Predictable Real-Time Scheduling for Air Traffic Control

Will Meilander, Johnnie Baker, Mingxian Jin
Department of Computer Science
Kent State University, Kent OH 44242
{willcm, jbaker, mjin}@cs.kent.edu

Abstract

A different approach for real-time Command and Control problems is presented, using the Air Traffic Control problem as an example. Current ATC approaches use "dynamic" scheduling algorithms, which by their very nature are unpredictable. The current ATC approach is extremely expensive. The April 19, 1999 issue of USA Today places the cost at $41 billion, and the system is not meeting current specifications. The current problems with ATC are predicted by theoretical results in Real-Time Scheduling. M. Klein et al state "an efficient real-time multiprocessor scheduling algorithm is not expected" [1]. J. Stankovic, [5] indicates that a polynomial time multiprocessor schedule for a C&C problem like ATC is unlikely. Our approach uses synchronous set processing of jobs in this real-time database problem and embodies the qualities of speed, predictability, adaptability and reliability. We show that an associative SIMD processor (AP) using synchronous set processing can complete all ATC tasks before their deadline.

Introduction

We're served a bowl of consommé. A delicious consommé is hard to eat with our fingers. We ask for silverware and we are served a fork. A fork won't do; we need a better tool! How about a spoon? Aha! Now we can enjoy our consommé. In Air Traffic Management (ATM) we need a new tool. With the tool we've been using data keeps slipping through our multiprocessors tines. And then, too many aircraft slip through the cracks. Hi-jacked aircraft disappear from the controllers' radar screens when the aircraft's "beacon" is turned off.

Another example: [7] a headline proclaims, "Air Force One Radar Glitch is Typical" with the subtitle, "Crowded skies cause planes to briefly disappear." Many people think it's a radar problem. It is not! The radar data that could support Air Force One on the controller's screen existed during the whole time it "was missing." The problem is the tool that processes the radar data. Too much of the data is slipping through the tines of the multiprocessor because the data cannot be usefully processed. It has to be thrown away because it doesn't pass through the "data rejection filter". Does anyone care? The controllers care! The news item continues; "Controllers said as many as 50 aircraft are lost each day for up to one minute." And later, from the same news item: "In a subsequent statement, the agency (*FAA*) said a similar problem occurred six minutes later, when the aircraft (*Air Force One*) had traveled about 42 miles to the northeast."

Let's look at the facts. No one can deny that the controllers did not see the hi-jacked aircraft. No one can deny that the controllers did not see Air Force One. No one can deny that the hi-jacked aircraft and Air Force One were certainly "seen" by several radars in the area. Further the radar site that was being used to follow the "disappearing" aircraft was actually providing two distinct types of radar data. The first type, called "primary" data, that could easily have followed the planes was discarded. The other type, called "secondary" radar, was used and as mentioned above (i.e., "…as many as 50 aircraft are lost each day for up to one minute.") it regularly loses data. The "data rejection filter" regularly rejects the more accurate and more reliable primary radar data because the ATM data processor is incapable of properly processing it. The ATM multiprocessor is an inadequate tool. Would you want to eat soup with a fork? Should our ATM system use an inadequate tool?

System improvement

The associative processor (AP) is a proven alternative. It works because it can synchronously process a set of jobs. In particular, it can process thousands of items of data with one instruction while the conventional computer processes a single item using a more complex instruction. The AP has been demonstrated in an ATM environment, and has been in use, since 1983, in a highly demanding military environment automatically tracking up to 1,750 targets based on primary radar data. This performance was achieved in a processor with 450-nanosecond memory using only 7.6% of radar scan time. That is considerably more primary radar traffic than most of the ATM systems in our Nation can handle today.

The AP can be characterized as an associative SIMD. It consists of an array of processing elements (PEs) connected by a bus and an instruction stream processor (IS) that broadcasts commands and data to all of the PEs on the bus. Each PE has access to an addressable segment of common memory and can perform all the usual local operations of a sequential processor other than issuing instructions. Each PE can be active or inactive, based on the result of a data test. An active PE executes the instructions issued by the IS, while an inactive PE listens to but does not execute the instructions. Assuming that the word length is a constant, the AP supports several important constant time operations, namely broadcasting, global OR/AND, maximum/minimum, and associative search which identifies the cells whose data values match the search pattern (called responders) or do not match (called non-responders). Two parallel architectures that fully support the AP (in hardware) are the STARAN and the ASPRO. An extended description that supports multiple ISs is given in [6] and further information about the properties of AP can be inferred from [6, 9].

