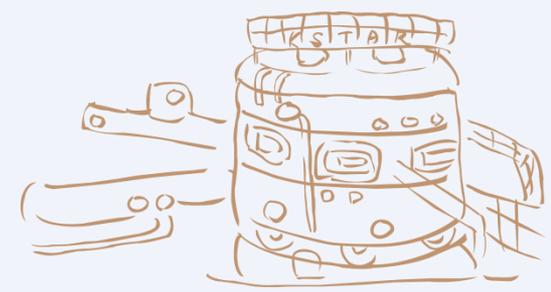




EPICS collaboration meeting fall 2012
22-26 October 2012

by Pohang Accelerator Laboratory (PAL)



Interface of EPICS with MARTE for Real-time Application

Fusion

Future Vision of Green Energy

October 23, 2012

Sangwon Yun



NFRI 국가핵융합연구소
National Fusion Research Institute

- **Introduction**
 - **ITER Task**
- **Overview of main technical subjects**
 - **Multi-threaded Application Real-Time executor (MARTE)**
 - **Portable Channel Access Server (pCAS)**
- **Interface of EPICS with MARTE**
- **The result of performance evaluation of MARTE**
- **Conclusions**
- **Acknowledgements and references**

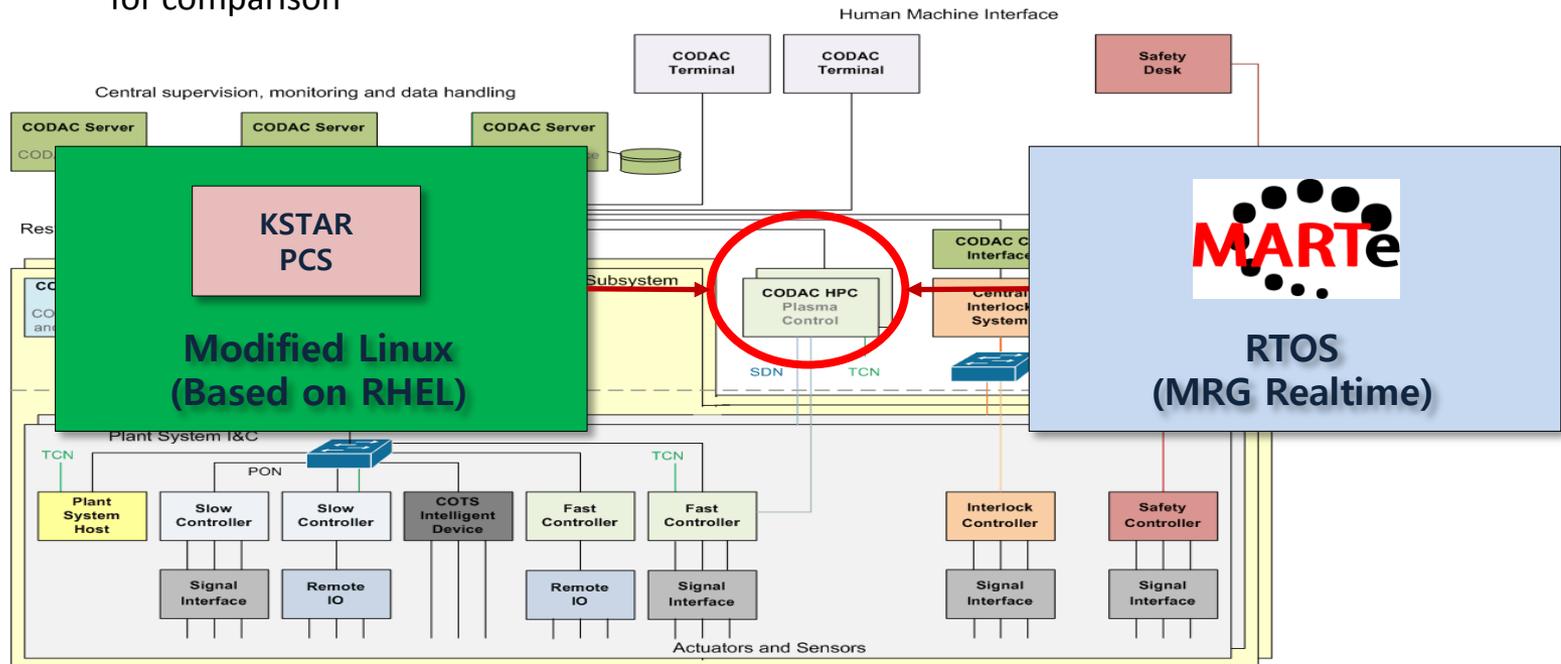
ITER Task

◆ CODAC HPC (High Performance Computer)

- ✓ Dedicated computers running plasma control algorithms
- ✓ The Plasma Control System(PCS) requires the hard real-time operation
- ✓ The hard real-time control system requires the predictability in response time and deterministic upper bound in latency. In this regard, RTOS and real-time software framework are required to achieve those

◆ ITER Task : Evaluation and Demonstration of ITER CODAC Technologies at KSTAR

- ✓ Subtask : Implement density feedback control and verify its performance in a running device
 - To collect elements for decision making on PCS software framework
 - Deploy different PCS software framework based on **KSTAR PCS on modified Linux** and **MARTe on RTOS** for comparison



