Parallel Computing for Space Surveillance

L. M. Healy, S. L. Coffey (Naval Research Laboratory)

Abstract

This paper reports on an application of massively parallel processors to multiple satellite propagation and the calculation of miss distances between objects (COMBO). Unlike serial computations, we do not pre-filter the data but rather sort the data set in a way that dramatically cuts the number of comparisons required in order to be assured of a complete catalog-to-catalog comparison. The same general algorithm allows two logical sets to be compared to each other.

Run time for this demonstration code on an 8K Connection Machine is about one second per time step, including propagation, complete catalog-to-catalog calculation of miss distances, plotting satellite positions, and recording of the miss distances to a file. Propagation of the objects is performed with an analytic propagator, using J_2 only at present, though the code may easily be extended to other propagators.

We demonstrate a second application of parallel computing to the problem of debris propagation resulting from a satellite breakup. The spread of such debris into n pieces is simulated by replicating the element set for the original satellite n times, then altering each to represent a distribution of velocities relative to the center of mass.

1 Introduction

One of the most important tasks performed by the central space surveillance processing sites, US Space Command (USSPACECOM) in Colorado Springs, Colorado and Naval Space Surveillance (NAVSPASUR) in Dahlgren, Virginia, is the maintenance of a catalog of elements on space objects. The magnitude of the tasks associated with the satellite catalog are of daunting proportions. The catalog consists of thousands of objects requiring the processing of tens of thousands of observations daily. Many of the tasks embody operations that are parallel in nature, e.g. orbit propagation, orbit determination, retagging observations and determining the miss distance between objects. The maturation of massively parallel computers like the Connection Machine (CM) presents a lot of potential for space surveillance. The availability of this computer at the Naval Research Laboratory prompted us to develop a program for performing the function COMBO (Calculation of Miss distance

Between Objects). The goals for this project were twofold; first develop the specific application thereby providing evidence of how well these computers can perform this function; second, to gain an appreciation of what role parallel computers can play for other space surveillance problems. We believe that COMBO is merely a small portion of the potential of parallel computing for space surveillance. We will discuss some prospective applications in the Section 4.

The catalog maintained at the central sites consists of functioning satellites as well as spent rocket bodies and fragments from exploded satellites. The size of the objects ranges from space stations like Soyuz and SKYLAB, before it decayed, to small pieces of satellites only a few centimeters across. The catalog currently consists of more than 7000 objects. The elements for this catalog must be updated and disseminated to users every day. Every element set in the catalog is updated by processing the multiple observations provided by the ground stations which are distributed around the world. In 1987, it was reported that, as a result of several major breakups, the number of observations being processed by the USSPACECOM Space Surveillance Center (SSC) had reached a record 71,891 observations in one 24 hour period [5]. The large number of observations and the large number of satellites in the catalog provides an indication of the difficulty of maintaining an up-to-date satellite catalog. A further complication is the number of uncorrelated observations, observations that are not immediately identified as belonging to a known object, that arrive at the central sites each day. It is estimated that 15%-20% of all observations are uncorrelated and that more than 10 times the normal amount of processing is required to tag these observations to the correct satellite. The computational burden associated with processing the observations has prompted some to propose a distributed processing approach to the problem. The idea would be to perform at the tracking sites some of the functions currently done at the central sites. This approach will probably incur difficulties in producing results that are compatible across the different tracking stations, and thus may be expensive to implement. We believe that centralized processing is still an economical means to process the data with parallel computers. This paper is the first step in demonstrating how parallel processing can alleviate some of the difficulties at the central sites.

2 Parallel Computers and the Connection Machine

In this section we discuss some specifics of massively parallel computers that make them so amenable to the tasks performed by the surveillance sites. We will frequently refer to the Connection Machine, CM2 and its recent upgrade, the CM200, when discussing SIMD computers since this is the best known computer of this type and because this is the machine available to us at the Naval Research Laboratory. This should not be interpreted as an endorsement of this brand of parallel computer, it is meant merely to provide a specific example of the unique architecture and capability of these types of computers.

Parallel computers can be divided into two categories: Single Instruction Multiple Data (SIMD) or Multiple Instruction Multiple Data (MIMD). A MIMD machine typically has a small number (< 128) of powerful processors and shared memory, each processor executing a different task or process. On the other hand, SIMD computers such as the CM2/CM200

have only one instruction stream, each instruction is broadcast simultaneously to each of the processors. The individual processors are comparatively weak and there are more of them (typically at least 1024) than on MIMD computers. What varies for the processors is the data, hence the term "data-parallel." Each processor has its own local memory for data; currently 128K bytes on the CM200. These two distinct kinds of architectures are now giving way to mixed architecture machines such as the Intel iPSC with 128 processors or the CM5 with 1K SPARC processors.

