MARTe

Multi-platform Real-Time Framework

André Neto*+, F. Sartori,
F. Piccolo, R. Vitelli,
G. De Tommasi, L. Zabeo,
A. Barbalace, H. Fernandes,
D.F. Valcárcel, A.J.N. Batista
and JET-EFDA Contributors

*Instituto de Plasmas e Fusão Nuclear
Instituto Superior Técnico
Lisbon, Portugal
http://www.ipfn.ist.utl.pt

+JET-EFDA
Culham Science Centre, UK
http://www.jet.efda.org/

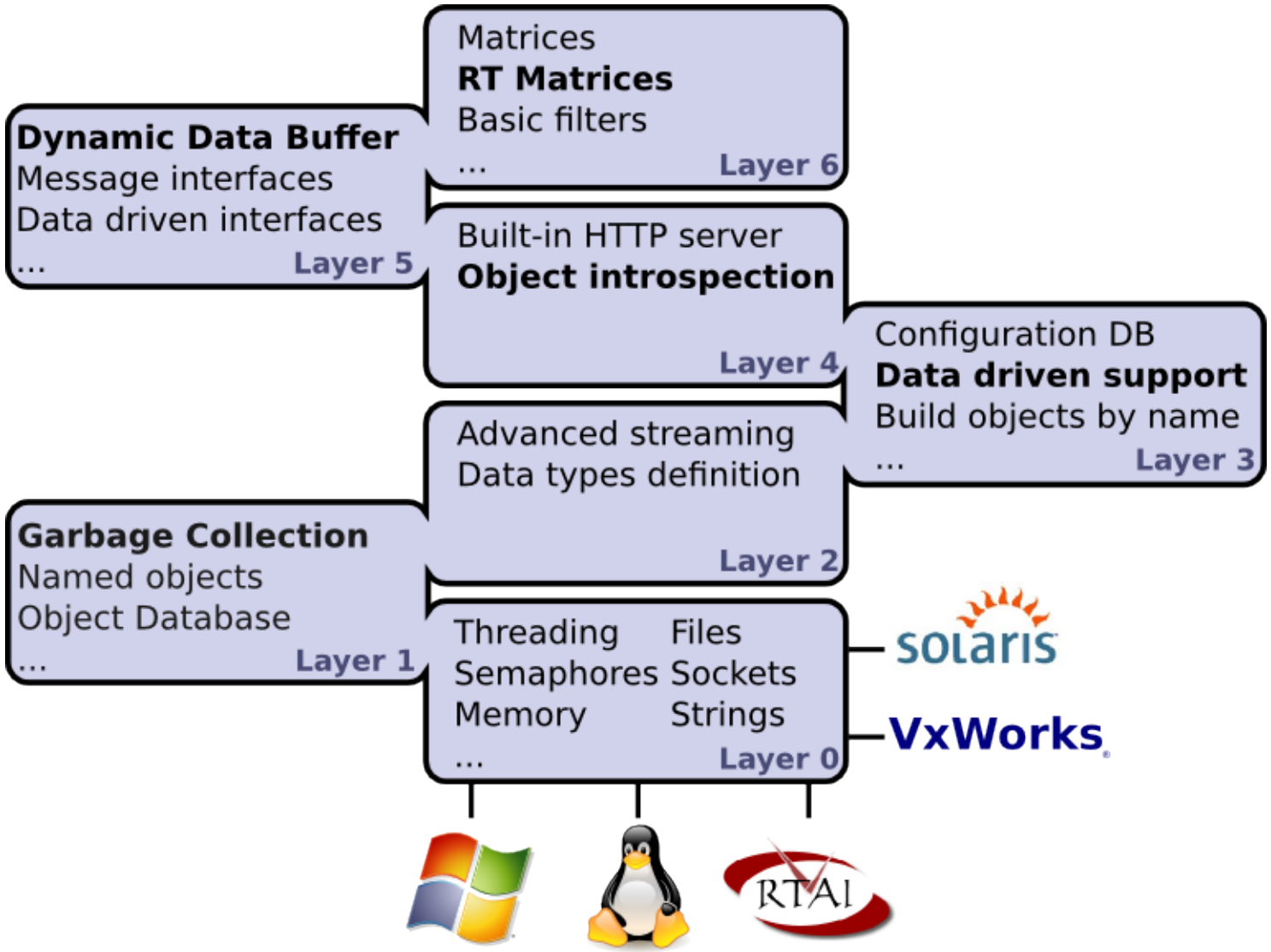
Outline

- What is MARTe?
- MARTe foundations - RT Library
- Components description
- Interfacing with MARTe
- Vertical Stabilisation – an example
- Future developments
- Conclusions

What is MARTe?

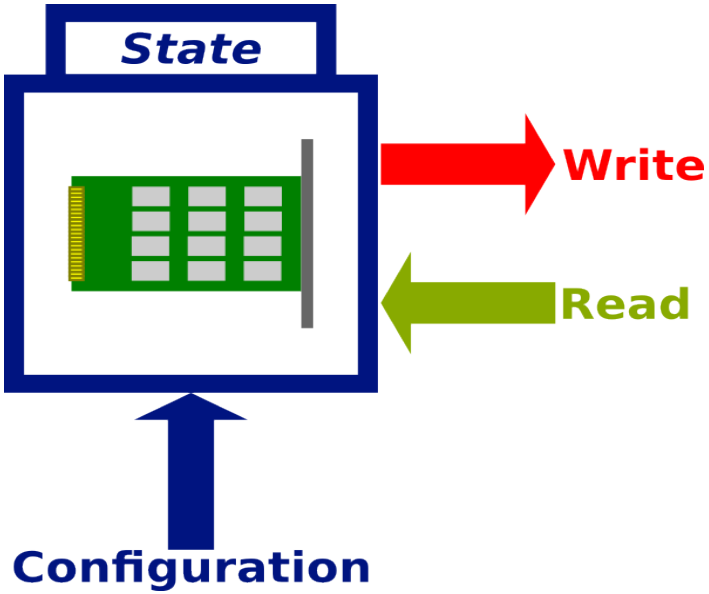
- **Multi-plaform C++ real-time framework**
  - Modular
    - Clear boundary between algorithms, hardware interaction and system configuration
    - Reusability and maintainability
    - Simulation
  - Develop / debug in non real-time environments
  - Data driven
  - Provide live introspection tools
    - HTTP (+CINT), logging

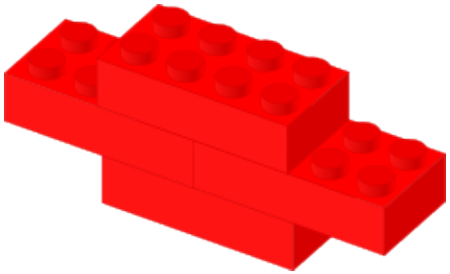
BaseLib2 – support library



Generic Application Module(GAM)

- Atomic block
  - Algorithms
  - Hardware interface
- Only piece of the system which may need to be developed
- Data driven configuration
- Can be inserted and removed at *anytime*
- Consume and produce data





Object Configuration

- Structured syntax
- Similar to XML
- Classes are automatically created
- Configuration is validated by the created object
- Asserting and parsing functions available

```
+HttpServer = {
        Class = HttpService
        Port = 8084
}
...
+Control = {
        Class = ControlGAM
        Controller = {
                NoPlasmaVelocityGain = 0.0
                NoPlasmaCurrentGain = 40.0
                IPWaveform = {
                        Times       = {0 120}
                        Amplitudes = {0.5 0.5}
                        Rounding    = 50
                }
                ...
```

