



**TECHNISCHE  
UNIVERSITÄT  
DRESDEN**

Center for Information Services and High Performance Computing (ZIH)

# **In-Depth Performance Analysis for OpenACC/CUDA/OpenCL Applications with Score-P and Vampir**

**Hands-on-Lab @ GTC2015**



**Guido Juckeland ([guido.juckeland@tu-dresden.de](mailto:guido.juckeland@tu-dresden.de))**



Center for Information Services &  
High Performance Computing



- Motivation
- Performance Analysis 101
- Generating Traces with Score-P
- Visualizing Traces with Vampir
- Special Treat: OpenACC Tracing
- Looking a Little Deeper

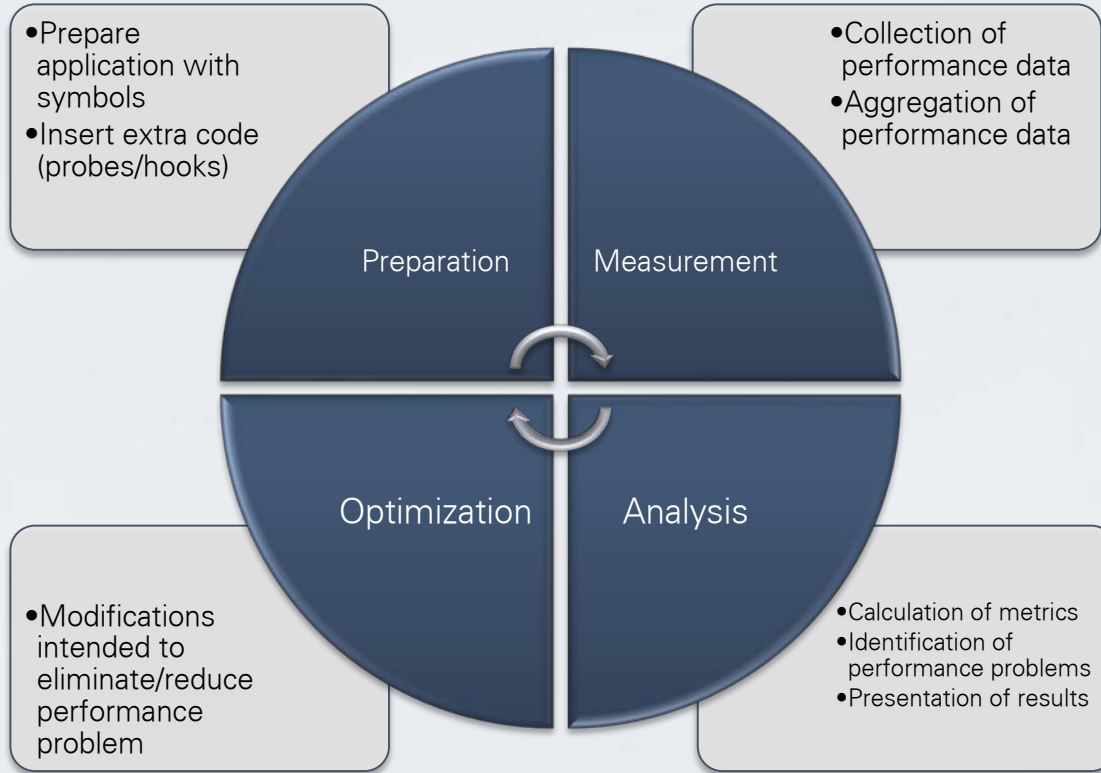


# Motivation



**Guido Juckeland ([guido.juckeland@tu-dresden.de](mailto:guido.juckeland@tu-dresden.de))**







TECHNISCHE  
UNIVERSITÄT  
DRESDEN

Center for Information Services and High Performance Computing (ZIH)

