OpenACC Tools

Sandra Wienke, M.Sc.
wienke@rz.rwth-aachen.de
Center for Computing and Communication
RWTH Aachen University

PPCES, March 2013

Agenda

- **IDE**
  - → Eclipse
- **Debugging**
  - → Tips & Tricks
- **Profiling**
  - → NVIDIA Visual Profiler

IDE - Eclipse

- **Eclipse (+ Plugins): IDE for GPU programming**
  - → Parallel Nsight: OpenACC profiling, CUDA syntax highlighting/debugging/profiling
  - → Parallel Tools Platform (PTP): OpenACC programming

- **Using Eclipse**
  - → Load your needed modules first, e.g. `module switch intel pgi` (for OpenACC)
  - → Default Eclipse version in RWTH Cluster environment (not able to do GPU profiling or syntax highlighting): `module load MISC eclipse; eclipse &`
  - → Parallel Nsight in RWTH Cluster environment: `module load cuda; nsight` (or download from http://www.nvidia.com/object/nsight.html)
  - → PTP: Download from http://www.eclipse.org/ptp/ , extract and start `eclipse &`

recommended

For the OpenACC labs, we put Eclipse+PTP into your Home directory.

IDE - Eclipse

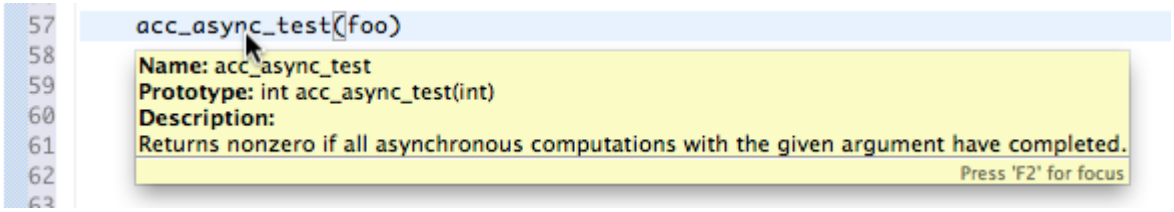
- **Create Makefile projects with Eclipse**
  1. Start `eclipse` and chose workspace
  2. File → New → Makefile Project with Existing code
  3. Chose file directory of your project
  4. [If CUDA is used: Toolchain: CUDA Toolkit 5.0]
  5. Create Makefile targets (Makefile Tab in right pane), e.g. release, run
  6. Double click on Makefile target for execution

- **PTP & OpenACC**
  - → Doc: http://help.eclipse.org/juno/topic/org.eclipse.ptp.doc.user/html/toc.html
  - → Recognizing PGI compiler errors: http://help.eclipse.org/juno/topic/org.eclipse.ptp.doc.user/html/errorParsers.html?cp=51_8
  - → Running PTP OpenACC tools: http://help.eclipse.org/juno/topic/org.eclipse.ptp.pldt.doc.user/html/runOpenACC.html?cp=51_14_8

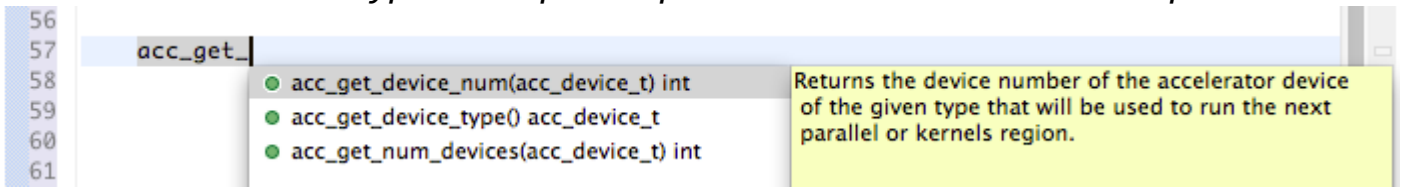
IDE - Eclipse

- **PTP & OpenACC** (taken from http://help.eclipse.org/juno/topic/org.eclipse.ptp.pldt.doc.user/html/runOpenACC.html?cp=51_14_8)

