Revolutionizing Audio with Descript and Temporal

This case study is based on an interview with Nicolas Gere-lamaysouette, engineer on Descript's audio transcription service. And - yes - we used Descript to transcribe the Descript interview!

Problem

Descript offers a revolutionary editing experience enabling users to modify spoken language in audio and video as if it's written in a Google Doc. The foundation of Descript is transcription, which is achieved using a blend of proprietary AI algorithms. Synthetically generating a speaker's voice, deleting filler words and producing shareable assets are all made possible by Descript's proprietary transcription process.

No transcription, no Descript. Therefore, this service needs to be fast and reliable. Regardless of whether you are transcribing a 5 minute clip or a 3 hour Clubhouse conversation.

When a user hits "transcribe", Descript fires off an asynchronous, multistage and parallelized process that involves re-encoding audio, chunk-splitting, external API calls, merging results that may potentially arrive out of order, and verifying their alignment.

Until recently, this complex multistage task was handled via a patchwork of handwritten scripts and queues built with Node.js, RabbitMQ, and PostgreSQL. Since application state was distributed across various queues and databases, it was hard to test end to end logic. This quickly resulted in Descript encountering production issues that were nearly impossible to debug. There were lots of production incidents, and when they happened it was extremely hard to understand which part of the pipeline was stuck.

> "We had one incident every week just on the transcription workflow because it was too complicated to maintain... You can test each worker separately, but it was impossible to test the logic for every new feature end to end, so we were afraid of doing any changes in that code path."

To code or not to code

This homegrown system was sufficient to get going, but as Descript scaled up, they quickly realized that they wanted an orchestration engine built for this job. The most important features they needed were **testing, debugging, monitoring, and scalability**.

The team evaluated a number of workflow orchestration tools, including Argo Workflows. However they soon realized that they strongly preferred the "Workflow-as-Code" approach of Temporal. Descript wanted to write, test, deploy and version control workflows with the exact same languages and tools used in the rest of their stack. This ability to model complex dependencies and error handling as idiomatic Go code was a better fit than simple JSON/YAML DSL's, particularly for local debugging and production monitoring. They also had scalability and latency concerns with Argo, because it was designed for CI/CD workloads so that each activity requires starting up a Kubernetes pod. Descript's data pipeline needs would involve spinning up thousands of pods, making that a nonstarter.

> "It was a game changing revelation - Temporal gave us the ability to test our workflows with unit tests, basically just writing code instead of writing JSON or YAML, which are completely untestable."

Migrating to Temporal

The migration process was relatively simple. Transcription is a self contained service that doesn't rely on Descript's core databases. So the process was to create a new workflow and begin progressively migrating workloads over using feature flags. The end results of the pipeline were checked via a single table, making it easy to effectively port over by effectively switching from handwritten queues to Temporal task queues.

Even though the migration was prototyped with a self hosted cluster of Temporal (using our Kubernetes helm chart), they decided to sign up for Temporal Cloud as an early design partner. As a small startup, they didn't want to manage Temporal themselves. "We migrated progressively using feature flags, and verified performance as we went, but we had confidence in the Temporal team so we didn't stress about load testing."

Looking Back

Temporal has helped Descript solve fundamental reliability issues with its core transcription loop. The frequency of production incidents has declined from once-a-week to virtually zero. Nicolas attributes this success to the ability to test everything with unit tests as well as normal end-to-end testing on a staging environment.

Testing is important, but Temporal opened up more opportunities since migration:

- **Orchestration of multiple Workflows**: you can flexibly reuse a workflow once it is written, allowing the transcription workflow to be easily composed with other APIs. This helped Descript develop a major transcription correction feature much more quickly than with the previous architecture.
- **Specialized Compute for Machine Learning**: Some parts of the transcription process use Python components that generate CUDA instructions - these must be run on workers with GPUs, and then reliably merged back into a general data processing workflow. Temporal's ability to route activities to multiple Task Queues made this straightforward.

- **Session Routing for Stateful Execution**: You can finetune the execution of sequential activities on the same worker, using Go SDK Sessions, which Descript will need (but doesn't yet use).

An unexpected advantage that Descript is enjoying is better observability of workflows, which **helps improve customer service** even when Descript is not at fault.

> "When a customer reports an issue, it's very easy for us to just put the Workflow ID into the Temporal Web UI to see what is going on... I can see a summary of workflow status and access the full event history for audit needs."

What's Next?

Now that Descript has had production experience running with Temporal Cloud, more opportunities are presenting themselves in a diverse array of use cases, including:

- billing and subscription management
- other CPU intensive backend workloads
- the (fan favorite) Identify Speaker feature

Other teams at Descript are waiting for our NodeJS client to explore migrating other forms of workloads.

> "Everyone in the company can see more and more use cases where Temporal would be useful."

Working with Temporal

Descript has found Temporal’s Slack very helpful. It can be difficult to adopt a core infrastructure technology like Temporal without help from the community and Temporal's core developers. It has also helped make the case internally to see other well regarded engineering teams ask similar questions and adopt alongside Descript.

> "The Temporal Community is great, the company is so helpful! Temporal is a complex piece of software and there are a lot of corner cases, so it is nice that the people who create Temporal are easy to reach to help us with our issues."