BEST PRACTICES SERIES

Financial Services Information Systems

Editor

JESSICA KEYES

Boca Raton London New York Washington, D.C.

Also available as a printed book
see title verso for ISBN details

Contents

Chapter 2

Distributed Integration: An Alternative to Data Warehousing

Dan Adler

DATA WAREHOUSING HAS, PERHAPS, BEEN THE MOST COSTLY systems development in the history of financial services. The concept of data warehousing grew out of regional and departmental consolidation projects in which large relational databases were used to store relatively large quantities of data and make these data available to standard query tools such as SQL and various SQL-based interfaces. These queries were very useful in helping departments track customer behavior, P&L, and, in combination with applications written in VB, C++, and S-PLUS, they helped departments and regional entities within financial institutions track market trends, manage risk, and develop predictions.

Many managers, however, when they saw the success of the special-purpose database, were inspired to take this concept to the next level. "If we can do so much with a relational database in one department, I wonder how much value we could extract by representing our entire firm in a relational database?" went the reasoning. And, for expanding global trading businesses that needed to control the risks associated with numerous local portfolios composed of complex financial instruments, the centralized data warehouse containing "everything" was particularly appealing.

At most financial institutions, however, extending numerous departmental and regional data collection projects to create a "firm-wide" data warehouse representing a firm's entire global business just did not work. Wall Street — and retail banking's Main Street — are littered with tales of multiyear, multimillion-dollar data warehousing projects that were killed

0-8493-9834-7/00/$0.00+$.50

because they did not produce anything near what their architects promised. Problems include: transforming data stored in numerous formats, ensuring the data is "clean" and correct, developing and maintaining a firm-wide data model describing interrelationships between different types of data, managing various types of middleware to transport data from place to place, limited network resources, etc. And, of course, users became frustrated waiting for the "warehouse" and it associated promised functionality.

Still, extracting value from firm-wide information and controlling global market risk remain top priorities for financial institutions and their information technology departments, which are struggling to develop alternatives to data warehousing. One of the most promising such developments is Distributed Integration.

Distributed Integration is a new approach to integrating both firm-wide data and analytics based on Internet technologies, such as cascaded Internet servers and data caching. The "Distributed" refers to data and analytics that reside in numerous physical locations. The "Integration" refers to users' ability to access these disparate data and analytics through an *analytic browser,* which can be any application residing on any desktop which maintains an active connection to one or more *Distributed Integration servers.*

This chapter will describe how Distributed Integration solves some data integration problems that data warehousing projects do not always successfully address and show how Distributed Integration leverages existing desktop and intranet technologies to deliver this integrated data (and analytics) to large communities of internal and external users at a very reasonable price point.

THE HISTORY OF DATA WAREHOUSING

The firm-wide relational data warehouse has been proposed as a solution to numerous business issues facing financial institutions during the 1980s and 1990s.

First, in wholesale banking, per-trade margins have steadily declined in the mature foreign exchange and interest rate markets as the number of market participants increased and interest rate markets became more liquid. To respond more nimbly to market movements and to identify long-term trends, "data mining" — in which information is collected and analyzed to identify underlying patterns — became extremely popular. The data warehouse has been proposed as means of conveniently storing large quantities of historical and real-time data in order to facilitate the data mining process.

An important subset of the data mining issue for wholesale bankers and money managers is the management of time series data. These are data

that are periodically collected and time stamped. This sort of data is very difficult to store in relational formats because time series records — when expressed in tabular format — are very repetitive. Likewise, time series data are collected more or less continuously. Therefore, they tend to rapidly overpopulate relational databases and reduce performance. So, time series records are more often stored in file format for convenience. Blending time series and relational data for data mining purposes has often been an objective of data warehousing initiatives.

Another trend in wholesale banking has been the development of complex derivative instruments in response to the shrinking margins described above. These more complex instruments, often developed by local trading offices in response to specific customer demands, have raised control issues highlighted by a spate of "derivatives losses" stories in the mid 1990s as banks, corporates, and investment managers recognized the importance of, first, understanding their derivatives exposures in the context of their whole portfolios and, second, keeping a close eye on rapidly expanding foreign offices. Indeed, the downside of extreme decentralization was dramatically illustrated by the failure of U.K.-based Barings Bank. In the wake of the Barings scandal, many financial institutions turned to data warehousing as a solution to their "control and security" issues.

