Strategies for Improving the Use of the Memory Hierarchy in an Implementation of the Modified Equivalent Current Approximation (MECA) Method

Hipólito Gómez-Sousa[1], José Á. Martínez-Lorenzo[1], Oscar Rubiños-López[1], Javier G. Meana[2], María Graña-Varela[1], Borja Gonzalez-Valdes[1], and Marcos Arias-Acuña[1]

[1] Department of Signal Theory and Communications
University of Vigo, ETSI de Telecomunicación, Campus Universitario, E-36310 Vigo, Spain
{hgomez, oscar}@com.uvigo.es

[2] Department of Electrical Engineering
University of Oviedo, Edificio Polivalente de Viesques, Campus Universitario, E-33203 Gijón, Spain

Abstract ─ In this paper, we investigate different techniques for improving the cache memory use when running a parallel implementation of the modified equivalent current approximation (MECA) method. The MECA method allows the analysis of dielectric and lossy geometries, and reduces to the well-studied physical optics (PO) formulation in case of PEC scatterers. We discuss several memory-hierarchy-based optimization techniques and present how to implement them in C. We show through simulations that these optimization strategies are effective for reducing the total execution time when calculating the scattered fields with a parallel implementation of the MECA method.

Index Terms ─ Memory-hierarchy-based optimization, parallel programming, physical optics (PO).

I. INTRODUCTION

Physical optics (PO) is a well-known asymptotic high frequency computational technique that is widely used in computing the electromagnetic scattering from electrically large and complex structures [1, 2]. In contrast to full wave methods, like the method of moments (MoM), PO does not need a huge amount of computational resources to solve these problems with a high degree of accuracy and efficiency.

The modified equivalent current approximation (MECA) method [3, 4] has extended PO to lossy materials with a complex effective permittivity by calculating the equivalent electric and magnetic currents based on the oblique incidence of a plane wave on the interface, together with a field decomposition into TE and TM components. Unlike the method of stationary phase, the surface is discretized into flat triangular facets where the current distribution has constant amplitude and linear phase variation. As a consequence, the radiation integral can be solved analytically and so problems which are prohibitive for full-wave simulation, especially at very high frequencies, are successfully modelled by MECA.

On the other hand, we have witnessed the emergence and sustained growth and improvement of high-speed and high-capacity personal computers during the last years. New programming paradigms can be used in order to improve the performances of the computational techniques (MPI, OpenMP). Two recent papers [5, 6] have analyzed parallel implementations of electromagnetic modeling codes which have been tested on several high-performance computer systems. To reduce the total runtime of MECA, we have also developed a parallel version of the code. We have selected an OpenMP paradigm because it can be applied in shared memory machines. In addition, we have implemented memory-

hierarchy-based optimization techniques [10, 11] in our code.

In this paper, we demonstrate the usefulness of employing all these computational techniques to take advantage of the new computational resources available in personal computers. The main content of this paper is divided as follows. Section II gives an overview of the MECA method and briefly explains our parallel algorithm. In Section III, the proposed techniques for improving the use of the memory hierarchy are addressed. Afterwards, performance results obtained through simulations are presented in Section IV.

II. PARALLEL IMPLEMENTATION OF MECA

A. The MECA method

In the MECA method, the equivalent magnetic and electric current densities at the barycenter of each facet are calculated according to the following two equations respectively:

$$\mathbf{M_{i0}} = E^i_{TE}\left(1+R_{TE}\right)\left(\hat{\mathbf{e}}_{\mathbf{TE}}\times\hat{\mathbf{n}}_{\mathbf{i}}\right) + \\ + E^i_{TM}\cos(\theta_i)\left(1+R_{TM}\right)\hat{\mathbf{e}}_{\mathbf{TE}}\Big|_{S_i}, \quad (1)$$

$$\mathbf{J_{i0}} = \frac{E^i_{TE}}{\eta_1}\cos(\theta_i)\left(1-R_{TE}\right)\hat{\mathbf{e}}_{\mathbf{TE}} + \\ + \frac{E^i_{TM}}{\eta_1}\left(1-R_{TM}\right)\left(\hat{\mathbf{n}}_{\mathbf{i}}\times\hat{\mathbf{e}}_{\mathbf{TE}}\right)\Big|_{S_i}, \quad (2)$$

where η_1 is the impedance of the medium of incidence, and R_{TE} (R_{TM}) is the TE (TM) reflection coefficient. For the expressions of R_{TE} and R_{TM}, see [4, 12]. As shown in Fig. 1, $\mathbf{E^i_{TE}} = E^i_{TE}\hat{\mathbf{e}}_{\mathbf{TE}}$ and $\mathbf{E^i_{TM}} = E^i_{TM}\hat{\mathbf{e}}_{\mathbf{TM}}$ are the TE and TM components of the incident electric field at the barycenter of surface S_i, $\hat{\mathbf{p}}_{\mathbf{i}}$ is a unit vector pointing in the propagation direction of the incident wave, θ_i is the angle of incidence, and $\hat{\mathbf{n}}_{\mathbf{i}}$ is the outward unit normal vector to the triangular patch S_i. The first medium is characterized by its constitutive parameters: permittivity ε_1, permeability μ_1, and conductivity σ_1. Similarly, the second medium is characterized by $(\varepsilon_2, \mu_2, \sigma_2)$.

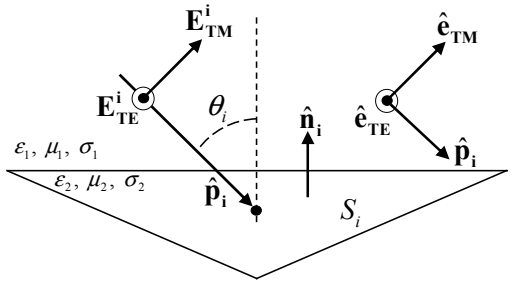


Fig. 1. Oblique wave incidence on a triangular facet S_i.

