VOL. 11, NO. 7, JULY 2016 ISSN 1990-6145
ARPN Journal of Agricultural and Biological Sciences

www.arpnjournals.com

CLOUD COMPUTING FRAMEWORK FOR RESOURCE MANAGEMENT SYSTEM: A CASE STUDY OF CROSS-SPECIES SEQUENCE COMPARISONS

Saeed Ullah, M. Daud Awan and M. Sikandar Hayat Khayal
Faculty of Computer Science, Preston University, Islamabad, Pakistan
E-Mail: saeedullah@gmail.com

ABSTRACT

Knowledge about the sequence of genes in different species at varying evolutionary distances helps to identify coding and functional non-coding sequences among organisms as well as sequences that are unique for a given organism. Cross-species sequence comparison is a useful technique to understand genome, both in similarity and differences, to study changes in human genomes in different diseases. The research in genetic similarity is advancing our understanding of muscle and organ development. Automatic and manual annotation for more than 26,000 protein-coding genes have showed 70% of identical similarity with human protein genes to zebrafish protein genes while 84% genes associated with human diseases have a counterpart in zebrafish. With the introduction of cloud computing environment in scientific applications, single instance based BLAST cloud AMIs have been introduced to speed up the computation, required for cross-species sequence comparisons to better understand the genomes, both in similarity and differences. One of the limitations of these standalone instances is that these instances operate in isolation and do not distribute sequence searches across multiple instances. We propose a cloud computing framework for resource management system to speed-up the computation process. Experimentation results show that our proposed framework achieved the objective by significantly reducing execution time.

Keywords: cloud computing, bioinformatics, cross-species sequence comparisons, BLAST+

INTRODUCTION

Bioinformatics is an emerging interdisciplinary field of scientific research where the statistical, mathematical, engineering and computer science models are used to analyze biological data. Genetic similarities and differences can be compared using different bioinformatics methods. One of the common use of bioinformatics techniques is to identify candidate genes and nucleotides to better understand the genetic basis of disease, desirable properties and nucleic acid and protein sequences [1]. DNA sequencing is made with aim of determining the precise order of nucleotides within a DNA molecule. Each DNA strand is composed of four bases: adenine, guanine, cytosine, and thymine. To determine the precise order of four bases within DNA, different methods and tools can be used to identify precise location and sequences of genomes of numerous types and species of life. DNA sequencing may be used to determine the sequence of individual genes, larger genetic regions such as clusters of genes or operons, full chromosomes or entire genomes. This information is useful to various fields of biology and other sciences, medicine, forensics, and other areas of study. DNA sequencing techniques have been applied in numerous fields including molecular biology, evolutionary biology, meta-genomics, medicine and forensics [2].

Knowledge about the sequence of genes in different species at varying evolutionary distances helps to identify coding and functional non-coding sequences among organisms as well as sequences that are unique for a given organism. For instance, many of the homo-sapiens (Human) genes are identical or similar to those found in other species. The comparison process requires at least two DNA sequences, related by convergent evolution or divergent evolution from a common ancestor to determine the level of similarity of two sequences.

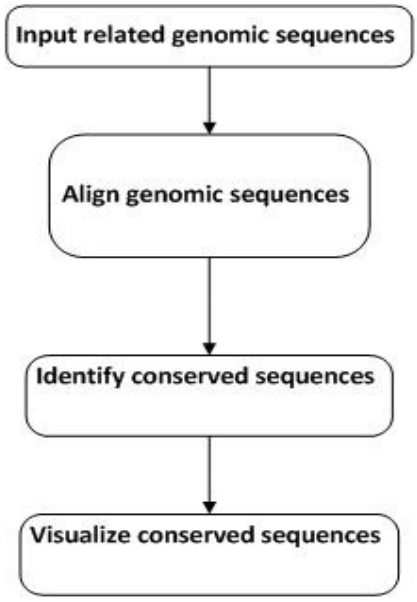

Figure-1. Cross-species sequence analysis process, adapted from [3], [4].

