System Design

Friday, 13 August 2021 2:30 PM

System design basics - https://towardsdatascience.com/system-design-101-b8f15162ef7c
Choosing the right DB:

- https://bikas-katwal.medium.com/mongodb-vs-cassandra-vs-rdbms-where-do-they-st
theorem-1bae779a7a15
- https://www.youtube.com/watch?v=cODCpXtPHbQ&t=110s
- https://www.codekarle.com/system-design/Database-system-design.html
- ACID Properties in RDBMS
    - https://www.geeksforgeeks.org/acid-properties-in-dbms/
    - Atomicity - either full or rollback completely
    - Consistentcy - consistent before and after the transaction.
    - Isolation - one transaction should not affect another
    - Durabililty - durable after system failures
- CAP Theorem

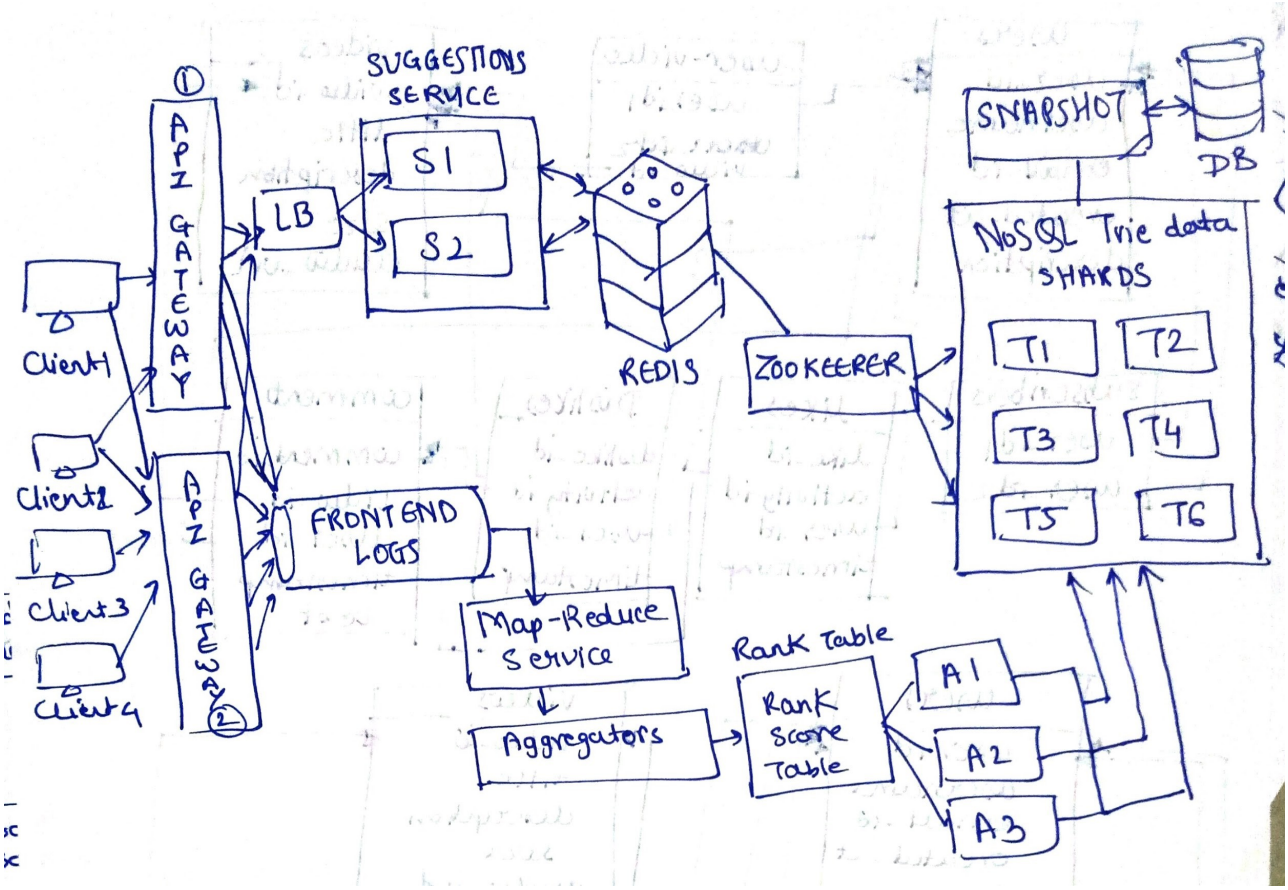
- System Designs
    - https://github.com/codekarle/system-design
    - https://github.com/codekarle/system-design/tree/master/system-design-prep-mater
    - Checklist:
        - https://towardsdatascience.com/system-design-interview-checklist-a-gateway-t
- Fault Tolerance:
    - Fault Tolerance > High Availability
    - Load Balancing & Failover key for Fault Tolerance
    - https://www.imperva.com/learn/availability/fault-tolerance/
- AutoComplete/Type Ahead System Design
    - Amazon interview question: System design / Architecture for auto suggestions | type

