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

Characterisation Report
D4.1

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

Start date of project: 1 October 2012

Document type: Deliverable
Activity: RTD
Work package: WP4

Editor: Peter Pietzuch (IMP)

Contributing partners: IMP, EPL

Reviewers: Alexandros Koliousis, Gabriel Figueiredo

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	2013/08/12	Peter Pietzuch	IMP	Outline
0.2	2013/08/12	Peter Pietzuch	IMP	Initial draft of in-network aggregators characterisation
0.3	2013/08/26	Peter Pietzuch	IMP	Work on executive summary
0.4	2013/08/27	Peter Pietzuch	IMP	Additional material on heterogeneous network technologies
0.5	2013/08/27	Katerina Argyraki	EPL	Material on SDN and software packet processing
0.6	2013/08/30	Peter Pietzuch	IMP	Further updates
0.7	2013/10/05	Katerina Argyraki	EPL	Material on SDN and software packet processing
0.8	2013/10/05	Peter Pietzuch	IMP	Improvements to applications benefits discussion
1.0	2013/10/10	Alexander Wolf	IMP	Final review and edits by Coordinator

Tasks related to this deliverable:

Task No.	Task description	Partners involved°
T4.1	Define heterogeneous communication resources model	IMP*, EPL

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

Executive Summary

WP4 focuses on different heterogeneous network technologies. Therefore, an obvious starting point for designing the *Hardware- and Network-Enhanced Software Systems for Cloud Computing* (HARNESS) network layer that will exploit multiple network technologies is to understand their respective strengths and weaknesses. In this deliverable, we conduct an initial study of the opportunities that we foresee that specific heterogeneous network technologies will have for the HARNESS platform.

To explore the entire design space of network heterogeneity at different layers of the network stack, we consider the following three emerging types of network technologies, which are representative of different classes of network devices:

1. *software-defined networking* (SDN) switches,
2. *software packet-processing platforms* and
3. *in-network processors*.

These three types of network devices differ in their performance characteristics, management capabilities, and programming models. For example, SDN switches support programming of their forwarding state only, whereas software packet-processing platforms support programming of their entire packet-processing functionality. Each raises its own opportunities and challenges when applied to the HARNESS demonstrators.

While the characteristics of SDN switches and software packet processing platforms are well understood and have been studied extensively in the literature, application-specific processing using in-network processors is largely unexplored. As a consequence, our characterisation report puts particular emphasis on in-network processors, describing the opportunities and challenges in the context of a restricted case study: in-network data aggregation. We conclude by describing the benefits that each of the three HARNESS demonstrators is likely to gain from the network technologies considered.

Contents

Acronyms

CPU *central processing unit*. 4

HARNESS *Hardware- and Network-Enhanced Software Systems for Cloud Computing*. i, 1–4, 9

OLTP *on-line transaction processing*. 2, 9

PCIe *peripheral component interconnect express*. 9

RTM *reverse time migration*. 2, 9

SDN *software-defined networking*. i, iii, 1, 3, 9

ToR *top-of-rack switch*. 6

1 Introduction

The goal of the *Hardware- and Network-Enhanced Software Systems for Cloud Computing* (HARNESS) project is to improve cloud computing platforms through the use of heterogeneous technologies. For distributed applications in a cloud environment, the network is a crucial resource that often determines the performance of applications.

In the recent past, research in data centre networking has led to a large number of proposals for solving specific problems, ranging from physical-layer technologies all the way to application-specific solutions. For example, *software-defined networking* (SDN) provides logically centralised control for network management functions, which permits data centre operators to manage their network infrastructure more flexibly and efficiently; *software-defined routers* and *middleboxes* support the implementation of novel types of packet processors and the fast deployment of custom routing protocols; and new types of network topologies aim to overcome performance bottlenecks in data centres by exploiting multi-path routing and offering full bisection bandwidth between edge servers.

