Evaluating Cloud Computing in the NASA DESDynI Ground Data System

John J. Tran[1,2], Luca Cinquini[1], Chris A. Mattmann[1,2], Paul A. Zimdars[1], David T. Cuddy[1], Kon S. Leung[1], Oh-Ig Kwoun[1], Dan Crichton[1], Dana Freeborn[1]

[1]Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109 USA

[2]Computer Science Department
Univ. of Southern California
941 W. 37th Place
Los Angeles, CA 90089 USA

john.j.tran@jpl.nasa.gov

ABSTRACT

The NASA Deformation, Ecosystem Structure and Dynamics of Ice (DESDynI) mission is a first-of-breed endeavor that will fundamentally change the paradigm by which Earth Science data systems at NASA are built. DESDynI is evaluating a distributed architecture where expert science nodes around the country all engage in some form of mission processing and data archiving. This is compared to the traditional NASA Earth Science missions where the science investigator-led processing is typically centralized. What's more, DESDynI is poised to profoundly increase the amount of data collection and processing well into the 5 terabyte/day and tens of thousands of job range, both of which comprise a tremendous challenge to DESDynI's proposed distributed data system architecture. In this paper, we report on a set of architectural trade studies and benchmarks meant to inform the DESDynI mission and the broader community of the impacts of these unprecedented requirements. In particular, we evaluate the benefits of cloud computing and its integration with our existing NASA ground data system software called Apache OODT. The preliminary conclusions of our study suggest that the use of the cloud and OODT together synergistically form an effective, efficient and extensible combination that can meet the challenges of NASA science missions requiring DESDynI-like data collection and processing volumes at reduced costs.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Design

Keywords

NASA, DESDynI, OODT, Cloud Computing, Benchmark

1. INTRODUCTION

The NASA DESDynI mission [7] is a tier-1 National Research Council (NRC) Earth Science Decadal Survey [4] mission focused on measuring the deformation, eco-system structure, and dynamics of ice. DESDynI leverages a radar and a lidar instrument combination to produce interferometry at high resolution, that is combined into deformation maps, and other higher order science data files (called "products") for use in identifying land surface changes and hazards, and in understanding Earth's climate [7].

This proposed architecture is in stark contrast to the centrally managed NASA science investigator-led processing systems (or SIPS) of most current Earth Science missions. DESDynI's ground data system will leverage the NASA Jet Propulsion Laboratory (JPL) architecture and associated implementation infrastructure called the Process Control System or PCS. PCS is becoming a common terminology in the NASA Earth Science mission world and it refers to the configured and deployed set of software components from the Apache Object Oriented Data Technology (OODT) framework [9], and its associated Catalog and Archive (CAS) infrastructure [10]. To date, OODT's support of Earth Science missions has traditionally focused on supporting centralized a processing approach which includes NASA's Atmospheric Carbon Observations from Space (ACOS), Orbiting Carbon Observatory (OCO2) [10] and Soil Moisture Active Passive (SMAP) [15] missions, and the NPOESS Preparatory Project's Sounder PEATE [10] mission.

At JPL, we have begun a study evaluating the architectural benefits and tradeoffs of the proposed DESDynI science data system approach. Our study has three high level goals/objectives: (1) obtaining benchmarking and results in the area of job throughput, metadata extraction time, ingestion time, and processing item; (2) evaluating the PCS's ability to support DESDynI data pipelines; and (3) evaluating the PCS as deployed for DESDynI in the context of cloud computing. There is a growing interest in the adoption of cloud computing for scientific software applications [16, 8, 6] due to the cloud's elasticity and scalability, two properties that traditionally escape the rigidity of existing NASA science data system software.

The remainder of the paper is organized as follows. Section 2 discusses the background and related work on DESDynI, science data processing systems, Apache OODT and on PCS. Section 3 describes the DESDynI data system architecture. Section 4 explains our experiment configuration and objectives. Section 5 analyzes the result of our experiment and discusses our findings. Finally, section 6 summarizes our experience and points to future work.

2. BACKGROUND AND RELATED WORK

We provide a brief background on the mission objectives for DESDynI which suggest cloud computing as a viable option to meet some of its challenging demands. Some background and related efforts in evaluating cloud computing are then described. We round out the section with an explanation of Apache OODT, and the JPL Process Control System architecture as applied to the DESDynI mission.

