Streaming Readout and Data-Stream Processing with ERSAP

Παντα ρει

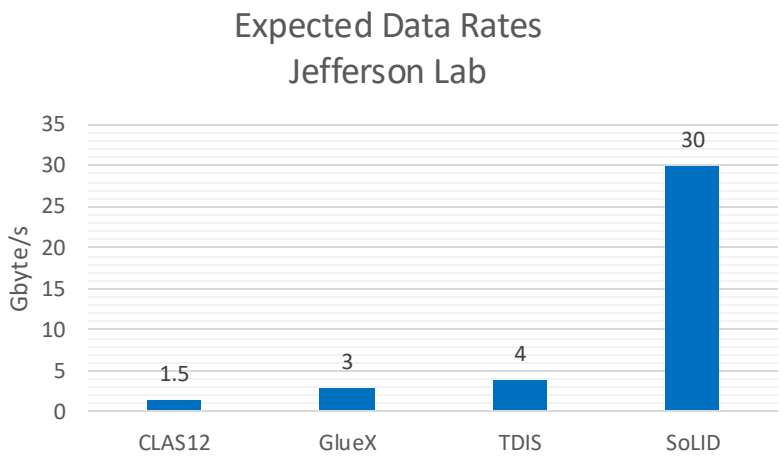
V. Gyurjyan
gurjyan@jlab.org

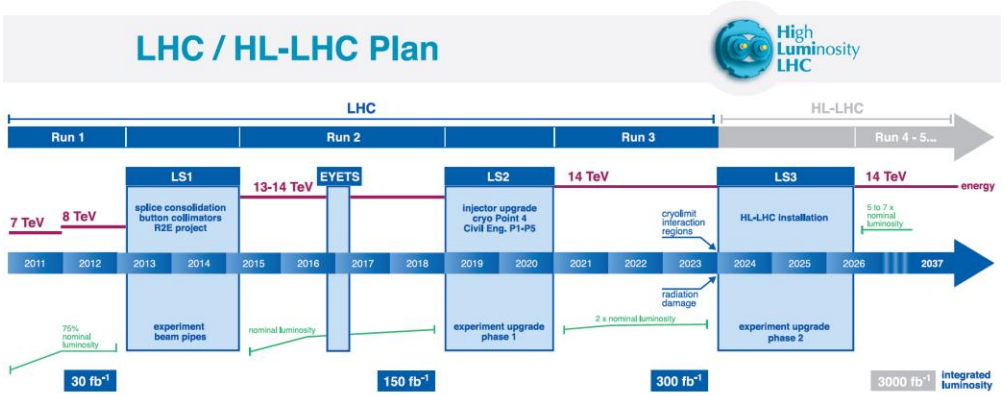
Jefferson Lab

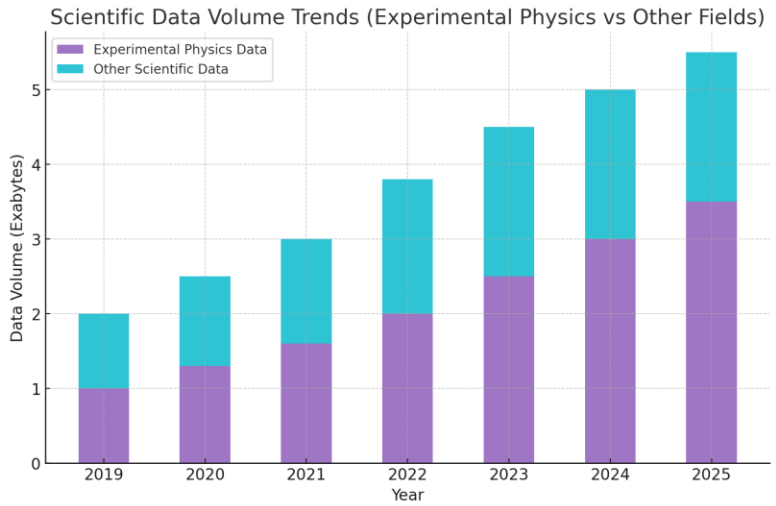
U.S. DEPARTMENT OF ENERGY | Office of Science

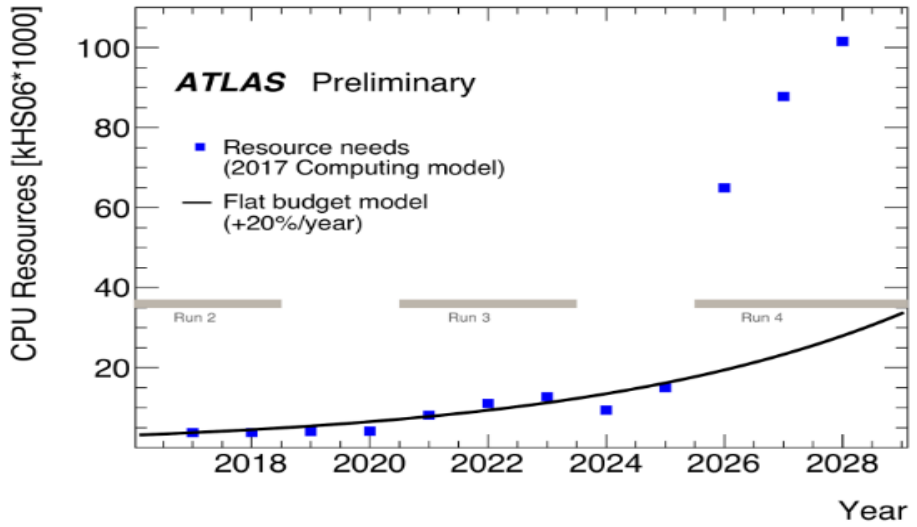
JSA

Scientific Data Expansion









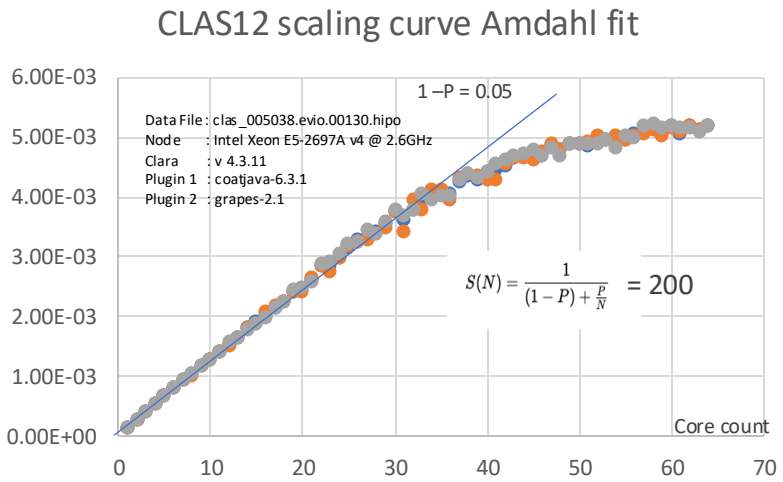
Vertical and horizontal scaling and it's limitations

