LINCOLN LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

LLSC News

Lincoln Laboratory Supercomputing Center

March 12, 2024

The Cloud Outgrows Linux, and Sparks a New Operating System

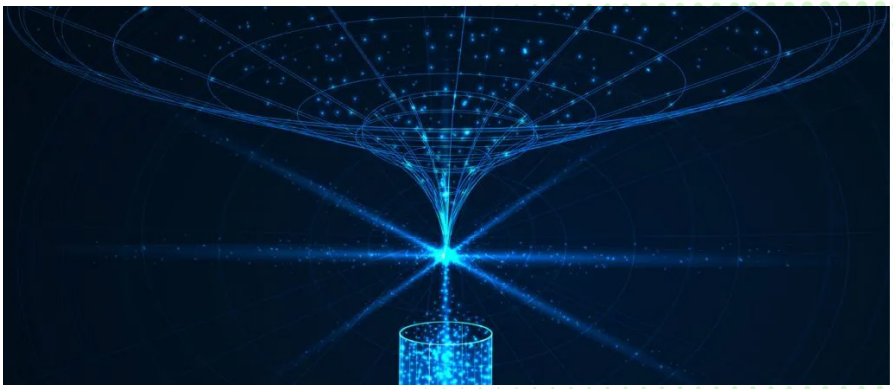
Timothy Prickett Morgan | nextplatform.com

Ultimately, every problem in the constantly evolving IT software stack becomes a database problem, which is why there are 418 different databases and datastores in the DB Engines rankings and there are really only a handful of commercially viable operating systems. But what if the operating system is the problem?

We are so used to thinking of the operating system as the foundation of the system that this kind of talk seems more weird than it does heresy, but make no mistake. When Michael Stonebraker and Matei Zaharia and a team of techies from the Massachusetts Institute of Technology and Stanford University are involved in creating a new operating system, it is definitely going to be heresy.

Stonebraker says that the spark for the idea for DBOS, which is short for database operating system, came when he was listening to a talk by Zacharia, who among other things was the creator of the Spark in-memory database while at the AMPLab at the University of California Berkeley and the co-founder and chief technology officer of Databricks, which has commercialized Spark.

"This talk was at Stanford three and a half years ago," Stonebraker tells The Next Platform. "And Matei said that Databricks was routinely orchestrating a million Spark subtasks on sizeable clouds and that Databricks had to keep track of scheduling a million things.



He said that this can't be done with traditional operating system scheduling, and so this was done out of a Postgres database. And then he started to whine that Postgres was too slow, and I told him we can do better than that."

Stonebraker, who is an adjunct professor at MIT and a member of the vaunted CSAIL research team that has brought so many innovations to information technology, would know.

Like all of the other database pioneers from the late 1970s and early 1980s, Stonebraker read the early relational data model papers by IBMer Edgar Codd, and in 1973 started work on the Ingres database while at Berkeley, and created the Postgres follow-on to it. Stonebraker was chief technology officer at relational database maker Informix, was one of the researchers on the C-Store shared-nothing columnar database for data warehousing (which was eventually commercialized as Vertica), and was part of the team that created H-Store, a distributed, in-memory OLTP system (which was eventually commercialized as VoltDB). More recently, Stonebraker led an effort to create an array-based database called SciDB that was explicitly tuned for the needs of technical applications, which think in terms of arrays not tables as in the relational model.

So Zaharia saying that Postgres performance was poor was like calling Stonebraker's child a bit slow. . . .

And rather than fight about it, Stonebraker and Zaharia teamed up to create an operating system based on a database rather than a database bolt on for an operating system.

In an interview with The Next Platform back in August 2017, we talked to Stonebraker about how hardware drives the shape of databases as the storage hierarchy changes, but this might be a case where a database operating system kernel might start

driving the shape of the hardware. (We will see how this DBOS idea takes off.) After that Stanford talk, Stonebraker and Zaharia played around with ideas, and built a prototype operating system on VoltDB to prove it would work; then they founded a company to commercialize the idea in April 2023 and secured $8.5 million initial seed funding to start building the real DBOS. Engine Ventures and Construct Capital led the funding, along with Sinewave and GutBrain Ventures.