GAM features

- Large part of the code expected to be executed during configuration phase
  - Trap errors as soon as possible
- Special kind of GAMs provide interface with hardware
  - Unique high level interface to any kind of hardware
  - Clear boundary between algorithms and hardware
- Simulation
  - GAMs to simulate the inputs (replacing the hardware)
  - Predict the output of the system using models
  - Live swap of GAMs when required
  - Debug and error tracking

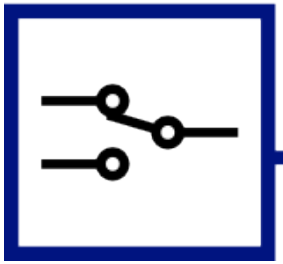
Common GAMs


Hardware I/O


Persistence


Algorithms

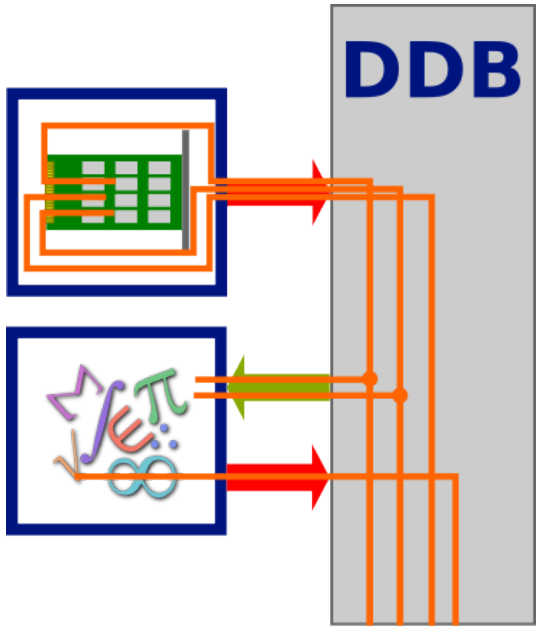

Debug


Decision taking


Information

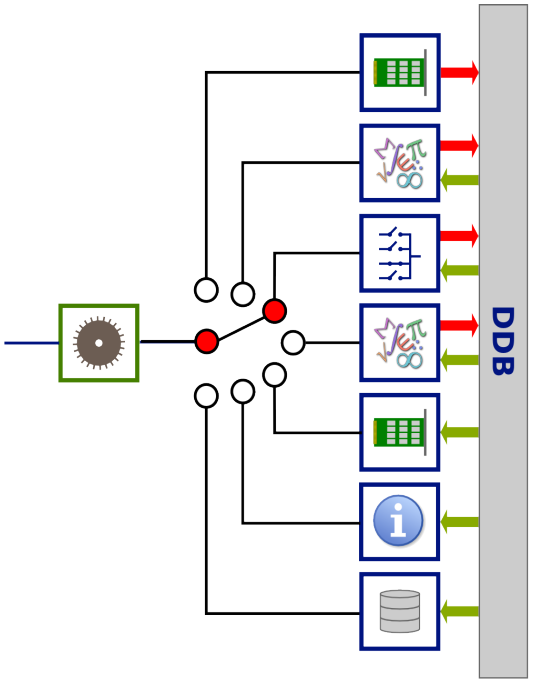
Dynamic Data Buffer

- GAMs shared data through a memory bus
- MARTe guarantees coherency between requested and produced signals
- Set of GAMs allow to stream data to different MARTe systems



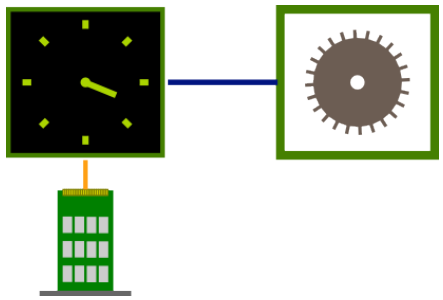
RT-Thread

- Sequentially executes GAMs
  - Works as micro-scheduler
  - Can be allocated to specific CPUs
- Keeps accurate information about execution times
- Requires an external time and triggering mechanism



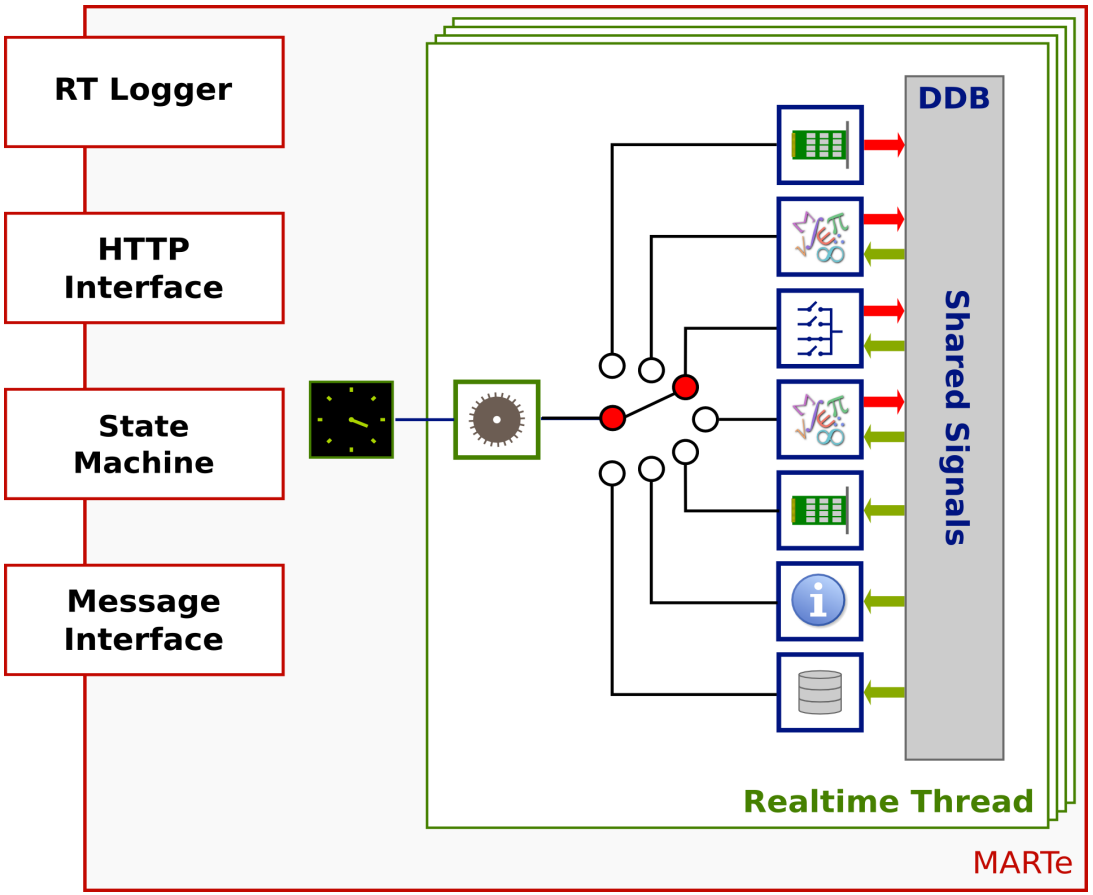
Synchronisation

- Timing for the real-time thread(s)
- Usually provided by an external hardware
- Cycle time can be specified
  - Defines when a new cycle should start
- MARTe supports polling and interrupt based sources



MARTe Components

- At least one RT-Thread
- RT-Threads can execute in parallel
- Several utility components available
  - Logger
  - HTTP server
  - ...

