Fourth IEEE International Conference on eScience

CloudBLAST: Combining MapReduce and Virtualization on Distributed Resources for Bioinformatics Applications

Andréa Matsunaga, Maurício Tsugawa and José Fortes
Advanced Computing and Information Systems Laboratory
Department of Electrical and Computer Engineering, University of Florida
PO Box 116200, Gainesville, FL, 32611-6200, USA
{ammatsun, tsugawa, fortes}@ufl.edu

***Abstract*—This paper proposes and evaluates an approach to the parallelization, deployment and management of bioinformatics applications that integrates several emerging technologies for distributed computing. The proposed approach uses the MapReduce paradigm to parallelize tools and manage their execution, machine virtualization to encapsulate their execution environments and commonly used data sets into flexibly deployable virtual machines, and network virtualization to connect resources behind firewalls/NATs while preserving the necessary performance and the communication environment. An implementation of this approach is described and used to demonstrate and evaluate the proposed approach. The implementation integrates Hadoop, Virtual Workspaces, and ViNe as the MapReduce, virtual machine and virtual network technologies, respectively, to deploy the commonly used bioinformatics tool NCBI BLAST on a WAN-based test bed consisting of clusters at two distinct locations, the University of Florida and the University of Chicago. This WAN-based implementation, called CloudBLAST, was evaluated against both non-virtualized and LAN-based implementations in order to assess the overheads of machine and network virtualization, which were shown to be insignificant. To compare the proposed approach against an MPI-based solution, CloudBLAST performance was experimentally contrasted against the publicly available mpiBLAST on the same WAN-based test bed. Both versions demonstrated performance gains as the number of available processors increased, with CloudBLAST delivering speedups of 57 against 52.4 of MPI version, when 64 processors on 2 sites were used. The results encourage the use of the proposed approach for the execution of large-scale bioinformatics applications on emerging distributed environments that provide access to computing resources as a service.**

Keywords-Cloud computing; virtualization; mapreduce; bioinformatics.

I. INTRODUCTION

There is tremendous potential for end users in many fields of science to routinely conduct large-scale computations on distributed resources by using a combination of the following emerging technologies:

- Distributed middleware for connecting data/cluster computing centers; this includes Grid-computing middleware [1][2][3][4] for user authentication and accounting, remote job submission, resource scheduling/reservation, and data management;
- Virtualization technologies capable of providing application-specific execution environments; these include hypervisor-based virtual machines (VMs) (e.g., Citrix Xen, VMware, Microsoft, KVM), virtual networks (VNs) [5][6][7][8][9][10], VM and VN management middleware [11][12][13];
- Programming platforms and runtime environments that facilitate the parallelization of applications and their execution as concurrent jobs on subsets of the input data. (Bioinformatics is an example of a scientific domain where many such applications exist.) These environments greatly facilitate or transparently provide management of application execution, e.g. for scheduling, monitoring and fault tolerance purposes; an example of this type of technology is MapReduce [14].

This paper shows that, by properly integrating the above technologies, one can envision services for end users to conveniently launch their scientific applications with minimal (if any) programming, deployment and management efforts. In addition, it addresses the open question of whether the execution overheads added by these technologies also introduce performance or scalability penalties that mitigate their stated advantages. In this context, the paper makes the following contributions:

- It shows that machine virtualization, network virtualization and Hadoop [15] can be combined to deploy important bioinformatics applications based on BLAST [16] on computer clusters on distinct administrative domains connected by a wide-area network (WAN).
- It experimentally validates the approach by deploying a Xen-based virtual cluster across two sites, at the University of Chicago (UC) and the University of Florida (UF), using virtual workspaces [11] for authenticating users and deploying VMs, and ViNe [6] for connecting the nodes behind NATs, over a 200Mbps WAN link.

978-0-7695-3535-7/08 $25.00 © 2008 IEEE
DOI 10.1109/eScience.2008.62

- It evaluates the approach by comparing its performance with both purely sequential implementations and a leading MPI-based solution (also proposed and deployed for the first time by this work on a non-emulated WAN-based virtual network crossing distinct administrative domains).
- It evaluates the overhead introduced by virtualization and Hadoop by comparing the performance of the BLAST application using physical and virtual machines, on LAN and WAN clusters, and with Hadoop and MPI.

Experimental results show that the overheads of the above-mentioned technologies (virtual workspaces and ViNe) are insignificant, thus demonstrating the feasibility and benefits of executing BLAST on virtualized resources and across sites. CloudBLAST and mpiBLAST presented similar performance with a small advantage to CloudBLAST in a scenario where the target database fits in memory.

This paper is organized as follows. Section II discusses how to address challenges faced in deploying bioinformatics applications on distributed resources, and presents the proposed approach combining VM, VN and MapReduce technologies. Section III describes the distributed environment used in the experiments. Section IV analyzes the experimental results and Section V discusses the conclusions from this work.

II. Distributed Bioinformatics Computing

Bioinformatics is being confronted with increasingly larger data sets leading to computational jobs that take unacceptably long times if done on a small number of machines. For these cases, distributed computing on multiple clusters at different locations is becoming an attractive, if not necessary, approach to achieve short execution times.

