White Paper

December 2015

Heterogeneous Cloud Computing Platform

New Generation Cloud Computing Platform

Heterogeneous Cloud for Increased Performance and Greater Utility

Today's cloud platforms are missing out on the revolution in new hardware and network technologies for realising vastly richer computation, communication, and storage solutions. Technologies such as FPGAs and GPGPUs, programmable network routers, general-purpose middleboxes and solid-state disks bring new degrees of freedom to resource allocation and optimisation. However, their heterogeneity and complexity makes their integration with standard cloud frameworks a fundamental challenge.

The HARNESS project has developed a new generation cloud computing platform that integrates heterogeneous hardware and networking resources in order to provide vastly increased performance for a broader array of applications. With HARNESS, cloud providers can profitably manage specialised hardware and network technologies much as they do today's commodity resources, and software engineers can seamlessly integrate them into the design of their cloud-hosted applications.

The public cloud services provider market is projected to reach nearly $22 billion in 2015. The goal of HARNESS is to enable those providers to offer new levels of service to cloud applications while opening a new market to purveyors of specialised hardware and network technologies.

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

Introduction

Modern cloud computing technologies can vastly improve the flexibility and ease-of-use of application systems while simultaneously simplifying administration, reducing downtimes and maintenance costs. However, they do not meet the stringent requirements of several application domains, including scientific computing, business analytics and on-line machine learning.

Cloud computing is reshaping IT, 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.

Moreover, these data-centre infrastructures are built primarily to service distributed N-tier web-based applications that can easily scale horizontally in response to demand. This ignores the needs of many scientific and engineering applications whose fundamental algorithms and complex task-communication interdependencies must rely on a variety of non-commodity technologies to achieve a sufficient level of performance. However, advanced hardware and network technologies for improving computation, communication, and storage performance 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 cloud computing platform (IaaS/PaaS) model.

To better serve workloads that are currently not supported by modern-day data centres, we created HARNESS, an *enhanced cloud platform stack* that fully embraces heterogeneity, allowing the side-by-side deployment of commodity and specialised resources, such as FPGAs, GPGPUs, middleboxes, hybrid switches and SSDs, in data centre infrastructures.

With HARNESS, a flexible application may be deployed in many different ways over different types and numbers of resources, each option having its own cost, performance, and usage characteristics. In this context, applications express their needs to the HARNESS 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 machines or amounts of storage, to encompass the specific characteristic of specialised technologies.

These specialised technologies are virtualised into resources that can be managed and accessed at the platform level. The idea is to provide flexibility to the platform as to which, when, and how many resources are used, and to separate that concern from the low-level management of the concrete technology elements. Associated with the virtualised resources are policies that govern how allocation and optimisation decisions are made, as well as facilities to track their capacity, usage, and general availability.

HARNESS Resource Set

The HARNESS consortium focused on a set of specialised resources and management technologies that enhance heterogeneous cloud computing platforms by improving performance, security and cost-profiles of cloud-hosted applications.

The following is a list of compute, network and storage systems that the HARNESS project worked to expose as cloud resources:

Hardware Accelerators are specialised compute devices that run considerably faster than general-purposed CPUs for specific types of workload. This efficiency, often allowing speed-ups by orders of magnitude over the CPU, comes at a price of flexibility and ease of programming, requiring expertise to fully leverage the capabilities of these devices. Hardware accelerators considered in HARNESS include FPGAs, GPGPUs, and Intel PHIs.

Dataflow Engine Cluster is a specialised compute resource that comprises one or more Maxeler MPC-X chassis interconnected by Infiniband in a data centre. Each MPC-X chassis hosts a number of Dataflow Engines (DFEs) that can communicate with each other using a dedicated high-speed MaxRing interconnect, and are accessible to any CPU client machine via the Infiniband network. A DFE is a single compute device that contains an FPGA as the computation fabric and RAM for bulk storage.

NetAgg is a software middlebox platform that provides a service for aggregating data on path so that the amount of data exchanged between end hosts is minimised. Using NetAgg, application performance can be enhanced when scarce network bandwidth is available at the end hosts of a data centre.

Hybrid switch is a network device that has been designed to hold millions of rules to provide access control in a multi-tenancy environment. As with virtual memory, the device stores the flow table in a memory hierarchy, where a small amount of fast-access memory acts as a cache for a significantly larger amount of slower memory (DRAM).

Heterogeneous Storage is managed by a storage system based on XtreemFS, which offers storage volumes that fulfil QoS requirements for specific types of access, including sequential and random.

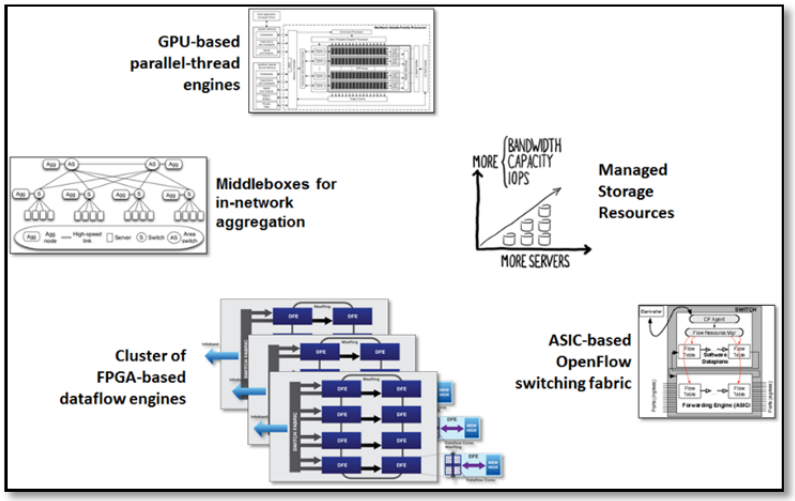

Figure 1. *Resources targeted by the HARNESS project.*

HARNESS Architecture

We developed a fully functional prototype of the HARNESS cloud platform, incorporating several technologies, including ConPaaS, OpenStack, Maxeler Dataflow Engines (FPGAs) and XtreemFS.

