AEGLE's Cloud Infrastructure for Resource Monitoring and Containerized Accelerated Analytics

K. Koliogeorgi[1], D. Masouros[1], G. Zervakis[1], S. Xydis[1], T. Becker[2], G. Gaydadjiev[2], D. Soudris[1]
[1]National Technical University of Athens, Greece
{konstantina, demo.masouros, zervakis, sxydis, dsoudris}@microlab.ntua.gr
[2]Maxeler Technologies, UK
{tbecker, georgi}@maxeler.com

***Abstract*—This paper presents the cloud infrastructure of the AEGLE project, that targets to integrate cloud technologies together with heterogeneous reconfigurable computing in large scale healthcare systems for Big Bio-Data analytics. AEGLEs engineering concept brings together the hot big-data engines with emerging acceleration technologies, putting the basis for personalized and integrated health-care services, while also promoting related research activities. We introduce the design of AEGLE's accelerated infrastructure along with the corresponding software and hardware acceleration stacks to support various big data analytics workloads showing that through effective resource containerization AEGLE's cloud infrastructure is able to support high heterogeneity regarding to storage types, execution engines, utilized tools and execution platforms. Special care is given to the integration of high performance accelerators within the overall software stack of AEGLE's infrastructure, which enable efficient execution of analytics, up to 140× according to our preliminary evaluations, over pure software executions.**

I. Introduction

The integration of the Information and Communication Technology (ICT) into Healthcare and Bio-medical domains has always been a challenging task. Although significant progress has been made in the Big Data domain, there is still much to be explored in the area of big data analytics for Health Bio-data. Data versatility, volume, velocity and veracity within the whole data value chain of healthcare analytics render essential the need for data-driven services that can handle the dramatically increasing heatlhcare data generation. In addition, the existence of an automated solution that could support effective storage, processing and visualization of huge amounts of data, would enable the provision of integrated and personalized healthcare services.

However, the healthcare industry appears hesitant to embrace Big Data Technologies as a solution to data volume and velocity issues that arise. Several European initiatives[1] have already pinpointed the importance and usefulness of healthcare big data, e.g. to predict the outbreak of an epidemic etc. Additionally, business interest is growing, as in the case of Open Data initiative, where big health data providers, governmental and research institutes as well as industry aim to develop a vendor-neutral Big-Data platform[2]. However, the strategic advantage brought by Big-Data in healthcare still materializes at slow paces, as only some large-scale organizations have established few pilot or proof-of-concept projects.

Nowadays, there is a gap in the area of big data analytics for Health Bio-data. The AEGLE project targets to address the aforementioned open issues by implementing a full data value chain to create new value out of rich, multi-diverse, big health data. The project builds upon the synergy of heterogeneous high performance reconfigurable architectures, Cloud and Big Data computing technologies. Recently, heterogeneous datacenter infrastructures [1] through tight coupling and integration of customized accelerators have emerged either through the proposition of specialized accelerator-based programming models [2], [3] or through accelerators integrated within big data engines [4]. However, the efficient resource management and seamless integration of customized accelerators are still open issues in the heterogeneous datacenter design.

In this paper, we introduce the design of AEGLE's accelerated infrastructure along with the corresponding software and hardware acceleration stacks to support various big data analytics workloads. We show how AEGLE's infrastructure through the effective services and resources containerization is able to support high heterogeneity regarding to storage types, execution engines, utilized tools and execution platforms (physical infrastructures). Specifically, we analyze in detail the mechanisms that enable the instantiation and integration of three different types of clusters, i.e. *storage servers cluster*, *high performance execution cluster*, and *Maxeler accelerated cluster*, forming the integrated AEGLE cloud infrastructure. Special focus is given on the integration of Maxeler accelerated services within the overall software stack of AEGLE's infrastructure. The mechanisms utilized to achieve efficient communication between analytics written in high level languages such as R and Python and accelerated kernels are presented, as well as the mechanisms that enable the overall accelerator's deployment, provisioning and management. Our preliminary experimental results over a representative kernel used in predictive machine learning show the efficiency and

[1]http://ec.europa.eu/digital-agenda/en/news/big-data-what-it-and-why-it-important

[2]www.healthstartup.eu/blog/top-big-data-opportunities-for-health-startups

scalability of the proposed accelerated infrastructure, achieving speedups of 140× over pure software executions.

