A MODULAR ENVIRONMENT FOR HIGH LEVEL APPLICATIONS

Guobao Shen, BNL, UPTON, NY 11973, U.S.A.

Abstract

For a modern large-scale accelerator complex such as NSLS-II, a high level application environment is indispensable and plays important role from beam commissioning to daily operation during its whole life time. There are many particular environments available and those environments have been well developed and been using at different facilities. Those existing environments usually are closed environments, provide their own application toolset, and talk with their build-in simulation engines. Their applications tie together functions through internal data or file structure, and that makes it difficult to share applications or algorithms between each other. To solve that problem, a modular environment for high level applications is proposed at NSLS-II and under development. The purpose of this new environment is to modularize the application, and build up a client-server based environment. This paper describes recent progress of our modular environment, particularly our development for model based control, and IRMIS database based applications.

INTRODUCTION

A dedicated computer software environment for accelerator beam control and manipulation, accelerator status control and analysis, which is usually known as a HLA (High Level Application), plays more and more important roles during machine study, commissioning, and daily operation. Conventional HLA environments usually are closed environments, provide whole set of application tools, and have their build-in simulation engine. Those environments have been used in many facilities and demonstrated that a unique environment is very efficient for the beam commissioning, operation, and maintenance during its whole life time. However, those environments tie together functions through internal data or file structure to provide a self-consist environment. None of them include any portions of the others. That makes it difficult to share applications or algorithms between each other. If desired application or algorithm does not exist in a selected environment, users have to develop their own even it is available in another environment. Eventually, each environment provides similar functionality with each other, and many duplicate applications and algorithms have been developed.

A novel architecture based on client-server for HLA is proposed at NSLS-II (National Synchrotron Light Source II) to address that problem. It tries to modularize each application, and makes all applications pluggable. One application provides one unique function, and is made as a standalone module. They communicate with each other through a unified communication protocol. For example, for a model based control, the simulation engine is built as a server. An application gets model parameters, twiss parameters for example, from model server instead of calling simulation code within itself. With this approach, a HLA developer can dedicate on new development and improvement, avoid duplicated development, and save the man-power.

Since the NSLS-II control system [1] is determined to adopt EPICS (Experimental Physics and Industrial Control System) as its device control framework, the development for our modular environment is designed and developed against EPICS system.

The paper describes the recent progress about our modular environment development. Section II gives an overview of system architecture. In Section III, model server and model based control are described. In Section IV, IRMIS aware application and IRMIS server are shown. Section V concludes the paper.

MODULAR ARCHITECTURE

Our effort is to develop a modular environment based on client-server architecture. Detailed design can be found in [2], and illustrated as Fig 1 briefly.

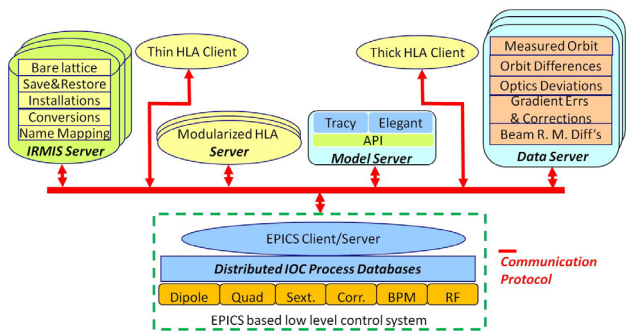

Figure 1: System architecture for a modular environment for high level application.

