Allen Core framework

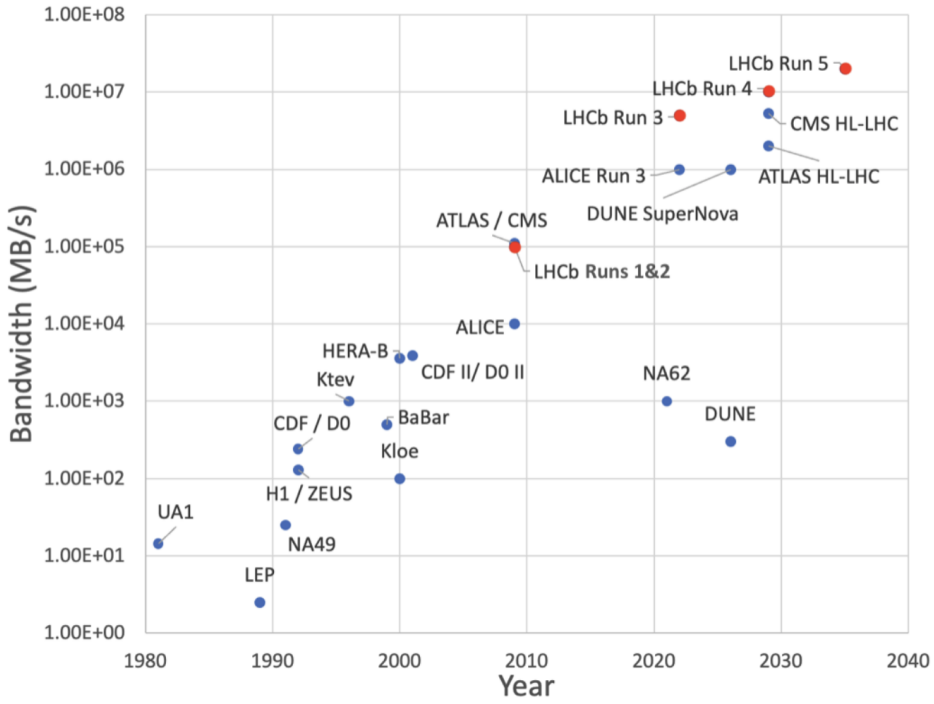
Gonzalo Díaz López – LPNHE

Workshop on high throughput heterogeneous computing I

28-30 January, ICJLab, Paris

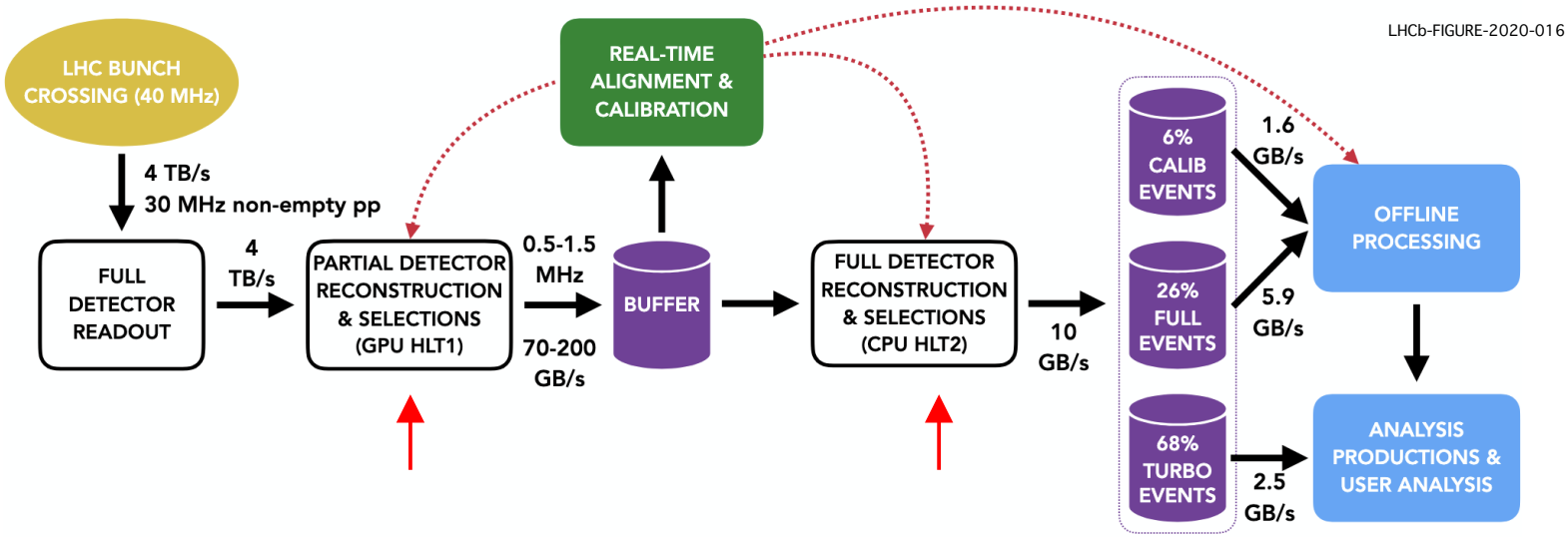
Context

- Bandwidth increase expected for HEP experiments in the near future — LHC high luminosity era
- Tackled by using accelerator devices — GPUs and FPGAs
- Infrastructure update needed
- LHCb successfully deployed a software based trigger for Run 3 — Allen



