**Military Technical College
Kobry El-kobbah,
Cairo, Egypt**

**5th International Conference on
Electrical Engineering
ICEENG 2006**

SYMMETRIC KEY CRYPTOGRAPHY USING WINDOWS-BASED GRID COMPUTING FRAMEWORK

Ahmed Serag Eldin[*], Ismail Abd Elghafar[*], Alaa Ahmed[*], Gouda Ismail[*]

ABSTRACT

This paper deals with one of the most critical computing problems which is Cryptography application, it is one of the problem that needs high computing power resources.
This paper introduces Symmetric Key Cryptography using DES (Data Encryption Standard) algorithm and employs a computing Grid for dealing with this problem. It presents a Grid for solving this problem that has been implemented by Alchemi which is an open source project developed at the University of Melbourne, which provides middleware for creating an enterprise grid computing environment by harnessing windows machines. The grid has been tested and results have been analyzed, there was an increase in performance over the single processor, but the performance improvement is limited by the I/O and communication overhead.

1 INRODUCTION

The last decade has seen a substantial change in the way we perceive and use computing resources and services. A decade ago, it was normal to expect one's computing needs to be serviced by localized computing platforms and infrastructures. This situation has changed. The change has been caused by, among other factors, the take-up of commodity computer and network components. A consequence of these changes has been the capability for effective and efficient utilization of widely distributed resources to fulfill a range of application needs. [1, 2, 3]

As soon as computers are interconnected and communicating, we have a distributed system, and the issues in designing, building and deploying distributed computer systems have now been explored over many years. An increasing number of research groups have been working in the field of wide-area distributed computing. These groups have implemented middleware, libraries and tools that allow the cooperative use of geographically distributed resources unified to act as a single powerful platform for the execution of a range or parallel and distributed applications. This approach to computing has been known by several names, such as metacomputing, scalable computing, global computing, Internet computing and lately as Grid computing. [4, 5, 6]

The term "the Grid" was coined in the mid1990s to denote a proposed distributed computing infrastructure for advanced science and engineering [7, 8]. Considerable progress has since been made on the construction of such an infrastructure, but the term "Grid" has also been conflated, at least in popular perception, to embrace everything from advanced networking to artificial intelligence. By analogy, we adopt the term computational grid for the infrastructure that will enable the increases in computation. A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities. [9]

*Department of Computer Science, Military Technical College, Cairo, Egypt.

Cryptography protects data from being viewed or modified and provides secure channels of communication over otherwise insecure channels. Secret-key encryption algorithms use a single secret key to encrypt and decrypt data. You must secure the key from access by unauthorized agents because any party that has the key can use it to decrypt data. Secret-key encryption is also referred to as symmetric encryption because the same key is used for encryption and decryption. Secret-key encryption algorithms are extremely fast (compared to public-key algorithms) and are well suited for performing cryptographic transformations on large streams of data.

Typically, secret-key algorithms, called block ciphers, are used to encrypt one block of data at a time. Block ciphers (like RC2, DES, TrippleDES, and Rijndael) cryptographically transform an input block of *n* bytes into an output block of encrypted bytes. If you want to encrypt or decrypt a sequence of bytes, you have to do it block by block. Because the size of *n* is small (*n* = 8 bytes for RC2, DES, and TripleDES; *n* = 16 [the default]; *n* = 24; or *n* = 32 bytes for Rijndael), values larger than *n* have to be encrypted one block at a time.

In this paper we introduce Symmetric Key Cryptography using DES algorithm which was developed at IBM in 1977. For DES, data are encrypted in 64-bit blocks; the algorithm transforms 64-bit input in a series of steps into a 64-bit output. The same steps, with the same key, are used to reverse the encryption. [10]

2 ALCHEMI

There is rapidly emerging interest in grid computing from commercial enterprises. A Microsoft Windows-based grid computing infrastructure will play a critical role in the industry-wide adoption of grids due to the large-scale deployment of Windows within enterprises. For this purpose, we use a Windows-based grid computing framework called Alchemi implemented on the Microsoft .NET Platform.

While the notion of grid computing is simple enough, the practical realization of grids poses a number of challenges. Key issues that need to be dealt with are security, heterogeneity, reliability, application composition, scheduling, and resource management. The Microsoft .NET Framework provides a powerful toolset that can be leveraged for all of these, in particular support for remote execution (via .NET Remoting and web services), multithreading, security, asynchronous programming, disconnected data access, managed execution and cross-language development, making it an ideal platform for grid computing middleware.[11]

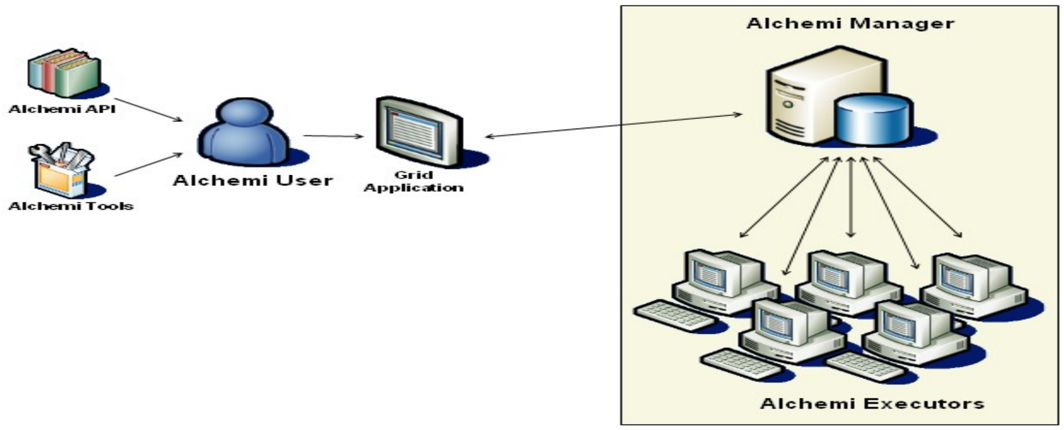


Fig. 1. Alchemi's main components

Alchemi is an open source project developed at the University of Melbourne, which provides middleware for creating an enterprise grid computing environment by harnessing windows machines. The main components of Alchemi are shown in Fig.1. Its main components are manager and executer that support a master-worker parallel model. One or more Users can execute their applications on the cluster by connecting to the Manager. An optional component, the Cross Platform Manager provides a web service interface to custom grid middleware. [12, 13, 14]

3 DESIGN AND ARCHETICTURE

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. The architecture of GridCryptoGraphy is shown in Fig.2.

