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

Validation Plan
D2.3

Due date: 30 September 2013
Submission date: 30 October 2013

Start date of project: 1 October 2012

Document type: Deliverable
Activity: RTD
Work package: WP2

Editor: Oliver Pell (MAX)

Contributing partners: IMP, MAX, SAP

Reviewers: Gabriel Figueiredo, Guillaume Pierre

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	2013/05/01	Peter Sanders	MAX	Outline
0.2	2013/07/29	Alexandros Koliousis, Eoghan O'Neill	IMP, SAP	Initial content for AdPredictor and Delta Merge use cases
0.3	2013/08/16	Oliver Pell	MAX	Revised outline
0.4	2013/08/21	Oliver Pell	MAX	Initial content for RTM use case and introduction
0.5	2013/08/23	John McGlone	SAP	Redrafted Delta Merge section
0.6	2013/08/27	Alexandros Koliousis	IMP	Redrafted AdPredictor section
0.7	2013/08/29	Oliver Pell	MAX	Global drafting, conclusions and executive summary
0.8	2013/09/02	Oliver Pell	MAX	Adding general requirements, general drafting changes
0.9	2013/09/07	Various	ALL	Miscellaneous minor redrafting and corrections
1.0	2013/09/28	Alexander Wolf	IMP	Final review and edits by Coordinator

Tasks related to this deliverable:

Task No.	Task description	Partners involved°
T2.3	Plan validation activity	MAX*, All

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

Executive Summary

This deliverable describes a plan for the implementation and evaluation of the *Hardware- and Network-Enhanced Software Systems for Cloud Computing* (HARNESS) validation use cases. The HARNESS project will validate the developed platform using three use case applications. The project is developing the platform and validation applications themselves in parallel. By M12, initial standalone versions of the applications were made available to the consortium.

There are three main milestones for validation at which we plan to demonstrate increasing sets of capability: M21, M30 and M36.

By **M21**, separate instances of the platform for each application will be able to demonstrate the basic dispatch of tasks to different types of compute resources. By **M30**, a single integrated platform will support running all three use case applications sequentially, making algorithm and resource-allocation choices based on system and application characteristics. Heterogeneous storage and communication technologies will also be exploited. By **M36**, the platform will be able to demonstrate effective multi-tenancy of the use case applications, and optimisation, not just within an application, but across applications.

The impact of the HARNESS project will be established according to specified evaluation criteria and compared to designated baseline implementations. The key criteria are performance for an application (both absolute and per unit of space or power) and resource utilisation of the cloud resources. The definition of performance itself is application dependent and is specified separately for each application. Resource utilisation of the cloud is an important global metric, since it shows how the HARNESS platform can perform global optimisation to maximise the return on capital investment for cloud providers.

Contents

Acronyms

ConPaaS *Contrail platform-as-a-service*. 20

CPU *central processing unit*. 6, 8, 9, 12, 14, 18–20

DFE *dataflow engine*. 8, 9, 12, 18, 20

DoW *description of work*. 3

FPGA *field-programmable gate array*. 14

GPGPU *general-purpose graphics processing unit*. 12, 14

HARNESS *Hardware- and Network-Enhanced Software Systems for Cloud Computing*. i, 1, 3–9, 11, 13–15, 17–21

HPC *high-performance computing*. 7

MAX *Maxeler Techologies*. 20

NaaS *network-as-a-service*. 20

OLAP *on-line analytics processing*. 11

OLTP *on-line transaction processing*. 11

PCIe *peripheral component interconnect express*. 12

ROC *receiver operating characteristic*. 19

RTM *reverse time migration*. 7–9, 21

SLA *service-level agreement*. 6

1 Introduction

This report describes the validation plan for the *Hardware- and Network-Enhanced Software Systems for Cloud Computing* (HARNESS) use case applications and identifies the criteria that will be used to evaluate the results.

Deliverable D2.1 [1] identified general requirements for the HARNESS platform in the four main areas of technology: platform, computation, communication and storage. Deliverable D2.2 [2] then identified specific requirements for the three industrial use case applications.

For the most part we plan to validate the HARNESS platform through the use case applications. Where some general requirements will not be sufficiently tested through purely looking at use case applications, we have identified additional steps in the validation plan to validate these.

