Multiple GPUs Acceleration of Electromagnetics Simulations Using MPI and OpenACC

Saber Feki[1], Ahmed Al-Jarro[1,2], and Hakan Bağci[3]

[1]KAUST Supercomputing Laboratory, King Abdullah University of Science and Technology, KSA
[2]Photonics Group, Department of Electronic and Electrical Engineering, University College London, UK
[3]Division of Computer, Electrical and Mathematical Sciences and Engineering King Abdullah University of Science and Technology, KSA

Abstract

Graphics Processing Units (GPUs) are gradually becoming mainstream in supercomputing as their capabilities to significantly accelerate a large spectrum of scientific applications have been clearly identified and proven. Moreover, with the introduction of high level programming models such as OpenACC [1] and OpenMP 4.0 [2], these devices are becoming more accessible and practical to use by a larger scientific community. In this work, an explicit marching-on-in-time (MOT)-based time domain volume integral equation (TDVIE) solver is ported to multiple GPUs in a distributed environment. To this end, OpenACC directive-based parallel programming model is used to accelerate the MPI version of the explicit MOT-TDVIE solver on multiple GPUs, overcoming two often-faced challenges in GPU programming: developer productivity and code portability. This high-level approach requires minimal changes to the CPU-based code unlike CUDA and OpenCL, where significant modifications to the original code are required. The MPI parallelization allows the explicit TDVIE solver to simulate transient electromagnetic wave interactions on electrically large structures discretized using a large number of spatial elements. The MPI and OpenACC implementation achieved significant performance improvements on multiple GPUs with up to 11.2X speedup against the MPI and OpenMP CPU code using recent hardware technology.

Time Domain Volume Integral Equation Solver

An explicit predictor-corrector scheme is employed [3]:

Step 1 : **Predictor** – $\mathbf{f}$ is evaluated from up to previous time step

$$\mathbf{E}^{P}(\mathbf{r}_i,t_n)=\bar{\mathbf{M}}^{-1}\cdot\{Q[2\mathbf{E}(\mathbf{r}_i,t_{n-1})-\mathbf{E}(\mathbf{r}_i,t_{n-2})]+\mathbf{E}_0(\mathbf{r}_i,t_n)+\mathbf{f}(\mathbf{r}_i,t_n)\}$$

Step 2 : **Corrector** – $\mathbf{f}$ is evaluated from an averaged contribution from previous and current time steps for increased stability

$$\mathbf{E}^{C}(\mathbf{r}_i,t_n)=\bar{\mathbf{M}}^{-1}\cdot\{Q[2\mathbf{E}(\mathbf{r}_i,t_{n-1})-\mathbf{E}(\mathbf{r}_i,t_{n-2})]+\mathbf{E}_0(\mathbf{r}_i,t_n)+0.5[\mathbf{f}(\mathbf{r}_i,t_n)+\mathbf{f}(\mathbf{r}_i,t_{n-1})]\}$$

$$\mathbf{F}(\mathbf{r}_i,t_n)=\sum_{j=1}^{N_t}\Delta d^3 p_i(\mathbf{r})\frac{1}{R_{ij}}\mathbf{E}(\mathbf{r}_j,t_n-R_{ij}/c_b)$$

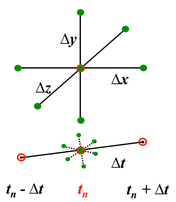
$$\mathbf{f}(\mathbf{r}_i,t_n)=\left(\nabla\nabla\%-\frac{\%}{c_b^2}\right)\mathbf{F}(\mathbf{r}_i,t_n)$$

$$\bar{\mathbf{M}}=[1+Q]\bar{\mathbf{I}}-\bar{\mathbf{D}}(\mathbf{r}_i)$$
$$Q=S(\mathbf{r}_i)/c_b^2\Delta t_n^2$$

$$\bar{\mathbf{D}}(\mathbf{r}_i)=p_i(\mathbf{r})\int_{V_i}\nabla'\nabla'\frac{dv'}{|\mathbf{r}_i-\mathbf{r}'|},\quad S(\mathbf{r}_i)=p_i(\mathbf{r})\int_{V_i}\frac{dv'}{|\mathbf{r}_i-\mathbf{r}'|}$$

Computational Cost: $O\left(N_e^2 N_t\right)$ *GPU Acceleration*

- Cubic discretization elements are used.
- The weak singularity is treated analytically.
- Linear interpolation on spheres of constant retarded time is applied to compute field values required for integration.
- The partial **spatial** and **temporal** derivatives are discretised using a combination of second order central and backward differencing schemes.



