SkyLab : A Web Application for vCluster

Vincent Paul L. Carpio
Institute of Computer Science
College of Arts and Sciences
University of the Philippines Los Banos
vplcarpio@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

Katrina Joy M. Abriol-Santos
Institute of Computer Science
College of Arts and Sciences
University of the Philippines Los Banos
kat@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 and more convenient. One of its features is vCluster, a tool that can deploy MPI clusters on demand[8]. We have developed SkyLab, a web interface for vCluster. It aims to simplify the process of running MPI tasks from the perspective of the end-user. Here, we describe the system design and features of SkyLab.

CCS CONCEPTS

•**Computer systems organization** → **Embedded systems;** *Redundancy;* Robotics; •**Networks** → Network reliability;

KEYWORDS

ACM proceedings, LaTeX, text tagging

ACM Reference format:
Vincent Paul L. Carpio, Joseph Anthony C. Hermocilla, and Katrina Joy M. Abriol-Santos. 2016. SkyLab : A Web Application for vCluster. In *Proceedings of ACM Conference, Washington, DC, USA, July 2017 (Conference'17),* 6 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

1.1 Background

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 handling computer 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 [4].

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. Fortunately, with the advancement in cloud computing, these tools can be deployed in the cloud without worrying about the costs of hardware purchase and maintenance [1]. To determine the cost of hosting scientific computing in cloud, the Scientific Computing Cloud (SciCloud) project is conducted in the University of Tartu. Results have shown that transmission delays are the major concern in pursuing HPC applications in the cloud[4]. To address this problem, a study is conducted by Hermocilla[8] in Peak-Two Cloud (P2C). One of the features introduced by P2C is vCluster. vCluster is a tool that enables a user to deploy a working (MPI) cluster on demand and to terminate it after execution. It uses a master-slave design implementation. However, vCluster is a command-line based application which has limited capacities for data manipulation, analysis, and presentation. Creating a web application that would host vCluster with additional features like graphical representation to visualize trends, and a Graphical User Interface(GUI) to offer convenient access. This paper will present SkyLab, a web application that aims to serve and to improve vCluster functionalities.

1.2 Significance of the Study

This study will help us visualize how HPC applications can be hosted in the cloud and understand why is this technology timely relevant. Even with capable hardware, computations might take hours or even days which makes it a hassle for users. Thus, there is a need for a more convenient way of access for HPC applications. Furthermore, this would encourage students to explore and research on HPC applications. HPC applications usually have a plain numerical

Conference'17, Washington, DC, USA

DOI: 10.1145/nnnnnnn.nnnnnnn

output. A means of presenting data graphically would make the interpretation of data more efficient.

The study would contribute to research on hosting HPC applications in the cloud. Problems encountered and solutions offered throughout the study will give insights to future researchers. Furthermore, the output application of the study can be used by the academe.

1.3 Scope and Limitations

The study focuses on vCluster and is not mainly concerned on P2C as a whole. The study initially bases on the current implementation of vCluster and eventually improve it and offer additional features[8].

1.4 Objectives

This study aims to develop SkyLab, a web application that would function as a front-end for vCluster. Specifically, it should be able to:

(1) allow users to select tools and execute them via web interface;
(2) create an extensible platform that would accommodate additional tools; and
(3) display output data of tools.

2 REVIEW OF RELATED LITERATURE

2.1 Performance Concerns for HPC Applications in the Cloud

Shajulin [2] has enumerated five performance concerns for HPC applications in the cloud. First concern is resource provisioning issues which includes memory management, energy management, response time, and resource outages. Hardware virtualization for multiple guest operating systems running concurrently on a host computer creates a concern for huge overheads, delays, or memory leakages caused by the process of allocating memory, remapping it to virtual machines, and cleaning it up. The scenario of multiple guest OS can also cause an OS to fail to make hardware use energy-efficient because the presence of multiple hardware virtualizations prevent the main OS from having full control over actual hardware. Configuration failures, update failures, or even unknown issues cause resource outages. Second, most programming models used in the cloud have inconsistencies on error recovery and storage handling by design and by execution. If these inconsistencies are not addressed, the system components might fail because they are not well-integrated. Third, enforcing terms of Service Level Agreement (SLA) is bound to performance issues due to loss of control over resource provisioning or deploying cloud services. Fourth, there is a lack of standard conventions in clouds. This means that there is no guarantee that the current configuration of the an application will be compatible with other providers. The user might need to modify the program for a specific provider. Fifth, security measures challenge performance since increasing security protocols decreases response time.

