International Journal of the Physical Sciences Vol. 6(17), pp. 4273-4281, 2 September, 2011
Available online at http://www.academicjournals.org/IJPS
DOI: 10.5897/IJPS11.488
ISSN 1992 - 1950 ©2011 Academic Journals

Full Length Research Paper

Security constrained economic dispatch using multi-thread parallel computing

Alimorad Khajeh Zadeh, Hossein Zeynal and Khalid Mohamed Nor*

Centre of Electrical Energy Systems (CEES), Faculty of Electrical Engineering, Universiti Teknologi Malaysia, (UTM), 81310 Skudai, Johor, Malaysia.

Accepted 05 May, 2011

This paper presents the development of an efficient security constrained economic dispatch using multi-thread parallel computations. One of the main obstacles in wide application of economic dispatch is the slowness of algorithm execution time, especially when a large system is utilized. Aside from the model development to overcome this, we have proposed a technique to perform parallel computing of the problem which can make solution converged in shorter time. In this sense, the serial economic dispatch is decomposed into four sub-process alternatives for multi-thread programming. Among them, the two processes of loss-coefficient and barrier-optimization are potentially qualified for multi-thread parallel computing. Results obtained shows that the multi-threaded loss-coefficient process has improved the speed-up for the serial computations in all the cases at where the multi-threaded barrier-optimization process can improve the speed-up only if the execution time for the process at single iteration can take more than one second, which mostly happens in the systems with a large number of units. The performance results of the multi-thread economic dispatch exhibit improved speed-up and performance characteristics compared to the serial execution time in all cases. A variety of case studies of standard IEEE 118, 300 bus and practical utility 664, 4995 bus are employed to validate the promising performance characteristics.

Key words: Security constrained economic dispatch, multi-thread parallel computing, barrier optimization, exact loss.

INTRODUCTION

Economic dispatch is one of the major problems in electrical power system analysis. It treats the system by an efficient allocation of the generator outputs in such a way that minimizes the total costs of the system while all the operating and physical constraints taken into account (Chowdhury and Rahman, 1990).

Economic dispatch problem has been traditionally solved by a set of coordination equations using Lambda-iteration method, the Newton method (Wood and Wollenberg, 1984), and the gradient method (Lee et al., 1984). A method to calculate the penalty factor which uses load flow Jacobian matrix has also been investigated (Happ, 1974). Quadratic programming has, however, been applied to solve the economic dispatch problems. This method possesses the good convergence characteristics and handles constraints efficiently. Also, it requires more computational storage than the classical algorithm and is restricted to quadratic cost functions. Apart from these methods, an algorithm developed in Khalid and Abdul (1991), by correctly fixing the output of the generators violating the constraints.

One of the major applications of economic dispatch is in unit commitment solution. Unit commitment is basically the most economical solution in power system operations and the speed of its solution is greatly tied to the economic dispatch problem. Therefore, its accuracy

*Corresponding author. E-mail: khalidmn@fke.utm.my.

is crucial to the final solution. In other words, the challenges in economic dispatch algorithm are in solution accuracy, nonlinear objectives and constraints, transmission loss models as well as execution speed (Nimje et al., 2011). It is noted that the previous works on unit commitment used simplified models of economic dispatch problems as a way of reducing the amount of time taken to solve the large number economic dispatch problems (Güvenç, 2010).

The advent of parallel computing offered opportunities for numerically intensive applications such as analysis and simulation of large-scale systems. The use of parallel computation in power system analysis has been investigated since 1990's, (IEEE Committee Report, 1992; Wu and Anjan, 1995). Three main issues identified in the application of parallel computation to power system analysis are processor architecture, software development and algorithm development.

As far as algorithm change is concerned, works reported in the literature tends to decompose the NR and FD algorithms so that they are amenable to parallel computation. The detailed calculations remain essentially the same. Therefore, what have been investigated has not significantly modified the actual computation in NR and FD methods.

Many efforts that have been reported in implementing parallel computation, concentrate on the processor architecture issue (Chen and Chen, 2001a, Chen 2004b; Tu and Flueck, 2001). In the past parallel processing computer system tends to consist of a number of physical processors connected externally by a very fast communication system (Chen and Chen, 2001a; Chen, 2004b). The system can be of distributed memory type where many memories are physically separate modules and each memory is accessed by specified processor. The other type is shared memory where all processors assess the same memory.

