Earth Science Data Fusion: NAIADS Concept and Development

PI: Constantine Lukashin (NASA LaRC, SD)

V. Gyurjyan (DOE Jefferson Lab)

A. Bartle, E. Callaway (Mechdyne, Inc.)

A. Vakhnin (SSAI, Inc.)

S. Mancilla, R. Oyazun (UTFSM, Chile)

Project is selected for the NASA Technology Infusion Reports 2015

- 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”*

Earth Climate and Weather Systems

- Geostationary mission are not shown (15);
- Next 10-15 years: JPSS-1 and JPSS-2, GOES-R;
- Decadal Survey and Venture 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.

Earth Observations and Climate Model Data

1. Observations:

- Expected volume of *used data products*: exceed 100 Petabytes in 10-15 years.
- Data is *distributed* nationally (NASA, NOAA), 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 different 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), NSIDC.
- Climate/Weather Models: Goddard, GISS and Ames (NASA), NCAR (NSF), LLNL and LBNL (DOE), etc., etc., etc.

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

1. National Strategy:

- 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. Relevancy: Outstanding Science Output

- CERES, Earth's Radiation Budget: multi-sensor data fusion, up to 16 (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)
- Required for future missions: CERES/RBI, TEMPO, CLARREO, ACE, and GEO-CAPE
- Required for future *satellite constellations* (baseline or small sats)

3. Major Challenges:

- 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);
- Traditional configurations are 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.

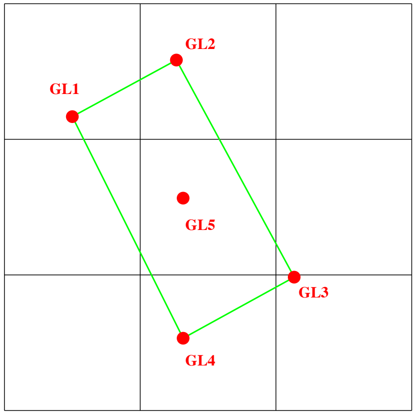
NAIADS Test Case: SCIAMACHY/MODIS Data Fusion

OBJECTIVES:

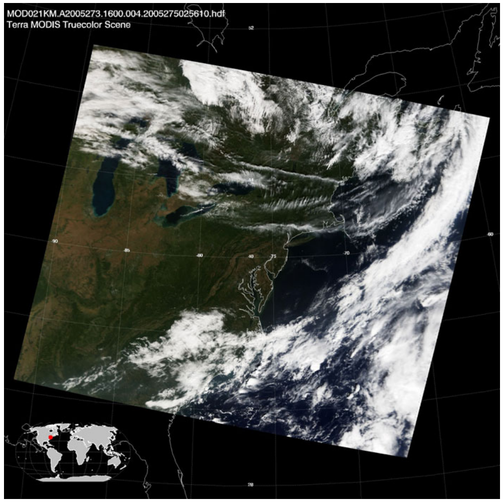
- To demonstrate NAIADS approach and full functionality using existing data;
- To benchmark NAIADS performance versus traditional job-per-file cluster approach;
- Available data: 9 years of near-coincident measurements of from SCIAMACHY and MODIS;
- Create new fused SCIAMACHY/MODIS data product (requested by a number of NASA projects).

