Earth Science Data Fusion with Event Building Approach

and

NASA Information And Data System (NAIADS)

Constantine Lukashin, NASA LaRC, Hampton, VA

V. Gyurjyan, Jefferson Lab, Newport News, VA

A. Bartle & E. Callaway, Mechdyne Co., VA Beach, VA

A. Vakhnin, SSAI, Hampton, VA

S. Mancilla & R. Oyarzun, UTFSM, Chile

IEEE Big Data, Geoscience, 2015

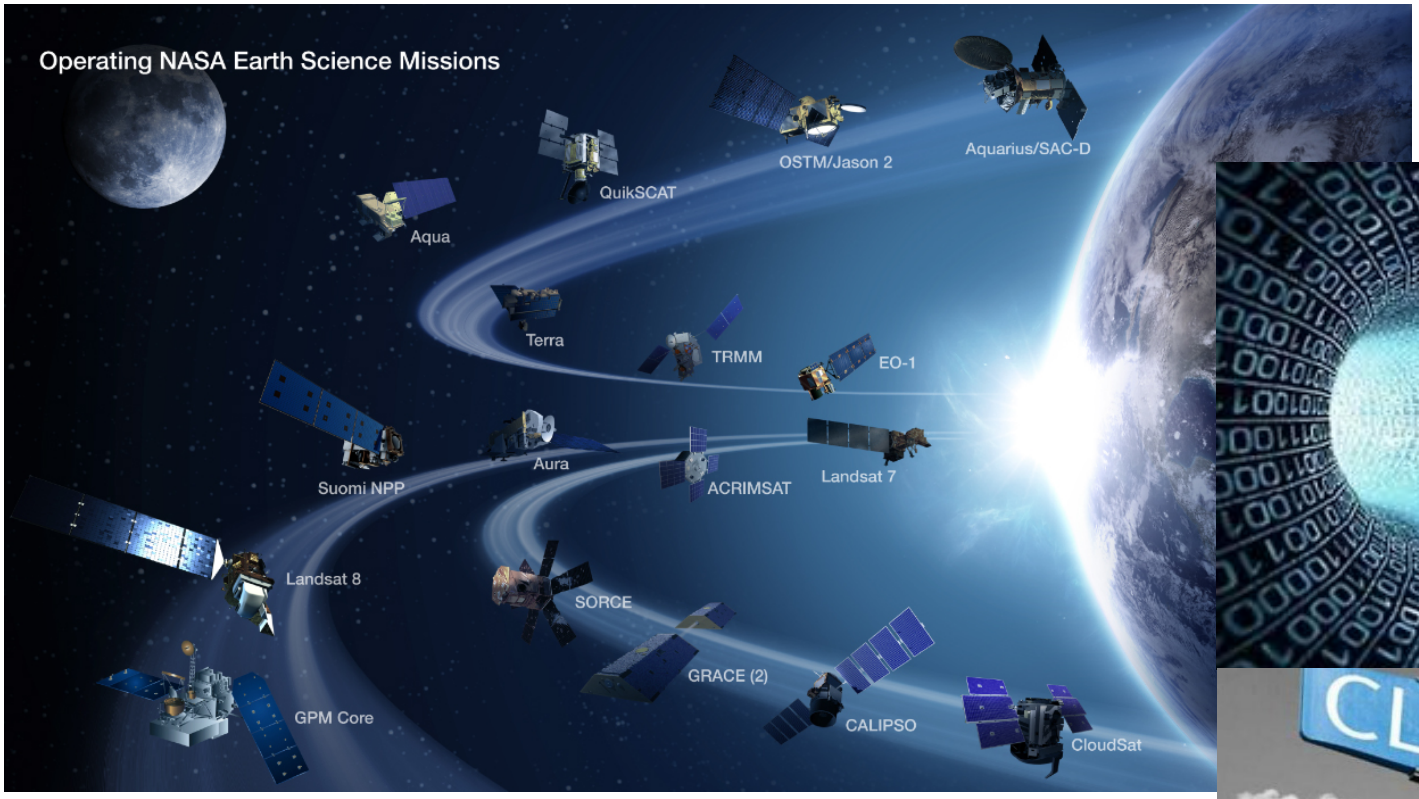
Presentation Outline:

- Earth Science Data Fusion: challenge & current state-of-the-art
- NAIADS objectives: data fusion advanced approach
- NAIADS system requirements
- NAIADS Project: NASA & DOE collaboration
- CLARA & ZeroMQ integration (Jefferson Lab)
- Earth Science data fusion: test cases
- NAIADS architecture and workflow
- NAIADS development: current status
- Summary and future work
- Proposal to ACCESS 2015 program
- Backup slides with technical details

Notes:

- NAIADS: NASA Information And Data System. *Naiad* is derived from the Greek *naein*, which means “to flow”, in this project context it is plural: *multi-data “flows”*
- Term “service” is used within Service Oriented Architecture concept: *service* is a functionality unit bound to orchestration (application). Essential features – loose coupling (independence), standardization, abstracting user from code and focusing on the data workflow.
- We may need a dictionary for the Service Oriented Architecture terms.

Earth Climate and Weather Systems



- Geostationary missions are not shown (15 internationally);
- Next 10 years: JPSS-1/2/3, GOES-R, PACE, TEMPO;
- New Decadal Survey and Venture Class missions;
- Missions last longer, sensors become more complex;
- We can predict the type and size of Earth Science data;
- The Climate Model outputs oversize the observations (currently 10s of PB);
- The OSSEs are becoming high priority.

Earth Science Data Fusion (L1 Requirement): To Maximize Information Content and Science Output

1. National Strategy (requirements):

- OSTP's Earth Civil Observation from Space (2013): Integrated Portfolio Management
- NASA Strategic Space Technology Investment Plan (2013) and Budget Memorandum (2014)
- NASA Strategy Plan 2014: Strategic Objective 2.2

2. Rationale: Outstanding Science Output

- CERES, Earth's Radiation Budget: multi-sensor (16) data fusion (NASA LaRC)
- CERES/MISR/MODIS data fusion: multi-sensor calibration validation on-orbit (NASA LaRC)
- CALIPSO, CloudSat, CERES, MODIS (A-Train) data fusion: Level-2 information (NASA LaRC)
- MODIS and PARASOL (A-train) data fusion: Level-1 measurements (U of Lille, France)
- Required for future missions: CERES/RBI, TEMPO, CLARREO, PACE, ACE, and GEO-CAPE
- Required for future *satellite constellations* (baseline or small satellites)

3. Major Challenges / Current state-of-the-art:

- IT infrastructure is not optimized for the task: data is distributed, slow connections, security…
- Traditional PGE-based workflow is I/O bound (job-per-file-per-CPU);
- Current infrastructure is network bound for heavy data handling;
- Time and resource intensive, the costs are very high.
- Opportunities are not realized: e.g. MISR/MODIS, A-train Level-1, etc.

