Estimation Algorithm for Ocean Tides, C90-->T3D

Andrea Hudson, Dennis Morrow, Nick Pavlis, and *Braulio Sanchez,*
Nasa Goddard Space Flight Center and Cray Research, Inc.

SUMMARY

This paper presents the results of porting a code from the Cray C90 to the T3D. The code is part of a global ocean modeling application that reaches a tidal solution by *objective analysis* which is an empirical approach based on extrapolating satellite altimetry data through the use of expansion coefficients determined by an estimation algorithm.

The time line for migration of the estimation algorithm begins with the C90 vector version (single CPU and autotasked). Next the code is ported to the T3D using a message passing strategy (*SHMEM* calls for communication). Single PE optimizations for memory and cache use are continuing and a *back-port* of the code is planned to the J932 based on the division of work in the T3D version.

The major portion of the CPU time in the estimation algorithm is in the Basic Linear Algebra Subprograms (BLAS) libraries which give good performance on both Cray platforms. A major advantage of the T3D is that the much larger problems that are planned and that require more memory can be solved by adding more T3D processors.

ABSTRACT

There is a need for more accurate models of ocean tides and circulations. The accuracy of ocean models is improved when the models combine a theoretical simulation approach with an empirical analysis of satellite altimetry data. Increasing model accuracy often requires more memory and more processors working simultaneously. Efficient parallel solution algorithms can reduce the wall-clock time t o get a tidal solution as well as take advantage of the distributed memory available to solve large memory problems on the T3D.

INTRODUCTION

Successfully interpreting oceanic measurements provided by space borne altimeters improves the accuracy of oceanic tide models. An empirical tidal model for processing TOPEX/Poseidon satellite altimeter data solves for tidal parameters by an estimation algorithm. The algorithm is to difference the data collected at each ground location, then set up a system of observation equations, form the weight matrix, and then form the normal equations which provide a least squares solution for the unknown tidal parameters.

The model is typically run on a Cray C90. The BLAS Level 2 and 3 are used extensively in the formation of the normal equations. The current problem solves for 5,120 unknowns and ultimately it is desirable to solve for 15,000 unknown tidal parameters simultaneously.

The large distributed memory and good performance of the BLAS libraries on the T3D motivated the migration of this application in anticipation of the upcoming need to solve for a larger number of unknown parameters. Results are presented for both the T3D and the C90/J90.

APPLICATION and ALGORITHM

This application involves altimeter data retrieval from the TOPEX/Poseidon satellite which orbits the earth in such a way that every ten days it passes over the exact same location on the earth's surface. The estimation algorithm begins with differencing the data collected by the satellite at each location on the ground track. (see Figure 1.)

