RC23199 (W0405-001) May 3, 2004
Computer Science

IBM Research Report

An Active Adapter with Edge Cache Approach for Order Status Information Integration

Jih-Shyr Yih, Shiwa S. Fu, Shyh-Kwei Chen, Sebastien Houillot
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

An Active Adapter With Edge Cache Approach For Order Status Information Integration

Jih-Shyr Yih, Shiwa S. Fu, Shyh-Kwei Chen, and Sebastien Houillot

IBM T. J. Watson Research Center
19 Skyline Drive
Hawthorne, N.Y. 10532, U.S.A.
{jyih, shiwa, skchen, sebastien}@us.ibm.com

Abstract

Sharing information among multiple business operations and distributed locations in an integrated manner has always been one of the central issues in business activity and process management. The effectiveness of the entire commerce value chain for an enterprise is largely dependent on the satisfaction of the information requesters. Improving upon the typical data federation approach, we propose to add data caches in the adapters that connect directly to the information requesters for fast data retrieval. To demonstrate the proposed approach, we have implemented the new active features for the IBM Websphere Business Integration middleware. Experimental results show that the active adapter approach significantly reduces the average data access time under a real order status tracking system.

Keywords: Data Federation, Collaboration, Broker, Adapter, Edge Cache, Electronic Commerce.

1. Introduction

In an enterprise IT operating environment, it is common to have multiple business operations and large number of data sources located at distributed locations. Sharing information among these operations and locations in an integrated manner has always been one of the central issues in business activity [1] and process management [2, 3]. After recent years of dot.com hot pursuit of unsustainable revenue growth, businesses have returned with ever more focus on cost cutting by streamlining the IT value chain operations. In addition to cost cutting as a business competitive advantage, it is also recognized that backend data mess can cause front-end revenue lost. The satisfaction of the information requesters largely determines the effectiveness of the entire commerce value chain for an enterprise.

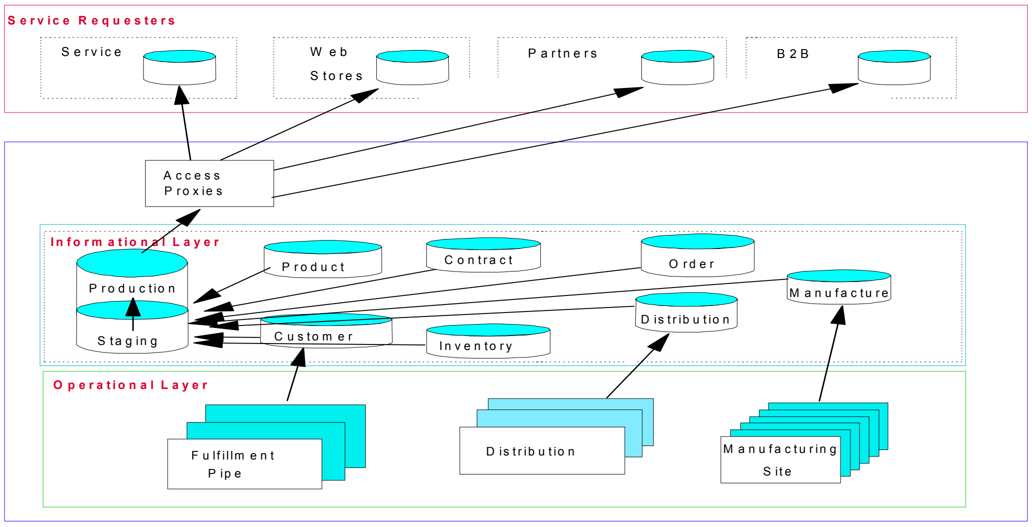

Figure 1: Order Status Data Consolidation.

The information integration problem can typically be addressed by the following approaches:

1. Data Replication [4, 5, 6, 7], which usually prepares a consolidated copy via data propagations from various data sources, and directs data retrieval to a production copy of the consolidated copy. Figure 1 shows an example of the architecture for order status, where data is first propagated to a staging database, and then replicated to a production database. However, in this approach, the overhead to replicate constantly can be prohibitive due to data size as well as the various propagation times. Moreover, the data requesters are usually not getting fresh data, subjected to by the frequencies for batch data propagations, say, hourly or daily overnight.
2. Data Federation [8, 9, 10], which builds virtual links without replicating data. Figure 2 shows a corresponding architecture for the order status information integration. In this case adapters are used to connect to application data sources. A collaboration broker then works with these adapters to service dynamic access and transformation. One major issue is that data retrieval can be time consuming, beyond an acceptable threshold.

