# Big Data Analytics on Heterogeneous Accelerator Architectures

Katayoun Neshatpour, Avesta Sasan, Houman Homayoun
Electrical and Computer Engineering Department, George Mason University
kneshatp@gmu.edu, asasan@gmu.edu, hhomayou@gmu.edu

## ABSTRACT

In this paper, we present the implementation of big data analytics applications in a heterogeneous CPU+FPGA accelerator architecture. We develop the MapReduce implementation of K-means, K nearest neighbor, support vector machine and Naive Bayes in a Hadoop Streaming environment that allows developing mapper/reducer functions in a non-Java based language suited for interfacing with FPGA-based hardware accelerating environment. We present a full implementation of the HW+SW mappers on the Zynq FPGA platform. A promising speedup as well as energy-efficiency gains of upto 4.5X and 22X is achieved, respectively, in an end-to-end Hadoop implementation.

## 1. INTRODUCTION

Emerging big data analytics applications require a significant amount of server computational power. However, these applications share many inherent characteristics that are fundamentally different from traditional desktop, parallel, and scale-out applications [6,13,14]. They heavily rely on specific deep machine learning and data mining algorithms. The characteristics of big data applications necessitate a change in the direction of server-class microarchitecture, to improve their computational efficiency. However, while demand for data center computational resources continues to grow with the growth in the size of data, the semiconductor industry has reached scaling limits and is no longer able to reduce power consumption in new chips. Thus, current server designs based on commodity homogeneous processors, are no longer efficient in terms of performance/watt to process big data applications [6,13,14].

To address the energy efficiency problem, heterogeneous accelerator architectures have emerged to allow each application to run on a core that best matches its resource needs, than a one size-fits-all processing node. A heterogeneous chip architecture integrates cores with various micro-architectures with on-chip GPU or FPGA accelerators, so that the application can find a better match among various components to improve energy efficiency. Most recent work on hardware acceleration of big data analytics have focused on the implementation of an entire particular machine learning application, or offloading an entire phase of MapReduce to the FPGA hardware [10,11,12]. While these approaches provide performance benefit, their implementations require excessive hardware resources and extensive design effort. As an alternative, hardware+software (HW+SW) co-design trades some speedup at the benefit of less hardware and more design automation using high-level synthesis (HLS) tools. To understand the potential performance gain of HW+SW co-design to accelerate analytics applications in MapReduce, in our most recent work [3,9,10,12] including this paper we present a full end-to-end implementation of big data analytics applications in a heterogeneous CPU+FPGA architecture, taking into account various communication and computation overhead in the system. The rest of the paper presents the result of our latest effort on big data acceleration.


*CODES/ISSS '16*, October 01-07, 2016, Pittsburgh, PA, USA
 ISBN 978-1-4503-4483-8/16/10…$15.00
DOI: http://dx.doi.org/10.1145/2968456.2976765

## 2. SYSTEM ARCHITECTURE

Fig. 1 shows the system architecture of a single-node, multi-core heterogeneous accelerator platform studied in this paper. While in a general purpose CPU, mapper/reducer slots are mapped to a single core, in a heterogeneous accelerator architecture depicted in Fig. 1, each mapper/reducer slot is mapped to a core that is extended with an FPGA accelerator. In a homogenous architecture, the cores are connected together through their shared-memory distributed interconnect. On the other hand, in a heterogeneous accelerator architecture the interconnection interface between the FPGA and CPU, is the main overhead in this architecture. However, given the tight integration between FPGA and CPU, the acceleration of the mapper/reducer slots, with the FPGA, introduces no significant off-chip data transfer overhead. For implementation purposes, we compare two types of general-purpose core architectures; low-power little Intel Atom C2758 core, and high-performance big Intel Xeon E5 core. Note that the general-purpose core is required for job scheduling as well as HDFS management. Each FPGA shown in Fig. 1 is a low cost Xilinx Artix-7. The integration between the core and the FPGA is compatible with the advanced micro-controller bus architecture (AMBA) and utilizes the Advanced eXtensible Interface (AXI)-interconnect. **Fig. 2** shows the system architecture of a multi-node cluster. The architecture consists of a homogeneous CPU as the NameNode, which is connected to several DataNodes with heterogeneous accelerators. The NameNode is responsible for job scheduling and workload distribution between all DataNodes. The number of map/reduce slots on each DataNode is based on its number of cores.

