KServe inference extension for an FPGA vendor-free ecosystem

Diego Ciangottini **

On behalf of the development team

The team:

- Giulio Bianchini *
- Mirko Mariotti *
- Daniele Spiga **
- Loriano Storchi **,***
- Giacomo Surace **

* A.D. 1308 unipg UNIVERSITÀ DEGLI STUDI DI PERUGIA

** INFN Sezione di Perugia

*** UdA Università degli Studi G. d'Annunzio Chieti Pescara

Foreword

The use case

- Machine Learning remote inference on FPGA
  - a first endeavour where the presented activity has been initiated
  - nevertheless it is hopefully extensible to more generic use cases

“Can we think of a way to manage a declarative access to FPGA resources?”

- ***Abstracting the vendor-specific parts of firmware***
- ***Automatically synthesize FPGA firmware from a generic ML model***

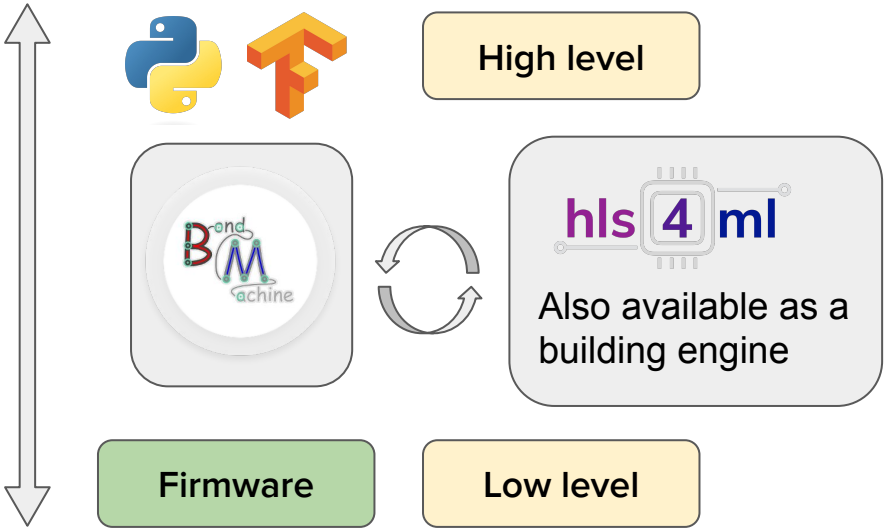
Result of the **integration of two in-house expertise domains**:

- FPGA programming
  - Next slide - building and loading firmware in a vendor-agnostic way
- Cloud-native solutions for scientific computing at INFN
  - Provisioning UI for data analysis on demand

ML Inference on FPGA

Implement customized and parallel architectures tailored to specific ML models, resulting in **faster processing speeds** and **lower power consumption** compared to traditional CPUs or GPUs.

Starting from high-level code and standard ML framework, with HLS tools like **BondMachine (next slide) and hls4ml**, get the firmware implementations of machine learning algorithms



The machine learning model is **trained with standard frameworks** and **synthesized in FPGA** as a graph of heterogeneous and interconnected processors.

- **Optimized resource usage**
  - lower resource usage handling the numerical precision as needed
- **Vendor independent**
  - build firmware for different boards and different vendors
- **User-friendly**
  - automation mechanisms make firmware synthesis easy and accessible

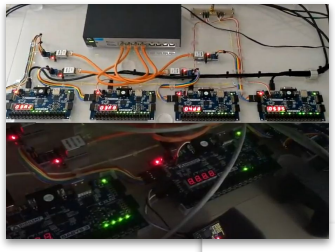
D.Ciangottini - ciangottini@infn.it - CHEP2023

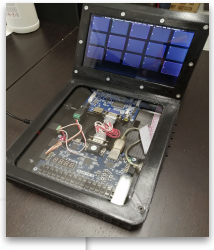
Building firmwares experience: some references

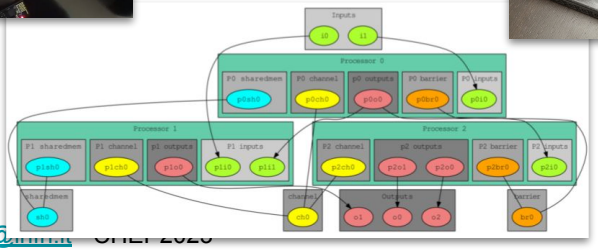
The BondMachine is an **open source** (https://github.com/BondMachineHQ) software ecosystem for the dynamical generation of computer architectures that can be synthesized on FPGA.

