PHILIPPINE ENGINEERING JOURNAL
PEJ 2011; Vol. 32, No. 1: 45-56

TIMING MEASUREMENTS FOR A PARALLEL NEWTON-RAPHSON LOAD FLOW PROGRAM FOR A CLUSTER OF WORKSTATIONS

Roberto Uy and Manuel L. Ramos, Jr.Ph. D.,
Electrical and Electronics Engineering
University of the Philippines, Diliman

ABSTRACT

This paper presents an alternative implementation of the Newton-Raphson power flow suitable for commodity clusters that utilizes widely available software components. The components used are specified and the data decomposition along with the changes to the power flow formulation to be more suitable to run on a cluster is described. Timing measurements for the power flow computation and the overall execution time are used to show the values of speedup obtained as the provided input sizes and number of processors is varied.

1. INTRODUCTION

Power flow analysis is concerned with the steady state operation of power network. Under such conditions, the power network can be represented as a single phase network. Equations governing the network is systematically formulated using the node-voltage method. Figure 1 shows a typical bus from a power system network. V_k is the voltage at the bus, I_k is the net current entering the bus. Transmission lines connecting the buses are usually provided with impedance data that are converted to per-unit admittances to minimize division operations in the calculations. The lines are represented by their equivalent pi-models with a per-bus admittance to ground y_{k0} and the line admittance between two buses k,m denoted by y_{km}.

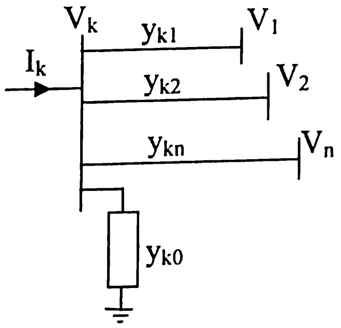

Figure 1: A typical Bus of a Power System Network

Correspondence to: *Institute of Electrical and Electronics Engineering, University of the Philippines, Diliman*

Received: May 9, 2005
Accepted: February 15, 2012

Network equations formulated in nodal admittance form produces complex linear equations in terms of node currents. Applying Kirchoff's Current law to the bus shown in Figure 1 results in the following expression for the current entering bus k in terms of the admittances between nodes and node voltages.

$$I_k = V_k \sum_{m=0}^{n} y_{km} - \sum_{m=1}^{n} y_{km} V_m \quad m \neq k \tag{1}$$

The Y_{bus} is a matrix of the admdittances in a power network defined such that $I_{bus} = Y_{bus}V_{bus}$, where I_{bus} is a column vector whose elements are the currents entering each node and V_{bus} is a column vector whose elements are the voltages at each node. Rewriting (1) in terms of the Y_{bus} elements results in

$$I_k = \sum_{m=0}^{n} Y_{km} V_m \tag{2}$$

It is more suited to use the polar representation of the complex variables I_k, Y_{km}, and V_m, denoted by $|I_{km}|\angle I_{km}$, $|Y_{km}|\angle Y_{km}$ and $|V_m|\angle V_m$, for the power flow computation since the data given for voltage controlled buses is the real power and the voltage magnitude. Expressing (2) in polar form results in (3).

$$I_k = \sum_{m=0}^{n} |Y_{km}||V_m|(\angle Y_{km} + \angle V_m) \tag{3}$$

In a power system, values of power rather than current is what is usually known. Modifying the network equations to be expressed in terms of voltage and power produces the power flow equation. The complex power at bus k is defined in (4).

$$P_k - jQ_k = V_k^* I_k \tag{4}$$

Substituting (3) into (4), and expressing V_i^* in polar representation results in the power or load flow equation as shown in (5).

$$P_k - jQ_k = (|V_k|\angle V_k)\sum_{m=0}^{n} |Y_{km}||V_m|(\angle Y_{km} + \angle V_m) \tag{5}$$

The load flow equations are nonlinear complex algebraic equations that must be solved using iterative techniques because the complex derivatives of the power flow equations do not exist. It is therefore necessary to separate the equations in terms of real functions and variables, (6) and (7), before applying the iterative technique. The most common iterative technique applied to power flow computation is the Newton-Raphson method.[1]

$$P_k = \sum_{m=1}^{N} |V_k||V_m||Y_{km}|\cos(\angle Y_{km} + \angle V_m - \angle V_k) \tag{6}$$

$$Q_k = -\sum_{m=1}^{N} |V_k||V_m||Y_{km}|\sin(\angle Y_{km} + \angle V_m - \angle V_k) \tag{7}$$

The power system buses have specified real power P_k and imaginary power Q_k for each load bus. For voltage regulated buses, a real power and the magnitude of the voltage at the bus are specified.