What is breaking the operating system, and making companies like Databricks do weird bolt-ons of Postgres to maintain the state of Spark clusters outside of an operating system, is that the state of an operating system has gone up by five or six orders of magnitude in the more than five decades that Stonebraker has been programming. He gives a personal example. Back when Stonebraker was tooling around with Unix in 1973 on a DEC PDP-1141, it had 48K of memory and 20 MB of disk capacity. DBOS was tested early on running on the MIT Super Cloud, a cluster with 32,000 cores, a few terabytes of main memory, and many terabytes of secondary storage. There is just so much more stuff to keep track of, and so many more services running on that stuff, too.

"The state that the operating system has to keep track of – memory, files, messages, and so on – is approximately linear to the resources you have got," says Stonebraker. "So without me saying another word, keeping track of operating system state is a database problem not addressed by current operating system schedulers. Moreover, OLTP database performance has gone up dramatically, and that is why we thought instead of running the database system in user space on top of the operating system, why don't we invert our thinking 180 degrees and run the operating system on top of the database, with all of the operating services are coded in SQL?"

All of the investors in DBOS said that using VoltDB at the heart of this thing was not possible because it was not open source (there would seem to be an easy fix for that) and that because DBOS had to be open source, the underlying database would have to be, too.

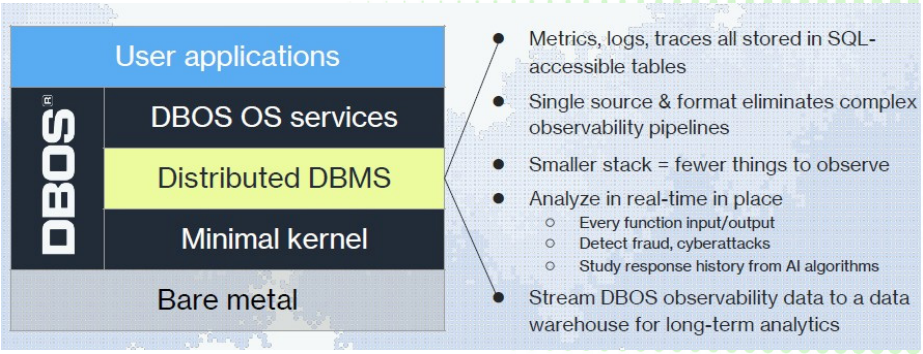
So the decision was made to use the FoundationDB distributed key-value store as the scheduling core of the first iteration of DBOS. FoundationDB was created Nick Lavezzo, Dave Rosenthal, and Dave Scherer, which was released in 2012, acquired by Apple in 2015, and open sourced by Apple in 2018. FoundationDB is a blazingly fast NoSQL database, which means that it does support the ACID properties of a relational database but which does not offer full SQL compliance. (Stonebraker tells us that DBOS eventually will do that, which seems to imply the underlying database engine will change.) Right now, DBOS has been tested running across 1,000 cores running applications coded in TypeScript, but Stonebraker says there is no reason to believe that DBOS can't scale across 1 million cores or more and support Java, Python, and other application languages as they are needed by customers.

The first iteration of DBOS runs on Amazon Web Services and uses the Firecracker microVM service, itself a stripped down KVM hypervisor running on a stripped down Linux microkernel, to create the user space for FoundationDB to run within. So technically, there is still Linux underneath DBOS. But nothing like the full blown Red Hat Enterprise Linux or SUSE Linux Enterprise Server that companies deploy or the homegrown, full-blown Linuxes that the hyperscalers and cloud builders have created for their own use. Stonebraker and Zaharia are working on ports to the Microsoft Azure and Google Cloud infrastructure, and it will be interesting to see how this is done. . . .

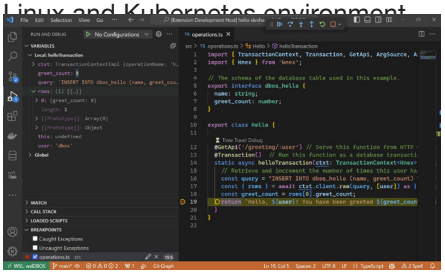
The point is, there is a minimal kernel underneath FoundationDB, which has device drivers, memory management, interrupt handlers, and some basic data movement functions, and the database services are written in TypeScript and their state tables can be queries in SQL. (Again, we would have preferred a relational database where DBOS services are written themselves in SQL, because that is a cleaner and funnier story.)

