Operational Information Systems - An Example from the Airline Industry

Van Oleson
Delta Technology
Georgia Tech
van.oleson@delta-air.com

Greg Eisenhaur
Georgia Institute of Technology
eisen@cc.gatech.edu

Calton Pu
Georgia Institute of Technology
calton@cc.gatech.edu

Karsten Schwan
Georgia Institute of Technology
schwan@cc.gatech.edu

Beth Plale
Georgia Institute of Technology
beth@cc.gatech.edu

Dick Amin
Delta Technology
dick.amin@delta-air.com

Abstract

Our research is motivated by the scaleability, availability, and extensibility challenges in deploying open systems based, enterprise operational applications. We present Delta's mid-tier Operational Information Systems (OIS) as an approach for leveraging its legacy operational OLTP infrastructure, to participate in the emerging world of electronic commerce, as well as enable new applications. The approach is to place minimally intrusive 'taps' into the legacy OLTP systems to capture transactions as they occur for consistent replay in the mid-tier OIS. One important issue addressed by our work is the processing, and dissemination of information in the mid-tier system itself, potentially serving hundreds of thousands of access and display points, distributed across a highly geographically distributed system (e.g. airports world wide), and also involving large `working sets' of operational data, used by applications that require rapid response and also rapid recovery from failures. To address the scaleability, availability, and cost of this OIS infrastructure, we are researching cluster computing techniques, as well as, devising replication and failover techniques. To address the communications scaleability requirements, we are experimenting with novel event-based implementations of information transport and processing, that include reliable multicast variations.

1. Introduction

Increased competition in the airline industry is stimulating the development of new applications of information technology, including a new strategic focus on electronic commerce at Delta Air Lines. Traditionally, large enterprise computing at companies like Delta has relied on using clusters of mainframes running proprietary information systems software. For example, Delta relies on a cluster of IBM S/390 mainframe computers running system TPF (Transaction Processing Facility), a specialized operating. These traditional online transaction processing systems (OLTP) support applications that automate the majority of the airline's operational services. The TPF and MVS systems architecture has proven to be highly scaleable and available, and the systems have operated successfully over the last 30 years and through the Y2K bug scare.

It is difficult to modify these existing OLTP applications to accommodate a changing business. Many of the applications were developed in assembly language and have evolved over a period of more than 30 years. Originally, the applications were designed to implement specific business models and offer little flexibility to support new business models and processes. Specifically, these applications maintain ownership of rigidly defined data sets, and their legacy data formats offer little opportunity for creating new relationships to other application data. Additionally, new business models result in new applications, some of which leverage the Internet. This exposes the legacy systems to unforeseen transaction volumes.

In response to these limitations, a novel strategy pursued by Delta is the addition of mid-tier enterprise information systems, termed Operational Information Systems (OIS). The wealth of information in the existing OLTP systems is harvested by "grabbing" strategic transactions as they occur in soft real-time. These transactions are then replicated and consistently replayed in the newly introduced OIS. In this new environment, data resulting from the transactions is mapped into alternative evolvable formats, which is correlated with previously unrelated information, as well as information from sources other than the OLTP systems. Additionally, the immediate correlation stimulates events, which are derived from the transaction histories. This capability enables an entirely new class of real-time event based applications, which have proven to radically improve the efficiency of airline operations.

The new mid-tier OIS, considered in concert with the legacy OLTP system, is the basis on which Delta constructs new applications and improves current business operations, including improving the "Customer Experience". The key element to their success is the development of new mission-critical software and hardware infrastructures that support these efforts.

In the remainder of this paper, we first characterize Delta's OIS strategy and its components in more detail. We then state the issues that motivate the academic research in to highly scaleable and highly available OIS implementations.

2. OIS Components

The systems model in figure 1 depicts the overall architecture including the major systems and physical components that implement the OIS.

Legacy OLTP Systems are long-lived information systems that continue to support application operations. These applications are `tapped', which result in transaction histories to be distributed to the Event Derivation Engine.

Event Derivation Engine is a Global Information Base comprised of a set of servants that internalize transaction histories from OLTP systems, as well as, other internal and external sources of information. The EDE correlates and consolidates this information and maintains an operationally narrow subset, or operational window, from which it derives events for publication. Additionally, the maintained consolidated information serves as a base for simple request and replies, as well as, initial states for subscribing clients.