The report is structured as follows. Section 2 describes the overall approach to the validation use cases, general purpose evaluation criteria and overall plan for the three validation use case milestones. Sections 3, 4 and 5 identify, for each application, the application use cases, application-specific evaluation mechanisms and plans for each milestone. In each section, the requirements that will be validated at each milestone are identified. Section 6 concludes this report.

2 Overall Validation Plan

2.1 Approach

The HARNESS project will validate the developed platform using three use case applications. The purpose of this validation is not to show the correct functioning of every part of the platform but rather to demonstrate the value delivered by the HARNESS approach in these three domains.

We target three milestones as identified in the *description of work* (DoW), corresponding to WP7 project deliverables:

1. M21: Initial platform and validation use cases (Deliverable D7.2.1)
2. M30: Updated platform and validation use cases (Deliverable D7.2.2)
3. M36: Final platform and validation use cases (Deliverable D7.2.3)

The project is developing the validation cases and the platform itself in parallel. Thus, at each milestone, we plan to demonstrate certain functionality in the validation applications interacting with certain functionality in the platform. This report identifies the key target functionality for validation at each milestone.

As well as qualitative evaluation, each validation use case will be assessed quantitatively against numerical evaluation criteria. Many of these criteria are common to all the use cases (e.g., performance metrics). However, they may be measured differently for each application. Other application-specific quantitative measures may be defined where the HARNESS platform is expected to have an impact.

For each use case application, we also define a baseline for comparison against some different HARNESS resource configurations that should be assessed.

2.2 Requirements

The validation plan described in this deliverable focuses primarily on validation of the overall HARNESS approach through the three validation use case applications. These applications have particular requirements that have been identified in Deliverable D2.2 [2].

General requirements have also been identified for the HARNESS platform in Deliverable D2.1 [1]. Many of these are closely related to application requirements, as shown in Table 2.1[1].

That the HARNESS platform supports most general requirements can therefore be observed through the use case applications. For general requirements that are not as closely related to the application use cases, validation will occur at a system level or with a standalone test, as detailed in Section 2.4.

[1] Related requirements are identified here from a validation plan perspective and, thus, are considered "related" in a somewhat different sense to the requirement dependencies detailed in Deliverable D2.2 [2].

	General requirement	Related requirements
R1	Provide an execution platform for using heterogeneous resources	R27, R30, R35
R2	Provide support for flexible applications	R26, R30, R34
R3	Provide an application description language	
R4	Provide an SLA description language	R25
R5	Implement application performance model generation	
R6	Implement resource co-allocation scheduler (per application)	R32, R35
R7	Implement resource co-allocation scheduler (multi-application)	
R8	Provide design patterns for flexible applications	
R9	Develop a heterogeneous computational resource discovery protocol	R29
R10	Develop a compiler infrastructure to derive multi-target design variants	R30, R34
R11	Provide characterisation of heterogeneous applications	R32
R12	Enable shared use of heterogeneous computational resources	R32, R33, R35
R13	Provide monitoring data to support decision making at the cloud platform layer	
R14	Optimise the use of resources according to optimisation goals within constraints provided by the cloud platform layer	R25, R32, R33
R15	Enable programming of network processing units	R36
R16	Implement verification of properties of network processing units	
R17	Provide a description of network resources	R29
R18	Manage the allocation of network resources	R36
R19	Provide reporting of network-resource usage	
R20	Implement resource reservations that are abstract from physical devices	R28, R37
R21	Implement time-dependent anticipated usage pattern	
R22	Implement virtual storage classes	R28
R23	Develop performance models	R37
R24	Provide run-time summary of available [storage] resources to the platform	

Table 2.1: General HARNESS requirements and their relationship to industrial application requirements.

2.3 Overall Evaluation Criteria

In this section we identify key quantitative metrics common to all three validation use cases.

2.3.1 Performance

Performance is a key benefit that should be managed by the HARNESS platform. Each application will define one or more performance measures that will be used to compare between its baseline running under different HARNESS configurations.

