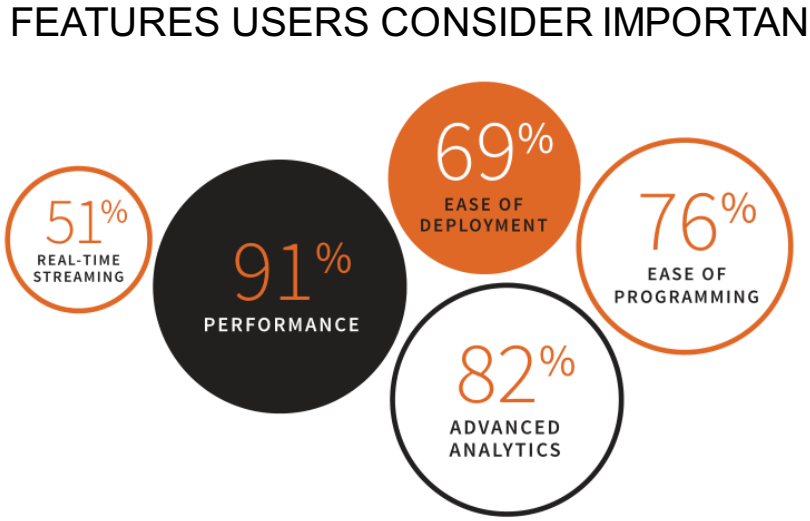
Speedup Spark Applications using FPGA Accelerators on the cloud

Christoforos Kachris
ICCS-NTUA

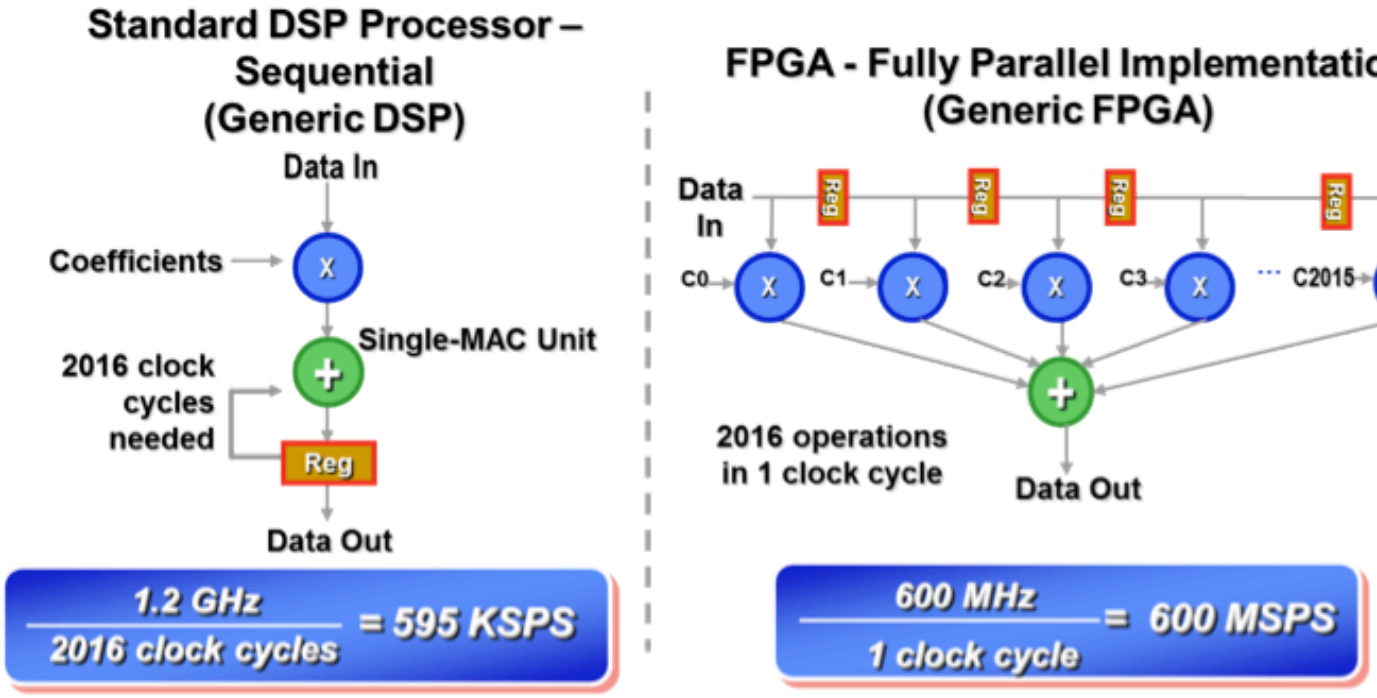
#EUres8

Performance in Spark

- 91% if Spark users care about performance



Accelerators versus Processors



[Source: National Instruments: Smart Grid Ready Instrumentation"

Flexibility versus Performance

CPU: High flexibility, lower efficiency

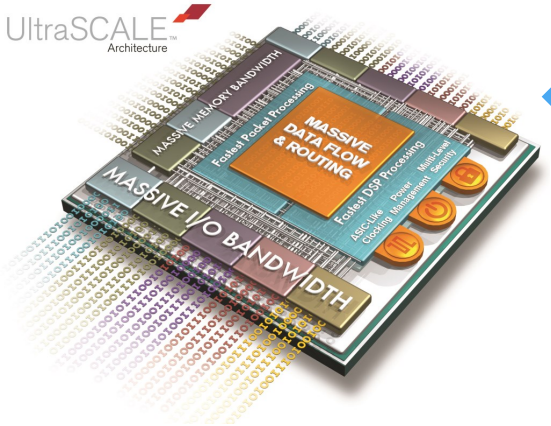
Accelerators: High throughput, higher efficiency

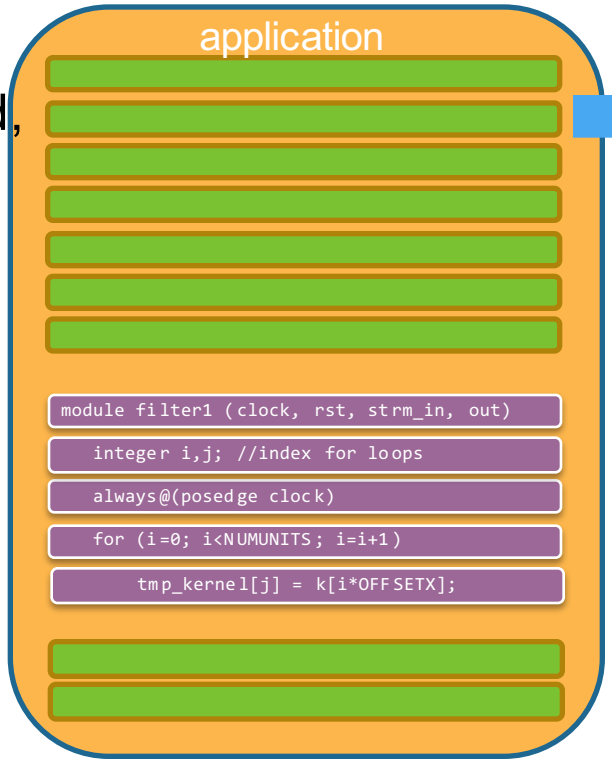
[Source: Amazon AWS f1]