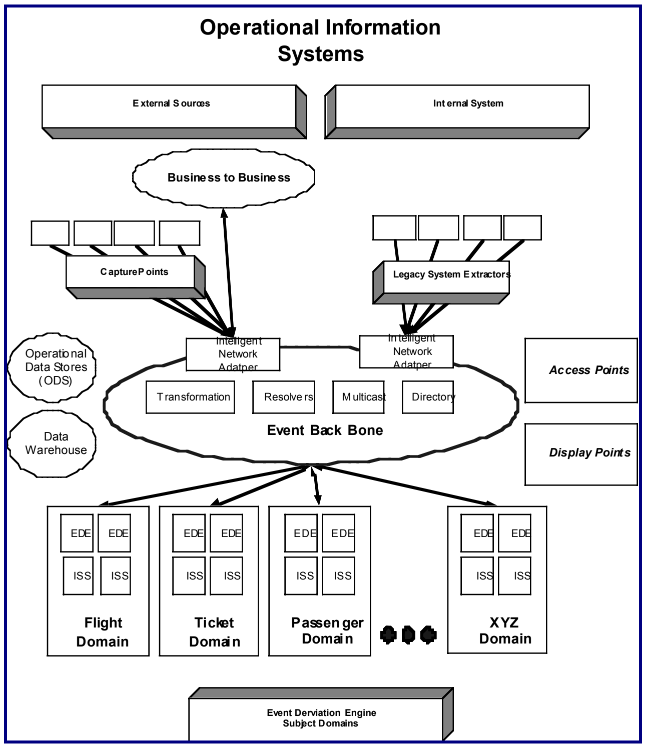

Figure 1 Systems Model

Intelligent Network is an IP based network that is embellished with strategically located resolvers and brokers to support inter-application communications. The application layer routing supports efficient and reliable message transportation.

Intelligent Network Adapters (INA) connect systems to the intelligent network. Delta's core legacy operational applications are implemented on the TPF operating system, which manages a loosely coupled cluster complex of IBM s/390s with a shared file system. Until recently, this operating system did not support a TCP stack and integration with IP networks was accomplished via gateways and custom protocols. As the TPF supported TCP stack matures, it will become compelling for some application interactions. However, a novel approach to this problem is being implemented by Delta. By using a hardware supported off-load engine that emulates the 3490 Tape interface, applications need not be modified to use the TCP interfaces. Applications simply continue to write and read to tape as they continue to assume a tape device.

This smart control unit also supports transformation to various contemporary message encodings, such as XML and additionally provides replication and brokering.

Operational Data Store (ODS) supplements the EDE. The ODS maintains a much larger operational window than that of the EDE. Additionally, the ODS accommodates alternative access styles such as complex analytical queries. The data store may also serve as a system of record for new operational objects that are not implemented in the legacy systems.

Initial State Service (IIS) is a mechanism, from which event based clients retrieve an initial view of information prior to receiving events that update that view. An example is a flight information display, where updates for flights may arrive at a client sparsely. That is, few events arrive over time. A passenger for a flight is interested in its current status. The initial view provides this current status in lieu of a status update event.

Access Points can both capture information and therefore, produce events, and also manipulate it. An important role of an AP is to permit the addition of new services, such as passenger paging upon flight arrival, dynamic pricing based on passenger profiles and current flight/airport status (e.g., availability of seats on competing flights), etc. These examples also demonstrate that APs may be connected to various output devices, such as pagers. Another AP is a baggage system for lost baggage. Passengers could register for baggage status events via a personal data assistant, which can be notified of ultimate arrival of the baggage. From these examples it is apparent that APs also vary, ranging from palmtops with wireless connections used by roving gate agents to the reservation-capable systems used by central airport agents.

Capture Points are any internal or external source of information. One example is an aircraft that emits positioning signals for capture in the operational service. The system's distributed capture points (CP) continuously emit events describing current status, using typed event records with unique instance IDs. CPs range from being low-end and ill-connected (e.g., wireless data entry devices used on the tarmac), to being high-end and well-connected, such as the customer-visible gate readers that scan boarding passes as passengers board, automating the boarding process. Consequently, the events produced by CPs also vary in complexity, one of the more complex events being an arrival event for a flight with a certain ID, such as a Surface Movement Advisory (SMA) at an airport; such events are contained in the FAA data feed.

