ACCELERATING DATA ANALYTICS - 50X FASTER APACHE SPARK RANDOM FOREST CLASSIFICATION

White Paper

Xelera Technologies GmbH
Alexander Lange
Alexander.Lange@xelera.io

Abstract

Modern companies are increasingly challenged to adapt to the rapid data growth and real-time processing requirements, i.e. the necessity to process raw data in ever shortening time intervals. As we show in a benchmark below, advanced Big Data analytics (building a machine learning model with a Random Forest classifier) using the popular Apache Spark Framework can take up to hours, even on moderate data set sizes. Secondly, this whitepaper presents a Xelera solution for the same data analytics problem which utilizes hardware accelerators like Field Programmable Gate Arrays (FPGAs) and Graphics Processing Units (GPUs) to accelerate such processes by up to 50 times. Furthermore, the Xelera Suite software provides features for easy-to-use scale-up/scale-out capabilities, virtualization/containerization and the monitoring of these accelerators in a cross-hardware, cross-platform and cross-language environment. The customer benefits are three-fold: faster processing, cost-reduction and higher flexibility.

The Real-Time Challenge

Companies which run commercial software applications inside datacenters or on the cloud are facing one of their biggest challenges of the current and the upcoming decades: More and more data has to be processed and analyzed in shorter periods of time. The rapidly increasing requirements originate from the global growth of data. More than 2,5 exabyte (10^{18}) are produced every day. According to the International Data Corporation (IDC) the global amount of data is doubled every two years (Garrity, 2014). As a result, traditional server architectures are struggling to keep up with the rising demand of processing capabilities.

