Accelerating Data-Intensive Applications: A Cloud Computing Approach to Parallel Image Pattern Recognition Tasks

Liangxiu Han, Tantana Saengngam, and Jano van Hemert

UK National e-Science Centre, School of informatics, University of Edinburgh, United Kingdom
liangxiu.han@ed.ac.uk; ttntans@gmail.com; j.vanhemert@ed.ac.uk

Abstract — **Performance is an open issue in data intensive applications, such as image pattern recognition tasks. To process large-scale datasets with high performance more resources and reliable infrastructures are required for spreading the data and running the applications across multiple machines in parallel. The current use of parallelism in high performance computing and with multicore hardware support is costly and time consuming. To remove the burden of building, operating and maintaining expensive physical resources and infrastructures, Cloud computing is emerging as a cost-effective solution to address the increased demand for distributed data, computing resources and services. In this paper, we explore and evaluate parallel processing performance of an image pattern recognition task in the Life Sciences based on a Cloud computing model: Infrastructure-as-a-Service. Namely, we rent computing infrastructures from cloud providers. We have developed the image pattern recognition task in both sequential and parallel ways, deployed them, and conducted our experiments on cloud infrastructure. The performance has been evaluated using speedup as a measurement. We have calculated the cost of our experiments, which demonstrates that cloud computing could be a cheaper alternative to supercomputers and clusters given this task.**

Keywords — Parallel Computing; Cloud Computing; Image Pattern Recognition; Life Sciences; Data intensive application.

I. INTRODUCTION

Advances in storage, pervasive computing, digital sensors, digital libraries and instrumentation have led to a massive growth in the volume of data collected and the number of geographically distributed data sources (e.g., in many fields, biomedical research or image-based diagnosis, data volumes are at least doubling each year). The efficient exploration on these large amounts of data is a critical task to enable scientists to gain new insights. Parallel computing is naturally as a solution to solve this kind of problems by dividing a large problem into smaller ones carrying out much small calculation concurrently. The current parallel processing systems are mainly supported by hardware with multi-core and multi-processors with multiple processing elements within a single machine and/or clusters and grids with multiple machines connected together to work on the same task simultaneously.

To deal with large datasets, more compute resources are required to access large amounts of data and perform many calculations across multiple machines concurrently. However, incorporating data and compute resources into parallel infrastructures (e.g., clusters) amenable to data exploration is not an easy task. One has to take into account scalability, reliability, fault-tolerance and cost-reduction.

To remove the burden of building, operating and maintaining expensive physical resources and infrastructures (e.g., hardware, clusters etc.), cloud computing is emerging as a cost-effective solution to address the increased demand for distributed data, computing resources and services.

Cloud computing [1][2] is a type of distributed computing paradigm augmented with a business model via a Service Level Agreement between providers and consumers. This definition has a two-fold meaning: 1) at a technical level, cloud computing offers distributed resources, infrastructures and services over the Internet to users, which are created, operated and maintained by the cloud providers. The consumers access these resources remotely without running applications on local computers. Cloud computing models are broadly classified into service and delivery models. In terms of services provided by cloud providers, there are three types of service models: Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS) and Infrastructure-as-a-Service (IaaS). Based on the way clouds are delivered, three major types of clouds include public, private and hybrid clouds; 2) at a business level, cloud providers lease these resources immediately and temporarily to cloud consumers when required. Often leases are paid by a regular credit card transaction on a consumption basis.

In this paper, we explore and evaluate the parallel processing performance of a task from the Life Science based on IaaS. Namely, we rent computing infrastructures from a cloud provider (i.e., Amazon Elastic Compute Cloud (Amazon EC2 [3]) with full control of those computing resources. We have developed the task in both sequential and parallel ways and deployed them onto the rented infrastructure. The performance has been evaluated and compared using speedup as a metric.

The rest of paper is structured as follows: Section 2 presents an image pattern recognition use case in the Life Sciences, to which we have applied parallelisation. Section 3 describes the experiments we have conducted. Section 4 concludes the work.

