inaccel

Leading Application Acceleration

helps companies speedup their applications

by providing ready-to-use accelerators-as-a-service in the cloud or on-prem

15x Speedup

4x Lower TCO

Zero code changes

Applications and Platforms

- Applications

Machine learning Financial Analytics Genomics

- Platforms

Alibaba Cloud, aws, Baidu, HUAWEI, NIMBIX, Tencent Cloud

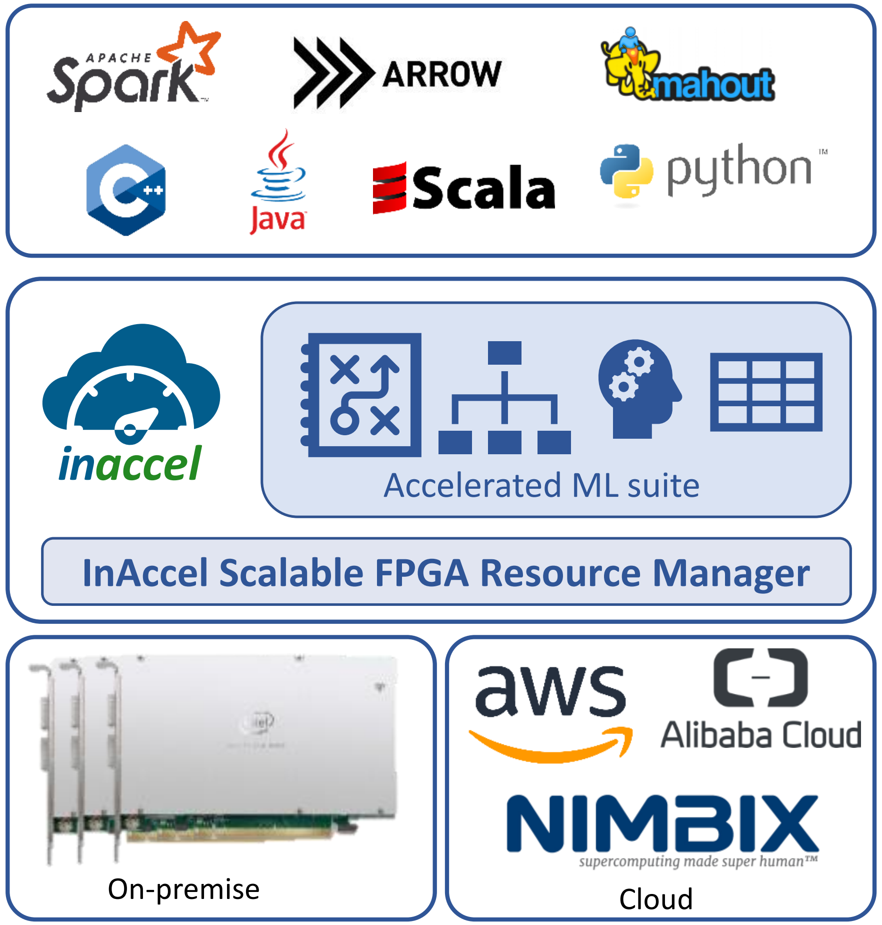
- Partnerships

aws partner network, aws marketplace, intel FPGA Design Solutions Network, Alibaba Cloud

Open FPGA Data Science

> We help Data science/engineers run up to 15x faster their applications without changing their code

Integration solution for Application Acceleration



Higher Performance

Up to 16x Speedup compared to highly optimized libraries

Lower Cost

Up to 4x lower TCO

Zero-code changes

Seamless integration to widely used frameworks

Easy deployment

Docker-based container for seamless integration

On-prem or on cloud

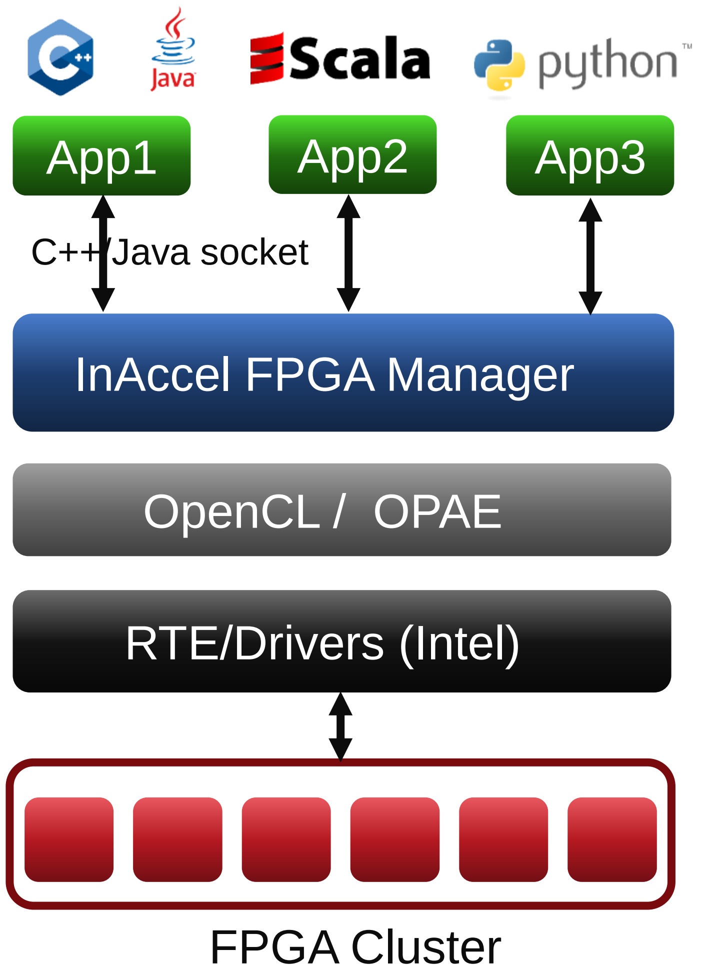
Available on cloud and on-prem

InAccel Coral FPGA Resource Manager

Worlds' best FPGA Orchestrator:

Program against your FPGAs like it's a single pool of accelerators

- > Coral abstracts FPGA resources (device, memory), enabling fault-tolerant heterogeneous distributed systems to easily be built and run effectively.



Current Framework for FPGAs on the cloud

Limitations without the InAccel Coral Manager

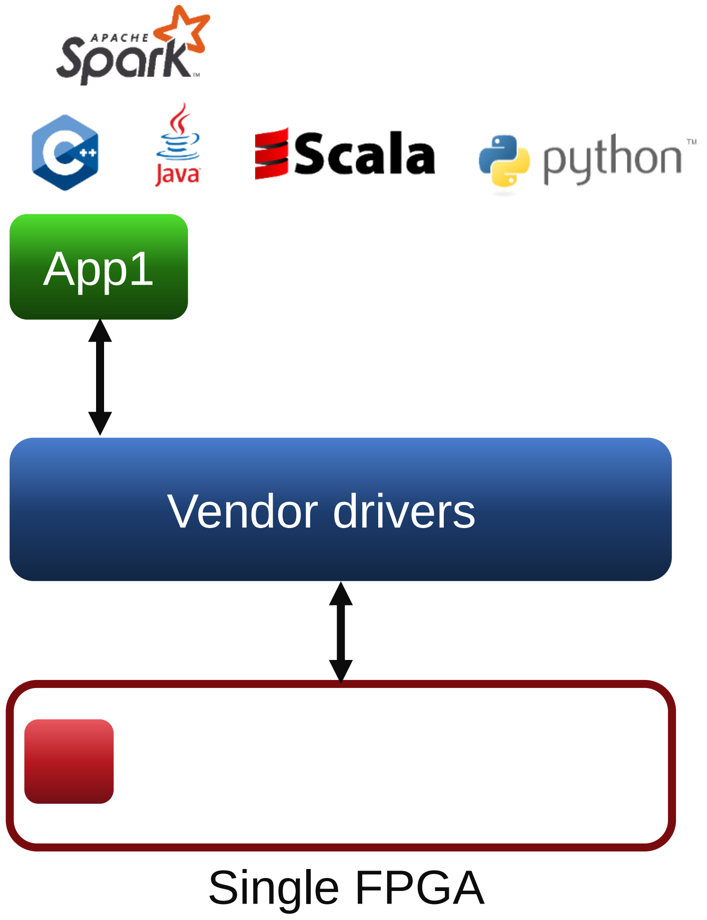
> Currently only **one application** can talk to each FPGA accelerator

> Every application can talk to a **single** FPGA.

> Complex device sharing

- From multiple threads/processes
- Even from the same thread

> Explicit allocation of the resources (memory/compute units)



InAccel's Coral FPGA Manager

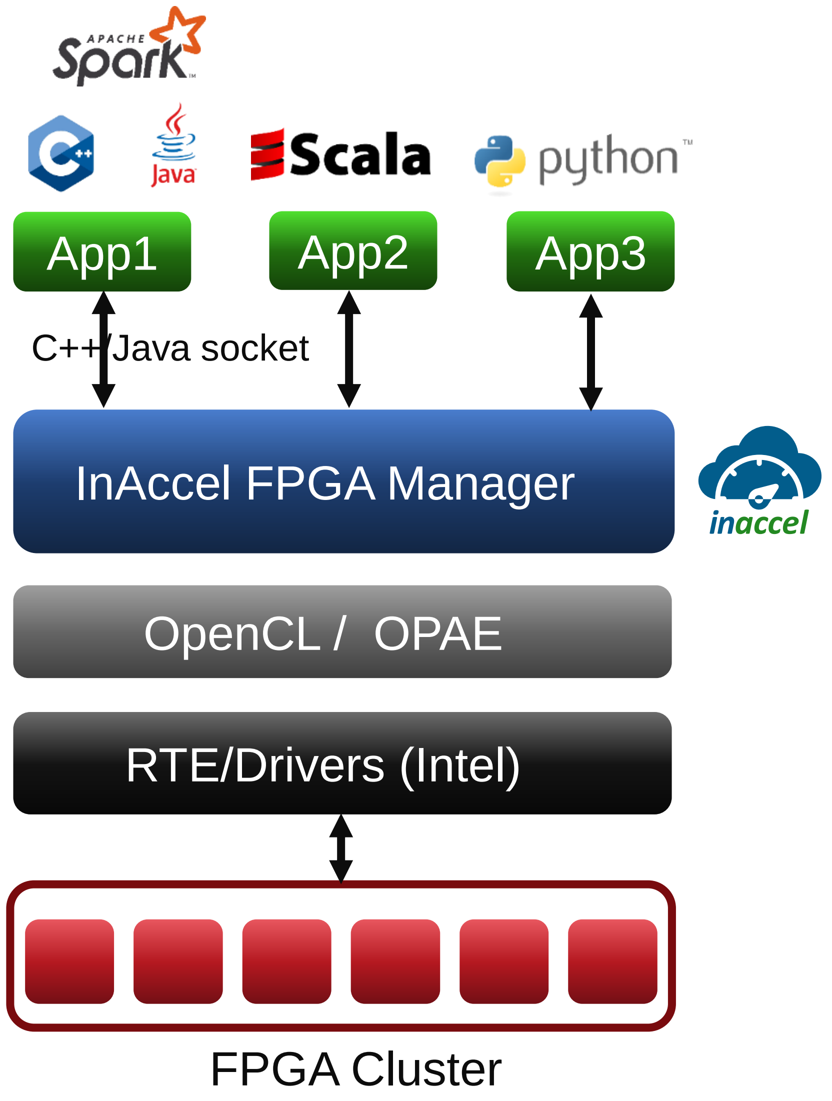
Acceleration abstraction layer to virtualize, manage and monitor the FPGA resources

> **Management**
- Automatic device (re-)configurations and efficient memory transfers

> **Fault-tolerance**
- Highly-available service on top of a cluster of FPGAs, each of which may be prone to failures

> **Scalability**
- Automatic scale-up from single devices (e.g. f1.x2) to multi-FPGA systems (e.g. f1.x4, f1.x16)

