Chapman, David, Tyler A. Simon, Phuong Nguyen, and Milton Halem. “A Data Intensive Statistical Aggregation Engine: A Case Study for Gridded Climate Records.” In 2013 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum, 2157–64, 2013. https://doi.org/10.1109/IPDPSW.2013.87.

https://doi.org/10.1109/IPDPSW.2013.87

A Data Intensive Statistical Aggregation Engine

A Case Study for Gridded Climate Records

David Chapman, Tyler A. Simon, Phuong Nguyen, Milton Halem
Computer Science and Electrical Engineering Dept.
University of Maryland Baltimore County (UMBC)
{dchapm2, tsimo1, phuong3, halem}@umbc.edu

***Abstract*— Satellite derived climate instrument records are often highly structured and conform to the "Data-Cube" topology. However, data scales on the order of tens to hundreds of Terabytes make it more difficult to perform the rigorous statistical aggregation and analytics necessary to investigate how our climate is changing over time and space. It is especially cumbersome to supply the full derivation (provenance) of this analysis, as is increasingly required by scientific conferences and journals. In this paper, we address our approach toward the creation of a 55 Terabyte decadal record of Outgoing Longwave Spectrum (OLS) from the NASA Atmospheric Infrared Sounder (AIRS), and describe our open source data-intensive statistical aggregation engine "Gridderama" intended primarily for climate trend analysis, and may be applicable to other aggregation problems involving large structured datasets.**

Keywords—Gridderama; Workflow; Scientific; Aggregation; Big-data

I. INTRODUCTION

The resolution of scientific instruments has grown exponentially, and the latest generation of atmospheric satellites exhibit data volumes in the realm of tens of Terabytes to low Petabytes over years to decades of measurement. Climatologists need to summarize entire instrument volumes by combining appropriate physics with statistical aggregation operators to compute global "climate trends". These calculations must be performed carefully and a processing workflow is necessary to account for physical artifacts such as orbital striping, spatial noise, and drifts in calibration. Furthermore, the scientific community is pushing for the provenance and reproducibility of experiments requiring adequate visualization and documentation of inputs for publication [1].

The most popular statistical analysis tools in the geosciences include MATLAB, ENVI/IDL, NCAR Graphics, GRADS, R, Fortran, Excel and others. These tools provide a wide array of statistical functions, and many provide extensive visualization and Earth-plotting packages. However, they all require that the problem size must fit within RAM. Moreover, as instrument resolution improves, Big Data Analytics tools are becoming necessary in order to continue this analysis into the future. For example, in order to calculate decadal trends in the Outgoing Longwave Spectrum (OLS) from the NASA Atmospheric Infrared Sounder (AIRS), we needed to aggregate 55 Terabytes (compressed) of Level 1b observational data.

Our case study in the creation of a climate trending record from AIRS has provided us insight to many of the processing challenges unique to the analysis of highly structured yet geographically skewed data records. While data resolution has increased, the limits of human comprehension remain constant. "Climate Resolution" is the solution in which data volumes are progressively aggregated to very coarse summaries until the volume becomes small enough to be plotted and interpreted by expert domain scientists.

Gridderama is a statistical aggregation engine designed to maximize interoperability with highly structured instrument datasets. Typically these datasets are provided natively as dense hierarchical directory structures or "Data Products" in standard geosciences formats such as HDF or NetCDF. Gridderama attempts to treat this hierarchical data structure as a first class Large Array Storage (LAS) representation. At a conceptual level, each statistical aggregation step in Gridderama would take one data product directory (large array) as input, and produce another coarser data product directory (smaller array) as output that is also in a similar HDF format.

Gridderama also tracks provenance for scientific reproducibility, and goes the extra step to make this record accessible to others through a browsable web interface. Gridderama plots every data file from every intermediate data product and places these images in a similar hierarchical directory structure that can be browsed online. This is important because a single very large outlier can disrupt many statistical inferences. It can be very helpful to drill down and detect mistakes either in processing or in data, or to allow others to review and investigate your published work.

II. RELATED WORK

A. Data Active Archive Centers (DAACs)

The Data Active Archive Centers (DAACs) provide Tens of Petabytes of high quality NASA data products that are well documented and tested and infer a large number of physical phenomena from Earth observing instrument records [2]. The data products are hosted on FTP servers, and each product appears as a hierarchical directory structure on a single file system exhibiting data, metadata, timestamps, and often images. The processing algorithms are developed by authorized science teams and endure a rigorous review process including the establishment of an Algorithm Theoretical Basis Document (ATBD). Each science team produces a Product Generation Executable (PGE) which is a binary program (often

written in Fortran or C) that implements the documented physical algorithm, and the DAAC engineers develop the infrastructure to parallelize and scale the execution of this code within a production environment. The DAACs, however, are very large divisions and cover an enormous breadth of instruments and retrieval algorithms requiring many teams of scientists and engineers to maintain [2, 3]. For example, the MODIS Adaptive Processing System (MODAPS) produces Level 2-4 weather retrievals from the (Moderate Resolution Imaging Spectro-radiometer), and expands 140 GB/day of MODIS calibrated radiances to ~16 TB/day of retrieved physical parameters. MODAPS requires thousands of active processors and hundreds of compute nodes [3], and is one of many processing groups in the NASA EOSDIS effort spanning a geographically distributed network of DAACs [2]. In this paper, we focus on a much more constrained problem to perform climate aggregations in a way that is lightweight, flexible, but scalable for a generation of instrument data products toward the calculation of climate trends.

