

Nick LaHaye
JPL Summer Intern
Mentor: Charlie Avis
Co-Mentor: Paul Springer
OCO-2 Project
August 20, 2012

Pleiades and OCO-2: Using Supercomputing Resources to Process OCO-2 Science Data **Final Report**

For a period of ten weeks I got the opportunity to assist in doing research for the OCO-2 project in the Science Data Operations System Team. This research involved writing a prototype interface that would work as a model for the system implemented for the project's operations. This would only be the case if when the system is tested it worked properly and up to the team's standards. This paper gives the details of the research done and its results.

Background/Objectives:

The OCO-2 satellite will be taking measurements and recording data regarding CO₂ sources and sinks in the Earth's atmosphere. This data will then get downlinked and sent to JPL for processing. About 6% of the data can be processed into deliverable science data locally and close to real-time. At the end of the mission, the data will need to be reprocessed with the new calibration data and algorithms that will be refined throughout the project. Certain requirements need to be met in this reprocessing campaign and, as cited by a presentation given by Elmain Martinez, an SDOS (Science Data Operations System) Systems Engineer, they include the following:

- The OCO-2 SDOS shall complete generation of all science data products within one month of the end of the mission.
- The OCO-2 SDOS shall use less than or equal to one fifth of the SDOS processing resources to perform the initial processing of science telemetry through level 2.
- The OCO-2 SDOS shall process no fewer than 6% of the OCO-2 space-based science measurements through Data Processing Level 2.

If the real-time processing and the reprocessing campaign are done locally, the local computing cluster would need an increase in size to support the initial processing, development, and limited science support. This would greatly increase the cost of hardware needed, and the processing time may end very close to the deadline. This is where the Pleiades supercomputer comes in.

The Pleiades supercomputer is at NASA's Ames Research Center. It has more than enough resources to meet the needs of the project, and if the system expands as predicted, the project may even be use able to use enough of the system to process close to 30% of the data in one-third of the time. The use of Pleiades decreases the hardware budget greatly, increases the science data return, the science team will be able to work on the algorithm for longer, and creates a small increase in the software budget. This increase is due to the fact that there needs to be an interface between the local cluster and Pleiades that sends data to Pleiades, processes the data remotely on Pleiades, sends the results back to the local cluster, and monitors the status of the interface. This interface is what I am prototyping with the OCO-2 SDOS team.

Approach:

The first step that was taken as a group was setting down requirements. Most of the high level requirements, or the requirements that deal with the system and its components in an abstract way, were all set before I started. Before each of the system's components can be created, its low level requirements, the requirements that deal with the details of the system, had to be solidified. This being said, like in every software development effort, requirements are subject to change at any time given the consent of the group. These requirements called for four main components in the interface; one for uploading data to Pleiades, one for processing data on Pleiades, one for downloading data from Pleiades to the local cluster, and one for monitoring jobs' statuses throughout their process.

My first task in this project was to create the monitoring component. I installed all necessary software and my mentors sent off a request for an account on Pleiades. Next, I read documentation on the functionality of Pleiades' system calls 'sup' and 'shiftc' that will be used in this interface. I also did some reading on Portable Batch Systems, TORQUE Resource Manager, the Sphinx documentation tool, and the Python programming language's subprocess library. After gathering the necessary knowledge, and talking to my mentors as well as Lan Dang, one of the system's operator, I began work on the monitoring component of the interface. My mentor, Charlie Avis, and I came up with a list of functional requirements. From those requirements I was able to create the monitoring tool. This tool allows the system's operator to view the status of the processing jobs in 4 different ways. I also created a separate component that stores the information gathered by the monitoring tool in a database for future reference. Because I did not yet have an account on Pleiades, I created my tool to work on the local cluster, with the necessary lines of code for the tool to work with Pleiades commented out. After fixing the bugs and modifying the output format, I documented my code using a documentation tool called Sphinx.

Next, I took over my Co-Mentor, Paul Springer's 'stats.py' program, a python script that gathers information about the processing of instrument data by L2 Full Physics, the key program used to create science products for the OCO-2 project. It outputs statistics on processing time, iterations, and other computational parameters. I have edited it, tested it, fixed bugs, documented the code, and implemented recommendations. This component will run on the data locally after it is returned from Pleiades.

After this, I created the submit script for the Pleiades interface. This script accepts a path to a processing aggregate as well as some other information and creates signal files the upload component uses as a flag to start uploads. After documenting and testing the script, I generated the signal files necessary for the first test of the upload component using the script.

Finally, after 6 weeks of waiting, I got my NASA SecureID RSA token on 7/16, necessary to access NAS, NASA Advanced Supercomputing, resources, Pleiades in this case. With this, I was able to test my monitoring tool's ability to access Pleiades and correctly report job status. After a few minor fixes due to documentation errors, the monitoring component was ready to be integrated into the interface.

Concurrently, as a side project, I worked with Dr. Richard Lee to automate the quality checking of OCO-2's instrument data. This software takes the packets that are produced, concatenates them based on dates and the type of instrument being tested, runs a previously created script on these files, and checks the output for errors that are not part of already noted recurring patterns. After working closely with Dr. Lee as well as with Brian Chafin the output of my script was approved, I have documented the code, as well as added to some previously existing documentation to tell testers how to run the necessary scripts. This code has been checked into subversion and is now in the project's computing cluster.