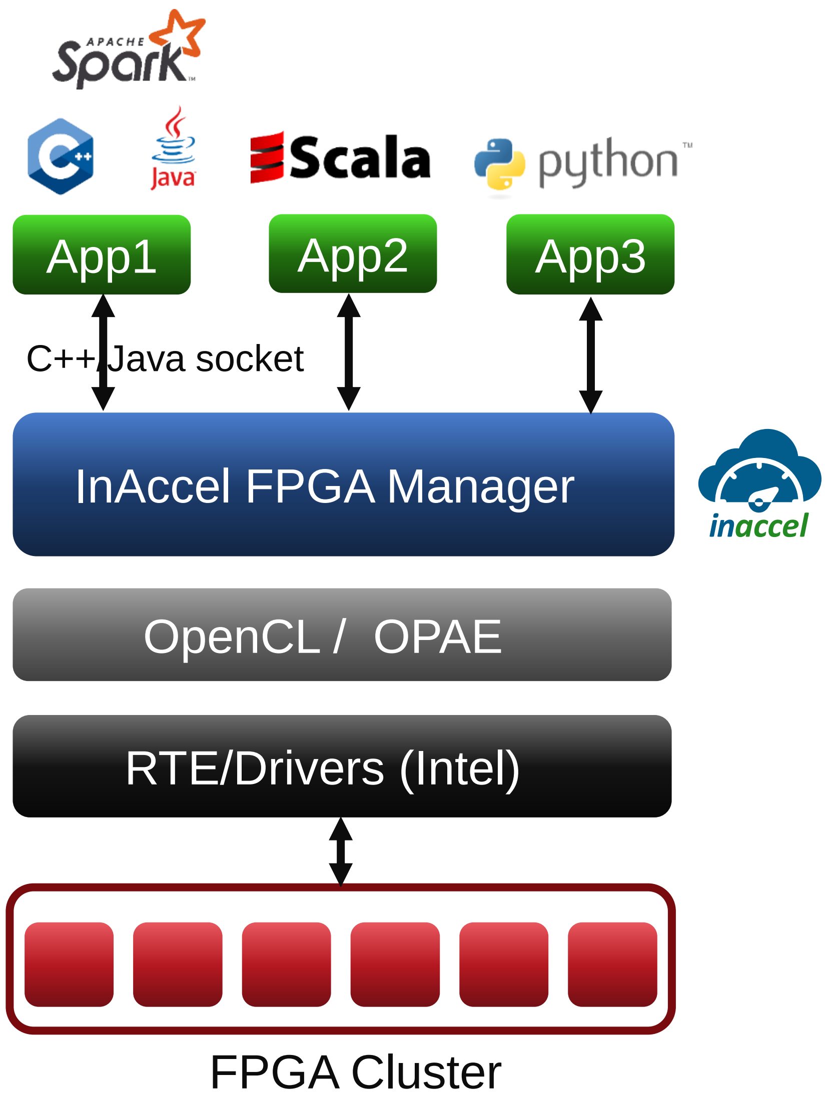
Documentation: https://docs.inaccel.com/latest/



FPGA Manager features

Ease of Use

> **Write applications quickly in C/C++, Java, Scala and Python.**
InAccel offers all the required high-level functions that make it easy to build and accelerate parallel apps. No need to modify your application to use an unfamiliar parallel programming language (like OpenCL)



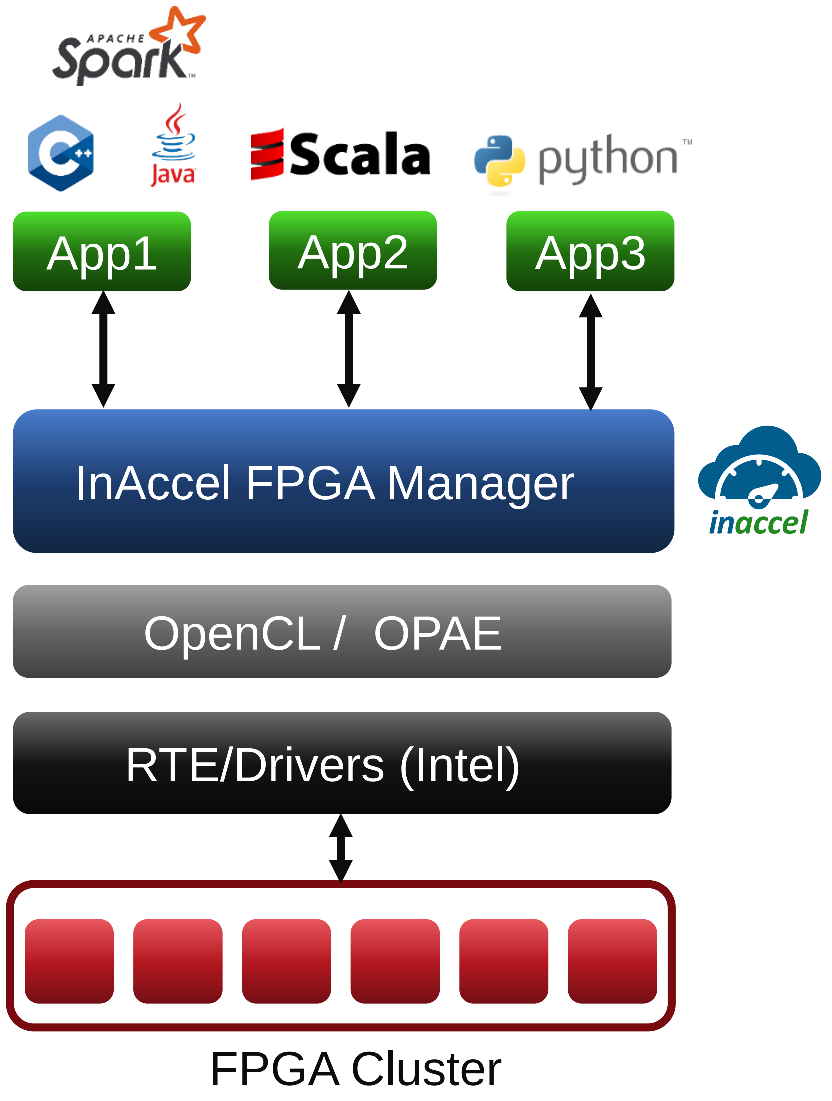
FPGA Manager features

Runs Anywhere

> Runs on any FPGA platform (**Intel**), giving you the freedom to take full advantage of on-premises, or public Cloud (**Alibaba, Nimbix, etc.**) infrastructure.

On-premise

Alibaba Cloud



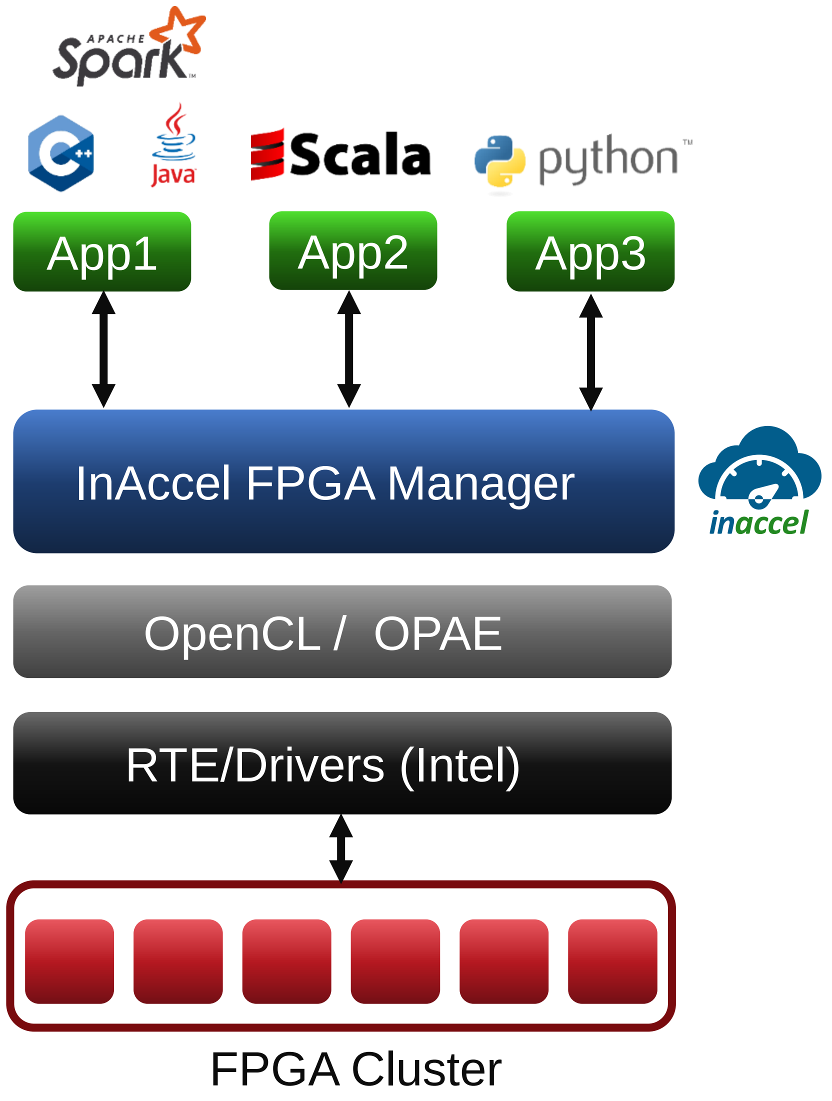
FPGA Manager features

Resource Management

> **Automatic resource configuration and task scheduling across entire FPGA clusters in private datacenters or public cloud environments.**
Coral examines the state of the FPGAs and implements load-balancing policies across them, efficiently taking care of all the required device configurations and memory transfers.

Privacy / Isolation

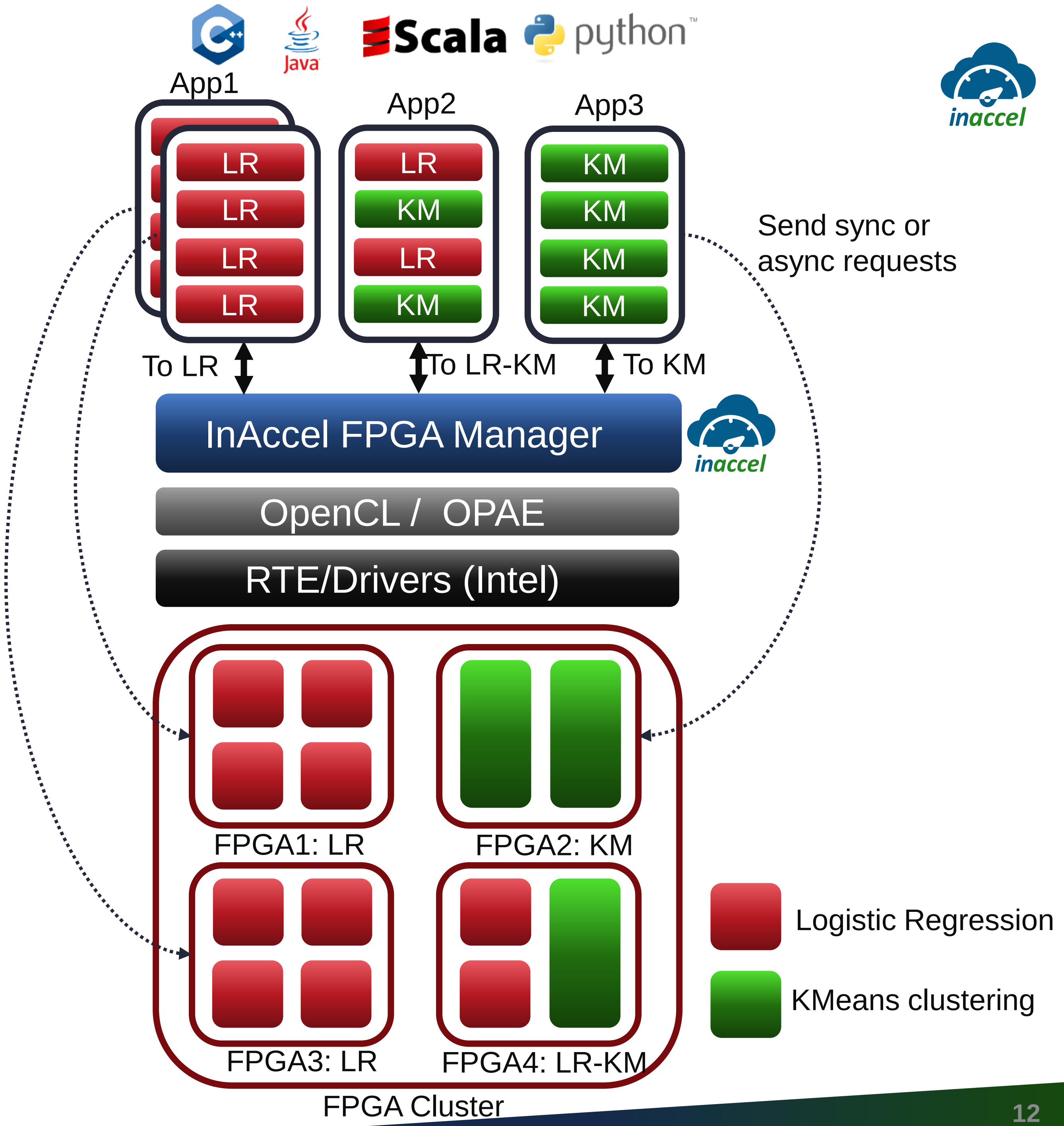
> **Coral allows the secure sharing of the hardware resources among different users and multiple processes or threads.**
First class isolation support for accelerator cores and FPGA memory.



