

# Profiling Tools Training Workshop Issues and Lessons Learned

Summit Profiling Tools Workshop  
Oak Ridge National Laboratory

George S. Markomanolis  
Mike Brim

9 August 2019

ORNL is managed by UT-Battelle, LLC for the US Department of Energy



U.S. DEPARTMENT OF  
**ENERGY**

# Extrac/Paraver Issues & Lessons Learned

ORNL is managed by UT-Battelle, LLC for the US Department of Energy



U.S. DEPARTMENT OF  
**ENERGY**

# Non-Capability Matrix – Extrae

Capability	Profiling	Tracing	Notes/Limitations
MPI, MPI-IO			
OpenMP CPU			
OpenMP GPU	X	X	PGI compiler with OpenMP is not supported. XL compiler is not supported Missing a lot of data regarding GPU
OpenACC	X	X	No support
CUDA	X	X	Missing a lot of data regarding GPU
POSIX I/O			
POSIX threads			
Memory – app-level			
Memory – function-level			
Hotspot Detection			
Variance Detection			
Hardware Counters			

# Issue #1: Merging of the traces cause errors

- Extrae creates traces and then a parallel tool merges them. This tool, which required a separate jsrun, causes errors and segmentation files.
- Solution: Merge in a separate job

## Issue #2: Not enough data to instrument GPU

- Extrae instruments codes with CUDA but not many events are recorded, thus it is not possible to analyze the performance of such applications

# Lessons Learned

- For compilation is required to activate the debug mode for your application.
- Paraver, the visualization tool has a significant learning curve, no guided analysis
- Paraver is one of the best tools to analyze OpenMP tasks.
- With the cut/filter functionalities we can visualize part of a large trace and identify bottlenecks



# Score-P Issues & Lessons Learned

ORNL is managed by UT-Battelle, LLC for the US Department of Energy



U.S. DEPARTMENT OF  
**ENERGY**

# Non-Capability Matrix – Score-P

Capability	Profiling	Tracing	Notes/Limitations
MPI, MPI-IO			
OpenMP CPU			
OpenMP GPU	X	X	instrumented code failed to link
OpenACC			
CUDA	X		Score-P runtime error during profiling
POSIX I/O			
POSIX threads			
Memory – app-level			
Memory – function-level			
Hotspot Detection			
Variance Detection			
Hardware Counters		X	NVIDIA GPU kernel counters (all zeroes)



# Issue #1: Code Instrumentation Problems

- Fortran with OpenMP
  - Problem: runtime error
  - Solution: use '--pdt' option
- CUDA with OpenMP
  - Problem: nvcc compile error
  - Solution: use options '--noopenmp --thread=none'
- OpenMP target
  - Problem: nvcc link error
  - Solution: TBD

```
+ scorep --cuda xlc_r -O3 -qsmp=omp -qoffload -qnostrict -qtgtarch=sm_70 -o nuccor_dgemm nuccor_dgemm.o  
get_wall_time.o  
nvcc fatal : Don't know what to do with 'valloc'  
[Score-P] ERROR: Execution failed: xlc_r nuccor_dgemm.scorep_init.o nuccor_dgemm.opari2_init.o nuccor_dgemm.o  
get_wall_time.o `/sw/summit/scorep/6.0/xl-16.1.1-3/bin/scorep-config --thread=omp --mpp=n  
one --io=none --mutex=none --noonline-access --preprocess --noopencl --noopenacc --memory=libc --constructor`  
`/sw/summit/scorep/6.0/xl-16.1.1-3/bin/scorep-config --thread=omp --mpp=none --io=none --  
mutex=none --noonline-access --preprocess --noopencl --noopenacc --memory=libc --ldflags` -O3 -qsmp=omp -qoffload  
-qnostrict -qtgtarch=sm_70 -Wl,-start-group `/sw/summit/scorep/6.0/xl-16.1.1-3/bin/s  
corep-config --thread=omp --mpp=none --io=none --mutex=none --noonline-access --preprocess --noopencl --noopenacc  
--memory=libc --event-libs` -Wl,-end-group `/sw/summit/scorep/6.0/xl-16.1.1-3/bin/s  
corep-config --thread=omp --mpp=none --io=none --mutex=none --noonline-access --preprocess --noopencl --noopenacc  
--memory=libc --mgmt-libs` -o nuccor_dgemm
```

