Applications of
High-Performance Computing
in Engineering
VII

Edited by

C.A. Brebbia
Wessex Institute of Technology, UK

P. Melli
IBM Italia S.P.A., Italy

A. Zanasi
TEMIS Text Mining Solutions, Italy

 WITPRESS Southampton, Boston

The VAMPIR and PARAVER performance analysis tools applied to a wet chemical etching parallel algorithm

S. Boeriu[1], and J. C. Bruch, Jr.[2]
[1]*Center for Computational Science and Engineering, University of California, Santa Barbara, CA 93106*
[2]*Department of Mechanical and Environmental Engineering, and Department of Mathematics University of California, Santa Barbara, CA 93106*

Abstract

A wet chemical etching parallel algorithm is studied using performance analysis tools in order to optimize its performance. The physical problem being solved is a moving boundary problem which is nonlinear and whose etching front is unknown *a priori* at each time. A fixed domain formulation of the problem is discretized and the parallel solution algorithm is of successive over-relaxation type. During the iteration process there is message-passing of data between the processors in order to update the calculations along the interfaces of the decomposed domains. A key theoretical aspect of the approach is the application of a projection operator onto the positive solution domain. This operation has to be applied at each iteration at each computational point.

The VAMPIR and PARAVER performance analysis software are used to analyze and understand the execution behavior of the parallel algorithm such as: communication patterns, processor load balance, computation versus communication ratios, timing characteristics, and processor idle time. This is all done by displays of post-mortem trace-files. Performance bottlenecks can easily be identified at the appropriate level of detail. This will numerically be demonstrated using example test data and comparisons of software capabilities will be made using the Blue Horizon parallel computer at the San Diego Supercomputer Center.

1 Introduction

The moving boundary problem which will be used as the working example for the computer software that will be investigated is wet chemical etching in semiconductor fabrication problems (Bruch et al. [1] and Vuik and Cuvelier [2]). This problem is non-linear and has one boundary which is *a priori* unknown and is obtained in the process of solving the problem. Consult references [1] and [2] for a detailed description and the theoretical considerations for the problem. A short description is given here.

An approximation for the physical problem is shown in Figure 1. Here a gap of width 2a and length L is to be etched in a flat plate. The remainder of the plate is covered with a protective (photoresist) layer. Since it is assumed that L is much larger than 2a, the problem can be considered as two dimensional.

The following four additional simplifying assumptions are used to make the problem tractable. There is no convection in the etching medium; the etching process is isotropic; the thickness of the photoresist layer is infinitely small; and only one component of the etching liquid determines the process.

Figure 2 presents the fixed domain problem that is used for the numerical solution. The etching fluid $\Omega(t)$ is bounded by the outer boundary Γ_1, the photoresist layer $\Gamma_2(t)$, and the moving boundary S(t). $D\backslash\Omega(t)$ denotes part of the solid. Let $\Omega(0)$ be a square region in the caustic fluid which is large enough so that increasing the size of $\Omega(0)$ will not change the etching process.

Let D_1 be a rectangular region in the plate's cross section such that all the etching will occur in D_1 for $t \in [0,T]$. Denote by Γ_3 the slit of the length 2a, $\Gamma_3 = \partial\Omega(0) \cap \partial D_1$. Then the domain of the problem, D, is the union of the upper rectangular region $\Omega(0)$, lower rectangular region D_1 in the plate, and Γ_3 along the boundary between the two, i.e., $D = \Omega(0) \cup D_1 \cup (\partial\Omega(0) \cap \partial D_1)$. Let $\Omega(t)$ be the open region in D at time $t \in (0,T)$ that is wet. Denote by Γ_1 the boundary of $\Omega(0)$ that is not along the mask or Γ_3, and $\Gamma_2(t)$ for the upper

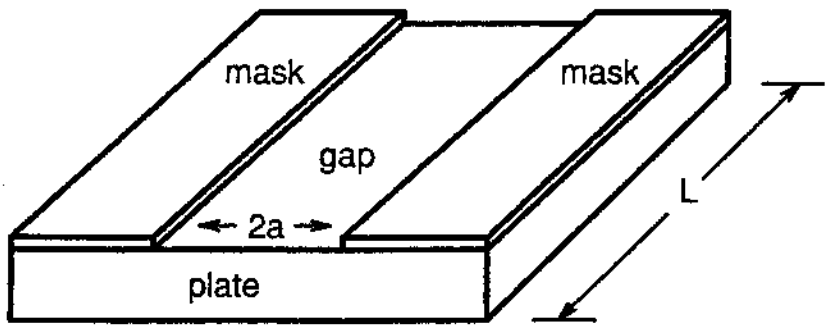

Figure 1: Physical Problem.

and lower surfaces of the mask that are exposed to the caustic fluid.

