Introduction to Parallel Programming

Scott L. Hamilton

August 1, 2007

Contents

Chapter 1

An Overview of Parallel and Serial Programming

You may be asking yourself, "Why take a course in parallel Programming?" Since late 1993,when Donald Becker and Thomas Sterling began sketching the outline of a commodity-based cluster system designed as a cost-effective alternative to large supercomputers, Parallel programming has been on the cutting edge of software engineering. The very first parallel computer systems were expensive, custom built super computers. They used a shared memory model, meaning that many processors worked from a bank of shared memory resources. Any changes made to memory by one processor was immediately known by the other processors.

1.1 A Brief History of Parallel Computing

In early 1994, working at CESDIS under the sponsorship of the HPCC/ESS project, the Beowulf Project was started. This project brought the world of parallel programming from the large corporations and government facilities to the hands of small business and educational institutions. This was made possible by the design of a parallel distributed memory system. In Beowulf style clusters, commodity computer hardware is utilized along with a network interconnect to do large scale computation. Each processor has its own memory and the only shared resource is the network and any attached storage. This made it necessary to create a Message Passing Interface (MPI) to share information between processors since they no longer had direct access to the same set of memory.

There are two classes of . Class I Clusters consist of standard PC hardware with no specialized networking or additional acceleration hardware. Class II Clusters use specialized hardware such as Myrinet or Infiniband network interconnects to increase communication speeds, or the addition of FPGAs, physics processors, or high end graphics cards to increase the computational power of the cluster.

There is one other class of Parallel Computer, the , which is much like the Beowulf cluster, but does not have the shared file system resources, or the high speed interconnections that a Beowulf cluster has. The most popular Grid computer application is the SETI@home project which uses spare CPU cycles on remote systems to analyze radio telescope signal for possible communications in the Search for Extraterrestrial Intelligence (SETI). Though it is possible to use MPI over the

slow connections of the Internet, it is not feasible and as such there is no message passing between nodes of a conventional Grid computer. Why did all of this get started in the first place? The need for faster computation with lower cost. The Beowulf cluster was born out of the need for manipulating the large data sets involved in Earth Sciences such as weather prediction simulations. You may be wondering exactly how building a parallel computer system helped to solve this need for faster computers that could operate on larger data-sets. The very first clusters were actually not much faster than the computer in most homes now, but as the standard PC improved its performance over the years, so has the Beowulf cluster.

1.2 Parallel vs Serial

The very first place one will find the terms serial and parallel in regards to computers is in the communications. Computers communicate in one of two ways serial and parallel. We have all heard of Serial Cables and Parallel Cables, but what exactly is the difference? We can see by looking at them that the parallel cable is thicker, and has a larger end than a serial cable, but there is more too it than the size of the cable. Before going on a little needs to be said about how computers communicate, they communicate using the binary number system, the only thing computers actually know about are ones and zeros. A single one or zero is called a Bit, a group of eight bits is called a byte. A serial cable sends data from one computer component to another sending one bit at a time, whereas a parallel cable sends eight bits or one byte at a time. This is because a serial cable contains one wire loop between the devices for carrying data, and a parallel cable contains eight wire loops for carrying data. At the same clock rate, the space between bits, a parallel cable can send data eight times faster than a serial cable. Figure 1.1 shows the parallel model and a serial model sending the same sequence of ones and zeros.

We can think of parallel computing in much the same way that serial and parallel communications differ. Serial communications send one bit at a time, serial programming processes one instruction at a time. Parallel communications send eight bits at a time, parallel programming processes n instructions at a time, n representing the number of processes. As you can see with the communications model, if every instruction takes the same amount of time, then theoretically every process added cuts the computation time by a linear factor. This theoretical maximum cannot be reached in the real world because of the communications between processes slowing things down. Generally speaking, every parallel application reaches a point where adding more tasks slows the processing down. This point is where the communication between processes beings to be the limiting factor of the problem rather than the computation itself. It is good in developing parallel applications to realize when this maximum has been reached. Figure 1.2 shows a graph of the run time of a parallel program as the number of processes increases both the real world numbers and the theoretical maximums showing the impact of communications on parallel applications.

