

PyBlulce

Modular MX Data Collection Software
for Fully Automated, High Data Rate and Long Duration Experiments

MCE 2021 Workshop @ NSLS-II
March 18, 2021
Mark Hilgart



U.S. DEPARTMENT OF
ENERGY

Argonne National Laboratory is a
U.S. Department of Energy laboratory
managed by UChicago Argonne, LLC.



MX Data Acquisition at GM/CA

Two ID MX Beamlines

- Sector 23 at the Advanced Photon Source, Argonne National Laboratory

JBlulce is written for and by GM/CA

- Tools are optimized for GM/CA capabilities

Major functions

- Raster, vector and helical collect
- Fluorescence spectrum and edge scan
- Screening
- Analysis

JBlulce-EPICS: Beamline ID-D Version mark-v2015.1.1 Build 4925 FPE Mar-02

File Network Tools Help

Hutch Sample Screening Raster Scan Collect Analysis Users Log

Control: Resume Pause Shutterless: On

Options: Save pos. SpotFinder Settings are default Change

Interactive Auto

Run 3 (rastering)

Reproc Update Delete Reset

Raster mode: ☒ Diffraction ☐ Fluorescence

Prefix: A11_raster

Dir: .../NFS/A11/raster6

Polygons: One + - C

Cell size: 5 x 5 μm

Beam size: 5 x 5 μm

Time: 0.050 sec

Delta: 0.000 deg

Distance: 500.000 mm

Attenuation: 20.00 factor

Res. Limit: L 50.0 H 2.6 \AA ☐ Fix

Macros: L X +

Numbering: None Cell Heatmap Coloring: None Spot Bragg Resolution

Move View Click or drag to view cell images

SpotFinder Results SpotFinder Output

#	Spot Total	Bragg Cand.	Res. Est.
1758			
1759			
1760	4	0	10.00
1761	4	0	10.00
1762			
1763			
1764			
1765			
1766			

[12:52:49] NOTE: Current FPE frame is 1778

APS Current 102.1 Shutter Permit Enabled A Shutter Open Endstation Shutter Open Endstation Secure Yes

State: Rastering ETA: 2.1 min EMERGENCY STOP Mono: 11.999 keV IZero: 1.01 V Control: Active Shutter: Open



U.S. DEPARTMENT OF
ENERGY

Argonne National Laboratory is a
U.S. Department of Energy laboratory
managed by UChicago Argonne, LLC.



75
1946-2021

PyBlulce Project

PyBlulce started as a project to bring new features to JBlulce
The benefits we found led us to decide on a full rewrite to Python

The new features we need are

Long Duration Experiments

- Million cell raster
- Multi-hour injector collect

High Data Rates

- 100Hz sustained

Full Automation

- Optical center
- Raster-based center
- Multi-site strategy
- Strategy-based collect

More data collection tools

- Several collect modes
- Multiple tools per mode



PyBlulce Project

PyBlulce features driving changes

Long Duration
Experiments

High Data Rates

Full Automation

Independent Collect
Plug-ins

Improvements needed to deliver these features

Performance

- Responsive GUI while viewing collection plans and results for million frame collections
- Viewing, processing and analysis at hours-long sustained high data rates

Modularity

- Independent collect plugins prevent too much complexity in shared components
- Complex operations need a simple interface for automation

Reliability

- Automation needs to run all day long and recover from errors



PyBlulce Project

PyBlulce features driving changes

Long Duration
Experiments

High Data Rates

Full Automation

Independent Collect
Plug-ins

Technical Changes Needed

Java & SWT

EPICS & SQL

Single process

Hardware abstraction
simulation

Python3 & Qt5

Redis

Thin GUI with HTTP RPC

Realistic beamline
simulation

Concise and efficient

Fast, atomic, random
access data passing

Non-GUI load
completely offloaded

Interactive and automatic
short & long-duration tests



Why focus on modularity?