## Issue #2: Inability to filter certain APIs

- MPI, OpenACC, etc. cannot be excluded
  - believed to be an issue with mechanism for intercepting APIs
- May limit user ability to produce reasonable trace sizes

```
> cat all.filter
SCOREP_REGION_NAMES_BEGIN
EXCLUDE *
SCOREP_REGION_NAMES_END
```

Request for all regions to be excluded

```
> scorep-score -f all.filter profile.cubex
```

Estimated aggregate size of event trace: 559MB  
Estimated requirements for largest trace buffer (max\_buf): 50MB  
Estimated memory requirements (SCOREP\_TOTAL\_MEMORY): 52MB  
(hint: When tracing set SCOREP\_TOTAL\_MEMORY=52MB to avoid intermediate flushes or reduce requirements using USR regions filters.)

flt	type	max_buf[B]	visits	time[s]	time[%]	time/visit[us]	region
-	ALL	56,251,529	21,259,014	443.14	100.0	20.84	ALL
-	OPENACC	40,727,960	17,312,520	79.20	17.9	4.57	OPENACC
-	MPI	11,311,476	2,002,458	333.18	75.2	166.38	MPI
-	COM	4,212,052	1,944,024	30.76	6.9	15.82	COM
-	SCOREP	41	12	0.00	0.0	87.02	SCOREP
*	ALL	52,039,477	19,314,990	412.38	93.1	21.35	ALL-FLT
-	OPENACC	40,727,960	17,312,520	79.20	17.9	4.57	OPENACC-FLT
-	MPI	11,311,476	2,002,458	333.18	75.2	166.38	MPI-FLT
+	FLT	4,212,052	1,944,024	30.76	6.9	15.82	FLT
-	SCOREP	41	12	0.00	0.0	87.02	SCOREP-FLT

e.g., only COM regions end up filtered

# Issue #3: CUDA Profiling

- Problem: Enabling CUDA in profiling mode resulted in LSMS abort
- Solution: TBD

## CUDA Profiling Error (Region Exit Mismatch)

```
> cat stderr.txt
[Score-P] src/measurement/profiling/scorep_profile_event_base.c:188: Error: Inconsistent profile. Stop profiling:
Exit event for other than current region occurred at location 6: Expected exit for region 'cudaLaunchKernel'. Exited
region 'cudaLaunchKernel'
[Score-P] src/measurement/profiling/scorep_profile_debug.c:223: Fatal: Cannot continue profiling. Activating core
files (export SCOREP_PROFILING_ENABLE_CORE_FILES=1) might provide more insight.
[Score-P] Please report this to support@score-p.org. Thank you.
[Score-P] Try also to preserve any generated core dumps.
[a28n16:24662] *** Process received signal ***
[a28n16:24662] Signal: Aborted (6)
[a28n16:24662] Signal code: (-6)
[a28n16:24662] [ 0] [0x2000000504d8]
[a28n16:24662] [ 1] /lib64/libc.so.6(gsignal+0x60)[0x20000a42fbf0]
[a28n16:24662] [ 2] /lib64/libc.so.6(abort+0x18c)[0x20000a431f6c]
[a28n16:24662] [ 3]
/sw/summit/scorep/6.0/gcc-6.4.0/lib/libscorep_measurement.so.0(SCOREP_UTILS_Error_Abort+0x34)[0x20000803db54]
[a28n16:24662] [ 4]
/sw/summit/scorep/6.0/gcc-6.4.0/lib/libscorep_measurement.so.0(scorep_profile_on_error+0x284)[0x2000080151d4]
[a28n16:24662] [ 5]
/sw/summit/scorep/6.0/gcc-6.4.0/lib/libscorep_measurement.so.0(scorep_profile_exit+0x1e4)[0x200008013784]
[a28n16:24662] [ 6]
/sw/summit/scorep/6.0/gcc-6.4.0/lib/libscorep_measurement.so.0(SCOREP_Profile_Exit+0xf0)[0x20000800a530]
[a28n16:24662] [ 7] /sw/summit/scorep/6.0/gcc-6.4.0/lib/libscorep_measurement.so.0(+0x7ae04)[0x20000800ae04]
[a28n16:24662] [ 8]
/sw/summit/scorep/6.0/gcc-6.4.0/lib/libscorep_measurement.so.0(SCOREP_Location_ExitRegion+0xd0)[0x200007fe20d0]
[a28n16:24662] [ 9] /sw/summit/scorep/6.0/gcc-6.4.0/lib/libscorep_adapter_cuda_mgmt.so.0(+0xc9dc)[0x20000855c9dc]
[a28n16:24662] [10] /sw/summit/cuda/10.1.105/extras/CUPTI/lib64/libcupti.so.10.1(+0xef198)[0x20000989f198]
```

