A SOFTWARE ARCHITECTURE FOR HIGH LEVEL APPLICATIONS *

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

Abstract

A modular software platform for high level applications is under development at the National Synchrotron Light Source II project. This platform is based on client-server architecture, and the components of high level applications on this platform will be modular and distributed, and therefore reusable. An online model server is indispensable for model based control. Different accelerator facilities have different requirements for the online simulation. To supply various accelerator simulators, a set of narrow and general application programming interfaces is developed based on Tracy-3 and Elegant. This paper describes the system architecture for the modular high level applications, the design of narrow and general application programming interface for an online model server, and the prototype of online model server.

INTRODUCTION

A software platform so-called a HLA (High Level Application) environment is indispensable in order to efficiently commission, operate and maintain a modern large-scale accelerator complex such as the NSLS-II [1] (National Synchrotron Light Source II) complex. A good HLA environment usually provides rich software tools to efficiently operate the beams, tune accelerator parameters, record system parameters, save online components setting, restore machine to previous status, archive live data, analyze measured and/or archived data, compare machine status with design, and so on. Different accelerator facilities developed different HLA environments such as MMLT[2] (Matlab Middle Layer Toolkit), which was developed jointly by ALS/LBNL and SPEAR3/SSRL, XAL [3] by SNS/ORNL, and SDDS[4] (Self Describing Data Sets) by APS/ANL. Each environment provides systematic and unified control of accelerator components, and has seamless integration with low level device control. Those environments have been used at many different accelerator facilities for many years, and have been demonstrated to be effective, and stable.

Those different environments provide similar functions, but all software applications provided by those environments tie together functions through data or file structures. It is very difficult to share applications between each other. Someone has to develop his own application or algorithm from scratch if it is not supported by his HLA environment.

A new environment for the HLA is under design based on the client-server architecture and preliminary research has been doing at NSLS-II project. With this architecture, the components of high level applications will be modular and distributed, and therefore reusable. To take advantage of existing HLA environment, the design is started based on the MMLT because it is based on a Middle Layer environment and the concept is closer to our goal than other environments.

Another problem for existing HLA environment is that each environment has a built-in simulation engine, but only supports its own engine. A standalone online model server is indispensable to construct a modular and distributed HLA environment. As a general online server, it is needed to support different simulators because different facilities have different requirements and therefore require different simulators. A set of narrow and general APIs (Application Programming Interfaces) is developed for the online server to support various simulators. The API is designed and prototyped based on Tracy-3 [5] and Elegant [6] simulation code.

This paper describes the design of the modular architecture for HLA environment, the narrow API development for online model server, and the prototypes for the online server.

SYSTEM ARCHITECTURE

An accelerator control system consists of 2 parts: low level device control and high level application as illustrated in Figure 1. A software toolkit known as EPICS [7] (Experimental Physics and Industrial Control System) has been used widely around the world for the low level device control. NSLS-II project has adopted EPICS as its standard platform for the control system construction [10] to control all hardware subsystems such as magnet power supply, and beam instrumentation. The communication between low level device control and HLA is based on the EPICS channel access protocol [8].

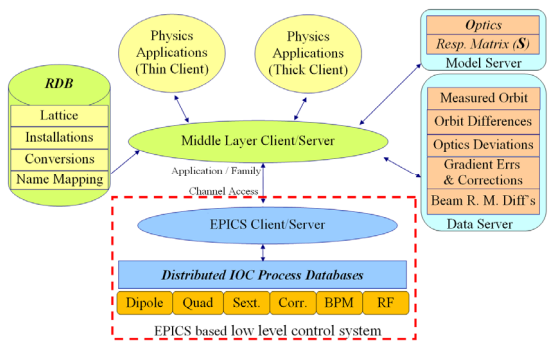

Figure 1: Architecture for a modular HLA environment.

As shown in Figure 1, different applications with different functionalities are distributed in different servers. The Middle Layer client-server, which is heart of the modular environment, is an application family. The application family consists of many different applications, such as data server, model server, relational database server, and other thin/thick applications. The data server takes charge of measuring and caching beam live orbit,

*Work performed under auspices of the U.S. Department of Energy under Contract No. DE-AC02-98CH10886 with Brookhaven Science Associates, LLC.

calculating orbit differences and beam optics deviations, measuring beam response matrix, and so on. It will be a collective server and locate on several computers physically. The model server runs site specified simulator, loads lattice configuration from relational database, and calculating theoretical beam optics and other desired beam parameters. The thin/thick application server could be an archiving server to archive interested data at desired repetition rate, a GUI (graphic user interface) to present machine real time status, or others. A relational database server such as IRMIS [11] can be used to store the information of lattice design, hardware component installation, machine status snapshot, and so on. Each application or server can be plugged into the Middle Layer client-server.

