Using Local Grid and Multi-core Computing in Electromagnetic Transients Simulation

R. Singh, A. M. Gole, P. Graham

J. C. Muller, R. Jayasinghe, B. Jayasekera and D. Muthumuni

***Abstract* — This paper presents a method to implement the Electromagnetic Transients (EMT) simulation algorithm on a multi-core or grid processing platform. The simultaneous use of multiple processors to divide and solve portions of the simulation has the potential to speed up the overall simulation significantly. However, communication bottlenecks can reduce its effectiveness. This paper uses the Electric Network Interface (ENI), which is a TCP based communication interface implemented using transmission lines (t-lines) as natural interface ports. ENI allows the sub-systems on either side of t-lines to be simulated on separate processors on both local host and distributed computers connected by standard local area networks (LAN). Using several implementation examples, it is shown that the communication bottleneck is significant when the execution time for each subsystem is small, and can result in slower simulations than on a single processor. However, with sufficiently large subsystems, there is significant speed-up to the overall execution time.**

***Keywords*: Electromagnetic Transients Simulation, Parallel Computing, Grid Computing, Problem Decomposition, Electric Network Interface (ENI).**

I. Introduction

HIGH Performance Computing (HPC) has always attracted users, from various fields of study, who need to solve large and complex computational problems. Often, depending on the fields of study and the nature of the problem, practical systems cannot be built before assessment of the robustness of the system is accomplished. Simulated studies are a great way of analyzing the accuracy, efficiency and robustness of any such problem. In Power Systems, often, the nature and complexity of the problem leads to detailed models that rapidly increase the size of the equations to be solved. Such analyses, when performed using simulation software that is built to use conventional computers (single CPU) take a long time. With the advancement in multiprocessor computer technologies and such computers becoming commodity hardware, power systems simulation software needs to be able to harness the multiple-core computing power generally available. Electromagnetic transient (EMT) simulation uses highly accurate models of power equipment [1]. These models typically require small time steps, and are hence computation-ally intensive. In EMT simulation transmission lines can be used to separate a system into coupled sub-systems which can then be solved on a single core each and exchange results at the end of computation. This is due to the finite travel time of information (imposed by the relativistic speed limit) across the line. Hence, each of these subsystems can largely be solved independently and thus in parallel. This property has been exploited in the past to construct real-time simulators using specialized hardware [2]. This paper discusses implementation of a software parallelization on a combined multi-core/grid platform. The methodology and performance of this approach are reported in this paper.

R. Singh is with the University of Manitoba, Electrical Engineering Department, Winnipeg, MB, Canada (Email: rsingh@mhi.ca).

A. M. Gole is with the University of Manitoba, Electrical Engineering Department, Winnipeg, MB, Canada (Email: gole@cc.umanitoba.ca).

P. Graham is with the University of Manitoba, Computer Science Department, Winnipeg, MB, Canada (Email: pgraham@cs.umanitoba.ca).

J. C. Muller is with Manitoba HVDC Research Centre, Winnipeg, MB, Canada (Email: cmuller@pscad.com).

R. Jayasinghe is with Manitoba HVDC Research Centre, Winnipeg, MB, Canada (Email: jayas@hvdc.ca).

B. Jayasekera is with Manitoba HVDC Research Centre, Winnipeg, MB, Canada (Email: bathiyaj@mhi.ca).

D. Muthumuni is with Manitoba HVDC Research Centre, Winnipeg, MB, Canada (Email: dharshana@hvdc.ca).

Paper submitted to the International Conference on Power Systems Transients (IPST2013) in Vancouver, Canada July 18-20, 2013.

