# 使用"PERF"；收集和繪製CPU統計資訊NSO中的工具

## 目錄



## 簡介

本文檔介紹如何在NSO主機上使用perf工具調查效能問題。

## 必要條件

### 需求

思科建議您瞭解以下主題：

- 基本Linux/Unix命令列使用
- NSO(Network Services Orchestrator)系統體系結構和操作
- CPU分析和分析概念
- 熟悉效能故障排除工作流程

### 採用元件

本文中的資訊係根據以下軟體和硬體版本：

- 在受支援的Unix/Linux主機上的NSO系統或本地安裝
- Linux發行版，如Ubuntu、Debian、Fedora或RedHat衍生版
- perf工具（Linux效能分析工具）

本文中的資訊是根據特定實驗室環境內的裝置所建立。文中使用到的所有裝置皆從已清除（預設）的組態來啟動。如果您的網路運作中，請確保您瞭解任何指令可能造成的影響。

# 背景資訊

Perf是Linux中功能強大的效能分析工具，主要用於分析CPU。它通過捕獲和分析低級功能的負載來深入瞭解CPU當前的工作方式。這有助於確定哪些功能或進程佔用CPU，這對於確定效能瓶頸至關重要。

Perf還可以生成火焰圖，這是一種特殊的圖表，可以直觀地表示程式的哪些部分佔用了最多的CPU時間。火焰圖可以更輕鬆地找出代碼中需要最佳化的區域。

重要的是，效能還會按照NSO業務部門(BU)的建議，包含在記憶體不足(OOM)案例的主要資料收集核對清單中。 有關OOM故障排除的更多詳細說明，請聯絡Cisco TAC。

# 排除NSO效能問題的效能使用故障

本節提供在NSO主機上安裝、使用和分析來自perf工具的資料以排除效能問題的綜合工作流。

## 安裝效能

步驟 1:在Linux發行版上安裝perf。對您的作業系統使用適當的命令：

對於Ubuntu:

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

對於Debian:

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

對於Fedora/RedHat衍生工具：

```
dnf install -y perf
```

有關安裝perf時的已知警告的詳細資訊，請聯絡Cisco TAC團隊。

## 對資料進行取樣

步驟 1:確定主NSO流程。

使用以下命令查詢NSO進程(ncs.smp):

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

輸出示例：

```
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
```

步驟 2:或者，您必須使用與NSO關聯的主要Java進程的PID，尤其是當重點關注Java操作時。執行：

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

輸出示例：

```
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
```

步驟 3:執行有問題的測試用例或使用例以驗證效能方案。

步驟 4:在不同的終端視窗中，根據相關進程ID(PID)運行perf。 使用以下給定的命令格式，用上面獲得的PID替換XX、YY、ZZ:

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

例如，要分析系統範圍並收集特定PID的99Hz呼叫圖：

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

輸出示例：

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

選項說明：

- -a:全CPU;從所有CPU進行系統範圍的收集（如果未指定目標，則為預設值）。
- -g:捕獲呼叫圖（堆疊跟蹤）。 標識函式被呼叫的位置。
- -F:取樣頻率(Hz)。較高的頻率可提高精度，但會增加開銷。
- -p:指定進程ID。

步驟 5:收集完樣本後，使用Ctrl+C停止執行操作：

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

您現在可以在當前目錄中看到perf.data檔案。

步驟 6:使用以下命令生成摘要報告：

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

選項說明：

- -n:顯示不帶分組的符號（平面檢視）。
- -stdio:強制輸出到標準輸出（終端）。

此時，您必須先儲存這兩個檔案(perf.data和perf_report.txt)，並將其與支援聯絡人共用，然後再繼續進行進一步分析。

如果捕獲成功，perf_report.txt將顯示表示分層呼叫圖的樹狀結構。百分比有助於確定佔用大部分CPU時間的熱點。

示例摘錄：

```
# 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, B
#                                                    |          |--1.26%--PhaseChaitin::post_allocate_copy_re
```

解釋：

- 進程/執行緒：正在分析C2 CompilerThre執行緒。
- CPU總使用率：此執行緒佔用CPU時間的30.61%。
- 函式流：該執行緒以`start_thread`開頭，並跨多個層委託工作。大部分CPU時間(30.47%)花在`C2Compiler::compile_method`中，表示潛在的熱點。