In retail banking, the consolidation and acquisition of numerous banks meant fierce competition, and product innovation and sales became ever more critical. Data mining became very popular in the retail deposit and credit card businesses, wherein effectively dissecting the customer's spending and living habits equals better product development and sales.

Finally, both retail and wholesale financial services providers became increasingly concerned with the rapid distribution and analysis of so-called "real time" data. Indeed, the global bull market, while providing satisfying returns across the board, also makes it more difficult for investment managers to differentiate themselves from the pack or from benchmark indices. Data warehousing capable of handling real data, then, became a Holy Grail for many firms.

CHOOSE YOUR WEAPONS

While data warehousing seemed like a sound solution to many of the above-mentioned business challenges at the conceptual level, the actual implementation of data warehousing solutions did not live up to the initial promise. First, this is because of the many difficulties associated with obtaining, converting, and transporting data that effectively limit the scalability of most data warehousing solutions.

Second, if it is possible to get data into a warehouse, it is often challenging to get the data out of the warehouse and into the hands of end users who need it.

For example, traditional relational data warehouses are built in conjunction with implementation of a large, costly "enterprise" system. Such enterprise systems are typically limited to a certain number of high-priority users. Providing general access to such a system is usually too costly due to licensing fees. And, if the enterprise system performs crucial operations functions, analysts and other nonoperations staff may not be allowed to apply query tools to the data warehouse; if they did, they could slow down mission-critical processes. When general queries are permitted against the data warehouse, they are often very slow. In most cases, specialized query tools must be purchased in order to identify, copy, and manipulate the relevant portion of warehoused data.

There are three major types of data warehousing implementations, and each has different benefits and drawbacks; these are listed below. However, while all of these solutions have been successfully implemented at the local level, none has successfully scaled up to the enterprise level.

Options for Data Warehousing

Numerous Interfaces. This is the classic, old-style data warehouse in which numerous "interface" programs are developed to create even more numerous download files which, in turn, may be uploaded to a centralized data warehouse during nightly, weekly, or monthly batch processing. While the interface method does possess a certain conceptual simplicity and in fact is sometimes the only option available to handle extremely proprietary — or antiquated — bits of data, it traditionally has a low success rate. This is because the sheer number of interface programs which must be successfully run and kept in sync is usually so large. This creates problems because, first, if something goes wrong in any one of these procedures, the entire data warehouse may become inaccurate. Second, these interfaces are often difficult to fix because they are often proprietary and, wherever there is IT turnover, impenetrable and undocumented.

Replication. Data replication can be used to create both a data warehouse and numerous redundant data sources by periodically copying new records across a suitable WAN. For example, let's say bank X, headquartered in New York, has a London office and a Singapore office. Let's also say the data warehouse resides in New York. As new trades are entered in London and Singapore, they are both stored in local relational databases and copied or "replicated" and sent through the WAN to the data warehouse in New York. The same replication procedure can be used to populate a backup data warehouse located in, say, London. While this method is straight-

forward and has worked well for medium-sized and smaller portfolios, it also has distinct scalability problems. This is often due to excessive network traffic created by the constant copying and transmission of each and every transaction. As the network slows down, it becomes difficult to keep the data warehouse up-to-date, and network failures can result in corrupt, incorrect, or incomplete data.

Middleware. Middleware is a catch-all term referring to "plumbing" software that is typically transparent to the user and may function as an engine for any or all of the following: data transformation, data integrity checking, transaction monitoring, data transformation, data distribution, and/or object/application communication. Some financial institutions have attempted to populate large physical data warehouses through the innovative use of multiple forms of middleware. Others have attempted to link together various types of middleware in order to create their own "virtual data warehouse" in which middleware supplies applications with RAM copies of relevant data.

