4.10 Ressourcenengpässe erkennen

Ein Dienst kann korrekt konfiguriert sein und trotzdem langsam, instabil oder nicht erreichbar werden, wenn ihm oder dem Gesamtsystem benötigte Ressourcen fehlen.

> **Grundsatz:**
> Ein einzelner hoher Messwert beweist noch keinen Engpass. Entscheidend sind Messdauer, Vergleichswerte, Auslastungsursache und der zeitliche Zusammenhang mit der Störung.

Ziele dieser Seite

Nach dieser Seite sollst du:

- CPU-, Arbeitsspeicher- und I/O-Engpässe erkennen können,
- Lastspitzen von dauerhaftem Ressourcenmangel unterscheiden können,
- System- und Prozesswerte getrennt untersuchen können,
- physischen, virtuellen und residenten Speicher unterscheiden können,
- Paging und Swapping richtig bewerten können,
- Datenträgerlatenz und Warteschlangen berücksichtigen können,
- Handle-, Thread-, Prozess- und Dateideskriptorgrenzen prüfen können,
- Ressourcenwerte über mehrere Messpunkte beobachten können,
- Betriebssystemprotokolle nach Ressourcenfehlern durchsuchen können,
- Ergebnisse mit dem Zeitpunkt einer Dienststörung verbinden können.

---

1. Welche Ressourcen können einen Dienst begrenzen?

Ressource	Typisches Fehlerbild
CPU	Hohe Antwortzeiten, Zeitüberschreitungen, verzögerte Verarbeitung
Arbeitsspeicher	Paging, Prozessabbrüche, fehlgeschlagene Allokationen
Swap beziehungsweise Auslagerungsdatei	Hohe Latenz durch Speicherauslagerung
Datenträger-I/O	Lange Wartezeiten, blockierte Prozesse, langsame Datenbank
Speicherplatz	Schreiben, Starten oder Protokollieren schlägt fehl
Inodes	Linux kann trotz freiem Speicherplatz keine Dateien anlegen

Ressource	Typisches Fehlerbild
Dateideskriptoren	Neue Dateien oder Sockets können nicht geöffnet werden
Windows-Handles	Zugriff auf Dateien, Registry, Events oder Prozesse schlägt fehl
Threads	Neue Aufgaben oder Verbindungen können nicht verarbeitet werden
Prozesslimit	Neue Prozesse können nicht erzeugt werden
Netzwerkbandbreite	Langsame Übertragung und Warteschlangen
Verbindungspool	Neue Datenbank- oder HTTP-Verbindungen warten
Portbereich	Neue ausgehende TCP-Verbindungen schlagen fehl
GPU oder Beschleuniger	Spezialisierte Berechnung kann nicht ausgeführt werden
Quota	Benutzer oder Dienst darf keine weiteren Daten speichern
Containerlimit	Dienst erreicht CPU-, RAM- oder Prozessgrenze des Containers

2. Auslastung und Engpass unterscheiden

Zustand	Beschreibung
Auslastung	Eine Ressource wird verwendet
Hohe Auslastung	Ressource wird stark verwendet, Dienst funktioniert aber noch
Sättigung	Neue Arbeit muss warten
Engpass	Ressourcenbegrenzung beeinträchtigt die benötigte Funktion
Überlast	Eingehende Arbeit übersteigt dauerhaft die Verarbeitungskapazität
Ressourcenleck	Verbrauch steigt, obwohl abgeschlossene Arbeit Ressourcen freigeben sollte
Grenzwertfehler	Konfiguriertes Limit wird vor der technischen Kapazität erreicht

Beispiel:

```
CPU-Auslastung: 95 %
Antwortzeit:     normal
Warteschlange:   niedrig
Fehler:          keine
```

Das ist hohe Auslastung, aber noch kein bewiesener Engpass.

```
CPU-Auslastung: 95 %
Antwortzeit:     stark erhöht
Warteschlange:   wächst
Timeouts:        vorhanden
```

Hier besteht ein belastbarer Hinweis auf CPU-Sättigung.

---

3. Immer über mehrere Messpunkte prüfen

Eine Momentaufnahme kann zufällig während einer kurzen Lastspitze entstehen.

Empfohlene Messstruktur:

```
Zeitpunkt
CPU gesamt
CPU des Dienstprozesses
verfügbarer Arbeitsspeicher
Swap- beziehungsweise Paging-Aktivität
Datenträger-I/O und Wartezeit
Anzahl aktiver Anfragen
Anwendungsantwortzeit
Fehlerrate
```

Sinnvolle Vergleichszeiträume:

- unmittelbar vor der Störung,
- während der Störung,
- unmittelbar danach,
- normaler Vergleichszeitraum,
- gleiche Uhrzeit an einem störungsfreien Tag,

- Zustand vor und nach einer Maßnahme.

> Ein Messwerkzeug erzeugt selbst Last. Abtastrate, Anzahl der Messwerte und Detailgrad müssen zum System passen.

