2012

Significantly reducing the processing times of high-speed photometry data sets using a distributed computing model

Paul Doyle
Technological University Dublin, paul.doyle@tudublin.ie

Fredrick Mtenzi
Technological University Dublin, Fredrick.Mtenzi@tudublin.ie

Niall Smith
Munster Technological University

See next page for additional authors

Recommended Citation

Paul Doyle, Fred Mtenzi, Niall Smith, Adrian Collins, Brendan O'Shea, "Significantly reducing the processing times of high-speed photometry data sets using a distributed computing model," Proc. SPIE 8451, Software and Cyberinfrastructure for Astronomy II, 84510C (24 September 2012); DOI: 10.1117/12.924863

Significantly reducing the processing times of high speed photometry data sets using a distributed computing model

Paul Doyle*[a], Fred Mtenzi[a], Niall Smith[b] , Adrian Collins[b], Brendan O'Shea[a]
[a]Dublin Institute of Technology, Kevin Street, Dublin 8, Ireland; [b]Cork Institute of Technology, Bishopstown, Cork, Ireland

ABSTRACT

The scientific community is in the midst of a data analysis crisis. The increasing capacity of scientific CCD instrumentation and their falling costs is contributing to an explosive generation of raw photometric data. This data must go through a process of cleaning and reduction before it can be used for high precision photometric analysis. Many existing data processing pipelines either assume a relatively small dataset or are batch processed by a High Performance Computing centre. A radical overhaul of these processing pipelines is required to allow reduction and cleaning rates to process terabyte sized datasets at near capture rates using an elastic processing architecture. The ability to access computing resources and to allow them to grow and shrink as demand fluctuates is essential, as is exploiting the parallel nature of the datasets. A distributed data processing pipeline is required. It should incorporate lossless data compression, allow for data segmentation and support processing of data segments in parallel. Academic institutes can collaborate and provide an elastic computing model without the requirement for large centralized high performance computing data centers. This paper demonstrates how a base 10 order of magnitude improvement in overall processing time has been achieved using the "ACN pipeline", a distributed pipeline spanning multiple academic institutes.

Keywords: High-Speed Photometry, Distributed Computing, Cloud Computing

1. INTRODUCTION

The generation of large volumes of scientific data presents an epic challenge to many scientific disciplines [1]. Large-scale data generation requires us to review how we replicate, transfer and process data. The problem of expanding datasets exists for observatories and institutes big and small due to the availability of affordable high performance CCD (charge coupled device) and CMOS (complementary metal oxide semiconductor) image capture devices. Standard CCD image reduction must be performed using bias and flat field frame to calibrate each of the data frames, followed by basic photometric analysis involving image centering, estimation of the sky background and magnitude intensity estimations for point sources. As the number of data frames increases, the amount of work to be performed also increases, having a direct correlation to the processing time when using a sequential processing pipeline. For data reduction processing pipelines to expand at the same rate as data generation we must re-evaluate existing processing techniques. In this paper we propose a distributed processing pipeline that exploits the parallel nature of the data to be processed which we will refer to as the ACN pipeline. The ACN pipeline performs data reduction and basic photometry on CCD data images. In this solution all data is compressed and uploaded to the Amazon Simple Storage Service (S3) facility. Once uploaded a central queue is constructed and a set of Astronomical Computing Nodes (ACNs) query the queue for work and download a copy of an image file from S3 for processing.

Using the ACN pipeline we have demonstrated the following:

1. Identified opportunities to clean and reduce CCD images in parallel such that the processing time for all files is a function of the processing time for a single file rather than the number of files to be processed.
2. Dynamically incorporated additional computing resources to a running, distributed pipeline to reduce overall processing time.

In Section 2 we review the background to this research identifying existing techniques and their performance. Section 3 proposes a distributed approach, which is expanded into an experimental design to validate our approach in Section 4. Results and Analysis follow in Sections 5 and 6, followed by our conclusion where we consider future work.

2. BACKGROUND

2.1 High Speed Photometry

The increase in affordable high resolution imaging devices such as CCD or CMOS has meant that almost all observatories and research groups have the capability to capture significant amounts of data in reasonably short timescales. The point of reference for this paper has been the Blackrock Castle Observatory (BCO), Ireland, which is engaged in high-speed photometry research [2]. The result of this process is the generation of large volumes of raw data images that must be calibrated and prepared for analysis. The standard process of data reduction and cleaning required for these images [3] is the focus of the ACN pipeline (pixel calibration using flat field and bias frames followed by image centering, sky background estimation and estimation of point source intensity for each object of interest).

2.2 Expanding Data

The rate of data acquisition that can now be accomplished, with modest facilities, is a very real obstacle to overcome in any data pipeline. As we can see in Figure 1, as the camera resolution moves from a 512x512 resolution to 1 megapixel we can acquire approximately 1 terabyte per day. Work is already underway at BCO to use a 5 megapixel CMOS camera, which in 8 hours could capture 20-25 terabytes of data.

