Local v.s. AWS Provisioning: Experience Fusing a Month's Data on AWS and local provisioning

Tim Boehrner, Landon Clipp, Christopher Kuehn: University of Illinois

Gregory Daues, Donald Petravick, Matias Carrasco Kind: NCSA

October, 2018

Table of Contents

Summary

The Terra ACCESS project produced a 2.4 PB data embodying a 15 years record of earth observations by five instrument on the NASA Terra satellite. The fusion computations were originally performed at NCSA on the Blue Waters and ROGER computing systems. Because NASA's future architecture is cloud oriented, this project describes the work done to gain experience on the AWS cloud under the Illinois contract. The Terra ACCESS computations were repeated for a month's data on AWS and the Illinois Campus Cluster (ICC).

A summary of the significant insights are as follows:

- Currently, compute and storage costs on the Illinois Campus Cluster are significantly less expensive than the AWS charges under the current AWS contract.
- From our understanding there are opportunities to improve AWS storage pricing
- Illinois should provide for volume discounts when negotiating cloud contracts, ahead of need.
- There is a need to protect U of I researchers from competitive disadvantages where cloud costs are significantly higher than at competitive researchers' institutions.
- For the type of trivial parallel processing needed for this application, no substantial usability issues were detected porting Basic Fusion production to AWS or at the Illinois Campus Cluster.
- Consider an architecture that makes optimal use of resource offerings from both AWS, University of Illinois, and other providers.
- The current grant proposal mechanisms do not provide funding for computing charges for investigations such as the Terra ACCESS project.
- The Terra ACCESS proposal would not have been submitted to the ACCESS proposal mechanisms if computing was constrained to AWS under the Illinois contract, due to budget limits in the proposal solicitation, and the risks of underestimating the costs of the needed provisioning.

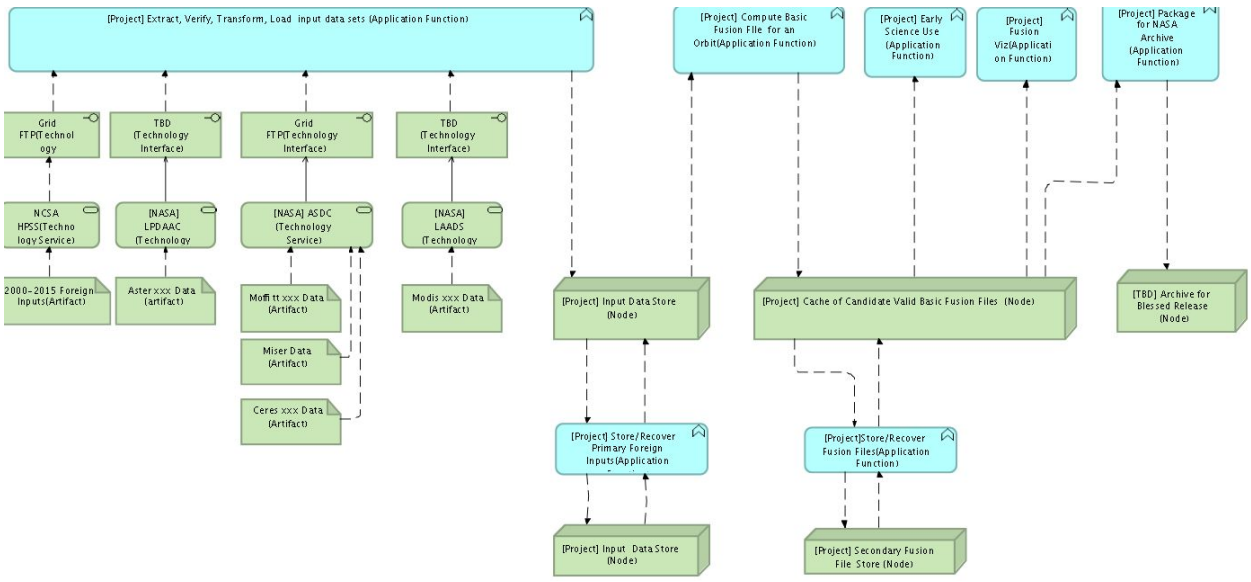
Needs Assessment

One Month of Orbits -- The System Under Study

The Terra ACCESS project obtains the inputs to be fused from NASA Data Access and Archive Centers (DAAC). The acquisition of data for the first 15 years of Terra data involved considerable time. Because of this, the project retains the inputs. These inputs are currently stored in the NCSA tape archive, and would likely be held in Glacier storage on Amazon Web Services (AWS). A working store is used for fusion production and investigation of inputs. At Illinois, this was implemented as storage in Blue Waters and Roger computers. For AWS, this would be implemented as S3 standard storage.