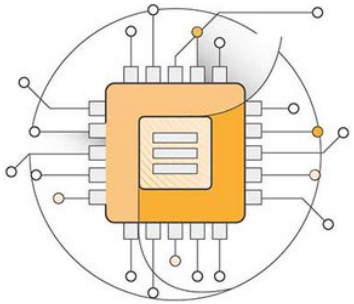
GPUs and FPGAs can provide massive parallelism and higher efficiency than CPUs for certain categories of applications

Hardware Acceleration

FPGA handles compute-intensive, deeply pipelined, hardware-accelerated operations



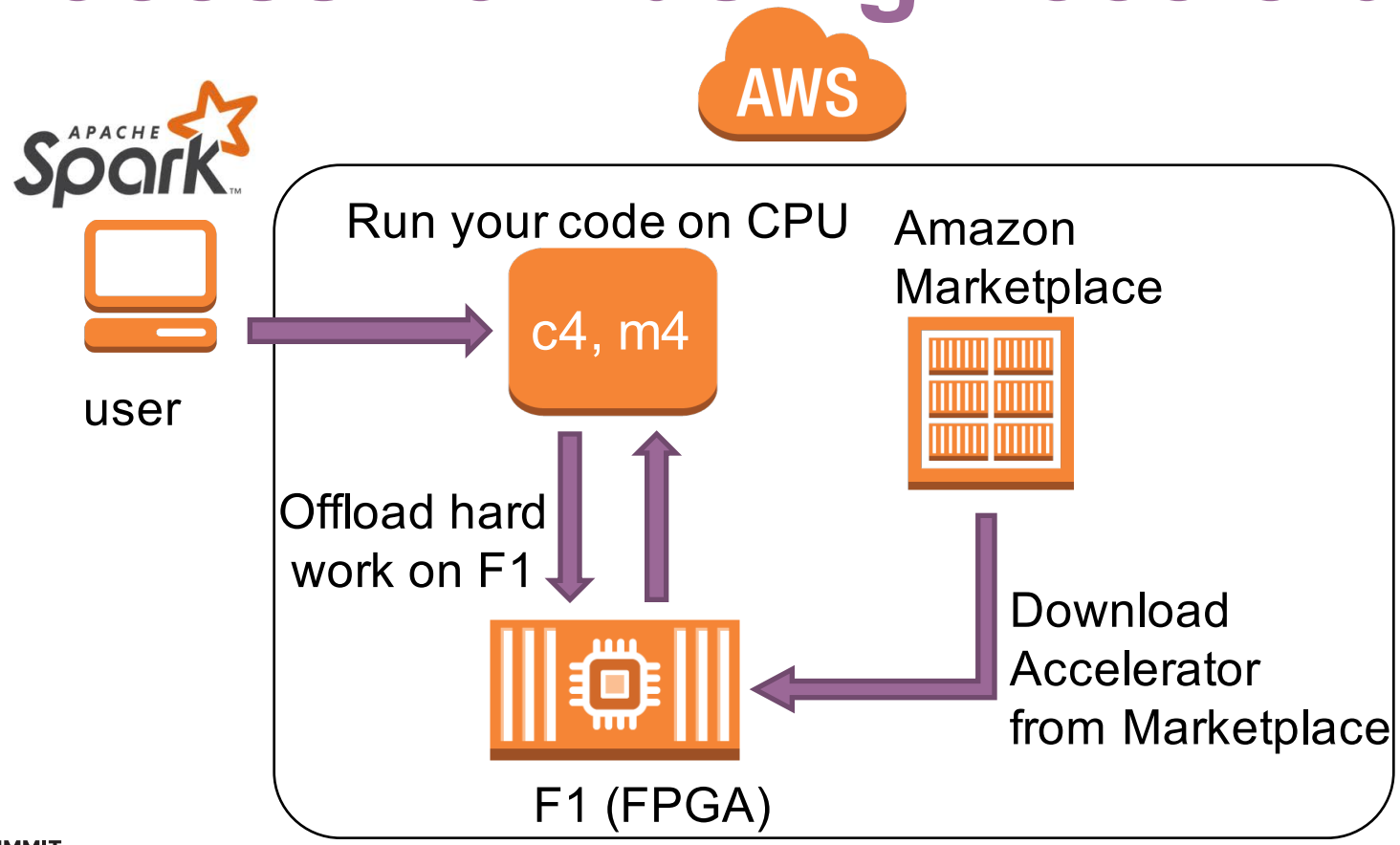




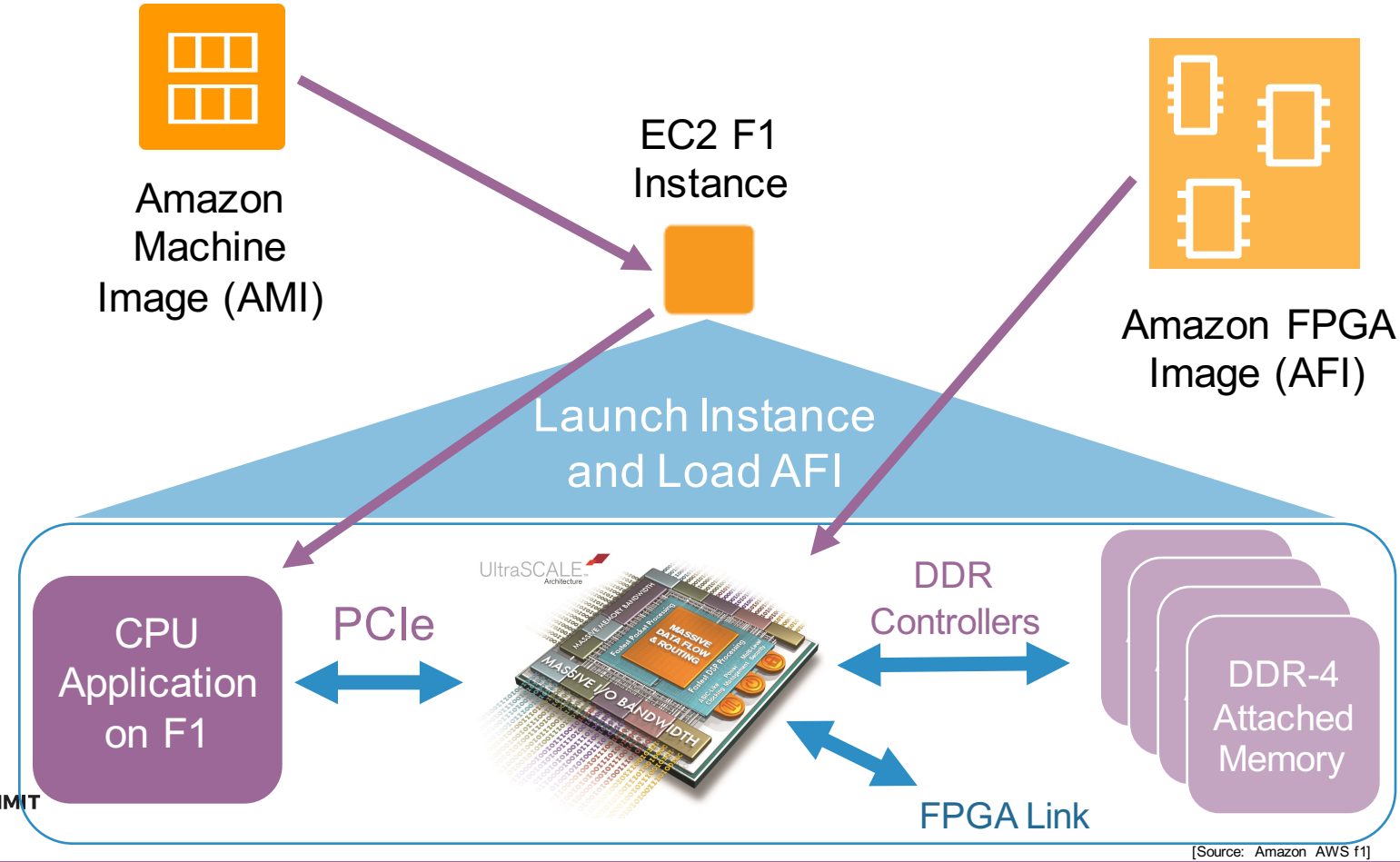
CPU handles the rest

[Source: Amazon AWS f1]

Process flow using Accelerators

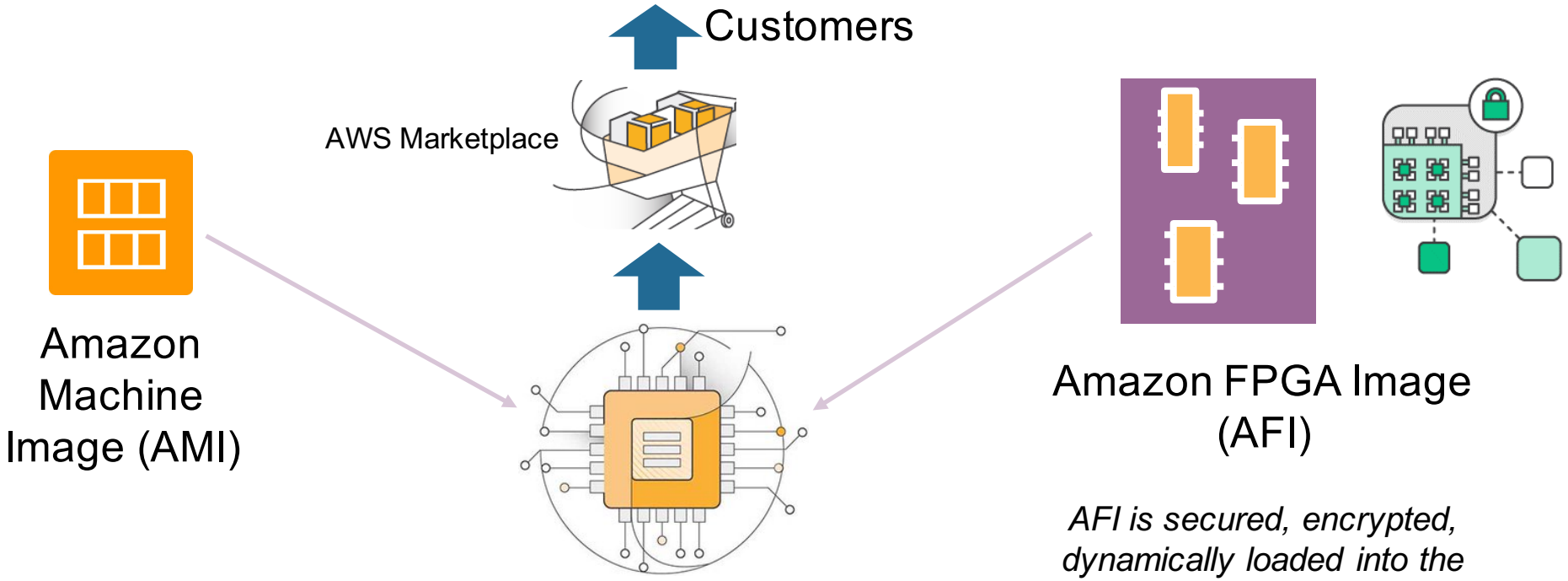


Amazon Resources



