HARNESS Project: Managing Heterogeneous Computing Resources for a Cloud Platform*

J.G.F. Coutinho[1], Oliver Pell[2], E. O'Neill[3], P. Sanders[2], J. McGlone[3], P. Grigoras[1], W. Luk[1], and C. Ragusa[3]

[1] Imperial College London, UK
[2] Maxeler Technologies, UK
[3] SAP HANA Cloud Computing, Systems Engineering, Belfast, UK
jgfc@doc.ic.ac.uk

Abstract. Most cloud service offerings are based on homogeneous commodity resources, such as large numbers of inexpensive machines interconnected by off-the-shelf networking equipment and disk drives, to provide low-cost application hosting. However, cloud service providers have reached a limit in satisfying performance and cost requirements for important classes of applications, such as geo-exploration and real-time business analytics. The HARNESS project aims to fill this gap by developing architectural principles that enable the next generation cloud platforms to incorporate heterogeneous technologies such as reconfigurable Dataflow Engines (DFEs), programmable routers, and SSDs, and provide as a result vastly increased performance, reduced energy consumption, and lower cost profiles. In this paper we focus on three challenges for supporting heterogeneous computing resources in the context of a cloud platform, namely: (1) cross-optimisation of heterogeneous computing resources, (2) resource virtualisation and (3) programming heterogeneous platforms.

1 Overview

The current approach for building data centres is to assemble large numbers of relatively inexpensive personal computers, interconnected by standard routers and supported by stock disk drives. This model for cloud computing leverages commodity computation, communication, and storage to provide low-cost application hosting. The efficacy of this platform depends on the providers' ability to satisfy a broad range of application needs while at the same time capitalising on infrastructure investments by making maximal use of the platform's resources. Two key concepts related to cloud data centres are managed *multitenancy* and *elasticity* [6]. To support multitenancy, the provider must accommodate and reconcile the resource needs of several applications simultaneously, while elasticity allows an application to run on a platform using a pool of resources that can

* The HARNESS Project is supported by the European Commission Seventh Framework Programme, grant agreement no 318521 http://www.harness-project.eu

D. Goehringer et al. (Eds.): ARC 2014, LNCS 8405, pp. 324–329, 2014.

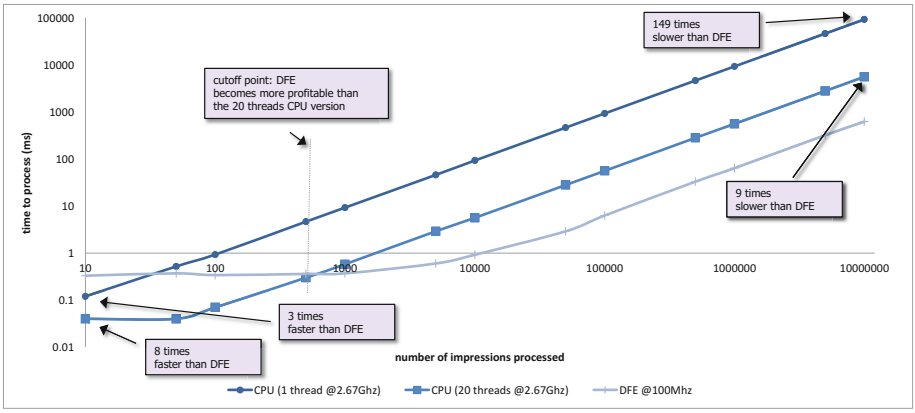

Fig. 1. Results comparing three AdPredictor implementations running on a CPU and DFE platforms. In this example, the size of the task (number of impressions to be processed) affects how each implementation fares against other implementations.

grow and shrink over time. At play here are many conflicting concerns involving application requirements, resource capacity and availability, and pricing.

The HARNESS project envisions an enhanced cloud Platform-as-a-Service (PaaS) software stack that not only supports existing commodity technologies, but also incorporates heterogeneous technologies such as Dataflow Engines (DFEs) [5], programmable routers and different types of storage devices, to provide vastly increased performance, reduced energy consumption, and lower cost profiles. To realise this goal, we are working on a platform design that abstracts the underlying infrastructure, enabled by runtime management systems, programming tools and middleware layers.