In the GridCryptoGraphy application, we have developed three main classes. The first class (GridCryptForm) is the interface to control and monitor the progress of the encryption and decryption and connection with Alchemi manager and can be used to configure the number of threads to be submitted and specify the location of the Alchemi manager. The classes (GridEncryptThread and GridDecryptThred) are the thread classes that are run under Alchemi and it uses the algorithm classes.

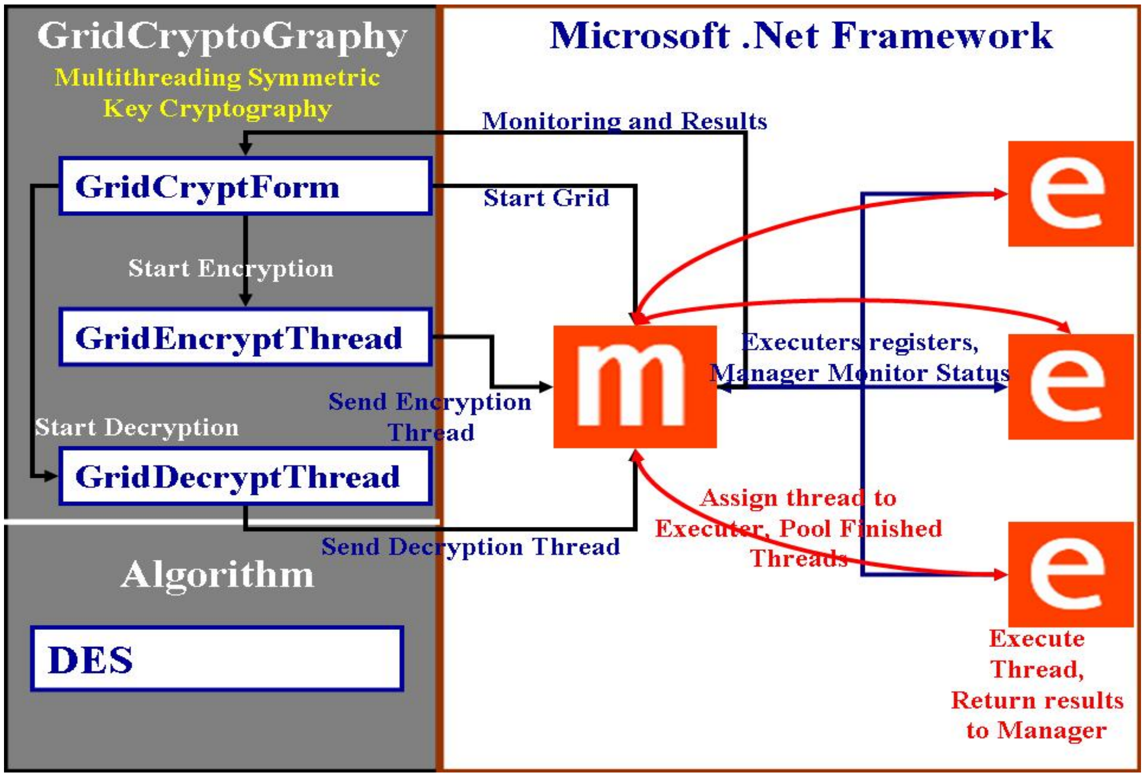


Fig.2. GridCryptoGraphy Architecture

Multithreading Symmetric key Cryptography In our design of multithreaded symmetric key cryptography, parallelization is being carried to process the data that need to be encrypted. We use the Task-Farming (master-slave) model for execution and principles of SPMD (Single Program Multiple Data) model for application parallelization. The effect of this parallelization method is that we have to find a way to divide the files into several blocks so that the process can be done in parallel on each block of data.