FPGA Manager use case

Abstracts away FPGA cluster

- App1: 2 threads that use LR
  - Just send LR requests
    - The FPGA manager send the requests to FPGAs programmed with LR (no need to specify which FPGA/kernel)
- App2: Uses both LR and KM
  - Just send LR-KM requests
    - The FPGA manager send the requests to FPGAs programmed with LR and KM kernels
- App3: 1 thread uses KM
  - Just send Km requests
    - The FPGA manager sends the requested to FPGAs programmed with KM kernels



Pricing model

FPGA Resource manager

> **Pricing model per node (server)**

> **Each node can have 1 to 8 FPGAs**

FPGA Resource manager	20 nodes or less	More than 20 nodes
Monthly	$150/node	$100/node
Yearly	$1,500/node	$1000/node

ML Accelerators

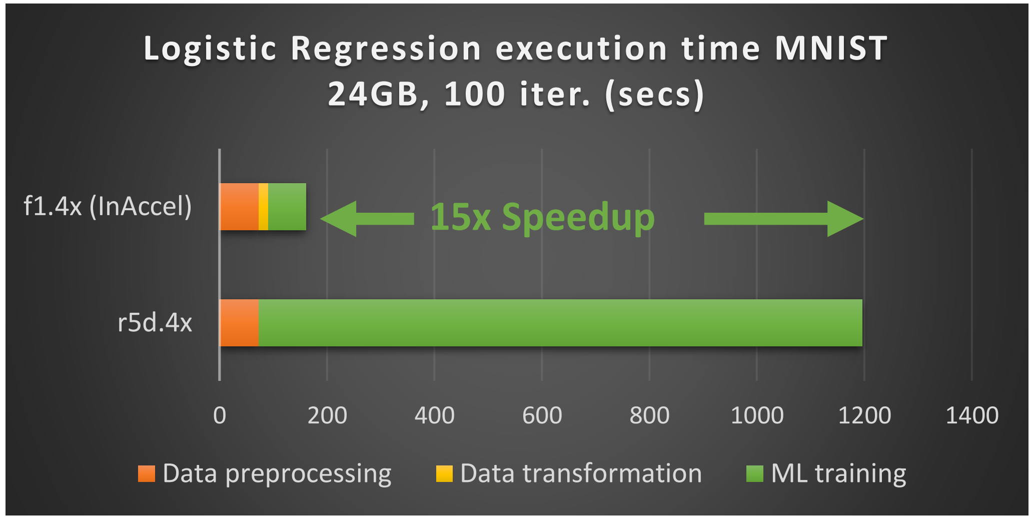
> **Pricing model per node (server)**

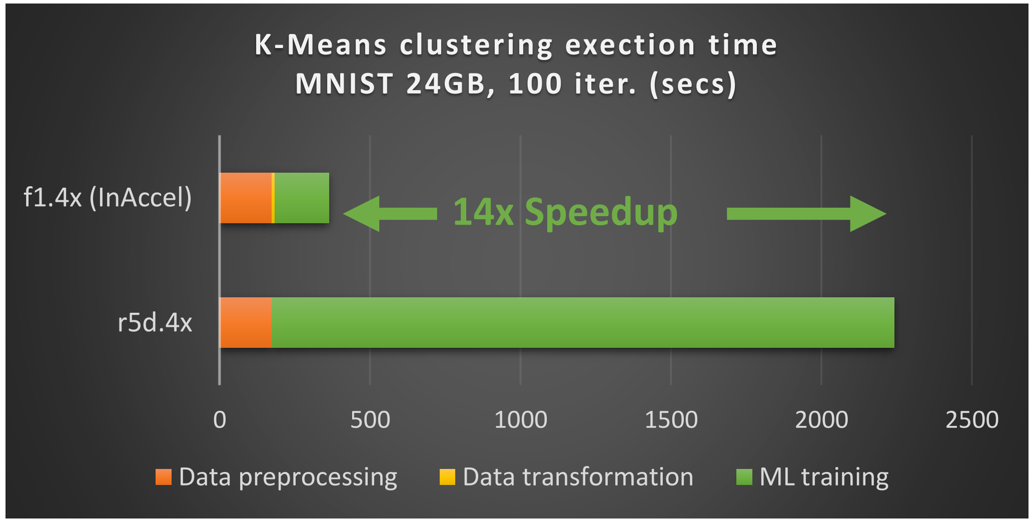
ML Accelerator	20 nodes or less	More than 20 nodes
Yearly	$10,000/node	$8,000/node

Performance evaluation on Machine Learning

- **Up to 15x speedup for LR ML (7.5x overall)**
- **Up to 14x speedup for Kmeans ML (6.2x overall)**

- **F1.4x**
  - 16 cores + 2 FPGAs (InAccel)
- **R5d.4x**
  - 16 cores

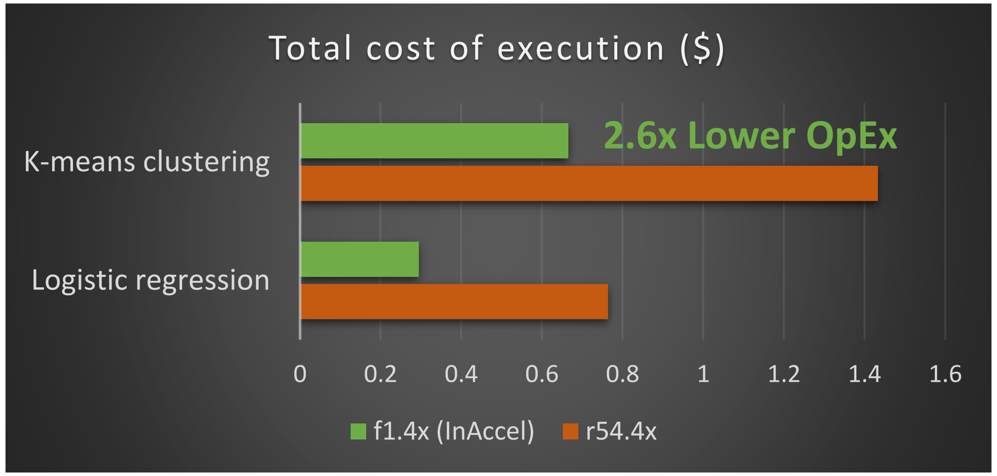




Cost reduction

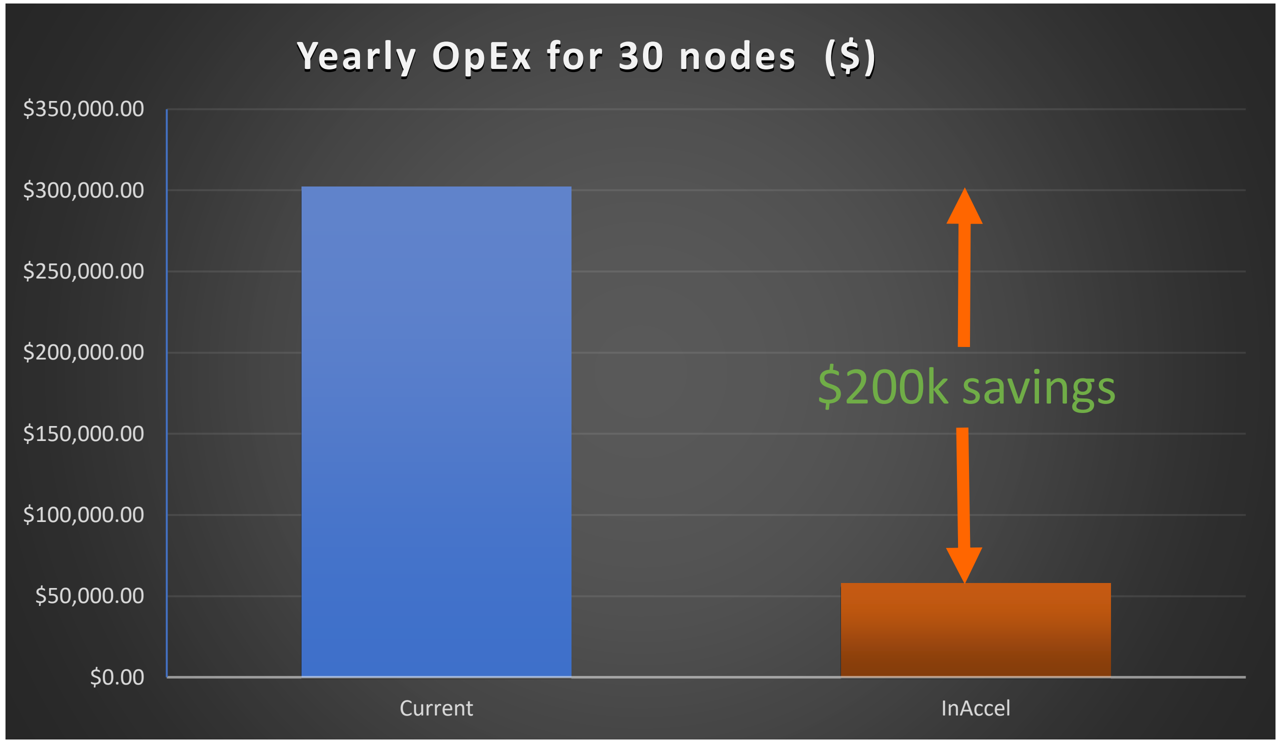
- **Up to 2.6x lower cost and 15x speedup**

- **F1.4x ($3.3)**
  - 16 cores + 2 FPGAs (InAccel)
- **R5d.4x ($1.15)**
  - 16 cores



Cost reduction

- For a cluster of 60 nodes on aws (24/7 – 1 year):
  - R5d.4x ($1.15/hour) : $302,220
- For a cluster of only 2 f1 nodes on aws (24/7 – 1 year) providing the same performance:
  - F1.4x ($3.3/hour) : $57,816
    + 20k license of ML accelerators
  - **Over $200k savings per year**
- **16x smaller footprint**



Personnel cost savings

> Reduction on the complexity to program the FPGAs

> Over 4x lower LoC (lines of code)

