On Incorporating Reconfigurable Architectures into Grid Environments Using GridSim

Mahmood Ahmadi, and Stephan Wong
Computer Engineering Laboratory
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
{mahmadi, stephan}@ce.et.tudelft.nl

Abstract— **Grid computing has emerged as the next-generation parallel and distributed computing methodology, which aggregates dispersed heterogeneous resources for solving various kinds of large-scale parallel applications in science, engineering and commerce. In order to evaluate the performance of Grid resource management and application scheduling algorithms, we need to conduct repeatable and controlled experiments, which are difficult due to Grid's inherent heterogeneity and its dynamic nature. Additionally, Grid testbeds are limited, and creating an adequately-sized testbed is expensive and time consuming. Moreover, resources are autonomous and owned by different organizations, it requires the handling of different administration policies at each resource. Due to these reasons, it is easier to use simulation as a means of studying complex scenarios. GridSim is a software platform that enables users to model and simulate the characteristics of Grid resources and networks with different configurations. 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. In this paper, we present an extension of Gridsim simulator to incorporate reconfigurable architectures in grid environments that called collaborative reconfigurable grid simulator (CRGridsim). CRGridsim enables gridsim to collaborate reconfigurable architectures on grid.**

Keywords: **Reconfigurable architecture, Grid computing, Gridsim**

I. 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[5][6][9]. 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. In further increasing their applicability, they are placed close to programmable processors (e.g., PC processors) and are starting to be tightly integrated with instruction-set architectures of these processors. 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. The characteristics of this network are:

- The processing elements comprise a general-purpose processor augmented with reconfigurable hardware elements which the general-purpose processor can be of any architecture and the reconfigurable hardware components can be implemented in any technology.
- The general-purpose processor is the main processor that controls the processing elements in the following sense:
  1. Reconfiguration of reconfigurable elements of processing element.
  2. Decide when and how to use the reconfigurable elements.
  3. Decide when to ask for assistance from neighborhood processing elements.

To achieve the mentioned characteristics 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 v.4) that we called the collaborative reconfigurable grid simulator (CRGridSim) to support reconfigurable architecture modeling and neighborhood concept. The CRGridSim toolkit provides facilities for the modeling and simulation of resources (general-purpose and reconfigurable elements) and network connectivity with different capabilities, configurations. The results show that the utilization of reconfigurable elements in grid computing increases performance when compared to the simple GPP 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 implementation. Section 5 presents our simulation results. Section 6, we draw the overall conclusions.

II. RELATED WORK

In this section, we take a brief look at the previous work regarding the high performance reconfigurable architecture. In [6], 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 as 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 [4], a limited flooding approach for mobile ad hoc networks was proposed that utilized neighbors information to limited broadcast redundancies. It reduced control overhead of ad hoc routing protocols and introduced some benefits including efficient flooding, density and mobility adaptation. In [8] 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 targeted at improving the performance of microprocessor based systems in computationally intensive application domains. DPN presents the general architecture for solving the problem using reconfigurable computing. It does not include any result to show the reasonability of the proposed solution. Our approach, utilizes an extended version of Gridsim toolkit that called CRGridsim to simulate and model collaborative reconfigurable architecture behavior on grid environment.

III. Collaborative Reconfigurable Architectures on Grid Environment

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

A. 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. Furthermore, they allow various programs to run on the same system and the same program to run over multiple processing families. 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 [5][6][8][9][10]. 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. Not all applications are amenable to acceleration with reconfigurable hardware; the part of the application to be accelerated must typically exhibit a high degree of intrinsic parallelism and data locality in order to make the conversion to hardware worthwhile[6]. 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).

In CRGC, mostly locally and possibly wirelessly networked (low power/low-performance) processing elements offload part of their computational workload to higher-performance computing resources. In both types of computing, various software codes targeting different processing architectures are stored either in centralized or decentralized manner and must be distributed to the computing resources when needed. 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.