The project saves it outputs as it develops the Basic Fusion (BF) products into final products, until the products are ready for ingest to a store where the outputs may be analyzed. The storage systems at NCSA and within AWS would be the same.

Finally, the products are tendered to an analysis store. For the current Basic Fusion grant, the goal is to tender the 2.4 PB of outputs to the NASA Cloud architecture. NCSA plans to retain a copy of this data to support local hosting and analysis of the data.



Measurements

The project repeated the computation of Basic Fusion files for a representative month. A month was chosen to average out variation in the amounts of data in any particular orbit. The computation consisted of processing Basic Fusion files for over 400 orbits. A month's data was also a deemed to be a reasonable scalable unit for extrapolation of resource requirements for understanding processing conditions in AWS and at the University of Illinois.

Overview of Use of AWS

In order to perform the experiment on AWS, the team met with an AWS solutions architect to discuss possible ways to process the data, as well as various funding methods. The University of Illinois granted the team $1,000 in AWS credits to perform the experiment, which would cover both the intellectual "spool-up" needed to experiment with different computing frameworks, as well as the final run of Basic Fusion file generation.

The original Basic Fusion production was done on local high-performance computing (HPC) machines at NCSA. Applications used in scientific research have varying degrees of fault tolerance, ranging from completely fault intolerant, all the way to robust and heavily optimized checkpointing frameworks. Users of these applications sometimes create their processing frameworks without maximal attention to runtime, due to the invisible financial costs associated with running programs on an HPC system (i.e. a program that runs for 24 hours versus 12 hours takes twice as long, but does not financially cost the user twice as much). When dealing with a pay-as-you-go system such as AWS, every hour of runtime costs real money. Thus, users must take every possible measure they can to reduce incurred costs.

The two main ways of requesting EC2 instances on AWS is via on-demand, whereby the instances are immediately available at the published EC2 pricing, and via the spot market. The spot market works by bidding for instances, where users will specify a maximum percentage of the on-demand price that they are willing to pay (spot percentage). Instances are granted to those users whose spot percentages are above the current instance market value. In general practice, requesting a spot instance can lead to substantial cost savings over on-demand, as the market value often hovers around 20-30% of on-demand. However, the main caveat to spot pricing is that Amazon can at any point in time reclaim the instance from you with only a few minutes warning. Applications that cannot rigorously handle nodes spontaneously being removed from the system will fail.

The initial thinking for this experiment was to simply re-create an HPC environment on the cloud so that the applications designed for such an environment could be ported with minimal effort. This effort was accomplished on a small scale using the tool "cfncluster," however the team quickly realized that this was a naive approach. As far as the team could tell, usage of cfncluster requires the use of on-demand resources for the assurance that the instances will not be

spontaneously removed (as in the case of spot), especially in the case of the login node. Although it is not inconceivable to create a spot-tolerant HPC cluster, currently the team is not aware of such a solution. Because of this on-demand requirement, the applications running on this system would not be taking advantage of the significant cost-savings promise of the spot market.

Fortunately, the Basic Fusion program by nature is highly granular and embarrassingly-parallel, meaning one output file is created by one single-threaded instance of the program with almost no data reuse. There are about 80,000 such files for the Terra mission, meaning that the processing algorithm can be neatly and easily scaled out. The best framework for this type of processing was found to be AWS Batch. Batch works by declaring the CPU, memory, and storage requirements of a single instance of the program, and then also defining the Docker container to use and how to pass arguments to each Docker process. More importantly, Batch can be told to use spot instances and to retry any Docker processes that fail. Batch will then perform the heavy work of instantiating EC2 instances, auto scaling your cluster, sending work to Docker daemons, monitoring the health of EC2 instances, sending EC2 metrics to CloudWatch, sending program logs to CloudWatch, and many other useful tasks.