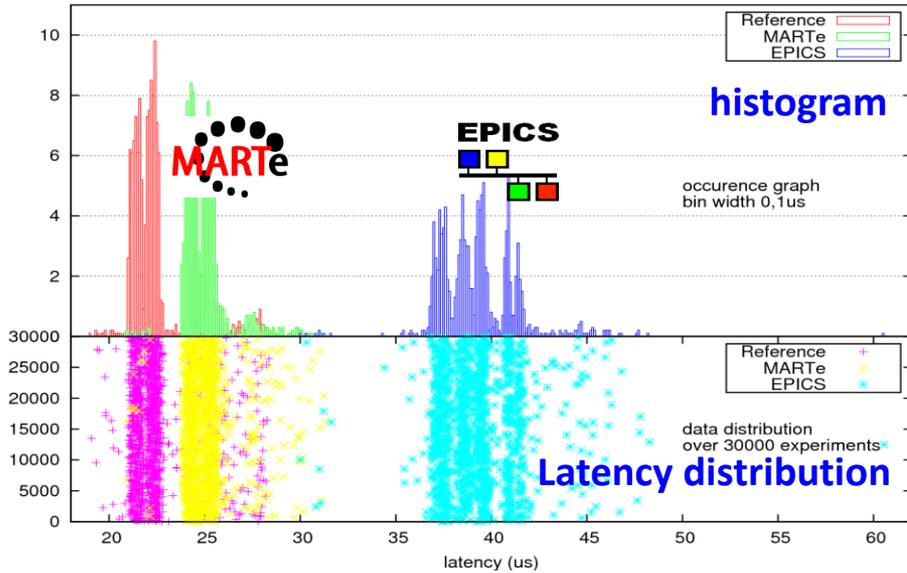
See: Fast Controller Workshop, Feb 28, 2011. ITER IO

Interface of EPICS with MARTe for Real-time Applications, October 23, 2012

Introduction

MARTE framework for Hard Real-time Control System

MARTE is a solution to be considered for a Real-time control system



- The three considered system
 - ✓ The optimized reference program
 - ✓ EPICS IOC
 - ✓ MARTE application
- Test conditions
 - ✓ Run the system by providing an external sampling clock of 1 kHz
 - ✓ 30,000 cycles
 - ✓ Priority of the thread for ADC has been set to the highest
 - ✓ CPU affinity (excepts EPICS IOC)
- Result
 - ✓ MARTE provides a shorter and, above all, more bounded latency
 - ✓ The added latency due to MARTE in respect of the reference program is on average 2.7 μs
 - ✓ The added latency due to EPICS is 17 μs
- New comparison the affinity patched EPICS

	Reference Prog.	EPICS IOC	MARTE
min	19.0 us	30.0 us	20.9 us
MAX	28.3 us	60.5 us	31.3 us
Avg	22.3 us	39.3 us	25.0 us

min, MAX and average overall IO latency

See: A. Barbalace, et. al, Performance comparison of EPICS IOC and MARTE in a Hard Real-Time Control Application, in IEEE-NPSS RT 2010

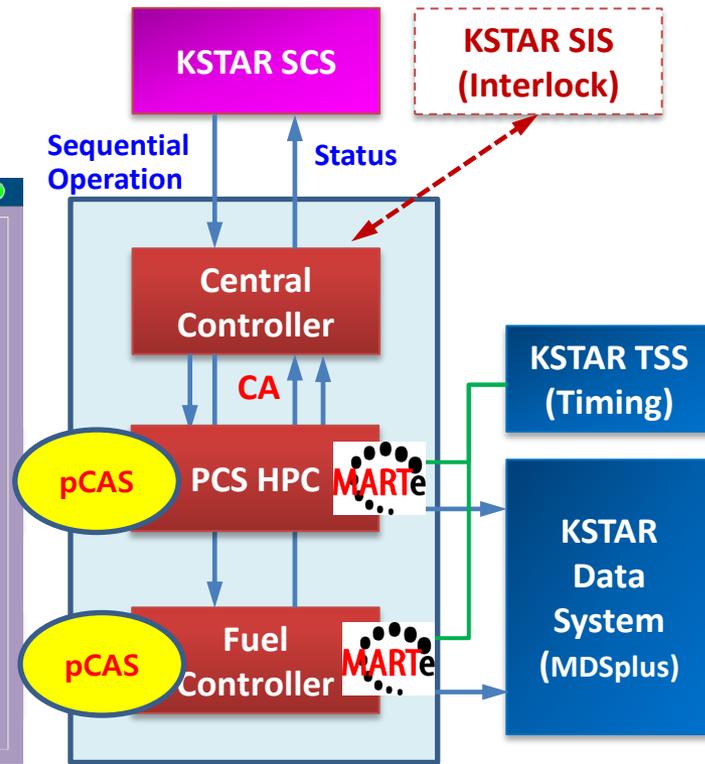
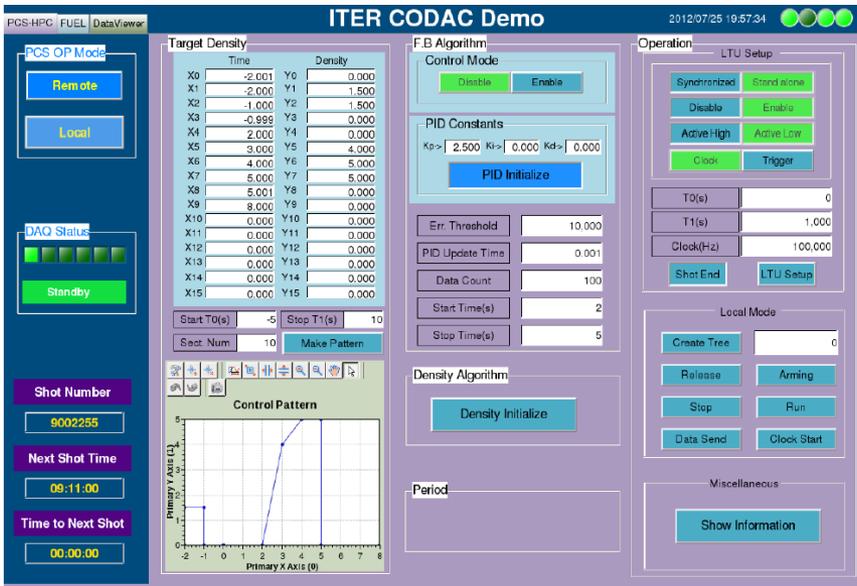
● Integration of MARTe applications with KSTAR Systems

◆ Interface with EPICS as a framework for the control systems

- ✓ The EPICS is a standard framework for the control systems of ITER and KSTAR
- ✓ Sequential operation for plasma experiment
- ✓ Configuration and monitoring the MARTe applications and signals

◆ Interface with MDSplus as a database for the experimental data based on a pulse

