A High-Performance Hybrid Computing Approach to Massive Contingency Analysis in the Power Grid

Ian Gorton, Zhenyu Huang, Yousu Chen, Benson Kalahar, Shuangshuang Jin, Daniel Chavarría-Miranda, Doug Baxter, John Feo

Pacific Northwest National Laboratory,
Richland WA 99352, USA
[*]ian.gorton@pnl.gov

Abstract

Operating the electrical power grid to prevent power black-outs is a complex task. An important aspect of this is contingency analysis, which involves understanding and mitigating potential failures in power grid elements such as transmission lines. When taking into account the potential for multiple simultaneous failures (known as the N-x contingency problem), contingency analysis becomes a massively computational task. In this paper we describe a novel hybrid computational approach to contingency analysis. This approach exploits the unique graph processing performance of the Cray XMT in conjunction with a conventional massively parallel compute cluster to identify likely simultaneous failures that could cause widespread cascading power failures that have massive economic and social impact on society. The approach has the potential to provide the first practical and scalable solution to the N-x contingency problem. When deployed in power grid operations, it will increase the grid operator's ability to deal effectively with outages and failures with power grid components while preserving stable and safe operation of the grid. The paper describes the architecture of our solution and presents preliminary performance results that validate the efficacy of our approach.

Introduction

Electric power transmission involves the delivery of electricity from power generators to consumers. A power transmission network connects power plants to multiple substations in a populated area, and the wiring from substations to customers provides the final step of electricity distribution. Electric power transmission allows widely geographically dispersed energy generators (such as hydroelectric, coal and nuclear power plants) to be connected to consumers.

A power transmission network is called a *grid*. Multiple redundant lines between nodes in the network are provided so that power can be routed using a variety of paths from any power plant to any load center. The exact route chosen can be based on the economics of the transmission path and the cost of power.

If a problem occurs on the electric power grid, such as a transmission line going out of service due to contact with vegetation, there are various issues that may arise. For example, if large amounts of electric power are being transferred from one geographic area to another, the loss the connecting line will have impacts on loads and voltages across the grid. Loads may be lost and voltages may drop or even collapse in various areas as a result, leading to power outages.

Power grid operators in North America refer to such unexpected outages as "contingencies" and manage the system in a way that ensures *any single contingency will not propagate into a cascading blackout*. This is known as the N-1 contingency standard issued by the North American Electric Reliability Corporation (NERC).

Past power grid blackouts like the Northeast Blackout in 2003[1] have involved 10-20 simultaneous contingencies happening across the grid. Therefore, if multiple contingencies occur simultaneously, the urgency to restore the grid to a normal condition is far greater. This clearly indicates the need for N-x contingency analysis, i.e. analysis of the potential simultaneous occurrence of multiple contingencies. N-x contingency analysis can prepare grid operators with mitigation procedures so as to avoid cascading failures. N-x contingency analysis is also, as we explain later, an extremely complex computational task.

In this paper we describe our computational approach for N-x contingency analysis. The approach exploits the multithreaded Cray XMT architecture to select likely contingencies from the power grid network. It transfers the identified contingencies to a conventional compute cluster to perform contingency analysis on selected cases based AC power flow. This analysis generates potentially terabytes of data which must be transferred back to the XMT to identify likely power grid vulnerabilities and utilize advanced visual analytical methods to present the results to operators for remedial actions.

We have previously demonstrated the potential of the Cray MTA-2 (the XMT's predecessor) for solving the static state estimation problem for the electric power grid [2]. Based on this experience, this project exploits the unique heterogeneous architecture of the Cray XMT to tackle the enormous computational challenge of advanced contingency analysis for the electric power grid [3,4,5].

A novel aspect of our approach is the hybrid architecture we employ. We draw on the strengths of the Cray XMT to perform the necessary graph analyses for contingency selection [7]. The subsequent contingency analysis involves the execution of thousands of independent tasks, and is hence most appropriately executed on a conventional cluster architecture. We connect the two computational platforms using our MeDICi integration framework [6], which provides high-performance transport of large amounts of data.

The paper briefly explains the complexity of N-x contingency analysis, and describes our hybrid architecture and the major aspects of the algorithms used on each platform. It then presents some initial end-to-end performance figures which demonstrate the viability of this approach.

