Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 327 (2016) 93–107

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Model-based Dynamic Control of Speculative Forays in Parallel Computation [4]

Kalyan S. Perumalla[1] Mohammed M. Olama[2]
Srikanth B. Yoginath[3]

Computational Sciences and Engineering
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA

Abstract

In simulations running in parallel, the processors would have to synchronize with other processors to maintain correct global order of computations. This can be done either by blocking computation until correct order is guaranteed, or by *speculatively* proceeding with the best guess (based on local information) and later correcting errors if/as necessary. Since the gainful lengths of speculative forays depend on the dynamics of the application software and hardware at runtime, an online control system is necessary to dynamically choose and/or switch between the blocking and speculative strategies. In this paper, we formulate the reversible speculative computing in large-scale parallel computing as a dynamic linear feedback control (optimization) system model and evaluate its performance in terms of time and cost savings as compared to the traditional (forward) computing. We illustrate with an exact analogy in the form of vehicular travel under dynamic, delayed route information. The objective is to assist in making the optimal decision on what computational approach is to be chosen, by predicting the amount of time and cost savings (or losing) under different environments represented by different parameters and probability distribution functions. We consider the cases of Gaussian, exponential and log-normal distribution functions. The control system is intended for incorporating into speculative parallel applications such as optimistic parallel discrete event simulations to decide at runtime when and to what extent speculative execution can be performed gainfully.

Keywords: Reversible execution, Parallel Computing, Speculative Execution, Model-based Execution

[1] Email: perumallaks@ornl.gov

[2] Email: olamahussemm@ornl.gov

[3] Email: yoginathsb@ornl.gov

[4] This paper has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the U.S. Dept. of Energy. Accordingly, the U.S. Government retains and the publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for U.S. Government purposes.

http://dx.doi.org/10.1016/j.entcs.2016.09.025

1571-0661/© 2016 The Author(s). Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

1 Introduction

1.1 Speculative Reversible Parallel Computation

Reversible computing is a relaxation of conventional forward-only computing [1]. In reversible computing, execution is designed to make it possible to go forward as well as backward: the application running on any processor is designed to change the direction of execution on demand. Such a reversible execution framework is useful in many contexts, such as database transactions and parallel discrete event simulations. In these applications, inter-processor synchronization forms a major cost which arises from the need of each processor to get information often from other processors on the next local trajectory to follow. One way to avoid blocking and wasting time waiting for information from other processors is to proceed with a best estimate based on local information, and then rely on reversible computation to retrace any incorrect portions of the trajectory. These portions of local execution without waiting for perfect global knowledge are called speculative forays. The higher the fraction of correctness of the speculative foray, the greater the gain; the lower the fraction, the lower the gain (or, worse, the more negative the gain). It is very hard to ascertain the best strategy ahead of time; in fact, it is preferable to have a runtime controller that can dynamically make decisions on when and to what extent the speculative foray should be allowed. In other words, an online control system is needed so that the speculative forays are dynamically controlled.

Here, we investigate an approach based on a model-based control system design that is agnostic to the specific application. We explore different stochastic distributions for the main runtime dynamics, and analyze their efficacies in synthetic experiments. The overall problem is defined in terms of two competing objectives: time to completion and total cost for completion. We explore the space with representative (normalized) parameter values in order to gain an initial understanding of the efficacy of our approach. Our work presented here differs from previous analyses in the literature on rollback-based parallel computation. Previous works focused on analyzing the applications and application classes as a whole in determining apriori the average, best and worst cases of blocking and speculative strategies [5,6,7,8,9]. In contrast, we focus on the design problem of online control to dynamically choose (and potentially switch) between blocked and speculative execution, and also attempt to maintain the lengths of speculative forays as an option of runtime control.

1.2 Analogy

Consider a setting in which a driver is driving a vehicle to some final destination **F** to which the path is incrementally obtained. Suppose the driver is currently at some milestone point **A**. Further suppose that, after reaching **A** some processing time $R(t)$ is needed to determine the route to the next milestone point **B**. So, the vehicle is stationary while the driver waits for the next milestone **B** to be determined. To save time, the driver may guess the next route and start driving. By the time the next milestone **B** is determined, the vehicle may have already gone ahead from **A** to another milestone **C**. It has to now drive back from **C** to the point **D** at which the

Download English Version:

https://daneshyari.com/en/article/4950096

Download Persian Version:

https://daneshyari.com/article/4950096

Daneshyari.com