ONLINE PHYSICS MODEL PLATFORM*

P. Chu#, V. Vuppala, Y. Zhang, FRIB, East Lansing, MI 48823, USA
D. Dohan, G. Shen, BNL, Long Island, NY 11973, USA
J. Wu, SLAC, Menlo Park, CA 94025, USA
C. Benatti, NSCL, East Lansing, MI 48823, USA

Abstract

For a complex accelerator such as the Facility for Rare Isotope Beams (FRIB), a transfer matrix based online model might not be sufficient for the entire machine. On the other hand, if introducing another modelling tool, physics applications using modelling tools have to be rewritten. A platform that can host multiple modelling tools would be ideal for such scenario. Furthermore, the model platform along with infrastructure support can be used not only for online applications but also for offline purposes, for example multi-particle tracking simulation. In order to achieve such a platform, a set of common physics data structures has to be set. XAL's accelerator hierarchy based data structure is a good choice as the common structure for various models. Application Programming Interface (API) for physics applications should also be defined within a model data provider. A preliminary platform design and prototype is discussed.

INTRODUCTION

For modern accelerators, online model is necessary for beam control and physics studies. The XAL [1] online model is not sufficient for many physics problems and the runs are not in near real time. There are many other simulation codes for accelerator modelling. Each one has some strength but not all. To utilize their strength, an approach is to provide a platform to host multiple modelling tools. In order to achieve such a platform, a set of common physics data structure has to be set. Application Programming Interface (API) for physics applications should also be defined within a model data provider. On the other hand, the software infrastructure for such model platform should be robust. To make such platform useful for beam control, the performance should also be greatly improved for many beam dynamics codes.

OVERVIEW

The model codes are the engine for beam computation. A supporting infrastructure, which prepares model input parameters, sets up a model run and packages model run result, is called Model Server [2]. The Model Server then serves up model data via a commonly used communication protocol to its client applications. In this paper, a set of client applications for the model service is also identified.

Model platform is capable of running some predefined model codes. Model server should contain the following parts:

Model Data Structure

It is necessary to define a common data structure as an interface between various modelling tools and physics applications. The XAL data structure provides a hierarchical view of an accelerator which can be the common format for model input data. Model output data from various modelling tools is saved in relational database (RDB).

Model Data adaptor

Each specific code needs a conversion between the common data structure and its own format for both model input and output. For most modelling results, it is not difficult to write a parser, e.g. a Matlab script was written to parse IMPACT model result [3].

Model Data Storage

Modern RDB is a good choice for model data storage for its efficiency and maintainability. Furthermore, RDB provides functions such as searching and sorting which can be useful for offline analysis.

Model Data for Client Applications

For typical access to the model data in RDB, it is more efficient to define intuitive API methods such as *getTwissForElement("The_Element")*. For performance or management reasons, a computer server with proper protocols is configured to fulfil any client applications' model data needs. The data access API and the database schema should be matched as close as possible; otherwise, it may cause inefficient or unnecessarily complicated queries.

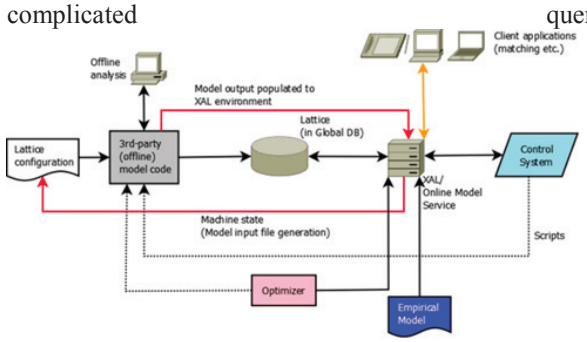

Figure 1: Model Platform schematic diagram.