In a study by Jackson et al.[12] conducted on Amazon EC2, benchmarks are used to imitate the workload of a typical HPC application. Results of the study show that the percentage of the communication time of the application is strongly correlated to its overall performance. Communication time and frequency of global communications are inversely proportional to the application's performance. Lastly, performance variability is introduced by the added layers of abstraction caused by hardware virtualization.

Another study conducted by Juve et al.[13], analyzed running scientific workflows in Amazon EC2. It has been found that getting resources is the primary cost factor to consider in executing workflow tasks while storage has relatively cheaper costs. Storing data in the cloud rather that transferring it to each workflow effectively reduces the high cost of data transfer. These results show that the cloud is a good alternative for running scientific workflow applications but for cloud providers to be able compete with existing physical clusters in terms of performance of HPC applications would need high-speed networks and parallel file systems[20]. While this may mean that cloud computing is yet to support large-scale HPC applications, it can still be considered as an alternative for deploying simple HPC tasks on demand[11]. Despite the performance trade-off, EC2 offers ease-of-use and cheaper costs which are marginal factors in consideration for academic purposes[9].

2.2 Related Work

Ganglia is a system designed to monitor high performance computing systems. It uses a hierarchical model in managing the system of clusters. It uses optimized data structures and communication algorithms to achieve scalability with high concurrency. It is claimed to be used by over 500 clusters around the world. This implies that the system is tested and trusted to be used for real-world applications[15].

One of the main inspirations for developing SkyLab is the Yabi system. 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[10].

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[18].

3 METHODOLOGY

3.1 Materials

Programming Language: Python
Web Framework: Django
DBMS: MariaDB
Cloud Infrastructure: Peak-Two Cloud

3.2 System Architecture

SkyLab functions as a web front-end for vCluster which is built on top of Peak-Two Cloud.

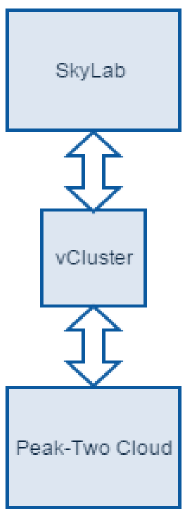

Figure 1: System Architecture of SkyLab

3.3 Requirements Specification

3.3.1 *Functional Requirements.*

- Users can browse and select tools with an interface. It includes, but not limited to:
  AutoDock
  It is a software used to simulate protein-ligand docking[16].
  AutoDock Vina
  It is a software similar to AutoDock 4 but on the average, it provides faster and more accurate computations[19].
  DOCK
  It is used to predict the small molecule-target interactions[14].
  Quantum ESPRESSO
  It is an integrated software suite of tools for ab-initio molecular dynamics (MD) simulations and electronic structure calculations[6].
  GAMESS
  It is used for ab initio molecular quantum chemistry. [17]
  Ray
  It is uses parallel genome assemblies for parallel DNA sequencing [3].
- Users can upload input data to the server.
- Users can view the results of the used tools.
- The system is integrated with vCluster functionalities.

3.3.2 *Non-functional Requirements.*

- Users must be authenticated to be able to use the system.
- The system must be easy to use and user friendly.
- The system must be highly maintainable for future development.

3.4 Evaluation of Methodology

The use cases given ensure that the user will be able to select tools and execute them with the system. Also, the output files of user tasks will be served by the system. The system code is required maintainable for future inclusion of other tools. The methodology provided is sufficient to achieve the given objectives of the study. The created software will be subjected to user acceptance test for further evaluation.

4 RESULTS AND DISCUSSION

4.1 System Design

4.1.1 MPI clusters. The system spawns a thread (MPIThread) 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.

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.

