# MongoDB TPS, CPU, and Latency Monitoring Script – Explanation

This document explains how the monitoring script captures real-time MongoDB throughput (TPS), latency, and CPU utilization metrics.

## 1️⃣ Data Source

The script connects to the primary MongoDB instance using the 'mongosh' shell and runs:  
 db.serverStatus()  
  
This command provides live metrics for operation counters, latency, and system resource statistics.

## 2️⃣ TPS (Transactions Per Second) Calculation

The script extracts data from the 'opcounters' field in db.serverStatus():  
 {  
 'insert': 123456,  
 'query': 456789,  
 'update': 111222,  
 'delete': 33344,  
 'command': 987654  
 }  
  
It takes two snapshots a few seconds apart and calculates per-second operation rates:  
 Inserts/s = (current.insert - previous.insert) / interval  
 Updates/s = (current.update - previous.update) / interval  
 Deletes/s = (current.delete - previous.delete) / interval  
 Reads/s = (current.query - previous.query) / interval  
 Commands/s= (current.command - previous.command) / interval  
  
TotalOps/s = sum of all above.

## 3️⃣ Latency Capture

MongoDB’s 'opLatencies' metrics provide cumulative latency (in microseconds) and operation counts:  
 'reads': { 'latency': 12500, 'ops': 3000 },  
 'writes': { 'latency': 6000, 'ops': 2000 },  
 'commands': { 'latency': 9000, 'ops': 5000 }  
  
The script converts these to milliseconds and computes average latency per operation:  
 ReadLatency (ms) = reads.latency / reads.ops / 1000  
 WriteLatency (ms) = writes.latency / writes.ops / 1000  
 CmdLatency (ms) = commands.latency / commands.ops / 1000  
  
The weighted mean of these gives the overall AvgLatency.

## 4️⃣ CPU Utilization

The script uses the Linux 'top' command to capture system CPU utilization:  
 top -bn1 | grep 'Cpu(s)'  
  
It extracts the non-idle percentage to compute total CPU usage:  
 CPU = 100 - idle%  
  
This represents how much of total CPU is being consumed at that moment.

## 5️⃣ CPU per Operation

To relate CPU usage with MongoDB workload, the script computes:  
 CPU/Op = CPU% / TotalOps/s  
  
This shows the approximate CPU cost per MongoDB operation.

## 6️⃣ Output and Logging

At each sampling interval, the script prints and logs metrics in the following format:  
 Timestamp | Inserts/s | Updates/s | Deletes/s | Reads/s | Commands/s | TotalOps/s | ReadLatency | WriteLatency | CmdLatency | AvgLatency | CPU | CPU/Op  
  
The results are continuously written to:  
 /tmp/mongo\_tps\_cpu\_latency\_report.csv

## ✅ Summary Table

|  |  |  |
| --- | --- | --- |
| Metric | Source | Description |
| Inserts/s, Updates/s, Deletes/s | opcounters | Operation rates per second |
| Reads/s, Commands/s | opcounters | Read and command throughput |
| ReadLatency, WriteLatency, CmdLatency | opLatencies | Average latency (ms) per operation type |
| AvgLatency | Calculated | Weighted average of all latencies |
| CPU % | top command | System-wide CPU usage |
| CPU/Op | Calculated | Approximate CPU consumed per MongoDB operation |

This setup helps correlate MongoDB operation throughput, latency, and CPU usage to identify performance bottlenecks effectively.