SkyLab : An extensible workflow web application for HPC on the cloud

Vincent Paul L. Carpio
Institute of Computer Science
College of Arts and Sciences
University of the Philippines Los Banos
vlcarpio@up.edu.ph

Katrina Joy M. Abriol-Santos
Institute of Computer Science
College of Arts and Sciences
University of the Philippines Los Banos
khmagno@up.edu.ph

Joseph Anthony C. Hermocilla
Institute of Computer Science
College of Arts and Sciences
University of the Philippines Los Banos
jchermocilla@up.edu.ph

ABSTRACT

Most scientific applications require high performance computing (HPC) which utilizes parallel processing to run tasks quickly and efficiently. MPI clusters are often used to cater this type of tasks but the hardware required can be costly. Peak-Two Cloud (P2C) can host HPC applications in a cloud environment which is relatively cheaper due to resource reuse and more convenient due to on-demand provisioning. One of the key features of P2C is vCluster, a tool that can deploy MPI clusters usable through the command line. In this paper, we present the design, implementation, and user-evaluation results of SkyLab, a workflow web application on top of vCluster to simplify the process of running MPI applications for users not accustomed to the command line. SkyLab currently supports applications used in bioinformatics, molecular dynamics, molecular docking, and quantum chemistry. The extensible design of SkyLab enables additional tools to be incorporated easily as modules.

CCS CONCEPTS

•Information systems → Computing platforms; •Networks → Cloud computing;

KEYWORDS

high performance computing, workflows

ACM Reference format:
Vincent Paul L. Carpio, Katrina Joy M. Abriol-Santos, and Joseph Anthony C. Hermocilla. 2017. SkyLab : An extensible workflow web application for HPC on the cloud. In *Proceedings of Philippine Computing Science Congress, Cebu City, Philippines, March 2017 (PCSC2017),* 6 pages.
DOI:

1 INTRODUCTION

Cloud computing has marked significant developments and possibilities in the industry. It focuses on offering services for the different needs of the modern society.

There are three categories of cloud computing services namely, Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). Organizations provide SaaS depending on the demand. Google Apps is one example of SaaS that can be used to manage email and create documents, etc. PaaS offers developers a platform where they can build and deploy applications. IaaS provides storage, servers and clusters. These tools are primarily created to serve computational needs [1]. A cloud computing platform dynamically allocates, configures, reconfigures and deallocates servers as requested or on demand. This approach ensures the elasticity of cloud computing [3].

Most scientific applications require high performance computing (HPC) which needs CPU intensive computations and large data storage. To be able to host such applications, several computers interconnected in a network such as clusters are needed. This makes scientific computing very costly in terms of hardware infrastructure investment. With the advancement in cloud computing, these scientific applications can be deployed in the cloud without worrying about hardware costs and maintenance [1]. However, studies have shown that network transmission delay is the major drawback in deploying HPC applications in the cloud[3].

Peak-Two Cloud (P2C) is a private cloud based on OpenStack designed for research in deploying HPC applications in the cloud[8]. One of the features introduced by P2C is vCluster. vCluster is a tool that enables a user to deploy a working (Message Passing Interface) cluster on demand and to terminate it after use. P2C has been used by researchers in various fields including bioinformatics, quantum chemistry, and molecular dynamics. These researchers belong to different research groups who have little or no investment in HPC infrastructure due to limited funding but requires heavy computing resources for their research. vCluster, however, is a command line application which make it difficult for non-technical users (physicists, chemists, and biologists) to use. A more user-friendly interface is needed in order to enable scientists to focus more on their science rather than on learning and using the command line.

Presented in this paper is SkyLab[1], a workflow web application on top of vCluster that addresses the concern above. Specifically, SkyLab will

(1) allow users to execute HPC tools via web interface;
(2) enable developers to easily extend it to support additional HPC tools;
(3) enable users to share their instantiated clusters; and
(4) support displaying of results using third party tools.

The following are the tools that are currently supported by SkyLab. These are commonly used by collaborators from different research groups.

- *AutoDock* - A software used to simulate protein-ligand docking[13].

[1]https://github.com/vincentpaul12/SkyLab

- *AutoDock Vina* - A software similar to AutoDock 4 but on the average, it provides faster and more accurate computations [17].
- *DOCK* - Used to predict the small molecule-target interactions [11].
- *Quantum ESPRESSO* - An integrated software suite of tools for ab-initio molecular dynamics (MD) simulations and electronic structure calculations[6].
- *GAMESS* - Used for ab initio molecular quantum chemistry [14].
- *Ray* - Uses parallel genome assemblies for parallel DNA sequencing [2].
- *Impi* - MPI implementation of some image processing routines [16].

