CPU-Statistiken mit "PERF" erfassen und grafisch darstellen Tool im NSO

Inhalt

Einleitung

In diesem Dokument wird beschrieben, wie Sie das Perf-Tool auf NSO-Hosts verwenden, um Leistungsprobleme zu untersuchen.

Voraussetzungen

Anforderungen

Cisco empfiehlt, dass Sie über Kenntnisse in folgenden Bereichen verfügen:

- Grundlegende Linux-/Unix-Befehlszeilenverwendung
- NSO-Systemarchitektur und -betrieb (Network Services Orchestrator)
- Konzepte für CPU-Profilerstellung und -Analyse
- Vertrautheit mit Leistungsproblem-Problembehebungs-Workflows

Verwendete Komponenten

Die Informationen in diesem Dokument basierend auf folgenden Software- und Hardware-Versionen:

- NSO-System oder lokale Installation auf einem unterstützten Unix/Linux-Host
- Linux Distributionen wie Ubuntu, Debian, Fedora oder RedHat Derivate
- perf-Tool (Linux Performance Analysis Tool)

Die Informationen in diesem Dokument beziehen sich auf Geräte in einer speziell eingerichteten

Testumgebung. Alle Geräte, die in diesem Dokument benutzt wurden, begannen mit einer gelöschten (Nichterfüllungs) Konfiguration. Wenn Ihr Netzwerk in Betrieb ist, stellen Sie sicher, dass Sie die möglichen Auswirkungen aller Befehle kennen.

Hintergrundinformationen

Perf ist ein leistungsstarkes Tool zur Leistungsanalyse in Linux, das hauptsächlich für die CPU-Profilerstellung verwendet wird. Es bietet Einblicke in die aktuelle Arbeitsweise der CPU, indem es die Last untergeordneter Funktionen erfasst und analysiert. Dies hilft bei der Identifizierung von Funktionen oder Prozessen, die die CPU belegen, und ist für die Ermittlung von Performance-Engpässen unerlässlich.

Perf kann auch Flammendiagramme erzeugen, die spezielle Diagramme sind, die visuell darstellen, welche Teile eines Programms die meiste CPU-Zeit verbrauchen. Flammendiagramme erleichtern das Auffinden von Codebereichen, die optimiert werden müssen.

Wichtig ist, dass die Leistung entsprechend den Empfehlungen der Geschäftseinheit des NSO auch in der Hauptprüfliste für die Datensammlung bei "Out of Memory" (OOM)-Fällen enthalten ist. Detaillierte Anleitungen zur Fehlerbehebung bei OM-Problemen erhalten Sie beim Cisco TAC.

Fehlerbehebung bei NSO-Leistungsproblemen

In diesem Abschnitt wird ein umfassender Workflow für die Installation, Verwendung und Analyse von Daten aus dem perf-Tool auf NSO-Hosts beschrieben, um Leistungsprobleme zu beheben.

PERF installieren

Schritt 1: Installieren Sie perf auf Ihrer Linux-Distribution. Verwenden Sie den entsprechenden Befehl für Ihr Betriebssystem:

Für Ubuntu:

```
apt-get update && apt-get -y install linux-tools-generic
```

Für Debian:

```
apt-get update && apt-get -y install linux-perf
```

Für Fedora/RedHat-Derivate:

```
dnf install -y perf
```

Weitere Informationen zu bekannten Vorbehalten bei der Installation von PERF erhalten Sie vom Cisco TAC-Team.

Abfragen der Daten

Schritt 1: Identifizieren Sie den Hauptprozess des NSO.

Verwenden Sie den unten angegebenen Befehl, um den NSO-Prozess (ncs.smp) zu finden:

```
ps -ef | grep ncs\.smp
```

Beispiel:

```
root    120829       1  16 13:23 ? 00:11:08 /opt/ncs/current/lib/ncs/erts/bin/ncs.smp -K true -P 277140
root    121424   120604  0 14:30 pts/0 00:00:00 grep --color=auto ncs.smp
```

Phase 2: Alternativ müssen Sie die PID des mit dem NSO verknüpften Java-Hauptprozesses verwenden, insbesondere wenn Sie sich auf Java-Operationen konzentrieren. Ausgeführt:

```
ps -ef | grep NcsJVMLauncher
```

Beispiel:

```
root    120903   120833  6 13:32 ? 00:03:40 java -classpath :/opt/ncs/current/java/jar/* -Dhost=127.0.0
root    121435   120604  0 14:33 pts/0 00:00:00 grep --color=auto NcsJVMLauncher
```

Schritt 3: Führen Sie den problematischen Test oder Anwendungsfall aus, um das Leistungsszenario zu validieren.