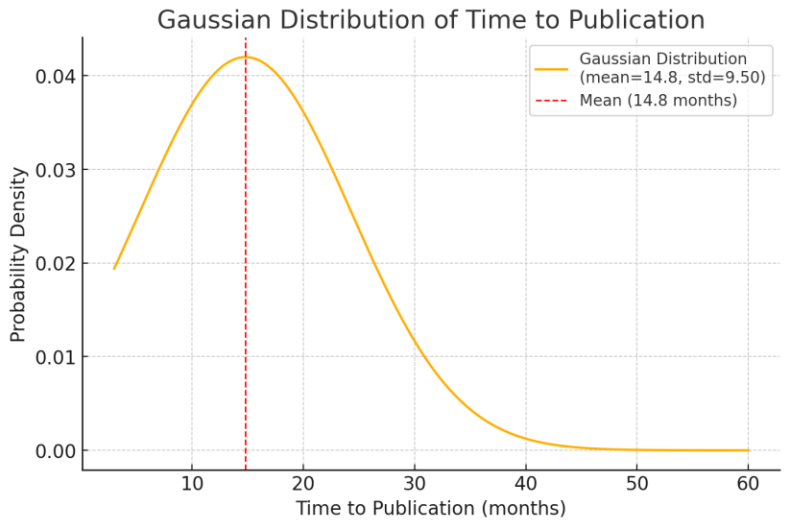
Vertical scaling

- Some portion of the code remains serial
- Memory contention, leading to a plateau in performance.
- Concurrent accesses to CPU caches, increasing cache misses. If threads rely on disk or network I/O, and context switching.

Horizontal (X) scaling

- Batch processing
  - Issues:
    - Limited local resources
    - Require data migration and temporal persistency (IO latency)

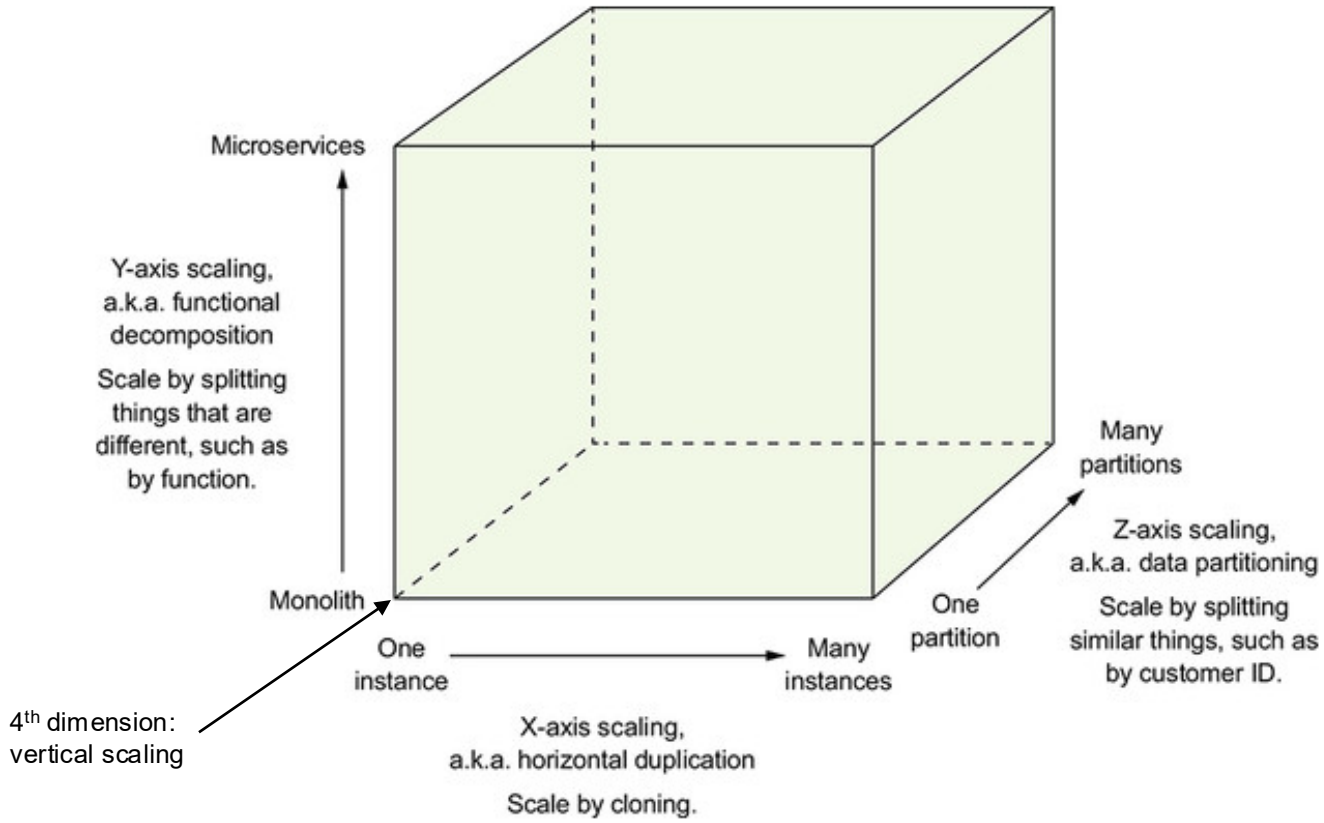




Can we do better?