The use case on which SkyLab was designed is shown in Figure 2. Users must first create a cluster then activate the desired tool to use.

2 DESIGN AND IMPLEMENTATION

SkyLab is implemented as a web application (using the Django Web Framework[2]) to provide users access to their HPC applications through a web browser. This makes it challenging to implement given that multiple tools must be supported. Also, HPC applications executed through SkyLab have their own process space, separate from the process on which SkyLab is running. This makes it difficult to keep track of the applications and may even pose security threats. Django was selected because it is written in Python which has a vast collection of libraries for interacting with operating system services needed by SkyLab.

Figure 1 shows the layers on which SkyLab is built on. At the bottom layer is *P2C* which provides the cloud infrastructure. *vCluster* is for on-demand provisioning and termination of MPI clusters. *p2c-tools* is the command-line tool for activating the required HPC tool.

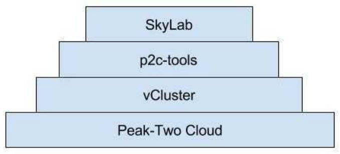


Figure 1: Layers on which SkyLab is built on.

2.1 User Access

Access to SkyLab is restricted to individuals with *@up.edu.ph* email (hosted by Google) only. This is implemented using the django-allauth[3] package. Tables used with *socialaccount_* prefix, as shown in Figure 3, are used for authentication. Users logged in to this email can directly access SkyLab.

[2]https://www.djangoproject.com/
[3]http://www.intenct.nl/projects/django-allauth/

2.2 Bootup

SkyLab is deployed using the Apache Web Server via the *mod_wsgi* module. During bootup, toolsets are first loaded (by parsing the *modules* directory) and logging is initialized. As a final step, an instance of *MPIThreadManager* class is created. *MPIThreadManager* (see bootskylab.py) is responsible for maintaining the connection to the node on which vCluster can be invoked. It opens a shell used to issue commands to the vCluster node and then creates an *MPIThread* for each MPI cluster in the database.

2.3 MPI cluster management

An *MPIThread*(see bootskylab.py) is spawned for each active cluster which handles the connection to the assigned cluster via Secure Shell (SSH). Creation and deletion of clusters is done by using *vCluster* commands while tool activation is done by using *p2c-tools*. The thread also manages task queuing and execution. Table *skylab_mpicluster* in Figure 3 stores data on MPI clusters.

A cluster is either classified as public or private. If it is set to public, every user in the system can use it. On the other hand, for private clusters, the cluster will only be visible to the owner. The owner has the option to share the cluster to other users via the share key generated for the said cluster.

2.4 Toolsets

As described in the Bootup section, the system searches for packages inside the *modules* directory and load them at bootup. The toolsets will then be available for use with the system. The package must have a module named *install.py* which contains function calls for integrating the package to SkyLab. The package must also contain the corresponding *views* and *executable* classes for each tool. Tables *skylab_toolset* and *skylab_tool* in Figure 3 stores data for the tools.

The object-oriented design of SkyLab makes it easy to extend. All toolset classes must inherit from the *P2CToolGeneric* class (see basetool.py) and override the following methods to handle tool specific operations.

- *test_ssh_connection* - Tests the SSH connection to the MPI cluster hosting the tool.
- *sftp_file_transfer_callback* - Callback method after an sftp operation. sftp is used to transfer files to and from a cluster.
- *clear_or_create_dirs* - Create task directories on the cluster.
- *run_commands* - Invoke the tool specific commands on the cluster. Must be overridden.
- *handle_input_files* - Process the input files for the tool. Must be overridden.
- *run_tool* - Method that gets executed in a task. Must be overridden. Listing 1 shows an example implementation.
- *handle_output_files* - Process output files. Must be overridden.

Listing 1: run_tool method implementation for GAMESS.

```
def run_tool(self, **kwargs):
    self.task.change_status
        (status_msg='Task started',
        status_code=150)
    self.clear_or_create_dirs
```

