InfoWorld

October 7, 2015

GET TECHNOLOGY RIGHT®

INSIDER

Review: Amazon Aurora rocks MySQL

Amazon's revved-up database service offers five times the throughput of MySQL and a formidable alternative to Oracle and SQL Server

THE 20-YEAR-OLD MYSQL DATABASE IS the world's most widely used open source RDBMS, and it's especially prevalent in Web applications built on the usual Linux stack. The common wisdom is that MySQL will give you good read performance in a multiuser, multithreaded scenario until your application becomes big enough to push the limits of the database.

At that point, you can try adding replicas to improve read performance, caching with memcached to improve read performance, or sharding the database into a number of distributed server clusters to avoid storage limits and improve read and write performance. However, sharding comes at the cost of ugly maintenance issues and possibly higher query latency.

Sometimes applications that get to this point are moved from MySQL to another, more scalable database such as Oracle Database with Real Application Clusters (RAC). While that works, the licensing costs of Oracle RAC are significant, and the conversion of MySQL stored procedures and other database objects to Oracle objects can be time-consuming, even with a tool such as Oracle SQL Developer.

Enter Amazon Aurora.

Inside Amazon Aurora

Amazon Aurora is a MySQL 5.6-compatible relational database service designed to deliver the speed and reliability of high-end commercial databases with the simplicity and cost-effectiveness of open source databases. Amazon claims that Aurora can deliver up to five times the throughput of standard MySQL running on the same hardware with four 9s (99.99 percent) reliability, and some Aurora customers have reported even better results.

For example, John Newton of Alfresco reports that the Alfresco document management system on Aurora scaled to 1 billion documents with a throughput of 3 million documents per hour, which is 10 times faster than Alfresco's MySQL environment.

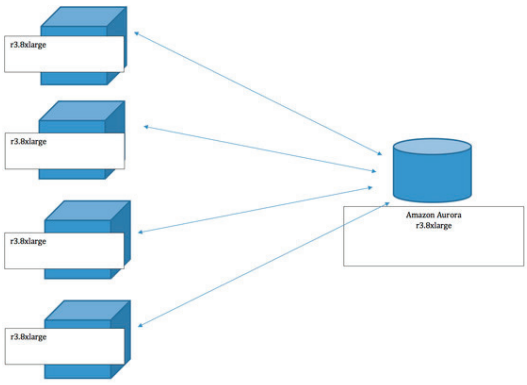
Part of the Aurora secret sauce is tight integration between the database engine and an SSD-backed, virtualized, fault-tolerant, and self-healing storage layer. Aurora automatically detects database crashes and restarts without the need for crash recovery or to rebuild the database cache. If the entire instance fails, Amazon Aurora will automatically fail over to one of as many as 15 read replicas. You can (and should) distribute your Aurora read replicas to multiple availability zones; Amazon claims a lag time of approximately 20 milliseconds for updating read replicas, which is much faster than MySQL read replicas.

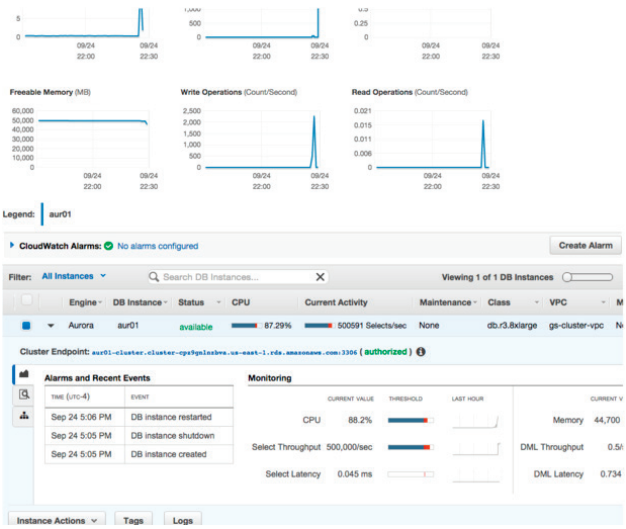
According to Amazon, you should use Aurora when you expect to have terabytes of data and heavy usage. Like all Amazon Relational Database Service (RDS) databases, you can scale your Aurora instances up and down as needed to match the load. A minimum Aurora instance gets two virtual CPUs and 15GB of memory at a cost of 29 cents an hour; a maximum instance gets 32 virtual CPUs and 244GB of memory at a cost of $4.64 an hour. Storage and I/O costs are in addition to the instance cost.

