SkyLab : A Web Application for vCluster

Vincent Paul L. Carpio and Katrina Joy M. Abriol-Santos and Joseph Anthony C. Hermocilla

***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. [1] 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.**

***Index Terms*— cloud computing, high performance computing, web interface**

I. Introduction

A. 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 [2]. 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. 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 [2]. 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 [3]. To address this problem, a study is conducted by Hermocilla 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 [1]. 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.

Presented to the Faculty of the Institute of Computer Science, University of the Philippines Los Baños in partial fulfillment of the requirements for the Degree of Bachelor of Science in Computer Science

B. 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 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.

C. 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 [1].

D. 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.

II. Review of Related Literature

A. Performance Concerns for HPC Applications in the Cloud

Shajulin [4] 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 the memory. The scenario of multiple guest OS can also cause an OS to fail to make hardware use energy-efficient due to lack of control over hardware resources. Configuration failures, update failures, or even unknown issues cause resource outages. Second, most programming models used in the cloud have inconsistencies between the fault recovery mechanisms in execution and storage processes. Inconsistencies can result to poor performance. 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, lack of best practices or standards has hindered the utilization of the cloud. If there is no standard interoperable cloud interfaces, users might need to modify their application accordingly with the cloud provider. Fifth, security measures challenges performance since increasing security protocols decreases response time [4].

In a study by Jackson et al. [5], performance of a set of benchmarks designed to represent a typical HPC workload run on Amazon EC2 have been quantitatively examined. Results of the study show that the percentage of the communication time of the application is strongly correlated to its overall performance in EC2. The more communication, the worse it performs. In addition, applications with significant global communication perform relatively worse than those with less. Lastly, variability can be significant in EC2. Variability is introduced by multiple virtual machines, by the network and by underlying non-virtualized hardware [5].

Another study conducted by [6], analyzed running scientific workflows in EC2. It has been found that the primary cost is in getting resources to execute workflow tasks, and storage costs are relatively cheaper. 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 systems in terms of performance of HPC applications would need high-speed networks and parallel file systems [6] [7]. While current cloud computing services are insufficient for large-scale scientific computing, it may still be a viable option for the scientists who need resources instantly and temporarily [8]. Despite the performance tradeoff, EC2 offers ease-of-use and cheaper costs which are marginal factors in consideration for academic purposes [9].

B. Related Work

The Yabi system [10] targets the non-technical audience with an intuitive drag-and-drop scalable web-based workflow environment that allows HPC to be appreciated a broader audience. Yabi also enables other workflow systems that have limited access to multiple existing HPC to utilize its data and access models.

Another related project is WImpiBLAST [11], an open-source web interface for parallel BLAST searches. BLAST is the most extensively used gene sequence analysis program for sequence homology similarity search in large databases of sequences. mpiBLAST is an open-source parallelization of BLAST that can speedup large-scale annotation by using supercomputers and HPC clusters. WImpiBLAST is created to help researchers who lack expertise in using Message Passing Interface (MPI) benefit from its advantages.

III. Methodology

A. Materials

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

B. System Architecture

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

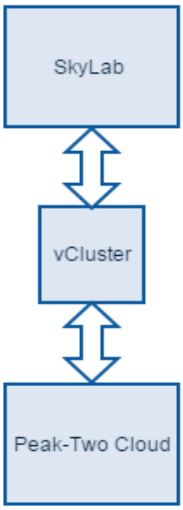

Fig. 1: System Architecture of SkyLab

C. Requirements Specification

1) Functional Requirements:

- Users can browse and select tools with an interface. It includes, but not limited to:

  AutoDock
  : It is a suite of automated docking tools designed to predict how small molecules bind to a receptor of known 3D structure. [12]

  AutoDock Vina
  : It achieves significant improvements in the average accuracy of the binding mode predictions, while also being up to two orders of magnitude faster than AutoDock 4. [13]