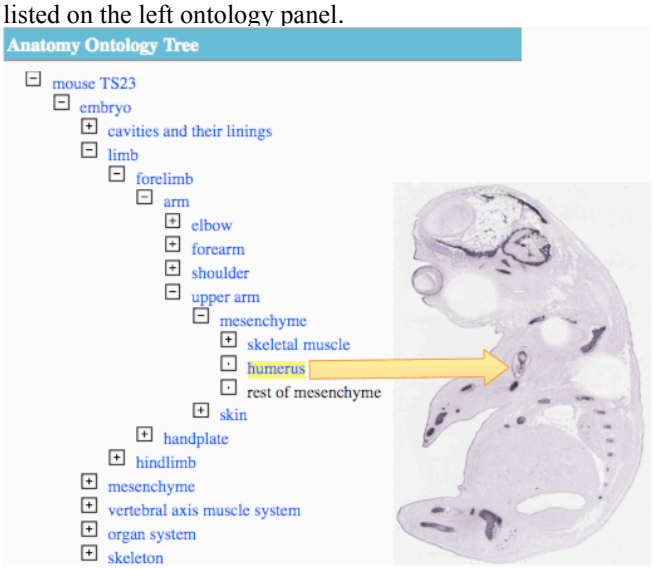
II. PARALLEL DATA INTENSIVE APPLICATIONS: AN IMAGE PATTEN RECOGNITION CASE STUDY

A. Backgroung of the use case

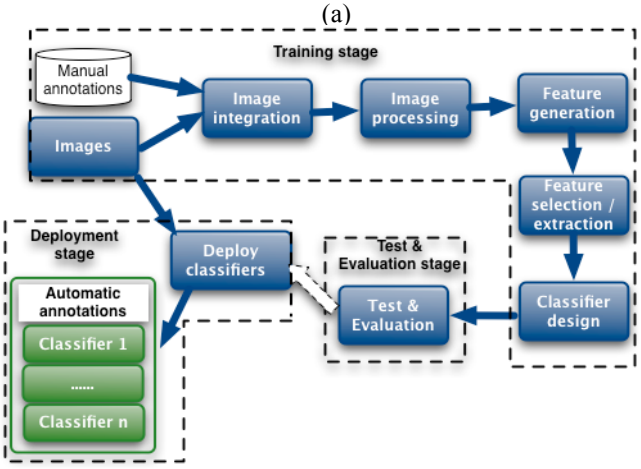
The use case is from EUREXpress [4][5], which aims to build a transcriptome-wide atlas for developing mouse embryo established by RNA in situ hybridisation. The project uses automated processes for in situ hybridisation experiments on all genes of whole-mount wild-type mouse embryos at the Stage 23. The result is many images of embryo sections that are stained to reveal where RNA is

 ISBN: 978-1-61208-101-4

present, namely, where gene patterns are expressed in embryos. These images were then annotated by human curators. The annotation consists of tagging images with anatomical terms from the ontology for mouse anatomy development. If an image is tagged with an anatomical term, it means that anatomical component is present in the image and it is exhibiting gene expression in some part of the component. So far, 80% of images (4 Terabytes in total) have been manually annotated by human curators. The goal is to automatically perform annotation by tagging the remaining 20% with the correct terms of anatomical components (there are still 85,824 images to be annotated with a vocabulary of 1,500 anatomical terms) and to provide a means of tagging future data automatically. The input is a set of image files and corresponding metadata. The output will be an identification of the anatomical components that exhibit gene expression patterns in each image. This is a typical pattern recognition task. As shown in Figure1 (a), we first need to identify the features of `humerus' in the embryo image and then annotate the image using ontology terms listed on the left ontology panel.



(a)



(b)

Figure 1. An image pattern recognition task

To automatically annotate images, three stages are required: at the training stage, the classification model has to be built, based on training image datasets with annotations; at the testing stage, the performance of the classification model has to be tested and evaluated; then at the deployment stage, the model has to be deployed to perform the classification of all non-annotated images. We mainly focus on the training stage in this case. The processes in the training stage include integration of images and annotations, image processing, feature generation, feature selection and extraction, and classifier design, as shown in Figure1 (b).