II. Overview of AEGLE EU funded project

AEGLE, a European funded project under the H2020 program, aims to promote data-driven research across Europe and serve as a platform that enables and encourages the use of different data technologies, by providing a framework that can create new value out of rich, multi-diverse, big health data. AEGLE will develop and provide a Big Data Analytics infrastructure that will deliver integrated and personalized health-care services, available to data scientists and medical researchers across Europe.

A. AEGLE use cases

AEGLE system will be validated over the following use case scenarios, that are currently under development:

- **Chronic Lymphocytic Leukemia (CLL):** CLL is a chronic, incurable disease, that causes great distress to patients and their families and creates huge costs for the health care system. AEGLE offers research ground for developing new, personalized treatment plans while also exploring the resulting medical costs. AEGLE's CLL analytic pipelines will offer the possibility to propose and evaluate treatment plans, while at the same time evaluate the potential cost.
- **Intensive Care Unit (ICU):** In an ICU context, patient biosignals are continuously monitored and displayed to detect alerting events and notify the medical stuff. The recordings of clinical and laboratory data as well as physiologic waveforms can be analysed and displayed in an easy-to understand manner for clinicians. AEGLE aims to provide a set of scalable and automated analyses of these functions to detect unusual, unstable or deteriorating status of patients and thus predict and eventually prevent severe episodes.
- **Type 2 Diabetes (T2D):** AEGLE T2D ecosystem will focus on analysing the inter-dependencies of the factors that are known to have a detrimental effect in type 2 diabetes, including medication, and develop analytics that provide an estimate of potential deterioration. This would allow for the development of a personalized therapeutic schema that reduces mortality rate, complications and hospitalization time, which in turn reduces the overall health costs.

III. AEGLE's Big Data Framework

AEGLE's Big Data Framework will comprise of i) a big data storage platform hosting a large set of anonymized and verified clean healthcare data, ii) a repository of both software and hardware accelerated libraries implementing state-of-the-art big data analytics methods, iii) an advanced resource management framework in order to effectively provide its services as well as efficiently execute and monitor its workloads, and iv) an interface enabling people of the non ICT domain to perform complex analyses and visualize their results and the available data with ease of use. AEGLE will provide the user with a rich repository of different analytics and tools, enabling efficient and fast research on the medical domain.

