

Reconfigurable Architectures in Collaborative Grid Computing: An Approach

Stephan Wong and Mahmood Ahmadi

Computer Engineering Laboratory
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
`stephan,mahmadi@ce.et.tudelft.nl`

Abstract. The continuing need for faster high-performance computing is expected to continue in the coming years due to the increasing complexity of scientific and biologic computing applications. These applications are currently moving beyond the realm of geographically bounded supercomputers into distributed grids that tie together many millions of various heterogeneous computing systems. At the same time, we are witnessing an interesting trend in reconfigurable computing moving towards becoming more general-purpose in terms of utilization and especially programmability. Reconfigurable architectures have proven to be highly flexible and capable of quickly improving the performance of any application that has been targeted so far. Collaborative reconfigurable grid computing (CRGC) systems will exploit the availability of any reconfigurable hardware to accelerate compute-intensive tasks/operations. In this paper, we investigate using simple models and realistic assumptions to predict possible performance gains of these collaborative reconfigurable grids. The collaborative nature is further investigated by exploring the neighborhood concept that assumes neighbor nodes can request assistance to perform specific tasks when they are “free”. We will analyze the lower and upper bound of performance for CRGC systems. The results indicate that the proposed approach is capable of increasing the performance when compared to traditional grid systems that solely utilize general-purpose processors (GPPs).

Key words: Reconfigurable grid computing, collaborative processing, neighborhood policy, grid simulator

1 INTRODUCTION

In recent years, networked and reconfigurable platforms (mostly utilizing field-programmable gate arrays (FPGAs)) have gained much importance in many industrial applications/products and foremost academic research projects[4][5][8]. This is mainly due to the flexibility in increasing the performance of varying applications (both embedded and general-purpose) without the need to introduce a multitude of new individual hardware accelerators. As a result, they are becoming just another additional computing resource to be shared among many

applications running on the same (reconfigurable) processor that is stand-alone or connected/clustered in a local or wide-area network. This is opening up new research ground in exploiting the added performance of reconfigurable hardware in distributed and collaborative computing environments in large-scale heterogeneous or smaller-scale (wireless) networks. In such networks, idle processor cycles ‘are being used’ to assist other applications running elsewhere in the network by working on producing intermediate results or working on different data sets. In this paper, we describe our idea to build a network of heterogeneous processing elements that are able to collaboratively work on any task that has been inserted into the network on any processing element. Therefore, we provide a framework to collaborate between different processing elements and grid resources therefore, we define a neighborhood concept as a main part of collaboration policy that utilizes a set of primitives in a network of processing elements. These primitives implement different collaboration methods. To investigate the idea, we extended a version of traditional grid simulator (GridSim v4) that we called the collaborative reconfigurable grid simulator (CRGridSim) to support reconfigurable architecture modeling and neighborhood concept. We also analyze a lower and upper bounds of performance for this approach. The results show that the utilization of reconfigurable elements in grid computing increases performance when compared to the simple general-purpose processing elements in the grid and non-grid environments. This paper is organized as follows. Section 2 presents related work. Section 3 describes collaborative reconfigurable architectures on grid environments. Section 4 describes the implementation. Section 5 presents our simulation results. In Section 6, we draw the overall conclusions.

2 RELATED WORK

In this section, we take a brief look at the previous work regarding the high performance reconfigurable architectures. In [5], the design and implementation of a metacomputer designed to simplify the development of applications for clusters containing reconfigurable hardware are presented. The Distributed Reconfigurable Metacomputing (DRMC) has been defined as “the use of powerful computing resources transparently available to the user via a networked environment”. DRMC provides an environment in which computations can be constructed in a high-level manner and executed on clusters containing reconfigurable hardware. In DRMC applications are executed on clusters using the *Condensed graphs* model of computation that allows the parallelism inherent in applications to be executed by representing them as set of graphs. In [7], a set of rules and guidelines for the implementation of Distributed Processing Network (DPN) as the basis for a dynamic reconfigurable architecture is introduced that target improving the performance of microprocessor-based systems in computationally intensive application domains. Our approach introduces a different technique for collaboration between processing elements using neighborhood policy in grid environments where processing elements consist of reconfigurable and general purpose elements.

