Chapter 16
Analyzing Concurrent Programs Title for Potential Programming Errors

Qichang Chen
University of Wyoming, USA

Liqiang Wang
University of Wyoming, USA

Ping Guo
University of Wyoming, USA

He Huang
University of Wyoming, USA

ABSTRACT

Today, multi-core/multi-processor hardware has become ubiquitous, leading to a fundamental turning point on software development. However, developing concurrent programs is difficult. Concurrency introduces the possibility of errors that do not exist in sequential programs. This chapter introduces the major concurrent programming models including multithreaded programming on shared memory and message passing programming on distributed memory. Then, the state-of-the-art research achievements on detecting concurrency errors such as deadlock, race condition, and atomicity violation are reviewed. Finally, the chapter surveys the widely used tools for testing and debugging concurrent programs.

DOI: 10.4018/978-1-60960-215-4.ch016

INTRODUCTION

The development in the computing chip industry has been roughly following Moore's law in the past four decades. As a result, most classes of applications have enjoyed regular performance gains even without real improvement on the applications themselves, because the CPU manufacturers have reliably enabled ever-faster computer systems. However, the chip industry is now facing a number of engineering challenges associated with power consumption, power dissipation, slower clock-frequency growth, processor-memory performance gap, etc. Instead of driving clock speeds and straight-line instruction throughput ever higher, the CPU manufacturers are instead turning to multi-core architectures.

With the prevalence of multi-core hardware on the market, the software community is witnessing a dramatic shift from the traditional sequential computing paradigm to the parallel computing world. Parallel computing exploits the inherent data and task parallelism and utilizes multiple working processes or threads at the same time to improve the overall performance and speed up many scientific discoveries. Although threads have certain similarities to processes, they have fundamental differences. In particular, processes are fully isolated from each other; threads share heap memory and files with other threads running in the same process. The major benefits of multithreading include faster inter-thread communication and more economical creation and context switch.

Here, we use "concurrent" and "parallel" interchangeably, although there is a little difference between them. Usually, "parallel programming" refers to a set of tasks working at the same time physically, whereas "concurrent programming" has a broader meaning, *i.e.*, the tasks can work at the same time physically or logically.

Although for the past decade we have witnessed increasingly more concurrent programs, most applications today are still single-threaded and can no longer benefit from the hardware improvement without significant redesign. In order for software applications to benefit from the continued exponential throughput advances in new processors, the applications will need to be well-written concurrent software programs.

However, developing concurrent programs is difficult. Concurrency introduces many new errors that are not present in traditional sequential programs. Recent events range from failing robots on Mars to the year 2003 blackout in northeastern United States, which were both caused by a kind of concurrency error called race condition. Debugging concurrent programs is also difficult. Concurrent programs may behave differently from one run to another because parallelism cannot be well determined and predicted beforehand. Existing debugging techniques that are well adopted for sequential programs are inadequate for concurrent programs. Specialized techniques are needed to ensure that concurrent programs do not have concurrency-related errors. Detecting concurrency errors effectively and efficiently has become a research focus of software engineering in recent years.

In the rest of the chapter, we review the state-of-the-art research achievements on detecting concurrency errors as well as the corresponding parallel programming models. Major debugging tools are also introduced and compared with regard to their usability and capability.

PARALLEL COMPUTING PLATFORMS

Advances on Architecture: Multi-Core Processor

Due to the physical limitations of the technology, keeping up with Moore's Law by increasing the number of transistors on the limited chip area has been becoming a more difficult challenge for the CPU industry. In the past decade, we have