CSS BOY OPI for PCS HPC



ITER CODAC Demo System

● A bit of history

- ◆ **MARTe has been developed at JET in collaboration also with RFX**
- ◆ **At the end of 90s at JET only dedicated hardware/software solutions**
 - ✓ The Joint European Torus (JET) is the world's largest and most powerful tokamak
 - ✓ Limitations : No flexibility, No easy debugging and testing, Long commissioning time
- ◆ **JETRT framework (2002/2003)**
 - ✓ Based on a cross-platform library: BaseLib
 - ✓ Clear separation between application and infrastructure software
 - ✓ Application can abstract from the plant interfaces
 - ✓ Increase code reusability, Achieve standardization
 - ✓ Limitations
 - It didn't provide **a real separation between the user application from the plant interface software**
 - Need to be recompiled entirely in case of changing in both interface side and/or application side
- ◆ **MARTe (2006 ~)**
 - ✓ An implementation of numerous concepts during the last 10 years of work mainly at JET
 - ✓ Underlying **BaseLib2** is the supporting collection of libraries that make MARTe possible.
 - The **net separation** between I/O and application has been the driving concept.
 - **High modularity and separation** allow fast debugging and testing, short commissioning time, high portability and adaptability.
 - Programmability of single GAM allows to use the same algorithm in many application

● MARTe in Fusion

◆ Machines adopted MARTe as real-time framework for their control system

◆ JET : CCFE, UK

- ✓ Vertical Stabilization System (VS) : RTAI , 50 μ s Cycle time with jitter < 1 μ s, ATCA
- ✓ Error Field Correction Coils (EFCC) : VxWorks, 200 μ s Cycle time, VME
- ✓ Real Time Protection System : VxWorks 6
- ✓ Vessel Thermal Map (VTM) : Linux
- ✓ WALLS2011 : Linux

◆ ISTTOK : IST Lisbon, Portugal

- ✓ Real-time tomography : Linux-RTAI, 100 μ s Cycle time, ATCA

◆ COMPASS : Prague, Czech Republic

- ✓ Plasma Magnetic Control System : Linux, 50~500 μ s Cycle time, ATCA

◆ RFX : Consorzio RFX Padua, Italy

- ✓ Magnetic Control System : Linux (Preempt RT), 200~250 μ s Cycle time, PXI

◆ FTU : ENEA Frascati, Italy

- ✓ Lower Hybrid power ratio control : RTAI, 250 μ s Cycle time, VME

● MARTe Framework is

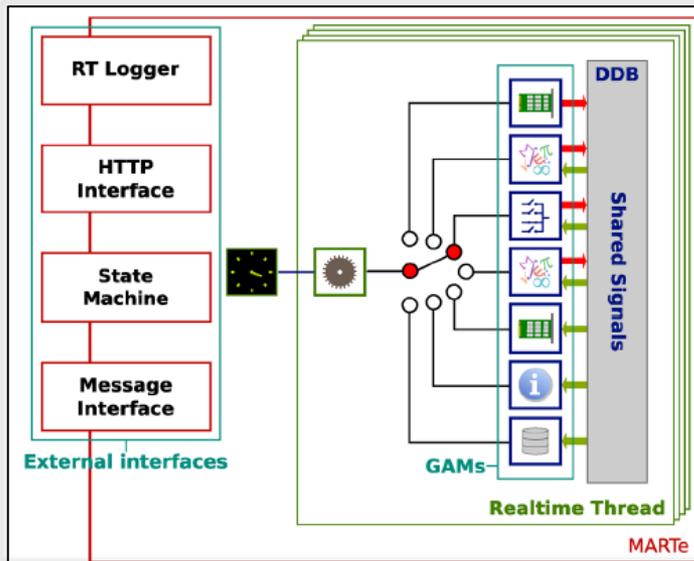
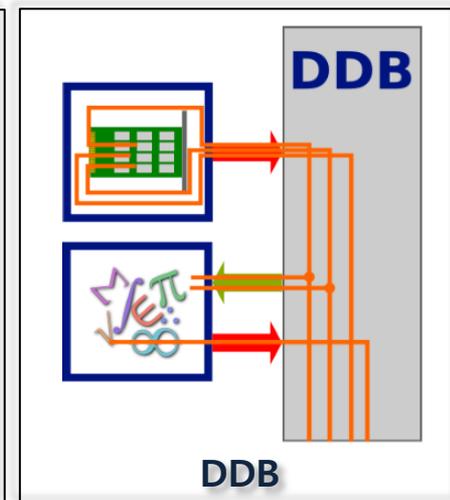
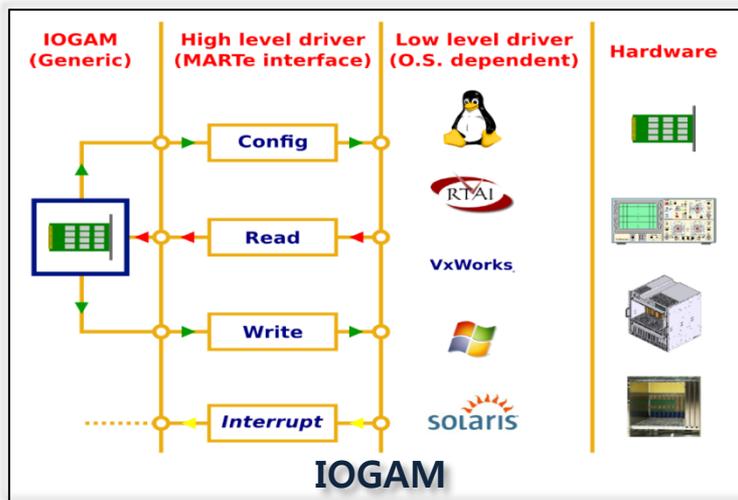
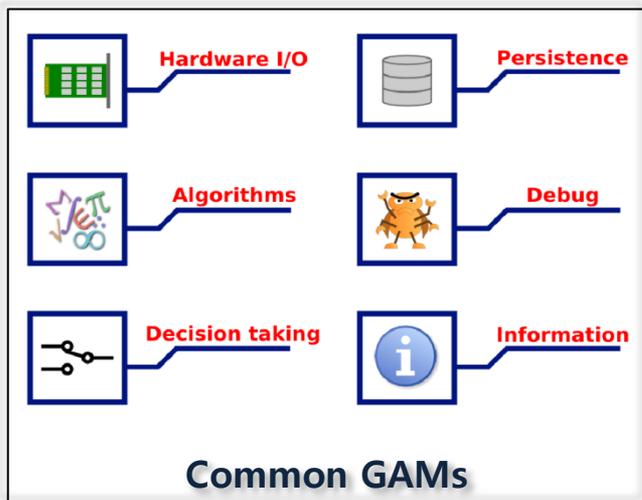


