KLEPSYDRA
TECHNOLOGIES

RISC-V IN SPACE

14.12.2022

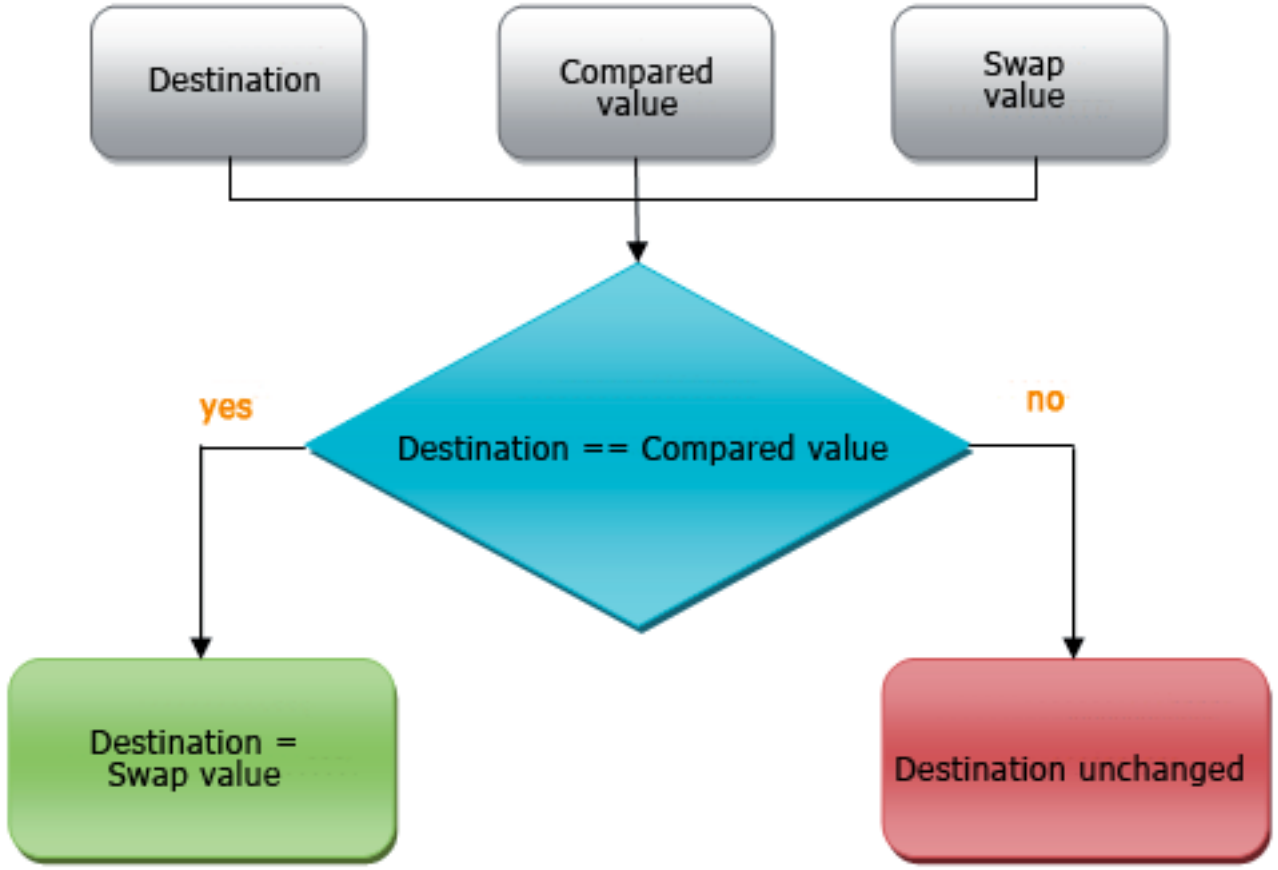
Klepsydra Technologies

pablo.ghiglino@klepsydra.com
www.klepsydra.com

Part 1
Pre-existing work

KLEPSYDRA TECHNOLOGIES

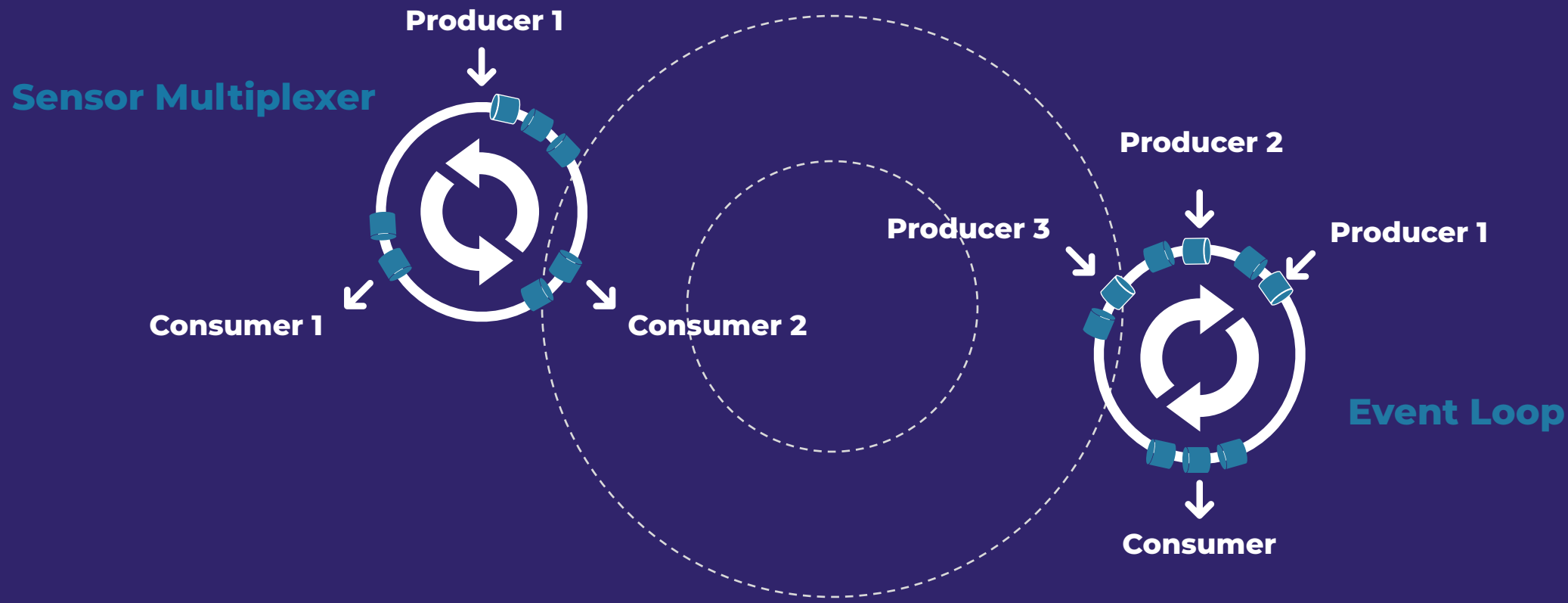
COMPARE AND SWAP



- **Compare-and-swap (CAS)** is an instruction used in multithreading to achieve synchronisation. It compares the contents of a memory location with a given value and, only if they are the same, modifies the contents of that memory location to a new given value. **This is done as a single atomic operation.**
- Compare-and-Swap has been an integral part of the **IBM 370 architectures since 1970**.
- **Maurice Herlihy (1991)** proved that CAS can implement more of these algorithms than **atomic read, write, and fetch-and-add**

