Introduction

Good [morning/afternoon everyone. Today I'll be presenting an operational analysis of ScyllaDB, a high-performance NoSQL database system.

In this presentation, I'll cover four key areas of ScyllaDB's operational capabilities:

1. CRUD Operations: Demonstrating the fundamental Create, Read, Update, and Delete operations using ScyllaDB's CQL syntax
2. Query Optimization Techniques: Focusing on partition key selection and efficiency strategies
3. Advanced Features and Transaction Management: Including materialized views, batch operations, and time-to-live functionality

Performance Monitoring and Troubleshooting: Using built-in tools and external monitoring systems

SCYLLA DB

OPERATIONAL ANALYSIS

ScyllaDB Operational Analysis

```
cqlsh> CREATE KEYSPACE tamu_aggies WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 3};

Warnings :
SimpleStrategy replication class is not recommended, but was used for keyspace tamu_aggies. You may suppress this warning by delisting SimpleStrategy from replication_strategy_warn_list confi
this replication strategy on replication_strategy_fail_list.

cqlsh> use tamu_aggies;
cqlsh:tamu_aggies> CREATE TABLE my_class (
              ...   id UUID PRIMARY KEY,
              ...   name TEXT,
              ...   value INT
              ... );
cqlsh:tamu_aggies>
cqlsh:tamu_aggies>
cqlsh:tamu_aggies> CREATE TYPE address (
              ...   street TEXT,
              ...   city TEXT,
              ...   zip_code INT
              ... );
cqlsh:tamu_aggies>
```

Part 1: CRUD Operations

Understanding Keyspaces

In ScyllaDB (and Cassandra), a keyspace is similar to a database schema in relational systems:

- Acts as a top-level namespace defining replication across nodes
- Contains tables, replication strategy, and replication factor

```
CREATE KEYSPACE family
  WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 1};
```

- **Replication Strategy**: Defines how data copies are distributed
- **Replication Factor**: Number of data copies (1 = development, 3+ = production)

Create Operations

```
CREATE KEYSPACE tamu_aggies
  WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 3};
```

```
CREATE TABLE my_class (
  id UUID PRIMARY KEY,
  name TEXT,
  value INT
);

CREATE TYPE address (
  street TEXT,
  city TEXT,
  zip_code INT
);
```

Primary Keys in ScyllaDB

Primary keys:

- Must be unique for each row
- Cannot be null
- Determine data distribution across the cluster
- UUID is ideal as it's globally unique and can be generated client-side

Alter Operations

```
ALTER TABLE my_class ADD email TEXT;

ALTER TABLE my_class WITH compaction = {
  'class': 'SizeTieredCompactionStrategy',
  'max_threshold': 32
};
```

```
cqlsh:tamu_aggies> ALTER TABLE my_class ADD email TEXT;
cqlsh:tamu_aggies> ALTER TABLE my_class WITH compaction = {
               ...   'class': 'SizeTieredCompactionStrategy',
               ...   'max_threshold': 32
               ... };
```

Drop Operations

```
USE tamu_aggies;
DROP TABLE my_class;
DROP KEYSPACE tamu_aggies;
```

```
DROP TYPE address;
```

```
cqlsh> DROP KEYSPACE tamu_aggies;
cqlsh> DROP TYPE address;
InvalidRequest: Error from server: code=2200 [Invalid query] message="No keyspace has been specified
. USE a keyspace, or explicitly specify keyspace.tablename"
cqlsh> use tamu_aggies;
InvalidRequest: Error from server: code=2200 [Invalid query] message="Keyspace 'tamu_aggies' does no
t exist"
```

