Interview question: Design Twitter (Episode 5)

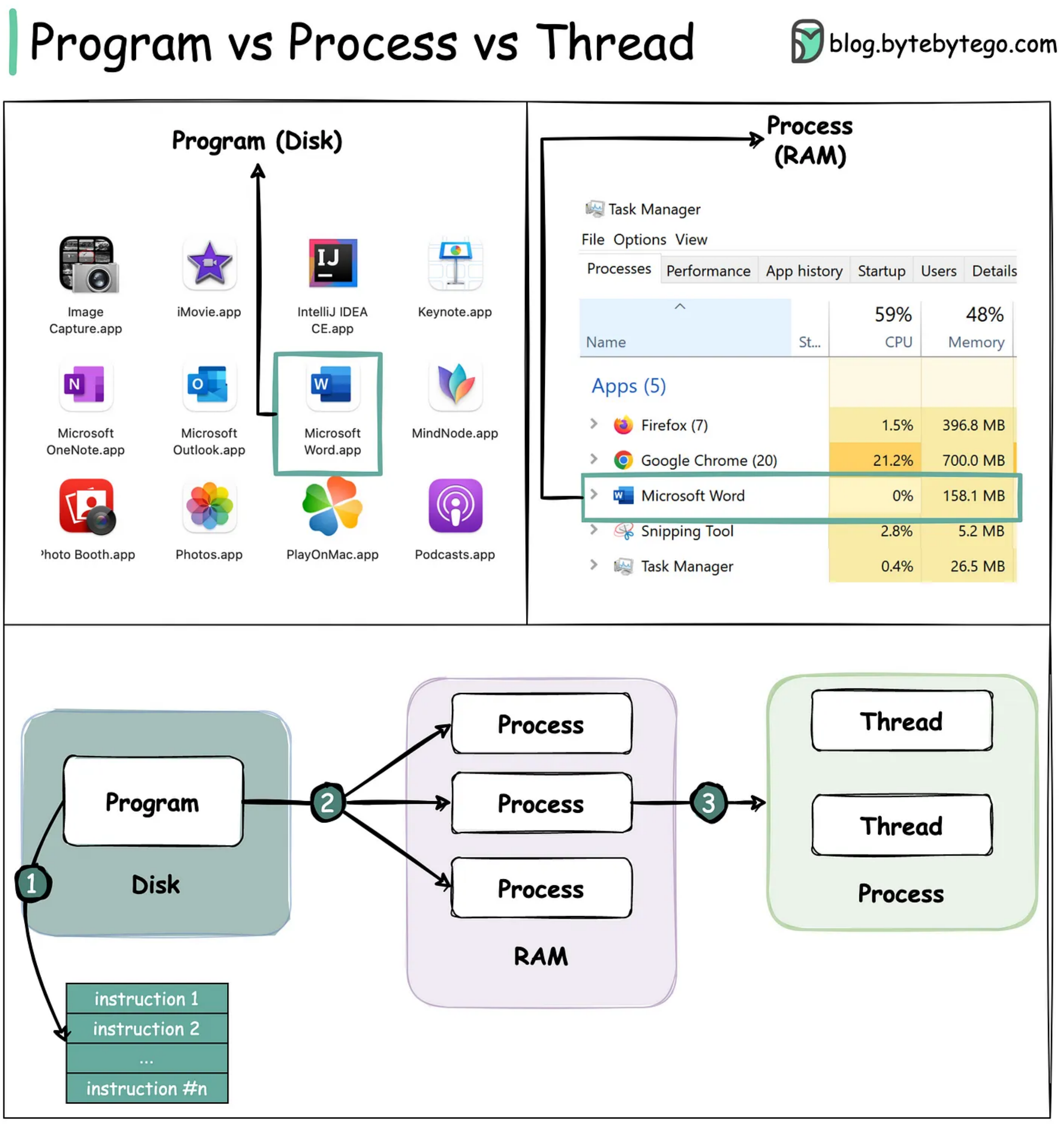
ALEX XU
APR 29, 2022

107 4 Share

In this newsletter, we will talk about the following:

- Process vs Thread
- Interview Question: Design Twitter
- A visual guide on how to choose the right Database.
- Unique ID Generator

Popular interview question: What is the difference between Process and Thread?