2.1 DESDynI

DESDynI is one of four tier-1 Earth Decadal Survey Missions identified in the National Research Council (NRC) Earth Science Decadal Survey [4]. It is the second decadal mission assigned for execution at NASA JPL, following the Soil Moisture Active Passive (SMAP) mission that started in 2008.

DESDynI is five-year L-band InSAR and Multibeam Lidar mission [1] with objectives to:

1. Determine the likelihood of earthquakes, volcanic eruptions, and landslides.
2. Predict the response of ice sheets to climate change and impact on sea level.
3. Characterize the effects of changing climate and land use on species habitats and carbon budget.
4. Understand the behavior of subsurface reservoirs

DESDynI is currently undergoing its pre-mission study and Mission Concept Review (MCR) preparation and is projected to start in 2011. Preliminary analysis of DESDynI's mission objectives points to tremendous data volume challenges. At the moment, DESDynI's radar expects to collect upwards of 4.9TB (terabytes) of raw radar signal data a day (likely $\tilde{1}$6 TB processed data), which in turn will generate approximately some 16PB (petabytes) of data products per year. DESDynI's nominal processing and data ingest cycle make it a strong candidate for the compute and storage services provided by cloud computing. We will provide some context on the subject of cloud computing below.

2.2 Cloud Computing

Cloud computing is the terminology used to refer to canonical services for data processing and storage whose scalability and elasticity are dictated by predictable costing models and levels of service [8]. For the purposes of our work on DESDynI, we are looking at the cloud as a provider for data storage scale-up, during nominal daily processing or reprocessing campaigns, and in addition, as a set of extra CPUs and cores that could be leveraged to speed up DESDynI data pipelines and job throughput.

There are a number of parallel research efforts exploring the performance and effectiveness of cloud computing. These include Cumulus studies, where Vrable et al. examined performance and overhead cost for thin- and thick-cloud layers as a storage and backup system [14]. In in other architecture studies, Vogels studied the offsetting tradeoffs between consistency and availability at Amazon's cloud [12]. Voicu et al. [13] suggested that the cloud is an extension or rework of the Grid middleware infrastructure. Our own work within DESDynI is geared towards evaluating the cloud as a low level platform to increase DESDynI's job processing scalability, and its data storage capacity, at low costs.

In the next section, we will provide background information regarding the underlying software platform that DESDynI will leverage to perform job execution and data storage and ingestion: Apache OODT [9].

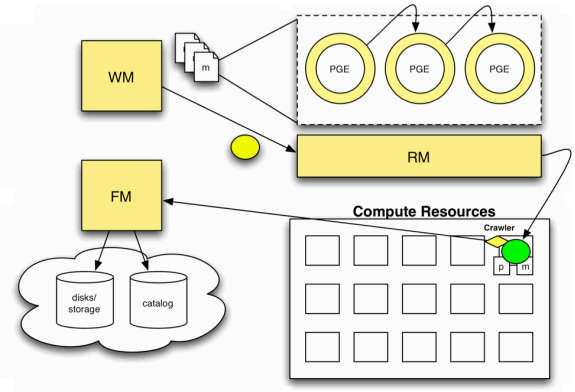


Figure 1: The PCS canonical processing architecture. FM stands for "file manager", WM for "workflow manager", RM for "resource manager", p for "product", m for "metadata". The yellow enclosing "egg" around the PGEs is CAS-PGE, the PCS "wrapper", which feeds the PGE the necessary information from the FM, WM and RM

2.3 Apache OODT and PCS

Apache OODT [9] is a software framework that provides sets of components useful in two fundamental areas: (1) information integration; and (2) data processing and computation.

OODT information integration components include the `Profile Server`, responsible for metadata retrieval and resource discovery; the `Product Server`, responsible for data product delivery; and the `Query Server`, which combines the Profile and Product Server to locate data resources, and then (semi-)automatically package up data and deliver it back to interested users. We do not focus on these components in this paper. We point the interested reader to [9] for a more complete description.

OODT data processing and retrieval components center around the `Catalog and Archive Service`, or CAS component. CAS provides a `file management service` for cataloging data product metadata, a `workflow management service` for managing jobs that operate on data products made available by the file manager, and a `resource management service` used to allocate workflow jobs onto computational resources. The core CAS services are depicted in Figure 1.

