Enabling Earth Science Through Cloud Computing

Sean Hardman, Andres Riofrio, Khawaja Shams, Dana Freeborn, Paul Springer, Brian Chafin
Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Drive Pasadena, California 91109
{shardman, ariofrio, ksshams, danaf, pls, bchafin}@jpl.nasa.gov

Abstract—**Cloud Computing holds tremendous potential for missions across the National Aeronautics and Space Administration. Several flight missions are already benefiting from an investment in cloud computing for mission critical pipelines and services through faster processing time, higher availability, and drastically lower costs available on cloud systems. However, these processes do not currently extend to general scientific algorithms relevant to earth science missions. The members of the Airborne Cloud Computing Environment task at the Jet Propulsion Laboratory have worked closely with the Carbon in Arctic Reservoirs Vulnerability Experiment (CARVE) mission to integrate cloud computing into their science data processing pipeline. This paper details the efforts involved in deploying a science data system for the CARVE mission, evaluating and integrating cloud computing solutions with the system and porting their science algorithms for execution in a cloud environment.**

TABLE OF CONTENTS

1. INTRODUCTION 1
2. BACKGROUND 2
3. ARCHITECTURE 3
4. CONTRIBUTIONS 5
5. CONCLUSIONS 6
REFERENCES.....6
BIOGRAPHIES.....7
ACKNOWLEDGEMENTS.....8

1. INTRODUCTION

A compute cloud consists of a cluster of computers whose location and networking details have been abstracted or virtualized. This abstraction allows a compute cloud to be offered as a service of computing resources available at any given moment. A cloud computing service allows a customer to utilize these available resources (machines, storage, and messaging services, among others) on a pay-as-you-go basis. The initial cost for instantiating a compute cloud instance is minimal and depending on the provider, may be accomplished with little or no human interaction. The entire transaction is facilitated through an automated web interface or through an application programming interface. This allows the customer to use and pay for computing resources as soon as the need arises.

Cloud computing has been shown to offer tremendous potential for missions across the National Aeronautics and Space Administration (NASA). It has been used in flight operations with great success, offering faster processing time, higher availability, and drastically reduced costs. For example, cloud computing is being used by the Mars Science Laboratory (MSL) mission, among others, to tile and scale thousands of images for public consumption with more reliability, timeliness, and cost-effectiveness than previously possible. [1] Cloud computing has also been used by the Mars Exploration Rover (MER) mission for tactical planning. [2] However, the advantages of cloud computing have not been extended to general scientific algorithms relevant to earth science missions. Moreover, there are significant barriers to wide scale adoption of cloud computing across the earth science community that include: vendor lock-in, data transfer performance, security and compliance issues, as well as the lack of metrics to evaluate the suitability of cloud solutions.

The members of the Airborne Cloud Computing Environment (ACCE) task at the Jet Propulsion Laboratory (JPL) are working with the Carbon in Arctic Reservoirs Experiment (CARVE) mission to build and deploy a science data processing pipeline to process and manage their data products. Due to the processing demands of the Level 2 Full Physics algorithm, CARVE has decided to utilize a cloud computing solution to facilitate execution of this algorithm and generation of the associated products. In addition to providing the pipeline, ACCE is also providing the interface to the desired cloud provider.

Building upon previous work with distributed workflows, ACCE leverages the Polyphony framework developed at JPL [1] for interfacing with the cloud provider. This allows integration of our efforts into a general framework that can be used in missions across NASA, and allows CARVE to take advantage of the scalability, modularity, and resiliency that has already been integrated into Polyphony for numerous other missions. Polyphony supports hybrid workflows allowing machines running on any cloud, or even locally, to contribute to the pipeline processing capacity. This integrated solution can stream the data from machines at JPL as it becomes available, eliminating the need to have all of the data ready before processing begins. This is discussed further in the Architecture section.

