Contents lists available at ScienceDirect

The Journal of Systems and Software

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

# Load-prediction scheduling algorithm for computer simulation of electrocardiogram in hybrid environments

Wenfeng Shen[a,*], Zhaokai Luo[a], Daming Wei[a,b], Weimin Xu[a], Xin Zhu[c]

[a] School of Computer Engineering and Science, Shanghai University, Computer Building, 333 Nanchen Road, Shanghai, 200444, China
[b] Graduate School of Medicine, Tohoku University, Seiryo-machi, Aoba-ku, Sendai, Miyagi 980-8575, Japan
[c] Biomedical Information Technology Lab, The University of Aizu, Aizu-Wakamatsu, Fukushima 965-8580, Japan

ARTICLE INFO

*Article history:*
Received 8 March 2014
Revised 4 December 2014
Accepted 7 January 2015
Available online 15 January 2015

*Keywords:*
Computer simulation of ECG
Load-Prediction Scheduling
Sliding Window Mechanism

ABSTRACT

This paper proposes an algorithm that allows fully utilize the Central Processing Unit–Graphics Processing Unit (CPU–GPU) hybrid architecture to conduct parallel computation and reasonable scheduling for computer simulation of electrocardiogram (ECG). This algorithm is realized by accelerating calculation speed and increasing platform adaptability of the parallel algorithm.

Today, many algorithms have been proposed to dynamically schedule a set of tasks in CPU–GPU hybrid environments. Among these scheduling algorithms, only Pure Self-Scheduling (PSS) algorithm can achieve load balancing in such an extremely heterogeneous environment. However, Pure Self-Scheduling can neither fully exploit the advantages of GPU performance, nor efficiently minimize the dynamic scheduling overhead. In this paper, Load-Prediction Scheduling (LPS) has been introduced to solve the aforementioned problems. Furthermore, to meet the demand for the best performance in a hybrid environment, which is formed by many heterogeneous computers, we propose an approach to adjust scheduling parameters dynamically. In order to validate our parallel algorithm and scheduling approach, we performed ECG simulation to confirm the efficiency and accuracy of ECG simulation algorithms based on the proposed method. At first, LPS predicts the workloads of each step in the simulation. The prediction results help to schedule heavy workloads to components with strong computational ability and light workloads to components with weak computational ability. LPS also synthesizes dynamic-scheduling and static-scheduling methods to minimize the disadvantages of these two scheduling methods. In the meantime, a Sliding Window Mechanism (SWM) adjusts the boundary between dynamic-scheduling and static-scheduling to make LPS perform better in hybrid environments. Experimental results of LPS on the computer simulation of ECG show that the LPS algorithm is more efficient than PSS. The ECG simulation is improved by about 20 times by using our proposed method. The ECG simulation of LPS with SWM is about 21% faster than that without SWM.

## 1. Introduction

With rapid development of computer technologies, especially multi-core technology, the computational capacity of the CPU is increasing dramatically. Furthermore, the emergence of Graphics Processing Unit (GPU) upgrades computing node's computing ability. Many researchers have developed methods to accelerate computing speed by the means of a GPU. For example, Sato et al. employed the GPU to accelerate the simulations of electrical wave propagation in myocardium, and achieved a computational speed that was 30 times faster than that achieved with a single 2.0 GHz AMD Opteron processor for 2D tissue simulations (Sato et al., 2009). Han et al. presented a high-performance software router framework for general packet processing with GPU acceleration (Han et al., 2010). In the meantime, many researchers are also developing methods to fully harness the computational power of CPU and GPU simultaneously. Feichtinger et al. have addressed the issue wherein sustaining a large fraction of single GPU performance in parallel computations is considered a major problem of GPU-based clusters in the context of a lattice Boltzmann flow solver (Feichtinger et al., 2011). Their multi-GPU implementation uses a block-structured MPI parallelization and it is suitable for load balancing and heterogeneous computations on CPUs and GPUs (Feichtinger et al., 2011).

Recently, to fully combine the computing power of various accelerating components, high performance computing clusters have increasingly embraced GPU–CPU heterogeneous architectures. To exploit the computing power of both CPU and GPU, many research teams have studied accelerating computing speed with CPU and GPU.

* Corresponding author. Tel.: +86 021 66135375; fax: +86 021 66135517.
*E-mail address:* wfshen@mail.shu.edu.cn (W. Shen).

http://dx.doi.org/10.1016/j.jss.2015.01.015
0164-1212/© 2015 Elsevier Inc. All rights reserved.

Dziekonski et al. realized numerical results for a setup consisting of a Fermi GPU (GTX 480) and a Xeon six-core CPU showed that the proposed approach allows one to handle systems involving millions of unknowns and reach a speedup factor of almost four compared to the CPU-only implementation (Dziekonski, 2011). Kerr et al. reported an empirical evaluation of 25 CUDA applications on four GPUs and three CPUs leveraging the Ocelot dynamic compiler infrastructure, which can execute the same CUDA applications on either target (Kerr et al., 2010). Although the studies mentioned above have shown that the CPU–GPU heterogeneous combination can provide powerful computing ability, there are still few solutions that can be used to present a viable approach to fully utilize the CPU–GPU collaborative computing ability.