After obtaining $\mathbf{M_{i0}}$ and $\mathbf{J_{i0}}$, an analytical solution for the radiation integral at the observation point $\mathbf{r_k}$, located in the far field of each triangular patch, can be derived. The scattered electric field $\mathbf{E^s_k}$ at $\mathbf{r_k}$ due to the contribution of all the facets i of a given mesh geometry can be stated as [13]:

$$\mathbf{E^s_k} = \frac{j}{2\lambda}\sum_i \frac{e^{-jk_1 r_{ik}}}{r_{ik}}\left[\mathbf{E^a_{ik}} - \eta_1 \mathbf{H^a_{ik}}\times\hat{\mathbf{r}}_{\mathbf{ik}}\right], \quad (3)$$

where λ is the wavelength, k_1 is the wave number in the medium of incidence, and $\mathbf{r_{ik}} = r_{ik}\hat{\mathbf{r}}_{\mathbf{ik}}$ is the position vector from the barycenter $\mathbf{r_i}$ of the i-th facet to the observation point $\mathbf{r_k}$. Figure 2 summarizes the notation for the position vectors involved in the scattering calculations throughout this paper.

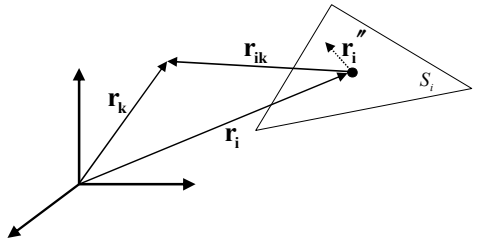


Fig. 2. Facet S_i, observation point $\mathbf{r_k}$ and the corresponding position vectors. $\mathbf{r_i}''$ is a variable vector from barycenter $\mathbf{r_i}$ to any point on S_i.

Assuming that the currents have constant amplitude and linear phase variation depending on the direction of propagation $\hat{\mathbf{p}}_{\mathbf{i}}$ of the incident

wave, the vector values $\mathbf{E}^{\mathbf{a}}_{\mathbf{ik}}$ and $\mathbf{H}^{\mathbf{a}}_{\mathbf{ik}}$ in Eq. (3) can be calculated as [4]:

$$\mathbf{E}^{\mathbf{a}}_{\mathbf{ik}} = (\hat{\mathbf{r}}_{\mathbf{ik}} \times \mathbf{M}_{\mathbf{i0}})\, I_i(\hat{\mathbf{r}}_{\mathbf{ik}}), \tag{4}$$

$$\mathbf{H}^{\mathbf{a}}_{\mathbf{ik}} = (\hat{\mathbf{r}}_{\mathbf{ik}} \times \mathbf{J}_{\mathbf{i0}})\, I_i(\hat{\mathbf{r}}_{\mathbf{ik}}), \tag{5}$$

where $\mathbf{M}_{\mathbf{i0}}$ and $\mathbf{J}_{\mathbf{i0}}$ are the current densities in Eqs. (1) and (2), and $I_i(\hat{\mathbf{r}}_{\mathbf{ik}})$ is an integral given by:

$$I_i(\hat{\mathbf{r}}) = \int_{S_i} e^{jk_1(\hat{\mathbf{r}}-\hat{\mathbf{p}}_{\mathbf{i}})\cdot\mathbf{r}_{\mathbf{i}}''}\, ds_i. \tag{6}$$

$\mathbf{r}_{\mathbf{i}}''$ denotes a vector from the barycenter $\mathbf{r}_{\mathbf{i}}$ of the i-th facet to the source points on the triangular surface S_i (see Fig. 2). In the particular case that the observation point $\mathbf{r}_{\mathbf{k}}$ is in the absolute far field of the whole structure, then $\hat{\mathbf{r}}_{\mathbf{ik}} \approx \hat{\mathbf{r}}_{\mathbf{k}}$ and $r_{ik} \approx r_k$ for all the values of i, resulting:

$$\mathbf{E}^{\mathbf{a}}_{\mathbf{ik}} = e^{jk_1\hat{\mathbf{r}}_{\mathbf{k}}\cdot\mathbf{r}_{\mathbf{i}}}(\hat{\mathbf{r}}_{\mathbf{k}} \times \mathbf{M}_{\mathbf{i0}})\, I_i(\hat{\mathbf{r}}_{\mathbf{k}}), \tag{7}$$

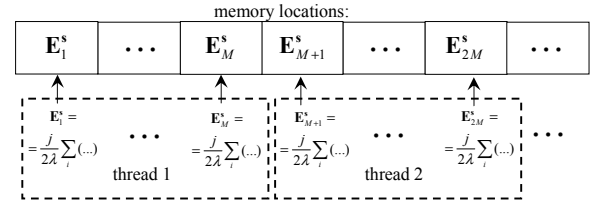
$$\mathbf{H}^{\mathbf{a}}_{\mathbf{ik}} = e^{jk_1\hat{\mathbf{r}}_{\mathbf{k}}\cdot\mathbf{r}_{\mathbf{i}}}(\hat{\mathbf{r}}_{\mathbf{k}} \times \mathbf{J}_{\mathbf{i0}})\, I_i(\hat{\mathbf{r}}_{\mathbf{k}}). \tag{8}$$

The explained current distributions allow modeling with facets larger than those employed in other approaches. This fact implies a computational cost decrease in terms of both time and memory.

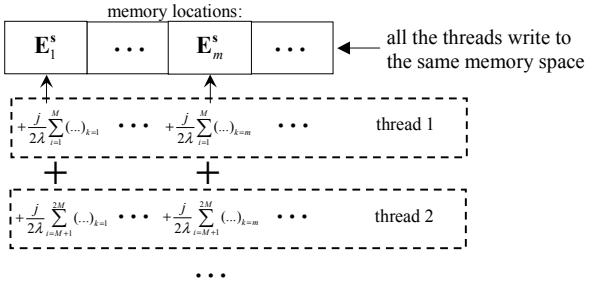
The integral in Eq. (6) always has an analytical solution [14]. The method for analytically solving this integral is summarized in Appendix I.

