Sources

- David J. Agans - Debugging - The 9 Indispensable Rules for Finding Even the Most Elusive Hardware and Software Problem (2002, Amacom)
- **Brendan Gregg - Systems Performance_ Enterprise and the Cloud (2020, Pearson)**
- [The practice of system administration] Thomas A. Limoncelli , Strata R. Chalup , Christina J. Hogan - The Practice of Cloud System Administration_ Designing and Operating Large Distributed Systems 2 (2014, Addison

Good framework for most problems

Idea: system, narrow, part of OS (SNP)

- Logs - app, kernel, network for any errors
    - App - look for status codes/errors, retries, latency
        - In var/log
        - journalctl -u <service-name>
        - Balancing issue
            - Look forupstream troubles, latency, specific geolocation/ips failing / retries
            - Check sticky violation by seeing ua=x then seeing ua=y (ua is upstream address) too often (unless is stateless then ok. Cart no change. Otherwise somewhat sticky -> bad is > 1); diff backend_id (group) even tho same cookie
            - Latency given by rt=__ (can be due to bad routing)
            - View by:
                - tail -f /var/log/nginx/access.log
                    - Line looks like: ip - [timestamp] “GET method” 200 245_bytes_sent “browser” rt=0.027 upstream=10.0.0.3:8080 cookie=session_id=abc123 backend_id=backend_1
                - tail -f /var/log/haproxy.log
                    - Apr 18 14:02:45 haproxy[ha_p_id]: ip:port [18/Apr/2025:14:02:45.100] frontend_name~ backend_1_group/server03 0/0/2/10/12_timers 200 245_bytes_sent - - term_state 1/1/0/0/0_queue_info 0/0_retries/conn_attempts "GET /api/user HTTP/1.1"
                        - - - is capture cookies

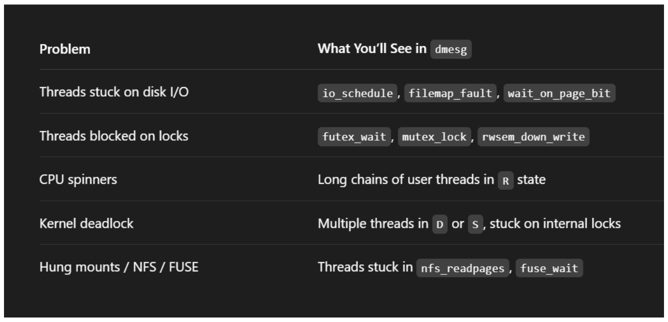
- Set option httplog in /etc/haproxy/haproxy.cfg
  - And add frontend my_frontend
  - http-request capture hdr(cookie) len 64
- journalctl -u nginx --since "10 min ago"
- Or
  - grep 502 access.log
  - awk '{print $NF}' access.log | sort | uniq -c
- Check lb algo used in /etc/nginx/nginx.conf,
  - Check upstream backend block
  - /etc/haproxy/haproxy.cfg
    - Check backend balance block
  - Supported: round robin, ip, custom hash, least conn
    - Others: most throughput, fastest response
    - Haproxy also supports: hdr (based on headers like host header), random, uri (part of url)
- Haproxy is reverse proxy/lb/tls offloader, nginx is reverse proxy/lb/api gateway

- Kernel
  - dmesg | grep -i SEARCH_TERM
    - Out of memory, driver crash, i/o error, connect/disconnect devices, kernel panic,
    - -w for live updates, -T for human readable
  - Or journalctl -k
- Network problems / request overload
  - Network check:
  - Client side on my pc: test it as client. then dig +short, ping -c 100 (avg latency over sample size), mtr -rwzbc 100 10.0.0.10 client to server. Send many to have average latency over sample size. Some drop is okay at one point (router may limit icmp) as long as if not continuous over each route
    - If drops are in your servers, check health/lb/firewall, otherwise make ticket
  - Can reach service (tcp, http request, headers): curl -I http://myapp.example.com
    - Check dns, ttfb: curl -s -o /dev/null -w "DNS: %{time_namelookup}s Connect: %{time_connect}s TTFB: %{time_starttransfer}s Total: %{time_total}s\n" http://myapp.example.com
  - If slow ttfb, http error, conn refuse, timeout/no response: On server (either end or dropping server): sudo tcpdump -i eth0 port 80
  - If is upstream: Check load balancer: tail -f /var/log/nginx/access.log / tail -f /var/log/haproxy.log