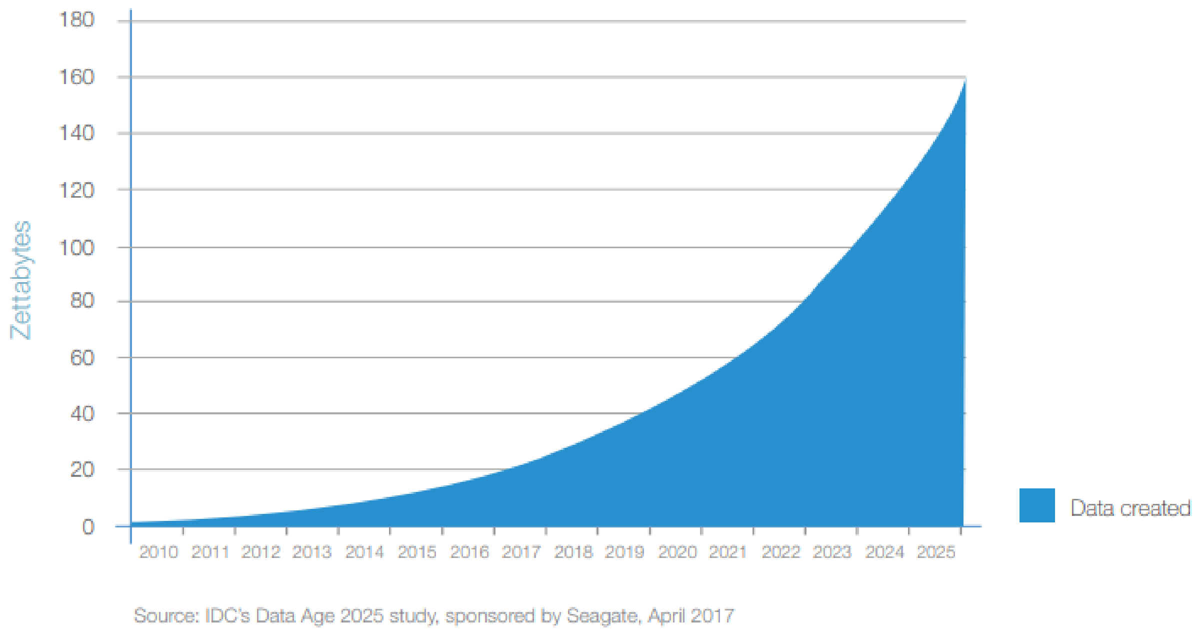

Figure 1 - Annual Size of the Global Datashere

A simple example to clarify the problem could be the following: A data analyst wants to create a model for a new advertisement campaign based on all the customer behavior his company has gathered within the previous months. Nowadays this data does not only consist of what customers have bought and what not. Instead every single click they do on their way to the potential purchase can be utilized for such a model. Although the creation of such a model out of a big amount of data is typically done inside a big cluster of multiple server machines, it can take up to several hours or even days. If the result is unsatisfying it must be redone with different settings and parameters. That way the whole process can easily take up weeks. This, however, brings data analytics into a conflict with one of the biggest requirements of modern companies: real-time capability. Companies must be able to react immediately to all changes within the market in order to keep up with competitors. Data is generated at increasingly higher rates so that predictive models must be updated more and more frequently. In a survey from OpsClarity from 2016 already 65% of the companies have stated they were starting to implement real-time analytics and an additional 24% were planning to do so soon (Agarwal, 2016).

