COMPARATIVE ANALYSIS OF EPICS IOC AND MARTe FOR THE DEVELOPMENT OF A HARD REAL-TIME CONTROL APPLICATION*

A. Barbalace†, A. Luchetta, G. Manduchi, C. Taliercio, Consorzio RFX, Padova, Italy
B.B. Carvalho, D.F. Valcárcel, IPFN, Lisboa, Portugal

Abstract

EPICS is used worldwide to build distributed control systems for scientific experiments. The EPICS software suite is based around the Channel Access (CA) network protocol that allows the communication of different EPICS clients and servers in a distributed architecture. Servers are called Input/Output Controllers (IOCs) and perform real-world I/O or local control tasks. EPICS IOCs were originally designed for VxWorks to meet the demanding real-time requirements of control algorithms and have lately been ported to different operating systems.

The MARTe framework has recently been adopted to develop an increasing number of hard real-time systems in different fusion experiments. MARTe is a software library that allows the rapid and modular development of stand-alone hard real-time control applications on different operating systems. MARTe has been created to be portable and during the last years it has evolved to follow the multi-core evolution.

In this paper we review several implementation differences between EPICS IOC and MARTe. We dissect their internal data structures and synchronization mechanisms to understand what happens behind the scenes. Differences in the component based approach and in the concurrent model of computation in EPICS IOC and MARTe are explained. Such differences lead to distinct time models in the computational blocks and distinct real-time capabilities of the two frameworks that a developer must be aware of.

INTRODUCTION

Developing a real-time control application is a complex task if it has to be designed from the ground up. Everything must be checked and carefully reviewed to guarantee bug-free code and bounded execution times. Likely there exist many different programming environments to code a real-time control system and many are suited for scientific experiments. In this work the EPICS [1] and the MARTe/BaseLib [2] (in the following referred as MARTe) software frameworks are analyzed.

The EPICS software suite is built around the Channel Access (CA) network protocol that allows the communication of different EPICS clients and servers in a distributed architecture. Servers are called Input/Output Controllers (IOCs) and perform real-world I/O or local control tasks.

MARTe is a software library that allows the rapid and modular development of stand-alone hard real-time control applications on different Operating Systems (OSs).

As far as we know there are no published comparative analyses of software frameworks for the development of control applications, especially involving EPICS and MARTe. This work is intended to complement an earlier paper [3] where we presented a performance comparison of the two software frameworks.

We analyze EPICS version 3.14.11 and MARTe CVS version June 2011. All facts presented here are OS-independent, whereas paper [3] refers to Linux PREEMPT_RT. In this paper we address several implementation differences and similarities of the two software frameworks which are both written in the C/C++ language.

ARCHITECTURAL OVERVIEW

Component Based Approach

Both EPICS IOC and MARTe present a modular architecture where components can be connected to carry out the required control algorithm. Components in EPICS IOC are implemented by *records*, while in MARTe they are implemented by *Generic Application Modules* (GAMs). Records are logically grouped in *databases*, GAMs belong to a *Real Time Thread*.

In EPICS new record types (or *Record Supports*) can be created by providing a set of routines and data structures adhering to a given C prototype; the newly created record will carry out the required computation. For every record type it is possible to define a new *Device Support* that enables the record to interact with an I/O device.

New components in MARTe are created by subclassing the GAM class or any other descendant of the GAM class. By subclassing, all the functionalities required to properly interact with the MARTe environment are inherited by the new component. In MARTe only two special subclasses of the GAM can handle data communication with a device driver: `InputGAM` and `OutputGAM`. Such classes can be associated to an I/O device via a *Generic ACQuisition Module* (GACQM), similar to what is called Device Support in an EPICS IOC.

Configuration Capability

The current configuration of both frameworks, i.e. the sets of component instances and the way they are intercon-

* This work was set up with financial support by Fusion for Energy.
† telephone: +39 049 8295039, e-mail: antonio.barbalace@igi.cnr.it

nected, is defined in one or more text files which get parsed to produce the target application.

In an EPICS IOC the records that populate a database are declared in a configuration file. In MARTe a configuration file contains not only the functional blocks (GAMs) of the control algorithm but also instance descriptions of generic C++ classes. Such instance descriptions can be specified to extend the functionality of the target MARTe application. As an example, to monitor the data exchanged by the GAMs in a control loop an instance description of a Web Server class can be included in the configuration file.

