Flowstate

Crash-Proofing your Serverless Applications

Team 77
Team: Santiago Garcia Santos, Brian Williams, Katherine Wang, Victoria Zammit, Anna Bay
Advisor: Sebastian Angel

***Executive Summary**:*

Flowstate is an innovative middleware solution designed to solve a core challenge in serverless computing: state management. While serverless architectures offer cost efficiency, auto-scaling, and rapid deployment, they lack inherent state management, forcing developers to build custom, often error-prone solutions. Flowstate leverages the proven Beldi approach for exactly-once semantics to ensure fault-tolerant, durable execution of stateful serverless functions.

At the heart of Flowstate is a custom data structure, the LinkedDAAL, which is the heart of key operations such as Read, Write, and SyncInvoke, which enable efficient state retrieval, atomic updates, and reliable function invocation. These operations log activities for traceability and consistency, seamlessly integrating with existing Node.js environments via TypeScript bindings through N-API. An example setup with an HTML frontend and Rust-based backend highlights Flowstate's real-world application, ensuring minimal disruption to existing workflows.

Targeted at developers and organizations using serverless platforms like AWS, Google Cloud, and Azure, Flowstate simplifies state management without forcing a change in database or infrastructure, distinguishing it from competitors such as DBOS, OpenFaaS, or Temporal. Its cloud-agnostic design reduces development overhead, mitigates risks, and enhances operational reliability.

Financially, Flowstate uses cost-efficient testing and a revenue model that passes standard cloud costs to users with a modest markup, reflecting the added value of Flowstate for reducing engineering costs by 10-20x through robust state management. As the serverless market grows rapidly, Flowstate is well-positioned to empower organizations to harness the full potential of stateful serverless computing while minimizing complexity and risk.

Technical Implementation:

Serverless computing has emerged as a transformative paradigm in application development, enabling developers to deploy and scale applications without the burden of managing traditional server infrastructure. By abstracting away server management, this model offers significant benefits such as automatic scaling, cost efficiency, and rapid deployment. However, the stateless nature of serverless functions introduces a critical challenge: managing persistent state across discrete function invocations. In conventional serverless architectures, developers must create their own solutions to maintain state, a process that leads to huge increases in complexity and overhead, concurrency issues, and an elevated risk of bugs, often culminating in unexpected system crashes. Our project, Flowstate, directly addresses these challenges by building on the foundational concepts presented in the Beldi paper, which demonstrated a fault-tolerant, log-based approach to state management with guaranteed exactly-once semantics.

At the heart of Flowstate is our custom data structure, LinkedDAAL, which underpins the robust state management capabilities of our solution. LinkedDAAL is purpose-engineered to support three primary operations: Read, Write, and SyncInvoke. The Read operation is designed to allow developers to efficiently access the most recent state by retrieving the tail of the LinkedDAAL. Each read not only provides the necessary data but also logs the tail value into a dedicated ReadLog table, ensuring both traceability and consistency. Complementing this, the Write operation ensures atomic updates by carefully scanning existing logs and integrating new values in a manner that preserves concurrency and prevents race conditions. Perhaps the most critical component, SyncInvoke, facilitates the seamless invocation of stateful serverless functions (SSFs). This operation maintains transaction order and meticulously records invocation metadata in an InvokeLog table, thereby guaranteeing durable execution and robust crash recovery with exact state reproduction.

To demonstrate the practical utility and integration ease of Flowstate, we developed a comprehensive demo that showcases its full range of functionalities. Central to this demo are the TypeScript bindings we created using N-API, which expose our library as a pre-compiled Node.js addon. These bindings allow developers to interact with Flowstate in a familiar coding environment, thereby reducing the learning curve and facilitating rapid adoption. The demo itself features an HTML frontend that connects to our Rust-based backend via the NAPI-RS library, providing stakeholders with an intuitive, real-world example of how stateful serverless functions can be seamlessly integrated into existing workflows. Additionally, our engineering approach emphasized robust testing practices: we leveraged LocalStack to simulate AWS Lambda functions and DynamoDB locally. This local mocking framework not only ensures accurate testing under near-production conditions but also effectively eliminates development costs associated with cloud resources.

***Value Proposition*:**

Serverless computing has revolutionized the way applications are developed and deployed, offering scalability, cost-effectiveness, and ease of use. However, introducing statefulness in serverless architectures remains a significant challenge, often requiring developers to create complex, custom solutions to manage and continuously validate state across stateless function executions. This gap in the ecosystem creates a high risk of bugs, increased development time, and potential failures in dynamic application workflows. Flowstate directly addresses this problem by abstracting state management complexities, offering developers a robust, fault-tolerant library for stateful serverless functions that is both flexible and cloud-agnostic. Built on the foundation of Beldi, a proven log-based library/runtime, Flowstate provides seamless integration with popular cloud providers such as AWS, Google Cloud, and Azure, empowering organizations to use their existing databases while enhancing fault tolerance and crash recovery.