Grid-computing approaches have been developed with the primary goal of enabling the use of distributed pools of resources. The currently popular notion of "cloud computing" [17][18][19], while imprecisely defined, broadly refers to the use of managed distributed resources to deliver services to multiple users, often using virtualization to provision execution environments as needed by applications and users. The vision, shared with Grid and utility computing, is for users to be able to access and pay for computational services just as conveniently as when using electrical appliances powered by the electrical grid [17].

Users of bioinformatics tools are typically unprepared to take advantage of these and other emerging technologies for distributed computing. One reason is the need to recast sequential tools as distributed jobs. Another reason is the challenge of understanding Grid technologies well enough to create and manage a distributed environment, which may require changes or additions of system software, and capabilities to schedule tasks, recover from failures and monitor distributed tasks.

More specifically, a bioinformatics researcher would face the following interrelated tasks:

1. Parallelize the tool so it can run as multiple concurrent computational jobs.
2. Identify, allocate and prepare machines with the necessary software environment for the execution of each of the parallel jobs and the storage of datasets.
3. Ensure that machines are networked so that the necessary communication among concurrent jobs can take place.
4. Deploy and manage the execution of the parallel tool on distributed resources.

In the following subsections the options available for each of the tasks are discussed, and the choices proposed by this paper are identified on the basis of their suitability for integration into a combined solution with good performance, low management complexity and minimal overhead.

A. Parallelizing Applications

Though many bioinformatics tools are parallelizable, parallelism is usually not explicitly expressed in their original codes. Users interested in parallelizing the execution of an application are required to either (a) develop or adapt existing software to distribute and manage parallel jobs, or (b) modify existing applications to make use of libraries that facilitate distributed programming such as MPI, OpenMP, RPC and RMI. When modifying applications, the effort is recurrent since new versions of the original sequential code may render the parallelized application obsolete. The amount of work involved may lead to parallel versions that lag and lack in features of the latest sequential tool version. Typically, modified applications will not include mechanisms to automatically handle failures.

Often, users opt for approach (a) because it is too expensive to modify existing code or source code may not be available. This is particularly true when it is relatively easy to parallelize applications that can run on multiple resources, each of which executes the sequential application on a subset of the inputs. Nevertheless, this solution requires additional software to manage and monitor job distribution and possibly offer fault tolerance:[20][21][22][23][24] are a small sample of Grid-based solutions for bioinformatics applications that automate the process of transferring or sharing large files, selecting appropriate application binaries for a variety of environments or existing services, managing the creation and submission of a large number of jobs to be executed in parallel and recovering from possible failures. This work advocates the use of the MapReduce framework [14], which combines many features and lessons learnt from the Grid solutions.

To better depict the parallelization process, it is useful to consider a specific widely used tool - Basic Local Alignment Search Tool (BLAST) [16]. BLAST finds regions of local similarity between nucleotide or protein sequences. Given a set of k sequences $S=\{s_1, \ldots, s_k\}$, the program compares each sequence s_i to a database of n sequences $D=\{ds_1, \ldots, ds_n\}$ and calculates the statistical significance of matches. Two usual approaches to execute BLAST in a distributed

environment (Fig. 1) are: (a) to partition the input sequences so that each of separate instances of BLAST (called workers) looks for similarities of a subset of S, or (b) to partition not only the input sequences but also the database D. The second approach allows the database to grow and still fit on the local disk or local memory of worker nodes, at the cost of workers needing to communicate among themselves to generate the final combined comparison result.

The first approach (i.e. partitioning only the input sequences) can be easily implemented using a scripting language to identify the input sequences, execute BLAST, and concatenate the result. For example, a bash script would simply use csplit, ssh/rsh, and cat commands. However, in practice users still face the following challenges: (a) finding the ideal number of workers to use (or that are available) and split accordingly, (b) load balancing and providing balanced partitions, since the time of BLAST execution is highly dependent on the length of an input sequence, and (c) recovering from the potential failure of some workers in order to avoid obtaining only partial results.

To handle these issues more efficiently, the use of MapReduce for this category of embarrassingly parallel applications is shown next. The MapReduce as a software framework for distributed computation with fault tolerance was proposed by Google, inspired by functional languages, to realize parallel computing on a large number of unreliable resources. The programmer is required to implement a map function and a reduce function. The map function takes input records and generates intermediate key and value pairs. The reduce function takes all the values for an intermediary key and generates the result of processing these values. MapReduce framework is responsible for automatically splitting the input, distributing each chunk to workers (called mappers) on multiple machines, grouping and sorting all intermediary values associated with an intermediary key, passing these values to workers (called reducers) on multiple resources, and monitoring the execution of mappers and reducers as to re-execute them when failures are detected. These characteristics greatly simplify deployment and management of jobs (task 4) as jobs are submitted and controlled from a central location.

A classical example [14] of using the MapReduce framework is that of counting the number of occurrences of words in a large collection of web documents. The mapper analyzes an input and for each word w, a key/value pair w/1 is generated. The reducer receives a collection of one (1) values for a particular word, which can be added to produce the desired count for that word.

One possible setup to run the sequential BLAST program in the MapReduce framework is to assign to each mapper the execution of the BLAST operation for a subset of sequences thus eliminating the need for a reducer. However, in other bioinformatics applications the reducer would be useful – for example, the user may require the output of BLAST to be classified in some manner.

