
# Algean: An Open Framework for Machine Learning on a Heterogeneous Cluster

**Naif Tarafdar**[1], Giuseppe Di Guglielmo[2], Philip C Harris[3], Jeffrey D Krupa[3], Vladimir Loncar[4], Dylan S Rankin[3], Nhan Tran[5], Zhenbin Wu[6], Qianfeng Shen[1] and Paul Chow[1]

University of Toronto[1]
Columbia University[2]
Massachusetts Institute of Technology[3]
CERN[4]
Fermilab[5]
University of Illinois[6]

# Machine Learning

- One of the most popular topics of research
  - In many areas, many applications (e.g medical, financial, safety, transportation etc.)
  - Also within the computing community
- Wide usage in world pushes limits of devices
  - Metrics include performance and energy
  - Leading many researchers to consider heterogeneity!

# Heterogeneity All Around Us






# Applying Machine Learning to a Heterogeneous Environment


- Challenge: How do you design machine learning algorithms for a heterogenous space?
  - Hard enough with a homogenous computing environment
  - Is there a framework for such a thing?
- Challenge: If such a framework exists can we get both flexibility and performance?



# Outline

- Brief Motivation
- Overview of machine learning frameworks
  - Categorized as an abstraction layer stack
- Overview of Algean
  - HLS4ML
  - Galapagos
- Results




# MACHINE LEARNING FRAMEWORKS

# Many Popular Examples!

- Such as
  - Tensorflow
  - PyTorch
  - Caffe
  - Intel DLA
  - Xilinx XfDNN



- What do these different frameworks offer?
  - Depends on who you ask!

# Machine Learning Stack

# Machine Learning Stack


E.g: Neural net layers, quantization, compression, pruning

# Machine Learning Stack


E.g: Physical Connections (PCIe, ethernet etc.), Communication Protocols

# Machine Learning Stack


E.g: Hardware circuit (multipliers, shifters), memory architecture (caching etc.)

# Machine Learning Stack


- Allows researchers to pick and choose layers they wish to configure
- Collapsable/Expandable for specific application and infrastructure!




# AIGEAN OVERVIEW

# AIGean Introduction

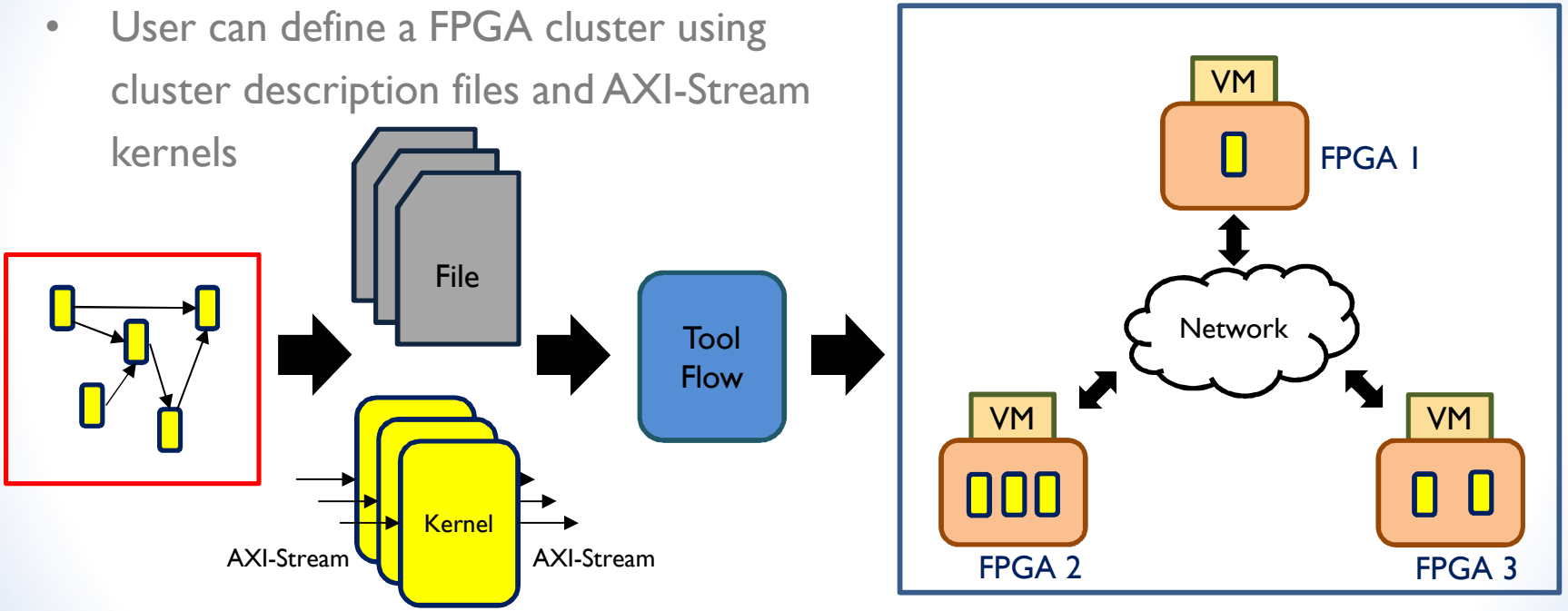
- Like the archipelago and sea
- Combines two existing frameworks:
  - HLS4ML:
    - HLS IP cores of ML IP
  - Galapagos
    - Connects and deploys heterogeneous distributed application across multiple nodes