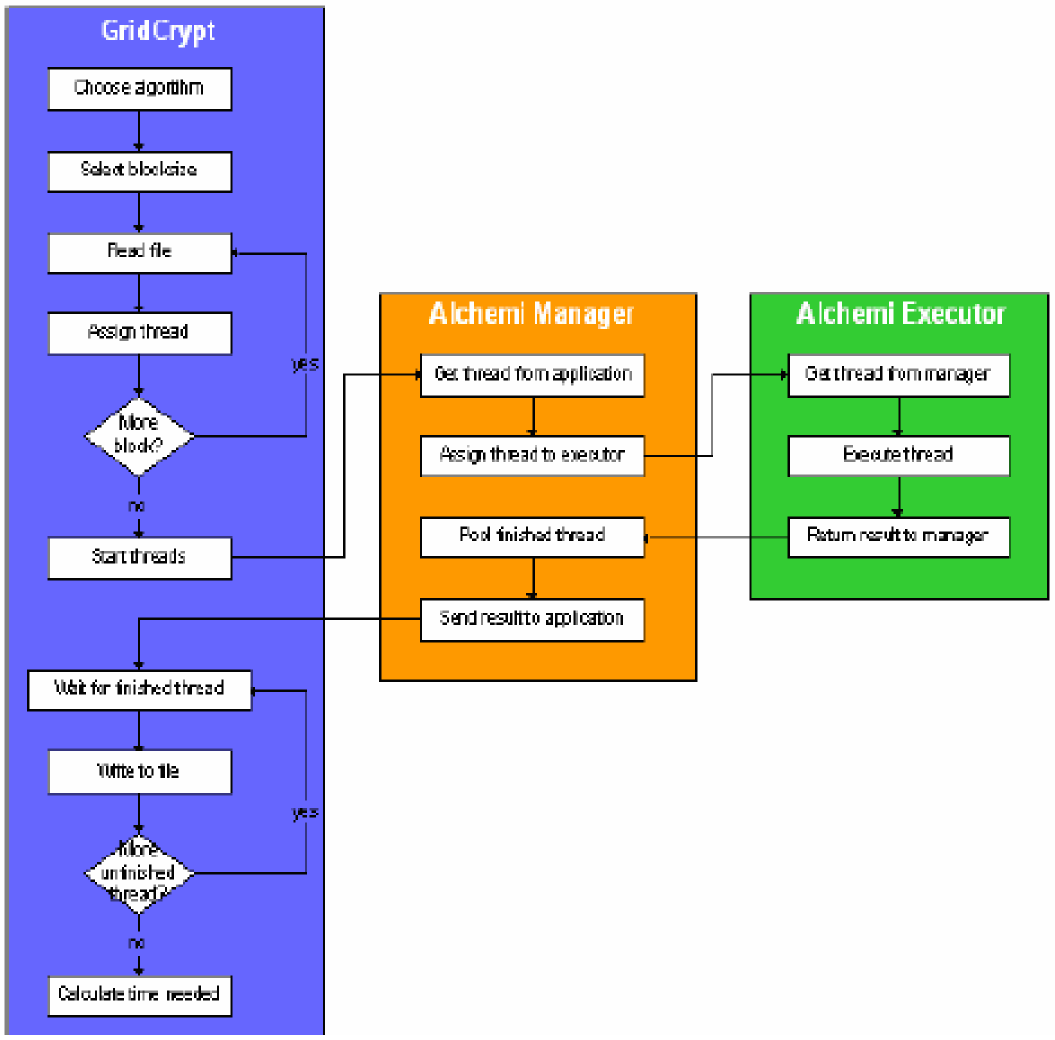


Fig. 3. GridCryptoGraphy flow of process.

We divide the raw file into several blocks based on the configuration form. An analysis of the DES algorithm reveals that the size of the block to be able to run each encryption algorithm should be divisible by 8. This is because in the DES algorithm the encryption was done 64 bit (8 byte) at a time, so if the block is not divisible by 8 the encryption algorithm that we implement will simply pad the block so that it will be divisible by 8. The padding is carried out on the last block of the file.

The flow of GridCryptoGraphy program is shown in Fig.3. It separates the input file into several parts in order to parallelize the encryption process. After the separation of the input file, each part of the file (block) is assigned to thread including the last block whose size is the remainder of the predefined block size.

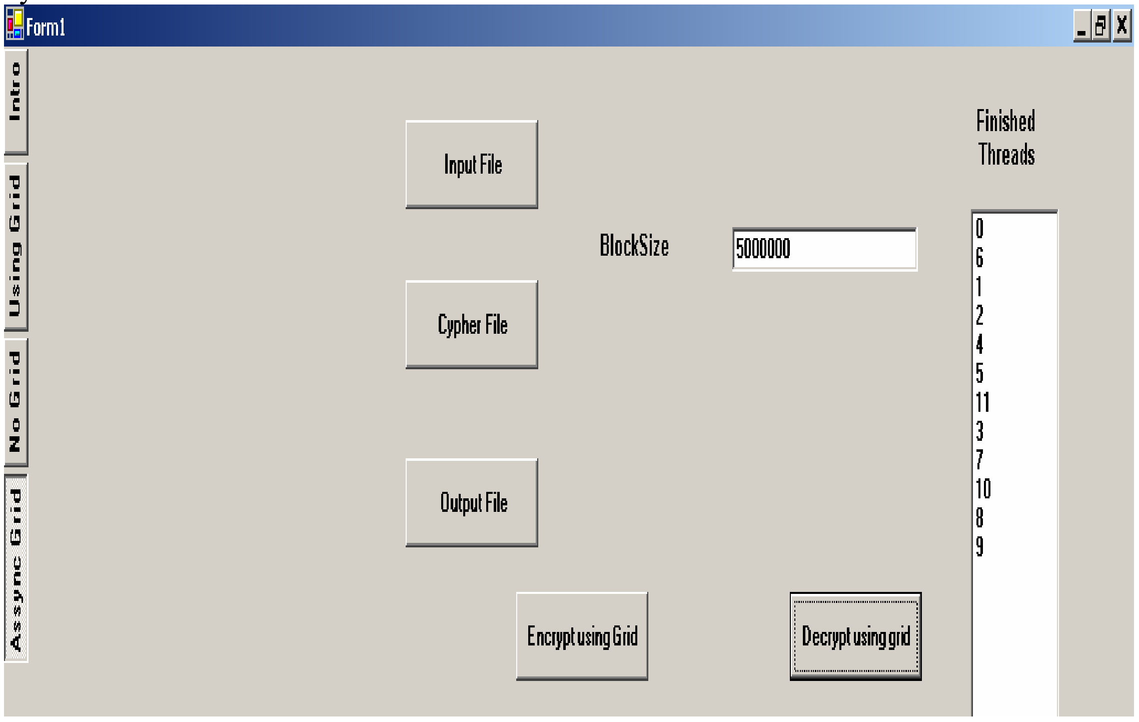
The file separation process is done by reading the file sequentially according to the block size. After the threads return with the encrypted result, GridCryptoGraphy writes the encrypted data to a random access file according to the index that is carried within the thread.

4 PERFORMANCE EVALUATION

The user interface of GridCryptoGraphy is shown in Fig.4; it displays an instance of the application during DES encryption on a large file. The GridCryptoGraphy program in Fig.4 monitors each of the threads it sends to the Alchemi manager and returns successfully finished back to the program. In the figure, we can see the monitoring panel at the right portion of the user interface. We can observe that the threads are submitted asynchrony to the random access file. The id of the finished thread is written in order so we can observe that thread no 6 finished and written to the random file before thread no 1 and 2.

If we use an input file of size (56610116 bytes) after encryption using block size (5000000) splits to 12 working unit (No of blocks) i.e. (11 blocks multiply 5000000 bytes block size) + (1 last block of size 1610116 bytes). Note that larger the block size configuration for the file split process means lesser number of work units so when using a block size of 5-mega bytes it yields to 12 work units but when using a block size of 1-mega bytes it yields to 57 work units .