        -
        - - Check for 5xx, retries, dropped connections
    - If is server since is getting packets: ss -s to check request overload, check if app is listening ss -ltnp | grep :80, check sudo iptables -L -v -n for drop rules
- Network overload
    - Ss -s
        - shows total TCP states (e.g., estab, listen, synrecv, etc.)
        - 🔍 Look for:
            - High estab → overload
            - High recv-q → app can't keep up
        - netstat -antp | grep -i estab
            - High TIME_WAIT, SYN_RECV, or CLOSE_WAIT → slow cleanup or flood
    - Htop
        - High ksoftirqd or interrupt → CPU bottleneck from NIC
    - Dmesg | grep -i drop
        - netdev: packet dropped, skbuff, tcp_retransmit errors
    - App logs: 429, timeout, dropped
    - Fixes
        - Fixes
            - Add rate limiting or queueing
            - Tune socket buffers (sysctl net.core.rmem_max, wmem_max); increase receive / wmem (send) size
            - Use SO_REUSEPORT for parallelism - reuse ports
            - Scale horizontally or rebalance traffic
- Port conflict
    - sudo lsof -i :443
    - Sudo ss -ltnp - overall
    - Fix
        - Kill or reconfigure the conflicting process
        - Use fuser -k <port>/tcp to forcibly release
- Flapping - link goes up/down repeatedly (link is connection in network of interface w/others)
    - Bad conn basically
    - Ip link
        - Check if desired nic is up or lower up not down (lower up is not enabled but is connected). Current state
            - Lower up + up means reached and turned on. Need both
    - Dmesg | grep -i link
        - See historical data
- Tcp issues (connection). live
    - Sudo Tcpdump -i eth0 port 443 (filter packets to eth0, port)

        - Format: Timestamp Ip A > B Flags[S (syn) or S. (synack), . is ack], seq#, win_size,options, length of msg
        - Use -tt or -tttt to get timestamps with milliseconds
    - Look for retransmits, dupes, syn with no syn-ack (handshake failures), slow handshakes
    - See if request reaches server. If no traffic, is firewall/lb/dns
- dropping/routing issue
    - Ss -ltnp
    - iptables -L -v -n to check packet filter rules
        - DROP means drop if is combo of protocol, source, destination (ip); packet/byte count means how many met this rule
    - Wget - used to download files/websites (simple, recursive, retry/resumption)
        - Curl -i (include http headers) -s (silent) -o /dev/null (discard body) -w (write only status/time)   - better for interacting with APIs due to supporting more protocols/upload
            - -w useful for timing response time; –spider to test no download, –server-response to show header/response
    - Ping
        - Uses icmp echo request to get reply and see if live; get times
        - 64 bytes from 8.8.8.8: icmp_seq=1 ttl=117 time=10.2 ms; > 100 ms bad (esp for lan)
        - Packet loss, reachability
    - Traceroute
        - Look for high latency or missing hops (timeouts = dropped packet)
        - Output
            - 1 router.local (192.168.1.1) 1.01 ms
            - 2 isp.gateway.net (10.10.1.1) 10.5 ms
            - 3 * * *
            - 4 google.edge.net (8.8.8.8) 25.2 ms
        - *** means dropped; look for slow latency. Good for routing bottlenecks
    - Fixes
        - Check firewall (iptables, ufw)
        - Check backend is reachable + bound to 0.0.0.0, not 127.0.0.1
        - Reboot router/NAT or fix subnet masks
- Dns
    - Nslookup NAME - resolve domain to ip
    - Dig name +short (quick)

                - Dig dns_server_ip name   - test specific dns server
            - Ping hostname   - fails but ip succeeds -> dns issue
            - Fixes
                - Check /etc/resolv.conf
                - Use faster DNS (e.g., 1.1.1.1, 8.8.8.8)
                - Fix split DNS or DNS poisoning issues
        - Sudo Ip addr
            - Shows IPs, MACs, interface status of server (who am i)
            - Ip addr add/del   - delete/add ip/route
                - sudo ip addr add 192.168.1.100/24 dev eth0 (assign to eth0)
        - Ip route   - shows routing table (possible routes)
            - Correct default gateway
            - Proper route to subnet
        - Sudo Ip link
            - Check links in network (up/down). Live
            - Turn link on as admin - sudo ip link set eth0 up
        - Ip Replaces Ifconfig -a, route, netstat in one interface
    -

- Dashboards - let's assume none though (would just be reading charts) to make this more interesting and linux focused
- System hang? No way to call htop (doesnt load)?
    - echo t > /proc/sysrq-trigger to dump userspace state in kernel log, then dmesg
    - Shows each thread + process. processes' stack traces and states and cmd name. Likely hung due to being stuck in kernel or scheduler is blocked
    - Can see which processes are D (uninterruptible sleep, io blocked), S (sleeping; waiting on io, sleep, mutex/futex (fast userspacemutex between threads)), R (running and hogging cpu)
    - Helps with: finding futex/mutex blocks / deadlocks, cpu spinners (many r's), threads stuck in io
        - 

