2.13 Monitoringdaten und Leistungswerte richtig interpretieren

Monitoring erfasst den Zustand von Systemen und Diensten über einen längeren Zeitraum. Im Gegensatz zu einer einzelnen Momentaufnahme zeigt es Entwicklungen, wiederkehrende Muster und Zusammenhänge zwischen verschiedenen Messwerten.

Monitoringdaten helfen unter anderem bei folgenden Fragen:

- Wann begann eine Leistungsverschlechterung?
- Ist ein Problem dauerhaft, periodisch oder nur kurzfristig?
- Welche Ressource ist ausgelastet oder überlastet?
- Betrifft die Störung einen einzelnen Host oder mehrere Systeme?
- Tritt das Problem nur zu bestimmten Tageszeiten auf?
- Gab es unmittelbar vorher ein Update, Deployment oder Backup?
- Steigt die Fehlerrate gleichzeitig mit der Antwortzeit?
- Wird eine technische Auffälligkeit von Benutzern tatsächlich wahrgenommen?
- Entwickelt sich ein Wert langsam in Richtung eines Kapazitätsproblems?
- Fehlen Messwerte, weil der Zielhost oder nur das Monitoring ausgefallen ist?

> **Grundregel:** Ein einzelner hoher Messwert beweist noch keine Ursache. Entscheidend sind Verlauf, Dauer, Vergleichswerte, Benutzerwirkung und die Korrelation mit anderen Signalen.

---

1. Kennzeichnungen und Sicherheitsregeln

Kennzeichnung	Bedeutung
[RO]	Liest Informationen aus, ohne den Zustand absichtlich zu verändern
[TEST]	Führt eine aktive Messung durch oder erzeugt zusätzliche Last
[PRIV]	Benötigt möglicherweise Administrator- oder Root-Rechte
[FILE]	Erzeugt oder verändert eine Datei
[SENS]	Ausgabe kann vertrauliche Daten enthalten
[CHANGE]	Verändert Einstellungen oder den Systemzustand
[DISRUPT]	Kann Dienste oder Systeme beeinträchtigen

Monitoringdaten können vertrauliche Informationen enthalten:

- Hostnamen und IP-Adressen,
- Benutzer- und Prozessnamen,
- interne URLs und Dienstbezeichnungen,
- Standort- und Kundenzuordnungen,
- Kapazitäts- und Auslastungsdaten,
- Datenbank- und Mandantennamen,
- Informationen über Sicherheits- und Netzwerkinfrastruktur.

Dashboards, Exporte und Screenshots dürfen deshalb nur kontrolliert weitergegeben werden.

2. Was ist der Unterschied zwischen Monitoring und Observability?

Begriff	Bedeutung
Monitoring	Erfasst vorher festgelegte Zustände und Messwerte
Logging	Dokumentiert einzelne Ereignisse und Meldungen
Metriken	Numerische Werte über einen Zeitraum
Tracing	Verfolgt eine Anfrage durch mehrere Komponenten
Observability	Ermöglicht, den internen Systemzustand aus Metriken, Logs und Traces abzuleiten
Alerting	Meldet definierte Zustände oder Abweichungen
Dashboard	Visualisiert ausgewählte Messwerte
Profiling	Untersucht, wo ein Programm Rechenzeit oder Speicher verwendet

Zusammenhang

Metrik zeigt:

Die Antwortzeit ist seit 09:42 Uhr erhöht.

Log zeigt:

Datenbankabfragen laufen in ein Timeout.

Trace zeigt:

Die Verzögerung entsteht im Datenbankzugriff.

Profiling zeigt:

Eine bestimmte Funktion erzeugt besonders teure Abfragen.

Monitoring zeigt häufig, **dass** ein Problem besteht. Logs, Traces und weitere Diagnosewerkzeuge helfen anschließend festzustellen, **warum** es besteht.

3. Wie wird eine Baseline erstellt?

Eine Baseline beschreibt den normalen Zustand eines Systems unter vergleichbaren Bedingungen.

Sinnvolle Vergleichszeiträume sind beispielsweise:

- gleicher Wochentag der Vorwoche,
- gleiche Uhrzeit an mehreren Werktagen,
- Zeitraum vor einem Update,
- Zeitraum mit ähnlicher Benutzerzahl,
- funktionierendes Referenzsystem,
- Normalbetrieb außerhalb eines Backups,
- typischer Monats- oder Quartalsverlauf.

Beispiel

Messwert	Normalbetrieb	Störungszeitraum
CPU-Auslastung	25–45 %	85–100 %
Antwortzeit p95	180–260 ms	3.800 ms
Fehlerrate	0,2 %	12 %
Datenbankverbindungen	20–35	100
Datenträgerlatenz	2–8 ms	140 ms
Anfragen pro Sekunde	100–140	110

Da die Anfragemenge nahezu unverändert ist, die Antwortzeit, Fehlerrate und Datenträgerlatenz aber stark gestiegen sind, sollte die Untersuchung auf Datenträger und Datenbank konzentriert werden.

Eine sinnvolle Baseline berücksichtigt:

- Tages- und Wochenmuster,
- Geschäftszeiten,
- Backups und Wartungsfenster,
- geplante Batch-Verarbeitung,
- saisonale Last,

- unterschiedliche Hardware,
- unterschiedliche Softwareversionen,
- Anzahl der Benutzer oder Anfragen.

Ein Grenzwert ohne Baseline ist häufig willkürlich.

4. Welche Arten von Metriken gibt es?

Metriktyp	Verhalten	Beispiel
Counter	steigt normalerweise an und kann bei Neustart zurückgesetzt werden	Gesamtzahl der HTTP-Anfragen
Gauge	kann steigen und fallen	aktuelle Speichernutzung
Histogram	zählt Beobachtungen in Wertebereichen	Verteilung von Antwortzeiten
Summary	berechnet Beobachtungen und Quantile clientseitig	Antwortzeitquantile
Rate	Änderung eines Counters pro Zeiteinheit	Fehler pro Sekunde
Ratio	Verhältnis zweier Werte	Fehleranteil an allen Anfragen

Counter

```
requests_total = 150000
```

Der Gesamtwert ist für eine aktuelle Belastung meist weniger aussagekräftig als seine Änderungsrate.

