Universidade Estadual de Feira de Santana

Tutorial de Utilização do Nvprof

Feira de Santana - BA

Julho, 2016

1 Introdução

Este tutorial é um resultado do projeto de pesquisa do estudante bolsista (Fapesb) do LaCAD, Cássio Silva de Sá Santos e tem como objetivo apresentar um manual de utilização da ferramenta de medição de desempenho conhecida como Nvprof 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 [8] 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 Nvprof

A ferramenta nvprof permite que sejam coletados dados de perfis através da linha de comando (console) disponível em Linux, Windows e OS X.

Nvprof permite coletar um cronograma de atividades relacionadas com o CUDA, tanto na CPU quanto na GPU, incluindo a execução do kernel, transferências de memória, conjuntos de memória e chamadas à API CUDA. Os resultados dos perfis são exibidos no console e também podem ser guardados para visualização posterior pelo Nvprof ou pelo Visual Profiler. [1]

Para gerar o perfil de uma aplicação CUDA usando o nvprof basta usar o seguinte padrão:

```
$ nvprof [opcoes] [aplicacao-CUDA] [argumentos-da-aplicacao]
```

Obs.: NVPROF e Command-Line-Profiler são ferramentas que se excluem mutualmente. Se o nvprof for chamado quando o command-line profiler estiver ativo, o nvprof vai apresentar um erro e fechar [Figura 1].

```
[root@enxu testes]# nvprof --print-api-trace ./async
======== Error:
nvprof cannot profile the application when the "COMPUTE_PROFILE" or
"CUDA_PROFILE" environment variable is set to non-zero. Either unset the
environment variable, or run the application outside of nvprof. See the
CUDA profiler user's guide for more detail.
======== Error: incompatible environment variable.
[root@enxu testes]#
```

Figura 1: Erro apresendado pelo Nvprof quando a variável de ambiente do Command-Line-Profiler está ativa.

Caso este erro seja apresentado, basta modificar o valor da variável de ambiente que controla o Command-Line-Profiler, para isso, digite no console:

```
$ export COMPUTE_PROFILE=0
$ export CUDA_PROFILE=0
```

2.1 Modos de Profiling do Nvprof

Cada saida do nvprof é iniciada com ==<pid>== no qual <pid> representa o Id do processo da aplicação a qual esta sendo gerado o perfil.

O nvprof pode operar executando profiling em basicamente quatro modos, estes modos serão listados a seguir:

2.1.1 Modo Resumo

O modo resumo é o modo de operação padrão do Nvprof. Neste modo o nvprof apresenta como saída uma linha para cada função kernel ou copia de memória (memcpy). Este resumo agrupa todas as chamadas para um mesmo kernel juntas apresentando informações quanto ao tempo total de todas as instâncias do procedimento em questão bem como a quantidade de chamadas e os tempos mínimo, médio e máximo para cada tipo de procedimento, seja ele uma função kernel ou uma operação de copia de memória. [6]

Para executar o nvprof no modo resumo basta digitar no console:

```
$ nvprof ./seu_exec <argumentos_da_aplicacao>
```

