
Universidade Estadual de Feira de Santana


# Tutorial de Utilização do Nvidia Visual Profiler

Feira de Santana - BA

Julho, 2016

# 1 Introdução

Este tutorial é um resultado do projeto de pesquisa do estudante bolsista (Fapesb) Cássio Silva de Sá Santos e tem como objetivo apresentar um manual de utilização da ferramenta de medição de desempenho conhecida como Nvidia Visual Profiler que é disponibilizada através do Cuda ToolKit da Nvidia o qual já teve o seu tutorial de instalação desenvolvido e publicado por este bolsista.

Este tutorial tem como principal fonte a documentação oficial da Nvidia [6] e como complemento os testes e estudos feitos pelo bolsista.

O conteúdo deste tutorial foi criado para fins de pesquisa e pode ser usado livremente desde que citada a fonte. O LaCAD não se responsabiliza pelo uso dessas informações.

# 2 Nvidia Visual Profiler

O NVIDIA Visual Profiler é uma ferramenta criada pela Nvidia e disponível como parte integrante do Cuda Tool Kit que tem como principal tarefa ajudar no desenvolvimento de aplicações CUDA com maior desempenho.

O NVIDIA Visual Profiler é uma ferramenta gráfica criada pela Nvidia que tem como principal tarefa ajudar no desenvolvimento de aplicações C rodando na GPU permite o profiling destas aplicações. Devolve informações importantes para a otimização de aplicações CUDA C/C++. É disponível no Kit de Ferramentas CUDA (CUDA Toolkit) e é suportado por todas as plataformas que suportam o CUDA Toolkit. [1]

Entre suas principais características estão:

- Identifica rapidamente possíveis gargalos de desempenho na aplicação usando tabelas altamente configuráveis e visualizações gráficas.
- Realiza análise automática da aplicação para identificar os gargalos de performance e obter sugestões de otimização que podem ser usadas para melhorar o desempenho.
- Ver a atividade CUDA ocorrendo tanto em CPU quanto em GPU em uma única linha de tempo, incluindo as chamadas de API CUDA, transferências de memória e execuções de Kernel.
- Confirmar as melhorias de desempenho ao comparar com as versões anteriores. Observar como a energia, temperatura e valores de clock da GPU variam durante a execução da aplicação.
- Utilizar o Command-Line-Profiler usando variáveis de ambiente para coletar dados de múltiplos (outros) sistemas e analisar seus resultados no Nvidia Visual Profiler.

# 3 Executando o Profiling

## 3.1 Pela linha de comando

Para executar o profiler de uma aplicação usando o Nvidia Visual Profiler, você necessita ter um executável da sua aplicação previamente compilado.

**1.** Compilando a aplicação usando nvcc

```
$ nvcc x.cu
```

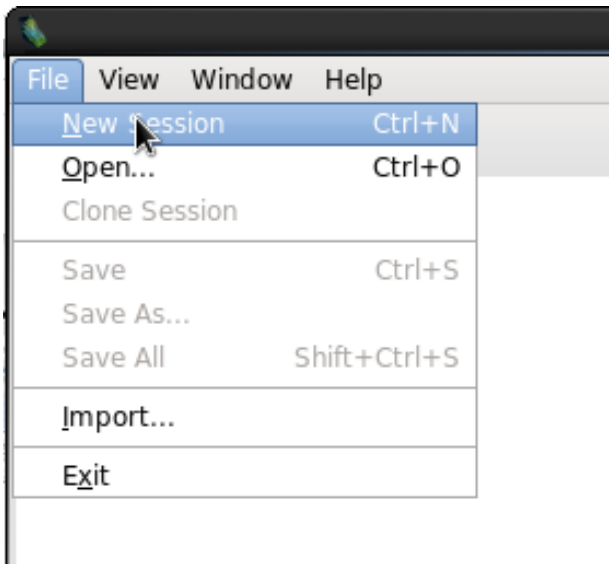
**2.** Através da linha de comando você pode executar o Nvidia Visual profiler:
Iniciando uma nova sessão de executável através da execução do nvvp com o nome do executável seguido opcionalmente dos argumentos. [2]

```
$ nvvp ./execname [execArgs]
```