In addition, as we stated before in section 3 that the DES algorithm reveals that the size of the block to be able to run each encryption algorithm should be divisible by 8. This is because in the DES algorithm the encryption was done 64 bit (8 byte) at a time, so if the block is not divisible by 8 the encryption algorithm that we implement will simply pad the block so that it will be divisible by 8. The padding is carried out on the last block of the file, so since the last block size is (1610116 bytes) which when we divide by 8 the result will be (201264.5) in which we observe that it is not divisible by 8 and we have 0.5 factor which yields to 4 extra bytes, so the algorithm pads the last block by 4 bytes so the cypher file size after encryption will be (56610120 bytes) which is equal to the actual input file size(56610116 bytes) + 4 bytes.



BlockSize 5000000

Finished Threads:
0
6
1
2
4
5
11
3
7
10
8
9

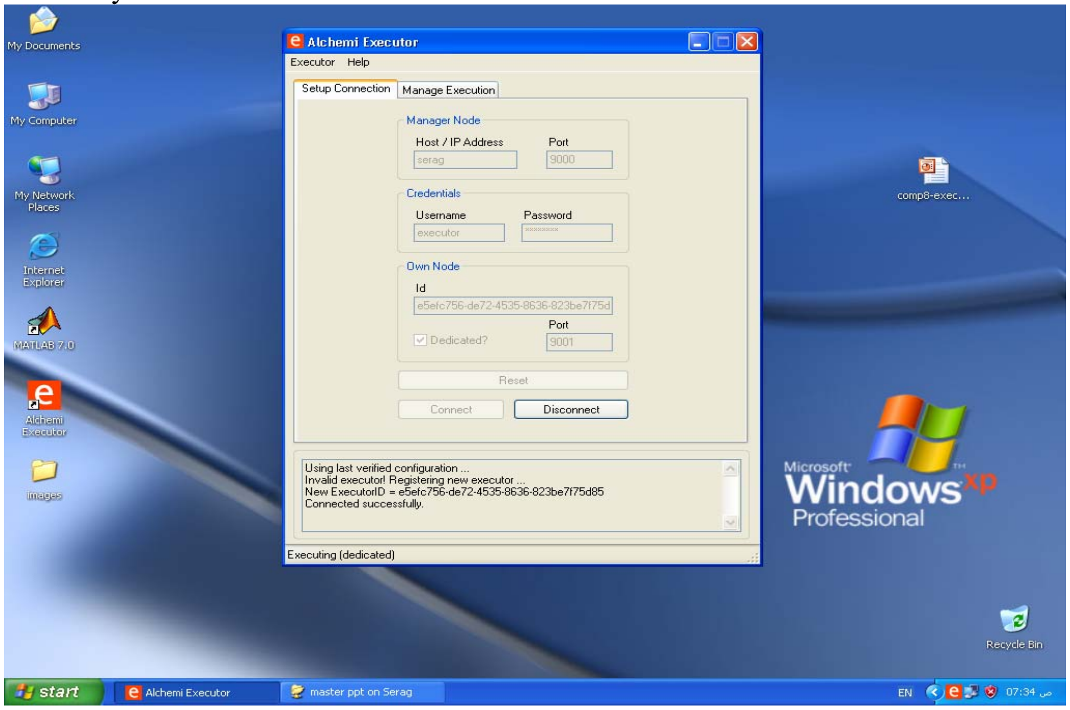
Fig.4. GridCryptoGraphy at runtime (monitoring of finished threads).

4.1 Runtime Comparison

We have done a runtime comparison for the GridCryptoGraphy application using 6 executor nodes as shown in Fig.5 and Fig.6 each with the same specification of:

Computer: Intel® Pentium®4 CPU 1.60 GHz, 128 MB of RAM.

System: Microsoft Windows XP Professional Version 2002 Service Pack 2.



Manager Node: Host / IP Address: serag; Port: 9000

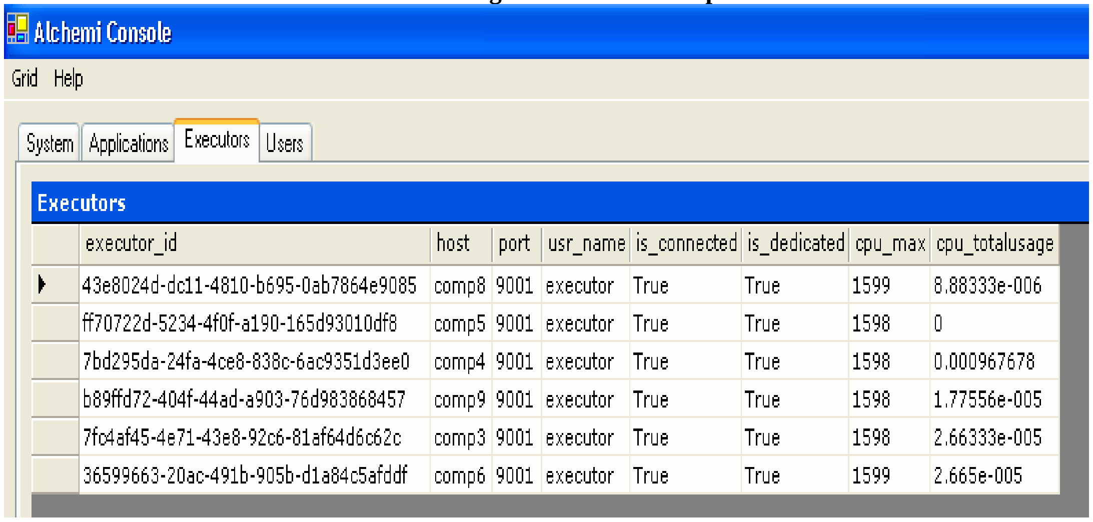
Credentials: Username: executor; Password: ********

Own Node: Id: e5efc756-de72-4535-8636-823be7f175d; Port: 9001; Dedicated? ✓

```
Using last verified configuration ...
Invalid executor! Registering new executor ...
New ExecutorID = e5efc756-de72-4535-8636-823be7f175d85
Connected successfully.
```

Executing (dedicated)

Fig.5. Executer desktop.



