mongoDB

UTSA
The University of Texas at San Antonio™

Understanding MongoDB Design Architecture and Performance

Presentation By: Amjad Alqahtani, Nicholas Winkelmann & Caleb Alva

Agenda

- Introduction
- Background
- System Architecture of MongoDB
- Performance Experiment
- Demo *[if there is time]*
- Results and Analysis
- Conclusion & Future Work

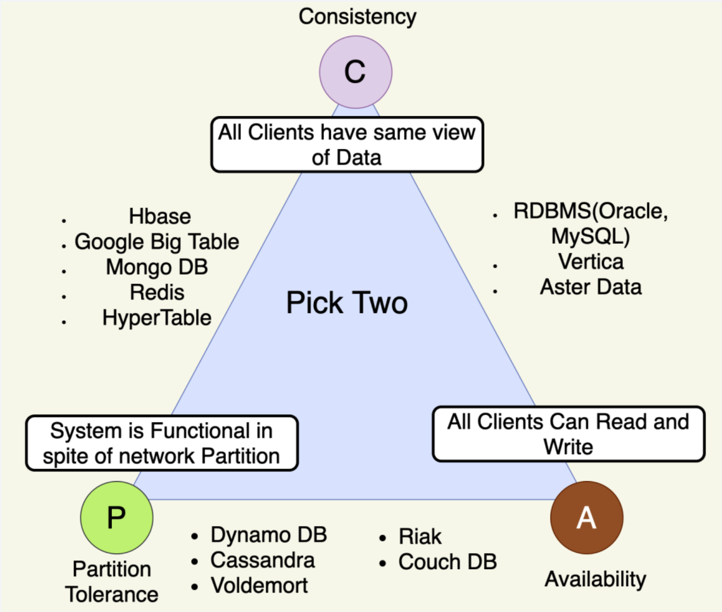
Introduction

Emergence of NoSQL Databases

- Growth of **web applications** and **cloud computing** makes single-node data management impractical.
- NoSQL databases address challenges in distributing large datasets across multiple machines:
  - Examples: MongoDB, Couchbase, Redis, Amazon DynamoDB, Apache Cassandra, Google Cloud Bigtable, Neo4j, ArangoDB, ScyllaDB.

MongoDB’s Distributed System Architecture

- Relies on **sharding** to partition and distribute data across nodes, enabling **horizontal scalability**.
- Optimized for:
  - **High availability**
  - **Partition tolerance**
  - **Tunable consistency** (guided by CAP theorem).



Advantages and Challenges

Advantages of MongoDB

- **Flexible schema design** (JSON-like documents):
    - Simplifies representation and adapts to semi-structured data.
    - Accommodates evolving document structures.
- Supports **high-throughput operations**, **horizontal scaling**, and **failover mechanisms**.

Challenges in Distributed Systems

- Trade-offs between **availability, consistency,** and **fault tolerance** (CAP theorem).
- MongoDB faces challenges with:
    - **Write** and **scan operations** in distributed environments.
    - **Latency** due to coordination across nodes.

Objective of This Study

- Evaluate MongoDB's performance in **cloud environments** using **Yahoo! Cloud Serving Benchmark (YCSB)**.
- Focus on **read, write, and scan workloads** to highlight strengths and identify performance bottlenecks.

Background

MongoDB

MongoDB: A Leading Document-Based NoSQL Database

- Recognized for its efficient management of distributed systems.
- Comparative analyses (e.g., with CouchDB and Couchbase) aid users in selecting suitable databases for applications.

Schema Design Flexibility

- MongoDB's **JSON-like document structure** is a key factor driving its popularity and adaptability.

Motivation for this Study

Existing Research Observations

- MongoDB performs well in **high-throughput read operations**.
- **Write** and **scan operations** face challenges in distributed environments due to coordination overhead.

Need for Further Study

- Limited focus on MongoDB’s real-world performance under diverse workloads.
- Opportunities to explore and address **bottlenecks** in distributed setups.

