Practical Parallel Simulation Applied to Aviation Modeling

Dr. Frederick Wieland
Center for Advanced Aviation Systems Development
The MITRE Corporation
1820 Dolley Madison Dr., McLean, VA 22102
fwieland@mitre.org

Abstract

This paper analyzes the Detailed Policy Assessment Tool (DPAT) as an example of a practical real-world aviation simulation that uses optimistic simulation technology. We present a review of analyses that have used DPAT results to support their conclusions, and discuss the design and performance of the system in the context of parallel simulation. The thrust of this paper is to explain how DPAT avoids some of the problems associated with optimistic simulation, while exploiting its strengths. A key conclusion is that parallel simulation technology, particularly optimistic synchronization, needs to be built into the product from its conceptual design through implementation, as opposed to adding it afterwards.

1. Introduction and Background

Designing parallel simulations that are optimistically synchronized is more of an art than a science. While there are some heuristics for good performance, such as maximizing granularity and lookahead while minimizing the coupling between logical processes (LPs), combining these heuristics together with a useful model to produce simulations that are both practical and well-performing has rarely been done. Compounding the problem is the myriad of difficulties that surround optimistic simulation that are absent in sequential systems, such as managing concurrent threads of execution, verifying correct execution, and handling risky events that may cause severe difficulties. Despite these apparent difficulties, the MITRE Corporation has developed a useful and high performing optimistically synchronized parallel simulation for the analysis of air traffic control. The simulation, called the Detailed Policy Assessment Tool (DPAT), is fully optimistic. There is no throttling of optimism, and yet it is devoid of rollback explosions, cascading rollbacks, and other problems that have been widely publicized in the parallel simulation literature.

That DPAT has had a significant and positive contribution on the aviation community is unassailable. Over twenty analysis projects used DPAT during the last four years, some of them reported in the popular press. An analysis of the proliferation of regional jets and their impact on airspace and airport congestion using DPAT can be found in the magazine "Air Traffic Management"; the remarkable accuracy of the predictions from this study have been lauded by industry leaders [1]. Prediction of future airport bottlenecks in the Asia-Pacific region using DPAT is outlined in a publicly available report [2]. The United States Federal Aviation Administration (FAA) has itself performed a detailed validation study of DPAT [3]. The results show that the model produces credible answers that are consistent with actual data collected from FAA's own systems. At the time of this writing, they are continuing to use the model for infrastructure investment analysis. Additionally, DPAT results have been used for testimony before the United States Congress [4].

While these reports are all publicly available, there have been many more proprietary studies using DPAT. The model has been used to study the impact of reducing required airplane separation, and its impact on system wide throughputs and delays [5]. It has also been used to study the tradeoffs between enroute and terminal-area efficiency improvements [6]. One of the most impressive non-public studies simulated each of the 365 days in 1997, with actual traffic and weather conditions, *twice*, one time with a proposed avionics system and once without. The analysts used the results from the 730 model runs to determine the annual cost/benefit of the proposed avionics improvement [7].

DPAT has been the subject of papers in recent years on its internal representation of the air traffic control system as a queueing network [8], and on its use for system wide analysis [9]. Although these papers mention DPAT's use as a parallel simulation, there are a number of important questions that remain unanswered. How can an optimistically synchronized simulation produce such a great impact on the real world? Are there not major problems with such simulations, including runaway rollback explosions, cascading antimessage chains, risky processing leading to erroneous states that crash the

