*Progress In Electromagnetics Research M, Vol. 67, 169–176, 2018*

# Docker-Enabled Scalable Parallel MLFMA System for RCS Evaluation

**Jian Zhou, Shaowei Bie*, Ling Miao, Yuhao Zhang, and Jianjun Jiang**

**Abstract**—Research on RCS evaluation for electrically large objects has been a hot topic for decades. Although multilevel fast multipole algorithm (MLFMA) has been the most popular method in scattering computation, due to the limitation of both CPU speed and memory size in a single computer, realistic large targets require discretization with millions of unknowns still cannot be solved by sequential implementations of MLFMA. In this paper, we introduce a Docker-enabled parallel MLFMA computing system based on MPI, which is proved to be friendly for deployment and economical for scalability, to solve electrically large scattering problems. In addition, the capability of the proposed system has been demonstrated by several canonical examples.

## 1. INTRODUCTION

The solutions for electromagnetic scattering problems have been studied intensively in many scientific researches, such as antenna and radar applications [1–3]. However, solving scattering problems of electrically large objects, especially the RCS evaluation problems, is usually a time-consuming task. Various computational electromagnetic techniques have been proposed to reduce the computational complexity. The technology of electromagnetic modeling, NURBS, is widely used to represent complex bodies for RCS calculation, and it requires very little memory and computing time but performs difficultly in modeling [4]. High frequency asymptotic methods like GO and PO are widely used to efficiently solve electrically large complex scattering problems at high frequency range, but they are not wideband solutions [5, 6]. Arguably, the use of boundary integral equations with iterative solvers is one of the most powerful and popular methods to solve electrically large scattering problems [7, 8]. Obviously, MLFMA is the most successful one. A high performance parallel MLFMA has been introduced by Chew and his group to successfully solve RCS problems up to ten million unknowns using supercomputer in CEMS [9–11]. And billions of unknowns have been solved using the parallel MLFMA and a Tier 1 supercomputer by Fostier et al. [12]. The traditional parallel MLFMA solutions often rely on supercomputers, which are very large-scaled and well designed. Although their computational ability is very powerful, these supercomputers can be very expensive, and it is not affordable for us. With the steady development of the capability of the commodity computer, it becomes available to set up a parallel cluster which can offer a large scale memory and computational ability. However, there is still an obstacle, since building environment for every node can really require a significant investment of time and effort, also require users to have additional technical knowledge. Nguyen and Bein have used modern Docker technology to tackle this challenge, which brings great benefits to the development, building and deployment [13]. In this paper, we propose a scalable parallel MLFMA computing system supported by Docker on relatively inexpensive computing platforms to solve RCS problems of a couple of representative objects, namely, NASA almond and electrically large conducting spheres.

*Received 19 February 2018, Accepted 9 April 2018, Scheduled 17 April 2018*

* Corresponding author: Shaowei Bie (bieshaowei@hust.edu.cn).

The authors are with the School of Optical and Electronic Information, Huazhong University of Science and Technology, Wuhan 430074, China.

The rest content of the paper is organized as follows. In Section 2, we present the architecture of the parallel MLFMA computing cluster supported by Docker containers. In Section 3, we summarize the pseudo code of the parallel MLFMA implementations and relevant parallel strategy. Finally, numerical accuracy analysis and parallel performance analysis are presented in Section 4 and Section 5, respectively, followed by our concluding remarks in Section 6.

## 2. ARCHITECTURE OF THE PARALLEL MLFMA COMPUTING CLUSTER

A graphical representation of the architecture of the proposed parallel MLFMA computing system is presented in Fig. 1. The overall schematic diagram can be divided into two parts, the parallel MLFMA cluster and the Docker development, respectively. First of all, let's take a look at the structure of the parallel MLFMA cluster.

