Collaboration of Reconfigurable Processors in Grid Computing for Multimedia Kernels

Mahmood Ahmadi[1], Asadollah Shahbahrami[2], and Stephan Wong[1]

[1] Computer Engineering Laboratory, Delft University of Technology, The Netherlands

[2] Department of Computer Engineering, Faculty of Engineering, University of Guilan, Rasht, Iran

{m.ahmadi,a.shahbahrami,j.s.s.m.wong}@tudelft.nl

Abstract. Multimedia applications are multi-standard, multi-format, and compute-intensive. These features in addition to a large set of input and output data lead to that some architectures such as application-specific integrated circuits and general-purpose processors are less suitable to process multimedia applications. Therefore, reconfigurable processors are considered as an alternative approach to develop systems to process multimedia applications efficiently. In this paper, we propose and simulate collaboration of reconfigurable processors in grid computing. Collaborative Reconfigurable Grid Computing (CRGC) employs the availability of any reconfigurable processor to accelerate compute-intensive applications such as multimedia kernels. We explore the mapping of some compute-intensive multimedia kernels such as the 2D DWT and the co-occurrence matrix in CRGC. These multimedia kernels are simulated as a set of gridlets submitted to a software simulator called CR-GridSim. In addition, the behavior of multimedia kernels in the CRGC environment is presented. The experimental results show that the CRGC approach improves performance of up to 7.2x and 2.5x compared to a GPP and the collaboration of GPPs, respectively, when assuming the speedup of reconfigurable processors 10.

Keywords: Reconfigurable processors, grid computing, multimedia kernels, high-performance computing.

1 Introduction

Multimedia standards such MPEG-1/2/3, JPEG 2000, and H.263/264 continually put increased strain on the performance of current and future processor architectures as new (usually more compute-intensive) algorithms are being introduced and the amount of data is increasing continuously. Examples of these architectures range from application-specific to domain-specific to multimedia-extended general-purpose processors (GPPs). Examples of complex algorithms that were recently introduced are 3D video rendering and real-time stereo vision processing. In essence, the major drawback is that the mentioned architectures are not able to fully marry high performance with high flexibility that is required in multimedia processing. A promising candidate to overcome this drawback are reconfigurable processors that contain a reconfigurable hardware fabric capable of changing its own functionality (static or dynamic) and execute at high speed (when enough parallelism is inherent in the algorithms). A logical "next step" in utilizing reconfigurable processors is combining them in a distributed grid computing environment to improve the performance and flexibility of the grid itself.

In this paper, we proposed the approach of collaborative reconfigurable grid computing (CRGC) that introduces reconfigurable processors in processing (grid) nodes and a scheme that allows for different nodes to cooperate together in processing a single application [13]. The main concept lies is the fact that the reconfigurable processors adapt themselves to the needed processing requirements (and functionality) without the need to introduce fixed hardware accelerators to improve the overall grid performance. Furthermore, we introduced the neighborhood concept as the collaboration concept between neighboring nodes in order to limit the communication throughout the whole grid. To investigate this concept, we introduced a set primitives (in which the communication could be simulated) and adapted the grid simulator GridSim v4 to simulate the CRGC concept together with the neighborhood concept. We termed the adapted simulator CRGridSim. To obtain results of real cases, we explored the mapping two computationally intensive multimedia kernels: the discrete wavelet transform (DWT) and the co-occurrence matric. These kernels were subdivided into gridlets for simulation in the CRGridSim simulator. Our experimental results show that speed-ups of up to 7.2 can be achieved when comparing to a grid with only general-purpose processor and assuming a per-processor speedup of a factor of 10 when comparing executing the same kernel in a GPP or a reconfigurable processor.

This paper is organized as follows. Section 2 presents related work. We describe the collaboration of reconfigurable processors (elements) in a grid environment in Section 3. In Section 4, we explain the chosen compute-intensive multimedia kernels in more detail. Simulation environment and tools are described in Section 5 followed by a discussion of the evaluation results in Section 6. Finally, conclusions are drawn in Section 7.

