Online Scheduling for FPGA Computation in the Cloud

Guohao Dai[1], Yi Shan[2], Fei Chen[3], Yu Wang[1], Kun Wang[4], Huazhong Yang[1]
[1]Dept. of E.E., Tsinghua National Laboratory for Information Science and Technology,
Tsinghua University, Beijing, China {dgh14@mails., yu-wang@mail., yanghz@}tsinghua.edu.cn
[2]Institute of Deep Learning (IDL), Baidu, Inc. , Beijing, China {shanyi@baidu.com}
[3]IBM China Research Lab, Beijing, China {uchen@cn.ibm.com}
[4]Microsoft ARD, China Innovation Group, Beijing, China {kuwang@microsoft.com}

***Abstract*—The popularization and application of Cloud Computing have provided a new approach for users to get computing resources in recent years. Meanwhile, due to the advantages including programmability and power-efficiency, FPGAs have been applied to custom computing in many domains. Previous work has made resources of FPGA available under the cloud environment. However, the effective usage of FPGAs in the cloud requires efficient online task scheduling: to properly assign as many tasks from different tenants as possible to the FPGAs. In this paper, we propose a *benefit-based* scheduling metric to evaluate the task assignment. Based on the metric, we accelerate task execution according to our *benefit-based* scheduling algorithms. By applying our *benefit-based* scheduling metric to a real OpenStack-based cloud environment, 60.32% computing resources are saved compared with the conventional *throughput-based* metric. Furthermore, a *Replacement-Considering* algorithm, which considers the task replacement, is proposed taking the characteristics of cloud into account. The results show that our FPGA accelerated cloud system is 1.386 times faster than using the previous algorithm.**

***Keywords*-Cloud Computing; FPGAs; Scheduling; Benefit-based Metric**

I. Introduction

With the demand of computing resources and services exploding, cloud computing is now creating broader market value, and is continuously growing rapidly [1]. Many dominant IT companies have delivered their cloud solutions. Moreover, due to the advantages including programmability and power-efficiency, FPGAs have been applied to many computational domains, achieving remarkable performance improvement.

In light of the fact that both the cloud and FPGAs can lead to excellent computing performance improvement, it is considerable to adopt FPGAs to the cloud and get better performance. However, integrating FPGAs into the cloud as accessible resources for tenants requires abstraction and management of FPGA resources is nontrivial. In order to enhance the computation capability of FPGAs under the cloud environment, effective scheduling is of vital importance.

In the cloud environment, it is not suitable to consider them using just one of the traditional metrics e.g. throughput. In this paper, we propose a novel task scheduling framework for FPGA computation in the cloud that utilizes a new metric, called *benefit-based* metric. The *benefit-based* scheduling metric represents the FPGA computation capacity as the number of virtual CPUs. Two algorithms are proposed based on this metric, targeting task assignment and replacement. The contributions of this paper can be summarized as follows:

- Proposing a *benefit-based* metric and two online *benefit-based* algorithms for scheduling computation resources on FPGAs in could computing scenario. The scheduling based on this metric could achieve 2.52 times improvement than *throughput-based* metric;
- Testing our *benefit-based* metric and algorithm on a FPGA-based heterogeneous cloud computing platform under OpenStack framework, achieving performance improvement compared to conventional scheduling methods.

II. Related Work

There are already some work that propose heterogeneous cloud computing with FPGAs and GPUs. Our previous work [2] build up an Openstack-based FPGA accelerated cloud system, by dividing FPGA chips into slots which can be configured to different accelerators using the DPR technique, providing both pre-designed and user-required accelerator logic. Another similar work is that Byma *et al.* [3] do, which also divides FPGA chips into regions, focusing on the latency of booting virtual machines (VMs). The same idea of both work is dividing a FPGA chip into several parts, each of which can be configured to required logic, without influencing other parts.

In the cloud environment, offline scheduling [4–6] for FPGA computation will not work due to the variable workloads requested by the tenants. Online scheduling for multitask on FPGA or heterogeneous platform with FPGA has also been studied a lot [7–9]. In those work, reconfigurable hardware (RH) within processors is used to accelerate computing sensitive customized instructions. However, we should assign the best fit tasks to the FPGA based on the cloud provider's view instead of single quantity. In this work, a *benefit-based* scheduling objective is used to evaluate FPGA computation in the cloud.

III. FPGA Accelerated Cloud System Design

In this section, we introduce the implementation of our online scheduling for FPGA computation in the Cloud based on our system [2]. Our implementation uses X86 physical machines running Linux and KVM-Qemu [10]. An OpenStack cloud environment is deployed to manage machines with FPGAs attached by PCIe interface. Scheduling modules are added to our system [2].