---

4. Schnellübersicht für Windows, Linux und macOS

Aufgabe	Windows	Linux	macOS
Systemlaufzeit	`[RO] (Get-Date) - (Get-CimInstance Win32_OperatingSystem).LastBootUpTime`	`[RO] uptime`	`[RO] uptime`
Logische Prozessoren	`[RO] (Get-CimInstance Win32_ComputerSystem).NumberOfLogicalProcessors`	`[RO] nproc`	`[RO] sysctl -n hw.logicalcpu`
Prozessübersicht	`[RO] Get-Process`	`[RO] top`	`[RO] top`
Speicherübersicht	`[RO] Get-CimInstance Win32_OperatingSystem`	`[RO] free -h`	`[RO] memory_pressure`
Swap	`[RO] Get-CimInstance Win32_PageFileUsage`	`[RO] swapon --show`	`[RO] sysctl vm.swapusage`
Datenträgerübersicht	`[RO] Get-Volume`	`[RO] iostat`	`[RO] iostat`
Prozesslimits	Betriebssystemspezifisch	`[RO][FILE] cat /proc/<PID>/limits`	`[RO] launchctl limit`
Systemprotokolle	`[RO] Get-WinEvent`	`[RO][PRIV] sudo journalctl`	`[RO][PRIV] sudo log show`

---

5. CPU-Grunddaten unter Windows prüfen

Anzahl logischer Prozessoren anzeigen:

```
[RO] Get-CimInstance Win32_ComputerSystem |
    Select-Object NumberOfProcessors,
                  NumberOfLogicalProcessors
```

Gesamte CPU-Auslastung als Momentaufnahme:

```
[RO] Get-CimInstance `
    Win32_PerfFormattedData_PerfOS_Processor `
    -Filter "Name='_Total'" |
    Select-Object Name,
                  PercentProcessorTime,
                  PercentUserTime,
                  PercentPrivilegedTime,
                  PercentIdleTime
```

Mehrere Messpunkte mit Get-Counter erfassen:

```
[RO] Get-Counter `
    -Counter '\Processor(_Total)\% Processor Time' `
    -SampleInterval 2 `
    -MaxSamples 5
```

> Die Namen klassischer Windows-Leistungsindikatoren können auf lokalisierten Windows-Systemen übersetzt sein. Falls ein Counterpfad nicht gefunden wird, müssen die auf diesem System vorhandenen Zählersätze ermittelt werden.

Verfügbare Zählersätze suchen:

```
[RO] Get-Counter -ListSet * |
    Select-Object CounterSetName |
    Sort-Object CounterSetName
```

---

6. CPU-verbrauchende Windows-Prozesse ermitteln

Prozesse nach kumulierter CPU-Zeit sortieren:

```
[RO] Get-Process |
    Sort-Object CPU -Descending |
    Select-Object -First 15 Id,
```

```
                ProcessName,
                CPU,
                StartTime
```

Die Eigenschaft `CPU` enthält die gesamte bisher verbrauchte Prozessorzeit des Prozesses in Sekunden. Sie ist kein momentaner Prozentwert.

Momentane Prozessauslastung über formatierte Leistungsdaten:

```
[RO] Get-CimInstance Win32_PerfFormattedData_PerfProc_Process |
    Where-Object Name -NotIn "_Total", "Idle" |
    Sort-Object PercentProcessorTime -Descending |
    Select-Object -First 15 Name,
                    IDProcess,
                    PercentProcessorTime,
                    ThreadCount,
                    WorkingSetPrivate
```

Bestimmte PID untersuchen:

```
[RO] Get-Process -Id <PID> |
    Select-Object Id,
                  ProcessName,
                  StartTime,
                  CPU,
                  Threads,
                  HandleCount
```

> Die Prozess-CPU-Ausgabe kann bei mehreren logischen Prozessoren anders skaliert wirken als die gesamte Systemauslastung. Werte müssen im Kontext des verwendeten Messwerkzeugs interpretiert werden.

---

7. Arbeitsspeicher unter Windows prüfen

Gesamten und freien physischen Speicher anzeigen:

```
[RO] Get-CimInstance Win32_OperatingSystem |
    Select-Object @{
        Name="TotalRAM_GiB"
        Expression={[math]::Round($_.TotalVisibleMemorySize / 1MB, 2)}
    }, @{
        Name="FreeRAM_GiB"
        Expression={[math]::Round($_.FreePhysicalMemory / 1MB, 2)}
    }, @{
        Name="FreeVirtualMemory_GiB"
        Expression={[math]::Round($_.FreeVirtualMemory / 1MB, 2)}
    }
```

Die CIM-Werte werden hier von KiB in GiB umgerechnet.

Speicherindikatoren als Momentaufnahme:

```
[RO] Get-CimInstance Win32_PerfFormattedData_PerfOS_Memory |
    Select-Object AvailableMBytes,
                  PercentCommittedBytesInUse,
                  PagesPerSec,
                  PageReadsPerSec,
                  PageWritesPerSec,
                  PoolPagedBytes,
                  PoolNonpagedBytes
```

Mehrere Messpunkte des verfügbaren Speichers:

```
[RO] Get-Counter `
    -Counter '\Memory\Available MBytes' `
    -SampleInterval 2 `
    -MaxSamples 5
```

> Ein niedriger Wert bei „freiem“ Speicher ist nicht automatisch ein Fehler, weil Betriebssysteme ungenutzten Speicher als Cache verwenden. Verfügbarer Speicher, Paging-Aktivität und Anwendungsfunktion müssen gemeinsam betrachtet werden.

8. Speicherverbrauch von Windows-Prozessen prüfen

Nach Working Set sortieren:

```
[RO] Get-Process |
    Sort-Object WorkingSet64 -Descending |
    Select-Object -First 15 Id,
                        ProcessName,
                        @{
                            Name="WorkingSet_MiB"
                            Expression={[math]::Round($_.WorkingSet64 / 1MB, 2)}
                        },
                        @{
                            Name="PrivateMemory_MiB"
                            Expression={[math]::Round($_.PrivateMemorySize64 / 1MB, 2)}
                        }
```

Bestimmten Prozess untersuchen:

```
[RO] Get-Process -Id <PID> |
    Select-Object Id,
                ProcessName,
                WorkingSet64,
                PrivateMemorySize64,
                VirtualMemorySize64,
                PagedMemorySize64,
                PeakWorkingSet64
```

Wert	Bedeutung
`WorkingSet64`	Derzeit im physischen Speicher befindliche Seiten
`PrivateMemorySize64`	Privat zugeordneter Speicher
`VirtualMemorySize64`	Virtueller Adressraum
`PagedMemorySize64`	Auslagerungsfähiger Speicher
`PeakWorkingSet64`	Bisheriger Höchstwert des Working Sets

> Virtueller Speicher darf nicht direkt mit physisch belegtem RAM gleichgesetzt werden.