Past multiprocessor efforts

How can we bring this new paradigm to the attention of the primary buyer – The FAA. But more to the point, many scientists when studying scheduling of multiprocessors in problems of the ATM type have shown scheduling problems using multiprocessor systems to be intractable [1, 2, 3, 5, 8]. The exponential nature of these solutions has been borne out by the many failed attempts to develop an ATC/ATM system around some multiplicity of conventional processors. The variations on that theme are many. They have all failed to meet the system requirements. Some like DABS/IPC appear to have been abandoned. Others like the original (1963) enroute requirement have had the requirement modified so that the system could be accepted.

The Advanced Automatic Air Traffic Control System known as AAS started differently. Two "proof of performance" development contracts were awarded to initiate AAS. After expenditure of about 1 billion dollars the contracts were halted. This failure is not surprising, due to the exponential nature of the solutions being proposed. Despite the demonstrated difficulty in meeting the proof of performance requirements, a production contract was awarded. After the expenditure of many billions of dollars in fruitless production effort, AAS was canceled in June 1994.

Many researchers have shown that multiprocessor scheduling problems that are typically solved as part of the ATM problem are NP-hard [1, 2, 3, 5, 8, 11]. These theoretical results and the repeated failure of successive air traffic control systems seem to have led to the almost universal belief that the air traffic management problem is intractable. In contrast, we believe that there is an easily predictable, static, polynomial-time schedule for the ATM problem and that the ATM problem itself can be solved in polynomial time. The alternate approach that provides the basis for this claim is examined in the next section.

ATM environment and problems

The main aspects of the ATM problem will be discussed next. The problem involves a large number, up to 14,000, aircraft flying within, or adjacent to (for backup), a region of air space. These aircraft are flying between a known number of terminals. They often have to contend with inclement weather. Each aircraft is of a known type, having defined characteristics. Many of this set of aircraft (4,000) are under control and have a flight plan identifying a path and conditions of flight they plan to follow through the airspace. The entire set of aircraft is under observation by a number of sensors that are continually reporting the positions of aircraft observed by each sensor. We will assume that the number of sensor reports may reach a peak of 12,000 reports per second. The number of aircraft and reports are considered to be a worst-case situation.

A relational database

The ATM problem is a database management problem, and can be readily represented as a relational database. We present a number of tasks that are required and some that are suggested for future ATM use.

1. Report correlation. The most difficult problem in ATM is that of accepting the set of reported sensor observations and developing an optimal track for each aircraft from the data received [1,2]. Each track should be supported by at least two sensors to approach optimal performance and achieve fail-safe reliability. The set of 14,000 developed tracks that have been updated to _ second less than current time is joined with the set of 6,000 reports that have been received during the previous half-second. This join process, as can be seen when a set processor is used, is an order O(r) task where r is the number of reports received during the last half second. In particular, each report of the set of reports can be broadcast and simultaneously compared with every track in the set of tracks in one constant time operation in the associative processor. Since the associative processor is assumed to have at least 14,000 PEs, each track will be stored for access by a different PE. Therefore this operation is _(r) in the AP we suggest, and its running time on this task is completely independent of the number of tracks being maintained by the ATM system. Handling 12,000 of these join operations per second is a simple task for an AP.

With a multiprocessor performing the equivalent *join* function, the minimal number of operations (without sorting) is O(r * t), which would require a minimum of $168 * 10^6$ operations per second for our worst-case environment. When the considerable set of omnipresent data management functions such as sorting, indexing, processor assignment, data movement, interprocessor communication, record locking and lock management, concurrency management, serializability control, etc. is considered, the algorithm for this join could conceivably push the problem into intractability of the first and second kind mentioned above. The above requirements are at least difficult and likely impossible for present ATM systems to meet. However, the AP we suggest can easily satisfy the requirement in about 8% of the available time.

2. **Smoothing and prediction of tracks.** After the tracks have been correlated with the set of reports they are smoothed to minimize the noise introduced by the errors present in the reports, and the next half-second position is predicted. Since the same algorithm is executed simultaneously for all tracks and is executed only once, the entire operation requires only constant time, as shown in [4]. A typical MP iterative solution is O(n). (In [4] we show measured performance improvement of about 27,500 %[1] over the Navy's dual processor in the E2C airborne early warning system, based on the 1979 ASPRO design.)