Schritt 4: Führen Sie in einem anderen Terminal-Fenster PERF für die entsprechenden Prozess-IDs (PIDs) aus. Verwenden Sie das unten angegebene Befehlsformat, und ersetzen Sie XX,YY,ZZ durch die oben erhaltenen PIDs:

```
perf record -F 100 -g -p XX,YY,ZZ
```

So erstellen Sie beispielsweise systemweite Profile und erfassen Anrufsdiagramme mit 99 Hz für bestimmte PIDs:

```
perf record -a -g -F 99 -p 120829,120903
```

Beispiel:

```
Warning:
PID/TID switch overriding SYSTEM
```

Optionsbeschreibungen:

- -a: Alle CPUs systemweite Erfassung aller CPUs (Standard, wenn kein Ziel angegeben ist).
- g: Aufrufdiagramme erfassen (Stack Traces). Gibt an, wo Funktionen aufgerufen werden.
- F: Abtastfrequenz in Hz. Höhere Frequenzen erhöhen die Präzision, führen jedoch zu mehr Overhead.
- S: Gibt die Prozess-ID(s) an.

Schritt 5: Wenn Sie alle Samples gesammelt haben, beenden Sie perf mit Strg+C:

```
^C
[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 0.646 MB perf.data (4365 samples) ]
```

Im aktuellen Verzeichnis wird nun die Datei perf.data angezeigt.

Schritt 6: Generieren Sie mit dem folgenden Befehl einen zusammenfassenden Bericht:

```
perf report -n --stdio > perf_report.txt
```

Optionsbeschreibungen:

- n: Symbole ohne Gruppierung anzeigen (flache Ansicht).
- —stadio: Erzwingen Sie die Ausgabe auf die Standardausgabe (Terminal).

An dieser Stelle müssen Sie beide Dateien (perf.data und perf_report.txt) speichern und für Ihren Supportkontakt freigeben, bevor Sie mit der weiteren Analyse fortfahren können.

Wenn die Erfassung erfolgreich war, zeigt perf_report.txt eine baumartige Struktur an, die ein

hierarchisches Aufrufdiagramm darstellt. Mithilfe von Prozentsätzen können Sie Hotspots identifizieren, in denen die meiste CPU-Zeit aufgewendet wird.

Beispielauszug:

```
# Children      Self       Samples  Command          Shared Object         Symbol
# ........  ........  ............  ...............  ....................  ............................
#    30.61%     0.00%             0  C2 CompilerThre  libc.so.6             [.] start_thread
#            ---start_thread
#               thread_native_entry(Thread*)
#               Thread::call_run()
#               JavaThread::thread_main_inner()
#               CompileBroker::compiler_thread_loop()
#                           --30.58%--CompileBroker::invoke_compiler_on_method(CompileTask*)
#                                     --30.47%--C2Compiler::compile_method(ciEnv*, ciMethod*, int, bool
#                                     Compile::Compile(ciEnv*, ciMethod*, int, bool, bool, bool, bool,
#                                               |--17.57%--Compile::Code_Gen()
#                                               |          |--12.46%--PhaseChaitin::Register_Allocate()
#                                               |          |          |--2.79%--PhaseChaitin::build_ifg_
#                                               |          |          |           --1.05%--PhaseChaitin
#                                               |          |--1.49%--PhaseChaitin::Split(unsigned int,
#                                               |          |--1.26%--PhaseChaitin::post_allocate_copy_r
```

Dolmetschen:

- Prozess/Thread: Der C2-CompilerThree-Thread wird analysiert.
- Gesamte CPU-Auslastung: Dieser Thread ist für 30,61% der CPU-Zeit verantwortlich.
- Funktionsablauf: Der Thread beginnt mit `start_thread`, und Delegaten arbeiten über mehrere Ebenen hinweg. Der Großteil der CPU-Zeit (30,47 %) wird in `C2Compiler::compile_method` verbracht, was auf einen potenziellen Hotspot hinweist.