# HLS4ML

- Open source project
- Input:
  - Description of FPGA resources
    - LUT, BRAM, DSP
  - Description of neural net
    - PyTorch support
- Output:
  - HLS synthesizable C++ that fits within resource constraints implementing neural net
- Tunable HLS code, made to fit the FPGA

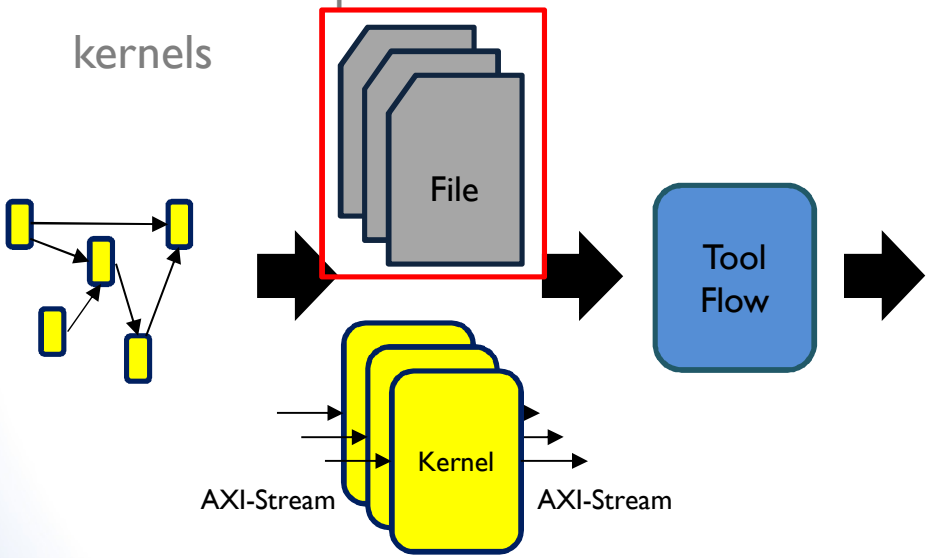
# Galapagos

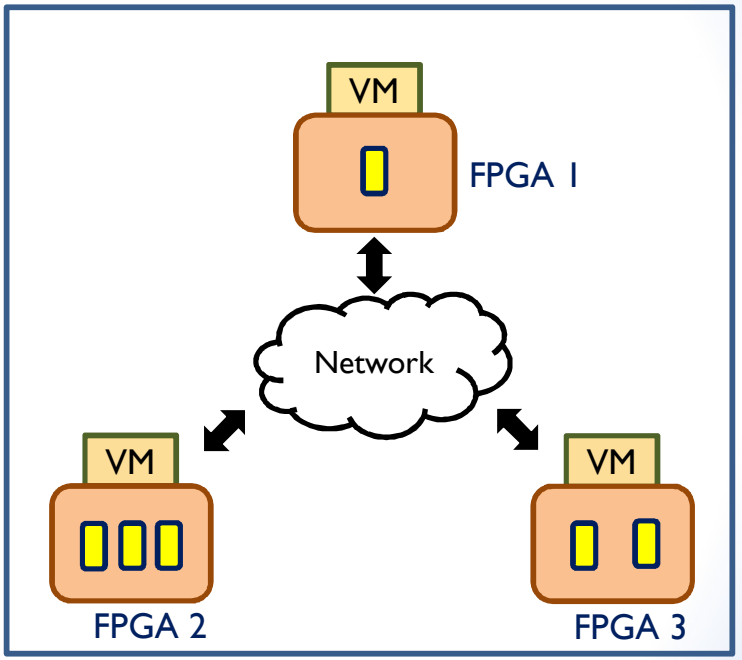
- User can define a FPGA cluster using cluster description files and AXI-Stream kernels



# Galapagos

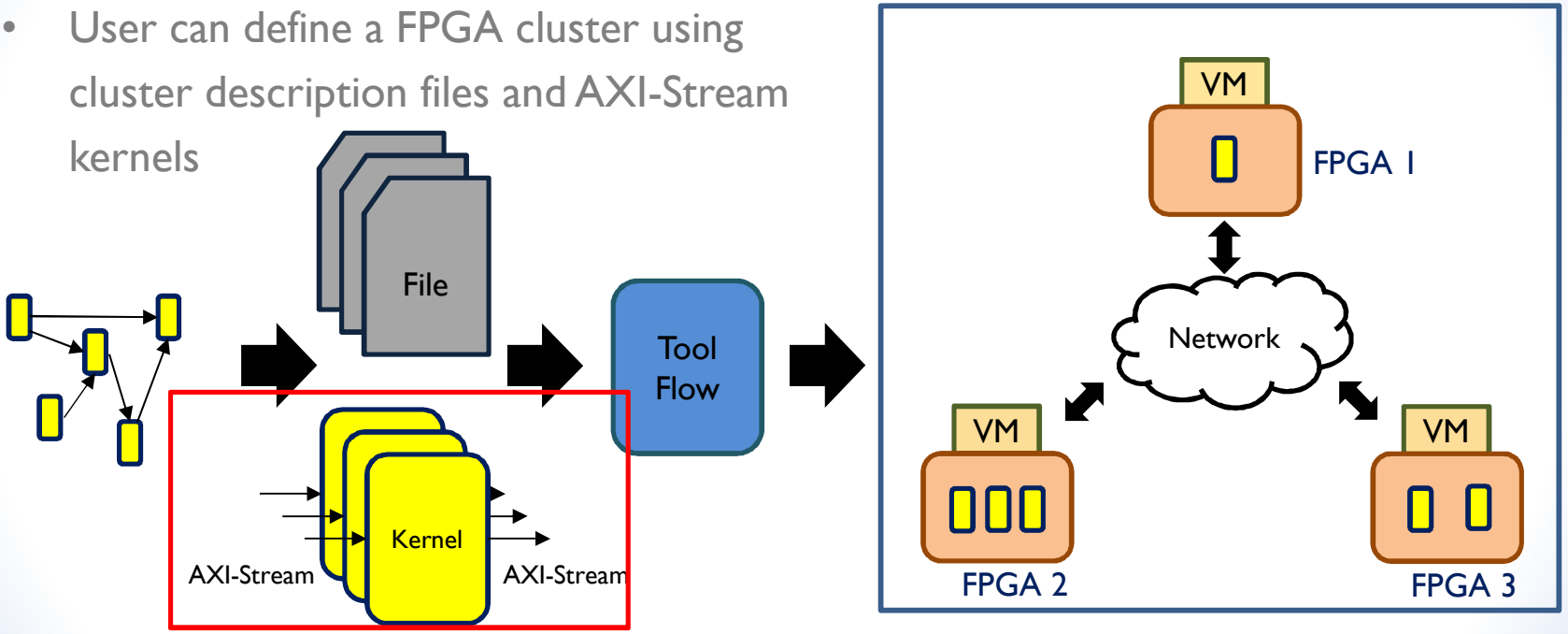
- User can define a FPGA cluster using cluster description files and AXI-Stream kernels