A. C. Neto et al.,

MARTe: a Multi-Platform Real-Time Framework,
IEEE Transactions on Nuclear Science, vol. 57(2), Apr. 2010

- ◆ **Multi-threaded Application Real-Time executor (MARTe)**
- ◆ **Multi-platform C++ middleware designed for real-time control system**
 - ✓ It runs on Linux, Linux+RTAI, VxWorks, Solaris, Windows and Mac OS X
 - The supporting multi-platform allows to develop and debug the real-time application in non RT targets
 - This gives usually better developing environment and allows short commissioning time
 - ✓ Middleware for Real-time Control development
- ◆ **Simulink-like way of describing the problem**
- ◆ **Modularity (GAMs)**
 - ✓ The atomic element of MARTe is named Generic Application Module (GAM), and all applications built using the framework are designed around these components
 - ✓ Data is transferred between GAMs by using an optimized memory bus named Dynamic Data Buffer (DDB). The DDB is the only available way for GAMs to interchange information
 - ✓ This characteristic of the GAM and DDB guarantee the clear boundary between algorithms, hardware interaction and system configuration. In addition, reusability and maintainability are also increased
 - ✓ Simulation
 - Replace actuators and plants with models
 - Keep all the other modules untouched
 - ✓ Minimize constraints with the operational environments (portability)
- ◆ **Data driven**
- ◆ **Provide live introspection tools without sacrificing RT**

Quick look at MARTe



- ✓ The real-time thread is a **container of GAMs** and acts as a **GAM micro scheduler** being responsible for their sequential execution. The thread can be **configured to run in specific processors** and can be **assigned to a specified priority**.
- ✓ The figure depicts a possible set of modules that start by acquiring signals from a hardware device, processing and taking decisions upon this data, and finally outputting the signals to both a device and a storage scheme.
- ✓ Threads can be **configured to run at a specific frequency**, and all the GAMs are expected to execute within this time.

RT-Thread

● Quick look at MARTE

EPICS IOC

MARTE

Similarities and Differences

Records

modular components based architectures

GAMs

ReCompile

current configuration is defined in a text file

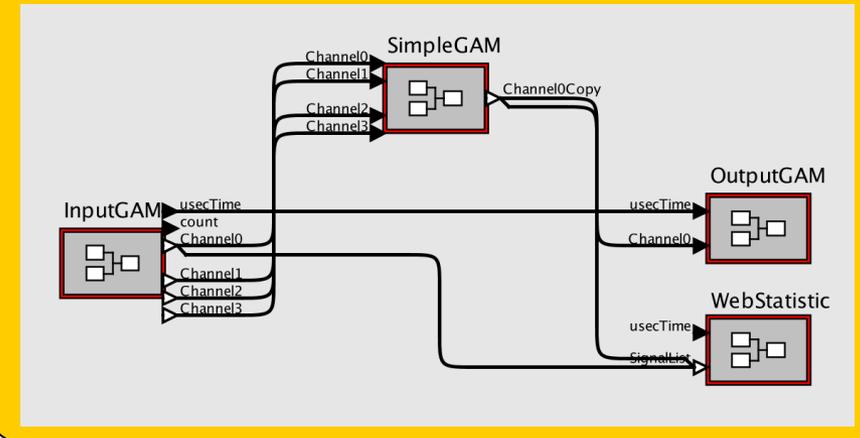
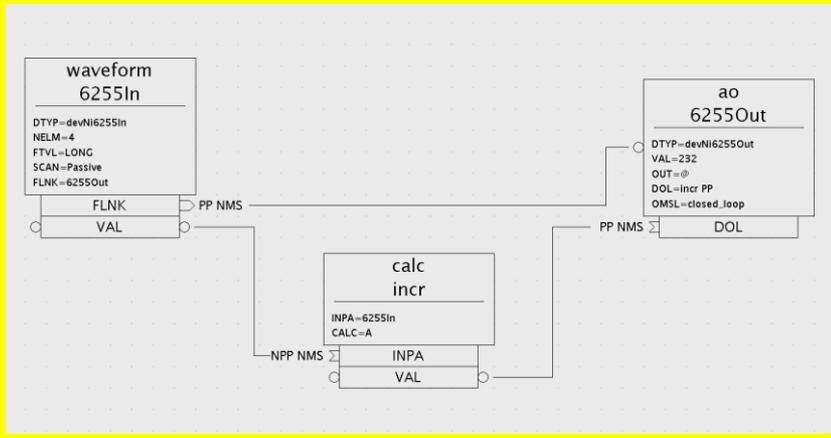
ReLoading