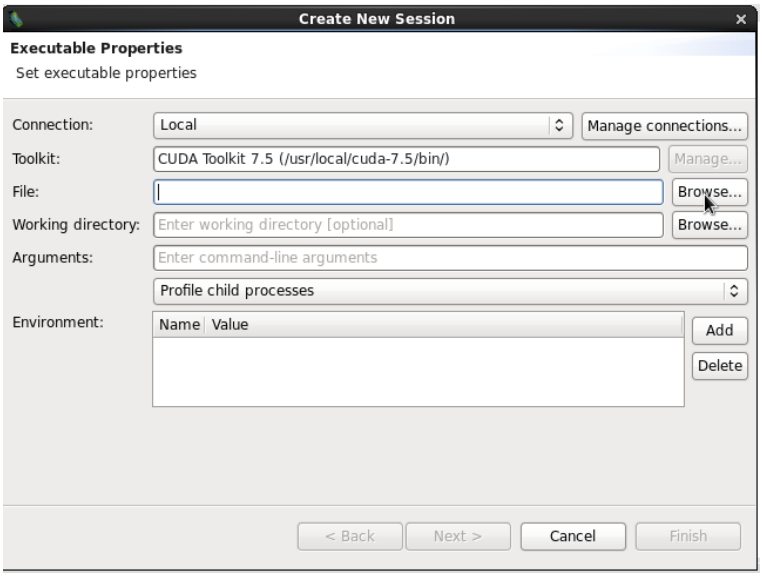
## 3.2 Pela interface gráfica

Para criar uma nova sessão utilizando a interface gráfica.

**1.** Inicie uma nova sessão (*Ctrl+N*)



**2.** Selecione o arquivo executável para gerar o perfil e configure da forma que for necessária.



# 4 Linha do tempo

A visualização de linha de tempo, permite que o usuário visualize a atividade na CPU e na GPU enquanto a aplicação esta sendo instrumentalizada. A **[Figura 1]** representa um exemplo de visualização de linha do tempo para uma aplicação Cuda [3].

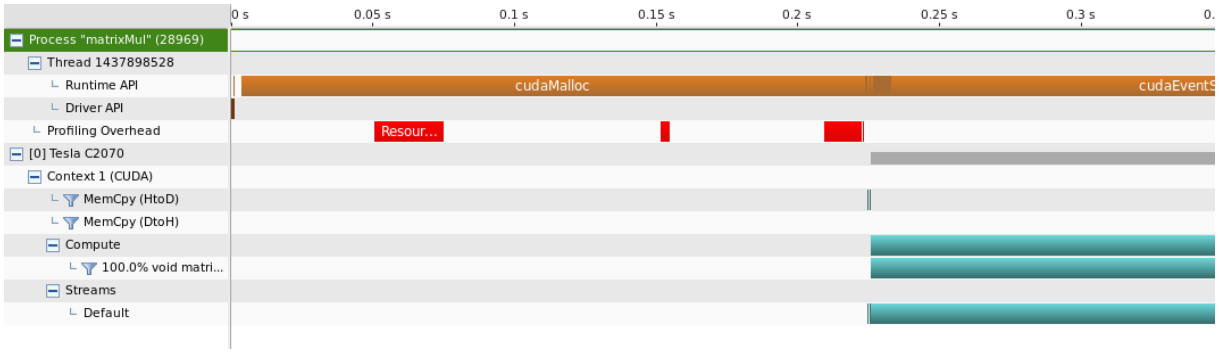



**Figura 1:** Visualização da linha do tempo do Nvidia Visual Profiler

## 4.1 Entendendo a visualização

Na parte superior da linha do tempo pode ser vista uma regua mostra o tempo decorrido desde o inicio da instrumentação da aplicação.

Do lado esquerdo pode ser visto uma arvore de comandos que descreve o que está sendo exibido em cada linha da *TimeLine*. Entre estas linhas que são exibidas no visão da linha do tempo podemos encontrar os tipos descritos na **[Tabela 1]**.

