Heterogeneous Servers based on Programmable Cores and Dataflow Engines

Yun Wu,Charles Gillan,Umar Minhas,Sakil Barbhuiya,
Alexsandar Novakovic, Kostas Tovletoglou, George Tzenakis,
Hans Vandierendonck,Georgios Karakonstantis,Dimitrios Nikolopoulos
Queen's University Belfast
yun.wu,c.gillan, u.minhas, sbarbhuiya03@qub.ac.uk
a.novakovic, ktovletoglou01, gtzenakis01@qub.ac.uk
h.vandierendonck,g.karakonstantis,d.nikolopoulos@qub.ac.uk

ABSTRACT

The continuous growth of the internet, and the data volumes associated to it, put severe pressure on the computational, storage and networking resources available in today's data-centers. The stringent power budgets and the difficult expansion of current Internet communication infrastructures necessitates the design of new server architectures, programming models and run-time systems. In this paper, we present promising options in the design of energy-efficient servers based on programmable accelerators which can help push the limits on the computation of future data centers. We show that one of the proposed micro-server prototypes based on custom tiny cores can achieve 40% better energy-efficiency than a standard Xeon Server, while dataflow engines can help speedup execution by up to 374x for various workloads.

Keywords

Big Data; IoT; Data Center; Heterogeneous Computing; Accelerators; Dataflow Engines

1. INTRODUCTION

The rapid growth over recent years in the number of devices being connected to the Internet is driving a rapid upscaling in the amount of the data that are being both generated and communicated to data centers [5]. Presently, we rely on massive data centers that contain tens of thousands of servers equipped with multiple cores and huge amounts of memory for processing and storing those data. However, as the Internet-of-Things (IoT) unfolds, the number of smart connected devices will increase dramatically and it will become doubtful if we can keep servicing the associated data volumes by merely scaling up the current infrastructure.

In fact, it is difficult to imagine how to transfer these data over the existing public networks and satisfy the low response latencies of many emerging applications without further substantial investments to expand the existing networks, which might take years to complete if at all technologically feasible. Such a challenge have urged designers to rethink the Cloud computing model leading to the introduction of a new paradigm of Edge computing according to which data are being pre-processed at the edge of the Clouds close to the users. Such a paradigm can help alleviate the pressure of high data volume and fast response latency from data centers. However, to enable such a paradigm new servers are needed that can easily be deployed and maintained at edge environments and are much more energy efficient than the classical ones.

In fact, new more energy efficient server architectures are needed not only for enabling the Edge Computing paradigm but also for continuing supporting the Cloud Computing since due to the utilization wall that has been recently discovered and which hinders the up scaling of the server processing capabilities. In particular, studies have shown that such a utilization wall puts a limit to the number of transistors within a chip that can be powered up at full speed, a phenomenon called Dark Silicon [6]. The scale of the challenge is so dramatic that some in the industry started [12] to warn that by 2020 the packing of billions of transistors and the very tight power budgets may permit to be active only nine percent of the on-chip transistors at any point in time. Such claims have elevated power as a prime design parameter making apparent that in order to improve performance by using more resources or operating at a higher frequency, new server architectures are needed to provide the required levels of energy efficiency.

To address such a challenge, new heterogeneous server architectures based on energy-efficient programmable accelerators on Field Programmable Gate Arrays (FPGAs) [10, 11] and/or embedded-processors [7] have been introduced. Such heterogeneous architectures may have the potential to achieve energy-efficient computation at latencies, throughput and density that exceed that of general-purpose processors energy efficiency but there is still a need to quantify the achieved trade-offs while rethinking the hardware-system software interfaces for enabling efficient utilization of the heterogeneous resources.

In this paper, we discuss our research efforts, challenges and results on the design and software integration of heterogeneous server architectures, based on reconfigurable custom processors, referred to as nanocores as well as on programmable dataflow engines.

In particular, in Section 2 we discuss a new micro-server architecture based on reconfigurable tiny cores optimized for FPGAs and on a novel bare-metal Ethernet networking infrastructure, which achieves low latency access and sharing of accelerators without affecting the host processor architecture.

