CLARA’s adaptive workflow management system

Reactive micro-services based data processing orchestration.

V. Gyurjyan
gurjyan@jlab.org
Will address

• Heterogeneous data-processing optimization with CLARA's adaptive workload orchestration

• NUMA-aware workflow management system
Outline

- Problem statement
- Micro-services vs Monolithic architecture
- Flow-based programming paradigm
  - Passive vs Reactive programming
  - Event vs message driven communication
- CLARA reactive micro-services based data-stream processing framework.
- Framework level workflow orchestration
- Data-processing performance optimization across diverse hardware and software infrastructures.
JLAB CLAS12

- Thomas Jefferson National Accelerator Facility (TJNAF), commonly known as Jefferson Lab or JLab, is a U.S. national laboratory located in Newport News, Virginia.
- Superconducting RF technology based accelerator provides 12 GeV continuous electron beam with a bunch length of less than 1 picosecond.
- Nuclear physics experiments in 4 end-stations (A,B,C,D)
- CLAS12 is a large acceptance spectrometer installed in Hall B to study
  - Quark-gluon interactions with nuclei
  - Nucleon-nucleon correlations
  - Nucleon quark structure imaging,
  - etc.
Problem we face

Expected Data Rates

Jefferson Lab

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CLAS12</td>
<td>1.5</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>GlueX</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>TDIS</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>SoLID</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
</tbody>
</table>

Global Digital Data

Exabyte

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

ATLAS Preliminary

CPU Resources [kHS06^10000]

Year

Run 2

Run 3

Run 4

Resource needs (2017 Computing model)

Flat budget model (+20%/year)
CPU based architecture limitations

Memory Latency

Today's computers now take much longer to fetch or store than to add and multiply.

MIPS/clock speed plateau
Squeezing more cores per chip becomes difficult

von Neumann Bottleneck

Central Processing Unit
- Control Unit
- Arithmetic/Logic Unit
- Memory Unit

Input Device → Central Processing Unit → Output Device

FPGA Performance (GMACs)

CPU Performance (GFLOPs)
“Frameworks face the challenge of handling the massive parallelism and heterogeneity that will be present in future computing facilities, including multi-core and many-core systems, GPUs, Tensor Processing Units (TPUs), and tiered memory systems, each integrated with storage and high-speed network interconnections.”

“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
The Scale-Cube

Scale by splitting similar things, such as events

Vertical scaling or multi-threading

Scale by cloning things, such as multiprocessing

Horizontal scaling

Scale by splitting different things, such as by function

Micro-services vs Monolithic architecture

**Pros**
- Strong coupling, thus better performance
- Full control of your application

**Cons**
- No agility for isolating, compartmentalizing and decoupling data processing functionalities, suitable to run on diverse hardware/software infrastructures
- No agility for rapid development or scalability

**Pros**
- Technology independent
- Fast iterations
- Small teams
- Fault isolation
- Scalable

**Cons**
- Complexity networking (distributed system)
- Requires administration and real-time orchestration
What is micro about a service?
Passive vs Reactive

- **S1**: Proactive, responsible for change in S2
- **S2**: Passive, unaware of the dependency

Passive programming

- **S1**: Broadcasts its own result
- **S2**: Subscribes S1 change events and changes itself

Reactive programming

Enables event driven stream processing

Publisher/Producer

Subscriber/Consumer

Feedback to control backpressure
Event-Driven vs Message-Driven

Event Driven

Message Driven

S1 event broadcasting

S1 message has a clear destination
CLARA Framework

Reactive, event-driven data-stream processing framework that implements micro-services architecture and FBP

- Defines streaming transient-data structure
- Provides service abstraction (data processing station) to present user algorithm (engine) as an independent service.
- Defines service communication channel (data-stream pipe) outside of the user engine.
- Stream-unit level workflow management system and API
- Supports C++, JAVA, Python languages

http://claraweb.jlab.org
https://claraweb.jlab.org/clara/docs/clas/hands-on.html
Basic components and a user code interface

Data Processing Station | Data-Stream Pipe | Orchestrator

Data processing Engine | Data Processing Station | Data Processing Micro-Service

Engine Tutorials
- https://claraweb.jlab.org/clara/docs/quickstart/java.html
- https://claraweb.jlab.org/clara/docs/quickstart/cpp.html
- https://claraweb.jlab.org/clara/docs/quickstart/python.html
Data Processing Station

Runtime Environment

Multi-threading

Communication

Data Processing Station

Configuration

Language Bindings

- https://github.com/JeffersonLab/clara-java.git
- https://github.com/JeffersonLab/clara-cpp.git
- https://github.com/JeffersonLab/clara-python.git
Data Stream Pipe

Communication

Language Bindings

- https://github.com/JeffersonLab/xmsg-java.git
- https://github.com/JeffersonLab/xmsg-cpp.git
- https://github.com/JeffersonLab/xmsg-python.git

