Scalable Symmetric Key Cryptography Using Asynchronous Data Exchange in Enterprise Grid

Medhat Awadallah[1] and Ahmed Youssef[2]

**[1] Electrical and Computer Engineering Dept, Sultan Qaboos University
Muscat, Oman**

**[2] Information Systems Department, King Saud University
Riyadh, 11543, KSA**

Abstract

Symmetric key cryptography is one of the most critical computing problems that need high performance computing power resources. The use of large key sizes and complex encryption/decryption algorithms to achieve unbreakable state has led to an increased time computational complexity. Traditionally, this problem is solved in the grid environment by partitioning data streams into several blocks of a predefined size. This is done while sequentially reading the data from the raw data file. The grid manager node then takes the responsibility of passing these blocks to the executer nodes where different blocks are processed separately and simultaneously. Although this technique allows parallel processing to speed up the encryption/decryption process, creating blocks by sequentially reading the data file and distributing these blocks on executers synchronously by the central manager node is a poor technique and a source of delay. In this paper, we present a novel approach that tackles this problem by allowing executers to access data file at random and asynchronously exchange the blocks among them, thereby, delay is significantly reduced and data size can be scaled up. In order to show the merit of our approach experiments have been conducted through a system-level middleware for grid computing called Alchemi. The results show a remarkable performance enhancement in our approach over traditional approaches in terms of speed.

Keywords: Grid computing, Grid Middleware, Alchemi, Data Encryption/Decryption, Symmetric Key Cryptography.

1. Introduction

The concept of grid computing is gaining popularity with the emergence of the Internet as a medium for global communication and the wide spread availability of powerful computers and networks as low-cost commodity components [1]. The computing resources and special class of scientific devices or instruments are located across various organizations around the globe. These resources could be computational systems (such as traditional supercomputers, clusters [2], or even powerful desktop machines), special class of devices (such as sensors, radio telescope, and satellite receivers), visualization platforms, or storage devices. A number of applications need more computing power than can be offered by a single resource/reasonable time and cost. This promoted the exploration of logically coupling geographically distributed high-end computational resources and using them for solving large-scale problems. Such emerging infrastructure is called computational (power) grid [3]. Computational grids are expected to offer dependable, consistent, pervasive, and inexpensive access to high-end resources irrespective of their physical location and the location of access points [3].

The grid must be designed and created in such a way that their components (fabric, middleware, and higher-level tools) and applications handle the key design issues in a coordinated manner. For instance, grid middleware offers services for handling heterogeneity, security, information, allocation, and so on. Higher level tools, such as resource brokers, support dynamic adaptability through automatic resource discovery, trading for economy of resources, resource acquisition, scheduling, the staging of data and programs, initiating computations, and adapting to changes in the grid status [4]. In addition, they also need to make sure that domain autonomy is honored but still meets user requirements such as quality of service in coordination with other components.

Symmetric key cryptography is one of those complex large-scale problems that need high computing power to be solved efficiently. Cryptanalysis on this problem is

encouraging the use of larger key sizes and complex algorithms to achieve an unbreakable state [5]. However, this leads to an increase in computational complexity. Therefore, many researchers investigated the deployment of high performance computing approaches such as grid computing, cluster computing and Peer-to-Peer (P2P) to develop efficient and cost-effective symmetric key cryptography schemes. By utilizing these approaches, the performance of symmetric key cryptography can be improved through parallel execution [5].

Traditionally, this problem is solved in the Grid environment by partitioning data streams into several blocks of a predefined size [5, 14]. This is done while sequentially reading the data from the raw data file. The grid manager node then takes the responsibility of assigning these blocks to the executer nodes where different blocks are processed separately and simultaneously. Although this technique allows parallel processing to speed up the encryption/decryption process, creating blocks by sequentially reading the data file and distributing these blocks on executers synchronously by the central manager node is poor technique and a source of delay. In this paper, we present a novel approach that tackles this problem by allowing executers to access data file at random and asynchronously exchange data blocks among them. The proposed approach is faster and more scalable than traditional approaches since it avoids the delay occurs due to partitioning the data into blocks by the grid application while reading the file and passing large sets of data by the manager to the executers. The validity and the feasibility of the proposed approach is examined through a system level middleware for creating grid computing environment called Alchemi. Experiments show a remarkable performance enhancement in our approach over traditional approaches.

The rest of this paper is organized as follows: in section 2 we outline background information. Section 3 presents the open source, Alchemi, which provides the middleware for creating an enterprise grid-computing environment. Section 4 presents DES (Data Encryption Standard); Encryption and Decryption using Alchemi grid computing framework. Our proposed approach is presented in Section 5. Section 6 presents performance evaluation experiments conducted through Alchemi and discusses the results. Finally, section 7 gives our conclusions.