II. Grid Computing

The term “The Grid” [3] [4] [5] was suggested in the mid 1990's to denote the construction of a nation-wide computing infrastructure analogous to the power grid. A very large scale distributed computing system was envisioned connecting geographically distributed machines and network resources available at various institutions and organizations. Grid computing systems, even at much smaller scale (as in this paper), provide a distributed computing environment consisting of shared distributed computing resources often ranging from off-the-shelf personal computers (now with multiple cores) to high-end clusters (specially designed computers with high-end processors and communication backplane). Such small-scale grid environments are easily affordable to engineering organizations or can even be built using LAN interconnected workstations already available.

In this paper ‘grid computing’ refers to the use of a parallel algorithm run on locally distributed computing resources. This could be one physical location (single machine with multiple cores) or on several locations (multiple machines with one or several cores). Although the examples shown were implemented on a single computer with multiple cores, the implemented algorithm can be used in both cases.

A. Parallel Computing using the grid

Parallel Computing is normally an integral part of grid computing. By definition, it is the solution of a computational problem by decomposing it into multiple parts and executing each part simultaneously on a separate computing resource [6]. In contrast, in traditional sequential computing, the instructions are executed one after the other on a single processor. As task execution is sequential, even non-interdependent tasks have to wait until the processor finishes other tasks. This limitation is removed with parallel processing, thereby often speeding up the computation. A decade ago, expensive parallel computing machines having several CPUs were used to implement such parallel algorithms. Recently, with multiple core machines becoming a commodity, more affordable parallel computing platforms are now available.

There are several models of parallel computing, such as shared memory based parallel programming and distributed memory based parallel programming. In the former (shared memory architecture), a number of processors use the same shared memory. In the latter (distributed memory programming model), each processor has its own memory. As the latter is more common, and can be created by simply harnessing the power of separate computers in a grid, it is the one used in this paper. A block diagram of this architecture is shown in Fig. 1, where separate but connected computers each model a process and have their own dedicated memory. Within this architecture style, two approaches are considered: In the first, a single machine with multiple cores and very fast communication links between them was used. However, the maximum speed-up possible is limited to the number of cores (which is relatively small). In the second approach, multiple machines are connected over a local area network (LAN). Here, an arbitrary number of parallel CPUs become available; however this is at the price of reduced communication speed.

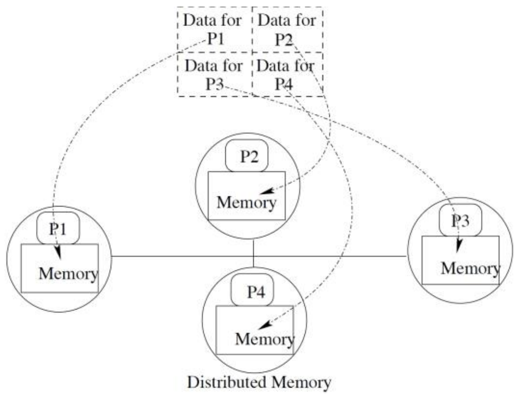

Fig. 1. Distributed Memory Architecture

For communication between the processes in an EMT simulation, a message passing based interface is needed that allows distributed processes to exchange electrical values with one another. Fig. 1 also shows how, in distributed memory systems, the data is localized to specific processors.

III. Adapting Simulation Software for Grid Computing

In this research we used the PSCAD/EMTDC [7] software as the EMT solver. To manage the grid computing environment the Xoreax Grid Engine (XGE) software [8] was used. The grid engine consists of a coordinator (that schedules the tasks) and several agents (that execute the tasks). The computing environment used consisted of 40 cores (by combining various desktop PCs) running at different clock speeds connected over a standard local area network (LAN). The interface to the grid engine is an XML script file that describes the processes as tasks and how they are to be scheduled on the grid. The grid coordinator controls the scheduling of tasks on corresponding agent computers.

The EMT solver (EMTDC) is a sequential program. In this paper, it was adapted for use in parallel computing environment.

