The Wikimedia Foundation

Streamlined Management

Jaime Crespo, from the Wikimedia Foundation, discusses the challenges of managing one of the world's largest collaborative knowledge projects. *By Mayank Sharma*

At Percona Live Europe 2019, I joked with Jaime Crespo, Senior Database Administrator (DBA) for the Wikimedia Foundation, that the Wikipedia project is essentially just one big database. You might expect that it takes hordes of DBAs to keep one of the world's largest knowledge pools constantly online, but you'd be wrong. Crespo shares the unique challenges Wikimedia faces in managing its servers.

Linux Magazine: Tell me about yourself and how you got started as a database administrator.

Jaime Crespo: My name is Jaime Crespo, and, by my name, it's easy to spot that I'm from Spain. I'm currently the Senior Database Administrator for the Wikimedia Foundation. A bit of follow up to that I have to do is that the Wikimedia Foundation is the nonprofit organization that provides support, mostly technology, legal and management, to the Wikipedia project and to a lot of other open knowledge projects that we maintain. The actual work is done by everyone, that is, our volunteers. I'm part of the SRE team (Site Reliability Engineering) that helps maintain the servers, so that volunteers and contributors can edit.

And concerning the second part, I was always interested in computing. Like many people, I started with more of a developer profile, but I quickly got more interested into systems, into the back end side of things. Almost by chance, I got involved into MySQL, because the only MySQL AB partner in Spain wanted someone else to deliver their courses.

LM: Was training your first interest?

JC: Yeah, I knew MySQL, and I had used it before, but that was when I got like very deep into it. Fast forward to like four or five, six years ago, that's when the Foundation had an open position for a database administrator. I was already involved with Wikipedia, as a volunteer. At the beginning, I was just to help the other DBA, but then the other DBA left, and that's how I got here. I was the only one for some time, and then we finally brought Manuel [Arostegui] onboard, who is the other current DBA.

LM: What is the scale of data at Wikipedia and what's the risk?

JC: Oh, as you joked earlier, Wikipedia is "almost just a database." That's obviously an oversimplification, but for the presentation here at Percona Live Europe 2019, we did a bit of research on "how large is Wikipedia?" Because we work with it every day, we know every single database server and every single database instance, but we never did the math of how large it is. In terms of relational data (so think MySQL/MariaDB), we have over half a petabyte of data in total. That's only the relational data, so think of that as wiki content, plus a lot of miscellaneous internal services for development and support, you know, those kinds of things that people normally don't see. But that's just the text kind of content. And then there is the other side that is media files. And I know that's much larger. Images, videos, and other non-relational data are stored in OpenStack Swift.

LM: Is Wikipedia a read-intensive deployment?

JC: Yes. So that's one of the benefits and curses of our kind of installation; the fact that we can cache very effectively. So anecdotally, I don't have firsthand data on this, but I think around 90 percent of the requests that we get are served directly from the edge cache. We maintain our own CDN [content delivery network] and 90 percent [of] the requests don't even have to reach the application or the database. Those other 10 percent, they normally arrive at the application, and we also have several layers of rather complex caching, and some of them end up on the application and end up querying the databases. In terms of that database traffic, again, I'm speaking loosely here, I think we've got around 400,000 SQL queries per second. We serve that with around 150 to 200 MySQL instances. They're not one to one to physical machines, because we do some consolidation.

So what happens if the database doesn't work? We are relatively safe with reads, because they end up in caching; 90 percent of the reads usually will continue to be served. The problem is editing. The reads and writes that cannot be cached will obviously have to go to the database. MySQL is essential for the editors basically. That's why we need high availability; we need redundancy on the database system.

LM: Has something like that happened during your time at Wikipedia?

JC: We regularly schedule downtime in terms of "we have to go to read-only." When that happens, the systems are still available for reads. People don't actually realize many of the times we do maintenance, because it's a very compartmentalized system, and even if there is an

unscheduled outage, it only affects a small part of the database. But it will be very rare to have like a full blown outage, because first we have redundancy within each wiki; but whenever there's a larger issue, usually it would only affect a small subset of the wikis. So that is my job: to make it more reliable. We're constantly working to make things more stable, more fast, more performant.

LM: In addition to managing the data, which features are essential to your deployment?

JC: So I'm talking only about the database. There are various things that we put in place. The first one is redundancy. Our topology is that we have, as I said, those different sections (replica sets). We have around 24 replica sets, which is basically groups of servers, independent servers which are physically isolated. So they're on different parts of the data center with its own redundant power and redundant network, and each of them is a replica of each other, and they are kept in sync. So if one fails, usually there's nothing to do manually. As usual, we need to do work on making things more automated. With such a size, the less the intervention the better.