The HARNESS architecture is split into three main layers: the *Platform Layer,* which manages applications, the *Infrastructure Layer,* which is responsible for managing all cloud resources and making them available on demand, and the *Virtual Execution Layer*, where applications actually run.

The **Platform Layer** includes three key components:

- The *Frontend* (ConPaaS) is a Web server that provides a graphical interface where users can manage their applications.
- The *Director* (ConPaaS) is in charge of authenticating users and instantiating one Application Manager for each application instance submitted by a user.
- The *Application Manager* is in charge of controlling the life cycle of one particular application. This includes: building performance models for arbitrary applications, choosing the type and number of resources that an application may need, provisioning these resources from the Infrastructure Layer, deploying the application's code and data on provisioned resources, and collecting application-level feedback.

The **Infrastructure Layer** comprises the following components:

- The *Cross-Resource Scheduler (CRS)* is in charge of managing all resources available in the cloud, in particular servicing allocation requests for a group of heterogeneous resources, with optional placement constraints between them.
- *Infrastructure Resource Managers (IRMs)* are responsible for translating agnostic management requests from the CRS to specific requests for Local Resource Managers.
- *Local Resource Managers* handle specific types of resources: OpenStack (commodity compute and network resources), MaxelerOS orchestrator (DFE cluster), SHEPARD compute (PCIe hardware accelerators and multicore CPUs), and XtreemFS (distributed file system).

The **Virtual Execution Layer** is realised as a VM/container image, and contains daemons and APIs that allow the deployed application to transparently interface with provisioned cloud resources. Components in this layer include:

- The *ConPaaS agent* performs management actions on behalf of the Application Manager, including configuring provisioned resources, starting the application, and finally collecting application-level feedback during execution.
- The *Executive* is a SHEPARD daemon that selects the most appropriate compute resource for a given application task.
- The *XtreemFS client* is in charge of mounting XtreemFS volumes in the VMs and making them available as regular local directories.

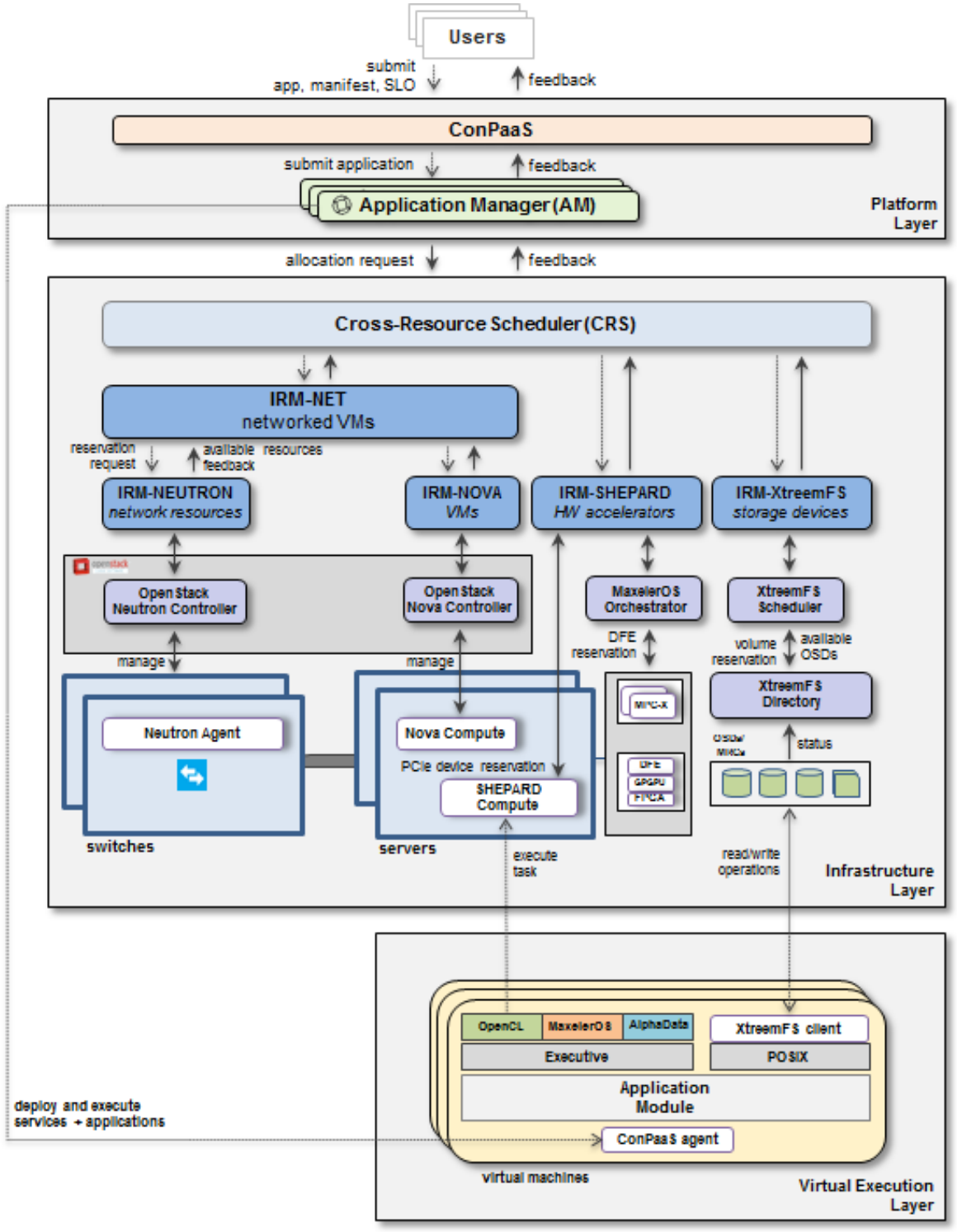

Figure 2. *The HARNESS architecture.*

Resource Management

HARNESS handles heterogeneity by abstracting cloud resources and supporting a hierarchical resource management architecture. This approach allows the cloud platform to be **resilient** to new forms of heterogeneity, so that new types of resources can be incorporated without having to redesign the entire system.

