An Empirical Analysis of Query Router Scalability in Sharded MongoDB Clusters

Adeniyi Babatope
School of Enterprise Computing and Digital Transformation, TU Dublin, Ireland
X00218722@mytudublin.ie

1. Introduction

As distributed databases become the backbone of modern large-scale applications, the ability to scale horizontally is paramount. While the scalability of the data layer (shards) is well-documented, the performance implications of the routing layer, specifically the ‘mongos‘ process remain under-explored. This thesis presents a rigorous empirical analysis of query router scalability in sharded MongoDB clusters, examining the effects of scaling ‘mongos‘ nodes under high-concurrency workloads to identify precise saturation points.

2. Problem Statement

- **The Problem:** Industry practices often rely on "rule of thumb" heuristics for provisioning routers, leading to either performance bottlenecks (under-provisioning) or resource wastage (over-provisioning).
- **The Gap:** Lack of empirical data quantifying the saturation point of a single ‘mongos‘ process under high concurrency.
- **Research Question:** *"How does horizontally scaling the number of ‘mongos‘ routers affect throughput and latency across varying shard counts (1 to 9)?"*

3. Methodology

A strictly controlled **Factorial Design** experiment was conducted using Infrastructure as Code (IaC) to ensure reproducibility.

Experimental Setup

- **Infrastructure:** AWS `c5.xlarge` (Compute Optimized) for all nodes.
- **Orchestration:** Terraform Ansible.
- **Workload:** YCSB Workload A (50% Read / 50% Update).
- **Dataset:** 1,000 Records (Fits in RAM) to isolate CPU/Network performance from Disk I/O.
- **Sharding: Hashed Sharding** on `_id` to guarantee uniform distribution and prevent "Hot Shards."

Matrix of Variables:

Variable	Values
Shards	1, 3, 6, 9
Mongos Routers	1, 3, 6, 9
Concurrent Clients	1, 3, 6, 9

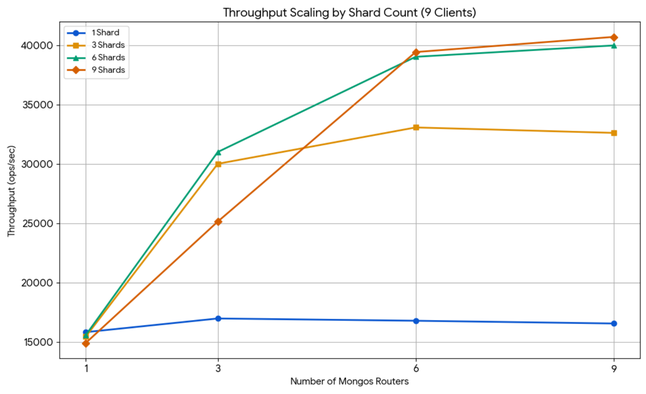
5. Impact of Consistency

A secondary analysis compared the Load Phase (‘w=majority‘) vs. Run Phase (‘w=1‘).

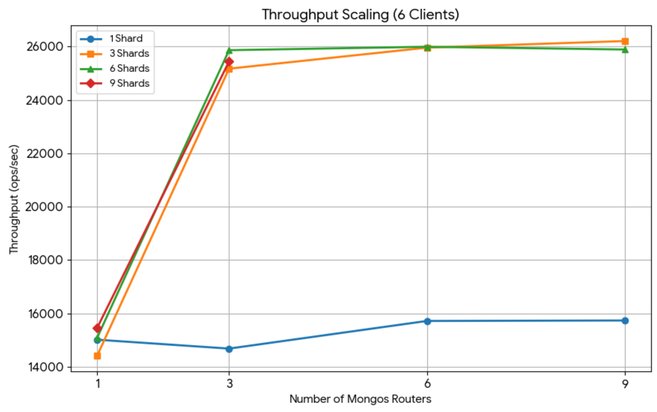
Mode	Throughput	Latency
`w=1`	≈ 4,500 ops/sec	2.3 ms
`w=majority`	≈ 1,350 ops/sec	6.0 ms

Insight: Strong consistency introduces a **3.3× latency penalty**, shifting the bottleneck from the router CPU to network replication lag.

4. Key Findings: The "Rule of 6"



Peak Load (9 Clients): Throughput plateaus at 6 Routers.



1-Shard Saturation: Adding routers yields zero gain.

Key Findings:

- **The "Rule of 6":** In a 9-Shard cluster under peak load (40k ops/sec), scaling from 1 to 6 routers yielded linear gains. However, scaling from **6 to 9 routers provided <2% improvement**, identifying the saturation point.
- **1-Shard Ceiling:** For a single-shard cluster, throughput flatlined at ≈16k ops/sec regardless of router count. This confirmed **Logical Contention** within the WiredTiger storage engine (locking overhead) was the bottleneck, not the routers.
- **Client-Bound Baseline:** Single-client tests flatlined at 4,500 ops/sec, proving that low-concurrency workloads are latency-bound (Amdahl’s Law) and cannot benefit from horizontal scaling.

6. Conclusions & Future Work

Conclusions

- **Diminishing Returns:** Over-provisioning routers (beyond a 1:1.5 ratio) wastes resources without improving throughput.
- **Bottleneck Shift:** The bottleneck dynamically shifts from *Client* (low load) → *Router* (medium load) → *Network/Locking* (peak load).
- **Efficiency:** The `mongos` process is highly efficient (<5% CPU usage), suggesting bottlenecks are due to network coordination, not compute.

Future Work

- **Sidecar Deployment:** Evaluating Client-Side `mongos` (e.g., in K8s pods).
- **Disk-Bound Tests:** Exceeding RAM to test "Thundering Herd" scenarios.
- **Client-Side vs. Server-Side Routing:** Investigating the performance impact and architectural trade-offs between client-routed and server-routed database queries.

Scan for Video