



Computational Science on Many-Core Architectures

360.252

Josef Weinbub
Karl Rupp



Institute for Microelectronics, TU Wien
<http://www.iue.tuwien.ac.at/>



Zoom Channel 621 2711 2607
Wednesday, October 25, 2023

Profiling and Debugging

NVPROF

- Command line profiler for CUDA applications
- GUI profiler available as well: NVVP

```
$> nvprof ./aa06697d.out
==32142== Profiling application: ./aa06697d.out
==32142== Profiling result:
```

	Type	Time(%)	Time	Calls	Avg	Min	Max	Name
GPU activities:		56.14%	2.0480us	1	2.0480us	2.0480us	2.0480us	transpose(double* , int)
		23.68%	864ns	1	864ns	864ns	864ns	[CUDA memcpy HtoD]
		20.18%	736ns	1	736ns	736ns	736ns	[CUDA memcpy DtoH]
API calls:		69.47%	184.21ms	1	184.21ms	184.21ms	184.21ms	cudaMalloc
		29.94%	79.390ms	1	79.390ms	79.390ms	79.390ms	cudaDeviceReset
		0.37%	989.24us	94	10.523us	8.3810us	63.136us	cuDeviceGetAttribute
....								

Profiling and Debugging

CUDA MEMCHECK

- Valgrind-port for CUDA devices
- Subtools:
 1. `memcheck`: Check for invalid memory access or non-free'd memory
 2. `racecheck`: Check for race conditions when accessing shared memory
 3. `synccheck`: Check for properly placed synchronizations
 4. `initcheck`: Check for access to uninitialized device memory

```
cuda-memcheck ./aa06697d.out
===== CUDA-MEMCHECK
===== Leaked 800 bytes at 0x7f9e86a00000
===== Saved host backtrace up to driver entry point at cudaMalloc time
===== Host Frame:/usr/lib/x86_64-linux-gnu/libcuda.so.1 (cuMemAlloc.v2 + 0x1b7) [0x2ba157]
===== Host Frame:./aa06697d.out [0x385d3]
===== Host Frame:./aa06697d.out [0xaf2b]
===== Host Frame:./aa06697d.out [0x48b98]
===== Host Frame:./aa06697d.out [0x672f]
===== Host Frame:./aa06697d.out [0x6352]
===== Host Frame:/lib/x86_64-linux-gnu/libc.so.6 (libc.start.main + 0xe7) [0x21bf7]
===== Host Frame:./aa06697d.out [0x60da]
=====
===== LEAK SUMMARY: 800 bytes leaked in 1 allocations
===== ERROR SUMMARY: 1 error
```

Profiling and Debugging

Reminder: Valgrind

- Checks for invalid memory accesses, non-free'd allocations, uninitialized memory, etc.
- Executes your code in a virtual environment with access monitoring
- Various subtools and options for deeper inspection
- Saves you A LOT of debugging time

```
valgrind ./aa06697d.out
==32397== Memcheck, a memory error detector
==32397== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==32397== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==32397== Command: ./aa06697d.out
==32397==
==32397== 800 bytes in 1 blocks are definitely lost in loss record 83 of 94
==32397==    at 0x4C31B0F: malloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==32397==    by 0x10E2E2: main (aa06697d.cu:37)
==32397==
==32397== LEAK SUMMARY:
==32397==    definitely lost: 800 bytes in 1 blocks
==32397==    indirectly lost: 0 bytes in 0 blocks
==32397==    possibly lost: 2,256 bytes in 15 blocks
==32397==    still reachable: 384,620 bytes in 78 blocks
==32397==    suppressed: 0 bytes in 0 blocks
==32397== Reachable blocks (those to which a pointer was found) are not shown.
==32397== To see them, rerun with: --leak-check=full --show-leak-kinds=all
==32397==
==32397== For counts of detected and suppressed errors, rerun with: -v
==32397== ERROR SUMMARY: 16 errors from 16 contexts (suppressed: 0 from 0)
```