An Open-source Framework for Integrating Heterogeneous Resources in Private Clouds*

Julio Proaño, Carmen Carrión and Blanca Caminero
Albacete Research Institute of Informatics (I3A), University of Castilla-La Mancha, Albacete, Spain

Keywords: Cloud Computing, FPGA, Hardware Acceleration, Energy Efficiency.

Abstract: Heterogeneous hardware acceleration architectures are becoming an important tool for achieving high performance in cloud computing environments. Both FPGAs and GPUs provide huge computing capabilities over large amounts of data. At the same time, energy efficiency is a big concern in nowadays computing systems. In this paper we propose a novel architecture aimed at integrating hardware accelerators into a Cloud Computing infrastructure, and making them available as a service. The proposal harnesses advances in FPGA dynamic reconfiguration and efficient virtualization technology to accelerate the execution of certain types of tasks. In particular, applications that can be described as Big Data would greatly benefit from the proposed architecture.

1 INTRODUCTION

The popularity of Cloud computing architectures is growing over the Information Technology (IT) services. Cloud computing is based on an on-demand approach to provide to the user different computing capabilities "as-a-service". Three Cloud usage models can be categorized in this computational paradigm, namely Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS), depending on the type of capability offered to the Cloud user which in turn is related to the degree of control over the virtual resource that support the application.These cloud models may be offered in a public, private or hybrid networks.

Nowadays, the main problem to infrastructure providers are concerned about the relationship between performance and cost. One of the solutions adopted has been adding specialized processors that can be used to speed up specific processing tasks. The most well-known accelerators are Field Programmable Gate Arrays (FPGAs) and Graphics Processors (GPUs) but paradoxically this use also directly related to the energy consumption of the system (Buyya et al., 2013), and cost.

Focusing on FPGAs, which have large resources of logic gates and RAM blocks to implement complex digital computations. This hardware devices are an order of magnitude faster for non-floating point operations than state of the art CPUs. Furthermore, FPGAs provides very good performance and power efficiency in processing integer, character, binary or fixed point data. And also deliver competitive performance for complex floating-point operations such as exponentials or logarithms. Additionally, the FPGA component can be reconfigured any number of times for new applications, making it possible to utilize the heterogeneous computer system for a wide range of tasks (Intel, 2013). In particular, many life science (LS) applications, such as genomics, generate immense data sets that require a vast amount of computing resources to be processed. FPGAs provide an intrinsic degree of parallelism amenable of being exploited by these applications. Moreover, LS applications change very quickly, thus making impractical the use of application specific integrated circuits (ASICs). To this end, the use of FPGAs seems a promising option. Finally, FPGAs exhibit a high computation/power consumption ratio, which turns them into an interesting option to lower operational expenses as well as minimize the carbon footprint of the data center.

Thus, the main objective of this work is to achieve an efficient low power architecture able to provide on-demand access to a heterogeneous cloud infrastructure, which can be especially exploited by big-data applications. FPGA-based accelerators are key computational resources in this cloud infrastructure. The proposal is aimed at enabling the use of FPGA-

*This work was supported by the Spanish Government under Grant TIN2012-38341-C04-04 and by Ecuadorian Government under the SENESCYT Scholarships Project.

Proaño J., Carrión C. and Caminero B..
An Open-Source Framework for Integrating Heterogeneous Resources in Private Clouds.
DOI: 10.5220/0004936601290134
In *Proceedings of the 4th International Conference on Cloud Computing and Services Science* (CLOSER-2014), pages 129-134
ISBN: 978-989-758-019-2

based accelerators without dealing with the complexity of hardware design. Furthermore, the vision is to provide users with a repository of bitstream[2] files of common applications, which can be used as a service but also as building blocks to develop more complex applications. It must be noted that Amazon AWS already offers GPU instances (AWS, 2013), and FPGA-based accelerators are offered by Nimbix within its JARVICE service. But, to the best of our knowledge, this is the first open-source proposal of an IaaS architecture in which the FPGA accelerators (among others) are considered as computational virtual resources.

The rest of the paper is organized as follows. In Section 2 some related works are reviewed. Then, the architecture of our proposal is described in Section 3, while some implementation details are given in Section 4. Finally, Section 5 concludes this work, and outlines some guidelines for future work.