> Significant savings in time to run and debug the FPGAs

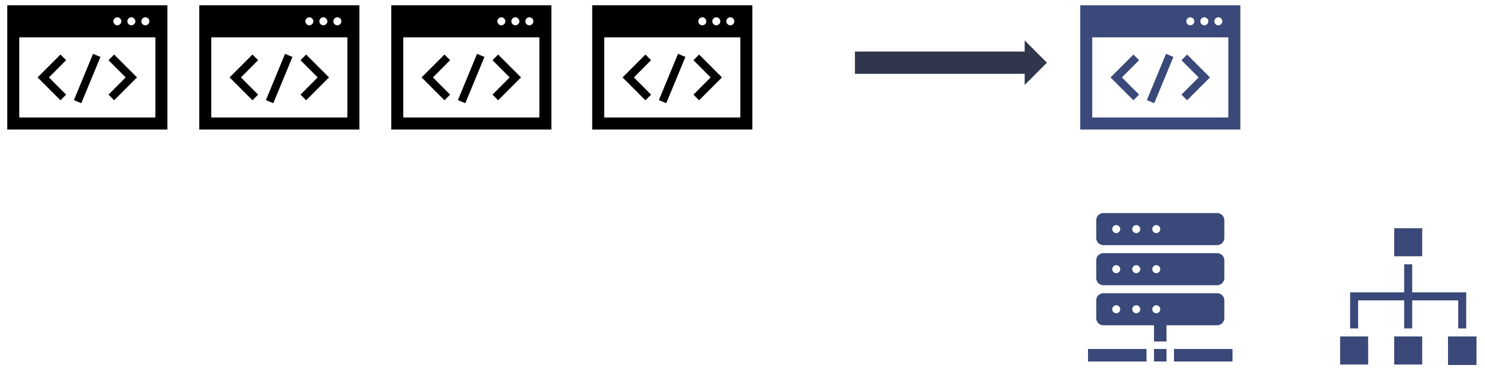
Integration with Intel FinLib

> **Original Finlib**

- Based on LACE framework
  - Methods/classes for each accelerator
- Distribution on the FPGAs through partitioning

> **InAccel Finlib with Coral Resource Manager**

- **Universal** approach for every accelerator
  - No need for specific methods for each accelerator
- Virtualization of the resources allows **multiple applications**/threads to have access to the resources
- Same performance, Zero overhead
- Closer to SW methodology, simple function invoking



Example on Black&Schole or MonteCarlo

Original SW-only code

```
// Reference
std::vector<bs_opt_param_inst> inRef;
std::vector<bs_opt_result_inst> outRef;

inRef.assign(in.begin(), in.end());
outRef.assign(out.begin(), out.end());

for (size_t i = 0; i < inRef.size(); i++) {
    outRef[i].premium = BSRef::BSOpt(
        inRef[i].cp,
        inRef[i].fwdPrice,
        inRef[i].strike,
        inRef[i].vol,
        inRef[i].r,
        inRef[i].t
    );
}
```

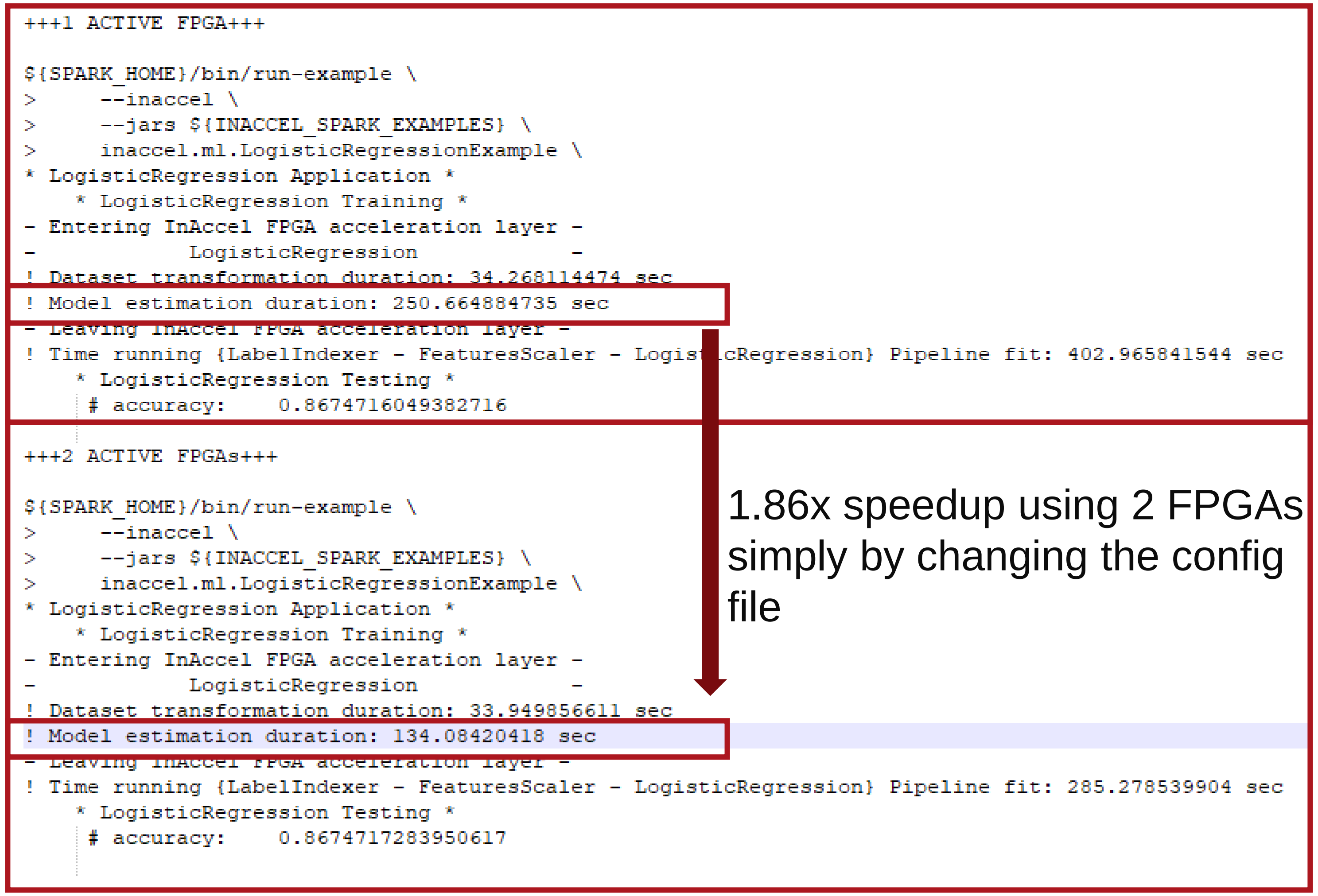
With Coral

```
int NUM_OPTIONS = 64 * (1 << 15);

inaccel::vector<bs_opt_param_inst> in(NUM_OPTIONS);
inaccel::vector<bs_opt_result_inst> out(in.size());

testGen(in, out);

inaccel::Request request
{"com.intel.finlib.tutorials.BlackScholes"};

request.Arg(in).Arg(out).Arg(NUM_OPTIONS >> 15);

inaccel::Coral::Submit(request);
```

Function to send request to InAccel FPGA manager.
Automatically sends the data to the FPGA cluster

3 Lines of Code
Software much closer to the original software-only

Example on scaling to 2 FPGA using the resource manager for logistic regression



```
+++1 ACTIVE FPGA+++

${SPARK_HOME}/bin/run-example \
>     --inaccel \
>     --jars ${INACCEL_SPARK_EXAMPLES} \
>     inaccel.ml.LogisticRegressionExample \
* LogisticRegression Application *
    * LogisticRegression Training *
- Entering InAccel FPGA acceleration layer -
-              LogisticRegression          -
! Dataset transformation duration: 34.268114474 sec
! Model estimation duration: 250.664884735 sec
- Leaving InAccel FPGA acceleration layer -
! Time running {LabelIndexer - FeaturesScaler - LogisticRegression} Pipeline fit: 402.965841544 sec
    * LogisticRegression Testing *
     # accuracy:     0.8674716049382716
```

```
+++2 ACTIVE FPGAs+++

${SPARK_HOME}/bin/run-example \
>     --inaccel \
>     --jars ${INACCEL_SPARK_EXAMPLES} \
>     inaccel.ml.LogisticRegressionExample \
* LogisticRegression Application *
    * LogisticRegression Training *
- Entering InAccel FPGA acceleration layer -
-              LogisticRegression          -
! Dataset transformation duration: 33.949856611 sec
! Model estimation duration: 134.08420418 sec
- Leaving InAccel FPGA acceleration layer -
! Time running {LabelIndexer - FeaturesScaler - LogisticRegression} Pipeline fit: 285.278539904 sec
    * LogisticRegression Testing *
     # accuracy:     0.8674717283950617
```

1.86x speedup using 2 FPGAs simply by changing the config file

```
inaccel start --fpga=intel:0,intel:1
```

You specify how many FPGAs you want to use

or

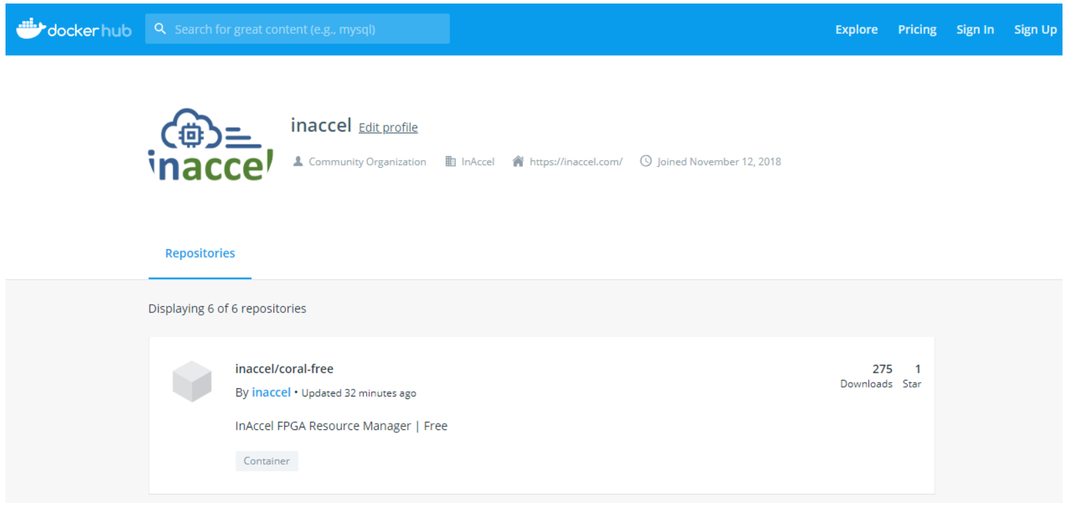
```
inaccel start --fpga=all
```

FPGA Manager deployment

Easy to Deploy

> Launch a container with InAccel's **Docker** image or even deploy it as a daemonset on a **Kubernetes** cluster and enjoy acceleration services at the drop of a hat.