2 Fixed domain formulation

The approach used to obtain the equations in Figure 2 uses the method of variational inequalities which uses a fixed domain method and a Baiocchi type transformation,

$$w(x,y,t) = \int_0^t \tilde{C}\,(x,y,\tau)\,d\tau \quad (x,y) \in D,\ t \in (0,T) \tag{1}$$

where $\tilde{C}(x,y,t)$ is an intermediate dependent variable which is equal to $C(x,y,t)$, the concentration of the component in $\Omega(t)$, in the etching fluid domain and has been extended continuously across the moving boundary, S(t), into the fixed domain $D\backslash\Omega(t)$, yielding the fixed domain formulation shown. Note Δw stands for the Laplacian of w and $\vec{n}$ is the outward unit normal vector.

$B = \tilde{D}/(\sigma C_0)$ a non-dimensional group, where $C_0 = C(x,y,0)$ being the initial concentration, $\tilde{D}$ is the diffusion coefficient, and σ is a material constant. This is the problem that shall be solved numerically using a parallel supercomputer. If w is a solution of this problem, then $\tilde{C} = \partial w/\partial t$ solves the original problem.

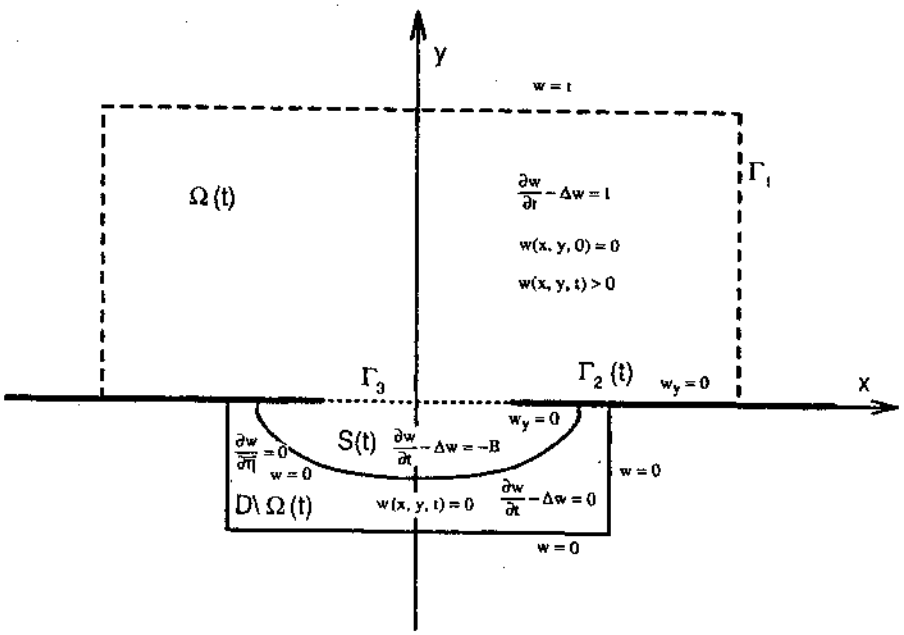

Figure 2: Fixed domain mathematical formulation.

3 Numerical algorithm

The basic numerical algorithm is:

$$(w_1)_{i,j,k}^{(n+1/2)} = \left(\frac{1}{\Delta t}+\frac{2}{(\Delta x)^2}+\frac{2}{(\Delta y)^2}\right)^{-1}\left\{1+\frac{1}{\Delta t}(w_1)_{i,j,k-1}\right.$$
$$\left.+\frac{1}{(\Delta x)^2}\left[(w_1)_{i-1,j,k}^{(n+1)}+(w_1)_{i+1,j,k}^{(n)}\right]+\frac{1}{(\Delta y)^2}\left[(w_1)_{i,j-1,k}^{(n+1)}+(w_1)_{i,j+1,k}^{(n)}\right]\right\} \quad (2)$$

with

$$(w_1)_{i,j,k}^{(n+1)} = (w_1)_{i,j,k}^{(n)}+\theta\left[(w_1)_{i,j,k}^{(n+1/2)}-(w_1)_{i,j,k}^{(n)}\right] \quad (3)$$