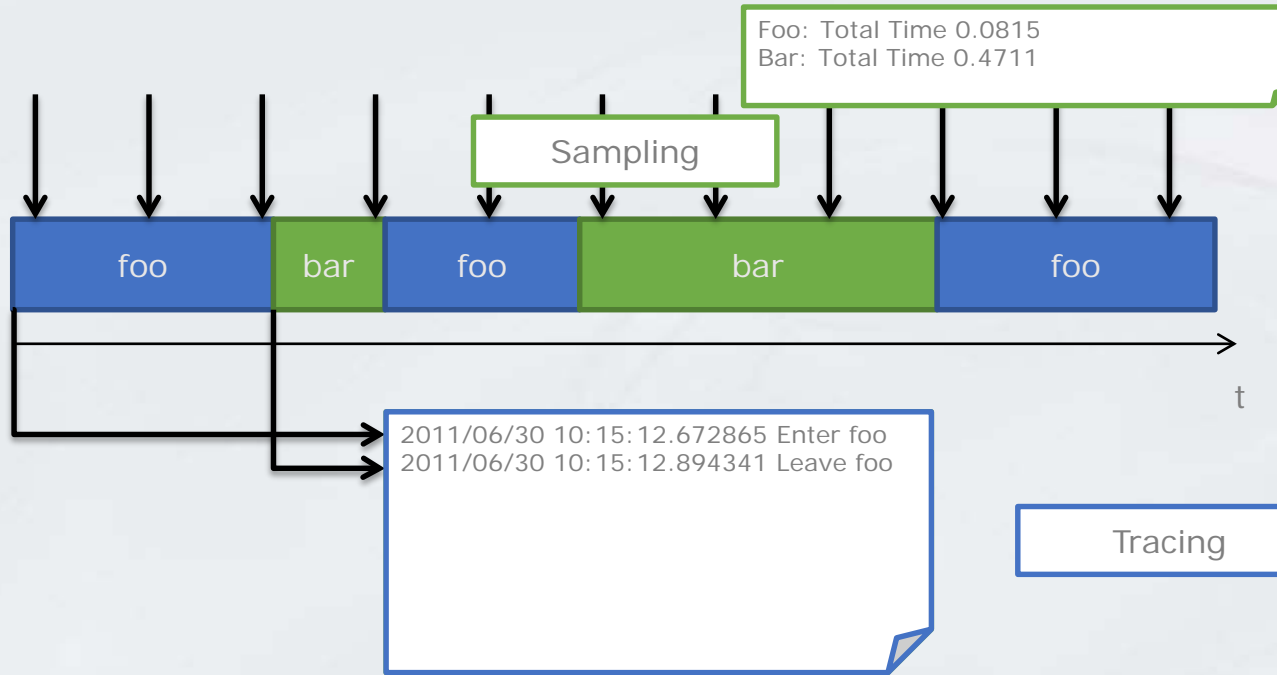
# Performance Analysis 101



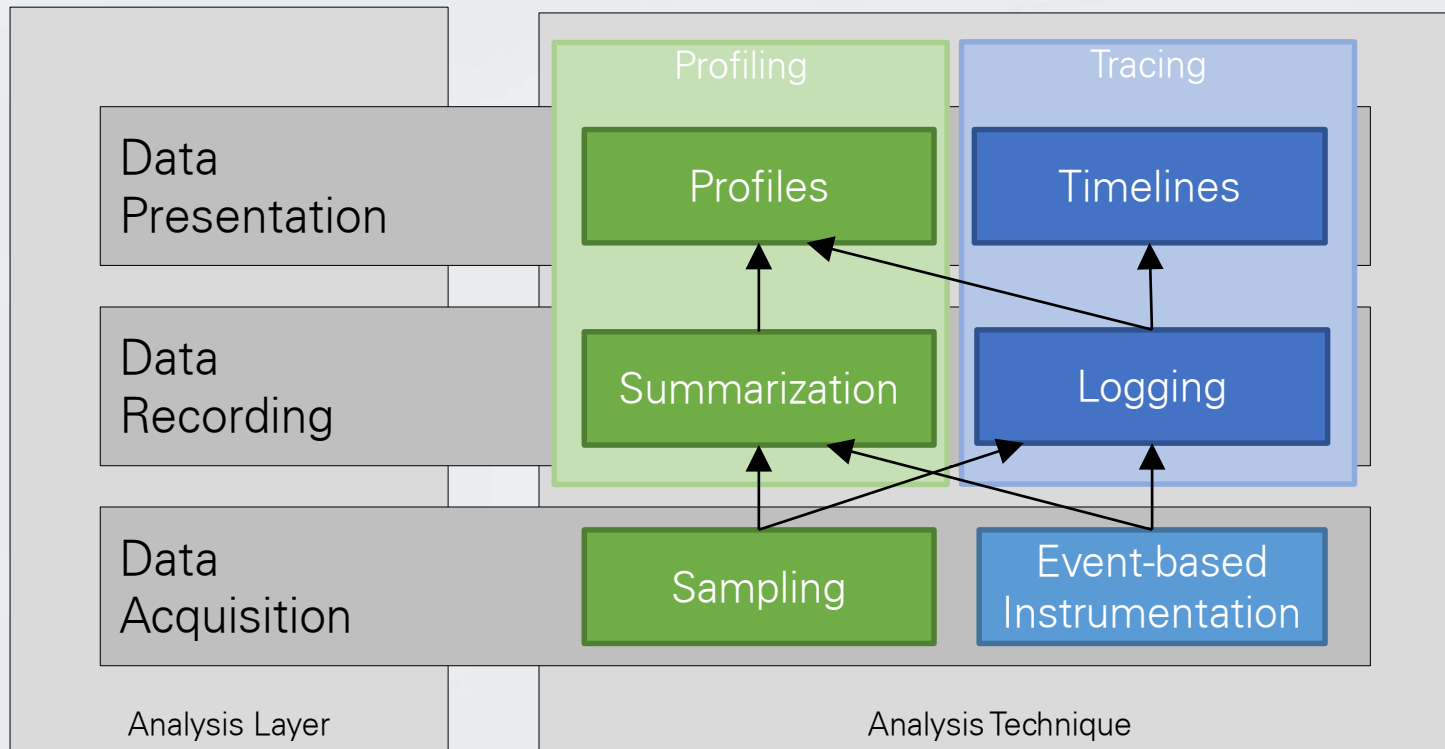
Guido Juckeland ([guido.juckeland@tu-dresden.de](mailto:guido.juckeland@tu-dresden.de))