Problem	What You'll See in `dmesg`
Threads stuck on disk I/O	`io_schedule`, `filemap_fault`, `wait_on_page_bit`
Threads blocked on locks	`futex_wait`, `mutex_lock`, `rwsem_down_write`
CPU spinners	Long chains of user threads in `R` state
Kernel deadlock	Multiple threads in `D` or `S`, stuck on internal locks
Hung mounts / NFS / FUSE	Threads stuck in `nfs_readpages`, `fuse_wait`

- *Vmstat - overall system view. Then narrow down to process:*
- Htop - for cpu% mem% rss vsz per process
    - Do this over ps aux since gets live updates; best to see any new developments. Also can sort/search/filter, see process tree, color, see shared memory. Ps aux only shows start as unique attribute
- Pidstat - once find suspicious process, use -p PID here

        - Monitor it over time
        - This gives context switches, io rates (write cancel rates too), page faults (minor/major)
- *Now narrow down to specific part of OS that's bottlenecked (or parts): memory, io, cpu, scheduling, network, file / lock:*
- High context switching?
    - /proc/sched/debug
        - Per cpu stats (load, tasks), cfs run queue per cpu (min_vruntime, nr_spread_over (spill over), time spent running (exec_clock), task info (priority (120 is normal), start, sum_exec_runtime, vruntime, number of switches)
            - Tells you which processes are starved (how many times it's been switched in, nr_switch), which ran the most (exec_runtime), which is highest priority (vr==min vruntime), load/tasks of cpu
            - Compare vruntime to clock of cpu. If clock, or internal time in nanoseconds of how much time has pased for cfs cpu, is greater, then is not very fair. This is like expected value of cpu runtime if fair
    - Look at scheduler delay + historical view of context switches
        - sudo perf sched record -a
        - sudo perf sched latency
    - High utilization is okay as long as high throughput + no saturation
    - Fixes - change algo, or decrease threads/processes / adjust nice values (see cpu section)
- Cpu problem? Saturation or utilization?
    - Bad utilization, aka not balanced?
        - Mpstat -P ALL 1 (otherwise will just show all avg since boot)
            - %idle (most important), % usr, sys, nice, iowait, irq, softirq
                - steal, guest, gnice
        - /proc/sched/debug. Look at each cpu entry. See if tasks (nr_runnign)/load differ a lot
    - Htop, the pidstat
        - Use htop to sort/search/filter for highest cpu
        - Pidstat -uwrd that found process
            - Pidstat - context switch (involutnary, voluntary)
            - High cpu % - check if high usr or sys. High sys maybe bad driver, io (random access), interrupts
                - If high sys, check io delay, r/s, w/s
            - High involuntary - cpu saturated
                - Check vmstat r
                - Check prod/sched/debug for task q length / scheduling
    - If is usr cpu bound:
        - Perf - mem, io, syscall, sched, cpu profiling
            - Look at cpu cache; better hit ratio = faster, better locality

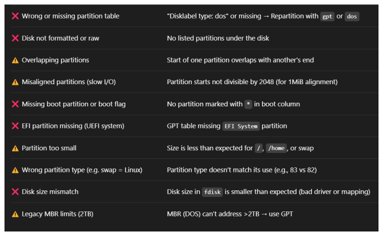
- Check cpu cache for cmd; best for optimizing cpu bound processes
  - perf stat -e cache-references,cache-misses <command>
  - perf stat -e \
    L1-dcache-loads,L1-dcache-load-misses,\
    LLC-loads,LLC-load-misses \ <command>
  - Tells cache references, cache misses; l1 l2 llc (l3). Cache miss / references = miss rate
- Fixes: do more sequential, not random, since cpus load in cache lines, which are bigger than elements. Sequential means adjacent are relevant since are next. use better data structures, like arrays not linked lists, prefetching, loop ordering (go by col, not row)

* Process hotspots
  - Sudo perf top - quick, live, no stack traces unlike record/report)
  - Sudo perf report - gives you snapsbhot tui breakdown of hot functions, inclusive and exclusive cpu time, strack traces (with -g). Text version of flamegraph
  - Sudo perf record -g -p PID -F 99
    - Need one off frequency to not miss things at end of 100 (offset changes start/end every time)
    - Can make flamegraph with this; snapshot

- Bpftrace - true tracing / scripting for tracing; can trace any func + perf
  - Lower overhead compared to perf, strace. Can also add to any user/kernel func, not just syscalls. Stronger filtering + debugging. Attach on fly.
  - Use kprobe for runtime debug logging, tracepoint for compile debug logging. Logging leads to aggregation + counting stats. Uprobe for userspace
  - Can be used to trace syscalls (args, syscall name, return value, time), see what files are being opened, trace fucntions, sched. Can make latency histograms
    - Trace syscalls: sudo bpftrace -e 'tracepoint:syscalls:sys_enter_* { @[probe] = count(); }'
    - Trace functions - sudo bpftrace -e 'kprobe:vfs_read { printf("vfs_read called by %s\n", comm); }'