Allow more
overall
complexity

Remove specifics
from shared
components

Clearly define API
between components
to expose shared data

Speed
development

Reduced
dependencies makes
changes easier

Enable a larger
development
team

Scripts used by the
server are short and
easy to modify for
new tasks

The same API used by
server components is
clearly documented
and accessible

Tolerate offline
hardware

Individual testing of
modules eliminates
unnecessary
dependencies



Achieving modularity

Independent scripts

- Complex operations are moved to independent scripts with no server imports, only HTTP/Redis API usage
- Separates high-level, unique tasks from core, shared components
- Provides templates for developers to start with

Unit tests

- Enforce modularity by only starting the modules that are needed

Redis

- Passing data via organized, documented Redis DB locations makes debugging easier
- Easier to diagnose and fix broken database links instead of more complex code-based links

Collect plug-ins

- Moving collect mode-specific code to plug-ins contains their complexity



Reliability for more complex tasks

Lights-out automation

Detect errors that occur only after repeated or certain patterns of calls

Recover from hardware errors

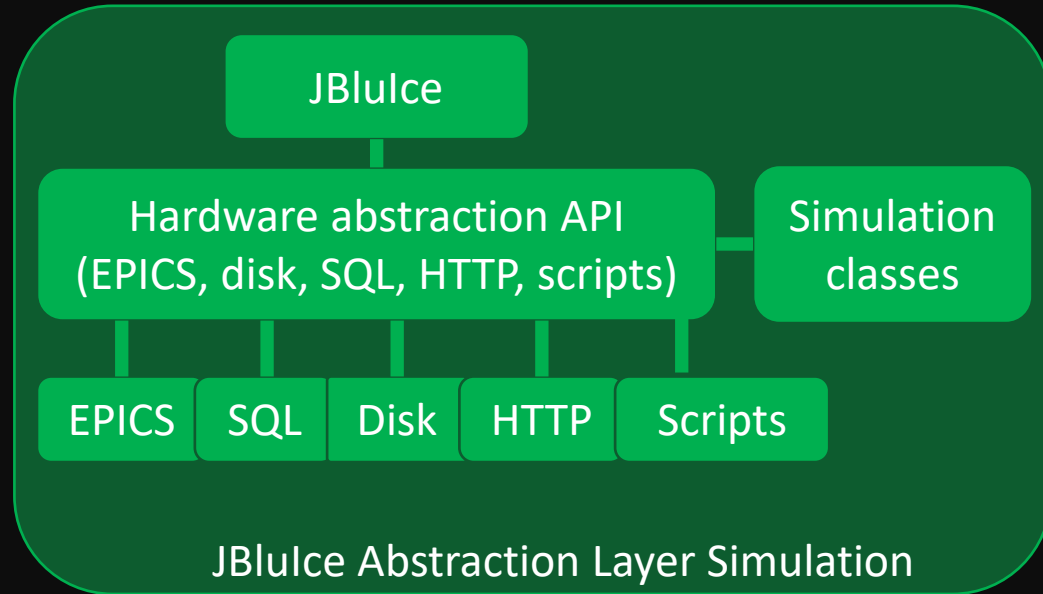
Know when to stop and prompt the user

Reliable scripting interface

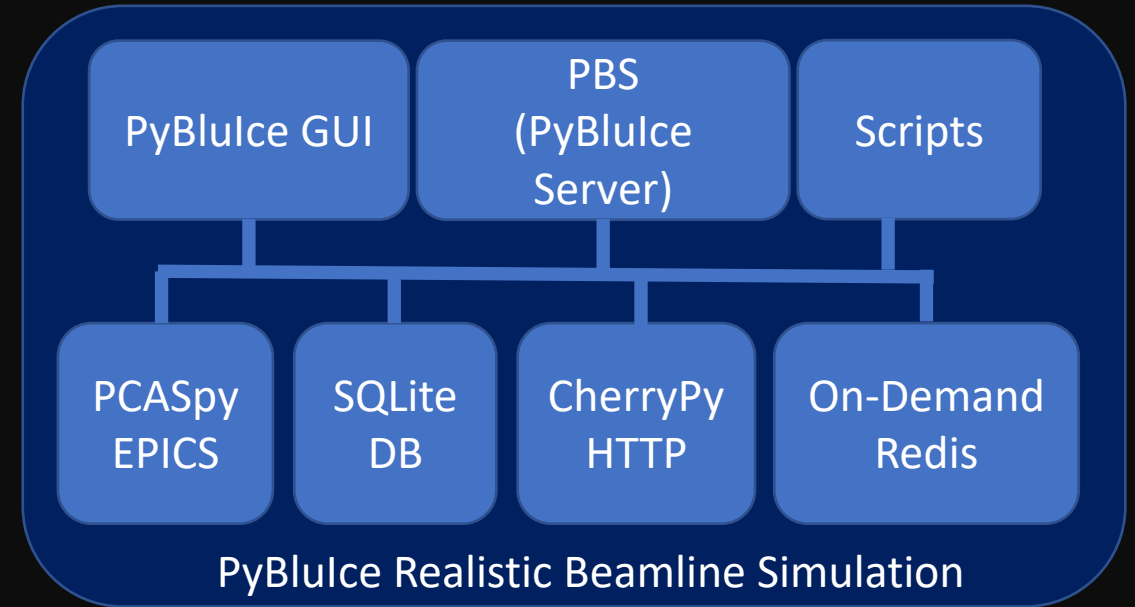
All API options should be tested even if not currently in use



