

Tools for Debugging & Profiling

OpenACC Course 2016

Contents

What you will learn. Hopefully.

- OpenACC can greatly speedup porting to GPU
- But many details hidden from user
- Compiler makes assumptions
- Programmer makes mistakes
- ⇒ Insight into program needed

Contents

What you will learn. Hopefully.

- OpenACC can greatly speedup porting to GPU
- But many details hidden from user
- Compiler makes assumptions
- Programmer makes mistakes
- ⇒ Insight into program needed

Introduction

PGI Tools

Runtime Measurements

pgprof

NVIDIA Tools

cuda-memcheck

cuda-gdb

nvprof

Visual Profiler

Tasks

Task 1

Task 2

```
$ ./spmv
```

```
call to cuStreamSynchronize returned error 700: Illegal address during kernel  
↪ execution
```

- Where does error come from?
- Is it an error at all?
- ... and how do I find out?

- Building for debugging
 - g Add debug information to executable; adds overhead → program performs slower
Usually, in host code, -g has little impact.
 - ta=tesla:lineinfo Add information to assembly to relate instructions to source code (*light debug info*)
- Check compiler output: -Minfo=accel

PGI Runtime Measurements

For quick sanity checks

- Applications compiled with PGI compiler: Analyze via environment variables
- Maybe simplest/quickest check

PGI_ACC_TIME Lightweight profiler for time of data movement and kernels

PGI_ACC_NOTIFY Print information for GPU-related events.
Set to number, to print ...

- =1 ... kernel launches only
- =2 ... data transfers only
- =3 ... kernel launches and data transfers
- =4 ... region entry/exits only
- =5 ... region entry/exits and kernel launches
- =8 ... wait operations, synchronizations
- =16 ... (de)allocation of device memory

Usage: PGI_ACC_NOTIFY=3 ./app

```
task01 — ssh jureca -Y — jureca — ssh jureca -Y
launch CUDA kernel file=/homeb/zam/aherten/NVAL/Courses/OpenACC-2016/Debugging/task01/spmv.c function=main line=42 device=0 thr
eadid=1 num_gangs=65535 num_workers=4 vector_length=32 grid=65535 block=32x4 shared memory=1072
launch CUDA kernel file=/homeb/zam/aherten/NVAL/Courses/OpenACC-2016/Debugging/task01/spmv.c function=main line=42 device=0 thr
eadid=1 num_gangs=65535 num_workers=4 vector_length=32 grid=65535 block=32x4 shared memory=1072
launch CUDA kernel file=/homeb/zam/aherten/NVAL/Courses/OpenACC-2016/Debugging/task01/spmv.c function=main line=42 device=0 thr
eadid=1 num_gangs=65535 num_workers=4 vector_length=32 grid=65535 block=32x4 shared memory=1072
launch CUDA kernel file=/homeb/zam/aherten/NVAL/Courses/OpenACC-2016/Debugging/task01/spmv.c function=main line=42 device=0 thr
eadid=1 num_gangs=65535 num_workers=4 vector_length=32 grid=65535 block=32x4 shared memory=1072
launch CUDA kernel file=/homeb/zam/aherten/NVAL/Courses/OpenACC-2016/Debugging/task01/spmv.c function=main line=42 device=0 thr
eadid=1 num_gangs=65535 num_workers=4 vector_length=32 grid=65535 block=32x4 shared memory=1072
launch CUDA kernel file=/homeb/zam/aherten/NVAL/Courses/OpenACC-2016/Debugging/task01/spmv.c function=main line=42 device=0 thr
eadid=1 num_gangs=65535 num_workers=4 vector_length=32 grid=65535 block=32x4 shared memory=1072
launch CUDA kernel file=/homeb/zam/aherten/NVAL/Courses/OpenACC-2016/Debugging/task01/spmv.c function=main line=42 device=0 thr
eadid=1 num_gangs=65535 num_workers=4 vector_length=32 grid=65535 block=32x4 shared memory=1072
download CUDA data file=/homeb/zam/aherten/NVAL/Courses/OpenACC-2016/Debugging/task01/spmv.c function=main line=60 device=0 thr
eadid=1 variable=y bytes=14633352
download CUDA data file=/homeb/zam/aherten/NVAL/Courses/OpenACC-2016/Debugging/task01/spmv.c function=main line=60 device=0 thr
eadid=1 variable=y bytes=16777152
download CUDA data file=/homeb/zam/aherten/NVAL/Courses/OpenACC-2016/Debugging/task01/spmv.c function=main line=60 device=0 thr
eadid=1 variable=y bytes=16777152
download CUDA data file=/homeb/zam/aherten/NVAL/Courses/OpenACC-2016/Debugging/task01/spmv.c function=main line=60 device=0 thr
eadid=1 variable=y bytes=16777152
Runtime 0.045130 s.
aherten@jrc0002:~/NVAL/Courses/OpenACC-2016/Debugging/task01$ PGI_ACC_NOTIFY=3 ./spmv
```