2.3.2 Performance per unit power

When HARNESS enables applications to effectively exploit resource heterogeneity, this should deliver improvements in energy use, since resources most optimal to a particular problem will be utilised. We will thus measure the impact on power consumption and performance/power for each application. Power will be measured in Watts for the resources executing an application, with each resource's power scaled by the proportion of that resource that is being utilised by that application.

2.3.3 Performance per unit space

Heterogeneous resources will have different density characteristics than racks of homogeneous servers. Computational resources have different sizes per unit of performance, while heterogeneous communication and storage resources will also consume different amounts of space. Space will be measured in rack units, with each resource's space requirement scaled by the proportion of the resource that is being utilised by that application.

For simplicity, space measurements will only include computation, communication and storage resources that are the focus of HARNESS. Space for other rack infrastructure, including power distribution and management, will be ignored.

2.3.4 Resource utilisation

The HARNESS platform should enable cloud providers to maximise the utilisation of the data centre. In particular, since applications will be able to exploit multiple resource types to execute particular tasks, it will be possible to utilise sub-optimal resources if the marginal cost of doing so is less than the benefit; thus overall utilisation should be improved.

Utilisation will be measured separately for each type of resource, averaged over the run of the application.

2.3.5 Cost

The HARNESS platform should enable cost to be minimised subject to other constraints, or a particular cost point to be targeted. There is no single meaningful definition of cost, and it is not within the scope of the project to study the economics of cloud providers specifically, so for HARNESS we will focus on developing several different cost models, and then evaluate the execution of the demonstrator applications against those models.

One possible cost model would be to consider power as the only cost. Another would be to consider factors for both operating expenditures (power, space, maintenance) and capital expenditures (equipment purchase).

Regardless of the cost model adopted, the impact of the HARNESS platform on this cost metric can be evaluated, compared to naive execution of the demonstrator applications on *central processing unit* (CPU)-only platforms.

2.4 Plan for Each Milestone

M21

At M21, each of the validation applications will run separately from each other, interacting with components of the HARNESS platform. Performance will be measured, and space, power and utilisation will be estimated based on reasonable coarse-grained assumptions (for example, that a dual socket CPU node consumes a certain amount of power).

Applications will be described using the application description language and *service-level agreement* (SLA) language (validating **R3** and **R4**). Basic verification of system function will be in place, validating **R16**. The use case applications will be used to extract common design patterns, validating **R8**.

M30

At M30, a single instance of the platform will be able to run all three validation applications sequentially, on an otherwise idle system. Performance will be measured, with space, power and utilisation based on improved estimates (for example, power consumption for a single node running an application will be measured and then scaled up to the test system).

While applications run separately, resource allocation decisions will be influenced by the general environment, for example system priorities given time of day, validating **R21**.

M36

At M36, the validation platform will be able to run all validation applications simultaneously, allowing interactions between the resource optimisation for each application to be explored and analysed. Space for the actual test system will be recorded and power consumption will be measured for the test system while running.

The multi-tenancy system will allow **R7** to be validated. The platform will make global decisions based on monitoring data, validating **R13**, **R19** and **R24**. Performance models for each application will be evaluated, validating **R5**.

3 HPC: Reverse Time Migration

Reverse time migration (RTM) is a *high-performance computing* (HPC) application drawn from the domain of geoscience where it provides commercially important information to the oil industry.

RTM creates an image of the subsurface of the earth from acoustic measurements recorded at the surface. It does this by modelling the propagation of seismic waves twice, one from the perspective of a seismic source and one from the perspective of the receivers. The pressure fields are cross-correlated at each point in space and time to create an image.

In the HARNESS project, two different RTM algorithms will be explored, providing the opportunity to study the trade-off between computation and storage requirements. More details about the RTM application can be found in Chapter 2 of Deliverable D2.2 [2].

3.1 Requirements

For convenience we list below the application requirements for RTM. A full description and discussion of these requirements can be found in Section 2.6 of Deliverable D2.2 [2].