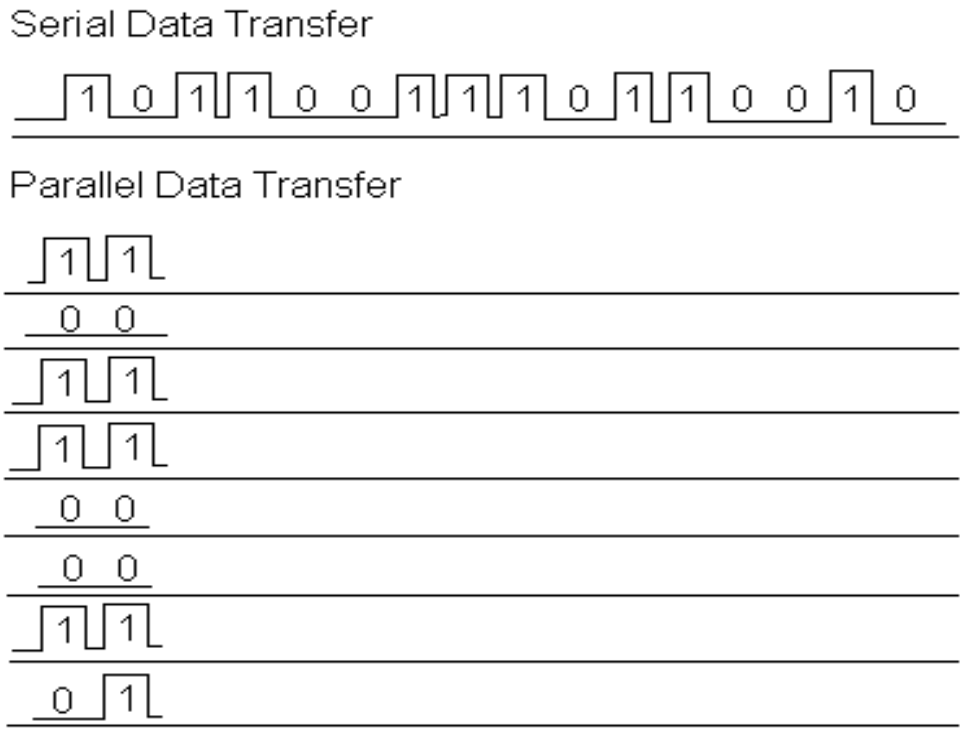

Figure 1.1: Shows parallel vs serial communications as a demonstration of way parallel processes are faster than serial processes.

