Running AI in JANA2

Running AI/ML based projects **in production**

in HENP software based on JANA2 framework

Dmitry Romanov

Contributors:

David Lawrence
Cissie Mei
Nathan Brei
Maxim Potekhin
Sergey Furletov

July 27, 2023

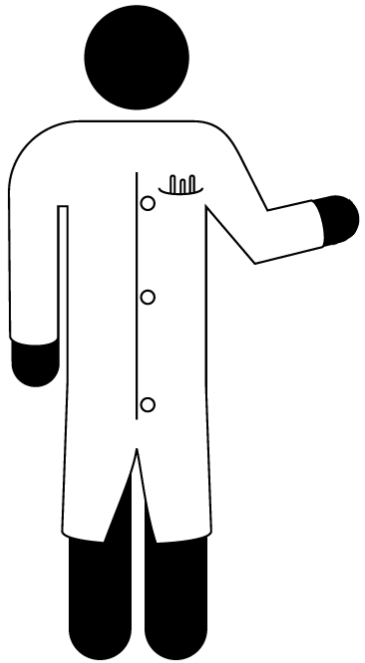
Jefferson Lab

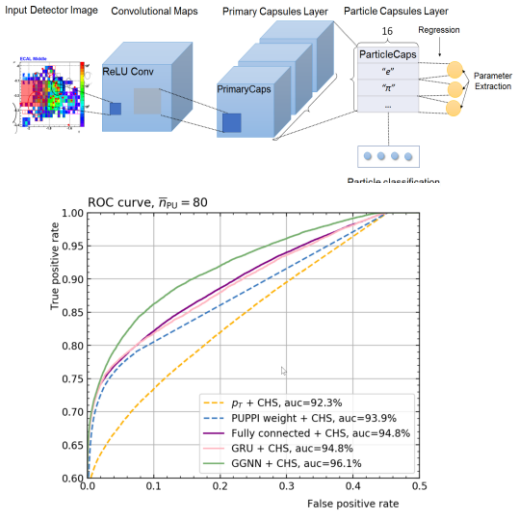
U.S. DEPARTMENT OF ENERGY | Office of Science

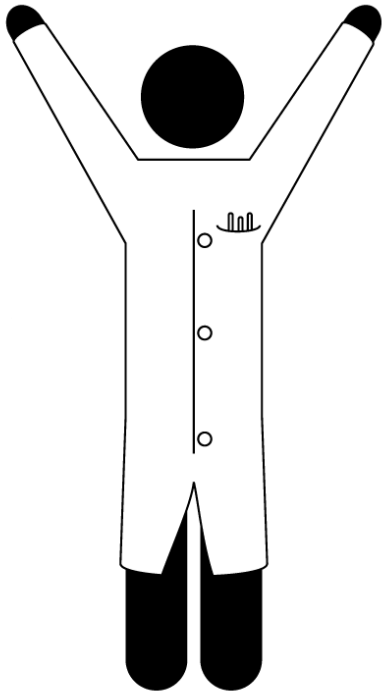
JSA

AI project application use case

- Scientist investigated AI/ML project
- Popular AI/ML libraries are used (PyTorch, Tensorflow, TMVA)
- Project works good enough to be run in production
- **How to run it in production?**







