PROTOTYPE OF A DDS-BASED HIGH-LEVEL ACCELERATOR APPLICATION ENVIRONMENT*

N.Malitsky, J.Shah, BNL, Upton, NY 11973, USA
N.Hasabnis, Stony Brook University, Stony Brook, 11794 NY, USA
R.Talman, Cornell University, Ithaca, 14853 NY, USA
S.Shasharina, N.Wang, Tech-X Corp., Boulder, CO 80303, USA

Abstract

The paper presents a prototype of the NSLS-II high level application environment including key middle layer servers, such as Machine, Online Model and Virtual Accelerator. The proposed environment is developed and evaluated on top of EPICS-DDS, an open source implementation of the DDS standard interface based on the EPICS Channel Access protocol.

BACKGROUND

A typical accelerator control system employs a three-tier distributed environment encompassing low-layer distributed front-end computers controlling physical devices, a set of middle layer servers maintaining common data structures and algorithms, and an open collection of high level thick and thin client applications. Despite the common infrastructure, requirements of each layer are different. As a result, in modern accelerator facilities, the middle layer servers also work as gateways connecting at least two communication protocols and interfaces. This paper presents a new homogeneous infrastructure for the NSLS-II high-level accelerator environment based on the Experimental Physics and Industrial Control System (EPICS [1]) and the data-centric publish-subscribe model of the OMG Data Distribution Service (DDS [2]). In this approach (see Figure 1), different layers and components communicate via the common EPICS protocol, and the high-level interface between servers and applications is implemented as the EPICS-DDS extension [3].

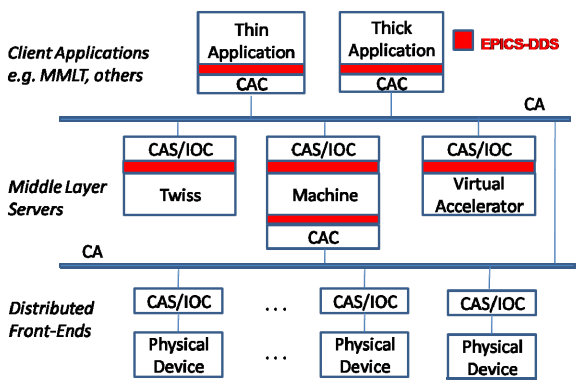

Figure 1: High-Level application environment based on the EPICS-DDS extension.

*Work supported by DOE contract DE-AC02-98CH10886

EPICS is an open source framework and a rich collection of tools developed collaboratively and used worldwide for building distributed real-time control systems in large-scale scientific projects: accelerators, detector systems, telescopes and others. Its base infrastructure consists of two layers: distributed Input/Output Controllers (IOC) and Operator Interface (OPI) applications. IOC provides uniform interfaces to heterogeneous physical devices. The heart of IOC is a memory resident database with a collection of records. Each record instance, called process variable (PV), has a record name, a value, and a type-specific set of associated fields, such as operating range, alarm limits, scanning rate, and so forth. Client applications can access IOC records via the Channel Access (CA) interface. The CA communication interface is based on the TCP protocol and is highly optimized for dealing with hundreds of thousands of process variables.

A limitation of the present version of EPICS is that process variables cannot be structures and are limited to primitive data types and arrays of scalars. This constraint seriously mismatches with requirements of high-level object-oriented applications. This necessitates the integration of additional middleware such as CDEV, CORBA, JMS, or DDS.

DDS is a next generation of middleware industrial standards, bringing a data-centric publish-subscribe (DCPS) architecture to distributed control systems. The overall conceptual model is shown in Figure 2 and encapsulates the following major concepts:

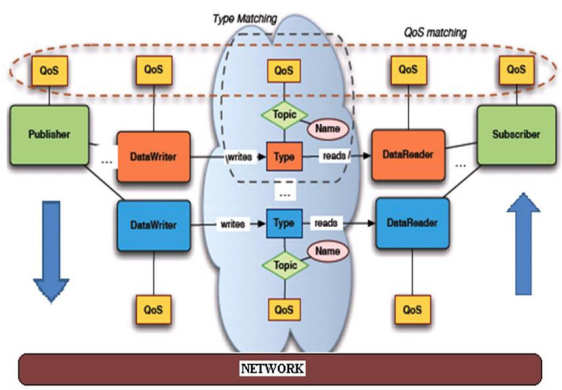

Figure 2: Data-centric publish-subscribe model [4].