This study reports experience gained from a business transformation solution with respect to commerce value chain development. In this case, customers need to check on all aspects of a master order, such as manufacturing, fulfillment and distribution events status before receiving the actual items. Customers may also review contract terms and conditions and check order history.

A new information integration mechanism is thus constructed, which improves upon the data federation approach by materializing data into small caches for shorter access times. The new insight in this approach is, rather than adding a data cache in the central broker, we propose to add data caches in the adapters that connect to the information requesters. We also observe that multiple requesters work with different parts of data from multiple data sources, so that adapters only need to subscribe to the relevant information. The multiple data caches also allow the accessing load to be naturally distributed amongst the adapters and be handled locally. By the pattern that customers rarely check individual detailed order status some time after receiving the shipment, we demonstrate how the adapter cache can interpret into the cached order data for cache management policies.

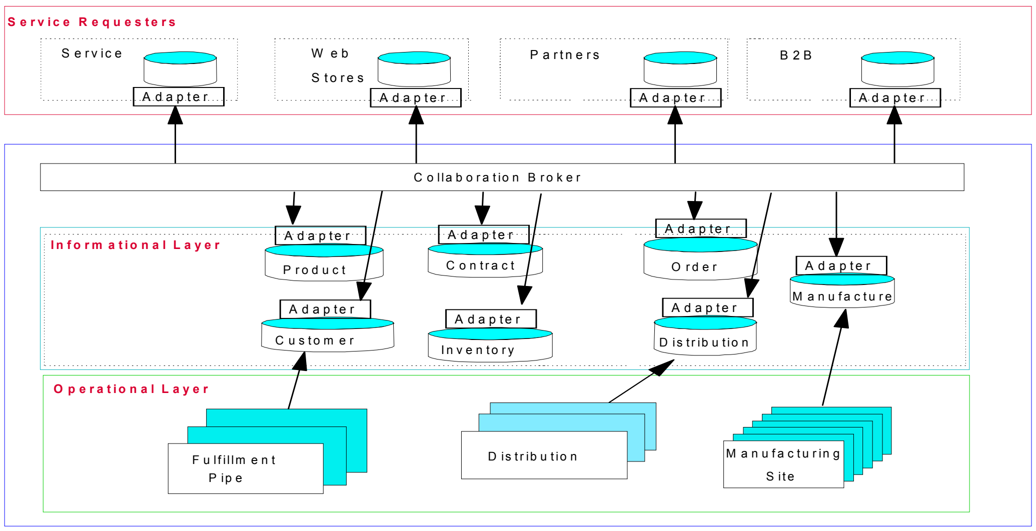

Figure 2: Order Status Data Federation.

The paper is organized as follows. Section 2 uses the IBM Websphere Business Integration (WBI) as a platform example to describe traditional application adapters. Section 3 discusses our proposed *active adapter* with *edge cache* and its architectural details. An application of active adapter to a real case order status tracking solution is described in Section 4, with performance results in Section 5. Finally, Section 6 summarizes the paper.

2. Websphere Business Integration Middleware

The IBM Websphere Business Integration (WBI) is a middleware that provides dynamic business process infrastructure for message delivery, document transformation, and application-to-application connectivity [11]. WBI includes graphical user interface tools that facilitate designing business collaborations (or business process logic), business object templates, object mappings, content relationships, and application adapters for external business applications.

WBI uses a common object model to facilitate business object exchange among all components and collaborations. WBI employs a hub-and-spoke architecture with a central broker called InterChange Server (ICS), and many application adapters. Each application adapter includes two components: controller and agent [11]. The adapter controller resides within ICS, and interacts directly with collaborations, and may translate business objects using *maps*. On the opposite end, the adapter agent resides in and interacts with the application, and may translate business data between external flat formats and internal pre-defined business object formats using *data handlers*. A controller and an agent may communicate using a messaging mechanism or the Internet Inter-ORB Protocol (IIOP), hence effectively connecting the ICS broker and the corresponding application. Dedicated application adapters such as for SAP R/3, PeopleSoft, SOAP, IBM MQ Series, relational databases through JDBC, are just some examples [12]. These application adapters can be easily modified and configured to fit new application requirements. For example, a JDBC adapter is basically configured to poll an event table for monitoring database updates and initiating new business processes based on the event trigger.