*Work supported by the U.S. Department of Energy Office of Science under Cooperative Agreement DE-SC0000661 and DE-AC02-98CH10886 and Michigan State University.
#chu@frib.msu.edu

ISBN 978-3-95450-115-1

Model Platform

Fig. 1 is a schematic design for the Model Platform. Both offline and online model data are saved in a RDB. XAL provides the hook to control systems and fitted empirical machine model via XAL's data structure and API. A model control program with adapter to a specific third party modelling code can prepare and initiate a model run for that modelling code. In addition to the data structure and API, XAL also provide an envelope based online model. On the other hand, offline code can also port its model output data back to the control systems through provided XAL API. An optimizer library with modelling code can provide beam optimization. The optimizer library should be general enough and residing outside XAL so non-XAL programs can take advantage of the optimizer.

In the model platform design, database is the central piece of the architecture; any clients using model data will only communicate with the database instead of any specific modelling code. Details of the essential model platform components are described in the next section.

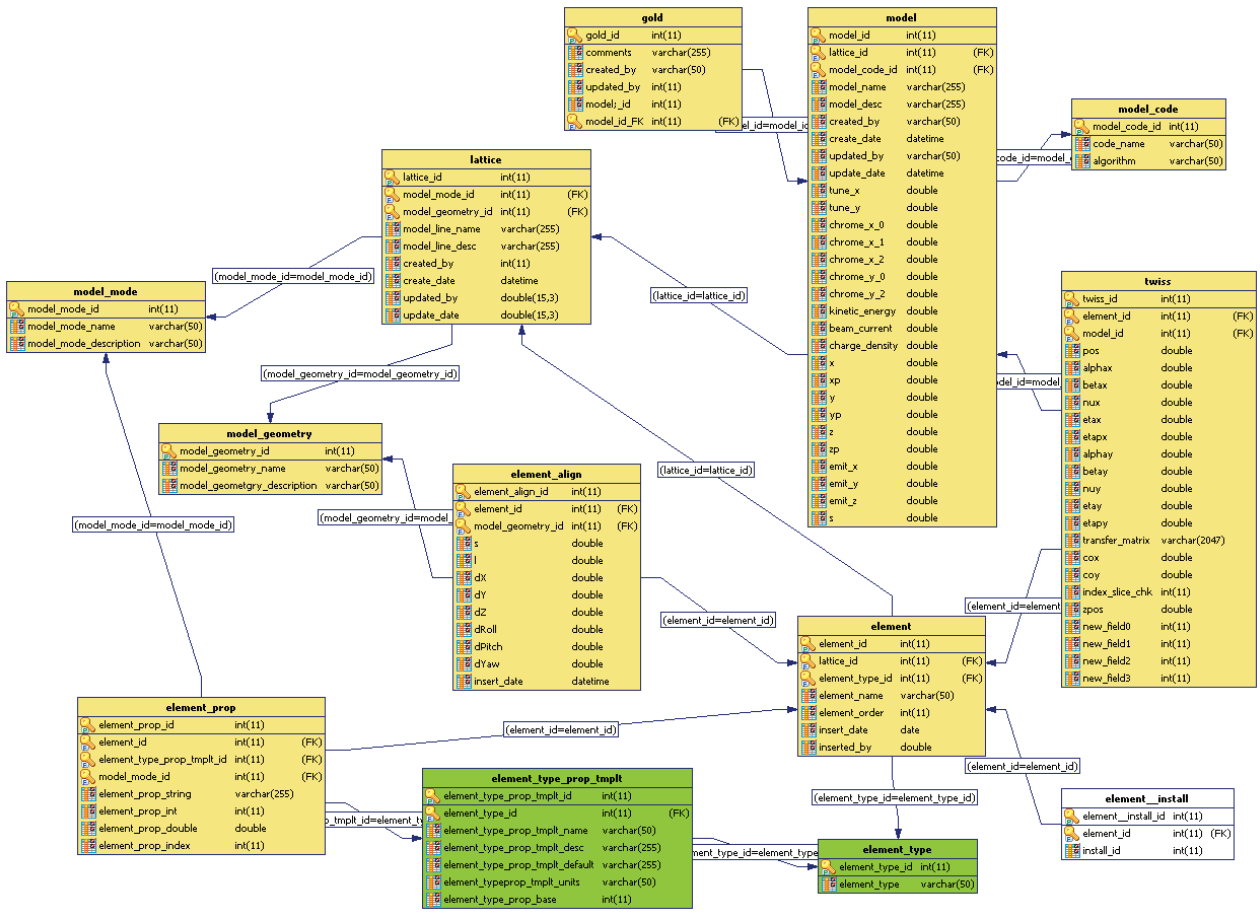

Figure 2: Database schema for storing model related data.