In this paper, we focus on the problem of making effective use of specialised computing resources, including DFEs and GPGPUs, in the context of a cloud platform. In particular, we have identified three key challenges in addressing this problem: performing global optimisation across a set of provisioned heterogeneous resources (Section 2), virtualising computing resources to enable sharing in a multitenant environment (Section 3), and programming heterogeneous platforms (Section 4).

2 Cross-Optimisation of Heterogeneous Resources

In current computational systems, heterogeneity is largely invisible to the operating system, and only minimal management functionality is provided. Accelerators such as GPGPUs and FPGAs are often accessed by applications as I/O devices via library-call interfaces. These accelerators must be manually managed by the application programmer, including not just execution of code but also in many cases tasks that are traditionally performed by the operating system such as allocation, de-allocation, load balancing, and context switching. If resources

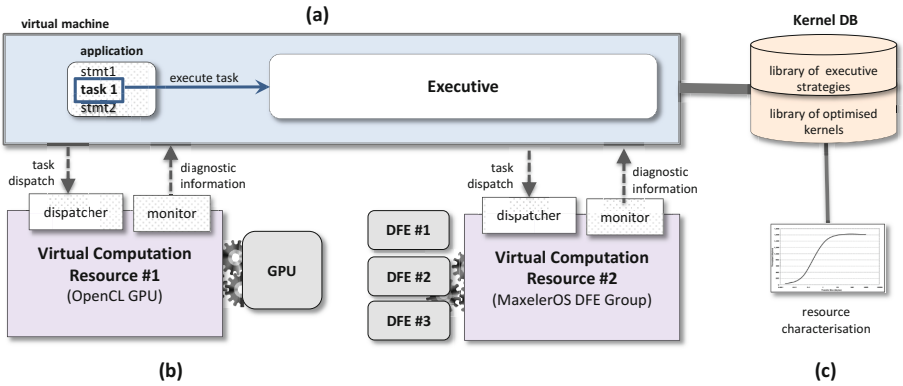

Fig. 2. Our runtime management system performs cross-optimisations over set of provisioned heterogeneous resources

are shared between several hosts, issues of contention, fairness and security become further pronounced. Since heterogeneous computing elements are generally outside the control of the operating system or other system software, global optimisation of resources for performance and energy efficiency, for instance, requires considerable programming effort and expertise.

The need for a system that automatically performs global optimisation of resources is illustrated in Fig. 1. In this example we compare the performance of the AdPredictor [2] training process (a computationally intensive machine-learning application) with three different implementations: a single-threaded CPU version (CPU-1), a 20-threaded CPU version (CPU-20) and a 100Mhz DFE version. The CPU platform is based on a dual Intel Xeon X5650 with 24 cores running each at 2.67Ghz. The DFE platform [5], on the other hand, contains a Virtex-6 FPGA as the computation fabric and external RAM for bulk storage. The AdPredictor training module processes a log of online advertisement views, called ad impressions, to update a model that predicts whether a user will click an ad when visiting a website. The size of an AdPredictor task corresponds to the number of ad impressions to be processed. It can be seen from Fig. 1 that the DFE version is not efficient for small tasks, however, at 10 million impressions, the DFE version runs an order of magnitude faster than the multithreaded version. In this example, the task size influences the relative performance of these three designs, with smaller task sizes performing better on the single and multithreaded CPU, and large tasks performing better with the DFE.