Background: Contingency Planning for the Power Grid

In North America there are three interconnected power grids – the eastern interconnection, western interconnection, and the Texas grid. Although each grid is a single interconnected machine, it is divided into different administrative entities called Balancing Areas (or BAs) which own, operate, and/or manage their own pieces of the grid. For example, the western interconnection has 35 BAs, as shown in Figure 1

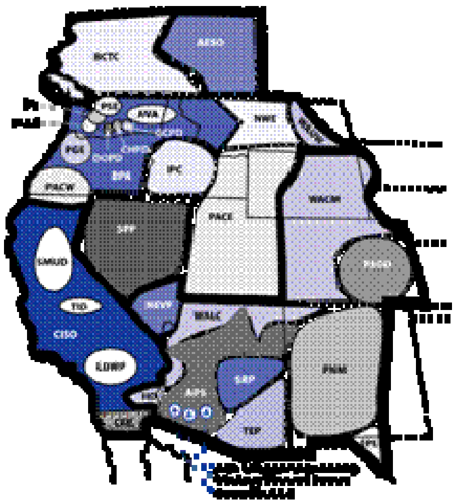


Figure 1 Major Balancing Areas in the Western power grid

When performing contingency analysis, each BA looks no further than its own boundaries. For areas within the interconnection where several BAs reside next to each other, the potential problem caused by simultaneous failures across multiple BAs can be missed. Individually, model results from each BA may show that each contingency does not cause a problem. However, if these contingencies occur simultaneously, there will likely be a very large system-wide impact. Unfortunately, the urgency to restore the system is not fully recognized with today's N-1 contingency analysis.

In the western power grid, there are approximately 20,000 elements that could fail. Checking each element takes about ½ cpu second, therefore, the entire N-1 contingency case set would take about 10^4 cpu seconds. In order to check all the combinations of x contingencies (x = 2,3,4,…), it would take 10^{4x} cpu seconds. Given that there are 35 different BAs in the western grid, one could consider there are often several simultaneous contingencies occurring – but in different BAs. Assuming x=10, checking all these N-10 contingency cases rigorously would require computational time approximately in the order of 10^{40} cpu seconds and is obviously infeasible.

Therefore, an important element in a practical solution is how to identify the N-x contingencies from a system-wide perspective. This will allow high performance computing to check a limited set of contingencies and identify detrimental consequences. For each contingency that exhibits these problematic results, automated algorithms must be created that will identify the remedial operator actions necessary to maintain grid stability should that contingency actually occur. The automation of this algorithm is necessary since this process will run continuously in an operational setting. The effectiveness of the evaluation will be limited by the efficiency and efficacy of the algorithms and the processing capacity of the computers used. As we explain in the next section, our solution uses a hybrid solution, exploiting the key strengths of the Cray XMT multithreaded system and a conventional supercomputing cluster.

Hybrid Computing for Contingency Analysis

Given the sheer number of contingency cases in the problem space and the real-time requirements for power grid operations, today's industry algorithms and tools are not able to handle comprehensive contingency analysis as described in the previous section. A comprehensive contingency analysis process includes the following three elements:

1) contingency selection,
2) parallel contingency analysis and

3) post-processing of contingency analysis

For contingency selection, we use a new method based on the concept of graph edge betweenness centrality. The power grid can be treated as a weighted undirected graph, and the edge betweenness centrality identifies the most "traveled" edges in the graph. The most traveled edges are considered the most important branches in a power grid, and their failures must be analyzed, while the least traveled edges are identified as low-impact branches, and their failures do not need to be analyzed because they are of little importance to power grid stability.

Detailed contingency analysis is performed for the selected contingency cases based on the ranking results from the contingency selection step. Each contingency case is a non-linear algebraic equation, called AC power flow. The Newton-Raphson method is used to solve the equation iteratively. As each case is independent, it can be run separately on multiple nodes in a cluster. Contingency analysis identifies *violations*, which are defined as electric quantities (bus voltages and power flows) that exceed their pre-specified limits. Violations are an indication of power grid vulnerability and should be identified and presented to power grid operators so the situation can be understood and remedial actions can be taken.