PromQL-Beispiel:

```
rate(http_requests_total[5m])
```

Gauge

```
active_connections = 42
```

Der Wert beschreibt den aktuellen Zustand und kann steigen oder fallen.

PromQL-Beispiele:

```
avg_over_time(active_connections[15m])
```

```
max_over_time(active_connections[15m])
```

Fehlerrate berechnen

```
sum(rate(http_requests_total{status=~"5.."}[5m]))
/
sum(rate(http_requests_total[5m]))
```

Für eine Prozentdarstellung:

```
100 *
sum(rate(http_requests_total{status=~"5.."}[5m]))
/
sum(rate(http_requests_total[5m]))
```

> `rate()` ist für Counter vorgesehen. Auf einen Gauge angewendet würde die Abfrage inhaltlich meist keinen sinnvollen Messwert ergeben.

5. Welche Messstrategien eignen sich für Infrastruktur und Dienste?

USE-Methode für Ressourcen

Buchstabe	Bedeutung	Fragestellung
U	Utilization	Wie stark wird die Ressource verwendet?
S	Saturation	Gibt es Warteschlangen oder Rückstau?
E	Errors	Treten Fehler auf?

Beispiel für einen Datenträger:

USE-Bereich	Messwert
Utilization	aktive Zeit des Datenträgers

USE-Bereich	Messwert
Saturation	Warteschlangenlänge
Errors	I/O-Fehler und Timeouts

RED-Methode für Dienste

Buchstabe	Bedeutung	Fragestellung
R	Rate	Wie viele Anfragen werden verarbeitet?
E	Errors	Wie viele Anfragen schlagen fehl?
D	Duration	Wie lange dauert die Verarbeitung?

Golden Signals

Signal	Bedeutung
Latency	Dauer einer Anfrage
Traffic	aktuelle Nutzung oder Anfragemenge
Errors	fehlgeschlagene Operationen
Saturation	Annäherung an eine Kapazitätsgrenze

USE hilft besonders bei der Untersuchung technischer Ressourcen. RED und die Golden Signals zeigen stärker die tatsächliche Benutzerwirkung.

6. Wie werden Durchschnitt, Maximum und Perzentile interpretiert?

Beispiel für zehn Antwortzeiten

```
100, 110, 115, 120, 125, 130, 140, 150, 200, 5000 ms
```

Der Durchschnitt wird durch den einzelnen sehr langsamen Wert deutlich beeinflusst. Gleichzeitig kann ein Durchschnitt die Verteilung und einzelne besonders langsame Anfragen verbergen.

Kennzahl	Aussage
Minimum	kleinster gemessener Wert
Maximum	größter gemessener Wert
Durchschnitt	arithmetischer Mittelwert

Kennzahl	Aussage
Median beziehungsweise p50	50 % der Werte liegen höchstens hier
p90	90 % der Werte liegen höchstens hier
p95	95 % der Werte liegen höchstens hier
p99	99 % der Werte liegen höchstens hier

Beispiel

```
p50 = 120 ms
p95 = 800 ms
p99 = 4.500 ms
```

Interpretation:

- Die typische Anfrage ist schnell.
- Ein kleinerer Anteil der Anfragen ist deutlich langsamer.
- Der Durchschnitt allein würde das Benutzerproblem möglicherweise verbergen.

Wichtig:

- Ein Perzentil ist kein Prozentwert der Auslastung.
- p95 bedeutet nicht, dass 95 % der Anfragen fehlerhaft sind.
- Perzentile verschiedener Gruppen dürfen nicht ohne Weiteres gemittelt werden.
- Bei sehr wenigen Messwerten können hohe Perzentile wenig belastbar sein.
- Erfolgreiche und fehlgeschlagene Anfragen sollten gegebenenfalls getrennt betrachtet werden.

7. Wie wird CPU-Auslastung richtig interpretiert?

Hohe CPU-Auslastung bedeutet zunächst nur, dass Rechenzeit verwendet wird. Sie kann sowohl normal als auch problematisch sein.

Zu prüfende Fragen

- Betrifft die Auslastung einen einzelnen Kern oder alle Kerne?
- Welcher Prozess verursacht sie?
- Ist die Auslastung nur kurzzeitig oder dauerhaft?
- Steigt gleichzeitig die Antwortzeit?
- Gibt es eine CPU-Warteschlange?
- Entsteht die Last im Benutzer-, Kernel- oder Interrupt-Kontext?
- Ist die CPU tatsächlich ausgelastet oder wartet das System auf I/O?

- Wird die CPU durch Virtualisierung oder Container begrenzt?
- Tritt Drosselung durch Temperatur oder Leistungsgrenzen auf?

Typische CPU-Zustände unter Linux

Zustand	Bedeutung
`us`	Zeit für Benutzerprozesse
`sy`	Zeit im Kernel
`id`	Leerlauf
`wa`	Warten auf I/O
`st`	Zeit, die einer VM durch den Hypervisor entzogen wurde
`hi`	Hardware-Interrupts
`si`	Software-Interrupts

Beispiele

Beobachtung	Untersuchungsrichtung
hohes `us`	Anwendungsprozess, Berechnung oder Schleife
hohes `sy`	Kernel, Treiber, Systemaufrufe oder Netzwerk
hohes `wa`	Datenträger oder anderes blockierendes I/O
hohes `st` in VM	Überbelegung oder Belastung des Hypervisors
hohe Interruptlast	Netzwerkadapter, Treiber oder Hardware
ein Kern bei 100 %	möglicherweise einzelner nicht parallelisierter Thread

Windows

```
[RO] Get-Counter '\Processor(_Total)\% Processor Time'
```

Alle logischen Prozessoren:

```
[RO] Get-Counter '\Processor(*)\% Processor Time'
```

Prozessbezogene CPU-Werte:

```
[RO] Get-Counter '\Process(*)\% Processor Time'
```