In this paper, we detail our efforts of integrating cloud computing into the processing pipeline for the CARVE earth science mission. In particular, we briefly discuss the supporting mission and the earth science Level 2 Full Physics algorithm as well as introduce crucial cloud concepts in the Background section. In addition, we detail the characteristics of our solution and how we developed it at length in the Architecture section. Finally, we discuss two important reusable contributions that our project has made: the porting of the Level 2 Full Physics algorithm to the cloud, and the generation of base Amazon Machine Images

978-1-4577-0557-1/12/$26.00 ©2012 IEEE

(AMIs) for scientific computation, in the Contributions section.

2. BACKGROUND

This section provides some background information on the participants and components involved in the effort.

Airborne Cloud Computing Environment

ACCE is a reusable cloud-capable science data system for sub-orbital instruments providing process control and data management services at low-cost, and low-risk, with a predictable, scalable cost model. The ACCE task is currently in its third year of a three-year funded effort to improve the return on airborne missions. Low budget airborne missions are typically deprived of infrastructure required for timely processing. Because of this, cloud computing is a natural fit for such missions. The task has developed a common science data system capability for airborne instruments that encompasses the end-to-end lifecycle covering planning, provisioning of data system capabilities, and support for scientific analysis in order to improve the quality, cost effectiveness, and capabilities to enable new scientific discovery and research in earth observation.

The science data system is built on Apache's Object Oriented Data Technology (OODT) suite of components. [9] Although this task is focused on supporting airborne missions, the methods utilized and the resulting software system are based on work from previous efforts that supported the Orbiting Carbon Observatory and NPP Sounder PEATE missions [14] and are applicable to spaceborne missions whether they be earth-based or planetary missions. Along with the suite of pipeline enabling components, ACCE also offers a unified portal providing instrument teams increased mobility, being able to access data and information from wherever they are located. Figure 1 details the high-level architecture of the ACCE science data system.

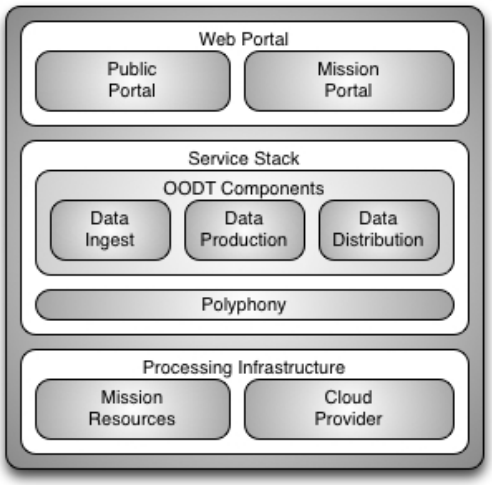

Figure 1: ACCE Architecture

Carbon in Arctic Reservoirs Vulnerability Experiment

Part of NASA's Earth System Science Pathfinder program, CARVE will collect the first simultaneous measurements of surface characteristics that control gas emissions and total atmospheric columns of important greenhouse gases, using an L-band spectrometer and a nadir-viewing Fourier transform spectrometer (FTS) aboard a Twin Otter aircraft flying over the Alaskan Arctic. Deployments, or flights, will occur during the spring, summer, and early fall in order to maximize the measurement opportunities. [3,4] The FTS instrument will produce over 700,000 soundings (spectrum measurements) over the life of the three-year mission. Each sounding is processed along with external data using the general Level 2 Full Physics algorithm to generate the total atmospheric columns of carbon oxide, carbon dioxide, and methane [5].

Level 2 Full Physics Algorithm

The Level 2 Full Physics (LP FP) algorithm is currently used across multiple earth science missions and tasks, including Orbiting Carbon Observatory (OCO) - 2 and Atmospheric CO2 Observations from Space (ACOS). [15] It is a converging algorithm that derives estimates of the column averaged atmospheric dry air mole fraction of certain greenhouse gases from the spectra returned from the FTS instrument. It is also used in the OCO - 2 mission to retrieve the atmospheric carbon dioxide dry air mole fraction from measurements [5].