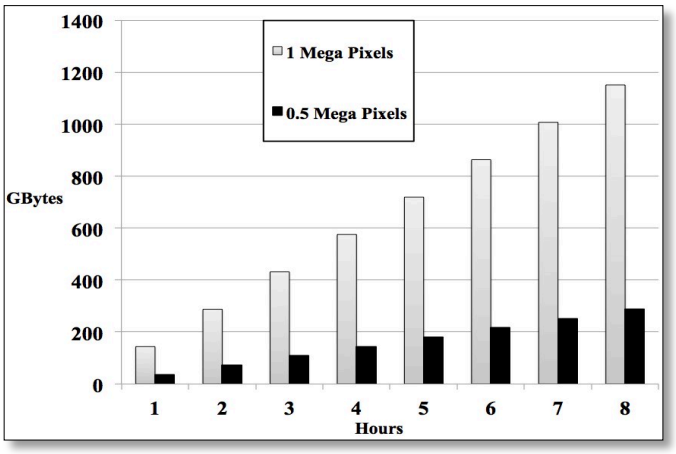


Figure 1. Data capture rates in Gigabytes from 1 to 8 hours for two resolution sizes. Devices are continuing to increase their resolution and still provide high image capture rates per second [4].

2.3 Inelastic and Elastic processing models

Most data reduction tools and algorithms used by software packages such as IRAF [5] and the Common Pipeline Library (CPL) [6] are basically sequential in nature, which process files interactively or in a batch sequence relying on high performance hardware devices to ensure that the data reduction process is kept within a reasonable timeframe. Work has been done to enhance the performance of the CPL library by adding multi-threaded support [7] to utilize multi-core hardware, but this approach is still single system focused and does not allow dynamic expansion of computing nodes. We can describe this as an *inelastic* processing model.

If we consider the data capture rate from a 1 megapixel camera from Figure 1, a single CCD device would require an end-to-end reduction pipeline that could process image files approximately 4MB in size every 0.1 seconds to match the acquisition rate. What an end-to-end pipeline means is that the cost of data transfer to computing devices is included in the overall performance metric for the pipeline. If we have an elastic processing model we can add additional computing resources to a running pipeline, increasing the overall image-processing rate. All systems that rely on single fast devices to process data are inelastic by nature. Using a proprietary MATLAB based reduction pipeline designed to batch process CCD images, BCO experienced processing rates in the order of 1 image per second, with a capture rate of 0.1 per second. The inelastic nature of this and similar sequential processing models means that additional hardware resources are not easily incorporated into the pipeline. While resources may be sufficient for existing processing rates, there are very real data processing limitations when planning upgrades to higher resolution/fast integration CMOS technology. These limitations provide a compelling case for the redesign of data reduction pipelines which are elastic and that can

reduce processing times when more hardware is available. To accomplish this we must ensure that we understand the parallel nature of the dataset itself.

2.4 Blackrock Castle Observatory Dataset

A 26-gigabyte dataset from BCO consisting of 36,820 images stored in data cubes of 10 data images per file, with data frame integration times of .08 seconds per image was used in our experiments. The images are stored in an uncompressed FITS (Flexible Image Transport System) file format approximately 7MB is size. The resolution of the camera used was 512x512 pixels with the image borders cropped by about 20 pixels around each image resulting in approximately 8.8 Billion pixels to be processed by the ACN Pipeline. This dataset has already been processed [8] and it is for this reason that it is a good reference point for this paper. The data requires two primary operations to be performed prior to analysis. First each pixel needs to be calibrated taking into account the flat field and the bias master images and secondly using these cleaned pixels a series of magnitude values need to be calculated for each of the reference points within each of the images. Both of these operations are well known and robust and easily validated against an existing pipeline.

3. OUR APPROACH

A principle requirement for the ACN pipeline was the ability to dynamically incorporate additional hardware to assist in the reduction of overall image processing time. This *elastic* capability insulates the pipeline from the limitations of a single high-performance system approach, which will eventually fail to meet the demands for data processing when faced with ever expanding data capture rates [9]. To accomplish this there was a need to identify opportunities in processing images concurrently. The starting point was to consider the existing sequential processing approaches such as those already mentioned in section 2.3. Typically images go through a series of jobs such as calibration followed by magnitude estimations for point sources, which we can generically describe as follows.

3.1 Sequential Pipelines

Most photometry reduction pipelines follow the same basic principle of an IRAF reduction process of sequential processing of files through various functions. In the ACN pipeline we consider all steps that must also be performed including the movement of data to the processing machines and the uploading of results files to an appropriate location. We can summarize the key steps within a sequential pipeline for a single file and represent it in Equation (1) where s is the number of stars to process in an image, i is the number of images in a file, and f is the number of files to process.

