kernel tuning/troubleshooting

Kernel tuning/troubleshooting involves **adjusting operating system (OS) parameters for performance/stability (tuning) or fixing crashes/issues (troubleshooting)**, often using tools like `sysctl`, `ulimit`, `make menuconfig` (Linux), or boot options/diagnostics (Mac/Linux), focusing on memory, filesystems, I/O, and software conflicts to match workloads, requiring careful baselining and testing to avoid instability.

Kernel Tuning (Linux Focus)

Goal: Optimize performance (speed, resource use) for specific workloads (e.g., databases, web servers).

Key Areas:

fs (Filesystem): Too many open files (`ulimit -n`), I/O schedulers.

vm (Virtual Memory): Swappiness (`vm.swappiness`), dirty ratios, memory management.

Networking: TCP/IP settings, buffer sizes.

Tools: `sysctl`, `/etc/sysctl.conf`, `make menuconfig` (for compiling custom kernels), `tuned` (profiles).

Kernel Troubleshooting

Common Issues: Kernel Panic (system crash), slow performance, device failures.

Steps:

Identify Cause: Check logs (`dmesg`, `/var/log/`), review crash reports (Mac), use `journalctl` (Linux).

Isolate: Boot in Safe Mode (Mac), use a known good/rescue kernel (Linux), disconnect hardware, remove recent software/drivers.

Check Hardware: Run diagnostics (Apple Diagnostics), check RAM.

Software Conflicts: Look for recently installed or updated apps/drivers causing issues.

Best Practices

Baseline: Measure performance *before* changes.

Change Incrementally: Adjust one or a few parameters at a time.

Document: Keep records of all changes and their effects.

Test Thoroughly: Verify stability and performance after each change.

Jupyter Notebook Kernel Issues (Related but Different)

If "kernel" means the Jupyter backend: Restart, update packages (Jupyter, Anaconda), check code for errors, monitor memory.

Revision #2
Created 29 October 2025 02:43:41 by AI API
Updated 11 December 2025 16:41:09 by AI Channel