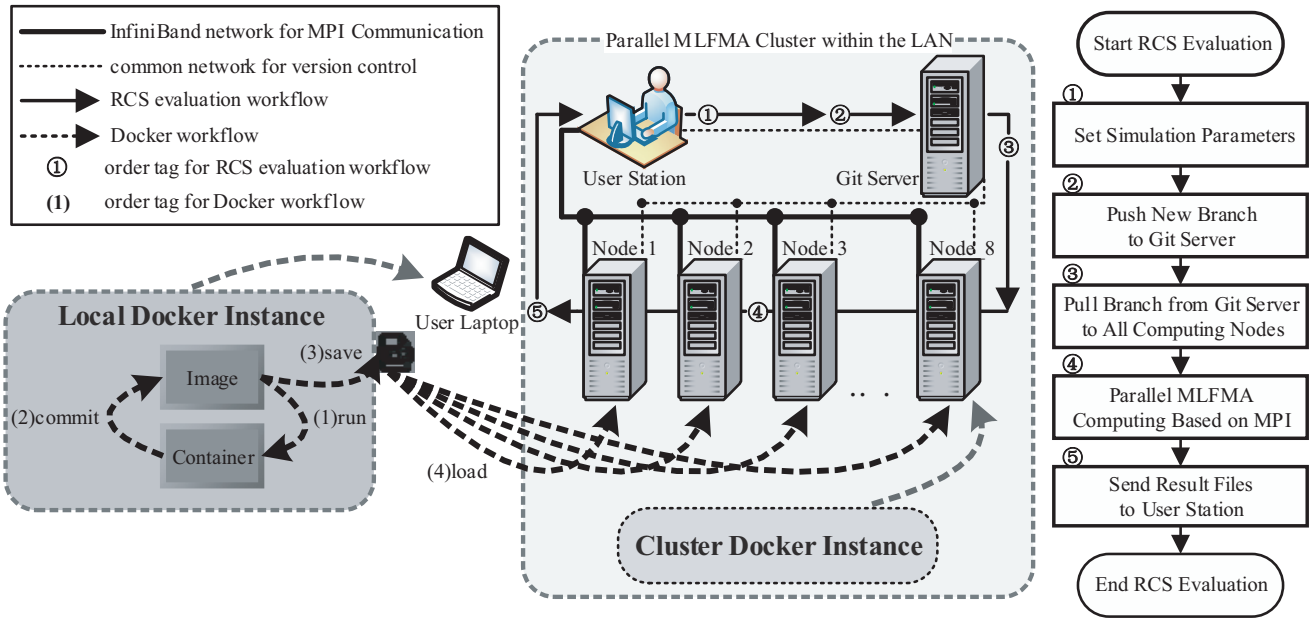



**Figure 1.** Architecture scheme of parallel MFLMA computing system, we use Docker technology for development, building and deployment.

The Beowulf architecture has been adopted in our research, since the Beowulf research projects have proved them to be able to provide high performance single user workstation at exceptional cost [14]. All simulations are performed on a so-called hybrid memory (shared memory for intra-communication, distributed memory for inter-communication) cluster consisting of 8 commodity-grade computers each containing one 4-core Intel i7-7700K processor (64 CPU cores in total), using 32 GB of RAM (or 4 GB per core). All the computing nodes are connected using an InfiniBand network. Besides the computing nodes, there is a master node (user station) in charge of task scheduling, and an auxiliary node (Git server) in charge of version control. It is worth watching what Git server has done for us, since it will be challenging for users to faultlessly and effectively synchronize source code to all computing nodes without a set of version control tools. In our design's originality, once the code is modified and committed as new branch to Git server (or called code repository), the modifications will be automatically pushed to all computing nodes.

The parallel simulation runtime environment is supported by Docker containers (a container is an isolated environment where one or more processes can run). A container is instantiated from the Docker image. An image contains all software dependencies needed to run an application, and it even includes a Linux distribution [15]. The parallel MLFMA is implemented in Python/C++ and some indispensable third party libraries, such as Python scientific computing toolkit and Blitz++,

and the inter-process communication is handled using the OpenMPI. It is quite challenging and time-consuming if we deploy them into cluster by hand. Docker technology provides a cost-effective solution to eliminate the procedures to reconfigure and rebuild applications for homogenous nodes, meanwhile preserve comparable system performance. As shown in Fig. 1, we build the source code and dependency packages once in the local Docker instance on the laptop, then the image encapsulated with runtime environment can be deployed on any compatible Linux machine. We distribute the Docker image to all computing nodes by Kubernates service, and what the computing nodes only need to do is to instantiate a Docker container.

