FPGA deployment and scaling made simple

Chris Kachris
chris@inaccele.com
What software developers want

% OF RESPONDENTS WHO CONSIDERED THE FEATURE VERY IMPORTANT
More than one feature could be selected.

91% PERFORMANCE

69% EASE OF DEPLOYMENT

76% EASE OF PROGRAMMING

Source: Databricks, Apache Spark Survey 2016, Report
What software developers want

% OF RESPONDENTS WHO CONSIDERED THE FEATURE VERY IMPORTANT

More than one feature could be selected.

91% PERFORMANCE

69% EASE OF DEPLOYMENT

76% EASE OF PROGRAMMING

Source: Databricks, Apache Spark Survey 2016, Report
inaccele mission

> Help companies **speedup** their applications by using **accelerators** (FPGAs) seamlessly

> How?

Unique InAccel FPGA orchestrator

Automated **deployment**, **scaling** and **management** of FPGA clusters
Main challenges of FPGAs

- Hard to develop accelerators
  - Solved

- Hard to scale-out (kernels/cards)

- Hard to share (process/users/apps)

- Hard to deploy/invoke kernels
## InAccel Products (Accelerators as IP)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Speedup</th>
<th>Cost reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logistic Regression</td>
<td>15x</td>
<td>4x</td>
</tr>
<tr>
<td>K-means Clustering</td>
<td>14x</td>
<td>4x</td>
</tr>
<tr>
<td>Naïve-Bayes</td>
<td>5x</td>
<td>2x</td>
</tr>
<tr>
<td>FAISS (Similarity search)</td>
<td>2x</td>
<td>1.5x</td>
</tr>
<tr>
<td>Xgboost</td>
<td>6x</td>
<td>2x</td>
</tr>
</tbody>
</table>

GitHub: [https://github.com/inaccel](https://github.com/inaccel)
Current limitations for FPGA deployment

> Currently only **one application** can talk to a single FPGA accelerator through **OpenCL**

> 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)

> User need to specify which FPGA to use (device ID, etc.)
From single instance to data centers

- Easy deployment
- Instant scaling
- Seamless sharing
- Multiple-users
- Multiple applications
- Isolation
- Privacy
Scalable Orchestrator for FPGA clusters

Automated Deployment, Scaling and Management of FPGA clusters

- Seamless invoking from C/C++, Python, Java and Scala. No need for OpenCL
- Automatic configuration and management of the FPGA bitstreams and memory
- Seamless sharing of the FPGA cluster from multiple threads/processes/applications/users
- Fully scalable: Scale-up (multiple FPGAs per node) and Scale-out (multiple FPGA-based servers over Spark)
FPGA bitstream repository

> FPGA Resource Manager is integrated with Jfrog bitstream repository that is used to store FPGA bitstreams

![Diagram showing FPGA Resource Manager integrated with Jfrog bitstream repository](https://store.inaccel.com)
FPGA repository - artifact

Artifact: JSON file + bitstream

- Vendor
- FPGA cards
- Version
- Kernels
- Arguments
FPGA repository CLI

- Simple CLI interface for the local repository
- Local repository can fetch artifacts from external links (e.g. Xilinx or InAccel repository) with proprietary artifacts

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inaccel bitstream decrypt</td>
<td>Decrypt an FPGA binary</td>
</tr>
<tr>
<td>inaccel bitstream encrypt</td>
<td>Encrypt an FPGA binary</td>
</tr>
<tr>
<td>inaccel bitstream parse</td>
<td>Parse FPGA binary build-metadata</td>
</tr>
<tr>
<td>inaccel bitstream install</td>
<td>Install a bitstream to the local or a remote repository, from a local or a remote source</td>
</tr>
<tr>
<td>inaccel bitstream list</td>
<td>List all the bitstreams or detailed information for specific bitstreams in the local or a remote repository</td>
</tr>
<tr>
<td>inaccel bitstream remove</td>
<td>Remove one or more bitstreams from the local or a remote repository</td>
</tr>
</tbody>
</table>
Compatible with any Xilinx Vitis libraries

Generality

> Build your own repository of accelerators.
InAccel provides a stack of cores including Machine Learning. You can combine all these libraries, along with your own ones, seamlessly in the same application.