B. Scientific "Data-Cube" Databases

Array databases are a more high level (easier) way to manage/process remote sensing datasets to produce data product arrays, but their use in the scientific community exhibits a relatively short history (by database standards) of approximately 16 years. Beginning with RasDaMan in 1996 [4], there have been a number of systems that have pushed the boundaries toward the creation of agile Large Array Storage (LAS) systems for scientific applications. Most of these systems decompose arrays into uniform (or non-uniform) rectilinear "chunks", and use data parallelism methods to improve the speed of query. Furthermore, queries are executed using a data parallel model because of the known dimensionality of a large dense array. RasDaMan was the first important array query database, and was developed in 1996 along with an array query language RasQL [4]. RasDaMan was used in many scientific processing systems including the Open Geospatial Consortium (OGS) Web Coverage Processing Service (WCPS).

Ballegooij et al. (2005) further improved these methods with the Relational Array Mapping (RAM) framework. RAM supports array parallel indexing by means of "Query Fragmentation" patterns that break a query into multiple fragments that can be executed in parallel and aggregated to recreate the final result [5].

SciDB, presented by Rogers et al, (2010), is a new high volume query engine for the slicing and dicing of large array storage. SciDB is unique because it is not dependent upon any relational or traditional database system. Another innovation of SciDB is the use of a tunable "overlap" of blocks, that for certain queries such as stencil kernels, clustering, and other proximity information can improve performance by a factor of 2 [6]. By eliminating all dependence on the relational model, SciDB can process instrument records using block parallel data decompositions.

Planthaber et. al (2012) have applied the SciDB framework to the processing of the NASA Moderate Resolution Imaging Spectroradiometer (MODIS) Level 1b orbital instrument data. Their EarthDB system works because the instrument swath data (such as MODIS or AIRS) maps well to the "Data-Cube" representation, and can be statistically aggregated once it is loaded into the SciDB framework [7]. Unfortunately, the ETL phase remains a tedious bottleneck, and their current implementation requires a sub-optimal series of conversions between binary and text formats. Even if this pipeline were optimized, storing the data to a non-native format would require extra disks if both datasets were to be maintained for provenance reasons. Gridderama by comparison can load these datasets without translation because it works with data arrays as a hierarchical directory of flat HDF files, which is standard across Earth Observing instrument records including MODIS and AIRS.

C. Workflow Systems

Several workflow models and tools such as Trident and Pegasus are proposed for composing complex scientific workflows and scheduling these workflows on a distributed heterogeneous resource environment such as Grids, clusters, and more recently onto clouds [8, 9, 10]. Pegasus is a workflow management system which provides an interface to map workflow specification onto the Condor system. Then Condor allocates machines for the execution of jobs as instructed by Pegasus. DAGMan (Directed Acyclic Graph Manager) is a scheduler for Condor that manages the dependencies of Condor jobs by organizing jobs as directed acyclic graphs (DAGs) in which nodes represent jobs and edges indicate dependencies [8]. Kepler is a scientific workflow system, and provides a visual interface to create workflows. It supports software or functional libraries as actors [10]. These systems meet some of the challenges and issues in a scientific workflow and have been successfully applied in scientific applications across many science disciplines. However, these systems either do not support parallel execution on partitioned data or need to provide fully parallel execution handling. These systems also lack adequate support for data models, and high degrees of tolerance to failures for very large data analysis problems at scale. Our work focuses on building a workflow application (case study) and a general runtime system Gridderama providing fine grain parallel task executions with failure tolerance and provenance tracking.

D. Swift Parallel Scripting

Another related approach was that of the Swift parallel scripting language at U. Chicago (2007) [11, 12]. Their contribution was the development of big data analysis scripting language for the parallel execution of loosely coupled components. This is of great benefit because scientific applications often chain multiple processes together to perform a single calculation. Gridderama also utilizes loosely coupled components called Product Generation Executables (PGEs), and simulates loop level parallelism by means of a task scheduling mechanism, in which task requests are first appended to a Jobfile queue, and then a runtime environment is invoked which is responsible for scheduling and resource management of these tasks. Gridderama does not provide its own scripting language, but the API is relatively simple, and can be invoked from any language including regular shell script. We have included a bash script example of this syntax in Fig. 1 of Section III, showing how the AIRS monthly aggregate grids are computed from daily grids.

E. Provenance and reproducibility

Scientific replication is a necessary condition for the widespread acceptance of any new result. Jansey describes "The results and conclusions from one study obtained independently in another" as the "scientific gold standard" from which analysis is measured [13]. In practice, however, successful replication of another scientist's results is a very difficult problem even with published literature [14, 15]. An active community is necessary to prove a single climate trend, and we are very fortunate that efforts of GISS, Berkeley, Hadley center and others have replicated the global surface warming trend independently. Likewise in 2005, the UAH and RSS groups have agreed upon MSU derived lower-tropospheric warming trends within 0.001K/y of each other, an amazing result that required 15 years of scrutiny and countless publications to achieve.

The series of steps to derive a single result can be represented and stored as a "Provenance Graph". The provenance graph describes the data flow from its initial inputs, algorithms, and connectivity to produce the scientific result along with version numbers, timestamps, and other information [1]. Tilmes and Fleig (2008) have improved these methodologies using hashable provenance graphs for NASA's MODAPS and OMIDAPS processing centers [16]. The incorporation of provenance into Big Data Analytics infrastructure is also an upcoming area. Ikeda and Widom (2010) are tracking of provenance in common analytical workflows and released the PANDA big provenance system [17, 18]. The Swift language also has good support for provenance tracking [11, 12].

