Available online at www.sciencedirect.com
SCIENCE DIRECT®
Electronic Notes in Theoretical Computer Science
ELSEVIER Electronic Notes in Theoretical Computer Science 128 (2005) 19–34
www.elsevier.com/locate/entcs

Load Balancing Parallel Explicit State Model Checking

Rahul Kumar [1] and Eric G. Mercer [2]

Verification and Validation Laboratory
Department of Computer Science
Brigham Young University
Provo, USA

Abstract

This paper first identifies some of the key concerns about the techniques and algorithms developed for parallel model checking; specifically, the inherent problem with load balancing and large queue sizes resultant in a static partition algorithm. This paper then presents a load balancing algorithm to improve the run time performance in distributed model checking, reduce maximum queue size, and reduce the number of states expanded before error discovery. The load balancing algorithm is based on generalized dimension exchange (GDE). This paper presents an empirical analysis of the GDE based load balancing algorithm on three different supercomputing architectures—distributed memory clusters, Networks of Workstations (NOW) and shared memory machines. The analysis shows increased speedup, lower maximum queue sizes and fewer total states explored before error discovery on each of the architectures. Finally, this paper presents a study of the communication overhead incurred by using the load balancing algorithm, which although significant, does not offset performance gains.

Keywords: model checking, parallel, distributed, load balancing, GDE, queue size, error detection, communication

1 Introduction

Explicit state model checking is a methodology to verify properties in a design through reachability analysis. The practical application of model checking, however, is hindered by the state explosion problem [5]. State explosion is a result of enumerating the state space of a concurrent system using interleaving

[1] Email: rahul@cs.byu.edu
[2] Email: egm@cs.byu.edu

doi:10.1016/j.entcs.2004.10.016

semantics where each concurrently enabled transition must be considered separately in any given state. Several techniques exist to address aspects of the state explosion problem. Symmetry and partial order reduction exploit structure and concurrency to reduce the number of states in the reachable state space that must be explored to complete the model checking problem [3][6]. Bit state hashing (supertrace) and hash compaction reduce the cost of storage states in the reachable state space [7][13]. All of these techniques enable the verification of larger problems, but in the end, are restricted to the number of states that can be stored on a single workstation. If the model checking algorithm exhausts resources on the workstation it is running on before completion of the verification problem, then the problem must be altered in some way to reduce the size of its reachable state space until it can fit into the available resources.

The goal of distributed model checking is to combine the memory and computational resources of several processors to enhance state generation and storing capacity. The seminal work in distributed model checking presented by Stern and Dill creates a static partition of the reachable state space during execution [14]. The workload observed as a function of time and communication overhead on each processor depends critically on how the states are partitioned between the verification processes. Several techniques such as caching (sibling and state), partial order reduction, symmetry reduction, different partition functions and dynamic partitioning have been explored in the past to reduce communication overhead and create perfect state distributions [11][10]. Even with the use of the above mentioned techniques, creating a perfect partition for any given problem while maintaining equally loaded processors requires *a priori* knowledge of the state space, which is the very problem we are trying to solve.

This paper presents an empirical study of the seminal static partition algorithm showing the level of load imbalance, regardless of the chosen static partition, that exists between the processes on different supercomputing platforms. The imbalance results in high idle times in several processors, as well as extremely large search queues. The high idle times indicate that many processors are not contributing to state enumeration, and the large search queues lead to premature termination by exhausting memory resources. Furthermore, the imbalance in the partition slows down error discovery since states leading to errors can be buried deep in the search queues. The paper further presents a load balancing algorithm based on generalized dimensional exchange(GDE) to mitigate idle time at the expense of additional communication overhead. Load balancing the state partition algorithm improves speedup in distributed model checking despite the increased communication. In addition, it reduces