KLEPSYDRA TECHNOLOGIES

Two main data processing approaches

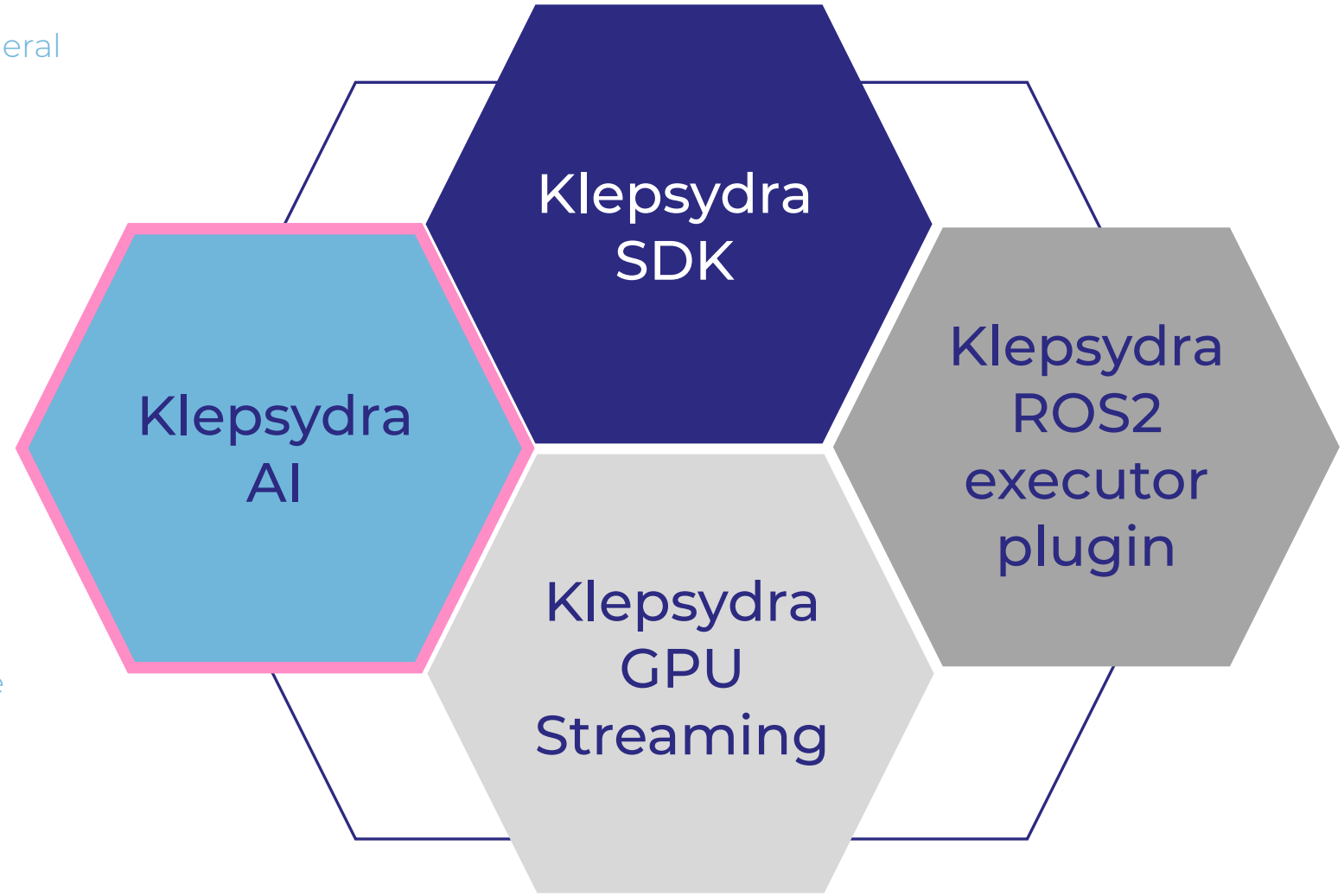


KLEPSYDRA TECHNOLOGIES

THE PRODUCT

Lightweight, modular and compatible with most used operating systems

- SDK – Software Development Kit
  Boost data processing at the edge for general applications and processor intensive algorithms
  - AI – Artificial Intelligence
    High performance deep neural network (DNN) engine to deploy any AI or machine learning module at the edge
    - ROS2 Executor plugin
      Executor for ROS2 able to process up to 10 times more data with up to 50% reduction in CPU consumption.
    - GPU (Graphic Processing Unit)
      High parallelisation of GPU to increase the processing data rate and GPU utilization



KLEPSYDRA TECHNOLOGIES

LOCK-FREE AS ALTERNATIVE TO PARALLELISATION

Parallelisation

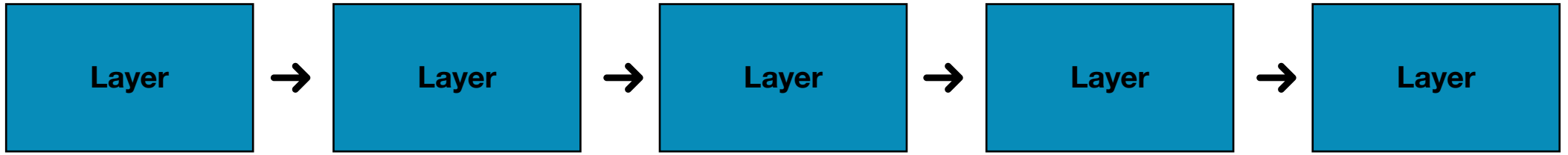
OpenMP

Pipeline

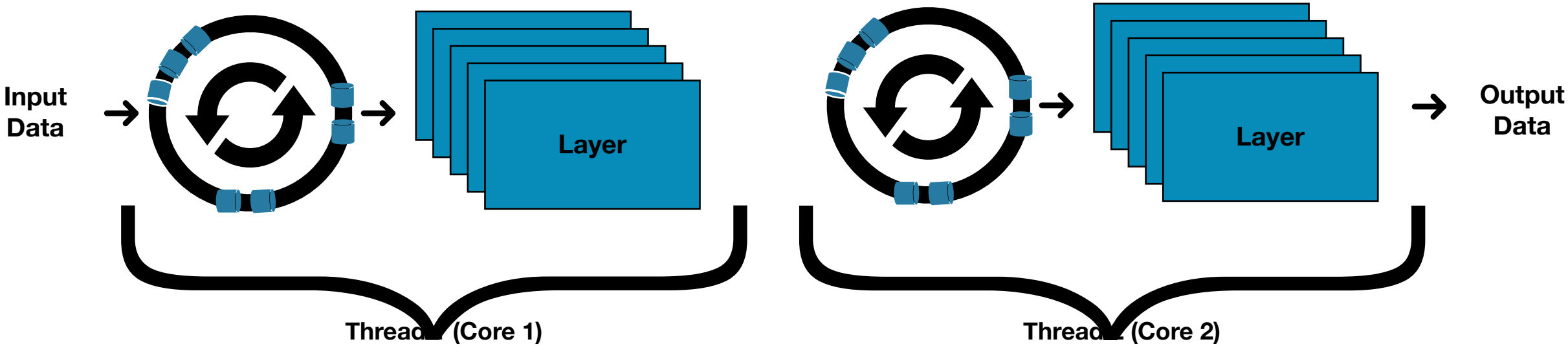
KLEPSYDRA TECHNOLOGIES

2-DIM THREADING MODEL

Deep Neural Network Structure



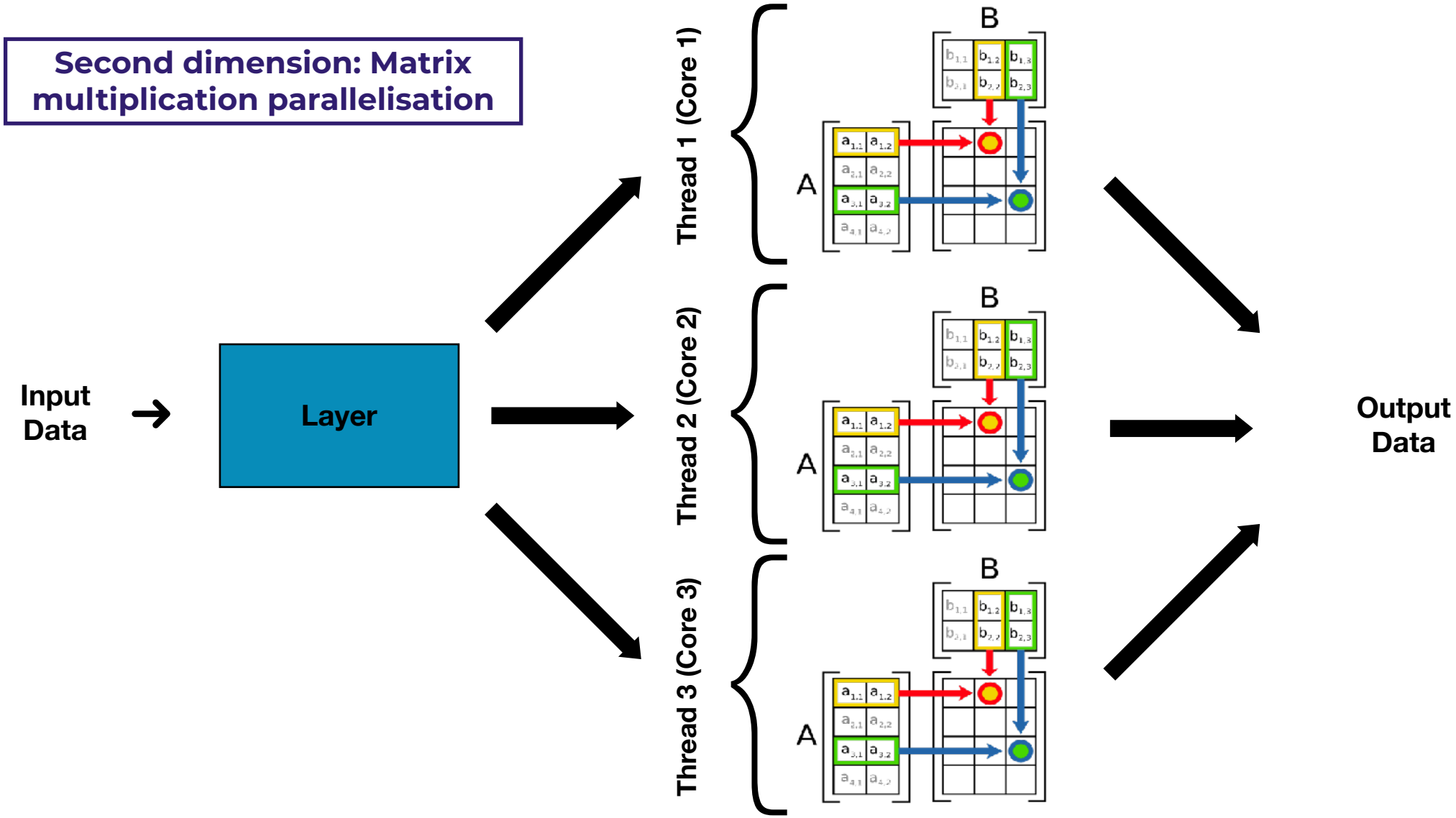
First dimension: pipelining



KLEPSYDRA TECHNOLOGIES

2-DIM THREADING MODEL

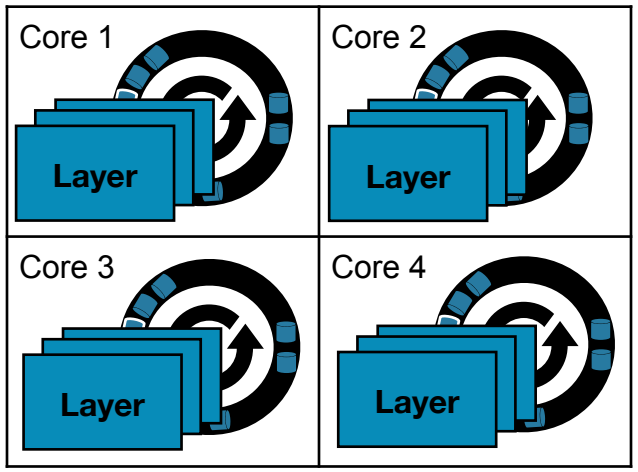
Second dimension: Matrix multiplication parallelisation



