Architectural Considerations for FPGA Acceleration of Machine Learning Applications in MapReduce

Conference Paper · July 2018

CITATIONS
3

READS
161

6 authors, including:

Katayoun Neshatpour
George Mason University
17 PUBLICATIONS **84** CITATIONS
SEE PROFILE

Hosein Mohammadi Makrani
George Mason University
13 PUBLICATIONS **50** CITATIONS
SEE PROFILE

Avesta Sasan
George Mason University
49 PUBLICATIONS **332** CITATIONS
SEE PROFILE

Hassan Ghasemzadeh
Washington State University
128 PUBLICATIONS **1,412** CITATIONS
SEE PROFILE

Architectural Considerations for FPGA Acceleration of Machine Learning Applications in MapReduce

Katayoun Neshatpour[1],Hosein Mohammadi Mokrani[1], Avesta Sasan[1], Hassan Ghasemzadeh[2]

Setareh Rafatirad[1], Houman Homayoun[1]

[1]{kneshatp, hmohamm8, asasan, srafatir, hhomayou}@gmu.edu, [2]hassan@eecs.wsu.edu
[1]Department of Electrical and Computer Engineering, George Mason University, Fairfax, VA
[2]School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA

1 INTRODUCTION

While demand for data center computational resources continues to grow as the size of data grows, the semiconductor industry has reached scaling limits and is no longer able to reduce power consumption in new chips. Unfortunately, the promise of analytics running on large systems (e.g., data-centers) over huge data, and scaling those systems into the future, coincides with an era when the end of Dennard scaling brings into serious question our ability to provide scalable computational power without prohibitive power and energy costs.

One of the most promising solution to address the computational efficiency crisis is heterogeneity and specialization.

In particular, heterogeneous hardware accelerators have received renewed interest in recent years in cloud scale architectures [1] [2], mainly due to slowing of MooreâĂŹs law scaling, along with compute and latency requirements of cloud workloads that is increasing beyond CPU-only capabilities. Several GPU and FPGA-based heterogeneous accelerators, have been proposed for big data analytics and cloud computing and in particular for MapReduce programming model, a de facto standard for big data analytics. Table 1 summarizes a number of these efforts.

As shown in Table 1, the recent efforts mainly focus on accelerating a particular MapReduce application and deploying it on a specific architecture that fits well the performance and power requirements of the application. Given the diversity of architectures presented in Table 1, the important research question is which architecture is better suited to meet the performance, power and energy-efficiency requirements of a wide and diverse range of machine learning applications implemented in MapReduce. To the best of our knowledge, there has been no prior effort on performing design space exploration to find the choice of hardware accelerator architecture that can be deployed in cloud for MapRecue analytics applications.

Thus in this paper we explore a large architecture space which spans FPGA technology (high-end vs low-end), core technology (big vs little core server), and interconnection technology (on-chip vs off-chip) to understand the benefits of hardware acceleration and power and performance trade-offs offered in each of these design points to gain architectural insights. Figure 1 shows the overview of the design space exploration of MapReduce in this paper.

Our experiments answer the questions of what is the role of processor after acceleration in a heterogeneous architecture; whether a high-end server equipped with big or a low-end server equipped with little core is most suited to run big data applications post

Table 1: Latest effort on hardware accelerator for cloud computing

	Interface	Processor	FPGA
[3]	Ethernet	Duo Core Intel	Xilinx Virtex-II Pro
[4]	PCIe	Intel Pentium 4	Altera Stratix II
[5]	AXI-Stream	Atom	Artrix-7
[6]	PCIe	ARM	Artrix-7 (Zynq)
[7]	QPI	Intel Xeon	Altera Stratix V
[8]	PCIe	Intel Core i5	Xilinx Kintex-7
[9]	AXI4	-	Xilinx Virtex-7

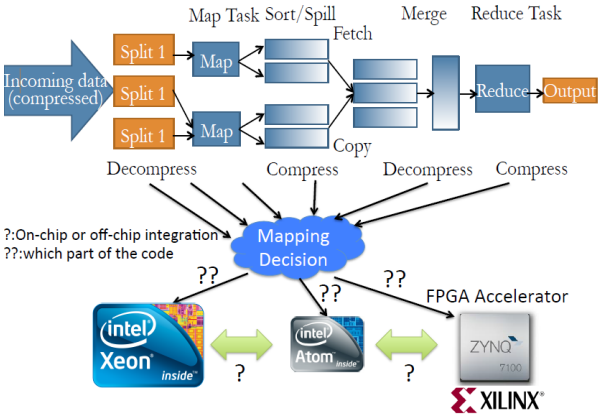

Figure 1: Hadoop MapReduce Framework and the mapping decisions

hardware acceleration? More specifically we answer the important architecture research question of whether on-chip heterogeneity is needed for a diverse range of ML-based MapReduce applications and how the core architecture needs to be integrated with the accelerator; whether off-chip (such as with PCIe) or on-chip (such as with AXI), and whether a high-end FPGA (such as Virtex-6) is needed or a low-end FPGA (such as Artix-7) is sufficient for energy-efficiency. The former question is particularly important as due to yield and other manufacturing challenges, it is not yet cost-efficient to integrate large reconfigurable fabric with the CPU architecture on the same die. Even recent integration technologies such as 2.5D interposer [10, 11], which enabled higher yields for integrating

reconfigurable fabrics, are not scalable cost-efficient solutions for server class architectures.

Based on the experimental data of accelerating several major Hadoop based machine learning algorithms on CPU+FPGA accelerated prototype platforms, this paper makes the following important observations:

- HW acceleration yields significantly higher speedup on Atom server, and more significant power reduction on Xeon server, reducing both the performance and power gap between little Atom and big Xeon after the acceleration.
- The type of CPU is the most important factor in determining the execution time and the power in CPU+FPGA architecture. The integration technology is the second most important design parameter. The type of FPGA is important only when the integration technology allows fast transfer of data between the CPU and the FPGA.
- The high bandwidth of recent off-chip interconnection protocols such as PCIe Gen-3, allows the data transfer time to be masked by the computation time in the highly parallel map functions, therefore eliminates the need for a high cost on-chip integration of FPGA and CPU in MapReduce.
- Considering the server capital cost, Atom with low-end FPGA is the most-cost-efficient solution. Combining Atom with high-end FPGA is not cost-efficient, as the resulting acceleration gain is small compared to additional cost of FPGA.