The Operational Information System is composed of four fundamental processes: event acquisition, event consolidation, operational data storage, and derived event publishing.

3. Event Mining and Acquisition

The first of the four basic processes of the OIS is the acquisition of events from source systems. This includes the techniques for mining and tapping event sources, the ordering properties of transaction histories, as well as the publishing and transportation of this captured information.

The model used by Delta, as in other operational settings, is that of acquiring and replicating transaction histories to the Event Derivation Engine. Some of the specific information captured, generated, and transported in Delta's OIS includes flight, passenger, crew, situational, and environmental data. Some of these flows are produced by internal OLTP systems, such as flows that contain flight, passenger, and baggage information. Other flows are provided by external sources, such as FAA feeds, which provide radar-gathered positional flight data and weather feeds provided by a weather service.

3.1 Transaction Tapping

Transaction snooping and software agents are two basic techniques for tapping transaction systems. In either case, it is imperative to minimize the intrusion in the legacy systems. The core OLTP system was initially planned to processes several million well-behaved transactions per day. When tapping the legacy OLTP systems, existing service agreements must be maintained so that current users see no degradation in performance. Therefore, techniques for tapping must be minimally intrusive.

Transaction snooping is using a non-intrusive means of `grabbing' transactions as they occur. For example, modern OLTP systems incorporate sequential transaction logs for recovery purposes. With knowledge of the log format and the ability to view the log, transactions can be detected and acquired. The captured transactions can then be forwarded to a brokering engine for dissemination. That is, by utilizing memory-based table references, the transactions can be decoded and reformatted for transmission to an OIS. This is straightforward for legacy applications that are altered infrequently, as the reference table must be updated with any transaction change. This technique is highly compelling, since hardware support can be used to snoop transactions as they are written to the logs.

An alternative technique is the utilization of software agents injected into the applications of an OLTP system. As non-intrusively as possible, these agents build records over the lifetime of some business transaction. They then fire triggers that generate appropriate events into a transport mechanism to make the data accessible to the new mid-tier OIS.

Both techniques are used at Delta, since many of the legacy TPF applications do not physically store the transaction boundaries in a transaction log for snooping. In order to capture the transaction context, the transactions must be gathered while they are occurring by a software agent. Upon commit, the transaction history is queued for I/O.

3.2 Transaction Ordering

The transaction histories must be complete histories of relevant interactions captured by the legacy system. Given such histories, the mid-tier OIS must be able to faithfully recreate and replay relevant operational state changes known to the legacy system and important to the mid-tier OIS.

Although some source systems provide consistent, reliable, and ordered messages that can be trivially internalized by an EDE, tapping some legacy transaction systems can result in an arbitrary re-ordering of the captured transaction histories.

The Intelligent Network Adapter used to integrate the TPF system with the OIS does not solely solve ordering anomalies that are introduced by the asynchronous I/O model used to transmit the captured transaction histories.

As an example of the ordering anomalies, consider the reservation system running on this loosely coupled cluster architecture. Specifically, when tapping this system's transactions for passenger status, we can acquire information about boarding status, seat assignment, customer status, etc.

To simplify the example, consider a system with 3 loosely coupled nodes, N1, N2, and N3. Assume that a transaction on a specific object instance can be arbitrarily routed to any node (this is a shared disk databases model, where any node can update an object such as a passenger record). For this example there are three transactions that update passenger record, “Jones”. They are identified as T1_Jones, T2_Jones, and T3_Jones, which execute on N1, N2, and N3 respectively. Each transaction is properly serialized by the shared database and ordered by its occurrence. In this case we order T1_Jones happens before T2_Jones and T2_Jones happens before T3_Jones.

The problem arises as the captured transactions are asynchronously scheduled for I/O by the node on which the transaction occurred. This allows for transactions to enter the network not in order of their occurrence. That is T3_Jones can be sent before T2_Jones and T2_Jones can be sent before T1_Jones. If not re-ordered by the EDE, this results in an inconsistent view of the working set.