- Single thread -> multithread
  - High cpu in one core
- Busy loop/polling -> add sleep/yield
  - Strace shows no syscalls but high cpu
- Starvation - bad cgroup cpu weight (increase it)
- High saturation

      - Too many threads -> set limit (sysctl, etc.)
      - High cswch/s, all cores high cpu but low throughput. High r in vmstat. High load
        - Add cpu pins, use taskset to rebalance threads, numactl to fix numa balance by assigning thread + mem to same node (task is in cpu with mem it needs instead of remote access), remove cpu pins, tune irq affinity for cpus
        - break up cpu hog into smaller processes, reduce number of threads, fix balance
        - Tune priorities (nice)
    - Lock contention -> reduce critical section size, use smaller-scoped locks (finer grain). Switch to mutex from spinlock. Avoid holding locks across io
      - Perf top shows futex_wait, threads stuck in d/s, high context switch rate
    - Thundering herd (cpu usage come in bursts) -> Rate-limit incoming requests, Introduce jitter/backoff
    - misc
      - Too much garbage collection -> increase heap size, change gc type
        - High %usr, %sys, low throughput
      - High %steal -> move vm to another host
  - Edge cases
    - Bad cpu pinning, numa/core imbalance, bad scheduler, locks, cpu herds, too many threads, starvation (bad nice or cgroup cpu weight)
- Process halt?
  - Strace
    - Use -T to see timing of each syscall
    - See what functions are hung on
  - Perf trace is preferred; doesn't interrupt each syscall so lower overhead. Has same filtering, timing. Also has function info with -g to give syscall context
- Mem problem? Saturation or utilization?
  - Utilization
    - Free -h - can see how much used, how much avail, how much free, cache, buffer, swap
    - Vmstat -s or proc/meminfo for breakdown of active vs inactive, slab vs not, reclaim vs not
  - Saturation
    - Kernel logs -> dmesg | grep -i “out of memory”
    - Vmstat -s - si, so
  - Fixes
    - Utilization
      - Restart memory hog and apply cgroup limit / fix it
        - Set different stack/rss sizes, lifetime of objects
      - Clear caches (esp slab)

            - Valgrind –leak-check / strace (look for sbrk/brk mmap but no munmap brk/sbrk to shrink)
                - Fix in program code
            - Tune reclamation (slab_reclaimation, etc.)
        - Saturation
            - Adjust swapiness, oom killer scores
            - Reduce utilization (see above)
- Io problem? Saturation or utilization? Or disk?
    - Cache issue?
        - Check page cache - requires bpc
            - Sudo cachestat
                - hit%
                    - dirty, drop (evict), cached_mb, buffer_mb
    - Utilization?
        - Df -h
            - see what filesystem file that process is using uses
                - df -T /path/to/file
            - Check if disk for file system that process is on is out of space
        - Fdisk -l   - fix partition issues like overlapping partitions, misaligned partition (not divisible by 2048 bytes), missing
            - Check partitioning issues: see device, start/end, sectors, size/type
                - Check that no overlaps and device size is expected
                - Check if have efi system partition if linux wont' boot
                - Disk underutilized -> check partitions (big enough)
                - New disk not visible -> check if is partitioned
                - 

✗ Wrong or missing partition table	"Disklabel type: dos" or missing → Repartition with `gpt` or `dos`
✗ Disk not formatted or raw	No listed partitions under the disk
⚠ Overlapping partitions	Start of one partition overlaps with another's end
⚠ Misaligned partitions (slow I/O)	Partition starts not divisible by 2048 (for 1MiB alignment)
✗ Missing boot partition or boot flag	No partition marked with `*` in boot column
✗ EFI partition missing (UEFI system)	GPT table missing `EFI System` partition
⚠ Partition too small	Size is less than expected for `/`, `/home`, or swap
⚠ Wrong partition type (e.g. swap = Linux)	Partition type doesn't match its use (e.g., 83 vs 82)
✗ Disk size mismatch	Disk size in `fdisk` is smaller than expected (bad driver or mapping)
⚠ Legacy MBR limits (2TB)	MBR (DOS) can't address >2TB → use GPT

    - Saturation?
        - Iostat -x
            - Queue length
            - r/w rates
            - r/w rqm/s (read/write merged per sec)
            - r/w_await - avg wait for _ request, queue + service time
            - Aqu-sz - average size of queue
            - r/w req-sz - average r/w request size
            - f/s - flushes per second. F_await - time spent waiting for flush in ms
            - %util - % of time device spent servicing. Svctm in older version is service time
            -