**Tabela 1:** Informações apresentadas em cada uma das linha do *TimeLine*

| | |
|---|---|
| **Process** | A linha do tempo irá conter uma linha "Process" para cada aplicação instrumentalizada. O número ao lado do nome da aplicação indica o Pid do processo (Process Id). |
| **Thread** | A linha do tempo irá conter uma linha "Thread" para cada thread da CPU na aplicação instrumentalizada que faz uma chamada as API CUDA runtime ou CUDA driver. |
| **Runtime API** | A linha do tempo irá conter uma linha "Runtime API" para cada thread CPU que faz chamadas a CUDA Runtime API. |
| **Driver API** | A linha do tempo irá conter uma linha "Driver API" para cada thread CPU que faz chamadas a CUDA Driver API. |
| **Profiling Overhead** | A linha do tempo irá conter uma unica linha "Profiling Overhead" para cada processo. Os intervalos desta linha indicam o tempo da execução de alguma atividade que é apenas requisitada para o perfil. Estas atividades não ocorrem quanto a aplicação não está sendo instrumentalizada. |

| | |
|---|---|
| **Device** | A linha do tempo irá conter uma linha "Device" para cada GPU usada na aplicação que está sendo instrumentalizada. Irá conter tanto o ID do dispositivo dentro dos cochetes quanto o nome do dispositivo. |
| **Memcpy** | Cada intervalo nesta linha representa a duração de um operação de manipulação(cópia) de memória executando na GPU, seja ela device-to-host, host-to-device, device-to-device ou peer-to-peer |
| **Compute** | Apresenta um resumo de toda a atividade de kernel que ocorrem em determinado dispositivo. |
| **Kernel** | Sempre seguida de uma linha "Compute", esta linha mostra a atividade de cada uma das execuções de kernel que ocorrem na aplicação. |
| **Stream** | A linha do tempo irá conter uma linha "Stream" para cada stream usada na aplicação. Cada intervalo nesta linha representa uma operação de manipulação de memória ou de execução de kernel que ocorrem nesta stream. |

Ao clicar em um dos intervalos podem ser vistas as propriedades (Aba Propriedades) das operações de manipulação de memória no que diz respeito a:

- Inicio
- Fim
- Duração
- Tamanho
- Vazão

E os intervalos de execução de kernel no que diz respeito a:

- Inicio
- Fim
- Duração
- Tamanho do Grid
- Tamanho do bloco
- Registradores por Thread
- Memória compartilhada por Bloco

# 5 Análise guiada

A ferramenta *Nvidia Visual Profiler* apresenta um guia passo a passo para aplicar a analise de performance na sua aplicação.

Esta visão é usada para controlar a análise da aplicação e para exibir os resultados. Neste modo (guiado), o sistema de análise irá guiá-lo em fases para ajudar a entender os limitadores de desempenho e quais são as oportunidades de otimização em sua aplicação.

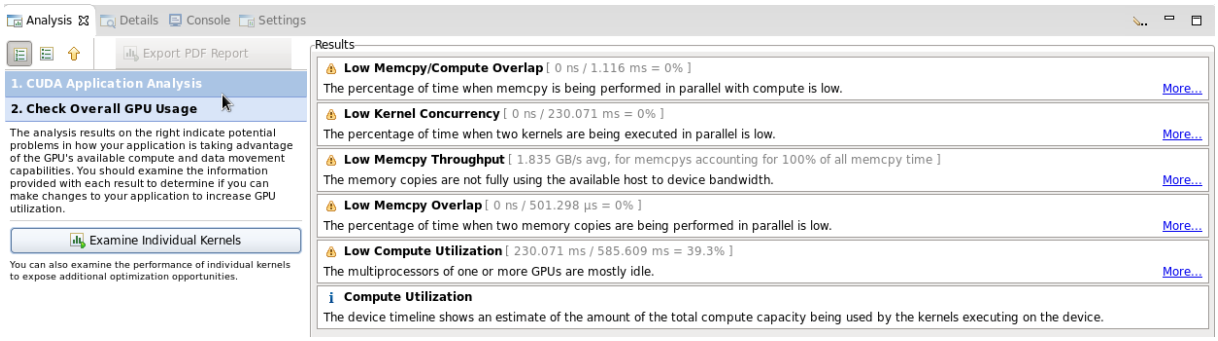



**Figura 2:** Análise guiada do Nvidia Visual profiler

Na **[Figura 2]** no lado esquerdo é possível ver as instruções passo a passo para lhe ajudar a otimizar sua aplicação. No lado direito são exibidos os resultados detalhados de cada parte da análise.

Clicando em "*more...*" para cada um dos resultados obtidos, você será redirecionado para a parte da documentação que diz respeito ao conteúdo necessário para fazer a otimização neste aspecto.

Você pode percorrer pelos passos apresentados no lado esquerdo e analisar as informações apresentadas ao lado direito para verificar porque este passo de otimização lhe foi sugerido.