Mehrere Messungen im Abstand von zwei Sekunden:

```
[RO] Get-Counter '\Processor(_Total)\% Processor Time' `
    -SampleInterval 2 `
    -MaxSamples 10
```

Linux

```
[RO] uptime
```

```
[RO] top
```

```
[RO] vmstat 1 10
```

Falls `sysstat` installiert ist:

```
[RO] mpstat -P ALL 1 10
```

Prozesse nach CPU-Auslastung sortieren:

```
[RO] ps -eo pid,ppid,user,%cpu,%mem,comm --sort=-%cpu | head
```

macOS

```
[RO] top -l 1 -o cpu
```

```
[RO] ps -Ao pid,ppid,user,%cpu,%mem,comm -r | head
```

> Eine kurzzeitige CPU-Auslastung von 100 % ist nicht automatisch ein Problem. Kritisch wird sie insbesondere dann, wenn sie länger anhält, Warteschlangen erzeugt und gleichzeitig die Antwortzeit oder Fehlerrate steigt.

8. Was bedeutet Load Average unter Linux und macOS?

Load Average wird typischerweise für die letzten 1, 5 und 15 Minuten angezeigt:

```
[RO] uptime
```

Beispiel:

```
load average: 8.20, 6.10, 3.40
```

Die Werte sind keine Prozentwerte.

Unter Linux umfasst die Last unter anderem ausführbare beziehungsweise auf CPU wartende Tasks und Tasks in nicht unterbrechbarem Wartezustand, beispielsweise bei bestimmten I/O-Vorgängen.

Logische CPUs bestimmen

Linux:

```
[RO] nproc
```

macOS:

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

Vereinfachtes Beispiel

```
8 logische CPUs
Load Average 1 Minute: 8
```

Das kann bedeuten, dass die ausführbaren Kapazitäten ungefähr vollständig beansprucht sind. Die genaue Bewertung erfordert jedoch zusätzliche Informationen.

```
8 logische CPUs
Load Average 1 Minute: 16
```

Das deutet auf mehr gleichzeitig wartende oder nicht unterbrechbar blockierte Tasks hin, als unmittelbar abgearbeitet werden können.

Wichtige Einschränkungen

- Load Average ist nicht gleich CPU-Auslastung.
- I/O-Wartezustände können die Load erhöhen.
- Container sehen je nach Konfiguration Host- oder begrenzte Ressourcen.
- CPU-Quotas und virtuelle CPUs müssen berücksichtigt werden.
- Ein Wert ist nur im Verhältnis zur verfügbaren Kapazität sinnvoll.

9. Wie wird Arbeitsspeicher richtig interpretiert?

Ein fast vollständig belegter physischer Arbeitsspeicher bedeutet nicht automatisch Speichermangel. Betriebssysteme verwenden freien Speicher unter anderem als Cache.

Wichtiger als der reine Wert „belegt“ sind:

- verfügbarer Speicher,
- Speicherdruck,
- Paging- beziehungsweise Swap-Aktivität,
- Commit-Nutzung,
- OOM-Ereignisse,
- Entwicklung des Prozessspeichers,
- Antwortzeit des Systems.

Begriffe

Begriff	Bedeutung
Physical Memory	tatsächlich vorhandener RAM
Available	kurzfristig für Anwendungen verfügbarer Speicher
Cache	für schnellere Zugriffe verwendeter Speicher
Working Set	aktuell im RAM befindliche Seiten eines Prozesses
Private Bytes	nur einem Prozess zugeordneter zugesicherter Speicher
Commit	zugesicherter virtueller Speicher
Pagefile beziehungsweise Swap	Auslagerungsspeicher
Page Fault	Zugriff auf nicht aktuell passend zugeordnete Speicherseite
Hard Page Fault	benötigte Seite muss aus Datei oder Auslagerung geladen werden

Viele Page Faults sind normal. Erst eine anhaltend hohe Rate teurer Datenträgerzugriffe zusammen mit Speicherdruck und schlechter Leistung ist ein deutlicher Problemhinweis.

Windows

```
[RO] Get-Counter '\Memory\Available MBytes'
```

```
[RO] Get-Counter '\Memory\% Committed Bytes In Use'
```

```
[RO] Get-Counter '\Memory\Pages/sec'
```

Prozesse nach Arbeitsspeichernutzung:

```
[RO] Get-Process |
    Sort-Object WorkingSet64 -Descending |
    Select-Object -First 10 Name,
        Id,
        @{Name='WorkingSetMiB';Expression={
            [math]::Round($_.WorkingSet64 / 1MB, 1)
        }}
```

Linux

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

```
[RO] vmstat 1 10
```

Prozesse nach Speicheranteil:

```
[RO] ps -eo pid,user,%mem,rss,vsz,comm --sort=-%mem | head
```

OOM-Ereignisse suchen:

```
[RO][PRIV] sudo journalctl -k |
    grep -Ei 'out of memory|oom-killer|killed process'
```

macOS

```
[RO] memory_pressure
```

```
[RO] vm_stat
```

```
[RO] top -l 1 -o mem
```

Hinweise auf ein mögliches Speicherproblem

- verfügbarer Speicher bleibt sehr niedrig,
- Swap- oder Paging-Aktivität ist dauerhaft hoch,
- Speicherdruck steigt,
- Anwendungen reagieren gleichzeitig langsamer,
- ein Prozess wächst ohne Rückgang weiter,
- der OOM-Killer beendet Prozesse,
- Commit nähert sich dauerhaft der Grenze.

10. Wie werden Datenträgerauslastung und I/O-Latenz interpretiert?

Zu einer Datenträgeranalyse gehören mindestens:

- Durchsatz,
- Ein- und Ausgabeoperationen pro Sekunde,
- Latenz,
- Warteschlangen,
- Auslastungsgrad,
- Fehler und Timeouts,
- freier Speicherplatz.