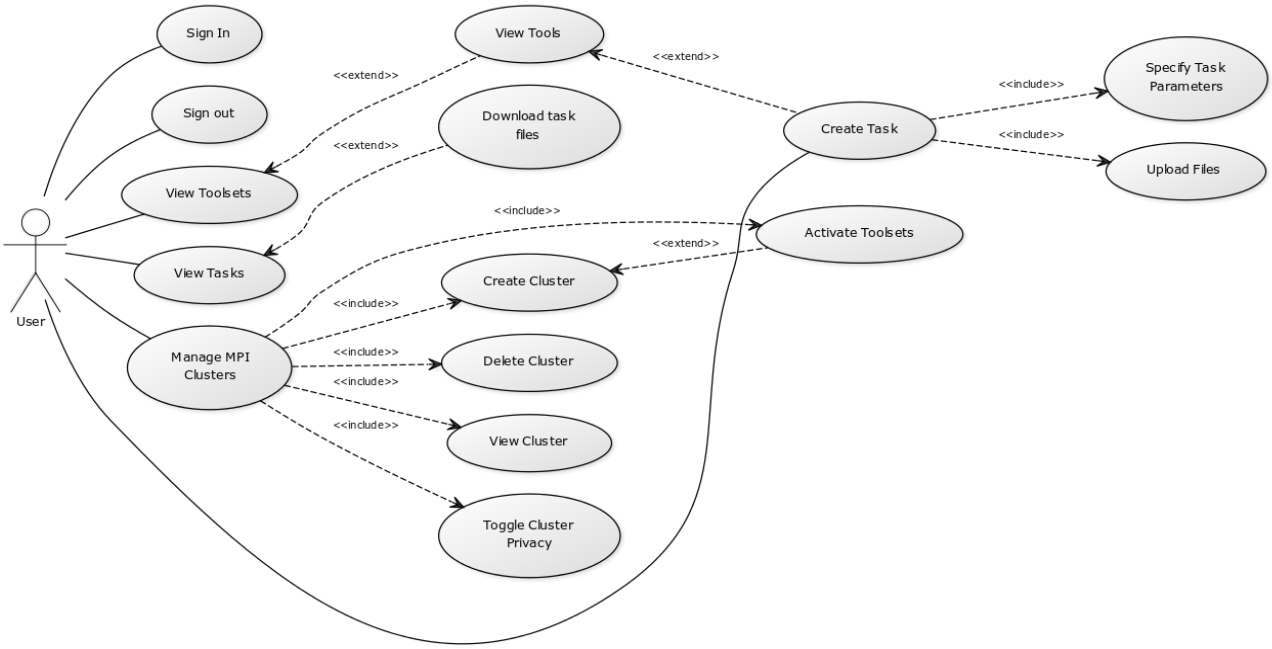

Fig. 2: Use Case Diagram of SkyLab

DOCK
: It is used to predict the small molecule-target interactions. [14]

Quantum Espresso
: It is an integrated suite of Open-Source computer codes for electronic-structure calculations and materials modeling at the nanoscale. [15]

GAMESS
: GAMESS is a program for ab initio molecular quantum chemistry. [16]

Ray
: It is uses parallel genome assemblies for parallel DNA sequencing [17]

- Users can upload input data to the server.
- Users can view the results of the used tools.
- The system is integrated with vCluster functionalities.

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.

D. 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.

IV. Results and Discussion

A. System Design

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 queueing 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.

2) Toolsets: The system searches for Python packages inside the assigned modules folder and install it on server start. The toolsets 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 subtool.

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.
2) Input files are uploaded to cluster.
3) List of commands given are executed.

4) Output files are sent back to the server.
5) Deletes remote task folder
6) Output files are served by the server.

B. 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.

Fig. 3: MPI creation form

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

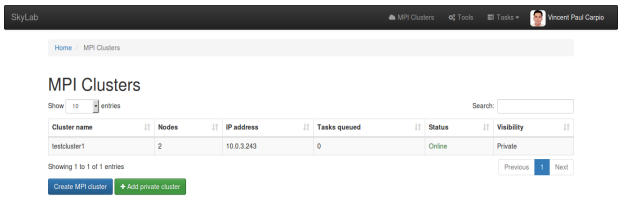

Fig. 4: MPI list view

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

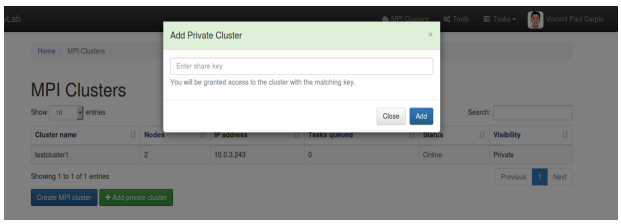

Fig. 5: Enter share key form

- 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.