- **High level programming language** (Golang) for both the hardware and software
- Computational graph and **Machine Learning Models synthesis**

http://bondmachine.fisica.unipg.it/

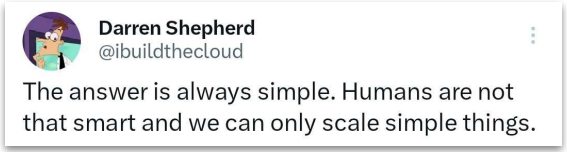






History and Major Highlight

- **InnovateFPGA 2018 Iron Award, Grand Final at Intel Campus (CA) USA**
- Invited lectures at FPGA workshops ICTP 2019 and 2022
- Golab 2018 talk and ISGC 2019 PoS
- **Article published on Parallel Computing, Elsevier 2022 DOI:10.22323/1.351.0020**



Bring it to cloud level: why?

So we “know” how to build firmware for ML inference in a vendor agnostic way. Can we **integrate it with cloud-native inference as-a-service** solution to get any advantage?

- **Ease of usage and flexibility**
  - Being able to deploy an inference algorithm on FPGA without caring for “where” the resources are
  - Accessing ML predictions from a remote computing resource **without having in place any specialized hardware or software piece**
    - At the cost of increased latency → to be carefully evaluated case by case
  - Sharing the access to the same model predictions with other collaborators
- **Democratic access and management**
  - Leveraging cloud/k8s native tools, you can reuse a well established way to orchestrate the bookkeeping and distribution of the payloads
- **Easy Prototyping**
  - Automation of the build and load process -> the framework take care of vendor specific details

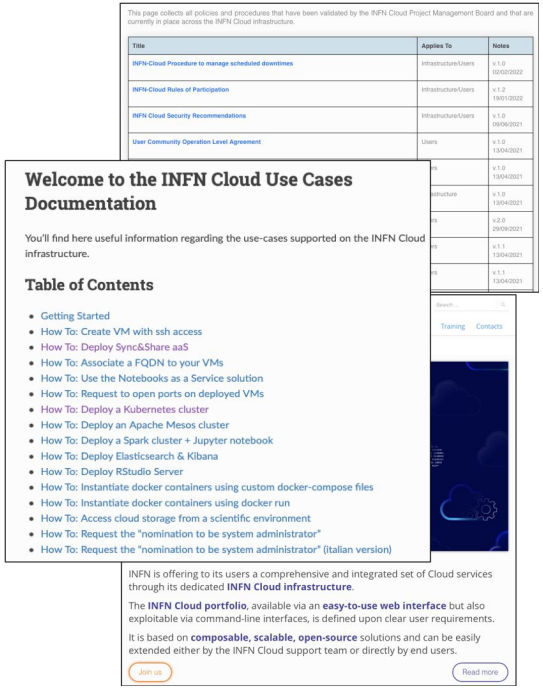
Bring it to cloud level: how?

