Configuration and Deployment of a Virtual Cluster for Molecular Docking Experiments

Kevin Lam, Jason Haga, Kohei Ichikawa

University of California San Diego, La Jolla, CA 92093, USA
Nara Institute of Science and Technology, Ikoma, Nara Prefecture 630-0192, Japan

---

Abstract

Molecular docking experiments that simulate ligand and protein receptor interactions is a popular tool among biomedical researchers. These tests can be done using programs such DOCK which virtually screens for compounds with the highest binding affinity. By executing the docking program on a grid system composed of interconnected computers, individuals will be able to narrow down their desired compound within a reasonable amount of time. In order to streamline and improve this method, the DOCK program was tested on a virtual cloud environment containing multiple virtual machines in a previous PRIME experiment [1]. Continuing from this, various virtual machines were created with different OS, compiler, and architecture configurations. The DOCK program was then similarly tested on each virtual machine. By comparing for accuracy between the results from the current experiment with the results from the previous experiment, the virtual machine configuration which yielded the most consistent results was determined. The chosen virtual machine was then cloned and deployed through automated scripts onto the PRAGMA grid system. With the virtual cluster accessible on the grid, future docking experiments will provide more consistent results and will be easier to manage.

Introduction

A grid is a collection of computers that share resources with each other through a network in order to complete a computationally intensive task [2]. The distributed grid system

used in this project is the Pacific Rim Applications and Grid Middleware Assembly (PRAGMA). PRAGMA is an organization composed of various researchers around the world [3]. With 35 research institutions contributing their expertise and resources, PRAGMA provides the environment necessary for grid and Cloud applications to develop [3].

Cloud computing is another function that PRAGMA offers. As opposed to grid computing, Cloud computing accesses resources through a service or application [4]. Since programs run off of virtual software rather than physical hardware, there is a higher level of scalability and portability [4]. In PRAGMA, each Cloud hosting site provides a server to store vi rtual machines (VM). These VMs are linked together across institutions to form the PRAGMA Cloud [3]. The Cloud is vital for this project. The primary reason for using the Cloud is for its support of virtual machines. Furthermore, in the case that more VMs are needed, the Cloud will easily be able to scale up or scale down on the amount. The flexibility of the Cloud will accommodate for any additional resources that are necessary to run the DOCK program.

Virtual machines make up most of the PRAGMA Cloud. A virtual machine is a type of application that emulates a computer. This creates an isolated virtual environment for users to utilize. Often, it is possible to run multiple operating systems on one computer using virtual machines [5]. By connecting these virtual machines together through a network and allowing a message-passing interface (MPI) software to exist between the machines, a cluster can be formed which allows it to share resources and behave as a simple computer [6]. Virtual machines are typically

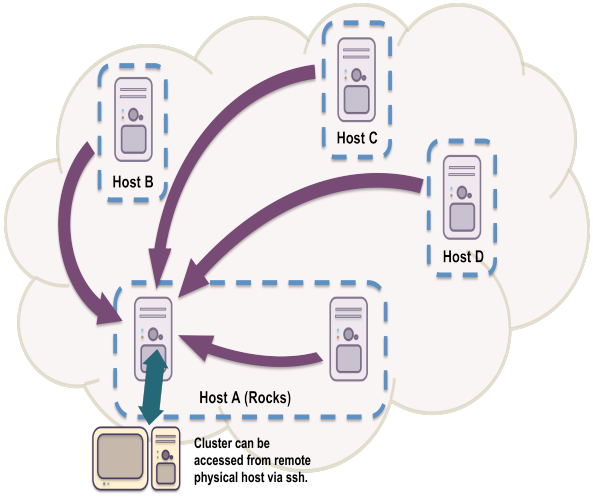

Figure 1. Virtual Cluster Setup

created using hypervisors which are also known as virtual machine managers [7]. These are usually installed separate from the virtual machine. However, recently, kernel-based virtual machines (KVM) have been available. This inbuilt virtualization software in Linux makes virtualization faster and more reliable.

There are a number of reasons why virtual machines and the Cloud are preferred over physical machines and the grid. The main reason is consistency of results. In the PRAGMA grid, every physical computer that is connected to the server has different specifications and configurations. This presents a problem for DOCK. The molecular docking program calculates its results based on a series of algorithms. There can often be differences in results caused by sources such as computer hardware, operating systems, and compilers that affect arithmetic precision [8]. Virtual machines can resolve this issue. As soon as a virtual machine is configured, they can be cloned to create replicas with identical specifications. With similar configurations, the results would be consistent from one virtual machine to the next. Another reason for choosing virtual machines over physical machines is the issue of scalability. It is relatively simple to increase or decrease the amount of computational resources necessary by cloning or deleting virtual machines. However, with physical machines, they would have to be purchased and manually configured-- both time-consuming and expensive.

DOCK is a program that virtually simulates the binding between a protein receptor and a ligand. A score is generated which shows the binding affinity between the two molecules. A smaller score would reflect a stronger bond. DOCK can be used to narrow down desired compounds before testing them in a lab. This can significantly lower costs in a research setting by reducing the amount of drugs that need to be tested.

Objective

The objective was to determine the virtual machine configuration which yielded the most consistent results. The virtual machine would then need to be cloned and deployed for testing on the PRAGMA Cloud. After approval of the machine, the clones would be distributed to host institutions. The virtual cluster would then be available for future usage.