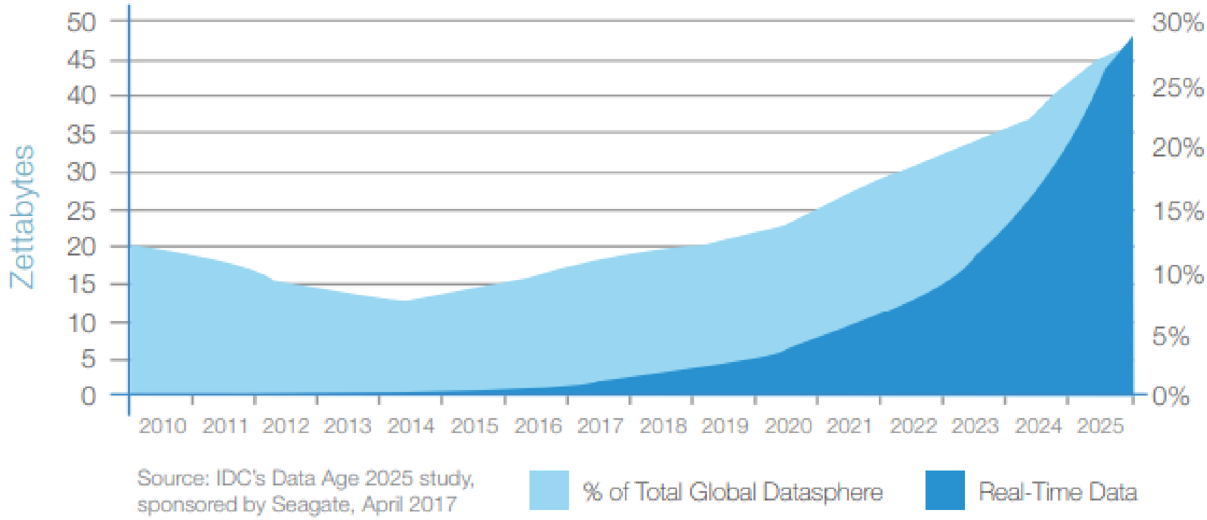

Figure 2 - Real-Time Data

Data Processing with Apache Spark

This paper addresses the challenges of accelerated data analytics based on the example of Random Forest Decision Tree Classification using the Machine Learning Library (MLlib) of the widespread large-scale data processing engine Apache Spark. Spark is designed to process data, either in batches or streamed, with high performance utilizing the power of whole compute clusters. It can be integrated in various clustering frameworks and supports hundreds of different data sources like databases or files. The MLlib is a part of Spark that contains a comprehensive collection of analytics functions, e.g. classification, regression, decision trees or clustering (Apache Spark Foundation, 2018).

Apache Spark provides excellent performance for a large variety of functions. It can supposedly run logistic regression algorithms 100x faster than its competitor Hadoop (Apache Spark Foundation, 2018). However, it still has weaknesses especially inside its MLlib.

Random Forest is a classification algorithm that generates multiple uncorrelated, randomized decision trees. These trees (the forest) are then used as models during the prediction whereby a majority vote decides which of all the predictions among the trees is selected (Koehrsen, 2018).

