The Mystery of Time in Distributed Systems

Part - III

Cassandra don't need vector clocks

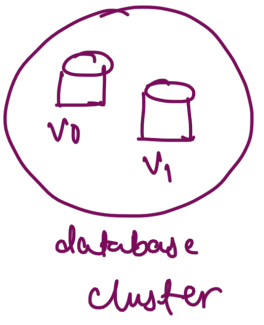
used for conflict resolution

How???

reference - datastax blog

Key = mandeep

values v_0 = {'email': 'msdeep14.ms@gmail.com'
'phone': 'nhi bataunga'
}



↓ update email address

v_1 = { 'email': 'msdeep14.ms@msdeepSingh.com',
'phone': 'nhi bataunga'
}

↓ update phone no.

read from replica and update it.

V_2 = { 'email': 'msdeep14.ms@gmail.com'
'phone': '7982sn3xyy'}

In general, last write wins algorithm picks most recent

write → V1 is discarded and V2 is picked up.

data loss

For vector clock, both V1 and V2 are retained.

↓

return both versions to client on next read and client can merge as appropriate.

Issues with vector clocks

① Performance → updating a field requires
↳ read & deserialize existing object
↳ update field
↳ serialize and write back

② Siblings → multiple versions generated by conflicting updates ← difficult to deal

③ vector clock don't actually specify how to resolve the conflicts.

solution in place @ Cassandra

↳ rows is broken into columns and updated independently.

```
CREATE TABLE users (
    username text PRIMARY KEY,
    email text,
    phone text
);

INSERT INTO users (username, email, phone)
VALUES ('jbellis', 'jbellis@example.com', '555-5555');

UPDATE users SET email = 'jbellis@illustration.com' WHERE username = 'jbellis';

UPDATE users SET phone = '444-4444' WHERE username = 'jbellis';
```

← image from blog

Benefits

① conflicts can be resolved automatically.

basis timestamp

↑

each column has its own

timestamp value.

See you in next video

Subscribe for more

YouTube - MsDeep Singh

Happy Learning