- Topics of Typed Global Data Space: a logical data space in which applications read and write data decoupled in space and time.
- Data Writer: data producer of the given topic.
- Data Reader: data consumer of the given topic. Data reader can obtain data two ways: (1) listener-based asynchronous mechanism and (2) waitset-based synchronous approach that blocks the application until designated conditions are met.
- QoS policies: a rich set of characteristics that define the behavior of the DDS systems (such as reliability, liveliness, durability, etc.)

Similar to the OMG CORBA specification, DDS is language neutral and can be implemented in programming languages like Java and C++. In contrast with CORBA, the DDS interface is also protocol neutral, facilitating its deployment in different distributed systems. Moreover, the DDS technology extends present run-time environments with the relational-oriented model. It creates a basis for developing consistent run-time interfaces to complex hierarchical structures using well-established software engineering techniques.

EPICS Channel Access and DDS represent unique approaches and our initial task was to identify the common concepts decorated by these different terminologies. The developer's intuition and optimism was based on the fact that both technologies have a long history of successful projects in neighboring domains. Moreover, the overall goal was practical: understanding how the DDS data-centric model can extend the CA interface for developing high-level applications. As a result, the initial scope of the EPICS-DDS middleware was narrowed to the following three DDS concepts expressed in CA terms:

- Topic: collection of record fields belonging to the different instances of the same record type.
- Data Reader: maintainer of the CA channels for getting data from the collection of record fields associated with the Data Reader's topic.
- Data Writer: maintainer of the CA channels for putting data to the collection of the record fields associated with the Data Writer's topic.

These suggested associations explicitly identify the core architecture of EPICS-DDS; setting the DDS Typed Global Data Space onto the EPICS I/O Controllers and considering the DDS participants as wrappers of the CA clients. Starting from this point, we consistently and incrementally continued to adapt different features of the CA interface towards the three-tier high level application environment. The initial list of these features (such as IOC-based middle layer server, asynchronous and synchronous cases, hosting and serializing user-specific structures based on waveform records) was described in the previous EPICS-DDS paper [3]. In this report, we combine them to build a high-level accelerator application environment including key middle layer servers, such as Machine, Online Model and Virtual Accelerator.

MIDDLE LAYER

According to the uniform scenario of the three-tier accelerator application environment, the different middle layer servers provide the states (the most current values) of the associated data structures shared by other servers and high-level client-subscribers. The majority of accelerator use cases require three data types: collection of the accelerator element parameters, design linear and non-linear optics functions, and turn-by-turn data measured or calculated in beam position monitors.

The description of the accelerator structure and accelerator devices is a key part of any accelerator algorithm. Designing the generic accelerator description is a serious challenge even in the context of off-line object-oriented programs. The UML model of the Accelerator Description eXchange Format (ADXF 2.0 [5]) represents one of the recent approaches addressing different types of accelerator computational tasks (see Figure 3).

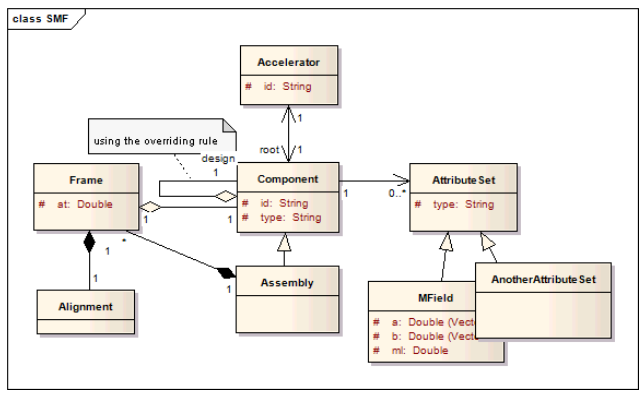

Figure 3: ADXF 2.0 model of the Machine server.

In the off-line environment, this object-oriented model has been mapped and deployed in several representations including the XML Schema and the IRMIS relational database. Adherence of the DDS specification to the relational model allows reuse of the same mapping procedure. Particularly, the prototype uses the following three topics with the associated data structures:

- AccComponent: accelerator component name and type (e.g. quadrupole);
- AccAssembly: a named sequence of frames with installed accelerator components;
- AccStrength: accelerator component name and primary type-specific attribute (e.g. quadrupole strength). In the next version, this structure will be extended with the composite attribute id for supporting arbitrary attributes.

In the context of the DDS specification, these topics are maintained and published by the Machine server. Other middle layer servers and high-level clients subscribe to these topics to synchronize their own containers.

Particularly, the Twiss and Virtual Accelerator servers recalculate and update their own states of the design optics and turn-by-turn beam data respectively.