3 Collaborative Reconfigurable Architectures on Grid Environment

In this section, we present the concept of collaborative reconfigurable architectures, their properties, and performance analysis.

3.1 Collaborative Reconfigurable Architectures

General-purpose processors allow us to run the same program over a range of implementations of the same architectural family in a compatible manner. One of the major continuous concerns of general-purpose processors is performance. Reconfigurable hardware coexisting with a core processor has been considered a good candidate to address such a concern [4][5][7][8]. Using this technique, applications may be accelerated by delegating some of their most commonly used functionality to an FPGA co-processor configured with a suitable hardware implementation. In grid computing, a large pool of heterogeneous computing resources is geographically dispersed over a large network, e.g, the Internet, but collaborating in solving a single large and mostly complex scientific problem. Given the performance increases through grid computing and reconfigurable architectures, a combination of both approaches known as *Collaborative Reconfigurable Grid Computing* (CRGC). The general platform of CRGC is depicted in Figure 1(A).

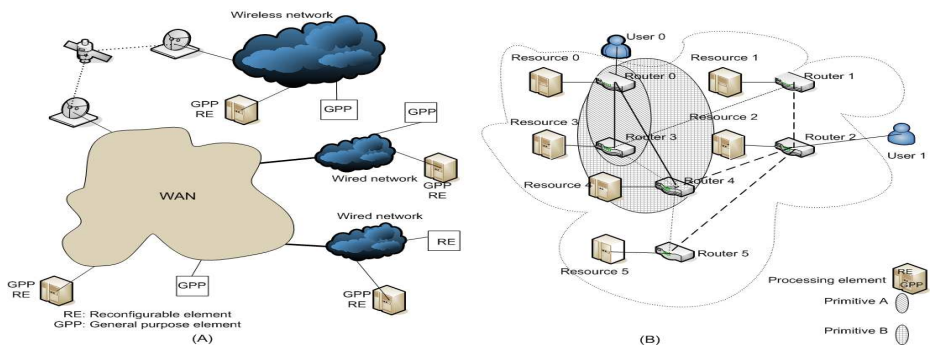


Fig. 1. (A) A collaborative reconfigurable architectures on grid environment with reconfigurable and general purpose elements. (B) A network with active primitives.

In traditional grid computing, the grid system is responsible for sending a job to a given machine to be executed. In CRGC, we introduce a different general job submission way in grid system (job submission policies are performed in these ways: scheduling, reservation, and scavenging [1]) that is called **neighborhood scheduling**. In this scheduling mechanism, each grid element requests assistance from neighborhood processing elements. In the neighborhood concept, the tasks can be inserted into the grid (network) through the grid elements. The neighborhood concept is implemented in different levels as follows: simple level

and hierarchical level. At the simple level, the neighbor processing elements are a direct neighbor to a requesting grid element. The direct neighbor is defined as a grid element that is physically located next to the current requesting grid element. The network backbone can be seen as a collection of primitives that the communication between requesting grid element and neighbors grid element are configured using these primitives. Some important primitives are depicted in Figure 2. Figure 2(A) depicts a primitive with one requesting element and one

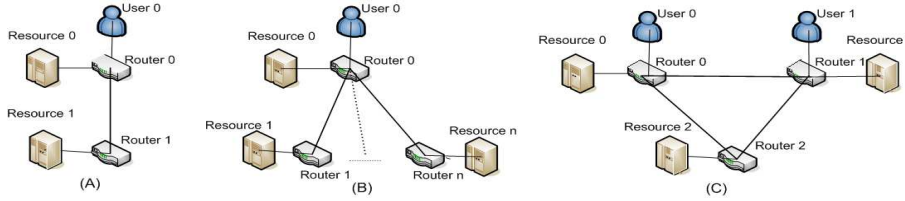


Fig. 2. Basic primitives that are utilized in neighborhood concept.

collaborator grid element. Figure 2(B) depicts a primitive with one requesting grid element and n collaborating grid elements (resources). A primitive with two requesting grid elements and one collaborating grid element is depicted in Figure 2(C). In these primitives, each grid resource consists of a general-purpose processor, reconfigurable element, or both. The neighborhood concept with active primitives in the network is depicted in Figure 1(B). Based on Figure 1(B), we can observe that each user and the related requesting grid element can find the correspondent neighbor grid element.