No Control

computational resource model

Full Control

Case Study

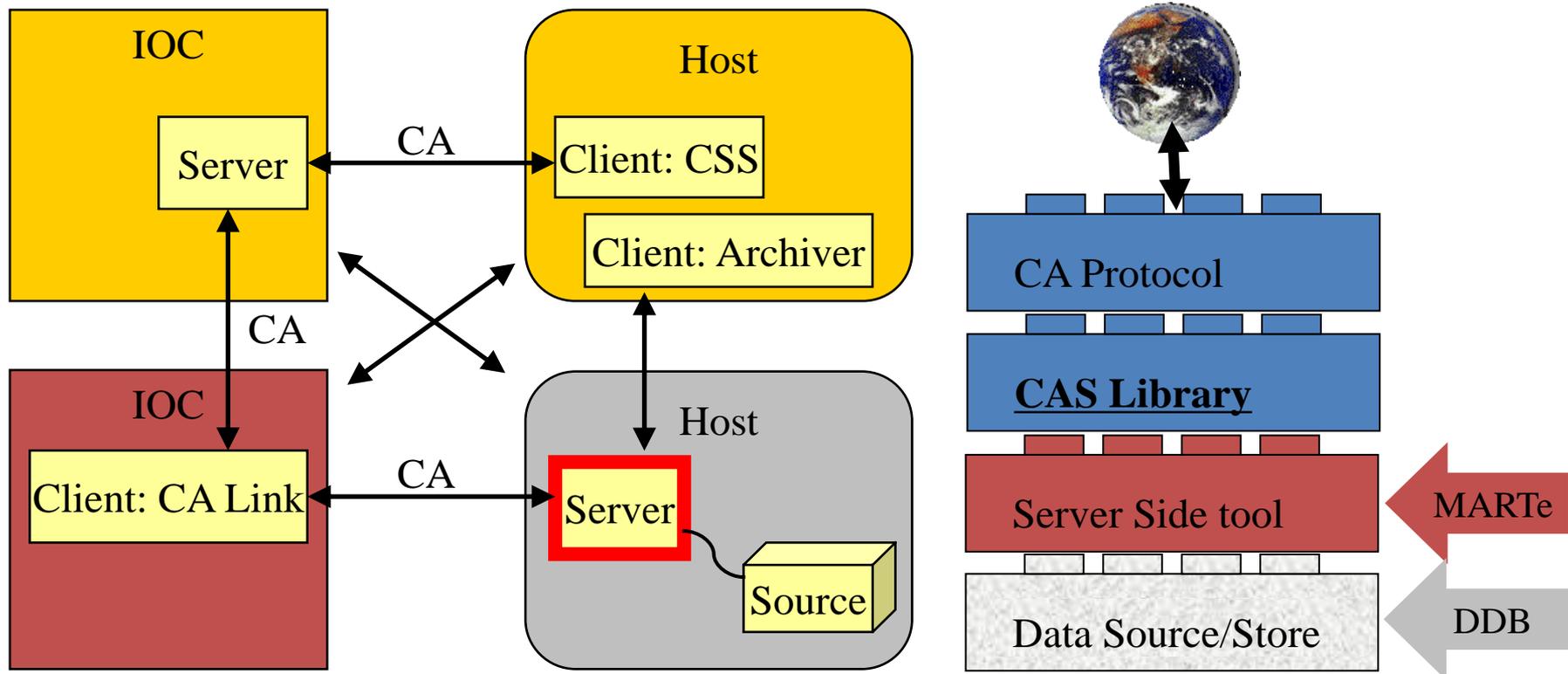


See: A. Barbalace, et. al, Performance comparison of EPICS IOC and MARTE in a Hard Real-Time Control Application, in IEEE-NPSS RT 2010

Portable Channel Access Server

● Channel Access : The EPICS Network Protocol

- ✓ The Channel Access server exports data to EPICS and the Channel Access client can read and write Process Variables over the network
- ✓ Any host want to export own data to EPICS should be the Channel Access Server
- ✓ The pCAS makes any host as a Channel Access Server
- ✓ The Server side tool uses the CAS library and the CAS library uses CA protocol



See: Jeff Hill, Kay-Uwe Kasemir, Channel Access Server Tool Developers Training

● Overview

◆ What is the Portable Channel Access Server (pCAS)?

- ✓ The pCAS consists of a **C++ library** with a simple class interface
- ✓ Using the **simple interface to the library**, a developer can create a **Channel Access server tool** that can interact with an EPICS database as well as other applications

◆ Functions of the Server Tool

- ✓ Creates/deletes server instance
- ✓ Responds to client requests
 - **PV search, Attach/detach request, Read/Write request**
- ✓ Posts change of state events

◆ Example CA servers

- ✓ **Channel access gateway**
- ✓ Directory server, Fault logger APT HPRF
- ✓ KECK instruments, KEKB gateway to LINAC control system
- ✓ SLAC gateway to SLAC control system, Gateways to other control systems at DESY

◆ Advantages of a Server Tool

- ✓ **MARTE application becomes an EPICS Channel Access Server tool**
- ✓ **MARTE signals become EPICS process variables**