In solution build-time, WBI uses meta-data driven approach to increase the flexibility for designing and configuring adapter agents. Object Discovery Agents (ODAs) for several applications are included, which can generate Business Object definitions (BODs) automatically based on the application meta-data. The BODs normally include application specific information that can instruct the adapter agents about the methods to interact with individual applications. WBI also provides design tools for authoring and customizing collaborations, BODs, maps, data handlers, and custom adapters.

Figure 3 shows a typical way of initiating a business collaboration (or process). Initially at Step 1, application X updates its data source, which triggers an event caught by Adapter X. The Adapter X agent translates the event from application X flat file format into Application Specific Business Object (ASBO) format through data handlers (Step 2). The ASBO X is then transmitted to Adapter X controller through IIOP (Step 3). WBI maintains a common set of generic business object formats or BODs. Adapter X controller then translates through a map (Step 4) the incoming ASBO X into Generic BO, which is passed to the WBI broker for handling (Step 5). The WBI broker executes the business logic (or collaboration), which may involve communicating or accessing other applications through specific adapters. As shown in the figure, the data flow (Steps 6-10) is reversed to reach the destination application Y. Note that the data flow can be bi-directional, and there can be more than one source or destination application.

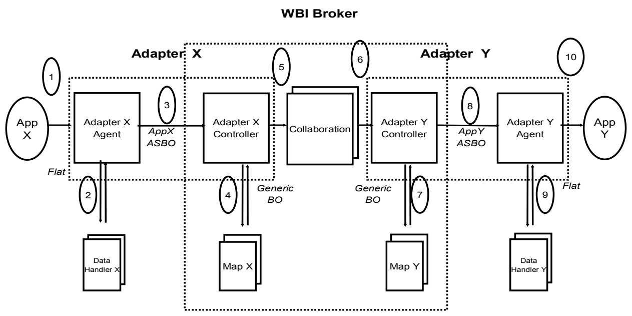

Figure 3: WBI Broker and Adapters Architecture and Process Flow.

3. The Active Adapter Approach

WBI employs a hub-and-spoke architecture, where every business process event needs to go through the central WBI broker for processing. To reduce the data access path for repetitive inquiries of specific data, we introduce a new feature that utilizes an edge cache for storing the most recently updated and viewed data. The new active adapter features allow a customer to frequently check the status of a purchase order that was recently placed.

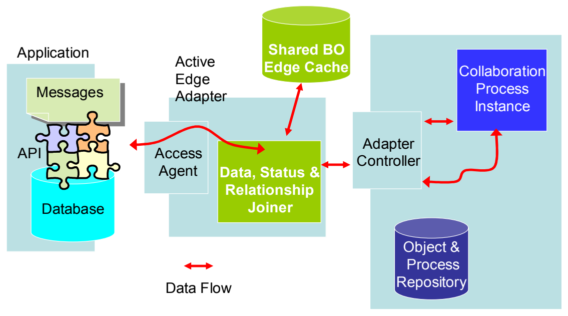

Figure 4: Proposed Active Adapter Architecture.

The approach also includes a *Data*, *Status*, and *Relationship* joiner component, where Data is for accessing the cache, Status is for summary, and Relationship is for mapping between event messages with the cache schema. Collectively, the component can decompose (or transform) a business object into sharable pieces and store them to the edge cache. Besides data transformation to/from the edge cache, the active adapter can maintain data persistency, aggregate and summarize data history (from edge cache), perform data filtering, handle cache management, and accept user subscription over specific business objects and viewing options. Figure 4 outlines the runtime architecture of the active adapter.

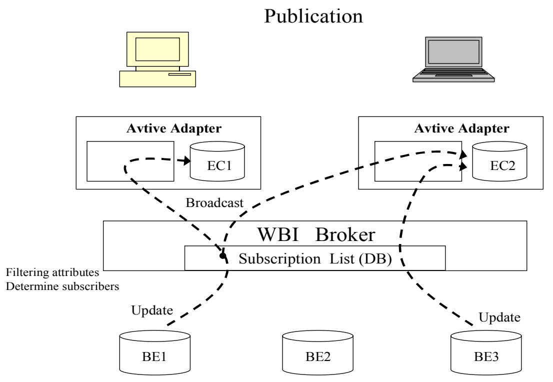

Figure 5: An Example of Data Propagation Flow from Data Sources to Active Adapters.