Newton's method is a successive approximation procedure for solving nonlinear systems of equations based on an initial estimate of the independent variable and the Taylor's series expansion about that estimate. For (6) and (7), the independent variables are $|V_k|$, $\angle V_k$, $|V_m|$ and $\angle V_m$. The suitable expressions to calculate the voltage magnitude and angle mismatches that apply the Newton-Raphson technique to the power flow equation is shown in (8) and (9). The iterations continue updating bus voltages until the magnitude of the maximum power mismatch fall below a specified value.

$$\Delta P_k = \sum_{m=1}^{N}\left(\frac{\partial P_k}{\partial \angle V_m}\Delta\angle V_m + \frac{\partial P_k}{\partial |V_m|}\Delta|V_m|\right) \tag{8}$$

$$\Delta Q_k = \sum_{m=1}^{N}\left(\frac{\partial Q_k}{\partial \angle V_m}\Delta\angle V_m + \frac{\partial Q_k}{\partial |V_m|}\Delta|V_m|\right) \tag{9}$$

The linear system that arises from the application of the Newton-Raphson method is shown in (10) below. The Jacobian matrix arises naturally from the linearization of the power flow equations. [1]

$$\begin{bmatrix} \Delta P_2 \\ \vdots \\ \Delta P_n \\ \Delta Q_2 \\ \vdots \\ \Delta Q_n \end{bmatrix} = \begin{bmatrix} \frac{\partial P_2}{\partial \angle V_2} & & \frac{\partial P_2}{\partial \angle V_n} & \frac{\partial P_2}{\partial |V_2|} & & \frac{\partial P_2}{\partial |V_n|} \\ \vdots & & \vdots & \vdots & & \vdots \\ \frac{\partial P_n}{\partial \angle V_2} & & \frac{\partial P_n}{\partial \angle V_n} & \frac{\partial P_n}{\partial |V_2|} & & \frac{\partial P_n}{\partial |V_n|} \\ \frac{\partial Q_2}{\partial \angle V_2} & & \frac{\partial Q_2}{\partial \angle V_n} & \frac{\partial Q_2}{\partial |V_2|} & & \frac{\partial Q_2}{\partial |V_n|} \\ \vdots & & \vdots & \vdots & & \vdots \\ \frac{\partial Q_n}{\partial \angle V_2} & & \frac{\partial Q_n}{\partial \angle V_n} & \frac{\partial Q_n}{\partial |V_2|} & & \frac{\partial Q_n}{\partial |V_n|} \end{bmatrix} \begin{bmatrix} \Delta\angle V_2 \\ \vdots \\ \Delta\angle V_n \\ \Delta|V_2| \\ \vdots \\ \Delta|V_n| \end{bmatrix} \tag{10}$$

The information desired is the voltage at the various nodes of the network. Given the voltage magnitudes and phase angles, additional information about the system such as power flows for each transmission unit and reactive power generated or absorbed at voltage regulated buses can be obtained.

The application of parallel processing in power flow computation aims to reduce the amount of time it takes to arrive at a solution.[2] The processing time for power flow computation increases with the number of buses and the complexity of the network. Larger power systems and more detailed network models both involve a greater number of buses. Data from two power system cases with 15359 and 30910 buses show that the power flow solution for one 1GHz Pentium III processor takes 1.04 seconds and 2.72 seconds respectively.[3] Unfortunately, the input cases used are not publicly available for direct comparison of the timing results.

Power systems analysis studies such as voltage stability analysis require power flow computations to examine the effect of scenarios such as load loss or generation loss for a given network. A large power network consisting of thousands of buses provides a multitude of input scenarios to consider. The number of input cases to consider amplifies the benefit of a reduction in execution time.

The use of commodity hardware cluster computing for power flow computation has been examined before[3][4]. Previous work focused on the solution of the linear system arising from the Newton-Raphson algorithm by developing a custom sparse matrix solver for the power flow computation. The linear solver is where the bulk of the time is spent in performing the power flow computation. Simulation results provided showed good speedup for the implementation.

This paper presents the alternative implemention of a parallel Newton-Raphson load flow program in terms of the hardware platform, its software components, the operations performed, the input cases used and some of the timing measurements taken to show speedup trends as the input size and number of processors are varied.