Several client frameworks communicate with the CAS services, including a `crawler framework` used to automatically identify and ingest files and metadata into the `file management service`; a `pushpull` service for obtaining remote content and handing it off to the `crawler`; and finally a

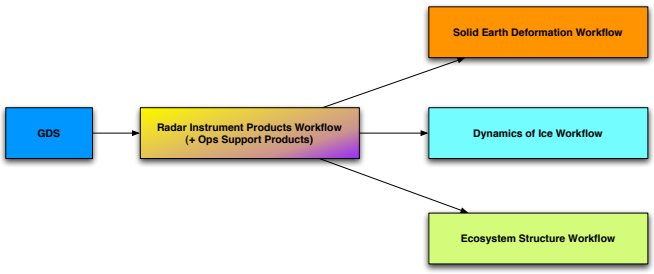


Figure 2: Schematic representation of DESDynI workflow

workflow task wrapper (called `CAS-PGE`) for allowing workflow tasks to leverage and communicate with CAS services to execute data processing jobs. The unique combination of CAS services as deployed for a mission has been dubbed the JPL Process Control System, or PCS architecture [10]. DESDynI is using the PCS to implement its geographically distributed science data system platform.

3. DESDYNI ARCHITECTURE

In this section we discuss the DESDynI science data system architecture, focusing in on its constituent PCS components.

3.1 Processing Architecture

It is currently envisioned that DESDynI will produce 21 different data products from processing of the original data feed received from the Ground Data System (GDS). The overall DESDynI data processing workflow can be schematically divided into 4 separate sub-workflows, as shown in Figure 2.

The Radar Workflow generates radar instrument products and operation support products upon receiving of raw (also called `Level 0`) data[1] from the science data system. The Solid Earth Deformation, Dynamics of Ice, and Ecosystem Structure Workflows, generate Level 3 science products once the appropriate radar instrument products have been generated.

Each data product is generated by a Product Generation Executable (PGE), which is a set of coordinated tasks that can be collectively described as reading some input data, running some scientific processing algorithm, and writing output data. PGEs are combined in sequence or in parallel to realize the overall data processing workflow(s). Typically higher level products are obtained by processing one or more lower level products, so in order to run they must rely on their input products to be available, either because all PGEs run on the same machine, or because the necessary products have been transferred to the required location.

A critical open question for DESDynI is whether the whole data processing workflow should be run on a single system, or the sub-workflows be executed on separate systems that are physically located close to the science teams for the various disciplines. As described later in this paper, part of our study consisted in trying to provide an empirical answer to this question by benchmarking a sample workflow on two different data processing architectures.

In the next section, we will describe the particular experiments that we performed on DESDynI's data system architecture.

[1]NASA follows an incremental "level" system to describe its science data product files. Level 0 is the raw data; Level 1 is spatially/temporally referenced; and Level 2 and above is ready for public consumption.

4. EVALUATION

The overall objectives of our benchmark experiments were two-fold: first, to study how the PCS software stack can be deployed flexibly onto a cloud environment (we chose the Amazon cloud service infrastructure); second, to study the performance impact of the PCS system in a stand alone system versus a cloud environment configuration. To do so, we measured the data processing pipeline throughput along two dimensions: the workflow and data transfer overheads when migrating DESDynI's data pipelines to the cloud.

4.1 Experimental Setup

We decided to use the existing infrastructure and support services provided by Amazon EC2 services for the DESDynI workflow. This included using the Amazon AWS EC2 developer toolkit which allowed us to custom build our own AMI (Amazon Machine Image) and propagate it between multiple Amazon regions and availability zones. We started by deploying a pre-built Amazon AMI based on CentOS 5 64bit. This is a publicly available image that can be used and customized by any third party group and which is provided by Amazon for unlimited use and without licensing fees. We initially installed a single Amazon EC2 instance based on the following specifications:

1. Type: m2.4xlarge
2. Memory: 68.4 GB of memory
3. CPU: 26 EC2 Compute Units (8 virtual cores @ 3.25 EC2 Compute Units Each)
4. DISK: 1690 GB of instance local storage (non-EBS)
5. I/O Performance: High

This instance type provided us with enough I/O performance to test and deploy the DESDynI software stack in a distributed topology. Our initial deployment was performed in the Amazon-WEST (Nothern California) region which allowed us to use Internet2 (I2 - CENIC) between JPL and Amazon EC2 based resources, as shown in left middle/upper portion of Figure 3. This default route provided us with a low latency, high bandwith link which was not available between NASA JPL and AMAZON EAST (Northern Virgina) as shown in the middle-right portion of Figure 3. We leveraged exisiting tools contained within the Amazon EC2 developer toolkit to bundle a new private AMI which contained all of our DESDynI configurations and our PCS software stack. We then moved our new AMI to our private Amazon S3 (Simple Storage Service) bucket located in Amazon WEST which provided us with a quick deployment path.