We designed a cloud computing platform that does not require its architecture to be re-engineered, nor its allocation algorithms rewritten, when introducing new types of resources, even when considering that each type of resource exposes different control mechanisms, different interfaces for acquiring feedback about availability and monitoring, and different semantics for expressing capacity and allocation requests.

Hierarchical Resource Management. The HARNESS resource management infrastructure is composed of multiple run-time resource managers that are combined hierarchically. In this organisation, the top levels of the hierarchy are the HARNESS resource managers that service requests using the HARNESS API. At the lower levels of the hierarchy we find vendor resource managers that handle specific devices. One of the responsibilities of the HARNESS resource managers is to translate agnostic requests defined by the HARNESS API into vendor-specific requests.

Supporting multiple hierarchical levels allows the system integrator to design an architecture using separation of concerns, such that each manager can deal with specific types of resources. In particular, a typical HARNESS platform deployment has a top-level manager that has complete knowledge about all the resources that are available in the system, but very little understanding of how to deal with any of these resources specifically.

Any management requests (such as reservation or monitoring feedback) are delegated to child managers that have a more direct understanding of the resource(s) in question. A child manager, can then handle a resource type directly or delegate the request further down to a more specific resource manager.

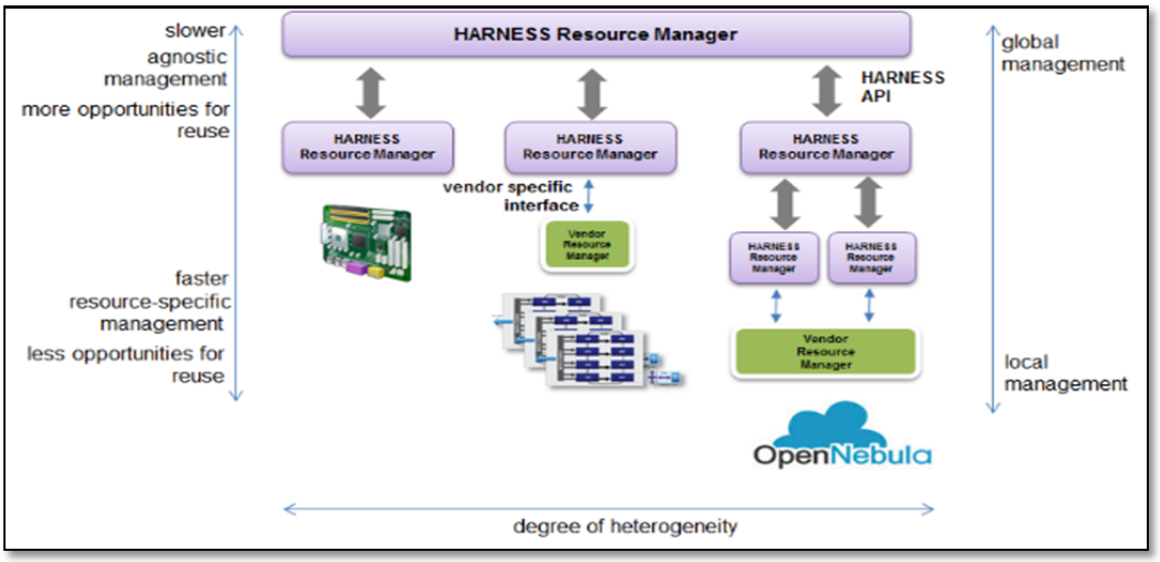

***Figure 3**. Extending the HARNESS platform.*

What we see is that at the top level of the hierarchy we obtain more agnostic management that can handle a wide range of resource types with a global view of the resources available in the data centre. As we go lower in the hierarchy, we have more localised and resource-specific management.

Every HARNESS resource manager implements the **HARNESS API**. This API mostly follows the RESTful style, where interactions between components are handled through HTTP requests. Resource managers supporting this API can handle one or more types of resources, from physical clusters of FPGAs to conceptual resources such as virtual links. We have grouped the HARNESS API functions into four categories:

- **Management.** Allows resource managers to connect to other HARNESS resource managers in a hierarchical organisation.
- **Resources.** Provides information about the state of available resources, and meta-information about allocation requests and resource capacity.
- **Reservations.** Allows resource instances to be created and destroyed.
- **Metrics and Logging.** Provides monitoring information about resources and their instances, and also logging information.

The HARNESS API sits slightly above the pure IaaS layer, and is focused on allocation, linking resources and supporting the application manager. Unlike established cloud standards, such as OCCI and CIMI, which come with built-in models of different resource "types", such as VMs, networks, and storage devices, the HARNESS API considers all abstract resources to be of the same (abstract) type.

This allows for a more flexible model that can accommodate a wider array of cloud-enabled devices, as well as supporting cross-cutting services such as pricing and monitoring, which do not have to conform to multiple resource models. Once resources have been allocated in HARNESS, the deployment phase allows each provisioned resource to be handled using specific APIs, tools and services to make full use of their capabilities.