Earth Observations and Climate Model Data

1. Observations:

- Expected volume of *data products*: exceed 100 Petabytes (?) in 10 years.
- Data is *distributed* nationally (NASA, NOAA, USGS), and internationally (ESA, etc.).
- NASA observational data *format is standard* HDF (Hierarchical Data Format)
- NOAA observational data *format is standard* NetCDF (Network Common Data Form)
- Observations from relevant sensors are *NOT synched / merged !*
- Each sensor/mission has a *separate data product line !*

2. Climate Models Output:

- Expected volume of CM outputs: may exceed Exabyte in 10-15 years.
- Data is *distributed* nationally (NASA, NOAA, DoE), and internationally (UK).
- Multiple Climate & Earth System models (30 – 50).
- Perturbed Physics Ensemble (PPE) for a single model: about 5 Petabytes.
- Climate Model output *format is standard* – NetCDF.

3. Relevant Data Centers:

- Observations: Langley and Goddard (NASA), NCDC (NOAA), LPDAAC (USGS), etc.
- Climate/Weather Models: Goddard, GISS and Ames (NASA), NCAR (NSF), LLNL and LBNL (DOE), etc., etc., etc.

NAIADS: L2 SYSTEM Requirements

- **IO Optimization:** Data *Event Builder in off-line software* before any scaling !
- **Networking:** Ability for local and distributed data-Event building.
  Distributed software approach – *move code to data, not otherwise !*
- **New Workflow / Scaling:** Complete in-memory data-Event streaming, massive scaling;
- **Multi-lingual:** Support new and heritage codes, optimal language for given service.
- **Flexibility:** Re-configurable to multiple applications (data fusion, processing, mining, etc.).
  Service Oriented Architecture (SOA) approach.
- **Adaptability:** Supporting various hardware configurations (cluster, cloud, servers, etc.),
  and various file systems: NFS, GPFS, Apache Hadoop, etc.
- **Portability:** Support various OS platforms (Unix, Linux, Apple OS X).
- **Standards:** I/O, internal transient data, and metadata.
- **Provenance:** Track metadata at any stage of processing (process, stats, phenomena).
- **Traceability:** End-to-end test cases, extensive and transparent code documentation.
- **Modern good practices:** From management to coding, maintenance, reusability.

NAIADS: NASA LaRC and DOE JLab Collaboration

1. NAIADS Key Data Fusion Principle:

Complete Earth Science data-Event Builder in off-line software for given algorithms !

2. xMsg (DOE Jefferson Lab):

- Publish/Subscribe messaging middleware (data streaming);
- Based on the ZeroMQ socket library (C++);
- Multi-lingual binding (Python, Java, C++ in development)

3. CLARA Framework (DOE Jefferson Lab):

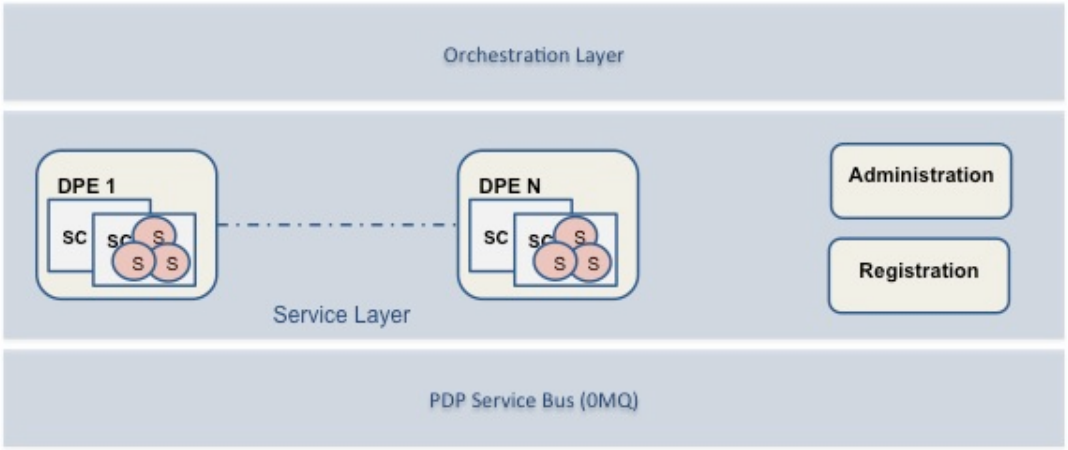
- Service Oriented Architecture framework for Physics data processing applications;
- Multi-lingual support (Python, Java, C++ in development);
- Previous Implementations:
  - CLAS12 physics data processing framework (Jefferson Lab, Hall-B);
  - Old Dominion University (Norfolk, VA) data mining.
- Publication:

V. Gyurjyan, et al. "CLARA: A Contemporary Approach to Physics Data Processing" Journal of Physics: Conference Series, 331 (2011).

4. NASA/DOE Inter-Agency Agreements (IAA):

- 1st IAA established in December 2014;
- 2nd IAA established in August 2015.
- Royalty free software license for the US Government use (not open source !)

CLARA Framework Architecture



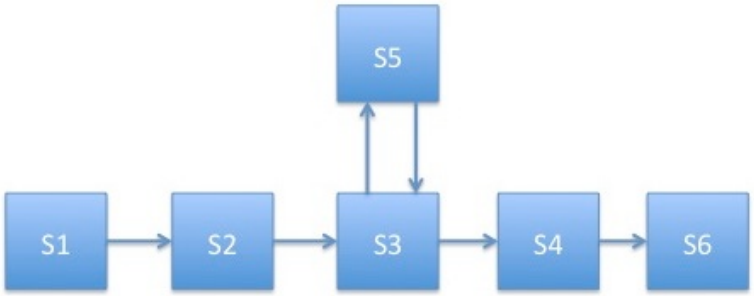
CLARA Web at Jefferson Lab: *https://claraweb.jlab.org*

Integration with ZeroMQ (iMatix):

- ZeroMQ is messaging socket library;
- Solves the challenge of creating large multicore applications in any language;
- Pub/Sub (asynch) patterns;
- Very fast (tests at CERN, 2011).

CLARA DPE (Data Processing Environment):
Described in the backup slides.