In Section 3, we discuss another option for the implementation of the accelerators based on data flow engines and analyze the results and the open challenges in enabling their

utilization by traditional data center programming frameworks.

Finally, conclusions are being drawn in Section 4.

2. SERVERS BASED ON RECONFIGURABLE CORES

2.1 System Architecture

One strategy in our work is to explore a scale-out microserver architecture that can serve the emerging edge computing ecosystem, namely the provisioning of advanced computational, storage, and networking capability near data sources to achieve both low latency event processing and high throughput analytical processing. The research targets the challenges of processing streams of data in real-time by using ARM based microservers and developing an Analytics-on-Chip (AoC) architecture based on Xilinx Zynq-7000 All Programmable SoC family which functions as an accelerator attached to these.

The AoC uses an amalgam of low-power RISC processors for the embedded systems domains and Nanocores, a new class of programmable compute units. The AoC processor is a heterogeneous SoC that reduces latency in processing of streaming operators issued to the micro-server with the latency-optimised RISC cores, while improving analytical processing throughput on compute and data intensive tasks with the Nanocores.

Complementing the AoC architecture is a low latency communication protocol which we call NanoWire. This enables communication between CPU hosts, either microserver or server class systems, and the AoC engines. Recognising that Ethernet is now a ubiquitous interconnect technology, NanoWire is enabled directly on the Ethernet layer. This means that it benefits from the scalability of Ethernet technology and also that one accelerator, enabled for NanoWire communications, can be easily shared between many hosts in the datacentre.

In the next two subsections we describe the Nanocore and NanoWire respectively. We then describe three of the exemplar applications which are executed on the system.

2.1.1 Nanocores

Nanocores are a new class of programmable and configurable processors. In previous publications [8, 9] we explained the initial implementation of the AoC architecture on a Zedboard. Recently we have moved this onto an Avnet mini-ITX platform, the specification for which is shown in table 1. This further level of integration reduces makes use of a larger FPGA device fitting more Nanocores (32 Nanocores compared to 8 Nanocores on Zedboard) on the SoC and enabling more efficient on-chip

parallelisation of a range of analytical tasks. As with the initial prototype, this AoC instance supports 32-bit and 64-bit fixed point arithmetic. The rearchitected fabric in this new version supports multiple Nanocore groups (4 groups) and allows chains of 8 Nanocores to be accessed independently. The key aspect of the Nanocore is to optimise the underlying FPGA hardware to allow the creation of a light core which operates substantially faster than existing FPGA-based cores. Figure 1 shows the architecture of the Nanocore.

It is widely known that the software development lifecycle for accelerators and in particularly for FPGAs is longer than for implementation on either CPUs or GPUs. This fact mitigates against use of FPGAs in industries which are sensitive to cost pressures and/or development times, a notable example being the high frequency trading field in finance. Agility and adaptability are important contributors to both the financial and reputational risk exposure of financial firms. The fact that Nanocores are significantly easily and quickly programmed than using VHDL or Verilog directly serves to remove the disadvantage associated with using FPGAs. Figure 2 indicates that the AoC can be operated in a similar way to GPU and other accelerators. The compiler on the right hand side of the figure is implemented by defining the nanocore instruction set as a new backend to LLVM.

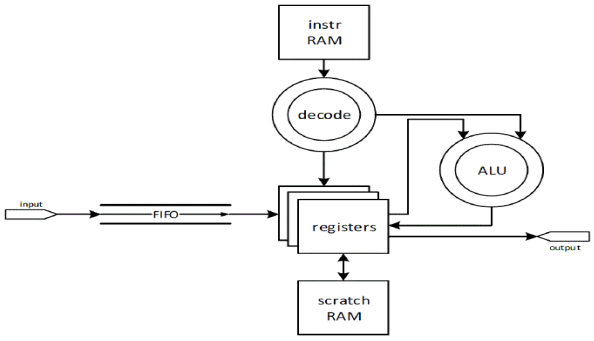


Figure 1: Block diagram of the Nanocore

Attribute	Value
Zynq device	**Z-7100**
Interconnect	1 Gb Ethernet
RAM	1GB PL, 1 GB shared DDR3
Processor Core	**Dual-core ARM Cortex A9**
Max frequency	800 MHz
Programmable Logic	**Kintex-7 FPGA**
LUTs	227,400
Flip-flops	554,800
BRAM	755
DSP48s	2,020