tand-in-the-cap-

ial/architecture-diagrams

to-faangs-2b7fac80e423

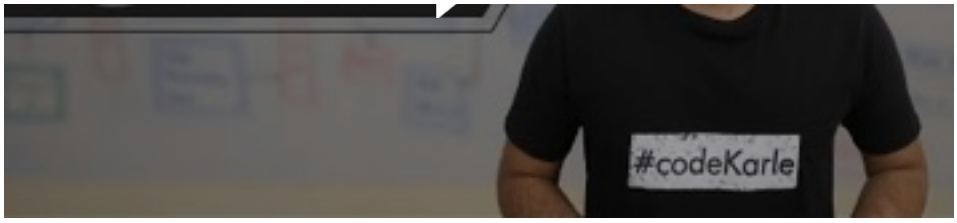
ahead

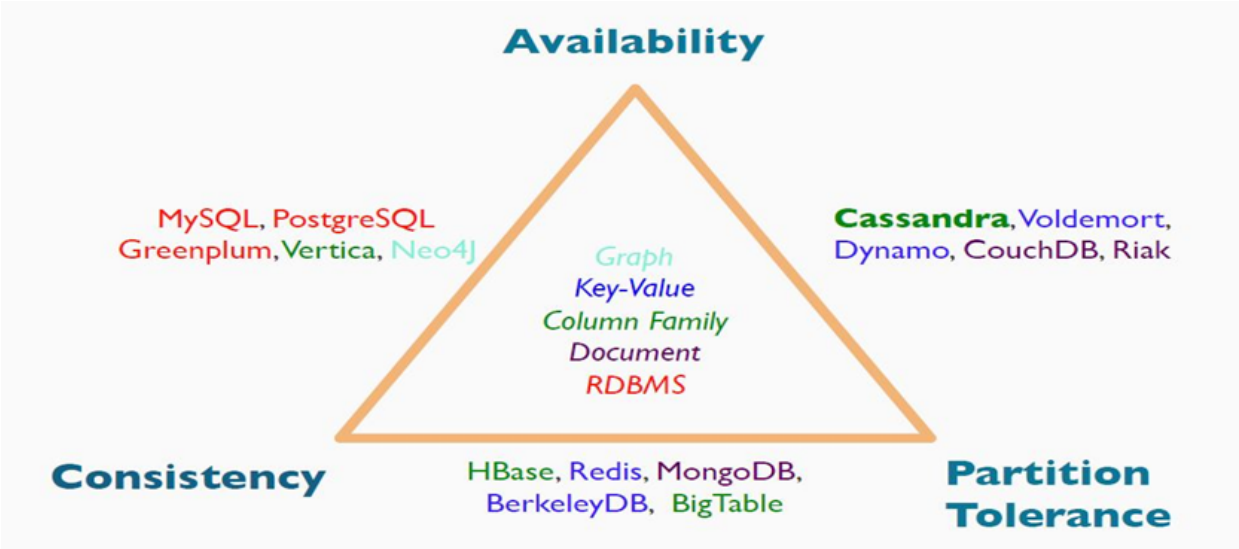
- Trie DataStructure
- Precompute the each nodes with possible suggestions & sort based on Ranking
- https://www.geeksforgeeks.org/auto-complete-feature-using-trie/
- https://gist.github.com/VarunVats9/436d612b7ae68d940822c46f535daa43
- 