Regardless of which sort of warehouse one is attempting to create, managing numerous forms of middleware has some substantial drawbacks. These include the extensive effort required to ensure that different middleware packages are compatible with each other and with critical data sources, scalability limits associated with various types of middleware (particularly ORBs when used in conjunction with a "virtual" warehouse), maintenance associated with upgrades, etc.

The Data Mart Variation

In response to the limitations of data warehousing, many financial institutions have abandoned multiyear data warehousing projects in favor of what is known as the "data mart" approach. Basically, the data mart approach refers to a data warehousing projects which is based on a number of local, regional or functional implementations. The prime benefit to this approach is that, unlike the traditional "big bang" style of warehouse building, users in specific regional or functional areas can actually see results within a more predictable period of time. And, for this reason, data mart projects have found a friendlier reception in financial institutions than "old style" data warehousing.

However, the final step of a data mart project is generally to combine all the data marts into a data warehouse by periodically copying these local databases in their entirety into a centralized, relational data warehouse. Often, it is easier to create a data warehouse from data marts than to build one from scratch because data marts are usually built with consistent technology, thus avoiding the need for massive data conversions.

However, these data mart-based warehouses do not usually work well for real-time analysis, in which constant data-copying would compromise performance both at the warehouse and data mart levels. And, a data mart-driven warehouse still comes up against the challenge of distributing the contents of the warehouse in a usable format to those who need them.

DISTRIBUTED INTEGRATION: A NEW WAY

Distributed Integration is a new way to integrate global financial data, analytics, and applications and quickly distribute them to a large community of users. Unlike most traditional data warehousing solutions, this does not require all data to be physically co-located in a single huge database. Instead, the Distributed Integration architecture relies on Internet technologies to create a virtual data warehouse that is optimized for both scalability and the cost-effective delivery of critical data to large user communities.

Data and analytics residing in multiple physical locations behave like a single, integrated virtual environment through Web-enabled *Distributed Integration servers.* These servers take advantage of Internet server cluster organization and data caching techniques and thus are configured for extreme scalability.

End users access the distributed integration virtual environment through a *Distributed Integration workstation,* which simply refers to any desktop which has a direct Internet connection to one or more Distributed Integration servers.

Almost any application (such as Excel™) residing on a distributed integration workstation can be enabled to view and manipulate integrated data and analytics; such applications are referred to *analytic browsers.*

Indeed, for the numerous highly paid analysts at most financial institutions who spend the majority of their time gathering relevant information to feed into their spreadsheets and then verifying that this information is clean, consistent, and correct, the analytic browser is truly a revolutionary concept.

Exhibit 2-1 provides a quick illustration of distributed integration. It depicts a global financial organization with two or more physical locations separated by a WAN or Internet connection.

The top location, say New York, has an equities group that maintains a database of equities using Sybase. They use a distributed integration server to make the data available to their own clients. They deliver analytics to traders at home through analytic browsers. They also have a group of analysts who write specific analytics and incorporate them into the Distributed Integration server, thus making them available to other analysts and to managers for control purposes.