The simplest description of the algorithm is that it runs a forward model that tries to predict the spectral values produced by the instrument for a given sounding. Some of the starting parameters for the model are known, and some have to be estimated using the best information available. The forward model itself uses other models, such as a solar model, and a radiative transfer model. The forward model's prediction is mathematically evaluated to determine how close it is to the actual values observed. If the difference is

above a certain threshold, input assumptions used by the forward model are tweaked, and the forward model is run again to produce a better prediction. This prediction iteration process continues until either the threshold is reached, or it is determined that the sequence will not converge. Once the predicted values are within the threshold of the observed values, the state of the model is assumed to be accurate enough to describe the contents of the column of air observed.

Cloud Computing As a Venue for Mission Data Processing

The amount of elasticity available to analysts, researchers, and operators in the cloud in a cost-effective manner make it a suitable venue for mission data processing. Traditionally, NASA missions procure all infrastructure required for data processing at least a year before they become operational. The fundamental issue with this practice is that it requires missions to predict the amount of capacity required. Over prediction typically leads to procurement of unnecessary resources and under prediction leads to processing delays. Meanwhile, accurate prediction is extremely challenging. Furthermore, during reprocessing, missions are typically confined to the capacity they have provisioned and have to wait for spare resources to finish reprocessing data. Most importantly, missions endure the risk of real-time processing falling behind in case of a major outage, which would make it extremely difficult to catch up.

Cloud computing offers a paradigm shift that enables missions to postpone infrastructure procurement until it goes operational. CARVE will be the first NASA mission to exercise this capability. CARVE goes operational in Spring of 2012, and we have not procured any processing infrastructure to handle the L2 FP processing. Furthermore, we are able to validate our algorithms in the cloud environment across a large cluster and then turn the machines off. Since cloud computing operates on a pay-as-you-go model, it obviates costs that missions incur after validating that their infrastructure is ready for operations. The CARVE data systems team did significant analysis of all available options before deciding to move forward with a cloud based model.

The cost to purchase and operate a cluster-based system capable of generating the Level 2 FTS products from these data was prohibitive. Through a concrete analysis, we determined that the cost of procurement and management of a cluster for the life of the mission would be $481,000. This cost does not include utilities and physical space that are paid for by burden funding. It is crucial to note that, since CARVE is an airborne mission with campaigns dispersed around the year, the processing for CARVE is extremely bursty. The pay-as-you-go nature of the cloud, as well as the idea that "using 1000 machines for 1 hour costs the same as using 1 machine for 1000 hours" [6], made cloud computing the solution of choice for CARVE data processing. Instead of buying hundreds of machines for a cluster system that will be underutilized, the mission will rent that same number of machines or more in the cloud for hours at a time at a reduced price. The cost for utilizing a cloud solution to generate the Level 2 FTS products was estimated at $196,000 for the life of the mission. This estimate was prepared over a year ago and was based on pricing information from Amazon Web Services at the time. Since then, although hardware prices have decreased, cloud computing prices have also decreased in a similar fashion.

ACCE and Polyphony are cloud provider agnostic. In the current design, the CARVE mission will utilize Amazon's Elastic Compute Cloud (EC2) for L2 FP processing. When launching a new machine on Amazon EC2, the operating system, software, configuration, and data on the machine can be stored in a virtual machine image. Within the EC2 environment, this image is referred to as an Amazon Machine Image (AMI). AMIs can be used to launch a machine from an image within Amazon, local installations of Eucalyptus [16], as well as NASA's Nebula cloud [17]. An AMI consists of a file system image coupled with a kernel. Developers have the ability to make an image from a pre-configured machine by a few simple clicks or a single API call. In a typical development scenario, a user configures a single machine with all of the required libraries and software. Once a single machine is ready, it can be imaged and the user can launch as many instances of the image as required to finish the task. Furthermore, users have the choice between a variety of instance types that vary in storage, RAM, CPU capacity and network characteristics. Since the L2 FP algorithm is primarily CPU-bound and has modest memory requirements, instances that offer highest CPU capacity work best for CARVE. Specifically, the Cluster Compute Units work really well for CARVE because they offer single tenancy, 16 hyper-threaded cores, and 23GB of RAM [8]. This particular instance has superfluous RAM as L2 FP only requires 8GB of RAM to fully subscribe the CPU capacity. In an ideal world, CARVE would have access to a cheaper instance type that offers less RAM.

