CLAS12 remote data-stream processing using the ERSAP framework

Παντα ρει

V. Gyurjyan
gurjyan@jlab.org
JLAB Experimental Halls

- Four experimental end-stations with different experimental equipment.
- Current and upcoming experiments require increased data acquisition, driving the demand for streaming technology.
“Enable full offline analysis chains to be ported into real-time, and develop frameworks that allow non-expert offline analysis to design and deploy physics data processing systems.”

A Roadmap for HEP Software and Computing R&D for the 2020s. HEP Software Foundation, Feb. 2018
JLAB Grand Challenge in Readout and Analysis for Femtoscale Science

Courtesy of Amber Boehnlein, et al.

* AI/ML

Courtesy of David Lawrence
• CPU runs a software component ROC. It is responsible for payload board configuration and readout, as well as data formatting and passing it to the next stage.

• VTP relieves the ROC of all the “Readout” tasks and implements them in the FPGAs.
• Triggered or Streaming readout from ALL payload modules in parallel
• The Software ROC is now primarily responsible for configuring, controlling, and monitoring the VTP-based DAQ.

• TI Trigger interface card, responsible for trigger and clock distribution.
The event loop is part of the application. They are deployed as a monolith.
ERSAP: Environment for Real-time Streaming, Acquisition, and Processing Framework

- Event reactive actors, networked by data pipelines.
- Compositional actors with conditional data routing at runtime.
- Flow-based programming paradigm

DPE: Data Processing Environment
SM: Shared Memory
DPS: Data Processing Station
UE: User Engine

Init(Object O)
Object process(Object O)
Data processing station: actor

- User *engine* run-time environment.
- Engine follows data-in/data-out interface.
- Engine gets JSON object for run-time configuration.

---

**UE : User Engine Interface**

- `init(JSON O)`
- `Object process(Object O)`
- `Object process(Object[] O)`
- `Object process(Map<String, Object> O)`
- `Object[] process(Object[] O)`
- `Map<String, Object> process(Map<String, Object> O)`
Data Acquisition and Processing Pipeline Designer
What do we need for successful data stream processing?

1. A framework capable of designing distributed data processing applications.
2. Low latency, reliable data-stream transport, and load-balancing system.
3. Elastic remote resource allocation, workflow deployment, and orchestration system.
EJFAT: ESnet FPGA Accelerated Transport System

EJFAT System Architecture

- **Load Balancer**
  - Host Control Plane
  - FPGA Data Plane

- **Control Plane**
  - Keep schedule density (proportional # of slots each host is allocated)
  - From host feedback, set new schedule density
  - From source feedback, set epoch boundary for change to new schedule
  - Update behavior of data plane

- **Switch**
  - Colors → Events
  - Shapes → Data Sources

- **Host registers & sends metrics**
  - Host 1
    - Event Building
    - Analysis

- **Event Building**
  - Event 1 Src 1
  - Event 1 Src 2

- **Reassembly**
  - Port 1
  - Port 2

Source sends latest event #

Courtesy of Mike Goodrich
• Elastic, distributed Kubernetes cluster based on opportunistic resources from various computing facilities that function independently of provider-specific setup requirements.
• Seamlessly deploys and scales user workloads over multiple computing facilities to ensure the best use of resources.
• Proactive resource provisioning based on ML models.
• Workflow-facility digital twin based on agent-based Bayesian probabilistic graph model.
Concept Validation Experiment
CLAS12 Data-Stream processing at NERSC
CLAS12 Stream Event Reconstruction: JLAB – Esnet - NERSC

JIRIAF Dashboard at JLAB

15
Summary

• An Event-driven reactive actor-based framework is under development at Jefferson Lab

• Along with simplifying user application development (e.g. data transport and multithreading) it provides distributed workflow orchestration, simplifying user application migration and deployment.

• It can adopt traditional data processing applications into streaming by suggesting user application decomposition followed by reactive actor representation.

• For applications that are difficult to modify, it provides remote provisioning for data files that are opaque to users.

• Using ERSAP, EJFAT, and JIRIAF, we successfully demonstrated remote data-stream processing for the first time.
Thank You