An implementation using widely available components has the benefit of building upon the improvements in its components and makes cluster computing more accessible for power systems analysis. The development of a parallel power flow program requires a parallel sparse matrix solver, an efficient linear algebra library, graph partitioning software for minimizing the data dependencies between the multiple processors, message passing software and the code that will properly divide the data and manage the communication between the different nodes.[4]

2. MATERIALS AND METHODS

2.1 Hardware and Software

The simulation setup involved nine computers that had the identical hardware configuration shown in Table 1.

Table 1
Hardware used for the Cluster

Processor	1.1GHz AMD Duron
Memory	128MB PC2100 (133MHz DDR) RAM
Video	Integrated, 8MB Shared video memory
Motherboard Chipset	NVIDIA nforce-220D
Hard Disk	40GB 5400 rpm
LAN	Realtek 8139C/D 10/100MBps Ethernet

The computers are connected with a 10/100 Mbps switch. The computer that receives commands from the external network is designated as the head node or root node. The message passing software assigns a process rank starting at 0 to each participating process running on the cluster. The first process runs on the head node.

Red Hat Linux 9[1] serves as the base operating system on top of which the Open Cluster Group's Open Source Cluster Application Resources (OSCAR) 3.0[2] is installed. The Message Passing Interface implementation used is provided by the Local Area Multicomputer/MPI 7.0.x[3], which is included in OSCAR 3.0. The optimized linear algebra routines of the Automatically Tuned Linear Algebra Software (ATLAS) 3.6.0[4] is used indirectly as it is linked with the Portable, Extensible Toolkit for Scientific Computation (PETSC) 2.2.0[5]. The graph partitioning routines of ParMETIS 3.1[6] are also linked in with PETSC and used to obtain the edge-cut reducing assignment of buses to processors. Argonne National

1 https://www.redhat.com/docs/manuals/linux/RHL-9-Manual/
2 http://oscar.openclustergroup.org/
3 http://www.lam-mpi.org/
4 http://math-atlas.sourceforge.net/
5 http://www.mcs.anl.gov/petsc/
6 http://www-users.cs.umn.edu/~karypis/metis/parmetis/

Laboratory's PETSc[6] toolkit provides routines to facilitate the solution of linear and non-linear systems arising from the solution to partial differential equations. The sparse linear solver component in the toolkit is utilized to solve the linear system that arises in the power flow solution. The default solver used is restarted GMRES, preconditioned with ILU(0) for the uniprocessor case and with the block Jacobi method for the multiprocess case. Default parameters were used for each linear solve.

The power flow application as a whole is written in C and linked with the libraries identified previously. The host operating system's C compiler and libraries were used to develop the power flow program.

2.2 Input cases

The benefits of cluster computing requires a problem large enough that the processing required to find the solution offsets the overhead introduced by using a cluster. In order to obtain a power system with a large number of buses, multiple instances of the IEEE 30-bus test case[1] inputs were connected together at the position of the swing bus. Only one swing bus is allowed in a power system so the swing buses for each other instance was converted to voltage regulated buses with a large MVAR generation capacity. Each input size is doubled to more clearly show the trend in speed up versus input size. There are 12 input cases with size ranging from $30\text{x}2^0$ buses to $30\text{x}2^{11}$ buses.

2.3 Timing Measurements

Elapsed time measurements were measured by saving time stamps between the segments of code of interest. The time stamps are taken using a PETSc routine with the resolution returned to be in the order of magnitude of $1\text{x}10^{-7}$s.

For the purpose of timing measurements, the operations performed in the power flow computation loop are divided into several phases. The operations performed in the power flow computation are shown in the flowchart in Figure 2.

The approach taken to develop the parallel power flow implementation began with a non-parallel implementation of the basic Newton-Raphson power flow algorithm. Portions of the implementation are modified for parallel execution while maintaining the consistency between the outputs of the original and modified versions. The processing phases that operate on the rows of the Jacobian matrix are modified to process only those rows corresponding to buses for which the current process is responsible.