2. Background

In order to meet the increasing demand of large-scale scientific computation in the fields of life sciences, biology, physics, and astronomy, the notion of "computational grid" was proposed in mid 1990s [6]. It has been observed that computers (such as PCs, work-stations, and clusters) in the Internet are often idle. Grid computing aims to integrate idle computational power over the Internet and provide powerful computation capability for users all over the world [7]. Since a grid connects numerous geographical distributed computers fashion, an important issue is how to evenly distribute submitted tasks to nodes. This is a load balancing problem, one of the scheduling problems on the grid. By solving this problem, the computational resources of the grid can optimally be utilized. To perform grid computation, the process must be divisible into several sub-processes and run in parallel. The following are some of famous projects that have been designed for grid computation.

The human genome is composed of 24 distinct chromosomes with about 3 billion DNA base pairs organized into 20,000~25,000 genes [8]. To identify these genes and determine the sequences of 3 billion DNA base pairs, running a computer simulation would be expensive and time consuming. Based on computational grid, the Human Genome Project was completed in 2003, three years ahead of the target goal. After the Human Genome Project was completed, scientists wanted to understand the function of human proteins, which affect human health, to discover the cure for diseases such as AIDS and cancer. Human Proteome Folding (HPF) project was started and ran on two computational grids [9].

Chemical reactions or molecular behavior can be huge and complicated processes. Some chemistry problems, like quantum mechanics, would take hundreds of years to simulate on a personal computer. Computational Chemistry Grid [10] is one of the most important virtual organizations, which provides all necessary software and resources for computational chemistry. Searching for extraterrestrial intelligence (SETI), is a compelling scientific research that utilizes grid computation technology to analyze space-based radio signals collected from a radio telescope, at Arecibo, Puerto Rico [12].

Grid computation is not only used in science, but also in business computation, where all corporate resources can be pooled so they can be processed efficiently in parallel, according to the business demand. The Oracle 10g [11] runs all database systems in a virtual environment (grid) where all systems are considered a resource pool, using resources efficiently and dynamically for business needs. Grid computation can also be used in financial modeling,

earthquake simulation, and climate/weather modeling, which are complex processes requiring an intricate infrastructure. A dynamic grid environment, which can perform parallel processing under a collaborative network, must be created to deliver the information. A number of projects worldwide are actively exploring the development of grid computing technology. They include Globus [13], Legion [15], NASA Information power grid [16], and Condor [17].

3. Windows-based grid computing framework (Alchemi)

The Alchemi grid-computing framework was conceived with the aim of making grid construction and development of grid software as easy as possible without sacrificing flexibility, scalability, reliability and extensibility. The key features supported by Alchemi are [18,19]:

- Windows based machine with .NET grid computing framework;
- Internet-based clustering of desktop computers without a shared file system
- Federation of clusters to create hierarchical, cooperative grids
- Dedicated or non-dedicated (voluntary) execution by clusters and individual nodes
- Object-oriented grid thread programming model (fine-grained abstraction)
- Web services interface supporting a grid job model (coarse-grained abstraction) for cross-platform interoperability (e.g., for creating a global and cross-platform grid environment via a custom resource broker component).

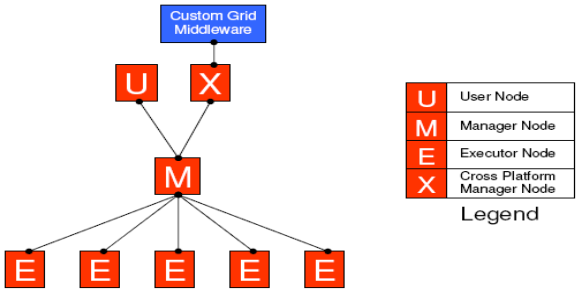
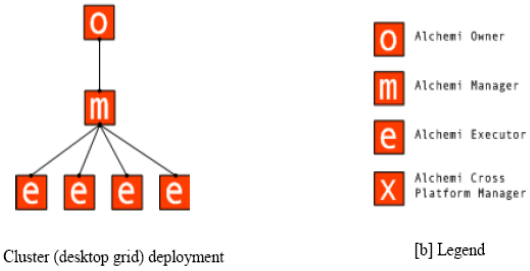


Fig. 1 Distributed components and their relationships [19]

Alchemi's distributed components consist of four types of nodes (or hosts) that take part in enterprise grid construction and application execution. These nodes include: User node, Manager node, Executer node and Cross platform Manger node, Fig. 1. An Alchemi enterprise grid is constructed by deploying a Manager node and one or more Executor nodes configured to connect to the Manager. One or more Users can execute their applications by connecting to the Manager. An optional component, the Cross Platform Manager, provides a web service interface to custom grid middleware. These components allow Alchemi to be utilized to create different grid configurations, which are desktop cluster grid, multi-cluster grid, and cross-platform grid (global grid). According to [19], these are described as follows:

Cluster (Desktop Grid): is the basic deployment scenario, a cluster (as shown in Fig. 2) consists of a single Manager and multiple Executors that are configured to connect to the Manager. One or more Owners can execute their applications on the cluster by connecting to the Manager. Such an environment is appropriate for deployment on Local Area Networks as well as the Internet.

