ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

IJIRSET

International Journal of Innovative Research in Science, Engineering and Technology

Volume 3, Special Issue 3, March 2014

2014 International Conference on Innovations in Engineering and Technology (ICIET'14)

On 21st & 22nd March Organized by

K.L.N. College of Engineering and Technology, Madurai, Tamil Nadu, India

Quantum Algorithm: A Classical Realization in High-Performance Computing Using MPI

Donald Jefferson Thabah N, Balachandran K, Anirban Roy, & John Kiran A

Dept of Computer science & Engineering, Christ University Faculty of Engineering, Bangalore, India.

Dept of Computer science & Engineering, Christ University Faculty of Engineering, Bangalore, India.

Dept of Computer science & Engineering, Christ University Faculty of Engineering, Bangalore, India.

Dept of Computer science & Engineering, Christ University Faculty of Engineering, Bangalore, India.

Abstract— Quantum computing has been an attractive method adapted for increasing the computational speed and it is governed by the laws of quantum mechanics. Quantum computers are not limited to two states as compared with classical systems, but they encode information as quantum bits or qubits. A qubit represents atoms, ions, photons or electrons and their respective control devices that are working together to act as computer memory and processors. The power of quantum computing lies in its multi-state representation which makes it million times more powerful than classical computer systems. Quantum computers also use another aspect of quantum mechanics known as entanglement, which is a property where multiple objects existing in the states that can be linked together.

There exists a standard/specification of message passing library, known as message passing interface (MPI) that can be used for achieving high-performance computing. The goal of MPI is to provide a standardized framework for writing message-passing programs which can be used for programming systems that require distributed processing. The superposition principle used by the quantum algorithm can be simulated in the MPI environment with a cluster of computers or multi-CPU systems. This paper focuses on exposing the strength of quantum algorithm in high performance computing by using MPI, and also on comparing the sequential and parallel execution of programs on the basis of execution time and CPU utilization.

Keywords—qubit, superposition, entanglement, vector space, ket, quantum Fourier transform, Shor's algorithm, message passing interface.

I. INTRODUCTION

Numerous non-polynomial hard problems exist in the field of computation which cannot be solved by normal classical computers [1]. A quantum system offers a platform to solve such type of problems [2-3]. Although the miniaturization of computer chips will ultimately result in reducing their dimension to the scale of quantum mechanical prediction, where the mechanisms are governed by the laws of quantum physics, so far, the development of computational machines is still based on the laws of classical physics. A classical computer uses a bit that is set either to 0 or 1, and it does computing using one bit at a time until a program is completed. Whereas, a quantum computer uses a qubit, which is simply a superposition of 0 and 1, and enables each step to run all variations of a bit at the same time, giving them exponentially higher processing power as compared with classical computers. Quantum objects display both particle-like and wave-like features at the same time, and the characteristic property of a wave that is useful in computation is the superposition. This means we will have a "single instruction multiple data" architecture, i.e., having multiple signals at the same position at the same time [4] which will lead to enhanced computational performance of the system.

Recently, there have been reports on work on quantum computing that aims at increasing the computation speed [5-8]. Here we report on our work on gaining speed in computing with specific cases such as factoring huge numbers. While there are different algorithms like Grover's search algorithm, Simon's algorithm and Deutsch-Jozsa algorithm [9, 10] for computation, we use

Shor's algorithm which is proved to be one of the best algorithms for finding factors of huge integers [11]. We show a classical simulation of Shor's quantum algorithm, where the superposition principle is achieved by allocating the states to multiple systems present in a cluster. The execution time and the processor utilization in parallel computing are measured and are compared with those in sequential implementation.

II. QUANTUM COMPUTING

A. Qubit

The basic unit of information in quantum computation is qubit, which is a quantum bit. Like a bit, a qubit can also be represented in two states labelled as |0> and |1>, and called as a vector or a ket. This notation is very convenient because it expresses that the state of a quantum system is a vector and at the same time denotes the physical quantity of interest (energy level, position, spin, polarization, etc). A qubit can exist in a superposition state, which is a linear combination of |0> and |1>.

A qubit can be represented as,

$$|\psi\rangle=\alpha|0\rangle+\beta|1\rangle, \quad (1)$$

where α and β are complex amplitudes, $|\alpha|^2+|\beta|^2=1$, and $|\alpha|^2$ and $|\beta|^2$ give the probability of finding |ψ> in |0> or |1> state, respectively. A qubit can also be represented as a vector, for example,

$$|\psi\rangle=\begin{pmatrix}\alpha\\ \beta\end{pmatrix} \quad (2)$$

The vector shown above is in a two dimensional complex vector space and it is normalized to 1 [9-12].