- https://systemdesignprep.com/autocomplete
- Rate Limiter
- Notification Service
  - Notification Service System Design Interview Question to handle Billions of users & No

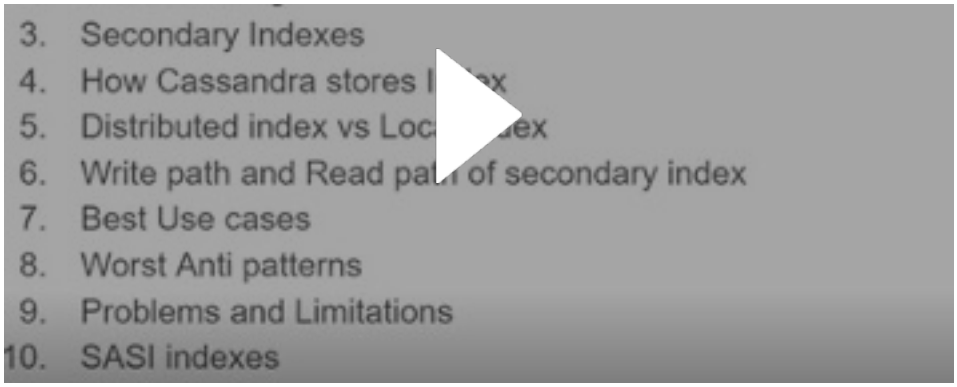
- Bloom Filter

otifications



- Bloom Filter
  - https://www.geeksforgeeks.org/bloom-filters-introduction-and-python-implementation/
- Twitter Design:
  - https://www.geeksforgeeks.org/design-twitter-a-system-design-interview-question/
  - 

- https://www.educative.io/courses/coderust-hacking-the-coding-interview?affiliate_id=5457430901161984
  Caching Strategy:
  https://www.geeksforgeeks.org/write-through-and-write-back-in-cache/
- **Quad Tree** in partitoning of grids in use cases like UBER
  - https://www.educative.io/edpresso/what-is-a-quadtree-how-is-it-used-in-location-based-services
  - **Dynamic Grids**
- **Distributed Transaction Management**
  - https://medium.com/design-microservices-architecture-with-patterns/saga-pattern-for-microservices-distributed-transactions-7e95d0613345
  - https://www.baeldung.com/cs/saga-pattern-microservices
- Cassandra
  - https://www.baeldung.com/cassandra-column-family-data-model
  - Secondary indexes -> Secondary indexes in Apache Cassandra



- Shrading
  - https://medium.com/@jeeyoungk/how-sharding-works-b4dec46b3f6
- Fault tolerance:
  - Fault tolerance is a feature that allows the system to continue to function even if some of its components fail.
  - Eg: using circuit breaker, cut off the api call to failed application
  - And store the message in kafka or fallback table
- Resilient Microservices
  - The ability of an application to **recover from failures** is referred to as **resilience**.
  - Avoid cascading failures
  - Avoid **single point of failures(SOF)** - single instance becomes **bottleneck** always
  - Handle failures gracefully using below patterns:
  - Retry pattern
  - Circuit breaker pattern
  - Timeout design pattern
  - Bulkhead design pattern

  - https://cloudcomputingtechnologies.com/how-to-make-microservices-resilient/
  - https://dzone.com/articles/making-your-microservices-resilient-and-fault-tole-1
  - https://www.jrebel.com/blog/microservices-resilience-patterns
  - **Identify bottlenecks and resolve them**
  - If there is any **single point of failure** in our system, we need to remove them. This may cause availability issues, which is a huge concern.