In both EPICS and MARTe a control application is described by a configuration file. The way in which such files are used to create the application differs between the two frameworks.

Creating a New Application In EPICS a configuration file is used to generate IOC source code before compilation. Instead, in MARTe, a configuration file is loaded by a running MARTe system.

Both frameworks come with a set of base components but the developer may write its own components (records or GAMs). When a new record is ready to be tested in EPICS the developer has to create a new project, insert the record with all ancillary files in the project, create the configuration files, compile and run the project. In MARTe the development process is straightforward: after compiling the new component the developer has only to write a configuration file (actually by modifying a template).

Updating a Running Application The value and compound data (database links included) of a record can be tuned at runtime being each field implemented by a Process Variable (PV). To load or remove records not previously loaded in EPICS a user has to reconfigure and recompile all the application from scratch and then run it again.

In MARTe in order to change a GAM parameter it is necessary to stop the execution of every MARTe activity and reload the framework. It is however not necessary to recompile the application. A new component which allows the runtime modification of parameters and signal values called *Configuration Library* (CL) [4] was recently introduced.

Internal Data Structures

In EPICS IOCs every record field is a PV, and in order to manage thousand of PVs, EPICS makes use of hash tables for PV lookup.

MARTe is mostly based on linked lists and sophisticated data structures have not been introduced. Data querying at runtime is still lacking.

Both frameworks create during initialization all data structures that are required at runtime. It is the responsability of the developer to implement efficient (time-bounded) code in the `process` routine (EPICS IOC) or in the `execute` method (MARTe) of a computational block. In a hard real-time application unbounded delays due to badly written algorithms or network accesses could deteriorate the control.

CONCURRENT MODELS OF COMPUTATION

Concurrent models of computation (CMoC) are of great interest when dealing with embedded systems, especially feedback controllers. A model of concurrence specifies how interaction, communication and control flow will be handled between components in a software application. In the following the models of computation of EPICS and MARTe are described.

EPICS IOC Record Processing

A single record in an EPICS IOC can be accessed concurrently for reading, writing or processing. Reading and writing can trigger processing, before fetching a value (demand-driven) and after updating a value (data-driven).

The processing of a chain of records takes place within a *scan* (an operating system's thread). A scan can be locally triggered periodically (*periodic scan*) or by software and hardware events (*event scan*, *IO event scan*); remotely by `caGet` or `caPut` operations. After the processing has been triggered on the first record of a chain those records that have to be executed next are determined by the associated *links* in the database. A link can carry data and processing (INLINK, OUTLINK) or simply processing (FWDLINK). A chain of records is executed until there exist processing links or records which can be passively processed by other scans (*passive scan* in EPICS terminology). Very complex processing chains can be designed in EPICS mixing demand (INLINK, FWDLINK) and data (OUTLINK) -driven processing.

Because a single record can be concurrently accessed by different scans, per-record locks exist. Synchronization delays due to concurrency can deteriorate the real-time response of a control algorithm. Typically such a situation arises when a record is concurrently updated locally, belonging to a processing scan chain, and remotely by a CA operation (Fig. 1). The local operation issued by the EPICS IOC has lower priority then the remote operation.

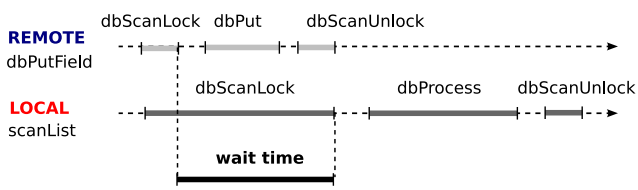

Figure 1: In EPICS a locally triggered scan may need to wait for the end of the processing of a remotely invocated record.

MARTe GAM Processing

MARTe has been designed as a software framework for hard real-time, low-latency applications. It eliminates concurrency because it is a source of delays and uncertainties in code execution. MARTe is a lock-free execution environment where each GAM executes serially in a thread (`RealTimeThread`) to which it is uniquely tied. MARTe has been designed to support a multiprocessor environment and the developer is encouraged to map a RealTimeThread per CPU.