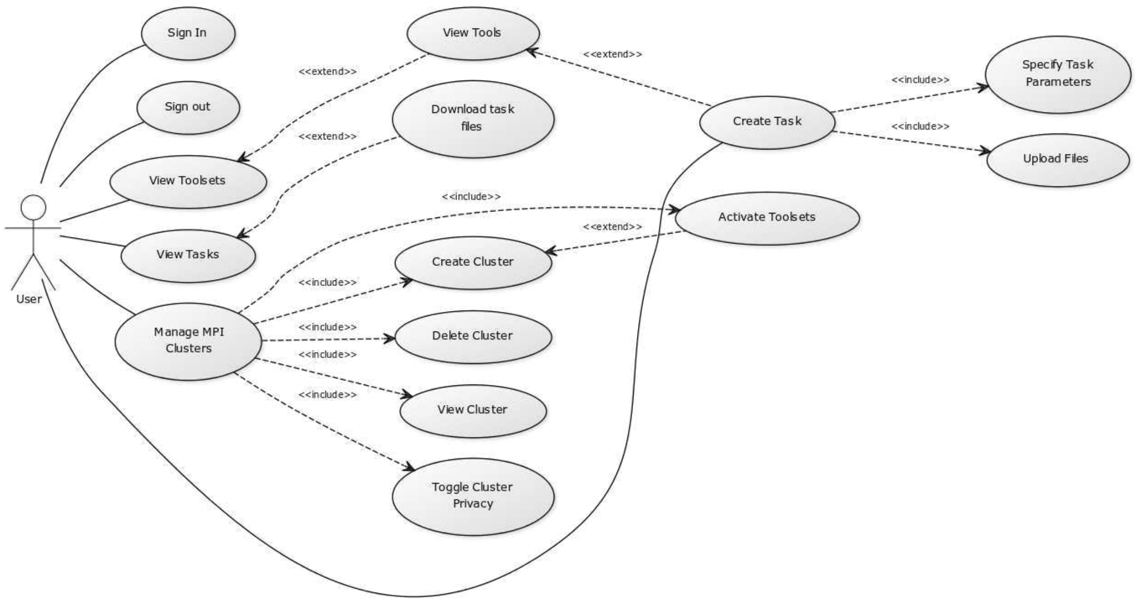
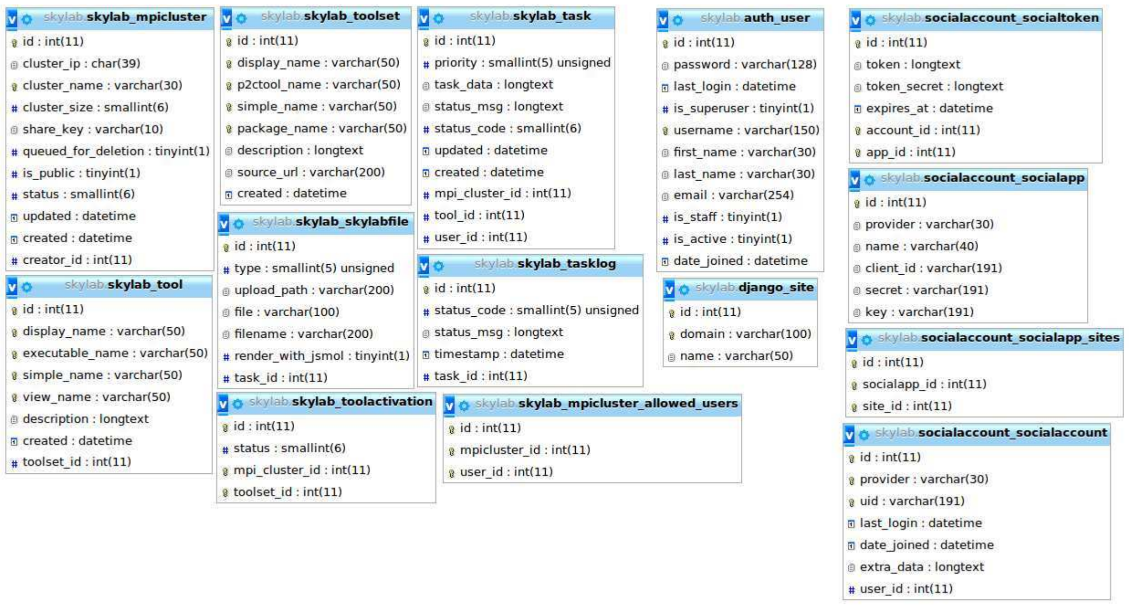


Figure 2: Use case diagram.


skylab.**skylab_mpicluster**
id : int(11)
cluster_ip : char(39)
cluster_name : varchar(30)
cluster_size : smallint(6)
share_key : varchar(10)
queued_for_deletion : tinyint(1)
is_public : tinyint(1)
status : smallint(6)
updated : datetime
created : datetime
creator_id : int(11)

skylab.**skylab_tool**
id : int(11)
display_name : varchar(50)
executable_name : varchar(50)
simple_name : varchar(50)
view_name : varchar(50)
description : longtext
created : datetime
toolset_id : int(11)

skylab.**skylab_toolset**
id : int(11)
display_name : varchar(50)
p2ctool_name : varchar(50)
simple_name : varchar(50)
package_name : varchar(50)
description : longtext
source_url : varchar(200)
created : datetime

skylab.**skylab_skylabfile**
id : int(11)
type : smallint(5) unsigned
upload_path : varchar(200)
file : varchar(100)
filename : varchar(200)
render_with_jsmol : tinyint(1)
task_id : int(11)