3. ARCHITECTURE

CARVE will become operational in spring of 2012. Being a low-cost airborne mission, we do not yet have instrument data from the actual mission. Nonetheless, we were able to validate our approach using data from the ACOS task, which was obtained from the Greenhouse Gases Observing Satellite (GOSAT) [15]. The Repeat Cycle benchmark is used to test compatibility of the L2 FP software across versions of the algorithm, and it consists of 3668 soundings (measurements) from GOSAT. During a repeat cycle, the satellite orbits the earth multiple times (44, in this case) across multiple days (3, in this case) [10] so that the position of the satellite above the earth at the end of the cycle is the same as it was at the beginning of the cycle. Depending on the complexity of the data and time required for the data to converge, each sounding takes between 10 to 90 minutes to be processed on a single core of a High-CPU Amazon EC2 machine. Our architecture employs a multi-process

approach that attempts to fully subscribe all the available cores on the machine.

Polyphony

The Polyphony framework developed at JPL enables developers to succinctly express complex workflows, which are then orchestrated across large clusters of nodes that can be running in a cloud environment, traditional data center, or a combination [1]. We used a branch of Polyphony that makes use of Amazon Simple Workflow (SWF) to coordinate the orchestration of tasks. While our in-house solution was appropriate, using SWF allows us to focus on the challenges specific to the mission and not on the general problems related to workflow management. In this section, we will outline the core components of a workflow and analyze the workflow we have designed for L2 FP.

A Polyphony workflow consists of a collection of tasks (implemented as SWF activities) as well as sub-workflows. Polyphony tasks can optionally have dependencies. For instance, a developer may want to express that task B can only run after task A has completed. Furthermore, tasks can also have dependencies on the types of nodes they run. For example, some tasks may require a 64-bit machine, while others may be able to run on any available node. These node-type dependencies are implemented as SWF capabilities. Sub-workflows behave as a single task to its parent workflow, but can be composed of multiple tasks. In this case, the sub-workflow is considered finished once all of its tasks are finished.

In the L2 FP pipeline, both kinds of dependencies are extensively employed. In the CARVE architecture, data arrives at JPL, and are made available to internal nodes via a Network File System (NFS). At this point, a Polyphony workflow is started to handle the new data. The Polyphony workflow consists of three steps highlighted in Figure 2. First, input data must be uploaded to a location where they are accessible by the cloud nodes. To streamline this process, Polyphony's optimized pipeline to upload data to Amazon Simple Storage Service (S3) is used to transfer data in parallel. Since the NFS servers where the input data resides is behind the institutional firewall, the upload tasks must be assigned to Polyphony nodes at JPL. Fortunately, Polyphony allows tasks to be routed to nodes with specific capabilities. The second step is to process the data. Ideally, most (if not all) of the processing tasks will be routed to EC2 Polyphony nodes, since we will only have modest compute nodes at JPL. This would allow us to keep JPL nodes available for other internal resources and tasks that may be more suitable for these nodes. After the data are processed, the third step is to download the results to JPL. This task also requires routing to JPL-specific nodes. To clarify, this workflow requires the three steps to be executed in order. For instance, the processing step cannot be scheduled until the upload step has finished.

The processing task has also undergone extensive design. For instance, we made a deliberate choice of using a distributed file system for scalability purposes. In a cluster setup with a head-node that serves the files over NFS, scalability issues arise as soon as the cluster goes beyond a small number of nodes. This is mainly due to the fact that each node generates traffic for the head node, which quickly becomes the bottleneck. By using S3, we obviate this bottleneck as each object may be stored on a different S3 node, and S3 automatically balances loads across replicas of objects under heavy read-loads. In the processing task of the workflow, each node starts by downloading the input data from S3, executes the L2 FP processor, and then uploads the results to S3. This approach allows the next task in the workflow (the download task) to retrieve the data from S3 and place it on a JPL machine.