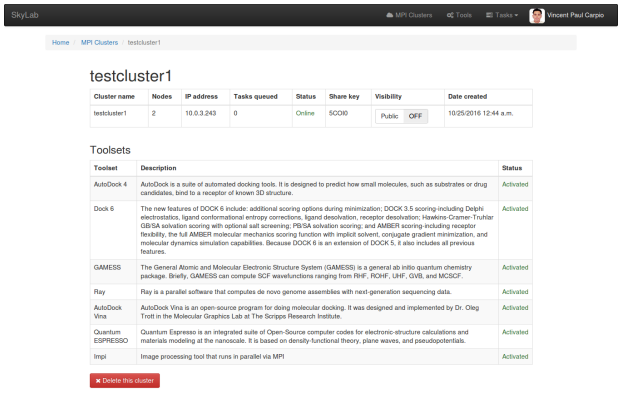

Fig. 6: MPI detail view

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

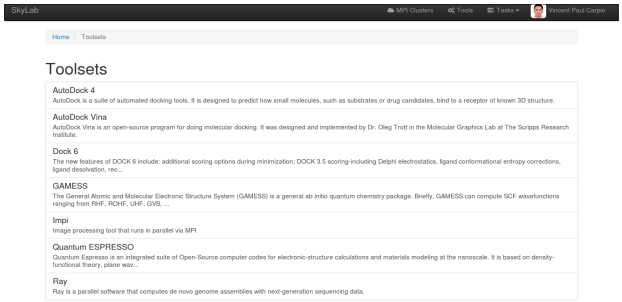

Fig. 7: GAMESS task creation form

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

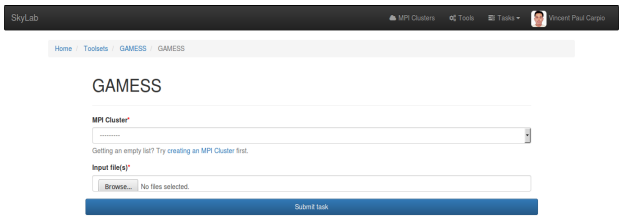

Fig. 8: GAMESS task creation form

- The user can monitor created tasks.

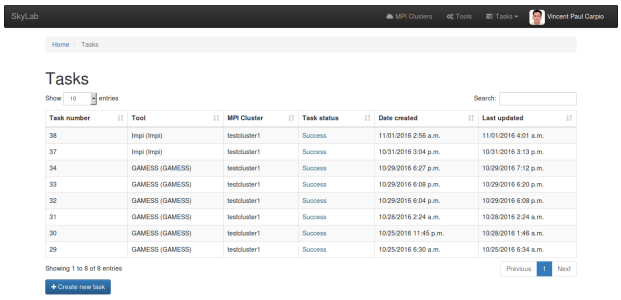

Fig. 9: Task list view

- The user can view results of tasks. JSmol renders the compatible output files [18].

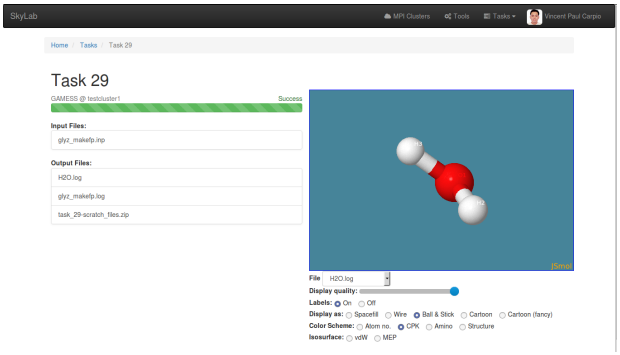

Fig. 10: Task detail view

C. User Acceptance

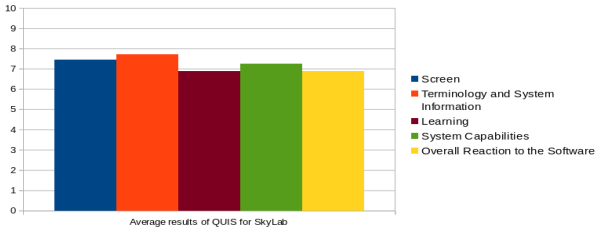

Fig. 11: Results of QUIS for SkyLab

The system has been evaluated by 56 respondents by answering a survey based on Questionnaire for User Interface Satisfaction (QUIS) [19]. Respondents are asked to test features of SkyLab by following a set of instructions and using input files provided. 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. Most of the tools supported by SkyLab have inherently long processing time and the system does not focus on optimizing the said tools to achieve better performance.

V. 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 [18]. The system is also configured to install toolsets found in the modules folder making it possible to accomodate additional tools. Based on the user acceptance test conducted, the users found the system to be ...

The system achieved its main objectives but it can still be improved by adding more features. Input file generation can make the process more interactive and more customizable. Workflow design support will enable users to run complex tasks. Task scheduling and resource management algorithms can be used to efficiently handle resource-intensive or time consuming tasks. A special feature to borrow resources to idle clusters can be implemented.

References

[1] J. A. C. Hermocilla, "P2c: Towards scientific computing on private clouds," in *Proceedings of the National Conference on Information Technology Education (NCITE 2014)*, 2014, pp. 162–167.