[Source: Amazon AWS f1]

FPGAs in Amazon AWS



AFI is secured, encrypted, dynamically loaded into the FPGA - can't be copied or downloaded

[Source: Amazon AWS f1]

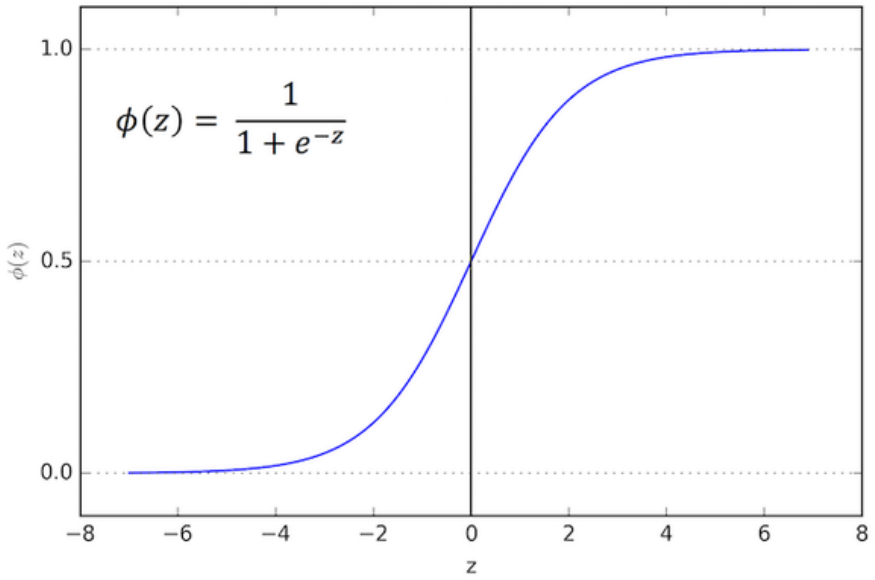
A use case on Logistic regression

LR is used for building predictive models for many complex pattern-matching and classification problems.

It can be applied widely in such diverse areas as

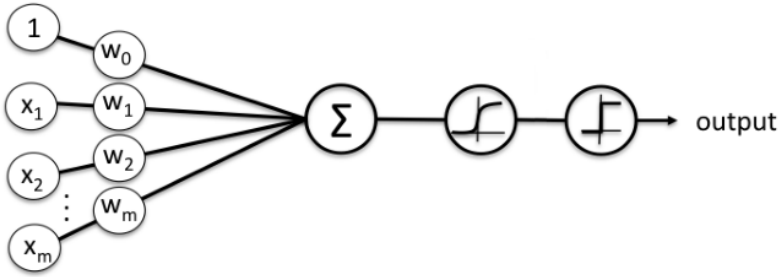
- bioinformatics,
- finance and
- data analytics.