Table 1: Parameters defining the Avnet Mini-ITX board on which the latest Nanocore prototype is implemented

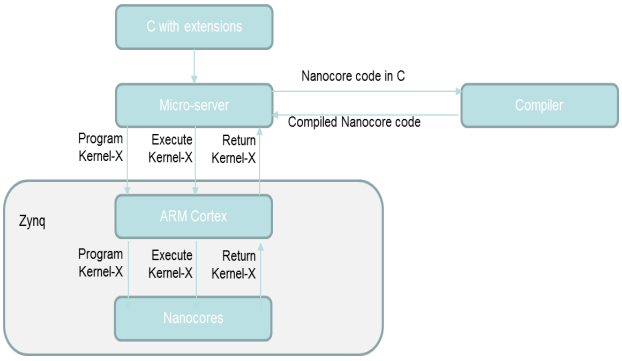


Figure 2: Illustration of the reprogramability of the nanocore.

2.1.2 NanoWire

NanoWire has two layers accounting on the one hand for the process of transporting packets between the host and accelerators and on the other the operation of issuing request to tasks running on the accelerator. We call these the Host-Accelerator Transport (HAT) and Task Issue Protocol (TIP) respectively. Together they create a communications substrate which:

1. has a simple and convenient API to virtualise and manage accelerators,
2. creates reliable, high-throughput and low latency transfers,
3. uses minimal CPU cycles on the host.

The layers of the NanoWire stack are shown in figure 3.

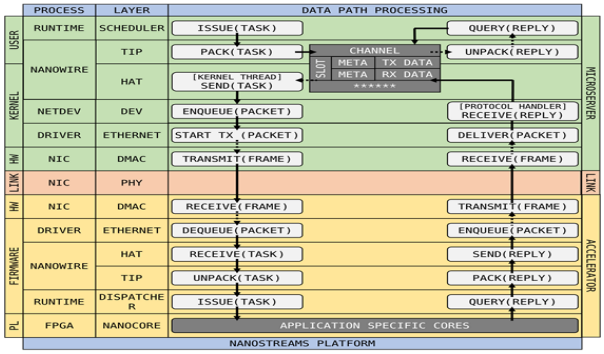


Figure 3: Layers of the NanoWire protocol

The HAT is the network tier of the overall microserver (called Nanostreams) system architecture and offers a common abstraction of the network level services and I/O primitives to both the host and accelerator nodes. It corresponds to encapsulating each NanoWire packet directly within one Ethernet frame, a process which requires the use of a custom EtherType field within the standard Ethernet header. HAT allows multiple hosts to share the same accelerator, it supports variable size packets (up to the Ethernet MTU size), and supports reliable transmission. Packets can be switched via Ethernet switches, however, HAT provides the ability to further customise the Ethernet header for efficiency, when switching is not required.

Finally, HAT provides lightweight connection-less channels as the lowest-level communication. A channel consists of a point-to-point unidirectional queue of packet slots used for communication between a source host node and a destination accelerator node. Resources per channel (e.g. element size) are chosen at creation time. Channels aim at providing a low overhead and low-latency communication path, while allowing the system to tune resources and resource placement for each channel.

NanoWire avoids the use of sockets and therefore eliminates the use of the kernel IP stack. Instead NanoWire offers TIP as a task queue layer that issues task requests from the hosts and receives task results from the accelerators. TIP therefore provides the network tier of NanoStreams and offers a common abstraction of the network level services and I/O primitives to both the host and accelerator nodes. TIP code runs on both sides of the interconnect and although the host side makes use of the host OS services, on the accelerator side TIP will run on more diverse platforms. In our prototype the accelerator side of HAT is implemented as custom firmware directly on top of the ARM processors of the FPGA card.

2.2 Case Studies and Results

We have applied the proposed architecture and software stack in various case studies. In the next paragraphs we discuss two commercial use cases covering financial markets and respiratory physiology in intensive care units.

