Co-funded by the European Commission within the Seventh Framework Programme

Project no. 318521

HARNESS

Specific Targeted Research Project
HARDWARE- AND NETWORK-ENHANCED SOFTWARE SYSTEMS FOR CLOUD COMPUTING

General Requirements Report

D2.1

Due date: 1 April 2013
Submission date: 25 April 2013

Start date of project: 1 October 2012

Document type: Deliverable
Activity: RTD
Work package: WP2

Editor: John McGlone (SAP)

Contributing partners: all

Reviewers: WP Leaders

Dissemination Level

PU	Public	√
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Revision history:

Version	Date	Authors	Institution	Description
0.1	2012/12/10	John McGlone	SAP	Outline
0.2	2013/03/22	*all*	*all partners*	Intial full draft
0.3	2013/03/29	John McGlone	SAP	Edits based on partner reviews
1.0	2013/04/24	Alexander Wolf	IMP	Final review and edits by Coordinator

Tasks related to this deliverable:

Task No.	Task description	Partners involved°
T2.1	Analyse state of the art and survey general requirements	IMP, EPL, UR1, ZIB, MAX, SAP*

°This task list may not be equivalent to the list of partners contributing as authors to the deliverable
*Task leader

Executive Summary

The *Hardware- and Network-Enhanced Software Systems for Cloud Computing* (HARNESS) project aims to advance the state-of-the-art in cloud data centres to enable cloud providers to profitably offer and manage the tenancy of specialised hardware technologies, while also enabling software developers to seamlessly, flexibly and cost-effectively integrate these technologies with their cloud-hosted applications.

This report offers a state-of-the-art analysis of cloud computing, separated into the key research areas of HARNESS, namely *platform-as-a-service* (PaaS), computation, communication and storage. It highlights the current limitations of each area in terms of scalability, heterogeneity, resource management and sharing, and performance. Looking specifically at current cloud platform offerings, most infrastructure providers offer their own in-house PaaS components. However, a number of innovative third parties provide PaaS offerings that cover multiple infrastructure providers. That said, support for specialised resources, and applications that can benefit, is not currently provided at the PaaS layer, most likely because of the complexity involved in their deployment and use, as well as the simple fact that they represent non-commodity investments.

Tenant over-provisioning to guarantee *quality of service* (QoS), coupled with the limited resource sharing options that exist in current clouds, lead to cloud providers not being able to fully exploit their infrastructure. In terms of performance, while cloud providers can offer considerable performance improvements for embarrassingly parallel applications, accelerated services are currently not directly available to tenants and, instead, are under the control of experts of the cloud provider. Within typical data centres, the network is provisioned in a tree-like structure, with edge switches at the edge of the tree and core switches at the root, and with links often over-provisioned. Such an infrastructure can result in high maintenance costs for the various switches and an inability to accommodate application-specific traffic patterns among servers.

Cloud providers usually offer limited choices from three categories of storage solutions: databases, file-based storage, and block-based storage. Elasticity is provided as vertical scaling for databases, and vertical and horizontal scaling for file-based storage. In the case of block-based storage, vertical scalability is limited by physical device sizes. Most providers offer storage based on both *hard disk drive* (HDD) and *solid-state drive* (SSD) devices. Many abstract based on performance, which can depend on the chosen cloud provider and on options selected by the tenant. Usually, storage solutions are implemented using standard open-source software, facilitating easy vendor migration.

Looking at past and present EC FP projects, this report identifies possibilities for technology transfer. While there are a number of projects targeting complementary PaaS and cloud management, which may provide a solid foundation, none of the projects specifically address the challenges of efficiently hosting applications in heterogeneous cloud environments. Computation-related projects provide insight into the programming models, design flows, real-time resource management and how to efficiently exploit reconfigurable devices in heterogeneous environments, but none address this within a cloud environment. Existing networking projects can be exploited for the management and/or programming of networking resources, while other projects can be useful to infer the correct interfaces for programming network

devices. However, the networking requirements within HARNESS are fundamentally different and it will therefore be highly unlikely that current interfaces defined in these projects can be used. Looking at storage, previous and on-going FP projects typically concentrate on high-level abstractions for data access and computational storage, while there are a number that address storage densities for rotating disks and *non-volatile random access memory* (NVRAM).

The general requirements presented in the report cover the key technologies that will be developed within HARNESS. For each technical area the report states how past and present FP projects relate or may possibly contribute to fulfilling these requirements, and also shows how the state-of-the-art is improved by the HARNESS requirements.

Requirements for the PaaS layer will ensure it is capable of executing applications on heterogeneous computation, communication and storage resources. Providing mechanisms for flexible specification of applications will also enable dynamic resource allocation for applications to meet requirements such as cost, resource usage and *service-level agreement* (SLA) constraints. This will also allow the cloud operator to manage applications to meet its objectives for load-balancing, resource utilisation, energy efficiency, and the like.

Requirements for the computation layer aim at ensuring the platform is informed of the available computational resources. It will also provide characterisation of heterogeneous applications so that run-time benefits can be ascertained and the platform can make informed decisions on application placement. Once placement decisions have been provided by the platform, the computational layer will then aim to optimise the use of resources according to the constraints provided. It will also provide monitoring data to the platform to support the placement decisions, and provide functionality to enable the shared use of heterogeneous resources such as *data-flow engine*s (DFEs) and *field-programmable gate array*s (FPGAs). Finally, the computational layer will also provide a compiler infrastructure for deriving multi-target design variants, reducing time-to-market of accelerated services for cloud tenants and enabling the tailoring of designs for particular requirements to provide a level of elasticity for heterogeneous resources.

Communication requirements will ensure that the platform layer is informed of available networking resources and capacity, and supported additional functionality for in-network processing and expected performance. The network layer will also allow the platform layer to allocate resources for applications and report network resource usage. Support for programmable networking will also require an interface is provided for leveraging *network processing unit*s (NPUs) and ensuring verification of NPUs.

The storage layer aims to provide various storage solutions with different performance profiles to meet differing application requirements, while also providing this information to the platform layer. This necessitates the mapping of several application requirements to singular devices, which will provide lower costs to tenants, but will require addressing the challenges of ensuring performance per application and implementing optimal mapping of reservations to devices.

The outlined requirements concentrate on the goals of the HARNESS project and involve much innovation. In meeting these requirements the project will develop the use of heterogeneous resources in cloud computing and considerably advance the state-of-the-art in cloud computing.

Contents

Acronyms

API *application programming interface*. 9, 12, 22, 23, 31

ASIC *application-specific integrated circuit*. 18

DFE *data-flow engine*. ii, 2–5, 16, 17

DSP *digital signal processor*. 8, 27

FPGA *field-programmable gate array*. ii, 2–4, 8, 15–17, 27

GPGPU *general-purpose graphics processing unit*. 2–4, 8, 9, 11, 15, 17, 28

HARNESS *Hardware- and Network-Enhanced Software Systems for Cloud Computing*. i, ii, iv, 1, 2, 4, 5, 7, 8, 13–17, 19, 21, 22, 24, 25, 27–33, 35

HDD *hard disk drive*. i, 12

IaaS *infrastructure-as-a-service*. 7, 9, 21, 23–25, 32

IOPS *input/output operations per second*. 12, 19

IoT *Internet of Things*. 30

LGPL *Lesser General Public License*. 25

MPSoC *multiprocessor system-on-chip*. 27

NPU *network processing unit*. ii, 4

NVRAM *non-volatile random access memory*. ii

OMG the *Object Management Group*. 23

PaaS *platform-as-a-service*. i, ii, 1, 2, 7–9, 13, 21–25

PCI *peripheral component interconnect*. 8

PCM *phase-change memory*. 2

POSIX *portable operating system interface*. 11

QoS *quality of service*. i, 9, 23, 24, 30, 31

RAID *redundant array of independent disks*. 11, 12

SLA *service-level agreement*. ii, 5, 13–15, 21, 24–26

SSD *solid-state drive*. i, 2–4, 12

VM *virtual machine*. 3, 8, 9, 24, 30

1 Introduction

1.1 Purpose and Outline of Report

This report documents the general requirements identified for the *Hardware- and Network-Enhanced Software Systems for Cloud Computing* (HARNESS) platform from an analysis of the state-of-the-art and relevant FP7 projects. This task extends the initial state-of-the-art analysis to identify requirements for the HARNESS platform that can be established from existing literature and projects.

In this first chapter, the key concepts of the project are introduced, highlighting the HARNESS vision and project objectives. Chapter 2 addresses the state-of-the-art in cloud computing with a focus on the key technology areas within the HARNESS project, namely cloud platforms, computation, communication and storage. Chapter 4 provides an overview of relevant projects, again covering the key technology areas, with a focus on FP7 projects and how past and present FP7 projects can relate and contribute to fulfilling HARNESS requirements. Chapter 3 presents the HARNESS general requirements, a description of each with details on Innovation, importance and dependencies along with task relationships. Chapter 5 concludes the document.

1.2 Concepts and Objectives

Cloud computing is reshaping the IT landscape, with increasingly large numbers of businesses, governments, and scientists seeking to offload mission-critical applications to third-party data centre providers.

Today, the dominant approach to constructing data centres is based on the assembly of large numbers of relatively inexpensive personal computers, interconnected by standard IP routers and supported by stock disk drives. This is consistent with the current business model for cloud computing, which leverages commodity computation, communication, and storage to provide low-cost application hosting. Taken together, the resources offered by cloud providers constitute a platform upon which to execute applications. The efficacy of this *platform-as-a-service* (PaaS) depends upon the provider's 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.

This allocation and optimisation problem exists whether the cloud data centre is operated for public use, as done by Amazon, Microsoft, and many smaller players, or for private use, such as by financial and government institutions. Two key concepts underlying the cloud data centre allocation and optimisation problem are *managed multi-tenancy* and *elasticity*. The first concept captures the idea that the provider must accommodate and reconcile the resource needs of several applications simultaneously, while the second captures the idea that an application's allocation of resources should grow and shrink over time. At play here are many conflicting concerns, principally application requirements, resource capacity and availability, and pricing (again, whether billed externally or internally).

In trying to solve this problem, providers are naturally led to a platform design that abstracts and homogenises, which explains today's ubiquitous platform elements: virtualisation, simple key/value stores

and distributed file systems, map/reduce data-flow computational structures, service-oriented architectures, and the like.

Despite continuing efforts to refine and improve this basic foundation through numerous research and innovation programmes, cloud data centre providers are realising that a fundamental limit has been reached in the ability of traditional application scaling alone to adequately achieve specific performance, availability, and cost requirements for many important applications. This limit is threatening the broader adoption of cloud computing, and thereby threatening the market for cloud services. Data centre providers must therefore rethink and reformulate the foundations of their computing platform, and look to radical new ways of satisfying application requirements.

