📊 Ultimate Guide: Monitoring & Bottleneck Identification Without APM Tools

---

🚀 Table of Contents

---

1 Principles & Strategy

✅ No APM? No Problem. Your Monitoring Strategy Must Be:

- **Layered:** OS, Network, JVM/Process, DB, App Logs, Infra Events.
- **Time-Correlated:** All observations tied to timestamps.
- **Low-Level First:** Start from hardware → OS → process → app.
- **Proactive & Reactive:** Baseline metrics + dynamic troubleshooting.
- **Repeatable:** Build reusable shell scripts, SOPs, and log parsers.

✅ Monitoring Goals:

Goal	Approach
Identify resource bottlenecks	CPU, Memory, Disk, Network metrics
Correlate with application behavior	Logs + process stats + thread dumps
Pinpoint root cause	Drill-down via commands & logs

Validate hypothesis	Reproduce, isolate, confirm with metrics

✅ **Monitoring Triad:**

- **Metrics**: CPU, Memory, Disk, Network, Threads.
- **Logs**: Application, System, Errors, GC, SQL, Service-specific.
- **Events**: Spikes, alerts, timestamps from user-reported issues.

2️⃣ **Layer-wise Monitoring Plan**

🖥️ **Layer 1: OS & Hardware**

Metric	Tool/Command	Diagnostic Focus
CPU	top, htop, mpstat	High usage? Spikes? Core affinity?
Memory	free -m, vmstat, /proc/meminfo	Leaks? Swap thrashing?
Disk I/O	iostat, iotop, df -h, du	IOPS, latency, full disks
Network	netstat, ss, iftop, sar -n	Packet drops, errors, retransmits
Load	uptime, vmstat	Correlate with CPU/IO wait

☕ **Layer 2: JVM/Process Level (Java)**

Metric	Command	Diagnostic Focus
Thread State	jstack	Blocked threads, deadlocks, high CPU threads
Heap Usage	jmap -heap, jstat -gcutil	Memory leaks, GC issues
GC Logs	GC log parse, grep	GC pauses, frequency, throughput loss
Open Files	lsof -p <pid>	File descriptor leaks
CPU per Thread	top -H, ps -L	Hot threads, spinning loops

💾 Layer 3: Database & Storage

Metric	Command/Technique	Diagnostic Focus
Slow Queries	Log grep + timestamps	Find long-running SQLs
DB Wait Events	v$session, v$system_event (Oracle)	Lock waits, I/O waits, latch contention
Disk Latency	iostat -xd	High service times (svctm), IOPS spikes
Connection Pool	App logs, netstat, lsof	Leaks, exhaustion, timeouts

---

🌐 Layer 4: Network & Connectivity

Check	Command	Purpose
Port Reachability	telnet, nc	Check service accessibility
DNS Resolution	dig, nslookup	Resolve delays in name resolution
Packet Loss	ping, mtr, traceroute	Find hops with high latency/loss
Socket Stats	ss -s, netstat -s	TCP retransmits, backlog issues

---

📝 Layer 5: Application Logs

Source	Technique	Goal
App logs	Timestamp-aligned grep	Errors, exceptions, stack traces
GC logs	Regex parse, timestamps	GC pauses, memory trends
Thread dumps	Analyze threads over time	Find stuck threads, deadlocks
Access logs	Request correlation	TPS, response time trends, 5xx patterns

---

3 Key Tools & Commands

Layer	Tool/Command	Example
OS Metrics	top, htop, vmstat, iostat, mpstat, nmon	vmstat 1 for 1-second interval sampling
CPU Threads	top -H, ps -L	Find CPU-hogging threads
Disk I/O	iostat -x 1, iotop	Disk latency & utilization
Memory	free -m, cat /proc/meminfo	Available RAM, cache, swap usage
Network	iftop, ip -s link, ss -s	Network interface traffic
Logs	grep, awk, sed, cut, split	Parse logs for errors, patterns
Correlation	ts (moreutils), awk	Add timestamps to output for alignment

---

4 Bottleneck Patterns & Diagnostic Workflows

Symptom	Bottleneck	Clues	Diagnostic Steps
High CPU	Hot code, loop, GC	top, jstack RUNNABLE threads	Thread analysis, code hotspot
High Memory	Leak, cache bloat	free -m, jmap, jstat	Heap dump, GC logs, histo analysis
High Disk I/O	DB, logs, cache	iostat, iotop	Query logs, file system usage
Network Latency	API/DB calls	ping, traceroute, netstat -s	Packet loss, hop bottleneck
Errors Spike	App faults, DB errors	Logs + metrics	Time-correlate logs, validate root cause
Slow DB	Lock waits, slow SQL	AWR/Statspack, grep	SQL trace, session wait analysis

---

5️⃣ Correlation Techniques

✅ Align Metrics & Logs:

- Use ts command to timestamp log/command outputs:
- vmstat 1 | ts
- tail -f app.log | ts
- Cross-match with error timestamps in logs.