2.2.1 Option pricing in the financial markets

Figure 4 shows the architecture of the proposed microserver (i.e. Nanostreams) system for pricing European stock options. Stock exchanges broadcast real-time price updates as a market data feed, with each exchange generally having some unique format. The linehandler software written by each client consumes the data feed converting it to a private format. In general the industry is dominated by UDP Multicast (UDP-MC/IP) and this is what we have used to relay the feed to the microservers in our lab set up.

As we have explained above, the AoC Accelerators are connected to the Ethernet in our lab and NanoWire allows for low latency communications between the microservers and the AoC. The number of AoCs in the systems (as well as Nanocores in each AoC) in the system is scalable and each can be dynamically shared as needed between the microservers.

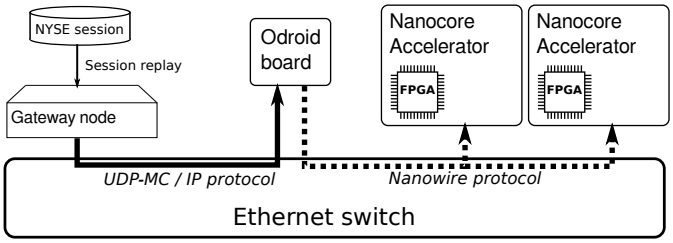


Figure 4: Architecture of the NanoStreams system computing financial option prices.

By using the first prototype of the AoC engine we have evaluated the performance of the option pricing case study and we compared it to an implementation on a single socket of a Intel Sandybridge server. The results are shown in table 2. Here, results are presented from pre power supply measurements which defines the total power budget.

Platforms	Power(W)	T_{opt} (s)	J_{opt} (J)
ARM + NanoCore (1st prototype)	5.1	0.201	1.025
ARM + NAnoCore (2nd prototype)	13.9	0.051	0.71
Intel	108.75	.016	1.74

Table 2: Performance parameters for the financial option pricing use case

The table compares the metrics of Joules per option J_{opt} and time per option in seconds T_{opt}. The NanoStreams system consumes only about 40% of the power compared to Intel platform.

Using the latest implementation on the Avnet Mini-ITX platform (2nd prototype) enhanced performance is obtained.

Performance increased fourfold, while the pre-PSU power consumption metrics increased by a factor three relative to the first prototype AoC.

2.2.2 *Intensive care medicine*

An intensive card unit (ICU) is a data rich, network enabled environment in which multiple physiological parameters for each patient are recorded into databases and monitored. In our work we are employing real-time analysis

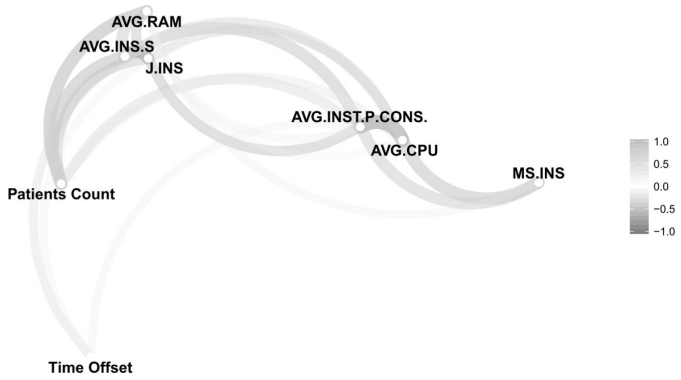


Figure 5: Graphical presentation of the Pearson correlation coefficients for database performance.

of streams of respiratory data from mechanical ventilators. Trend analysis and thresholding are applied to tidal volumes and airway pressure readings over varying time scales and alerts are issued via SMS to clinicians when pre-defined trends are observed. Determining such trends is a matter for clincial research, so that our tool is currently providing a mechanism to find the optimal intervention path.

The performance of the database at the heart of the application has been analysed for ingress of patient data. We measured similar metrics to those used in the financial case above. Figure 5 presents the Pearson correlation coefficients in an innovative graphical way. Each metric is a node in a graph and the proximity of the metrics to each other represents the overall magnitude of their correlations. Clusters are immediately evident in the figure therefore. For example, the cluster at the top left of the figures shows that Anerage RAM per insert (AVG.RAM) correlates strongly with Average Inserts per second (AVG.INS.S) and Joules per Insert (J.INS) Each path in figure 5 represents a correlation between the two metrics that it joins. A lighter path represents a positive correlation, and a darker path represents a negative correlation. The width and transparency of the line represent the strength of the correlation (wider and less transparent means stronger correlation).

