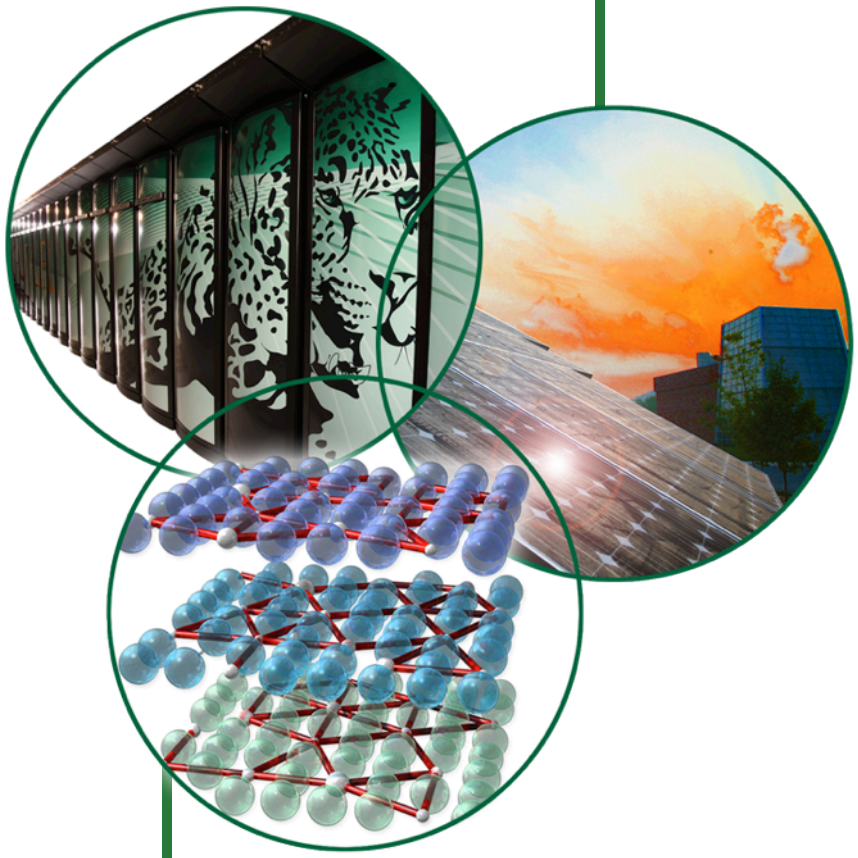
Performance Analysis with Vampir



17 July 2013

Joseph Schuchart

Vampir On-Site Support

U.S. DEPARTMENT OF ENERGY

OAK RIDGE NATIONAL LABORATORY

MANAGED BY UT-BATTELLE FOR THE DEPARTMENT OF ENERGY

The Vampir Toolsuite

- Scalable Trace-based performance analysis tools
- Detailed insight into parallel applications
- Two components:
  - **VampirTrace**: Instrumentation and Measurement
  - **Vampir**: Visual Analysis of trace data

CSMD

OAK RIDGE National Laboratory

VampirTrace

- **Instrumentation**
  - Insert **function-level instrumentation** (Compiler, TAU, Dyninst, Manual)
  - Provide **instrumented libraries**:
    - CUDA / CUPTI
    - MPI
    - libc
  - Instrument OpenMP (Opari)
- **Measurement**:
  - Record generated events
  - Manage event buffers
  - Take care of post-processing
  - Flexible configurable through environment variables
  - Profiling and tracing

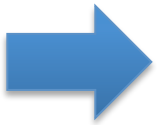
VampirTrace: Workflow

1. Instrument your code with VampirTrace
   1. Load the VampirTrace module

```
module load vampirtrace
OR
module load vampirtrace/5.14.3-chester
```

   2. Use compiler wrappers:

```
CC   = cc
CXX  = CC
F90  = ftn
NVCC = nvcc
```



```
CC   = vtcc
CXX  = vtcxx
F90  = vtf90
NVCC = vtnvcc
```