The specific processes are described as follows:

- Image integration: before starting the data mining, we need to integrate data from different sources: the manual annotations have been stored in the database and the images are located in the file system. The output of this process is images with annotations.
- The size of the images is variable and there is noise in the images. We use image scaling and image filtering methods to rescale and denoise the images. The output of this process is standardised and denoised images, which can be represented as 2-dimensional arrays.
- After image pre-processing, we generate those features that represent different gene expression patterns in images. The resulting features of wavelet transforms are 2-dimensional arrays.
- Due to the large number of features, the features need to be reduced and selected for building a classifier. Either feature selection or feature extraction or both can do this. Feature selection selects a subset of the most significant features for constructing classifiers. Feature extraction performs the transformation on the original features for the dimensionality reduction to obtain a representative feature vectors for building up classifiers.
- The main task in this case is to classify images into the right gene terminologies. The classifier needs to take an image's features as an input, and outputs a 'yes' or 'no' for each of anatomical features.

B. Parallel the image patten recognition task

1) Overview of parallel approach

It is well known that the speedup of an application to solve large computational problems is mainly gained by the parallelisation at either hardware or software levels or both (e.g., signal, circuit, component and system levels) [6]. Hardware parallelism focuses on signal and circuit levels and normally is constrained by manufacturers. Software parallelism at component and system levels can be classified into two types: automatic parallelisation of applications without modifying existing sequential applications and construction of parallel programming models using various software technologies to describe parallel algorithms and then match applications with the underlying hardware platforms. Since the nature of auto-parallelisation is to

recompile a sequential program without the need for modification, it has a limited capability of parallelisation on the sequential algorithm itself. Mostly, it is hard to directly transform a sequential algorithm into parallel ones. While parallel programming models try to address how to develop parallel applications and therefore can maximally utilise the parallelisation to obtain high performance, it does need more development effort on parallelisation of specific applications. In general, three considerations when parallelising an application include:

- How to distribute workloads or decompose an algorithm into parts as tasks?
- How to map the tasks onto various computing nodes and execute the subtasks in parallel?
- How to coordinate and communicate subtasks on those computing nodes.

There are mainly two common methods for dealing with the first two questions: data parallelism and task parallelism. *Data parallelism* represents workloads are distributed into different computing nodes and the same task can be executed on different subsets of the data simultaneously. *Task parallelism* means the tasks are independent and can be executed purely in parallel. There is another special kind of the task parallelism is called 'pipelining'. A task is processed at different stages of a pipeline, which is especially suitable for the case when the same task is used repeatedly. The extent of parallelisation is determined by dependencies of each individual part of the algorithms and tasks.

As for the coordination and communication among tasks or processes on various nodes, it depends on different memory architectures (shared memory or distributed memory). A number of communication models have been developed [7][8]. Among them, the MPI (Message Passing Interface) has been developed for HPC parallel applications with distributed memory architectures and has become the de-facto standard. There is a set of implementations of MPI, for example, OpenMPI [9], MPICH [10], GridMPI [11] and LAM/MPI [12].

2) Parallel approach in this use case

Based on the flow chart of the use case in Figure 1b, we model this image pattern recognition task as a direct acyclic graph. Each node of the graph is a functional module, a process or a subtask and the edge connected between nodes is data flow, which shows true data dependency between them. A direct acyclic graph for the training stage is shown in Figure 2. The left-hand side shows the atomic processes of the training stage. The right-hand side shows a higher-level abstraction of the task. For instance, feature selection and extract is composed of 'featureMean', 'featureVar' and 'featureExtract' atomic processes.

In terms of the nature of the algorithm used in this case, parallel approaches used are mainly data parallelism and a typical task parallelism (pipelining).