Jefferson Lab

Running model in production chain

Runnable:
- Real-time
- Batch

Data pipelines defined

Administration governance, stewardship

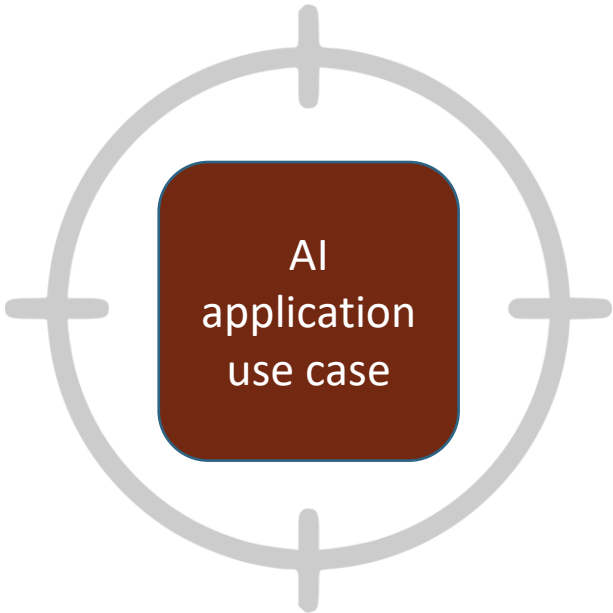
Integration with data frameworks

Feature extraction

Model development

Algorithm Selection

Models running in production



Running model in production chain

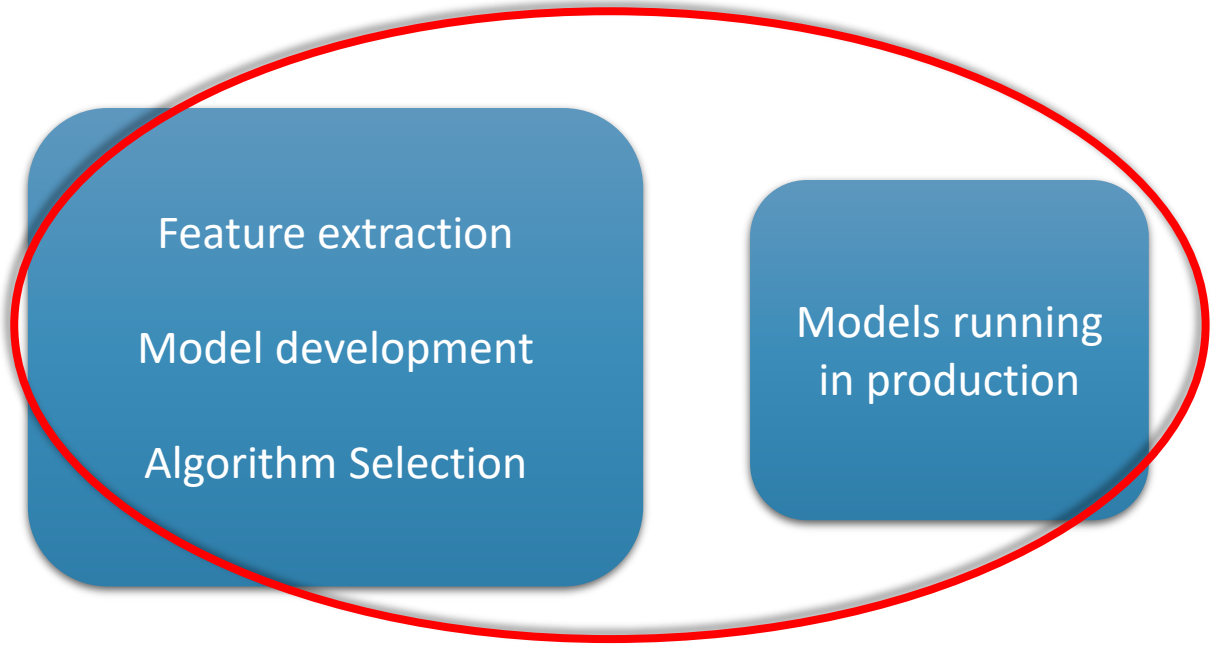
Runnable:
- Real-time
- Batch

Data pipelines defined

Administration governance, stewardship

Integration with data frameworks

Many projects of AI application for EIC is being made today



AI4EIC 2021 - 2023
November 28–December 1, CUA, Washington, DC.
https://indico.bnl.gov/event/19560/

Running model in production chain

Runnable:
- Real-time
- Batch

Related to Computing Model definition
Running both is not relevant as of today

Data pipelines defined

Some data infrastructure is there
Optimal is to be defined
Goes very close to reproducibility topic

Administration governance, stewardship

Can work as other algorithms as of now
ML life cycle management to be defined
Optimal is to be defined

Integration with data frameworks

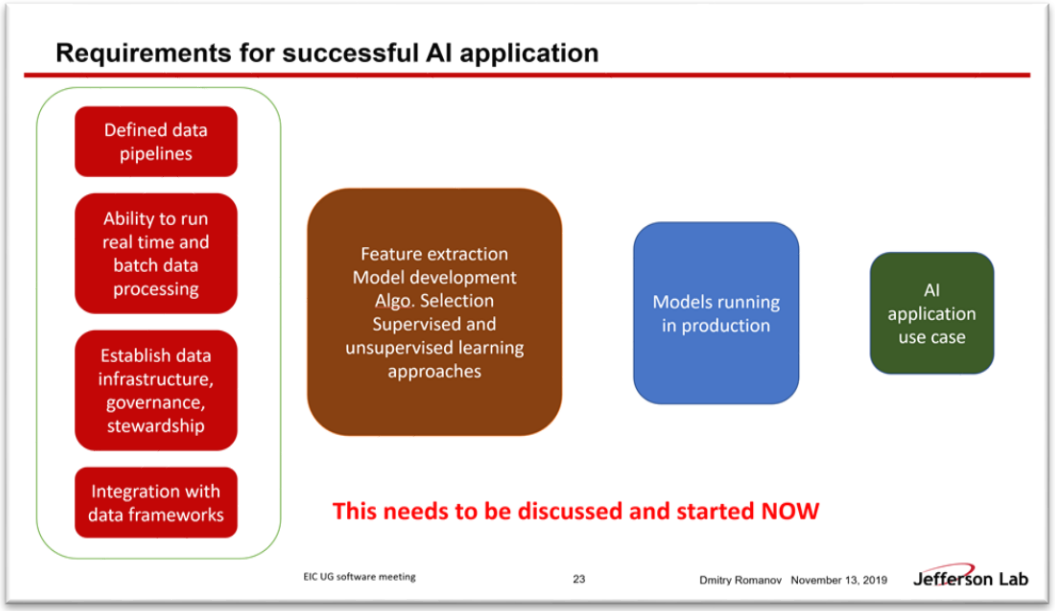
This talk is about it!
Many working examples