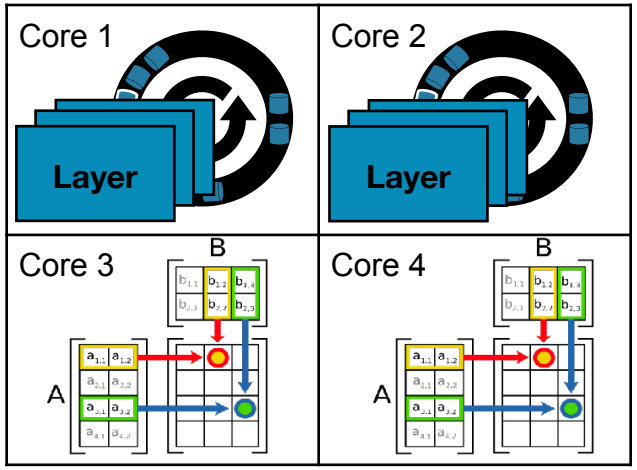
KLEPSYDRA TECHNOLOGIES

2-DIM THREADING MODEL

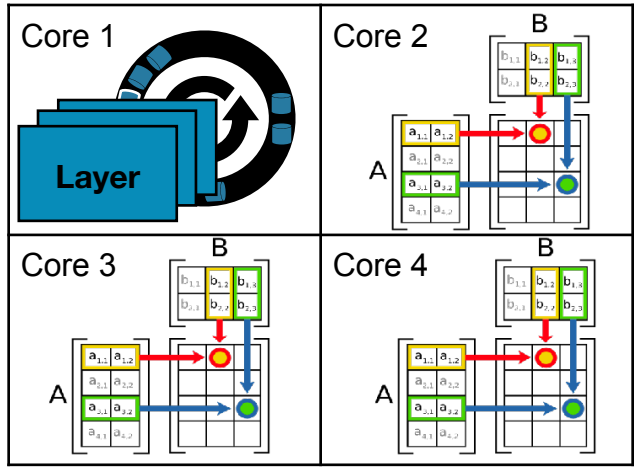
Threading model configuration



- **Low CPU**
- **High throughput CPU**
- **High latency**



- **Mid CPU**
- **Mid throughput CPU**
- **Mid latency**



- **High CPU**
- **Mid throughput CPU**
- **Low latency**

KLEPSYDRA TECHNOLOGIES

ONNX API

```
class KPSR_API OnnxDNNImporter
{
public:

    /**
     * @brief import an onnx file and uses a default eventloop factory for all processor cores
     * @param onnxFileName
     * @param testDNN
     * @return a share pointer to a DeepNeuralNetwork object
     *
     * When log level is debug, dumps the YAML configuration of the default factory.
     * It makes use of all processor cores.
     */
    static std::shared_ptr<kpsr::ai::DeepNeuralNetworkFactory> createDNNFactory(const std::string & onnxFileName,
                                                                                 bool testDNN = false);

    /**
     * @brief importForTest an onnx file and uses a default synchronous factory
     * @param onnxFileName
     * @param envFileName. Klepsydra AI configuration environment file.
     * @return a share pointer to a DeepNeuralNetwork object
     *
     * This method is intented to be used for testing purposes only.
     *
     */
    static std::shared_ptr<kpsr::ai::DeepNeuralNetworkFactory> createDNNFactory(const std::string & onnxFileName,
                                                                                 const std::string & envFileName);
};
```