Messwert	Aussage
IOPS	Anzahl der Operationen pro Sekunde
Throughput	übertragene Datenmenge pro Sekunde
Latency	Dauer einer Operation
Queue Length	wartende oder laufende I/O-Anfragen
Utilization	Anteil der aktiven Messzeit
Free Space	verfügbarer Speicherplatz

Messwert	Aussage
Errors	fehlerhafte oder abgebrochene Operationen

Hohe IOPS sind nicht automatisch problematisch. Ein Speichersystem kann viele kleine Operationen oder wenige große Übertragungen verarbeiten. Entscheidend ist, ob Latenz und Warteschlange unter der aktuellen Last steigen.

Windows

```
[RO] Get-Counter '\PhysicalDisk(*)\% Disk Time'
```

```
[RO] Get-Counter '\PhysicalDisk(*)\Avg. Disk sec/Read'
```

```
[RO] Get-Counter '\PhysicalDisk(*)\Avg. Disk sec/Write'
```

```
[RO] Get-Counter '\PhysicalDisk(*)\Current Disk Queue Length'
```

```
[RO] Get-Counter '\PhysicalDisk(*)\Disk Reads/sec'
```

```
[RO] Get-Counter '\PhysicalDisk(*)\Disk Writes/sec'
```

Freien Speicherplatz prüfen:

```
[RO] Get-Volume |
    Select-Object DriveLetter,
                  FileSystemLabel,
                  FileSystem,
                  HealthStatus,
                  SizeRemaining,
                  Size
```

> Windows Performance Counter für `Avg. Disk sec/Read` und `Avg. Disk sec/Write` werden in Sekunden ausgegeben. `0,020` entspricht 20 Millisekunden.

Linux

```
[RO] df -hT
```

```
[RO] df -i
```

Falls `sysstat` installiert ist:

```
[RO] iostat -xz 1 10
```

Prozessbezogene I/O-Werte:

```
[RO] pidstat -d 1 10
```

Kernelmeldungen zu I/O-Fehlern:

```
[RO][PRIV] sudo journalctl -k |
    grep -Ei 'I/O error|timeout|reset|read-only|filesystem'
```

macOS

```
[RO] df -h
```

```
[RO] iostat -w 1 -c 10
```

Typische Fehlinterpretationen

Beobachtung	Fehlinterpretation
Datenträger zu 100 % aktiv	maximale Datenübertragungsrate erreicht
wenig Durchsatz	Datenträger ist nicht belastet
hoher Durchsatz	Datenträger ist überlastet
voller Speicherplatz	einzige mögliche Ursache ist Datenmenge
hohe Latenz	physischer Datenträger ist zwingend defekt

Ein Datenträger kann bei kleinen zufälligen Zugriffen vollständig beschäftigt sein, obwohl der Datendurchsatz gering bleibt.

11. Wie werden Netzwerkmesswerte interpretiert?

Wichtige Netzwerkmesswerte:

Messwert	Bedeutung
Bandbreite	theoretische oder konfigurierte Übertragungskapazität
Throughput	tatsächlich übertragene Datenmenge
Utilization	genutzter Anteil der Kapazität
Packets per Second	Pakete pro Sekunde
Errors	fehlerhafte Frames oder Pakete
Discards/Drops	verworfene Pakete
Retransmissions	erneut übertragene TCP-Segmente
Latency	Laufzeit einer Übertragung
Jitter	Schwankung der Laufzeit
Packet Loss	Anteil verlorener Pakete
Connections	Anzahl aktiver Verbindungen

Windows

```
[RO] Get-NetAdapter |
    Select-Object Name,
                  InterfaceDescription,
                  Status,
                  LinkSpeed,
                  MacAddress
```

Adapterstatistiken:

```
[RO] Get-NetAdapterStatistics
```

Performance Counter:

```
[RO] Get-Counter '\Network Interface(*)\Bytes Total/sec'
```

```
[RO] Get-Counter '\TCPv4\Segments Retransmitted/sec'
```

Linux

```
[RO] ip -s link
```

```
[RO] ss -s
```

```
[RO] cat /proc/net/dev
```

TCP-Statistik:

```
[RO] nstat
```

Treiber- und Adapterstatistiken:

```
[RO][PRIV] sudo ethtool -S eth0
```

macOS

```
[RO] netstat -ib
```

```
[RO] netstat -s
```

```
[RO] ifconfig
```

Zusammenhänge

Beobachtung	Mögliche Untersuchungsrichtung
steigende RX-Errors	Kabel, Transceiver, Port, Duplex oder Hardware
steigende Drops ohne Linkfehler	Puffer, CPU, Treiber oder Überlastung

Beobachtung	Mögliche Untersuchungsrichtung
viele TCP-Retransmissions	Paketverlust, Überlastung oder instabile Verbindung
hohe Latenz ohne Paketverlust	Warteschlangen, Routing oder überlasteter Dienst
Bandbreite dauerhaft nahe Kapazität	Kapazitätsengpass möglich
geringe Bandbreitennutzung und hohe Antwortzeit	Problem möglicherweise in Anwendung oder Zielsystem
nur ein Client betroffen	lokales Interface, WLAN, Treiber oder Clientkonfiguration
alle Clients betroffen	gemeinsamer Pfad, Dienst oder Upstream

Zähler sollten mindestens zweimal gemessen werden. Ein seit dem Systemstart aufgelaufener Fehlerzähler ist ohne zeitliche Änderung nur begrenzt aussagekräftig.

12. Wie werden Latenz, Jitter und Paketverlust gemessen?

Erreichbarkeit und Round-Trip-Time

Aufgabe	Windows	Linux	macOS
vier ICMP-Anfragen	`[TEST] ping ziel.example`	`[TEST] ping -c 4 ziel.example`	`[TEST] ping -c 4 ziel.example`
fortlaufender Ping	`[TEST] ping -t ziel.example`	`[TEST] ping ziel.example`	`[TEST] ping ziel.example`
Route prüfen	`[TEST] tracert ziel.example`	`[TEST] traceroute ziel.example`	`[TEST] traceroute ziel.example`
kombinierte Pfadanalyse	`[TEST] pathping ziel.example`	`[TEST] mtr ziel.example`	`[TEST] mtr ziel.example`

Nicht jedes System beantwortet ICMP-Anfragen. Ein fehlgeschlagener Ping beweist daher nicht, dass der eigentliche Dienst nicht erreichbar ist.