# Issue #4: Sampling Mode with Unwinding

- Problem: Enabling sampling mode causes MiniWeather segfaults

- for both C and Fortran

- Solution: TBD

```
export SCOREP_ENABLE_TRACING=yes
export SCOREP_ENABLE_UNWINDING=true
export SCOREP_SAMPLING_EVENTS=perf_cycles@2000000
```

```
> gdb ./bin/miniweather_mpi_omp core.xxxx
```

```
Core was generated by `miniWeather_mpi_omp '.
Program terminated with signal 11, Segmentation fault.
```

```
#0 _ULppc64_step (cursor=0x200018d098b0) at ppc64/Gstep.c:457
```

```
457 ppc64/Gstep.c: No such file or directory.
```

```
Missing separate debuginfos, use: debuginfo-install glibc-2.17-260.el7_6.6.ppc64le libatomic-4.8.5-36.el7_6.2.ppc64le
libgcc-4.8.5-36.el7_6.2.ppc64le libibverbs-41mlnx1-OFED.4.5.0.1.0.45229.ppc64le
libmlx4-41mlnx1-OFED.4.5.0.0.3.45229.ppc64le libmlx5-41mlnx1-OFED.4.5.0.3.8.45229.ppc64le libnl3-3.2.28-4.el7.ppc64le
librx-41mlnx1-OFED.4.4.2.4.6.45229.ppc64le libstdc++-4.8.5-36.el7_6.2.ppc64le numactl-libs-2.0.9-7.el7.ppc64le
xz-libs-5.2.2-1.el7.ppc64le zlib-1.2.7-18.el7.ppc64le
```

```
(gdb) bt
```

```
#0 _ULppc64_step (cursor=0x200018d098b0) at ppc64/Gstep.c:457
```

```
#1 0x00002000007a3e7c in slow_backtrace (uc=0x200018d0aa30, size=32, buffer=0x200018d0b070) at mi/backtrace.c:45
```

```
#2 unw_backtrace (buffer=0x200018d0b070, size=<optimized out>) at mi/backtrace.c:72
```

```
#3 0x0000200002238f0c in opal_backtrace_print () from /opt/ibm/spectrum_mpi/jsm_pmix/./lib/libopen-pal.so.3
```

```
#4 0x000020000223222c in show_stackframe () from /opt/ibm/spectrum_mpi/jsm_pmix/./lib/libopen-pal.so.3
```

```
#5 <signal handler called>
```

```
#6 _ULppc64_step (cursor=0x2000023b46e8) at ppc64/Gstep.c:457
```

```
#7 0x00002000003bc4c8 in get_current_stack (unwindData=0x2000023b40f8) at
../src/services/unwinding/scorep_unwinding_cpu.c:438
```

```
#8 0x00002000003bd218 in scorep_unwinding_cpu_handle_enter (unwindData=0x2000023b40f8, contextPtr=0x200018d0d130,
instrumentedRegionHandle=0, callingContext=0x200018d0d01c,
```

```
unwindDistance=0x200018d0d014, previousCallingContext=0x200018d0d018) at
../src/services/unwinding/scorep_unwinding_cpu.c:802
```