The active adapter approach includes two business processes: *data publication* and *data access*. The data publication business process is about actively publishing any data changes from the backend data sources to the edge caches. In addition to the current publish-subscribe capability, the WBI broker maintains a new subscription list for matching the data updates with the appropriate edge caches. For example, as are shown in Figure 5, updates at Backend Application 1 (BE1) result in the updates being propagated to both Edge Caches 1 and 2 (EC1 and EC2); while updates to Backend Application 3 (BE3) are routed to EC2 only. The targeted edge caches are based on the subscription list maintained by the WBI broker. The data publication business process is implemented by a new collaboration running in the WBI broker and initiated by an event trigger (due to update) at the Backend Applications.

The data access business process allows customers to inquire data. When an event is received, the active adapter firsts check its own edge cache to obtain the data. If there is a hit, the active adapter composes the result data in a format specified by the customers. If the data is not in the cache, then a regular data access business process follows. The process may involve aggregating the accessed data from multiple backend applications. The resulting data is also propagated to the edge cache as the most recently viewed copy. For example, Figure 6 shows two separate data inquiries from two customers, where one inquiry receives data result from the edge cache EC1, and the cache miss inquiry receives the data result that is aggregated and transformed by the WBI broker from three backend applications BE1, BE2, and BE3 as described in the following. The inquiry from the customer (Step 1) is checked by the edge cache to determine if the requested data is in the local cache (Step 2). For a cache miss, a request is sent to the WBI broker (Step 3). Based upon the list of sources, the broker might retrieve the requested data from many separated data sources (Step 4). The retrieved data is aggregated and transformed (Step 5) to the desired format. The resulting data is propagated to the edge cache (Step 6). Finally, the edge cache incorporates the results into local copy (Step 7.a) and presents them to the customer (Step 7.b). A simple and effective LRU (Least Recently Used) cache replacement policy is enforced for data replacement.

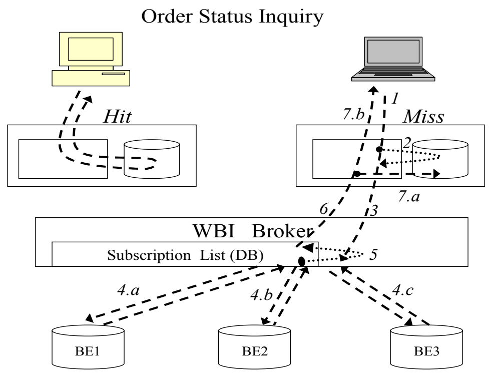

Figure 6: Examples of Cache-hit Local Edge Data Access and Cache-miss Remote Access Flows.

In solution build-time, our active adapter approach uses utility tools for cache schema generation, relationship discovery, transformation annotation, end-to-end debugging, and finer BOD attributes subscription capability. These tools are implemented as standalone applications or plug-ins using the Eclipse development platform, while the schemas are modeled based on the Eclipse Modeling Framework (EMF), a Java-based facility.

The proposed active adapter approach suits well for data retrieval, since frequently accessed data and most recently updated data are conveniently located at the edge cache. Cache management policy may also take into consideration the user access patterns and behaviors. For example, order status inquiries often occur after the order entry time and before receiving of the order shipment. Customer behaviors like repetitive inquiries can be satisfied, since the frequently requested data is likely already in the cache, which is updated only when there is a change in order status. Moreover, there is a pattern that customers rarely check individual detailed order status some time after receiving the shipment. Therefore, it is beneficial and practical to interpret into the order data for better cache management policy.

4. Case Study

Enabling purchasing orders is one of the most important business activities in electronic commerce. The process from placing an order until receiving the goods constitutes many commerce value chain activities. To keep track of the fulfillment progress for orders, the seller usually provides convenient order tracking services through a central server where buyers can check the individual order status. However, a buyer or a group of buyers in the same organization might be interested only in their own order status. It is therefore desirable to feed only the relevant information to the desired buyers. In this section, we describe a case study of a real order status tracking service. The performance of applying the proposed active adapter to the order status tracking is provided in the next section.

