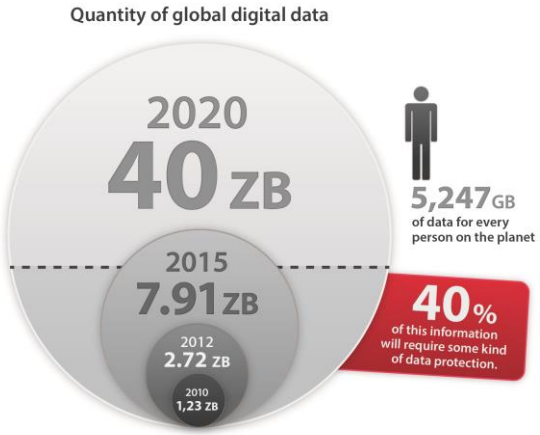
CLARA

CLAs12 Reconstruction and Analysis Framework

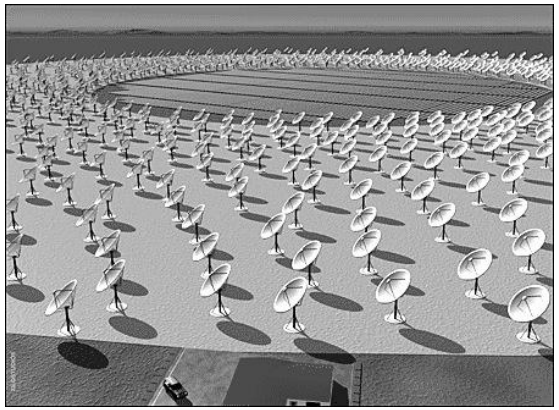
Vardan Gyurjyan (Jefferson Lab)
gurjyan@jlab.org

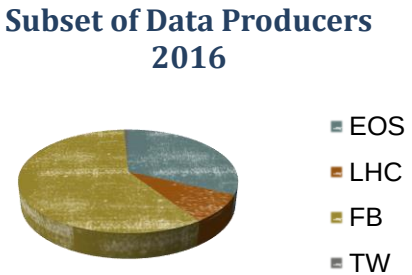
Global digital data demography

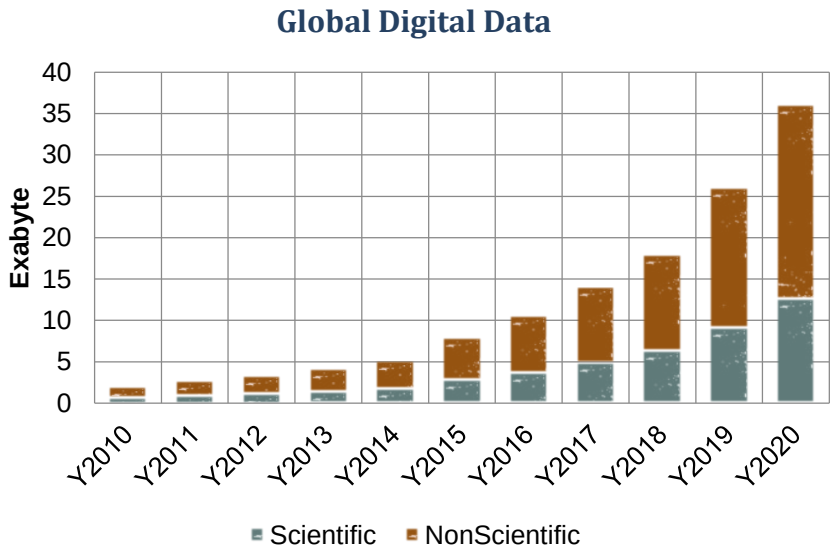


- Unprecedented 3V expansion of data
  - Volumes
  - Velocities
  - Varieties

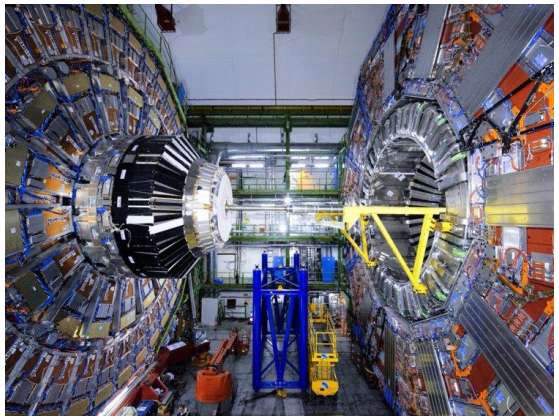
SKA Radio Telescope 700 TB/sec in 2020







ATLAS 5 EB in 2026



HL-LHC software R&D requirements

Jefferson Lab
Thomas Jefferson National Accelerator Facility

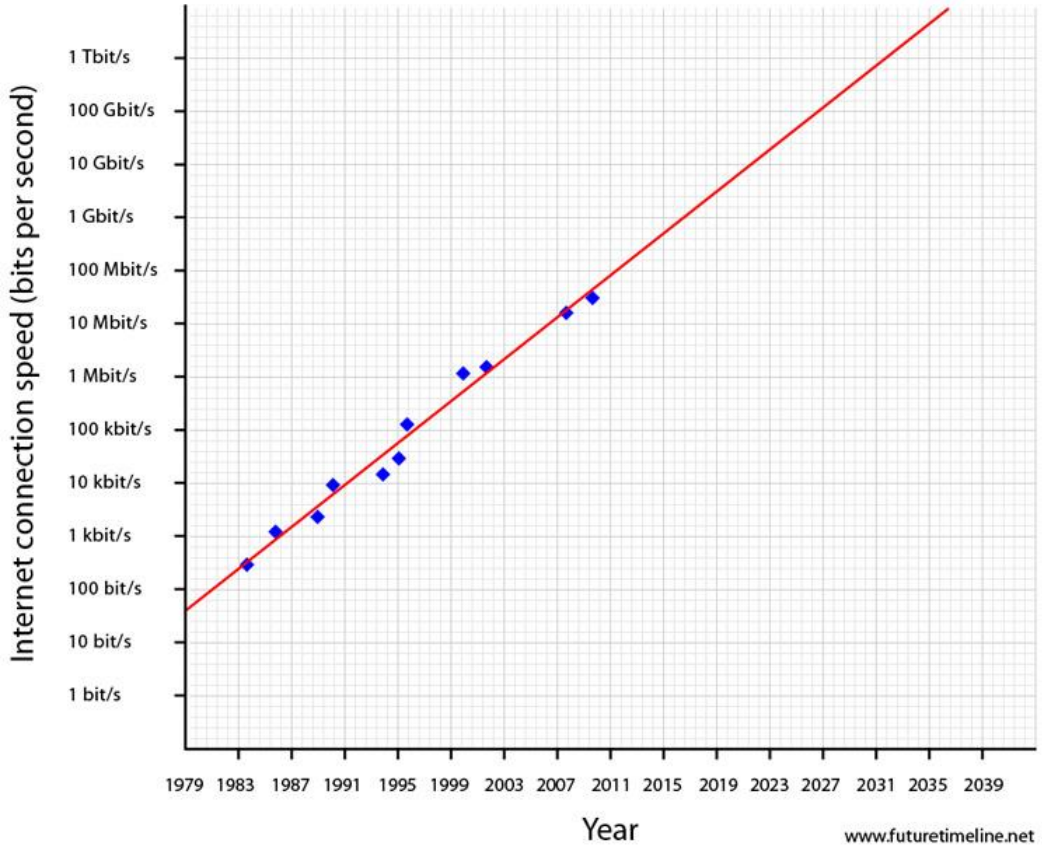
- Make use of advances in CPU, storage and network technologies
- Enable new approaches to computing and software
- Ensure the long-term sustainability of the software through the lifetime of the HL- LHC
- Attract the required new expertise

LHC software

- More than 20M lines of code (some of the components are 15 years old).
- Single architecture (x86_64)
- Self contained, monolithic
- Serial processing model
- Sustainability concerns
  - Original authors no longer in the field
  - Poor maintenance
  - Not well documented
  - Lacking unit tests

Future trends Moore's law is dead, long live Nielsen's law

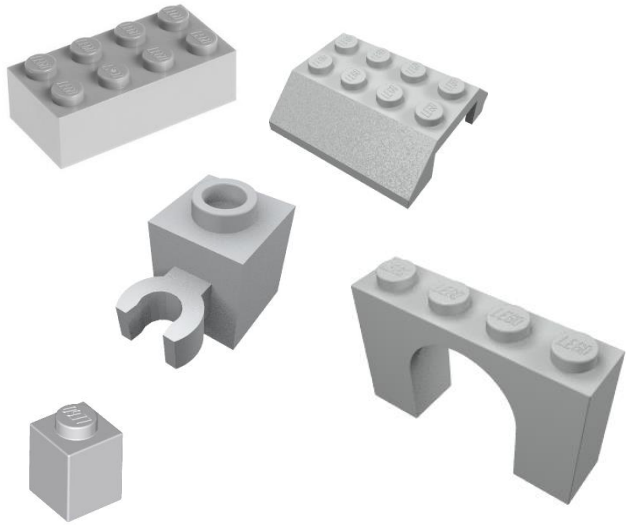
- Network bandwidth will continue to increase, but the ability to use it efficiently will need a closer integration with applications that supports:
  - Software distribution and distributed data and workflow management
  - Awareness of the extremely hierarchical view of data, from long latency tape access and medium-latency network access through to the CPU memory hierarchy.



Taking advantage of new architectures and programming paradigms will be critical for HEP to increase the ability of our code to deliver physics results efficiently, and to meet the processing challenges of the future.