0-7695-1104-X/01 $10.00 © 2001 IEEE

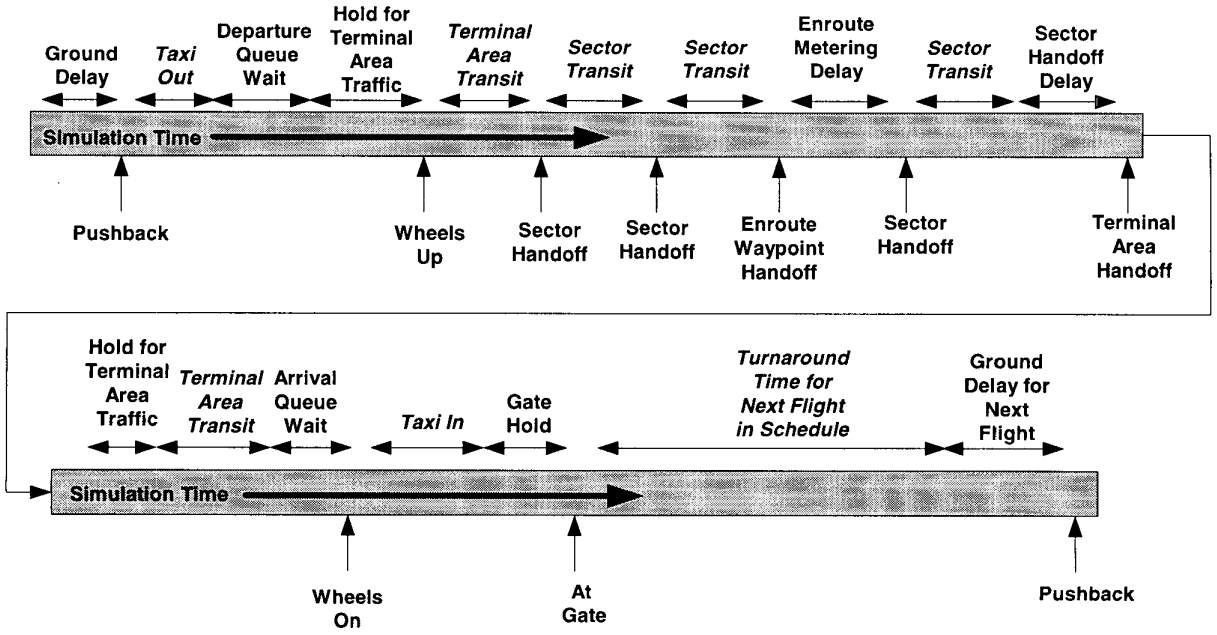

Figure 1. The modeled world. Time starts in upper left and flows to the right.

system? How are these problems avoided, and how do aviation analysts react to a system that processes both forwards and backwards? The remainder of this paper focuses on how the technical problems with parallel simulation are avoided, and the conceptual model exploited, to produce a good simulation.

2. Design Heuristics

Building a successful parallel simulation—whether conservative or optimistic—requires a creative blend of technical knowledge of parallel simulation and domain acumen about the physical system under study. Technical knowledge without any domain acumen produces "toy" simulations; domain acumen without any technical knowledge produces poorly performing, frequently crashing systems that are neither scalable nor useful for basic analysis.

On the technical side, it is well known that a number of factors contribute to good performance. Proper event granularity and good lookahead are both necessary ingredients. "Proper" event granularity is difficult to quantify, as it depends upon hardware characteristics, network performance, and whether an SMP or distributed system is employed. Studies of lookahead have shown that it influences greatly the probability of rollbacks and therefore the efficiency of the simulation engine. Generally, the greater the lookahead the greater the efficiency.

But there are other, more subtle, performance parameters that must also be considered. Most notably, computations in the model need to be "smoothed out" in three dimensions: simulation time, real time, and space (memory). In simulation time, it is well known that zero lookahead is to be avoided. Additionally, major computations should be staggered in simulation time. Instead of one event computing everything, the computation should be staggered among several events at different times. This can often be accomplished without compromising the integrity of the model, by merely changing its implementation. In real time, event explosions should be eliminated. An event explosion occurs when the minimum number of events that are needed to advance GVT by one time unit is very large. Finally, in space (i.e. memory), computations need to be as independent as possible, so that LPs do not need much knowledge of other LPs' states in order to compute an event.

3. The National Airspace System (NAS)

At its peak, the United States air traffic control (ATC) system currently handles 100,000 passengers per hour on 4,000 aircraft, or about 650 million passengers per year [10]. The volume is increasing at least as fast as the

general economy, and yet the number of airports and runways are increasing more slowly, and the volume of airspace is (obviously) static. The result can be unusually large delays when the system is stressed. About 70% of delays are due to bad weather; the remainder occurs for a variety of other factors.

DPAT is designed to predict system-wide delays resulting from congestion, weather-related problems, system outages, growth in air traffic volume, or any combination of these. It is built to handle worldwide air traffic, taking as input the airports, airspace, flights, routes, and system capacities that constitute the ATC system under study. The data can represent past historical days, the current situation, or future hypothesized scenarios. The current situation can be input from live data feeds provided by ATC authorities; for historical and future information, DPAT reads data files in a well-documented format. The DPAT User's Manual [11] explains the capabilities of the model as well as the input and output data provided.

