Available online at www.sciencedirect.com

SCIENCE DIRECT®

ELSEVIER

Comput. Methods Appl. Mech. Engrg. 194 (2005) 297–312

Computer methods in applied mechanics and engineering

www.elsevier.com/locate/cma

Performance analysis tools applied to a finite element adaptive mesh free boundary seepage parallel algorithm

S. Boeriu [a,*], J.C. Bruch Jr. [b]

[a] Center for Computational Science and Engineering, University of California, Santa Barbara, CA 93106, United States
[b] Department of Mechanical and Environmental Engineering, and Department of Mathematics, University of California, Santa Barbara, CA 93106, United States

Received in revised form 28 May 2004; accepted 28 May 2004

Abstract

A finite element, adaptive mesh, free surface seepage parallel algorithm is studied using performance analysis tools in order to optimize its performance. The physical problem being solved is a free boundary seepage problem which is non-linear and whose free surface is unknown a priori. 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 that will be made using the Blue Horizon parallel computer at the San Diego Supercomputer Center.
© 2004 Published by Elsevier B.V.

Keywords: Performance analysis tools; Finite element; Adaptive mesh; Free boundary seepage; Parallel computing

1. Introduction

The problem studied is the free surface seepage problem shown in **Fig. 1**. The following assumptions are made: the soil in the flowfield is homogeneous and isotropic; capillary and evaporation effects are neglected;

* Corresponding author.

0045-7825/$ - see front matter © 2004 Published by Elsevier B.V.
doi:10.1016/j.cma.2004.05.014

the flow obeys Darcy's Law; the flow is two-dimensional and at steady state. Because of the assumptions made, the problem is described by the velocity potential function, φ, whose governing differential equation and boundary conditions are also shown in **Fig. 1**. The relevant dimensions are taken to be: $x_1 = 40$, $y_1 = 10$ and $y_2 = 3$. In **Fig.** 1, Ω is the seepage region abdf. The location of the curve *fd*, $y = \bar{f}(x)$, is unknown a priori.

A fixed domain formulation for this problem can be obtained by using the Baiocchi method and transformation (see [2,10,11,3,6]). In this approach the a priori unknown solution region is extended across the free surface into a known region. The dependent variable is also continuously, similarly extended. Then a new dependent variable is defined using Baiocchi's transformation within these regions. The resulting problem formulation leads to a 'complementarity system' associated with its respective variational inequality formulation. This method has proven effective not only from the purely theoretical point of view, but also from the point of view of yielding new, simple, and efficient numerical solution schemes.

Fig. 2 shows the governing equations and boundary conditions that describe the fixed domain formulation of the problem presented in **Fig.** 1. D is the region abef. The variable w is the Baiocchi transformation of the extended potential function, i.e.,

$$w(x,y) = \int_y^{y_1} [\tilde{\varphi}(x,\bar{\eta}) - \bar{\eta}]\,\mathrm{d}\bar{\eta}, \tag{1}$$

where

$$\begin{aligned} \tilde{\varphi}(x,y) &= \varphi(x,y) \quad \text{in } \overline{\Omega}, \\ \tilde{\varphi}(x,y) &= y \quad \text{in } \overline{D} - \overline{\Omega}. \end{aligned} \tag{2}$$

The detailed derivations of these equations are given in [3].

The problem shown in **Fig.** 2 can be written as a 'complementary system' and its corresponding variational inequality formulation. Then the following theorem can be stated:

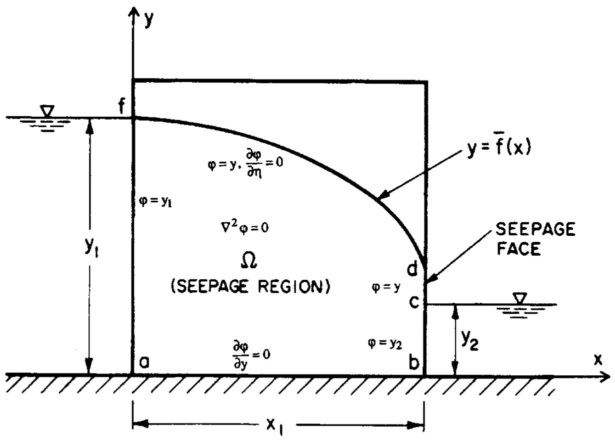


Fig. 1. The example physical problem (free boundary seepage).

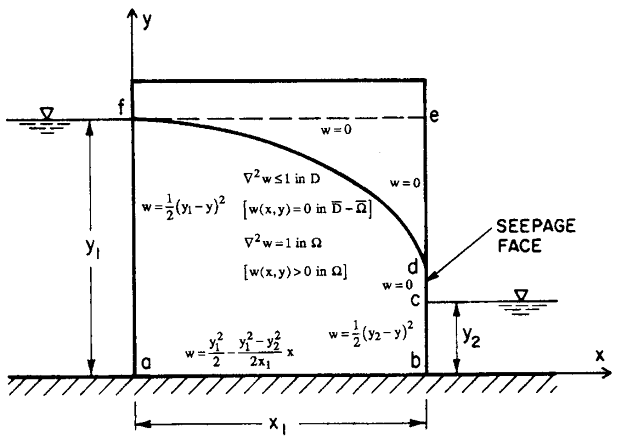


Fig. 2. The example physical problem (free boundary seepage) for numerical implementation.

Let $\bar{p}(x, y)$ be the Dirichlet data in **Fig.** 2 and define

$$K = \{v(x,y)\,|\, v \in H^1(D),\ v|_{\partial D} = \bar{p},\ v \geqslant 0 \text{ a.e. on } D\},$$

a closed convex set, $K \subset H^1(D)$.