2 RELATED WORK

Hardware FPGA devices used as co-processors can offer significant improvement to many applications (Guneysu et al., 2008) (Dondo et al., 2013). So, there are several attempts to integrate FPGAs into traditional software environments. Many solutions are provided by the industry, such as PicoComputing, Convey and Xillybus. These products connect software to FPGAs via a proprietary interface with their own languages and development environments. The Mitrion-C Open Bio Project (Mitrion-C: The Open Bio Project, 2013) tries to accelerate key bioinformatics applications by porting critical parts to be run on FPGAs. Open source proposals such as (Jacobsen and Kastner, 2013) provide a framework that uses FPGAs as an acceleration platform. It must be highlighted that in this work FPGAs are connected to PCs through PCIe. However, these examples lack the framework for developing parallel and distributed applications on clusters with multiple nodes.

There are systems where FPGA devices are used as the only computing elements forming the cluster (Guneysu et al., 2008). However, not all applications can be accelerated effectively using FPGAs. For example, the high clock rate and large numbers of floating point units in GPUs make them good candidates for hardware accelerators.

The use of GPUs as computational resources on cloud infrastructures has emerged in the last years (Suneja et al., 2011). Hence, nowadays several commercial companies offer GPU service to their clients (Nvidia, 2013). For example, Amazon EC2 (AWS, 2013) supports general purpose GPU workflows via CUDA and OpenCL with NVIDIA Tesla Fermi GPUs. The use of FPGAs in cloud systems has not been so successful until now. At this point it must be highlighted that Nimbix (Nimbix, 2013) offers a commercial cloud processing system with a variety of accelerated platforms. Recently, the company has launched JARVICE, a PaaS platform that includes the availability of GPUs, DSP and FPGAs for an application catalog and an API or command-line access to submit jobs.

Moreover, there are some efforts that combine accelerators (FPGAs and GPUs) and CPUs in cluster nodes such as Axel (Tsoi and Luk, 2010). Axel combines heterogeneous nodes with the MapReduce programming model resulting in a low-cost high-performance computing platform. The authors in this work highlight that the largest drawback of FPGA design is the difficulty of implementation compared with CUDA programming time.

Also, recent researches focus on dynamically reconfigurable FPGAs. Reconfigurable FPGAs are devices where software logic can be reprogrammed to implement specific functionalities on tunable hardware. Hence, in (Dondo et al., 2013) a reconfiguration service is presented. This service is based on an efficient mechanism for managing the partial reconfiguration of FPGAs, so that a large reduction in partial reconfiguration time is obtained. Providing this kind of service is a must in order to attach FPGA resources in cloud infrastructures. Nevertheless, more in-depth research is required to provide a virtual machine monitor able to manage the reconfigurable capabilities on FPGA devices.

The closest research to our vision is the HARNESS (Hardware- and Network-Enhanced Software Systems for Cloud Computing) European FP7 project (HARNESS FP7 project, 2013). The objective of this project is to develop an enhanced PaaS with access to a variety of computational, communication, and storage resources. The application consists of a set of components that can have multiple implementations. Then, an application can be deployed in many different ways over the resources obtaining different cost, performance and usage characteristics. However, for the time being no framework proposal nor proof-of concept has been published yet.

[2] A *bitstream* is the data file to be loaded into a FPGA.

3 ARCHITECTURE PROPOSAL

In this section we describe an heterogeneous architecture for building on-demand infrastructure services on Clouds systems. The architecture attempts to address the challenges of deploying an infrastructure as a service (IaaS) on a private datacenter in which FPGA devices are considered as computational virtual resources. The virtual FPGAs are computing resources managed by the system together with the virtual machines running on CPUs. To be more precise, FPGA devices are used as accelerators for some codes to be run. In the remainder of this section, we describe the main characteristics of the proposal.

