**Diagnóstico y mitigación durante pico de tráfico con alta CPU y lentitud**

**Paso a Paso para Diagnóstico:**

**A. Verificación del sistema operativo (nivel de host):**

* **top o htop**: Verificar qué procesos están consumiendo más CPU y memoria.
* **uptime**: Confirmar la carga promedio del sistema.
* **vmstat 1 5**: Observar estadísticas de memoria, procesos y CPU.
* **iostat -xz 1**: Revisar posibles cuellos de botella de disco.
* **free -h**: Validar consumo de memoria RAM y uso de SWAP.
* **dmesg**: Buscar errores recientes o activación del OOM Killer.

**B. Verificación de red:**

* **ss -tulnp o netstat -tulnp**: Verificar puertos abiertos y servicios escuchando.
* **iftop, nethogs o ip -s link**: Revisar tráfico de red en tiempo real.
* **ping, traceroute, mtr**: Comprobar latencia o pérdida hacia o desde la VM.

**C. Verificación del servicio/aplicación:**

* **journalctl -u myapp.service -f o tail -f /var/log/myapp/application.log**: Ver en tiempo real los errores reportados por la aplicación.
* **Logs del servidor web (Nginx/Apache)**: **tail -f /var/log/nginx/access.log y error.log.**
* **Validar que el proceso principal esté activo: ps aux | grep java** (o el nombre del binario de la app).
* **Consultar métricas de la aplicación** si expone endpoints de salud o métricas (/health, /metrics, Prometheus, etc.).

**Mitigación Inmediata (si es crítico):**

1. **Reiniciar el servicio** si está colapsado:

sudo systemctl restart myapp.service

1. **Aumentar recursos temporalmente (si es VM en la nube):**
   * Agregar CPUs o RAM si el entorno lo permite.
2. **Aplicar reglas de rate-limiting en Nginx o firewall si es un DoS legítimo.**
3. **Habilitar cacheo de respuestas si es posible** (en el frontend o proxy reverso).

**Causa raíz probable a investigar:**

* Aplicación no escalada para picos.
* Cuello de botella en código o base de datos.
* Ausencia de límites en la configuración del thread pool o heap de la JVM.
* Recursos de VM subdimensionados para la carga real.

**2. Herramientas o comandos para validar el estado del servicio en distintos niveles**

**Nivel de Red:**

* **ping:** Verifica conectividad IP básica.
* **traceroute o mtr:** Diagnóstico de ruta y latencia.
* **curl -I http://localhost:puerto**: Verifica respuesta HTTP local del servicio.
* **telnet IP PUERTO o nc -zv IP PUERTO**: Verifica conectividad hacia el puerto del servicio.
* **ss -ltnp**: Verifica que la app está escuchando en el puerto correcto.

**Nivel de Sistema Operativo:**

* **top, htop:** Carga del sistema y procesos.
* **df -h**: Uso del disco (espacio disponible).
* **free -h:** Uso de RAM y swap.
* **uptime:** Carga promedio.
* **systemctl status nombre-del-servicio**: Estado del servicio.
* **journalctl -xe**: Registros recientes del sistema.
* **dmesg:** Eventos del kernel (crashes, OOM, errores de hardware).

**Nivel de Aplicación:**

* **curl http://localhost:8080/health:** Endpoint de salud (si existe).
* **tail -f /var/log/myapp/application.log**: Logs en tiempo real.
* **grep "ERROR" /var/log/myapp/application.log**: Buscar errores puntuales.
* Herramientas específicas según tecnología (Java: **jstat, jmap, jstack**).