Theorem 1. *If $w \in K$ satisfies the governing equations and boundary conditions shown in* **Fig.** 2, *then it also satisfies the variational inequality*:

$$a(w, v-w) \geqslant L(v-w) \quad \forall v \in K, \tag{3}$$

where

$$a(w, v-w) = \int\int_D \nabla w \cdot \nabla(v-w)\,\mathrm{d}x\,\mathrm{d}y = \int\int_D \{w_x(v_x - w_x) + w_y(v_y - w_y)\}\,\mathrm{d}x\,\mathrm{d}y \tag{4}$$

and

$$L(v-w) = -\int\int_D (v-w)\,\mathrm{d}x\,\mathrm{d}y. \tag{5}$$

The finding of $w \in K$ is equivalent to solving the minimization problem

$$J(w) \leqslant J(v) \quad \forall v \in K, \tag{6}$$

where

$$J(v) = a(v,v) + 2(f,v) \tag{7}$$

in which $a(v, v)$ is a bilinear form, continuous, symmetric, positive definite on R and $f \in R$, i.e.,

$$a(v,v) = \int\int_D \nabla v \cdot \nabla v\,\mathrm{d}x\,\mathrm{d}y, \tag{8}$$

$$(f,v) = \int\int_D fv\,\mathrm{d}x\,\mathrm{d}y. \tag{9}$$

For this example problem, $f = 1$. The functional J has one and only one minimum in a closed convex set.

The minimum is found using the following finite element algorithm:

$$u_i^{(n+1/2)} = -\frac{1}{a_{ii}}\left(\sum_{j=1}^{i-1} a_{ij}u_j^{(n+1)} + \sum_{j=i+1}^{N} a_{ij}u_j^{(n)} + f_i\right), \tag{10}$$

$$u_i^{(n+1)} = P_i\left(u_i^{(n)} + \omega\left(u_i^{(n+1/2)} - u_i^{(n)}\right)\right) = \max\left(0, u_i^{(n)} + \omega\left(u_i^{(n+1/2)} - u_i^{(n)}\right)\right), \tag{11}$$

where $a_{ij} = a(N_i, N_j)$, $f_i = (f, N_i)$, N_i is the canonical basis of R^N, P_i is the projection on the convex set, $i = 1, \ldots, N$, N is the number of nodal points and ω is the relaxation factor. The relaxation factor is determined empirically. Linear triangular elements will be used in the discretization. It should be noted that the projection operation in the numerical scheme must be applied during the iteration process. It cannot be applied after the iteration process has been completed since if it were, an incorrect solution would be obtained.

2. Adaptive mesh finite element analysis

Error estimation and local mesh refinement are two major concepts of adaptive mesh finite element analysis. In addition, a mesh refinement algorithm is required to perform remeshing after obtaining the error of a finite element system. The error estimate decides how the computed results deviate from the exact solution. Local mesh refinement testing is performed to determine how the mesh is to be refined. The remeshing algorithm is then used to automatically generate a refined mesh according to the error obtained.

The error estimation procedure used herein was introduced by Zienkiewicz and Zhu [18]. It allows an accurate assessment of errors while remaining so simple that it can readily be implemented as a post-processor involving minimal computation. This computationally simple error estimator with the modification introduced by Burkley and Bruch [4] and Burkley et al. [5] is used for the solution of the free surface flow through an earth dam using a parallel computer. The modification used by Burkley and Bruch [4] was that, instead of using the Zienkiewicz–Zhu procedure (a projection method) to calculate the nodal estimates for the exact nodal fluxes, they performed a simple averaging technique. That is, for each node they added up say the x-fluxes (which were constants since linear elements were used) from the elements that contained that node and divided that sum by the number of contributing elements. The same was done for the y-fluxes. A mesh generator and a mesh refiner were used to perform the finite element analysis and the post-processing of the mesh refinement. Isosceles right triangle elements were used for the purpose of this study. A simple mesh generator and refiner for these elements were implemented to generate the initial mesh. The concept behind the mesh generation and mesh refinement is simple: divide an element into two by refining across its longest side. Also, in this study, incompatible elements were not allowed. Therefore, a recursive process proposed by Rivara [12,13] was used to avoid having such elements.

2.1. Notation and error definition

The example problem addressed is a free surface flow through a porous medium. The error estimator will be discussed here using Eq. (12) as an example. Let the governing equation for the dependent variable $u(x, y)$ in solution domain R be

$$\nabla^2 u(x,y) = S^{\mathrm{T}}Su = f \quad \text{in } R, \tag{12}$$

where

$$S^{\mathrm{T}} = \begin{bmatrix} \frac{\partial}{\partial x} & \frac{\partial}{\partial y} \end{bmatrix} \tag{13}$$

and $u(x, y)$ is subject to appropriate boundary conditions (either Dirichlet data or zero Neumann data). The corresponding functional for the problem is

$$J(u) = \int_R \left\{ \frac{1}{2} \left[\left(\frac{\partial u}{\partial x} \right)^2 + \left(\frac{\partial u}{\partial y} \right)^2 \right] + uf \right\} \mathrm{d}R. \tag{14}$$

For clarity the following notion is defined:

$$q_x \equiv \frac{\partial u}{\partial x}, \quad q_y \equiv \frac{\partial u}{\partial y}; \tag{15}$$

$\wedge$ above a symbol refers to the approximate value of that symbol; $^-$ above a symbol refers to the nodal values of that symbol; the subscript i associates the term with a particular element; N are the shape functions—in this case linear triangles; and terms without any modifications represent the exact value.