Hence, cross-species sequence comparisons is a useful technique to understand genome, both in similarity

and differences, to study changes in human genomes in different diseases [3], [4].

DNA sequencing is an essential resource to study health and genetic diseases in human being. Two large genomes that have been sequenced with the human reference genome are zebrafish and mouse genome. Zebrafish are biologically much similar to human genes which make them a reference study model to understand genes role in different human disease. According to a scientific study, automatic and manual annotation for more than 26,000 protein-coding genes showed 70% of identical similarity with human protein genes to zebrafish protein genes while 84% genes associated with human diseases have a counterpart in zebrafish [5].

Different studies have revealed that zebrafish may spontaneously develop any type of tumor known from human with similar morphology [6]. The research in genetic similarity is advancing our understanding of muscle and organ development. Zebrafish genes have been used to assert the evolution and formation of causal gene in muscular dystrophy disorders. These are also helpful in for modelling paediatric diseases in 80% of cases by altering the activity of genes in zebrafish embryos [7]. However, the process will take several years to read through the entire human genome to locate the sequence of genes in other species which is impractical. Hence, sophisticated tools and technological methods are required to carry out this process.

A useful bioinformatics tool in this area of research is BLAST+ (Basic Local Alignment Search Tool) [8], to search entire genomic libraries for identical or similar sequences. The software can be used as either a command line standalone version for local alignment search or by using the online version of the tool. Considering the nature of workload for two genomes with a large sequences, the search process may be much computationally expensive. For instance, On a Core i-5 processor with 4 GB RAM, the similarity search for two small sized genomes, zebrafish.1.protein.faa and mouse.1.protein.faa, may take computational time up to 15 hours. This computational delay is not affordable in bioinformatics and genetic engineering applications where organism's genomes are continuously manipulated. With the introduction of cloud computing environment in scientific applications, single instance based BLAST cloud AMI has been introduced by three cloud providers: Amazon Web Services (AWS), Google Compute Engine (GCE), and Microsoft Azure. The purpose of this case study was to distribute sequence searches across multiple instances using our designed framework, which is not currently supported in this environment.

CLOUD COMPUTING MODEL

From the last three decades, the trend of personalized, powerful and to-do-yourself personal computers has taken place over the use of large room based, centralized mainframe systems in the business, research and scientific communities. Earlier in computing history, when some complex and long computations has to be performed, services with their complex systems have to be bought by the company or personnel to perform necessary computation and calculations; that is now-a-days is just a matter of the off-the-shelf software installed in user devices. Thus, the idea of buying services in order to perform the tasks associated with the routine of the organizations, business or community emerged and this trend evolved to cloud computing [9].

Just like business organizations, scientific community is also getting benefits from cloud capacities by adopting the utility of cloud computing offered with low cost associated. As more resources are required for high performance computing (HPC) applications to accomplish their workload, the advent of cloud computing has shifted their computation from the dedicated clusters to the widely available cloud provided utilities in a pay as you go fashion.

PROPOSED FRAMEWORK

The proposed cross-layer design consists of three modules; each can directly communicate with other, whenever required. Job Management Module is responsible for job admission, negotiation and provisioning of resources based on the feedback from ranking module. When the negotiation is successfully completed, service level agreement is established between consumer and producer which specifies the service level objectives based on QoS. SLA management module is responsible for monitoring the violations of SLA. In case of violation, alarm notifier will trigger an alarm and the event will be stored in violation data repository. Based on this violation data repository and cloud provider (CP) catalog, which stores the promised QoS by each provider, QoS rank of each CP is calculated by the ranking module. The proposed framework is given in Figure-2.