Quanto a analise de latência de memória, o Nvidia Visual Profiler permite a analise da máxima utilização de qualquer subsistema de memória **[Figura 3]**, entre eles:

- L1/ Share Memory
- L2 Cache
- Texture Cache
- Device Memory
- System Memory

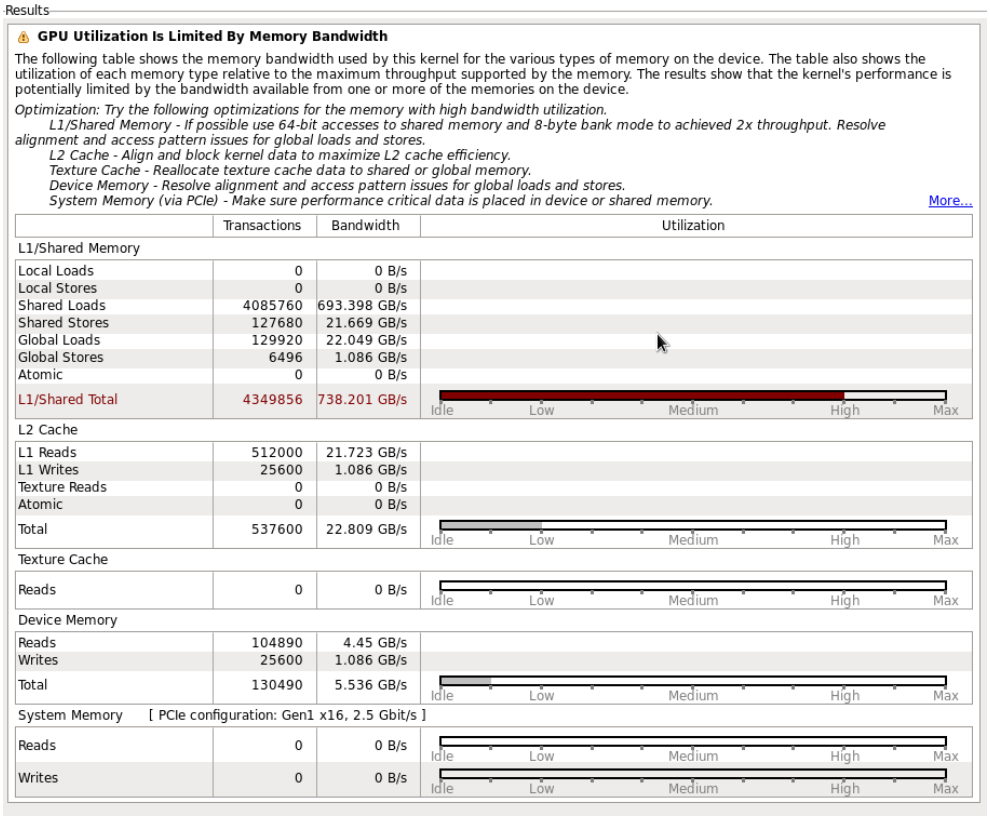



**Figura 3:** Análise de latência de memória pelo *Nvidia Visual Profiler*

Na **[Figura 4]**, podemos ver um exemplo de analise de ocupação proveniente de um dos passos de otimização indicados pelo Nvidia Visual Profiler. Nota-se que o resultado descreve em detalhes cada um dos passos apresentando até mesmo uma explicação sobre o passo de otimização em questão.