- ✓ **Explicit** → **No matrix inversion**
- ✓ **Nodal spatial discretization** → **Simple to implement**
- ✓ **Does not store interaction matrices** → **Computed on the fly** → **Memory efficient**

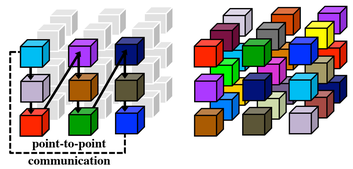
MPI and OpenACC Implementation of TDVIE Solver

- Unstructured mesh partitioning, invoked by METIS [4], is applied to ensure the even distribution of all computations and memory requirements of the explicit MOT-TDVIE solver.
- Efficient communication maps for the finite difference (FD) and correction-reevaluation operations are derived before time marching.
- The resulting memory efficient MPI code that implements the effective rotating tiles (**point-to-point MPI communication**) scheme [5] is annotated with OpenACC compiler directives.
- Compute intensive computations are offloaded to the GPUs - one GPU per MPI rank - using the OpenACC runtime library routine to set the device number *acc_set_device_num*.
- Compute intensive kernels are offloaded to GPUs with the *pragma acc kernels* directive
- Independent nested loops are annotated with the *#pragma acc loop independent* directive and further tuned with the *gang* and *vector* clauses
- Data transfers between host and GPUs are minimized with the OpenACC data directive.

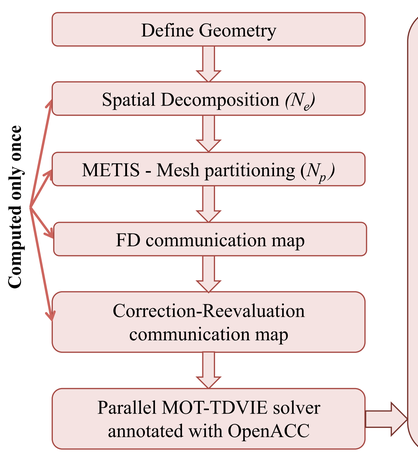
Unknowns N_e are equally distributed

$$O\left(\left(N_e N_g\right)/N_p\right)$$

MPI parallelization with OpenACC acceleration



Rotating Tiles MPI Communication Scheme with OpenACC Acceleration




```
#pragma acc data
for ( l = 0 ;  l < nt; ++l) {  // time loop
#pragma acc kernels
#pragma acc loop independent gang
  for (int i = 0; i < n; ++i){
#pragma acc loop independent gang,vector
    for (int j = 0; j < n; ++j){
#pragma acc loop independent gang,vector
      for (int k = 0; k < n; ++k){
        B[i][j][k] = B[i][j][k] + ....
  } } }
// MPI Communications implementing rotation
// tiles using point to point communication
#pragma acc kernels
#pragma acc loop independent gang
  for (int i = 0; i < n; ++i){
#pragma acc loop independent gang,vector
    for (int j = 0; j < n; ++j){
#pragma acc loop independent gang,vector
      for (int k = 0; k < n; ++k){
        B[i][j][k] = B[i][j][k] + ....
  } } }
} // end time loop
```