Such proposals have begun to introduce heterogeneity into the data centre network, which puts HARNESS in a unique position to exploit and enhance these new network technologies for the benefit of cloud applications. As an initial challenge, this report explores the space of heterogeneous network devices and technologies that can add new capabilities to cloud computing environments. We focus on three representative types of devices that are currently gaining traction and are representative of the features of different heterogeneous network elements:

- **SDN switches.** We choose SDN switches for two reasons: first, SDN is the strongest movement in the networking community and, second, despite the large number of related research proposals and products, we believe that the current design of SDN switches leaves significant room for improvement.

- **Software packet-processing platforms.** We choose them for two reasons: first, it is becoming increasingly popular within both the research and industrial communities and, second, existing work on this topic has not resolved the issue of unpredictable network behaviour and performance—one of the biggest practical challenges faced by software routers.

- **In-network processors.** We choose them for two reasons: first, in-network processors are a generalisation of network "middleboxes", which permit arbitrary, application-specific transformation of network flows "in-path". They therefore constitute the next evolutionary step of middleboxes, adding hitherto unprecedented levels of flexibility and programmability to data centre networks. Second, in-network processors are complementary to the other heterogeneous network technologies: they operate on application-level payload streams, in contrast to lower-level packet-based abstractions. We believe that they can offer useful insights into the generalisation of different programmable network technologies for the overall HARNESS platform.

These three type of heterogeneous network technologies also show different levels of maturity: SDN switches have reached the mainstream with a broad range of network equipment manufacturers supporting

the *OpenFlow* standard; research in software packet processors has resulted in a range of mature open-source platforms, including *Click*, *XORP*, and *RouteBricks*, which attracted a plethora of implementations of routing protocols, firewall functionality and network applications on top of them; in-network processing is a less mature technology with only early research proposals available. We therefore focus on the exploration of the characteristics of in-network processors in this report because this constitutes the most significant research contribution.

In summary, our goal is to describe the properties of these three network devices in relation to their use in a cloud computing environment. Based on their properties, we conduct an assessment of the benefits that they can bring in order to achieve the goals of the HARNESS project, i.e. , improve the performance of cloud applications and the management efficiency of cloud environments. We thus discuss their potential for improving the performance of the three HARNESS validation use cases: in-memory database acceleration for *on-line transaction processing* (OLTP), geoscience computing via the *reverse time migration* (RTM) algorithm, and distributed data-flow computation in AdPredictor.

This deliverable is related to one WP4 task:

- Task 4.1 ("Define heterogeneous communication resources model"). There are two general requirements [9] associated with this task: that we can provide a description of network resources (R17), which can be used to inform the decision-making carried out by the HARNESS platform, and (ii) that the resources provided remain verifiable (R16) so that applications can expect correct and predictable behaviour.

2 Technology Characterisation

In this chapter we discuss the three heterogeneous network technologies. In addition to giving a brief introduction to SDN switches and software packet-processing platforms, we focus on the potential of the most promising and disruptive network technology: arbitrary application-specific in-network computation.

2.1 SDN Switches

Overview. An SDN switch [22] differs from a traditional switch in that it does not run any control-plane functions locally. All control-plane functionality is delegated to a centralised controller, which is in charge of populating the forwarding state of all the switches in the network. Where traditional network devices determine the content of their forwarding state by participating in a distributed algorithm, SDN devices are *given* the content of their forwarding state by the centralised controller. This new approach promises to simplify networks by making it easier to deploy sophisticated network policies. The simplification comes from the fact that there exists one centralised entity (the controller) that has a global view of the network and makes decisions about the entire network. In contrast, in a traditional network architecture (with a distributed control plane), to enforce a given policy (e.g., that all traffic from a given tenant should go over low-latency paths), the operator must essentially manipulate distributed control-plane algorithms to obtain the desired outcome; the resulting network behaviour is often unpredictable and hard to reason about.