The number of physical processors available to the user is not an issue in the design of a program; the Connection Machine may be reconfigured to provide any number of virtual processors. For example, on our Connection Machine with 16K processors, splitting the memory in half effectively allows for variables with 32K elements. Thus a calculation with a 32K parallel variable means the processor calculates the first half of 16K entries and then in the next machine cycle it attends to the second half of the variable. Of course this doubles the execution time of an operation, but it does not bound the size of the application.

A data-parallel computer is ideally suited to many of the functions performed by the surveillance centers. The idea would be to assign individual processors to each satellite, the functions then performed for each satellite would be executed by the processor assigned to that satellite. Each processor would be operating on the data for that satellite.

There are currently available three major high-level languages for the Connection Machine: CM Fortran, C*, and *LISP, based respectively on the standard languages FORTRAN 77/Fortran 90, ANSI C, and Common LISP. We will restrict our attention here to CM Fortran.

The current standard for Fortran, FORTRAN 77 [6], while widely used, has no abstract way of manipulating arrays. Thus for instance, one can imagine a code fragment in FORTRAN 77 to add two arrays leaving the result in a third, and to shift the first array and leave the result in a fourth:

```
      integer i
      real a(100),b(100),c(100),d(100)
      do 10 i=1,100
        c(i)=a(i)+b(i)
        if (i.lt.100) d(i)=a(i+1)
   10 continue
      d(100)=0.0
```

We see here that, unavoidably, we have described the mechanism for doing the operation; looping with an index and performing the addition and shift elementwise – a mechanism that applies only to serial machines. On the other hand, a new standard for Fortran currently undergoing approval, called Fortran 90 [7], allows for abstract operations such as addition and shifting on whole arrays without specifying the mechanism:

```
      real a(100),b(100),c(100),d(100)
      c=a+b
      d=eoshift(a,1,1,0.0)
```

Code written this way can be run on serial or parallel computers; the compiler for each kind of architecture decides how the computation is to take place. For a parallel machine each addition, shift, etc. takes place in parallel.

CM Fortran for the Connection Machines is FORTRAN 77 with the array capability of Fortran 90, plus CM-specific features. The second code fragment above is essentially what one would program on the CM. Note that one is not concerned with manipulating virtual processors at the Fortran level; this is handled by the compiler. One can use arrays of any size and the only effect in going from one CM to another of a different size is the execution speed.

3 COMBO

The goal of COMBO is to determine close approaches of orbiting objects. Starting from some initial distribution, the catalog is propagated, then the positions of all objects are compared with all others in the catalog to find those that are within some critical distance c of another. This is an entire catalog-to-catalog comparison with no pre-screening of the data. The test data set we are working with is the full unclassified catalog of NAVSPASUR, with 6841 objects for the particular date of the set (October 8, 1991).

This demonstration of COMBO involves four basic elements: input, propagation, comparison, and output, both in tabular and graphical form. The initial element sets were supplied by NAVSPASUR on tape. These elements were converted to Keplerian elements and then stored on the Data Vault, an array of disks that allows direct parallel input and output for the processors.

3.1 Orbit Propagation

At the present, the propagation model incorporates only the J_2 secular terms from the averaged Hamiltonian. This model, which is more sophisticated than a Keplerian system, is sufficient for our purposes at this time. It provides for realistic orbital motion with a minimum of programming effort to implement.

We begin with the Hamiltonian where the potential has been restricted to the dominant zonal harmonic, J_2.

$$\mathcal{H} = \frac{1}{2}\left(R^2 + \frac{\Theta^2}{r^2}\right) - \frac{\mu}{r}\left(1 - \frac{J_2}{2}\left(\frac{\alpha}{r}\right)^2 (3s^2 \sin^2\theta - 1)\right), \tag{1}$$

This Hamiltonian is averaged with respect to the short period terms by putting it in normal form using Lie transformations [3]. We only use the secular terms derived from the averaged Hamiltonian. These equations are identical to Brouwer's [1] results. By restricting the expansion to first order, we do not encounter long-period terms. The propagator also does not include the short-period terms, thus there are no problems with singularities in e or $\sin I$. Similarly without the long-term coordinate transformations, we do not encounter problems at the critical inclination.