O resultado esperado para este tipo de execução pode ser visualizado na [Figura 2]

```
[root@enxu testes]# nvprof ./async
[./async] - Starting...
==2838== NVPROF is profiling process 2838, command: ./async
time spent executing by the GPU: 77.12
CPU executed 56409 iterations while waiting for GPU to finish
==2838== Profiling application: ./async
==2838== Profiling result:
Time(%)      Time     Calls       Avg       Min       Max  Name
 52.61%  40.826ms         1  40.826ms  40.826ms  40.826ms  [CUDA memcpy DtoH]
 42.91%  33.297ms         1  33.297ms  33.297ms  33.297ms  [CUDA memcpy HtoD]
  3.81%  2.9531ms         1  2.9531ms  2.9531ms  2.9531ms  increment_kernel(int*, int)
  0.68%  528.62us         1  528.62us  528.62us  528.62us  [CUDA memset]

==2838== API calls:
Time(%)      Time     Calls       Avg       Min       Max  Name
 53.35%  189.33ms         1  189.33ms  189.33ms  189.33ms  cudaMallocHost
 21.78%  77.312ms         1  77.312ms  77.312ms  77.312ms  cudaDeviceReset
 17.44%  61.892ms     56410  1.0970us  1.0020us  1.9534ms  cudaEventQuery
  6.92%  24.573ms         1  24.573ms  24.573ms  24.573ms  cudaFreeHost
  0.15%  516.42us         1  516.42us  516.42us  516.42us  cudaDeviceSynchronize
  0.09%  312.31us        83  3.7620us     285ns  129.79us  cuDeviceGetAttribute
  0.09%  307.14us         1  307.14us  307.14us  307.14us  cudaMalloc
  0.08%  277.57us         1  277.57us  277.57us  277.57us  cudaFree
  0.02%  85.604us         1  85.604us  85.604us  85.604us  cudaLaunch
  0.02%  60.581us         1  60.581us  60.581us  60.581us  cudaMemset
  0.01%  50.782us         2  25.391us  19.188us  31.594us  cudaMemcpyAsync
  0.01%  46.160us         1  46.160us  46.160us  46.160us  cuDeviceGetName
  0.01%  43.289us         1  43.289us  43.289us  43.289us  cuDeviceTotalMem
  0.01%  39.819us         2  19.909us     596ns  39.223us  cudaSetupArgument
  0.00%  16.190us         2  8.0950us  2.0000us  14.190us  cudaEventDestroy
  0.00%  15.278us         2  7.6390us  6.0650us  9.2130us  cudaEventRecord
  0.00%  12.060us         2  6.0300us  1.6840us  10.376us  cudaEventCreate
  0.00%  9.4030us         1  9.4030us  9.4030us  9.4030us  cudaConfigureCall
  0.00%  6.4010us         1  6.4010us  6.4010us  6.4010us  cudaEventElapsedTime
  0.00%  4.1460us         2  2.0730us     732ns  3.4140us  cuDeviceGetCount
  0.00%  1.2690us         2     634ns     467ns     802ns  cuDeviceGet
[root@enxu testes]#
```

Figura 2: Execução do Nvprof no modo resumo (Summary Mode)

O profiling é dividido em duas tabelas principais. A primeira que diz respeito as execuções de kernels da aplicação juntamente com os os tipos de copia de memória executados, e a segunda que

diz respeito as todas as chamadas de API CUDA em tempo de execução ou do driver.

Neste modo, por padrão, podemos observar em cada uma das linhas do resultado, da esquerda para a direita as informações apresentadas na [Tabela 1].

Coluna	Descrição
Time (%)	Porcentagem de tempo do kernel ou operação de manipulação de memoria em relação ao tempo total da aplicação
Time	O tempo total equivalente ao ao tempo de execução do kernel do dispositivo levando em consideração a soma de todas as execuções do kernel.
Calls	O número de chamadas feitas a determinada função kernel, procedimento de API ou operação de manipulação de memoria.
Avg	O tempo médio de execução do procedimento
Min	O tempo equivalente a execução do procedimento mais rápido de determinada função.
Max	O tempo equivalente a execução do procedimento mais lento de determinada função.
Name	O nome da função kernel, procedimento de manipulação de memória ou do método da API CUDA.

Tabela 1: Informações apresentadas na tabela no modo resumo

2.1.2 Modos Rastro de GPU e Rastro de API

O rastro GPU e o rastro API podem ser ativados individualmente ou ao mesmo tempo.
O modo rastro GPU resulta em uma linha do tempo de todas as atividades na GPU ordenadas de forma cronologica. Cada linha nesse tipo de perfil representa uma execução kernel ou uma operação de transferência de memória na qual são apresentadas informações tais como aos parâmetros do kernel, a memória compartilhada usada e vazão da memória transferida [7].