One obstacle for the adoption of Docker containers in MPI-based cluster is that Docker does not support inter-node communication between different physical machines inherently. When the Docker service is started, a virtual bridge called "docker0" will be created, and all container instantiated on that physical machine can only communicate with local host through the docker0 which means that the container on machine A cannot communicate with the container on machine B, as depicted in Fig. 2(a). As depicted in Fig. 2(b), we propose an approach using Linux network tools to establish inter-node communication: (1) set the docker0 down, and set a virtual bridge called "br0" up; (2) delete original eth0 route (192.168.1.91/24 in our case), and add a new br0 route (192.168.1.191/24 in our case); (3) add "$-$b = br0" parameter option to DOCKER_OPTS (in Unix file "/etc/default/docker"); (4) start a new Docker container service, and give it a virtual IP address using the pipework tool (192.168.1.11/24 in our case). Finally, the SSH service (MPI communicating protocol) is currently available for inter-node communication.

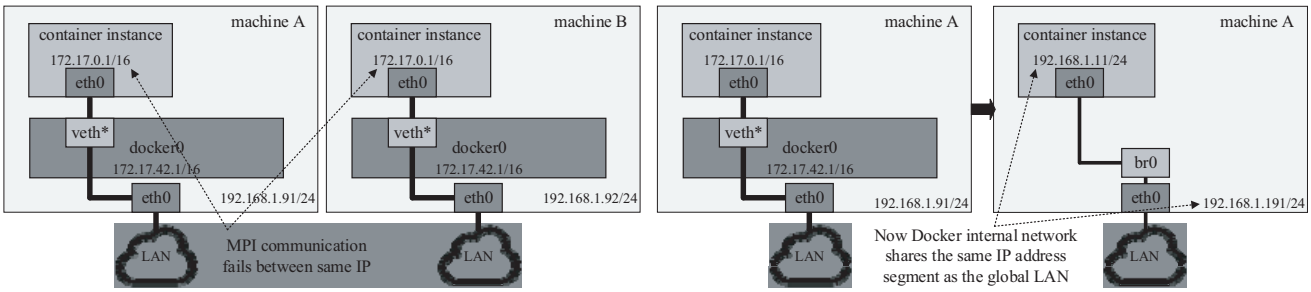



**Figure 2.** The transformation of Docker's network topology for inter-communication between multiple nodes.

## 3. PARALLEL IMPLEMENTS FOR MLFMA

The details of the parallel MLFMA code will not be discussed here, but we illustrate it in pseudo code as a whole picture in Table 1. Briefly, the parallel MLFMA implementations can be divided into four subroutines: (1) pre-processing; (2) constructing matrix equation; (3) solving matrix equation; (4) post-processing. Part (1) and part (4) are implemented in sequential code, while part (2) and part (3) are implemented in parallel code.

It is pivotal to program the subroutines of constructing matrix equation and solve matrix equation in the implementations of parallel MLFMA. The impedance matrix $\mathbf{Z}$ in the MLFMA is classified into a near-field interaction matrix and a far-field interaction matrix. Considering the near-field interaction matrix, rows can be assigned to processors in such a way that all processors have approximately equal numbers of near-field interactions due to their inherent sparsity. Before we detail how to set up the far-field interaction matrix, the parallel approach for far-field interaction should be declared first. To achieve good parallel efficiency, a transition level is introduced. The nonempty boxes are equally distributed among processes at levels lower than this transition level, and the far-field patterns are equally distributed among processes at levels higher than this transition level [10]. We choose level $\lceil L/2 \rceil$ ($L$ represents the number of total levels of the oct-tree) as the transition level. To evaluate the far-field interaction in the MLFMA, the shift matrix and interpolation matrix are needed in the aggregation stage to set up radiation patterns; the shift matrix and anterpolation matrix are needed in

**Table 1.** Pseudo code for parallel MLFMA.

```
Pre-processing {
  Using GMSH to generate meshing file;
  Reading coordinates of the center of every edge for MLFMA tree construction;
  Constructing the distributed oct-tree using 64-bit Morton keys; }
Setting up matrix equation {
  Setting up near-field interaction matrix;
  Setting up far-field interaction matrix {
    Setting up aggregation, translation and disaggregation matrix;
    Setting up shifting and interpolation/anterpolation matrix; }}
Solving matrix equation {
  Calculating matrix vector multiplication {
    Calculating near-field interaction;
    Calculating far-field interaction {
      Performing aggregation;
      Performing translation;
      Performing disaggregation;}}}
Post-processing {
  Calculating RCS; }
```

the disaggregation stage to set up receiving patterns; the translation matrix is needed in the translation to translate the radiation patterns into receiving patterns.