A few variations of parallel implementation of load flow analysis used the type of parallel computing architecture mentioned previously (Chen and Chen, 2001a; Chen, 2004b). One of the implementations is by using clusters of five transputers (Chen and Chen, 2001a; Chen 2004b). In another version, the hardware is based on clusters eight single processors connected by a Fast Ethernet network (Tu and Flueck, 2001). These systems used a specialized hardware which is more expensive than a general purpose computer.

The off-processor inter-process communication in the parallel computer systems discussed earlier creates speed latency. One of the ways to reduce the speed latency is by having the processors connected in a single chip package, such as by using multiple programmable chips connected in a cluster implemented in a single chip FPGA (Wang et al., 2007). The technique of using configurable chip designed specially to solve the power flow problem is diverging from the path of using commercial off the shelf (COTS) system, such as the personal computer (PC) or workstation. A specialized system that runs only load flow application would not be cost effective than the present serial (also known as sequential) load flow algorithm running in a serial computer that is available as COTS. In this case, the same serial computer can be used for many other applications, which is very important, as the PC has become one of the major computing platforms for engineering software applications.

However, in terms of the inter-process communication, the arrangement of multi-processing by using many cores in a single die is more efficient than if the same number of physically separate processor cores is con-nected through an external communication system. This arrangement is called the multi-core computing. Lately, nearly all the recent PCs are multi-core machines. In addition to being parallel due to its multiple processor cores, each core can execute multi-thread instructions. In other words, the PC is now a cost effective parallel computer.

There are a number of industry standards parallel application programming interfaces (API) that can help to develop a parallel program (OpenMP review Band, 2008; Brown and Ilya, 2007) in a parallel computer. These API's are integrated into compilers such as C/C++ and FORTRAN. An API manages the memory, under programming control, to ensure no conflicts arise. One of the API's is the OpenMP whose design makes it easy to modify an existing sequential code (Brown and Ilya, 2007). Furthermore, a program with OpenMP can run in serial computer as the OpenMP command will be treated just as a comment statement. In other words, the same program can be used for parallel and serial computation.

The combination of the multi-core and multi-thread hardware and the parallel compiler resolved the issue of software development for parallel processing identified earlier (IEEE Committee Report, 1992). The parallel compiler enables applications to be developed in a transparent manner, which means that they can be used for any number of cores and threads without rewriting. Serial program developed on the PC platform is portable in hardware that varies from notebooks, desktops and workstation computers. As the PC is practically a COTS system it is cheaper than the specialized parallel computing system of the past.

The objective of the research work reported in this paper is to develop the security constrained economic dispatch using multi-thread parallel computing.

MULTI-THREAD COMPUTING

In general an operating system controls the execution of many processes by a microprocessor core. Each process is independent of each other with its own state information,

address space and only interact with each other through an inter-process communication system. By using context switching, the processes are run in a multi-tasking environment but each process is run sequentially at any instance of time. Context switching is a process of saving and restoring the state of a CPU such that multiple processes can share a single resource.

Most common microprocessors nowadays, such as those from INTEL Corporation or AMD Corporation, are of the multi-core processor type, which consist of two or four execution processors on a single die. In this case, the operating system will control the many processes which some can be run in parallel at the same time, according to the number of core, instead of being multi-tasked.

Each core of an INTEL multi-processor, can execute two threads in parallel (Deborah et al., 2002). The time-division multiplexing used by multi-threading creates an environment where a program is configured to allow processes to fork or split into two or more threads of execution, as shown by Figure 1. Creating a fresh thread is very similar to forking a new process, but when a new process is forked, it shares relatively little data with the parent process which created it. This is in contrast with a fresh thread created at a point where much information are shared, such as all the global variables and static local variables, the open files and the process ID. A single process might contain multiple threads, which share the same state and same memory space. Since threads in a process share the same variables they communicate with each other directly (Deborah et al., 2002). This is more efficient than a multi-process as the time taken to pass control from one process to another is longer than the time required in passing control from a thread to another thread.

