Calhoun

Institutional Archive of the Naval Postgraduate School

Calhoun: The NPS Institutional Archive

Faculty and Researcher Publications | Faculty and Researcher Publications

1994

Performance of Analytic Orbit Propagators on a Hypercube and a Workstation Cluster

Neta, Beny

Naval Postgraduate School, Monterey, CA.

PERFORMANCE OF ANALYTIC ORBIT PROPAGATORS ON A HYPERCUBE AND A WORKSTATION CLUSTER

Beny Neta *

D. A. Danielson

Sara Ostrom

and

Susan K. Brewer

Naval Postgraduate School

Department of Mathematics

Code MA/Nd

Monterey, CA 93943

email: bneta@moon.math.nps.navy.mil

Abstract

In this paper we discuss the benefit of parallel computing in propagating orbits of objects. Several analytic methods are now in use operationally. We will discuss three such schemes. We demonstrate the benefit of parallelism by using an INTEL iPSC/2 hypercube and by using a cluster of Unix-based workstations running Parallel Virtual Machine (PVM). The software PVM allows a heterogeneous set of networked workstations to appear as a multicomputer.

We will show that one can achieve near 100% efficiency on the hypercube.

1   Introduction

The Naval Space Command (NAVSPACE-COM) and the Air Force Space Command (AFSPACECOM) currently track daily over 6000 objects in elliptical orbits around the Earth. To assist in identification and tracking of these objects in orbit, they both use an analytic satellite motion model. The Navy is using the subroutine PPT2 based on variation of elements model of artificial satellite motion around the Earth. The theory is due to Brouwer and Lyddane [9]. Given a set of satellite's "mean" orbital elements at a given epoch, the model predicts the state (position and velocity) vector at a future time. The model considers perturbing accelerations due to atmospheric drag, oblateness of the Earth, and asymmetry of the Earth's mass about the equatorial plane. The Air Force is using SGP4/SDP4, (Simplified General Perturba-

---

*Author to whom all correspondence should be addressed.

tions) based on the theory of Lane and Cranford [8]. The Deep space capabilities are due to Hujsak's [6] work. They replaced the old version SGP which was based on the work of Kozai [7] and Brouwer [2] and made operational by Hilton and Kuhlman [5]. The old version had no capabilities to track objects in "deep space" i.e. period greater than 225 minutes.

With the current increase in space operations, the number of objects necessary to be tracked is expected to increase substantially. Additionally, if there exists a desire to increase the accuracy of prediction, the resulting model would require even more computing resources and make achieving results even more time consuming.

Parallel computing offers one option to decrease the computation time and achieve more real-time results. Use of parallel computers has already proven to be beneficial in reducing computation time in many other applied areas.

Two common measures of effectiveness, accounting for both the hardware and the algorithm are speedup and efficiency. The speedup, S_p, of an algorithm is defined as

$$S_p = \frac{T_s}{T_p} \qquad \text{or} \qquad \frac{T_1}{T_p} \qquad (1)$$

where T_s is the time on a serial computer and T_i is the time on a parallel computer having i processors. The efficiency, E_p, is defined by

$$E_p = \frac{S_p}{p} \qquad (2)$$

and it accounts for the relative cost of achieving a specific speedup. many factors could possibly limit the efficiency of a parallel program. These factors include the number of sequential operations that cannot be parallelized, the communication time between processors, and the time each processor is idle

due to synchronization requirements, see e.g. Quinn [13].

Two decomposition strategies can be used in parallelization of any algorithm, i.e. control decomposition and domain or data decomposition. It was shown by Phipps et al [11] that control decomposition is inefficient for orbit computation using the analytic methods mentioned above.

In this paper, we will summarize the results of parallelization of the analytic orbit propagators using domain decomposition strategy. The INTEL iPSC/2 hypercube is used. We will also discuss the use of a cluster of Unix-based workstations networked and all running the Parallel Virtual Machine (PVM) software. PVM was developed by Oak Ridge National Laboratory. It is a software system that enables a collection of heterogeneous computers to be used as a coherent and flexible concurrent computational system (Geist et al, [4]). In the next section, we discuss the results of parallelization when using the INTEL hypercube. We give a brief introduction to PVM software in section 3. The results of parallelizing PPT2 on a cluster of workstations will be detailed in section 4. In section 5 we discuss PVM use in parallelizing the Air Force models. We give our conclusions in section 6.