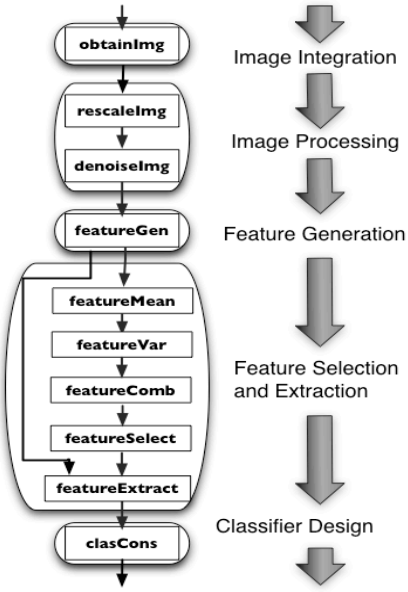


Figure 2. The processes of the task

In terms of the task graph shown in Figure 2 and the dependencies between the processes, data parallelism can be applied here. Processes such as 'rescaleImg' and 'denoiseImg' to 'featureGen' can have multiple instances invoked without any internal status to be maintained between these instances. After we get image samples from the process 'obtainImg', the samples can be partitioned into subsets and distributed to different nodes that run multiple instances of these processes. Therefore, the data can be parallelised.

Furthermore, parallelisation should consider how to decompose a process itself into parts and executed these parts in parallel. In this case, the decomposition of the algorithms mainly focuses on feature selection and extraction (i.e., Fisher's Ration algorithm [13]) and Classifier design (i.e., K-Nearest Neighbour-KNN [14]). We have developed parallel forms of these two processes, as shown in Figure 3 and Figure 4.

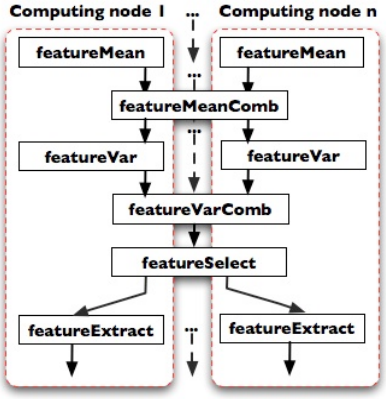


Figure 3. A parallel form of Fisher's Ratio

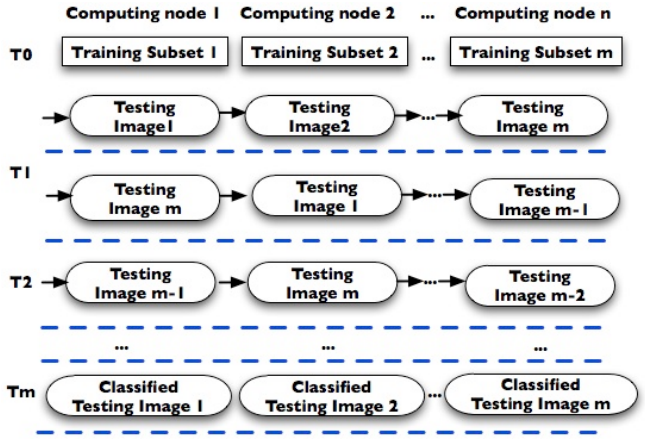


Figure 4. A parallel form of KNN

a) The rationale behind of parallel Fisher Ratio

The nature of the Fisher's Ratio algorithm for feature selection is to calculate Mean and Variance of image samples for two classes (C_1 and C_2), as shown in the following Equation 1 [13].

$$FisherRatio = \frac{(m_{1,i} - m_{2,i})^2}{(v_{1,i}^2 + v_{2,i}^2)} \quad (1)$$

where $m_{1,i}$ represents the mean of samples at the i[th] feature in Class C_1 and $m_{2,i}$ represents the mean of samples at the i[th] feature in Class C_2. $v_{1,i}$ represents the variance of samples at the i[th] in Class C_1. $v_{2,i}$ represents the variance of samples at the i[th] feature in C_2.

