DC Lab 8

Aim: To understand the concepts of distributed consistency management in distributed systems and to implement and observe different consistency models.

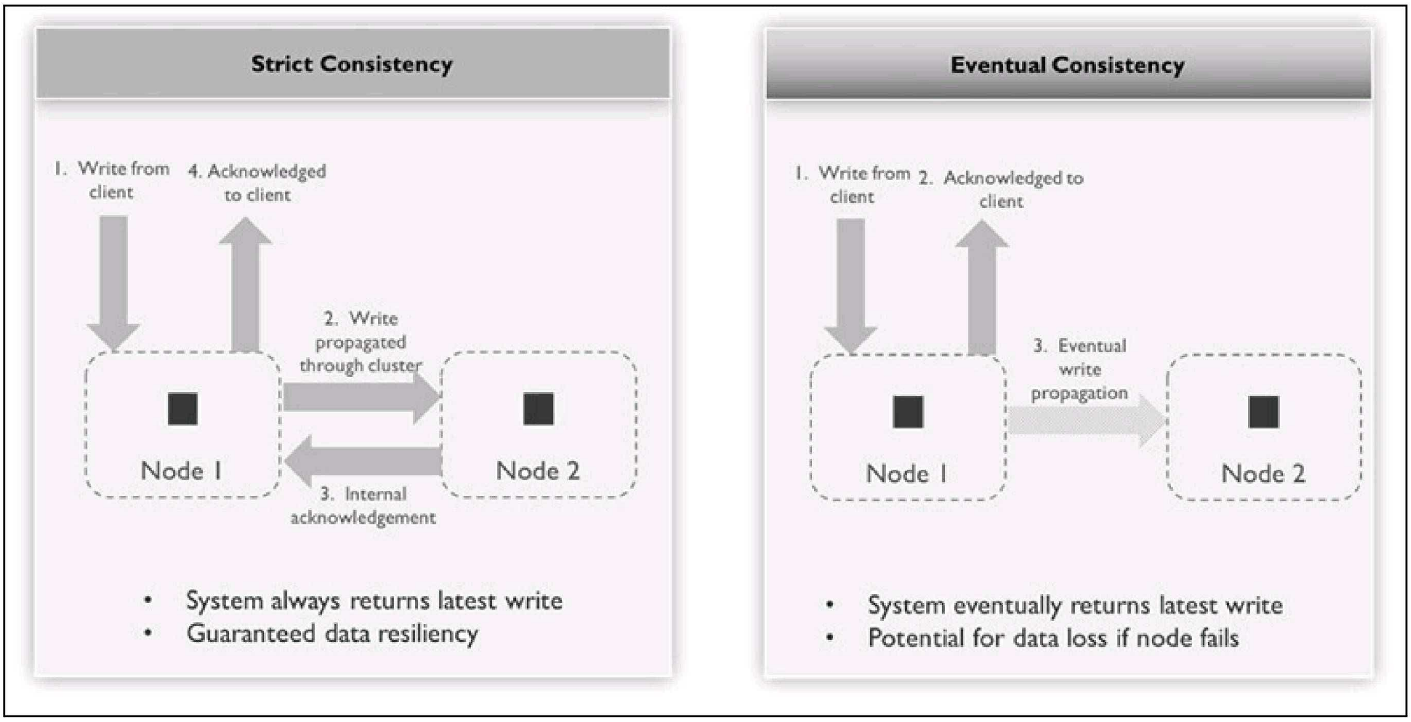
Theory:

Consistency: Refers to the agreement between multiple copies of data in a distributed system. Consistency ensures that all nodes in the system have the same view of data at any given time.

Distributed Systems: Systems composed of multiple interconnected nodes that communicate and coordinate to achieve a common goal. These nodes can be geographically distributed and may fail independently.

Consistency Models: Defines the level of consistency guaranteed by a distributed system. Common models include:

- **Strong Consistency:** All nodes see the same data at the same time. Any read operation returns the latest write.
- **Eventual Consistency:** All nodes eventually converge to the same state, though intermediate states may vary. Guarantees are relaxed, allowing temporary inconsistencies.
- **Causal Consistency:** Preserves causality; if event A causally precedes event B, all nodes will observe event A before event B.



CAP Theorem: States that in a distributed system, it's impossible to simultaneously achieve Consistency, Availability, and Partition tolerance. Distributed systems must sacrifice one of these aspects in the event of a network partition.

