Ressourcenüberwachung & Performance-Analyse in Proxmox (Host) und LXC-Containern (4–7 h)

Einleitung

Bevor wir in die Werkzeuge und Methoden einsteigen, ist es wichtig, die Grundbegriffe zu verstehen:

Ressourcen sind alle Hardware- und Software-Komponenten, die ein System für seine Arbeit benötigt. Dazu zählen insbesondere CPU (Rechenzeit), RAM (Speicherplatz), Festplatten/I/O-Systeme (Datenspeicher und Zugriffsgeschwindigkeit) sowie Netzwerk-Bandbreite. Diese Ressourcen sind begrenzt und werden von verschiedenen Prozessen und Containern gemeinsam genutzt.

Host vs. Container: Während der **Proxmox Host** die gesamte Hardware und die Summe aller Containerprozesse abbildet, zeigt ein **LXC-Container** nur die Ressourcen an, die ihm von cgroups zugeteilt sind. Container können also Engpässe erleben, auch wenn der Host noch Reserven hat – oder umgekehrt.

Ressourcenüberwachung bezeichnet die systematische Erfassung und Beobachtung dieser Kennzahlen im laufenden Betrieb. Ziel ist es, ein Bild der aktuellen Auslastung zu bekommen und festzustellen, welche Prozesse oder Container welche Ressourcen in welchem Umfang beanspruchen.

Performance-Analyse geht einen Schritt weiter: Hierbei werden die erhobenen Daten interpretiert, um Bottlenecks (Engpässe) zu identifizieren und die Ursachen für Leistungseinbußen zu verstehen. Dies ermöglicht es, fundierte Maßnahmen zur Optimierung einzuleiten – etwa das Anpassen von Limits, die Verlagerung von Workloads oder die Aufrüstung der Hardware.

Kurz gesagt: Ressourcenüberwachung sagt uns, *was* passiert – Performance-Analyse zeigt uns, *warum* es passiert und wie wir darauf reagieren können.

> **Ziel:** Überwachung von CPU, RAM, Netzwerk-I/O und Festplattennutzung **im Proxmox Host** und **in einzelnen LXC-Containern**, Interpretation der wichtigsten Metriken und Identifikation von Performance-Engpässen.
>
> **Werkzeuge:** `htop`, `nmon`, `iostat` (sysstat), `dstat`, `ss`
>
> **Umgebung:** Proxmox/LXC auf Linux (z. B. Ubuntu 22.04/24.04), Container wahlweise Debian/Ubuntu.
>
> **Vorkenntnisse:** Linux-CLI, grundlegende LXC-Befehle, sudo-Rechte.

---

1. Didaktische Übersicht

Baustein	Inhalte	Zeit	Ergebnis
Einführung	Monitoring-Grundlagen, Host vs. Container, cgroups v2, PSI	30–45 min	Gemeinsames Begriffsverständnis
CPU & RAM (htop)	Prozesse, Threads, Cgroups, Sortierung, Filter	45–60 min	Live-Sicht Host/Container

Baustein	Inhalte	Zeit	Ergebnis
Systemübersicht (nmon)	Interaktives Dashboard, CSV-Recording	45–60 min	Überblick Host/Container
Storage (iostat)	Durchsatz, IOPS, Latenzen, %util, Interpretation	45–60 min	I/O-Engpässe Host/Container
Kombi-Monitoring (dstat)	Kombinierte Metriken + Top-Verursacher	30–45 min	Korrelationen Host/Container
Netzwerk (ss)	Sockets, Zustände, Backlogs, Retransmits	30–45 min	Netz-Bottlenecks deuten
Labor & Fallstudien	Last erzeugen, messen, diagnostizieren	60–120 min	Praxisroutine
Reporting	Befund & Maßnahmen ableiten; Kurz-Report	20–30 min	Kurzreport erstellen

2. Grundlagen: Host vs. Container

- **Host (Proxmox Node)**: Gesamtsicht aller Ressourcen und Container. Hier wird sichtbar, wie stark die physische Hardware ausgelastet ist. Engpässe hier wirken sich auf *alle* Container aus.
- **Container (LXC)**: Sieht nur die ihm zugewiesenen Ressourcen (cgroups). Ein Container kann an seine CPU/RAM-Limits stoßen, während der Host noch freie Kapazitäten hat.
- **cgroups v2**: Kontrolliert CPU-Shares, RAM-Limits, I/O-Prioritäten.
- **PSI (Pressure Stall Information)**: Zeigt, wie stark Prozesse auf CPU, RAM oder I/O warten müssen – besonders relevant auf Hostebene.

Messorte:

- **Hostebene:** Gesamtsicht über alle Container und VMs.
- **Containerebene:** Nur die Prozesse des jeweiligen Containers.

---

3. CPU & RAM live beobachten mit `htop`

3.1 Auf dem **Host**

- `sudo apt install htop`
- `htop`
- Interpretation:
  - Containerprozesse erscheinen mit dem Host-Kernel.
  - Gesamtauslastung über alle Kerne sichtbar.
  - **Dauerhaft >80 % pro Kern** → Host-CPU am Limit.

3.2 Im **Container**

- Installation: `sudo apt install htop`
- Anzeige zeigt nur Prozesse des Containers.
- Interpretation:
  - Hohe CPU-Last → Container stößt an CPU-Limit (auch wenn Host frei ist).
  - Hoher SWAP-Einsatz → Container-RAM-Limit erreicht.