B. Parallel algorithm

In our parallel implementation of the MECA method, each thread computes the scattered fields $\mathbf{E}^{\mathbf{s}}_{\mathbf{k}}$ in a set of observation points $\mathbf{r}_{\mathbf{k}}$, *i.e.*, calculates all the summation terms in (3) for a range of values of k. A thread is a piece of computational work that runs independently. A program is parallel if more than one thread is executed concurrently during a time interval. We have selected the implementation strategy described above, instead of using each thread to compute, for all the observation points, the scattered fields due to a set of facets. As can be seen in Fig. 3, the chosen approach ensures that the threads do not compete for writing in the memory locations which must contain the values of $\mathbf{E}^{\mathbf{s}}_{\mathbf{k}}$ at the end of the parallel program. If each thread were utilized to calculate the contribution of a set of triangular patches, the runtime would be increased because some threads would have to wait to write their partial calculations of $\mathbf{E}^{\mathbf{s}}_{\mathbf{k}}$ (see Fig. 3b). These delays could be avoided using private variables for each OpenMP thread in order to store the partial results and, once all the threads have finished executing, employ these partial sums to compute the total values. Nevertheless, this solution can drastically increase the memory usage.



a) Each thread computes for a set of observation points (our implemented approach)



b) Each thread computes for a set of facets

Fig. 3. Different strategies for implementing the parallelization of MECA.

From now on we will use the term "task" to refer to an observation point. When allocating tasks to the threads, first we assign `floor(nr/nth)` tasks to each thread, where `nr` denotes the total number of tasks and `nth` represents the number of threads. Each of the remaining `nr-nth*floor(nr/nth)` tasks is allocated to a single thread. As a consequence, in general, some threads compute

`M=floor(nr/nth)` tasks, whereas others compute `M=floor(nr/nth)+1` tasks.

In our parallel version of the MECA method, each thread runs two main nested *for* loops. The outer loop goes through each observation point (index *k* in (3)), while the inner one goes through each facet (index *i* in (3)).

We have chosen the OpenMP paradigm because it provides a portable application programming interface (API) for high-performance parallel programs on shared-memory platforms. In general, OpenMP has better performance on symmetric multiprocessing (SMP) systems than MPI [15]. An SMP system involves a hardware architecture where two or more identical processors are connected to a single shared main memory.

III. MEMORY-HIERARCHY-BASED OPTIMIZATION

A. Loop tiling

Different techniques can be employed to improve the use of the memory hierarchy. One of these techniques is *loop tiling,* whose aim is to increase the reutilization of both instructions and data stored in the cache memory. An improvement in cache data reuse reduces time spent on transferring data from the main memory to the cache and vice versa. Figure 4 and the pieces of C code in Table 1 exemplify *loop tiling*. The implementation of this technique is simple, and it requires adding a new *for* loop, external to the two original loops, as seen in Table 1.

Without *loop tiling,* all the corresponding inner iterations are executed at each outer iteration. On the contrary, if *loop tiling* is applied, only a number `block_size` of original inner iterations are executed per outer loop iteration, as Fig. 4 shows. Once the outer loop iterations are completed, they are executed again for the next blocks of consecutive inner iterations. In our particular case, the blocks of `block_size` inner iterations of a complete run of the outer loop correspond to the same facets. As a consequence, *loop tiling* produces reutilization of instructions and loaded cache data shared by blocks of inner iterations, thus reducing the amount of data moving between the cache and the main memory.

The value of the integer parameter `block_size` which minimizes the execution time must be empirically determined for a particular program and computer. If `block_size` is too large, each iteration of the outer loop could not fit on the cache, preventing total data reutilization and forcing additional cache loads. On the other hand, if `block_size` is too small, there is a certain amount of space not used in the cache for data reutilization, and this fact leads the cache memory to be unnecessarily loaded and unloaded. In order to obtain the optimal value, a set of experimental runs should be executed with a reduced amount of facets. Figure 5 presents the results of our tests for determining the optimal value of `block_size` when the geometry has 2×10^5 facets in the experimental context described in Section IV.

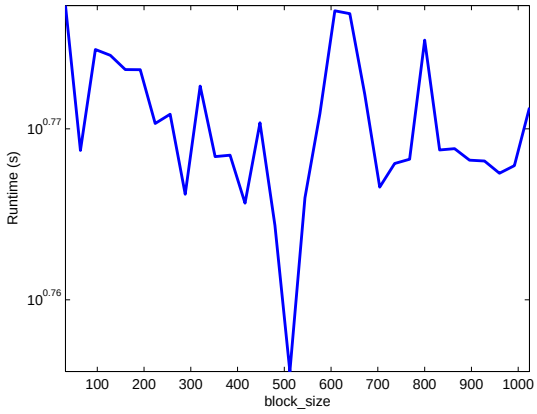


Fig. 5. Runtime vs. `block_size` with 2×10^5 facets. The values of the rest of the parameters can be found in Section IV.

B. Array fusion

Another memory hierarchy based optimization technique is *array fusion*, which consists of defining a single array of `structs` instead of several same-size arrays in the code. In the parallelization of MECA, we regrouped all the arrays which store information relative to the facets: magnetic/electric currents, barycenter points, etc. Let us consider that data is copied from main memory to cache, and back, in blocks of contiguous data. The elements of a `struct` are arranged in the memory in the same order as they are defined; hence, *array fusion* may lower the

total data flow from and to the cache. In our case, this technique reduces the execution time because the currents at barycenter *i* and the coordinates of barycenter *i* are used together to compute Eqs. (4), (5), (7), and (8). Figure 6 and the two pieces of data declaration code in C which appear in Table 2 illustrate array fusion.

Without *array fusion,* the values of the magnetic and electric currents at a particular barycenter and the vector with the coordinates of that barycenter are never stored in contiguous order in the main memory. Therefore, in this case, at least three accesses to main memory could be necessary to move the needed facet data to the cache in order to compute each term of the summation in Eq. (3). On the contrary, only one access could suffice if *array fusion* is used.

In our MECA parallel implementation, the array which contains the observation points cannot be grouped together with the information relative to the facets in the same array of `structs`. The obvious reason is that the number of observation points is generally different from the amount of facets. Moreover, for a given observation point, Eq. (3) must be computed using the information of all the facets, *i.e.*, with *array fusion*, it would be necessary to define additional fields in each `struct` for storing the coordinates of all the required observation points. Clearly, this solution would lead to a much higher memory usage, and, what is more important, the `struct` so defined could not fit on the cache.