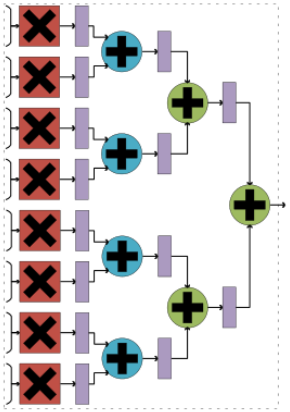
One of the most popular Machine Learning techniques.

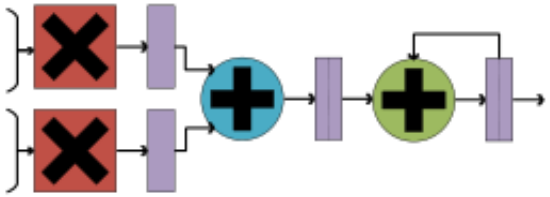


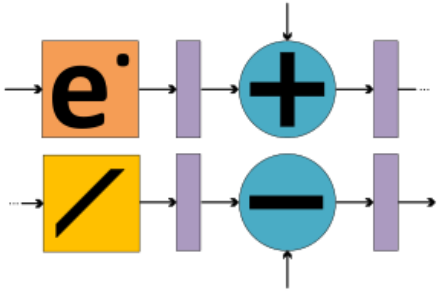
Hardware accelerator for LR

Given a new data point x, the model makes predictions by applying the logistic function $h(z) = \frac{1}{1+e^{-z}}$, where $z = w^T x$.



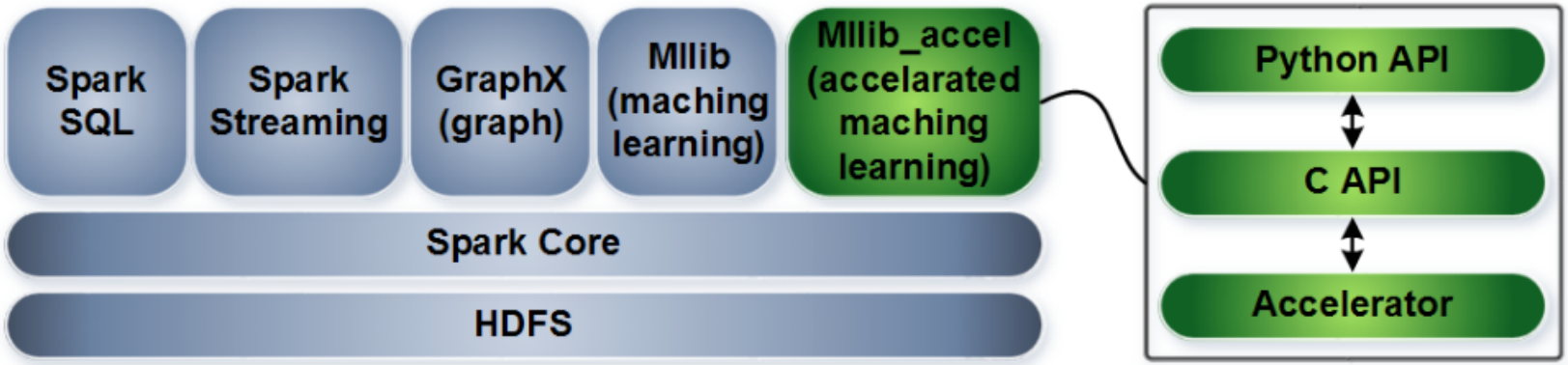






Accelerated Mllib

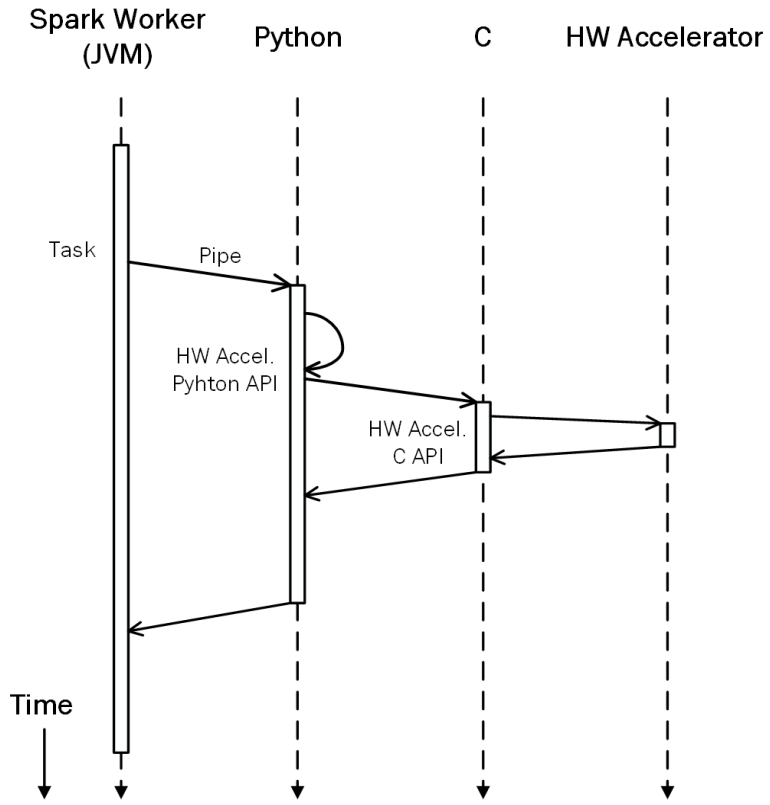
- Extension of Spark libraries



Spark – FPGA interface

- Spark worker
- Python API
- C API

Support for:



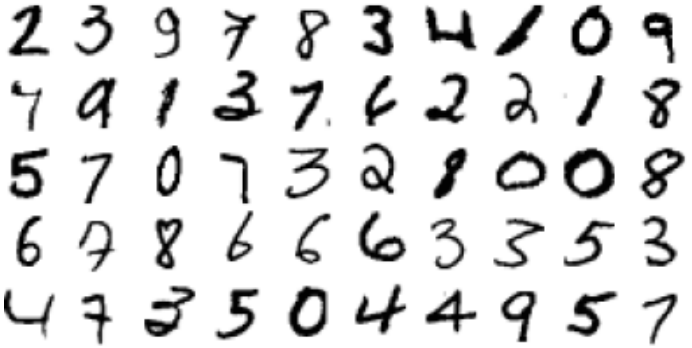
ML acceleration library

- Instantiate a LR model
- Train the model
- (Test the model)

```
from pyspark import SparkContext
from pyspark.mllib.regression import LabeledPoint
#from pyspark.mllib.classification import LogisticRegressionWithLBFGS
from pyspark.mllib_accel.classification import LogisticRegression

def parsePoint(line):
    """
    Parse a line of text into an MLlib LabeledPoint object.
    """

    data = [float(s) for s in line.split(',')]

    return LabeledPoint(data[0], data[1:])

if __name__ == "__main__":

    sc = SparkContext(appName = "Python Logistic Regression")

    trainRDD = sc.textFile(train_file, numPartitions).map(parsePoint)

    #LR = LogisticRegressionWithLBFGS.train(trainRDD, iterations, numClasses)
    LR = LogisticRegression(numClasses, numFeatures)
    LR.train(trainRDD, alpha, iterations)

    sc.stop()
```

Evaluation in cluster

- Evaluation on a Handwritten Digit Recognition Problem
  - Take an image of a handwritten single digit, and determine what that digit is.
- 10 classes, 786 features
- 40k lines train file
- Given a new data point, models will be run, and the class with largest probability will be chosen as the predicted class.