Multi-threading is purposely planned to achieve a more efficient use of computer resources by allowing resources that are already in use to be simultaneously utilized by a slight variant of the same process. As an example, when a thread gets a lot of cache misses the other threads can take advantage of the unused computing resources to continue. This is accomplished by context switching, which enables the saving and restoring the state of threads so that they can share a single resource, which is just as similar as context switching between processes. This will lead to faster overall execution, as those resources would have been idle had there been only a single thread executed sequentially (Deborah et al., 2002).

One of the slight disadvantages of multi-threading computing is that the speed-up may result in some conflicts between threads when attempting to share caches or other hardware resources. Multi-threading could also lower the response time of each single thread in the process affecting time savings that are generated by the multi-thread computing.

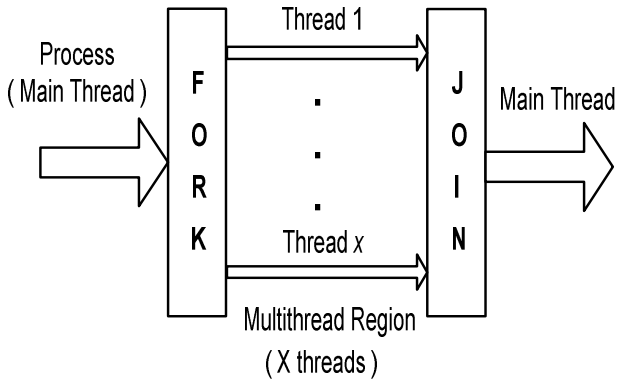

Figure 1. Multi-threading operational model.

PROBLEM FORMULATION

The intent of the economic dispatch is to allocate the generator's output while the total generation costs minimized and all constraints satisfied. Generator cost functions usually considered as quadratic polynomial functions, so in this work the total cost of all generators in the economic dispatch problem can be expressed as the follows:

$$\min_{p_{gi}} F = \sum_{i=1}^{NG} F_i(P_i) = \sum_{i=1}^{NG} (a_i + b_i P_i + c_i P_i^2) \quad (1)$$

The constraints can be listed as:

The total active power generation must be equal to the sum of demand plus the network losses, which can be expressed as:

$$\sum_{i=1}^{NB} p_{gi} = P_L + P_D \quad (2)$$

It is said that each generator has a certain capacity that should not be violated and has to be confined within the specific up and down bounds as:

$$p_{gi}^{\min} \le p_{gi} \le p_{gi}^{\max} \quad (3)$$

Line flows have to be monitored and retain within the transmission line capacities to avoid any unseen insecurity events. A set of generalized Z-Bus distribution factors (GZBDF) (Brar et al., 2004) is thus used to impose the line security in the problem as:

$$\sum_{i=1}^{NG} G_p(m,i) P_{gi} \le P_m^{\max} \quad (4)$$

Loss formulation

Transmission fixed loss is normally used in power system problems such as unit commitment, but it led an inaccurate result as the power loss in transmission lines is the function of active power injection into the network. Further, in each set of optimal generation schedules, the amount of loss is different. At this work, the most exact loss formula has been used. It is based on α, β loss coefficient, which can be derived as:

$$p_L = \sum_{i=1}^{NB}\sum_{j=1}^{NB}\alpha_{ij}p_{gi}p_{gj} + 2\sum_{i=1}^{NB}\sum_{j=1}^{NB}(p_{di}\alpha_{ij} - Q_i\beta_{ij})p_{gj} + \sum_{i=1}^{NB}\sum_{j=1}^{NB}(p_{di}\alpha_{ij} - 2Q_i\beta_{ij})p_{dj} + \sum_{i=1}^{NB}\sum_{j=1}^{NB}Q_i\alpha_{ij}Q_j \quad (5)$$

$$\alpha_{ij} = \frac{R_{Nij}}{|V_i||V_j|}\cos\theta_{ij} \quad \beta_{ij} = \frac{-R_{Nij}}{|V_i||V_j|}\sin\theta_{ij}$$

Line distribution factors

The most accurate line distribution factors are derived through a set of GZBDF factors. These factors enable one to add the security line constrained into the economic dispatch solution. These are the factors that behaving as a function of active power injections into the buses which can be expressed as Brar et al. (2004).

$$P_{Tm} = \sum_{i=1}^{Ng}[G_p(m,i)P_{Gi} - G_Q(m,i)Q_{Gi}] \quad \text{m = 1, 2, NL} \quad (6)$$

Execution time breakdown statistics

Economic dispatch analysis has been traditionally solved by a single processing machine used for its sequential process. This serial computation can be broken down into four sub-processes in order to examine which computational processes have excessive demand of getting parallelism. These are load-flow, loss-coefficient, line-distribution-factor and barrier-optimization processes. Table 1 shows the results for the serial economic dispatch and detailed execution time that has been apportioned into the five aforementioned processes. In this case, a variety of test systems ranging from standard IEEE 118, 300 bus up to practical utilities with 664, 4995 bus are employed to validate the performance characteristics of the economic dispatch at serial and multi-thread versions.