## 3. METHODOLOGY

This paper characterizes the FPGA acceleration potential for big data analytics applications. Accordingly, we develop MapReduce implementation of data mining and machine applications as examples of big data analytics applications. Subsequently, we accelerate the map/reduce function through HW+SW co-design.

It should be noted that Hadoop applications are mostly implemented in Java; however, the FPGA+CPU platforms allow hardware acceleration of C-based applications. However, Hadoop Streaming allows any executable or script to be used as the mapper and/or reducer function in the Hadoop. Our methodology for HW acceleration includes two levels of profiling.

1) *Profiling of the entire application in MapReduce:* MapReduce platform is comprised of several execution phases; we use the timing information to calculate the execution time of each phase of the entire MapReduce before the acceleration.

2) *Profiling of the map/reduce functions*: Since only the map/reduce functions are accelerated, we profile these functions in order to find out the execution time of different sub-functions within the map/reduce and select the sub-functions with the highest execution time (hotspot) to offload to the FPGA for acceleration. To estimate the execution time after the acceleration

accurately, we fully implement the map/reduce functions on the Zedboards, featuring XC7Z020 Zynq, which integrates ARM Cortex-A9 with an Artix-7 FPGA. The connections between the core and the FPGA is established through the AXI interconnect. Subsequently, based on the information about the execution time of each phase, and the speedup of the map/reduce functions, we perform a comprehensive analysis to find out how acceleration of the map/reduce functions contribute to the acceleration of the entire application on Hadoop MapReduce.

# 4. PARAMETER TUNING

The execution time, power and energy-efficiency of MapReduce, both before and after acceleration, is affected by various parameters at the system, architecture and application-level. We describe the major parameters briefly in the sequel.

1) *Size of HDFS block size:* Hadoop stores data in highly fault-tolerant distributed file system (HDFS). The size of the HDFS block size is optimized to account for the requirements of the Hadoop architecture. On one hand, reading and writing data at high volume (granularity) is efficient, as it improves performance by decreasing the amount of metadata that is tracked in the file system, and allows memory access latency to be amortized over a large volume of data. On the other hand, lower granularity allows better exploitation of parallelism and resources.

2) *Number of mapper/reducer slots:* An important criteria while making architectural decisions is the restrictions on the number of mapper/reducer slots. The MapReduce programmer decides the number of map and reduce tasks. First, the input data is divided into data splits. Then, based on the data split size, the number of map tasks is derived. However, the number of tasks that are executed in parallel depends on the hardware resources, namely mapper/reducer slots. Experiments show that for performance optimization the number of slots is best to be tuned in a range of (0.95-1.75)× the number of available cores [2].

3) *Core architecture:* The general-purpose core used for job scheduling and HDFS management greatly affects the acceleration benefits. While the acceleration could improve the speed more significantly in one architecture (little core), it could enhance the energy-efficiency more in the other (big core).

It should be noted that the optimal configuration for performance, energy or a trade-off of both is a function of all above parameters, thus, we carry-out real experiments for various corners to find the optimal solution for a given optimization goal. In the results section we show the results with the lowest post acceleration time.