Secondary Indexes

Create additional access patterns beyond primary key lookups:

```
CREATE INDEX ON my_table(email);
CREATE INDEX my_custom_index ON my_table(age);

-- Using indexes
SELECT * FROM my_table WHERE email = 'user@example.com';
```

```
cqlsh> CREATE KEYSPACE IF NOT EXISTS tamu_aggies
   ... WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 3};

Warnings :
SimpleStrategy replication class is not recommended, but was used for keyspace tamu_aggies. You may suppress this warning by delisting SimpleStrategy from replication_strategy_warn_list con
figuration option, or make it into an error by listing this replication strategy on replication_strategy_fail_list.

cqlsh> use tamu_aggies;
cqlsh:tamu_aggies> CREATE TABLE course_enrollment (
              ...     course_id uuid PRIMARY KEY,
              ...     course_name text,
              ...     department text,
              ...     max_capacity int,
              ...     current_enrollment int
              ... );
cqlsh:tamu_aggies> INSERT INTO course_enrollment (course_id, course_name, department, max_capacity, current_enrollment)
              ... VALUES (uuid(), 'Advanced Databases', 'ISTM', 70, 55);
cqlsh:tamu_aggies> INSERT INTO course_enrollment (course_id, course_name, department, max_capacity, current_enrollment)
              ... VALUES (uuid(), 'Data Mining', 'ISTM', 70, 55);
cqlsh:tamu_aggies> SELECT * FROM course_enrollment;

 course_id                            | course_name        | current_enrollment | department | max_capacity
--------------------------------------+--------------------+--------------------+------------+--------------
 f3204eb3-7f18-41fa-9c86-2d13976d5804 |        Data Mining |                 55 |       ISTM |           70
 32eee364-09cf-4c30-9a44-b8314a255ba8 | Advanced Databases |                 55 |       ISTM |           70
```

Part 2: Query Optimization

Partition Key Optimization

- Choose partition keys that evenly distribute data
- Well-designed keys improve read/write performance

- Example: UUID as partition key ensures even distribution

Lightweight Transactions

```
UPDATE course_enrollment
SET enrollment_count = enrollment_count - 1
WHERE course_id = [specific-uuid]
IF enrollment_count > 0;
```

```
cqlsh:tamu_aggies> UPDATE course_enrollment
                ... SET current_enrollment = 56
                ... WHERE course_id = 32eee364-09cf-4c30-9a44-b8314a255ba8
                ... IF current_enrollment = 55;

 [applied] | current_enrollment
-----------+--------------------
      True |                 55
```

- Uses IF clause for compare-and-set operations
- Maintains data integrity in distributed environment
- Performance impact: milliseconds with partition key vs. seconds without

Part 3: Transaction Management

Batch Operations

```
cqlsh:tamu_aggies> BEGIN BATCH
                ...   -- Update enrollment in Advanced Databases - using direct assignment instead
                ...   UPDATE course_enrollment
                ...   SET current_enrollment = 54  -- Assuming current value is 55, set it to 54
                ...   WHERE course_id = 32eee364-09cf-4c30-9a44-b8314a255ba8;
                ...
                ...   -- Insert a related course with your specified values
                ...   INSERT INTO course_enrollment
                ...   (course_id, course_name, department, max_capacity, current_enrollment)
                ...   VALUES (uuid(), 'Advanced Database Topics', 'ISTM', 70, 50);
                ... APPLY BATCH;
```

Example batch demonstrating atomic operations:

```
BEGIN BATCH
  UPDATE courses SET enrollment_count = enrollment_count - 1
  WHERE course_id = [Advanced-DB-UUID];

  INSERT INTO courses (course_id, name, max_capacity, enrollment_count)
  VALUES (uuid(), 'New Related Course', 70, 50);
APPLY BATCH;
```

- Operations either complete fully or fail entirely
- Maintains data integrity across multiple operations

