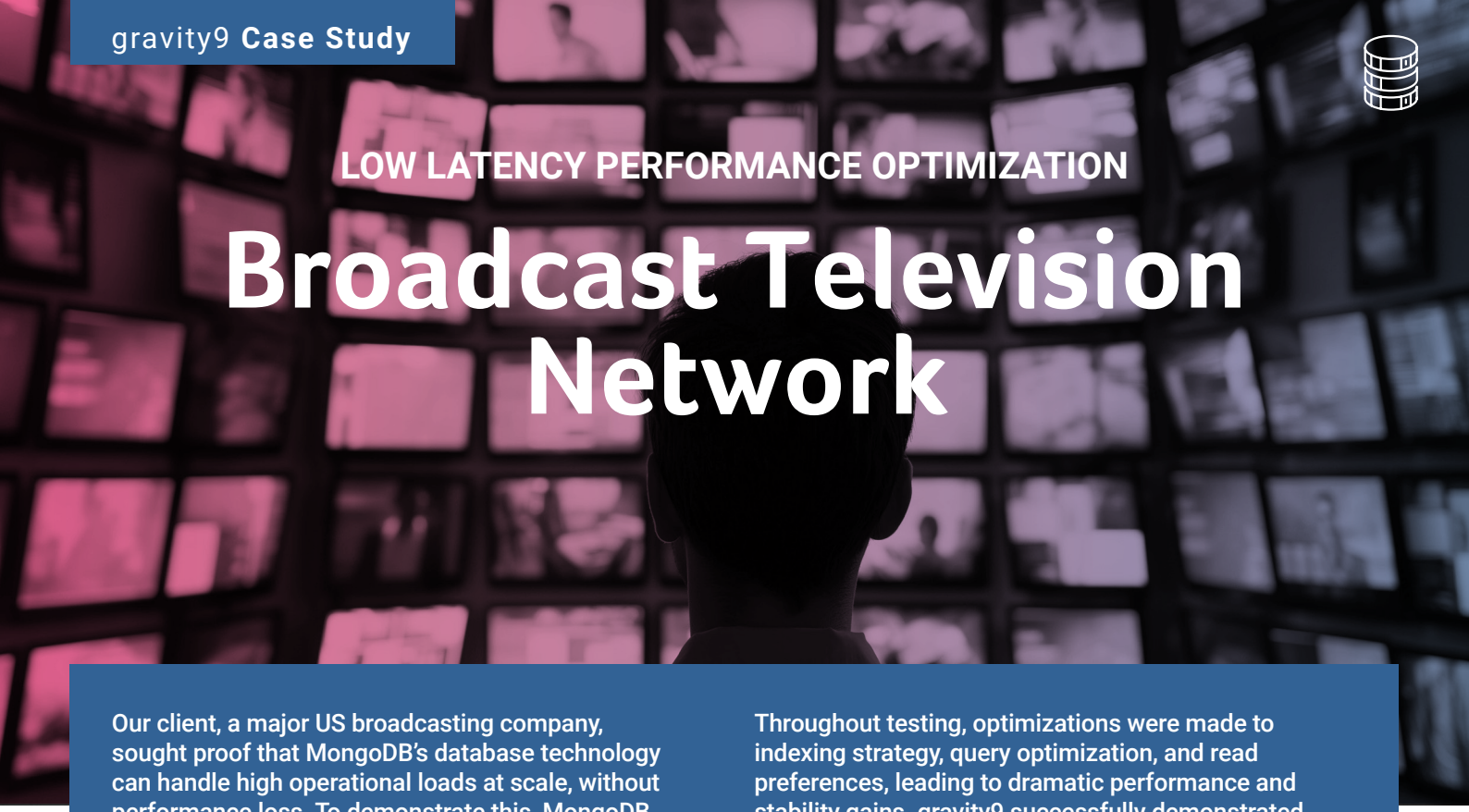


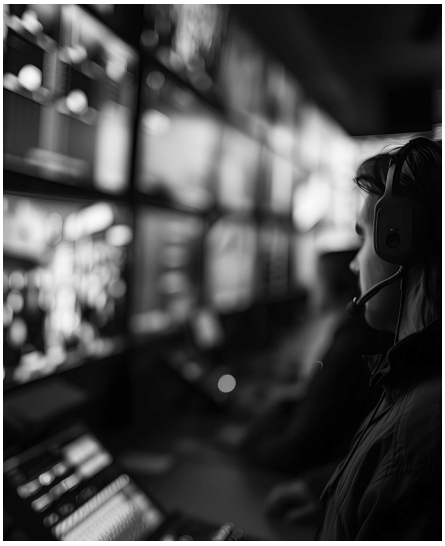
LOW LATENCY PERFORMANCE OPTIMIZATION

Broadcast Television Network

Our client, a major US broadcasting company, sought proof that MongoDB's database technology can handle high operational loads at scale, without performance loss. To demonstrate this, MongoDB approached gravity9 to design and carry out a variety of performance tests in an environment that could simulate several relevant real-world usage scenarios.

gravity9 devised a two-pronged approach, using NodeJS API to mimic database interactions and a Test Harness to conduct load tests on MongoDB Atlas that resemble customer use at escalating demand levels.

Throughout testing, optimizations were made to indexing strategy, query optimization, and read preferences, leading to dramatic performance and stability gains. gravity9 successfully demonstrated that MongoDB provides an excellent database for streaming service requirements for millions of concurrent users.