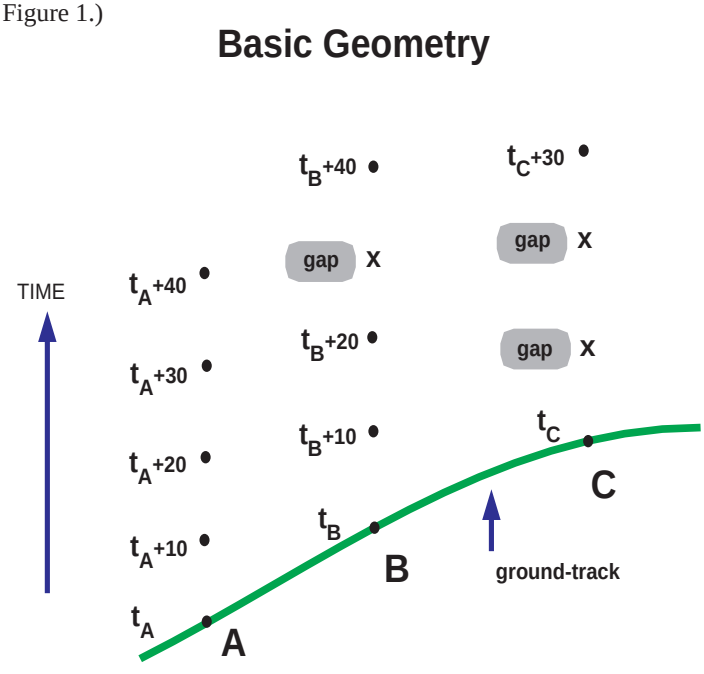

Figure 1. Basic Geometry

The differenced data form the set of *observation* equations. The observation equations are reduced to a symmetric matrix of order 43 (the current maximum number of differenced observations representing 430 days of TOPEX/Poseidon data). This matrix is the inverse of the variance/covariance matrix and is also called the *weight* matrix.

The algorithm is a loop through all the reference locations (currently 50,000). The observation equations are formed, and the contribution of each location to the normal equations is calculated by a series of four BLAS library routines: *STRMM, STRMV, SSYRK,* and *SGEMV*.

The algorithm can be parallelized since the calculations are independent for each location. This approach to parallelization, although straightforward, does not reduce the memory requirement. A second approach to parallelization is to divide the data and the work in updating the normal matrix in the *SSYRK* BLAS routine. Subdividing the data in this manner also reduces the memory requirement for each individual processor.

MIGRATION of ESTIMATION ALGORITHM

C90 Code characteristics

The code processes data for 50,000 locations and solves for a maximum of 43 observations per location. The plan is to move toward solving for 80 observations per location. The current code solves for 5,120 unknown parameters, and the plan is to move toward solving for 11,200 tidal parameters immediately and to eventually solve for 15,000 parameters.

The runtime profile of the code is dominated by the *SSYRK* BLAS routine which takes 98% of the total CPU time on the C90. The code runs for over 14 hours at a rate of 867 Mflops on one C90 processor and uses a maximum of 27 Mwords of memory.

I/O is not significant. The normal matrix and right-hand -side vector (about 105 Mbytes) are written out to disk at the conclusion of the code and a follow-on code actuall y solves the equations to retrieve the tidal parameters.

Migration to T3D

Figure 2 shows data layout scheme for distributing the right-hand-side vecto r **U**, the normal matrix **DN**, and the intermediate matrix **A**. This figure illustrates an equal division of data among four processors labeled PE0 through PE3.

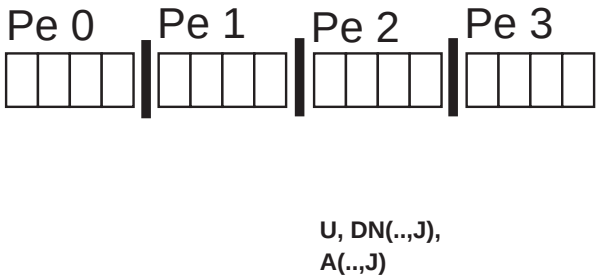

Figure 2. Distributed vector **U** and arrays **A** and **DN**

Each processor is assigned one quarter of the data. The dimension of the **U** vector is 1,280 on each of the four T3D processors, making up the total length of all 5,120 elements in **U** .

The work in calculating the **J** columns of both matrix **A** and **DN** is split similarly among the four processors. The minimum number of processors that could be used for this particular application is four because it takes at least four processors to split the large **DN** array (which has dimensions 5120 x 5120 on the C90). The dimension of **DN** on each T3D processor is 5120 x 1280.

Figure 3 shows the process of updating the **U** vector (the right- hand-side). This i s done with the *SGEMV* BLAS routine. The intermediate vector **DL** is replicated on each processor. Each processor has only one quarter of the **A** matrix and the **U** vector.

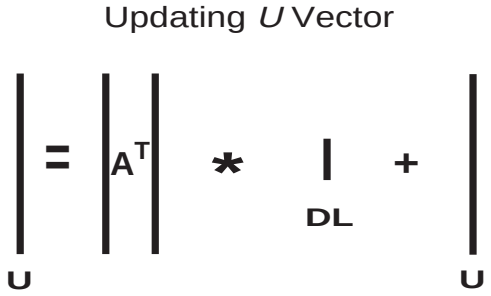

Figure 3. Updating the **U** Vector

The load is exactly balanced and there is no communication needed since the **U** vector is an end result, and therefore does not move out of the owning processor's memory until the end of the job.