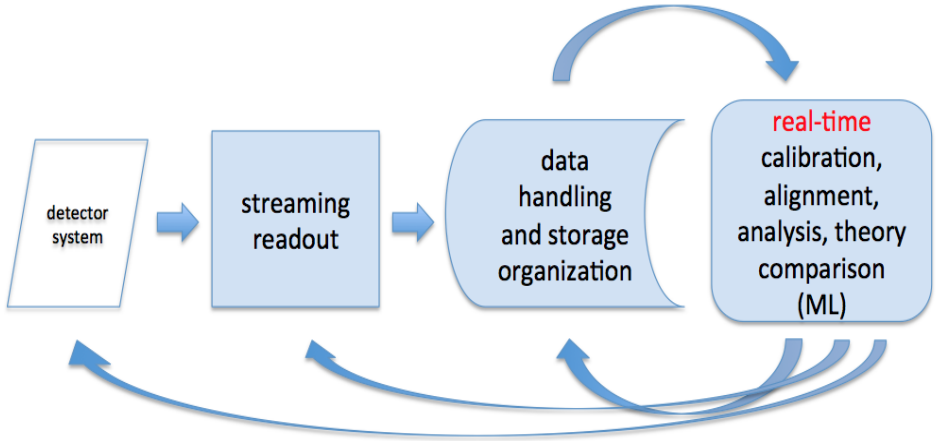
Four dimensional scaling

- Scaling cube plus vertical scaling
- From climate simulations, genomics to HEP/NP, scalable HPC ensures researchers stay ahead of the global data expansion.

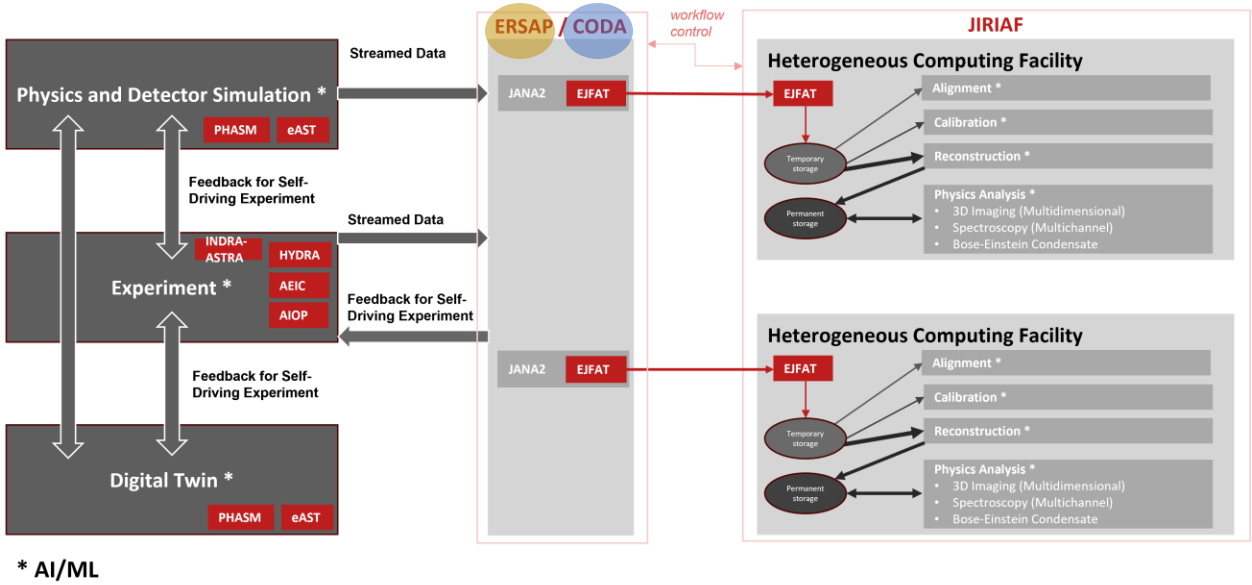


The Art of Scalability. by Martin L. Abbott and Michael T. Fisher. **ISBN-13:** 978-0134032801

JLAB Grand Challenge in Readout and Analysis for Femtoscale Science



Courtesy of Amber Boehnlein, et al.



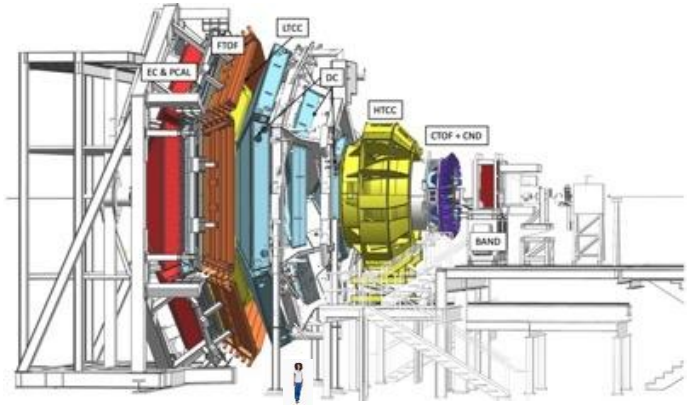
Courtesy of David Lawrence

"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."

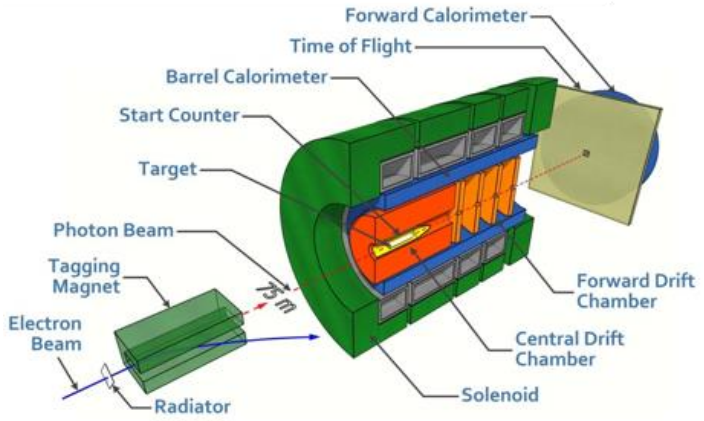
A Roadmap for HEP Software and Computing R&D for the 2020s. HEP Software Foundation, Feb. 2018