● C++ Server Interface

◆ The Portable Server API consists of 9 classes

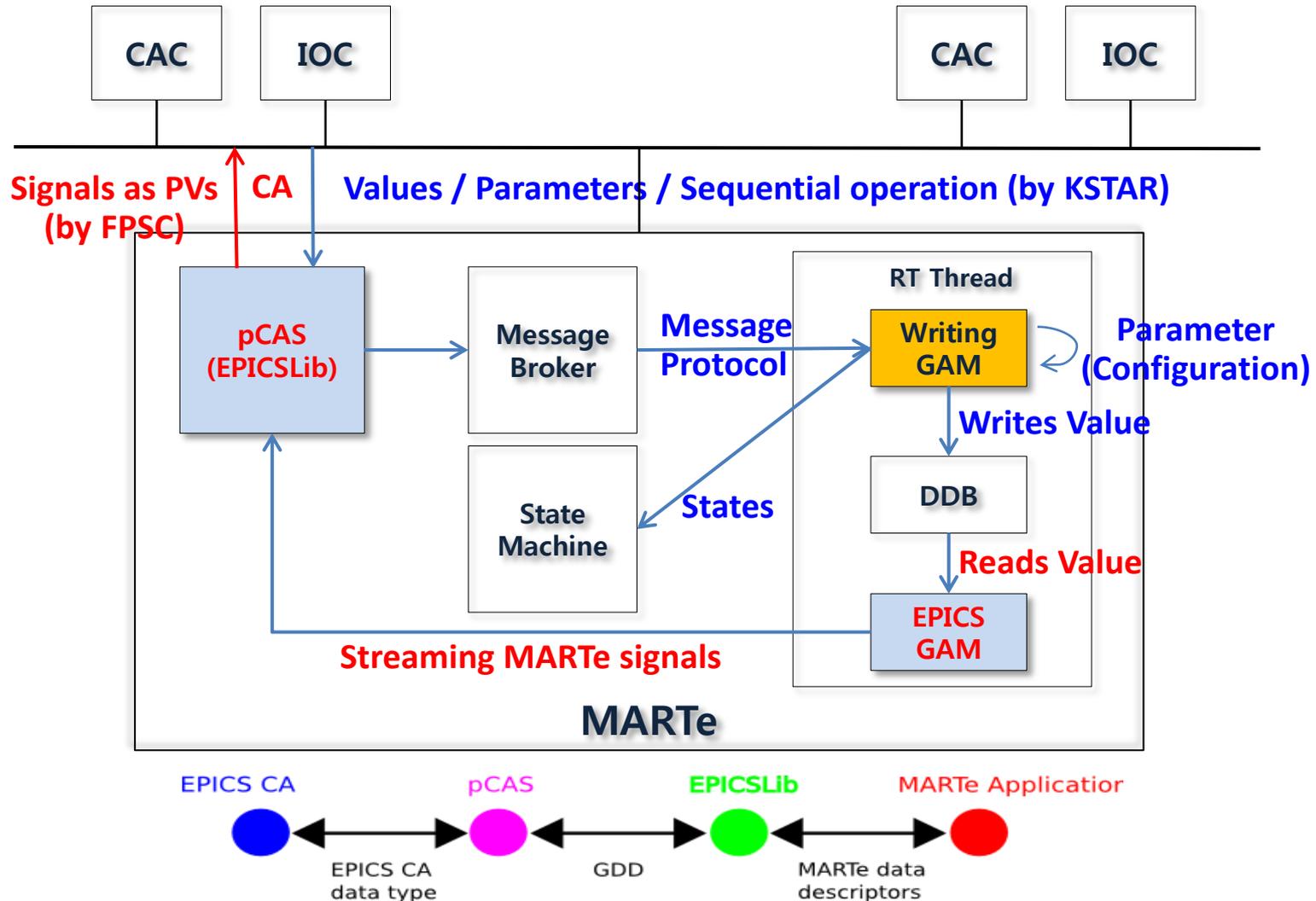
- ✓ **Server class, caServer**
 - ✓ Every server tool must include a class derived from the caServer class
 - ✓ Informs the server library if a PV is associated with the server tool
 - ✓ Attaches a PV when a client wishes to establish a connection
- ✓ **Process variable class, casPV**
 - Responds to read/write PV requests : read(), write()
 - Responds to a request for a PV monitor : interestRegister(), interestDelete(), postEvent()
- ✓ **pvExistReturn** : Response to a client CA search
- ✓ **pvAttachReturn** : Called when client wishes to attach to PV
- ✓ Channel class, casChannel
- ✓ casAsyncPVExistIO
- ✓ casAsyncCreatePVIO
- ✓ casAsyncReadIO
- ✓ casAsyncWriteIO

◆ How to develop the server tool

- ✓ The first four classes are required to implement the server tool
- ✓ The channel class and the asynchronous IO classes can be used to add more functionality
- ✓ Each class has several member functions which server tool must define