3. Conflict detection. Another major task in ATM is conflict detection. This process starts with the development of a future path envelope for each track in the system. Then each controlled track envelope is modified to add a minimum separation distance to its future path envelope. This controlled track envelope is then evaluated for intersection with any other aircraft envelope in space and time to assure conflict free operation for a twenty-minute "look ahead" interval. This process is carried out until all controlled flights have been compared with all flights. Detected conflicts are "flagged" to the controller and a conflict resolution maneuver may be determined and suggested to resolve the conflict.

In the AP, each projected controlled track is tested against the set of remaining tracks. Since this is a set operation, each of these tests takes constant time. The time required for the entire operation is O(c) where c is the number of controlled tracks. (We observed earlier the number of tracks might be substantially larger than the controlled tracks c because of uncontrolled tracks u.) Nevertheless, the operation time in the set processor is independent of the total tracks, c + u, and is dependent only c. On the other hand, a typical multiprocessor solution requires $O(c^2 + u*c)$ operations. This occurs because the process of comparing each future envelope of the controlled tracks with every other flight envelope is iterative over the entire set of tracks. When the omnipresent data management functions such as processor assignment, data movement, interprocessor communication, locking and lock management, concurrency management, serializability, etc. are added, the running time will be increased considerably.

4. Flight plan update and conformance. The flight plan update and conformance task updates each flight plan and compares each controlled track with its matching flight plan. In a relational database context it is readily seen that a 1:1 relationship exists between flight plan and track. Since each flight plan and track is part of the same record in an associative processor and records are accessible by each PE, the single execution of a program can simultaneously evaluate all flight plans and tracks for conformance in constant time. This operation over the entire set can be completed in less than seven microseconds (assuming an AP based on current technology). This job will probably be included in the conflict detection task. This operation is typically O(c) in a multiprocessor. Unfortunately, the necessary ubiquitous overhead data management functions will substantially increase this running time.

Other ATM functions

A number of other ATM functions have similar characteristics to those presented above. By way of example we list some of them:

[1] 27,500% is not a misprint.

5. Terrain avoidance. This is the process of evaluating each flight to determine if it may encounter terrain in the next k minutes, where k is a “look ahead” time. If this is conducted for all flights, as is reasonable at some near-future time, the process is O(c+u) since each one the worst-case set of tracks is evaluated for a large set ter of terrain features. In a multiprocessor this operation typically is O((c+u) * ter) (let’s handle the possible exponential increase in a section following this one, so we don’t have to keep repeating ourselves on statements involving intractability.)

6. Display data processing. This is the task of delivering data to the controllers and managers displays and is O(c+u). We believe the process is O(c+u) in the multiprocessor, and can be handled in a similar manner to that used in The AP.

7. Data insert. Inserting new data is O(d) where d is the amount of data being entered. This can include input from sensors, pilots, controllers, other ATM facilities, weather data, terminal conditions, etc. We believe the process is O(d) in the multiprocessor, under the same assumptions we use in the set processor.

8. ATM backup. ATM backup is the process of maintaining a duplicate set of the relations making up the database for use in emergency conditions. In an associative processor, this is readily achieved by providing a duplicate processor, and operating each in simultaneity. (We will not try to suggest an estimate for the multiprocessor.)

9. Cockpit display. Cockpit display processing would develop the set of tracks pertinent to a specific flight, convert them relative to flight heading and transmit the data to the airborne cockpit display. This process is O(p) in the AP where p is the number of pilot displays being processed.

10. Voice advisory. A similar function could automatically advise, by voice, an uncontrolled flight of near term conditions of other aircraft and terrain. Its time would be the order of the number of uncontrolled flights being handled.

11. Runway optimization. Runway optimization is the process of recognizing flights scheduled for a particular runway and ordering the set for each runway to optimize runway use. Of course, such optimization must include runway departures.

Items 1 through 11 can be shown to be executed in the AP in polynomial time for the worst-case ATM parameters presented above**.** Items 1, 2, 3, 5, 7 and 10 were demonstrated in the first AP installation in the Knoxville terminal in 1971. The initial requirements were for automatic track initiation, conflict detection, and conflict resolution. That program was developed and demonstrated in about six months. A program for automatic voice advisory (AVA) of approaching traffic for VFR flights was added. AVA was completed in about three months. Approximately 75 pilots participated in AVA. Another requirement, that of terrain avoidance was added and completed in about four months. The total AP programming and support effort was about four man-years and yielded 2,200 instructions. It was clear that additional tasks could be added easily in the AP. Current ATM systems are not doing what was done using an AP at Knoxville in 1971.