Post-processing is necessary to reduce the cognitive load on power grid operators. Today's industry tools simply presents the violation data in a tabular form without little post-processing. When the power grid is under stress, the number of violations can be very large, and operators have no way to understand the tabular presentation of so many violations in a real-time manner. In order to effectively interpret contingency analysis results and support situational awareness, post-processing of violation data is necessary. Violation data can be interpreted as impact of contingencies on various parts of the power grid. The more violations, the more severe the impact is. The impact can be visualized on a geographic map as colored contours indicating vulnerability levels with color intensity throughout the grid. Figure 2 shows an example graph of four contingency cases.

Our implementation of this 3 step process involves a Cray XMT and a conventional cluster-based supercomputer connected via a 1 GBit Ethernet network. To simplify the connectivity complexity, we utilize a MeDICi [6] pipeline.

The Cray XMT system is a scalable multithreaded high performance computing platform. The XMT supports a global shared memory accessible by all processors on the system and can scale to a total of 8,192 processors. With a hardware multithreading mechanism to tolerate memory access latencies frequently encountered in irregular, graph-processing algorithms, the XMT is a suitable platform for parallel processing of large-scale power grid graphs.

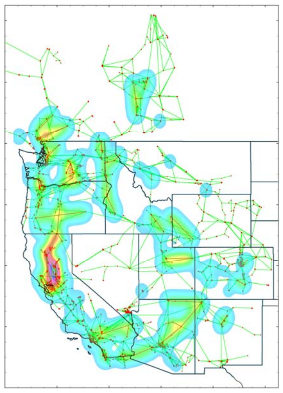


Figure 2 Power grid vulnerability assessment through visual analytics

Our implementation of this 3 step process involves a Cray XMT and a conventional cluster-based supercomputer connected via a 1 GBit Ethernet network. To simplify the connectivity complexity, we utilize a MeDICi [6] pipeline.

The Cray XMT system is a scalable multithreaded high performance computing platform. The XMT supports a global shared memory accessible by all processors on the system and can scale to a total of 8,192 processors. With a hardware multithreading mechanism to tolerate memory access latencies frequently encountered in irregular, graph-processing algorithms, the XMT is a suitable platform for parallel processing of large-scale power grid graphs.

Our current contingency analysis platform is a Linux cluster with 192 nodes with 8 cores each. The nodes are connected using a high-performance Quad Data Rate (QDR) InfiniBand network.

The MeDICi Integration Framework (MIF) is a middleware platform for integrating heterogeneous software codes (known as components) into a processing pipeline. MIF separates the integration logic from the processing logic used in components and the transports used for connectivity. Components are oblivious to the transports used to connect them together, and the topology that they are connected in. Transports can then be configured independently of the component logic, allowing a declarative configuration of a range of transports to meet the performance and quality of service requirements for an application.

Graph Analysis for Contingency Selection

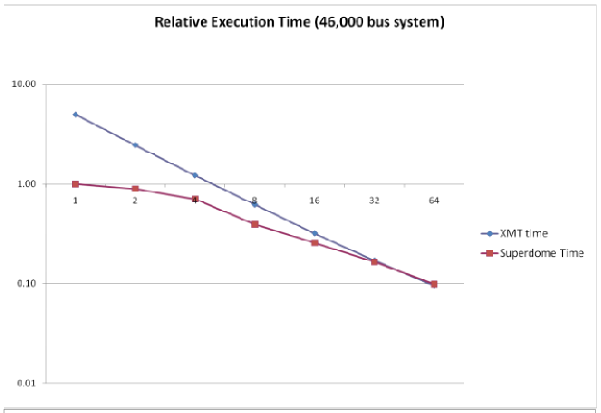
Vertex betweenness is a centrality measure of a vertex within a graph [8, 9]. Edge betweenness is a centrality measure of an edge within a graph. Edges that occur on many shortest paths between any vertex pair have higher betweenness than those that do not.

Ulrik Brandes' algorithm [10] is frequently used to calculate vertex betweenness for unweighted/weighted graphs. It requires *O(n+m)* space and *O(nm)* time for unweighted graphs, and $O(nm + n^2 \log n)$ time for weighted graphs, where *n* and *m* are the number of vertices and edges in the graph, respectively. In order to apply edge betweenness centrality to power grid graphs, we adapted Brandes' algorithm to calculate edge betweenness for weighted power grid graphs. In power grids it is possible that there are multiple shortest paths between two buses, therefore Dijkstra's algorithm [11] was also adapted to store multiple shortest paths. The pseudo code for this modified algorithm is given in [7].