# 5. RESULTS

Table 1 shows the results of the HW+SW acceleration of the mapper functions using the FPGA and interconnect resources in the Zynq platform. The non-accelerated parts of the mapper functions remain on the Atom and Xeon cores. Based on the results in the table the mapper function shown to gain more performance improvement on Atom. Table 2 and Table 3 show the execution time and power of the studied applications. Results show that an acceleration of upto 4.5X can be achieved. Moreover the speedup values are higher on Atom than Xeon. This is due to the fact that in Atom a higher portion of the execution time is spent in the accelerated phases (map/reduce). However, both before and after the acceleration, Xeon outperforms Atom. The power results show that the power consumption mostly increases after the acceleration of Atom, which is expected, as Atom cores are designed to be low power. However the acceleration shows potential for power reduction on Xeon. Based on the results from Table 2 and Table 3, the hardware acceleration improves the speed more on Atom while reducing power on Xeon. Overall the Energy-Delay Product (EDP), which is an indication of energy-efficiency, improves significantly in both architectures. The energy-efficiency gain varies significantly across applications and architectures. While in Naïve Bayes a 22X EDP gain is achieved, in SVM the energy-efficiency gain show to be only 8%.

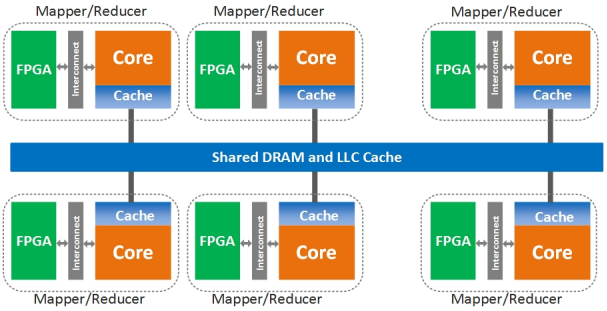


**Fig. 1. Single-node heterogeneous accelerator architecture.**

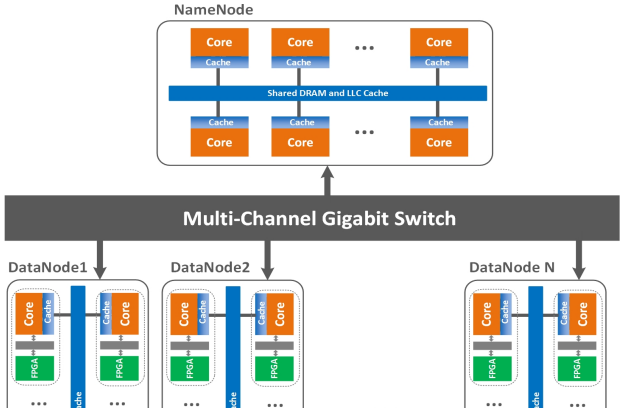


**Fig. 2. Multi-node heterogeneous accelerator architecture.**

## 5.1 Scalability in a Multi-node Architecture

In order to understand the scalability of our proposed acceleration method, we study HW+SW acceleration in a multi-node cluster and across a large range of input data size. We use a 12-node (1 NameNode and 11 DataNodes) cluster, each with dual Intel Xeon E5 allowing upto 176 mapper/reducer slots to run simultaneously. We execute the applications for various data sizes. It should be noted that as the input data size increases, the execution time does not increase linearly with the size of data. Thus, in Fig. 3 we show the execution time for each GB of data for various data sizes (for all applications except SVM as there is no significant performance benefit). The dashed line shows the time after the acceleration. The figure shows that as the size of data increases beyond 20GB, the execution time per 1GB of data converges to fixed values for each application both before and after acceleration. However with acceleration, this happens earlier, at a smaller data size. In fact acceleration increases the role of communication overhead in deciding the overall performance. Also increasing the data size increases the amount of intra-node communication and therefore makes the hardware acceleration less effective.

