Coletar e Gerar Gráficos de Estatísticas da CPU usando "PERF" Ferramenta em NSO

Contents

Introdução

Este documento descreve como usar a ferramenta perf em hosts NSO para investigar problemas de desempenho.

Pré-requisitos

Requisitos

A Cisco recomenda que você tenha conhecimento destes tópicos:

- Uso básico da linha de comando do Linux/Unix
- Arquitetura e operação do sistema NSO (Network Services Orchestrator)
- Conceitos de análise e criação de perfis de CPU
- Familiaridade com fluxos de trabalho de solução de problemas de desempenho

Componentes Utilizados

As informações neste documento são baseadas nestas versões de software e hardware:

- Sistema NSO ou instalação local em um host Unix/Linux suportado
- Distribuições Linux como Ubuntu, Debian, Fedora ou RedHat derivadas
- ferramenta perf (ferramenta de análise de desempenho do Linux)

As informações apresentadas neste documento foram criadas a partir de dispositivos em um ambiente de laboratório específico. Todos os dispositivos utilizados neste documento foram

iniciados com uma configuração (padrão) inicial. Se a rede estiver ativa, certifique-se de que você entenda o impacto potencial de qualquer comando.

Informações de Apoio

O Perf é uma poderosa ferramenta de análise de desempenho no Linux, usada principalmente para a criação de perfis de CPU. Ele fornece insights sobre o que a CPU está trabalhando atualmente, capturando e analisando a carga de funções de nível inferior. Isso ajuda a identificar quais funções ou processos estão ocupando a CPU e é essencial para identificar gargalos de desempenho.

Perf também pode gerar gráficos de chamas, que são gráficos especiais que representam visualmente quais partes de um programa usam a maior parte do tempo de CPU. Gráficos de chama facilitam a localização de áreas no código que precisam de otimização.

É importante observar que o desempenho também está incluído na lista de verificação da coleta de dados principal para casos de Memória Insuficiente (OOM), conforme recomendado pela Unidade de Negócios (BU) do NSO. Para obter orientações mais detalhadas sobre a solução de problemas de OOM, entre em contato com o TAC da Cisco.

Identificar e Solucionar Problemas de Desempenho de NSO

Esta seção fornece um fluxo de trabalho abrangente para instalar, usar e analisar dados da ferramenta perf em hosts NSO para solucionar problemas de desempenho.

Instalar Desempenho

Passo 1: Instale o perf na distribuição do Linux. Use o comando apropriado para o seu SO:

Para Ubuntu:

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

Para Debian:

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

Para os derivados Fedora/RedHat:

```
dnf install -y perf
```

Para obter mais informações sobre advertências conhecidas ao instalar o perf, entre em contato com a equipe do TAC da Cisco.

Amostragem dos dados

Passo 1: Identificar o processo NSO principal.

Use o comando fornecido abaixo para localizar o processo NSO (ncs.smp):

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

Saída de exemplo:

```
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
```

Passo 2: Como alternativa, você deve usar o PID do processo Java principal vinculado ao NSO, especialmente se estiver focalizando em operações Java. Executar:

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

Saída de exemplo:

```
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
```

Passo 3: Execute o caso de teste problemático ou o caso de uso para validar o cenário de desempenho.

Passo 4: Em uma janela de terminal diferente, execute perf com as IDs de processo (PIDs) relevantes. Use o formato de comando fornecido abaixo, substituindo XX,YY,ZZ pelos PIDs obtidos acima:

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

Por exemplo, para criar o perfil de todo o sistema e reunir gráficos de chamadas em 99 Hz para PIDs específicos:

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

Saída de exemplo:

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

Descrições da opção:

- -a: Todas as CPU; coleta em todo o sistema de todas as CPUs (padrão se nenhum destino for especificado).
- g: Capturar gráficos de chamadas (rastreamentos de pilha). Identifica onde as funções estão sendo chamadas.
- -f: Frequência de amostragem em Hz. Frequências mais altas aumentam a precisão, mas adicionam sobrecarga.
- -p: Especifica a(s) ID(s) do processo.

Passo 5: Quando terminar de coletar amostras, pare o desempenho com Ctrl+C:

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

Agora você verá um arquivo perf.data no diretório atual.

Passo 6: Gere um relatório de resumo com este comando:

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

Descrições da opção:

- n: Mostrar símbolos sem agrupamento (exibição simples).
- — stdio: Force a saída para a saída padrão (o terminal).

Neste ponto, você deve salvar os dois arquivos (perf.data e perf_report.txt) e compartilhá-los com seu contato de suporte antes de prosseguir com a análise.

Se a captura tiver sido bem-sucedida, o perf_report.txt mostrará uma estrutura em árvore representando um gráfico de chamadas hierárquico. As porcentagens ajudam a identificar os pontos de acesso onde a maior parte do tempo da CPU está sendo gasta.