Para executar o nvprof nesse modo basta passar a opção –print-gpu-trace ao nvprof, como por exemplo:

$$\$nvporf\ {-}{-}print{-}gpu{-}trace\ ./seu\_exec\ <argumentos\_da\_aplicaÃ§Ã£o>$$

O resultado esperado para este tipo de execução pode ser visualizado na [Figura 3]

```
[root@enxu testes]# nvprof --print-gpu-trace ./async
[./async] - Starting...
==3979== NVPROF is profiling process 3979, command: ./async
time spent executing by the GPU: 77.25
CPU executed 49628 iterations while waiting for GPU to finish
==3979== Profiling application: ./async
==3979== Profiling result:
   Start  Duration            Grid Size      Block Size     Regs*    SSMem*    DSMem*      Size  Throughput           Device   Context    Stream  Name
358.70ms  529.07us                    -               -        -         -         -  64.000MB  118.13GB/s  Tesla C2070 (0)         1         7  [CUDA memset]
359.31ms  33.302ms                    -               -        -         -         -  64.000MB  1.8768GB/s  Tesla C2070 (0)         1         7  [CUDA memcpy HtoD]
392.61ms  2.9504ms          (32768 1 1)       (512 1 1)       10       0B       0B         -           -  Tesla C2070 (0)         1         7  increment_kernel(int*, int) [101]
395.56ms  40.946ms                    -               -        -         -         -  64.000MB  1.5264GB/s  Tesla C2070 (0)         1         7  [CUDA memcpy DtoH]

Regs: Number of registers used per CUDA thread. This number includes registers used internally by the CUDA driver and/or tools and can be more than what the compiler shows.
SSMem: Static shared memory allocated per CUDA block.
DSMem: Dynamic shared memory allocated per CUDA block.
[root@enxu testes]#
```

Figura 3: Execução do Nvprof no modo rastro de GPU

Neste modo, por padrão, podemos observar em cada uma das linhas do resultado, da esquerda para a direita as informações apresentadas na [Tabela 2].

Coluna	**Descrição**
Start	Instante em qual foi iniciado a função kernel ou operação de manipulação de memória
Duration	Tempo total decorrido desde o inicio da chamada da operação, até o seu fim.
Grid Size	Quando a linha diz respeito a uma execução de kernel representa os tamanhos nas coordenadas x, y e z do Grid do kernel
Block Size	Quando a linha diz respeito a uma execução de kernel representa os tamanhos nas coordenadas x, y e z de cada bloco do kernel
Regs	Número de registrados usados por cada Thread CUDA. Este número inclui os registradores usados internalmente pelo driver CUDA
SSMem	Memória compartilhada estática alocada por cada bloco CUDA
DSMem	Memória compartilhada dinámica alocada por cada bloco CUDA
Size	Quando a linha diz respeito a uma operação de manipulação de memória, representa o tamanho dos dados a serem manipulados
Throughput	Quando a linha diz respeito a uma operação de manipulação de memória, representa a Vazão dos dados sendo transferidos do Host para o Device ou vice versa.
Device	O nome do dispositivo no qual está sendo executado a operação.
Context	
Stream	
Name	O nome da função kernel juntamente com os seus parâmetros ou procedimento de manipulação de memória.

Tabela 2: Informações apresentadas na tabela no modo rastro de Gpu

O modo rastro de API mostra a linha do tempo de todas as chamdas CUDA em tempo de execução e todas as chamadas a API do driver invocadas no host. Este perfil é ordenado de forma cronologica e é ativado de forma analoga ao rastro GPU.

Para executar o perfil neste modo basta adicionar a opção –print-api-trace ao nvprof como o exemplo a seguir:

```
$nvporf --print-api-trace ./seu_exec <argumentos_da_aplicacao>
```

O resultado esperado para essa execução é algo como o exibido na [Figura 4]

```
[root@enxu testes]# nvprof --print-api-trace ./async
[./async] - Starting...
==4347== NVPROF is profiling process 4347, command: ./async
time spent executing by the GPU: 77.12
CPU executed 54480 iterations while waiting for GPU to finish
==4347== Profiling application: ./async
==4347== Profiling result:

312.72ms  1.0420us  cudaEventQuery
312.73ms  1.0680us  cudaEventQuery
312.73ms  1.0730us  cudaEventQuery
312.73ms  1.0470us  cudaEventQuery
312.73ms  1.0470us  cudaEventQuery
312.73ms  1.0780us  cudaEventQuery
312.73ms  1.0530us  cudaEventQuery
312.73ms  1.0370us  cudaEventQuery
312.73ms  1.2380us  cudaEventQuery
312.74ms  1.0780us  cudaEventQuery
312.74ms  2.5060us  cudaEventQuery
312.74ms  6.8530us  cudaEventElapsedTime
391.26ms  14.281us  cudaEventDestroy
391.28ms  1.7590us  cudaEventDestroy
391.29ms  29.524ms  cudaFreeHost
420.82ms  256.27us  cudaFree
1.12360s  77.753ms  cudaDeviceReset
[root@enxu testes]#
```