The seed for a new approach can be found in the emergence of innovative hardware and network technologies. These technologies span from computational devices, such as *data-flow engine*s (DFEs) utilising *field-programmable gate array*s (FPGAs) and *general-purpose graphics processing unit*s (GPGPUs), to new communication fabrics such as programmable routers and switches, to new storage devices and approaches, such as *solid-state drive*s (SSDs) and *phase-change memory*s (PCMs).

Advanced hardware and network technologies for improving computation, communication, and storage have yet to be integrated into cloud computing platforms. The simple reason is that they break the abstract, homogeneous, and commodity nature of today's PaaS computing model. The exception that perhaps best proves this rule can be found in Amazon's recent offering of direct access to the GPGPUs installed on some of its data centre servers [5]. Application developers can choose to implement (portions of) their applications for execution in the standard CPU environment or in the highly parallel and non-traditional GPGPU environment. But they must make this choice explicitly, adopt radically different designs for each case, and find ways to manage the allocation of the resources themselves. This substantially increases the cost and complexity of application development, shifts some amount of responsibility for resource management from the cloud provider back onto the application operator, and significantly reduces the flexibility of the cloud provider to optimally manage multiple tenants.

We can summarise the current situation by observing that the existing model of the cloud data centre is a barrier to the adoption and exploitation of the newest generation of computing technologies. These technologies offer significant benefit to end users and society at large in terms of reduced costs, reduced energy use, and increased performance. However, they also present enormous challenges to both application developers and cloud operators to deliver this benefit.

1.3 HARNESS Vision

HARNESS aims to advance the state of the art in cloud data centre design so that:

1. *cloud providers* can profitably offer and manage the tenancy of specialised hardware and network technologies, much as they do today's commodity resources; and

2. *software developers* can seamlessly, flexibly, and cost-effectively integrate specialised hardware and network technologies into the design and execution of their cloud-hosted applications.

To realise this goal, we plan to develop an enhanced cloud PaaS software stack that not only incorporates a wide variety of specialised technologies, but embraces the heterogeneity (of performance and of cost) that those technologies embody. These resources offer a much richer, but also more complex, context

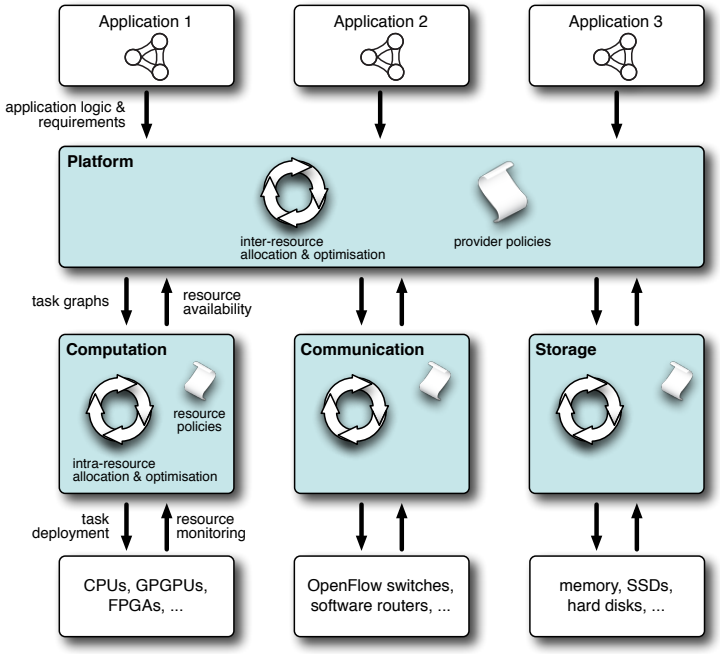

Figure 1.1: Overview of the HARNESS approach.

in which to make price/performance trade offs, bringing wholly new degrees of freedom to the cloud resource allocation and optimisation problem. Further, those trade offs are inherently time dependent and time critical, with the dynamic goals and behaviour of the application, the user, the other cloud tenants, and the cloud provider all intersecting and sometimes conflicting through time.

In our vision, depicted in Figure 1.1, a demanding cloud application consists of a number of components, some of which can have multiple, alternative implementations that exhibit different resource demands, performance characteristics, and cost. Applications express their computing needs to the cloud platform, as well as the price they are prepared to pay for various levels of service. This expression of needs and constraints builds upon what can be expressed through today's simple counts of *virtual machine*s (VMs) or amounts of storage, to encompass the specific and varied new factors characteristic of specialised hardware and network technologies.

The cloud platform will have access to a variety of resources to which it can map the components of an application. Resources include standard VMs, commodity CPUs, and stock disk stores, of course, but also specialised devices such as various types of high-end FPGA-based DFEs and GPGPUs, network appliances, and advanced storage media such as SSDs. A flexible application may potentially be deployed in many different ways over these resources, each option having its own cost/performance/usage characteristics.

Our approach operates at three levels. At the lowest level are the specialised technologies themselves. These are virtualised into the computation, communication, and storage resources available at the next level. The platform level at the top provides a management and programming abstraction to applications. The idea is to provide flexibility to the platform as to which and how many resources

within a category are used and when, and to separate that concern from the low-level deployment and monitoring of the concrete elements.

Overall, the platform is charged with making informed decisions as to the optimal deployment that can simultaneously satisfy tenant applications (e.g., performance and price) and the cloud provider (e.g., profitability or energy efficiency). We see these deployment decisions taking place upon application start up or at well-defined points during execution (i.e., as a dynamic re-deployment or reconfiguration). The optimal deployment may thus change at run time based on dynamic criteria such as variations in the workload, changing performance expectations, and the spot price and availability of specialised computing devices.

Instead of exposing heterogeneity at the bare device level (as done today, for example, by Amazon with its GPGPU facility mentioned above), tenants should be able to describe needs and constraints for their applications, including an indication by when they will require results to be delivered or how much they are willing to pay, so that the provider, based on the current state of the data centre, can decide how and when to allocate the appropriate mix of commodity and specialised resources. This will be a fundamental paradigm shift from the traditional resource-oriented view of cloud computing to one that we characterise as *results oriented*. This shift has broad implications and far-reaching consequences for the design, development, and use of cloud data centres.

1.4 Scientific and Technical Objectives

This section details the main scientific and technical objectives of the HARNESS project.

1.4.1 Models for heterogeneous resources and application characterisation

The first objective is a precise characterisation of the heterogeneous hardware resources used in HARNESS. These include computational resources (e.g., DFEs/FPGAs and GPGPUs), network switches and routers (e.g., *network processing unit*s (NPUs)), and storage systems (e.g., SSDs). The goal is to extract the key properties of these resources, for example in terms of processing throughput or access latency. These will be needed to build accurate performance models used during the resource allocation and optimisation processes.

1.4.2 Development of a management platform for heterogeneous clouds

A key outcome of HARNESS will be to provide a software infrastructure to manage heterogeneous resources. We envision that this platform will operate at both the resource and data centre levels. At the resource level, the HARNESS platform will be responsible for executing applications (or parts of them) on appropriately virtualised heterogeneous resources. At the data centre level, the platform must manage all resources and mediate between different applications and resources.

1.4.3 Design of new programming interfaces and abstractions

HARNESS will shift away from the traditional, resource-oriented cloud programming models in favour of results-oriented paradigms, in which tenants specify their application logic and requirements in high-level languages. This will enable cloud providers to satisfy these requirements by mapping applications

to specific heterogeneous resources using the platform. This requires flexible and efficient programming interfaces and abstractions. On the one hand, they should be expressive enough to allow applications to achieve high performance; on the other hand, they must be flexible enough to permit cloud providers the opportunity to decide at run time which and how many resources to dedicate to a given application, potentially re-allocating resources when operating conditions (or prices) change. HARNESS will develop two distinct sets of programming interfaces. One will be used by cloud tenants to specify desired application behaviour in a resource-agnostic fashion. After the resource allocation and optimisation performed by the HARNESS platform, the other, resource-specific programming interfaces will be used by the platform to deploy and control application components on particular resources, such as DFEs and software routers.

1.4.4 Support for run-time monitoring, fine-grained resource accounting, and pricing models

To make decisions about the best mapping of application components to heterogeneous resources, the HARNESS platform must monitor resource availability and observed application performance. Particular attention must be devoted to identifying "straggling" resources, that is, ones performing significantly worse than expected. This is especially relevant for distributed computations, such as MapReduce [26], whose total completion time is dominated by the slowest task to finish. Resource monitoring ensures that resources that deviate from their performance targets are identified early and replaced by others through task re-allocation. Run-time monitoring must also provide fine-grained accounting statistics on resource usage that can inform the pricing and billing models of the cloud provider. Information about resource usage must be collected in a scalable fashion and with necessary detail, without impacting application performance. Pricing models used by cloud providers can then reflect the heterogeneous nature of resources in terms of the widely differing capital and operational expenditures of individual heterogeneous resources.

1.4.5 Algorithms for heterogeneous resource allocation and optimisation

At the core of the HARNESS approach are algorithms that receive as input the application requirements and characteristics, the current state of the heterogeneous resources annotated with their performance models, and the provider's internal policies and tenant *service-level agreement*s (SLAs), and then output the set of resources that should be assigned to an application. This resource allocation process is not a static process, but rather it should run continuously and adapt to changing conditions, such as when new tasks arrive or resource performance degrades.

1.4.6 Technologies for resource virtualisation and sharing

Closely related to the objective of resource allocation and optimisation is the question of how to share heterogeneous resources efficiently. This objective involves the development of new approaches for the efficient and secure virtualisation of heterogeneous resources, including hardware accelerators, programmable network devices, and storage technologies. HARNESS will address this objective along three dimensions: scalability, performance isolation, and security.

2 State of the Art in Cloud Computing

In this section we discuss the current state-of-the-art in cloud computing. This is broken down into four technical areas: platform, computation, communication, and storage. For each area we present the status quo and highlight the limitations of the technology in terms of scalability, heterogeneity, resource management and sharing, and performance. It is the purpose of the HARNESS project to address these limitations; the general requirements to do so are presented in Chapter 3.

2.1 Platform

PaaS refers to a variety of cloud middleware services intended to facilitate the use of cloud infrastructure resources by application programmers. As such, it covers a wide variety of services, ranging from elementary components such as load balancers and NoSQL databases, to MapReduce [26] and similar high-performance computational frameworks, to integrated run-time environments capable of hosting entire complex applications in the cloud.