Trecho de exemplo:

```
# 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
```

Interpretação:

- Processo/Thread: O thread C2 CompilerThre está sendo analisado.
- Uso Total da CPU: Esse segmento é responsável por 30,61% do tempo da CPU.
- Fluxo de função: O segmento começa com `start_thread` e delega o trabalho em várias camadas. A maior parte do tempo da CPU (30,47%) é gasta em `C2Compiler::compile_method`, indicando um ponto de acesso em potencial.

Geração de um gráfico de chama

Passo 1: Gerar uma amostra de desempenho de todas as CPUs e processos em um intervalo definido (por exemplo, 60 segundos):

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

Saída de exemplo:

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

Passo 2: Copie ou transfira este arquivo perf.data para um host do qual você possa fazer download do repositório de modelos do flamegraph.

Passo 3: Converta o arquivo perf.data em um formato de texto:

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

Passo 4: Clonar o repositório FlameGraph GitHub e colocar data.perf neste diretório:

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

Passo 5: Recolher os rastreamentos de pilha para processamento de flamégrafo:

<#root>

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

Passo 6: Gere o arquivo SVG do gráfico de chama:

<#root>

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

Note: Se você encontrar o erro "não é possível localizar open.pm em @INC" no CentOS ou RHEL, instale o módulo Perl necessário:

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

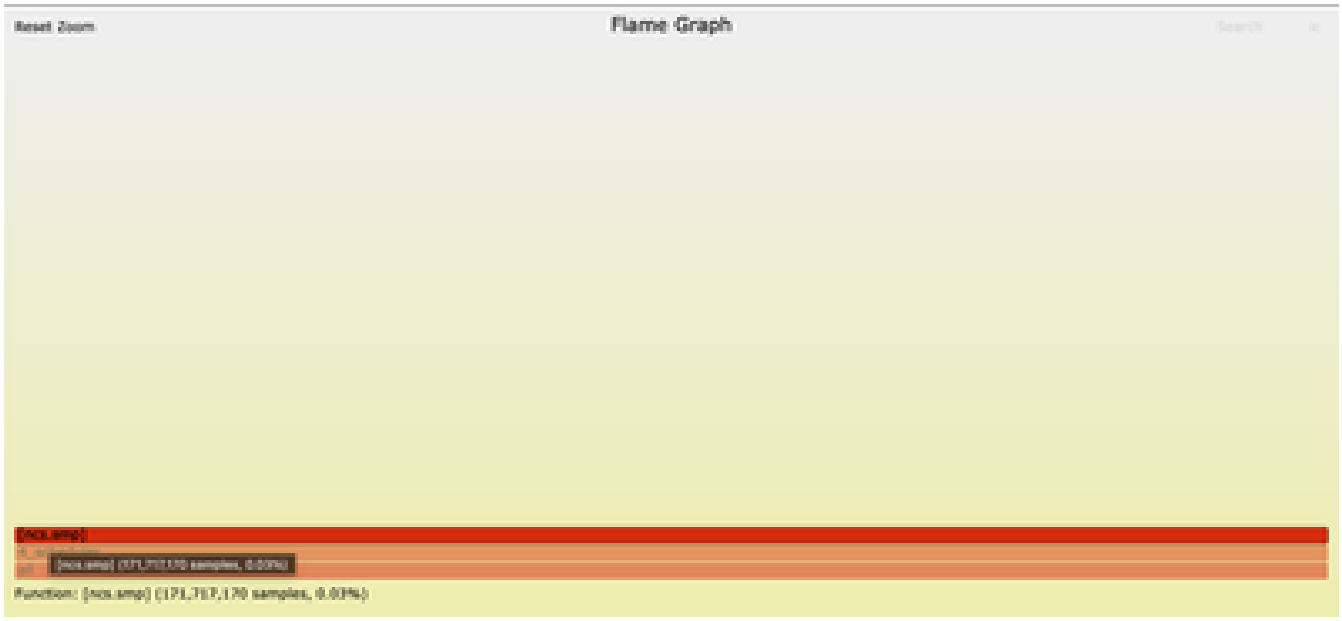
Passo 7: Abra o arquivo data.svg no navegador da Web de sua preferência para visualizar o gráfico de chamas.

Navegue pelo gráfico do Flame

Uma vez que o arquivo do gráfico de chamas está aberto no seu navegador, você pode interagir com ele clicando em qualquer caixa para ampliar essa função e sua pilha de chamadas. O comprimento de cada caixa representa a quantidade de tempo de CPU gasto nessa função e em

sua pilha de chamadas. Essa visualização facilita a identificação de pontos de acesso e áreas para otimização.

Ampliado em ncs.smp:



Informações Relacionadas

- Avisos conhecidos de desempenho do Linux
- Suporte técnico e downloads da Cisco

Sobre esta tradução
A Cisco traduziu este documento com a ajuda de tecnologias de tradução automática e humana para oferecer conteúdo de suporte aos seus usuários no seu próprio idioma, independentemente da localização.

Observe que mesmo a melhor tradução automática não será tão precisa quanto as realizadas por um tradutor profissional.

A Cisco Systems, Inc. não se responsabiliza pela precisão destas traduções e recomenda que o documento original em inglês (link fornecido) seja sempre consultado.