2 Related Work

In this section, we take a brief look at the previous work regarding high-performance reconfigurable computing. In [6], the design and implementation of a metacomputer based on reconfigurable hardware was presented. The Distributed Reconfigurable Metacomputing (DRMC) is defined as “the use of powerful computing resources transparently available to the user via a networked environment”. The DRMC provides an environment in which computations can be constructed in a high-level manner and executed on clusters containing reconfigurable hardware. In the DRMC architecture, 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 [8], a performance model for fork-join class and Synchronous Iterative Algorithm (SIA) was presented. They considered the division of computation between the workstation processor and the reconfigurable processor. They focused on algorithms and applications that fit into the fork-join class and SIAs types.

In [12], the 2D-FFT application has been implemented on both the standard cluster and the prototype Adaptable Computing Cluster (ACC). The ACC is an architecture that attempts to improve high-performance cluster computing with FPGAs, but not by merely adding reconfigurable computing resources to each node. Rather, by merging cluster and reconfigurable technologies and enhancing the commodity network interface. In [11], performance of single reconfigurable processor for grey level co-occurrence matrix (GLCM) and Haralick texture feature for image sizes $512*512$, $1024*1024$ and $2048*2048$ was presented. Speedups of 4.75 and 7.3 were obtained when compared with a general-purpose processor for GCLM and Haralick co-occurrence matrix, respectively. The target hardware for this work was Celoxica RC1000-PP PCI-based FPGA development board equipped with a Xilinx XCV2000E Virtex FPGA. In addition, a co-occurrence matrix media kernel has been implemented on the various FPGA devices such as Virtex2 and Spartan3 and on a media-enhanced GPPs using MMX technology in [5]. Speedups of 20 were obtained using FPGA implementations over media-enhanced GPPs, for an image size $512 * 512$.

In [13], we investigated the concept of collaboration of reconfigurable processors in grid computing. In this paper, we simulate several computationally intensive media kernels and map on the proposed architecture. The experimental results show that collaboration of reconfigurable processors in grid computing achieves much more performance than the collaboration of GPPs.

In current work, we further investigate the performance of CRGC by looking at realistic loads and real kernels execution characteristics.

3 Collaboration of Reconfigurable Processors in Grid Computing

In this section, we present the concept of collaboration of reconfigurable processors and their properties. In grid computing, a large pool of heterogeneous computing resources is geographically dispersed over a large network, e.g., the Internet. Our approach to achieve high-performance and flexibility is to utilize reconfigurable processors in grid computing. We termed the utilization of reconfigurable processors that collaborate together in grid environment Collaboration Reconfigurable Grid Computing (CRGC). The general platform of CRGC is depicted in Figure 1. Reconfigurable elements are a part of the resources in grid computing.

Processing processors offload part of their computational workload to reconfigurable computing resources. In this type of computing, various software codes targeting different processing architectures are stored either in a centralized or a decentralized manner and must be distributed to the computing resources when needed. In CRGC, processing elements communicate and collaborate together based on the *neighborhood concept*. Each grid processing element requests assistance from neighboring processing elements. The tasks can be inserted into the

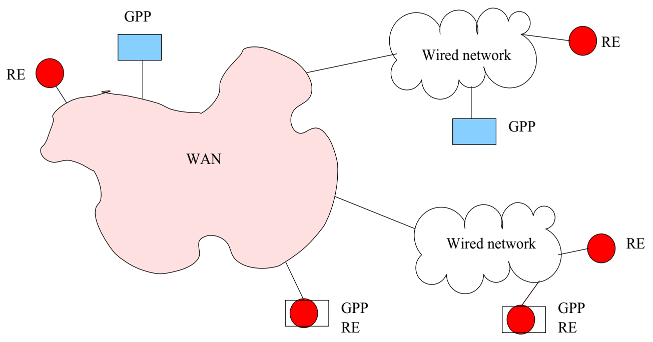

Fig. 1. A general view of a collaboration of reconfigurable elements in grid environment. RE shows reconfigurable elements (processors).

grid through existing grid elements. In our implementation of the neighborhood concept, 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 (or geographically closely) located next to the current requesting grid element. The neighborhood concept is defined by some primitives. A primitive is defined as a processing element with related communication link and its equipments, e.g., routers and switches, to the main processing element. The network backbone can be seen as a collection of primitives. Some important primitives are depicted in Figure 2.