Utilized Technology Stack

- Cloud: AWS
- Database: MongoDB Atlas
- Backend: NodeJS

Review of Challenges

Our client is a leading United States' television and radio broadcasting organization.

They approached MongoDB for a proof-of-concept solution to demonstrate that MongoDB databases can handle scaling and extreme operational loads without impacting operational or end user performance. MongoDB turned to gravity9 to partner on the project, recognizing our expertise in rapidly developing high-performance applications and MongoDB based systems. The media streaming industry demands robust, scalable, efficient database systems capable of handling vast volumes of data in real-time, and our client needed to stress-test MongoDB with real-world type scenarios to determine whether it could meet challenges encountered in the real world.

Failure to meet these intense demands could lead to performance degradation, which would quickly drive viewers away to a more reliable streaming platform. This outcome would be frustrating for viewers and unacceptable for streaming businesses, resulting in lost revenue and potential impact to reputation.

Our Solution

Initial requirement gathering with our client and MongoDB stakeholders established the scope and prerequisites of the project, formulating a suitable two-pronged approach to demonstrate MongoDB's suitability for the client's needs.

MongoDB Atlas was used as the testing environment, primarily for its cloud capabilities for scalability and reliability. To mirror the client's operational scale, the database was populated with more than 100 million documents between 250GB and 500GB in volume. A range of Atlas clusters between M50 and M140 were leveraged to assess performance across different hardware capabilities. Two primary collections were used to simulate our client's data structure, and steps were taken to optimize indexing and authentication checks.

NodeJS API, which can manage many concurrent connections, was utilized to simulate database interactions. The API leverages the MongoDB Driver for NodeJS. This setup mimics the high-traffic conditions for streaming service operations. AWS Fargate containerized and deployed the NodeJS API, allowing cost-effective scalability without the overhead of managing servers.

The infrastructure included a Load Balancer to distribute incoming traffic evenly, ensuring that no single instance bore too much load, which could skew test results. Load tests were designed to incrementally increase the number of simulated users, pushing MongoDB's performance boundaries. Using a Test Harness service built on AWS Fargate, concurrency was simulated, and MongoDB's response to the escalating load was carefully monitored.

The testing framework, AWS's Distributed Load Testing (DLT) tool, was pivotal. It allowed gravity9 to generate the necessary load using a JMeter script, which directed a mixture of read and write operations to the database through our NodeJS API. This setup was critical for evaluating MongoDB's real-world applicability under high-demand scenarios common in streaming services.

Throughout testing, optimization techniques were employed to enhance MongoDB's performance. One key strategy was covering queries and compound indices, which reduced disk I/O by satisfying query operations entirely through index scans. This approach effectively minimized response times and maintained high throughput under load. Experimentation was also carried out with MongoDB's read preferences, initially directing all read operations to the primary node. As the load increased, the read preference was adjusted to distribute read operations across secondary nodes, balancing the load more effectively. This proved instrumental in maintaining performance as user concurrency reached peak levels.

Our Approach

A two-pronged approach was devised and implemented to demonstrate to the client that MongoDB can handle extreme-load situations. gravity9 implemented a NodeJS API that mimics a client's database interactions and a Test Harness to conduct load tests, simulating customer usage at various levels. These tests challenge MongoDB's operational capacity, scrutinizing its performance under increasingly demanding conditions.

The significance of this exercise lies in its real-world applicability. Streaming services, characterized by their high data throughput and need for quick data retrieval, rely on the underlying database's ability to efficiently handle concurrent interactions. By simulating these interactions across different volume loads and database configurations, gravity9 showcased MongoDB's robustness, testing its ability to maintain performance integrity in the face of substantial demand.

Our client has gained reassurance has gained reassurance that the technical solution can exceed their customers' needs...

"... scalability and ability to handle high-load applications like streaming services."

Subsequent Outcomes

Thanks to gravity9's accurate recreation of real-world usage conditions, via testing, and expertise in MongoDB database configuration and optimization, MongoDB delivered exceptional performance across all metrics. Notably, response times for the 99.9+ percentile of all tests remained well under client operational requirements and consistently under 100ms. This result was a testament to MongoDB's scalability and ability to handle high-load applications like streaming services.

Strategically adjusting the read preferences to include secondary nodes was a pivotal move that contributed significantly to the system's ability to sustain performance under increased load. Combined with our indexing and query optimization strategies gravity9 showcased MongoDB's flexibility and robustness as a data management solution for high-demand environments.

Through a series of load tests, gravity9 demonstrated how MongoDB's performance could be optimized to meet and exceed the stringent requirements of real-time data processing and retrieval. The successful outcome of this project not only solidified MongoDB's standing as a scalable, efficient database solution but also provided valuable insights into the strategies and configurations that can help harness its full potential.

Our client has gained reassurance that the technical solution can exceed their customers' needs during periods of high load and usage uncertainty, resulting in a stable platform on which they can rely.

Meanwhile, MongoDB used the results from this undertaking to demonstrate product suitability for intensive use environments. Additionally, MongoDB has retrieved valuable insight into optimum configurations and optimizations for real-world applications.

Visit our Insights page for more articles about emerging technology trends, the IT industry, interviews, and more!

Visit our Insights page for more articles about emerging technology trends, the IT industry, interviews, and more!

gravity9