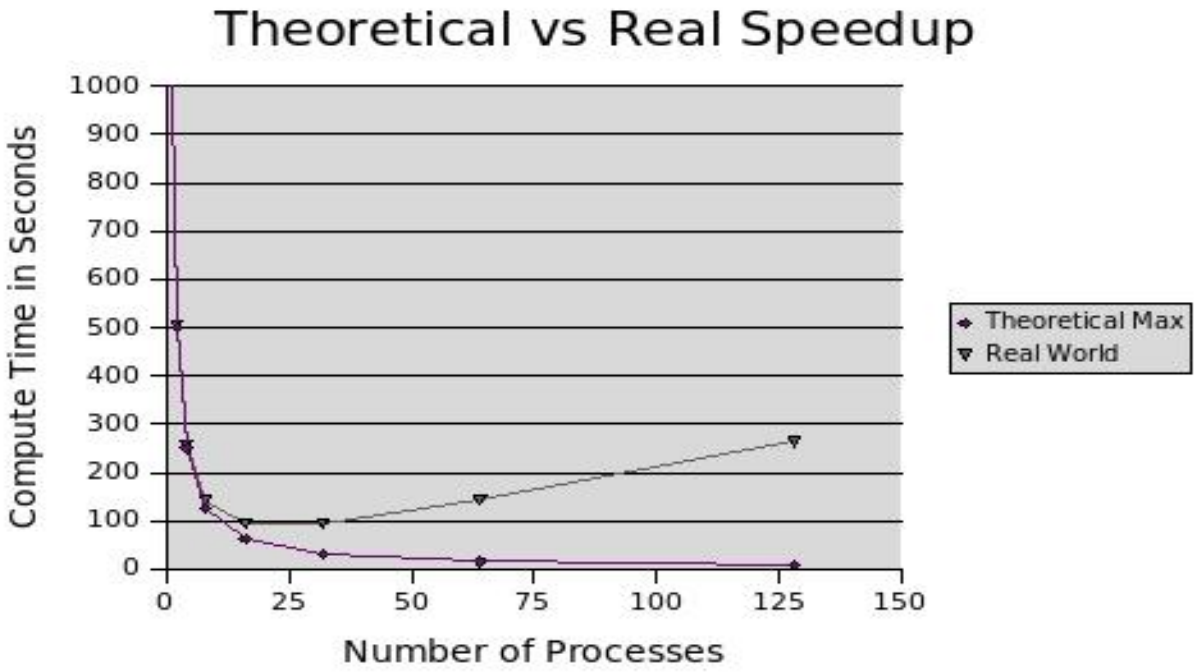

Figure 1.2: The impact of communications on parallel applications.

1.3 Serial Thinking

A serial problem is a problem or task that can only be done by one person, or device, and must follow a set order of instructions that all must follow one another in sequence. An example of a serial problem would be slicing a ham, more than one person cannot fit around a single ham, nor can the second slice be cut before the first. A serial computation problem would be long division. You always follow the same steps, you divide the first digits of the dividend by the divisor, multiply the divisor by the first digit of the quotient, subtract the result from the dividend, calculate the difference, and bring down the next digit of the divisor. These steps are repeated until the problem is complete. You cannot do any step before the others are complete and expect to get the correct answer. There are many serial problems that can be broken down into smaller problems and solved in parallel. A good example is peeling potatoes, if one person is peeling potatoes, it is a serial task, they can only peel one potato at a time, but if a group of people work together more potatoes get peeled in less time.

1.4 Parallel Thinking

Some things in life are embarrassingly parallel, meaning that they can be done by as many people or devices at a time as there are devices or people. A good example is cutting grass. Sure you could cut your grass one blade at a time with a tiny pair of scissors, if you wanted it to take all year, or you could invite ten thousand friends to each cut a single blade of grass in your yard and still not get it done in a day. The lawn mower is a parallel device. It cuts thousands of blades of grass at once and allows you to complete an average lawn in a matter of hours. If you had ten friends with ten mowers, you could mow your lawn in minutes. Mowing the lawn is an embarrassingly parallel task. Other things in life are inherently parallel, meaning there a parallel components to them, but they still have a serial component that must be done as one work until. A good example would be a math worksheet containing long division problems. One person doing the worksheet would have to do one problem at a time, but if twenty people worked on it twenty problems could be done at one time, but each problem has its serial steps. Another example is that of building a car. There are several stages to building a car, but the engine can be built at the same time the body is being built and the two larger parts assembled to form a complete car later in the process. The main goal of this first chapter is to help you find the parallel aspects of everyday problems, both mathematical problems and other problems.

1.5 The Need for More Computational Power

Over the past several years advancements in science have resulted in the need for more computational power. The more we learn about the way out physical world works, the larger the amount of data we collect, and the more computational power we need to process the collected data. Just a short time ago the standard home computer came with a 2 Gigabyte hard drive and we all wondered if we would every fill it up. Now you cannot even buy a hard drive less than 20 Gigabytes in size and the standard is in the hundreds of Gigabytes. Before the dawn of the computer

era, the ability to carry out a few hundred calculations an hour was impressive, but then the computer provided the ability to carry out a few hundred operations per second. The thought then occurred to them that if a computer could do hundreds of calculations a second, could it be faster still? The current top computer system in the world, based on the Top500.org list is the Earth-Simulator at the Earth Simulator Center in Japan with 5120 Processors operating at a speed of 40 trillion operations a second. Yet there is still a striving for more power. You may think 40 trillion operations a second is way faster than anyone could possibly need, but let us discuss what the Earth-Simulator is used for. This center has created a computational model of the earth and its weather systems. The goal is to predict future weather patterns and climatic changes based on historical information and current data. Let us consider for a moment the type of data necessary for this kind of predictive modeling.