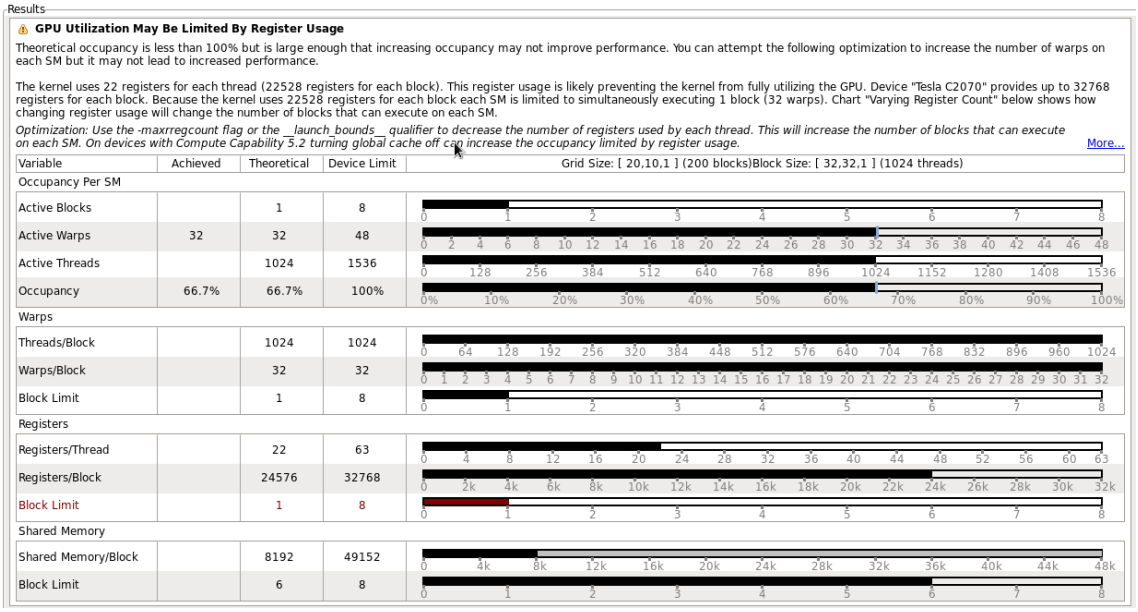



**Figura 4:** Exemplo de Análise de Ocupação

O Nvidia Visual Profiler consegue ainda indicar se sua aplicação em determinado dispositivo é limitada pelo computo ou pela memória. Caso limitada pela memória, um dos passos de otimização irá retratar justamente isso mostrando graficamente a relação entre os níveis de utilização, entre Computo ou Transferência de memória **[Figura 5]**.

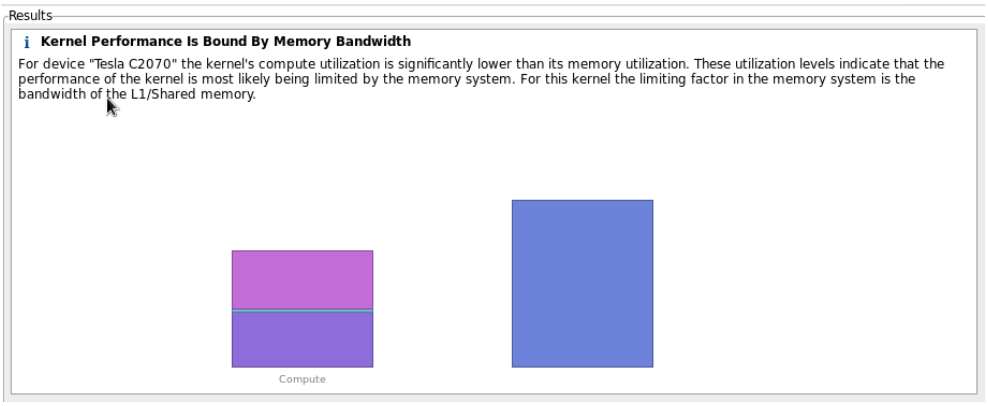



**Figura 5:** Análise Utilização de computo X Utilização de memória

Estes são apenas alguns exemplos de resultados apresentados pelo *Nvidia Visual Profiler*, a visualização de outras analises dependem se estas são limitantes de tempo de sua aplicação ou não (Análise guiada).

## 5.1 Gerando relatório da Análise Guiada

A ferramenta lhe da a possibilidade de gerar um relatório em formato PDF com todos os passos e informações necessárias para fazer a otimização através da Analise Guiada. Para isso, basta clicar

no botão destacado na **[Figura 6]** e o arquivo de relatório será gerado.