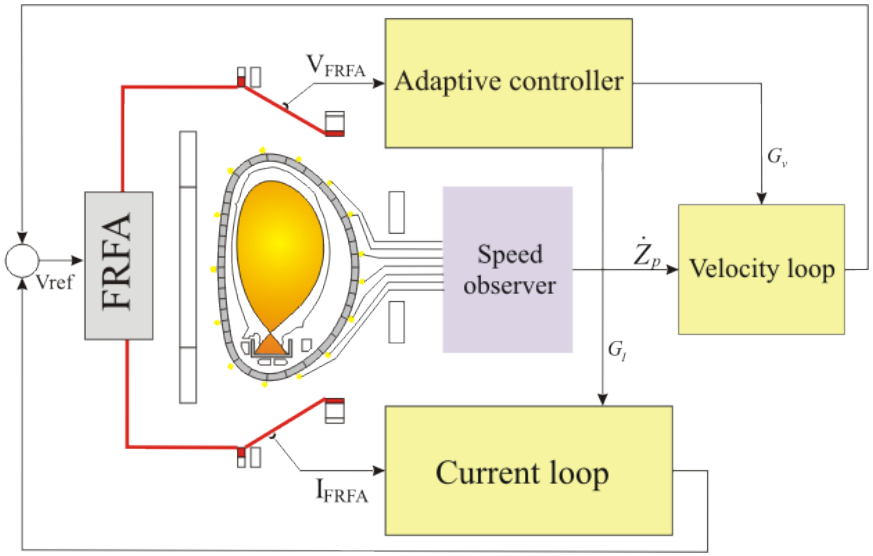


Interfacing with MARTe

- The preferred way is through a *Message Interface*
  - High level protocol
  - Implemented in BaseLib2
- HTTP interaction is widely used for retrieving information
  - Can also be used to change values
    - GAMs configuration
    - State machine
    - ...

Vertical Stabilisation

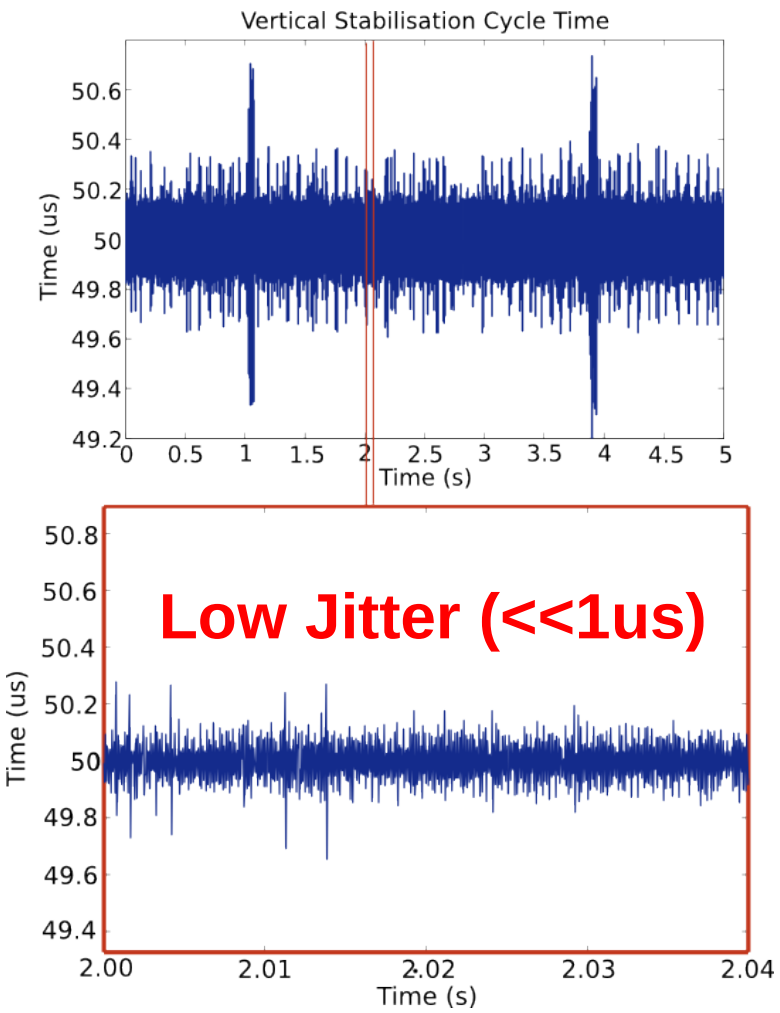
- Elongated tokamak plasmas are susceptible to a vertical axisymmetric instability
- Dedicated Vertical Stabilisation System required
- Essential system for operation
- Growth rate of 1000s^{-1}
- Loss of control can produce forces in the order of the 100's of tonnes