- **Platform operators.** Many cloud infrastructure providers complement their commercial offer with PaaS components. For example, the Amazon cloud provides services such as RDS [10] (a relational database service), DynamoDB [6] (a non-relational database service), Elastic Beanstalk [8] (a Web application container service), and many more. Similarly, Google provides App Engine [36] (a Web application container service) and Microsoft provides Windows Azure [52] (an application container service). Acting as both the infrastructure and the platform service provider offers a number of technical benefits, such as giving low-level monitoring information to the platform layer.

  However, some of the most innovative PaaS offerings are developed by third-party entities that exploit computing resources acquired from classical *infrastructure-as-a-service* (IaaS) providers. For example, Heroku [44] provides easy-to-use hosting services for applications written in a large variety of programming languages. ConPaaS [20, 63] supports applications designed as complex compositions of elementary elastic services. Separating the platform operator from the infrastructure provider allows certain platforms to support a variety of underlying cloud infrastructures. For example, CloudFoundry [19] (developed as open-source software by VMware) obviously supports VMware's vSphere virtualisation infrastructure, but it can also run in the Amazon cloud. Similarly, ConPaaS supports the execution of cloud applications across multiple cloud infrastructures.

- **Application domains.** A vast majority of applications running in the cloud are arguably Web-based. As a consequence, most of the cloud platform services specifically focus on this particular domain. Another important application domain in the cloud is high-performance computing (data mining, scientific applications, etc.). A number of cloud platforms target this type of application by providing, for example, MapReduce frameworks. Interestingly, although many real-world

applications combine aspects of a high-performance execution back end and a Web-based front end, few platforms integrate both functionalities in a single offering.

- **Heterogeneity.** We are not aware of public platform services that are specifically capable of exploiting heterogeneity in the cloud. All the well-known PaaS systems rely exclusively on traditional server, networking, and storage resources [8, 20, 36, 44, 52]. It is clear, however, that good use of such heterogeneous resources can only increase the scalability and cost-efficiency of demanding applications. The main expected contribution of the HARNESS project is therefore to enable cloud applications to benefit from the combined strengths of a variety of heterogeneous resources in present and future cloud infrastructures.

2.2 Computation

The cloud infrastructure fundamentally hinges on servers, switches, and disks. By abstracting IT services from the underlying hardware infrastructure, the cloud platform can provide computational resources that can be scaled on demand in a multi-tenant environment. In this section, we provide an overview of the state-of-the-art in cloud computation according to some of its relevant features [67].

- **Scalability.** In the current cloud landscape, data centres scale horizontally to cope with the increasing complexity of computational requirements, number of users, and economies of scale. In horizontal scaling, hosts are replicated to meet demand. However, this solution requires computations to be balanced across a wider range of resources, placing an extra burden on the network infrastructure as well as increasing the energy footprint. Vertical scaling, on the other hand, is seldom exploited by cloud providers, apart from upgrading to faster machines. This is because vertical scaling requires complex analysis of the application logic running on the cloud, such as internal dependencies and communication requirements, to exploit the capabilities of local computational nodes, such as parallel architectures and specialised functional units.

- **Heterogeneity.** The cloud infrastructure is generally built around a homogeneous set of commodity hardware. Since general-purpose processors (CPUs) are limited by power and compute density, heterogeneous resources acting as accelerators, such as FPGAs, *digital signal processor*s (DSPs), and GPGPUs, are capable of improving these metrics by orders of magnitude [55]. In addition, such accelerators can be specialised for specific types of computations to achieve more performance and energy efficiency. Cloud providers such as Amazon are recognising the benefits of heterogeneity for the larger high-performance computing community, and offer GPGPU cluster instances [5]. However, one of the challenges for adopting heterogeneity in cloud environments is that native virtualisation support in accelerators is limited at best, but more typically non-existent. This makes them difficult to share across multiple VMs. A notable exception is NVIDIA, which introduced the VGX platform [57] in 2012. This platform allows a new generation of GPGPUs to be virtualised by supporting, for instance, a hardware memory management unit that translates virtual addresses, and a software-based GPGPU hypervisor allowing multiple VMs to use a single physical GPGPU.

  For hardware that does not support virtualisation, a common strategy, as adopted by Amazon EC2 [7] for their GPGPU cluster instances, is to use *peripheral component interconnect* (PCI)

pass-through to provide the VM direct access to accelerators [77]. On the other hand, instead of passing the physical accelerators to the VM, approaches such as gVirtuS [35], vCuda [68], and GViM [42] virtualise GPGPUs by using *application programming interface* (API) call interception and redirection between the software API operating in the VM and the physical GPGPU installed in the host system.

Another form of virtualisation is through *elastic functions* [48, 76]. Unlike normal C-like functions, elastic functions can capture multiple implementations (variants) with meta-information, such as assumptions about characteristics of the input data. This allows a run-time system to make informed decisions about which implementations to use to optimise the application code on potentially different architectures. Developers simply need to call an elastic function, and rely on the resource management system to make run-time decisions about how the workload is going to be distributed across a multi-core heterogeneous system.

- **Resource management.** A typical cloud computing service, such as Amazon EC2, provides users the option to select the number of virtual instances, the amount of memory, number of cores, amount of storage and type of architecture (32-bit/64-bit). In addition, users can opt to use automatic scaling, in which the number of virtual instances is automatically increased or decreased according to actual usage. This dynamic management of resources is yet far from optimal. To start with, it is difficult for potential tenants to assert the relationship between pricing, effort and benefits of using the cloud. This is largely due to the difficulty of establishing cost models and mechanisms that allow tenants to perform capacity planning. To do so, tenants must understand compute patterns, such as how they change periodically, and what kind of infrastructure is necessary to support them [4]. Current automatic scaling technology is not fast enough to respond to sudden changes in capacity needs [51, 79]. This effectively forces cloud tenants to over-provision resources to guarantee quality of service at higher costs. On the other hand, cloud providers are not able to capitalise under-utilisation to fully exploit their infrastructure.

- **Resource sharing.** Cloud platforms allow multiple users to deploy applications on the same hardware infrastructure. Currently there are different mechanisms that support multi-tenancy according to the cloud model used. For IaaS approaches, such as Amazon EC2 and Google Compute Engine [37], the same physical hardware can be shared by multiple operating system instances using a hypervisor. On the other hand, in a PaaS context, such as Heroku [44] and Google App Engine [36], code and data are isolated for different applications and users on the same operating system instance. This prevents the need to allocate virtual instances for every launched application or service, but creating adequate isolation mechanisms remain a major challenge. In particular, one of the difficulties of implementing isolation is distinguishing how specific users affect resource consumption. This is a necessary capability not only to assess usage/cost value for cloud tenants, but also to guarantee *quality of service* (QoS) on a per-tenant basis.

  In addition, current resource sharing technology is still vulnerable to security issues [39] where, for instance, a tenant's data and code spaces can be compromised. Examples of computational resource vulnerabilities include: attackers analysing and exploiting vulnerabilities in virtual template images that are cloned to provide on-demand services; data leakage caused by VM replication; and cryptographic vulnerabilities due to weak random number generation brought by virtualisation layers.

- **Performance.** Cloud platforms are natural candidates for providing high-performance computing due to their potential to support large-scale hardware infrastructures. But there are factors as mentioned previously that affect performance, such as scalability level, type of infrastructure used, how the application was developed, and how resources are exploited. As the infrastructure becomes larger, more complex and heterogeneous, it becomes increasingly difficult for cloud tenants to effectively exploit the cloud infrastructure (which is mostly hidden from them), and cloud providers to take full potential of their infrastructure. A popular programming paradigm for exploiting clusters and data centres is MapReduce, which allows tenants to reformulate embarrassingly data-parallel programs to effectively exploit multiple distributed compute nodes. Other forms of exploiting the cloud infrastructure is the use acceleration services provided by the cloud provider in specific application domains [70].

2.3 Communication

Cloud providers do not typically disclose any information on their network architecture (e.g., how their servers and other computing devices are interconnected) or network management (e.g., how they make sure that the network is not overloaded, or how they handle link or network-device failures). Based on the glimpses that researchers get and report every now and then [3], the typical cloud network is organised in a tree: servers and other computing devices are physically connected to *edge switches* that are in turn physically connected to (fewer) *aggregation switches* that are physically connected to (even fewer) *core switches*. In this setting, then, the edge switches constitute the leaves of the tree, while the core switches are at the root. To reduce cost, the network links are typically oversubscribed and so cannot accommodate just any traffic pattern among the servers.

The research community has already identified at least two disadvantages of this approach: (1) the high cost of maintaining different kinds of switches, especially the aggregation and core switches, which need to satisfy significantly higher bandwidth requirements than the edge switches, and (2) the lack of support for full bisection bandwidth. The latter refers to the inability to accommodate arbitrary traffic patterns among the servers due to the over-subscription of the network links. To remedy these problems, researchers have proposed network architectures that offer full bisection bandwidth at a lower cost; they achieve this by using more sophisticated interconnection topologies (e.g., a fat tree instead of a classic tree [3]) and more sophisticated routing algorithms (e.g., valiant load balancing among some of the switches [38]). An alternative approach to reducing cost, although not necessarily offering full bisection bandwidth, is to replace traditional network devices with commodity servers [22, 40, 41].

We do not provide more details regarding the current and upcoming cloud network architectures, because our goal is not to innovate in that area. In particular, we do not want to change the network interconnect or routing algorithms, but rather the *programmability* of the network. Current network devices are typically not programmable, which means that once they are deployed, one cannot make them support new types of traffic processing. Yet, we have strong evidence that the performance of certain data centre applications, such as MapReduce, would benefit significantly from certain types of in-network computation [23]. What we want to do, then, is to enable the data centre operator to reprogram the network devices with new types of in-network computation, depending on the applications that are to be run.

Our goal is related to the recent work on software routers, which are network devices built from

programmable components, such as CPUs and GPGPUs. Several research prototypes have demonstrated that such programmable hardware is capable of high-performance packet processing (i.e., line rates of 10 Gbps or more), assuming simple, uniform workloads, where all the packets are subjected to one particular type of packet processing: IP forwarding [27], GPGPU-aided IP forwarding [43], multi-dimensional packet classification [50], or cryptographic operations [46]. Similar to this work, we want to build high-performance, programmable network devices. However, unlike this work, we are interested in network devices that can run a wide range of packet-processing applications and simultaneously serve multiple clients with widely different needs.

2.4 Storage

Storage in clouds comes in three flavours: databases, file-based storage, and block-based storage. We consider the state-of-the-art for each in turn.

- **Databases.** Most cloud providers provide managed databases for structured data. The offers can roughly be split into SQL and NoSQL databases. Almost all cloud providers provide some managed SQL databases such as basic Oracle Database [62] or MySQL [54]. NoSQL databases are usually Google BigTable [16] clones or data warehouse infrastructures such as Apache Hive [11].