System Architecture of MongoDB

The Document Model

- **Intuitiveness**
  - Maps documents directly to code objects for efficiency.
  - Data stored together is accessed together, reducing unnecessary restructuring.
- **Flexibility**
  - Self-described schemas eliminate the need for pre-defining.
  - Supports varying document structures within the same database.
  - Schema validation is optional for added structure control.
- **Universality**
  - Leverages **Binary JSON (BSON)** for improved data representation.
  - Adaptable for diverse applications and optimized for binary data processing.

Availability and Scalability

Availability

- **Replica Sets**
  - Up to 50 data copies ensure high availability and resiliency.
  - Enables scaling of read operations and minimizes query delays.
- **Write Concerns**
  - Customize replication for enhanced data safety.

Scalability Features

- **Vertical Scaling**
  - Adjust instance sizes as needed.
- **Sharding**
  - Automates horizontal scaling for write-heavy workloads.
  - Types:
    - **Ranged Sharding**: Groups documents by shard key value.
    - **Hashed Sharding**: Ensures uniform data distribution.
    - **Zoned Sharding**: Applies rules for document placement.

Privacy and Security

- **Authentication**:
  - Uses **SCRAM-256** and enterprise integrations.
- **Authorization**:
  - **Role-Based Access Controls (RBAC)** restrict data access.
- **Auditing**:
  - Comprehensive audit logs for security oversight.
- **Network Isolation**:
  - Hosted in **Virtual Private Cloud (VPC)** environments.
- **Encryption**:
  - End-to-end encryption ensures data security during storage and transfer.

Performance Experiment

Experiment Setup

Objective

- Evaluate MongoDB performance in a distributed environment using **Yahoo! Cloud Serving Benchmark (YCSB)**.
- Focus on **read, write, and scan workloads** to highlight strengths and identify performance bottlenecks.

Database Configuration

- **MongoDB Atlas (Free Tier)** on AWS (M0 Sandbox).
- Replica set with 3 nodes in **us-east-1** region.
- Basic deployment for learning and exploration.

Virtual Machines

- Shared vCPUs and RAM.
- 512 MB max storage.
- No specific throughput guarantees.

Development Tools

- **Visual Studio Code** with MongoDB extension for cluster interaction.
- **YCSB GitHub Repo** for benchmark implementation.
- **Java** and **Maven** for runtime environment and workload generation.

Experiment Workflow

- **Cluster Setup:**
  - Created using MongoDB Atlas M0 Sandbox.
  - Configuration included replication for redundancy.
  - Visual Studio Code was used to interact with the cluster.
- **Benchmark Preparation:**
  - Cloned **YCSB Repo** to the testing machine.
  - Installed **Java** and **Maven** for dependencies and execution.
  - Local MongoDB installation enabled seamless cluster interaction.
- **Execution of YCSB Workloads:**
  - Tested six workloads: **Workload A-F** on the MongoDB cluster.
  - Each workload simulated different operational patterns to evaluate performance.

Experiment Evaluation

- **Performance Metrics Tested**
  - **Workload A:** Read-Write mix.
  - **Workload B:** Read-heavy.
  - **Workload C:** Read-only.
  - **Workload D:** Read operations with user distribution.
  - **Workload E:** Short-range scans.
  - **Workload F:** Read-modify-write operations.
- **Initial Findings**
  - **Strengths:** MongoDB demonstrated high efficiency in **read-heavy** workloads.
  - **Challenges:** Write-heavy and scan operations faced performance bottlenecks, aligning with expected distributed system behavior.