An associative processor static schedule for ATM

We develop a static schedule that uses a single, sequential, nonpremptible instruction stream thus making it possible to calculate the time for each task, and sum the time for all tasks. The algorithm can be evaluated for the worst-case problem conditions to determine whether the problem is solved before deadline time *D.* Since the solution time can be determined it is seen that the solution is polynomial.

Garey and Johnson show a similar condition in their book [8, SS11], which concludes, for a multiprocessor that if each task has a release time and a deadline time, that the problem can be solved in polynomial time *if all tasks have no more than one immediate successor.* Most real-time multiprocessor systems using dynamic scheduling will be unable to define each tasks release time in such a way that assures all tasks *have no more than one immediate successor*. However, all tasks in the AP have only *one immediate successor.* All AP tasks also have only one immediate predecessor (see Figure 1 below).

We show a schedule and timing estimates for the worst-case set of conditions based on an AP designed using the current state of the art.

Reports per second	12,000
IFR flights	4,000
VFR/backup flight	10,000
Controllers	600

Table 1. ATC Tasks – Worst Case Environment

Task	p	$j*10^{-6}$	C	DL	Proc.Time
1. Report Correlation & Tracking	.5	15	.09	.10	1.44
2. Cockpit Display 750 /sec)	1.0	120	.09	.20	.72
3. Controller Display Update (7500/sec)	1.0	12	.09	.30	.72
4. Aperiodic Requests (200 /sec)	1.0	250	.05	.36	.4
5. Automatic Voice Advisory (600 /sec)	4.0	75	.18	.78	.36
6. Terrain Avoidance	8.0	40	.32	2.93	.32
7. Conflict Detection & Resolution	8.0	60	.36	3.97	.36
8. Final Approach (100 runways)	8.0	33	.2	6.81	.2
Summation of Tasks	Sec/8 sec				4.52

P is Time in an 8 second period in which all tasks must be completed,
$p_{i+1} = p_i + p$ the next task release time,
j is the execution time for each job in a task, where each task is a set of jobs

C is the cost for each task for the worst-case set of jobs,
DL the deadline time for each task p + c + 10 (includes 10 msec interrupt processing per task)

Table 2. Statically Scheduled Solution Time

Figure 1 shows the execution timing for the tasks presented in Table 2. Execution starts at time 0 for "start" and continues a new major period P at time 16.

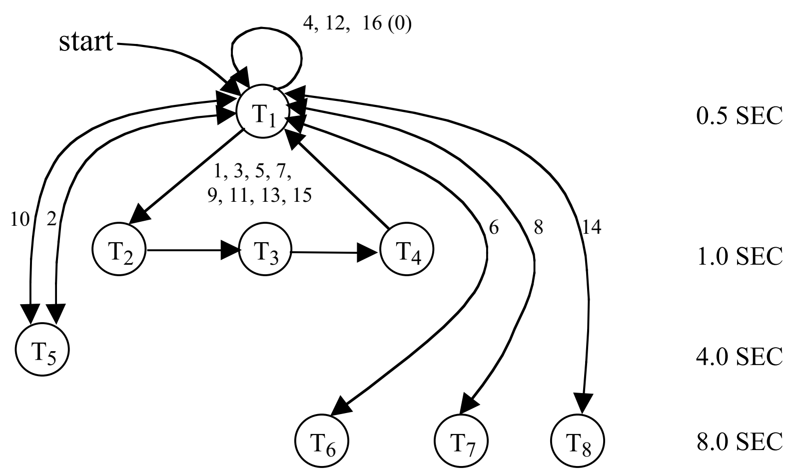

Figure 1. ATC schedule

Difficulties with multiprocessors in real-time systems

The following real time data management requirements exist in most real-time multiprocessor systems but are virtually non-existent in real-time AP systems. None of these processes do any data processing. In fact, it is essential to ascertain that no changes are made to data when these data management functions are used.

- dynamic scheduling software;
- task assignment software;
- queue management software;
- data assignment software;
- concurrency control software;
- coherency management software
- table and data locking software;
- cache management software;
- task predictability evaluation software;
- multitasking software;
- indexing software;
- sorting software;

- mosaicing software;
- data pointer assignment and update software;
- and iterative processing of a set of data.

Even more important! The AP also eliminates the processing time needed to execute the above multiprocessor data management software.

We include several results that further describe the difficulty of multiprocessors in real-time applications.