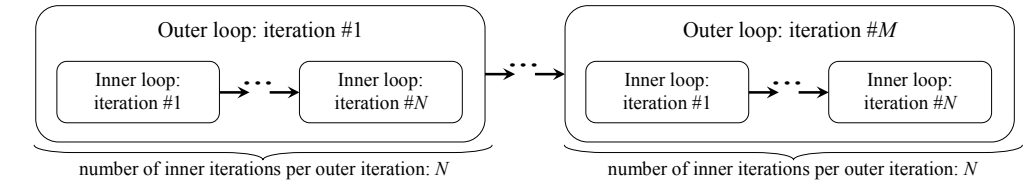
Table 1: Loop tiling implementation

Without *loop tiling:*

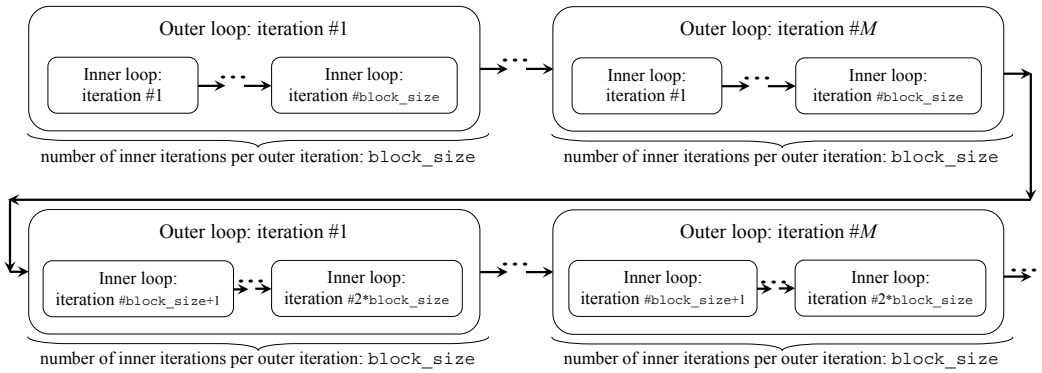
```
for (ind1=0; ind1<M; ind1++)
{
/* THIS OUTER LOOP GOES THROUGH */
/* EACH OBSERVATION POINT       */
   for (ind2=0; ind2<N; ind2++) {
   /* THIS INNER LOOP GOES THROUGH EACH FACET */...}}
```

With *loop tiling:*

```
for(ind_tiling=0;ind_tiling<N;ind_tiling+=block_size){
   for (ind1=0; ind1<M; ind1++) {
   /* THIS OUTER LOOP GOES THROUGH */
   /* EACH OBSERVATION POINT       */
      for (ind2=ind_tiling;
           ind2<MIN(ind_tiling+block_size, N);
           ind2++){
           /* THIS INNER LOOP GOES */
           /* THROUGH EACH FACET   */...}}}
```



a) Without *loop tiling*



b) With *loop tiling*

Fig. 4. Execution order of the loop iterations. With *loop tiling,* there exists an increase in the reutilization of instructions and loaded cache data shared by blocks of inner iterations.

Table 2: Array fusion implementation

Without *array fusion:*

```
double J_Real[3*N], J_Imag[3*N], M_Real[3*N], M_Imag[3*N],
       Barycenter[3*N];

// For instance, { J_Real[3*i+0], J_Real[3*i+1], J_Real[3*i+2] } are the
// 3 real Cartesian components of the electric current density at
// barycenter i>=0.
// Single-dimensional arrays are used here to ensure that the 3
// components of each vector are contiguous in memory independently of
// the programming language, the compiler and the platform.
```

With *array fusion:*

```
struct reg{
    double J_Real[3], J_Imag[3], M_Real[3], M_Imag[3], Barycenter[3];
}
struct reg v[N];

// With array fusion, for each i the following 3-element arrays are
// contiguous in memory:
// v[i].J_Real, v[i].J_Imag, v[i].M_Real, v[i].M_Imag and
// v[i].Barycenter.
```

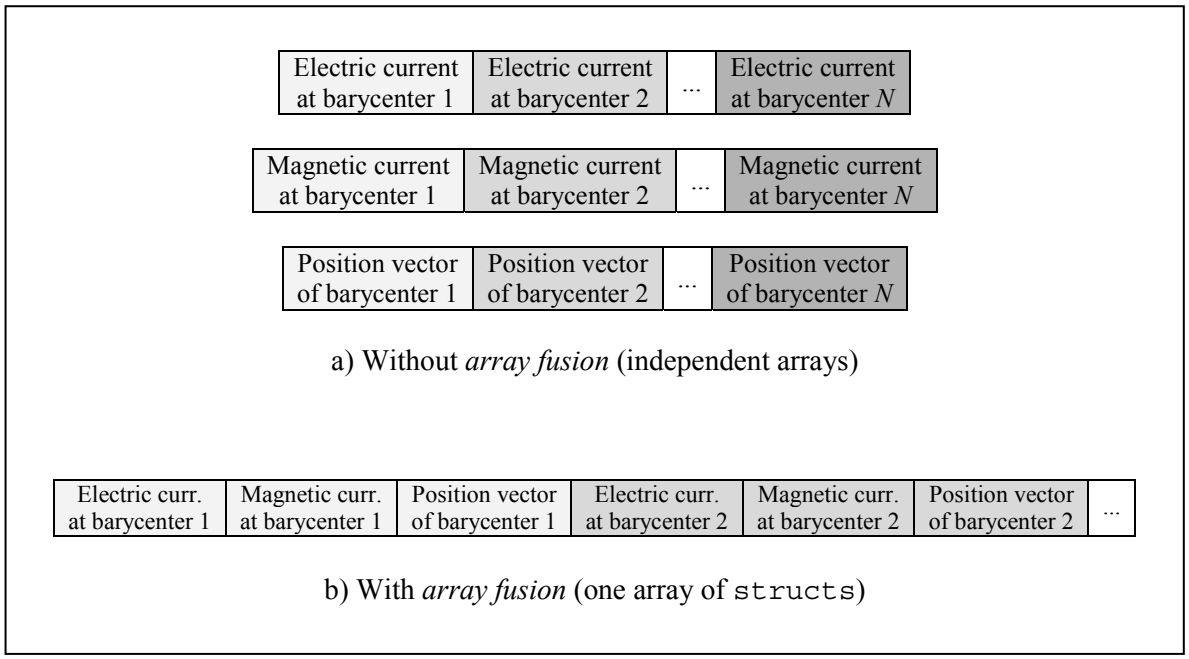


Fig. 6. Storage order in the main memory. With *array fusion,* all the data relative to each facet is stored contiguously.