Reliability for more complex tasks: Realistic Simulation



- For JBlulce we wrote an API for each type of hardware device
- This required implementing some part of each protocol e.g. SQL parsing
- High maintenance costs
- Did not achieve offline development

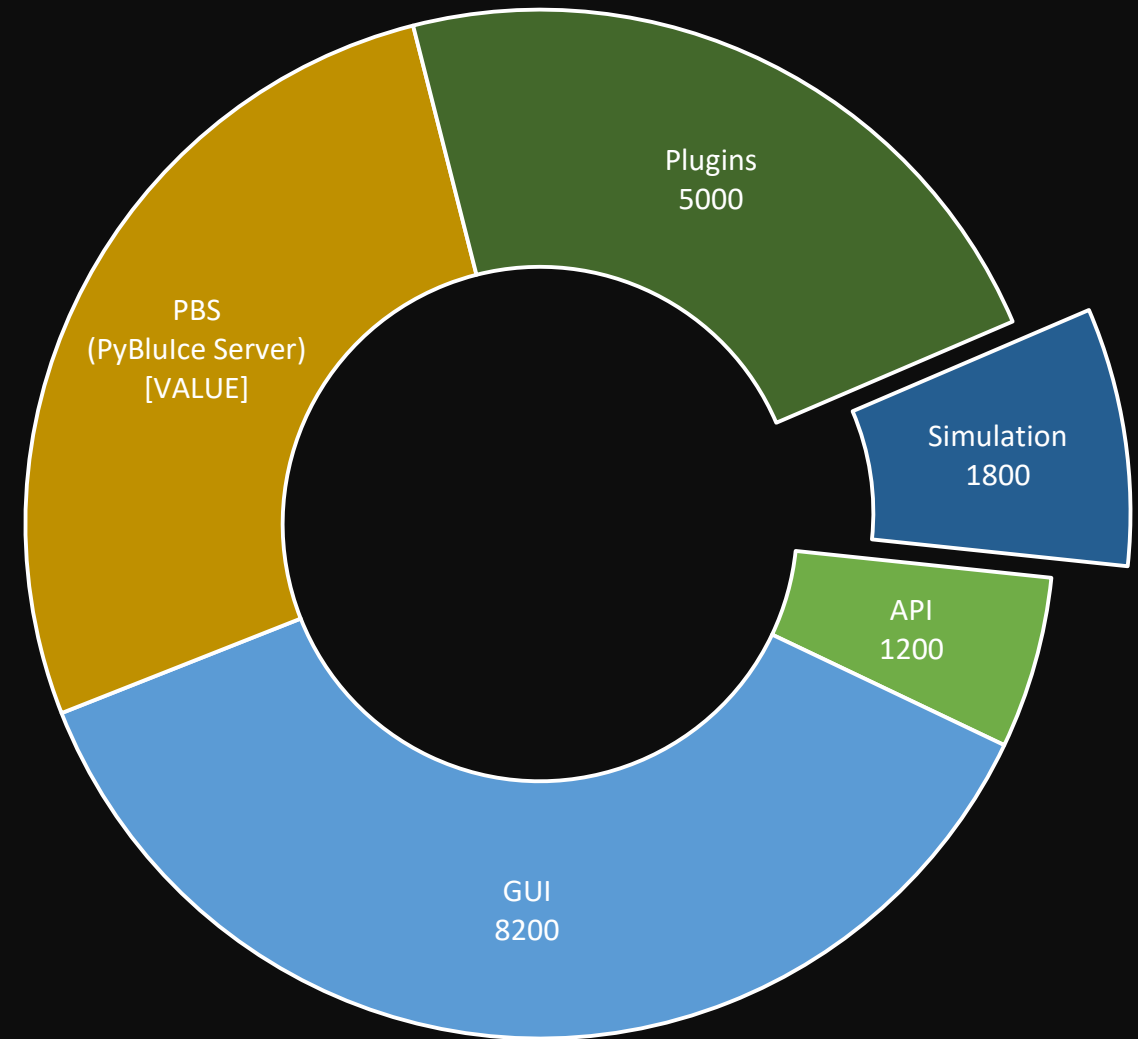


- Python libraries are available to simulate EPICS, SQL and HTTP
- Clients see real EPICS, Redis and HTTP servers
- Enables automated testing and offline development and debugging
- Nearly all operations are currently supported

Lines of Code

Simulation to support all major operations adds 7% to code base

Total size is one fifth of JBlulce's 124,000 LOC



Reaching performance goals

Efficient API for offloading tasks

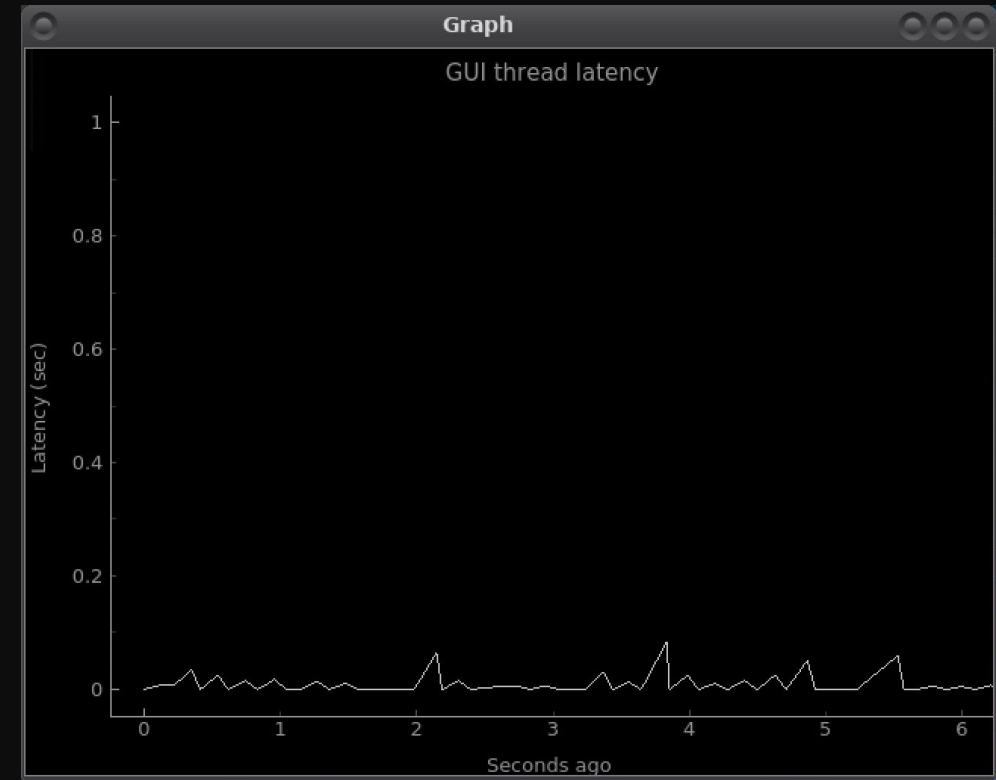
- When offloading is as easy as not, this makes keeping the GUI lean possible
- Slurm tasks also offload processing and drop their results into the database without needing to connect back to the server

High performance PyQt widgets

- pyqtgraph makes video and image viewing at high speed simple
- Lazy tables are more straightforward than in SWT

Modularity and Simulation contribute

- Concise, self-contained methods and modules make refactoring and optimization easier
- Simulation makes regular latency and resource monitoring possible offline



Built-in GUI thread latency viewer

Foundation: Redis API

```
import api.redis_api as ra
ra.run_b.delta_deg.set(0.2, id=1)
```

Setting a data collection parameter using the PyBlulce Redis API import

PyCharm integration

- Tab completion for all levels of field hierarchy
- Control+Q on fields shows their description and formatting