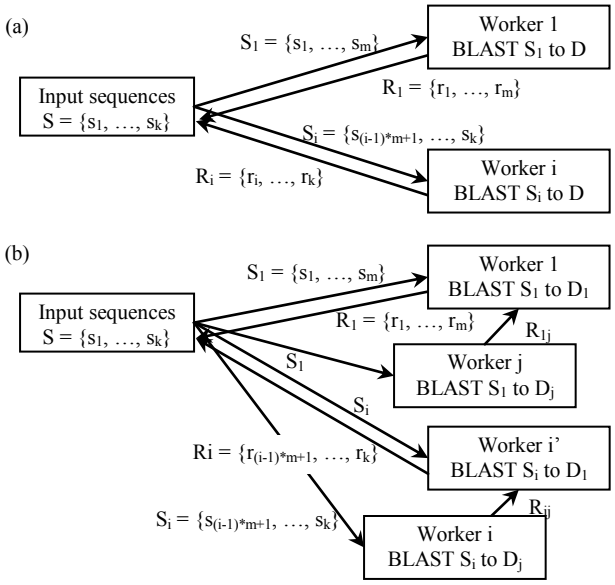

Figure 1. Parallelization of BLAST: (a) input sequences are partitioned into i subsets, each of which is processed by a worker, and results are concatenated in order, (b) input sequences are partitioned into i subsets and the database is partitioned into j subsets, each pair of subsets is processed by a worker, results from each database partition are sorted and merged, and results are concatenated in order.

In this work, Apache Hadoop [15], an open-source implementation of the MapReduce paradigm, was used to parallelize the execution of NCBI BLAST2, a sequential implementation of BLAST made publicly available by the National Center for Biotechnology Information. The parallelization approach consists of segmenting the input sequences and running multiple instances of the unmodified NCBI BLAST2 on each segment, as illustrated in Fig. 1a. Hadoop offers a streaming extension that allows easy execution of existing applications. To deal with the fact that Hadoop splits the input file into chunks of approximately the same size independently of the boundaries of a nucleotide or protein sequence, a pattern record reader was developed to retrieve entire sequences as keys of a key/value pair. To combine all results, the merge command of the Hadoop Distributed File System (DFS) was used. This process is depicted in Fig. 2.

Although only experiments with BLAST are shown, the proposed solution also applies to other applications with similar execution profile, such as HMMER [25], megablast [26] and other derivatives of BLAST.

B. Application Deployment and Maintenance

With a parallelized application at hand, the next step is to find resources with the execution environment required by the application (e.g., MPI, Hadoop); this can be an arduous task as the resources on the Grid are naturally heterogeneous with respect to operating system and libraries supporting applications. For example, an MPI-based approach requires the same MPI library (e.g. MPICH, OpenMPI) which in turn may require machines to run one of several possible

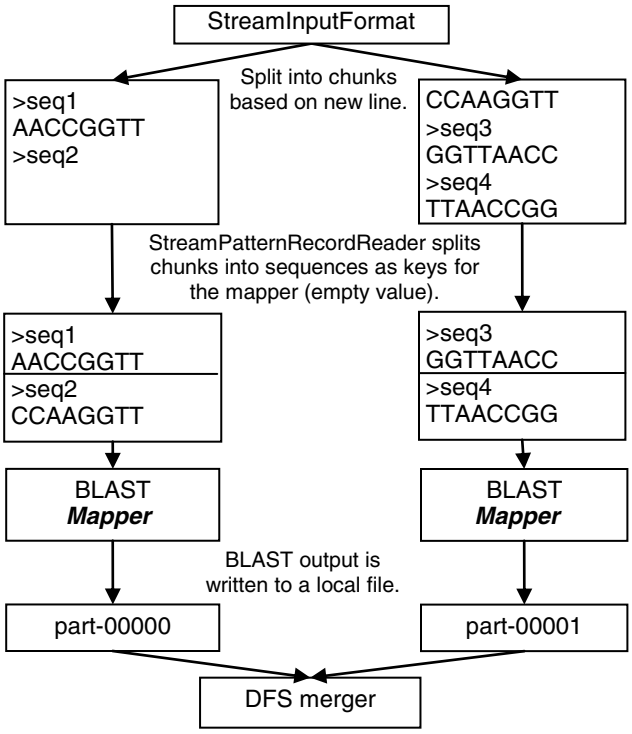

Figure 2. BLASTing with MapReduce. Given a set of input sequences, Hadoop splits the input file into chunks of same size, possibly splitting in the middle of a sequence. StreamPatternRecordReader was developed to interpret correctly the boundaries of sequences, which are passed as keys of a key/value pair to the Mapper that executes BLAST. DFS retrieves all local file outputs combining into a single result.

operating systems and additional software (e.g. to support a particular type of network protocol or technology). Similarly, a Hadoop-based approach requires machines with the Hadoop runtime and file system software, which can only execute if one of several acceptable version of Java is also installed. Both approaches also require ssh software for starting the distributed runtime environment. In the approach envisioned by this paper, machine virtualization is used to enable the use of VMs that already encapsulate these environments, effectively becoming appliances for execution of bioinformatics applications.

The long-term viability and desirability of the proposed approach as a sustainable and effective solution stems from the following three facts:

1. VM solutions for PC servers have matured in recent years [27] with many open source and commercial offerings (e.g., Citrix Xen, VMware, Microsoft, and KVM). Commercially available Grid infrastructures using VMs include 3tera [28], Amazon Elastic Computing Cloud (EC2) [29], Engine yard [30], GoGrid [31], and Joyent [32]. Examples of open-source solutions are virtual workspaces [11], Eucalyptus [12] and OpenNebula [13].