First of all, some arrangements must be fixed to orchestrate the physical computational components (CPUs, FPGAs and GPUs) of the proposed infrastructure. This proposal is based on uniform nodes composed of heterogeneous resources, as shown in Figure 1. Each node is composed of at least one CPU, and some extra COTS components can be included, which means that a FPGA on its own cannot be a compute node and every node must be provided with a CPU resource. The CPU of a node accesses and controls the task assigned to the accelerators. Therefore, a physical connection is required between the VM running on the CPU and the COTS.
Secondly, in order to support the heterogeneous cloud IaaS, some extra functionalities must be provided by the Virtual Machine Manager (VMM). The VMM is in charge of the deployment of VMs, and monitoring and controlling their state. Hence, since a mechanism is required to manage the heterogeneous architecture, our proposal makes use of a Hardware Accelerator Manager (HAM). HAM is an independent open-source component that makes use of the functionality of the VMM and extends it by providing the management of accelerators, namely, FPGAs and GPUs. When a user requests the use of the cloud infrastructure, HAM is in charge of deploying the cluster infrastructure, for example, for a Hadoop application (Apache, 2013), to run on it. This also includes the proper configuration of the accelerators so that they can be effectively exploited by the VMs. So, the system has different states which include the set-up state and the running one. To be more precise, in the mentioned case, during the set-up state, HAM deploys the master VM together with the worker VMs required to attend the request of the user on time.

As the integration of GPUs into a virtual infraestucture has already been addressed and even ofered by some Cloud providers (i.e.,Amazon), we will focus on the description on the integration of FPGAs.

3.1 Overall Description

In our system, we propose that the users use FPGAs to execute certain tasks as hardware acceleration modules integrating FPGAs as part of a private cloud. In general, the Hardware Acceleration Manager controls the communication and full processing of preparing and deploying hardware / software elements.

A general view of the proposed architecture is shown in Figure 1. Our system is composed of two elements, namely a **Hardware Accelerator Manager (HAM)** and a **Catalog**. On one hand, HAM is responsible of receiving requests from clients and deploying virtual hardware/software acceleration elements.

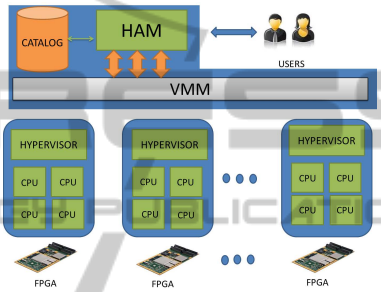

Figure 1: Arquitecture Overview.

HAM is supported by the Catalog, which consists of a set of files where all the meta-data and information about software, hardware, bitstream files, hosts, IP address, FPGA resources are stored, that is, the information stored in the Catalog consists of the state of the resources and the code of all the tasks loaded in the system up to now. Note that the repository of bitstreams is not fixed and can be increased over time. HAM will access this information when needed.

HAM is implemented as a software component which uses the functionalities of a Virtual Machine Manager (VMM) to monitor and deploy virtual infrastructures. It also harnesses the Intel VT-d technology support within the hypervisor to communicate hardware and software through the PCIe host port.

By using HAM, users will be able to execute jobs that can be partitioned into simple tasks, which can be implemented as hardware modules into an FPGA device, and other tasks which will be deployed in virtual CPUs. To illustrate the use of our architecture, let us assume that a user wants to run an application based on matrix-vector multiplication. Firstly, HAM processes the request from the user and tries to supply the virtual infrastructure required to run the matrix multiplication. To be more precise, HAM searches for efficient resources, including virtual machines with FPGA accelerators. Then, the system provides the user with the list of available resources, including the

corresponding FPGA IDs. HAM is responsible for preparing and deploying the virtual environment to process the request. Focusing on the FPGA accelerator, deploying the virtual infrastructure consists on providing access to the resource by using the Intel VT-d technology and loading the multiplication bitstream file in the FPGA. The multiplication bitstream file can be provided by the user or loaded from the Catalog. When the virtual infrastructure is successfully created, the system sends the data (matrix to be multiplied) and waits for the results. Finally, the results are sent back to the user. Also, the multiplication bitstream file can be stored in the Catalog so that it is available for future use.

HAM uses four modules (referred to as Controllers) to provide full deployment of virtual acceleration elements in the cloud environment. HAM Controllers will be described next.

