Node.js Helps NASA Keep Astronauts Safe and Data Accessible

During a spacewalk in 2013, Italian astronaut Luca Parmitano found himself in grave danger—water was leaking into his helmet.

The water quickly migrated in zero-G to his eyes, ears and nose. Quickly, Luca was struggling to hear and see and he began having trouble breathing. Fortunately, his partner for the spacewalk, Chris Cassidy, was nearby and guided Luca into the airlock and safety.

Space is hard enough to survive in without water-filled helmets, so NASA halted all non-emergency spacewalks and launched an investigation to figure out exactly what happened. Contaminants introduced into the system had caused the water separator to malfunction and water backed up into the helmet just 30 minutes into what was supposed to be a 6.5-hour spacewalk.

The data on the spacesuit specs, maintenance schedules and previous excursions was spread out across too many different locations. When NASA went to collect all possible data on the situation as part of the investigation, it turned out to be a task of astronomical proportions.

Collin Estes is the Director of Software Engineering and Chief Architect at Mathematical Research Institute Technologies (MRI Technologies), who partners with United Technologies Corporation Aerospace Systems, the company that creates and maintains the spacesuits for extra vehicular activity (EVA) missions at NASA. Once the scope of the data problem was revealed, he and his team took on the challenge of creating an end-to-end system for data on the full lifecycle of spacesuits. But this is government work, remember, so it had to be done well, fast and as inexpensively as possible. He chose Node.js to make it work.

“NASA is always looking to find efficiencies and figure out how best to fund a specific project—no penny should go to waste,” Estes said. “We took that on as a challenge, to not just continue to support our systems today, but also to make this conscious migration into the cloud with this web API, and to do that with no additional contract cost. My model centered around creating a an API-driven Node.js enterprise architecture.”

> The system he is creating uses a microservices architecture with separate APIs and applications built in Node.js to move data related to the EVA spacesuits from three separate legacy databases to a cloud database. Now users can query that one database for everything, reducing the time to access a comprehensive set of data by about 300 percent.

Why Node.js?

For starters, Estes' team was already writing JavaScript-heavy applications. "When we selected Node.js, one of our selling points was centering around a single platform," he said.

Some other key reasons NASA choose Node.js were:

- The relative ease of developing data transfer applications with JavaScript, and the familiarity across the organization with the programming language, which keeps development time and costs low.
- Node.js' asynchronous event loop for I/O operations makes it the perfect solution for a cloud-based database system that sees queries from dozens of users who need data immediately.
- The Node.js package manager, npm, pairs incredibly well with Docker to create a microservices architecture that allows each API, function and application to operate smoothly and independently. This encourages each of the three legacy database systems to play nice with each other while the data is transferred to the cloud.

Siloes: Good for Rockets, Bad for Data

According to Sandeep Shetye, NASA's chief data architect, NASA has traditionally kept data siloed in individual missions mostly out of convenience.

That approach was already changing when the mishap occurred. Shetye said the mandate coming out of that investigation gave the agency the nudge it needed to rapidly break the data barriers.

NASA also creates a lot of data—hello, rocket science—and historical data is just as vital as the new data that's being created every day. And, if you can get at that information in a few steps instead of dozens, you've now freed up time for NASA's scientists and engineers to do awesome space science and engineering instead of hunting down manilla file folders.

> Shetye said hardware data processing used to be a 28-step and somewhat manual process (read: printing emails, looking up files). Now with an end-to-end system, which was created with the introduction of Node.js, it's been reduced to just seven steps.

Today no information related to EVA spacesuits is in one place, from concept and drafting data, information about the suits as they're created, and shipping data to maintenance scheduling and details on every mission astronauts have undertaken while wearing the suits. All this data is crucial for instances like figuring out what went wrong during EVA 23 and how to prevent it from happening again.

Centrally located information is also critical for building the next-generation of spacesuits, something that's not too far down the road given that NASA has essentially been using the same suit designs since the 1970s.

> Not surprisingly Estes' Node.js system rapidly won hearts and minds of NASA's engineers and developers.

Node.js Brings Big Data to the Cloud, so Astronauts Can Live and Work in Space

What MRI is building is a new enterprise database architecture centering around Node.js. That architecture is built on completely decoupled data module APIs and applications running as independent microservices. The APIs send document data from the legacy Oracle and Microsoft SQL Server databases to RethinkDB instances running on Amazon Web Services to be indexed via nightly Extract-Transform-Load (ETL) jobs.

"With those APIs in place, we've also transitioned to creating Node.js backend applications around this new data," Estes said. "The data residing in the RethinkDB, from legacy, is indexed nightly into Elasticsearch. We're able to correlate the metadata records from our structured databases, with the actual PDF and design documents."

Once these documents are migrated to the cloud and to the S3, the system performs Optical Character Recognition, and the results are included in the metadata in Elasticsearch, an open-source search engine.

"With those two things included, those nightly ETL jobs are able to create an integration point between structured data and paper documents and paper data," Estes said.

The next phase is to continue creating the microservices architecture and building Node.js data transfer and query applications, with the ultimate goal of replacing the legacy systems entirely. The catch: you can't disrupt day-to-day business while astronauts are still up in space, conducting spacewalks. So, users can still use the new system or the old system, depending on which is the right for the job, because the RethinkDB NoSQL database is sending data back to the legacy databases nightly as well.

"Users can enter data into one place, then make sure it works in the old system," Estes said.

"This is costly in terms of development time and complexity, but it's the best path to deconstructing a monolith 20-year-old application a little bit at a time. It will take longer, but ultimately that rate of change for our user community is what we can handle as a business and maintain the safety and the security of our spacesuit."

npm and Docker Keep the Airlocks Shut Tight

Estes said all high order database functions are divided into separate applications. What is today one Oracle forms and reports system will ultimately be a collection of 18-25 separate application modules and API modules. This keeps the data accessible and available, where the older databases would sometimes refuse to play nice with each other.

Estes is using Docker to make sure everything runs smoothly and suppress a versioning nightmare. Docker "containerizes" applications so everything they need to run—code, runtime, tools, system libraries—live independently and separately from the rest of the system.

"We have several Node.js services running on a single host, and some of our dependencies to connect to Oracle require older versions of Node.js than our other services would," Estes said.

"Docker lets us do that without having to code to the least common denominator. When in the past, one discrepancy would have held us back, now with our Docker implementation, each one is its own standalone sandbox and revisits whatever it needs to. It's not dependant on the others."

The goal is to make sure the data is moved to the cloud without errors, and that it's accessible at all times. When you've got the safety of astronauts on the line, little hiccups and service interruptions turn into life-and-death situations. From EVA data to astronauts up in space, Node.js helps ensure there's a safe home for everything and everyone.