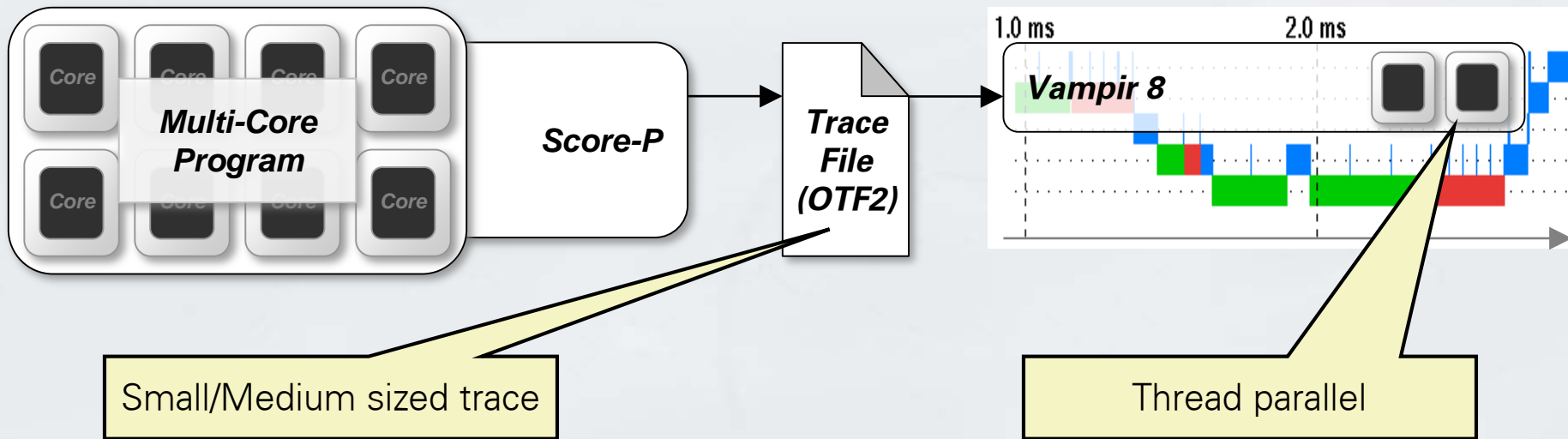
# Sampling vs. Tracing

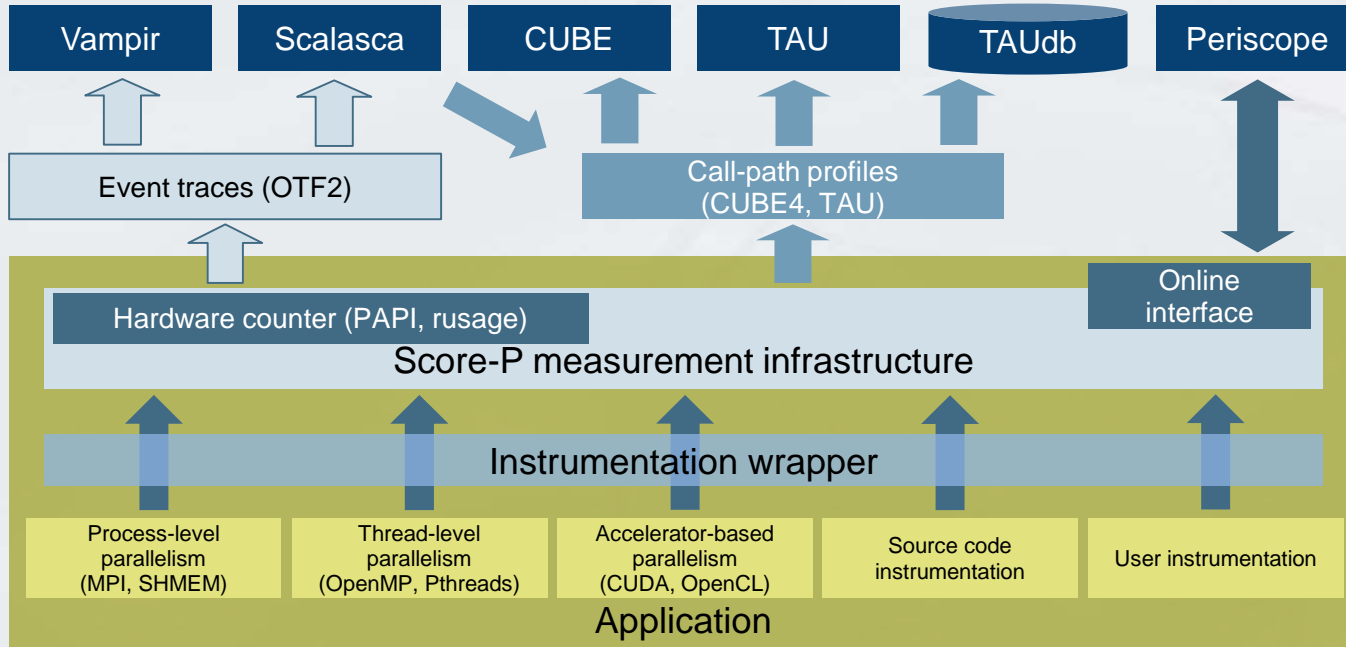


# Terms Used and How They Connect



# Score-P/Vampir Workflow for Small-Medium Sized Applications





# Partners



- Forschungszentrum Jülich, Germany
- German Research School for Simulation Sciences, Aachen, Germany
- Gesellschaft für numerische Simulation mbH Braunschweig, Germany
- RWTH Aachen, Germany
- Technische Universität Dresden, Germany
- Technische Universität München, Germany
- University of Oregon, Eugene, USA



UNIVERSITY OF OREGON





TECHNISCHE  
UNIVERSITÄT  
DRESDEN

Center for Information Services and High Performance Computing (ZIH)

# Hands-on: CUDA Tracing in Your Own AWS Instance



Guido Juckeland ([guido.juckeland@tu-dresden.de](mailto:guido.juckeland@tu-dresden.de))



Center for Information Services &  
High Performance Computing



- Navigate to [nvlabs.qwiklab.com](http://nvlabs.qwiklab.com)
- Login or create a new account
- Select the “[Instructor-Led Hands-on Labs](#)” class
- Find the lab called “[Analysis for OpenACC/CUDA/OpenCL Applications with Score-P and Vampir \(S5721 - GTC 2015\)](#)” and click [Start](#)
- After a short wait, lab instance connection information will be shown
- Please ask Lab Assistants for help!

# Performance Analysis Steps

---



1. Reference preparation for validation
- 2. Program instrumentation**
- 3. Event trace collection**
- 4. Event trace examination & analysis**

# Start a Terminal



NoMachine - Connection to ec2-54-67-45-48.us-west-1.compute.amazonaws.com

Terminal File Edit View Search Terminal Help

ubuntu@ip-172-31-14-255: ~

```
ScoreP version 1.4 loaded
SCOREP_ROOT=/opt/scorep
ubuntu@ip-172-31-14-255:~$
```

- Go to CUDA Example

```
% cd codes/cuda
```

- Compile

```
% make  
scorep --cuda /usr/local/anaconda/bin/mpicxx -Icommon/inc  
-o simpleMPI_mpi.o -c simpleMPI.cpp  
scorep --cuda "/usr/local/cuda-6.5"/bin/nvcc -ccbin g++ -Icommon/inc  
-m64 -gencode arch=compute_30,code=sm_30 -gencode arch=compute_35,  
code=sm_35 -gencode arch=compute_37,code=sm_37 -gencode  
arch=compute_50,code=sm_50 -gencode arch=compute_50,code=compute_50  
-o simpleMPI.o -c simpleMPI.cu  
scorep --cuda /usr/local/anaconda/bin/mpicxx -o simpleMPI  
simpleMPI_mpi.o simpleMPI.o -L"/usr/local/cuda-6.5"/lib64 -lcudart
```

## ● Run

```
% mpiexec -np 4 ./simpleMPI  
  
Running on 4 nodes  
Average of square roots is: 0.667305  
PASSED
```

## ● Find Tracefile appearing

```
% ls  
  
Makefile                simpleMPI                simpleMPI_mpi.o  
NsightEclipse.xml      simpleMPI.cpp           simpleMPI.o  
readme.txt              simpleMPI.cu  
scorep-20150311_2045_907655747320 simpleMPI.h
```

## Score-P performance monitor loaded on login

- Done via an environment module
- Also sets the following environment variables (it would be up to you)

```
% export SCOREP_ENABLE_TRACING=true  
% export SCOREP_ENABLE_PROFILING=false  
% export SCOREP_OPENCL_ENABLE=true  
% export SCOREP_CUDA_ENABLE=driver, kernel, memcpy, flushatexit  
% export SCOREP_OPENACC_ENABLE=true
```

### Makefile modified to instrument application

- Using **scorep** compiler wrapper
- Before:

```
NVCC := $(CUDA_PATH)/bin/nvcc -ccbin $(GCC)
MPICXX ?= $(shell which mpicxx 2>/dev/null)
```

- After:

```
NVCC := scorep --cuda $(CUDA_PATH)/bin/nvcc -ccbin $(GCC)
MPICXX ?= scorep --cuda $(shell which mpicxx 2>/dev/null)
```



TECHNISCHE  
UNIVERSITÄT  
DRESDEN

Center for Information Services and High Performance Computing (ZIH)

# Trace Visualization with Vampir

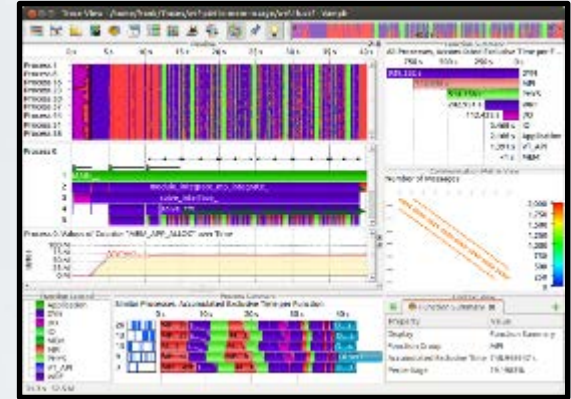


Guido Juckeland ([guido.juckeland@tu-dresden.de](mailto:guido.juckeland@tu-dresden.de))



## Typical questions that Vampir helps to answer:

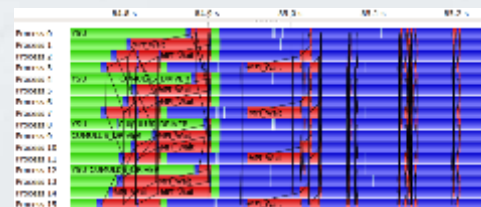
- What happens in my application execution during a given time in a given process or thread?
- How do the communication patterns of my application execute on a real system?
- Are there any imbalances in computation, I/O or memory usage and how do they affect the parallel execution of my application?