Therefore, the feature selection can be decomposed into smaller subtasks 'featureMean' and 'featureStd' and executed with subsets of image samples on nodes. The parallel form of the algorithm can be presented in Figure 4.

b) The Rationale behind of Parallel KNN

KNN is a classification algorithm to identify unknown samples into a class based on the nearest distance with the training samples. The common distance function is Euclidean distance. In this case, the samples are represented with features as vectors. The Euclidean distances therefore are calculated between each training sample and a testing sample, and then the nearest ones can be chosen. For all of training samples, the calculation is a typical iteration task. To exploit parallelisation in this algorithm, we use a special task parallelism called 'pipelining'. We divide an iteration of the KNN task into several pipeline stages. The sample images can be partitioned to subsets. The task of the distance calculation between subsets of training samples and the unclassified testing sample are executed at different stages respectively. Figure 4 shows the parallel form of the KNN algorithm using the concept of 'pipelining'.

III. EXPAERIMENTATION AND EVALUATION IN CLOUD COMPUTING

To evaluate the performance of the parallel image pattern recognition task described above we must conduct experiments on many physical computer nodes. However, to buy and maintain physical resources is costly and time consuming. We therefore make use of IaaS in the form of cloud computing to perform our evaluation.

A. Overview of Cloud Computing

Cloud computing is an evolution of various forms of distributed computing systems: from original distributed computing, parallel computing (cluster, to service-oriented computing (e.g., grid). Cloud computing extends these various distributed computing forms by introducing a business model, in which the nature of on-demand, self-service and pay-by-use of resources is used. Cloud computing sells itself as a cost-effective solution and reliable platform. It focuses on delivery of services guaranteed through Service Level Agreements. The services can be application software—SaaS, development environments for developing applications—PaaS, and raw infrastructures and associated middleware —IaaS.

In this study, without buying expensive clusters or supercomputers, we adopt the IaaS model that enables us to rent compute infrastructures from cloud providers. We have developed and then deployed our application onto rented compute infrastructure.

Among cloud providers (Amazon, Google, Saleforce, IBM, Microsoft, etc), we have chosen Amazon EC2, one of the most popular cloud providers, who provides Infrastructure-as-a-Service to users, with a capability to allow users to elastically expand or shrink the amount of resources used. Unlike traditional physical resource leasing, Amazon EC2 uses virtualisation techniques (e.g., Xen [15]) and releases virtual machines (instances) to users. Table 1 lists standard virtual instances from Amazon as an example (please refer to other types of virtual instances in [3]). Users can choose any instance and operating systems and pay for the time an instance is 'switched on' (Table 1 also shows the pricing for using Linux/Unix operating system on 1 June 2010).

TABLE I. AN EXAMPLE OF INSTANCE TYPES OF AMAZON EC2

Stand. Instance	Cores	RAM	Bit	I/O	Disk	Cost Linux/Unix
Small (m1.small)	1	1.7GB	32	Med.	160GB	$0.085/h
large (m1.large)	4	7.5GB	64	High	850GB	$0.34/h
Extra large (m1.xlarge)	8	15GB	64	High	1690gb	$0.68/h

B. Experimentatation and Evaluation in the Cloud

1) Performance metrics

We have used speedup as a performance indicator. Speedup is considered as a ratio between the execution time of the image pattern recognition task (Ts) on one single computing node and the execution time of the task on multiple computing nodes (Tm), represented as follows:

$$S = \frac{T_s}{T_m}$$

The execution on one single node means all of processes, data, and storage on one computing node. The execution on multiple computing nodes includes any of these situations: distributed data, distributed processes and distributed storage.

2) Experiment configuration

We have developed the image pattern recognition task in both sequential and parallel modes. We deployed them and conducted our experiments on small instances (virtual computers) of Amazon's EC2, with speedup as a performance indicator. To create an infrastructure on EC2, we have registered a user account with Amazon EC2. We launch instances by specifying the instance types and virtual images that we have created for the image pattern recognition task. In this study, we have used 12 small instances for experimentation (at this moment Amazon EC2 limits users to a maximum of 20 instances running concurrently). The specific configuration is listed in the second row in Table 1 (m1.small).