B. Quantum Gates

In a quantum computer, information is processed using gates and quantum gates are unitary operators which are reversible [13]. A unitary operator U is the one where the adjoint is equal to the inverse, meaning

$$U^{\dagger}=U^{-1} \quad (3)$$

In addition, if H is Hermitian operator, then $U=e^{iHt}$ is unitary. A quantum gate with *n* inputs and *n* outputs can be represented by a matrix of degree 2^n.

C. Operator

An operator is a mathematical rule that can be applied to a function to transform it into another function. For instance, an operator is applied to a ket |ψ> to transform it into another ket |Ø>.

E.g., $A|\psi\rangle=|\emptyset\rangle$ where *A* is a unitary operator. (4)

An operator can also act on bra as well, i.e., $\langle u|A=\langle v|$. The product of ket and bra can be written as |u><v| (outer product), which is also known as the projection operator.

D. Measurement

Measurement is a projection of the state to the standard basis. Given a state | ψ> in a two dimensional Hilbert space with orthogonal bases (inner product states |0> and |1> is zero) |0> and |1>, the projection of the state |ψ> to the standard basis |0> or |1> is known as measurement. The probability that a state |ψ> being projected to |0> or |1> is given by the square of the complex amplitudes associated with states |0> and |1>.

E. Quantum Fourier transform

Quantum Fourier transform (QFT) is a quantum implementation of the discrete Fourier transform. It transforms the *n* qubit state vector $|\alpha\rangle=\alpha_0|0\rangle+\alpha_1|1\rangle+\cdots\cdots\alpha_n|n\rangle$ into its Fourier transform $|\beta\rangle=\beta_0|0\rangle+\beta_1|1\rangle+\cdots\cdots\beta_n|n\rangle$, and a measurement on |β> will return only one of its *n* components. The QFT has the following properties:

a. QFT is unitary
b. Linear shift
c. Period/Wavelength Relationship

These are the properties that are needed to develop the Shor's algorithm [10]. The discrete quantum Fourier transform can be expressed as a unitary change of basis,

$$\sum_x f(x)\,|\,x\rangle=\sum_y\Big(\frac{1}{\sqrt{N}}\sum_x e^{2\Pi ixy/N} f(x)\Big)\,|\,y\rangle \quad (5)$$

Writing integer *x* and *y* in a binary notation, e.g. $x=x_{n-1}.2^{n-1}+\ldots..+x_1.2+x_0$, the non-trivial fractional part of the exponent can be written as Fraction$(xy/N)=y_{n-1}(.x_0)+y_{n-2}(.x_1x_2)+\ldots+y_0(.x_{n-1}\ldots x_0)$. Then the unitary rotation of the QFT factorizes as

$$|\,x\rangle=\frac{1}{\sqrt{N}}\sum_y e^{2\Pi ixy/N}\,|\,y\rangle=\frac{(|0\rangle+e^{2\Pi i(.x_0)}|1\rangle)}{\sqrt{2}}\frac{(|0\rangle+e^{2\Pi i(.x_1x_0)}|1\rangle)}{\sqrt{2}}\ldots\ldots\frac{(|0\rangle+e^{2\Pi i(.x_{n-1}\ldots.x_0)}|1\rangle)}{\sqrt{2}}, \quad (6)$$

where the sum over y has been expanded in terms of the two values of each of its *n* bits. The factorization has converted the sum over *N* different values of *y* to a product of *n* single qubit notation. Complete factorization of the transform provides the maximal *O(N/log N)* gain for the algorithm. Now the period finding problem requires only one result, i.e., the period from multiple evaluation of *f(x)*, which is possible with maximal quantum superposition of *x* values and a single run of QFT. This period finding using QFT gains another factor of *N/logN* in complexity [4, 13].

F. Period/wavelength relationship

Suppose the function *f* is periodic with a period *r*. For example see Fig. 1.