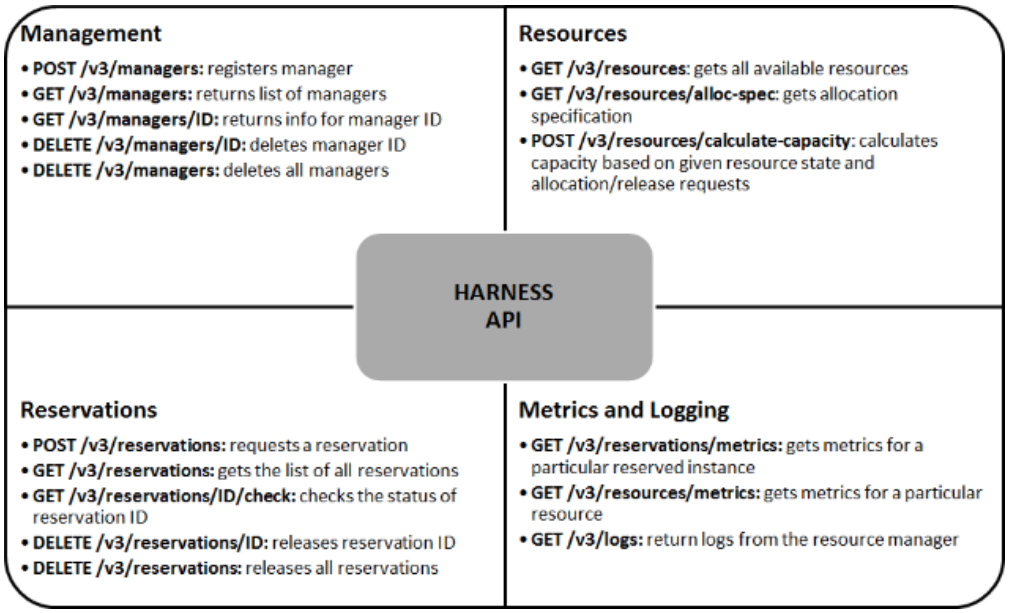

Figure 4. *The HARNESS API.*

Application Management

The HARNESS cloud computing platform can automate the choice of resources to run an application: Cloud tenants indicate requirements for job completion time and/or financial cost. The platform then decides which set of resources best satisfy these objectives and constraints.

To execute an application in HARNESS, a cloud user provides:

- An application manifest describing the structure of the application and the types of resources it can potentially execute on.
- A service-level objective (SLO) defining non-functional requirements for this execution, such as the maximum execution latency or the maximum monetary cost.

The HARNESS platform then deploys applications over well-chosen sets of resources such that the manifest and the SLO are respected. An important part of this process is developing performance models that guide this selection. In HARNESS, we developed several techniques to reduce the profiling effort of generic applications, including taking into account monitoring information to generate high-quality performance models at a fraction of time, as well as extrapolating production-size inputs using reduced-size datasets.

Figure 5. *The HARNESS platform frontend. Once the application has been submitted, the HARNESS platform can profile the application with different resource configurations, resulting in different performance and cost options.*

Cross-Resource Scheduling

The HARNESS Cross-Resource Scheduler (CRS) manages all the infrastructure resources in a data centre. Its novel features include: managing commodity and specialised resources as first-class entities and supporting placement constraints between resources.

The CRS implements the Infrastructure-as-a-Service (IaaS) interface of the HARNESS cloud, providing similar functionality as established IaaS systems such as OpenStack, CloudStack and OpenNebula. While these systems are very mature, HARNESS is unique in the way it handles heterogeneous cloud resources.

Current IaaS cloud systems consider computer nodes as the only first-class type of resources. As such, these systems can only offer VMs as cloud resources. Other resources, such as hardware accelerators, storage and network, are offered either as services or as attributes of provisioned VMs. For example, in Amazon EC2, it is not possible to reserve a virtualised GPGPU as a cloud resource. Instead, a GPGPU is only provided as an attribute of certain types of VM instances, similarly to memory or cores.

However, this model does not work for many classes of heterogeneous cloud resources. For example, a Maxeler MPC-X hosts a number of Dataflow Engines, and thus has a different characterisation than a compute node; since it is shared, it cannot be characterised as an attribute of a particular VM. Hence, the HARNESS CRS treats different types of resources, such as VMs, heterogeneous storage, GPGPUs, FPGAs and network devices, as **first-class resources**. They can be requested independently from each other.

Another limitation of existing cloud platforms is the limited support they offer to control the placement of resources assigned to an application. For example, an application may need to impose latency/bandwidth constraints between multiple resources.

In HARNESS, an application may request one virtual machine and one FPGA located close to each other. It uses the network proximity maps to decide which set of physical resources can accommodate each request. Once this selection has been made, it delegates the actual provisioning of resources to corresponding Infrastructure Resource Managers (IRM), which are respectively in charge of managing specific types of heterogeneous resources, including VMs, GPGPUs, FPGAs and storage and network devices.

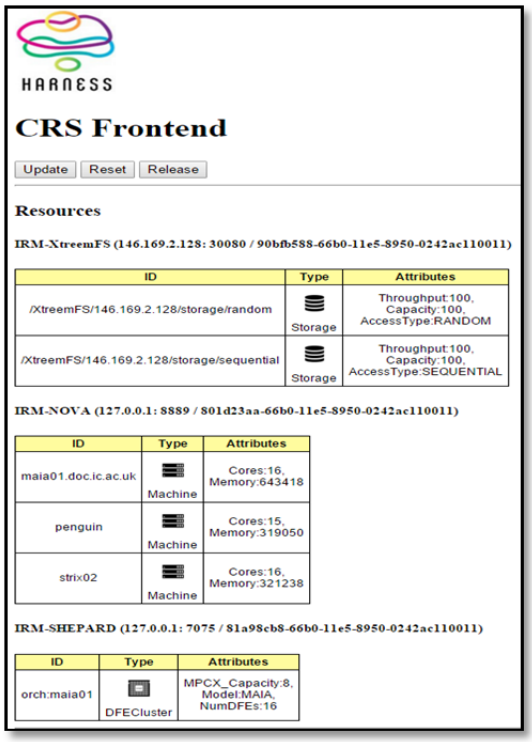

Figure 6. *Cross-Resource Scheduler (CRS) frontend. This figure shows the capacity of available resources (computer nodes, storage devices and the DFE cluster).*

Cross-Compute Optimisation

The compute requirements of applications have increased due to demands from high performance computing, big data and real-time analytics. How can developers build applications efficiently so that they can leverage heterogeneous compute technologies, while allowing the cloud infrastructure to manage workloads effectively?