3) Evaluation result

We have measured the performance under two factors: 1) changing the size of input (i.e., number of image samples) and 2) varying number of nodes (i.e., virtual computers). We report averages over 30 independent runs for each scenario (the choice of 30 runs is based on statistics). The evaluation result is shown in Figure 5, Figure 6 and Figure 7. To validate the cost-effectiveness of Infrastructure-as-a-Service via cloud provision, we calculated the cost shown in Figure 8. Figure 5 shows the speedup. The X-axis represents input size and the Y-axis represents the speedup. The result demonstrates that the speedup increases with the increase of the number of computing nodes. The speedup increases with the increase of input size when the number of machines is 12. It fully embodies the advantage of parallel computing when processing heavy loads (i.e., with 4000 images).

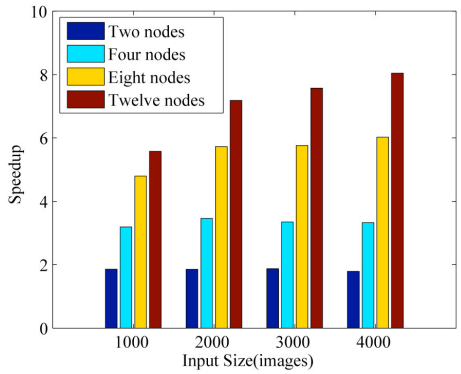


Figure 5 Average speedup for experiments with 1000, 2000, 3000 and 4000 images when using 2, 4, 8 and 12 nodes running concurrently

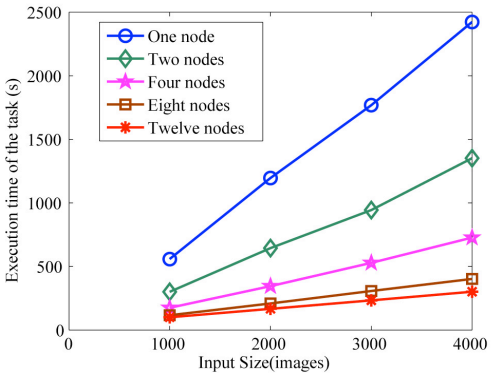


Figure 6 Average execution time of the task with increasing number of images for different number of virtual nodes running concurrently

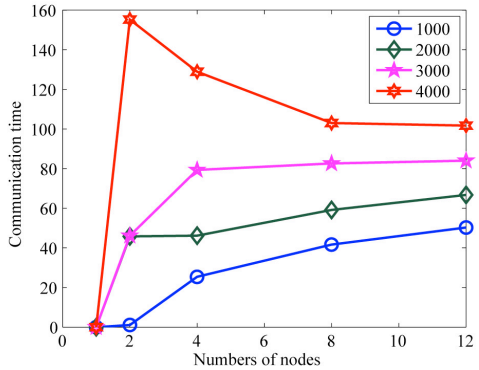


Figure 7 Average communication time vs. the numbers of nodes for increasing number of nodes with different number of images

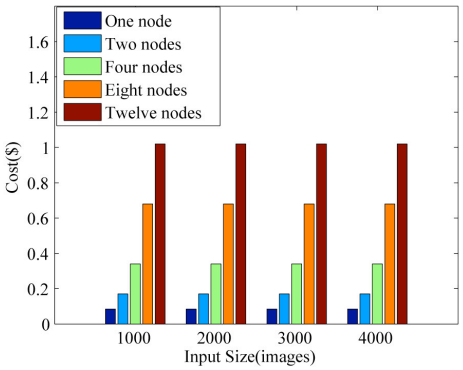


Figure 8 Average cost for performing the task using Amazon EC2 in US Dollars with 1000, 2000, 3000 and 4000 images when using 1, 2, 4, 8 and 12 virtual nodes running concurrently

Figure 6 shows the execution time under different input sizes and different numbers of computing nodes. With the increase of input size, the execution time increases; with the increase of numbers of nodes, the execution time decreases.

Figure 7 shows the relationship between communication time and numbers of nodes. With the increase of input size,

the communication time increases; with the increase of the number of nodes, the communication time increases in the first instance and then the increase rate slows down. In Figure 7, the communication time with input size 4000 at two nodes has a spike. This is mainly caused by the latency of the Cloud during the execution after we have checked various similar experiments.

