DEVELOPMENT PROGRESS OF NSLS-II ACCELERATOR PHYSICS HIGH LEVEL APPLICATIONS*

Lingyun Yang†, Jinhyuk Choi, Yoshiteru Hidaka, Guobao Shen, Guimei Wang
Photon Sciences Directorate, BNL, Upton, NY 11973, USA

Abstract

The High Level Applications (HLA) for NSLS-II commissioning is a development in progress. It is in a client-server framework and uses Python programming language [1] for scripting and graphical user interface application development. This new development provides both scripting and graphical user interface (GUI) controls. The services developed in controls group provide name server, archiving, machine snapshot, etc. The clients are developed mainly in the physics group and have measurement, analysis and modeling capabilities.

INTRODUCTION

NSLS-II (National Synchrotron Light Source II) is a state-of-the-art third-generation light source under construction at BNL (Brookhaven National Laboratory) [2]. The commissioning of LINAC has started and the storage ring commissioning is expected in about one year.

The development of high level applications (HLA) for commissioning and accelerator physics study is a joint effort from both controls and accelerator physics group. It fits the client-server framework naturally where the controls group provides services and hides the low level details of equipment controls. The higher-level controls which are either scripts or applications with graphical user interfaces can talk to the services with defined application programming interfaces (APIs). For better maintenance and flexibility, the APIs and the dependency between services are reduced on purpose. As a backup, some of the services have local native library as a replacement in HLA, but in this case these replacements are not available for users outside of Python HLA.

SERVICES

The services developed in controls group are meant to support a wider collaboration between accelerator facilities. Most of them have no physics logic inside and can be plug-and-play according to their clients. These services are either based on EPICS V4 or will be ported.

Element Controls and Channel Finder Service

The lowest level of control in HLA is to read/write a single property of magnet or diagnostics equipment. In EPICS terminology it is equivalent to read/write a channel or process variable (PV). For a facility like NSLS-II storage ring, the HLA needs to access tens of thousands of PVs. Using the PVs directly is not a pleasant way to write our own script even with a naming conversion which is well designed and followed. This may become more difficult as the machine evolves. For many high level controls, we need not only the setpoint/readback value of a piece of equipment but also its metadata like the location and type of equipment. Thus, use of one PV or a set of PVs is not sufficient for high level controls.

Table 1: CFS Record Example: Properties

elemName=FYM1G4C02A	element name
devName=FM1G4C02A	device name
elemType=VFCOR	element type
cell, girder=C02, G4	cell name
symmetry=A	symmetry
elemField=y	element field
handle=READBACK	read/write
sEnd,length=65.5222,0.044	physics location, length
ordinal=264	index in lattice file

HLA uses a directory service called channel finder service (CFS) [3] to manage these auxiliary pieces of information. The channel (PV) name is the key of a record. Each record can have properties and tags. Properties are keyword-value pair data, whereas tags are plain strings. As an example, Table 1 and 2 show the properties and tags, respectively, of a record whose key is the channel name `SR:C02-MG:G04BHCor:M1Fld-I`. This PV corresponds to the readback value of a horizontal corrector in girder 4 and cell 2. Note that certain naming conventions should be established to avoid tag name collision if multiple users are allowed to tag each record.

Different from the properties' keyword-value pair data format, tags are only a string. Table 2 shows the tags of the same record as Table 1. Certain conventions can be applied for different users.

With CFS, HLA can construct one or more accelerator lattices. Each lattice has enough information to access setpoints and read-backs, and to find neighboring elements. Python language has good standard libraries for all these operations.

Machine Snapshot

The Machine Snapshot and Retrieve service (MASAR) is another example of services developed in our controls group. The architecture of MASAR is shown in Fig. 1. It adopts EPICS-V4 as a middle service layer and uses chan-

* Work supported by Brookhaven Science Associates, LLC under Contract No. DE-AC02-98CH10886 with the Department of Energy.
† lyyang@bnl.gov

Table 2: CFS Record Example: Tags

tag	description
aphla.eget	a default read PV for this element
aphla.eput	a default write PV for this element
aphla.x	horizontal (kicker in this case)
aphla.y	vertical (kicker in this case)
aphla.sys.SR	storage ring system
aphla.sys.LTB	linac to booster system
aphla.sys.LTD1	linac to dump 1
aphla.sys.LTD2	linac to dump 2

Figure 1: Architecture of MASAR.

nel RPC as the communication protocol. MASAR saves a predefined set of channels and compares the current machine data with previously saved data. A configuration defines the set of channels to be saved. The channels can be of any available record data types such as scalar values, arrays and strings. The configurations have a relational database back-end (e.g. SQLite3) and could be connected to the global IRMIS system. A prototype GUI application for this service has been implemented using PyQt4 with more features to be added in the future. The Python APIs for MASAR service provide thin interface but meet all HLA requirements.