Each core in the grid is assigned one specific circuit to solve, corresponding to a subsystem separated from other subsystems by transmission lines (t-lines) or cables. The t-lines introduce a natural time delay between sending of the electrical signal to receiving it and hence become a natural point to split the system into parts that may execute concurrently. The split parts of a simulation can then each be solved independently on a single processor core, with a separate EMTDC instance running on each core. The computations for the t-line or cable model itself are assigned to the core simulating any one of the interconnected parts. A communication interface had to be designed for the individual parts running on separate processor cores to exchange values of the electrical quantities at their interfaces. Grid coordination software (Xoreax) was used to spawn the individual EMTDC instances. Minor modifications were added to the PSCAD coordinator to facilitate the message passing between the processes. This is referred to as the Electric Network Interface (ENI) discussed below.

A. Electric Network Interface (ENI)

The ENI is a TCP Socket based communication interface developed for the EMTDC solver that enables communication of electrical values between several sub-system simulations each running on its own processor core. The concept is illustrated in Fig. 2. Consider one of the examples used in this paper – a hypothetical 273 bus system constructed by connecting 7 IEEE 39 bus system kernels [9] including generators, transmission lines and loads, connected via t-lines to form a 273 bus system.

Fig. 2 (A) shows a traditional serial EMT solver consisting of the models for 7 such IEEE 39 bus systems bundled in a single executable program (i.e. "binary") and running on a single CPU. Fig. 2 (B) shows the implemented Parallel EMT Solver where each of the seven IEEE 39 bus system models are solved by one EMT Solver each of which is running on one processor core. All the solvers are connected via ENI. The t-line connections in the system define the placement of each TCP-based ENI communication pipe between the solvers.

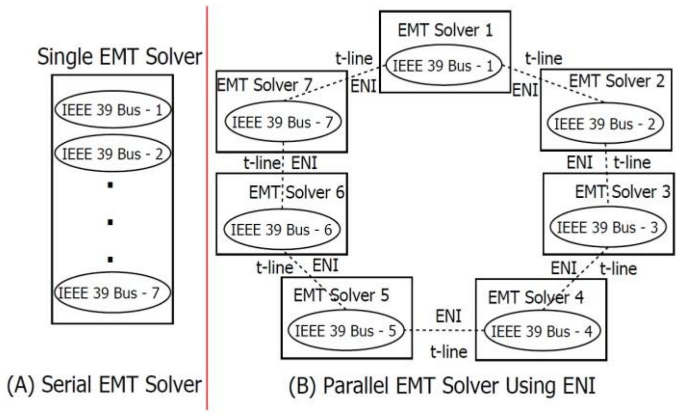

Fig. 2. Electric Network Interface

The connected EMT Solvers exchange electrical values using this communication pipe. The t-line model can be solved by either of the EMT Solvers at the ends of their ENI pipe. Comparing Figs. 2 (A) and 2 (B) the parallelism of the solution is evident. However, the parallelism does not automatically imply faster simulation, because it also introduces an additional layer of inter-processor communication overhead that did not exist in the serial solver. Nevertheless, results presented later in the paper confirm that such ENI based parallel simulation of a sufficiently large power system will execute faster than a serial simulation.

IV. Parametric studies

To test the effectiveness of ENI, simulation experiments were conducted with standard IEEE test bus systems by running the cases on a single processor core and on multiple processor cores using the ENI communication interface. In successive experiments, the complexity and size of the systems were increased. The goal of these experiments was to analyze the speed-up achieved in each test case and study the effect of communication overhead involved when using ENI. It will be shown that there is a critical size above which the computational advantage outweighs the communications overhead resulting in faster simulations.