---

9. Windows-Auslagerungsdatei prüfen

Nutzung der Auslagerungsdatei anzeigen:

```
[RO] Get-CimInstance Win32_PageFileUsage |
    Select-Object Name,
                  AllocatedBaseSize,
                  CurrentUsage,
                  PeakUsage
```

Die Größen werden üblicherweise in MiB angegeben.

Konfiguration der Auslagerungsdatei anzeigen:

```
[RO] Get-CimInstance Win32_PageFileSetting |
    Select-Object Name,
                  InitialSize,
                  MaximumSize
```

Mögliche Hinweise auf Speicherdruck:

- `PercentCommittedBytesInUse` bleibt sehr hoch,
- verfügbarer Speicher bleibt niedrig,
- Page Reads steigen dauerhaft,
- Anwendungen melden fehlenden virtuellen Speicher,
- Prozesse werden beendet,
- Antwortzeiten verschlechtern sich parallel zur Paging-Aktivität.

> Das Vorhandensein einer verwendeten Auslagerungsdatei ist nicht automatisch ein Fehler. Entscheidend ist eine dauerhaft hohe Paging-Aktivität zusammen mit Leistungsproblemen.

---

10. Windows-Datenträger-I/O prüfen

Formatierte Datenträgerleistungsdaten anzeigen:

```
[RO] Get-CimInstance Win32_PerfFormattedData_PerfDisk_PhysicalDisk |
    Where-Object Name -ne "_Total" |
    Select-Object Name,
                  DiskReadsPerSec,
                  DiskWritesPerSec,
                  DiskReadBytesPerSec,
                  DiskWriteBytesPerSec,
                  AvgDiskQueueLength,
                  CurrentDiskQueueLength,
                  PercentDiskTime
```

Mehrere Messpunkte mit Get-Counter:

```
[RO] Get-Counter `
    -Counter '\PhysicalDisk(*)\Avg. Disk sec/Read',
             '\PhysicalDisk(*)\Avg. Disk sec/Write',
             '\PhysicalDisk(*)\Current Disk Queue Length' `
    -SampleInterval 2 `
    -MaxSamples 5
```

Auch hier können lokalisierte Leistungsindikatornamen abweichen.

Zu prüfen:

- steigt die Warteschlange dauerhaft?
- sind Lese- oder Schreiblatenzen erhöht?
- ist nur ein Datenträger betroffen?
- korreliert die I/O-Last mit dem Dienstfehler?
- erzeugt Sicherung, Virenscan oder Update gleichzeitig Last?
- ist das Volume lokal, virtuell oder netzwerkbasiert?
- meldet das System zusätzlich Datenträgerfehler?

---

11. Windows-Handles und Threads prüfen

Gesamtanzahl von Prozessen, Threads und Handles:

```
[RO] Get-CimInstance Win32_PerfFormattedData_PerfOS_System |
    Select-Object Processes, Threads
```

```
[RO] Get-CimInstance Win32_PerfFormattedData_PerfOS_Objects |
    Select-Object Processes,
                  Threads,
                  Events,
                  Mutexes,
                  Sections,
                  Semaphores
```

Prozesse nach Handleanzahl sortieren:

```
[RO] Get-Process |
    Sort-Object HandleCount -Descending |
    Select-Object -First 15 Id,
                            ProcessName,
                            HandleCount,
                            @{
                               Name="ThreadCount"
                               Expression={$_.Threads.Count}
                            }
```

Bestimmten Prozess prüfen:

```
[RO] Get-Process -Id <PID> |
    Select-Object Id,
                  ProcessName,
                  HandleCount,
                  @{
                     Name="ThreadCount"
                     Expression={$_.Threads.Count}
                  }
```

Ein mögliches Ressourcenleck zeigt sich eher durch kontinuierliches Wachstum als durch einen einzelnen hohen Wert.

---

12. Windows-Protokolle nach Ressourcenfehlern durchsuchen

Systemereignisse der letzten zwei Stunden:

```
[RO][SENS] Get-WinEvent -FilterHashtable @{
    LogName   = "System"
    StartTime = (Get-Date).AddHours(-2)
    Level     = 1, 2, 3
} |
    Select-Object TimeCreated,
                  ProviderName,
                  Id,
                  LevelDisplayName,
                  Message
```

System- und Anwendungsereignisse gemeinsam:

```
[RO][SENS] Get-WinEvent -FilterHashtable @{
    LogName   = "System", "Application"
    StartTime = (Get-Date).AddHours(-2)
} |
    Where-Object Message -Match "memory|resource|disk|paging|quota|handle|thread" |
    Sort-Object TimeCreated |
    Select-Object TimeCreated,
                  LogName,
                  ProviderName,
                  Id,
                  Message
```

> Textfilter hängen von Sprache und Formulierung der Ereignismeldung ab. Anbieter, Ereignis-ID und Zeitfenster sind zuverlässigere zusätzliche Filter.

