



These days, just about all applications need to pull data from multiple sources, and do it quickly, to complete transactions or deliver up meaningful, usable information. For companies that conduct business online, performance pressure is especially intense.

Consider Ameritrade Inc., which provides online brokerage products and services for self-directed investors: Clients are guaranteed that a marketable order will get done in five seconds or less. The volume can be staggering: as many as 350,000 trades per day, requiring well over a million transactions to complete.

Plus, this transaction demand does not flow steadily. "A significant amount arrives at market open and market close," said Ameritrade chief information officer Asiff Hirji. "It's very, very spiked and peaky."

eToys Direct Inc. (which operates eToys.com) contends with a different kind of peak: the Christmas season, when volume jumps 10- to 20-fold from other times of the year. During the holiday season, noted Web services director Mick Lester, the firm's Web site handles as many as 5.2 million page requests per day, which can peak at 210,000 page requests per hour.

So how do developers build cost-effective systems and databases that can maintain exacting performance levels and very fast response times while contending with huge peaks in (ever-growing) transaction volumes?

It begins with the architecture. "If you are dealing with high-volume, high-transaction-flow systems," said Hirji, "one of your starting premises has to be that a lot of this must be done without ever touching the database."

For Mike Kirschner, vice president of IT business services at Office Depot Inc., performance is best achieved via a

service-oriented architecture (SOA). "By that I mean loosely coupled systems based on open standards—discreet components or services rather than monolithic applications," he explained.

HORIZONTAL SCALABILITY

Ameritrade, too, is service-oriented; applications are distributed over "hundreds and hundreds of servers," said Hirji. "To scale those applications, you simply add more servers." Developers no longer need to think about how much hardware is needed to deploy an application.

This approach presents some challenges, however: how to distribute transaction flow across various servers, how to know which server is handling which transaction, how to spot when there's a problem with a server. Ameritrade has found several solutions:

Communicating asynchronously. "Design so that a module sends the message but keeps working, aware that it's still waiting for a response and able to react if it doesn't get a response after a certain amount of time," said Hirji.

Leveraging guaranteed message-delivery capabilities in development platforms like JBoss. The runtime envi-

much cross-module communication.

"For instance," Hirji said, "we have a piece of code whose sole job is to do nothing but receive quotes from the marketplace for all securities and broadcast these internally to our transaction processing modules. Each module listens, but it only listens for the pieces of data that it's actually interested in at that moment."

There are a couple of other things to watch out for as well.

"You need a high level of skills in the folks designing your services," said Office Depot's Kirschner. "You must get those services defined correctly because they all have to work together. Instead of building one large application, you're building a hundred small, flexible and extendable applications and trying to string them all together—it's more complex and your design must be better upfront."

He also warns about the need for a more granular approach to monitoring transactions. "In the past," he said, "you'd look at just one monolithic application and ask, 'What's my response time?' Now, you're looking at a hundred little components and asking, 'What's my response time with each one of those?'"

GOING STATELESS

"If everything is stateless," said Kirschner, "load can be spread across anywhere from two to 2,000 servers. When a developer writes a stateless component, he doesn't care if it's for two or 2,000—it's not his problem. Creating the necessary server farm is a problem for the operations people."

This, he said, enables him to leave database scaling to the vendors (his firm uses DB2, SQL Server and Oracle, among others), who are regularly adding features and capabilities like clustering, workload management and database par-

tioning. "I'm finding if I do my applications correctly and stay with a good proven vendor, the database layer kind of takes care of itself as long as I'm doing the basics—getting indexes on tables, optimizing my SQL statements, having database administrators that know how to put the databases onto physical devices."

At Ameritrade, there's less reliance on database vendors. "We're one of the few in the industry that owns and builds the platform from end to end," said Hirji. "None of our clients come to us because we use Oracle or DB2. They come for our proprietary systems and software. So it's up to us to make sure the application is efficient as possible."

How? "We try to do away with the database," Hirji explained, "and get to the point where, for interim state transitions, the transaction flow is actually stateless—so if the thing crashes, it can actually recover fairly quickly."