The rest of the paper is organized as follows. Section 2 introduces the experimental setup. Section 3 describes the modeling of MapRedue hardware acceleration. Section 4 presents the results, followed by a discussion on architectural decisions and how they effect the overall performance in section 5. Section 6 covers the related work. Finally in section 7 we conclude the paper.

2 EXPERIMENTAL SETUP

MapReduce is the programming model developed by Google to handle large-scale data analysis [12]. The MapReduce framework is a well-utilized implementation for processing and generating large data sets in which, the programs are parallelized and executed on a large cluster of commodity servers. MapReduce consists of map and reduce functions where, the map functions parcel out work to different nodes in the distributed cluster, and the reduce functions collate the work and resolve the results. Fig. 1 shows Hadoop MapReduce Framework.

2.1 System architecture

Fig. 2 shows the system architecture of a single node of the studied platform, where each mapper/reducer slot is mapped to a core that is integrated with the FPGA. For implementation purposes, we studied two very distinct server microarchitectures; a high performance big Xeon core and another a low power embedded-like little Atom core. These two types of servers represent two schools of thought on server architecture design: using big core like Xeon, which is a conventional approach to designing a high-performance server, and the Atom, which is a new trajectory in server design that advocates the use of a low-power core to address the dark silicon challenge facing servers [13].

For the choice of FPGA technology we studied two different generations of FPGAs, an Artix-7 representing a low-end FPGA and a Virtex-6 representing a high-end FPGA.

It should be noted that on-chip integration of the accelerator with the core allows faster transfer of data between the core and the accelerator. On the other hand, off-chip integration results in slower transfer rate between the two, but allows more flexibility, since the type of the FPGA and the core are selected without being confined to using the limited available on-chip FPGA+CPU platforms. To study the impact of integration technology, we model existing off-chip and on-chip interconnect protocols, PCI-Express Gen3 and AXI-interconnect, respectively.

2.2 Methodology

While native C/C++ injection into Hadoop is on the way, various utilities have been used to allow Hadoop to run applications, the map and reduce functions of which, are developed in languages other than Java. We employ Hadoop streaming [14], a utility that along with the standard Hadoop distribution, allows running MapReduce jobs with any executable or script as the mapper and/or the reducer. We provide C-based map and reduce functions, which allows using HLS tools to provide a Hardware Descriptive Language (HDL) representation for the desired accelerators. Subsequently, we carry out two levels of profiling.

- *Profiling of the Application on MapReduce*: In order to calculate the potential speedup on the Hadoop platform after the acceleration of map/reduce functions, we carry out a detailed analysis and profiling for various phases.(i.e. map, reduce, shuffle, etc.) with various profiling tools including Intel V-tune [15] to find out the contribution of each phase to the total execution time and identify the performance hotspot for acceleration. As we will present later in this paper, we identified map function to be the dominant performance hotspot across all studied applications.
- *Profiling of the map function*: To accelerate the map functions through HW+SW co-design, we profile them to find out the execution time of different sub-functions and select the sub-functions to be offloaded to the FPGA. The HDL description of the selected sub-functions is created utilizing HLS tools [16]. Each map function is carried out on data splits that are mostly the size of one HDFS block for data locality purposes, which varies from 64MB, 128MB to even higher [17]. For each application, we execute the map function on the data splits and profile it on the two big and little server architectures.

3 MODELING THE HARDWARE-SOFTWARE CO-DESIGN

A comparison of various models of computation for hardsware+software co-design has been presented by [18]. The classical FSM representation or various extension of it are the most well-known models for describing control systems, which consists of a set of states, inputs, outputs and a function which defines the outputs in terms of inputs and states, and a next-state function. Since they do not allow the concurrency of states and due to the exponential growth of the number of their states as the system complexity rises, they are not the optimal solution for modeling HW+SW co-design. Dataflow graphs have been quite popular in modeling data-dominated systems [19]. In such modeling, computationally intensive systems

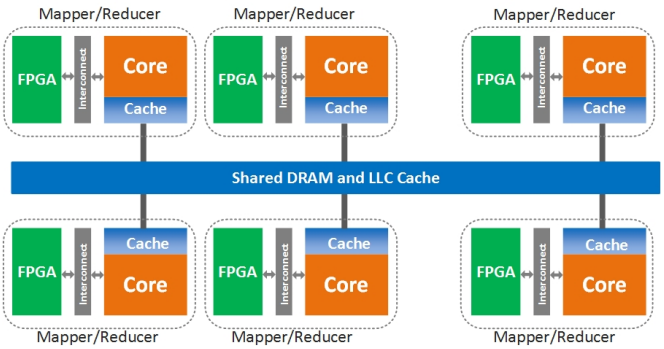

Figure 2: System architecture of HW accelerator for a single node.

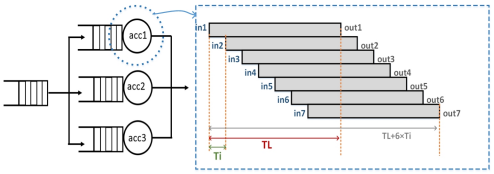

Figure 3: The queuing network with three identical HW accelerators and the pipeline timing of the accelerator.

and/or considerable transportation of data is conveniently represented by a directed graph where the nodes describe computations and the arcs represent the order in which the computations are performed.

In case of acceleration of the map phase in a MapReduce platform, what needs to be taken into account is the highly parallel nature of the map functions, which allows higher acceleration by concurrent processing of multiple computations that have no data dependencies. Most efforts for modeling of the hardware+software co-design have found data dependencies to be an important barrier in the extent to which a function is accelerated, however this is not the case for MapReduce, In the mapper part of most machine-learning applications a small function, i,e., an inner product or a Euclidean distance calculation is the most time-consuming part of the code, where multiple instances of a small function can be executed in parallel with no data dependencies. In such cases, a simple queuing network can be deployed to model a map function, with only one accelerated sub-function.