4.1. Business Scenario

A purchase order (to buy a batch of different workstations, for example) may contain many items (hardware components and software packages) that may be fulfilled by different fulfillment locations independently. As a result, a large enterprise usually has an order routing center. The center splits order transactions originating from the front-end order placement systems and routes them to the appropriate back-end fulfillment systems. Order is subject to cancellation and purchased item modifications requested by the customer, price change due to contract condition, and status updates according to the progress of value chain activities. The value chain activities include scheduling/releasing order for delivery, back order, invoice, etc. The processes of handling and tracking all those activities are referred to as order status tracking. To better understand the processes, we outline the order status flow, based on a real order tracking system implementation, in the next subsection.

4.2. Order Status

In Figure 7, an oval represents a status, and an arrow connecting two ovals indicating the transition from one status to another due to a triggering event. Initially, a new item in an order can be either “Accepted” or “Rejected”, arisen from incorrect or insufficient information provided by the end users. An “Accepted” item becomes “Scheduled” when the item is ready for delivery or “Back Ordered” if it is not in stock. A “Back Ordered” item becomes “Scheduled” once it can be fulfilled. Items that are in “Scheduled” status will then be “Released” to manufacturing for delivery. Once the product is delivered, its status becomes “Shipped”. Finally, the status becomes “Billed” if the back-end fulfillment system is ready to invoice the buyer; at this point, the order status cycle is considered completed. Note that an item cannot be cancelled once it is ‘released’ to manufacturing for delivery. The flow of item return is not in the scope of this paper.

To determine the overall order status, we need to sort out the priority among individual item status. The most advanced status for individual item in Figure 7 is “Billed”, followed by “Shipped”, “Released”, “Scheduled”, “Back Ordered”, and “Accepted”. Both “Cancelled” and “Rejected” are in the same priority as “Billed".

A purchase order containing multiple items can either be fulfilled at different fulfillment locations independently or be partially fulfilled at the same fulfillment locations. Each individual item thus could carry a different status. By taking into account unsynchronized status among individual items, overall order status is determined according to the highest priority of individual item status as depicted in Figure 7, where four more statuses are created: “Partially Billed”, “Partially Shipped”, “Partially Released”, and “Partially Scheduled”. As an example, an order containing one “Released” item and one “Shipped” item is considered “Partially Shipped” according to the priority rule mentioned in the previous paragraph.

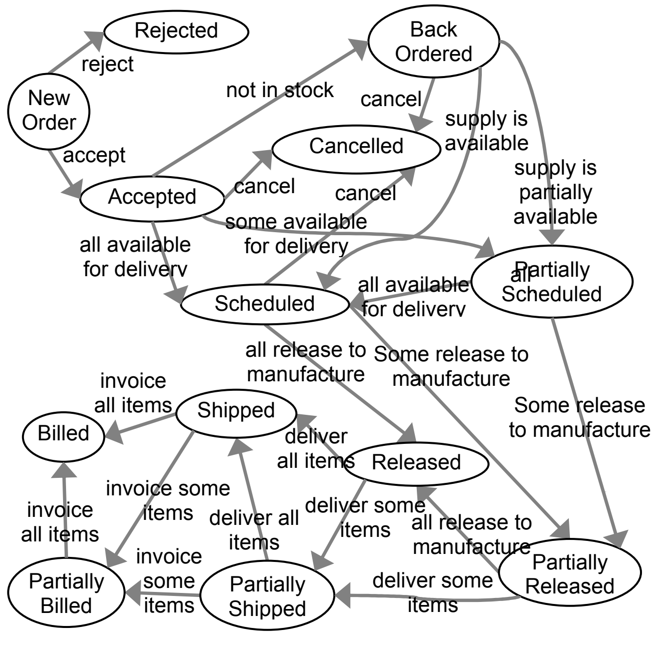

Figure 7: Order Status State Transition Diagram.

5. Performance Evaluation

5.1. Experiment Configuration

To measure the performance of edge cache data access against remote access, we collect samples from a real internal fulfillment center's data log. The fulfillment center receives order status updates from many SAP fulfillment systems in different geographic zones, distribution systems, and manufacture sites. We selected six samples with a wide range of data sizes, from three to 202 purchase items. The performance measure of interest is access time, which is defined as the average time from requesting the order status until receiving the response. All details (such as individual line items, invoice, and shipping) tied to the requested order are retrieved.