GAMs communicate only within thread bounds where communication is made safe by the inherently serial execution of the thread. Inter-thread communication exists, but is managed by special GACQMs. GAMs communicate by means of a thread-level data buffer called *Dynamic Data Buffer* (DDB); every GAM knows how to access their private data area. This model implies a strictly data-driven processing.

As mentioned in the previous section, parameter and signal values can be modified via CL. Remotely changing a value in a running MARTe application is possible and is completely asynchronous with the execution of the RealTimeThread (highest priority task).

Processing Details

Stack Frame MARTe is less memory resource-starved than EPICS. A long EPICS' chain of passive records with many links can produce an out-of-stack exception, while MARTe will never exhibits this problem because the number of execution frames on the stack is $O(1)$ while in EPICS it is $O(n)$ where n is the number of passive chained records.

Processing Links Links in MARTe define only the flow of data and therefore they establish the data dependency between computational blocks. In an EPICS IOC links define the flow of data and processing.

Data Conversion at Interfaces Data diversity between blocks can be handled by the framework to let blocks with different data interfaces communicate. Data conversion can be managed at the node level via polymorphic get/put routines (Fig. 2) or by means of an external (global) adapter (Fig. 3). EPICS supports data diversity by means of a rich set of protocol adapters; MARTe does not support data diversity: two communicating GAMs must adhere to the same data interface.

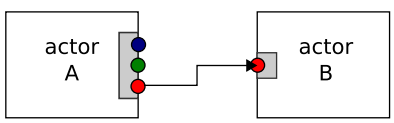

Figure 2: Actor A shown a polymorphic interface: the same **get** method will be overloaded to return different data types in function of the data type required by actor B.

Fanout In electronic circuits the output from one Integrated Circuit (IC) can be connected to several ICs. The fanout indicates how many ICs can be connected to a single output. The same concept also applies to records and GAMs. In MARTe one output signal from a GAM can be connected to as many GAMs as needed.In EPICS it is possible to have many INLINKs to the same record's field but it is not possible to have an OUTLINK or a FWDLINK connected to more than one record's field; to fulfill this need a `fanoutRecord` is supplied with EPICS.

Data Tokens In both software the processing is due to a new data token and there is no buffering between nodes and the number of tokens to be consumed in any processing of a node is always one.

CONCURRENT MODELS OF TIME

The natural way to code a control algorithm is via discretization of a continuous time model. The discretized algorithm is not aware of time, moreover it is usual practice to maintain a discrete function $f(n)$ instead of $f(Tn)$ where T, the sampling period, is explicit in the formula. The algorithm runs after the samples are transferred from the data acquisition subsystem, once per sampling period. The period T is usually fixed a priori during the design phase.

Coding the Algorithm in MARTe

In MARTe each GAM is triggered once per period and the time elapsed between calls to GAMs is equivalent to period T. Periodic timing is provided by a GACQM interfaced with a `TimeInput` GAM. Algorithmic GAMs are aware of the absolute time and of the execution period T.

Coding the Algorithm in EPICS

EPICS' records can be processed concurrently and lock mechanisms have been developed to synchronize their processing. In EPICS no time model is assumed, a record can be processed at any time. Careful design of an EPICS IOC database is required to achieve periodic processing of a record. To develop a record, a user have to take into account the possibility that a discrete algorithm will be executed in a non-uniform sampling environment.

The smart way chosen in EPICS to develop a PID controller (`pid`, `cpid` and `epid` records) is to accept that the

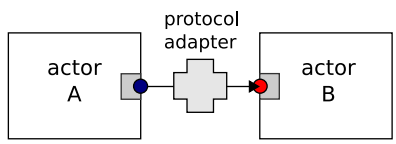

Figure 3: Both actors have a rigid interface: a protocol adaptor is necessary. A's **get** method returns a unique data type and B's **put** method fetches a precise data type.

PID will be called at non-uniform time intervals, and to define a minimum amount of time between record processings. This is defined in the Minimum Delta Time (MDT) field of the PID record. If the amount of time between the last time the record was processed and the current time is less than MDT, then the record is not processed.

REAL-TIME SUPPORT

Both EPICS IOC and MARTe have been used on VxWorks installations but this is not enough to state that they are hard real-time software frameworks. In [3] we have compared the same test control loop implemented as an EPICS IOC and as a MARTe application on a Linux PREEMPT_RT system. EPICS IOC exhibits a wider jitter compared to the MARTe counterpart. In this section we figure out from where those sources of jitter come.