Composition Example: Data Calibration




```
S1 + S2 + S3;
if ( S3 == ”calibration”) {
   S3 + S5;
} else {
   S3 + S4 + S6;
}
```

List of Orchestration operators:
see the Backup Slides

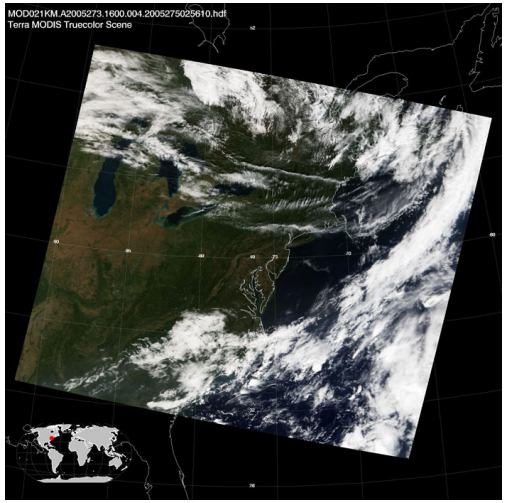
NAIADS Data Fusion: SCIAMACHY L1 / MODIS L2 / ECMWF

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 data product (requested by a number of NASA 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.

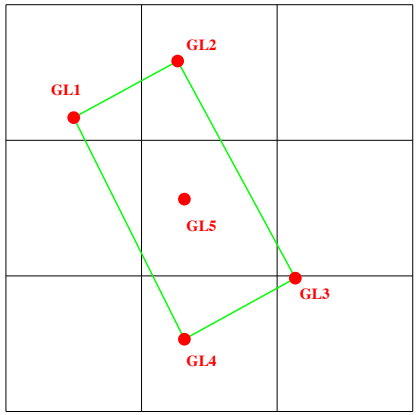


MODIS/Terra Level-2 Data:

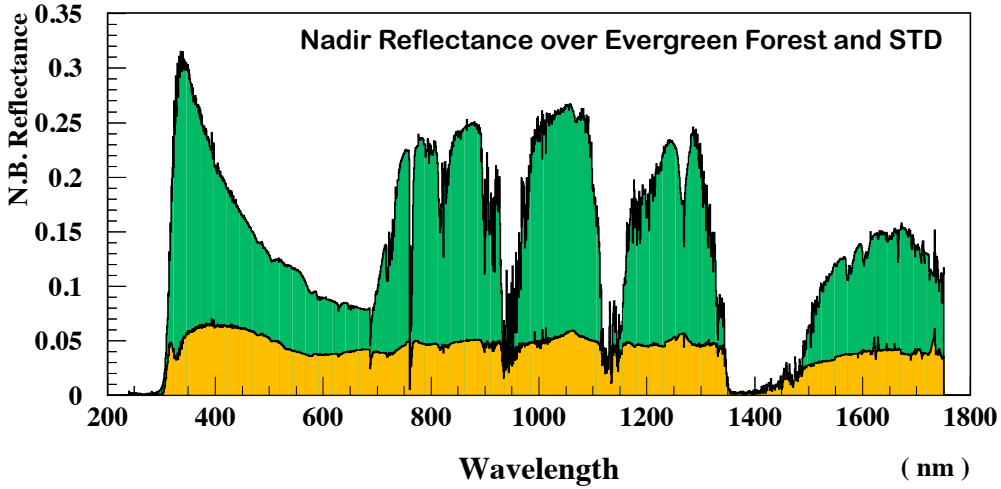
- Level-2 Cloud and Aerosol Data
- Spatial scale: 1 / 5 km and 10 km spatial;
- Swath 2500 km (global coverage daily);
- 10:30 AM Sun-synch orbit.

ECMWF Data (re-analysis):

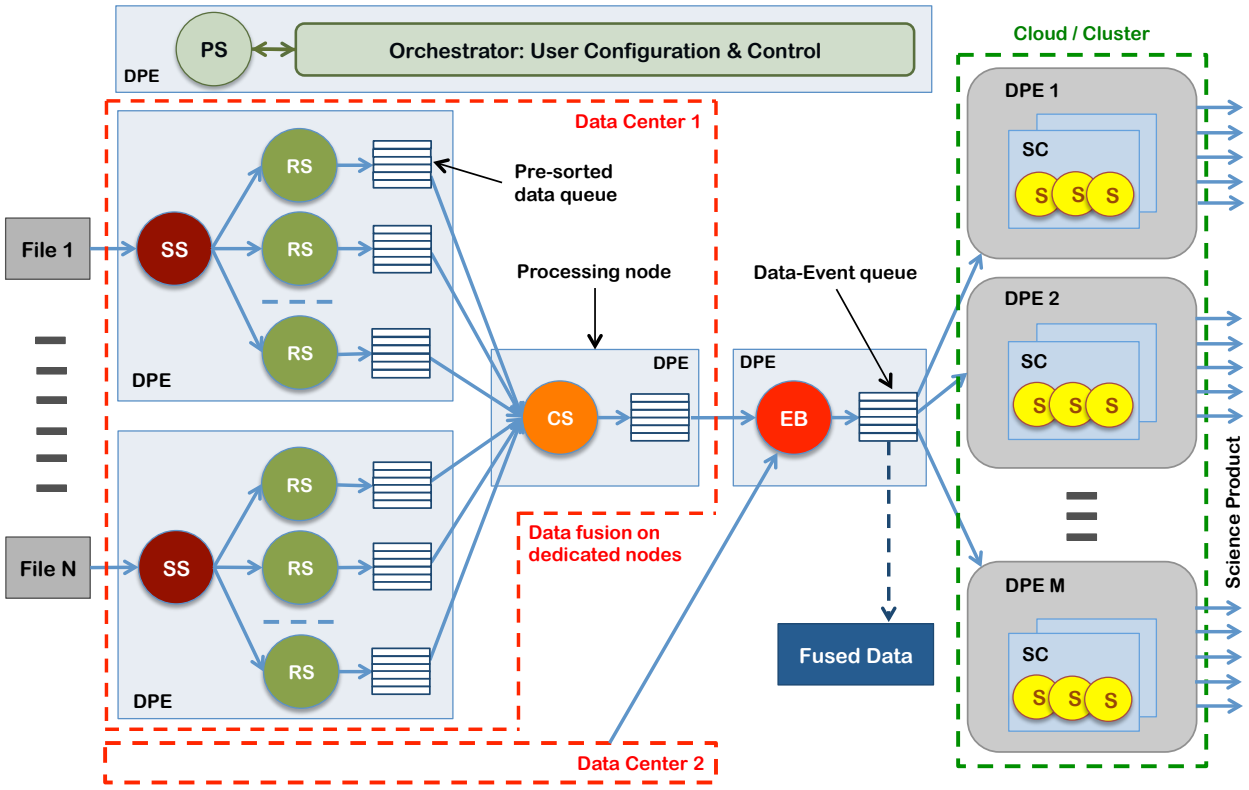
Gridded (0.25° or 0.5°) weather parameters.