Part 4: Performance Monitoring

Built-in Tracing Tools

```
TRACING ON;
SELECT * FROM course_enrollment WHERE course_id = [specific-uuid];
```

```
cqlsh:tamu_aggies> TRACING ON;
Now Tracing is enabled
cqlsh:tamu_aggies> SELECT * FROM course_enrollment WHERE course_id = 32eee364-09cf-4c30-9a44-b8314a255ba8;

 course_id                            | course_name        | current_enrollment | department | max_capacity
--------------------------------------+--------------------+--------------------+------------+--------------
 32eee364-09cf-4c30-9a44-b8314a255ba8 | Advanced Databases |                 54 |       ISTM |           70

(1 rows)

Tracing session: 868f8590-0c5d-11f0-9442-5ea1199cef4f

 activity                                                                                                                                                  | timestamp
   | source      | source_elapsed | client
-----------------------------------------------------------------------------------------------------------------------------------------------------------+---------------------------
---+-------------+----------------+------------
                                                                                                                                        Execute CQL3 query | 2025-03-29 05:20:27.7530
00 | 172.17.0.2 |              0 | 172.17.0.2
                                                                                                                        Parsing a statement [shard 15] | 2025-03-29 05:20:27.7534
09 | 172.17.0.2 |              1 | 172.17.0.2
                                                                                  Processing a statement for authenticated user: anonymous [shard 15] | 2025-03-29 05:20:27.7534
88 | 172.17.0.2 |             80 | 172.17.0.2
                                                                                                     Executing read query (reversed false) [shard 15] | 2025-03-29 05:20:27.7535
86 | 172.17.0.2 |            177 | 172.17.0.2
                               Creating read executor for token -764619882566839294 with all: [172.17.0.2] targets: [172.17.0.2] repair decision: NONE [shard 15] | 2025-03-29 05:20:27.7536
94 | 172.17.0.2 |            286 | 172.17.0.2
                              Creating never_speculating_read_executor - speculative retry is disabled or there are no extra replicas to speculate with [shard 15] | 2025-03-29 05:20:27.7536
95 | 172.17.0.2 |            287 | 172.17.0.2
                                                                                                            read_data: querying locally [shard 15] | 2025-03-29 05:20:27.7536
99 | 172.17.0.2 |            291 | 172.17.0.2
                                                    Start querying singular range {{-764619882566839294, pk{001032eee36409cf4c309a44b8314a255ba8}}} [shard 3] | 2025-03-29 05:20:27.7537
91 | 172.17.0.2 |              8 | 172.17.0.2
                                                                                 [reader concurrency semaphore user] admitted immediately [shard 3] | 2025-03-29 05:20:27.7537
96 | 172.17.0.2 |             12 | 172.17.0.2
                                                                                      [reader concurrency semaphore user] executing read [shard 3] | 2025-03-29 05:20:27.7537
99 | 172.17.0.2 |             15 | 172.17.0.2
                                   Querying cache for range {{-764619882566839294, pk{001032eee36409cf4c309a44b8314a255ba8}}} and slice [(-inf, +inf)] [shard 3] | 2025-03-29 05:20:27.7544
45 | 172.17.0.2 |            661 | 172.17.0.2
 Page stats: 1 partition(s), 0 static row(s) (0 live, 0 dead), 1 clustering row(s) (1 live, 0 dead), 0 range tombstone(s) and 4 cell(s) (4 live, 0 dead) [shard 3] | 2025-03-29 05:20:27.7544
75 | 172.17.0.2 |            692 | 172.17.0.2
                                                                                                                      Querying is done [shard 3] | 2025-03-29 05:20:27.7544
86 | 172.17.0.2 |            703 | 172.17.0.2
                                                                                                Done processing - preparing a result [shard 15] | 2025-03-29 05:20:27.7571
18 | 172.17.0.2 |           3710 | 172.17.0.2
                                                                                                                                Request complete | 2025-03-29 05:20:27.7567
27 | 172.17.0.2 |           3727 | 172.17.0.2
```

Tracing reveals:

- Total query execution time
- Internal steps and timings
- Node involvement
- Potential bottlenecks