Intellectual and Human Cost of AWS

When researchers port over code to AWS, the intellectual cost associated with learning and understanding the AWS ecosystem, including knowledge of how to define, create, and implement the necessary framework for one's code, is a non-trivial process. AWS is infamous for being fairly complex to learn and demands a steep learning curve from users not already familiar with cloud computing. The main issue with this is that it poses a barrier that researchers have to jump through in order to do their science. Instead of doing scientific research, they are having to spend significant time creating frameworks as applied to their software. This discourages researchers, particularly those who do not have a background in computer and system programming, from performing research.

The cloud experiment and port of the Basic Fusion code was performed by a senior Computer Engineering student who had no previous cloud experience. It took about 3 weeks before an appropriate framework was identified and implemented for the code. Much of this time was spent learning the AWS ecosystem, going down false paths, testing, debugging, setting up proper AWS logging (including instance-level metric and program stdout logging), and porting the code. The main obstacles that frustrated the efforts was simply an ignorance of AWS and a lack of expert guidance.

There are a few non-exhaustive scenarios that can be immediately seen for a researcher's user experience on the cloud:

Scenario 1

When a researcher needs to use the cloud, they will need to spend a significant amount of time learning how the cloud works and how they can fit their program into the ecosystem. The only resources they will have will be published documentations and user forums. After spending the time identifying which AWS tools are appropriate for their use, they will then need to implement the environment that is capable of running their codes, as well as creating a cost model to estimate the monetary cost for their analysis (assuming their cloud activities are not covered by some umbrella billing entity). This framework implementation can wildly vary in complexity depending on the requirements of the program.

Scenario 2

Instead of having to research all of the various AWS-provided frameworks as applied to the researcher's specific application, they could instead traverse some published list (perhaps a blog post, or some kind of form) that maps application requirements to a recommended AWS solution. AWS does provide use-cases for each of their solutions as well as many useful blog posts on how to implement their solutions, but there are many pitfalls that are not well documented, and there could be multiple different AWS products all perfectly capable of running the application. For instance, a researcher might learn that they can create an HPC-like environment (with login nodes, compute nodes, interconnect, a batch scheduler, etc) on the cloud that would make it easy to port existing codes that are typically run on local cluster computers, but may later find that a better, more cost efficient solution exists elsewhere (as discussed above with the use of cfncluster and AWS Batch). The existence of this recommended solutions list would help prevent the researcher from exploring dead ends or false paths.

Scenario 3

The computing framework will be pre-implemented by a team of framework engineers. This would include one or more frameworks, the sum of which is capable of handling all different kinds of application requirements. For instance, there may be separate frameworks for MPI applications, single-threaded batch jobs, Hadoop jobs, event-driven jobs, data-scraping, etc. Each of these frameworks would tailor to specific types of requirements and would include well-defined interfaces and documentation on how to use them. So instead of the researcher having to build the computing infrastructure for their application, the infrastructure would be already pre-built. This would drastically lower the necessary learning curve because researchers would not necessarily have to understand the inner workings of AWS, but only the interface to the framework that uses the underlying AWS system. This is roughly analogous to current HPC systems: instead of each individual researcher having to build and maintain their own cluster computer, that task is delegated to a group of system administrators that perform the heavy lifting and provide a nice interface to the underlying compute resources.

Performance

The performance of on AWS was measured using AWS standard utilities, including cloudwatch. As a summary, use of the AWS resources was efficient, as measured by running production codes with a fine-grained samling time. Note that these runs were repeated with coarse grained sampling for the costing exercise described in a following section. The run times of the Basic Fusion codes on AWS were comparable with the team's experience on local HPC systems. The most salient figures for the cluster's performance can be seen below.