B. Collaborative Reconfigurable Architectures and Gridsim

The GridSim toolkit provides a comprehensive facility for simulation of different classes of heterogeneous resources, users, applications, resource brokers, and schedulers. The features of the GridSim toolkit include the following:

- It allows modeling of heterogeneous types of resources.
- Resources can be modeled operating under space- or time -shared mode.
- Resource capability can be defined (in the form of MIPS as per SPEC benchmark).
- Resources can be located in any time zone.
- Weekends and holidays can be mapped depending on

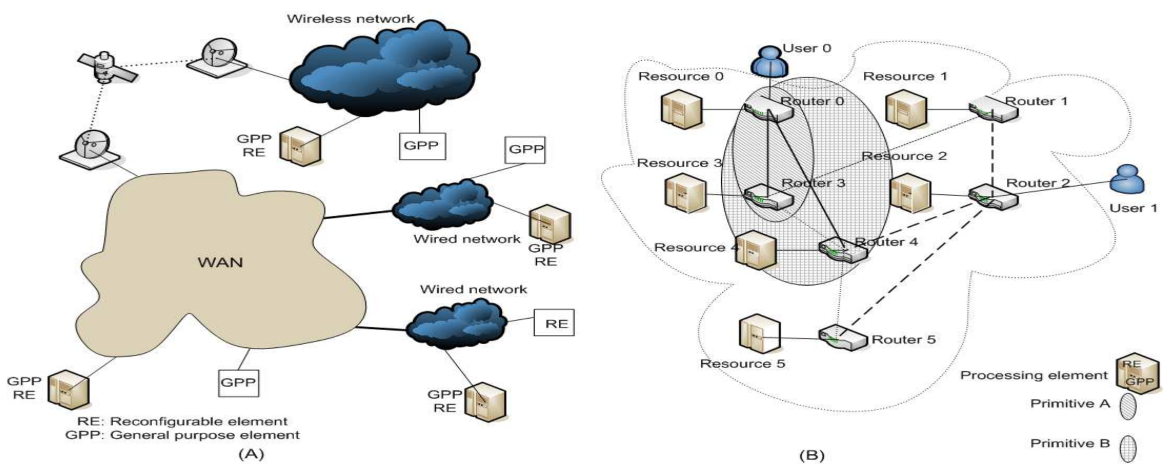

Fig. 1. (A) A collaborative reconfigurable architectures on grid environment with reconfigurable and general purpose elements. (B) A network with active primitives.

resources local time to model non-Grid (local) workload.

- Resources can be booked for advance reservation.
- Applications with different parallel application models can be simulated.
- There is no limit on the number of application jobs that can be submitted to a resource.
- Application tasks can be heterogeneous and they can be CPU or I/O intensive.
- Multiple user entities can submit tasks for execution simultaneously in the same resource, which may be time -shared or space-shared. This feature helps in building schedulers that can use different market-driven economic models for selecting services competitively.
- Network speed between resources can be specified.
- It supports simulation of both static and dynamic schedulers.
- Statistics of all or selected operations can be recorded and they can be analyzed using GridSim statistics analysis methods

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 collaborator grid element. Figure 2(B) depicts a primitive with one requesting grid element and n collaborating grid elements (resource). A primitive with two requesting grid elements and one collaborating grid element is depicted in Figure 2(C). It should be noted that the second and third primitives can be demonstrated using the first primitive therefore the first primitive is called the *prime primitive*. 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. For example, user 0 and resource 0 can operate based on primitive A or B, respectively. In the first scenario, resource 0 is assisted by resource 3 and in the second scenario, resource 0 is assisted by resource 0 and resource 4. We have a similar condition for user 1, in this case user 1 gets help from resource 1, resource 4, and resource 5.

IV. Implementation

In this section, we present CRGC using an extended version of Gridsim (a traditional Java-based discrete-event grid simulator) that we called CRGridSim[2][3][7]. We extended reconfigurable architecture modeling and collaborative processing support using neighborhood concept in Gridsim. we investigate a case study including different primitives where these primitives construct a backbone of reconfigurable grid architecture. In these primitives, different processing elements collaborate together 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 MIPS rating of 100. Each reconfigurable element accelerates the submitted subtasks in comparison to a GPP that this acceleration rate is represented by 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

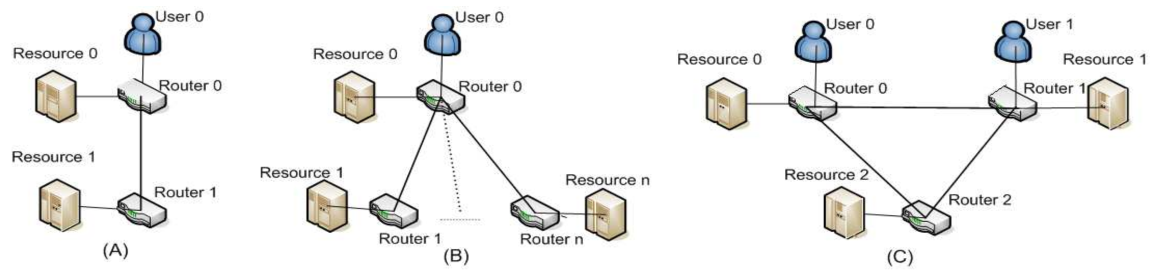

Fig. 2. Basic primitives that are utilized in neighborhood concept.