In the finite element approximation using these definitions, values would be represented as

$$u \approx \hat{u} = N\bar{u}. \tag{16}$$

The linear system of equations used is derived by the Galerkin method using Eq. (12) or the Ritz method using Eq. (14) and is

$$Ku = F, \tag{17}$$

where

$$K = \int_R (SN)^{\mathrm{T}} (SN)\, \mathrm{d}R \tag{18}$$

and

$$F = \int_R fN^{\mathrm{T}}\, \mathrm{d}R. \tag{19}$$

The derivatives are approximated by

$$q \approx \hat{q} = S\hat{u} = SN\bar{u}. \tag{20}$$

Error is defined as the difference between the exact solution and the approximate solution

$$e_u = u - \hat{u} \tag{21}$$

and

$$e_q = q - \hat{q}. \tag{22}$$

Zienkiewicz and Zhu [18] presented the argument for the use of an improved estimate of $\hat{q}$ which could be used as an approximation for q in Eq. (22). A summary of their argument applied to the case of linear triangles which we used herein is:

'From Eq. (17) a linear approximation for u is calculated, this is noted as $\hat{u}$. The variable q is defined as the derivative of u, thus $\hat{q}$, being the derivative of $\hat{u}$, must be constant in a given element. It then follows that $\hat{q}$ will be discontinuous across neighboring elements and a potentially poor approximation to q. To find better results for an approximation of q, nodal values ($\bar{q}$) are calculated so that a $\tilde{q}$ which is interpolated by the same shape function as $\hat{u}$ can be found. $\tilde{q}$ will vary linearly across an element and will be a better approximation to q.'

Eq. (22) then becomes

$$e_q \approx \tilde{q} - \hat{q} = \hat{e}_q \tag{23}$$

with $\tilde{q} = N\bar{q}$. An averaging method is used for calculating the values of $\bar{q}$.

2.2. Averaging technique

As the name implies, the averaging technique used solves for the value of the derivative, $\bar{q}$, at a node by averaging the values of the derivatives of all elements which contain that node. The assumption is that the value of the node will be most heavily influenced by neighboring elements, thus simplifying the computations by making it only dependent on the neighboring elements. The technique is simple to implement and computationally inexpensive.

2.3. Error norm

A convenient form for expressing the error is the L_2 norm. It can be used with the errors predicted in any calculation. Its general form is

$$\|e\|_{L_2} = \left[\int_R e^{\mathrm{T}} e\, \mathrm{d}R\right]^{1/2}. \tag{24}$$

For the specific example of the e_q error, Eq. (22), it becomes

$$\|e_q\|_{L_2} = \left[\int_R (e_q)^{\mathrm{T}}(e_q)\, \mathrm{d}R\right]^{1/2}. \tag{25}$$

Thus, for the example presented, the L_2 norm and the energy norm are the same.

The above norm is defined over the entire domain R. It can also be specified by the sum of the squares over all the elements.

$$\|e\|^2 = \sum_{i=1}^{N_e} \|e\|_i^2, \tag{26}$$

where the subscript i represents the element number and N_e is the total number of elements. It is optimal to have the elemental error norms equal for all elements [1,7].

For this case

$$\|e_q\|_i^2 = \int_{A_i} \left[(e_{q_x})^{\mathrm{T}}(e_{q_x}) + (e_{q_y})^{\mathrm{T}}(e_{q_y})\right] \mathrm{d}A, \tag{27}$$

where the i subscript denotes a particular element with area A_i which is approximated by

$$\|\hat{e}_q\|_i^2 = \int_{A_i} \left[(\hat{q}_x - N\bar{q}_x)^2 + (\hat{q}_y - N\bar{q}_y)^2\right] \mathrm{d}A. \tag{28}$$

2.4. Error measures

An easily examined value for error is the relative percentage error η defined as

$$\eta = \frac{\|e_q\|}{\|q\|} \times 100\%, \tag{29}$$

where

$$\|q\| = \left[\int_R q^2\,\mathrm{d}R\right]^{1/2}. \tag{30}$$

η can be defined on both the local and global levels. The importance being that with a known percentage error there is the possibility to refine on the elemental level. The above formula is the exact value of η. The predicted value of η, denoted as ${}^0\eta$, is

$${}^0\eta = \left[\frac{\|\hat{e}_q\|^2}{\|\tilde{q}\|^2}\right]^{1/2}, \tag{31}$$

where

$$\|q\|^2 \approx \|\tilde{q}\|^2 = \int_R (\tilde{q}_x^2 + \tilde{q}_y^2)\,\mathrm{d}R = \int_R \left[(N\bar{q}_x)^2 + (N\bar{q}_y)^2\right]\mathrm{d}R. \tag{32}$$

Finally, an index to measure the effectiveness of the error estimate is defined by

$$\theta = \frac{\text{predicted error}}{\text{actual error}} = \frac{\|\hat{e}_q\|}{\|e_q\|}. \tag{33}$$

While θ is a worthwhile quantity, since relative error is a more relevant quantity than absolute error, an index for measuring performance of relative error predictions would be useful. This will be defined by

$$\beta = \frac{{}^0\eta}{\eta}. \tag{34}$$

An empirical correction factor has been found for θ [18]. It depends upon the elements being used. For the purpose of this paper, no correction factors were used in the example.

