Earth Science Data Fusion with Event Building Approach

C. Lukashin[1], A. Bartle[2], E. Callaway[2], V. Gyurjyan[3], S. Mancilla[4], R. Oyarzun[4], and A. Vakhnin[5]

[1] NASA Langley Research Center, Hampton, VA
[2] Mechdyne Corporation, Virginia Beach, VA
[3] Thomas Jefferson National Accelerator Facility, Newport News, VA
[4] University Tecnica Federico Santa Maria, Chile
[5] Science Systemt and Applications Inc., Hampton, VA

Abstract: Objectives of the NASA Information And Data System (NAIADS) project are to develop a prototype of a conceptually new middleware framework to modernize and significantly improve efficiency of the Earth Science data fusion, big data processing and analytics. The key components of the NAIADS include: Service Oriented Architecture (SOA) multi-lingual framework, multi-sensor coincident data Predictor, fast into-memory data Staging, multi-sensor data-Event Builder, complete data-Event streaming (a workflow with minimized IO), on-line data processing control and analytics services. The NAIADS project is leveraging CLARA framework, developed in Jefferson Lab, and integrated with the ZeroMQ messaging library. The science services are prototyped and incorporated into the system. Merging the SCIAMACHY Level-1 observations and MODIS/Terra Level-2 (Clouds and Aerosols) data products, and ECMWF re-analysis will be used for NAIADS demonstration and performance tests in compute Cloud and Cluster environments. ***Keywords: earth science, data fusion, framework, event builder***

I. Introduction

One of the key elements of advancing our understanding of Earth's weather and climate via remote sensing is integration of diverse measurements into the observing system. As remote measurements capture larger amounts and higher quality of data, the demand for advanced data applications and high-performance information processing systems becomes a greater challenge. These challenges are outlined in the OSTP Guidelines for Civil Space Observations (2013), recognized in the NASA Strategic Space Technology Investment Plan (2013), and addressed in the NASA Strategic Objective 2.2 and its implementation by *"...developing new technologies and predictive capabilities, and demonstrating innovative and practical uses of the programs data and results for societal benefit"* (2014). The concept of maximizing information content by combining coincident multi-sensor data and enabling advanced science algorithms, was successfully used by several past and on-going projects: CERES experiment [1] for deriving accurate radiation fluxes, fusion of the CERES, MODIS and MISR observations for estimating instantaneous shortwave flux uncertainties, and multi-instrument calibration comparison [2, 3], fusion of MODIS and PARASOL observations to enhance cloud and aerosol retrievals, fusion of data from CALIPSO, CloudSat, CERES, and MODIS (A-Train constellation) for comprehensive aerosol and cloud information [4]. Advanced science algorithms allowed to reduce uncertainty in weather and climate parameters. The future satellite constellations and NASA missions: RBI, TEMPO, CLARREO, ACE, and GEO-CAPE will require tools for efficient data fusion and process scaling.

In response to these challenges, we develop the *NASA Information And Data System (NAIADS)* – a prototype framework for the next generation Earth Science multi-sensor data fusion and processing. The NAIADS' goal is to provide a novel approach to significantly improve efficiency in the Earth Science multi-sensor big data processing and analysis by deploying conceptually new workflow and state-of-the-art software technologies.

II. CLARA Data Streaming Framework

The NAIADS is integrated with CLARA, a Service Oriented Architecture (SOA) framework [5]. developed at the Thomas Jefferson National Accelerator Facility (Jefferson Lab), and ØMQ socket library [6, 7]. The CLARA framework is designed with a service-oriented architecture to enhance the efficiency, agility, and productivity of data processing tasks [8]. Data processing application, developed using the CLARA framework, consist of chained services, which are loosely coupled and can participate in multiple algorithmic compositions. It is important to mention that CLARA makes a clear separation between the service programmer and the data processing application designer. An application designer can be productive by designing and composing data processing applications using available, efficiently and professionally written software services without knowing service programming technical details. Services usually are long-lived and are maintained and operated by their owners in the distributed CLARA software. This approach provides an application designer the ability and flexibility to modify data processing applications by incorporating different services in order to find optimal operational conditions, thus demonstrating the overall agility of the CLARA framework approach.