Executors

	executor_id	host	port	usr_name	is_connected	is_dedicated	cpu_max	cpu_totalusage
▶	43e8024d-dc11-4810-b695-0ab7864e9085	comp8	9001	executor	True	True	1599	8.88333e-006
	ff70722d-5234-4f0f-a190-165d93010df8	comp5	9001	executor	True	True	1598	0
	7bd295da-24fa-4ce8-838c-6ac9351d3ee0	comp4	9001	executor	True	True	1598	0.000967678
	b89ffd72-404f-44ad-a903-76d983868457	comp9	9001	executor	True	True	1598	1.77556e-005
	7fc4af45-4e71-43e8-92c6-81af64d6c62c	comp3	9001	executor	True	True	1598	2.66333e-005
	36599663-20ac-491b-905b-d1a84c5afddf	comp6	9001	executor	True	True	1599	2.665e-005

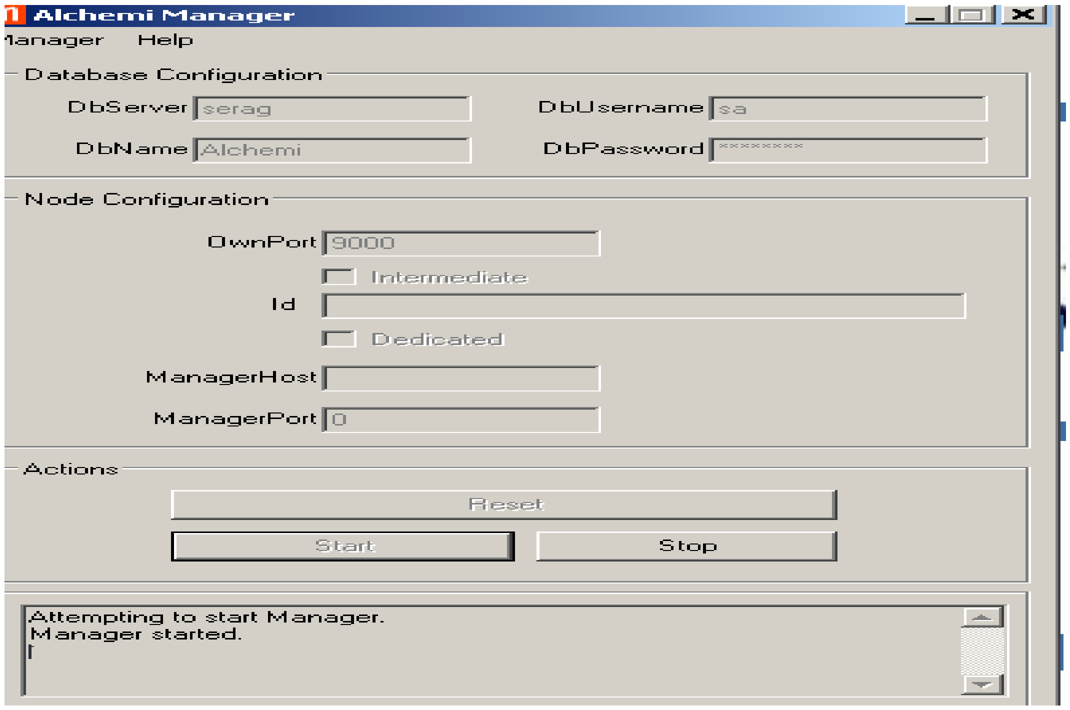
Fig.6. Using 6 executers.

All these nodes were interconnected over a shared LAN network of 100 Mbps. The Alchemi manager in Fig.7 was installed on a separate computer together with SQL Server 2000 and has the following specification:

Computer: Intel® Pentium®4 CPU 3.00 GHz, 504 MB of RAM.

System: Microsoft Windows Server 2003 Standard Edition.

The executions of the GridCryptoGraphy application run on the same computer with the manager.