Interface of EPICS with MARTe

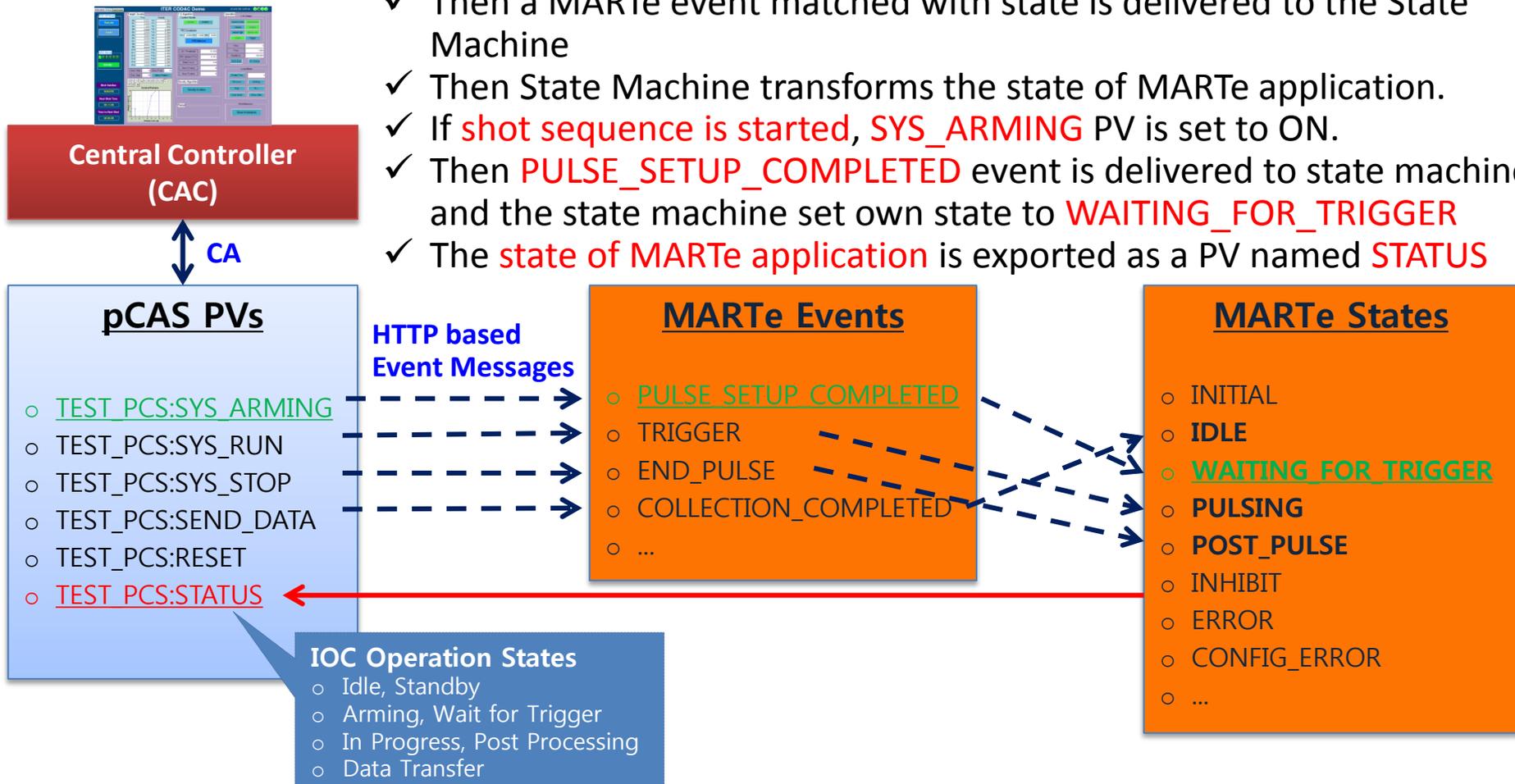
● Software Architecture



Interface of EPICS with MARTE

● How to synchronize the two STATE Machines EPICS IOC and MARTE

- ✓ The Central controller EPICS IOC synchronized with KSTAR SCS writes pCAS PVs in MARTE based on state of the sequential operation
- ✓ Then a MARTE event matched with state is delivered to the State Machine
- ✓ Then State Machine transforms the state of MARTE application.
- ✓ If **shot sequence is started**, **SYS_ARMING** PV is set to ON.
- ✓ Then **PULSE_SETUP_COMPLETED** event is delivered to state machine and the state machine set own state to **WAITING_FOR_TRIGGER**
- ✓ The **state of MARTE application** is exported as a PV named **STATUS**



Interface of EPICS with MARTe

● EPICSLib

- ✓ EPICSLib **instantiates an EPICS CAS** while in the MARTe configuration file the description of the Process Variables to be setup on the CAS
 - Ex) loads process variables TARGET_X0 and SHOT_NUMBER
- ✓ The current version of the code **mimics the listed EPICS records**
 - bi, mbbi, ai, longin, waveform (From FPCS)
 - **bo, ao** : supported newly for this task (New)
- ✓ The “**mimic**” in the sense that all fields required by those records can be specified in the MARTe configuration file and are used to generate events such as alarms, monitoring and archiving
- ✓ The supported fields of EPICS PV are following:

NAME	TYPE	LEN	PREC	SYNC	EGU	HOPR	LOPR	HYST	ADEL
HIHI	HIGH	LOW	LOLO	HHSV	HSV	LSV	LLSV	MDEL	SCAN

- ✓ EPICSLib has two threads.
 - One **handles messages from EPICS**
 - Another **handles value of signals written by the EPICSGAM**
- ✓ The **RunOnCPU** parameters **allocate each thread to the dedicated CPU**

➤ MARTe Configuration File

```
+EPICSLib =
{
  Class = EPICSHandler
  //PREFIX = "MARTe:"
  debugLevel = 0
  scanOn = true
  asyncScan = true
  asyncDelay = 0.1
  maxSimultAsyncIO = 100
  RunOnCPU = 2
  RunOnCPU_event = 4

  ProcessVariable = {
    TEST-PCS:TARGET_X0 = {
      NAME = "TEST-PCS:TARGET_X0"
      TYPE = aitEnumFloat32
      SYNC = excasIoSync
      SCAN = "I/O Intr"
    }
    TEST-PCS:SHOT_NUMBER = {
      NAME = "TEST-PCS:SHOT_NUMBER"
      TYPE = aitEnumInt32
      SYNC = excasIoSync
      SCAN = "I/O Intr"
    }
  }
  ...
}
```

● EPICSGAM

- ✓ A component that lives in the **Real Time Thread** and **interacts with the DDB reading signals** on it. Then it **copies them to the EPICSLib**
- ✓ To copy signals to the EPICSLib, EPICSGAM searches EPICSLib object in the **GlobalObjectDataBase**
- ✓ Every **Process Variable** the system has to export to EPICS must be **configured in EPICSLib**
- ✓ The **map** between MARTe signals and EPICS Process Variables must be **defined in the EPICSGAM** configuration section
- ✓ At initialization, EPICSGAM after locating EPICSLib reads all signal descriptions and tries to subscribe its interest for each read signal to EPICSLib
- ✓ The EPICS **update rate can be configured** through **ServerSubSampling** parameter

➤ MARTe Configuration File

```
+Thread_1 = {
...
+EPICSStream = {
  Class = EPICSGAM
  UseTimeSignalName = useTime
  NOfAcquisitionSamples = 2000
  SignalsServer = "EPICSLib"

  EventTrigger = {
    TimeWindow0 = {
      NOfSamples = 2000
      UsePeriod = 1000
    }
  }

  Signals = {
    TEST-PCS:TARGET_X0 = {
      SignalName = "TEST-PCS:TARGET_X0"
      ServerName = "TEST-PCS:TARGET_X0"
      SignalType = float
      ServerSubSampling = 100
    }
  }
}
...
}

Online = "Timer PcsProc UDPOutput
DACBoard WebStatistic
Collection EPICSStream"
Offline = "Timer PcsProc UDPOutput
DACBoard WebStatistic
EPICSStream"
}
```

Interface of EPICS with MARTe

● GAM for writing PVs to signals

- ✓ EPICSGAM allows only **reading signals on the DDB**
- ✓ New GAM was implemented to support **writing signals, parameters and states via CA**
- ✓ Writing signals should be defined in the **OutputSignals** section
- ✓ When any PV is changed, exPV class derived from casPV class sends a **MARTe message** to the GAM. This GAM acts as a device support of EPICS, classifies the requests into writing value, sequential operation and configuration of GAM
- ✓ The requested writing value is stored to the DDB

➤ MARTe Configuration File

```
+Thread_1 = {
  +PcsProc = { // GAM
    ...
    OutputSignals = {
      TEST-PCS:TARGET_X0 = {
        SignalName =
          "TEST-PCS:TARGET_X0"
        SignalType = float
      }
    }
  }
}
```

```
caStatus exPV::write ( const casCtx &, const gdd & valueIn )
{
  if ( valueIn.applicationType() == gddDbrToAit[DBR_PUT_ACKT] )
    return this->putAckt( valueIn );
  else if ( valueIn.applicationType() == gddDbrToAit[DBR_PUT_ACK] )
    return this->putAcks( valueIn );
  caStatus ret = this->update ( valueIn, true, true );
  double value;
  this->pValue->get( value);

  // Send a MARTe message to the GAM
  // for writing signal value on the DDB
  CodacMessage::SendMessage (CODAC_MSG_TARGET_PCSPROC,
    0, "WRITE_SIGNAL", getName(), value);
  return (ret);
}
```