With this platform, the components of each application can be modularized. The development for algorithm, GUI presentation, and other application can be separated by well design and developer can dedicate on their own development. It is very flexible for user because any user can plug in his own application developed with his favorite programming, or his own simulator without changing other application. It will be easy to reuse exist application module.

API DESIGN FOR ONLINE MODEL

Model based control is one most important component for a HLA environment. The heart for the model based control is its online simulator, which will run as an online model server. As describes above, a narrow and general API interface is required to support various simulators. Our design was started based on the MMLT toolkit because its structure especially the concept of Middle Layer, which is shown in Figure 2, is closer to our architecture than other environments. Another reason is that the MMLT toolkit is using by many light source facilities, and there are various applications available. Since the NSLS-II is a light source, we can port the MMLT applications to our platform easily.

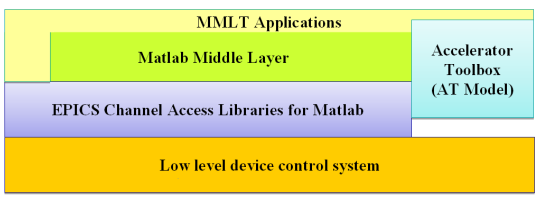

Figure 2: Software structure of MMLT.

We enhanced the MMLT to support Tracy-3 and Elegant simulators, and clarified the border between application and model. Meanwhile, we prototyped the narrow API interface.

Narrow Interface Design

As describe above, the MMLT environment provides its own simulator which is known as the AT [9] (Accelerator Toolbox). The AT code is implemented in pure matlab language, and integrated in the MMLT seamlessly.

The data structure is the heart part of MMLT, and all applications tie together through data including the AT model. The border between applications and between application and AT model is not very clear.

By analyzing the MMLT data flow, we found that the data required by applications from the AT model is the beam optic parameters. We defined a set of API to get data and set configuration.

Because Tracy-3 and Elegant code are written in C/C++, the API interface is developed in C/C++ naturally. The API code is compiled with Tracy-3 or Elegant into a shared library, which can be called by a matlab command. The system structure is illustrated as Figure 3.

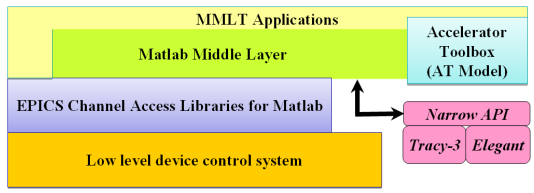

Figure 3: Enhanced MMLT and narrow API design.

All communication between MMLT and Tracy-3 or Elegant is through the narrow API interface, and border is clear between each other. The API consists of 3 parts: (a) simulator configuration and initialization; (b) beam optics and other beam parameters; and (c) miscellaneous. The part (a) for both Tracy-3 and Elegant is similar, and has slight difference. The part (b) is exact same for both simulators. The part (c) is a place holder for future extension.

Applications for NSLS-II

The MMLT configuration for Tracy-3 supporting has been full developed and some MMLT applications have been tested against Tracy-3. We did not complete the configuration for Elegant because of 2 reasons. (a) One is the configuration development is not our major target, and we are much more interested in the narrow interface. (b) Another one is that the architecture for all simulators is same, especially the interface for beam optics is exact same. We assume that, the configuration can be used for Elegant with tiny modification.

Some MMLT applications are tested against Tracy-3 without any modification such as plotfamily for plotting, quadrupole center, chromaticity measurement, response matrix measurement, and so on. Figure 4 shows a beta-plotting for 2 planes plotted by plotfamily.

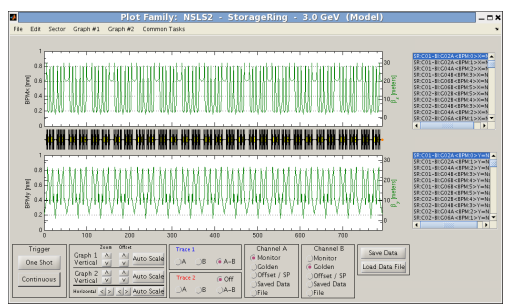

Figure 4: Plotfamily runs against Tracy-3. The upper part plots the beta in horizontal plane, and the lower part plots the beta in vertical plane.

Figure 5 shows another MMLT application to measure the chromaticity. Figure 5-a is the result running against Tracy-3 simulator, and Figure 5-b is the result running against AT. The application code works for both simulators without any modification. The results from different simulators are same.