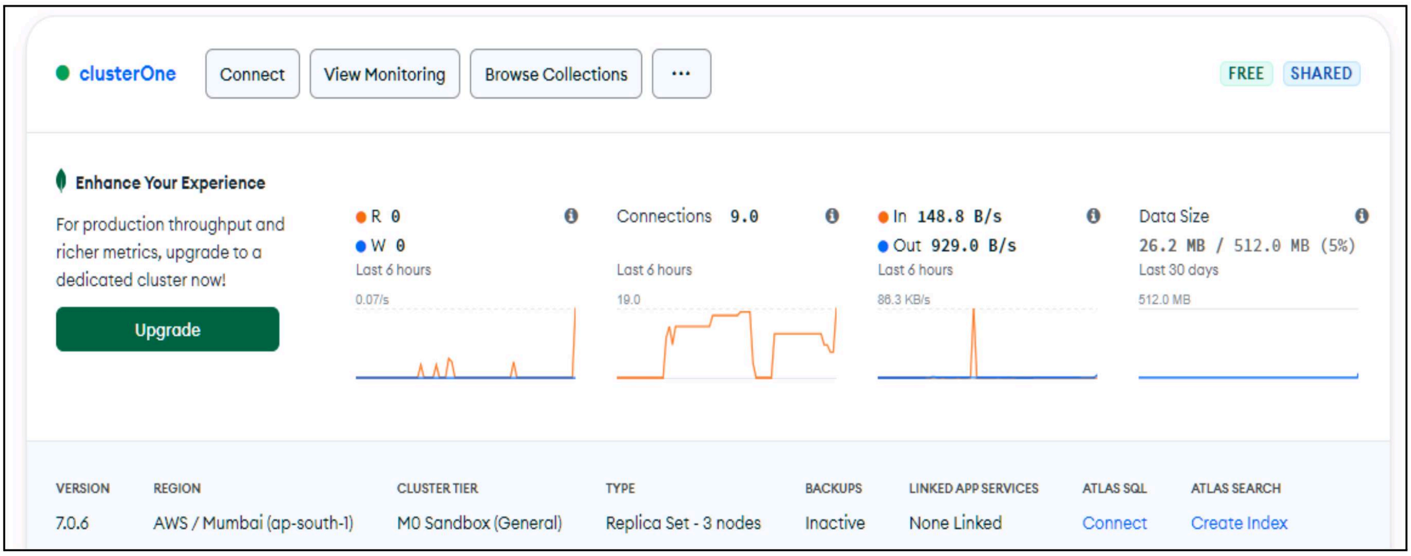
- **Consistency-Partition Tolerance (CP):** Sacrifices availability to maintain consistency.
- **Availability-Partition Tolerance (AP):** Sacrifices consistency to maintain availability.
- **Consistency-Availability (CA):** Sacrifices partition tolerance.

Replication: The process of maintaining copies of data across multiple nodes in a distributed system to improve fault tolerance and availability.

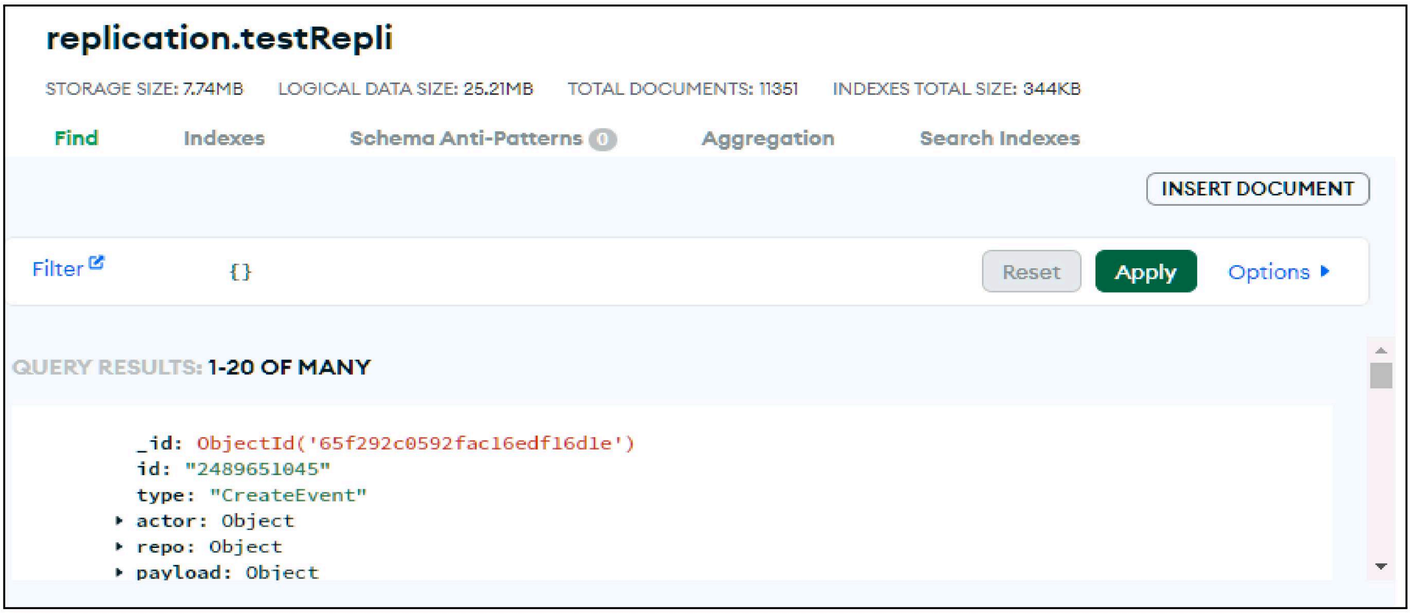
Implementation:

Strong consistency:

3 Nodes on MongoDB (1 primary and 2 replica set):



Adding JSON data into replication Database:



Checking the replication status of clusters on mongoDB:

```
>_MONGOSH

> rs.status()
```

Primary Node: (checking the 'optimeDate' property)

```
{
  _id: 1,
  name: 'ac-n9iwuiz-shard-00-01.awyx4aq.mongodb.net:27017',
  health: 1,
  state: 1,
  stateStr: 'PRIMARY',
  uptime: 130563,
  optime: [Object],
  optimeDate: 2024-03-14T09:41:58.000Z,
```

Secondary Node 1: (checking the 'optimeDate' property - 2 second lag wrt primary node)

```
{
  _id: 0,
  name: 'ac-n9iwuiz-shard-00-00.awyx4aq.mongodb.net:27017',
  health: 1,
  state: 2,
  stateStr: 'SECONDARY',
  uptime: 130077,
  optime: [Object],
  optimeDurable: [Object],
  optimeDate: 2024-03-14T09:41:56.000Z,
```

Secondary Node 2: (checking the 'optimeDate' property - 2 second lag wrt primary node)

```
{
  _id: 2,
  name: 'ac-n9iwuiz-shard-00-02.awyx4aq.mongodb.net:27017',
  health: 1,
  state: 2,
  stateStr: 'SECONDARY',
  uptime: 129615,
  optime: [Object],
  optimeDurable: [Object],
  optimeDate: 2024-03-14T09:41:56.000Z,
```

Eventual Consistency:

Setting up 3 nodes on Cassandra:

```
C:\Users\Administrator>docker run --name cassandra-1 -p 9042:9042 -d cassandra:3.7
Unable to find image 'cassandra:3.7' locally
3.7: Pulling from library/cassandra
6a5a5368e0c2: Downloading [=========================================>         ]  42.8MB/51.35MB
97e4c6575710: Download complete
d8288e3be5a2: Download complete
d111f542073e: Download complete
2549cfb76ce6: Download complete
a375ee20c601: Download complete
2e678e60bfc4: Downloading [=================>                                 ]  37.01MB/108.7MB
c2d5e7ed7dfc: Download complete
21015df69ccb: Download complete
a5b3a5d43f72: Download complete
```

```
PS C:\Users\Administrator> $INSTANCE1=$(docker inspect --format="{{ .NetworkSettings
.IPAddress }}" cassandra-1)
PS C:\Users\Administrator> echo "Instance 1: ${INSTANCE1}"
Instance 1: 172.17.0.2
PS C:\Users\Administrator> docker run --name cassandra-2 -p 9043:9042 -d -e CASSANDR
A_SEEDS=$INSTANCE1 cassandra:3.7
36fdbc8f6ed9d87f9d69db4a851ea4acbdfff08d51169a6587fe235facdf1708
PS C:\Users\Administrator> $INSTANCE2=$(docker inspect --format="{{ .NetworkSettings
.IPAddress }}" cassandra-2)
PS C:\Users\Administrator> echo "Instance 2: ${INSTANCE2}"
Instance 2: 172.17.0.3
```

```
PS C:\Users\Administrator> docker run --name cassandra-3 -p 9044:9042 -d -e CASSANDR
A_SEEDS=$INSTANCE1,$INSTANCE2 cassandra:3.7
2a542d01d6bfd2dccef672ef8c88db1cb03a5ba968a0bb4bd0556319e06e562f
PS C:\Users\Administrator> $INSTANCE3=$(docker inspect --format="{{ .NetworkSettings
.IPAddress }}" cassandra-3)
PS C:\Users\Administrator> echo "Instance 3: ${INSTANCE3}"
Instance 3: 172.17.0.4
```

