EVALUATING THE OMG DATA DISTRIBUTION SERVICE (DDS) FOR ACCELERATOR CONTROL SYSTEMS*

N. Wang†, S. Shasharina, Tech-X Corporation, Boulder, CO 80303, U.S.A.

Abstract

As accelerators become bigger, traditional ways of building the control systems based on a single framework no longer scale. A Service-Oriented Architecture (SOA) adopting both a Remote-Procedure-Call (RPC) and a Message-Oriented Middleware (MOM) standard service buses promotes multiple levels of loose coupling to increase the robustness and adaptability of overall control applications without sacrificing performance. The emerging OMG DDS specification defines a data-centric communication standard with rich supports for quality-of-service (QoS). It is especially suited as the SOA's MOM service bus for control systems. DDS helps extend the modules built with proven control development frameworks such as Experimental Physics and Industrial Control System (EPICS,) into standard services in a SOA environment and facilitate flexible data exchanges between services. In this paper, we review various features in the OMG DDS standards and their applications in high-level control applications. We also illustrate how, collaborating with Web Services, DDS fits into a SOA for accelerator control systems. Finally, we present how we are evaluating performance benchmarking results of several DDS implementations, including an EPICS-DDS, which is an open source implementation of OMG DDS.

BACKGROUND AND INTRODUCTION

Accelerator control systems (ACS) coordinate the interactions among control hardware, data acquisition instruments, logging and data storage devices, and operator's interface. High-level accelerator control applications encompass activities such as operator control panels, tune measurement, orbit control, parameter save/restore, feedback, optic optimization, and parameter scanning, to allow physicists and operators to control and reason accelerator behaviours in physically meaningful abstractions. There exist many tools and frameworks to help bring modern software engineering practices to the development and integration of lower-level hard real-time controls and the high-level soft real-time applications with great success.

Emerging Trends and Challenges

Control systems are often built on top of a set of existing tools and platforms that suit the needs of their target. For examples, the EPICS [1] toolkit provides a standard for low-level controller architecture and a set of interoperable tools and engineering applications to assist control system developments. Depending on the scale of the target accelerator, high-level applications are often developed as a monolithic Graphical User Interface (GUI), a simple script, or a library routine. As in the case of generalized ACS control environments, many tools and frameworks such as Unified Accelerator Language (UAL) and Matlab Middle Layer Toolkit (MMLT), are available to assist the integration and interaction among high-level applications and device controllers (built, *e.g.*, using EPICS.)

All the different development environments and tools do not generally interoperate with one another. This is not a major issue for small- or medium-sized accelerators. However, such *ad hoc* approach no longer scales for modern large-scale accelerator facility such as the new NSLS-II, Project X, and the International Linear Collider (ILC). This challenge is reflected in both the ILC Reference Design Report and the NSLS II Preliminary Design Report which both call for a separate "service tier/middle layer" to provide device and functional abstractions as units of integrations.

SOA: SERVICE-ORIENTED ARCHITECTURE

SOA [2,3] has gained wide-spread acceptance in the business/enterprise software world as it has shown to facilitate the integration and composition of disparate software services across enterprises and businesses boundaries. SOA is neither a single technology nor as a silver bullet to address the challenges facing large-scale IT application development. Rather, SOA represents technology-independent, high-level concepts that provide architectural blueprints for these systems. These architectural blueprints focus on the partitioning enterprise application into components that are created and exposed as services, and the composition of services to realize an enterprise mission. Applying SOA principles in ACS is a promising approach in isolating and managing the complexity. In fact, many existing accelerator control systems have already adopted many SOA guidelines and principles. To address the needs and challenges of next-generation, large-scale accelerator control systems, we are developing a SOA environment for next-generation large-scale accelerator control systems to manage the complexity and contain the cost of developing future accelerator control systems and upgrading existing ones.

ENABLING SOA FOR HIGH-LEVEL APPLICATIONS IN CONTROL SYSTEMS

Both the SOAP-based and RESTful Web Services have long been viewed as the middleware for middleware, i.e., they are used as an interoperability mechanism among diverse middleware technologies such as CORBA and

*Work supported by US Department of Energy under contract DE-FG02-08ER85043, and Tech-X Corporation.
†nanbor@txcorp.com

Java RMI, that are used for building smaller scaled distributed applications. Web Services have long been the *de facto* standard service bus for enterprise and e-commerce applications due to its simplicity. There are alternative efforts to adopt other more efficient and versatile middleware standards such as CORBA, in a SOA to address the performance and lack of features issues. However, there are still limitations with these standards. Specifically, the point-to-point, request-reply, RPC-styled communication model may impose scalability issues as the size and complexity of future accelerator control systems grow.