Gridderama goes one step beyond the tracking of provenance by attempting to make this information accessible through an interactive web catalog. By providing this information online, it is possible for one scientist to interactively review another's published results without needing to download Terabytes of data locally. Gridderama records the provenance information at a file granularity and provides a visualization of all intermediate data products as well as a detailed description that shows how the output results can be tracked in order to resolve bugs and errors in scientific results. Provenance is a highly effective way to track down errors in processing, and helps to ensure that scientific results can be repeated and scrutinized.

III. GRIDDERAMA

Gridderama is a statistical product aggregation engine for scientific calculations that shares some similarities between the DAACs as well as the LAS databases. As an aggregation engine, it provides a set of core statistical operators that take one (or multiple) data products as input and produce another product as output. Thus, Gridderama is capable of generating a large number of data products with a relatively small workflow. Each data product is stored in a hierarchical directory structure using flat .hdf datafiles. Netcdf support is currently in development. The use of standard geosciences formats for all intermediate LAS helps to eliminate the ETL barrier when interoperating with instrument records from the climate and weather domains.

Gridderama was designed for multivariate scientific array datasets stored and addressed in a Network Storage model (such as NFS, PVFS, or Luster), and provides a Gridderama Runtime Environment, GRE that makes use of a master/worker task pool to parallelize the computation of new data product arrays using statistical Product Generation Executables (PGEs).

We applied Gridderama to compute the decadal trends in Outgoing Longwave Spectra (OLS) from the NASA Atmospheric Infrared Sounder (AIRS), a hyperspectral IR instrument onboard the Aqua satellite platform, and demonstrated increased absorption of CO2, O3 and CH4 since 2003 [21, 22]. This is a new climate result that requires the aggregation of 55 Terabytes of Level1b radiances. Gridderama is open-source and, the entire computation of the AIRS 10 year climate record is available online for inspection and visualization at the following URL:

http://bluegrit.cs.umbc.edu/~dchapm2/airs-grid

The available data begins with a 55 TB mirror of the input AIRS Level 1b radiance granules from Sep 1st 2002 thru Aug 31st 2012, and ends with several 77 Kilobyte text-files showing the decadal trends under various processing decisions. We provide all of the gridded data products toward the generation of these trends. Gridderama also maintains provenance of the data flow for our portion of the processing chain by providing detailed log files at every step of the production, and by keeping track of all data inputs. Finally, we provide a web based catalog infrastructure to easily visualize or inspect many data files at once. Because manual inspection is such a powerful validation tool, every data file we have produced has already been manually inspected and is available for manual inspection by others.

Because the AIRS Level1b calibrated dataset is 55 Terabytes in size, we recommend a Linux cluster with 10+ Intel x64 nodes, and ~70 Terabytes of available network storage. The software dependencies are: gcc, bash, python, libhdf4, libjpeg, libz, libpng, ssh, gnuplot, and imagemagick. The total output size of our monthly gridded products is ~300GB (compressed) on a 2x2 degree Lat-Lon grid, and a single monthly array (ascending and descending) is ~24GB (compressed). We also compute daily 2x2 degree Lat-Lon gridded products that total ~9 Terabytes in output size. These products are available for quick inspection through our browser based "data catalog" service.

A. Parallel Execution

At its heart, a climate job involves aggregating a large array dataset to produce a coarser resolution version. Although a full data analysis requires a complex workflow, any individual task of this workflow is a simple data parallel operation. A "Product Generation Executable" (PGE) acts as a compute kernel for any operation to be parallelized [2, 3]. A data-parallel operation can be described using nested data parallel for-loops over the dimensions of the datasets. This process is embarrassingly parallel because each array element (file) is independent of the others and need not communicate. Gridderama provides a workflow scheduling runtime environment (Described in section IV) and tasks are scheduled

from a Jobfile queue is invoked using this data-parallel platform.

Figure 1 shows an example shell script (bash) to produce monthly aggregated AIRS OLS grids from daily AIRS OLS grids by using an exhaustive for loop. The PGE's are appended by means of a parallel Jobfile over the dataset which we have implemented by using a workqueue shell-script. Then the Gridderama Runtime Environment (GRE) is invoked to provide data-parallel execution. PGE programs can be written in any language and we can deploy PGEs to create hierarchies of images, graphs, or even html webpages all of which are created in parallel by using the same workqueue process.

```
1  # Queue up the jobfile "$q"
2  q="jobqueue.txt"
3  for y in (2002:2012) ; do                                  ┐
4      for m in (1:12) ; do                                   │
5          for t in day night ; do                            │
6              for pge in average median min max ; do         ├ array dimensions (for loop)
7                  inputFiles=$(                              │ "alternate solutions"
8                      for d in `dayom $m $y` ; do            │
9                          echo $dir/daily/$t/$y/$d.hdf       ├ array sub-dimensions (for loop)
10                     done                                   │
11                 )                                          │
12                 outputFile=$dir/monthly/$pge/$t/$y/$m.hdf  │
13                 echo $pge $outputFile bt 0 2378 $inputFiles >> $q ┘ enqueue command to jobfile $q
14             done
15         done
16     done
17 done
18 # Run the job in parallel across the cluster
19 arria < $q                                                 ┘ execute jobfile in parallel
```

Fig. 1. Shell script (bash) to aggregate "monthly" resolution grids from daily resolution grids via the "average, median, min and max" operators.

