Accelerating Satellite Image Processing through Cloud Computing

Kutila Gunasekera, Jane Hunter
School of Information Technology & Electrical Engineering
The University of Queensland
Brisbane, Australia
{k.gunasekera; j.hunter}@uq.edu.au

Marites Canto, Scarla Weeks
School of Geography, Planning & Environmental Management
The University of Queensland
Brisbane, Australia
{m.canto; s.weeks}@uq.edu.au

***Abstract*—This paper describes the deployment of an array of Web-based satellite image processing pipelines (used by oceanographers to extract and analyse ocean colour data) onto Microsoft's Azure cloud computing platform. The paper firstly describes the background and related work. It then describes the technical architecture and migration of the NASA SeaDAS image processing pipelines to Azure. Thirdly it describes the user interface that allows oceanographers to configure their processing pipelines and visualize the results. Finally it describes the results of a series of experiments that assess the scalability and improvements in performance that are achievable by distributing the image processing tasks across multiple Azure Virtual Machine (VM) instances.**

***Keywords*— satellite image processing, oceanography, cloud computing, parallel processing**

I. INTRODUCTION

Remote sensing data captured via satellites provides oceanographers with an extremely valuable source of information. For example, oceanographers study the impact of climate change and human activities on the ocean by analyzing data products such as ocean colour, sea surface temperature and sea surface salinity, which are extracted from remote-sensing satellite imagery [1]. One of the principle tools that oceanographers employ to extract data from satellite imagery is NASA'a SeaDAS image analysis package [2]. SeaDAS enables researchers to interactively process satellite images from a range of missions, apply quality control and visualize and interpret ocean colour data over short, medium and long term spatio-temporal scales. Ocean colour data (also known as chlorophyll-a) provides them with an indication of the concentration of the ocean's optical constituents such as phytoplankton or suspended sediments.

However oceanographers face a number of challenges in processing the large volumes of satellite imagery that are generated daily by orbiting instruments such as the Coastal Zone Color Scanner Experiment (CZCS), the Ocean Color and Temperature Scanner (OCTS), MODIS Aqua and SeaWiFS [3]. Scientists need robust and easily adaptable workflows capable of downloading, storing, indexing, and curating the large volumes of remote sensing imagery that is captured daily. They need to be able to specify regions, time periods and data products of interest. They need access to re-usable workflows that they can tweak for particular scenarios. They need massive computational power to process the images and deliver results to their desktops in near real-time. And they need backend systems that automatically store the input and output datasets and capture the provenance of the derived data products generated by the processing pipelines.

The OceanSpace system described in this paper is the result of a collaboration between the School of ITEE eResearch Lab and the Biophysical Oceanography Group at the University of Queensland. The aim of the project is to overcome the challenges outlined above and support the oceanographers' requirements by deploying SeaDAS image processing pipelines on Microsoft's Azure cloud computing platform.

The remainder of the paper is structured as follows. Section II describes related work. Section III describes the technical architecture of the OceanSpace system. Section IV describes the migration of SeaDAS pipelines to Azure. Section V describes the user interface. Section VI describes the evaluation experiments that were carried out to assess the improvements in performance that were achievable using Azure. Section VII provides a discussion of the results and highlights limitations and fertile areas for future research. Section VIII provides brief concluding remarks.

II. RELATED WORK

The most similar related work to the research outlined in this paper is the ModisAzure system [4,5, 14]. MODISAzure is a MODIS satellite data reprojection and reduction pipeline built on the Azure cloud computing platform [6]. The pipeline consists of three stages: *data collection*, *reprojection*, and *analysis and reduction*. In the *data collection* stage, data is downloaded to the Azure instance(s) from the Azure Blob storage or from remote NASA repositories as required. In the second stage, data is *reprojected* to homogenize it before further processing can occur. The *analysis and reduction* stage executes custom algorithms provided by scientists which have been implemented as self-contained Windows command-line executables. MODISAzure exploits the inherently parallel nature of the reprojection and reduction tasks, to accelerate

DOI: 07. ICCTIS.2015.1.6

these processing steps by distributing them over multiple Azure instances.