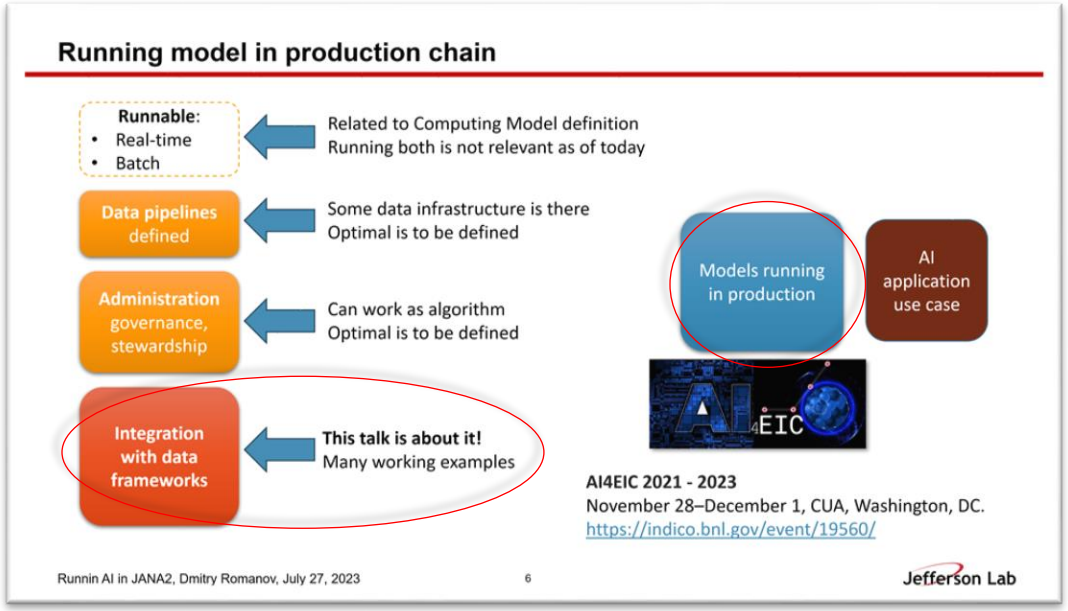
Models running in production

AI application use case

AI4EIC 2021 - 2023
November 28–December 1, CUA, Washington, DC.
https://indico.bnl.gov/event/19560/

Tremendous progress





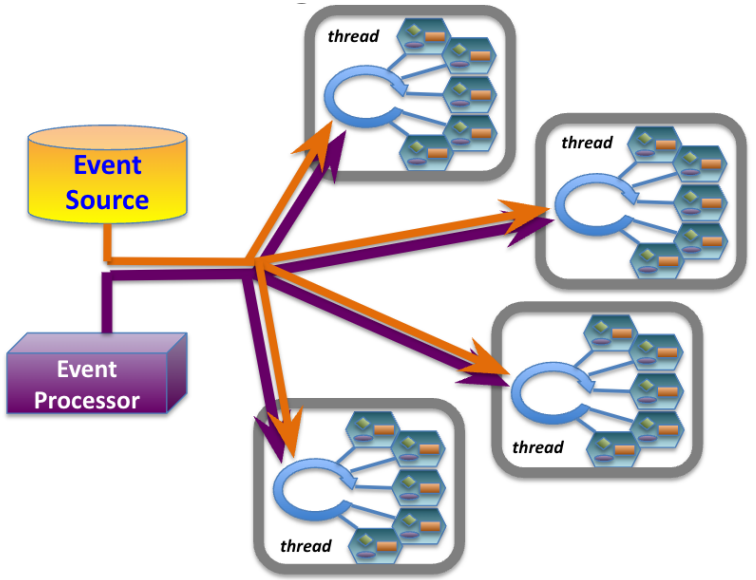
Over the last several years we went from bare planning to ready to use infrastructure

JANA2 overview

- JANA2 second generation C++ framework with nearly 2 decades of experience behind it

- Modern coding and CS practices
- Streaming DAQ and heterogeneous hardware support
- Active development
- ***EICrecon*** implements algorithms for ePIC in JANA2
  https://github.com/eic/EICrecon