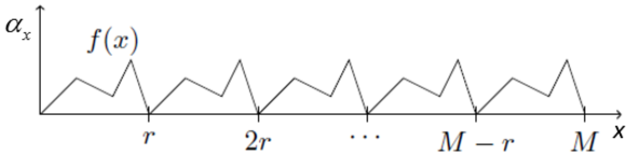


Fig. 1 Periodic function

Then, $\hat{f}$ (the Fourier transform of *f*) is supported only on multiples of *M/r*. Thus $\hat{f}(x)=0$ unless $x=kM/r$. If *r* is the period of function *f*, then we can think of *M/r* as the wavelength of *f*. The wider the range of function is the sharper the range in the Fourier on the domain, and vice versa.

For the Shor's algorithm, the period function is written as

$$f(j)=\begin{cases}\sqrt{\frac{r}{M}} & if\ \ j\equiv 0(\bmod r)\\ 0 & otherwise\end{cases} \tag{7}$$

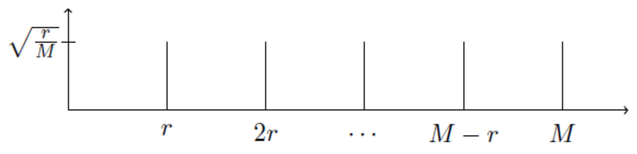


Fig. 2 Input superposition of periodic function

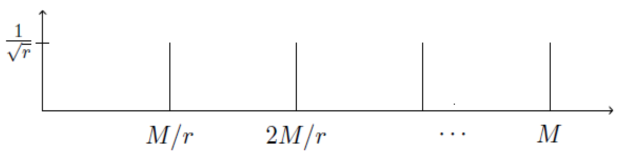


Fig. 3 Quantum Fourier transform of periodic function

The Fourier Transform of equation (7) is

$$f\wedge(x)=\sqrt{\frac{r}{M}}\sum_{i=0}^{\frac{M}{r}-1}\omega^{rij}\ , \tag{8}$$

$where\ \omega=e^{2\Pi ri/M}$.

Further simplification of equation 3 will lead to

$$f(j)=\begin{cases}\sqrt{\frac{r}{M}}, & if\ \ j\equiv 0(\bmod r)\\ 0, & otherwise\end{cases}\ , \tag{9}$$

, where ω is called the primitive M/r root of unity.

III. DEMONSTATION OF SHOR'S ALGORITHM

A. Breaking the RSAencryption

RSA (Ron Rivest, Adi Shamir and Leonard Adleman) encryption is based on finite groups, where the group operation is based on multiplication modulo method for some fixed integer N [14]. It is implemented by choosing two huge prime numbers p, q and a large encoding c that has no common factor in common with $(p-1)\times(q-1)$. Then inverse can be calculated as $cd\equiv 1(mod(p-1)\times(q-1))$ where d is some integer i.e. c and $p\times q$ are known publicly and d is private. Hence to encrypt a message M we perform the operation $b\equiv M^c(mod\ (p\times q))$ and to decrypt b we perform the operation $M\equiv b^d(mod\ (p\times q))$

It can be proved that it is sufficient to decrypt the encrypted message b if we know the order of b. The order r of b divides the order $(p-1)\times(q-1)$ of G_{pq} , and c has no common factor with $(p-1)(q-1)$. Therefore,

$cd'\equiv 1\ (mod\ r)$ (10)

r divide $cd'-1$

$cd'-1=jr$

$cd'=1+jr$ (Here j, d' are some integer d' is the inverse of c) (11)

Therefore,

$b^{d'}\equiv M^{cd'}=M^{1+jr}=M(M^r)^j\equiv a(mod(p\times q))$ (12)

Hence, if the eavesdroppers manage to find the period $b\equiv M^c(mod(p\times q))$ then the eavesdroppers can reveal the original message M without knowing the value of d. Thus, the problem is reduced to quantum period finding [15, 16].

B. Using Shor's Algorithm for factorizing composite integers

We begin the quantum period finding by using the quantum computer in the following ways: say $N=p\times q$ and function $f(x)\equiv b^x\ (mod\ N)$ is the function for which we want to find the order.