Queuing system models are useful for analyzing systems where inputs arrive sporadically or the processing time for a request may vary [19]. In a queuing model, customers (in this case, the data to be processed by the accelerator) arrive at the queue at some rate; the customer at the head of the queue is immediately taken by the processing node (the accelerator hardware), but the amount of time spent by the customer in processing must be specified (service time). Typically, both the customer arrival rate and processing time are modeled as Poisson random variables. In our case, however, since we are using one or multiple copies of the same accelerator, the service time for all data is fixed and is determined by the maximum frequency of the accelerator on the FPGA and the number of clock cycles it takes to finish the processing of each batch of data. Moreover, we assume that the data arrives at the accelerator at a fixed rate, determined by the bandwidth of the interconnect (transmission link).

Considering efficient buffer sizes, when the first batch of input data required for one call of the accelerated function arrives at the FPGA, the FPGA starts processing of data. Extensive pipelining allows the FPGA to start the processing of the next batch of data before the processing of the previous batch is finished. Fig. 3 shows the processing time of the batches of data, where T_L is the latency (the time it takes to produce the first output in accelerator) and T_i is the interval, which is the time before the accelerator can initiate a new set of reads and process the next set of input data.

Fig. 3 assumes a pipelined architecture in which, the interface provides the input data at the same rate as it is being consumed by the FPGA. Thus, the processing of the data is optimal in FPGA. Based on Fig. 3 The total processing time is $func\_call \times T_i + T_L$, and $T_i < T_L$. However, in a non-pipelined architecture the processing time is $func\_call \times T_L$, which leaves less room for exploiting parallelism.

Moreover, since all function calls are data independent, multiple instances of the same function can be implemented on FPGA (considering FPGA resources). However, the data movement overhead of the inputs becomes a potential bottleneck. In other words, the data arrival rate, which is decided by the interconnect bandwidth, should optimally be less or equal to the service time, otherwise the processing time is limited by the transfer time since the computation units have to wait for the data to arrive at the FPGA. Assuming a pipelined architecture consisting of N accelerators, in which the transmission link is not the bottleneck, the total processing time is reduced to $*\frac{func\_call}{N} \times T_i + T_L$.

3.1 On-chip vs off-chip interconnect

Due to yield and other manufacturing challenges, it is not cost-efficient to integrate large reconfigurable fabric with the CPU architecture on the same die. While recent integration technologies such as 2.5D interposer enabled higher yield for integrating reconfigurable fabrics, still primarily due to cost, they are not scalable solutions, in particular for server class architectures. Another example is the Zedboard which integrate two 667 MHz ARM CortexA9 with an Artix-7 FPGA with 85 KB logic cells and 560 KB block RAM. While interconnection between the two is established through the AXI-interconnect, which achieves maximum bandwidths of 600-9600 MBps, depending on the physical DDR3 data width (8-128 bits) and Memory clock (300-400MHz) [20], the FPGA is much smaller than other stand-alone Xilinx FPGAs including Kintex-7 and Virtex-7 with upto 215K and 475K logic cells and 4MB and 8MB block RAM, respectively. On the other hand off-chip integration allows connection to wide range of processors including Intel Atom and Intel Xeon. However, their interconnection would be through standard off-chip bus protocols such as PCI-type interconnects, which are typically slower than on-chip interconnect.

The specified per lane maximum transfer rate of generation 1 (Gen1), Gen2 and Gen3 PCI EXpress is 312.5 MBps, 625MBps and 1GBps, respectively [21]. It is important to note that, these rates specify the raw bit transfer rate per lane in a single direction. Effective data transfer rate is lower due to overhead and other system trade-offs including transaction layer packets (TLP) overhead and the traffic overhead. While Gen1 and Gen2 protocols use an 8B/10B

Table 2: Architectural parameters

Processor	Intel Atom C2758	Intel Xeon E5-2420
Operating Frequency	2.4 GHz	1.9 GHz
Micro-architecture	Silvermont	Sandy Bridge
L1i Cache	32 KB	32 KB
L1d Cache	24 KB	32 KB
L2 Cache	4×1024 KB	256 KB
L3 Cache	-	15MB
PageTable	16972 KB	4260 KB
System Memory	8 GB	32 GB
TDP	20 W	95 W
network interface		
model	ST1000SPEXD4	
speed	1000Mbps	

encoding scheme on the transmission lane to maintain DC balance, Gen3 uses a 128/130B encoding scheme. The encoding schemes guarantee a transition-rich data stream at the cost of 20% throughput loss in Gen1 and Gen2, and 2% in Gen3. Thus the effective bandwidth of Gen1, Gen2 and Gen3 transmission line drops to 250, 500 and 1000 MBps per direction lane, respectively [21]. It should be noted that, the data link layer and physical layer add overhead to each TLP, thereby further reducing the effective data transfer rate.

For our experiments for on-chip integration, we assume an AXI interconnect with the DDR2 data bandwidth of 64bits, memory clock of 400MHz, and a the maximum bandwidth of 6400MBps [20]. For the PCI-Express Gen3 we use the effective Bandwidth of 1GBps.

4 IMPLEMENTATION RESULTS

A total of eight machine learning algorithms including Kmeans, K nearest neighbor (KNN), singular value decomposition (SVD), support vector machine (SVM), hideen markov models (HMM), logistic regression (LR), collaborative filtering (CF) and Naive Bayes are implemented in Hadoop MapReduce, based on [22], [23], [24], [25], [26], [27] ,[28] and [29], respectively. Subsequently, we profile each application with 1 TB of input data, with HDFS block size of 64MB, 8 mappers on a 4-node cluster of Intel Atom and Intel Xeon servers. The processor cores deployed in these two servers have different microarchitectures (see Table 2). Therefore it is important to consider the microarchitecture differences when comparing the two design. We take into the capital cost when comparing these two servers post FPGA acceleration.

Due to space limitation, when we present the results per benchmark, we only report KNN, Kmeans, SVM, and Naive Bayes results. The average results, however, takes all studied applications data into account.