Data Stream Pipe

Meta-description, Serialization, De-serialization

Transient data

OMG /POSIX_SHM/Data-Grid
Structure

Orchestration Layer

Service Layer

<table>
<thead>
<tr>
<th>Meta-Data</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZeroMQ(xMsg)</td>
<td>POSIX Shared Memory (FIPC)</td>
</tr>
<tr>
<td>in-Memory Data-Grid</td>
<td></td>
</tr>
</tbody>
</table>
Streaming Data-Flow Processing

In-Memory Data-Grid

Detector-1
Detector-2

Node-1

In-Process SHM

DPE-1

Node-2

In-Process SHM

DPE-2

POSIX Shared Memory

DPE-3

Meta-data

1
2
3

4
5
6

7

In-Memory Data-Grid

Node-1

In-Process SHM

DPE-1

Node-2

In-Process SHM

DPE-2

POSIX Shared Memory

DPE-3
Transient data unit (meta-data + data)

- Topic
- Message-Location
  - Envelope
  - Shared-Memory Key
- xMsgMeta
  - Version
  - Description
  - Author
  - Status
  - Severity-ID
  - Sender
  - Sender-State
  - Communication-ID
  - Composition
  - Execution-Time
  - Action
  - Control
  - Data-Type
  - Data-Description
  - Reply-To
  - Byte-Order
- xMsgData-Object
- Byte-Array

Exception Reporting
- Topic
- Message-Location
  - Envelope
  - Shared-Memory Key
- xMsgMeta
  - Version
  - Description
  - Author
  - Status
  - Severity-ID
  - Sender
  - Sender-State
  - Communication-ID
  - Composition
  - Execution-Time
  - Action
  - Control
  - Data-Type
  - Data-Description
  - Reply-To
  - Byte-Order
- xMsgData-Object
- Byte-Array
CLAS12 Data Processing Applications

Event Reconstruction Application

Data Acquisition System

Data Quality Assurance

Data Provisioning

Physics Analysis Application
Event Reconstruction Application (sub-event level parallelization)
Workflow orchestrator

Application Monitoring, Real-time Benchmarking

Application Deployment and Execution

Exception Logging and Reporting

Command-Line Interface

Hardware Optimizations

Service Registration/Discovery

Data-Set Handling and Distribution

Farm (batch or cloud) Interface
Heterogeneous deployment algorithm

\[ P_g = \frac{\sum_{i=1}^{t_i} CR_{FTOF}(t_i)}{\sum_{i=1}^{t_i} CR_{DCHB\_GPU}(t_i)} \]

\[ P_c = \frac{\sum_{i=1}^{t_i} CR_{FTOF}(t_i)}{\sum_{i=1}^{t_i} CR_{DCHB\_CPU}(t_i)} \]

*if* \( P_g < P_c \)* route data-stream through DCHBg*
Data-quantum size and GPU occupancy
Thread motion and DVFS

- Per-core, independent voltage control becomes impractical
- Limited number of independent DVFS systems for multicore systems
- Large core density systems are deploying a new power management technique that migrates threads from core to core to adjust power and performance to the time-varying needs of a running program.
Data-processing chain per NUMA

Start DPE pinned to a NUMA socket.

Farm Node

NUMA 0

In-Process SHM

NUMA 1

In-Process SHM

Java-DPE

Back pressure control
Results

Rate vs. Threads for a Single NUMA Socket
CLAS12 Reconstruction Application: v. 5.9.0, Data File: clas_004013.hipo, NUMA 0

- AMD EPYC 7502: 1.5MHz, 128/128, NUMA-2
- Xeon E-2687A: 2.6GHz, 32/32, NUMA-2
- Xeon Gold 6148: 2.4GHz, 40/40, NUMA-4

CLAS12 Reconstruction Application Vertical Scaling

Data File: clas_005038.evio.00130.hipo
Node: Intel Xeon E5-2697A v4 @ 2.6GHz
Clara: v 4.3.11
Plugin 1: coatjava-6.3.1
Plugin 2: grapes-2.1

- P=0.995
- 99.5% parallel efficiency over physical cores

CLAS12 Reconstruction Application Virtual Scaling

Data File: clas_005038.evio.00130.hipo
Node: Intel Xeon E5-2697A v4 @ 2.6GHz
Clara: v 4.3.11
Plugin 1: coatjava-6.3.1
Plugin 2: grapes-2.1

- P=0.995
• Frameworks based on the micro-services architecture are in a better position to address massive parallelism and heterogeneity of current and future computing facilities.

• CLARA is a mature, micro-services based, data stream processing framework in production-use at JLAB and NASA.

• Internal, stream-unit level workflow management system is designed with adaptive functionalities that guarantees maximum data processing performance across diverse hardware and software infrastructures.