Stonebraker says that what he and Zaharia have really created is a transactional serverless platform that can run stateful applications. For now, DBOS can give the same kind of performance as that full blown Linux operating system, and thanks to the distributed database underpinnings of its kernel, it can do things that a Linux kernel just cannot do. And it can do all of these things without a full Linux OS and without Kubernetes containing things, and without having to bolt Postgres onto the side of the database middleware.

One is provide reliable execution, which means that is a program running atop DBOS is ever interrupted, it starts where it left off and does not have to redo its work from some arbitrary earlier point and does not crash and have to start from the beginning. And because every little bit of the state of the operating system – and therefore the applications that run atop it – is preserved, you can go backwards in time in the system and restart the operating system if it experiences some sort of anomaly, such as a bad piece of application software running or a hack attack. You can use this "time travel" feature, as Stonebraker



calls it, to reproduce what are called heisenbugs – ones that are very hard to reproduce precisely because there is no shared state in the distributed Linux and Kubernetes environment



and that are increasingly prevalent in a world of microservices.

Here is what the time travel screen looks like:

This time travel feature also lets you run new code against historical system state.

The other benefit of the DBOS is that it presents a smaller attack surface for hackers, which boosts security, and that you analyze the metrics of the operating system in place since they are already in a NoSQL database that can be queried rather than aggregating a bunch of log files from up and down the software stack to try to figure out what is going on.

By the way, if you look on GitHub to take a gander at the DBOS code, you will find code, but we do not believe it is for this particular instance of DBOS. It is for a DBOS project that was started by Peter Kraft and Qian Li, who were PhD students at Stanford and who we are guessing now work on the formal DBOS project.

DBOS Cloud, as the formal product is called, comes in two versions at the moment. There is a free version that can use the RDS Postgres service at AWS as an application datastore running on the db.ts.micro instance size only, and it is scaled to handle 1 million service calls per month. (We assume that means API service calls). This free tier holds operating system log data for three days and is allowed to have one developer on the account. Support is through Discord and the SDK only works with TypeScript and Postgres.

There is also a custom tier for DBOS, which we presume costs money, that can use other databases and datastores for user application data, stores more than three days of log data, can have multiple users per account, that adds email and Slack support with DBOS techies, and that is available on other clouds as well as AWS. Being a startup, new clouds, new languages, and human tech support will happen as enough people ask for them. No startup, not even one started by Stonebraker and Zaharia, can boil the ocean.

In a way, you really need to think of DBOS as a competitor to Linux, Windows Server, or Unix but to the AWS Lambda serverless function as a service stack. Stonebraker and Zaharia do:

	DBOS Cloud	Amazon Lambda	Amazon Lambda + Amazon Step Functions
Scalable serverless computing	✅	✅	✅
Fault tolerant workflows	✅	⛔	✅
Guaranteed once and only once execution	✅	⛔	⛔ (at least once only)
Built-in state management	✅	⛔	⛔
Time-travel debugging	✅	⛔	⛔
SQL accessible observability data	✅	⛔	⛔
Cyber-resilience	✅	⛔	⛔

One last thing. We know of operating systems that had an intimate relationship with a database, but this twist is actually a new one in that the operating system kernel/schedular is itself largely a database and services are created in database languages.

For example. IBM's System/38 and AS/400 minicomputers had a relational database at the heart of the operating system and in fact the database was the only file system allowed on these machines from 1978 through 1996, at which time IBM pulled the database out of the operating system and added in the OS/2 Parallel File System to give a POSIX-compliant, ASCII formatted file system for the AS/400. (Which is known today as the IBM i proprietary operating system.) The Pick operating system similarly had an integrated database, too. And of course, the "Longhorn" version of Windows Server 2008 was supposed to have the WinFS file system, which was based on a relational database, embedded in it, but that effort was spiked a decade and a half ago.

Which brings us to that one last thing: There is no reason why DBOS cannot complete the circle and not only have a database as an operating system kernel, but also have a relational database as the file system for applications.