        - Fixes
            - Utilization
                - Delete files (see what's taking up with du -sh. See disk health with df -h)
                - Lower dirty page ratio
                    - `vm.dirty_ratio=10`, `dirty_background_ratio=5`
                - Fix batching + Adjust fsync rate
                    - Fix access patterns - seq instead of random
                    - Align reads
            - Saturation
                - Tune io scheduler / throttle disk users
                - Increase ram / prevent swap by decreasing swappiness
                - Defrag / spread load across disks better
- X preventing file system unmount
    - See what filesystem file that process is using uses
        - df -T /path/to/file
    - Fix:
        - See what files open in mount - lsof +D /mount/point
        - See what processes holding it lsof -p PID file
        - Kill process
        - Exit mount with terminal (so pwd doesn't hold it up)
        - Lazy unmount `umount -l /mount/point`
        - `Force:   umount -f /mount/point (careful!)`
- File or lock problem? Contention, permissions, bad file?
    - Directory / file size problem?
        - du -sh - shows size of each file + dir (to see what's taking up space)
            - S is summary, h is human readable
    - Lock contention?
        - lsof file will show those with file open but not those who hold lock
        - If want to see what files pid is holding, do lsof -p PID
        - For locks see /procs/lock
            - Grep filename_inode /proc/lock (see if is read/write/advisory (not enforced))
                - Find inode with stat file
    - File permissions, last accecssed?
        - Stat FILE
            - Uid, gid, file type, device: 820h/2080d (device driver, instance of device driver)
            - Access perms (chmod stuff): numeric/chars in user|group|others
                - First - indicates it is regular file, not dir, link, etc. (otherwise would be d or l). Rwx
                - Last is sticky bit (can create but can't delete/rename others' esp in tmp unless they are owner (dir, file) or root)

        - Size, links, blocks, inode, io block (size of read/writes)
        - access, modify, last time changed metadata, birth
            - Has timezone offset, nanosecond precision, date

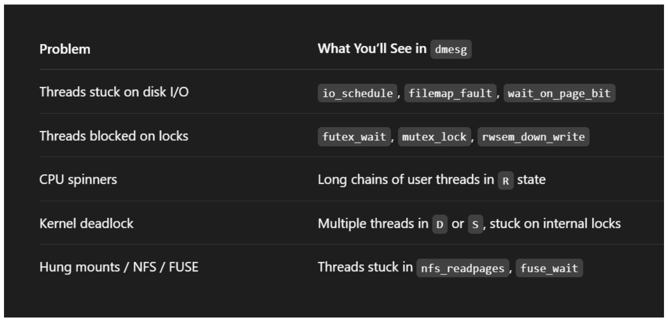
General framework based on Section

- Shortened network ver
- Client-side (on my PC) – check as client first:
-
- 1. DNS Resolution
-    └─ dig +short myapp.example.com
-
- 2. Network Latency and Routing Health
-    ├─ ping -c 100 myapp.example.com
-    │   └─ Avg latency over sample size using ICMP
-    └─ mtr -rwzbc 100 myapp.example.com
-        ├─ Route to server, drop/latency per hop
-        ├─ Some drop is OK (router may deprioritize ICMP)
-        └─ If consistent loss across hops or at destination:
-            └─ Check infra or escalate (ticket)
-
- 3. Application Layer Reachability
-    └─ curl -I http://myapp.example.com
-        ├─ TCP + HTTP headers
-        └─ See if server is responding, what status (e.g. 200, 503)
-
- 4. Measure Response Timings
-    └─ curl -s -o /dev/null -w "DNS: %{time_namelookup}s Connect: %{time_connect}s TTFB: %{time_starttransfer}s Total: %{time_total}s\n" http://myapp.example.com
-        ├─ -s: silent (no progress bar)
-        ├─ -o /dev/null: discard body
-        ├─ -w: write custom timing info
-        └─ -i (optional): include response headers
-
-        □ If TTFB is slow, timeout, connection refused, or HTTP error:
-
-            5. Check if server is even receiving packets
-                └─ sudo tcpdump -i eth0 port 80
-
-            6. If it’s upstream (load balancer, not app):
-                └─ tail -f /var/log/nginx/access.log

```
•           └─ tail -f /var/log/haproxy.log
•              └─ Look for 5xx errors, retries, dropped conns
•
•        7. If server **is** receiving packets:
•           ├─ ss -s
•           │   └─ Check for request overload
•           ├─ ss -ltnp | grep :80
•           │   └─ Is app listening on port 80?
•           └─ sudo iptables -L -v -n
•              └─ Any drop rules interfering?
•
•
```

