CONF-9210177-21

The submitted manuscript has been authored by a contractor of the U. S. Government under contract No. W-31-109-ENG-38. Accordingly, the U. S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U. S. Government purposes.

Received by Doc
SEP 13 1992

**PARALLELIZING THE TRACK-TARGET MODEL
FOR THE MIMD MACHINE**
1992 IEEE Systems, Man and Cybernetics Conference Proceedings

Wei ZhongXiong
Dr. Craig Swietlik
Advanced Computer Application Center
Argonne National Laboratory
Argonne, Illinois 60439

ANL/CP--76975

DE92 041100

Abstract

Military Tracking-Target systems are important analysis tools for modelling the major functions of a strategic defense system operating against a ballistic missile threat during a simulated end-to-end scenario.[1] As demands grow for modelling more trajectories with increasing numbers of missile types, so have demands for more processing power. Argonne National Laboratory has developed the parallel version of this Tracking-Target model. The parallel version has exhibited speedups of up to a factor of 6.3 resulting from a shared memory multiprocessor machine. This paper documents a project to implement the Tracking-Target model on a parallel processing environment.

Alliant FX/8 is a shared memory multiprocessor system with eight high-speed computational elements (CEs) which optimize performance on compute intensive problems. The FORTRAN compiler supports both vector mode and concurrency. The operating system is Concentrix, which is a multiprocessor UNIX-based system.

During this parallelization task, a data dependency analysis was conducted and changes to convert the serial version to the equivalent parallel version were analyzed in two separate areas. First, the physical forces acting on the missile (gravity, thrust and drag) were determined to be independent and were tasked to separate processors for parallel execution. Secondly, the Runge-Kutta method of solving ordinary differential equations was restructured, so that parallel execution could be achieved.

I. INTRODUCTION

The Tracking-target model serves as a trajectory generation and scenario definition module. It can model ballistic missile threats with components of booster stages, post-boost vehicles, clusters, light decoys, and heavy decoys. The calculations of the model are essentially trajectory generations. It is primarily used for non-stiff differential equations $dy/dt=f(t,y)$ when derivative evaluations are inexpensive. Here $y=(y(1),...,y(NEQN))$ is a system of equations to be integrated. The program was implemented for a DEC/VAX system, but due to extended run times and system load, timely turnaround results were not achievable.

In late 1991, the Advanced Computer Applications Center at Argonne National Laboratory was led to develop a parallel version of this Tracking-Target model, which runs on the eight processor Alliant FX/8 platform. The

II. TRACKING-TARGET MODEL

The Tracking-Target model is a program based on a DEC VAX/VMS platform in FORTRAN, developed by Science Applications International Corp. To generate trajectories, the program initializes the user-defined missile parameters and trajectory parameters, then applies a series of processing functions. The missile parameters are used for flight calculations in the model. The missile can have up to three thrust stages with each stage requiring certain parameters. The trajectory parameters include altitude, launch and target positions. The model can then feed this information to the target algorithms for trajectory generation. Pseudo-code for a simplified processing flow is presented below.

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

```

MAIN:
  do for each trajectory
    if missile type change
      initialize missile parameters
      and trajectory parameters;
    endif;
    call TARGET ALGORITHM;
  enddo

TARGET ALGORITHM( )
  ..
  call RK45;(4th-5th order Runge-Kutta
  method)
  ..
  return;

RK45( )
  compute indices for the splitting of the
  work space;
  compute machine epsilon;
  ..
  call RCKTFN;(rocket motion differential
  equations)
  ..
  return;

RCKTFN( )
  compute thrust;
  compute gravity;
  compute drag;
  return;

```

The initialization of missile and trajectory parameters accounted for as little as 5% of the execution time on sequential machines. The remaining 95% of time was spent executing the targeting algorithm where the 4th-5th order Runge-Kutta ordinary differential equation solver coupled with the physics motion equations modeled the flight. As a result, the model is compute intensive and requires extensive CPU time. To address the problem of long run times, the task was established to convert this model from the VAX to the parallel environment. This environment was the Alliant FX/8 vector multiprocessor system. The first step of this task was to port and optimize the entire model on Alliant FX/8 system. The Alliant compiler provides automatic optimization flags and compiler directives to be used in a manual mode to optimize code.[2] With careful tuning, the program was automatically optimized by compiler, and the performance of the entire model was improved by about

30% of the CPU time. As efforts proceeded in further refining the optimizations, a diminishing return was discovered, as expected. Since the computation of trajectories is inherently highly parallel, because of the independent equations used to calculate the trajectory, the only dependencies among the processes was the initialization for each missile type. Efforts were redirected toward parallelizing the trajectory computation across multiple processors. Therefore, the removal of the data dependent relations between the initialization process and computation process was a crucial issue for the design of the parallel version of model. The execution profile also indicated during the Runge-Kutta computation, that the 3-dimensional equations of motion accounting for the three forces, gravity (g), thrust (T) and drag (d) were consuming the most CPU time. The model sequentially computes the value for T, g and d. We identified that these computations can be parallelized due to their nature, $dv/dt = T + g + d$.