INFN Cloud, https://www.cloud.infn.it/

- In **production** since March 2021.
- The **initial seed** of a National Datalake for research and beyond, building on (existing|renewed|new) e-Infrastructures.
- The **base of the evolution** of the INFN Distributed Computing vision.
- Built on a **thin middleware layer** running on top of *federated clouds*, decoupling physical and logical views via a **service composition** mechanism.
- The **INFN foundation** for the NRRP computing-related initiatives

The idea is to put our experience at INFN to good use:

- **Cloud tools experience at INFN Cloud**
  - In particular on automation and integration of services on Kubernetes
- The remote inference still an open field on many aspects, regardless we **started from one of the main emerging ecosystems for ML → Kubeflow**
  - **KServe** in particular is the component responsible for providing inference endpoint as-a-Service

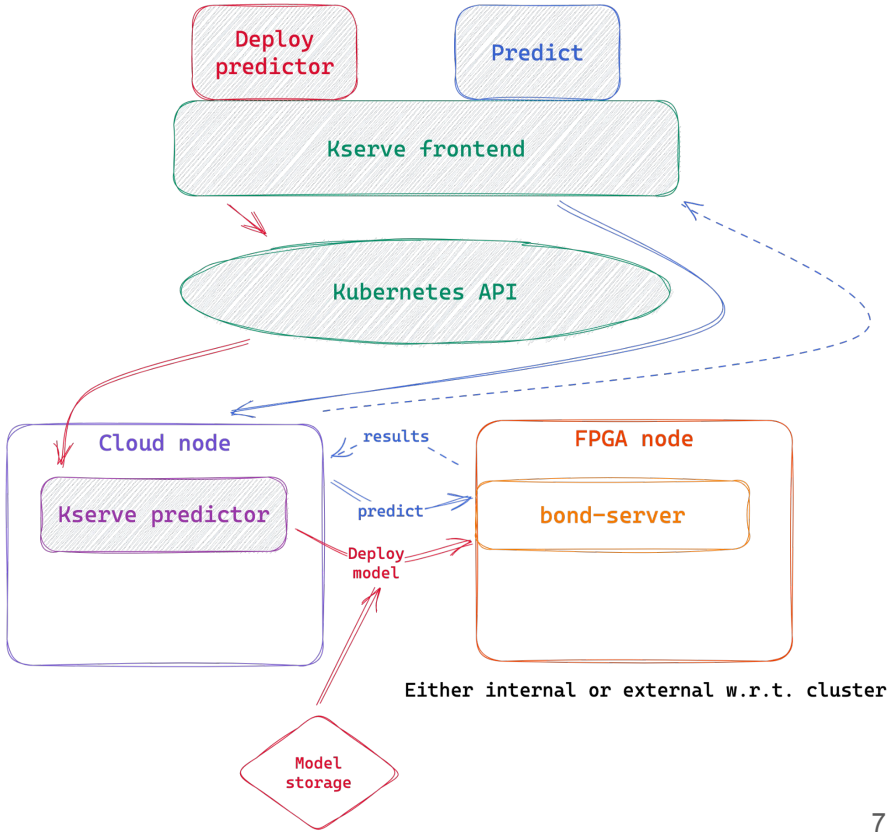


Kubeflow

Implementing a KServe FPGA extension

We started with a simple workflow in mind:

1. **Train your model** with your preferred framework (e.g. TF)
2. **Store the model** on a remote storage
   a. S3 storage is the one used for our tests
3. Deploying the **same model on a remote FPGA via a user friendly UI**
4. Get back the **details of the endpoint to interact with**
   a. Either via HTTP or grpc protocols