4 IMPLEMENTATION DETAILS

In this section, we discuss the details of each part of the HAM architecture focusing on the integration of FPGAs as accelerators. The key software controllers in HAM are depicted in Figure 2, and are the following:

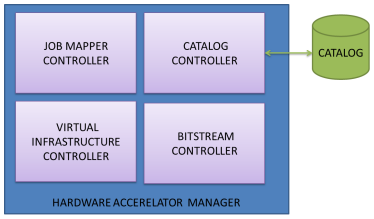

Figure 2: Hardware Accelerator Manager (HAM) Controllers.

- **The Catalog Controller (CC)** keeps the dataset of the **Catalog Module** updated.
- **The Bitstream Controller (BC)** is in charge of programming the FPGA, so that a given task can be executed on it.
- **The Virtual Infrastructure Controller (VIC)** deploys the virtual infrastructure required by an application.
- **The Job Mapper Controller (JMC)** interacts with the user and controls the execution of the tasks in the resource of the system.

When a request arrives, the Job Mapper Controller analyzes the state of the system by consulting the Catalog and supplies different options to the user. Then, the user will select the best computational resources

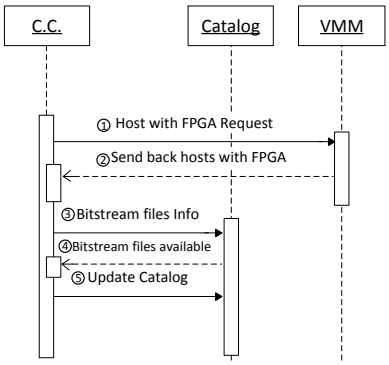

Figure 3: Catalog Controller.

suitable for running his/her application and the Virtual Infrastructure Controller will go ahead with the set-up of the infrastructure. This controller will be in charge of the deployment of the virtual resources and will interact with the Bitstream Controller each time an FPGA is required as computational resource. After the set-up, the tasks of the application will be executed on the provided resources. At the end of the execution, the Job Mapper Controller will collect the results and will send them to the user.

A more detailed description of the functionality and the interaction of the controllers with the environment (the Catalog, the VMM, FPGAs,...) is given next. Note that to get a clear explanation only one VM and one FPGA accelerator are considered in the following description, in spite of the fact that HAM is able to manage all the datacenter resources.

The Catalog Controller (CC) charge of monitoring available hardware/software resources and updating the Catalog. Figure 3 shows a sequence diagram illustrating the functionality of this controller. Requests to the Virtual Machine Manager (step 1) are used to collect all the information about FPGAs, bitstream files and Virtual Machines. This information is then collected and transferred to the Catalog (step 3). Also, the Catalog Controller is used to register new bitstream files corresponding to new computational resources being accelerated (step 5).

The Bitstream Controller (BC) is responsible for programming FPGAs with bitstream files, as shown Figure in 4. The mechanism used to program FPGAs consists of transferring the bitstream file to a host with an associated FPGA (which is known through a query to the Catalog, steps 1 and 2) via `ssh` (step 3). Each host with an associate FPGA contains a FPGA driver, which is used to program the FPGA device (step 4).

The Virtual Infrastructure Controller (VIC) is responsible for preparing and deploying the virtual environment which will be used to communicate the software (i.e., the application running in the virtual

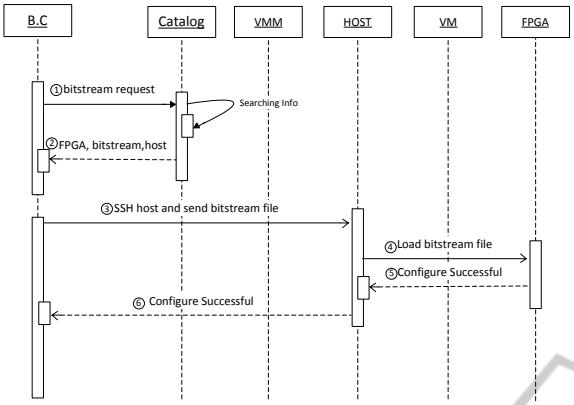

Figure 4: Bitstream Controller.