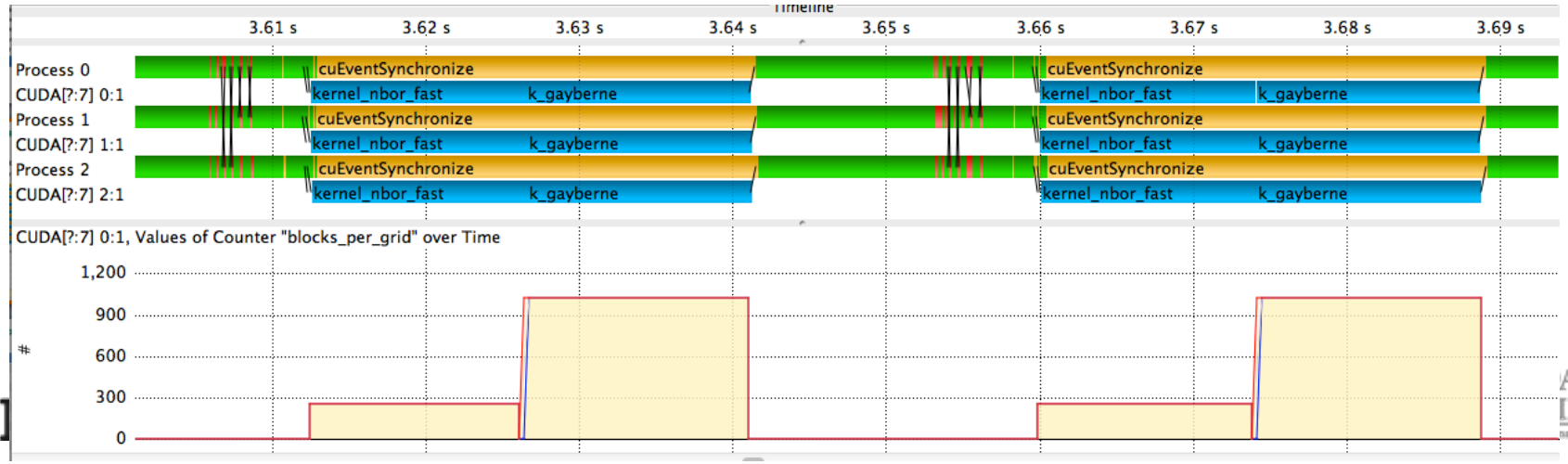
   3. Recompile/Relink
2. Run your code with an appropriate test set
3. Analyze the data using Vampir

VampirTrace: Environment Variables

Environment Variable	Description	Default
VT_BUFFER_SIZE	Size of event buffer	32M
VT_MAX_FLUSHES	Maximum number of flushes	1
VT_PFORM_GDIR	Directory to hold trace data	./
VT_METRICS	Colon-separated list of PAPI metrics to record	-
VT_FILTER_SPEC	Define a filter file for functions	-
VT_IOTRACE	Enable I/O tracing	no
VT_MEMTRACE	Enable memory tracing	no
VT_MODE	Colon-separated list of VampirTrace modes: Tracing (TRACE) or Profiling (STAT)	TRACE
VT_VERBOSE	Controls verbosity: Quiet (0), Warnings (1), Information (2)	0

VampirTrace: GPU Support

- VampirTrace supports GPU applications
  - Uses **NVIDIA CUPTI: CU**DA **P**erformance **T**ools **I**nterface
  - Low overhead for the GPU parts
  - Records GPU activities
    - Data movement (Host <-> Device)
    - Kernel runtimes
    - CUDA function calls
  - Can be used for CUDA and OpenACC applications
    - Waiting for the OpenACC tools API
  - Has to be explicitely enabled using VT_GPUTRACE



VampirTrace: VT_GPUTRACE

Option	Description
`cuda`	Enable support for CUDA
`runtime`	CUDA runtime API tracing
`driver`	**CUDA driver API tracing**
`kernel`	Record kernel activity
`idle`	Record GPU idle times
`memcpy`	Record GPU memory copies
`stream_reuse`	force reusing of CUDA streams after `cudaStreamDestroy()`
`memusage`	Record GPU memory usage
`yes|default`	Same as "`cuda,runtime,kernel,memcpy`"
`no`	Disable CUDA measurement