2.5. Mesh refinement

The refinement procedure herein uses the desired tolerance for error and compares it to the predicted value for error. As described in the previous section, it is desirable to be allowed to specify the error tolerance in a percentage (relative) form. Further, it should be possible to specify both the overall maximum percentage error as well as a local maximum percentage error. This is useful because the global error could meet the specified tolerance while the local error in a few elements could be dramatically greater than desired. The condition thus specified is then

$$\eta \leqslant \eta_{\max}, \tag{35}$$

where $\eta_{\max}$ is the specified error tolerance.

Assuming that the error is equally distributed between elements, the requirement, Eq. (35), can be translated into calculation of a maximum absolute error. For a mesh having N_e elements satisfaction of Eq. (35) requires

$$\|\hat{e}_q\|_i \leqslant \eta_{\max}\left[\frac{\|\tilde{q}\|^2}{N_e}\right]^{1/2} = e_{\max}, \tag{36}$$

where $e_{\max}$ is the maximum allowable elemental error.

Criteria for refinement are calculated by comparison of elemental error versus $e_{\max}$. Defining

$$\xi_i = \frac{\|\hat{e}_q\|_i}{e_{\max}} \tag{37}$$

gives the criteria. Values of $\xi_i < 1$ indicate that larger elements can be adopted, while $\xi_i > 1$ requires smaller elements. Denoting A_i as the current element area size and the refinement follows

$$(A_i)_{\text{new}} = \frac{A_i}{\xi_i} \quad \text{for } \xi_i > 1 \tag{38}$$

giving the predicted element area size. Since the domain being modeled is a rectangle, the elements used are isosceles right triangles. The starting mesh for the problem considered is uniform.

Refinement consists of targeting an element for refinement, then dividing the element in two by refining across its longest side (remember isosceles right triangles are being used). Because it is desirable to keep the elements in an optimum shape (see [14]) if the common nodes between the element being refined and its neighbor do not define the longest side of the neighbor, the neighbor must be refined first. This sets up the possibility of a recursive process. This strategy causes all elements to be similar triangles.

3. Parallel iterative scheme

Wang and Bruch [16] proposed parallel iterative Gauss–Seidel and SOR iterative schemes. Conventional Gauss–Seidel and SOR iteration schemes need to be performed sequentially. Reordering the equations alters these two schemes into fully parallel iterative schemes. Wang and Bruch [16] used this intuitive idea and implemented it on a free boundary seepage problem. Speed-ups were obtained that were superlinear (speed-up larger than the number of processors used).

The basic essence of the approach is as follows: after the computation domain is subdivided into subdomains (see [16] for the procedure for this load balancing), the problem domain boundary remains a boundary and the interfaces of a subdomain become new boundaries. Thus, the computation of values at interior mesh points for one subdomain is uncoupled from the other subdomains. Also, the computation of values at mesh points of an interface is uncoupled from the other interfaces. The iterative schemes use a combination of newly computed values and old values at mesh points surrounding a mesh point to compute the new value at that mesh point. Therefore, the iterative process can be performed for the interior mesh points of a subdomain using the old values at interface mesh points. Moreover, the values at interface mesh points can be updated using the newly computed values at interior mesh points by the iterative process. An example and explanation are given in [16,17].

Accordingly, this parallel iterative scheme simply reorders the computing sequence such that the values at interior mesh points are computed first, then those at the interface mesh points are computed. With this parallel scheme, all processors can compute concurrently for the new values at the interior mesh points. Also, all processors update the values at mesh points on the interface in parallel.

4. Performance tools and considerations

The VAMPIR [19] and PARAVER [20] performance analysis software are used to analyze and understand the execution behavior of the finite element parallel 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 [19]) is a post-mortem trace visualization tool from Pallas GmbH. It uses the profiling extensions to MPI (Message Passing Interface) 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 seepage test example, all tracing was done on the IMB SP2 Blue Horizon, using a 4-node run. Selection of "Global Displays/Activity Chart," pops up a window that consists of pie charts for every process [4 nodes]. Load imbalances can be identified by this view.

PARAVER (Parallel Program Visualization and Analysis Tool [20]) 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 invest the programming effort to optimize an application. The result is a reduction of the development time as well as the minimization for the hardware resources required for it.

5. Running the test case

To run the adaptive mesh finite element program, a user must generate the initial mesh first, then run the finite element program with an error estimate. With the error estimate, a new refined mesh will be created for another finite element analysis. This process is repeated until the error from the finite element analysis is satisfied. The program GEN is used to create the initial mesh and perform mesh refinement. The GEN also performs the domain decomposition and generates data for the parallel finite element analysis. The program FEA is a parallel finite element analysis program that performs finite element analysis in parallel.

The physical problem considered for the test case is shown in **Fig.** 2. A relaxation factor of 1.85, a stopping error criterion of 10^{-4} for the maximum absolute difference between iterates at a mesh point and $\eta_{\max} = 0.05$ were used in obtaining the numerical results.

Every stage is run in two steps:

- mesh generation, by running the GEN module;
- running the parallel module FEA, using the mesh generated by GEN.

The mesh map for Stage 1 is presented in **Fig.** 3 for processor 0 and in **Fig.** 4 for processor 3 using 32 elements on each processor.

The speed-up on the Blue Horizon for the first adaptive mesh refinement solution (designated as Stage 1) is shown in **Table** 1 resulting in a super-scalar speed-up, due to the cache effect.