This framework was designed based on a specific set of principles. As mentioned above, the fundamental unit of CLARA based data processing application logic is the service. Services exist as independent software programs with a common interface defined by the framework. User classes, encapsulating specific algorithms and compliant to the required interface, can be presented as CLARA services (the CLARA Software-as-a-Service: SaaS implementation). Each service has its own set of data processing functionalities. These functionalities or capabilities, suitable for invocation by other services, can be discovered via registration information available from the CLARA platform registry. One of the service design recommendations is to keep a small and simple service code base, which will help future programmers to easily extend, modify, maintain and port services. Services must be agnostic to any eternal data processing logic. Services must be discoverable and able to take part in complex service compositions. By standardizing communication between services, adapting a data processing application to changes in one of its components becomes easier and simplifies data transfer security (for example by deploying a specialized access control service).

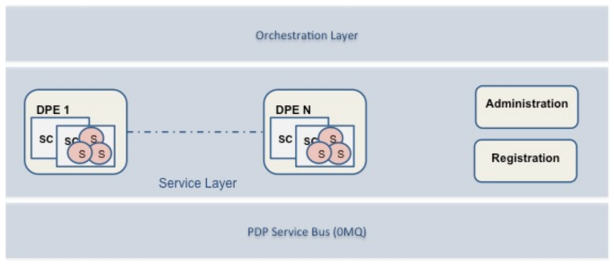

Figure 1: CLARA 3-layer architecture.

The CLARA architecture consists of tree layers as shown in Figure 1: The first layer is the xMsg Service Bus that provides the ØMQ-based publish-subscribe messaging system. Every service or component from the orchestration layer communicates via this bus, which acts as a messaging tunnel between services. Such an approach has the advantage of reducing the number of point-to-point connections between services required to allow them to communicate in the distributed CLARA computing environment. The xMsg is a messaging system, build upon the ØMQ socket library [6], and can scale to tens of thousands of processes if needed. It implements communication patterns such as topic pub-sub, workload distribution, and request-response. The service layer houses the inventory of services used to build data processing applications. The Administrative & Registration stores information about every registered service in the service layer, including address, description and operational details. The orchestration of data analyses applications is accomplished by the help of an application controller, resident in the orchestration layer of the CLARA architecture.

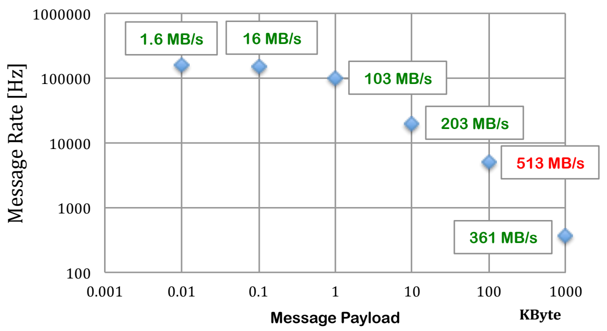

Figure 2: CLARA Service Bus performance: capability of transporting up to $\sim$ 513 MB/sec within single node.

The benchmark measurements, Figure 2, were performed on an Intel 2.3 GHz i7 CPU, utilizing a single core. The results show that CLARA's xMsg-messaging is capable of transporting 360 MByte/sec data between processes/services within a single node. The NAIADS data processing test-case implementations suggest that data processing latency is expected to be many orders of magnitude slower than CLARA/xMsg data transfer latencies, and CLARA framework overhead would be negligible.

III. NAIADS Architecture

The NAIADS design is based on the implementation of specific or algorithms/functionalities, required for Earth Science data tasks, and their integration with the CLARA framework. The NAIADS architecture and workflow is shown in Figure 3. Blue and grey rectangles represent framework's Data Processing Environment (DPE) units, *services* are shown with circles, and blue arrows represent transient data flow. Data staging (SS), reading and pre-sorting (RS), concentrating (CS) and data-Event building (EB) is performed on dedicated nodes (within red dashed line), data-Event-based streaming and processing on Cloud/Cluster with minimized IO indicated within green dashed line.

The overall workflow supports multi-stream data fusion by mapping input files into virtual memory from servers with optimal IO access to files. The raw data is then locally filtered and sorted into an in-memory data queue based on fusion parameters. Records from these queues undergo one or two levels of concatenation to