MULTI-THREAD ECONOMIC DISPATCH

Implementing the multi-threading computing pattern into the economic dispatch process required an observation of what processes are more potential for multi-thread computing. However, these potential alternatives are determined through the serial economic dispatch solution. The key is to find a process which consumes more computing powers in serial version. Therefore, the result obtained for serial economic dispatch tends to ease these decisions. Table 2 showed the execution time contributions of each process in serial economic dispatch in various test cases. In general, the results reveal the major execution times spent in the calculations of load-flow, loss-coefficient and barrier-optimization. As seen in Table 2, in the case of the 118 bus system with a total unit of 54, the load-flow by 18% and barrier-optimization by 75% computing times mostly occupied the major time of economic dispatch process, whereas in the case of the 300 bus system with 69 units, dominant execution times were taken by three processes of load-flow with 13%, loss-coefficient with 13% and barrier-optimization by 71%. In addition, these processes are noticed in the case of the practical large systems of 664 with 144 units and 4995 bus systems with 483 units. However, the major costly processes are those of loss-coefficient with 30 and 51% respectively and barrier-optimization processing time of 58 and 40% accordingly.

The proposed multi-thread economic dispatch is coded using C++ programming with the INTEL C++ compiler in Visual Studio IDE, INTEL (2009). A PC with an INTEL Pentium Core Duo E2180, 2.00 GHz with 2.00 GB of RAM has been used as the hardware platform.

Multi-thread load flow

Load-flow computation happens in every iteration throughout the economic dispatch process. In other words, whenever any generator's output changes there has always been an essential need of the fresher load-flow solution in order to re-establish the present state of the system in terms of voltage and angle across the grid. This process has been earlier nominated as a potential alternative process to bend multi-threading parallel computing right into its calculations at all cases. However, the computations in the full Newtown-Raphson load-flow can be commonly summarized into four sub-procedures as of the mismatch, constructing Jacobian matrix elements, solving the linear equation and the update of the variables. Among these common tasks, the most challenging part of computation is in the solution of a set of linear equations, which can be typically cast as:

$$AX = B \quad (7)$$

It is noted that the solution to this equation plays an important role in the load-flow entire performance. This solution is depended on a massively matrix operation to

Table 1. Serial execution times in Milisecond.

Gen	Iter	LF	sLoss	DF	sBO	Total
54	4	17.81	5.29	2.00	76.20	101.4
69	5	41.86	42.20	6.13	225.2	315.7
144	4	48.12	168.6	23.63	327.4	568.5
483	6	754.8	9824	1017	7622	19223

Table 2. Percentage of execution times in economic dispatch.

Gen	Iter	LF	sLoss	DF	sBO
54	4	18	5	2	75
69	5	13	13	2	71
144	4	8	30	4	58
483	6	4	51	5	40

invert the Jacobian matrix A where it has greatly taken excessive processing power to end up the process. However this calculation is the dominant total execution time in load-flow process. In this case, SuperLU direct linear solver can be used to yield a sharper solution for the load flow process and facilitate inverting the Jacobian matrix A in the less computational cost (Demmel et al., 1999). As a result, it is recognized that SuperLU package is one of the efficient software, which is freely available in the public domain. Further, it has been universally accepted for its powerful performance, especially in solving large sparse matrix set of linear equations. This software has three libraries which include: sequential SuperLU, Multi-thread SuperLU and distributed SuperLU at which every package is aimed in certain aspect of the solution with predefined memory architectures. Multi-thread SuperLU library is eventually used in our load flow where it is designed to work on a shared memory architecture.

