturation point. Optimal performance curve was derived for parallel MCML as a function of problem size. Typical speedup achieved for parallel MCML (up to 326x) demonstrated linear increase with problem size. Precision of MCML results was estimated in a series of tests — problem size of 10^6 photon packets was found optimal for calculations of total optical response and 10^8 photon packets for spatially-resolved results. The presented parallel versions of MCML and IAD programs are portable on multiple computing platforms. The parallel programs could significantly speed up the simulation for scientists and be utilized to their full potential in computing systems that are readily available without additional costs.

Program summary

Program title: MCMLMPI, IADMPI

Catalogue identifier: AEWF_v1_0

Program summary URL: http://cpc.cs.qub.ac.uk/summaries/AEWF_v1_0.html

Program obtainable from: CPC Program Library, Queen's University, Belfast, N. Ireland

Licensing provisions: Standard CPC licence, http://cpc.cs.qub.ac.uk/licence/licence.html

No. of lines in distributed program, including test data, etc.: 53638

No. of bytes in distributed program, including test data, etc.: 731168

Distribution format: tar.gz

Programming language: C.

Computer: Up to and including HPC/Cloud CPU-based clusters.

Operating system: Windows, Linux, Unix, MacOS — requires ANSI C-compatible compiler.

Has the code been vectorized or parallelized?: Yes, using MPI directives.

RAM: From megabytes to gigabytes (MCMLMPI), kilobytes to megabytes (IADMPI)

Classification: 2.2, 2.5, 18.

☆ This paper and its associated computer program are available via the Computer Physics Communication homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

* Correspondence to: College of Engineering, 712F Boyd Graduate Studies, University of Georgia, Athens, GA 30602, USA. Tel.: +1 706 542 4696.
E-mail address: cyli@uga.edu (C. Li).

http://dx.doi.org/10.1016/j.cpc.2015.02.029

External routines: dcmt-library (MCMLMPI), cweb-package (IADMPI)

Nature of problem:
Photon transport in multilayered semi-transparent material, estimation of optical properties (IADMPI) and optical response (MCMLMPI) of multilayered material samples.

Solution method:
Massively-parallel Monte-Carlo method (MCMLMPI), Inverse Adding-Doubling method (IADMPI)

Unusual features:
Tracking and analysis of photon packets in turbid media (MCMLMPI)

Running time:
Many small problems can be solved within seconds, large problems might take hours even on HPC clusters (MCMLMPI); hours on single computer and seconds on HPC cluster (IADMPI)

1. Introduction

Study of light transport in turbid media is a significant topic in bio-optical research. The typical subjects of interest in such studies are optical responses (reflectance, transmittance, and absorbance) and optical properties (coefficients of absorption and scattering as well as scattering anisotropy factor) of biological tissues. While basic optical responses for an examined tissue is acquired experimentally, data processing and modeling of complex cases require special numerical tools. Inverse Adding Doubling (IAD) method [1] and Monte Carlo Multi Layered (MCML) simulation program [2] are often employed at this stage as prospective tools for numerical study of optical parameters of biological materials. In the context of tissue examination, IAD handles conversion of optical responses into optical properties and MCML uses optical properties to simulate light transport in complex multi-layered biological tissues. Principles and numerical implementations of sequential version of IAD are well explained in the work of Scott Prahl [1,3] and details on sequential implementation of MCML are given by Lihong Wang [4] and Steven Jacques [5]. In the past years, sequential versions of these tools were extensively tested by the scientific community [1,6–11] and were accepted as gold standard for study of light propagation in turbid media. At the same time, necessity of more productive parallel versions of these tools was identified for processing of plentiful optical data generated by the spectrometric imaging technique and for comprehensive study of complex multilayered tissues under different simulated scenarios. Numerical processing of spectrometric data with IAD and subsequent simulations with MCML are time consuming tasks: typical computation time for IAD processing is within hours; for MCML this time is in the order of tens of hours. Study of different case-scenarios using MCML increases processing time to days. Parallel implementation of IAD and MCML attempted in this study is intended to significantly accelerate the computation process and acquire simulation results within short period of time.

Prior to the attempts of parallel conversion we considered available parallel versions of IAD and suitable Monte Carlo (MC) programs. To date there was no parallel version of IAD available. However, there existed a few parallel MC-based programs simulating photon transport [2,12–15] in different types of materials. Versions of parallel MC programs for photon migration were previously proposed by a variety of authors [16–21]. For example, Page et al. [16] developed a Java-based application with in-house-made MC-code and tested it on a distributed network of general purpose personal computers. Colasanti et al. [17] converted MC-code designed earlier by their group into a parallel version using F90 version of Fortran-language. They used "SHared MEMory access library" that brings communication time in distributed systems close to that of shared memory systems, using low-level features of hardware developed by Silicon Graphics Inc. Lo et al. [18] developed hardware-based simulations of photon propagation in 5-layered materials using field-programmable gate arrays (FPGA) electrically driven in accordance with principles implemented in MCML. Badal et al. [19] proposed a set of Linux scripts that distribute MC-code (or any other sequential code) over a network of computers, execute the code via SSH commands, and gather results after computations are finished. Wang et al. [20] implemented MPI-based parallel scheduler in EGS5 MC-code [14] and tested it on Amazon Elastic Cloud Compute (EC2) cluster. Pratx et al. [21] evaluated MC321 MC-code [15] using Hadoop framework [22] deployed over Amazon EC2 cluster. They adopted MapReduce [23] programming model to provide parallel capabilities to their program. Among these programs, only MCML provided accurate, well-tested, and simple approach to simulate samples that have geometry of a parallel slab—such geometry is specific to biological materials. Based on the current literature, there existed two parallel implementations of MCML: the first one was developed by Alerstam et al. [24] with parallel optimization focused on General Purpose Graphic Processing Units (GPGPU) [24]; the second one was developed by Lo et al. [18] and featured hardware-based implementation. The GPGPU version [24] was closely bound to NVidia chipsets due to substantial low-level optimization introduced into the code and the use of NVidia's genuine Compute Unified Device Architecture (CUDA). While demonstrating significant speedup (up to 621× at 480 CUDA cores), it was designed to work only with NVidia hardware and did not allow easy modification of the code as the addition of new features for scientific applications compromised the low-level optimization. The hardware version of MCML [18] was able to simulate light absorption in 5-layered model of human skin featuring 80× acceleration in comparison to sequential version of MCML executed on a 3 GHz Intel Xeon processor. This version was focused on simulation of only light absorption in tissues, dropping light reflectance and transmittance in favor of the overall performance of the system. According to Lo et al., it was subject to memory constraints, and required 1 person–year development time using the known MCML algorithm to produce simulations for 5 layers of tissue. The listed MC-programs for simulation of light propagation in multi-layered biological tissues have limited applicability and GPGPU-optimized and FPGA-optimized MCML implementations have hardware restrictions. These limitations facilitated the authors' decision to proceed with parallel conversion of MCML which was based on x86 instruction set, standard C-language, and Message Passing Interface (MPI). Compliance with these standards made the parallel MCML suitable for execution at most High Performance Computing (HPC) clusters, as well as on regular desktop computers with multi-core CPUs. In addition, it