Turning to data retrieval and computation of trends and reporting, table 3 shows the breakdown in terms of percentage CPU time for each operation. This shows that the NanoCore is ideally suited for off-loading the metric computation. This mathematical operation, which is similar to a moving average filter utilizes the following characteristics on the Nanocore.

Functionality	Percentage CPU time
Data retrieval	19
Compute metrics	68
Report formatting	13

Table 3: Partition of CPU effort for thresholding of ICU parameters

Nanocores are light weight cores capable of processing data being streamed in order. Stream of multiple patients' vital parameters is scattered in a way that each core processes one patient. That offers parallelism and easy tracking of data against patients' IDs. Each core keeps past values in memory and computes the new running average after receiving the current value. The new average is compared against threshold and transmitted to microserver. The processing time increases with a larger time scale but for the tested cases of up to 5-15 mins, each core can process about 4-4.5 Mega readings per second. The number scales almost linearly with addition of multiple cores.

3. ACCELERATORS BASED ON RECONFIGURABLE DATAFLOW ENGINES

In this section, we discuss a server architecture based on dataflow engines and present some initial comparative results against a classical multi-core server.

3.1 System Architecture

3.1.1 *Dataflow Engine*

Programmable accelerators based on dataflow engines is another promising solution that offers a revolutionary way of performing computation, completely different to computing with conventional CPUs [13, 3]. Instead of control flow based computation, the dataflow based computation is introduced where the powerful multi-tasking CPU cores are replaced with a vast number of single-tasking dataflow cores on FPGAs which are specifically customized and optimized fitting different algorithm features. By focusing on optimizing the movement of data in an application, number of order of magnitude benefits in performance, space and power consumption are gained through massive parallel processing.

A key challenge that prohibits the wide utilization of such dataflow engines in servers lies on the difficult integration of the models that are needed for programming such accelerators that may differ significantly from multi-core programming frameworks. The main difficulties lie on the semantic gap between accelerators and general-purpose processors (e.g., differences in the memory model), as well as on the lack of resource management support (e.g., for allocation and scheduling of tasks on accelerators). Currently the high-performance computing community has adopted hybrid programming models for enabling the use of accelerators, however such solutions are highly undesirable for virtualized accelerators in the cloud where the underlying resources must be transparent to the cloud users. To address the limitations of the hybrid programming models we propose alternatively to adopt a library of implementations of algorithms on accelerators and built a suitable infrastructure for being used from high-level programming frameworks.

3.1.2 *Spark*

Spark is an open sourced general framework for distributed computing that offers high performance for both batch and interactive processing and a fast and general engine for large-scale data processing [2]. It exposes APIs for Java, Python, and Scala and consists of Spark core and several related projects which offers seamlessly API support database queries, scalable fault-tolerance, machine learning and graph-parallel computation. By running Spark application locally or distributed across a cluster, it is interactively executed and commonly performed during the data-exploration phase with ad-hoc analysis capability.

As a programming platform with unified data management, it greatly speeds up the operation and maintenance as an 'all-in-one' solutions [1]. The interactive consoles offers interactive data analysis which seamlessly hooks up to a connected cluster. However, in order to adopt the advantages of Spark on DFE server, a new interfacing 'glue' is lacked with seamless adaptation to the feature of dataflow processing. Due to the different parallelism degree of DFE cluster compared to traditional CPU servers, a efficient data processing venue is required to accommodate the merits of both high level language and DFE.

3.1.3 *Infrastructure*

Those challenges to address in this work are to (i) hide the accelerator from the programmer by presenting it as a library function, embeddable in query processing, data processing or aggregation tasks; (ii) extend the runtime systems of high-level analytics languages to handle efficiently scheduling, communication, and synchronization with programmable accelerators; and (iii) improve the performance robustness of analytics written in high-level languages against performance artefacts of virtualization.