Enforcement of multi-threading in load-flow computations was previously sound demanding but there are certain bottlenecks for a forceful multi-thread load-flow process. First of all, Tables 1 and 2 show that albeit load-flow has significant contribution time of 18% in 118 bus case as well as 13% in 300 bus systems but since their quantities are 17.81 ms out of 101.4 ms and 41.85 ms out of 315.72 ms, which are basically considered as the short-living processes compared the total execution time require for serial economic dispatch thus they are technically unwilling to become multi-threaded. It should be noted that the multi-thread technique has been designed to alleviate the process with massively high computational cost. However, the key is that the time spent for load flow process is relatively very small, and even it becomes crucial when the dimension of the system grows up. Additionally, it is noted that the quantity of the time given in the Table 1 should be divided by the number of its iterations, which eventually worsen the case for seeking multi-threading opportunities in the load-flow process. It is technically worthwhile if one can pick the processes that are computationally costly, another way, it is avoided applying the multi-threading at where it is barely required.

Since a power system by nature is a sparse system at where a few connections are only drawn among generations in the far and the loads in an area. Therefore, the number of non-zero elements of the Jacobian matrix is always small. As Table 3 shows, the percentage of non-zero elements has been drastically reducing while the size of the system increased. Although in the case of a large 4995 bus system, a small percentage of non-zero elements made it very sparse so that the case is unattractive for multi-threading to over come the process. Generally speaking, in the systems with a set of linear equation multi-threading approach is said to be extremely precious whenever it involves with a matrix that has a great number of non-zero elements in the matrix.

Thereby, the sequential load flow would be a better alternative for an efficient economic dispatch where the serial value seemed unnecessary to get multi-threaded. However the complete proposed multi-thread economic dispatch algorithm has been depicted in Figure 2, where necessary parts for parallel computation have been painted as the symbol in Figure 1. Figure 2 shows that there are two most cumbersome processes in economic dispatch which required parallel computation, although the other process looked demanding at first glance but they have appeared uncompetitive to other heavy processes. In the subsequently, the procedure and results on

Table 3. Sparsity and problem size for both full and decoupled Newton-Raphson power-flow methods.

No. of bus (N)	No. of Gen	No. of lines (M)	Full Jacobian matrix		
			Problem size	Non-zero elements	
			2N×2N	4N+8M	%
118	54	179	55696	1904	3.42
300	69	409	360000	4472	1.24
664	144	757	1763584	8712	0.49
4995	483	5217	9980010	61716	0.06

how the parallel programming helps the existing serial economic dispatch will be assessed.

Multi-thread loss coefficient

Loss coefficients have to be obtained by Equation 5, and this process took a number of summation loops in its calculation. It is explicitly seen in the shape of equation in the Equation 1. The process of making loss-coefficient has been increasingly soliciting for better speed up. Similarly, it was also discovered that one of the potential processes that multi-thread computing can be advantageous. Therefore, this process has been going through the multi-threading computing pattern and results are reflected in Table 4. It showed how multi-threading is acting when a certain number of the threads is applied. In case of the 118 bus system with the serial solution of 5.28 ms has been boosted to a faster solution of 1.29 ms when a total of two threads handled the process concurrently. Further acceleration obtained by using three and four threads, which respectively reduced the execution time of the process to 0.954 ms and 0.795 ms. The same story goes for the other case studies which all then came up with better performance once the number of used threads increased. In order to examine how fast each additional thread can speed up the loss-coefficient process. Therefore, Table 4 has also provided the amount of speed up received by multiple use of threads. For instance, in the case of 54 units the speed up for using two threads rather than one thread or serial version marked by 4.09, and even it becomes faster by using three and four threads at where the speed up climbed to 5.54 and 6.65, respectively. Identical improvements and speed up can be achieved for other case studies over their serial values, which have been given in Table 4.

Having said that the initial intent of multi-threading or in general sense the parallel computing is to solve the massively complex problems with a heavy computational burden. Therefore, it likely has been designed to gain the speed up in the process that employs a large and complex system. Results emphasize that better speed up are achieved with extra thread used.