Compared with ModisAzure, OceanSpace is much more flexible and extensible. It supports a much more diverse, and customizable set of data processing pipelines and provides oceanographers with the ability to add new pipelines and processing algorithms. Since OceanSpace is built on top of NASA's SeaDAS software package, it is capable of supporting data from sources other than MODIS instruments, and leveraging their plug-in capabilities to support new data processing functionalities. This requires the development of plug-ins using either Java or Python programming language. The challenge is that different pipeline types in OceanSpace inherently possess different levels of internal parallelism, so each pipeline needs to be asessed individually in terms of its ability to scale its processing steps over multiple Azure instances.

In [7], Zinn et. al. apply a similar approach using the Azure platform to process Geostationary Operational Environmental Satellite (GOES) data and calculate daily evapotranspiration coefficients. Their workflow includes uploading satellite data from a desktop computer to Azure's Blob storage, and then copying data from Blob storage to Azure virtual machines for use in the workflow. It was found that uploading data to Blob storage was time consuming and the solution was to stream data from the desktop computer to the process on Azure virtual machines. The Blob storage was only used to publish results. In OceanSpace, satellite data products are downloaded from NASA servers and stored on the Azure Blob storage in advance (i.e. daily). Therefore, when a pipeline is being executed on-demand, data only needs to be downloaded from the Blob storage. Hence, OceanSpace users do not experience a delay due to the need to retrieve data from the cloud storage.

III. TECHNICAL ARCHITECTURE

The OceanSpace system consists of three main components as shown in Figure 1:

- The Download Manager module consists of a set of scripts that periodically download satellite data from NASA servers (NASA Ocean Biology Distributed Active Archive Centre (OBDAAC)) [8];
- The Spectral Image Repository- is the local storage for the downloaded satellite images and workflow outputs;
- The Web Data Processor module - contains the Web-based on-demand satellite data processing system. Users access data processing workflows through the Web interface and interactively select, configure (using sub-form config files and pipeline template XML files) and execute them.

OceanSpace uses the SeaWiFS Data Analysis System (SeaDAS) software [1] as its underlying satellite data processing engine. SeaDAS is an open source visualization, analysis and data processing package for satellite ocean colour data developed by the NASA Ocean Colour Biology Processing Group (OBPG). The main visualization and analysis components of SeaDAS are based on the European Space Agency's BEAM toolbox [9]. SeaDAS provides plug-in points for adding new extension modules as well as a Java API (Application Programming Interface) for creating remote sensing related applications.

The Graph Processing Framework (GPF) [10] is the programming model provided to incorporate new data processors into SeaDAS. Within the GPF, a satellite image processing workflow is represented as a directed acyclic graph (DAG) with processing occurring at the nodes of the graph. The framework controls graph execution and processes images in smaller tiles to parallelize execution where possible and take advantage of multi-core CPUs. Processing nodes of a graph are implemented as GPF operators. SeaDAS contains a number of GPF operators which can perform tasks such as Read/Write data products from/to disk, extract sub-regions from a product, and perform various mathematical functions. In addition, it is possible to implement new data processing algorithms as GPF operators in Java and Python programming languages. OceanSpace makes use of this feature and implements a number of GPF operators to perform specific data processing tasks required for its workflows. Customized data analysis algorithms, not currently supported, can easily be added to the OceanSpace workflows, by developing new GPF operators.