Job management module

This module is responsible for managing the requests from the consumer of cloud federation along with their SLAs and negotiation contracts to interact with the match maker module in order to invoke resource provisioning mechanism between the consumer and provider which suits best to the requirements specified in form of SLA. Four further subcomponents in this module are:

Negotiation engine

This is an important component of the Job management module. With the negotiation terms, the strategies that need to be adopted in form of penalties in case of SLA violations from the cloud provider are mostly referred [10]. Moreover, this component is solely negotiating the QoS parameters such as availability, bandwidth, storage and so on, that are to be required by the consumer from the provider after the one time negotiation settlement with the provider along with their priorities. This also covers the cost factors for the

resources that will be utilized once the settlements have been made. The contents of negotiation terms are exchanged in XML schema of SLA.

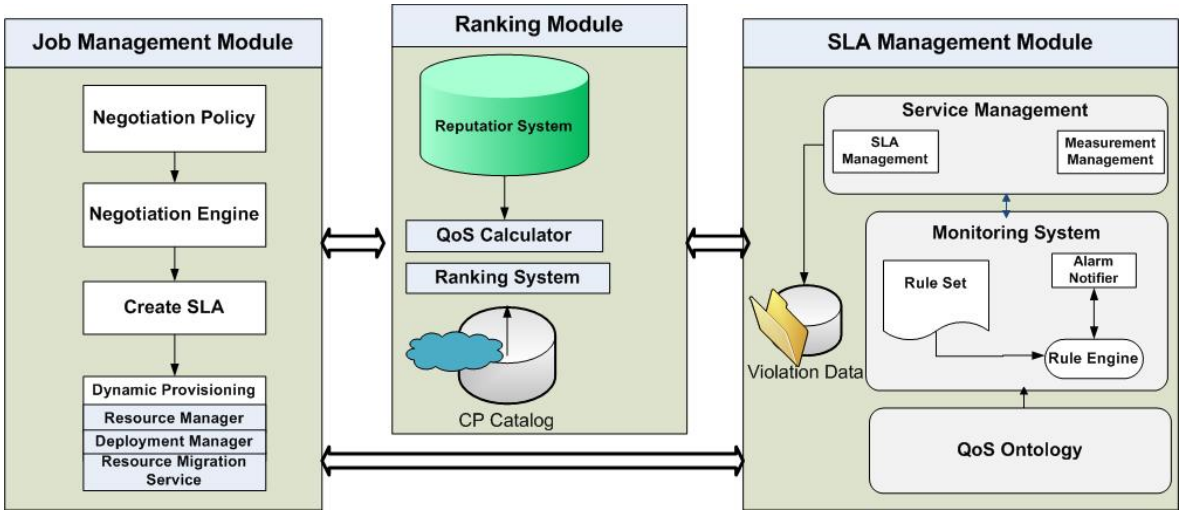

Figure-2. Proposed broker based cloud computing framework for resource management system.

Resource manager

This module ensures the provisioning of resources by the provider to its consumer once the request is initiated by the job management module taking into account the negotiation terms and settlements. The degree of provisioning SLA assurance however is evaluated by the SLA manager.

Deployment manager

It is responsible for creation of the virtual machine pool according to actual state information of the user job. Based on the QoS requirements, the module aims to complete job within budget and deadline constraints.

Resource migration service

In case of early termination or service interruption due to low QoS, provisioning of resources is re-evaluated to meet QoS and deadline constraints. Necessary measures would be taken to migrate the VM resources to the subsequent best matched cloud provider.

SLA management module

Service level agreement is defined as: "a contract between a cloud provider and a cloud consumer, with details related to the provisioned resources along with the price charged, and the associated quality attributes while ensuring the use of provisioned resources provided by the CP" [11]. SLAs are composed of service level objectives that define quality attributes such as availability as well as associated penalty in case of SLA violations. SLAs are used as a driving force for negotiation between different cloud providers. In our earlier study, SLAs of different cloud providers were benchmarked for their QoS aspects and based on the user requirements, the candidate best fit cloud providers were selected, depending upon the specific QoS requirements [12]. SLAng [13], a XML based language for specification of QoS aspects of service level agreements, was extended to provide a negotiation mechanism between different cloud providers.