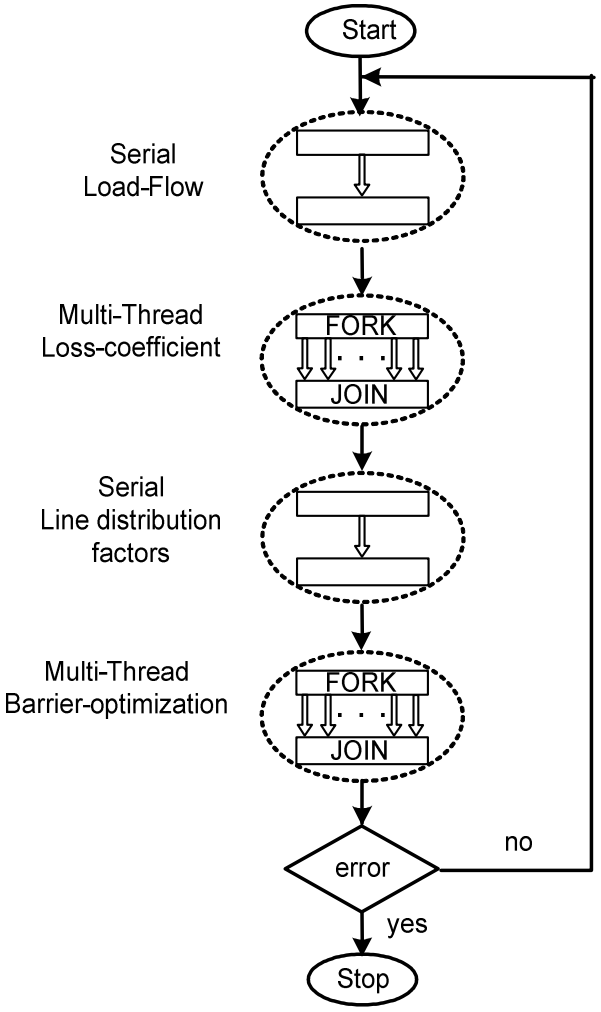

Figure 2. Flow chart for multi-thread security constrained economic dispatch.

Multi-thread barrier optimization

Solving the economic dispatch problem with a set of linear and quadratic constraints required appropriate optimization alternatives. Among the interior point optimization

Table 4. Comparative performance of multi-thread and serial loss calculation.

Gen	Execution time (ms)				Speed-Up		
	1 TD	2 TD	3 TD	4 TD	$\frac{Serial}{2Tread}$	$\frac{Serial}{3Tread}$	$\frac{Serial}{4Tread}$
54	1.43	0.31	0.31	0.30	4.58	4.60	4.73
69	5.3	1.29	0.95	0.80	4.10	5.54	6.65
144	42.2	9.59	6.33	5.22	4.40	6.66	8.08
483	168	37.9	25.9	19.3	4.46	6.49	8.72

Table 5. Comparative performance of multi-thread and serial barrier optimization.

Unit	Execution time (ms)				Speed-Up		
	1 TD	2 TD	3 TD	4 TD	$\frac{Serial}{2Tread}$	$\frac{Serial}{3Tread}$	$\frac{Serial}{4Tread}$
54	76	220	249	335	0.35	0.31	0.23
69	225	376	443	665	0.60	0.51	0.34
144	327	479	549	521	0.68	0.60	0.63
483	7622	5556	4776	4299	1.37	1.60	1.77

optimization methods, the barrier optimization technique has been used to solve the economic dispatch problem. Interior point barrier algorithm, an efficient quadratic programming algorithm, is used to solve economic dispatch problem. The barrier algorithm also caters for any combination of polynomial functions such as polynomial cost functions. Nonlinear characteristics of an electrical power system such as generators limitations, transmission losses and nonlinear cost functions are considered in the economic dispatch formulation. Moreover, this process in economic dispatch has previously qualified as an alternative process for multi-threading technique to fork in.

As a result, Table 5 shows how multi-thread computing technique is working on the process where a particular number of concurrent threads utilized. Unlike the 4995 bus case with total 483 units, in all the test cases the degrees of speed up are declining by the rising number of threads used.

An ILOG CPLEX Barrier Optimizer used, which is available as a callable C/C++ library, IBM ILOG CPLEX. It tends to provide a solution for quadratic constrained programs and second-order cone programming (SOCP) problems. ILOG CPLEX barrier optimizer which is based on a primal-dual, predictor-corrector method, is able to deal with large-scale linear programs. In this work, a multi-thread CPLEX used to achieve the promising performance.

DISCUSSION

The multi-thread parallel computing is developed in security constrained economic dispatch process. There have been three sub-processes required the multi-thread computation, which are the load-flow, loss-coefficient and barrier-optimization.