Thus Ameritrade avoids the performance penalty resulting from putting interim state transitions of the transactions into the database. Instead, the interim state transitions are stored in physical memory on the server and committed to the database only when the transaction is complete.

The result, Hirji said, is reduced latency and higher throughput. "You're basically optimizing for the server and the processor that you've got, as opposed to the database," he said. Additionally, he said, it's easier to recover from failure, and costs are lower thanks to more modest database investments.

Ameritrade is migrating much of its database infrastructure to open source. "For a lot of things we do," said Hirji, "we don't need some of the bells and whistles that come with today's databases" such as DB2 and Oracle.

'The need to scale is ingrained in everything we do, from technology to people.'

—Chris Cummings, chief information officer at eToys Direct Inc.

ronment will guarantee that if one module sends a message to another, it will be received, so developers don't have to write all that messaging control logic and can focus on the application logic instead.

Deploying a publish-listener (also called publish/subscribe) structure for

Best Practices for High-Transaction Environments

BY SUSAN MESSEHEIMER

Here are some best practices gleaned from Ameritrade Inc., Auctiondrop Inc., eToys Direct Inc., Mobliss Inc. and Office Depot Inc.:

Hire the right people. Auctiondrop CTO Andy Jeffrey said the right people are key to building quality systems. "We've hired people who have experience with high-volume, high-transaction systems for large retailers or secure transactional systems for financial services companies."

Design for high availability. "This

means that for each component you need at least two servers that can be included and removed from the application flow at will," said John Avery, CTO at Mobliss. "So the process is: Direct all traffic to server 1, update server 2, point all traffic to server 2, update server 1, direct traffic to both 1 and 2."

At Auctiondrop, separation of functionality helps. "When we add functionality to our processing center, it's not going to impact our customer-facing stuff and vice versa," Jeffrey explained. "For the customer-facing stuff, we typi-

cally do a live upgrade that won't impact the customer because we'll segment systems so that some are still running the old stuff as we add new stuff to what's offline, then bring what's offline back online and take off the other systems."

Build in the ability to shut down gracefully. "If you don't consider this early in the design, it's easy to paint yourself into a corner," said Avery. "Just killing the process is not the best approach. Applications need to make sure that intermediate queues are emptied or persisted before shutdown."

Angels in the Architecture

← continued from page 23

Office Depot's Kirschner said he tries to leverage open standards and open-source products as much as possible, though he warned that it's important to make sure there's adequate support.

DRAWN TO SCALE

"The need to scale is ingrained in everything we do, from technology to people," said Chris Cummings, eToys Direct's chief information officer. "We think about doing everything in really, really large volumes. Not just the Web site, but also the order management system, the financial system, the warehouse management system, the customer service system."

Techniques used in support of scalability include:

Caching, to reduce hits to the database. "We use BlueCoat edge cache servers that do caching of HTML pages based on the URL, and Akamai serves up all our image requests," said Cummings. "This is probably the single biggest factor in our ability to scale because when [the holiday] season hits, the number of searches goes up dramatically, and most page requests are cacheable. We also do distributed caching, more for performance than scalability."

Auctiondrop Inc., which helps consumers sell goods on eBay, also relies on caching. "There are many different types of caching that we'll do," explained Andy Jeffrey, chief technology officer and co-founder. To sustain an ability to process tens of thousands of queries and transactions per day, Auctiondrop uses data caching, bringing content into memory and keeping it there to reduce database hits. The company also uses rich content caching, which puts into memory not only images and text that are often used and little changed, but also more complex user interface elements—so these don't need to be rebuilt.

Database connection pooling, to control the number of database connections that are active at one time. During its 1999 start-up, eToys Direct struggled

with connections to its database. "It's easy to throw Web servers out there," said Cummings, "but when each of those servers needs to connect to the database, and the number of database processes starts exceeding a thousand or so on a server, you run out of memory and you go into a major tailspin."

The solution? "Reliable, scalable database connection pooling," he said. "We solved that in our custom Web server environment by deploying a custom Java database connection-pooling application on our application server layer. We can scale to thousands and thousands of Web server client processes, and those processes can share a more limited number of database connections."