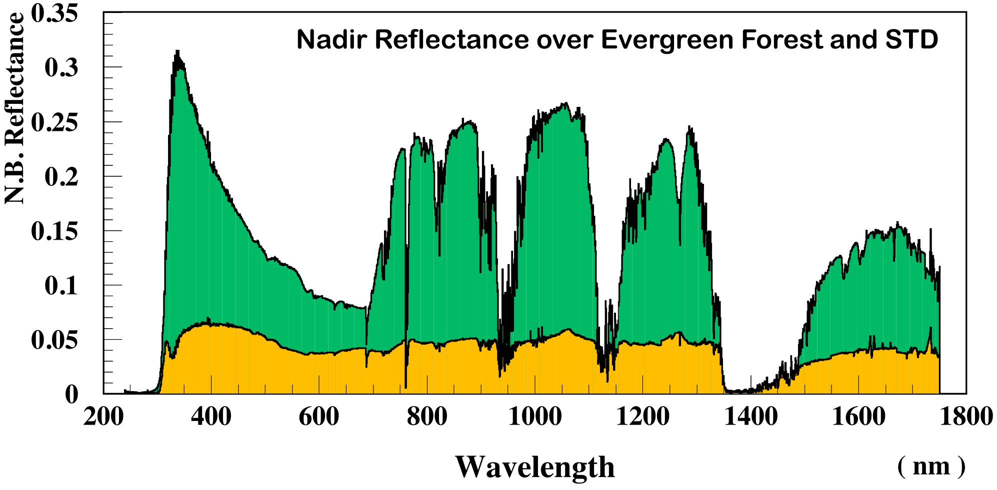
SCIAMACHY DATA:

- Spectral measurement for every footprint: 30 km x 230 km;
- Swath 950 km (4 footprints) from 10 AM Sun-synch orbit.



SCIAMACHY FOOTPRINT 30 x 230 km





MODIS/Terra DATA:

- Multi spectral measurement: 19 RS Bands;
- Level-2 Cloud and Aerosol Data
- Spatial scale: 1 / 5 km and 10 km spatial;
- Swath 2500 km (global coverage daily);
- 1:30 AM Sun-synch orbit.

NAIADS Framework L2 Requirements

- **IO Optimizer:** 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 (fusion, processing, data mining, etc.).
  Service Oriented Architecture (SOA) approach.
- **Adaptability:** supporting various hardware configurations (cluster, cloud, servers, etc.),
  and various file systems: NFS, GPFS, Apache Hadoop / Spark, etc.
- **Portability:** Support various OS platforms (Unix, Linux , MacOS, …)
- **Standards:** I/O, transient data, and metadata.
- **Provenance:** track metadata at any stage of processing.
- **Traceability:** end-to-end test cases, extensive and transparent code documentation.
- **Modern good practices:** from management to coding, persistence.

Technology Infusion: CLARA Framework from Jefferson Lab

1. xMsg (DOE Jefferson Lab):

- Publish/Subscribe messaging middleware;
- Based on the ZeroMQ socket library (C++);
- Multi-lingual binding/support (Python, Java, C++)
- Author: V. Gyurjyan

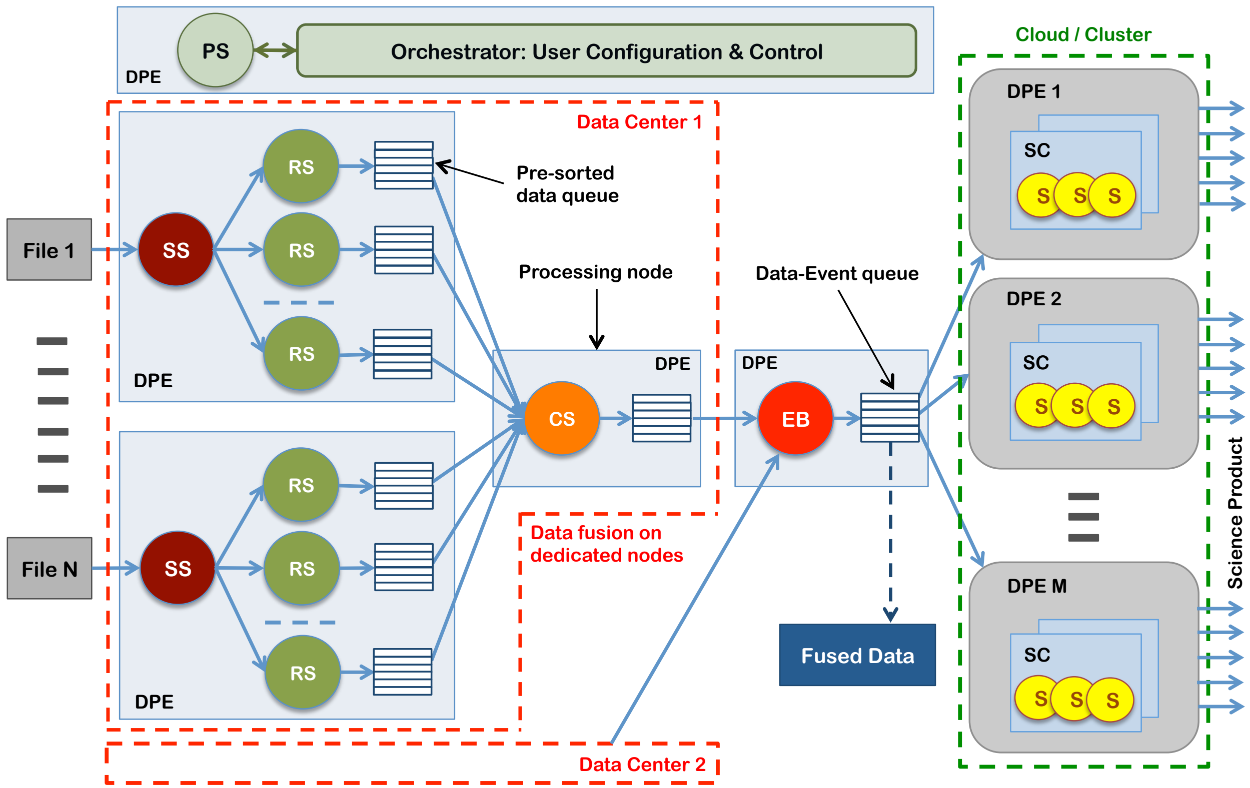
2. CLARA Framework (DOE Jefferson Lab):