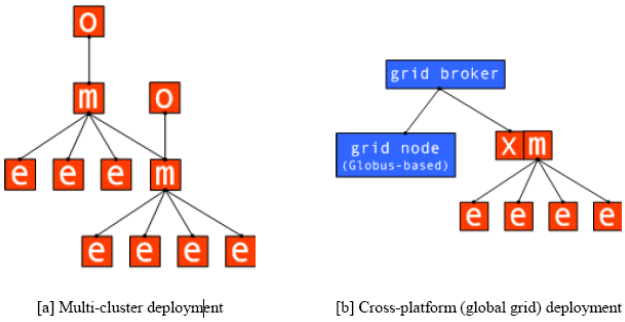


[a] Cluster (desktop grid) deployment [b] Legend

Fig. 2 Cluster (desktop grid) deployment [19]

Multi-cluster environment: is created by connecting Managers in a hierarchical fashion, Fig. 3.a. As in a single-cluster environment, any number of Executors and Owners can connect to a Manager at any level in the hierarchy. An Executor and Owner in a multi-cluster environment connect to a Manager in the same fashion as in a cluster and correspondingly their operation is no different from that in a cluster.

Global Grid: the cross platform manager is used to construct a grid conforming to the classical global grid model, Fig. 3.b. A grid middleware component such as a broker can use the Cross-Platform Manager web service to execute cross-platform applications (jobs within tasks) on an Alchemi node (cluster or multi-cluster) as well as resources grid-enabled using other technologies such as Globus.



[a] Multi-cluster deployment [b] Cross-platform (global grid) deployment

Fig.3: Alchemi deployment in [a] multi-cluster [b] global grid environments [19]

4. DES Encryption/Decryption using Alchemi grid computing framework

In this paper, we are concerned with symmetric key encryption algorithms such as DES and RC4 [5] as a grid application that runs under Alchemi framework. These algorithms are extremely fast (compared to public-key algorithms) and are well suited for performing cryptographic transformations on large streams of data. Typically, these algorithms are used to encrypt one block of data at a time. Block ciphers cryptographically transform an input block of n bytes into an output block of encrypted bytes. The Enterprise Grid Middleware (Alchemi) is used to solve the symmetric key cryptography problem as shown in Fig. 4. Alchemi provides a Software Development Kit (SDK) that can be used by developers to develop grid applications. The SDK includes a Dynamic Link Library (DLL) that supports object oriented programming model for multithreaded applications.

A grid application, called GridCryptoGraphy, has been built on top of Alchemi middleware grid environment as shown in Fig. 5. In this application, three main classes have been developed. The first class (GridCryptForm) is the interface to control and monitor the progress of the encryption and decryption process. It is also used to specify the location, connect user, and configure number of threads to be submitted to Alchemi manager. The classes GridEncryptThread and GridDecryptThred are the thread classes that run under Alchemi and they use the DES algorithm.

The flow of GridCryptoGraphy program starts by dividing the raw file into several blocks and separating these blocks in order to parallelize the encryption process. The block separation process is done by reading the data file sequentially according to the block size. Each part of the file (block) is assigned to a thread including the last block whose size is the remainder of the predefined block size. The manager node passes the threads to the executer nodes. After the threads return with the encrypted results, GridCryptoGraphy saves the encrypted data to an output file according to the order of the threads.

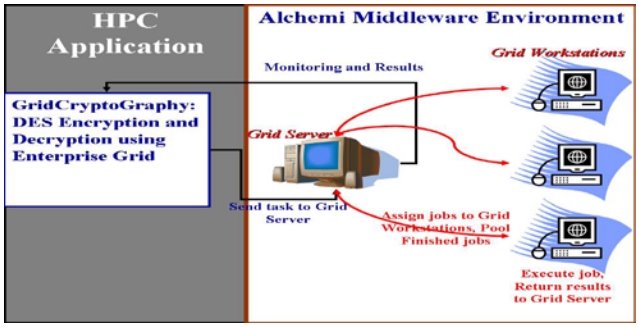


Fig.4: Symmetric Key Cryptography using Alchemi Middleware

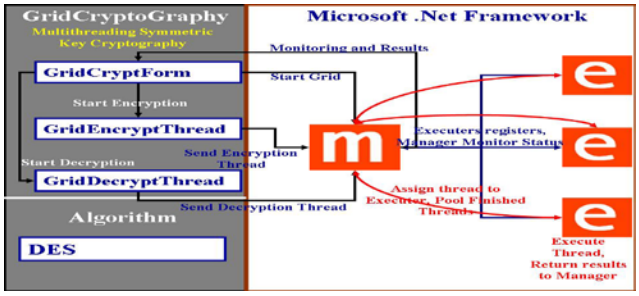


Fig. 5 GridCryptoGraphy Architecture

5. Cryptography using asynchronous data exchange

Our approach modeifies GridCryptoGraphy application to enable every executer accesses the input data file directly and at random by developing two new classes in this application. The classes are (AssGridEncryptThread and AssGridDecryptThread) which are the thread classes that run under Alchemi. The GridCryptoGraphy application only passes the names of input and output files to the manager as strings. It also passes the block size that each thread will access. So each thread will access the random access file according to each thread id multiplied with the block size, so if the block size is 1000, then thread number (0) will access the file in byte number (0) and thread