https://arxiv.org/abs/2503.24106

Context — LHCb trigger

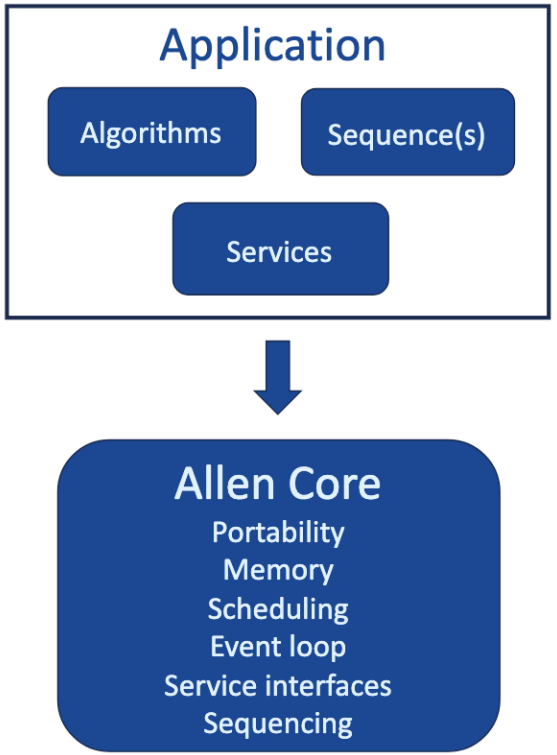


- Trigger is split in 2
  - HLT1 (Allen) — standalone application(*) runs on GPUs
  - HLT2 — Gaudi framework, runs on CPUs

* in fact it also depends on Gaudi so to use LHCb components

The idea — Allen Core

- Software **framework** for high throughput data processing using heterogeneous architectures — based on **Allen**
- Provides the tools **to create applications** without the need to deal with lower level components
- Features
  - **general use case** — many clients
  - **interoperable** — modular and extensible, interfaces
  - **easy to use** — simple and concise APIs
  - **documented** — fast conception to implementation
  - **tested** — on as many platforms and devices as possible



Repositories

- **Allen Core group** — https://gitlab.cern.ch/allen-core

The framework (preliminary working version)
https://allen-core.docs.cern.ch/

Application example (preliminary dummy version):
- Open Data Detector (planed)
- Benchmarking (planed)

How to create an application — overview

Allen Core Demonstrator

```
gdiazlop@lpnlp235:allen-core-demonstrator$ tree -L 2
.
├── algorithms
│   ├── CMakeLists.txt
│   ├── device
│   └── host
├── CMakeLists.txt
├── configuration
│   └── sequences
├── README.md
├── run_project.py.in
└── services
    ├── CMakeLists.txt
    ├── input_provider
    └── non_event_data
```

- directory structure and **cmake** build files
- algorithms
- sequence(s)
- services
  - input provider
  - non event data
- application launcher script

How to create an application — build system

```
1   cmake_minimum_required(VERSION 3.26)
2
3   project(Demonstrator VERSION 0.0.0 LANGUAGES CXX)
4
5   find_package(AllenCore REQUIRED)
6
7   # include from Allen Core
8   include(AllenCoreProjectSettings)
9   include(AllenCoreDependencies)
10  include(AllenCoreHelpers)
11  include(AllenCoreAlgorithmGeneration)
12
13  # set install path variables for Gaudi components
14  set(GAUDI_INSTALL_PLUGINDIR
15      "${CMAKE_INSTALL_PREFIX}/${CMAKE_INSTALL_LIBDIR}")
16  set(GAUDI_INSTALL_PYTHONDIR
17      "${CMAKE_INSTALL_PREFIX}/${CMAKE_INSTALL_LIBDIR}/python")
18
19  # define project pythondir - install sequences and algorithms.py here
20  set(PROJECT_PYTHONDIR "${GAUDI_INSTALL_PYTHONDIR}/${PROJECT_NAME}")
21  allencore_generate_pyinit("${PROJECT_PYTHONDIR}")
22
23  # add project subdirectories
24  add_subdirectory(algorithms)
25  add_subdirectory(services)
```