- Service Oriented Architecture framework for Physics data processing applications;
- Multi-lingual support (Python, Java, C++ planned);
- Documentation: claraweb.jlab.org
- Implementations up to date:
  - 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).

3. NASA LaRC & DOE Jefferson Lab: Inter Agency Agreement

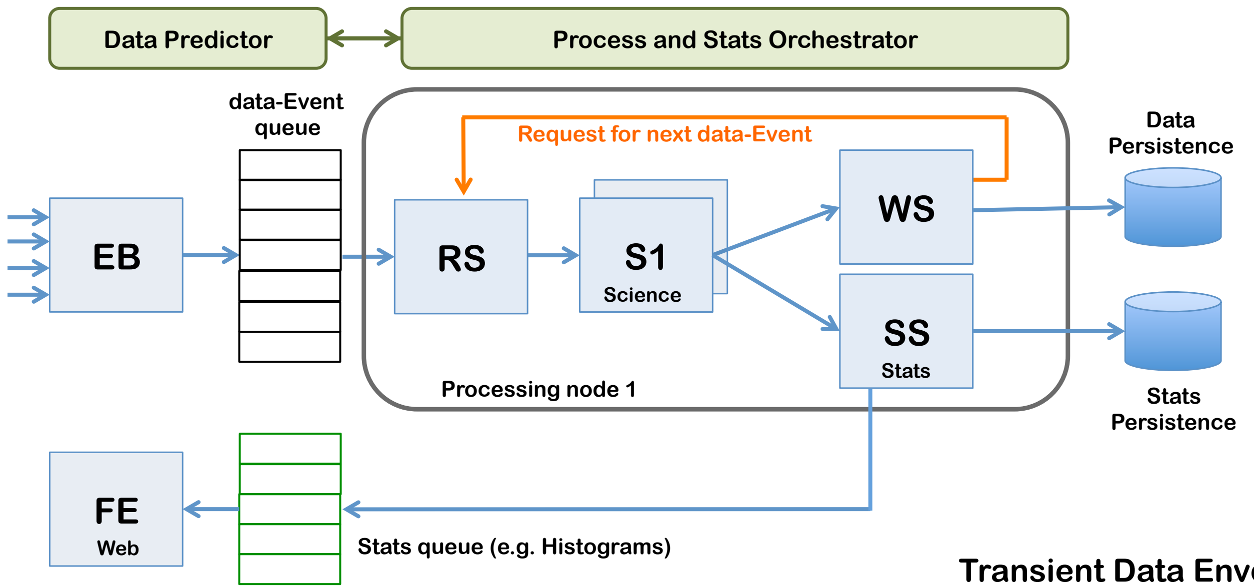
- Established in December 2014;
- Renewed on annual basis;
- Royalty free software license for the US Government use.