Flowstate's key value proposition is its ability to simplify the developer experience. This not only reduces development overhead but also minimizes the risk of "death by a thousand cuts", the accumulation of small, hard-to-detect bugs that can lead to system failures. Flowstate enhances operational reliability by ensuring durable execution, allowing serverless functions to resume from their exact state after crashes. This allows developers to save both time and resources on execution overhead and state management, allowing them to spend additional resources elsewhere. Additionally, its middleware approach ensures compatibility with a wide range of databases and cloud platforms, unlike competitors such as DBOS, which require developers to adopt new databases or frameworks. This flexibility reduces the migration barrier for organizations and supports diverse workflows without locking users into a single platform or technology. From a market perspective, Flowstate fills a critical gap for serverless developers seeking efficient, transparent solutions for stateful functions. By addressing pain points such as crash recovery, multi-cloud integration, and minimal disruption to existing codebases, Flowstate offers tangible value to its target users. Ultimately, Flowstate empowers organizations to unlock the full potential of serverless computing while mitigating the challenges of state management, setting the stage for broader adoption and innovation in this rapidly growing domain.

***Stakeholders*:**

Flowstate's stakeholders include developers, cloud or database providers, business leaders, academic researchers, and end-users of serverless applications. Developers are the primary stakeholders, because they directly interact with Flowstate to simplify the complexities of state management. These

users, including individual developers, small teams, and large organizations, benefit from Flowstate's abstractions that reduce development overhead, mitigate the risk of bugs, and enhance workflow reliability. For developers accustomed to serverless environments like AWS Lambda or Google Cloud Functions, Flowstate eliminates the need for custom solutions, offering a seamless, user-friendly library that integrates with their existing infrastructure while ensuring durability and crash recovery. Cloud providers such as AWS, Google Cloud, and Azure represent another critical stakeholder group, because Flowstate enhances the utility and versatility of their serverless platforms by addressing a major pain point, state management, potentially driving increased adoption and customer retention. Similarly, database providers, such as DynamoDB, have an interest in Flowstate's success, as it enables wider adoption of their technologies by making stateful operations easier and more reliable. Business leaders in organizations, such as CTOs or project managers, are also key stakeholders. They prioritize cost efficiency, scalability, and operational reliability, all of which are supported by Flowstate's flexible design that works across multiple databases and cloud platforms, reducing migration barriers and offering compatibility with existing systems. Flowstate's emphasis on transparency and developer ease-of-use minimizes learning curves and accelerates implementation, aligning with the goals of these executives to optimize resources and time-to-market. Academic researchers, particularly those focused on distributed computing or serverless architectures, are indirect stakeholders, as Flowstate's development builds upon and validates research like Beldi while offering a practical tool that can serve as a teaching or experimental platform. Finally, end-users of serverless applications also benefit indirectly from Flowstate, as its adoption leads to more reliable and consistent services, particularly for use cases requiring multi-step workflows or high fault tolerance. Some examples include developers implementing user registration workflows, payment processing systems, inventory management, multi-step validation processes, and much more.

***Market Research*:**

The serverless computing market presents many opportunities for Flowstate to capture the substantial growth and evolving trends within the market that align well with the product's value proposition mentioned above. Specifically, Flowstate has the ability to penetrate the growing serverless architecture market, which was valued at $12B in 2023 and is projected to grow at a CAGR of 20%+ between 2024 to 2032 (Global Market Insights, 2024). Additionally, Flowstate's ability to work across multiple cloud platforms is especially relevant with the recent trend in multi-cloud serverless solutions. Developers seek to avoid vendor lock-in costs while leveraging different cloud providers' strengths, so Flowstate's cloud-agnostic design has the potential to create value with multi-cloud adoption (Coruzant, 2024).

According to multiple interviews with developers who use serverless computing, three main pain points are current state management challenges, multi-cloud complexity, and individual implementation preferences. Developers currently build their own solutions for state management with serverless functions. The need for durable execution is particularly critical in scenarios involving multiple steps like user registration flows where actions such as database insertions, email confirmations, and CRM updates must execute reliably and atomically. With multi-cloud complexity, managing different APIs across cloud providers poses a significant challenge, yet the benefits echoed briefly before are worth the additional overhead. Developers must handle the logic of creating custom schemas to handle retries and state tracking across different implementations. However, the growing demand for multi-cloud solutions is driven by cost optimization, via spot instances, where organizations can benefit from discounted

computing resources. In terms of individual implementation preferences, the flexibility of relational databases is considered key to adoption – developers show a stronger preference for relational databases such as PostgreSQL over key-value stores. Overall, developers prefer to maintain their existing database interaction patterns rather than adopting new approaches. Based on these interviews, Flowstate's durable execution capabilities could provide value by managing state across the unpredictable terminations of serverless functions. With seamless integration into existing workflows, Flowstate will be a reliable tool for developers to handle complex state management transparently in their existing applications.

