Concurrent approach to Data Parallel Model using Java

Bala Dhandayuthapani Veerasamy

Abstract— Parallel programming models exist as an abstraction of hardware and memory architectures. There are several parallel programming models in commonly use; they are shared memory model, thread model, message passing model, data parallel model, hybrid model, Flynn's models, embarrassingly parallel computations model, pipelined computations model. These models are not specific to a particular type of machine or memory architecture. This paper expresses the model program for concurrent approach to data parallel model through java programming.

Keywords— Concurrent, Data Parallel, JDK, Parallel, Thread

I. Introduction

THE term processor, or microprocessor, refers to the central processing unit (CPU) [1]. It's a single chip responsible for the execution of instructions given by a computer program, memory management and address translation, integer and floating point operations, and cache management. Uniprocessors have a single processor. Application instructions and hardware calls are executed in order, one at a time (sequentially). The system appears to run concurrent processes, but the processor actually switches back and forth between instructions.

Two events are said to be concurrent if they occur within the same time interval. Two or more tasks executing over the same time interval are said to execute concurrently. Tasks that exist at the same time and perform in the same time period are concurrent. Concurrent tasks [1] can execute in a single or multiprocessing environment. In a single processing environment, concurrent tasks exist at the same time and execute within the same time period by context switching. In a multiprocessor environment, if enough processors are free, concurrent tasks may execute at the same instant over the same time period. The determining factor for what makes an acceptable time period for concurrency is relative to the application.

Concurrency techniques [3], [6] are used to allow a computer program to do more work over the same time period or time interval. Rather than designing the program to do one task at a time, the program is broken down in such a way that some of the tasks can be executed concurrently. In some situations, doing more work over the same time period is not the goal. Rather, simplifying the programming solution is the goal. Sometimes it makes more sense to think of the solution to the problem as a set of concurrently executed tasks. This technique is used in the parallel computer architectures.

Parallel programming and distributed programming [1] are two basic approaches for achieving concurrency with a piece of software. They are two different programming paradigms that sometimes intersect. In the past programming life, we were mostly using sequential programming. But, today's life style is going with more faster than the past decades. Also, solving problems on the computers are enormous. Parallel computer [1] can executes two or more job within a same period of time.

Java is just a computer language [5] that has secure, portable, object-oriented, multithreaded [3], [4], [6], interpreted, byte-coded, garbage-collected, language with a strongly typed exception-handling mechanism for writing distributed programs [4]. Java is an object-oriented programming language, which added the new features such as overriding, interface and etc. Java supports multithreaded programming, which allows you to do many things simultaneously on the same time interval. Java enables the creation of cross-platform programs by compiling into an intermediate representation called java byte code. JVM (Java Virtual Machine) is an interpreter for java. Java is designed for the distributed environment on the Internet. Java has technology called RMI (Remote Method Invocation) that brings unparalleled level of abstraction to client / server programming. Byte code is a highly optimized set of instructions designed to be executed by the java run-time system, which is called Java Virtual Machine (JVM). Java handles de-allocation for you automatically, this technique called garbage collection. The Java Developers Kit (JDK) is a set of command-line tools that can be used to create Java programs. The current release of the JDK is version 1.6

II. The concept of Data Parallel Model

The data parallel model [7] has characteristics that most of the parallel work focuses on performing operations on a data set. The data set is typically organized into a common structure, such as an array or cube. A set of tasks work

Department of Computing, Mekelle University, Mekelle, Ethiopia (Phone: +251 914740551; e-mail: dhanssoft@gmail.com).

collectively on the same data structure, however, each task works on a different partition of the same data structure.

The tasks can perform the same operation on their partition of work. On the shared memory architectures, all tasks may have access to the data structure through global memory. On the distributed memory architectures the data structure is split up and resides as "chunks" in the local memory of each task. Programming with the data parallel model is usually accomplished by writing a program with data parallel constructs. The constructs can be calls to a data parallel subroutine library or, compiler directives recognized by a data parallel compiler. Compiler directives allow the programmer to specify the distribution and alignment of data. FORTRAN implementations are available for most common parallel platforms.

FORTRAN 90 and 95 (F90, F95)**:** It is an ISO/ANSI standard extension to Fortran 77. It contains everything that is in Fortran 77. It has new source code format, additions to character set, additions to program structure, commands, variable additions methods, arguments, pointers, dynamic memory allocation, array processing, recursive, intrinsic functions and many other new features. These implementations are available for most common parallel platforms.