- ## Logs - app, kernel, network for any errors
    - App - look for status codes/errors, retries, latency
        - In var/log
        - journalctl -u <service-name>
    - Kernel
        - dmesg | grep -i SEARCH_TERM
            - Out of memory, driver crash, i/o error, connect/disconnect devices, kernel panic,
            - -w for live updates, -T for human readable
        - Or journalctl -k
- ## Network
- DPMC2|TSI|HA
    - Client side on my pc: test it as client. then dig NAME +short, ping -c 100 (avg latency over sample size using icmp), mtr -rwbzc 100 10.0.0.10 client to server. Send many to have average latency over sample size. Some drop is okay at one point (router may limit icmp) as long as if not continuous over each route
        - If drops are in your servers, check health/lb/firewall, otherwise make ticket
    - Can reach service (application layer, tcp, http request, headers): curl -I http://myapp.example.com
        - Check dns, ttfb, total time: curl -s -o /dev/null -w "DNS: %{time_namelookup}s Connect: %{time_connect}s TTFB: %{time_starttransfer}s Total: %{time_total}s\n" http://myapp.example.com
            - -s is silent, -o is write response to null (discrad), -w is write custom output, -i include response header like content type, etc.
            - Ttfb is time til first byte
    - If slow dns, fix by: choosing better dns providers in /etc/resolv.conf, flush dns (sudo systemd-resolve --flush-caches)
    - If slow ttfb, http error, conn refuse, timeout/no response: On server (either end or dropping server): sudo tcpdump -i eth0 port 80
        - Look for dupes, no syn-synack-ack, rst, no http request, delays

    - If is upstream: Check load balancer: tail -f /var/log/nginx/access.log / tail -f /var/log/haproxy.log
        -
        - - Check for 5xx, retries, dropped connections
    - If is server since is getting packets: ss -s to check request overload, check if app is listening ss -ltnp | grep :80, check sudo iptables -L -v -n for drop rules
- Other network
    - Port conflict
        - sudo lsof -i :443
        - Sudo ss -ltnp - overall
        - Fix
            - Kill or reconfigure the conflicting process
            - Use fuser -k <port>/tcp to forcibly release
    - Flapping - link goes up/down repeatedly (link is connection in network of interface w/others)
        - Bad conn basically
        - Ip link
            - Check if desired nic is up or lower up not down (lower up is not enabled but is connected). Current state
                - Lower up + up means reached and turned on. Need both
        - Dmesg | grep -i link
            - See historical data
        - Fixes:
            - Pin stable route:
                - ip route add <destination_ip> via <stable_gateway_ip> dev eth0
            - For DNS-based changes, set lower TTLs and validate authoritative DNS
            - If have control, set bgp flap
    - Dns - dig x +short fails
        - Fixes
            - Check /etc/resolv.conf
            - Use faster DNS (e.g., 1.1.1.1, 8.8.8.8)
            - Fix split DNS or DNS poisoning issues
    - Fixes for bad routing (loops, drops, jump locs, high latency, a->b good but b->a bad)
        - Check firewall (iptables, ufw)
        - Check backend is reachable + bound to 0.0.0.0, not 127.0.0.1
        - Reboot router/NAT or fix subnet masks

- Dashboards - let’s assume none though (would just be reading charts) to make this more interesting and linux focused
- System hang? No way to call htop (doesnt load)?
    - echo t > /proc/sysrq-trigger to dump userspace state in kernel log, then dmesg
    - Shows each thread + process. processes’ stack traces and states and cmd name. Likely hung due to being stuck in kernel or scheduler is blocked
    - Can see which processes are D (uninterruptible sleep, io blocked), S (sleeping; waiting on io, sleep, mutex/futex (fast userspacemutex between threads)), R (running and hogging cpu)
    - Helps with: finding futex/mutex blocks / deadlocks, cpu spinners (many r’s), threads stuck in io
        - 

Problem	What You'll See in `dmesg`
Threads stuck on disk I/O	`io_schedule`, `filemap_fault`, `wait_on_page_bit`
Threads blocked on locks	`futex_wait`, `mutex_lock`, `rwsem_down_write`
CPU spinners	Long chains of user threads in `R` state
Kernel deadlock	Multiple threads in `D` or `S`, stuck on internal locks
Hung mounts / NFS / FUSE	Threads stuck in `nfs_readpages`, `fuse_wait`