## Issue #5: OpenACC hides CUDA

- Problem: Despite ' - - cuda ', OpenACC profiles/traces show no CUDA-level interactions
- Solution: TBD

## Issue #6: Large Trace Analysis

- Problem: VampirServer takes **way too long** to load large traces
  - investigate how to make it use more processes
  - check that servers are being effectively placed across Summit node cores
- Solution: TBD



# Lessons Learned

- General Performance Analysis Methodology
  - Starting with existing application test cases covering different scales is essential
- Profiling mode is relatively lightweight
  - and very useful for pre-trace filtering
- Filtering
  - is required before tracing C++
  - is highly recommended for focused tracing of large-scale runs
- Using Vampir and Cube on local desktop is easy and useful
  - best when combined with remote file access (e.g., sshfs), since you don't need to move files between systems

# Scalasca Issues & Lessons Learned

ORNL is managed by UT-Battelle, LLC for the US Department of Energy



U.S. DEPARTMENT OF  
**ENERGY**

# Non-Capability Matrix – Scalasca

Capability	Profiling	Tracing	Notes/Limitations
MPI, MPI-IO			
OpenMP CPU			
OpenMP GPU	X	X	No information displayed for the GPU
OpenACC	X	X	
CUDA	X	X	No information displayed
POSIX I/O			
POSIX threads			
Memory – app-level			
Memory – function-level	X	X	
Hotspot Detection			
Variance Detection			
Hardware Counters		X	NVIDIA GPU kernel counters (all zeroes)

# Issue #1: Installation

- Scalasca requires Score-P:
  - The available Score-P version, was not supporting PGI compiler on Power processor
  - Score-P team provided a special version
  - Now the Score-P v6.0 supports PGI on Power processor
  - Too many dependencies, used Spack to avoid a lot of compilations

## Issue #2: Stability

- Scalasca crashes when we try to open the documentation (it could be related to our system-java).
- Solution: TBD

# Lessons Learned

- Pattern Analysis
  - The sophisticated and automatic pattern analysis provides the essential information to understand better the insight of an application
- For instrumentation, it is used the Score-P tool with all its benefits
- It is not complicated to apply the pattern analysis on a large execution
- Cube is an easy and useful interface where with just a glimpse we can observe many different performance aspects
- The compilation of an application with Score-P sometimes is not straightforward



# TAU Issues & Lessons Learned

ORNL is managed by UT-Battelle, LLC for the US Department of Energy



U.S. DEPARTMENT OF  
**ENERGY**

# Non-Capability Matrix – TAU

Capability	Profiling	Tracing	Notes/Limitations
MPI, MPI-IO			
OpenMP CPU			
OpenMP GPU			
OpenACC		X	No CUPTI metrics
CUDA		X	There was issue with CUPTI metrics
POSIX I/O			
POSIX threads			
Memory – app-level			
Memory – function-level			
Hotspot Detection			
Variance Detection			
Hardware Counters		X	Issues about using CUPTI metrics

# Issue #1: Installation

- Required support from the TAU team regarding some MPI options for the configuration on Spectrum MPI
- An error about -hwloc appeared and the TAU team fixed the bug

## Issue #2: tau\_exec was not working

- We were getting strange errors with tau\_exec:

ERROR: ld.so: object '/autofs/nccs-svm1\_sw/summit/.swci/1-com' from LD\_PRELOAD cannot be preloaded: ignored.

ERROR: ld.so: object 'ute/o' from LD\_PRELOAD cannot be preloaded: ignored.

