IMPLEMENTATION AND PERFORMANCE ANALYSIS OF PARALLEL ASSIGNMENT ALGORITHMS ON A HYPERCUBE COMPUTER

Barry A. Carpenter[1] and Nathaniel J. Davis IV

Department of Electrical and Computer Engineering
Air Force Institute of Technology
Wright-Patterson AFB, Ohio 45433

ABSTRACT

The process of effectively coordinating and controlling resources during a military engagement is known as *battle management/command, control, and communications* (BM/C3). One key task of BM/C3 is allocating weapons to destroy targets. The focus of this research is on developing parallel computation methods to achieve fast and cost effective assignment of weapons to targets. Using the sequential Hungarian method for solving the assignment problem as a basis, this paper presents the development and the relative performance comparison of four parallel assignment methodologies that have been implemented on the Intel iPSC hypercube computer. The first three approaches are approximations to the optimal assignment solution. The advantage to these is that they are computationally fast and have proven to generate assignments that are very close the the optimal assignment in terms of cost. The fourth approach is a parallel implementation of the Hungarian algorithm, where certain subtasks are performed in parallel. This approach produces an optimal assignment as compared to the sub-optimal assignments that result from the first three approaches. The relative performance of the four approaches is compared by varying the number of weapons and targets, the number of processors used, and the size of the problem partitions.

1. INTRODUCTION

Parallel processing is a method of computation that exploits the concurrent events that occur in the solution of many different problems. Parallel computers employing multiple processors exploit these concurrent events by assigning each event to a different processor for simultaneous processing. Recent software implementations have shown that significant reductions in processing times are possible using parallel processing. The ability to compute faster solutions to large scale problems appeals to many researchers in government and industry. One particular government organization with a keen interest in the increased processing speeds provided by parallel processing is the Strategic Defense Initiative Organization (SDIO).

The Strategic Defense Initiative (SDI) was launched in 1983 as a research and development program to determine if a "smart" system of nonnuclear defense could effectively destroy incoming offensive ballistic missiles before they detonate over our country. The overall system architecture of the SDI system is envisioned as one of several defensive layers corresponding to the different phases that occur in the trajectory of a ballistic missile. Those phases are the boost phase, the midcourse phase, and the reentry or terminal phase. Within each defensive layer, computers will use information gathered from sensors to detect, classify, and track potential targets. Using this information and predefined engagement strategies, weapons will be assigned to destroy certain high-threat targets. After firing on assigned targets, the effectiveness of the weapons would be evaluated and used to make future weapon engagement decisions. The combination of all of these processes is known as battle management/command, control and communication or BM/C3 [SeD85].

One of the critical BM/C3 tasks is the assignment of weapons to targets. Situations similar to the problem of assigning weapons to targets frequently occur in other areas such as operations research, logistics management, and even in a computer's internal management of its resources. Typically, there exists a number of resources available to be allocated to a number of requesters. In most cases, there are more requesters than there are resources. In cases such as these, decisions must be made as to which requesters are allocated resources and which requesters are denied resources. The problem is generally known in the literature as the assignment problem and usually involves allocating available resources to competing requesters in such a way as to maximize some measure of profit or award, or to minimize some measure of penalty [Kuh55, Chu57, Kur62].

The assignment problem can be solved in many different ways. The brute force method would be to enumerate all the possible ways resources could be allocated to requesters and then choose the combination that provides the best allocation. This method might work well for a very small number of resources and requesters, but for any realistically sized system, the time required to enumerate all of the possibilities would be prohibitive. A significant amount of research has been conducted over the last forty years in an effort to provide more time efficient methods of arriving at the best, or very close to the best, allocation of resources. The objective of this paper was to implement and analyze the performance of assignment algorithms on a parallel multiprocessor computer. In the analysis, attention was focused on the effects of inter-processor communications, load balancing among processors, and execution times. The parallel computer used

[1]Captain Carpenter is currently assigned to the 31st Technical Evaluation Squadron (SAC), Edwards AFB, CA.

This work was supported by a grant from the Strategic Defense Initiative Office.

for the implementations was the Intel iPSC (Intel Personal Super Computer) multiprocessor system.

The experimental model and the assumptions governing its analysis are presented in Section 2. Section 3 describes the four parallel approaches used to solve the assignment problem. The performance of the four approaches is discussed in Section 4. Section 5 presents conclusions to the research.

