1. Debugging Steps to Identify Root Cause

- **Reproduce the Issue:** Simulate peak load conditions in a test or staging environment.
- **Collect Logs:** Enable detailed logging around critical components (network, processing, I/O).
- **Check Resource Usage:** Monitor CPU, memory, disk I/O, and network utilization to identify bottlenecks.
- **Analyze Thread/Process States:** Use tools to check for thread contention, deadlocks, or excessive context switching.
- **Trace Execution:** Use distributed tracing or profiling to identify slow functions or code paths.
- **Review External Dependencies:** Verify if database, external services, or APIs are causing delays.
- **Isolate Components:** Narrow down which subsystem (network, processing, storage) causes the delays.

2. Measuring and Profiling System Performance

- **Latency and Jitter Measurement:**
    - Use **high-resolution timers** (e.g., std::chrono in C++) to measure response times.
    - Capture **histograms** or **percentiles** (P99, P99.9) for latency distribution.
    - Measure jitter as variability in response times over intervals.
- **Profiling Tools:**
    - **Linux:** perf, strace, tcpdump for system call and network tracing.
    - **Windows:** Windows Performance Recorder (WPR), Windows Performance Analyzer (WPA).
    - **Application Profilers:** Valgrind, Visual Studio Profiler, Intel VTune.
    - **Distributed Tracing:** Tools like Jaeger, Zipkin for microservices tracing.
    - **Network Analysis:** Wireshark for packet-level inspection.

- **Techniques:**
    - Instrument code with **logging and metrics** to track timing.
    - Use **profiling under load** to catch peak hour behavior.
    - Analyze **garbage collection pauses** or resource contention.