Figure 4 shows the process of updating the **A** *intermediate* matrix. This is accomplished with a single call to the *STRMM* BLAS routine. Each processor has the entire **R** *weight* matrix, and does calculations on only one quarter of the **A** matrix. The **R** matrix is a square matrix of order 43. (Figure 4 is not to scale).

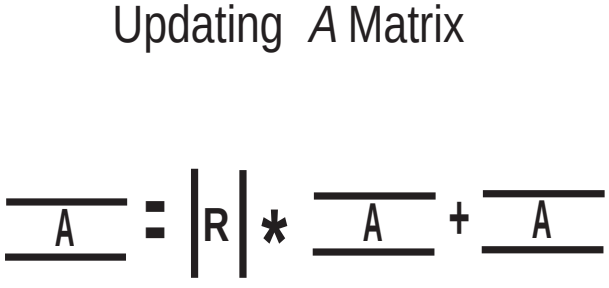

Figure 4. Updating the **A** Intermediate Matrix.

Communication is then needed among processors since an up-to-date copy of t he entire **A** matrix is needed on all processors for the next BLAS library routine. This communication is shown in Figure 5.

Communication is first accomplished using the *shmem_put* primitive from the SHMEM library to build a composite **A** matrix on PE0. Next the *shmem_broadcast* routine is used to distribute the composite **A** matrix to all PEs. Figure 5 shows a schematic of the redistribution of the composite **A** matrix on eight PEs. This communication scheme needs to be optimized because it requires a barrier synchronization and it also creates a *hot spot* where all PEs are trying to communicate with PE0 at the same time. The *Apprentice* performance analysis tool showed that there was very little gain over having all the PEs redundantly calculate the entire **A** matrix, and then avoiding this particular communication altogether.