number (1) will access the file in byte number (1000), and so on. This process is accomplished through the reading and writing of the file, so the whole process is asynchronous as shown in Fig. 6. The job of the manager will only be initiating the threads, and not passing large datasets, thereby, we avoid delays due to creating large data blocks and passing them to executers.

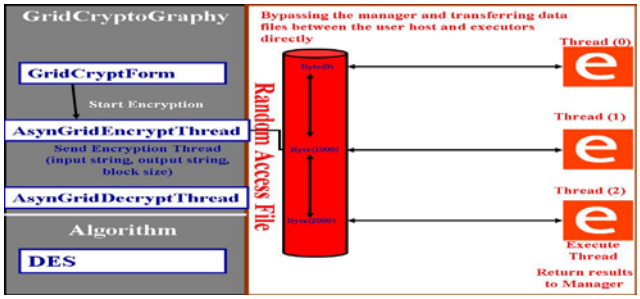


Fig. 6 GridCryptoGraphy Second experiment Architecture

6. Performance evaluation

Two experiments on large datasets have been conducted. Experiment 1 is a repetition of the GridCryptoGraphy application in [5] for a quantitative comparison with the results of experiment 2 that implements our approach. Eight executers have been used in each experiment with the following specification: for experiment 1, Intel® Pentium® 4 CPU 2.40 GHz, 512 MB RAM. For experiment 2, Intel® Pentium® 4 CPU 1.60 GHz, 128 MB RAM. Microsoft Windows XP Professional Version 2002 Service Pack 2. The nodes were interconnected over a shared LAN network of 100 Mbps.

The Alchemi manager was installed on a separate computer together with SQL Server 2000 and has the following specification: Intel® Pentium®4 CPU 3.00 GHz, 512 MB RAM. Microsoft Windows Server 2003 Standard Edition. The executions of the GridCryptoGraphy application run on the same computer with the manager.

A separate computer is used for monitoring the performance of the application with the following specification: Intel® Pentium®III CPU 731 MHz, 128 MB RAM. Microsoft Windows XP Professional Version 2002 Service Pack 2.

The encryption and decryption experiments were conducted on files of size 9645200 bytes (approximately 10 MB), 56610116 bytes (approximately 57 MB), 104858112 bytes (approximately 105 MB), 597393408 bytes (approximately 598 MB) and 1060842110 bytes (approximately 1061 MB) with different block sizes. For each file the encryption and decryption was carried on 1,2,3,4,5,6,7 and 8 executer nodes. The encryption experiments were conducted on file of size 104858112 bytes (approximately 105 MB) with 1, 5 and 10 Mb block size, which lead to the creation of 105, 21 and 11 work units respectively. For each experiment, the encryption was carried on 1, 2, 3, 4,5,6,7 and 8 executor nodes. Some snapshots of the program running are illustrated in Fig. 7(a-d).

The time performance results of experiment 2 are shown in Table 1.a and in Fig. 8.a. The speedup performance results are shown in Table 1.b and in Fig. 8.b where the speedup calculation is based on the following formula:
*Speedup=(((Time taken by 1 executer using 1 megabytes block size - Time taken by m executer using n megabytes block size)/ Time taken by 1 executer using 1 megabytes block size)*100)+100)*

Figure 9 and figure 10 show the comparison between the results of experiment 1 and experiment 2. Although executers used in experiment 1 have higher specifications (Pentium IV 2400 MHz processor and 512 MB of memory) than those (Pentium IV 1600 MHz processor and 128 MB of memory) used in experiment 2, It has been found that:

- Fig. 9 and Fig. 10 show remarkable improvements in the performance of our approach (experiment 2) compared to that of the traditional approach (experiment 1).
- In the first experiment, there is a drop in the performance after using 4 executers. In contrary, in the second experiment there was improvement in performance till 8 executers, therefore, larger files as the video file of size 1060842110 bytes (approximately 1061 MB) could successfully be encrypted.
- Although increasing the block size creates less work units and so the performance should be increased. It is found that the performance in experiment 1 is reduced compared with experiment 2.