2. EXPERIMENTAL MODEL

Because the main focus of this study is on the implementation of a parallel weapon-target assignment algorithm, an entirely realistic simulation of missile trajectories and distribution patterns of missiles within the different geographic regions is not attempted. For this reason, exact details of the battle management system such as how the individual targets are detected and tracked; the specifics of particular weapons; the operation and sensitivity of sensor devices; and the three-dimensional and rotational characteristics of weapon-to-target geometry are not addressed and are assumed to function independently with respect to the targeting assignment mechanism.

A number of simplifying assumptions pertaining to the operation of the BM/C3 system have been made in order to concentrate on the weapons-to-target assignment problem These assumptions do not detract from the problem's solutions. First, the number and location of potential targets, along with their relative importance, are assumed to be available on demand. Likewise, the number and status of available resources or weapons are also assumed to be immediately available when requested. Problems associated with detecting and classifying potential targets, and the details of evaluating the effectiveness of weapons already assigned to targets are not considered, although simulated results of those functions are supplied as input data to the programs. Weapons are considered to be reuseable with a finite number of "shots," and are assignable to one target at a time for a single "shot." Each instance of assignment is assumed to be one "snapshot" of the dynamic process of missiles in some phase of their trajectory. The assignment process is further assumed to be memoryless. This means that each assignment iteration is based only on the current cost information provided to it and is unaffected by previous assignments. However, the assignment process may choose to allow certain weapons to remain idle for future use if the present cost of utilization is considered too high.

Plausible missile attack scenarios have been generated and evaluated using an unclassified ballistic missile defense simulation program. Factors such as space-based weapons platform orbits, rotation of the earth, and plausible missile trajectories are accounted for in the simulation program [Odo85]. The scenarios are used as a basis for constructing "cost" data as input to the parallel assignment implementations developed in this study.

The data generation program developed in this research uses a random number generator to produce cost values in the range of 1 to 2500, which correspond to the range of values for the heuristic just described. The lower data values correspond to a high probability of kill and low cost assignments. The higher values indicate low probability of kill and high cost assignments (i.e., long distances, small angles of impact). Although the data used in the cost matrix is random, provisions could be made to produce lower values in some sections of the matrix and higher values in others. The groupings of low and high values would represent groups of weapons that have similar opportunities for engaging the same targets.

3. IMPLEMENTATION DESCRIPTIONS

As background for this research, techniques for developing parallel algorithms were reviewed and evaluated for their suitability to the assignment problem. Sequential algorithms that have been developed to solve the assignment problem were also evaluated. The Bourgeois and Lassalle (B&L) [BoL71a, BoL71b] version of the Hungarian method was chosen as a basis for the parallel implementations because of its ability to handle the case of non-square cost matrices without the addition of dummy variables, typically used in the traditional Hungarian method.

In this section, four different parallel implementations of the assignment algorithm are presented. The first three programs are closely related and vary mainly in the amount of interprocessor coordination. Although all of the first three programs use the B&L algorithm to perform the assignment task in parallel, the final assignments produced are not optimal. Heuristics are used to reduce the number of redundant assignments and will be fully discussed below. The fourth program is an effort to implement a parallel version of the B&L algorithm whose final solution is optimal.

The first implementation: no communications: The first parallel implementation, or level 1 program, is the case where there is no coordination between any of the processors in the iPSC. The initial matrix of assignment cost information is partitioned into strips (groups of rows) and distributed among the different processor nodes. Each processor utilizes the sequential B&L algorithm to compute assignments within its cost submatrix and works entirely independent of the other processors. Figure 1 illustrates the relationship between the individual processors in the cube and the cube manager. With a 5-dimension hypercube, up to 32 processors can operate in parallel on different portions of the cost matrix. The execution time is expected to be much shorter than either of the sequential implementations, however the resulting assignment will not be optimal.

The non-optimal assignment solution of this implementation results from the individual processors not communicating with each other about which targets have been assigned. As a consequence, one processor may assign a weapon to a certain target while another processor may assign a different weapon to the same target. This wastes one weapon that could have been assigned to another target. A larger number of processors will most likely result in more redundant assignments and more wasted weapons, but will calculate these results much faster than could a single processor implementation.