Figure 6 shows the entire system architecture of our approach. By using the Spark as high level programming language for big data and DFE as the accelerating units in the data center, the system is glued with Java Native Interface (JNI) and SLiC/MexelerOS (a run-time that allows accessing DFE through C based API). The DFE is generated through OpenSPL [4] programming model which produces a repository of a library for different run-time configurations as files with suffix '.max'. The Resilient Distributed Dataset (RDD) is transparent from Spark to DFE through our middle-ware and transmitted to DFE through infinite band between the host and accelerators.

3.2 Initial Results

In Figure 7, the Linear Regression from Maxeler Application Gallery is utilized as the benchmarks that we used for evaluating our infrastructure. By using the math library in Spark, the comparison and breakdown is carried out for the task execution between host CPU and the DFE accelerators.

In our experiments, we have characterized the overhead of off-loading tasks from Spark onto Maxeler DataFlow Engines (DFE) and evaluated the performance of linear correlation when executing on the DFE in comparison to its execution in Spark on 8 CPUs. Our results revealed a speedup of up to 374x for data set sizes ranging from 100 to 10M pairs of floats as shown in Figure 7a. Moreover, we have measured that moving data between the Java Virtual Machine and the DFE has about 3% overhead over execution of the DFE kernel as shown in Figure 7b, which makes this overhead practically negligible. In contrast, allocating and

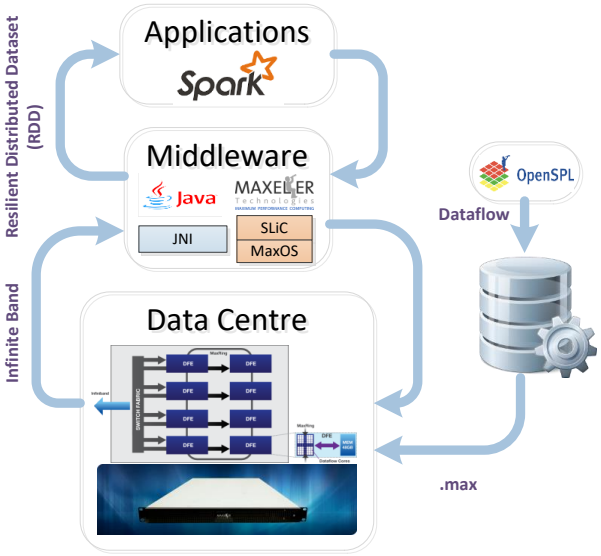
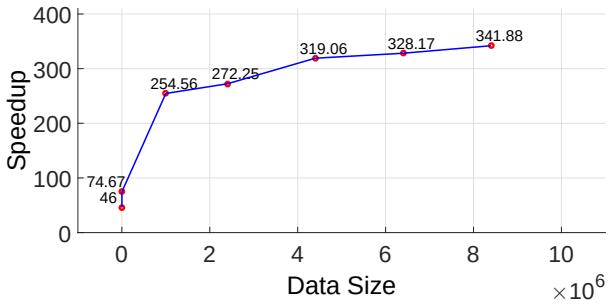
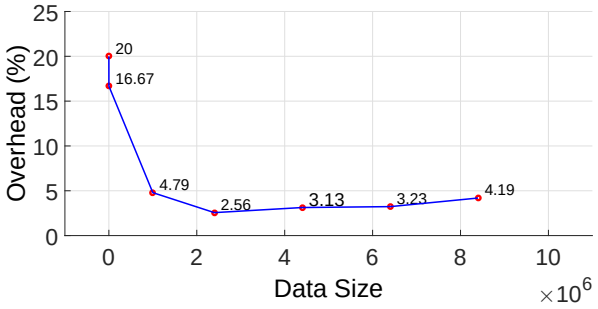


Figure 6: The Big Data Infrastructure on DFE

configuring the DFE takes about 3.2 seconds, and bears significant overhead.