KLEPSYDRA TECHNOLOGIES

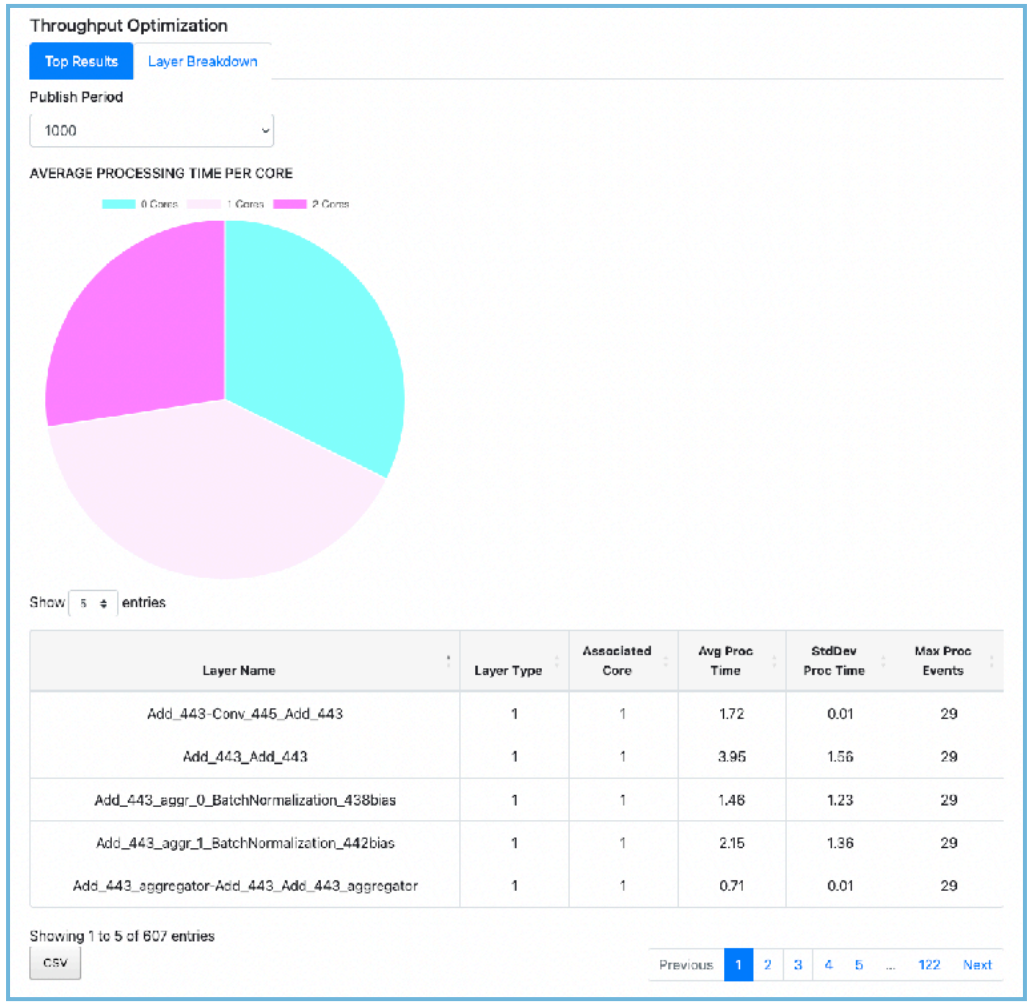
Core API

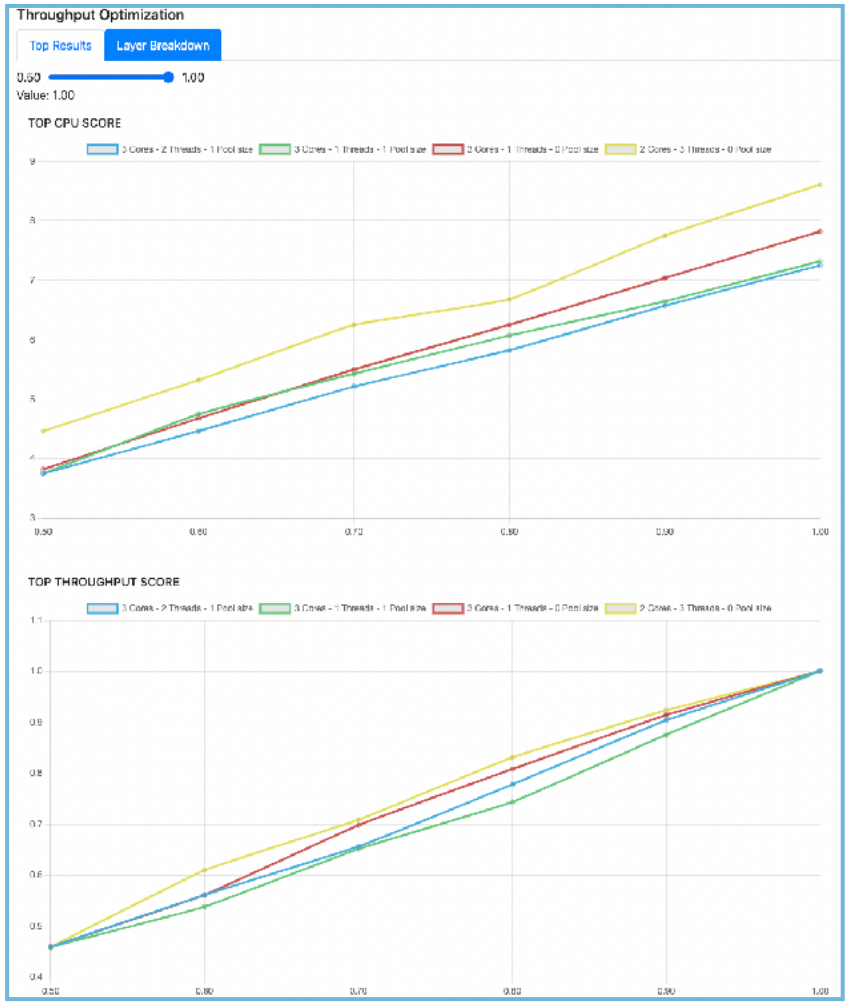
```
class DeepNeuralNetwork {
public:

    /**
     * @brief setCallback
     * @param callback. Callback function for the prediction result.
     */
    virtual void setCallback(std::function<void(const unsigned long &, const kpsr::ai::F32AlignedVector &)> callback) = 0;

    /**
     * @brief predict. Load input matrix as input to network.
     * @param inputVector. An F32AlignedVector of floats containing network input.
     *
     * @return Unique id corresponding to the input vector
     */
    virtual unsigned long predict(const kpsr::ai::F32AlignedVector& inputVector) = 0;

    /**
     * @brief predict. Copy-less version of predict.
     * @param inputVector. An F32AlignedVector of floats containing network input.
     *
     * @return Unique id corresponding to the input vector
     */
    virtual unsigned long predict(const std::shared_ptr<kpsr::ai::F32AlignedVector> & inputVector) = 0;

};
```

KLEPSYDRA SDO PROCESS

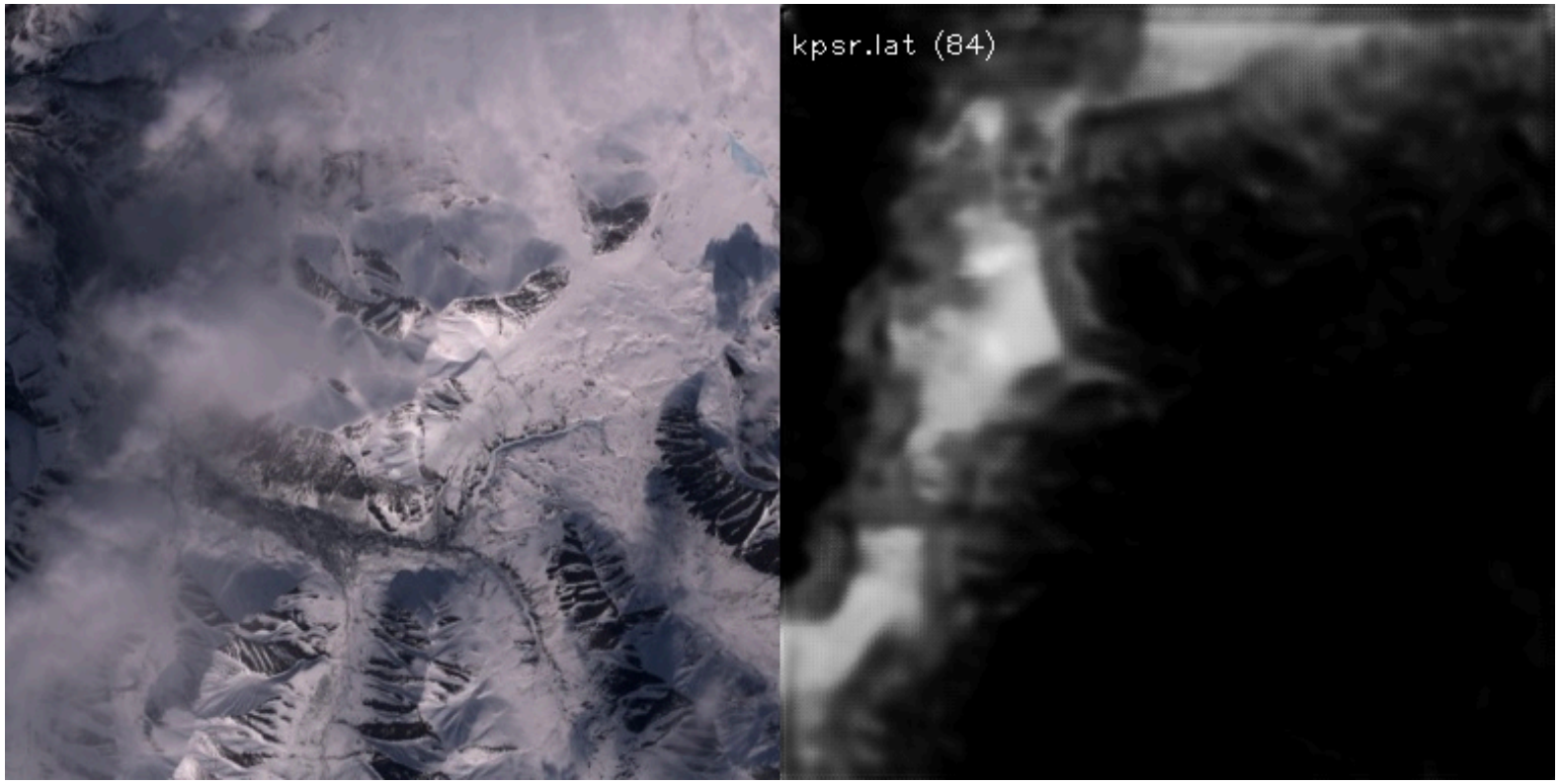
- **Klepsydra Streaming Distribution Optimiser (SDO):**
  - Runs on a separate computer
  - Executes several dry runs on the OBC
  - Collect statistics
  - Runs a genetic algorithm to find the optimal solution for latency, power or throughput
  - The main variable to optimise is the distribution of layers are the two dimension of the threading model

KLEPSYDRA STREAMING DISTRIBUTION OPTIMISER (SDO)