Message-Oriented Middleware

The emerging DDS [4] is a new class of MOM standard specified by the Object Management Group (OMG) that complement RPC-styled client-server middleware while address their many limitations. DDS is a natural extension to many existing accelerator control frameworks. In particular many correlated EPICS' Process Variables (PV's) can be made available as DDS topics. Because DDS inherently support many quality-of-service (QoS) policies such as message priority and deadline, that are necessary for mission-critical applications, DDS is a natural selection to act as the alternative service-bus in a SOA for control systems.

Figure 1 illustrates an SOA for high-level accelerator environment where applications exchange data using DDS as the common standard service bus. As shown in the figure, client applications can readily act as gateways to another enterprise service bus such as Web Services.

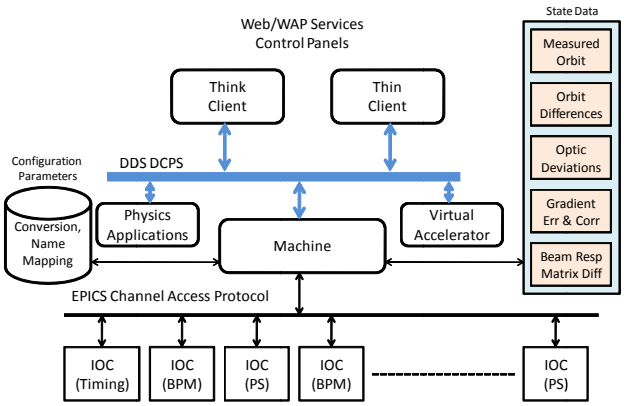

Figure 1: SOA for High-Level Application over DDS.

Overview of DDS

At the core of DDS is the Data-Centric Publish-Subscribe (DCPS) model. As shown in Figure 2, the specification defines standard interfaces that enable applications running on heterogeneous platforms to write/read data to/from a global data space in a distributed system. Applications that want to share information with others can use this global data space to declare their intent to publish data that is categorized into one or more topics of interest to participants.

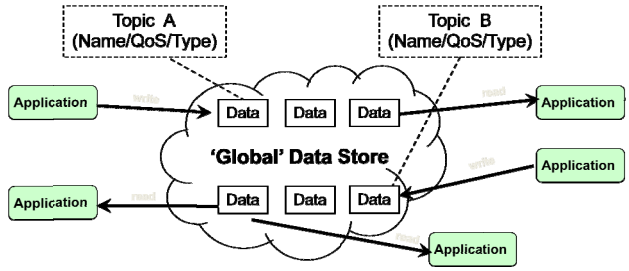

Figure 2: Data-centric publish/subscribe interaction model in DDS.

Similarly, applications that want to access topics of interest can also declare their intent to become subscribers with in the same data space. The underlying DCPS mechanism propagates data instances written by publishers into the global data space, where it is disseminated to subscribers interested in the information. The DCPS model decouples the declaration of information access intent from the information access itself and thereby enables the DDS middleware to support and optimize QoS-enabled communication under the hood without close intervention from participating applications.

EPICS-DDS

Other than evaluating the use of commercial and open-source DDS implementations for ACS, we will also evaluate the use of EPICS-DDS [6] for ACS. EPICS-DDS is an open source implementation of the OMG's DDS based on the Channel Access (CA) protocol of EPICS. It exposes EPICS variables using DDS APIs to support the data-centric communication model in ACS. Under this approach, the higher-level applications and server uses the DDS standard interfaces to publish and subscribe data topics. These DDS interfaces then propagate the data using the common EPICS CA protocol to update the corresponding PV's residing at a EPICS server. In the context of DDS specifications, the different middle layers servers are considered as the corresponding DDS publishers designed to provide the states of the associated data structures shared by high-level client-subscribers.

EVALUATING DDS THROUGH PERFORMANCE TESTING

Accessing the performance of the messaging middleware and the overall systems and applications provide critical information to help making key design decision such as:

- **Selection of DDS implementations:** Different DDS implementations make different tradeoffs and adopt different implementation strategies to realize the movement of data from publishers to subscribers according to the QoS policies specified by all the entities involved. As a result, certain implementations perform better under certain operation environment while others scale better. It is important to understand how they behave under the environments applications are intended to run.