- **File-based storage.** can be used as an interface to content-delivery networks, where the cloud provider distributes copies of the file world-wide, and access time for customers is lower. It can also be used as a front end to backup services, where multiple replicas of the objects (files) are stored to increase durability. We are not aware of any major cloud provider offering distributed file systems with *portable operating system interface* (POSIX) semantics, which would make the sharing of data between legacy applications easier.

- **Block-based storage.** This form of storage provides only raw block devices without a file system. The advantage is that the user can decide which file system they want to use and they can aggregate several *redundant array of independent disks* (RAID) volumes into one. The storage is either provided by local disks on the same physical node or on remote disks over the network, such as in Amazon EBS [9].

In the following, we use the same five features from the state of the art in cloud computation description to analyse storage.

- **Scalability.** For scalability, we have to analyse the three different storage flavours separately. For traditional SQL databases the most commonly used option is vertical scaling. Except for expensive hardware solutions, horizontal scaling is oriented towards read-only replicas. NoSQL databases are focused on horizontal scaling. This is transparently handled by the provider. In managed deployments, users either provision throughput or the number of servers.

  For file-based storage, both vertical and horizontal scalability can be handled seamlessly by the cloud provider. There are no practical limits on the number of objects (files) or the access frequency.

For block-based storage, vertical scalability is limited by the physical limits of the devices and how many devices can be aggregated using RAID. These are inherent limits that also apply to non-cloud environments.

- **Heterogeneity.** All major cloud providers have started to adopt SSDs. Depending on the application profile, users can now choose between *hard disk drive*s (HDDs) and SSDs. Often, it is also possible to provision resources on a more abstract level by specifying desired *input/output operations per second* (IOPS).

- **Resource management.** For databases and file-based storage, there are automatic scaling options within limits. For block-based storage, the reserved resources are usually fixed. But changing the deployment of data-intensive services is expensive because data must be copied.

- **Portability.** Storage portability is easier than for computational resources. Databases are often based on standard open-source software, which makes changing providers easier. For file-based storage, smaller providers tend to copy the market leader's APIs.

- **Performance.** Performance tuning is a major issue with storage. Some products come with tight performance guarantees, while others only provide best effort. Moreover, they lack mechanisms for isolating multiple tenants on the same device or service.

  Studies have shown that when Amazon introduced block storage with provisioned IOPS, storage performance became much more predictable. However, they can only guarantee performance for random access, as there are still no products that can guarantee performance for sequential access.

3 HARNESS General Requirements

This section presents the general requirements for HARNESS. These requirements cover the main technologies to be developed within the project: a PaaS platform for managing heterogeneous technologies and the infrastructures for managing heterogeneous computation, communication, and storage resources and their interactions with the PaaS platform management layer. Each requirement has an accompanying description, the level of innovation required to fulfil the requirement, the level of importance in fulfilling the requirement, possible dependencies with other requirements, and the related DoW tasks charged with fulfilling the requirement.

3.1 Platform

The goal of the HARNESS platform is to offer an execution environment capable of deploying applications over a collection of heterogeneous computation, communication, and storage resources in the cloud. It will support the specification of flexible applications capable of exploiting this heterogeneity and dynamically change its use of resources based on its current workload, the current cost and availability of specialised resources, and the SLA that it is supposed to achieve. Similarly, the cloud platform operator will be able to steer the resource usage of applications to satisfy their own objectives such as load balancing, consolidation, energy efficiency, and profit.

R1	**Provide an execution platform for using heterogeneous resources**
The HARNESS platform should provide an execution platform capable of deploying cloud applications over heterogeneous resources. State-of-the-art focus: *platforms, heterogeneity*.	
Task: T6.1, T6.2, T6.3	**Innovation: high** **Importance: critical** **Dependencies: R9, R11-R15, R17-24**

R2	Provide support for flexible applications
The HARNESS platform should support the execution of applications offering a level of freedom in the set of computational, communication, and storage resources they require. This may be realised by a variety of mechanisms: (i) defining rules for horizontal scalability of the application; (ii) authorising application developers to provide multiple implementations of the same functionality featuring different trade offs between the required resources and the achieved performance; (iii) automatically recompiling applications as a means to automate the generation of multiple versions of the application. The platform will autonomously choose one of the available deployment options in order to best enforce the required SLA. State-of-the-art focus: *platforms, heterogeneity*.	
Task: T6.1, T6.2, T6.3	**Innovation: high** **Importance: critical** **Dependencies: R3-R4,R6-R8, R11-R12, R14-R16, R20-23**

R3	Provide an application description language
The HARNESS platform should provide a description language that application developers can use to specify the functional and non-functional requirements of an application. The functional requirements include the list of software components of the application as well as the location of executables, input and output data. Non-functional requirements include a description of the different ways the application may be deployed on various sets of resources. State-of-the-art focus: *platforms*.	
Task: T6.1	**Innovation: medium** **Importance: critical** **Dependencies: none**

R4	Provide an SLA description language
The HARNESS platform should provide a language to allow application users to specify their expectations of application performance and execution costs. State-of-the-art focus: *platforms*.	
Task: T6.1	**Innovation: low** **Importance: moderate** **Dependencies: none**

R5	Implement application performance model generation
The platform should be able to conduct experiments in order to learn the performance behaviour of new flexible applications. State-of-the-art focus: *platforms*.	
Task: T6.2	**Innovation: high** **Importance: moderate** **Dependencies: R1-R2**

R6	Implement resource co-allocation scheduler (per application)	
The platform should be able to translate resource co-allocation requests into a set of single-resource reservation requests in order to best satisfy an application's SLA. State-of-the-art focus: *platforms*.		
Task: T6.2	**Innovation: high** **Dependencies: none**	**Importance: critical**

R7	Implement resource co-allocation scheduler (multi-application)	
When confronted with conflicting resource reservation requirements from multiple applications, the platform should make informed decisions in order to maximise the cloud operator's profits. State-of-the-art focus: *platforms*.		
Task: T6.3	**Innovation: high** **Dependencies: none**	**Importance: critical**

R8	Provide design patterns for flexible applications	
The project should identify best practices for the development of flexible applications to be deployed on the HARNESS platform. State-of-the-art focus: *platforms, application domains*.		
Task: T6.4	**Innovation: medium** **Dependencies: R2**	**Importance: moderate**

3.2 Computation

The next set of requirements relate to compile-time and run-time generation and exploitation of heterogeneous computation resources in the context of the HARNESS platform.

R9	Develop a heterogeneous computational resource discovery protocol	
The protocol should provide the cloud platform layer with an inventory of available computational resources such as CPUs, FPGAs, and GPGPUs (see Figure 1.1). This will be coarse grained in nature, providing the level of granularity required by the cloud platform to make high-level job placement decisions. Placement decisions will then be passed to a computational resource manager for run-time allocation. State-of-the-art focus: *heterogeneity, resource management*.		
Task: T3.1	**Innovation: medium** **Dependencies: none**	**Importance: critical**

R10	Develop a compiler infrastructure to derive multi-target design variants	
The compiler infrastructure should allow domain-specific knowledge about the application and/or hardware platform to be codified and subsequently applied in a systematic and automated way to generate multiple designs that satisfy different non-functional concerns such as performance, resource utilisation and energy efficiency. The development of high-performance applications is typically a long and error-prone process due to the complexity of the hardware processing elements and the multitude of possible design choices. With the proposed compiler infrastructure, cloud providers will reduce time-to-market of new acceleration services for cloud tenants, exploiting the underlying hardware infrastructure. In addition, multi-target design variants will support vertical scaling (elasticity) in HARNESS by allowing a resource manager to select which designs to use according to an established usage/cost trade off. State-of-the-art focus: *scalability, heterogeneity, performance*.		
Task: T3.1	**Innovation: high** **Dependencies: none**	**Importance: moderate**

R11	Provide characterisation of heterogeneous applications	
Characterise each low-level task, by heterogeneous resource used, based on the quantity of data, data transfer speed and other metrics, such that the benefit of using such a resource is known at run time. Using these low-level task characterisations, we will be able to provide an implementation evaluation that can be used by the HARNESS platform to select a particular application implementation in the absence of any accrued historical data. State-of-the-art focus: *scalability, resource management*.		
Task: T3.1	**Innovation: high** **Dependencies: none**	**Importance: critical**

R12	Enable shared use of heterogeneous computational resources	
While conventional CPUs can be shared between multiple processes, through the use of operating-system scheduling techniques, or multiple users entirely isolated, through the use of virtual machines, no such mechanisms exist for the full range of heterogeneous computational resources that will be exploited by HARNESS. Resource sharing will be critical for supporting multi-tenancy and maximising the benefits of heterogeneous computational units, particularly for high-value resources such as FPGA-based DFEs, where under-utilisation represents a significant wasted capital investment. State-of-the-art focus: *heterogeneity, resource management, resource sharing, performance*.		
Task: T3.3	**Innovation: high** **Dependencies: none**	**Importance: critical**

R13	Provide monitoring data to support decision making at the cloud platform layer
The cloud platform layer will allocate resources to applications based on an estimate of the performance of such an allocation. Estimates will be created from measured performance over the lifetime of the cloud's operation. In order to maximise the accuracy of these estimates, applications should measure and feedback performance data so that the cloud platform layer can learn from and improve upon allocation decisions. State-of-the-art focus: *resource management, resource sharing*.	
Task: T3.3	**Innovation: medium** **Importance: critical** **Dependencies: none**

R14	Optimise the use of resources according to optimisation goals within constraints provided by the cloud platform layer.
Given a budget or allocation of resources from the cloud platform layer, applications should aim to optimise their performance by choosing to run compute tasks on the best available resources at run time, based on the current demand placed upon the computational resources and the expected performance. The best device is chosen to be the one that can best satisfy the performance constraints that the cloud layer communicates, such as minimal execution time, highest precision or lowest power consumption. The resource manager will operate at a node level and have access to CPUs, local co-processors, such as GPGPUs and remote computational elements such as DFEs. Achieving best use of these resources is required to maximise profit for the cloud vendor and minimise cost for the consumer. Therefore, a key requirement of the resource manager is to manage application interactions with available resources to meet the goals set by the cloud platform layer. State-of-the-art focus: *scalability, resource management, performance*.	
Task: T3.2	**Innovation: high** **Importance: critical** **Dependencies: R6**

3.3 Communication

The next set of requirements relate to the integration of dependable and predictable programmable software routers and in-networking processing into the HARNESS platform.