4.1 FPGA results

The map functions of the studied applications are profiled for 64MB (The size of the HDFS block size) using GPROF on both Atom and Xeon. Based on the profiling, for each application one sub-function is selected within the map phase, which takes up most of the execution time, to be offloaded onto the FPGA. Most of these sub-functions include adder trees, and a collective of MAC units. The Vivado high-level synthesis tool (HLS) is used to automatically generate accelerator IPs for the selected sub-functions. Pipelining,

Table 3: FPGA implementation of map one instance of sub-function on Artrix, (multiple instances can work in parallel to enhance the processing power)

Resources	Available	Utilization (%)			
		SVM	K-means	KNN	Naive Bayes
FF	35200	24.2	8.7	4.2	22.3
LUT	17600	22.4	40.9	7.1	15.1
BRAM	60	13.04	32.2	47.5	32.5
DSP48	80	4	30	0	0
Maximum frequency [MHz]		86	125	300	333
T_i on Artrix [ns]		4431	8	6.66	3
T_L on Artrix [ns]		40646	208	198	21

Table 4: FPGA implementation of one instance of map sub-functions on Virtex (multiple instances can work in parallel to enhance the processing power.

Resources	Available	Utilization (%)			
		SVM	K-means	KNN	Naive Bayes
FF	301440	7.2	2.0	2.1	9.2
LUT	150720	14.3	4.2	6.3	5.2
BRAM	832	2.1	32.1	13.2	7.1
DSP48	768	1	7	2	0
Maximum frequency [MHz]		173	289	289	333
T_i on virtex [ns]		2010	3.3	3.33	3
T_L on virtex [ns]		34726	130	50	21

loop-unrolling and other techniques are used to reduce the latency and most importantly the interval (the time before the accelerator can initiate a new set of reads and process the next set of input data). The throughput and FPGA power of the accelerator are proportional to the inverse of the interval (T_i), however, the first data will only be ready after T_L (see Fig. 3). Based on this data the time per each accelerator call is calculated and compared to the time per call on Atom and Xeon. Table 3 and 4 shows the results on Artrix-7 and Virtex-6 FPGAs, respectively.

Based on the results in Table 3 and 4, the time per call for each accelerator is T_i, however, the first results is generated on T_L after which, one output is generated after each T_i seconds (Fig. 3). The FPGA resources and FPGA processing time of each accelerated sub-function is different based on the levels of granularity that is explored in the the acceleration of the map function. For instance, in SVM the inner product for a large vector is implemented on the FPGA, while for KNN a small adder tree is accelerated.

The resource utilization in Table 3 and 4 shows that none of the accelerators take up all the resources, even for an FPGA as small as Artix, suggesting that where the transfer bandwidth allows, multiple instances of the same accelerator can be implemented on the FPGA and work in parallel. Similarly, common accelerators for different functions can be offloaded to the same FPGA, eliminating the need to re-program the FPGA for each new application.

Based on the results in Table 3 and 4, we compare the resource utilization and FPGA processing time between a high-end Virtex FPGA and a much smaller low-end Artix FPGA. The results show that as expected, the high-end FPGA can achieve higher frequencies. Moreover they allow more pipelining, which in turn reduces

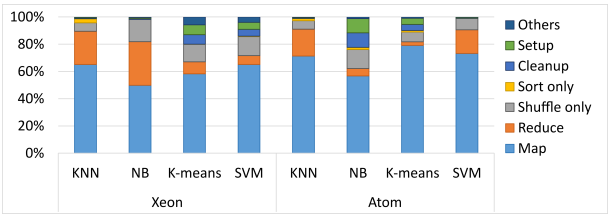

Figure 4: Breakdown of the Hadoop timing on various phases.

the interval time. The high-end FPGA incorporates more resources in terms of flip-flops, BRAMS, LUTs allowing more instances to be configured and run in parallel. However this does not necessarily mean whether using a high-end FPGA is more beneficial for speedup purposes. This can be the case only if the execution time on the FPGA is one of the bottlenecks of the overall design.

Based on the the number of bytes required for each call of the accelerated sub-function, and the number of sub-function calls during the processing of one HDFS block in the map phase, we calculate the data transfer overhead for various types of interconnects. It should be noted that, since the accelerator is pipelined, the incoming data does not need to wait for the previous processing to be finished. While in absence of data dependency the transfer time is masked by the accelerator processing time of the previous data, in case the data transfer rate is slower than the processing rate of the accelerator, the transfer time becomes the bottleneck. Thus, the execution time of the map function is determined by the maximum of the accelerator processing time and the data transfer time. Regardless, the latency for the the first accelerator call and transfer latency of the first batch of data is added to the execution time, which is negligible when the number of sub-function calls is high (which is the case is our experiments).

4.2 Execution time after acceleration

To calculate the contribution of the map phase acceleration in an end-to-end Hadoop MapReduce environment, we illustrate the breakdown of the timing of different phases for the studied applications in Fig. 4, which shows that the fraction of the time spent in the map phase is higher for the Atom server compared to Xeon. This is expected, since the computation-intensive part of the MapReduce lies in the map and reduce phase, and Atom server is designed with less aggressive superscalar features compared to Xeon. Based on this figure, we expect Atom server to gain higher speedup through acceleration.

Fig. 5 shows the execution time per GB of the accelerated MapReduce at various transfer bandwidths for the studied applications. We also include the non-accelerated execution time in the captions for reference.

As shown in Table 3 and 4, the FPGA resources of the both FGPAs allow multiple substantiations of each accelerator, thus we report the results for both FPGAs with maximum resource utilization, .i.e., we instantiate as many accelerators as possible on each FPGA. It should be noted, that while making a higher use of available resources of the FPGA increases the processing capability of the platform and is beneficial in terms of energy efficiency (The utilization of the FPGA does not affect its power consumption significantly), in absence of sufficient transfer bandwidth, it does not enhance the performance of the end-to-end system; and, conversely, it increases the need for larger number of buffers resulting in increased execution time due to transmission delays.

4.3 power

An important benefit of HW+SW acceleration is the improvement in the energy-efficiency. General-purpose CPUs such as Atom and Xeon are not designed to provide energy-efficient solution for every application. Accelerators help improve the efficiency by not only speeding up the execution time, but also executing the task with just enough required hardware resources. To this end, we measure the power and calculate the energy delay product (EDP) both before and after the acceleration.