Synchronous, atomic, fast

- Grouping reads and writes into “pipelines” ensures atomicity
- Couple milliseconds per round trip allows for multiple calls within a GUI thread callback
- Can be combined with HTTP calls and the HTTP receiver is guaranteed to see the new values with no delay

Flexible, open

- Above can be done by any script, including external to PyBlulce
- This exposes the inner workings of the servers and makes them accessible for anyone to add on to
- Redis provides linked lists, random-access binary arrays, and other data types



Foundation: HTTP API

```
import api.redis_api as ra
import api.http_api as ha

ra.run_b.delta_deg.set(0.2, id=1)
ha.collect_runs_py(runs=1).post()
```

Starting data collection with a Redis setting followed by an HTTP call

Used for RPC

- Provides reliable parameter passing and immediate or delayed response
- ZeroMQ has protocols that are well-suited to RPC, but HTTP is sufficient, more widely known and has more tools available

Language agnostic

- Any language can either use a library or call curl
- Future web-based interfaces can directly connect to the PyBlulce servers



Auto Tab Preview

Event history

- Shows the user everything automation is doing
- Manual operations and crystal info are shown too
- Previews results: raster grids, camera snapshots, analysis summaries
- Links to full results in other tabs or external browsers
- Sorted by sample ID to include unmounted and manual mount sample events

Global path config

- In automatic mode, operations need a way to determine an output path

Automation queue

- In screening mode, the queue is fixed
- Future modes may allow arbitrary user editing of the queue

Ergonomics

- Compact mode for laptops and large text mode for certain large monitors
- Optional dark mode can reduce eye strain
- Resizable and pop-out widgets make zooming in easy

PyBluice-EPICS: Beamline SIM version v2021.1

File View Network Tools Help

Auto Sample Collect Analysis Scan SONICC Users Hutch

Samples

Collapse Expand 1 Expand All View HTML Location view Show all Import

Sample/Event	Details
Center	[7:46pm] OK
Camera	[7:46pm] OK
Strategy	[7:46pm] OK Collected prot_efg10_1
- E11: prot_efq11	Strategy taken
Mount	[7:46pm] OK
Center	[7:47pm] OK
Camera	[7:47pm] OK
Strategy	[7:47pm] OK Collected prot_efg11_1
- E12: prot_efq12	Strategy taken
Mount	[7:47pm] OK
Center	[7:47pm] OK
Camera	[7:47pm] OK
Strategy	[7:47pm] OK Collected prot_efg12_1
- E13: prot_efq13	Strategy taken
Mount	[7:47pm] OK
Center	[7:47pm] OK
Camera	[7:47pm] OK
Strategy	[7:47pm] OK Collected prot_efg13_1
- E14: prot_efq14	Strategy taken
Mount	[7:48pm] OK
Center	[7:48pm] OK
Camera	[7:48pm] OK
Strategy	[7:48pm] OK Collected prot_efg14_1
- E15: prot_efq15	Centered
Mount	[7:48pm] OK
Center	[7:48pm] OK

Paths

Directory: ~/23SIM_2021_03_15/prot_efg15/[task]
File name: prot_efg15_#####.cbf
Parent dir: Sample prefix: prot_efg15
Sample dir: prot_efg15 ✓ Task dir Browse Terminal Reset

Automation

Mode
• Screen

Config
Exposure (sec): 0.20
Delta (degr): 0.20
Det dist. (mm): 300
Atten (fact): 100
Beam size (wxh um): 20
Update

Commands
[] ✓ Mount
[] ✓ Center mode Loop
[] ✓ Pause
next [] ✓ Camera
[] ✓ Strategy images 2 φ start 0
[] Collect φ range 0
[] ✓ Pause
Start Pause

Dewar

View: Mount order Progress

Fill	Reset	Undo
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16		
A B C D E F G H I J K L M N O P Q R		

Robot

Operation
Log SMR: Ready
sample E15 loaded
SMR: Ready
Command load E A 15
State Ready
Warmup 0

Mount
Location E15
Sample prot_efg15
Pin sensor Empty

Commands
Ready
Pause
Warmup
Unmount
Mount A1
Rinse 3x
Wash 3x

Operation
Automation Inactive
Collect Inactive

Positions
Beam size (um) nanxnan
Atten. (fact) nan
Omega (degr) 0.000
Det. dist. (mm) 300.0

Details
Disk usage 0
Log Automation log

APS Current 7.7 mA Shutter Permit Disabled A Shutter Closed ES shutter Open ES secure No
Server log GUI log ETA --- Mono 12.660 keV EMERGENCY STOP IZero 2.15 V Control Passive Fast shutter Closed



U.S. DEPARTMENT OF
ENERGY

Argonne National Laboratory is a
U.S. Department of Energy laboratory
managed by UChicago Argonne, LLC.