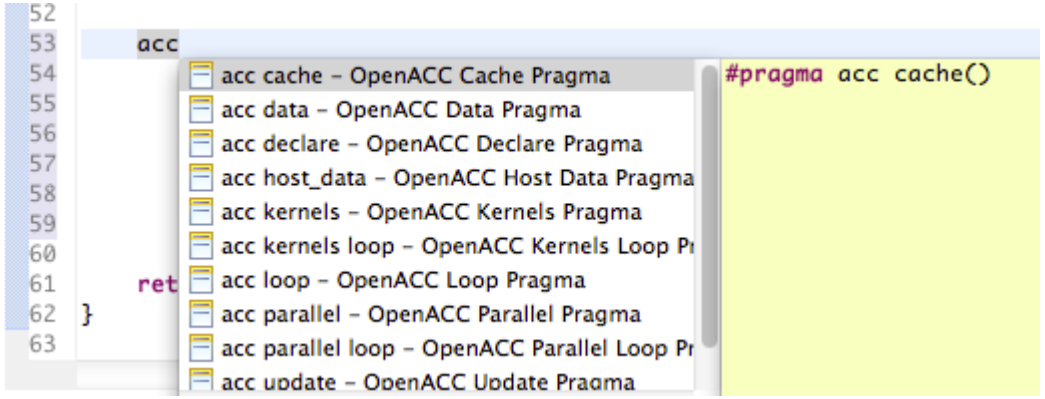
  → Hover help: *In the editor, hover an OpenACC API.*



  → Content Assist: *Type incomplete OpenACC function and hit Ctrl-Space.*



  → Code Templates: *Type* `acc` *and hit Ctrl-Space* *twice*.



Agenda

- **IDE**
  - → Eclipse
- **Debugging**
  - → Tips & Tricks
- **Profiling**
  - → NVIDIA Visual Profiler

Debugging

- **Debugging host code as usual (e.g. using TotwlView)**
- **Debugging GPU kernels requires special tools**
  - → CUDA debuggers available
  - → OpenACC debuggers (for PGI compiler) not available ☹

RWTH Cluster environment:
`module load totalview`

- **Tips & Tricks**
  - → Set **`ACC_NOTIFY`** to nonnegative integer to get a command line message
    - →=1: on launch of kernels
    - →=2: on moving data between device & host
    - →=3: 1 & 2
  - → Set **`PGI_ACC_DEBUG=1`** to get detailed debug information

Agenda

- **IDE**
  - → Eclipse
- **Debugging**
  - → Tips & Tricks
- **Profiling**
  - → NVIDIA Visual Profiler

Profiling

- **Profiling = Analyze behavior of application during runtime**
  - → e.g. runtime of functions, memory throughput
- **Command line: `export PGI_ACC_TIME=1`**
- **NVIDIA Visual Profiler for CUDA, OpenACC & OpenCL codes**
  - → Profiles only GPU data movement & computation (not host code)
  - → Use stand-alone version (**`module load cuda; nvvp &`**)

    or Eclipse plugin (see IDE Parallel Nsight)
    - → Start Nsight
    - → Use the release target of Makefile to create executable
    - → Press profile button (watch)
    - → Proceed (even if errors)
    - → Switch perspective
    - → Ouput/ interpretation see chapter NVIDIA Visual Profiler (next slide)