1. Pick a number $q=2^{t_A}$ such that $N^2\leq q<2N^2$, where N is the number to factorize
2. Pick t_b, i.e the number of bits required to represent N.
3. Pick a random integer x that is co-prime to N
4. Initialize Register A (having t_A qubits) to be |0> and Register B (having t_B qubits) to |0>, i.e.,

$|\Psi\rangle=|0\rangle|0\rangle$ (13)

5. Then apply Hadamard gate [9] (Hadamard gate transforms the qubits into a linear combination of vectors that is superposition) to all the qubits in the register1

$$H=\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix},\ \text{Hence}\ |\psi\rangle\frac{1}{\sqrt{2^{t_A}}}\sum_{x=0}^{2^{t_A}-1}|\,x\rangle_{t_A}|\,0\rangle_{t_B} \tag{14}$$

6. Now modular exponentiation is applied, i.e. the operation $|z\rangle|y\rangle=|z\rangle|f(x)\rangle$. We can measure (measurement consist of performing a certain test on each qubit with the outcome of either 0 or 1, and it is carried out by 1-qubit measurement gate) the t_B Qubits of register2. If the measurement value is f_0, then the generalized Born rule tells us that the state of the t_A-*qubits* register1 can be taken to be

$$|\,\Psi\rangle=\frac{1}{\sqrt{m}}\sum_{k=0}^{m-1}|\,x_0+kr\rangle_{t_A}|\,f_0\rangle_{t_B} \tag{15}$$

x_0 is the smallest value of x $(0\leq x_0<r)$ for which $f(x_o)=f_0$ and m is the smallest integer for which $mr+x_0\geq 2^{tA}$, so $m=\lceil 2^n/r\rceil$ or $m=\lceil 2^n/r\rceil+1$,

If we can produce a small number of identical copies of the above equation (15) the job is done. The measurement in the computational basis would give random value of x_0+kr, and the difference between the results of pairs of measurement on such identical copies would give us a collection of random values of r from which r can be extracted easily. But this possibility is ruled out because all we can extract is a single value of x_0+kr for unknown random value x_0. This is because of displacement by an unknown random value x_0, which prevents any information about r from being extracted in a single measurement. Hence, unitary quantum Fourier transform comes into play which transforms x_0 dependence into a harmless overall phase factor [17-19].

7. Apply the quantum Fourier transformation on register1.

$$|\,\Psi\rangle=\frac{1}{\sqrt{tm}}\sum_{c=0}^{t_A-1}\sum_{d=0}^{m-1}e^{(2\Pi ic(x_0+kr)/t_A}|\,c,\,f0\rangle$$
$$=\sum_{c=0}^{t_A-1}\frac{e^{2\Pi icx_0}/t_A}{\sqrt{t_Am}}\sum_{d=0}^{m-1}e^{(2\Pi ickr)/t_A}|\,c,\,f0\rangle \tag{16}$$

8. Measure register1. We observe register1 to be in the state |c> with probability

$$pr(c) = \frac{1}{tm} \left| \sum_{d=0}^{m-1} e^{2\Pi ickr/q} \right|^2 \quad (17)$$

9. Continued Fraction Convergent

$\frac{c}{t_A}$, we stop the convergent before the denominator exceeds *N* and denote the value as *r1*, possible values of *r* are the multiples of *r1* such that $f(x) \equiv b^x \pmod{N}$) is 1.

IV. SIMULATION AND RESULT

We here demonstrate the working of Shor's algorithm using a simple example of factorizing a number. The simulation was carried out on a cluster of three computers.

A. Classical simulation overview

1. Message passing interface (MPI)

The message passing interface (MPI) is a library specification for message-passing, proposed as a standard by broadly based committee of vendors, implementors and users [20]. It is designed for high performance computing on both massive parallel machines and on workstation clusters. In this paper, it is implemented on a clusters of workstations. MPICH is a high performance and a widely portable implementation of the MPI standard, and is used here which support of C++.

2. TORQUE Resource Manager

TORQUE (Terascale Open-Source Resource and QUEeu Manager) Provides control over bath jobs and distributed computing resources.

B. System and Cluster configuration

Three systems were used in a cluster, each running CentOS 6.4(Final). The following is the configuration used

1. Master node and two slave nodes.
2. MPICH3 with TOUQUE resource manager were used.
3. All systems have the same configuration with processor Intel(R) Core(TM) i3-2100 CPU @ 3.10GHz and 3Gb RAM.

C. Simulation environment

We present a realization of Quantum algorithm (Shor's algorithm) using MPI in a cluster implemented in C++. There are four restrictions for the Shor's algorithm when it is used for factorizing a number. They are

1. The number to be factorized should be greater than 15
2. The number to be factorized must be odd
3. The number to be factorized must not be a prime power

D. Results

Table I shows a comparison of the execution time taken by the cluster of computers in parallel and serial mode, respectively, for the chosen problem of factorization of given numbers. It is clear that the time taken in parallel execution is very less, and CPU utilization was 100 % (Figs. 5-7). These results are interesting, showing that quantum computing is able to increase the computational speed.