1. Copy a raw FITS file to the pipeline's processing system ($\alpha = time\ to\ copy$)
2. Open a FITS file ($k = time\ to\ pixel\ clean\ one\ image$)
3. Clean each pixel using the Master Bias & Flat ($\beta = time\ to\ pixel\ clean\ one\ image$)
4. Generate a cleaned version of the FITS file ($\nu = time\ to\ save\ a\ file$)
5. Open cleaned FITS file and estimate the magnitude for each star ($\delta = time\ to\ generate\ magnitudes\ per\ star$)
6. Generate a set of results files containing the magnitudes ($\nu = time\ to\ save\ a\ file$)

We should note that this process contains a high rate of file I/O and is a function of the number of files to process. It is also a case that a two-pass process requires a doubling of the data storage and a double read and write for every file processed.

$$\Big(\big(\ \big((\delta * s) + \beta\big) * i\ \big) + (\alpha + 2\kappa + 2\nu)\Big) \ * \ f = T_{total} \tag{1}$$

It is this sequential process where T_{total} is a function of the number of files that is reconsidered in our proposed approach. The requirement is to take advantage of all possible concurrent-processing opportunities and exploit them.

3.2 Distributed Processing and Opportunities for concurrent processing

All aspects of the reduction pipeline must be considered when dealing with large volumes of data when seeking processing optimizations. We first considered how to use an existing pipeline tool such as IRAF, which has batch processing capabilities, and to run multiple instances in parallel however the setup and configurations for simple batch processes was considered to be excessive for the simple processing required on our data set. A more lightweight tool was instead developed in C using the CFITSIO [10] library to perform data reduction and photometry within the ACN pipeline. This new utility *acn-aphot* takes in a FITS file for processing and optionally calibrates the pixels prior to

magnitude estimation if bias and flat frames are provided. This tool was designed to run on a Linux platform and is the primary astronomical processing component of the ACN pipeline.

In a review of the raw FITS files we observe that each pixel within each image follows the standard calibration process using their corresponding bias and flat field values from the master bias and master flat field frames [3].

$$\frac{Raw\ Frame\ Pixel - Bias\ Frame\ Pixel}{Normalised\ Flat\ Field\ Frame\ Pixel} \tag{2}$$

This calculation can be run for each pixel in isolation from all other pixels once there is a master bias and master flat file available. This means all pixels could be calibrated in parallel, although file I/O costs would make this an expensive operation.

In a review of the magnitude generation, it was noted that the pixels used for each calculation are approximately 80x80 pixels in dimensions. These are the minimum amount of pixels to be considered as a logical group, approximately 6,400 pixels in total for each point source. Within this region basic photometry is performed, i.e. centering using the WPHOT algorithm [11], calculation of the sky background and estimation of the point source using the following standard equation [3] where I is an estimate of the collected source intensity, and C is a constant used to place the source magnitude on a standard magnitude scale.

$$Magnitude = -2.5\ log_{10}(I) + C \tag{3}$$

If we combined these two observations such that the cleaning of pixels was only done on a pixel used in the calculation of the magnitude, we can achieve a number of improvements.

1. The number of pixels to be cleaned is drastically reduced but is related to the number of objects of interest in each data image. Assuming five stars of interest in a 512x512 image with a clip region of 80x80 for each star we can clean 77% less pixels (1.17 billion pixels instead of 8.8 billion) using Equation 2, without affecting the final magnitude calculation.
2. Producing magnitude estimations directly from raw FITS image files eliminates the need for intermediate cleaned files, halving the storage requirements of the pipeline.
3. A single read and write operation is required per file, halving the file I/O processing.
4. We can run multiple programs concurrently working on separate raw files without affecting the overall process.

If we now have the ability to process each file completely independently and at the same time, we can modify equation (1) such that it reflects an overall processing time that is inversely proportional to the number of ACN computing nodes N used in the pipeline as we have shown in Equation 4.

$$\frac{\left(\left(\left((\delta * s) + \frac{\beta}{8}\right) * i\right) + (\alpha + \kappa + \nu)\right) * f}{N} = T_{total} \tag{4}$$

To coordinate multiple running instances of the *acn-aphot* we require a central point of communication and coordination to ensure each instance is processing its own file with limited or minimum duplication occurring. A simple queue system that is public to all concurrent instances which allows each instance to uniquely lock a file for processing is required. The raw image data must also be available on a storage device for running instances to access. This storage must provide sufficient performance to service high volumes of concurrent file requests.

4. EXPERIMENTAL DESIGN

Given the objective of building a processing framework capable of elastic allocation of computing resources to a running pipeline and the objective of processing raw CCD images an order of magnitude faster than the capture rates we designed a system that is robust and within our resource limitations. A key aspect of the design was to allow any type of hardware that could run a compiled instance of the *acn-aphot* to contribute to the processing of FITS images while the system is running. This allowed the use of older computing devices in our pipeline, which on their own, were considered too slow to contribute to a sequential pipeline given their limited CPU and memory configurations.