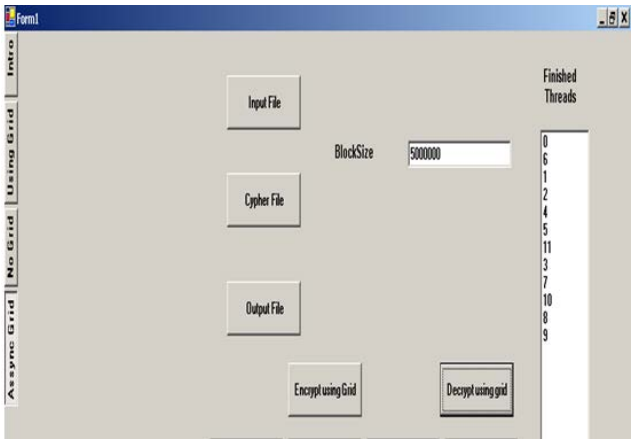


Fig. 7a: GridCryptoGraphy at runtime (monitoring of finished threads)

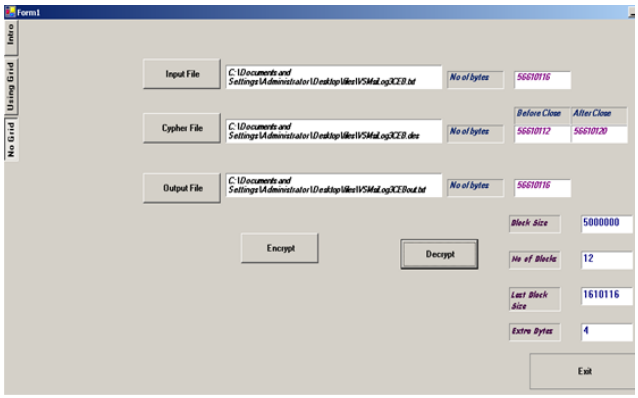


Fig.7b: GridCryptoGraphy at runtime (initializing files using 5-mega block size and 12 working unit)

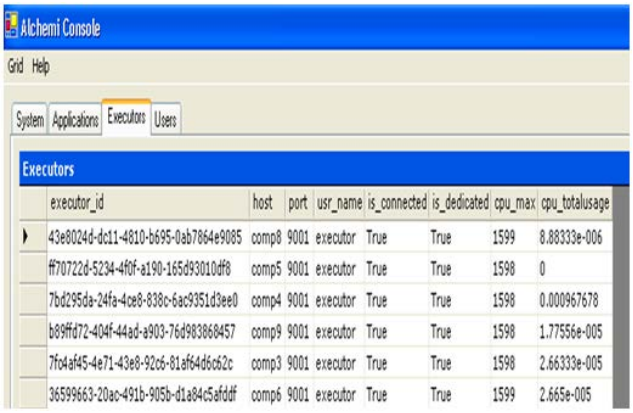


Fig. 7c: Six executers are working

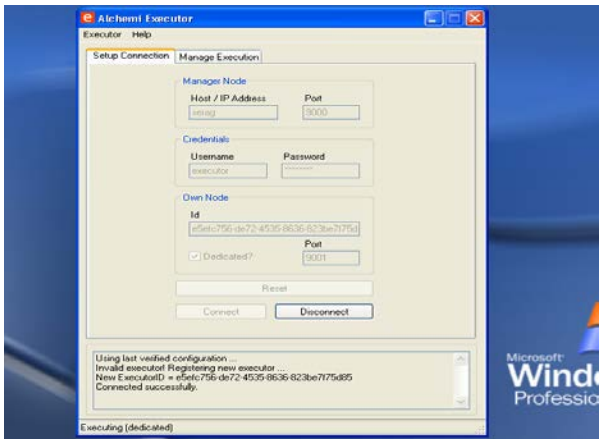


Fig. 7.d: Execution desktop

Table 1.a: Encryption time Performance results of 105Mega bytes file size

Block size / No of Executer	*(1 Mega) min :sec*	*(5 Mega) min :sec*	*(10 Mega) min: sec*
1	**00:42.563**	**00:35.469**	**00:35.141**
2	**00:23.625**	**00:26.328**	**00:24.063**
3	**00:23.469**	**00:24.266**	**00:24.013**
4	**00:22.641**	**00:23.328**	**00:23.375**
5	**00:21.859**	**00:21.281**	**00:23.078**
6	**00:20.078**	**00:20.828**	**00:22.188**
7	**00:21.313**	**00:20.391**	**00:20.172**
8	**00:21.547**	**00:18.469**	**00:20.141**

Table 1.b: Encryption Speedup Performance results of (105Mega bytes) file size

Block size / No of Executer	*(1 Mega) Speedup (%)*	*(5 Mega) Speedup (%)*	*(10 Mega) Speedup (%)*
1	**100.00**	**116.67**	**117.44**
2	**144.49**	**138.14**	**143.46**
3	**144.86**	**142.99**	**143.58**
4	**146.81**	**145.19**	**145.08**
5	**148.64**	**150.00**	**145.78**
6	**152.83**	**151.07**	**147.87**
7	**149.93**	**152.09**	**152.61**
8	**149.38**	**156.61**	**152.68**