Load testing, to ensure systems can handle the peaks. "A critical part of being ready for the volume growth is to test in advance that we can really handle it," Cummings said. "So we use an external vendor, Keynote, to stress our systems. Our goal is to stress them to roughly double what our highest projected sales plan is, to make sure that we're ready for pretty much anything. It's essential that we prove our performance capability before peak season, including integration points between systems—every integration has to be tested for performance. If we don't and something happens during peak, you can't deal with it because the volume never slows down enough to get your head above water."

DEALING WITH THE DATABASE

Large, monolithic databases make people like Cummings nervous. "We've broken the database out and partitioned it into different application uses," he said. "We've got one primary transaction database that takes care of shopping carts and checkout on the Web site; we have another that handles customer service activities, and we've got a couple of lighter-weight database boxes that handle query-only product requests."

eToys has also moved to smaller hardware, although it still uses a large Sun 6500 server. "We had big boxes at one stage," said Lester, "and we decided to go more with the small pizza boxes—Apache, Linux. If one dies it's not the end of the world, whereas if you're relying on a big box and it fails, you've potentially lost a third of your architecture."

Auctiondrop's Jeffrey added: "We've separated our database into customer-facing, transaction-processing systems—



'Now that you have the best database in the world, the next rule is: Use it as little as possible.'

—John Avery, chief technology officer at Mobliss Inc.

you separate those wherever possible so you can keep the transaction processing very lean and mean and very fast—and back-end reporting and analytic systems, which are offloaded onto another system. In effect, we've fine-tuned that customer system to be sort of a racecar database."

Jeffrey acknowledged that this approach adds some overhead, since it requires applications to locate needed data and make sure it's in the appropriate place at the appropriate time. "But you solve those issues once," he said, "and then you have the luxury of saying, 'Ah, we can keep what the customer needs over on the customer side and we can keep what back-end processing needs on the processing side.'"

Mobliss Inc., a provider of mobile media and marketing services, receives thousands of text messages per second. Chief technology officer John Avery has a rule of thumb for keeping things humming: "Spend most of your money and resources on your database solution—hardware, software and talent. No matter what else you do, the database will always be the critical component in your system. Now that you have the best database in the world, the next rule is: Use it as little as possible." ■

Use small, reliable, highly interchangeable modules. "When something works, use it wherever you can," Avery advised. "Only create a new module if the currently available ones can't meet your need, especially when it comes to tool components."

Avoid/reduce interdependencies. "Software should isolate critical functionality," Avery said. Developers need to consider reducing interlibrary (jar file) dependencies, and whether problems in the logging/monitoring system could or should bring down the whole application. They also should consider such things as intermediate queues, if certain modules can occasionally have long response times.

Keep your software lean. "The reason all the pipes and electrical conduits are visible in the hallways of a battleship but not a cruise ship is to facilitate easy monitoring and repair, even while under attack," Avery said. When it comes to software, this means reducing the number of wrapper classes and layers of APIs. "Code," said Avery, "should be thin and in your face." Beware of wrapper application programming interfaces (APIs) that hide layers where problems may lurk.

Employ stored procedures judiciously. "It's possible today to build almost any application completely inside the database with stored procedures," Avery said, "but my experience is that doing so is generally a mistake." Why? Because when your database maxes out, everything suffers, he explained. Instead, Avery advised scaling horizontally by adding application servers.

Yet stored procedures have their place. "There are great performance economies that you get by doing some things in stored procedures," Auctiondrop's Jeffrey pointed out. "You need to think through it before you just say, 'This goes in the database tier and this goes in the app tier.'"

Tune, tune and tune again. "We're constantly investing effort in performance tuning the Web servers and app servers," said eToys Direct CIO Chris Cummings. "And we make sure our database server is highly tuned from a physical database standpoint, a file system layout standpoint and a query performance standpoint."

Use a buddy system. At Ameritrade, programmers are paired: One person has primary responsibility for writing the code, the other is the sounding board and sanity checker. "What that creates," said Ameritrade CIO Asiff Hirji, "is a lot better code."

Tighten the development loop. "Our typical project is somewhere between six and 10 weeks long," said Hirji. "We don't do multimonth, multiyear things because we're convinced everyone is useless at estimating in technology. There are just too many unknowns." ■