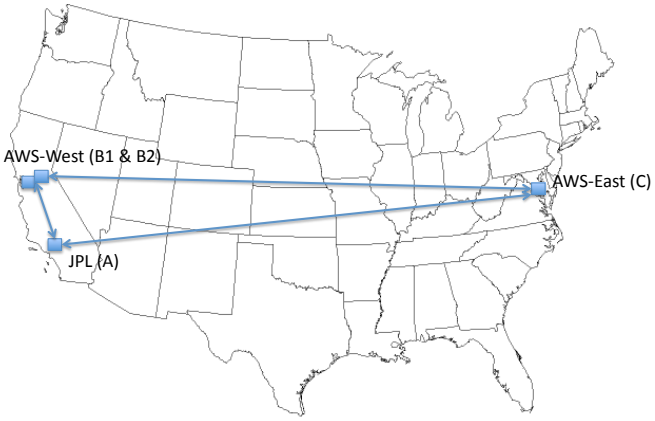


Figure 3: Map configuration for experiment

Unfortunately, the AMI build process unfortunately is all executed on the command line, since no Graphical User Interface (GUI) is provided by Amazon to make the process more user friendly. On the other hand, the advantage of using the command line is that we know exactly what is being built, how it is configured, and how our software is deployed from start to finish. We also encountered issues with how Amazon segregates regions and availability zones. We noticed that many things would work in one region but would not work in a different region such as our AMI, network layout, or Amazon EBS (Elastic Block Storage) mount points. We had to manually migrate all of these different areas between regions which was time consuming and the process was not intuitive. Eventually, after some trial and error, we managed to successfully demonstrate that by using Amazon AWS cloud services we could easily deploy PCS fully configured and ready for immediate use. The final product allowed us to deploy our DESDynI workflow in parallel by leveraging the versatility and power of the cloud.

4.2 Benchmarking the DESDynI Pipeline

As mentioned earlier, the objectives of our study (recall section 1) were to research the tradeoffs of different possible architectures for the future DESDynI data processing pipeline. To this purpose, we conducted an experiment to benchmark the execution time of a comparable DESDynI workflow called ROIPAC. ROIPAC is a demo workflow composed of 4 Product Generation Executables (PGE), which represent a partial path within the complete DESDynI pipeline shown previously in Figure 2. Each ROIPAC PGE read the input data from the previous PGE (or the simulated data feed for the first PGE), executed a real science analysis algorithm (which was considerably I/O and CPU intensive), and produced a data product composed of one or more files. We compared the total execution time when the same number of ROIPAC workflows was run sequentially on a single Amazon EC2 cloud server, and when it was run on a distributed topology composed of three identical Amazon EC2 servers, each of which setup to process a specific PGE.

The two data processing pipelines for this study were setup using CAS-PGE to integrate ROIPAC into the CAS `workflow management service`, and then ran using `workflow management service` on our Amazon cloud instance. In the

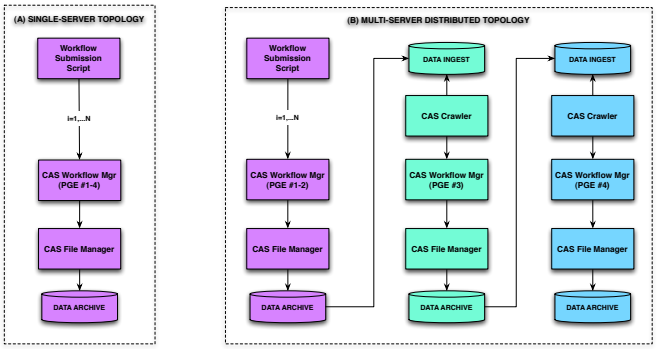


Figure 4: Architecture of the two topologies used to benchmark the DESDynI ROIPAC workflow

first topology (see Figure 4 A), a batch script was used to submit a workflow composed of the full set of 4 ROIPAC PGEs to the `workflow management service`, which took care of running them sequentially and passing the generated data products to the `file management service` for indexing and archiving. In the second topology (see Figure 4 B), processing was distributed across 3 servers, and on each server the `workflow management service` was configured to run a different workflow composed of one or two PGEs only, followed by a data transfer task to the next server in the pipeline. In order to maximize the overall throughput, special consideration was given to the following two aspects of the architecture:

1. Minimize the data transfer between servers, so that it becomes a negligible factor in the overall time cost. By employing high-speed data transfer protocols such as GridFTP over a fast network as the internal Amazon cloud, the PGE on one server executed while the data output for the next PGE was being transferred from the previous server in the pipeline.
2. Employ a data product detection and notification service, such that the PGE on one server can be started as soon as its needed has arrived from the previous server. Such a service is provided by the `crawler framework` which can be setup to monitor ingestion of data in a given directory, and send a message to the `workflow manager` to start execution.

Below, we discuss our experimental setup to evaluate how different data movement technologies are affected by the cloud environment. Because of DESDynI's data volume requirements, the movement of data was another important area to pay attention to.

4.3 Benchmarking Data Transfers

In this section, we detail a set of data movement experiments that we performed on the Amazon Elastic Compute Cloud (EC2) and its Simple Storage Service (S3) cloud computing facilities. Our goal was to evaluate modern data movement technologies for their potential use in the DESDynI ground data system. The first step in the study was to select the data movement technologies that we would evaluate. We chose:

FTP – the baseline, most commonly used data transfer protocol of the past 30 years.

SCP – a secure data transfer protocol based on SSH, used to baseline our study.

bbFTP – similar to GridFTP, but constructed by CNRS in France, it is optiized for large files transfer protocol. It leverages a similar strategy to GridFTP in terms of parallelizing TCP/IP transfers, but it is somewhat easier to install since it is bundled as a standalone product [2].

GridFTP – the flagship data movement product of the Globus Toolkit, a parallelized, configurable alternative to FTP that saturates underlying TCP/IP networks [3].

UDT – the reliable UDP-based transfer protocol exploits the UDP protocol (unreliable, asynchronous data delivery) to send files around [5].

It should be noted that we considered testing UFTP, but because of its nature (multicast transfer) the underlining communication paradigm fits into those applications that are typically gather/scatter and relatively short-distance. In our case this technology is not applicable. A preliminary experiment using point-to-point mode immediately eliminated UFTP from consideration, as the transfer speed was orders of magnitude slower than any other protocol described above.

Amazon has two main data centers in the U.S. as detailed earlier and shown in Figure 3. The first step in performing our transfers was setting up an Amazon image with the necessary data movement software, and then duplicating that image at AMAZON WEST (B1 & B2 or simply B) and AMAZON EAST (C), so that we could test transfers from JPL (A) to B, from A to C and from B to C.

We benchmarked data transfers to and in between cloud servers using binary NetCDF files [11] of approximately highly compressed 1 GB and 10 GB sizes. Future DESDynI missions will likely use either NetCDF files or HDF equivalent. Thanks to their data format and API-level similarity, for the purposes of this experiment, they are interchangeable.

In general, we noticed that transfer speeds to and in between cloud servers varies considerably probably because of the concurrent network use of other projects. When transferring data within the cloud, explicitly using the EC2 internal network seemed to consistently give much higher rates than when generically specifying transfers over the public network. The network used in the transfer was determined by the choice of public or private IP address for the destination server.

By nature, this experiment is highly repetitive and sometimes error-prone. As such, we put together a suite of scripting tools to assist with the various benchmarks. We also noted that the majority of setup time revolved around configuring firewall and port security policies. None of the conducted experiments was accompanied by any kernel tuning, e.g. adjustment of the transfer window or TCP/IP buffer size. We believe that, while these experiments are samples of a stock configuration, they are more accurately representative of a typical science data system user in a cloud-to-cloud or cluster-to-cloud environment.

5. RESULTS

Overall, the results from our experiment are in line with our expectations. This section describes our findings and draws some conclusions on tradeoffs that researchers and developers should consider when migrating their science data software to a cloud computing environment.