Efforts are underway to code the standard routines SGP4 of the US space command [8] and PPT2 of NAVSPASUR [9] for the CM. This effort will result not only in parallel

versions of these propagators, but will provide indications of the difficulty in transcribing these codes from serial to parallel machines. This will be useful for orbit determination or other applications. The propagator of the current code is a single subroutine so that with the development of new parallel propagators, the code may be updated in a straightforward fashion.

Furthermore, we have left open the possibility of other propagators from more complete theories, for instance, inclusion of higher orders and higher zonal harmonics [3] [4].

3.2 Close miss-distance computation

The computation of close miss-distances involves finding all pairs of satellites within a set that are within some specified critical distance c at a given time. The general approach is as follows. One may take the Cartesian coordinates $\vec{r} = \{x, y, z\}$ in arrays stored on the CM, and duplicate the set.

$\vec{r}_1$	$\vec{r}_2$	...	$\vec{r}_n$
$\vec{r}_1$	$\vec{r}_2$	...	$\vec{r}_n$

Then, with one copy fixed, we shift all three arrays by one and compute in parallel the distance of side-by-side elements,

$\vec{r}_1$	$\vec{r}_2$	...	$\vec{r}_n$	
	$\vec{r}_1$	...	$\vec{r}_{n-1}$	$\vec{r}_n$
	↓	Compute distances	↓	
	$\lvert\vec{r}_2 - \vec{r}_1\rvert$	...	$\lvert\vec{r}_n - \vec{r}_{n-1}\rvert$	

saving in a separate array pairs that are closer than some critical distance c. Then we repeat the shift – distance – save operation until all possible pairs have been compared.

This method, however, is unnecessarily time consuming. We may greatly reduce the computations by sorting the elements in advance. The position vectors are rearranged so that one Cartesian coordinate, say x, is sorted in ascending order. First observe that

$$|x_i - x_j| > c \Rightarrow |\vec{r}_i - \vec{r}_j| > c. \tag{2}$$

Then as a consequence of the sorted coordinates

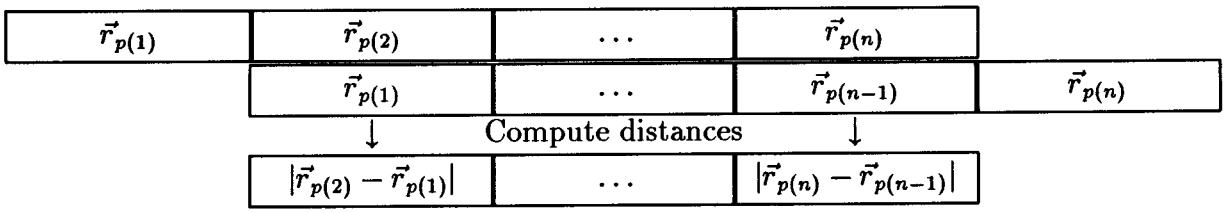
$$|x_i - x_j| > c \Rightarrow |\vec{r}_i - \vec{r}_j| > c \tag{3}$$

$$k > j > i \quad \text{and} \quad |x_j - x_i| > c \Rightarrow |x_k - x_i| > c. \tag{4}$$

These together imply that if, at a certain stage of the computation, the minimum of the differences of the x values is greater than c, no further shifts will produce an aligned pair whose Cartesian distance is less than c. Thus we modify the algorithm above; if we let p be the permutation that puts sets in x-order, i.e.

$$x_{p(1)} \leq x_{p(2)} \leq \ldots \leq x_{p(n)},$$

shifting and finding distances produces:



Now, however, after each shift an extra step checks the minimum x differences in parallel of all aligned pairs. If this minimum is greater than c, we terminate the computation, confident that all pairs that are within a distance c of each other have been found.

In this way, the actual number of computations is greatly reduced. With 6841 objects from NAVSPASUR for October 8, 1991, and a critical distance $c = 50$km, the number of shifts required to do a complete catalog-to-catalog comparison is reduced from 6840 to about 40–50. This speeds up the computation to the point where a complete step, consisting of a propagation, catalog-to-catalog comparison, and plotting takes less than one second.

3.3 Output

There are three different graphical displays available in this demonstration code:

1. A Mercator projection showing the whole earth and subsatellite points. Optionally, ground tracks may be recorded for particular satellites.