Figura 4: Execução do Nvprof no modo rastro de API

Neste modo, por padrão, podemos observar em cada uma das linhas do resultado, da esquerda para a direita as informações apresentadas na [Tabela 3].

Coluna	Descrição
Start	Instante em qual foi iniciado a chamada da API
Duration	Tempo total decorrido desde o inicio da chamada da operação, até o seu fim.
Name	Nome do método da API CUDA

Tabela 3: Informações apresentadas na tabela no modo rastro de API

2.1.3 Modo de Evento/Métrica resumidos

Você pode coletar métricas ou eventos de sua aplicação agrupados por Kernel, para isso, execute o nvprof com as flags - -metrics e - -events

```
$ nvprof --metrics nome_da_metrica_01,nome_da_metrica_02 \
--events nome_do_evento_01,nome_do_evento_02 \
./seu_exec <argumentos da aplicacao>
```

É possível coletar múltiplos eventos ou métricas ao mesmo tempo, para isso separe o nome de referência desta métrica/evento por virgula.

Listando métricas e eventos

Para ver a lista de todas as métricas e eventos disponíveis/suportados na sua GPU NVIDIA digite respectivamente:

```
$ nvprof --query-metrics
$ nvprof --query-events
```

Para referências sobre as métricas acesse [5]:

Coletando todas as métricas e eventos

Para coletar todas as métricas e eventos use respectivamente as opções:
–metrics all e –events all
Exemplo:

```
$ nvprof --metrics all --events all ./seu_exec
```

Exemplos de execução do –events all e –metrics all podem ser visualizados respectivamente nas figuras [5] e [6]

```
[root@enxu testes]# nvprof --events all ./async
[./async] - Starting...
==4380== NVPROF is profiling process 4380, command: ./async
==4380== Some kernel(s) will be replayed on device 0 in order to collect all events/metrics.
==4380== Replaying kernel "increment_kernel(int*, int)" (done)
time spent executing by the GPU: 1015.52
CPU executed 43654 iterations while waiting for GPU to finish
==4380== Warning: The following aggregate event values were extrapolated from limited profile
tex0_cache_sector_queries,l1_local_load_hit,elapsed_cycles_sm,tex0_cache_sector_misses,l1_glob
_miss,l1_local_store_hit,l1_local_store_miss
==4380== Profiling application: ./async
==4380== Profiling result:
==4380== Event result:
Invocations                                Event Name         Min         Max         Avg
Device "Tesla C2070 (0)"
        Kernel: increment_kernel(int*, int)
          1                                 local_load           0           0           0
          1                                local_store           0           0           0
          1                                gld_request      524288      524288      524288
          1                                gst_request      524288      524288      524288
          1                                shared_load           0           0           0
          1                               shared_store           0           0           0
          1                                     branch     1048576     1048576     1048576
          1                            divergent_branch           0           0           0
          1                               active_cycles    15291846    15291846    15291846
          1                                 inst_issued    28837488    28837488    28837488
          1                               inst_executed    28835840    28835840    28835840
          1                              warps_launched      524288      524288      524288
          1                             threads_launched    16777216    16777216    16777216
          1                    thread_inst_executed_0   461373440   461373440   461373440
          1                    thread_inst_executed_1   461373440   461373440   461373440
          1                              active_warps   624868629   624868629   624868629
          1                tex0_cache_sector_queries           0           0           0
          1                 tex0_cache_sector_misses           0           0           0
          1                         sm_cta_launched       32760       32760       32760
          1                       l1_local_load_hit           0           0           0
          1                      l1_local_load_miss           0           0           0
          1                       l1_local_store_hit           0           0           0
```