Enabling FPGAs in the cloud is shown in Figure 1. Except the modules in gray, this figure shows the typical components of an OpenStack-based cloud. A number of compute nodes, each of which is a physical machine, provide physical resources including CPUs, memory, disks and networking. A control node handles requests from tenants, schedules resources and creates VMs on selected physical machines. Tenants then get access to their virtual machines and deploy applications on them. The modules in gray are what we added to the compute nodes to enable the FPGA computation.

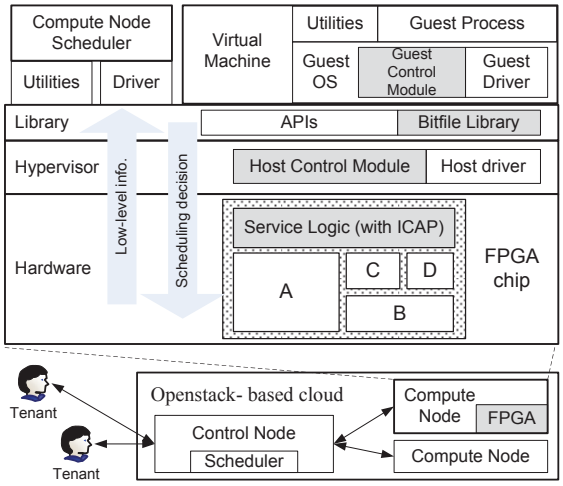

Figure 1. FPGA Framework in a Cloud [2]

To facilitate the description in Figure 1, we divide the compute nodes into 4 layers: hardware, hypervisor, library and application. In the hardware layer, an accelerator pool (AP) abstraction is proposed to use FPGAs in the cloud. In the AP abstraction, each FPGA chip has several pre-defined accelerator slots, e.g. slots A, B, C and D shown in Figure 1. Each slot can only host an accelerator with compatible resource requirements and interface design. Instead of requesting programmable resources, a tenant directly requests accelerator functions. The cloud provides a list of pre-defined accelerators, handles tenant requests and configures accelerators into idle slots. Tenant can also submit their own designs to the cloud owner. We introduce a service logic (SL) module to configure and manage the accelerator slots. DPR mechanism of modern FPGAs is used here to reconfigure the slots through ICAP interface. To schedule tasks under the cloud environment, scheduling logic is added to [2], including the Compute Node Scheduler (collecting low-level information from hardware) and the Control Node Scheduler (analyzing the low-level information and making decisions).

With hardware/software co-design in these four layers, the FPGA computation is enabled in the OpenStack-based cloud environment. Meanwhile, the low-level real-time information can be easily exposed to the scheduler (both Control Node Scheduler and Compute Node Scheduler), which makes the decision of online scheduling.

IV. Online Scheduling Methodology

In this section, we firstly introduce the FPGA accelerated cloud scheduling model. And then a benifit-based scheduling metric is proposed, which can evaluate the profits of FPGA acceleration in the cloud computing. Finally, the scheduling algorithm based on this metric is introduced.

A. Scheduling Model

Based on our FPGA accelerated cloud system design, a scheduling model is proposed in Figure 2. In the FPGA accelerated cloud environment, computing nodes are composed of computing resources including CPUs and FPGAs, storage, networking, etc. Also, there is a Computing Node Scheduler in each computing node to get runtime information and send the information to the Control Node Scheduler. Each FPGA chip is divided into several slots, which can be configured to accelerators designed for different applications. When tasks are executed under the cloud environment, the cloud provider should assign each task to proper computing device, FPGAs (actually FPGA slots in our design) or CPUs, according to a well-designed cloud scheduling mechanism. That is to say, the scheduling objects are tasks running under the cloud environment and computing devices. As shown in Figure 2. By getting runtime information from the Computing Node Scheduler, Control Node Scheduler allocates tasks to suitable computing devices (dyed gray in Figure 2) after scheduling.

