WIPL-D Parallelization Effort

Christopher Card
Black River Systems Company,
Utica, New York 13502
card@brsc.com

Abstract: This paper presents the results of the final year of a development effort to provide a scalable, portable, parallel scene generation tool that provides the capability to rapidly generate scenes of radiating and scattering structures in realistically complex electromagnetic environments. The benefit of such a tool is that it will provide users with the capability to solve large problems that cannot be currently solved with existing sequential electromagnetic modeling tools. This tool supports a broad range of users including researchers, algorithm developers, analysts, and system developers. This paper will present the parallelization process and will show the final results of the project.

The project presented here is the parallelization of WIPL-D, an electromagnetic modeling tool, which picks up in time from where [1] left off. Through parallelization, the well known and commercially available tool became faster and now possesses increased capabilities. This paper will walk you through the parallelization process, providing the strategies used and the results received.

Keywords: WIPL-DP, WIPL-D, Electromagnetic Modeling, Parallel Processing, CHSSI

1. Background

The parallel electromagnetic modeling tool (WIPL-DP) that has been developed under this effort is leveraged from a proven commercially successful sequential electromagnetic modeling and scene generation tool called WIPL-D (Wires, Plates and Dielectrics) which is described in [1]. This WIPL-DP effort was funded under the Common High Performance Computing Software Support Initiative (CHSSI) as described in [1].

This was a three year development effort with three major milestone reviews. We have completed and successfully passed all of the evaluations. The three major milestone reviews were used to determine if the project was meeting the guidelines set forth by the CHSSI program. The first two years are discussed in detail in [1]. This paper is a continuation and picks up at the start of the third and final year. The results from all work completed for the final milestone Beta review will be presented here.

2. WIPL-DP Development – Final Year

The third and final year of the WIPL-DP development effort completed in 2004. The goal for the last year was to further increase the performance of WIPL-DP by decreasing run time and adding additional functionality. This was accomplished by parallelizing the impedance matrix generation and solution, and the far field calculations. This parallelization was in addition to the frequency domain parallelization completed in the previous year, described in [1]. The results were presented at the Beta Test milestone which occurred in June 2004. The three specific critical test parameters that had to be met

were as follows. The scaled speedup was to exceed 80% (25% as a minimum) of the single processor C version on 64 nodes (32 nodes as a minimum). The code was to be run on three High Performance Computing platforms (minimum of 2) producing valid results. The single processor version of the code was to produce an optimal accuracy of 2% (3% minimum) when compared to the commercial version of WIPL-D.

The parallelization of the impedance matrix generation provided increased functionality, as well as improved speedup. By distributing the matrix, a greater number of unknowns were able to be used, thus allowing for larger simulations to be solved. The impedance matrix is made up of complex values and has the dimensions of number of unknowns by number of unknowns. Each processor performing the generation holds a fraction of the complete matrix. The number of processors used was determined by dividing the number of unknowns for the project by the number of unknowns that can be held in a single processor's memory. The matrix is constructed by looping over the number of elements (number of wires and plates) and performing impedance calculations. There is an outer and inner loop that cycle over the number of elements. The outer loop iterations are independent of each other, except for accumulations at the end of the inner loop. Each processor entering the loop was assigned iterations to be executed in parallel. Results prior to the accumulation were saved off, and the actual accumulations were performed afterward. All of the calculations in the matrix construction are distributed over the number of processors performing the generation, providing processing speed benefits.

The solving of the impedance matrix was the major bottleneck of the program, consuming around 75% of the processing time. Through parallelization the solution took significantly less processing time. Solving the system of equations was originally done using LU Decomposition. In the commercial version matrix solution, the order of execution creates dependencies between each iteration of the main loop. In order to parallelize the solution, the section of code that performs the current matrix solution was replaced with a parallel implementation. The impedance matrix is placed in a form that is needed by a parallel implementation through a distribution equation and then the solution can be transformed back to the form required for WIPL-D. The parallel complex LU Decomposition function that was implemented is contained in the Scalapack libraries. It handles the decomposition by distributing blocks of data among a rectangular grid of processors in a 2-Dimensional block-cyclic method. This distribution of data provides for good load balancing and cuts down on the amount of broadcasting. This solver was chosen since Scalapack is highly portable and free, although it is not the most efficient parallel matrix solver out there. The code was implemented such that different parallel matrix solving functions may be substituted in to perform the matrix solution.

The parallelization of the far field calculations allowed for scaled speedup. This portion of the code loops over the phi and theta directions present for the problem. The outer loop of the far field calculations contains independent iterations. The total number of directions was divided among the available processors to provide for scaled speedup. There was output file printing contained in the loop which had to be pulled out and then merged at the end in order to preserve the WIPL-D file structure.

For the Beta Test, the High Performance Computers that were selected to meet the requirements set forth by CHSSI were Huinalu and Tempest at the Maui High Performance Computing Center and the Compaq SC/45 at the Aeronautical Systems Center Major Shared Resource Center (ASC MSRC). The Huinalu and Tempest machines are described in detail in [1]. The Compaq SC/45 machine contained 128

nodes, with 4 EV6.8 Alpha 1GHz processors with 1 GB of memory per node. A modified version of the test case used for the Alpha Test in [1] was used for the Beta Test. Changes were made to make the model more realistic, which can be seen in Figure 1.