Requirements for RTM	
R25	The HARNESS platform shall allow the expression of target latency or target cost
R26	The HARNESS platform shall support multiple algorithms
R27	The HARNESS platform shall exploit heterogeneous compute resources
R28	The HARNESS platform shall support fast linear read/write access to temporary data storage with understanding of duty cycle

3.2 Application Use Cases

3.2.1 Workloads

We will test RTM on two different size problems, representing small- and medium-size commercial jobs. Large jobs will have the same basic properties as the medium-size job we describe here, except that it demands larger amounts of memory and compute resources.

Data set 1

Data set 1 is a small processing job with a spatial domain of 400^3 grid points. Shot duration is 8 seconds, modelled as 4000 timesteps with a 2ms timestep.

Data set 2

Data set 2 is a larger job with a spatial domain of 800^3 grid points. Shot duration is 8 seconds, modelled as 8000 timesteps with a 1ms timestep.

Data set 2 involves approximately 8× the computation as data set 1 per shot.

3.2.2 Resources

To assess the efficient exploitation of heterogeneous resources, the RTM case study will be evaluated in two resource environments:

- *compute rich*, with a large number of *dataflow engine*s (DFEs) or CPUs, and relatively low-performance storage and
- *storage rich*, with a more modest amount of compute resources and high-performance storage.

3.3 Evaluation Mechanism

3.3.1 Baseline for comparison

The baseline for comparison will be RTM run using a stored-snapshot algorithm on a CPU-only system.

3.3.2 Performance measures

Throughput

RTM is a throughput-oriented application. The key metric is shots imaged per unit of time (e.g., per hour).

Since data set 2 will naturally exhibit much lower throughput than data set 1, due to the higher computational cost, we can also assess normalised throughput, where the shot run time is normalised by the theoretical compute cost for that shot (i.e., 1.0 for data set 1 and 8.0 for data set 2).

Shot latency

Since geoscientists will typically perform quality control checks as an RTM job is running, the latency per shot is also significant, particularly for the first few shots of the overall RTM job. This will be measured as start-to-end time for a single shot image to be produced.

3.3.3 Other evaluation criteria

We are implementing two RTM algorithms (based on stored snapshots and reverse propagation) with different trade-offs of compute and storage requirements. These algorithms will produce somewhat different numerical results. However, we do not plan to assess the quality of the overall RTM image based on the composition of these results as part of this project; this is a matter of geoscience and out of scope for HARNESS.

3.4 Plan for Each Milestone

The plan for development of the RTM validation case is as follows.

3.4.1 M21

Four standalone implementations of the RTM application have been completed as of M12 and made available to the consortium. These versions provide the two RTM algorithms and, for each algorithm, a CPU-intensive and DFE-intensive implementation.

By M21, the RTM application will be integrated with the HARNESS platform, validating **R27** by showing the selection of different compute resources on a per shot basis. The platform will be able to invoke either RTM algorithm based on user parameters. Shots will run sequentially or with some limited parallelism.

Deliverable D7.2.1 will report the initial RTM implementation details, as well as the impact on performance of utilising different algorithms and computational resources.

3.4.2 M30

By M30, the RTM application will demonstrate the HARNESS platform automatically selecting from between the two available algorithms, as well as the different hardware resources to optimise cost or latency, validating **R25** and **R26**. Multiple shots will be migrated on the demonstration system simultaneously.

Deliverable D7.2.2 will report on the effectiveness of the HARNESS platform's automatic selection of algorithms and allocation of resources.

3.4.3 M36

By M36, the RTM application will make optimised utilisation of storage resources and the HARNESS platform will show that it is able to select storage resources by taking into account write-duty cycles for RTM temporary data, validating **R28**. Validated cost models will be developed.

For the final Deliverable D7.2.3, we will evaluate the different approaches and determine how effectively the HARNESS platform is managing the trade-off between different compute resources and storage capabilities.

4 In-Memory Databases: Delta Merge

The delta merge process is a fundamental component of the SAP HANA in-memory database and is integral in maintaining the read and write performance of the database. SAP HANA supports both transactional, *on-line transaction processing* (OLTP), and analytical, *on-line analytics processing* (OLAP), queries. Due to the nature of how data are stored within HANA, transactional queries are first stored in a delta storage enabling HANA to maintain fast read and write performance across the database. The delta merge process takes the updates within the delta store and merges these to the main store where most of the data are held in a highly compressed fashion. A full description of the intricacies of the delta merge process can be found in Chapter 3 of Deliverable 2.2 [2].