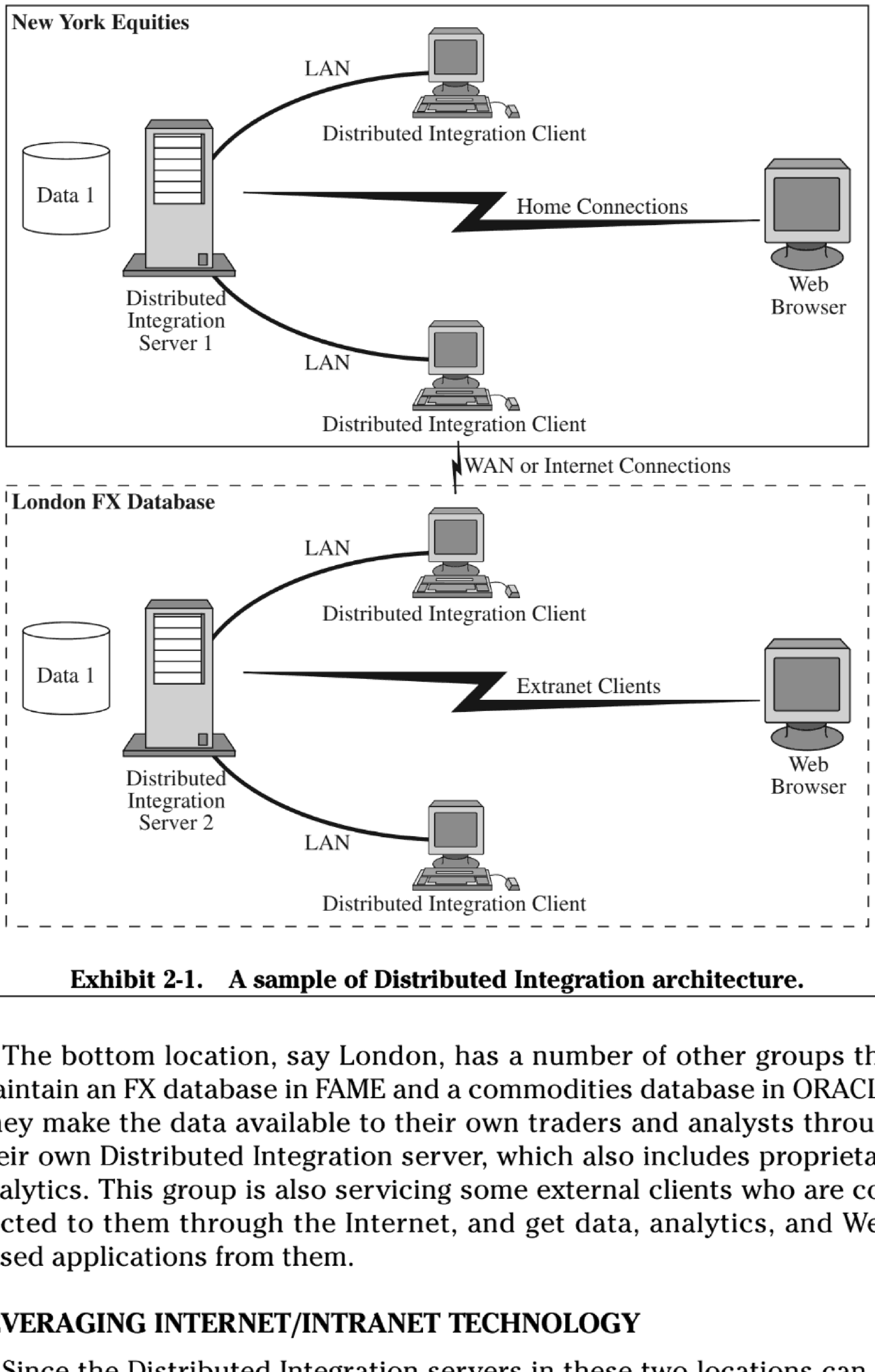

Exhibit 2-1. A sample of Distributed Integration architecture.

The bottom location, say London, has a number of other groups that maintain an FX database in FAME and a commodities database in ORACLE. They make the data available to their own traders and analysts through their own Distributed Integration server, which also includes proprietary analytics. This group is also servicing some external clients who are connected to them through the Internet, and get data, analytics, and Web-based applications from them.

LEVERAGING INTERNET/INTRANET TECHNOLOGY

Since the Distributed Integration servers in these two locations can be cascaded, as shown in Exhibit 2-2, each set of end-users can see the data in the other location. Moreover, since the Distributed Integration architecture

