Stanford Exploration Project, Report 70, December 18, 1997, pages 1–65

Stanford Exploration Project, Report 70, December 18, 1997, pages 1–65

Short Note

Experience with CM Fortran

Francis Muir[1]

keywords: *development, production, velocity, anisotropy, linear*

INTRODUCTION

I had originally intended to call this Short Note "Experience with the Connection Machine", but on reflection it is the Fortran interface that I know, not the machine itself, and this is how it should be. I am in no sense a system programmer, but I have many years experience developing algorithms and translating them into Fortran subroutines, so it is with this background that I discuss my chosen topic.

THE CONNECTION MACHINE DISCONNECTED

The Front End

The CM-2 has no life of its own, but works in symbiosis with a front end, in our case a SUN-4 named `oas`. Since the front-end does all the scalar arithmetic, the model should be chosen with this in mind. While the CM-2 is a single task (per partition) machine, there is no such restriction on the front-end. I had never worked with a SUN before coming to the Connection Machine, so I had some learning to do on the X interface. For those not familiar, X-Windows is powerful but rather klugie in operation when compared, say with NeXTview or the MAC interface. I find myself using two features. While oas is grinding away at compilation and linking, I can be doing other work on hanauma in another window. Also, the cut-and-paste facilities in X are very useful, if a bit substandard in operation.

The Frame Buffer

Our CM-2 is in no sense a production machine. There is no way, for example, to get data in and out of the device in a hurry. Also, it is quite small in size (two 4K

[1]**email:** not available

partitions), and has almost antedeluvian memory (256Kbit DRAM). Nevertheless it is almost an ideal configuration for algorithmic and code development. One reason for this is the frame buffer. This is a means of transferring data on the fly from the core of the machine to a graphics monitor with a simple subroutine call. If, for example, this call is in a loop over time, and if the image represents some field variable, then the graphics monitor will present a movie of the field as it changes from moment to moment. The frame buffer turns the CM-2 into a very interactive machine that I find invaluable for code development. For example, I have had up to twenty turn-arounds in a single session.

The Fortran Language

CM Fortran is a combination of Fortran 90 (more or less) and some CM-specific pre-compiler commands. One of the first things that I noticed about the Fortran 90 language was the possibility for compact code, which would probably lead to fewer coding errors. As an example from my own code, the Fortran 77 loop structure which I would otherwise need:

```
      DO j1 = 1,n1
        DO j2 = 1,n2
          DO j3 = 1,n3
            DO j4 = 1,n4
              DO k = 1,24
                huygens(k,j4,j3,j2,j1) = 0.0
              END DO
            END DO
          END DO
        END DO
      END DO
```

is replaced by the single line statement:

```
      huygens = 0.0
```

There is more to Fortran 90 than this, but it indicates some of the flavor.

The pre-compiler directives that are specific to the Connection Machine are much more interesting. Of them all, the most powerful is the `LAYOUT` directive which for my huygens array looks like this as it is embedded in some declarative code:

```
      INCLUDE 'param.inc'! which inputs the values of n1,n2,n3,n4
      REAL, ARRAY (24,n4,n3.n2,n1) :: huygens
CMF$  LAYOUT huygens (:SERIAL,:NEWS,:NEWS,:NEWS,:NEWS)
```

What the `LAYOUT` line does is to define, via `NEWS` (North,East,West, South), the dimensionality and size of a virtual parallel machine tailored to the problem. There is a virtual processing unit at each of the `n1*n2*n3*n4` nodes on this machine, and a 24-vector attached to each node. As defined, the 4-space is a closed space. That is, there are circular boundaries to the four `NEWS` dimensions. A small restriction on the `NEWS` dimensions is that they are limited to powers of 2. I find the circular boundaries more of a help than a hindrance. My lattice Boltzmann code occupies about a quarter of the space of my old Fortran 77 code, and took about a tenth of the time to write, and the seemless space was a major contributor to this saving in coding time and code length.