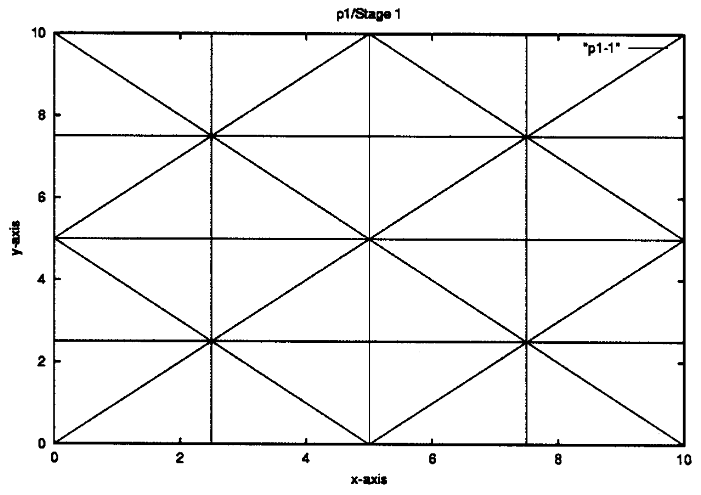


Fig. 3. Stage 1—Processor 0—Mesh Map.

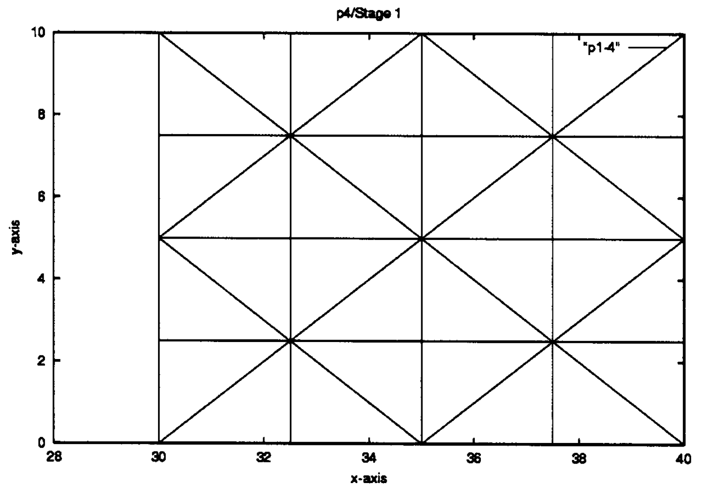


Fig. 4. Stage 1—Processor 3—Mesh Map.

Table 1
Ideal and FEA speed-up

# of processors	Ideal speed-up	FEA speed-up	(Time/Proc.)
1	1.0	1.00	0.8702
2	2.0	2.04	0.4250
4	4.0	4.30	0.2023
8	8.0	8.27	0.1051
16	16.0	18.79	0.0463

VAMPIR [19] and PARAVER [20] were used to find and correct performance anomalies and inefficiencies in the FEA program. Both utilities can analyze and offer specific guidance for problems such as:

- load imbalance
- excessive communication
- excessive serialization
- poor use of memory hierarchy

The Activity Chart of Stage 1 obtained with the VAMPIR utility is presented in Fig. 5 and is confirming the good performance of the FEA code Stage 1 of computation. The green areas refer to the amount of computer operations not including message passing, while the red areas refer to the amount of message passing. The Zoom Display of Stage 1 obtained with the PARAVER utility is presented in Fig. 6 and confirms the results of the VAMPIR utility. Here again the green designations refer to computer operations not including message passing, while the yellow refer to message passing.

Starting with Stage 3 we can detect a load imbalance, which is more evident in Stage 4, presented in Fig. 7. We "zoom" into the part which shows the load imbalance, presented in Fig. 8. From Figs. 7

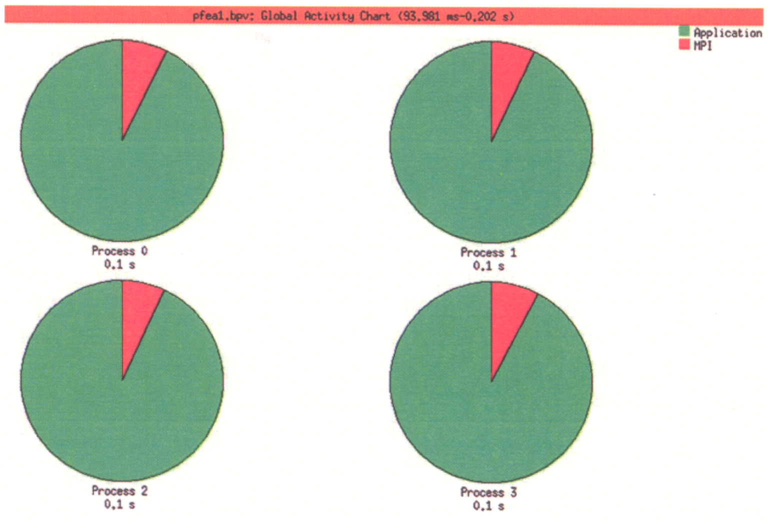


Fig. 5. Stage 1—VAMPIR—Activity Chart.

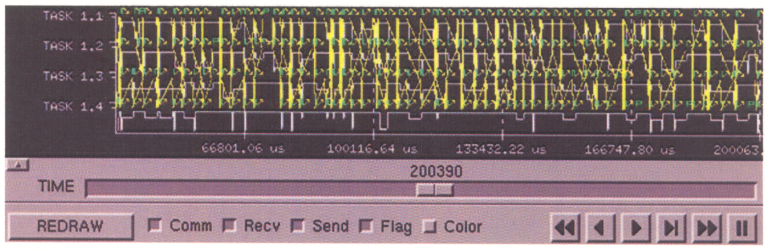


Fig. 6. Stage 1—PARAVER—Zoom Display.