## 生成火焰圖形

步驟 1:在定義的時間間隔（例如60秒）內從所有CPU和進程生成效能示例：

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

輸出示例：

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

步驟 2:將此perf.data檔案複製或傳輸到可從其下載Flamegraph模板儲存庫的主機。

步驟 3:將perf.data檔案轉換為文本格式：

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

步驟 4:克隆FlameGraph GitHub儲存庫並將data.perf放入此目錄：

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

步驟 5:摺疊用於火焰圖處理的堆疊跟蹤：

<#root>

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

步驟 6:生成火焰圖SVG檔案：

<#root>

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

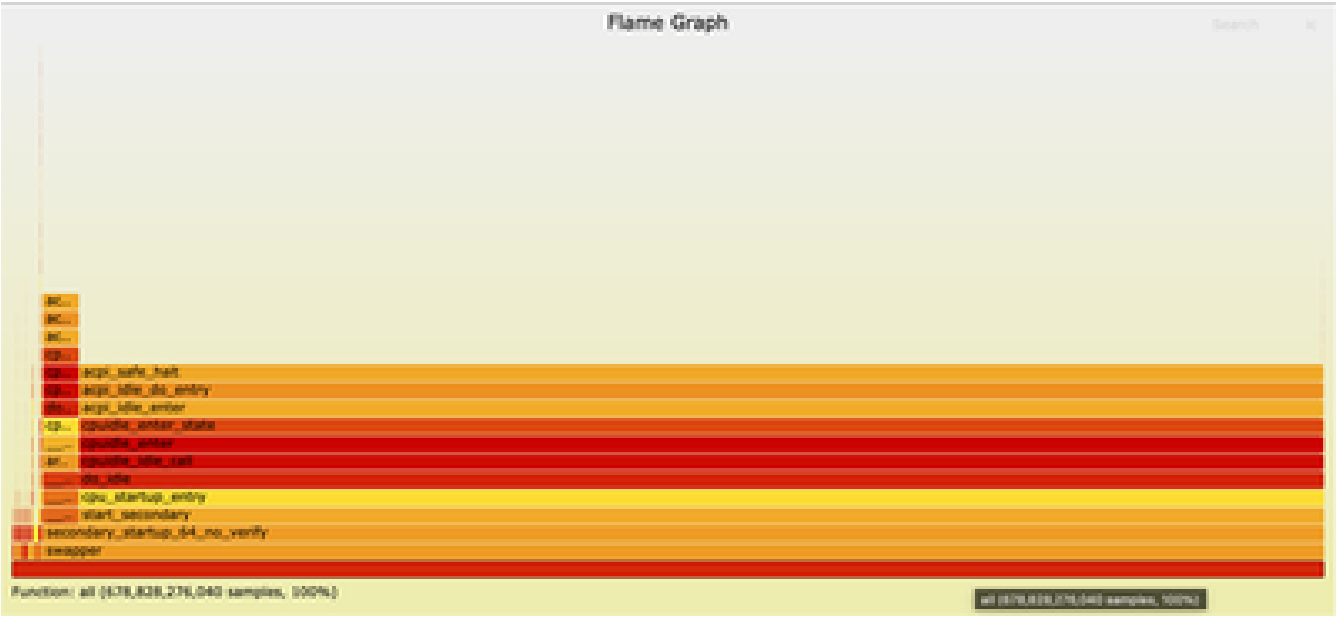
附註：如果您在CentOS或RHEL上遇到「cannot locate open.pm in @INC」錯誤，請安裝所需的Perl模組：

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

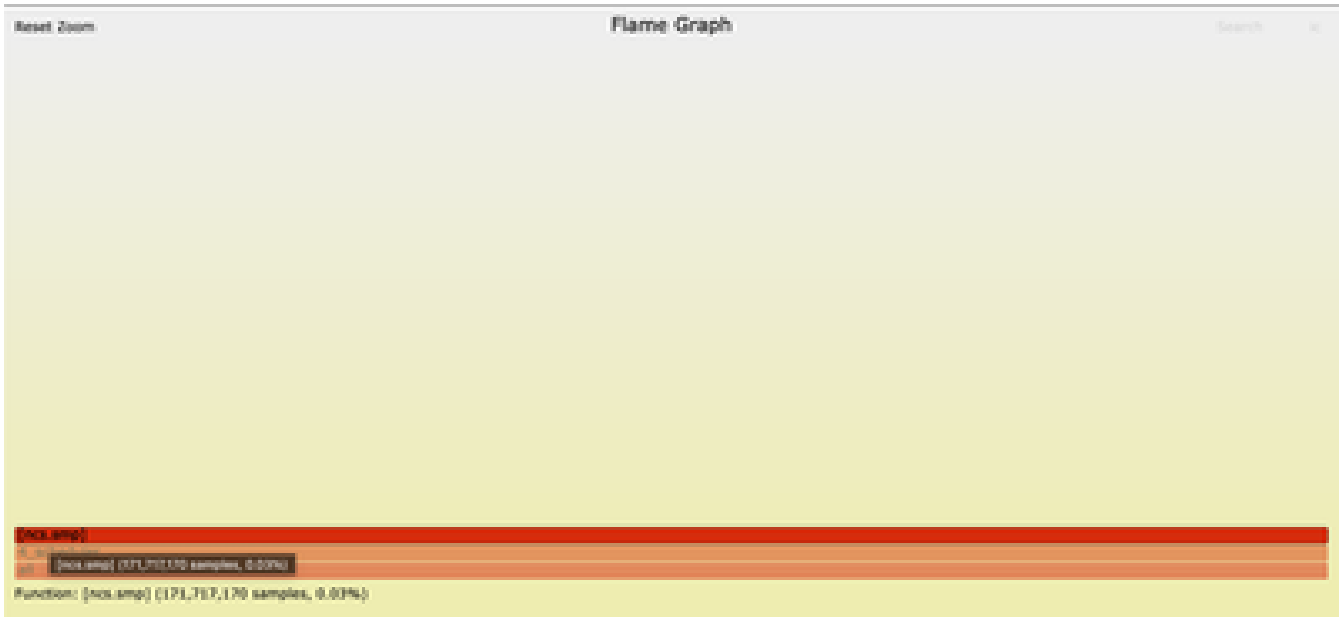
步驟 7:在首選Web瀏覽器中開啟data.svg檔案以視覺化火焰圖形。

## 瀏覽火焰圖形

在瀏覽器中開啟火焰圖檔案後，您可以通過按一下任何框來與其互動，以放大該函式及其呼叫棧。每個框的長度表示該函式及其呼叫堆疊中花費的CPU時間。此視覺化技術可以輕鬆識別熱點以及需要最佳化的區域。




放大ncs.smp:




# 相關資訊

- Linux效能已知警告
- 思科技術支援與下載

關於此翻譯

思科已使用電腦和人工技術翻譯本文件，讓全世界的使用者能夠以自己的語言理解支援內容。請注意，即使是最佳機器翻譯，也不如專業譯者翻譯的內容準確。Cisco Systems, Inc. 對這些翻譯的準確度概不負責，並建議一律查看原始英文文件（提供連結）。