A Massively Parallel Virtual Machine for SIMD Architectures

M. Youssfi and O. Bouattane
Lab. SSDIA, ENSET
University Hassan II of Casablanca, Morocco

M. O. Bensalah
University Mohammed V of Rabat
Faculty of Science, BP 1014 Rabat, Morocco

Abstract

In this paper, we present a new model of a massively parallel Single Instruction Multiple Data (SIMD) structure machines in a distributed system. Among the modeled machines, we distinguish the linear, 2D, 3D meshes, pyramidal structures and GPU structure. All these computers are based physically on a multitude of fine grained processing elements (PE) arranged and coupled according to their associated topological pattern. In this model the host is represented by a distributed agent. Each virtual host agent, deployed in a physical computer, manages a local parallel virtual computer composed by a set of virtual processing elements (VPE). Each VPE is represented by a self threaded object. The distributed virtual host agents are interconnected through a multi agent system platform. The developed software is based on a hard kernel of a parallel virtual machine in which we translate all the physical properties of its different components. This kernel is based on an abstract layer which can be easily extended to the other topological structures. To implement a parallel program in the proposed platform, we have developed a new parallel programming language based on XML and its compiler which allow editing, compiling and running parallel programs. To illustrate the performance of this model, we present an example of a parallel program implementation for the edge detection of a brain MRI image presenting pathology.
Keywords: Parallel Programming, SIMD, Emulation, Parallel Virtual Machine

1. Introduction

In the field of high performance computing, the computationally intensive applications using high volumes of data require huge computing power and data storage. Today, the parallel and distributed computing is more necessary than ever. New massively parallel architectures have emerged and are being heavily exploited. This is the case of the Mesh Connected Computer (MCC) [1], the Reconfigurable Mesh Computer (RMC) [2, 3] or the graphics processing unit (GPU) [4]. However, the performance of these architectures remains limited. The theorists innovate more and more by imagining new parallel machines having well adapted topologies to specific problems. These innovations lead to new algorithms for high performance calculation. Unfortunately, the technology is not able to follow the scientist’s imaginations. Due to the unavailability or to the high cost of this kind of real parallel machines, we conclude that creating parallel virtual architectures is the first good way to validate and execute the parallel algorithms on serial machines. In this context, the realization of an emulator for SIMD parallel machines has been the first exploited model in our research works [5, 6, 7, 8]. This emulator has been of great use to elaborate, validate and test new parallel algorithms without need the real parallel machines. However, emulating a parallel machine on a single processor machine does not introduce any enhancement in terms of performance at runtime. It is therefore necessary to look for other ways to achieve these emulated parallel applications in a real environment that offers a high performance gain. In this paper, we present a new model to describe and emulate a polymorphic massively parallel single Instruction Multiple Data (SIMD) structure machines in a distributed system. Among the modeled machines, we distinguish the linear, 2D, 3D meshes, pyramidal structures and GPU structure. All these computers are based physically on a multitude of fine grained processing elements (PE) arranged and coupled according to their associated topological pattern. In this model the host is represented by a distributed agent. Each virtual host agent, deployed in a physical computer, manages a local parallel virtual computer composed by a set of virtual processing elements (VPE). Each VPE is represented by a self threaded object. The distributed virtual host agents are interconnected throw a multi agent system platform. This feature allows combining the performances of a set of physical computers to build a high performance massively parallel virtual machine. The developed software is based on a hard kernel of a parallel virtual machine in which we translate all the physical properties of its different components. This kernel can be easily extended to the other mentioned topological structures. To implement a parallel program in the proposed platform, we have developed a new parallel programming language based on XML and its compiler which allow editing, compiling and running parallel programs.
This paper is organized as follows. In Section 2, we will present the proposed massively parallel computational model. In the newt section, we illustrate the performance of this model by presenting an example of a parallel program implementation for the edge detection of a brain MRI image presenting pathology. Finally, the last section gives some concluding remarks and some exploiting perspectives.

2. The Proposed Parallel computational model

In this work, our virtual Reconfigurable Parallel Computer (PRC) is based on the reconfigurable 2D mesh easily extensible to the other mentioned architectures. A 2D Reconfigurable Mesh Computer is a massively parallel machine having M x N Processing elements (PEs) arranged on a 2-D matrix. It is a Single Instruction Multiple Data (SIMD) structure, in which each PE is localized in Cartesian coordinates (i, j). The PEs can carry out arithmetic and logical operations using its Arithmetic and Logic Unit (ALU). They can also carry out reconfiguration operations to exchange data over the mesh. Each PE is a self Thread autonomous component. All the PEs are managed by a virtual host manager represented by a distributed agent that is designed to load parallel program and global data to distribute them over the mesh of PEs for any execution. A virtual host agent can communicate with other host agents deployed in the distributed system. This feature gives the possibility to combine the performances of a set of distributed physical computers in order to build a massively parallel virtual machine.