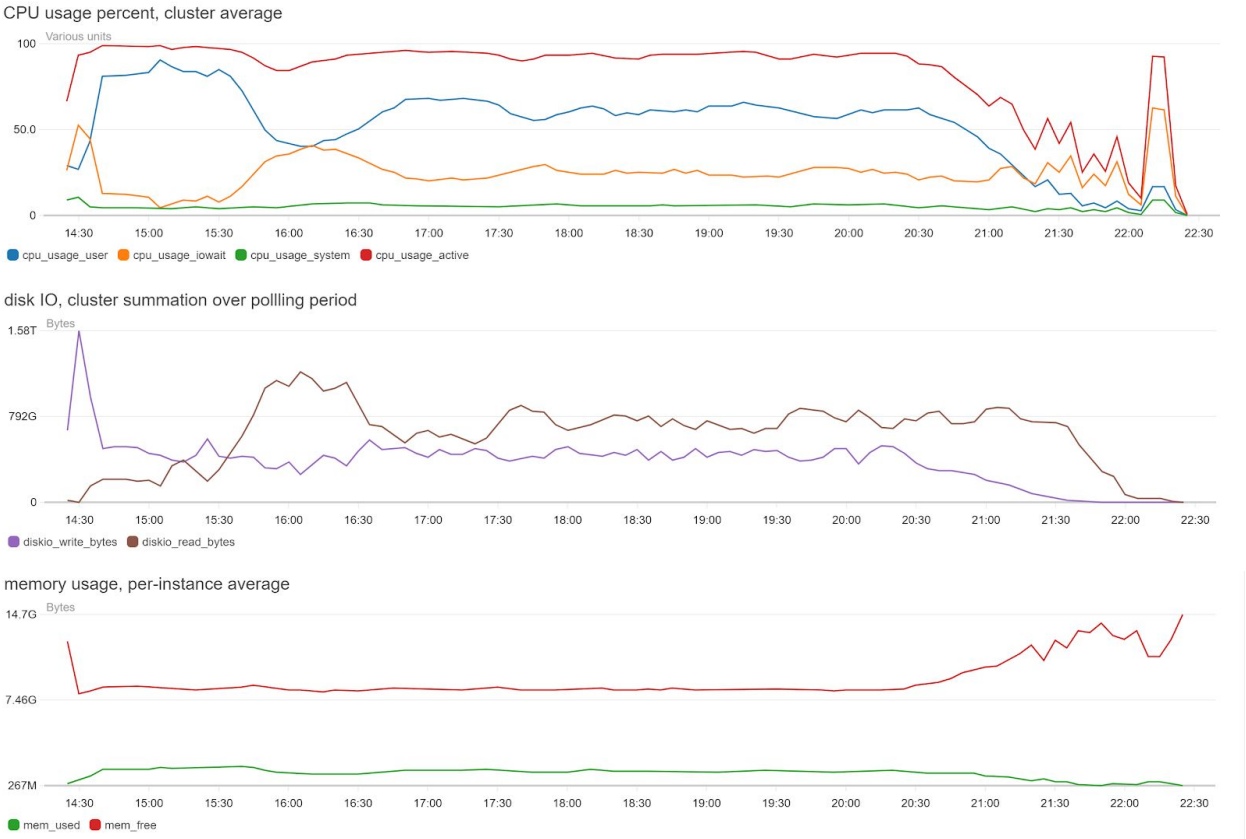


Figure: *The uppermost plot shows the cluster CPU usage was maintained at a high level throughout the period of bulk production. The disk IO was consistent with maintaining performance. The memory usage remains roughly constant at expected levels during production.*

Overview of Computations at the University of Illinois

Illinois Campus Cluster

After consultation and discussions on the computing paradigms available, the University of Illinois Campus Cluster (ICC) staff provided the Terra Fusion effort with access to their Research Computing as a Service (RCaaS) processing queues. For storage, they established a project allocation of 30TB on the ICCP GPFS parallel file system. High performance access to the GPFS file system (e.g., from Blue Waters Nearline/scratch) was provided via the illinois#iccp Globus Online endpoint. The decision to use the cluster charge-by-the-hour costing model was taken as a first approach to an analogous to "On Demand" usage on AWS, deferring for follow-up investigations the consideration of opportunistic models such as using the University of Illinois' nascent HTC resource, or use of the ICC secondary queue to process on resources 'discovered to be available'.