Requirements. HARNESS, as would any cloud platform, can benefit from SDN switches because of their promised simplicity in deploying network policies. In the HARNESS project we are not exploring new ways of leveraging SDN switches, as it is one of the most well-explored topics in the networking community. However, we do explore new ways of designing and building SDN switches that would improve their performance scalability; this is discussed in Deliverable D4.4.1 [11].

2.2 Software Packet-Processing Platforms

Overview A *software packet-processing platform* [3, 7, 17] differs from a traditional network device in that it performs all packet processing in software running on general-purpose hardware. In contrast, traditional high-end network devices perform most of their packet processing in specialised hardware. The main reason for this latter choice is performance: network devices have traditionally performed a small number of tasks, but they need to perform them at a very high speed, hence a hardware implementation makes sense. However, in recent years, there is increasing demand for new kinds of packet processing, from intrusion detection to transcoding to application acceleration. Implementing packet processing in hardware has made the network inflexible, as deploying new functionality on hardware-based network devices is practically impossible. Therefore, software packet-processing platforms promise to make the

network more flexible, in the sense that adding new functionality to the network becomes equivalent to performing a software update.

Requirements HARNESS, as would any cloud platform, can benefit from software packet-processing platforms because of this flexibility in the evolution of network functionality. In the HARNESS project we are not exploring new ways of leveraging software packet-processing platforms, as there already exists significant work on this topic [15, 16, 19, 25, 26]. However, we do explore new ways of designing and building packet-processing platforms that would improve their reliability; this is discussed in Deliverable D4.4.1 [11].

2.3 In-Network Processors

In cloud data centres, distributed applications, such as batch processing frameworks, distributed data management systems and online services, often incur significant network usage, which can make their performance network bound. Yet today, tenants have little or no control over the network that supports their applications. While tenants can accurately select the number and types of computational and storage resources needed for their applications, they cannot adapt the network. This means that all application-specific processing of network data must occur at the end hosts. Services that could reduce network bottlenecks, such as packet deduplication, in-network data aggregation and multicast, are not supported by cloud providers and instead must be implemented less efficiently by individual applications using overlay networks.

Network bottlenecks in cloud data centres, however, can be mitigated by exposing *in-network processing* functionality to tenants in an efficient manner. Through *in-network processors*, it is possible to allow tenants to control additional computing resources collocated with switches and routers. The availability of multi-core *central processing unit*s (CPUs) and programmable hardware at low cost makes it feasible to create flexible programmable switches that can perform application-specific packet processing at line rates. By modifying the content of packets on-path, such a model enables tenants to efficiently build custom, application-specific network services for data aggregation, stream processing, caching and deduplication, which improve the performance of network-bound applications. Conserved network capacity can be made available to other tenant applications.

In order to understand the opportunities and challenges that in-network processing brings, we focus on a restricted form of in-network computation: *in-network aggregation* of data. In this section, we discuss the characteristics of in-network aggregation and explore, through a simulation study, the requirements that must be satisfied for such a service to benefit applications.

Overview In-network aggregation represents a clear departure from existing mainstream network technologies. Rather than removing network bottlenecks by increasing the bandwidth in the core of the network or at the end hosts to cope with the large amount of traffic generated during the aggregation phase, we propose to reduce this traffic by aggregating the data within the network.

This approach exploits the observation that a *partition/aggregation* pattern is at the core of many large-scale data centre applications. A partition/aggregation application consists of a set of *worker nodes*, deployed on *edge servers*. In addition, a *master node* for an application, such as a front-end server, acts as the entry point for requests into the application, dispatches partitioned work to the worker nodes and

aggregates partial results. For example, in MapReduce as used for the AdPredictor demonstrator [10], the input data are partitioned into small chunks (with a typical size of 64 t0 128 MB) and processed through a number of parallel map tasks (the *map* phase). The intermediate results are then sent to one or many reduce tasks (the *shuffle* phase) that perform the final step of aggregation (the *reduce* phase).

