WP3 Node Architecture and Building Blocks

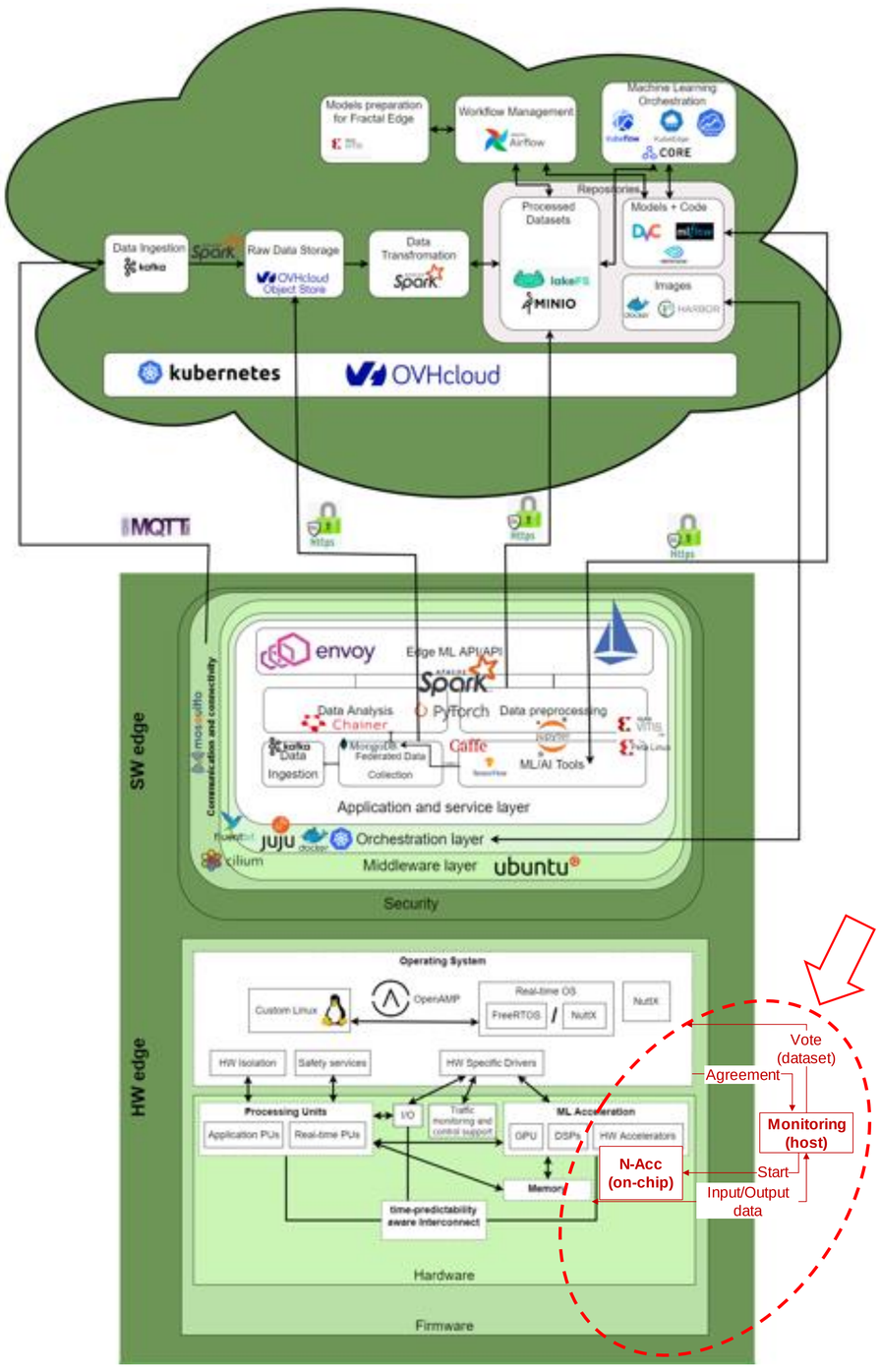
WP3T32-06 Redundant Acceleration Scheme

Developed by: UPV

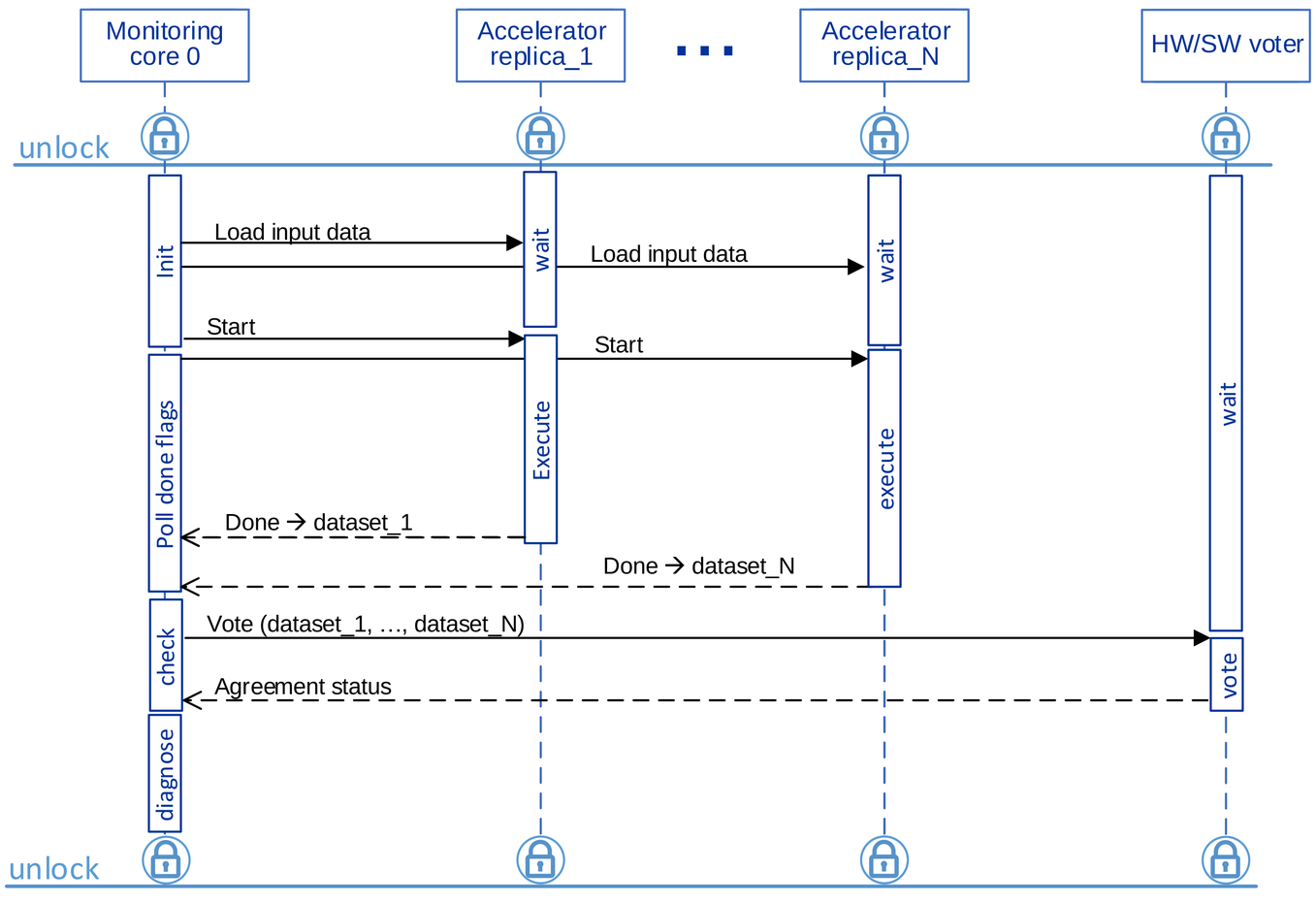
Redundant Accelerators description:

- **Objective:**
  - ✓ Hardware acceleration of a neural network inference processes
  - ✓ Fault-tolerance through N-modular redundancy (replication accelerators IPs)
- **Fractal features associated:** Safety, Redundancy, Self-monitoring
- **Inputs:** Memory-mapped configuration registers and input data buffers
- **Outputs:** AXI master ports can be used to access the main memory and read the output data
- **Integration:** NOEL-V SoC and GRMON debugger
- **Redundant Accelerators approach**: Execute an algorithm on several replicated accelerators, vote result in HW/SW

Component location



Structure and operation of Redundant Accelerators scheeme



Get started

1. Customize configuration file:

```
#define W              256       // Width of the data
#define H              256       // Height of the data
#define I                4       // Number of input channels
#define O                4       // Number of output channels
#define KW               3       // Convolutional kernel width
#define KH               3       // Convolutional kernel height
#define PT_SIM           1       // Top padding
#define PB_SIM           1       // Bottom padding
#define PL_SIM           1       // Left padding
#define PR_SIM           1       // Right padding
#define SH               1       // Vertical stride
#define SW               1       // Horizontal stride
#define USE_RELU         1       // relu activation functions
#define USE_STM          1       // STM functions
#define USE_MAXPOOLING   1       // enables the maxpooling layer
#define USE_AVPOOLING    1       // enables the avpooling layer