C. False-sharing reduction

The cache is subdivided into cache lines which represent the minimum amount of data transferable between cache and main memory. These cache lines are organized into C sets of K lines, as explained in Fig. 7. The content of main memory line X can only be stored in cache set X mod C.

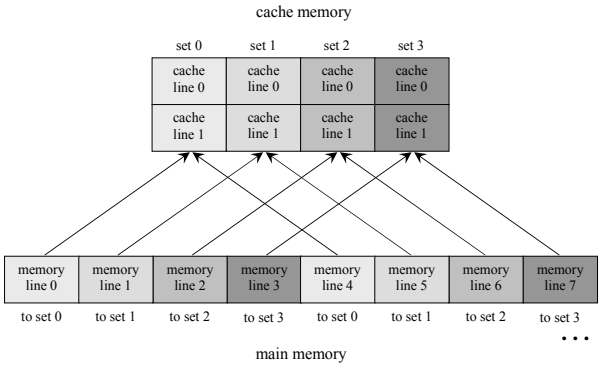


Fig. 7. Example: cache with C=4 sets of K=2 lines each.

When running a parallel application, false sharing occurs when two threads access different data elements in the same cache line for reads and writes. This situation is represented in Fig. 8. This particular problem could seriously degrade the performance of an application because some threads might have to wait until the preceding writing operations in the queue have completed.

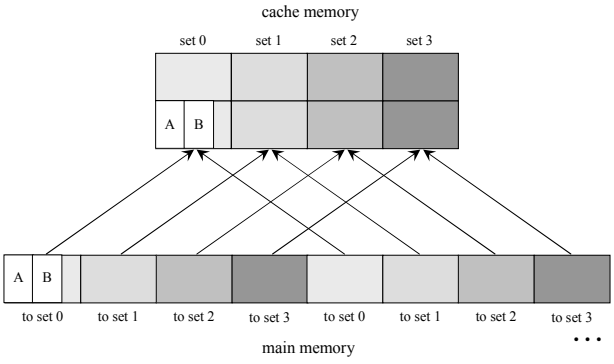


Fig. 8. Example: false sharing occurs when different threads modify variables A and B which reside in the same cache line.

To reduce false sharing, we use the `firstprivate` clause on the OpenMP task `pragma`. This clause declares one or more input variables to be private to a thread, and initializes each of these variables with the value that the corresponding original variable has when the task `pragma` is encountered. As we have seen, the utilization of `firstprivate` in our parallel program increases the performance.

IV. RESULTS

The algorithm for calculating the scattered field in Eq. (3) was implemented in C using the OpenMP library. We ran our parallel algorithm on a server with two Quad-Core Intel® Xeon® processors with 6 MB of L2 cache per processor, yielding a total of eight cores, each core running at 2.66 GHz. The parameter `block_size` was set to the value 512 (the optimal value, presented in Fig. 4), the number of observation points was 722, the maximum number of facets was 5×10^6, and we used 8 threads. A square plate geometry was chosen for our performance tests. Figures 9 and 10 show the total runtime as a function of the number of facets for our parallel implementation with and without all the optimization techniques described in Section III. For comparison, these figures, also, include the total runtime for a MATLAB® single-thread version of the MECA method, *i.e.*, a sequential program version in MATLAB®, without memory-hierarchy-based optimization techniques, executed on the same computer.

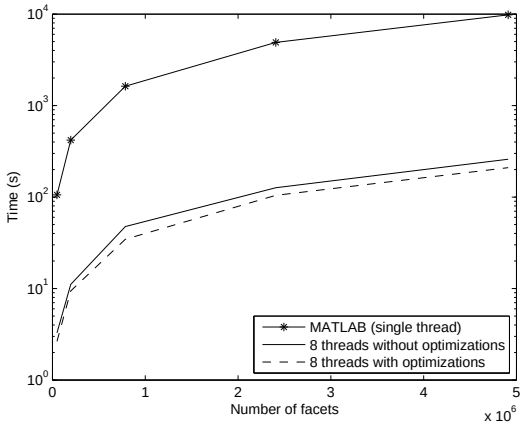


Fig. 9. Runtime vs. number of facets for near-field calculations.

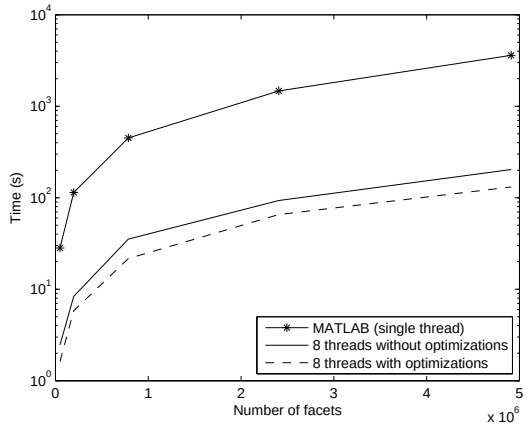


Fig. 10. Runtime vs. number of facets for far-field calculations.

Under the experimental conditions explained above, and employing the `firstprivate` clause, Figs. 11 and 12 show the effect on the runtime of the separate use of *array fusion* and *loop tiling*.

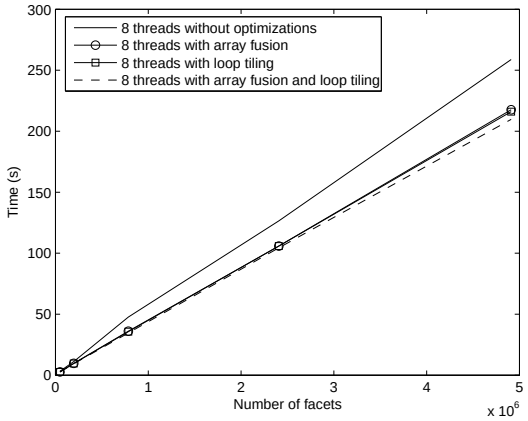


Fig. 11. Runtime vs. number of facets for near-field calculations with the separate use of *array fusion* and *loop tiling*.