As can be seen in Fig. 1, the aggregation is written as nested for-loops over the dimensions that append to a jobfile "$q". The last line executes this job-file in parallel. The orange bracket represents the outer dimensions (year, month, day/night, and PGE). In this example, $pge represents one of four basic statistical operators (average, median, min and max), and the result is stored as a separate dimension. The red bracket shows the sub-dimension: day-of-month, which specifies the input dimensions. Lines 9 and 12, show how these dimensions are concatenated to index into the hierarchical directory structure. Line 19 executes this entire jobfile in parallel using the workqueue process.

IV. GRE Runtime Environment

The Gridderama Runtime Environment (GRE) provides for low level parallel task scheduling, and workflow processing. GRE is a parallel runtime system that is designed to create and map very fine grained tasks to computing resources. GRE selectively replicates tasks to circumvent faulty compute resources and network connections, as opposed to a synchronous checkpoint/restart of an entire application. This is done without the need for explicit message passing within the application. Our runtime environment uses data parallelism at the loop level to execute individual operations on multiple data sources, and remote compute resources. By gathering and using runtime performance metrics to both effectively scale and manage computational tasks and resources, this system demonstrates not only more effective utilization of computational resources, but improved application scalability, and performance. GRE provides runtime resource measurement analysis and strategically maps I/O and compute intensive tasks onto heterogeneous resources using a shared task, master-worker parallel programming paradigm as shown in Fig. 2. This is a reasonable approach because it is naturally fault tolerant and scales well once a problem has been broken down into independent subproblems. GRE incorporates a feedback mechanism that schedules and groups tasks together with task level fault tolerance through voting and replication and requiring no additional programming directives or the need for language explicit parallel constructs, such as Send() or Receive().

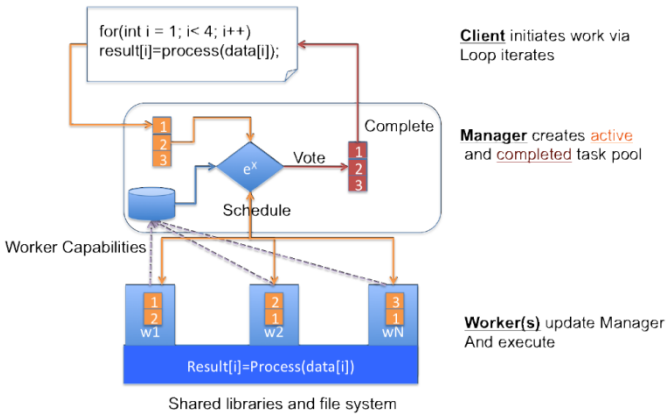

Fig. 2. GRE Shared task master worker paradigm

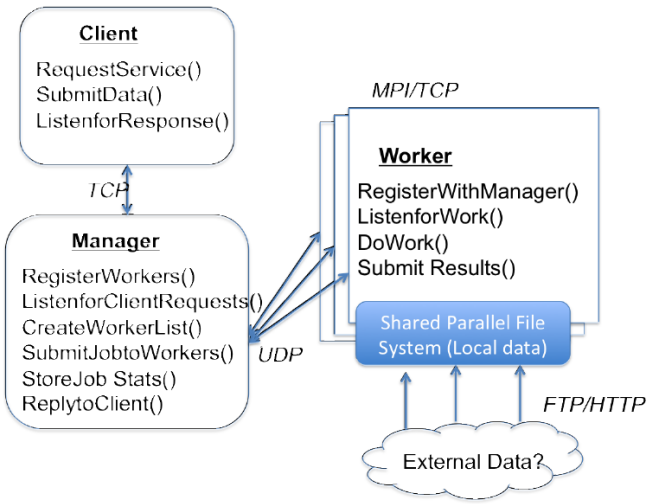

Fig. 3. GRE Process Overview

A. Architecture

The GRE system is composed of three fundamental and separate processes, each running as a concurrently executing thread and communicating over sockets using custom messaging interface over UDP. The primary communicating processes are a manager, client, and one or many workers. These are presented in Fig. 3, and their capabilities are discussed below. With regards to resiliency, periodic liveness checks are performed. These checks determine if an application is not performing as anticipated. If an application thread is detected as compromised during a liveness check, it will be destroyed and replaced using the uncompromised residual members of the group. This "hot-start" recovery mechanism [20] ensures that the newly recreated thread begins execution

from the most recent state rather than a state where the compromise occurred.

A. Client

The Client consists of an application programmer interface that profiles an application, breaks it down into subtasks and submits these tasks to the manager via a ServiceRequest(). The input from the client, which is typically run in a traditional cluster environment is sent to the manager for use by the HPC system via SubmitData(). The client then awaits a response from the manager with the complete results through ListenforResponse(). The entire workflow latency, from Client to Manager, Manager to Workers, Workers to Manager and Manager to Client is on the order of tens of milliseconds when no Worker execution time is measured.

B. Manager

When the Manager is started a registration message from any participating Worker process from RegisterWorkers() is received, These can be within the local cluster or on remote machines. The only requirements for either a Client or Worker are the ability to establish a socket connection to the manager. The Manager then listens for a ClientRequest() which is then served with a list of services for the Client to populate it's interface listing. Once the Client has entered its parameters and data location the message is sent to the Manager, who compiles a list of capable Workers via CreateWorkerList(). With this information the manager selects a subset of workers to perform the execution of the client task(s). The Manager logs each task in a local database, which can be used for task replication and voting. When the Manager receives a response from the pool of workers the manager checks for correctness is made and the results are sent on to the Client.