4.1.2 Tool sets. The system searches for Python packages inside the assigned modules folder and install it on server start. The tool sets will then be available for use with the system. The package must have a Python module named *install.py* which contains function calls for integrating the package with the system. The package must also contain the corresponding views and executable classes for each sub-tool.

4.1.3 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 exponentially before retrying. If the server crashes while running task execution, the task is just restarted.

Default task execution flow via executable class:

(1) Needed remote and local directories for execution are cleared or created.

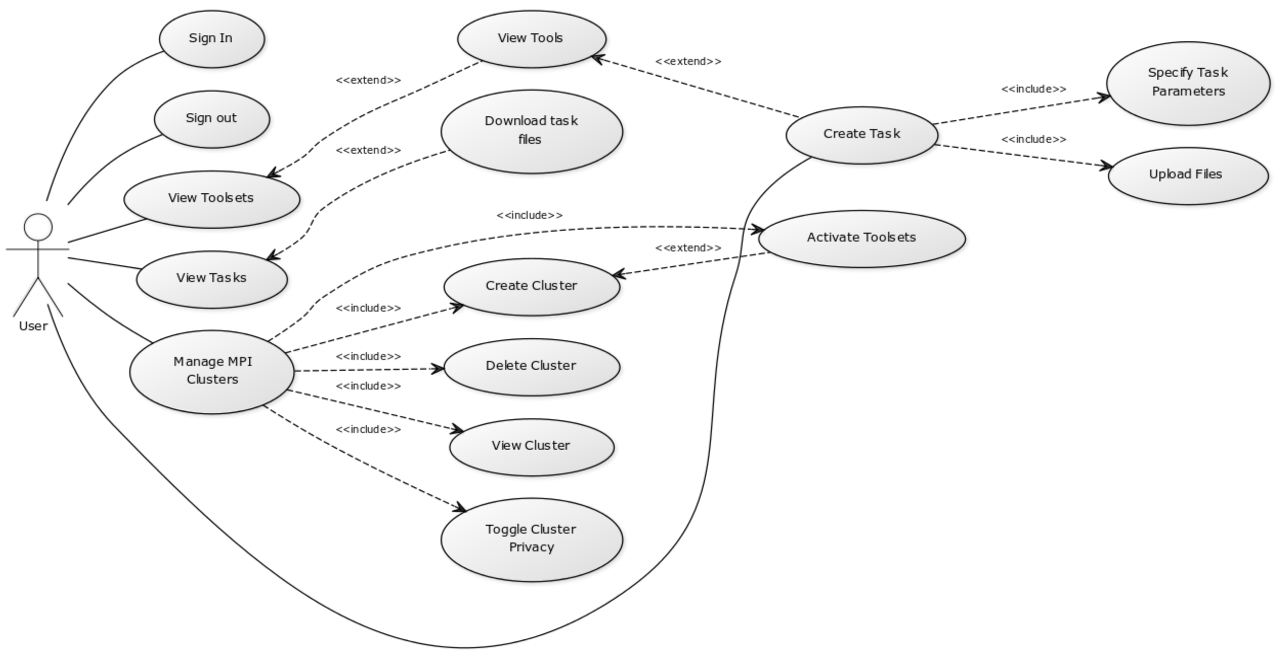

Figure 2: Use Case Diagram of SkyLab

(2) Input files are uploaded to cluster.
(3) List of commands given are executed.
(4) Output files are sent back to the server.
(5) Remote task folder is deleted.
(6) Output files are served by the server.

4.2 System Features

The system's interface offers different functionalities that simplifies MPI cluster and task management.

- The user is authenticated by logging in with his @up.edu.ph Google account.
- The user can create an MPI cluster with optional tool activations.

Figure 3: MPI creation form

- The user can monitor visible public and private MPI clusters.

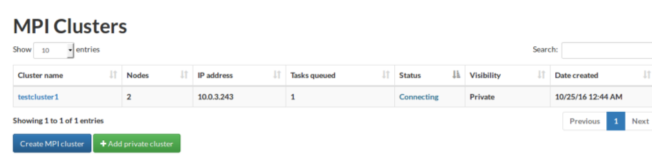

Figure 4: MPI cluster table

- The user can make a private cluster visible by entering a valid share key.