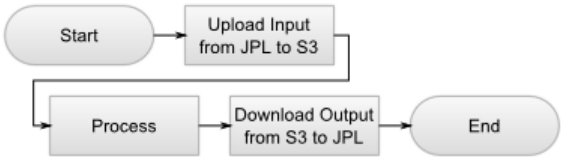

Figure 2: Sounding Workflow

When running the Repeat Cycle benchmark as highlighted in Figure 3, we can create workflows that consist of a group of pseudo-randomly selected soundings; this provides a size-adjustable, repeatable, and representative sample on which to conduct tests. This selection happens at JPL nodes, since they have direct access to all the Repeat Cycle data. Then, a sub-workflow is started for each sounding. After every sub-workflow is finished, the workflow gathers statistics that can then be compared to previous in-house results.

Typically, we want to run a few hundred soundings per test. However, SWF imposes a limit of 128 sub-workflows for each parent workflow. This means that we can run up to 128 sounding sub-workflows per test. We work around this limitation by using dummy "batching" sub-workflows that run up to 128 sounding sub-workflows. This way, we can run up to 128^2 = 16,384 sounding sub-workflows. This is illustrated in Figure 4.

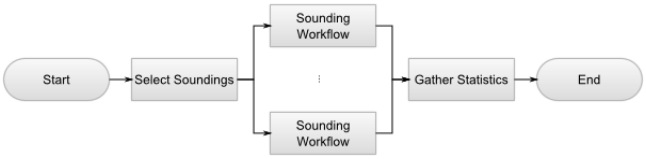

Figure 3: Repeat Cycle Benchmark Workflow (without batching)

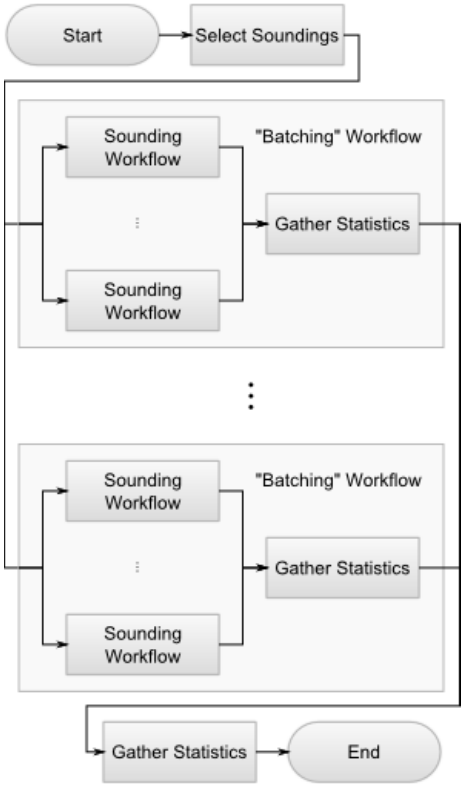

Figure 4: Repeat Cycle Benchmark Workflow (with batching)

The process of building the Polyphony client and starting the EC2 nodes in the workflow is illustrated in Figure 5. To start a large number of EC2 nodes, we first create an AMI that serves as a template for the nodes. Once the instances are running, the workflow can be started. Finally, when the workflow finishes, the instances can be terminated.

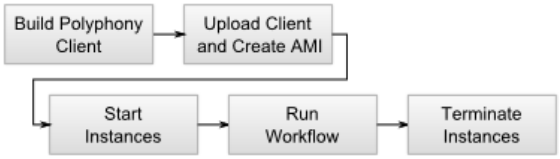

Figure 5: Development Process

Data Distribution

Due to security concerns, worker machines outside of JPL, such as those in Amazon EC2, should not be able to access machines inside the JPL network. This means that the input and output data must be read from and written to a storage container outside of JPL, which acts as a broker between the machines in our workflow.