Flammendiagramm erstellen

Schritt 1: Generieren Sie eine Leistungsabtastung aller CPUs und Prozesse über ein definiertes Intervall (z. B. 60 Sekunden):

```
perf record -a -g -F 99 sleep 60
```

Beispiel:

```
[ perf record: Woken up 32 times to write data ]
[ perf record: Captured and wrote 10.417 MB perf.data (67204 samples) ]
```

Phase 2: Kopieren oder übertragen Sie diese perf.data Datei auf einen Host, von dem Sie das

flamegraph Template Repository herunterladen können.

Schritt 3: Konvertieren Sie die Datei perf.data in ein Textformat:

```
perf script > data.perf
```

Schritt 4: Klonen Sie das FlameGraph GitHub-Repository, und platzieren Sie data.perf in diesem Verzeichnis:

```
cp data.perf $PWD/FlameGraph/.
```

Schritt 5: Die Stapelspuren für die Flammengrafik-Verarbeitung reduzieren:

<#root>

```
cat data.perf | ./stackcollapse-perf.pl > data.perf-folded
```

Schritt 6: Flammendiagramm SVG-Datei erzeugen:

<#root>

```
./flamegraph.pl data.perf-folded > data.svg
```

Anmerkung: Wenn Sie den Fehler "can not locate open.pm in @INC" auf CentOS oder RHEL feststellen, installieren Sie das erforderliche Perl-Modul:

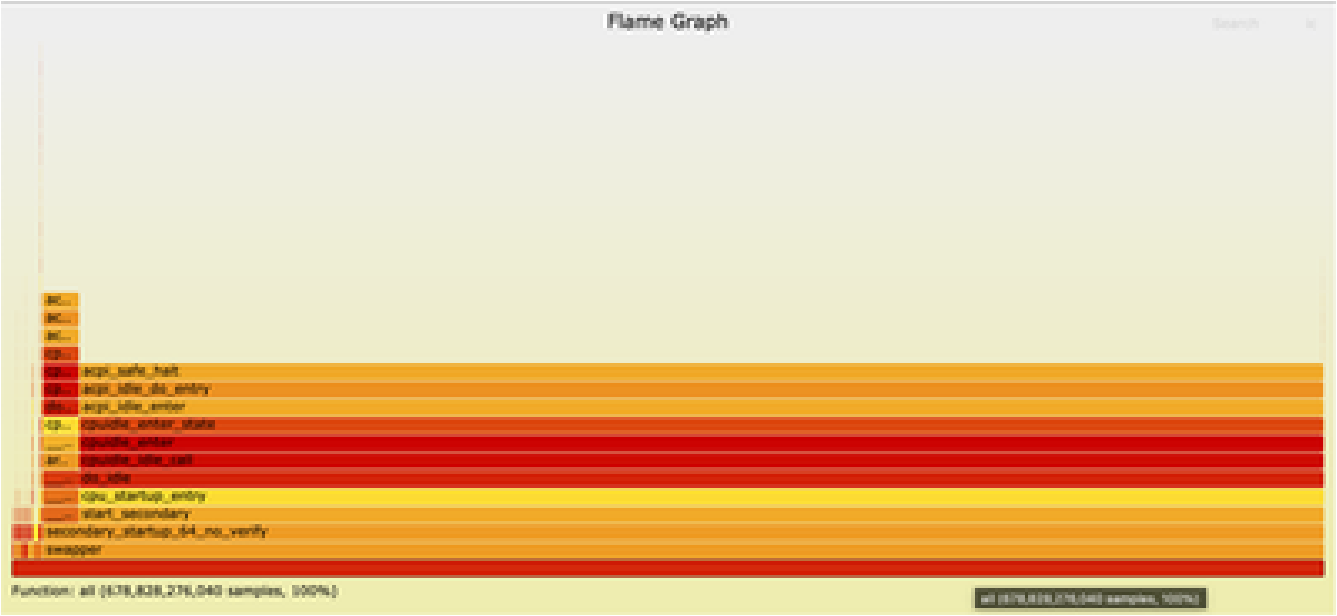
```
yum install perl-open.noarch
```

Schritt 7: Öffnen Sie die Datei data.svg in Ihrem bevorzugten Webbrowser, um das Flammendiagramm darzustellen.