import Allen Core

define Gaudi dirs

define python dir

add project dirs

```
27  # generate and install algorithms.py
28  allencore_generate_algorithms("algorithms.py")
29  install(
30    FILES "${CMAKE_CURRENT_BINARY_DIR}/generated_code/algorithms.py"
31    DESTINATION "${PROJECT_PYTHONDIR}"
32  )
33
34  # generate the sequences:
35  # - move sequences to install dir
36  # - generate the json files
37  install(
38    DIRECTORY "configuration/sequences/"
39    DESTINATION "${PROJECT_PYTHONDIR}"
40    FILES_MATCHING PATTERN "*.py"
41  )
42  allencore_generate_sequences(
43    "${PROJECT_PYTHONDIR}"
44    "${CMAKE_INSTALL_PREFIX}/sequences"
45  )
46
47  # configure application launcher script
48  configure_file(
49    "${CMAKE_SOURCE_DIR}/run_project.py.in"
50    "${CMAKE_INSTALL_PREFIX}/${CMAKE_INSTALL_BINDIR}/run_project.py"
51    @ONLY
52  )
```

generate algorithms module

install and generate sequences

generate launcher script

How to create an application — algorithms

cmake target — Algorithms

```
gdiazlop@lpnlp235:algorithms$ tree
.
├── CMakeLists.txt
├── device
│   ├── CMakeLists.txt
│   ├── reconstruction
│   │   ├── CMakeLists.txt
│   │   ├── inc
│   │   │   └── Reconstruction.cuh
│   │   └── src
│   │       └── Reconstruction.cu
│   └── selection
│       ├── CMakeLists.txt
│       ├── inc
│       │   └── Selection.cuh
│       └── src
│           └── Selection.cu
└── host
    └── CMakeLists.txt
```

inc/Reconstruction.cuh

```
3   #include "kernel/algorithms/AlgorithmTypes.cuh"
4
5   namespace dummy_reconstruction {
6     struct Parameters {
7       HOST_INPUT(host_event_list_t, unsigned) host_event_list;
8       HOST_INPUT(host_raw_bank_version_t, int) host_raw_bank_version;
9       MASK_INPUT(dev_event_list_t) dev_event_list;
10      DEVICE_INPUT(dev_raw_input_t, char) dev_raw_input;
11      DEVICE_INPUT(dev_raw_input_offsets_t, unsigned) dev_raw_input_offsets;
12      DEVICE_INPUT(dev_raw_input_sizes_t, unsigned) dev_raw_input_sizes;
13      DEVICE_INPUT(dev_raw_input_types_t, unsigned) dev_raw_input_types;
14      DEVICE_OUTPUT(dev_reconstruction_t, float) dev_reconstruction;
15    };
16
17    struct dummy_reconstruction_t : public DeviceAlgorithm, Parameters {
18      void set_arguments_size(ArgumentReferences<Parameters>, const RuntimeOptions&
19          , const Constants&) const;
20
21      void operator()(
22        const ArgumentReferences<Parameters>&,
23        const RuntimeOptions&,
24        const Constants&,
25        const AllenCore::Context& context) const;
26
27      private:
28          AllenCore::Property<bool> m_property {this, "dummy_property", true, "dummy property"};
29          AllenCore::Property<dim3> m_block_dim {this, "block_dim", {32, 1, 1}, "block dimensions"};
30    };
31
32    __global__ void dummy_reconstruction(Parameters, const bool);
33  } // namespace dummy_reconstruction
```

Algorithms data deps.

Algorithm class

Algorithm kernel

src/Reconstruction.cu

```
16  REGISTER_ALGORITHM(dummy_reconstruction::dummy_reconstruction_t)
```

How to create an application — sequence

```
gdiazlop@lpnlp235:configuration$ tree
.
└── sequences
    └── dummy_sequence.py
```

```python
# Allen Core imports
# - sequencing tools
from allencore.sequencing.generator    import make_algorithm
from allencore.sequencing.control_flow import NodeLogic, CompositeNode
# - builtin algorithms
from allencore.sequencing.builtin_algorithms import host_init_event_list_t
from allencore.sequencing.builtin_algorithms import host_init_number_of_events_t
from allencore.sequencing.builtin_algorithms import dummy_data_provider_t

# Demonstrator imports
from Demonstrator.algorithms import dummy_reconstruction_t
from Demonstrator.algorithms import dummy_selection_t
```

Allen Core imports

project algorithms

```python
def sequence_node():

    # - algorithm 1: dummy data provider
    initialize_number_of_events = make_algorithm(
            host_init_number_of_events_t, name = "initialize_number_of_events"
            )
    initialize_event_lists = make_algorithm(
            host_init_event_list_t, name = "initialize_event_lists"
            )
    dummy_data_provider = make_algorithm(
            dummy_data_provider_t, name = "dummy_data_provider"
            )

    # - algorithm 2: dummy reconstruction
    dummy_reconstruction = make_algorithm(
        dummy_reconstruction_t, name = "dummy_reconstruction"
        , host_event_list_t       = initialize_event_lists.host_event_list_output_t
        , host_raw_bank_version_t = dummy_data_provider.host_raw_bank_version_t
        , dev_raw_input_t         = dummy_data_provider.dev_raw_banks_t
        , dev_raw_input_offsets_t = dummy_data_provider.dev_raw_offsets_t
        , dev_raw_input_sizes_t   = dummy_data_provider.dev_raw_sizes_t
        , dev_raw_input_types_t   = dummy_data_provider.dev_raw_types_t
        , dummy_property = True)
```