Our team has also conducted many studies on parallelization. Shen et al. presented a parallel algorithm using a GPU for computer simulation of electrocardiogram (ECG) based on a 3-dimensional (3D) whole-heart model (Shen et al., 2009). This study is the basis of our work. Thereafter, considering that the GPU is unsuited for branch structure, Shen et al. proposed a GPU-based algorithm that concentrates on eliminating branches in computation and optimizes the calculation of electric potentials by means of load-prediction (Shen et al., 2012). To fully utilize all accelerating components of a computer, Shen et al. employed parallel computation for computer simulation of ECGs on a CPU–GPU cluster using a hybrid parallel algorithm with parallel program development tools—MPI, OpenMP, and CUDA (Shen et al., 2013). That is the basis of the paper about the original concept of Load-Prediction Scheduling (LPS). This study is focused on accelerating the ECGs simulation using multiple computing node and multiple accelerating components without considering how to adjust the workloads among multiple devices.

In this paper, we propose an approach to address the issue mentioned above. To take full advantage of the computing power of the CPU–GPU hybrid structure node, it is important to tackle the process of assigning tasks to achieve the balance of computer loads. In other words, a method needs to be determined to assign different parts of a parallel application to the computing components, in order to minimize the overall running time (Xu et al., 2013). We employ the LPS to parallelize serial calculation with CPU and GPU. Thus, both CPU and GPU can share the workload reasonably. Because the CPU and GPU's computing abilities are different, the approach described in this paper has the ability to adjust its workload scheduling to fully exploit the computing ability of the CPU and GPU in hybrid environments. We also embrace Sliding Window Mechanism (SWM) for scheduling workloads according to the computing ability of the CPU and GPU. In order to validate our approach, we perform the parallel computation of electrocardiogram simulation based on the proposed method. For comparison, we also conduct three additional experiments: serial algorithm, parallel algorithm without SWM, and parallel algorithm with SWM, and demonstrate that our approach's performance is the best.

The rest of this paper is divided into four parts. We describe our approach's details including LPS and SWM in Section 2. In Section 3, ECG simulation will be adopted to discuss the effectiveness of our approach. Section 4 exhibits the experiments of parallelizing ECG simulation with our LPS and SWM approach.

## 2. LPS and SWM

Before conducting parallel computing, the important point to be considered first is data dependency. Data dependency means that each step of the serial algorithm needs the previous step's result as input parameter. Such algorithm cannot fully utilize all the calculating components.

There are many methods on solving data dependency between steps. In this paper, in order to address the data dependency issue, the instance of ECG simulation that will be described in detail in Section 3.

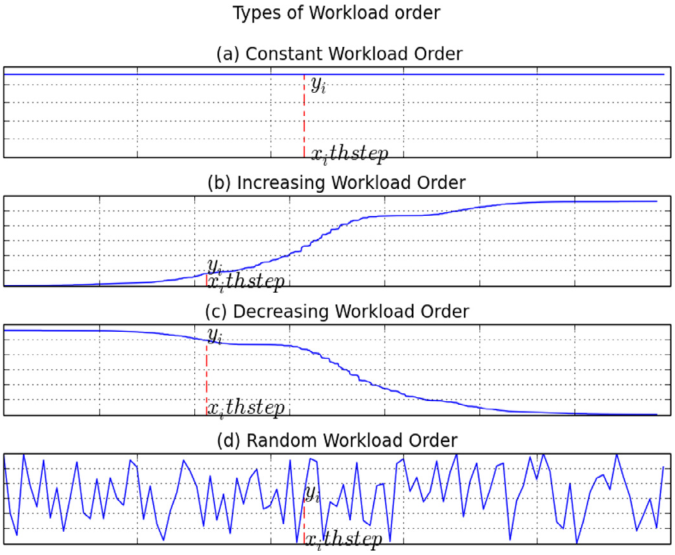

**Fig. 1.** Types of workload order.

On the other hand, if all steps in the workload order are isolated, the issue is how to allocate the workload of each step to different computing components. Workloads are one of the important sources of parallelism in scientific computing programs and therefore a lot of research was focused in this area (Wu et al., 2012). A step's workload is called a parallelizable workload if there is no data dependency among all steps, i.e., workloads can be processed in any order or even simultaneously (Han and Chronopoulos, 2013). The order of workload can be roughly divided into four kinds as shown in Fig. 1: constant workload order in Fig. 1(a), increasing workload order in Fig. 1(b), decreasing workload order in Fig. 1(c), and random workload order in Fig. 1(d). In Fig. 1, the *y*-axis is the amount of workload. The *x*-axis represents the step corresponding to the workload, for example, in Fig. 1(a), the workload amount of $x_i$th step is $y_i$.

For achieving a good performance, this paper adopts LPS. LPS contains two stages: pretreatment and scheduling stages. The details are described as follows.

At first, in order to fully utilize the computing components in the node, we would like to allocate heavy workloads to the accelerating component with strong computing ability. Likewise, we would like to allocate the light workloads to the accelerating component with weak computing ability. For simplicity, we quote the research conducted previously to make a hypothesis stating that the CPU's computing ability is weaker than that of the GPU (Shen et al., 2009, 2012, 2013). Therefore, our approach would allocate the heavy workloads to GPU and assign the light workloads to CPU.

Therefore, a pretreatment stage is needed before employing workload scheduling. In view of the different workload order in Fig. 1, constant workload order, increasing workload order, and decreasing workload order make it easy to estimate the workloads of certain parts. However, constant workload order is infrequent, and the most common situation is the random workload order. Therefore, steps in the workload order are isolated after addressing the data dependency issue. Thus, the workload order can be sorted, and the experiments' result generated by the sorted workload order is correct. In our approach, pretreatment refers to sorting of the workload in descending order after addressing the data dependency problem.

After pretreating the workload order, scheduling is the next core problem of our approach. Before introducing our scheduling approach, we will discuss the existing scheduling strategies. Generally, two types of scheduling strategies exist: static scheduling and