R15	Enable programming of network processing units
The network should export and implement an interface through which cloud applications can leverage network processing units. For example, MapReduce applications can benefit significantly from in-network aggregation. The network should provide a way to add such aggregation functionality to programmable network-processing units, such as general-purpose CPUs or FPGAs. State-of-the-art focus: *network programmability, heterogeneity*.	
Task: T4.3	**Innovation: high** **Importance: critical** **Dependencies: none**

R16	Implement verification of properties of network processing units	
The properties of network-processing units should be verifiable. For example, suppose that a network processing unit is programmed to perform a certain kind of packet filtering. It should be possible to verify (prove) that this filtering does not cause the network processing unit to crash and introduces no more than a known, acceptable per-packet latency. State-of-the-art focus: *network heterogeneity, performance*.		
Task: T4.1, T4.3	**Innovation: high** **Dependencies: none**	**Importance: critical**

R17	Provide a description of network resources	
The network should report to the platform: (i) the availability of in-network processing units: (ii) the functions they support; (iii) their performance properties; and (iv) the capacity between each pair of directly connected network processing units. For example, the platform should know that there exists a switch with 64 10 Gbps ports, interconnected through an ASIC that performs switching and filtering at full bisection bandwidth, and that this ASIC is connected through a 100 Gbps link to 16 general-purpose 2.8 GHz cores that are programmed to perform in-network aggregation at a certain rate. This is necessary to enable the platform to allocate specific network resources to specific tenant applications. State-of-the-art focus: *network management*.		
Task: T4.2	**Innovation: low** **Dependencies: none**	**Importance: critical**

R18	Manage the allocation of network resources	
The network should allow the platform to allocate network resources to specific tasks. For example, the platform should be allowed to dictate that a particular distributed application be allocated 5 Gbps between two general-purpose cores (located either at a server or inside a network-packet-processing unit). This is necessary to enable the platform to ensure a certain level of performance to each task. State-of-the-art focus: *network management, heterogeneity, performance*.		
Task: T4.2	**Innovation: medium** **Dependencies: R6**	**Importance: critical**

R19	Provide reporting of network-resource usage	
The network should report to the platform how much each network resource is currently used. For example, the platform should know that a particular in-network processing unit is idle. This is necessary to enable the platform to reuse recently freed network resources. State-of-the-art focus: *network management*.		
Task: T4.2	**Innovation: low** **Dependencies: none**	**Importance: critical**

3.4 Storage

The goal of the storage component is to provide different kinds of storage with different performance profiles. Instead of reserving physical devices per reservation, HARNESS will map several reservations to the same device to lower the cost to the user. The challenge will be the enforcement of the guaranteed performance and the optimal mapping of reservations to devices.

The final set of requirements relate to heterogeneous storage in the HARNESS platform.

R20	**Implement resource reservations that are abstract from physical devices**	
An abstract description of the required storage resources should be specified, rather than a list of physical devices. For example, users can decide between four different storage classes: random access, sequential access, best effort, and cold storage. They must only provide expected IOPS. Additionally, the platform must describe durability requirements. State-of-the-art focus: *heterogeneity, resource management*.		
Task: T5.2, T5.3	**Innovation: medium** **Dependencies: R6**	**Importance: critical**

R21	**Implement time-dependent anticipated usage pattern**	
It should be possible to describe how the storage usage pattern will change over time. This will allow the scheduler to make better use of physical resources by collocating reservations with complementary usage patterns. State-of-the-art focus: *resource management, performance*.		
Task: T5.3	**Innovation: high** **Dependencies: none**	**Importance: moderate**

R22	**Implement virtual storage classes**	
Storage devices should be virtualised. By placing several reservations from potentially different storage classes, we can make more efficient use of resources. This contrasts with today's practice, where every physical device is used for just one storage class. State-of-the-art focus: *resource management*.		
Task: T5.3	**Innovation: high** **Dependencies: none**	**Importance: critical**

R23	**Develop performance models**	
Performance models of the different kinds of storage devices must be developed. For example, the model should describe how they perform under random, sequential, and concurrent access. The performance models will be used by the scheduler. State-of-the-art focus: *performance*.		
Task: T5.1	**Innovation: medium** **Dependencies: none**	**Importance: critical**

R24	**Provide run-time summary of available resources to the platform**
For scheduling decisions, the storage layer must provide summaries of the available storage resources, both statically and dynamically. State-of-the-art focus: *resource management*.	
Task: T6.2	**Innovation: low** **Importance: critical** **Dependencies: none**

4 Relevant EC FP Projects

This chapter summarises relevant EC-funded Framework Programme (FP) projects with specific focus on the main research areas within HARNESS: cloud platforms, computation, communication, and storage. We consider each research area in turn, summarising the goals and/or achievements of each project, indicating whether HARNESS partners are or were involved in the project, and indicate how they could contribute to the general requirements presented in Chapter 3.

4.1 Cloud Platforms

In this section we summarise the contributions of relevant FP projects that provide cloud PaaS and IaaS environments.

Project	Relevant Focus	Period	HARNESS Partners
4CaaSt	Complex multi-tier applications	2010–2013	SAP, ZIB
ADVANCE	Performance feedback for concurrent programs on heterogeneous resources	2010–2013	SAP
Cloud-TM	Distributed transactional memory	2010–2013	
ConPaaS (Contrail)	Integrated run-time environment for elastic cloud applications	2010–2013	UR1[1]
CumuloNimbo	Scalability of transactional consistency	2010–2013	SAP
REMICS	Reuse and migration of legacy applications	2010–2013	
SRT-15	Content-based routing on hybrid cloud computing infrastructures	2010–2013	SAP
Cloud4SOA	Interoperability and vendor lock-in	2010–2013	
Contrail	Cloud federations	2010–2013	UR1[1]
InterSECTION	Security of complex networked systems	2008–2010	
OPTIMIS	Optimising the full cloud service life cycle	2010–2013	
RESERVOIR	Reference architecture for next generation IaaS clouds	2008–2011	SAP
SLA@SOI	SLAs	2008–2011	SAP
TClouds	Security and privacy of distributed systems	2010–2013	

4.1.1 4CaaSt

4CaaSt [1] aims to build an advanced PaaS platform with a special focus on complex multi-tier applications. Applications will be described by a *blueprint*, which is a declarative language for describing the interaction of different services. Customers can use preexisting blueprints from a marketplace provided by 4CaaSt or design their own applications. The platform will automatically optimise performance and scale the deployment to support SLAs.

[1] UR1's team leader was previously a member of the Contrail project and the main architect of ConPaaS.

A smaller part of the project tries to make existing telecommunication APIs available to cloud applications. Two large telecommunication companies are members of the project and support this effort.

4.1.2 ADVANCE

ADVANCE [2] aims to use statistical performance feedback to dynamically adapt concurrent programs to heterogeneous resources. The core of the project is to create a cost-centric directed virtualisation layer for hardware present in a system. The goal of this costing approach is to make access to devices transparent to the programmer while maintaining requirements of performance and power consumption. The project looks at language extensions to capture performance information and aims to create a "write-once, deploy-anywhere" programming paradigm using a virtualisation layer to transform code into targeted implementations. Other work within the project creates resource prediction models from user-supplied information and program feedback, looks at compilation costing of the program to target the virtualisation layer, and uses program cost models and run-time feedback to place execution.

4.1.3 Cloud-TM

Cloud-TM [17] aims to define a novel programming paradigm to facilitate the development and administration of cloud applications. It develops a *distributed transactional memory* middleware that relieves programmers of the burden of coding for distribution, persistence and fault tolerance, letting them focus on delivering differentiating business value. Further, the Cloud-TM platform aims to minimise the operational costs of cloud applications, pursuing optimal efficiency via autonomic resource provisioning and pervasive self-tuning schemes.

4.1.4 ConPaaS

ConPaaS [20, 63] is a run-time environment for hosting applications in the cloud. It aims to offer the full power of the cloud to application developers while shielding them from its associated complexity. ConPaaS is designed to host both high-performance scientific applications and online Web applications. It automates the entire life cycle of an application, including collaborative development, deployment, performance monitoring, and automatic scaling. Finally, it runs on a variety of public and private clouds, and is easily extensible. This allows developers to focus their attention on application-specific concerns rather than on cloud-specific details. Services can be easily composed and are elastic.

The main architect of the ConPaaS system is now working at the HARNESS partner UR1. We plan to use ConPaaS as the basis for further developments in the HARNESS project.

4.1.5 CumuloNimbo

CumuloNimbo [25] aims to address "vital" issues in future cloud computing platforms by developing a highly scalable PaaS. The platform will provide full transactional guarantees without constraining how applications are developed. The project addresses the scalability and dependability of service platforms with the goal of achieving high scalability (100+ nodes) with strong transactional consistency and ease of programming. Research activities focus on five main areas: strong consistency; update scalability; dependability; elasticity; and low latency and high throughput. The project aims to adopt a "holistic" approach, by considering all of the application server, database server, file system and storage tiers, as

well as distribution and transactional coordination. Currently there are no software outputs from the project and no details on plans for this.

4.1.6 REMICS

REMICS [65] proposes methodologies for the migration of legacy systems to service clouds by providing a model-driven methodology and tools. This will be achieved by driving the standardisation work in the *Object Management Group* (OMG) and by providing project results under open-source licenses.

4.1.7 SRT-15

SRT-15 [71] combines complex event processing, content-based routing and dependability, and privacy technologies to produce a distributed service platform that allows enterprise applications to interact. This platform aims to facilitate the processing of large volumes of data (coming to and from a variety of heterogeneous distributed enterprise services) on a hybrid cloud computing infrastructure. Developed under the SRT-15 project, StreamLine3G [72] is a scalable, elastic and fault tolerant event processing/data streaming engine inspired by MapReduce [26]. It aims to overcome the high latency of the batch processing involved in the store and process method. It is publicly available as of May 2012 and has been used by a number of companies. PASC is a Java library that prevents the propagation of errors caused by data corruption in processes of a distributed system [30]. Within the context of the SRT-15 project, the Italian SME Epsilon has developed an application that provides dependable QoS monitoring of enterprise services running in the cloud. It is being tested and validated with a smart metering case study that implements remote monitoring of power consumption in a smart grid environment.

4.1.8 Cloud4SOA

Cloud4SOA [18] addresses the issue of interoperability and vendor lock-in with respect to cloud computing platforms. The project aims to interconnect PaaS offerings in a way that allows for data to be managed and migrated across platforms that use different data models and APIs. Interconnection is restricted to platforms that share the same background technology, such as Java-to-Java or PHP-to-PHP, and is achieved though platform-specific adaptors that act as proxies between the Cloud4SOA system and the various provider platforms. Developers wanting to deploy their application on the cloud can take advantage of the system's algorithm for matching application profiles to semantic descriptions of available PaaS offerings. The health and performance of distributed applications can also be monitored. Cloud4SOA is scheduled for a beta release in October 2013 and will be available on GitHub. Developers can create new adaptors and customise existing adaptors to suit their needs.