Partition/aggregation applications typically generate *many-to-few* network flows when transferring data. This network activity has a significant impact on application performance. In Facebook MapReduce jobs, network transfers on average are responsible for 33% of the execution time of jobs having a reduce phase, with 16% of those jobs accounting for more than 70% of the execution [5]. Similarly, in Microsoft Scope jobs, the network is responsible for a 62% increase in the reduce time in the median case [1].

Our approach is motivated by the observation that the vast majority of partition/aggregation applications running in today's data centres exhibit the following two properties:

1. aggregation functions required by data centre applications are typically *associative* and *commutative* [27] and

2. the size of the aggregated results is often a small fraction of the intermediate data generated [4, 6].

The first property implies that the aggregation process can be performed through a sequence of *partial* aggregations without affecting the correctness of the final result. Typical examples of aggregation functions exhibiting this property are `max`, `min`, `sum`, `count` and their derivatives. (`average`, for example, is not associative and commutative but can become so if expressed as `sum` and `count`.)

The second property shows that, in practice, there is significant redundancy in the data generated during the partition step. According to Dean and Ghemawat [6], the average final output size in Google jobs is 40.3% of the intermediate data set sizes. In the Facebook and Yahoo jobs analysed by Chen et al. [4], the reduction in size between the intermediate and the output data is even more pronounced: in 81.7% of the Facebook jobs with a reduce phase, the final output data size is only 5.4% of the intermediate data size, while for Yahoo jobs the number is 8.2% in 90.5%.

These two properties together indicate that performing in-network aggregation is both *feasible*, due to the associative and commutative property, and *useful*, due to the high redundancy observed.

We considered different approaches for performing in-network aggregation. A seemingly attractive option would be to perform this operation directly within the switch, such as by leveraging programmable hardware (e.g. NetFPGAs [21]) or exploiting software packet processing platforms [8, 24], as described in Section 2.2. This approach, however, would significantly increase deployment complexity. Switches and routers represent a key component of the data centre and are carefully selected based on a number of constraints, including cost, power consumption, reliability and form factor. It would be difficult (if possible at all) to replace the existing network infrastructure with custom aggregation devices.

In contrast, we propose to perform in-network aggregation by deploying aggregation nodes (agg nodes for short), directly attached to each of the switches, as shown in Figure 2.1. Packets belonging to partition/aggregation applications are steered by the switches toward the agg nodes, which aggregate them in a stream fashion and forward them to the next switch on route to the destination.

Requirements. For in-network aggregation, a key issue is the relationship between the processing throughput R of an agg node and the overall performance. To answer this question, we perform a sensitivity analysis using a flow-level simulator. This allows us to scale our experiments to large server counts and to explore the parameter space.

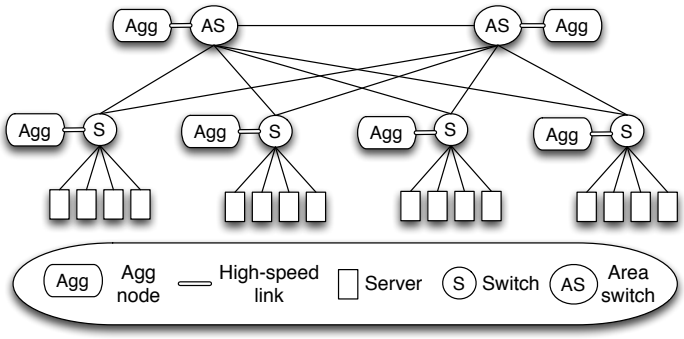

Figure 2.1: Data centre network topology with in-network aggregation.