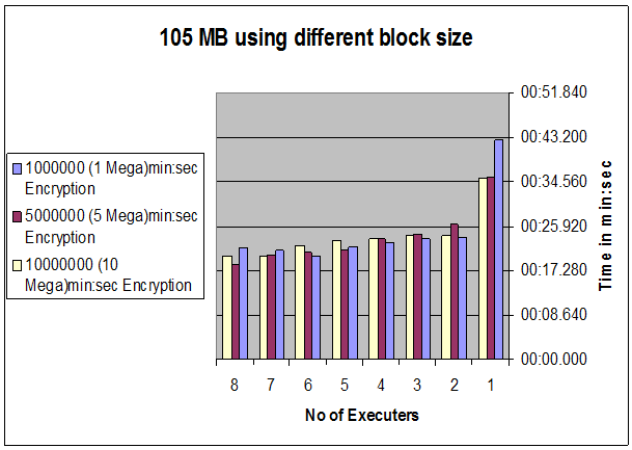


Fig. 8.a: Result graph of (104858112 bytes) file size with different block sizes

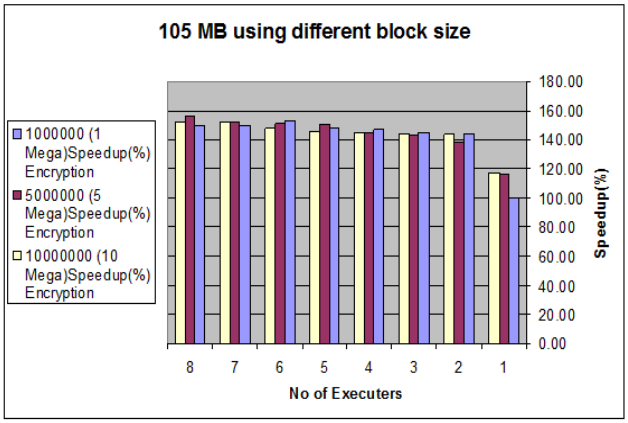


Fig. 8.b Speedup Result graph of (104858112 bytes) file size with different block sizes

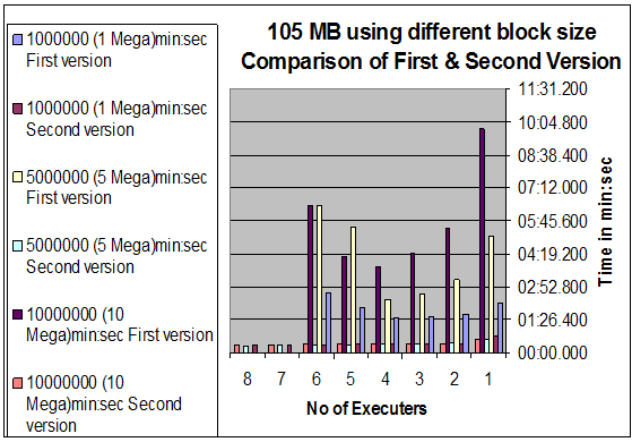


Fig 9.a: A time comparison of results to the first and second experiments (104858112bytes) file size with different block sizes

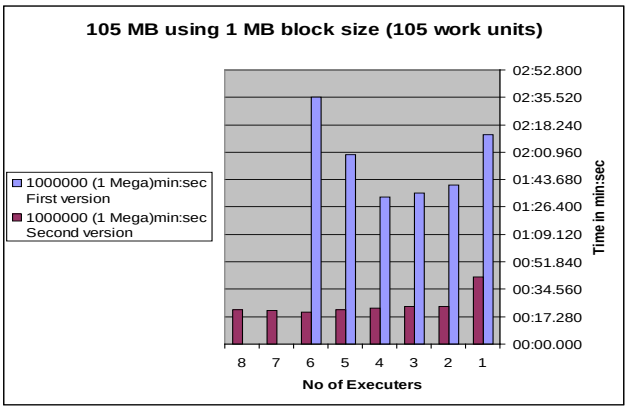


Fig. 9.b A time comparison of results to the First and Second Experiments of (104858112 bytes) file size with 1 Megabytes block sizes

