Orbis

UUID Generation, using Consistent Hashing in Erlang

UUID

[42-bit Timestamp, 12-bit Shard, 10-bit Sequence]

Timestamp

- Use Unix Epoch, in milliseconds, with an offset.
- Subtract offset at generation time, add on decoding time.
- Allows for enough values to outlive the lifetime of the system itself.

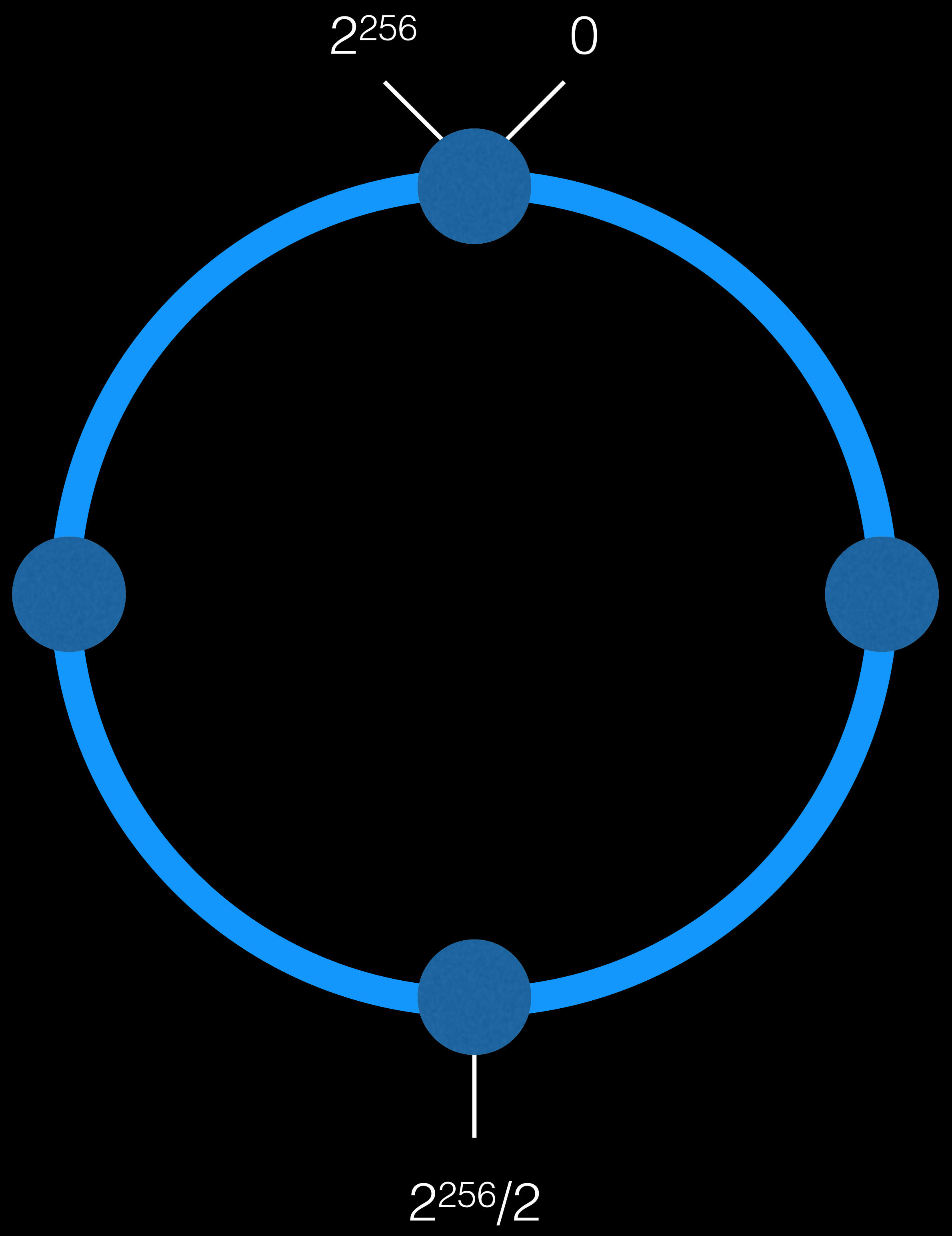
Sequence

- Rolling sequence. Like serial in PostgreSQL. One sequence counter per shard.

Shard

- Allows us to distribute our data out nicely to multiple backend data stores.
- We shard data early, in a sensible way, which allows us to easily move data around later.
- Each shard represents one worker in our ring of workers.

Consistent Hashing


2^{256}
0
$2^{256}/2$