PGPROF Graphical Performance Profiler

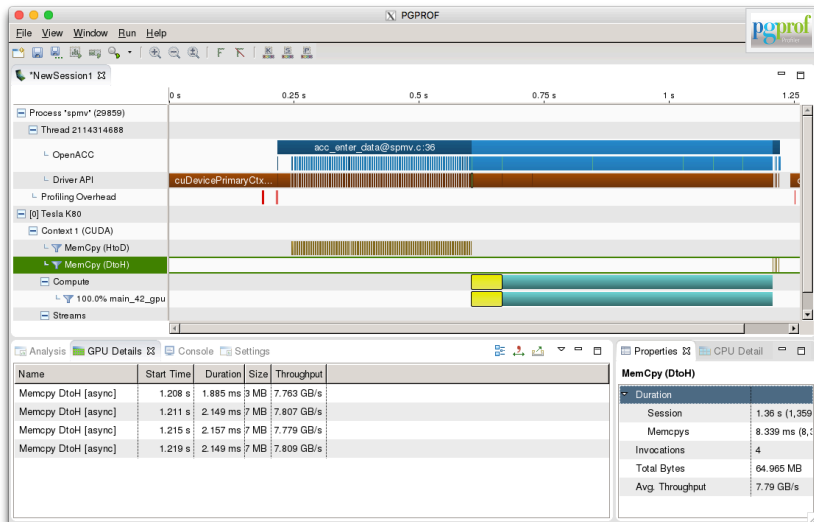
PGI's graphical profiler

- Graphical, interactive profiler
- Comes with PGI's compiler collection
- Nice visualizations, quick insight
- For OpenACC, OpenMP, CUDA
- Close to NVIDIA Visual Profiler