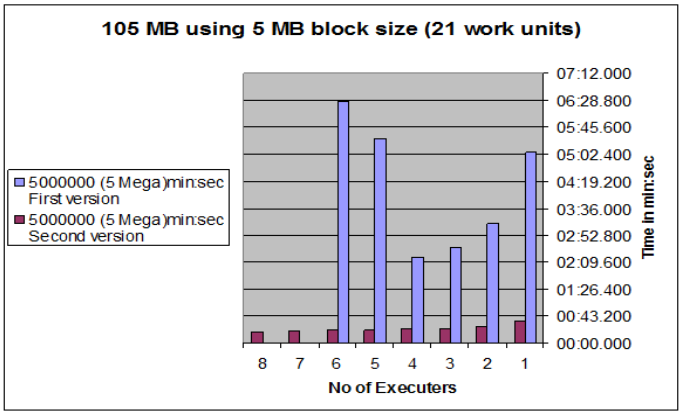


Fig. 9.c A time comparison of results to the First and Second Experiments of (104858112 bytes) file size with 5 Megabytes block sizes

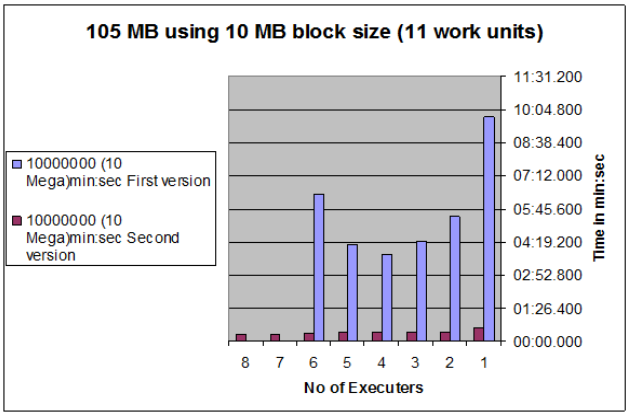


Fig. 9.d A time comparison of results to the First and Second Experiments of (104858112 bytes) file size with 10 Megabytes block sizes

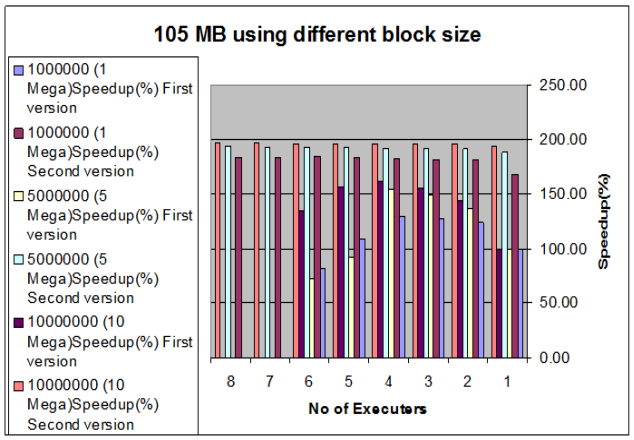


Fig. 10.a Speedup comparison of results to the First and Second Experiments of (104858112 bytes) file size with different block sizes

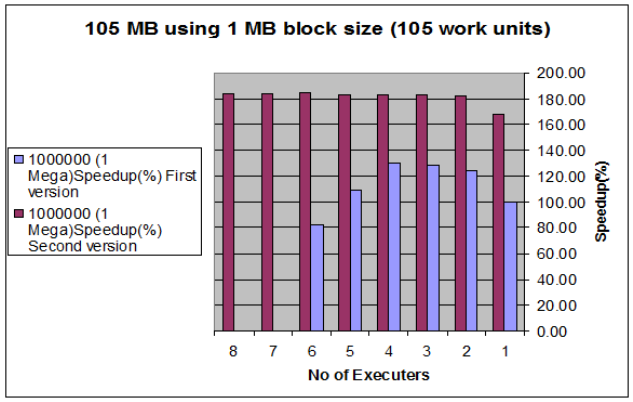


Fig (10.b) a speedup comparison of results to the First and Second Experiments of (104858112 bytes) file size with 1 Megabytes block sizes

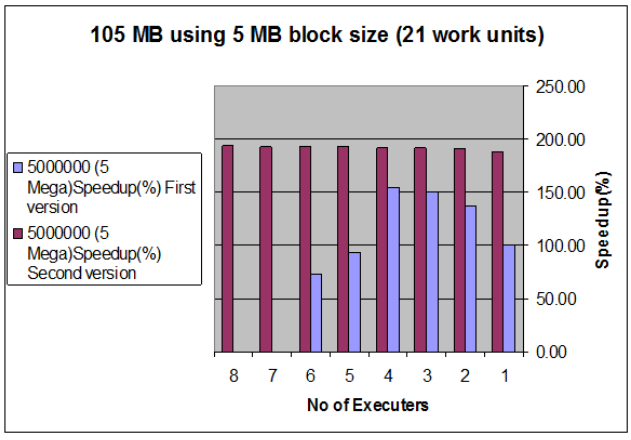


Fig. 10.c A speedup comparison of results to the First and Second Experiments of (104858112 bytes) file size with 5 Megabytes block sizes