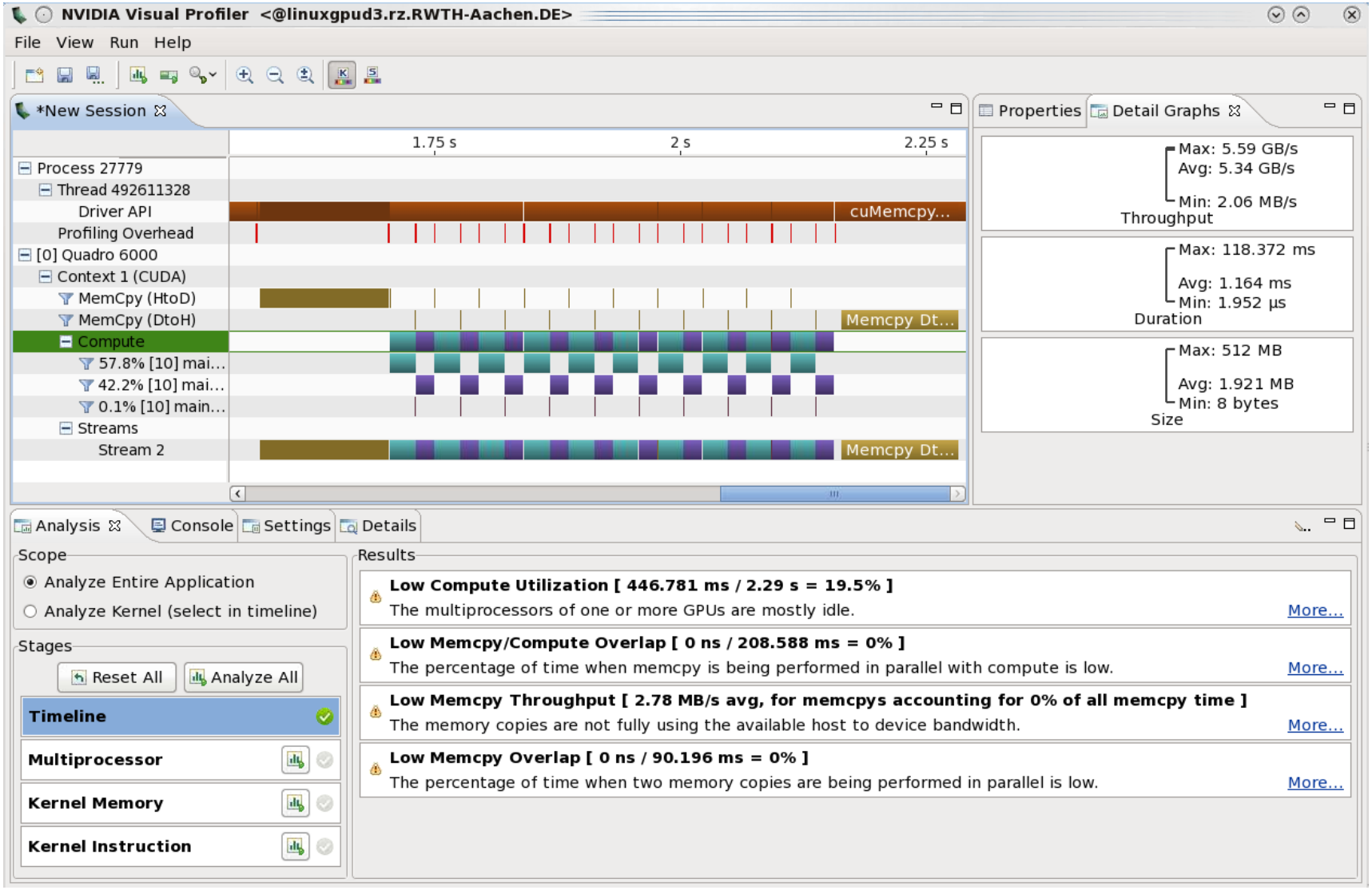
Profiling

RWTH AACHEN UNIVERSITY

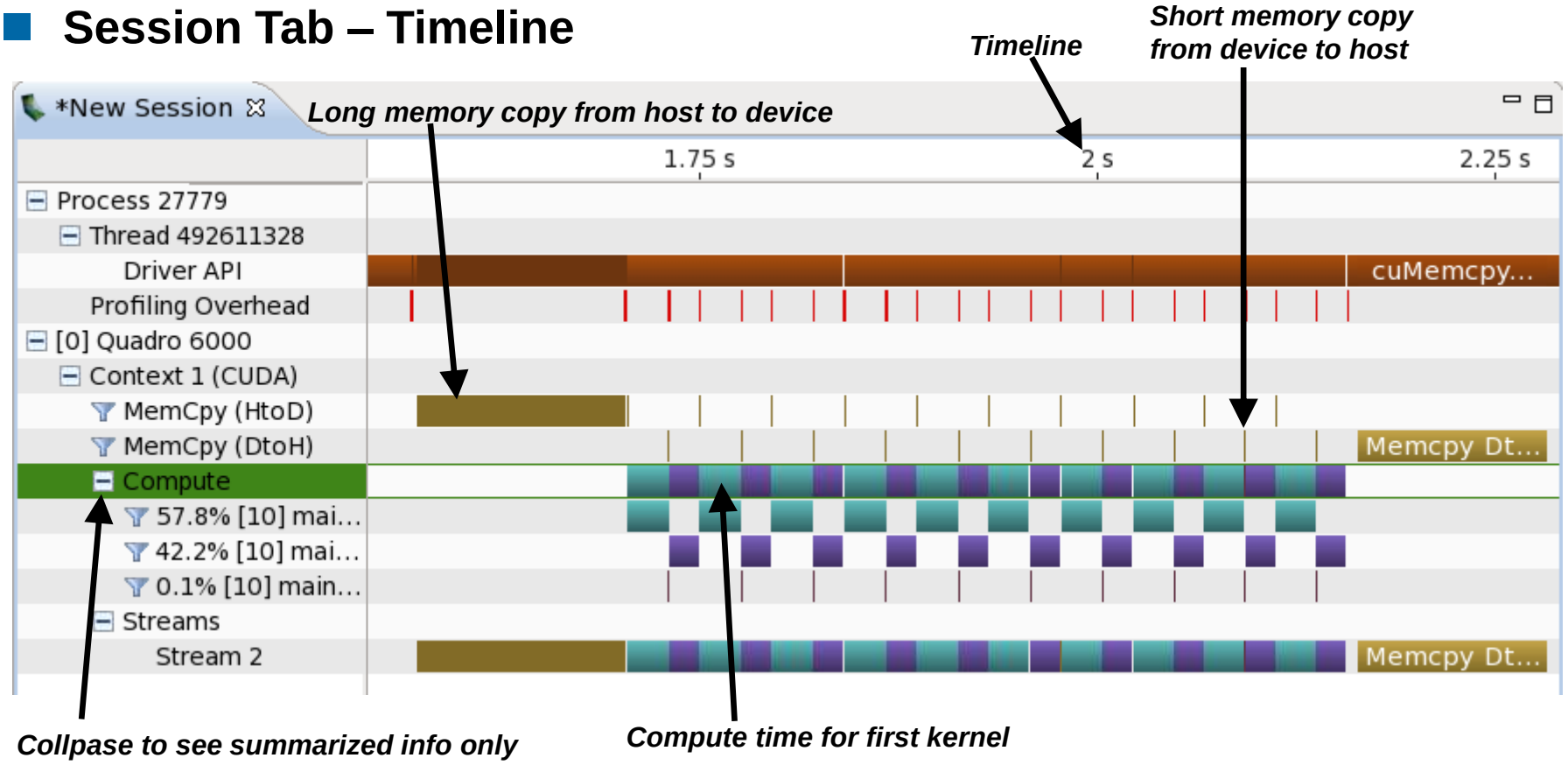
Use the NVIDIA Visual Profiler

1. Make sure that you have an executable program & start the profiler
2. Select `File` → `New Session`
3. Chose your executable as file
   - → Specify arguments (if needed)
   - → Specify environment variables (if needed)
4. If you want to shorten the execution time, set a timeout limit
5. `Finish` the session configuration & wait for results