  - We need to have **enough replicas of the data** to still serve our users if we

    lose a few servers. If there is no replica of the data, and for some reason, data

    is lost, the system does not have the data. The system will have **reliability**

issues.

- Similarly, we need to have enough **copies of different services** running so that a few failures do not cause a system's total shutdown.

- Bulkhead pattern
  - Restrict cascading failures by isolating the threads only for specific operations. - resilience4j provides bulkhead using semaphore with
    - Concurrent requests
    - Wait time of threads in blocking queue.
  - https://dzone.com/articles/resilient-microservices-pattern-bulkhead-pattern
- Camunda as stateful resilence
  - https://blog.bernd-ruecker.com/use-camunda-without-touching-java-and-get-an-easy-to-use-rest-based-orchestration-and-workflow-7bdf25ac198e
- EDA vs RDA
  - https://www.techtalksbyanvita.com/post/event-driven-vs-request-driven-rest-architecture
  - https://www.neebal.com/blog/advantages-and-disadvantages-of-event-driven-architecture
- SOA vs Microservices Adv & DisAdv
  - https://cloudacademy.com/blog/microservices-architecture-challenge-advantage-drawback/

- **CQRS**
  - https://medium.com/design-microservices-architecture-with-patterns/event-sourcing-pattern-in-microservices-architectures-e72bf0fc9274
  - https://medium.com/design-microservices-architecture-with-patterns/materialized-view-pattern-f29ea249f8f8
- Fail Fast and then reverse using camunda
  - https://blog.bernd-ruecker.com/fail-fast-is-not-enough-84645d6864d3
  - https://blog.bernd-ruecker.com/architecture-options-to-run-a-workflow-engine-6c2419902d91
- Microservices design pattern
  - https://towardsdatascience.com/microservice-architecture-and-its-10-most-important-design-patterns-824952d7fa41
- 2 phase lock
  - https://www.geeksforgeeks.org/two-phase-locking-protocol/
  - https://www.tutorialspoint.com/explain-about-two-phase-locking-2pl-protocol-dbms
- Transaction locking & row versioning

    - https://docs.microsoft.com/en-us/sql/relational-databases/sql-server-transaction-locking-and-row-versioning-guide?view=sql-server-ver16
    - **open a transaction and do a SELECT 1 FROM mytable WHERE clause to match row FOR UPDATE**;
    - https://stackoverflow.com/questions/20933528/postgresql-lock-row-on-indefinitely-time#:~:text=You%20can%20just%20open%20a,open%20until%20you're%20done.
    - https://www.postgresql.org/docs/current/explicit-locking.html#ADVISORY-LOCKS
    - https://support.unicomsi.com/manuals/soliddb/100/index.html#page/SQL_Guide/5_ManagingTransactions.06.4.html
    - https://www.baeldung.com/jpa-pessimistic-locking
        - Shared lock
        - Explicit lock
    - https://www.baeldung.com/jpa-optimistic-locking
    - https://www.baeldung.com/java-jpa-transaction-locks
        - `@Lock(LockModeType.PESSIMISTIC_READ) @QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value = "3000")})`
    - https://hackernoon.com/optimistic-and-pessimistic-locking-in-jpa
- Distributed Caching
    - https://medium.com/system-design-concepts/distributed-cache-system-design-9560f7dd07f2
    - https://www.educative.io/courses/grokking-the-system-design-interview/YQlK1mDPgpK
    - https://www.enjoyalgorithms.com/blog/consistent-hashing-in-system-design
- Consistent hashing
    - https://www.acodersjourney.com/system-design-interview-consistent-hashing/
    - https://medium.com/system-design-blog/consistent-hashing-b9134c8a9062
    - https://www.toptal.com/big-data/consistent-hashing#:~:text=according%20to%20Wikipedia).-,Consistent%20Hashing%20is%20a%20distributed%20hashing%20scheme%20that%20operates%20independently,without%20affecting%20the%20overall%20system.
    - https://www.enjoyalgorithms.com/blog/consistent-hashing-in-system-design
    - https://ably.com/blog/implementing-efficient-consistent-hashing
- Load Balancer types
    1. Consistent Hashing
    2. Round Robin
    3. Weighted Round Robin