SCIAMACHY FOOTPRINT 30 x 230 km

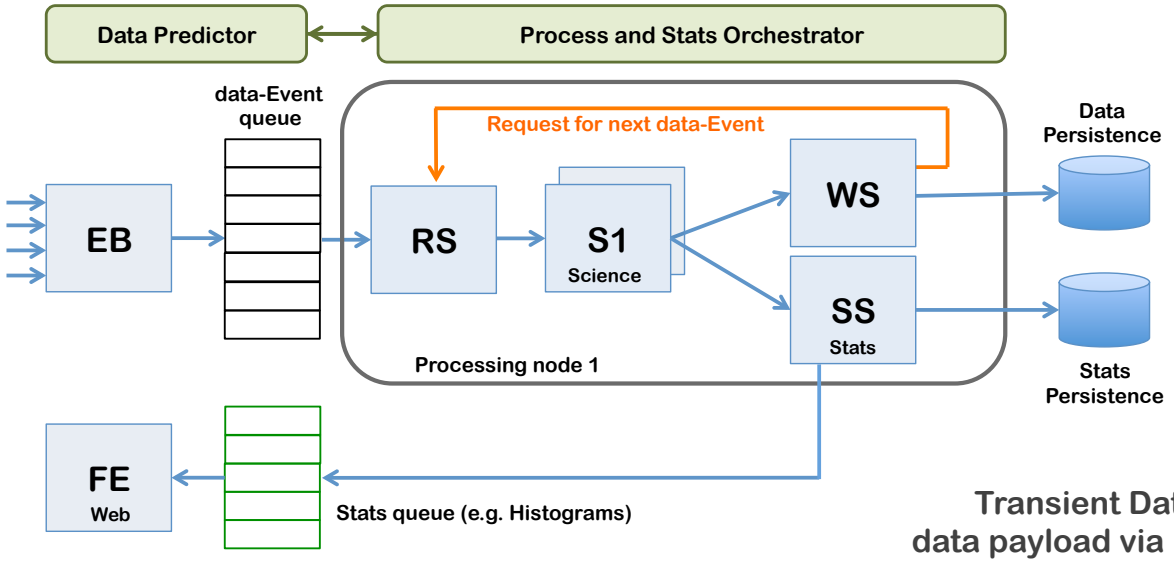


NAIADS Architecture



- CLARA/xMsg for flexible and multi-lingual applications (Java, Python, C++).
- PS: multi-sensor coincident data prediction service.
- SS: into-memory fast data staging service (multi-file).
- RS: parallel from-memory data reader service (pre-sorting).
- CS: data concentrator service in a data center (IO/network optimization).
- EB: complete data-Event Builder - all required data for given application !
- Scaling: data-Event streaming to Cloud with minimized IO.

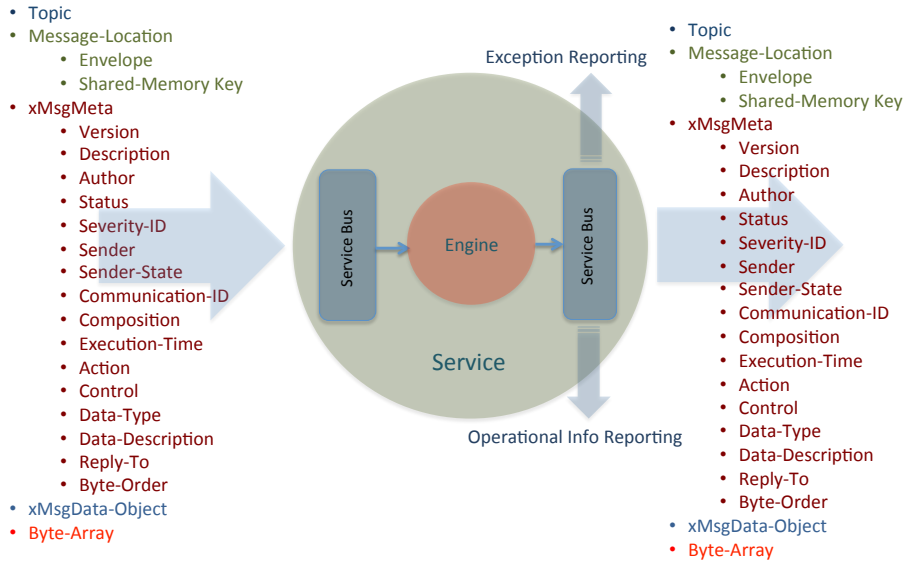
NAIADS Workflow Example: shown for a Single Node



- xMsg (ZeroMQ) messaging.
- EB: complete data-Event Builder.
- RS: parallel data reader service.
- S1: science algorithm service.
- WS: data persistence service.
- SS: statistics service (e.g. histograms).
- FE: front end user web service.

Scaling: Automatic multi-treading in nodes.

Transient Data Envelop and data payload via NetCDF Streaming



NAIADS: Current Status

1. DATA:
- SCIAMACHY data location: AWS S3 Bucket (9 years of data);
- SCIAMACHY data format: binary (small header + spectral records [5287]);
- Aux data: stationary IGBP Surface Index (binary, 10° map with 20 integer values per grid-box);

- MODIS Level-2 (HDF) and ECMW (NetCDF) test data staged on AWS S3 Bucket.

2. DATA FUSION:
- EB: Provide 5 locations of SCIAMACHY footprint (corners and center) with IGBP value.
- Option: persist merged data before processing.