3.2 Lower Bound and Upper Bound of Performance Analysis

This analysis is performed to determine the lower and upper performance bound for the collaborative reconfigurable grid systems. To describe the related equations, the following notations are defined: N =number of subtasks, S =size of subtasks= MI instructions, T_{GPP} = processing time for each instruction by GPP, T_{RE} =processing time for each instruction by RE= T_{GPP}/k , T_{Com} = communication time between different processing elements in collaborative systems, k =speedup factor, N_1 =number of subtasks that can be processed by neighbors, Pkt =packet size, Bw =network bandwidth and N_{pkt} = number of packets. The following equations represent the processing time in collaborative and non-collaborative systems.

$$T_{Noncol} = \sum_{i=1}^N MI_i T_{GPP} \quad (1)$$

$$T_{Col} = \sum_{i=1}^{N-N_1} MI_i T_{GPP} + \sum_{i=N-N_1+1}^N MI_i T_{RE} + N_1 T_{Reconf} + T_{Com} = \sum_{i=1}^{N-N_1} MI_i T_{GPP} + \frac{\sum_{i=N-N_1+1}^N MI_i}{k} T_{GPP} + N_1 \frac{Reconf\ file\ size}{Reconf\ speed} + T_{Com} \quad (2)$$

In Eq. 2, T_{Com} is defined as follows:

$$T_{Com} = \text{packet trans time} \cdot N_{pkt} = \frac{Bw}{Pkt} N_1 \cdot S/Pkt \quad \text{with} \quad N_{pkt} = \frac{N_1 \cdot S}{Pkt} \quad (3)$$

In Eq. 3, the *packet trans time* represents the packet transmission time where the subtasks are organized as a sequence of packets to be sent to the neighbors. The *Reconf file size* represents the size of reconfiguration file and *Reconf speed* represents the reconfiguration speed for reconfigurable elements. Using these equations, we can find the lower and upper performance bound using metrics such as response time and waiting time. In Eq. 2, the third term represents the reconfiguration time for the processig elements. When the submitted tasks can not be distributed to the collaborator elements (reconfigurable elements), a GPP as main processing element executes all submitted tasks without any assistance from neighbors. Therefore, $T_{Com} = 0$, $T_{Reconf} = 0$ and $N_1 = 0$ and Eq. 2 for upper bound of performance is rewritten as follows:

$$T_{Col} = \sum_{i=1}^N MI_i T_{GPP} = T_{Noncol} \quad (4)$$

The lower bound can be found using Eq. 2. When all submitted tasks are executed using reconfigurable elements on high speed networks with small reconfiguration time Eq. 2 is represented as follows:

$$T_{Col} = \sum_{i=1}^{N_1} MI_i T_{RE} + N_1 T_{Reconf} + T_{Com} \quad \text{with} \quad N = N_1 \quad (5)$$

In Eq. 5, $N = N_1$ (all subtasks execute on reconfigurable elements), T_{Reconf} and T_{com} are very small. Therefore, we can rewrite Eq. 5 as follows:

$$T_{Col} = \sum_{i=1}^{N_1} MI_i \frac{T_{GPP}}{k} = \frac{1}{k \cdot N_1} T_{Noncol} \quad (6)$$

In the other words, the lower bound for execution time in collaborative system is $\frac{1}{N_1 \cdot k}$ execution time of non collaborative systems with N_1 being the number of reconfigurable elements (number of forwarded subtasks to reconfigurable elements) and k the average speedup of reconfigurable elements. This concept is expressed using the following inequality.

$$T_{Noncol} \leq T_{Col} < \frac{1}{k \cdot N_1} T_{Noncol} \quad (7)$$

Eq. 7 represents the lower bound and upper bound of performance in collaborative reconfigurable grid systems.