Name	St...	59% CPU	48% Memory
Apps (5)			
Firefox (7)		1.5%	396.8 MB
Google Chrome (20)		21.2%	700.0 MB
Microsoft Word		0%	158.1 MB
Snipping Tool		2.8%	5.2 MB
Task Manager		0.4%	26.5 MB

To better understand this question, let's first take a look at what is a Program. A **Program** is an executable file containing a set of instructions and passively stored on disk. One program can have multiple processes. For example, the Chrome browser creates a different process for every single tab.

A **Process** means a program is in execution. When a program is loaded into the memory and becomes active, the program becomes a process. The process requires some essential resources such as registers, program counter, and stack.

A **Thread** is the smallest unit of execution within a process.

The following process explains the relationship between program, process, and thread.

1. The program contains a set of instructions.

2. The program is loaded into memory. It becomes one or more running processes.

3. When a process starts, it is assigned memory and resources. A process can have one or more threads. For example, in the Microsoft Word app, a thread might be responsible for spelling checking and the other thread for inserting text into the doc.

Main differences between process and thread:

- Processes are usually independent, while threads exist as subsets of a process.
- Each process has its own memory space. Threads that belong to the same process share the same memory.
- A process is a heavyweight operation. It takes more time to create and terminate.
- Context switching is more expensive between processes.
- Inter-thread communication is faster for threads.

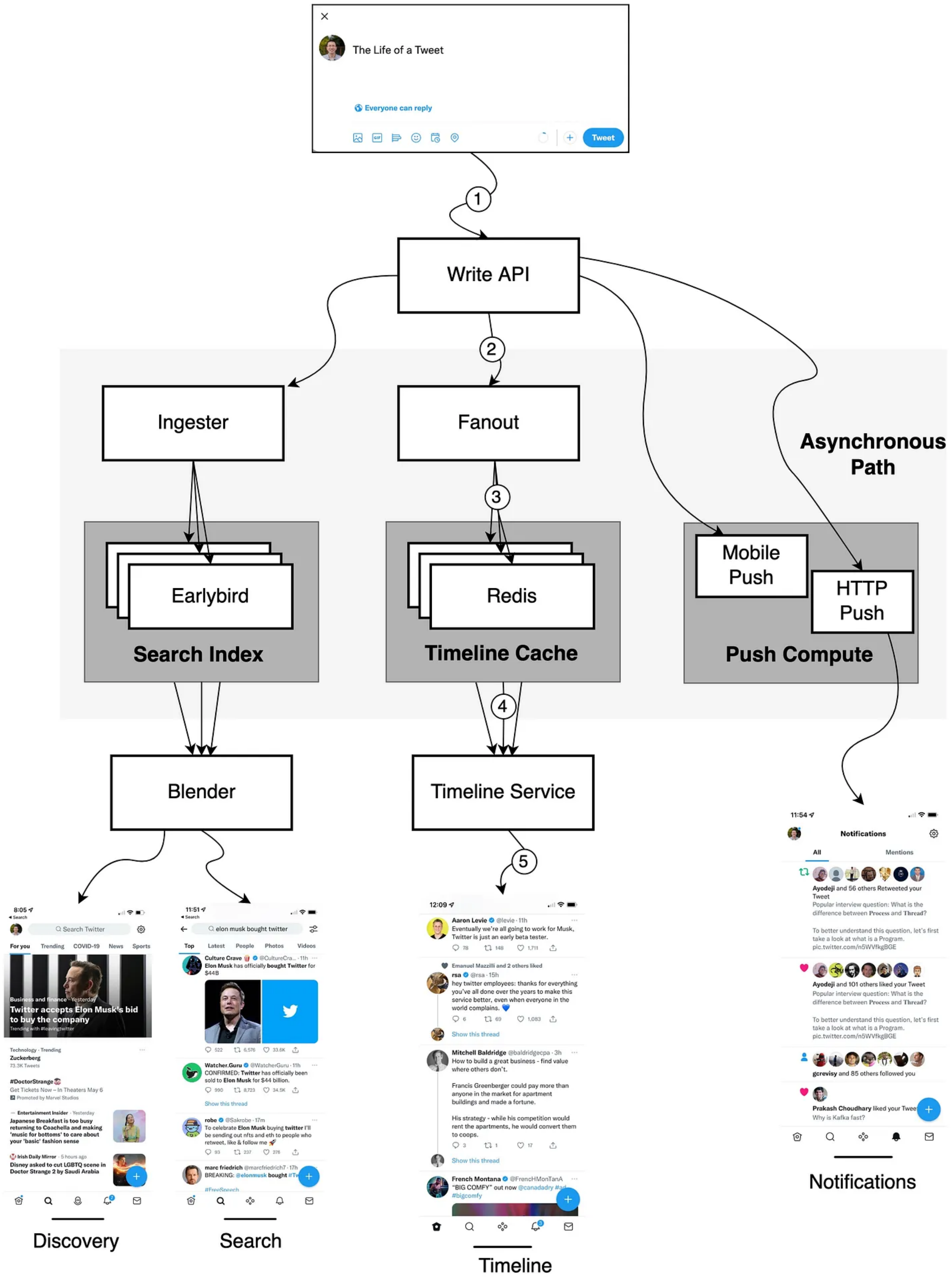
Over to you:

1). Some programming languages support coroutine. What is the difference between coroutine and thread?

2). How to list running processes in Linux?

Interview Question: Design Twitter

This post is a summary of a tech talk given by Twitter in 2013. Let’s take a look.



The life of a Tweet

$\boxed{1}$ A tweet comes in through the Write API.

$\boxed{2}$ The Write API routes the request to the Fanout service.

$\boxed{3}$ The Fanout service does a lot of processing and stores them in the Redis cache.

$\boxed{4}$ The Timeline service is used to find the Redis server that has the home timeline on it.

$\boxed{5}$ A user pulls their home timeline through the Timeline service.

Search & Discovery

- Ingester: annotates and tokenizes Tweets so the data can be indexed.
- Earlybird: stores search index.
- Blender: creates the search and discovery timelines.

Push Compute

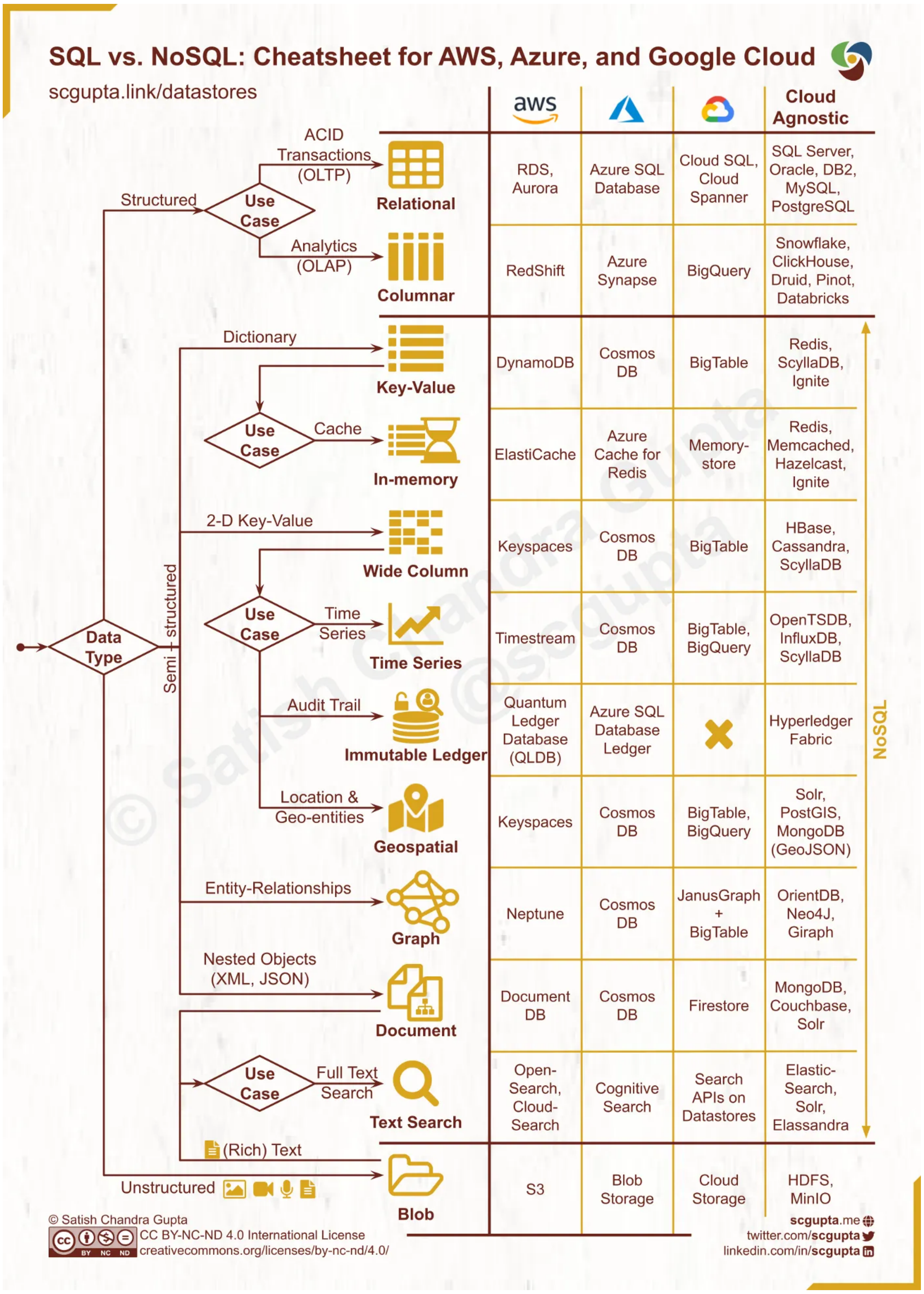
- HTTP push
- Mobile push

Disclaimer: This article is based on the tech talk given by Twitter in 2013 (https://bit.ly/3vNfjRp). Even though many years have passed, it's still quite relevant. I redraw the diagram as the original diagram is difficult to read.

Over to you: Do you use Twitter? What are some of the biggest differences between LinkedIn and Twitter that might shape their system architectures?

A visual guide on how to choose the right Database.

Picking a database is a long-term commitment so the decision shouldn't be made lightly. The important thing to keep in mind is to choose the right database for the right job.