One of the important criteria for selecting a high-level communication interface was support of user-specific data types. In DDS, it is based on an additional pre-processor procedure generating the corresponding Data Reader and Data Writer classes from the CORBA IDL file describing the new user-specific data structures. Recently, the EPICS-pvData project [6] introduced a more dynamic approach derived after the Type Object pattern. According to this approach, different user-specific classes are implemented as instances of a single class PVStructure containing an array of PV Field's. The content of this array is defined in the corresponding StructureType object that can be created dynamically or registered in the introspection service. The pvData approach complements the DDS IDL-based procedure without mismatch, since the PV Structure can be considered as a legitimate data type. In the EPICS-DDS environment, PV Structure's Reader and Writer classes were implemented directly in a framework and used for processing all accelerator-specific middle layer containers.

The middle layer servers themselves were built on the EPICS Portable CA Server (PCAS [7]) framework. The structures of the Machine and Twiss servers, for example, are shown in Figure 4.

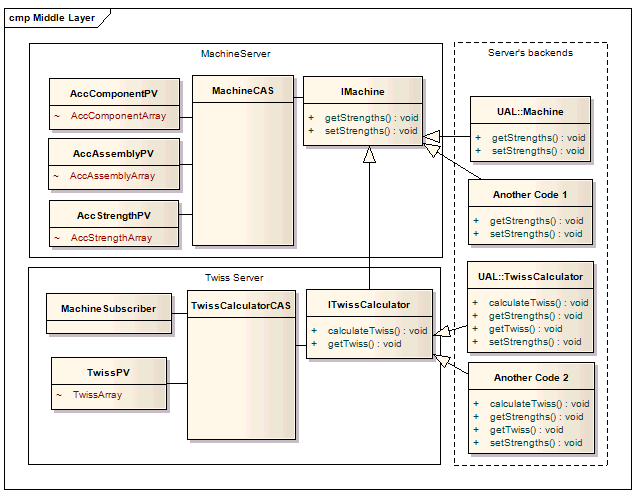

Figure 4: PCAS-based Machine and Twiss servers.

In the spirit of the Service- Oriented Architecture, these online services can be configured with the different off-line accelerator programs. The present prototype was tested with the Unified Accelerator Libraries (UAL [8]) backend.

CONCLUDING REMARKS

This paper presents a prototype high-level accelerator application environment based on EPICS-DDS, a new middleware extension of EPICS, approaching the Data Distributed Service (DDS) interface based on the EPICS protocol. The integration of these two technologies addresses five major tasks. First, DDS brings an industrial standard interface to the accelerator online environment allowing a variety of high-level applications and toolkits to be decoupled from the underlying low-level control systems, such as EPICS, TINE, TANGO, and others. Second, the DDS topic-oriented approach elevates the EPICS Channel Access protocol to high-level applications, replacing their RPC-like communication interfaces. Third, DDS creates a basis of Service-Oriented Architecture (SOA) promoting decoupling of the service interfaces from their project-oriented implementations. In the context of high-level application environment, it provides flexibility in selecting and connecting the most appropriate modelling algorithms and programs. Fourth, the DDS specification introduces some guidance for extending the EPICS infrastructure with the relevant set of qualities of service. Finally, the DDS technology extends the EPICS run-time environment with the relational model, facilitating the design of consistent run-time interfaces to complex hierarchical structures according to well-established software engineering techniques, such as object-relational mapping. Moreover, adherence to the relational approach creates a platform for integration of full-scale Data Stream Management Systems (DSMS) for data stream processing and archiving.

The positive experience gained from this project encourages us to further explore and extend the EPICS-DDS middleware in the development of the full-scale high-level accelerator application environment.

ACKNOWLEDGEMENTS

The EPICS-DDS project has been shaped and consolidated from numerous discussions and valuable inputs of members of Control and Accelerator Physics groups of the NSLS-II project. We would like also to thank B. Dalesio and S. Stoller for their support.

REFERENCES

[1] L.Dalesio *et al*., The Experimental Physics and Industrial Control System Architecture, ICALEPCS'93 http://www.aps.anl.gov/epics/.

[2] OMG, Data Distribution Service for Real-time Systems, Version 1.2, formal/07-01-01.

[3] N.Malitsky, J.Shah, N.Hasabnis, EPICS-DDS, PAC'09 http://sourceforge.net/projects/epics-dds.

[4] A.Corsaro, Advanced DDS Tutorial, Real-Time and Embedded Systems Workshop, July 2008.

[5] N.Malitsky and R.Talman, Accelerator Description Formats, ICAP'06.

[6] M.Kraimer et al., EPICS-pvData, http://sourceforge.net/projects/epics-pvdata.

[7] J.Hill. A Server Level API for EPICS, ICALEPCS'95

[8] N.Malitsky and R.Talman, UAL, AIP 391, 1996, http://code.google.com/p/ual.