JLAB Data Processing

CLAS12 Detector

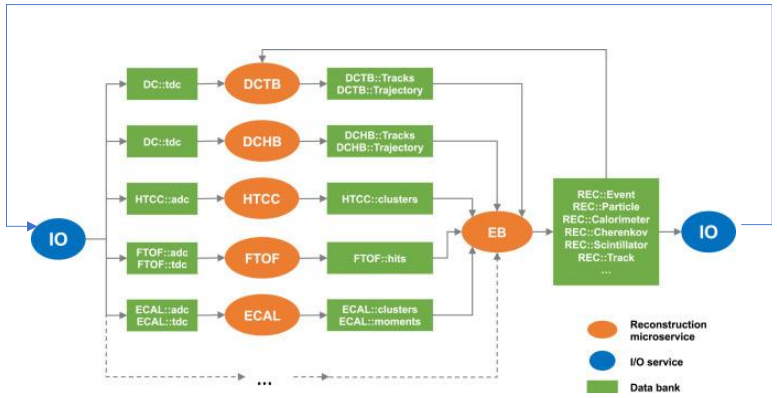


GlueX Detector

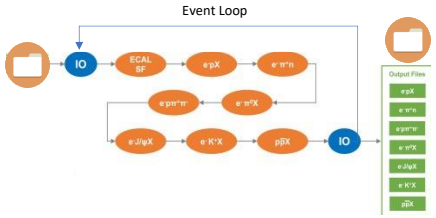


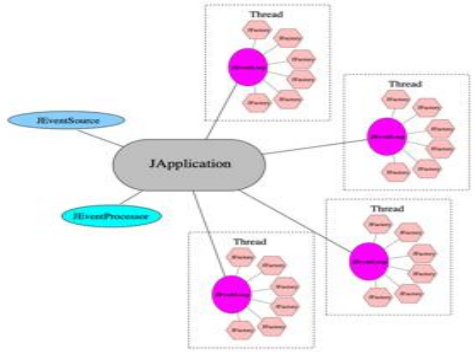
All existing data processing applications are deployed as monolith.