The second implementation: partial communications, single iteration The "level 2" parallel implementation introduces some coordination between the processors computing assignments for certain partitions of the cost matrix. The coordination is performed by processors designated as *controller* processors. The processors performing the assignments are known as *assign* processors.

Partitions of the cost matrix sent to the controller processors are further subdivided by the controllers and sent to the appropriate assign processors. The weapon-target pairings from the assign processors are examined by their associated controllers to identify redundant assignments. The controllers then eliminate redundancies by comparing the individual costs of those assignments that are conflicting. The controllers allow the lowest cost weapon allocations to remain and sets all the higher cost, redundantly assigned weapons to an idle state. The communications paths for this approach are shown in Figure 2, for two control processors and 6 assign processors. Since redundant assignments can be identified within controller groups, the expected benefits of this approach are fewer overall redundancies and lower assignment costs. The coordination overhead within the control processors can become significant if large numbers of

redundancies occur. This may degrade the system performance.

The third implementation: partial communications, multiple iterations The "level 3" parallel implementation increases the amount of coordination performed in the controller processors. Instead of idling the redundantly assigned weapons as in level 2, these weapons are made available for assignment to other targets not yet assigned. The iterative process of reallocation continues until all weapons within a control group have been assigned to unique targets. Note that redundancies can still exist between control groups. Thus, this implementation, like the first two, will not generate the optimum solution.

The fourth implementation: parallel matrix operations The "level 4" implementation is a different approach from the first three parallel implementations. Instead of replicating the serial code in several nodes, certain time consuming operations of the B&L algorithm were implemented in parallel on multiple processors.

In the three previous parallel implementations, there is a problem with the redundant assignment of weapons to the same target. In this implementation, the problem is eliminated by fully coordinating the assignment process. During a parallel assignment iteration, each processor makes assignments on a different *independent* windows of the cost matrix. The term independent means that the targets being considered for assignment are not being considered by any other processor during the present iteration. After each iteration, the individual assignment contributions from each processor are used to update global variables that are then broadcast to each processor and the next set of independent windows are searched for possible assignments. The algorithm terminates when, as in the sequential B&L algorithm, all weapons have been assigned. The assignment solution produced by this implementation will be the minimum-cost optimal assignment.

4. RESULTS

The four parallel implementations of the assignment algorithm were evaluated for target-to-weapon ratios of 1:1, 5:1, and 10:1. These ratios are representative of the SDI environment during the early boost-phase of an engagement. For each ratio, the number of weapons was chosen to range from 32 to 128. The performance of each parallel implementation was evaluated for each combination of number of weapons to be assigned and target-to-weapon ratio. The results of the evaluations, discussed below, were shown to be statistically valid by replicating the evaluations over a large set of cost matrices.

The computation times varied widely between the different implementations, as shown in Table 1 and Figure 3 (for 96 weapons and 960 targets). As a benchmark for comparison, the time required to reach the optimal solution on a single processor is given by the level 1, one processor case in the table. The fastest computation times were consistently those of the level 1 and the level 2 programs, while the slower times were where those of the level 3 and level 4 programs. This difference in performance is primarily attributed to the volume and frequency of communications between processors in the different implementations. One drawback of the faster solutions is that they are not optimal because of redundancies, weapons idled, and reassignments to other targets performed by the different implementations. However, in most cases, the advantage of fast processing times allows many iterations of sub-optimal assignments to be computed in the time required to compute only one optimal solution. In the level 3 and 4 implementations, the cost of doing large volumes of slow interprocessor communications dominates the actual computation time within the processors. As the level of coordination and communications increases, the average computation times also increases but at much slower rate. As a result, if the interprocessor communications delays could be significantly reduced, as predicted for the iPSC V2, implementations of levels 3 and 4 might become more useful.

Although processing times and speedups were the main measures of performance emphasized, the manner in which the available weapons are utilized is also very important. If an algorithm is extremely fast but yields poor weapons utilization, it will not be very useful. Weapon effectiveness is defined as the percentage of weapons assigned to a unique target. All of the implementations produced weapon effectiveness above 80% for the 10:1 and 5:1 weapon-to-target ratios. An example of the results for a 10:1 target-to-weapon ratio is shown in Table 2. The best overall assignment performance, in terms of computational speed and weapons effectiveness, was obtained with the level 2 implementation. In most all 5:1 and 10:1 ratio cases, it wasted less than 10% of the weapons. In general, the level 2 program idled more weapons than it wasted while yielding effectiveness percentages comparable to the other implementations. The idling of weapons rather than wasting them is important, especially when weapons are scarce. Idled weapons can be withheld until a later assignment iteration when they may be utilized in a more cost effective manner.