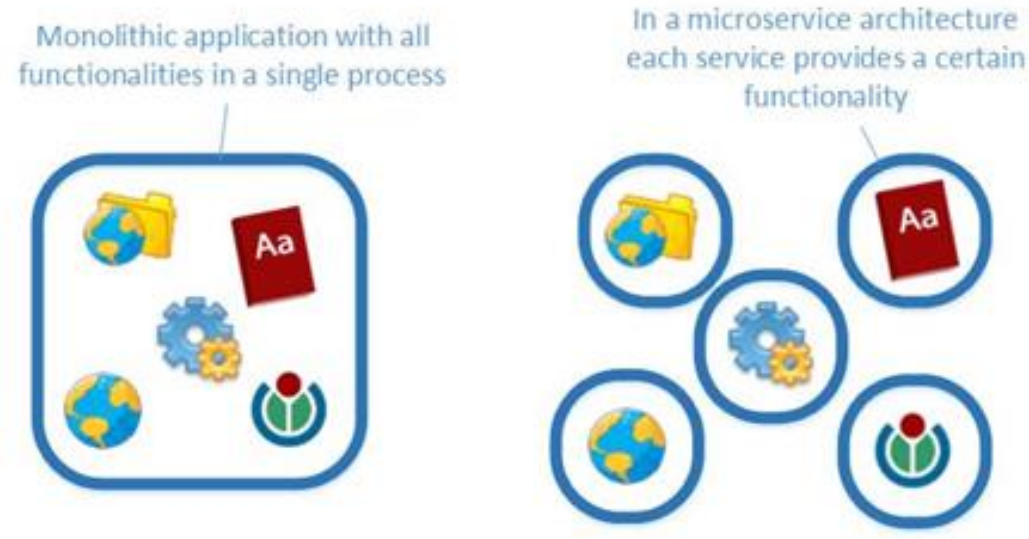
(A Roadmap for HEP Software and Computing R&D for the 2020s. arXiv:1712.06982)

Perfect software



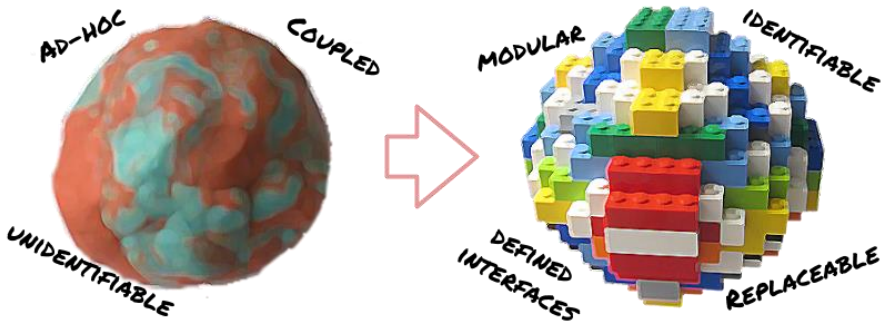
CS architectures to help

- Micro-services architecture
- Flow based reactive programming (FBP)



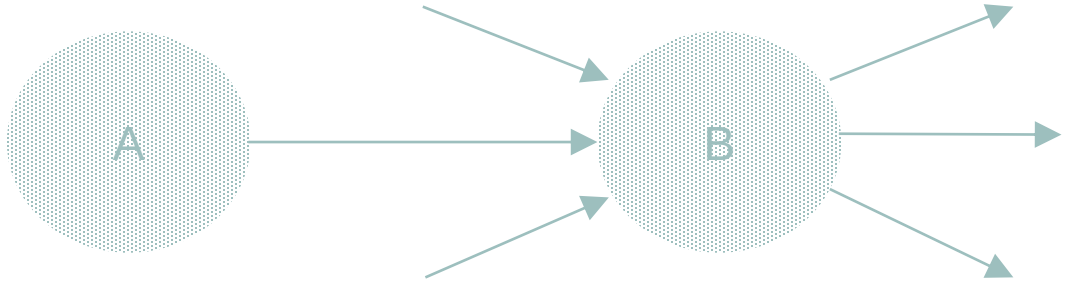
Micro-services: Advantages

- Application is made of components that communicate data
  - Small, simple and independent
  - Easier to understand and develop
  - Less dependencies
  - Faster to build and deploy
  - Reduced develop-deploy-debug cycle
  - Easy to migrate to data
  - Scales independently
  - Independent optimizations
  - Improves fault isolation
  - Eliminates long term commitment to a single technology stack.
  - Easy to embrace new technologies



Flow based programming (FBP)

- Glue between micro-services
- How they connect to achieve computational goals
- Makes application truly distributed

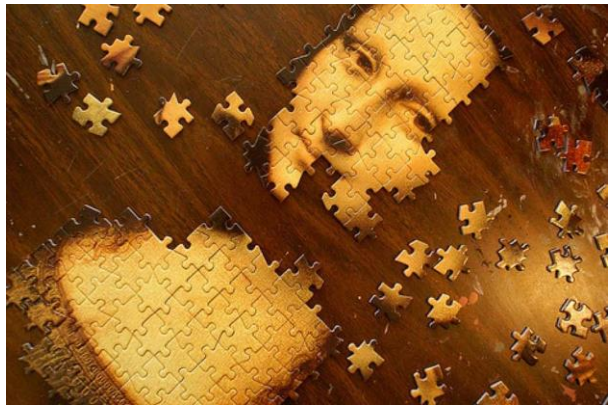


- B reacts on message
- Named inputs
- Sync/async message passing

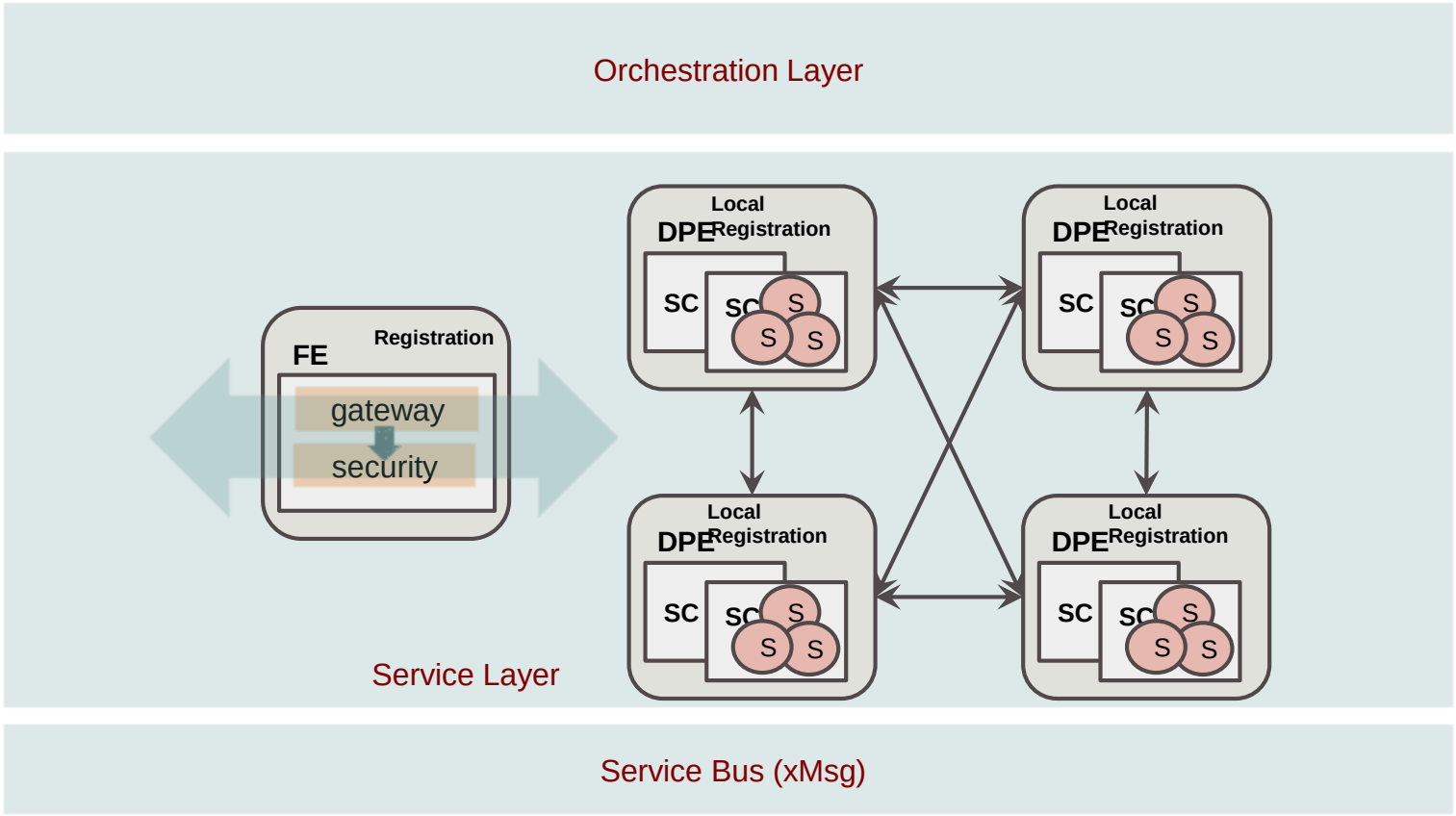
{ A calls B } v.s. { A message B }

CLARA implements micro-services and FBP

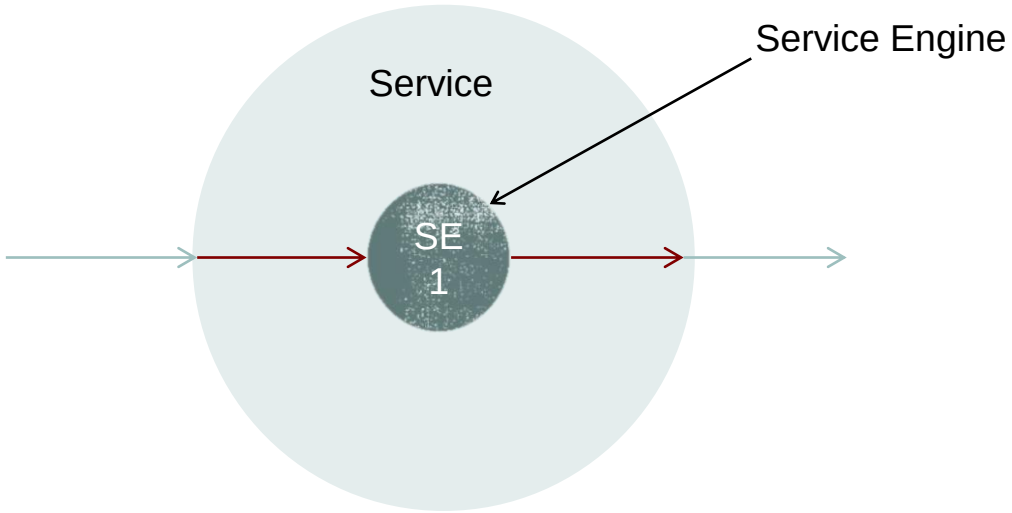
- Application is defined as a network of loosely coupled processes, called services.
  - Services exchange data across predefined connections by message passing, where connections are specified externally to the services.
  - Services can be requested from different data processing applications.
  - Loose coupling of services makes polyglot data access and processing solutions possible (C++, Java, Python, Fortran)
  - Services communicate with each other by exchanging the data quanta.
    - Thus, services share the same understanding of the transient data, hence the only coupling between services.