skylab.**skylab_toolactivation**
id : int(11)
status : smallint(6)
mpi_cluster_id : int(11)
toolset_id : int(11)

skylab.**skylab_task**
id : int(11)
priority : smallint(5) unsigned
task_data : longtext
status_msg : longtext
status_code : smallint(6)
updated : datetime
created : datetime
mpi_cluster_id : int(11)
tool_id : int(11)
user_id : int(11)

skylab.**skylab_tasklog**
id : int(11)
status_code : smallint(5) unsigned
status_msg : longtext
timestamp : datetime
task_id : int(11)

skylab.**skylab_mpicluster_allowed_users**
id : int(11)
mpicluster_id : int(11)
user_id : int(11)

skylab.**auth_user**
id : int(11)
password : varchar(128)
last_login : datetime
is_superuser : tinyint(1)
username : varchar(150)
first_name : varchar(30)
last_name : varchar(30)
email : varchar(254)
is_staff : tinyint(1)
is_active : tinyint(1)
date_joined : datetime

skylab.**django_site**
id : int(11)
domain : varchar(100)
name : varchar(50)

skylab.**socialaccount_socialtoken**
id : int(11)
token : longtext
token_secret : longtext
expires_at : datetime
account_id : int(11)
app_id : int(11)

skylab.**socialaccount_socialapp**
id : int(11)
provider : varchar(30)
name : varchar(40)
client_id : varchar(191)
secret : varchar(191)
key : varchar(191)

skylab.**socialaccount_socialapp_sites**
id : int(11)
socialapp_id : int(11)
site_id : int(11)

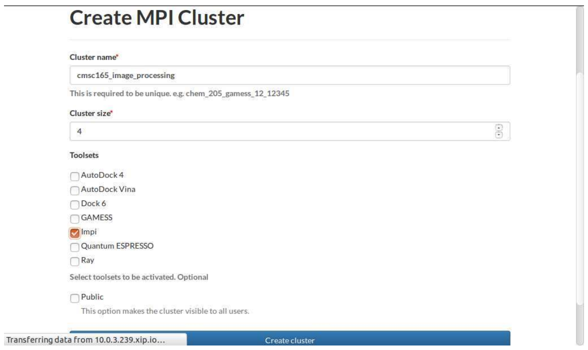
skylab.**socialaccount_socialaccount**
id : int(11)
provider : varchar(30)
uid : varchar(191)
last_login : datetime
date_joined : datetime
extra_data : longtext
user_id : int(11)

Figure 3: MariaDB database schema generated from Skylab's data models (see models.py).

```
        (task_remote_subdirs=
        ['input', 'output'])
    self.handle_input_files()
    self.run_commands()
    self.handle_output_files()
```

2.5 Tasks

The system creates a task object for each task input by the user. A signal will then be sent and it is then received by the corresponding *MPIThread* which queues the task for execution. When a task is executed, it calls the assigned executable class with the given parameters. On connection error, the task waits using exponential



Create MPI Cluster

Cluster name*

cmsc165_image_processing

This is required to be unique. e.g. chem_205_gamess_12_12345

Cluster size*

4

Toolsets

- [ ] AutoDock 4
- [ ] AutoDock Vina
- [ ] Dock 6
- [ ] GAMESS
- [✓] Impi
- [ ] Quantum ESPRESSO
- [ ] Ray

Select toolsets to be activated. Optional

- [ ] Public

This option makes the cluster visible to all users.

Transferring data from 10.0.3.239.xip.io...

Figure 4: MPI cluster creation. A cluster named *cmsc165_image_processing* is being created with the *Impi* tool checked to be activated.

backoff before retrying. If the server crashes while running task execution, the task is just restarted. Table *skylab_task* in Figure 3 stores data for the tasks.

Default task execution is as follows:

(1) Needed remote and local directories for execution are cleared or created.
(2) Input files are uploaded to the cluster.
(3) List of commands given are executed.
(4) Output files are retrieved from the cluster.
(5) Remote task folder is deleted.
(6) Output files are served to the user.

2.6 Output Files

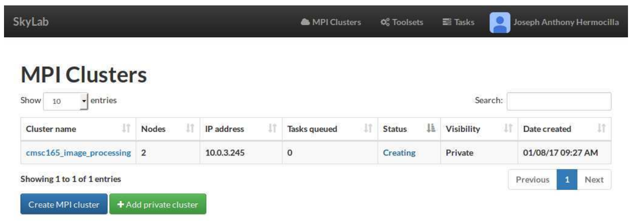
Tools generate output files that must be returned to the user. Since SkyLab can support many tools, it handles output files differently for each tool. Toolsets must implement the *handle_output_files* method of the *P2CToolGeneric* class. This method's implementation typically downloads output files from the cluster running the tool to SkyLab's *output* directory using sftp. If a tool has multiple output files, these are zipped first before downloading. The user will then be given links to download the output files. For some modules, like GAMESS, the ouput can be directly rendered on the browser (see Figure 11).