Since the translation matrix used to be the main bottleneck of the memory requirement, it is vital to reduce its store requirement. We adopt a method that distributes the translation matrix among processes in higher levels and use a compressed representation for the translation matrix in lower levels [16].

The calculation of the matrix equation in the MLFMA is divided into two parts, the calculation of the near-field interaction and the calculation of the far-field interaction. The parallel calculation of

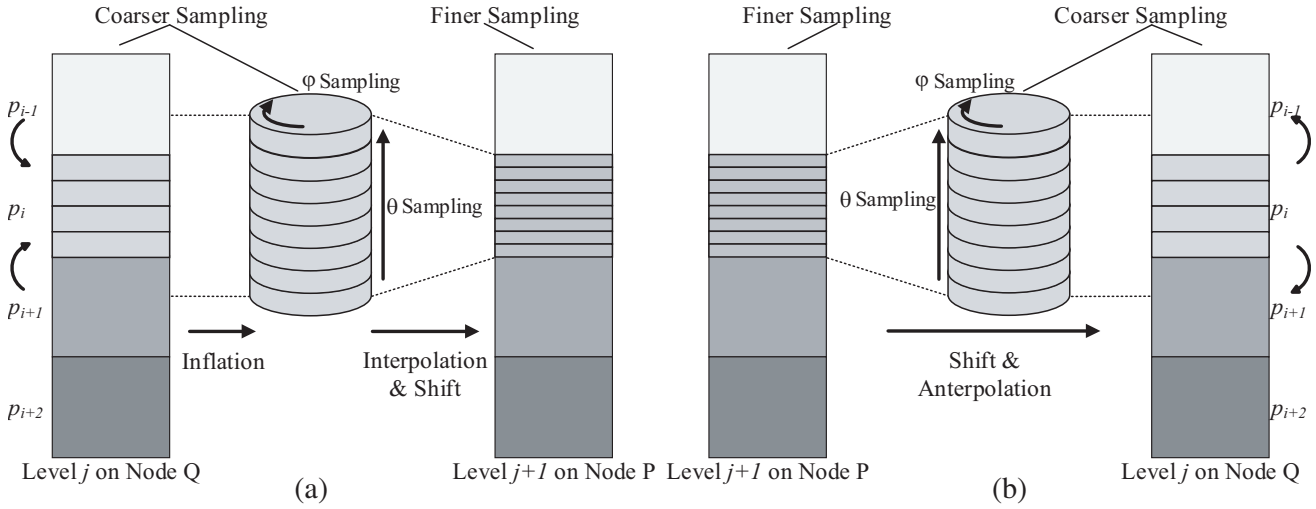



**Figure 3.** Parallel implementations during MLFMA stages, $p_i$ represents process $i$, (a) interpolation during aggregation stage; (b) anterpolation during disaggregation stage.

the near-field interaction is straightforward since the workload of the near-field interaction is equally distributed among processes. To perform the far-field interaction, both the radiation patterns and receiving patterns are needed to be stored for each nonempty box. As depicted in Fig. 3, we implement far-filed sampling not only along $\theta$ direction but also along $\varphi$ direction for effective parallelism. One-to-one communication is usually needed both for interpolation phase (in Fig. 3(a)) to set up radiation patterns and anterpolation phase (in Fig. 3(b)) to set up receiving patterns.

## 4. NUMERICAL ACCURACY ANALYSIS

The parallel MLFMA code is verified by comparing the results with those calculated by FEKO for conducting NASA Almond and by Mie scattering theory for conducting sphere. Flat triangular patches and RWG basis functions are used to discretize the target objects and a Galerkin scheme to discretize the surface integral formulations. The following numerical computation cases are set according to Table 2 for parallel MLFMA simulation settings.