> https://hub.docker.com/u/inaccel/



inaccel Edit profile

Community Organization · InAccel · https://inaccel.com/ · Joined November 12, 2018

Repositories

Displaying 6 of 6 repositories

inaccel/coral-free

By inaccel • Updated 32 minutes ago

InAccel FPGA Resource Manager | Free

Container

275 Downloads · 1 Star

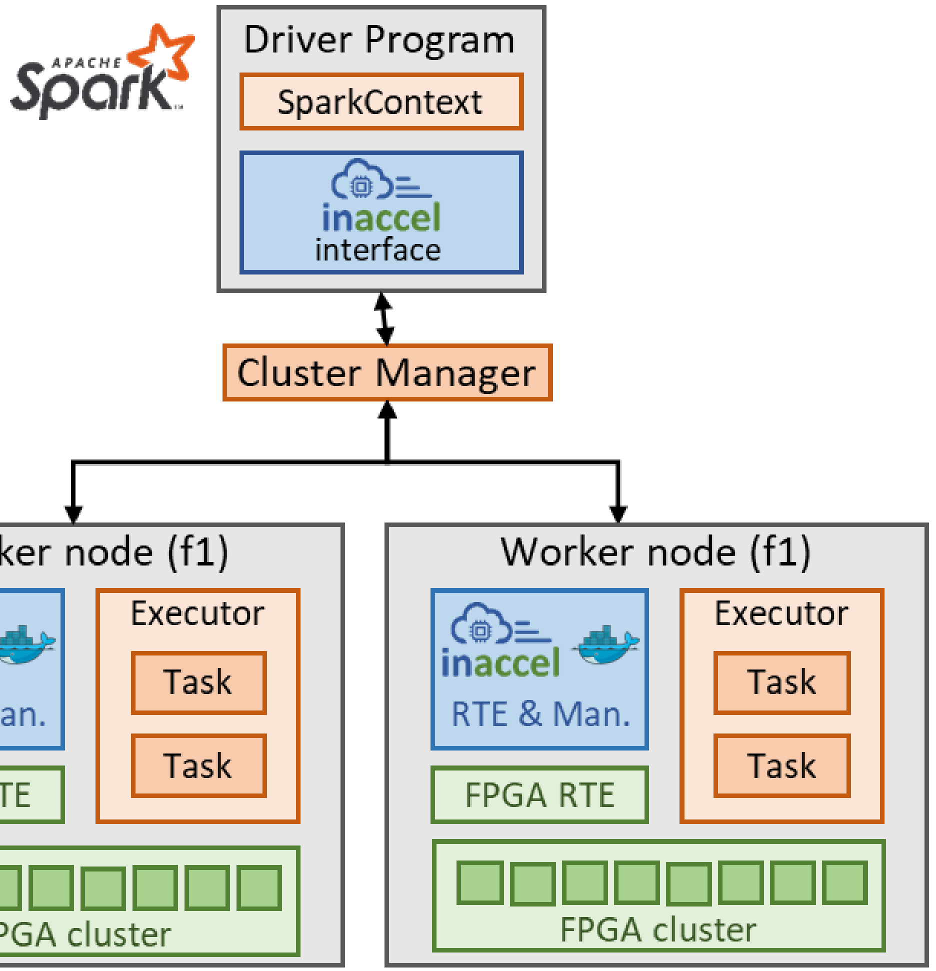
FPGA Manager

- Easy deployment
- Easy scalability
- Easy integration

InAccel's Coral manager integrated with Spark

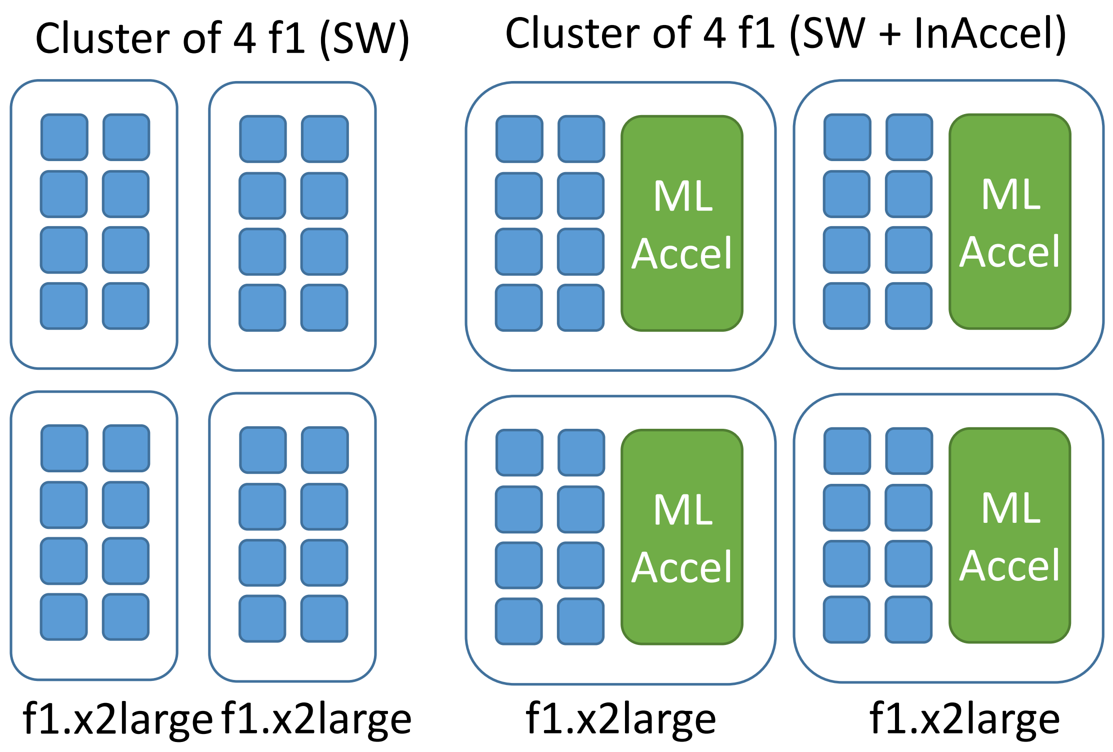
> **Integrated solution that allows**

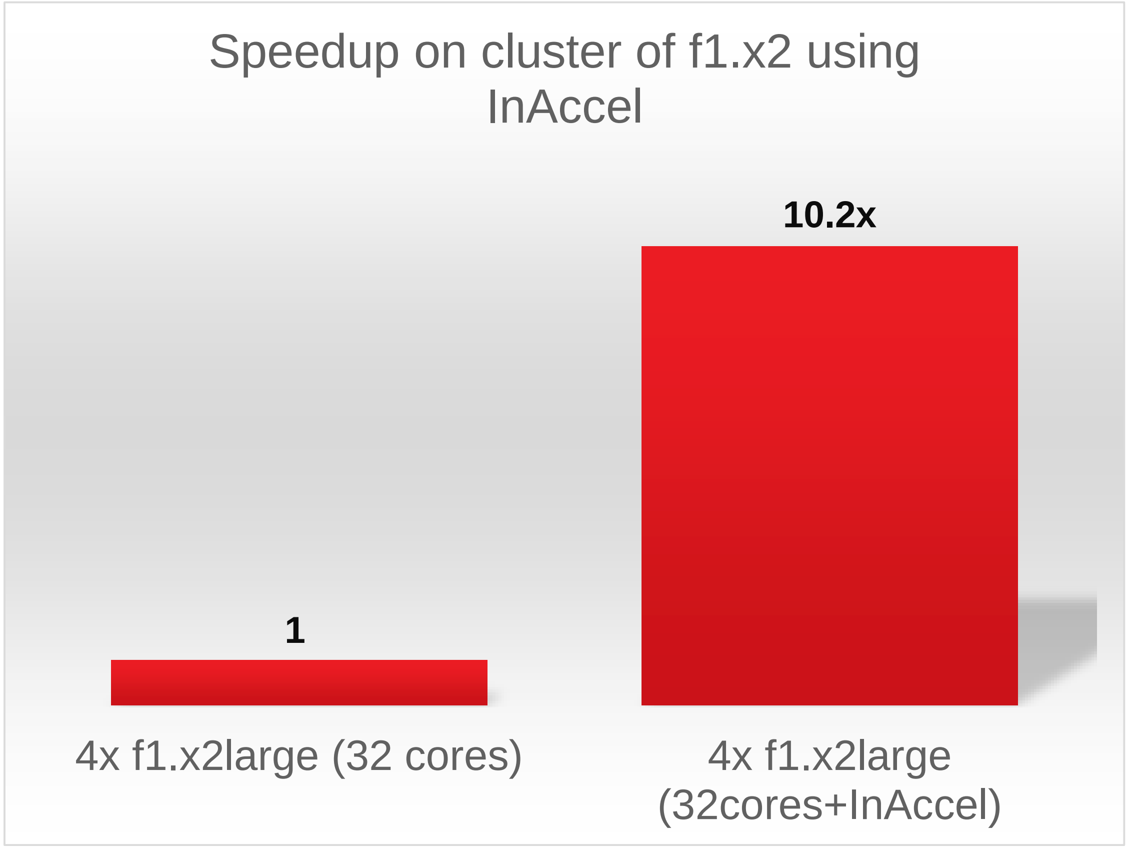
- Scale Up (1, 2, or 8 FPGAs per node)
- Scale Out to multiple nodes (using Spark API)
- Seamless integration
- Docker-based deployment



Speedup comparison - Scalability

> **Up to 10x speedup compared to 32 cores based on f1.x2**

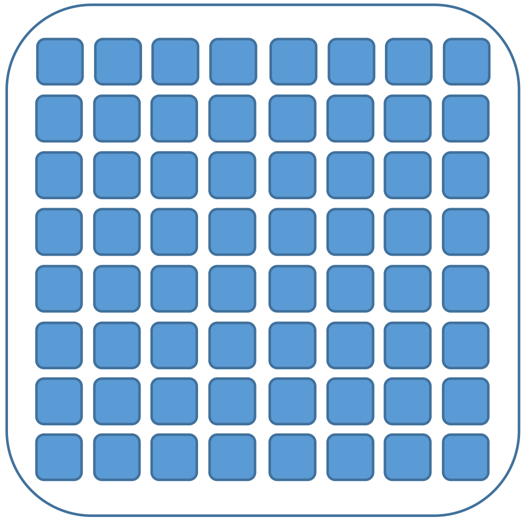




Speed up

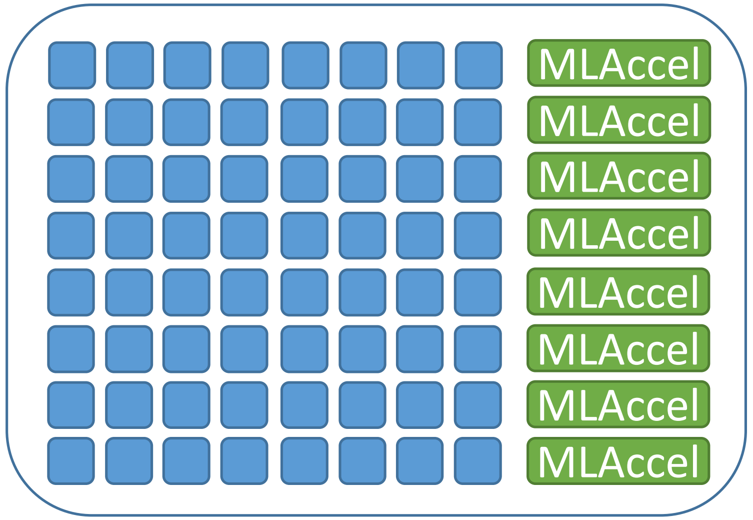
> **Up to 12x speedup compared to 64 cores on f1.x16**

f1.x16large (SW)