4. The Modeled World

DPAT represents the ATC system as a network of queues. Flights acquire and release a predictable set of resources as they travel gate-to-gate from their origin to their destination. The resources can be gates, taxiways, runways, blocks of airspace, waypoints in space called "fixes," and so forth, and the set of resources modeled by DPAT is shown in the time diagram of Figure 1. In that figure, simulation time is advancing from left to right. Below the gray bar, discrete events are shown, while above the gray bar, intervals representing activities occurring during a flight are shown. The intervals listed in *italic font* represent quantities that DPAT reads directly from a data file. Examples are the taxi in and out times, as well as minimum transit times in terminal areas and sectors. These times are always nonzero. Intervals listed in normal font are computed by DPAT as a result of congestion; examples include the various delays and holding patterns that occur during a flight. These times may be zero, depending upon congestion. The particular timeline shown in Figure 1 is for illustrative purposes. Some flights in an actual DPAT scenario may have as many as twenty (or more) enroute sectors, others might have no computed delay anywhere along their timeline, and still others may have excessive delays in some areas and none in others.

While the set of resources required by a flight is predictable, the interaction of these flights with other flights is unpredictable, and may result in delays. The resources themselves are the servers in the queueing system. Some of the resource capacities are specified as service rates, while others are specified as maximum queue lengths or a combination of queue length limits and the number of input queues. These rates can be a time varying function of random variables such as the weather, system outages, and closures. Flow control mechanisms can be applied on the ground or en-route, further reducing service rates and increasing delays. Many of the service rate changes are input to the model, while some are derived by DPAT itself and applied as the simulation evolves.

The net effect of these factors causes the queueing network to be analytically intractable. The most expedient way to analyze the system is to simulate actual flights traversing the network of resources, and to compute the resulting throughputs and delay at each point. It is this computation that is realized by the DPAT model.

5. Parallel Decomposition

Central to any parallel simulation system is the decomposition of the model into logical processes (LPs) that interact in parallel. Criteria for making this decision involve considerations of LP independence and scheduling overhead. It is desirable for the parallel LPs to be as independent as possible; this maximizes the amount of parallel computation and minimizes communication and synchronization costs. There is also a tradeoff between parallelism and scheduling overhead. The more parallel LPs that exist per processor, the greater the scheduling overhead but also the greater potential that the processor will have useful work to do.

In DPAT, stationary objects are implemented as LPs, while moving objects are implemented as messages passed among the stationary LPs. Practically, this means that airports, terminal airspace, enroute airspace, and enroute waypoints called "fixes" are modeled as LPs, while the flights are messages that move among these LPs.

Several criteria were used in making this decision. First, it maps to the physical system nicely: airplanes move among stationary resources, much as messages are moving among stationary LPs. Secondly, it allows easy proximity detection. The moving objects that are near each other will be located in one LP. Because LP boundaries are precise, it is easy for LPs to share information with their neighbors to resolve inter-LP proximity.

There is a potential drawback with this approach. Because the number of airports and airspace sectors is potentially large, the number of LPs will correspondingly be large. On a small number of processors, this might increase the scheduling overhead and therefore negatively impact performance.

For scenarios that concentrate on North American airspace, DPAT is typically configured with 500 airports, 20 additional "super source/sink" airports, and about 730 enroute airspace sectors, for a total of 1250 LPs. Some scenarios require speed restrictions to be placed enroute; these scenarios would add about a dozen "fix" logical processes. The largest DPAT scenario ever built consisted of over 1800 LPs representing global air traffic; the smallest consisted of only 30 LPs modeling East Asian traffic. DPAT performs well at both extremes.

6. Performance Measurements

The performance measurements presented below are from a standard DPAT scenario. The standard scenario uses traffic from May 6, 1993, with 520 modeled airports and 730 modeled airspace sectors, for a total of 1,250 LPs. There are 487,238 committed events. This scenario is typical of those used for actual analysis with DPAT; most scenarios contain between 400,000 and 600,000 events. Each LP contains state ranging from one to two kilobytes, and the state is incrementally saved. The message sizes are only a few hundred bytes, as the messages (which represent flights) contain pointers to static read-only data (such as flight itineraries, aircraft performance, and so forth). Because DPAT runs on a shared memory SMP, the use of pointers is nonproblemmatic.