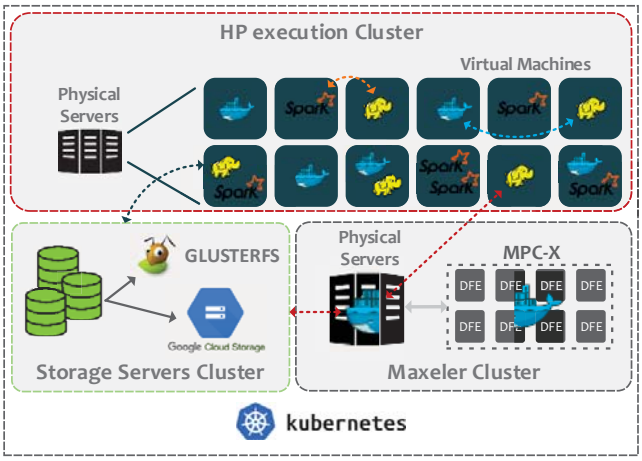

Fig. 1. Overview of AEGLE's infrastructure

AEGLE's platform is characterized by high heterogeneity, since it supports many execution engines and platforms (conventional and hardware accelerated clusters), and satisfies varying requirements, e.g. efficient and reliable data storage, high data availability, high performance analytics execution and hardware accelerated execution. AEGLE analytics are divided into two categories, conventional and accelerated analytics. The conventional analytics are developed over state-of-the-art Big Data platforms, e.g. Hadoop [5], Spark [6], and the hardware accelerated analytics are developed and executed using the Maxeler's DataFlow acceleration engines [7]. However, in addition to the developed analytics, AEGLE's platform will support a variety of existing analytics, tools, and execution engines, that are widely used in the medical research domain, e.g., SeqMule [8], TopHat [9]. To manage heterogeneity, AEGLE cloud infrastructure instantiates three different types of clusters, i.e. *storage servers cluster*, *high performance execution cluster*, and *Maxeler accelerated cluster*. In order to effectively integrate these clusters and schedule-monitor the different workloads, all the AEGLE services and libraries are dockerized [10] providing efficient resource management and orchestration through Kubernetes [11].

A. AEGLE's Heterogeneous Cluster Infrastructure

Fig. 1 shows the overview of AEGLE's infrastructure. Execution and data storage machines are decoupled inside AEGLE's framework due to the different requirements that characterize the analytics related features. The variability of analytics introduces further requirements which impose that AEGLE must support both conventional software execution as well as hardware accelerated execution. Therefore, with respect to the physical resources, three different types of clusters are provisioned, i.e. *storage servers cluster*, *high performance execution cluster*, and *accelerated cluster*, respectively. Storage optimized machines are used as data servers

and computation-optimized/hardware-accelerated machines as execution servers. Moreover, this separation enables higher and personalized services scalability, more efficient resource management, and data sharing across the accelerated and conventional execution clusters. Depending on the workload and the storage needs, the respective resources can be resized accordingly. Especially for the execution clusters, the utilized resources can be kept to minimum when in idle condition, leading to more power and cost efficient solutions.

AEGLE's *high performance execution cluster* consists of either physical or virtual machines. The software stack of AEGLE's Big Data Framework can be deployed on both i) an execution optimized cluster obtained by a cloud provider or ii) the available local infrastructure. Regarding the local infrastructure, AEGLE's Big Data framework creates a virtual cluster by automatically deploying virtual machines running CoreOS on the available physical servers [12]. Regarding the *storage cluster*, AEGLE's Big Data framework supports multiple storage solutions, enabling its deployment in both private local and cloud infrastructures. Using Kubernetes, AEGLE's framework exposes storage drivers for the most popular online cloud providers, i.e., Google Cloud Storage, Microsoft Azure and Amazon Web Services. In the case of local private infrastructures, the *storage cluster* is implemented using the GlusterFS [13] network file-system and the respective Kubernetes storage driver. GlusterFS is an open source software that provides distributed storage solutions intended for use in user space. It provides great flexibility, data elasticity, volumes grouping, data replication and re-balancing and can be deployed on commodity hardware synthesizing a fast file system.

Finally, AEGLE's acceleration cluster consists of conventional servers connected through Infiniband with Maxeler MPC-X servers[14]. Maxeler provides complete platforms for high-performance computing based on dataflow technology. This approach enables applications to offload the compute-intensive parts to dedicated Dataflow Engines (DFEs) and typically delivers significant improvements in performance/Watt and performance/space. AEGLE's libraries and software stack run on the conventional servers of the *Maxeler cluster*, whereas the parts requiring acceleration are executed on the MPC-X servers. The MPC-X node utilized in AEGLE framework belongs to the current generation MAX4 DFE architecture. MPC-X nodes are based on the concept of using dedicated dataflow nodes and linking them with standard CPU-only servers. They contain only DFE cards and no CPUs. A single node holds 8 MAX4 DFE cards into a dense 1U industry-standard chassis. Each DFE card contains 48 GB of DRAM as LMEM and DFEs are directly connected through MaxRing in a bidirectional 1D array topology. DFEs are also reachable through the Infiniband from a standard x86 server. This allows a various number of DFEs to be allocated dynamically.

A conventional server is directly connected with the MPC-X through two 56Gb/s Infiniband cards. This architecture, as illustrated in Fig. 2, uses the Infiniband switch between the MPC-X node and the CPU nodes and, thus, is ideal for dynamic applications with flexible allocation or multi-tenant access to DFEs. Therefore, the use of two Infiniband cards doubles the available bandwidth and the switched architecture allows all hosts to have access and use the MPC-X node.

