KLEPSYDRA
TECHNOLOGIES

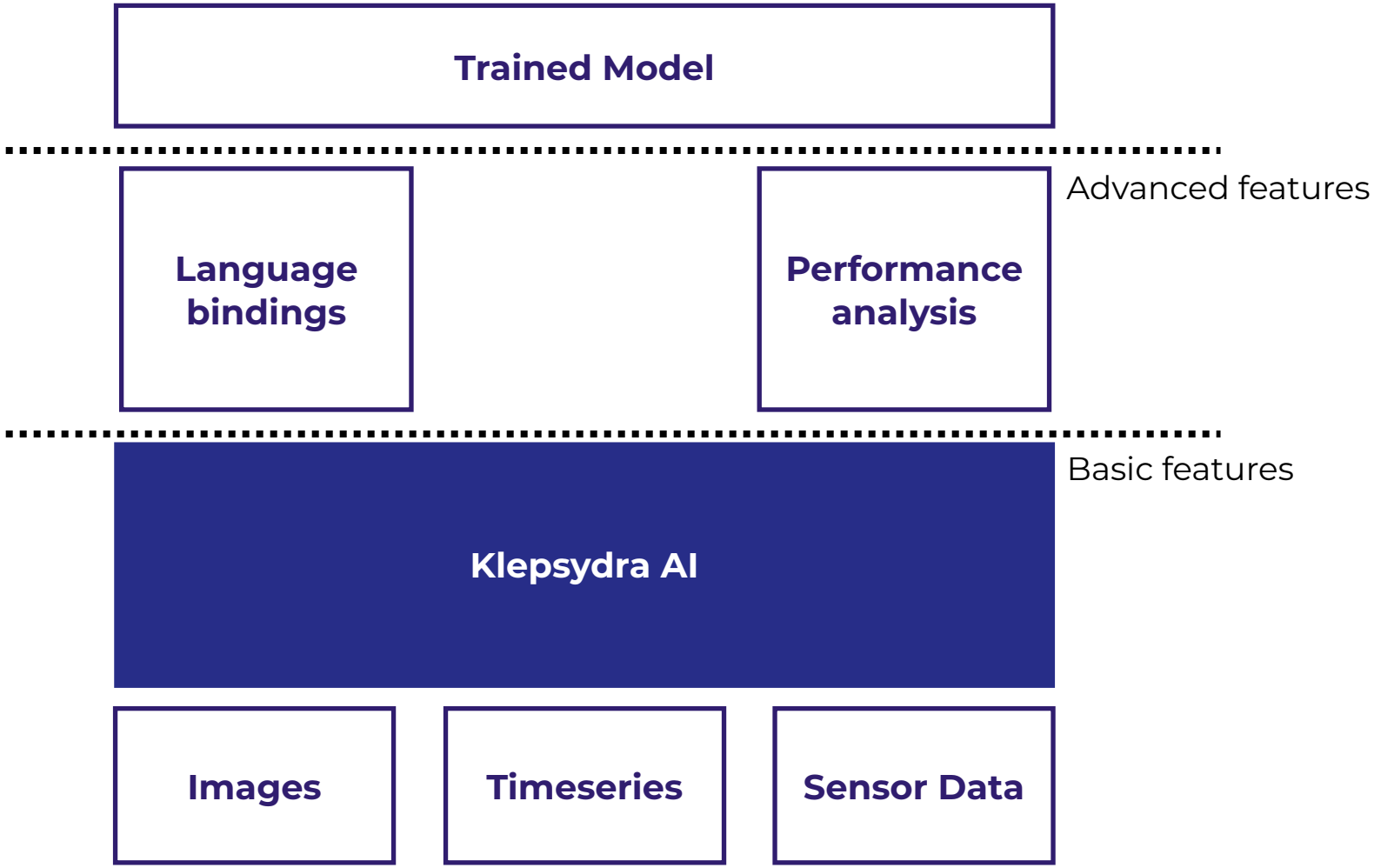
OBDP 2021 PRESENTATION

DR PABLO GHIGLINO

A Low Power And High Performance Artificial Intelligence Approach To Increase Guidance Navigation And Control Robustness

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

KLEPSYDRA AI OVERVIEW



KLEPSYDRA TECHNOLOGIES

KLEPSYDRA AI DISTRIBUTION

1. Developer station distribution:
    - Libraries + header files
    - Development tools (performance analysis, etc.)
    - Unlimited seats.