# 6. RELATED WORK

The performance and bottlenecks of Hadoop MapReduce, as a de facto standard for big data analysis, have been extensively studied in recent work [4,5,6,13]. To enhance the performance of MapReduce and based on the performance bottlenecks found, hardware accelerators are finding their ways in cloud scale system architectures. MARS [7], a MapReduce framework on an NVIDIA G80 GPU and Cluster GPU MapReduce [8] are examples of the GPU-based accelerator platforms. However, GPU

**Table 1. Acceleration and resource utilization of HW+SW acceleration of mapper functions**

| Resources | Available | Utilization (%) | | | |
|---|---|---|---|---|---|
| | | SVM | K-means | KNN | NB |
| FF | 35200 | 7.57 | 6.38 | 4.07 | 4.28 |
| LUT | 17600 | 12.03 | 11.98 | 7.13 | 8.66 |
| Memory LUT | 6000 | 1.23 | 2.03 | 1.14 | 1.23 |
| BRAM | 60 | 59.29 | 37.86 | 47.5 | 62.14 |
| DSP48 | 80 | 2.27 | 9.55 | 0 | 0 |
| BUFG | 32 | 3.13 | 3.13 | 3.13 | 3.13 |
| Mapper acceleration on Atom | | 1.028 | 4.2 | 2.51 | 8.99 |
| Mapper acceleration on Xeon | | 1.042 | 7.9 | 3.9 | 18.52 |

**Table 2. Execution time and speedup of the end-to-end MapReduce after acceleration of the map phase**

| Application | Core | time[s] | Accel time [s] | Speedup |
|---|---|---|---|---|
| Naïve Bayes | Atom | 1256.32 | 275.48 | 4.561 |
| | Xeon | 756.59 | 182.68 | 4.142 |
| K-means | Atom | 60.22 | 18.60 | 3.238 |
| | Xeon | 41.34 | 23.19 | 1.783 |
| KNN | Atom | 707.14 | 332.66 | 2.126 |
| | Xeon | 369.87 | 240.45 | 1.538 |
| SVM | Atom | 51.90 | 50.44 | 1.029 |
| | Xeon | 35.19 | 34.89 | 1.008 |

**Table 3. Power of the end-to-end MapReduce before and after the acceleration of the map phase**

| Application | Core | Initial power [w] | Accel power [w] | $\frac{initial\ power}{accel\ power}$ | EDP ratio |
|---|---|---|---|---|---|
| Naïve Bayes | Atom | 5.94 | 11.2 | 0.53 | 11.04 |
| | Xeon | 38.37 | 29.65 | 1.29 | 22.20 |
| K-means | Atom | 3.65 | 7.49 | 0.49 | 5.11 |
| | Xeon | 12.99 | 14.33 | 0.91 | 2.88 |
| KNN | Atom | 3.53 | 5.29 | 0.67 | 3.02 |
| | Xeon | 27.69 | 25.33 | 1.09 | 2.48 |
| SVM | Atom | 3.56 | 3.52 | 1.01 | 1.07 |
| | Xeon | 26.05 | 24.81 | 1.05 | 1.08 |