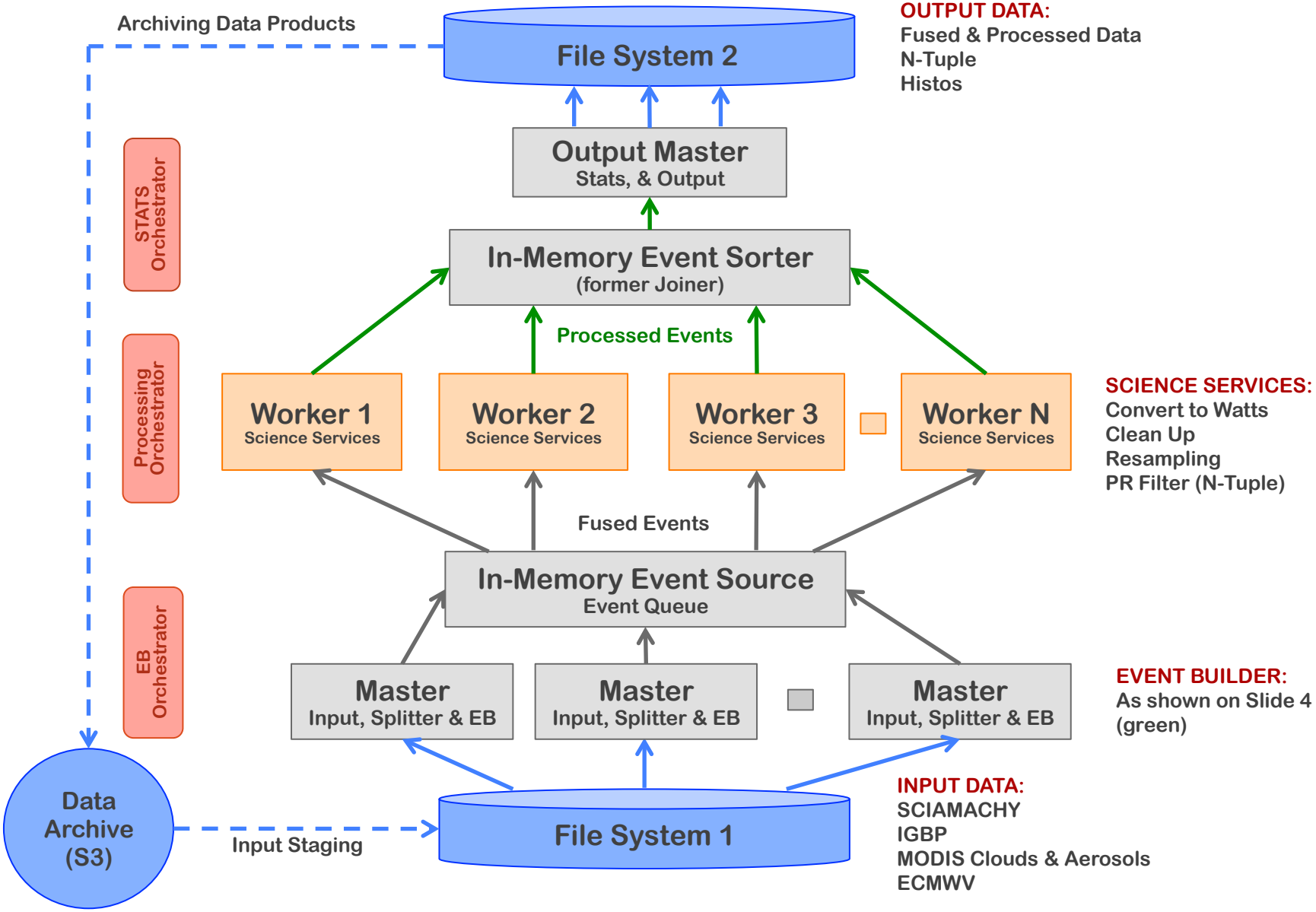
3. CLARA FRAMEWORK:
- Java implementation (jCLARA), optimized. Chosen as the NAIADS Baseline implementation.
- Python implementation (pCLARA), optimized.

4. DATA FORMATS:
- NetCDF and HDF I/O formats, NetCDF transient format (NetCDF Streaming in Java)

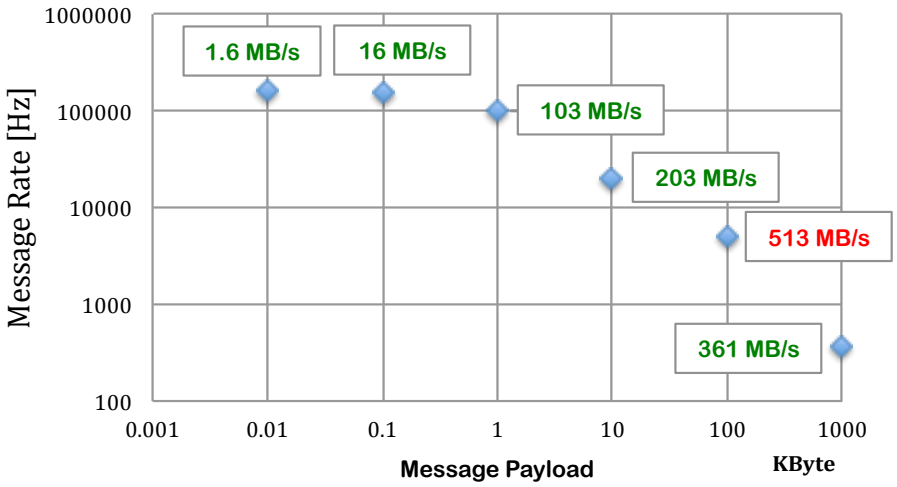
5. DATA PROCESSING (SC services are SCALED):
- SC1: Convert N-photons to Watts for every spectral record [5287]; (calibration !)
- SC2: Clean up every spectral record [5287] from bad data; (data quality !)
- SC3: Gaussian spectral resampling: from original [5287] to [1510] (1 nm wavelength sampling); (process !)
- SC4F: Data Filter for IGBP = 17 (ocean) only; (sub-setting !)
- WS: Write new data product (data-Event sorting by orbit ! Every event has an ID).

5. STAT SERVICES (under development, Java): (process control & data mining !)
- Histogram Types: H1D, H2D.
- Profile Types (averaged data objects): P1D, P2D [with automatic array loop]
- Generalized methods (add, subtract, normalize, project, fit)

NAIADS Cloud



NAIADS Status and Progress: CLARA/xMsg & Python Tests



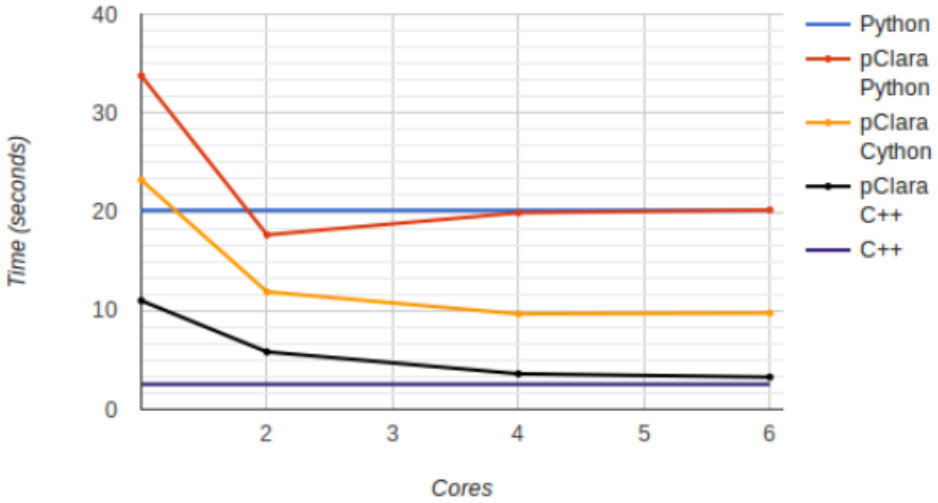
CLARA/xMsg Framework Performance:

jCLARA/xMsg transport is capable of transporting 513 MByte/sec data between processes/services within a single node (test on Intel 2.3 GHz i7 CPU);