→ <https://www.pgroup.com/products/pgprof.htm>

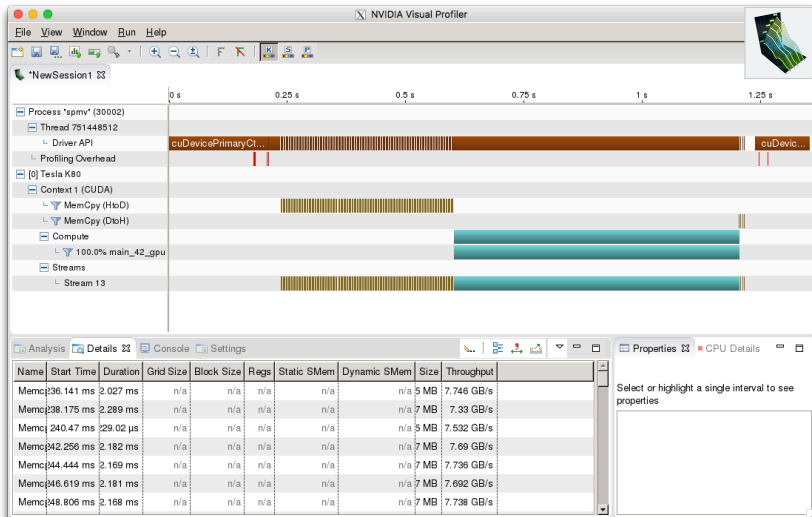
PGPROF Graphical Performance Profiler

PGI's graphical profiler



PGPROF Graphical Performance Profiler

NVIDIA Visual Profiler



- Memory error detector; similar to Valgrind's memcheck
- One of most helpful tools for error-finding
 - Out-of-bounds accesses
 - Kernels/API execution failures
 - Memory leaks
- Has sub-tools, via `cuda-memcheck --tool NAME`:
 - memcheck: Memory access checking (*default*)
 - racecheck: Shared memory hazard checking
 - Also: synccheck, initcheck
- Remember to compile program with debug information: `-g`

→ <http://docs.nvidia.com/cuda/cuda-memcheck/>

cuda-memcheck

Example

Start via `cuda-memcheck app`

```
task01 — ssh jureca -Y — jureca — ssh jureca -Y
aherten@jrl08:~/NVAL/Courses/OpenACC-2016/Debugging/task01$ srun cuda-memcheck spmv
=====
CUDA-MEMCHECK
=====
Invalid __global__ read of size 8
=====
at 0x00000348 in main_42_gpu
=====
by thread (26,3,0) in block (175,0,0)
=====
Address 0x23ac364310 is out of bounds
=====
Saved host backtrace up to driver entry point at kernel launch time
=====
Host Frame:/usr/lib64/nvidia/libcuda.so (cuLaunchKernel + 0x2cd) [0x15865d]
=====
Host Frame:/usr/local/software/jureca/Stages/2016a/software/PGI/16.3-GCC-4.9.3-2.25/Linux86-64/16.3/lib/libaccnmp.
so (_pgi_uacc_cuda_launch + 0x1b96) [0x15263]
=====
Host Frame:/usr/local/software/jureca/Stages/2016a/software/PGI/16.3-GCC-4.9.3-2.25/Linux86-64/16.3/lib/libaccgmp.
so (_pgi_uacc_launch + 0x210) [0x133fb]
=====
Host Frame:spmv [0x170b]
=====
Host Frame:/usr/lib64/libc.so.6 (__libc_start_main + 0xf5) [0x21b15]
=====
Host Frame:spmv [0xf29]
=====
Invalid __global__ read of size 8
=====
at 0x00000348 in main_42_gpu
=====
by thread (25,3,0) in block (175,0,0)
=====
Address 0x23ac364308 is out of bounds
=====
Saved host backtrace up to driver entry point at kernel launch time
=====
Host Frame:/usr/lib64/nvidia/libcuda.so (cuLaunchKernel + 0x2cd) [0x15865d]
=====
Host Frame:/usr/local/software/jureca/Stages/2016a/software/PGI/16.3-GCC-4.9.3-2.25/Linux86-64/16.3/lib/libaccnmp.
so (_pgi_uacc_cuda_launch + 0x1b96) [0x15263]
=====
Host Frame:/usr/local/software/jureca/Stages/2016a/software/PGI/16.3-GCC-4.9.3-2.25/Linux86-64/16.3/lib/libaccgmp.
so (_pgi_uacc_launch + 0x210) [0x133fb]
```

cuda-gdb

Symbolic debugger

- Powerful symbolic debugger for CUDA code
- Built on top of `gdb`
- Full usage: own course needed

cuda-gdb

Symbolic debugger

- Powerful symbolic debugger for CUDA code
- Built on top of `gdb`
- Full usage: own course needed

```
cuda-gdb 101
```

```
run Starts application, give arguments with set args 1 2 ...
```

```
break L Create breakpoint
```

L: function name, line number LN, or FILE:LN

```
continue Continue running
```

```
print i Print content of i
```

```
info locals Print all currently set variables
```

```
info cuda threads Print current thread configuration
```

```
cuda thread N Switch context to thread number N
```

