# MySQL Performance Monitoring: Prometheus vs. MySQL Built-in Tools

## What Can Be Monitored? (Performance Metrics List)

| Performance Metric | Prometheus (With mysqld\_exporter) | MySQL Built-in Tools (Performance Schema, SYS, etc.) |
| --- | --- | --- |
| MySQL Uptime | ✅ mysql\_global\_status\_uptime | ✅ SHOW GLOBAL STATUS LIKE ‘Uptime’; |
| Active Connections | ✅ mysql\_global\_status\_threads\_connected | ✅ SHOW STATUS LIKE ‘Threads\_connected’; |
| Total Queries Per Second | ✅ rate(mysql\_global\_status\_queries[5m]) | ✅ SHOW STATUS LIKE ‘Queries’; |
| Slow Queries Per Second | ✅ rate(mysql\_global\_status\_slow\_queries[5m]) | ✅ SHOW STATUS LIKE ‘Slow\_queries’; |
| CPU Usage | ✅ rate(process\_cpu\_seconds\_total[5m]) | ❌ No direct built-in metric |
| Memory Usage (InnoDB Buffer Pool) | ✅ mysql\_innodb\_buffer\_pool\_pages\_dirty | ✅ SHOW ENGINE INNODB STATUS; |
| Query Execution Time | ✅ histogram\_quantile(0.95, rate(mysql\_info\_schema\_query\_response\_time\_bucket[5m])) | ✅ SELECT \* FROM performance\_schema.events\_statements\_summary\_by\_digest; |
| Max Allowed Connections | ✅ mysql\_global\_variables\_max\_connections | ✅ SHOW VARIABLES LIKE ‘max\_connections’; |
| Slow Query Log Analysis | ❌ Not available in Prometheus | ✅ SELECT \* FROM mysql.slow\_log; |
| Disk Usage for MySQL Data | ✅ node\_filesystem\_avail\_bytes{mountpoint=“/var/lib/mysql”} | ❌ No direct built-in metric |
| Replication Monitoring | ✅ mysql\_slave\_status\_seconds\_behind\_master | ✅ SHOW SLAVE STATUS; |
| Lock Waits & Deadlocks | ✅ mysql\_innodb\_row\_lock\_time\_avg | ✅ SHOW ENGINE INNODB STATUS; |
| Temp Table Usage | ✅ mysql\_global\_status\_created\_tmp\_tables | ✅ SHOW STATUS LIKE ‘Created\_tmp\_tables’; |
| Query Cache Efficiency | ❌ Deprecated in MySQL 8.0 | ✅ SHOW STATUS LIKE ‘Qcache\_hits’; (MySQL <8.0 only) |

## Prometheus vs. MySQL Built-in Performance Tools: Which is Better?

| Feature | Prometheus (mysqld\_exporter) | MySQL Built-in Tools (Performance Schema, SYS, etc.) |
| --- | --- | --- |
| Real-time Monitoring | ✅ Yes (with a 5s or 10s scrape interval) | ❌ No (only shows snapshots of the current state) |
| Historical Data Retention | ✅ Yes (stores long-term data) | ❌ No (data resets on restart) |
| Graphical Visualization | ✅ Yes (with Grafana or Prometheus UI) | ❌ No (only text-based queries) |
| Query-Level Analysis | ❌ Limited (only pre-defined metrics) | ✅ Yes (detailed query profiling with Performance Schema) |
| Alerting System | ✅ Yes (supports custom alerts) | ❌ No built-in alert system |
| Custom Metrics Support | ✅ Yes (can define custom metrics) | ❌ No (only predefined metrics) |
| Lightweight on MySQL | ✅ Yes (low overhead) | ❌ No (Performance Schema adds some overhead) |
| Complex Query Debugging | ❌ No | ✅ Yes (Performance Schema + Slow Query Log) |
| Integration with Other Tools | ✅ Yes (works with PromQL, Grafana, etc.) | ❌ No (MySQL-specific only) |

## Which One Should You Use?

### ✔️ Use Prometheus if:

* ✅ You need real-time monitoring and alerting.
* ✅ You want to track long-term performance trends.
* ✅ You need an external monitoring system that doesn’t slow down MySQL.
* ✅ You are working with DevOps tools and want easy integration.

### ✔️ Use MySQL Built-in Tools if:

* ✅ You need detailed query-level analysis.
* ✅ You want to debug slow queries and locks.
* ✅ You need deep insights into query execution plans.
* ✅ You don’t want to set up an external monitoring system.

## Final Verdict: Which One is Better?

### For General Monitoring → Prometheus is Better

* Great for real-time metrics, alerting, and long-term storage.
* Less impact on MySQL performance.

### For Deep Query Analysis → MySQL Performance Schema is Better

* Best for query optimization, deadlock detection, and detailed execution statistics.
* More useful for DBAs and developers troubleshooting slow queries.

## Recommended Approach for R&D

Since you are doing R&D, the best approach is:

1. Use Prometheus to monitor overall MySQL health (CPU, queries/sec, slow queries, memory).
2. Use MySQL’s Performance Schema when you need deep query-level insights (execution time, locks, indexing issues).
3. Use MySQL Workbench for graphical performance reports and query tuning.