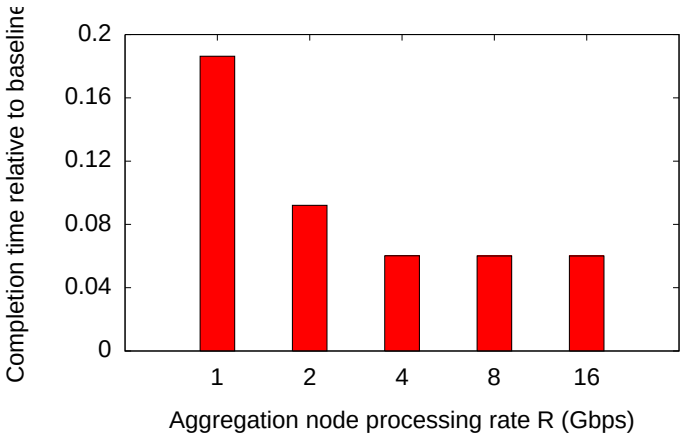

Figure 2.2: Reduction in total flow completion time for different values of the agg node processing throughput R.

We model a typical multi-rooted network topology with 8192 servers, similar to the one depicted in Figure 2.1. Each server is connected through a 1 Gbps link to the *top-of-rack switch* (ToR) switch. We conservatively assume an over-subscription ratio of 1:4, which is consistent with those reported in the literature [2, 14]. We consider a workload comprising 10000 flows. Based on the traces used by Benson et al. [2], we model the flow inter-arrival times using a Poisson process with a mean of 1 ms. The size of each flow is drawn by an exponential distribution with mean equal to 500 MB.

Figure 2.2 shows the reduction in the total flow completion time for different values of R. To model a generic application scenario, we assume that only 25% of the flows can be aggregated (*aggregation flows*). This accounts for scenarios in which partition/aggregation applications share the network with other applications, as well as for applications that generate a heterogeneous set of flows of which only a subset can be aggregated. As our baseline, we consider an overlay solution in which flows are aggregated at the rack level, before being sent to the final aggregator. This is representative of commonly adopted practices [13, 18, 20].

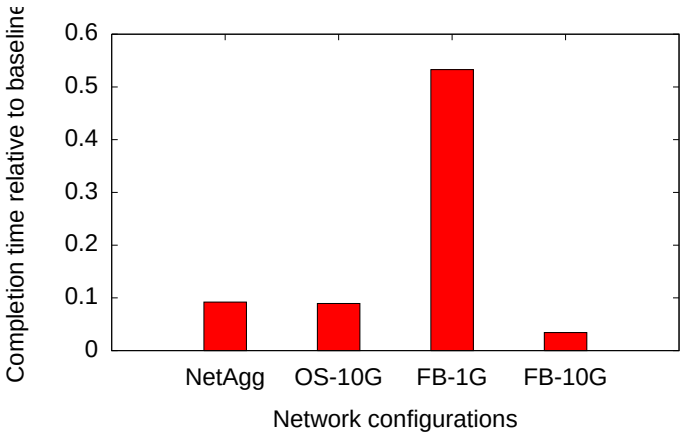

Figure 2.3: Performance of different designs.

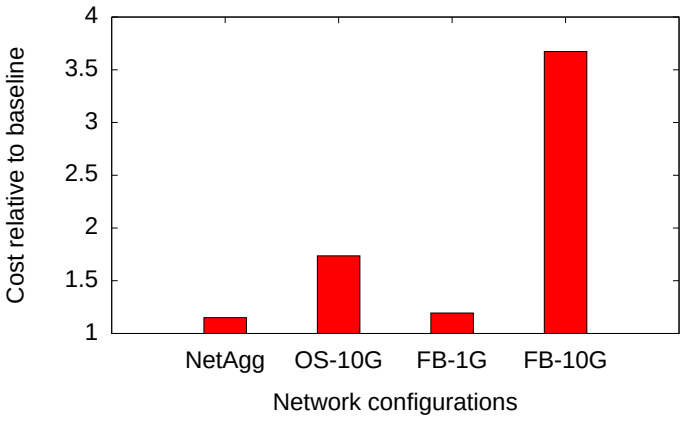

Figure 2.4: Cost of different designs.