The parallel speedup values under the experimental conditions described above were obtained varying the number of threads, up to the number of cores. In our particular case, it was noted that having more threads than cores degrades performance compared to the optimal solution of using as many threads as cores. The experimental speedup is defined by the following formula: $speedup(n \text{ threads}) = \frac{runtime(1 \text{ thread})}{runtime(n \text{ threads})}$.

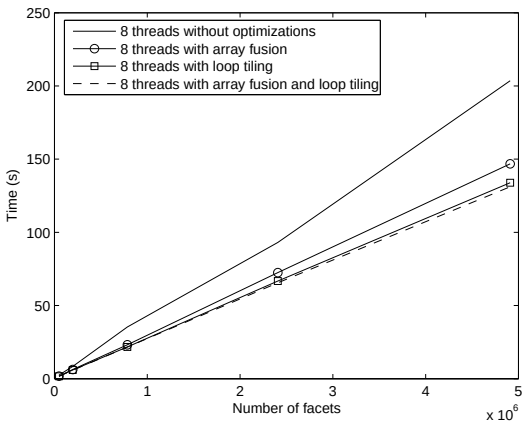


Fig. 12. Runtime vs. number of facets for far-field calculations with the separate use of *array fusion* and *loop tiling*.

The influence of the `firstprivate` clause is represented in Fig. 13.

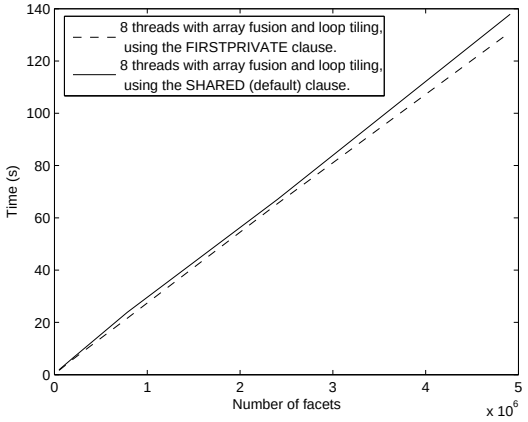


Fig. 13. Runtime vs. number of facets for far-field calculations, with and without the `firstprivate` clause.

The speedup so calculated, up to the number of real cores, allows inferring the scalability of our program, namely the optimal performance behavior of the program as a function of the number of cores.

If n threads are used in an ideal scenario with n cores, the runtime is reduced by a factor of n when compared to the runtime of one thread. The reason is that the total computational load is distributed evenly amongst the threads, and each thread is executed by a core. Then the ideal speedup is

simply *speedup*(n threads)=n . In a real scenario, the speedup achieved is lower due to constraints such as the effect of accessing a shared memory or the communication times between threads. In our parallel algorithm, communication between the threads is not performed, but multiple threads simultaneously access the same shared memory.

The scalability obtained through simulations is illustrated in Fig. 14, showing the experimental speedup. Looking at this figure, the excellent scalability of our parallel implementation is clear.

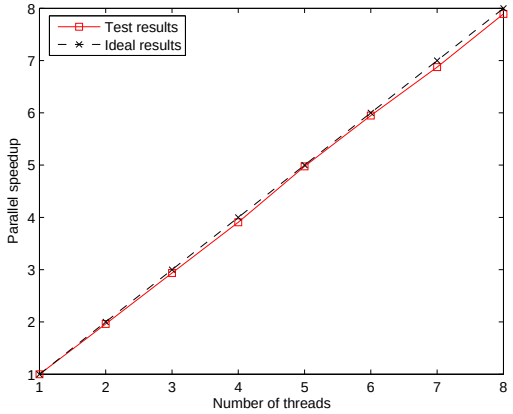


Fig. 14. The experimental speedup as a function of the number of threads (each thread assigned to a real core) for far-field calculations and a scattering problem with 5×10^6 facets.

Finally, Figs. 15 and 16 show a comparison between the far-field results of MECA and MoM. The geometry consists of a square PEC plate whose length is 3 cm, located in the XY plane. The incident field is a plane wave polarized along the direction $\hat{\theta}$, with amplitude 1 V/m, f = 94 GHz, and normal incidence on the interface.

V. CONCLUSION

This paper presents a parallel version of a modified PO method, known as the modified equivalent current approximation (MECA) method, valid for both PEC and dielectric objects. Our experimental results show that the computational performance of this parallel implementation is increased by applying techniques to improve the use of the memory hierarchy. These optimization techniques are

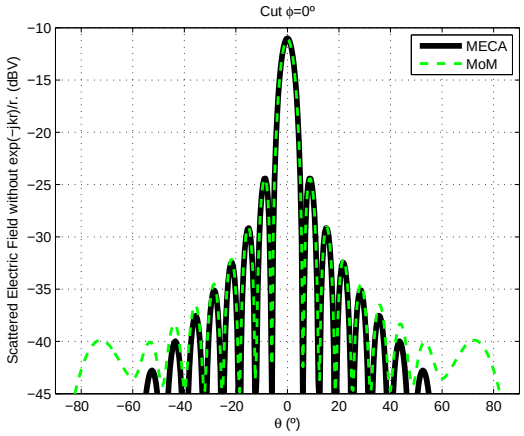


Fig. 15. Comparison between MoM and MECA solutions along the observation cut $\phi = 0º$ for a square PEC scatterer of side length 3 cm.

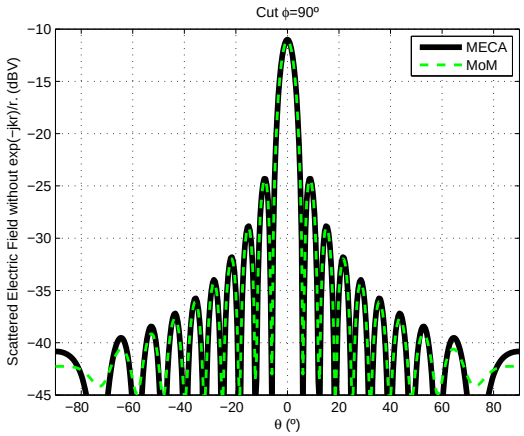


Fig. 16. Comparison between MoM and MECA solutions along the observation cut $\phi = 90º$ for a square PEC scatterer of side length 3 cm.

simple and easy to understand, and they could be very effective in improving the programmed algorithm performance (either sequential or parallel programs) in many other methods for calculating scattered fields.