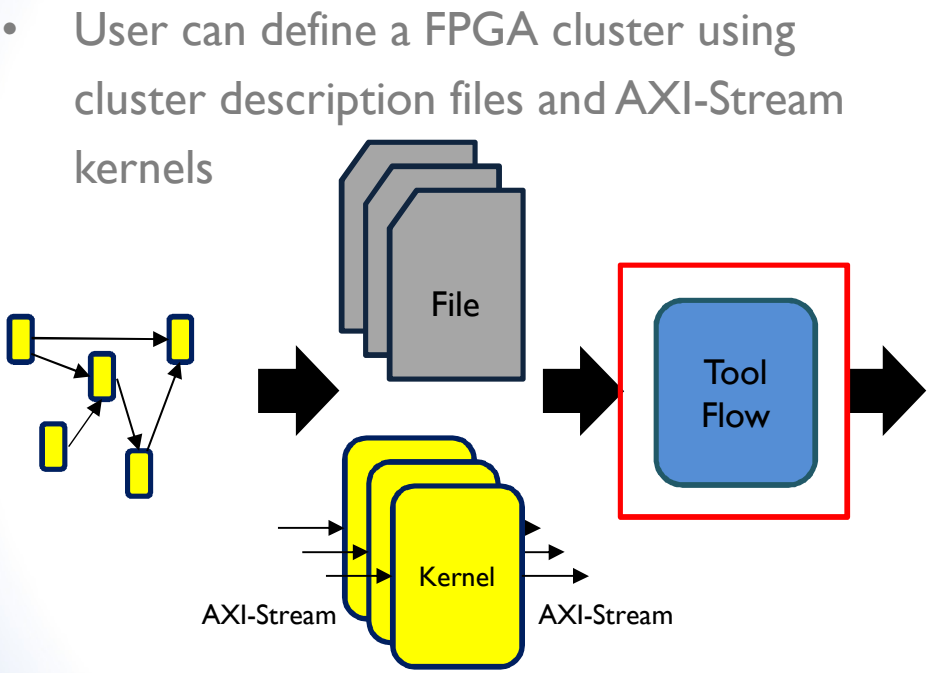
# Galapagos

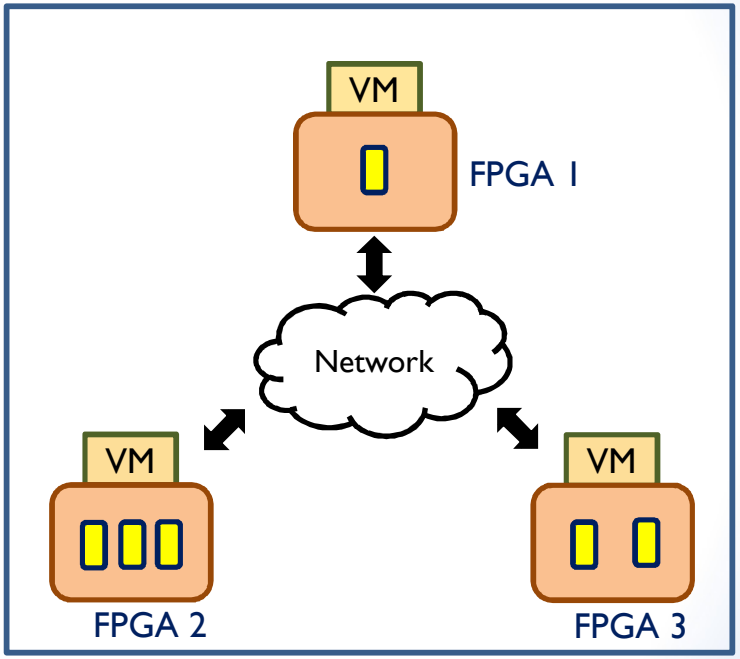
- User can define a FPGA cluster using cluster description files and AXI-Stream kernels