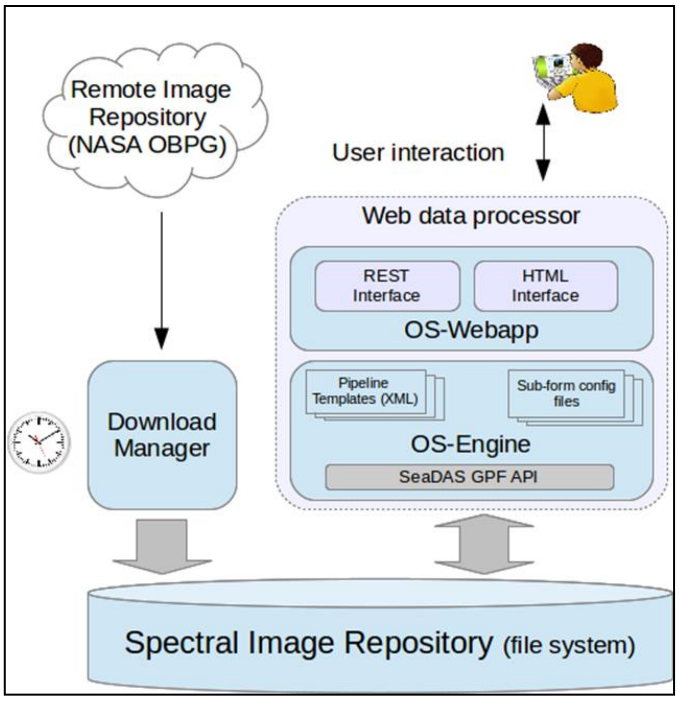


Fig. 1. System Architecture

IV. EXECUTING SEADAS PIPELINES ON AZURE

Because SeaDAS is not a distributed application and can only read data products from a file system, the Virtual Machine (VM) model was identified as the suitable Azure compute service. Ubuntu 12.04 LTS was selected as the VM Operating System due to its widespread adoption and long term support. The next two sub-sections below outline the challenges and corresponding solutions associated with managing storage and scaling pipelines over multiple Azure instances.

A. *Storage Issues*

The MODIS satellite data products used in OceanSpace are provided by NASA in a compressed format (bzip2) due to their large size. For example, a daily MODIS Aqua "Chlorophyll" product for the Great Barrier Reef region at a 1km resolution can be around 25MB when uncompressed. With bzip2 compression, the file shrinks to less than 2MB. Therefore, the OceanSpace Download Manager downloads and stores these data products in their compressed format in order to minimize disk space usage. They are extracted to disk by the Web Data Processor prior to being used. The time taken for extracting data products is non-negligible and can cause a considerable delay in pipelines when reading a large number of data products. Hence this is performed daily for the area under investigation (Great Barrier Reef) and stored on Azure storage.

OceanSpace also needs disk space to store its outputs which are primarily in the form of satellite data products (in HDF, NetCDF formats) and images (in PNG and JPG formats). Currently these output files are stored in a temporary "cache" which is periodically deleted. But if the outputs are to be stored persistently, then output storage becomes an issue.

The oceanographers from the University of Queensland's Biophysical Oceanography Group have currently configured OceanSpace to download a subset of MODIS satellite data covering the Great Barrier Reef region. At present, this amounts to 161GB of data, in 104,643 (compressed) files covering the 2000 to 2015 (current) time period. It is possible that this subset will be expanded to include more data types as well as more geographic regions in the future. Altogether, OceanSpace at present contains close to 700GB of data and it is expected that the storage requirements of OceanSpace for the next four years to be in the range of 4-5 terabytes.

The Azure platform provides several data storage offerings: local storage, block Blobs, page Blobs, tables, queues and an SQL database [11]. Of these, the high-performing temporary local storage and Blob storage are inherently suited for the OceanSpace storage requirements. However, block Blobs cannot be accessed as files on disk, which is a requirement of OceanSpace. A page Blob on the other hand can be mapped as a disk on an Azure virtual machine. Therefore, a first option considered for OceanSpace on Azure was to use page Blobs to store compressed satellite data products and to expand them to local storage prior to use. However, a Blob has a throughput limit of 60MB per second or 500 requests per second. Since with page Blobs, multiple data products are stored in a single Blob and may need to be read by a pipeline, this limit proved to be a speed bottleneck.

It was therefore decided to use block Blobs to store compressed data products and implement a pre-processing procedure which would download and extract the required products to local storage when a pipeline execution is requested. Data products extracted and stored on the local storage are available for use by subsequent pipelines until the VM restarts or runs out of space, at which time the extracted data products must be deleted. The Download Manager module was also modified to store downloaded satellite data products in the Azure Blob storage. This has the additional benefit of being a central storage accessible by several OceanSpace instances distributed over multiple Azure Virtual Machines (VMs).

B. *Scaling over Azure*

A primary advantage of cloud computing is the ability to easily and dynamically increase the hardware resources used by an application/service as the need arises. Scaling OceanSpace over multiple virtual machines can be useful in two scenarios:

- To distribute a single pipeline's execution over multiple virtual machines (if the pipeline is made up of tasks which can be executed in parallel);
- When multiple pipeline executions are requested simultaneously, they can be distributed to separate virtual machines.

In both these situations, reductions in pipeline execution times can be expected as the workload is distributed over multiple hardware resources. However, scaling pipeline execution over multiple virtual machines requires modifying the Web data processor module illustrated in Figure 1. The OceanSpace Web data processor module was redesigned as a scalable application that can be distributed over multiple computers using software agents. The new design employs software agents, a proven paradigm for distributed computing, autonomous and social entities that sense and react to the environment while working towards achieving specific goals.