Wattsup pro meter [30] was used for power readings on Xeon and Atom servers. We measured the average power for individual phases on Xeon and Atom using Wattsup pro power meter. Moreover, for each mapper function on the FPGA board, we used picoScope digital oscilloscope. For the power dissipated by the transfer of data, we investigated two specific cases, .i.e. AXI-interconnect and the PCI-e Gen3. The AXI interconnect is implemented on the FPGA, and the power is calculated from the FPGA. For the PCI-e, we use the specifications for the synopsys low Power PCI Express 3.1 IP Solution in [31], which reduces standby power to less than 10 uW/lane and active power to well below 5 mW/Gb/lane while meeting the PCIe Gen-3.1 electrical specification.

5 DISCUSSION

To better understand how various architectural decisions affect the performance, power and energy efficiency, the results for each application were normalized to the largest number. Due to space limitations the average results are presented in Fig. 6, where the top bar shows the least efficient design and the bottom bar shows the most efficient one.

5.1 Integration technology between FPGA and CPU

As we deploy a higher-end FPGA, the integration technology between FPGA and CPU become more influential in performance. Moreover, saturating point is pushed to the right, as shown in the Fig. 5z.

Fig. 6-a shows that the off-chip integration of the FPGA increases the average execution time of the application. Fig. 6-b shows that, by deploying an efficient PCI-e [31], with reduced standby and active power, the power of the off-chip FPGA is lower than the higher transfer rate off-chip FPGAs. Taking into account both execution delay and power in EDP and ED2P metrics, Fig. 6-c show that on-chip integration is a more efficient design, regardless of the type of CPU and the FPGA technology that is being deployed.

5.2 FPGA Technology

Results show that on-chip integration benefits more from an advanced FPGA technology, since at transfer rates achievable only by the on-chip integration of the core and the FPGA, the transfer rate is less likely to become the performance bottleneck. While, Fig. 6 shows that deploying a lower-end FPGA reduces the power consumption, taking into account both execution delay and power, EDP results show that a high-end FPGA is still more energy-efficient.

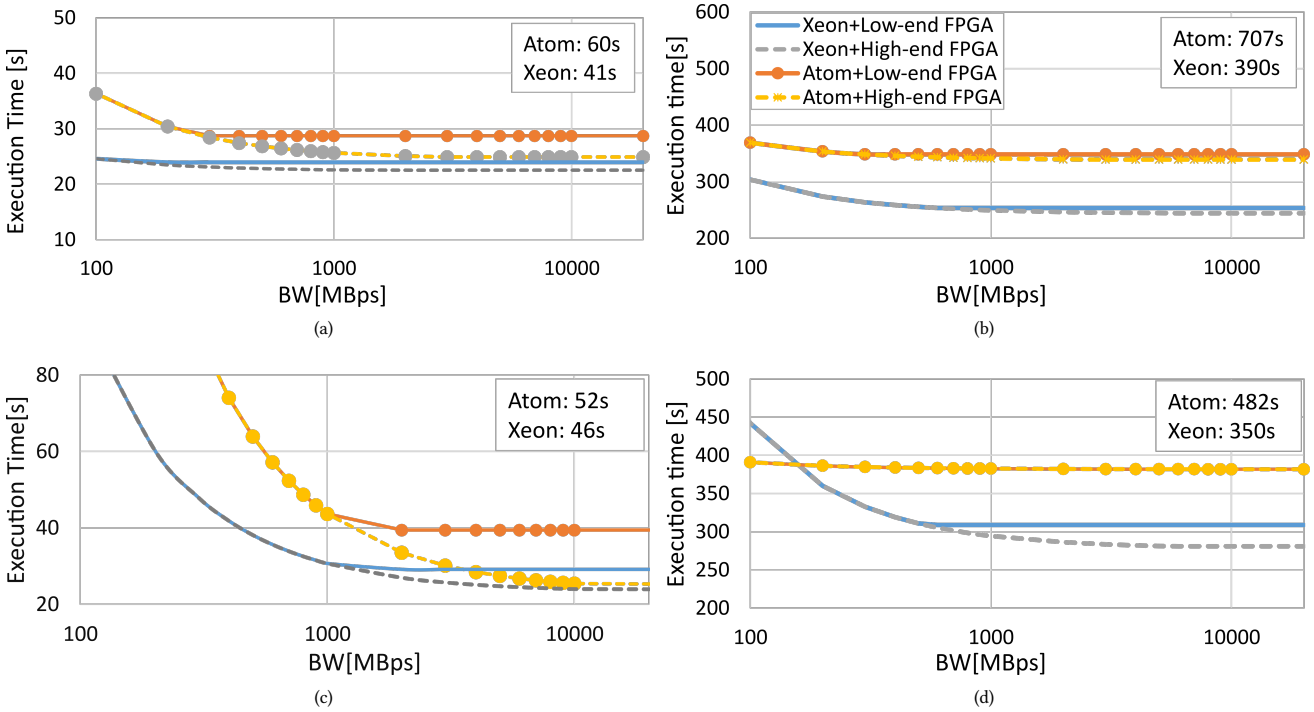

Figure 5: Execution time per GB of the accelerated MapReduce at various transfer bandwidths for (a) Kmeans (b) KNN (c) SVM (d) Naive Bayes.

5.3 CPU technology

The results shows that the acceleration reduces the execution time on Atom more significantly than that of Xeon, reducing the gap between the Atom and Xeon. Moreover, Fig. 5 shows that the impact of the type of FPGA is more significant on Xeon at high transfer rates, since the faster execution of the software parts of the application on Xeon pushes the bottleneck toward the hardware part of the application, allowing more room for performance gains by deploying a higher-end FPGA technology. While the power reduction due to the acceleration is more significant on Xeon reducing the power gap between the two servers, even after the acceleration, the power consumption of the Atom and Xeon servers are not comparable.

5.4 Interplay effect of design parameters

Fig. 7 sums up the normalized results to better understand the importance of architectural parameters on both power and performance. As expected, Atom exhibits higher execution delays and lower power, while Xeon has lower execution time and higher power. We use the gap between various data points presented in the figure to find architectural insights. A major observation from Fig. 7 and Fig. 6 is that the type of CPU is the most important factor in determining the execution time and the power. The integration technology is the second important design parameter considering both power and execution time. The type of FPGA becomes of high importance only when the integration technology allows fast transfer of data between the CPU and the FPGA.