---

13. CPU und Load Average unter Linux prüfen

Systemlaufzeit und Load Average anzeigen:

```
[RO] uptime
```

Nur Load Average anzeigen:

```
[RO][FILE] cat /proc/loadavg
```

Anzahl verfügbarer Verarbeitungseinheiten anzeigen:

```
[RO] nproc
```

Interaktive Prozessübersicht:

```
[RO] top
```

Nach CPU-Verbrauch sortieren:

```
[RO] ps -eo pid,ppid,user,stat,%cpu,%mem,etime,comm \
  --sort=-%cpu |
  head -n 16
```

Falls sysstat installiert ist – CPU je Prozessor messen:

```
[RO] mpstat -P ALL 2 5
```

> `mpstat`, `pidstat` und `iostat` gehören häufig zum Paket `sysstat`, sind aber nicht auf jeder Installation vorhanden.

---

14. Linux Load Average richtig interpretieren

Die drei Load-Average-Werte beziehen sich üblicherweise auf ungefähr:

```
1 Minute
5 Minuten
15 Minuten
```

Load Average ist nicht dasselbe wie CPU-Prozent. Unter Linux werden dabei auch bestimmte Prozesse in nicht unterbrechbarem Wartezustand berücksichtigt.

Grobe Einordnung:

```
Load Average
geteilt durch
Anzahl verfügbarer CPUs
```

Dies ist nur eine Orientierung.

Beobachtung	Mögliche Interpretation
Hoher Load und hohe CPU-Nutzung	CPU-Sättigung möglich
Hoher Load und niedrige CPU-Nutzung	I/O-Wartezustände möglich
Kurzzeitig hoher 1-Minuten-Wert	Lastspitze
Hohe 1-, 5- und 15-Minuten-Werte	Länger andauernde Belastung
Load steigt, Warteschlange wächst	System verarbeitet Arbeit nicht schnell genug

> Ein Load-Wert von `4` ist auf einem System mit zwei CPUs anders zu bewerten als auf einem System mit 32 CPUs.

---

15. Linux-CPU über mehrere Messpunkte prüfen

Systemweite Messreihe mit vmstat:

```
[RO] vmstat 2 5
```

Wichtige Spalten:

Spalte	Bedeutung
`r`	Ausführungsbereite Prozesse
`b`	Prozesse in blockiertem Zustand
`us`	CPU-Zeit im Benutzerbereich
`sy`	CPU-Zeit im Kernelbereich
`id`	Leerlauf
`wa`	I/O-Wartezeit
`st`	Durch Hypervisor entnommene CPU-Zeit
`si`	Swap-In
`so`	Swap-Out

> Die erste Zeile von `vmstat` kann Durchschnittswerte seit dem Systemstart darstellen. Für eine aktuelle Bewertung sind die folgenden Intervallzeilen wichtiger.

Bestimmten Prozess mit installiertem pidstat beobachten:

```
[RO] pidstat -p <PID> 2 5
```

Alle Prozesse mit hoher CPU-Aktivität beobachten:

```
[RO] pidstat -u 2 5
```

---

16. Arbeitsspeicher unter Linux prüfen

Lesbare Speicherübersicht:

```
[RO] free -h
```

Kernel-Speicherinformationen:

```
[RO][FILE] cat /proc/meminfo
```

Wichtige Werte:

Wert	Bedeutung
`MemTotal`	Gesamter physischer Speicher
`MemFree`	Vollständig ungenutzter Speicher
`MemAvailable`	Schätzung des für neue Anwendungen verfügbaren Speichers
`Buffers`	Pufferspeicher
`Cached`	Dateicache
`SwapTotal`	Gesamter Swap
`SwapFree`	Freier Swap
`Dirty`	Noch nicht auf Datenträger geschriebene Seiten
`Slab`	Kernel-Datenstrukturen

> Für die praktische Bewertung ist `MemAvailable` meist aussagekräftiger als `MemFree`, weil Linux freien RAM bewusst als Cache nutzt.

---

17. Linux-Prozesse nach Speicherverbrauch untersuchen

Nach residentem Speicher sortieren:

```
[RO] ps -eo pid,ppid,user,stat,%cpu,%mem,rss,vsz,etime,comm \
  --sort=-rss |
  head -n 16
```

Bestimmten Prozess anzeigen:

```
[RO][SENS] ps -p <PID> \
  -o user,pid,ppid,stat,%cpu,%mem,rss,vsz,etime,cmd
```

Detaillierte Prozessspeicherwerte:

```
[RO][FILE][SENS] cat "/proc/<PID>/status"
```

Relevant sind unter anderem:

```
VmPeak
VmSize
VmHWM
VmRSS
RssAnon
RssFile
VmSwap
Threads
```

Wert	Bedeutung
`VmSize`	Virtueller Adressraum
`VmRSS`	Residenter physischer Speicher
`VmHWM`	Bisheriger Höchstwert des residenten Speichers
`VmSwap`	Für den Prozess ausgelagerter Speicher
`Threads`	Threadanzahl

---

18. Swap und Paging unter Linux prüfen

Aktive Swap-Bereiche anzeigen:

```
[RO] swapon --show
```

Gesamtnutzung anzeigen:

```
[RO] free -h
```

Swap-Aktivität über mehrere Messpunkte:

```
[RO] vmstat 2 5
```

Relevant:

Spalte	Bedeutung

si	Von Swap eingelesene Daten
so	In Swap geschriebene Daten

Bewertung:

- verwendeter Swap allein beweist keinen aktuellen Engpass,
- dauerhaft hohe Werte bei si und so weisen auf aktiven Speicherdruck hin,
- gleichzeitige hohe I/O-Wartezeit kann die Anwendung stark verlangsamen,
- einzelne Prozesse können einen großen Teil des Speichers belegen,
- ein Container kann sein eigenes Speicherlimit erreichen, obwohl der Host noch freien RAM besitzt.