NAIADS Architecture

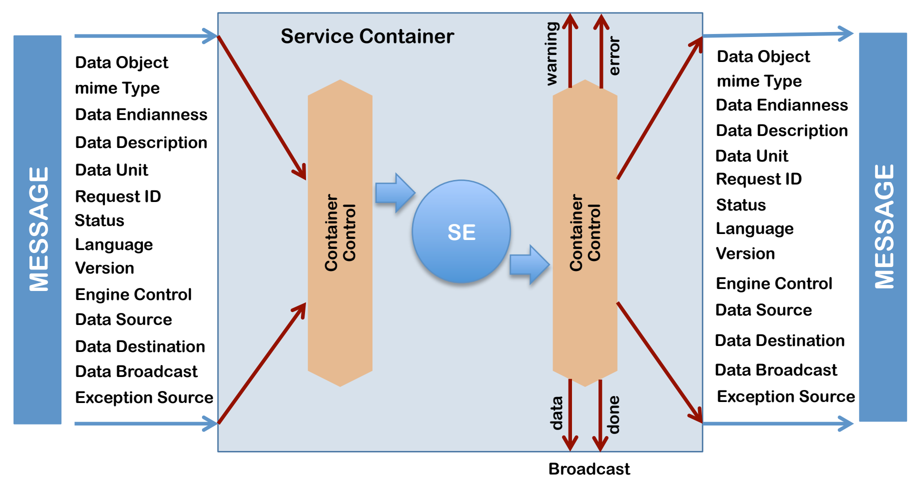


- SOA CLARA/xMsg for flexibile and multi-lingual applications.
- PS: multi-sensor coincident data predictor service.
- SS: into-memory fast data staging service (multi-file).
- RS: parallel from-memory data reader service (pre-sorting).
- CS: data concentrator service in a data center (IO/network optimization).
- EB: complete data-Event Builder (adaptation to algorithm).
- Scaling: data-Event streaming to Cloud with minimized IO.

NAIADS Workflow Example: shown for a Single Node



Transient Data Envelop: data payload via NetCDF streaming



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

Scaling: Automatic multi-treading in nodes.

NAIADS Project Management

NAIADS WBS (Level-2)

- WBS 01 Management
  - WBS 01.01 LaRC Management
  - WBS 01.02 JLab Management
  - WBS 01.03 Planning & Control
  - WBS 01.04 AMS Cloud Management
  - WBS 01.05 Document Management
  - WBS 01.06 Project Reviews
- WBS 02 CLARA
  - WBS 02.01 pCLARA
  - WBS 02.02 jCLARA
  - WBS 02.03 C++CLARA
  - WBS 02.04 In-Memory Services
  - WBS 02.05 CLARA Sum. Services
  - WBS 02.06 CLARA Website
- WBS 03 Science Serv.
  - WBS 03.01 Orchestrator Services
  - WBS 03.02 Reader Services
  - WBS 03.03 Data-Event Builder
  - WBS 03.04 Science Algorithms
  - WBS 03.05 Histogram Services
  - WBS 03.06 Fitting Services
  - WBS 03.07 N-Tuple Services
  - WBS 03.08 Persistence Services
- WBS 04 Web Services
  - WBS 04.01 User Services
  - WBS 04.02 Monitoring Services
  - WBS 04.03 Visualization Services
- WBS 05 AWS Cloud
  - WBS 05.01 Cluster Cloud
  - WBS 05.02 NAIADS Cloud
- WBS 06 Earth Science Data
  - WBS 06.01 SCIAMACHY
  - WBS 06.02 MODIS
  - WBS 06.03 ECMWF Data
- WBS 07 Integration
  - WBS 07.01 Transient Data
  - WBS 07.02 Database Services
  - WBS 07.03 NAIADS Security
  - WBS 07.04 NAIADS Builds
  - WBS 07.05 Benchmarking
- WBS 08 Documentation
  - WBS 08.01 Wiki Pages
  - WBS 08.02 User Guide
  - WBS 08.03 Reference Guide
  - WBS 08.04 NAIADS Paper
  - WBS 08.05 CLARA Paper
- WBS 09 Outreach
  - WBS 08.01 JLab Seminar
  - WBS 08.02 LaRC Seminar
  - WBS 08.04 CDT Seminar
  - WBS 08.05 Conferences