	aws	Azure	Google Cloud	Cloud Agnostic
Relational	RDS, Aurora	Azure SQL Database	Cloud SQL, Cloud Spanner	SQL Server, Oracle, DB2, MySQL, PostgreSQL
Columnar	RedShift	Azure Synapse	BigQuery	Snowflake, ClickHouse, Druid, Pinot, Databricks
Key-Value	DynamoDB	Cosmos DB	BigTable	Redis, ScyllaDB, Ignite
In-memory	ElastiCache	Azure Cache for Redis	Memory-store	Redis, Memcached, Hazelcast, Ignite
Wide Column	Keyspaces	Cosmos DB	BigTable	HBase, Cassandra, ScyllaDB
Time Series	Timestream	Cosmos DB	BigTable, BigQuery	OpenTSDB, InfluxDB, ScyllaDB
Immutable Ledger	Quantum Ledger Database (QLDB)	Azure SQL Database Ledger	✗	Hyperledger Fabric
Geospatial	Keyspaces	Cosmos DB	BigTable, BigQuery	Solr, PostGIS, MongoDB (GeoJSON)
Graph	Neptune	Cosmos DB	JanusGraph + BigTable	OrientDB, Neo4J, Giraph
Document	Document DB	Cosmos DB	Firestore	MongoDB, Couchbase, Solr
Text Search	Open-Search, Cloud-Search	Cognitive Search	Search APIs on Datastores	Elastic-Search, Solr, Elassandra
Blob	S3	Blob Storage	Cloud Storage	HDFS, MinIO

Data can be structured (SQL table schema), semi-structured (JSON, XML, etc.), and unstructured (Blob).

Common database categories include:

- Relational
- Columnar
- Key-value
- In-memory
- Wide column
- Time Series
- Immutable ledger
- Geospatial
- Graph
- Document
- Text search
- Blob

Thanks, Satish Chandra Gupta

Over to you - Which database have you used for which workload?