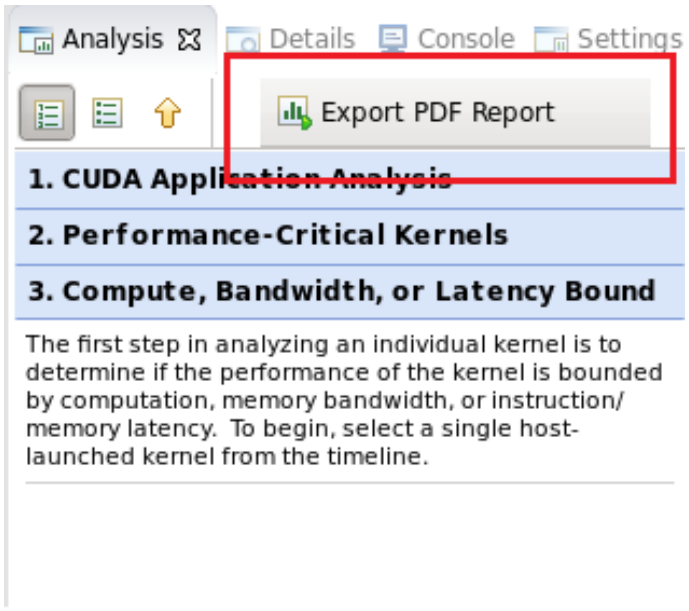



**Figura 6:** Exportando PDF do relatório da analise Guiada

# 6 Análise não guiada

Neste modo você pode explorar manualmente todos os resultados de analise coletados para sua aplicação. [5]

Para trocar da análise guiada para a analise não guiada clique no botão indicado pela seta apresentada na **[Figura 7]**.

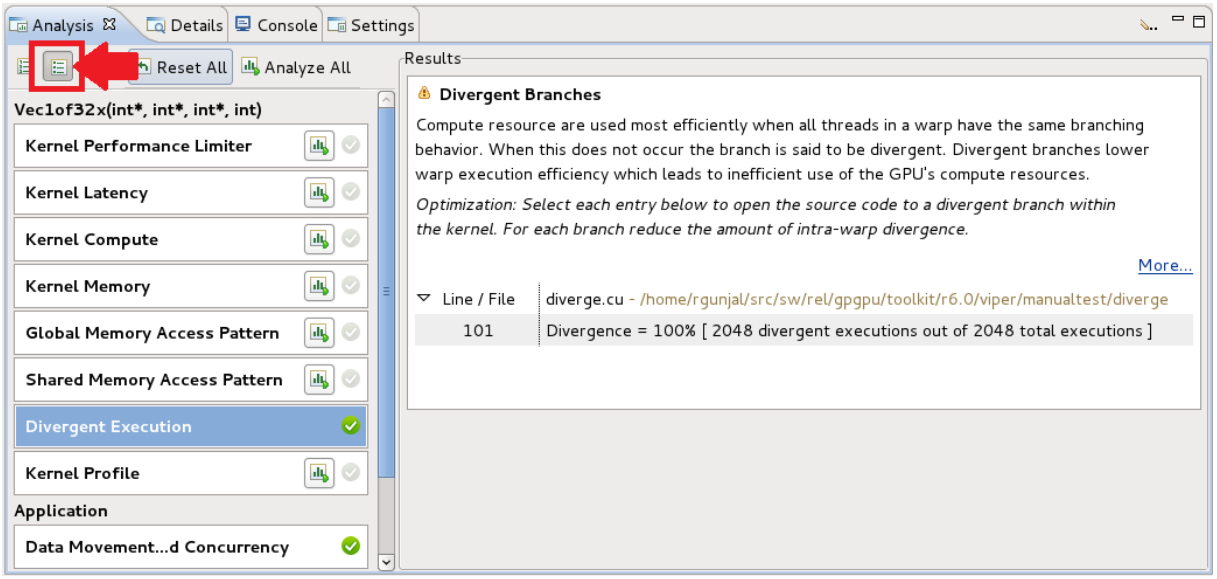



**Figura 7:** Analise não guiada do Nvidia Visual Profiler [5]

Na barra da esquerda é possível escolher qual análise deve ser feita de forma manual. Para isso, basta clicar no botão ao lado no nome de cada uma das análises.