Synchronous coordination of the outbound transactions is detrimental to high throughput and scalability of the clustered complex. The asynchrony of the node processing can lead to non-deterministic delays. These delays result in large I/O queue depths that can ultimately result in back-pressure that affects existing service levels. That is, normal application processing can be affected.

Another scenario is the failure of a node, for which a transaction occurred. The transaction is not scheduled for I/O until the node is recovered. In this case a failure of N2, would result in a significant, possibly indefinite delay of T2_Jones. The EDE can’t allow T3_Jones to execute, since this results in an inconsistency. If the node is not recovered in a reasonable time, the OIS must then re-synchronize with the legacy OLTP database for that instance “Jones”.

Unfortunately, the legacy TPF applications do not encode the transaction boundaries in a transaction log and there is no corresponding unique transaction identifier. As demonstrated above, the ability to re-order a transaction-history is vital to the consistent reply of transaction histories in the EDE.

To account for the arbitrary re-ordering, Delta incorporates instance-based application sequencing to order the captured transactions. An instance is an object, "Jones", that has been modified by a transaction. All transactions occurring on the passenger record, "Jones", are sequenced by a monotonically increasing sequence number.

This instance based sequence number allows the EDE to appropriately re-order the transaction histories. The instance-based sequencing technique has a profound advantage over a traditional unique transaction identifier. The concurrency potential is dictated at the instance level. That is, when a message for the instance, "Jones", is indefinitely delayed, all other instances can be consistently re-played in the EDE. Only the instance, "Jones", is required to be re-synchronized. This allows the EDE to achieve optimal levels of parallelism, by using an instance based concurrency controller.

Application instance sequencing is critical in the loosely coupled cluster since there are more opportunities for ordering anomalies by the cluster. Additionally, the relative frequency of updates to an instance is high therefore the probability for re-ordered transactions is high.

As an example of an intolerable inconsistency, consider gate agents utilizing a new application of the OIS infrastructure. By using real-time updated seat maps, agents have current knowledge of seat assignments. However, if inconsistencies were allowed, a passenger could show up with a valid boarding pass, however, information reflecting this may not be consistent at the gate. In fact, the passenger could be denied immediate boarding as he scans the boarding pass, which is rejected. Of course the passenger will be allowed to board after reconciliation, with the legacy system. However, this defeats the benefits of such a system to improve boarding times, and the overall customer experience.

3.3 Event Taxonomy

A challenge exists in that the information streaming from the loosely coupled systems (and/or from other sources, such as the FAA data feeds) is not delivered at the granularity useful to current or future applications.

Unfortunately, the resulting events produced by the legacy system do not individually contain the information needed by various business processing performed in the mid-tier EDE. To address this issue and to be able to handle diverse input streams to the EDE, we have developed the following characterization of events produced by external systems:

Discrete events are semantically meaningful to some OIS application. Upon receipt by the EDE, such can be immediately published.

Partial events implement state changes that in themselves are not useful to an application. Such events are directed to state engines, which will eventually produce a discrete event for some application. Partial events may be received from multiple sources (i.e., event channels) before causing a state change and therefore, a discrete event relevant to an application.

Incomplete events result from the ordering anomalies introduced by the clustered OLTP systems. As described previously, these events must be stalled until missing events arrive or the system deems this instance is not recoverable, resulting in re-synchronization processes.

Complex events are comprised of some combination of discrete and partial events. Such events are useful when applications require larger granularity activations than those resulting from discrete events.

These classes of events motivate a consolidation and correlation tier, where the Event Derivation Engine collects the event flows and derives events application-friendly events.

4. Event Derivation Engine

The second process of an OIS infrastructure is the correlation and consolidation of the tapped data from internal and external sources.

When information from internal and external capture-points is acquired and delivered to the OIS, the EDE exercises business rules to create new associations and representations of the information. For example, when the status of a flight changes, these state changes are delivered as events from capture points (e.g., the aircraft or the dispatcher) to the EDE. Here, the resulting updated status is internalized and represented in the current operational working set. With this working set defined, interfaces are provided for interested applications, which may request the current state of these new information representations and subscribe to the resulting state changes as events.