4.1 The ACN pipeline

All measurements within the ACN pipeline incorporate all aspects of the processing pipeline. While it may be appealing to measure the processing time alone, ignoring the orchestration required in preparing the data for processing, this would provide an incomplete view of pipeline's true performance. If it takes hours to prepare a system for analysis, including transferring, formatting or compressing the data and only seconds to process the data it would be misleading to represent the processing as the cost of the pipeline. We do however require an initial assumption for our starting point, which is that the dataset is available over NFS (Network File System) on an IBM eServer 326 (7 year old server, 4GB Ram, Opteron 2.8 GHz CPU) running FreenNAS 8. The cost of the pipeline will include any time moving or processing this data through to the generation of our results files. Another assumption is that we have already created available astronomical computing nodes (ACNs).

We will use the BCO 36Gbyte dataset referenced in section 2.4 consisting of 3682, 7MByte files, which had a capture time of approximately 1 hour. This requires that the ACN pipeline performance must be ≤ 0.01 seconds per image to reach our goal of an order of magnitude faster cleaning time compared to capture time. In other words the ACN pipeline has 6 minutes to process the dataset of 26Gbytes, which was captured in 1 hour.

The ACN pipeline can be summarized as follows (Figure 2.)

1. Compress Data files on the NFS Share
2. Upload all data to an Amazon S3 bucket
3. Generate a public queue accessible to all ACN nodes
4. Prepare all ACN nodes by downloading required software to process image files
5. Activate all ACN processing machines
    a. Lock an available file on the queue
    b. Download the file to local storage
    c. Clean and calculate magnitude values using *acn-aphot*
    d. Upload results file and performance statistics
    e. Look for another file from the queue

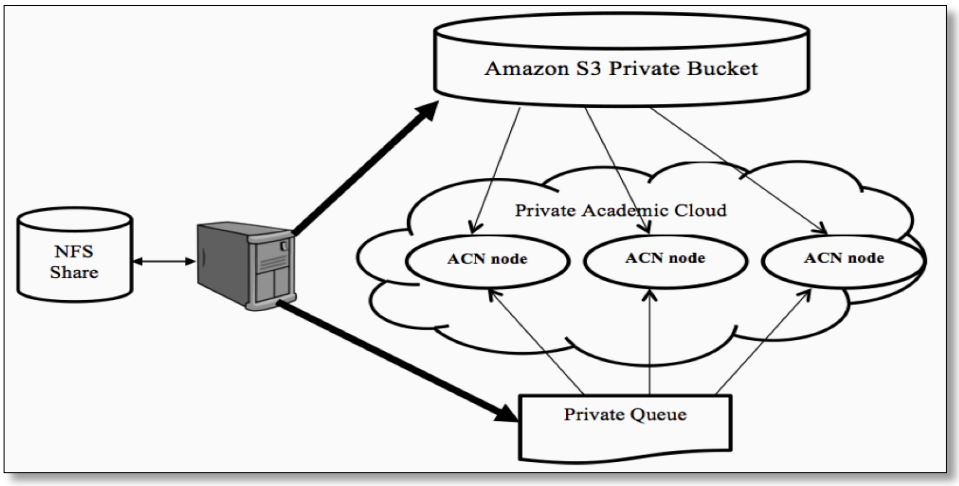


Figure 2. The ACN Distributed pipeline using the Amazon S3 storage to store compressed FITS images that are made available for download by the ACN nodes running the *acn-aphot* utility.

4.2 Data Compression

The FITS format can be compressed using the *fpack* utility, a standard utility available within the CFITSIO library. Using the default parameters we can compress the entire 26GByte dataset down to 4.7GBytes without affecting the resulting magnitude calculation [12]. To accomplish this a multi-core Dell 410 PowerEdge Server with 64GBytes of RAM was used, as the compression speed is a core piece of the sequential part of this pipeline. This is the only purpose

specific system used as we are looking to use where possible older systems which are easily obtained. Compression was seen as an essential step as it helped address slower transfer rates across networks.

4.3 S3 Data Storage

A high performance data storage system is essential to this design, as all nodes must have fast access to raw data files. Initially a local *FreeNAS* file system was used but the lack of data replication caused bottlenecks in processing file requests from a large number of ACN nodes, so Amazon's S3 was used instead. The Linux utility *s3cmd* was used to create storage 'buckets' on S3. Each file is downloadable using the *wget* Linux utility making it accessible to all ACN nodes. The S3 storage also provides automatic data replication and allows for a larger number of concurrent file requests and while it has a significant delay in reading a single file, that delay is not cumulative and remains constant over multiple concurrent file requests.

4.4 Building a queue

The process by which the ACN nodes determine what file is available for processing is based on NFS file locking. Each ACN will traverse the full queue as a directory listing looking at all filenames. When it finds a file that does not have a tag of "LOCKED" it will attempt to rename the file and if it succeeds it will download the file from S3, process it and upload the results file to an NFS share or an S3 storage location.