An architectural overview of the redesigned approach is shown in Figure 3. The updated system consists of three primary modules: a Web application, Worker Agents and the Download Manager. The Web application (OS-WebApp) accepts user requests and generates SeaDAS graphs. Multiple distributed Worker Agents are capable of executing SeaDAS graphs simultaneously. A software agent (Front Agent) embedded in the Web application communicates with the Worker Agents using asynchronous messages and assigns graphs/tasks to them. The Download Manager periodically downloads satellite data products from the NASA servers and stores them in the Azure Blob storage. Output products and other files generated as a result of graph execution are stored back to Blob storage where they can be accessed by the Web application.

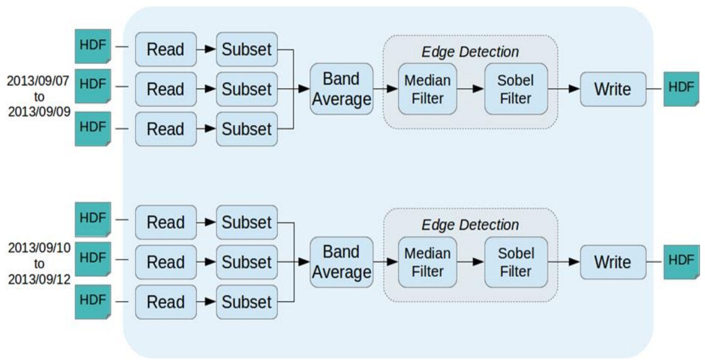


Fig. 2. The Pipeline corresponding to Pixel Gradient Intensity Computation

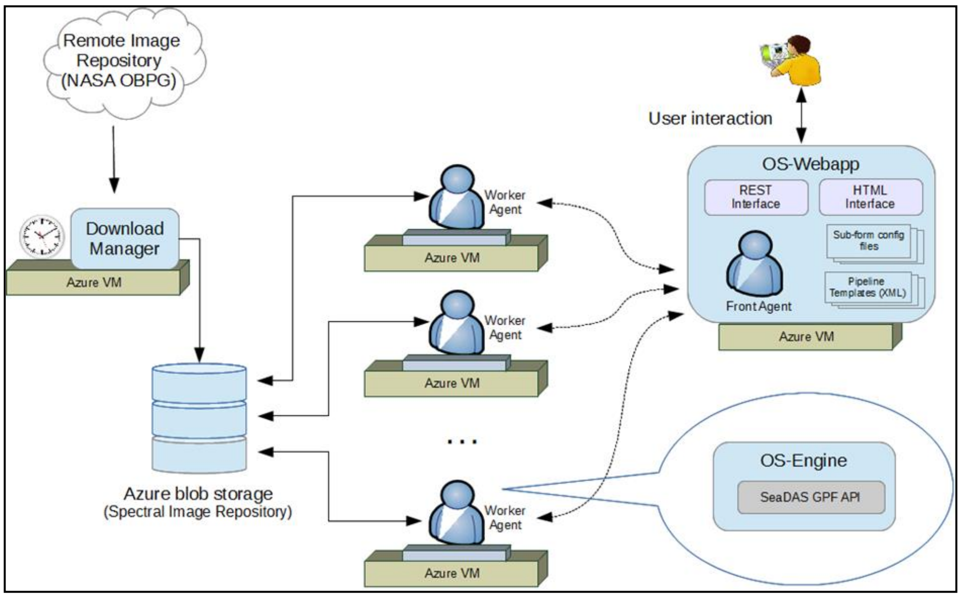


Fig. 3. Scaling of SeaDAS Pipelines over Azure

The use of software agents reduces OceanSpace's dependency on Azure-specific services (e.g. Azure queues, workers), easing the process of migrating OceanSpace to a different platform if needed. For example, OceanSpace could be deployed on a local cluster of computers with only minor modifications required to replace Azure Blob storage with an alternative storage service.