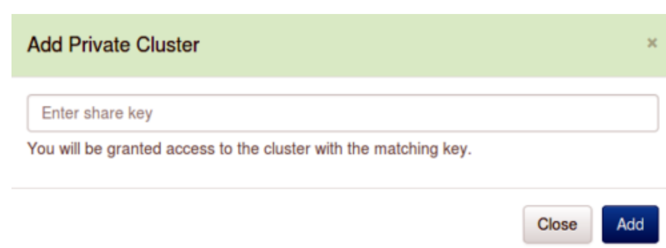

Figure 5: Add private cluster form

- The user can view details about a MPI cluster. If the user is the cluster's owner he has the option to delete the cluster.

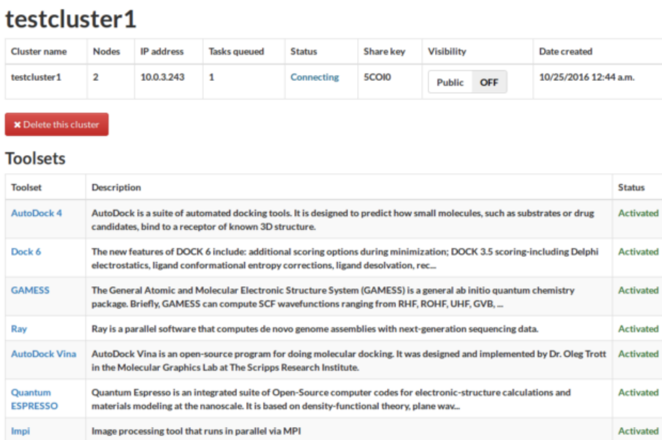

Figure 6: MPI detail view

- The user can select from a list which tool does he want to use.

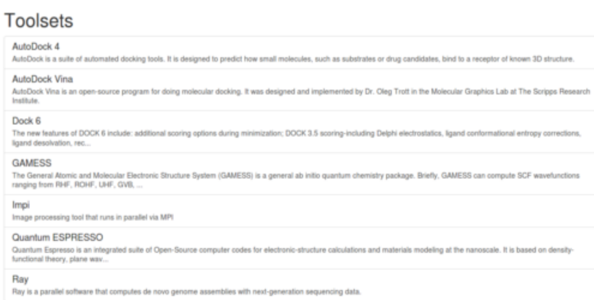

Figure 7: Tool set list view

- The user can submit a task by filling up a tool's task creation form.

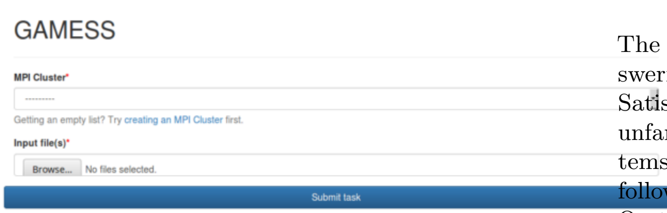

Figure 8: GAMESS task creation form

- The user can monitor created tasks.

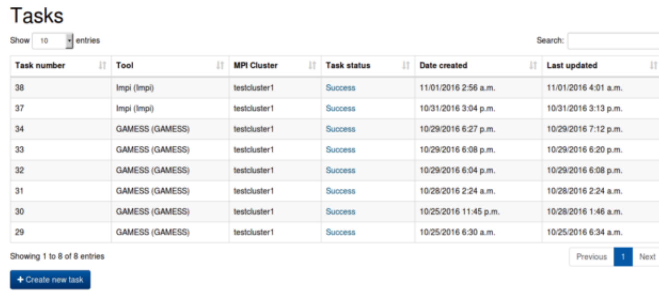

Figure 9: Task table view

- The user can view results of tasks. JSmol renders the compatible output files[7].