CPU) with the hardware accelerator. For example, for a Hadoop application, a cluster composed of several VMs will be deployed. Figure 5 depicts how this controller works. First of all, the Virtual Infrastructure Controller uses the Catalog Controller to collect all the necessary information to deploy a virtual environment (step 1). For example, a template which contains a definition of a Virtual Machine, the host name which contains the FPGA device associated, FPGA Id device and finally the PCIe port information.

The Virtual Machine Manager is used then to create a virtual machine (step 3) and then, when the virtual machine has been successfully created (step 5), the controller uses an "attach" functionality (steps 6 and 7) to create the access between the virtual machine (software) and the FPGA (hardware). Finally, the full system (composed of both the software and the hardware) is started up and ready to use (step 9).

It is important to clarify that the Intel VT-d technology (Intel, 2013) is used to communicate virtual machines and FPGA hardware because, it allows an important improvement on the performance achieved by accessing I/O devices in virtualized environment.

Finally, **The Job Mapper Controller (JMC** is responsible for the management of data sharing, monitoring the execution of jobs and sending files and reports to the clients. Additionally, it also makes use of an efficient scheduling algorithm to assign task to resources. This algorithm must have into account the requirements of the client applications, the available hardware accelerator resources and the state of the CPUs. The main characteristic that enable job acceleration is that data should be allowed to be partitioned among the available computing resources (i.e., CPU and FPGA) and be processed independently in order to improve performance. When the Job Mapper Controller (JMC) receives a job from a client, it adds the job to the global scheduler queue. Then, the Catalog Controller (CC) is consulted to discover the FPGA

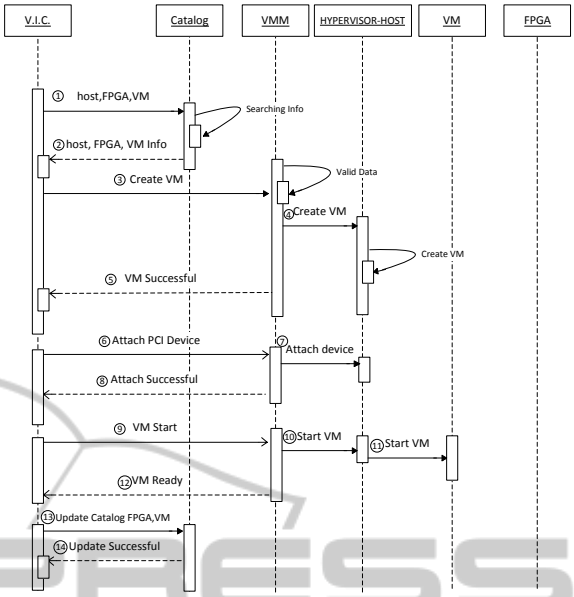

Figure 5: Virtual Infrastructure Controller.

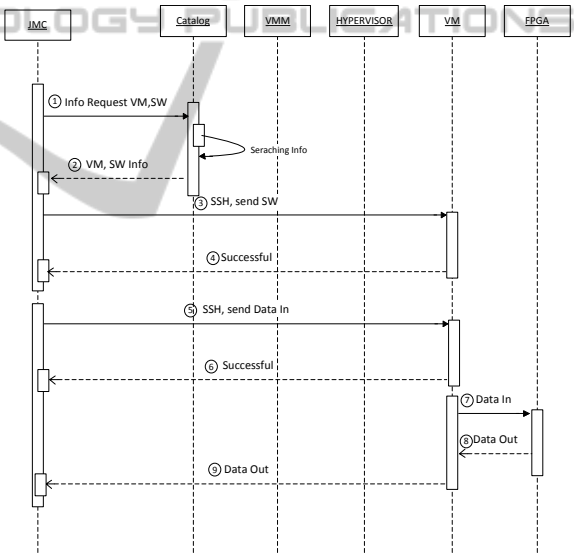

Figure 6: Job Mapper Controller.

resources available on where the data related to the job reside, to split the job into tasks, and to distribute these tasks according to the resources and data availability at each FPGA. These process is depicted in figure 6. The Catalog Controller (CC) is used by the Job Mapper Controller to retrieve information relative to previously deployed software application and virtual machine identification, which users can use (step 1). The Catalog sends back the necessary information identification for preparing software acceleration elements. Then, the job is transferred to the allocated Virtual Machine (step 3) through ssh. The application (bitstream file) programmed into the FPGA is de-

signed to receive a stream of data as ”input” as shown in (step 5) the (Data-In). Next, the FPGA processes data and, when the results are ready, the FPGA sends back a stream containing data results as ”output” (step 8). Finally, the JMC sends the results to the user.