2. An orthographic (3D) projection showing the hemisphere of the earth facing a particular point in space and the satellites visible from that point.

3. A "window" around a particular satellite, showing the adjacent subsatellite points and a portion of the earth's surface moving as the satellite moves.

There are several mechanisms for graphical output: the framebuffer attached to the Connection Machine itself is most direct and thus fastest; additionally, a color or black and white X Windows server can display the graphics. Finally, any single image may be generated in color or black and white postscript for hardcopy output. Figure 3.3 shows a black-and-white picture of the Mercator projection. The dots indicate the position of the satellites while the cross marks (+) indicate those satellites that are within 50km of another satellite.

We also provide a mechanism for saving to a file a record of all close satellite encounters. The output includes the identification number element set, time, and the distance to the close satellite.

3.4 Multiple Sets and Debris

The scheme described above provides for exhaustive comparison of a set of satellite positions to itself. The program also allows for having two separate sets of data. One can then perform inter-set comparisons between objects in the two sets, while inhibiting intra-set comparisons. Possible applications range from single satellite comparison with the large catalog to looking at interactions between the catalog and a debris cloud emanating from an exploded satellite.

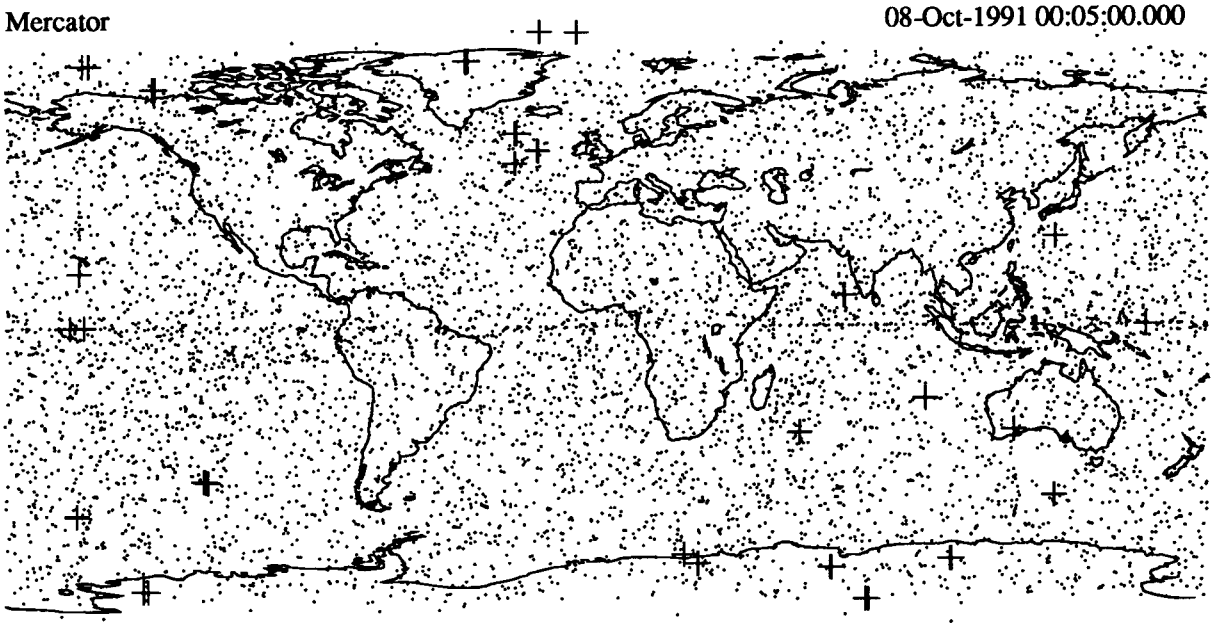


Figure 1: Mercator projection of subsatellite points

Propagating satellites for a range of values of the orbital parameters allows one to determine, in parallel, which potential orbits for a satellite are safe and which ones may result in collisions. This would allow for analysts to investigate how orbit errors would affect the safety of a mission, which is of particular interest for shuttle missions. This gives some indication of how the program ORBWIN [2], which is an expansion of the COMBO concept, could be implemented on the parallel processor.

For debris calculations, we construct a distribution of objects emanating from a single source. The simulation of a satellite's disintegration is computed by adding a small random component, for each fragment, to the source's momentum. The resulting particles are then propagated and compared to the catalog just like any other set of objects.