To address this problem, we have developed a programming and runtime framework called **SHEPARD**. SHEPARD typically manages critical workload tasks, submitted by users, and at run time determines which provisioned device it should execute on depending on available implementations, the estimated task performance and the current level of workload already present on each device. SHEPARD supports two key mechanisms: managed tasks and run-time allocation.

Managed tasks are special functions invoked in the application source code, allowing the SHEPARD run-time manager to decide on task placement. The use of the managed-task mechanism allows the complete removal of device-specific code from cloud-based applications. Managed tasks also remove the need to create workload management strategies from within applications.

This is particularly necessary as individual applications will not have knowledge of workloads from other applications and, therefore, any static allocation made by an individual application will be sub-optimal in the presence of external workloads.

Since device-specific implementations must be provided at some point, the SHEPARD framework supports an implementation library. This library stores device-specific implementations of tasks as well as cost models that describe their performance on the devices they support. With knowledge of this repository, when applications wish to execute tasks at run time, the SHEPARD framework can allow applications to dynamically load and execute task-specific implementations. The use of managed tasks therefore allows the separation of device-specific code and high-level application logic. This allows two types of developers to be identified:

- The *Application Developer* is concerned with the application functionality.
- The *Expert Developer* is responsible for optimised and device-specific codes that can be accessed and loaded at run time by SHEPARD.

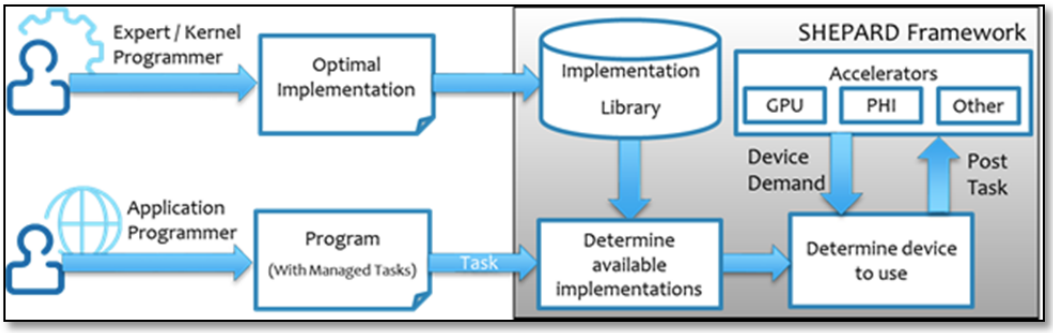

Figure 7. *The SHEPARD workflow.*

The **Run-Time Allocation** process load balances critical tasks submitted by applications across all available devices. This avoids any single device becoming a system bottleneck while other devices may idle, which is a likely scenario when static task allocation is used. By dynamically executing tasks at run time, application developers can be assured that applications will take advantage of the best devices available, and cloud providers can be assured that their hardware will be continually utilised.

When a task needs to be executed, the SHEPARD framework performs a number of steps to determine when and where it should execute. The basic overview of this process is as follows:

- Determine, from the SHEPARD repository, which devices have implementations that support the required task.
- From the task implementations that exist, cost models are used to estimate the running time of the task on each supported device.
- The length of all tasks already queued to each device is used to estimate how long any new task must wait before it can begin executing.
- Using both the expected wait time and task execution time for each device, the device that yields the lowest turnaround time is chosen.

By allowing task implementations to be loaded and task allocations to be made at run time, applications can dynamically adjust their behaviour to fit the resources available. This means that in a cloud as flexible as HARNESS, applications can now match this flexibility to take advantage of all resources that are available.

SHEPARD has been integrated into an in-memory database (IMDB) system to manage and allocate tasks to accelerators, speeding up certain analytic queries submitted by database users. Using SHEPARD, a limited set of compute processing resources is shared among up to 100 users, all wishing to execute accelerated implementations. Results show that by using SHEPARD to manage accelerated tasks, average query time for all users can be reduced by 2-13 times. For the database performance itself, benchmark results clearly show that using SHEPARD to offload additional query workload to accelerators can reduce CPU demand on the host system significantly and improve general database performance.

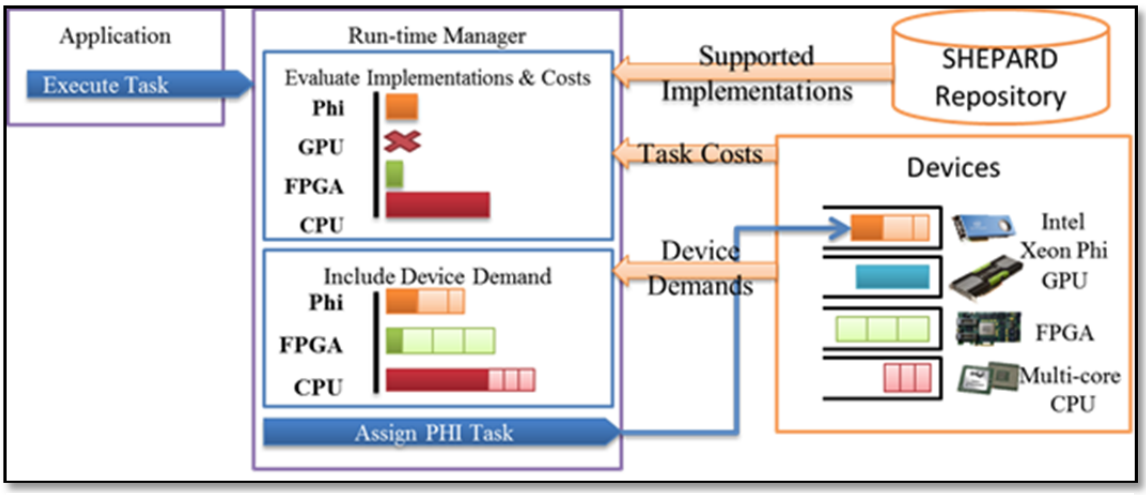

Figure 8. *The SHEPARD framework.*

Dataflow Engine Virtualisation

Field-programmable gate arrays (FPGAs) can offer invaluable computational performance for many compute-intensive algorithms. In HARNESS, we have developed a novel technique for virtualising FPGAs for cloud computing platforms called **DFE groups**, which aggregates the computational power of multiple physical FPGAs, while supporting elasticity for multi-tenancy environments.