**Table 2.** Parameters setting.

| Integral Equation | Combination Factor | Iterative Solver | Solver Tolerance |
|---|---|---|---|
| CFIE | 0.5 | Parallel BiCGSTab | 1e-3 |

### 4.1. Numerical Results of 9.936 Inches NASA Almond

In Fig. 4, monostatic RCS values of the NASA almond whose length is 9.936 inches for both HH and VV polarizations are plotted in dBsm at 9.92 GHz as a function of the azimuthal angle $\varphi$. The almond is originally modeled by GMSH using flat triangular patches whose mesh size $l_c = 0.1\lambda$, leading to a 19650 unknown problem. The metallic NASA almond is shown in Fig. 5(a), it is on the $x$-$y$ plane. Zero

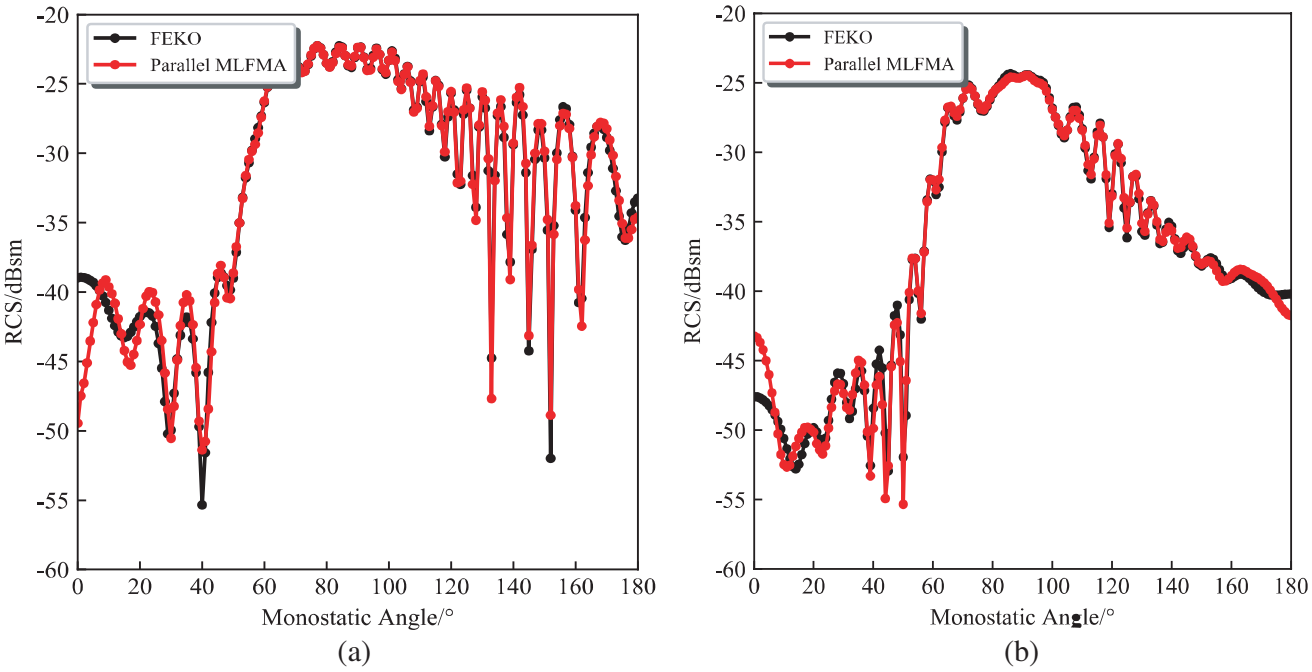
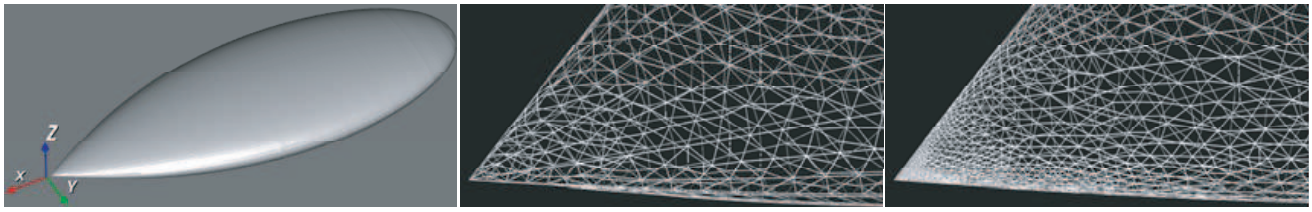