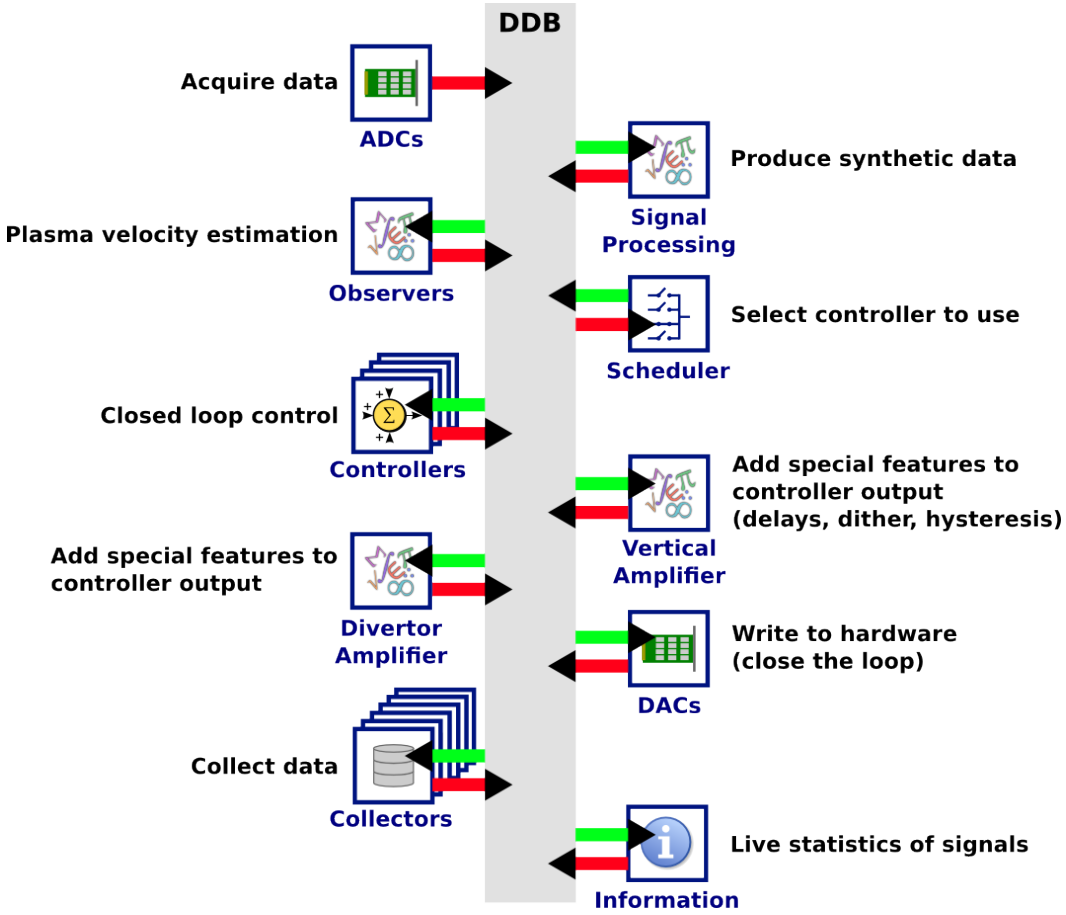


VS - An example

- Closed loop
  - 50 ± 1 μs (with max. jitter of **2.5** μs)
- 18 GAM instances
  - Altogether execute in less than 40 μs
  - Synchronization always achieved within 0.8 μs
- 192 signals acquired by ADCs and transferred at each loop
- Always in real-time (**24 hours per day**)
  - ***1.728 x 10^9 50 μs cycles per day***

VS Execution





Future developments

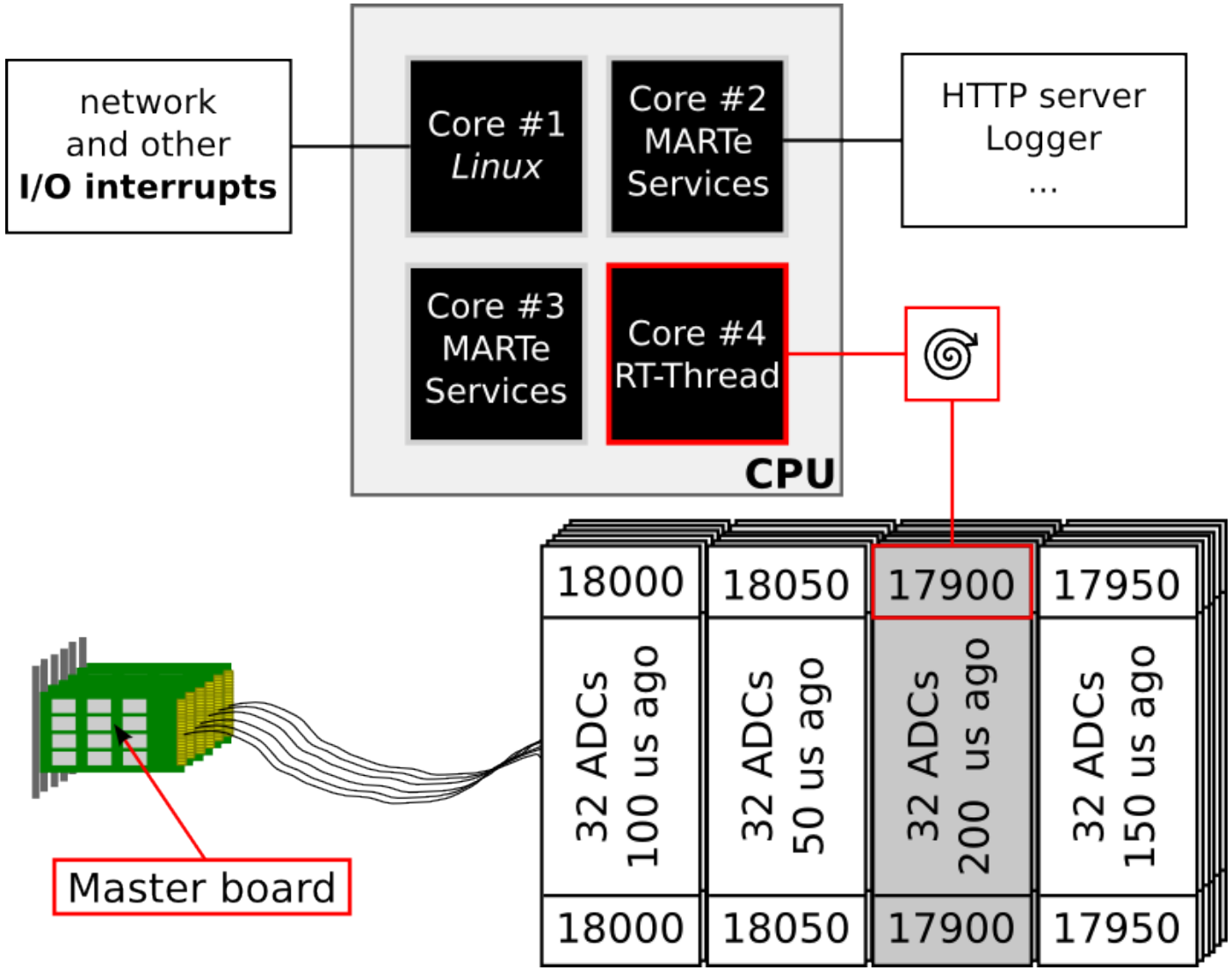
- System is always RT
  - JET is a pulsed based machine
  - ITER will have very long pulses
- First tests with continuous data streaming
  - **Without compromising RT**
  - Bandwidths of ~70MBytes/s
  - To be published
- Testing remote driving of systems

Conclusions

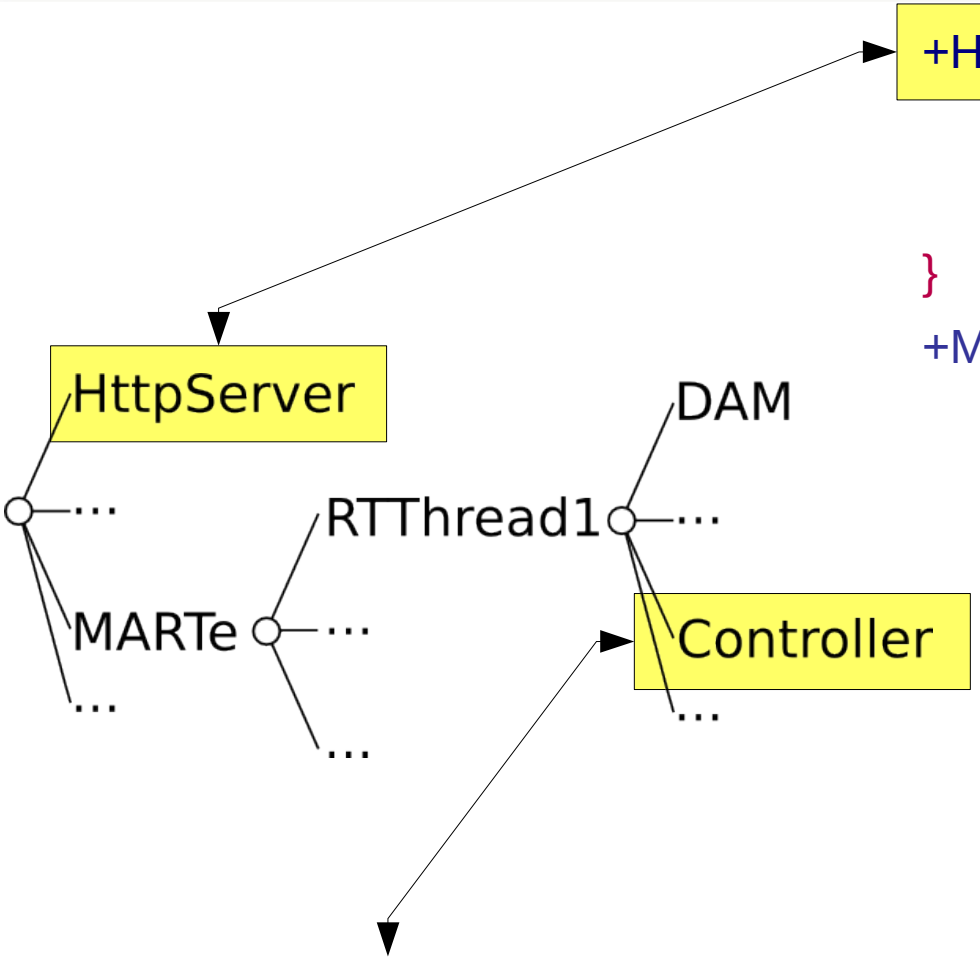
- Modular C++ framework
  - Based on multi-platform RT library
  - Built over data driven modules
  - Clear boundary between hardware and algorithms
  - Interface and behaviour fully configurable
- Drives the VS system
  - Essential system, faults are not acceptable
  - **50µs loop cycle with a jitter inferior to 1µs**
    - Average jitter in the order of 200ns