The system is measured on a six processor Sun SPARC SMP with 450 MHz processors and four gigabytes of memory. The parallel simulation system used is the Georgia Tech Time Warp (GTW) system, which is composed of about 20,000 lines of "C" code and uses multiple threads on an SMP as the basis for parallelism [12]. Each performance measurement was computed five times, to get an idea of the standard deviation around the mean results.

6.1. Granularity Measurement

It has been shown that larger granularity can produce better performance, particularly if the LPs are somewhat independent. DPAT is a queueing-based model, so its granularity is relatively small. Because we are using GTW on an SMP system, the small granularity impacts performance only slightly.

The granularity distribution is shown in Figure 2, and has a high variance as well as being highly skewed. There appears to be no regularity in the length of events in DPAT. Although the model is fundamentally a queueing network, the servers have different traffic patterns, with different routing characteristics, queue lengths, service times, and internal decision algorithms. Thus a particular

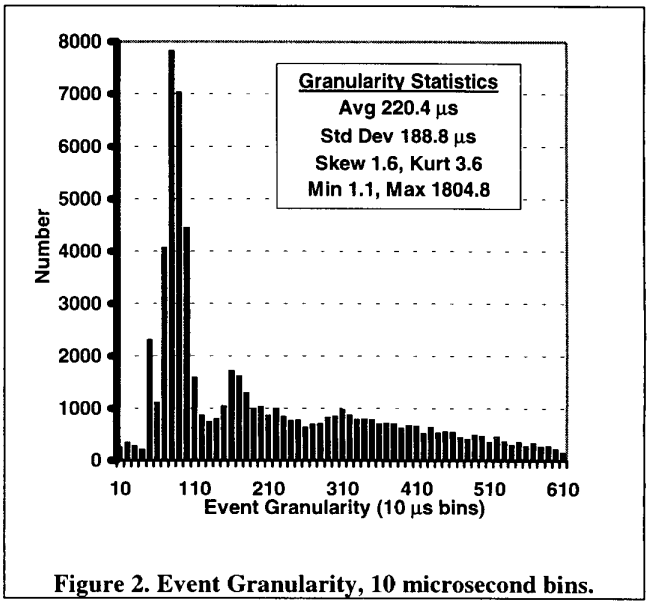

Figure 2. Event Granularity, 10 microsecond bins.

type of event at one LP takes a different amount of real time than the exact same type of event at a different LP. In terms of performance modeling, the high variance combined with the skewness of the distribution means that it is difficult to characterize using a standard probability distribution. In fact, different DPAT scenarios manifest different granularity distributions.

6.2. Lookahead Exploitation

Among the most sensitive indicators of parallel performance is the amount of lookahead in the model. Figure 3 is a histogram of the lookahead in the sample scenario for DPAT. The histogram is given in logarithmic units. Each bin spans a factor of ten, and there are 65,523 events that are sampled for this distribution from a total count of 487,238 committed events for this run of DPAT. The sample was taken from a sequential run of DPAT, and thus is not influenced by out-of-order risky event processing introduced by parallelism. Because DPAT yields the same number of committed events for both parallel and sequential runs, the data here are a valid look at a DPAT scenario.

Like the granularity, the lookahead of the model is highly skewed. The model runs for a total of 3,000 simulation time units, however the model completes 95% of its total events by time 2,000. Thus the average lookahead value of 62.6 represents a lookahead of about 2% of the total simulation time, or about 3% of the bulk of the simulation activity. The variance of the distribution is quite high, with a coefficient of variation of about 2.9.

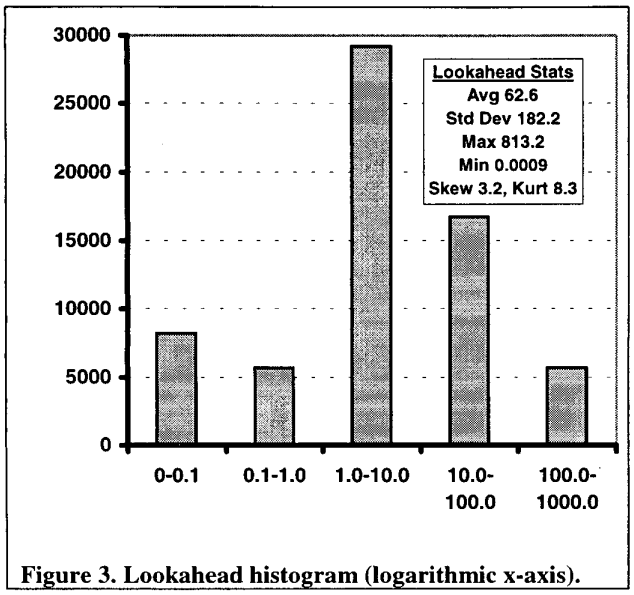

Figure 3. Lookahead histogram (logarithmic x-axis).