→ [cheat sheet](#)

cuda-gdb

Symbolic debugger

- Powerful symbolic debugger for CUDA code
- Built on top of `gdb`
- Full usage: own course needed

```
cuda-gdb 101
```

```
run Starts application, give arguments with set args 1 2 ...
```

```
break L Create breakpoint
```

L: function name, line number LN, or FILE:LN

```
continue Continue running
```

```
print i Print content of i
```

```
info locals Print all currently set variables
```

```
info cuda threads Print current thread configuration
```

```
cuda thread N Switch context to thread number N
```

→ *cheat sheet*

→ <http://docs.nvidia.com/cuda/cuda-gdb/>

cuda-gdb can be used for OpenACC as well!

- Problem: Name of OpenACC-generated kernel?

→ Recipe: `strings ./app | grep .*_gpu | sort | uniq`

`strings ./app` Print occurrences of ≥ 4 printable characters

`grep .*_gpu` Search for `_gpu` line endings

`sort | uniq` Eliminate duplicates from list

- Examples of kernel names

Pattern: `function_line_gpu`

C `main_42_gpu`

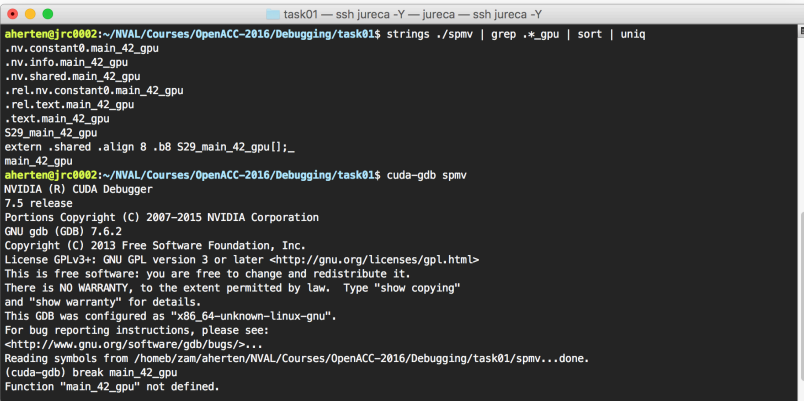
Fortran `spmv_26_gpu`

cuda-gdb

Example

Start via `cuda-gdb app` → run

Set breakpoint with `break func` or `break L` or `break file.c:L`



```
task01 — ssh jureca -Y — jureca — ssh jureca -Y
aherten@jrc0002:~/NVAL/Courses/OpenACC-2016/Debugging/task01$ strings ./spmv | grep .*_gpu | sort | uniq
.nv.constant0.main_42_gpu
.nv.info.main_42_gpu
.nv.shared.main_42_gpu
.rel.nv.constant0.main_42_gpu
.rel.text.main_42_gpu
.text.main_42_gpu
S29_main_42_gpu
extern .shared .align 8 .b8 S29_main_42_gpu[];_
main_42_gpu
aherten@jrc0002:~/NVAL/Courses/OpenACC-2016/Debugging/task01$ cuda-gdb spmv
NVIDIA (R) CUDA Debugger
7.5 release
Portions Copyright (C) 2007-2015 NVIDIA Corporation
GNU gdb (GDB) 7.6.2
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-unknown-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/zam/aherten/NVAL/Courses/OpenACC-2016/Debugging/task01/spmv...done.
(cuda-gdb) break main_42_gpu
Function "main_42_gpu" not defined.
```