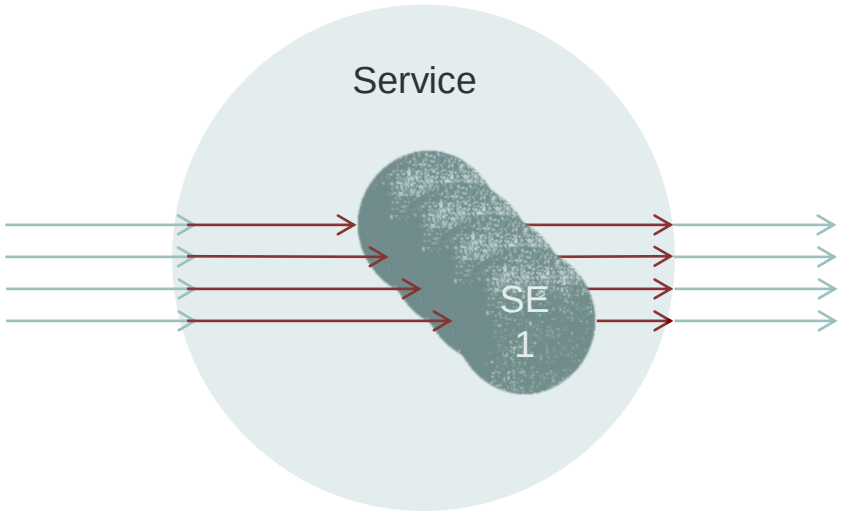
CLARA architecture



CLARA service

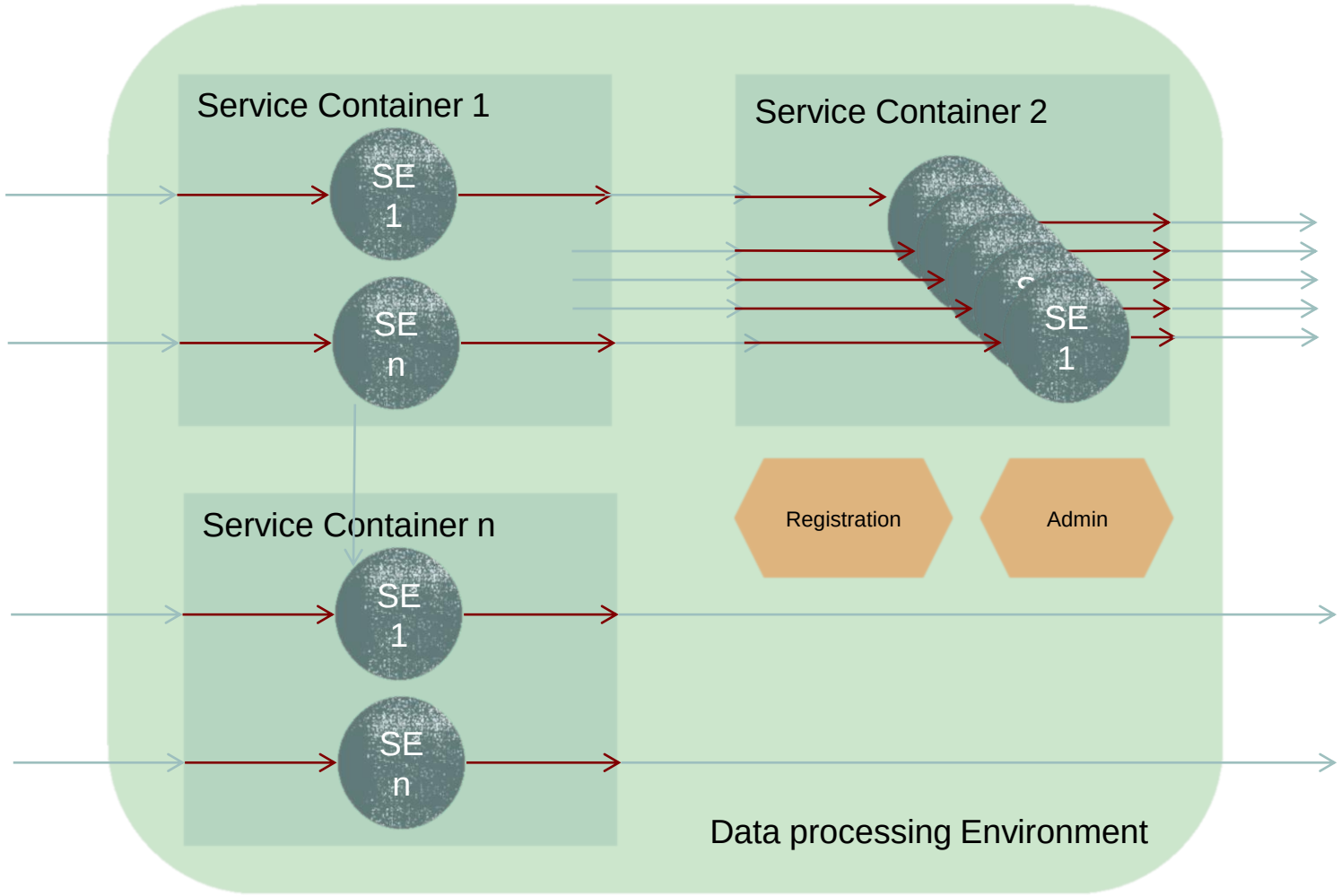


One request at a time



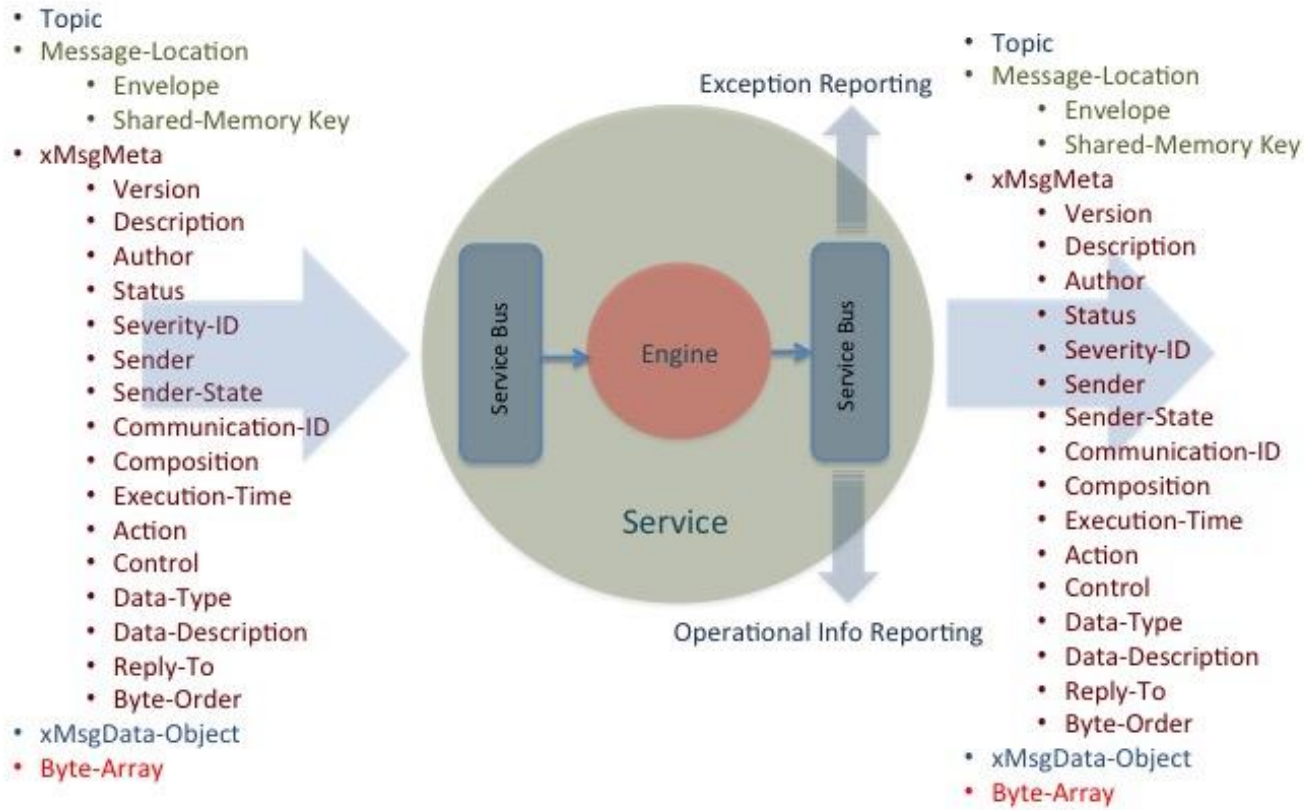
Multiple simultaneous requests

CLARA data processing environment

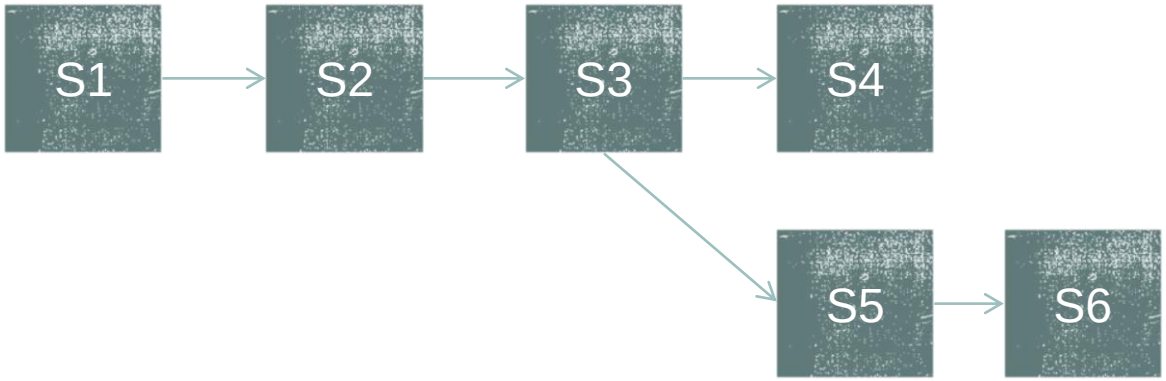


Data unit (event) stream processing

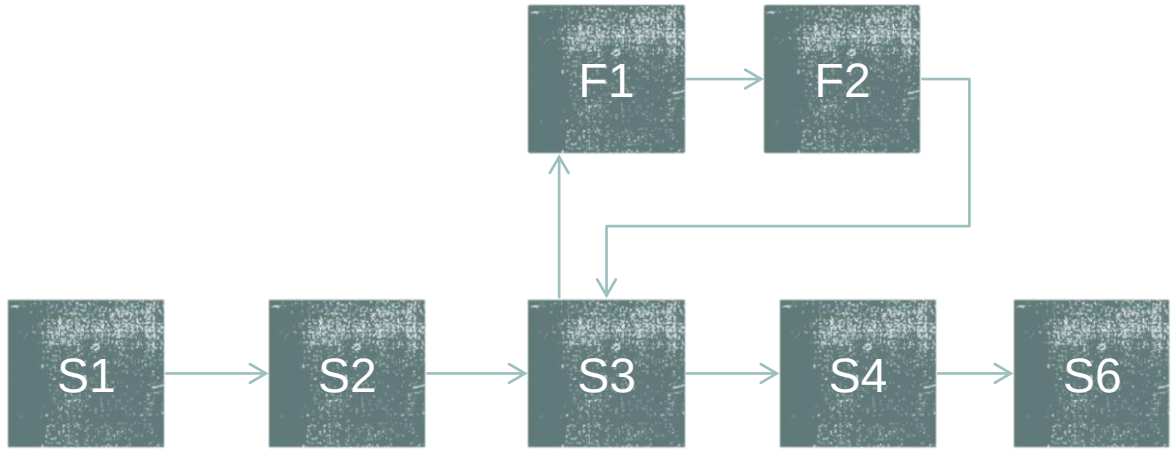
- Data driven, data centric design.
  - The focus is on transient data event modifications. Advantage over algorithm driven design is that a much greater ignorance of the data processing code is allowed (loose coupling).
- Design = service composition + event-routing.
  - Self routing (no routing scheduler)