4 Implementation And a Case Study

In this section, we present a case study of CRGC. To investigate the behavior of CRGC, we extended GridSim (a traditional Java-based discrete-event grid simulator) that we called CRGridSim to study the behavior of the basic primitives as a case study[2][3][6]. We extended reconfigurable architecture modeling

and collaborative processing support using the neighborhood concept in Gridsim and after that investigate a case study including different primitives where these primitives construct a backbone of a reconfigurable grid architecture. In these primitives, different processing elements collaborate to solve a problem in a grid environment. Each processing element can be a general-purpose processor or a reconfigurable element. The processing elements can be defined (in the form of MIPS (Million Instructions Per Second) as per SPEC (Standard Performance Evaluation Corporation) benchmark). In CRGridsim, each application can be broken down to different subtasks called gridlets. Each application is packaged as gridlets whose contents include the task length in millions of instructions (MI). The task length is expressed in terms of the items of the time it takes to run on a standard GPP with a MIPS rating of 100. Each reconfigurable element accelerates the submitted subtasks compared to a GPP represented by a speedup factor. For example, a speedup factor of 10 means that the execution time is 10 times faster than when only using a GPP for the same task. Analysis of the problem based on different parameters such as: network bandwidth, reconfiguration method (e.g. partial or full), application model, background traffic, application size and hardware technology is complicated. Therefore, to observe the role of each parameter, we assume the other parameters are fixed. Hence, the following assumptions have been made:

- Reconfigurable elements do not support partial reconfiguration.
- There is not background traffic on the network and reconfiguration codes for reconfigurable elements are stored in the target collaborator elements since online code generation is time consuming.

5 Results

In this section, we present the simulation results using the CRGridSim simulator for the given primitives. These primitives were simulated with the following specifications: maximum packet size = 1500 Byte/sec, user-router bandwidth = 10 Mb/sec, router-router bandwidth = 100 Mb/sec, number of gridlets = 10, Number of users = 2, size of gridlets = 5000 MI (Million Instructions), the size of all gridlets is the same, MIPS rating of GPP = 377 MIPS, minimum speedup for reconfigurable elements=2, maximum speed up for reconfigurable elements =10, reconfiguration file size = 6 Mb, reconfiguration speed = 2 Mb/sec, reconfiguration time = reconfiguration file size/ reconfiguration speed = 3 sec.

The waiting time and turnaround time graphs for primitives in Figures 2(A) and 2(B) are depicted in Figures 3(A) and 3(B).

From Figures 3(A) and 3(B), we can observe that the utilization of collaborators decreases the waiting time and turnaround time. The lowest waiting time and turnaround time are produced when the number of collaborators is set to 3 RE with a speedup factor of 10. In Figure 3, the longest waiting time and turnaround time are when the resource encompasses a GPP without collaboration (straight line). The fluctuations on the graphs are due to reconfiguration

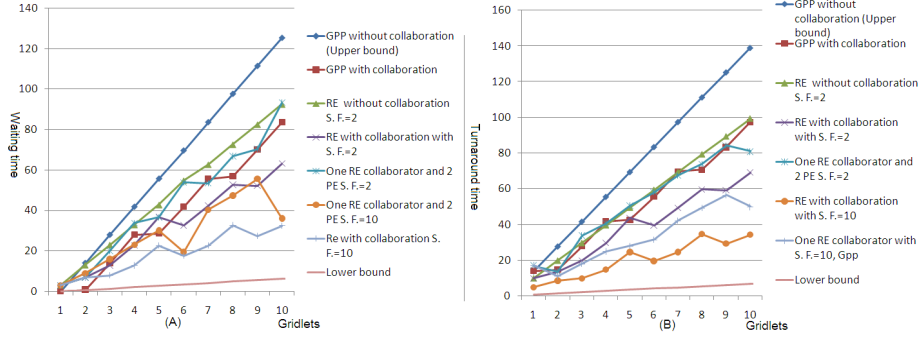


Fig. 3. The waiting and turnaround time graphs for the primitives in the Figures 2(A) and 2(B).

time in the collaborator resources. The waiting time and turnaround time graphs for the primitive in Figure 2(C) are depicted in Figure 4.