Demo

Results and Analysis

Results Overview

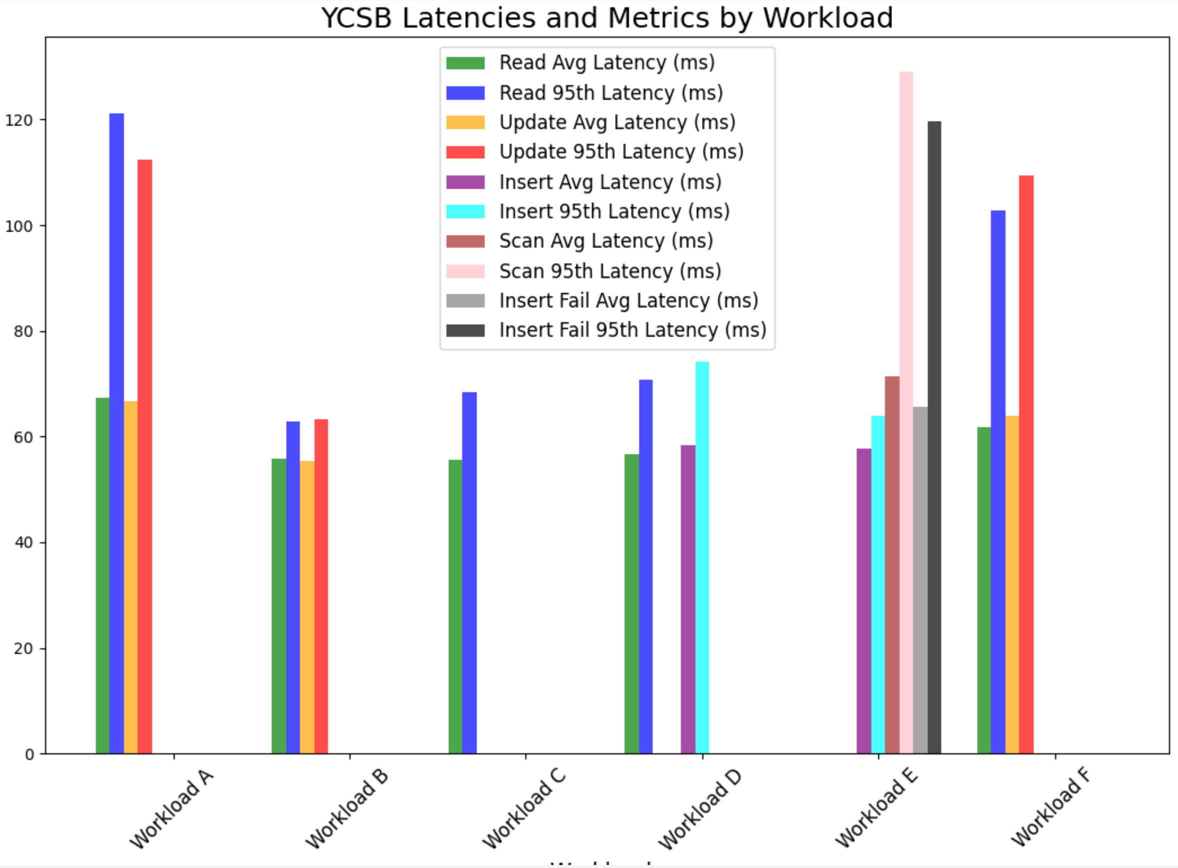
Performance Evaluation

- **Benchmark Tool**: Yahoo! Cloud Serving Benchmark (YCSB).
- **Workloads Tested**: A, B, C, D, E, and F.
- Each workload simulates unique application scenarios with varying combinations of **read**, **write**, **update**, and **scan** operations.

