UNCLASSIFIED

Defense Technical Information Center
Compilation Part Notice

ADP023867

TITLE: Optimizing Finite Element Programs on the Cray X1 Using Coloring Schemes

DISTRIBUTION: Approved for public release, distribution unlimited

This paper is part of the following report:

To order the complete compilation report, use: ADA492363

The component part is provided here to allow users access to individually authored sections of proceedings, annals, symposia, etc. However, the component should be considered within the context of the overall compilation report and not as a stand-alone technical report.

The following component part numbers comprise the compilation report:
ADP023820 thru ADP023869

UNCLASSIFIED
Abstract

Using the Environmental Quality Modeling program FEMWATER as a test-bed code, 27 percent of the time needed to run a given groundwater flow application on the ERDC Cray X1 using four multistream processors (MSPs) was spent assembling the global stiffness matrix. This poor performance is because the above code cannot multistream without help. The technique of "coloring" the elements makes it possible to multistream this section of the code, thus taking advantage of the hardware capability of the machine. Coloring for assembling the global stiffness matrix involves dividing the elements into different groups such that no node point touches any elements with the same color. This paper will present a simple coloring algorithm in FORTRAN and show how it was implemented into FEMWATER to achieve multistreaming on the ERDC Cray X1. It will then give a detailed description on how the program was modified, what compiler options were used, and what compiler directives worked best. Finally, timing results will be given. Some programs that have good MPI (or equivalent) communication are better suited for running in the single-streamed processor (SSP) mode. In the SSP mode, coloring of the elements is not needed for assembling the global stiffness matrix. Timings for running in the SSP mode will be shown, too.

1. Introduction

Vectorizing and multistreaming application codes on the Cray X1 is essential to good performance. Because of this, bottlenecks in code performance can occur in surprising places. Sometimes the algorithm is just not suited for the X1. Many times, however, special techniques and algorithms can be applied to remedy the situation. This paper illustrates one such example of where "coloring" the finite elements into separate groups can significantly help. Assembling the element stiffness matrices into the global stiffness matrix originally took 27 percent of the total run time. This part of the program now runs eight times faster when using the coloring scheme described in this paper.

2. FEMWATER

FEMWATER\(^{4,5}\) is a standard Galerkin finite element program for flow and transport. METIS\(^2\) was used to partition the mesh. A conjugate gradient, iterative solver with an incomplete lower-upper preconditioner\(^1\) was used to solve the resulting system of linear equations obtained from a Picard iteration of the nonlinear equations. The ghost node updates are done in this study using MPI.

2.1. Test Problem.

Figure 1 illustrates the top view of a typical three-dimensional (3-D) finite element mesh for a remediation study. Several layers are used to model the soil layers underneath this surface. The mesh has 102,996 nodes and 187,902 3-D prism elements. Runs using two and four times the original number of elements are also done.

![Figure 1. FEMWATER test problem](image-url)
2.2. Assembly of Global Stiffness Matrix.

Figure 2 shows the original assembly process. Subroutine fg468 computes element stiffness matrix information. The i and j loops are from 1 to 6, representing prism elements. ni is the node or row number in the global stiffness matrix, and jj is the column number in the global stiffness matrix. These indices were reversed in storage to have stride 1 computations. Finally, a search for jj must be done inside the j loop. Figure 3 shows how coloring modifies the basic routine. This modified version is described in more detail below.

3. Coloring

Coloring for assembling the global stiffness matrix involves dividing the elements into different groups such that no node point touches any elements with the same color. Figure 4 illustrates the process of generating the different groups through coloring. The algorithm presented in this paper written in FORTRAN (see Figure 5) is very similar to that discussed in Reference 3. Starting with color 1 (blue), an element is first painted. Then all elements that touch this element are marked for not allowing blue. A second blue element is done, again marking elements touching the new blue element as not allowing blue. After all the possible blue elements have been painted, this procedure is repeated for the second color (red). This algorithm is completed when all elements have been provided a color. Finally, all elements that contain the same color belong to the same group.

The code in Figure 2 can now be changed to the code in Figure 3. The do loop over mm can now be multistreamed and vectorized. This will now be described.
4. Multistreaming and Vectorizing

Space allows a brief description of what was then done to achieve vectorization and multistreaming of the code. Figure 6 shows a portion of the new algorithm with compiler notation provided. This listing was generated using the command,

```
103. do ig = 1, nog
104. mm1 = istg(ig)
105. mm2 = istg(ig + 1) - 1
108. !csd$ parallel do private
109. (mm, m, node, mtyp, iq, iem, ni, jq, nj)
115. do mm = mm1, mm2
117. m = ielg(mm)
118. NODE = IJNOD(M)
120. M = MTYP = IE(M, 9)
121. ALP = PROPF(7, MTYP)
122. POR = PROPF(8, MTYP)
124. DO IQ = 1, NODE
125. DO IQ
126. END DO
```

Intermediate computations.

```
276. !dir$ concurrent
277. DO IQ = 1, NODE
278. NI = IEM(IQ)
279. MV = RLD(NI) = RLD(NI) + RQ(IQ)
280. DO JQ = 1, NODE
281. NJ = IEM(JQ)
282. MV = QA(IQ, JQ) \* DELT
283. MV = RLD(NJ) = RLD(NI) + (QA(IQ, JQ) \* W2 \* QB(IQ, JQ)) \* HP(NJ)
284. MV = cmatrx(iw(iq, jq, m), ni) + qa(iq, jq) + * qb(iq, jq)
285. MV = cmatrx(iw(iq, jq, m), ni)
300. MV = qa(iq, jq) + * qb(iq, jq)
301. END DO
302. END DO
```

Figure 6. Assembly process showing compiler notations

**Figure 5. Coloring algorithm in FORTRAN**
5. Performance Results

Table 1 shows timing results for the original mesh, twice the size of the original mesh, eight times the size of the original mesh, and sixteen times the size of the original mesh. For each problem, the modified MSP version of the assembly process was sped up approximately eight times. The problems were also run using 16 SSPs. Despite the good MPI communication, the modified MSP version runs four times faster than the original SSP version. Coloring of elements is not needed in the SSP mode, because this is for multistreaming only.

| Nodes     | 102,996 | 197,409 | 763,887 | 1,519,191 |
| Elements  | 187,902 | 375,804 | 1,503,216 | 3,006,432 |
| Mesh size increase | 1X | 2X | 8X | 16X |
| Total time - MSP - original | 745 | 1,419 | 5,444 | 11,171 |
| Total time - SSP - original | 257 | 493 | 1,958 | 3,636 |
| Assembly time - MSP - original | 187 | 379 | 1,458 | 2,733 |
| Assembly time - MSP - modified | 23 | 46 | 176 | 330 |
| Assembly MSP Ratio | 8.1 | 8.2 | 8.3 | 8.3 |
| Assembly time - SSP - original | 81 | 158 | 604 | 1,165 |

Acknowledgment

This work was supported in part by a grant of computer time from the DoD High Performance Computing Modernization Program at the ERDC MSRC, Information Technology Laboratory, Vicksburg, MS.

References