in $\Omega(0)$ and

$$(w_2)_{i,j,k}^{(n+1/2)} = \left(\frac{1}{\Delta t}+\frac{32}{(\Delta x)^2}+\frac{32}{(\Delta y)^2}\right)^{-1}\left\{-B+\frac{1}{\Delta t}(w_2)_{i,j,k-1}\right.$$
$$\left.+\frac{16}{(\Delta x)^2}\left[(w_2)_{i-1,j,k}^{(n+1)}+(w_2)_{i+1,j,k}^{(n)}\right]+\frac{16}{(\Delta y)^2}\left[(w_2)_{i,j-1,k}^{(n+1)}+(w_2)_{i,j+1,k}^{(n)}\right]\right\} \quad (4)$$

with

$$(w_2)_{i,j,k}^{(n+1)} = \max\left\{0,(w_2)_{i,j,k}^{(n)}+\theta\left[(w_2)_{i,j,k}^{(n+1/2)}-(w_2)_{i,j,k}^{(n)}\right]\right\} \quad (5)$$

in D_1 where use was made of a first order backwards difference formula for $\partial w/\partial t$, second order central difference formulas for Δw, n denotes the SOR iterate, and θ is the SOR relaxation factor. Since there is a line of symmetry along the y-axis through the middle of the region, only the right half of the region will be solved numerically.

The parallel scheme presented in Bruch et. al. [1] is used on the problem divided into horizontal strips such as the subregion case seen in Figure 3. For the details concerning the parallel algorithm see Bruch et al. [1].

4 Test case

The mathematical problem considered for the test case is shown in Figure 2 and the input entries are:

Maxrow1 (number of rows in the top region) = 280, maxcol1 (number of columns in the top region) = 321, maxrow2 (number of rows in the bottom region) = 80, maxcol2 (number of columns in the bottom region) = 161, maxtime (number of time steps) = 5, Δt (size of time steps) = 1.0, θ (successive

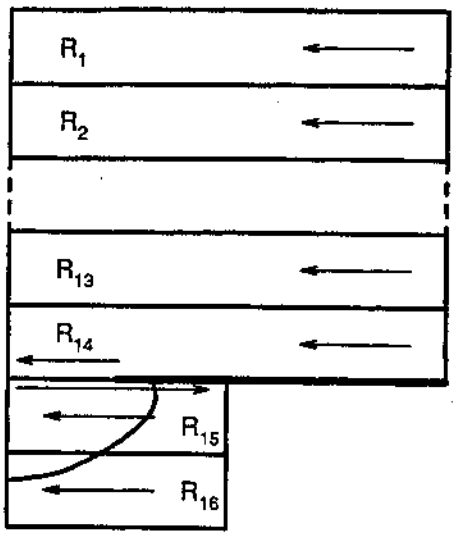

Figure 3: Domain decomposition of mathematical problem into sixteen subregions showing the flow of computations in each.

over-relation factor) = 1.935, and B (non-dimensional number) = 10.00.

The domain decomposition for the 16 node case is shown in Figure 3. The load balancing for the test case is shown in Table 1.

Table 1: Load balancing information for the test case.

Nodes	2	4	8	16	32	64
bottom nodes	1	1	1	2	4	8
bottom points	12880	12880	12880	6440	3220	1610
top nodes	1	3	7	14	28	56
top points	89880	≤ 30174	12840	6420	3210	1605

5 Performance tools and considerations

The VAMPIR and PARAVER performance analysis software are used to analyze and understand the execution behavior of the ETCH program. Performance optimization of parallel programs is dominated by many more different and complex principles than its sequential counterpart. Typically, the parallel program is monitored while it is executed. Monitoring produces performance data that is interpreted in order to reveal areas of poor performance. The program is altered and the process is repeated until an acceptable level of performance is reached.

VAMPIR (Visualization and Analysis of MPI Resources-VAMPIR 2.0 User's Manual [4]) is a post-mortem trace visualization tool from Pallas Gmbh. It uses the profiling extensions to MPI and permits analysis of the message events where data is transmitted between processors during execution of a parallel program. It has a convenient user-interface and an excellent zooming and filtering. Global displays show all selected processes:

-Global Timeline: detailed application execution over time axis.

-Activity Chart: presents per-process profiling information.

-Summaric Chart: aggregated profiling information.