4.1 Requirements

For convenience we list below the application requirements for delta merge. A full description and discussion of the requirements can be found in Section 3.6 of Deliverable D2.2 [2].

Requirements for Delta Merge	
R29	The HARNESS platform shall provide support for specification of particular resources
R30	The HARNESS platform shall support multiple application implementations
R31	The HARNESS platform shall efficiently schedule small compute jobs
R32	The HARNESS platform shall intelligently schedule jobs to resources
R33	The HARNESS platform shall support elastic resource allocation

4.2 Application Use Cases

The delta merge operation executes over an entire column-oriented database table. As such the delta merge operation is phrased as a number of independent tasks per column, with the opportunity to execute each task in parallel.

4.2.1 Workloads

The inputs to a delta merge operation are highly dependent on the HANA system usage and vary between systems. Within any HANA system there will exist a number of tables ranging in size from a very small number of columns to thousands. The column size for each table will also vary greatly with particularly large tables containing columns with the order of millions of entries. The result is a wide range of possible input values to the delta merge operation and an associated range of performance characteristics.

The delta merge is also sensitive to the ratio between main dictionary size and the delta dictionary as well as the percentage of unique values within each dictionary. Placement of individual delta merge tasks per column is therefore sensitive to the input. Very small tasks incur little overhead on the system and

are often better served on the host CPU. Processing large columns on closely coupled accelerators can provide benefits in two ways. Firstly, larger data sets can achieve measurable speed-up on accelerators such as a *general-purpose graphics processing unit* (GPGPU); and secondly, by offloading the workload to an accelerator the demand on the host CPU is reduced, allowing other database operations to be serviced more quickly.

As the merge operation is highly dependent on the dictionary size, column size and data type, we will use data sets providing a variety of different characteristics. For particular tables, we will follow the Pareto principle existent in database data sets where the majority of tables have a small number of entries, while a minority of tables will have a large number of entries.

4.2.2 Resources

The nature of the delta merge operation and its sensitivity to inputs means that the platform on which the operation is performed can have a profound impact on the allocation decisions for each task. Therefore validation of the delta merge use case will require a diverse set of resources and a number of platforms to compare execution.

Typical HANA systems. are deployed on nodes with high-end server CPUs. In this scenario the CPU is responsible for all operations within a HANA system. This means that the delta merge operation can have an impact on other HANA operations, creating a trade-off between the effects of allowing the delta dictionary to grow and performing a delta merge. In this scenario it may be advisable to limit the number of CPU threads that the delta merge operation uses to temper the impact on the HANA system as a whole.

Systems with tightly coupled accelerators., such as GPGPUs over *peripheral component interconnect express* (PCIe), introduce a great deal of flexibility to the execution of delta merge operations. Accelerators can provide benefit by improving the execution time of merge tasks or by simply removing load from the host CPU. However, as the size of a table and the load on a HANA system varies, adequately deciding where to process delta merge tasks remains a challenge. Under scenarios of high load it may be prudent to favour accelerators to free CPU capacity. Alternatively, when the CPU is idle it may be better equipped to perform some delta merge tasks better than the GPGPU, especially for smaller column sizes. There is also the possibility to share tasks between accelerators and host CPUs and indeed handle multiple delta merge operations simultaneously. In these cases, distribution of the tasks is important so that no one resource becomes over utilised which will result in performance degradation of the entire delta merge process.

Systems with remote processing resources., such as DFEs, provide a further alternative to satisfying delta merge tasks. In these cases the overhead of transferring data to and from the external resource will have a major impact on performance. This requires that estimation of overall execution time be sensitive to memory transfer and factor this in as part of the decision making process when allocating tasks to processing resources.

4.3 Evaluation Mechanism

Evaluation of the delta merge operation on the HARNESS platform will be dependent on its ability to improve not only the delta merge operation as a whole, but also its ability to prevent the delta merge from negatively impacting HANA system performance.