SELECT * FROM SYSTEM_SIZE_ESTIMATES

```
cqlsh:tamu_aggies> SELECT * FROM system.size_estimates
               ... WHERE keyspace_name = 'tamu_aggies';

 keyspace_name | table_name         | range_start           | range_end             | mean_partition_size | partitions_count
```

```
cqlsh:tamu_aggies> SELECT * FROM system_traces.sessions;

 session_id                           | client      | command | coordinator | duration | parameters
                                                                                                                 | request             | request_size | response_size | started_at
                      | username
-------------------------------------+-------------+---------+-------------+----------+-----------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------+---------------------+--------------+---------------+------------
----------------------+-----------
 868f8590-0c5d-11f0-9442-5ea1199cef4f | 172.17.0.2 |   QUERY |  172.17.0.2 |     3727 | {'consistency_level': 'ONE', 'page_size': '100', 'query': 'SELECT * FROM course_enrollment WHERE cour
se_id = 32eee364-09cf-4c30-9a44-b8314a255ba8;', 'serial_consistency_level': 'SERIAL', 'user_timestamp': '1743225627752925'} | Execute CQL3 query |          108 |           338 | 2025-03-29
05:20:27.753000+0000 | anonymous

(1 rows)
cqlsh:tamu_aggies> SELECT * FROM system_traces.events;

 session_id                           | event_id                             | activity
                                          | scylla_parent_id | scylla_span_id  | source      | source_elapsed | thread
-------------------------------------+--------------------------------------+----------------------------------------------------------------------------------------------------------------------------
-----------------------------------------+------------------+-----------------+-------------+----------------+-----------
 868f8590-0c5d-11f0-9442-5ea1199cef4f | 868f958b-0c5d-11f0-9442-5ea1199cef4f |
                     Parsing a statement |                0 |   4415110950987 | 172.17.0.2 |              1 | shard 15
 868f8590-0c5d-11f0-9442-5ea1199cef4f | 868f989f-0c5d-11f0-9442-5ea1199cef4f |                                                                                                      Processing a st
atement for authenticated user: anonymous |                0 |   4415110950987 | 172.17.0.2 |             80 | shard 15
 868f8590-0c5d-11f0-9442-5ea1199cef4f | 868f9c70-0c5d-11f0-9442-5ea1199cef4f |
    Executing read query (reversed false) |                0 |   4415110950987 | 172.17.0.2 |            177 | shard 15
 868f8590-0c5d-11f0-9442-5ea1199cef4f | 868fa0ae-0c5d-11f0-9442-5ea1199cef4f |                                  Creating read executor for token -764619882566839294 with all: [172.17.0.2] ta
rgets: [172.17.0.2] repair decision: NONE |                0 |   4415110950987 | 172.17.0.2 |            286 | shard 15
 868f8590-0c5d-11f0-9442-5ea1199cef4f | 868fa0bd-0c5d-11f0-9442-5ea1199cef4f |                                 Creating never_speculating_read_executor - speculative retry is disabled or ther
e are no extra replicas to speculate with |                0 |   4415110950987 | 172.17.0.2 |            287 | shard 15
 868f8590-0c5d-11f0-9442-5ea1199cef4f | 868fa0dc-0c5d-11f0-9442-5ea1199cef4f |
               read_data: querying locally |                0 |   4415110950987 | 172.17.0.2 |            291 | shard 15
 868f8590-0c5d-11f0-9442-5ea1199cef4f | 868fa477-0c5d-11f0-a357-5ea6199cef4f |                                             Start querying singular range {{-764619882566839294, p
k{001032eee36409cf4c309a44b8314a255ba8}}} |    4415110950987 | 124029003917579 | 172.17.0.2 |              8 |  shard 3
 868f8590-0c5d-11f0-9442-5ea1199cef4f | 868fa4a5-0c5d-11f0-a357-5ea6199cef4f |                                                                                                      [reader concurr
ency semaphore user] admitted immediately |    4415110950987 | 124029003917579 | 172.17.0.2 |             12 |  shard 3
 868f8590-0c5d-11f0-9442-5ea1199cef4f | 868fa4c4-0c5d-11f0-a357-5ea6199cef4f |                                                                                                         [reader c
oncurrency semaphore user] executing read |    4415110950987 | 124029003917579 | 172.17.0.2 |             15 |  shard 3
 868f8590-0c5d-11f0-9442-5ea1199cef4f | 868fbe02-0c5d-11f0-a357-5ea6199cef4f |                                       Querying cache for range {{-764619882566839294, pk{001032eee36409cf4c309a4
4b8314a255ba8}}} and slice [(-inf, +inf)] |    4415110950987 | 124029003917579 | 172.17.0.2 |            661 |  shard 3
 868f8590-0c5d-11f0-9442-5ea1199cef4f | 868fbf2f-0c5d-11f0-a357-5ea6199cef4f | Page stats: 1 partition(s), 0 static row(s) (0 live, 0 dead), 1 clustering row(s) (1 live, 0 dead), 0 range to
mbstone(s) and 4 cell(s) (4 live, 0 dead) |    4415110950987 | 124029003917579 | 172.17.0.2 |            692 |  shard 3
 868f8590-0c5d-11f0-9442-5ea1199cef4f | 868fbf9c-0c5d-11f0-a357-5ea6199cef4f |
                        Querying is done |    4415110950987 | 124029003917579 | 172.17.0.2 |            703 |  shard 3
 868f8590-0c5d-11f0-9442-5ea1199cef4f | 8690266a-0c5d-11f0-9442-5ea1199cef4f |
     Done processing - preparing a result |                0 |   4415110950987 | 172.17.0.2 |           3710 | shard 15

(13 rows)
cqlsh:tamu_aggies> TRACING OFF;
Disabled Tracing.
```

```
   tamu_aggies | course_enrollment | -3649185051365608035 | -3648679467481425086 |                   0 |                0
   tamu_aggies | course_enrollment |  -371867932353059173 |  -281048287044735901 |                   0 |                0
   tamu_aggies | course_enrollment | -3783549675032091211 | -3649185051365608035 |                   0 |                0
   tamu_aggies | course_enrollment | -3982219112397168481 | -3783549675032091211 |                   0 |                0
   tamu_aggies | course_enrollment | -4006056807664523468 | -3982219112397168481 |                   0 |                0
   tamu_aggies | course_enrollment | -4039919941292295443 | -4006056807664523468 |                   0 |                0
   tamu_aggies | course_enrollment | -4143264996811021494 | -4039919941292295443 |                   0 |                0
   tamu_aggies | course_enrollment | -4200676460757592165 | -4143264996811021494 |                   0 |                0
   tamu_aggies | course_enrollment | -4385118449920616163 | -4200676460757592165 |                   0 |                0
   tamu_aggies | course_enrollment | -4670969658236431222 | -4385118449920616163 |                   0 |                0
```