SLA service management

This module checks individual resource providers to confirm the service requirements based on QoS requirements. The proposed schema is a group of SLOs that is derived for network, storage, memory and computing quality attributes. The first set of service level objectives defined in schema are related to the computing capacity of the resources followed by storage, network and memory. Attributes that are associated with the computing requirements are CPU cores, VM speed, availability, OS type, cost and boot time of VM instances. Quality metrics like Input/ Output operations per second and latency are also taken in account while representing the storage set of terms. Upload/ download speed and ping are other important attributes while representing the network set of requirements in SLA. Provider's location attribute is given in the schema in order to represent the geographical boundary of a particular datacenter so that resource outsourcing may be achieved with minimal latency using the most nearest data centers.

Resource monitoring module

This module continuously monitors secured resources against service abruption and violation of SLA. Taking into consideration the existing studies on SLA violation [14], [15], the flow of events is as under:The module is

invoked when the agreement is signed between home and foreign cloud providers on agreed upon QoS terms. A small monitoring component is installed at the foreign cloud provider end to collect necessary statistical usage and evaluation data.

1. Monitoring component periodically updates the broker about usage data
2. If case of any SLA violation, an alarm is triggered which is handled by alarm manager. The details of violation is recorded and compared against allowed threshold given in rule set. In case of severe violation, resource switchover takes place and the VM is migrated to the candidate cloud provider.

RESULTS AND DISCUSSION

For the experimentation purposes, VM instances of 64-bit Amazon m3.xlarge BLAST AMI hosted on Ubuntu 12.04 were used as a reference cloud provider VMs. BLAST+ server images for Google Compute Engine were leased, when required. Since dedicated instances for BLAST+ package are not installed on Rackspace servers, these instances were prely installed and configured with necessary BLAST+ packages and tools.

Each search query consisted of hundreds of FASTA sequences of zebrafish and mouse proteins that were compared with human protein structure. These query files were hosted on a FTP server and were parsed by GNU parallel [16] to create a parallel pipeline based on the number of available virtual machines. A simple FASTA file with two sequences is presented in the Figure-3.

>gi|51467976|ref|NP_001003855.1| alpha-(1,6)-fucosyltransferase [Danio rerio]

MRPWTGSWRWIALVLLAWGTLLFYIGGHLVKDSEHAPRSSRELAKILTKLERLKQQNEDLRR
MAQSLRIPEGQSDGPISS

GRLRSLEEQLSRAKQKIQSFQRLSGEGPGKDHEELRRKVENGVRELWYFVRSEVKKLPLMET
GAMHKHVDTLMQDLGHQQ

RSVMTDLYHLSQADGAGDWREKEANELSDLVQNRIMYLQNPQDCSKARKLVCNINKGCGY
GCQLHHVVYCFMIAYGTQRT

LILESQNWRYATGGWETVFKPVSDTCTDRTGASTGHWSGEAHDRDVQVVELPIVDSLHPRPP
YLPLAVPEDLAPRLQRLH

GDPSVWWVSQFVKFLVRPQAWLEKEIQETCLKLGFKHPIIGVHVRRTDKVGTEAAFHPIEEY
MVHVEDHYQSLAQRMHVD

KKRVYLATDDPSLLQEAKTKYPDYEFISDNSISWSAGLHNRYTENSLRGVILDIHFLSRTNYLV
CTFSSQVCRVAYEIMQ

TLHPDASSYFYSLDDIYYFGGQNAHNQIAIYPHQPRNSDDIPLEPGDVIGVAGNHWDGYSKGI
NRKTGRTGLYPSYKVKE

KIETVKYPTYPEADKLLKKP

>gi|130487273|ref|NP_001076269.1| leucine-rich repeat-containing protein 30 [Danio rerio]

MCSKLEVLSLANNHLTGLPASLSALVGLKKLNLSHNNITHIPGCVYTMRNLVFLQLACNNLE
NIADQIQALTDLKILIVE

GNCIHSLPKMLCCLTKLELLNVDFNDIQNVPAEMHKLKRLEKLACHPLDKGLHIMHNPLLKPI
KEVLDGGLQALYCYLKAT

Figure-3. FASTA file with two sequences.