Event Loop



Event Loop

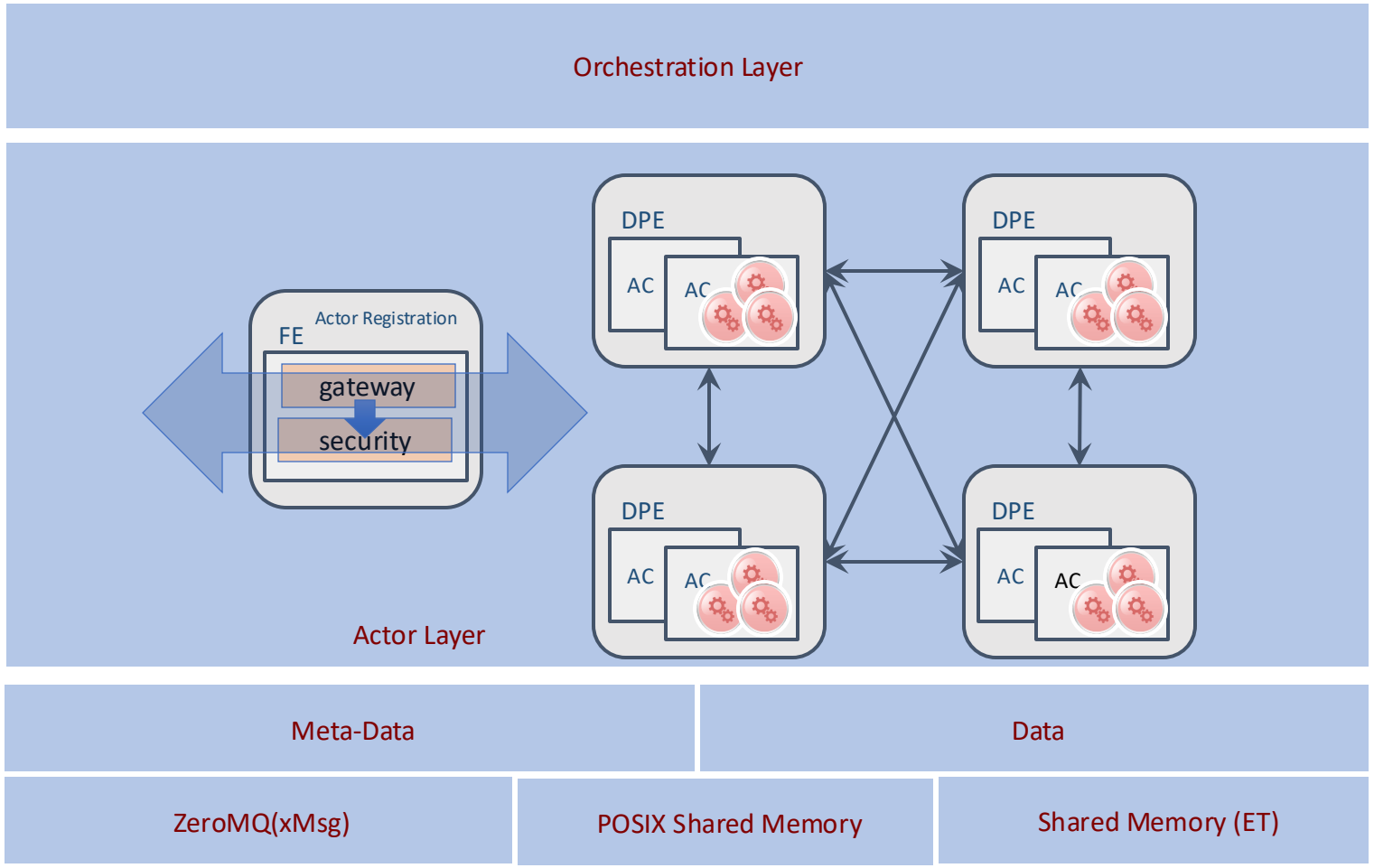




ERSAP Uses Flow-Based Programming Paradigm

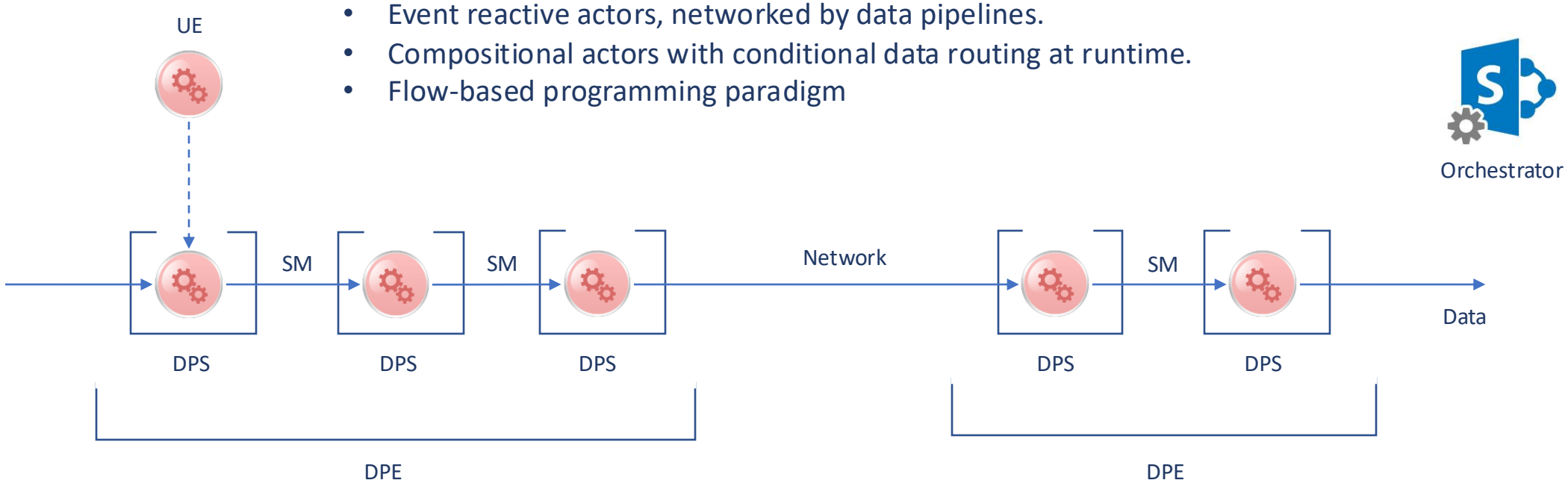
- Proposed in the late 60s by J. Paul Rodker Morrison
- “Assembly line” data processing
- Data flows through asynchronous, concurrent processors (“black box” actors)
- Actors communicate via data chunks (called information packets or data-quanta)
- Data-quanta are traveling across predefined connections (conveyor belts), where connections are specified externally to the processors.
- Data is pushed through actors, while actors are reacting on passing data quantum.
- Actors are performing independent, well-defined functions
- Simple reconfigure
- Fault tolerant

ERSAP 3-layer structure



ERSAP framework architecture

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

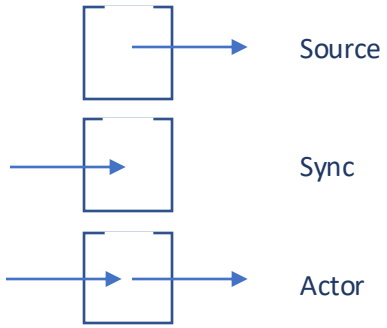


DPE : Data Processing Environment

SM : Shared Memory

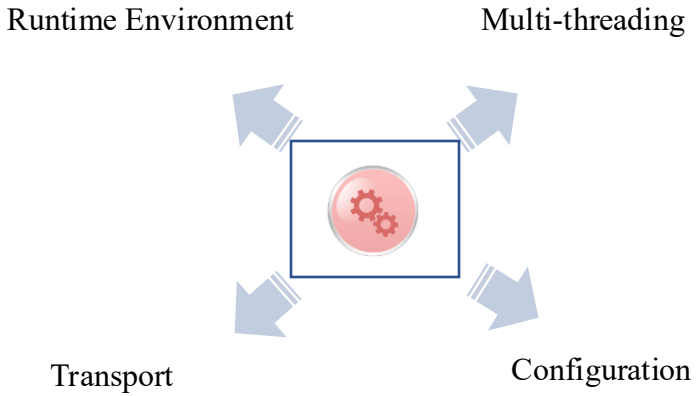
UE : User Engine

DPS : Data Processing Station



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)

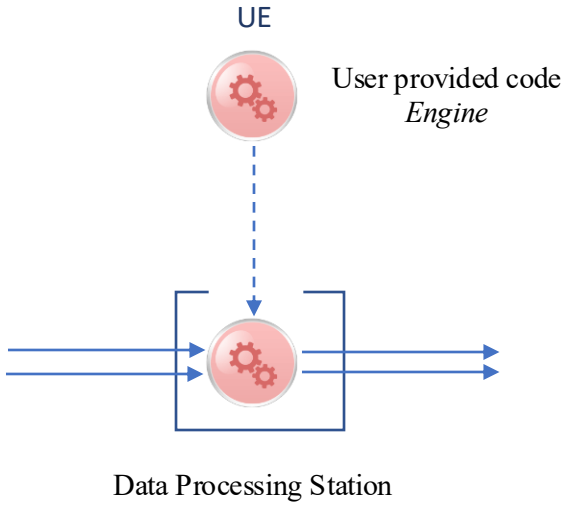
Object *process*(Object O)

Object *process*(Object[] O)

Object *process*(Map<String, Object> O)

