Making Web applications more efficient

August 31 2012, by Larry Hardesty

Graphic: Christine Daniloff

Most major websites these days maintain huge databases: Shopping sites have databases of inventory and customer ratings, travel sites have databases of seat availability on flights, and social-networking sites have databases of photos and comments. Almost any transaction on any of these sites requires multiple database queries, which can slow response time.

This week, at the 38th International Conference on Very Large Databases—the premier database conference—researchers from MIT's Computer Science and Artificial Intelligence Laboratory presented a new system that automatically streamlines websites' database access patterns, making the sites up to three times as fast. And where other systems that promise similar speedups require the mastery of special-purpose programming languages, the MIT system, called Pyxis, works with the types of languages already favored by Web developers.

Alvin Cheung, a graduate student in the Department of Electrical Engineering and Computer Science (EECS), is first author on the paper. He's joined by his advisor, EECS professor Sam Madden, and by Owen Arden and Andrew Myers of Cornell University's Department of Computer Science.

A Web-services transaction typically involves both data retrieval—say, the flights on a given route with available seats—and computation—say, whether the difference in flight times would allow the traveler to make a connection. Typically, data is stored on one server, and the computation, or "application logic," is executed on another. The application server and the database might have to exchange information multiple times only to conclude that, no, a given itinerary won't work.

But if a few frequently used chunks of application logic could run on the database server instead, it would save time, by limiting the number of cross-server transactions, and bandwidth, since the sole remaining transaction could be sending the single bit of information "no."

But application logic and database queries are generally written in very different languages, which are optimized to handle different types of operations, so moving code to the database can require not only rewriting it, but also rethinking the way it's implemented. And it's difficult to split a program in two without introducing bugs —without, say, losing track

of which server needs to modify which variable at which point.

Finally, even if the programmer invests the time to establish that the partitioning of the program won't introduce errors, there's still the difficulty that the demands on the database server are constantly changing. During normal operation, the database server's CPU may have plenty of capacity to handle a little application logic. But a sudden spike in traffic could so burden the CPU that the extra computation puts it over its limit, causing much worse delays than shuttling data between application and database would.

Graphic results

Pyxis solves all three problems. It automatically partitions a program between application server and database server, and it does it in a way that can be mathematically proven not to disrupt the operation of the program. It also monitors the CPU load on the database server, giving it more or less application logic to execute depending on its available capacity.

Pyxis begins by transforming a program into a graph, a data construct that consists of "nodes" connected by "edges." The most familiar example of a graph is probably a network diagram, in which the nodes (depicted as circles) represent computers, and the edges (depicted as lines connecting the circles) represent the bandwidth of the links between them. In this case, however, the nodes represent individual instructions in a program, and the edges represent the amount of data that each instruction passes to the next.

"The code transitions from this statement to this next statement, and there's a certain amount of data that has to be carried over from the previous statement to the next statement," Madden explains. "If the next statement uses some variable that was computed in the previous

statement, then there's some data dependency between the two statements, and the size of that dependency is the size of the variable." If the whole program runs on one computer, then the variable is stored in main memory, and each statement simply accesses it directly. But if consecutive statements run on separate computers, the data has to make the jump with them.

"There's some cost to shipping data across the network, and there's some cost to every network round-trip that you do," Madden says. "So we want to find a placement of these nodes on the two different servers that minimizes the overall cost—or overall runtime—of the program."

In fact, Cheung adds, Pyxis finds several such placements of the nodes, some that push more computation to the database server and some that push less. "Our tool is able to dynamically switch between them based on the current load on the server," Cheung says.

Benchmarking

In experiments involving a standard set of simulated database transactions, Pyxis was three times as fast as a typical implementation, while cutting the bandwidth consumed almost in half. Moreover, the improvements it afforded were within a few percent of those that resulted from careful optimizations hand-coded by software engineers.

At the moment, Pyxis works with programs written in Java, which Madden says is the language favored by many commercial Web developers. But adapting it to other popular languages would require revising only the code that translates programs into graphical models; the rest of the system would remain the same.

"Usually, partitioning systems are automated, not automatic—automated in the sense that some programmer input is taken into consideration, and

then the system handles some of the more difficult aspects of the partitioning process," says Eli Tilevich, an associate professor of computer science at Virginia Tech. With Pyxis, however, "they partition things completely automatically, without requiring any user input."

"They have an program-analysis technique, called a partition graph," Tilevich adds, "which is an interesting innovation that has not been applied to prior systems." And while even successful academic research projects usually require a lot of work before they're ready for commercial implementation, Tilevich says, "their ideas, their technologies, certainly have commercial applicability."

In ongoing work, the four researchers on the Pyxis paper are collaborating with Armand Solar-Lezama, an assistant professor of computer science and electrical engineering at MIT, to make it even easier to streamline database-dependent Web applications. Most databases are written in so-called declarative languages such as SQL, which allow programmers to issue high-level commands, such as finding the largest value of some variable, without specifying a computational approach. The database system then automatically chooses the most efficient algorithm for executing the command, depending on the data characteristics.

Web programmers who are better-versed in Java than SQL will sometimes move large batches of data from the database to the application server, only to perform operations on it that SQL would have done more efficiently anyway. The researchers are developing a system that builds on Pyxis, dubbed StatusQuo, that can identify such inefficient application logic. It then automatically converts the application code into a SQL query, which the database then executes by whatever means it deems most efficient.

More information: people.csail.mit.edu/akcheung/papers/vldb12.html