sequence node

input data

reconstruction

Selection lines and full node

```python
    # - algorithm 3: dummy selection line(s)
    line_1 = make_algorithm(
        dummy_selection_t, name = "dummy_selection_1"
        , host_number_of_events_t    = initialize_number_of_events.host_number_of_events_t
        , dev_reconstruction_input_t = dummy_reconstruction.dev_reconstruction_t
        , lower_lim = 1.0
        , upper_lim = 10.0
        , pre_scaler_hash_string  =  "dummy_selection_1" + "_pre"
        , post_scaler_hash_string =  "dummy_selection_1" + "_post")

    line_2 = make_algorithm(
        dummy_selection_t, name = "dummy_selection_2"
        , host_number_of_events_t    = initialize_number_of_events.host_number_of_events_t
        , dev_reconstruction_input_t = dummy_reconstruction.dev_reconstruction_t
        , lower_lim = 1.0
        , upper_lim = 5.0
        , pre_scaler_hash_string  =  "dummy_selection_2" + "_pre"
        , post_scaler_hash_string =  "dummy_selection_2" + "_post")

    # finally define full node and generate sequence
    full_node = CompositeNode(
        "dummy_sequence", [line_1, line_2]
        , NodeLogic.NONLAZY_AND, force_order=False
        )

    return full_node
```

Sequences are generated at compile time

How to create an application — Gaudi services

```
gdiazlop@lpnlp235:services$ tree
.
├── CMakeLists.txt
├── input_provider
│   ├── CMakeLists.txt
│   ├── inc
│   │   └── InputProviderSvc.h
│   └── src
│       └── InputProviderSvc.cpp
└── non_event_data
    ├── CMakeLists.txt
    ├── inc
    │   ├── Constants.cuh
    │   ├── Consumers.h
    │   ├── NonEventDataSvc.h
    │   └── Producers.h
    └── src
        ├── Consumers.cpp
        └── NonEventDataSvc.cpp
```

concrete interface functionalities under dev

inc/InputProviderSvc.h

- extends interface **IInputProviderSvc.h**

```
5   #include <GaudiKernel/Service.h>
6
7   #include <services/input_provider/IInputProviderSvc.h>
8
9   class InputProviderSvc final : public extends1<Service,
    AllenCore::Interfaces::IInputProviderSvc> {
10  public:
11
12      // TODO: dummy InputProviderSvc, does nothing
13
14      InputProviderSvc(std::string name, ISvcLocator* sl) : base_class{ name, sl } {}
15
16      StatusCode initialize() override;
17
18      size_t events_per_slice() const override {};
```

input_provider/CMakeLists.txt

- links AllenCore interface
- creates Gaudi module

```
8   target_link_libraries(InputProviderSvcInt
9     INTERFACE
10      AllenCore::IInputProviderSvcInt
11      AllenCore::global_flags
12  )
13
14  gaudi_add_module(InputProviderSvc
15                   SOURCES "src/InputProviderSvc.cpp"
16                   LINK PUBLIC GaudiKernel
17                               InputProviderSvcInt)
```

How to create an application — launcher script

run_project.py.ini

might be automatically generated by Allen Core in the future

```
1  import os
2  import sys
3
4  from cppyy import load_library
5  from cppyy import add_include_path, add_library_path
6
7  sys.path.append("@AllenCore_PYTHONDIR@")
8  from allencore.main.launcher             import run
9  from allencore.application.setup         import set_gaudi_runtime_env
10 from allencore.application.setup         import add_paths_recursively
11 from allencore.application.configuration import application_manager
12
13 set_gaudi_runtime_env( "@GAUDI_INSTALL_PLUGINDIR@"
14                        , "@GAUDI_INSTALL_PYTHONDIR@")
15
16 add_paths_recursively( "@CMAKE_INSTALL_PREFIX@/@CMAKE_INSTALL_LIBDIR@"
17                        , add_library_path)
18
19 # create Gaudi's Application Manager
20 appmgr = application_manager()
21
22 # include compulsory services (TODO: move to application_manager?)
23 appmgr.ExtSvc += ["NonEventDataSvc"]
24 appmgr.ExtSvc += ["InputProviderSvc"]
25
26 # include extra services used within this project's Algorithms
27 # appmgr.ExtSvc +=
28
29 # load project's Algorithms
30 # load_library("libHostAlgorithms.so")
31 load_library("libDeviceAlgorithms.so")
32
33 run()
```