High Performance FORTRAN (HPF): It extensions to Fortran 90 to support data parallel programming. It contains everything in Fortran 90. It directives to tell compiler how to distribute data added. It assertions that can improve optimization of generated code added. The data parallel constructs can be added. These implementations are available for most common parallel platforms.

Distributed memory implementations of this model usually have the compiler convert the program into standard code with calls to a Message Passing Library (MPI) to distribute the data to all the processes. All messages passing are done invisibly to the programmer.

III. DATA PARALLEL PROGRAMMING IMPLEMENTATIONS

One of the characteristics that make Java a powerful programming language is its support of multithreaded programming [6] as an integrated part of the language. This is unique because most modern programming languages either do not offer multithreading or provide multithreading as a non-integrated package. Java ever offers a single integrated view of multithreading. Multithreading [6] is an extension of the multitasking paradigm. But rather than multiple programs, multithreading involves multiple threads of control within a single program. Not only is the operating system running multiple programs, each program can run multiple threads of control within the program. For example, using a Web browser, you can print one web page, download another, and fill out a form in a third-all at the same time. A thread is a single sequence of execution within a program. Thread behavior is completely dependent on the state a thread is in. The state of a thread defines its current mode of operation, such as whether it is running or not. Thread states are New, Runnable, Not running, and Dead. (See Fig. 1)

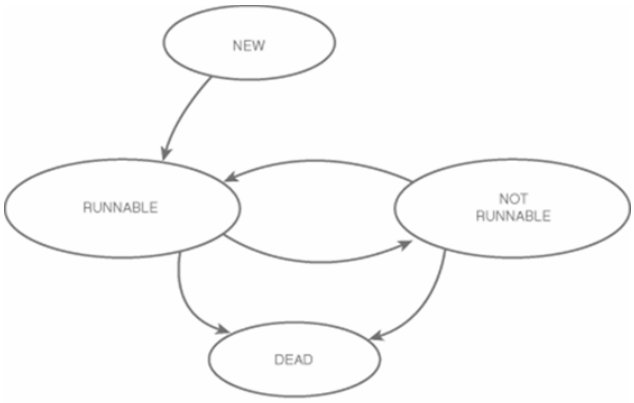

Fig.1 State of Thread

A thread is in the "new" state when it is first created until its start method is called. New threads are already initialized and ready to get to work, but they haven't been given the cue to take off and get busy. When the start method is called on a new thread, the run method is in turn called and the thread enters the "runnable" state. You may be thinking this state should just be called "running," because the execution of the run method means a thread is running. However, you have to take into consideration the whole priority issue of threads having to potentially share a single CPU. Even though every thread may be running from an end-user perspective, in actuality all but the one currently accessing the CPU are in a "runnable" wait state at any particular instant. You can still conceptually think of the "runnable" state as the "running" state; just remember that all threads have to share system resources. The "not running" state applies to all threads that are temporarily halted for some reason. When a thread is in this state, it is still available for use and is capable of re-entering the "runnable" state at some point. For each of these actions causing a thread to enter the "not running" state, there is an equivalent response to get the thread running again. A thread enters the "dead" state when it is no longer needed. Dead threads cannot be revived and executed again. A thread can enter the "dead" state through one of two approaches, which the run method finishes executing or stop method is called. The first approach is the natural way for a thread to die; you can think of a thread dying when its run method finishes executing as death by natural causes. In contrast to this is a thread dying by way of the stop method; calling the stop method kills a thread in an asynchronous fashion. Thread class provides constructors, methods to use this concept.

The following program 1 represents the concurrent approach to data parallel model. The result of the program is given in Fig. 3. In the program, class DP consists of four

threads called “task1, task2, task2, and task4”. The DP class consists of an array, initiated with totally 20 elements. In this, the “task1” uses 0 to 4, “task2” uses 5 to 9, “task3” uses 10 to 14 and the “task4” uses 15 to 19. This is illustrated in the Fig. 2. All the four tasks or threads will concurrently utilize the array called data parallel programming model.