ACKNOWLEDGMENT

This work was supported by the Spanish Government Grants CONSOLIDER-INGENIO 2010 CSD2008-00068 and “Ramón y Cajal” RYC-2009-04180, and by Xunta de Galicia Grant INCITE08PXIB322219PR.

APPENDIX I

In this appendix, we explain how to solve the integral in Eq. (6) using the procedure detailed in [14].

Let us consider a triangular flat patch as seen in Fig. 17. The triangle i is defined by three points $\mathbf{P_1}$, $\mathbf{P_2}$, $\mathbf{P_3}$, and $\mathbf{r_i}$ is a reference point placed at the barycenter ($\mathbf{r_i} = (\mathbf{P_1} + \mathbf{P_2} + \mathbf{P_3})/3$). We define $\mathbf{v_{mn}}$ as the vector $\mathbf{v_{mn}} = \mathbf{P_n} - \mathbf{P_m}$. The normal vector $\hat{\mathbf{n}}$ of the triangle i is defined such that $\mathbf{v_{12}} \times \mathbf{v_{13}} = 2A_i\,\hat{\mathbf{n}}$, as seen in Fig. 17, where A_i is the area of this triangle.

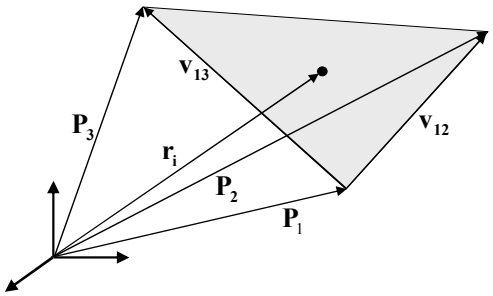


Fig. 17. Triangular patch with barycenter $\mathbf{r_i}$ and vertices $\mathbf{P_1}$, $\mathbf{P_2}$ and $\mathbf{P_3}$.

A coordinate system is used with scalar variables (u,v) such that any point $\mathbf{r_i}''$ on the triangle surface can be described as:

$$\mathbf{r_i}'' = \mathbf{P_1} - \mathbf{r_i} + u \cdot \mathbf{v_{12}} + v \cdot \mathbf{v_{13}}. \tag{9}$$

The integral (6) is then given by:

$$I_i(\hat{\mathbf{r}}) = 2A_i\, e^{-j\frac{\alpha+\beta}{3}} \int_{u=0}^{u=1} \int_{v=0}^{v=1-u} e^{j(\alpha u + \beta v)}\, dv du, \tag{10}$$

whose solution is

$$I_i(\hat{\mathbf{r}}) = 2A_i\, e^{-j\frac{\alpha+\beta}{3}} \left[\frac{\alpha e^{j\beta} - \beta e^{j\alpha} + \beta - \alpha}{\alpha\beta(\alpha-\beta)} \right], \tag{11}$$

where

$$\alpha = k_1 \mathbf{v_{12}}(\hat{\mathbf{r}} - \hat{\mathbf{p}}_\mathbf{i}), \tag{12}$$

$$\beta = k_1 \mathbf{v_{13}}(\hat{\mathbf{r}} - \hat{\mathbf{p}}_\mathbf{i}). \tag{13}$$

The expression (11) has the following singular values:

$$\alpha = 0, \beta \neq 0 \Rightarrow I_i(\hat{\mathbf{r}}) = 2A_i\, e^{-j\frac{\beta}{3}} \frac{1 + j\beta - e^{j\beta}}{\beta^2}, \tag{14}$$

$$\alpha \neq 0, \beta = 0 \Rightarrow I_i(\hat{\mathbf{r}}) = 2A_i\, e^{-j\frac{\alpha}{3}} \frac{1 + j\alpha - e^{j\alpha}}{\alpha^2}, \tag{15}$$

$$\alpha = \beta \neq 0 \Rightarrow I_i(\hat{\mathbf{r}}) = 2A_i\, e^{j\frac{\alpha}{3}} \frac{1 - j\alpha - e^{-j\alpha}}{\alpha^2}, \tag{16}$$

$$\alpha = \beta = 0 \Rightarrow I_i(\hat{\mathbf{r}}) = A_i. \tag{17}$$

REFERENCES

[1] C. Uluisik, G. Cakir, M. Cakir, and L. Sevgi, "Radar cross section (RCS) modeling and simulation, part 1: a tutorial review of definitions, strategies, and canonical examples," *Antennas and Propagation Magazine, IEEE,* vol. 50, no. 1, pp. 115-126, Feb. 2008.

[2] J. A. M. Lorenzo, A. G. Pino, I. Vega, M. Arias, and O. Rubiños, "ICARA: induced-current analysis of reflector antennas," *Antennas and Propagation Magazine, IEEE,* vol. 47, no. 2, pp. 92-100, April 2005.

[3] J. G. Meana, J. A. M. Lorenzo, F. Las-Heras, and C. Rappaport, "A PO-MoM comparison for electrically large dielectric geometries," *Antennas and Propagation Society International Symposium,* 2009. APSURSI '09. IEEE, 1-5 June 2009.

[4] J. G. Meana, J. A. M. Lorenzo, F. Las-Heras, and C. Rappaport, "Wave scattering by dielectric and lossy materials using the Modified Equivalent Current Approximation (MECA)," *Transactions on Antennas and Propagation, IEEE,* vol. 58, no. 11, pp. 3757-3761, 2010.

[5] D. Daroui and J. Ekman, "Parallel Implementations of the PEEC Method," *ACES Journal,* vol. 25, no. 5, pp. 410-422, 2010.

[6] R. J. Burkholder, Ç. Tokgöz, C. J. Reddy, and W. O. Coburn, "Iterative Physical Optics for Radar Scattering Predictions," *ACES Journal,* vol. 24, no. 2, pp. 241-258, 2009.

[7] S. R. Rengarajan and E. S. Gillespie, "Asymptotic approximations in radome analysis," *Transactions on Antennas and Propagation, IEEE,* vol. 36, no. 3, pp. 405-414, 1988.