4.1.9 Contrail

Contrail [21] vertically integrates an open-source distributed operating system for autonomous resource management in IaaS environments with high-level services and run-time environments as foundations for PaaS. The ConPaaS system mentioned above is one outcome of the Contrail project. The main achievement will be a tightly integrated software stack in open source including a comprehensive set of system, run-time and high-level services providing standardised interfaces for supporting cooperation and resource sharing over cloud federations. Contrail will address key technological challenges in

existing commercial and academic clouds: the lack of standardised rich and stable interfaces; limited trust from customers; and relatively poor QoS guarantees regarding the performance and availability of cloud resources. Addressing these important issues is fundamental to support large user communities formed of individual citizens and/or organisations relying on cloud resources for their mission-critical applications. The main contribution of Contrail is an integrated approach to virtualisation, offering IaaS services for IaaS cloud federation, and PaaS. It will aim at equalling current commercial clouds, and surpassing them in a number of selected key domains to facilitate industrial uptake of federated cloud computing.

4.1.10 InterSECTION

InterSECTION [45] aimed to develop algorithms and techniques that enhance the security of complex networked systems. The project was focused on distributed systems, interconnected over heterogeneous networks. A representative example would be a cloud that consists of multiple data centres, interconnected over the Internet. One aspect that makes such a system particularly challenging to secure is that the networks that interconnect its components are themselves vulnerable to a large variety of attacks. The project addresses this challenge through new algorithms and techniques for network monitoring, anomaly and intrusion detection, malware detection and analysis, and network status visualisation. InterSECTION is relevant to HARNESS in the sense that it could help secure any cloud infrastructure. Its focus, however, is on the security of heterogeneous networks, not on heterogeneous resource management.

4.1.11 OPTIMIS

OPTIMIS [60] provides an architectural framework and development toolkit for optimising the full cloud service life cycle, service construction, deployment and operation in the cloud. The toolkit measures what it calls "key cloud deployment variables"—trust, risk, eco-efficiency, and cost—that can then used to allow cloud service providers to make decisions on deployment, while also enabling deployment on all cloud variants (private, hybrid, federated and multi). The OPTIMIS toolkit [61] is released as an open-source download. A key feature is a monitoring infrastructure that gathers real-time information on physical and virtual resources. Another component gathers information on trust, risk, eco-efficiency and cost for both infrastructure providers and service providers. Other components manage service construction, deployment and optimisation, image creation, data management across cloud environments, VM management, including elasticity management, and handling fault tolerance of virtual machine instances.

4.1.12 RESERVOIR

RESERVOIR [66] defined a reference architecture designed to meet next generation IaaS cloud requirements, such as guaranteeing SLAs, automating service deployment, provisioning and scalability. The architecture is independent of different virtualisation technologies, supports federation across public, private and hybrid clouds and makes use of open/standard specifications. The CLAUDIA platform is a spin-out technology from RESERVOIR. CLAUDIA manages services as a whole, controlling the configuration of multiple VMs, virtual networks and storage. It has a number of plug-ins that orchestrate

virtual resource allocation using a virtual infrastructure manager, such as OpenNebula [59], Eucalyptus [28] or vSphere [75], for deploying services to private clouds with support for deployment to public clouds such as Amazon, Flexiscale [32] and others. This is available as a software download and is also integrated with OpenNebula. The RESERVOIR project made use of the Lattice monitoring framework [47] for monitoring of components. Lattice can be used to build a bespoke monitoring subsystem and so may be of interest to HARNESS for monitoring components. It is distributed under a *Lesser General Public License* (LGPL) license and available for download.

4.1.13 SLA@SOI

SLA@SOI [69] aimed to support a service-oriented economy, where IT-based services can be flexibly traded as economic goods, that is, under well-defined and dependable conditions and with clearly associated costs. The idea is that this would allow for dynamic value networks that can be flexibly instantiated, thus driving innovation and competitiveness. SLA@SOI attempted to provide three major benefits to the provisioning of services: *predictability and dependability*, such that the quality characteristics of services can be enforced at run time; *transparent SLA management*, such that the SLAs defining the conditions under which services are provided/consumed can be transparently managed across the whole business and IT stack; and *automation*, such that the process of negotiating SLAs and provisioning, delivery and monitoring of services can be automated, allowing for highly dynamic and scalable service consumption.

4.1.14 TClouds

TClouds [73] aims to develop a framework that enhances the security and privacy of distributed systems. The project is focused on systems whose components potentially belong to different administrative entities. A representative example would be a multi-cloud, where each constituent cloud is located in a different country, hence subject to different security and privacy regulations. The challenge is how to interconnect these components in a way that allows us to reason about the security and privacy, not only of each separate component, but of the distributed system as a whole. A key goal of the project is to identify and address the legal and business implications of such interconnections. TClouds is relevant to HARNESS in the sense that it could be used to interconnect one HARNESS clouds. Its focus, however, is on the security and privacy boundaries between different administrative domains, not on heterogeneous resource management.

4.1.15 Relation to HARNESS General Requirements

The projects described above target various complementary forms of PaaS and IaaS environments. Several address questions such as resource discovery, scheduling, SLA negotiation and monitoring. Some address various security concerns in a cloud environment. Together, they provide a conceptual foundation that the HARNESS project can exploit wherever possible. However, none of them explicitly addresses the specific challenge of efficiently hosting applications in heterogeneous clouds.

The following two tables indicate which projects could contribute to the HARNESS requirements described in Section 3.1.

	4CaaSt	ADVANCE	Cloud-TM	ConPaaS	CumuloNimbo	REMICS	SRT-15
R1: Provide an execution platform for using heterogeneous resources	•	•		•	•		
R2: Provide support for flexible applications		•					
R3: Provide an application description language	•			•			
R4: Provide an SLA description language							
R5: Implement application performance model generation		•		•			
R6: Implement resource co-allocation scheduler (per application)							
R7: Implement resource co-allocation scheduler (multi-application)							
R8: Provide design patterns for flexible applications	•					•	

	Cloud4SOA	Contrail	InterSECTION	OPTIMIS	RESERVOIR	SLA@SOA	TClouds
R1: Provide an execution platform for using heterogeneous resources		•		•			
R2: Provide support for flexible applications							
R3: Provide an application description language	•						
R4: Provide an SLA description language						•	
R5: Implement application performance model generation		•					
R6: Implement resource co-allocation scheduler (per application)					•		
R7: Implement resource co-allocation scheduler (multi-application)					•		
R8: Provide design patterns for flexible applications	•						

4.2 Computation

In this section we summarise the contributions of relevant FP projects that provide the basis for exploiting heterogeneous computational resources in HARNESS.

Project	Relevant Focus	Period	HARNESS Partners
MORPHEUS	Heterogeneous multi-core platform with reconfigurable fabric	2006–2008	
HARTES	Programming methodology and compilation design-flow for heterogeneous platforms	2006–2009	IMP
PEPPHER	Unified programming framework for heterogeneous platforms	2010–2012	
REFLECT	Aspect-oriented design-flow for capturing non-functional concerns	2010–2012	IMP
FASTER	Advanced mapping and verification schemes for reconfigurable hardware	2011–2013	MAX, IMP

4.2.1 MORPHEUS

MORPHEUS [53] provided a heterogeneous platform for embedded devices. The project designed an *multiprocessor system-on-chip* (MPSoC) that included an ARM9 general-purpose processor that coordinated three types of reconfigurable units acting as co-processors and operating under different levels of granularity. In addition, the MORPHEUS project provided a programming model that viewed its heterogeneous platform as a single virtual processor, and in which reconfigurable accelerators acted as customised functional units to extend the instruction set architecture.

Developers, however, were required to manually partition the application, and map each partition to suitable accelerators. By carefully deciding which functions would be offloaded to reconfigurable units and which would be implemented in software, one could expect an increase in performance, flexibility and reuse capability. Some of the techniques used in MORPHEUS for offloading parts of the computation to multi-core reconfigurable accelerators acting as co-processors will prove useful in the context of HARNESS to exploit the underlying cloud infrastructure.

4.2.2 HARTES

HARTES [13] provided a compilation approach for embedded heterogeneous multi-core platforms, in which the complexity of the underlying hardware, including CPU, DSP and FPGA processing elements, was abstracted from developers.

The HARTES compilation framework provided developers a familiar programming paradigm, operating on sequential C-language descriptions, with all the steps of mapping the application to the multi-core platform fully automated to achieve real-time constraints. To achieve this purpose, the HARTES compilation framework included a number of compilation engines, such as a task transformation performing code and loop transformations that could be customised for a particular processing element, a task partitioning engine that restructured the application to exploit different task granularities, and a mapping selection process [49] that assigned existing tasks to available processing elements. In addi-

tion, the compiler framework could integrate different compiler back ends specific to each processing element.

The tool chain also supported an extensive pragma annotation facility inspired by OpenMP [58], which allowed developers to control parallelism, partitioning and mapping selection, thereby supporting a semi-automatic design flow. These compilation engines, the annotation facility, and the programming model that hides away heterogeneity, will be part of the technology that allows programmers to develop cloud-based applications in HARNESS.

4.2.3 PEPPHER

PEPPHER [12] developed a unified framework for programming and optimising applications for heterogeneous multi-core CPUs with GPGPU-type accelerators. In particular, it focused on how to enable performance and portability while minimising programming effort, rather than requiring applications or algorithms to be repeatedly re-implemented for each generation of hardware.

The key idea behind PEPPHER was to maintain multiple implementation variants of performance-critical components of the application and schedule these efficiently either dynamically or statically across the available CPU and GPGPU resources. Implementation variants are developed manually, through compilation support, by composition, or by auto-tuning.

While PEPPHER focused on a subset of the heterogeneous computational resources addressed by HARNESS, the results could be leveraged by HARNESS to reduce programming effort required for low-level resource utilisation.

4.2.4 REFLECT

REFLECT [64] focused on compilation techniques aimed at increasing design productivity and maintainability, and assisting developers in generating and exploring alternative and competing hardware/software designs when mapped onto heterogeneous architectures. With the REFLECT tool chain, developers were able to decouple functional and non-functional concerns. In particular, functional concerns capture algorithmic behaviour and are implemented using traditional languages such as C. Non-functional concerns, on the other hand, deal with desired qualities of the application, such as performance and resource efficiency. For this purpose, a new aspect-oriented language called LARA [15] was developed to capture non-functional concerns. The weaving process automatically combines application sources and LARA aspects to derive an augmented application at compile time that satisfies both functional and non-functional concerns.