FPGAs have become increasingly popular within the high-performance computing community for their excellent computation density, performance/price and performance/energy ratios. For instance, FPGAs are 40 times faster than CPUs at processing some of Microsoft Bing's algorithms. FPGAs are commonly used in domains a diverse as financial market data processing, signal processing, and DNA sequence alignment.

There are several challenges in turning complex FPGA devices into easy-to-use virtual cloud resources. One particular issue is the lack of satisfactory techniques for virtualising FPGAs. Current solutions are based either on statically partitioning the gate array between multiple applications (i.e., sharing the FPGA in space), or on naive context switching (i.e., sharing the FPGA in time). Both techniques exhibit significant problems: sharing in space implies place and route constraints, as well as a smaller number of digital gates and reduced I/O bandwidth, thereby negatively impacting performance. Conversely, naive time sharing incurs prohibitive context-switching costs, as reconfiguring an FPGA from one circuit design to another may take on the order of a couple of seconds.

We developed a new virtualisation technique called **DFE groups** targeting Dataflow Engines. A DFE group is composed of one or more physical FPGAs that are configured with the exact same circuit design. By load-balancing incoming execution requests between its members, a DFE group can be considered by its clients as a *virtual* FPGA that aggregates the computational capacity of multiple physical FPGAs. DFE groups are elastic, as one can easily add or remove physical devices to/from a group.

A DFE group may be, for example, provisioned for cloud applications with computational needs that exceed the capacity of a single FPGA. DFE groups may also be shared among multiple tenants who wish to use a domain-specific library implementing, for instance, data analysis (with functions such as regression, correlation and clustering), and multimedia (with functions such as video encoding and fast Fourier transform).

Finally, multiple DFE groups may also compete for the use of a limited set of physical devices. In this case, we designed an *auto-scaling* algorithm that dynamically assigns physical DFEs to DFE groups. This algorithm maximises utilisation (which improves the cloud provider's revenues), while reducing individual task execution times in most cases.

The figure below shows the web interface of two Reverse-Time Migration (RTM) applications deployed on the HARNESS platform. RTM represents a class of computationally intensive HPC applications used to process large amounts of data. Some of the most computationally intensive geoscience algorithms involve simulating wave propagation through the earth. The objective is typically to create an image of the subsurface from acoustic measurements performed at the surface.

To create this image, a low-frequency acoustic source is activated and the reflected sound waves are recorded, typically by tens of thousands of receivers. We term this process a *shot*, and repeat it many thousands of times while the source and/or receivers are moved to illuminate different areas of the subsurface. The resulting dataset is dozens or hundreds of terabytes in size. The problem of transforming this dataset into an image is computationally intensive and can be approached using a variety of techniques.

In this example, we consider deploying two 10-shot RTM jobs. On the left side, we deployed RTM with one DFE (specified in the application manifest), while for the second job we request a DFE group backed by four physical DFEs. In this case, the second job runs faster, since shots can be processed in parallel and we have four independent workers.

While in this example each job runs with dedicated DFEs, the DFE group mechanism allows both jobs to share the same pool of physical DFEs. In this case, physical DFEs would migrate from one group to another according to workload.