-Communication Statistics: message statistics for each process pair.

-Global Communication Statistics: collective operations statistics.

The Activity Chart display shows a statistic about time spent in each activity for each process. By default, the Activity Chart display gives information for the "execution" time of all activities. The display can be made to show the states belonging to a particular activity by using a submenu of the context menu. By selecting on the activity names, the display will show the "execution" times of all the states belonging to that activity.

Considering the etch test example, all tracing was done on the IBM SP2 Blue Horizon, using a 8-node run. In order to keep the trace-file small only 2 time steps were used.

Selection of "Global Displays/Activity Chart", pops up a window that consists of pie charts for every process [8 nodes]. Load imbalance can be identified by this view – Figure 4. As the number of points per node is almost identical [Table 1], we have to identify the reason of imbalance between node 7 and nodes 0-6. From Figure 3 we see that the number of columns of the top domain is larger than the number of columns of the bottom domain, which gives cache misses and page faults. In fact, the Fortran loops in the ETCH code were written as:

```
do i ...number_of_rows
   do j ... number_of_columns
      matrix(i,j) = ...
   enddo
enddo
```

When a memory location is referenced, its associated cache lines are checked to see if the data is present in cache. If present, the data can be loaded into a register, if not, a CACHE MISS occurs. A cache miss means the processor must now spend cycles trying to find where the requested data actually resides, and then bring that data into cache.

To minimize cache misses, it is necessary to:

- Avoid processing data with large strides. Ideally, data should be processed sequentially as it is stored in memory.
- Avoid cache associativity conflicts. This requires understanding how memory is mapped into cache lines and will vary across different architectures.

Stride Minimization:

- In a loop, STRIDE is defined as the distance between successively accessed elements of a matrix in successive iterations of the loop. For example:

Stride 1:

```
do i = 1,1000
   do j = 1,1000
      C(j,i) = a(j,i)*b(j,i)
   enddo
enddo
```

Stride 1000:

```
do j = 1,1000
   do i = 1,1000
      C(j,i) = a(j,i)*b(j,i)
   enddo
enddo
```

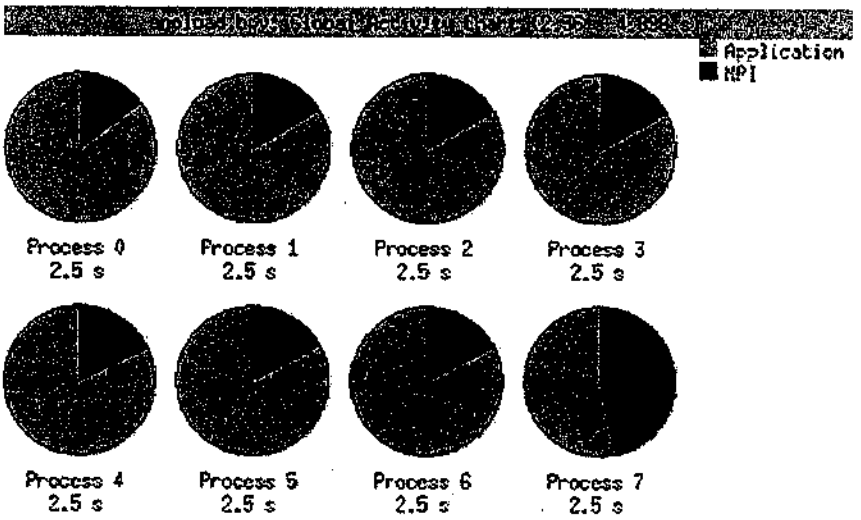

Figure 4: Activity Chart of ETCH.