# Galapagos

- User can define a FPGA cluster using cluster description files and AXI-Stream kernels



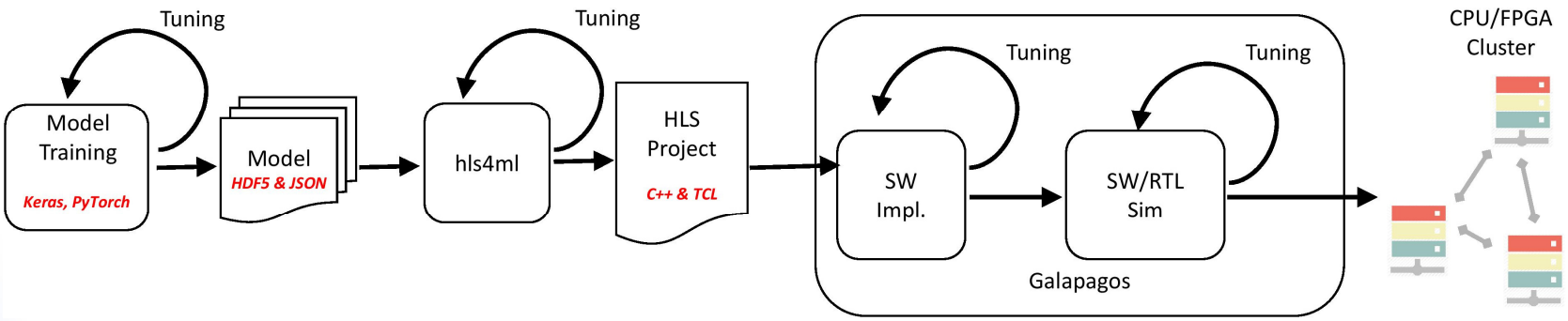
# Galapagos

- Heterogeneous Stack
- Allows users to create flexible heterogeneous clusters across CPUs/FPGAs
- Seamlessly prototype by implementing both on CPU and FPGA
  - Galapagos ensures functional portability for network communication
  - Essentially "network-connected" HLS kernels
    - For both SW and HW
  - Iterative development, selectively move bottleneck from SW to hardware without modifying code
- Flexibly change communication protocol without modifying user application
  - TCP, UDP, L1 etc
  - User application is agnostic to this