- Alternative and supplement to automatic analysis
- Show dynamic run-time behavior graphically at any level of detail
- Provide statistics and performance metrics

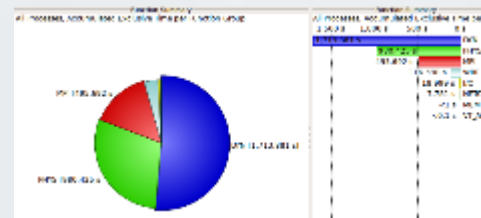
## Timeline charts

- Show application activities and communication along a time axis







## Summary charts





- Provide quantitative results for the currently selected time interval



## Timeline Charts:

-  Master Timeline
-  Process Timeline
-  Counter Data Timeline
-  Performance Radar

## Summary Charts:

-  Function Summary
-  Message Summary
-  Process Summary
-  Communication Matrix View

# Let's Open Your Tracefile



## Start Vampir

```
Terminal File Edit View Search Terminal Help
NoMachine - Connection to ec2-54-67-45-48.us-west-1.compute.amazonaws.com
6:43 PM
ubuntu@ip-172-31-3-169: ~/codes/cuda
ScoreP version 1.4 loaded
SCOREP_ROOT=/opt/scorep
ubuntu@ip-172-31-3-169:~$ cd codes/cuda/
ubuntu@ip-172-31-3-169:~/codes/cuda$ make
scorep --cuda /usr/local/anaconda/bin/mpicxx -Icommon/inc -o simpleMPI_mpi.o
-c simpleMPI.cpp
scorep --cuda "/usr/local/cuda-6.5"/bin/nvcc -ccbin g++ -Icommon/inc -m64 -
gencode arch=compute_30,code=sm_30 -gencode arch=compute_35,code=sm_35 -gencode
arch=compute_37,code=sm_37 -gencode arch=compute_50,code=sm_50 -gencode arch=com
pute_50,code=compute_50 -o simpleMPI.o -c simpleMPI.cu
scorep --cuda /usr/local/anaconda/bin/mpicxx -o simpleMPI simpleMPI_mpi.o sim
pleMPI.o -L"/usr/local/cuda-6.5"/lib64 -lcudart
ubuntu@ip-172-31-3-169:~/codes/cuda$ mpiexec -np 8 ./simpleMPI
Running on 8 nodes
Average of square roots is: 0.667337
PASSED
ubuntu@ip-172-31-3-169:~/codes/cuda$
```



# Let's Open Your Tracefile (2)



## Click on "Open Other"

The screenshot shows a terminal window titled "Vampir" with the following content:

```
ScoreP version 1.4 loaded
SCOREP_ROOT=/opt/scorep
ubuntu@ip-172-31-3-169:~/codes/cuda/
ubuntu@ip-172-31-3-169:~/codes/cuda$ make
scorep --cuda /usr/local/anaconda/bin/mpicxx -Icommon/inc -o simpleMPI_mpi.o
-c simpleMPI.cpp
scorep --cuda "/usr/local/cuda-6.5/bin/linux-gcc46-gcc" -Icommon/inc -m64
gencode arch=compute_35 -code=sm_35 -arch=compute_35 -code=sm_35 -o simpleMPI_mpi.o sim
```

An "Open Recent" dialog box is open, displaying the Vampir 8 logo and a list of recent files:

- /home/ubuntu/NVIDIA\_CUDA...\_0750\_272347783738446/traces.otf2
- /home/ubuntu/NVIDIA\_CUDA...\_0745\_272020827242074/traces.otf2

The "Open Other..." button at the bottom of the dialog is highlighted with a green circle.



# Let's Open Your Tracefile (3)



## Select "Local File"

The screenshot shows a terminal window titled "Vampir" with the following content:

```
ScoreP version 1.4 loaded
SCOREP_ROOT=/opt/scorep
ubuntu@ip-172-31-3-169:~$ cd codes/cuda/
ubuntu@ip-172-31-3-169:~/codes/cuda$ make
scorep --cuda /usr/local/anaconda/bin/mpicxx -Icommon/inc -o simpleMPI_mpi.o
-c simpleMPI.cpp
scorep --cuda "/usr/local/cuda-6.5/bin/x86_64-linux-gnu-gcc" -Icommon/inc -m64 -
gencode arch=compute_35 -code=sm_35
ubuntu@ip-172-31-3-169:~/codes/cuda$
```

An "Open New" dialog box is overlaid on the terminal. It features the Vampir 8 logo and three main options:

- Local File**: Represented by a document icon with a Vampir logo, circled in green.
- Remote File**: Represented by an icon of three monitors.
- Comparison Session**: Represented by an icon of three overlapping documents.

At the bottom of the dialog is a "Cancel" button.

# Let's Open Your Tracefile (4)



Navigate to "home", "ubuntu", "codes", "cuda", "scorep\*", Open "traces.otf2"

Terminal output:

```
ScoreP version 1.4 loaded
SCOREP_ROOT=/opt/scorep
ubuntu@ip-172-31-3-169:~$ cd codes/cuda/
ubuntu@ip-172-31-3-169:~/codes/cuda$ make
scorep --cuda /usr/local/anaconda/bin/mpicxx -Icommon/inc -o simpleMPI_mpi.o
-c simpleMPI.cpp
scorep --cuda "/usr/local/cuda-6.5"/bin/nvcc -c -x cuda -Icommon/inc -m64 -x
gencode -x arch=compute_35 -x code=sm_35 -x device_compute_capabilities=3.5
pi.o sim
```

Open dialog box details:

- Path: /prep-20150305\_1842\_204951628913/
- File selected: traces.otf2
- File type: All trace files (\*.otf, \*.otf2, \*.elg, \*.esd, \*.vcompare)
- Buttons: Open, Open Subset..., Cancel