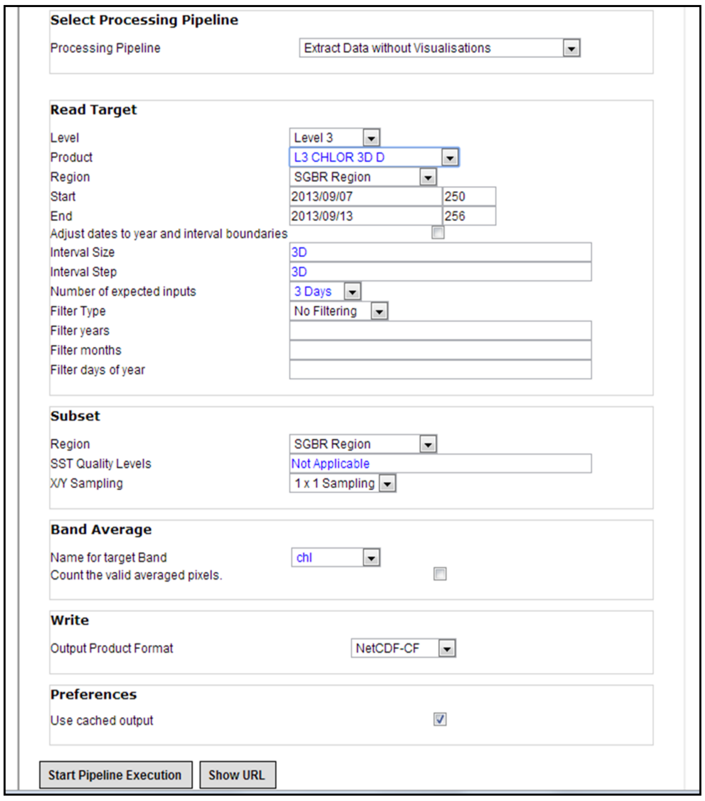
V. USER INTERFACE

From an end users' perspective, the first step involves logging into the OceanSpace Web portal using a valid login ID and password. Users then select a processing pipeline from a drop-down list. Frequently used pipelines that are supported (in order of increasing computational demand) include:

- *Extract and Average* – extracts a sub-region from a larger area and averages selected data products over the region for a given time interval. This is useful for reducing data quality issues such as missing data or cloud cover and produces both HDF files and PNG images.
- *Mean and Interpolate* – extends the Extract and Average pipeline by interpolating any missing pixels.
- *Pixel Gradient Intensity Computation* – this pipeline enables the detection of oceanic fronts [12] via edge detection performed by applying Median and Sobel filters [13] (see Figure 2).

Based on the selected pipeline, a Web page is displayed that lists user-configurable parameters for the selected pipeline. These are in order of processing steps with default values provided where relevant (Figure 4). Reconfigurable parameters typically include product type, region of interest (e.g. Southern Great Barrier Reef), time period of data products to process, output data product format, and other parameters specific to the data processing tasks involved (e.g., Band Average).

After inputting the required parameter values, the user invokes execution of the pipeline by pressing the "Start Pipeline Execution" button at the bottom of the page. At this point, a request is sent to the OceanSpace server which generates one or more SeaDAS XML graphs (based on the user provided parameters), which are assigned by the FrontAgent to WorkerAgents which concurrently execute them on Azure VMs using SeaDAS GPF.



Select Processing Pipeline

Processing Pipeline	Extract Data without Visualisations

Read Target

Level	Level 3	
Product	L3 CHLOR 3D D	
Region	SGBR Region	
Start	2013/09/07	250
End	2013/09/13	256
Adjust dates to year and interval boundaries	☐	
Interval Size	3D	
Interval Step	3D	
Number of expected inputs	3 Days	
Filter Type	No Filtering	
Filter years		
Filter months		
Filter days of year		

Subset

Region	SGBR Region
SST Quality Levels	Not Applicable
X/Y Sampling	1 x 1 Sampling

Band Average

Name for target Band	chl
Count the valid averaged pixels.	☐

Write

Output Product Format	NetCDF-CF

Preferences

Use cached output	✓

Start Pipeline Execution | Show URL

Fig. 4. Choosing and Configuring a Pipeline

Once the execution has completed, results are displayed in multiple tabs at the bottom of the page (Figure 5). These tabs include links to:

- Downloadable outputs (if any);
- The XML graph generated and executed by GPF/OceanSpace;
- Logs generated during pipeline execution;
- Image outputs (if the pipeline generated images).

Users also have the option to visualize the images as side-by-side tiles or to animate the generated images by displaying them in sequence at 5 frames per second (Figure 5).