We have implemented this algorithm on the Cray XMT using the parallelization capabilities of the XMT's multithreading C/C++ compiler. In the edge betweenness centrality computation, the modified Brandes' algorithm is called for each vertex in the power grid graph. In each iteration, all the shortest paths are identified between this given vertex and any other vertices in the graph, and then the algorithm accumulates the calculated edge betweenness values for particular edges. After all the vertices are scanned, edge betweenness of all edges is available. Each iteration is independent and can be executed in parallel since it analyzes independently-sourced shortest paths. We have used XMT-specific pragma (`#pragma mta parallel`) to guide the compiler in parallelizing other sections of the code. We also use the atomic update pragma (`#pragma mta update`) before updates to the shared edge betweenness variables so the compiler knows that this statement should be performed atomically.

We have also implemented a portable OpenMP version of the same code for comparison purposes on mainstream CPUs. We have used a 128-core (64 dual Itanium 2 1.5 GHz processors, 256 GB shared memory) HP Superdome system to compare its performance against the Cray XMT.

Figure 3 compares the performance of our implementations of power grid betweenness centrality on both the Cray XMT and the HP Superdome for two problems sizes: 46,000 buses (comparable to the Eastern US power grid) and 170,000 buses (comparable to a more detailed view of the Eastern US grid). The data presented on both figures is execution time normalized to the time on one processor on the HP Superdome (time on one HP Superdome processor is 1.0). Both figures use a logarithmic y-axis to improve the readability of the graph. As can be seen from both figures the per-processor performance of the XMT is lower than the Superdome, but its scalability is better leading to overall reduced execution time at scale.



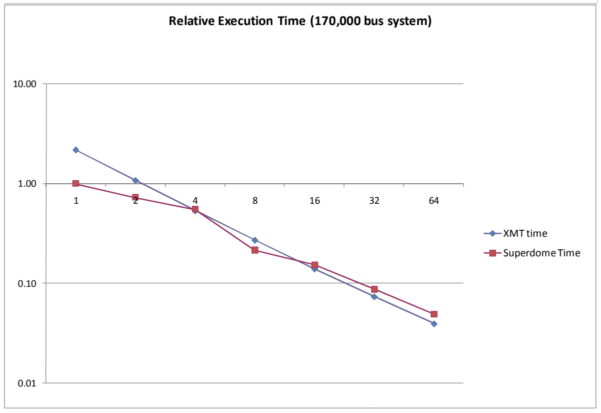
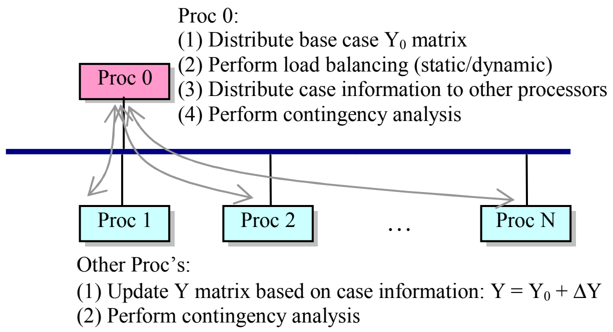


Figure 3 Comparison of XMT and Superdome performance for Contingency Selection for 2 different problem sizes

Parallel Contingency Analysis

Our framework for parallel contingency analysis is shown in Figure 4. Each contingency case is essentially a power flow run. To manage the contingency case allocation and load balancing, one processor is designated as the master process (Proc 0 in Figure 4), in addition to running contingency cases.



Proc 0:
(1) Distribute base case Y_0 matrix
(2) Perform load balancing (static/dynamic)
(3) Distribute case information to other processors
(4) Perform contingency analysis
Proc 0
Proc 1
Proc 2
…
Proc N
Other Proc's:
(1) Update Y matrix based on case information: $Y = Y_0 + \Delta Y$
(2) Perform contingency analysis

Figure 4 Framework for parallel contingency analysis

A straightforward mechanism for load balancing is to statically pre-allocate an equal number of cases to each processor, i.e. static load balancing. With static load balancing, the master processor only needs to allocate the cases once at the beginning of the analysis. Since the convergence performance is different for different cases, each contingency analysis run will require a different number of iterations and thus take a different time to finish. The extreme case would be non-converged cases which iterate until the maximum number of iterations is reached. These variations in execution time would result in unevenness, with the overall computational efficiency determined by the longest execution time of individual processors. Therefore, computational power is not fully utilized as many processors are idle while waiting for the last one to finish.