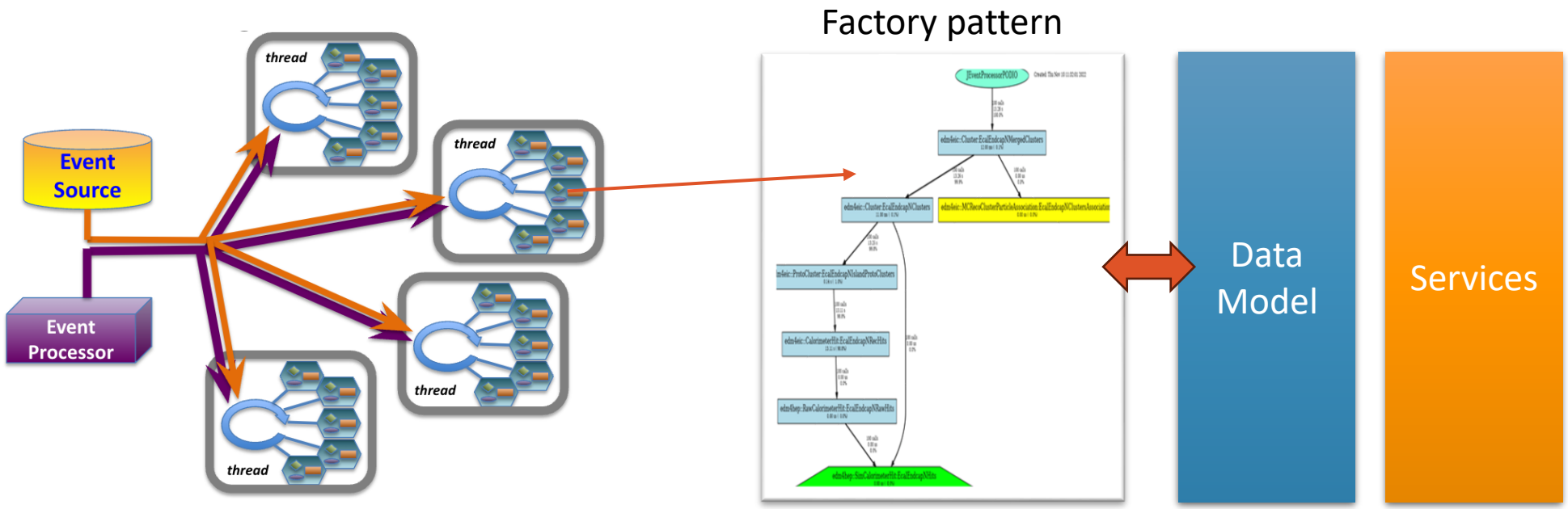
Documentation: https://jeffersonlab.github.io/JANA2/
Examples: https://github.com/JeffersonLab/JANA2/tree/master/src/examples
Example projects: https://github.com/JeffersonLab/JANA2_Examples
https://github.com/fJeffersonLab/EIC_JANA_Example/tree/TObject_example



Publications:

https://arxiv.org/abs/2202.03085 *Streaming readout for next generation electron scattering experiments*
https://doi.org/10.1051/epjconf/202125104011 *Streaming Readout of the CLAS12 Forward Tagger Using TriDAS and JANA2*
https://doi.org/10.1051/epjconf/202024501022 *JANA2 Framework for Event Based and Triggerless Data Processing*
https://doi.org/10.1051/epjconf/202024507037 *Offsite Data Processing for the GlueX Experiment*

JANA2 functional parts

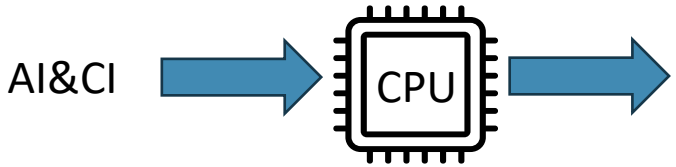


There are several ways how AI algorithms could be integrated into JANA2
Possible toolset is:

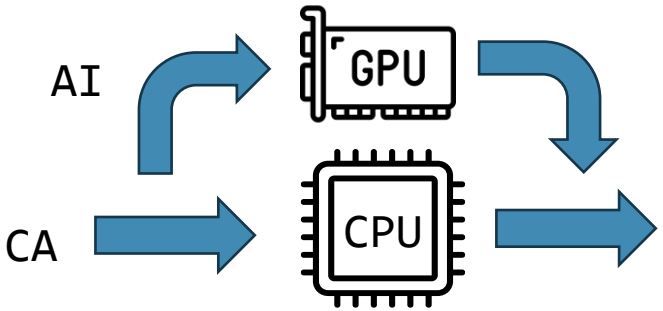
- Event level – factories, subevent offloading
- Data model level – utilizing columnar storage (theoretical)
- Global level support - services

Types of running tasks with JANA

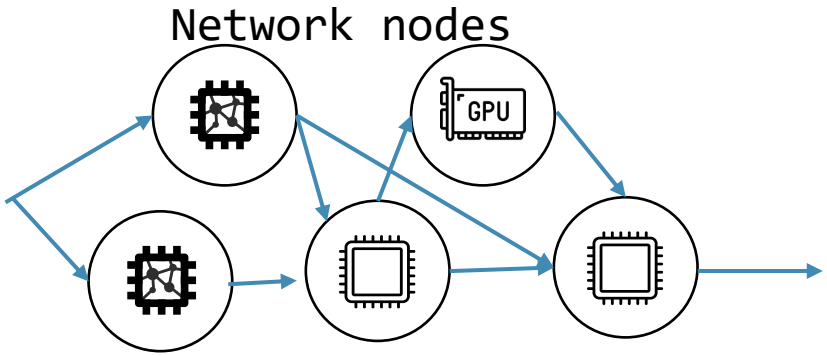
1. **Processor operated** (regular JANA2 flow)
   Working examples: **Tensorflow-lite in GluEx**



2. **Heterogenous hardware**
   Working examples:
   - JANA2 CUDA example
   - Phasm – surrogate model



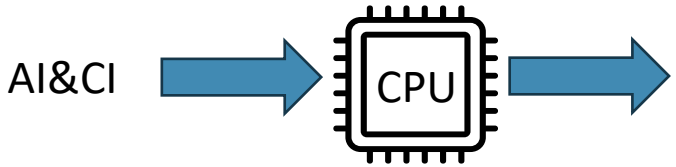
3. **Distributed calculation**
   Working examples: JANA4ML4FPGA