# Let's Open Your Tracefile (5)



## Maximize the Vampir window

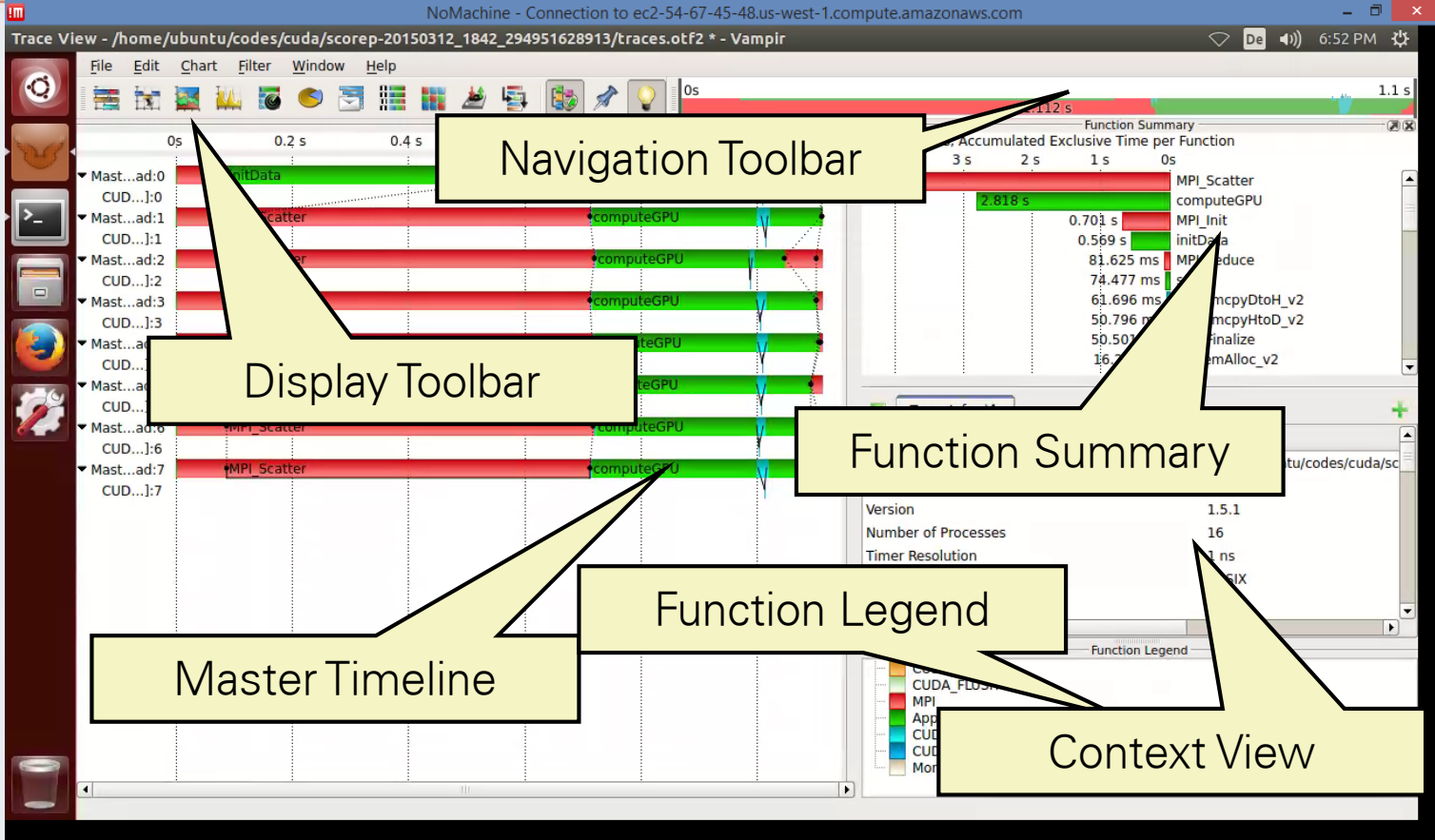
The screenshot shows the Vampir performance analysis tool interface. The main window displays a timeline view of MPI\_Scatter operations across multiple processes (Mast...ad:0 to Mast...ad:6). A 'Trace View' window is overlaid, displaying a function summary table and a function legend.

Function	Time
MPI_Scatter	4.45 s
computeGPU	2.818 s
MPI_Init	0.701 s

Function Legend:

- CUDA\_SYNC
- CUDA\_FLUSH
- MPI
- Application
- CUDA\_API

# What Do You See?



Navigation Toolbar

Display Toolbar

Function Summary

Master Timeline

Function Legend

Context View



- Clicking on anything provides details in the context view
- Zooming is done by click, hold, release
  - Horizontal (Undo: Ctrl+Z, Reset: Ctrl+R)
  - Vertical (Undo: Ctrl+Z, Reset: Ctrl+Shift+R)
- Navigation Toolbar provides ways of sliding and zooming
- Adding more displays via display toolbar
- Moving displays around, dock to any border

**Now you go ahead!**



## Right click on anything

NoMachine - Connection to ec2-54-67-45-48.us-west-1.compute.amazonaws.com

Trace View - /home/ubuntu/codes/cuda/scorep-20150312\_1842\_294951628913/traces.otf2 \* - Vampir

File Edit Chart Filter Window Help

Timeline 0s 0.2 s 0.4 s 0.6 s 0.8 s 1.0 s 1.112 s 1.1 s

Function Summary

All Processes, Accumulated Exclusive Time per Function

Function	Time
MPI_Scatter	4.45 s
computeGPU	2.818 s
MPI_Init	0.703 s
initData	0.569 s
MPI_Reduce	81.625 ms
sum	74.477 ms
cuMemcpyDtoH_v2	63.696 ms
cuMemcpyHtoD_v2	50.796 ms
MPI_Finalize	50.501 ms
cuMemAlloc_v2	16.234 ms

Trace Info

Property	Value
File	/home/ubuntu/codes/cuda/sc
Creator	Score-P 1.4
Version	1.5.1
Number of Processes	16
Timer Resolution	1 ns
File Substrate	POSIX
Trace Compression	NONE

Function Legend

- CUDA\_SYNC
- CUDA\_FLUSH
- MPI
- Application
- CUDA\_API
- CUDA\_KERNEL
- Monitor

Context View

Adjust Process Bar Height to

Group CUDA Streams

Ungroup

Ungroup All

Find... Ctrl+F

Reset Horizontal Zoom Ctrl+R

Reset Vertical Zoom Ctrl+Shift+R

Reset Process Order

Expand All

Collapse All

Show All Group Members

Hide All Group Members

Options

Export Image...