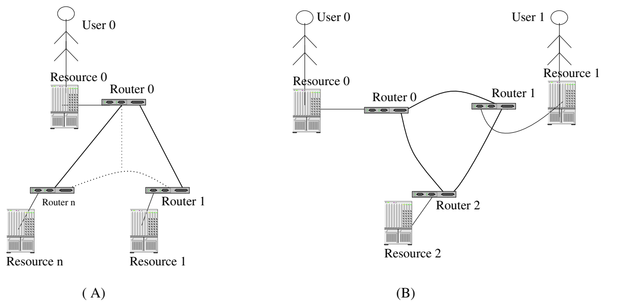

Fig. 2. Basic primitives that are utilized in neighborhood concept

Figure 2 (A) depicts a primitive with one requesting processing element and n collaborating processing elements. A primitive with two requesting processing elements and one collaborating processing element is depicted in Figure 2 (B). The neighborhood concept with active primitives in the real grid is depicted in Figure 3.

Based on Figure 3, we can observe that each user and the related requesting processing element can find the correspondent neighbor processing element. For example, user 0 and resource 0 can operate based on primitive in Figure 2 (A). From Figure 3, in the first scenario, resource 0 is assisted by resource 1 and

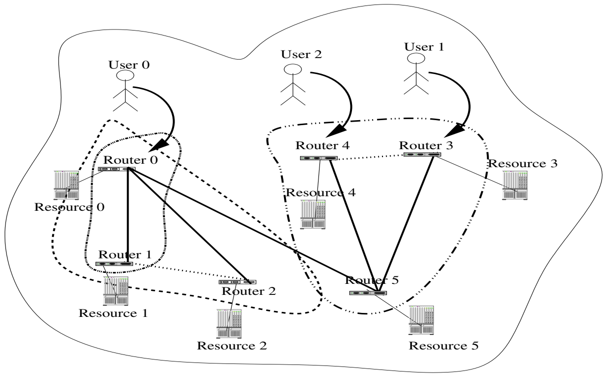

Fig. 3. Active primitives in the sample network

in the second scenario, resource 0 is assisted by resources 1 and 2. We have a similar condition for user 1, in this case resource 3 gets help from resource 5.

4 Multimedia Kernels

In this section, we explain the two chosen compute-intensive multimedia kernels, discrete wavelet transform and co-occurrence matrix. We have selected these multimedia kernels because they are compute-intensive [7].

4.1 Discrete Wavelet Transform

The digital wavelet representation of a discrete signal X consisting of N samples can be calculated by convolving X with the lowpass and highpass filters and down-sampling the output results by 2, so that the two frequency bands each contains $N/2$ samples. With the correct choice of filters, this operation is reversible. This process decomposes the original image into two sub-bands: the lower and the higher band [9][7]. This transform can be extended to multiple dimensions by using separable filters. A 2D DWT can be performed by first performing a 1D DWT on each row (*horizontal filtering*) of the image followed by a 1D DWT on each column (*vertical filtering*).

4.2 Co-occurrence Matrix

Texture features are usually used in image and video processing. Texture features determine the dependencies between neighboring pixels within region of interest in an image [3]. The co-occurrence matrix captures second order gray level information, which is a well known tool for texture analysis. Haralick et. al [4] have defined some texture features which use co-occurrence matrices. The features are related to neighboring pixels at different directions and distance. The number of occurrences of two neighboring pixels with a distance d and with a certain direction is stored in co-occurrence matrix. The co-occurrence matrix is always a square matrix of size $N_{gl} \times N_{gl}$, where N_{gl} is the number of available gray levels in the image. Consequently, the size of this matrix is independent from distance and direction neighboring pixels and also from the image size. Co-occurrence matrix consists of relative frequencies $P(i, j; d, \delta)$ of two neighboring pixels i, j separated by distance d at orientation δ in an image.