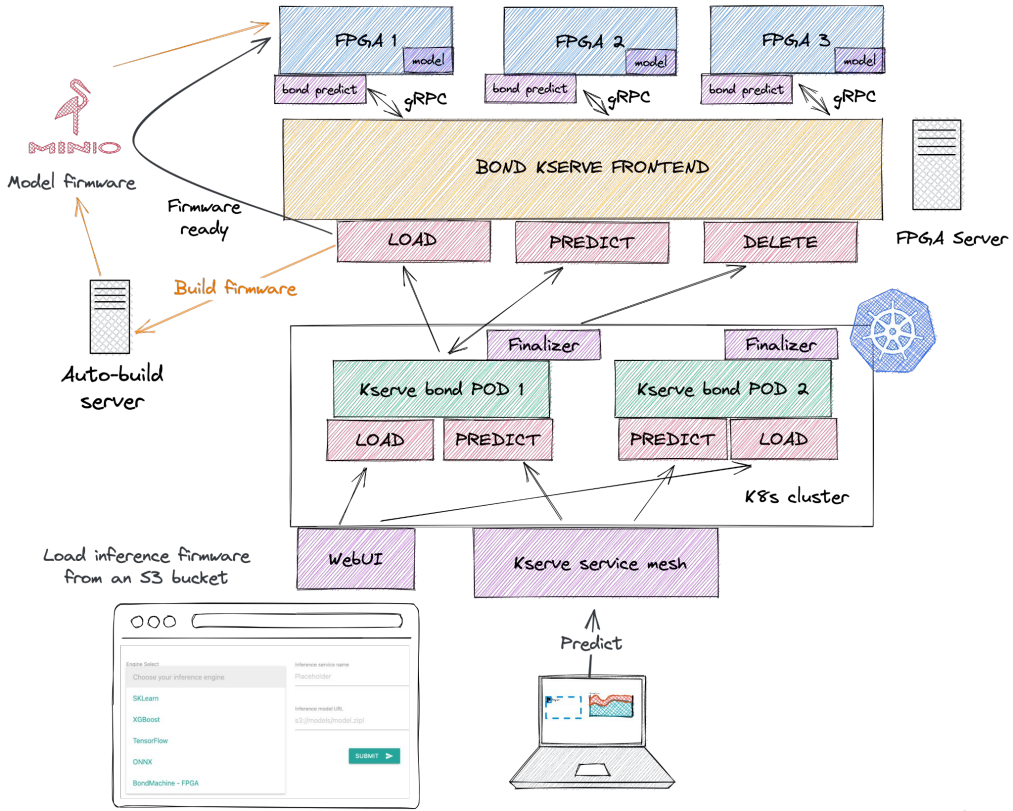


Kserve extension implementation

We tested workflows for both small and ML board (ebaz4205, zedboard, alveo u50)

The main components that we developed are:

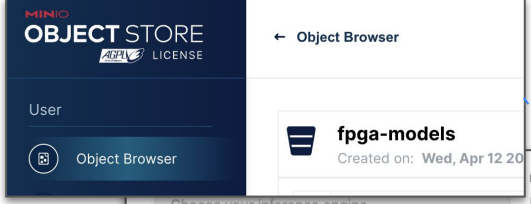
- **Custom WebUI** to hide complexity to the user
  - A Kubeflow managed solution exists, we are planning to integrate this work eventually
    - We need additional metadata to be passed (e.g. board model, provider, hls engine etc)
- Translate a **model load** request into conditional actions
  - Load the bitstream file from the remote location directly
    - Pre built by the user on its own
  - **building a firmware** “seamlessly” on an external building machine
- **Eventually load the firmware** on the FPGA board via the development of a grpc server installed on the machine that have access to the board