Unique ID Generator

IDs are very important for the backend. Do you know how to generate globally unique IDs?

In this post, we will explore common requirements for IDs that are used in social media such as Facebook, Twitter, and LinkedIn.

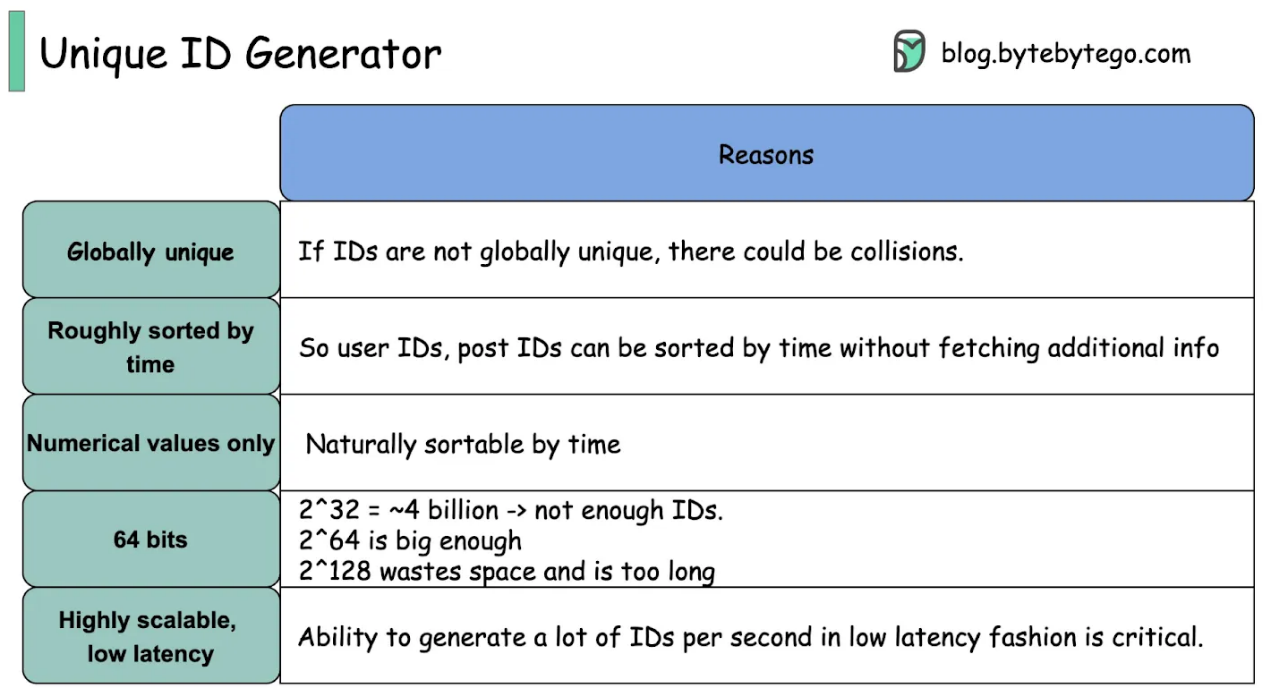
Requirements:

- Globally unique

- Roughly sorted by time
- Numerical values only
- 64 bits
- Highly scalable, low latency

The implementation details of the algorithms can be found online so we will not go into detail here.