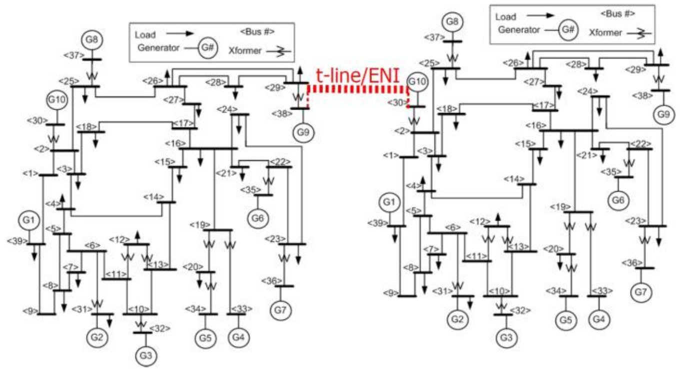

Fig. 3. Pictorial Representation of construction of larger systems using the IEEE 39 Bus System as a kernel

The test cases for the experiments were constructed from the standard IEEE 14 bus system [10], IEEE 39 bus system, IEEE 118 bus system [10] and IEEE 300 bus system [10]. The standard IEEE test systems used are as follows:

i. The IEEE-14 bus test system consists of 11 loads, 5 synchronous machines, 17 lines and 36 branches.
ii. The IEEE-39 bus test system consists of 39 loads, 10 synchronous machines, 46 lines and 95 branches.
iii. The IEEE-118 bus test system consists of 91 loads, 54 synchronous machines, 177 lines and 331 branches.
iv. The IEEE-300 bus test system consists of 195 loads, 69 synchronous machines, 304 lines and 674 branches.

Test systems of larger size were constructed from the systems described above, by connecting several identical systems using transmission lines. For example, Fig 3 shows the construction of a 78 bus system by connecting together 2 IEEE 39 bus systems. Here bus 30 of one system is connected to bus 38 of the other using a t-line (dotted line), which indicates the need for an ENI interface during parallel simulation.

A. *Experiment Methodology*

The experimental setup was as shown in Fig. 2. Fig. 2 (A) represents the sequential execution of a single binary on a single processor core and Fig. 2 (B) represents executing multiple binaries in parallel using ENI on multiple cores.

To conduct the experiments, sequential execution of the cases on a single processor core and parallel execution of the same case on multiple processor cores (up to a maximum of 8) using ENI was compared. A typical modern engineer's desktop has multiple cores in it. ENI uses TCP Socket based communication interface and thus its use is not restricted to a single multi-core machine. For example, several desktop PC's with multiple cores can be interconnected using a TCP/IP LAN to provide an environment with hundreds of cores.

Systems of different sizes were simulated. For example, simulations were carried out for a single 39 bus system, then the 39 bus system kernel was duplicated into a 78 bus system (see Fig. 3), then triplicated to a 117 bus system and so on up to 39 × 7 busses. These were implemented for:

a) Single processor and single binary, as in Fig. 2 (A).
b) Multiple binaries implemented on multiple processors using ENI for communication.

Similar experiments were performed for 14, 78, 118 and 300 bus systems. For example, for seven interconnected 300 bus system kernels, a large 2100 bus system was executed sequentially on a single processor core and in parallel on seven processor cores.

B. *Observations*

The execution for each simulation approach was measured. An index called the speed-up factor was used to determine the advantage afforded by the parallel implementation. This is the ratio of execution time with the parallel approach to the execution time for a single-processor implementation. The communication overhead was also measured. All experiments were done on an 8 core, 2.4 GHz Intel Xeon based system with 16 GB memory. ENI communication was used on the local machine for consistency with multi-machine implementations even though a specialized, machine-local communication mechanism would have provided lower latencies.

1) *IEEE 14 Bus System*

Fig. 4 shows the results from simulating a network composed of several identical IEEE 14 bus system interconnected via t-lines using ENI on multiple cores; and also on a single core. For the multi-core simulations, additional cores were utilized as the system size grew. That is, the first test system was a single 14 bus system simulated on one core, the second was a 2 × 14 bus system, simulated on two cores, and so on, the last being a 7 × 14 bus system, simulated on 7 cores. Each core therefore simulates a single 14 bus system using a separate EMTDC solver. Note that, each processor handles essentially the same computation load (each models the identical 14 bus kernel system). This provides a method of estimating the communication overhead - the difference between the time required for multi-core simulation, minus the time required to simulate the 14 bus system on a single core.