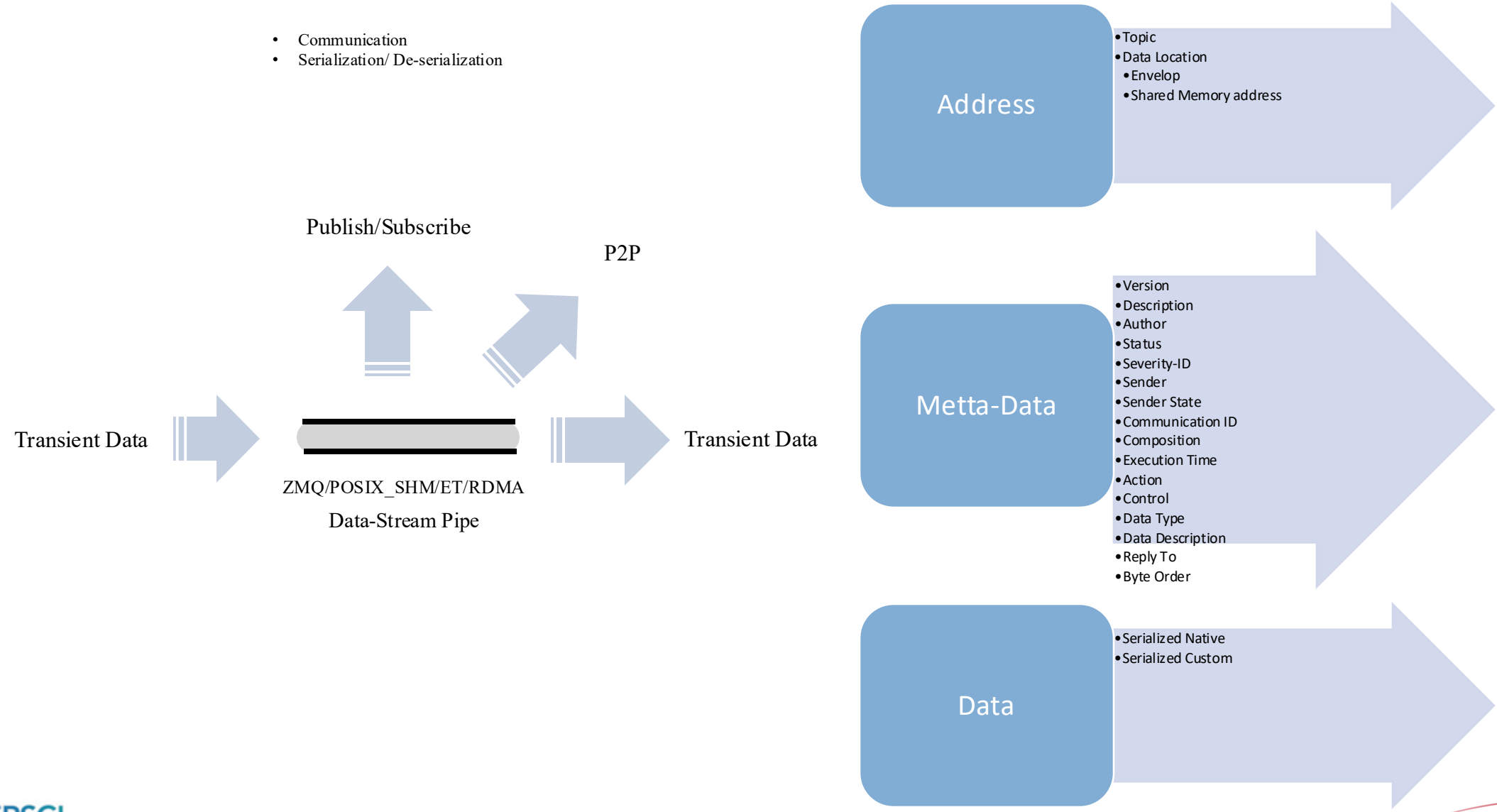
Object[] *process*(Object[] O)

Map<String, Object> *process*(Map<String, Object> O)