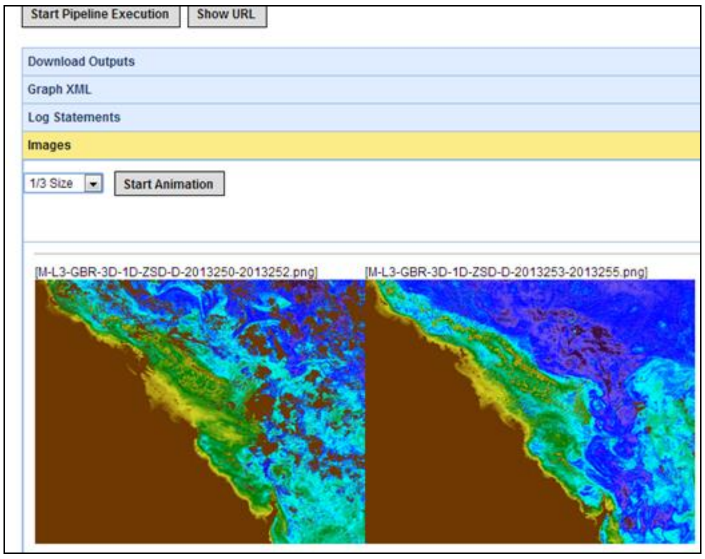


Fig. 5. User Interface showing display of Output Images

OceanSpace also supports the specification of new data processing pipelines by modifying a set of configuration files on the OceanSpace server and saving them to a new pipeline. This may require the development of new GPF operators.

VI. EVALUATING PERFORMANCE ON AZURE

The main objective of the experiments described here was to compare the performance of OceanSpace when deployed on a moderately powerful desktop computer with its performance when deployed on an Azure compute-intensive virtual machine. The measure of performance used is "time consumed to execute a data processing pipeline".

Table I. Test computer configurations

Desktop computer	Azure virtual machine
CPU: Intel core i7-3770 4 cores* @ 3.4GHz RAM: 8GB Disk: 1TB SATA Network: 1Gbps OS: Ubuntu 12.04 (*8 cores with hyperthreading)	CPU: Intel Xeon E5-2670 16 cores @ 2.6 GHz RAM: 112GB Disk: 382GB temporary resource disk Network: 10Gbps OS: Ubuntu 12.04

Since the Azure servers that we used are located in the US West region, and accessed from Australia, there is a noticeable latency in communicating with them. However, our focus is on pipeline execution performance, and therefore we only measure time from the moment the server receives a processing request until execution completes. The configurations of the two test machines used for experiments are given in Table I.

The experimental workflow selected for evaluation the Pixel Gradient Intensity computation pipeline to compute 3-day composite chlorophyll levels over the Southern Great Barrier Region for one month (September 2011). This is the most computationally-intensive workflow from the three types used in our evaluations. With the given parameters, it reads 30 input products, produces 10 output products and has 10 parallel execution paths within its graph.

A. Experiment I

The first experiment involved running the Pixel Gradient Intensity Computation pipeline on both the Desktop computer and on a single Azure VM (with 16 cores), varying the size of the median filter from 1 to 36 (in increments of 4) and measuring the corresponding pipeline execution time.

The graph in Figure 6 plots average pipeline execution time in the two environments as filter size increases. While processing time increases with filter size as expected, the results clearly show that executing the pipeline on Azure VM takes less than half the time to execute as the Desktop across all median filter sizes.

In this experiment, the number of input and output products remained constant through all runs, and the number of parallel execution paths also remained unchanged at 10. Increasing filter size increased the size of the square kernel around each pixel to be used by the median filter. Therefore, the processing requirements were varied in this experiment. With more CPU cores and memory at its disposal, the Azure VM comfortably outperforms the desktop computer in this scenario.

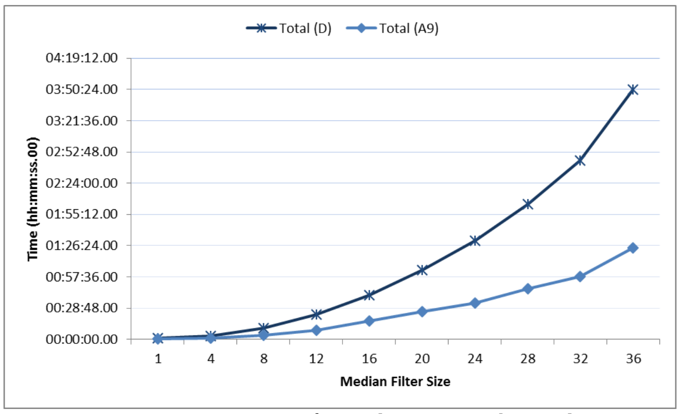


Fig. 6. Comparing execution times for pipelines on a Desktop and Azure VM

B. Experiment II

The second experiment involved running the one month pixel-gradient intensity computation pipeline described above, on an OceanSpace deployment which distributed the 10 parallel processing paths associated with this pipeline, across 1-8 A8

Azure Virtual Machines. The average time consumed to complete all tasks was recorded.