4.3.1 Baseline for comparison

The baseline for evaluation of the use case is based on typical resource usage of the delta merge operation in an actual HANA system. In general, the guideline resource allocation for a delta merge operation is two threads within the HANA system. Each thread is responsible for performing a single sequential delta merge task on an individual column. This allocation is static and determined by the system administrator.

4.3.2 Performance measures

Total delta merge execution time

When possible, the HARNESS platform should provide measurable speed-up for delta merge tasks compared to the base line scenario of a typical HANA system.

Average delta merge execution time per column

The HARNESS platform should allocate column delta merge tasks to the processing device that is best able to perform the task. The best able device is that which can process the task and return the result in the lowest amount of time. This will depend on the current load on each device.

Reduction of delta merge impact on HANA system

The HARNESS platform should consider the current workload demand placed on each processing resource when making allocation decisions. As a result, delta merge tasks should aim to make use of under-utilised resources when this can improve performance to avoid placing unnecessary workload upon already busy resources.

4.3.3 Other evaluation criteria

The delta merge process must support *adaptive allocation of tasks*. These can be summarised as:

The ability to allocate tasks based on input Each delta merge task is sensitive to its input, with execution time scaling according to the size of the input. Therefore, placement of a particular task will require the ability to estimate performance for each device based on input.

The ability to use devices that have available implementations The HARNESS project should enable applications to access devices based on available implementations. Should devices exist on a platform that do not have available implementations they should be automatically discounted from all allocation decisions without the need to explicitly exclude them. This means that for the delta merge, where implementations for certain data types may only be available on select devices, the HARNESS platform is able to alter its allocation decision accordingly.

The ability to adapt to resource allocation Since the delta merge may impact the performance within a HANA system its run-time allocation of resources may need to be altered dynamically. The HARNESS platform will provide a budget to a SAP HANA instance and it is vital that the delta merge can adapt to this allocation. Therefore the delta merge operation should be able to scale across available processing resources and avoid over-utilisation of any single resource to help maintain overall system performance.

The ability to schedule tasks without imposing significant overhead on performance The benefit provided by the allocation mechanism should outweigh any execution overhead on the system. In particular, the delta merge can be performed on tables containing a large number of columns which contain only a handful of values. In this case execution should not suffer as a result of the scheduling mechanism.

The ability to handle task parallel load balancing over heterogeneous resources Validation of the HARNESS prototype will focus on the performance of the delta merge operation and the ability of the HARNESS prototype to allocate and balance tasks between available processing resources. The HARNESS prototype should be able to adequately estimate task performance on a per column basis and appropriately place execution on the best available device. This requires the development of performance models for each device implementation. These models can then be evaluated at run-time using live task input parameters to estimate task performance. Secondary to this goal, the HARNESS prototype should be able to load balance tasks between available processing resources to avoid performance degradation due to over utilisation of a single resource. To this end the HARNESS prototype will need to estimate the current workload on a given device and factor this into the decision making process.

4.4 Plan for Each Milestone

This validation case requires implementation of the delta merge operation for a selection of hardware architectures, such as multi-core CPU and GPGPU. With the availability of these implementations profiling can be performed to log execution times which can then be used to derive performance models for each implementation and associated devices. Initial implementation will require exploration executions on each device that will be performed over a representative set of inputs.

The plan for the development of the delta merge validation case is as follows.

4.4.1 M21

CPU implementations for all common data types were completed and made available to the consortium at M12. A GPGPU-based implementation for 32-bit integer is now also available. Investigation and design of other data types, and the use of other possible accelerators for the delta merge algorithmm such as *field-programmable gate array*s (FPGAs), will take place beyond M12. By M21, the HARNESS platform will be able to specify resources for the delta merge application, as an additional requirement of a HANA system, validating **R29**. The delta merge application will be able to run within the HARNESS platform on traditional CPU resources and at least one other type of compute resource.

Deliverable D7.2.1 will report on accelerator-based implementations of delta merge, while also showing the integration of the merge process with the HARNESS platform and how the platform handles the specification of resources for the merge process of a HANA system.

4.4.2 M30

By M30 the integrated delta merge application will provide support for multiple implementations, in the form of differing data types and differing implementations on accelerators thereby validating **R30**.