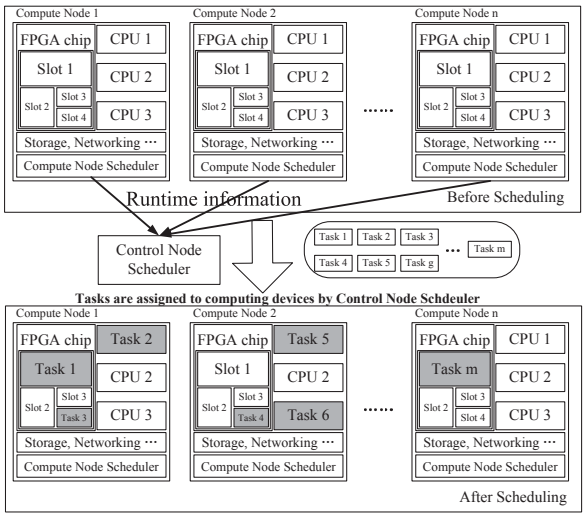

Figure 2. Scheduling Model of FPGA Accelerated Cloud System

B. Benefit-based Scheduling Metric

The FPGA accelerated cloud system is aimed at improving the computing capacity of the cloud. However, not all of the tasks will get profits when being executed on FPGAs, and the change of workloads of tasks may also cause performance declined. In the cloud provider's point of view, the tasks with the most profits on FPGA should be assigned to the FPGA. For this reason, how to correctly evaluate the profits is important. In previous work, FPGA acceleration or heterogeneous computing usually adopts *throughput* as a performance metric [5]. However, some applications may achieve large but similar *throughput* on CPUs and FPGAs. If we use FPGA to accelerate this kind of applications, although the *throughput* will be good, the profits will be limited.

To solve this problem, we propose a metric that describes the computing capacity of FPGA equivalent to a certain number of standard virtual CPUs (vCPUs). So, if running a task can achieve *n* times speedup on FPGA compared with running it on a standard vCPU, then the computing capacity of the FPGA area occupied by this task is equal to *n* vCPUs. We could use an equation as the summation of all the tasks speedup compared with a vCPU to describe the profit of the whole FPGA chip. We name the metric, *benefit*. In Equation (1), t_i represents the task to be executed.

$$Benefit_{FPGA} = \#vCPU = \sum_{\forall t_i \in task} Speedup_{t_i} \qquad (1)$$

Usually, the cloud tenants will pay for the the cloud provides according to the number of vCPUs they use. Using this metric, the FPGA platform could be evaluated by money cloud providers earn. In the cloud computing environment, we should put tasks that could earn most *benefit* and cost least resources on FPGA to make the whole *benefit* maximal.

C. Scheduling Algorithms

With the module, an effective online scheduling algorithm is proposed based on *benefit-based* scheduling metric. Similar to other optimization problems, we have some constraints, such as area, IO, and bandwidth of the off-chip memory for FPGAs.

Algorithm 1 Online scheduling for new task assignment

Input:
new tasks sequence t_i;
existing accelerators acc_j and slots $slot_k$;
performance estimation $model$;
Output:
t_i and $slot_k$ assignments;
1: **for** t_i belong to a time period **do**
2: $perf_i$ = Estimate(t_i, acc_j, $model$);
3: **end for**
4: $task_{candidate}$ = Max($perf_i$);
5: **for** $slot_k$ **do**
6: **if** Satisfy_constrain($slot_k$, $task_{candidate}$) **then**
7: Assign $task_{candidate}$ to $slot_k$;
8: **end if**
9: **end for**

The Algorithm 1 focuses on assignment of a new task. Considering there are several new tasks to be assigned to FPGA in a time period, the best fit one should be chosen. We will use the model proposed in Section IV-B to evaluate the *benefit* of each task, both new task t_i and existing tasks acc_j. The overall *benefit* of all the tasks, $perf_i$, when assign a new task t_i to FPGA, will be achieved by $Estimate$ (Line 2, Algorithm 1) function, according to Equation (1). The task causing most overall *benefit* will be checked if it satisfies the constraints. If satisfied, this task will be assigned to FPGA. If not satisfied, the task with the second *benefit* will be checked. This kind of scheduling will be executed every interval time to fully utilize the computation capacity of FPGAs.

Algorithm 2 Online scheduling *Replacement-Considering* algorithm

Input:
new tasks sequence t_i;
existing accelerators acc_j and slots $slot_k$;
performance estimation $model$;
Output:
t_i and $slot_k$ assignments;
1: **for** t_i belong to a time period **do**
2: $perf_i$ = Estimate(t_i, acc_j, $model$);
3: **end for**
4: $task_{candidate}$ = Max($perf_i$);
5: **for** $slot_k$ **do**
6: **if** Satisfy_constrain($slot_k$, $task_{candidate}$) **then**
7: Assign $task_{candidate}$ to $slot_k$;
8: **else if** $perf_i - perf_{least} > reconfig\_cost$ **then**
9: Migrate $task_{least}$ to CPU;
10: Assign $task_{candidate}$ to $slot_{least}$;
11: **end if**
12: **end for**

In algorithm 1, we only think about the scheduling for new tasks. Once the task is running on FPGA, it will not be terminated or moved to other platforms until all the workloads for this task are done. However, some tasks may not be suitable for running on FPGA just in such cases:

- The workload of one task changes during runtime causing its *benefit* declined.
- New tasks with more *benefit* come, calling for limited FPGA resources.