We are developing a computation management system [4] that automatically makes these allocation decisions at runtime (Fig. 2). In particular, our runtime management system processes jobs dispatched by the cloud platform using the set of provisioned computing resources to satisfy a given goal or policy, such as minimising job completion. Each job, triggered when an application is launched on a cloud platform, is processed in a virtual machine (VM). An application contains two types of code: *standard code* that is executed directly by the CPU hosting the VM, and *managed tasks*. Managed tasks are special program functions

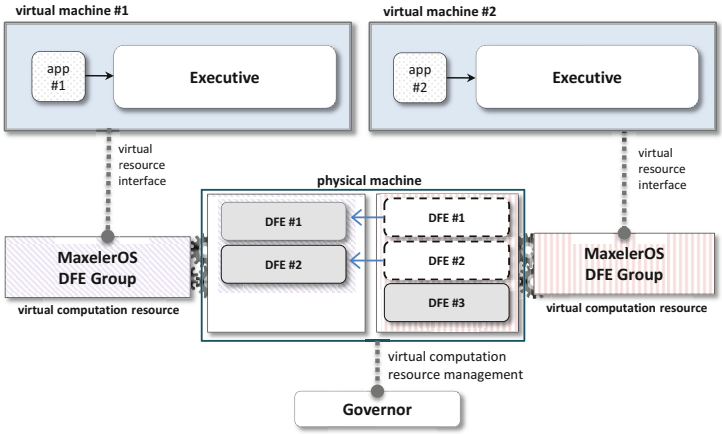

Fig. 3. Virtualisation of Maxeler DFEs to support resource sharing and elasticity

that are executed onto one or more provisioned computing resources through a queue-based mechanism. In particular, during the application execution, managed tasks are dispatched to a component called the *executive* (Fig. 2**(a)**), which decides how to allocate workload (Fig. 2**(b)**) based on the availability of optimised kernel implementations stored in a database (Fig. 2**(c)**) along with associated performance models, such as the one presented in Fig. 1. These performance models allow the executive to make intelligent decisions about how to optimise workload based on runtime conditions, such as task size. Other factors that can affect these decisions include accrued historical data, dependencies between tasks and availability of computing resources.

3 Virtualising Specialised Computing Resources

Specialised computing resources, such as FPGAs and GPGPUs, are designed to be single-tenant devices and typically do not provide native mechanisms that allow these resource to be shared by multiple users. In contrast, CPUs are managed by the operating system which transparently stores and restores the context of a process, so that multiple processes can share a single CPU.

We have designed a virtualisation mechanism for DFEs that in addition of supporting resource sharing, also supports elasticity where a single virtual computing resource can accumulate or shed multiple physical resources according to workload. Virtual computing resources (Fig. 2**(b)**) supported by our runtime management system adhere to the same interface, which allows the *executive* component to dispatch tasks and acquire diagnostic information, such as temperature and power consumption, without having to deal with proprietary interfaces.

To illustrate our virtual computing resource mechanism we present an example in Fig. 3 in which two applications are running each on a VM. In this example,

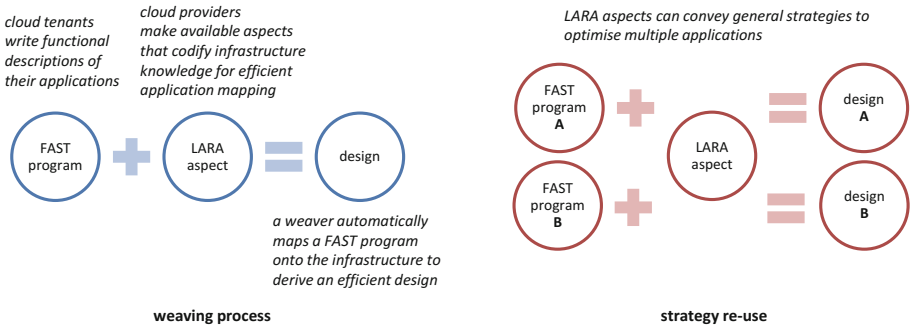

Fig. 4. Aspect-oriented programming methodology to support the HARNESS Cloud Platform

each of the applications has one virtual DFE provisioned. A virtual DFE can be instantiated to employ a fixed or variable number of physical DFEs. If the virtual DFE (also known as a *DFE group*) is configured with variable physical resources then physical DFEs are automatically re-allocated from one virtual resource to another depending on the workload. This management of shared resources is performed by a component called the *governor*. Shared resources are particularly important for online jobs: cloud tenants, rather than provisioning exclusive resources that are only used some of the time due to temporary bursts of workload, can instead share those resources with other tenants to minimise their cost, while cloud providers are able to maximise resource utilisation.