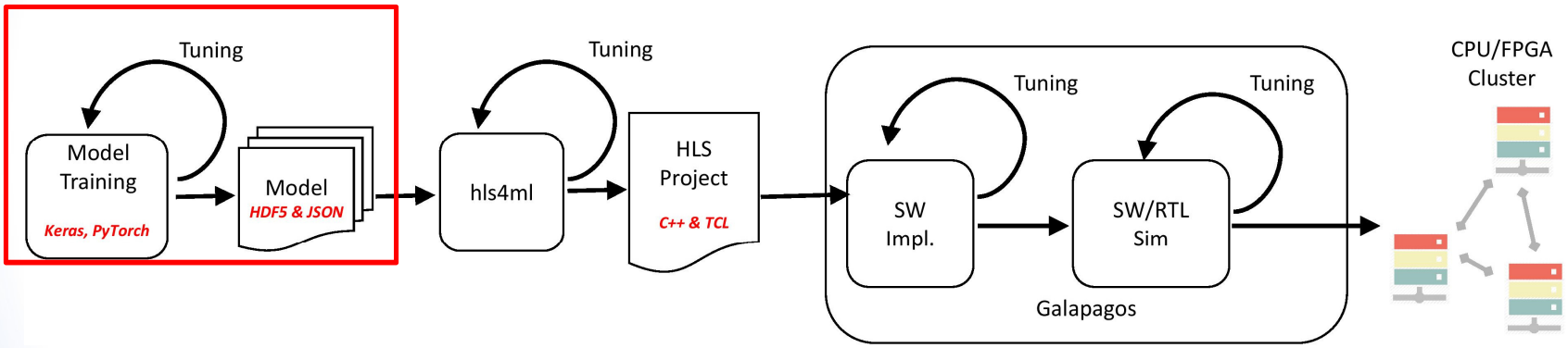
# Birth of Algean

- HLS4ML creates HLS IP core to maximize FPGA utilization
- Galapagos can give a multi-FPGA fabric
- Tools combined to deploy neural-net on multi-FPGA Fabric