C. Worker

Each Worker sends a status message to the Manager that contains a "Characteristics" structure which outlines its computational capabilities, such as load average, number of cores, core frequency, memory in use, memory available, bandwidth to local storage, bandwidth to remote storage and or whether or not a GPGPU is present, this structure is presented in Listing 1. The Worker then polls a work queue and if it is listed for job execution it then executes the job and submits the results to the manager. The worker then updates an "Experience" structure as shown in Listing 2 with new information that can be used by the manager making task placement decisions.

```
typedef struct WORKER_CHARACTERISTICS
{
    int nprocs_per_node;
    int cpu_frequency;
    float memory_available;
    float total_memory_capacity;
    int gpu;
    int fpga;
} WORKER_CHARACTERISTICS
```

Listing 1. Worker Characteristics shared structure

```
typedef struct WORKER_EXPERIENCE
{
    float connection_throughput_local;
    float connection_throughput_remote;
    float memory_available;
    double loadavg;
    double numa_hits;
} WORKER_EXPERIENCE;
```

Listing 2. Worker Experience shared structure

V. Case Study: AIRS OLS Climate Trending

Polar orbiting weather instruments exhibit complicated scan geometries, and the data distribution is highly skewed. Fig. 4. shows the number of daytime AIRS measurements per gridcell for a daily and a monthly average. The Aqua satellite orbits in a sun synchronous manner at 805km altitude in near polar orbit. As a result, the poles are measured very frequently (14 times per day), whereas the equator is sampled much more sparsely (less than once per day). This orbital pattern appears as vertical "stripes" on a lat-lon projection (striping).

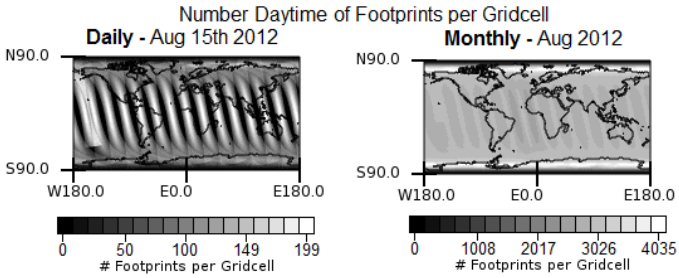

Fig. 4. Number of Measurements per Gridcell for a daily grid at 100 km Lat Lon resolution.

Our goal was to compute the decadal trends in Outgoing Longwave Spectra (OLS) from AIRS, a calculation that provides new insight to the Earth's radiative budget. Because AIRS has 2378 spectral channels spanning most of the Infrared spectrum, it provides the first direct measurement of changes in outgoing longwave atmospheric absorption by Greenhouse gasses including Carbon Dioxide, Ozone and Methane [21, 22, 23, 24]. The data intensive nature of this calculation required a parallel workflow to maintain exhaustive provenance and visualization. In this section we describe the workflow that was implemented to accomplish this goal.

A. AIRSC2X2 Processing Workflow

The workflow for these data products is represented with a single short bash shell script (appropriately labeled "run.sh"). Monthly products are produced from daily products, and daily products are produced from the Level 1b orbital granules. Fig. 5 shows a diagram of the data flow to produce all of the gridded data products. All of the daily and monthly products (square corners) exhibit a hierarchical directory structure. AIRS was inactive for two 3-week periods Oct 29th thru Nov 17th 2003, and Jan 9th thru Jan 26th 2010. Thus, this computation is performed twice: once without and once with fill data for the three week periods of AIRS inactivity. The "Filled" data products use daily climatology's to augment the

missing data days. Thus the filled daily products require both the daily images and the daily climatology's. This workflow is described in Figure 5.

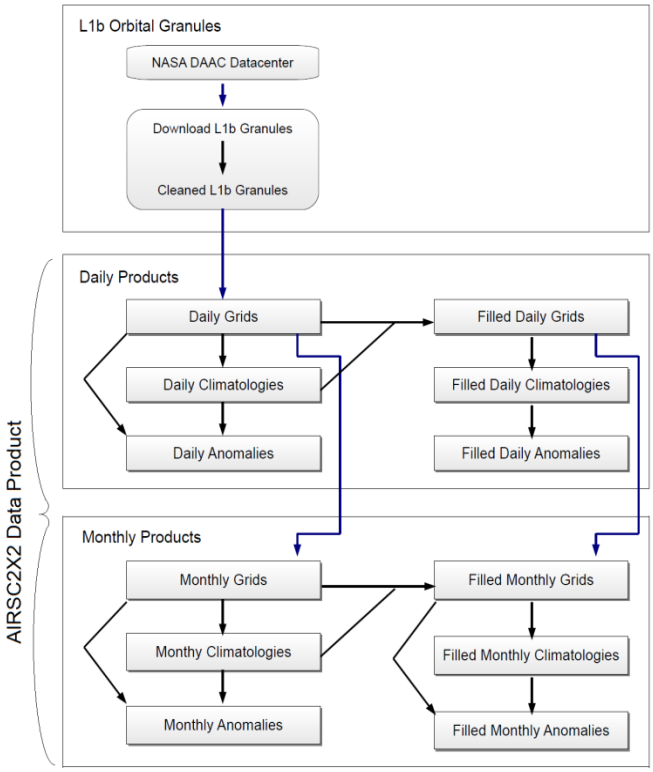

Fig. 5. Workflow graph of the AIRSC2X2 Data Product. Each box represents a separate LAS data product.