- Profiles CUDA kernels and API calls; also CPU code!
 - Suitable for OpenACC as well
 - pgprof: Very similar to nvprof, but different default options
 - Generate performance reports, timelines; measure events and metrics
- ⇒ Powerful complete tool for GPU application analysis
- <http://docs.nvidia.com/cuda/profiler-users-guide/>

nvprof

Example

Start via nvprof `./app`

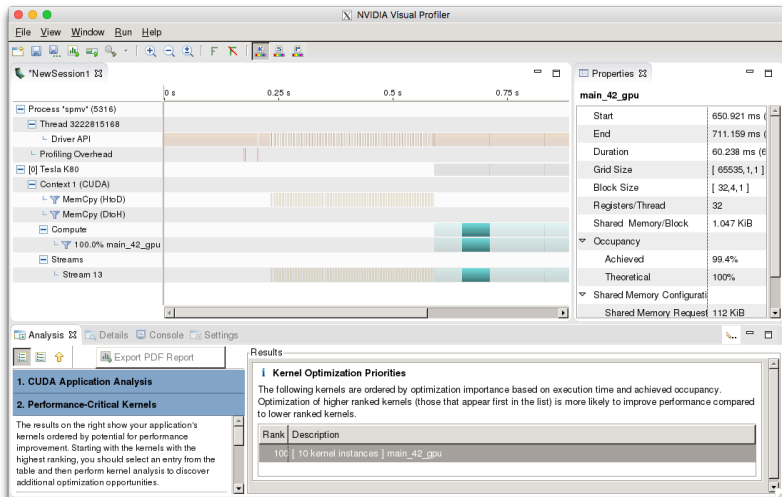
```
task01 — ssh jureca -Y — jureca — ssh jureca -Y
aherten@jrl08:~/NVAL/Courses/OpenACC-2016/Debugging/task01$ srun nvprof ./spmv
==5148== NVPROF is profiling process 5148, command: ./spmv
Runtime 0.060289 s.
==5148== Profiling application: ./spmv
==5148== Profiling result:
Time(%)   Time      Calls      Avg      Min      Max  Name
61.77%   602.58ms      10   60.258ms  60.235ms  60.278ms  main_42_gpu
37.36%   364.44ms     168   2.1693ms  226.81us  2.2910ms  [CUDA memcpy HtoD]
 0.86%    8.4285ms       4   2.1071ms  1.9034ms  2.1832ms  [CUDA memcpy DtoH]

==5148== API calls:
Time(%)   Time      Calls      Avg      Min      Max  Name
59.46%   607.37ms      12   50.614ms  2.1370ms  60.288ms  cuStreamSynchronize
19.41%   198.23ms       1   198.23ms  198.23ms  198.23ms  cuDevicePrimaryCtxRetain
10.43%   106.53ms       1   106.53ms  106.53ms  106.53ms  cuDevicePrimaryCtxRelease
 6.96%    71.118ms     172    413.48us  1.4320us  2.1914ms  cuEventSynchronize
 2.38%    24.332ms       1    24.332ms  24.332ms  24.332ms  cuMemHostAlloc
 0.87%    8.8901ms       1    8.8901ms  8.8901ms  8.8901ms  cuMemFreeHost
 0.28%    2.8180ms       6    469.67us  163.53us  1.1385ms  cuMemAlloc
 0.10%    1.0364ms     168    6.1680us  5.3790us  29.102us  cuMemcpyHtoDAsync
 0.05%    555.35us       1    555.35us  555.35us  555.35us  cuMemAllocHost
 0.03%    300.07us     174    1.7240us  1.4480us  3.1320us  cuEventRecord
 0.01%    150.53us      10    15.053us  6.9240us  39.904us  cuLaunchKernel
 0.01%    111.61us       1    111.61us  111.61us  111.61us  cuModuleLoadData
 0.00%    34.647us       4     8.6610us  4.5880us  18.862us  cuMemcpyDtoHAsync
 0.00%    16.987us       1     16.987us  16.987us  16.987us  cuStreamCreate
```