Xelera conducted a benchmark on Spark's Random Forest Classification in order to validate statements about its lack of speed and unproportionally high resource consumption. The test was performed on the Amazon Web Services (AWS) Cloud using a single r4.16xlarge instance which provides a powerful Intel Xeon E5-2686 v4 CPU with 64 virtual cores as well as 488 GB RAM for 4,256 $/h. A subset of 1,400,000 rows and 10 columns of the dataset “Airline on-time performance” from 2004 (Statistical Computing, 2009) was used. Three configurations with different numbers of decision trees as results were tested (more trees provide better algorithmic accuracy and require more work). As Figure 3 shows, even though only a medium sized dataset was processed on a powerful machine the algorithm took between 0,5 and 3,5 hours to finish. Bigger datasets could not be tested on a single machine because Apache Spark requires too much memory to run Random Forest and 488 GB were not enough for more than 1,4 million rows.

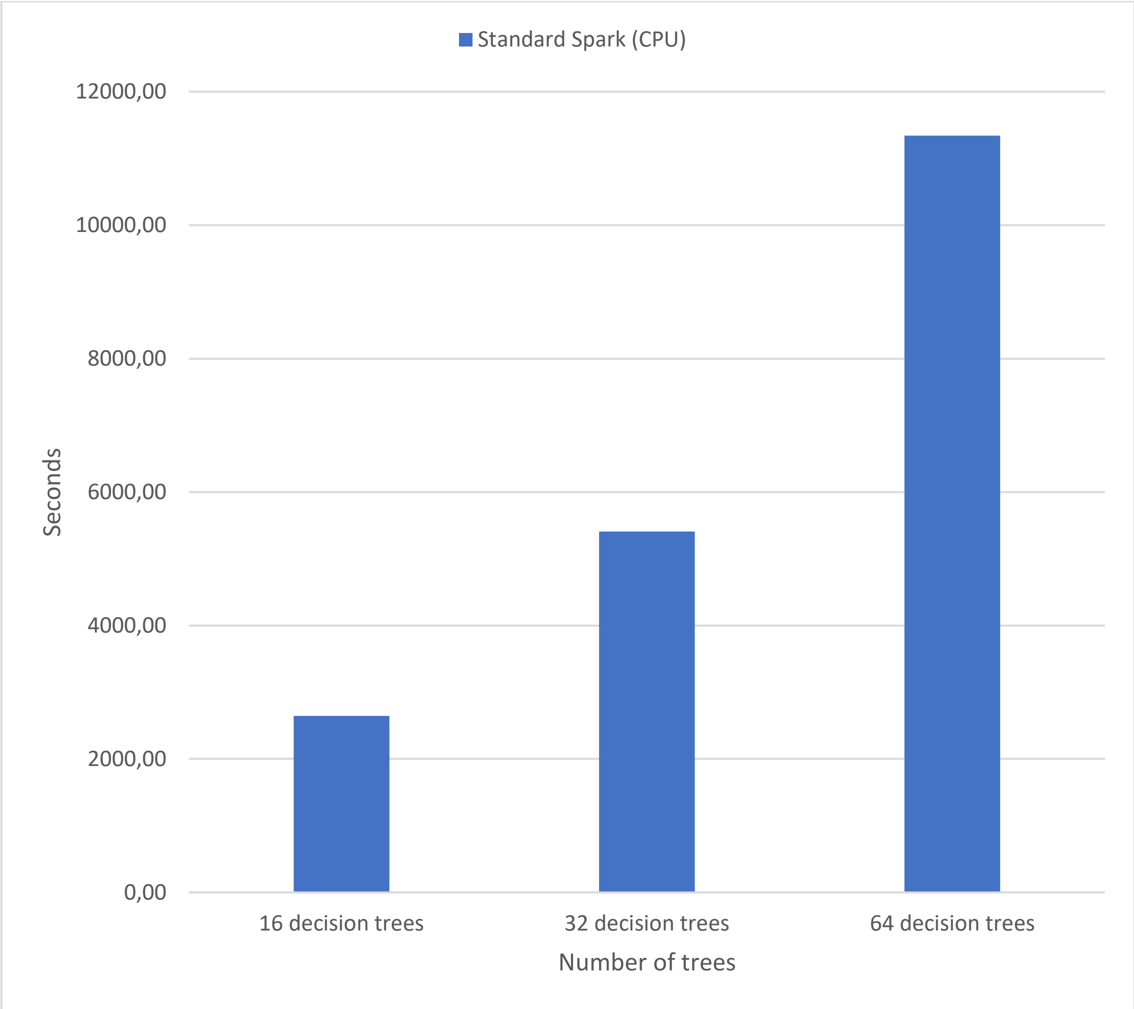

Figure 3 - Random Forest CPU Benchmark

Accelerating Analytics with Xelera

Xelera provides an efficient and easy-to-use method to accelerate this process and many others by utilizing acceleration platforms such as FPGAs and GPUs. The foundation of this system is the Xelera Suite which is shown in Figure 4.