Therefore a dynamic load balancing scheme was used to allocate tasks to processors based on the availability of a processor. Contingency cases are dynamically allocated to free individual processors to evenly distribute load and minimize processor idle time. The scheme is based on a shared task counter updated by atomic fetch-and-add operations. The master processor (Proc 0) does not distribute all the cases at the beginning. Instead, it maintains a task counter. Whenever a processor finishes its assigned case, the processor requests more tasks from the master processor and the task counter is updated. This process is illustrated in Figure 5. With this approach, the number of cases executed by each processor will not be equal, but the computation time on each processor is optimally utilized.

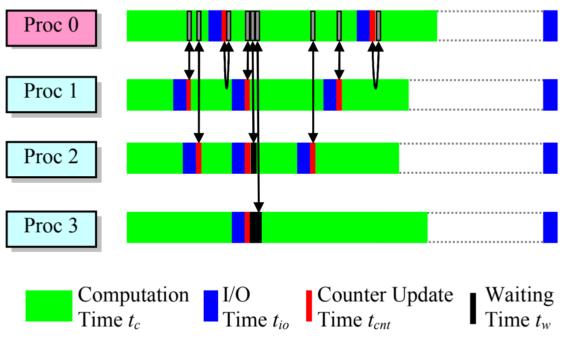


Figure 5 Task-counter-based dynamic computational load balancing scheme

This approach has been compared with a static allocation scheme for 512 "N-1" contingency cases on a 32 node cluster. The results are shown in Figure 6. The performance of dynamic load balancing, in comparison with its static counterpart, shows much greater linear scalability. The dynamic scheme achieves eight times more speedup with 32 processors, and the difference is expected to be greater with as the problem is executed on a much large cluster.

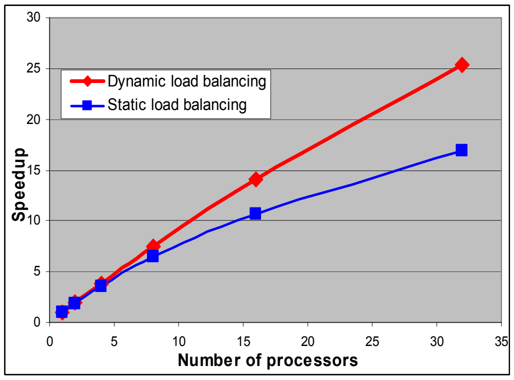


Figure 6 Performance comparison of static and dynamic computation load balancing schemes with 512 WECC contingency cases

As further validation, Figure 7 compares the processor execution time for the case with 32 processors. With dynamic load balancing, the execution time for all the processors is within a small variation of the average 23.4 seconds, while static load balancing has variations as large as 20 seconds or 86%. The dynamic load balancing scheme successfully improves speedups. It is also worth pointing out that the contingency analysis process of 512 WECC cases with full Newton-Raphson power flow solutions can be finished within about 25 seconds. It is a significant improvement compared to several minutes in current industry practice. More case studies with up to 300,000 contingency cases can be found in [12]. All the results show the advantages of dynamic load balancing scheme.

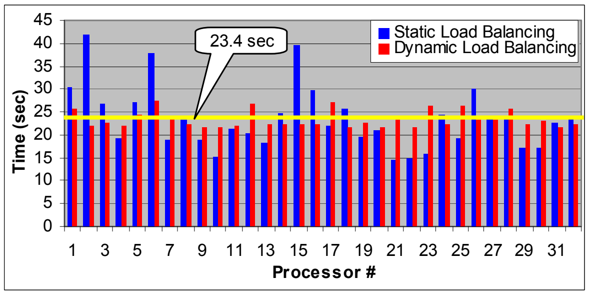


Figure 7 Evenness of execution time with different computational load balancing schemes

Integration with MeDICi

The MeDICi Integration Framework (MIF) is used to facilitate seamless, flexible communication between the XMT and the traditional compute cluster. The framework abstracts away the complexity of high performance networking code and allows the programmer to concentrate on higher level integration logic. General MIF applications take the form of a processing pipeline, comprising software components connected asynchronously or synchronously. Data flows between components which process the data in some manner and pass the results to the next component(s) in the pipeline. Individual connections between components can be declaratively configured to specify a range of protocols supported by MIF, allowing for simple tuning to match quality of service requirements without making changes to the integration code [6].