#define USE_BATCH_NORM   1       // enables applying batch normalization
#define USE_ADD          1       // enables add module
#define USE_CLIPPING     1       // enables applying clipping to the output
#define USE_SHIFT        1       // enables applying shift to the output
#define USE_UPSIZE       1       // enables upsize (resize)
```

2. Run bare metal test (host):

```
| launching kernel 0 (output iterations 0 to 0)                                                                                                  |
==================================================================================================================================================

Allocate memory buffers
Set configuration parameters

==================================================================================================================================================
| In: 256 x 256 x   4 | Out: 256 x 256 x   4 | Kernel:   3 x   3  | Pad (TBLR): 1 x 1 x 1 x 1 | Stride:   1 x   1  |
|------------------------------------------------------------------------------------------------------------------------------------------------|
| ReLU: No  | STM: No  | MaxP: N | AvgP: N | BN: N |Add: N | Clip: N ( 0: 0) | Shift: N (LEFT , 0) | Upsize N|
==================================================================================================================================================

Start accelerator replica 1
Start accelerator replica 2
Start accelerator replica 3

Polling accelerators
Ended polling

Check results

====> SUCCESS
    *** *** *** ***
    Results are good
    *** *** *** ***
```


EU2020 Horizon **Project N.877056**

ECSEL Joint Undertaking
Electronic Components and Systems for European Leadership

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 877056. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Spain, Italy, Austria, Germany, Finland, Switzerland.