The distribution is highly skewed and is leptokurtotic. The maximum lookahead is 813 units of simulation time.

Figure 4 shows the lookahead probability from the same run. It is essentially a magnification of the histogram for the first 20 minutes of lookahead. The probability is binned into 0.125 simulation minute intervals, and the distribution is cut off at a lookahead of 20 simulation minutes. This cutoff results in a smaller set of 30,282 events from the total sample of 65,523 events. Within this smaller set, the mode is bin 0-0.125, which occurred 20.7% of the time (this value is truncated in Figure 4 because of the vertical scale); the next most frequently occurring lookahead value is bin 4.5-4.625, which occurred 1.44% of the time. A detailed analysis of the data shows that the true mode is a lookahead of 0.01 simulation minutes, which occurs 4,787 times, which is 15.8% of this smaller data set or 4.3% of the larger sample of 65,523 events. Ignoring the spike at the first bin, this distribution resembles a Poisson distribution.

DPAT is essentially a queueing network applied to aviation. Lookahead characteristics of queueing networks have been extensively studied. In [13], a generalized queueing network is studied wherein each server schedules the arrival of the customer at the next queue immediately upon entering service in the previous queue; the next queue then computes a service time for the new arrival. As with this general model, DPAT also knows the queue to which each aircraft will be routed upon completing service. However, two additional considerations cause it to behave differently than in Nicol's general case. First, it is possible for the receiving queue to *block* the arrival of a new customer; the conditions governing whether the customer is blocked depend upon the number of customers in the receiver's own queue just prior to the attempted handoff. Secondly, the service rates for any queue in DPAT, rather than being sampled from a constant distribution, are sampled from a distribution that varies nonlinearly as a function of queue length, and by simulation time. The variation by queue length models airport configurations when an arrival or departure "push" results from hub-and-spoke scheduling by the airlines. The variation by simulation time allows analysts to model weather events that change service times. These nonconstant distributions allow realism in modeling airport and airspace operations, both of which are sensitive to congestion and local weather conditions.

To exploit lookahead under these conditions, DPAT schedules the next queue arrival when an airplane enters a server, similar to Nicol's model cited above. Unlike this general model, if the next queue blocks the arrival, then a *near zero lookahead* (NZL) message is sent back to the previous server, at the time of the attempted handoff. An NZL message is one whose lookahead time is nonzero but less than 0.01 simulation time units. The receipt of this event causes the airplane to wait at the previous server for an additional amount of time, constituting a queueing delay. At some later time, when the next server has cleared a spot in its queue, a second NZL message is sent to the previous server that moves the airplane between the two queues. These NZL events occur infrequently, from about 3% of the total events for a typical scenario, up to 20% for a pathologically congested scenario. Such NZL events are an anathema to conservatively synchronized simulations, while the optimistic Time Warp engines handle them with little problem.

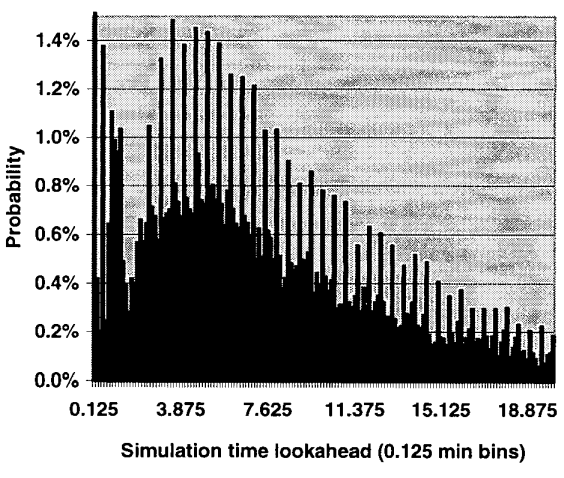

Figure 4. Lookahead probability below 20 minutes.