- The problem of finding a partitioning of tasks to processors so that all processors can meet a fixed deadline D that is NP-complete [8, SS8].
- "The scheduling of accesses to resources other than the CPU is a particularly difficult problem in the context of real-time scheduling. In the presence of shared resources, the complexity of feasibility analysis becomes exponential. Specially, the problem of deciding whether a set of periodic tasks is feasibly schedulable when semaphores are used to enforce mutual exclusion is NP-hard." [12, pp. 121]
- "Some researchers in control theory have drawn the conclusion that *computer systems are inherently probabilistic in terms of their timing behavior."* [13]
- "…the use of contemporary computers inhibits real-time control."[13]
- "One of the goals of hard real-time scheduling is to find feasible schedules for a multiprocessor system. It is a difficult problem and has been shown to be NP-complete even for simple models."[15]
- "Clearly, what we are striving for is a scheduling algorithm that is able to find a feasible schedule for a set of tasks, if such a schedule exists. Obviously, a heuristic algorithm cannot be guaranteed to achieve this."[15]

Summary

We have attempted to show that problems like the Nation's Air Traffic Control are amenable to solution when an AP can be applied to the problem. We realize it is hard for people to accept this in the face of the preponderance of problem intractability in a multiprocessor configuration. We offer the following in our attempt to clarify the reasons why a multiprocessor problem can be NP-hard while the AP offers problem tractability.

J. Stankovic writes [2], "*A highly integrated and time-constrained resource allocation approach is necessary to adequately address timing constraints, predictability, adaptability, correctness, safety and fault tolerance. For a task to meet its deadline resources must be available in time and events must be ordered to meet precedence constraints. Many coordinated actions are necessary for this type of processing to be accomplished on time. The state of the art lacks completely effective solutions to this problem*." The AP approaches an effective solution for problems in the class of real-time database requirements.

References:

1. M.H. Klein, J.P. Lehoczky, and R. Rakumar, "Rate-Monotonic Analysis for Real-Time Industrial Computing", IEEE Computer, pp. 24-33, 1994.
2. J. A. Stankovic, "Real-Time and Embedded Systems," The Computer Science and Engineering Handbook, Ed. Allen B. Tucker, Jr., CRC Press, 1997.
3. M. R. Garey, R. L. Graham and D. S. Johnson "Performance Guarantees for Scheduling Algorithms", Operations Research, Vol 26, No.1 Jan-Feb, 1978.
4. Will C. Meilander, Jerry Potter, Kathy Lizcka, Johnnie Baker " Real-Time Scheduling in Command and Control", MWPP workshop, Aug., 1999, http://www.mcs. kent.edu/~ parallel.
5. J. A. Stankovic, M. Spuri, M. Di Natale, G. C. Buttazzo, "Implications of Classical Scheduling Results for Read-Time Systems"|, Computer June, 1995, pp. 16-25.
6. J. L. Potter, J.W. Baker, S. Scott, A. Bansal, C. Leanguksun, and C. Asthagiri, "ASC: An Associative. Computing Paradigm, Computer", 27(11), 1994.
7. Predictability for Real-Time Command and Control Will C. Meilander, Johnnie W. Baker, Jerry Potter IPDPS 2001, IEEE Electronic Library, unofficial version at http://www.mcs.kent.edu/~parallel
8. M. R. Garey and D. S. Johnson, *Computers and Intractability: a Guide to the Theory of NP-completeness*, W.H. Freeman, New York, 1979, pp.65, pp. 238-240
9. M. Jin, J.W. Baker W. Meilander, The Power of SIMDs vs MIMDs in Real-Time Scheduling for MPP workshop at IPDPS'02, http://www.mcs.kent.edu/~ parallel.

10. A recent study of ATC flight plan to track association [4] shows the ratio of memory access for the current multiprocessors versus the AP approach greater than 90 to 1. A worst case ATC problem is tractable in an AP, but is generally considered NP-hard due to a number of scheduling limitations when the use of multiprocessors are required.
11. C. Murthy and G. Manimaran, *Resource Management in Real-Time Systems and Networks*, The MIT Press, Cambridge, Massachusetts, 2001
12. J. Stankovic, M. Spuri, K. Ramamtitham and G. Buttazzo, *Deadlines Scheduling for Real-time Systems*, Kluwer Academic Publisher, 1998
13. Martin Torngren, "Fundamentals of Implementing Real-Time Control Applications in Distributed Computing", Real-Time Systems, 14, pp.219-250, 1998, Kluwer Publisher
14. Fuxing Wang Krithi Ramamritham John A. Stankovic, Bounds on the Performance of Heuristic Algorithms for Multiprocessor Scheduling of Hard Real-Time Tasks, Proceedings of Real-time Systems Symposium, 1992, pp.136-145.