The Xelera Suite is the core library that encompasses a large variety of features. It supports accelerator hardware such as GPUs from NVIDIA and AMD, FPGAs from Intel and Xilinx as well as traditional multi-core CPUs from Intel and AMD. It can be deployed on different cloud platforms like AWS, OTC and Nimbix, or on privately hosted on-premise servers. A cross-language interface enables Xelera as well as third-party developers to implement accelerator IP easily, similar to using NVIDIA's CUDA but compatible with all listed platforms and much more intuitive than OpenCL. The list is supplemented by a scale-up and scale-out control that distributes jobs across multiple accelerators on the same physical machine or a whole cluster using network functionalities for even higher performance boosts. The Xelera Suite can be deployed inside virtual machines and containers and offers integrations into management systems like Kubernetes and OpenStack. Monitoring, reporting and dashboards complete the experience by further enhancing the usability.

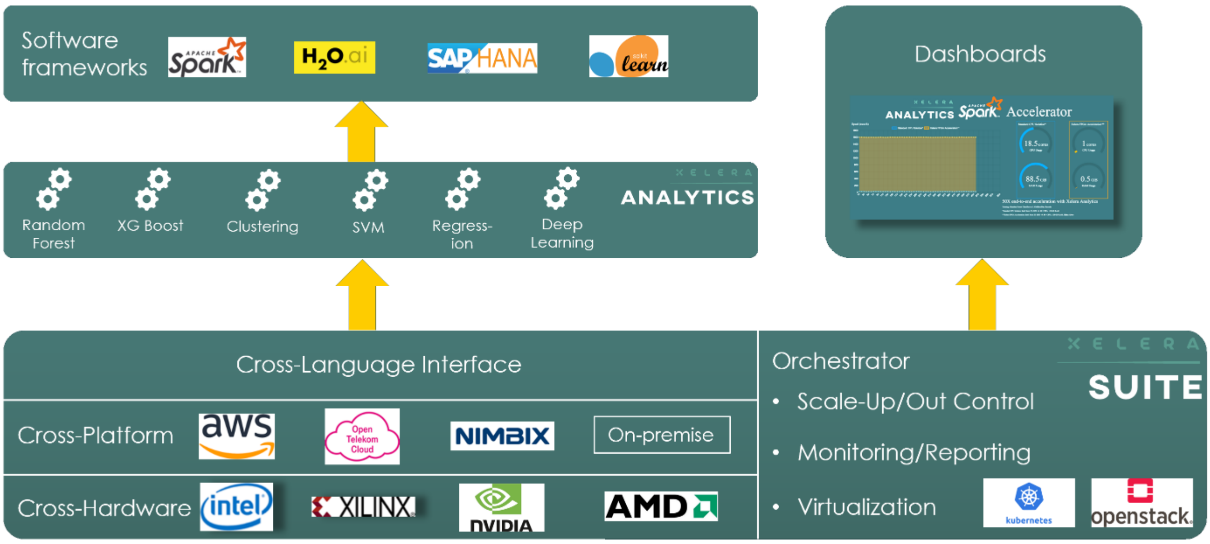

Figure 4 - Xelera Suite

Attached to the Xelera Suite are the three plugin families: Analytics, Net and DB with their respective purposes being to accelerate big data analytics/machine learning, network functionality and database access. This paper focusses on Xelera Analytics which currently encompasses a variety of popular machine learning algorithms such as Random Forest classification, clustering, regression and neural networks (Deep Learning). The plugins themselves are bitstreams and binaries which are loaded into their accelerator platforms by the Xelera Suite dynamically whenever needed.