Method

Four virtual machines were created using the virtual machine manager in order to test DOCK's performance with different configurations. Each VM was created using the KVM virtualization infrastructure that is integrated into Linux operating systems. The first instance, titled CentOS1, had a 64 machine bit size, a 64 operating system bit size, CentOS version 6.4, compiler gcc 4.7.7, and DOCK version 6.2 installed. The second instance, named Sailboat, had a 64 machine bit size, a 64 operating system bit size, CentOS version 5.9, compiler gcc 4.1.2, and DOCK version 6.2 installed. The third instance, named Master, had a 64 machine bit size, a 32 operating system bit size, CentOS version 5.2, compiler version gcc 4.1.2, and DOCK version 6.2 installed. The last instance, called Masternode, had a 32 machine bit size, a 32 operating system bit size, CentOS version 5.9, compiler version gcc 4.1.2, and DOCK version 6.2 installed.

Table 1. Configurations of Virtual Machines

Virtual Machine	Masternode	Master	Sailboat	CentOS1
Machine Bit	32	64	64	64
Operating bit	32	32	64	64
CentOS Version	5.9	5.2	5.9	6.4
Compiler Version	4.1.2	4.1.2	4.1.2	4.7.7
Dock Version	6.2	6.2	6.2	6.2

With the virtual machines created, the host files, resolv.conf files, network files, and ifcfg files were changed to allow for internet connection and communication between the machines.

Password-less Secure Shell was enabled for file transfer capability. A user account was created to allow general usage in the virtual machine. DOCK and MPICH were installed which enabled parallel processing between the nodes. The DOCK version 6.2 test suite was run using the standardized files located in the test folder. In order to determine DOCK result consistency, the test results need to be compared with the standardized developer's scores as well as with the scores from the previous PRIME project [1]. The machine with the most accurate results was selected and cloned. Lastly, both the cloned and original virtual machine were deployed by automated scripts onto the PRAGMA Cloud.

Results

The virtual machine, Masternode, had the most consistent results. In the previous PRIME project, Yim was attempting to match the results she obtained with the developer's standardized scores [1]. Our goal was similar except we tested various virtual machine setups. Masternode and Master matched the developer's scores on nine out of eleven ligands, whereas, Sailboat and CentOS1 only had similar results in six out of the eleven ligands. In this case, both Masternode and Master had similar results. Masternode was chosen because it contained CentOS version 5.9--a more preferred recent update of the operating system. Using the same configurations, a new VM named Barco was created along with a user account named Capitan. Automated scripts were used to deploy the virtual machine onto PRAGMA servers. N2N, a peer-to-peer virtual private network, was installed to allow data transmission between VMs despite firewall security.

Conclusion

The virtual machine Barco, with the same configurations as Masternode, has been created and successfully deployed onto the PRAGMA Cloud using automated scripts. The operating

Table 2. Virtual Machine Test Suite Results. Scores highlighted in red indicate a deviance from the developer's standardized scores.

Ligand IDs	Developer's	Masternode	Master	Sailboat	CentOS1
ZINC00158751	-1.058107	138.11386	138.113861	138.114029	138.114029
ZINC00157960	847.37207	21535.20898	21535.20898	21535.30859	21535.30859
ZINC00158442	52.56588	52.56588	52.56588	52.565788	52.565788
ZINC00013564	10.801939	10.801939	10.801939	-9.400218	-9.400218
ZINC01555236	503.249725	503.249725	503.249725	1800544512	1800544512
ZINC00150863	21.139143	21.139143	21.139143	-12.467216	-12.467216
ZINC00152265	30.238361	30.238361	30.238361	30.238028	30.238028
ZINC00157111	-12.916615	-12.916615	-12.916615	-12.916644	-12.916644
ZINC00157152	-10.137384	-10.137384	-10.137384	-10.137392	-10.137392
ZINC00157402	168513.625	168513.625	168513.625	168506.4063	168506.4063
ZINC00157467	-8.706671	-8.706671	-8.706671	-8.706783	-8.706783

system, machine bit, and other settings were chosen because it proved to produce the most consistent results out of all the tested virtual machines. For future studies, the virtual machines and virtual clusters can be extended for further research. The capabilities of these clusters have yet to be fully tested. With the development of these virtual machines, projects in the future involving DOCK will be streamlined and more simplified.

References

1. Yim, W., Chien S., Kusumoto Y., Date S., Haga J. (2010). Grid Heterogeneity in In-silico Experiments: An Exploration of Drug Screening Using DOCK on Cloud Environments. *Studies in Health Technology and Informatics*, 181-190.

2. Kaufman, J. H., Deen G., Lehman T. J., and Thomas J. (2003). Grid Computing Made Simple. *The Industrial Physicist*, 31-33.

3. PRAGMA, Pacific Rim Applications and Grid Middleware Assembly. (2013). Cloud/Grid Operations Center. Retrieved October 16, 2013 from http://goc.pragma-grid.net/.

4. Hashemi, S. M., Bardsiri, A. K. (2012). Cloud Computing Vs. Grid Computing. *ARPN Journal of Systems and Software*, 2(188-194).

5. Kietzman, S. (2013). What Is a Virtual Machine?. Retrieved October 16, 2013 from http://www.wisegeek.org/what-is-a-virtual-machine.htm.

6. Scalable Computing Laboratory: Ames Lab. (2013). What is a Cluster Computer? Retrieved October 17, 2013 from http://www.scl.ameslab.gov/Projects/parallel_computing/cluster_basics.html.

7. Hypervisor. (n.d). In *Technopedia*. Retrieved from http://www.techopedia.com/definition/4790/hypervisor.

8. UCSF. (2013). Dock 6.6 Users Manual. Retrieved October 17, 2013 from http://dock.compbio.ucsf.edu/DOCK_6/dock6_manual.htm.