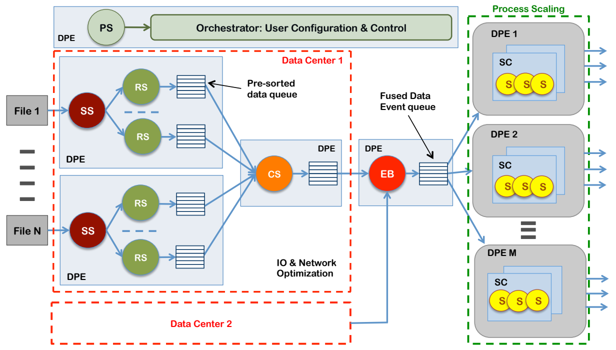

Figure 3: NAIADS architecture and workflow: data staging (SS), in-memory reading (RS), data concentrating (DS), and data-Event building (EB) is performed on dedicated nodes (within red dashed line) for IO and networking optimization. Data-Event processing is scaled on Cloud/Cluster (within green dashed line).

produce the final data-Events. In the case where all processing takes place over a local high-speed network, the first level of concatenation is unnecessary. In the case where data is fused from multiple remote locations (Data Center 1, 2, etc.) it is advantageous to perform the final concatenation at the site that provides the optimal volume/concentration of pre-sorted data. Completed data-Events are stored in a queue and can then be consumed by separate science processing workflows.

Data Predictor Service (PS): The PS is a service for predicting time, location and geometry of near-coincident data for given sensors. This service involves orbital simulation of spacecraft location using Simplified General Perturbations (SGP4), and modeling instrument data acquisition mode (e.g. cross-track operation from LEO or scanning from GEO platform).
Orchestrator: Deploys and configures services for user defined data processing, monitoring, exception handling and recovery processes. Orchestrator builds applications based on available services, and it designs and controls data-flow by linking all services together.
Data Staging Service (SS): The SS maps the initial data file into virtual memory, the data size is defined at the service configuration stage by user (Orchestrator). The SS is the only NAIADS system component that performs multi-file IO.
Data Reader Service (RS): Once data is staged, the RS starts a worker and sender threads. Worker thread reads the input buffer, filters and sorts based on required parameters, and fills the pre-sorted data queue. When the pre-sorted data queue is filled up to a defined water mark – sending-thread will start sending records to the Data Concentrator Service (CS).
Data Concentrator Service (CS): Does the partial event building, by concatenating records found at the specific data center. This service is used to reduce the volume of communication between data centers, and to optimize the network load.
Data Event Builder (EB): Is the final stage of complete data-Event building that generically combines data records from multiple data sources. Each data-Event represent a self-sufficient data object for algorithms defined by user (orchestrator) for entire application.
Science Services (S): Science algorithms applied to the fused data-Events, which are streamed to all available nodes and multi-threaded to all available cores in each node.
Data Streaming to Cloud/Cluster: Process scaling is achieved by data-Events streaming to multi-CPU system, computing Cloud or traditional Cluster.

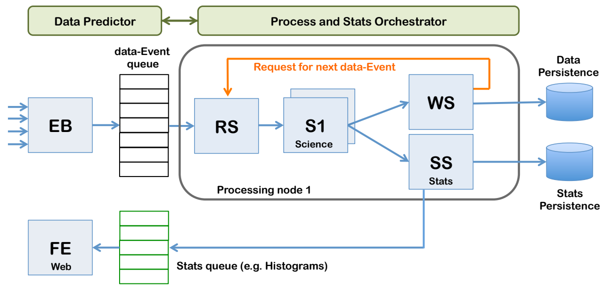

Figure 4: The NAIADS's workflow on one node: multi instrument observations data are streamed from the Event Builder (EB), science algorithms (S1) are multi-core scaled, Data Statistics is produced (SS) and can be returned to user's on-line interface. Output results are persisted (WS).

An example of NAIAD's workflow on one node is shown in Figure 4. The data IO and event building are decoupled from process scaling on compute Cloud, and controlled by dedicated Orchestrators. The Event Builder (EB) fills in-memory data-Event queue, Reader Service (RS) sends data envelope to a Science algorithm (S1), which passes the processed data to the Writer Service (WS), which stores the science output in a specified way. The S1 services are automatically scaled within each node of compute Cloud. This example also includes passing output of Science algorithm to Statistical Service (SS), and then streaming results (e.g. histograms) into statistics in-memory queue, which can be displayed via Front End (FE) web user interface or stored on disk.

The NAIADS transient message format (NcTransient) has been enhanced and more thoroughly integrated into the NAIADS prototype. NcTransient is based on the experimental NetCDF ncstream library, which defines an on-the-wire format for common data model (CDM) datasets. We started with ncstream, fixed bugs that prevented it from working with our datasets, and optimized it to handle our primary use case: fast serialization and de-serialization of CDM datasets to and from CLARA message payloads.