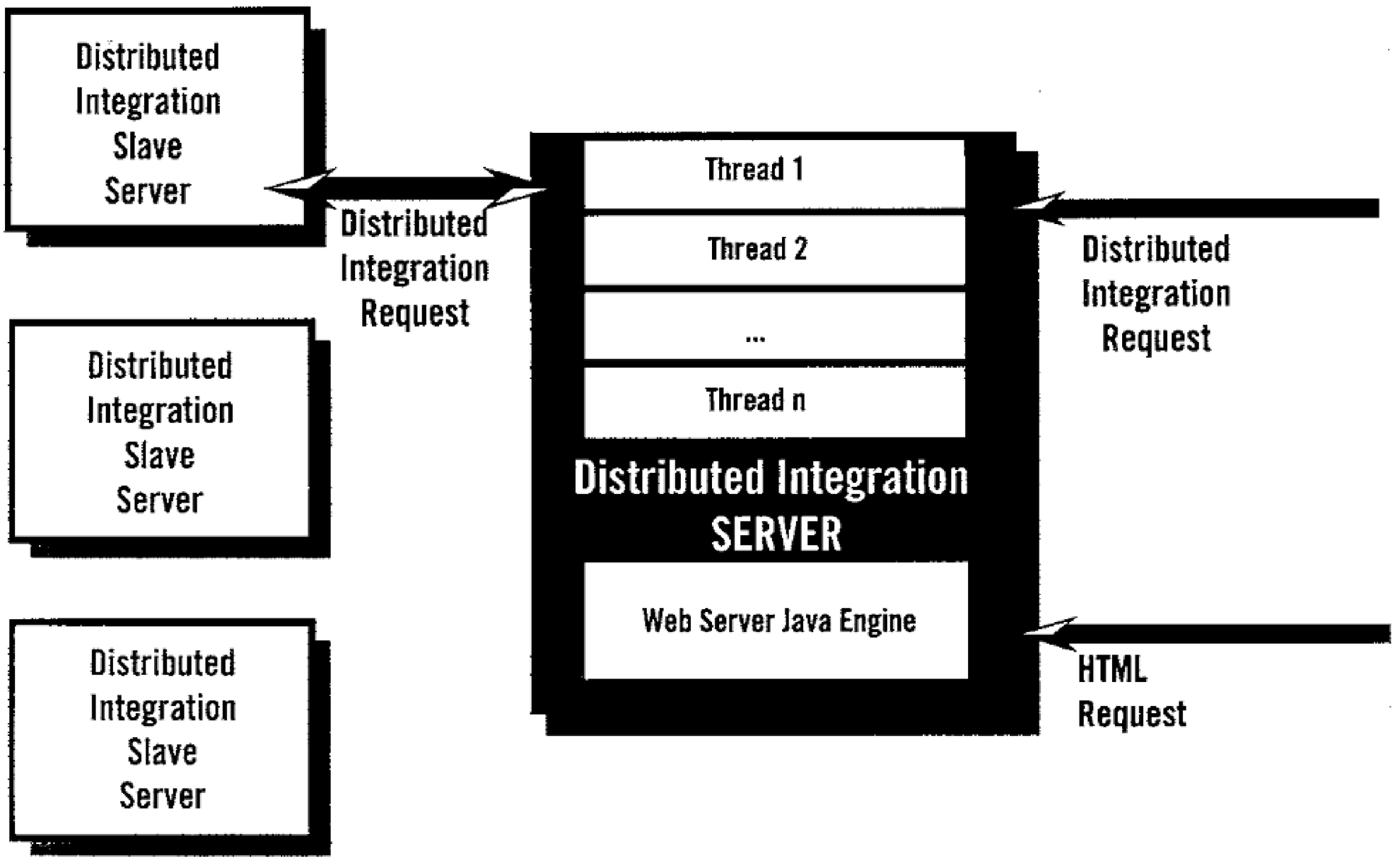

Exhibit 2-2. Distributed Integration takes advantage of cascaded server technology developed for the Internet to ensure fast and efficient management of Distributed Integration processes.

is characterized by transparent caching, the access times for local and remote data are almost identical on average. Data are transparently replicated to the location where they are used without significant IT involvement or administrative overhead; this is a far cry from the various data warehousing techniques described above, which require extensive IT resources to implement. As new locations are added with their own Distributed Integration servers, it is a simple step to make those servers known to the existing ones, thereby creating global multidirectional connectivity.

From an administrative point of view, the data, analytics, and applications are being maintained by the group that originated, and most "cares" about them, and they maintain it in whichever database they have chosen. However, because these data are integrated on the server side, local administration and autonomy does not come at the price of effective controls. This is a critical point for risk managers, HQ treasury managers, and compliance managers, who are often strong advocates of data warehousing initiatives which centralize data and in theory enhance controls.