The processing of the 407 orbits of the One Month of Orbits campaign targeted a fairly uniform set of resources within the diverse set of processing servers in the ICCP, choosing nodes with 64 GB of memory that possessed either 16 or 20 cores. The ICCP uses the Torque Resource Manager which is based on OpenPBS. Hence, PBS job descriptions for placing the processing of 16 orbits on a single node were drafted.

For reproducibility and mobility of compute, the One Month of Orbits campaign was executed using containers. On the AWS side Docker was used, and on the ICC platform Singularity was exercised. Singularity (https://www.sylabs.io) is a container platform well suited to shared infrastructure (for example, Leadership class supercomputers, XSEDE sites, Open Science Grid sites, University computing clusters, project resources, etc) where an unprivileged user with an account/allocation wishes to execute the same software environment encapsulated in an image. For the One Month of Orbits campaign, processing derives from an image on the terrafusion basicfusion repository on DockerHub (https://hub.docker.com/r/terrafusion/basicfusion/). For the ICC processing the Docker image is transformed into a Singularity image file on disk ("basicfusion.94aa261188fb.img") via a *singularity pull* command. This single image file may be copied to other platforms (e.g., Blue Waters, SDSC Comet, etc.) and used to promptly execute the basicfusion application (i.e., without a customized build/compile on the platform.)

Processing each orbit with a *singularity exec* using the basicfusion Singularity image file, the One Month of Orbits campaign on the Illinois Campus Cluster clustered the 407 orbits into 26 PBS jobs (of roughly 16 orbits each) submitted through the RCaaS queue. For the first execution of this set of jobs, we observe that the jobs do not all execute simultaneously on the shared resource, though it was observed that ~17 were running simultaneously within the campaign, and so the turnaround time was good. An output dataset of 11.42 TB of basicfusion files were written to the Illinois Campus Cluster GPFS file system.

Other portability demonstrations Blue Waters and Condor Cluster

The singularity containers were used to demonstrate the feasibility of running on large-scale HPC machines (The Blue Waters Supercomputer) and a high throughput Condor installation associated with the campus cluster. The condor job demonstrated use of a “shared-nothing” computer. Input data staged was staged in via `wget` from a remote http server, output data files staged out by HTCondor. Space used on the shared nothing nose was 15 GB inputs, 35 GB outputs on a node with 103 GB available to the HTCondor job. The HTCondor job stages a basic fusion Singularity image and does a `singularity exec` to run the basic fusion app in a container.

Cost

Computation Costing

In the following table, you will find an overview of the costs associated with computing the sample month of orbits across various platforms. There is an additional option for the ICC not previous discussed, the Investment, or Purchase, price for using the ICC.

A hardware investment in the ICC involves the direct purchase of computing hardware and paying associated infrastructure fees that cover the shared costs of operating the cluster environment. No staff time is included in the infrastructure fee, nor is it charged to investors. The costs were calculated for a computing resource of sufficient size to mimic the run-time performance of the RCaaS jobs run during this project. This cost was then divided by the number of CPU cores purchased and the number of hours in the 5-year lifespan of the compute node to determine the effective core-hour cost of a purchased compute node.

Compute -- From 1 Month Basic Fusion Products				
	AWS Bid Price	AWS Actual Price	ICC Purchase (Newest, *)	RCaaS Actual
sample month	$56.60	$22.64	$3.58	$12.45
15 years	$10,188.00	$4,075.20	$644.40	$2,241.00

Storage Costing

There are two principal aspects to storage costing: Storage needs for the production of basic fusion files, and storage needs for serving the files for analysis. Recall that 1 PB is the storage capacity of the basic fusion inputs, and 2.4 PB is the size of the basic fusion files for Terra data acquired through 2015. We assume that only the 2.4 PB needs to be present for analysis.

Illinois Costing

At Illinois, storage is available via rental on the Illinois Campus Cluster Program's Research Storage as a Service (RSaaS) offering, or by investing (effectively purchasing) 5-year storage rights in the storage infrastructure associated with the Illinois Campus Cluster (ICC).

Costing as of the writing of this report is as follows.