SELECT * FROM SYSTEM_TRACE_SESSIONS

External Monitoring

I was unable to perform the node tools or any type of monitoring because of run time errors and advising me that I did not have enough memory resources in the version of Scylla DB to perform the operation.

- Use `nodetool status` for cluster health
- Monitor for:
    - High latencies in trace results
    - Large partition sizes causing distribution problems
    - Resource constraints (CPU, memory, disk I/O)

Conclusion

ScyllaDB represents a powerful evolution in NoSQL database technology:

- Combines CQL syntax accessibility with exceptional distributed performance
- Balances performance with data integrity through proper query optimization
- Provides comprehensive monitoring for visibility into system performance

These features make ScyllaDB ideal for organizations requiring:

- High-throughput, low-latency operations at scale
- Performance, reliability, and observability for modern enterprise applications

Thank you for your time.

References

"Cqlsh: The CQL Shell." *ScyllaDB Open Source Documentation*, ScyllaDB, https://opensource.docs.scylladb.com/stable/cql/cqlsh.html. Accessed 30 Mar. 2025.

"ScyllaDB - Part 1 - An Introduction." *YouTube*, uploaded by Programming with Mosh, 17 Oct. 2023, https://youtu.be/MWFWM_LcouY?si=TqnqvmMRM2m0gGmu.

"ScyllaDB - Part 2 - Data Modeling Basics." *YouTube*, uploaded by Programming with Mosh, 18 Oct. 2023, https://youtu.be/D3j5rSUSaCE?si=UtkYH27irhuZkA-o.

"ScyllaDB - Part 3 - Collections, Counters and Frozen Types." *YouTube*, uploaded by Programming with Mosh, 19 Oct. 2023, https://youtu.be/JPkrdWMVpPk?si=xNnx2n8K-BwR4aJC.

"ScyllaDB - Part 4 - Indexing and Materialized Views." *YouTube*, uploaded by Programming with Mosh, 20 Oct. 2023, https://youtu.be/cOlyquVuFJU?si=s0oidMDDiJZTnIJ3.

"ScyllaDB - Part 5 - Lightweight Transactions." *YouTube*, uploaded by Programming with Mosh, 23 Oct. 2023, https://youtu.be/A5TkqoMxoJk?si=3zh75V_kUNEsCysI.