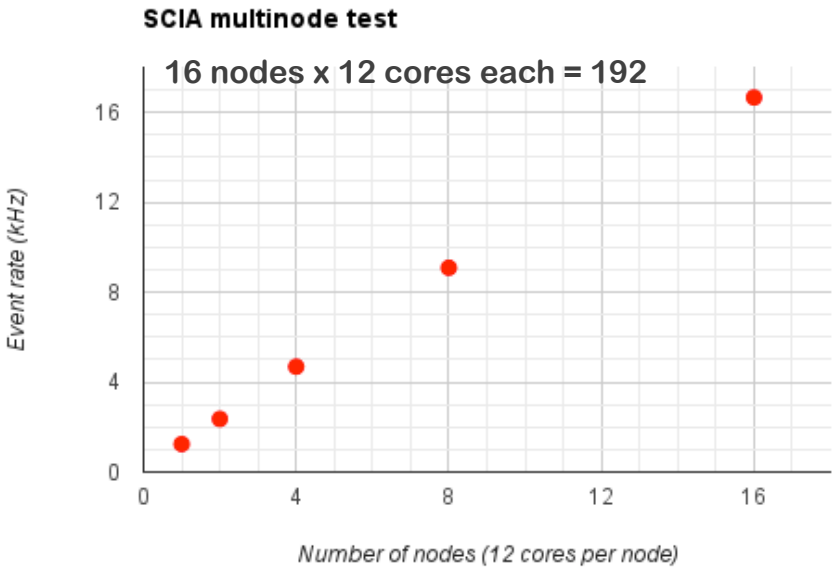
With heavy processing load: jCLARA/xMsg overhead contribution should be negligible.

NAIADS in Python Dialects:

NAIADS performance with pCLARA easily outperforms a *traditional* Python solution and can be coerced to approach the performance of a *traditional* C++ solution.



NAIADS Status and Progress: Streaming, Scaling and Web API



Java NAIADS/CLARA performance test:

Java FastMath lib: with 25% of the C++ base case;
Data-Event streaming implemented;
All Science Services implemented;
Writer Service with sorting output data-Events;
Java implementation - baseline for tuning up.
64 SCIAMACHY L1 files, data moved to local disks.

WEB API Objectives:

Get the registration information from all components;
Get filtered information by querying the system;
Deploy CLARA DPEs;
Deploy CLARA Containers;
Deploy NAIADS Services;
Handle NAIADS/CLARA message subscriptions.

I/O and Transient Data: NetCDF Streaming

Input: Binary, NetCDF, HDF;
Output: NetCDF;
Transient data payload: NetCDF;
From Unidata (UCAR) fixed and debugged;
Java implementation (Python and C++ are planned).

Examples of Web APIs (Python and the Django REST):

applications Show/Hide | List Operations | Expand Operations | Raw

GET	/applications/	List all Applications
POST	/applications/	Create new Clara Application
GET	/applications/{application_id}/	Retrieve an application
DELETE	/applications/{application_id}/	Removes an application

containers Show/Hide | List Operations | Expand Operations | Raw

GET	/dpes/{DPE_id}/containers/	Find all containers for determined dpe
POST	/dpes/{DPE_id}/containers/	Create a new Clara Container
GET	/dpes/{DPE_id}/containers/{container_id}/	Get the registration information of a Clara Container using its id
DELETE	/dpes/{DPE_id}/containers/{container_id}/	Delete an specific Container instance using its id
GET	/dpes/{DPE_id}/containers/{container_id}/services/	Get the registration information of the Service Engines for a specific container
POST	/dpes/{DPE_id}/containers/{container_id}/services/	Deploy a new service at Container
GET	/dpes/{DPE_id}/containers/{container_id}/services/{service_id}/	Get the registration information of a Service Engine for specific container
GET	/containers/	Find all containers
POST	/containers/	Create a new Clara Container
GET	/containers/{container_id}/	Get the registration information of a Clara Container using its id
DELETE	/containers/{container_id}/	Delete an specific Container instance using its id

NAIADS Statistics Services (Java):

REQUIREMENTS for Histograms (frequency) and Profiles (averages):

Binned Objects	Methods
H1D (frequency)	Addition/Subtraction; Normalization; Integration; Fitting
H2D (frequency)	Addition/Subtraction; Normalization; Integration; Projections; Fitting
P1D (averages)	Normalization; Combining (+/-): means, STD, counts; Fitting
P2D (averages)	Normalization; Combining (+/-): means, STD, counts; Fitting

PROGRESS:

- Data interface with NetCDF streaming is implemented;
- Data gridding/averaging part is implemented;
- We use numerically stable algorithm for the statistics calculations;
- Methods: add, subtract, normalize, and projection of 2D to 1D are all implemented;
- Exporting the data from stats structures into JSON, and using Highcharts (Javascript library) to make interactive visualization for the browser.

NAIADS Approach and Resulting Impacts:

1. **Novel Data Fusion Approach:**
   Optimization of the IO in large data applications & improving efficiency of process scaling.

2. **Multi-Lingual Solution:**
   Support for most used programming languages (C++, Java, Python), providing framework from new development and heritage codes. Provide choice for optimal implementation for given algorithm.

3. **Generalized Data Statistics:**
   Set of essential statistical analysis tools for data of multiple types: from low to high level data products, observations or models, space or ground experiments.

4. **Flexible Data-Streaming Framework Platform:**
   Workflow adaptability to multiple applications – data mining, metadata generation, sub-setting, data analysis.

5. **Foundation and test bed for a large distributed community data system.**

IMPACT: These technologies has potential to improve efficiency and reduce costs of data handling within Earth Science community.

BACKUP SLIDES

NAIADS Technical Summary

SUMMARY:

- Objectives: data fusion with existing 9-years of Earth Science data;
- CLARA/xMsg framework in Java and Python;
- Implemented test: 4 science services for existing Earth Science data (Java and Python);
- Simple data-Event Builder in software implemented;
- Data-Event streaming in NetCDF implemented (Java);
- IO formats in NetCDF/HDF implemented;
- Initial WEB APIs prototyped (REST);
- Stat services prototyped (Java, High-charts);
- Automation and testing in AWS Cloud compute environment.

WORK IN PROGRESS & FUTURE WORK:

- The NAIADS Test Case: extend to MODIS L2 and ECMWF re-analysis data;
- Development of data fusion Event Builder algorithm;
- Benchmark performance of the NAIADS IO and transient data envelope;
- Benchmark the NetCDF and HDF into-memory data Staging Services;
- Development of C++ CLARA, testing and benchmarking;
- Implement NetCDF Streaming in Python and C++;
- Extensive testing on the AWS Cloud with large number of nodes;
- Continue to development statistical services;
- Continue to improve jCLARA, pCLARA performance;
- Continue to develop NAIADS/CLARA Web APIs.