Online Model and Unit Conversion

We have been testing our HLA applications against a Tracy-II based virtual accelerator. When the real machine is online, this virtual accelerator can still serve as an online calculator. It will be useful to construct a lattice simulation based on current machine readings and the HLA provides the capability to do this online or semi-online.

Both the online and offline calculation need to convert the magnet strength from machine units to physics units. This will be available to not only the Python HLA but potentially to other customers. A unit conversion service whose architecture is shown in Fig. 2 would access the magnet measurement data in the central database and provide the conversion between different units. This service has been planned and a Python API set will be developed.

We have also finished the design of TWISS service for the online calculation of beam/machine properties.

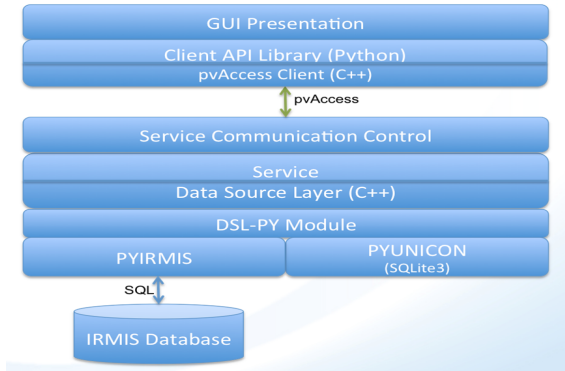

Figure 2: Architecture of UNICON.

CLIENTS

The client side consists of python scripts, notebooks and GUI applications. The core HLA library is designed to minimize the memorization of lattice structure by including powerful search capability. All operations are in search-verify-operate mode. Searching will find the elements matching certain patterns. A pattern could be a magnet type, a cell number, or girder number or Unix filename wildcard pattern. A code example is shown in Fig. 3. Verification is necessary if the operations to be followed are critical. For example if an algorithm requires exactly 4 BPMs, then a verification on the length of a returned element list is needed. This is easy in Python HLA. In HLA, a typical operation is performed on the ”field” of elements. The fields have familiar names like ”k1” for the strength of a quadrupole and ”f” for the frequency of an RF cavity. The values of these fields are in physics units, and thus can be directly used in lattice modeling.

```
#!/usr/bin/env python
import hla
# create lattice from channel finder
hla.initNSLS2VSR()
hla.machines.use('SR') # 'LTB'/'LTD1'/'LTD2'
# searching based on element type, cell, girder, symmetry
bpm = hla.getElements('BPM')
for b in bpm: print b.value
for b in bpm:
    print b.name, b.cell, b.girder, b.sb, b.x, b.y

trim = hla.getElements('HCOR')
for t in trim: t.value = 0.0
for t in trim: print t.name, t.value
```

Figure 3: Code example of HLA.

Launcher

The launcher is similar to a file browser like Windows Explorer, the launcher organizes the available scripts and applications for HLA in a tree structure (see Fig. 4). Searching and smart matching is possible with PyQt4 support.

A standard list of available applications comes with the HLA Python installation package. Although this standard

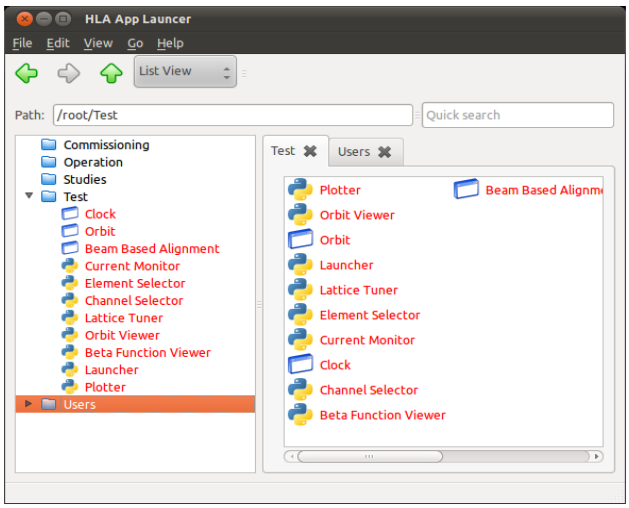

Figure 4: Launcher of HLA.