Plotted as a function of the network size (number of 3-phase buses) are the following:

a) the single-core execution time (Fig. 4)

b) total execution time on the multiple cores (Fig. 4)

c) largest communication overhead including latency and wait times (‘b’- Execution Time for 1 kernel on 1 core) (Fig. 4)

d) speed up factor [ratio of ‘b’ to ‘a’] (Fig. 5)

The horizontal axis is the system size in numbers of 14 bus systems connected to form the larger system.

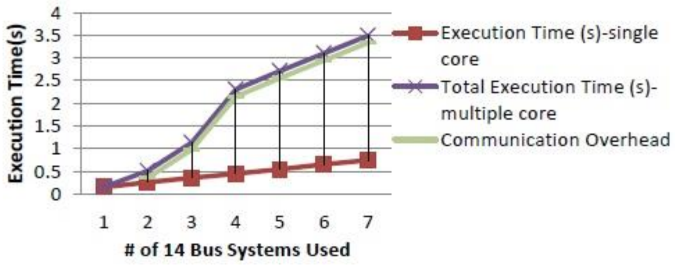

Fig. 4. 14 Bus System Kernels: Parallel vs. Sequential Execution Times and Communication Overhead of ENI

Fig. 5, shows that for this problem, the multiple cores do not provide any advantage as the speed-up factor is less than unity. This is because the solution time for any subsystem on a single core is about 0.5 s which is comparable to; or, as the total system size grows, even smaller than the communication overhead. Thus the communication time dominates and parallelization leads to no advantage.

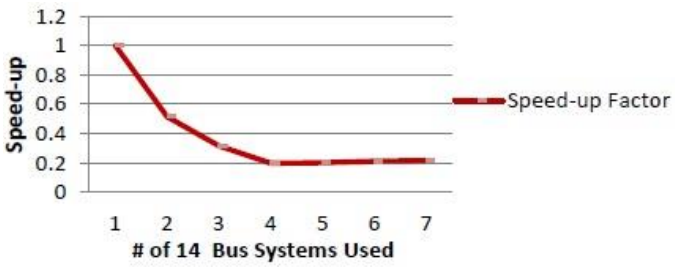

Fig. 5. 14 Bus System Kernels: Speed-up Factor

2) *IEEE 39 Bus System*

The experiments as discussed in the preceding sub-section were repeated, but this time with a larger 39-bus kernel replacing the 14 bus kernel. The well known IEEE 39 Bus System was used for the kernel system. Fig. 6 shows the results. The increase of the size and complexity of the kernel system simulated on each core now requires a longer simulation time as compared with the 14 bus system. Hence the communication overhead becomes a smaller fraction of the execution time. However, the communication overhead is still large such that single core outperforms the multiple core execution using ENI.

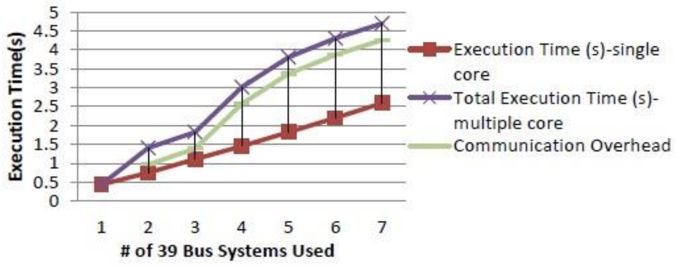

Fig. 6. 39 Bus System Kernels: Parallel vs. Sequential Execution Times and Communication Overhead of ENI

The speed-up factor is still below unity (Fig. 7), so the size of the 39 bus system is still not large enough to gain advantage using ENI on multiple machines.