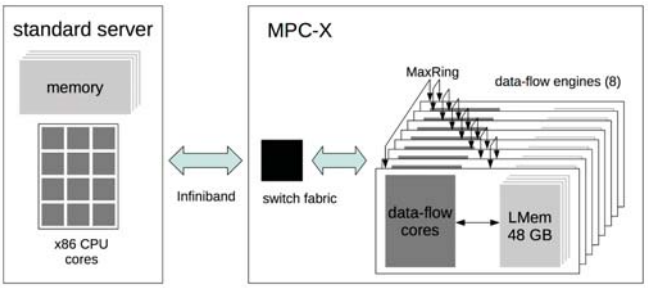

Fig. 2. Structure of a Maxeler MPC-X dataflow node consisting of 8 dataflow engines. The system is connected to a standard CPU server via Infiniband.

B. AEGLE's Software Stack

The Software stack is composed of several tools and libraries which facilitate the needs of distributed data storage and distributed analytics execution. As far as the distributed data storage and processing are concerned, Apache Hadoop [5] 2.7.2 is utilized. Hadoop consists of two major parts, the Hadoop Distributed File System (HDFS) and the MapReduce execution engine. Distributed execution of analytics is achieved through Apache's Spark [6] framework, which provides an interface for programming entire clusters with implicit data parallelism and fault tolerance. The aforementioned engines constitute the major part of the AEGLE's software stack, however, all types of software solutions can be executed over AEGLE's Big Data Framework, e.g., the SeqMule [8] and TopHat [9] tools are included in AEGLE's software libraries.

Targeting infrastructure independence, fast deployment and scaling, as well as security resources provisioning and flexibility, every service and library of AEGLE's Big Data Framework runs inside a different docker container [10] and the respective docker containers are deployed on the AEGLE's hardware stack (clusters of physical/virtual machines with or without accelerators). In order to organize, manage, and coordinate AEGLE's services and resources, Kubernetes [11] is adopted as the main orchestrator of AEGLE's software stack.

Fig. 3 depicts the main software components defining the Big Data Framework of AEGLE. The major components, are briefly described below:

- **Kubernetes**: It provides an open source system for automating deployment, scaling and management of containerized applications.[11]
- **HDFS2**: It implements the Hadoop distributed file system efficient data storage, and provides access and resilience in AEGLEs cloud.[5]
- **YARN**: It implements the actual resource manager of the HDFS cluster.[15]

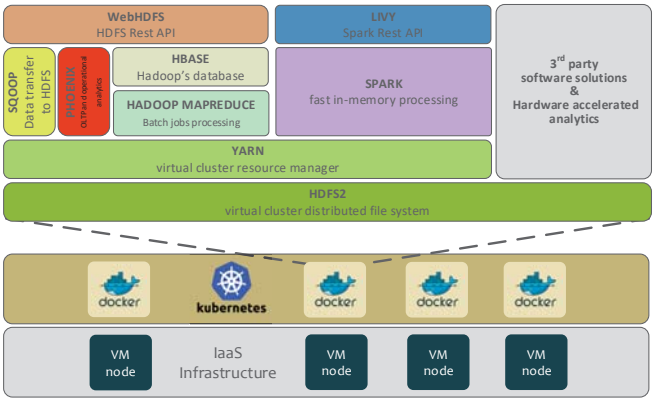

Fig. 3. AEGLE's Software stack components

- **Hadoop MapReduce**: It constitutes the Hadoop framework implementing the map-reduce programming model for parallel and scalable data processing.[5]
- **HBase**: It implements random, realtime read/write access to Big Data by providing Bigtable-like capabilities on top of Hadoop and HDFS.[16]
- **Sqoop**: It implements an API for efficiently transferring data in/out of HDFS from third party data sources, e.g. external SQL databases[17].
- **Phoenix**: It enables Online transaction processing and operational analytics in Hadoop for low latency applications.[18]
- **Spark**: It constitutes a scalable execution engine supporting data caching, also implementing the mapreduce programming model, for data intensive workloads, more specialized for iterative applications.[6]
- **Livy**: It implements a REST interface for interacting with SPARK from anywhere. It supports executing snippets of code or programs in a SPARK context that runs locally or in Hadoop YARN.[19]

C. AEGLE's Integrated Maxeler Acceleration Services

The accelerated services provided by the AEGLE Infrastructure constitute one of its main contributions and advantages in comparison to other similar platforms. Therefore, it is essential to elaborate on the technical aspects of the integration of the Maxeler infrastructure into the overall framework.