A "Climatology" (or inter-annual mean) is defined as the average of a particular day (or month) over the entire 10 year period. For instance, the September climatology would be the gridded average of all 10 Septembers within the dataset. An "Anomaly" is defined as the difference of a specific day (or month), and the climatology of that day (or month). For instance, the September'07 anomaly is the difference of the September'07 average minus the September climatology.

B. Monthly Average Dimensionality

The AIRSC2X2 "Data-Cube" representation is used to describe all of data products that were produced to perform the calculation of AIRS OLS trends. The first three dimensions: Longitude, Latitude, and Spectral Frequency represent a standard 3D geo-image. Additional dimensions include the day of year, the diurnal phase (day/night), and the aggregation operator (average, min, max, or count). Each "Data-Cube" product is a dense array, because these polar orbiting instruments are contiguously measuring and recording data at a uniformly sampled rate. Table 1 shows the dimensions of this "Data Cube". This is an 8-D array. All of our gridded data products thus far contain all (or a subset of) these 8 dimensions. Monthly anomalies would exhibit the same dimensions whether the computations include fill data or not. Therefore our workflow can compute cross-comparable geophysical trends from any of these intermediate data products (in Fig 5.), since they all exhibit the same 8D topology and structured representation shown in Table 1.

idx	Dimension	Typical Ranges	Implementation
8	Monthly Average Strategy	Avg(grid), Weight(foot), Max, Min	Directory
7	Diurnal Phase	Daytime(asc) OR Nighttime(dec)	Directory
6	Year	2002 to 2012	Directory
5	Day (or Month) of Year	1 to 366 OR 1 to 12	Filename
4	Daily Averaging Method	Avg Max Min OR Count	File Extension
3	Spectral Channel	1 to 2378	Internal
2	Latitude	1 to 90	Internal
1	Longitude	1 to 180	Internal

Table 1. Eight dimensional array representing the decadal gridded hyperspectral monthly averages within the AIRSC2X2 gridded data product.

We also see in Table 1, that the three spatio-spectral dimensions (Lat, Lon, Spectral Channel), are all implemented as "internal" (I. E. they reside within a single file), whereas the other 5 dimensions are all "external" to the file. These are encoded into the hierarchical directory structure. The "external" dimensions are slash delimited directories that use a short string to represent the column identifier. Some Product Generation Executable (PGEs) require multiple arrays within a single file, as this is supported by the HDF and NetCDF standards. For example, if we wish to perform a "weighted time series average" of daily average Brightness Temperature (BT) grids, then we would require both the "daily BT grids" as well as the "daily counts" (number of footprints per grid). For this reason we encoded these as multiple file extensions within the same array rather than splitting these records into multiple arrays. We also provide additional "txt" files to represent provenance and error logging for any a given data record.

An example of the Gridderama directory structure implementation is described in Illustration 1. All of our data products use the same directory structure. All of the data products reside within the "storage area", and are given a particular "product-name". Everything after the "product name" represents the "array" dimensions that are slash delimited.

Illustration 1. Example of a Gridderama file path for a daily gridcell average.

C. AIRSC2X2 Gridded Dimensionality

Some of the AIRSC2X2 gridded data products have slightly fewer dimensions (7 or 6). This makes these data products a subset of the total number of dimensions. The daily grids do not contain a *"monthly averaging strategy"* dimension, because they are not monthly averages (dim8). Climatologies do not contain the *"yearly"* dimension (dim6), because it is an inter-annual record. There is also a difference in the length of dim. 5 (Day or Month of year), between daily and monthly records because the dailies have 365 or 366 days whereas the monthlies have 12 months. These differences are described in Table 2.

	(1) Longitude	(2) Latitude	(3) Spectral Channel	(4) Daily Averaging Method	(5) Day (or Month) of Year	(6) Year	(7) Diurnal Phase	(8) Monthly Average Strategy
Monthly Average	x	x	x	x	x	x	x	x
Monthly Anomaly	x	x	x	x	x	x	x	x
Monthly Climatology	x	x	x	x	x		x	x
Daily Grid	x	x	x	x	x	x	x	
Daily Anomaly	x	x	x	x	x	x	x	
Daily Climatology	x	x	x	x	x		x	

Table 2. Major data products (vertical) vs. the 8 available dimensions (horizontal). X means that a dimension exists for that data product.

D. AIRS Regional Averages and Trends

The regional averages are computed from a multivariate array of tabular CSV formatted regional BT averages stored in a consistent text format also encoded as a hierarchical directory structure. These tabular arrays are computed for many zonal means. Table 3 shows a description of the regions for which tabular csv information is provided. These zones were selected to provide trend analysis over latitudinal belts and other standard geographic areas.

index	Region	S	N	W	E
1	Global	90S	90N	180W	180E
2	United_States	30N	50N	124W	70E
3	North_Polar	80N	90N	180W	180E
4	North_Mid_Latitude	20N	60N	180W	180E
5	Tropical	20S	20N	180W	180E
6	South_Mid_Latitude	60S	20S	180W	180E
7	Arctic	60N	90N	180W	180E
8	Antarctic	90S	60S	180W	180E
9	South_Polar	90S	80S	180W	180E
10	El_Nino_Index	18S	18N	144E	144W

Table 3. Regions selected for spectral trend graphs.