Figura 5: Execução de Modo Resumo para Eventos

```
[root@enxu testes]# nvprof --metrics all ./async
[./async] - Starting...
==4427== NVPROF is profiling process 4427, command: ./async
==4427== Some kernel(s) will be replayed on device 0 in order to collect all events/metrics.
==4427== Replaying kernel "increment_kernel(int*, int)" (done)
time spent executing by the GPU: 2462.86
CPU executed 43591 iterations while waiting for GPU to finish
==4427== Profiling application: ./async
==4427== Profiling result:
==4427== Metric result:
Invocations                               Metric Name                        Metric Description         Min         Max         Avg
Device "Tesla C2070 (0)"
        Kernel: increment_kernel(int*, int)
          1                  l1_cache_global_hit_rate                      L1 Global Hit Rate       0.00%       0.00%       0.00%
          1                        branch_efficiency                        Branch Efficiency     100.00%     100.00%     100.00%
          1                   l1_cache_local_hit_rate                       L1 Local Hit Rate       0.00%       0.00%       0.00%
          1                            sm_efficiency                  Multiprocessor Activity      99.78%      99.78%      99.78%
          1                       achieved_occupancy                       Achieved Occupancy    0.850207    0.850207    0.850207
          1                  gld_requested_throughput         Requested Global Load Throughput  32.790GB/s  32.790GB/s  32.790GB/s
          1                  gst_requested_throughput        Requested Global Store Throughput  32.790GB/s  32.790GB/s  32.790GB/s
          1                                       ipc                              Executed IPC    1.885817    1.885817    1.885817
          1                    sm_efficiency_instance                  Multiprocessor Activity      99.78%      99.78%      99.78%
          1                              ipc_instance                              Executed IPC    1.885817    1.885817    1.885817
          1                             inst_per_warp                 Instructions per warp   55.000000   55.000000   55.000000
          1                       gld_transactions               Global Load Transactions      524440      524440      524440
          1                       gst_transactions              Global Store Transactions      524328      524328      524328
          1                  local_load_transactions                Local Load Transactions           0           0           0
          1                 local_store_transactions               Local Store Transactions           0           0           0
          1                 shared_load_transactions               Shared Load Transactions           0           0           0
          1                shared_store_transactions              Shared Store Transactions           0           0           0
          1            gld_transactions_per_request     Global Load Transactions Per Request    1.000290    1.000290    1.000290
          1            gst_transactions_per_request    Global Store Transactions Per Request    1.000076    1.000076    1.000076
          1       local_load_transactions_per_request  Local Memory Load Transactions Per Reque    0.000000    0.000000    0.000000
          1      local_store_transactions_per_request  Local Memory Store Transactions Per Requ    0.000000    0.000000    0.000000
          1      shared_load_transactions_per_request  Shared Memory Load Transactions Per Requ    0.000000    0.000000    0.000000
          1     shared_store_transactions_per_request  Shared Memory Store Transactions Per Req    0.000000    0.000000    0.000000
          1                   local_load_throughput            Local Memory Load Throughput  0.00000B/s  0.00000B/s  0.00000B/s
          1                  local_store_throughput           Local Memory Store Throughput  0.00000B/s  0.00000B/s  0.00000B/s
          1                  shared_load_throughput           Shared Memory Load Throughput  0.00000B/s  0.00000B/s  0.00000B/s
          1                 shared_store_throughput          Shared Memory Store Throughput  0.00000B/s  0.00000B/s  0.00000B/s
          1                       shared_efficiency                Shared Memory Efficiency       0.00%       0.00%       0.00%
          1                           flop_count_sp  Floating Point Operations(Single Precisi           0           0           0
          1                       flop_count_sp_add  Floating Point Operations(Single Precisi           0           0           0
          1                       flop_count_sp_mul  Floating Point Operation(Single Precisio           0           0           0
          1                       flop_count_sp_fma  Floating Point Operations(Single Precisi           0           0           0
          1                           flop_count_dp  Floating Point Operations(Double Precisi           0           0           0
          1                       flop_count_dp_add  Floating Point Operations(Double Precisi           0           0           0
          1                       flop_count_dp_mul  Floating Point Operations(Double Precisi           0           0           0
          1                       flop_count_dp_fma  Floating Point Operations(Double Preciso           0           0           0
          1                    flop_count_sp_special  Floating Point Operations(Single Precisi           0           0           0
          1                        stall_inst_fetch  Issue Stall Reasons (Instructions Fetch)      40.32%      40.32%      40.32%
```