- Event routing graph defines application algorithm



Algorithm examples

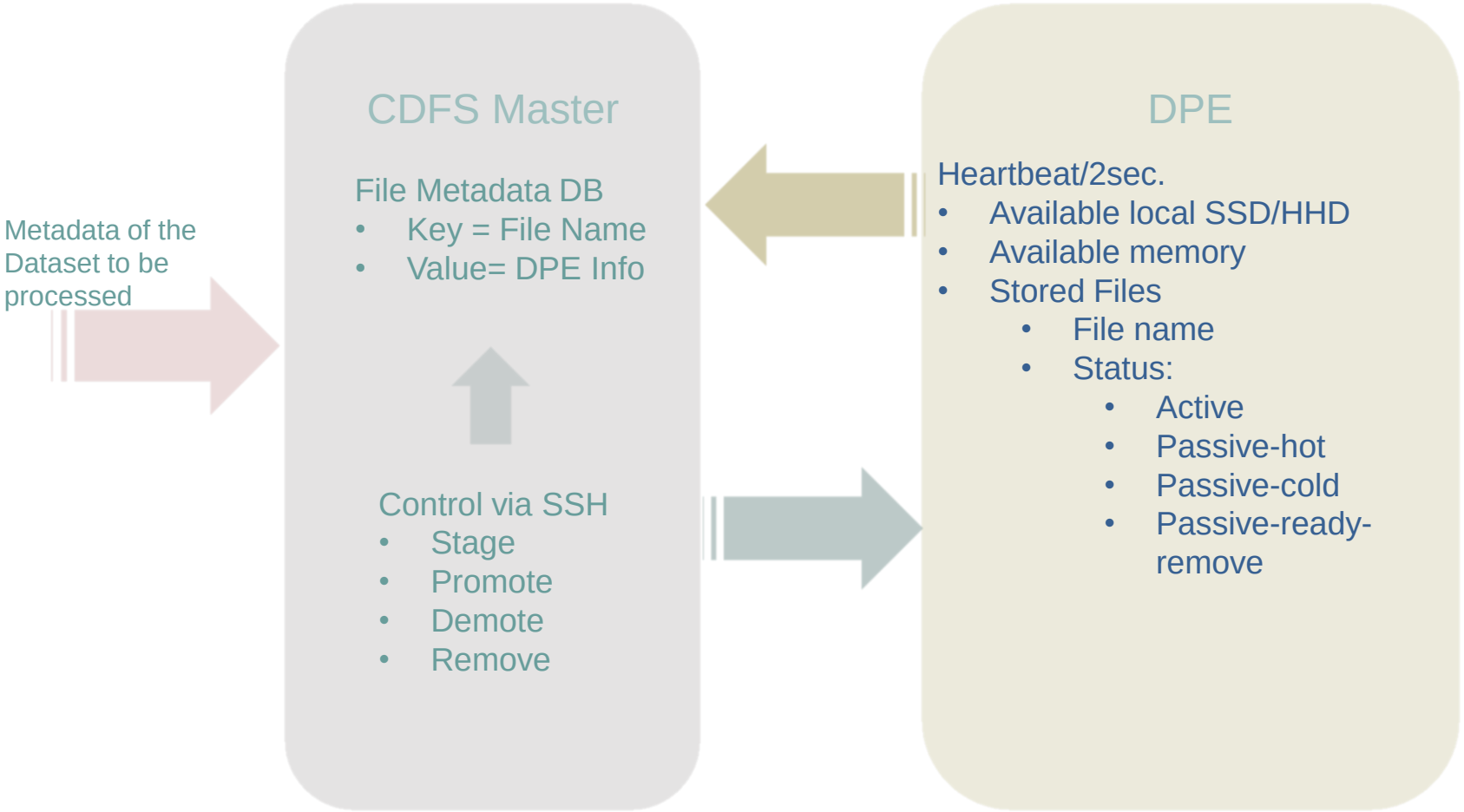



```
S1 + S2 + S3 + S4;
S3 + S5 + S6;
```




```
S1 + S2 + S3;
while( S3 == ”needs calibration") {
    F1 + F2 + S3;
}
   S3 + S4 + S6;
```

CLARA distributed file storage system

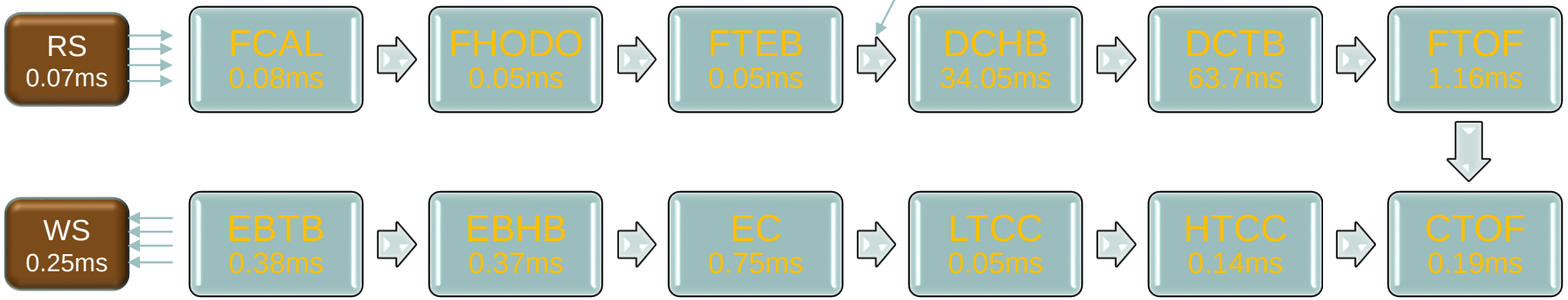


CLAS12 reconstruction

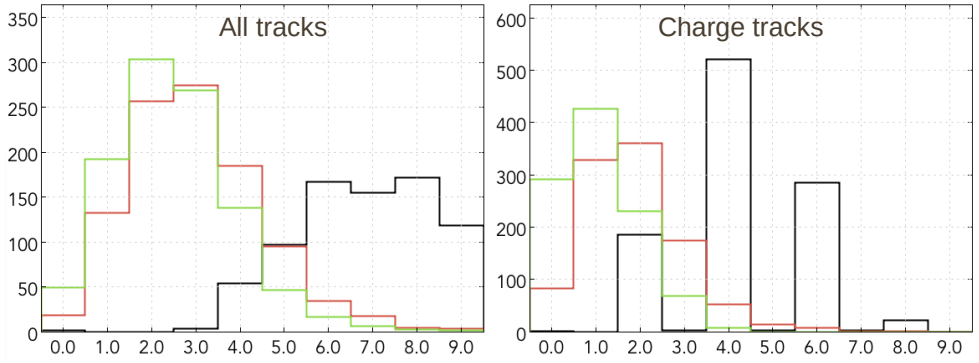
Jefferson Lab
Thomas Jefferson National Accelerator Facility

CLARA micro-service
Can be deployed as a separate process or a thread within a process. Multi-threaded

CLARA transient data-stream
Message passing through pub-sub middleware. No dependencies between micro-services.