Current Illinois Costing at the ICC			
	1 PB-YEAR input	2.4 PB-YEAR input	3.4 PB-YEAR input
ICC RCaas (rental)	$56,000.00	$134,400.00	$190,400.00
ICC 5 yr investment	n/a	n/a	n/a
Illinois "Tape" Storage	No-Cost Award	No-Cost Award	No-Cost Award
Illinois "Tape" Data Access	No-Cost Award	No-Cost Award	No-Cost Award

5 year Storage Costing at ICC			
	1*5 PB-YEAR input	2.4*5 PB-YEAR input	5*3.4 PB-YEAR input
ICC RCaas (rental)	$280,000.00	$672,000.00	$952,000.00
ICC 5 yr investment	$240,000.00	$576,000.00	$816,000.00
Illinois "Tape" Storage	Unknown	Unknown	Unknown
Illinois "Tape" Data Access	Unknown	Unknown	Unknown

AWS Costing

Storage Current Illinois AWS Contract			
	1 PB-YEAR input	2.4 PB-YEAR input	3.4 PB-YEAR input
S3 Standard Storage, Current year	$289,407	$694,577	$983,984
Glacier(retain Only)	$50,332	$120,796	$171,128

Glacier (full retrieval)	$2,621	$6,291	$8,913

Effect of Example 5-year AWS cost reductions for 2.4 petabyte S3 Standard							
		Y1	Y2	Y3	Y4	Y5	**Naive Total**
Assume 0% decline/year	0	$694,577	$694,577	$694,577	$694,577	$694,577	**$3,472,884**
Assume 10% decline/year	0.1	$694,577	$625,119	$562,607	$506,346	$455,712	**$2,844,361**
Assume 20% decline/year	0.2	$694,577	$555,661	$444,529	$355,623	$284,499	**$2,334,889**
Assume 30% decline/year	0.3	$694,577	$486,204	$340,343	$238,240	$166,768	**$1,926,131**

Conclusions

Implications for cloud contracts

Findings:

- Currently, compute and storage costs on the Illinois Campus Cluster are significantly less expensive than the AWS charges under the current AWS contract.
- Unlike the current system, where infrastructure is provided at no-cost based on scientific merit, projects facing direct costs must take every possible measure they can to reduce incurred costs.

Current Illinois Contract: As mentioned above, the one year cost for S3 standard storage for the 3.5 PB of inputs and outputs for basic fusion inputs and outputs at the time of the experiments is approximately $1M/year. Co-incidentally, this is approximately the one-time cost of provisioning 3.5 PB of storage at the University of Illinois.

5-Year Estimated Cost profiles for "large data" cloud contract.							
		Y1	Y2	Y3	Y4	Y5	**Naive Total**
Assume 30 % discount 10% decline/year	0.1	$688,789	$619,910	$557,919	$502,127	$451,914	**$2,820,658**
Assume 30 % discount 20% decline/year	0.2	$688,789	$551,031	$440,825	$352,660	$282,128	**$2,315,432**
Assume 40 % discount 10% decline/year	0.1	$590,390	$531,351	$478,216	$430,394	$387,355	**$2,417,707**
Assume 40 % discount 20% decline/year	0.2	$590,390	$472,312	$377,850	$302,280	$241,824	**$1,984,656**

Table -- Representative cost based on current estimates of storage pricing obtained by organizations committed to large data volumes. Note: Present value of money not considered.

The table below presents a number of representative cost scenarios a cloud vendor would have to meet to approximately match the costs of provisioning 3.5 PB of storage at illinois at the time of writing this report.

5-Year cost profiles for a cloud vendor to meet local ~$1M provisioning costs.							
		Y1	Y2	Y3	Y4	Y5	**Naive Total**
Assume 0% decline/year	0	$200,000	$200,000	$200,000	$200,000	$200,000	**$1,000,000**
Assume 10% decline/year	0.1	$250,000	$225,000	$202,500	$182,250	$164,025	**$1,023,775**
Assume 20% decline/year	0.2	$300,000	$240,000	$192,000	$153,600	$122,880	**$1,008,480**
Assume 30% decline/year	0.3	$350,000	$245,000	$171,500	$120,050	$84,035	**$970,585**

Table -- Representative initial costs, and cost improvement needed to approximate current provisioning costs at Illinois. Note: Present value of money not considered.