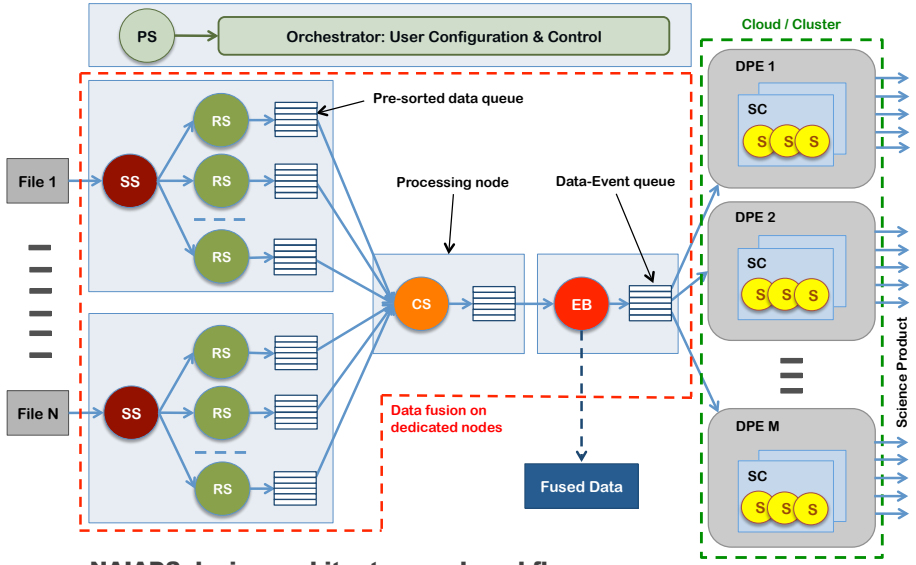
NASA Information And Data System (NAIADS) for Earth Science Data Fusion and Analytics

Principal Investigator: Constantine Lukashin / NASA LaRC, Hampton, VA.

Description and Objectives:

- Earth Science observations:
  - Large and fast growing data volume;
  - Distributed nationally and internationally;
  - Data is not merged into a system;
  - Scaling is a challenge due to intensive I/O.

- NAIADS Objectives:
  To develop prototype of new framework for Earth Science data fusion (maximize information content) and processing. The focus of new middleware:
  - Optimize the I/O and data workflow;
  - Massive process scaling with data streaming;
  - Operations with large and distributed data sets.



NAIADS design architecture and workflow

NAIADS Approach:

- Service Oriented Architecture (SOA).
- Leverage existing middleware CLARA (Jefferson Lab).
- Leverage existing ZeroMQ messaging technology.
- Event Building approach in software – I/O optimization.
- In-memory processing workflow.
- Optimization to the Cloud Computing environment.
- SOA code portability, flexibility and re-use.
- Generalized data services: read, control, analytics.

Co-I's: A. Bartle (Mechdyne), V. Gyurjyan (Jefferson Lab)
C. Roithmayr (NASA LaRC), A. Vakhnin (SSAI)

Milestones / Schedule:

- Formulation & Definition of Requirements (03/2015)
- Establish Software Approach, Environment, Prototyping (05/2015)
- Test Case 1: SCIAMACHY/IGBP, Build-1 (08/2015)
- Software Generalization, Services for MODIS L2 Data (11/2015)
- Test Case 2: SCIAMACHY/MODIS L2, Build-2, TRL4 (02/2016)
- Data Services Generation, Web Services Prototyping (05/2016)
- Data Fusion Data Set (9 years), NAIADS Build-3, TRL5 (08/2016)
- System Generalization and Optimization (11/2016)
- NAIADS Performance Benchmark, Build-4, TRL6 (01/2017)

Entry TRL = 3 **Current TRL = 3+**

NAIADS: Current Status

LANGUAGES for SERVICES: Java (baseline & optimization) and Python (optimization).

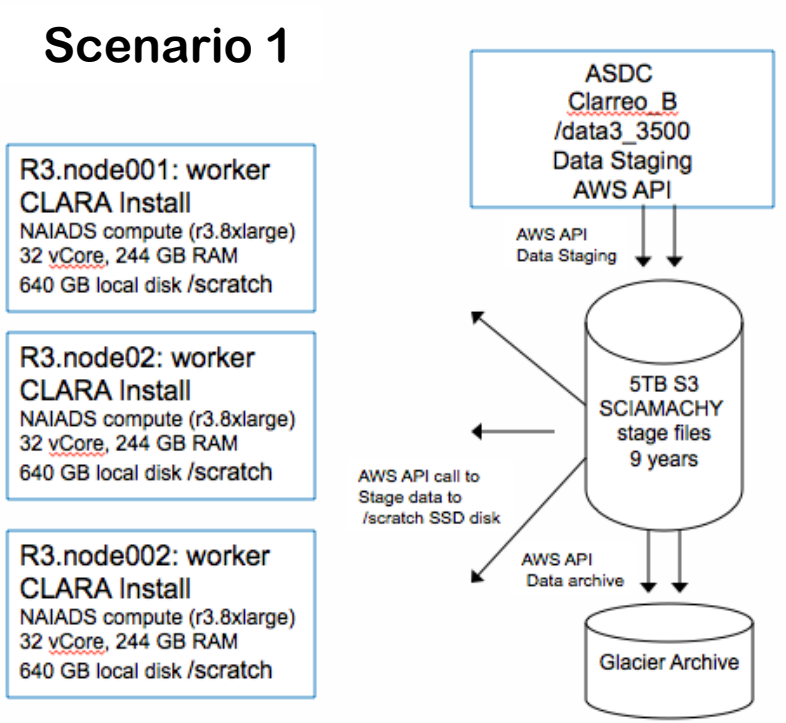
BASE CASE to compare with: C++ optimized code (traditional file-per-job-per-core).

AWS Current Setup:

- S3 Bucket / Storage;
- Cloud configurations:
  Scenario 1
  Scenario 2

AWS Current Setup, Scenario 1: Approach with *in-every-node* Event Building and process scaling.