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 other parameters in same condition and analyze the related parameter. Hence, the following assumptions have been made:

- Reconfigurable elements do not support partial reconfiguration.
- There is not any 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.

V. 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 gridlets execution sequence for primitives in Figures 2(A) and 2(B) is presented in Table I. (In these tables, S.F denotes speedup factor.)

In this table, the column 'configuration' consists of two sub-columns 'main' and 'collaborator'. The column main shows main resource that receives gridlets in related primitive and column 'collaborator' shows the number and type of resource that collaborates to main resource. For example, in Figure 2 (A) the main processing element is a GPP and a collaborator resource can be a GPP or reconfigurable elements with different speedup factors. In this table, first row represents a primitive with only one GPP that does not exploit any collaboration, therefore the submitted gridlets are processed sequentially by resource 0 (GPP). The other rows represent the effect of injection of new resource as collaborator. As an example in row 3, one RE with speedup factor 2 is utilized and collaborates to process gridlets: 1, 2, 4, 5, 7 and 8, since 4 gridlets are processed by main resource and 6 gridlets are processed by resource 1 (reconfigurable element) as collaborator. Based on the last row of the Table I, the gridlet 0 is processed by main resource and other 9 remaining gridlets at the same time are processed by resources 1, 2, and 3 that included reconfigurable elements. This fact is due to the speedup of reconfigurable elements in the mentioned resources. Since each gridlet is processed 10 times faster by these resources in comparison to when it is processed by main resource. This resource requests help to process other gridlets therefore, resources 1, 2, and 3 process other grislets. The resource collaboration sequence for third primitive in Figure 2 (C) is depicted in Table II.

The primitive in Figure 2(C), a shared resource (resource 2) collaborates with two other resources (resource 0 and 1) since resource 2 receives some gridlets from resource 0 and 1. In this primitive the main and collaborator resources can be GPP or RE. In Table II, the first row represents resource 0 that receives all of gridlets without any collaboration. The fourth row represents the case that the main resource is configured as an RE with a speedup factor 2 without any collaboration. The last row of the table represents the main and collaborator resources that are reconfigurable elements with speedup factor 10. In this case, the main resource requests help from resource 2 to process the submitted gridlets. Gridlets 0, 2, 3, 5, 6, and 8 are processed by main resource and gridlets 1, 4, and 7 are processed by collaborator resource 2. The shared collaborator resource 2 is simultaneously utilized by resource 0 and 1, therefore it receives 3 gridlets from resource 0 and 3 gridlets from resource 1. The resource collaboration sequence for user 1 is similar to user 0.

The average waiting and turnaround time for the primitives in the Figures 2(A), 2(B) and 2(C) are presented in Figure 3 (A), respectively.

From Figure 3, we can observe that the growth of the number of reconfigurable elements and speedup factor decrease the average waiting time and turnaround time. In Figure 3 (A), when the number of reconfigurable elements is set to 3 and speedup factor is set to 10 the average waiting time and turnaround time decrease 6.5 and 4.32 times, respectively. In Figure 3, when the main processing elements and collabora-