6.3. Risk Management

Risk management refers to how the optimistic simulation programmer deals with optimistically generated errors. Such errors are the result of out-of-order optimistic processing of events and do not exist in sequential simulations, optimistic but risk-free simulations, or conservatively synchronized parallel simulations. As such, they represent potentially catastrophic behavior unique to the fully optimistic technique that is difficult to predict but very easy to manage.

Risk management has been known since the invention of optimistic simulation by Jefferson and Sowizral in the early 1980's. Awareness of this issue has been recently elevated through a high-profile paper on its dangers [14]. The problem is that the programmer must ensure that events executing in a risky manner avoid errors that bring down the entire simulation. Such catastrophic errors can be caused by floating point over/underflow (such as dividing by zero or taking the logarithm of a negative number); by memory overwrites; by array bounds overflows; and so forth. It is possible to write a simulation that behaves correctly when executed sequentially, but exhibits one or more of these errors when executed optimistically.

In practice these errors are very easy to manage. First, the parallel simulator can itself trap most of the problems (such as floating point exceptions, divide by zeros, over/underflow, and so forth). When these exceptions occur, the state can be marked in error and, if the state is rolled back and executed properly, then the error is effectively "undone." This strategy was employed and implemented successfully in the Time Warp Operating System. Beyond that, the simulation programmer can prevent the conditions that lead to erroneous behavior by putting guards in the code—either explicit conditional tests or constructs like the C++ try/throw block.

A skeptic would argue that the latter is a heavy burden on the programmer. They would argue that the number of guards that need to be placed in the code, plus the possibility of missing one, is such a high cost that it renders optimistic simulation useless. They would further argue that the burden of guarding all function calls, library routines, and so forth makes such implementations impractical. It is certainly true that building an optimistically synchronized parallel simulation is harder than building its sequential counterpart. On the other hand, many of the risk-driven errors are indistinguishable from errors that occur in ordinary sequential simulations, albeit through other mechanisms. For example, how do sequential simulation programmers guard against bad input data that causes a negative logarithm to be computed? A human pushing a key at exactly the wrong time? A combination of events in a scenario that might lead to division by zeros? Buffer overflow from a large data set? The most common method is through validation checks on the input data, assertions throughout the code, and conditional tests that trap such errors, combined with very thorough and careful testing. These are the same techniques that an optimistic simulation programmer would use to resolve risk-generated errors.

In this respect the risky aspect of optimistic simulation actually is a *benefit* to the simulation developer. Each execution on multiple processors generally follows a different code branch—because the set of risky events run in successive replications is usually different. Thus the code is "automatically tested," although in a nontraditional sense. The sequential programmer would have to develop dozens of data sets to discover what an optimistic programmer would find out in only a few runs of the simulation. Such rigorous testing is not normally done in academic environments, and yet it is *necessary* and *forced* when running optimistically.

Our experience is that the only risk-generated errors encountered in a run of DPAT—running out of space in an array or dividing by zero when computing average values—are easily guarded in the code and thus easily avoided.

6.4. Verification

But how do we know if DPAT produces correct answers? How can the DPAT developers guarantee to the analyst that the result is not faulty due to some combination of risky events that were not rolled back properly? The chief verification tool is the observation that the set of committed states from a multiprocessor run of an optimistically synchronized simulation must exactly match the set of states produced by the same simulation when run sequentially. This is the essence of the term "logical correctness." Thus, if a sequential run of DPAT repeatedly produces *identical* output to the parallel run, regardless of the number of processors used, then the system is free of errors that might be caused by optimistic processing.

Verifying that a sequential run is identical to a parallel run is an *enormous* task, even for a small model such as DPAT. However, in ensuring such consistency, many model bugs (that are there anyway—even in a sequential version) are eliminated. In DPAT, inconsistent results are usually due to one of two problems: (1) inconsistent treatment of simultaneous events or (2) roundoff errors when converting from one precision to another. These bugs tend to be *nonrepeatable*—when the state rolls back and re-executes, a different value is generated the second

time from the first (and therefore, a different value is generated vis-à-vis the sequential run). These problems are there even in a sequential model, except that there is no mechanism to discover their source, and therefore many extant sequential models latently exhibit these problems. In DPAT, it is true that every run using the same initial random number seeds—regardless of the number of processors that it uses—generates the same number of committed events, and the same exact answer, byte-for-byte.