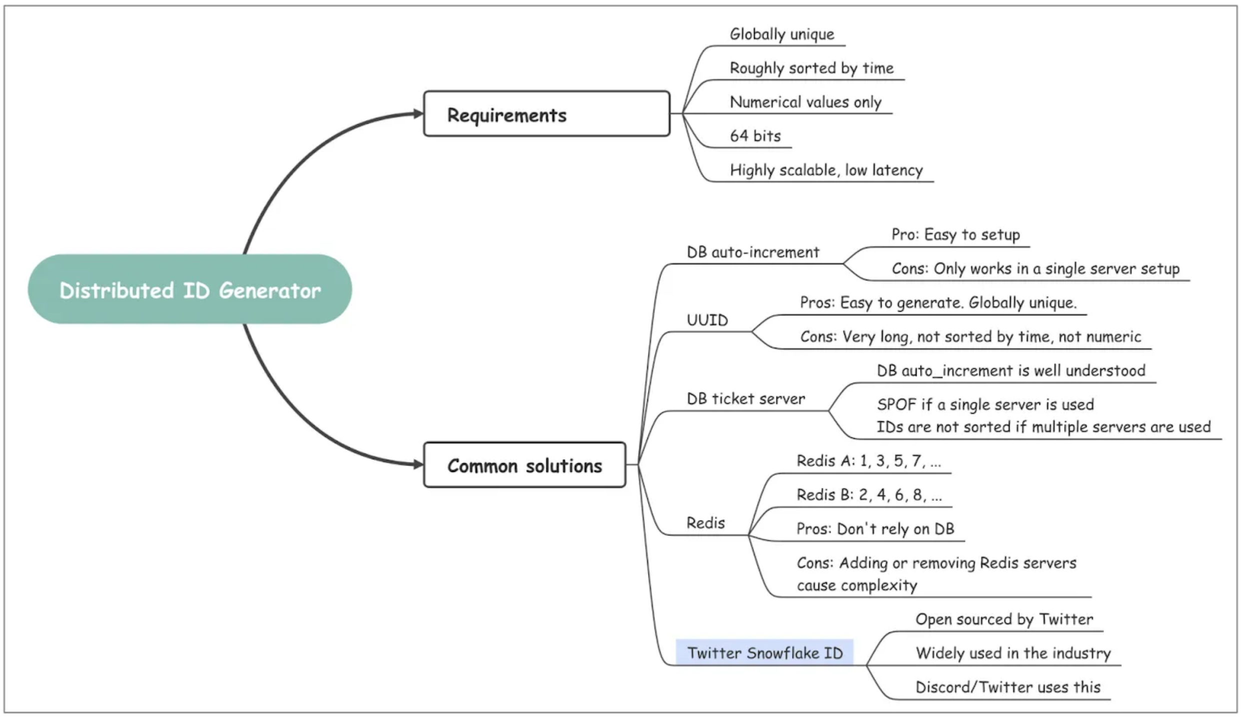
Over to you: What kind of ID generators have you used?



Unique ID Generator

blog.bytebytego.com

	Reasons
Globally unique	If IDs are not globally unique, there could be collisions.
Roughly sorted by time	So user IDs, post IDs can be sorted by time without fetching additional info
Numerical values only	Naturally sortable by time
64 bits	2^32 = ~4 billion -> not enough IDs. 2^64 is big enough 2^128 wastes space and is too long
Highly scalable, low latency	Ability to generate a lot of IDs per second in low latency fashion is critical.


Distributed ID Generator
Requirements
Globally unique
Roughly sorted by time
Numerical values only
64 bits
Highly scalable, low latency
Common solutions
DB auto-increment
Pro: Easy to setup
Cons: Only works in a single server setup
UUID
Pros: Easy to generate. Globally unique.
Cons: Very long, not sorted by time, not numeric
DB ticket server
DB auto_increment is well understood
SPOF if a single server is used
IDs are not sorted if multiple servers are used
Redis
Redis A: 1, 3, 5, 7, ...
Redis B: 2, 4, 6, 8, ...
Pros: Don't rely on DB
Cons: Adding or removing Redis servers cause complexity
Twitter Snowflake ID
Open sourced by Twitter
Widely used in the industry
Discord/Twitter uses this

107 Likes

4 Comments

Write a comment...

Asif May 1, 2022

Big fan of your work and book. I've been reading your blogs regularly.

One request: Can you please also add the sources for the content. It would be helpful to explore that in detail.

LIKE (3) REPLY SHARE

1 reply by Alex Xu

Xi Cheng Apr 30, 2022

Fyi, I love your content in general, but this twitter design article misses the use case/functional breakdown part, which is the thing I love the most about your content (like your book).

Context: I am a product manager and I read your book to understand how to do function/api/system design based on high-level product solution (e.g. eCommerce payment -> pay in and pay out). There are a lot of internet articles like this twitter article (which feels like architecting) but there are very few article about functional design (or at least I don't know how to find them, searching by keywords is quite futile)

LIKE (1) REPLY SHARE

1 reply by Alex Xu

2 more comments...