5.2. Performance Results

The results illustrated in Figure 8 are average values over runs. The measured access time for the remote data access is due to edge cache miss, whereas edge cache data access occurs when there is a data hit in the cache. It can be seen from Figure 8 that the remote data access time grows quickly as the number of items increases. It inflates almost 137% and 647%, when the number of items increases from 10 to 52 and from 10 to 202, respectively. The overhead of remote access is due to broker overhead, database access, and data transformation. In comparison, the edge cache access time increases only slightly when the number of items reaches 202, indicating fast data access once the required data is in the edge cache. This is especially important for the end users conducting repetitive inquiry such as order status checking.

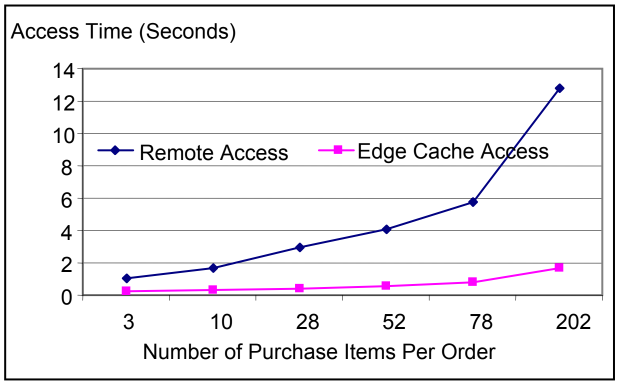

Figure 8: Performance Comparison of Remote and Edge Cache Data Access.

The data size for each purchase order in Figure 8 is shown in Table 1. It is 72.2 Kbytes for the order with 10 purchase items and almost 1.4 Mbytes for the order containing 202 items.

Number of Items in An Order	Data Size (Kbytes)
3	25.4
10	72.2
28	192.6
52	353.6
78	527.8
202	1358.5

Table 1: Data Size of Studied Samples.

The remote access time shown in Figure 8 can be further broken down into three components: broker overhead time refers to Steps 3 and 6 in Figure 6, database access time to retrieve the requested order (Step 4), and data transformation time (Step 5). The broker overhead includes communication time, adapter processing time, and collaboration thread setup time, whereas the data transformation, done by collaboration, includes data aggregation, data filtering, and data mapping. It is observed from Figure 9 that broker overhead time dominates most part of remote access time, followed by database access time. The data transformation time is relatively small.

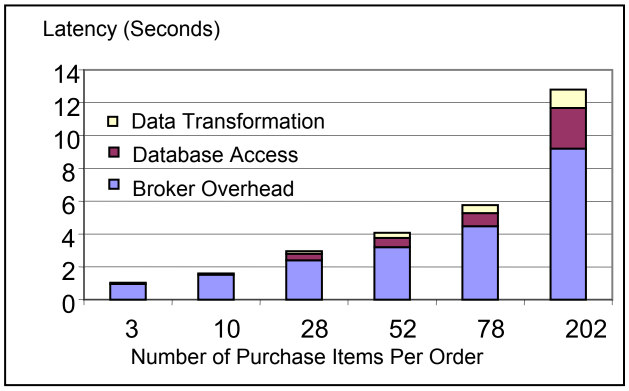

Figure 9: Breakdown Remote Access Time.

To better understand the cache miss behavior, we illustrate the latency of cache miss detection, remote access, and edge cache update. Cache miss detection time (referred to Step 2 in Figure 6) is the time to determine if the requested order is in the edge cache. Edge cache update time (Step 7a) arises from updating the edge cache with the retrieved order. As can be seen from Figure 10, the cache miss detection time is fixed and negligible small for all cases. On the other hand, edge cache update time grows significantly along with remote access time when the number of items increases.