---

19. Linux-Datenträger-I/O prüfen

Grundlegende Datenträgerstatistik:

```
[RO] iostat
```

Falls sysstat installiert ist – erweiterte Messreihe:

```
[RO] iostat -xz 2 5
```

Wichtige Felder können je nach Version enthalten:

Feld	Bedeutung
r/s	Lesevorgänge pro Sekunde
w/s	Schreibvorgänge pro Sekunde
rkB/s	Gelesene KiB pro Sekunde
wkB/s	Geschriebene KiB pro Sekunde
await	Durchschnittliche Wartezeit
aqu-sz	Durchschnittliche Warteschlangenlänge
%util	Anteil der Zeit mit aktiven I/O-Anfragen

Prozessbezogene I/O-Aktivität mit pidstat:

```
[RO] pidstat -d 2 5
```

> `%util` und Warteschlangenwerte müssen zur Datenträgerart und Speicherarchitektur passend bewertet werden. RAID, SSD, SAN und virtuelle Datenträger verhalten sich unterschiedlich.

---

20. Linux Pressure Stall Information prüfen

Auf unterstützten Linux-Kernels stehen Druckinformationen unter `/proc/pressure` bereit.

CPU-Druck:

```
[RO][FILE] cat /proc/pressure/cpu
```

Speicherdruck:

```
[RO][FILE] cat /proc/pressure/memory
```

I/O-Druck:

```
[RO][FILE] cat /proc/pressure/io
```

Mögliche Zeilen:

```
some
full
```

Wert	Bedeutung
`some`	Mindestens eine Aufgabe wurde durch Ressourcenmangel verzögert
`full`	Alle nicht im Leerlauf befindlichen Aufgaben waren gleichzeitig verzögert
`avg10`	Durchschnitt der letzten 10 Sekunden
`avg60`	Durchschnitt der letzten 60 Sekunden
`avg300`	Durchschnitt der letzten 300 Sekunden
`total`	Kumulierte Verzögerungszeit in Mikrosekunden

> Fehlen die Dateien, unterstützt oder aktiviert das System diese Schnittstelle möglicherweise nicht.

---

21. Linux-Dateideskriptor- und Prozessgrenzen prüfen

Grenzwerte eines laufenden Prozesses:

```
[RO][FILE] cat "/proc/<PID>/limits"
```

Besonders relevant:

```
Max open files
Max processes
Max locked memory
Max address space
Max core file size
```

Anzahl geöffneter Dateideskriptoren:

```
[RO][PRIV] sudo find "/proc/<PID>/fd" \
  -mindepth 1 \
  -maxdepth 1 \
  -printf '.' |
    wc -c
```

systemd-Grenzwerte anzeigen:

```
[RO] systemctl show "<DIENST>" \
  --
property=LimitNOFILE,LimitNPROC,TasksCurrent,TasksMax,MemoryCurrent,MemoryMax,CPUQuotaPerSecU
Sec
```

Aktuelle Shell-Grenzen anzeigen:

```
[RO] ulimit -a
```

> `ulimit -a` zeigt die Grenzen der aktuellen Shell und nicht automatisch die Grenzen eines bereits laufenden Dienstes.

---

22. Linux-Protokolle nach Ressourcenfehlern durchsuchen

Warnungen und Fehler des aktuellen Systemstarts:

```
[RO][SENS][PRIV] sudo journalctl \
  -b \
  -p warning \
  --no-pager
```

Kernelmeldungen nach Speicherproblemen durchsuchen:

```
[RO][SENS][PRIV] sudo journalctl \
  -k \
  -b \
  --no-pager |
    grep -i -E -- "out of memory|oom|killed process|memory cgroup"
```

Nach I/O- und Dateisystemproblemen suchen:

```
[RO][SENS][PRIV] sudo journalctl \
  -k \
  -b \
  --no-pager |
    grep -i -E -- "i/o error|filesystem|read-only|blocked for more than"
```

Dienstprotokoll im Störungszeitraum:

```
[RO][SENS][PRIV] sudo journalctl \
  -u "<DIENST>" \
  --since "2026-07-31 09:10:00" \
  --until "2026-07-31 09:20:00" \
  --no-pager
```

---

23. CPU und Load Average unter macOS prüfen

Systemlaufzeit und Load Average:

```
[RO] uptime
```

Logische und physische CPUs:

```
[RO] sysctl -n hw.logicalcpu
```

```
[RO] sysctl -n hw.physicalcpu
```

Interaktive Übersicht nach CPU-Nutzung:

```
[RO] top -o cpu
```

Fünf Messungen im Abstand von zwei Sekunden:

```
[RO] top \
  -l 5 \
  -s 2 \
  -o cpu \
  -stats pid,command,cpu,mem,threads,state,time
```

Bestimmten Prozess beobachten:

```
[RO] top \
  -l 5 \
  -s 2 \
  -pid <PID> \
  -stats pid,command,cpu,mem,threads,state,time
```

> Bei `top -l` kann die erste CPU-Messung ungeeignet sein, weil zur Prozentberechnung noch kein vollständiges vorheriges Intervall vorliegt.