2. For resource providers, VM technologies offer easier management with better control of the shared resources and better isolation among users. From the perspective of clients, the ability to run the operating system of choice on the Grid makes it possible to homogenize, at least in terms of execution environment, the heterogeneous set of available resources. It is no longer necessary to prepare different application binaries to run in different execution environments.

3. Applications can be encapsulated into VMs and, since VM images are used as templates to deploy multiple copies when needed, only the template images need to be maintained. When application updates are necessary, a new template image is created and multiple copies are automatically distributed by the Grid services [11]. This relieves users and/or administrators, potentially in multiple sites, from application installation, maintenance and deployment tasks. Application users are freed from the need to deal with application installation, compilation and tests. Templates prepared by specialists can be used, and biologists can concentrate on providing input data and interpreting the results, as exemplified by the mpiBLAST appliance available for Amazon EC2 [33].

C. *Biological Application Database*

Bioinformatics applications make use of biological data sets of ever increasing sizes, as shown by statistics from the protein sequence UniProtKB/TrEMBL database [34] (Fig. 3). Periodical updates of these databases are necessary to offer up-to-date results to biologists. Propagation of these updates is not an easy task when dealing with thousands of resources.

Data management techniques in Grid computing (e.g., GVFS [35], SRB [36]) could potentially be used to automate the distribution of updated data to VMs on the cloud. However, in general, bioinformatics applications are not designed and implemented to take advantage of Grid data management services. In addition, best performance is obtained when datasets are as close as possible to compute nodes, ideally in the local storage. Hadoop DFS takes advantage of the local storage by replicating input and output data.

On a VM-powered distributed environment (as in this work), the proposed approach is to encapsulate biological data into VM images or into independent virtual disk images. Replications of disk images are offered as services in cloud infrastructures, facilitating data maintenance as only

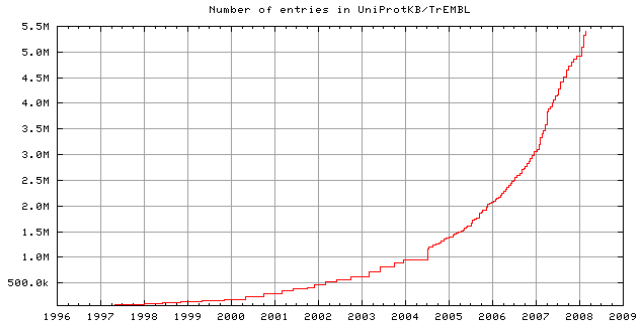

Figure 3. Growth of protein sequences in UniProtKB/TrEMBL database. Release 38.2 (08-Apr-2008): it contains 5577054 sequence entries comprising 1803615193 amino acids. 216176 sequences have been added since release 38, the sequence data of 422 existing entries has been updated and the annotations of 1612992 entries have been revised. [34]

a reduced number of images require manual update interventions and data replication is inherited from disk replication.

D. Parallel Applications Require Network Connectivity

Independently of the parallelization solution chosen, distributed applications require all computing nodes to be reachable with static IPs [37]. Having all-to-all communication among nodes is particularly challenging in multi-site multi-domain deployments due to the presence of NATs and firewalls.

To support distributed applications running in a large number of nodes spread across the Internet, different overlay networks architectures have been proposed, including NAT-aware network libraries and application programming interfaces (APIs) [9][38][39][40][41], virtual networks (VNs) [6][7][8], and peer-to-peer (P2P) systems [10][42][43].

In this work, the ViNe approach [6], developed by the authors, is proposed to provide the necessary connectivity among VMs deployed in different sites (Fig. 4). ViNe is a good solution because it does not interfere with intra-site communication – full LAN performance is available to the nodes on the same LAN. In addition, ViNe imposes low overhead on cross-site communication. ViNe routers are typically deployed at each site, and responsible for intercepting packets that cross LAN boundaries. Intercepted packets are transported to the destination (in private networks) using firewall and NAT traversal techniques.

One important advantage of ViNe over other solutions is its performance. For example, for the experimental setup described in the next section, ViNe routes packets at rates over 800Mbps, which is much higher than the available bandwidth (200Mbps). In terms of latency, less than 0.5 ms round-trip overhead on inter-site communication was observed (over the 28 ms physical network performance). As a result, the approximate packet encapsulation overhead was only 5%.

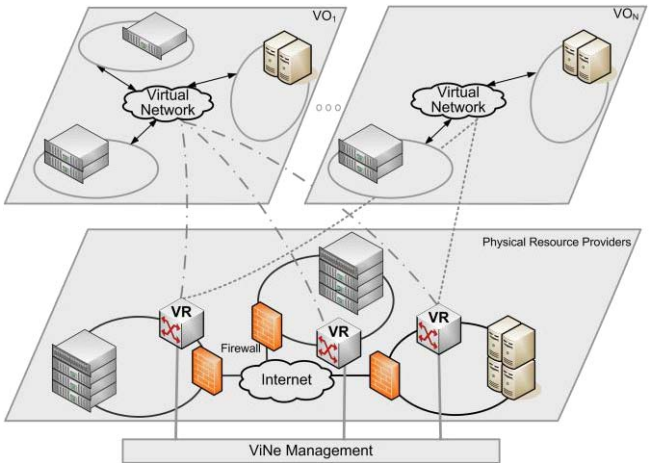

Figure 4. Virtual Network (ViNe) architecture for Grid computing. Multiple independent virtual networks (VNs) are overlaid on top of the Internet. ViNe routers (VRs) control VN traffic. Management of VNs is accomplished by dynamically reconfiguring VRs.