Figura 6: Execução de Modo Resumo para Métricas

2.1.4 Modo de rastro de Evento/Métrica

No modo Trace(Rastro) de Evento e métricas, os eventos e métricas são exibidos para cada execução de kernel. Por padrão, estes valores são agregados através de todas as unidades de GPU (–aggregate-mode on).[9]

Para executar este modo basta adicionar tanto a opção de evento/metrica (–events e –metrics) quanto a opção de Trace em gpu (–print-gpu-trace):

```
$ nvprof --events evento_1,evento_2 --print-gpu-trace ./seu_exec
$ nvprof --metrics metrica_1,metrica_2 --print-gpu-trace ./seu_exec
```

Por padrão, os eventos de multiprocessos são agregadossão agregados através de todos os multiprocessadores na GPU. Caso deseje exibir valores para cada unidade, deve ser especificado como opção de –aggregate-mode off assim como na Figura 7

```
[root@enxu testes]# nvprof --aggregate-mode off --events local_load --print-gpu-trace  ./async
[./async] - Starting...
==4516== NVPROF is profiling process 4516, command: ./async
time spent executing by the GPU: 79.53
CPU executed 44671 iterations while waiting for GPU to finish
==4516== Profiling application: ./async
==4516== Profiling result:
          Device         Context          Stream              Kernel  local_load (0)  local_load (1)  local_load (2)  local_load (3)  local_load (4)  lo
l_load (10)  local_load (11)  local_load (12)  local_load (13)
Tesla C2070 (0)             1                7  increment_kernel(int               0               0               0               0               0
          0               0               0               0
[root@enxu testes]#
```

Figura 7: Execução de Modo Rastro para Evento

O –aggregate-mode também pode ser aplicado a métricas. Porém, algumas métricas só estão disponíveis no modo agregado e outras estão disponíveis apenas no modo não agregado.

2.2 Limitando o dispositivo

Por padrão, o nvprof gera o perfil de todos os kernels executando em todos os dispositivos CUDA visíveis. Para limitar teste escopo, o nvprof pode ser utilizado com a seguinte opção:

```
$ nvprof --devices numero_do_device ./seu_exec
```

2.3 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.

2.3.1 Passos:

1.1. Adicione a Inicialização do perfil dentro de seu código.

```
cudaProfilerInitialize();
```

1.2. Adicione cudaProfilerStart() / cudaProfilerStop() no código indicando onde deve ser iniciado e finalizado o profiling

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

1.3. Desative o profiling no inicio da aplicação, para isso adicione o seguinte parâmetro na execução do Nvprof:

```
$ nvprof --profile-from-start off ./seu_exec
```

2.4 Exportando e Importando um perfil

O Nvprof permite que o usuário armazene informações de perfils em arquivos que possam ser futuramente lidos tanto pelo nvprof quanto pelo nvvp (Visual Profiler)[2].

2.4.1 Exportando:

Para exportar um arquivo utilizando o Nvprof você deve utilizar a opção "-o" no momento que for executar o nvprof e indicar o nome do arquivo no qual deve ser exportado o perfil:

```
$ nvprof -o profile.out ./seu_exec <argumentos da aplicaÃ§Ã£o>
```

Neste caso, o perfil será armazenado no arquivo profile.out.

2.4.2 Importando:

Para importar um perfil e gerar saidas textuais utilizando o nvprof, você deve utilizar a opção "-i" e indicar o nome do arquivo do qual deve ser lido as informações de perfil:

```
$ nvprof -i profile.out
$ nvprof --import-profile timeline.nvprof
```

2.5 Exportando um Perfil de Sistema Remoto

Como dito anteriormente, o Nvprof permite que o usuário armazene informações de perfils em arquivos que possam ser futuramente lidos tanto pelo nvprof quanto pelo nvvp (VIsual Profiler).[2] Esta característica é muito importante quando estiver fazendo acesso remoto (SSH) a uma máquina, pois permitira que o host tenha todo o suporte necessário para fazer a analise de desempenho de uma aplicação remota.