Evaluation

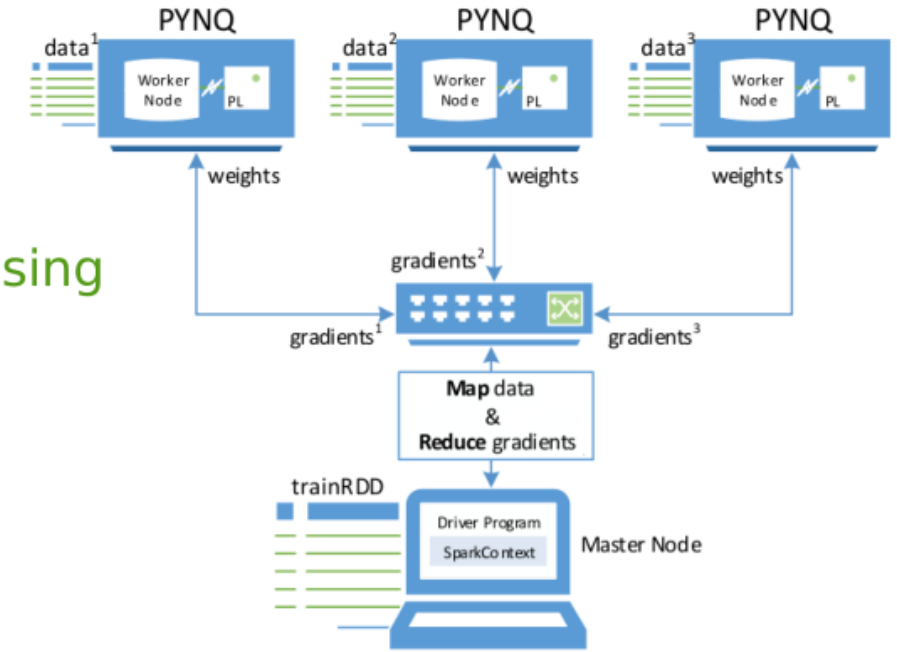
- Evaluation on a Handwritten Digit Recognition Problem
  - Take an image of a handwritten single digit, and determine what that digit is.
- Features of evaluated platforms

Features	Server	PYNQ-Z1
Vendor	Intel	Xilinx
Device	E5-2658	Zynq XC7Z020
Cores(threads)	12(24)	2
Processor	E5-2658	A9
Architecture	64-bit	32-bit
INstruction Set	CISC	RISC
Process	22nm	28nm
Clock Frequency	2.2GHz	667MHz
Level 1 cache	380kB	32kB
Level 2 cache	3072kB	512kB
Level 3 cache	30MB	-
TDP	105W	3.5
Operating System	Ubuntu	Ubuntu

Main features of the evaluated Xeon and Zynq Platforms

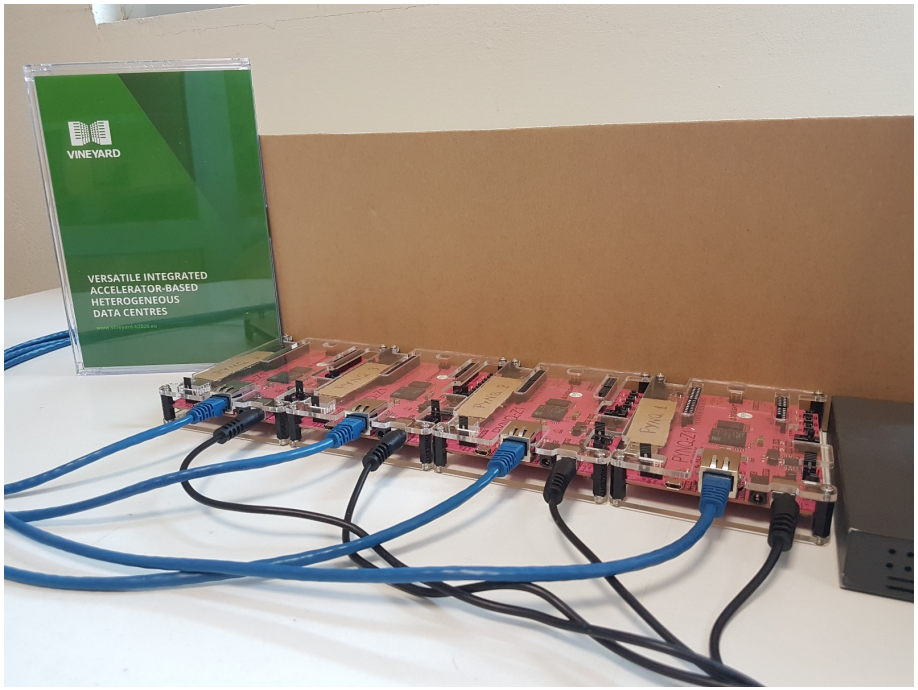
Spark implementation

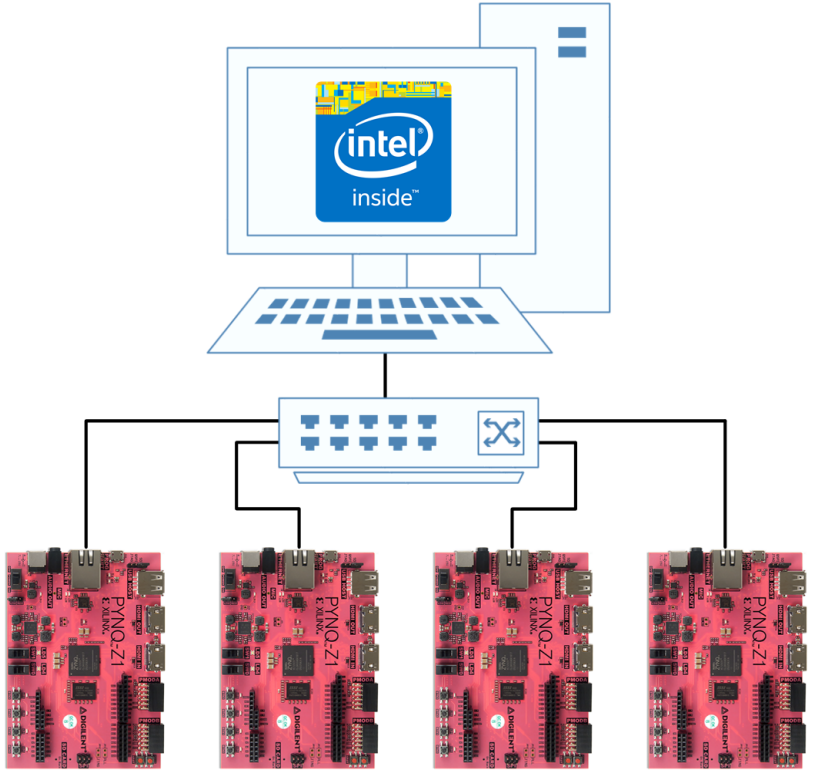
- Run Batch Gradient Descent (BGD) in parallel.
  - Averaging the subgradients over different partitions is performed using one standard spark map-reduce in each iteration.