Deliverable D7.2.2 will report on the multiple implementation support now provided within HARNESS and report on the effectiveness of the run-time selection of particular implementations and resource allocation.

4.4.3 M36

By M36 the HARNESS platform will demonstrate both low level and high level algorithms for efficiently and intelligently scheduling resources for the delta merge process across heterogeneous resources, validating **R31**, **R32** and **R33**. The platform will also support elastic resource allocation and demonstrate this for the delta merge process. Validated cost models will be derived to assist platform decision making. The final deliverable, D7.2.3, will detail and evaluate this work demonstrating the platforms capability to manage heterogeneous resources for an in-memory database.

5 Data-Flow Computing: AdPredictor

AdPredictor represents a class of modern industrial-scale application, commonly known as *recommender systems*, that target either open-source or proprietary on-line services. In general, items are matched with users and, due to today's data deluge, these computations are usually run in large-scale data centres.

AdPredictor is an on-line Bayesian machine learning system that is used by the Microsoft Bing search engine for advertisement ("ad") recommendations. The training phase of the AdPredictor model is typically a multi-round map/reduce job: as the number of rounds increases, the model converges and the accuracy of results increases. The input is a collection of ad impressions, recorded during user search sessions. During the map phase, each task computes partial updates to the model based on a subset of the ad impressions. The results are aggregated during the reduce phase, where the original model is updated. The newly computed model serves as an input to the next map/reduce round, and so on. Chapter 4 of Deliverable D2.2 [2] provides a more detailed description of the application.

5.1 Requirements

For convenience we list below the application requirements for AdPredictor. A full description and discussion of these requirements can be found in Section 4.6 of Deliverable D2.2 [2].

Requirements for AdPredictor	
R34	The HARNESS platform shall support coarse-grained, resource-agnostic, goal-centric specifications of inference tasks
R35	The HARNESS platform shall efficiently schedule inference tasks on heterogeneous processors
R36	The HARNESS platform shall support in-network aggregation
R37	The HARNESS platform shall support multiple data-storage access patterns

5.2 Application Use Cases

5.2.1 Workloads

AdPredictor will be evaluated using two workloads: the 2012 KDD Cup data set[1] and a synthetic data set. The former will enable us to evaluate the application in terms of accuracy, while the latter admits to analysis of scalability.

2012 KDD Cup data set

The 2012 KDD Cup training data set contains 149,639,105 ad impressions. These training instances have been derived from session logs of a proprietary search engine, `soso.com`, appropriately anonymised. A

[1] `http://www.kddcup2012.org/c/kddcup2012-track2`

Attribute name	*Description*
Ad id	The ad displayed during a user search session
Depth	The total number of ads displayed during that session
Position	The order of *ad id* amongst *depth* advertisements
Query id	The query (encoded as a set of tokens) posed by the user to the search engine
Advertiser id	The advertiser
Keyword id	The set of tokens for which the advertiser has bid
URL	The URL of the ad
Title id	The title (a set of tokens) of the ad
Description id	The description (a set of tokens) of the ad
User id	The user id, also corresponding to an *age* and *gender* group
#impressions	The number of times `ad id` was impressed to the user
#clicks	The number of times the user clicked on the ad

Table 5.1: Ad impression attributes from the 2012 KDD Cup data set.

Attributes	#	min	max	μ
Ad titles	3,735,796	1	1,350,400	40.055
Ad descriptions	2,934,101	1	1,350,400	50.999
Purchased keywords	1,188,089	1	2,432,657	125.949
Search queries	2,412,207	1	2,437,411	6.203
Users	22,023,547	1	37,733,352	6.794

Table 5.2: Parsing 149,639,105 lines of training data. String attributes, such as search queries and ad titles, are subsets of 1,070,853 unique tokens.

summary of the attributes recorded for each ad impression is presented in Table 5.1. A further analysis of this data set yielded size and attribute characteristics that are summarised in Table 5.2. In the AdPredictor model, each attribute is modelled as a mixture of Gaussian distributions. For example, the "user" attribute will be presented to our technology as approximately 23 million Gaussian distributions.

Synthetic data

In order to test the storage and communication capabilities of the HARNESS platform, we are going to generate data based on the distribution characteristics of the KDD Cup data set (see Table 5.2). The main difference lies with volume (i.e., the number of impressions used for training).