Performance Results

- The test bed used for performance evaluation consists in a dual socket CPU system hosting four NVIDIA Kepler K20c GPUs. Each socket is an eight-core Sandy Bridge Intel(R) Xeon(R) CPU E5-2650, running at a clock speed of 2.00GHz.
- Figure 1 shows a significant performance speedup ranging from **7.4X** to **11.2X** in comparison of the MPI and OpenMP implementation to multiple GPUs executions using a similar programming model: MPI and OpenACC. It is also observed that as N_e increases, higher speedup is achieved as the GPU is supplied with larger computational loads; therefore, taking better advantage of its computational capacity.
- The parallel efficiency of the MPI and OpenACC implementation of the TDVIE solver executed on two and eight GPUs ranges from **82%** to **94%** as shown in figure 2.
- Another advantage of using NVIDIA GPUs is the energy consumption reduction, which was recoded to reach up to **2.4X**

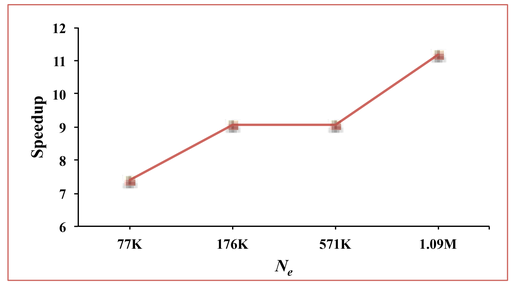


Fig. 1: Performance speedup of MPI and OpenACC on four K20c GPUs against MPI and OpenMP on 16 cores SandyBridge

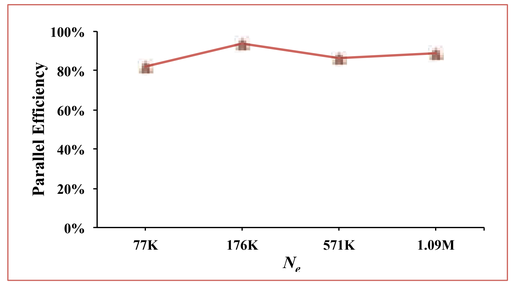


Fig. 2: Parallel efficiency of the MPI and OpenACC implementation scaling from two to eight GPUs

Conclusion

The experience of porting time-domain volume integral equation solver using MPI and OpenACC to multiple GPUs was effective; resulting in faster simulations by an order of magnitude of up to 11.2X when compared to the MPI and OpenMP version of the code on 16 CPU cores. A parallel efficiency of up to 94% is achieved while evaluating the multi-GPUs code scalability from two to eight GPUs. The scalability study will be performed at larger scale when computing resources become available to us. The OpenACC API has the advantage of transparently porting the MPI code to multiple GPUs environment, therefore increasing the developer productivity while keeping the legacy of the original CPU code. Furthermore, this parallelization allows the explicit TDVIE solver to efficiently simulate transient electromagnetic wave interactions on electrically large structures discretized using a large number of spatial elements, up to 1M spatial elements on four K20c GPUs. Additionally, an energy saving of up to 2.4X is observed when using NVIDIA GPUs. For all of the above, the GPUs are identified as the preferred computing platform in our overall performance analyses of the explicit MOT-TDVIE solver.

Future Work

- Further tuning of the ported application for better usage of the GPU capability and adaptation to different multi-GPUs environments.
- Performance evaluation on the latest NVIDIA GPUs and beyond.
- Scalability study on large scale GPU accelerated supercomputers, including Titan.

Acknowledgements

- Michael Young (KAUST IT)
- **NVIDIA for the hardware donations**

References

[1] OpenACC standard *www.**openacc-standard**.org/*

[2] OpenMP 4.0 specification *www.openmp.org/mp-documents/OpenMP4.0.0.pdf*

[3] A. Al-Jarro, M. A. Salem, H. Bağcı, T. M. Benson, P. Sewell and A. Vukovic, "Explicit solution of the time domain volume integral equation using a predictor-corrector scheme", IEEE Trans. Antennas and Propagat., vol. 60, no. 11, 2012.

[4] G. Karypis, and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 359-392, 1998.

[5] A. Al-Jarro, M. Cheeseman, and H. Bağcı, "An unstructured mesh partitioning and a memory efficient MPI or hybrid MPI/OpenMP parallelization scheme for the explicit marching-on-in-time solution of the time domain volume integral equation," submitted, Journal of Computational Physics, 2014.