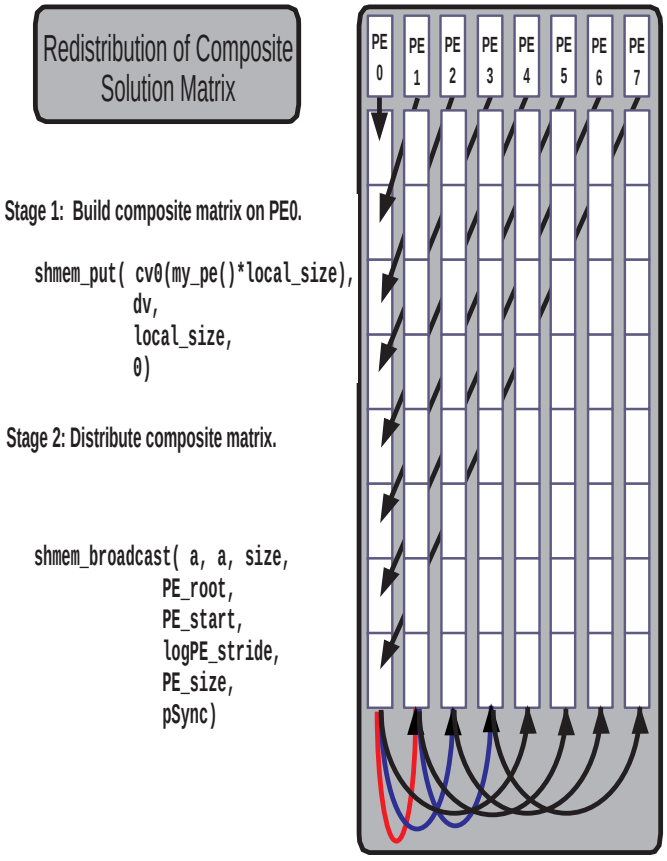

Figure 5. Communicating the **A** Matrix

Figure 6 shows the process of updating the **DN** *normal* matrix. On the C90, this is accomplished with a single call to the *SSYRK* BLAS routine. On the T3D this is accomplished by a loop over a call to the *SGEMV* blas routine. Each processor has the entire **A** matrix and exactly one quarter of the entire **DN** matrix. An initial strategy of assigning PE0 the first set of columns in **DN**, PE1 the second set, etc. lead to *load imbalance* and required a call to a barrier routine for synchronization. This was because PE0 had all the short length columns, and therefore finished first. A subsequent strategy of assigning *stripes* of columns balanced the workload because each PE then had a mix of short and long columns from the normal matrix. No communication is involved because, like the **U** right-hand-side vec tor, the normal matrix is a final result and is not needed until the end of the job.

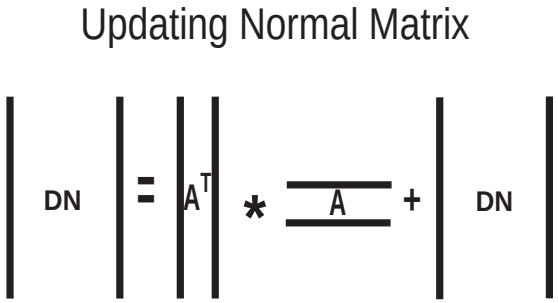

Figure 6. Updating the **DN** Matrix.

In summary, migration from the C90 to the T3D is accomplished by copying the weight matrix **R** and other initialization data to each processor. Each processor does redundant work in calculating the **DL** intermediate vector and in equation setup. The work in the *STRMM* BLAS routine can be split up among the processors, but this requires communication. The *SSYRK* routine can be split into a series of calls to the *SGEMV* routine, and this allows the rank *K* update to be done in parallel. Finally *shmem_min* and *shmem_max* routines are used to identify the minimum and maximum element in both the normal matrix and the right-hand-side ve ctor in order to verify results with the C90.

RESULTS

Table 1 presents the YMP-C916 non-dedi cated runs on 1-8 c pus for 5,120 unknowns and with only 100 locations . One hundred locations is a small but representative subset of the 50,000 actual locations that need to be processed.

5120 Unknowns and 100 Locations

# cpus	CPU sec.	wall sec.	Mflops /CPU	Speedup	total Gflops
1	102	102	867	1.0	0.87
2	103	73	852	1.7	1.4
4	103	43	853	2.5	2.1
8	103	33	853	3.3	2.8

Table 1. YMP-C90 r uns

All the listed computer runs were made during the day on production systems and none of the results are *benchmark* runs. The reported wall-clock times w ill vary depending on the load and number of jobs in the system. The listed CPU times and megaflop-per-CPU rates are fairly independent of system load. All the runs listed were made with autotasking and the major factor in speedup is the autotasked BLAS routines in the *SCILIB* scientific library.

The CPU times increase somewhat with additional CPUs and the reduction in wall clock time illustrates the tradeoff between turnaround time and CPU cycles.

# cpus	CPU sec.	wall sec.	Mflops /CPU	Speedup	total Mflops
1	470	471	188	1.0	188
2	476	283	185	1.7	314
4	477	186	185	2.6	481
8	478	138	184	3.5	642
12	482	123	183	4.0	732
16	486	115	182	4.3	779

Table 2. J916 runs.

Table 2 shows the YMP-J916 non-dedicat ed runs on 1-16 cpu s for the same size problem using the autotasked BLAS routines. The speedup *flattens out* at just above four. Autotasked BLAS routines provide a good parallel solution on a small number of CPUs with minimal overhead on a loaded system.