For HARNESS, this aspect-oriented approach will bring two benefits to cloud providers offering acceleration services. First, as functional and non-functional concerns are maintained independently, they can be updated, removed or added more easily, thus significantly improving the maintainability and portability of computational services across multiple cloud infrastructures. Second, as non-functional concerns can be codified, this approach allows the development of compilation strategies that can be reused and applied to different applications and target architectures, thus increasing design productivity.

4.2.5 FASTER

FASTER [29] addresses the development of applications for partially reconfigurable hardware. The project adopts a holistic approach, developing a complete methodology for implementing and verifying a partially reconfigurable system on a platform that combines processors with reconfigurable devices.

The FASTER tool chain aims to support a programmer in moving from a static version of an application (possibly software only) to a dynamic, reconfigurable hardware/software version. FASTER will include a run-time scheduler that manages the reconfiguration of device resources during application execution—a challenge that has similarities with those that must be addressed by the HARNESS platform, but on a much more restricted scale.

We anticipate that the methodology and tool flow developed by FASTER could be used to program reconfigurable computational resources that in the HARNESS platform. HARNESS partners IMP and MAX are partners in FASTER and will help to ensure coordination between the two projects.

4.2.6 Relation to HARNESS General Requirements

The projects described above provide insight into programming models, compilation design flows, run-time resource management, and how to efficiently exploit reconfigurable devices in a heterogeneous environment. Three of these projects (HARTES, REFLECT and FASTER) have partners that are now involved in HARNESS, which will facilitate technology transfer among them. One of the key contributions from these projects will be the LARA aspect-oriented work [15] developed in REFLECT, which will be adapted to generate design variant implementations in the context of the HARNESS platform.

The following table indicates which projects could contribute to the HARNESS requirements described in Section 3.2.

	FASTER	HARTES	MORPHEUS	PEPPHER	REFLECT
R9: Develop a heterogeneous computational resource discovery protocol		•			
R10: Develop a compiler infrastructure to derive multi-target design variants		•		•	•
R11: Provide characterisation of heterogeneous applications		•			
R12: Enable shared use of heterogeneous computational resources	•		•	•	
R13: Provide monitoring data to support decision making at the cloud					
R14: Optimise the use of resources according to optimisation goals within constraints provided by the cloud platform layer	•				

4.3 Communication

In this section we summarise the contributions of relevant FP projects related to the communication aspect of the HARNESS project.

Project	Relevant Focus	Period	HARNESS Partners
BonFIRE	Multi-site cloud infrastructure for IoT	2010–2013	SAP
GEYSERS	Networking architecture for optical and IT resources	2010–2013	SAP
FI-WARE	Internet architecture focused on QoS and security guarantees	2011–2014	SAP
NOVI	Access to and management of federated resources in support of the Future Internet	2010–2013	

4.3.1 BonFIRE

BonFIRE [14] provides a state-of-the art multi-site cloud facility for applications, services and systems research. The facility gives researchers access to large-scale virtualised computational, communication, and storage resources with high level control and monitoring services for detailed experimentation of their systems and applications. The facility allows the evaluation of cross-cutting effects of converged service and network infrastructures. Specific features of BonFIRE allow the reservation of large-scale computational capacity with options for exclusive access to physical hosts and control over VM placement within a test bed. VM images can be customised for a specific number of CPU cores and RAM sizes with options for adding more storage and levels of persistence. In terms of networking, network topology can be fully customised, so various networking configurations are possible. Bandwidth, latency and loss rates can be statically or dynamically configured with further options to set background traffic levels. BonFIRE extends Zabblix [78] to create a custom monitoring framework that allows fine-grained monitoring of both virtual and physical resources. Another interesting feature is the ability to subscribe to notifications regarding state and resource changes.

4.3.2 GEYSERS

GEYSERS [34] aims to create an architecture for the provisioning of optical network and IT resources, by virtualising both traditional resources and optical networking resources. Two key components are required to enable this functionality: the Logical Infrastructure Composition Layer (LICL) and the Network Control Plane (NCP). The LICL controls deployment of virtual resources upon the physical network and IT infrastructures layers. It enables application service infrastructures to interconnect logical resources based on the requirements of virtual networking infrastructure operators while also handling monitoring of the physical layer. The NCP is responsible for mapping the application requirements and managing the logical infrastructure created by the LICL. Each independent logical infrastructure is managed by one NCP that is responsible for the dynamic provisioning, monitoring and recovery functions. Currently, there is no software released for open source consumption but this is planned for mid-2013.

4.3.3 FI-WARE

FI-WARE [31] aims to develop a new Internet architecture that offers QoS and security guarantees, as well as extensibility. The latter property relies on "generic enablers", which are pieces of functionality deployed at key network points that can be combined to offer a new service. In some sense, the FI-WARE project is about turning the Internet into a massive cloud, and about providing the APIs through which end users and operators will be able to program this cloud with new functionality. The HARNESS context is a more modest one, as we do not aim to redesign the Internet, just the resource management of clouds and data centres. However, the two projects could potentially benefit from each other: If we view the Internet as a cloud, then it is a heterogeneous cloud, so our ideas of managing heterogeneous resources apply. At the same time, we could inherit ideas on how to provide QoS and security guarantees over a best-effort network.

4.3.4 NOVI

NOVI [56] shares the same goal with FI-WARE (summarised above): turn the Internet into a massive cloud and provide APIs through which end-users and operators can manage this cloud. Instead of FI-WARE's "generic enablers," NOVI relies on "slices" of the network resources allocated to different applications and users. One focus area of the NOVI project is the incorporation of Semantic Web concepts. Another focus area is the "federation" of data, control, monitoring and resource management across different administrative entities. HARNESS can certainly benefit from the resulting ideas, although our context is more modest—one cloud with a wide variety of heterogeneous resources.

4.3.5 Relation to HARNESS General Requirements

We have identified two kinds of related projects. The first kind (BonFIRE and GEYSERS) provide ways to allocate network resources to applications, as well as to monitor and report usage of network resources. We can benefit from these by reusing some of their allocation, monitoring, and reporting interfaces and techniques. The second kind (FI-WARE and NOVI) provide ways to extend the Internet architecture with new functionality. We can benefit from these by learning from their experiences in network programming (although programming the Internet is a much more ambitious goal than ours, which is to program switches and middleboxes in a data centre).

The similarity between these projects and ours is that, at a high level, we all provide interfaces and techniques for the management and/or programming of network (and other) resources. BonFIRE and GEYSERS, in particular, provide interfaces and techniques for allocating, monitoring, and reporting on the usage of network resources. We will try to benefit from them as much as possible, because our goal is not to innovate in these particular aspects (resource allocation, monitoring, and reporting).

On the other hand, the communication part of the HARNESS project has a distinct angle that is not covered by any existing project: a cloud that consists of heterogeneous and programmable network devices (e.g., OpenFlow switches [33] and software middleboxes). The HARNESS project must understand how to best leverage these in order to improve application performance. For instance, the performance of MapReduce applications can benefit significantly from in-network aggregation [24]. Hence, the HARNESS platform tries to allocate to each such application, not only end servers that perform mapping and reducing, but also network middleboxes that perform in-network aggregation. The

difficult questions that we aim to answer are: what are the "right" interfaces for programming network devices and how can we prove useful properties for them, such as that a network device performing in-network aggregation will not introduce more than a certain amount of latency. FI-WARE and NOVI can help us to answer the first question, although their level of programming is quite different from ours, so it is unlikely that we can use their specific interfaces.

The following table indicates which projects could contribute to the HARNESS requirements described in Section 3.3.

	BonFIRE	GEYSERS	FI-WARE	NOVI
R15: Enable programming of network processing units			•	•
R16: Implement verification of properties of network processing units				
R17: Provide a description of network resources	•	•		
R18: Manage the allocation of network resources	•	•		
R19: Provide reporting of network-resource usage	•	•		

4.4 Storage

In this section we summarise the contributions of relevant FP projects that explicitly address storage. While many of the projects described above also contain some aspect of storage, it is not their major focus. For the purposes of this report, then, the three of interest are Contrail (4.1.9), RESERVOIR (4.1.12), and VISION Cloud (4.4.1).

Project	Relevant Focus	Period	HARNESS Partners
Contrail	Cloud federations	2010–2013	UR1[1]
RESERVOIR	Reference architecture for next generation IaaS clouds	2008–2011	SAP
VISION Cloud	Managing and accessing large data volumes in the cloud	2010–2013	SAP

4.4.1 VISION Cloud

VISION Cloud [74] focuses on the challenge of processing and managing large data volumes in the cloud. VISION Cloud aims to enable the provision of data and storage services across administration boundaries while maintaining quality of service and security guarantees. A main theme of VISION Cloud is "computational storage" whereby execution is performed local to the data as much as possible. The project also aims to create access to data, not through conventional targeting of the underlying storage containers, but rather through rich meta-data and relationships via topology creation. The primary deliverable from VISION Cloud (due September 2013) is an infrastructure demonstrating computational storage and content-specific access to data. The two main uses cases are health care, allowing access to personal data based on context and need, and telecommunications media streaming.

4.4.2 Relation to HARNESS General Requirements

All major platform projects contain a storage component. But the number of projects solely focused on storage is still small. We expect a large number of storage projects in the next FP calls with a focus on so-called Big Data. Similar to the platform projects, they will also cover large parts of the cloud software stack, but they will usually target a specific application with a heavy focus on machine learning and decision making.

VISION Cloud is one of the early storage and Big Data projects. They concentrate on high-level abstractions for data access and computational storage. Heterogeneous storage devices are not an identified topic, as it is for HARNESS.

The following table indicates which projects could contribute to the HARNESS requirements described in Section 3.4.

	Contrail	*RESERVOIR*	*VISION Cloud*
R20: Implement resource reservations that are abstract from physical devices	•	•	•
R21: Implement time-dependent anticipated usage pattern			
R22: Implement virtual storage classes	•	•	•
R23: Develop performance models			
R24: Provide run-time summary of available resources to the platform			

5 Conclusions

This report presents the general requirements of the HARNESS project. HARNESS will develop an enhanced cloud platform that utilises a collection of advanced, heterogeneous technologies. To realise such a platform, a number of technical objectives must be met (Section 1.4).

We have evaluated the current state of the art in cloud computing (Chapter 2) and found that the lack of fully integrated heterogeneous hardware into current cloud platforms leads to limitations, particularly in performance and scalability. The requirements of the project address these issues.

We have assembled a list of general requirements for the project (Chapter 3) . These requirements are unique to the project and its specific objectives and, as such, most will require a high level of innovation. When these requirements are met, the resulting cloud platform will present a significant advancement in cloud computing.

Finally, we have performed a comprehensive analysis of ongoing FP projects relevant to HARNESS (Chapter 4). While they will provide valuable insight as the project progresses, none explicitly addresses the challenges of exploiting heterogeneous hardware.