A common approach to modeling any system is to break the system into a grid of data points. Current satellite technology gathers information at a resolution of around 10 square meters so let us take the entire planet and break it into 10 meter squares. The surface area of the earth is approximately 5.1 x 1014 square meters so this would result in 5.1 x 1012 squares. Just as a guess let us assume that we need to do 100 operations to model the weather for a one hour prediction of one square. This would require 5.1 x 1014 calculations to model the global weather. One trillion is 1x1012 so we need to be able to do 510 Trillion operations to predict one hour worth of weather. This would require a computer capable of 0.1416 Trillion Calculations per second. It is not hard to image the need for greater speed when you realized that the hurricane prediction models currently in use have thousands of variables and around ten thousand calculations performed to model the hurricane track it is not hard to imagine that we are way under on out weather model of the world and could easily need 140 Trillion calculations a second to just stay one second ahead of the weather make our predictions fairly useless. The governments of every world power have spent trillions of dollar in researching higher power computers for solving such problems and the results have been excellent, but we still have not reached the maximum potential, nor have we met the ultimate goal of accurately modeling earth systems for prediction and prevention of catastrophic weather events.

As you can easily see, the types of problems scientists are working on today require way more computational power than can be made possible with a single machine. This makes the need for parallel computers and parallel programmers, a constant need. It has been demonstrated by Peter Pacheco in his book Parallel Programming with MPI that it is impossible for a single computer to even perform the simple operation of adding two arrays of one trillion floats in a second, and we are want a computer to do calculations of 140 trillion calculations a second. He proves that just to store the data necessary we would need a way of storing 64-bits of data on a something smaller than an atom, and just the internal processor communications would have to be faster than the speed of light. Obviously if we cannot build a single computer to achieve our computations, we must find a way to use multiple computers working together on the common problem in order to arrive at a solution, by simple definition this is a parallel computer.

We have some good news, a system for utilizing computers in parallel has been designed and in use for several years. It has become very stable and is being used on a daily basis throughout the world. The bad news, it is not that easy to wrap

ones mind around the complexities of such a system. It is not as simple as giving each computer a program and letting it compute. Let us think for a moment about a human task, for example the writing of a book. Everyone has seen a text book with multiple authors. Can we just set each author down in a room and give them a notebook, a pencil, and the title of a book and tell them write a chapter of the book? Of course not, no one would know where to begin, what chapter to write, how it ties to the other chapters of the book, what the other authors are thinking, etc. It would not even be as simple as giving each author a chapter to write, you still would not be able to get a coherently flowing book without giving even more guidance and getting the authors to communicate with one another. This is a tough undertaking, but you of course would hire authors to write the book that understood the subject matter. Programming a parallel computer is much more difficult of a problem to wrap ones mind around, because it is like tackling the writing of a book with a group of authors that don't speak the same language, have never picked up a pencil, and have never heard about the topic of the book. The publisher not only would have to assign tasks to each author, but he would have to teach the authors a common language, teach them about the subject matter, and teach them how to use a pencil.

Just having a collection of processors, is exactly like having a collection of untrained people from different parts of the world working on a project. There is still a tremendous amount of work to be done before any processing can begin. Decisions must be made on the following:

- How the processors communication with each other?
- What information they need to share?
- Which calculations they must perform?
- How one processor's calculations affect the other processors?
- How do we divide the master algorithm into sub programs?
- How do we combine the results from each sub program into the final result?

These are critical issues in the development of parallel programming and one must begin to think about problems in a manner that allows them to be broken into individual parts that can be done concurrently and still come up with the same final result.

The good news is that much of the underlying communications, the common language, and the ability to share information has already been developed in a standard library call the Message Passing Library (MPI) throughout the remainder of this book we will be focusing on utilizing this standard library in C and C++. There will be many examples provided and many assignments made to help you learn the methods necessary to program a parallel system. There are other parallel programing libraries available and the concepts outlined in this book can be applied to any parallel system with the simple task of understanding the other libraries.

1.6 The Rest of The Book

Throughout the remainder of the book certain criteria will be followed when it comes to the examples, and system commands. The examples and system commands are

all based off of ROCKS 5.2 Cluster Distribution derived from CENTOS 5. Other RedHat based Linux distributions should follow the same conventions and aside from system nuances such as the package management systems all the examples should work correctly on any Linux distribution.

Discussion Questions

1. Describe a problem that can be solved in a serial manner and provide the steps necessary to perform the task. The problem can involve a real world situation such as preparing a meal, or a computation problem.

2. Describe a problem that can be solved in a parallel manner and provide the steps necessary to perform the task. The problem can involve a real world situation such as preparing a meal, or a computation problem.

3. Can you provide a method of performing the task in question one utilizing more than one resource (computer, person, etc).

4. Can you provide a method of performing the task in question two utilizing more than one resource (computer, person, etc).