Bioinformation Volume 5

open access

www.bioinformation.net Issue 5

Hypothesis

A comparative analysis of dynamic grids vs. virtual grids using the A^{3p}viGrid framework

Avinash Shankaranarayanan[1,] *, Christine Amaldas[2]

[1]School of Chemical and Biotechnology, SASTRA University; [2] Business Information Systems, Royal Melbourne Institute of Technology, Avinash Shankaranarayanan: Email- avigrid@gmail.com; *Corresponding author.

Received July 26, 2010; Accepted August 26, 2010; Published November 1, 2010

Abstract:
With the proliferation of Quad/Multi-core micro-processors in mainstream platforms such as desktops and workstations; a large number of unused CPU cycles can be utilized for running virtual machines (VMs) as dynamic nodes in distributed environments. Grid services and its service oriented business broker now termed cloud computing could deploy image based virtualization platforms enabling agent based resource management and dynamic fault management. In this paper we present an efficient way of utilizing heterogeneous virtual machines on idle desktops as an environment for consumption of high performance grid services. Spurious and exponential increases in the size of the datasets are constant concerns in medical and pharmaceutical industries due to the constant discovery and publication of large sequence databases. Traditional algorithms are not modeled at handing large data sizes under sudden and dynamic changes in the execution environment as previously discussed. This research was undertaken to compare our previous results with running the same test dataset with that of a virtual Grid platform using virtual machines (Virtualization). The implemented architecture, A^{3p}viGrid utilizes game theoretic optimization and agent based team formation (Coalition) algorithms to improve upon scalability with respect to team formation. Due to the dynamic nature of distributed systems (as discussed in our previous work) all interactions were made local within a team transparently. This paper is a proof of concept of an experimental mini-Grid test-bed compared to running the platform on local virtual machines on a local test cluster. This was done to give every agent its own execution platform enabling anonymity and better control of the dynamic environmental parameters. We also analyze performance and scalability of Blast in a multiple virtual node setup and present our findings. This paper is an extension of our previous research on improving the BLAST application framework using dynamic Grids on virtualization platforms such as the virtual box.

Keywords: Agents, Blast, Coalition, Grids, Virtual Machines and Virtualization.

Background:
Bioinformatics heavily **[7, 8]** relies upon statistical and analytical methods of processing biological data. Some of the important biological research aims at studying the evolutionary effects of gene mutation and similarity between gene sequences using computer technology. This aids biologists to find and cure disease causing viruses by applying new and faster methods of drug discovery in the laboratory. Substantial discoveries of new life forms and drugs takes place on a daily basis leading to biological data being stored into remote databases (resources). The exponential increase in the size of datasets makes it mandatory for biologists to opt for better methods of computing genomic data. Biologists use different types of sequence comparison tools and software packages to speed up experimental research. The problem of organizing information and sharing knowledge with the scientific community at the gene level isn't being tackled by developing a nomenclature. Instead, computational techniques were applied to improve the organization of information in databases which lead to the era of computational biology.

The paper is subdivided into the following sections: Section II will give an overview of current Blast Literature with insights into the distributed systems and Virtual Grids; Section III will talk about the A^{3p}viGrid framework **[4]** and how it functions followed by the differences in performances between running our Blast application in our mini-grid test-bed and comparing it to running individual agents on virtual machine work spaces; in Section IV and V we conclude the paper with discussions about the results obtained followed by future enhancements to our research work.

About Virtual Grids and Bioinformatics Blast:
Virtual grids are described by a virtual grid resource specification that is presented by the application to acquire resources for execution. A virtual grid resource specification captures the desired resources for an application, and its explicit resource structure can be used by the application designer to express parallelism, communication, and other forms of optimization. The primary goal of grid computing platforms is to seamlessly multiplex distributed computational resources with its associated providers and end users across wide area networks **[12]**. In traditional computing environments, resources are multiplexed based on typical operating systems confined to limited resources. With the proliferation of Quad/Multi-core micro-processors in mainstream platforms such as desktops and workstations; a large number of unused CPU cycles can be utilized for running virtual machines as dynamic nodes in distributed environments as Grid services and its service oriented business broker now popularized as cloud computing. Numerous advantages such as dynamic sizing of compute nodes and resources are presented here which can be user controlled within a secured environment. Further the deployment of image based virtualization platforms enables resource management and dynamic fault management in a controlled manner. End users of high performance compute nodes always expect to meet some challenges while deploying Grid resources in the form of services. In this paper, we propose a new methodology for Grid computing; to use virtual machines as Virtual Grid Environments (VGE) that provides computing resources to Grid users having customized requirements originating from different platforms having varied Quality of Service (QoS) constraints. The