5.2.2 Resources

The AdPredictor application can benefit from heterogeneity in computation, communication, and storage resources. Thus, we will explore its behaviour under three different system configurations:

- *DFE rich*, with a high ratio of DFEs to CPU cores;

- *storage rich*, with fast access to sizeable disk and memory technologies; and
- *network rich*, with fast access to in-network aggregation switches.

Notice that these three configurations highlight each of the three base technologies explored in the HARNESS project.

5.3 Evaluation Mechanism

5.3.1 Baseline for comparison

The baseline for AdPredictor will be a CPU-only Hadoop MapReduce implementation running over a standard network with homogeneous disk storage.

5.3.2 Performance measures

Number of impressions per second

For the computation-intensive map phase, the performance of the application is determined by its ability to process a large number of ad impressions quickly.

In-network data reduction ratio

For the communication-intensive shuffle phase, the performance of the application is determined by its ability to aggregate intermediate results as they pass through compute-enabled network switches.

I/O operations per second

Intermediate results from map tasks are stored on local disks. Furthermore, the AdPredictor model needs to be written to the distributed file system by reducers, and read from mappers on every iteration.

Job completion time

The use of HARNESS computation, communication, and storage technologies should overall improve the end-to-end completion time of training. Eventually, we will be able to attribute (breakout) this speed-up for each choice of technology.

5.3.3 Other evaluation criteria

Since the results of the AdPredictor application are affected by coarse-grained parallelism, we must also evaluate the accuracy of any execution to ensure fair comparisons. An evaluation metric for accuracy is, for example, the area under a *receiver operating characteristic* (ROC) curve of true positive over false positive predictions. The KDD Cup data set also ships with a testing data set that will help us build our evaluation methodology in this regard.

5.4 Plan for Each Milestone

The plan for development and evaluation of AdPredictor is as follows.

M21

At M12, a CPU-only Hadoop version of AdPredictor was completed and made available to the consortium.

Partner *Maxeler Techologies* (MAX)'s MaxCompiler SLiC interface, enabling use of DFEs for AdPredictor, will be integrated with the MapReduce scheduler by M21. This will allow us to evaluate **R34** and **R35**. Thus, for Deliverable D7.2.1, we plan to report on the impact of heterogeneous compute resources within the HARNESS platform for AdPredictor.

M30

By M30 we plan to integrate AdPredictor with *network-as-a-service* (NaaS) and XtreemFS. This will allow us to evaluate **R36** and **R37**. For Deliverable D7.2.2 we therefore plan to report on the impact of heterogeneous communication and storage technologies on AdPredictor.

M36

By M36 we plan to derive a cost model and a programmatic model to present AdPredictor as a service to cloud tenants. AdPredictor will also be integrated with *Contrail platform-as-a-service* (ConPaaS). For the final Deliverable D7.2.3, we will evaluate our approaches and see which of computation, communication, and storage acceleration provides the best accuracy versus performance versus cost trade-offs.

6 Conclusions

This report has detailed the plan for validation of the HARNESS platform via the three application use cases: RTM, Delta Merge and AdPredictor.

The overall approach is to develop the validation use cases in parallel with the HARNESS platform, validating and expanding set of features of the platform at three milestone points: M21, M30 and M36.

Initial standalone versions of all three validation cases have been made available to the consortium at M12. By M21, separate instances of the platform for each application will be able to demonstrate the basic dispatch of tasks to different types of compute resources.

By M30, a single integrated platform will support running all three validation use case applications sequentially, with the platform demonstrating automatic optimisation at the resource and algorithm level, making algorithm and resource-allocation choices based on system and application characteristics. Heterogeneous storage and networking technologies will also be exploited.

By M36, the platform will be able to demonstrate effective multi-tenancy of the use case applications, and optimisation not just within an application but across applications. Validated cost models will support platform decision making.

Bibliography

[1] FP7 HARNESS Consortium. General requirements. Project Deliverable D2.1, 2013.

[2] FP7 HARNESS Consortium. Industrial requirements. Project Deliverable D2.2, 2013.