III. Experimental Setup

To validate and evaluate the performance of the proposed approach, a distributed environment consisting of Xen-based VMs was deployed on resources at the UC [7] and at the UF [44], interconnected through ViNe. The specifications of the Xen VMs deployed in each site are shown in TABLE I.

TABLE I. Specification of VMs at Each Site.

	UC	UF
Xen VMM	3.1.0	3.1.0
Guest kernel	2.6.18-x86_64	2.6.18-i686
CPU architecture	AMD Opteron 248	Intel Xeon Prestonia
CPU clock	2.2 GHz	2.4 GHz
CPU cache	1 MB	512 kB
VCPUs	2	2
Memory	3.5 GB	2.5 GB

Half of the needed VMs for each experiment are created in each site (except for the 64 processor or VCPU case, when 12 nodes at UC and 24 at UF were used).

Gigabit Ethernet provides intra-site connectivity in both UC and UF clusters. Xen VMs were deployed in a private network without global addressability. As previously described, ViNe was used to provide the necessary connectivity.

As a representative bioinformatics application, BLAST was chosen, as both sequential [16] and MPI-based parallel [45] versions are available. The MPI version uses the approach that segments not only the input sequences, but also the database to be more efficient when the entire database does not fit into memory. The CloudBLAST approach segmenting only the input sequences has been developed using Apache Hadoop as detailed in Section A.

All the necessary binaries (i.e., BLAST 2.2.18, mpiBLAST 1.5.0beta1, MPICH2 1.0.7, Hadoop 0.16.2, Java 1.6.0, and Python 2.5.1) were installed in a Xen guest domain image and uploaded to virtual workspaces image repositories at UC and UF. The non-redundant (NR) protein sequence database from NCBI split into 1 fragment and 30 fragments (total of 3.5 GB of data) was also placed in the VM image.

The process of deploying a virtual BLAST cluster consists of the following steps (Fig. 5):

1. A VM image with the necessary software is prepared and uploaded to the virtual workspaces factory server.
2. A biologist requests replicas of the VM.
3. The virtual workspaces factory server in each site selects available physical nodes and starts the VM cloning process.
4. VM clones are booted and configured with appropriate network addresses. Network routes are adjusted to use VRs to reach nodes crossing site boundaries.

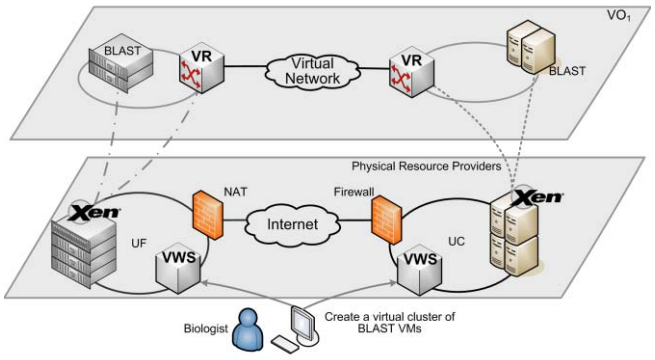

Figure 5. Experimental setup. 24 Xen-based VMs at UF and 12 Xen-based VMs at UC are connected through ViNe to provide a distributed environment to execute Hadoop and MPI-based BLAST applications. A user can request a virtual cluster for BLAST execution by contacting the virtual workspaces server (VWS).

5. A VR VM is cloned and started in each site, offering all-to-all connectivity to all VMs.
6. The biologist logs into one of the VMs, uploads the input sequence and starts executing BLAST jobs.

In the BLAST experiments conducted, 2 sets of nucleotide sequences, each with 960 sequences were used as input. The blastx program from the NCBI BLAST suite, which compares nucleotides sequences against protein sequences by translating the nucleotides into amino acids, was used. Characteristics of the 2 input sets are shown in TABLE II.

TABLE II. EXPERIMENTAL SEQUENCES CHARACTERISTICS.

	960 long	960 short
# of sequences	960	960
Longest sequence	5,813	850
Shortest sequence	215	101
Standard deviation	515	170
Total number of nucleotides	1,072,143	425,735
Average nucleotides per sequence	1116.82	443.47
Sequential execution time	43 hours and 46 minutes	17 hours and 06 minutes

IV. EVALUATION AND EXPERIMENTAL RESULTS

To evaluate the overhead of machine virtualization, experiments with BLAST were conducted on physical machines (pure Linux systems without Xen VMM) and later on the Xen VMs hosted by the same machines, using the physical nodes at UF. Virtualization overhead is barely noticeable when executing BLAST as it can be observed in Fig. 6. This is due to the minimal need for I/O operations as a result of replicating the database in each node and fully allocating it to main memory. It also confirms other observations of virtualization having negligible impact on compute-dominated applications or even improving performance due to artifacts of implementation, e.g. file system double caching [46].