- Vmstat - overall system view. Then narrow down to process:
- Htop - for cpu% mem% rss vsz per process
    - Do this over ps aux since gets live updates; best to see any new developments. Also can sort/search/filter, see process tree, color, see shared memory. Ps aux only shows start as unique attribute
- Pidstat - once find suspicious process, use -p PID here
    - Monitor it over time
    - This gives context switches, io rates (write cancel rates too), page faults (minor/major)
- Now narrow down to specific part of OS that’s bottlenecked (or parts): memory, io, cpu, scheduling, network, file / lock:
- High context switching?
    - /proc/sched/debug
        - Per cpu stats (load, tasks), cfs run queue per cpu (min_vruntime, nr_spread_over (spill over), time spent running (exec_clock), task info (priority (120 is normal), start, sum_exec_runtime, vruntime, number of switches)
            - Tells you which processes are starved (how many times it’s been switched in, nr_switch), which ran the most (exec_runtime), which is highest priority (vr==min vruntime), load/tasks of cpu

            - Compare vruntime to clock of cpu. If clock, or internal time in nanoseconds of how much time has pased for cfs cpu, is greater, then is not very fair. This is like expected value of cpu runtime if fair
    - Look at scheduler delay + historical view of context switches
        - sudo perf sched record -a
        - sudo perf sched latency
    - High utilization is okay as long as high throughput + no saturation
    - Fixes - change algo, or decrease threads/processes / adjust nice values (see cpu section)
- Cpu problem? Saturation or utilization?
    - Bad utilization, aka not balanced?
        - Mpstat -P ALL 1 (otherwise will just show all avg since boot)
            - %idle (most important), % usr, sys, nice, iowait, irq, softirq
                - steal, guest, gnice
        - /proc/sched/debug. Look at each cpu entry. See if tasks (nr_runnign)/load differ a lot
    - Htop, the pidstat
        - Use htop to sort/search/filter for highest cpu
        - Pidstat -uwrd that found process
            - Pidstat - context switch (involutnary, voluntary)
            - High cpu % - check if high usr or sys. High sys maybe bad driver, io (random access), interrupts
                - If high sys, check io delay, r/s, w/s
            - High involuntary - cpu saturated
                - Check vmstat r
                - Check prod/sched/debug for task q length / scheduling
    - If is usr cpu bound:
        - Perf - mem, io, syscall, sched, cpu profiling
            - Look at cpu cache; better hit ratio = faster, better locality
                - Check cpu cache for cmd; best for optimizing cpu bound processes
                    - perf stat -e cache-references,cache-misses <command>
                    - perf stat -e \ L1-dcache-loads,L1-dcache-load-misses,\ LLC-loads,LLC-load-misses \ <command>
                    - Tells cache references, cache misses; l1 l2 llc (l3). Cache miss / references = miss rate
                - Fixes: do more sequential, not random, since cpus load in cache lines, which are bigger than elements. Sequential means adjacent are relevant since are next. use better data structures, like arrays not linked lists, prefetching, loop ordering (go by col, not row)

    - Process hotspots
      - Sudo perf top - quick, live, no stack traces unlike record/report)
      - Sudo perf report - gives you snapsbhot tui breakdown of hot functions, inclusive and exclusive cpu time, strack traces (with -g). Text version of flamegraph
      - Sudo perf record -g -p PID -F 99
        - Need one off frequency to not miss things at end of 100 (offset changes start/end every time)
        - Can make flamegraph with this; snapshot
  - Bpftrace - true tracing / scripting for tracing; can trace any func + perf
    - Lower overhead compared to perf, strace. Can also add to any user/kernel func, not just syscalls. Stronger filtering + debugging. Attach on fly.
    - Use kprobe for runtime debug logging, tracepoint for compile debug logging. Logging leads to aggregation + counting stats. Uprobe for userspace
    - Can be used to trace syscalls (args, syscall name, return value, time), see what files are being opened, trace fucntions, sched. Can make latency histograms
      - Trace syscalls: sudo bpftrace -e 'tracepoint:syscalls:sys_enter_* { @[probe] = count(); }'
      - Trace functions - sudo bpftrace -e 'kprobe:vfs_read { printf("vfs_read called by %s\n", comm); }'
- Single thread -> multithread
  - High cpu in one core
- Busy loop/polling -> add sleep/yield
  - Strace shows no syscalls but high cpu
- Starvation - bad cgroup cpu weight (increase it)
- High saturation
    - Too many threads -> set limit (sysctl, etc.)
    - High cswch/s, all cores high cpu but low throughput. High r in vmstat. High load
      - Add cpu pins, use taskset to rebalance threads, numactl to fix numa balance by assigning thread + mem to same node (task is in cpu with mem it needs instead of remote access), remove cpu pins, tune irq affinity for cpus
      - break up cpu hog into smaller processes, reduce number of threads, fix balance
      - Tune priorities (nice)
  - Lock contention -> reduce critical section size, use smaller-scoped locks (finer grain). Switch to mutex from spinlock. Avoid holding locks across io
    - Perf top shows futex_wait, threads stuck in d/s, high context switch rate