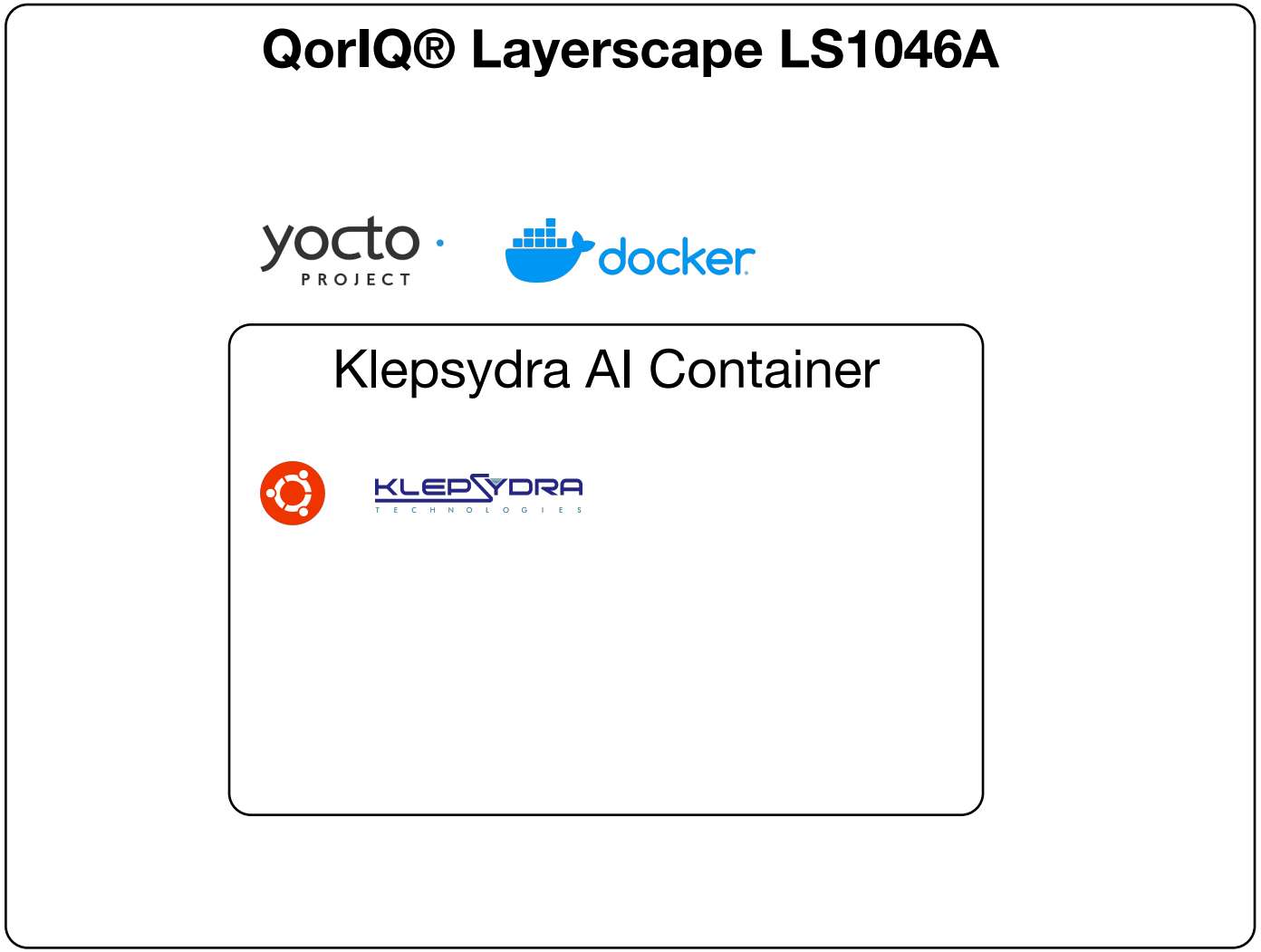




Part 2
The KATESU Project



QORIQ® LAYERSCAPE LS1046A MULTICORE PROCESSOR

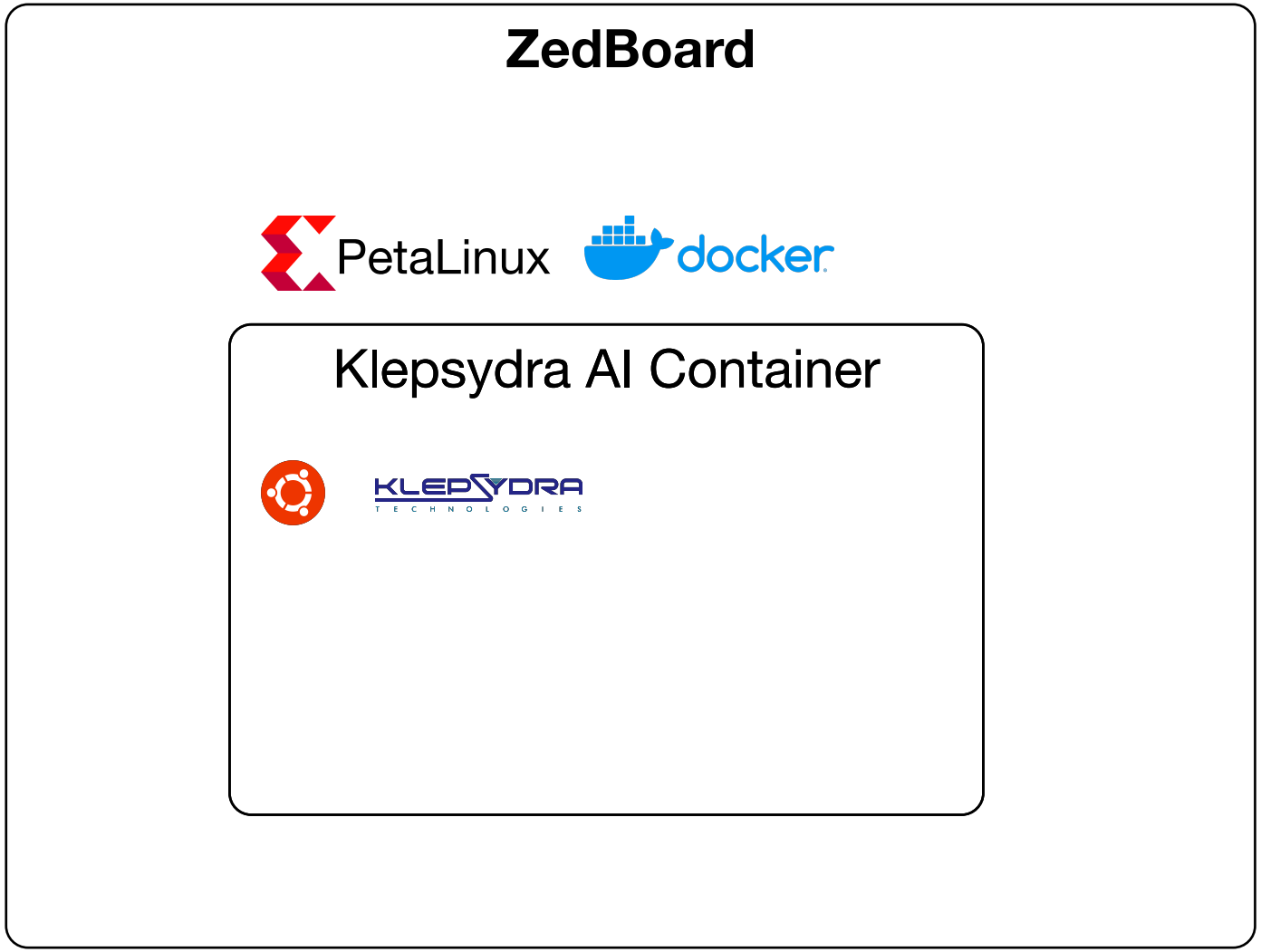


KLEPSYDRA TECHNOLOGIES

STATUS

- **Successful installation of the following setup:**
  - LS1046 running Yocto Jethro
  - Docker Installed on LS1046
  - Container with the following:
    - Ubuntu 20.04
    - Klepsydra AI software fully supported (quantised and non-quantised)

XILINX ZEDBOARD



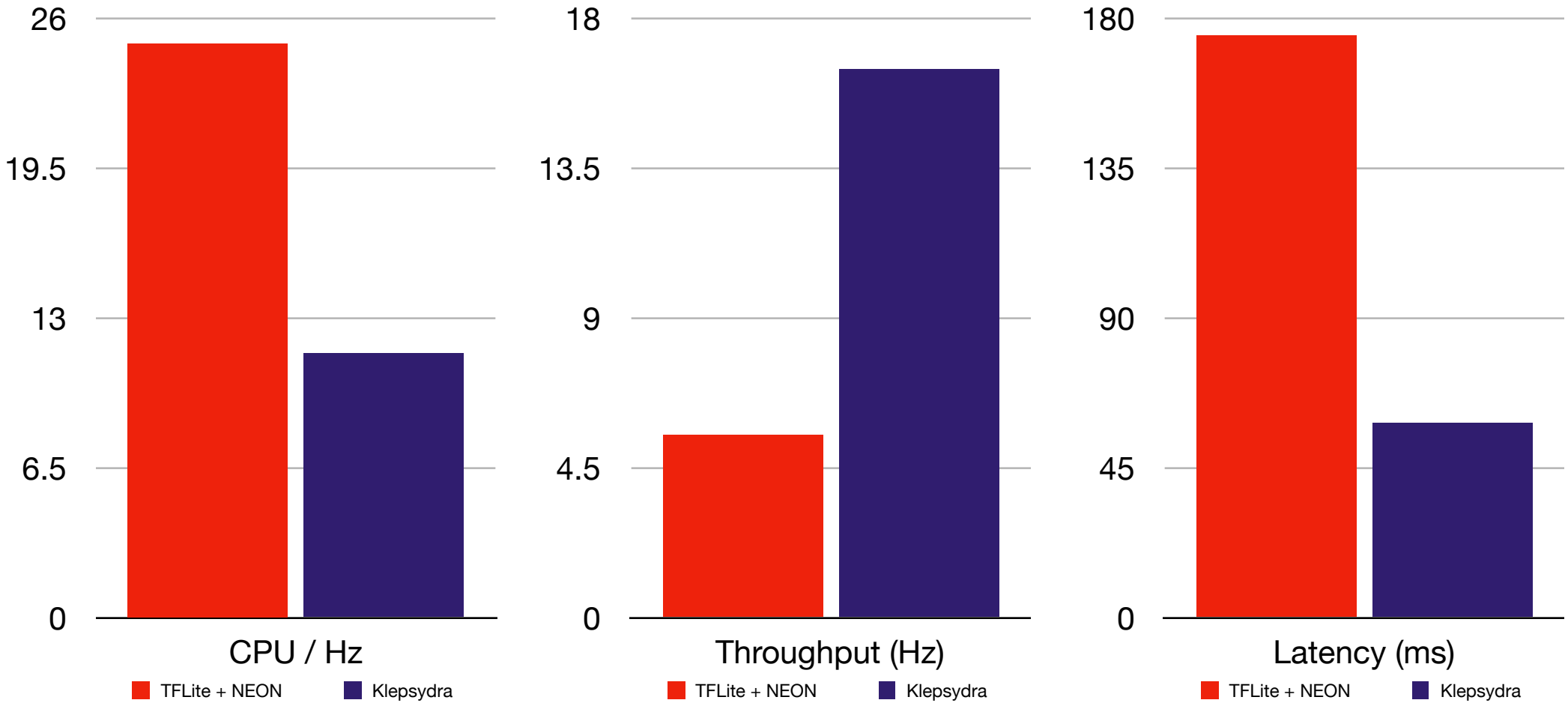
KLEPSYDRA TECHNOLOGIES

STATUS

- **Successful installation of the following setup:**
  - ZedBoard running PetaLinux 2019.2
  - Docker Installed on ZedBoard
  - Container with the following:
    - Ubuntu 20.04
    - Klepsydra AI software with quantised support only