In Figure-3, first FASTA query of 580 characters represents amino acid molecule type of alpha-(1,6)-fucosyltransferase. The best identical match of this sequence with other species is Cyprinus carpio fish with 100% similarity value. However, the FASTA sequence of zebrafish.1.protein with human.1.protein.faa query returned very interesting pattern, as shown in Figure-4.

```
# BLASTP 2.3.0+
# Query: gi|578845957|ref|XP_006726675.1| PREDICTED:
uncharacterized protein C16orf52 homolog B [Homo sapiens]
# Database: zebrafish.1.protein.faa
# Fields: query id, subject id, % identity, alignment length,
mismatches, gap opens, q. start, q. end, s. start, s. end,
evalue, bit score
# 2 hits found
gi|578845957|ref|XP_006726675.1|gi|326666289|ref|XP_003198233.1|
     100.000    167   0     0     1     167   1     167   2.56e-119
     339
gi|578845957|ref|XP_006726675.1|gi|115495237|ref|NP_001070129.1|
     89.820     167   17    0     1     167   1     167   2.22e-109
     313
```

Figure-4. FASTA sequence comparison of zebrafish.1.protein with human.1.protein.faa

Homologs of the C16orf52 gene show that this gene is conserved in chimpanzee, Rhesus monkey, dog, cow, mouse, rat, chicken, zebrafish, fruit fly, mosquito, and frog as well as human. Table 1 shows total number of sequences used during the experimentation. It is worth mentioning that a subset of protein databases was used as complete database search was not feasible at this stage.

Table-1. Number of sequences.

Species	Number of sequences
Human	24, 534
Zebrafish	18,314
Mouse	22, 802

Each experiment consisted of FASTA sequence comparison of the zebrafish and mouse genes against human genes to construct region of similarity in cross species. The experimentation was aimed at reducing overall searching time for query sequences so cost and other factors were ignored. Since FASTA sequences are stored in a text format, the cloud coordinator of reference cloud provider (Amazon in this case) parallelized the process by splitting the search query into a chunk of 200K FASTA file to distribute the workload across multiple VM instances. These query files were then pipelined into parallel to speed up the execution process. Each input job file was assigned to a single core of worker virtual machines.

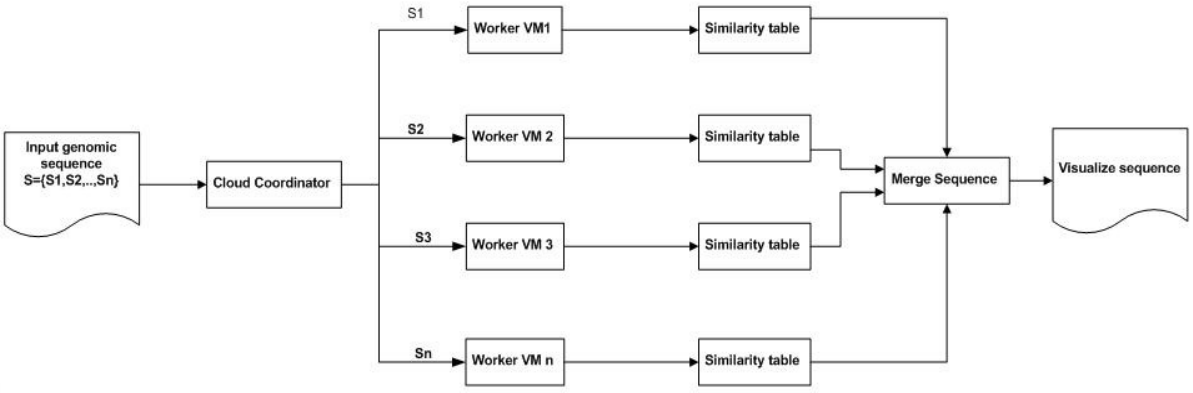

Figure-5. Bio-sequence analysis parallelization pipeline for BLAST.

