Physics High-level Applications and Toolkit for Accelerator System
An overview of FRIB high-level physics applications development

Tong Zhang
Controls Physicist
E-mail: zhangt@frib.msu.edu
Presented at EPICS Collaboration Meeting, 11-15 June 2018, APS, ANL
Acknowledgements

Dylan Maxwell, Martin Konrad, Daron Chabot, Eric Berryman, Steven Beher, Diego Omitto, Guobao Shen, Michael Davidsaver, Steven Lidia

Kei Fukushima, Harry He, Tomofumi Maruta, Takashi Yoshimoto, Jonathan Wong, Alexander Plastun, Qiang Zhao, Yue Hao, Steve Lund, Peter Ostroumov, Masanori Ikegami
# Outline

1. **Designed Architecture**
   - Introduction
   - Device Abstraction

2. **Key Features**
   - Virtual Accelerator
   - Online Model

3. **Conclusions**
OUTLINE

1. **Designed Architecture**
   - Introduction
   - Device Abstraction

2. **Key Features**

3. **Conclusions**
**Accelerator System**

- particle source, beam transport, end stations, ...
- devices: optics, diagnostics, ...
- distributed controls units: EPICS input & output controllers (IOCs)

**Purpose:** have robust and functional beam tuning algorithms

**Final goal:** operating accelerator facility

**Solution:** software environment for high-level physics controls

**High-level Physics Applications = Physics Algorithms + Controls Software**
**Introductions to High-level Physics Applications**

**Accelerator System**
- particle source, beam transport, end stations, ...
- devices: optics, diagnostics, ...
- distributed controls units: EPICS input & output controllers (IOCs)

**Final goal:** operating accelerator facility

**Purpose:** have robust and functional beam tuning algorithms

**Solution:** sof_tware environment for high-level physics controls

**High-level Physics Applications = Physics Algorithms + Controls Sof_tware**
Accelerator System

- particle source, beam transport, end stations, ...
- devices: optics, diagnostics, ...
- distributed controls units: EPICS input & output controllers (IOCs)

Introduction to High-level Physics Applications

- Final goal: operating accelerator facility
- Purpose: have robust and functional beam tuning algorithms
- Solution: software environment for high-level physics controls

High-level Physics Applications = Physics Algorithms + Controls Software
Accelerator System

- particle source, beam transport, end stations, ...
- devices: optics, diagnostics, ...
- distributed controls units: EPICS input & output controllers (IOCs)

High-level Physics Applications

- Final goal: operating accelerator facility
- Purpose: have robust and functional beam tuning algorithms
- Solution: software environment for high-level physics controls
## Accelerator System

- particle source, beam transport, end stations, ...
- devices: optics, diagnostics, ...
- distributed controls units: EPICS input & output controllers (IOCs)

## High-level Physics Applications

- **Final goal:** operating accelerator facility
- **Purpose:** have robust and functional beam tuning algorithms
- **Solution:** software environment for high-level physics controls
### Accelerator System
- particle source, beam transport, end stations, ...
- devices: optics, diagnostics, ...
- distributed controls units: EPICS input & output controllers (IOCs)

### High-level Physics Applications
- **Final goal**: operating accelerator facility
- **Purpose**: have robust and functional beam tuning algorithms
- **Solution**: software environment for high-level physics controls
**Introductions to High-level Physics Applications**

**Accelerator System**
- particle source, beam transport, end stations, ...
- devices: optics, diagnostics, ...
- distributed controls units: EPICS input & output controllers (IOCs)

**High-level Physics Applications**
- **Final goal**: operating accelerator facility
- **Purpose**: have robust and functional beam tuning algorithms
- **Solution**: software environment for high-level physics controls

High-level Physics Applications = Physics Algorithms + Controls Software
**Software Solution upon Python Programming Language**

- **High-level Physics Applications**
- **Other applications (non-Python)**
- **Python applications**
- **Web applications**

**Fundamental Requirements**

- Quick prototyping: dynamic programming language
- Functional: plenty of third-party packages
- Agile development: develop → build → test → deploy
PHANTASY
Physics High-level Applications and Toolkit for Accelerator System

**Features Highlight**
- Device configuration management
- Device abstraction
- Online modeling
- Python interactive scripting environment for high-level controls
- Virtual accelerator based on EPICS control environment
- Web service integration (channelfinder, UNICORN, scanserver)

**Deployment**
- Target OS: Debian 8 (Jessie)
- Main packages: *python-phantasy*, *phantasy-machines*
- Physics model engines: *python-flame*, *python-impact*
**Physics Applications Architecture**

**TOOLKIT**

CLI commands, data management, convenient scripts, ...

**APPLICATIONS**

Virtual accelerators, orbit correction, parameters scan/optimization, ...