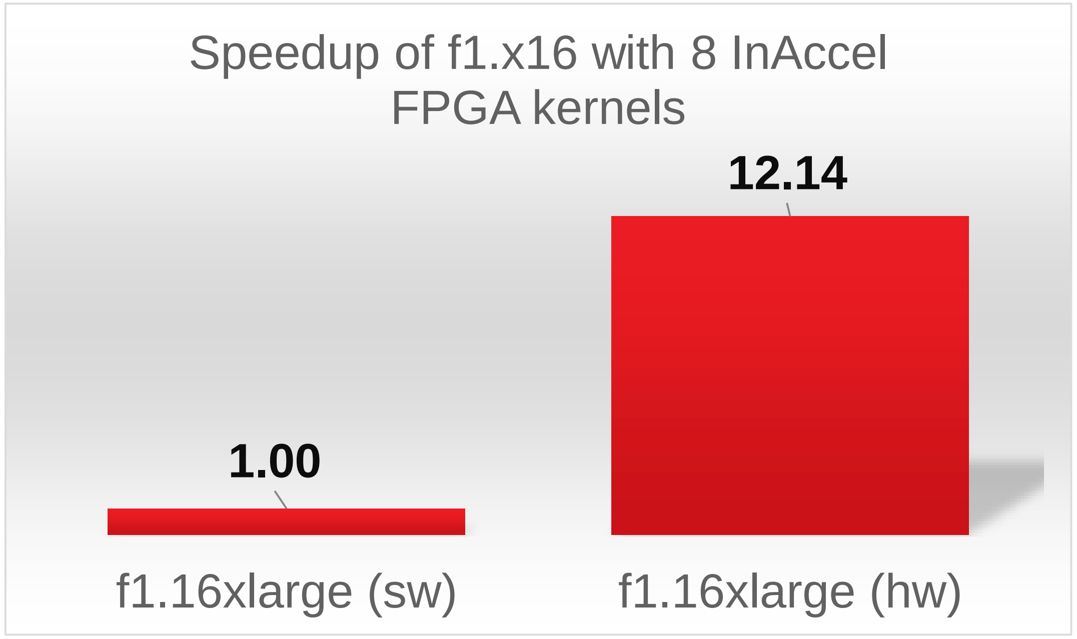


64 cores

f1.x16large (SW + 8 InAccel cores)



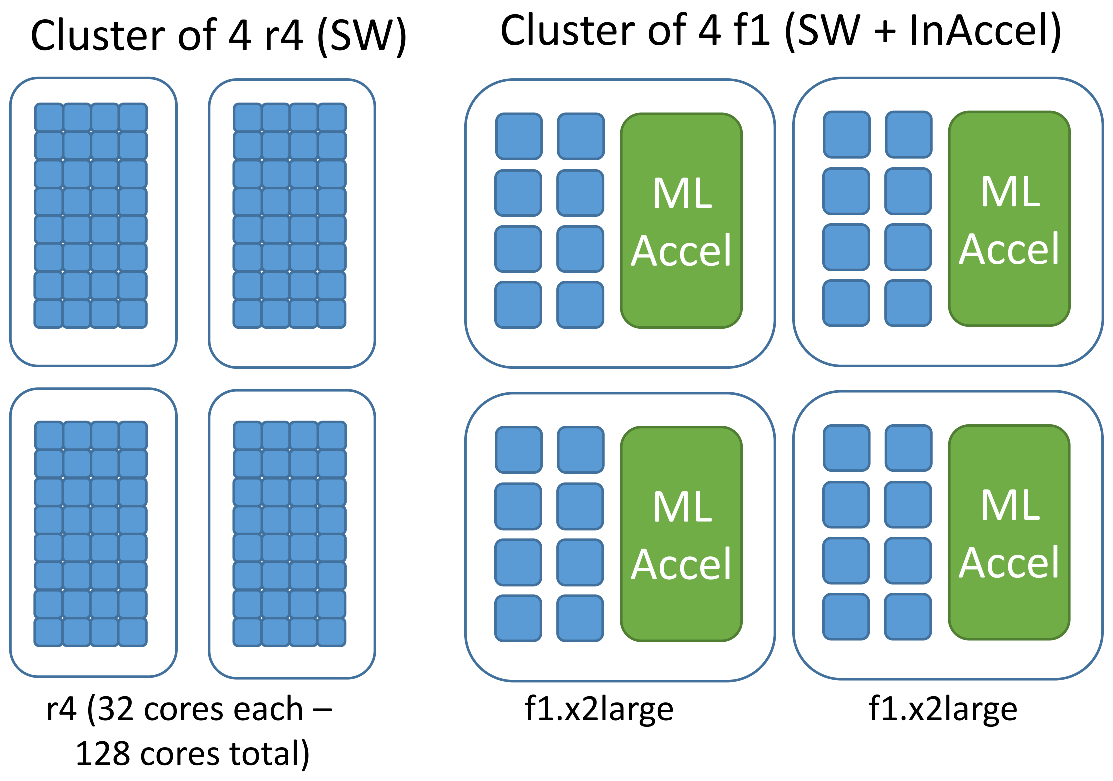
64 cores + 8 FPGAs with InAccel

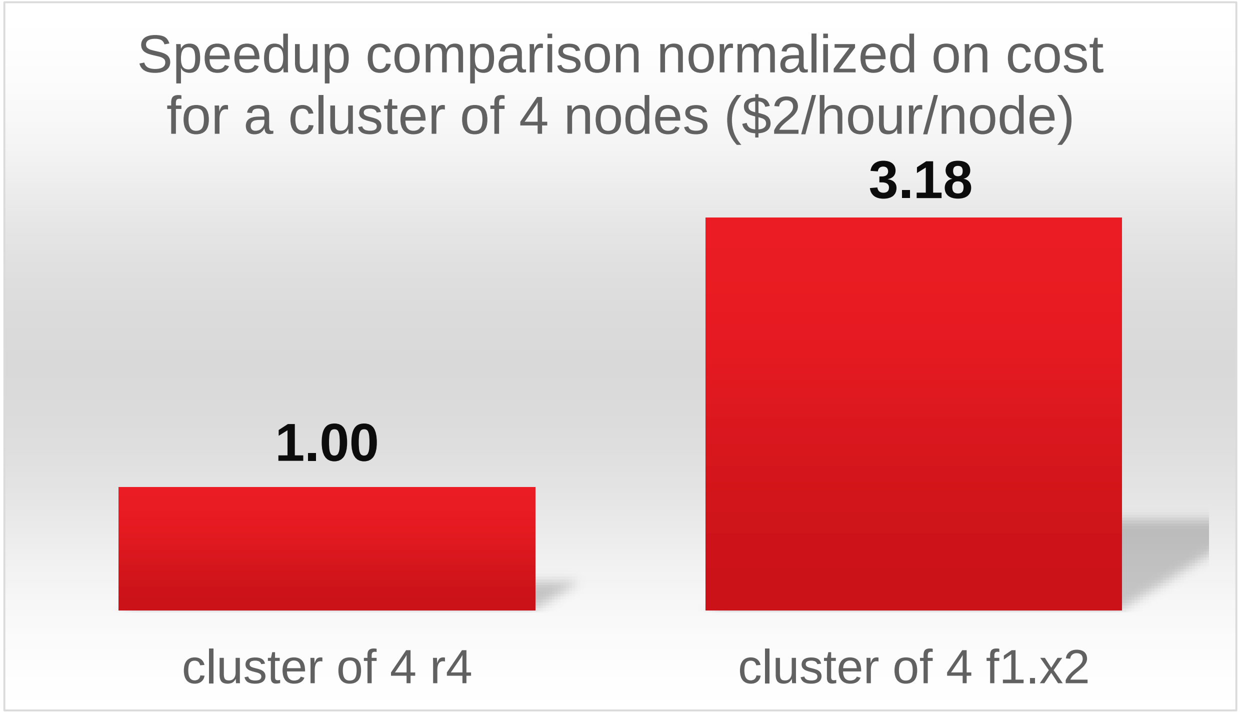


Speedup comparison

> 3x Speedup compared to r4

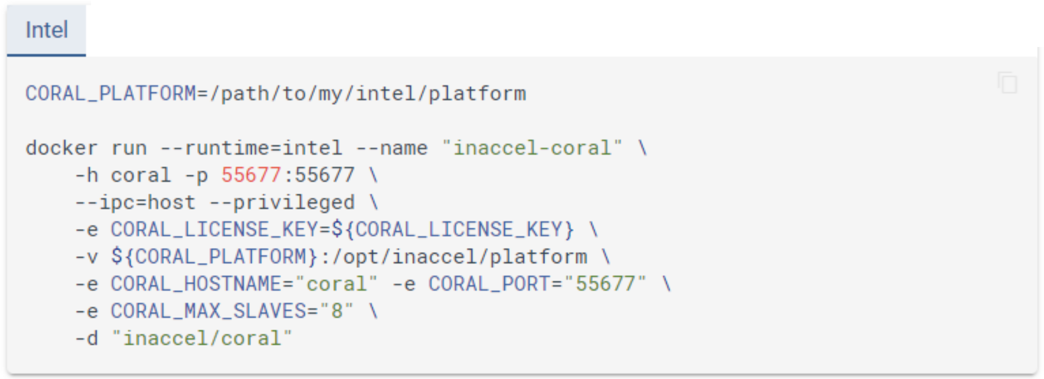
> 2x lower OpEx





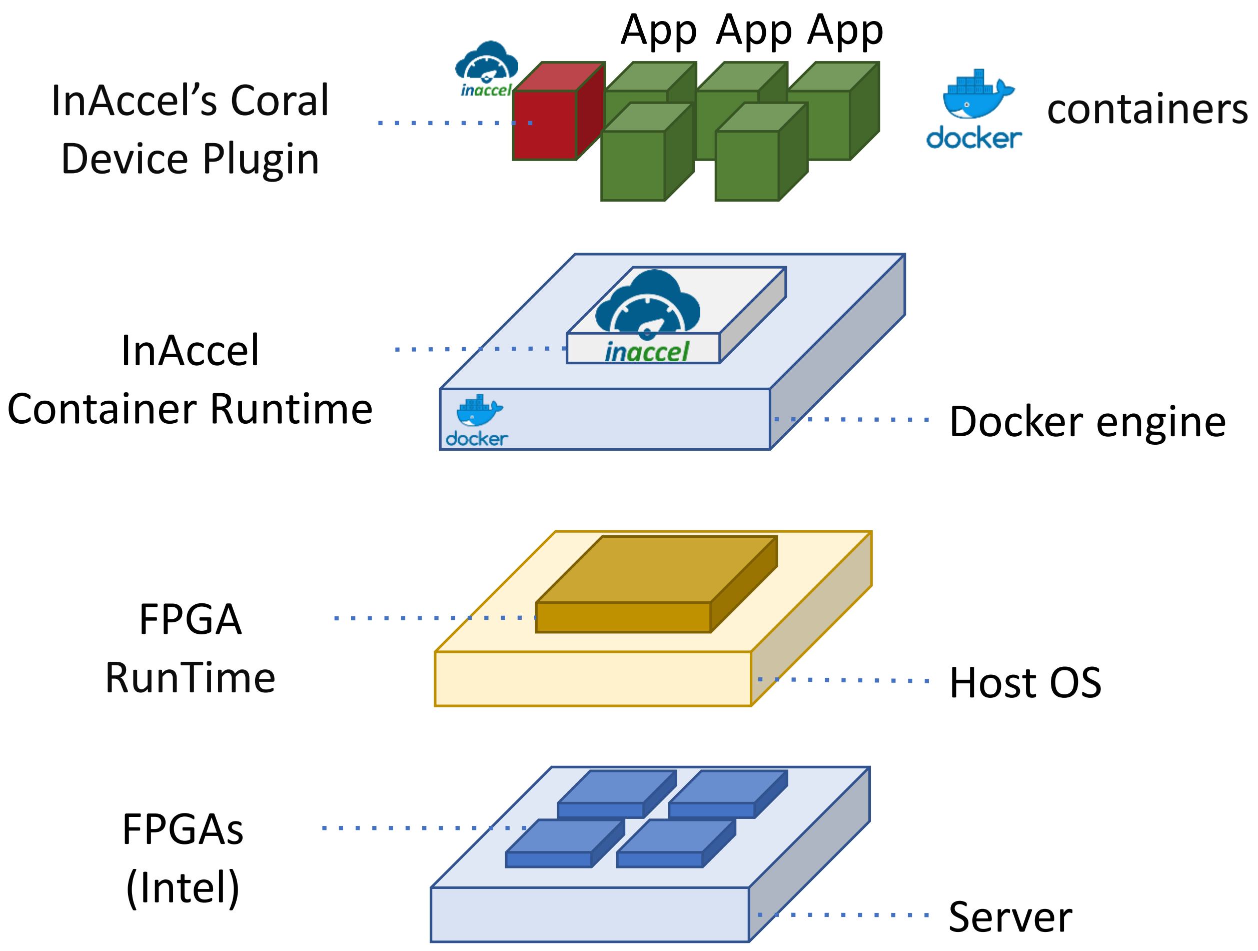
InAccel Docker Service

> **Sustain FPGA driver compatibility between the host and the containers**
- discover available resources
- mount/isolate visible devices
  - forget *--priviledged*
- resolve library dependencies