All Processes, Exclusive Time per Function Group

80 %

60 %

40 %

20 %

0 %

MPI

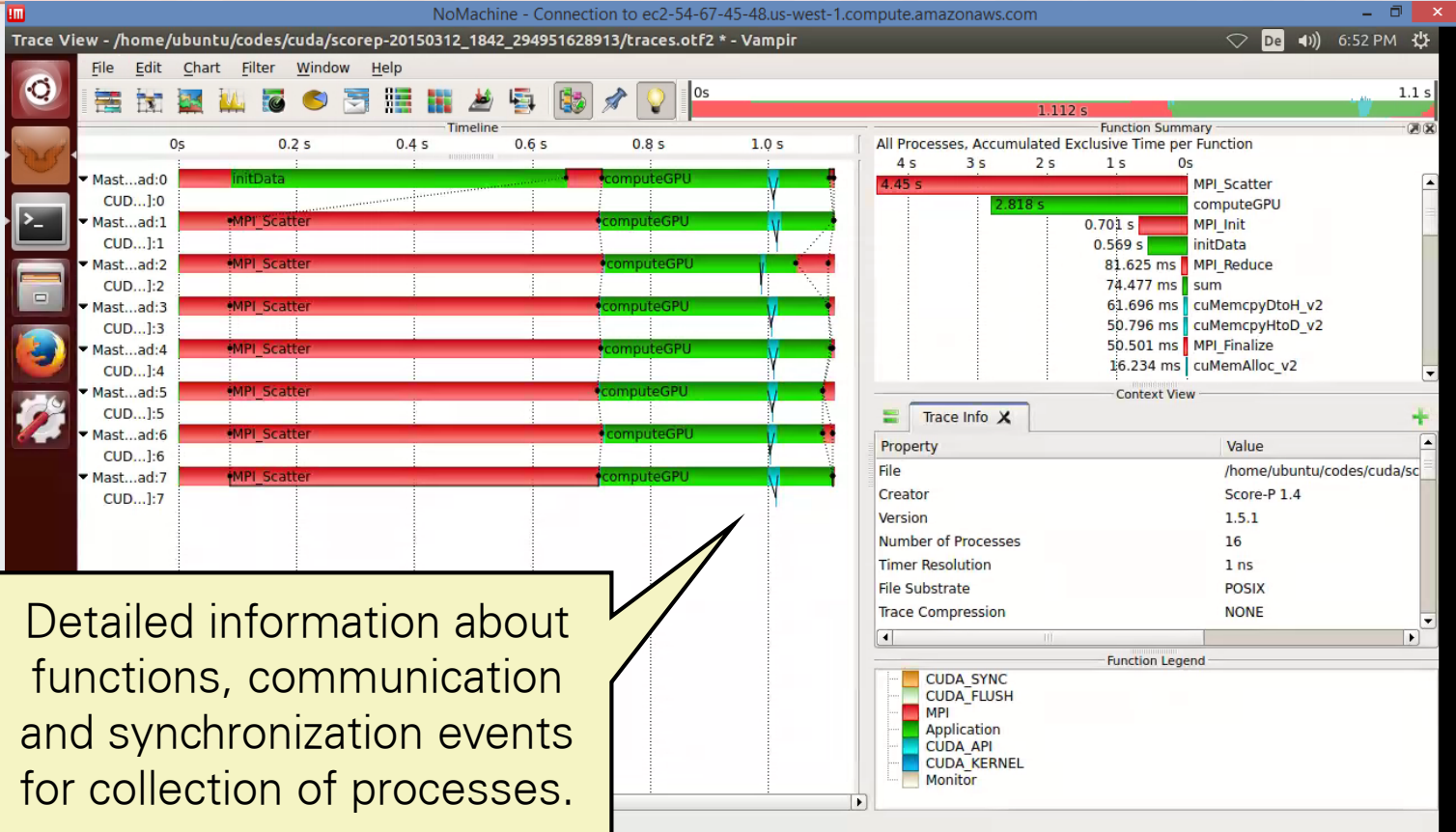
Application





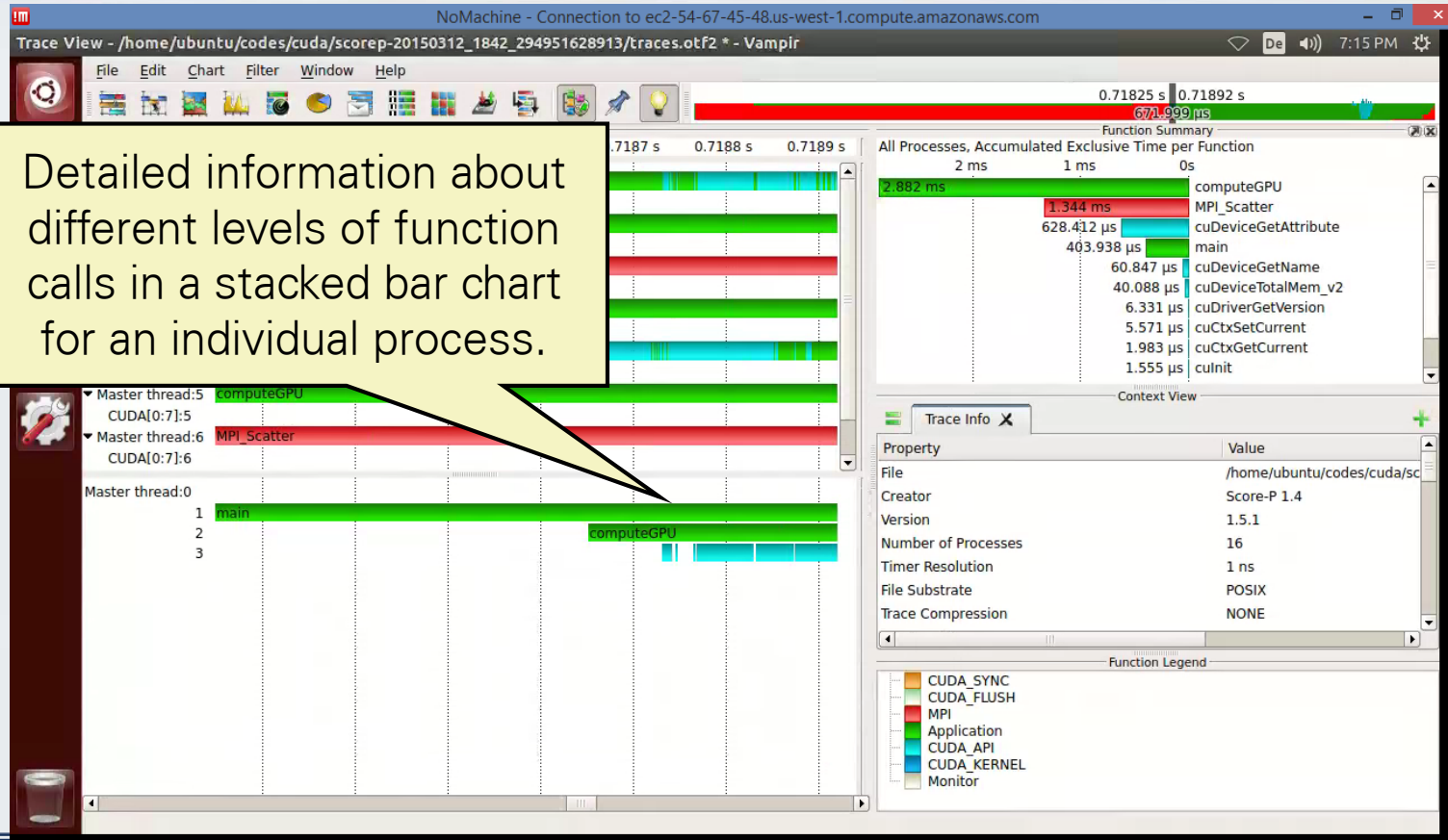
- Right click into Master Timeline
- Adjust Process Bar Height to fit Chart Height
- Determine length of initialization phase
- Determine length of compute phase
- Determine kernel runtime
- Determine message sizes

