Process Speicherüberwachung

Beschreibung:

Ein lokal check der alle Prozesses nach Speichergröße sortiert.
Auf Grund der Prozessmenge auf 20 Stück Limitiert.
Bei diesem Check gehts auch nur darum die Speicherfresser zu finden.
Die Schwellwerte WARN und CRIT können über die Variablen WARN und CRIT gesetzt werden. Die Einheit ist MB

Das Script

Einmal als Einzelauflistung:

```
#!/usr/bin/python3
import subprocess
import re

# Schwellenwerte in Megabyte
WARN = 1000  # Beispiel: 1000 MB
CRIT = 2000  # Beispiel: 2000 MB
COUNT_PROC = 20  # Anzahl der Prozesse, die ausgeben werden sollen - maximal 20

# Programmvariablen, bitte nicht verändern
output = "P \"Speicherverbrauch der Prozesse in MB\" "

# Führe den `ps`-Befehl aus, um Prozessname und Speicherverbrauch (RSS) zu bekommen
process = subprocess.Popen(['ps', '-eo', 'rss,comm', '--sort=-rss'], stdout=subprocess.PIPE)
stdout = process.communicate()[0]

# Verarbeite jede Zeile der Ausgabe
for i, line in enumerate(stdout.decode('utf-8').strip().split('\n')[1:]):  # Überspringe die Kopfzeile
```

```
    if i >= COUNT_PROC:  # Limit auf COUNT_PROC Prozesse
        break

    parts = line.split(None, 1)
    memory = int(parts[0]) // 1000  # Konvertiere RSS von KB zu MB
    process = re.sub('[^a-zA-Z0-9]', '_', parts[1])  # Ersetze alle nicht alphanumerischen Zeichen durch
Unterstriche

    # Hinzufügen der Performance-Daten für jeden Prozess zum Output
    output += f"NR-{i:02d}-{process}={memory};{WARN};{CRIT}|"

# Entferne das letzte Semikolon
output = output.rstrip('|')

# Entferne Kommas und Unterstriche (wenn nötig)
output = output.replace(',', '').replace('_', '')

# Ausgabe des gesammelten Outputs
print(output)
```

Ausgabe:

```
P "Speicherverbrauch der Prozesse in MB" NR-00-kvm=67338;1000;2000|NR-01-kvm=4382;1000;2000|NR-02-
kvm=4307;1000;2000|NR-03-cephosd=3305;1000;2000|NR-04-cephosd=3099;1000;2000|NR-05-
cephosd=2875;1000;2000|NR-06-cephosd=2679;1000;2000|NR-07-kvm=1879;1000;2000|NR-08-
kvm=1264;1000;2000|NR-09-kvm=916;1000;2000|NR-10-java=576;1000;2000|NR-11-
cephmon=491;1000;2000|NR-12-kvm=436;1000;2000|NR-13-kvm=420;1000;2000|NR-14-
cephmgr=325;1000;2000|NR-15-mongod=167;1000;2000|NR-16-corosync=165;1000;2000|NR-17-
pveproxy=157;1000;2000|NR-18-launcher=154;1000;2000|NR-19-pveproxy=144;1000;2000
```

Als Gesamtspeicherzusammenfassung:

```
#!/usr/bin/python3
import subprocess
import re

# Schwellenwerte in Megabyte
WARN = 1000  # Beispiel: 1000 MB
CRIT = 2000  # Beispiel: 2000 MB
COUNT_PROC = 20  # Anzahl der Prozesse, die ausgegeben werden sollen - maximal 20
```

```
# Programmvariablen, bitte nicht verändern
output = "P \"Speicherverbrauch der Prozesse in MB\" "

# Führe den `ps`-Befehl aus, um Prozessname und Speicherverbrauch (RSS) zu bekommen
process = subprocess.Popen(['ps', '-eo', 'rss,comm', '--sort=-rss'], stdout=subprocess.PIPE)
stdout = process.communicate()[0]

# Speichere Speicherverbrauch pro Prozessname in einem Wörterbuch
memory_usage = {}

# Verarbeite jede Zeile der Ausgabe
for line in stdout.decode('utf-8').strip().split('\n')[1:]:  # Überspringe die Kopfzeile
    parts = line.split(None, 1)
    memory = int(parts[0]) // 1000  # Konvertiere RSS von KB zu MB
    process_name = re.sub('[^a-zA-Z0-9]', '_', parts[1])  # Ersetze alle nicht alphanumerischen Zeichen durch
Unterstriche

    # Addiere den Speicherverbrauch für gleiche Prozessnamen
    if process_name in memory_usage:
        memory_usage[process_name] += memory
    else:
        memory_usage[process_name] = memory

# Sortiere Prozesse nach Speicherverbrauch, beschränke auf COUNT_PROC Einträge
sorted_processes = sorted(memory_usage.items(), key=lambda x: x[1], reverse=True)[:COUNT_PROC]

# Generiere die Ausgabezeile für die Prozesse
for i, (process, mem) in enumerate(sorted_processes):
    output += f"{process}={mem};{WARN};{CRIT}|"

# Entferne das letzte Semikolon
output = output.rstrip('|')

# Ausgabe des gesammelten Outputs
print(output)
```

Ausgabe:

```
P "Speicherverbrauch der Prozesse in MB"
kvm=80987;1000;2000|ceph_osd=12535;1000;2000|java=588;1000;2000|ceph_mon=485;1000;2000|pvedaemon_worke=423;1000;2000|pveproxy_worker=423;1000;2000|ceph_mgr=325;1000;2000|corosync=165;1000;2000|mongod=165;1000;2000|pveproxy=157;1000;2000|launcher=154;1000;2000|pvedaemon=136;1000;2000|pvescheduler=113;1000;2000|pve_ha_crm=111;1000;2000
```

Installation

Auf dem zu überwachenden Server nach

```
nano /usr/lib/check_mk_agent/local/process.py
chmod +x /usr/lib/check_mk_agent/local/process.py
```

kopieren oder editieren über einfügen

Version #9
Erstellt: 10 April 2024 06:18:29 von Admin
Zuletzt aktualisiert: 16 April 2024 08:15:05 von Admin