To evaluate the scalability of mpiBLAST and CloudBLAST, experiments with different numbers of nodes, processors, database fragments and input data sets were executed. Exactly 2 processors are used in each node. Whenever possible, the numbers of nodes in each site are kept balanced (i.e., for experiments with 8 processors, 2 nodes in each site are used; for experiments with 16 processors, 4 nodes are used in each site, and so on). For experiments with 64 processors, 12 nodes at UC and 20 at UF were used due to limited number of available nodes at UC.

Results are summarized in Fig. 7. For all input data sets, CloudBLAST shows a small advantage over mpiBLAST. For input data set of long sequences a 57-fold speedup can be observed with CloudBLAST when using 64 processors on 2 sites, while mpiBLAST delivers 52-fold speedup. The sequential execution time for the long input set of 43.76 hours on a single UF machine is reduced to 46 minutes on the cloud with CloudBLAST and to 50 minutes with mpiBLAST. Similarly, for the short input set, 17.1 hours of sequential execution is reduced to 19.5 minutes with CloudBLAST and 25 minutes with mpiBLAST. The performance difference between CloudBLAST and mpiBLAST is more noticeable when sequences are short, indicating an efficient implementation of input sequence split and job coordination by Hadoop. Two-site results show better speedup than one-site due to processors that are fast at UC (approximately 1.2 times faster). The publicly available mpiBLAST can be further hand-tuned for higher performance, but such optimized implementations are not publicly accessible [47].

Although the entire target database fits into memory in the setup used, runs with the database segmented into 30 fragments were performed in order to increase the amount of network communication for mpiBLAST runs (Fig. 8). Speedup graphs (Fig. 7) show that there was no impact on the performance due to VN.

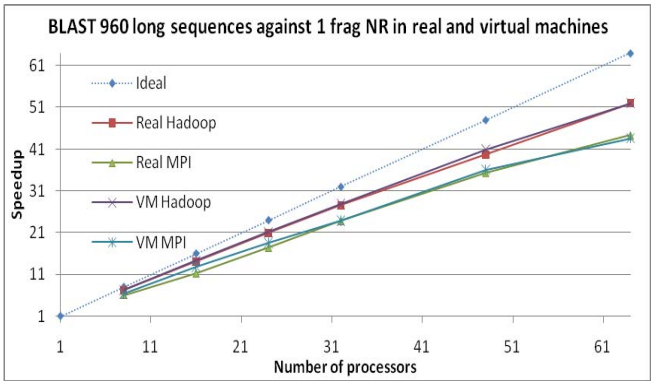

Figure 6. Comparison of BLAST speed-ups on physical and virtual machines. Speedup is calculated relative to sequential BLAST on virtual resource. VMs deliver performance comparable with physical machines when executing BLAST.

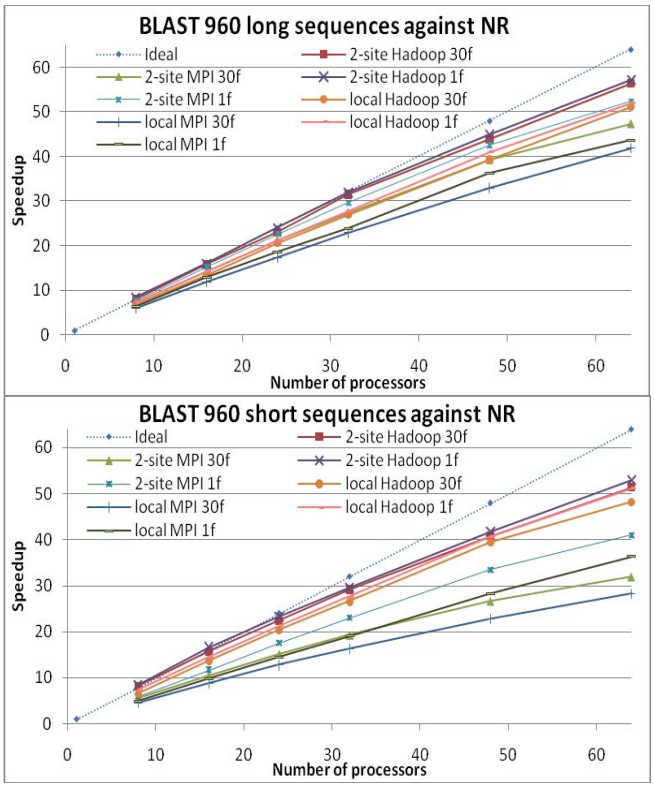

Figure 7. Speedup curves for CloudBLAST (Hadoop) and mpiBLAST: 960 short and long sequences against NR database segmented into 1 and 30 fragments on 1-site and 2-site resources. CloudBLAST presents a slightly better performance in particular when sequences are shorter.