Anwendungsnahe HTTPS-Messung

Windows, Linux und macOS:

```
[TEST] curl -o /dev/null -sS \
  -w 'DNS: %{time_namelookup}\nTCP: %{time_connect}\nTLS: %{time_appconnect}\nTTFB: %{time_starttransfer}\nGesamt: %{time_total}\n' \
  https://example.com/
```

curl-Wert	Bedeutung
`time_namelookup`	Dauer bis zum Abschluss der Namensauflösung
`time_connect`	Dauer bis zur TCP-Verbindung

curl-Wert	Bedeutung
`time_appconnect`	Dauer bis zum Abschluss von TLS
`time_starttransfer`	Zeit bis zum ersten Antwortbyte
`time_total`	Gesamtdauer

Interpretationsbeispiel

```
DNS:    0,010 s
TCP:    0,030 s
TLS:    0,080 s
TTFB:   3,500 s
Gesamt: 3,510 s
```

DNS, TCP und TLS sind schnell. Die lange Zeit bis zum ersten Byte weist eher auf Verarbeitung im Server oder Backend hin.

13. Was bedeutet Sättigung und wie wird sie erkannt?

Auslastung und Sättigung sind nicht dasselbe.

```
Auslastung:
Wie stark wird eine Ressource verwendet?

Sättigung:
Wie viel Arbeit muss warten, weil die Ressource nicht sofort verfügbar ist?
```

Beispiele

Ressource	Auslastung	Sättigung
CPU	CPU-Zeit in Prozent	ausführbare Warteschlange
RAM	belegter Speicher	Speicherdruck und Paging
Datenträger	aktive Zeit	I/O-Warteschlange
Netzwerk	übertragene Bitrate	Drops und Warteschlangen
Datenbank	aktive Verbindungen	wartende Abfragen
Threadpool	aktive Threads	wartende Tasks

Ressource	Auslastung	Sättigung
Connection Pool	belegte Verbindungen	wartende oder abgewiesene Anfragen

Ein System kann noch unter 100 % Auslastung liegen und trotzdem bereits Verzögerungen aufweisen.

Linux Pressure Stall Information

Wenn vom Kernel unterstützt:

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

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

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

Beispiel:

```
some avg10=4.20 avg60=2.10 avg300=0.80 total=1234567
full avg10=1.00 avg60=0.40 avg300=0.10 total=234567
```

Feld	Bedeutung
`some`	mindestens einige Tasks waren durch die Ressource blockiert
`full`	alle nicht untätigen Tasks waren gleichzeitig blockiert
`avg10`	durchschnittlicher Zeitanteil der letzten 10 Sekunden
`avg60`	durchschnittlicher Zeitanteil der letzten 60 Sekunden
`avg300`	durchschnittlicher Zeitanteil der letzten 300 Sekunden
`total`	gesamte Stall-Zeit in Mikrosekunden seit dem Start

PSI misst den Zeitverlust durch Ressourcenknappheit und ergänzt reine Auslastungswerte.

14. Wie werden virtuelle Maschinen und Container richtig bewertet?

Bei virtuellen Systemen existieren mehrere Messebenen:

```
Physischer Host
  → Hypervisor
    → virtuelle Maschine
      → Container
        → Anwendung
```

Ein Wert innerhalb eines Containers zeigt nicht zwingend die vollständige Situation des Hosts.

Zu prüfen

- zugewiesene virtuelle CPUs,
- CPU-Limits und CPU-Quotas,
- Arbeitsspeicherlimit,
- Swap-Konfiguration,
- Hypervisor-Überbelegung,
- Storage-Latenz des Hosts,
- gemeinsam genutzte Netzwerkressourcen,
- Container-Restarts,
- OOM-Beendigungen,
- Ressourcenreservierungen.

Docker-Ressourcen anzeigen

```
[RO] docker stats --no-stream
```

Nur ausgewählte Spalten:

```
[RO] docker stats --no-stream \
  --format 'table
{{.Name}}\t{{.CPUPerc}}\t{{.MemUsage}}\t{{.MemPerc}}\t{{.NetIO}}\t{{.BlockIO}}\t{{.PIDs}}'
```

Containerstatus prüfen:

```
[RO] docker ps -a
```

Konfigurierte Limits eines Containers untersuchen:

```
[RO][SENS] docker inspect beispiel-container
```

OOM-Status prüfen:

```
[RO] docker inspect \
  --format '{{.Name}} OOMKilled={{.State.OOMKilled}} ExitCode={{.State.ExitCode}}' \
  beispiel-container
```

Wichtige Grenzen

- Die CPU-Prozentdarstellung kann sich auf mehrere Kerne beziehen.
- Ein Container kann vom Host gedrosselt werden.
- Host-Caches beeinflussen die wahrgenommene Speicherbelegung.
- Container-I/O kann durch andere Workloads auf demselben Storage beeinflusst werden.
- Ein Neustart setzt bestimmte Anwendungszähler zurück.

15. Wie werden fehlende oder veraltete Messwerte erkannt?

Keine Daten bedeuten nicht automatisch, dass alles in Ordnung ist.

Mögliche Ursachen:

- Zielsystem ist ausgefallen,
- Exporter oder Agent läuft nicht,
- Netzwerkverbindung ist unterbrochen,
- Firewall blockiert die Abfrage,
- Monitoringserver ist gestört,
- Authentifizierung ist fehlgeschlagen,
- Zeitstempel liegen außerhalb des Abfragefensters,
- Metrik wurde umbenannt,
- Ziel wurde aus der Konfiguration entfernt,
- Abfrage ist fehlerhaft,
- Dashboardvariable filtert alle Werte heraus.

Prometheus-Verfügbarkeit eines Targets

```
up
```

Nur nicht erfolgreich abgefragte Targets:

```
up == 0
```

Fehlende Zeitreihe erkennen:

```
absent(up{job="beispiel"})
```

Metriken ohne aktuelle Stichprobe können veraltet sein. Deshalb müssen folgende Zeitpunkte unterschieden werden:

- Zeitpunkt der Messung,
- Zeitpunkt der Übertragung,
- Zeitpunkt der Speicherung,
- Zeitpunkt der Dashboardabfrage,
- Zeitpunkt der Alarmauswertung.