Imports and setup to use Gaudi components and project (algorithm) libraries

Gaudi application manager configuration

Load project algorithms

Run the application

Running the application

App runner thread

Read CL arguments

Gaudi configuration and initialisation

Event Loop thread starts

Non Event Data generation & allocation

Instantiation of sequence algorithms

```
root@ccwgislurm0100:install-demo-cuda$ python bin/run_project.py -v 1 -i none -g none -s sequences/dummy_sequence.json
app runner           INFO Initializing Allen Core - 4.170s
app runner           INFO
------- Arguments -------
|- input_filenames   : none
|- detector_geometry : none
|- sequence_json     : sequences/dummy_sequence.json
|- number_of_events  : 0
|- number_of_slices  : 0
|- events_per_slice  : 1000
|- number_of_threads : 1
|- device_memory     : 1000
|- host_memory       : 200
|- verbosity         : 1
|- device            : 0
|- monitoring_filename: monitoring.root
----------------------- - 4.170s
app runner          DEBUG Initialize Gaudi application and get locator service - 4.170s
ApplicationMgr    SUCCESS
================================================================================================================
                                          Welcome to Allen Core (GaudiCoreSvc v40r1)
                                    running on ccwgislurm0100 on Thu Jan 22 19:10:35 2026
================================================================================================================
ApplicationMgr       INFO Application Manager Configured successfully
ApplicationMgr       INFO Application Manager Initialized successfully
ApplicationMgr    SUCCESS **************************** Algorithm Sequence ****************************
ApplicationMgr    SUCCESS ****************************************************************************
app runner          DEBUG Get ZeroMQ service and create control socket - 5.802s
app runner          DEBUG Get non event data service - 5.871s
app runner           INFO Launch event loop - 5.906s
event_loop           INFO Event loop started
app runner          DEBUG Wait for event loop connection - 5.944s
event_loop          DEBUG Using device Tesla V100-SXM2-32GB:0
event_loop          DEBUG Connection established with control socket at inproc://allencore_control
event_loop          DEBUG Reading sequence: sequences/dummy_sequence.json
app runner           INFO Connection established with even loop thread - 6.018s
app runner          DEBUG Send signal to start the event loop - 6.018s
INonEventDataSv...  DEBUG Reserve memory for algorithm constants
INonEventDataSv...  DEBUG Registry check successful
Dummy::produce      DEBUG Data produced for id - dummy
Dummy::consume      DEBUG Consumed data for id - dummy
HostBuffersMana...  DEBUG 2 host buffers, of which 2 are empty and 0 filled
event_loop          DEBUG Sequence algorithms initialized successfully
ScheduledSequen...   INFO Sequence:
                                initialize_event_lists
                                dummy_data_provider
                                dummy_reconstruction
                                initialize_number_of_events
                                dummy_selection_2
                                dummy_selection_1
```

device

threads
- app runner (control)
- event loop (coordinator)
- streams (sequences)
- I/O

Running the application

Caution
This is not a benchmark test, but a demonstration of portability

GPU device — Tesla V100-SXM2-32GB

```
event_loop           INFO Running stream test
Scheduler::run      DEBUG Invoking initialize_event_lists
Scheduler::run      DEBUG Invoking dummy_data_provider
Scheduler::run      DEBUG Invoking dummy_reconstruction
dummy_reconstru...  DEBUG Running workload
dummy_reconstru...  DEBUG Time: 0.000185064 seconds
Scheduler::run      DEBUG Invoking initialize_number_of_events
Scheduler::run      DEBUG Invoking dummy_selection_2
Scheduler::run      DEBUG Invoking dummy_selection_1
event_loop           INFO Event loop finished
app runner           INFO Allen Core ran successfully - 6.308s
ToolSvc              INFO Removing all tools created by ToolSvc
ApplicationMgr       INFO Application Manager Finalized successfully
ApplicationMgr       INFO Application Manager Terminated successfully
```