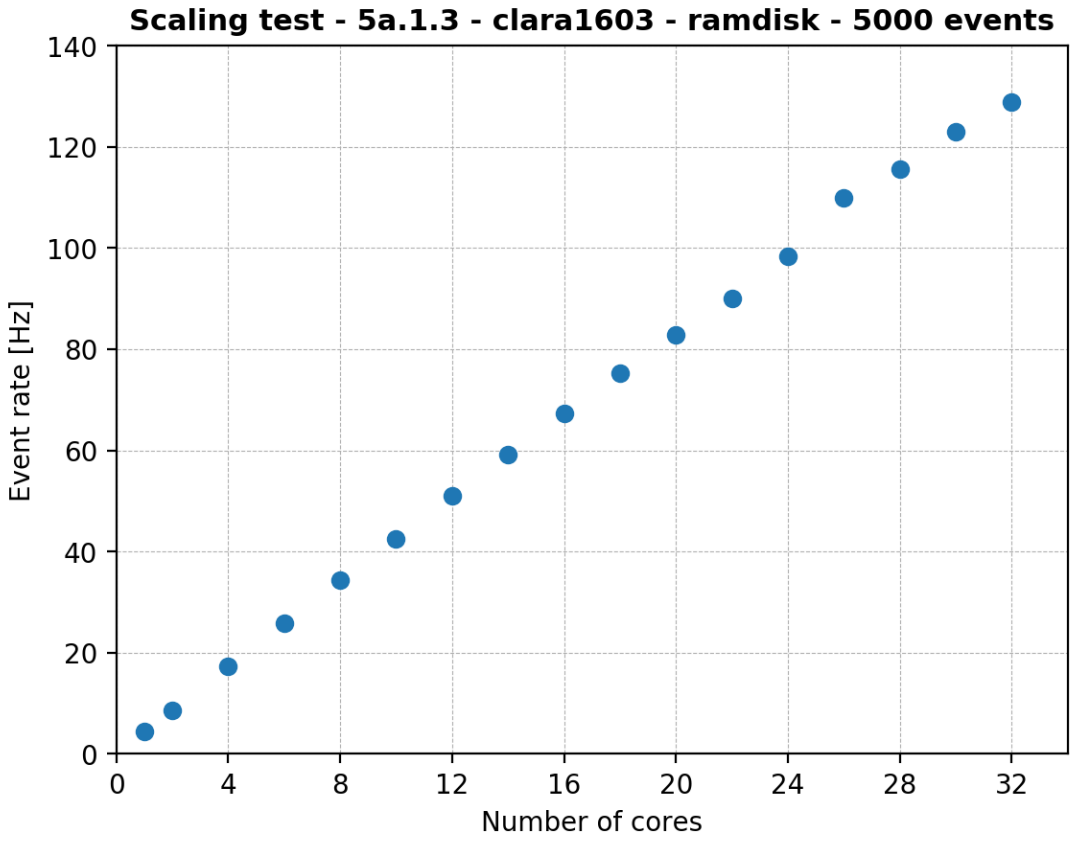


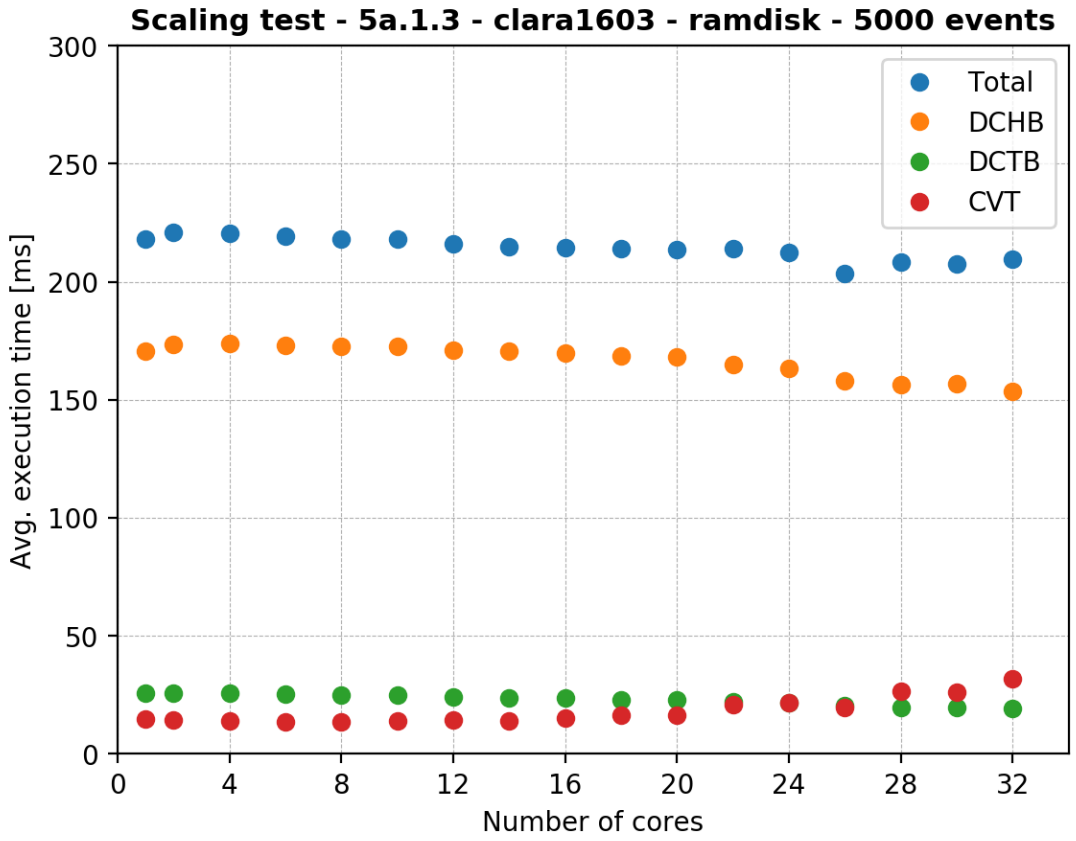
Total data processing latency per event, per core = 101.29ms
(Note: dependents on magnetic field swimming step size, number of track and background)
For simulated data file = sidisSkim10K.hipo
Trigger efficiency = 100%
Track multiplicity >=2
No background
Node = Intel(R) Xeon(R) CPU E5-2697A v4 @ 2.60GHz 2x16



Black generated, Red: reconstructed hit-based, Green: reconstructed time-based

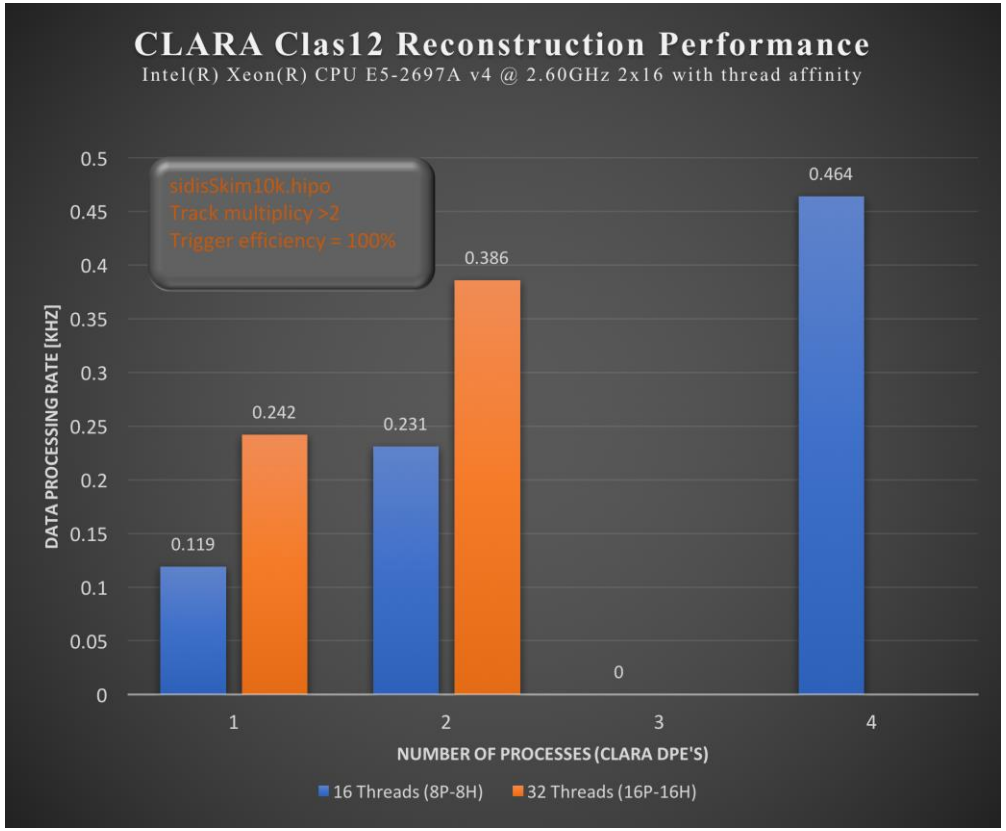
Vertical scaling





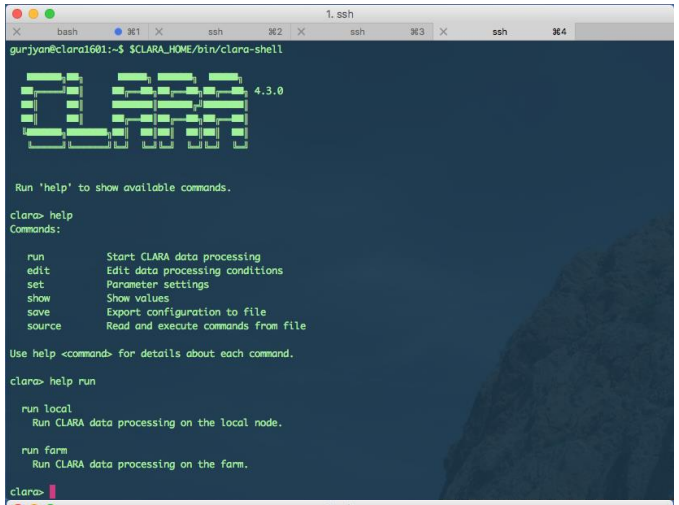
NUMA studies (Intel® Xeon® CPU E5-2697A V4 2.60GHz 2x16)