In our contingency analysis architecture, a MIF pipeline has been constructed to:

1) Accept the selected contingencies from the XMT and transfer the data to the compute cluster
2) Invoke the contingency simulation code and gather the results
3) Transfer the results back to the XMT and make available for the post-processing phase

For step (1), the data sizes are the order of 10-150KB, and hence we configure MIF to use the TCP transport to facilitate a low overhead connection with guaranteed delivery of data. A MIF component on an XMT Linux node accepts a list of selected contingencies from the analysis code on the XMT and sends it via TCP to the compute cluster. A MeDICi component running on the cluster communicates with the job scheduler to launch the contingency simulation. The next component in the pipeline then monitors for output and forwards completed contingency data to the XMT.

When running simulations for large contingency lists, LZO compression is used to decrease both disk and network overhead. LZO is a compression algorithm and associated library that achieves up to 75% compression on our data. It is exceedingly fast, so it does not introduce unacceptable levels of computational overhead.

As the compressed data is currently in the 100's of MB to GB range depending on the number of contingency cases, the MIF pipeline uses the high performance *bbcp* protocol, which provides multithreaded chunked transfer, with no encryption overhead. It provides a multithreaded chunking transfer, with no encryption overhead to transfer the data files. It is invoked by MeDICi's external command module, which makes it entirely transparent to the rest of the system, and simple to replace with a faster protocol. In our tests, the MIF pipeline driving this protocol delivers on average 30MB/sec over the shared laboratory gigabit network between the XMT and the compute cluster.

Performance Analysis

In order to validate our hybrid system's performance, we have integrated all of the software codes that are critical to contingency analysis to create an end-to-end contingency test platform. The test case includes all elements except the post-processing and visualization, which is not performance critical.

Our current hardware test-bed is:

- 64 node Cray XMT with 512Mb of shared memory
- 8 dual quad-core Intel Xeon "Clovertown" nodes running at 2.33 GHz (64 cores total), with nodes connected by a high-performance Quad Data Rate (QDR) InfiniBand network**.**
- 1 gigabit shared interconnect

Figure 8 shows the total computation time in seconds of test cases ranging from 200 to 17000 selected contingencies. Contingency analysis, including the transfer of data to/from the cluster using MeDICi, dominates these times, ranging from approximately 85% of the total time for 200 contingencies, to 98% of the total time for 17000 contingencies. However, our tests only utilized 64 cores, and hence the contingency analysis time for the larger contingency cases will be greatly reduced on larger clusters, as we have demonstrated in [12].

Data sizes transferred to the cluster after contingency selection are in the 10's of Kbytes ranges, and hence their transfer time is negligible. Compressed data sizes generated by the contingency analysis grow non-linearly and range from 44Mbytes for 200 contingencies to 2.5GBytes for 17000 contingencies. The transfer time becomes a significant fraction of the cluster-based contingency analysis time for larger number of cases. For example, for 200 cases, data transfer is less than 2 seconds, or approximately 3% of the contingency analysis. This rises to 5% for 500 cases, 7% for 1000 cases, and eventually 17% for 17000 cases. Again, the performance of the data transfer would be significantly improved on a dedicated high speed network between the two compute platforms. When fully optimized, *bbcp* is capable of transferring data at close to line speeds, which is 4 times faster than the performance we are experiencing on the shared laboratory network used in these tests. This would significantly reduce the overheads of data transfer as data sizes grow.

Contingency selection time on the XMT is almost constant in all these test cases (12.83 seconds for 200 cases and 13.16 seconds for 17000). Other variables in performance are introduced by sorting and transferring larger data sizes as the number of cases is increased.