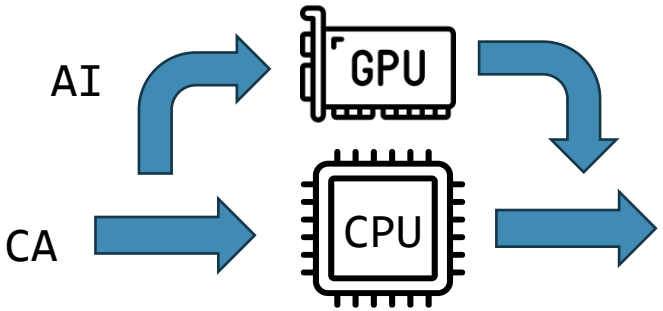
CA – conventional algorithms

Types of running tasks with JANA

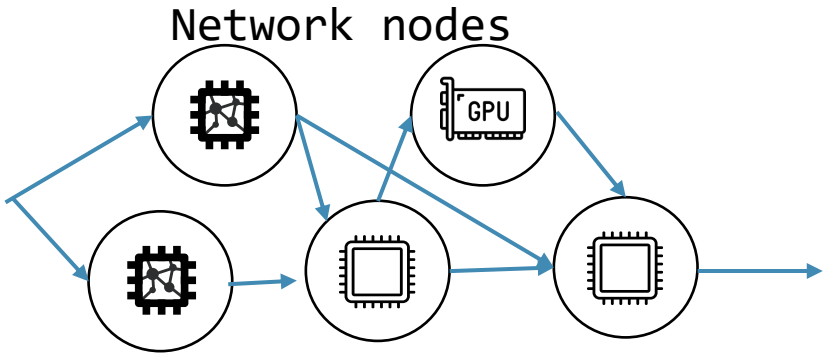
1. **Processor operated** (regular JANA2 flow)
   Working examples: **Tensorflow-lite in gluex**



2. **Heterogenous hardware**
   Working examples:
   - JANA2 CUDA example
   - Phasm – surrogate model



3. **Distributed calculation**
   Working examples: JANA4ML4FPGA



CA – conventional algorithms

CPU processing vs GPU/TPU processing

Most common bottlenecks when using GPUs (sorted by how common it is):

1. **Data Transfer:** The time it takes to move data from the CPU to the GPU memory is often the most significant bottleneck.
2. **Memory Constraints:** GPUs have limited memory compared to CPUs, which can become a significant bottleneck for large models or datasets.
3. **Problem Size:** If the task or problem size is small, the GPU may not be fully utilized, which means you don't get as much benefit from the parallel processing power of the GPU.
4. **Parallelism:** Not all tasks can be efficiently parallelized. For tasks that have a high degree of inherent sequential processing, the benefits of using a GPU may be limited.
5. **Coding Complexity and Maintenance:** Writing efficient code for GPUs can be more
6. **Farms of GPU configuration (or its absence):** How to run the software on different farms hardware sets and be sure it produces the same results?

Outline: Sometimes (often) ML algorithms are are simple and effectively run on CPU so no need in overcomplicating.

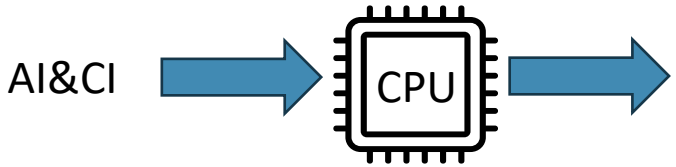
Packages to build with

Packet	Spack	Conda	Building from scratch
Tensorflow	✓	✓	Very hard
Tensorflow-lite	✓	✓	Hard
PyTorch	✓	✓	Hard
TMVA	✓	✓	Part of ROOT
JANA2	✓	In process	Easy
EPIC stack	✓ - In container	X – not planned	Medium

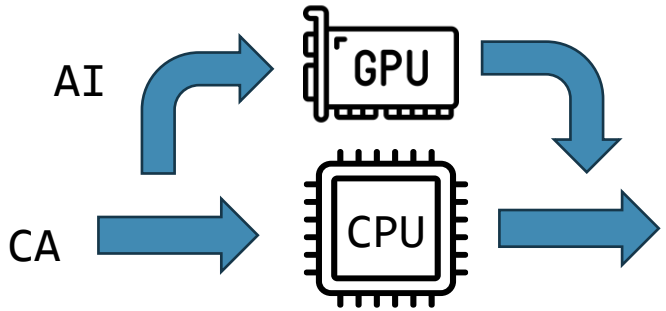
Strategies how to develop with EPIC software and EICrecon:
1. Extend existing container and install AI packages
2. Use conda and build EPIC stack there

Types of running tasks with JANA

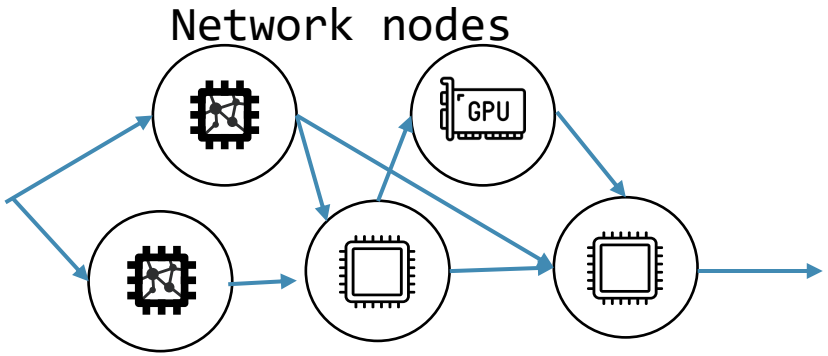
1. **Processor operated** (regular JANA2 flow)
   Working examples: **Tensorflow-lite in gluex**