My next task was to work on building and installing the newest version of L2 Full Physics on Pleiades. This needed to be done because the version previously on Pleiades was an older one, and multiple different versions of the program will need to be used during operations. I had to edit a few of the build scripts as well as install a few other programs on Pleiades in order for L2 Full Physics to build

and install correctly. I then had to learn how to run the complex program on the local computers in order to test my install on Pleiades. The install was successful. After it was installed, I ran a few diagnostics to make sure this version was running properly with our interface. First, I compared the output data from my run on Pleiades to that of an identical run on the local machines using a python script provided by the L2 Full Physics developers. This test showed that the data from each run was actually different. After this difference showed up on two more runs, I consulted the code's developers. They informed me that this difference was a known difference that occurred when working on different operating systems. With that knowledge I was able to move on, and run some timing diagnostics on the system's processing speed versus that of the local machines. As expected, Pleiades ended up being faster and more efficient.

While I was working on these tasks, my mentor, Paul Springer, had developed and successfully tested the processing, download, and upload components according to the set list of functional requirements. With all of the components done it was time for an end-to-end test. Due to Paul Springer being on jury duty the test was postponed a week, so I was able to go back to the components I built to ensure that they still worked correctly given all of the updates to the system. I was able to complete this, and in the next week the end-to-end test was run.

Results:

Because of permissions issues with OCO-2's Process Control System, Albert Chang had to start up the end-to-end test. The first issue that the team came across was that a file that indicated processing completion to the system was being created with the wrong name. There was also an issue with permissions. The process is being started from one user's account, but all of the Pleiades data has to be accessed from a different account, so permission modifications had to be added to the code as well. After these fixes were implemented, the system reported success for all 166 jobs in the first end-to-end test. The interface was able to effectively and efficiently process the data and return it. More tests need to be run to make sure the interface can handle the load that will be given to it during operations, but those will be run at a later time.

Discussion:

The project's outcome did not deviate much from its expected results. The biggest deviation will come from what we learned from the Pleiades developers during a teleconference. We spoke with them in order to ensure our current architecture could be supported by their system and that it followed their security policies as well as our own. As far as accounts were concerned we wanted to be able to access a group account on Pleiades from a local group account. We wanted this because to use Pleiades' remote system calls, the account passwords need to be renewed once a week. Using two group accounts would eliminate dependence on one person, making it easier for multiple operators to use the system. Also, from the Pleiades group account we wanted to be able to run cronjobs, jobs that wake up and run on a set time interval, and daemons, jobs that are started and then continually run in the background until stopped.

We got a mix of good and bad news from them in return. They told us we would be able to have a shared account on Pleiades that we could run cronjobs and daemons on; all that needed to be done was to file more paperwork. The account however, could not be directly accessed. You would have to log into a personal account and then log into the group account. This would work for us, except for the fact that they do not allow us to access that group account from a shared account, meaning that we could not use a local group account to access Pleiades. The main problem with this is that shiftc, a robust data transfer client installed on Pleiades, can only be used to push data to Pleiades, not vice

versa, and our idea was to initiate data transfer from a shared account as to not rely on one person since there are multiple operators. Without a group account here to access the Pleiades group account, this solution would not work.

After some thought another solution was proposed that allows us to still use shiftc and keep much of the same architecture in place. In this solution, each operator would have a data transfer specific local account, a specific Pleiades account, and a group Pleiades account would be created. In the user specific accounts, both local and on Pleiades, there would be simulated links, or pointers to data in other areas. These links would point to shared data on both sides of the interface. This way each operator can take control of in progress data transfers without causing any issues. Furthermore, this allows us to still utilize shiftc for data transfer instead of having to create our own tool. This is the biggest change that will need to be implemented.

In regards to the actual interface itself, it does not seem like much change will be needed. The final product will probably be a much more robust solution than the prototype, but the tests proved that this system works effectively and efficiently.

Conclusion:

At the beginning of this experiment it was known that if this interface was successful it would lower the project's hardware costs, possibly create five times more science data than a local cluster, and due to the fact that the data will take less time to process than if it were processed locally, it will allow for more time to write the algorithm software. Now that we know this interface can be implemented effectively, this will be a great resource saver for the project. There won't be training necessary, since the original project Process Control System is integrated into the front end, and the operators were very involved in the creation of this prototype. It would be a simple transition if it is decided that this interface is going to be implemented into the final operations' design. Overall, I think this research greatly benefitted the OCO-2 project.

Acknowledgements:

I would like to thank my mentors Charlie Avis and Paul Springer for putting me on this project, giving me all of the necessary resources, and providing help and feedback whenever necessary. Next, would like to thank the OCO-2 SDOS team for assisting me when I had questions or needed help and for all of their work to make this project successful. Lastly, I would like to thank JPL and Cal Tech for making this opportunity possible.

This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, and was sponsored by the JPL Summer Intern Program and the National Aeronautics and Space Administration.

Sources:

- 1) PleiadesForReprocessing_v3.pptx – A PowerPoint presentation given by Elmain Martinez.
- 2) Plan.docx – A schedule estimate created by Charlie Avis and edited by Paul Springer.
- 3) WhatdoesOCOMEasure.pdf – A document explaining what the OCO project did created by Charles Miller, the Principal Investigator of the OCO project.
- 4) My project outline.
- 5) A presentation given by Elmain Martinez
- 6) "OCO." eosps0.gsfc.nasa.gov. NASA, n.d. Web. 31 May 2012. <http://eosps0.gsfc.nasa.gov/eos_homepage/mission_profiles/docs/OCO.pdf>. "OCO."

- 7) "Pleiades Supercomputer." NASA Advanced Supercomputing Division. NASA, n.d. Web. 31 May 2012. <http://www.nas.nasa.gov/hecc/resources/pleiades.html>
- 8) "OCO-2." OCO-2. NASA, n.d. Web. 31 May 2012. <<http://oco.jpl.nasa.gov/>>.