---

**PHANTASY**

- **Toolkit**
  - Devices
    - IOC-1
    - IOC-2
    - ...
    - IOC-n
  - Meta Devices (SoftIOCs)
    - IOC-1
    - IOC-2
    - ...
    - IOC-n

- **Applications**

- **Web Services**
  - ChannelFinder
  - Scan Server
  - Olog
  - UNICORN
  - data

- **APIs**
from phantasy import MachinePortal

mp1 = MachinePortal(machine='FRIB', segment='LEBT')
mp2 = MachinePortal(machine='FRIB', segment='MEBT')
lattice1 = mp1.use_lattice('lattice-1')
lattice2 = mp1.use_lattice('lattice-2')

elem1 = mp1.get_elements(type='SOL')[0]
elem2 = mp1.get_elements(name='*D0709*', type='HCOE')[0]

Field1
Field2
FieldN

Element-1
Element-2
Element-N

Lattice-1
Lattice-2
...
Lattice-N

fld1 = elem1.get_field('B')
fld2 = elem1.get_field('I')

Element

get_config(B)  elem.B
set_config(B)  elem.B = <new-value>
get_config(I)  elem.I
set_config(I)  elem.I = <new-value>

gate config  fld.get('readback')
sat config  fld.set(<new-value>, 'setpoint')

Scaling laws linked between I & B

element-wise

Field-wise

f1 = elem.get_field('B')

Static Properties
- name
- length
- sb
- se
- family
- index
- tags

depends on specific element

Dynamic Properties
- <fields>
Information abstraction and aggregation:

Quad:
- `readback`: ['PSQ1:V RD', 'PSQ2:V RD']
- `setpoint`: ['PSQ1:V_CSET', 'PSQ2:V_CSET']

# PS1: +V1, PS2: -V2 ==> value: -(V1+V2)/2
read_policy = lambda x: 0.5 * (-x[0].value+x[1].value)
write_policy = lambda x,v,**kws: [x[0].put(-v,**kws), x[1].put(v,**kws)]

read : quad.V
write: quad.V = 1000

How to return value as device settings
read_policy

[RBDK_PV-P51, RDBK_PV-P52]
readback PVs

How to set device with new settings
write_policy

CSET_PV-P51, CSET_PV-P52
setpoint PVs
Information abstraction and aggregation:
1. Designed Architecture

2. Key Features
   - Virtual Accelerator
   - Online Model

3. Conclusions
Create EPICS controls environment for development, physics behavior simulated by model engine (FLAME, IMPACT).
Create EPICS controls environment for development, physics behavior simulated by model engine (FLAME, IMPACT).
CREATE A FULL-FEATURED HIGH-LEVEL ABSTRACTED SOFTWARE ENVIRONMENT, 
ACCELERATOR PHYSICISTS FOCUS ON SOLVING PHYSICS PROBLEMS.
REST APIs to evoke scaling laws: Python-client or web page

Represent devices with an informative way

Manage scaling rules in a friendly way

Debian package:

- Web application: unicorn-webapp
- Python interface: python-unicorn, python3-unicorn
### Functions

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Invoked</th>
<th>Definition</th>
<th>Last Updated</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FE_MEBT-Q_D1057-P</strong></td>
<td>I to G</td>
<td>3</td>
<td><code>def f(x, **kws): x1 = kws.get('x1', 89.6)</code></td>
<td>2018-06-05 09:26:21 EDT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>a1 = kws.get('a1', 0.0)</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>b1 = kws.get('b1', 0.15)</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>c1 = kws.get('c1', 2.24e-5)</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>a2 = kws.get('a2', -5.12)</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>b2 = kws.get('b2', 0.264)</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>c2 = kws.get('c2', -6.14e-4)</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>if 0 &lt;= x &lt; x1:</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>return a1 + b1 * x + c1 * x * x</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>elif x &lt;= 200:</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>return a2 + b2 * x + c2 * x * x</code></td>
<td></td>
</tr>
</tbody>
</table>

| **FE_MEBT-Q_D1057-N** | G to I      | 1       | `def f(x, **kws): x1 = kws.get('x1', ...)`                                | 2018-06-05 09:27:48 EDT |

Showing 1 to 2 of 2 entries (filtered from 64 total entries)
UNICORN: Unit Conversion Web Application

Inspection of Function: FE_MEBT:Q_D1057-P

ARGS
x, x1, a1, b1, c1, a2, b2, c2

AUTHOR
uadmin

DESCRIPTION
I to G

INVOKED
3

LASTIN
(x: 20.0)

