Chapter 7: Design A Unique ID Generator In Distributed System.

You are asked to design a unique ID generator.

Primary key with auto_increment won't work good in distributed systems

Functional Requirements:

* IDs must be unique and sortable
* IDs generated should be incremental, IDs created in the evening should be larger than IDs created in morning in the same day
* IDs only contain numerical values
* IDs should fit into 64 bit requirement
* The system should be able to handle 10000 IDs per second.

Few basic designs which might handle this.

(Multi-Master Replication)

Two Sql master dBs using auto incremental, the new Id that they generate, is just incremented by K to the previous Id generated,

K = No of Sql servers.

instance1 1, 1+3, 4+3

instance2 2, 2+3, 5+3

Instance3 3, 3+3, 6+3

Draw backs.

- Request Routing order should be maintained.
  If 3 requests at first are routed to instance1
  1, 4, 7
  and 4th request goes to instance
  2 → 2
  No garantee of incremental.
- It does not scale well when a server is added or removed.

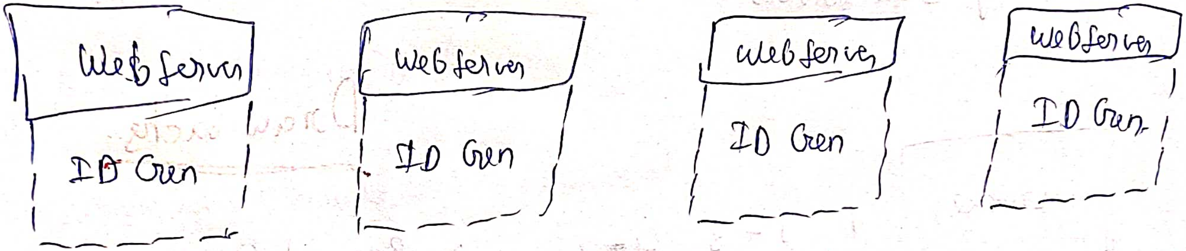
Universally Unique Identifier (UUID)

UUID is a 128-bit number used to identify information.

UUID has very low probability of getting collusion.

"After generating 1 billion UUID's every second for approximately 100 years would the probability of creating a single duplicate reaches 50%".

[Each Webserver contains an ID generator, a web server is responsible for generating IDs independently]



Drawback

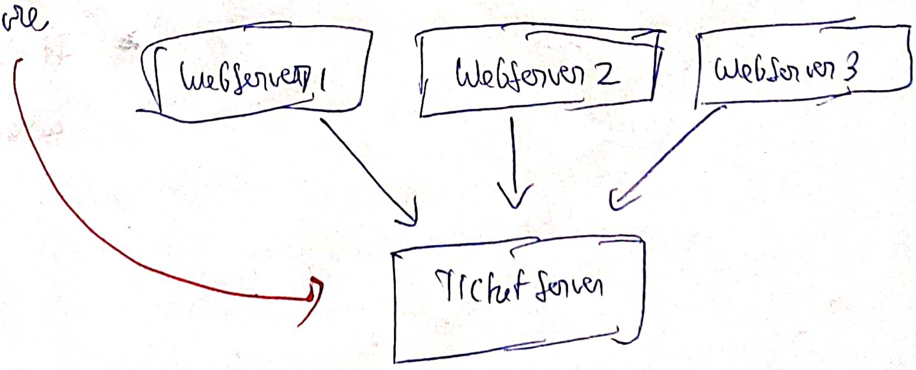
- IDs do not go up with time
- IDs generated are non-numeric, So noway to compare

Ticket Server.

Centralised single database which is used by all webservers.

The database provides a new key which is generated by auto-increment

Cons: Single Point of Failure



(Twitter Snowflake Approach) ***

twitter's unique ID generation system called "snowflake" is.

Instead of generating an ID directly, we divide an ID into different sections.

1bit	41bits	5bits	5bits.	12bits
0	timestamp	dataCenter ID	machine ID	sequence No.

↳ Sum of bits = 64

- Sign bit: 1bit. It will always be 0. This is reserved for future use cases
- Timestamp: 41bits [when the request is coming].
- DataCenter: 5bits. = 2^5 = 32 data Centers are possible
- Machine ID: 5bits = 2^5 = 32 machines can be in each dataCenter
- Sequence number: 12bits. If you ~~can~~ see the timestamp tab, it is trying to store timestamp in millisecond.

(010101010000 1 ----) 41bits $\xrightarrow[\text{decimal}]{\text{to}}$ 291671296188 (epoch)

↓ convert to millisecond.

(Apr 9 2020 16:51:31:958)

So the sequence Number increments by 1 for each new request the machine gets within the same millisecond.

After 1 millisecond. Sequence is set to 0 again.

$2^{41} - 1$ = (Somevalue in(ms)) ~ 69 years.

Sequence Number = 12 bits

(2^{12} = 4096 Combinations). This field is 0, unless the same machine receives a request within 1millisecond

	41 bits	5 bits	5 bits	12 bits
0 bit	timestamp	data Center	machine	Sequence No

Since timestamp is at top so will always be sorted by time.

Within the same timeStamp if two events/requests occur.

Distinguishing b/w them will be done by the data center & machine the request goes to.

If goes to the same machine we have Sequence No to take care of it.

(In 1millisecond Same Machine can uniquely generate 2^{12} new IDs.)

Major problem that happens in this is how to ensure clock synchronisation across different machines?

they do it using Network Time Protocol,

Network Time Protocol Sync between two machines.

(Machine 1 , Machine 2)

* 1 Machine is designated as NTP client and the other as NTP server.

* Client sends request to server asking for current Time
This request contains [Time stamp when Request is Sent].

* Server Response :
→ 1) Time the request was received
2) Current Server Time when response is sent

* Client Receives the Responese

Now it has 4 things.

1. Time request was sent (T_1)
2. Time server received request (T_2)
3. Time response sent by server (T_3)
4. Time response was received by client (T_4).

Client has all details → 1) Clock difference
2) Delay ⇒ It can sync properly Now with machine.