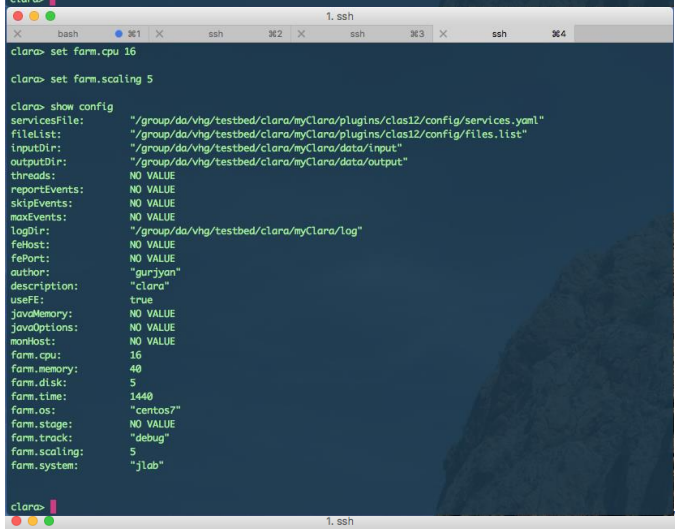
Measurement	1 Process		2 Processes		4 processes	
	8P-8H ms	16P-16H ms	8P-8H ms (N proc.)	16P-16H ms (N proc.)	8P-8H ms (N proc)	
1	8.50	4.10	8.13(1)	5.18(1)	8.51(1)	9.34(3)
			9.30(2)	5.22(2)	7.99(2)	9.20(4)
2	8.32	4.13	8.19(1)	5.16(1)	8.48(1)	8.90(3)
			9.31(2)	5.22(2)	8.05(2)	9.19(4)
3	8.33	4.16	8.14(1)	5.19(1)	8.55(1)	8.90(3)
			9.17(2)	5.10(2)	8.10(2)	8.42(4)
Aver. proc. 1	8.38	4.13	8.15	5.18	8.51	
Aver. proc. 2			9.26	5.18	8.06	
Aver. proc. 3					9.05	
Aver. proc. 4					8.88	
Rate in KHz	0.119	0.242	0.123+0.108=0.231	0.193+0.193=0.386	0.117+0.124+0.110+0.113=0.464	

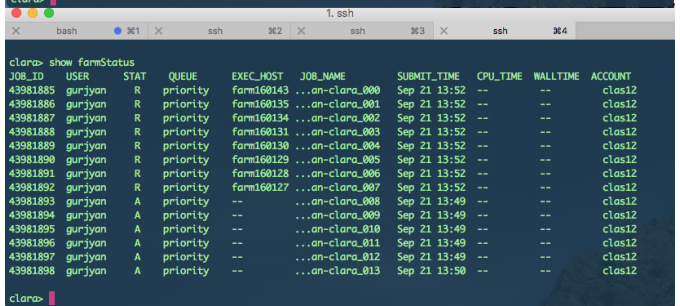


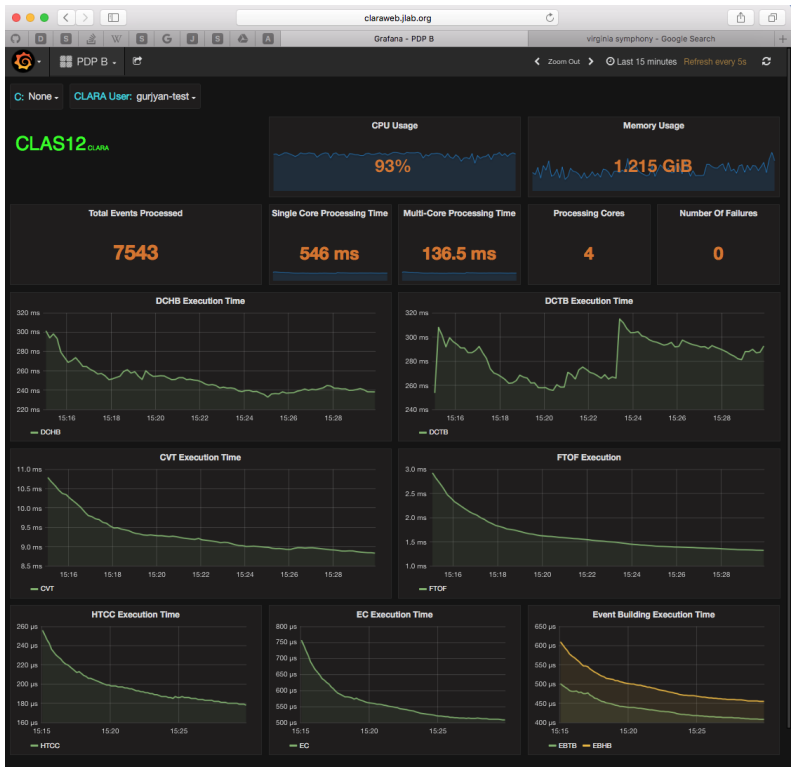
User experience

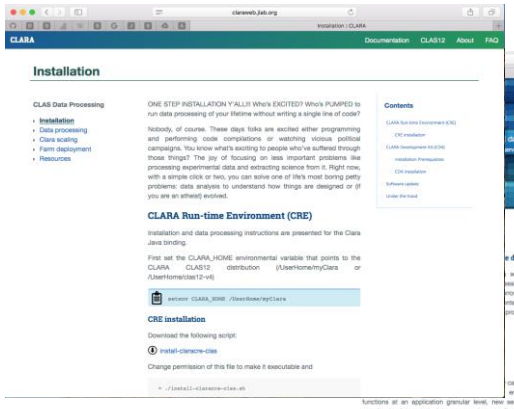
Jefferson Lab
Thomas Jefferson National Accelerator Facility