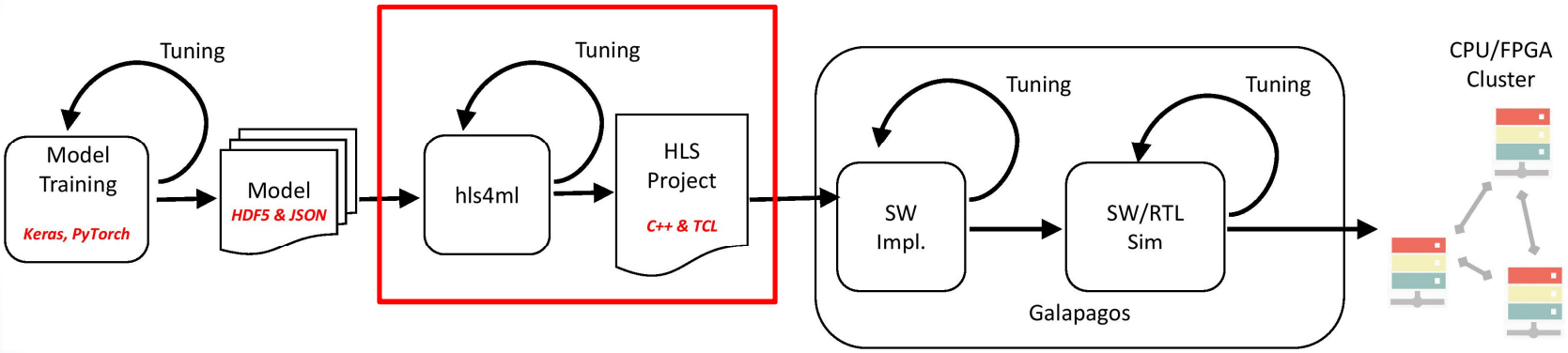
# Algean Tool Flow



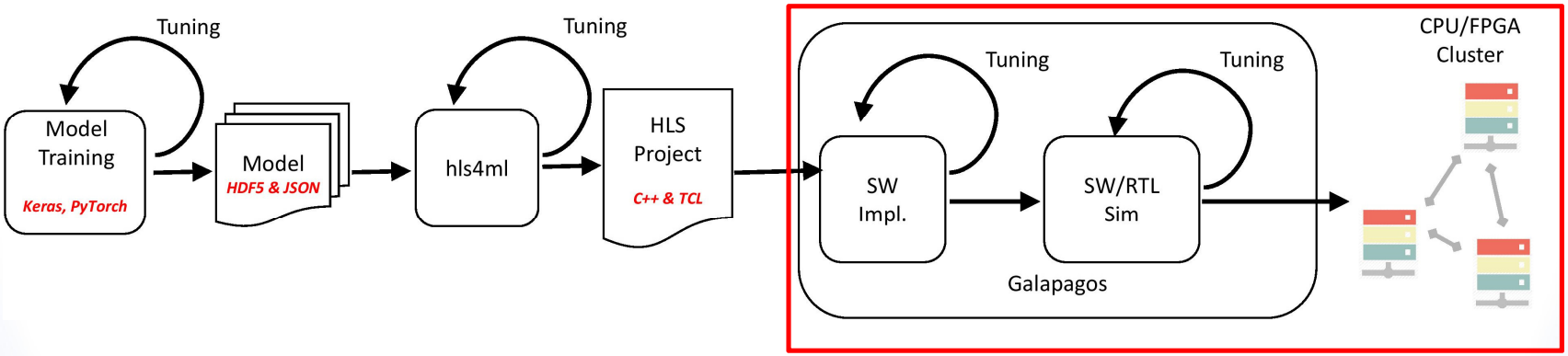
# Algean Tool Flow



# Algean Tool Flow

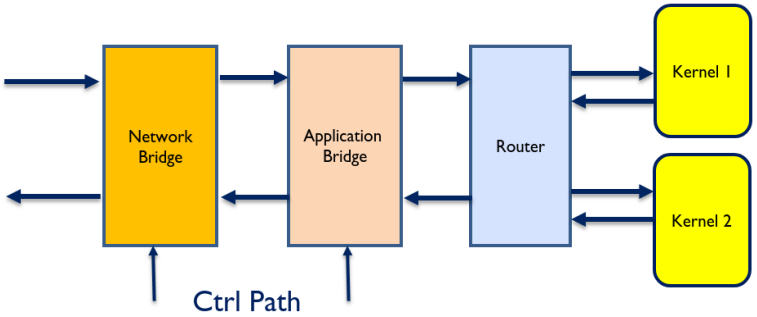


# Algean Tool Flow



# Galapagos/HLS4ML Modifications

- Galapagos uses bridging to connect HLS kernels together
  - By default uses AXI-Stream protocol
  - When kernels sends packet to an off-chip kernel, AXI-Stream packet is transformed into a network packet via a Galapagos bridge
- Galapagos modified to automate the formation of user specified bridges
  - Convert user protocol (HLS4ML) into Galapagos AXI-Stream
- HLS4ML modified to stream packets between IP cores

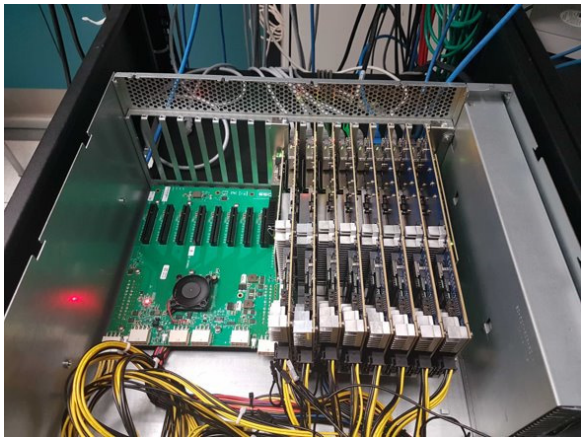





# RESULTS

# Experiment Setup

- CPUs
  - Xeon E5-2650
    - 24 Cores at 2.2 GHz
- FPGAs
  - Fidus Sidewinder
    - ZU19EG FPGA
      - ~1 Million logic cells, 35 MB BRAM, 1968 DSP slices
    - 100 GB network interface
  - 100 GB UDP core



# Microbenchmarks

- Latency send single flit
- Throughput: maximum throughput of link (varying packet size for software)

| Link | Latency | Throughput |
|---|---|---|
| Software to Hardware | 0.029 ms | 0.244 GB/s |
| Hardware to Hardware | 0.00017 ms | 100 GB/s |
| Hardware to Software | 0.0203 ms | N/A |




# Microbenchmarks

- Larger the packet, higher the throughput.
- UDP packet size limited
  - No segmentation
  - MTU size
  - Jumbo Frames: 8K

| Link | Latency | Throughput |
|---|---|---|
| Software to Hardware | 0.029 ms | 0.244 GB/s |
| Hardware to Hardware | 0.00017 ms | 100 GB/s |
| Hardware to Software | 0.0203 ms | N/A |

# Microbenchmarks

- Line-rate, same throughput at small and large packet size

| Link | Latency | Throughput |
|---|---|---|
| Software to Hardware | 0.029 ms | 0.244 GB/s |
| Hardware to Hardware | 0.00017 ms | 100 GB/s |
| Hardware to Software | 0.0203 ms | N/A |