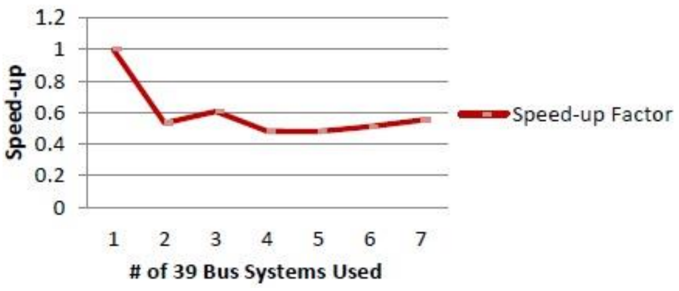

Fig. 7. 39 Bus System Kernels: Speed-up Factor

3) *IEEE 78 Bus System*

A 78 Bus kernel System was created by doubling the size of the 39 bus systems and the same tests as described above were conducted. As shown in Fig. 8, the single core execution and multiple cores execution time plots now essentially overlap each other, resulting in a near-unity speed-up factor as the system size increases.

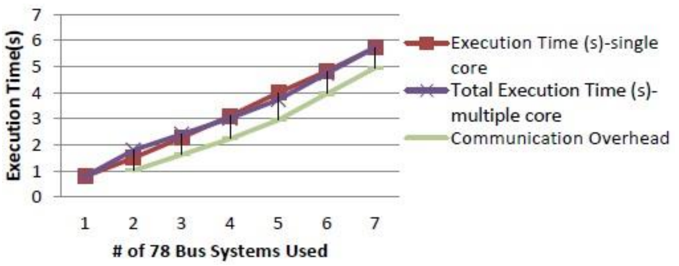

Fig. 8. 78 Bus System Kernels: Parallel vs. Sequential Execution Times and

Communication Overhead of ENI

The observations suggest that the 78-bus kernel size is the break-even system size, where the speed-up (shown in Fig. 9) due to the use of more cores exactly balances the disadvantage of increasing communication delay. Of course, the system cost is higher so parallelization is still ineffective. Using a larger kernel size will result in useful improvements in overall simulation times, because the impact of the communication overhead will be outweighed by the computation savings due to parallelism.

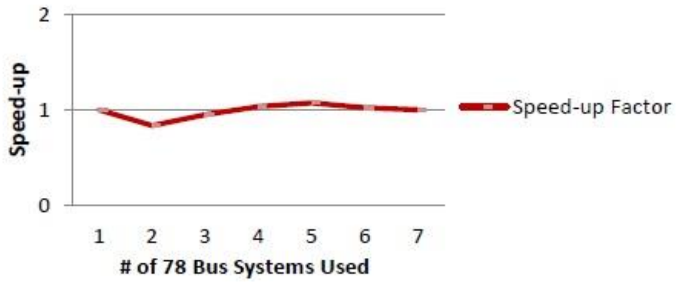

Fig. 9. 78 Bus System Kernels: Speed-up Factor

4) IEEE 118 and 300 Bus System

Experiments using much larger system kernels, such as IEEE 118 and IEEE 300 bus systems, were performed. As shown in Fig. 10, the parallel execution over ENI out-performs the single core execution and the speed-up factor now increases with the increase in the overall system size. However, the speed-up (see Fig. 11) is still not equal to the number of cores used as it would be with negligible communication overhead.