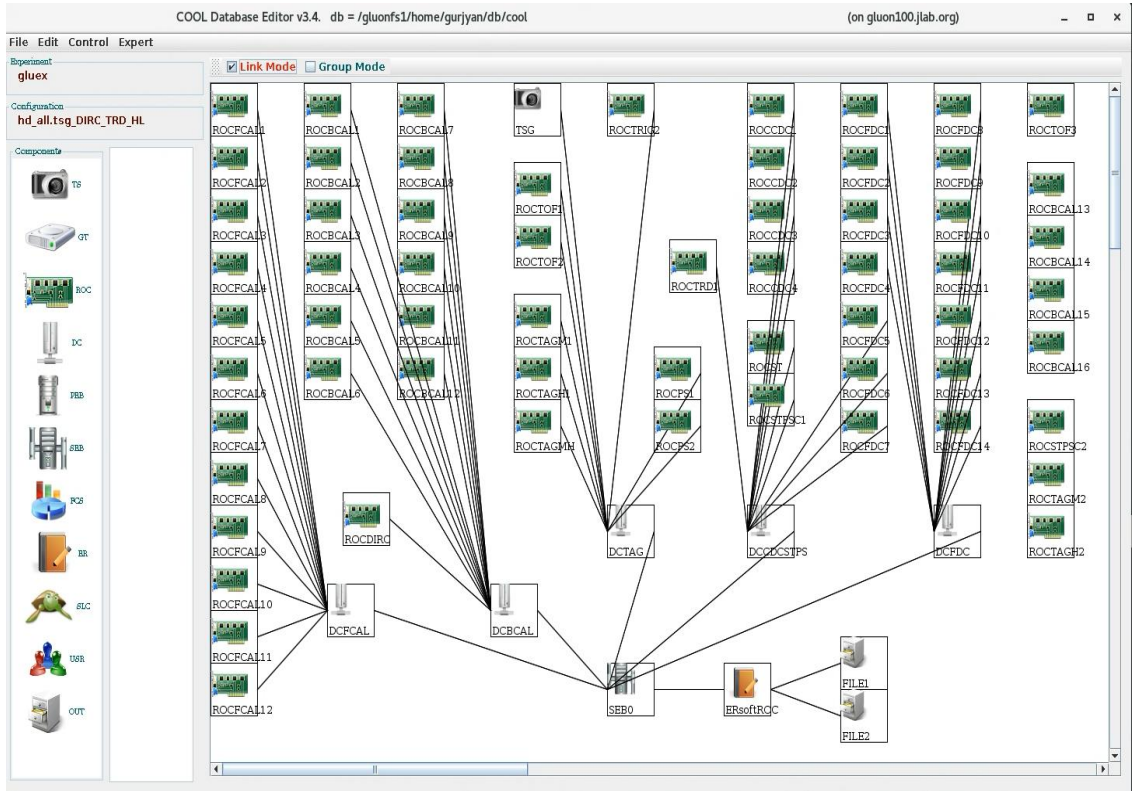


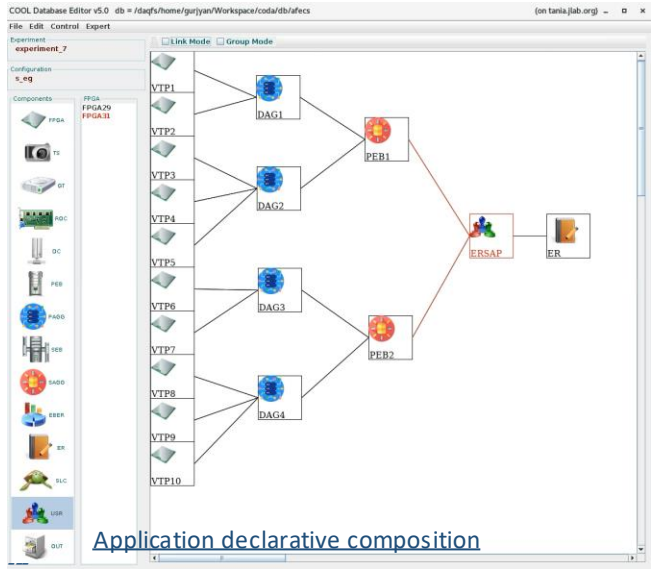
Streaming data transport

- Communication
- Serialization/ De-serialization



Data Acquisition and Processing Pipeline Designer





Application declarative composition

```
io-services:
  reader:
    class: org.jlab.ersap.demo.services.ImageReaderService
    name: ImageReaderService
  writer:
    class: org.jlab.ersap.demo.services.ImageWriterService
    name: ImageWriterService

services:
  - class: org.jlab.ersap.demo.services.FaceDetectorService
    name: FaceDetectorService
  - class: pupil_detector_service
    name: PupilDetectorService
    lang: cpp

mime-types:
  - binary/img-file
```

EJFAT streams terabit-rate data from instruments to ASCR computing facilities over continental-scale distances.

Site B
FABRIC Nodes
3 Nodes
FABRIC

Site A
FABRIC Nodes
3 Nodes
FABRIC

Control and Performance

JIRIAF API Server
Control and Monitoring
Data Source
EJFAT Testbed
Jefferson Lab

Raw Data

Perlmutter nodes
80 nodes
NERSC

ESnet Load Balancer 1

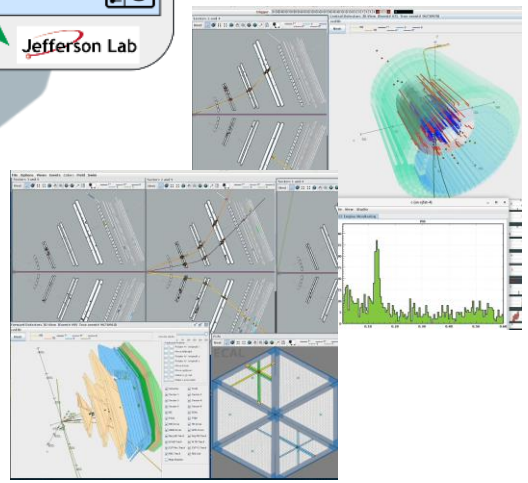
ESnet Load Balancer 2

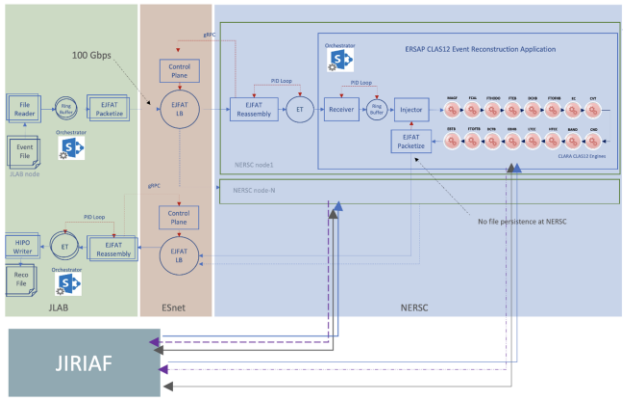
Defiant nodes
4 nodes
OAK RIDGE National Laboratory

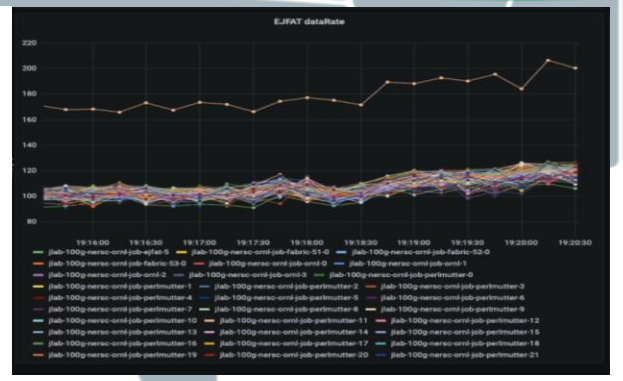
Processed Data

CLAS12 detector streaming readout

Processed event stream return for real-time visualization, physics validation, and persistence