The first step towards developing a dataflow application is splitting the application into a compute intensive part that runs on the DFE and a CPU host application that carries out control operations and sets up execution on the DFE. The host application is developed in C, C++ or other conventional programming languages whereas the DFE application is described in a dataflow language called MaxJ (a Java-based metaprogramming approach). A compiler, MaxCompiler, performs the scheduling of operations and the balancing of the data paths inside a kernel. Maxeler provides an application programming interface, called SLiC (Simple Live CPU) interface, for seamless CPU-DFE integration. Once the development phase has been completed and all necessary validation tests have been successfully passed, the accelerator can be built so that it executes on actual hardware.

However, since AEGLE services target the health domain and address medical and data experts, most required analytics are developed in languages such as R and python that combine features suitable for data analysis as well as for statistical computing and data visualization. The Maxeler programming model provides basic SLiC Skins so that DFEs can be integrated into applications/libraries written in these languages. For advanced interface purposes and complicated data manipulation of the input stream, the proposed framework introduces an intermediate step of calling a C function to actually invoke the accelerator. In both cases, the C implementation of the communication between CPU and DFE remains fairly the same. The only prerequisite is that inputs are passed as pointers and returned values are passed by reference. The C source code is compiled using the necessary flags to produce position independent code and the resulting C and maxfile objects are compiled into a shared library. Both python and R provide build-in functions to load this library and eventually invoke the accelerator through it.

D. Deploy AEGLE's Services with Kubernetes Provisioning

All three AEGLE's clusters are managed by the Kubernetes orchestrator. Kubernetes can automatically deploy, manage and scale containerized applications (Pods) into the cluster nodes. Horizontal scaling can be achieved either with a simple command by the developer, or automatically based on resources usage. In addition, Kubernetes provides fault tolerant capabilities by restarting containers that fail, replacing and rescheduling containers when nodes fail, killing containers that do not respond to user-defined health check, and by not exposing them to clients until they are ready to serve. Moreover, Kubernetes enables the communication of containers running in the same or different clusters without exposing them to the world (if not needed) providing, thus, high security levels. Finally, Kubernetes provides a resource manager (Resource Quality of Service) for the containers. Every time new containers (Pods) request for resources, Kubernetes computes the available resources per cluster node and decides about which node to place Pods on. When a pod is successfully scheduled, the container is guaranteed the amount of resources requested.

The deployment of AEGLE's Big Data Framework is summarized as follows. Firstly, Kubernetes is deployed using Kargo [20] on all the nodes of the *high performance execution cluster* and the conventional nodes of the *Maxeler cluster*. If the *storage cluster* is not hosted on a cloud provider, then Kubernetes is also deployed on the *storage cluster* nodes. In the latter case, after Kubernetes is deployed, Kubernetes deploys containerized GlusterFS in the *storage cluster*. Secondly, Kubernetes deploys Hadoop containers in the *high performance execution cluster*. For data persistency, Kubernetes creates the respective volumes at the *storage cluster* and mounts the HDFS and Hadoop Namenode data folders of the containers to them. The Hadoop containers are deployed as Kubernetes Stateful Sets, obtaining unique identifiers. As a result, even