4 Application Development

Developers must acquire considerable knowledge and expertise to effectively program heterogeneous platforms. Heterogeneous platforms may include an arbitrary number of computing resources, such as DFEs, GPGPUs and multi-core CPUs. Developers of these platforms must, therefore, be aware of a number of architectural details including: the different types of processing cores which may exhibit various levels of complexity, the communication topology between processing elements, the hierarchy between different memory systems, and built-in specialised architectural features. There are two common programming approaches that address heterogeneity: (1) a uniform programming framework supporting a single programming language and semantics to target different types of computing resources; (2) a hybrid programming framework in which developers must manually partition and map their applications using the most suitable languages and tools.

We are developing the *Uniform Heterogeneous Programming* approach (see Fig. 4), which aims to combine the benefits of the above two approaches by using two complementary languages: FAST [3] and LARA [1]. With FAST, developers (cloud tenants) use a single software language (based on C99) to implement

their applications with the possibility of using multiple semantics to describe alternative versions of the same algorithm. For instance, with dataflow semantics, C99 code is translated into functional units that are mapped into reconfigurable logic to realise deep pipelined architectures, in which data is computed in parallel and the output is forwarded synchronously to the next functional unit. We believe FAST simplifies not only the compilation and optimisation design-flow using a single code base, but also simplifies the programming effort when targeting specialised computing resources. With LARA, on the other hand, hardware infrastructure experts (for instance, working on behalf of cloud providers) can codify domain specific knowledge into special programs called *aspects* which analyse and manipulate (naive) FAST programs. Subsequently, a process called *weaving* automatically combines non-functional (LARA aspects) and functional concerns (FAST programs) to derive designs that are optimised for a specific cloud platform and infrastructure.

5 Conclusion

In this paper we presented HARNESS, an FP7 project which aims to develop the architectural principles that enable the next generation of cloud platforms to provide increased performance, reduced energy consumption, and lower cost profiles. In the context of this project, we are developing a runtime management system that supports cross-optimisation and virtualisation of heterogeneous resources to provide managed multitenancy and elasticity. In addition, we are developing an aspect-oriented programming approach which allows programs capturing multiple semantics to be mapped efficiently to the HARNESS cloud platform. Future work includes integrating and evaluating our heterogeneous cloud platform and development tools with industrial use cases.

References

1. Cardoso, J.M.P., Carvalho, T., Coutinho, J.G.F., Luk, W., Nobre, R., Diniz, P., Petrov, Z.: LARA: An aspect-oriented programming language for embedded systems. In: Proceedings of the Annual International Conference on Aspect-Oriented Software Development, pp. 179–190 (2012)
2. Graepel, T., et al.: Web-scale Bayesian click-through rate prediction for sponsored search advertising in Microsoft's Bing search engine. In: Proc. of the Intl. Conf. on Machine Learning, pp. 13–20 (2010)
3. Grigoras, P., Niu, X., Coutinho, J.G.F., Luk, W., Bower, J., Pell, O.: Aspect driven compilation for Dataflow designs. In: Proc. of the IEEE Conference on App-Specific Sys. Arch. and Proc. (ASAP), pp. 18–25 (2013)
4. O'Neill, E., McGlone, J., et al.: SHEPARD: Scheduling on HEterogeneous Platforms using Application Resource Demands. In: Proc. of the Intl. Conf. on Parallel, Distributed and Network-based Processing (2014) (to appear)
5. Pell, O., Averbukh, V.: Maximum performance computing with Dataflow engines. Computing in Science Engineering 14(4), 98–103 (2012)
6. Schubert, L., et al.: Advances in clouds: Research in future cloud computing. Expert Group Report, European Commission, Information Society and Media (2012)