**Figure 4.** Monostatic RCS of the 9.936 inches NASA almond (planar triangular meshing model, discretized with 19650 unknowns) at 9.92 GHz as a function of $\varphi$ in the horizontal plane, (a) HH polarization; (b) VV polarization.




(a) (b) (c)

**Figure 5.** Meshing distribution around the tip of NASA almond, (a) 3D model of NASA almond; (b) GMSH meshing; (c) FEKO meshing.

degree ($\varphi = 0$) corresponds to normal incidence to the tip, and we measure it with its broad side flat. As depicted in Fig. 4, the RCS calculated using the parallel MLFMA agrees well with that using FEKO for both HH and VV polarizations except for the angle range approximately between 0° and 20°.

Since the discretization of surface around the tip region in FEKO (as shown in Fig. 5(c)) is quite different from that in GMSH (as shown in Fig. 5(b)), actually, meshing distribution in FEKO around the tip is denser than GMSH. We consider it as a reasonable explanation for the numerical difference between 0° and 20°.

### 4.2. Numerical Results of Electrically Large Conducting Sphere

To verify the accuracy for electrically large problems, the solution for a conducting sphere of radius $a = 50\lambda$ is considered. The sphere is also modeled by GMSH using flat triangular patches whose mesh size $l_c = 0.1\lambda$, leading to 12466833 unknowns. The sphere is illuminated by a plane wave propagating in the $-x$ direction with the electric field polarized in the $y$ direction. Fig. 6 presents the bistatic RCS on the $x$-$y$ plane as a function of the bistatic angle $\varphi$ from 0° to 180°, where 0° and 180° correspond to the back-scattering and forward-scattering directions, respectively. As depicted in Fig. 6, the numerical results are in good agreement with the analytical ones obtained by the Mie theory solution.