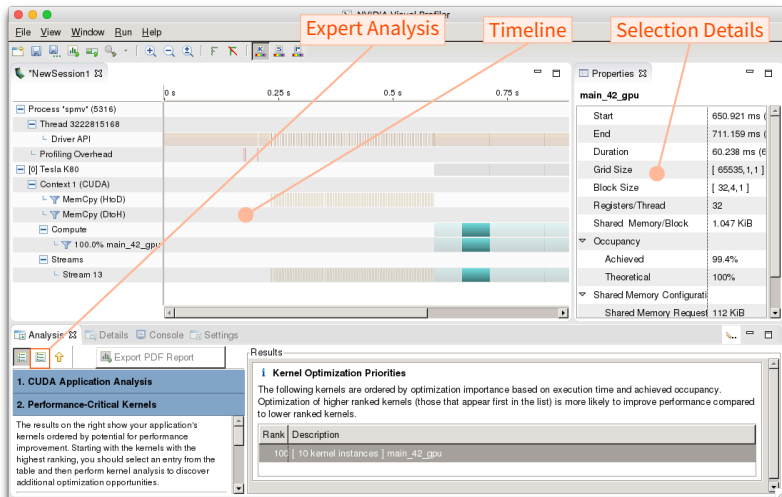
- Timeline view of all things GPU (API calls, kernels, memory)
→ study stages and interplay of application
- View launch and run configurations
- Guided and unguided analysis, with (among others):
 - Performance limiters
 - Kernel and execution properties
 - Memory access patterns

→ <https://developer.nvidia.com/nvidia-visual-profiler>

Start via nvvp → *File* ↪ *New Session*



Start via nvvp → *File* ↪ *New Session*

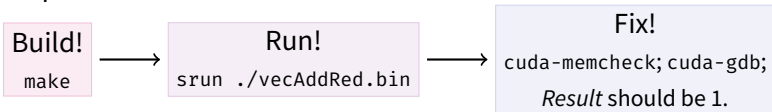


Task 1

- Location of tasks: *ACCOUNT*/Course/Debugging/
- Tasks available in C and Fortran

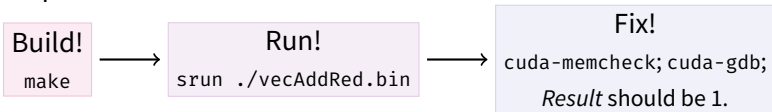
Task 1

- Location of tasks: *ACCOUNT/Course/Debugging/*
- Tasks available in C and Fortran
- **Task 1:** Vector addition and reduction: $\vec{a} = \vec{b} + \vec{c} \rightarrow \gamma = \sum_i c_i$
- Steps



Task 1

- Location of tasks: *ACCOUNT/Course/Debugging/*
- Tasks available in C and Fortran
- **Task 1:** Vector addition and reduction: $\vec{a} = \vec{b} + \vec{c} \rightarrow \gamma = \sum_i c_i$
- Steps



JURECA Getting Started

```
module load PGI CUDA
salloc --reservation=openacc --partition=gpus --nodes=1 --time=1:30:00
↪ --gres=mem128,gpu:4
srun cuda-memcheck ./vecAddRed.bin
```

- **Task 2:** Sparse Matrix-Vector Product (*SpMV*): $\vec{x} = \mathbf{A}\vec{y}$

- **Task 2:** Sparse Matrix-Vector Product (*SpMV*): $\vec{x} = \mathbf{A}\vec{y}$
- CSR data layout

- **Task 2:** Sparse Matrix-Vector Product (*SpMV*): $\vec{x} = \mathbf{A}\vec{y}$
- CSR data layout

	0	1	2	3	4
0	-2	1	0	0	0
1	1	-2	1	0	0
2	0	1	-2	1	0
3	0	0	1	-2	1
4	0	0	0	1	-2

- **Task 2:** Sparse Matrix-Vector Product ($SpMV$): $\vec{x} = \mathbf{A}\vec{y}$
- CSR data layout

	0	1	2	3	4
0	-2	1	0	0	0
1	1	-2	1	0	0
2	0	1	-2	1	0
3	0	0	1	-2	1
4	0	0	0	1	-2

	0	1	2	3	4
0	-2	1			
1	1	-2	1		
2		1	-2	1	
3			1	-2	1
4				1	-2

- Task 2: Sparse Matrix-Vector Product ($SpMV$): $\vec{x} = \mathbf{A}\vec{y}$
- CSR data layout