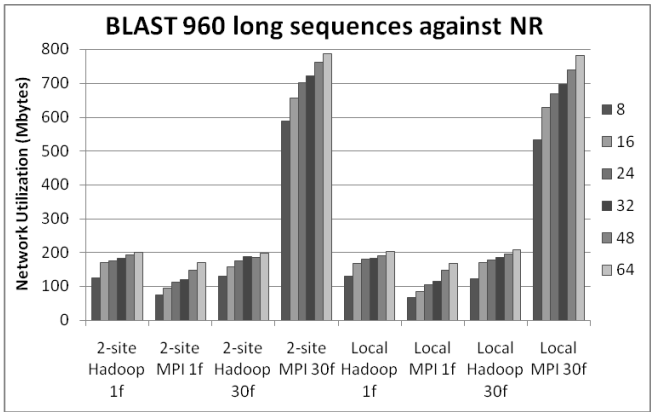

Figure 8. Network usage for CloudBLAST (Hadoop) and mpiBLAST: 960 long sequences against NR database segmented into 1 and 30 fragments on 1-site and 2-site resources.

V. CONCLUSIONS

The research reported in this paper investigated an efficient approach to the execution of bioinformatics applications and validated it by demonstrating its low overheads and high performance by developing CloudBLAST, a distributed implementation of NCBI BLAST. The crux of the approach is the integration of the MapReduce approach to the parallel execution of applications with the encapsulation of software environments and data in virtual machines connected by virtual networks. The experiments were all conducted in a real deployment, with Internet-connected resources across the University of Chicago and the University of Florida. When compared with a leading parallel version of BLAST (mpiBLAST), CloudBLAST exhibited better performance than the publicly available version of mpiBLAST. CloudBLAST has also advantages in terms of simpler development, management and of sustainability since it easily accommodates releases of new versions of the sequential NCBI BLAST. Surprisingly, both mpiBLAST and CloudBLAST showed performance gains with increases in the number of available processors. No noticeable difference in execution time was observed when allocating all resources from one site versus allocating half of the resources from each site, even in small scales (e.g., 4 processors). All software components used in this work are open-source and readily available from the respective project sites.

At a more general level, the three main conclusions from the work reported in this paper are as follows:

1. For applications whose dependency structure fits the MapReduce paradigm, the CloudBLAST case study suggests that few (if any) performance gains would result from using a different approach that requires reprogramming. Conversely, a MapReduce implementation such as Hadoop brings with it significant advantages from the perspective of management of failures, data and jobs. This conclusion reinforces similar claims of proponents of the MapReduce approach and demonstrates them in the context of bioinformatics applications.

2. Using virtual machines with software and data needed for execution of both the application and MapReduce greatly facilitates the distributed deployment of sequential codes. The middleware used for creation, cloning and management of virtual machine images can be presented to users as a service that transparently provides environments with enough resources for the size of the problem to be solved.

3. The use of virtual networks is essential when VM providers belong to different administrative domains behind NATs/firewalls. They must also have good performance in order to take advantage of physical speeds of both local and global network connections. The use of the VN could be transparent to the user when providers use it to connect machines in different facilities. Alternatively, users could use the VN to connect resources leased from different providers. In both cases, VNs greatly facilitate the scaling out of applications which, like those found in bioinformatics, use increasingly larger data sets.

ACKNOWLEDGMENTS

This work is supported in part by NSF grants No. OCI-0721867, CNS-0821622 and the BellSouth Foundation. Parts of this work reflect preliminary outcomes of

collaborative research between UC and UF to investigate the integration of virtual network services and virtual workspace services [11]. The authors are grateful to Dr. Kate Keahey and Tim Freeman for providing access to resources and virtual workspaces at UC as well as support during experiments. Aplysia sequences used in the experiments were provided by Dr. Leonid Moroz. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF or BellSouth Foundation.

REFERENCES

[1] I. Foster and C. Kesselman, The Grid 2: Blueprint for a New Computing Infrastructure, Morgan Kaufmann Publishers Inc, 2003.

[2] Globus Toolkit. http://www.globus.org/toolkit/

[3] Open Middleware Infrastructure Institute. http://www.omii.ac.uk/

[4] Condor Project. http://www.cs.wisc.edu/condor/

[5] B. Gleeson, A. Lin, J. Heinanen, G. Armitage and A. Malis, "A framework for IP-based virtual private networks," RFC2764, Feb. 2000.

[6] M. Tsugawa and J. Fortes, "A Virtual Network (ViNe) Architecture for Grid Computing," Proc. of the 20th Intl. Parallel and Distributed Processing Symp, April 2006, doi:10.1109/IPDPS.2006.1639380.

[7] A. Sundararaj and P. Dinda, "Towards Virtual Networks for Virtual Machine Grid Computing," Proc. of the 3rd USENIX Virtual Machine Research and Technology Symp., May 2004.

[8] D. Joseph, et. al., "OCALA: An Architecture for Supporting Legacy Applications Over Overlays," Proc. of the 3rd Symp. on Networked Systems Design & Implementation, May 2006.

[9] J. Maassen, and H. E. Bal, "SmartSockets: Solving the Connectivity Problems in Grid Computing," Proc. of the 16th Intl. Symp. on HPDC, June 2007, doi:10.1145/1272366.1272368.

[10] L. Gong, Project JXTA: A Technology Overview, Tech. Report, Sun Microsystems, Oct. 2002.

[11] Virtual Workspaces, http://workspace.globus.org/clouds/

[12] Eucalyptus. http://eucalyptus.cs.ucsb.edu

[13] OpenNebula. http://opennebula.org

[14] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," Proc. of the 6th Symp. on Operating Systems Design & Implementation, 2004, pp.137-150.

[15] Apache Hadoop project. http://hadoop.apache.org/

[16] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman., "Basic Local Alignment Search Tool", Journal of Molecular Biology, 1990, v. 215(3), pp.403-410, doi:10.1006/jmbi.1990.9999.