As showed in Fig. 1, we have 5 application categories that are explained as below:

1. Model server. A simulation engine runs in a model server, and beam parameter such as beam optics is provided by this server. Multiple simulation servers are allowed in our architecture. A narrow and well defined API [1] is developed at NSLS-II to provide transparent access to different simulation engines. At NSLS-II, we choose Tracy-3 [3] and Elegant [4] for our development;
2. IRMIS (Integrated Relational Model of Installed Systems) [5] database server. The IRMIS database is designed for a general purpose for accelerator complex. It can be used for documenting accelerator hardware and its installation, cabling, and system configuration. In this paper, we will introduce using to manage lattice and its revolution, and for model based control;
3. Application server. The application server is a server family; each application server provides dedicated functionalities such as archiving, EPICS

processing variable logging, alarm handler, and so on. It can be a server that provides model based control, optics matching for example, or a general purpose server, providing numerical algorithm for example;

4. Data server. The data server store all data from measurement and simulation such as measured beam orbit, measured response matrix, response matrix from online model;
5. Application client. It could be either a thin client or a thick client according requirement. The presentation will be done in this layer, and most important rule is to develop a GUI (Graphic User Interface) to accelerator control, commissioning, and operation.

Application is modularized into standalone server, and each application is loose coupling with others, modular, pluggable, and therefore reusable. All servers and clients communicate with EPICS based low level device control system through a unique communication protocol known as channel access.

MODEL SERVER

An accelerator is usually designed with a methodology so-called MBD (Model Based Design). There are 4 steps to use MDB for an accelerator design:

1) Establishing a geometric sequence for accelerator;
2) Analyzing and synthesizing all controllers, particularly magnets;
3) Simulating and optimizing design;
4) Integrating all these phases by deploying it.

With all above steps, various considerations have been performed during the design, for example single particle dynamics and realistic magnetic lattice, and a realistic model can be established.

Bunches of computer simulation codes have been developed for that purpose, and most are used widely to study the beam behaviors and verify designer's consideration. Some challenges have been addressed in the simulation codes such as how to combine the numerical methods for modeling of a realistic lattice with the analytical techniques for analysis.

It is nature to adopt a MBC (Model Based Control) to re-use the realistic model and related algorithms developed during the design phase to control and manipulate a beam. The MBC method is important for an accelerator beam study, commissioning, and operation. It provides an efficient approach for establishing communication between a realistic model and physical machine. In terms of control requirements, a realistic accelerator model contains information that enables prediction of the beam properties, beam orbit for example, of changing process operating conditions, magnetic strength for example. Conventional solution to use MBC is to build the simulation code into application through some variety of data structures, files, or methods. However, this approach replies on its build-in code and none of them have capability to talk with other simulation code.

At NSLS-II, a standalone server for MBC is under development. A server for MBC is developed already against current EPICS release (v3) as described in [2], but some new requirements arise during the development. Most important requirement is how to organize data efficiently, change the data structure dynamically and transfer data to client according client specified filters. However those functionalities are lack of current EPICS release. Our current model server cannot provide those functions.

Fortunately, those functionalities will be addressed in next major EPICS release (v4) [6]. A significant improvement in EPICS v4 is a pvData (Process Variable Data), which is a definition and implementation of memory resident data. The pvData provides an efficient way to store, access, and transfer memory resident data. All memory resident data is organized as structured data with a PVStructure container. The EPICS channel access protocol is updating to transfer the structured data over network.

IRMIS SERVER

IRMIS is a RDB (Relational DataBase) tools which originally was developed at ANL (Argonne National Laboratory) for documenting. It is a collaborative effort now between several EPICS facilities including ANL, and BNL. The IRMIS has the capability to capture all of the system parameters needed to document the installation, the process variables of control system, the lattice of realistic model, and the revolution history.

Lattice Database

With IRMIS database, each lattice configuration for realistic model can be captured and restored according user requirement. A library has been prototyped to fetch lattice information from a Tracy or Elegant deck, write them into IRMIS lattice database, and extract a particular lattice from IRMIS as shown in Fig. 2.