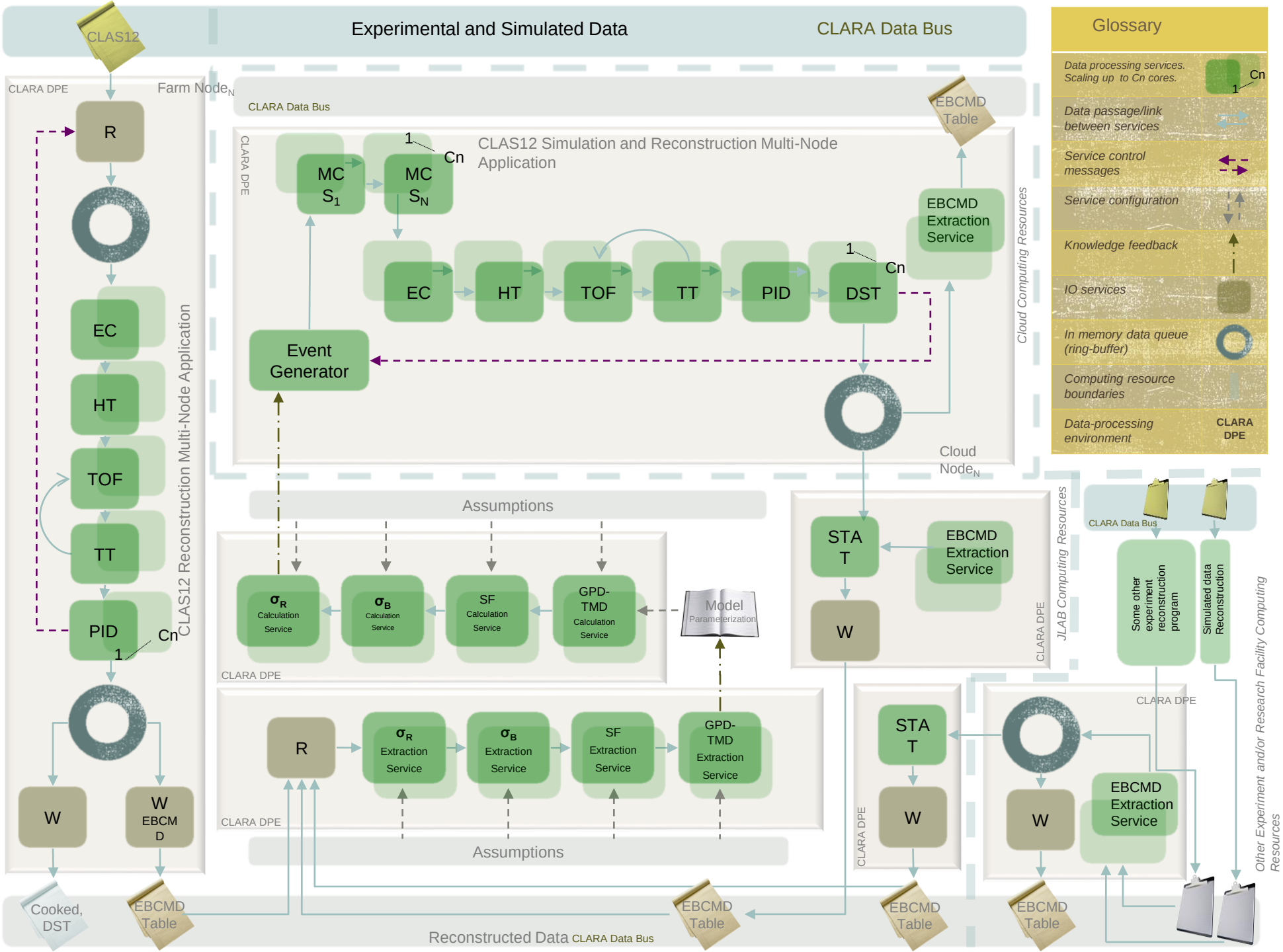








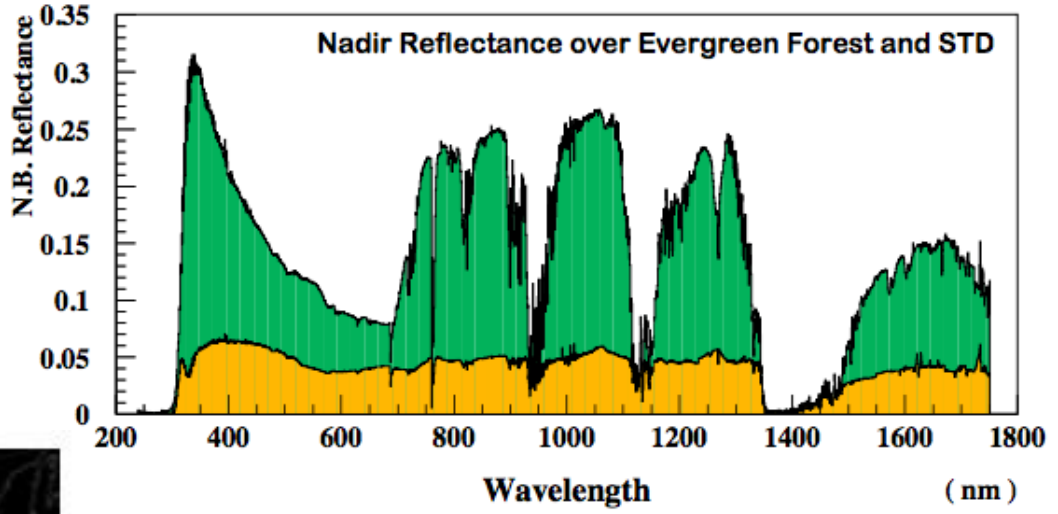


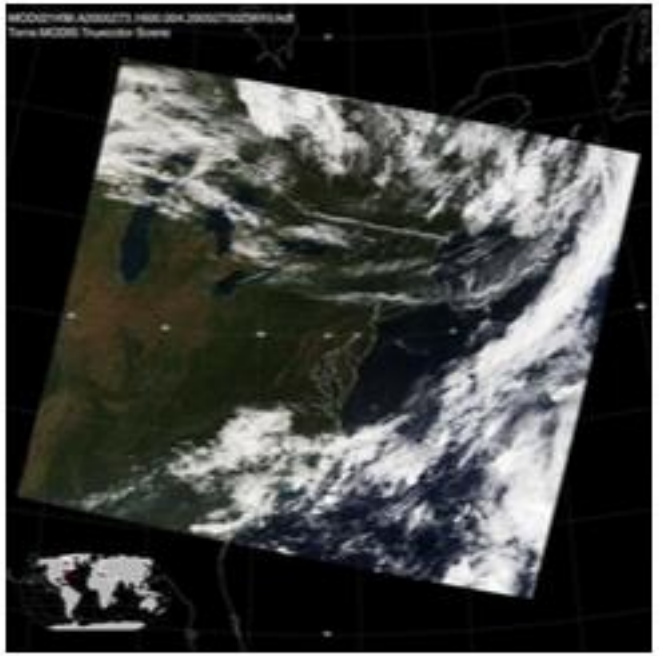

CLAS12
Experimental and Simulated Data
CLARA Data Bus
Glossary
Data processing services. Scaling up to Cn cores.
Data passage/link between services
Service control messages
Service configuration
Knowledge feedback
IO services
In memory data queue (ring-buffer)
Computing resource boundaries
Data-processing environment
CLARA DPE
CLARA DPE
Farm Node$_N$
CLAS12 Reconstruction Multi-Node Application
R
EC
HT
TOF
TT
PID
Cn
1
W
W EBCMD
CLARA Data Bus
CLAS12 Simulation and Reconstruction Multi-Node Application
MC S$_1$
MC S$_N$
EC
HT
TOF
TT
PID
DST
Event Generator
EBCMD Table
EBCMD Extraction Service
Cloud Computing Resources
Cloud Node$_N$
Assumptions
σ_R Calculation Service
σ_B Calculation Service
SF Calculation Service
GPD-TMD Calculation Service
Model Parameterization
R
σ_R Extraction Service
σ_B Extraction Service
SF Extraction Service
GPD-TMD Extraction Service
Assumptions
STAT
W
EBCMD Extraction Service
JLAB Computing Resources
CLARA Data Bus
Some other experiment reconstruction program
Simulated data Reconstruction
Other Experiment and/or Research Facility Computing Resources
Cooked, DST
EBCMD Table
Reconstructed Data CLARA Data Bus
Jefferson Lab
Thomas Jefferson National Accelerator Facility

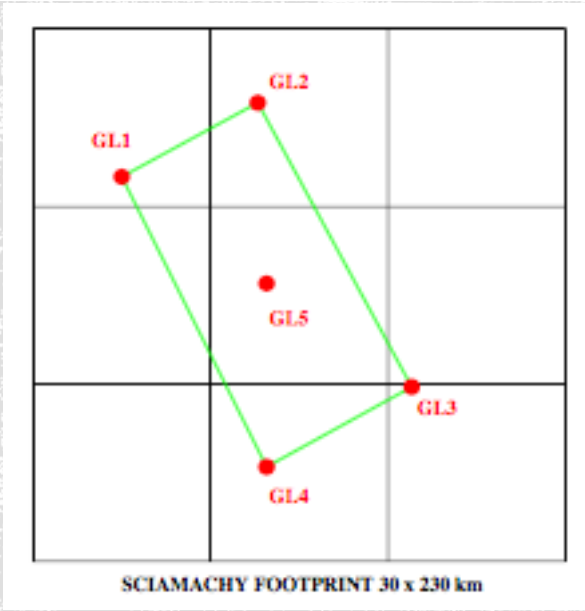
NASA EOS data fusion and processing

Jefferson Lab
Thomas Jefferson National Accelerator Facility

- OBJECTIVES:
  - To demonstrate NAIADS approach and full functionality using existing data;
    To benchmark NAIADS performance;
    Available data: 9 years of near-coincident measurements of from SCIAMACHY and MODIS;
    Create new fused SCIAMACHY/MODIS/ECMWF data product (requested by a number of projects).
- SCIAMACHY Level-1 Data:
  - Spectral measurement for every footprint: 30 km x 230 km; Swath 950 km (4 footprints) from 10 AM Sun-synch orbit.
- ECMWF Data (re-analysis):
  - Gridded (0.125o);6 weather parameters; Map every 6 hours;
- MODIS/Terra Level-2 Data:
  - Level-2 Cloud and Aerosol Data Spatial scale: 1 / 5 km and 10 km spatial; Swath 2300 km (global coverage daily); 10:30 AM Sun-synch orbit.