4.5 Building a multi-institute network

The design demonstrates the use of computing devices of varying capabilities located in multiple institutes allowing the addition of systems to a running pipeline in an elastic manner. To demonstrate this, a private layer 2, point-to-point network, was constructed with the assistance of HEAnet, (Ireland's National Educational & Research Network) between three academic institutes in Ireland: Dublin Institute of Technology (DIT), Cork Institute of Technology (CIT) and the Institute of Technology Tallaght (ITTD) as shown in Figure 3. A private IP network was constructed and 8 IBM eServer 326 machines were added to the network from each location, each one operating as an ACN node. FreeNAS storage devices were added to provide a central queue along with common utilities used by ACN nodes. Additional ACN nodes were added to the network including 30 VMware instances running across 4 x4150 Sun servers running VMWare ESXi, a Dell 410 PowerEdge server and a Mac Server. The network was available through an IP gateway system with all nodes able to access the Internet through routing gateways.

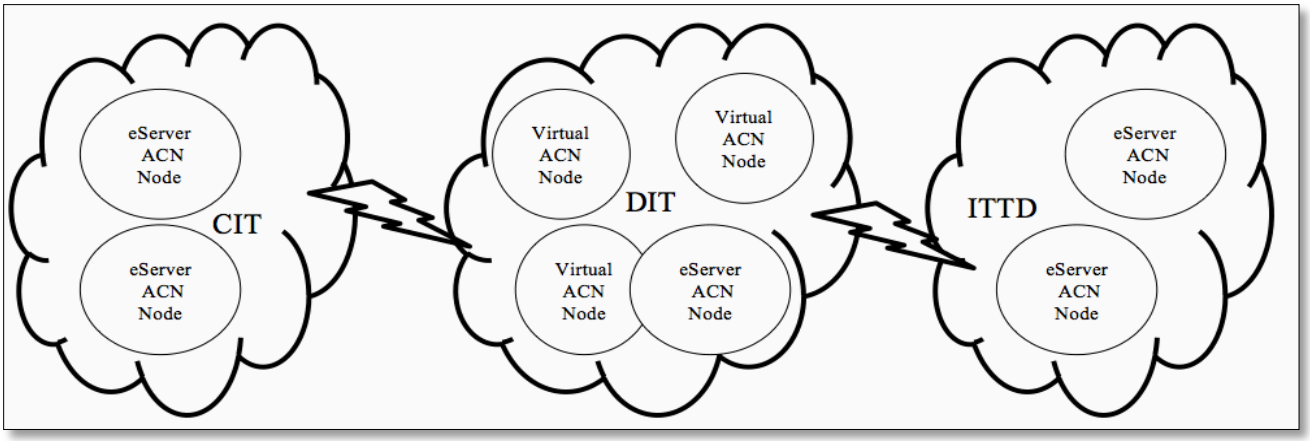


Figure 3. ACN Pipeline Multi-Institute network for distributed processing showing various ACN nodes running in each location. The network is a private layer 2, point-to-point network, provided by HEAnet.

4.6 ACN nodes

The *acn-aphot* utility, which runs at the heart of each ACN node, is a C program compiled for multiple Linux systems including the MacOSX. Its primary function is to generate a valid magnitude estimate for stars in an image, ensuring that the pixels used in the calculation are correctly calibrated. The ACN node is a system which when started; downloads the *acn-aphot*, Master Bias and Master Flat images along with other various utilities; goes to the queue for work; downloads the file it locked from an S3 bucket; processes the file and uploads the result. The running C program has a memory

footprint of only 1.2Mbytes and consumes typically less than 100% of a 1Ghz CPU. An ACN node can be directly started remotely or can watch for a valid queue to be available and then start working. If a device is added asynchronously to an active queue it simply looks for the next available item on the queue and joins the cleaning process contributing to the overall cleaning rate.

5. RESULTS

The BCO MATLAB based pipeline processed the dataset used in our experiments in approximately 10 hours at a rate of 1 image processed per second. However rather than use the BCO pipeline as a processing benchmark we focus on the capture rate of 10 images per second and use this as our starting point requiring a processing rate of 100 images per second to achieve an order of magnitude faster processing time over capture time. We review each of the steps in the ACN pipeline to optimize the overall processing time for the 26GByte dataset.

5.1 Compressing FITS files

The first step in our process was to perform a compression of the dataset to reduce the amount of time copying data files to ACN nodes. The compression used was the CFITSIO *fpack* utility, which was run in standard mode. To determine the optimal performance of the Dell 410 server, FITS lossless compression was run using two different methods. The first ran the file in sequential mode where a file-list was given to a single instance of the utility. The second approach started thousands of processes at staggered intervals where each process was running concurrently utilizing large portions of the available memory on the server. This approach was only valid for a server with large quantities of memory and multiple CPU cores. The only option for running this on an older server was in sequential mode. The compression time reading and writing files on the NFS server mount point are shown in Figure 4. The parallel performance of the Dell 410 was instrumental in the pipeline providing a compression time considerably faster than the sequential method. It was interesting to note that the older IBM systems provided comparable compression times to the Dell when run in sequential mode. The resulting size change was significant with the dataset being reduced to 4.6GBytes from 25.6GBytes.