The plugins on the other hand are integrated into high-level frameworks and user applications like Apache Spark or an SAP application system in order to accelerate already existing functions without any code change required by the user. Both, the plugins and their integration hooks are created by Xelera Technologies as well as by third-party developers.

In this paper, we demonstrate the integration into Apache Spark. As shown in Figure 5, the integration hooks directly into Spark's MLlib inside the Java Virtual Machine (JVM) replacing MLlib's machine learning functions. Instead of distributing tasks in the usual way, all information (dataset and parameters) are redirected using Xelera's standard API to the Xelera Suite, which is deployed on each physical or virtual machine that contains accelerators or Spark nodes. The Suite then takes care of distributing the task across all connected instances and deploys the required analytics plugins onto the destined accelerators. After the tasks are completed, the finished models are transferred back into Spark on the same way, where they can be used just as their native counterparts without any code changes required.

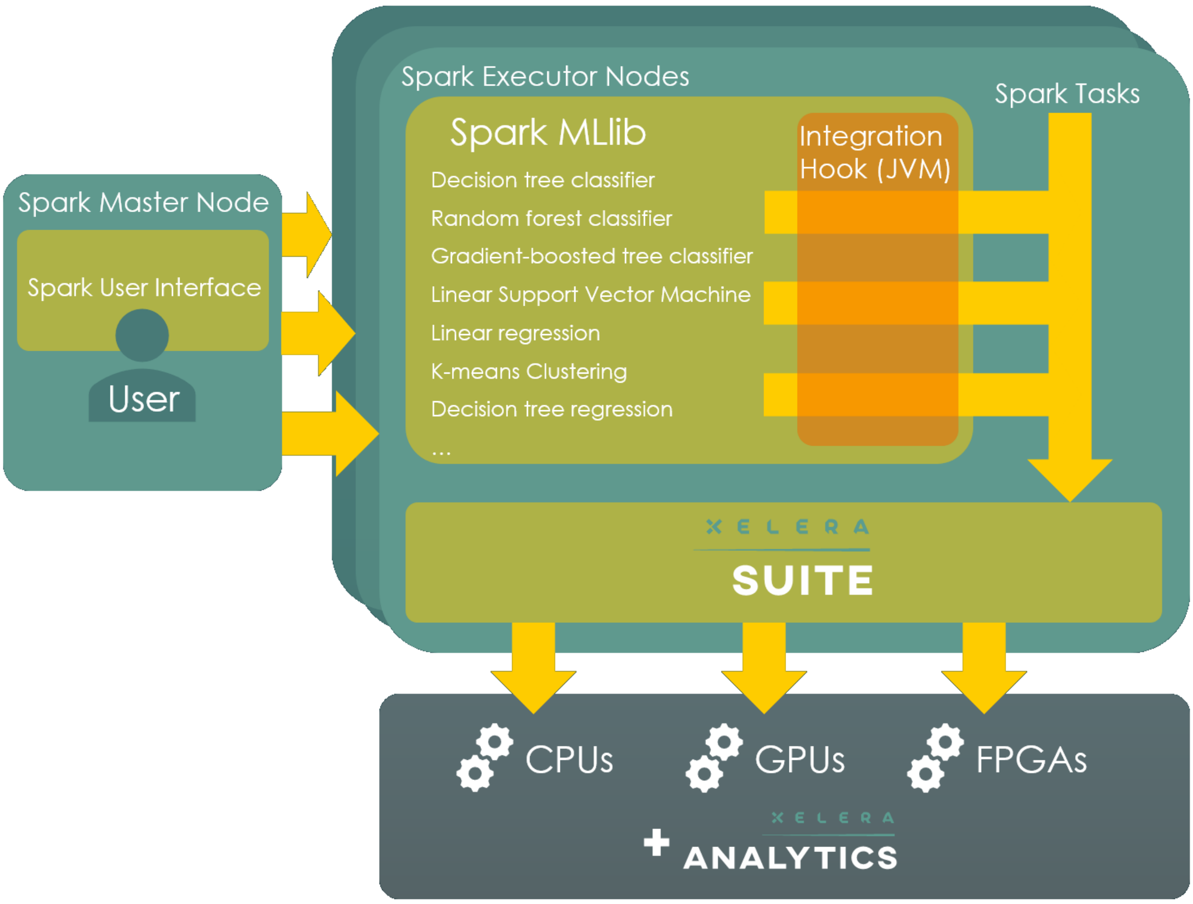

Figure 5 - Xelera Spark Integration

We conducted a benchmark by replacing the Random Forest Classifier of Apache Spark with the Xelera Analytics plugin for Xilinx FPGAs. The same script, dataset and parameters were used as in the native Spark benchmark presented in the previous chapter. The first test was performed on the Amazon Web Services (AWS) cloud using a single f1.2xlarge instance which provides an Intel Xeon E5-2686 v4 CPU with 8 virtual cores, 122 GB RAM and a Xilinx Virtex UltraScale+ VU9P FPGA. Again, the same three configurations with different amount of decision trees as results were tested. In a second test two f1.2xlarge instances were used to demonstrate the scale-up/scale-out capabilities of the Suite. The acceleration factor scales almost linearly with only a small loss from communication overhead. As Figure 6 shows the accelerated version is between 46 and 63 times faster per FPGA than the native CPU implementation while also scaling better with bigger models. This does not only save time but also resources. During the process Spark would require many cores and a huge amount of memory to compute the models. When this is offloaded to the FPGA the majority of processing takes place there instead and the host computer can be used for other tasks in the meantime.