5 Simulation Environment and Tools

In this section, we present our simulation environment. We investigated multimedia kernels as case study on a network that includes primitives to construct a backbone of reconfigurable processors in grid. In these primitives, different processing elements collaborate together to execute a multimedia kernel in a grid environment. Each processing element can be a GPP or a reconfigurable processor (Reconfigurable Element (RE)). The specification of processing elements is defined in the form of Million Instructions Per Second (MIPS) for the Standard Performance Evaluation Corporation (SPEC) benchmark. In this paper, we have used 30, 35, 40 and 50 MIPS for processing elements.

In order to understand how many instructions are required to execute the discussed multimedia kernels, we have executed both the 2D DWT and co-occurrence matrix kernels using the SimpleScalar toolset [1] for an image of size 1024×1024. The number of committed instructions for the 2D DWT kernel is 46160416, while the number of committed instructions for the second kernel is 83899404. This means that in order to provide each value for the first decomposition level of 2D DWT, 44 instructions should be processed, while for the second kernel 80 instructions should be processed.

The simulation environment is an extended version of the GridSim (a traditional Java-based discrete-event grid simulator) [2][10]. We configured and prepared the GriSim simulator based on the multimedia kernels properties to support the collaborative processing between reconfigurable processors. This extension of the GridSim is called CRGridSim. 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 time it takes to run on a standard GPP. To simplify the simulation of the proposed approach, we assumed that reconfigurable processors do not support partial reconfiguration.

The multimedia kernels were simulated using the primitive in Figure 2 (A) with 3 and 2 collaborator processing elements. The specifications of the simulated environment and primitives are depicted in Table 1. The speedup of reconfigurable processors over GPPs for multimedia kernels has been set to 10. This is because, as we mentioned in Section 2 the average speedup for the multimedia kernels in single reconfigurable processor is $(4.75 + 7.3 + 20)/3$.

The images size and specification of the processing elements are presented in Table 2 and Table 3, respectively. In Table 2, 4 groups of different images with various sizes have been collected. Each image is sent in the uncompressed form to the processing elements. The required instructions to process each image is packed

Table 1. Specifications of the simulated environment

Parameter	**Value**
Maximum packet size	32 and 65 KBytes
User-router bandwidth	100 Mb/sec
Router-router bandwidth	1000 Mb/sec
Number of images	40 (Table 2)
Number of users	1
Size of images	Based on Table 2
PE specification (MIPS)	Based on Table 3
Speedup for RE	10 in compared to a GPP
Reconfiguration file size	3 Mb
Reconfiguration speed	3 Mb/sec
Reconfiguration time	1 sec
Number of bits per pixel	24 bit

Table 2. Images and their correspondence gridlets specifications for different multimedia kernels

Image			**# of instructions**	
Group	**Size**	**# of images**	**2D DWT**	**Co-occ. matrix**
1	768 × 1024	10	35	64
2	1024 × 1024	10	46	84
3	1200 × 1600	10	84	156
4	2134 × 2848	10	267	493

Table 3. The specification of processing elements in terms of MIPS

Processing elements	**MIPS**
Main GPP	30
Collaborator 1 (GPP or RE)	35
Collaborator 2 (GPP or RE)	50
Collaborator 3 (GPP or RE)	40

up as a gridlet and is submitted to the related processing elements either a GPP or a RE. Table 3 depicts the specification of processing elements in terms of MIPS.