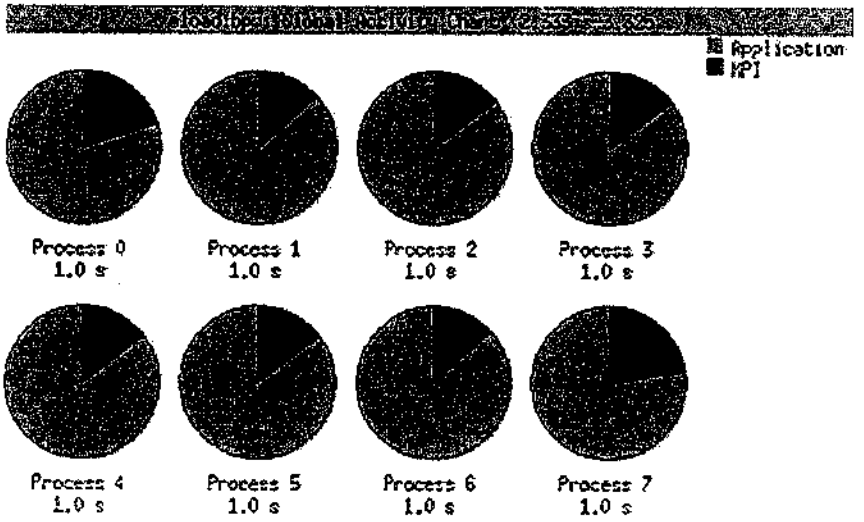

Figure 5: Activity Chart of ETCH.

Processing matrices with minimal stride takes advantage of spatial locality. When any element is referenced, and needs to be brought into cache from memory, an entire line worth of data is brought with it:

- Small stride exploits the extra data brought in with the cache line, since the data to be processed is already in cache.
- Large stride does not exploit the extra data brought in with the cache line, and in fact, may require a new cache line to be loaded for each element accessed.

On the Blue Horizon the following differences between a Stride 1 run and a Stride 1000 run were obtained:

Time [i,j] = 0.107007980346679688
Time [j,i] = 0.440217018127441406

Interchanging the rows with columns in the loops of the ETCH program:

```
do i ...number_of_columns
   do j ... number_of_rows
      matrix(j,i) = ...
   enddo
enddo
```

we obtain the Global Activity Chart presented in Figure 5. There is a much better load balance and the computation time has decreased from an average of 2.1198 sec/node to 0.902 sec/node.

PARAVER (Parallel Program Visualization and Analysis Tool [5]) is a flexible parallel program visualization and analysis tool based on an easy-to-use Motif GUI. PARAVER was developed to respond to the basic need to have a qualitative global perception of the application behavior by visual inspection and then to be able to focus on the detailed quantitative analysis of the problems.

PARAVER provides a large amount of information on the behavior of an application. This information directly improves the decisions on whether and where to invert the programming effort to optimize an application. The result is a reduction of the development time as well as the minimization of the hardware resources required for it.

The results of the test case examples obtained with PARAVER – before and after the cache optimization – are presented in Figures 6 and 7.

The speedups for the various number of nodes are presented in Figure 8.

6 Conclusions

A significant factor that affects the performance of a parallel application is the balance between communication and workload. The challenge of the message passing model is in reducing message traffic over the interconnection network. To fully understand the performance behavior of such applications, analysis and visualization tools are needed. Two such tools, VAMPIR and PARAVER, were

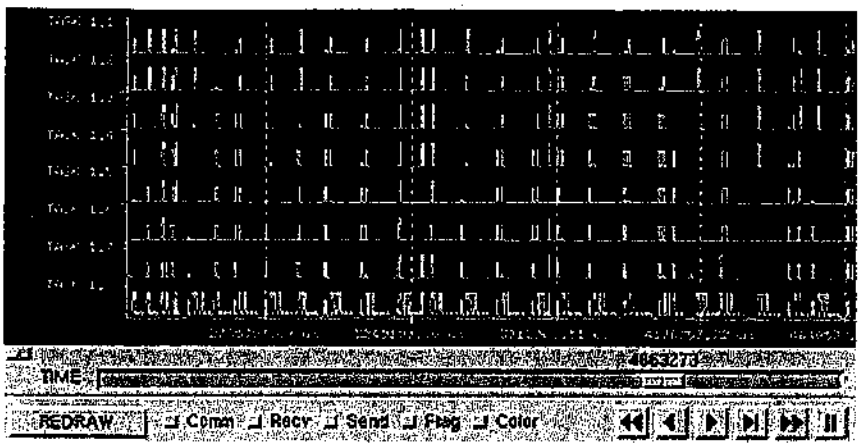

Figure 6: Visualization of the MPI application before cache optimization in the ETCH code.