2 Parallel Versions of PPT2, SGP, SGP4/SDP4

In this section, we discuss the parallelization of PPT2 as well as SGP, SGP4/SDP4. The idea id to let one processor read and distribute the data to the other $(p - 2)$ processors which propagate the orbit and send their results to another processor, the collector, which writes to the disk, see figure 1.

The results for n satellites, $36 < n < 10000$, are given in Table 1 for a hypercube consisting of 8 processors.

It is clear that P^3T is more efficient. This should not be of a surprise, since PPT2 requires more computation time (11.2 msec) than the others. Note also that the efficiency is improving with the number of processors.

The question is now how to find the optimal number of processors to use. Phipps et al [11, 12] have developed a model for the execution time to propagate n objects using p processor. The time $t(p)$ is given by

$$t(p) = t_{w1}(p) + t_{w2}(p) + t_c(p) \qquad (3)$$

where $t_{w1}(p)$ is the time the last node must wait to receive its first data set, $t_{w2}(p)$ is the total time the last node must wait for all its subsequent data, and $t_c(p)$ is the time for each node to propagate its share of the n objects. It was shown there that

$$t_{w1}(p) = (p - 3)t_m(1) \qquad (4)$$

$$t_{w2}(p) = \begin{cases} 0 & \text{if } t_{w1} < t_1 \\ \left(\frac{n}{p-2} - 1\right)(t_{w1}(p) - t_1) & \text{if } t_{w1} \geq t_1 \end{cases} \qquad (5)$$

$$t_c(p) = \frac{n t_1}{p - 2} \qquad (6)$$

where $t_m(1)$ is the time to send a single message between the distributing and working node and t_1 is the time to propagate one object. These were found to be

$t_m(1) = .0374$ msec   and $t_1 = 4.60$ msec.

Therefore, the speedup and efficiency for $n = 5000$ objects, can be plotted as a function of the number of processors. It can be seen in the next figure that for P^3T the maximum efficiency is 87% and is achieved when using 16 processors. For PSGP, the maximum efficiency is over 90% using 128 processors.

For PSGP4 and PSDP4, the maximum efficiency (over 90%) can be achieved when using 64 processors. Figures 2-5 show the plots of the efficiency of each code as a function of p. Note that the number of objects propagated by each code is different. When using SGP, one handles all orbits the same, but when using SDP4, only the "deep space" orbits are considered. The rest are handled by SGP4.

As a result of discussion with AFSPACE-COM, we realized that the propagator in usually called several times for each object. Each call corresponds to a specified time beyond epoch. SGP4 propagates data for low earth objects which requires more frequent tracking than deep space satellites. Thus, a relatively large number of observations are received per day by the AFSPACECOM for each low earth satellite. The estimated number of calls to SGP4 for each object is 75 and to SDP4 is 25. To analyze the speedup and efficiency, we note that each time a new set of satellite data is received by SGP4 an initialization subroutine is called before the SGP4 main subroutine is called. For every other incremented time specified for the same satellite, the initialization program is not called. Thus the execution time can be modeled by (Ostrom [10])

$$t_1 = t_f + (m - 1)t_s \qquad (7)$$

where t_f is the time to propagate the satellite including initialization, and t_s is the propagation time without initialization. The values of t_f, t_s as measured on the iPSC/2 hypercube are

$$t_f = 6.6\text{msec}, \qquad t_s = 2.2\text{msec},$$

thus

$$t_1 = 169.4\text{msec}.$$

Figure 6 depicts the speedup and efficiency versus hypercube dimension when propagating 5950 satellites to 75 times each. Clearly

3

much higher speedups are obtainable in this case. The maximum efficiency is nearly 100% when using a hypercube having 256 nodes.

A similar analysis for SDP4 (Ostrom, [10]) shows that $t_1 = 106.8$ msec. Using now 1050 satellites (15% of a total of 7000 objects) one finds near 100% efficiency using a 128-node hypercube, see Figure 7. This analysis can be extended to PPT2.