![UML class diagram of the main components of the parallel virtual computer](image)

The proposed parallel virtual machine is represented by a set of distributed agents. Each agent represents the host of its local virtual machine. Each host has a set of
virtual processors arranged according to the topology of the virtual machine (2D, 3D pyramidal, etc...). The virtual PE is a self threaded objet which is ready to run the basic instructions of the parallel program broadcasted by the virtual host. In the proposed model, we have defined an abstract layer of the virtual machine which represents the core of this framework. We have defined some concrete implementations of the virtual machine, but the programmer can add other implementations extending the features of the model. The UML class diagram of Figure 1 shows the main components of the proposed model. In the realized framework, the parallel programmers must edit or open an edited parallel program and compile it before its execution. In the developed emulator, we have proposed a new parallel programming language based on XML language according to a specific developed XML schema.

3. Application

In this section, we present an example of a parallel algorithm implementation for contour detection of a gray levelled image using Sobel operator [9]. The operator uses two 3×3 kernels which are convolved with the original image to calculate approximations of the derivatives: one for horizontal changes, and one for vertical. If we define A as the source image, and Gx and Gy are two images which at each point contain the horizontal and vertical derivative approximations, the computations are as follows [10]:

\[
G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * A \quad \text{and} \quad G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A
\]

Where * here denotes the 2-dimensional convolution operation. The x-coordinate is defined here as increasing in the right direction, and the y-coordinate is defined as increasing in the down direction. At each point in the image, the resulting gradient approximations can be combined to give the gradient magnitude, using: \( G = \sqrt{G_x^2 + G_y^2} \)

In the sequential algorithm the complexity of this algorithm is evaluated to \( N \) where \( N \) represents the pixel count of the image. We will show that in this parallel implementation the complexity is \( O \) One time.

The structure of a parallel program implemented in the defined XML language is described as bellow:

0: <parallelProgram>
   1: <loadImage file="brain1.jpg" reg="0" codage="8"/>
   2: <markAllPEs/>
   3: <exchangeData reg="0" />  
   4: <compute  
   5: <compute  


Instruction 1: Loading the image the massively virtual machine:
As shown in figure 2, in this step, the gray level value of each pixel of the image is
loaded in the register 0 of each PE of the virtual computer.

Instruction 2: The statement <markAllPEs/> is used by the host agent
to mark all the virtual PEs which are designated to participate to the parallel computing.

Instruction 3: As shown in figure 3, In this instruction each PE will exchange
the data stored in register 0 with its 8 neighbours, if they exist. The received data are
stored in other registers reg[1], reg[2], reg[3], reg[4], reg[5], reg[6], reg[7] and reg[8]

Instructions 4, 5 and 6: Each PE compute Sobel operator Gx, Gy and G,
then save them, respectively, in reg[9], reg[10] and reg[1].

Figure 2: Data registers contents of 9 PEs after loading the image

Figure 3: Data registers contents of 9 PEs after the exchanging data operation

Figure 4 shows the parallel program result of a component contour detection of a
gray levelled image using Sobel operator. The image in figure 4.a represents the
input image representing a brain magnetic resonance image. The resulted output
image after the parallel Sobel algorithm is shown in figure 4.b.
4. Conclusion

In this paper, we have presented a new model of a polymorphic massively parallel single Instruction Multiple Data (SIMD) structure machines in a distributed system. In this model the host is represented by a distributed agent. Each virtual host agent, deployed in a physical computer, manages a local parallel virtual computer composed by a set of virtual processing elements (VPE). Each VPE is represented by a self threaded object. The obtained parallel virtual machine and its programming language compiler can be used as a high performance computing system. This platform can be used to resolve massively parallel applications related to other domains. In this model, we have proposed an abstract layer representing the core of the framework to allow other developers adding new extensions for new parallel virtual machine structures. Using a new parallel programming language based on XML is a good solution for this platform. However, we are working to develop other modules which allow writing parallel programs using the scientific programming languages like C++, Java or Fortran.

References


Received: February 4, 2015; Published: March 9, 2015