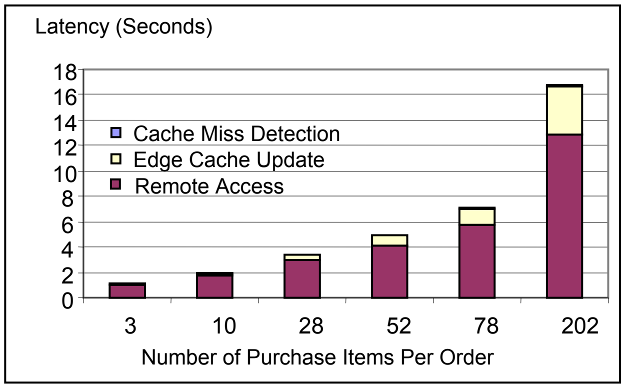

Figure 10: Latency Due to Cache Miss.

6. Conclusions

The information integration problem can typically be addressed by two approaches: data replication and data federation. The major concern associated with data replication is how to reduce the overhead of constant and frequent batch data propagations without sacrificing the information freshness, whereas the main issue lingers with data federation is the long data retrieval time that could go beyond an acceptable threshold.

In this paper, we demonstrated a new information integration solution add-on mechanism for the IBM Websphere Business Integration suite. The proposed active adapter improves upon the data federation approach by materializing data into small caches for shorter retrieval times. Instead of using a data cache in the central broker, the proposed approach adds data caches in the active adapters that connect to the information requesters. The proposed concept is further enhanced by the observations that multiple requesters usually work with different parts of data from multiple data sources, and adapters only need to subscribe to the relevant information. The multiple data caches allow the access load to be naturally distributed amongst the adapters and be handled locally, thus shortening data retrieval times. This paradigm is applicable for many other commerce applications beyond order status tracking system. Experimental results indicate that the data cache in the proposed approach significantly reduces data access time under a real order status tracking system.

Acknowledgments

The authors gratefully acknowledge the assistance from Margo Mao, Revlino Pascual, Charles C. Sung, Chandrasekara Narapireddy, Doru Ambrus, and Jerry F. Otruba.

References

[1] J.-J. Jeng, H. Chang, and J.-Y. Chung, “A Policy Framework for Business Activity Management,” *2003 IEEE International Conference on Electronic Commerce 2003*, pp. 238-245.

[2] C. McGregor and S. Kumaran, “Business Process Monitoring Using Web Services in B2B e-Commerce,” *16th International Parallel and Distributed Processing Symposium,* April 2002.

[3] S. S. Fu, S.-K Chen, J.-S. Yih, F. Pinel, and T. C. Chieu, "Web-based Sell-side Commerce Aggregation," *International Conference on Internet Computing 2002 (IC'2002),* pp. 303-310, June 2002.

[4] R. Alonso, D. Barbara, and H. Garcia-Molina, “Data Caching Issues In An Information Retrieval System,” *ACM Transactions Database Systems*, Vol. 15(3):359-384, 1990.

[5] O. Wolfson, S. Jajodia, and Y. Huang, “An Adaptive Data Replication Algorithm,” *ACM Transactions on Database Systems (TODS)*, Vol. 22(4): 255-314, June 1997.

[6] J. Sidell, P.M. Aoki, A. Sah, C. Staelin, M. Stonebraker, and A. Yu, “Data Replication In Mariposa,” *International Conference on Data Engineering (ICDE)*, pp. 485-495, February 1996.

[7] Barry Devlin, “Information Integration Distributed Access and Data Consolidation”, ftp://ftp.software.ibm.com/software/data/pubs/papers/etl.pdf, March 2003.

[8] H. Garcia-Molina et al., “The TSIMMIS Approach To Mediation: Data Models And Languages,” *Journal of Intelligent Information Systems*, 8(2): 117-132, 1997.

[9] M. Genesereth, A. Keller, and O. Duschka, “Infomaster: An Information Integration System,“ pp. 539-542, *SIGMOD 1997.*

[10] A. Tomasic, L. Raschid, and P. Valduriez, “Scaling Access To Heterogeneous Data Sources With Disco,” *IEEE Trans. On Knowledge and Data Engineering*, 10(5):808-823, Sep/Oct. 1998.

[11] “Technical Introduction to IBM WebSphere InterChange Server,” http://www-306.ibm.com/software/integration/wicserver/library/doc/wics420/welcome.html.

[12] S. S. Fu, S.-K. Chen, Y.-H. Liu, and J.-S. Yih, "SAP Integration Using Enterprise Application Integration

Software," *International Conference on Internet Computing 2003 (IC'2003),* pp. 547-550, June 2003.