            - Thundering herd (cpu usage come in bursts) -> Rate-limit incoming requests, Introduce jitter/backoff
            - misc
                - Too much garbage collection -> increase heap size, change gc type
                    - High %usr, %sys, low throughput
                - High %steal -> move vm to another host
    - Edge cases
        - Bad cpu pinning, numa/core imbalance, bad scheduler, locks, cpu herds, too many threads, starvation (bad nice or cgroup cpu weight)
- Process halt?
    - Strace
        - Use -T to see timing of each syscall
        - See what functions are hung on
    - Perf trace is preferred; doesn't interrupt each syscall so lower overhead. Has same filtering, timing. Also has function info with -g to give syscall context
- Mem problem? Saturation or utilization?
    - Utilization
        - Free -h - can see how much used, how much avail, how much free, cache, buffer, swap
        - Vmstat -s or proc/meminfo for breakdown of active vs inactive, slab vs not, reclaim vs not
    - Saturation
        - Kernel logs -> dmesg | grep -i "out of memory"
        - Vmstat -s - si, so
    - Fixes
        - Utilization
            - Restart memory hog and apply cgroup limit / fix it
                - Set different stack/rss sizes, lifetime of objects
            - Clear caches (esp slab)
            - Valgrind –leak-check / strace (look for sbrk/brk mmap but no munmap brk/sbrk to shrink)
                - Fix in program code
            - Tune reclamation (slab_reclaimation, etc.)
        - Saturation
            - Adjust swapiness, oom killer scores
            - Reduce utilization (see above)
- Io problem? Saturation or utilization? Or disk?
    - Cache issue?
        - Check page cache - requires bpc
            - Sudo cachestat
                - hit%
                    - dirty, drop (evict), cached_mb, buffer_mb

  - Utilization?
    - Df -h
      - see what filesystem file that process is using uses
        - df -T /path/to/file
      - Check if disk for file system that process is on is out of space
    - Fdisk -l - fix partition issues like overlapping partitions, misaligned partition (not divisible by 2048 bytes), missing
      - Check partitioning issues: see device, start/end, sectors, size/type
        - Check that no overlaps and device size is expected
        - Check if have efi system partition if linux wont' boot
        - Disk underutilized -> check partitions (big enough)
        - New disk not visible -> check if is partitioned
  - Saturation?
    - Iostat -x
      - Queue length
      - r/w rates
      - r/w rqm/s (read/write merged per sec)
      - r/w_await - avg wait for _ request, queue + service time
      - Aqu-sz - average size of queue
      - r/w req-sz - average r/w request size
      - f/s - flushes per second. F_await - time spent waiting for flush in ms
      - %util - % of time device spent servicing. Svctm in older version is service time
      -
    - Fixes
      - Utilization
        - Delete files (see what's taking up with du -sh. See disk health with df -h)
        - Lower dirty page ratio
          - `vm.dirty_ratio=10`, `dirty_background_ratio=5`
        - Fix batching + Adjust fsync rate
          - Fix access patterns - seq instead of random
          - Align reads
      - Saturation
        - Tune io scheduler / throttle disk users
        - Increase ram / prevent swap by decreasing swappiness
        - Defrag / spread load across disks better
- X preventing file system unmount
  - See what filesystem file that process is using uses
    - df -T /path/to/file
  - Fix:
    - See what files open in mount - lsof +D /mount/point

        - See what processes holding it lsof -p PID file
        - Kill process
        - Exit mount with terminal (so pwd doesn't hold it up)
        - Lazy unmount `umount -l /mount/point`
        - `Force:   umount -f /mount/point (careful!)`
- File or lock problem? Contention, permissions, bad file?
    - Directory / file size problem?
        - du -sh - shows size of each file + dir (to see what's taking up space)
            - S is summary, h is human readable
    - Lock contention?
        - lsof file will show those with file open but not those who hold lock
        - If want to see what files pid is holding, do lsof -p PID
        - For locks see /procs/lock
            - Grep filename_inode /proc/lock (see if is read/write/advisory (not enforced))
                - Find inode with stat file
    - File permissions, last accecssed?
        - Stat FILE
            - Uid, gid, file type, device: 820h/2080d (device driver, instance of device driver)
            - Access perms (chmod stuff): numeric/chars in user|group|others
                - First - indicates it is regular file, not dir, link, etc. (otherwise would be d or l). Rwx
                - Last is sticky bit (can create but can't delete/rename others' esp in tmp unless they are owner (dir, file) or root)
            - Size, links, blocks, inode, io block (size of read/writes)
            - access, modify, last time changed metadata, birth
                - Has timezone offset, nanosecond precision, date
    -
-

●