One thing we are working on right now is redundancy at the data center level. So if the worst case scenario happens – a meteorite hits the US East Coast – I think we can be on the other data center in at most 15 minutes. We already have full duplication of all the resources on two different geographical locations. So we have, I think, a relatively serious approach to availability.

The problem right now is that the setup is active/passive for databases. So one of the data centers is less used. It's a complicated issue for the database which has most of the application state, but we are working right now on making the setup active/active; the primary data center for both reads and writes and for reads only on the second one. And that, in addition to availability, will improve also performance, for editors.

LM: What tools do you develop in house?

JC: So we DBAs are not into [the] heavy kind of development except in terms of "system development," so think automation, configuration management, and such. We at Wikimedia develop a lot of tools in terms of the application layer. One of the most famous is MediaWiki, and that by itself has a lot of features, such as automatic load balancing, and so in many ways, those features are done kind of by the application itself.

We DBAs are heavy reusers of existing open source solutions, for things like middlewares or proxies, rather than trying to develop our own. We're always keeping an eye out for something interesting. And in terms of the tools that we use, we try to use open source tools, because again, with the small resources that we have, we take all open source solutions, and we integrate with them. In my talk [at Percona Live Europe 2019], I talk about two of them: mydumper, which was initially developed by Domas Mituzas (former Wikimedia volunteer, who is now working for Facebook) and now mainly maintained by Percona, and XtraBackup, which is Percona although we use MariaDB, and the MariaDB version of XtraBackup (Mariabackup) is maintained by the MariaDB project. Those projects, as well as the database themselves, usually only get light patches from us DBAs.

We DBAs do some development, but in most cases they're not big, reusable tools. What we do normally in SRE is small utilities that glue existing ones to deploy them in our environment. What we give in exchange for that is, first, everything that we do is obviously always open source. So if you want to replicate our way of doing things, you can directly take it. In fact, I was talking to someone before, and they were shocked that in order for something to be in production in Wikipedia, it has to be in a public repository first. So everything is transparent.

There are, however, some bits of proper in-house developed standalone software. One of them is called Cumin [1], which is kind of an orchestration tool. We use it, for example, to schedule backups and to do remote copies. Another is called PyBal [2] that's our load balancer manager. It uses LVS (Linux Virtual Server) to load balance the traffic, but not for the database, only for the HTTPS traffic. Because as I said earlier, the database load balancing is part of the application layer. We also employ the main developer of our DNS server, gdnsd [3].

LM: Has Wikipedia always used MariaDB? What made you upgrade and how did you decide to do it?

JC: A bit of a disclaimer: I wasn't there when the decision was taken. Look at the blog post for more accurate reporting [4]. But let me give you my spin on it as I saw it. I understand why it was done, and I would probably have done the same thing. At that time, MySQL was going through a bit of a rocky phase with several purchases of several companies. So first of all, there was some fear in terms of the future of MySQL that on the kind of open source policy side. And the other thing was in the pure technical side [that] vendors like Percona and MariaDB were doing a lot of cool developments, a lot of really nice things. I remember, one of the technical reasons why MariaDB was needed was because it supported useful features much earlier than its peers, features like multisource replication – the idea of having several servers replicating to a single one. So the combination of, "Oh, now there is a MariaDB Foundation, which has pledged to support open source and we are a foundation too." So we went through several iterations including regular MySQL, and I think at some point we used Facebook's version of MySQL.

LM: Wikimedia used Facebook's version of MySQL?

JC: Yes, and I know that there was either some testing or some at production. But at that time, the number of servers was really small. I think maybe three servers or three servers and three backups, so relatively small. I think the previous DBA had worked with MariaDB, so he knew that better than the other products.

That's when I came in, and I finished the migration to MariaDB. At that time, we were migrating either to Maria DB 5.5 or MariaDB 10.0. We are now so large that migrations kind of overlap. By the time you finish one, there's a new version to migrate to. We just finished the move from MariaDB 10.0 to MariaDB 10.1.

Right now, we have to decide what's the next migration path. Just speaking from [the] technical side, we are seeing

MySQL doing a lot of cool things. For example, let me tell you about the specific features that we are looking at. The new data dictionary in MySQL is very interesting for us, because we have a lot of objects on certain databases, and there's a lot of improvements in that regard. So there are features that we are looking at on the newer versions, both MariaDB and MySQL, that we want to have.