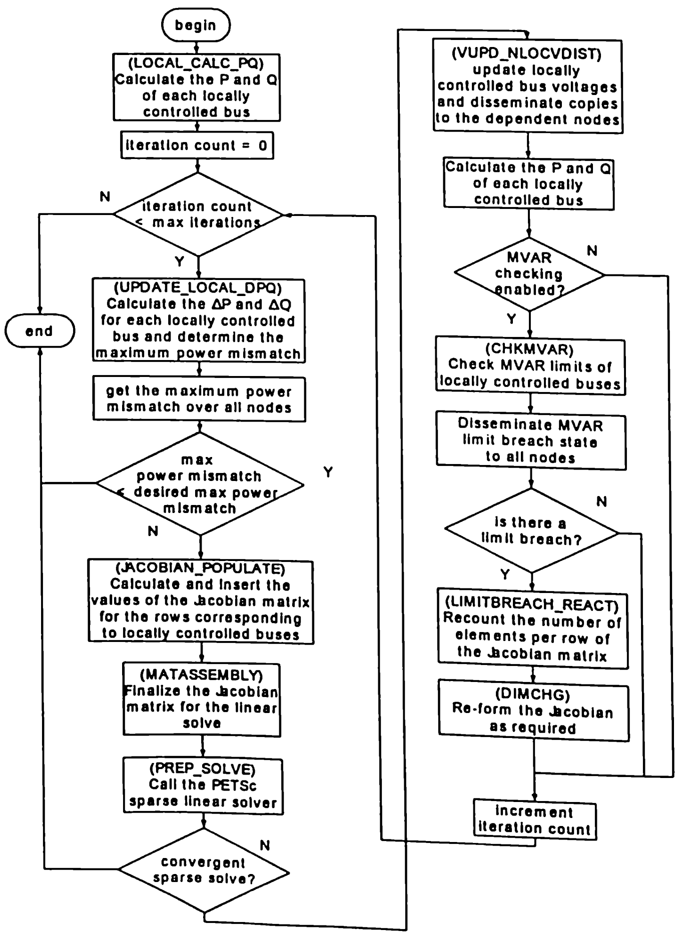
begin
(LOCAL_CALC_PQ) Calculate the P and Q of each locally controlled bus
iteration count = 0
iteration count ≤ max iterations
N
Y
end
(UPDATE_LOCAL_DPQ) Calculate the ΔP and ΔQ for each locally controlled bus and determine the maximum power mismatch
get the maximum power mismatch over all nodes
max power mismatch ≤ desired max power mismatch
Y
N
(JACOBIAN_POPULATE) Calculate and insert the values of the Jacobian matrix for the rows corresponding to locally controlled buses
(MATASSEMBLY) Finalize the Jacobian matrix for the linear solve
(PREP_SOLVE) Call the PETSc sparse linear solver
convergent sparse solve?
N
(VUPD_NLOCVDIST) update locally controlled bus voltages and disseminate copies to the dependent nodes
Calculate the P and Q of each locally controlled bus
MVAR checking enabled?
N
Y
(CHKMVAR) Check MVAR limits of locally controlled buses
Disseminate MVAR limit breach state to all nodes
Is there a limit breach?
N
Y
(LIMITBREACH_REACT) Recount the number of elements per row of the Jacobian matrix
(DIMCHG) Re-form the Jacobian as required
increment iteration count

Figure 2. Flowchart for the Parallel Power Flow Calculation

2.4 Data Decomposition

The buses of the power system are associated with a voltage, a real power P and imaginary power Q. The dependence between the parameters of different buses is determined by the connectivity of the buses, which is represented in the admittance matrix Y_{bus}. This dependence implies a communication operation if the required values are non-local to the process that needs it. One of the challenges to producing an effective cluster application is the decomposition of the data and computation to reduce communication between the nodes. The Y_{bus} of the power system network is produced in the compressed sparse row format for use with the graph partitioning library. The identification of the groups of interconnected buses while minimizing the inter-group connectedness is the important requirement for the graph partitioning process.

The Y_{bus} elements are stored as utilized in cartesian form $G + jB$ and the formulation of the Newton-Raphson mismatch equation is modified as shown in (11).

$$\begin{bmatrix} \Delta P \\ \vdots \\ \Delta Q \end{bmatrix} = \begin{bmatrix} \frac{\partial P}{\partial \angle V} & |V|\frac{\partial P}{\partial |V|} \\ \frac{\partial Q}{\partial \angle V} & |V|\frac{\partial Q}{\partial |V|} \end{bmatrix} \begin{bmatrix} \Delta \angle V \\ \vdots \\ \frac{\Delta |V|}{|V|} \end{bmatrix} \qquad (11)$$