The SLA, as shown in table 2, was used to enforce service level objectives (SLOs) including availability, incoming bandwidth, outgoing bandwidth, CPU performance and storage.

Table-2. SLA Monitor settings for BLAST+

SLA objective	Value	Threat threshold
Availability	≥99.9%	99.9%
Memory	≥4GB	----
CPU Power	≥92 % (broker unit)	90%
Storage	≥10GB	
Incoming Bandwidth	≥400 Mbit/s	300 Mbit/s
Outgoing Bandwidth	≥100 Mbit/s	70 Mbit/s

SLA settings suggest that a single failure of virtual machine may result in resource switchover as real time applications may not afford resource unavailability during their execution. For the set of experiments, ten local Amazon instances were provided while five instances from other two cloud providers could be leased based upon the workload and job completion deadline. Table 5.19 shows the experimentation results of a single experiment which clearly depicts that total execution time was reduced when more virtual machines were utilized. With one VM instance of Amazon xlarge, the experimentation process was completed in around 8 hours while with 18 virtual machines, the experimentation time was reduced to around 44 minutes. For the comparison purposes, two types of experiments were performed. In the first set of experiments, referred as local cloud completion, local in-house resources were utilized and the experiments were restricted to total number of available BLAST+ instances, which in this case, were a maximum of ten instances. On the contrary, multi-cloud completion time scheme used cloud outsourcing model based on the SLA requirements of the experiments. VM resources, matching SLA requirements, were leased from other two cloud providers and the sequence comparison queries were executed using both in-house and outsourced VMs. Experimentation was repeated for a total of 3 times to calculate average values.

Table-3. Experimentation results of BLAST+ for two policies.

No. of virtual machines			Multi-cloud completion time HH:MM:SS	Local cloud completion time HH:MM:SS
Amazon	Google	Rackspace		
1	0	0	08:14:22	---------same as BOR----
4	0	0	02:57:38	---------same as BOR----
8	0	0	01:31:18	---------same as BOR----
10	1	1	01:06:36	01:18:51
10	4	4	00:44:15	---------same as above----

Results from Table-3 also present an interesting picture of parallelism that can be achieved during such experimentations. The resultant job completion curve tend to decrease at a sharp rate when few virtual machines were used but with the higher increment of virtual machine instances, very small increase was observed in parallelism. This was due to the network overhead which was involved between the worker virtual machines and the cloud coordinator within inter-cloud and intra-cloud for transfer of query and result files. The execution curve is presented in the Figure-4.

The BLAST+ output files were parsed by individual works and final results were stored and synthesized at cloud coordinator end. The more number of hits shown in the results, the more data sequences are aligned. We set the e-value confidence to 1e-50 to get the high quality similarly aligned search sequences. Finally BLAST+ package tools were used to visualize the similarity results which are shown in Figure-5.

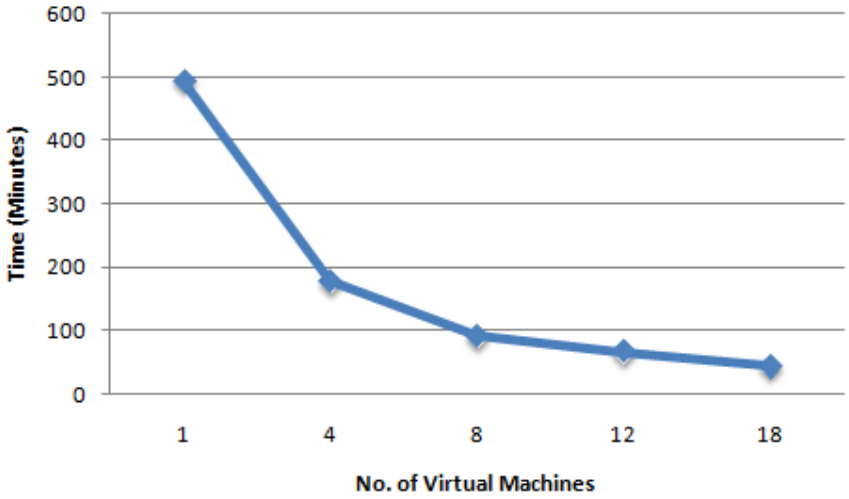

Figure-6. Level of parallelism for different virtual machines.