NAIADS Wiki and Git Repository on EARTHDATA: https://wiki.earthdata.nasa.gov/display/NAIADS/

NAIADS Test Case 1: SCIAMACHY/IGBP Data

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);

2. DATA FUSION:

- EB: Provide 5 locations of SCIAMACHY footprint (corners and center) with IGBP value.
- Enable option: persist merged data before processing.

3. DATA PROCESSING (SC services, SCALED, output persisted via WS):

- SC1: Convert N-photons to Watts for every spectral record [5287];
- SC2: Clean up every spectral record [5287] from bad data;
- SC3: Gaussian spectral resampling: from original [5287] to [1510] (1 nm wavelength sampling);
- SC4F: Data Filter for IGBP = 17 (ocean) only;
- WS: Persist new data product (data-Event sorting by orbit ! Every event has an ID)

4. STAT SERVICES (under development, Java):

- Histogram Types: 1DF, 2DF,
- Averaged data objects: 1DAv, 2DAv

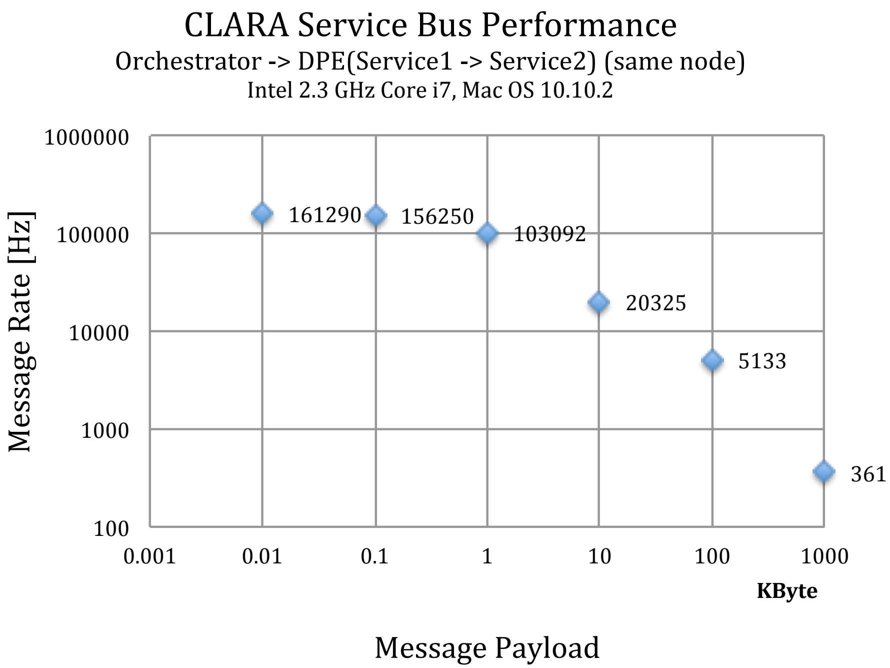
LANGUAGES: Java (baseline & optimizations) and Python.

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

AWS Setup:

- S3 Bucket / Storage;
- Initial Cloud configurations: traditional Cluster (shared file system) and Cloud.