4 Other Potential Applications for Parallel Processing

We present here several potential applications for parallel processing. These applications represent investigations we intend to pursue over the next few years.

The simple idea behind the applications we have in mind is that each processor (or virtual processor) on the parallel computer would be assigned to a particular satellite. Each processor would then operate on the data for its satellite. The idea is most easily seen in the case of orbit propagation discussed previously. Here it is assumed the propagator is identical for each satellite, only the element set (i.e. the data) is different. Thus propagation for each satellite can be performed in parallel.

For the estimation of the orbital parameters both batch and sequential estimation appears feasible. In either case individual processors would perform all of the algebraic operations in updating the element set for the satellite to which that processor is assigned.

For batch estimation, the number of observations need not be more of a concern than on a

sequential computer, as long as the program is careful to not exceed the memory limitations of the processors. Of course the number of iterations required for the estimation to converge and the number of observations being processed will produce different completion times for each element. Once convergence is achieved for one processor, it could either remain in the loop doing meaningless iterations or it could be deselected and relegated to waiting for the remainder of the processors to complete their computations. Thus the program would run as long as required to get the slowest converging element. But, again the filter is being run in parallel for every satellite, resulting in a tremendous savings in time.

Sequential estimation is perhaps more straightforward conceptually, since the observations would be processed one at a time. One could consider maintaining the current elements in the memory of the computer. Then the observations could be processed in real time as they arrive from the tracking stations. Updates could be computed every few minutes. Of course the time between the update runs could be used for other functions.

A separate problem associated with satellite observations concerns retagging of the uncorrelated observations. Certain criteria have been prescribed for determining the satellite to which an observation belongs. If an observation does not meet this criteria, then it is classified as uncorrelated and separate processing is performed to try to assign it to the appropriate satellite. Of course this retagging effort is quite important because these observations may represent new satellite launches with potential military significance. It is now yet known exactly what the retagging process is, but it certainly is a candidate for parallel processing, merely because the process must be performed for so many observations and involves accessing the whole satellite catalog.

5 Conclusions

We have looked at several applications of parallel processing to space surveillance activities. We have produced a working demonstration for COMBO, which provides impressive improvements over serial methods. Although it has been difficult to make direct comparison to the operational programs running on the serial computers at the surveillance centers, estimates made by the COMBO experts at NAVSPASUR indicate that their computers would take many hours for the all-to-all satellite propagation and comparison. The demonstration code shown also has the ability to follow debris from an exploded or disintegrated object.

We have speculated on other applications. We hope in time to extend our number of demonstrations to include orbit determination and observation processing and thus answer the questions of how feasible parallel computing is to those problems. Eventually we may come to a point where parallel computers will be installed in the surveillance centers to execute those functions that they are clearly superior at performing.

References

[1] Brouwer, D.: "Solution of the problem of artificial satellite theory without drag," *Astron. J.*, **64** (1959), 378–397.

[2] Bredvik, G. D. and Straub, J. E., "Determination of Acceptable Launch Windows for Satellite Collision Avoidance," AAS/AIAA Astrodynamics Specialists Conference, 19-22 Aug. 1991.

[3] Coffey, S. L. and Deprit, A.: "Third-order solution to the main problem in satellite theory," *J. Guidance, Control and Dynamics*, **5** (1982), 366–371.

[4] Coffey, S., Deprit, A., Deprit, E., "Painting Phase Spaces to Put Frozen Orbits in Context," AAS/AIAA Astrodynamics Specialists Conference, 19-22 Aug. 1991.

[5] Cook, D. G.: "The Smart Catalog," AAS/AIAA Astrodynamics Specialists Conference, Kalispell, Montana, 10-13 Aug. 1987.

[6] *American National Standard Programming Language FORTRAN*, ANSI X3.9-1978 and ISO 1539-1980, American National Standards Institute, 1430 Broadway, New York, New York 10018.

[7] *Fortran 90*, ISO/IEC 1539:1991, Global Engineering Documents, 2805 McGraw Ave., Irvine, CA 92714.

[8] Hoots, F. R., Roehrich, R. L.: "Spacetrack Report 3: Models for Propagation of NORAD Element Sets," Office of Astrodynamics, Aerospace Defence Center, Peterson AFB, CO, (1980).

[9] Solomon, D.: "The NAVSPASUR Satellite Motion Model," Prepared for the Naval Research Laboratory, Naval Center for Space Technology, contract no. N00014-87-C-2547, 8 Aug. 1991.