The NcTransient protocol was enhanced to be both write and read optimized. For read optimization, we only de-serialize variable data if the client actually reads it. We also store large variables in multiple "chunks" which may further limit de-serialization. For write optimization, NcTransient allows datasets or variables to be marked as read-only. This enables us to replace slow serialization with a fast byte array copy from existing de-serialization buffers.

IV. Data Test Case

Data fusion of SCIAMACHY Level-1 observations, MODIS/Terra Level-2 (Cloud, Aerosol, and Land) data products, and ECMWF re-analysis data is used for NAIADS demonstration and performance tests in compute Cloud and Cluster environment.

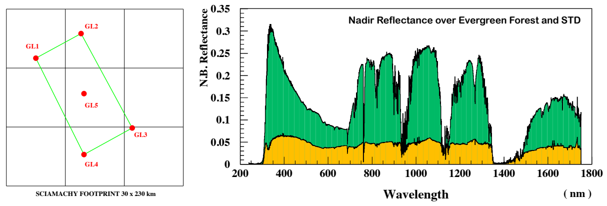

Figure 5: The Level-1 SCIAMACHY hyperspectral data for the test case 1: size and 5 geolocation points of near-nadir SCIAMACHY footprint (left), and SCIAMACHY-derived spectral reflectance over evergreen forest and its STD (right).

Essential features of the SCIAMACHY Level 1 nadir spectral data are illustrated in Figure 5:
◇ Nine years of near-nadir measurements;
◇ From 10 AM Sun-synch orbit;
◇ Swath 950 km (4 footprints in cross track);
◇ Footprint size 30 km by 230 km;
◇ 5 geo-location points per footprint;
◇ SCIAMACHY Level-1 data volume 2.2 TB.

We implemented test case 1 – merging the data from SCIAMACHY Level-1 data product and IGBP surface index. The IGBP surface index represent an auxiliary information provided in a static geo-grid with (1/6)° resolution. Data volume is 7 MB.

V. NAIADS Performance Tests

We have performed extensive test using Amazon Web Services (AWS) Cloud. The NAIADS AWS Cloud configurations included up to 16 compute nodes with 32 cores (c4.8xlarge compute optimized instances) and data stored at the WAS S3 bucket. The test data was staged on each node's local SSD storage.

A. Python Implementation

We began our hands-on evaluation with pCLARA, the Python implementation of the CLARA framework. Python is not an aggressively performant language by design and initial evaluations showed that pCLARA with the standard Cython interpreter lagged far behind the performance of a traditional C++ implementation. In an effort to match the performance of the traditional C++ implementation, a survey was performed of various higher performance Python dialects. Cython compiles Python code to C++ modules that are callable directly from Python, which results in a performance boost of about 30%. Alternatively, Cython can be used to allow Python to call external C++ code more or less directly, which makes the Cython solution as performant as the C++ code that it is calling, with the caveat that overhead is added whenever Python data types are converted to C++ data types and back. PyPy, a just in time interpreter, was also evaluated and was shown to have very encouraging performance, but was not evaluated with pClara because it does not appear to support all of the necessary libraries at this time. The Cython and Cython wrapping C++ implementations were ported back to pCLARA and were benchmarked.

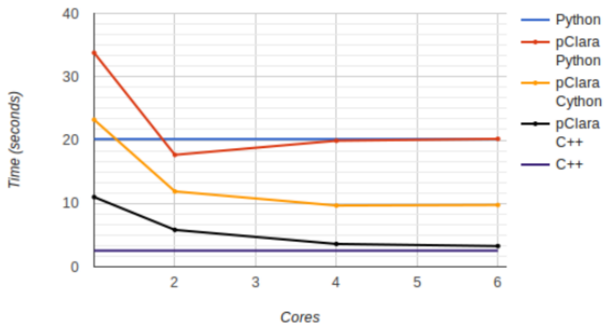

Figure 6: NAIADS performance with pCLARA easily outperforms a traditional Python solution and can be coerced to approach the performance of a traditional C++ solution.

The results, illustrated in Figure 6, show that pCLARA can easily out perform well optimized Python, and can come very close to the performance of C++. It is worth noting that portability is not an issue with any of the above options as they will run anywhere that Python and pCLARA will run, with the only understanding that Cython scripts need to be compiled natively before being run on different machines. For performance driven reasons, we will halt our evaluation of pCLARA as a computation framework and will reserve its use for code that does have strict time constraints, such as asynchronous monitoring and graphing.