Bioinformation Volume 5

open access

www.bioinformation.net Issue 5 **Hypothesis**

ability to share resources is a basic requirement for the deployment of grids while observing the integrity and security of shared resources is of utmost importance. Security models need to address resource providers who enable trust or integrity mechanisms that restrict the application of grids based on mutual trust between resource providers (brokers) and users.
Virtual machines address three fundamental requirements: support for legacy applications, security against un-trusted program execution and users, and independent resource deployment and administration. Virtual machines can be divided into two main categories **[10]**: those that virtualize complete instruction set architecture (ISA-VMs) including both user and system instructions; supports an application binary interface with virtualization of system calls **[2].** An important class of virtual machines **[11, 12, 14]** consists of ISA-VMs that support same-ISA execution of entire operating systems such as IBM S/390 series **[18]** and VMware **[10]**, and the open-source project Virtual box **[10]** used in our test case. Virtual machines can be highly customizable without requiring system restarts. It is possible to specify virtual hardware parameters: memory and disk sizes; system software parameters such as operating system modules **[13, 15, 18]** loading on demand and kernel configuration. We can agree that deploying virtual environments for Grid computing can bring about user enabled compute and resource customization, QoS sharing, data manipulation and easy management. Instead of complicating users with a Grid middleware and Virtualization Engines our existing framework A^{3p}viGrid **[4, 6]** architecture was utilized for virtualization.

Biologists often require sequence comparison and alignment applications such as Basic Local Alignment Search Tools or BLAST **[9, 13]**, which are effectively utilized for processing large sets of gene sequences for similarity matching. These tools have been previously extensively investigated **[4]** and evaluated. BLAST is a set of programs used for searching sequence databases with that of the input query sequence for similarity matching. BLAST is a heuristic search method which makes assumptions about the data based on experience. This implies that it is not guaranteed to find the best alignment in all possible circumstances. It sacrifices some accuracy for a great increase in speed. The BLAST has similarities to the Smith-Waterman algorithm **[15]**, which is slow but guaranteed to get the best possible alignments given certain input parameters. BLAST uses a special database format to speed up the search operation. Several pre-packaged databases exists, and the most notable is the "nr" database which is the non-redundant database consisting of all sequences in GenBank. BLAST users can take advantage of low-cost, efficient Linux cluster architectures such as Beowulf. Unfortunately, the efficiency declines when scaled to hundreds of nodes because of serial result-merging and output domination **[10]**. A 300-KB query against the 5.1-GB uncompressed 'nt' database takes 1346 minutes (or 22.4 hours) on one compute node. The same query was run within 8 minutes on 128 nodes on the Green Destiny supercomputing cluster. A more detailed performance analysis and evaluation can be found in the green destiny paper **[5]**. Arun Krishnan in his paper **[1]**, talks about applying BLAST to the Globus Grid platform **[17]** using Perl scripts called GridBLAST on a mini-grid test bed. When looking at the computational aspects of BLAST **[16]**, typically a full scale BLAST job across whole genomes is highly computationally intensive due to the size of the databases queried upon. The following section will briefly describe our frame work which was deployed on a virtualization platform and compared to our previous results **[4]**.

Running the A^{3p}viGrid agents on virtual machines:
The ability to invoke a program or workflow say a servlet using a web server can be effectively utilized towards distributed processing of data. This is termed as the *"power server model"* of computing. The advantage is the simplicity of the model where the client connects to a bunch of web servers to enable the consumption of remote services using web pages.