---

24. Speicher unter macOS prüfen

Speicherdruck anzeigen:

```
[RO] memory_pressure
```

Virtuelle Speicherstatistik:

```
[RO] vm_stat
```

Gesamten physischen Speicher anzeigen:

```
[RO] sysctl -n hw.memsize
```

Swap-Nutzung anzeigen:

```
[RO] sysctl vm.swapusage
```

Prozesse nach Speicherverbrauch sortieren:

```
[RO] ps -Amcwwxo pid,ppid,user,state,%cpu,%mem,rss,vsz,etime,command
```

Bestimmten Prozess anzeigen:

```
[RO][SENS] ps -p <PID> \
  -o user,pid,ppid,state,%cpu,%mem,rss,vsz,etime,command
```

> macOS verwaltet Arbeitsspeicher unter anderem durch Cache, Kompression und Swap. „Freier Speicher“ allein ist deshalb weniger aussagekräftig als Speicherdruck, Swap-Aktivität und Anwendungsreaktion.

---

25. vm_stat unter macOS interpretieren

`vm_stat` zeigt Speicherwerte in Seiten an. Die Seitengröße wird in der ersten Ausgabezeile genannt.

Wichtige Werte können sein:

Wert	Bedeutung
`Pages free`	Freie Seiten
`Pages active`	Aktiv verwendete Seiten
`Pages inactive`	Inaktive, möglicherweise wiederverwendbare Seiten
`Pages speculative`	Spekulativ geladene Seiten
`Pages wired down`	Nicht auslagerbarer Speicher
`Pages occupied by compressor`	Durch Speicherkompression belegte Seiten
`Pageins`	Vom Datenträger eingelesene Seiten
`Pageouts`	Auf Datenträger ausgelagerte Seiten
`Swapins`	Aus Swap eingelesene Seiten
`Swapouts`	In Swap geschriebene Seiten

> Viele Werte sind kumuliert seit dem Systemstart. Für aktuelle Aktivität müssen Messungen über einen definierten Zeitraum verglichen werden.

---

26. Datenträger-I/O unter macOS prüfen

Datenträgerübersicht:

```
[RO] iostat
```

Fünf Messungen im Abstand von zwei Sekunden:

```
[RO] iostat -w 2 -c 5
```

CPU- und Datenträgerstatistik ausführlicher anzeigen:

```
[RO] iostat -d -c 5 -w 2
```

Prozessbezogene Beobachtung mit top:

```
[RO] top \
  -l 5 \
  -s 2 \
  -o cpu \
  -stats pid,command,cpu,mem,purg,cmprs,pageins,faults
```

Zu prüfen:

- steigt Datenträgeraktivität während der Störung?
- tritt gleichzeitig Speicherdruck auf?
- verursacht ein anderer Prozess hohe I/O-Last?
- ist das Ziel ein internes, externes oder Netzwerkvolume?
- gibt es Dateisystem- oder Hardwaremeldungen?
- wird gleichzeitig gesichert, indiziert oder aktualisiert?

---

27. Prozess- und Dateigrenzen unter macOS prüfen

Grenzwerte von launchd anzeigen:

```
[RO] launchctl limit
```

Grenzwerte der aktuellen Shell anzeigen:

```
[RO] ulimit -a
```

Offene Ressourcen eines Prozesses anzeigen:

```
[RO][SENS][PRIV] sudo lsof -nP -p <PID>
```

Anzahl der ausgegebenen offenen Ressourcen grob zählen:

```
[RO][SENS][PRIV] sudo lsof -nP -p <PID> |
    wc -l
```

Die Kopfzeile von `lsof` wird dabei mitgezählt.

Threadanzahl anzeigen:

```
[RO] ps -M -p <PID>
```

> Die Grenzwerte einer Shell sind nicht automatisch mit denen eines launchd-Dienstes identisch. Für den Dienst müssen launchd-Konfiguration, Prozesskontext und tatsächlicher Verbrauch berücksichtigt werden.

---

28. macOS-Protokolle nach Ressourcenproblemen prüfen

Fehler und Faults der letzten Stunde:

```
[RO][SENS][PRIV] sudo log show \
  --last 1h \
  --predicate 'messageType == error OR messageType == fault' \
  --style compact \
  --no-pager
```

Dienstprozess filtern:

```
[RO][SENS][PRIV] sudo log show \
  --last 1h \
  --predicate 'process == "<PROZESS>"' \
  --style compact \
  --no-pager
```

Nach ressourcenbezogenen Meldungstexten suchen:

```
[RO][SENS][PRIV] sudo log show \
  --last 1h \
  --predicate 'eventMessage CONTAINS[c] "memory" OR eventMessage CONTAINS[c] "resource" OR
eventMessage CONTAINS[c] "disk"' \
  --style compact \
  --no-pager
```

> Textfilter sind nur eine Ergänzung. Prozess, Zeitfenster, Subsystem und konkrete Fehlermeldung müssen ebenfalls berücksichtigt werden.

---

29. CPU-Engpass systematisch erkennen

Belastbare Hinweise:

- CPU-Auslastung bleibt über längere Zeit hoch,
- ausführungsbereite Prozesse warten,
- Antwortzeiten steigen gleichzeitig,
- Anwendungswarteschlange wächst,
- Timeouts treten auf,
- ein bestimmter Prozess oder Thread verursacht den Großteil der Last,
- virtueller Host meldet hohe CPU-Steal- oder Ready-Zeit,
- CPU-Limit eines Containers wird erreicht.

Mögliche Ursachen:

- regulär hohe Benutzerlast,
- Endlosschleife,
- fehlerhafte Abfrage,
- Kompression oder Verschlüsselung,

- Virenscan,
- Sicherung,
- Softwareupdate,
- zu viele Prozesse oder Threads,
- zu niedriges Containerlimit,
- fehlende Hardwarebeschleunigung.