Interestingly, even relatively low values of processing throughput R are sufficient to achieve significant benefits. For example, R = 1 Gbps is sufficient to reduce the total completion time by more than 80% (90.7% for R = 2 Gbps). This is an important result because it shows that these benefits can be reached using commodity hardware, without requiring expensive or custom-made solutions. As we show in Deliverable D4.4.1 [11], our agg node prototype is able to aggregate packets at a rate ranging from 1.5 to 4.8 Gbps that, based on the results presented here, are well above the minimum requirements.

Another interesting property of our solution is that, besides reducing the completion time of aggregation flows, it also reduces the completion time of *all* flows. For example, for R = 2 Gbps, the median completion time of all flows decrease by 66% (80% for the 95th percentile). The reason is that by reducing the size of the aggregation flows, more bandwidth becomes available for the other flows. This means that our solution can be beneficial even in shared clusters, where partition/aggregate applications co-exist along with other applications, whose flows do not hold the properties discussed at the beginning.

To complete our study of the characteristics of in-network aggregation, we perform a simple cost analysis aimed at understanding the trade off between the different options. In particular, we are interested in understanding the cost of deploying our agg node against the cost of either reducing the over-subscription (by deploying more switches) or increasing the end-host bandwidth (by using 10 Gbps links) or both.

Along with NETAGG (R = 2 Gbps), we consider three different alternative designs: a full-bisection data centre with 1 Gbps server connectivity (FB-1G); a 1:4 over-subscribed network with 10 Gbps server connectivity (OS-10G); and a full-bisection data centre with 10 Gbps server connectivity (FB-10G).

Accurately costing hardware equipment in data centres is hard because prices are typically the result of proprietary negotiations and not publicly disclosed. Further, even if these figures were available, they would soon be obsolete. However, as the purpose of this analysis is qualitative rather than quantitative, we use the prices adopted in a recently published study [23]. We ignore cabling costs and assume a hardware specification for servers and aggregation nodes as the one used in our testbed [11, 12]. Although the prices have changed since the publication of this study, we expect the relative costs between the devices to be similar and, therefore, we consider this a good starting point for our analysis.

Figures 2.3 and 2.4 show the performance and cost trade off of the aforementioned design options. NETAGG strikes a good compromise between cost and performance. For example, our solution achieves a factor of 91% improvement in performance, while only costing 1.15 times more. In contrast, FB-10G achieves the best performance (99.9% lower than baseline and 63% lower than NETAGG) but with a cost that 3.67 times higher than baseline.

3 Application Benefits

Next we discuss the benefits that the three HARNESS demonstrator applications can expect when deployed on a heterogeneous cloud platform that has access to the three heterogeneous network technologies, which we have studied as part of the first-year effort.

OLTP Acceleration. This demonstrator relies on a high-bandwidth low-latency interconnect between the database system and the off-loaded delta merge computation. Typically, the compute accelerator for the delta merge computation would be collocated on the same machine as the database system.

Due to these specific requirements, communication for this application would typically rely on the *peripheral component interconnect express* (PCIe) bus, which makes the described general-purpose network technologies less applicable.

Reverse Time Migration. This demonstrator application requires custom high-bandwidth network paths between storage and compute servers. This may include guaranteed bandwidth reservations in order to sustain a given transfer rate from storage to compute servers. The particular network paths chosen depend on the locations of the end points, which are selected as part of the overall resource allocation performed by the HARNESS platform.

Given the above requirements, **SDN** switches offer the network control to set up custom routing paths for transmitting large volumes of data between network end points. They permit fine-grained dynamic routing policies for scheduling traffic flows between virtual machines and other compute elements in a data centre. In the case of RTM, SDN switches can enable the HARNESS platform to offer the required network communication paths needed when the application is scheduled across different set of heterogeneous compute accelerators in the data centre.

A **software packet processing platform** can offer custom network services, implemented as middleboxes, that can be deployed quickly and easily. In the context of RTM, such middleboxes could be for transparent traffic shaping of network flows, enforcement of security policies through firewall, and the implementation of new routing protocols. New routing protocols could take the specific traffic patterns of RTM into account and realise custom routing policies that avoid communication bottlenecks and increase overall RTM communication bandwidth.