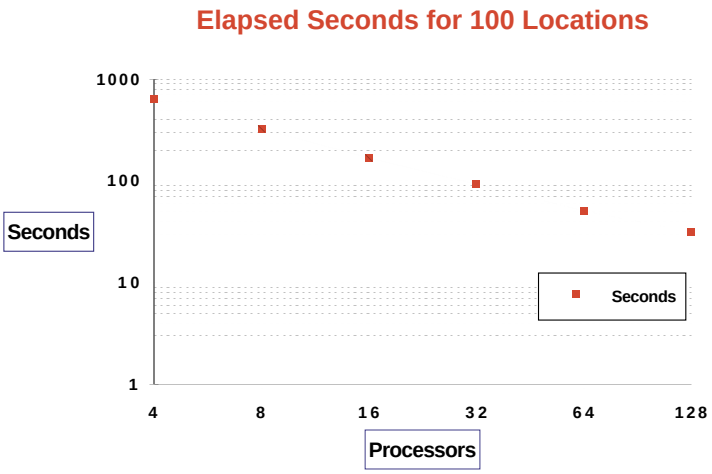

Figure 7. Elapsed T3D seconds.

Figures 7 and 8 show the T3D results. Figure 7 shows elapsed seconds and Figure 8 shows equivalent C90 Mflops. Both figures have the number of T3D processors plotted on the X-axis.

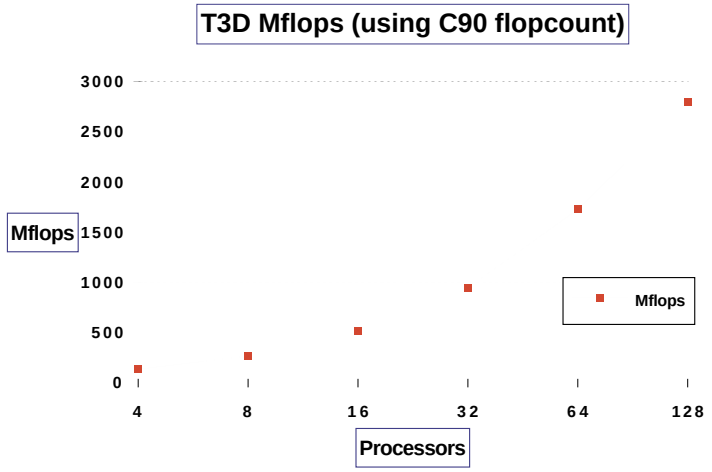

Figure 8. C90-equiva lent Mflops on the T3D

The T3D Mflops reported in Figure 8 are *C90-equivalent* Mflops conservatively calculated by taking the raw flop count from the hardware performance monitor on the C90 and dividing that number by the elapsed time on the T3D. The T3D version of the estimation algorithm makes redundant calculations and therefore achieves a higher actual Mflop rate.

FUTURE PLANS

Our future plans are as follows:

1. Larger problems (more unknowns and more observation per location)
2. Single PE optimizations for memory and cache use
3. *Back-port* the T3D version of the algorithm to the J932
4. Implement the PBLAS (Parallel Basic Linear Algebra Subprograms) on the T3D and compare performance with the *SHMEM* implementation
5. Remove the *power-of-two* restriction on the number of unknown tidal parameters
6. Parallel I/O

CONCLUSIONS

The profile of the C90 and J90 autotasked implementation showed that 98% of the total CPU time is spent in BLAS library routines. Using autotasking, speedups ranged from 2 to 4 out of 12 CPUs. *Production* rates on the J916 were 780 Mflops and 2.8 Gflops on the C98.

The distributed memory implementation on the Cray T3D attained 2.8 Gflops (conservatively based on C90-equivalent flop count divided by the elapsed time). The speedup curve up to 128 processors is proportional to the number of processors. The curve does not appear to have *flattened* yet, but eventually the serial code will begin to dominate as the number of processors increases.

ACKNOWLEDGEMENTS

The Cray supercomputer used in this investigation was provided through funding by NASA Offices of Mission to Planet Earth, Aeronautics, and Space Science.

REFERENCES

1. Cray Research, Inc., *Unicos Math and Scientific Library Reference Manual, Volume 3*, SR-2081 6.0, 1991.
2 Sanchez, B.V., Rao, D.B., and Wolfson, P.G., *Objective Analysis for Tides in a Closed Basin*, Marine Geodesy, 9(1), 71-91, 198 5.