So it is reasonable to think about replacing this kind of tasks online. Algorithm 2 cares about both assignment and replacement of new tasks. The least *benefit* task, $task_{least}$, is always recorded at runtime. When new tasks come, in the event that there is no spare FPGA slot or the spare slots can not satisfy the constraints of new tasks, the replacement will be considered. If the *benefit* achieved by replacement is more than the reconfiguration cost, then the task with the least *benefit* will be migrated to CPU for execution and the new task will be assigned to FPGA.

V. Experimental Results

The evaluation is conducted on an IBM X3650M server with one Intel Xeon X5690 processor. The processor provides 6 cores, 12 hyper threads, and works at a frequency of 3.47GHz. 16GB DDR3 memory is installed in the server and memory operates at 1067MHz. The OS on the server is a Fedora Core 12 with Linux kernel 2.6.31.5. The Qemu used for VM is a modified qemu-kvm-1.0. We define the *standard virtual CPU* that runs at 1 core with 2GB memory.

A PCIe card with one Xilinx Kintex-7 XC7K325T FPGA is used as our FPGA subsystem, and the PCIe interface is configured as Gen2, 8 lanes with a peak bandwidth of 5 GB/s, including 8b/10b coding overhead and PCIe transaction overhead. Our system operates at 100MHz. In an instance with an 8-slot task queue and a 4-port switch, the resource usage of our SL is 3,579 Slices and 1,268Kb BRAM, which are only 7.02% and 3.64% of our FPGA chip. Adding one more switch port will use another 290 Slices and 8Kb BRAM. The 3 of the 4 slots cost the same area and the other one costs more to hold the complex accelerators.

From various application domains, six algorithms are selected to evaluate our prototype, i.e. AES, SHA, Stereo Matching (Stereo), Gaussian mixture model (GMM), Optical Flow (OF), and Matrix-Vector Multiply (MVM). The accelerators have different requirements on computation and IO resources. Among those six, only AES is preemptable, permitting a job to be interrupted. All accelerators shares one DMA engine, of which the peak bandwidth is 1.28GB/s. The $speedup_{ideal}$ is achieved by executing the accelerator only on both standard virtual CPU and FPGA.

We assign proper tasks to different accelerators to make them have equal ideal finish time, 1 second, when running separately on FPGA. Tasks are generated randomly each interval time. The parameter $alpha$, denotes the ratio of ideal finish time and interval of each two tasks, which directly affects the frequency of workloads. By varying $alpha$, workloads of different situation can be simulated. The performance is normalized as the number of standard virtual CPUs. The higher the performance of FPGA computation is, the more CPU cycles are saved. Because we focus on the online scheduling for FPGAs, only the performance of

FPGAs is counted here. Two experiments would be shown below to verify our metric and algorithms.

The first experiment is designed to evaluate our proposed metric for FPGA computation in the cloud. The *throughput-based* method arranges the task, which can perform highest *throughput*, to FPGA computation. However, some tasks may also have high *throughput* on CPU and the implementation using FPGA may not provide much more computation capacity, e.g. AES. As shown in Figure 3, the scheduling using *benefit-based* metric is 2.52 times faster than the one using *throughput-based* metric. On the other hand, we can save 60.3% cpu resources to achieve the same performance. When the workload is light, the computation capacity of FPGA is not fully utilized and the performance will increase as the workloads increase for both method. For the *benefit-based* scheduling, as the workloads further increase, the tasks with more *benefit* will be assigned to the FPGA until the computation capacity is fully utilized.