Figure 7 illustrates that the pipeline completion time falls as the number as virtual machines increases from 1 to 8. The 10 parallel execution paths are distributed as evenly as possible over the available number of virtual machines. For example, if there are 3 VMs, then 4 tasks (graph fragments) are assigned to VM1, 3 tasks to VM2 and 3 tasks to VM3. If there are 4 VMs available, then 3 tasks are assigned to each of VM1 and VM2 and 2 tasks are assigned to each of VM3 and VM4.

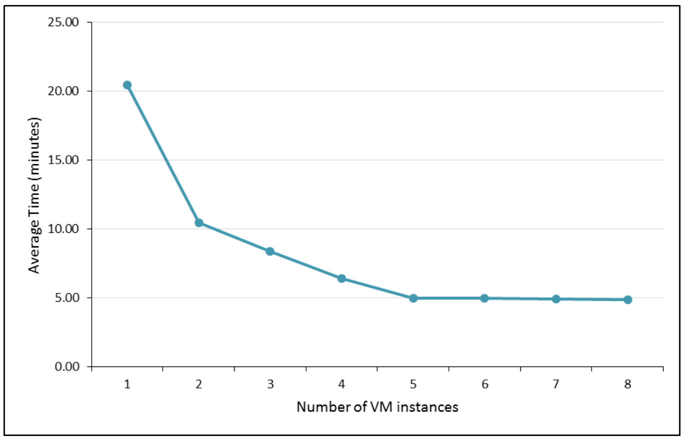


Fig. 7. Pipeline execution time versus number of Azure VMs

It can be observed that distributing the workload over multiple virtual machines reduces the time of pipeline execution, but the improvement in performance levels off after the first 5 virtual machines. The reason for this behavior is that the time taken to complete the entire workflow is equivalent to the longest time needed i.e., the virtual machine allocated the most tasks (as shown in the equation below).

$$T_{completion} = maximum(t_{vm1}, t_{vm2}, t_{vm3}, \ldots)$$

Thus, adding the second virtual machine almost halves the required time as the workload of an individual virtual machine is reduced by 50% from 10 to 5. Adding the 3rd and 4th virtual machines reduce the maximum workload of a virtual machine by 1 graph fragment. From 5th to 8th virtual machines, the maximum workload remains at 2 graph fragments, and therefore does not generate a noticeable reduction in the total time. Accordingly, we can expect a reduction at 10 virtual machines (because there are 10 parallel execution paths) but no further time reductions thereafter.

VII. DISCUSSION

The first experiment described in Section VI compares performance of the compute-intensive pixel gradient intensity computation pipeline on Azure with its performance on a standard desktop computer. The experimental results showed that the more powerful Azure VM is able to better handle increased processing needs much better than the Desktop.

A further experiment where the number of parallel execution paths in a graph was varied showed that OceanSpace on Azure still performs better than on our local hardware irrespective of the parallelism of the pipeline. Experiment 2 was designed to evaluate how OceanSpace performs as it is scaled up to use multiple Azure VMs. This experiment showed that performance improves when it is possible to distribute the workflows over multiple VMs.

Additional experiments were also conducted to evaluate the system performance when running *Extract & Average* and *Mean & Interpolate* pipelines. These experiments indicated that the Azure VM is more capable of handling pipelines where large volumes of input data needs to be processed, while the Desktop computer struggled with large volumes of input data.

The three SeaDAS pipelines that were specifically evaluated (*Extract&Average*, *Mean&Interpolate*, *PixelGradientIntensity Computation*) all possess an inherent level of parallelism which allow the XML graphs to be split and distributed over multiple virtual machines. However, not all workflows have multiple paths of execution which allow distribution over multiple virtual machines. For example, workflows to generate Hovmoeller diagrams or Standard Deviation data products do not contain multiple execution paths. Hence, an OceanSpace workflow may not always benefit from the availability of multiple virtual machines.