Figure 8 shows the capital cost of studied applications normalized to the cost of Atom server. We utilize conventional costs for various components with Xeon, low-end FPGA and high-end FPGA having 4.1×, 2.5×, and 30% of the cost of Atom server. The figure shows that while execution time of Xeon+High-end FPGA is the lowest, they are not the most cost-efficient solutions. On the other hand adding low-cost low-end FPGA would be a cost effective solution to get enhanced performance without having to move to an expensive server.

6 RELATED WORK

The performance and bottlenecks of Hadoop MapReduce have been extensively studied in recent work [32–36]. To enhance the performance of MapReduce and based on the bottlenecks found for various applications, hardware accelerators are finding their ways in system architectures. While the GPU-based platforms have achieved significant speedup across a wide range of benchmarks [37, 38], their high power demands preclude them for energy-efficient computing [39]. Alternatively, FPGAs have shown to be more energy efficient [40]. Moreover, they allow the exploitation of fine-grained parallelism in the algorithms.

Microsoft Catapult project [2] has built a composable, reconfigurable fabric consisting of 2-D torus of Stratix V FPGAs accessible through PCIe. Alternatively, Heterogeneous architecture research platform (HARP), integrates Intel CPU with Altera FPGAs.

Such exiting FPGA platforms have been deployed to accelerate MapReduce applications [3, 4, 41–52]. While their reported performance and/or power boost is significant, their scope is limited to one particular architecture with either on-chip or off-chip FPGA, and few particular applications. However, this work analyzes a

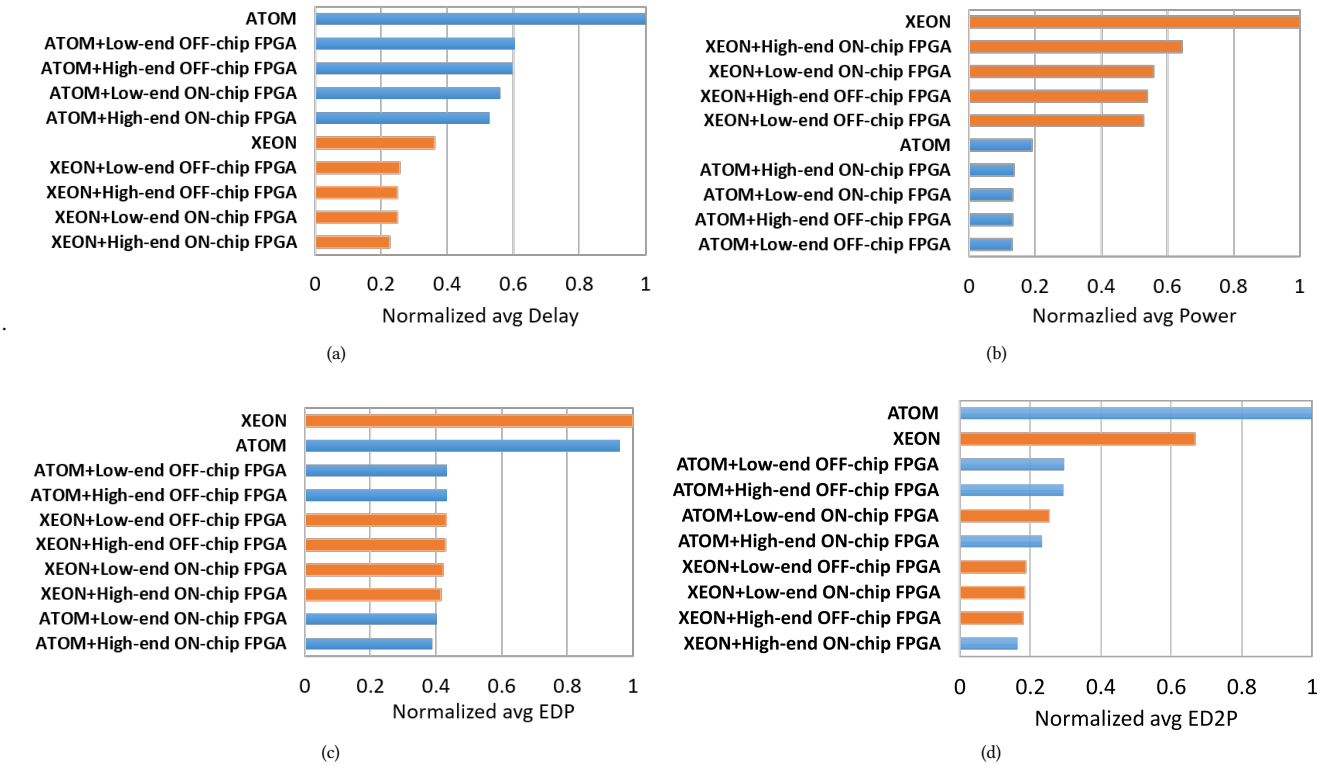

Figure 6: Average normalized results for (a) delay (b) power (c) EDP and (d) ED2P.

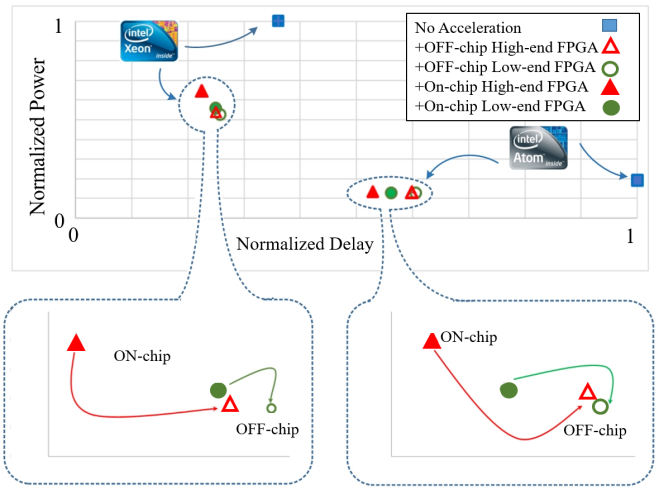

Figure 7: Interplay effect of the parameters on power and execution delay.