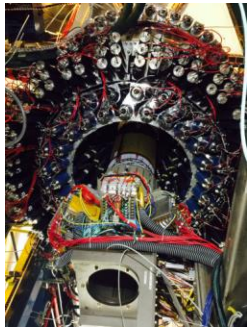


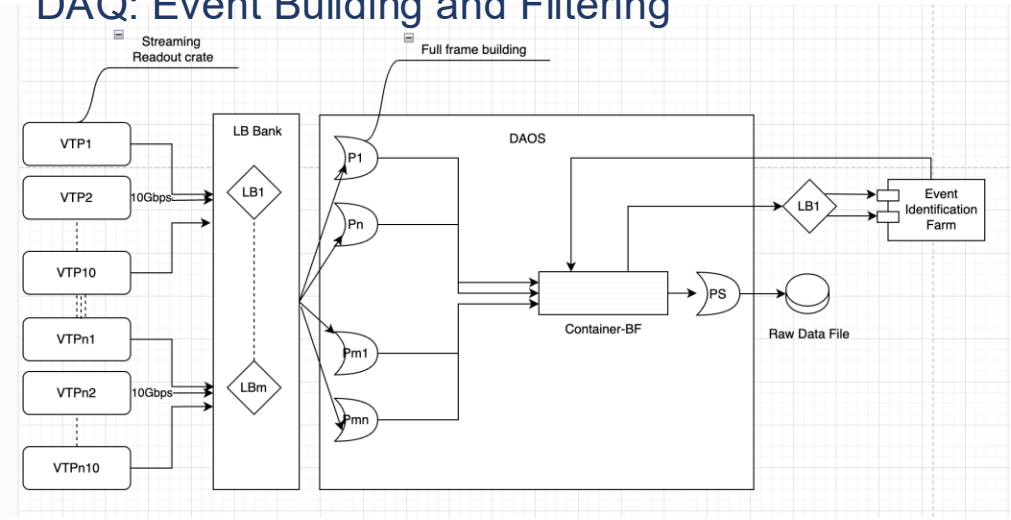




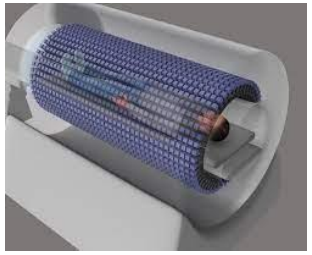
Projects using ERSAP

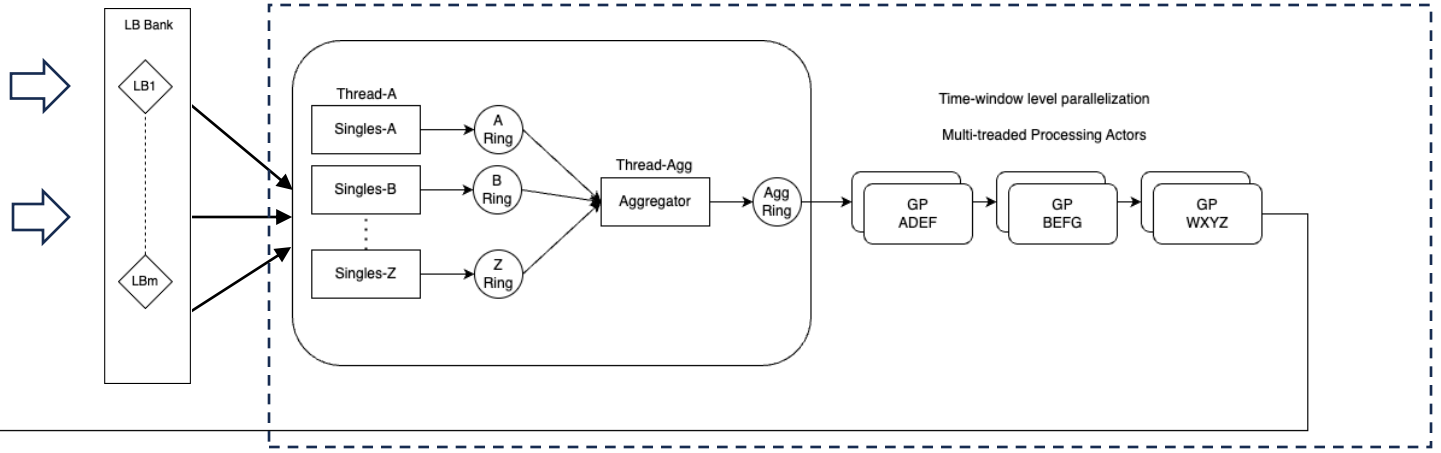
DAQ: Event Building and Filtering





Total Body PET Scanner





JLAB LDRD 2509 ID:000001.10.LD.029

Summary

- Event-driven reactive actor-based framework
- Design real time as well as offline data-stream processing applications
- Agile framework that makes easy software evolution over time
- Proof concept application designs for streaming readout and data-stream processing.
- First time, a fully remote, distributed data stream processing workflow has been successfully demonstrated using production-level physics data across DOE computing facilities using ERSAP.
- Many new pipelines in development.

ERSAP: Towards Better NP Data-Stream Analytics with Flow-Based Programming
V. Gyurjyan, D. Abbott, N. Brei, M. Goodrich, G. Heyes, E. Jastrzembski, D. Lawrence, B. Raydo, C. Timmer
Published at: IEEE Xplore
DOI: https://doi.org/10.1109/TNS.2023.3242548

ERSAP C++ Binding

ERSAP Java Binding

Thank You

ERSAP Native Data Formats

```
// ... Data class ...
message xMsgData {

   optional sint32 VLSINT32 = 1;    // variable length signed int32
   optional sint64 VLSINT64 = 2;    // variable length signed int64
   optional sfixed32 FLSINT32 = 3; // fixed length signed int32
   optional sfixed64 FLSINT64 = 4; // fixed length signed int64
   optional float FLOAT = 5;
   optional double DOUBLE = 6;
   optional string STRING = 7;       // contains UTF-8 encoding or 7-bit ASCII text
   optional bytes BYTES = 8;         // contains arbitrary sequence of bytes

   repeated sint32 VLSINT32A = 9;      // array of variable length signed int32s
   repeated sint64 VLSINT64A = 10;     // array of variable length signed int64s
   repeated sfixed32 FLSINT32A = 11; // array of fixed length signed int32s
   repeated sfixed64 FLSINT64A = 12; // array of fixed length signed int64s
   repeated float FLOATA = 13;           // array of floats
   repeated double DOUBLEA = 14;    // array of doubles
   repeated string STRINGA = 15; // array of UTF-8 encoded or 7-bit ASCII strings
   repeated bytes BYTESA = 16;    // array of arbitrary sequence of bytes
}

//... Payload class ...
message xMsgPayload {

   message Item {
      required string name =1; // payload name
      required xMsgData data = 2; // data
   }
   repeated Item item = 1;
}
```

CEBAF Online Data Acquisition and Processing System

- 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.