***Customer Segment*:**

Flowstate's primary customer segment consists of developers working with serverless functions – individual developers, small teams, and large organizations who can benefit from Flowstate's abstractions that reduce development overhead, mitigate the risk of bugs, and enhance workflow reliability. At least one-third of all cloud developers currently use serverless computing, and this number is steadily increasing. This provides Flowstate with a substantial addressable market with AWS commanding 47.3% and Azure holding 38.9% of the market share (Omdia, 2024). According to the U.S. Bureau of Labor Statistics, the number of app developers is projected to increase by 17%-24% by the end of 2024, and this growth is relevant for Flowstate given startups are increasingly choosing to go fully serverless from the beginning in order to save on costs from normal cloud computing (Straits Research, 2024). These developers work across a multitude of vertices, with particular clusters in IT, finance, and healthcare since scalability and operational efficiency are crucial. Flowstate plans on first targeting developers who use AWS Lambda, starting implementation for DynamoDB, as it has the largest limitations in terms of atomicity scope. However, as progress on Flowstate continues, more functionality will be built out for other databases and serverless platforms.

***Competition*:**

Flowstate's main competitors are DBOS, OpenFaaS, and Temporal. DBOS Cloud was founded by PostgreSQL creator Mike Stonebraker and Apache Spark founder Matei Zaharia. DBOS is a transactional serverless platform, offering automatic logging of application steps, reliable execution capabilities, and unique debugging features. However, DBOS requires developers to use their specific TypeScript or Python frameworks and runs on a proprietary platform that implements OS services in SQL. While they boast impressive performance claims, showing 25x faster execution of durable workflows compared to AWS Lambda and Step Functions, it requires complete platform adoption and lacks multi-cloud flexibility. Flowstate holds a competitive moat over DBOS in that they operate as a flexible middleware solution that integrates with existing infrastructure, unlike DBOS who requires complete platform adoption. Flowstate's annotation-based approach allows developers to maintain their existing codebases and database choices. Additionally, Flowstate's cloud flexibility allows multi-cloud deployments, whereas DBOS operates as a more closed ecosystem.

OpenFaaS operates as an open-source serverless platform built on top of containers, specifically for Docker and Kubernetes integration. The platform has gained decent traction on Github and offers language-agnostic function deployment and efficient auto-scaling. Their strengths lie in their simplicity, broad language support, and flexibility in deployment across various environments. Flowstate differentiates itself with its specialized focus on state management and durability. OpenFaaS limits their scope to general serverless deployment without directly addressing state management challenges across function executions. Flowstate's built-in fault tolerance and state management capabilities, derived from

Beldi, provide a more comprehensive solution for stateful applications. Additionally, Flowstate's approach to handling spot instances and multi-cloud deployments offers more sophisticated state management capabilities than OpenFaaS's general-purpose container orchestration.

Temporal is an incumbent player in the durable execution space, having approximately 2,000 customers. They target large tech companies with extensive DevOps needs, offering features like workflow orchestration, automatic retries, and comprehensive observability tools. While Temporal provides robust lifecycle management and polyglot support across multiple languages, their solution requires significant infrastructure setup and maintenance. Limitations include a 2,000 concurrent operation limit and strict size restrictions. Flowstate differentiates itself by offering a more lightweight, serverless-native approach that eliminates the need for external orchestrators and reduces network latency. Unlike Temporal's complex deployment requirements, Flowstate integrates seamlessly with existing serverless infrastructure while providing similar durability guarantees at a fraction of the operational overhead.

Cost Model:

Flowstate's main cost base includes basic development fees and other R&D costs. To test our middleware, we have been using LocalStack to test AWS Lambda functions and DynamoDB locally and reduce our AWS costs to $0. Right now, we are using the free tier of LocalStack. However, if we need to scale our LocalStack usage to include other locally mocked services, it will cost a minimum of $35 per license per month for a single developer. If our entire team needs licenses, this could cost upwards of $70 per license per month. Other costs include AWS storage costs for tables used in production, which includes our ReadLog, InvokeLog, and LinkedDAAL tables. AWS has a standard storage rate of $0.25 per GB per month. Additionally, calling write operations incurs costs of $0.125 per million writes, and calling read operations costs $0.13 per million reads. However, we plan on presenting a revenue model that passes this variable cost burden onto our customers based on their individual usage.

Revenue Model:

Our team considered multiple options for an effective revenue model. We considered an open source model with the option to pay for additional dev support, however we thought this method wouldn't yield strong retention nor recurring revenue given demand for dev support is not always consistent. Another option we considered was charging a flat license fee. This model is common for middleware software, however there is a lack of opportunity to upsell existing customers and boost revenue growth besides acquiring new customers, which is harder in scaling.

We decided to opt for a revenue model where the users pay for their own development costs that they incur plus an additional spread for our product value. For example, users using AWS lambda will pay the standard AWS costs of $0.125 per million write operations, $0.13 per million read operations, and $0.25 per GB per month for DynamoDB storage. Additionally, we will charge the users for their storage costs incurred for Flowstate specific tables, like our ReadLog table, LinkedDAAL, intent log table, etc. using the standard DynamoDB rate plus a 30% markup. The markup prices in Flowstate's guarantee to provide exactly once semantics when using serverless computing, seamless integration into a user's codebase, and multihosting across various cloud providers. We believe a 30% markup factors in the value proposition of Flowstate, saving developers valuable time with state management and fault tolerance for their apps using serverless.