In practice, the event rates have already exceeded a non uniform distribution of over 12 million messages per day (Figure 2) from the internal and external sources that publish to the OIS infrastructure. This number is expected to rise significantly as more useful information is captured for event derivation.

Although this rate appears trivial, it is not amortized uniformly over the 24-hour period. Airlines incur high frequency peaks and transaction rates can double during holidays, fare wars, and strikes.

The EDE must maintain some subset of these flows as a base from which to derive application-friendly events. That is, information from external and internal sources does not typically match the expectations of consuming applications and must be correlated with other data to establish meaningful events. Additionally, initial states, described later, queried from the base.

Activity	**Messages/Day**
Flight Progress (TPF)	500,000
Flight Progress (FAA)	250,000
Passenger Information	3,500,000
Tickets	2,000,000
Inventory	5,000,000
Seats	500,000
Customer	100,000
Fares and Rules	200,000
Total	12,050,000

Figure 2 Message Rates From External Sources

For a perspective, as compared to data warehouses, which typically contain tremendous volumes of historical information, an OIS contains only the fundamental subset of information required to run day-to-day operations. Although the operational working set is a much smaller set, the aggregation of operational flows from external and internal sources can result in operational data stores of Terabytes in magnitude.

As previously stated, flow aggregation results in Terabyte-size databases. Maintaining these databases coupled with analytical processing on the data are two fundamental tasks of the mid-tier OIS. Other tasks include the acquisition, derivation, and publication of events with low latencies and in soft real time. Considering the demands of these tasks, an important observation is that the order of magnitude of the data from which application events are derived can be dramatically reduced, by focusing on precisely the data needed for near-term operational decisions and actions.

Therefore, we improve event throughput and latencies by defining a derivation subset, named the Derivation Working Set (DWS), which is of much smaller scale. The DWS contains the minimal amount of information needed to derive the events required by OIS applications. Performance of data storage and access for event derivation is improved substantially because this working set can be implemented as a main-memory database that is optimally organized to accommodate event derivation and initial state queries.

The DWS scoped via a window scheme, where content is rolled in and out of the DWS based on relevance. Specifically, in this set is kept all state of `current interest', so that it is rapidly accessible to relevant business logic. For instance, data about a flight's departure is kept in the DWS until the flight has arrived, whereupon `business logic' adds to the DWS that the information that a certain flight leg has been completed. The lifetime, or window, of the information contained in the DWS is based on business operations for a specific business domain. For example, years of experience in dealing with flight information resulted in the identification of a window of flight data and behavior for some number of days (n) in the past to some number (m) days in the future. Lifetimes vary across business domains, and they may also be dynamic, such as lifetimes based on event arrivals. For instance, as a flight leaves a gate and begins taxiing, the boarding process for that flight is no longer relevant and may be flushed to the Operational Data Store (ODS) and possibly, to the data warehouse.

The EDE is the primary data provider and consumer for additional services associated with the operational subsystem, such as Internet-based reservations and flight information services, the reservation system used by external systems in a business to business model. Finally, the EDE also directly distributes events to display points, such as flight displays in airports, resulting in the need for high scalability (in terms of numbers of displays) for some of the event output streams emanating from the EDE.

EDE Processing Model

1. Transaction History (TH) arrives from source system.
2. Durably store the TH.

3. De-marshal TH.
4. Exercise concurrency control rules
5. Exercise internalization business rules.
6. Release TH
7. Represent TH in DWS
8. Derive corresponding application event.
9. Publish event.

In general, the role of the EDE is to create meaningful global states from event streams that provide limited ordering guarantees.

6. Operational Data Store

The third processing component of the OIS is the ODS. The ODS window is much larger than that of the EDE and is typically implemented with traditional relational databases. Operational Decision Support applications as well as Data Mining applications use the ODS to execute analytical queries in the ODS environment where they do not compete with the real time event derivations performed by the EDE. Additionally, the ODS serves as a staging area for populating the DWS. For example, passengers can book seats on flights in the distant future. This information is considered `operational' by Delta. However, it is outside of the DWS window for the passenger domain. The magnitude of this type of future information is quite large, so it is staged in the ODS until it falls within the DWS window.