5. CONCLUSIONS

Achieving fast and efficient results from a parallel processing system appears to rely on three fundamental rules: (1) The problem must be partitionable into a number of independent subproblems. (2) The communications between the processing elements must be kept to a minimum. (3) The computations performed by each processor must be approximately equal and simultaneous.

This paper has presented four parallel implementations of resource allocation algorithms and evaluated their usefulness in an SDI-oriented environment. The first three implementations partitioned the overall cost matrix into independent submatrices and solved each on independent processors. The differences in the implementations focused on the level of interprocessor communications that was used to identify redundant assignments and, possible, reallocate the duplicates. The fourth implementation decomposed the serial B&L algorithm for execution on a parallel processor. The factor that limited this implementation's performance was the long delays incurred during periods of interprocessor communications. Examining the four implementations in light of the tradeoff between the computational speed and the effectiveness of the resultant assignments, level 2, which involved a modest amount of coordination and communication, produced the best overall performance. The approach used in level 2 would easily map onto a geographically distributed SDI architecture, if such a system were to be deployed.

References

[BoL71a] Bourgeois, L. and J. Lassalle. "An Extension of the Munkres Algorithm for the Assignment Problem to Rectangular Matrices," *Communications of the ACM,* 14: pp. 802-804 (December 1971).

[BoL71b] ——. "Algorithm 415: Algorithm for the Assignment Problem (Rectangular Matrices)[H]" *Communications of the ACM,* 14: pp. 805-806 (December 1971).

[Chu57] Churchman, C.W., R.L. Ackoff, and E.L. Arnoff. *Introduction to Operations Research,* New York: John Wiley and Sons Incorporated, 1957.

[Cve87] Cvetanovic, Z. "The Effects of Problem Partitioning, Allocation, and Granularity on the Performance of Multiple-Processor Systems," *IEEE Transactions on Computers,* C-36: pp. 421-432 (April 1987).

[Kuh55] Kuhn, H.W. "The Hungarian Method for the Assignment Problem," *Naval Research Logistics Quarterly,* 2: pp. 83-97 (1955).

[Kur62] Kurtzberg, J. M. "On Approximation Methods for the Assignment Problem," *Journal of the ACM,* 9: pp. 419-439 (1962).

[Odo85] Odom, P. *A Method for Improving Technology Research and Development Decisions Regarding BMD and ASAT, Volume III - User Manual and Simulation Description: Final Report,* Contract DAAH01-84-C-0486. Huntsville, Alabama: DESE Research and Engineering, Incorporated, July 1985 (AD-B093915).

[SeD85] Seward, W.D., and N.J. Davis IV. "Opportunities and Issues for Parallel Processing in SDI Battle Management/C3," Presented at the AIAA Computers in Aerospace V Conference, October 1985.