2. Target computer distribution:
    - Libraries + header files
    - License file per seat.

KLEPSYDRA TECHNOLOGIES

SUPPORT MATRIX

Architecture	Supported Processors
ARM	Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
	Samsung Exynos5 Octa ARM Cortex™-A15 Quad 2Ghz and Cortex™-A7 Quad 1.3GHz CPUs
	64-bit Arm Neoverse core
	6-core Carmel ARM v8.2 64-bit CPU, 8MB L2 + 4MB L3
x86	Intel Xeon E5-2686 v4
	2.4GHz 8-core Intel Core i9
	2.9GHz quad-core Intel Core i7

OS Type	Supported OS
Linux	Ubuntu 18.04, 20.04, Docker, Embedded Linux, FreeRTOS, PykeOS

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 correspoding 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 correspoding to the input vector
     */
    virtual unsigned long predict(const std::shared_ptr<kpsr::ai::F32AlignedVector> & inputVector) = 0;

};
```

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

APPROACHES TO CONCURRENT ALGORITHMIC EXECUTION

Parallelisation

Pipeline

KLEPSYDRA TECHNOLOGIES

BENCHMARK DESCRIPTION

Description

- Given an input matrix, a number of sequential multiplications will be performed:
    - Step 1: A => B = A x A => Step 2 : C = B x B...
    - Matrix A randomly generated on each new sequence

Parameters:

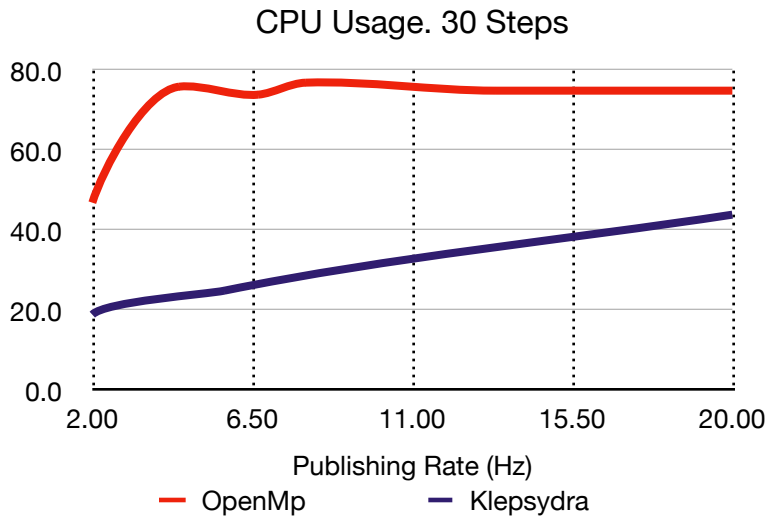
- Matrix dimensions: 100x100
- Data type: Float, integer
- Number of multiplications per matrix: [10, 60]
- Processing frequency: [2Hz - 100Hz]

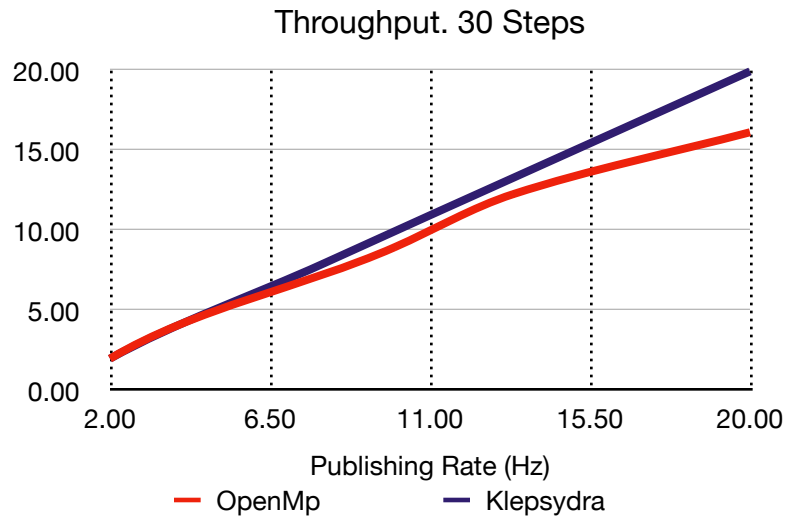
Technical Spec

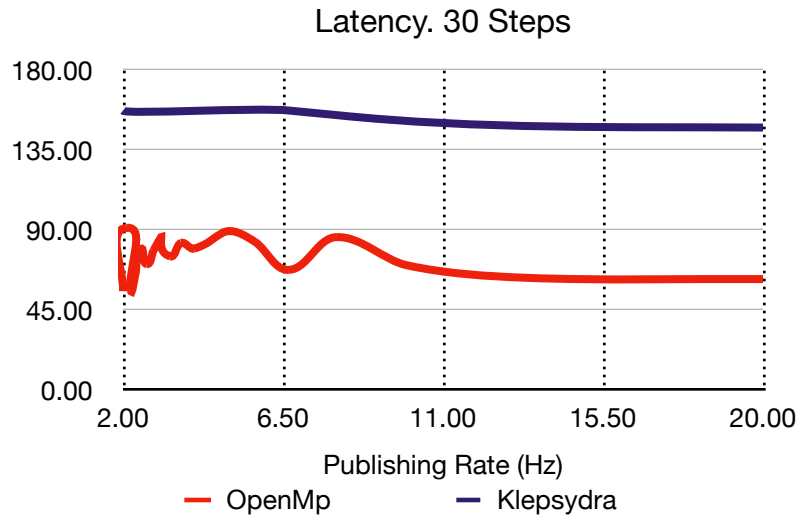
- Computer: Odroid XU4
- OS: Ubuntu 18.04

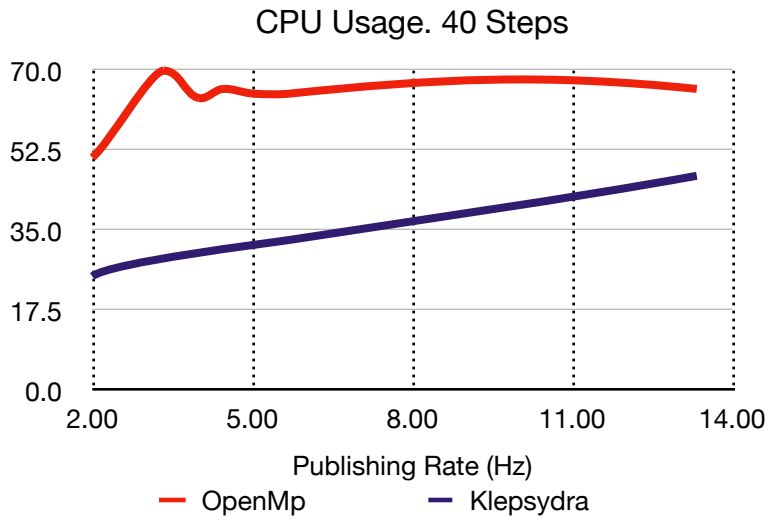
KLEPSYDRA TECHNOLOGIES

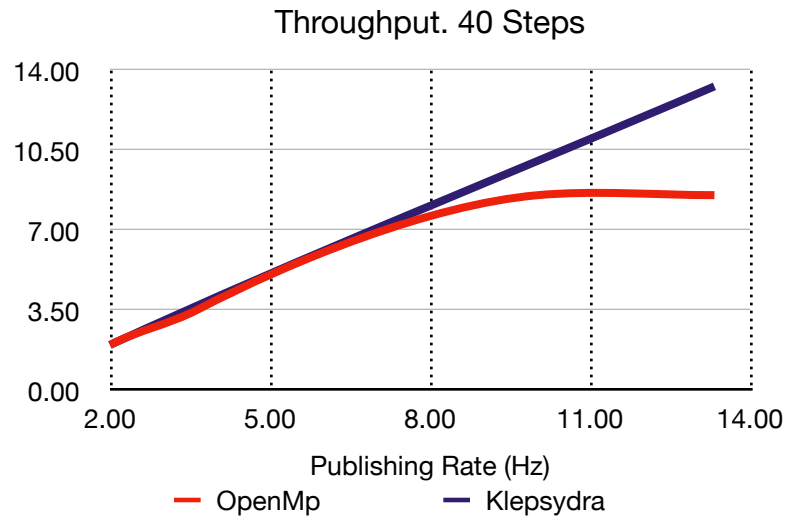
FLOAT PERFORMANCE RESULTS

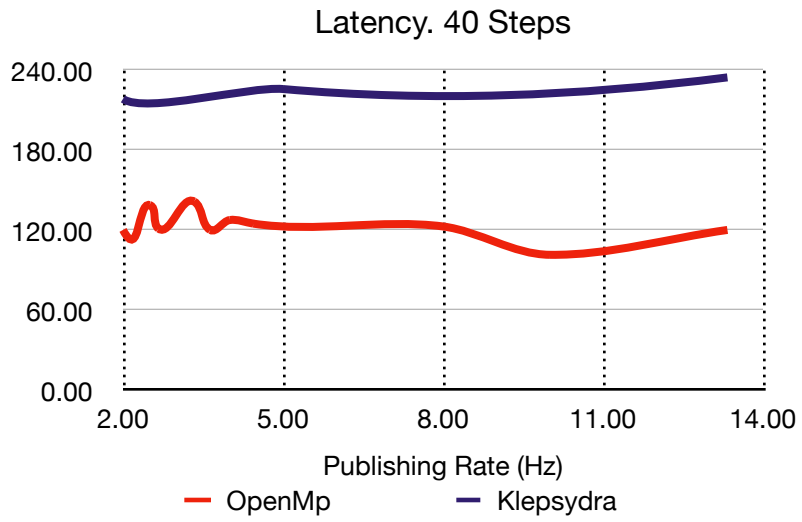






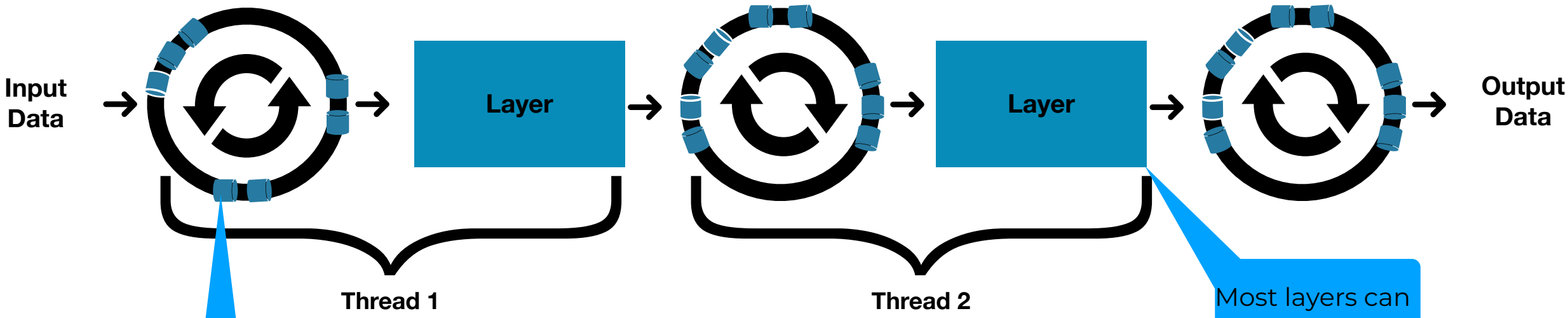






KLEPSYDRA AI DATA PROCESSING APPROACH

Klepsydra AI threading model