- **Assisting hardware design:** For large-scale systems, it is possible that the amount of messages exchanged across the wire may just be too much for the middleware to maintain the necessary QoS even though there is enough bandwidth to transmit all the data. It will be helpful to foresee such limitations and revise the design accordingly, e.g., separate traffic by adding a new Ethernet.
- **Assisting in DDS configuration:** DDS supports a rich set of QoS policies. Configuring a system using different set of policies can affect the overall performance in different ways. For example, setting priority on one data stream can affect the overall behaviours of other data streams. Being able to perform tests to observe system behaviours at similar scales can provide essential guidance on design strategies.

Benchmarking Infrastructure and Target Scenarios

Existing DDS performance tests, however, are often one-off solutions that measure system behaviours in a simple environment. To address the limitation, we are building a performance measurement tool suite by combining two different approaches. First, we will develop the generic benchmarking application similar to the open-source Touchstone performance tool. Similar to that of Touchstone's, the benchmarking application can be configured to instantiate test components such as transceiver and transponder for latency test, via special messages of a set of control topics. Users can design and instantiate tests of different scales and with different combinations of QoS policies easily. Such approach allows users to create scenario-based performance evaluations easily.

Another challenge that our test suite will address is the difficulty in deploying large scale test bed over distributed environments. We plan to leverage existing deployment and configuration framework to remotely deploy benchmarking test applications, be it written in C, C++, or Java.

We plan to perform application scenarios relevant to ACS. For example, for high-level application integration, we will implement an optimization framework that couples machine, online model such as Tracey, via a common DDS Twiss parameters topic. We will configure the test to measure the application and middleware performance in a similar scale and QoS policies. Moreover, connecting ACS data to GUI control panel and other Web Services/WAP servers to present data in a timely manner has great potential. We also plan to demonstrate system behaviours under such scenarios.

CONCLUDING REMARKS

This paper describes how a dual service-bus SOA provides an ideal development environment that promotes medullisation of high-level application components and facilitates dynamic composition of high-level ACS applications. In the context of the high-level application environment, it means flexibility in selecting and connecting the most appropriate modelling algorithms and programs. To support such environment, we are bringing the next generation data-centric publish-subscribe middleware called DDS by investigating various usage scenarios and programming patterns of DDS in ACS. Furthermore, we are developing a DDS performance test suite and performing benchmarking tests to evaluate various open source and commercial DDS implementations.

Other than readily available DDS implementations, we are also evaluating and contributing to the development of a new extension to EPICS that supports DDS interfaces on top of the CA protocol. The integration of these two technologies provides many benefits. First, DDS brings an industrial standard middleware to the accelerator online environment, allowing the decoupling of a variety of high-level applications and toolkits from the underlying low-level control system frameworks such as EPICS. Conversely, the DDS topic-oriented approach elevates the EPICS Channel Access protocol to the high-level applications. Third, the DDS specification introduces some guidance for extending the EPICS infrastructure with the relevant set of quality of service. Finally, the DDS technology extends the EPICS run-time environment with the relational database model creating a platform for relational queries and integration of full-scale Data Stream Management Systems (DSMS) for data stream processing and archiving.

To assist ACS developers to take full advantage of DDS, we are developing tools to help them to understand how various design decisions affect overall system performance. Furthermore, we are designing tools to help facilitate the design and configuration of DDS.

REFERENCES

[1] L. Dalesio *et al*., "The Experimental Physics and Industrial Control System Architecture," ICALEPCS'93, Berlin, Germany, October 1993, http://www.aps.anl.gov/epics/

[2] Dirk Krafzig and Karl Banke and Dirk Slama. Enterprise SOA – Service-Oriented Architecture Best Practices. Prentice Hall, 2005.

[3] Eric Newcomer and Greg Lomow. Understanding SOA with Web Services. Addison Wesley, New Jersey, 2005.

[4] OMG, "Data Distribution Service for Real-time Systems, Version 1.2," formal/07-01-01, http://www.omg.org/cgi-bin/doc?formal/07-01-01

[5] M. Kraimer *et al*, "EPICS Application Developer's Guide," January 2009.

[6] M. Nikolay *et al*, "Prototype of a DDS-based High-Level Accelerator Application Environment," ICALPCS09, Kobe, Japan, Oct 2009.