VIII. CONCLUSION

The research described in this paper investigates the deployment of SeaDAS image processing pipelines to the Azure cloud computing platform and presents the results of experiments designed to evaluate achievable improvements in performance. The evaluations to date indicate that performance improvements up to a factor of approx. 4 are achievable by scaling a set of predefined, commonly used pipelines over multiple (2-8) Azure virtual machines. The limitation is that currently the sub-division and distribution step is performed manually by assessing the individual pipelines and sub-tasks. The scaling up process has not been automatically integrated into the OceanSpace system. Future work aims involve designing and implementing mechanisms which can automatically and dynamically analyse newly defined pipelines, split them where possible into concurrent sub-tasks and distribute their execution over multiple virtual machines, to optimize the performance on-the-fly.

ACKNOWLEDGMENT

The authors of this paper are extremely grateful to Microsoft Research for providing us with research funding and access to the Windows Azure Platform for the purposes of this research project.

REFERENCES

[1] Weeks, S.; Werdell, P.J.; Schaffelke, B.; Canto, M.; Lee, Z.; Wilding, J.G.; Feldman, G.C. Satellite-Derived Photic Depth on the Great Barrier Reef: Spatio-Temporal Patterns of Water Clarity. Remote Sens. 2012, 4, 3781-3795

[2] Feldman, Gene C, “SeaDAS,” 2014. [Online]. Available: http://seadas.gsfc.nasa.gov. [Accessed 14/05/2015].

[3] NASA Goddard Space Flight Center, SeaWiFS Project. 2014 [Online]. Available: http://oceancolor.gsfc.nasa.gov/SeaWiFS/ [Accessed 14/05/2015].

[4] Li J, Agarwal D, Humphrey M, van Ingen C, Jackson K and Ryu Y (2010) eScience in the cloud: a MODIS satellite data reprojection and reduction pipeline in the windows azure platform. In: Proceedings of the International Parallel & Distributed Processing Symposium, IEEE International Symposium on , vol., no., pp.1,10, 19-23 April 2010

[5] MODIS Web, NASA, 2014. [Online]. Available: http://modis.gsfc.nasa.gov. [Accessed 14/05/2015].

[6] Microsoft Azure, [Online]. Available: http://www.azure.microsoft.com. [Accessed 14/05/2014].

[7] D. Zinn, Q. Hart, B. Ludascher and Y. Simmhan, "Streaming Satellite Data to Cloud Workflows for On-Demand Computing of Environmental Products," in 5th Workshop on Workflows in Support of Large-Scale Science (WORKS), 2010.

[8] NASA OBDAAC, 2014. [Online]. Available: https://earthdata.nasa.gov/data/data-centers/obpg [Accessed 14/05/2015]

[9] BEAM Earth Observation Toolbox and Development Platform, Brockmann Consult, 2014. [Online]. Available: http://www.brockmann-consult.de/cms/web/beam/. [Accessed 14/05/2015].

[10] The BEAM Graph Processing Framework (GPF), Brockmann Consult, 2014. [Online]. Available: http://seadas.gsfc.nasa.gov/help/. [Accessed 14/05/2015].

[11] Microsoft, Data Storage Offerings on the Azure Platform, 2014. [Online].Available: http://social.technet.microsoft.com/wiki/contents/articles/1674.data-storage-offerings-on-the-azure-platform.aspx [Accessed 14/05/2015].

[12] P. I. Miller, "Detection and Visualisation of Oceanic Fronts from Satellite Data, with Applications for Fisheries, Marine Megafauna and Marine Protected Areas," in Handbook of Satellite Remote Sensing Image Interpretation: Applications for Marine Living Resources Conservation and Management, J. Morales, V. Stuart, T. Platt and S. Sathyendranath, Eds., Dartmouth, Canada, 2011.

[13] I. Sobel, "An Isotropic 3x3 Image Gradient Operator," 2014. [Online]. Available:https://www.researchgate.net/publication/239398674_An_Isotropic_3_3_Image_Gradient_Operator [Accessed 14/05/2015].

[14] M. Humphrey, Z. Hill, K. Jackson, C. van Ingen and Y. Ryu, "Assessing the Value of Cloudbursting: A Case Study of Satellite Image Processing on Windows Azure," in Seventh IEEE International Conference on eScience, 2011.