The expressions used for each element of the Jacobian matrix is modified to take advantage of the calculated values of bus real power P and imaginary power Q shown in equations (12) to (15) with $\theta_{km} = \angle V_k - \angle V_m$.[7]

$$\frac{\partial P_k}{\partial \angle V_m} = \begin{cases} |V_k||V_m|(G_{km}\sin(\theta_{km}) - B_{km}\cos(\theta_{km})) & m \neq k \\ -Q_k - B_{kk}|V_k|^2 & m = k \end{cases} \qquad (12)$$

$$|V_k|\frac{\partial P_k}{\partial |V_m|} = \begin{cases} |V_k||V_m|(G_{km}\cos(\theta_{km}) - B_{km}\sin(\theta_{km})) & m \neq k \\ P_k + G_{kk}|V_k|^2 & m = k \end{cases} \qquad (13)$$

$$\frac{\partial Q_k}{\partial \angle V_m} = \begin{cases} -|V_k||V_m|(G_{km}\cos(\theta_{km}) + B_{km}\sin(\theta_{km})) & m \neq k \\ P_k - G_{kk}|V_k|^2 & m = k \end{cases} \qquad (14)$$

$$|V_k|\frac{\partial Q_k}{\partial |V_m|} = \begin{cases} |V_k||V_m|(G_{km}\sin(\theta_{km}) - B_{km}\cos(\theta_{km})) & m \neq k \\ Q_k - B_{kk}|V_k|^2 & m = k \end{cases} \qquad (15)$$

The calculation of each element of the Jacobian matrix requires elements from the Y_{bus}. The Y_{bus} is replicated across all processing nodes to eliminate communication of Y_{bus} values between different nodes.

The choice of linear solver routine used for the power flow implementation requires that the matrix be divided row-wise. The order of the rows in the Jacobian matrix is rearranged as shown in (16) to make the ΔP and ΔQ of buses occupy adjacent rows. The ordering of the rows of the Jacobian is also adapted to make rows corresponding buses belonging to the same processing node contiguous. These two measures reduce the communication requirements of calculating the real and reactive power mismatches. [5]

$$\begin{bmatrix} \Delta P_2 \\ \Delta Q_2 \\ \vdots \\ \Delta P_n \\ \Delta Q_n \end{bmatrix} = \begin{bmatrix} \frac{\partial P_2}{\partial \angle V_2} & |V_2|\frac{\partial P_2}{\partial |V_2|} & \cdots & \frac{\partial P_2}{\partial \angle V_n} & |V_n|\frac{\partial P_2}{\partial |V_n|} \\ \frac{\partial Q_2}{\partial \angle V_2} & |V_2|\frac{\partial Q_2}{\partial |V_2|} & \cdots & \frac{\partial Q_2}{\partial \angle V_n} & |V_n|\frac{\partial Q_2}{\partial |V_n|} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{\partial P_n}{\partial \angle V_2} & |V_2|\frac{\partial P_n}{\partial |V_2|} & \cdots & \frac{\partial P_n}{\partial \angle V_n} & |V_n|\frac{\partial P_n}{\partial |V_n|} \\ \frac{\partial Q_2}{\partial \angle V_2} & |V_2|\frac{\partial Q_n}{\partial |V_2|} & \cdots & \frac{\partial Q_n}{\partial \angle V_n} & |V_n|\frac{\partial Q_n}{\partial |V_n|} \end{bmatrix} \begin{bmatrix} \Delta \angle V_2 \\ \frac{\Delta |V_2|}{|V_2|} \\ \vdots \\ \Delta \angle V_n \\ \frac{\Delta |V_n|}{|Vn|} \end{bmatrix} \qquad (16)$$

3. RESULTS AND ANALYSIS

For each problem size, the power flow program is run varying the number of processors from 1 to 9. The results for input sizes less than $30x2^5$ buses are not displayed since the results show that these input sizes are too small to derive any benefit from running on the cluster. The elapsed time of the program fragments of interest are collected and speedup and efficiency is calculated. [5]

3.1 The Power Flow Computation

The power flow computation elapsed time is measured from the time the necessary data structures are prepared for both processing and communication until just before the data structures are destroyed. The solution process took 5 Newton-Raphson iterations to reduce the maximum power mismatch to less than $1x10^{-6}$ per unit. Figure 3 shows the variation of elapsed time with respect to the number of processors using logarithmic time (Y) and data size axis. Input sizes greater than $30x2^6$ buses exhibit a decrease in execution time when utilizing multiple processors. For the largest input size, the average measured value of 10 runs is shown. The maximum spread of the measurement over the runs is shown on the data series. For the rest of the input sizes, the data used is the first of 10 runs.