3 RESULTS

Figures 4 to 11 show some of the screenshots that realize the use case (Figure 2) for SkyLab. The screenshots were obtained from a SkyLab instance that is hosted on P2C itself.

4 EVALUATION

The system has been evaluated by 56 respondents by answering a survey based on Questionnaire for User Interface Satisfaction (QUIS) [4]. The respondents are students who are unfamiliar with both HPC tools and the concept of MPI systems. They are asked to test features of SkyLab by following a set of instructions and using input files provided. On the average, the users rated their overall experience with SkyLab to 6.9/10. The users listed the simplicity of the user interface to be the most positive aspect of the system



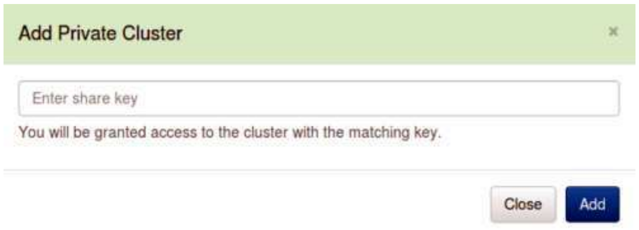
MPI Clusters

Show 10 entries Search:

Cluster name	Nodes	IP address	Tasks queued	Status	Visibility	Date created
cmsc165_image_processing	2	10.0.3.245	0	Creating	Private	01/08/17 09:27 AM

Showing 1 to 1 of 1 entries

Figure 5: MPI cluster table. Currently deployed MPI clusters are shown with some of their properties such as IP address and creation timestamp.

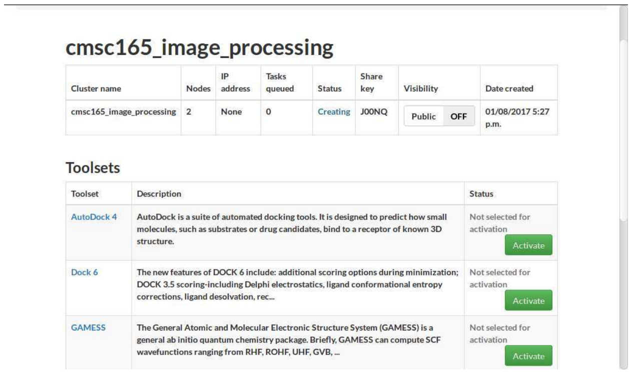


Add Private Cluster

Enter share key

You will be granted access to the cluster with the matching key.

Figure 6: Set cluster visibility. A user can make a private cluster visible by entering a valid shared key.



cmsc165_image_processing

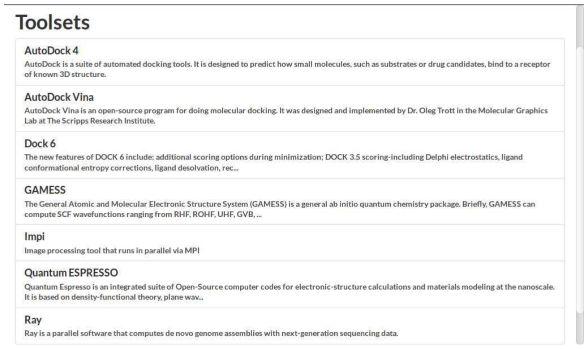
Cluster name	Nodes	IP address	Tasks queued	Status	Share key	Visibility	Date created
cmsc165_image_processing	2	None	0	Creating	J0ONQ	Public OFF	01/08/2017 5:27 p.m.

Toolsets

Toolset	Description	Status
AutoDock 4	AutoDock is a suite of automated docking tools. It is designed to predict how small molecules, such as substrates or drug candidates, bind to a receptor of known 3D structure.	Not selected for activation Activate
Dock 6	The new features of DOCK 6 include: additional scoring options during minimization; DOCK 3.5 scoring-including Delphi electrostatics, ligand conformational entropy corrections, ligand desolvation, rec...	Not selected for activation Activate
GAMESS	The General Atomic and Molecular Electronic Structure System (GAMESS) is a general ab initio quantum chemistry package. Briefly, GAMESS can compute SCF wavefunctions ranging from RHF, ROHF, UHF, GVB, ...	Not selected for activation Activate

Figure 7: MPI cluster details view. The detailed view of the *cmsc165_image_processing* MPI cluster with a list of tools that can be activated.

while the slow speed of task processing is said to be the most negative. Majority of the tools supported by SkyLab have inherently long processing time which is not known to the respondents. The system does not focus on optimizing the said tools to achieve better performance but rather it focuses on simplifying the user's task submission process.