➤ **exPV.cpp**

```
bool CodacPcsProc::ProcessMessage(GCRTemplate<MessageEnvelope> envelope) {
  GCRTemplate<Message> message = envelope->GetMessage();
  FString messageContent = message->Content();
  FString messageSender = envelope->Sender();

  if(messageContent == "WRITE_SIGNAL"){
    GCRTemplate<GCNString> signalName = message->Find(0);
    GCRTemplate<GCNString> signalValue = message->Find(1);

    double value;
    sscanf (signalValue->Buffer(), "%lf", &value);

    return (WriteSignal (signalName, value));
  }
  else if(messageContent == "STORE_RESULT"){
    iocStatus.setStatus (CODAC_TASK_STANDBY);
  }

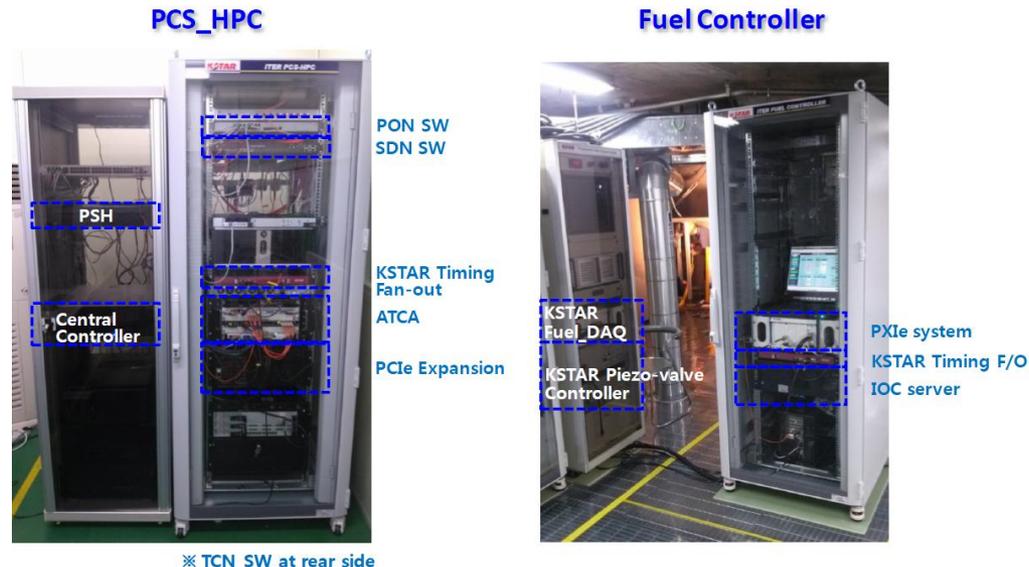
  return False;
}
```

➤ **GAM**

Evaluating performance of MARTe

● Results of performance evaluation of the MARTe framework

- ✓ The performance is mainly improved by **Real-time O/S** itself
- ✓ Secondly, the improvement can be achieved by tuning such as **CPU affinity** and **priority**
- ✓ MARTe provides a method to support the real-time functionality in a framework environment, and in the case of Linux, the **CPU affinity and priority for the real-time threads can be designated in the application configuration file**
- ✓ A certain degree of **overhead is identified in the application based on MARTe framework**, and its difference from reference program in performance is more remarkable on MRG-R rather than on RHEL 5.5. The added **latency due to MARTe is about 17 μ s** but **this latency increase is not considered large proportion in considering the performance of the entire fast control system**
- ✓ The addition of interfaces with EPICS and MDSplus gives little degradation in real-time performance



- The MARTe (Multi-threaded Application Real-Time executor) framework has been evaluated as a candidate real-time framework for a real-time feedback control of the plasma density for a Plasma Control System (PCS)
- Portable Channel Access Server (pCAS) has been adopted into the MARTe application to interface with EPICS
- The MARTe application with interface of EPICS has been operated successfully in KSTAR sequential operation mode
- **Future works are following:**
 - ✓ Improve the Writing GAM as a generic GAM
 - ✓ Support more record types such as stringin and stringout
 - ✓ Comparison the performance of affinity patched EPICS with MARTe

Acknowledgments and references

● Acknowledgments

- ✓ This work was performed within the cooperation defined in the Memorandum of Understanding (MoU) between ITER International Organization (IO) and NFRI
- ✓ This work was supported by ITER CODAC Team and MARTE community
- ✓ This work used the results of ITER FPSC Project by ITER IO and FPSC Team

● ITER

- ✓ Fast Controller Workshop, Anders Wallander, Feb28, 2011

● Portable Channel Access Server

- ✓ Channel Access Server Tool Developers Training, Jeff Hill, Kay-Uwe Kasemir, LANL
- ✓ Portable Channel Access Server, Marty Kraimer, US Particle Accelerator School

● MARTE

- ✓ MARTE Framework – Middleware for RT Control Development, Andre Neto, et. al, IPFN/IST, FPSC Workshop
- ✓ Performance Comparison of EPICS IOC and MARTE in a Hard Real-Time Control Application, A. Barbalace, RFX
- ✓ RFX-mod Feedback Control System Upgrade, G. Manduchi, A. Barbalace, RFX
- ✓ MARTE in fusion, L. Zabeo & MARTE team, ITER Fast Controller Workshop
- ✓ MARTE-EPICS Integration for the ATCA FPSC, Bernado B. Carvalho, et. al, Jan 25, 2012, IST
- ✓ ITER FPSC Project, MARTE to EPICS Process Variable Interface
- ✓ MARTE-EPICS A Wining Combination for the ATCA FPSC, Bernado B. Carvalho, et. al, Feb 11, 2011, Diagnostics & Data Acquisition

Thanks for your attention