The load-flow process has been dismissed for multi-threading computations as of its quick solution and the highly sparse Jacobian matrix that appears in the load flow equations.

The loss-coefficient process has been multi-threaded and an improved speed up attained by multi-thread computing of the loss-coefficient process which is promising for all the case studies. Table 6 demonstrates the speed-up obtained in multi-thread economic dispatch where only a loss-coefficient process has been multi-threaded. The speed-up amounts can be yielded by comparison of multi-threaded economic dispatch values, using four threads, over its serial values. Therefore, it is clearly showing that the performance of multi-thread economic dispatch enhanced in all the test systems with a rising degree of speed-up in all cases. Moreover, the number of buses is a key word in the calculations of the loss-coefficients because the volume of the computations is tied with the number of buses, in other words, having a huge number of repetitive loops in the formulation of Equation 5 of the loss-coefficient which makes the calculation for loss-coefficients quite hefty. Hence, the results shown in Table 6 acknowledge that the higher speed-up attained for the larger system with a bigger number of buses they have.

The barrier-optimization process has also been going through the multi-threading effort and the performance results for the multi-thread economic dispatch over its serial version have been depicted in Table 7. However, it

Table 6. Comparative performance of attending multi-thread loss coefficient in economic dispatch.

Unit	Iter	sED	mtED	Speed-up
54	4	101.40	96.91	1.05
69	5	315.72	278.75	1.13
144	4	568.59	419.26	1.36
483	6	19223.22	11135.47	1.73

Table 7. Comparative performance of attending multi-thread barrier optimization in economic dispatch.

Unit	Iter	sED	mtED	Speed-up
54	4	101.40	360.31	0.28
69	5	315.72	755.02	0.42
144	4	568.59	762.10	0.75
483	6	19223.22	15899.88	1.21

is noted that the results are with the situation that only the barrier-optimization process has been multi-threaded. Furthermore, results show that the multi-threading can sometimes even prolong a process. It is happening if the number of the optimization variables was few, in the other hand, the number of units in the system becomes determinative in such a way to gain better improvement on the execution time when the multi-threading is applied. Thereby, the system with 483 units has been boosted its speed up over the serial version. It can be seen that in such a huge system with a larger number of variables the speed up can be growing by the number of the threads used. However, the speed up of 1.20 is achieved in the case with 483 units and a number of four concurrent threads applied.

As a result, enforcing the multi-thread loss-coefficient in the all cases plus the multi-thread barrier-optimization where it needed. It can increase the speed-up for the final multi-thread economic dispatch as shown in Table 8 at which demonstrated an efficient speed-up in all the cases occurred. Speed-up for multi-thread economic dispatch has been growing up where a 1.05 speed-up is achieved in 54 unit system, and the speed-up rate has been climbed up to the 2.47 for the 483 unit system.

Conclusions

The idea of multi-thread computing technique has been implemented on the security constrained economic dispatch in order to exploit the powerful processing capacity of multi-thread parallel computing for ace-lerating the execution time of the solution. In this sense, serial economic dispatch can be decomposed into four sub-processes at which one can deftly observe what process needs to be multi-threaded. The most potential processes found to be those of load-flow, loss-coefficient and barrier-optimization processes. The detail results of the multi-thread economic dispatch have then unveiled that the most advantageous process for multi-thread computation is the loss-coefficient process along with the barrier-optimization process. The multi-threading of the loss-coefficient process has always improved the speed-up of the process itself as well as infused the speed-up for the whole economic dispatch, whereas the multi-threaded barrier-optimization process can be effective when the number of system units is large and the processing time taken for the calculations in a single iteration can jump beyond one second.

As the future direction to this research, the algorithm can be modified in such a way that it can encompass more practical constraints in economic dispatch. The security assessment of economic dispatch has highly been an enticing issue recently in power market and system restructuring, where the system operator should be borne, excessive financial strain to draw enough security in the system. Therefore, the mechanism of practical economic dispatch may become more complicated and the beauty of parallel computing outfit can effectively make the problem working in the place. The authors are undergoing a process to verify how parallel computing may untie the computational burdens in practical contingency-based economic dispatch and unit commitment.