[2] S. P. Ahuja and S. Mani, "The state of high performance computing in the cloud," *Journal of Emerging Trends in Computing and Information Sciences*, vol. 3, no. 2, pp. 263–266, Feb. 2012. [Online]. Available: http://www.chinacloud.cn/upload/2012-03/12031713036456.pdf

[3] I. Brandic, I. Raicu, S. N. Srirama, O. Batrashev, P. Jakovits, and E. Vainikko, "Scalability of parallel scientific applications on the cloud." *Scientific Programming*, vol. 19, no. 2/3, pp. 91 – 105.

[4] S. Benedict, "Performance issues and performance analysis tools for hpc cloud applications: a survey." *Computing*, vol. 95, no. 2, pp. 89 – 108.

[5] K. Jackson, K. Muriki, S. Canon, S. Cholia, J. Shalf, H. Wasserman, and N. Wright, "Performance analysis of high performance computing applications on the amazon web services cloud," 2010. [Online]. Available: https://www.nersc.gov/assets/NERSC-Staff-Publications/2010/CloudCom.pdf

[6] G. Juva, E. Deelman, K. Vahi, G. Mehta, B. Berman, B. Berriman, and P. Maechling, "Scientific workflow applications on amazon ec2." [Online]. Available: http://www.isi.edu/ gideon/publications/JuveG-EC2.pdf

[7] E. Walker, "Benchmarking amazon ec2 for high-performance scientific computing," Oct 2008. [Online]. Available: https://www.usenix.org/legacy/publications/login/2008-10/openpdfs/walker.pdf

[8] A. Iosup, S. Ostermann, N. Yigitbasi, R. Prodan, T. Fahrigner, and D. Epema, "Performance analysis of cloud computing services for many-tasks scientific computing."

[9] Z. Hill and M. Humphrey, "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).)*, Oct 2009. [Online]. Available: http://www.cs.virginia.edu/ humphrey/papers/QuantitativeAnalysisEC2.pdf

[10] A. A. Hunter, A. B. Macgregor, T. O. Szabo, C. A. Wellington, and M. I. Bellgard, "Yabi: An online research environment for grid, high performance and cloud computing." *Source Code for Biology and Medicine*, vol. 7, no. 1, pp. 1 – 10.

[11] P. Sharma and S. S. Mantri, "Wimpiblast: Web interface for mpiblast to help biologists perform large-scale annotation using high performance computing." *PLoS ONE*, vol. 9, no. 6, pp. 1 – 13.

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

[13] O. Trott and A. J. Olson, "Autodock vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading," *Journal of Computational Chemistry*, vol. 31, no. 2, pp. 455–461, 2010. [Online]. Available: http://dx.doi.org/10.1002/jcc.21334

[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, "Dock 6: Combining techniques to model RNA-small molecule complexes," *RNA*, vol. 15, pp. 1219–1230, 2009.

[15] P. Giannozzi, S. Baroni, N. Bonini, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, G. L. Chiarotti, M. Cococcioni, I. Dabo, A. Dal Corso, S. de Gironcoli, S. Fabris, G. Fratesi, R. Gebauer, U. Gerstmann, C. Gougoussis, A. Kokalj, M. Lazzeri, L. Martin-Samos, N. Marzari, F. Mauri, R. Mazzarello, S. Paolini, A. Pasquarello, L. Paulatto, C. Sbraccia, S. Scandolo, G. Sclauzero, A. P. Seitsonen, A. Smogunov, P. Umari, and R. M. Wentzcovitch, "Quantum espresso: a modular and open-source software project for quantum simulations of materials," *Journal of Physics: Condensed Matter*, vol. 21, no. 39, p. 395502 (19pp), 2009. [Online]. Available: http://www.quantum-espresso.org

[16] "General atomic and molecular electronic structure system," *M. W. Schmidt, K. K. Baldridge, J. A. Boatz, S. T. Elbert, M. S. Gordon, J. H. Jensen, S. Koseki, N. Matsunaga, K. A. Nguyen, S. Su, T. L. Windus, M. Dupuis, J. A. Montgomery J. Comput. Chem.*, vol. 14, 1993.

[17] S. Boisvert, F. Raymond, . Godzaridis, F. Laviolette, and J. Corbeil, "Ray Meta: scalable de novo metagenome assembly and profiling," *Genome Biology*, vol. 13, no. 12, p. R122, 2012. [Online]. Available: http://genomebiology.com/2012/13/12/R122

[18] R. M. Hanson, J. Prilusky, Z. Renjian, T. Nakane, and J. L. Sussman, "Jsmol and the next-generation web-based representation of 3d molecular structure as applied to proteopedia," *Israel Journal of*

Chemistry, vol. 53, no. 3-4, pp. 207–216, 2013. [Online]. Available: http://dx.doi.org/10.1002/ijch.201300024

[19] J. P. Chin, V. A. Diehl, and K. L. Norman, *Development of an instrument measuring user satisfaction of the human-computer interface*. New York: Proceedings of SIGCHI '88, 1988.