NAIADS Status and Progress: CLARA/xMsg & Python Tests



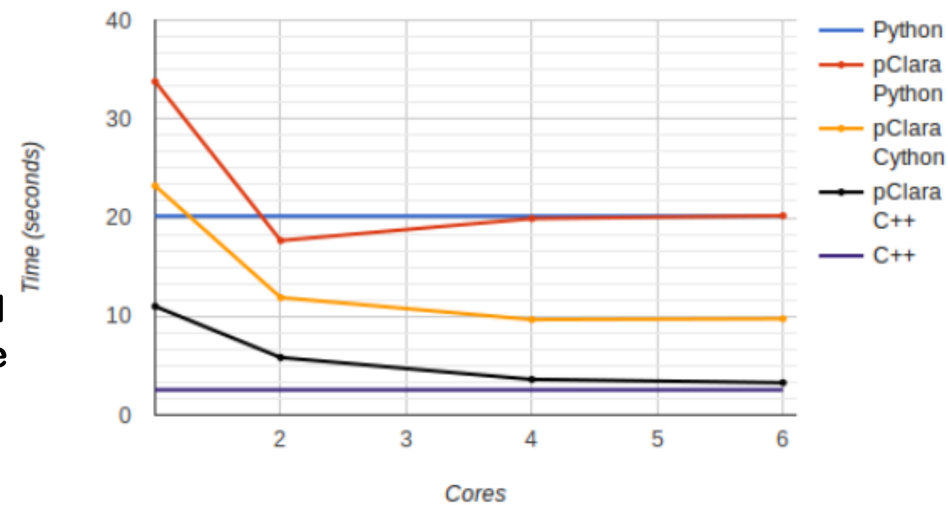
CLARA/xMsg Framework Performance:

jCLARA/xMsg transport is capable of transporting 360 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.

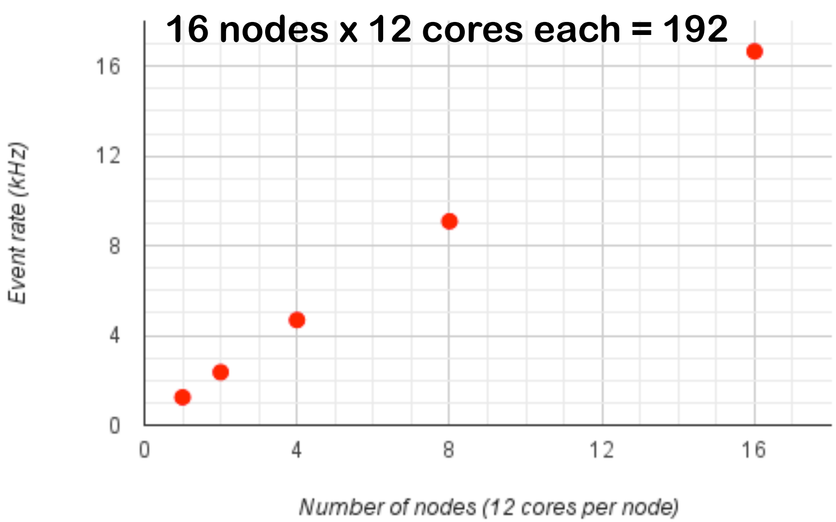
NAIADS in Python Dialects:

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

SCIA multinode test



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 (C++ is possible).