Figure 8 shows average costs for running a full task in US Dollars using Amazon's EC2. In this case, since the maximum execution time of the whole task for various experiments running on one node is taken less than one hour, the cost increases with the number of the virtual computing nodes, for example, for one node, the cost is $0.085; for 12 nodes, the cost is $1.02. From cost-effectiveness point of view, the users may consider running fewer virtual nodes (despite that the execution time of the task running on 12 nodes is less than 5 minutes, comparing with 41-minute task execution on one node).

IV. CONCLUDING REMARKS

In this paper, by a way of a case study, we explore parallel approaches for an image pattern recognition task in the Life Sciences and have developed both sequential and parallel versions for this task. We have conducted a comparison of different parallel setups that ran in a cloud infrastructure provided by Amazon's EC2. The performance of parallel processing of the task has been evaluated where the speedup increases with the increase of numbers of virtual computer nodes and we achieve a linear scale up when using maximum input size of 4000 images. The communication time increases with the increase of input size. With the increase of number of virtual computer nodes, the communication time increases at the first beginning and then the increase rate slows down.

We have calculated the average cost for running the whole task. The maximum cost is $1.02 when lunching 12 virtual nodes, which shows that the use of Cloud computing is still a cheaper solution comparing with that of buying supercomputers or clusters. Nevertheless, it is also found the cost increases with the number of computing nodes as long as the maximum execution time of the task running one node is less than one hour. In this case, there is still a possibility that the users may choose to lunch fewer virtual nodes for cost saving.

REFERENCES

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud Computing and Emerging IT Platforms: Vision, Hype and Reality for Deliverung Domputing as the 5th Utility," Future Generation Computer Systems, Vol.25, No. 6., pp.599-616, June 2009.

[2] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud Computing and Grid Computing 360-Degree Compared," Grid Computing Environments Workshop, 2008.

[3] Amazon Elastic Compute Cloud, http://aws.amazon.com/ec2/, Retrieved 15, May, 2010.

[4] L. Han, J. van Hemert, R. Baldock, and M. Atkinson, "Automating Gene Expression Annotation for Mouse Embryo," In Lecture Notes in Computer Science (Advanced Data Mining and Applications, ADMA 2009), vol. LANI 5678, pp.469-478, 2009.

[5] EURExpress-II project, http://www.eurexpress.org/ee/, Retrieved 10, May, 2010.

[6] L. Silva, and R. Buyya, "High Performance Cluster Computing: Programming and Applications," ch. Parallel Programming Models and Paradigms, pp. 4–27. No. ISBN 0-13-013785-5. Prentice Hall PTR, NJ, USA, 1999

[7] P. S. Pacheco Parallel Programming with MPI. Morgan Kaufmann Publishers, Inc., 1997.

[8] PVM, 2009, http://www.csm.ornl.gov/pvm/, Retrieved 5 May, 2010.

[9] OpenMPI, 2009, http://www.open-mpi.org/, Retrieved 5 May, 2010.

[10] MPICH, http://www.mcs.anl.gov/research/projects/mpi/mpich1/, Retrieved 5 May, 2010.

[11] GridMPI, http://www.gridmpi.org/index.jsp, Retrieved 5 May, 2010.

[12] LAMMPI, http://www.lam-mpi.org/, Retrieved 5 May, 2010.

[13] R. O. Duda and P. E., Hart, "Pattern Classification and Scene Analysis," John Wiley & Sons, 1973.

[14] T. M. Cover and P. E. Hart, "Nearest Neighbor Pattern Classification," Information Theory, IEEE Transactions on, Vol. 13, No. 1. pp. 21-27,1967.

[15] P. Barham, B. Dragovic, K. Fraser, S. Hand, T.L. Harris, A. Ho, R. Neugebauer, I. Pratt and A. Warfield , "Xen and the art of virtualisation," In SOSP, pp. 164-177. ACM, 2003.

 ISBN: 978-1-61208-101-4