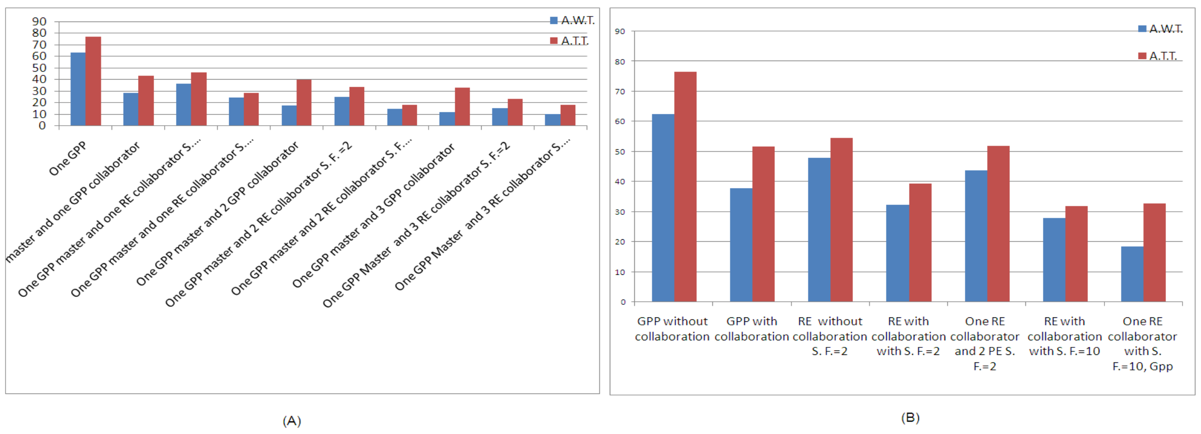

Fig. 3. (A) The average waiting and turnaround time for the primitives in the Figures 2(A) and 2(B). (B) The average waiting and turnaround time for the primitive in the Figures 2(C)

tor elements are reconfigurable elements and speedup factor are set to 10 the average waiting time and turnaround time decease 3.4 and 2.4 times, respectively. From Figure 3, we can observe that the primitive in Figure 2(C) decreases the average waiting time and turnaround time in comparison to the primitives in Figures 2(A) and 2(B). This fact is due to the following points: the number of reconfigurable elements in the primitive in Figure 2(C) is more than reconfigurable elements in the primitives in Figures 2(A) and 2(B), the reconfigurable elements in the primitives in Figures 2(A) and 2(B) are shared between two requesting reconfigurable elements.

VI. 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. The neighborhood concept was implemented using an extended version of gridsim simulation software. 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-b.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] T. C. Chiang, P. Y. Wu, and Y. M. Huang. “A limited flooding scheme for mobile ad hoc networks”. In *Proceedings of the IEEE International Conference on Wireless And Mobile Computing, Networking And Communications*, pages 473–478, August 2005.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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*, pages 1363–1375, November 2004.

[10] S. Wong and M. Ahmadi. “Reconfigurable Architectures in Collaborative Grid Computing: An Approach”. In *Proc. 2nd Int. Conf. on Networks for Grid Applications (GridNets 2008)*, October 2008.

Configuration		Gridlets									
Main	**Collaborator**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
One GPP	0	0	0	0	0	0	0	0	0	0	0
One GPP	One GPP	0	0/1	0	0/1	0	0/1	0	0/1	0	0/1
One GPP	One RE, S.F=2	0	0/1	0/1	0	0/1	0/1	0	0/1	0/1	0
One GPP	One RE, S.F=10	0	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0	0/1
One GPP	Two GPP	0	0/1	0/2	0	0/2	0/1	0	0/1	0/2	0
One GPP	Two RE, S.F=2	0	0/1	0/2	0/1	0/2	0	0/2	0/1	0/2	0/1
One GPP	Two RE, S.F=10	0	0/1	0/2	0/1	0/2	0/1	0/2	0/1	0/2	0/1
One GPP	Three GPP	0	0/3	0/1	0/2	0	0/1	0/2	0/3	0	0/2
One GPP	Three RE, S.F=2	0	0/3	0/2	0/1	0/3	0/2	0/1	0	0/2	0/1
One GPP	Three RE, S.F=10	0	0/1	0/3	0/2	0/1	0/3	0/2	0/1	0/3	0/2

TABLE I

Collaboration between resources for primitives in Figures 2(A) and 2(B) when the subtasks are submitted by user0 (The content of righthand cells shows resource numbers '0','1','2' and '3' for resource 0, resource 1, resource 2 and resource 3).

Configuration		Gridlets									
Main	**Collaborator**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
One GPP	One GPP	0	0/2	0	0	0/2	0	0	0/2	0	0
One RE, S.F=2	0	0	0	0	0	0	0	0	0	0	0
One RE, S.F=2	One RE, S.F=2	0	0/2	0	0	0/2	0	0	0/2	0	0
One GPP	One RE, S.F=2	0	0/2	0	0/2	0	0/2	0	0/2	0	0/2
One GPP	One RE, S.F=10	0	0/2	0/2	0/2	0/2	0	0/2	0/2	0/2	0
One RE, S.F=10	One RE, S.F=10	0	0/2	0	0	0/2	0	0	0/2	0	0

TABLE II

Collaboration between resources for primitive in Figure 2(C)when the subtasks are submitted by user0 (The content of righthand cells shows resource numbers).