TABLE I
COMPARISON OF EXECUTION TIME IN SERIAL AND PARALLEL COMPUTING

Input integer	serial (ms) No of process 4	parallel (ms) No of process 8
15	10	10
21	20	10
35	40	20
77	340	150
91	1080	570
195	12070	4640
255	18830	6950
315	62330	35520
713	407470	51730

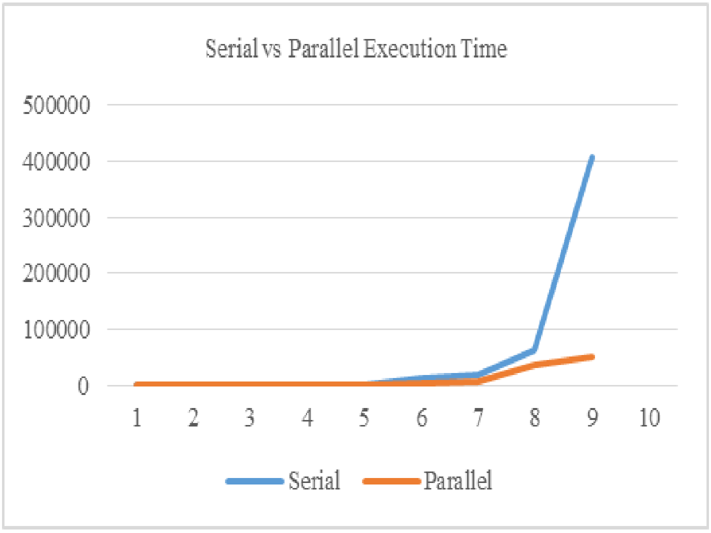
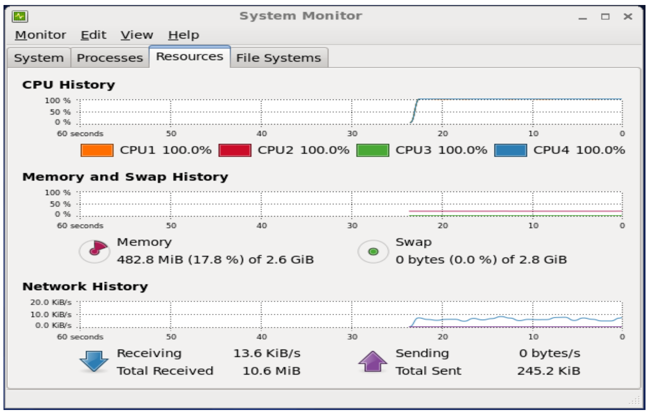


Fig. 4 Execution time in serial and parallel computing using one computer for serial and three computers for parallel execution

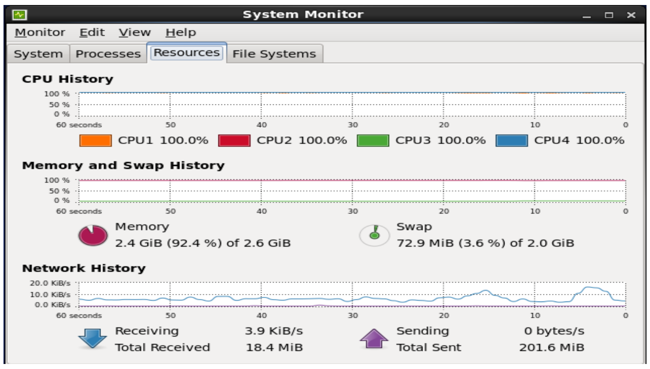


🟧 CPU1 100.0% 🟥 CPU2 100.0% 🟩 CPU3 100.0% 🟦 CPU4 100.0%

Memory 482.8 MiB (17.8 %) of 2.6 GiB
Swap 0 bytes (0.0 %) of 2.8 GiB

Receiving 13.6 KiB/s
Total Received 10.6 MiB
Sending 0 bytes/s
Total Sent 245.2 KiB

Fig. 5 CPU utilization of the computing node i.e. slave node 1271 is factorized (master)