E. Data Catalogs

Gridderama provides a catalog system to browse images for all of the data, and allows users to inspect large portions of the data at once by displaying hundreds of images on a single webpage. This system allows instant point and click access to view any portion of the dataset. Our data catalog is a low latency web interface designed to allow the user to inspect a large portion of the data as quickly as possible essentially by splaying out slices of the "Data Cube". For this reason, we pre-computed all of the images available for inspection through the catalog, and these images are stored in a hierarchical directory structure. The data catalog is also a powerful "debugging" tool. It is very easy to accidentally produce large volumes of erroneous output. Extensive visualization from the data catalog system has been very helpful for us to detect obvious data problems often fixable using the provenance information and detailed logs provided at file granularity.

This data catalog system also provides the data user with the necessary empirical evidence to explore the data and form hypotheses. Figure 6 shows a screenshot of the catalog browsing system for a filled daily data product. We are displaying 365 images on a single page, by clicking the buttons on the "Navigate" panel we may walk through any part of the large data array instantly on click.

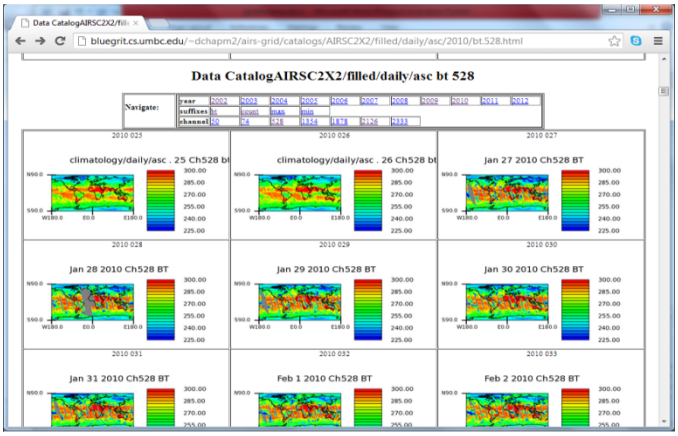

Fig. 6. Screenshot of the data catalog system for the ascending filled daily AIRSC2X2 product.

VI. Experimental Results

In order to evaluate the flexibility and scalability of our workload runtime environment, we developed a benchmark representative of the gridding workload. The benchmark performs a global monthly means calculation for 12 months of data. We have a total of 30 200MB files representing each month. We ran our experiments on a 10 node Linux Cluster, with each node consisting of dual quad-core Intel Xeon E5504 (Nehalem) processors running at 2.0 GHz, thus 8 compute cores per node. Each node had 26GB of main memory and all files were accesses from a single directory mounted over NFS. The network interconnect was Gigabit Ethernet.

We performed a strong scaling study to evaluate the performance of the GRE when adding nodes to the fixed workload benchmark. We ran a single manager process on a dedicated node and up to 9 worker processes. Each worker performed a single month global mean calculation, thus performing a read, summation and average on 30 200MB files over NFS.

The client process contained a loop that dispatched a task to the manager for each month. The GRE manager used round robin task placement to place each mean calculation on a single node, and wrapped around the available resources. Our performance metrics involved an average between five timings of the execution of the entire yearly workload from the client using the Unix time command.

Fig. 7 represents the scaling of the GRE system on the annual mean benchmark for up to 9 compute nodes. The x-axis is the number of nodes performing the concurrent execution of the benchmark, and the y-axis is the total workload runtime in seconds. The ideal curve demonstrates perfectly linear scaling. Our measurements show very good scaling, which is expected due to the simple parallelism

inherent in the data and computational layout. However, our measurements show performance flattens out after five nodes, this is due to bandwidth saturation for NFS. For GB Ethernet this is around 120 MB/s, which is reached when using 5 nodes 120MB/s=((360*200)/5)/120s, where 120s is the time to run the workload in parallel. Because we had 12 months, we had more than 2 jobs running on each node for experiments run below 6 nodes.

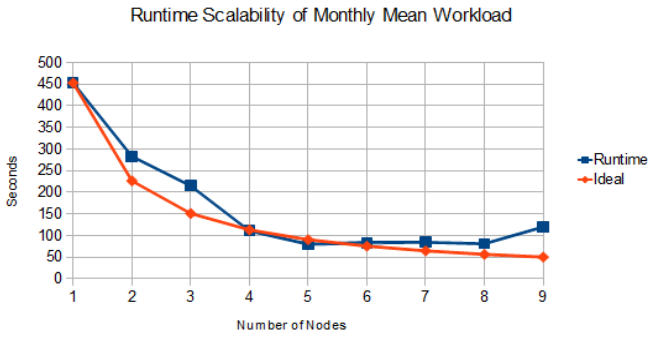

Fig. 7. Strong Scalability analysis of the Monthly Mean Workload from. Red line shows ideal scalability (by extrapolation).

VII. CONCLUSION

We present Gridderama, a statistical aggregation engine for large structured data records that is designed for out of the box compatibility with geoscience datasets. Further our case study and specific workflow implementation enable climate trending analysis of a 55 Terabyte decadal record of Outgoing Longwave Spectrum (OLS) from the NASA Atmospheric Infrared Sounder (AIRS). An important feature of Gridderama is its handling of hierarchical directory structures as a "Data Cube" representation for statistical aggregations. These aggregations can be executed in parallel with good scalability using the GRE runtime environment. One of the limitations of Gridderama is its requirement for IO Bandwidth to and from the filesystem and we were limited in our results by Gigabit Ethernet. As a future direction we plan to experiment with the scalability on clusters/clouds with faster interconnect to IO. Finally, we would like to extend Gridderama to a few more of the many highly structured scientific instrument records that adhere to the "Data cube" topology.