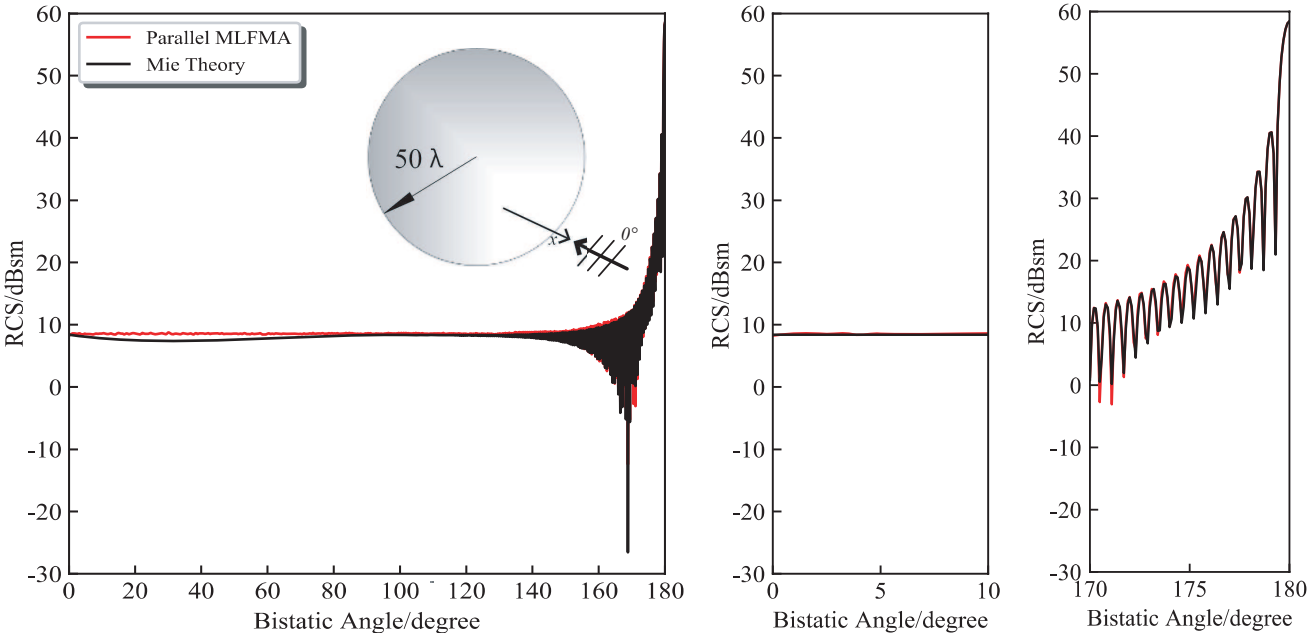



**Figure 6.** Bistatic RCS of a conducting sphere with a radius of 50λ (discretized with 12466833 unknowns). 0° and 180° correspond to the back-scattering and forward-scattering direction, respectively.

## 5. PARALLEL PERFORMANCE ANALYSIS

The accuracy of the computed results have been compared with the dataset offered by FEKO and with analytical results. In this section, we present numerical results demonstrating the parallel performance of the designed algorithm. A set of results are presented from a scattering problem involving a conducting sphere of radius $a = 40\lambda$ discretized with 8018784 unknowns. Let $T_1$ be the time for parallel MLFMA solution on a single processes, and let $T_p$ be the time on $p$ processes, then, the parallel speedup is defined as $T_1/T_p$. We consider the CPU times as a function of the number of processes in Fig. 7 and also plot the speedup as a function of the number of processes in the same figure. As depicted in Fig. 7, the complete MLFMA solution performs best at aspecific number of parallel processes, according to our coarse-grained settings, and 48 processes outperform the others. However, further increase in the processes deteriorates the performance. The reason for this phenomenon is that when we increase the number of processes, the communication data are possibly also increased, resulting in more time for communication, and total solution time cost may not decrease. Communication cost has been the bottleneck of parallel MLFMA. Furthermore, memory requirement for the translation matrix has been the bottleneck of communication as it is duplicated in every process [10]. We recommend to apply more well-designed parallel strategies to improve the process utilization [17].

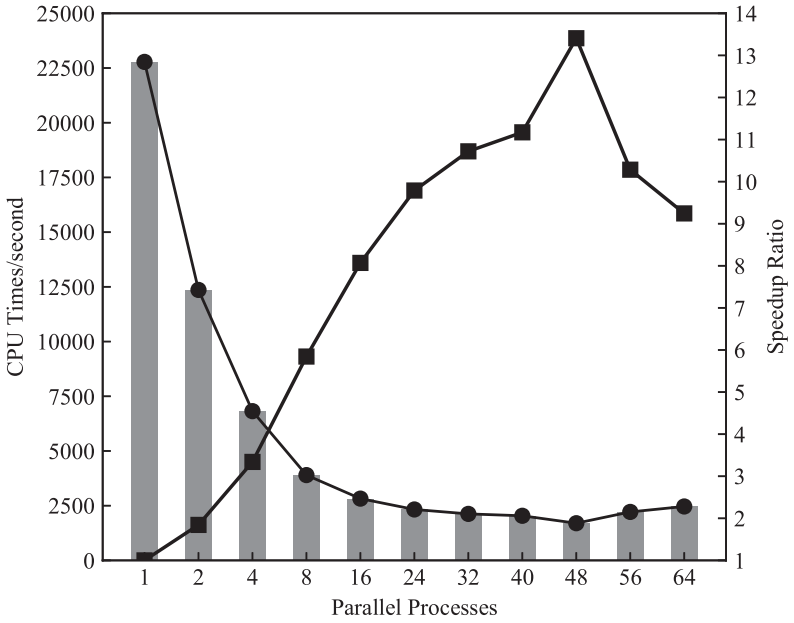



**Figure 7.** CPU times with respect to number of processes for bistatic RCS computing of a conducting sphere with a radius of $40\lambda$ (discretized with 8018784 unknowns). $T_1$ is the used time for parallel MLFMA solution on a single process, and $T_p$ is the used time on $p$ processes.