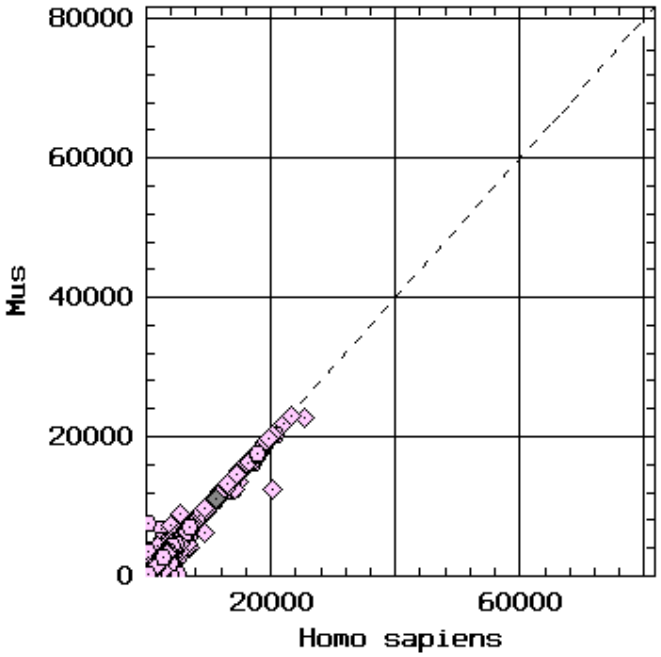

Figure-7. Similarity distribution of Zebrafish with human and mouse.

The comparative protein structure distribution which represents total number of hits, as shown in figure 5, clearly depicts that the three species human, zebrafish and mouse are very much related to each other in protein structure and hence further investigation of similarity can deepen the understanding of muscle and organ development in human being.

RELATED WORK

Several studies have addressed the parallelization issue by deploying cluster based sequential analysis services for scientific applications. mpiBLAST [17] is aimed at improving NCBI BLAST performance by utilizing distributed computational resources through parallel I/O, database fragmentation, query segmentation

and scaling to hundreds of processing elements. In contrast, CloudBLAST [18] integrates MapReduce to model parallelism for the query-segmentation data-parallel pattern. CloudBLAST experiments have been conducted on virtual clusters between two distinct universities where the test database was statically bound with the deployment. The performance comparison results between mpiBLAST and cloudBLAST show that cloudBLAST significantly outperformed mpiBLAST by achieving better performance efficiency. CloudBurst [19] is a parallel sequence alignment model, implemented in Hadoop MapReduce programming model. The algorithm is optimized for mapping many short reads for similarity or differences from sequence matching machines against a reference genome. The sequential analysis results on Amazon EC2 depict near linear speedup as the number of processors increases. The authors in [20] presented a novel approach to combine the dynamic programming with the Hadoop based computational parallelism data grids to speed up the multiple sequence alignment. The proposed method significantly reduced time complexity for very large sequences. Our work differs from the earlier studies as VM instances from multiple cloud providers were utilized to speed up the computational process where the job execution process is not dependent on any specific tool and technology.

CONCLUSIONS AND FUTURE WORK

In this paper, we presented a cloud broker based resource management framework for computationally expensive scientific applications to speed up the execution process. We implemented a multi-cloud environment for parallel execution of BLAST+ sequence alignment tool to reduce overall search and alignment completion process. Application level scalability and elacity was achieved by using our proposed cross-layer design to coordinate and monitor for job demands, requiring high performance throughput. In contrast to other models, such as CloudBLAST or CloudBurst, which are Hadoop MapReduce based alignment models and are technology dependent, our model achieved much better results by using the proposed multi-cloud resource management model.