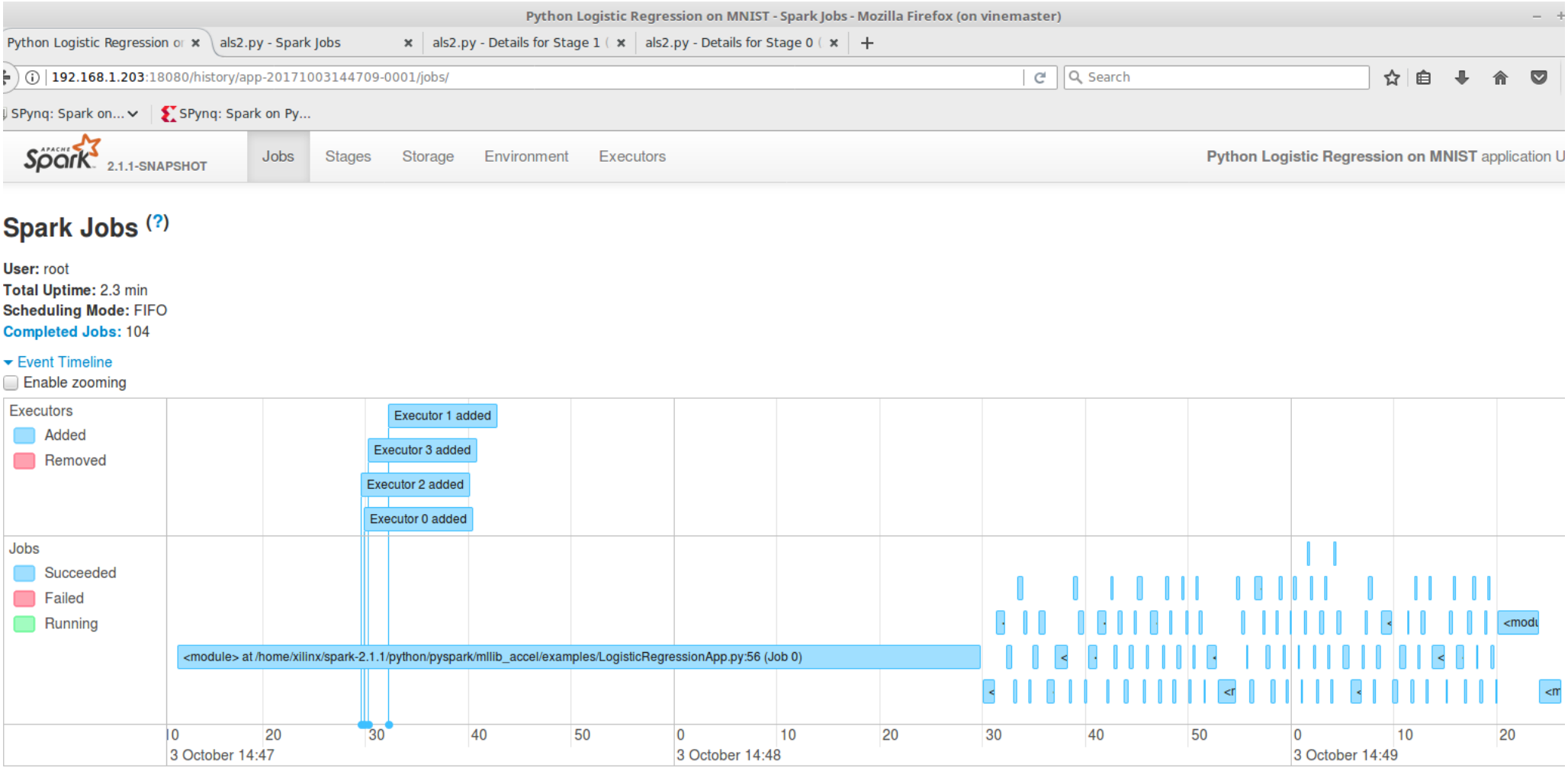



```
if weights is None:
    weights = zeros((numClasses, 1 + numFeatures))

for t in range(0, iterations):
    gradients = trainRDD.map(lambda data: gradients_kernel(data, weights).reduce(lambda a, b: add(a, b))

    weights = subtract(weights, (alpha / numSamples) * gradients)
```

