Chapter 13
Programming Paradigms in High Performance Computing

Venkat N Gudivada
Marshall University, USA

Jagadeesh Nandigam
Grand Valley State University, USA

Jordan Paris
Marshall University, USA

ABSTRACT

Availability of multiprocessor and multi-core chips and GPU accelerators at commodity prices is making personal supercomputers a reality. High performance programming models help apply this computational power to analyze and visualize massive datasets. Problems which required multi-million dollar supercomputers until recently can now be solved using personal supercomputers. However, specialized programming techniques are needed to harness the power of supercomputers. This chapter provides an overview of approaches to programming High Performance Computers (HPC). The programming paradigms illustrated include OpenMP, OpenACC, CUDA, OpenCL, shared-memory based concurrent programming model of Haskell, MPI, MapReduce, and message-based distributed computing model of Erlang. The goal is to provide enough detail on various paradigms to help the reader understand the fundamental differences and similarities among the paradigms. Example programs are chosen to illustrate the salient concepts that define these paradigms. The chapter concludes by providing research directions and future trends in programming high performance computers.

INTRODUCTION

Availability of multiprocessor and multi-core chips and GPU accelerators is making desktop supercomputers a reality (Ajima, Sumimoto, and Shimizu, 2009; Donofrio et al., 2009; Hoisie and Getov, 2009; Keckler and Reinhardt, 2012; Sodan et al., 2010; Torrellas, 2009; Tumeo, Secchi, and Villa, 2012; Wilde et al., 2009). Manufacturers of such commodity chips include Nvidia, Intel, AMD, and IBM. For example, NVidia markets Tesla GPU accelerators that can

DOI: 10.4018/978-1-4666-7461-5.ch013

be used to turn an ordinary desktop computer into a personal supercomputer. The Tesla K20 GPU accelerator delivers 1.17 Tflops double-precision and 3.52 Tflops single-precision floating point performance. These impressive advances in processor speeds provide unprecedented computational power to solve problems such as visualizing molecules, analyzing air traffic flow, and identifying hidden plaque in arteries. Furthermore, the need for high performance computing has never been greater for the reasons discussed below.

About 400 years ago Galileo wrote "… the book of nature is written in the language of mathematics." This statement is even more relevant today given the need for analyzing and interpreting massive amounts of data (aka Big Data) generated by the synergistic confluence of pervasive sensing, computing, and networking. This data is heterogeneous and the volumes are unprecedented in scale and complexity. Big Data is the next frontier for innovation, competition, and productivity (Manyika et al., 2011). Big Data presents opportunities as well as challenges. For example, low-cost high throughput technologies in genomics, real-time and very high resolution imaging, and mass spectrometry-based flow cytometry are transforming the way research is conducted in life sciences (Schadt et al., 2010).

Many of the challenges in genomics derive from the informatics needed to store and analyze the large-scale high-dimensional datasets that are being generated so rapidly. A prime example of this is the 1000 Genomes project with a 200 TB dataset (1000 Genomes, 2012; Amazon, 2012). This project aims to build the most detailed map of human genetic variation with the genomes of more than 2,600 people from 26 populations around the world.

In the physical sciences domain, astronomers are collecting more data than ever. Currently 1 petabyte (PB) of this data is electronically accessible to public, and this volume is growing at 0.5 PB per year (Berriman and Groom, 2011; Hanisch, 2011). The STScI (Space Telescope Science Institute) reports that more papers are published with archived datasets than with newly acquired data (STScI, 2012). It is estimated that more than 60 PB of archived data will be accessible to astronomers (Hanisch, 2011).

Special programming techniques are needed to harness the power of supercomputers in solving compute-intensive problems listed above. Primarily there are two paradigms for programming high performance computers: shared-memory and distributed memory models (Pacheco, 2011). In the *shared-memory model*, processors share certain memory locations to exchange data and results between the processors. OpenMP, OpenACC, CUDA, OpenCL, and the concurrent programming model of Haskell fall under this category. In the *distributed memory model*, processors do not share memory but exchange data and results through interprocess messages. MPI, MapReduce, and the concurrent programming model of Erlang fall under this category.

In this chapter, we provide an introduction to the above programming paradigms. Using succinct example programs, we illustrate fundamental features of these paradigms. The primary goal for this chapter is to help the reader gain conceptual understanding of the principles that underlie various programming paradigms. This in turn will help the reader in choosing a relevant paradigm for a given problem.

We do not discuss the installation of software libraries and compilers needed to run the programs illustrated in this chapter because of space constraints. Installation instructions vary from one operating system to another and also from one release to the next. The reader should consult documentation supplied by the providers of software libraries and compilers. It is assumed that the reader is familiar with C/C++ programming.