In addition to pre-compiler directives there are also some new functions such as `CSHIFT` (circular shift), which shifts a multi-dimensional array along one of the dimension directions by a stated amount. CSHIFT can be cascaded, which is useful in my linear lattice Boltzmann code where each of the elements of a 24-vector in a 4-space must be shifted to a different next-nearest neighbour.

Bugs

CM Fortran is a new product and it still has problems. Example. Two arrays which were aligned on the virtual machine in the main program became unaligned in subroutines which used them. This had a catastrophic effect on timing (worse than 100 times slower). While this and other problems are recognized by Thinking Machines, I suspect that the compiler will always be somewhat fragile, and will repay simple coding. Error tracebacks seem to be well handled, but it would be nice to see how the code is really mapped on to the machine in some simple, non-techie way. One of the attractions of the Convex fortran compiler is the way that it makes suggestions. I'd like to see the same sort of friendly intercourse available on the CM-2

THE LINEAR LATTICE BOLTZMANN PROBLEM

It has been about four years since I last worked on my linear lattice Boltzmann stuff, and the Connection Machine seemed to offer me good reason to start up again. I now have a better understanding of anisotropy, and it became clear to me that I should be able to extend previous work in this area by myself (Muir, 1986, 1087a, 1987b) ind others (Sword, 1987, and Zhang, 1988) in three directions.