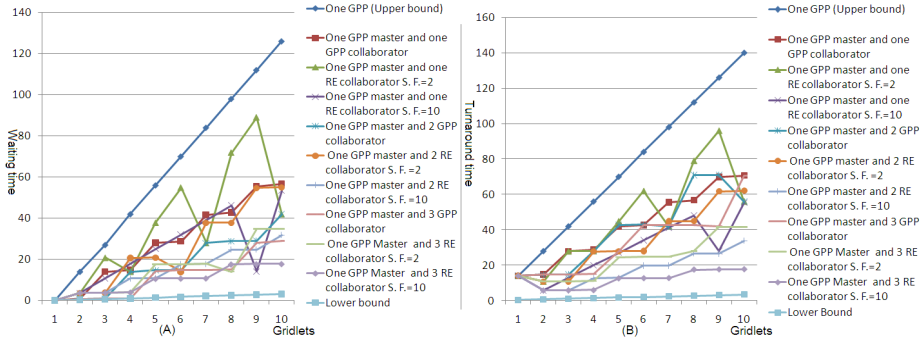


Fig. 4. The waiting and turnaround time graphs for the primitives in the Figure 2(C).

Figure 4, depicts the effect of a shared collaborator resource between different requesting resources. Since one shared collaborator resource (resource 2) is utilized by resource 0 and 1. From Figure 4, we can observe that the longest waiting time and turnaround time are when the resource encompasses a GPP without collaboration (straight line).

6 Overall Conclusions

In this paper, we presented the concept of collaborative reconfigurable grid computing in which grid resources utilize reconfigurable elements in the grid. Subsequently, an approach called neighborhood policy to collaborate between processing elements was proposed. We investigated the lower bound and upper bound of performance for the collaborative reconfigurable grid computing. The results show that CRGC increases performance in compared to traditional grid systems.

Therefore, it is beneficial to incorporate reconfigurable architectures in grid computing to perform certain compute-intensive tasks due to their inherent parallel architecture and speedup. In comparison to general-purpose architectures the utilization of reconfigurable architectures increase performance but some issues should be investigated in more details such as: mapping of tasks on to reconfigurable elements, code generation methods, and capabilities of neighbors elements to execute the source applications.

References

1. V. Berstis. “Fundamentals of Grid Computing”. <http://publib.boulder.ibm.com/Redbooks.nsf/>, November 2002. IBM Redbooks paper.
2. R. Buya and M. M. Murshed. “GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing”. *Concurrency and Computation: Practice and Experience*, 14(13-15):1175–1220, May 2002.
3. R. Buyya and M. Murshed. “GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing”. <http://www.citebase.org/abstract?id=oai:arXiv.org:cs/0203019>, 2002.
4. G.K. Kuzmanov, G. N. Gaydadjiev, and S. Vassiliadis. “The Molen Media Processor: Design and Evaluation”. In *Proceedings of the International Workshop on Application Specific Processors, WASP 2005*, pages 26–33, September 2005.
5. John P. Morrison, Philip D. Healy, and Padraig J. O’Dowd. “Architecture and Implementation of a Distributed Reconfigurable Metacomputer”. In *Proceedings. Second International Symposium on Parallel and Distributed Computing*, pages 153–158, October 2003.
6. Anthony Sulistio, Gokul Poduval, Rajkumar Buyya, and Chen-Khong Tham. “On Incorporating Differentiated Levels of Network Service into GridSim”. *Future Generation Computer Systems*, 23(4):606–615, 2007.
7. F. M. Vallina, E. Oruklu, and J. Saniie. “Distributed Processing Network Architecture for Reconfigurable Computing”. In *Proceeding of the International IEEE Conference Electro Information Technology*, volume 3, page 6, May 2005.
8. S. Vassiliadis, S. Wong, G. N. Gaydadjiev, K.L.M. Bertels, G.K. Kuzmanov, and E. Moscu Panainte. “The Molen Polymorphic Processor”. *IEEE Transactions on Computers*, 13:1363–1375, November 2004.