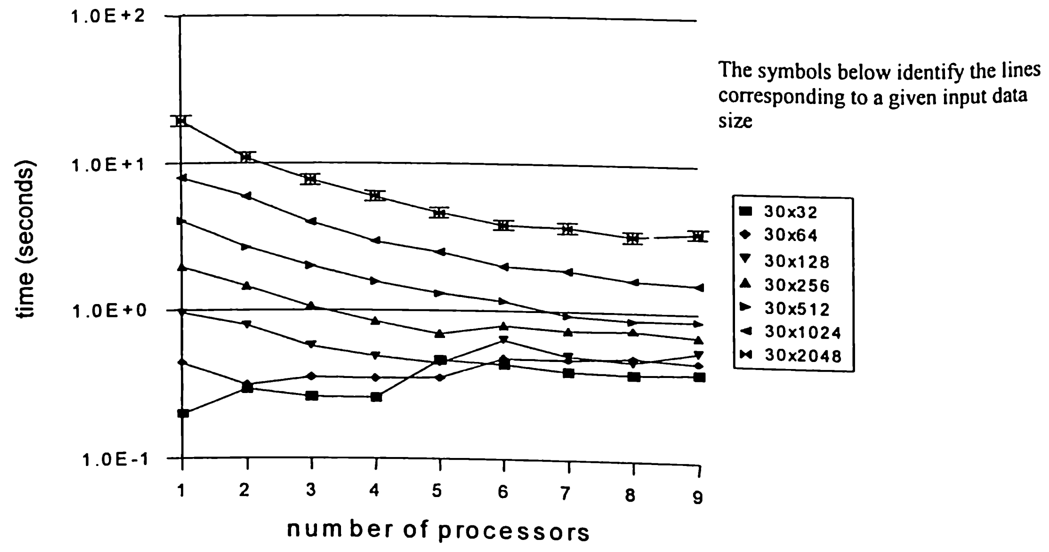

Figure 3. Power Flow Computation Elapsed Time
Each line represents data for a given input size

$$speedup = \frac{sequential\_execution\_time}{parallel\_execution\_time} \quad (17)$$

The speedup is calculated as defined in equation (17), except that a separate sequential program was not created and the sequential execution time is interpreted as the execution time of the parallel program running on a single processor. It can be seen from Figure 4 that a power system size greater than $30x2^5$ buses is required for the cluster to provide a benefit over the single process case.