CPU device — Intel Xeon Silver 4214R

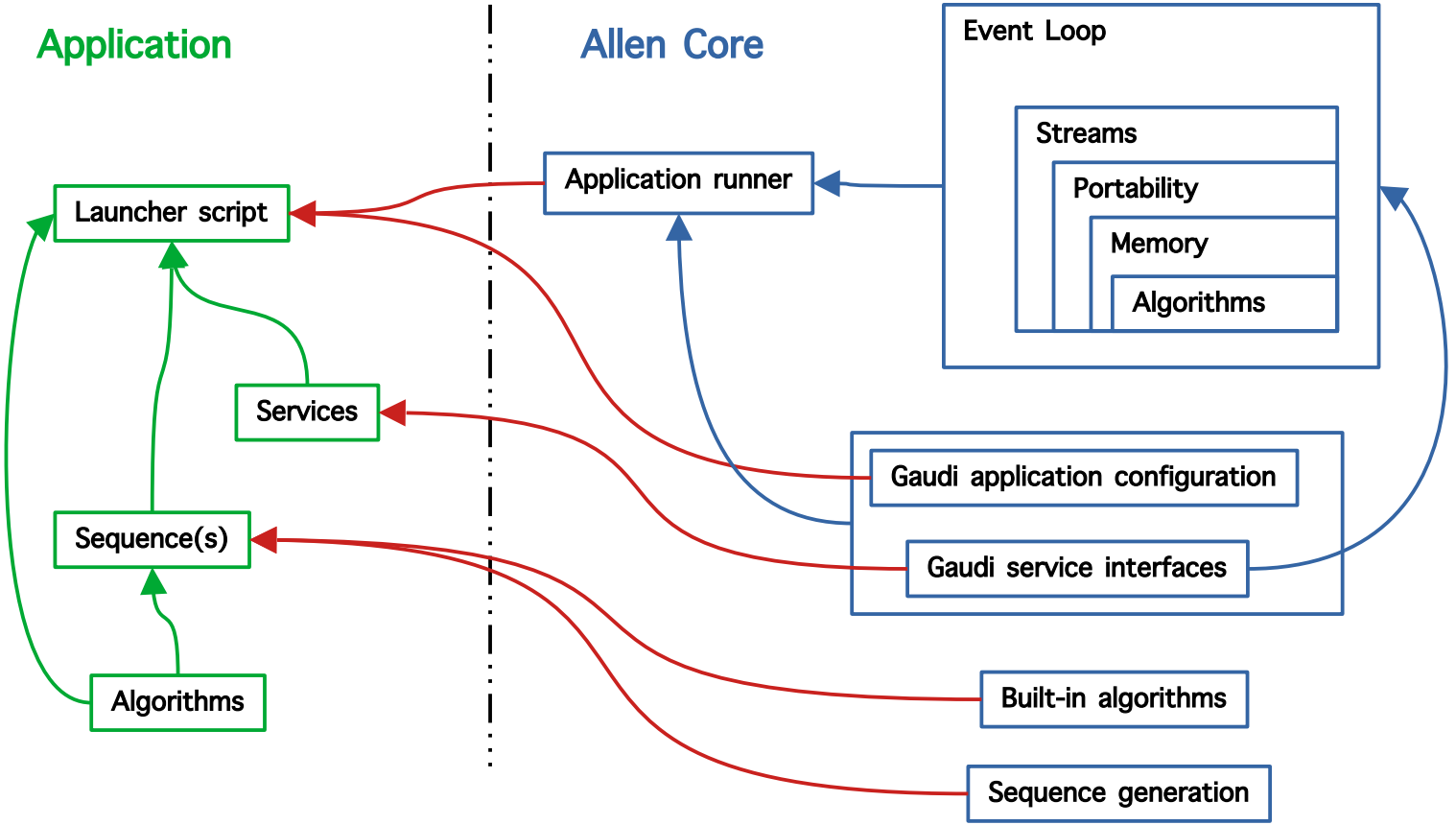
```
event_loop           INFO Running stream test
Scheduler::run      DEBUG Invoking initialize_event_lists
Scheduler::run      DEBUG Invoking dummy_data_provider
Scheduler::run      DEBUG Invoking dummy_reconstruction
dummy_reconstru...  DEBUG Running workload
dummy_reconstru...  DEBUG Time: 1.91182 seconds
Scheduler::run      DEBUG Invoking initialize_number_of_events
Scheduler::run      DEBUG Invoking dummy_selection_2
Scheduler::run      DEBUG Invoking dummy_selection_1
event_loop           INFO Event loop finished
app runner           INFO Allen Core ran successfully - 7.983s
ToolSvc              INFO Removing all tools created by ToolSvc
ApplicationMgr       INFO Application Manager Finalized successfully
ApplicationMgr       INFO Application Manager Terminated successfully
```