> Test it with any Xilinx Vitis/SDAccel example

https://docs.inaccel.com/latest/manager/examples/
Simple invoking, deployment

No need for OpenCL

- Much simpler invoking
- Software-alike function invoking
- No need for OpenCL directives
- Same API for C/C++, Java, Python
- Native API

No need to allocate buffers
No need to specify bitstreams
No need to program specific device

```
std::string binaryfile = bin_file;
size_t vector_size_bytes = sizeof(float) * DATA_SIZE;
cl_int err;
cl::Context context;
cl::Kernel kmw_vector_add;
cl::CommandQueue q;

// Allocate memory in host memory
// when creating a buffer with user pointer (cl_mem_use_host_ptr), under the host user ptr
// Is used if it is properly aligned; when not aligned, runtime had no choice but to create
// a non host side buffer. So it is recommended to use this allocator if user wants to
// create buffer using CL_MEM_USE_HOST_PTR to align user buffer to page boundary. It will
// ensure that buffer is used when user creates buffer from object with CL_MEM_HOST_PTR
std::vector<int, alignas(sizeof(int))> source_int_array(DATA_SIZE);
std::vector<float, alignas(sizeof(float))> source_Float_array(DATA_SIZE);

// Create the test data
std::generate(source_int_array.begin(), source_int_array.end(), std::rand);
std::generate(source_Float_array.begin(), source_Float_array.end(), std::rand);
for (int i = 0; i < DATA_SIZE; i++) {
    source_int_array[i] = source_int_array[i] + source_Float_array[i];
    source_Float_array[i] = 0;
}

// OPENCL HOST CODE AREA START
// get_hwa_devices() is a utility API which will find the silicon
// platforms and will return list of devices connected to silicon platform
auto devices = kmw::get_hwa_devices();

// load_binary_file() is a utility API which will load the binaryfile
// and will return the pointer to file buffer.
auto fileBuffer = kmw::load_binary_file(binaryfile);
c1::Program instances_bin(fileBuffer.data(), fileBuffer.size());
int valid_dev = 0;
for (unsigned int i = 0; i < devices.size(); i++) {
    auto device = devices[i];
    // Creating context and Command Queue for selected device
    auto cl = kmw::create_command_queue(device);
    auto clDev = kmw::create_context(device);
    std::cout << "Device name: " << device.getInfo<CL_DEVICE_NAME>() << std::endl;
    std::cout << "Device GUID: " << device.getInfo<CL_DEVICE_ID>() << std::endl;
    std::cout << "Device Vendor: " << device.getInfo<CL_DEVICE_VENDOR>() << std::endl;
    if (err != CL_SUCCESS) {
        std::cout << "Failed to program device" << std::endl;
    } else {
        std::cout << "Device(" << device.getInfo<CL_DEVICE_ID>() << ") program successful!" << std::endl;
    }
    cl::CommandQueue q;
    cl::Kernel kmw_vector_add;
    cl::Program kmw_vector_add(cl::Program(program_dir(), "add", err), err);
    cl::Event kmw_vector_add_start;
    valid_dev = 0;
}
```
Instant scaling

inaccl coral start [command options]

Distribution of multi-thread applications to multiple clusters With a single command

FPGA bitstream repository

https://docs.inaccl.com/
Example on scaling to 2 FPGAs using the resource manager for logistic regression

1.86x speedup using 2 FPGAs simply by changing a line

Zero code changes

inaccel start --fpga=alveo:0

You specify how many FPGAs you want to use

or

inaccel start --fpga=all
Instant Scalability

> Instant scalability to

  >> Multiple kernels

  >> Multiple FPGAs in the same server

  >> Multiple servers (using Kubernetes)

inaccel start --fpga=xilinx:0,xilinx:1

Throughput (MB/sec)

Lower than 2x speedup is due to the lower clock rate and PCI conflicts
Zero overhead, Improved Throughput

- Zero overhead
- Improved Performance
- Instant scalability
- Fully virtualization
- Simpler programming

Throughput for Gzip (MB/sec)

<table>
<thead>
<tr>
<th>Throughput Type</th>
<th>Reference</th>
<th>Coral</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single File</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Batch (12 files)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Reference: Coral
Optimized Pipeline

> No need for OpenCL event to synchronize pipeline execution

> Instant pipeline using InAccel FPGA manager => Optimized pipeline
  >> Higher Throughput
  >> Simplified code (no need for OpenCL events, barriers)
  >> Useful for multiple kernels, multiple threads/applications
Seamless sharing of the resources