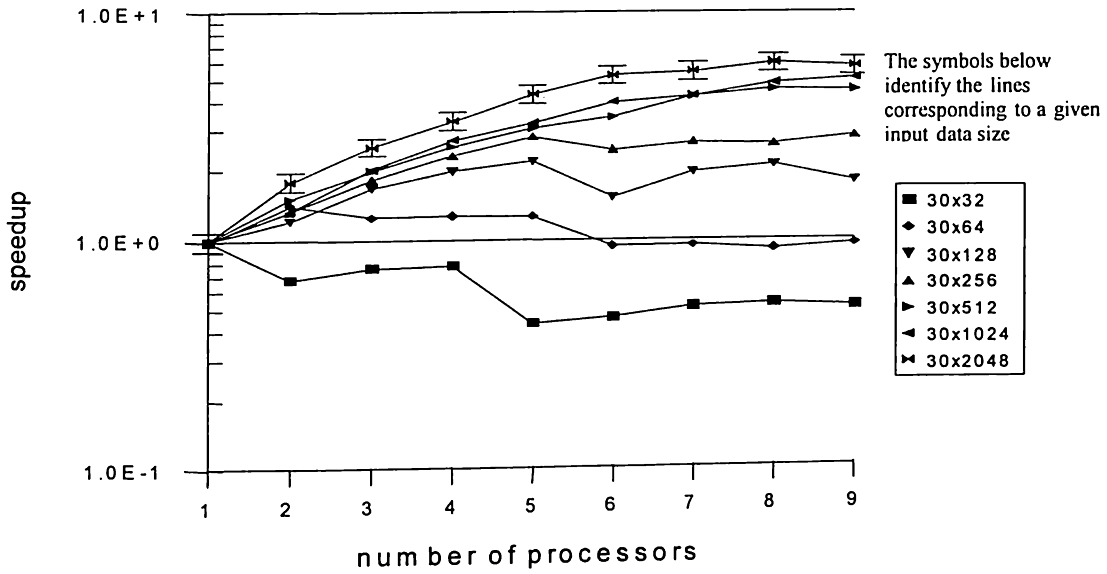

Figure 4: *Speedup vs Number of Processors per Input Size*

$$efficiency = \frac{speedup}{p} \qquad \textbf{(18)}$$

Based on the efficiency calculated according to equation (18) and shown in Table 2, for this implementation of power flow computation, processor utilization decreases with an increase in the number of processors. This means that the communication and processing overhead of working with multiple processors limits the processor utilization even as the problem size increases up to $30x2^{11}$ buses. Larger input sizes are necessary to provide enough computational load per processor to offset the communication overhead. However, there is still a decrease in elapsed time of the power flow computation for higher number of processors despite the lower processor utilization efficiency.

Table 2

Power Flow Computation Efficiency

input size (number of buses)	Number of processors								
	1	2	3	4	5	6	7	8	9
30x32	100.00%	34.07%	25.48%	19.48%	8.65%	7.63%	7.27%	6.56%	5.65%
30x64	100.00%	71.02%	41.88%	31.86%	25.34%	15.55%	13.44%	11.33%	10.59%
30x128	100.00%	60.87%	55.98%	49.44%	43.45%	25.20%	27.63%	26.05%	19.61%
30x256	100.00%	67.12%	61.06%	57.71%	56.16%	40.68%	37.63%	32.32%	31.25%
30x512	100.00%	75.66%	66.50%	63.85%	61.22%	57.12%	60.36%	56.64%	49.84%
30x1024	100.00%	67.34%	67.34%	68.03%	64.78%	66.33%	60.47%	60.24%	56.26%
30x2048	100.00%	86.70%	81.19%	79.41%	81.29%	83.11%	72.78%	72.92%	61.76%

3.2 Overall Performance

The total elapsed time for the power flow implementation, including the file input processing overhead, is shown in Table 3. The contribution of the non-parallel file input processing prevents the further reduction of elapsed time below 11.5 seconds for the largest input case with a number of processors greater than 6.

Table 3
Over All Elapsed Time (Load File and Calculation)

Input size (number of buses)	Number of processors								
	1	2	3	4	5	6	7	8	9
30x32	2.95E-1	3.84E-1	3.61E-1	3.63E-1	5.71E-1	5.51E-1	5.08E-1	5.03E-1	5.15E-1
30x64	6.12E-1	4.96E-1	5.46E-1	5.57E-1	5.60E-1	6.93E-1	6.92E-1	7.15E-1	7.11E-1
30x128	1.29E+0	1.16E+0	9.67E-1	9.06E-1	8.63E-1	1.08E+0	9.35E-1	1.12E+0	1.03E+0
30x256	2.58E+0	2.21E+0	1.87E+0	1.69E+0	1.61E+0	1.77E+0	1.67E+0	1.74E+0	1.73E+0
30x512	5.22E+0	4.20E+0	3.64E+0	3.26E+0	3.14E+0	3.01E+0	2.79E+0	2.79E+0	2.93E+0
30x1024	1.04E+1	9.09E+0	7.27E+0	6.44E+0	6.06E+0	5.82E+0	5.76E+0	5.52E+0	5.76E+0
30x2048	2.38E+1	1.74E+1	1.46E+1	1.31E+1	1.22E+1	1.16E+1	1.17E+1	1.16E+1	1.20E+1

The calculated speedup values for the power flow implementation are plotted in Figure 5. Speedup values greater than 1 is achieved only for input sizes of 30x2^5 buses with number of processors less than 6 and for input sizes greater than 30x2^6 buses. The file input constitutes a large serial component of computation that limits the speedup to a maximum of approximately 2.