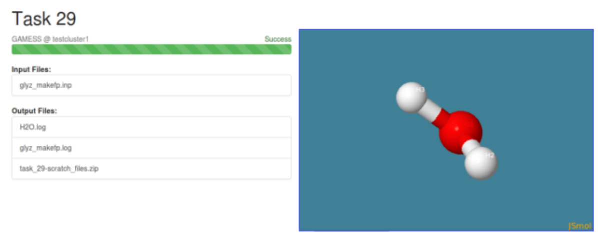

Figure 10: Task detail view

4.3 User Acceptance

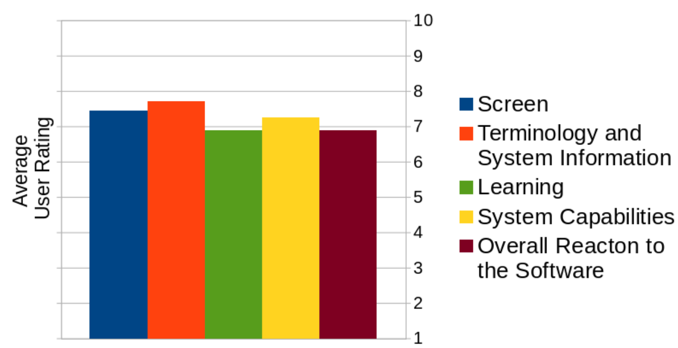

Figure 11: Results of QUIS for SkyLab

The system has been evaluated by 56 respondents by answering a survey based on Questionnaire for User Interface Satisfaction (QUIS) [5]. Respondents are students who are unfamiliar with both HPC tools and the concept of MPI systems. Respondents 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 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.

5 CONCLUSION AND FUTURE WORK

The system created allows users to manage MPI clusters and submit tasks without the need for technical expertise in 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[7]. 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.

The system 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.

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] Shajulin Benedict. 2013. Performance issues and performance analysis tools for HPC cloud applications: a survey. *Computing* 95, 2 (2013), 89 – 108. http://search.ebscohost.com/login.aspx?direct=true&db=a9h&AN=85211339&site=ehost-live

[3] Sbastien Boisvert, Frdric Raymond, lnie 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

[4] 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

[5] 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.

[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] Zach Hill and Marty Humphrey. 2009. A Quantitative Analysis of High Performance Computing with Amazons EC2 Infrastructure: The Death of the Local Cluster?. In *Proceedings of the 10th IEEE/ ACM International Conference on Grid Computing (Grid 2009).)*. http://www.cs.virginia.edu/~humphrey/papers/QuantitativeAnalysisEC2.pdf

[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] Alexandru Iosup, Simon Ostermann, Nezih Yigitbasi, Radu Prodan, Thomas Fahrigner, and Dick Epema. 2010. Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing. (Nov 2010). http://www.st.ewi.tudelft.nl/~iosup/cloud-perf10tpds_in-print.pdf

[12] Keith Jackson, Krishna Muriki, Shane Canon, Shreyas Cholia, John Shalf, Harvey Wasserman, and Nicholas Wright. 2010. Performance Analysis of High Performance Computing Applications on the Amazon Web Services Cloud. (2010). https://www.nersc.gov/assets/NERSC-Staff-Publications/2010/CloudCom.pdf

[13] Gideon Juve, Ewa Deelman, Karan Vahi, Gaurang Mehta, Bruce Berriman, Benjamin P. Berman, and Phil Maechling. 2009. Scientific workflow applications on Amazon EC2. In *2009 5th IEEE International Conference on E-Science Workshops*. IEEE, 59–66. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5408002

[14] 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.

[15] Matthew L. Massie, Brent N. Chun, and David E. Culler. 2004. The ganglia distributed monitoring system: design, implementation, and experience. *Parallel Comput.* 30, 7 (2004), 817 – 840. http://search.ebscohost.com/login.aspx?direct=true&db=a9h&AN=13956548&site=ehost-live

[16] 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.

[17] 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

[18] 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

[19] 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

[20] Edward Walker. 2008. Benchmarking Amazon EC2 for High-Performance Scientific Computing. (Oct 2008). https://www.usenix.org/legacy/publications/login/2008-10/openpdfs/walker.pdf