# Microbenchmarks

- HW at line-rate
- UDP, SW can't keep up and we see packet drop

| Link | Latency | Throughput |
|---|---|---|
| Software to Hardware | 0.029 ms | 0.244 GB/s |
| Hardware to Hardware | 0.00017 ms | 100 GB/s |
| Hardware to Software | 0.0203 ms | N/A |

# Small Neural Network: Results

- Single CPU, single FPGA, used in physics application to calculate energy of a particle
- 16K inferences
- SDAccel (without AIgean) 3 ms
- AIgean 6.3 ms
  - Latency of single inference 0.08 ms, we can do this since streaming, not possible via SDAccel
- **Bottleneck: Sending data to FPGA via CPU network link**

# Small Neural Network: Takeaway


- Comparison vs SDAccel shows that network link for a single FPGA can be competitive with PCIe
  - Network link wins in terms of scalability, many more available FPGAs via network vs PCIe
- Can stream data
  - Latency of single inference a lot faster
- Should target larger application
  - We can do this as we have a large multi-FPGA fabric!



# Autoencoder: Results

- Autoencoder implemented in both SDAccel on single FPGA and AIgean using 3 FPGAs
- SDAccel: Single FPGA, higher reuse factor to fit logic
  - 0.26 ms
- AIgean: Three FPGAs
  - 0.08 ms, more than 3x improvement

# Autoencoder: Takeaway

- Using a larger fabric allows us to implement larger circuits
- The difficulty of communication between multi-FPGA is abstracted away

# Resnet-50

- Large fabric allows us to target Resnet-50
- Individual layers per FPGA
- Work in progress
- Estimated throughput: 5 ms (longest layer) per image
  - Functionality first, no internal optimizing yet

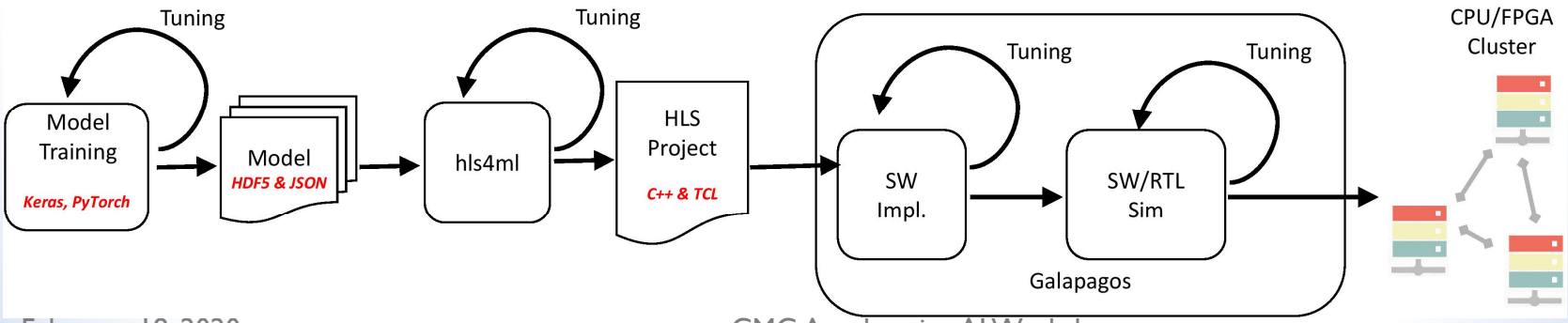



# SUMMARY AND CONCLUSION

# Summary

- Multi-FPGA/CPU neural net framework by leveraging and combining HLS4ML and Galapagos frameworks
- Tunable IP cores, flexible communication
- ML HLS IP cores deployed onto cluster of network connected FPGAs and CPUs
- Communication abstracted away from user





# Conclusion

- Network connected FPGAs/CPUs are more scalable than traditional PCIe
- Creation of larger fabrics with network connected FPGAs opens door for more complex algorithms
- Many opportunities to explore in multi-FPGA ML