The DWS support the inter-operation of event flows with the EDE and the ODS, in which, the ODS and EDE collaborate to provide traditional transaction processing to this new base of operational information, without impeading the requirements for low latency events.

7. Derived Event Publishing

The fourth processing component of an OIS is the dissemination of events derived by the EDE to its large numbers of subscribers (e.g., airport displays). In fact an existing deployment already exceeds 10,000 workstations, which must display flight status information.

The highest profile service of the OIS infrastructure is the support of soft real-time delivery of event information to subscribing clients. Real-time event applications are the impetus for re-thinking business processes and revolutionizing the operations of the airline. The benefits are profound. Consider an example where gate agents are provided with heads-up displays that present a current view of relevant flight information, including seat maps for the flights they are working. The traditional request/reply strategy is limiting as agents spent their time working at the computer terminal, typically typing requests to answer basic customer questions. The heads-up displays allow both customers and agents to be informed releasing the agents to spend their time responding to more difficult issues, including facilitating the boarding process.

To achieve low latencies and high scalability, the relaxation of the reliability of event transmission is based on application characteristics. While some applications require stringent guarantees, others can operate successfully under relaxed rules, which we term the `reliability spectrum'. To exploit this spectrum the use of a hybrid sender and receiver-initiated multicast protocol can provided dramatic improvements in the latency and communications scaleabilty of an EDE.

Initial states must be obtained for any event-based application that requires the current value of a set of information to begin operation. An application that demonstrates this requirement is Flight Information Display Systems (FIDS), for which there are many receivers (e.g., the large number of airport displays) that join and leave this service in a periodic fashion. A FIDS display requires initial values for presentation and subsequently, adjusts these states as flight changes are received. For FIDS, the determination of initial state is as simple as capturing a small set of initial data, for others this may require making a copy of the entire operational working set.

8. Experiences

The overall architecture of the OIS components has evolved over time as the scaleability and availability requirements have changed. Initially the system was a proof of concept that acquired immediate success and was deployed well-beyond its designed capacity. The currently deployed system has been refined to meet the scaleability and availability requirements.

The initial EDE design used a commercial relational database to internalize the transaction histories and represent the operational working set. The original intent was to enable fast, flexible queries, along with low latency event distribution. However, as the operational working sets grew to Terabyte magnitudes, we quickly realized the competition between maintaining large databases and fast event derivation from this database. Through our experience with this deployed architecture, we realized disk-resident relational data offered insufficient performance to solely handle all of the work required of the OIS infrastructure. Not only must the OIS process the variable peeks of the 12 million source messages per day, the OIS must additionally derive at least that many application friendly/discrete events to an initial deployment of 10,000 workstations. The desired number of workstations is expected to grow dramatically in the near future. Additionally, the explosion of initial state queries occur as workstations dynamically join/subscribe, which require initial states. For FIDS applications this initial state, which results in a XML result set of 5 MB places a tremendous load on the system. In an unlikely but worst case scenario, all current 10,000 workstations could come on-line at once requiring 10,000 queries.

This situation is exacerbated further by the existence of additional external systems, including passenger-booking traffic via the Internet. This will result in the existence of many more information flows and resulting analysis tasks in the future, ranging from `small' flows like automatic passenger paging services, to multimedia flows.

Delta immediately discontinued supporting the important feature of analytical queries from the OIS and began maintaining a scaled down in-memory representation of the working set. The relational database image was used for recovering this scoped state upon failures. Unfortunately, if failures in this system occur frequently, a business could face significant downtime. The time to replace the working cache from the several tera-byte RDBMS is on the order of 45 minutes.

This large volume operational working set motivated the partitioning of the set of information required for deriving events, the DWS, and the Operational Data Store. The ODS is organized to handle ad-hoc analytical queries, while the EDE derives events with lower latencies.

Additionally, client connectivity in the existing system, is based on a hierarchical fan out based on TCP socket concentrators. Delta has identified that this approach introduces un-necessary moving parts and adds latency as events traverse the hops.

Delta's experience and requirements in developing a commercially deployed OIS infrastructure has motivated our academic research in this space. The current scaleability challenges coupled with future scaleability projections stimulate a clean slate approach to researching more optimal architectures for an OIS.