algorithms/device/src/Reconstruction.cu

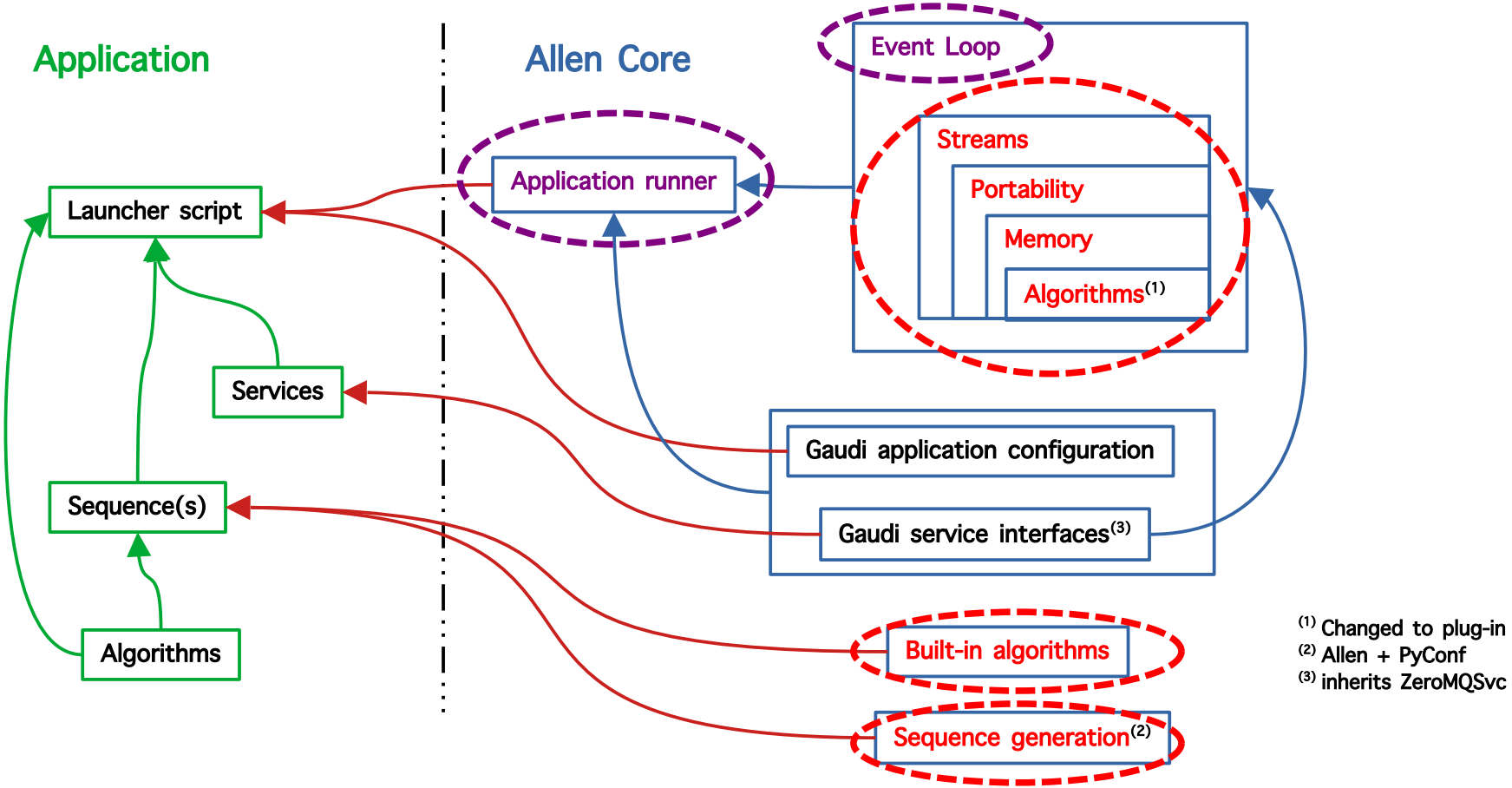
Total number of iterations = 10^8, since
gridDim = (100, 100, 10)
blockDim = (1000, 1, 1)
The length of dev_reconstruction is 10^8

```
__global__ void dummy_reconstruction::dummy_reconstruction
(dummy_reconstruction::Parameters parameters, const bool property)
{
    for (unsigned i=threadIdx.x; i<1e3; i+=blockDim.x){
        uint32_t idx = blockIdx.x * gridDim.z * gridDim.y
                     + blockIdx.y * gridDim.z
                     + blockIdx.z * blockDim.x + threadIdx.x;
        parameters.dev_reconstruction[idx] = 1664525*i + 1013904223; //ranqd1
    }
}
```