Toolsets

AutoDock 4
AutoDock is a suite of automated docking tools. It is designed to predict how small molecules, such as substrates or drug candidates, bind to a receptor of known 3D structure.

AutoDock Vina
AutoDock Vina is an open-source program for doing molecular docking. It was designed and implemented by Dr. Oleg Trott in the Molecular Graphics Lab at The Scripps Research Institute.

Dock 6
The new features of DOCK 6 include: additional scoring options during minimization; DOCK 3.5 scoring-including Delphi electrostatics, ligand conformational entropy corrections, ligand desolvation, rec...

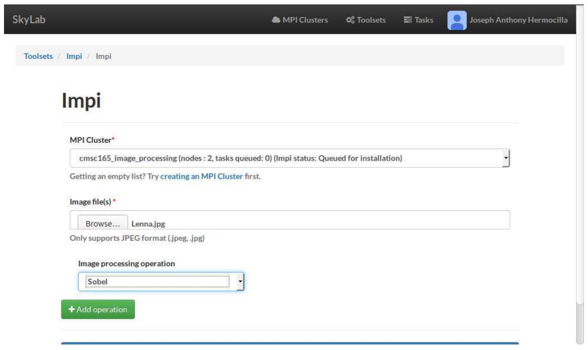
GAMESS
The General Atomic and Molecular Electronic Structure System (GAMESS) is a general ab initio quantum chemistry package. Briefly, GAMESS can compute SCF wavefunctions ranging from RHF, ROHF, UHF, GVB, ...

Impi
Image processing tool that runs in parallel via MPI

Quantum ESPRESSO
Quantum Espresso is an integrated suite of Open-Source computer codes for electronic-structure calculations and materials modeling at the nanoscale. It is based on density-functional theory, plane wav...

Ray
Ray is a parallel software that computes de novo genome assemblies with next-generation sequencing data.

Figure 8: Toolset list view. The user can select from a list which tool to use.



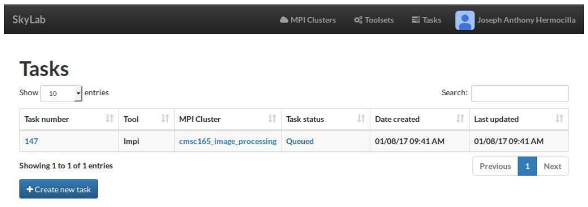
Impi

MPI Cluster*
cmsc165_image_processing (nodes : 2, tasks queued: 0) (Impi status: Queued for installation)
Getting an empty list? Try creating an MPI Cluster first.

Image file(s) *
Browse... Lenna.jpg
Only supports JPEG format (.jpeg, .jpg)

Image processing operation
Sobel

Figure 9: Impi task creation form. The user will run an Impi task (sobel edge detection) with Lenna.jpg as input.



Tasks

Show 10 entries Search:

Task number	Tool	MPI Cluster	Task status	Date created	Last updated
147	Impi	cmsc165_image_processing	Queued	01/08/17 09:41 AM	01/08/17 09:41 AM

Showing 1 to 1 of 1 entries

Figure 10: Task table view. Impi task on the cmsc165_image_processing cluster is currently queued for execution.

5 RELATED WORK

Traditional usage of HPC applications requires users to learn job submission systems like Torque[4] or Slurm [5] that are used via the command line. Scientific workflow systems make it easier for users to use their tools without mastering the command line.

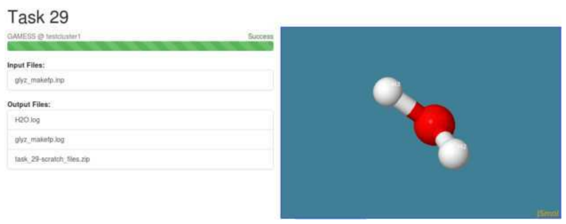
The main motivation for developing SkyLab is Yabi[10]. It provides a web interface with support for workflow environments with focus on introducing HPC applications to non-technical audience. Users can create and reuse workflows, and manage large amounts of data while system administrators can configure tools via the web interface as well. It is currently in use by multiple institutions, and is maintained as an open-source project.

Another related project is Web Interface for mpiBLAST (WImpiBLAST). It supports mpiBLAST, a parallel implementation of Basic Local Alignment Search Tool (BLAST). BLAST is a software used for sequence homology similarity search in large databases of gene sequences. mpiBLAST can utilize HPC clusters to achieve faster computing speeds but it requires knowledge in using MPI commands to benefit from its advantages. WImpiBLAST addresses this problem by providing the user a web interface to simplify the steps to use mpiBLAST[15].