2. **Heterogenous hardware**
   Working examples:
   - JANA2 CUDA example
   - Phasm – surrogate model



3. **Distributed calculation**
   Working examples: JANA4ML4FPGA



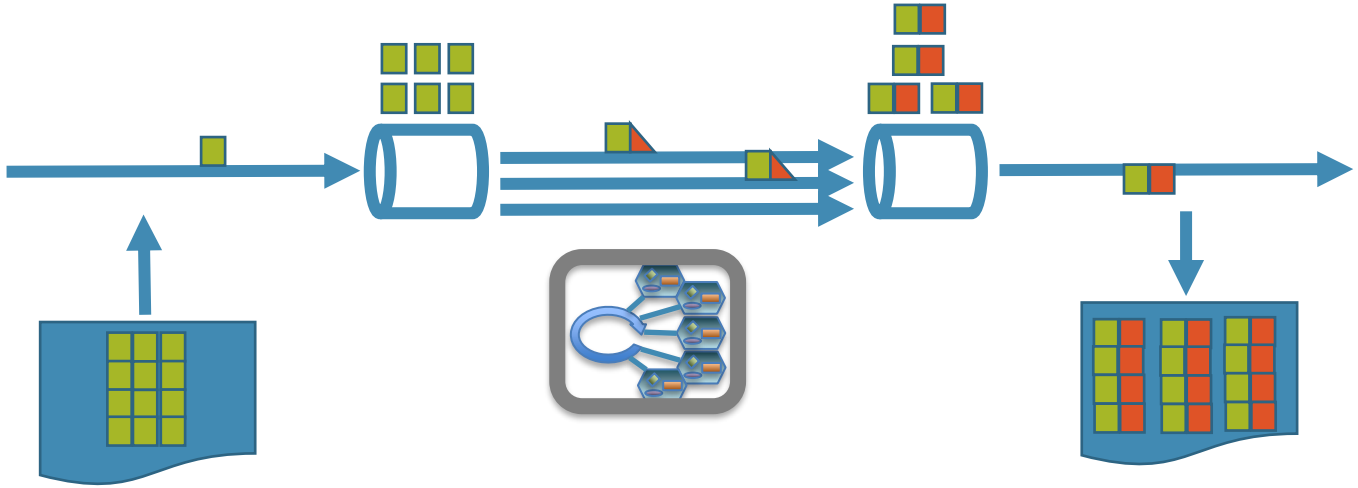
CA – conventional algorithms

A graphical notation for queues and arrows

Sequentially, read events from input file (e.g. sim or DAQ)

In parallel, run all algorithms for one event (in parallel)

Sequentially, write events to output file



- Arrows are allowed produce a different number of output objects than they consume, and to choose from different output queues
- This allows us to do things like
  - Split/merge
  - Gather/scatter
  - Filtering
  - Streaming joins
- This lets us have different units of parallelism at different points in our computation
  - Event blocks
  - Events
  - Subevents

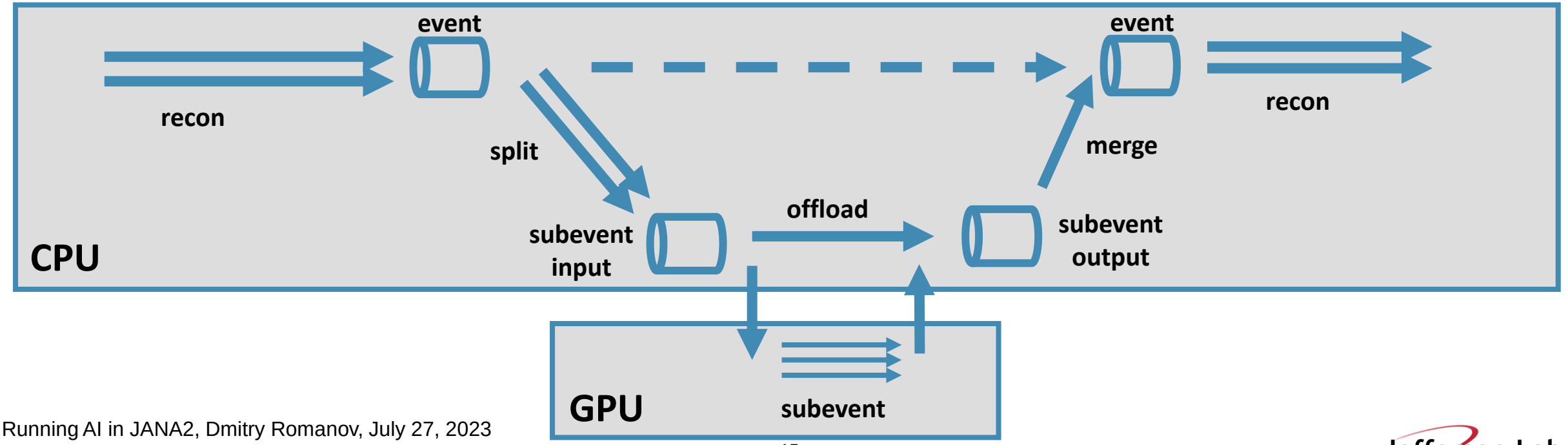
- Heterogeneous Computing and SRO in JANA2, Nathan Brei, January 9, 2023
- JANA2: Multi-threaded Event Reconstruction - David Lawrence - JLab - HSF Framework WG Sep. 21, 2022