Threading model consists of:

- Number of cores assigned to event loops
- Number of event loops per core
- Number of parallelisation threads for each layer

KLEPSYDRA TECHNOLOGIES

Performance tuning

Performance Criteria

- CPU usage
- RAM usage
- Throughput (output data rate)
- Latency

Performance parameters

- `number_of_cores`

Number of cores where event loops will be distributed (by default one event loop per core). High throughput requires more cores, i.e., more CPU usage, low throughput requires low number of cores, therefore **substantial** reduction in CPU usage.

Performance parameters:

- `pool_size`

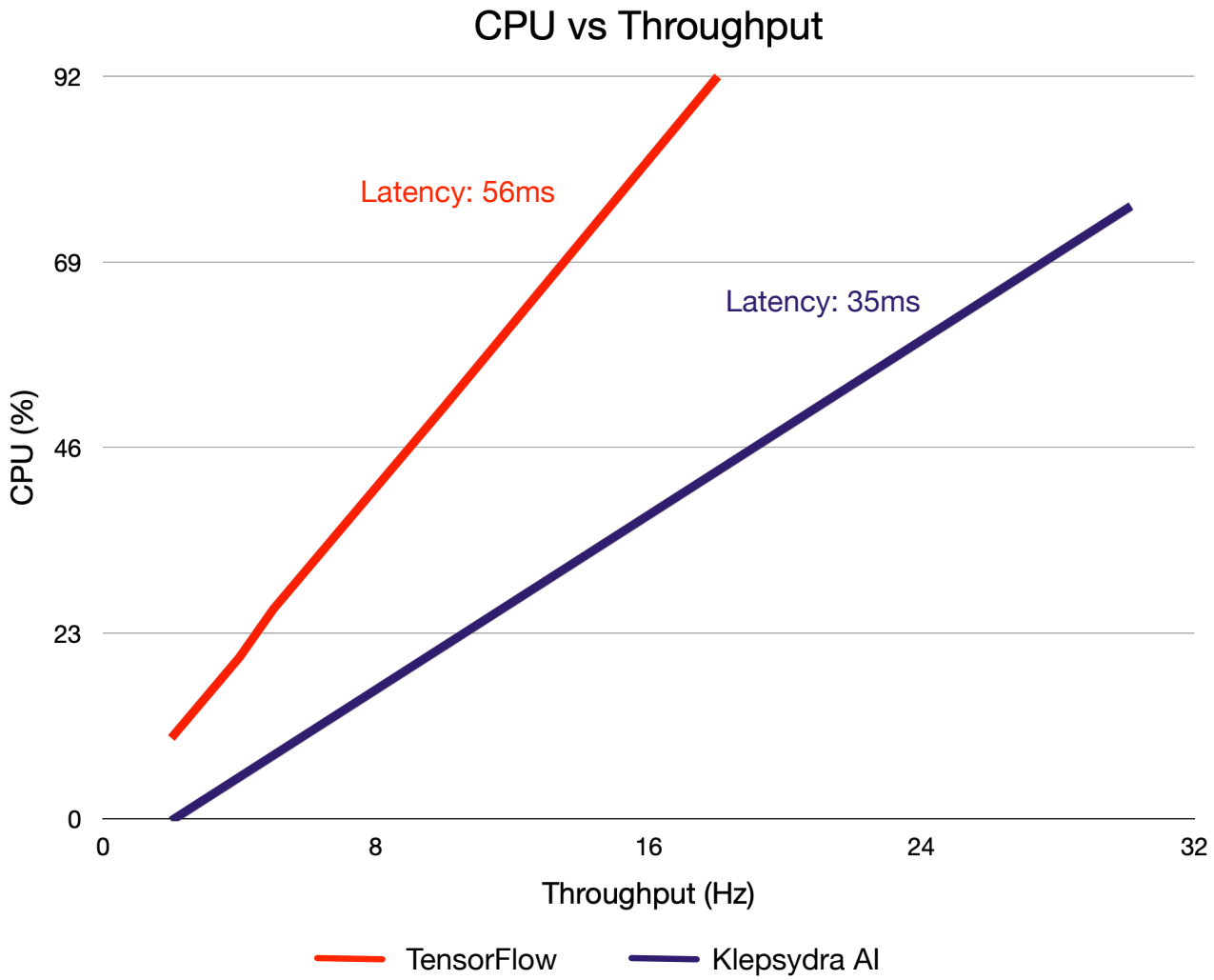
Size of the internal queues of the event loop publish/ subscribe pairs.
High throughput requires large numbers, i.e., more RAM usage, low throughout requires smaller number, therefore less RAM.

Performance parameters

- `number_of_parallel_threads`

Number of threads assigned to parallelise layers. For low latency requirements, assign large numbers (maximum = number of cores), i.e., increase CPU usage. For no latency requirements, use low numbers (minimum = 1), therefore **substantial** reduction in CPU usage.

Example of performance benchmarks



KLEPSYDRA TECHNOLOGIES

ROADMAP

Q2 2021

- No third party dependencies.
  - Binaries are C/C++ only
  - Custom format for models

Q3 2021

- FreeRTOS support (alpha version)
  - Xilinx Ultrascale+ board
  - Microchip SAM V71

Q4 2021

- PykeOS support (alpha version)
  - Xilinx Zedboard

Q1 2022

- NVIDIA Jetson TX2 Support (alpha release)
- Quantisation support

Q2 2022

- Graphs support
- Memory allocation new model
- C support

Legend:

Hard deadlines

Flexible dates

CONTACT INFORMATION

Dr Pablo Ghiglino

pablo.ghiglino@klepsydra.com

+41786931544

www.klepsydra.com

linkedin.com/company/klepsydra-technologies