A^{3p}viGrid works on the principle of the power server model of computing. Each of the clients run the A^{3p}viGrid server which is a simplistic http web server running services in the form of CGI/Perl wrapper Scripts. The client side-coding model enables the developer to develop services using the common gateway interface (CGI) and can use any of the languages that support CGI scripting. For the sake of simplicity and rapid development of services we have used Perl as the language of choice due to its availability and portability for most platforms. The A^{3p}viGrid uses a decentralized directory structure (APM) to enable peers to register and de-register peers and their respective services **[4]**.

A random set of 10 machines was used for job processing. All the nodes ran A^{3p}viGrid web servers. The Blast.apm file, a directory structure file that is local to all nodes was downloaded by all the peers as part of the initialization phase. This file contains information such as location information of nearby agents, domain and IP address and other important data. Each of the nodes compute the ideal set of nodes using a basic ping test based on the Blast grid service list. As all the nodes are capable of receiving jobs, one of them was randomly chosen for job execution (Originator). A Fasta formatted Sequence database (human DNA sequence from clone RP11-10K8 on chromosome 1) was used to evaluate the Blast searches. The input query file was obtained, and a set of jobs for job processing was prepared using the optimal coalition list. Based on QoS characteristics namely Latency, Load **[3]** and CPU time, the Originator of the job computes the most optimal coalition. Once the coalition list is computed the data files are migrated using the POST method to all the members of the coalition. Each of the coalition members start to search using the input query files and output the results to an output file.

The output of the Search Phase is appended to a file using POST back to Originator where the results are formatted using the Blast format perl script and stored as a file or displayed in the browser of the originator. Each of the agents ran on a virtual machine test bed having their own execution environments. For the sake of true heterogeneous functionality and testing, four operating environments were deployed namely: Fedora Linux Core, Windows Vista Ultimate, Mac OS Leopard and Sun's Open Solaris 10. Each of the agents were given a resource limit which shared the following specifications: 10 GB disk space; 4 GB RAM and Dual 2 GHZ CPU Cores. All VM's were equally created as disk images and were run on 10 networked computers each hosting the four agents (on four core operating environments). The new Gigabyte IRAM modules were installed towards testing the improvements in I/O access to the data file where all VM's were equally loaded using the Virtual Box open source virtualization software. To cater to a heterogeneous environment and make it truly a peer-to-peer model of computing, all nodes were connected over the Internet using DSL or Cisco routers and Cable modem lines.

The turnaround and compute time were computed as follows: we assume N data distributed over $P = 2^d$ tasks, with N an integer multiple of the computation costs which comprise of the initial comparisons performed during the communication phase where $d = \log P$. The former involves a total of $P = 2^d$ comparisons, while the latter requires at most $(N_d (d+1) /2)$ comparisons. Because the algorithm is perfectly balanced, we assume that idle time is negligible. Our results were obtained by running Gridblast code on Linux Clusters (Fedora Core) with 2.0 GHz Duo core CPU's and 4GB RAM. A heterogeneous set of peers (three nodes running Linux Fedora core; four nodes running Windows Vista Ultimate, three nodes running Sun Open Solaris 10) having different configurations were used for running the algorithm as a Grid service using the A^{3p}viGrid agents running on their VM's or individual user space. In this project, human DNA sequence (GenBankID: AL611946) has been used as the database of choice. The size of this sequence is 44,921 base pairs (bp).