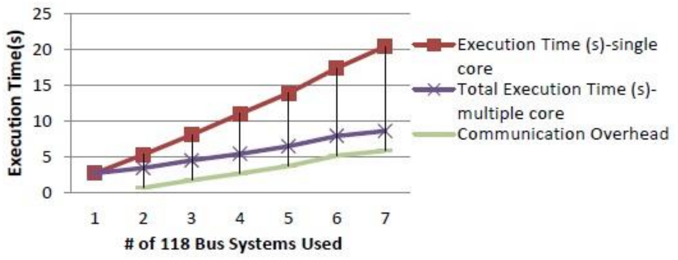

Fig. 10. 118 Bus System Kernels: Parallel vs. Sequential Execution Times and Communication Overhead of ENI

With a 300 bus kernel, as shown by the results plotted in Fig. 12, the communication overhead is very small compared to the execution time and the speed-up factor increases linearly and is essentially equal to the number of processors used, as shown in Fig. 13. This is close to the theoretical maximum speed-up that can be achieved. Thus, the use of t-lines to decompose a model provides good parallel speed-up as long as the resulting sub-models are reasonably large and hence compute intensive.

C. Partitioning a large system for ENI computing

The above analysis gives some guidance for partitioning a system for ENI computing. It is not always beneficial to partition it into several small systems to accommodate all the available cores. It is always important to make sure that the core size is larger than 78 (for the tested models) before such a partition is attempted. For example on an 8-core machine, a 320 bus system partitioned into four 80 bus systems each implemented on 1 core would require smaller simulation time than distributing it onto all 8 cores, even though the remaining 4 cores would remain idle.

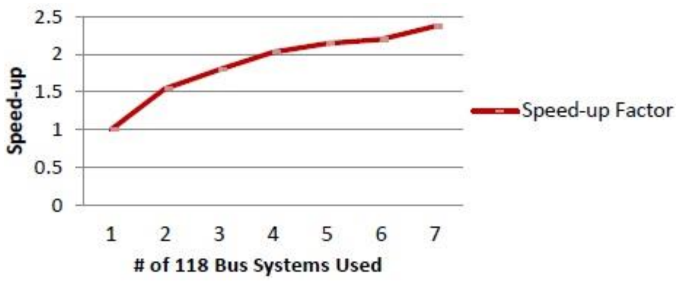

Fig. 11. 118 Bus System Kernels: Speed-up Factor

Other partitions may also yield higher speedups (e.g. 320 bus system split into 3 approximately equal size systems implemented on 3 cores, or split into two 160 bus systems on 2 cores). In such situations, the computation time would increase but the communication time would decrease. In general the optimum number of processors to be used depends on the relative proportion of the computing and communication times.

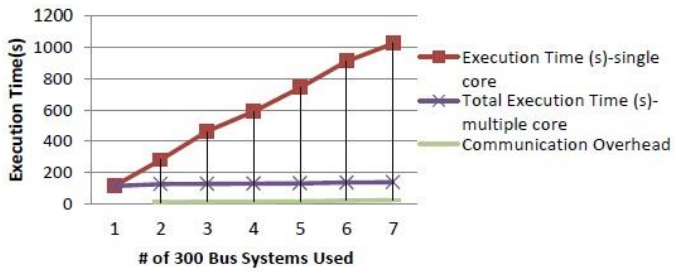

Fig. 12. 300 Bus System Kernels: Parallel vs. Sequential Execution Times and Communication Overhead of ENI

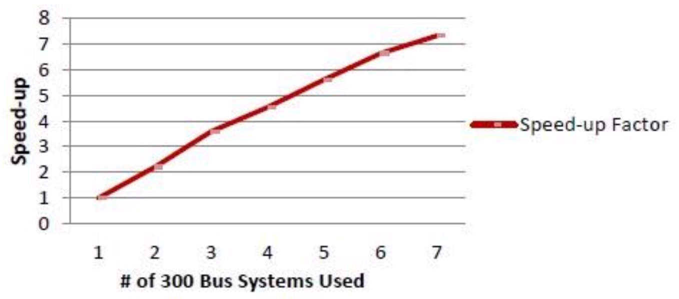

Fig. 13. 300 Bus System Kernels: Speed-up Factor

V. TECHNOLOGY AND MODEL DEPENDENT ISSUES

A. Technology related issues

The results in the paper are technology specific. The critical system size of 78 could easily be smaller with improved technology that reduces communication overheads.