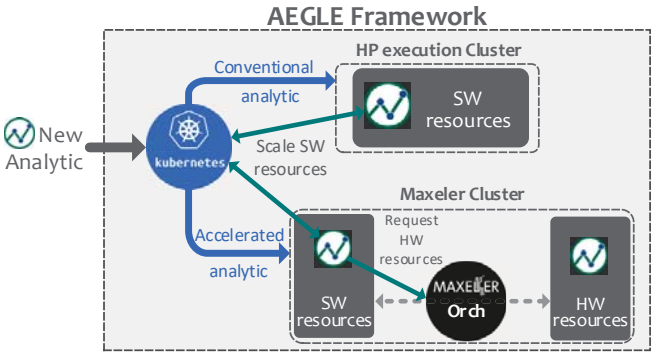

Fig. 4. Analytic execution in AEGLE's Big Data Framework

if a Hadoop container fails, Kubernetes will deploy a new one to replace it with the same exact identity and therefore, the new container will be mounted to the same volume that was previously created in the *storage cluster*, retaining the same already stored data. Thirdly, Kubernetes deploys all the containers implementing the AEGLE services, e.g. user interface, user authentication, data upload, visualization etc. Finally, note that, the containers implementing the AEGLE's analytics, e.g. MapReduce analytics, analytics running over Spark, SeqMule etc., are deployed on demand and read/write data from/to the Hadoop HDFS containers.

Fig. 4 depicts the procedure followed in order to execute an analytic in the AEGLE's Big Data Framework. Every time a user selects in AEGLE's user interface to execute an analytic tool/workflow, the user interface submits the respective configuration file to Kubernetes. In this configuration file all the relative parameters are specified, i.e. the container implementing the analytic, the number of the containers, and the amount of required resources for its execution. The latter have been obtained through offline profiling of the analytics of the AEGLE's libraries. For example, a conventional analytic could require a Spark cluster with 20 workers and 16GB RAM per container. Then, Kubernetes finds the *execution cluster* nodes that can serve the requested resources and deploys the containers on these nodes. Finally, if the user wants to execute an analytic that contains an accelerated part, Kubernetes finds the nodes of the *Maxeler cluster* that can allocate the requested "software" resources (CPU threads and RAM memory) and deploys the containers on these nodes. Following, since the containers are deployed, the accelerated analytic requests DFEs from the Maxeler Orchestrator and the latter responds with the location of DFEs requested (DFEs identifier). After obtaining the requested DFEs, the container running the analytic communicates directly with the respective DFEs via Infiniband network, transfers the data inputs to the DFEs and receives the outputs. Finally, considering that all the conventional servers of the *Maxeler cluster* are connected to all the MPC-X machines via Infiniband switch and that their requests for DFEs are granted by the Maxeler Orchestrator, Kubernertes needs to be aware only of the available "software" of the conventional nodes, in order to schedule efficiently the accelerated analytics containers.

IV. Experimental Results

The described infrastructure is validated through the execution of an acceleration task, since it runs on all levels of components. The scalability in respect to dataset size and the overhead of multiple DFEs operating simultaneously are also examined.

Regarding the experimental setup, AEGLE's acceleration cluster consists of one Intel Xeon CPU E5-2658A server that operates on 2.20GHz frequency. The server is connected with a Maxeler MPC-X server [14] via two 56Gb/s Infiniband cards. MPC-X includes 8 MAX4 (Maia) data-flow engines, each of which contains 48 GB of DRAM as LMEM.

As a use case, we demonstrate the result for accelerating Gaussian Process Regression from scikit-learn Python library. Gaussian Process Regression includes both training and prediction phases that utilize the same computational kernels. In [21] a detailed performance analysis of these methods is conducted in order to determine the computationally intensive code regions subject to acceleration. As candidates for acceleration, we consider the matrix multiplication and spatial distance kernels that consume around 50% of training and prediction phases runtime. The tiled accelerated design [22] of matrix multiplication is used for both these kernels. These two kernels use different arithmetic operators between array elements. For matrix multiplication only the product is computed, whereas for *pdist()* the squared difference of the same elements is calculated.