9. Research Goals

The Operational Information System has a mission-critical dependence upon information that is acquired, processed, transported, and delivered with well-defined quality of service properties. The intent is to attain competitive advantages in decision-making, customer care, and to react in a timely manner to changes in current state. We must manage the processing and communications performed by multiple components of a large-scale distributed application. This application consists of large number of complex information flows, where operations applied to these flows have the purpose of extracting `useful' data from them. Information extraction must be performed under constraints like timeliness and continuous availability. Not meeting these constraints results in costs incurred by the organization.

In this section, we describe the research opportunities and academic interest in components of the OIS infrastructure.

9.1.1 EDE Scaleability

The EDE must be highly available and scaleable while also employing rapidly evolveable, plug-and-play hardware and software infrastructures, so that new services are easily added, data formats changed or updated, and additional streams and clients supported.

An opportunity to improve scalability of the EDE is parallelism of the transaction histories. Specifically, transaction histories from TPF can be executed in the EDE under the assumption of causality, which maximizes the parallelism in the stream.

An additional enhancement of the EDE is one that also replicates the DWS itself, to attain high levels of availability and scalability. We are designing solutions based on event mirroring and/or hot standbys, again using additional cluster computing engines.

9.1.2 Low Latency Events

One issue is the latency of outbound events, generated in response to the receipt of new input events by the EDE and subsequent state updates. High latency for such events has strong effects on operational capabilities and on the customer experience, the latter exemplified by Delta's ability to rapidly display flight and gate change information. Latency is affected by both event transport and event processing overheads. We first consider processing overheads. While these overheads are reduced by using an in-memory representation of the EDE' operational working set, there exist additional latency and bandwidth issues caused by verbose event representations and the unnecessary copying of events in interactions with the event-based communication infrastructure.

We are addressing these issues as performance opportunities by incorporating portable binary input/output (PBIO) encodings of event transfer formats along with “Just In Time XML” [2],[3],[4]. Effectively, by organizing the in-memory representation of the DWS to more closely match the application-friendly/discrete events, the PBIO technique allows the EDE to simply drop the memory image on the wire. This dramatically reduces buffer coping in the sender, since marshaling to an intermediate representation such as XDR or XML is not necessary. To deal with sender memory packing and endianness, the receiver has pre-cached meta-information that describes the memory structures that it receives as events from the sender. The receiver re-constructs the data structure in its native packing and endianness format and additionally translate the message into a contemporary format such as XML. This process occurs under the PBIO library and is abstracted from the receiver’s application code via some middleware package.

9.1.3 Communications Scaleability

Many applications can operate successfully in the presence of message loss and can take advantage of relaxed reliability protocols. This characteristic does not imply that the applications must be concerned with inconsistent views of the data. This characteristic means there are natural alternative means to ensure application information integrity. What is fundamentally required in this scenario, is the ability to detect event loss and the ability to re-synchronize a client application upon detection of message loss. Such is the case of the FIDS application of the OIS. If message loss occurs the FIDS client can re-synchronize by requesting an initial state and begin receiving events that update that state.

The performance/reliability tradeoffs of receiver- vs. sender-initiated multicast protocols are well known, offering stronger vs. weaker reliability vs. throughput, respectively [6]. For our environment, we have developed and are now evaluating a hybrid approach, in which attributes of both types of protocols are used to achieve a compromise for required reliability with high throughput. In this protocol, the receiver is responsible for lost message detection via sequence number analysis, and the sender buffers messages to accommodate retransmission requests. Periodically, the receiver issues a consolidated ACK for some sequence of messages, such that the sender can purge buffers.

To account for the implosion of these consolidated ACKS and NAKS by many receivers, NAK and ACK concentrators serves as a representative for some number of assigned receivers [5]. A sender is initiated and waits for a specified number of concentrators to join as representatives of the receivers. Next, the concentrators are launched, where they wait for a specified number of receivers to join their respective concentrator. When all expected receivers join the concentrator, the concentrator joins the sender and message flow begins. All ACKS and NAKS are forwarded via the concentrator.