However, there are several research dimensions that can be improved in the proposed model. Current state of executed tasks are not saved and hence execution is retrieved from the beginning rather than restarted from the last stable checkpoint. Fault tolerance mechanism can be incorporated in the proposed model for better workload and deadline management. The proposed model can be validated on other cloud providers to achieve better efficiency and economics of scale.

REFERENCES

Bioinformatics. Accessed: July 14, 2015, Available from: https://en.wikipedia.org/wiki/Bioinformatics.

DNA sequencing. Accessed: July 18, 2015, Available from: https://en.wikipedia.org/wiki/DNA_sequencing.

K.A. Frazer, L.Elnitski, D.M. Church, I. Dubchak, R.C. Hardison, Cross-species sequence comparisons: a review of methods and available resources. Genome Research. 13(1), pp. 1-12, 2003.

G.S. Gerhard, Comparative aspects of zebrafish (Danio rerio) as a model for aging research. Experimental gerontology. 38(11), pp. 1333-1341, 2003.

Zebrafish Genome Found Strikingly Similar to Humans. Accessed: July 20, 2015, Available from: http://www.sci-news.com/genetics/article01036.html.

H. Feitsma , C. Edwin, Zebrafish as a cancer model. Molecular Cancer Research. 6(5), pp. 685-694, 2008.

K. Howe, M.D. Clark, C.F. Torroja, J. Torrance, C. Berthelot, M. Muffato, S. McLaren, The zebrafish reference genome sequence and its relationship to the human genome. Nature. 496(7446), pp. 498-503, 2013.

BLAST+. Accessed: August 04, 2015, Available from: https://blast.ncbi.nlm.nih.gov/Blast.cgi.

Cloud computing. [cited August 20, 2014], Available from: http://www.explainthatstuff.com/cloud-computing-introduction.html.

F. Faniyi, R. Bahsoon, G. Theodoropoulos. A dynamic data-driven simulation approach for preventing service level agreement violations in cloud federation. in International Conference on Computational Science, ICCS 2012. pp. 1167-1176,2012.

Al-Ghuwairi, Abdel-Rahman, J. Cook, Modeling and Enforcement of Cloud Computing Service Level Agreements. Technical Report, 2012.

S. Ullah, M. Daud Awan, M.S.H. Khayal, A Price-Performance Analysis of EC2, Google Compute and Rackspace Cloud Providers for Scientific Computing. Submitted for publication, 2015.

L.D. Davide, J. Skene, W. Emmerich. SLAng: a language for service level agreements. in The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems. pp. 100-106,2003.

A. Sahai, V. Machiraju, M. Sayal, L.J. Jin, F. Casati. Automated SLA Monitoring for Web Services. in 13th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management: Management Technologies for E-Commerce and E-Business Applications pp. 28-41,2002.

V.C. Emeakaroha, M.A.S. Netto, R.N. Calheiros, I. Brandic, R. Buyya, C.A.F.D. Rose, Towards autonomic detection of SLA violations in cloud infrastructures. Future Generation Computer Systems. 28(7), pp. 1017–1029, 2012.

GNU Parallel. Accessed: August 08, 2015, Available from: http://www.gnu.org/software/parallel/.

A.E. Darling, L. Carey, W.c. Feng. The design, implementation, and evaluation of mpiblast. in Proceedings of ClusterWorld.2003.

A. Matsunaga, M. Tsugawa, J. Fortes. Cloudblast: Combining mapreduce and virtualization on distributed resources for bioinformatics applications. in IEEE International Conference on eScience.2008.

M.C. Schatz, Cloudburst: highly sensitive read mapping with mapreduce. Bioinformatics. 25(11), 2009.

G.S. Sadasivam , G. Baktavatchalam, A novel approach to multiple sequence alignment using hadoop data grids. International Journal of Bioinformatics Research and Applications. 6, 2010.