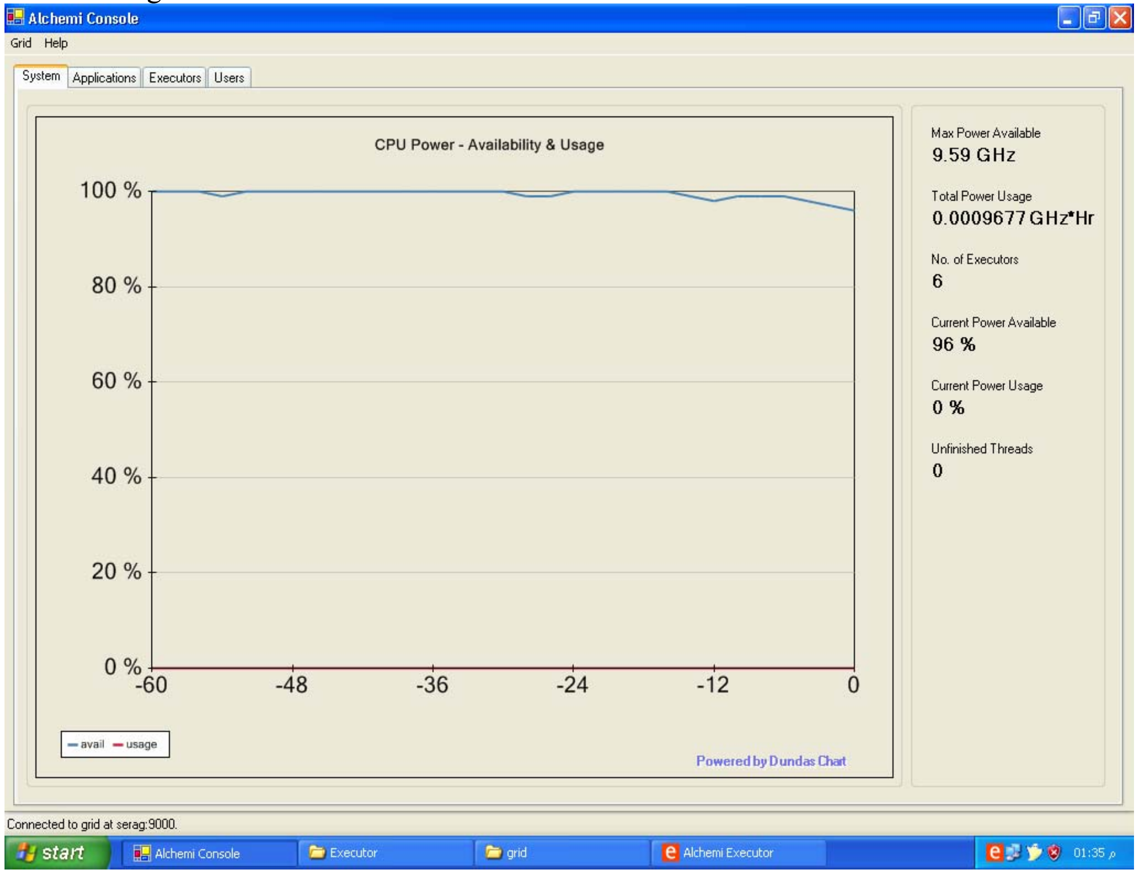


DbServer serag
DbUsername sa
DbName Alchemi
DbPassword *********
OwnPort 9000
Id
ManagerHost
ManagerPort 0

```
Attempting to start Manager.
Manager started.
```

Fig.7. The manager.

We monitored the CPU usage and the threads execution details using the Alchemi console as shown in Fig.8.



Max Power Available
9.59 GHz
Total Power Usage
0.0009677 GHz*Hr
No. of Executors
6
Current Power Available
96 %
Current Power Usage
0 %
Unfinished Threads
0

Connected to grid at serag:9000.

Fig.8. Monitoring the CPU usage & thread execution details.

The encryption and decryption experiments were conducted on files of size 9645200 bytes (approximately 10 MB), 56610116 bytes (approximately 57 MB) and 104858112 bytes (approximately 105 MB) with different block sizes. For each file the encryption and decryption was carried on 1, 2,3,4,5 and 6 executer nodes. The performance results of the 105 MB file size experiment will be shown next.

In all experiments, there was a reasonable performance improvement when 2, 3 and 4 executors were used. This gain is not linear. Although there was a reasonable performance improvement when up to 4 executors were used, there was drop in performance gain when number of executors was increased as shown next. This is due to various overhead factors including (a) involvement of large datasets with low computation to communication ratio, (b) existence of serial processing component (file splitting and collating results), (c) the use of slow and shared network, and (d) the overhead of the distributed execution environment (e.g., distribution of executable, initiation of execution on a remote node, and management of threads).

Results of a video file (104858112 bytes) size

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 and 6 executor nodes. The performance results of these experiments are shown in Table 1 and Fig.9, Fig.10, Fig.11 and Fig.12.

Table 1. Performance results of (104858112 bytes) file size experiment.

Block size / No of Executer	***1000000(1 Mega)min:sec***		***5000000(5 Mega)min:sec***		***10000000(10 Mega)min:sec***	
	Encryption	***Decryption***	***Encryption***	***Decryption***	***Encryption***	***Decryption***
1	02:11.906	01:15.688	05:04.984	02:28.219	09:45.859	03:56.422
2	01:40.375	00:56.266	03:10.875	01:17.531	05:25.813	02:47.594
3	01:35.016	00:55.625	02:33.844	00:56.156	04:20.172	02:00.953
4	01:32.750	01:12.109	02:18.391	01:13.469	03:47.406	01:39.453
5	01:59.500	01:55.344	05:27.266	01:41.641	04:11.578	02:01.469
6	02:35.688	01:57.844	06:27.188	03:10.109	06:24.813	02:56.422