[17] Ian Foster, "There's Grid in them thar Clouds," personal blog, http://ianfoster.typepad.com/blog/2008/01/theres-grid-in.html.

[18] Marc-Elian Bégin, "An EGEE Comparative Study: Grids and Clouds - Evolution or Revolution?," 23rd Open Grid Forum (OGF23), June 2008.

[19] R. Buyya, C. S. Yeo, and S. Venugopal, "Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities," Proc. of the 10th IEEE Intl. Conference on HPCC, September 2008, pp.5-13, doi:10.1109/HPCC.2008.172.

[20] myGrid. http://www.mygrid.org.uk

[21] Arun Krishnan, "GridBLAST: a Globus-based high-throughput implementation of BLAST in a Grid computing framework," Concurrency and Computation, v.17, Issue 13, pp. 1607-1623, 2005, doi:10.1002/cpe.v17:13.

[22] H. Stockinger, M. Pagni, L. Cerutti, L. Falquet, "Grid Approach to Embarrassingly Parallel CPU-Intensive Bioinformatics Problems," Proc of the 2nd IEEE Intl. Conf. on e-Science and Grid Computing, 2006, doi:10.1109/E-SCIENCE.2006.70.

[23] J. Andrade, M. Andersen, L. Berglund, and J. Odeberg, "Applications of Grid Computing in Genetics and Proteomics," LNCS, 2007, doi:10.1007/978-3-540-75755-9.

[24] P. Balaji, et. al., "Distributed I/O with ParaMEDIC: Experiences with a Worldwide Supercomputer," Proceedings of the IEEE International Supercomputing Conference, June 2008.

[25] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison, Biological sequence analysis: probabilistic models of proteins and nucleic acids, Cambridge University Press, 1998.

[26] Z. Zhang, S. Schwartz, L. Wagner, and W. Miller, "A greedy algorithm for aligning DNA sequences", J Comput Biol; 7(1-2):203-214, 2000, doi:10.1089/10665270050081478.

[27] J. E. Smith, R. Nair, Virtual Machine Versatile Platforms for Systems and Processes, Morgan Kaufmann Publishers, June 2005.

[28] 3tera, http://www.3tera.com/

[29] Amazon Elastic Compute Cloud (Amazon EC2), http://aws.amazon.com/ec2.

[30] Engine yard, http://engineyard.com/

[31] GoGrid, http://www.gogrid.com/

[32] Joyent, http://joyent.com/

[33] mpiBlast in Amazon EC2. http://mpiblast.pbwiki.com/AmazonEC2

[34] EMBL-EBI UniProtKB/TrEMBL database. http://www.ebi.ac.uk/trembl/

[35] M. Zhao, V. Chadha, and R.J. Figueiredo, "Supporting Application-Tailored Grid File System Sessions with WSRF-Based Services," Proc of the 14th IEEE Intl. Symp. on HPDC, 2005, pp.24-33, doi:10.1109/HPDC.2005.1520930.

[36] C. Baru, R. Moore, A. Rajasekar, M. Wan, "The SDSC Storage Resource Broker," Proc.of CASCON'98 Conference, 1998.

[37] M. Tatezono, N. Maruyama, and S. Matsuoka, "Making Wide-Area. Multi-Site MPI Feasible Using Xen VM," LNCS, 2006, doi:10.1007/11942634.

[38] A. Denis, et. al., "Wide-Area Communication for Grids: An Integrated Solution to Connectivity, Performance and Security Problems," Proc. of the 13th IEEE Intl. Symp. on HPDC, Jun. 2004, doi:10.1109/HPDC.2004.1323501.

[39] D. Caromel, C. Delbé, A. di Costanzo, and M. Leyton, "ProActive: an Integrated Platform for Programming and Running Applications on Grids and P2P systems," Computational Methods in Science and Technology, vol. 12(1), 2006.

[40] S. Son and M. Livny, "Recovering Internet Symmetry in Distributed Computing," Proc. of the 3rd Intl. Symp. on Cluster Computing and the Grid, May 2003, doi:10.1109/CCGRID.2003.1199412.

[41] M. Matsuda, T. Kudoh, Y. Kodama, R. Takano, "Efficient MPI Collective Operations for Clusters in Long-and-Fast Networks,". 2006 IEEE Intl. Conference on Cluster Computing, 2006, doi:10.1109/CLUSTR.2006.311848.

[42] A. Ganguly, A. Agrawal, P. O. Boykin, and R. Figueiredo, "WOW: Self-Organizing Wide Area Overlay Networks of Virtual Workstations," Proc. of the 15th Intl. Symp. on HPDC, June 2006, doi:10.1109/HPDC.2006.1652133.

[43] Bram Cohen, "Incentives build robustness in BitTorrent," First Workshop on the Economics of Peer-to-Peer Systems, Berkeley, CA, June 2003.

[44] ACIS cloud, http://www.acis.ufl.edu/vms

[45] A. Darling, L. Carey, and W. Feng, "The Design, Implementation, and Evaluation of mpiBLAST," Proc. of the 4th Intl. Conf. on Linux Clusters, 2003, 14p.

[46] Kate Keahey, personal communication on 03/2006.

[47] Wu Feng, personal communication on 07/2008.