Het Hardware: Subevents/Event collating

Problem: Offloading the data to a GPU/TPU/etc is a **sequential bottleneck**! We want to **batch** our data so that we can utilize the full parallelism of the heterogeneous hardware.

Solution: Split/merge pattern:

- Split each event into subtasks
- Batch subtasks together, **possibly across event boundaries**
- Send the batched subtasks to the hardware and wait for the results
- Attach the subtask results to their corresponding event



JANA2 CUDA example

- Example on GitHub (JANA2/Examples)
- Minimal example of using CUDA processing
- Using subevent parallelism
- Creating custom processing topology
- Try running it!

Run on JLab ifarm GPUs

This is an example that can run on a single GPU.

1. On the `ifarm` node, ask for a GPU compute node with 8000 MB memory.

```
module load cuda

srun --gres gpu:${gpu_type}:1 -p gpu --mem-per-cpu=8000 --pty bash
```

${gpu_type} can be `T4`, `TitanRTX` or `A100`.

2. On the compute node, build the whole application with cmake option "USE_CUDA=On".

```
# bash-4.2$ rm -rf build/*
bash-4.2$ cmake -DUSE_CUDA=On -S . -B build  # add the cmake option to config
#### output
#-- Found CUDA: /apps/cuda/11.4.2 (found version "11.4")
bash-4.2$ cmake --build build -j 32 --target install  # build and install as normal
```

Launch the application.

```
bash-4.2$ ./install/programs/SubeventCUDAExample
```

Topology creation from the example

```
142         auto topology = app.GetService<JTopologyBuilder>()->create_empty();
143         auto source_arrow = new JEventSourceArrow("simpleSource",
144                                                   {source},
145                                                   &events_in,
146                                                   topology->event_pool);
147         auto proc_arrow = new JEventProcessorArrow("simpleProcessor", &events_out, nullptr, topology->event_pool);
148         proc_arrow->add_processor(new SimpleProcessor);
149
150         topology->arrows.push_back(source_arrow);
151         topology->sources.push_back(source_arrow);
152         topology->arrows.push_back(split_arrow);
153         topology->arrows.push_back(subprocess_arrow);
154         topology->arrows.push_back(merge_arrow);
155         topology->arrows.push_back(proc_arrow);
156         topology->sinks.push_back(proc_arrow);
157
158         source_arrow->attach(split_arrow);
159         split_arrow->attach(subprocess_arrow);
160         subprocess_arrow->attach(merge_arrow);
161         merge_arrow->attach(proc_arrow);
162
163         app.Run(true);
```

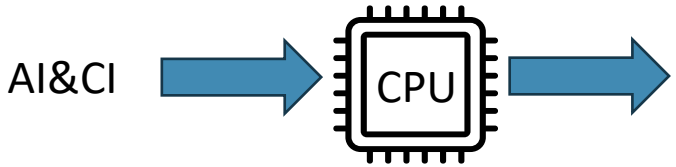
Using PyTorch with JANA2 – Phasm project

- **PHASM** ("**P**arallel **H**ardware vi**A** **S**urrogate **M**odels")
- Research work on experimental toolkit focused on getting old,legacy code to run on shiny, new, highly parallel hardware, by replacing the most intensive parts of the code with neural net surrogate models.
- LDRD wiki page
- GitHub page
- PyTorch attached as a plugin (code)
- Model lifecycle management (MLFlow)
- Recommended example