ACKNOWLEDGMENT

This work is partially sponsored by the Malaysian Government under FRGS grant Vot No. 78460.

Table 8. Multi-thread economic dispatch.

Unit	Iter	sED	mtED	Speed-up
54	4	101.40	96.91	1.05
69	5	315.72	278.75	1.13
144	4	568.59	419.26	1.36
483	6	19223.22	7792.30	2.47

Nomenclature: ***i, j,*** Suffix or prefix of busbar indices; **sED,** serial economic dispatch; **sLoss,** serial loss-coefficient; **sBO,** serial barrier optimization; **mtED,** serial economic dispatch; **mtLoss,** multi-threaded loss-coefficient; **mtBarr,** multi-threaded barrier optimization; **Iter,** number of iterations; **NB, NG,** number of buses and generators; **M,** number of branches; **NNZ,** number of non-zero elements; **PTm,** active line flow; p_{gi}, $p_{gi}^{\min}, p_{gi}^{\max}$, active generation, minimum and maximum generations in bus i; $G_p(m,i), G_Q(m,i)$, active and reactive line distribution factor.

REFERENCES

Brar YS, Dhillon JS, Kothari DP (2004). Fuzzy Logic Approach for Generation Dispatch of Electric Power System with Conflicting Objectives", IEEE proceeding: pp. 123-130.

Brown B, Ilya S (2007). High-Scalability Parallelization of a Molecular Modeling Application: Performance and Productivity Comparison Between OpenMP and MPI Implementations', Int. J. Parallel Programming, 35(5): 441–458.

Chen S (2004). A study based on the factorization-tree approach for parallel solution of power network equations', Elect. Power Syst. Res., 72: 253–260.

Chen SD, Chen JF (2001). A novel node ordering approach for parallel load flow analysis', Eur. Trans. Elect. Power, 11(4): 56-64.

Chowdhury BH, Rahman S (1990). "A review of recent advances in economic dispatch" IEEE Trans. Power Syst., 5(4): 1248-1259.

Deborah TM, Frank B, David LH, Glenn H, David AK, Alan MJ, Michael U (2002). Hyper-Threading Technology Architecture and Microarchitecture' INTEL Technol. J. Q1, 6(1).

Demmel JW, Eisenstat SC, Gilbert JR, Li XS, Liu JWH (1999). "A supernodal approach to sparse partial pivoting", SIAM Journal on Matrix Analysis and Applications. 20(3): 720 -755.

Güvenç U (2010). Combined economic emission dispatch solution using genetic algorithm based on similarity crossover. Sci. Res. Essays, 5(17): 2451-2456.

Happ HH (1974). Optimal Power Dispatch. IEEE Trans. PAS-93(3): 820-830.

IEEE Committee Report (1992). Parallel processing in power systems computation. IEEE Trans. Power Syst., 7(2): 629–38.

INTEL (2009). ® C++ Compiler User and Reference Guides, Document number: 320916-001US.

Khalid MN, Abdul HAR (1991). Efficient economic dispatch algorithm for thermal unit commitment. IEEE Proc. C, Generation, Trans. Distribution, 138(3): 213-217.

Lee KY, Park YM, Ortiz JL (1984). Fuel cost minimization for both real- and reactive power dispatches, Proc. Inst. Elect. Eng., Gen., Trans. Distribution, 131(3): 85–93.

Nimje AA, Panigrahi CKr, Mohanty AK (2011). Enhanced power transfer capability by using SSSC. J. Mech. Eng. Res., 3(2): 48-56.

OpenMP Architecture Review Board (2008). OpenMP Application Program Interface Version 3.0'.http://www.openmp.org/mp

Tu F, Flueck AJ (2001). A Message-Passing Distributed-Memory Newton-GMRES Parallel Power Flow Algorithm. IEEE Power Eng. Soc. Meeting, 3(1): 178-185.

Wang X, Ziavras SG, Nwankpa C, Johnson J, Nagvajara P (2007). Parallel solution of Newton's power flow equations on configurable chip. Int. J. Elect. Power Energy Syst., 29(5): 422-431.

Wood AJ, Wollenberg BF (1984). Power Generation, Operation and Control. New York: Wiley.

Wu JW, Anjan B (1995). Parallel solution of large sparse matrix equations and parallel power flow. IEEE Trans. Power Syst., 10(3): 1343- 1349.