Resources sharing from multiple applications

Utilization from:
- Multiple threads
- Multiple processes
- Multiple applications
- Multiple users
- Multiple containers

https://docs.inaccel.com/
Jupyter - JupyterHub

- Deploy and run your FPGA-accelerated applications using Jupyter Notebooks

- InAccel manager allows the instant deployment of FPGAs through JupyterHub
JupyterHub on FPGAs

- Instant acceleration of Jupyter Notebooks with zero code-changes
- Offload the most computationally intensive tasks on FPGA-based servers
BinderHub on FPGAs

> BinderHub enables an end user to easily specify a desired computing environment from a Git repo

> InAccel integration allows users to accelerate their applications from git repo using remote FPGA-based accelerators

Unique FPGA monitoring tool

- com.inaccel.ml.logisticregression.gradients

com.inaccel.ml.logisticregression.gradients (1225630506)

Success: 3151 requests  Failure: 0 requests

Waiting
1 requests

Running
1 requests

Read time (ms) per request

Execute time (ms) per request

Write time (ms) per request

www.inaccel.com™
FPGA monitoring tool

- Utilization of the kernels
  - Time to read from DDR
  - Time to execute
  - Time to write back in DDR

- Useful for performance optimization

- Support for multiple kernels, multiple FPGA cards

- Integrated with Coral FPGA manager
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/](https://hub.docker.com/u/inaccel/)

- Easy deployment
- Easy scalability
- Easy integration
Compatibility with Xilinx Ecosystem

> FPGA repository for the Vitis Library
  >> Pre-compiled bitstreams for all Vitis Library for Alveo family cards (U50, U200, U250, U280)

> Scaling of Vitis libraries to multiple kernels, multiple Alveo cards, AWS f1.4x, f1.16x
Accelerated ML on top of SQL 2019

> Seamless integration with Spark ML over HDFS
  >> Zero Code changes
  >> Higher Performance
  >> Lower OpEx

https://www.youtube.com/watch?v=qu7ct_4CknM
Integration with SQL Big Data Cluster 2019

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

>> No-need to specify which FPGA to use
Serverless deployment

> Integrated framework for serverless deployment

> Compatible with Kubernetes

> Compatible with Kubeless, Knative

> Users only have to upload the images on the S3 bucket and then InAccel’s FPGA Manager automatically deploy the cluster of FPGAs, process the data and then store back the results on the S3 bucket.

> Users do not have to know anything about the FPGA execution.

https://medium.com/@inaccel/fpgas-goes-serverless-on-kubernetes-55c1d39c5e30
The InAccel Coral runtime specification aims to specify the configuration, and execution interface for the efficient management of any accelerator-based hardware resource.

Hook Developers

Hook developers can extend the functionality of a compliant runtime by hooking into an accelerator's lifecycle with an external application. Example use cases include sophisticated hardware configuration, advanced logging, IP licensing (hardware identification - authentication - decryption), etc.

Platform Developers

Platform developers can build runtime implementations that expose diverse hardware resources and system configuration, containing low-level OS and hardware-specific details, on a particular platform.

https://github.com/InAccel/runtime
Pricing model (enterprise version)

FPGA Resource manager

> Pricing model per node (server)

> Each node can have 1 to 8 FPGAs

<table>
<thead>
<tr>
<th>FPGA Resource manager</th>
<th>20 servers or less</th>
<th>More than 20 servers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monthly</td>
<td>$300/servers</td>
<td>$250/server</td>
</tr>
<tr>
<td>Yearly</td>
<td>$3,000/server</td>
<td>$2,500/server</td>
</tr>
</tbody>
</table>
Successful Use cases, Integrations

Scala, Kubeflow, binder, Spark, Python, Java, SQL Server, APACHE, Arrow, KubespHERE, Docker, Keras, Kubernetes, Jupyter, sklearn, learn
Partnerships

- Intel FPGA Design Solutions Network
- Xilinx Alliance Program Member
- Achronix Semiconductor Corporation
- Amazon Web Services Advanced Technology Partner
- Alibaba Cloud Registered Technology Partner
- Nimbix
- Huawei
- Tencent Cloud
Making FPGA-based acceleration easy

info@inaccel.com

www.inaccel.com