LM: So let me see if I understand this correctly: You don't want to move from the current version of MariaDB to a later version, because, if you do that, it will be very difficult for you to migrate to another database, since MySQL and MariaDB are no longer drop-in replacements.

JC: One of the things in terms of the vision of the Wikimedia Foundation is that we don't like vendor lock-ins. Because we've had bad experiences with open source vendors, that they either decide to go for a non-free license or an open-core model. We don't have any problems with the open-core model as such, but the issue, in my opinion, is that some companies tend to focus on the non-free product, the one that is paying their payrolls. And there have been instances where we had to abandon a product, because the open source version available was either abandoned or didn't have the features that we needed. We want to always have an easy way out. This is why we have been talking with a lot of people about what they're doing, what they're using, what are their plans, and both users and vendors.

LM: You mentioned how privacy is an important consideration for the Wikimedia Foundation. You enforce SSLs and don't use public CDNs and public clouds. In terms of infrastructure, what complications do these steps introduce that you need to address?

JC: Increased complexity. So it would be very easy if we were in a different environment where privacy was an afterthought. We could just say, "Oh, let's put things in the cloud" and be done with it. The people we had to hire to be at the data centers and many of the things that we do will be taken care of automatically. The way we face it is "no, we want to host our own data." The data is not ours, in a way. Well, first we try to store the minimal amount of data, and, when I talk about private data for example, I don't even mean things like publicly identifiable information. We of course have passwords and user accounts, but typically private data means useful things to preventing vandalism, since you can even edit Wikipedia without an account.

We have a responsibility. It's not our data, those are not our edits, so we have to safeguard them. As a DBA, this is something that I'm very worried about. It has to be available, but also it has to be protected. As a foundation, our value is the trust the users have in what we do.

What does that imply in terms of infrastructure is that we have to host everything ourselves. So for example, typically you would have an external CDN; they would handle for you most of the traffic, the caching layers, and other things. The disadvantage to that, in most cases, [is that] they will ask you for your private certificates to serve the TLS termination. Basically, they will announce: "We are Wikimedia Foundation, send traffic to us," and we don't want to do that.

What are the consequences? Sometimes that creates a lot more work. We give us more work, because I have to handle bare metal. We create a bit of extra work, since we have to self-manage everything, but that is a value that our users demand.

My answer could look like as if "the cloud" is the way to go, and there are only advantages, while going bare metal is a "mere policy" reason. Not at all. Bare metal and self hosting also have a lot of advantages technically, like more independence and control on day-to-day operations, specially during outages. Being able to have a person physically on the data center has saved the day for us many times.

LM: What changes do you plan to introduce to the system in the future?

JC: As far as databases are concerned, one of them is backups. I particularly am concerned about the long-term survivability of the project. If we lose data, we lose the edits and that's a big deal. So right now I am working on focusing a lot on improving our backup system. Just to reassure people, we've always had a backup system. The problem that we have is that the backups haven't historically been fast to recover from. After the work we did, they are now, but we still have a lot of work to do.

Secondly, since we use Debian in most cases for our base system and now that we have completed the database upgrade from MariaDB 10.0 to 10.1 a few months back, we would like to migrate from Stretch to Buster. And as part of that, it is most likely that we will upgrade the database as well. The third thing we're working on always is improving automation.

LM: Finally, what's a typical day in the life of a Wikipedia DBA?

JC: I am not sure if there is a typical day, because there's always something happening. We don't have a lot of outages, but we may have from time to time things that may be not going as fast as they normally do, such as long running queries, plus there are things such as maintenance and upgrades, almost always ongoing. So there is two kinds of work. One is more reactive: response to incidents, user and developer requests. And then there are projects – more long term tasks such as major version migration or a roll in of a new feature.

For the reactive tasks, it mainly involves interacting on Phabricator (our ticket management system) [5] and answering questions and requests, as well as code reviews or small code patches. Regarding projects, I am currently focusing on backups, so that involves quite some more design discussions and larger programming sessions.

The good and, at the same time, the bad thing about our infrastructure is that there is always something to work on! ■■■

Info

[1] Cumin: *https://wikitech.wikimedia.org/wiki/Cumin*

[2] PyBal: *https://wikitech.wikimedia.org/wiki/PyBal*

[3] gdnsd: *https://github.com/gdnsd/gdnsd*

[4] Blog post on MariaDB migration: *https://blog.wikimedia.org/2013/04/22/wikipedia-adopts-mariadb*

[5] Wikimedia Phabricator: *https://phabricator.wikimedia.org*