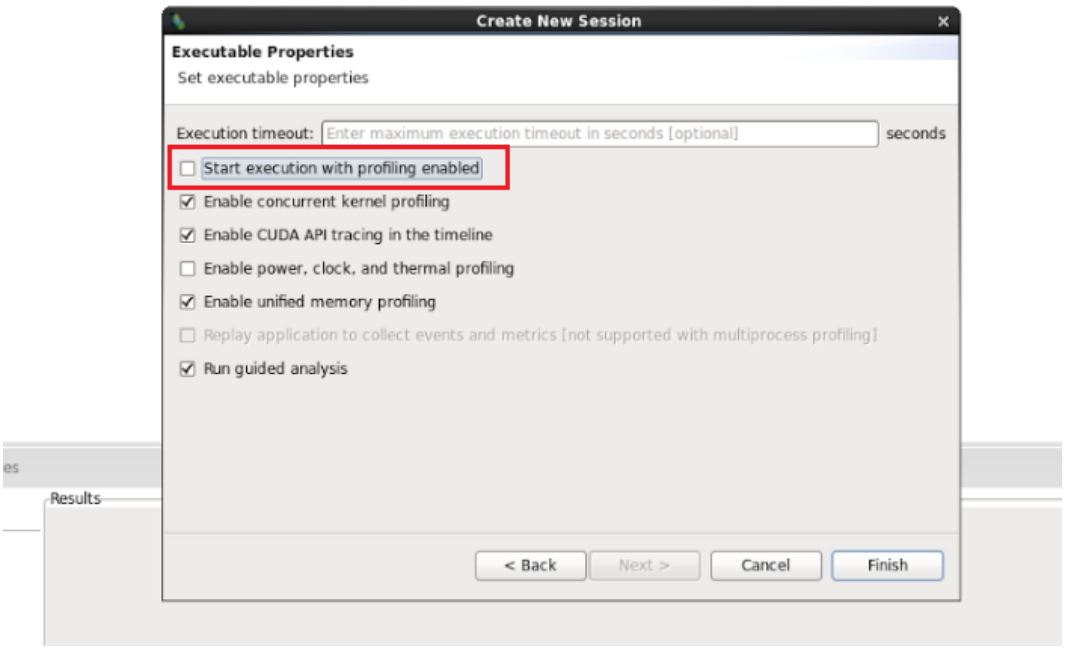
# 7 Limitando a região do Profiler

Por padrão, as ferramentas de *profiling* coletam dados de toda a execução da aplicação. Porém é possível limitar a região de profiler para reduzir a quantidade de dados de perfil que a ferramenta deve processar e, portanto, focar a atenção na parte do código no qual o resultado de uma otimização vai ser mais significativo no ganho de performance.

**Passos: 1.1.** Adicione cudaProfilerStart() / cudaProfilerStop()

```
for (i = 0; i < N; i++) {
        if (i == 12) cudaProfilerStart();
        if (i == 15) cudaProfilerStop();
}
```

**1.2.** Desative o profiling no inicio da aplicação (Configurações na criação da nova sessão)



# 8 Importando perfis

## 8.1 Importando um perfil do Nvprof

O Nvidia Visual Profiler permite o usuário importar perfis gerados a partir do Nvprof. As formas de fazer essa importação são as seguintes:

### 8.1.1 Pela linha de comando

**Digite:**

```
$ nvvp profile.out
```

No qual profile.out foi um arquivo de saída exportado utilizando o nvprof (Ver tutorial do Nvprof [4])

### 8.1.2 Pela interface gráfica

1. Na aba de ferramentas clique em "File-> Import..."
2. Selecione a opção "Nvprof" e clique em "Next >"
3. Selecione a opção "Single process" e clique em "Next >"
4. Em "Timeline data file" selecione o caminho do arquivo de perfil gerado pelo Nvprof e clique em Next.

## 8.2 Importando um perfil do Command-Line-Profiler

**Restrições:**

Para importar um perfil do Command-Line-Profiler usando o Nvvp o arquivo de perfil gerado pelo Command-Line-Profiler deve:

**1.** Ter sido criado no modo separado por virgula:

```
$ export COMPUTE_PROFILE_CSV=1
```

**2.** Ter o arquivo de configuração apontando para um arquivo que contem os parâmetros "gpustarttimestamp" e "streamid":

```
$ touch config
$ vi config
(Insira os dois parametros cada um em uma linha)
$ export COMPUTE_PROFILE_CONFIG= ./config
```

Por fim deve-se seguir os passos apresentados para importação de perfis gerados do Nvprof, alterando apenas a seleção de "Nvprof" para "Command Line Profiler" quando for exigido (Caso importando em interface gráfica). Para mais informações sobre o Command Line Profiler, veja o seu tutorial [7]

# Referências

[1] https://developer.nvidia.com/nvidia-visual-profiler

[2] http://docs.nvidia.com/cuda/profiler-users-guide/#command-line-arguments

[3] http://docs.nvidia.com/cuda/profiler-users-guide/index.html#timeline-view

[4] http://lacad.uefs.br/wp-content/uploads/2016/07/tutorial-nvprof.pdf

[5] http://docs.nvidia.com/cuda/profiler-users-guide/#unguided-analysis

[6] http://docs.nvidia.com/cuda/index.html

[7] http://lacad.uefs.br/wp-content/uploads/2016/07/tutorial-command-line.pdf