NAIADS deployment on AWS

Jefferson Lab
Thomas Jefferson National Accelerator Facility

AWS c4.8xlarge instances,
36 vCPUs ~= 18 physical cores

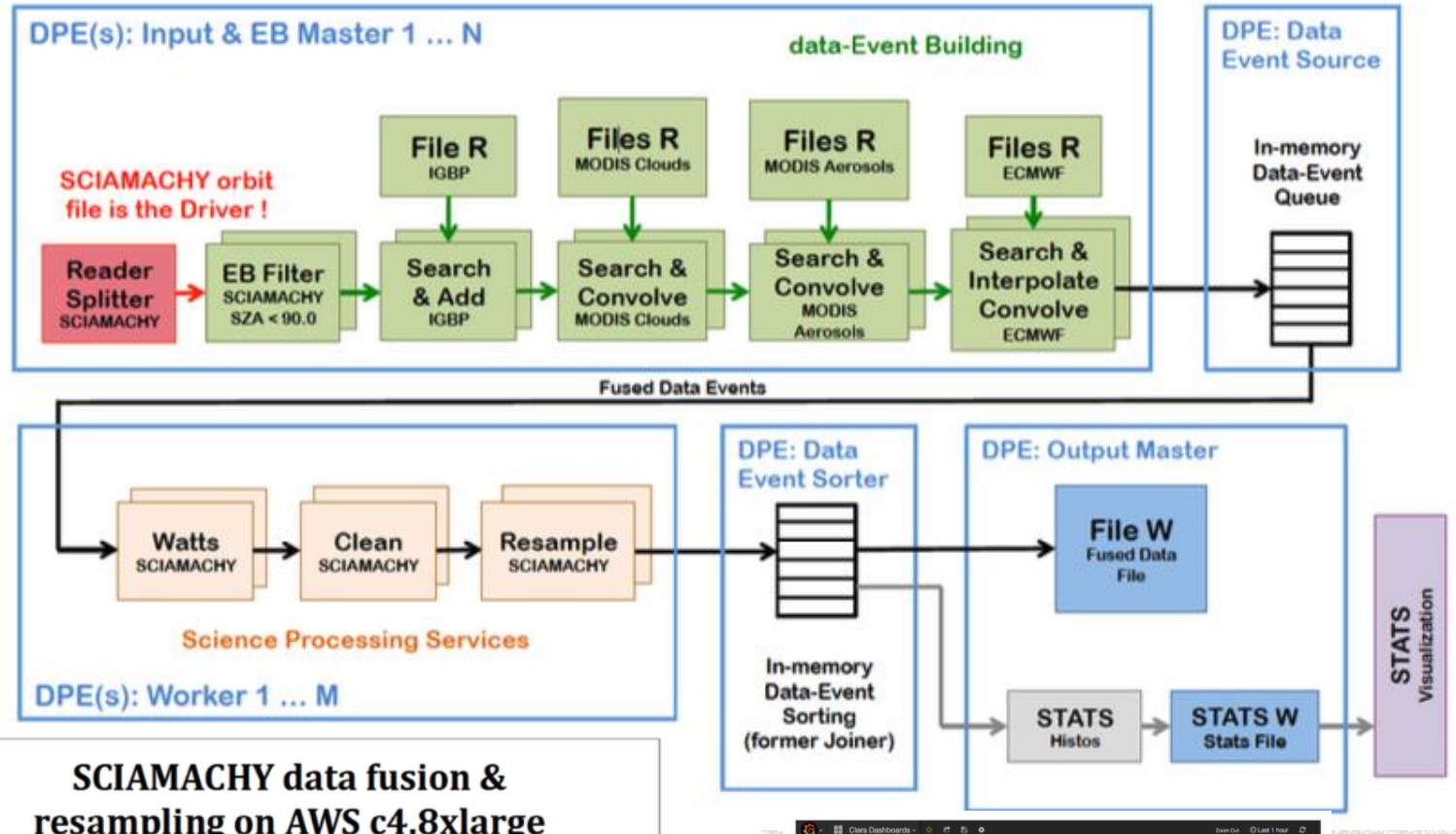
Staging data from AWS S3

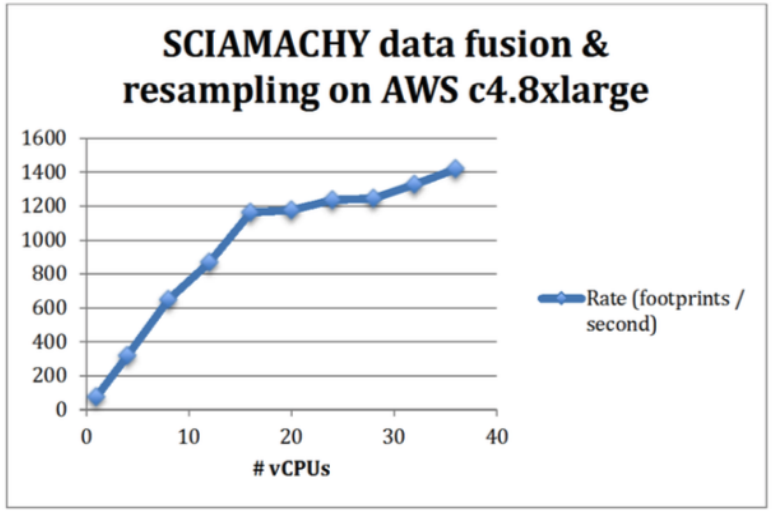
Data processing rate based on average workflow execution over 10 SCIAMACHY files.

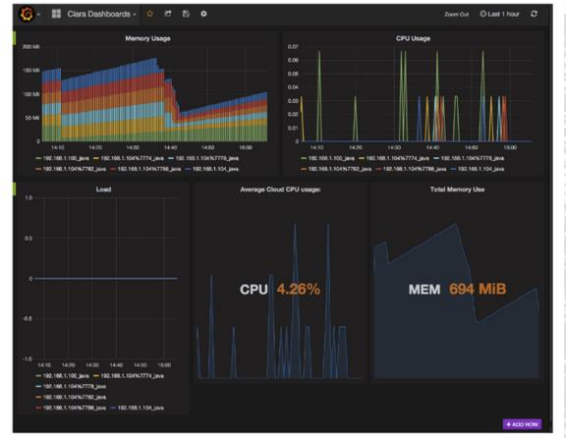
Vertical scaling up to 1.4KHz on single AWS node

Data processing continuous web monitoring

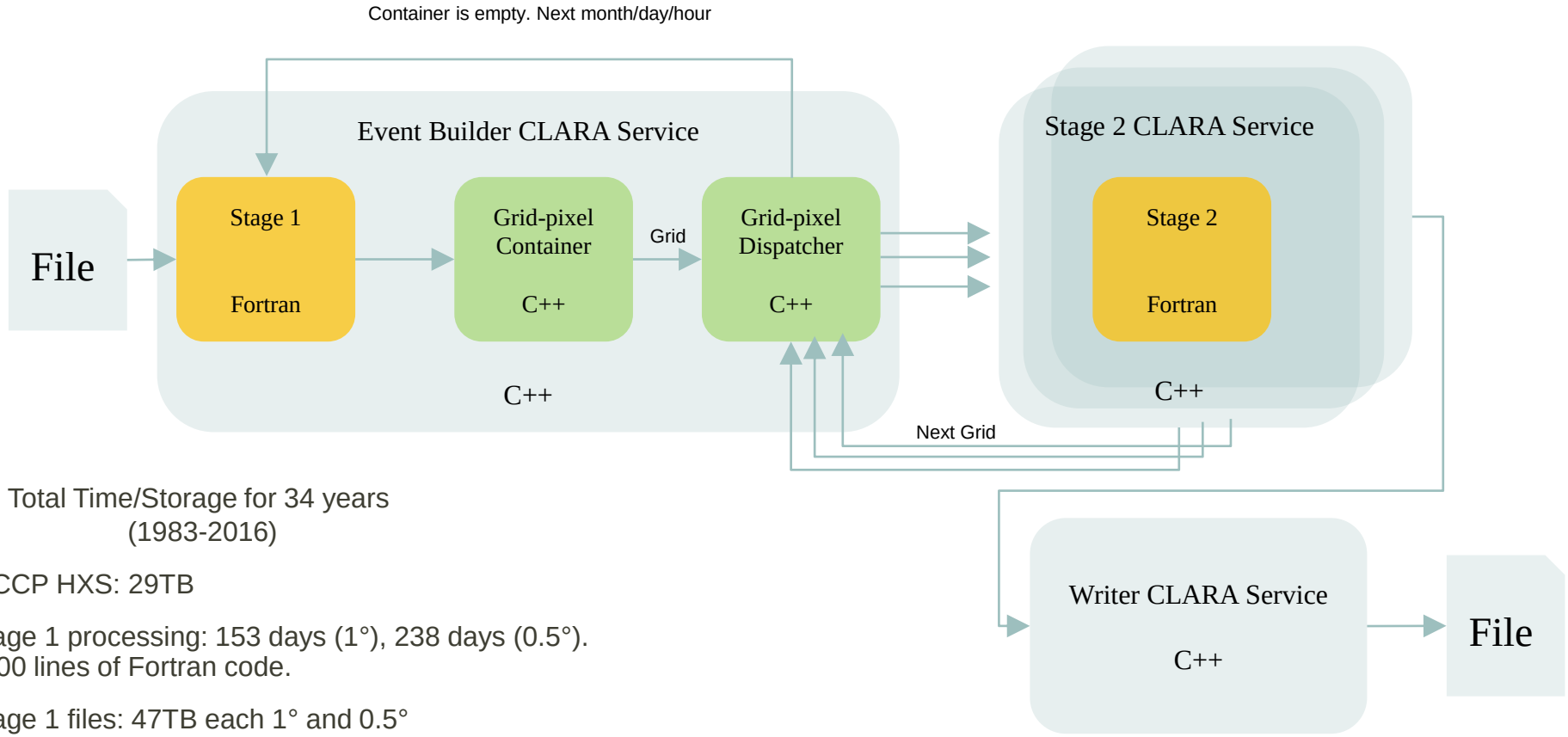
9 years of data has been processed. More data is currently being processed.







NASA SRB data flow diagram



Total Time/Storage for 34 years (1983-2016)

- ISCCP HXS: 29TB
- Stage 1 processing: 153 days (1°), 238 days (0.5°). 2200 lines of Fortran code.
- Stage 1 files: 47TB each 1° and 0.5°
- Stage 2 processing: 85 days (1°), 119 days (0.5°). 1700 lines of Fortran code.
- Stage 2 files: 1.1 TB (1°), 4.1TB (0.5°)

21 hours to process 1 month data ⟶ 45 minutes to process 1 month data

Conclusion

Jefferson Lab
Thomas Jefferson National Accelerator Facility

CLARA Reactive Data-stream Processing Framework

Micro-services architecture for data analytics

About:
We use CLARA to create software applications as a suite of independently deployable, small, modular services in which each service runs a unique algorithm, communicating with each other through a well-defined mechanism to serve data processing goals. Services of the same software application can be written in different programming languages, where each service is developed, optimized and maintained independently.

http://claraweb.jlab.org
https://github.com/JeffersonLab/clara-java
https://github.com/JeffersonLab/clara-cpp
https://github.com/JeffersonLab/clara-python

Resilient:
CLARA system stays responsive in the face of failure. Resilience is achieved by replication, containment, isolation and delegation. Failures are contained within each service, isolating services from each other and thereby ensuring that parts of the system can fail and recover without compromising the system as a whole.

Elastic:
CLARA system stays responsive under varying workload. It reacts to changes in the input data rate by increasing or decreasing the resources allocated to service these inputs (horizontal and vertical scaling). CLARA implements predictive, as well as reactive, scaling algorithms by providing relevant live performance measures. We achieve elasticity in a cost-effective way on commodity hardware and software platforms.

Robust:
CLARA uses asynchronous message-passing to establish boundaries between services that ensure loose coupling, isolation, location transparency, and provides means to delegate errors as messages. Employing explicit message-passing enables load balancing and overall data-flow, i.e. application algorithm control and orchestration.