5.1 DESDynI Pipeline Results

A single ROIPAC workflow composed of 4 PGEs was measured to take approximately an hour when run on a single EC2 Amazon server. We measured the total execution time when a sequence of 5 such ROIPAC workflows were run on that server, versus the case they were run on the distributed topology, with data transferred from one server to another using the internal EC2 network and the GridFTP protocol [3].

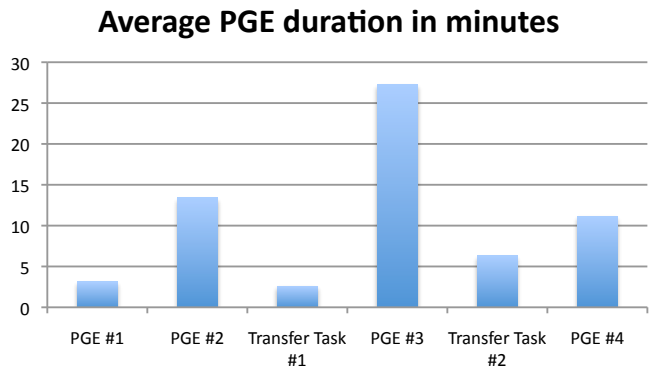


Figure 5: Duration of Roipac PGEs and data transfer tasks, in minutes

Figure 5 reports the average time for the 4 ROIPAC PGEs and the two transfer tasks. It can be seen that the longest PGE was the PGE #3, which took approximately 30 minutes, and that the two transfer tasks took approximately 3 and 6 minutes from the second and third servers, respectively. It must be noted that the transfer times reported in this study represent a high-end estimate of what would be required in a real production environment, because the entire PGE directory tree (input and output) was transferred to the next server (as opposed to only the files used in the next PGE).

Figure 6 shows the total execution time for the 5 ROIPAC workflows when run on the standalone topology (first bar in the graph) versus on the distributed topology (last three bars in the graph). Because on the distributed topology the PGEs can be run in parallel, the total elapsed clock time for the 5 workflows was measured to be 196 minutes (from 20:12 to 23:28), which is considerably lower than the time measured when all PGEs are run on a single server: 306 minutes (from 16:31 to 21:37).

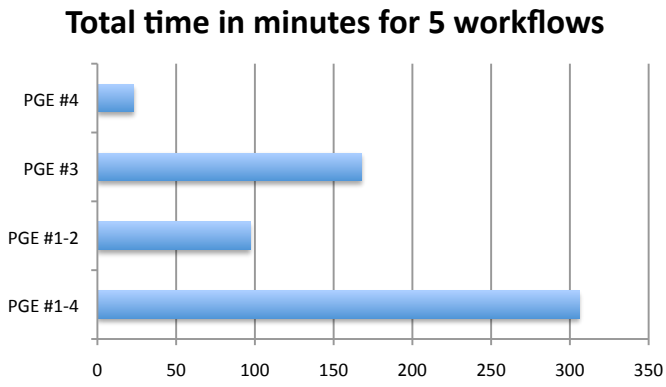


Figure 6: Total duration of 5 Desdyni workflows executed sequentially, for the two topologies setup in the experiment

5.2 Data Transfer Benchmark Results

Figures 7 and 8 summarize our findings with respect to the file transfer benchmark.

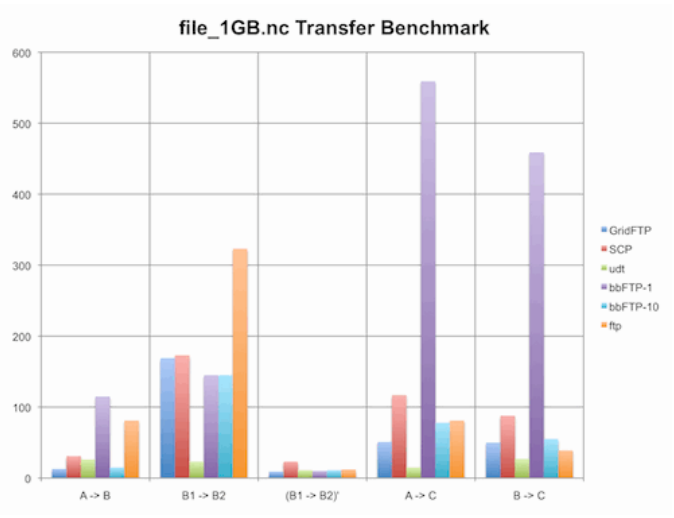
For transfer of both file sizes, 1GB and 10GB, the udt and GridFTP transfers consistently give best performance independent of network configuration. FTP and bbFTP yield the worst transfer performance. Internal network transfer at AWS-West, as expected yields the best performance, yet external interfaces between two nodes in the same cloud yields far inferior performance. This observation implies that geographical proximity alone does not guarantee speed gain. In some, but not all cases, we see 10x performance gains when we increase the number of pipes for bbFTP transfer.