VampirTrace: Additional Variables

Variable	Description
`VT_GPUTRACE_KERNEL=[yes|2]`	`yes`: record kernel activity (same as `kernel`) `2`: record additional kernel counter
`VT_CUDATRACE_SYNC=[0|1|2|3]`	Controls how VampirTrace handles synchronization (CUDA runtime wrapper only)
`VT_CUPTI_METRICS`	Record additional CUPTI metrics, e.g., `VT_CUPTI_METRICS=local_store:local_load`
`VT_CUPTI_SAMPLING=[yes|no]`	Poll for CUPTI counter values during kernel execution
`VT_GPUTRACE_MEMUSAGE=[yes|2]`	`yes`: Add counter `gpu_mem_usage` (same as `memusage`) `2`: Also print missing `cudaFree()` calls to `stderr`

- **For CUPTI counter overview:**
  - http://docs.nvidia.com/cuda/cupti/index.html#r_metric_api

VampirTrace: OpenACC

- Most compilers use the CUDA driver API:
  - Set `VT_GPUTRACE=cuda,cupti,runtime,driver,kernel,memcpy,memusage,idle`
- The default CUPTI buffer size is too small (64k)
  - Set `VT_CUDATRACE_BUFFER_SIZE=64M`

Vampir: Example

1. Load Vampir module:

```
module load vampir
OR
module load vampir/8.1.0-130626-chester
```

2. Launch VampirServer

```
vampirserver start -a <projectID> -n <N>
```

3. Setup SSH tunnel as given by Vampir

```
ssh -L 30001:nid00809:30066 \
<user>@titan-internal.ccs.ornl.gov
```

4. Connect your local Vampir installation

Note: X-Forwarding possible but slow!

Vampir: Hints

- Number of server processes:
  - Usually memory bound
    - Depends on trace size
    - About 4x deflate
- Swap PrgEnv before loading the Vampir module
  - PGI-built module has problems with MPI
- Consider using Lens for analysis

How to get Vampir

- ORNL has a site-wide license
  - You can download and install Vampir on your local machine
  - Activate it at www.vampir.eu/activation
- Remote-only version available if you're not on-site
- See: https://www.olcf.ornl.gov/kb_articles/software-vampir/?softwaretype=kb_software_debugging_and_profiling

```
$ scp <user>@home.ccs.ornl.gov:/sw/sources/vampir/client/vampir-*.dmg .
$ scp <user>@home.ccs.ornl.gov:/sw/sources/vampir/client/vampir.license .
# Proceed with dmg installation by opening it and dragging the Vampir
  icon to your preferred launch spot. Start Vampir using this icon.
```

VampirTrace: Hints

- Traces can become huge:
  - Esp. C++ problematic
  - Solution: Filter events
  - Runtime filter: reduce trace size but do not reduce overhead
  - TAU: Compile time filter
  - GCC (just in case ;)) instrumentation filter

```
1 MPI_*  -- -1
2 std::* -- 0
3 *      -- 1000
```

```
BEGIN_EXCLUDE_LIST
init_*
boost::*
END_EXCLUDE_LIST

BEGIN_FILE_EXCLUDE_LIST
/usr/include/*
END_FILE_EXCLUDE_LIST
```

```
-finstrument-functions-exclude-file-list=include/g++,math_extra
-finstrument-functions-exclude-function-list=map,timing,Timer,operator
```

VampirTrace: Hints

- Use profiles to determine heavily called functions
  - Set `VT_MODE=STAT`
- The default buffer size is quite small:
  - Consider setting `VT_BUFFER_SIZE=512M`
  - Allocates 10% of that per thread
    - Override with `VT_THREAD_BUFFER_SIZE`
  - Also true for CUPTI events:
    - Override with `VT_CUDATRACE_BUFFER_SIZE`
- The compiler wrapper doesn't recognize the parallelization paradigm:
  - Use `-vt:mpi`, `-vt:mt`, or `-vt:hyb`

VampirTrace: Hints

- Take caution with large core counts:
  - Produces a large number of files (two per process/thread)
  - Can cause problems with Lustre
- VT comes with an I/O forwarding solution
  - Allocate additional nodes for I/O server
  - Launch server in your job script
  - Trace data will be written through the server
  - Only two files per server

```
module load vampirtrace

source vtiofsl-start.sh -n M

aprun -n N a.out

source vtiofsl-stop.sh
```

Vampir: Live demo