9.1.4 Bandwidth Conscious Initial States

Some applications of the OIS require initial states on the order of 3 to 5 MB (non-compressed) in size. In situations where large numbers of display points simultaneously join the OIS event flows, this can have dramatic implications on the server load as well as bandwidth consumption. Since, display points are typically connected to the OIS via a WAN link on the order of 6 Mbs, the bandwidth alone allows 15 initial states per minute. Incorporating 90% compression increases the rate to 150 per minute. However, compression induces additional overhead and 150 initial states per minute remains far from sufficient.

Delta’s current OIS utilizes a hierarchically organized distributed server caching architecture for handling initial state load. Since caching and the computation and re-computation of initial states can have significant performance effects, especially for the highly available, replicated EDE, we are pursuing a scaleable and bandwidth conscious solution to this problem. Potential techniques include a combination of techniques, such as, PBIO, Forward Error Correction, Local Recovery, Periodic State Multicasting, and Distributed Initial State Servers.

10. Simulation

To simulate a possible future commercial deployment, we define our target simulation of an OIS that can support on the order of 100,000 capture points, with larger future systems having up to 1,000,000 capture points. We vary the event types, from 100 for a basic system, to 5,000 for a complex system. We model a basic, mid-end AP system as supporting 5 output devices and 5 input devices per AP, a complex system supporting an additional 500 output devices (e.g., passenger pagers) and able to deal with inputs from other sources (e.g., direct conversations with other APs). Consequently, the number of event types handled per AP varies from 50 to 500. Initially, we assume the current transaction rate of the commercial system. However, our research will experiment with much more demanding event systems, including those that emit continuous events, such as real-time aircraft status including pitch, yaw, and thrust, which would be tapped directly from an on-board control bus.

11. Conclusions, Status and Future Work

We have described the research and commercial opportunities presented by operational information systems, and their strategic importance to Delta Air Lines. An interesting approach to building new systems pursued by Delta is to tap its legacy operational systems, then reproduce desired images of operational information for new, mid-tier operational information systems (OIS). The idea is to create additional systems on which new business applications can be developed, without jeopardizing already existing systems and their operation.

Issues that arise for operational information systems include: (1) the efficient capture, transport, and delivery of events carrying operational data, in the wide-area environment in which Delta must operate, (2) the online derivation of consistent views of operational data, based on which various applications in the OIS may be run, including dealing with incomplete operational data and with diverse orderings and timing for event receipt, and (3) creating highly available OIS components.

We are continuing our research into the efficient, scaleable, and low latency processing and distribution of events, by evolving our communication/event infrastructures and OIS event processing and storage engines. In addition, we will pursue scaleable methods for wide area event distribution and collection. Finally, we are investigating the creation of highly available COTS cluster machines for the new mid-tier OIS' operational data engines and stores, so that these systems can offer the availability and reliability now offered by the existing legacy infrastructure.

12. Acknowledgments

We thank Mark Whitney, Scott Gosline, Deborah Callahan, and Rick Lawhorn whose vision, dedication, and perseverance have resulted in the novel OIS described here. Their work has already been recognized by receipt of the "Smithsonian Award". Thanks also to other members of the Delta team including Ron Byrd, Dean Compton, Bret Martin, and Mike Manatrizio for supporting our research. Patrick Widener and Mustaq Ahamad have also participated in our research and simulation activities.

References

[1] M. Ahamad. Causal memory: defnitions, implementation, and programming. Distributed Computing, pages 37{49, 1995.

[2] Greg Eisenhauer. Portable self-describing binary data streams. Technical Report GIT-CC-94-45, College of Computing, Georgia Institute of Technology, 1994. http://www.cc.gatech.edu/tech reports.

[3] Greg Eisenhauer, Fabian Bustamente, and Karsten Schwan. A middleware toolkit for client-initiated service specialization. In Submitted to PODC Middleware Symposium, 2000.

[4] Fabian Bustamente, Greg Eisenhauer, Karsten Schwan and Patrick Widener, "Efficient Wire Formats for High Performance Computing", To appear at SC'2000

[5] J. C. Lin and S. Paul. RMTP: A reliable multicast transport protocol. pages 1414-1424, March 1996.

[6] Sridhar Pingali, Don Towsley, and James F. Kurose. A comparison of sender-initiated and receiver-initiated reliable multicast protocols. In Proceedings of 1994 conference on Measurement and modeling of computer systems, pages 221-230, 1994.