AdPredictor. The AdPredictor demonstrator is an example of a distributed data-flow computation that follows a partition/aggregate pattern, which can be aggregated within the network, as described in Section 2.3. Therefore, this demonstrator will benefit from **in-network aggregation** of data as part of the reduce phase in the map/reduce computation of AdPredictor. Rather than increasing the available bandwidth, in-network aggregation can *reduce the network traffic* by aggregating the data within the network. As opposed to aggregation at the edge of the network, this allows us to minimise server bandwidth usage as well as to reduce congestion in the core of the network.

Bibliography

[1] G. Ananthanarayanan, S. Kandula, A. Greenberg, I. Stoica, Y. Lu, B. Saha, and E. Harris. Reining in the outliers in map-reduce clusters using mantri. In *OSDI*, 2010.

[2] T. Benson, A. Akella, and D. A. Maltz. Network Traffic Characteristics of Data Centers in the Wild. In *IMC*, 2010.

[3] B. Chen and R. Morris. Flexible Control of Parallelism in a Multiprocesor PC Router. In *USENIX ATC*, 2001.

[4] Y. Chen, A. Ganapathi, R.Griffith, and R. Katz. The Case for Evaluating MapReduce Performance Using Workload Suites. In *MASCOTS*, 2011.

[5] M. Chowdhury, M. Zaharia, J. Ma, M. I. Jordan, and I. Stoica. Managing Data Transfers in Computer Clusters with Orchestra. In *SIGCOMM*, 2011.

[6] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. *Communications of the ACM*, 51(1), Jan. 2008.

[7] 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.

[8] 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.

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

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

[11] FP7 HARNESS Consortium. Experimental Network Prototype and Report (initial). Project Deliverable D4.4.1, 2013.

[12] FP7 HARNESS Consortium. Setup of integration, testing and demonstration testbed. Project Deliverable D7.1, 2013.

[13] Google Inc. Google tree distribution of requests. Available at `http://goo.gl/RpB45`.

[14] 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.

[15] 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.

[16] 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.

[17] E. Kohler, R. Morris, B. Chen, J. Jannoti, and M. F. Kaashoek. The Click Modular Router. *ACM Transactions on Computer Systems (TOCS)*, 18(3):263–297, 2000.

[18] D. Logothetis, C. Trezzo, K. C. Webb, and K. Yocum. In-situ MapReduce for Log Processing. In *USENIX ATC*, 2011.

[19] 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.

[20] S. Melnik, A. Gubarev, J. J. Long, G. Romer, S. Shivakumar, M. Tolton, and T. Vassilakis. Dremel: Interactive Analysis of Web-scale Datasets. In *VLDB*, 2010.

[21] J. Naous, G. Gibb, S. Bolouki, and N. McKeown. NetFPGA: Reusable router architecture for experimental research. In *Processing of the International Workshop on Programmable Routers for Extensible Services of Tomorrow*, 2008.

[22] Open Networking Foundation. OpenFlow. Available at `http://www.openflow.org/`.

[23] L. Popa, S. Ratnasamy, G. Iannaccone, A. Krishnamurthy, and I. Stoica. A Cost Comparison of Data Center Network Architectures. In *CoNEXT*, 2010.

[24] L. Rizzo. Netmap: A Novel Framework for Fast Packet I/O. In *USENIX ATC*, 2012.

[25] V. Sekar, N. Egi, S. Ratnasamy, M. K. Reiter, and G. Shi. Design and Implementation of a Consolidated Middlebox Architecture. In *NSDI*, 2012.

[26] Why use Vyatta? Available at `http://www.vyatta.org/getting-started/why-use`.

[27] Y. Yu, P. K. Gunda, and M. Isard. Distributed aggregation for data-parallel computing: Interfaces and implementations. In *Proceedings of the ACM Symposium on Operating Systems Principles*, 2009.