III. THE PARALLEL VERSION OF TRACK-TARGET MODEL

As stated earlier, two kinds of changes were made to parallelize Track-Target Model. One was to remove missile initialization from the trajectory computation loop, so that trajectory computations can be distributed across 8 processors and completed in parallel. Another was to modify the motion equation routine, so that three processes can be created and each one assigned to calculate the thrust, gravity and drag in parallel. The final algorithm differed from the sequential version, but was more natural to the problem. The pseudo-code that follows describes the parallel algorithm implemented on the Alliant FX/8 machine.

Change 1:

```

MAIN:
  do for each missile type
    initialize missile parameters and
    trajectory parameters;
  do for each trajectory with that missile
    create COMPUTE
    PROCESSES; (one per
    processor)
  enddo
enddo

COMPUTE PROCESSES: (8 processors)

  call TARGET ALGORITHM;

```

```
return;
```

Change 2:

```
RK45:
compute indices for the splitting of the work
space;
compute machine epsilon;
..
do 1 to 3
    create processes (one per processor)
    process 1 compute thrust;
    process 2 compute gravity;
    process 3 compute drag;
enddo
..
return;
```

We used the following techniques to create a parallel version of Track-Target Model. To insure proper asynchronization, we created an internal table with an index for the process identifier. This allowed each process to write data independently to an array held in memory. After the computations for a missile type were completed, the memory was written to an output file. We also removed several procedure calls from the RK45 algorithm and replaced them with the procedures themselves to reduce calling overhead (inlining). All I/O statement used for the diagnostics file were saved in the internal arrays and written out after the parallel computation was done.

IV. PERFORMANCE

To measure the performance of parallel algorithms implemented on the multiprocessor system, the speedup and efficiency are the important metrics. The common definition of speedup achieved by a parallel algorithm running on n processors, is the ratio between the elapsed time taken by that multiprocessor system executing the serial algorithm and the elapsed time taken by the same system executing the parallel algorithm using n processors. The efficiency of a parallel algorithm running on n processors is the speedup divided by n .

In our parallel model, we have carefully studied the output of the parallel version before conducting any timing tests. The correct output of the parallel version was verified by comparing the results with the output from the serial version. The results of the two versions were verified to be identical.

Next, performance of the parallel version was measured

using two sets of test data, one contained 5 missile types with 76 trajectories, another contained 5 missiles types with 142 trajectories. The results from the experiments on the Alliant FX/8 are shown as below in Table 1 for Timing Data and Table 2 for Efficiency Data.

Table 1: Timing Data

Trajectories	Elapsed Time (serial)	Elapsed Time (parallel)	Speedup
76	8126.4	1480.6	5.6
142	15102.3	2398.4	6.3

Table 2: Efficiency Data

Trajectories	Speedup	Processs	Efficiency
76	5.6	8	0.7
142	6.3	8	~0.8

The above speed up factor is the elapsed time (wall clock time) of the serial version divided by the times for the parallel version. The result above indicates that an increase in the problem size would improve the speedup. This reflects the Amdahl effect.[3] If there is not enough work to be done by the number of processors available, then the program would show constrained speedup.

V. CONCLUSIONS

The goals of the project were met. We have demonstrated substantial speedups by parallelizing the Track-Target model program. Further research on this model is planned to parallelize the initialization phase, and to extend the research to other parallel machines, such as the Sequent Symmetry with 24 processors, to research the optimum number of processors for the parallel environment.

Since design of a truly efficient parallel program model for a MIMD (Multiple Instruction Multiple Data) machine is a difficult task, the analysis of such an application model for an MIMD machine in a realistic setting has been largely overlooked. The approach presented in

this paper provides a general concept for parallelizing real application programs. We believe this concept, in most cases, can also be adapted to many other parallel problems.

ACKNOWLEDGMENTS

This work was supported under a military interdepartmental purchase request from the U. S. Department of Defense, J-8 Directorate of the Joint Chiefs of Staff, through U. S. Department of Energy contract W-31-109-Eng-38.

The authors express their appreciation for support in particular from LCDR Horn, CDR Petersen, CDR Morris and Lt Col Fallstead. Appreciation is also extended to Peter Campbell, John Gabriel, Lou Kvitek, and Roger Hanscom.

REFERENCES

- [1]. "Mission Effectiveness Model Version 11 Technical Reference Manual", Science Applications International Corporation, September 1989
- [2]. "FX/Fortran Programmer's Handbook", Alliant Computer Systems Corporation, May 1985
- [3]. Quinn, Michael J., "Designing Efficient Algorithms For Parallel Computers", pp. 17-18, McGraw-Hill Book Company, 1987

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

END

**DATE
FILMED**

12 / 9 / 92