Pegasus [5] is another scientific workflow system that allows users to specify workflows at an abstract level. This tool is particularly useful in grid computing with a lot of distributed resources.

The Kepler [12] system supports low-level workflows for grid engineers at the same time analytical knowledge discovery workflow for scientists.

Taverna [9] is currently gaining popularity as a modern domain-independent workflow management system. It is written in Java and has moved to the Apache Software Foundation incubator project[6].



Task 29
GAMESS @ testcluster1 Success
Input Files:
glyz_makefp.inp
Output Files:
H2O.log
glyz_makefp.log
task_29-scratch_files.zip

Figure 11: Task detail view. JSmol [7] renders the compatible output files from a GAMESS task.

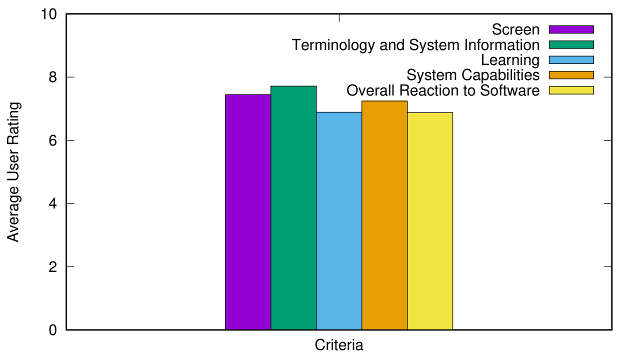


Figure 12: Results of QUIS for SkyLab.

[4]http://www.adaptivecomputing.com/products/open-source/torque/
[5]https://slurm.schedmd.com/

[6]https://taverna.incubator.apache.org/

It allows scientists to construct complex analysis given limited background in computing.

6 CONCLUSION AND FUTURE WORK

SkyLab allowed users to manage MPI clusters and submit tasks without the need for technical expertise in command line and scripting. This makes the advantages of HPC available to non-technical users. This is achieved by parsing form inputs to generate commands for task execution. Task files can be download from the server and output files are displayed with the help of JSmol. The system is also configured to install tool sets found in the modules folder making it possible to accommodate additional tools. Based on the user acceptance test conducted, the users found the system to be acceptable in terms of the criteria provided, in general.

SkyLab achieved its main objectives but its features can still be improved and additional features can be introduced. Improved input parameter checking and error handling will make the system more robust. There are still use cases of tools that are yet to be supported. Input file generation can make the process more interactive and more customizable. Workflow design support will enable users to run complex tasks. Support for custom MPI programs will make it easier for developers to utilize the system as a test environment. Task scheduling and resource management algorithms can be used to efficiently handle resource-intensive or time consuming tasks. For example, a cluster can borrow resources from idle clusters. These recommendations will provide the users a better experience in using the system for academic and research purposes.

ACKNOWLEDGMENT

This work is supported by the Philippine Department of Science and Technology Accelerated Science and Technology Human Resource Development Program. We also would like to thank our collaborators who provided the use cases for the tools: Dr. Abraham Padama (Material Science), Dr. Marlon Manalo (Chemistry), and Prof. Arian Jacildo (Bioinformatics). Finally, we acknowledge the reviewers for their time and valuable comments.

REFERENCES

[1] Sanjay P. Ahuja and Sindhu Mani. 2012. The State of High Performance Computing in the Cloud. *Journal of Emerging Trends in Computing and Information Sciences* 3, 2 (Feb. 2012), 263–266. http://www.chinacloud.cn/upload/2012-03/12031713036456.pdf

[2] Sbastien Boisvert, Frdric Raymond, ftlnie Godzaridis, Franois Laviolette, and Jacques Corbeil. 2012. Ray Meta: scalable de novo metagenome assembly and profiling. *Genome Biology* 13, 12 (2012), R122. DOI:http://dx.doi.org/10.1186/gb-2012-13-12-r122

[3] Ivona Brandic, Ioan Raicu, Satish Narayana Srirama, Oleg Batrashev, Pelle Jakovits, and Eero Vainikko. 2011. Scalability of parallel scientific applications on the cloud. *Scientific Programming* 19, 2/3 (2011), 91 – 105. http://search.ebscohost.com/login.aspx?direct=truedb=a9h&AN=66692024site=ehost-live

[4] J. P. Chin, V. A. Diehl, and K. L. Norman. 1988. Development of an instrument measuring user satisfaction of the human-computer interface. In *Proceedings of SIGCHI '88 ACM/SIGCHI.* New York, 213–218.

[5] Ewa Deelman, Gurmeet Singh, Mei-Hui Su, James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Karan Vahi, G Bruce Berriman, John Good, and others. 2005. Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming* 13, 3 (2005), 219–237.