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

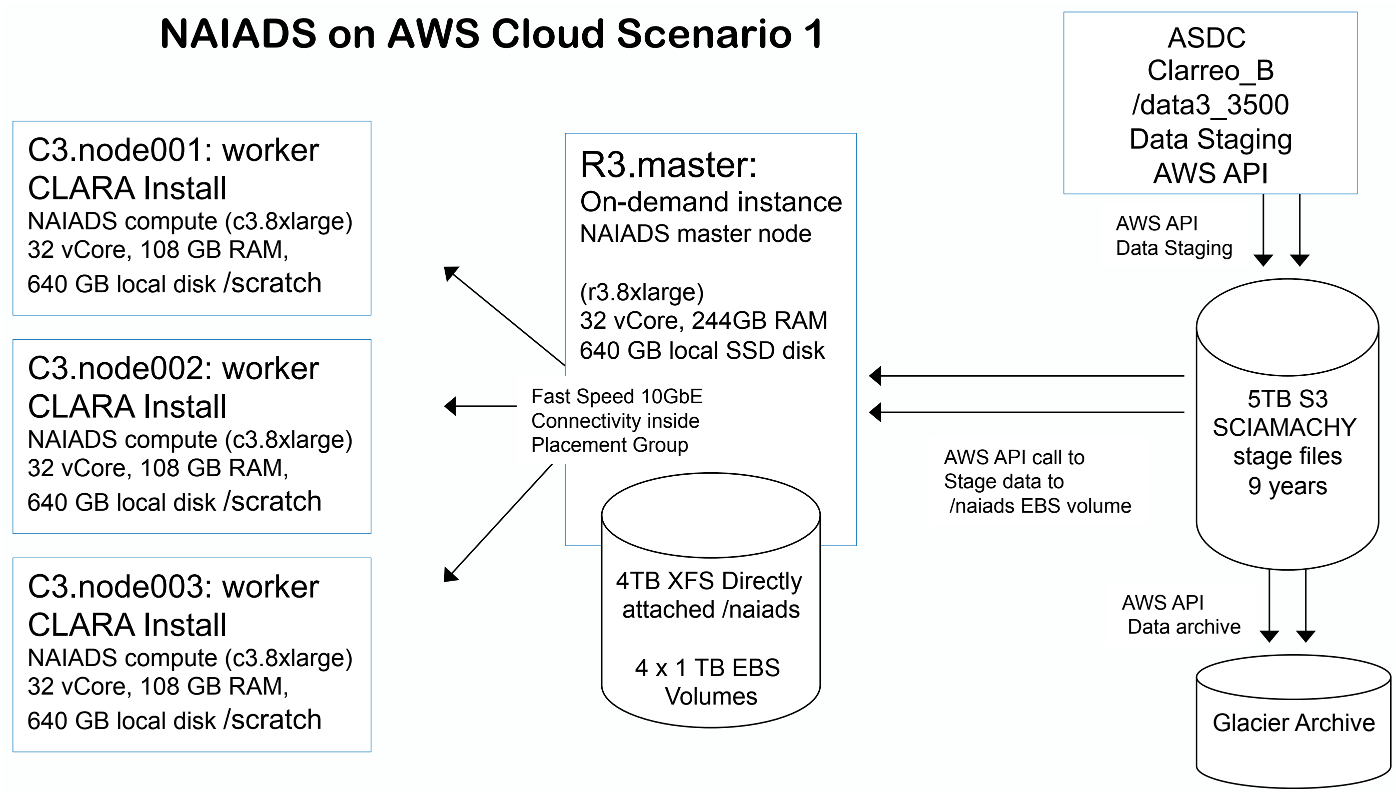
applications — Show/Hide | List Operations | Expand Operations | Raw

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 Status and Progress: AWS Cloud Configurations



AWS Compute Facility Configurations:

- **Traditional Cluster configuration with shared file system;**
- **Cloud Star Cluster configuration;**
- **NAIADS Cloud Scenario 1 (shown);**
- **NAIADS Cloud Scenario 2: performance worker nodes only w/ RAM disks. (expected winner)**

NAIADS Summary & Future Work

SUMMARY:

- Objectives: real data fusion with existing 9-years of Earth Science data;
- CLARA/xMsg technology infusion in Java and Python;
- Test case: 4 science services for existing Earth Science data;
- Simple data-Event Builder in software implemented;
- Data-Event streaming implemented;
- IO and Transient data via NetCDF Streaming;
- Initial WEB APIs.
- Testing in AWS Cloud compute environment.

WORK IN PROGRESS:

- The NAIADS Test Case: extend to MODIS L2 and ECMWF re-analysis data;
- Development of Test Case data-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;
- Design and development of NAIADS statistical services;
- Continue to improve jCLARA, pCLARA performance;
- Continue to develop NAIADS/CLARA Web APIs.

Near-term Objective: NAIADS stable software Build-1, achieving TRL4.

NAIADS/CLARA Demo: Java Implementation

[jNAIADS/jCLARA demo]