The Distributed Integration server caching will "equalize" the access speed such that the back-end database does not become a bottleneck even if it's a small ACCESS database running on someone's desktop PC.

Distributed Integration makes it possible to then selectively share data, analytics, and applications across the entire organization without any additional integration work.

This architecture is highly scalable, since no single point can become a bottleneck as the number of locations increases. Thus, Distributed Integration avoids the scalability pitfalls associated with replication-based data warehousing solutions that tend to "clog up" a company's network.

For optimal scalability, the Distributed Integration server should be configured as a server cluster. This means it is not physically limited to a single process on a single machine. Such a server solution is implemented as a front-end multithreaded cluster manager process, with any number of symmetrical back-end servers that share the same configuration. Moreover, such clusters can always be cascaded amongst themselves, so there truly is no limit to the scalability of the Distributed Integration platform.

THE DISTRIBUTED INTEGRATION SERVER

Distributed Integration servers integrate historical and real-time data, metadata (i.e., data describing data), analytics, and applications. Because the server is capable of handling complex metadata and, by extension, almost any sort of data transformation, Distributed Integration is particularly well suited to address financial institutions' need for integrating time series and relational data.

In order to handle Web-based applications naturally and efficiently, Distributed Integration servers must also, by extension, be Internet servers. Indeed, Distributed Integration servers have been built as plug-ins to pre-existing Internet servers. As is consistent with Distributed Integration's reliance on the Web and Web-related technologies, it is most efficient to organize Distributed Integration servers as server clusters.

These Distributed Integration server clusters are stateless and connectionless, just like most Web servers, and should be capable of supporting the HTTP 1.1 keep-alive option for multiple related requests with a single connection for best utilization of network resources. This is also important from a user-functionality perspective; once data have been requested and cached, a user should be able to query that data without recopying it across the network. Likewise, incremental changes should be transmitted without the necessity of replicating the entire data set in which the changes originated.

The server cluster organization is one important reason why Distributed Integration can scale beyond traditional data warehousing solutions. As displayed in Exhibit 2-2, the front-end Distributed Integration server funnels each request to a number of preconfigured slave servers that actually handle the data and analytical requests.

Scalability is achieved by the fact that any number of slave servers can be added to a cluster, and they are automatically load-balanced by the master server. Fault tolerance is achieved by virtue of the fact that slave servers can be distributed on multiple machines, thereby reducing the chances that a single machine failure will halt the system.

A separate spool of threads handles standard Web server requests such as getting HTML pages, downloading Java applets, etc. This integrated architecture means that application, content, and data can be freely mixed to create a powerful application development and deployment model that is Web-centric and leverages the currently existing infrastructure.

Distributed Integration, it is important to note, performs best in a reasonably consistent hardware environment in which the server machines may be considered more or less interchangeable, and all the slave servers must have access to all the same data and analytics. This implies a shared configuration, as well as shared metadata and data caches. Thus, the user sees the entire cluster as a single server, with one URL address. And this URL address, then, becomes the user's gateway to the firm's total knowledge base.

The Distributed Integration server cluster architecture delivers:

- Performance — through the use of multiple machines coupled with load balancing.
- High availability — through redundancy of slave servers.
- Scalability — you can add more processes and machines as the number of users increases.

THE GATEWAY TO DISTRIBUTED INTEGRATION

Once one or more Distributed Integration servers have been configured to create a single point of integration for data and analytics, the next critical step is to cost-effectively distribute information to a large community of users. This has, in fact, been a challenge for those traditional data warehousing projects that have managed to get off the ground. A Java interface — or Distributed Integration gateway — can be used to connect desktop machines to integrated data and analytics residing on Distributed Integration servers.

When the gateway is active, a machine becomes a Distributed Integration workstation that is capable of "browsing" through all the information that is part of the Distributed Integration environment. Data and analytics can be accessed through a standard Web browser or by building software that connects existing desktop applications to the Distributed Integration environment. Applications that have access to Distributed Integration are called analytic browsers. It is possible to develop Distributed Integration add-ins capable of converting almost any application into an analytic browser.