Prüfreihenfolge

1. Zeitbereich des Dashboards prüfen.
2. Zeitpunkt des letzten Datenpunkts prüfen.
3. Target- oder Agentstatus kontrollieren.
4. Datenquelle direkt abfragen.
5. Netzwerkverbindung zwischen Monitoring und Ziel prüfen.
6. Konfigurationsänderungen kontrollieren.
7. Monitoringfehler getrennt vom überwachten Dienst bewerten.

16. Wie werden sinnvolle Schwellenwerte festgelegt?

Ein sinnvoller Schwellenwert basiert auf:

- normalem Verlauf,
- Benutzerwirkung,
- technischer Kapazität,
- Dauer der Überschreitung,
- Wachstumsrate,
- Tages- und Wochenmuster,
- Wartungsfenstern,
- genügend Reaktionszeit,
- Erfahrungen aus früheren Störungen.

Ungeeignete Regel

```
CPU > 80 % → sofort kritischer Alarm
```

Bessere Regelidee

```
CPU-Auslastung über 90 % für mindestens 15 Minuten
UND
Antwortzeit p95 über dem vereinbarten Zielwert
```

Beispiel für mehrstufige Grenzwerte

Zustand	Bedingung	Reaktion
Information	ungewöhnlicher Trend ohne Auswirkung	Dashboard beobachten
Warnung	Grenzwert länger überschritten	während Betriebszeit untersuchen
Kritisch	Benutzerwirkung oder unmittelbarer Ausfall	sofortige Bearbeitung
Kapazitätswarnung	prognostizierte Erschöpfung in 14 Tagen	Kapazität planen

Hysterese

Unterschiedliche Ein- und Ausschaltschwellen verhindern, dass ein Alarm ständig wechselt.

```
Alarm aktivieren: Wert über 90 %
Alarm beenden: Wert unter 80 %
```

Pending Period

Eine Bedingung muss für eine festgelegte Dauer bestehen, bevor der Alarm ausgelöst wird.

```
CPU > 90 % für 10 Minuten
```

Dadurch führen kurze Lastspitzen nicht sofort zu einem Alarm.

17. Was sind Flapping und Alert Fatigue?

Flapping

Ein Alarm wechselt häufig zwischen aktiv und normal:

```
09:00 Alarm
09:01 behoben
```

```
09:02 Alarm
09:03 behoben
```

Mögliche Gegenmaßnahmen:

- längeres Auswertungsfenster,
- Pending Period,
- Hysterese,
- gleitender Durchschnitt,
- sinnvollere Schwellenwerte,
- getrennte Warn- und Kritisch-Stufen.

Alert Fatigue

Zu viele oder nicht relevante Meldungen führen dazu, dass wichtige Alarme übersehen werden.

Typische Ursachen:

- Alarm bei jeder kurzen Lastspitze,
- mehrere Meldungen für dieselbe Ursache,
- fehlende Zuständigkeit,
- keine Handlungsmöglichkeit,
- falsche Priorität,
- keine Wartungsfenster,
- unklare Meldung,
- nicht gepflegte Regeln.

Ein guter Alarm beantwortet:

1. Was ist betroffen?
2. Welche Benutzerwirkung besteht?
3. Seit wann besteht das Problem?
4. Welcher Messwert löste den Alarm aus?
5. Wie lange besteht die Bedingung?
6. Wer ist zuständig?
7. Welches Dashboard und Runbook gehören dazu?
8. Welche ersten Prüfungen sind erforderlich?

Wenn aus einer Meldung keine sinnvolle Handlung folgt, eignet sie sich möglicherweise besser für ein Dashboard als für eine Alarmierung.

18. Wie werden Dashboards bei einer Störung gelesen?

Empfohlene Reihenfolge

1. Zeitfenster auf den Störungszeitraum einstellen.
2. Zeitzone des Dashboards prüfen.
3. Zeitpunkt einer gemeldeten Störung markieren.
4. Benutzerorientierte Signale prüfen:
   - Verfügbarkeit,
   - Antwortzeit,
   - Fehlerrate,
   - Anfragemenge.
5. Technische Ressourcen prüfen:
   - CPU,
   - RAM,
   - Datenträger,
   - Netzwerk.
6. Deployments, Updates und Wartungsereignisse einblenden.
7. Betroffene Instanzen mit funktionierenden Instanzen vergleichen.
8. Zeitfenster vor und nach dem Ereignis betrachten.
9. Rohdaten oder detailliertere Ansicht öffnen.
10. Hypothese anhand von Logs oder Traces prüfen.

Zoomfehler vermeiden

Ein Wert kann je nach gewähltem Zeitraum unterschiedlich wirken:

```
24-Stunden-Ansicht:
kurzer, kaum sichtbarer Ausschlag

5-Minuten-Ansicht:
deutliche Lastspitze von drei Minuten
```

Umgekehrt kann ein extrem kurzes Zeitfenster einen normalen Ausschlag dramatischer erscheinen lassen, als er im Betriebszusammenhang ist.

Zu prüfen

- automatische Aggregation,
- Abfrageintervall,
- Datenauflösung,
- Mittelwert oder Maximum,
- ausgeblendete Datenreihen,
- verwendete Einheit,
- logarithmische oder lineare Achse,
- Beginn der Y-Achse,

- lokale Zeit oder UTC.