Nicht ausreichend als Beweis:

```
Ein einzelner Messwert von 100 %
```

---

30. Arbeitsspeicherengpass systematisch erkennen

Belastbare Hinweise:

- verfügbarer Speicher bleibt dauerhaft niedrig,
- Paging oder Swapping ist kontinuierlich aktiv,
- Speicherdruck steigt,
- Antwortzeiten steigen parallel,
- Prozesse werden durch Speichermangel beendet,
- Anwendung meldet Allokationsfehler,
- Prozessspeicher wächst kontinuierlich,
- Container erreicht `MemoryMax` oder sein Plattformlimit,
- Cache kann nicht ausreichend zurückgewonnen werden.

Mögliche Ursachen:

- Speicherleck,
- zu großer Cache,
- unerwartet viele Sitzungen,
- zu große Datenmenge,
- mehrere speicherintensive Dienste,
- zu kleines VM- oder Containerlimit,
- fehlerhafte Heap-Konfiguration,
- fehlende Begrenzung von Workerprozessen.

> Ein Dienstneustart kann den Speicherverbrauch kurzfristig reduzieren, beweist aber nicht, dass die Ursache beseitigt wurde.

---

31. I/O-Engpass systematisch erkennen

Belastbare Hinweise:

- I/O-Wartezeiten steigen,
- Warteschlange bleibt gefüllt,
- Prozesse verbleiben im Wartezustand,
- Datenbankabfragen werden langsam,
- Schreibvorgänge blockieren,
- gleichzeitig erscheinen I/O- oder Dateisystemfehler,
- Netzwerkstorage zeigt hohe Latenz,
- Sicherung oder Indizierung erzeugt konkurrierende Last.

Mögliche Ursachen:

- langsamer oder fehlerhafter Datenträger,
- RAID-Rebuild,
- defekter Datenträger,
- überlastetes SAN oder NAS,
- hoher Speicherdruck und Paging,
- ungeeignete Datenbankabfrage,
- fehlender Cache,
- Sicherung oder Virenscan,
- zu viele parallele Schreibvorgänge,
- Volume- oder IOPS-Limit einer Cloudplattform.

> Hoher Datendurchsatz kann normal sein. Ein Engpass zeigt sich durch Wartezeit, Warteschlange und beeinträchtigte Anwendung.

---

32. Ressourcenleck erkennen

Ein Ressourcenleck wird durch eine Zeitreihe sichtbar.

Mögliche Lecks:

- Arbeitsspeicher,
- Handles,
- Dateideskriptoren,
- Threads,
- Netzwerkverbindungen,

- temporäre Dateien,
- Datenbankverbindungen,
- Warteschlangeneinträge.

Typisches Muster:

```
Dienststart          niedriger Verbrauch
nach 1 Stunde        höherer Verbrauch
nach 6 Stunden       weiter gestiegen
nach 24 Stunden      Grenzwert erreicht
nach Dienstneustart  wieder niedriger Verbrauch
```

Nachweisstrategie:

1. gleiche Messgröße verwenden,
2. regelmäßige Zeitpunkte erfassen,
3. Last und Benutzerzahl mit dokumentieren,
4. Prozessneustarts berücksichtigen,
5. Grenzwert und Fehlereintritt bestimmen,
6. Herstellerdiagnose oder Profiler kontrolliert einsetzen.

> Wachstum kann durch einen vorgesehenen Cache entstehen. Ein Leck ist erst wahrscheinlich, wenn Ressourcen trotz abgeschlossener Arbeit nicht angemessen freigegeben werden und der Verbrauch problematisch weiter steigt.

---

33. Virtualisierung und Containerlimits berücksichtigen

Ein Gastbetriebssystem kann normale Hostwerte anzeigen, obwohl die Virtualisierungsplattform Ressourcen begrenzt.

Zu prüfen:

- zugewiesene virtuelle CPUs,
- CPU-Limit oder CPU-Anteile,
- CPU-Steal- beziehungsweise Ready-Zeit,
- zugesicherter und maximaler RAM,
- Memory Ballooning,
- Host-Swapping,

- IOPS- und Durchsatzlimits,
- Container-CPU-Quota,
- Container-Speicherlimit,
- Prozess- beziehungsweise PID-Limit,
- gemeinsam verwendete Hostressourcen,
- konkurrierende virtuelle Maschinen oder Container.

Typische Fehlinterpretation:

> „Der Host hat noch freien RAM, deshalb kann der Container keinen Speichermangel haben.“

Ein Container kann sein eigenes Limit erreichen, obwohl der Host noch freie Ressourcen besitzt.

---

34. Engpass anhand einer Vergleichsmatrix eingrenzen

CPU	RAM	I/O-Wartezeit	Mögliche Richtung
Hoch	Normal	Niedrig	CPU-intensive Verarbeitung
Niedrig	Knapp	Hoch	Paging oder Speicherdruck
Niedrig	Normal	Hoch	Datenträger- oder Netzwerkstorage
Hoch	Knapp	Hoch	Gesamtsystem überlastet
Niedrig	Normal	Niedrig	Externes Backend, Sperre oder Anwendung
Normal	Verbrauch steigt	Zunehmend	Mögliches Speicherleck
Normal	Normal	Normal	Anwendungs-, Netzwerk- oder Konfigurationsfehler

Diese Matrix ist eine Eingrenzungshilfe und kein automatischer Ursachenbeweis.