# Displays: Master Timeline

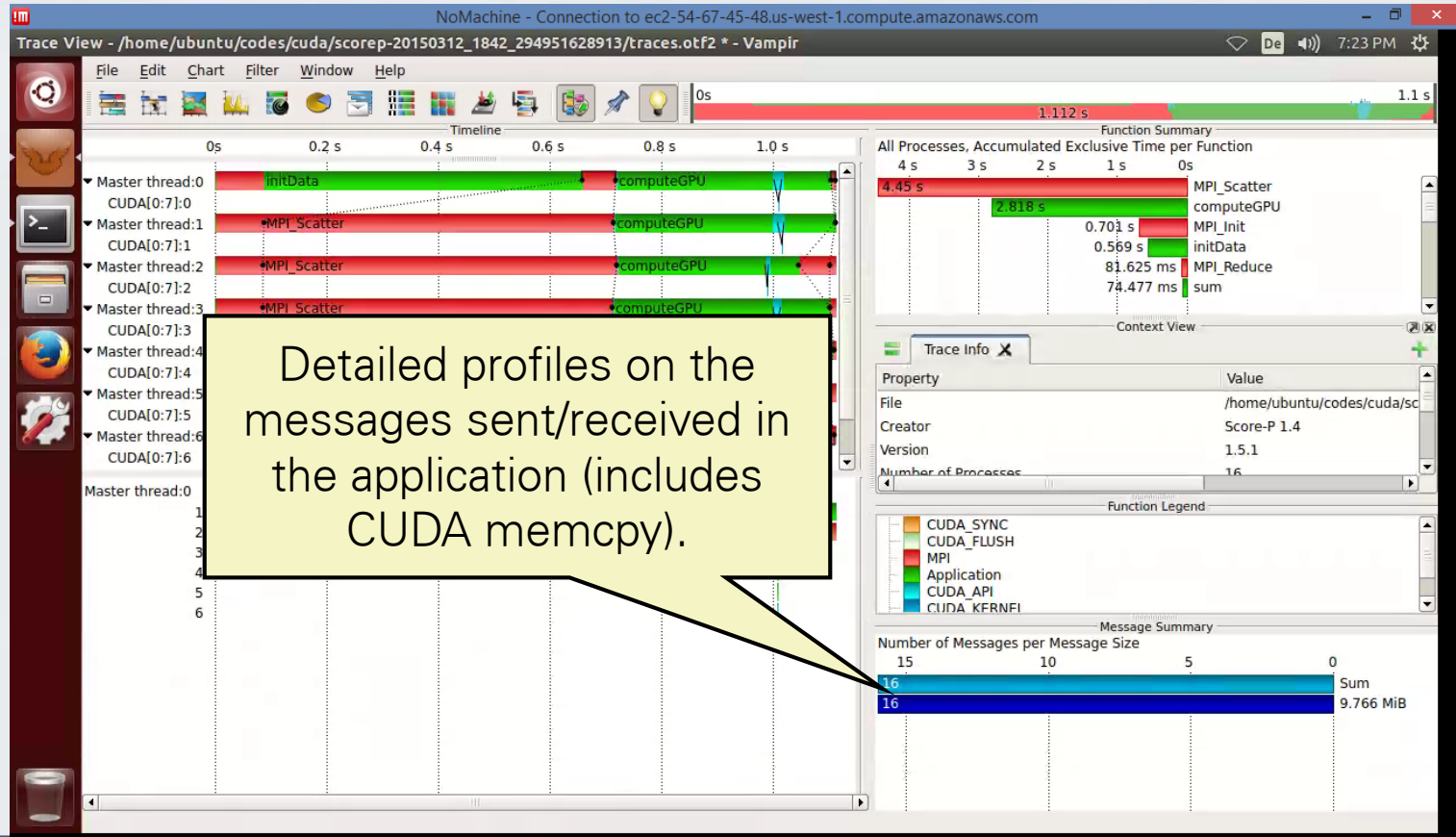


Detailed information about functions, communication and synchronization events for collection of processes.

# Displays: Process Timeline



# Displays: Message Summary





- All displays are updated to the currently zoomed time interval
- Function Summary
  - Include/exclude functions
  - Change metric
  - Select processes used for profile
- Message Summary
  - Change metric
  - Select only specific senders/receivers

## There Is an Example Trace to Play With

---



**Go and look under `/home/ubuntu/traces/cuda` for more traces**

**Now go and play with your or my trace –  
tell me how to improve the application**



**TECHNISCHE  
UNIVERSITÄT  
DRESDEN**

Center for Information Services and High Performance Computing (ZIH)

# A Look Ahead: OpenACC Tracing



**Guido Juckeland ([guido.juckeland@tu-dresden.de](mailto:guido.juckeland@tu-dresden.de))**



Center for Information Services &  
High Performance Computing



- Your are looking at a prototype
- Only works with PGI compilers and developer version of Score-P
- If you find it cool – talk to your OpenACC compiler vendor ☺

# Start a Terminal



NoMachine - Connection to ec2-54-67-45-48.us-west-1.compute.amazonaws.com

Terminal File Edit View Search Terminal Help

ubuntu@ip-172-31-14-255: ~

```
ScoreP version 1.4 loaded
SCOREP_ROOT=/opt/scorep
ubuntu@ip-172-31-14-255:~$
```

The terminal window shows a dark purple background with a lighter purple terminal area. On the left side, there is a vertical dock with several icons. The terminal icon, which is a white square with a black border and a white cursor, is circled in green. Other icons in the dock include a gear, a folder, a globe, and a wrench. The top of the window has a blue title bar with standard window controls and system status icons like Wi-Fi, language, and volume.

# Switch to developer version of Score-P



```
% ubuntu@ip-172-31-3-169:~$ module purge
ScoreP version 1.4 unloaded

% module av
----- /usr/share/modules/versions -----
3.2.10
----- /usr/share/modules/modulefiles ----
dot                modules                scorep/dev-openacc
module-git         null                    use.own
module-info       scorep/1.4(default)
```

```
% module load scorep/dev-openacc

ScoreP version openacc loaded
  SCOREP_ROOT=/opt/scorep-openacc
```



- Go to OpenACC Example

```
% cd codes/openacc
```

- Compile

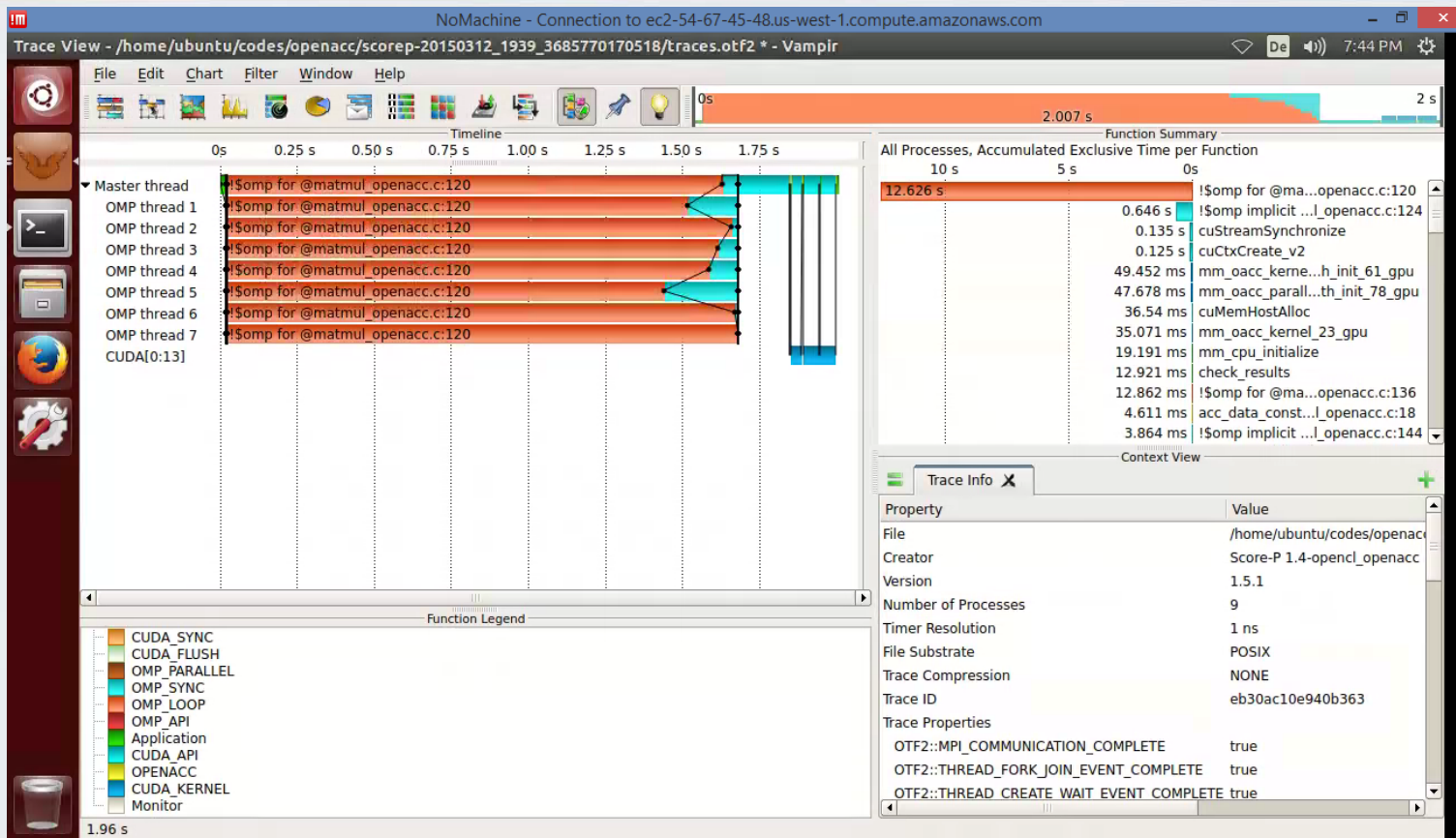
```
% make  
scorep --cuda pgcc -mp -ta=nvidia matmul_openacc.c -o matmul_openacc
```