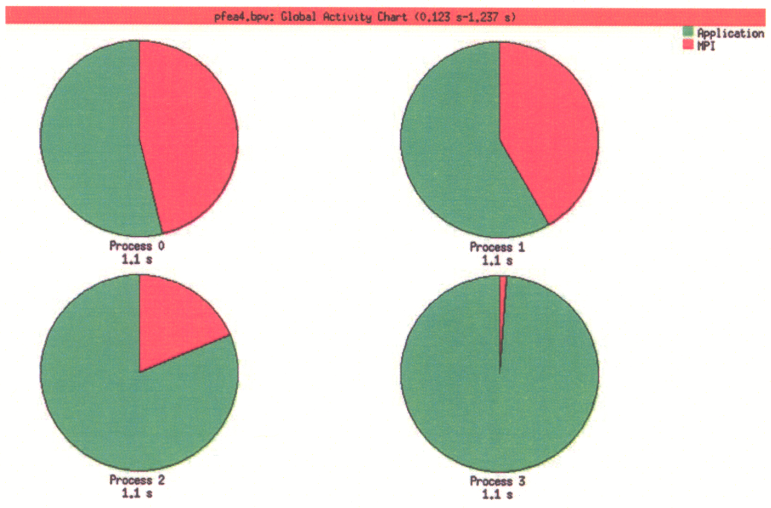
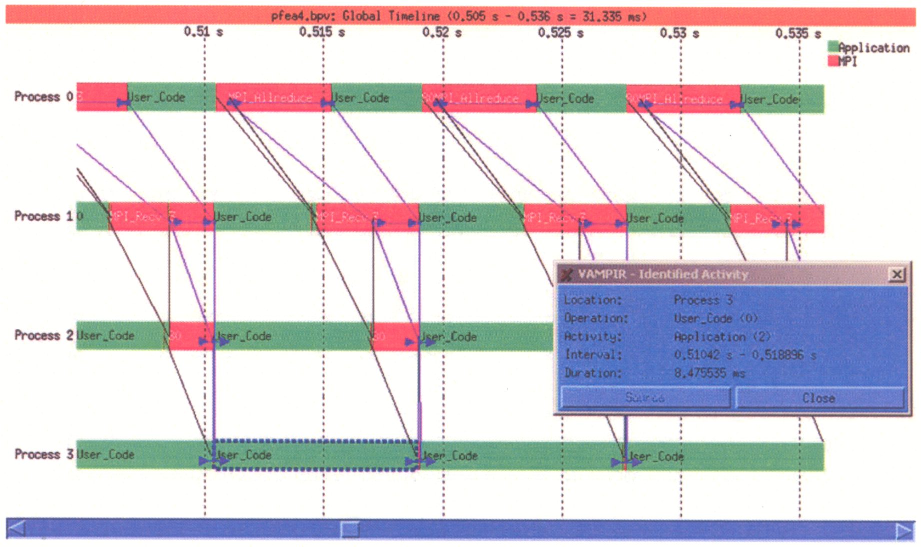


Fig. 7. Stage 4—VAMPIR—Activity Chart.



Location: Process 3
Operation: User_Code (0)
Activity: Application (2)
Interval: 0.51042 s - 0.518896 s
Duration: 8.475535 ms

Fig. 8. Stage 4—VAMPIR—Zoom Display.

and 8 we conclude that processor 0 has the least amount of computation, i.e., the least amount of computer operations not including message passing, to perform. As the number of mesh points and the amount of mathematical operations is equal for all processors, we suspect a poor use of memory hierarchy in the SOR routine.

We use the “HPMCOUNT” performance utility [8] in order to find quantitative information regarding the resource usages. In Table 2 we see important differences between Stage 1 and Stage 4 for the TLB misses.

Table 2
TLB misses

Stages	Proc. 1	Proc. 4
1	9464	7870
4	12,210	208,341

The Blue Horizon, as well as all modern virtual memory machines, has a special cache called a "translation lookaside buffer" or TLB for virtual to physical memory address translation. Like other kinds of caches, the TLB is limited in size. It does not contain enough entries to handle all of the possible virtual to physical address translations for all the programs that might run on the machine. Larger pools of address translations are kept out in memory, in the page tables. If the program asks for a virtual to physical address translation, and the entry does not exist in the TLB, then there is a "TLB miss". A new page will have to be created in memory and possibly, depending on the circumstances, refilled from disk. Although they take a lot of time, page faults are not errors. Even under optimal conditions every program will suffer some number of page faults [9].

Mesh refining—from 32 elements/processor in Stage 1 to 1262 elements/processor in Stage 4—created adjacent points with non-adjacent memory addresses, the main reason for large TLB misses. **Figs. 9 and 10** present the mesh maps in Stage 4 for Processor 0 and Processor 3, respectively.

In order to correct the load imbalance due to the TLB misses, we decrease/increase the number of points/processor proportional to the computational time on different processors in the SOR module—**Table 3**.

Analyzing the results again with VAMPIR and PARAVER we obtain the Activity Chart presented in **Fig.** 11, the Zoom Display in **Fig.** 12 and the Zoom Display (which shows the message passing information with the other computer operations filtered out) in **Fig.** 13. We actually did not correct the TLB misses, but did a more correct load balance. We "zoom" into a part which shows the load balance in Stage 4, presented in **Figs.** 12 **and** 13. One way to correct the large number of TLB misses, would be to renumber the elements and mesh points at each Stage so as to have a different memory access pattern in order to optimize the memory references [15].