In the financial services industry, however, one of the most useful applications of the analytic browser will be to connect spreadsheets to the Distributed Integration environment, thus allowing analysts to access integrated data without having to convert files or otherwise participate in the data-cleaning and -collection process. More advanced analytic applications will also become much more powerful when combined with Distributed Integration; indeed, one of the likely benefits of dynamically connecting end-users to an integrated environment is to speed up the pace of financial innovation.

LONG-TERM IMPLICATIONS OF DISTRIBUTED INTEGRATION

Distributed Integration, as it is implemented on a more widespread basis throughout the financial services industry, is likely to have important, even revolutionary, effects on how banks, brokerages, etc. do business in the future. And, as early adopters, the experiences of these financial services firms will serve as a model for other industries. Long-term implications of the Distributed Integration model include the following advantages.

Facilitates Financial Innovation

The desktop PC is synonymous with the modern workplace, and the PC-based spreadsheet application is synonymous with modern finance. However, the demands of today's businesses are straining the practical limits of PC-based spreadsheet analysis. A common illustration of this phenomenon is the fact that numerous highly paid financial analysts spend the majority their time gathering relevant information to feed into their spreadsheets and then verifying that this information is clean, consistent, and correct.

By providing integrated firm-wide data and analytics accessible through spreadsheets — such as Excel™ — Distributed Integration frees financial analysts from time-consuming data management duties and allows them to focus on value-added tasks. End users will also benefit from the opportunity to view and use the sum total of their firms' intellectual capital; they may combine this information in new and profitable ways.

Thus, the widespread adoption of Distributed Integration is likely to foster a new period of rapid financial and business innovation as analysts are simultaneously freed from the demands of data gathering and provided with accurate, integrated information.

Encourages a New Form of Corporate Organization: The Internet Management Model

As the number of Web-enabled workstations within corporations reach critical mass, organizations will be able to learn from the Internet and adopt new ways of managing themselves that go beyond the traditional trade-offs between centralization and decentralization. The Internet Management

Model (IMM) is one such method. It applies the Internet philosophy — that information and services are made globally available by the special interests that cherish them at low or no cost across a system which, to the customer, looks consistent, unified, and available on demand — to the corporate organization.

Distributed Integration, which allows local offices to manage and maintain their own mission-critical information while giving everyone access to the entire firm's intellectual capital, is an ideal vehicle for implementing IMM. Thus, rapid growth is sustained while top managers and business analysts have unprecedented access to the "big picture" presented by their firms' collective data and information systems. Islands of data ownership are eliminated.

By allowing firms to track individual and departmental contributions to firm-wide intellectual capital, it becomes possible to identify high- and low-performing areas. Simultaneously, IMM gives individuals and functional units much greater freedom to incorporate new data and analytics into their business activities without obtaining time-consuming approvals from central IT and business planning groups.

Allows Corporations and Financial Institutions to Manage the WHEN Dimension

As businesses' profitability becomes increasingly dependent on the timeliness and quality of the information which serves as the basis for key production, marketing, and strategic decisions, the ability to view, manipulate, and analyze data by time will become a matter of "life and death." Time-centric data, however, is often repetitive and difficult to store in conventional database formats. The Distributed Integration architecture is optimized for the rapid storage, transmission, and analysis of time-centric data as part of an integrated systems environment.

Author Bio

Dan Adler, *Chief Technology Officer at Inventure America, is an authority in analytical and financial software development. From 1990-1996 he was responsible for the development of RANGER's predecessor, TicShell, at Tudor Investment Corp., as well as a system for global risk management. He also conducted research on the application of advanced statistical, neural, and genetic search algorithms to trading. In 1984, after graduating with a BS degree in Computer Engineering from the Technion, Israel Institute of Technology, he was employed by several world class organizations including Motorola, where he designed VLSI Microchips and Mentor Graphics. At Mentor Graphics, Dan led a variety of CAD projects in the areas of simulation and analysis of complex integrated circuits. In addition to his Computer Engineering degree, he also holds an MS degree in Computer and Electrical Engineering from Rutgers University. Dan has published papers in IEEE journals and conferences in the areas of CAD and Genetic Algorithms.*