Intel

```
CORAL_PLATFORM=/path/to/my/intel/platform

docker run --runtime=intel --name "inaccel-coral" \
    -h coral -p 55677:55677 \
    --ipc=host --privileged \
    -e CORAL_LICENSE_KEY=${CORAL_LICENSE_KEY} \
    -v ${CORAL_PLATFORM}:/opt/inaccel/platform \
    -e CORAL_HOSTNAME="coral" -e CORAL_PORT="55677" \
    -e CORAL_MAX_SLAVES="8" \
    -d "inaccel/coral"
```

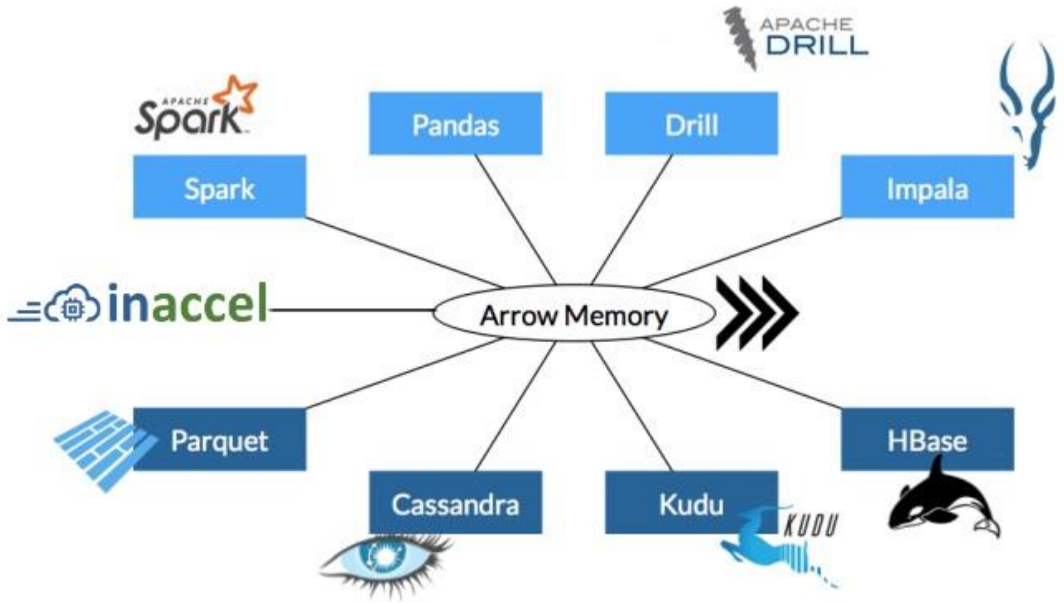


Integration with Arrow

> Seamless experience for the application developer writing software using Arrow-backed dataframes.

> Arrow adoptance growth makes our integration even more profound.

> Zero extra overhead to other operations supported by Arrow - e.g serialization



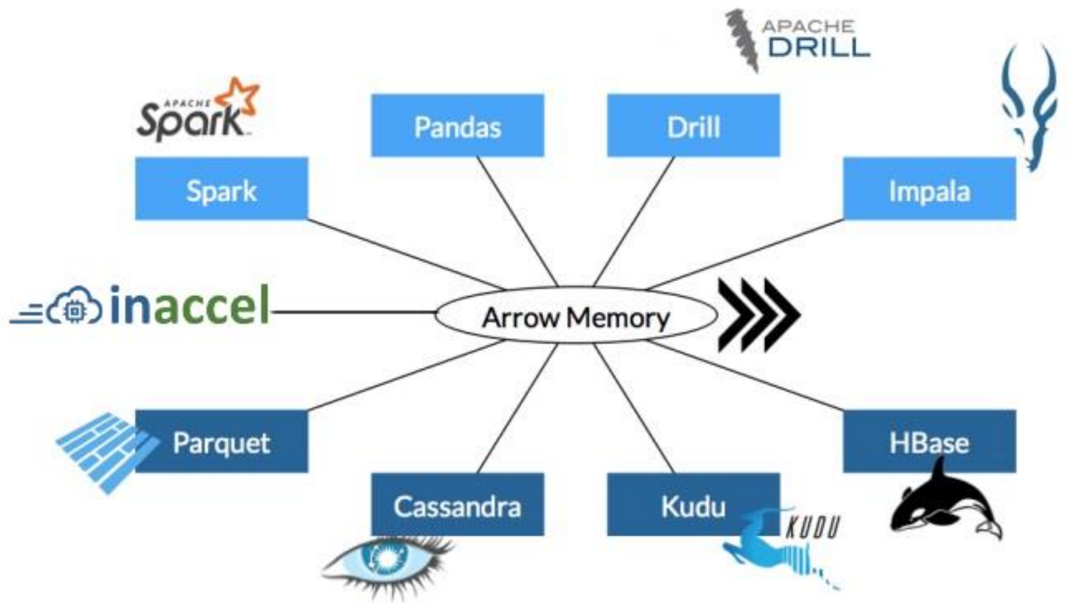
Apache Arrow specifies a standardized language-independent ***columnar memory format for flat and hierarchical data****, organized for efficient analytic operations on modern hardware. The Arrow memory format supports zero-copy reads for efficient data-access without serialization overhead. —* ***Apache Software***

Integration with Arrow

> Minimize copy overheads to convey data to the accelerator data plane.

> Single DMA operation from host memory to DDR memory on FPGA.

> Shared memory between CPU and FPGA will reduce the overhead to zero.
>> (CAPI – CCIX)



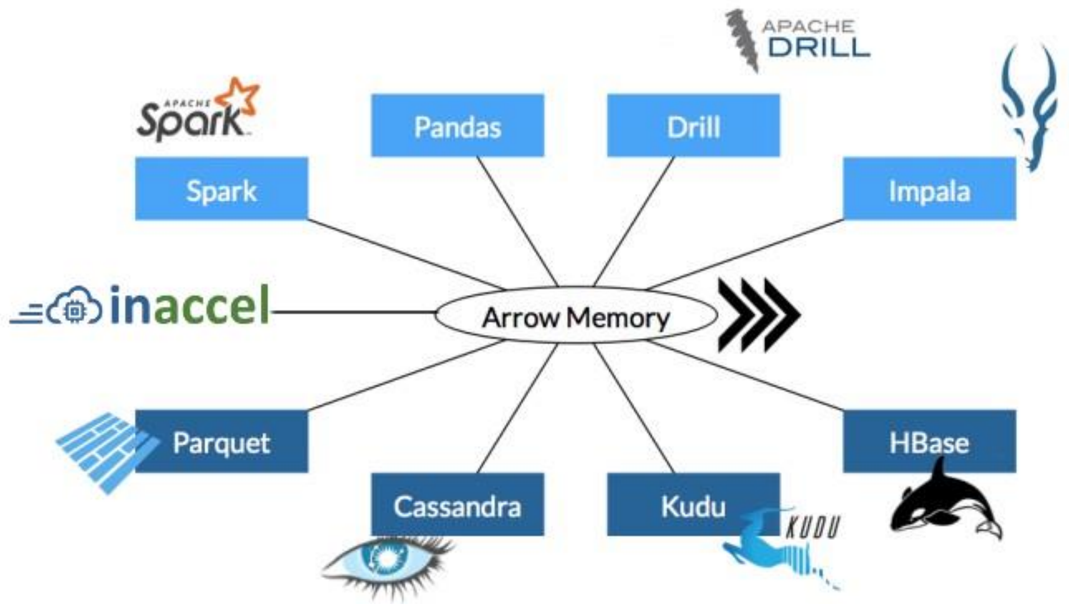
Apache Arrow specifies a standardized language-independent ***columnar memory format for flat and hierarchical data****, organized for efficient analytic operations on modern hardware. The Arrow memory format supports zero-copy reads for efficient data-access without serialization overhead. —* ***Apache Software***

Arrow Integration - Allocation flow

> Trigger the inaccel Arrow allocator by supplying metadata on a per-column basis.

> Arrow columns enabled for acceleration are mapped internally to shared memory.

> Coral - our FPGA manager - accesses Arrow-backed data with zero-copy.

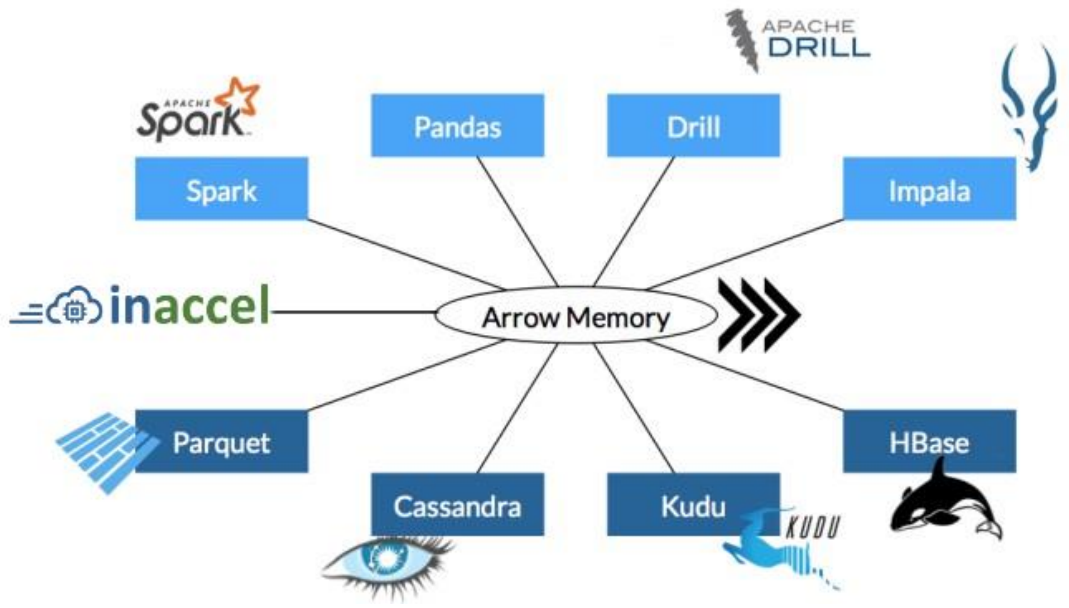


Apache Arrow specifies a standardized language-independent ***columnar memory format for flat and hierarchical data****, organized for efficient analytic operations on modern hardware. The Arrow memory format supports zero-copy reads for efficient data-access without serialization overhead. —* ***Apache Software***

Arrow Integration - Allocation flow

> Padding is added to page-align and boost DMA performance

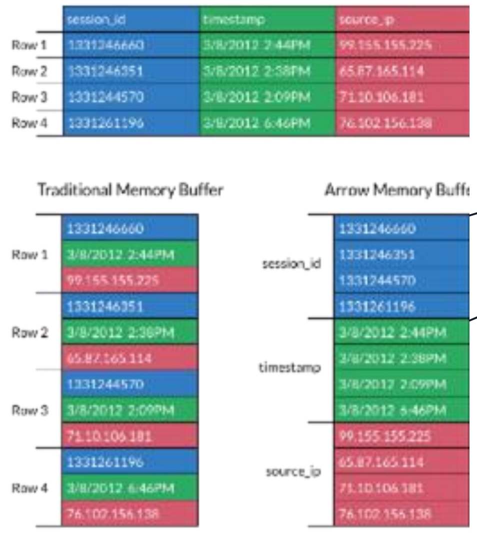
> Padding is ignored on serialization which ensures no extra transfer cost