Support slides

Hardware Synchronization



Configuration DB




```
+HttpServer = {
        Class = HttpService
        Port = 8084
}
+MARTe = {
        Class = MARTeContainer
        +RTThread1 = {
            Class = RealTimeThread
            +Controller = {
                NoPlasmaCurrentGain = 40.0
                IPWaveform = {
                        Times       = {0 120}
                        Amplitudes = {0.5 0.5}
                }
                ...
```

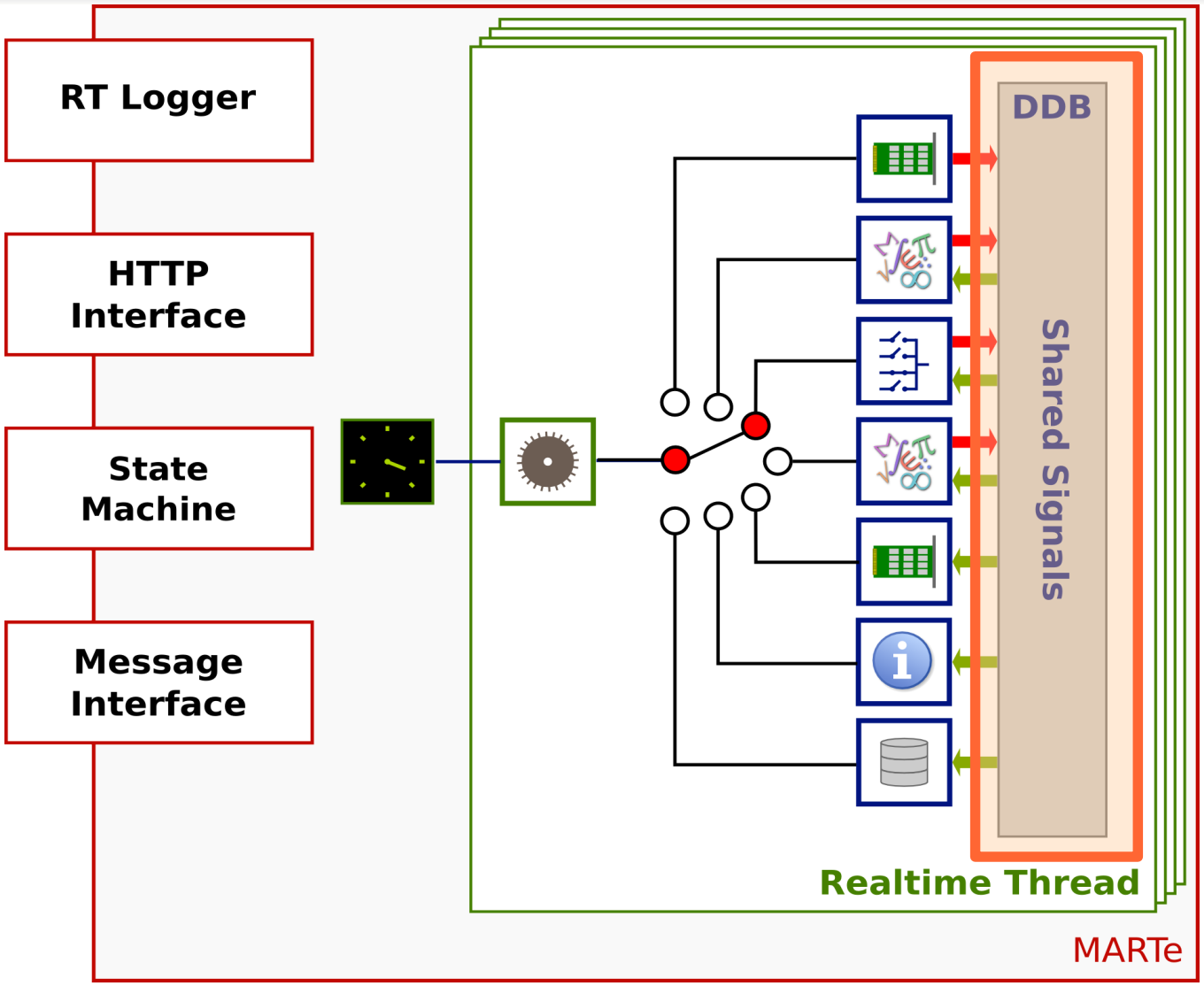
Messages

- Objects that implement the **Message Interface** are able to send and receive synchronous and asynchronous messages
- Messages allow to interface by protocol
- Messages can contain object configurations
- Can be sent through the network
- Important mechanism in the design of MARTe