	0	1	2	3	4
0	-2	1	0	0	0
1	1	-2	1	0	0
2	0	1	-2	1	0
3	0	0	1	-2	1
4	0	0	0	1	-2

	0	1	2	3	4
0	-2	1			
1	1	-2	1		
2		1	-2	1	
3			1	-2	1
4				1	-2

val

-2	1	1	-2	1	1	-2	1	1	-2	1	1	-2
----	---	---	----	---	---	----	---	---	----	---	---	----

- Task 2: Sparse Matrix-Vector Product ($SpMV$): $\vec{x} = \mathbf{A}\vec{y}$
- CSR data layout

	0	1	2	3	4
0	-2	1	0	0	0
1	1	-2	1	0	0
2	0	1	-2	1	0
3	0	0	1	-2	1
4	0	0	0	1	-2

	0	1	2	3	4
0	-2	1			
1	1	-2	1		
2		1	-2	1	
3			1	-2	1
4				1	-2

val

-2	1	1	-2	1	1	-2	1	1	-2	1	1	-2
----	---	---	----	---	---	----	---	---	----	---	---	----

- Task 2: Sparse Matrix-Vector Product ($SpMV$): $\vec{x} = \mathbf{A}\vec{y}$
- CSR data layout

	0	1	2	3	4
0	-2	1	0	0	0
1	1	-2	1	0	0
2	0	1	-2	1	0
3	0	0	1	-2	1
4	0	0	0	1	-2

	0	1	2	3	4
0	-2	1			
1	1	-2	1		
2		1	-2	1	
3			1	-2	1
4				1	-2

col_ptr

0	1	0	1	2	1	2	3	2	3	4	3	4
---	---	---	---	---	---	---	---	---	---	---	---	---

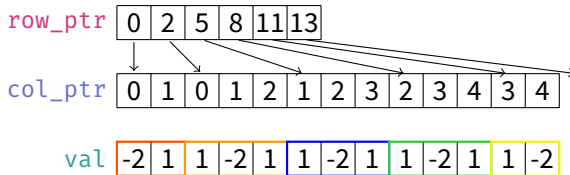
val

-2	1	1	-2	1	1	-2	1	1	-2	1	1	-2
----	---	---	----	---	---	----	---	---	----	---	---	----

- Task 2: Sparse Matrix-Vector Product ($SpMV$): $\vec{x} = \mathbf{A}\vec{y}$
- CSR data layout

	0	1	2	3	4
0	-2	1	0	0	0
1	1	-2	1	0	0
2	0	1	-2	1	0
3	0	0	1	-2	1
4	0	0	0	1	-2

	0	1	2	3	4
0	-2	1			
1	1	-2	1		
2		1	-2	1	
3			1	-2	1
4				1	-2



- **Task 2:** Sparse Matrix-Vector Product (*SpMV*): $\vec{x} = \mathbf{A}\vec{y}$
- CSR data layout
- Build! \longrightarrow Run! \longrightarrow Fix!

- **Task 2:** Sparse Matrix-Vector Product ($SpMV$): $\vec{x} = \mathbf{A}\vec{y}$
- CSR data layout
- Build! \longrightarrow Run! \longrightarrow Fix!

JURECA Getting Started

```
module load PGI CUDA
salloc --reservation=openacc --partition=gpus --nodes=1 --time=1:30:00
↪ --gres=mem128,gpu:4
srun cuda-memcheck ./spmv.bin
```

- All the CUDA debugging and performance measurement tools work
 - pgprof
 - cuda-memcheck
 - cuda-gdb
 - nvprof
 - Visual Profiler
- Sometimes, a little digging is needed to find automatically-generated function names

- All the CUDA debugging and performance measurement tools work
 - pgprof
 - cuda-memcheck
 - cuda-gdb
 - nvprof
 - Visual Profiler
- Sometimes, a little digging is needed to find automatically-generated function names

Happy Debugging!
a.herten@fz-juelich.de