3   Parallel Virtual Machine

Parallel Virtual Machine (PVM) is a small ($\sim$1 Mbytes of C source code) software package that allows a heterogeneous network of Unix-based computers to appear as a single large distributed-memory parallel computer. The PVM package is good for large-grain parallelism; that is, as least 100 kbytes/node. The term <u>virtual machine</u> is used to designate a logical distributed-memory computer and <u>host</u> is used to designate one of the member computers.

The PVM software, developed at Oak Ridge National Laboratory (see Dongara et al [3] and Sunderam et al [15]) supplied the functions to automatically start up tasks to communicate and synchronize with each other. A problem can be solved in parallel by sending and receiving messages to accomplish multiple tasks, similar to send and receive on the hypercube.

PVM handles all message conversion that may be required if two computers use different data representations. PVM also ensures that error messages generated on a remote computer are displayed on the user's local screen.

The PVM system is actually composed of two parts, the daemon and a library of PVM interface routines. The daemon (pvmd or pvmd3) resides on all the computers making up the virtual machine. When a user desires to run a PVM application, he/she executes pvmd on one of the computers which in turn starts up pvmd on all the others. The library of PVM interface contains routines for message passing, spawning processes, coordinating tasks, and modifying the virtual machine.

4   Parallelization of PPT2 using PVM

Stone [14] has tried four possibilities of domain (data) decomposition.

- The master sends one satellite to each working node, then sends one satellite at a time upon request (ds1).

- The master sends one satellite to each working processor then continues in round-robin fashion (ds2).

- The entire data set is divided to p (number of working nodes) blocks. The master sends a block to each working node (ds3).

- The entire data set is divided to $2p$ blocks. The master sends one block to each and then the other block to each (ds4).

In the second option we save on communications. In the third case we save even more on communication because we reduced the number of times required to send data. On the other hand, sending such large blocks forces the others to wait. Thus the last case is an attempt to compromise between the previous two.

For these experiments, PVM was started on eighteen different workstations so measurements could be taken for one to sixteen working nodes. The workstations are SUN

Sparc II and Sparc IPX having 40 MHz processors and configured with 32 Mbytes of system memory. The workstations are connected by a 10 Mbytes Ethernet based network. Stone experimented with 600 and 1200 objects in the data set. We give here the result for 1200 (figure 8). It is clear that four working processors suffice to minimize the computing time and that the fourth possibility is the best. Stone [14] has shown that a speedup of almost 6 was achieved when using 8 SUN workstations.

5 Parallelization of SGP4 using PVM

Brewer [1] has tried three possibilities for domain (data) decomposition.

- Answer Back Method (ABM)

  The master sends one block of m satellites to each working node. Upon request a working processor receives another block of m satellites until the data set is processed.

- Successive Deal I (SDI)

  The master sends one block of m satellites to each working node and continues to deal such blocks in round-robin fashion.

- Successive Deal II (SDII)

  The master sends one block of m satellites to each working node. The rest of the data set is divided by $2p$ (twice the number of working nodes). Blocks of this size are given to each working nodes in round-robin fashion (2 blocks each).

The second method will eliminate the communication time by the workers requesting more data. The third method will cut the communication overhead. This is different from SDI with a larger m, because in SDII large blocks are sent while the workers are busy propagating the first m satellites.

We have experimented with various values of m and chosen 4,8 and 16 processors (i.e. 2,6,14 working nodes, respectively). The number of satellites taken to be 7000, 15% of which were considered deep-space. For a deep-space satellite 25 calls were made to SDP4. For the other satellites, 75 calls were made to SGP4.

The first measure is the end-to-end time. This is the most important, since it is a reflection of the total performance of each algorithm. The Answer Back Method was superior when using 4 or 8 processors. When using 16 processors, ABM was faster in most cases. See Figures 9-11.

We can look at this from another point of view. In the next three figures, we plot the end-to-end time for each method. It is clear from figure 12 that a choice of 8 or 16 processors is the best (shortest time) for ABM. For SDI and SDII a choice of 16 processors is best.

The second measure is the percent of time a working processor spent on communication. From the next three figures 13-15 is clear that SDII requires less communication time, which shouldn't be surprising. It is also clear that the more working nodes we have the higher the percentage.