- Worker nodes: C3.8xlarge instances;
- NAIADS test runs automated;

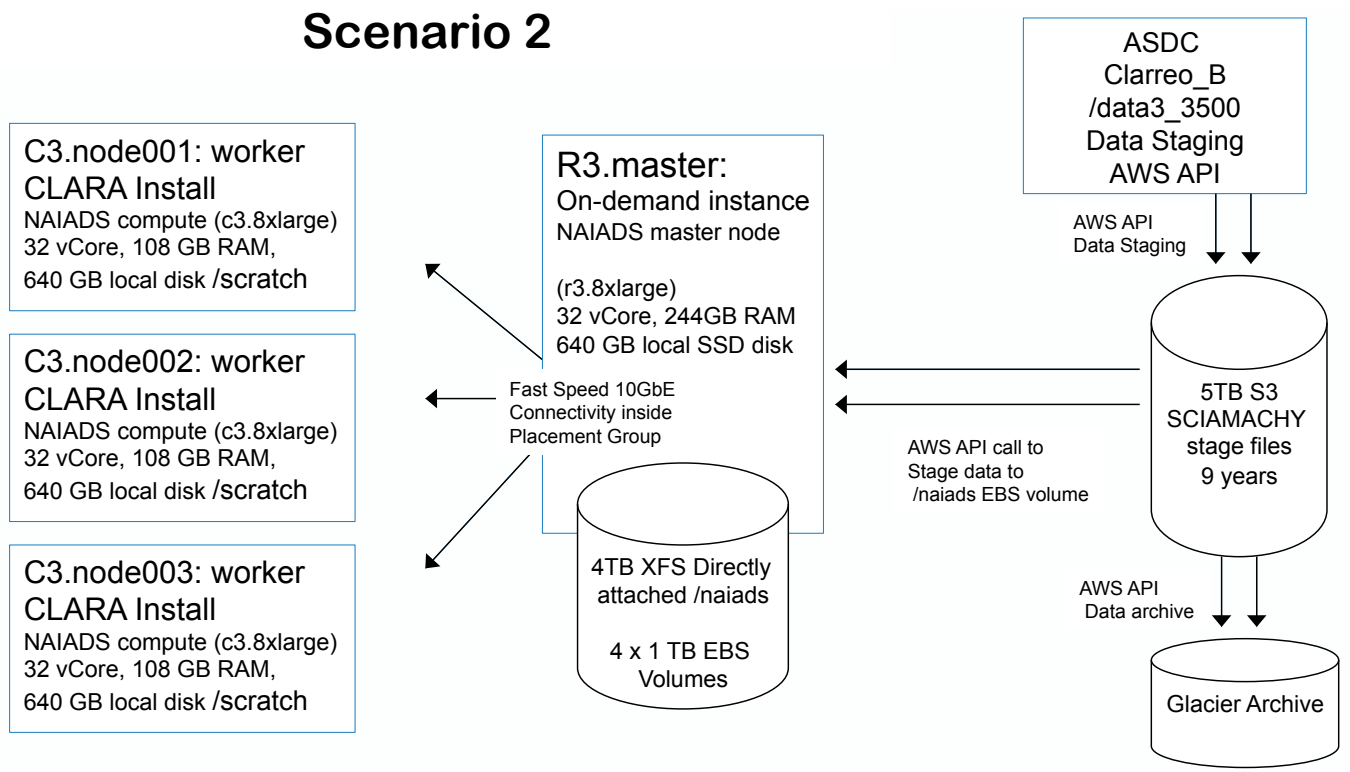


AWS Compute Facility Configurations:

- Cloud Cluster configuration with shared file system (when available in AWS East);
- NAIADS Cloud Scenario 1 (shown): performance worker nodes only w/ RAM disks.
- NAIADS Cloud Scenario 2 (see next slide);

NAIADS: Current Status

AWS Current Setup, Scenario 2:
Approach with dedicated Event Building in Master node (or nodes) and process scaling in worker nodes
Framework networking



AWS Compute Facility Configuration

NAIADS/CLARA Orchestrator: New Operators

Operator	Description	Example
+	data link operator defining the data route	s1 + s2
,	data multiplexing or *logical OR* operator	s1 + s2,s3 s1,s2 + s3
&	logical AND operator	s1,s2 +&s3
;	data branching operator indicating the end of a statement	s1+s2; s2 +s3;

{ }	scope operator defines a set of routing instructions	unbound {
		s1+s2;
		s2+s3+s4;
		}

==	service state equal operator	if(s1==state1)
!=	service state is not-equal operator	if(s1 != state2)

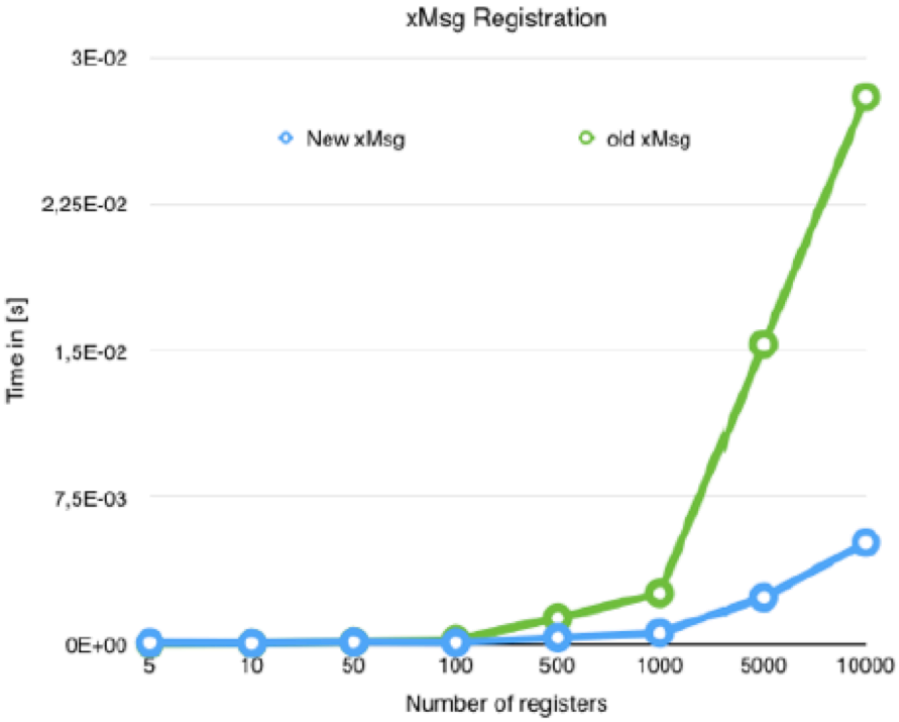
if,elseif,else	conditional statement operators	If(s1 == state1) {
		} elseif(s1 != state2) {
		} else {
		}

unbound	defines a scope not bound to a condition	unbound{ }

&&	logical AND operator	if ((s2 === "xyz") &&
		(s1 == "abc")) {
		s2 + s3;
		}

||	logical OR operator	if ((s2 === "xyz") ||
		(s1 == "abc")) {
		s2 + s3;
		}

pCLARA/xMsg Optimization (Python):



The message size was set up to 50 kB (size of SCIAMACHY record), amount of messages varied from 10,000 to 100,000 with 10,000 step.

Registration retrieval and time of response for the Registration services have improved considerably – compare green (old) and blue (new) xMsg-Python performances at large number of registrations.