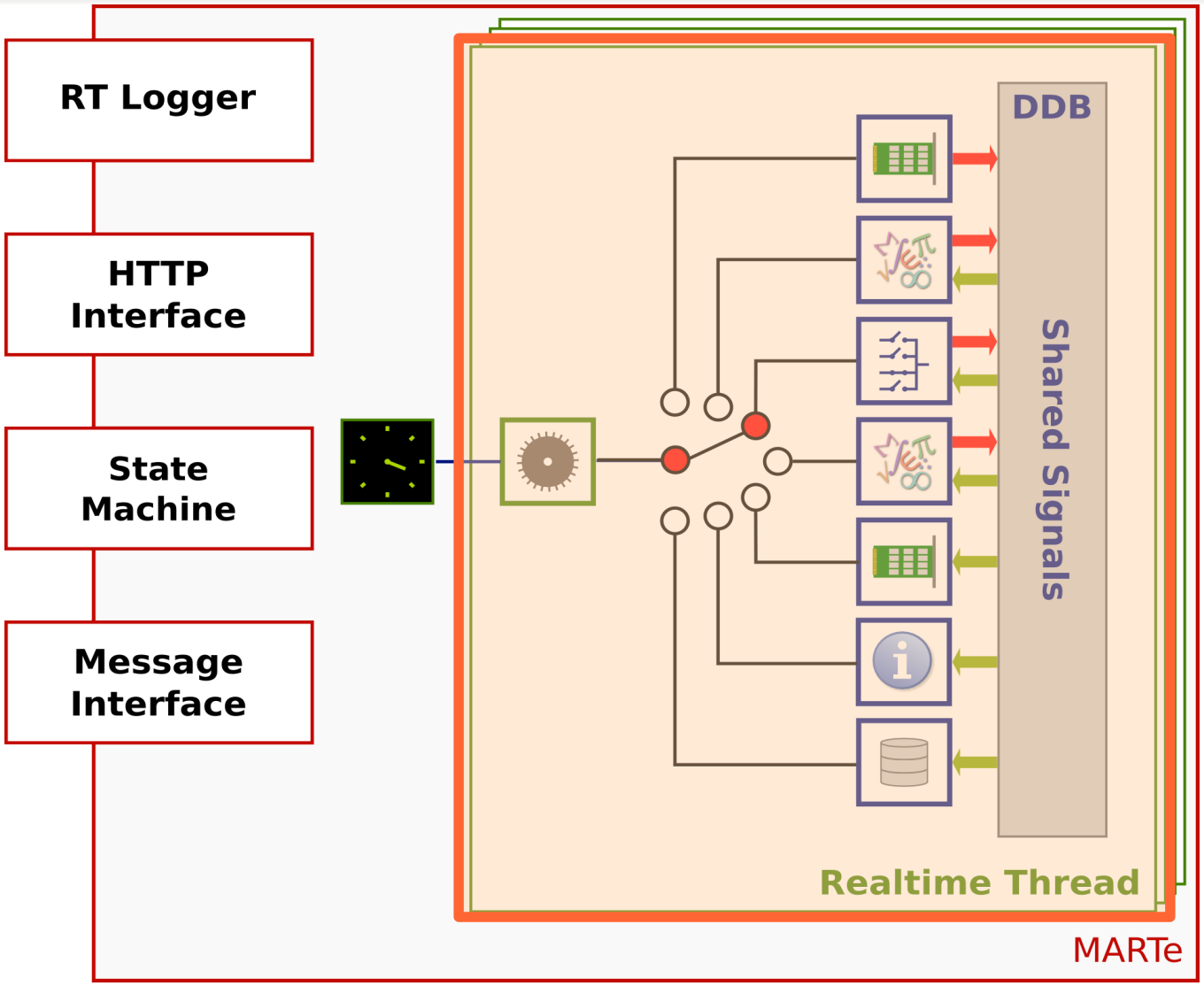
VS performance

- **Closed loop**
- Only possible because RT processes aren't disturbed
- Full access to the computer without compromising RT
- **50µs loop cycle with a jitter inferior to 1µs**