Beyond verification, there is the additional task of validation—ensuring that the model is producing results that are credible with respect to the physical world it is simulating. Validation is also an enormous task, and has been done several times. Most noteworthy is the validation conducted by the Federal Aviation Administration, which revealed that four of five metrics tested produced results close to the real world [3]. The fifth metric, passenger arrival delay, is influenced by factors that are not modeled in detail by DPAT.

6.5. Execution Time Performance

The performance of DPAT on the standard scenario is shown in Figure 5. Both the run time and the number of rollbacks are included in the figure. The results were computed on a six processor machine. Results are shown up to eight processing elements (PEs) because the GTW system creates threads for each PE. Therefore it is possible to create more PEs than there are physical processors on the system. As can be seen in the Figure, when that occurs (at 7 and 8 PEs), the performance suffers. Not only does the run time begin to suffer, but the variance in the run time from successive executions begins to become large. The larger variance is due to the fact that the operating system is now scheduling more than one thread per processor, and the nonrepeatabilities in such scheduling between successive runs cause different execution times. In all cases, as noted above in the discussion on verification, the result produced to a DPAT analyst is identical. There is about a three fold reduction in run time up to three processors (linear speedup); beyond that, the run time levels off.

6.6. DPAT and the High Level Architecture

The United States Department of Defense has produced a specification for simulation interconnection commonly known as the High Level Architecture (HLA). As DPAT is capable of producing throughputs and delays quickly, federations containing a DPAT component have been proposed. One such federation is a combination of the Sector Design Analysis Tool (SDAT) with DPAT. SDAT allows aviation analysts to reconfigure both the shape and

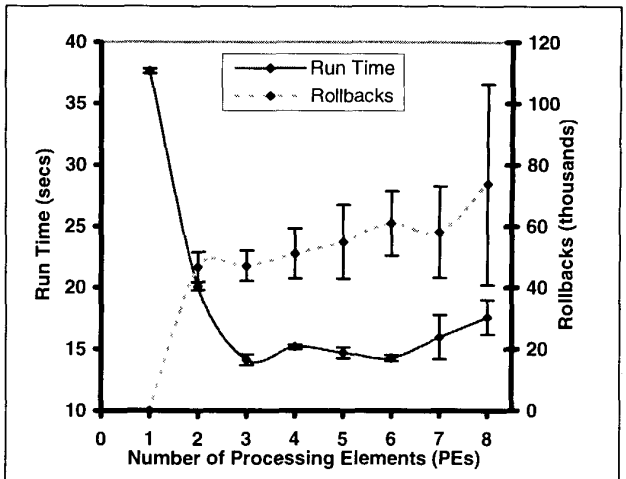

Figure 5. Performance of DPAT.

number of airspace sectors in a region. Airspace geometry affects the workload of controllers in a very profound way: the greater the number of crossing tracks, the more work controllers must do to separate planes. Accordingly, the fewer airplanes a controller will allow in their airspace.

To assess the delays caused by reconfiguring sectors, DPAT has been federated with SDAT. The resulting tool allows analysts to effectively redesign airspace while assessing the affect different traffic loads might have on the results. Federating the models required an initialization sequence, during which the models register their objects and interactions with the HLA's Run Time Infrastructure (RTI), followed by a processing sequence in which SDAT publishes its new routing structure and DPAT then publishes the predicted delays. The interaction diagram for the processing sequence is shown in Figure 6. It should be noted that there are no limit on the number of SDAT federates that can be incorporated in the system, due to the inherent flexibility of HLA.

7. Conclusions

We can make some general statements about why DPAT avoids many of the problems associated with optimistic computation. First, in the hundreds of scenarios run through DPAT, we have observed no "rollback explosion." A typical configuration of DPAT, containing over 1,200 LPs on six PEs, yields an average of 200 LPs per PE. The large number per PE means that each PE will generally see "average" behavior of the system, preventing any one from computing too far ahead or lagging too far behind. The result is a well-balanced, stable system. Secondly, the state size of about 2 kilobytes per LP,

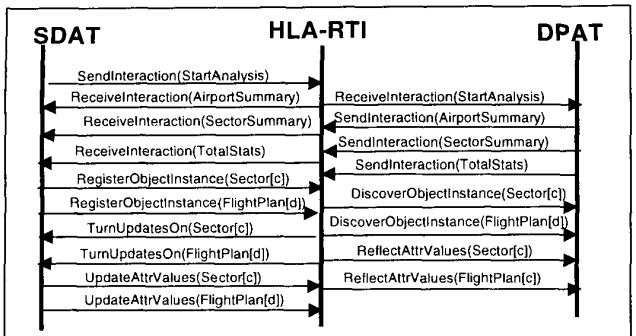

Figure 6. HLA interaction diagram (not in time order).