In the examined the kernels, the tile size is set to 144 elements. The clock frequency for synthesis, placement and routing is 100 MHz. The dimensions of the input matrices scale from 600×400 to 12000×8000. The latency required for Kubernetes to schedule and start the docker containers of the accelerated kernels (on a conventional node of the *Maxeler cluster*), is measured up to 2 seconds. Targeting Kubernetes scheduling performance, all the docker containers are pre-pulled in all the host machines. Fig. 5 illustrates the speedup in execution time between the DFE and CPU implementations of the examined applications. It is shown that the attained speedup is remarkably high, reaching even $140\times$. The designs also reports high scalability in respect to the increased input dataset sizes.

An interesting parameter to examine is the overhead introduced when more than one DFE is configured and operating in parallel. This is very crucial for the framework, so that multiple users can execute multiple accelerated analyses simultaneously, without severe latency issues. For that purpose, we instantiate many DFEs with the accelerators described above. For each number of operating DFEs, the average execution time of the accelerators is taken into account. Fig.6 illustrates the relative execution time of each case in comparison to the time required when only one DFE is used, i.e. without any interference.

The execution time does not increase considerably when the number of DFEs remains less or equal to 4. For 8 simul-

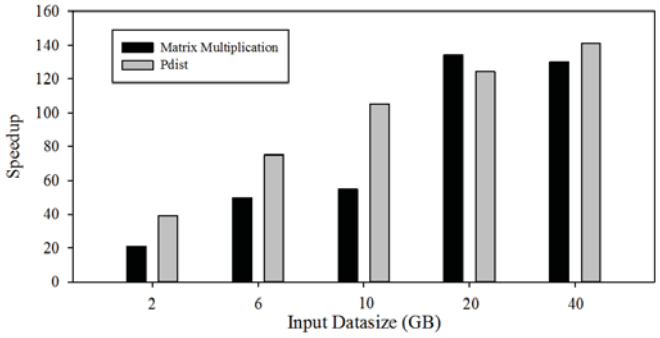

Fig. 5. Speedup of the DFE execution of the Matrix Multiplication and Pdist kernels over their respective software implementations for scaled input size.

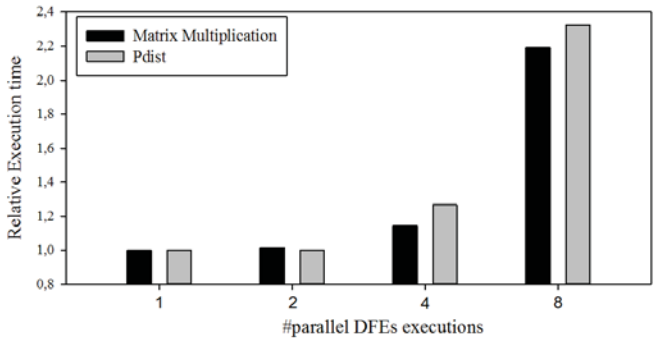

Fig. 6. Relative execution time of parallel DFE executions in respect to single DFE execution for Matrix Multiplication and Pdist kernels. The input datasize is 25GB.

taneously operating DFEs, the execution time doubles, thus introducing a significant overhead. The overhead is attributed to the increased traffic congestion in the Infiniband, that needs to transfer 25GB for each operating DFE. Even in the latter case though, 8 instances of an accelerator execute in parallel 60 times faster than a single instance in a conventional server.

V. Conclusion

In this work the accelerated infrastructure for AEGLE EU funded project was presented. The framework consists of both software and hardware acceleration stacks to support various big data analytics workloads, essential in the bio-medical and health-care domain. Specifically, three different types of clusters have been integrated into the framework, i.e. *storage servers cluster*, *high performance execution cluster*, and *Maxeler accelerated cluster*. The resources and services provided by this stack have been containerized to allow heterogeneity and variety regarding all these three respects. Results of verifying the functionality and efficiency of the framework, targeting accelerators for Gaussian Process Kernels, were very encouraging. The framework exhibits remarkable speedups (reaching 140×) and high scalability in respect to increased dataset sizes and acceptable overhead regarding parallel execution of accelerated kernels.

VI. Acknowledgements

This research is supported by the E.C. funded program AEGLE www.aegle-uhealth.eu under H2020 Grant Agreement No:644906 (H2020-ICT-2014-1 call). The duration of the project is 42 months and start date is March 1st 2015.