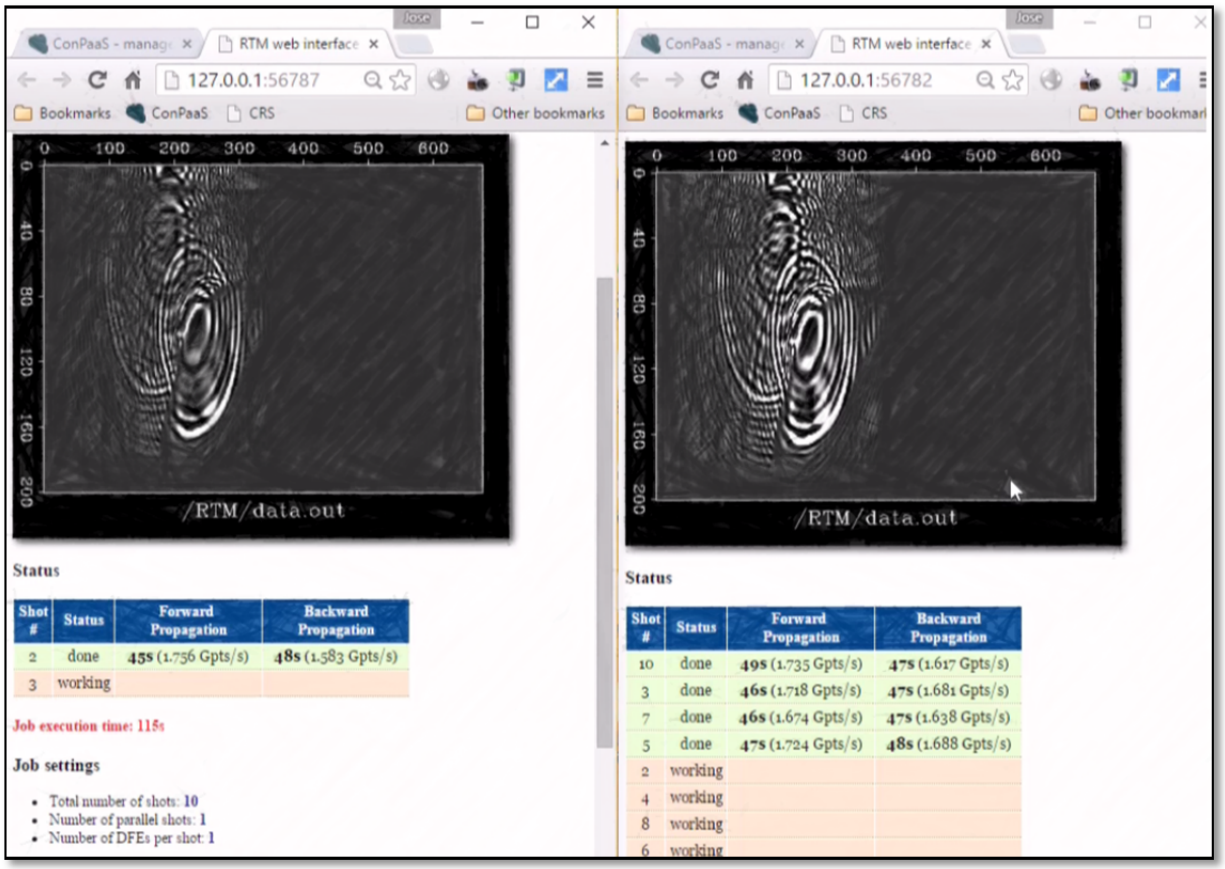

Figure 9. *Running two RTM jobs on the HARNESS cloud platform with different DFE resources.*

Network Management

What types of network resources should a heterogeneous cloud platform expose to its tenants to satisfy non-functional requirements, such as performance and security?

With HARNESS, cloud tenants can request custom network topologies to meet their application requirements. For instance, in addition to reserving VMs with a specific number of cores and amount of memory, tenants can further request that these VMs be interconnected with a specific transmission rate and propagation delay.

We consider three types of network resources:

- **General network functions** are application-agnostic functions implemented by the infrastructure and not by tenants. For instance, the L2 network, L3 router, a load-balancer and access control.
- **Network links** allow cloud tenants to request network performance: e.g., provide VM-to-VM latency below one msec, or VM-to-VM capacity above one Gbps.
- **Application-specific network functions** allow cloud tenants to provide their own network functions. For example, a tenant may define its own access-control function, or a new in-network aggregation function to reduce network congestion.

A novel aspect of the HARNESS platform is its support for network proximity guarantees, something currently not provided by cloud environments such as Amazon EC2 and OpenStack. In order to support network placement constraints in HARNESS, we have designed and implemented the Network Infrastructure Resource Manager (IRM-NET), which communicates with the Cross-Resource Scheduler (CRS) to observe, handle, schedule and/or reserve any network-related resources or constraints:

- **Bandwidth** is exposed to the CRS as a resource between any two heterogeneous resources, e.g., "Between Rs and Rt reserve 100 Mbit/sec."
- **End-to-end latency** is exposed to the CRS as a property that can be used to express constraints between any two heterogeneous resources. For example: "Between Rs and Rt the maximum end-to-end latency should be five msec."

We conducted a case study where we fully deployed HARNESS in Grid'5000 on compute nodes spanning the French cities of Rennes and Nantes. We demonstrated how cloud tenants can gradually eliminate suboptimal resource placements by allocating bandwidth and specifying latency constraints, resulting in improved performance for a Hadoop MapReduce job.

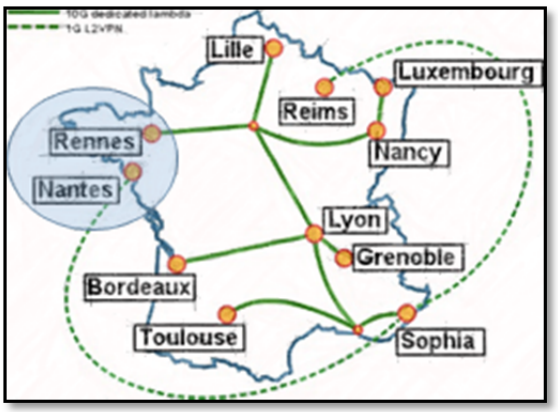

Figure 10. *HARNESS deployment in Grid'5000.*

Storage Management

Most cloud providers support storage services with guaranteed capacity and random access for random patterns. In HARNESS, we support a more general storage reservation system where tenants can request a volume optimised for random or streaming access, as well as performance and capacity.

Storage devices come with a fixed set of performance properties. For example, a hard disk with 4TB of capacity can perform about 120 random access operations per second (IOPS). To optimise utilisation of that device, applications must reserve 33GB (4TB/120) of storage capacity for each reserved IOP. Any deviations will either waste capacity or IOPS. For streaming access, a hard disk can easily achieve 150 MB/s for a wide range of reservation sizes. SSDs on the other hand are much more flexible. These storage devices can easily mix random and streaming access on the same device. Managing heterogeneous storage devices to meet performance and capacity requirements is therefore not trivial, especially when considering multi-tenancy environments.

In HARNESS, we extended **XtreemFS**, an existing cloud file system, with mechanisms for virtualising and reserving storage resources. More specifically, we developed support for performance requests for storage devices and performance isolation for multi-tenancy scenarios. For instance, when users want to reserve storage, they have to specify the capacity and the expected performance. For performance, they can specify the throughput and the access pattern of their application (e.g., streaming or random access). XtreemFS will subsequently reserve the necessary resources from a pool of different devices (e.g., solid state disks, rotating disks, or large RAID systems).

