IBM DATABASE MAGAZINE » Powering business information

Quarter 2, 2008 Vol. 13, Issue 2

http://www.dbmag.intelligententerprise.com/iduguserview/showArticle.jhtml?articleID=207801294

IDUG User View: Common DB2 Performance Pitfalls

By David Beulke

Build your applications right from the start by avoiding these typical missteps.

On the Internet, competition is just a click away. A delay in returning information to users can mean lost business. That's why applications must have performance considerations designed in from the start.

The most common problem I find in reviewing my clients' applications and infrastructure is insufficient connections, memory, or computing power to serve a distributed Java application. In service-oriented environments, processes are distributed across many local and remote servers. An analysis of one customer's Java application showed that the application had to connect to seven different servers to complete critical transactions. Some of these servers were already running at 100 percent utilization, so the application had to wait for connections. As a result, the application suffered unpredictable transaction results. Of course, the database is typically the first thing blamed; in this case, the poor performance wasn't the result of the database, it was the result of everything else.

When I asked for statistics, monitoring reports, and error logs for the distributed servers, I found that monitoring had been turned off to save resources. Once monitoring was turned back on, tracking the transaction "services" through their seven-server route and coordinating the different time zone timestamps on the logs was difficult.

In some of the services, no SQL error-code checking was occurring. So, whether the database SQL was successful or not, the service didn't report an error. This transaction and database integrity issue was resulting in bad data because the services were not handling the errors from the SQL and the database did not have referential integrity

relationships defined within the table structures.

A problematic Java application coding practice that should be corrected during application testing (common in object relational mapping, Hibernate, or iBathis application architectures) is putting an entire SQL result set into memory or a persistence layer. This approach usually works fine in a test environment where the database only has a few hundred or thousand rows. When it moves to production, the application retrieving millions of rows from the database either overwrites and corrupts memory or doesn't work and doesn't report an error. To avoid this problem, programmers must learn how to use the FETCH FIRST xx ROWS SQL cursor statements.

Web workload peaks can be five to 10 times greater than average transaction rates, so capacity limits can be reached very quickly. Ask four simple questions at the beginning of the application project to avoid capacity crises:

How many SOA transactions is the application configured to handle? You should get an answer like this one: "We're expecting 5M transactions doing 1M updates, 2M inserts with 15M-20M page views per day." Ask until you get enough information.

What are the servers' hardware components? The answer should include the number of CPUs/cores, the speed of the CPUs, memory allocations for the servers involved, and the number and speed of network connections.

What monitoring facilities are being used? Ask for a performance report at the start of the project to see what CPU load statistics and current network traffic utilization figures are available.

What services, servers, and database application DB2 packages are involved? Because Java applications are typically dynamic JDBC SQL, tracking application execution can be difficult. Ask the architects and application programmers for an application execution map. Ask which services or processes can call which other services. Build a matrix of this service-call information you can use when someone wakes you up at night with a "database problem."

Be prepared with the right infrastructure and monitoring information. Remember, your next "database problem" is only a click away. You'll find more performance tips at IDUG conferences (including the one in Dallas this month) and IDUG.org.

David Beulke [davebeulke@cs.com] is a DB2 consultant, author, and lecturer who specializes in database performance, data warehouses, and Internet applications.

Return to Article