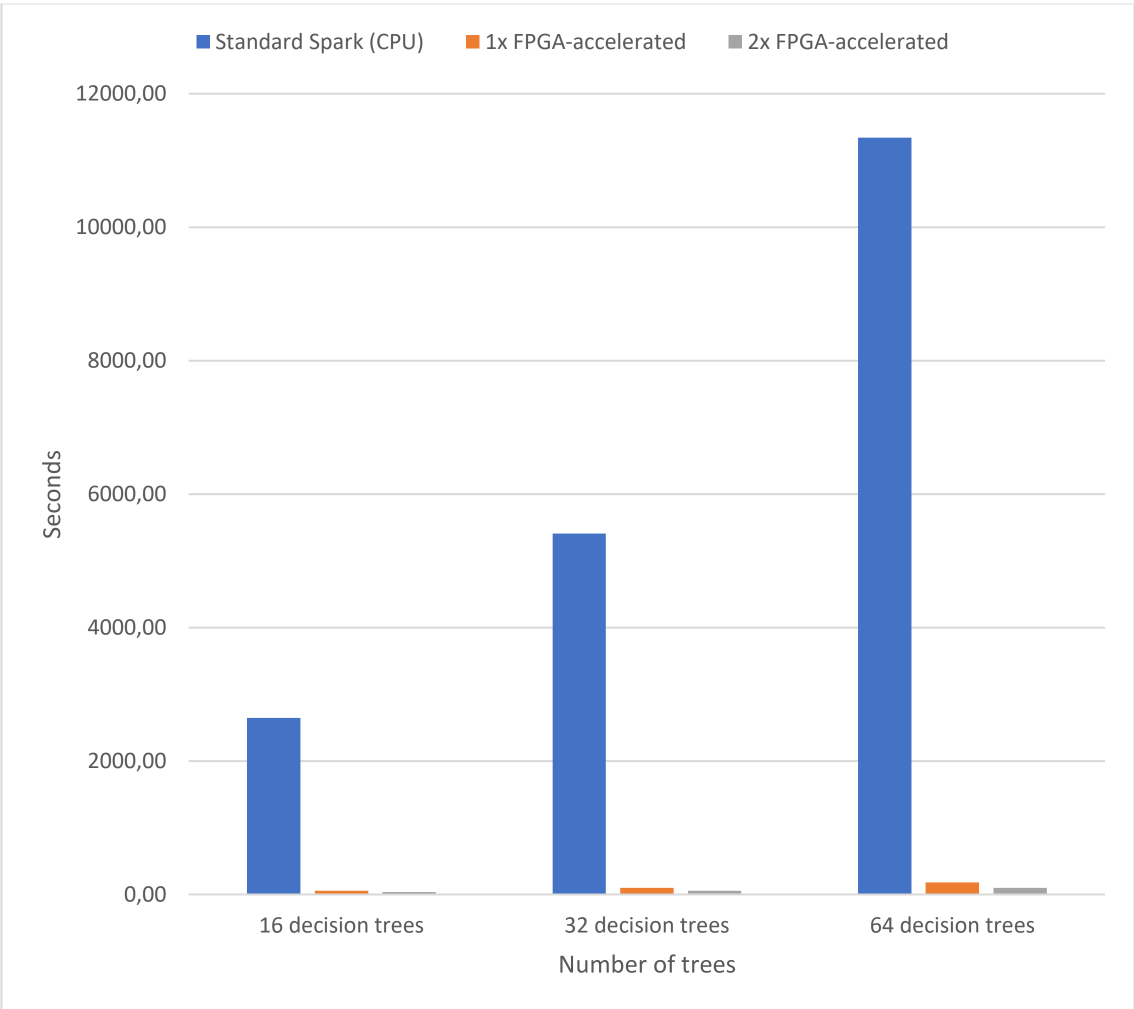

Figure 6 - Random Forest CPU and FPGA Benchmarks

Conclusion

Xelera offers an accessible and efficient platform for users to speed up compute intensive processes without requiring any knowledge about the utilized hardware accelerators and, for developers, an easy way to implement and integrate their own IP.

A significant performance boost like this has three major implications for customers:

Time is money. If we assume (simplified) that a cloud instance is rented solely for the purpose of running the random forest classification, then a 50 times shorter time runtime would also result in a 50 times shorter uptime for the instance. Additionally, the CPU instance used for the benchmark was significantly more powerful and expensive than the FPGA instance which in total would result in a cost reduction of 120x. $13,41 compared to $0,08 for a single iteration.

Accuracy increase. While machine learning models created by the Xelera IP generally have the same accuracy as their Spark equivalents, the higher processing speed allows data analysts to run their algorithms much more often. When they have more chances to adjust dataset selections and parameters, the accuracy can be increased which directly affects the business intelligence located above.

Real-time enterprises. Modern companies are not only facing the challenge of processing more data than ever before but also to do it in shorter timeframes. To be able to react to customer needs in real-time additional acceleration is a requirement since waiting hours for updated models is in many cases no longer possible. Xelera enables its customers to not only keep up with the rising amount of data but also enables them to develop completely new, real-time based business processes.

References

Agarwal, S. (2016, June 28). *OpsClarity*. Retrieved from 2016 State of Fast Data and Streaming Applications Survey: https://www.opsclarity.com/2016-state-fast-data-streaming-applications-survey/

Apache Spark Foundation. (2018). *MLlib*. Retrieved from Apache Spark: https://spark.apache.org/mllib/

Garrity, J. (2014). *The Internet of Everything: How the Network Unleashes the Benefits of Big Data.* Retrieved from World Economic Forum: http://www3.weforum.org/docs/GITR/2014/GITR_Chapter1.2_2014.pdf

Koehrsen, W. (2018, August 30). *Towards Data Science*. Retrieved from An Implementation and Explanation of the Random Forest in Python: https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76

Statistical Computing. (2009). *Airline on-time performance*. Retrieved from Statistical Computing: http://stat-computing.org/dataexpo/2009/