ERROR: ld.so: object 't/s' from LD\_PRELOAD cannot be preloaded: ignored.

ERROR: ld.so: object 'ack/20180914/linux-rhel7-' from LD\_PRELOAD cannot be preloaded: ignored.

- Solution:

We found in tau\_exec script a command that was replacing some characters on the LD\_PRELOAD which was not working on our system, we commented this command and the problem solved

# Issue #3: Instrumentation with CUDA

- Problem:

Error: **nvcc fatal : A single input file is required for a non-link phase when an outputfile is specified**

- Solution:

Using a TAU\_MAKEFILE with the PAPI enabled, causes issues as we add another library to be linked in a non-link phase. By creating a TAU\_MAKEFILE without PAPI, solved this issue.

# Issue #4: CUPTI error

- **Problem1:**

TAU: CUPTI error in cuptiActivityEnable (CUPTI\_ACTIVITY\_KIND\_ENVIRONMENT):  
CUPTI\_ERROR\_NOT\_COMPATIBLE

- **Solution1:**

TAU team could reproduce this issue and provided a fix.

- **Problem2:**

After the above fix, I had this error:

TAU: Error: Unknown metric: CUDA.Tesla\_V100-SXM2-  
16GB.domain\_d.active\_warps

TAU: Error: Unknown metric: CUDA.Tesla\_V100-SXM2-  
16GB.domain\_d.active\_cycles

- **Solution2:**

Another bug fixed



# Issue #5: Memory Error

- Problem:

We got the error: “Tau\_MemMgr\_malloc: MMAP MAX MEMBLOCKS REACHED!” during the instrumentation of an application.

- Solution:

We should configure TAU with -DISABLE\_MEMORY\_MANAGER. TAU team created a patch to fix this issue for the newer version.

# Issue #6: File formats for visualization

- The tau2otf tool supports only MPI and OpenSHMEM
- The tau2slog tool worked but jumpshot crashed because of an error

# Issue #7: Instrumentation of MPI+OpenMP+CUDA

- Problem:

Error: Only counters for a single GPU device model can be collected at the same time.

- Solution: TBD

# Lessons Learned

- TAU is a instrumentation tool that tries to cover many different topics with low overhead for profiling
- Metrics for OpenACC instrumentation will be available until SC19
- It is required to use the tau\_exec instrumentation approach for GPUs, although this probably will change soon
- TAU provides tools such as PerfExplorer to study the scalability of many experiments
- It is possible to instrument dynamic/static phases through the Program Database Toolkit
- User used the tau\_exec to instrument an ocean model on the first day of hands-on sessions

# Capability Matrix - Final

	Extræ	Score-P	Scalasca	TAU
Capability	Profiling			
MPI, MPI-IO	✓	✓	✓	✓
OpenMP CPU	✓	✓	✓	✓
OpenMP GPU	✗	✗	✗	✓
OpenACC	✗	✓	✗	✓
CUDA	✗	✗	✗	✓
POSIX I/O	✓	✓	✓	✓
POSIX threads	✓	✓	✓	✓
Memory – app-level	✓	✓	✓	✓
Memory – function-level				✓
Hotspot Detection	✓	✓	✓	✓
Variance Detection	✓		✓	✓
Hardware Counters	✓	✓	✓	✗

# Capability Matrix - Final

	Extrac	Score-P	Scalasca	TAU
Capability	Tracing			
MPI, MPI-IO	✓	✓	✓	✓
OpenMP CPU	✓	✓	✓	✓
OpenMP GPU	✗	✗	✗	✗
OpenACC	✗	✓	✗	✗
CUDA	✗	✗	✗	✗
POSIX I/O	✓	✓	✓	✓
POSIX threads	✓	✓	✓	✓
Memory – app-level	✓			✓
Memory – function-level				✓
Hotspot Detection	✓	✓	✓	✓
Variance Detection	✓		✓	✓
Hardware Counters	✓	✗	✗	✗



Thank you!  
Questions?