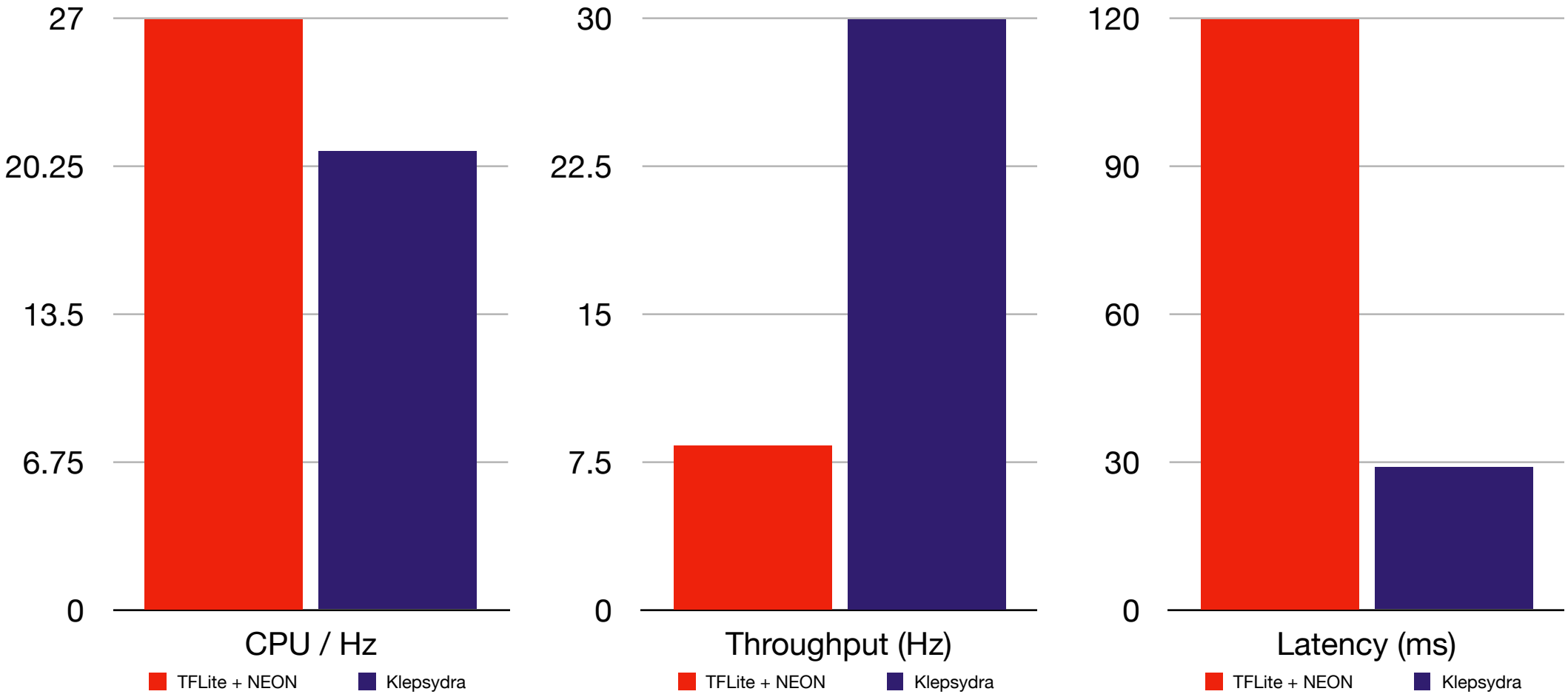
KLEPSYDRA TECHNOLOGIES

PERFORMANCE RESULTS: CME ON LS1046



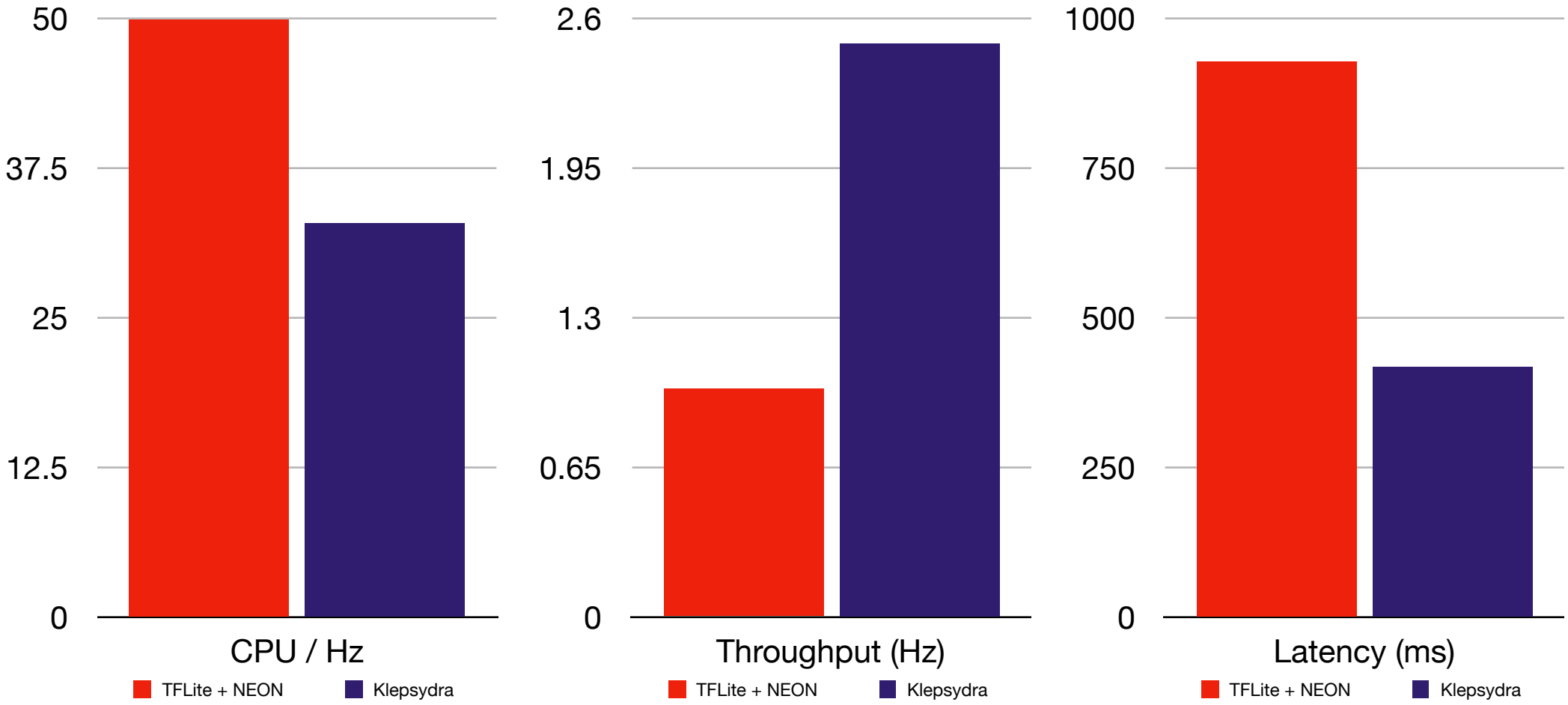
KLEPSYDRA TECHNOLOGIES

PERFORMANCE RESULTS: CME-Q ON LS1046

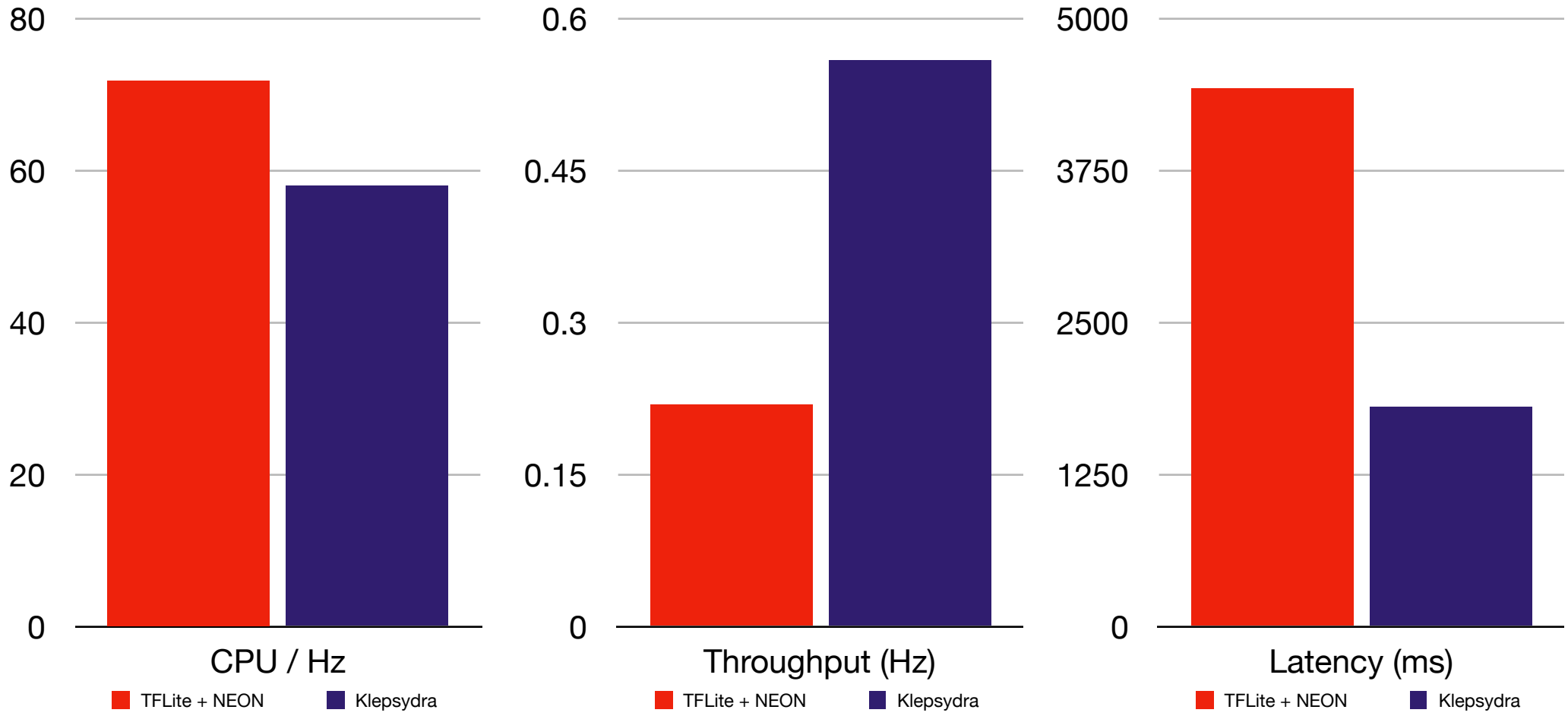


KLEPSYDRA TECHNOLOGIES

PERFORMANCE RESULTS: CME-Q ON ZEDBOARD



PERFORMANCE RESULTS: BSC ON LS1046



Part 2