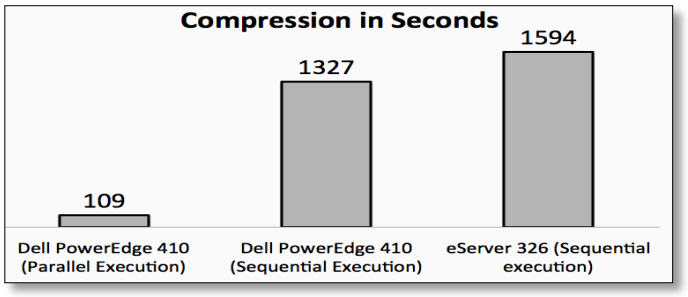


Figure 4. The compression time for the full dataset running in multiple modes using a single core IBM eServer 326 and a Dell 410 PowerEdge multi-core system.

5.2 Uploading files to S3

Figure 5 shows the transfer times of all multiple datasets to an S3 storage bucket. The private network was connected via a Gigabit switch to the DIT network, which has an external institute-wide Internet connection of 1 Gigabits. Approximately 20% of the bandwidth was used in this transfer although the use decreased to approximately 8% as the number of files was increased. The nature of the network is that it is variable, however the experiments were run late at night when the Institute's network was lightly loaded. The clipped and compressed data files were files where the FITS files were clipped into the smallest possible region around the star so we did not need to transport any pixels we did not intend to clean. The time saved in data transfer however was negligible. The upload of compressed versus uncompressed was reasonably linear with an 80% saving in transfer time.

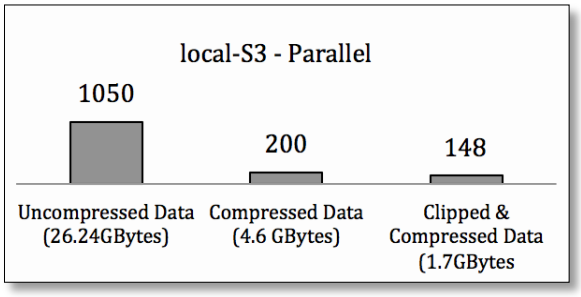


Figure 5. Transfer times for data from the local NFS storage within the private network to an S3 storage bucket

5.3 ACN node performance

Each ACN node performs at minimum, a queue lookup, a file download, a file clean and a file upload. In a sequential system the steps for calibration, using Equation 2, are usually performed first requiring a full read and write of all of the data, duplicating the amount of storage required. This is followed by a magnitude estimation of each point source using Equation 3. We can refer to this as a 2-pass system where files are processed twice. Where both of these operations are carried out during a single read of the files we refer to this as a 1-pass system. To determine the performance of the acn-aphot utility it was run in a 2-pass and 1-pass mode and the results were compared to the BCO system. While the BCO system performs slightly more processing on each file (approximately 20% overhead) we can use it for general comparison purposes. In Figure 6 we can see that the acn-aphot utility running in 2-pass mode on the IBM eServer 326 significantly outperformed the MATLAB system. Much of the enhancement in processing over MATLAB may be attributed to the C language itself and to variations in some of the processing steps performed so this is not a reliable point of comparison. Of more interest is the reduction in processing time due to the use of the 1-pass method where we clean pixels as we calculate magnitude values resulting in a 40% improvement. Another interesting observation is that the cost of using an S3 storage bucket over a local NFS share was only about a 6% increase in overall processing time.