## 6. CONCLUSION

The paper presents a brief summary of a scalable parallel MLFMA computing system and the code that we have developed for RCS evaluation on relatively inexpensive computing platforms with hybrid memory architecture. Representative results have demonstrated the accuracy and efficiency of parallel MLFMA. There is still some room for improvement in future work to achieve higher parallel scalability.

## ACKNOWLEDGMENT

The authors are grateful for technical support from Dr. Idesbald Van den Bosch, European Patent Office.

## REFERENCES


1. Valagiannopoulos, C. A. and N. L. Tsitsas, "Integral equation analysis of a low-profile receiving planar microstrip antenna with a cloaking superstrate," *Radio Science*, Vol. 51, No. 12, 2012.
2. Valagiannopoulos, C. A., "Semi-analytic solution to a cylindrical microstrip with inhomogeneous substrate," *Electromagnetics*, Vol. 27, No. 8, 527–544, 2007.
3. Valagiannopoulos, C. A., "Arbitrary currents on circular cylinder with inhomogeneous cladding and RCS optimization," *Journal of Electromagnetic Waves and Applications*, Vol. 21, No. 5, 665–680, 2007.
4. Zhao, Y., X. W. Shi, and L. Xu, "Modeling with NURBS surfaces used for the calculation of RCS," *Progress In Electromagnetics Research*, Vol. 78, 49–59, 2008.
5. Xu, L., J. Tian, and X. W. Shi, "A closed-form solution to analyze RCS of cavity with rectangular cross section," *Progress In Electromagnetics Research*, Vol. 79, 195–208, 2008.
6. Li, X. F., Y. J. Xie, and R. Yang, "Bistatic RCS prediction for complex targets using modified current marching technique," *Progress In Electromagnetics Research*, Vol. 93, 13–28, 2009.
7. Zhang, G. H., M. Xia, and X. M. Jiang, "Transient analysis of wire structures using time domain integral equation method with exact matrix elements," *Progress In Electromagnetics Research*, Vol. 92, 281–298, 2009.
8. Valagiannopoulos, C. A., "An overview of the Watson transformation presented through a simple example," *Progress In Electromagnetics Research*, Vol. 75, 137–152, 2007.
9. Song, J., C. C. Lu, and W. C. Chew, "Multilevel fast multipole algorithm for electromagnetic scattering by large complex objects," *IEEE Transactions on Antennas & Propagation*, Vol. 45, No. 10, 1488–1493, 2002.
10. Velamparambil, S., W. C. Chew, and J. Song, "10 million unknowns: Is it that big?," *IEEE Antennas & Propagation Magazine*, Vol. 45, No. 2, 43–58, 2003.
11. Velamparambil, S. and W. C. Chew, "Analysis and performance of a distributed memory multilevel fast multipole algorithm," *IEEE Transactions on Antennas & Propagation*, Vol. 53, No. 8, 2719–2727, 2005.
12. Fostier, J., B. Michiels, et al., "Solving billions of unknowns using the parallel MLFMA and a Tier 1 supercomputer," *IEEE Radio Science Conference*, 1–1, 2015.
13. Nguyen, N. and D. Bein, "Distributed MPI cluster with Docker swarm mode," *IEEE Computing and Communication Workshop and Conference*, 1–7, 2017.
14. Sterling, T., "BEOWULF: A parallel workstation for scientific computation," *International Conference on Parallel Processing*, 11–14, 1995.
15. Merkel, D., "Docker: Lightweight Linux containers for consistent development and deployment," *Linux Journal*, No. 2, 2014.
16. Pan, X. M. and X. Q. Sheng, "A highly efficient parallel approach of multi-level fast multipole algorithm," *Acta Electronica Sinica*, Vol. 20, No. 8, 1081–1092, 2007.
17. Takrimi, M., E. Özgür, and V. B. Ertürk, "A novel broadband multilevel fast multipole algorithm with incomplete-leaf tree structures for multiscale electromagnetic problems," *IEEE Transactions on Antennas & Propagation*, Vol. 64, No. 6, 2445–2456, 2016.