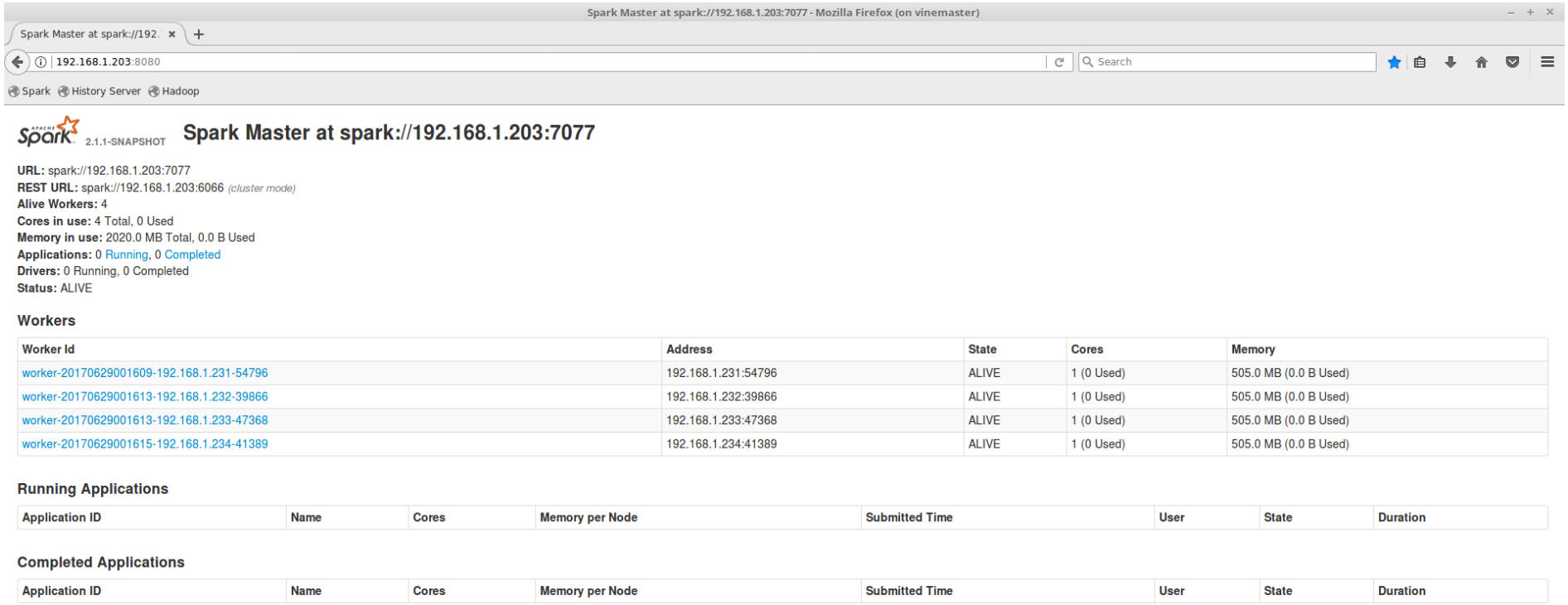
Cluster on FPGA nodes





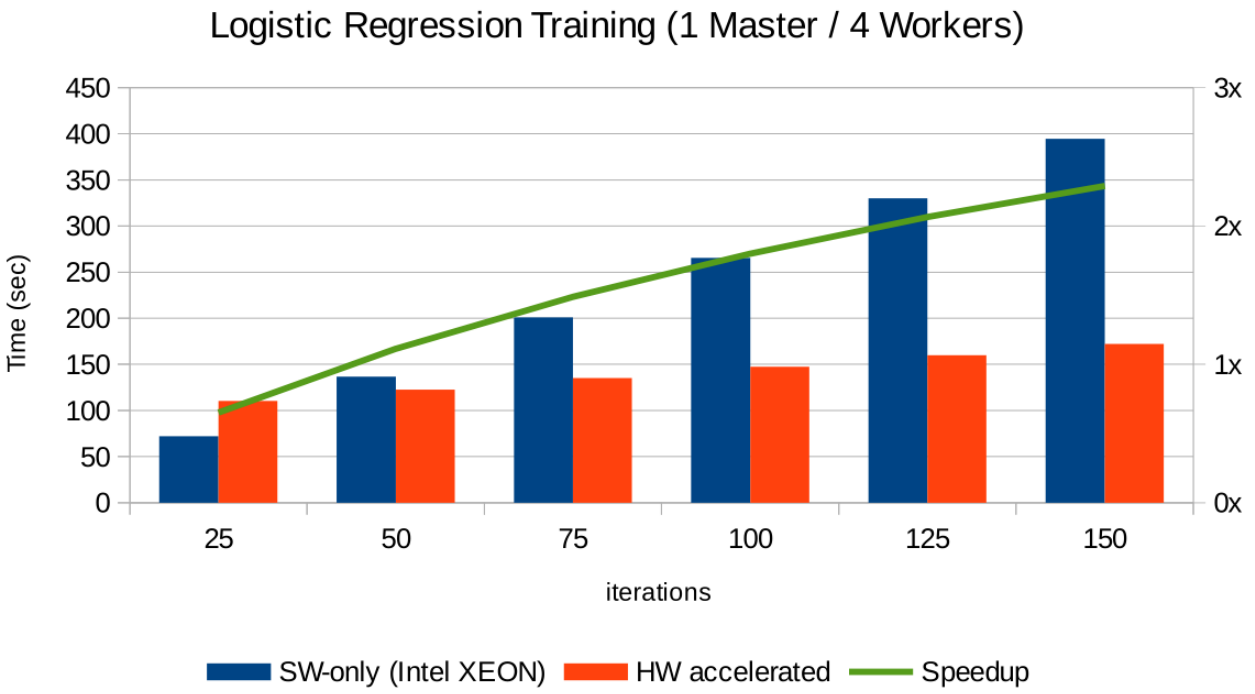

Python Logistic Regression on MNIST - Spark Jobs - Mozilla Firefox (on vinemaster)
Python Logistic Regression or
als2.py - Spark Jobs
als2.py - Details for Stage 1
als2.py - Details for Stage 0
192.168.1.203:18080/history/app-20171003144709-0001/jobs/
Search
SPynq: Spark on...
SPynq: Spark on Py...
Spark 2.1.1-SNAPSHOT
Jobs
Stages
Storage
Environment
Executors
Python Logistic Regression on MNIST application U
Spark Jobs (?)
User: root
Total Uptime: 2.3 min
Scheduling Mode: FIFO
Completed Jobs: 104
Event Timeline
Enable zooming
Executors
Added
Removed
Executor 1 added
Executor 3 added
Executor 2 added
Executor 0 added
Jobs
Succeeded
Failed
Running
<module> at /home/xilinx/spark-2.1.1/python/pyspark/mllib_accel/examples/LogisticRegressionApp.py:56 (Job 0)
10
20
30
40
50
0
10
20
30
40
50
0
10
20
3 October 14:47
3 October 14:48
3 October 14:49

Spark running on FPGA-based cluster



Speedup

- Xeon vs Xeon + small FPGA cluster



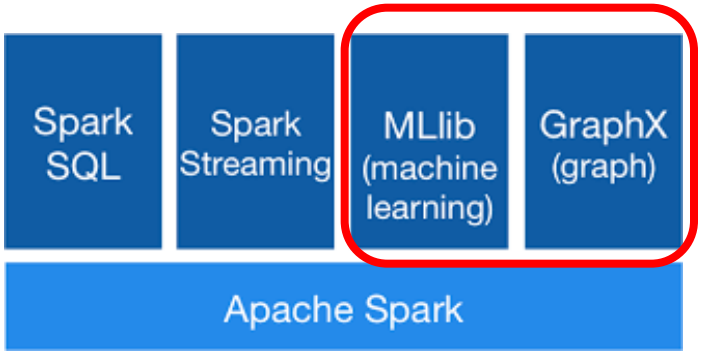
Comparison

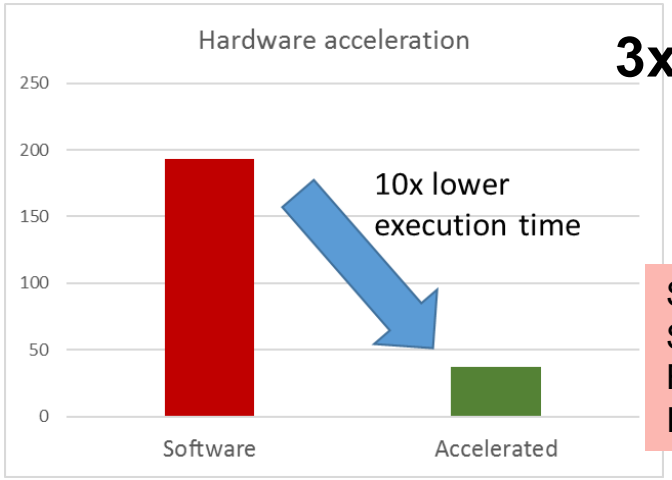
SW (Intel Xeon)

SW+HW
(Accel on FPGA SoCs)

Spark Accelerators

- Develop hardware components as IP cores for widely used applications
  - Spark
    - Logistic regression
    - Recommendation (ALS)
    - K-means
    - Linear regression
    - PageRank
    - Graph computing