## Run

```
% export OMP_NUM_THREADS=8
% ./matmul_openacc

CPU MM with 8 threads
MM on CPU: 1.658984 sec
mm_oacc_kernel(): 0.207447 sec
OpenACC matrix multiplication test was successful!
mm_oacc_kernel_with_init(): 0.052948 sec
OpenACC matrix multiplication test was successful!
mm_oacc_parallel_with_init(): 0.051797 sec
OpenACC matrix multiplication test was successful!
Total runtime: 0.325640 sec
```

# Open the New Tracefile



## There Is an Example Trace to Play With

---



**Go and look under `/home/ubuntu/traces/openacc` for more traces**

**Now go and play with your or my trace –  
tell me how to improve the application**



**TECHNISCHE  
UNIVERSITÄT  
DRESDEN**

Center for Information Services and High Performance Computing (ZIH)

# Looking a Little Further

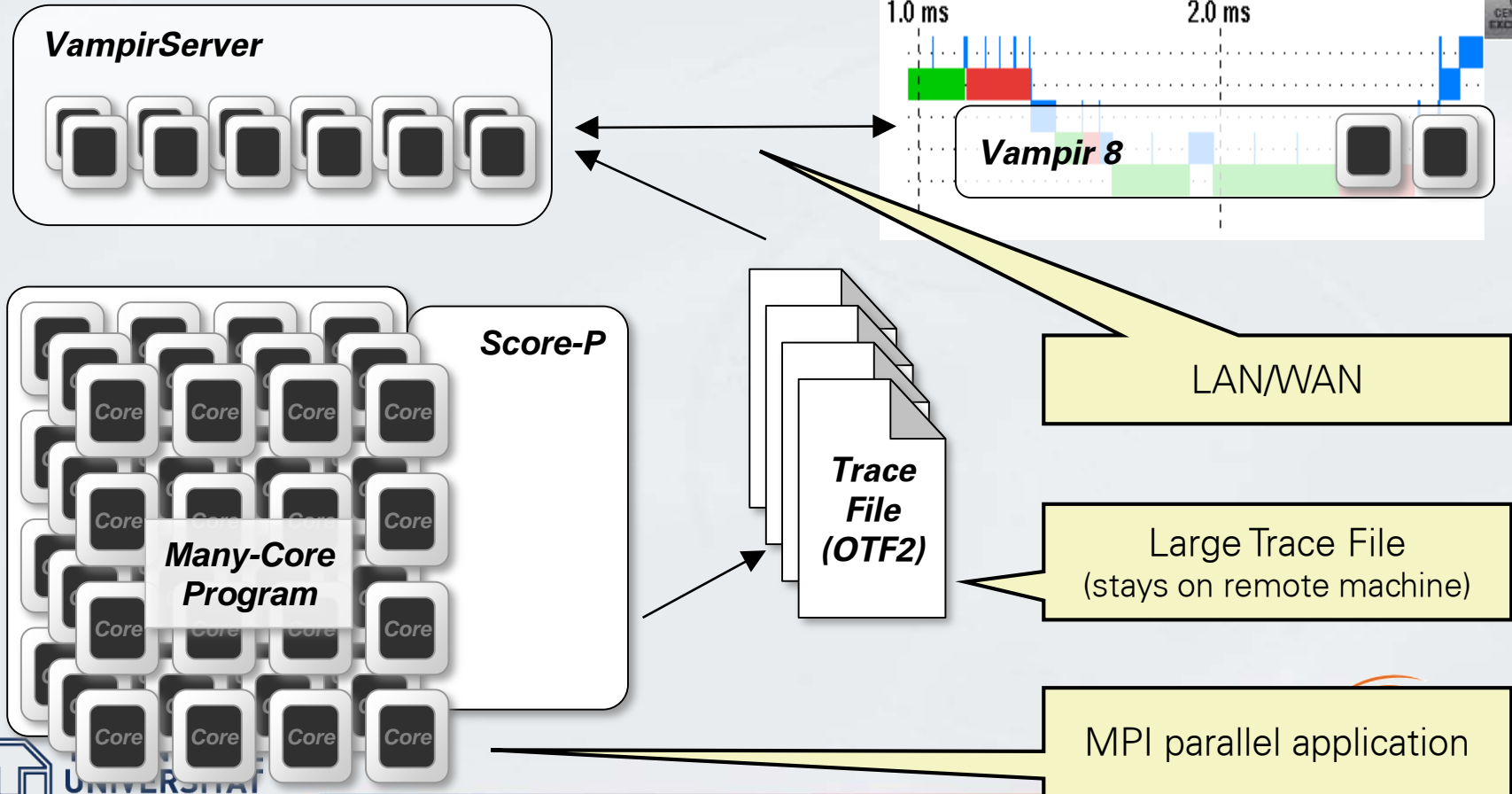


**Guido Juckeland ([guido.juckeland@tu-dresden.de](mailto:guido.juckeland@tu-dresden.de))**



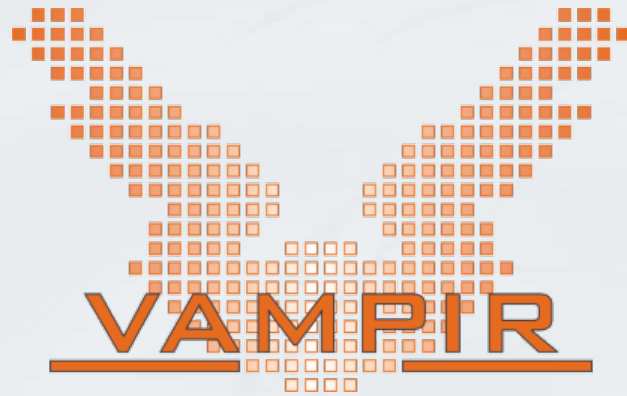
Center for Information Services &  
High Performance Computing

# How About Very Large Tracefiles?





- Performance Analysis is valuable
- Use “easy” tools first
- Score-P can record any concurrent activity
- Vampir can visualize all that activity
  
- The rest is experience and up to you 😊



Vampir is available at <http://www.vampir.eu>,  
get support via [vampirsupport@zih.tu-dresden.de](mailto:vampirsupport@zih.tu-dresden.de)