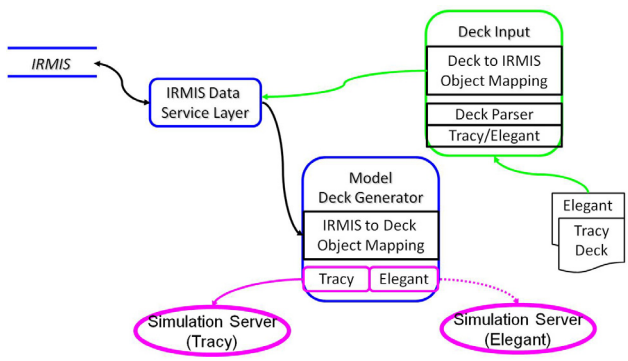

Figure 2: Schematic of using IRMIS to manage lattice.

Different simulation code has different definition for accelerator element type. To simplify IRMIS lattice, we define an IRMIS aware element type for different accelerator components. The lattice deck to IRMIS object mapping is finished in the library when writing into or extracting from IRMIS lattice database. The library

communicates with IRMIS through IRMIS data service layer [7].

IRMIS Server

The lattice database records bare lattice information, including element geometric location, and designed magnetic strength. Both location and strength are ideal value, and we hope to combine them with engineering error or alignment offset. It is required to use live magnetic strength instead of design when producing a realistic model.

The lattice information is stored in lattice database as shown in Fig. 3. The engineering error and alignment offset are stored in another IRMIS database, installation database, and the live strength is saved in IRMIS Log database. We are going to use two mapping tables to associate the IRMIS lattice database with installation database and with the log database. A library is under development. Once we finish it, an IRMIS server will be created which provides the capability to feed lattice into model server directly from IRMIS database.

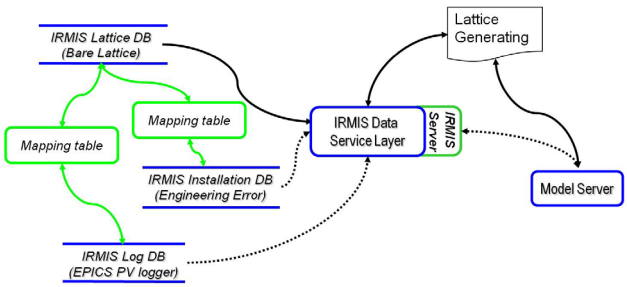

Figure 3: Schematic of extracting realistic lattice and using IRMIS server to feed model server directly.

SUMMARY

A modular environment for high level application is proposed at NSLS-II project. The goal for this environment is to make application modular, distributed, pluggable, and therefore reusable. The system consists of model server for model based control, IRMIS server to track various machine parameters and configuration, data server for data store, and application server family. Some detailed design and implementation is presented for the model server and IRMIS server.

ACKNOWLEDGEMENT

The author would like to thank Johan Bengtsson for his helpful discussions and comments on the model server development. He wants to thank Donald Dohan and G. Carcassi for their contribution on the IRMIS database and IRMIS data service layer. He also wants to express this thanks to Leo Bob Dalesio for his continuous support and encouragement.

REFERENCES

[1] G. Carcassi, D.Dohan, et.al "NSLS II Control System", this proceeding, TUP104.

[2] G. Shen, "A software architecture for high level applications", Proceeding of PAC 2009, FR5REP004, May 2009, Vancouver.

[3] J. Bengtsson, "TRACY-2 User's Manual", SLS Internal Document, February 1997; M. Boege, "Update on TRACY-2 Documentation", SLS Internal Note, SLS-TME-TA-1999-0002, June 1999.

[4] M. Borland, "elegant: A Flexible SDDS-Compliant Code for Accelerator Simulation," APS LS-287, 2000.

[5] IRMIS:http://www.aps.anl.gov/epics/irmis/index.php; D. A. Dohan, L. R. Dalesio, G. Carcassi, "High Availability On-Line Relational Databases for Accelerator Control and Operation", Proceedings of PAC 09, May 2009, Vancouver.

[6] M. R. Kraimer, L. R. Dalesio, K. Zagar, M. Sekoranja, "Evolution of the EPICS Channel Access Protocol", this proceeding, MOD005.

[7] G. Carcassi, "A REST Service For IRMIS3", this proceedings.