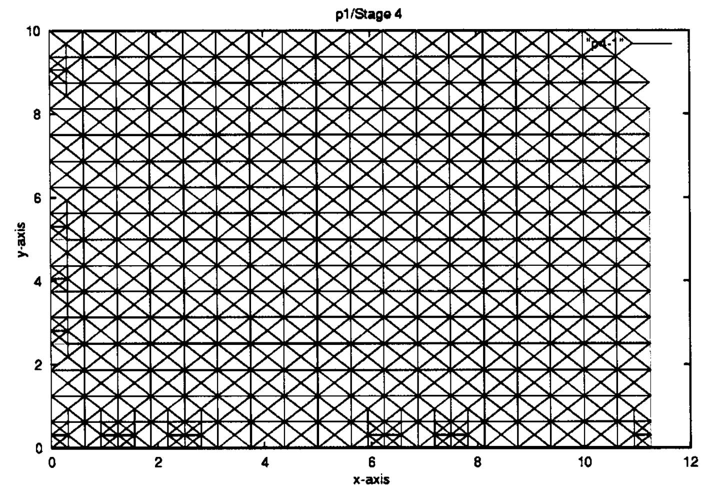


Fig. 9. Stage 4—Processor 0—Mesh Map.

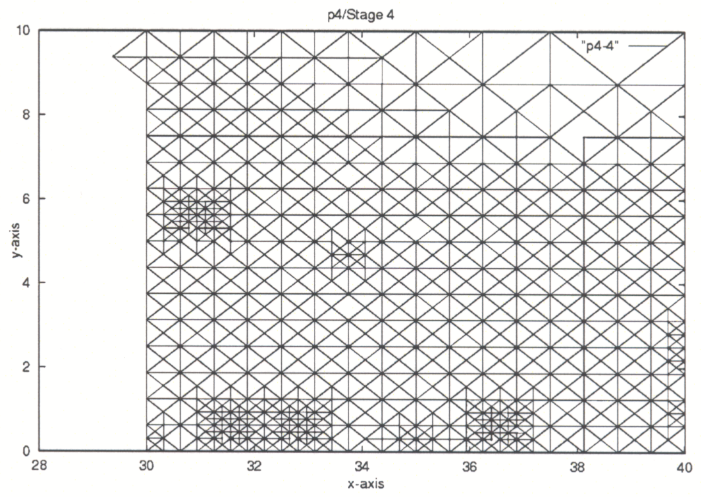


Fig. 10. Stage 4—Processor 3—Mesh Map.

Table 3
Stage 4 timing of the SOR module

Processor	Time spent in SOR
0	0.3671
1	0.4068
2	0.6940
3	0.8393

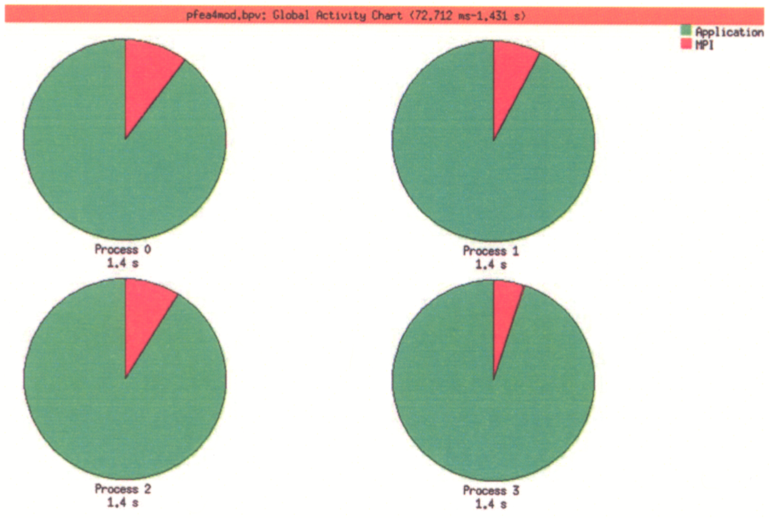


Fig. 11. Stage 4—VAMPIR—Activity Chart.

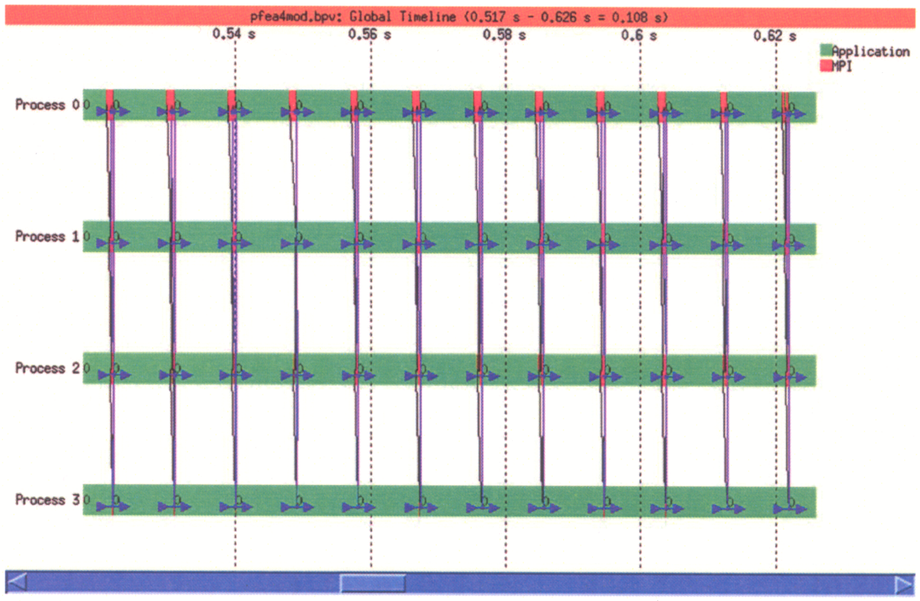


Fig. 12. Stage 4—VAMPIR—Zoom Display.