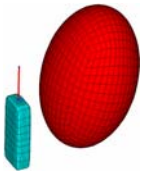

Figure 1. Cell Phone Alongside a Dielectric Ovoid

This test case contained 5,662 unknowns and was run over the frequency range of 0.9 to 2.4 GHz. The performance tests consisted of running the parallel code on 1 to 64 processors, with the number of frequencies for each case equal to the number of processors used. This allowed us to increase the problem size/time to demonstrate scaled speedup. The scaled speedup was calculated by taking the time for the code to run on a single processor and comparing it to the time it took on N processors (T(1)/T(N)). The accuracy of the code was determined by comparing the output files generated by the parallel code to the output files created by running the original WIPL-D PC version. Originally, three tests were planned for each case. The frequency and matrix parallelizations were to be tested separately and in conjunction with each other as a hybrid. For the formal Beta Review only the hybrid case was requested for determining if the code met the requirements. The speedup/timing results for the three target machines can be seen in Figures 2-4. They contain all three separate cases described above.

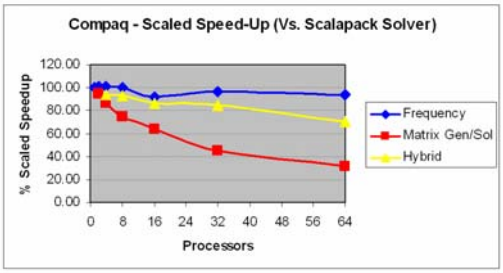

Figure 2. Scaled Speedup Results for Tests Run on the Compaq SC/45

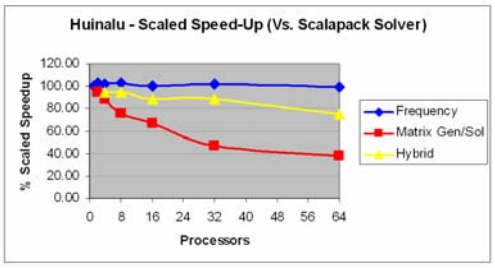

Figure 3. Scaled Speedup Results for Tests Run on the Huinalu Linux Cluster

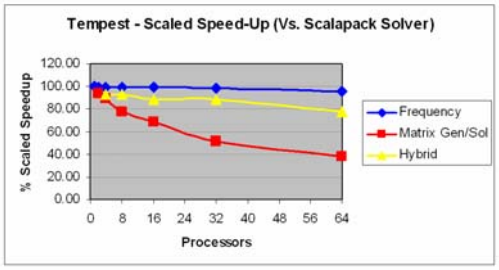

Figure 4. Scaled Speedup Results for Tests Run on the Tempest IBM SP3

The above figures show that the Hybrid case exceeded the optimum requirement of 80% scaled speedup for all but the 64 processor/frequency case, where it was well above the minimum requirement. The frequency parallelization alone performed exceptionally well for all cases. The matrix parallelization scaled speedup fell off as the number of processors/frequencies increased due to the matrix size we were using being relatively small. When the matrix was divided by more processors, the size of the local matrix

partitions held on each processor became so small that the communication overhead's impact was more significant. Another factor was that the parallel matrix solution routine we used was not highly efficient for this case. As mentioned above, the Scalapack libraries were chosen based on project requirements, not performance.

The accuracy results were determined by taking the percent error between the parallel version's output files and the commercial PC version. A summary of these results can be seen in Table 1. The worst case percent error is listed for each file. The overall worst case error of 0.082% fell well below the minimum requirement of 2%.

Accuracy Summary			
Output File	**Max Error (%)**	**Frequency Case**	**HPCC System**
ad1	0.0027746	16 (F, M, H)	Tempest, Huinalu
cu1	0.0048228	16 (F, M, H)	Tempest, Huinalu, Compaq
nf1	0.0221264	64 (F, M, H)	Compaq
ra1	0.0823788	4 (F, M, H)	Tempest, Huinalu, Compaq

Table 1. Percent Error for Parallel WIPL-D

The above listed results met the Critical Test Parameters set forth by the CHSSI Program for the Beta Review. The project was named a success by the CHSSI office.

3. Future Developments

With the CHSSI Project being completed, some follow on projects are being looked into. The applications team would have liked to see the solvable number of unknowns meet and exceed 100,000 for the tool to be highly beneficial. At this point the number of solvable unknowns is around 30,000. Late in the project a problem was identified with high numbers of unknowns that unfortunately was not able to be resolved before the end of the project due to time and money constraints. Increasing the number of unknowns was not a requirement under this project, but rather an outcome that was desired from it. Other desired improvements to the code include replacing the Scalapack libraries with a more efficient parallel solver and creating a graphical user interface WIPL-DP. Also, since the version of WIPL-D that we were provided with is several versions older compared to the latest, we would like to incorporate any changes that have been made.

4. Conclusion

Parallel WIPL-D provides users with an electromagnetic simulation modeling tool that can solve larger problems quickly and with increased capability. The code has passed all of the requirements set forth by the CHSSI Program, far exceeding the optimum requirements in all but one case. The ability to solve a larger number of unknowns and decreased processing time provide an improved tool, but in order for Tri-Service applications to highly benefit from this software the solvable number of unknowns must be raised even further. This continued work will need be performed under follow on projects since the CHSSI Program has ended.

References

[1] C. Card, "Alpha Test Analysis of WIPL-DP," 20th Annual Review of Progress in Applied Computational Electromagnetics, 2004 ACES Conference Proceedings, Syracuse, NY, April 19-23, 2004.