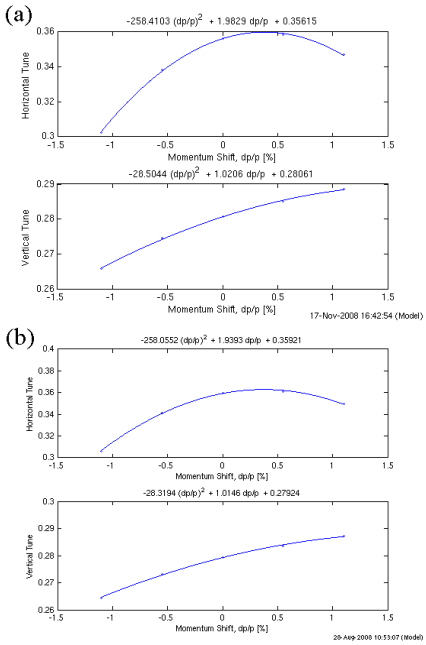

Figure 5: Chromaticity measurement against different simulators: (a) is the result against Tracy-3, and (b) is the result against AT.

ONLINE SERVER PROTOTYPE

With above interface, a prototype for online model server is developed. A so-called "virtual accelerator" (VA) is introduced for the development. The VA is originally for high-level application developing, testing and debugging at an early stage. The VA developed by SOLEIL and DLS (Diamond Light Source) consists of Tracy-2 tracking code, and wrapped the simulator into EPICS system. We updated it to Tracy-3 and communicated with the EPICS system through our API interface as illustrated in Figure 6.

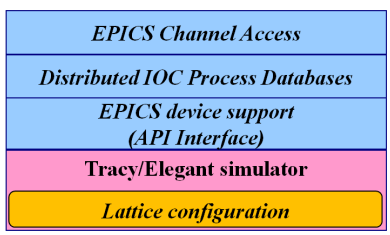

Figure 6: Structure of current online model server.

As shown in Figure 6, we also wrapped the Elegant code into EPICS device support using the same API interface. We found that there is memory leak problem inside the Elegant code and it caused the server unstable. The Elegant library can be called through the EPICS about a few thousands time. But the API interface works for both simulators.

SUMMARY

A modular architecture is proposed at NSLS-II project to make the high level application modular, distributed, and therefore reusable. A narrow and general API interface for online model server is designed and prototyped by enhancing current MMLT environment to support various simulators such as Tracy-3 and Elegant. Some MMLT applications have been tested against Tracy-3 after that without any modification. With the API interface, a standalone online model server has been prototyped based on a "virtual accelerator". Both Tracy-3 and Elegant have been integrated into the online model server. The narrow API interface works well in both MMLT environment and online model server.

ACKNOWLEDGEMENT

The author would like to thank Johan Bengtsson, Weiming Guo, Donald Dohan, and Boaz Nash of BNL and Michael Borland of ANL for their helpful discussions and comments. He also wants to thank Leo Bob Dalesio for his continuous encouragement and support.

REFERENCES

[1] http://www.bnl.gov/nsls2.
[2] G. Portmann, J. Corbett, A. Terebilo, "Middle Layer Software Manual for Accelerator Physics," LSAP-302, 2005; J. Corbett, A. Terebilo, G. Portmann, "Accelerator Control Middle Layer," PAC 2003.
[3] J. Galambos, et al, "XAL Application Programming Framework", Proceedings of ICALEPCS 2003, Gyeongju, Korea.
[4] http://www.aps.anl.gov/Accelerator_Systems_Division/Operations_Analysis/manuals/SDDStoolkit/SDDStoolkit.html
[5] J. Bengtsson, "TRACY-2 User's Manual", SLS Internal Document, February 1997; M. Böge, "Update on TRACY-2 Documentation", SLS Internal Note, SLS-TME-TA-1999-0002, June 1999
[6] M. Borland, "Elegant: A flexible SDDS-compliant code for accelerator simulation," Advanced Photon Source Light Source Note LS-287 (2000).
[7] http://www.aps.anl.gov/epics/.
[8] http://www.aps.anl.gov/epics/base/R3-14/10-docs/CAref.html.
[9] A. Terebilo, "Accelerator Toolbox for MATLAB", SLAC-PUB-8732 (2001).
[10] L. Dalesio, "NSLS-II Control System", Proceedings of ICALEPCS 2007, TPPB41, Knoxville, Tennessee, 2007, USA;
[11] D. A.Dohan, L. R. Dalesio, G. Carcassi, "High Availability On-Line Relational Databases for Accelerator Control and Operation", this conference