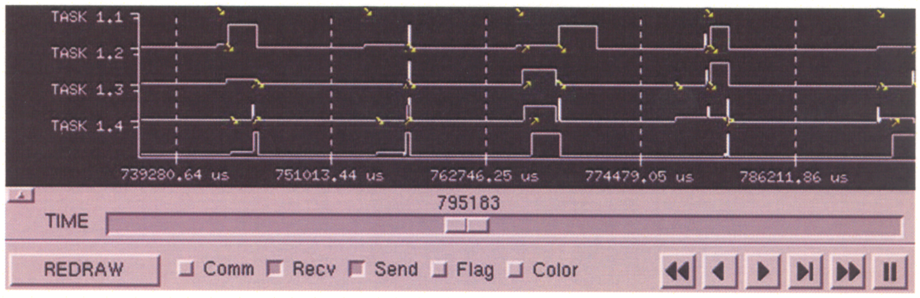


Fig. 13. Stage 4—PARAVER—Zoom Display.

6. Conclusions

A significant factor that affects the performance of a parallel application is the balance between communication and workload. To fully understand the performance behavior of such applications, analysis and visualization tools are needed. Two such tools, VAMPIR and PARAVER, were used to analyze the performance of the seepage 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.

Acknowledgment

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 algorithm used to solve the free boundary problem is based upon work supported by the National Science Foundation Grant # ECS-90006516. This research was supported in part by NSF cooperative agreement ACI-9619020 through computing resources provided by the National Partnership for Advanced Computational Infrastructure at the San Diego Supercomputer Center.

References

[1] I. Babushka, W.C. Rheinboldt, Analysis of optimal finite element method in R^1, Math. Comput. 33 (1979) 435–463.
[2] C. Baiocchi, A. Capelo, Variational and Quasivariational Inequalities, John Wiley, New York, 1984.
[3] J.C. Bruch Jr., A survey of free boundary value problems in the theory of fluid flow through porous media: variational inequality approach, Adv. Water Resour. Part I 3 (1980) 65–80, Part II 3 (1980) 115–124.
[4] V.J. Burkley, J.C. Bruch Jr., Adaptive error analysis in seepage problems, Int. J. Numer. Methods Engrg. 31 (1991) 1333–1356.
[5] V.J. Burkley, J.C. Bruch Jr., O.C. Zienkiewicz, Adaptive meshes used in solving a free surface seepage problem, in: J.R. Whiteman (Ed.), The Mathematics of Finite Elements and Applications, VII, Academic Press, 1991, pp. 101–110.
[6] J. Crank, Free and Moving Boundary Problems, Clarendon Press, Oxford, UK, 1984.
[7] E.R. de Arantes, E. Oliveira, Optimization of finite element solutions, in: Proceedings of the Third Conference on Matrix Methods in Structural Analysis, Wright-Patterson Air Force Base, Ohio, 1971.
[8] L. DeRose, Hardware Performance Monitor (HPM) Toolkit, Version 2.4.3, Advanced Computing Technology Center, IBM Research, 2003.
[9] C. Dougan, P. Mackerras, V. Yodaiken, Optimizing the idle tasks and other MMU tricks, in: Proceedings of the Third Symposium on Operating Systems Design and Implementation, Department of Computer Science, New Mexico Institute of Technology, 1999, pp. 229–238.
[10] D. Kinderlehrer, G. Stampacchia, An Introduction to Variational Inequalities and Their Applications, Academic Press, New York, 1980.
[11] J.T. Oden, N. Kikuchi, Theory of variational inequalities with applications to problems of flow through porous media, Int. J. Engrg. Sci. 18 (1980) 1173–1284.
[12] M.C. Rivara, Algorithms for refining triangular grids suitable for adaptive and multigrid techniques, Int. J. Numer. Methods Engrg. 20 (1984) 745–756.
[13] M.C. Rivara, A grid generator based on 4-triangles conforming mesh-refinement algorithms, Int. J. Numer. Methods Engrg. 24 (1987) 1343–1354.
[14] L.J. Segerlind, Applied Finite Element Analysis, John Wiley and Sons, New York, 1984.
[15] C. Severance, K. Dowd, High Performance Computing, second ed., O'Reilly & Associates, Inc., 1998, 464 pages.
[16] K.P. Wang, J.C. Bruch Jr., A SOR iterative algorithm for the finite difference and the finite element methods that is efficient and parallelizable, Advances in Engineering Software 21 (1) (1994) 37–48.
[17] K.P. Wang, J.C. Bruch Jr., An efficient iterative parallel finite element computational method, in: J.R. Whiteman (Ed.), The Mathematics of Finite Elements and Applications, John Wiley and Sons, Inc., 1994, pp. 179–188, Chapter 12.
[18] O.C. Zienkiewicz, J.A. Zhu, A simple error estimator and adaptive procedure for practical engineering analysis, Int. J. Numer. Methods Engrg. 24 (1987) 337–357.
[19] VAMPIR, Visualization and Analysis of MPI Resources, VAMPIR 2.0. User's Manual, Pallas GmbH, http://www.pallas.de.
[20] PARAVER, Parallel Program Visualization and Analysis Tool, Reference Manual, http://www.cepba.upc.es.