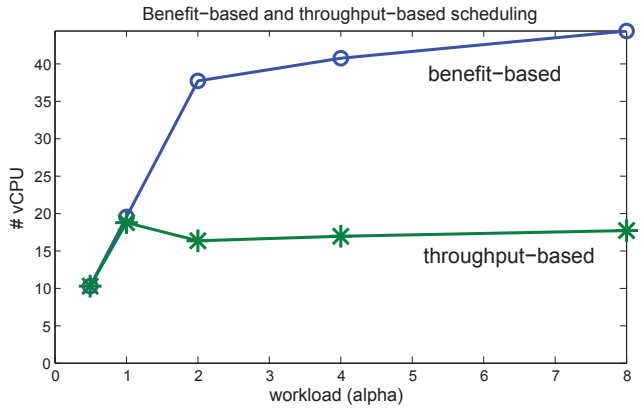

Figure 3. *Benefit-based* and *Throughput-based* Scheduling Comparison

The second experiment is designed to evaluate the scheduling algorithm with or without replacement. Using the scheduling Algorithm 1 without replacement means that the slots will be configured as fixed accelerators in a certain time until all the tasks for these accelerators are done. In this way, the reconfiguration cost is saved but there may be some tasks have low utilization or severe interference along the time. The Algorithm 2 with replacement will replace this kind of accelerator by dynamic partial reconfiguration. Figure 4 shows on average 38.60% improvement is achieved by involving replacement scheme. When the workload is light, the implementation with replacement will finish all the tasks, but the one without replacement will reject some tasks according to the fixed configuration. As the workloads increase, the later one will have more tasks when making decision and give similar assignment as the former one.

VI. Conclusion

Heterogeneous cloud computing with FPGAs may be a promising solution to achieve power efficiency. In contrast to traditional private heterogeneous computing, the optimization object of FPGA computation has changed. In this paper we propose a *benefit-based* scheduling metric which could assign the tasks with most *benefit* to FPGA. Based on this metric, two scheduling algorithms of task assignment and replacement are introduced. To demonstrate the effectiveness, we test our method on a prototype of FPGA-based

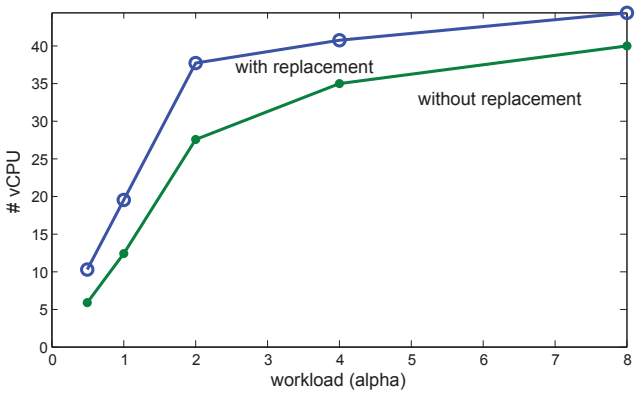

Figure 4. Scheduling Comparison with and without Replacement

heterogeneous cloud computing under OpenStack framework. The experimental results show that algorithm based on the proposed metric is 2.52 times faster than the result of the *throughput-based* one.

In this work, we focus on the online scheduling for FPGA in one physical machine under the cloud environment. The scheduling for multiple heterogeneous machines is covered by original scheduler in the OpenStack framework. For the reason that the scaling maybe one issue, we will investigate the scalability of this scheduling problem in our future work.

VII. Ackonwledgement

This work was supported by Huawei Innovation Research Program, IBM Research China University Relationship Program, 973 project 2013CB329000, National Science and Technology Major Project (2011ZX03003-003-01), National Natural Science Foundation of China (No.61373026), and Tsinghua University Initiative Scientific Research Program.

References

[1] "Global Cloud Computing Market: $270 Billion By 2020," Website, 2012, http://www.cloudcomputingmarket.com/.

[2] Fei Chen *et al*, "Enabling fpgas in the cloud," in *CF*, no. 3, 2014.

[3] Stuart Byma *et al*, "Fpgas in the cloud: Booting virtualized hardware accelerators with openstack," in *FCCM*, 2014, pp. 109–116.

[4] Matthew A. Watkins *et al*, "Remap: A reconfigurable architecture for chip multiprocessors," *IEEE Micro*, vol. 31, no. 1, pp. 65–77, 2011.

[5] Garcia, Philip *et al*, "Kernel sharing on reconfigurable multiprocessor systems," in *FPT*, 2008, pp. 225–232.

[6] Chen Liang *et al*, "Shared reconfigurable fabric for multi-core customization," in *DAC*, 2011, pp. 830–835.

[7] Shafique, Muhammad *et al*, "Minority-game-based resource allocation for run-time reconfigurable multi-core processors," in *DATE*, 2011, pp. 1–6.

[8] Chen Liang *et al*, "Online scheduling for multi-core shared reconfigurable fabric," in *DATE*, 2012, pp. 582–585.

[9] Rupnow K. *et al*, "Block, drop or roll(back): Alternative preemption methods for rh multi-tasking," in *FCCM*, 2009, pp. 63–70.

[10] "Kernel Based Virtual Machine," Website, http://www.linux-kvm.org/page/Main_Page.