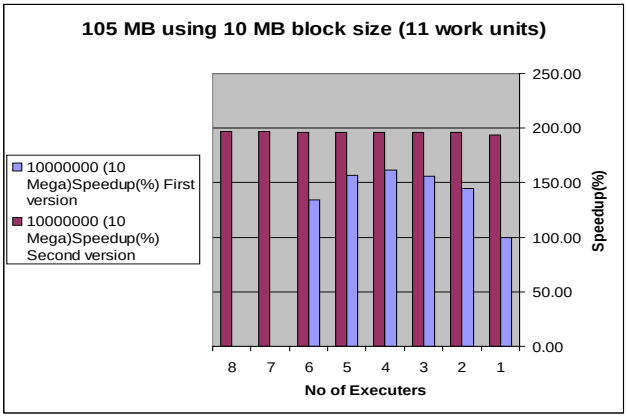


Fig (10.d) a speedup comparison of results to the First and Second Experiments of (104858112 bytes) file size with 10 Megabytes block sizes

7. Conclusions

This paper presents a grid based solution for solving the complex and large-scale problem of symmetric key cryptography that requires high performance computing resources. The problem was solved through a system-level middleware infrastructure called Alchemi. Alchemi is capable of creating an enterprise grid computing environment by harnessing windows machines and provide users with seamless computing ability and uniform access to resources in the heterogeneous grid environment. The proposed approach enhances the performance in terms of speed and limits the communication overhead. It is also scalable and cost-effective due to the effective and efficient utilization of a commodity-based high performance-computing platform.

References

[1] R. Buyya, D. Abramson and J. Giddy, "Driven Resource Management Architecture for Computational Power Grids". The 2000 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'2000), Las Vegas, 2000.

[2] http://recerca.ac.upc.edu/conferencies/AGC2007/, "Agent based Grid Computing".7th IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2007) Rio de Janeiro, Brazil, 154-17 May 2007.

[3] I. Foster and C. Kesselman, (editors), "The Grid: Blueprint for a New Computing Infrastructure", Morgan Kaufmann Publishers, USA, 1999.

[4] I. Foster, "Service-Oriented Science". Science, vol. 308, May 6, 2005.

[5] A. Setiawan, D. Adiutama, J. Liman, A. Luther and R. Buyya, "GridCrypt: High Performance Symmetric Key Cryptography Using Enterprise Grids". Liew, K. M. (editors) PDCAT, Springer-Verlag, pp. 872-877, 2004.

[6] I. Foster, "What is the Grid? A Three Point Checklist", GRIDToday, July 20, 2002.
[7] I. Foster, "The Grid: A New Infrastructure for 21st Century Science". Physics Today, 55(2):42-47, 2002.
[8] International Human Genome Sequencing Consortium. 2004
[9]The Human Proteome Folding Project, http://www. Grid.org/projects/hpf/
[10] http:// www.worldcommunitygrid.org/
[11] D. Kusnetzky and C. W. Olofson, "Oracle 10g: Putting Grids to Work", http://www.sswug.org/articles/viewarticle.aspx?id=18542
[12] http://setiathome.ss1.berkeley.edu/
[13] J. Bresnahan, M. Link, G. Khanna, Z. Imani, R. Kettimuthu and I. Foster. "Globus GridFTP: What's New in 2007" (Invited Paper), in Proceedings of the First International Conference on Networks for Grid Applications (GridNets 2007), Oct, 2007
[14] R. Kettimuthu, W. Allcock, L. Liming, J. Navarro and I. Foster. "GridCopy: Moving Data Fast on the Grid", in Proceedings of the Fourth High Performance Grid Computing Workshop to be held in conjunction with International Parallel and Distributed Processing Symposium (IPDPS 2007), March, 2007
[15] Legion – http://legion.verginia.edu/
[16] NASA IPG-http://www.ipg.nasa.gov
[17] Condor – http://www.cs.wisc.edu/condor/
[18] R. Ranjan, X. Chu, C. A. Queiroz, A. Harwood, R. Buyya. "A self organizing federation of Alchemi Desktop grids". Grids lab and P2P group, Australia, 2007.
[19] A. Luther, R. Buyya, R. Ranjan and S. Venugopal, "Alchemi: A .NET-based Grid Computing Framework and its Integration into Global Grids", Technical Report, GRIDS-TR-2003-8, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, December 2003.

Medhat Awadallah is an assistant professor at Electrical and Computer Engineering Department, Sultan Qaboos University. He obtained his PhD from university of Cardiff, UK. MSc and BSc from Helwan university, Egypt. His research interest includes cloud computing, sensor networks, high performance computing and real time systems.

Ahmed Youssef is an assistant professor at college of computer and information sciences, King Saud University. He obtained his P.h.D. and M.Sc. in computer science and engineering from university of Connecticut, USA. M.Sc and B.Sc in electronics and communications engineering from Helawn university, Egypt. His research interest includes cloud computing, mobile computing, high performance computing and information security.