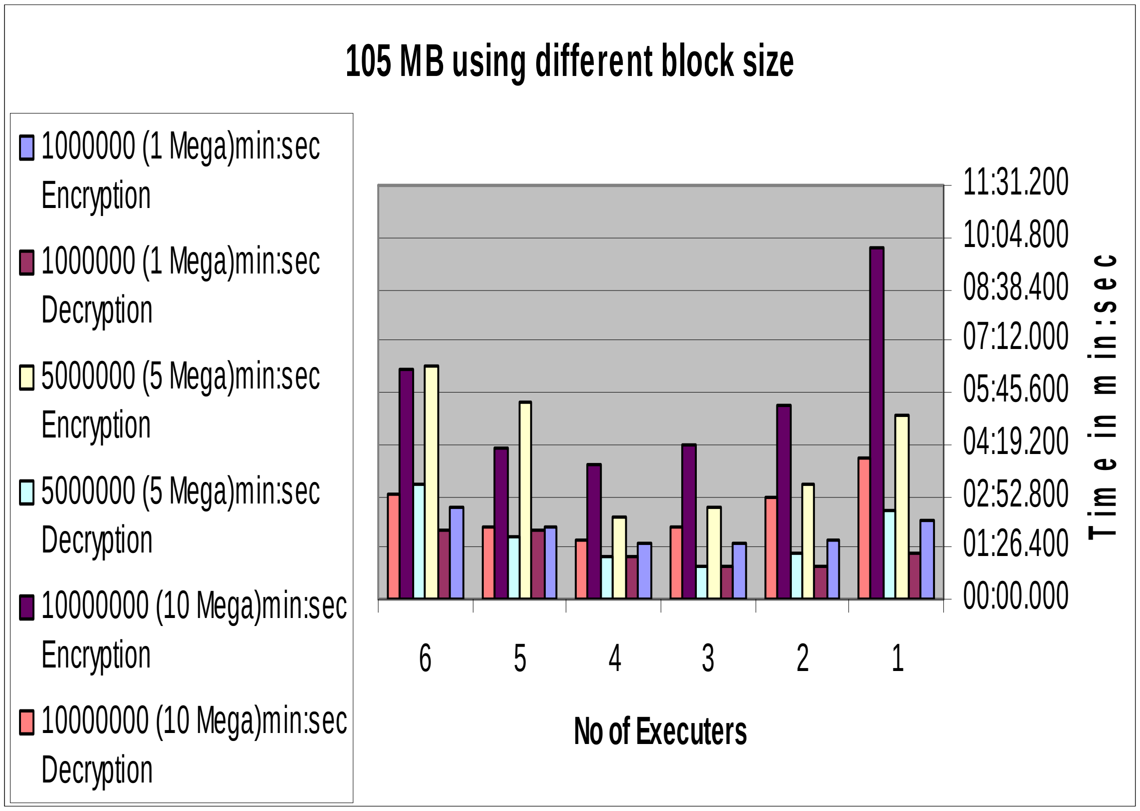


Fig.9. Result graph of (104858112 bytes) file size with different block sizes.

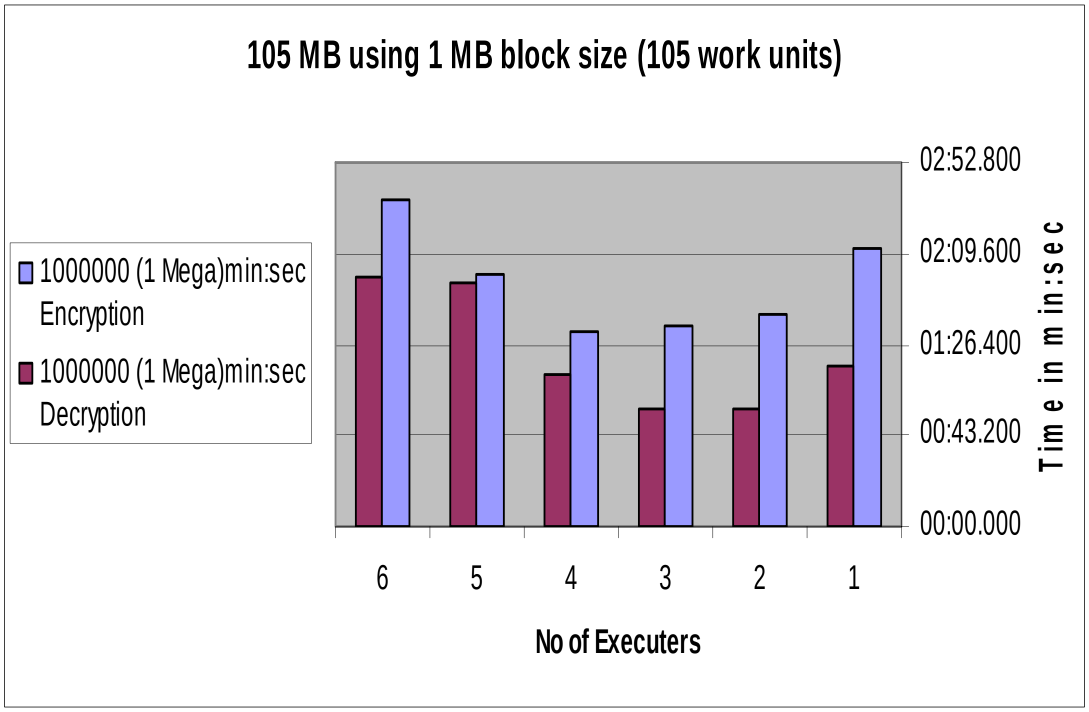


Fig.10. Result graph of (104858112 bytes) file size with 1 Mega block sizes.

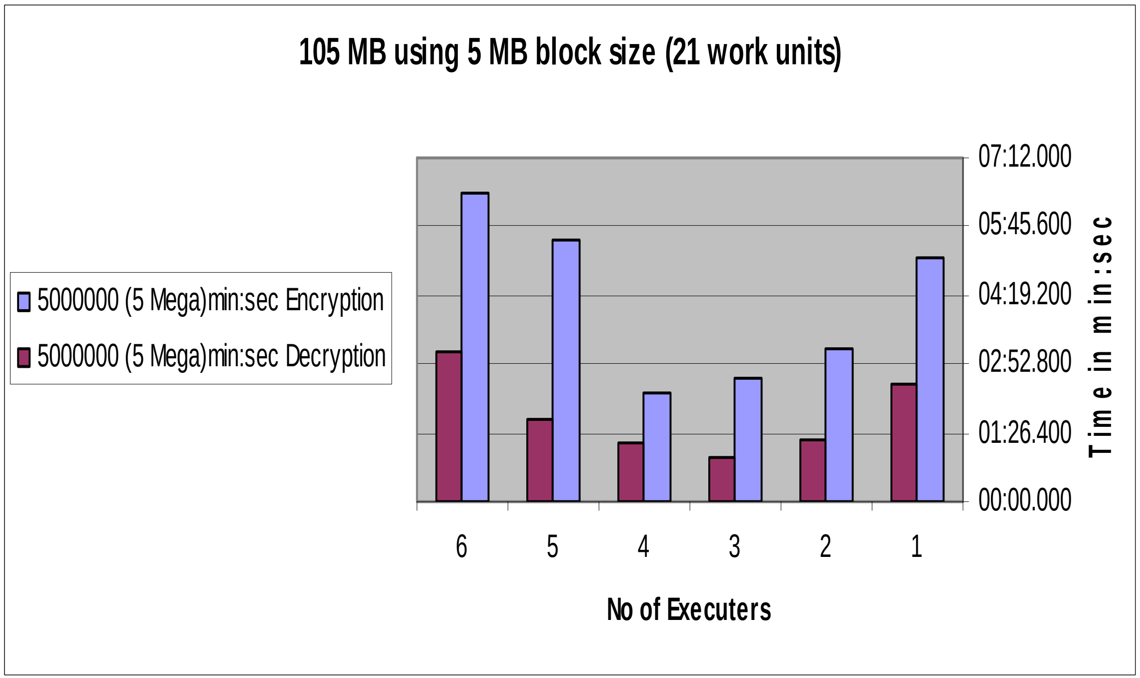


Fig.11. Result graph of (104858112 bytes) file size with 5 Mega block sizes.