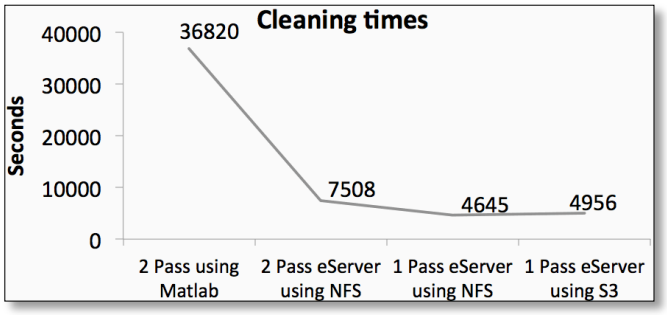


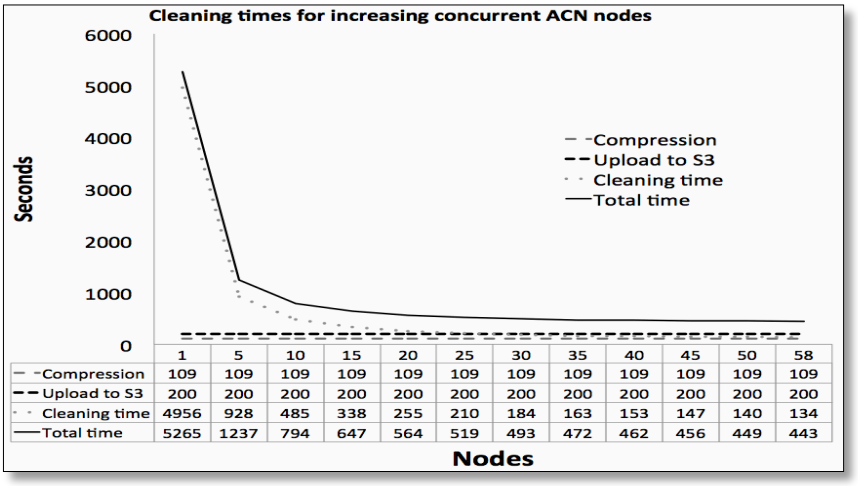
Figure 6. Time to clean data using a 1 and 2 pass method including processing times reading from local NFS or S3 storage.

5.4 ACN Pipeline performance

In Figure 7 the full pipeline processing times for the 26-gigabyte dataset is shown for a range of concurrently running ACN nodes. Significant reductions in processing time were achieved with a concurrency rate of 58 running nodes processing the entire dataset in 7 minutes 23 seconds. We can compare this to the processing time of the following

a) Original MATLAB pipeline: 443 seconds versus 36,000 second giving a saving of 98.76%
b) Initial *acn-aphot* program running in 2-pass mode: 443 seconds versus 5,520 seconds giving a saving of 91.51%

Since our dataset was captured in 1 hour and was now processed in 443 seconds we can show a saving of 87.69% with a pipeline that can clean 83 images per second. The minimum processing time for this pipeline however is now the fixed sequential time of compression (109 seconds), uploading time to S3 (20 seconds) and the time to process one file (approximately 3 seconds), giving us 312 seconds, with a possible maximum cleaning rate of ≈118 image per second. This gives us a capture rate of 10 images a second with a potential cleaning rate of 118 images per second, but with an actual cleaning rate of 83 images per second, just short of 1 order of magnitude improvement.



	1	5	10	15	20	25	30	35	40	45	50	58
Compression	109	109	109	109	109	109	109	109	109	109	109	109
Upload to S3	200	200	200	200	200	200	200	200	200	200	200	200
Cleaning time	4956	928	485	338	255	210	184	163	153	147	140	134
Total time	5265	1237	794	647	564	519	493	472	462	456	449	443

Figure 7. Cleaning times in a distributed system for varying numbers of concurrent ACN nodes showing a reduction in processing time for the dataset from over 1hr 27 minutes down to just over 7 minutes.

6. ANALYSIS

The overall aim of this paper was to design a system that could dynamically incorporate additional hardware to help reduce the cleaning time of images and to identify opportunities in processing data concurrently to achieve processing rates close to one order of magnitude faster than the capture rates for a single CCD device.

The system as designed allows any processing device capable of running Linux kernel to contribute to the overall processing of a dataset. ACN nodes can join or leave during a processing run without impacting the overall process. The memory footprint is extremely low requiring less than 2MB of memory to be available to the acn-aphot utility. ACN nodes require Internet access and run on an Ubuntu system but are easily compiled for any Unix platform. Porting to mobile devices will require a compilation of the CFITSIO or java equivalent and would require moving to an alternative queue mechanism.

The distributed process has moved the emphasis away from the parallel cleaning of data files to the sequential portions of the pipeline. The data compression and uploading are now significant percentages in the overall cleaning time, 25% and 50% respectively. The system relies upon fast compression and uploading to S3 using a Dell 410 PowerEdge server but this is the only high performance system used. The contribution made to the pipeline by single core servers which otherwise may not have been seen as viable computing devices for image processing shows how effective the distributed approach can be in reusing older devices. There are a number of possible reasons for the extended tail of the graph in Figure 7. The design relies on a single NFS share for queuing which is a shared resource, some of the additional systems were virtual machines using shared network and CPU resources so were not providing a fully independent computing resource.

The performance of the S3 data storage for uploading and downloading data was a significant improvement on attempts at building a replicating data store as it provided consistent download times to ACN nodes when retrieving data.

7. CONCLUSION AND FUTURE WORK

This paper has identified parallel elements within the raw image data and has shown that processing these in parallel can yield close to an order of magnitude faster processing times compared to capture rates for CCD devices with a resolution of 512x512. We have used Amazon's S3 storage without introducing processing bottlenecks and have used FITS compression to reduce the amount of data transferred between systems. The small *acn-aphot* memory footprint enables older machines to contribute effectively to the process and offers the opportunity to review a processing model that could incorporate mobile applications into the pipeline. It is proposed that the NFS queue used must migrate to the Amazon

Simple Queuing Service (SQS) or similar web service to remove the restriction of requiring ACN nodes to mount the NFS based queue. Further optimizations may also be possible by focusing on the clipping of stars to further reduce the size of images being moved across networks, but to overcome the file I/O bottlenecks associated with this we will require images to be stacked in cubes greater than 10, and possibly closer to 100. Further evaluation and comparison with other processing pipelines is also planned along with additional monitoring of network traffic to assist in identifying any bottlenecks in data transmission, which could be optimized. As image resolutions continue to increase more consideration will be given to image clipping and compressing for each object of interest prior to processing.