When comparing transfer time between 1GB file and 10GB files, we observed that transfer time is not necessarily linear. In some cases, e.g. FTP, the transfer of a larger data file seemed to give an overall better transfer rate value. We believe this can be attributed to the streaming nature of the transfer technology.

Based on our experimental observations, it is clear that if security and data integrity are to be maximized, GridFTP gives the best overall performance, though it should be noted that setting up GridFTP is a fairly involved task, and by far consumed most of our integration effort with the DESDynI science data system, and with PCS. On the other hand, if speed and ease of implementation are to be maximized (at the expense of accompanying middleware infrastructure support), UDT is the appropriate choice for data volumes similar to those of DESDynI-like data systems.

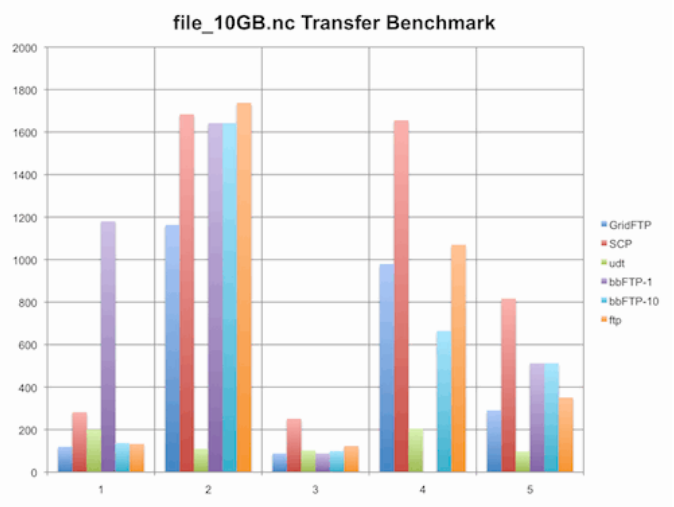
6. CONCLUSION AND FUTURE WORK

NASA's DESDynI mission is set to produce unprecedented data volumes (5 TB/day), require comparatively large (10s of thousands) numbers of processing jobs per day, and do so with a geographically distributed architecture. At NASA's Jet Propulsion Laboratory, we have undertaken an architectural study to investigate our options for meeting these stringent Earth Science mission requirements. Specifically we are evaluating the use of cloud computing technologies and environments, along with JPL's PCS data management



file_1GB.nc	JPL to AWS-west1	AWS-west1 to AWS-west2 (external)	AWS-west1 to AWS-west2 (internal)	JPL to AWS-east	AWS-west to AWS-east
GridFTP	13-28	169	9-14	51-57	50
scp	31	173	23	117-191	88-100
udt	26	23	11	15	27
bbFTP (1 connection)	115	145	10	559	459
bbFTP (10 connections)	15	145	11	78	55
ftp	81	323	12	81	39

Figure 7: Transfer time (in secs) for 1 GB NetCDF file



file_10GB.nc	JPL to AWS-west1	AWS-west1 to AWS-west2 (external)	AWS-west1 to AWS-west2 (internal)	JPL to AWS-east	AWS-west to AWS-east
GridFTP	120-180	1663	87-112	979-1252	291
scp	282	1684	251	1656	817
udt	200	109	102	204	98
bbFTP (1 connection)	1181	1642	88	> 6600	512
bbFTP (10 connections)	137	1642	99	664	512
ftp	133	1738	123	1070	351

Figure 8: Transfer time (in secs) for 10 GB NetCDF file

software, which is built on the open source Apache OODT framework, to support DESDynI.