Docker containers:

Name	Image	Status	Port(s)	Last started	Actions
cassandra-1 40a4c2063be8	cassandra:	Running	9042:9042	48 minutes ago	
cassandra-2 56520f3d1901	cassandra:	Running	9043:9042	48 minutes ago	
cassandra-3 1fafb25cb1ff	cassandra:	Running	9044:9042	47 minutes ago	

Showing 3 items

Checking status:

```
PS C:\Users\Administrator> docker exec cassandra-3 nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load       Tokens       Owns (effective)  Host ID
            Rack
UN  172.17.0.3  102.54 KiB  256          70.7%             307fe375-4824-4825-8e80-a
e9b36993eba  rack1
UN  172.17.0.2  107.95 KiB  256          63.4%             2f8e0ab8-e34b-43d2-a491-5
0bc8c8f976a  rack1
UN  172.17.0.4  83.72 KiB  256          65.9%             1017a7d9-9059-44b3-9d59-7b
f0115da490  rack1
```

```
PS C:\Users\Administrator> docker exec -it cassandra-1 cqlsh
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.7 | CQL spec 3.4.2 | Native protocol v4]
Use HELP for help.
```

Creating keyspace and table in Cassandra Node 1:

```
cqlsh> DESCRIBE keyspaces;

system_traces  system_schema  system_auth  system  system_distributed

cqlsh> CREATE KEYSPACE learn_cassandra
   ...   WITH REPLICATION = {
   ...    'class' : 'NetworkTopologyStrategy',
   ...    'datacenter1' : 3
   ...   };
```

```
cqlsh> CREATE TABLE learn_cassandra.users_by_country (
   ...     country text,
   ...     user_email text,
   ...     first_name text,
   ...     last_name text,
   ...     age smallint,
   ...     PRIMARY KEY ((country), user_email)
   ... );
```

Adding 578 rows in the table in Cassandra Node 1:

```
cqlsh> SELECT * FROM learn_cassandra.users_by_country
   ... ;

 country | user_email | age | first_name | last_name
---------+------------+-----+------------+-----------

(0 rows)
cqlsh>
cqlsh> INSERT INTO learn_cassandra.users_by_country (country,user_email,first_name,last_name,age) VALUES('US', 'michael@email.
com', 'Michael','Jordan',58);
cqlsh> INSERT INTO learn_cassandra.users_by_country (country,user_email,first_name,last_name,age) VALUES('US', 'sarah@email.co
m', 'Sarah','Connor',35);
cqlsh> INSERT INTO learn_cassandra.users_by_country (country,user_email,first_name,last_name,age) VALUES('US', 'emily@email.co
m', 'Emily','Smith',42);
att2@emacqlsh> INSERT INTO learn_cassandra.users_by_country (country,user_email,first_name,last_name,age) VALUES('UK', 'james@
email.com', 'James','Bond',45);
cqlsh> INSERT INTO learn_cassandra.users_by_country (country,user_email,first_name,last_name,age) VALUES('UK', 'emma@email.com
', 'Emma','Watson',31);
cqlsh> INSERT INTO learn_cassandra.users_by_country (country,user_email,first_name,last_name,age) VALUES('UK', 'daniel@email.c
om', 'Daniel','Craig',53);
rcqlsh> INSERT INTO learn_cassandra.users_by_country (country,user_email,first_name,last_name,age) VALUES('UK', 'olivia@email.
com', 'Olivia','Jones',29);
cqlsh> INSERT INTO learn_cassandra.users_by_country (country,user_email,first_name,last_name,age) VALUES('US', 'william@email.
com', 'William','Shakespeare',456);
```