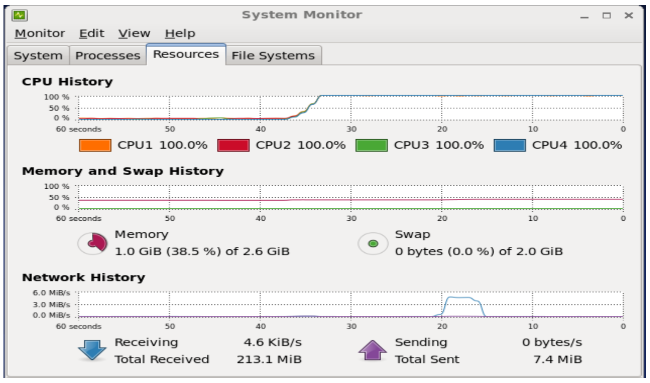


CPU1 100.0% CPU2 100.0% CPU3 100.0% CPU4 100.0%

Memory 2.4 GiB (92.4 %) of 2.6 GiB
Swap 72.9 MiB (3.6 %) of 2.0 GiB

Receiving 3.9 KiB/s
Total Received 18.4 MiB
Sending 0 bytes/s
Total Sent 201.6 MiB

Fig. 6 CPU utilization of the computing node i.e. slave node 1271 is factorized (slave 1)



CPU1 100.0% CPU2 100.0% CPU3 100.0% CPU4 100.0%

Memory 1.0 GiB (38.5 %) of 2.6 GiB
Swap 0 bytes (0.0 %) of 2.0 GiB

Receiving 4.6 KiB/s
Total Received 213.1 MiB
Sending 0 bytes/s
Total Sent 7.4 MiB

Fig. 7 CPU utilization of the computing node i.e. slave node 1271 is factorized (slave 2*)*

Table II demonstrates that by increasing the number of processes in a parallel computing one would achieve further increment in speed. Figure 8 graphically represents the processing time taken by the system of three computers for solving factorization problem. It is clear from the table II that the time consumed for factorizing small integers is more in the case of 64 processes, whereas it significantly gets reduced eventually when the integer becomes big, when compared with cases with 16 and 32 processes.

TABLE II.
COMPARISON OF EXECUTION TIME IN PARALLEL COMPUTING WITH DIFFERENT PROCESS NUMBER.

Input integer	**parallel (ms) No of process 16**	**parallel(ms) No of process 32**	**parallel (ms) No of process 64**
15	40	50	170
21	40	80	170
35	60	90	180
77	70	90	200
91	170	100	210
195	1160	400	350
255	1690	900	370
315	8030	2080	860
713	33160	9310	1610
1271	-	-	37090

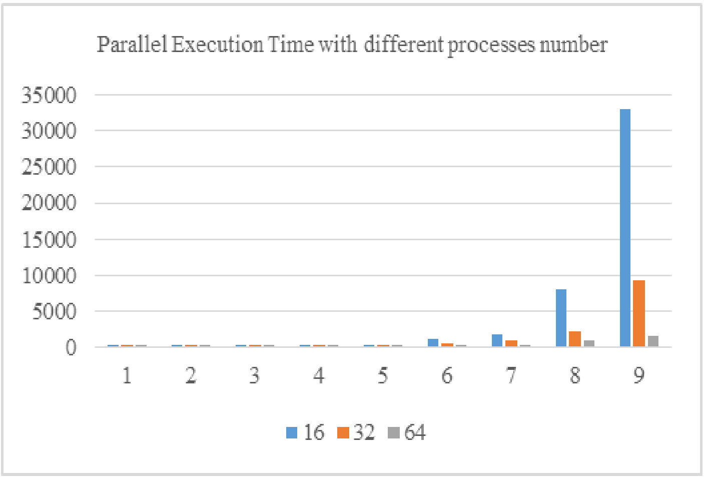


Fig. 8 Chart showing the execution time in parallel computing with different process number

The present work has considered solving factorization problem of integers using Shor's algorithm. One limitation encountered in this work is the representation of a huge integer by the existing register memory. The future work will consider solving this problem by dividing the quantum states into multiple systems that can represent huge numbers.

Shor's algorithm has been employed by various researchers in solving problem of factoring huge composite integers on different computational platforms like Sun, Enterprise4500 [21, 22]. A direct comparison of our work with the literature report is difficult at the moment due to varying system configurations involved in computation.