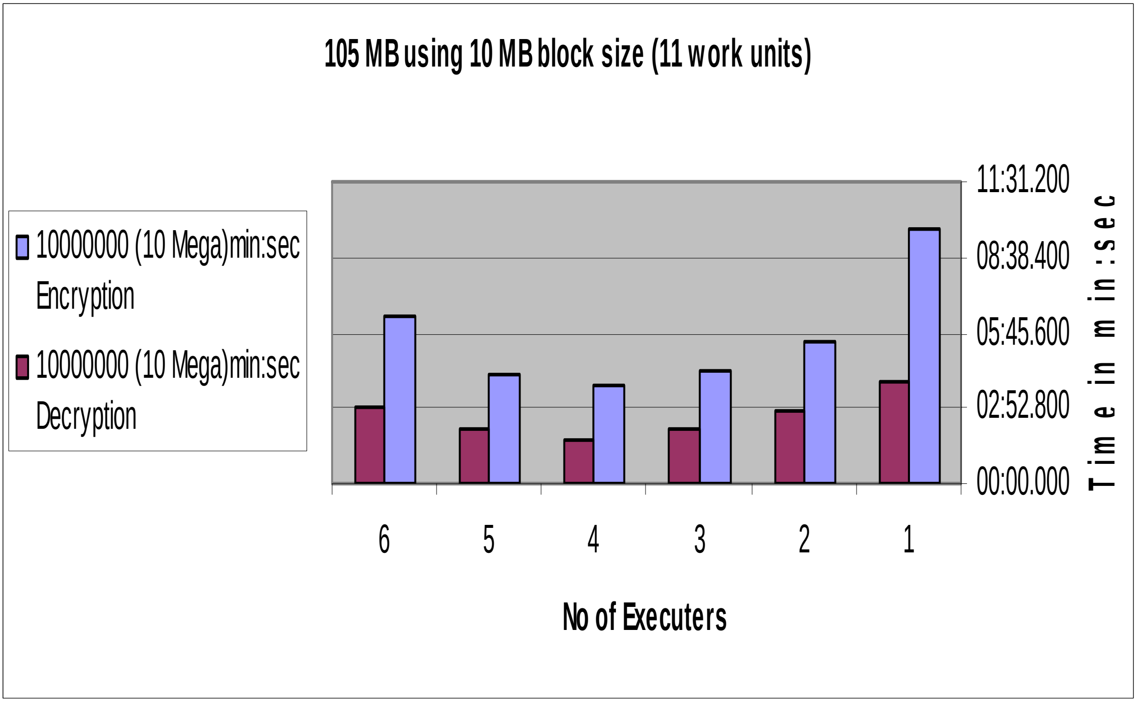


Fig.12. Result graph of (104858112 bytes) file size with 10 Mega block sizes.

A file copy test between two computers in the network for 110 MB file took an average of 20 seconds, which indicates effective usable bandwidth of 5.5 MB/sec. in our shared network. Considering the data transfer between the GridCryptoGraphy application and executor nodes is via the Alchemi manager, data for each work unit has to travel across the network 4 times. Thus, the file transfer overhead alone contributes about 60-70% of the processing time in this experiment.

The use of a faster network such as Gigabit Ethernet and faster storage systems will help minimize the overhead. In addition, although this overhead can be approximately halved by bypassing the manager and transferring date files between the user host and executors directly, it violates the current Alchemi security model. Now we are updating the GridCryptoGraphy application to make every executer access the file directly without transferring data files, this will greatly enhance the performance.

5 CONCLUSION AND FUTURE WORK

In this paper we show the performance of the GridCryptoGraphy application that was implemented using Alchemi to encrypt and decrypt large files using DES algorithm, there was an increase in performance over the single processor, but the performance improvement is limited by the I/O and communication overhead. The use of high performance networks can enhance performance. Another way to increase performance is via transferring the data directly between the user application and executers.

We are now updating the GridCryptoGraphy application to directly transfer data to executers and trying to use high performance network.

Also we are designing a new application using Alchemi to brute force attack the DES algorithm.

ACKNOWLEDGEMENT

This work was supported by the Military Technical College.

References

[1] Abbas, A. GRID COMPUTING: A Practical Guide to Technology and Applications.

[2] Grid Computing Info Centre (GRID Infoware) webpage. URL http://www.gridcomputing.com.

[3] Global grid forum webpage. URL http://www.gridforum.org.

[4] Foster, I, Kesselman, C, and Tuecke, S. The anatomy of the grid: Enabling scalable virtual organizations. International Journal of Supercomputer Applications.

[5] Foster, I, Kesselman, C, Jeffrey M. Nick and Tuecke, S. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration.

[6] Fran Berman, Geoffrey Fox, Tony Hey. Grid Computing: Making the Global Infrastructure a Reality.

[7] Foster, I. and Kesselman, C. (eds.). The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, 1999.

[8] Foster, I., "What Is the Grid? A Three Point Checklist," GRIDtoday, 1(6), July 22, 2002. URL: http://www.grigtoday.com

[9] Foster, I. and Kesselman, C. (eds.). The Grid2: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, 2004 (second edition).

[10] William Stallings, Ph.D. Network and Internetwork security principles and practice.

[11] Luther, A. Buyya, R. Ranjan, R. Venugopal, S. Alchemi: A .Net-based Grid computing Framework and its Integration into Global Grids.

[12] Luther, A. Buyya, R. Ranjan, R. Venugopal, S. Peer-to-Peer Grid computing and a .Net based Alchemi Framework.

[13] Luther, A. Buyya, R. Ranjan, R. Venugopal, S. (2005) Alchemi: A .Net-based Enterprise Grid computing System.

[14] Luther, A. Buyya, R. Nadiminti, K. (July 2005) Alchemi: A .NET-based Enterprise Grid System and Framework. User Guide for Alchemi 1.0.