✅ Identify Root Cause Chain:

- Example Flow:
  - High CPU at 10:02 → jstack shows RUNNABLE threads.
  - At 10:02 in logs: Surge in API requests → bottleneck in API call.
  - DB slow at 10:02 → iostat shows disk busy 99% → storage issue.

✅ Log Parsing Patterns:

- Grep specific errors:
- grep "Exception" app.log
- Find top N slow queries:
- grep "SQL" db.log | awk '{print $NF}' | sort | uniq -c | sort -nr | head

---

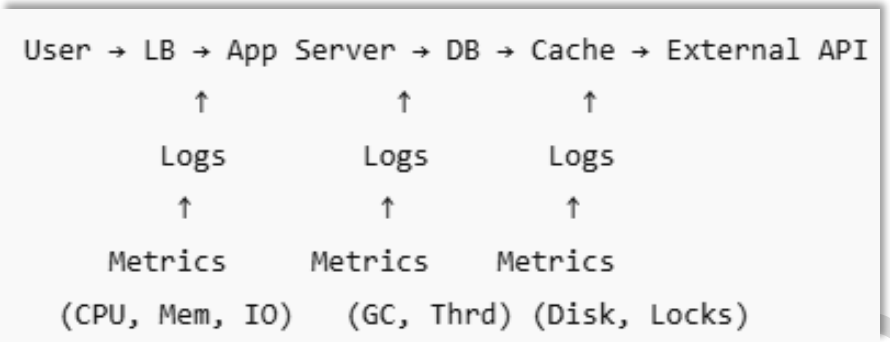
6️⃣ Real-World Diagnostic Playbook

Scenario	Steps
CPU Spike	1️⃣ top -H → 2️⃣ jstack → 3️⃣ Match threads to code → 4️⃣ Code hotspot analysis
Memory Leak	1️⃣ jmap -histo → 2️⃣ Heap dump → 3️⃣ Leak suspects (large objects)
Disk I/O Bottleneck	1️⃣ iostat -x 1 → 2️⃣ DB log analysis → 3️⃣ Query optimization
Network Latency	1️⃣ ping, traceroute → 2️⃣ ss -s → 3️⃣ App-level timeouts/logs

Application Errors	1️⃣ Log analysis → 2️⃣ Correlate with infra metrics → 3️⃣ RCA drill-down

---

7️⃣ Architecture Mapping for Bottleneck Tracing



✅ Trace **end-to-end flow** for bottleneck mapping:

- Time-based correlation.
- Layer-specific metrics + logs.
- Isolate: Which hop shows degradation first?

8️⃣ Case Study: 100% CPU Issue in Production

🕑 **10:02 AM**: Alerts fired for high CPU.
✅ top -H → One thread at 99%.
✅ jstack → Thread stuck in com.myapp.cache.Loader.load().
✅ App log at 10:02: Surge in API calls to /getData.
✅ DB log: Cache miss → Full DB fetch → Slow query.
✅ iostat: Disk IO 95% busy.
✅ Root Cause: Cache layer bug → DB fetch storm → Disk IO saturation → CPU spin.
✅ Fix: Optimize cache loader, add fallback circuit breaker, tune DB indexes.

9️⃣ Final Checklist & Best Practices

✅ **Build Monitoring Toolkit:**

- Create shell scripts for common commands (e.g., collect_sysstats.sh).

- Automate log timestamping (ts + tee).

✅ **Establish Baselines:**

- Regularly capture vmstat, iostat, jstat under normal load.

✅ **Maintain SOPs:**

- **CPU Issues**: Commands, logs, thread dump.
- **Memory Issues**: Commands, heap dump, histograms.
- **Disk Issues**: Commands, query patterns, disk metrics.
- **Network Issues**: Commands, traceroutes, socket stats.

✅ **Time Alignment is Everything**:
Always **align metrics, logs, and user reports** to **the same time window**.

✅ **Documentation**:
Keep a **diagnostic diary** of past issues, solutions, and patterns.