The third measure is efficiency. In all three cases, the ABM was more efficient. The next three figures 16-18 show that for each method it is more efficient to use 4 or 8 processors rather than 16.

In closing we should note that with the use of an open network, there are great fluctuations in the amount of time taken to perform a given task. The execution time depends on the number of current users and the percentage of the CPU allocated to each user. To

partially compensate for that, we averaged 10 run times to arrive at our results.

6 Conclusions

In this paper we have shown the benefit of MIMD parallel computers in predicting the orbit of objects. Analytic orbit propagators currently in use by the Navy and Air Force were implemented on an INTEL iPSC/2 hypercube and on a cluster of networked Unix-based workstations running PVM. The efficiency of the algorithms nears 100% when using the optimal number of processors. This optimal number depends on the number of satellites, the orbit propagator used and the number of calls to the propagator per satellite. For a cluster of workstations we have used the software PVM and have shown that it is more efficient to use 4 or 8 workstations than 16. The speedup is almost 6 when using 8 workstations.

REFERENCES

1. S.K. Brewer, Air Force Space Command Satellite Orbit Predictor Using Parallel Virtual Machines, M. Sc. Thesis, Naval Postgraduate School, Department of Mathematics, Monterey, CA, 1993.

2. Brouwer, D., Solution of the Problem of Artificial Satellite Theory without Drag, Astronomical Journal, **64**, (1959) 378-398.

3. Dongara, J., Geist, G. A., Mancheck, R., Sunderam, V. S., Integrated PVM Framework Supports Heterogeneous Network Computing, Computers in Physics, pp 166-175, 1993.

4. Geist, G. A., Beguelin, A., Dongara, J., Jiang, W., Mancheck, R. and Sunderam, V. S., PVM 3 User's Guide and Reference Manual, Oak Ridge National Laboratory Technical Report ORNL/TM-12187, Oak Ridge, TN, 1993.

5. Hilton, C.G. and Kuhlman, J.R., Mathematical Models for the Space Defense Center, Philco-Ford Publication Number U-3871, pp. 17-28, November 1966.

6. Hujsak, R.S., A Restricted Four Body Solution for Resonating Satellites with an Oblate Earth, AIAA Paper Number 79-136, June 1979.

7. Kozai, Y., The Motion of a Close Earth Satellite, Astronomical Journal, **64**, (1959), 367-377.

8. Lane, M.H. and Cranford, K.H., An Improved Analytical Drag Theory for the Artificial Satellite Problem, AIAA Paper Number 69-925, August 1969.

9. Lyddane, R.H., Small Eccentricities or Inclinations in the Brouwer Theory of the Artificial Satellite, Astronomical Journal, **68**,(1963), 555-558.

10. Ostrom, S.R., Parallelization of the Air Force Space Command Satellite Motion Models, MS thesis, Naval Postgraduate School, 1993.

11. Phipps, W.E., Neta, B. and Danielson, D.A., Parallelization of the Naval Space Surveillance Satellite Motion Model, Journal Astronautical Sciences, **41**, (1993), 207-216.

12. Phipps, W.E., Neta, B., and Danielson, D.A., Parallelization of the Navel Space Surveillance Center Satellite Motion Model, Proceedings of the 1993

Space Surveillance Workshop, Lincoln Laboratory, Massachusetts Institute of Technology, March 30-April 1, Vol II, pp 71-79.

13. Quinn, M.J., Designing Efficient Algorithms for Parallel Computers, McGraw-Hill, New York, 1987.

14. Stone, L.C., Parallel Processing of Navy Specific Applications Using a Workstation Cluster, MS Thesis, Naval Postgraduate School, Department of Electrical and Computer Engineering, Monterey, CA, December 1993.

15. Sunderam, V.S., Geist, G.A., Mancheck, R., The PVM Concurrent Computing System: Evolution, Experience and Trends, Parallel Computing, 1993, submitted.

Serial Code	Parallel Code	$\max_n E_4$	$\max_n E_8$	time (msec) to propagate an object
SGP	PSGP	.38	.56	4.6
SGP4	PSGP4	.40	.60	6.6
SDP4	PSDP4	.43	.64	10.8
PPT2	P^3T	.45	.67	11.2

Table 1: Maximum efficiency of various propagators

List of Figures