Performance Improvement by MPI Parallelization in 3D Device Simulation

Introduction

As the design technology for power devices, such as MOSFET, GTO, and IGBT has matured, the importance of large domain 3D TCAD simulation has increased rapidly. Distributed computing is one of the attractive solutions for such simulations, because the system’s performance and capability is not limited by the number of CPUs or the total amount of memory on a specific computer. This advantage of distributed computing is expected to be increasingly advantageous, as the size and mesh point count for these devices becomes ever larger.

Silvaco’s TCAD applications provide the user with the distributed computing feature which is supported in the solution of linear systems using the PAM solver [1, 2]. The PAM solver is a domain decomposition type solver that runs in parallel using MPI (Message Passing Interface). The user can set up the distributed computing feature with MPI parallelization easily, with the addition of a few simple settings on a Linux operating system [1].

In this article, we demonstrate good performance from the PAM solver with MPI parallelization using Victory Device on a blade server with a total of 120 threads. In addition, we verify the dependence of performance improvement by MPI parallelization, on the device size and number of mesh points.

Simulation Conditions

Table 1 shows the specification of the server used in this work. It consists of 6 nodes of a Dell PowerEdge C8220/C8220x blade server, with 20 threads of execution per node and 64 Gbytes of memory available. To achieve optimal performance in MPI parallelization, it is important to use a high speed network interconnect between the cluster nodes. InfiniBand FDR (Fourteen Data Rate, 14Gb/s data rate per lane) was utilized for the interconnection here.

On this cluster system, we carried out a DC $I_c-V_c$ simulation for the multiple cell array of a standard punch-through (PT) type 3D IGBT using the PAM solver in Victory Device. As the physical model in this simulation, we considered the field and concentration dependent mobility, SRH and Auger recombination, together with impact ionization.

The PAM solver is a domain decomposition type linear solver specially designed for very large sparse linear systems. Figure 1 shows a schematic diagram of the simulation method using the PAM solver with MPI parallelization. Each MPI process handles the solution of one part of the linear system and the MPI processes are run in parallel on each CPU thread. After each MPI process finishes with it's part of the linear system, the solution is sent back to the main MPI process, and the solution to the global linear system is re-formed and returned [2].

We carried out the same device simulation by using the various numbers of threads, up to 120 on the cluster system shown in Table 1. MPI parallelization was applied to every simulation with 2 threads and more. This means that the MPI processes by default, are spawned across the individual server nodes, even if the number of threads specified in the simulation is less than that on one server node. For example, if the user specifies parallelization with 3 threads, the first process is assigned to node #1, the next to the node #2, and the last to the node #3, respectively.

In order to verify the dependence of performance improvement by MPI parallelization on the device size and number of mesh points, we ran device simulations for two cell sizes of IGBT array. One was a 2×2 cell structure resulting in 134K mesh nodes and the other was a 6×5 cell structure with 977K mesh nodes.

<table>
<thead>
<tr>
<th>Node Type</th>
<th>Memory</th>
<th>CPU</th>
<th>Networking</th>
</tr>
</thead>
<tbody>
<tr>
<td>PowerEdge C8220</td>
<td>64GB, DDR3</td>
<td>ES-2680 v2 (25M Cache, 2.20GHz, 20threads)</td>
<td>2x 1Gb + 1x IB FDR</td>
</tr>
<tr>
<td>PowerEdge C8220</td>
<td>64GB, DDR3</td>
<td>ES-2680 v2 (25M Cache, 2.20GHz, 20threads)</td>
<td>2x 1Gb + 1x IB FDR</td>
</tr>
<tr>
<td>PowerEdge C8220</td>
<td>64GB, DDR3</td>
<td>ES-2680 v2 (25M Cache, 2.20GHz, 20threads)</td>
<td>2x 1Gb + 1x IB FDR</td>
</tr>
<tr>
<td>PowerEdge C8220</td>
<td>64GB, DDR3</td>
<td>ES-2680 v2 (25M Cache, 2.20GHz, 20threads)</td>
<td>2x 1Gb + 1x IB FDR</td>
</tr>
<tr>
<td>PowerEdge C8220x</td>
<td>64GB, DDR3</td>
<td>ES-2680 v2 (25M Cache, 2.20GHz, 20threads)</td>
<td>2x 1Gb + 1x IB FDR</td>
</tr>
</tbody>
</table>

Table 1. Specification of the server used in this work.
Results and Discussions

Figure 2 shows the dependence of simulation time on the number of threads parallelized by MPI. For relatively small numbers of threads the simulation time was drastically reduced as the number of threads increased, but the improvement saturated when using a much larger number of threads. The number of threads that gave the fastest simulation time was 32 for the 2×2 cells and 80 for the 6×5 cells, respectively. As expected, the number of parallel threads for the fastest calculation tends to increase as the device size and/or the number of mesh increases.

This feature can expand the device size for which the device simulation is completed in a practical time range. It will certainly be useful for the user who wants to simulate a large device especially in power device design.

Figure 3 shows the dependence of speed-up rate on the number of threads for the 2×2 cells structure. The speed-up rate is defined as the ratio of the single thread simulation time to the parallel thread simulation time. For the number of threads less than the optimal value, it was observed that the speed-up rate exceeded the ideal line on the assumption of the proportion to the number of threads. The speed-up rate reached the peak value of 50x at the optimal 32 threads.

The speed-up rate, however, saturated or decreased gradually for the number of threads more than this optimal value. It is not so easy to clarify the reason because this is complicatedly related to not only the software factor like solver performance or mesh quality, but also the hardware factor like CPU capability or network interconnecting speed. More comprehensive design of the cluster system can probably help the user achieve better MPI performance.
As a result, it can be confirmed from this result that Victory Device has the capability to improve the simulation time by orders of magnitude by using the distributed computing feature.

Figure 4 shows the overlay plot of the Ic-Vc characteristics simulated with the various numbers of threads in parallel. It can be seen from this figure that all the curves are identical with each other and therefore the accuracy of the simulation result is kept independently of the number of parallel threads. MPI parallelization by the PAM solver can provide the user with performance improvement without decreasing the accuracy.

Conclusion

We have demonstrated a 3D device simulation parallelized by MPI using Victory Device on a blade server with a total of 120 threads. As a result, the simulation time was drastically reduced without decreasing the accuracy as the number of threads increased and the speed-up rate reached the peak value of 50 at the optimal number of threads. Moreover, it has been confirmed that the number of parallel threads for the fastest calculation tends to increase as the device size and/or the number of mesh increases.

Silvaco’s TCAD having the distributed computing feature can expand the device size for which the device simulation is completed in a practical time range. We believe that it can help the user who wants to do 3D TCAD simulation with a large domain especially in power device design.

Acknowledgments

We gratefully acknowledge the support of Dell Solution Center Tokyo and HPC Solutions Inc. that provided the server and the equipment used in this work.

References