Profiling


NVIDIA Visual Profiler <@linuxgpud3.rz.RWTH-Aachen.DE>
File View Run Help
*New Session
1.75 s
2 s
2.25 s
Process 27779
Thread 492611328
Driver API
cuMemcpy...
Profiling Overhead
[0] Quadro 6000
Context 1 (CUDA)
MemCpy (HtoD)
MemCpy (DtoH)
Memcpy Dt...
Compute
57.8% [10] mai...
42.2% [10] mai...
0.1% [10] main...
Streams
Stream 2
Memcpy Dt...
Properties
Detail Graphs
Max: 5.59 GB/s
Avg: 5.34 GB/s
Min: 2.06 MB/s
Throughput
Max: 118.372 ms
Avg: 1.164 ms
Min: 1.952 µs
Duration
Max: 512 MB
Avg: 1.921 MB
Min: 8 bytes
Size
Analysis
Console
Settings
Details
Scope
Analyze Entire Application
Analyze Kernel (select in timeline)
Stages
Reset All
Analyze All
Timeline
Multiprocessor
Kernel Memory
Kernel Instruction
Results
Low Compute Utilization [446.781 ms / 2.29 s = 19.5%]
The multiprocessors of one or more GPUs are mostly idle.
More...
Low Memcpy/Compute Overlap [0 ns / 208.588 ms = 0%]
The percentage of time when memcpy is being performed in parallel with compute is low.
More...
Low Memcpy Throughput [2.78 MB/s avg, for memcpys accounting for 0% of all memcpy time]
The memory copies are not fully using the available host to device bandwidth.
More...
Low Memcpy Overlap [0 ns / 90.196 ms = 0%]
The percentage of time when two memory copies are being performed in parallel is low.
More...

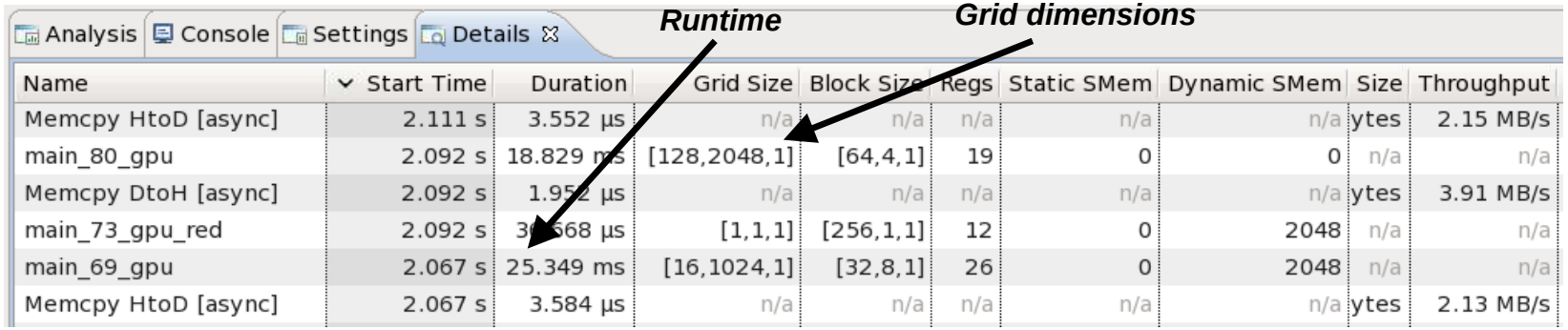
Session Tab – Timeline



→ Is data only transfered when needed?

→ Which kernel does need the most time?

- **Analysis Tab**
  - → Gives hints for optimization (not always useful)
- **Details Tab**
  - → Switch from Analysis Tab to the Details Tab



→ On right hand side, activate summary view