We used Amazon Simple Storage Service (S3) as the storage container. Amazon S3 is a distributed storage service that can be used to store and retrieve data. The user of Amazon S3 does not need to manage these machines, but needs only to use the REST-style interface provided by Amazon [11]. Another reason to use S3 is that EC2 machines are able to connect to S3 over the local network as these services are collocated in the same set of data centers. Additionally, we are able to gain very high throughput to S3 thanks to internally built capabilities to optimize transfer to S3 that mostly rely on parallel transfer streams to and from S3.

To further amortize the time delay of transferring files from S3 to EC2, we included commonly used data—for example, the Absorption Coefficient (ABSCO) data tables—in the AMI that was used to launch the machines. We did this by using Secure Shell (SSH) to upload the data to the machine from which we then generated the AMI.

Security/Integrity

To limit our exposure, our solution does not require any machine outside of JPL to have access to machines inside of JPL. All communication between JPL and the Amazon Web Services (AWS) including EC2, S3, and SWF is done using Secure Sockets Layer (SSL) [12] to ensure the privacy and integrity of our data and messages when they are in transit. Furthermore, access to cloud services is restricted through private, user-specific credentials (access keys, X.509 certificates and SSH key pairs) [7] [13]. Access keys are generated by Amazon and downloaded by the developer using SSL. They are used to make secure requests to S3, SWF, and EC2. X.509 certificates are used to make secure requests to EC2, and consist of a certificate and a private key. SSH key pairs are used to access EC2 machines, and consist of a public key and a private key. Using either kind of authentication, the service has access to the certificate or public key, and the trusted developer uses the private key to authenticate to the service. The X.509 certificate-pair or SSH key pair can be generated by the developer, who then uploads the certificate or public key to AWS.

Lastly, we store our data privately in S3, so that proper authentication is required to retrieve or update the data. Our data are always protected by Amazon's security policies and/or with SSL encryption whenever they leave JPL's network [18].

4. CONTRIBUTIONS

Porting L2 FP - Earth Science Impact

As part of our effort to integrate cloud computing with CARVE's data processing pipeline, we ported the Level 2 Full Physics algorithm to run in a cloud environment. This was complicated by the fact that scientific code is generally intended to run in a certain machine or cluster of machines, and so might depend on idiosyncratic libraries and settings. In particular, the L2 FP algorithm required a specific version of the GNU Compiler Collection (GCC), 4.5.1, to compile properly. It also required the user to set a few environmental variables to point to the data that comes with the L2 FP distribution, which is required for testing the correctness of the installation.

While JPL uses the Intel Fortran compiler in its internal clusters, the license terms are not compatible with bursty scenarios where machines might only exist for a few hours at a time. Instead, we used the gfortran compiler, which is part of GCC. The combination of different hardware and different compiler gave slightly different results in the output.

We also automated the installation of dependencies installable using YUM and the compilation of GCC and L2 FP in a series of shell scripts, so that we are able to upgrade to new versions of L2 FP in a matter of hours with little manual intervention. After we installed a version of L2 FP on an EC2 machine, we created a reusable AMI from the machine. These techniques significantly sped up development.

Base AMIs for Scientific Computation

While using shell scripting to automate the installation of software and data on a cloud machine is useful, upgrading an element of the system can be time consuming, as all the elements must be rebuilt. For example, when creating a new AMI with a new version of L2 FP, our script also rebuilt GCC, an operation that can be time consuming. Instead of starting from a vanilla installation of Linux, we built a base AMI with our custom installation of all dependencies for the L2 FP algorithm. As mentioned earlier, this base AMI streamlines the update process.

We created another base AMI with these and other dependencies, such as FFTW and gfortran, that are common to numerous other science processing tasks. We believe that this base AMI will continue to streamline the process of creating AMIs for other missions.