---

4. Systemüberblick & Recording mit `nmon`

4.1 Auf dem **Host**

- `nmon` starten → Übersicht über alle Container zusammen.
- CPU-Übersicht zeigt Gesamtauslastung des Hosts.
- Disk- und Network-Panels zeigen kumulierte Host-Last.
- Recording: `nmon -f -s 10 -c 60` → für Trendanalysen.

4.2 Im **Container**

- `nmon` zeigt nur die Containerprozesse.
- Network & Disk nur soweit sichtbar, wie Container Zugriff hat.
- Gut geeignet, um Workload des Containers selbst zu überwachen.

---

5. Storage-Analyse mit `iostat` (sysstat)

5.1 Auf dem **Host**

- `iostat -xz 2`
- Interpretation:
  - `%util ~100 %` → physisches Device am Limit.
  - Hohe `await` (>20 ms bei SSDs) → Storage-Flaschenhals.
- Alle Container teilen sich dieselben physischen Geräte → Hostwerte entscheidend.

5.2 Im **Container**

- Container sieht i. d. R. **keine echten Blockgeräte**, sondern nur virtuelle Mounts.
- Aussagekraft begrenzt – I/O-Engpässe immer auf Host analysieren!

---

6. Kombi-Monitoring mit `dstat`

6.1 Auf dem **Host**

- `dstat -c -m -n -d --top-cpu --top-mem`
- Kombination aus CPU, RAM, Netzwerk, Disk.
- Top-Verursacher über alle Container sichtbar.

6.2 Im **Container**

- Zeigt nur eigene Prozesse.
- Hilfreich, um Workload innerhalb des Containers zu verstehen.

---

7. Netzwerk-Analyse mit `ss`

7.1 Auf dem **Host**

- `ss -s` → Gesamtsicht aller Netzwerkverbindungen.
- `ss -tlpn` → Prozesse mit offenen Ports (inkl. Container-Prozesse).
- Interpretation:
  - **Retransmits** → mögliche Hardware/Netzwerkprobleme.
  - **Backlogs** → Host oder Container-Applikation zu langsam.

7.2 Im **Container**

- Zeigt nur Verbindungen, die im Container existieren.
- Gut für Applikationsdiagnose (z. B. Webserver-Last).

---

8. Praxislabore & Fallstudien

- **Labor A: CPU-Stress**
  - Tool: `stress-ng --cpu 2 --timeout 60s`
  - Host: sieht die Gesamtauslastung steigen.
  - Container: sieht nur seine eigenen Lastspitzen.
- **Labor B: RAM-Stress**
  - `stress-ng --vm 2 --vm-bytes 512M --timeout 60s`
  - Container stößt an sein RAM-Limit → Host zeigt Swap nicht unbedingt an.
- **Labor C: Disk-I/O**
  - `dd if=/dev/zero of=testfile bs=1M count=2000 oflag=dsync`
  - Host: erkennt I/O-Last, Engpass sichtbar.
  - Container: sieht nur eigene `dd`-Operation.
- **Labor D: Netzwerk-Traffic**
  - `iperf3 -s` / `iperf3 -c <IP>`
  - Host: sieht Gesamtlast im Netzwerkinterface.
  - Container: misst nur eigene Bandbreite.

---

9. Reporting

1. **System & Kontext erfassen**
    - Host/Node, Containername, Workload, Zeitraum.
2. **Kernaussagen aus Messungen**
    - Host: Gesamtressourcen, überlastete Devices.
    - Container: Limit-Überschreitungen, Prozesslast.
3. **Engpässe identifizieren**
    - Host-Engpass = betrifft alle.
    - Container-Engpass = betrifft nur diesen Container.
4. **Empfohlene Maßnahmen**
    - Host: Hardware aufrüsten, Workload verteilen.
    - Container: Limits erhöhen, Applikation optimieren.
5. **Belege anfügen**
    - Screenshots, Tool-Outputs.

---

10. Diagnose-Muster & Maßnahmen (Cheatsheet)

- **Host CPU hoch, Load hoch, iowait niedrig** → Gesamte Rechenlast hoch, evtl. mehr CPU nötig.
- **Container CPU hoch, Host CPU frei** → Container-Limit zu niedrig.
- **Host RAM voll, Swap hoch** → physischer RAM Engpass.
- **Container RAM voll, Swap hoch** → Container-Limit erreicht.
- **Host I/O await hoch** → Storage zu langsam.
- **Netzwerk Retransmits auf Host** → physische Netzwerkprobleme.
- **Netzwerkprobleme nur im Container** → Container-App/Config prüfen.

---

11. Best Practices

- Monitoring **immer auf beiden Ebenen** durchführen.
- Hostwerte → Hardware- und Gesamtkapazität.
- Containerwerte → Workload-Sicht der Applikation.
- Historische Daten (Grafana/Prometheus) sammeln.
- Alerts für Host und Container definieren.

---

12. Quick Reference (Befehle)

```
# Hostebene
iostat -xz 2
dstat -c -m -n -d --top-cpu --top-mem
ss -s
```

```
# Containerebene
htop
nmon
ss -tlpn
```

Abschluss

Mit den Tools `htop`, `nmon`, `iostat`, `dstat` und `ss` lässt sich zwischen **Hostebene (Proxmox Node)** und **Containerebene (LXC)** klar unterscheiden. Nur so kann man Lastspitzen und Engpässe korrekt zuordnen und entscheiden, ob Maßnahmen im Container (Limits, Applikationstuning) oder am Host (Hardware, Gesamtkapazität) erforderlich sind.

Lizenz