REFERENCES

[1] Moreau, Luc, Ben Clifford, Juliana Freire, Joe Futrelle, Yolanda Gil, Paul Groth, Natalia Kwasnikowska et al. "The open provenance model core specification (v1. 1)." Future Generation Computer Systems. Vol 27, iss. 6 pp. 743-756. 2011J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[2] H. Ramapriyan, J. Behnke, E. Sofinowski, D. Lowe, M. Esfandiari, "Evolution of the Earth Observing System Data and Information System (EOSDIS)". Proceedings of the 2010 Roadmap for Digital Preservation Interoperability Framework Workshop. ACM, 2010.K. Elissa, "Title of paper if known," unpublished.

[3] E. Masuoka, C. Tilmes, N. Devine, Ye. Gang, M. Tilmes, "Evolution of the MODIS science data processing system". Geoscience and Remote Sensing Symposium, 2001, IGARSS'01

[4] Baumann et. al., "The multidimensional database system RasDaMan." In Proc. of the SIGMOD Conf., pages 575{577,1998.

[5] Ballegooij et. al. Distribution rules for array database queries. In 16th. DEXA Conf., pages 55{64, 2005.

[6] Rogers et. al. Overview of SciDB: Large scale array storage, processing and analysis. In proc. of the SIGMOD Conf.2010.

[7] G. Planthaber, M. Stonebraker, J. Frew, "EarthDB: Scalable Analysis of MODIS Data using SciDB," 1st ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data, Redondo Beach, CA, Nov 2012

[8] Ewa Deelman, Gurmeet Singh, Mei-Hui Su et al. Pegasus: a Framework for Mapping Complex Scientific Workflows onto Distributed Systems. Scientific Programming Journal, Vol 13(3), pages 219-237, 2005.

[9] Trident: Scientific Workflow Workbench for Oceanography. http://www.microsoft.com/mscorp/tc/trident.mspx

[10] Barseghian, Derik and Altintas, et al ..., "Workflows and extensions to the Kepler scientific workflow system to support environmental sensor data access and analysis", Ecological Informatics. Volume 5. Pages: 42 - 50 2010

[11] Zhao Y., Hategan, M., Clifford, B., Foster, I., vonLaszewski, G., Raicu, I., Stef-Praun, T. and Wilde, M Swift: Fast, Reliable, Loosely Coupled Parallel Computation IEEE International Workshop on Scientific Workflows 2007.

[12] Swift provides a parallel scipting language Michael Wilde, Mihael Hategan, Justin M. Wozniak, Ben Clifford, Daniel S. Katz, Ian Foster Swift: A language for distributed parallel scripting Parallel Computing 2011.

[13] B. Jansy, G. Chin, L. Chong, S. Vignieri, "Again, and Again, and Again …," Editorial from Science Magazine, Vol 334, pp. 1225, 2011

[14] W. G. Dewakd, J. G. Thursby, R. G. Anderson, "Replication in Empirical Economics: The Journal of Money Credit and Banking Project," American Economic Review Vol 76, pp. 587-603, 1986

[15] S. Feigenbaum, D. Levy, "The market for (Ir)reproducible econometrics," Social Epistemology, Vol 7, pp. 215-32, 1993

[16] C. Tilmes, and A. Fleig. "Provenance tracking in an earth science data processing system." Provenance and Annotation of Data and Processes, pp. 221-228, 2008

[17] R. Ikeda and J. Widom. Panda: A System for Provenance and Data. IEEE Data Engineering Bulletin, Special Issue on Data Provenance, 33(3):42-49, September 2010.

[18] R. Ikeda, J. Cho, C. Fang, S. Salihoglu, S. Torikai, and J. Widom Provenance-Based Debugging and Drill-Down in Data-Oriented Workflows. To appear in Proceedings of the 28th International Conference on Data Engineering, Washington, DC, April 2012

[19] These guys are tracking provenance through online analytics, and show how it can be useful as a debugging tool. We want to do this with scientific analytics.

[20] S. Jajodia, C. D. McCollum, and P. Ammann, "Trusted recovery," Commun. ACM, vol. 42, pp. 71–75, July 1999.

[21] D. Chapman, "A Decadal Gridded Hyperspectarl Infrared Record for Climate. Sep 1st 2002 – Aug 31st 2012," Ph.D. dissertation, Dept. Comp. Sci. & Elect. Eng., Univ. Maryland Baltmire County (UMBC), Baltimore, MD, 2012

[22] P. Nguyen, D. Chapman, J. Avery, M. Halem, "A Near Fundamental Decadal Data Record of AIRS Infrared Brightness Temperatures", Int'l Geosciences and Remote Sensing Systems Conference, IGARSS, Munich Germany, July 20-27 2012

[23] R. Goody, R. Haskins, W. Abdou, L. Chen, "Detection of climate forcings using emission spectra," Earth Observation and Remote Sensing, 1995

[24] J. Harries, H. Brindley, P. Sagoo, R. Bantges. "Increases in greenhouse forcing inferred from the outgoing longwave radiation spectra of the Earth in 1970 and 1997." Letters to Nature, Vol. 410, iss. 6826, pp. 355-355