B. Java Implementation: NAIADS/CLARA system has shown very good linear scalability. Every node process a file isolated from the others, and the network communication is only control messages between the orchestrator and the DPEs. Since the orchestrator has been optimized to communicate with nodes in parallel, the overhead of using many nodes to run files in parallel is minimum. The Figure 7 shows the linear scalability when using up to 16 nodes to process the same set of files.

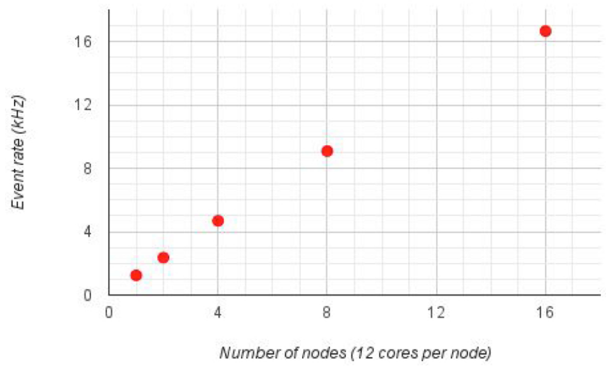

Figure 7: NAIADS' linear scalability when using up to 16 nodes for processing. Each node has 12 cores.

VI. Conclusions

We develop a conceptually novel framework for the Earth Science big data fusion. The NASA Information And Data System (NAIADS) software is integrated with CLARA framework and messaging based on the ØMQ socket library. The team has implemented the 1st data test case based on SCIAMACHY Level-1 hyperspectral data and IGBP Map. The NAIADS science algorithms have been implemented as framework services: data reading, data-event building, quality control, spectral re-sampling, and product sorting and persistence on disk storage. The implementation was tested with a number of Python dialects and Java. The NAIADS transient data format, NetCDF streaming, and IO services for NetCDF and HDF file formats were developed. Initial tests demonstrated linear multi-note and multi-core scaling. The Cloud configurations for performance benchmarking are designed, tested, and the process is automated.

Acknowledgement

We used SCIAMACHY Level-1 data, distributed by the European Space Agency. The NAIADS project is funded by the NASA ROSES 2014 – ESTO Advanced Information Systems Technology program.

References

[1] Wielicki, B. A., B. R. Barkstrom, E. F. Harrison, R. B. Lee, G. L. Smith, and J. E. Cooper, "Clouds and the earth's radiant energy system (CERES): An earth observing system experiment," *Bulletin of the American Meteorological Society*, 77(5), pp. 853 – 868, 1996.

[2] Loeb, N.G., W. Sun, W.F. Miller, K. Loukachine, and R. Davies: "Fusion of CERES, MISR and MODIS measurements for top-of-atmosphere radiative flux validation", *J. of Geophys. Research*, 111, D18209, doi:10.1029/2006JD007146, 2006.

[3] Loeb, N.G., B.A. Wielicki, W. Su, K. Loukachine, W. Sun, T. Wong, K.J. Priestley, G. Matthews, W.F. Miller, and R. Davies: "Multi-instrument comparison of top-of-atmosphere reflected solar radiation", *J. of Climate*, v. 20, No. 3, 575-591, 2007.

[4] Kato, S., S. Sun-Mack, W.F. Miller, F. Rose, Y. Chen, P. Minnis, and B.A. Wielicki, "Relationships among cloud occurrence frequency, overlap, and effective thickness derived from CALIPSO and CloudSat merged cloud vertical profiles," *J. of Geophysical Research*, v. 15, D00H28, doi:10.1029/2009JD012277, 2010.

[5] Thomas, E., "SOA: Principles of Service Design," *Prentice Hall*, ISBN: 0-13-234482-3, 2007.

[6] Hintjens, Peter, "ZeroMQ: Messaging for Many Applications," *O'Reilly Media*, pp. 484, March 2013.

[7] ØMQ Home Page, 2015, iMatix: *http://zeromq.org*

[8] Gyurjyan, V., D. Abbott, J. Carbonneau, G. Gilfoyle, D. Heddle, G. Heyes, S. Paul, C. Timmer, D. Weygand, E. Wolin, "CLARA: A Contemporary Approach to Physics Data Processing," *J. of Physics: International Conference on Computing in High Energy and Nuclear Physics*, Conference Series 331, doi:10.1088/1742-6596/331/3/032013, 2011.

[9] CLARA Framework Home Page at Jefferson Lab: *https://claraweb.jlab.org*