19. Wie werden typische Messwertkombinationen interpretiert?

Beobachtung	Mögliche Richtung	Nächster Prüfschritt
CPU hoch, Antwortzeit normal	erwartete Verarbeitung	Kapazitätsreserve und Dauer prüfen
CPU hoch, Antwortzeit hoch	CPU-Engpass möglich	Prozess, Threads und Run Queue prüfen
CPU niedrig, Load hoch	I/O-Wartezustände möglich	`vmstat`, `iostat` und PSI prüfen
RAM belegt, Available ausreichend	möglicherweise normaler Cache	Paging und Speicherdruck prüfen
RAM knapp, Swap steigt	Speicherdruck	Prozesse und Wachstum untersuchen
Datenträger aktiv, Latenz niedrig	hohe, aber verarbeitbare Last	Warteschlange und Trend prüfen
Datenträger aktiv, Latenz hoch	I/O-Engpass möglich	Prozess-I/O und Storage prüfen
Netzwerkdurchsatz hoch, keine Fehler	möglicherweise normale Übertragung	Kapazität und Anwendungskontext
Retransmissions steigen	Paketverlust möglich	Interfacefehler und Pfad prüfen
Antwortzeit hoch, Ressourcen normal	externe Abhängigkeit möglich	Traces, DNS und Backenddienste
Fehlerquote hoch, Traffic normal	Funktions- oder Backendfehler	Logs und Deployments prüfen
Traffic steigt, Fehler und Latenz steigen	Kapazitätsgrenze möglich	Sättigung und Skalierung prüfen
Messwerte verschwinden	Monitoring- oder Zielausfall	Target, Agent und Datenquelle prüfen
nur eine Instanz auffällig	lokales Problem	Konfiguration und Host vergleichen
alle Instanzen gleichzeitig auffällig	gemeinsame Abhängigkeit	Datenbank, Netzwerk und Deployment

Diese Kombinationen sind Ausgangspunkte für Hypothesen und keine automatischen Ursachenfeststellungen.

20. Praxisfall – Anwendung ist zu bestimmten Zeiten langsam

Ausgangslage

Benutzer melden täglich zwischen 02:00 und 02:30 Uhr lange Antwortzeiten.

Vorgehen

1. Benutzerwirkung mit p95- oder p99-Antwortzeit bestätigen.
2. Anfragerate und Fehlerrate im selben Zeitraum prüfen.
3. CPU-, RAM-, Datenträger- und Netzwerkverlauf vergleichen.
4. Geplante Aufgaben und Backups prüfen.
5. Datenbankverbindungen und Abfragedauer kontrollieren.
6. Logs nach Timeouts und Warteschlangen durchsuchen.
7. Mit einem störungsfreien Zeitraum vergleichen.
8. Abhängigkeiten und Storage überprüfen.
9. Ursache durch kontrollierte zeitliche oder technische Änderung testen.
10. Ergebnis über mehrere Tage beobachten.

Beispiel einer Korrelation

```
02:00 Uhr: Backup beginnt
02:02 Uhr: Datenträgerlatenz steigt
02:03 Uhr: Datenbankabfragen werden langsamer
02:04 Uhr: Antwortzeit p95 steigt
02:05 Uhr: erste HTTP-Timeouts
02:30 Uhr: Backup endet
02:32 Uhr: Werte normalisieren sich
```

Diese zeitliche Kette ist ein starker Hinweis, aber die Hypothese muss durch einen kontrollierten Test bestätigt werden.

21. Praxisfall – freier Speicherplatz nimmt kontinuierlich ab

Erforderliche Messwerte

- aktueller freier Speicherplatz,
- Änderungsrate pro Stunde oder Tag,
- betroffene Partition beziehungsweise Volume,
- größte Verzeichnisse,
- Logwachstum,
- temporäre Dateien,
- Datenbank- und Backupwachstum,
- Zeitpunkt der voraussichtlichen Erschöpfung.

Windows

```
[RO] Get-Volume |
    Select-Object DriveLetter,
                  FileSystemLabel,
                  HealthStatus,
                  SizeRemaining,
                  Size
```

Größte Dateien in einem bekannten Untersuchungsverzeichnis:

```
[RO][SENS] Get-ChildItem 'C:\Logs' -File -Recurse -ErrorAction SilentlyContinue |
    Sort-Object Length -Descending |
    Select-Object -First 20 FullName,
        @{Name='SizeMiB';Expression={
            [math]::Round($_.Length / 1MB, 1)
        }}
```

Linux

```
[RO] df -hT
```

```
[RO][PRIV] sudo du -xhd 1 /var | sort -h
```

macOS

```
[RO] df -h
```

```
[RO][PRIV] sudo du -xhd 1 /Library | sort -h
```

Prognosebeispiel

```
Freier Speicher: 100 GB
Verbrauch: 5 GB pro Tag
Vereinfachte Restzeit: ungefähr 20 Tage
```

Die Wachstumsrate kann schwanken. Eine Prognose sollte deshalb auf mehreren Messpunkten und einem geeigneten Zeitraum beruhen.

Dateien dürfen erst gelöscht werden, wenn Zweck, Eigentümer, Aufbewahrungspflicht und Wiederherstellbarkeit geklärt sind.

22. Kompakte Befehlsübersicht für Windows, Linux und macOS

Aufgabe	Windows	Linux	macOS
Leistungsanzeige öffnen	[RO] perfmon.msc	abhängig vom Werkzeug	[RO] open -a "Activity Monitor"
Prozessübersicht	[RO] Get-Process	[RO] top	[RO] top -l 1
CPU messen	[RO] Get-Counter '\Processor(_Total)\% Processor Time'	[RO] mpstat -P ALL 1 10	[RO] top -l 1 -o cpu
Load Average	kein direkt gleichwertiger Standardwert	[RO] uptime	[RO] uptime
logische CPUs	[RO] (Get-CimInstance Win32_ComputerSystem).NumberOfLogicalProcessors	[RO] nproc	[RO] sysctl -n hw.logicalcpu
verfügbarer RAM	[RO] Get-Counter '\Memory\Available MBytes'	[RO] free -h	[RO] memory_pressure
Speicheraktivität	[RO] Get-Counter '\Memory\Pages/sec'	[RO] vmstat 1 10	[RO] vm_stat 1
Prozesse nach RAM	[RO] Get-Process | Sort-Object WorkingSet64 -Descending	[RO] ps -eo pid,%mem,rss,comm --sort=-%mem	[RO] top -l 1 -o mem
Dateisystembelegung	[RO] Get-Volume	[RO] df -hT	[RO] df -h
Inode-Nutzung	nicht direkt vergleichbar	[RO] df -i	[RO] df -i
Datenträger-I/O	[RO] Get-Counter '\PhysicalDisk(*)\% Disk Time'	[RO] iostat -xz 1 10	[RO] iostat -w 1 -c 10
Interfaceübersicht	[RO] Get-NetAdapter	[RO] ip link	[RO] ifconfig
Interfacezähler	[RO] Get-NetAdapterStatistics	[RO] ip -s link	[RO] netstat -ib
TCP-Zusammenfassung	[RO] Get-NetTCPConnection	[RO] ss -s	[RO] netstat -s
Erreichbarkeit	[TEST] ping ziel.example	[TEST] ping -c 4 ziel.example	[TEST] ping -c 4 ziel.example
Pfadprüfung	[TEST] tracert ziel.example	[TEST] traceroute ziel.example	[TEST] traceroute ziel.example
Containerressourcen	[RO] docker stats --no-stream	[RO] docker stats --no-stream	[RO] docker stats --no-stream
Linux-Ressourcendruck	Nicht zutreffend	[RO] cat /proc/pressure/{cpu,memory,io}	Nicht standardmäßig vorhanden