Event Triggering

In MARTe the way in which a device driver (GACQM) waits for a hardware event is selectable (polling or interrupt) and defined in the configuration file. In EPICS the developer can not choose how to wait for an event, it depends on the device driver. Polling may be advantageous because it can eliminate the OS's process switching (which could introduce further delays due to scheduling other tasks).

Event Dispatching

In EPICS every event is queued for execution and never lost. There are different queues of execution and the execution policy is FIFO. One queue handles CA requests; three queues (low, medium and high priority) are shared between events and IO events and there is one queue per periodic scan (Fig. 4). This mechanism decouples an event from the associated servicing code but can generate unbounded latencies. Different scans that originate from different interrupt sources but with the same priority are enqueued on the same queue. Such an approach can introduce considerable jitter in the generated waveforms and task deadline misses. MARTe does not queue any event: events can be lost but there is no jitter due to events enqueuing.

Exploiting Multiprocessors

MARTe allows the developer to assign threads and IRQs to specific processors. This facility protects the execution and promotes a deterministic response of the thread. The affinity mask of threads and IRQs on the processors is a parameter in the MARTe configuration file.

The EPICS Operating System Interface (OSI) layer does not support multiprocessor environments, i.e. `epicsThreadCreate` does not accept any CPU affinity argument. A static set of queues is defined for each instance of EPICS IOC. This approach does not scale to multiprocessors.

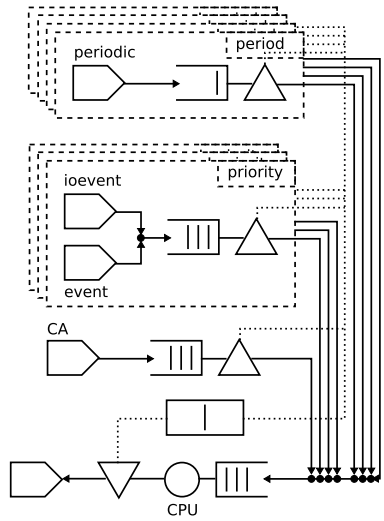

Figure 4: **EQN model** of the EPICS events queuing system running on a single processor/core machine.

FINAL CONSIDERATIONS

Since EPICS has been on the scene since 1989, the number of tools available to design and configure a system has grown. The same does not hold for the real-time support and especially for the exploitation of emerging processor technologies (multiprocessor architectures). Coding a discretized control algorithm can be tricky.

MARTe, on the other hand, has so far only a limited set of design and configuration tools. Some work is ongoing to exploit the Ptolemy [5] project as a tool for configuring and simulating MARTe applications. MARTe appears to be crafted for real-time, low latency applications. To cope with such demanding requirements it exploits emerging multi-core architectures.

REFERENCES

[1] A. Johnson. (2007) Epics about [Online]. Available: http://www.aps.anl.gov/epics/about.php

[2] A.C. Neto, F. Sartori, F. Piccolo, R. Vitelli, G. De Tommasi, L. Zabeo, A. Barbalace, H. Fernandes, D.F. Valcárcel, and A.J.N. Batista, "MARTe: A Multiplatform Real-Time Framework", *IEEE Transactions on Nuclear Science*, vol. 57 no. 2, pp. 479 – 486, Apr. 2010.

[3] A. Barbalace, G. Manduchi, A. Neto, G.D. Tommasi, F. Sartori, and D.F. Valcárcel, "Performance comparison of EPICS IOC and MARTe in a hard real-time control application", *Real Time Conference (RT), 2010 17th IEEE-NPSS*, pp. 1 – 5, May. 2010.

[4] D.F. Valcárcel, A. Barbalace, A. Neto, A.S. Duarte, D. Alves, B.B. Carvalho, P.J. Carvalho, J. Sousa, H. Fernandes, B. Gonalves, F. Sartori, and G. Manduchi, "EPICS as a MARTe configuration environment", *IEEE Transaction on Nuclear Science*, vol. 58 no. 4, pp. 1472 – 1476, Aug. 2011.

[5] U. B. E. Department. (2011) Ptolemy project home page [Online]. Available: http://ptolemy.eecs.berkeley.edu/