The PATTERN Project

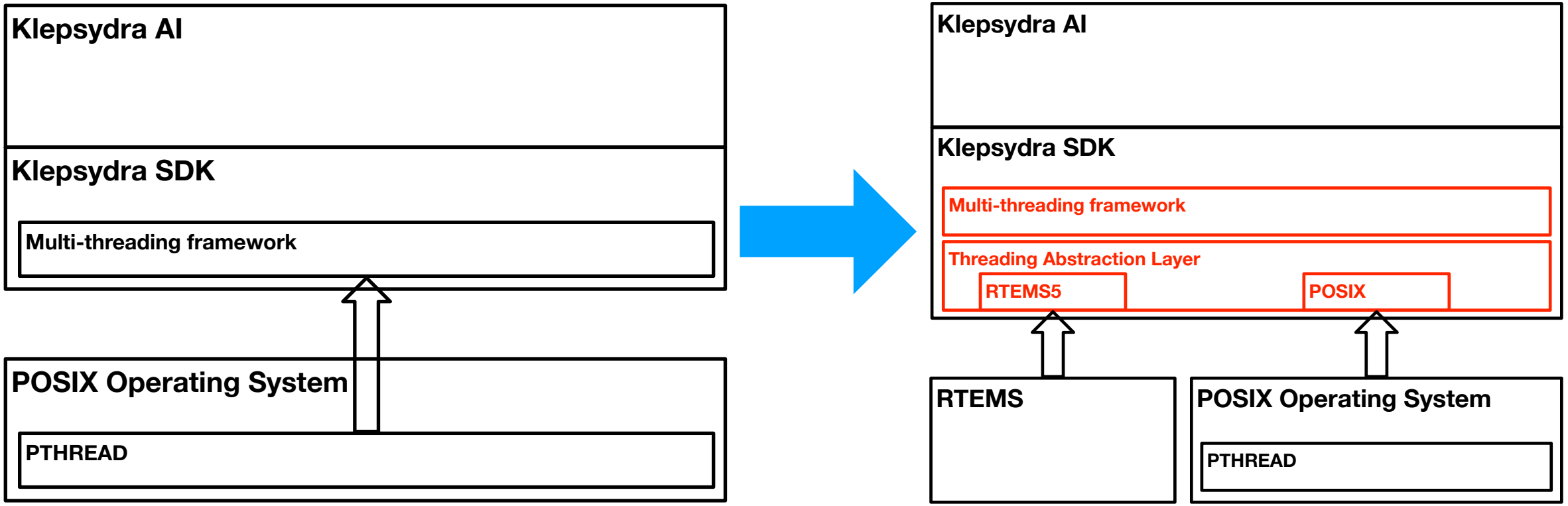
KLEPSYDRA TECHNOLOGIES

THE PATTERN PROJECT

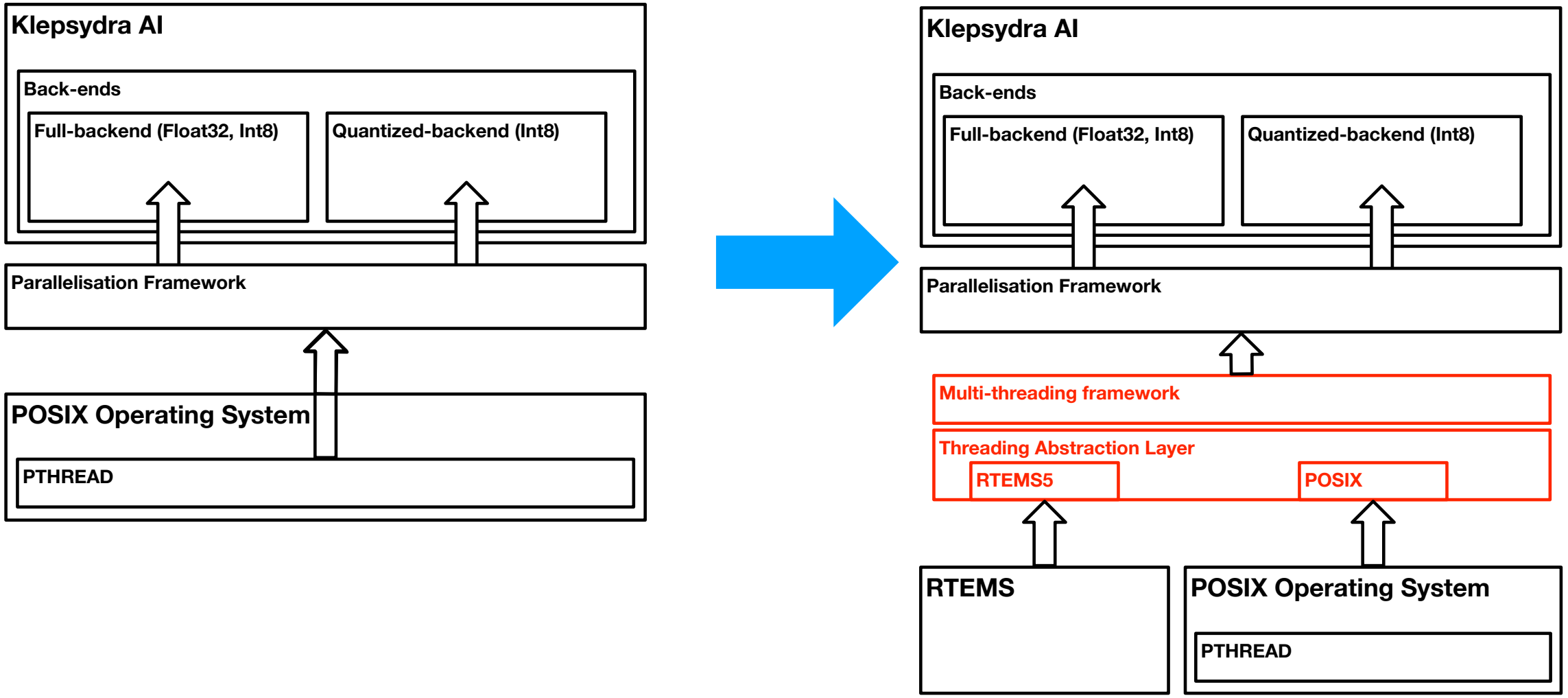
PATTERN: Kle**p**sydra **A**I por**t**ed **t**o th**e** G**R**740 a**N**d RISC-V

- Target Processor: GR740, GR765 (Leon5 & Noel-V)
- Target OS: RTMES5
- Development on commercial FPGA board
- Validation on Space qualified hardware