[6] Paolo Giannozzi, Stefano Baroni, Nicola Bonini, Matteo Calandra, Roberto Car, Carlo Cavazzoni, Davide Ceresoli, Guido L Chiarotti, Matteo Cococcioni, Ismaila Dabo, Andrea Dal Corso, Stefano de Gironcoli, Stefano Fabris, Guido Fratesi, Ralph Gebauer, Uwe Gerstmann, Christos Gougoussis, Anton Kokalj, Michele Lazzeri, Layla Martin-Samos, Nicola Marzari, Francesco Mauri, Riccardo Mazzarello, Stefano Paolini, Alfredo Pasquarello, Lorenzo Paulatto, Carlo Sbraccia, Sandro Scandolo, Gabriele Sclauzero, Ari P Seitsonen, Alexander Smogunov, Paolo Umari, and Renata M Wentzcovitch. 2009. QUANTUM ESPRESSO: a modular and open-source software project for quantum simulations of materials. *Journal of Physics: Condensed Matter* 21, 39 (2009), 395502 (19pp). http://www.quantum-espresso.org

[7] Robert M. Hanson, Jaime Prilusky, Zhou Renjian, Takanori Nakane, and Joel L. Sussman. 2013. JSmol and the Next-Generation Web-Based Representation of 3D Molecular Structure as Applied to Proteopedia. *Israel Journal of Chemistry* 53, 3-4 (2013), 207–216. DOI:http://dx.doi.org/10.1002/ijch.201300024

[8] Joseph Anthony C. Hermocilla. 2014. P2C: Towards Scientific Computing on Private Clouds. In *Proceedings of the National Conference on Information Technology Education (NCITE 2014).* 162–167.

[9] Duncan Hull, Katy Wolstencroft, Robert Stevens, Carole Goble, Mathew R Pocock, Peter Li, and Tom Oinn. 2006. Taverna: a tool for building and running workflows of services. *Nucleic acids research* 34, suppl 2 (2006), W729–W732.

[10] Adam A. Hunter, Andrew B. Macgregor, Tamas O. Szabo, Crispin A. Wellington, and Matthew I. Bellgard. 2012. Yabi: An online research environment for grid, high performance and cloud computing. *Source Code for Biology and Medicine* 7, 1 (2012), 1 – 10. http://search.ebscohost.com/login.aspx?direct=true&db=a9h&AN=74110216&site=ehost-live

[11] P. T. Lang, S. R. Brozell, S. Mukherjee, E. T. Pettersen, E. C. Meng, V. Thomas, R. C. Rizzo, D. A. Case, T. L. James, and I. D. Kuntz. 2009. DOCK 6: Combining Techniques to Model RNA-Small Molecule Complexes. *RNA* 15 (2009), 1219–1230.

[12] Bertram Ludäscher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward A Lee, Jing Tao, and Yang Zhao. 2006. Scientific workflow management and the Kepler system. *Concurrency and Computation: Practice and Experience* 18, 10 (2006), 1039–1065.

[13] G. M. Morris, R. Huey, W. Lindstrom, M. F. Sanner, R. K. Belew, D. S. Goodsell, and A. J. Olson. 2009. Autodock4 and AutoDockTools4: automated docking with selective receptor flexiblity. *J. Computational Chemistry* 2009 (2009), 2785–91.

[14] Michael W. Schmidt, Kim K. Baldridge, Jerry A. Boatz, Steven T. Elbert, Mark S. Gordon, Jan H. Jensen, Shiro Koseki, Nikita Matsunaga, Kiet A. Nguyen, Shujun Su, Theresa L. Windus, Michel Dupuis, and John A. Montgomery. 1993. General atomic and molecular electronic structure system. *Journal of Computational Chemistry* 14, 11 (1993), 1347–1363. DOI:http://dx.doi.org/10.1002/jcc.540141112

[15] Parichit Sharma and Shrikant S. Mantri. 2014. WImpiBLAST: Web Interface for mpiBLAST to Help Biologists Perform Large-Scale Annotation Using High Performance Computing. *PLoS ONE* 9, 6 (2014), 1 – 13. http://search.ebscohost.com/login.aspx?direct=true&db=a9h&AN=96861207&site=ehost-live

[16] Axel S. Trajano and Joseph Anthony C. Hermocilla. 2010. *Implementation of selected image processing routines for single and distributed processors.* Technical Report. Undergraduate Special Problem, University of the Philippines, Los Banos.

[17] Oleg Trott and Arthur J. Olson. 2010. AutoDock Vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of Computational Chemistry* 31, 2 (2010), 455–461. DOI:http://dx.doi.org/10.1002/jcc.21334