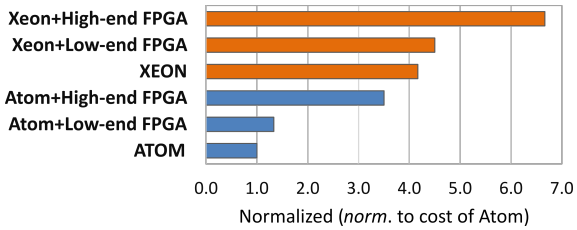

Figure 8: Cost comparison for various architectures.

large number of machine learning kernels , and explores a large architecture space which spans FPGA technology (high-end vs low-end), core technology (big vs little core server), and interconnection technology (on-chip vs off-chip) to understand the benefits of hardware acceleration and power and performance trade-offs offered in each of these design points to gains architectural insights.

7 CONCLUSIONS

Given the rise of hardware accelerators for big data analytics in recent years, this paper answers the important research question of how to architect a heterogeneous hardware accelerator to meet the performance, power, and energy-efficiency requirements of a diverse range of ML-based analytics applications. This paper first analyzed several important machine learning algorithms in Hadoop MapReduce to understand the power and performance benefits of HW acceleration in a HW+SW co-design framework. It then explored a large architecture space which spans FPGA technology (high-end vs low-end), CPU technology (big high performance vs

little low power server), and interconnection technology (on-chip vs off-chip) to understand the benefits of hardware acceleration and power and performance trade-offs offered in each of these design points to gains architectural insights. Our sensitivity analysis results revealed that among the three architectural parameters, the type of CPU is the most dominant factor influencing the execution time and power in a CPU+FPGA architecture. The integration technology and FPGA type come next, with the power and performance least sensitive to FPGA type. We conclude that with the available off-chip interconnection protocols such as PCI-e Gen3, which allow transmission bandwidths of upto 1GBps, off-chip integration of FPGA and CPU is the most cost-effective solution for Hadoop MapReduce applications, since, it allows a more flexible combination of CPU+FPGA in which many types of CPU can be integrated with several types of FPGA.

REFERENCES

[1] A. M. Caulfield and et. al, "A cloud-scale acceleration architecture," in *Microarchitecture (MICRO), 2016 49th Annual IEEE/ACM International Symposium on.* IEEE, 2016, pp. 1–13.

[2] A. Putnam and et al., "A reconfigurable fabric for accelerating large-scale datacenter services," in *Computer Architecture (ISCA), 2014 ACM/IEEE 41st International Symposium on*, June 2014, pp. 13–24.

[3] D. Yin, G. Li *et al.*, "Scalable mapreduce framework on fpga accelerated commodity hardware," in *Internet of Things, Smart Spaces, and Next Generation Networking*. Springer, 2012, pp. 280–294.

[4] Y. e. a. Shan, "FPMR: Mapreduce framework on fpga," in *FPGA*, 2010, pp. 93–102.

[5] K. Neshatpour, M. Malik, M. A. Ghodrat, A. Sasan, and H. Homayoun, "Energy-efficient acceleration of big data analytics applications using fpgas," in *Big Data (Big Data), 2015 IEEE International Conference on.* IEEE, 2015, pp. 115–123.

[6] E. Ghasemi, "A scalable heterogeneous dataflow architecture for big data analytics using fpgas," Ph.D. dissertation, University of Toronto (Canada), 2015.

[7] Y.-k. Choi and et. al, "A quantitative analysis on microarchitectures of modern cpu-fpga platforms," in *Design Automation Conference (DAC), 2016 53nd ACM/EDAC/IEEE.* IEEE, 2016, pp. 1–6.

[8] Y.-M. Choi and H.-H. So, "Map-reduce processing of k-means algorithm with fpga-accelerated computer cluster," in *IEEE ASAP*, June 2014, pp. 9–16.

[9] C. Kachris, D. Diamantopoulos, G. C. Sirakoulis, and D. Soudris, "An fpga-based integrated mapreduce accelerator platform," *Journal of Signal Processing Systems*, vol. 87, no. 3, pp. 357–369, 2017.

[10] S. W. Ho, M. Z. Ding, P. S. Lim, D. I. Cereno, G. Katti, T. C. Chai, and S. Bhattacharya, "2.5d through silicon interposer package fabrication by chip-on-wafer (cow) approach," in *2014 (EPTC)*, Dec 2014, pp. 679–683.

[11] "Heterogeneous 2.5d integration on through silicon interposer," in *Applied Physics Reviews*, 2015.

[12] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," in *OSDI 2004*, 2004.

[13] N. e. Hardavellas, "Toward dark silicon in servers," *IEEE Micro*, vol. 31, pp. 6–15, 2011.

[14] "Hadoop programming with arbitrary languages," https://rcc.fsu.edu/docs/hadoop-programming-arbitrary-languages.

[15] "Intel vtune amplifier xe performance profiler." http://software.intel.com/en-us/articles/intel-vtune-amplifier-xe/, accessed: 2014-11-30.

[16] "Vivado design suite user guide: High-level synthesis," http://www.xilinx.com/support/documentation/sw_manuals/xilinx2013_2/ug902-vivado-high-level-synthesis.pdf.

[17] T. White, *Hadoop: The Definitive Guide.* O'Reilly Media, Inc., 2009.

[18] L. A. Cortes, L. Alej, R. Corts, P. Eles, and Z. Peng, "A survey on hardware/software codesign representation models," 1999.

[19] P. Eles and et. al, "Scheduling of conditional process graphs for the synthesis of embedded systems," in *DATE*, 1998.

[20] "Logicore axi dma," *Product Guide for Vivado Design Suite*, 2013.

[21] "white paper, understanding performance of pci express sytems," https://www.xilinx.com/support/documentation/white_papers/wp350.pdf.

[22] W. Zhao, H. Ma, and Q. He, "Parallel k-means clustering based on mapreduce," in *Proceedings of the 1st International Conference on Cloud Computing*, 2009, pp. 674–679.

[23] P. P. Anchalia and K. Roy, "The k-nearest neighbor algorithm using mapreduce paradigm," in *Proceedings of the Fifth International Conference on Intelligent Systems, Modelling and Simulation*, 2014, pp. 513–518.