In the simulation environment using CRGridsim, the following steps should be considered in order to execute an application using CRGC. First, a network topology based on the availability of neighbor processing elements is defined and parameters such as network bandwidth and packet size are configured. Second, the processing elements are defined and reconfigurable processing elements are configured based on the application characteristics. Third, an application mapping policy and the number of subtasks (gridlets) for each application are determined. Finally, the main processing element packetizes the subtasks and send them to the appropriate REs. Additionally, the collaborator

REs depacketizes the received packets and process them and send back the calculated results to the main processing element. Finally, the main processing element receives the final results and send the remaining other subtasks to the idle REs.

6 Experimental Results

In this section, we present the experimental results which have been obtained using the CRGridSim simulator for the 2D DWT, co-occurrence matrix, and combination of both kernels.

In order to evaluate the proposed approach, we considered two packet sizes, 32 KBytes and 65 KBytes (the largest packet sizes in the networks). Our results show that using larger packet sizes lead to higher performance than smaller packet sizes. Larger packet sizes decreases the communication overhead due to sending less packets. In our case, we utilized 40 images with a total size of 294 MBytes. These images translate into 4539 packets of 65 KBytes or 9219 packets of 32 KBytes. In addition, we considered four different configurations, collaboration of 3 GPPs, a GPP with 2 REs, 4 GPPs, and a GPP with 3 REs. The configuration of 3 GPPs means that 2 GPPs are collaborating with a main GPP. The star topology has been used for the collaboration mechanism that a structure of this topology was depicted in Figure 2 (A).

Figure 4 depicts the speedups of the first two configurations, 3 GPPs and a GPP with 2 REs over a GPP. Figure 5 depicts the speedups of the last two configurations, 4 GPPs and a GPP with 3 REs, over a GPP.

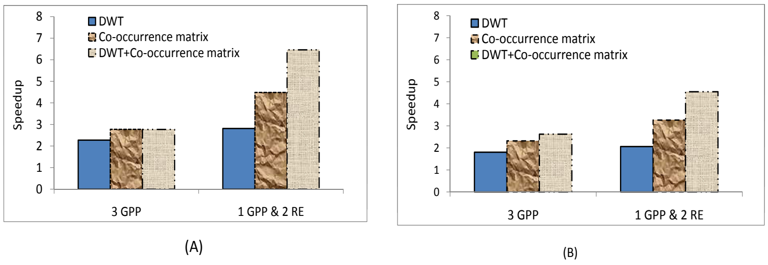

Fig. 4. Speedup for different configurations with 2 collaborator processing elements over one GPP. (A) Maximum packet size is 65 KBytes. (B) Maximum packet size is 32 KBytes.

The packet size for the Figures 4 (A) and 5 (A) is 65 KBytes, and the packet size for the Figures 4 (B) and 5 (B) is 32 KBytes. Our observations from these figures are the following. First, increasing the packet size from 32 KBytes to 65 KBytes improves the performance. As can be seen that the speedups of the left side figures are larger than the speedups of the right side figures. This is due to the fact that larger packet sizes decreases the communication overhead. Second, the collaboration of reconfigurable processors improves the performance

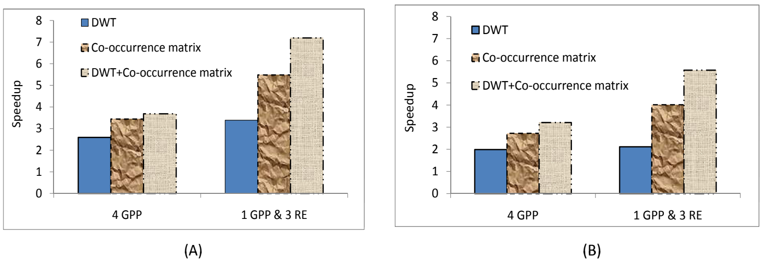

Fig. 5. Speedup for different configurations with 3 collaborator processing elements over one GPP.(A) Maximum packet size is 65 KBytes. (B) Maximum packet size is 32 KBytes.

more than the collaboration of GPPs. For our configuration, the performance improvement of the collaboration of reconfigurable processors over the collaboration of GPPs is of up to 2.5. Finally, executing computationally intensive applications yields much more performance than non-computationally intensive applications. This is because the impact of the communication overhead will be reduced compared to the computational time. As can be seen in those figures, a combination of both kernels obtains more speedups than the execution of each kernel separately. Additionally, increasing the number of collaborator processing elements improves the performance. This is because the submitted subtasks to each collaborator are decreased. This reduces the number of processed instructions by each processing element.