References

[1] Jason Cong, Muhuan Huang, Di Wu, and Cody Hao Yu. Invited - heterogeneous datacenters: Options and opportunities. In *Proceedings of the 53rd Annual Design Automation Conference*, DAC '16, pages 16:1–16:6, New York, NY, USA, 2016. ACM.

[2] Fei Chen, Yi Shan, Yu Zhang, Yu Wang, Hubertus Franke, Xiaotao Chang, and Kun Wang. Enabling fpgas in the cloud. In *Proceedings of the 11th ACM Conference on Computing Frontiers*, CF '14, pages 3:1–3:10, New York, NY, USA, 2014. ACM.

[3] S. A. Fahmy, K. Vipin, and S. Shreejith. Virtualized fpga accelerators for efficient cloud computing. In *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 430–435, Nov 2015.

[4] Yu-Ting Chen, Jason Cong, Zhenman Fang, Jie Lei, and Peng Wei. When apache spark meets fpgas: A case study for next-generation dna sequencing acceleration. In *Proceedings of the 8th USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'16, pages 64–70, Berkeley, CA, USA, 2016. USENIX Association.

[5] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. The hadoop distributed file system. In *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, MSST '10, pages 1–10, Washington, DC, USA, 2010. IEEE Computer Society.

[6] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. In *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'10, pages 10–10, Berkeley, CA, USA, 2010. USENIX Association.

[7] Maxeler technologies. http://www.maxeler.com.

[8] Yunfei Guo, Xiaolei Ding, Yufeng Shen, Gholson J Lyon, and Kai Wang. Seqmule: automated pipeline for analysis of human exome/genome sequencing data. *Scientific reports*, 5:14283, 2015.

[9] Cole Trapnell, Lior Pachter, and Steven L. Salzberg. Tophat: discovering splice junctions with rna-seq. *Bioinformatics*, 25(9):1105, 2009.

[10] Dirk Merkel. Docker: Lightweight linux containers for consistent development and deployment. *Linux J.*, 2014(239), March 2014.

[11] Brendan Burns, Brian Grant, David Oppenheimer, Eric Brewer, and John Wilkes. Borg, omega, and kubernetes. *Queue*, 14(1):10:70–10:93, January 2016.

[12] Georgios Zervakis, Sotirios Xydis, and Dimitrios Soudris. Performance-power exploration of software-defined big data analytics: The aegle cloud backend. In *Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS), 2016 International Conference on*, pages 312–319. IEEE, 2016.

[13] Alex Davies and Alessandro Orsaria. Scale out with glusterfs. *Linux J.*, 2013(235), November 2013.

[14] Mpc-x series, maxeler technologies. https://www.maxeler.com/products/mpc-xseries/.

[15] Vinod Kumar Vavilapalli, Arun C. Murthy, Chris Douglas, Sharad Agarwal, Mahadev Konar, Robert Evans, Thomas Graves, Jason Lowe, Hitesh Shah, Siddharth Seth, Bikas Saha, Carlo Curino, Owen O'Malley, Sanjay Radia, Benjamin Reed, and Eric Baldeschwieler. Apache hadoop yarn: Yet another resource negotiator. In *Proceedings of the 4th Annual Symposium on Cloud Computing*, SOCC '13, pages 5:1–5:16, New York, NY, USA, 2013. ACM.

[16] The Apache Software Foundation. Welcome to hbase! http://hbase.apache.org, 2010.

[17] Ankit Jain. *Instant Apache Sqoop*. Packt Publishing, 2013.

[18] The Apache Software Foundation. Apache phoenix. https://phoenix.apache.org/, 2017.

[19] Welcome to livy. https://github.com/cloudera/livy.

[20] Setup a kubernetes cluster. https://github.com/kubernetes-incubator/kargo, 2016.

[21] Michail Doukas, Sotirios Xydis, and Dimitrios Soudris. Dataflow acceleration of scikit-learn gaussian process regression. In *PARMA-DITAM*, 2017.

[22] Maxpower library. https://github.com/maxeler/maxpower, 2016.