---

35. Typische Fehlinterpretationen

Fehlinterpretation	Richtige Bewertung
„100 Prozent CPU bedeutet immer einen Fehler.“	Kurze oder produktive Vollauslastung kann normal sein

Fehlinterpretation	Richtige Bewertung
„Freier RAM ist niedrig, also fehlt Speicher.“	Cache und verfügbarer Speicher müssen berücksichtigt werden
„Swap wird verwendet, also ist das System überlastet.“	Aktuelle Swap-Aktivität und Speicherdruck sind entscheidend
„Hoher Datendurchsatz bedeutet I/O-Engpass.“	Wartezeit und Warteschlange sind entscheidender
„Load Average entspricht CPU-Prozent.“	Load berücksichtigt unter Linux weitere Wartezustände
„Die erste vmstat-Zeile zeigt die aktuelle Last.“	Sie kann Durchschnittswerte seit dem Start enthalten
„Get-Process CPU ist der aktuelle Prozentwert.“	Es ist kumulierte CPU-Zeit
„Virtueller Speicher ist vollständig belegter RAM.“	Virtueller Adressraum und residenter Speicher unterscheiden sich
„Ein Neustart behebt das Ressourcenproblem.“	Er kann Zähler und Verbrauch nur vorübergehend zurücksetzen
„Der Host hat Ressourcen, also hat der Container sie ebenfalls.“	Container und VM können eigene Limits besitzen

36. Dokumentationsvorlage für Ressourcenengpässe

Störung:

Server:

Dienst:

Prozess und PID:

Betriebssystem:

Virtualisiert beziehungsweise containerisiert:

Messzeitraum:

Abtastintervall:

CPU gesamt:

CPU des Prozesses:

Load Average:

Verfügbarer Arbeitsspeicher:

Speicher des Prozesses:

Swap- beziehungsweise Paging-Aktivität:

Datenträgerlatenz:

Datenträgerwarteschlange:

Dateideskriptor- beziehungsweise Handleanzahl:

Threadanzahl:

Konfigurierte Limits:

Anwendungsantwortzeit:

Fehlerrate:

Gleichzeitige Benutzer oder Aufgaben:

Relevante Protokollmeldung:

Vergleichswert:

Bewertung:

Nächster Prüfschritt:

37. Checkliste zur Ressourcenanalyse

[ ] Störungszeitraum genau bestimmt

[ ] System- und Prozesswerte getrennt erfasst

[ ] Mehrere Messpunkte aufgenommen

[ ] Normalen Vergleichszeitraum bestimmt

[ ] Anzahl logischer CPUs berücksichtigt

[ ] CPU-Auslastung und Warteschlange geprüft

[ ] Load Average richtig eingeordnet

[ ] Verfügbaren Arbeitsspeicher geprüft

[ ] Prozessspeicher geprüft

[ ] Paging beziehungsweise Swapping geprüft

[ ] Speicherdruck berücksichtigt

[ ] Datenträgerdurchsatz geprüft

[ ] I/O-Wartezeit und Warteschlange geprüft

[ ] Speicherplatz und Dateisystem grob geprüft

[ ] Handle- beziehungsweise Dateideskriptoranzahl geprüft

[ ] Thread- und Prozessgrenzen geprüft

[ ] Ressourcenlecks über Zeitreihe geprüft

[ ] Betriebssystemprotokolle geprüft

[ ] Anwendungsprotokolle geprüft

[ ] VM- und Containerlimits berücksichtigt

[ ] Gleichzeitige Sicherungen und Updates berücksichtigt

[ ] Messwerte mit Antwortzeit und Fehlerrate korreliert

[ ] Noch keine ungeprüfte Limitänderung durchgeführt

[ ] Ursache und nächster Schritt dokumentiert

Bewertung des Ergebnisses

Ergebnis	Nächster Schritt
CPU dauerhaft gesättigt	Verursachenden Prozess, Threads und Arbeitslast untersuchen
Hoher Load bei niedriger CPU	I/O-Wartezustände und blockierte Prozesse prüfen
Speicherdruck und aktives Swapping	Prozessspeicher, Limits und mögliches Leck untersuchen
Prozessverbrauch steigt kontinuierlich	Zeitreihe und Herstellerdiagnose erstellen
Datenträgerlatenz erhöht	Datenträger, Dateisystem, Storage und konkurrierende Last prüfen
Dateideskriptor- oder Handlegrenze erreicht	Leck, Grenzwert und benötigte Kapazität untersuchen
Containerlimit erreicht	Sollressourcen und Plattformkonfiguration prüfen
Host überlastet	Andere VMs, Container und Hostprozesse berücksichtigen
Ressourcenwerte normal	Abhängigkeiten, Sperren, Netzwerk und Anwendung untersuchen
Speicherplatz oder Dateisystem auffällig	Mit Seite 4.11 detailliert weiterprüfen

---

Merksatz

> **Ein Ressourcenengpass wird nicht durch einen hohen Wert bewiesen, sondern durch dauerhaftes Warten, erreichte Grenzen und eine gleichzeitig beeinträchtigte Dienstfunktion.**

Weiterführende Quellen

- Microsoft Learn – Get-Counter
- Microsoft Learn – Get-Process
- Microsoft Learn – Get-CimInstance
- Microsoft Learn – Win32_Process
- Microsoft Learn – Windows-Leistungsüberwachung
- Linux-Handbuch – proc_loadavg
- Linux-Handbuch – proc_meminfo
- Linux-Handbuch – proc_pid_limits
- Linux-Handbuch – proc_pressure
- systemd – systemd.resource-control
- Apple – top-Handbuchseite

- Apple – vm_stat-Handbuchseite
- Apple – iostat-Handbuchseite
- Apple – launchctl-Handbuchseite