LASTOUT
3.00896

NAME
FE_MEBT:Q_D1057-P

TIMESTAMP
2018-06-05 09:28:21 EDT

UDEF

```python
def f(x, **kws):
x1 = kws.get('x1', 89.6)
a1 = kws.get('a1', 0.0)
b1 = kws.get('b1', 0.15)
c1 = kws.get('c1', 2.24e-5)
a2 = kws.get('a2', -5.12)
b2 = kws.get('b2', 0.264)
c2 = kws.get('c2', -6.14e-4)
if 0 <= x < x1:
    return a1 + b1 * x + c1 * x * x
elif x <= 200:
    return a2 + b2 * x + c2 * x * x
```
UNICORN: Unit Conversion Web application

```python
In [1]: import unicorn

In [2]: # admin client to UNICORN service
admin_client = unicorn AdminClient('http://localhost/unicorn')

In [ ]: # get all functions
admin_client.get()

In [4]: # get function by name
admin_client.get(name='FE_SCS1:SOLR_D0704-P')

Out[4]: {u'function': {u'args': u'x,k',
                  u'author': u'uadmin',
                  u'code': u"def f(x, **kws):
                    k = kws.get('k', -2.89e-3)\n                    return k * x
                  ",
                  u'description': u'1 to B',
                  u'invoked': 3,
                  u'last': u'{"x": 10.0}',
                  u'lastout': u'-0.02890000000000002',
                  u'name': u'FE_SCS1:SOLR_D0704-P',
                  u'timestamp': u'2018-05-22 11:00:12 EDT',
                  u'def': u"def f(x, **kws):
                    k = kws.get('k', -2.89e-3)\n                    return k * x",
                  u'uri': u'http://localhost:5000/functions/FE_SCS1:SOLR_D0704-P',
                  u'uri-api': u'http://localhost:5000/api/v1.0/functions/FE_SCS1:SOLR_D0704-P'}

In [5]: # api client, the base url + '/api/v1.0'
api_client = unicorn.ApiClient('http://localhost/unicorn/api/v1.0')

In [6]: # get function execution result
api_client.get('FE_MEBT:Q_D1057-P', x=20)

Out[6]: {u'result': 3.00896}

In [7]: # get result from reverse function
api_client.get('FE_MEBT:Q_D1057-N', x=3.00896)
```
deployment (I)

- FRIB controls network
  git → stash → jenkins → puppet → target workstations

- Local development
  VirtualBox Appliance

- Cloud development
  Docker container based web computing platform (configurable-proxy, docker images)
Deployment (II)

Welcome to the Computing Platform for High-level Physics Controls of FRIB Accelerator.

**Guideline**

- Login via [here](#), for new user, sign up first.
- Click the user name to access user configuration management.
  - Update basic user information via [profile](#).
  - Update service via [service](#).
- Click Service Name to control service.
- Click Notebook URL to get access into notebook.

**Container Configuration**

- **Username**: user1
- **Container Name**: peaceful_lichterman
- **Container ID**: 5ca84c2006
- **Notebook URL**: /user1

**Service and Section**

- **Service**: physspo 1.6 ss
- **Section**: [LEBT](#), [LEBT](#), [MERT](#), [LS1](#)

**Status**

- **Status**: RUNNING
- **Owner**: user1
- **Image**: tonyyang/physspo release-1.6-ss
- **ID**: 5ca84c2006446671e6b13b8deeeb00d2f82253f9d056cd0d080bee072a1d51d
- **Uptime**: 0:01:12:37:36:17
- **Ports**: [31000, 30000]
### Deployment (II)

```bash
TOKEN = 6520fbd2223339e729c99b4f1730f1dd2098b57c3f3d692a37ba6fecd553
ETHO := enx18dbf2615ea9
IPNOW := $(shell ifconfig $(ETHO) | \
    /bin/grep "inet addr" | \
    awk -F: '{print $2}' | \
    awk '{print $1}')
IMAGE_MNB := "tonyzhang/phyapps-notebook:latest"
IMAGE_PROXY := "jupyter/configurable-http-proxy"
DPATH := $(shell pwd)

deploy: proxy mnb
stop: stop-proxy stop-mnb
stop-proxy:
    @docker container stop proxy
    @docker container rm proxy
stop-mnb:
    @docker container stop mnb
    @docker container rm mnb

proxy:
    @docker run -d \n        -e CONFIGPROXY_AUTH_TOKEN=$(TOKEN) \n        --name=proxy \n        --net=host \n        -v $(shell pwd)/ssl:/ssl \n        $(IMAGE_PROXY) \n        --log-level debug \n        --ssl-key /ssl/key.pem \n        --ssl-cert /ssl/cert.pem \n        -ip $(IPNOW) \n        -port 8000 \n        --default-target http://127.0.0.1:5050
mnb:
    @docker run -t -d \n        -e PROXY_TOKEN=$(TOKEN) \n        -e PROXY_BASE="http://127.0.0.1:8001/api/routes" \n        -e DPATH=$(DPATH) \n        -p 5050:5050 \n        --name=mnb \n        --net=host \n        -v /var/run/docker.sock:/var/run/docker.sock \n        $(IMAGE_MNB)
```

*make deploy
  make stop*
1 Designed Architecture

2 Key Features

3 Conclusions
Conclusions

- Established Python-based software infrastructure for high-level physics controls
- The solution for systematic high-level device abstraction
- Dedicated web application and Python interface for units interpretation
- Continuous integration and delivery at FRIB

Future Plans

- Operation: develop mature physics algorithms into soft-IOCs
- Python ecosystem: data management
Thank you for your attention!