Key Metrics Evaluated

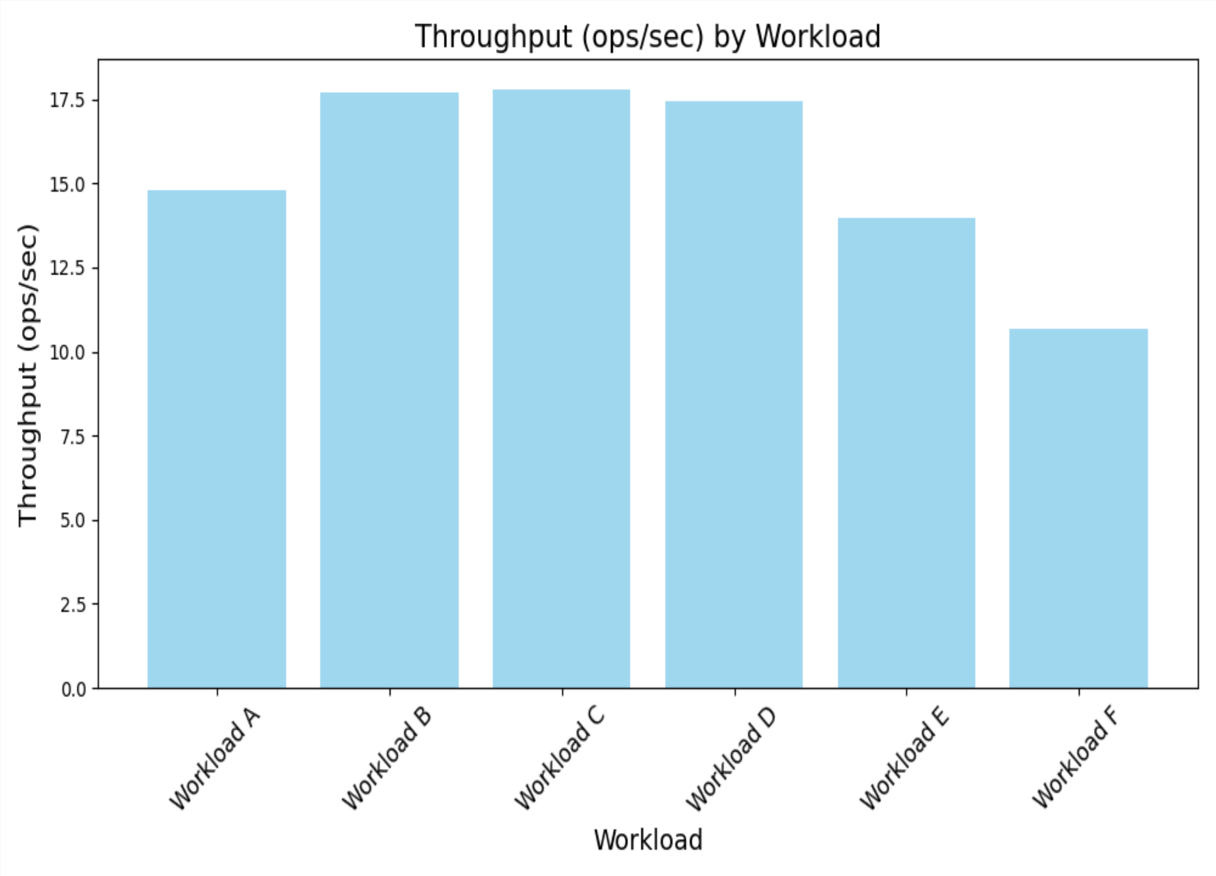
- **Latency** (Average and 95th Percentile).
- **Throughput** (Requests per Second).

Latency Performance Across Workloads



MongoDB excels in **read operations** but shows challenges in **update-heavy and scan-heavy workloads**, as reflected in the latency metrics.

Throughput Performance Across Workloads



- **Workload C (Read-Only)**: Highest throughput (~19 ops/sec), showcasing MongoDB's strength in handling read-intensive applications.
- **Workloads B & D**: Strong throughput (~17 ops/sec), driven by MongoDB's optimized read operations and efficient replica sets.
- **Workload A (Update-Heavy)**: Balanced throughput (~15 ops/sec), demonstrating consistent performance for read and update operations.
- **Workload E (Scan-Heavy)**: Moderate throughput (~14 ops/sec), reflecting challenges in scan-heavy analytical queries.
- **Workload F (Read-Modify)**: Lowest throughput (~10 ops/sec), highlighting the performance impact of distributed coordination in complex workloads.

Conclusion

Evaluated MongoDB’s **system architecture** and **performance** using YCSB workloads in a cloud-based environment.

Study Summary

Key Strengths

- Excels in **read-heavy scenarios**.
- Utilizes **replica sets** and **efficient index management** effectively.

Key Challenges

- Struggles with **write-intensive** and **scan-heavy workloads** due to:
  - Distributed synchronization overhead.
  - inefficiencies in query execution for analytical operations.

Future Work

Future Work

Focus Areas for Improvement

- **Optimizing Query Planning:**
  - Advanced techniques to improve performance for analytical and scan-heavy workloads.
- **Enhancing Transaction Coordination:**
  - Improved handling of **read-modify-write operations**.
  - Reduction of latency in distributed environments.
- **Exploring Advanced Strategies:**
  - **Machine learning-driven predictive indexing** for better query optimization.
  - **Alternative sharding strategies** for scalability in diverse use cases.

Comparative Analysis

- Future testing of MongoDB against other NoSQL databases under similar conditions to identify relative strengths and weaknesses.

Thank you!

utsa.edu