1. Introduce heterogeneity both in velocity and impedance.

2. Develop large-scale anisotropy from micro-heterogeneous isotropy.

3. Extend the scalar, acoustic model to a vector, elastic one.

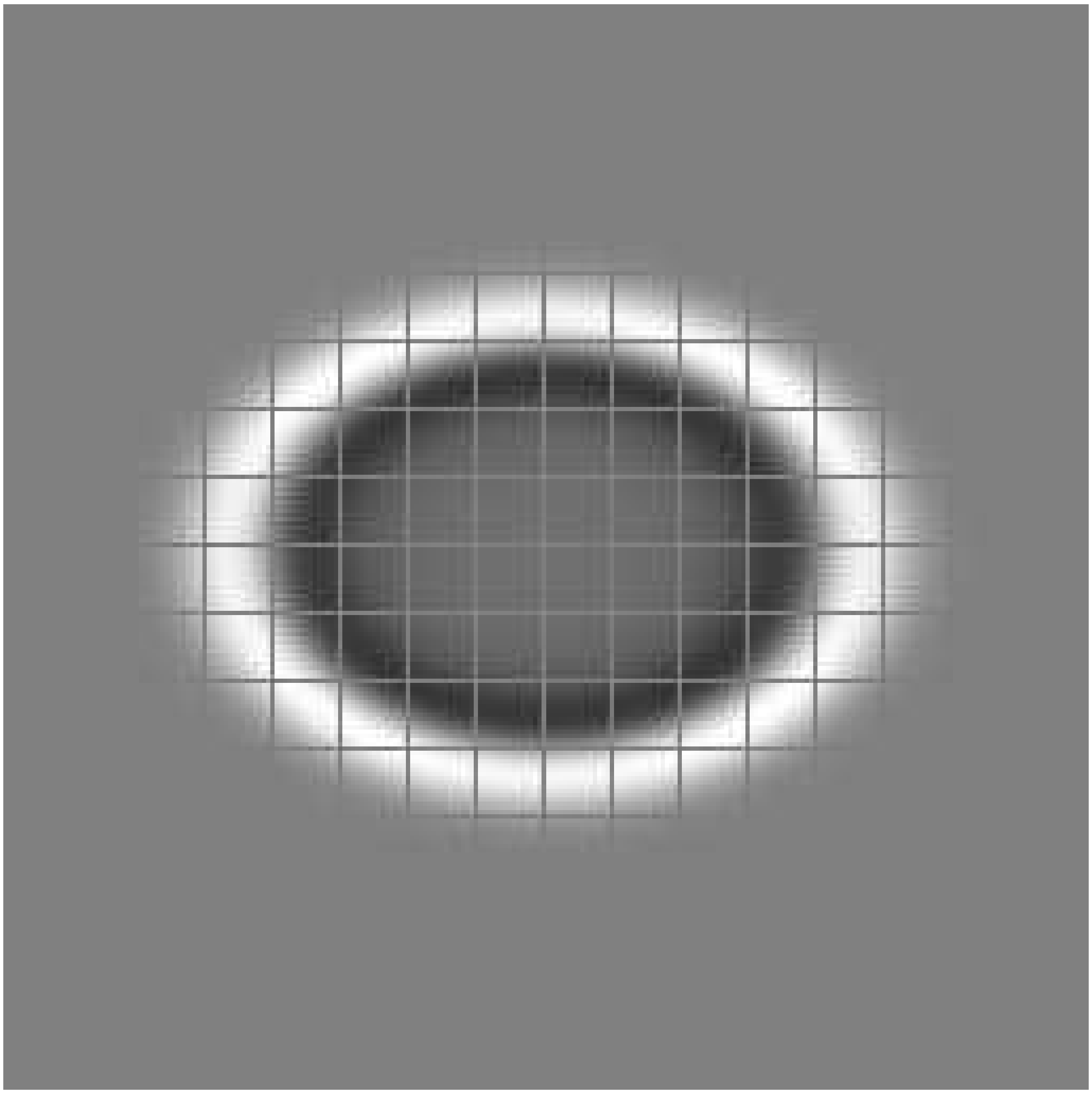

Figure 1: An elliptic impulse from a layered, isotropic medium $\boxed{\text{francis3-elliptic}}$ [NR]

So far the full elastic automaton has eluded me, but velocity anisotropy is quite well understood, and an example forms the subject of my illustration.

Recall from previous reports, and particularly (Muir, 1987b) that at each timestep I do three things:

1. Map the data field from conservation to Huygens coordinates.
2. Scatter the 24 elements of the Huygens vectors to their respective next-nearest neighbours.
3. Map the data field back from Huygens to conservation coordinates.

In Fortran 77 the first and third of these items would involve a nest of six DO loops. In CM Fortran they involve only two, and the first step looks like this:

```
      DO j = 1,24
        DO k = 1,5
          huygens(j,:,:,:,:) = huygens(j,:,:,:,:) +
     +      ctoh(k,j)*conserve(k,:,:,:,:)
        END DO
      END DO
```

The scatter loop is equally impressive:

```
      DO j = 1,24
        huygens(j,:,:,:,:) = CSHIFT(CSHIFT(
     +    huygens(j,:,:,:,:),d1(j),s1(j)),d2(j),s2(j))
      END DO
```

where the `d1(), d2(), s1(), s2()` arrays point in the direction and sign of the required shift.The scatter code I show above is vanilla code, good for a homogeneous environment, but the heterogeneous code uses very similar constructs, and there is no loss in parallelism because it is heterogeneous. As an example of what the Connection Machine and Frame Buffer Display can do, the Figure in this paper shows the impulse response of a layered medium where there is a 2:1 velocity contrast but no impedance change between the layers.

CONCLUSIONS

The CM Fortran compiler is an excellent but fragile piece of software that can produce efficient parallelized code from an intuitive set of constructs that are quite easily learned. It still has problems, and I think I did better than most by keeping the code very simple. The CM-2 machine we have is not suitable for any production work, but provides a fine development environment.

ACKNOWLEDGMENTS

As always, the graduate students on the fourth floor were patient at explaining things to me, and Biondo Biondi, once of SEP and now of Thinking Machines, was a tower of strength when the going got tough.

REFERENCES

Muir, F., 1986, An automaton for propagating waves through heterogeneous media: SEP–**50**, 79–84.

Muir, F., 1987a, Three experimental modeling systems: SEP–**51**, 119–128.

Muir, F., 1987b, M3 to M4: from Huygens to conservation variables: SEP–**56** 19–22.

Sword, C., 1987, Numerical dispersion and attenuation in the M3 modeling system: SEP–**56**, 1–18.

Zhang, L., 1988, Simulation of fluid dynamic processes with a probabilistic lattice gas: SEP–**57**, 259–278.