Bibliography

[1] 4CaaSt Project. Available at http://4caast.morfeo-project.org/.

[2] Advance project. Available at http://www.project-advance.eu/.

[3] M. Al-Fares, A. Loukissas, and A. Vahdat. A Scalable, Commodity Data Center Network Architecture. In *Proceedings of the ACM SIGCOMM Conference on Data Communication*, 2008.

[4] J. Allspaw. *The art of capacity planning: Scaling Web resources*. O'Reilly Media, 2009.

[5] Amazon, Inc. Cluster GPU Instance. Available at http://aws.amazon.com/hpc-applications/.

[6] Amazon, Inc. DynamoDB. Available at http://aws.amazon.com/dynamodb/.

[7] Amazon, Inc. EC2 Instance Types. Available at http://aws.amazon.com/ec2/instance-types.

[8] Amazon, Inc. Elastic Beanstalk. Available at http://aws.amazon.com/elasticbeanstalk/.

[9] Amazon, Inc. Elastic Block Store. Available at http://aws.amazon.com/ebs/.

[10] Amazon, Inc. Relational Database Service. Avaialble at http://aws.amazon.com/rds/.

[11] Apache. Hive. Available at http://hive.apache.org/.

[12] S. Benkner, S. Pllana, J. Traff, P. Tsigas, U. Dolinsky, C. Augonnet, B. Bachmayer, C. Kessler, D. Moloney, and V. Osipov. PEPPHER: Efficient and productive usage of hybrid computing systems. *IEEE Micro*, 31(5):28–41, 2011.

[13] K. Bertels, V. Sima, Y. Yankova, G. Kuzmanov, W. Luk, J. Coutinho, F. Ferrandi, C. Pilato, M. Lattuada, D. Sciuto, and A. Michelotti. Hartes: Hardware-software codesign for heterogeneous multicore platforms. *IEEE Micro*, 30(5):88–97, 2010.

[14] BonFIRE Project. Available at http://www.bonfire-project.eu/.

[15] J. Cardoso, T. Carvalho, J. Coutinho, W. Luk, R. Nobre, P. Diniz, and Z. Petrov. LARA: An aspect-oriented programming language for embedded systems. In *Proceedings of the 11th Annual International Conference on Aspect-Oriented Software Development*, pages 179–190, 2012.

[16] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. BigTable: A distributed storage system for structured data. In *Proceedings of the Symposium on Operating System Design and Implementation*, Nov. 2006.

[17] Cloud-TM Project. Available at http://www.cloudtm.eu/.

[18] Cloud4SOA Project. Available at http://www.cloud4soa.eu/.

[19] CloudFoundry. Available at http://www.cloudfoundry.com/.

[20] ConPaaS Project. Available at http://www.conpaas.eu/.

[21] Contrail Project. Available at http://www.contrail-project.eu/.

[22] P. Costa, A. Donnelly, G. O'Shea, and A. Rowstron. CamCube: A key-based data center. Technical Report TR-2010-74, Microsoft Research, 2010.

[23] P. Costa, A. Donnelly, A. Rowstron, and G. O'Shea. Camdoop: Exploiting in-network aggregation for big data applications. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation*, 2012.

[24] P. Costa, M. Migliavacca, P. Pietzuch, and A. L. Wolf. NaaS: Network-as-a-service in the cloud. In *Proceedings of the 2nd USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services*, Apr. 2012.

[25] CumuloNimbo Project. Available at http://www.cumulonimbo.eu/.

[26] J. Dean and S. Ghemawat. MapReduce: A flexible data processing tool. *Communications of the ACM*, 53(1), 2010.

[27] M. Dobrescu, N. Egi, K. Argyraki, B.-G. Chun, K. Fall, G. Iannaccone, A. Knies, M. Manesh, and S. Ratnasamy. RouteBricks: Exploiting parallelism to scale software routers. In *Proceedings of the ACM Symposium on Operating Systems Principles*, 2009.

[28] Eucalyptus Project. Available at http://www.eucalyptus.com/.

[29] FASTER Project. Available at http://www.fp7-faster.eu/.

[30] D. G. Ferro. PASC: Practical arbitrary state corruptions. Available at https://github.com/dgomezferro/pasc/.

[31] FI-WARE Project. Available at http://www.fi-ware.eu/.

[32] Flexiant. Flexiscale. Available at http://www.flexiscale.com/.

[33] O. N. Foundation. OpenFlow. Available at http://www.openflow.org/.

[34] GEYSERS Project. Available at http://www.geysers.eu/.

[35] G. Giunta, R. Montella, G. Agrillo, and G. Coviello. A GPGPU transparent virtualization component for high performance computing clouds. In *Proceedings of the 16th International Euro-Par Conference on Parallel Processing*, number 6271 in Lecture Notes in Computer Science, pages 379–391. Springer-Verlag, 2010.

[36] Google, Inc. App Engine. Available at `https://developers.google.com/appengine/`.

[37] Google, Inc. Compute Engine. Available at `https://cloud.google.com/products/compute-engine/`.

[38] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. VL2: A scalable and flexible data center network. In *Proceedings of the ACM SIGCOMM Conference on Data Communication*, 2009.

[39] B. Grobauer, T. Walloschek, and E. Stocker. Understanding cloud computing vulnerabilities. *IEEE Security and Privacy*, 9(2):50–57, 2011.

[40] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu. BCube: A high performance, server-centric network architecture for modular data centers. In *Proceedings of the ACM SIGCOMM Conference on Data Communication*, 2009.

[41] C. Guo, H. Wu, K. Tan, L. Shiy, Y. Zhang, and S. Luz. DCell: A scalable and fault-tolerant network structure for data centers. In *Proceedings of the ACM SIGCOMM Conference on Data Communication*, 2008.

[42] V. Gupta, A. Gavrilovska, K. Schwan, H. Kharche, N. Tolia, V. Talwar, and P. Ranganathan. GViM: GPU-accelerated virtual machines. In *Proceedings of the 3rd ACM Workshop on System-Level Virtualization for High Performance Computing*, pages 17–24, 2009.

[43] S. Han, K. Jang, K. Park, and S. Moon. PacketShader: A GPU-accelerated software router. In *Proceedings of the ACM SIGCOMM Conference on Data Communication*, 2010.

[44] Heroku. Available at `http://www.heroku.com/`.

[45] InterSECTION Project. Available at `http://www.intersection-project.eu/`.

[46] K. Jang, S. Han, S. Han, S. Moon, and K. Park. SSLShader: Cheap SSL acceleration with commodity processors. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation*, 2011.

[47] The Lattice Monitoring Framework. Available at `http://clayfour.ee.ucl.ac.uk/lattice/`.

[48] M. D. Linderman, J. D. Collins, H. Wang, and T. H. Meng. Merge: A programming model for heterogeneous multi-core systems. In *Proceedings of the ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2008.

[49] W. Luk, J. Coutinho, T. Todman, Y. Lam, W. Osborne, K. Susanto, Q. Liu, and W. Wong. A high-level compilation toolchain for heterogeneous systems. In *IEEE International Conference on System-on-Chip*, pages 9–18, 2009.

[50] Y. Ma, S. Banerjee, S. Lu, and C. Estan. Leveraging parallelism for multi-dimensional packet classification on software routers. In *Proceedings of the ACM SIGMETRICS Conference*, 2010.

[51] M. Mao and M. Humphrey. Auto-scaling to minimize cost and meet application deadlines in cloud workflows. In *Proceedings of the IEEE International Conference on High Performance Computing, Networking, Storage and Analysis*, pages 1–12, 2011.

[52] Microsoft, Inc. Azure Services Platform. Available at `http://www.azure.net/`.

[53] MORPHEUS. Available at `http://www.morpheus.arces.unibo.it/`.

[54] MySQL Database. Available at `http://www.mysql.com/`.

[55] X. Niu, Q. Jin, W. Luk, Q. Liu, and O. Pell. Exploiting run-time reconfiguration in stencil computation. In *IEEE Conference on Field Programmable Logic and Applications*, pages 173–180, 2012.

[56] NOVI Project. Available at `http://www.fp7-novi.eu/`.

[57] NVIDIA, Inc. Introducing the GPU-accelerated cloud. Available at `http://www.nvidia.com/object/vdi-desktop-virtualization.html`.

[58] OpenMP. Available at `http://openmp.org/`.

[59] OpenNebula. Available at `http://www.opennebula.org/`.

[60] OPTIMIS. Available at `http://www.optimis-project.eu/`.

[61] OPTIMIS Toolkit. Available at `http://www.optimis-project.eu/downloads/`.

[62] Oracle Database. Available at `http://www.oracle.com/us/products/database/`.

[63] G. Pierre and C. Stratan. ConPaaS: A platform for hosting elastic cloud applications. *IEEE Internet Computing*, 16(5):88–92, September-October 2012.

[64] REFLECT Project. Available at `http://www.reflect-project.eu/`.

[65] REMICS Project. Available at `http://www.remics.eu/`.

[66] RESERVOIR Project. Available at `http://www.reservoir-fp7.eu/`.

[67] L. Schubert and K. Jeffery. Advances in clouds—Research in future cloud computing. Expert Group Report, European Commission, Information Society and Media, 2012.

[68] L. Shi, H. Chen, J. Sun, and K. Li. vCUDA: GPU-accelerated high-performance computing in virtual machines. *IEEE Transactions on Computers*, 61(6):804–816, 2012.

[69] SLA@SOI Project. Available at `http://www.sla-at-soi.eu/`.

[70] W. W. Smari, S. Fiore, and D. Hill. High performance computing and simulation: Architectures, systems, algorithms, technologies, services, and applications. *Concurrency and Computation: Practice and Experience*, 2013.

[71] SRT-15 Project. Available at `http://www.srt-15.eu/`.

[72] StreamMine3G. Available at `https://streammine3g.inf.tu-dresden.de/trac/`.

[73] TClouds Project. Available at `http://www.tclouds-project.eu/`.

[74] Vision Cloud Project. Available at `http://www.visioncloud.eu/`.

[75] VMware, Inc. vSphere. Available at `http://www.vmware.com/products/datacenter-virtualization/vsphere/`.

[76] J. R. Wernsing and G. Stitt. Elastic computing: A framework for transparent, portable, and adaptive multi-core heterogeneous computing. In *Proceedings of the ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems*, pages 115–124, 2010.

[77] Xen.org. PCI Passthrough. Available at `http://wiki.xen.org/wiki/Xen_PCI_Passthrough`.

[78] Zabbix. Available at `http://www.zabbix.com/`.

[79] Q. Zhang, L. Cheng, and R. Boutaba. Cloud computing: State-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1(1):7–18, 2010.