Allen Core components

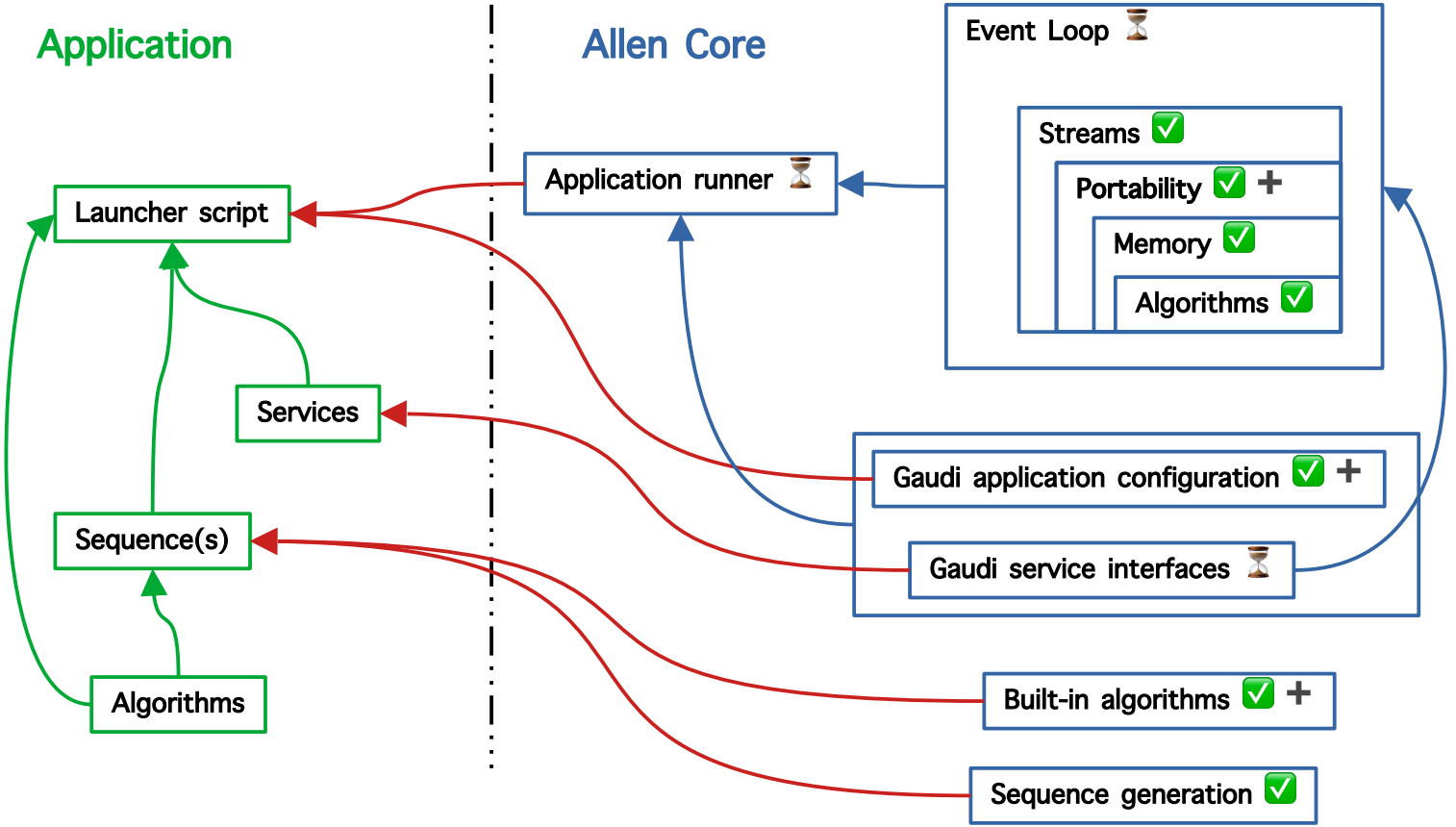


Allen Core components — Where is Allen?



Allen Core components — status

Completed ✅
Extensible ✅ +
In progress ⏳




```
gdiazlop@lpnlp235:allen-core$ tree -L 1
.
├── algorithms
├── ci
├── cmake
├── CMakeLists.txt
├── configuration
├── doc
├── kernel
├── main
├── README.md
├── requirements.txt
├── services
└── test
```

Next steps

- **Allen Core**
  - ⬆ I/O Data, Non Event Data and Monitoring interfaces
  - ⬆ Finish event loop (single stream thread now) — I/O threads
  - Define and implement (unit) tests
  - Extend portability — optimise for CPU and add more devices
- **Allen Core Demonstrator**
  - ⬆ Sequence: input data and reconstruction algorithms (ACTS interoperability?)
  - ⬆ Non Event Data service: Open Data Detector geometry and conditions
  - Monitoring service
  - Define and implement benchmark tests
- **Build system**
  - Implement CVMFS builds

Summary, conclusions and last comments

- **Allen Core** is basically **Allen**'s performance critical parts with a system to build external applications
- Some significant differences though
  - Use of Gaudi as a system for services — avoid standalone/non-standalone builds as in Allen, now algorithms can always use them
  - Plug-in system to register algorithms using an AlgorithmRegistry — dropped Allen's AlgorithmDB
  - Geometry and conditions to be updated through a service instead of a "fake" event loop as in Allen
  - Improved logging — based on Gaudi's MessageSvc for consistency
  - Event loop and control (application runner) fully refactored

- First working version (v0.1) of **Allen Core** — proven ability to run heterogenously
- Ready to use environments provided — **Allen Core Containers**
- Incomplete for production — many to-dos (I/O, Non Event Data, Monitoring…)
- Early stage of development, needs benchmarking and testing

Thanks for your attention

Backup

How to create an application — Gaudi services Non Event Data

preliminary approach

```
gdiazlop@lpnlp235:non_event_data$ tree
.
├── CMakeLists.txt
├── inc
│   ├── Constants.cuh
│   ├── Consumers.h
│   ├── NonEventDataSvc.h
│   └── Producers.h
└── src
    ├── Consumers.cpp
    └── NonEventDataSvc.cpp
```

- **Constants.cuh** defines the data used by the algorithms — e.g. a detector geometry class
- **Producers.h** instantiates data in host memory
- **Consumers.h** copies data to device (if used)

src/Consumers.cpp

copy data to device

```
void Consumers::Dummy::consume(const std::vector<std::any>& data){

    // TODO: dummy, just reads and allocates an integer

    // reads producer data (it is host data)
    const int* host_dummy = std::any_cast<int>(&data[0]);

    // get reference to device data variable
    int*& dev_dummy = m_constants.get().dev_dummy;

    AllenCore::malloc((void**) &dev_dummy, sizeof(int));
    AllenCore::memcpy(dev_dummy, host_dummy, sizeof(int), AllenCore::memcpyHostToDevice);

    debug_cout << "Consumed data for id - " << getid() << endmsg;
}
```