Checking consistency in Cassandra Node 3 and finding inconsistent data:

```
cqlsh> CONSISTENCY ALL
Consistency level set to ALL.
```

```
cqlsh> SELECT * FROM learn_cassandra.users_by_country;
```

```
 country     | user_email                        | age | first_name  | last_name
-------------+-----------------------------------+-----+-------------+---------------
    Zimbabwe |      sbroader7e@macromedia.com    |  46 |        Sela |        Broader
     Belarus |          bpudney9i@utexas.edu     |  13 |        Beck |         Pudney
     Belarus | gmcelory2e@huffingtonpost.com     |  98 |      Garvin |       Mc Elory
     Belarus |             rseel74@cam.ac.uk     |  57 |      Rudolf |           Seel
     Belarus |                  wbonyb2@g.co     |  80 |      Waldon |           Bony
   Argentina |             abourke9e@loc.gov     |   4 |      Aleece |         Bourke
   Argentina |             adiemer9m@163.com     |  55 | Archaimbaud |         Diemer
   Argentina |        bmconie2j@aboutads.info    |   4 |   Benedikta |         McOnie
   Argentina |         cdrewet7g@yahoo.co.jp     |  21 |       Clark |         Drewet
   Argentina |         cpearnec6@examiner.com    |  30 |        Cull |         Pearne
   Argentina |         htracey4r@oaic.gov.au     |  74 |      Howard |         Tracey
   Argentina |              ibehnal@dot.gov      |  64 |       Irina |           Behn
   Argentina |        jhollymanct@freewebs.com   |   3 |      Jdavie |       Hollyman
   Argentina |       jwaythingcu@auda.org.au     |  41 |       Jenna |       Waything
   Argentina |      mmctaggart7i@so-net.ne.jp    |  44 |     Marshal |      McTaggart
   Argentina |      rgildersleaves7y@de.vu       |  36 |     Roderic |  Gildersleaves
     Nigeria |            hcarabetd7@nih.gov     |  97 |       Hilde |        Carabet
    Ethiopia |            arosaac@wisc.edu       |  65 |   Angelique |           Rosa
   Australia |    jcarslake1k@arstechnica.com    |  50 |       Jaime |       Carslake
      Panama |         msimmankbm@booking.com    |  97 |      Myrtie |        Simmank
       Italy |           gcosin3f@joomla.org     |  95 |     Godfree |          Cosin
       Italy |      rpatching3h@bloomberg.com    |  70 |      Rowena |       Patching
     Hungary |     twoodroofaf@wikispaces.com    |  18 |       Tilly |       Woodroof
  East Timor |         vscourgie5s@hexun.com     |  90 |        Vere |       Scourgie
    Cambodia |          khobbing2x@amazon.de     |  26 |       Kerry |        Hobbing

(325 rows)
```

Demonstrating eventual consistency in Cassandra Node 3

Belarus	bpudney9i@utexas.edu	13	Beck	Pudney
Belarus	gmcelory2e@huffingtonpost.com	98	Garvin	Mc Elory
Belarus	rseel74@cam.ac.uk	57	Rudolf	Seel
Belarus	wbonyb2@g.co	80	Waldon	Bony
Argentina	abourke9e@loc.gov	4	Aleece	Bourke
Argentina	adiemer9m@163.com	55	Archaimbaud	Diemer
Argentina	bmconie2j@aboutads.info	4	Benedikta	McOnie
Argentina	cdrewet7g@yahoo.co.jp	21	Clark	Drewet
Argentina	cpearnec6@examiner.com	30	Cull	Pearne
Argentina	htracey4r@oaic.gov.au	74	Howard	Tracey
Argentina	ibehnal@dot.gov	64	Irina	Behn
Argentina	jhollymanct@freewebs.com	3	Jdavie	Hollyman
Argentina	jwaythingcu@auda.org.au	41	Jenna	Waything
Argentina	mmctaggart7i@so-net.ne.jp	44	Marshal	McTaggart
Argentina	rgildersleaves7y@de.vu	36	Roderic	Gildersleaves
Nigeria	hcarabetd7@nih.gov	97	Hilde	Carabet
Ethiopia	arosaac@wisc.edu	65	Angelique	Rosa
Australia	jcarslake1k@arstechnica.com	50	Jaime	Carslake
Panama	msimmankbm@booking.com	97	Myrtie	Simmank
Italy	gcosin3f@joomla.org	95	Godfree	Cosin
Italy	rpatching3h@bloomberg.com	70	Rowena	Patching
Hungary	twoodroofaf@wikispaces.com	18	Tilly	Woodroof
East Timor	vscourgie5s@hexun.com	90	Vere	Scourgie
Cambodia	khobbing2x@amazon.de	26	Kerry	Hobbing

(578 rows)

Conclusion:

In conclusion, this experiment helped us learn about managing consistency in distributed systems by trying out different methods. By seeing how each method worked, we understood the advantages and disadvantages, which is important for making distributed systems that work well.