coupled with the use of incremental state saving, means that the overhead of saving state and rollback is minimized. Thirdly, the messages average a few hundred bytes each, and are passed through pointers in shared memory; this minimizes message communication costs. Fourthly, the software contains only a few places where risk-driven event processing might cause errors, and those places are easily identified and fixed with proper testing. Finally, optimistic computing is not as sensitive to zero-lookahead messages as conservative systems, so modeling blocking queues (as is done in DPAT) can be accomplished with no change to the conceptual design.

Our main conclusion is that it is possible—even desireable—to build a model containing optimistic synchronization that avoids the problems associated with such systems and exploits its main advantages: fast run times and the ability to interconnect with other models easier. The key factor to achieving these goals is to design a system with hundreds to thousands of LP's with small (1-3 kilobyte) states, incremental state saving, to run on only a few processors (so that there are hundreds of LPs per processor). Systems configured in this manner, like DPAT, can be effective vehicles for delivering usable simulations to the analysis community.

8. Acknowledgements and References

We would like to thank Frank Carr for his work on the DPAT-SDAT HLA connection, as well as Dr. Len Wojcik for his support over the years. We would like to thank Dr. Richard Fujimoto's research group at the Georgia Institute of Technology for providing the GTW system.

The contents of this paper reflect the views of the author. Neither the Federal Aviation Administration nor the Department of Transportation makes any warranty or guarantee, or primise, expressed or implied, concerning the content or accuracy of the views expressed herein.

[1] Carroll McCormick, "Jam Tomorrow," *Air Traffic Management*, Euromoney Publications:London, **8**(2), March-April 1999, pp. 42-43.

[2] L. A. Wojcik, et. al. "World Regional Air Traffic Modeling with DPAT," MTR 97W00000070, September 1997.

[3] Daniel Citrenbaum, Nastaran Coleman, Robert Kennington, Bryan Baszczewski, Anh Nguyen, "The Detailed Policy Assessment Tool (DPAT) Validation Report: A Comparison of DPAT Simulation Results to National Data," Federal Aviation Administration, October, 1999.

[4] Dr. George Donohue, "Testimony before US House of Representatives Committee on Science, Subcommittee on Space and Aeronautics: Hearing on the 2001 NASA Budget Request," 106th Congress, April 11, 1999.

[5] "Separation Reduction Impact Analysis: Initial Results," MP 99W0000019, January, 1999 (not for public release).

[6] Suzanne Bradley, et. Al. "Improving NAS Efficiency: Assessment Approach and Initial Analyses," MP 99W0000020, January, 1999 (not for public release).

[7] Jim Cieplak, John DiLeo, Jonathan Hammer, and Stephen Yeh, "Annual DPAT Analysis Capability," briefing, September 1998 (not for public release).

[8] Frederick Wieland, "Parallel Simulation for Aviation Applications," *Proceedings of the 1998 Winter Simulation Conference*, (IEEE) Washington, DC, December 1998.

[9] Frederick Wieland, "Limits to Growth: Results from the Detailed Policy Assessment Tool," in Proceedings of the Digital Avionics Systems Conference, Irvine, CA, IEEE:1997.

[10] "FAA Disputes Airlines' Claim of Outdated ATC System," *Aviation Daily*, McGraw-Hill Publishing Company, July 19, 2000.

[11] Frederick Wieland, "The Detailed Policy Assessment Tool User's Manual," MITRE Technical Report MTR99W00000012 (not for public release).

[12] R. M. Fujimoto, "Time Warp on a Shared Memory Multiprocessor," Transactions of the Society for Computer Simulation, **6**(3):211-239, July1989

[13] David M. Nicol, "The Cost of Conservative Synchronization in Parallel Discrete Event Simulation," *Journal of the Association of Computing Machinery*, **2**(2), April, 1993, pp. 304-333.

[14] David M. Nicol, "The Dark Side of Risk (What your mother never told you about Time Warp)," *Proceedings of the 11th Workshop on Parallel and Distributed Simulation*, June 10-13, 1997 IEEE Computer Society Press:1997.