5. CONCLUSIONS

This paper details the efforts of the ACCE team members and the CARVE mission to develop and deploy a science data processing pipeline to process and manage the data from multiple flights over the Alaskan Arctic. The core of the science data system consists of tailored components from the Apache OODT suite. Although the system supports multiple data streams, the focus of this paper is on the Fourier transform spectrometer data stream, the associated Level 2 Full Physics algorithm and its port and execution in a cloud computing environment. Using the Polyphony orchestration framework, we developed a hybrid system that runs in the JPL and Amazon EC2 environments, is scalable, efficient, and resilient to individual machine failures. We also describe two components of the system that can be readily used by other missions: the ported L2 FP algorithm, and the construction of base AMIs for scientific computation.

At this point in the development effort, the integration between the OODT component suite and Polyphony is limited. Planned work for this fiscal year includes integration between OODT and Polyphony, expansion of ACCE portal capabilities and support of CARVE operations.

REFERENCES

[1] Khawaja S. Shams, Dr. Mark W. Powell, Tom M. Crockett, Jeffrey S. Norris, Ryan Rossi, Tom Soderstrom. "Polyphony: A Workflow Orchestration Framework for Cloud Computing." CCGRID, 2010, pp. 606-611.

[2] Joseph C. Joswig, Khawaja S. Shams. "Redefining Tactical Operations for MER using Cloud Computing." 2011. Aerospace Conference, 2011, pp.1-7, 5-12.

[3] "CARVE Mission (NASA Earth Exchange)." Internet: https://c3.nasa.gov/nex/projects/899/, Nov. 22, 2010 [Nov. 3, 2011].

[4] "CARVE Mission (NASA Science)." Internet: http://science.nasa.gov/missions/carve/, [Nov. 3, 2011].

[5] David Crisp et al. "OCO (Orbiting Carbon Observatory) - 2, Level 2 Full Physics Retrieval Algorithm Theoretical Basis." Internet: http://disc.sci.gsfc.nasa.gov/acdisc/documentation/OCO-2_L2_FP_ATBD_v1_rev4_Nov10.pdf, Nov. 10, 2010 [Nov. 3, 2011].

[6] Michael Armbrust et al. "Above the Clouds: A Berkeley View of Cloud computing." Technical Report No. UCB/EECS-2009-28, University of California at Berkley, USA. Feb. 10, 2009.

[7] "Amazon Elastic Compute Cloud User Guide." Internet: http://docs.amazonwebservices.com/AWSEC2/2011-07-15/UserGuide/, July 15, 2011 [Nov. 3, 2011].

[8] "Amazon Elastic Compute Cloud (Amazon EC2)." Internet: http://aws.amazon.com/ec2/, [Nov. 3, 2011].

[9] "Apache OODT." Internet: http://oodt.apache.org/, [Nov. 3, 2011].

[10] "Thermal And Near infrared Sensor for carbon Observation (TANSO) On board the Greenhouse gasses Observing SATellite (GOSAT), Research Announcement, Appendix A." Internet: http://www.gosat.nies.go.jp/eng/proposal/download/GOSAT_RA_A_en.pdf, [Nov. 3, 2011].

[11] "Amazon Simple Storage Service (Amazon S3)." Internet: http://aws.amazon.com/s3/, [Nov. 3, 2011].

[12] "Making Secure Requests to Amazon Web Services." Internet: http://aws.amazon.com/articles/1928, Jul. 6, 2009 [Nov. 3, 2011].

[13] "Amazon Web Services: Overview of Security Processes." Internet: http://aws.amazon.com/articles/1697, Dec. 5, 2008 [Nov. 3, 2011].

[14] Chris A. Mattmann, Dana Freeborn, Dan Crichton, Brian Foster, Andrew Hart, David Woollard, Sean Hardman, Paul Ramirez, Sean Kelly, Albert Y. Chang, Charles E. Miller. "A Reusable Process Control System Framework for the Orbiting Carbon Observatory and NPP Sounder PEATE missions". SMC-IT, 2009, pp. 165-172.