In particular, we made use of Amazon AWS cloud services to assess DESDynI's distributed architecture, focusing on a simulated DESDynI workflow called ROIPAC. We demonstrated that ROIPAC, when deployed using PCS, and when combined with high-speed data transfer protocols over fast networks considerably increases the pipeline data throughput. In our experiment, the overall clock time of a workflow composed of the 4 ROIPAC PGEs was reduced by a factor of 1/3 when deployed on a distributed server topology. Our experiments prove that data transfer between cloud servers did not affect the overall pipeline performance.

Though our early results were positive, a number of interesting questions have arisen. Specifically we are interested in: (1) evaluating and integrating cloud-aware scheduling algorithms with the PCS in order to leverage information to more efficiently deploy DESDynI algorithms and speed up the DESDynI pipeline; (2) furthering the work on CAS-PGE production rule specifications for DESDynI PGEs to get a head start on the data system implementation; and finally (3) generalizing our work to automatically integrate data transfer technologies in a science data system pipeline.

7. ACKNOWLEDGMENTS

This work was supported by the Jet Propulsion Laboratory, managed by the California Institute of Technology, under a contract with NASA.

8. REFERENCES

[1] Report of the desdyni applications workshop. `http://desdyni.jpl.nasa.gov/files/DESDynI\%20Applications\%20Report\%20V1.0.pdf`, 2008.

[2] bbftp. `http://doc.in2p3.fr/bbftp/`, 2011.

[3] Gridftp. `http://forge.gridforum.org/projects/gridftp-wg/`, 2011.

[4] *Earth science and applications from space :.* National Academies Press,, Washington, DC :, c2007.

[5] Y. Gu and R. Grossman. Supporting configurable congestion control in data transport services. In *Supercomputing, 2005. Proceedings of the ACM/IEEE SC 2005 Conference*, page 31, 2005.

[6] C. Hoffa, G. Mehta, T. Freeman, E. Deelman, K. Keahey, B. Berriman, and J. Good. On the use of cloud computing for scientific workflows. In *eScience, 2008. eScience '08. IEEE Fourth International Conference on*, pages 640 –645, 2008.

[7] O.-i. Kwoun, D. Cuddy, K. Leung, P. Callahan, D. Crichton, C. Mattmann, and D. Freeborn. A science data system approach for the desdyni mission. In *Radar Conference, 2010 IEEE*, pages 1265 –1269, May 2010.

[8] C. Mattmann, D. Crichton, A. Hart, S. Kelly, and S. Hughes. Experiments with storage and preservation of nasa's planetary data via the cloud. *IT Professional*, 12:28–35, 2010.

[9] C. Mattmann, D. J. Crichton, N. Medvidovic, and S. Hughes. A software architecture-based framework for highly distributed and data intensive scientific applications. In *ICSE'06*, pages 721–730, 2006.

[10] C. A. Mattmann, D. Freeborn, D. Crichton, B. Foster, A. Hart, D. Woollard, S. Hardman, P. Ramirez, S. Kelly, A. Y. Chang, and C. E. Miller. A reusable process control system framework for the orbiting carbon observatory and npp sounder peate missions. *Space Mission Challenges for Information Technology, IEEE International Conference on*, 0:165–172, 2009.

[11] R. K. Rew and G. P. Davis. Netcdf: An interface for scientific data access. *IEEE Computer Graphics and Applications*, 10(4):76–82, 1990.

[12] W. Vogels. Eventually consistent. *Commun. ACM*, 52:40–44, January 2009.

[13] L. C. Voicu and H. Schuldt. How replicated data management in the cloud can benefit from a data grid protocol: the re:gridit approach. In *Proceeding of the first international workshop on Cloud data management*, CloudDB '09, pages 45–48, New York, NY, USA, 2009. ACM.

[14] M. Vrable, S. Savage, and G. M. Voelker. Cumulus: Filesystem backup to the cloud. *Trans. Storage*, 5:14:1–14:28, December 2009.

[15] D. Woollard, O. ig Kwoun, T. Bicknell, R. West, and K. Leung. A science data system approach for the smap mission. In *Radar Conference, 2009 IEEE*, pages 1 –6, May 2009.

[16] D. Woollard, N. Medvidovic, Y. Gil, and C. A. Mattmann. Scientific software as workflows: From discovery to distribution. *IEEE Software*, 25:37–43, 2008.