Evaluating Amazon DocumentDB Compatibility with MongoDB

Amazon claims that migrating from MongoDB to DocumentDB is “as easy as changing the database endpoint to the new Amazon DocumentDB cluster”.

We regularly assess DocumentDB’s compatibility claims by running 6 MongoDB test suites, totaling 1,481 tests, against DocumentDB’s API emulation. This is the suite that we use to test MongoDB’s own conformity and correctness on every database release, and is the best representation of what the full MongoDB API is.

The tests comprise:

● jsCore: About 1,000 tests of MongoDB CRUD operations and database commands

● aggregation: Over 300 tests of the MongoDB aggregation pipeline

● jsCore_decimal: evaluates correct behavior of applications using the decimal data type for high precision, fractional numeric data typical in financial and scientific workloads

● change_streams: tests MongoDB change streams, used by developers to build event-driven data pipelines that react in real time to database changes

● jsCore_txns: evaluates correct behavior of multi-document ACID transactions in MongoDB

● jsonSchema: tests the data governance controls provided by MongoDB

At the time of testing (January 2022), DocumentDB emulated a subset of the MongoDB 4.0 protocol. Unlike DocumentDB, MongoDB Atlas — the fully managed MongoDB service — supported the latest MongoDB 6.0 release.

Test Results

Over 64% of all of the correctness tests failed when comparing the DocumentDB API emulation to MongoDB. For developers, this means that:

1. Any existing MongoDB apps relying on this functionality would need to be re-engineered if they are to be migrated to DocumentDB
2. Any new apps written against the DocumentDB API only support a subset of MongoDB's functionality
3. Any application written for DocumentDB will be locked-in to AWS.

	Number of Tests	Succeeded	Failed
jsCore	1030	381	649
aggregation	328	97	231
jsCore_Decimal	14	14	0
change_streams	39	8	31
jsCore_txns	49	21	28
jsonSchema*	21	2	19
Totals	1,481	523	958

*Feature not supported by the DocumentDB API emulation

In terms of functionality, DocumentDB most closely resembles MongoDB 4.0, released in 2018. As a result, developers will need to either:

1. Reimplement required database functionality back in the application tier, slowing down the pace of application development
2. Move multiple copies of the data into adjacent AWS technologies, with the associated increases in development and operational costs and platform complexity

Key gaps include:

- Only a subset of the MongoDB Query Language is supported by DocumentDB.
- Fewer than 50% of MongoDB 4.0 aggregation pipeline stages and query language operators are supported. Missing functionality includes:
    - No graph traversals, and no faceting or bucketing of data.
    - Only a limited number of arithmetic, array, and set operators are available.
    - No on-demand materialized views
    - There is also no way to use aggregation expressions with the query language.
    - No support for type conversions in the aggregation pipeline. These allow you to run sophisticated data transformations natively in the database, eliminating many costly, slow, and fragile ETL processes.
    - Aggregation pipeline stages and operators added in recent MongoDB releases, including regex, timestamps, trigonometry, merge, custom aggregation expressions, Union, and more
- No schema governance to control data quality
- Despite claiming support for change streams, over 80% of MongoDB's change streams correctness tests still fail, due to DocumentDB:
    - Not supporting change streams being opened against a non-primary node. Without sharding support, all of the application's writes also have to be serviced by that single DocumentDB primary node, so adding change streams will incur potentially significant contention, impacting application throughput and latency.

    - Not supporting DDL events, including drop, rename, and dropDatabase, preventing developers from responding to collection and database level changes
    - Not supporting $replaceRoot, $replaceWith, $redact, $set or $unset aggregation pipelines, limiting the ability to filter or modify change stream output
    - Resume_after does not conform to the MongoDB API, limiting the ability for users to transition apps between MongoDB and DocumentDB
    - No ability to set fine-grained, per-user permissions for change streams. If change streams are enabled on a database, any and all users with read permissions can see all changes on the database.
    - Change events are only stored for a default of 3 hours before being dropped. MongoDB can deliver change events until the oplog rolls over, with no hard limit.
    - Additional charges will be incurred by change streams for storing and delivering the changes. Change streams in MongoDB Atlas have no additional cost.
- No on-demand materialized views
- Limited cursor options: no support for collation, or tailable cursors
- No causal consistency guarantees, reducing data quality by eliminating the ability to perform monotonic reads across replicas
- DocumentDB does not implement the MongoDB API's tunable consistency options. Even in cases which call for higher throughput and reduced durability guarantees, like streaming IoT sensor data, user tracking, or large-scale social media platforms, clients must wait for all writes to reach a majority of those nodes

DocumentDB's multi-document ACID transactions support, added in their 4.0 update, are limited when compared to transactions in MongoDB, failing over half of the standard test suite:

- DocumentDB transactions deliver only ~65% of the throughput of an equivalent MongoDB Atlas configuration, with close to 20% higher latency.
- DocumentDB transactions can be indeterminate and "ambiguous".
- Transactions that are affected by a timeout or node failure will return an error that cannot tell the user whether the transaction succeeded or failed, potentially violating data integrity and ACID guarantees. To avoid this "ambiguity", Amazon's

documentation recommends users rewrite their code to make all updates idempotent, consuming precious development resources.
- Transactions cannot process more than 32MB of data.
- Because DocumentDB does not support sharding, transactions are scoped to running against collections on a single primary node only, limiting scalability.
- DocumentDB does not support retryable writes or commits, meaning developers must develop complex error handling code themselves.

MongoDB Atlas: Always fully featured

In contrast, Atlas is updated as soon as each new database release is declared as Generally Available, meaning developers don't have to wait months or years to access the latest platform enhancements. Since the release of MongoDB 5.0 in June 2021, Atlas is now updated even more frequently with new GA releases available every quarter, leaving DocumentDB even further behind. MongoDB 5.0 was the first to offer the Stable API, which allows users to pin their application to a specific version of the MongoDB API. This gives them the confidence that their code will continue to run for years, without interruption, even as the database is upgraded beneath it. The Stable API provides you with a level of investment protection and API stability that is simply not possible with most other databases, including DocumentDB.

The service is backed by thousands of support engineers, consultants, and solutions architects from MongoDB and the partner ecosystem who offer the benefits of collective MongoDB knowledge acquired supporting tens of thousands of MongoDB customers over the past decade.

Safe Harbor

The development, release, and timing of any features or functionality described for our products remains at our sole discretion. This information is merely intended to outline our general product direction and it should not be relied on in making a purchasing decision nor is this a commitment, promise or legal obligation to deliver any material, code, or functionality.