Bioinformation Volume 5

open access

www.bioinformation.net Issue 5

Hypothesis

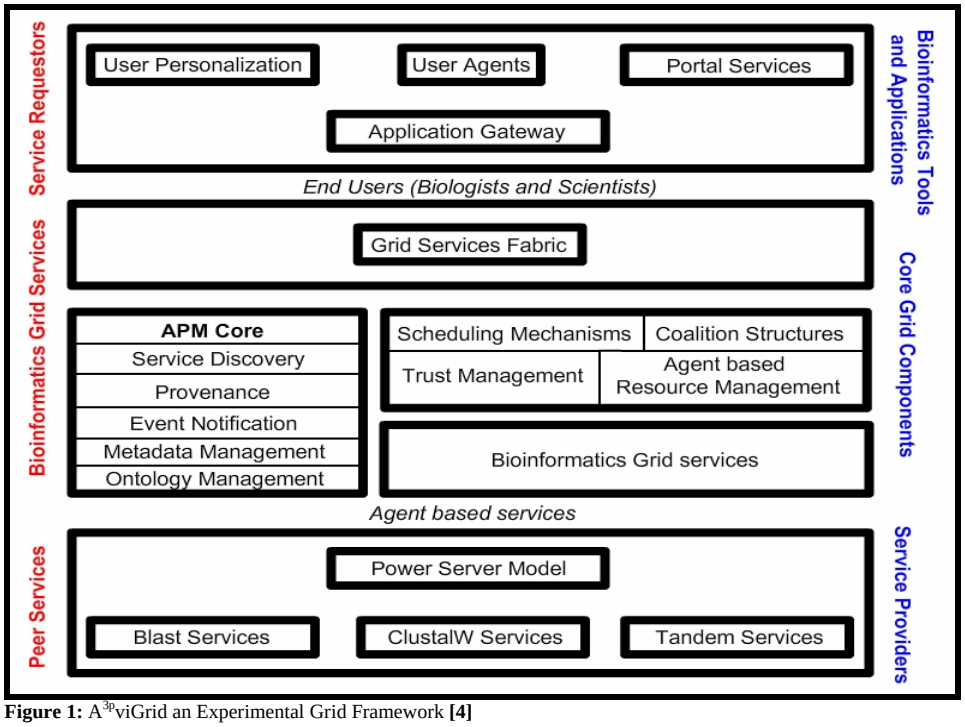

Figure 1: A[3p]viGrid an Experimental Grid Framework **[4]**

Results:

Initial Results:

All of the A[3p]viGrid agents initially ran on individual workstations and the initial results were obtained with a mini-grid test-bed of 10 nodes. The results indicate the time of execution taken as the average value of the two experiments with the same settings and parameters in place. **(Table 1, 2 see supplementary material)**

Results with Agents running on VM's:

The initial data set was stored and written to scratch disks created in RAM along with accessing and storing results on the iRAM installed on the head node (where the initial job was submitted).The results are shown in **(Table 3, 4 see supplementary material)**. From the data recorded we can estimate that the initial turnaround time was affected due to an increase in latency posed by the VM's during initialization and data retrieval. As we can observe from table 3 the overall turnaround time almost increases two fold during initial execution as resources are allocated dynamically by the agents during execution. From table 4, the researchers observed that once the data was made available, the execution time was decreased more than half after the agent and its environment were initialized. A two-fold speedup can be observed based on running agents in virtual machines as the input/output data access time is cut by half as resources and data were made available locally to the agents using virtual machines.

Conclusion and Future directions:

To improve application and agent specific performance, customized Virtual execution environments (Virtual Machines) were created for each of the agents running the A[3p]viGrid service. An increase in performance after initialization and execution of agents on the VM's was observed. A coalition based approach to solving a known problem in bioinformatics was undertaken. The use of RAM based scratch disks proved useful in improving the execution times of the BLAST searches on the Mini-Grid test bed. It was found that the A[3p]viGrid framework fairs well against embarrassingly parallel bioinformatics applications such as Blast. The scalability of the Mini-Grid test bed is based on numerous factors such as the resources available; the operating environments and the speedup observed after virtualization is applied. A query search approach was undertaken and we still need to try and apply query splitting to see how the A[3p]viGrid framework fairs with similar datasets. Future research would be towards this direction.

Acknowledgements:

This project is part of a Masters by Research dissertation at SASTRA University, Thanjavur, India. We would like to take this opportunity to thank Dr K.N. Somasekaran, Dean, Department of Chemical and Biotechnology, SASTRA University for his valuable comments and feedback.