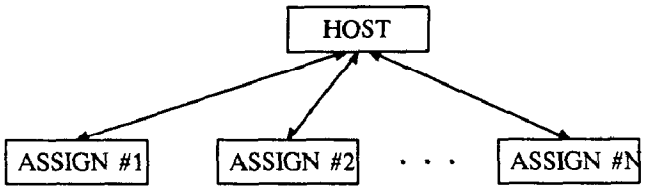

Figure 1. Processor Communication Paths for the first Level

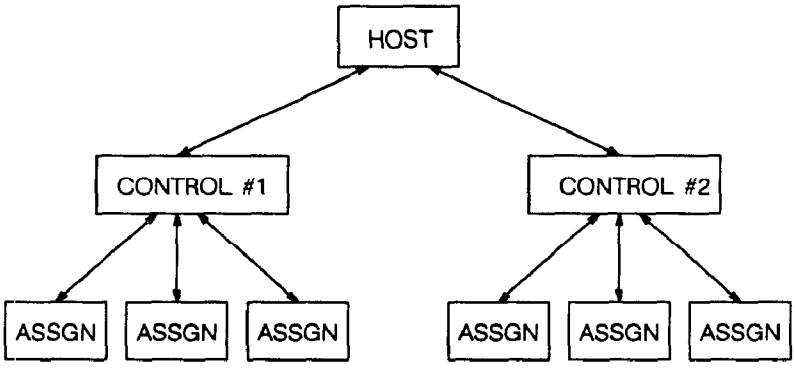

Figure 2. Communications paths for levels 2 and 3

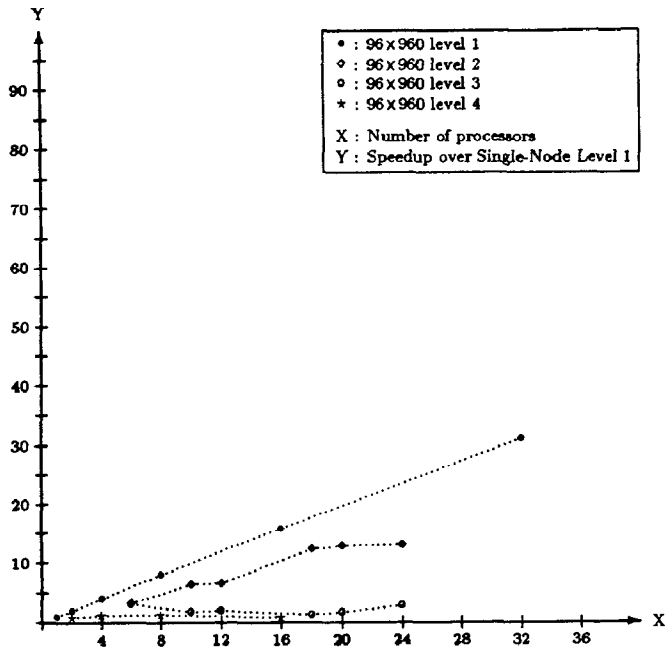

Figure 3. Speedup over a single-node level one implementation

Table 1: Processing Times of the Parallel Implementations

Level	Weapons	Targets	Cntrl	Proc/Cntrl	Tot Proc	Time (sec)
1	96	960	–	–	1	15.0570
1	96	960	–	–	2	7.5195
1	96	960	–	–	4	3.7635
1	96	960	–	–	8	1.8891
1	96	960	–	–	16	0.9518
1	96	960	–	–	32	0.4836
2	96	960	2	2	6	4.615
2	96	960	2	4	10	2.359
2	96	960	2	8	18	1.228
2	96	960	4	2	12	2.305
2	96	960	4	4	20	1.176
2	96	960	8	2	24	1.163
3	96	960	2	2	6	4.8910
3	96	960	2	4	10	8.4135
3	96	960	2	8	18	11.6000
3	96	960	4	2	12	7.6133
3	96	960	4	4	20	9.0953
3	96	960	8	2	24	5.2775
4	96	960	–	–	2	17.522
4	96	960	–	–	4	12.876
4	96	960	–	–	8	11.898
4	96	960	–	–	16	18.001

Table 2: Assignment Results of the Parallel Implementations

Level	Weapons	Targets	Cntrl	Proc/Cntrl	Tot Num Proc	% Effective	% Idle	% Wasted
1	96	960	–	–	1	100.0	–	0.0
1	96	960	–	–	2	92.1	–	7.9
1	96	960	–	–	4	87.7	–	12.3
1	96	960	–	–	8	84.6	–	15.4
1	96	960	–	–	16	83.3	–	16.7
1	96	960	–	–	32	82.5	–	17.5
2	96	960	2	2	6	87.7	4.8	7.5
2	96	960	2	4	10	84.6	7.9	7.5
2	96	960	2	8	18	83.3	9.4	7.3
2	96	960	4	2	12	84.6	3.1	12.3
2	96	960	4	4	20	83.3	4.6	12.1
2	96	960	8	2	24	83.3	1.5	15.2
3	96	960	2	2	6	89.8	–	10.2
3	96	960	2	4	10	88.1	–	11.9
3	96	960	2	8	18	87.3	–	12.7
3	96	960	4	2	12	86.3	–	13.7
3	96	960	4	4	20	85.2	–	14.8
3	96	960	8	2	24	84.0	–	16.0
4	96	960	–	–	2	100.0	–	–
4	96	960	–	–	4	100.0	–	–
4	96	960	–	–	8	100.0	–	–
4	96	960	–	–	16	100.0	–	–
4	96	960	–	–	32	100.0	–	–