We also developed a monitoring service. It analyses all accesses to a volume and provides feedback to the user. The monitoring service can tell the user, for instance, that at most 800 GB was used and, except for a few spikes, the used bandwidth was below 50 MB/s. For the next run, the user or the application manager can request a smaller, slower, and cheaper volume.

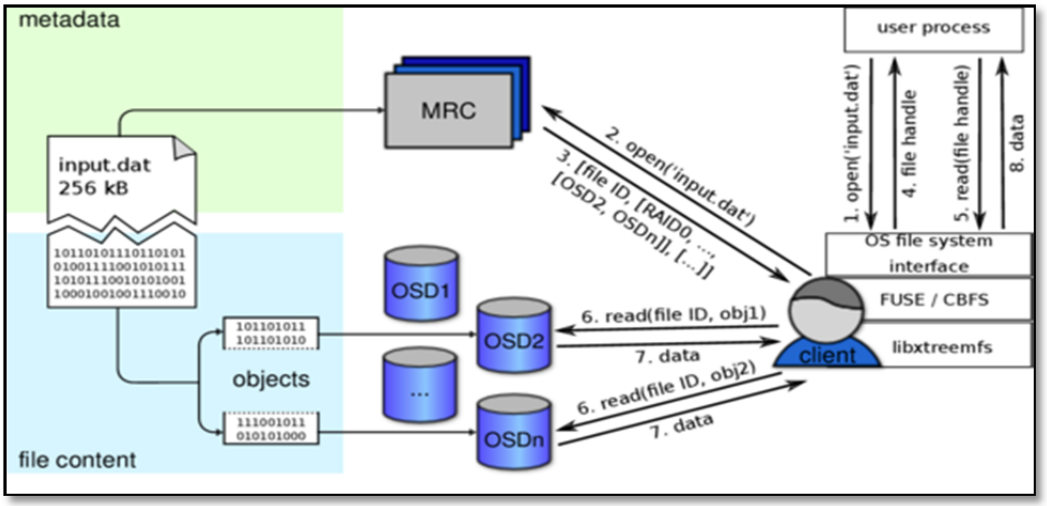

Figure 11. *The XtreemFS architecture.*

Summary

The stringent application requirements for many important business and scientific domains are at odds with the commodity approach followed by cloud providers to build data-centres. Many specialised resources such as GPGPUs, FPGAs, programmable network routers and solid-state disks, if leveraged correctly, can enhance data centres performance to meet these requirements. The key undertaking is not just to make them available to users, but to bring these specialised resources in-line with commodity-based cloud computing methodologies.

The **HARNESS Cloud Computing Platform** ensures this by virtualising specialised and commodity resources so that they can be managed and accessed at the platform level, thus providing flexibility to the platform as to which, when, and how many resources are used.

A **key design feature** of the platform is the **abstract and hierarchical resource management architecture**, where multiple run-time HARNESS resource managers can be combined hierarchically, enabling resource specific management at a local level with more agnostic management of high-level resources for a global platform management. All resource managers utilise the same **RESTful HARNESS API**, which is able to manage lower level resource managers, actual physical resources, resource reservation and monitoring information.

This enables the **key undertaking** of **virtualising resources**, making such resources available for the platform while also ensuring that the platform is **resilient to new forms of heterogeneity** so that new types of resources can be incorporated without having to redesign the entire system.

For **cloud users,** the platform can **automate** the **resource allocation** decision based on the user requirements such as job execution time and/or financial cost deciding which set of resources best satisfy these objectives and constraints.

Allocation of jobs within the HARNESS platform is handled by the **Cross-Resource Scheduler**, which is responsible for managing all resources within the data-centre. Analogous to IaaS systems such as OpenStack, the Cross-Resource Scheduler is unique in that it **handles all resources as first class**, a feature enabled by the hierarchical and agnostic resource management.

For managing compute resources in a heterogeneous environment where CPUs, GPGPUs and FPGAs are available, the HARNESS platform has a **cross-compute management framework** called **SHEPARD** that dynamically determines which provisioned devices should it execute on depending on available implementations, the estimated task performance and the current level of workload already present on each device.

In addition, the platform has for the first time added key tenets of cloud computing, in particular, **elasticity and multi-tenancy**, to **specialised compute technologies**. Specifically, for a DFE, a novel method for virtualising FPGAs has been implemented, aggregating computational power while also supporting elasticity for multi-tenancy environments.

The **cross-compute framework**, along with **virtualisation** and **multi-tenancy** enable the **effective and efficient use of heterogeneous compute resources.**

The ability to provision **custom network topologies** for particular classes of applications, by specifying network transmission rates and propagation delays between compute and storage resources, goes beyond the state of the art in today's data centre services such as EC2 and OpenStack. The **Network Infrastructure Resource Manager**, developed within HARNESS, operates in tandem with the **Cloud Resource Scheduler** to observe, handle, schedule and/or reserve network related resources or constraints such as available bandwidth or end-to-end latency.

Finally, the **HARNESS storage solution, XtreemFS,** with its inherent mechanisms for virtualisation and reservation, supports performance requests for storage devices and performance isolation for multi-tenancy scenarios, offering a more general storage reservation system where tenants can request a volume optimised for random or streaming access, as well as performance and capacity.

Learn more about HARNESS...

Technologies

Description of HARNESS technologies
http://www.harness-project.eu/?page_id=391

Architecture
http://www.harness-project.eu/?page_id=790

Publications
http://www.harness-project.eu/?page_id=21

Software

HARNESS modules and deployment
http://www.harness-project.eu/?page_id=721

ConPaaS
http://www.conpaas.eu/ConPaaS

XtreemFS
http://www.xtreemfs.org

Consortium

The consortium that developed the HARNESS platform consists of world-class, highly experienced academic and industrial research groups, and dynamic, fast-growing, and financially successful enterprises, including three universities (Imperial College London, EPFL, and University of Rennes 1), one research institute (ZIB), one large enterprise (SAP AG), and one small enterprise (Maxeler Technologies).

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