Apache Arrow specifies a standardized language-independent ***columnar memory format for flat and hierarchical data****, organized for efficient analytic operations on modern hardware. The Arrow memory format supports zero-copy reads for efficient data-access without serialization overhead. —* ***Apache Software***

Apache Arrow Summing up

> **Seamless Arrow integration**
>
> **Page-aligned columnar format**
>
> **Native memory map**
>
> **Zero-copy operations**



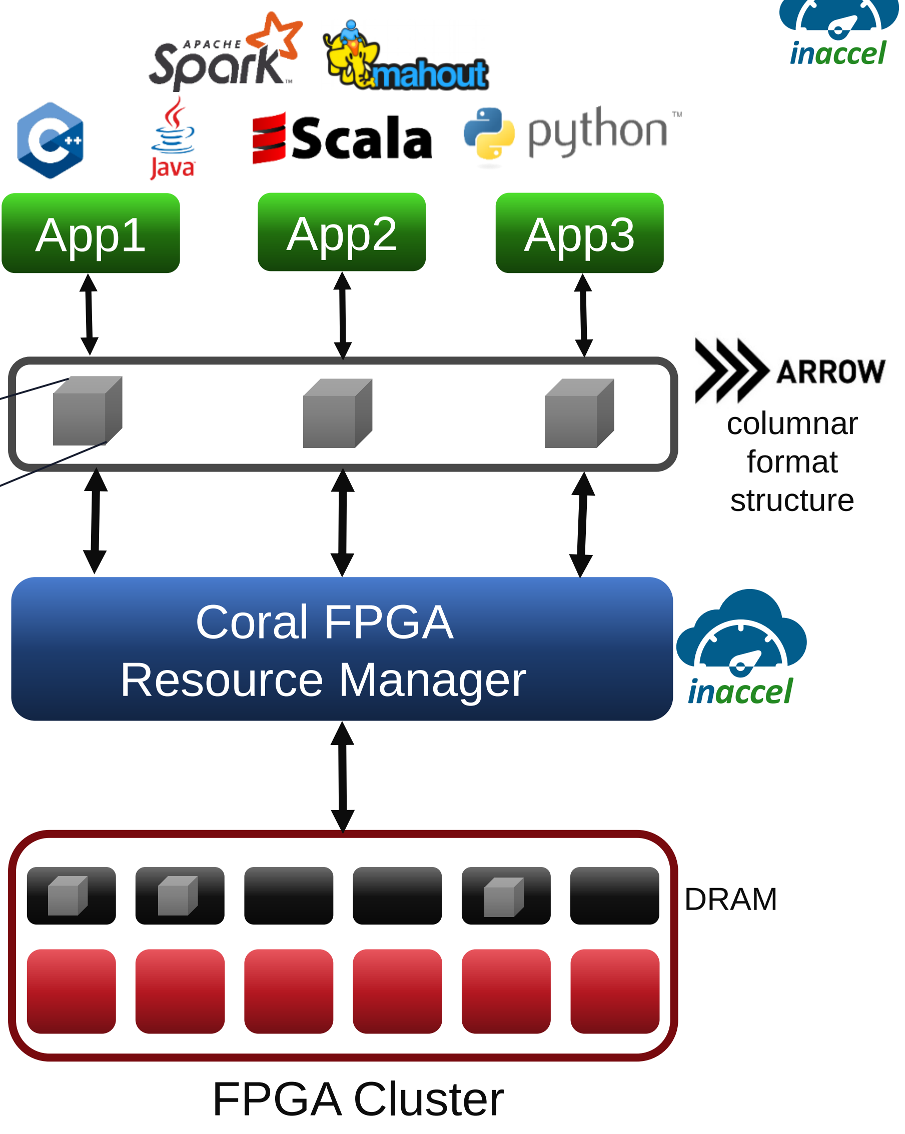
	session_id	timestamp	source_ip
Row 1	1331246660	3/8/2012 2:44PM	99.155.155.225
Row 2	1331246351	3/8/2012 2:38PM	65.87.165.114
Row 3	1331244570	3/8/2012 2:09PM	71.10.106.181
Row 4	1331261196	3/8/2012 6:46PM	76.102.156.138

Traditional Memory Buffer

Row 1	1331246660
	3/8/2012 2:44PM
	99.155.155.225
Row 2	1331246351
	3/8/2012 2:38PM
	65.87.165.114
Row 3	1331244570
	3/8/2012 2:09PM
	71.10.106.181
Row 4	1331261196
	3/8/2012 6:46PM
	76.102.156.138

Arrow Memory Buffe

session_id	1331246660
	1331246351
	1331244570
	1331261196
timestamp	3/8/2012 2:44PM
	3/8/2012 2:38PM
	3/8/2012 2:09PM
	3/8/2012 6:46PM
source_ip	99.155.155.225
	65.87.165.114
	71.10.106.181
	76.102.156.138



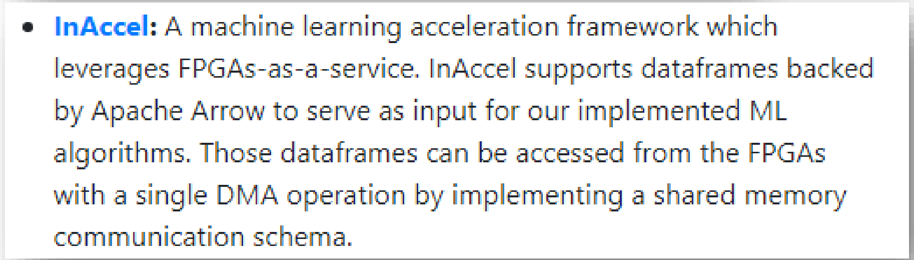
Example

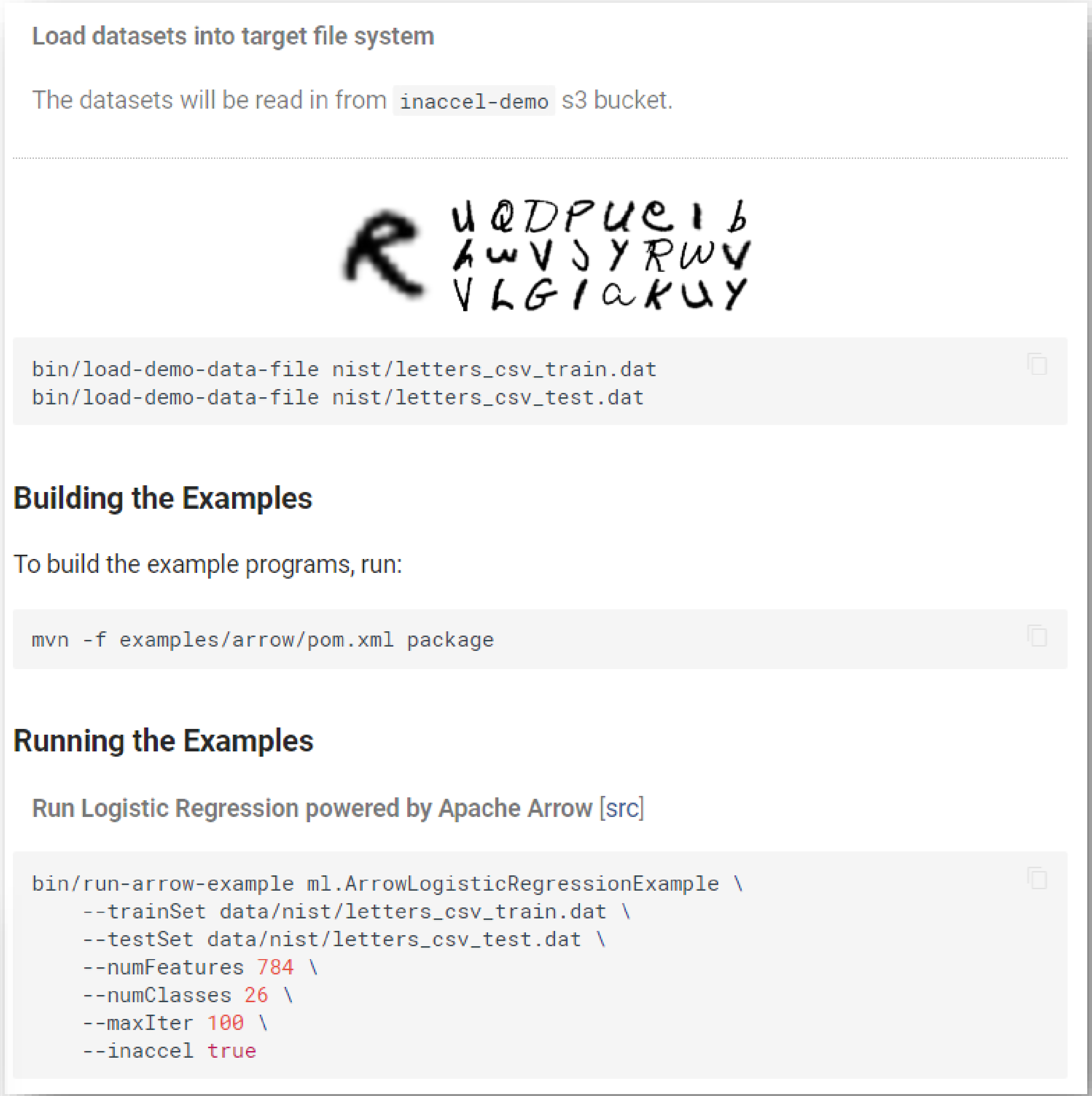
- > **Example on logistic regression on top of FPGA resource manager**

Apache Arrow™

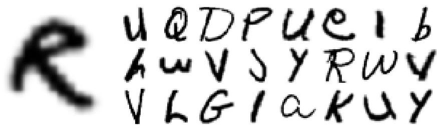
Powered By

Project and Product names using "Apache Arrow"


- **InAccel:** A machine learning acceleration framework which leverages FPGAs-as-a-service. InAccel supports dataframes backed by Apache Arrow to serve as input for our implemented ML algorithms. Those dataframes can be accessed from the FPGAs with a single DMA operation by implementing a shared memory communication schema.


Load datasets into target file system

The datasets will be read in from `inaccel-demo` s3 bucket.



```
bin/load-demo-data-file nist/letters_csv_train.dat
bin/load-demo-data-file nist/letters_csv_test.dat
```

Building the Examples

To build the example programs, run:

```
mvn -f examples/arrow/pom.xml package
```

Running the Examples

Run Logistic Regression powered by Apache Arrow [src]

```
bin/run-arrow-example ml.ArrowLogisticRegressionExample \
    --trainSet data/nist/letters_csv_train.dat \
    --testSet data/nist/letters_csv_test.dat \
    --numFeatures 784 \
    --numClasses 26 \
    --maxIter 100 \
    --inaccel true
```

https://docs.inaccel.com/latest/manager/examples/

InAccel, Inc. Corporate overview

> **Founded in January 2018**

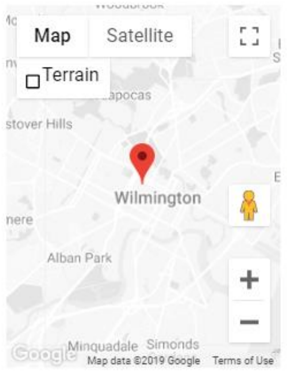
> **Registered in Delaware, USA**

> **June 2018: Seed Funding ($600,000 from Marathon VC)**

> **Membership:**

Alibaba Cloud

intel FPGA Design Solutions Network



Headquarters

500 Delaware Ave STE 1, #1960
Wilmington, DE 19801
USA

(+1) 408 260 5724

Design Center

Formionos 47
Kesariani 116 33
Athens, Greece

(+30) 211 1825 436

inaccel

Application Acceleration, seamlessly

www.inaccel.com

info@inaccel.com

USA:

500 Delaware Ave STE 1, #1960
Wilmington, DE 19801
USA

Europe (Design Center):

Formionos 47
Kesariani 116 33
Athens, Greece