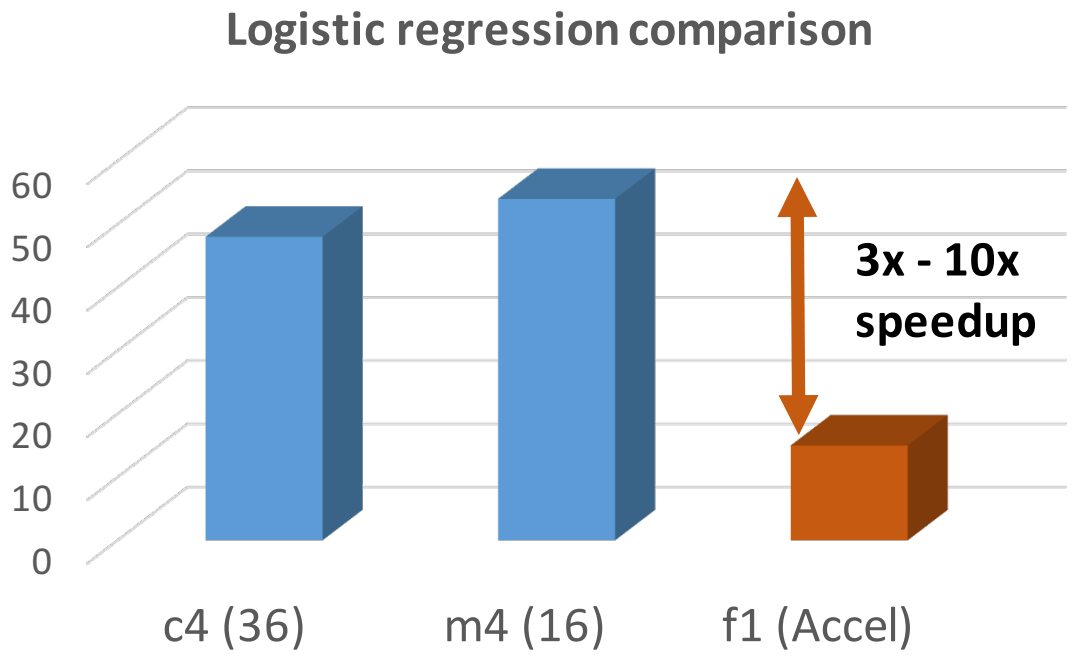


3x – 50x speedup

Source: C. Kachris et al., Spark acceleration using FPGAs, best paper award, IEEE MOCAST 2017

Comparison with Amazon AWS EC

- c4 (36 cores)
- m4 (16 cores)
- f1 with our Accelerator



Mllib library to be covered

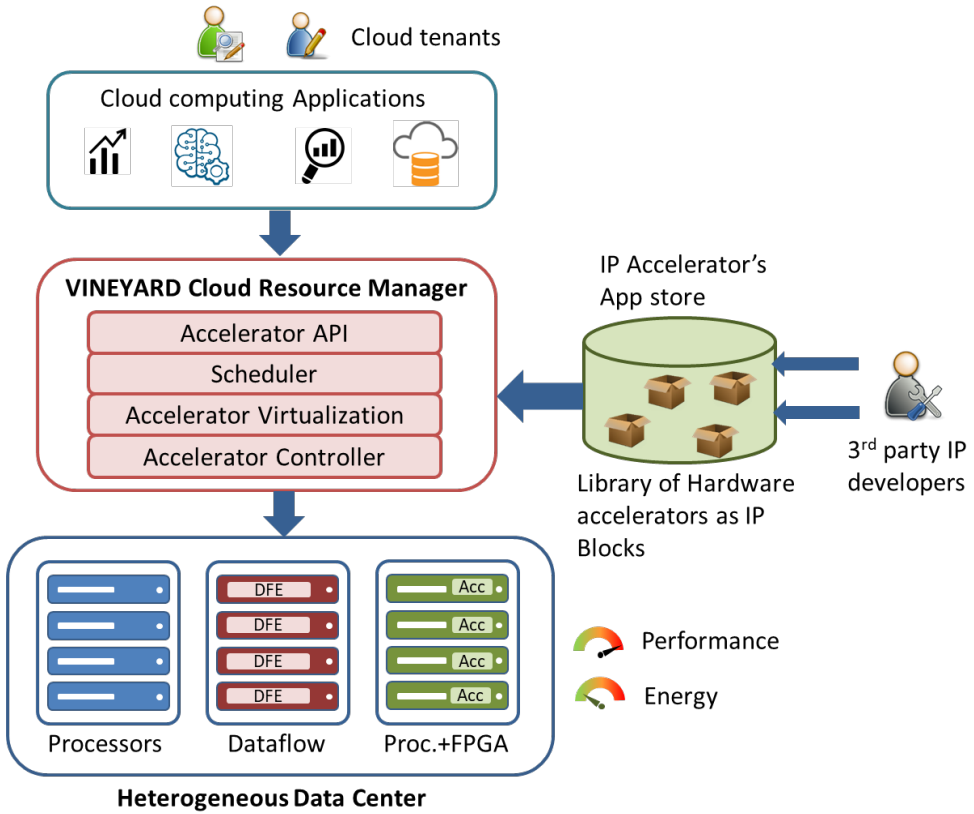
MLlib contains many algorithms and utilities.

- **Classification**: **logistic regression**, naive Bayes,...
- **Regression**: generalized linear regression, survival regression,...
- **Recommendation**: **alternating least squares (ALS)**
- **Clustering**: **K-means**, Gaussian mixtures (GMMs),...
- **Topic modeling**: latent Dirichlet allocation (LDA)
- Frequent itemsets, association rules, and sequential pattern mining

VINEYARD framework

- Utilize FPGA resources seamlessly

More info :
vineyardh2020.eu



29

VINEYARD

PARTNERS

(Coordinator)
www.microlab.ntua.gr

www.maxeler.com

www.bull.com

www.qub.ac.uk

www.ics.forth.gr

www.hartree.stfc.ac.uk
/hartree

www.neurasmus.com

www.neurocom.eu

www.helex.gr

www.leanxcale.com

www.loba.pt

Overview - InAccel

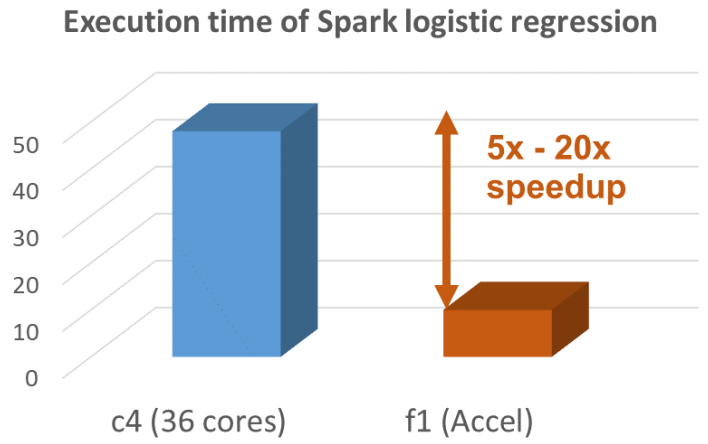
InAccel provides high performance accelerators for your application based on novel hardware reconfigurable engines as IP blocks.
The hardware accelerators can be used to improve the performance of your applications both in terms of throughput and execution time (latency).

Spark integration
The hardware accelerators of **InAccel** are compatible with the Apache Spark framework and allow faster execution of Spark applications based on MLlib (machine learning) and GraphX libraries.

Overview - InAccel

Amazon compatibility
InAccel develops hardware accelerators that are compatible with the Amazon AWS F1 platform. Use the hardware accelerators as an IP to speedup your application, seamlessly, without the purchase of any hardware.

No need for code changes
InAccel provides all the required APIs for the seamless integration of the accelerators without any modifications on your original code.



Thank you

Questions?

kachris@microlab.ntua.gr