> `mpstat`, `iostat`, `pidstat`, `nstat`, `mtr` und `ethtool` sind nicht auf jeder Linux-Installation standardmäßig vorhanden. Fehlende Werkzeuge dürfen nicht ohne Prüfung der Paketquelle und betriebliche Freigabe installiert werden.

23. Systematischer Ablauf einer Monitoringanalyse

Phase	Vorgehen
1. Symptom bestimmen	Benutzerwirkung und betroffene Funktion dokumentieren
2. Zeitraum festlegen	Fehlerbeginn, Ende und Zeitzone bestimmen
3. Datenqualität prüfen	letzte Messung, Lücken und Datenquelle kontrollieren
4. Baseline wählen	vergleichbaren funktionierenden Zeitraum bestimmen
5. RED prüfen	Anfragerate, Fehlerrate und Dauer untersuchen
6. USE prüfen	Auslastung, Sättigung und Fehler der Ressourcen untersuchen
7. Änderungspunkte prüfen	Updates, Deployments, Backups und Wartung einblenden
8. Umfang bestimmen	einzelne Instanz, Dienstgruppe oder gesamte Umgebung
9. Hypothese bilden	möglichen Zusammenhang konkret formulieren
10. Logs und Traces prüfen	technische Ursache weiter eingrenzen
11. Kontrolliert testen	nur eine begründete Änderung durchführen
12. Wirkung bestätigen	dieselben Messwerte erneut vergleichen
13. Langfristig beobachten	Rückfall und Nebenwirkungen ausschließen
14. Dokumentieren	Ursache, Messwerte, Änderung und Ergebnis festhalten

Formulierung einer guten Hypothese

```
Wenn die erhöhte Datenträgerlatenz die Ursache der langsamen Anwendung ist,
muss die Antwortzeit bei vergleichbarer Anfragemenge mit der
Datenträgerlatenz steigen und nach deren Normalisierung wieder sinken.
```

Diese Hypothese ist messbar und überprüfbar.

24. Dokumentationsvorlage für eine Monitoringanalyse

Ticketnummer:

Analysedatum:

Betroffener Dienst:

Betroffene Systeme:

Benutzerwirkung:

Fehlerbeginn:

Fehlerende:

Zeitzone:

Reproduzierbar: Ja / Nein

Verwendetes Monitoringsystem:

Datenquelle:

Abfrageintervall:

Auswertungsintervall:

Letzter Datenpunkt:

Datenlücken vorhanden: Ja / Nein

Vergleichszeitraum:

Begründung für den Vergleichszeitraum:

Benutzerorientierte Messwerte:

- Verfügbarkeit:
- Anfragerate:
- Fehlerrate:
- Antwortzeit p50:
- Antwortzeit p95:
- Antwortzeit p99:

Ressourcenwerte:

- CPU-Auslastung:
- CPU-Sättigung:
- verfügbarer RAM:
- Paging/Swap:
- Datenträgerauslastung:
- Datenträgerlatenz:
- Datenträgerwarteschlange:
- Netzwerkdurchsatz:
- Netzwerkfehler:

- TCP-Retransmissions:
- freier Speicherplatz:

Änderungen im Zeitraum:
- Deployment:
- Update:
- Backup:
- Wartung:
- Konfigurationsänderung:

Auffällige Korrelationen:
1.
2.
3.

Arbeitshypothese:
Erwartetes Messergebnis:
Kontrollierter Test:
Tatsächliches Ergebnis:

Ermittelte Ursache:
Durchgeführte Änderung:
Rückweg:
Abschlussprüfung:
Beobachtungszeitraum nach der Änderung:

Dashboard:
Abfrage:
Screenshot oder Export:
Sensible Daten redigiert: Ja / Nein

25. Offizielle Quellen und weiterführende Dokumentation

Microsoft

- Get-Counter – Microsoft Learn
- Windows Performance Monitor – Microsoft Learn
- Leistungsprobleme unter Windows untersuchen

- Get-NetAdapterStatistics – Microsoft Learn
- Get-Volume – Microsoft Learn

Linux

- Linux Pressure Stall Information – Kernel-Dokumentation
- Lokale Befehlsreferenzen: `man top`, `man vmstat`, `man iostat`, `man mpstat`, `man free` und `man proc`

Prometheus

- Prometheus-Metriktypen
- PromQL-Funktionen
- Prometheus Alerting Rules
- Prometheus Best Practices

Grafana

- Grafana Alerting
- Best Practices für Grafana-Dashboards
- Umgang mit No Data und Error

Docker

- docker stats – Docker-Dokumentation
- Docker Runtime Metrics

> Bezeichnungen, Einheiten und Berechnungsmethoden können sich zwischen Betriebssystemen, Exportern und Monitoringprodukten unterscheiden. Vor einem direkten Vergleich muss immer geprüft werden, was die konkrete Metrik tatsächlich misst.