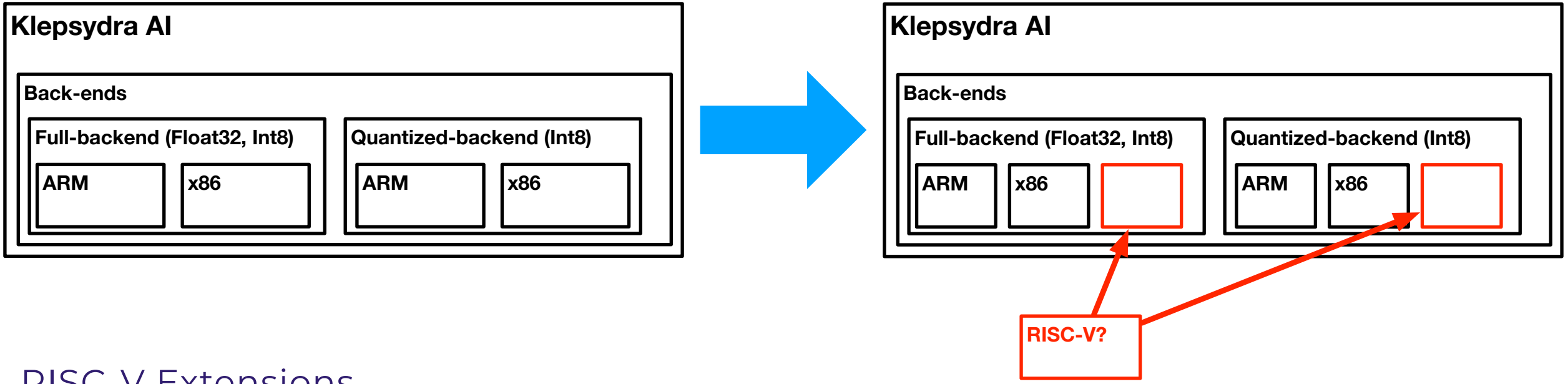
THE MULTI-THREADING API



THE PARALLELISATION FRAMEWORK



THE MATHEMATICAL BACKEND

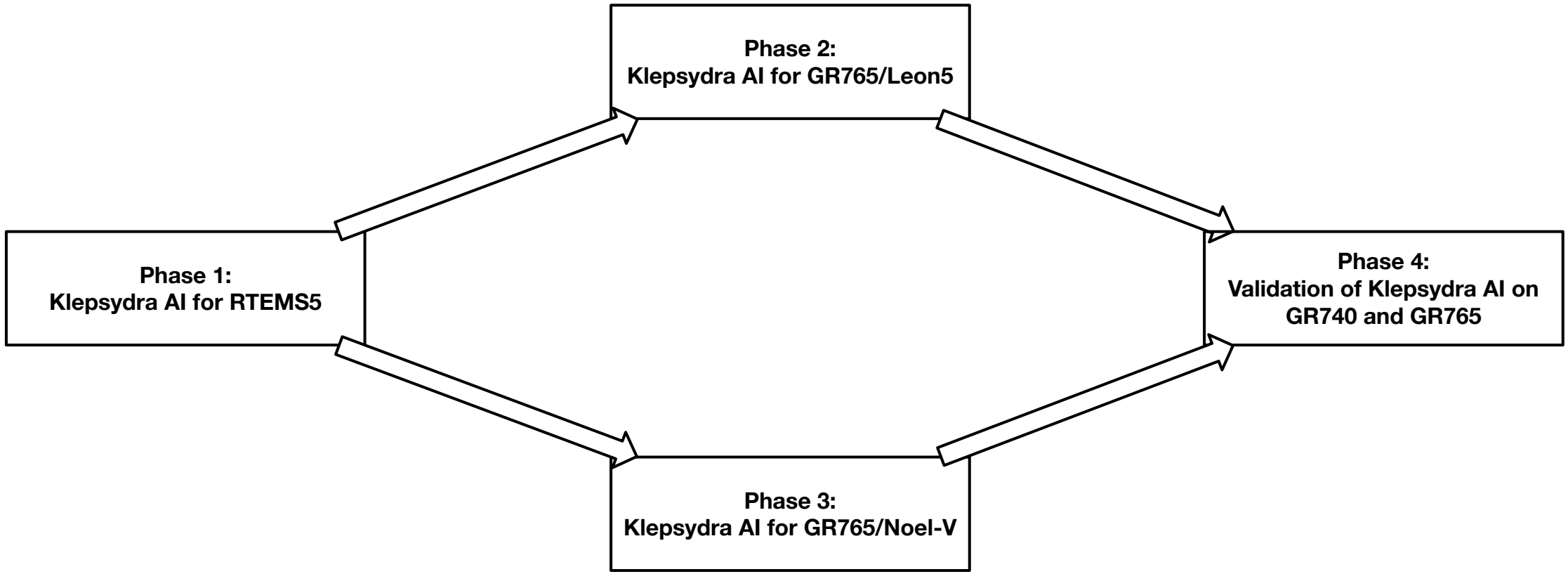


RISC-V Extensions

- Current version of Klepsydra AI supports RV32GV and RV64GC
- Preparation for NOEL-V in three modes:
  - 'Vanilla'
  - P-Extension and V-Extension,
  - And more....

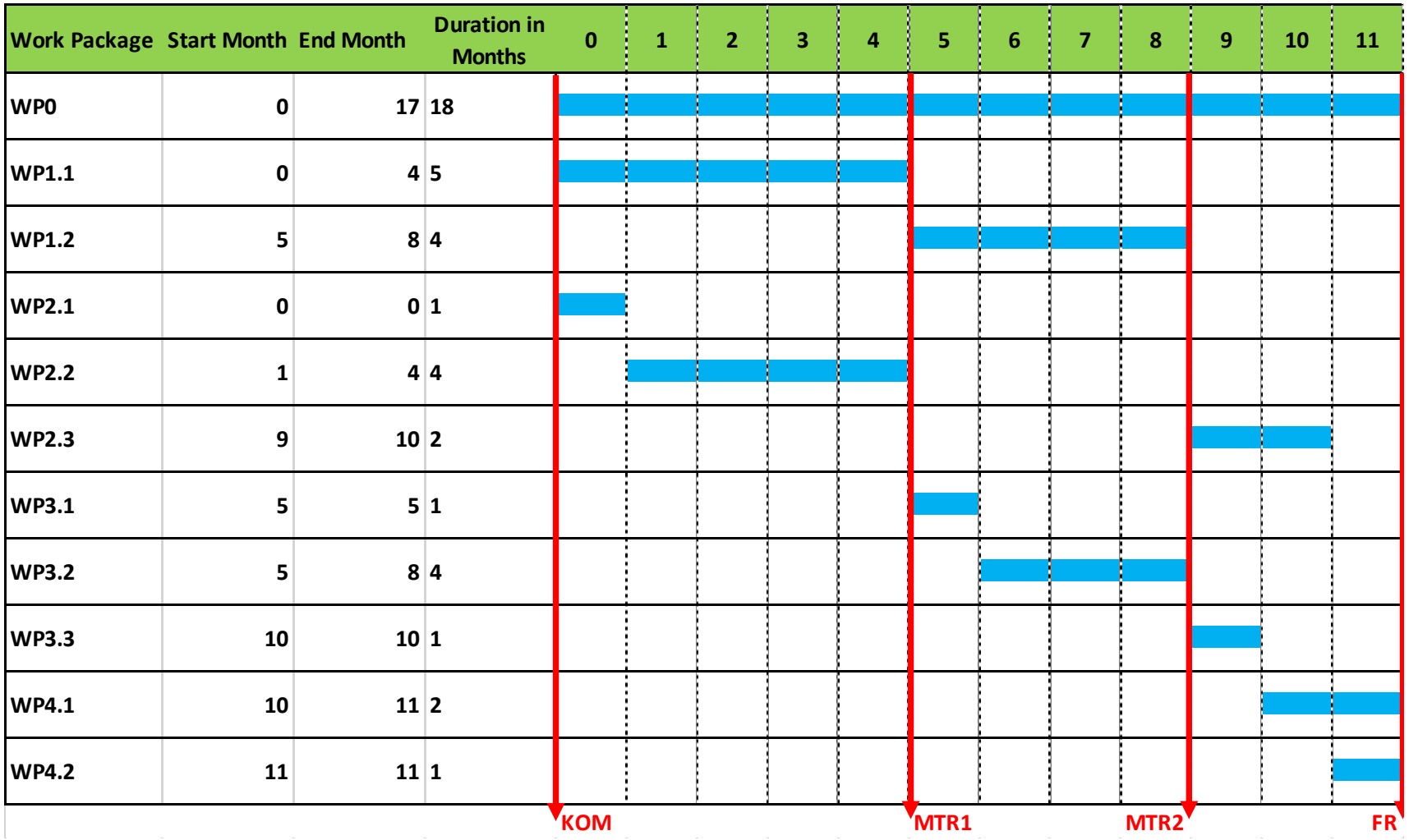
KLEPSYDRA TECHNOLOGIES

THE PLAN



THE SCHEDULE

Work Package	Start Month	End Month	Duration in Months
WP0	0	17	18
WP1.1	0	4	5
WP1.2	5	8	4
WP2.1	0	0	1
WP2.2	1	4	4
WP2.3	9	10	2
WP3.1	5	5	1
WP3.2	5	8	4
WP3.3	10	10	1
WP4.1	10	11	2
WP4.2	11	11	1



KLEPSYDRA TECHNOLOGIES

CONCLUSIONS

- **Enable <u>real</u> AI for future missions on the GR765/NOEL-V**
  - Very easy to use, via a simple API and web-based optimisation tool
  - Highly optimised for the GR765/NOEL-V processors
  - Lightweight software (current version is 4Mb)
  - Deterministic and full control of the dedicated resources

NEXT STEPS

- **In-orbit-demonstration:**
  - OPSSAT OBC: Using Onboard Altera FPGA and NOEL-V softcore
  - Other?
- **Health Monitoring (core operation failures, etc).**

CONTACT INFORMATION

Dr Pablo Ghiglino

pablo.ghiglino@klepsydra.com

+41786931544

www.klepsydra.com

linkedin.com/company/klepsydra-technologies