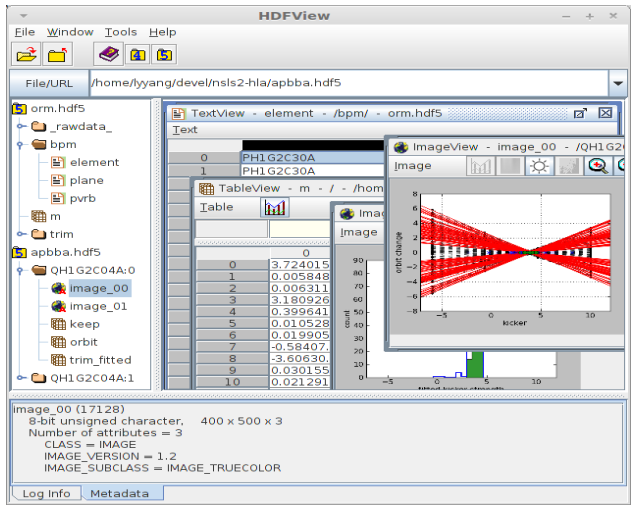

Figure 6: Beam based alignment.

list is read-only for end users, users can add a custom list of applications to the launcher for a quick access to frequently used applications. This custom list is stored in their `$HOME` directory as part of the HLA configuration file, like standard Linux applications.

Orbit and Beam Based Alignment

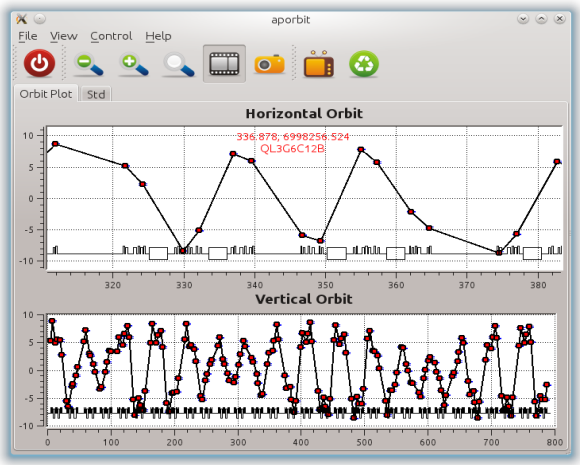

Figure 5: Orbit display and control.

The Orbit display shown in Fig. 5 integrates orbit control and magnet control functions. By zooming in/out and clicking the proper hot-zone a tuning control will be available to read and write the element properties. These properties are defined in CFS with familiar names as in the lattice design.

The beam based alignment (BBA) application follows the ALS algorithm [4]. The analyzed data including the pictures are stored as the HDF5 format. The file format is accessible to many programing languages and applications, e.g. Mathematica, Matlab, C/C++ and FORTRAN (see Fig. 6).

General Purpose Plotting

We are developing a general-purpose plotting application for data acquisition and analysis. Data from different sources such as the archiver data base and the actual machine can be visualized as a time-series plot, as well as against each other in a 2-D plot. A Python shell environment will be also embedded in the application. This allows not only raw data but also data derived from raw data to be plotted.

PyTracy and PyLOCO

Python binding of Tracy-II code has been developed for online and offline simulations. Following the Matlab version of LOCO [5], we are also developing a Python version for linear optics modeling.

ACKNOWLEDGMENT

We thank Kunal Shroff, Gabriele Carcassi, Ralph Lange for CFS, Don Dohan for IRMIS. The whole open source community for providing great tools.

REFERENCES

[1] Python Programming Language, http://www.python.org/

[2] NSLS-II preliminary design report, http://www.bnl.gov/nsls2/project/PDR/

[3] Directory Service for EPICS Channels, http://channelfinder.sourceforge.net/

[4] G. Portmann, D. Robin, and L. Schachinger, "Automated beam based alignment of the ALS quadrupoles," in PAC95.

[5] J. Safranek, "Experimental Determination of Storage Ring Optics Using Orbit Response Measurements," Nuclear Instruments and Methods A 388 (1997).

ISBN 978-3-95450-115-1