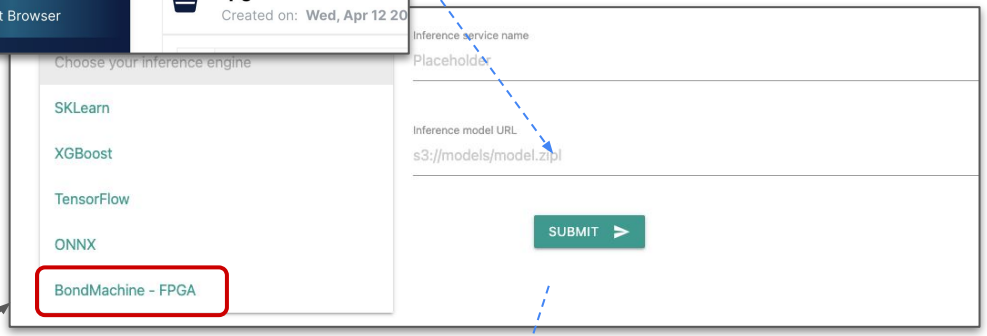


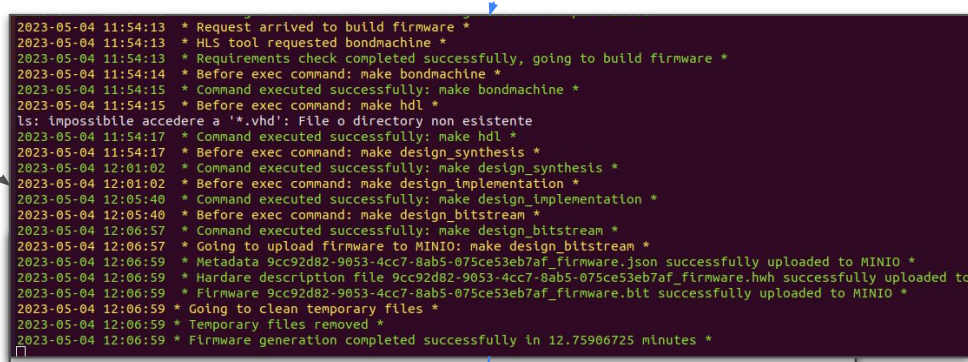
Where are we…

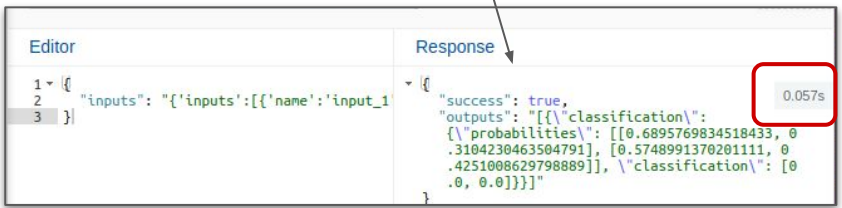
We have **validated an end to end workflow with a generic ML algorithm.** With the following steps:

1. Load the model description to an S3 bucket
2. Report the model URL and name in the WebUI
   a. Selecting HLS engine (BM in this case)
3. Wait for the build server to build and store your firmware for the available FPGAs
   a. Store back the firmware on S3 bucket for further reuse
   b. Load the created firmware on a FPGA
4. Publish the endpoint to send the prediction requests to and then do your prediction.









SERVICE_TYPE	API_VERSION	INFERENCE_SERVICE_NAME	SERVICE_HOSTNAME	MODEL URL
fpga-model	v1	test01	test01.default.fpga.infn.it	ghcr.io/bondmachinehq/bond-server

What's next?

1. **FPGA bookkeeping** (at K8s level) is one of the high priority target in the development
   a. Essentially make use of existing device plugins
2. Also, this work is enabling an approach to **firmware building and management via CLI**
   a. Docker-like hopefully?
      i. *bond build mymodel.json --model XXX --produced xilinx -t dciangot/mymodel:v2*
3. Definition of an OCI artifact spec allowing for storing model on compatible container registries:
   a. *bond push ghcr.io/dciangot/mymodel:v2*
4. Although not in the initial target, can we think of **leveraging the kserve extension also for use cases beyond inference?**
   a. E.g. I want to test my algorithm on a FPGA board that is somewhere shared with other people
   b. Loading firmware and spawn jupyterlab container with on the machine with direct access to the pre-programmed FPGA
5. Systematic **measurements of performances at the various stage of the chain**
   a. monitoring inference time at different layers (cycles, FPGA server and predictor machine etc) → often a useful feedback for who is developing the model

Plenty of opportunities for participating! Don't hold back if interested.

https://github.com/BondMachineHQ/kserve-bond-extension

http://bondmachine.fisica.unipg.it/

Discord channel for post talk discussion/setting up a chat.

Backup

Why KServe

- **Kubeflow** has a **wide community and a fairly complete toolset for ML workflows**
  - A Kserve extension will make us compatible with all of that
- **KServe** already support a **variety of prediction engines** (Tensorflow, Pytorch, sklearn, xgboost, ONNX, NVidia etc..)
  - This is an added value, since you have all of your models in a single interface
- Easy **plugin mechanism for the inference service**
  - built with extension in mind
  - literally just a handful of methods to be customized as you need: load, predict, cancel