Similarly, a change of model that required more or less computation time would change the critical system size. The implemented ENI interface was only tested on a multiple core computer environment. However, ENI also works in a grid based distributed computing environment where multiple computers (single or multi-core) are interconnected over a local area network (LAN). In that case inter-computer communication on the LAN would be slower than that between cores on the same computer. Hence a single 'critical' kernel size would not be possible to define, but as a rule, the average communication time would increase compared to that for multiple cores on a single computer, thereby requiring larger kernel sizes before becoming effective. It is also easy to visualize a hierarchical structure where smaller kernels are modeled on local cores, and the resulting larger systems are connected across the grid. For example, one may model 6 × 118 bus systems on a local computer with say 8 cores, and connect via the LAN to another 6 × 118 bus system modeled on another computer with 8 cores, thereby minimizing the LAN communication.

B. Model related issues

For purposes of experimentation and parametric analysis, the paper presented a contrived network of exactly identical kernels connected by t-lines. In an actual simulation, the kernel sizes would all typically be different resulting in various speed-up factors. Also, no power electronic devices such as HVDC converters or elements requiring iterative calculation such as surge arrestors were included in the test systems. With such systems, the computation time would increase for any kernel, and the critical system size would likely be smaller than 78.

VI. Conclusions

This paper presented an electric network interface (ENI), enabling a large network to be split at transmission line interconnections so that each of the sub-networks could be simulated on a separate core. The approach allows cores on a single computer to be used for parallel processing and also allows cores on computers connected by a local area network (LAN) to be harnessed.

Several simulation experiments were conducted using the ENI system and blocks or kernels of different sizes running on each of the cores. It was found that the effectiveness of the approach was limited to when the communication overhead was substantially smaller than the computation time on each processor. For the platform considered, a 78 bus kernel size was the breakeven point; only beyond which the speed-up due to parallel processing would warrant its use. For a 300-bus kernel size, the communication overhead became negligibly small compared to the computation time, and the speed-up factor was essentially equal to the number of cores, as is the case for full parallelism.

The 78 bus critical kernel size is based on modeling networks in which there are no power-electronic blocks requiring multiple switch operations or iterative computations such as those required for surge arrestors. The presence of such elements would increase the computing time for subsystems, thereby reducing the critical kernel size.

The ENI computing environment will result in significant savings in the case of sufficiently large networks, if they are partitioned properly into effective kernel sizes based on available t-lines in the system.

VII. References

[1] H. W. Dommel, "Digital Computer Solution of Electromagnetic Transients in Single and Multiphase Networks," *IEEE Transactions on Power Apparatus and Systems 88(4)*, pp. 388-399, 1969.

[2] RTDS Technologies Inc., Available at: http://www.rtds.com/ , February 2013.

[3] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid," *International Journal Supercomputer Applications, 15(3)*, 2001.

[4] I. Foster, C. Kesselman, J. Nick, and S. Tuecke, "The Physiology of the Grid," *Open Grid Service Infrastructure WG, Global Grid Forum*, June 22, 2002.

[5] I. Foster and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit," *International Journal Supercomputer Applications*, *11(2):115-128*, 1997.

[6] Blaise Barney, "Introduction to Parallel Computing. Available at: https://computing.llnl.gov/tutorials/parallel_comp/," March 2010.

[7] *EMTDC User's Guide*, 2003.

[8] Incredibuild Xoreax Grid Engine, Available at: http://www.incredibuild.com/ , February 2013.

[9] T. Athay, R. Podmore, and S. Virmani, "A Practical Method For The Direct Analysis Of Transient Stability," *IEEE Transactions on Power Apparatus and Systems*, Vol. PAS-98, No.2 March/April 1979.

[10] Rich Christie, "Power Systems Test Case Archive," Electrical Engineering, University of Washington, Available at: http://www.ee.washington.edu/research/pstca/ , February 2013.