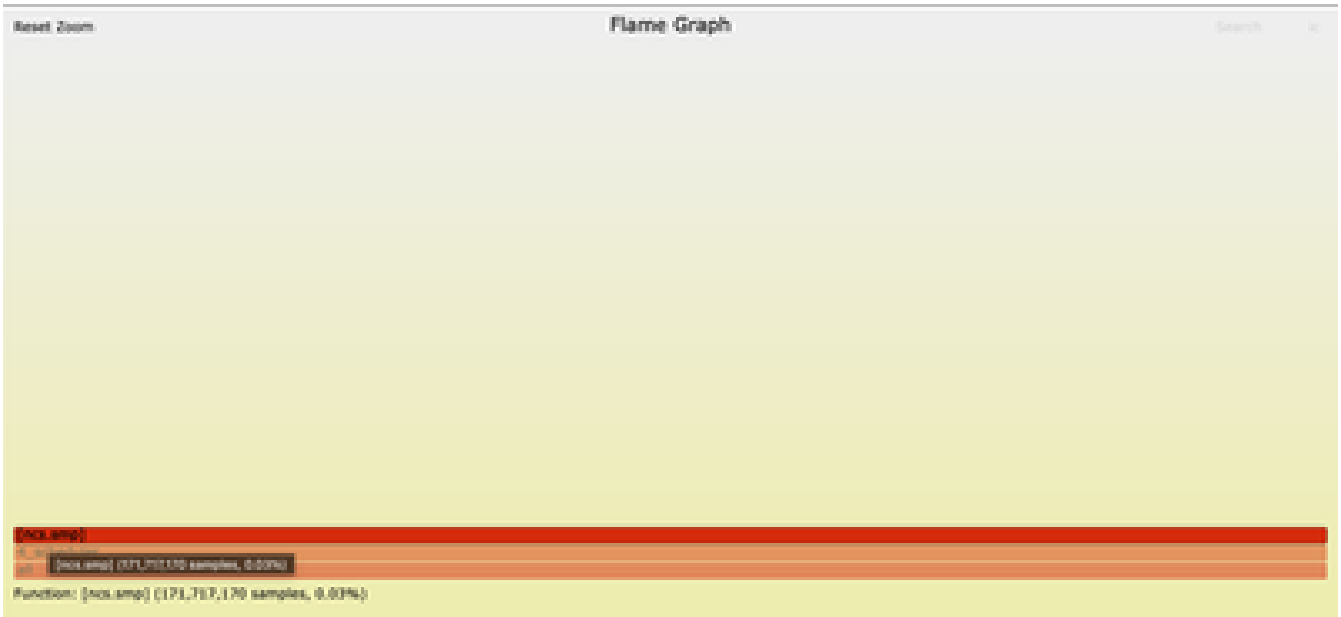
Durchsuchen des Flammendiagramms

Sobald die Flammendiagrammdatei in Ihrem Browser geöffnet ist, können Sie mit ihr interagieren, indem Sie auf ein beliebiges Kästchen klicken, um die Funktion und die Aufrufliste zu vergrößern. Die Länge jedes Felds gibt die CPU-Zeit an, die in dieser Funktion und ihrem Aufrufstack

verbracht wurde. Diese Visualisierung macht es einfach, Hotspots und Bereiche für eine Optimierung zu identifizieren.



Vergrößert ncs.smp:



Zugehörige Informationen

- Bekannte Vorbehalte zu Linux
- Technischer Support und Downloads von Cisco

Informationen zu dieser Übersetzung

Cisco hat dieses Dokument maschinell übersetzen und von einem menschlichen Übersetzer editieren und korrigieren lassen, um unseren Benutzern auf der ganzen Welt Support-Inhalte in ihrer eigenen Sprache zu bieten. Bitte beachten Sie, dass selbst die beste maschinelle Übersetzung nicht so genau ist wie eine von einem professionellen Übersetzer angefertigte. Cisco Systems, Inc. übernimmt keine Haftung für die Richtigkeit dieser Übersetzungen und empfiehlt, immer das englische Originaldokument (siehe bereitgestellter Link) heranzuziehen.