References:

[1] A Krishnan. *Concurrency and Computation: Practice and Experience* **17**:1607 (2005).
[2] Altschul *et al. Journal of Molecular Biology* **215**:403 (1990) [PMID: 2231712]
[3] A Shankar *et al. PDPTA* **27**:30 (2005)
[4] Shankaranarayanan *et al. International Journal of Genetic Engineering and Biotechnology* **1:** 23 (2010)
[5] Shankaranarayanan *et al. IEEE Computer Society, CIMCA-IAWTIC'06* **2**: 315 (2006)
[6] A Darling *et al. ClusterWorld Conference & Expo* **311** (2003).
[7] S Burt *et al. BMC Bioinformatics* **6**: 168 (2005) [PMCID: PMC1190154]
[8] C Gibas *et al. O'Reilly & Associates* 126 (2001).
[9] DG Higgins *et al. Methods Enzymol* **266**:383 (1996).
[10] J Marshall *et al. Comput Methods Programs Biomed* **93**(1): 73 (2009) [PMCID: PMC2665129]
[11] I Foster *et al. The International Journal of Supercomputer Applications and High Performance Computing.* **11**(2):115 (1997)
[12] A Konagaya. *BMC Bioinformatics* **7**(Suppl 5): S10 (2006)

[13] R Braun *et al. Future Generation Computer Systems* **17**(6):745 (2001)
[14] Joseph *et al. Microprocess Microsyst* **33**(4): 281 (2009) [PMCID: PMC2771927]
[15] Jacob *et al. ACM Trans Reconfigurable Technol Syst* **1**(2): 9 (2008) [PMCID: PMC2615407]
[16] Isaac *et al. BMC Bioinformatics* **8**:185 (2007) [PMCID: PMC1896180]
[17] Paulo *et al. BMC Bioinformatics* **6**: 197 (2005) [PMCID: PMC1190159]
[18] V Talukdar *et al. Biotechnol J* **4**(9):1244 (2009) [PMCID: PMC2697647]

Edited by P. Kangueane
Citation: Shankaranarayanan *et al.* Bioinformation 5(5): 186- 190 (2010)

Bioinformation Volume 5

open access

www.bioinformation.net Issue 5 **Hypothesis**

Supplementary material:

Tables 1: Turnaround times recorded on the A^{3p}viGrid test-bed without VM's **[4].**

	Processor 0	Processor 1	Processor 2	Processor 3
Seq Blastcode	43.0763	63.8885	48.526	83.8911
Grid Blastcode	61.633	59.6393	58.6436	58.6373
	Processor 5	Processor 6	Processor 7	Processor 8
Seq Blastcode	56.8615	44.2549	50.1826	41.5324
Grid Blastcode	58.6441	53.4334	62.6326	58.6406

Tables 2: Turnaround times recorded on the A^{3p}viGrid test-bed without VM's **[4].**

	Processor 4	Processor 5	Processor 6	Processor 7
7Seq Blastcode	52.575	53.1896	86.0856	118.026
Grid Blastcode	72.4102	72.4286	72.5143	72.4188
	Processor 4	Processor 5	Processor 6	Processor 7
Seq Blastcode	51.9337	52.769	51.5515	51.2998
Grid Blastcode	42.4297	32.4361	42.4246	56.4203

Tables 3: Turnaround Time (in seconds) is measured as wall-clock execution time from the beginning to the end of the task execution of the Agent run on VM's.

	Processor 0	Processor 1	Processor 2	Processor 3
Seq Blastcode	43.0763	63.8885	48.526	83.8911
Grid Blastcode (On VMs)	51.221	60.3244	59.6436	51.6373
	Processor 4	Processor 5	Processor 6	Processor 7
Seq Blastcode	56.8615	44.2549	50.1826	41.5324
Grid Blastcode (On VMs)	48.6441	44.4334	62.6326	48.6406

Tables 4: Turnaround Time (in seconds) is measured as wall-clock execution time from the beginning to the end of the task execution of the Agent run on VM's.

	Processor 4	Processor 5	Processor 6	Processor 7
Seq Blastcode	52.575	53.1896	86.0856	118.026
Grid Blastcode (On VM s)	42.4102	32.4286	47.5143	43.4188
	Processor 4	Processor 5	Processor 6	Processor 7
Seq Blastcode	51.9337	52.769	51.5515	51.2998
Grid Blastcode (On VM s)	40.4297	32.4361	36.4246	36.4203