While Amazon's implementation of SSD storage for MySQL requires you to allocate your storage ahead of time to avoid introducing latency and jitter when the storage is expanded, Aurora is designed to expand organically from 10GB to 64TB of storage. Amazon claims that Aurora I/O operations use distributed systems techniques such as quorums to improve performance consistency, resulting in high throughput with low jitter.

Benchmark conditions and results

If you read my reviews often, you know what's coming next: I tested Amazon's claims myself. Specifically, I ran four SysBench instances against Aurora in the configuration shown in Figure 1.





Amazon supplied a white paper in which Amazon engineers described their Aurora benchmarks and gave results. They found that four client instances were necessary to saturate the database and Aurora throughput was as much as five times as high as MySQL throughput on the same hardware. Amazon quotes 535,000 read requests per second totaled from the four client instances on a select workload, as well as 101,000 write requests per second and 25,000 transactions per second on a write-heavy load involving inserts, updates, and deletes.

In my tests I attempted to create the same set of conditions as Amazon's engineers did with theirs. It took me several tries, as I stumbled over incomplete or incorrect documentation, but I eventually got it all to work, more or less. Typically for Amazon, the company gave me the blow-by-blow manual instructions first, and only pointed me at the wizards (for creating the Aurora instance and VPC network, and running the benchmarks) at the end of the document.

I recorded 493,000 read requests per second from the select workload, slightly shy of what Amazon reported, and 205,000 writes per second from the write-only workload, nearly twice what Amazon reported for the test. The screenshot in Figure 2 is an instantaneous snapshot of what the server monitors. My recorded numbers are calculated from the sum of the averages of the four clients as logged, and as such represent the results of my experiment.

However, I ran each test section for only five minutes; I found out afterward from Amazon engineers that their recent experiments have been done with 30-minute durations. They report that as the test length increases, the read performance keeps rising, but the write performance decreases. Amazon's reported 100,000 writes per second are more likely to reflect real-world performance than the 200,000 I measured.

A bigger, better MySQL

This level of performance is far beyond any I've seen from other open source SQL databases, and it was achieved at far lower cost than you would pay for an Oracle database of similar power. Add to that the fact that Aurora is a drop-in replacement for MySQL, used in a good fraction of Web apps, and you have all the makings of a winner.

Aurora required almost no tweaking, and setting up the instance using the launch wizard was painless. It took about five minutes for Aurora to become active after I started it and about the same amount of time to spin down when I terminated the instance. That isn't bad at all, and it bodes well for the more common scenario of bumping the instance size up or down in response to changes in load, which Amazon says takes a few minutes.

Overall, I was quite impressed with Aurora. It's worth considering if you have an application that's getting too big for MySQL, and it's certainly a more manageable solution than sharding your MySQL data or migrating to Oracle. I'm not sure I would try moving from a big Oracle or SQL Server installation to Aurora, but it's technically feasible -- and doing so might relieve financial pressures when the end of a maintenance contract makes it cheaper to move the data than to pony up for a new enterprise database contract.

— Martin Heller — Contributing Editor

Martin Heller is a contributing editor and reviewer for InfoWorld. Formerly a Web and Windows programming consultant, he developed databases, software, and websites from his office in Andover, MA, from 1986 to 2010. More recently, he has served as VP of technology and education at Alpha Software and chairman and CEO at Tubifi.

Amazon RDS for Aurora / Amazon

AT A GLANCE

A high-performance, highly scalable plug-in replacement for MySQL 5.6, Amazon Aurora is an attractive option for Web applications that have outgrown MySQL and a possible alternative to Oracle Database or Microsoft SQL Server for applications that don't need the special features of those databases and don't have a large investment in stored procedures.

Hourly rate of 29 cents (db.r3.large) to $4.64 (db.r3.8xlarge) depending on instance size, as currently quoted for the AWS U.S. East region for on-demand instances. You can scale instances up and down as needed. Charges are rounded up to the next hour. Storage is billed at 10 cents per gigabyte-month, and I/O at 20 cents per 1 million requests.

PROS

- High throughput compared to MySQL (~5X depending on application and data size)
- Low read-replication lag time (~20ms)
- Fast crash recovery compared to MySQL
- Compatible with MySQL 5.6
- Inexpensive compared to high-performance proprietary databases

CONS

- Aurora is not portable to other clouds, although MySQL is compatible and portable
- Write performance cannot be expanded beyond the capabilities of a single db.r3.8xlarge instance, although read performance can be improved by adding replicas

InfoWorld Scorecard

	Management	Performance	Availability	Scalability	Value	Overall Score
	25%	25%	20%	20%	10%	
Amazon RDS for Aurora	9	9	10	9	10	9.3