(a) Performance Speedup



(b) Run-time Overhead

Figure 7: Linear Regression

As illustrated from the results in Figure 7, larger streaming data batch size gains significant performance improvement compared to the smaller ones. At the same time, the overhead is kept low once the streaming data batch size reaches certain level, 10^6 floats in this cases. Our initial results indicate that building big data applications on

DFE through our infrastructure is very promising. Accelerators based on DFEs can make a step change in the energy-efficiency and density of data centers. Seamless integration of accelerators in data center programming frameworks is however an open question, which will be addressed by future enhancements of our infrastructure.

4. CONCLUSIONS

This paper has presented two diferent approaches to the challenge of providing energy efficient servers targeting computing at the edge as well as in cloud environments. Nanocore and Nanowire bring together best practices from embedded systems design and high performance computing to achieve higher energy efficiency for analytical tasks on data streams than state of the art servers. A real-silicon prototype, based on the Xilinx Zynq platform and ARM-Linux has been shown to be competitive, for workloads drawn from different sectors, when compared to contemporary HPC servers, sustaining transactional throughput and improving system energy-efficiency and programmability.

The Dataflow approach, using Maxeler technology, provides an alternative solution which is also promising. While Nanocore and Nanowire focus on integer operations, the DFE enables floating point computation and provides speed-ups of over 300x compared to traditional servers.

Our work funded by the European Comission is contributing to the wider effort in Europe to create a sever ecosystem. It exploits intrinsic architectural variation to improve efficiency across a range of Edge computing and IoT workloads, areas which will dominate the application space for the foreseeable future.

5. ACKNOWLEDGMENTS

This research is supported in part by the European Community under the NanoStreams (FP7, contract 610528), VINEYARD (Horizon2020, grant 687628) and ASAP projects (FP7, grant 608224).

6. REFERENCES

[1] Apache Spark: An All-In-One Tool For The Data-Driven Enterprise. http://syntelli.com/blog/apache-spark-an-all-in-one-tool-for-the-data-driven-enterprise/. Accessed: 2016-11-10.

[2] Apache Spark^{TM} is a fast and general engine for large-scale data processing. http://spark.apache.org/. Accessed: 2016-11-10.

[3] Dataflow Computing. https://www.maxeler.com/technology/dataflow-computing/. Accessed: 2016-11-10.

[4] The Open Spatial Programming Language: OpenSPL. http://www.openspl.org/. Accessed: 2016-11-10.

[5] Cisco. *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2015âĂŞ2020 White Paper*. 2016.

[6] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger. Dark Silicon and the End of Multicore Scaling. *SIGARCH Comput. Archit. News*, 39(3):365–376, June 2011.

[7] G. Georgakoudis, C. Gillan, A. Hassan, U. Minhas, G. Tzenakis, I. Spence, H. Vandierendonck, R. Woods, D. Nikolopoulos, M. Shyamsundar, P. Barber, M. Russell, A. Bilas, S. Kaloutsakis, H. Giefers, P. Staar, C. Bekas, N. Horlock, R. Faloon, and C. Pattison. NanoStreams: Codesigned Microservers for Edge Analytics in Real Time. In *Proceedings: 2016 International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS XVI)*, 5 2016.

[8] G. Georgakoudis, C. J. Gillan, A. Sayed, I. Spence, R. Faloon, and D. S. Nikolopoulos. ISO-Quality of Service: Fairly Ranking Servers for Real-Time Data Analytics. In *Parallel Processing Letters*, 2015.

[9] G. Georgakoudis, C. J. Gillan, A. Sayed, I. Spence, R. Faloon, and D. S. Nikolopoulos. Methods and Metrics for Fair Server Assessment Under Real-Time Financial Workloads. In *Concurrency and Computation: Practice and Experience*, 2015.

[10] Z. Lin and P. Chow. ZCluster: A Zynq-based Hadoop Cluster. In *2013 International Conference on Field-Programmable Technology (FPT)*, pages 450–453, Dec 2013.

[11] P. Moorthy and N. Kapre. Zedwulf: Power-Performance Tradeoffs of a 32-Node Zynq SoC Cluster. In *2015 IEEE 23rd Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 68–75, May 2015.

[12] M. Muller. New mbed IoT Device Platform. *ARM TechCon*, 2014.

[13] O. Pell and V. Averbukh. Maximum Performance Computing with Dataflow Engines. *Computing in Science Engineering*, 14(4):98–103, July 2012.