São três os principais perfis que podem ser exportados e eles serão apresentados a seguir.[3]

2.5.1 Exportando Linhas do tempo

Para exportar linhas do tempo execute o nvprof com a opção –export-profile passando o nome do perfil a ser gerado bem como o executável da aplicação a ser analisada. Esta opção pode ser usada em qualquer modo de profiling do nvprof.

```
$ nvprof - -export-profile timeline.nvprof ./seu_exec <argumentos da aplicac
```

Assim, os dados de perfil serão coletados em timeline.nvprof (Nome exemplo) e poderão ser visualizados caso sejam importados usando o nvprof ou o nvvp.

2.5.2 Exportando métricas e eventos

Você pode exportar métricas ou eventos de sua aplicação, para isso, execute o nvprof com as flags –metrics e –events

```
    $ nvprof - -metrics nome_da_metrica_01,nome_da_metrica_02 \
- -events nome_do_evento_01,nome_do_evento_02 -o \
metrics_events.nvprof ./seu_exec <argumentos da aplicaÃ§Ã£o>
```

Assim, os dados de perfil serão coletados em metrics_events.nvprof (Nome exemplo) e poderão ser visualizados caso sejam importados usando o nvprof ou o nvvp

2.5.3 Listando métricas e eventos

Para ver a lista de todas as métricas e eventos disponíveis/suportados na sua GPU NVIDIA digite respectivamente:

```
$ nvprof --query-metrics
$ nvprof --query-events
```

Para referências sobre as métricas acesse [5]

2.5.4 Limitando dispositivos e kernels

Quando estiver coletando eventos e métricas, por padrão o nvprof gera o perfil de todos os kernels executados em todos os dispositivos CUDA visíveis.[4] Para limitar o dispositivo ou os kernels os quais devem ser gerado o perfil pode ser adicionado as seguintes opções:

```
- -devices <ID do dispositivo>
- -kernels <Filtro do kernel>
```

Exemplo:

```
    $ nvprof --devices 0 --metrics ipc --kernels "1:foo:bar:2" \
--events local_load a.out
```

2.5.5 Exportando Analise Guiada para Kernel Individual

O Nvprof permite ainda coletar as métricas necessárias para o sistema de analise guiada para um kernel individual. O arquivo gerado, ao ser importado para o nvvp (Visual Profiler) permite ao sistema de analise guiada analisar o kernel e responder com oportunidades de otimização para aquele kernel. Para isso, o nvprof deve ser executado com a opção identificando qual kernel deve ser analisado (–kernel), seguido da opção –analysis-metrics para coletar as métricas necessárias.

```
    $ nvprof --kernels <especificacao_do_kerne> \
- -analysis-metrics -o analysis.nvprof \
./seu_exec <argumentos da aplicacao>
```

Assim, os dados de perfil serão coletados em analysis.nvprof (Nome exemplo) e poderão ser visualizados caso sejam importados usando o nvvp ou o próprio nvprof.

Referências

[1] https://devblogs.nvidia.com/parallelforall/cuda-pro-tip-nvprof-your-handy-universal-gpu-profiler/

[2] http://docs.nvidia.com/cuda/profiler-users-guide/index.html#export-import

[3] http://docs.nvidia.com/cuda/profiler-users-guide/index.html#collecting-remote-data

[4] http://docs.nvidia.com/cuda/profiler-users-guide/index.html#profiling-scope

[5] http://docs.nvidia.com/cuda/profiler-users-guide/index.html#metrics-reference

[6] http://docs.nvidia.com/cuda/profiler-users-guide/index.html#summary-mode

[7] http://docs.nvidia.com/cuda/profiler-users-guide/index.html#gpu-trace-and-api-trace-modes

[8] http://docs.nvidia.com/cuda/profiler-users-guide/

[9] http://docs.nvidia.com/cuda/profiler-users-guide/index.html#event-trace-mode