5 CONCLUSIONS AND FUTURE WORK

We envision a heterogeneous cloud infrastructure architecture in which accelerators such as FPGAs and GPUs are used to reduce the power consumption of the datacenter. This will be possible when they get significantly easier to interact with. Hence, in this work an open-source IaaS architecture has been described focused on providing FPGA accelerators as easy-to-use virtual resources.

The future work will mainly consist on implementing the proposed architecture. As a matter of fact, a preliminary prototype built on top of the OpenNebula (Moreno-Vozmediano et al., 2012) Virtual Machine Manager is near to completion. Next, the focus will be placed on the problem of effective resource scheduling, addressing the issue of FPGA sharing among different virtual machines running in the same node.

Also, the feasibility of exploiting standard Single Root-I0 Virtualization (SR-IOV) (Intel LAN Access Division, 2011), which allows a PCIe device to appear as multiple virtual devices that can be accessed by a guest making use of VM passthrough will also be explored.

REFERENCES

Apache (Last access: 18th December, 2013). Hadoop. Available at: http://hadoop.apache.org/.

AWS (Last access: 10th December, 2013). Amazon EC2 Instances. Web page at http://aws.amazon.com/ec2/instance-types/.

Buyya, R., Vecchiola, C., and Selvi, S. T. (2013). *Mastering Cloud Computing: Foundations and Applications Programming*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition.

Dondo, J., Barba, J., Rincon, F., Moya, F., and Lopez, J. (2013). Dynamic objects: Supporting fast and easy run-time reconfiguration in FPGAs. *Journal of Systems Architecture - Embedded Systems Design*, 59(1):1–15.

Guneysu, T., Kasper, T., Novotny, M., Paar, C., and Rupp, A. (2008). Cryptanalysis with COPACABANA. 57:1498–1513.

HARNESS FP7 project (Last access: 12th December, 2013). HARNESS: Hardware- and Network-Enhanced Software Systems for Cloud Computing. Web page at http://www.harness-project.eu/.

Intel (2013). Intel Virtualization Technology of Directed I/O, Architecture Specification, Rev. 2.2. Available at http://www.intel.com.

Intel (Last access: 13rd December, 2013). Heterogeneous Computing in the Cloud: Crunching Big Data and Democratizing HPC Access for the Life Sciences. Web page at http://www.intel.com/.

Intel LAN Access Division (2011). PCI-SIG SR-IOV Primer: An Introduction to SR-IOV Technology. Available at http://www.intel.com/.

Jacobsen, M. and Kastner, R. (2013). RIFFA 2.0: A reusable integration framework for FPGA accelerators. In *23rd International Conference on Field programmable Logic and Applications, FPL 2013*, pages 1–8.

Mitrion-C: The Open Bio Project (Last access: 13rd May, 2013). Web page at http://mitc-openbio.sourceforge.net/.

Moreno-Vozmediano, R., Montero, R. S., and Llorente, I. M. (2012). IaaS Cloud Architecture: From Virtualized Datacenters to Federated Cloud Infrastructures. *IEEE Computer*, 45.

Nimbix (Last access: 13rd May, 2013). Nimbix Accelerated Compute Cloud (NACC). Web page at http://www.nimbix.net/.

Nvidia (Last access: 14th December, 2013). GPU Cloud Computing . Available at: http://www.nvidia.com/object/gpu-cloud-computing-services.html.

Suneja, S., Baron, E., Lara, E., and Johnson, R. (2011). Accelerating the Cloud with Heterogeneous Computing. In *Proceedings of the 3rd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'11, pages 23–23, Berkeley, CA, USA. USENIX Association.

Tsoi, K. H. and Luk, W. (2010). Axel: A Heterogeneous Cluster with FPGAs and GPUs. In *Eighteenth ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA'10)*.