[24] B. Bayramli, "Svd factorization for tall-and-fat matrices on map/reduce architectures," *arXiv:1310.4664*, 2014.

[25] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[26] A. J. Mount, "Large data logistic regression (with example hadoop code)," http://www.win-vector.com/blog/2010/12/large-data-logistic-regression-with-example-hadoop-code/, accessed: 2015-5-12.

[27] S. Huang, X. Jiang, N. Zhang, C. Zhang, and D. Dang, "Collaborative filtering of web service based on mapreduce," in *2014 International Conference on Service Sciences*, May 2014, pp. 91–95.

[28] G. Fung and O. L. Mangasarian, "Incremental support vector machine classification," in *7 th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, 2001, pp. 77–86.

[29] N. H. N., "Naive bayes," *IEEE Micro*, vol. 31, pp. 6–15, 2011.

[30] "Wattsuppro power meter," http://www.wattsupmeters.com/secure/index, accessed: 2014-11-30.

[31] "Industry's lowest power pci express 3.1 ip solution for mobile socs," http://www.synopsys.com/designware.

[32] T. Honjo and K. Oikawa, "Hardware acceleration of hadoop mapreduce," in *IEEE Int. Conf. Big Data*, Oct 2013, pp. 118–124.

[33] D. Jiang, B. C. Ooi, L. Shi, and S. Wu, "The performance of mapreduce: An in-depth study," *Proc. VLDB Endow.*, vol. 3, no. 1-2, pp. 472–483, Sep. 2010. [Online]. Available: http://dx.doi.org/10.14778/1920841.1920903

[34] C. Tian, H. Zhou, Y. He, and L. Zha, "A dynamic mapreduce scheduler for heterogeneous workloads," in *Grid and Cooperative Computing, 2009. GCC '09. Eighth International Conference on*, Aug 2009, pp. 218–224.

[35] M. Zaharia, A. Konwinski, A. D. Joseph, R. Katz, and I. Stoica, "Improving mapreduce performance in heterogeneous environments," in *OSDI*, Berkeley, CA, USA, 2008, pp. 29–42.

[36] M. Malik, A. Sasan, R. Joshi, S. Rafatirah, and H. Homayoun, "Characterizing hadoop applications on microservers for performance and energy efficiency optimizations," in *ISPASS.* IEEE, 2016, pp. 153–154.

[37] J. A. Stuart and J. D. Owens, "Multi-gpu mapreduce on gpu clusters," in *IPDPS*, Washington, DC, USA, 2011, pp. 1068–1079.

[38] B. He and et al., "Mars: A mapreduce framework on graphics processors," in *PACT*, 2008, pp. 260–269.

[39] L. Stolz, H. Endt, M. Vaaraniemi, D. Zehe, and W. Stechele, "Energy consumption of graphic processing units with respect to automotive use-cases," in *Energy Aware Computing (ICEAC), 2010 International Conference on*, Dec 2010, pp. 1–4.

[40] J. Fowers, G. Brown, P. Cooke, and G. Stitt, "A performance and energy comparison of fpgas, gpus, and multicores for sliding-window applications," in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, ser. FPGA '12, New York, NY, USA, 2012, pp. 47–56.

[41] Z. Lin and P. Chow, "Zcluster: A zynq-based hadoop cluster," in *2013 FTP*, Dec 2013, pp. 450–453.

[42] C. Kachris, G. C. Sirakoulis, and D. Soudris, "A configurable mapreduce accelerator for multi-core fpgas (abstract only)," in *Proc ACM/SIGDA Intl Symp FPGAs*, 2014, pp. 241–241.

[43] K. Neshatpour, M. Malik, and H. Homayoun, "Accelerating machine learning kernel in hadoop using fpgas," in *Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on.* IEEE, 2015, pp. 1151–1154.

[44] K. Neshatpour, M. Malik, M. A. Ghodrat, and H. Homayoun, "Accelerating big data analytics using fpgas," in *Field-Programmable Custom Computing Machines (FCCM), 2015 IEEE 23rd Annual International Symposium on.* IEEE, 2015, pp. 164–164.

[45] K. Neshatpour, A. Koohi, F. Farahmand, R. Joshi, S. Rafatirad, A. Sasan, and H. Homayoun, "Big biomedical image processing hardware acceleration: A case study for k-means and image filtering," in *Circuits and Systems (ISCAS), 2016 IEEE International Symposium on.* IEEE, 2016, pp. 1134–1137.

[46] K. Neshatpour, A. Sasan, and H. Homayoun, "Big data analytics on heterogeneous accelerator architectures," in *CODES+ISSS*, 2016, p. 16.

[47] K. Neshatpour, M. Malik, A. Sasan, S. Rafatirad, T. Mohsenin, H. Ghasemzadeh, and H. Homayoun, "Energy-efficient acceleration of mapreduce applications using fpgas," *Journal of Parallel and Distributed Computing*, vol. 119, pp. 1–17, 2018.

[48] M. Malik, K. Neshatpour, T. Mohsenin, A. Sasan, and H. Homayoun, "Big vs little core for energy-efficient hadoop computing," in *2017 Design, Automation & Test in Europe Conference & Exhibition (DATE).* IEEE, 2017, pp. 1480–1485.

[49] M. Malik, K. Neshatpour, S. Rafatirad, and H. Homayoun, "Hadoop workloads characterization for performance and energy efficiency optimizations on microservers," *IEEE Transactions on Multi-Scale Computing Systems*, 2017.

[50] H. Sayadi, D. Pathak, I. Savidis, and H. Homayoun, "Power conversion efficiency-aware mapping of multithreaded applications on heterogeneous architectures: A comprehensive parameter tuning," in *ASP-DAC*, 2018.

[51] H. Sayadi, N. Patel, A. Sasan, and H. Homayoun, "Machine learning-based approaches for energy-efficiency prediction and scheduling in composite cores architectures," in *ICCD*, 2017.

[52] K. Neshatpour, H. Mohammadi Makrani, A. Sasan, and H. H. Hassan Ghasemzadeh, Setareh Rafatirad, "Design space exploration for acceleration of machine learning applications," *FCCM*, 2018.