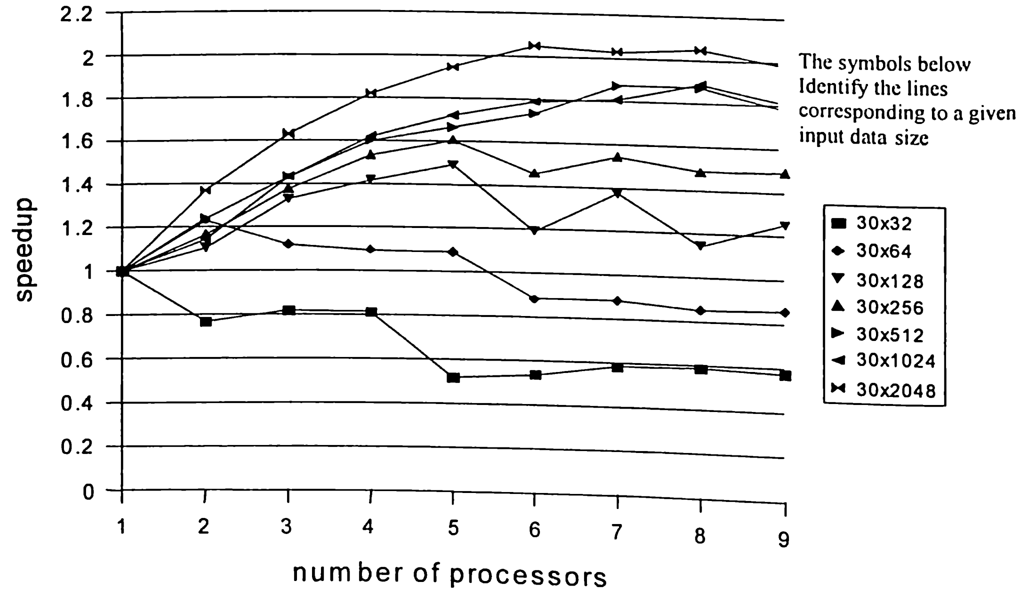

Figure 5. Over All Speedup (Load File and Calculation)

4. CONCLUSIONS AND RECOMMENDATIONS

The use of widely available computing libraries in conjunction with the decomposition of the power system in terms of groups of buses reducing inter-group connectivity, and the re-ordering of the rows of the Jacobian matrix to make contiguous the ΔP and ΔQ entries of a bus resulted in an improvement in speedup for an input size greater than $30x2^6$ buses.

The magnitude of the time spent on loading the file compared to power flow calculation time indicates that more efficient I/O is desirable as the input size further increases in order not to diminish the gains provided by solving the problem in parallel. More efficient and possibly parallel file loading is necessary to reduce the amount of time spent loading the input file. The reduction of execution time continues until the amount of buses per node drops to about $30x2^6$ buses, after which the per-iteration execution time improvement in the power flow computation becomes less than a second.

Further development of the implementation can then be focused on supporting larger systems, further optimizing performance by tuning the parameters of the software components utilized, evaluating the effects on simulation speed of more detailed modeling of a power system, as well as the development of parallelizable power flow input format and routines.

5. NOMENCLATURES

symbol	Description	Units
P	real power	Watt
Q	reactive power	Var
V	complex voltage	Volt
$\lvert V\rvert$	the magnitude of the complex value V	Volt
$\angle V$	the angle of the complex value V	Radians/Degrees
Y, y	complex admittance value	Siemens
Y_{bus}	bus admittance matrix	-
G	Conductance component of admittance	Siemens
B	Reactance component of admittance	Siemens

Subscripts, Superscripts and abbreviations	
k	referring to bus k
m	referring to bus m
km	referring to a quantity (e.g. admittance) related to bus k and m
MHz	measure of frequency, megahertz = 10^6 cycles/second
MB	measure of computer data storage, megabyte = 2^{20} bytes

REFERENCES

1 H. Saadat. "Power Systems Analysis". McGraw-Hill Book Companies Inc. (1999)

2 D. Tylavsky, et. al. Parallel Processing in Power Systems Computation, *IEEE Transactions on Power Systems*, **vol 7**, number 2, 629-638. (1992)

3 F. Tu, A.Flueck. A Message-Passing Distributed-Memory Parallel Power Flow Algorithm. *IEEE Power Engineering Society Winter Meeting*, **vol 1**, pp 211-216. (2002)

4 F. Tu, A.Flueck. A Message-Passing Distributed-Memory Newton-GMRES Parallel Power Flow Algorithm, in *Power Engineering Society Summer Meeting*, **vol 3**, pp 1477-1482. (2002)

5 R. B. Uy, "A Parallel Newton-Raphson Load Flow for a Cluster of Workstations", Masters Thesis, University of the Philippines, Diliman, Quezon City. (2005)

6 Balay, Satish, et. al. PETSc home page. <http://www.mcs.anl.gov/petsc> (April 2004)

7 J. Arrillaga and N. Watson. "Computer Modelling of Electrical Power Systems 2nd edition". West Sussex, England: John Wiley and Sons Ltd., (2001)