Comments:

- From our understanding there are opportunities to improve AWS storage pricing.

Other universities have made commits to at-scale cloud storage. We are unsure of the level of discounting obtained. The table above show costs for a set of discounts which seem plausible to us. Note that the time value of money is omitted from this calculation, but the numbers indicate that it the current contact, may place Illinois researcher's proposal at risk for future cloud-oriented large-data awards, due to cost differences in the cost of provisioning storage.

Recommendations:

- Illinois should provide for volume discounts when negotiating cloud contracts, ahead of need.
- There is a need to protect U of I researchers from competitive disadvantages where cloud costs are significantly greater than at competitive researchers' institutions.

University Cloud Vendor Contract Support

Findings:
The project probed the support which was available to the core team. The Illinois AWS solutions architect advised the project on the AWS deployment, costing and the architectural scope of the costing, and provided AWS Credits to support the AWS deployment. The Illinois Campus Cluster support organization provided deployment advice, costing and ICC credits to support the ICC deployment.

An undergraduate computer engineering student who ran the original basic fusion production codes combined elements of the AWS system and deployed the demonstration on AWS substantially independently, given the support from the AWS solutions architect. The undergraduate spent about a month full-time learning the AWS solutions environments. An NCSA staff member familiar with the Illinois Campus Cluster deployed the demonstrations on the Illinois Campus Cluster, Blue Waters, and the ICC High throughput extension, in a straightforward process.

Given the code was a production code, the exercise did not substantially deal with issues of software development or the scientific development needed to produce a valid scientific basic fusion datasets.

Comment:

- For the type if trivially parallel processing needed for this application, no substantial usability issues were detected porting basic fusion production to AWS or at the Illinois Campus Cluster.

Hybrid Architectures

Findings:
The forward-looking architecture for NASA Earth Science is cloud-based. The costing results show that for the current Illinois contract and campus service offerings, Provisioning at Illinois is less expensive for compute and considerably

less expensive for storage. Moreover, for the current Illinois contract, AWS egress charges are effectively waived.

The project demonstrated the portability of its core workload programs across public cloud, high-throughput and HPC resources. Docker technology was for the public cloud, and singularity technology for the Illinois resources. This demonstrates the feasibility of a portable workload environment (such as uses in the LHC and the Dark Energy Survey) where a researcher requests compute from central / bulk computing infrastructure service, and the service decides optimal place to instantiate the computation.

Comment:
The practical lack of egress charges from AWS for the Terra Fusion project allows consideration of the best use of the AWS cloud, other commercial provisioning, agency-provides provisioning, and local provisioning by the overall project.

There are many successful examples of hybrid systems, which are able to flexibly exploit commercial, agency and locally provisioned computing, and move large amounts of data between sites when needed. As an example, the Dark Energy Survey has a data system at NCSA that exploits bulk production using off-site resources, such as at Fermilab, with some work being done on AWS. The Dark Energy Survey have moved a petabyte in a month though the NPCF border router.

Recommendation:

- Consider a architecture that makes optimal use of resource offerings from bot AWS, University of Illinois, and other providers.

Costs and the Current Grant Programs.

Findings:
The work to produce the Basic Fusion files was performed in the context of an award program limited to $2M. The Cyber infrastructure support for for the production of the Basic Fusion files was provided to no cost to the NASA grant. Instead the infrastructure was provided on the basis of intellectual merit of the fusion work. Given that the cost of storage would approach $1M, the PI of the grant said that he would have never proposed the project had these limitations and cost risk factors been in effect.

Comment:
While exposing direct costs of the infrastructure needed to support a scientific project seems like a step towards a cost-based ecosystem, the system of grants which funds this work is not ready to provide for investigators to bear the the costs of the provisioning project. Exposing the

direct costs to a investigator exposes investigators to risks, such as a hostfall of provisioning needed to complete a project.

- The Current Grant proposal mechanisms do not provide funding for computing charges for investigations such as the Terra ACCESS project.
- The Terra ACCESS proposal would not have been submitted to the ACCESS proposal mechanisms if computing was constrained to AWS under the Illinois contract, due to dollar limits in the proposal solicitation, and the risks of underestimating the costs of the needed provisioning.