[8] R. E. Hodges and Y. Rahmat-Samii, "Evaluation of dielectric physical optics in electromagnetic scattering," in *Proceedings 1993 Antennas and Propagation Society International Symposium,* EE.UU., June 1993.

[9] F. Sáez de Adana, I. González, O. Gutiérrez, P. Lozano and M. F. Cátedra, "Method based on physical optics for the computation of the radar

cross section including diffraction and double effects of metallic and absorbing bodies modeled with parametric surfaces," *Transactions on Antennas and Propagation, IEEE,* vol. 52, no. 12, pp. 3295-3303, 2004.

[10] B. B. Fraguela, "Optimization of the use of the memory hierarchy," *Course Notes,* Department of Electronics and Systems, University of A Coruña, Spain, October 2009.

[11] G. Wu, J. Xu, Y. Dou, and M. Wang, "Computation rotating for data reuse," *Computer Systems Architecture Conference,* 2008. ACSAC 2008. 13th Asia-Pacific. Hsinchu, August 2008.

[12] D. H. Staelin, A. W. Morgenthaler, and J. A. Kong, *Electromagnetic Waves,* USA: Prentice Hall, 1994.

[13] C. A. Balanis, *Advanced Engineering Electromagnetics,* 1st ed. New York, USA: John Wiley and Sons, 1989.

[14] M. Arias, O. Rubiños, I. Cuiñas, and A. G. Pino, "Electromagnetic scattering of reflector antennas by fast physical optics algorithms," *Recent Res. Devel. Magnetics,* no. 1, pp. 43-63, 2000.

[15] G. Krawezik and F. Cappello, "Performance comparison of MPI and OpenMP on shared memory multiprocessors," *Concurrency Computat.: Pract. Exper.,* vol. 18, no. 1, pp. 29-61, Oct. 2005.

Hipólito Gómez-Sousa received the M.S. degree in telecommunications engineering from the University of Vigo, Vigo, Spain, in 2009.

Since 2009, he has been with the Department of Signal Theory and Communications, University of Vigo. His current research interests are on computational electromagnetism, THz sensing systems, and quantum cryptography.

José Ángel Martínez-Lorenzo (S'03–M'05) was born in Madrid, Spain, in 1979. He received the M.S. and Ph.D. degrees in telecommunications engineering from the University of Vigo, Vigo, Spain, in 2002 and 2005, respectively.

He was a Teaching and Research Assistant with the University of Vigo from 2002 to 2004. He joined the faculty at the University of Oviedo, Gijón, Spain, in 2004, where he was an Assistant Professor with the Department of Signal Theory and Communications until 2006. During the spring and summer of 2006, he was a Visiting Researcher with the Bernard Gordon Center for Subsurface Sensing and Imaging Systems (Gordon-CenSSIS), Northeastern University, Boston, MA. He was appointed as a Research Assistant Professor with the Department of Electrical and Computer Engineering, Northeastern University. He is currently a Ramon y Cajal researcher at the University of Vigo. He has authored over 80 technical journal and conference papers. His research is geared toward the understanding, modeling, and quantitative prediction of complex electromagnetic problems with special application to security sensing systems, communication systems, and biomedical systems.

Oscar Rubiños-López obtained the M.S. and Ph.D. degrees in telecommunication engineering from the Universidad de Vigo in 1991 and 1997, respectively.

He joined the Universidad de Vigo in 1991 and is currently an associate professor with the Dept. of Signal Theory & Communications at the Universidad de Vigo. From 2001 to 2006, he held the position of Vice-President of University Extension (2001-2002) and for University Extension and Students at the University of Vigo. His research interests include: the analysis and design of broadband antennas, numerical simulation of applied electromagnetic problems, terahertz technology for electromagnetic sensing applications, satellite systems and wireless communications

Javier Gutiérrez-Meana was born in Gijón, Spain, in 1982. He received his M.S. and Ph.D. degrees in electrical engineering from the University of Oviedo (Spain) in 2005 and 2010, respectively.

He joined the R and D department of CTIC Foundation in 2005, and since 2006, he is a Research Assistant with the Area of Theory of Signal and Communications (University of Oviedo). He was a Visiting Researcher at The Gordon CenSSIS – Northeastern University (Boston) in 2008. His interests and research studies are focus on the evaluation of electromagnetic coverage in rural/urban electrically large scenarios by means of high frequency techniques.

María Graña-Varela received the M.S. and Ph.D. degrees in telecommunication engineering from the University of Vigo, Spain, in 2000 and 2009, respectively.

From 2000 to 2005, she took part in several projects related to signal propagation and antennas design, initially with a research fellowship in the University of Vigo and in the Polytechnique University of Madrid and, from 2002, working in a Spanish telecommunication company as a network planning engineer. From 2005 to nowadays, she is with the Group of Antennas of the University of Vigo, involved in projects to design antennas for spatial communications. Her research interests include reconfigurable reflector antennas and computational electromagnetism.

Borja González-Valdés was born in Gijón, Spain, in 1981. He received the B.S. and M.S. degree in 2006 in telecommunications engineering from the University of Vigo.

Since this moment to present, he has been research grant holder in the Department of Signal Theory and Communications, University of Vigo, Spain. During 2008 and 2009, he was a Visiting Researcher at the CenSSIS (The Bernard M. Gordon Center for Subsurface Sensing and Imaging Systems), Northeastern University, Boston.

He received his Ph.D. in electromagnetic engineering in 2010. His current research interest is geared toward the modeling and simulation of electromagnetic systems, with special application to high-performance reconfigurable reflectors.

Marcos Arias-Acuña was born in Vigo, Spain, on June 1, 1968. He received the Ingeniero de Telecomunicacion degree and Doctor Ingeniero de Telecomunicacion degree from the University of Vigo in 1991 and 1997, respectively.

He is a Profesor Titular since 1998 and has been with the Department of Signal Theory and Communications teaching radio communications since 1992. He has worked in projects related with antennas for satellite and radioastronomy and communication systems such DVB-T, LMDS, and UMTS. His research interests include:

- Reflector antennas, feeder for reflector antennas (arrays, horns...), shaped reflectors.
- Communication systems.
- High frequency techniques for modeling electromagnetic problems.