MODEL PLATFORM COMPONENTS

The most important components for the Model Platform are RDB, data access service and client applications.

Database

A model database design is based on a new version of Integrated Relational Model of Installed Systems (IRMIS) and the model database from the Linac Coherent Light Source (LCLS).

The database schema shown in Fig. 2 is designed to accommodate various model data for a wide range of accelerators. There are thirteen tables in this schema:

- Element – basic information such as name, length and location for a model element.
- Element_align – alignment data for the element.
- Element_install – installation information for the element.
- Element_prop – properties for the element.
- Element_type – the element type.
- Element_type_prop_templt – a template to enter properties for a model element type.
- Gold – a table to track default models.
- Lattice – general information for a given lattice.
- Model – general information for a given model.
- Model_geometry – description for a specific modelling geometry.
- Model_line – the beamline for the model calculation.
- Model_mode – any uniquely predefined mode.
- Twiss – model run result data such as beta function, emittance and many other physics parameters.

Model Service

Model Service provides real-time model data as well as model run control such as run submission, status monitoring and aborting model runs. A new EPICS version (EPICS V4) [4] is under development for supporting complicated data structures such as physics model data. Another advantage of choosing the EPICS V4 for model data is that it is the same communication protocol for future control systems. Since the new protocol is still under development, a prototype model data service was tested with Java Persistent API (JPA) and Glassfish Web Service. A preliminary set of data access APIs has been defined. Client applications will call these APIs to get model data seamlessly from RDB, Model Service memory, or model run Meta data from file system.

Class	Variable/Method	Data Type	Arguments	Return	public
Device					
	Device()				Y
	DEVICE_ID	int			N
	DEVICE_NAME	String			N
	TYPE	String			N
	POS	double			N
	...				
	getID()			String	Y
	getType()			String	Y
	getPos()			double	Y
	...				
Element					
	Element()				Y
	ELEMENT_ID	int			N
	ELEMENT_NAME	String			N
	ALPHA_X	double			N
	BETA_X	double			N
	...				
	getAlphaX()			double	Y
	getBetaX()			double	Y
	putAlphaX(ALPHA_X)		double ALPHA_X		Y
	putBetaX(BETA_X)		double BETA_X		Y
	...				
Model					
	...				
	getAllElements()			Element[]	Y

Figure 3: Sample APIs for Model Service.

Fig. 3 shows a small portion of Model Service APIs. So far we have defined only a set of very basic Model Service APIs.

Client Applications

Client applications use the same APIs to access model data produced by different modelling codes. Because the client programming will be model code independent, it is possible to have typical model applications such as orbit correction as modules for different accelerator facilities.

OFFLINE MODELLING SUPPORT

The Model and Lattice subschemas in the Global DB can handle not only to online model data but also offline data. With proper data access services and API, any modelling tool can take advantage of this model data storage. For applications developed to display or manipulate database stored model data, physicists can use them for both online and offline purposes.

One research project is to develop a good multi-particle tracking program and store the data in the database. Furthermore, a study for using Graphics Processing Unit (GPU) as parallel clusters is underway.

ACKNOWLEDGMENT

The authors would like to thank the LCLS database team at SLAC for their original work for the XAL model data.

REFERENCES

[1] J. Galambos, et al, "XAL Application Programming Structure," p. 79, Proceedings of 2005 Particle Accelerator Conference.

[2] P. Chu et al, "Generic Model Host System Design", IPAC'10, Kyoto, Japan, May 2010, TUPEC071, p. 1883 (2010).

[3] C. Benatti et al, "XAL's Online Model at ReA3 to Understand Beam Performance", MOPPC095, these proceedings.

[4] L. Dalesio et al, "EPICS V4 Expands Support to Physics Application, Data Acquisition, and Data Analysis", p. 1338, Proceedings of 2011 International Conference on Accelerator and Large Experimental Physics Control Systems.

ISBN 978-3-95450-115-1