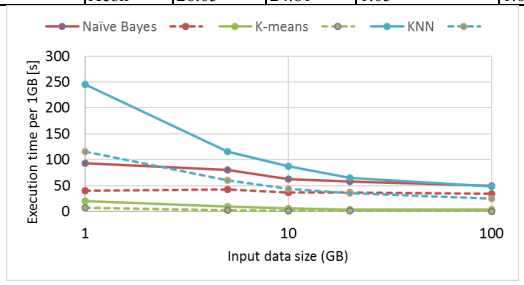


**Fig. 3. The execution time per 1GB of the MapReduce for various input sizes on a 12-node cluster.**

-based work mostly target performance optimizations at the cost of more power, therefore are not the most energy-efficient solution. Alternatively, FPGAs, by allowing hardware customizations to match application demands, have been used as a more energy-efficient solution. Examples of FPGA acceleration platforms for big data is Microsoft's composable fabric used to accelerate portions of large-scale software services, which consists of 6x8 2-D torus of Stratix V FPGAs embedded into a half-rack of 48-node machine [1]. Heterogeneous architecture research platform (HARP), is another example designed for server architectures that uses HW+SW acceleration by integrating Intel CPU with Altera FPGAs. These work mainly rely on high performance, high cost FPGAs with a fast dedicated interconnection network to improve performance. However, in our recent work [3,9,12] including this work we use a low cost, small FPGA fabric to improve performance of Hadoop MapReduce applications which makes it a cost-effective and scalable solution for big data analytics in cloud architectures. Given the importance of MapReduce for big data analytics, recent work [4,11] have used hardware accelerators to improve the speed of MapReduce on heterogamous platforms by offloading an entire MapReduce application on the accelerator. While these work have mainly focused on the implementation of an entire phase of MapReduce on the hardware and for particular applications, their implementations require excessive hardware resources and is not automated. As an alternative, in our work [3,9,10,12,14] the map phase in MapReduce is accelerated through HW+SW co-design, which trades some speedup at a benefit of less hardware and more automation. Moreover, we focus on the acceleration for non-Java based map and reduce functions, which makes it a favorite choice for hardware programming. C/C++ definition of such functions in Hadoop streaming allows a flexible partitioning of them between the FPGA and CPU core on platforms that allow their on-chip integration such as Xilinx Zynq and Altera Arria. This is the first step towards a full-automated solution [3,9,12].

# 7. CONCLUSION

In this paper, we presented a full end-to-end implementation of big data analytics applications in a heterogeneous CPU+FPGA architecture. We developed the MapReduce parallel implementation of K-means, KNN, SVM and naive Bayes in a Hadoop Streaming environment. We accelerated the mapper functions through HW+SW co-design with full implementations on the Zynq FPGA platform. Overall, the experimental results show that a low cost embedded FPGA platform, programmed using our proposed HW+SW co-design method, yields performance and energy efficiency gains for MapReduce computing in cloud-based architectures, significantly reducing the reliance on large number of big high-performance cores for high performance designs.

# 8. REFERENCES


[1] A. Putman, et al, "A configurable fabric for accelerating large-scale datacenter services," in ISCA 2014, pp. 13-24.

[2] K. Tannir, "Optimizing Hadoop for MapReduce Publishing," Packt Publishing Ltd, 2014.

[3] K. Neshatpour, et al. "Energy-efficient acceleration of big data analytics applications using fpgas." 2015 Big Data.

[4] T. Honio et. al., "Hardware acceleration of Hadoop MapReduce," in *2014 IEEE Int. Conf. Big Data*, 2013.

[5] D. Jiang, et. al., "The performance of MapRedce: An in-depth study, " *Proc. VLDB*., vol. 3, no. 1-2.

[6] M. Malik, et. al, "System and Architecture Level Characterization of Big Data Applications on Big and Little Core Server Architectures," 2015 IEEE Big Data.

[7] B. He, et. al, "Mars: a MapReduce framework on graphics processors", *PACT 2008*.

[8] J. A. Stuart and J.D. Owens, "Multi-GPU MapReduce on GPU clusters," 2011 IEEE IPDPS, 2011, pp. 1068-1079.

[9] K. Neshatpour, et al. "Accelerating big data analytics using fpgas." IEEE FCCM 2015.

[10] K. Neshatpour, et al. "Big Biomedical Image Processing Hardware Acceleration: a Case Study for K-means and Image Filtering", in ISCAS 2016.

[11] Y. Shan., et al. "FPMR: MapReduce framework on FPGA," in *2010 ACM/SIGDA Conf FPGA*, pp. 93-102

[12] K. Neshatpour, et al. "Accelerating machine learning kernel in hadoop using fpgas." IEEE/ACM CCGRID 2015.

[13] M. Malik , H. Homayoun. "Big data on low power cores: Are low power embedded processors a good fit for the big data workloads?" IEEE ICCD 2015.

[14] H. Homayoun. "Heterogeneous chip multiprocessor architectures for big data applications", ACM International Conference on Computing Frontiers, 2016.