V. CONCLUSIONS

The article demonstrates the simulation of quantum computing using Shor's algorithm with an example of factorization of given integers. It is found that the parallel computing using a cluster of classical computers is effective in increasing the computational speed. It implies that upon increasing the number of classical systems further in a cluster, one may achieve a better speed for computing.

REFERENCES

[1] G.A.A. Mary, J.Naresh , C.Chellapan, *A Parallel Algorithms for Solving Factorization and Knapsack Problems*, IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 2, No 1, March 2012.

[2] M.R.S Aghaei, Z.A Zukarnian, A. Mamat, H. Zainuddin, *A Hybrid Architecture Approach for Quantum Algorithm*, Journel of Computer Science Vol.5, pp. 725-731,2009.

[3] T.P. Spiller, *Quantum Information Processing: Cryptography, Computation, and Teleportation*, Proceesings of the IEEE, Vol. 84, No. 12, Dec 1996.

[4] A. Patel, *Quantum Computation: Particle and Wave Aspects of Algorithms*, Resonance- Journal of Science Education, Vol 16, pp.821-835, 2011.

[5] S. Nakayama, P. Gang, I. Iimura, *Study of quantum parallel processing by adiabatic quantum computation in Bernstein-Vazirani problem*,13th international conference on parallel and distributed computing application and technologies, China, Dec 12-16, 2012.

[6] D.F.Walls Gerard J. Milburn, *Quantum Optics,* 2nd Ed., Springer, USA, 2010.

[7] L.M.K. Vandersypen, M. Steffen, G. Breyta, C. S.Yannoni, M. H. Sherwood, I. L. Chuang, *Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance*, Nature, Vol 414, pp.883-887, 2001.

[8] G.D. Paparo and M.A Martin-Delgado, *Google in a Quantum Network*, arXiv:1112.2079v2 [quant-ph], 11 jul 2012.

[9] D. McMahon, *Quantum Computing Explained*, Wiley-Interscience A John Wiley & Sons, Inc., USA, 2007.

[10] M.A. Nielsen & I.L. Chuang, *Quantum Computation and Quantum Information*, 10th Anniversary Edition, Cambridge University press, UK, 2010.
P.W. Shor, *Algorithms for quantum computation:discrete logarithms and factoring,* 35th symposium on Foundations of computer science, Santa Fe, pp 124-134, Nov 20-22, 1994.

[11] F. Tabakin, B. Julia-Diaz, *QCMPI: A Parallel Environment for quantum computing,* arXiv:0902.0699v1 [quant-ph] 4 Feb 2009.

[12] T.P. Spiller, *Quantum Information Processing: Cryptography, Computation, and Teleportation*, Proceesings of the IEEE, Vol. 84, No. 12, Dec 1996.

[13] W.Zhang, C Xu, F. Li, J. Feng, *A Period-finding Methos for Shor's Algorithm*, International Conference on Computational Intelligenc and Security, Harbin, China, 15-19 Dec 2007.

[14] N.D. Merman, *Quantum Computer Science an Introduction*, Cambridge University press, USA, 2007.

[15] X. FU, W. BAO*, C. ZHOU, *Design and Implementation of Quantum Factorization Algorithm*, 3rd International Symposium on Intelligent Information Techhnology and Security Informatics, China, Dec 15-19, 2007.

[16] R. Kumar, A. Ranjan and P. Srivastava, *An Analytical Optimization Based on Quantum Computing Embedded into Evolutionary Algorithm*, 2nd International Conference on Education Technology and Computer (ICETC), China, June 20-24, 2010.

[17] J.S. Rahhal, Dia I. Abu-AL-Nadi and M. Hawa, *Viterbi Decoder Algorithm using Quantum Computing*, IEEE Congress on Evolutionary Computation (CEC), Singapore, Sep 25-28, 2007.

[18] C.P Williams, *Explorations in Quantum Computing,* 2nd edition, Springer, New York, 2011.

[19] J.L. Traff, W.D. Gropp, R. Thakur, *Self-consistent MPI performance guidelines*, IEEE Transactions on parellel and distributed systems vol. 21, pp.698-709, 2010.

[20] J.J. Vartiainen, A.O. Niskanen, M. Nakahara, M.M. Salomaa, *Implementing Shor's algorithm on Josephson charge qubits*, Vol. A 70, pp. 012319-1-11, Physical Review 2004

[21] C.S. Calude, M.J. Dinneen, F. Peper, *Unconventional models of computation*, Third International Conference, UMC 2002, Kobe, Japan, Oct 2002.