MCE 2021 Workshop @ NSLS-II

Mark Hilgart



75
1946-2021

Collect Tab Preview

Raster plug-in is shown

- Collect mode is selected at the top of each run
- Each plug-in:
 - Generates run field widgets
 - Calculates collect commands shown at bottom right
 - Generates widgets on the left half of the tab, though basic widgets like video view have standard versions

10Hz small-spots image viewer

- Server reads image paths from an image “stream”, which is a Redis List that clients watch the length of and read only new items from
- Images are processed at up to 10Hz, and results are saved to another Redis stream in raw binary format with numpy
- Redis+Python+numpy make these steps simple to write and debug
- Processing makes single-pixel spots visible even at the small size shown (algorithm is by David Kissick @ GM/CA)

Shared collect site list

- Site list is a Redis hash of JSON strings
- This is a trade-off making debugging easy, especially with external scripts that may manipulate the list
- Raster mode stores sites in the list, and site mode uses them to collect

PyBluice-EPICS: Beamline SIM version v2021.1

File View Network Tools Help

Auto Sample **Collect** Analysis Scan SONICC Users Hutch

Raster

Video

Mode: Move sample, Video view, Grid view

Site define: Manual select, Filter select

Rotate: +90, -90

Zoom: In, Out

Res Lim: Low, High

Numbers: Off, Cell, Heat map

Heat map: Spots, Res., Int.

Click to move the sample

Spot finder

	Spots	Res.	Int.
1	80	4.58	26358
2	98	4.38	27632
3	92	4.40	24060
4	93	4.47	32485

Diffraction

Contrast: 0 100 Step: - +

Sites

	Source	Details
1	Manual	Cell 18
2	Manual	Cell 25
3	Manual	Cell 37
4	Manual	Cell 53

Control

Collecting

Pause

Current State

Collect Running

Activity Busy

FPE Running

Omega -0.02

Beam Atten.

Det. Z 300.0

Res. Predictor

Follow tools

advx

small spots

SMX

Logs

Collect log

Image log

FPE log

Energy log

Configure

Run 1 (collecting)

Copy Update Delete Reset

Mode

Strategy Single Vector

Raster Site

Paths

Aut Mai Reset Trm Brws

Prefix prot_efg15_3

Dir prot_efg15

Common

Det dist mm 300

Atten fact 100

Beam size μm

Delta deg 0.2

Exposure sec 0.2

Grid

Outline 184x130

Mouse define

Enter size

Clear

Cell width μm 20

Cell height μm 20

Progress

	Progress	Frames	Prefix	Angle	Motion	Energy
1	Done	10	prot_e...	0	200um	current
2	Done	10	prot_e...	0	200um	current
3	9	10	prot_e...	0	200um	current
4		10	prot_e...	0	200um	current
5		10	prot_e...	0	200um	current
6		10	prot_e...	0	200um	current

APS Current -0.0 mA Shutter Permit Disabled A Shutter Closed ES shutter Open ES secure No

Server log GUI log ETA --- Mono 12.660 keV EMERGENCY STOP IZero 2.15 V Control Active Fast shutter Closed



Acknowledgements

GM/CA

Janet Smith
Robert Fischetti
Kristin Ahrens
Craig Ogata
Nukri Sanishvili
Michael Becker
Naga Venugopalan
Sergey Stepanov
Oleg Makarov
Qingping Xu
Sudhir Pothineni
Shenglan Xu
David Kissick
Dale Ferguson
Steve Corcoran

Funding

NCI
NIGMS



U.S. DEPARTMENT OF
ENERGY

Argonne National Laboratory is a
U.S. Department of Energy laboratory
managed by UChicago Argonne, LLC.

MCE 2021 Workshop @ NSLS-II

Mark Hilgart

Argonne
NATIONAL LABORATORY

75
1946-2021