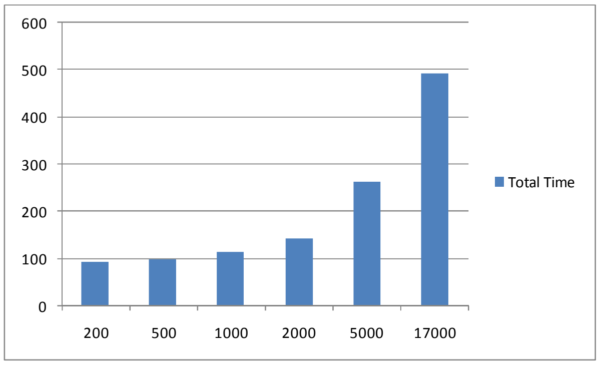


Figure 8 Total computation time for N-1 Contingency Analysis cases in seconds

Overall these initial results show that our hybrid architecture has the potential to perform contingency analysis in a time frame that is appropriate for power grid operations. These results provide a strong validation of the algorithms we are using and the hybrid computational approach.

Conclusions and Further Work

Providing a high-performance solution to the N-x contingency analysis problem is a key challenge for the future of power grid management. In this project, we are investigating how a hybrid solution can be built and scale to realistic problem sizes.

This paper describes our approach and describes performance results that validate the components of the hybrid architecture. We also present initial end-to-end performance results which clearly show the potential for our algorithms to scale and evolve into an operational scenario on next-generation hybrid architectures.

We are highly encouraged by these results and are currently working to scale up tests to a 128 node XMT and PNNL's 161 TFLOP supercomputer so that we can execute significantly larger N-x contingency analyses. We are already seeing the need for using significantly more cores for contingency analysis for N-2 cases, and increases in compressed data output sizes of up to five times. Hence, handling these larger computations will provide even deeper insights into the novel hybrid architecture we are using for this application.

8. References

1. U.S.-Canada Power System Outage Task Force, *Final Report on the August 14, 2003 Blackout in the United State and Canada: Causes and Recommendations*, April 2004. at https://reports.energy.gov/.
2. J. Nieplocha, D. Chavarría-Miranda, V. Tipparaju, Zhenyu Huang, A. Marquez. *Parallel WLS State Estimator on Shared Memory Computers*, in: Proceedings of *IPEC2007* – the 8th International Power Engineering Conference, Singapore, 3-6 December 2007.
3. J. Deuse, K. Karoui, A. Bihain, J. Dubois, *Comprehensive approach of power system contingency analysis*, 2003 IEEE Power Tech Conference Proceedings, Bologna, Volume 3, 23-26 June, 2003.
4. Q. Morante, N. Ranaldo, A. Vaccaro, E. Zimeo, *Pervasive Grid for Large-Scale Power Systems Contingency Analysis*, IEEE Transactions on Industrial Informatics, vol. 2, no. 3, August 2006
5. R.H. Chen, G. Jingde, O.P. Malik, W. Shi-Ying, N. Xiang, *Automatic contingency analysis and classification*, The Fourth International Conference on Power System Control and Management, 16-18 April, 1996.
6. I. Gorton, A. Wynne, J. Almquist, J. Chatterton: *The MeDICi Integration Framework: A Platform for High Performance Data Streaming Applications*, In Procs Working IFIP/IEEE Conference on Software Architecture, Feb 2008: 95-104, IEEE
7. Y. Chen, S Jin, D Chavarría-Miranda, Z Huang, *Application of Cray XMT for Power Grid Contingency Selection*, Proceedings of Cray User Group 2009, Atlanta, GA, May 4-7, 2009.
8. L.C. Freeman, *A set of measures of centrality based on betweenness*, Sociometry, 1977, 40:35-41.
9. J.M. Anthonisse, *The rush in a directed graph*, Technical Report BN 9/71, Stichting Mathematisch Centrum, Amsterdam, 1971.
10. U. Brandes, *A Faster Algorithm for Betweenness Centrality*, Journal of Mathematical Sociology, Vol. 25, 2001, pp. 163-177.
11. E. W. Dijkstra, *A note on Two Problems Inconnexion with Graphs*, Numerische Mathematik, 1 (1959), S. 269–271.
12. Z. Huang, Y. Chen, J. Nieplocha, *Massive Contingency Analysis with High Performance Computing*, Proc. IEEE PES General Meeting, Calgary, Canada, July 2009