4. Least Connection

- Database Shrading
  - https://www.geeksforgeeks.org/database-sharding-a-system-design-concept/?ref=rp
- Redis
  - Store list, TTL expiration
  - https://www.linkedin.com/pulse/system-design-basics-caching-omar-ismail?trk=articles_directory
- Optimizing kafka consumers
  - https://strimzi.io/blog/2021/01/07/consumer-tuning/
  - https://stackoverflow.com/questions/40781548/difference-between-request-timeout-ms-and-timeout-ms-properties-of-kafka-produce
  - https://www.javierholguera.com/2018/01/01/timeouts-in-kafka-clients-and-kafka-streams/
- Zookeeper reponsibilities
  - https://dattell.com/data-architecture-blog/what-is-zookeeper-how-does-it-support-kafka/
- Microservices pattern
  - https://akfpartners.com/growth-blog
- **Circuit Breaker**
  - https://www.credera.com/insights/circuit-breaker-pattern-in-spring-boot

```
hystrix:
  command:
    default:
      circuitBreaker:
        errorThresholdPercentage: 50 # 50%
        sleepWindowInMilliseconds: 5000 # 5s
    customCommandKey:
      fallback:
        enabled: false
      circuitBreaker:
        errorThresholdPercentage: 75 # 75%
        sleepWindowInMilliseconds: 15000 # 15s
```

  - https://www.baeldung.com/spring-cloud-netflix-hystrix

Commits

- **UI**
- Implemented event source(SSE interaction), mongo aggregations as part of this commit ->
  - https://tools.lowes.com/stash/projects/E-STM/repos/stamp-task-dashboard-

ui/commits/32eddf5c3ca8fdbaa3c21bd32b52d902471e2140#src/containers/CheckinLanding.js

- **BackEnd:**
- Created reactive application using Spring Webflux as part of this commit ->
  - https://tools.lowes.com/stash/projects/E-STM/repos/stamp-inventory-tenant/pull-requests/2/overview
- IMPLEMENTED SSE STREAMING LOGIC AS PART OF THIS COMMIT
  - https://tools.lowes.com/stash/projects/E-STM/repos/stamp-task-dashboard-query-service/pull-requests/4/diff#src/main/java/com/lowes/stores/taskdashboardqueryservice/service/DashboardStreamingServiceImpl.java
  - https://tools.lowes.com/stash/projects/E-STM/repos/stamp-task-dashboard-query-service/pull-requests/4/diff#src/main/java/com/lowes/stores/taskdashboardqueryservice/controller/DashboardStreamsController.java
- Event sourcing logic implemented in this commit
  - https://tools.lowes.com/stash/projects/E-STM/repos/stamp-task-status-processor-service/pull-requests/2/diff#src/main/java/com/lowes/stores/taskdashboardservice/commandservice/service/CommandServiceImpl.java
- Camunda based workflow application created as part of this commit:
  - https://tools.lowes.com/stash/projects/E-STM/repos/stamp-task-engine/pull-requests/4/diff#src/main/java/com/lowes/workflowengine/service/impl/HistoryManagementServiceImpl.java
- Confluence page on design:
  - https://tools.lowes.com/confluence/display/LES/Firebase+Notification+fallback+to+Event+Notification#FirebaseNotificationfallbacktoEventNotification-DesignImplementationinphases:
  - https://tools.lowes.com/confluence/display/LES/WebFlux+Reactive+Approach
  - https://tools.lowes.com/confluence/display/LES/STaMP+Task+Dashboard+Architecture
  - https://tools.lowes.com/confluence/display/LES/Firebase+Notification+fallback+to+Event+Notification#FirebaseNotificationfallbacktoEventNotification-DesignImplementationinphases:

Order mangement service
Person doing place order->order_packaging->deliver_order.

Store dashboard screen