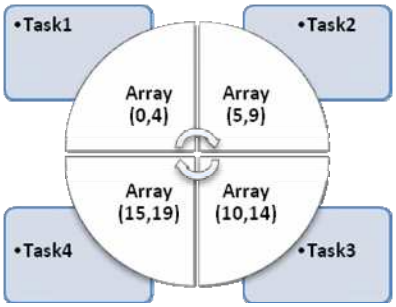

Fig. 2 Data Parallel Task Sharing on Array

Program 1. concurrent approach to Data Parallel

```
//DP.java
class DP implements Runnable{
int sum=0;
Thread t;
int aa[]=new int[20];
String name;
int beg, end;
static int ln=0;

DP(String str, int st,int en){
      t=new Thread(this,str);
      beg=st;
      end=en;
      name=str;
      t.start();   }

public void run(){
      try{  for(int i=beg; i<end;i++) {
         aa[i]=I; sum=aa[i];
         System.out.println(“Thread “+name +” “+sum);
         ln=ln+1;
         if(ln==4){
          System.out.println(“\n”);ln=0;}
         t.sleep(200);}
        }catch(Exception e){}  }

public static void main(String BDP[]){
   DP b1=new DP(“task1”,0,5);
   DP b2=new DP(“task2”,5,10);
   DP b3=new DP(“task3”,10,15);
   DP b4=new DP(“task4”,15,20);
   }}
```

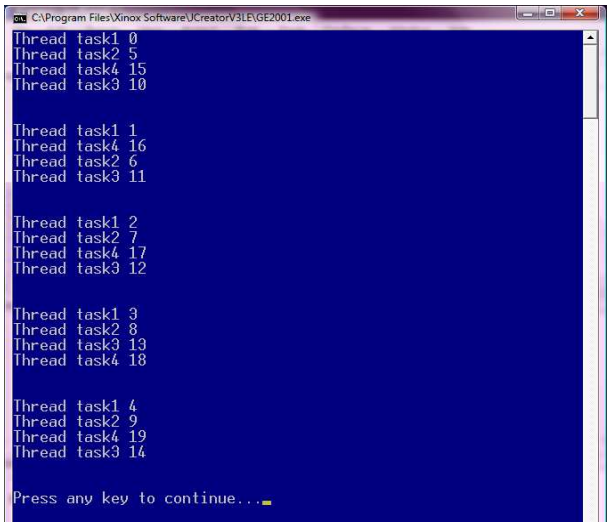

Fig. 3 Result of the Data Parallel Model

IV. Conclusion

Data parallel model usually have data parallel constructs or data sets. It allows accessing data values from data parallel constructs through numerous tasks or jobs. While executing a task over processor, it executes one task at a time. Though, task has got a change of accessing its own data values on data parallel constructs. Hence, this paper recommending you to have concurrent execution of task, which allow tasks utilize data parallel constructs at a time.

References

[1] Hesham El-Rewini, Mostafa Abd-El-Barr, “Advanced Computer Architecture and Parallel”, A John Wiley & Sons, Inc Publication, 2005.
[2] Tobias Wittwer, “An Introduction to Parallel Programming”, VSSD, 2006.
[3] Brian Goetz, Tim Peierls, Joshua Bloch, Joseph Bowbeer, David Holmes, Doug Lea, “Java Concurrency in Practice”, Addison Wesley Professional, 2006.
[4] Charles W. Kann, “Creating Components: Object Oriented, Concurrent, and Distributed Computing in Java”, Auerbach Publications, 2004.
[5] Peter Norton & Wiliam Stanek, “Java Programming”, Sams Publishing, 1996.
[6] Stephen J. Hartley, Concurrent Programming Using Java, Oxford University Press, 1998.
[7] Geoffrey C. Fox, Roy D. Williams, Paul C. Messina, Parallel Computing Works, Morgan Kaufmann Publishers, 1994.

Bala Dhandayuthapani Veerasamy was born in Tamil Nadu, India in the year 1979. The author was awarded his first masters degree M.S in Information Technology from Bharathidasan University in 2002 and his second masters degree M.Tech in Information Technology from Allahabad Agricultural Institute of Deemed University in 2005. He has published more than fifteen peer reviewed technical papers on various international journals and conferences. He has managed as technical chairperson of an international conference. He has an active participation as a program committee member as well as an editorial review board member in international conferences. He is also a member of an editorial review board in international journals.
He has offered courses to Computer Science and Engineering, Information Systems and Technology, since 8 years in the academic field. His academic

career started in reputed engineering colleges in India. At present, he is working as a Lecturer in the Department of Computing, College of Engineering, Mekelle University, Ethiopia. His teaching interest focuses on Parallel and Distributed Computing, Object Oriented Programming, Web Technologies and Multimedia Systems. His research interest includes Parallel and Distributed Computing, Multimedia and Wireless Computing. He has prepared teaching material for various courses that he has handled. At present, his textbook on "An Introduction to Parallel and Distributed Computing through java" is under review and is expected to be published shortly. He has the life membership of ISTE (Indian Society of Technical Education).