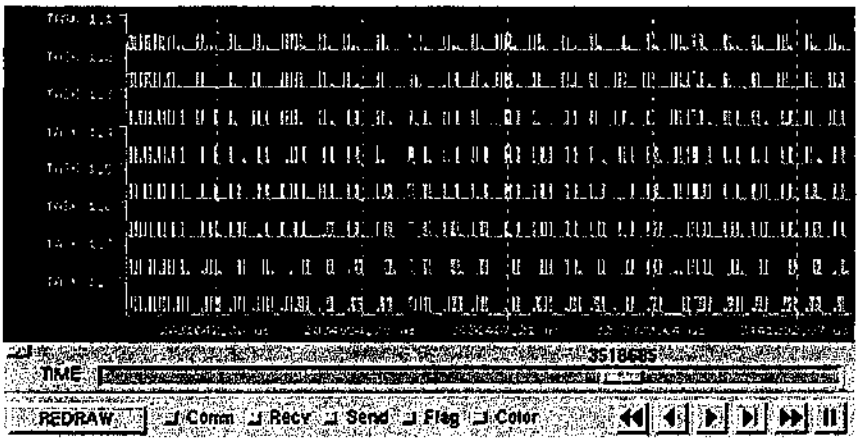

Figure 7: Visualization of the MPI application after cache optimization in the ETCH code.

used to analyze the performance of the etching application. It was seen that optimization of the parallel code can be carried out in an iterative process involving these tools to investigate performance issues.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant #0086262. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. The parallel

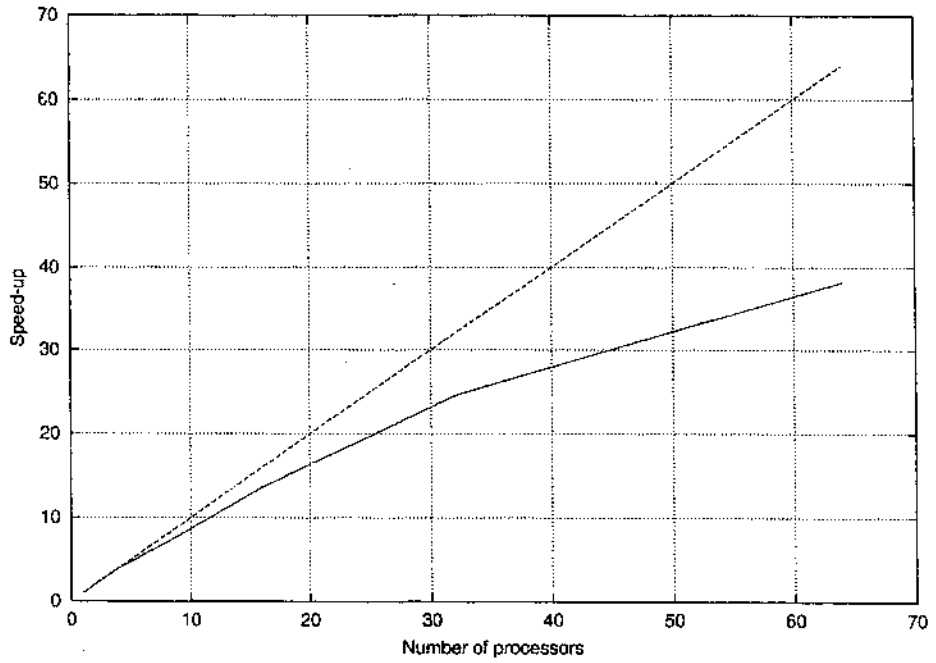

Figure 8: Ideal versus obtained speed up.

algorithm used to solve the moving boundary problem is based upon work supported by the National Science Foundation Grant # ECS-9006516. This research was conducted using the resources of the San Diego Supercomputer Center.

References

[1] Bruch, J. C., Jr., Papadopoulos, C.A., & Sloss, J.M., Parallel Computing Used in Solving Wet Chemical Etching Semiconductor Fabrication Problems, GAKUTO International Series, Mathematical Sciences and Applications, **1**, *Nonlinear Mathematical Problems in Industry*, pp. 281-292, 1993.

[2] Vuik, C. & Cuvelier, C., Numerical Solution of an Etching Problem, *J. Comput. Phys.*, **59**, pp. 247-263, 1985.

[3] Boeriu, S., & Bruch, J. C., Jr., Portable Message Passing Implementation and Performance Analyses for Engineering Application Modules, *Algorithms and Application of Parallel Computing*, edited by H. Power, WIT Press/Computational Mechanics Publications, Chapter 3, pp.73-116, 1999.

[4] VAMPIR, Visualization and Analysis of MPI Resources, User Guide, Pallas Gmbh, http://www.pallas.de.

[5] PARAVER, Parallel Program Visualization and Analysis Tool, Reference Manual, http://www.cepba.upc.es.