[15] "ACOS — GES DISC: Goddard Earth Sciences, Data & Information Services Center." Internet: http://disc.sci.gsfc.nasa.gov/acdisc/documentation/ACOS.shtml, Oct. 19, 2011 [Nov. 3, 2011].

[16] Daniel Nurmi, Rich Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff, and Dmitrii Zagorodnov. "The Eucalyptus Open-Source Cloud-Computing System." CCGRID, 2009, pp. 124-131.

[17] "Cloud Computing Case Examples." Internet: http://nebula.nasa.gov/media/uploads/nasa-nebula-in-action.pdf, [Nov. 3, 2011].

[18] "AWS Security and Compliance Center." Internet: http://aws.amazon.com/security/ [Dec. 13, 2011]

BIOGRAPHIES

Sean Hardman *is a software systems architect in the Data Management Systems and Technologies Group at JPL. He has a B.S. in Computer Information Systems from California State Polytechnic University, Pomona. He has developed and led development on several science data management systems at JPL and is a contributing software developer for the Object Oriented Data Technology (OODT) project, which is the first NASA software project to reach top-level status at the Apache Software Foundation.*

Khawaja Shams *is a member of the Operations Planning Software (OPS) Lab at the NASA Jet Propulsion Laboratory. Khawaja develops software that contributes to the operations of a variety of robotic assets including ground, airborne, and waterborne robots, as well as robots on Mars. He leads a variety of software projects, and he serves as the Cognizant Engineer of server side components for the Activity Planning and Sequencing Subsystem (APSS) for the Mars Science Laboratory. Khawaja works closely with the Office of the CIO at JPL to co-lead the efforts to securely deliver the benefits of cloud computing to missions across NASA. He serves as an advisor on the CIO Technology Advisory Board (CTAB) at JPL. Khawaja is currently pursing a PhD in robotics at USC, holds a Masters in Computer Science from Cornell, and a Bachelors in Computer Science from UC San Diego.*

Dana Freeborn *is the Technical Group Supervisor of the Data Management Systems and Technologies Group at JPL. The focus of her group is on providing data management solutions for science missions, enterprise applications and archive systems at a low cost with low risk. She holds a B.S. in Mathematics and her interests are in data modeling, database design, distributed architectures and the engineering, development and use of common reusable software frameworks for data- and compute-intensive systems. Dana is currently the Task Manager for ACCE.*

Andres Riofrio *is a sophomore at University of California, Santa Barbara. Andres spent the summer of 2011 as a research intern at NASA Jet Propulsion Laboratory, where he investigated novel approaches to integrate cloud computing for earth science missions. Andres has received a research fellowship from JPL to continue his research during the school year. Andres is actively conducting this research at the RACE Lab at UCSB under the advisement of Dr. Chandra Krintz.*

Paul Springer *is a senior researcher in JPL's High Capability Computing and Modeling group. He has very broad experience in developing and modifying applications to run on parallel hardware, as well as writing simulators of advanced parallel computers, such as the IBM BG/L and advanced processor-in- memory architectures. He has spent most of the past two years working with a TILE64TM development system, modifying applications to run on it, and analyzing their performance. He has filed 20 NASA new technology reports, and authored or co-authored over 20 papers, as well as a book.*

Brian Chafin *is a senior systems engineer in the Instrument Software and Science Data Systems Section at JPL. He has a Masters Degree in Systems Architecture and Engineering from the University of Southern California and a Bachelors Degree in Engineering and Applied Science from the California Institute of Technology. Brian is the Ground Data System Lead and Systems Engineer for the CARVE mission.*

ACKNOWLEDGEMENTS

The research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

Additionally, the authors of this document acknowledge Daniel Crichton, Emily Law, Elizabeth Kay-Im and the Project Engineering Management Council (PEMC) from the Jet Propulsion Laboratory for their support, leadership and funding of the Airborne Cloud Computing Environment.