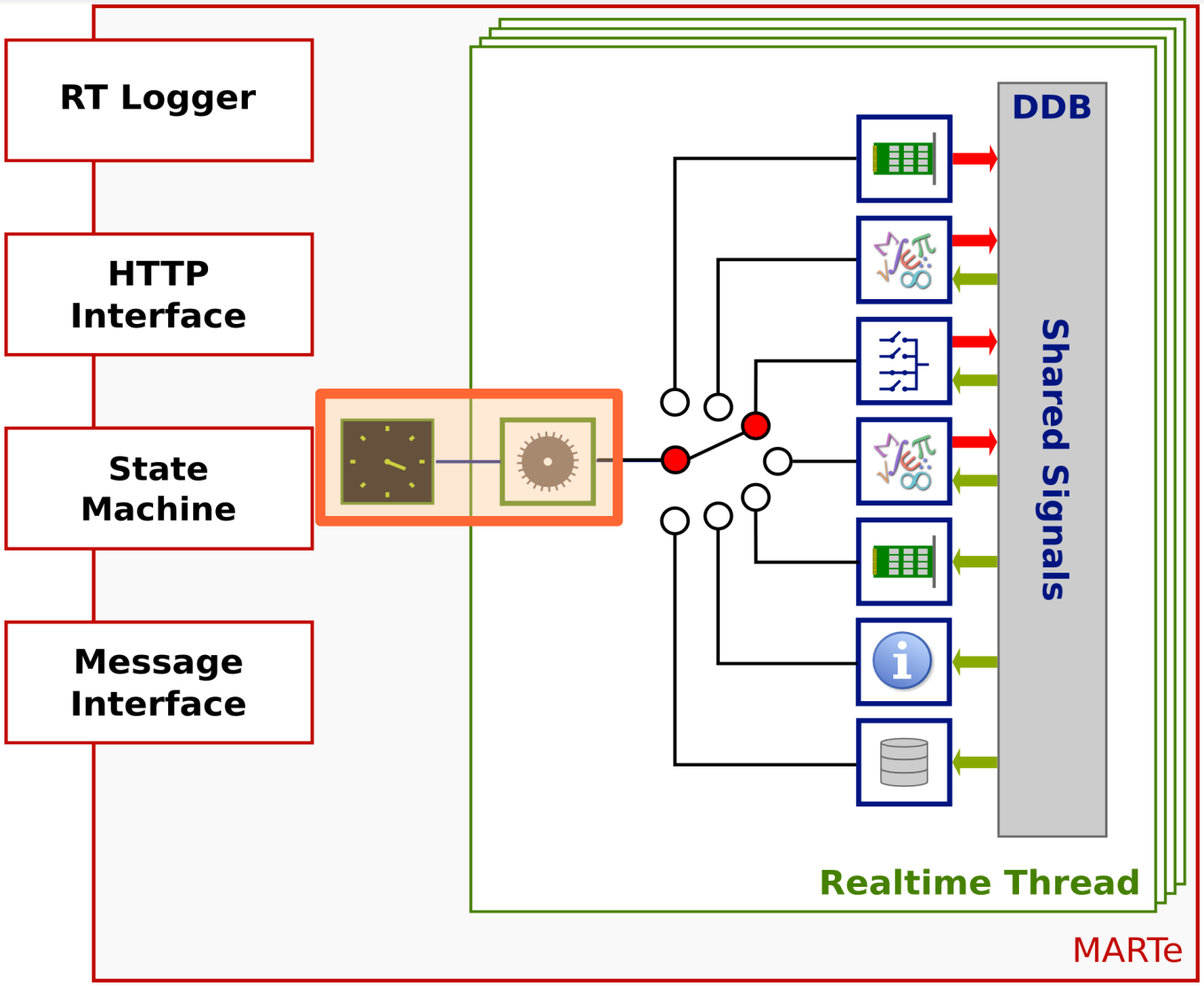
Dynamic Data Buffer



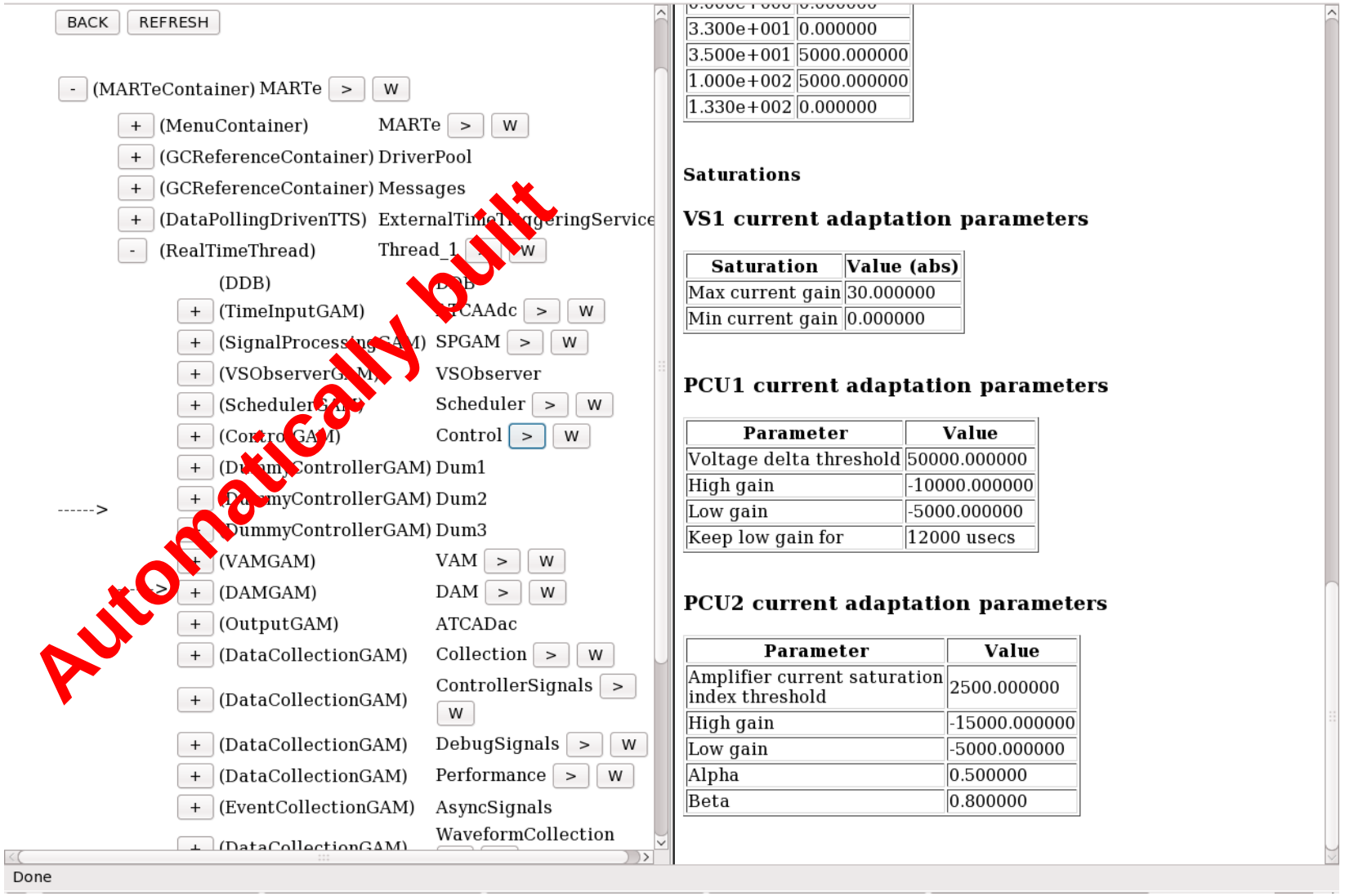
Real-time thread



Synchronisation

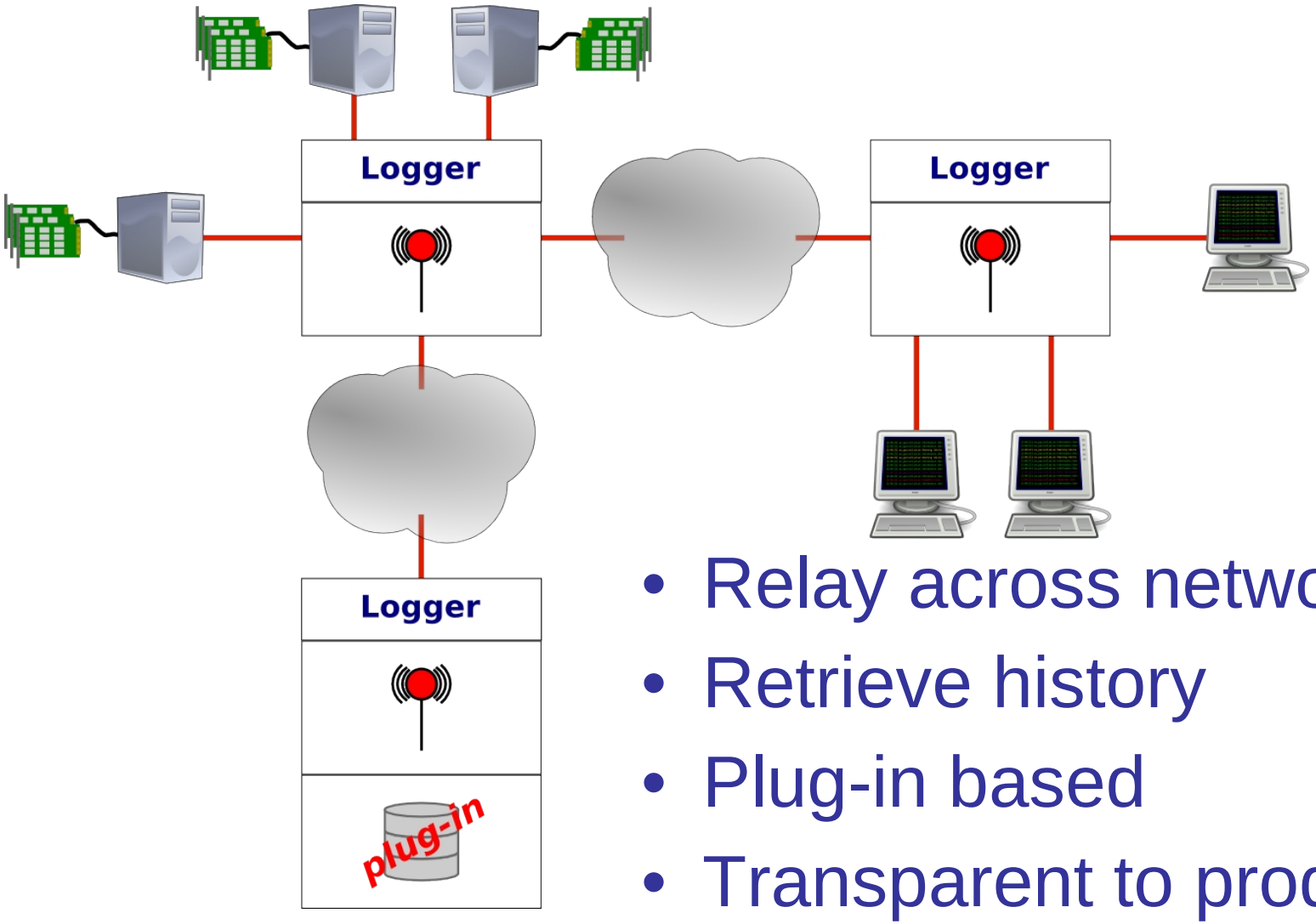


MARTe HTTP



Automatically built

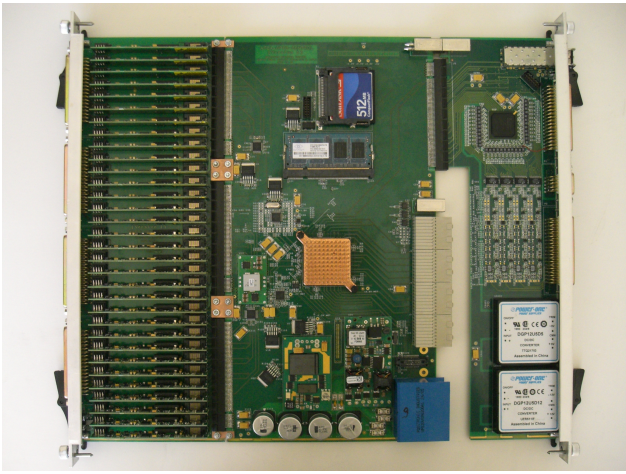
MARTe tools - logger

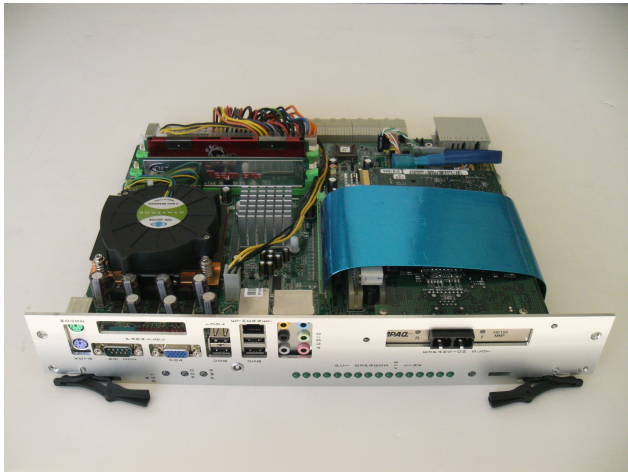


- Relay across networks
- Retrieve history
- Plug-in based
- Transparent to producers

VS - Hardware

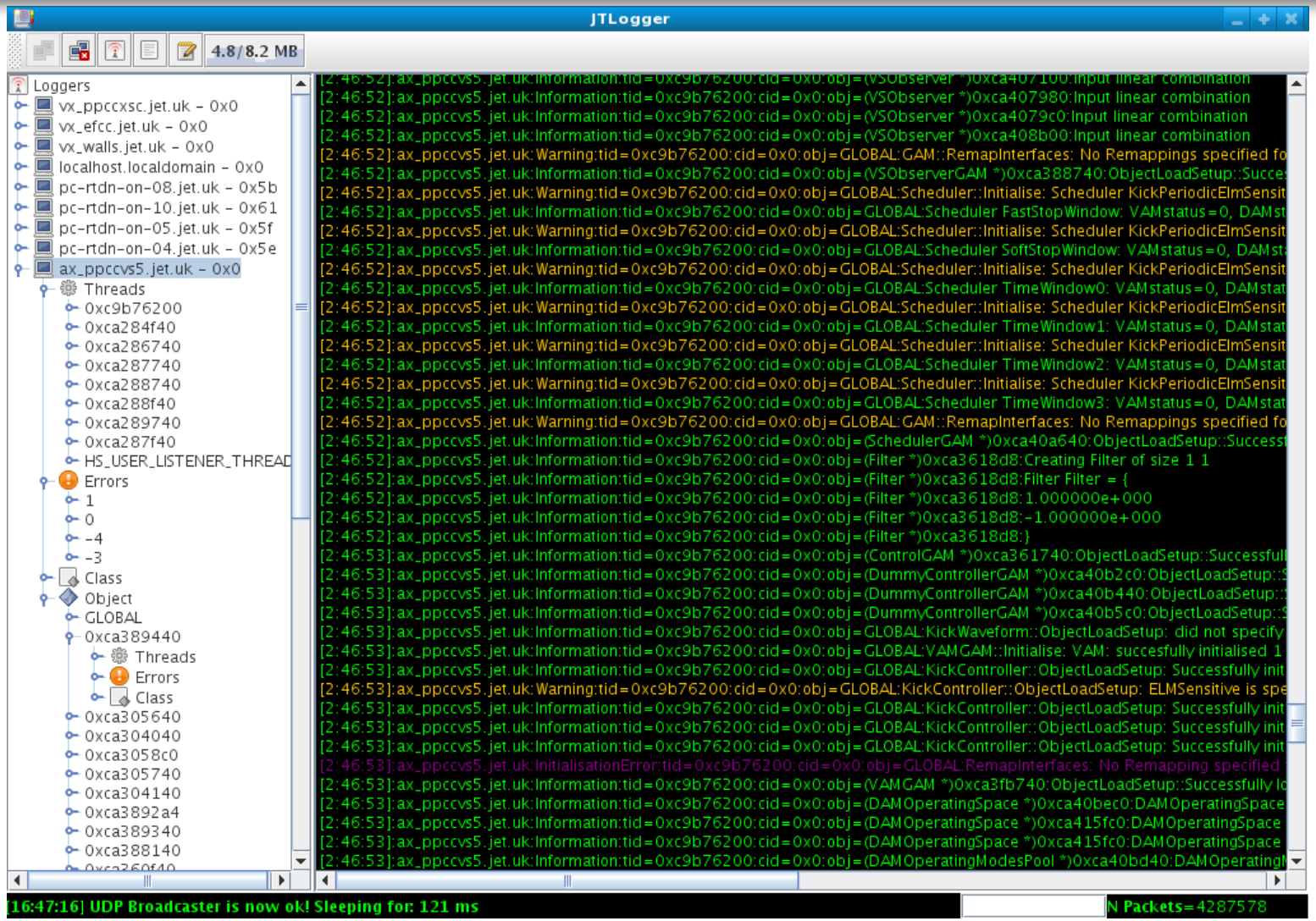
- 6 ATCA Data Acquistion Boards
- 2MSPS - 18 bit ADCs
- PCIe interface
- Intel Core 2 Quad
- Digital I/O
  - Connection to new amplifier





- *See **FESPP-25** in the Poster Area*

JTLogger


JTLogger
4.8/8.2 MB
Loggers
vx_ppccxsc.jet.uk - 0x0
vx_efcc.jet.uk - 0x0
vx_walls.jet.uk - 0x0
localhost.localdomain - 0x0
pc-rtdn-on-08.jet.uk - 0x5b
pc-rtdn-on-10.jet.uk - 0x61
pc-rtdn-on-05.jet.uk - 0x5f
pc-rtdn-on-04.jet.uk - 0x5e
ax_ppccvs5.jet.uk - 0x0
Threads
0xc9b76200
0xca284f40
0xca286740
0xca287740
0xca288740
0xca288f40
0xca289740
0xca287f40
HS_USER_LISTENER_THREAD
Errors
1
0
-4
-3
Class
Object
GLOBAL
0xca389440
Threads
Errors
Class
0xca305640
0xca304040
0xca3058c0
0xca305740
0xca304140
0xca3892a4
0xca389340
0xca388140
[16:47:16] UDP Broadcaster is now ok! Sleeping for: 121 ms
N Packets=4287578

The Goal

- Use MARTe to run the JET Vertical Stabilisation

Essential system for operation

Closed loop control cycle time:
Target:
50 ± 1 μs
(with max. jitter of **2.5** μs)

Achieved:
50 ± 0.10 μs
(max jitter of 0.80 μs)