Acknowledgments The network infrastructure for the ACN Pipeline was built with generous support from members of HEAnet and the IT departments in DIT, CIT and ITTD. The development of the *acn-aphot* utility has benefited enormously from the assistance and guidance provided by Alan Giltinan of BCO, and Kevin Nolan of ITTD.

REFERENCES

[1] M. J. Graham, "Astronomy 2020: A Pragmatic Approach," 2009.
[2] N. Smith, A. Giltinan, A. O'Connor, S. O'Driscoll, A. Collins, D. Loughnan, and A. Papageorgiou, "EMCCD Technology in High Precision Photometry on Short Timescales," in *High Time Resolution Astrophysics*, vol. 351, D. Phelan, O. Ryan, and A. Shearer, Eds. Springer Netherlands, 2008, pp. 257–279.
[3] S. B. Howell, *Handbook of CCD astronomy*, vol. 5. Cambridge Univ Pr, 2006, pp. 82-110.
[4] B. Fowler, C. Liu, S. Mims, J. Balicki, W. Li, H. Do, J. Appelbaum, and P. Vu, "A 5.5Mpixel 100 frames/sec wide dynamic range low noise CMOS image sensor for scientific applications," 2010, pp. 753607–753607–12.
[5] NAOA, "IRAF Project Home Page," *IRAF*. http://iraf.noao.edu/.
[6] "ESO - Common Pipeline Library." [Online]. Available: http://www.eso.org/sci/software/cpl/.
[7] L. de Bilbao, L. K. Lundin, P. Ballester, K. Banse, C. Izzo, R. Palsa, and C. E. García-Dabó, "Multi-Threading for ESO Pipelines," presented at the Astronomical Data Analysis Software and Systems XIX, 2010, vol. 434, p. 241.
[8] L. Ostorero, S. J. Wagner, J. Gracia, E. Ferrero, T. P. Krichbaum, S. Britzen, A. Witzel, K. Nilsson, M. Villata, U. Bach, D. Barnaby, S. Bernhart, M. T. Carini, C. W. Chen, W. P. Chen, S. Ciprini, S. Crapanzano, V. Doroshenko, N. V. Efimova, D. Emmanoulopoulos, L. Fuhrmann, K. Gabanyi, A. Giltinan, V. Hagen-Thorn, M. Hauser, J. Heidt, A. S. Hojaev, T. Hovatta, F. Hroch, M. Ibrahimov, V. Impellizzeri, R. Z. Ivanidze, D. Kachel, A. Kraus, O. Kurtanidze, A. Lähteenmäki, L. Lanteri, V. M. Larionov, Z. Y. Lin, E. Lindfors, F. Munz, M. G. Nikolashvili, G. Nucciarelli, A. O'Connor, J. Ohlert, M. Pasanen, C. Pullen, C. M. Raiteri, T. A. Rector, R. Robb, L. A. Sigua, A. Sillanpää, L. Sixtova, N. Smith, P. Strub, S. Takahashi, L. O. Takalo, C. Tapken, J. Tartar, M. Tornikoski, G. Tosti, M. Tröller, R. Walters, B. A. Wilking, W. Wills, I. Agudo, H. D. Aller, M. F. Aller, E. Angelakis, J. Klare, E. Körding, R. G. Strom, H. Teräsranta, H. Ungerechts, and B. Vila-Vilaró, "Testing the inverse-Compton catastrophe scenario in the intra-day variable blazar S5 0716+71. I. Simultaneous broadband observations during November 2003," *Astronomy and Astrophysics*, vol. 451, pp. 797–807, Jun. 2006.
[9] H. Ferguson, P. Greenfield, T. Axelrod, S. B. RIT, A. Conti, D. Crabtree, E. Feigelson, M. Fitzpatrick, W. Freedman, K. Gillies, and others, "Astronomical data reduction and analysis for the next decade," *The Astronomy and Astrophysics Decadal Survey*, 2009.
[10] "FITSIO Home Page." . http://heasarc.gsfc.nasa.gov/fitsio/.
[11] "wphot.". http://stsdas.stsci.edu/cgi-bin/gethelp.cgi?wphot#algorithms.
[12] W. D. Pence, R. Seaman, and R. L. White, "Lossless Astronomical Image Compression and the Effects of Noise," *Publications of the Astronomical Society of the Pacific*, vol. 121, pp. 414–427, Apr. 2009.