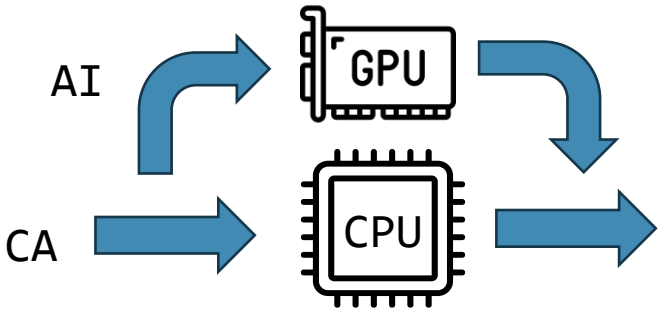
Jefferson Lab

Types of running tasks with JANA

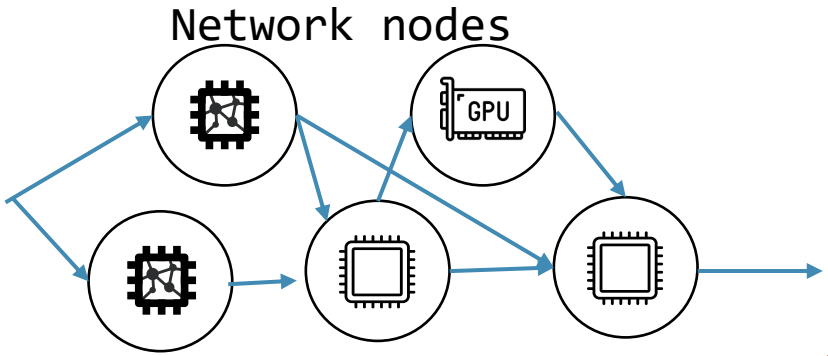
1. **Processor operated** (regular JANA2 flow)
   Working examples: **Tensorflow-lite in gluex**



2. **Heterogenous hardware**
   Working examples:
   - JANA2 CUDA example
   - Phasm – surrogate model

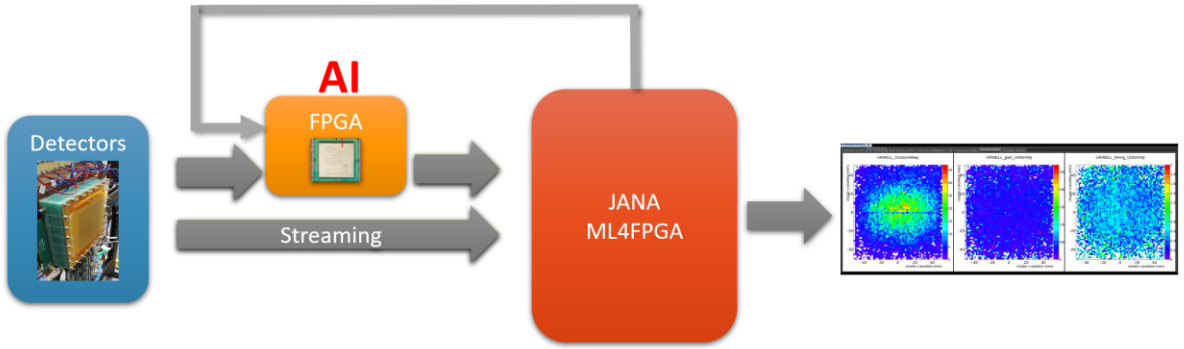


3. **Distributed calculation**
   Working examples: JANA4ML4FPGA

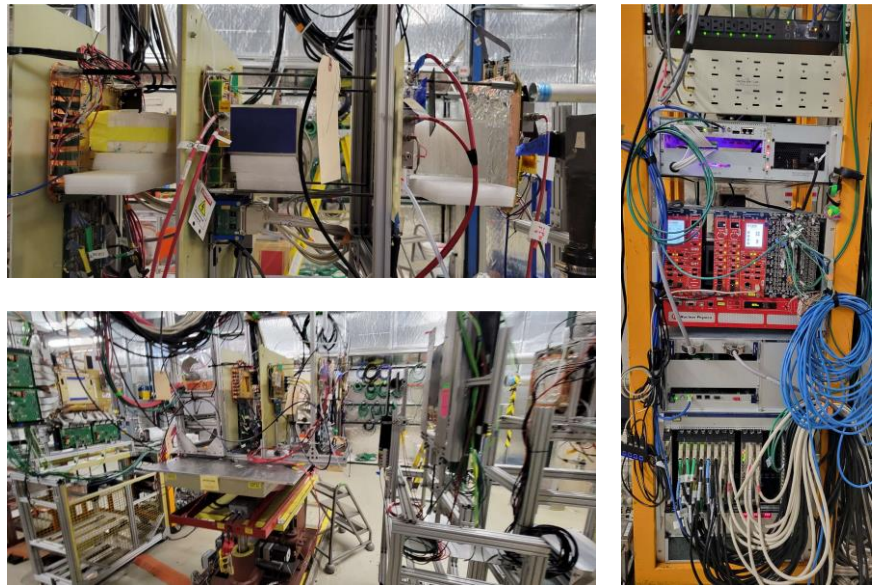


CA – conventional algorithms

ML4FPGA – JANA4ML4FGPA



- Close to detector AI in FPGA
- Communicates with JANA over network
- JANA is used as a node core/algorithm runner
- Node capabilities:
  - Receive FPGA processed stream
  - Communicate to FPGA (send data for processing)
  - Process raw stream
  - IO, event building operations
- JANA4ML4FPGA at GitHub



- Jlab tests winter/spring 2023
- Fermilab tests 2023
- **CERN tests (going on right now)**

Summary

Thank you!

- There is an increased interest in application of AI/ML projects in ePIC simulation and reconstruction software
- Current infrastructure allows to start putting ML algorithms in production (especially CPU based)
- Optimal data management and ML lifecycle procedures are to be planned and implemented
- JANA2 is a core framework for EPIC reconstruction software was created with utilizing heterogenous software in mind.
- The work is going on different AI projects using JANA2 which could be used as examples
- Waiting for the first algorithms to be integrated in ePIC!

Special thanks to:
- David Lawrence
- Cissie Mei
- Nathan Brei
- Maxim Potekhin
- Sergey Furletov

BACKUP