7 Conclusions

Multimedia kernels are compute-intensive tasks. In order to increase the performance of multimedia kernels, we have proposed and simulated the collaboration of reconfigurable processors in grid computing. The Collaborative Reconfigurable Grid Computing (CRGC) utilizes reconfigurable processing elements capabilities in the grid environment. We studied and mapped some multimedia kernels in a grid environment. This environment was simulated using CRGridSim simulator. The results show that the utilization of CRGC improves performance of up to 7.2 in comparison to a GPP when assuming the speedup of reconfigurable processors 10. The results show that the proposed CRGC approach is capable of improving the performance of compute-intensive multimedia kernels more than none compute-intensive kernels due to communication overhead.

References

1. Austin, T., Larson, E., Ernst, D.: SimpleScalar: An Infrastructure for Computer System Modeling. IEEE Computer 35(2), 59–67 (2002)
2. Buyya, R., Murshed, M.M.: 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 (2002)

3. Conners, R.W., Harlow, C.A.: Theoretical Comparison of Texture Algorithms. IEEE Trans. on Pattern Analysis and Machine Intelligence 2(3), 204–222 (1980)
4. Haralick, R.M., Shanmugam, K., Dinstein, I.: Textural Features for Image Classification. IEEE Trans. on Systems, Man, and Cybernetics 3(6), 610–621 (1973)
5. Iakovidis, D.K., Maroulis, D.E., Bariamis, D.G.: FPGA Architecture for Fast Parallel Computation of Co-occurrence Matrices. Microprocessors and Microsystems 31, 160–165 (2007)
6. Morrison, J.P., Healy, P.D., O'Dowd, P.J.: Architecture and Implementation of a Distributed Reconfigurable Metacomputer. In: Proc. 2nd Int. Symp. on Parallel and Distributed Computing, October 2003, pp. 153–158 (2003)
7. Shahbahrami, A., Ahmadi, M., Wong, S., Bertels, K.L.M.: A New Approach to Implement Discrete Wavelet Transform using Collaboration of Reconfigurable Elements. In: Proc. of Int. Conf. on ReConFigurable Computing and FPGAs (2009)
8. Smith, M., Peterson, G.D.: Parallel Application Performance on Shared High Performance Reconfigurable Computing resources. Performance Evaluation 60(1-4), 107–125 (2005)
9. Stollnitz, E.J., Derose, T.D., Salesin, D.H.: Wavelets for Computer Graphics: Theory and Applications. Morgan Kaufmann, San Francisco (1996)
10. Sulistio, A., Poduval, G., Buyya, R., Tham, C.K.: On Incorporating Differentiated Levels of Network Service into GridSim. Future Generation Computer Systems 23(4), 606–615 (2007)
11. Tahir, M.A., Bouridane, A., Kurugollu, F., Amira, A.: Accelerating the Computation of GLCM and Haralick Texture Features on Reconfigurable Hardware. In: Proc. of the Int. Conf. on Image Processing, pp. 2857–2860 (2004)
12. Underwood, K.D., Sass, R.R., Ligon, W.B.: Acceleration of a 2D-FFT on an Adaptable Computing Cluster. In: 9th Annual IEEE Symp. on Field Programmable Custom Computing Machines (FFCM 2001), pp. 180–189 (2001)
13. Wong, S., Ahmadi, M.: Reconfigurable Architectures in Collaborative Grid Computing: An Approach. In: Proc. 2nd Int. Conf. on Networks for Grid Applications (2008)